Advances in Computer Vision and Pattern Recognition

Rui Fan
Sicen Guo
Mohammud Junaid Bocus *Editors*

Autonomous Driving Perception

Fundamentals and Applications

Springer

Advances in Computer Vision and Pattern Recognition

Titles in this series now included in the Thomson Reuters Book Citation Index!

Advances in Computer Vision and Pattern Recognition is a series of books which brings together current developments in this multi-disciplinary area. It covers both theoretical and applied aspects of computer vision, and provides texts for students and senior researchers in topics including, but not limited to:

- Deep learning for vision applications
- Computational photography
- Biological vision
- Image and video processing
- Document analysis and character recognition
- Biometrics
- Multimedia
- Virtual and augmented reality
- Vision for graphics
- Vision and language
- Robotics

Rui Fan · Sicen Guo · Mohammud Junaid Bocus
Editors

Autonomous Driving Perception

Fundamentals and Applications

Springer

Editors

Rui Fan
Control Science & Engineering, Shanghai
Research Institute for Intelligent
Autonomous Systems
Tongji University
Shanghai, P. R. China

Sicen Guo
Control Science & Engineering, Shanghai
Research Institute for Intelligent
Autonomous Systems
Tongji University
Shanghai, P. R. China

Mohammud Junaid Bocus
Electrical and Electronic Engineering
University of Bristol
Bristol, UK

ISSN 2191-6586 ISSN 2191-6594 (electronic)
Advances in Computer Vision and Pattern Recognition
ISBN 978-981-99-4286-2 ISBN 978-981-99-4287-9 (eBook)
https://doi.org/10.1007/978-981-99-4287-9

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Paper in this product is recyclable.

Preface

With the recent advancements in artificial intelligence, there is a growing expectation that fully autonomous driving vehicles will soon become a reality, leading to significant societal changes. The core competencies of an autonomous vehicle system can be broadly categorized into four main categories: perception, prediction, planning, and control. The environmental perception system serves as the foundation of autonomous vehicles, utilizing cutting-edge computer vision and machine learning algorithms to analyze raw sensor data and create a comprehensive understanding of the surrounding environment. Similar to the visual cognition and understanding of humans, this process allows for a deep and nuanced perception of the world.

Conventional autonomous driving perception systems are often hindered by separate sensing, memory, and processing architectures, which may not meet the demand for ultra-high raw sensor data processing rates and ultra-low power consumption. In contrast, in-sensor computing technology performs signal processing at the pixel level by utilizing the collected analog signals directly, without requiring data to be sent to other processors. This enables highly efficient and low-power consumption visual signal processing by integrating sensing, storage, and computation onto focal planes with innovative circuit designs or new materials. Therefore, the in-sensor computing paradigm holds significant potential for autonomous driving. Furthermore, fish-eye cameras have emerged as an essential sensor in the field of autonomous driving. Thanks to the unique projection principle of fish-eye cameras, they offer a significantly larger field of view (FoV) compared to conventional cameras. This distinct characteristic makes fish-eye cameras highly versatile and suitable for a wide range of autonomous driving perception applications. In addition, computer stereo vision is a cost-effective and efficient method for depth information acquisition, and it has found widespread use in 3D environmental perception. Despite the impressive results obtained by state-of-the-art (SoTA) stereo vision algorithms that utilize convolutional neural networks, their training typically necessitates a substantial amount of accurately labeled disparity ground truth data. Consequently, self-supervised or unsupervised deep stereo networks have emerged as the dominant approach in this research area.

Research on semantic segmentation has been ongoing for over a decade. However, conventional single-modal networks are unable to fully utilize the spatial information provided by range sensors, making them less effective in diverse weather and illumination conditions. To address this challenge, data-fusion semantic segmentation networks have been developed, which employ multiple encoders to extract deep features from different visual information sources. These deep features are subsequently fused to provide a more comprehensive understanding of the surrounding environment. 3D object detection is also a crucial component of autonomous driving systems that has made remarkable progress in recent years. Nonetheless, the various perceptual sensors used for object detection present their unique challenges. Cameras are vulnerable to issues such as foreshortening and flickering effects, over-exposure problems, as well as environmental variations like lighting and weather conditions. Similarly, LiDARs and RADARs suffer from low-resolution and sparse data representations. Furthermore, occlusion presents a significant challenge to object detection, leading to the partial or complete invisibility of obstructed objects. To address these challenges, collaborative 3D object detection has been proposed as an alternative to conventional approaches. Collaborative object detection facilitates information sharing between agents, enabling them to perceive the environments beyond line-of-sight and FoV. This approach holds great promise in overcoming the limitations of individual sensors and achieving more robust and accurate 3D object detection in autonomous driving systems.

The application of the simultaneous localization and mapping (SLAM) technique to autonomous driving also presents several challenges. Over the past three decades, researchers have made significant progress in addressing the probabilistic SLAM problem by developing a range of theoretical frameworks, efficient solvers, and complete systems. Visual SLAM for texture-less environments is an especially challenging task, as multi-view images cannot be effectively linked using reliable keypoints. However, researchers continue to develop new techniques and algorithms to overcome this limitation. Moreover, the enhancement of SLAM systems is also being driven by the emergence of new sensors or sensor combinations, such as cameras, LiDARs, IMUs, and other similar technologies. As these sensors become more advanced and sophisticated, they offer new opportunities to improve the accuracy and reliability of SLAM systems for autonomous driving applications.

Multi-task learning has become a popular paradigm for simultaneously tackling multiple tasks while using fewer computational resources and reducing the inference time. Recently, several self-supervised pre-training methods have been proposed, demonstrating impressive performance across a range of computer vision tasks. However, the extent to which these methods can generalize to multi-task situations remains largely unexplored. Additionally, the majority of multi-task algorithms are tailored to specific tasks that are usually unrelated to autonomous driving, posing difficulties when attempting to compare state-of-the-art multi-task learning approaches in the domain of autonomous driving.

Bird's eye view (BEV) perception involves transforming a perspective view into a bird's eye view and performing various perception tasks, such as 3D detection, map segmentation, tracking, and motion planning. Thanks to its inherent advantages in 3D space representation, multimodal fusion, decision-making, and planning, the topic of BEV perception has attracted significant interest among both academic and industrial researchers.

Road environment perception, which includes 3D geometry reconstruction of road surfaces and the intelligent detection of road damages, is also critical for ensuring safe and comfortable driving. Road surface defects can be extremely hazardous, especially when hit at high speeds, as these can not only damage the vehicle's suspension but also cause the driver to lose control of the vehicle. When one of the vehicle's tyres enters a pothole, the weight distribution across all tyres becomes unbalanced, causing the vehicle to tilt and shift more towards the tyres that are lower relative to the pothole. This uneven weight distribution can produce a considerable and focused force on the tyre when it hits the edge of the pothole, resulting in deformation, breakage, or even bending of the rim. The damage inflicted on the tyre impacts the driving experience, making it challenging to maintain a straight driving trajectory.

This book provides an in-depth, comprehensive, and SoTA review on a range of autonomous driving perception topics, such as stereo matching, semantic segmentation, 3D object detection, simultaneous localization and mapping, and BEV perception. The book's webpage can be accessed at mias.group/ADP2023.

The intended readership for this book primarily comprises of tertiary students who seek a comprehensive and yet an introductory understanding of the fundamental concepts and practical applications of machine vision and deep learning techniques. It is also directed at professionals and researchers in autonomous driving who are seeking an assessment of the current state-of-the-art methods available in existing literature, and who aspire to identify potential areas of research for further exploration. The extensive range of topics covered in this book makes it an invaluable resource for a variety of university programs that include courses related to machine vision, deep learning, and robotics.

In Chapter 1, the book discusses the use of in-sensor visual devices for autonomous driving perception. Chapter 2 provides a thorough and up-to-date review of SoTA environmental perception algorithms that are specifically designed for fish-eye cameras. In Chapter 3, the theoretical foundations and algorithms of computer stereo vision are discussed. Chapter 4 presents a review of SoTA single-modal and data-fusion semantic segmentation networks. Chapter 5 reviews 3D object detection methods for autonomous driving. Chapter 6 provides an assessment of the current SoTA collaborative 3D object detection systems and algorithms. In Chapter 7, sensor-fusion robust SLAM techniques for mobile robots are introduced. Chapter 8 discusses visual SLAM in texture-less environments. Chapter 9 presents a comprehensive survey on multi-task perception frameworks. Chapter 10 specifically covers state-of-the-art BEV perception algorithms. Finally, Chapter 11 discusses road environment perception techniques for safe and comfortable driving.

Shanghai, P. R. China	Rui Fan
Shanghai, P. R. China	Sicen Guo
Bristol, UK	Mohammud Junaid Bocus

Acknowledgements This book was supported by the National Key R&D Program of China under Grant 2020AAA0108100, the National Natural Science Foundation of China under Grant 62233013, the Science and Technology Commission of Shanghai Municipal under Grant 22511104500, and the Fundamental Research Funds for the Central Universities.

Contents

Chapter 1
In-Sensor Visual Devices for Perception and Inference

Yanan Liu, Hepeng Ni, Chao Yuwen, Xinyu Yang, Yuhang Ming, Huixin Zhong, Yao Lu, and Liang Ran

Abstract The traditional machine vision systems use separate architectures for perception, memory, and processing. This approach may hinder the growing demand for high image processing rates and low power consumption. On the other hand, in-sensor computing performs signal processing at the pixel level, directly utilizing collected analogue signals without sending them to other processors. This means that in-sensor computing may offer a solution for achieving highly efficient and low-power consumption visual signal processing. This can be achieved by integrating

Y. Liu
School of Microelectronics, Shanghai University, Shanghai, China
e-mail: yanan.liu@ieee.org

H. Ni (✉)
School of Mechanical and Electrical Engineering, Shandong Jianzhu University, Jinan, China
e-mail: nihepeng20@sdjzu.edu.cn

C. Yuwen (✉)
GAC R&D Center, Guangzhou, China
e-mail: yuwenchao@gacrnd.com

X. Yang
Lancaster University, Bailrigg, Lancaster, UK
e-mail: xinyu.yang@ieee.org

Y. Ming
School of Computer Science, Hangzhou Dianzi University, Hangzhou, China
e-mail: yuhang.ming@ieee.org

H. Zhong
Department of Computer Science, University of Bath, Bath, UK
e-mail: hz877@bath.ac.uk

Y. Lu
School of Computer and Information Security, Guilin University of Electronic Technology, Guilin, China
e-mail: sunnyluyao@vip.qq.com

L. Ran
Pixelcore.ai, Beijing, China
e-mail: light.ran@pixelcore.ai

R. Fan et al. (eds.), *Autonomous Driving Perception*, Advances in Computer Vision and Pattern Recognition, https://doi.org/10.1007/978-981-99-4287-9_1

sensing, storage, and computation onto focal planes with novel circuit designs or new materials. This chapter aims to describe the proposed image processing algorithms and neural networks of in-sensor computing, as well as their applications in machine vision and robotics. The goal of this chapter is to help developers, researchers, and users of unconventional visual sensors understand their functioning and applications, especially in the context of autonomous driving.

1.1 Introduction

The importance of vision as a means of perception cannot be overstated, as it enables efficient collection and interpretation of information [1]. To apply this capability to fields like machine vision, robotics, Internet of Things (IoT), and artificial intelligence (AI), there is a pressing need to develop visual information processing methods and technologies that operate at ultra-high speeds while consuming minimal energy [2, 3]. The conventional machine vision systems and their associated technologies face major constraints in terms of system latency, power consumption, and privacy issues [2, 3]. Unlike the mammalian retina mechanism that rapidly processes raw signals through several layers of cells, the visual signal digitization, storage, and transmission processes involved in conventional machine vision systems can introduce significant time latency, which hinders quick responses to dynamic changes and results in inefficiencies due to irrelevant data processing [2, 3]. Additionally, external image processors like CPU/GPU/VPU/DSPs consume high amounts of power, which is unfavorable for portable tasks [2, 3]. The overwhelming amount of data generated by ubiquitous sensors may obscure the useful information, thus necessitating the extraction of critical information by terminal sensors to reduce data movement from the sensing chip to processing units [4, 5]. Moreover, privacy-sensitive scenarios may require the extraction of crucial information from raw analog signals rather than collected images.

To address these challenges, a paradigm shift towards in-sensor computing is proposed [6]. This approach is inspired by the mammalian retina (Fig. 1.1a) and involves the vision sensor not only acquiring visual information but also processing it to produce highly compressed information instead of video frames (Fig. 1.1c). In-sensor computing offers image-free visual signal processing, which ensures data confidentiality. This interdisciplinary field encompasses various technologies, including sensors, analogue signal processing, near-sensor computing, and in-memory computing (Fig. 1.3). In-sensor computing devices are sensors that integrate perception, temporary storage, and data processing and analysis with raw analogue signals within the sensing chip. While near-sensor computing can reduce the physical distance between sensing and computing, data movement from sensors to processors is still necessary. In-memory computing uses memristors for both memory and computing [7], utilizing tunable resistance as the synaptic weights.

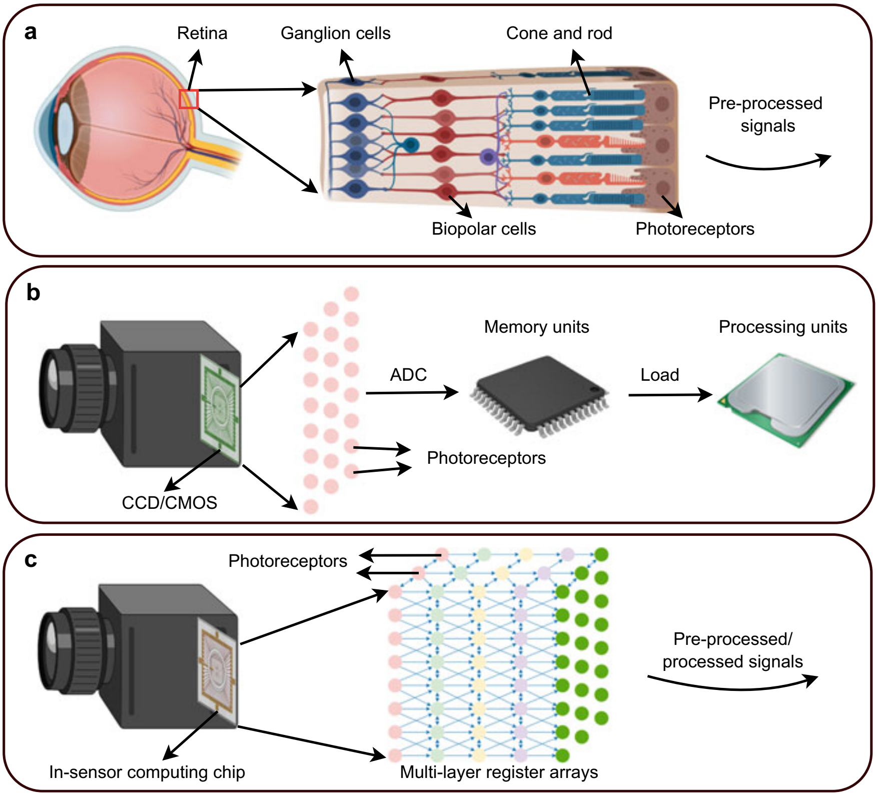


Fig. 1.1 The origin of in-sensor visual computing. a The concept of in-sensor computing is bio-inspired by the retina mechanism where visual signals can be generated and pre-processed by different types of cells [8]. **b** Conventional machine vision system: light density needs to be read out first and converted to digital data which being loaded into memory and then processing units for meaningful information extraction. **c** Visual data can be generated, stored, and processed in sensor through the bio-inspired hardware design

This chapter firstly illustrates the common in-sensor visual computing hardware architecture. Then various emerging in-sensor computing visual sensors are introduced in terms of hardware, software, algorithms, and applications within the category of in-sensor computing architecture. Finally, a summary and future prospective of in-sensor visual computing technology are made in the conclusion.

1.2 In-Sensor Computing Devices

1.2.1 Architecture

In recent years, progress has been made in the development of in-sensor computing devices. By far, there are mainly two types of in-sensor computing architectures:
(1) **In-Sensor architecture by integrating sensing, memory, and computing units**: A Focal-Plane Sensor Processor (FPSP) [9] integrates visual sensing, storage and computing units on the focal plane under the architecture of cellular neural networks (Fig. 1.2a). As for each Processing Element (PE), the generated analogue signals from the pixel can be transferred to the temporal memory units through the bus and processed using ALU units and registers. Each PE plays a role as a cell interacting with its neighbours for signal exchange and processing. Hence, the in-sensor visual inference is realised by the hardware cellular neural network and its synaptic weights in memory. The representative devices under the FPFS architecture mainly include the SCAMP Pixel Processor Array (PPA), Q-Eye [10] MIPA4k [9], Asynchronous-Synchronous Focal-Plane Sensor-Processor Chip (ASAP) [9], KOVA1 [11], and Aistorm Mantis2 [12], where the SCAMP PPA is comparatively mature with continuous research and application outputs.

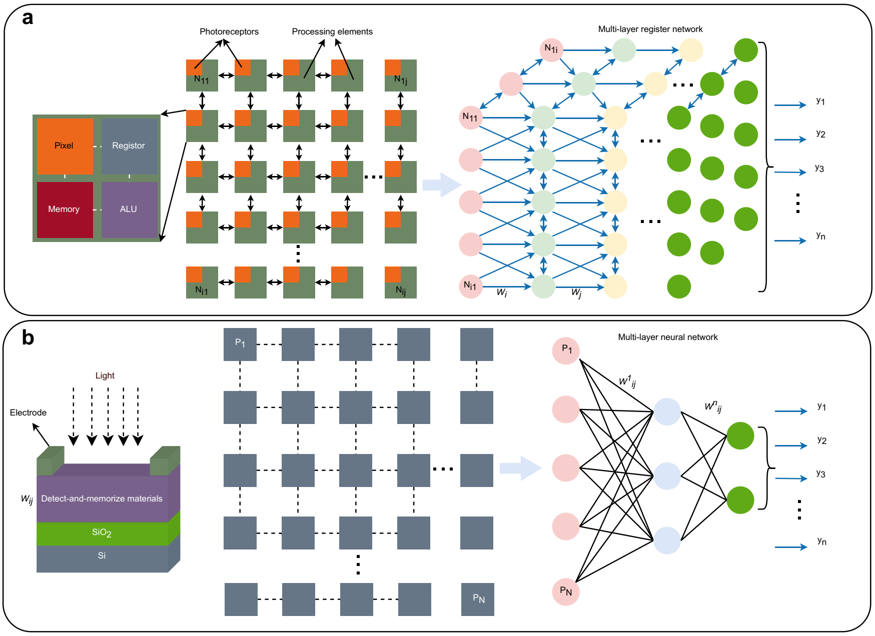


Fig. 1.2 In-sensor perception and computing architectures and their associated artificial networks. a An in-sensor cellular network can be built with an array of PEs which integrates sensing, memory, and computing units. **b** A neural network with detect-and-memorise materials

Table 1.1 In-sensor computing architecture comparison between FPFS and DAM scheme

Scheme	Maturity level	Speed	Application	General-purpose	Programmable	Efficiency
FPFS [15, 16]	Mature	High-speed	Many	Yes	Yes	High
DAM [5, 6, 17–19]	Immature	Ultra-fast	Few	No	Partial	Ultra-high

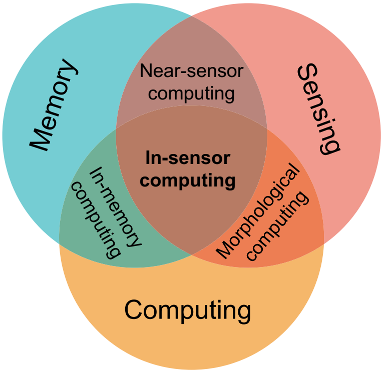


Fig. 1.3 The position of in-sensor computing in the existing knowledge

(2) **Detect-and-memorise materials for in-sensor computing architectures:** Material-based detect-and-memorise (DAM) devices (Fig. 1.2b) have recently been proposed to mimic the functional mechanism of the photonic synapses to implement artificial neural networks [5, 13]. Among these emerging materials and devices, the memristor is representative as it facilitates sensing, temporal memory, and computing capability when combined with other photo-sensitive devices [14]. Specifically, visual signals generated from photoreceptors such as photodiodes can be further processed within the artificial networks composed of memristors with tunable resistance as the weights.

Table 1.1 shows the difference between the two rising in-sensor computing architectures. As can be seen from Table 1.1, the DAM-based in-sensor computing sensors are new and immature compared to the scheme by sensor, memory, and computing integration. Hence, this chapter mainly reviews devices and algorithms based on the first architecture scheme (Fig. 1.3).

1.2.2 *Focal-Plane Sensor Processor (FPFS)*

Conventional sensors mainly play the role of information collectors. In recent years, with the development of techniques on integrated circuit design and the growing need for low-power and lower-latency edge-computing, a sensor has gradually been

Table 1.2 List of in-sensor and in-memory processing vision systems

System names	Resolution	In-sensor storage	Speed (FPS)
Aistorm Mantis2	96 × 96	Gray-scale image	50 K [12]
Eye-RIS	176 × 144	7 gray + 4 binary	10 K [22]
Photodiode array	–	–	20 M bins [14]
SCAMP PPA	256 × 256	7 gray + 13 binary	100 K [23]
Kovilta's KOVA1	96 × 96	–	>1 K [24]
MIPA4k	64 × 64	–	>1 K [25]
DVS	In-senor pre-processing	Binary events	>10 K [26]

integrated with the ability of signal processing independent from general-purpose computers. The goal of near-sensor processing is to use a dedicated machine learning accelerator chip on the same printed circuit board [20], or perhaps 3D-stacked with the CMOS image chip [21]. Despite the fact that this allows CMOS image chip data to be processed closer to the sensor rather than in the cloud, data transport expenses between the sensing chip and the processing chip still exist. In contrast, the in-sensor computing paradigm aims to embed processing capability for each individual pixel. This section introduces classic in-sensor visual computing devices. Table 1.2 lists the differences among these above-mentioned in-sensor computing devices.

1.3 SCAMP-5d Vision System and Pixel Processor Array

1.3.1 Introduction

SCAMP vision system is one of the emerging in-sensor visual computing devices. Currently, the most up-to-date version of SCAMP series system is the SCAMP-5d (Figs. 1.4 and 1.5) which consists of 256 × 256 processing elements (PEs) weighted 171 g with a normal lens. SCAMP-5d vision system is a general-purpose programmable massively parallel vision system [23] that was invented, designed, and developed by University of Manchester. By far, SCAMP-5d enjoys many applications in the field of robotics [27–30] and computer vision [31–33]. As for the PPA shown in Figs. 1.3 and 1.4, the photo-detector converts light into an analogue signal which can be directly parallelly processed on AREG. Different from the current hardware design structure of computer vision systems, the PPA gets rid of the Analogue/Digital Conversion (ADC) after sensing and directly operates on analogue electric current using an arithmetic unit, hence accelerating the signal processing speed and, in the meantime, avoiding the bottleneck of ADC and data transmission process. However, errors can be introduced when performing arithmetic operations or temporal information storage on AREG [6].

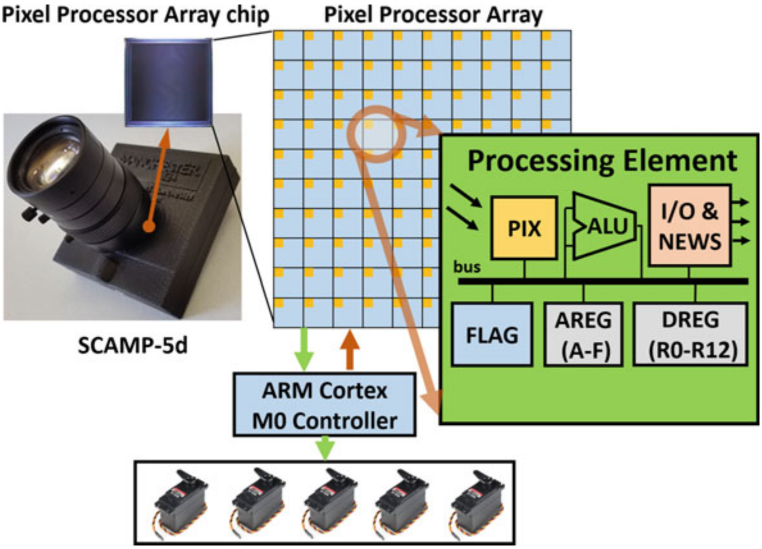


Fig. 1.4 SCAMP-5d vision system and the pixel processor array (PPA). SCAMP-5d consists of PPA with 256 × 256 Processing Elements (PE) and ARM Micro-controller where parallel image processing is conducted on PPA by directly operating on analogue signal (electric current from PIX, which is proportional to the light intensity) within AREG and bit operation within in DREG. Hence, there is no need for time-consuming and energy-inefficient analogue-to-digital conversion. Bio-inspired by the efficient information processing of neurons connected by synapses, PPA is designed to have highly interconnected PE and registers where information can be shared and accessed adjacently enabling efficient parallel machine vision computing. ARM micro-controller is in charge of sending instructions to the PPA, receiving the processed information from the PPA, and more fully-connected layers for deeper CNN

In terms of hardware techniques, the PPA integrates information storage on registers, image processing and analogue information operation (arithmetic operation, shifting, etc.), digital/bit operation, and logical operations. As can be seen from Fig. 1.4, for each Processing element (PE), there are seven read/write AREG (A to F) which can be used for signed value storage and computation with basic arithmetic operations, such as addition, subtraction, division, etc. In addition, thirteen 1-bit DREG (R0 to R12) in each PE (256 × 256 in total) can execute the Boolean logical operations, such as AND, OR, XNOR, and NOT [15] with information after binary thresholding on AREG. Each register in PE executes identical instructions synchronously under SIMD instructions, hence enabling parallel image processing. In addition, the FLAG register can activate different areas of registers given corresponding patterns for more flexible operation. With the neighbour access function where each pixel is able to communicate with its four neighbours (north, west, east, south), an efficient parallel image shifting can be implemented easily. Instructions for the PPA are dispatched by the ARM-based micro-controller with a Cortex M0 running at 204 MHz. The analogue operations is executed at 5 MHz and digital at 10 MHz.

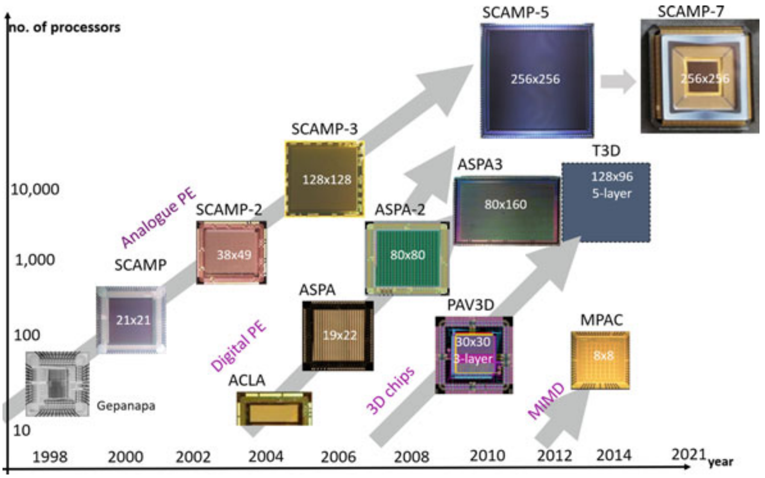


Fig. 1.5 The development process of SCAMP series vision system from the University of Manchester. This review mainly focuses on the SCAMP-3 and SCAMP-5 vision system because their higher resolution and performance would enable more research and applications. *Source* Piotr Dudek's talk in Second International Workshop on Event-based Vision and Smart Cameras (CVPRW) [39]

Other I/O functions, such as USB2.0, GPIO, SPI, and UART, are performed on Cortex M4 Core [15]. Notice that other names for a similar type of focal-plane sensor processor can be seen from [9] with names, e.g. ASPA (Asynchronous-synchronous Focal Plane Sensor Processor Array), FPSP (Focal-Plane Sensor-Processor).

The PPA is a hardware implementation of Cellular Neural Network (CeNN) with the new optimisation on a mixture of both analogue and digital computing using AREGs and DREGs, respectively. The studies based on the PPA reviewed in this work utilise the parallel nature of the CeNN architecture for efficient and high-performance computing, where each "cell" is intricately connected with its four neighbours and information can be shared efficiently. Hence, the PPA can be modelled as a CeNN architecture for visual information computing. The CeNN processing circuit architecture was first proposed by Leon Chua and Lin Yang [34], followed by the CeNN universal machine [35] as a prototype. After that, as an invention of new circuit architecture and a parallel computing paradigm, it enjoys widespread popularity in academia with a substantial number of research outputs and applications in pattern recognition [36], image processing [37], and biological vision modelling [38]. With above-mentioned hardware features, the SCAMP PPA mainly consists of the following advantages over conventional machine systems.

Efficiency and Low Latency: It is obvious to see from Fig. 1.1c that in-sensor computing gets rid of signal digitisation, transmission, and storage processes onto external devices, hence enabling high-speed image processing [23] and CNN inference [40] which can be integrated with agile mobile robot platforms [27–30]. In addition, the PE distribution and simultaneous instruction execution on PEs allow efficient parallel signal processing. Carey et al. [23] demonstrate an object detection with a frame rate of 100,000 per second using the SCAMP vision system and Liu et al. [40] propose binary shallow neural network on the PPA with binary classification problem at up to 17,000 FPS. These works show the efficiency of image processing in a sensor once the parallelism mechanism of the PPA is fully taken advantage of.

Low power consumption: According to Fig. 1.1c, there are no external processing units or data processing needed, hence the power consumption can be saved to a large degree. The maximum power cost of the SCAMP-3 vision system for a complex object tracking and counting system is 29 mW [41]. And the overall power consumption on image processing and CNN inference tasks within a SCAMP-5 vision system is lower than 2 W [42]. This feature makes the SCAMP vision system suitable for mobile platforms, usually with short battery life. In addition, according to the power consumption test from work [43], given 8 popular kernel filters, the SCAMP PPA generates the same convolution results with much lower power consumption ($>$20 times) at a higher speed ($>$100 times) compared to common CPUs and GPUs.

Data Security and Privacy Protection: An unique but non-negligible feature of in-sensor analogue computing with the PPA is its inherent feature of data security and privacy protection. Data security is feasible because of the focal-plane analogue information processing without ADC, extra data recording, storage, or intermediate transmission procedures. Usually, the only output after analogue computing is the extracted useful target information without redundant information, which is hardly reversible to get the original data for sensitive information or user re-identification [44]. Data security and privacy protection have become prominent challenges with the emergence of the internet of things. Smart devices such as autonomous vehicles, domestic robots, and smart household appliances are usually equipped with perceptual sensors and collect data pervasively in public and private spaces, threatening users' privacy and data security. Conventional sensors usually directly upload raw data to the cloud for data processing [45], which can be a fault line of data security. When data is processed manually or the network is attacked, crucially sensitive data can be directly obtained. The acquired data can then be applied to determine individuals' habits (e.g., motion sensors) or to conduct surveillance (e.g., facial recognition systems), which can cause significant violations of EU GENERAL DATA PROTECTION REGULATION Article 25.[1] In sensor computing first enables only valuable information to be extracted as it's output, without redundant information. Moreover, the minimised raw data are further mocked by the analog signals, which leads to re-identification almost impossible. Hence, users' privacy can be strictly protected with in-sensor processing mechanisms.

[1] https://gdpr-info.eu/art-25-gdpr/.

1.3.2 Algorithms and Applications

This section mainly illustrates algorithms and their potential applications (Fig. 1.7) based on the SCAMP-5 vision systems (Table 1.3 and Fig. 1.6).

1.3.2.1 Image Enhancement

Image enhancement comes along with the imaging process on the PPA compared to the conventional image enhancement which only happens after the image data is captured. Later, other methods are exploited on different image processing tasks. For example, Wang et al. [67] proposed a simple coarse grain mapping method to process bigger images than the PPA resolution itself by temporarily storing sub-images into different registers.

Table 1.3 List of main studies with the SCAMP PPA

Image processing methods	Applications	References
Background extraction	Segmentation	[46, 47]
Contour extraction	Object detection	[48–50]
Skeleton extraction	Shape simplification	[51–53]
HDR	Image enhancement	[31, 48, 54, 55]
Feature corner/edge extraction	Edge/feature-based VO	[33]
Target detection/localisation	High-speed object tracking	[23, 56, 57]
Neural network	High-level inference	[40, 42, 58–61]
Depth estimation/visual odometry	Robot navigation	[29, 30, 32, 60, 62–65]
Automatic code generation	Neural network inference, face detection	[43, 66]

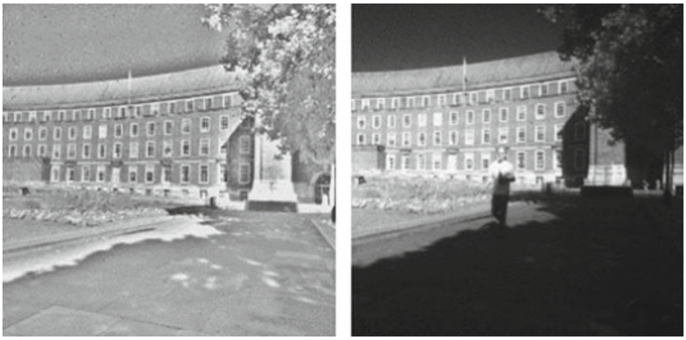


Fig. 1.6 Examples of two images with(left)/without(right) HDR algorithms towards the same scene in an outdoor environment

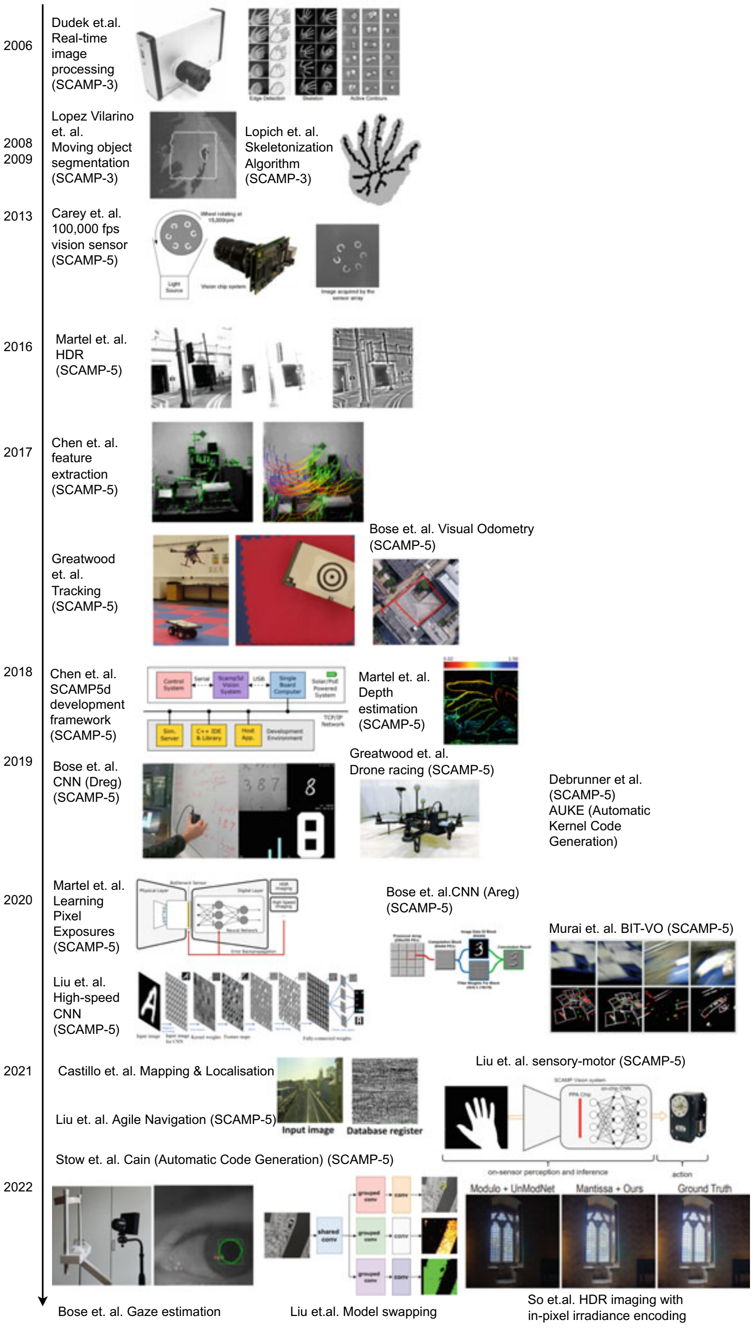

2006
Dudek et.al. Real-time image processing (SCAMP-3)
2008
2009
Lopez Vilarino et. al. Moving object segmentation (SCAMP-3)
Lopich et. al. Skeletonization Algorithm (SCAMP-3)
2013
Carey et. al. 100,000 fps vision sensor (SCAMP-5)
2016
Martel et. al. HDR (SCAMP-5)
2017
Chen et. al. feature extraction (SCAMP-5)
Bose et. al. Visual Odometry (SCAMP-5)
Greatwood et. al. Tracking (SCAMP-5)
2018
Chen et. al. SCAMP5d development framework (SCAMP-5)
Martel et. al. Depth estimation (SCAMP-5)
2019
Bose et. al. CNN (Dreg) (SCAMP-5)
Greatwood et. al. Drone racing (SCAMP-5)
Debrunner et al. (SCAMP-5) AUKE (Automatic Kernel Code Generation)
2020
Martel et. al. Learning Pixel Exposures (SCAMP-5)
Bose et. al.CNN (Areg) (SCAMP-5)
Murai et. al. BIT-VO (SCAMP-5)
Liu et. al. High-speed CNN (SCAMP-5)
2021
Castillo et. al. Mapping & Localisation
Liu et. al. sensory-motor (SCAMP-5)
Liu et. al. Agile Navigation (SCAMP-5)
Input image
Database register
Stow et. al. Cain (Automatic Code Generation) (SCAMP-5)
on-sensor perception and inference
action
2022
Modulo + UnModNet
Mantissa + Ours
Ground Truth
Bose et. al. Gaze estimation
Liu et.al. Model swapping
So et.al. HDR imaging with in-pixel irradiance encoding

Fig. 1.7 Milestones SCAMP PPA-based work and key SCAMP PPA studies and applications during last 16 years

HDR (Fig. 1.6) is a basic low-level image pre-processing method to obtain rich image information even facing extreme lighting conditions, such as the mixture of dim and strong light intensity. However, conventional image sensors rely on either a global or rolling shutter to form an image, which limits the efficiency of HDR imaging [31, 68]. Back in 2006, Dudek [48] proposed sensor-level adaptive sensing and image processing with SCAMP-3 [41, 69], where different exposure settings are combined for an image with a high dynamic range. Martel et al. [55] make significant contributions in this area using the PPA. The first HDR image generation in-sensor is from [54] where pixel-wise exposure can be controlled to generate HDR images, followed by automotive applications [70]. Furthermore, Bose et al. [32] take advantage of the HDR image to extract edges as the robust input information for visual odometry estimation. However, the usage of iterative exposure for different regions of the image slows down the image pre-processing. To speed up the HDR imaging, Martel et al. [31] propose the learning shutter function for PEs to expose each pixel independently with an end-to-end training strategy. They obtain an exposure function by training a U-Net neural network and compiling these trained functions on the sensor for inference. Later, So et al. [71] presented in-pixel control for snapshot HDR imaging with irradiance encoding.

1.3.2.2 Contour and Skeleton Extraction

Contours are important features for objects within an image, which can help to identify different entities. Contour extraction algorithms were proposed based on a pixel-level snake with very low latency [49]. In 2007, Alonso-Montes et al. proposed the in-sensor automatic retinal vessel tree extraction based on the Cellular Neural Networks [50]. The shared key concept for these works [48–50] is to extract contour iteratively based on the active contour model and Cellular Neural Networks. In 2008, [72] proposed an image pre-processing method based on the cellular automata for a robotic scenario. The skeleton within a binary image shows the object size, position, and simplified shape. Fast image skeletonization [51] is implemented by [52] based on the wave-trigger propagation/collision. Examples of image contour and skeleton extraction based on the SCAMP PPA can be seen in Fig. 1.8.

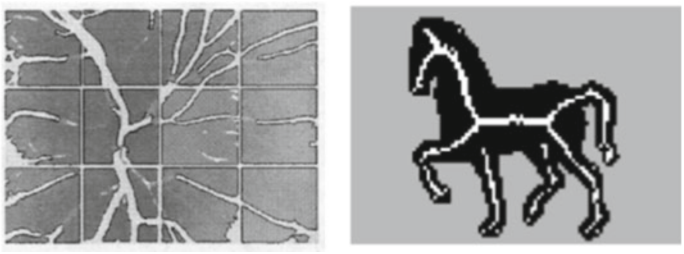


Fig. 1.8 Examples of image contour and skeleton extraction using SCAMP PPA. Left: Extracted Retinal Vascular Tree, Figure from [50]. Right: Extracted skeletons, Figure from [47]

Other Feature Extraction Methods: Other image processing methods, such as background extraction, is exploited by Wang et al. [46, 47]. For higher-level feature extraction, the edge feature can be obtained by deploying Sobel kernel filters or Laplacian filters, which are used in the later work for focal plane visual odometry [32] and neural networks [60]. As for other features, such as corner points extraction, Chen [33] utilised the DREG for corner points extraction based on the FAST16 algorithm, which is used in later work on visual odometry [65]. Based on the above-mentioned low- and mid-level image processing methods, researchers are motivated to exploit more general high-level image processing with up-to-date techniques by taking advantage of the earlier milestone work and the state-of-art progress, such as neural networks which would be illustrated in Sect. 1.3.2.4.

1.3.2.3 In-Sensor Visual Feature Extraction for Robots

Two major constraints that preclude mobile robots from long-term and diverse applications are their short battery life and limited load. Emerging sensors may hold the key to solving this challenge due to their unique low-level hardware design. The portable SCAMP-5d vision system (171 g including the lens) can perform spatial AI processing in-sensor, reducing data transfer pressure between sensing and the main processor, hence increasing overall processing efficiency while maintaining low power consumption [73].

(a) SCAMP PPA on a Quadrocopter

The SCAMP-5d vision system has been integrated into quadrocopter systems for target tracking, visual odometry and racing. Greatwood et al. perform various experiments by integrating a SCAMP-5d vision system and a quadrotor [28, 29, 74]. Figure 1.9 shows a flight control system in terms of hardware integration and control block diagram, where a pre-set target can be tracked with extracted useful information on sensor even facing short periods of target tracking loss [28]. In this application, the direct in-sensor target position extraction releases the pressure of image capturing, transmission and processing for the whole system. Later, Greatwood et al. proposed the in-sensor visual odometry using perspective correction on an agile micro air

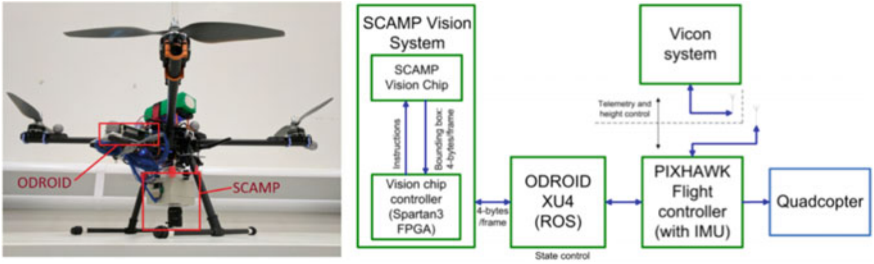

ODROID
SCAMP
SCAMP Vision System
SCAMP Vision Chip
Instructions
Bounding box: 4-bytes/frame
Vision chip controller (Spartan3 FPGA)
4-bytes /frame
ODROID XU4 (ROS)
State control
Vicon system
Telemetry and height control
PIXHAWK Flight controller (with IMU)
Quadcopter

Fig. 1.9 Left: The integration of a quadrotor and SCAMP-5 vision system for object tracking. Right: a diagram of system hardware (Figure from [28])

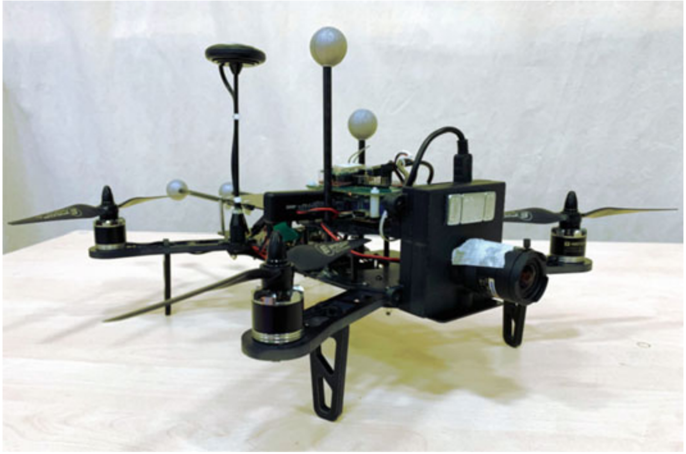


Fig. 1.10 Quadrotor setup for the drone racing with a front-facing SCAMP (Figure from [74])

vehicle based on a similar hardware platform. After that, a drone racing Fig. 1.10 within a pre-set environment is demonstrated by taking advantage of the efficient image processing ability on the PPA [74], where the target position can be estimated at around 500 FPS. McConville et al. [30] apply the in-sensor visual odometry developed by Bose et al. [32] on an unmanned aerial system for real-time control purposes.

(b) SCAMP PPA for Mobile Robot Reactive Navigation

In terms of navigation with a SCAMP PPA, Liu et al. [27] proposed reactive agile navigation on a non-holonomic ground vehicle using PPA by robustly recognising pre-set patterns out of complex environment background. Although being very efficient and accurate, using a pre-set fixed pattern for target tracking is difficult to expand in the generalised environment where there are usually random features. With this in mind, Chen et al. [60] use in-focal plane feature extraction from the environment to perform a recurrent neural network on the M4 micro-controller using this extracted information to estimate the proximity to the ambient objects for obstacle avoidance purposes. A similar pattern of concentric circles was employed in [27, 28, 30] to effectively extract the dot centre in the circles out of the complex environment (Fig. 1.11).

(c) In-Sensor Computing for Mapping and Localisation

Mapping and localisation are useful techniques for robot navigation. In-sensor mapping and localisation are lightweight and low power cost solutions for mobile platforms. Castillo-Elizalde et al. [75] for the first time proposed 1-D mapping and localisation technique. For this method, features are firstly extracted as the database from a sequence of images. Then, the incoming image can be localised by comparing with the database and the prior knowledge of the motion model. In their work, two methods were utilised to down-sample the original images: direct resizing and local binary pattern to apply them to different localisation situations.

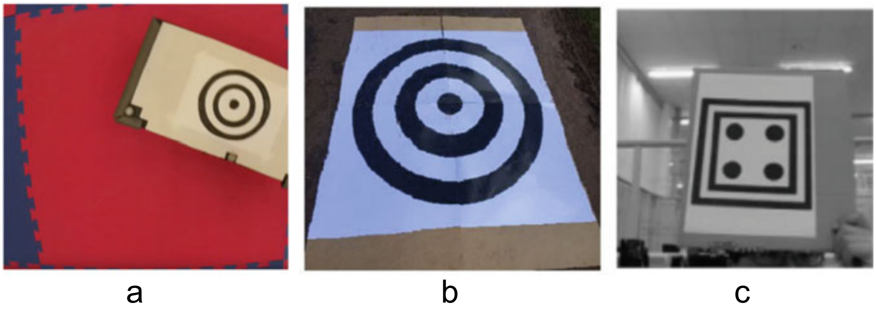


Fig. 1.11 Tracking pattern for the drone and ground vehicle. **a** Tracking a ground vehicle [28], **b** Tracking a moving target while performing a visual odometry [30], **c** Tracking a fixed pattern with a mobile ground vehicle [27]

(d) Pose and Depth Estimation

For decades, egocentric state estimation has been studied using conventional cameras, emerging DVS devices, and CPU/GPUs. In recent years, there have been some studies utilising SCAMP PPA. For example, Bose et al. [32] for the first time, proposed in-sensor 4 Degree-of-Freedom (DoF) visual odometry wholly on the sensor by mapping the real-time input image with the previous keyframe through image scaling, shifting, rotation and alignment. They demonstrate the visual odometry estimation at over 1000 Hz with around 2 W power cost. Debrunner et al. [76] use the SCAMP to estimate its global motion with the tiling method at 60 FPS with a low power cost of 100.2 mW. After that, Murai [65] proposed 6 DoF visual odometry based on edge and corner points extracted on sensor and post-processing on a computer with a frame rate of 300 FPS. They take advantage of feature edge, and corner extraction methods [33] and calculate the visual odometry off sensor using a similar strategy with the standard Visual Odometry (VO) systems [77]. Although they combine in-sensor feature extraction and ready-to-use VO computing method off the sensor, it is promising to be a direction in the future to combine the efficient image pre-processing in-sensor and high-volume post-processing with a powerful CPU/GPU, especially when facing storage shortage and general calculation resources for the large-scale computing.

In addition, the SCAMP vision system can also work with other accessories to share the computation burden for more applications. For example, Martel et al. [62–64] mounted a controllable liquid lens to generate a semi-dense map in real-time, which is the first work on depth estimation to take advantage of external physical accessories. With this focus-tunable lens, a vast amount of computation pressure on the sensor is relieved. This in-sensor feature extraction and post-image processing on controller scheme are also widely used in many different applications [60, 65], where the task requirement of storage and computing resources is out of the capacity of the PPA.

1.3.2.4 Research Progress on Neural Networks with a SCAMP PPA

The algorithms of SCAMP PPA proposed earlier mainly focus on low-level image processing and/or machine vision methods to enhance image quality and extract basic textures with combinations of inherent built-in functions based on SCAMP-3 and SCAMP-5 with a PE resolution of 128 × 128 and 256 × 256, respectively. It should be noticed that these developed image processing methods are deeply related to the hardware design of the SCAMP vision system. For example, common methods used in this period are cellular-based algorithms, including cellular neural networks, because the SCAMP PPA itself is a cellular processor array.

Research on neural network inference with the SCAMP PPA has been active in recent years. Table 1.4 lists the main research work in the area of neural networks, which covers fully convolutional neural networks and binary convolutional neural networks using DREG or AREG with various datasets and applications. High-level image processing, such as object classification, localisation and segmentation in sensor, is achieved with the neural network. The deployment of neural networks onto the PPA is a breakthrough since it enables the PPA open to more possibilities with universal methods, which is unlike the conventional development methods with some combinations of low-level image processing methods for specific tasks. With the use of CNN, several types of tasks, such as classification, regression, localisation,

Table 1.4 Different convolutional neural network implementation with SCAMP and performance comparison.

Network	Filter number	Layers (Conv+FC)	Dataset	Accuracy	Frame rate (fps)	In-sensor/ Near-sensor
Bose [42]	16 5 × 5	1 + 1	MNIST	≈94.2%	210	1Conv/1FC
Bose [58]	16/64 4 × 4	1 + 1	MNIST	≈93%	>3000	In sensor
–	16+16 4 × 4	2 + 1	MNIST	92–94%	224	In sensor
Liu [40]	64 4 × 4	1 + 1	8 Plankton	80.5 %	4016	In sensor
–	16 4 × 4	1 + 2	8 Hand gestures	<98.7%	2092	1Conv+1FC/1FC
–	16 4 × 4	1 + 1	Roshambo	<97.73%	8264	In sensor
–	64 4 × 4	1 + 1	0,1 in MNIST	<99.1%	17543	In sensor
Liu FCN [59]	16+64 4 × 4 64 1 × 1	3 + 0	Simulation	–	283	In sensor
Liu [78] Binarized CNN	16+64 4 × 4	2 + 2	EMNIST	<86.74%	178	2Conv+1FC/ 1FC
Chen [60]	–	0 + N	Collected indoor	–	–	Near sensor
AnalogNet [61]	3 3 × 3	1 + 3	MNIST	96.9%	2260	1Conv/2FC

and segmentation, can be feasible, hence enabling more applications. Table 1.4 shows the neural network-related work based on the SCAMP PPA vision system.

The research on CNN implementation and inference within PPA is pioneered by Bose et al. [79] where a CNN with a single convolutional layer performed upon the PPA array and a fully-connected layer upon its controller chip (M0). They performed 16-bit image convolution operations using 4 × 4 DREG "Super Pixel" blocks and demonstrated live digit classification based on MNIST dataset at around 200 FPS. In their work, the ternary $\{-1, 0, 1\}$ kernel filters are stored on the flash (M4) of the PPA system, and are effectively encoded in the instructions/operation sent to the PPA array, performing convolutions sequentially. Furthermore, a mobile car localisation task is then explored using synthetic datasets, where the pre-processed edge information is mainly the clues for network inference. Notice that the localisation is realised by classifying the car's position along the x and y axis, respectively. To fully take advantage of PPA's parallel computing characteristics and to further improve the CNN inference efficiency, Bose et al. [58], for the first time, proposed the idea of in-pixel weight storage, where the network's weights are directly stored within the registers of the PPA's PEs. This method enabled both parallel computations of multiple convolutions, and implementation of a fully connected layer upon the PPA array, resulting in a ×22 faster CNN inference (4464 FPS) on the same digit recognition task. Based on these two works, [40] further proposes a high-speed lightweight neural network using BinaryConnect [80] with a new method for computing convolutions upon the PPA, allowing for varying convolutional strides. This work demonstrated four different classification tasks with frame rates ranging from 2,000 to 17,500 per second with different stride setups. Later, based on this network, a direct servo control using CNN results [81] and a simulated robot tracking from a drone [82] with in-sensor CNN computing results are exploited. In addition, the AnalogNet2 [61, 83] extends the earlier work in [84], implementing a CNN which reaches 96.9% accuracy on the MNIST dataset at a speed of 2260 fps. However, their method requires all fully connected layers to be performed externally to the PPA array with only 3 convolutional kernel filters implemented in sequence on the PPA as the first layer. More kernel filters would significantly slow down the inference process. Notice that, in our work [60], a recurrent neural network is implemented on the micro-controller with features extracted on a sensor. In this manner, the fully-connected layer of a neural network can be deployed similarly with conventional embedded devices. It is notable that Martel et al. trained a neural network of exposure time for each individual pixel off the sensor for HDR imaging and video compressive sensing [31].

Furthermore, work [78] binarized CNN with batch norm both for classification and coarse segmentation. To deal with the classifications application with more labels and more segmentation tasks, they propose the idea of dynamic model swapping by uploading weights of trained models in sequence or according to the last inference result, targeting multiple sub-tasks decomposed from a more sophisticated task. They then demonstrate a servo control directly using the CNN inference results [81], which potentially indicates that motion control platforms, such as a ground vehicle or drones can have a light-weight servo control system without using external control units in the future.

Notice that the preceding neural network-related work mainly focuses on classification or classification-based localisation, both of which require fully connected layers. However, the parameters in fully-connected layers are typically substantially larger than those in convolutional layers due to the dense connections of each individual neuron. Thus, work developed a fully-convolutional neural network (FCN) [59], not only presenting in-sensor image segmentation and localisation but also eliminating dense layers for a smaller memory footprint [85].

1.3.2.5 In-Sensor Cellular Automata

The PPA itself is a cellular neural network architecture where each 'cell' is closely connected with its four neighbours, hence information can be shared efficiently. With this in mind, the author is inspired to explore the possibility to perform cellular behaviour, such as Conway's game of life (demonstration shown from[2]) and elementary cellular automata (demonstration Rule 90 seen from[3]) based on the theory of cellular automata [86]. With the rule of the game of life, all 'cells' can update their states (alive or dead) in-sensor as fast as 53 μs for each iteration based on the bit-operation with DREG. As can be seen from Fig. 1.12, a Sierpiński triangle is efficiently generated based on bit operation on the sensor with 730 μs of 255 iterations to fill the whole chip.

(a) Elementary CA
One-dimensional CA is one of the simple CA algorithms. Some classical updating rules, such as Rule 30, Rule 90, and Rule 110, can be implemented by logic bit

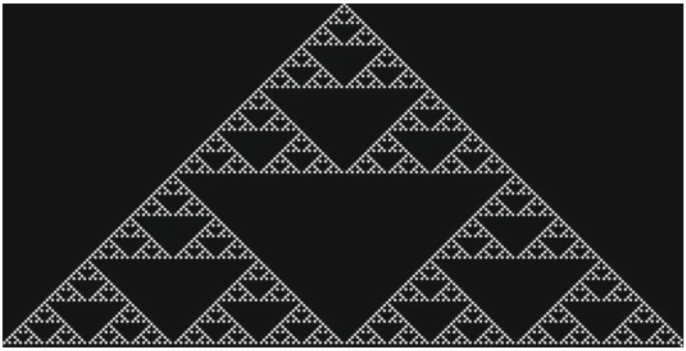


Fig. 1.12 Our demonstration of elementary cellular automata with Rule 90 on the SCAMP PPA. This pattern is generated from top to bottom. We have made this project available from https://github.com/yananliusdu/1D_CellularAutomata

[2] https://youtu.be/X_t4c3f-T4s.

[3] https://youtu.be/HgPvoK5EJ_s.

operations. Let use L,C,R as the three continuous grids and C1 is the grid that should be updated, then rules can be represented as follows:

```
R30: C1 = L XOR (C OR R )
R90: C1 = XOR(L, R)
R110: C1 = XOR(OR(R, C), AND(R, C, L))
```

The pseudo codes for R90 can be shown as follows:

```
\begin{code}{ElementaryCellularAutomataR90}
for(int i = 1; i <= iterations; i++)
{
    scamp5_kernel_begin();
       MOV(R5,R6); //R5 = R6
       DNEWS(R0,R6,west); //R0 = shift R6 to right for one step
       DNEWS(R6,R0,west);
       XOR(R7,R5,R6); // R7 = XOR(R5,R6)
       DNEWS(R0,R7,north);
       DNEWS(R6,R0,east);
       OR(R6,R5); R6 = OR(R6,R5)
    scamp5_kernel_end();
}
\end{code}
```

Implementation for R30 and R110 can also be done similarly with R90 using corresponding unit logic operations.

(b) In-Sensor Conway's Game of Life based on the CA

The rule of Game of Life is a new independent non-linear computing scheme.

Figure 1.13 shows an updating 3×3 block to illustrate the rule and implementation using a cellular neural network.The pseudo code for the Game of Life can be represented as follows according to its definition:

```
if B0 = 1 & SUM(B1,..,B8) <= 2
   B0 = 0
If else B0 = 0 & SUM(B1,..,B8) == 3
   B0 = 1
else
   B0 = 0
```

where $B_i \in \{0, 1\}, i = \{0, 1, 2, \ldots, 8\}$.

This chapter tries to implement and run the Game of Life using the DREG of SCAMP-5 vision systems and its parallel nature. We use three DREGs 3-bit

Fig. 1.13 The update block for the Game of Life

B1	B2	B3
B4	B0	B5
B6	B7	B8

$R6$, $R7$, $R8$ to count the number of ‘1’s around the pivot $B0$. $R5$ is the source binary image in this illustration. $R1$, $R2$, $R3$, $R4$ and $R9$, $R10$ are DREGs to temporarily store intermediate states.

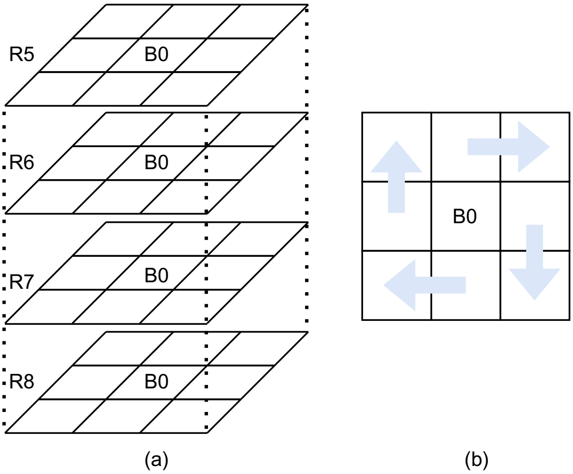
Neighbour Number Counting

As the first step of the 2D cellular automata, the number of live neighbours should be counted. As shown in Fig. 1.14a, $R5$ is the source binary image. $R6$, $R7$, $R8$ are 3-bits to represent the number of live neighbours of R5 in the corresponding position. For example, if $R5(B0)$ has 2 live neighbours, then $R6(B0) = 0$, $R7(B0) = 1$, $R8(B0) = 0$. In conclusion, binary digits $R6$, $R7$, $R8$ are used to record the number of live neighbours of corresponding cell. Figure 1.14b shows for each counting step (8 in total around a pivot, $R6$, $R7$, $R8$ are updated according to the states of each cell.

State Update for Cells

With live neighbour information stored in three DREGs, cells’ state can be updated according to the rule of the Game of Life as described in the aforementioned pseudo code. We make the codes of Game of Life on the SCAMP-5 vision system available from https://github.com/yananliusdu/GameofLife. In the future, more image

Fig. 1.14 Neighbour number counting using DREGs



processing-related work can be potentially explored as long as proper update rules and associated steps are trained with neural network methods [37, 87].

1.4 Eye-RIS

1.4.1 Introduction

Eye-RIS [22, 88] commercial vision system on Chip (VSoC) extends CMOS pixel functionality with image storage (7 gray-scale images and 4 binary images) and digital/analogue signal processing ability. Specifically, a 32-bit RISC (Reduced Instruction Set Computer) is integrated with a vision sensor for image post-processing after the parallel in-sensor pre-processing (Fig. 1.15). The resolution of the Eye-RIS vision sensor is 176 × 144. Notice that Eye-RIS's overall functional diagram is similar to that in the SCAMP vision system, where the counterpart of RISC is the M0 microcontroller in the SCAMP PPA [89]. The most significant difference, though, is that the Eye-RIS has a DICop part, which is a digital image co-processor that handles geometric transformations and can send the results back to the pixel level for more processing.

The Eye-RIS Vision System on Chip (VSoC) is an autonomous device combining a parallel CMOS image sensor processor with 32-bit RISC microprocessor performing post-processing and system control tasks, several I/O and high-speed communication ports that allow the system to communicate and/or to control external systems, and on chip memory. The combination of massive parallel image pre-processing in the sensor with complex image post-processing in the microprocessor results in ultra compact implementation of a vision system able to perform complex machine vision algorithms at speeds of several thousands of images per second. The Eye-RIS VSoC

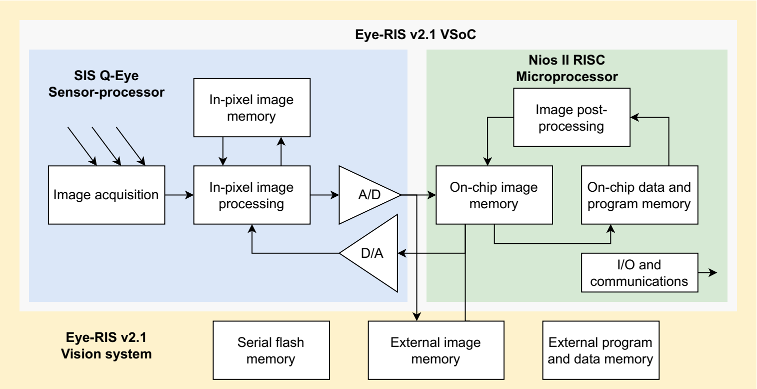

Eye-RIS v2.1 VSoC
SIS Q-Eye Sensor-processor
In-pixel image memory
Image acquisition
In-pixel image processing
A/D
D/A
Nios II RISC Microprocessor
Image post-processing
On-chip image memory
On-chip data and program memory
I/O and communications
Eye-RIS v2.1 Vision system
Serial flash memory
External image memory
External program and data memory

Fig. 1.15 Eye-RIS v2.1 VSoC architecture

features a complete application development software environment allowing easy control of the device and is optimized for industrial applications requiring image sensing, image processing, and decision making at extreme frame rates.

1.4.2 Applications

The applications with Eye-RIS consist of automotive, machine vision, security, games and battery powered products. Caballero-Garcia and Jimenez-Marrufo [90] proposed a series of techniques to deploy image processing algorithms on the Eye-RIS Vision System on Chip (VSoC) for various applications. One unique characteristic using the Eye-RIS vision platform compared to conventional visual sensors is the simultaneous image acquisition and early processing in the analogue domain.

Paper [91] aims to describe how the AnaFocus' Eye-RIS family of vision systems has been successfully embedded within the roving robots developed under the framework of SPARK and SPARK II European projects to solve the action-oriented perception problem in real time. With the ability of low power cost and efficient parallel image processing, Eye-RIS has been equipped to many different mobile robot platforms, such as Rover II wheeled robot and Gregor III hexapod robot. Visual homing, object tracking, and navigation using landmarks are demonstrated based on the robot platforms and in-sensor real-time image processing algorithms (Fig. 1.16).

Optical flow based on the Lucas and Kanade by Guzman et al. [92] (Fig. 1.17) is implemented on the Eye-RIS platform taking advantage of both analogue and digital signals processing using Q-Eye and Nios II RISC respectively (Fig. 1.15). In the experiment, the optical flow estimation reaches over 25 fps which can be used in the area of robotics in real time. Specifically, the optical flow constraint equation:

$$uI_x + vI_y + I_t = 0 \tag{1.1}$$

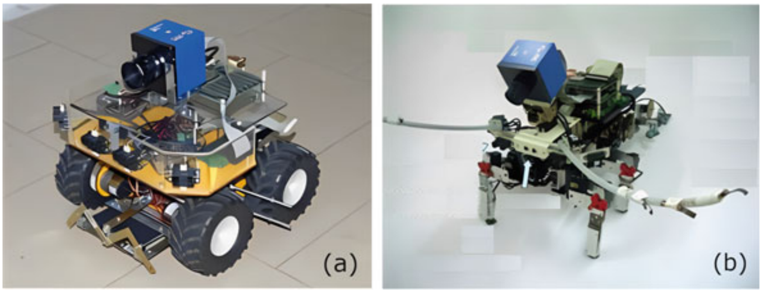


Fig. 1.16 Eye-RIS VSoC equipped onto robot platforms. **a** Rover II wheeled robot, **b** Gregor III hexapod robot (Figure from [91])

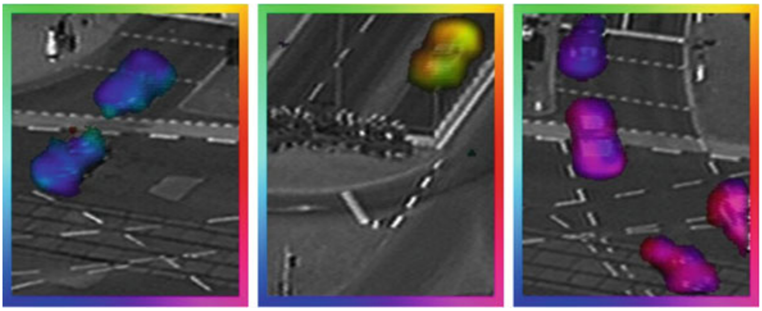


Fig. 1.17 Optical flow estimation in traffic sequences using Eye-RIS VSoC architecture (Figure from [92])

where the partial derivatives of I are denoted by subscripts, which can be obtained from the image. u and v are the x and y components of the optical flow vector, which are the optical flow vectors that need to be found out.

Lucas–Kanade method [93] is a classical method to deal with the optical flow constraint problem. According to the assumption of Lucas–Kanade method, given a small pixel block, 3 × 3 for example, the optical flow remains identical within this small block. Then, we can have following equation group:

$$\begin{aligned} uI_{x1} + vI_{y1} &= -I_{t1} \\ uI_{x2} + vI_{y2} &= -I_{t2} \\ &\dots \\ uI_{xn} + vI_{yn} &= -I_{tn} \end{aligned} \tag{1.2}$$

Equation 1.2 can then be represented as

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \dots \\ I_{xn} & I_{yn} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \dots \\ -I_{tn} \end{bmatrix} \tag{1.3}$$

we assign Eq. 1.3 as $A\overrightarrow{V} = -b$, here the least squares method can be used to get optical flow vector $\overrightarrow{V}$, then $A^T A \overrightarrow{V} = A^T(-b)$, hence, the optical flow vector can be obtained through:

$$\overrightarrow{V} = (A^T A)^{-1} A^T(-b) \tag{1.4}$$

In detail, Eq. 1.4 can be shown as:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} I_{xi}^2 & \sum_{i=1}^{n} I_{xi} I_{yi} \\ \sum_{i=1}^{n} I_{xi} I_{yi} & \sum_{i=1}^{n} I_{yi}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_{i=1}^{n} I_{xi} I_{ti} \\ -\sum_{i=1}^{n} I_{yi} I_{ti} \end{bmatrix} \tag{1.5}$$

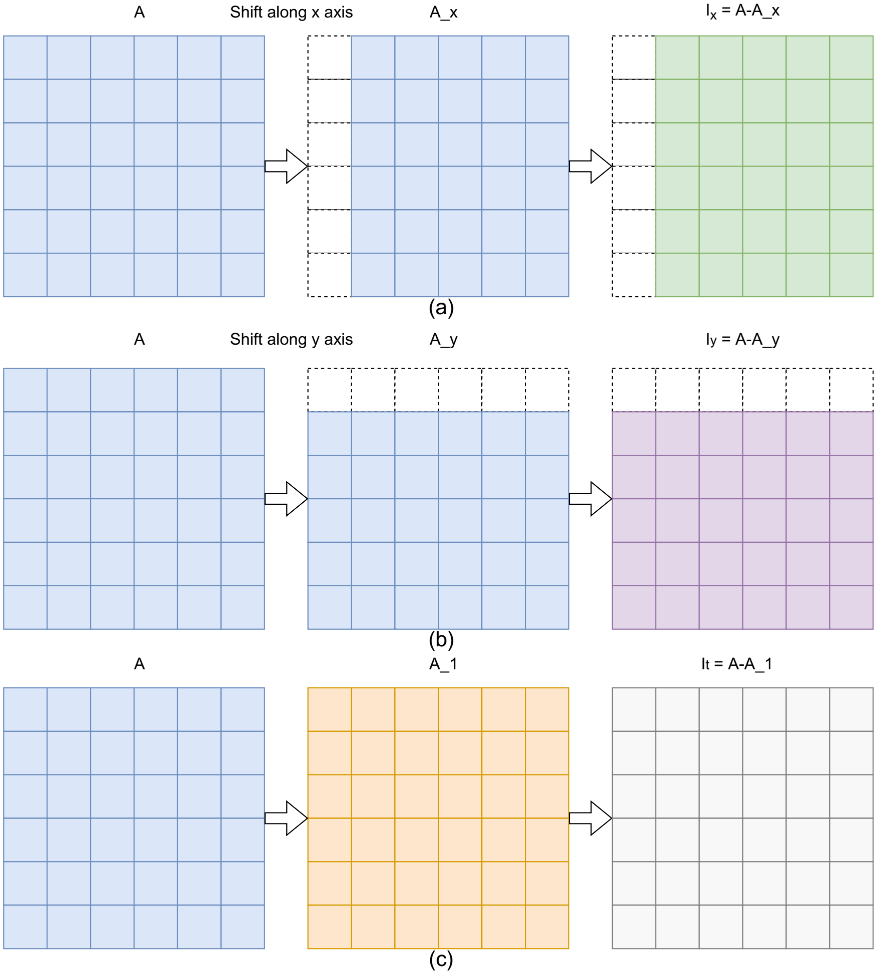


Fig. 1.18 The parallel calculation of I_x, I_y, I_t in the focal plane using analogue signals

Through Eq. 1.5, the optical flow vector can be estimated through a sequence of images. These intensity derivatives including I_x, I_y, I_t can be efficiently obtained by shifting and subtracting between frame sequences as illustrated from Fig. 1.18. After that, the optical vector can then be calculated using conventional computing units. As for the implementation of optical flow using the above-mentioned formulations, work [22] takes advantage of both in-sensor pre-processing and post-processing with computing units achieving up to 28.9 fps.

In the study of [94], authors explored different methods of point tracking on the platform of Eye-RIS, which is able to equip Unmanned Arial Vehicles (UAVs) with the ability of on-board sensing and computing with low load and power consumption.

Nicolosi et al. [95] applied the Eye-RIS vision platform to control welding process by using the cellular neural network based visual algorithms. [96] extended their work onto vision based closed loop control for partial penetration welding of overlap joints.

Work [97] proposed algorithms for the cellular neural network to detect rapidly approaching object which is bio-inspired by mammalian retina that consists of looming sensitive circuit based on local interaction of cells.

1.5 Kovilta's KOVA1

1.5.1 Introduction

The KOvilta Vision Array (KOVA1) (depicted in [11]) employs a meticulously designed pixel-level processing circuitry that utilizes an efficient combination of analogue and digital (mixed-mode) computation techniques to execute a diverse range of pre-programmed operations. These operations include automated sensor adaption, grayscale filtering, segmentation, and complex object-level visual analysis. The selection of operations to be performed at the pixel-level hardware can be customized based on the specific requirements of the application, thereby enhancing the overall implementation efficiency.

The KOVA1 is Kovilta's inaugural silicon rendition of the KOvilta Vision Array structure, and it comprises a 96×96 pixel focal-plane processor array fabricated using 180 nm CMOS technology. The sensor-processor chip is integrated into a miniature smart camera system equipped with FPGA-based control and Ethernet I/O. In this focal plane processor architecture, each pixel-cell of the sensor array includes a reconfigurable processing element that operates directly on the output of the analog photodiode. This allows real-time data compression for capturing images with high dynamic ranges without compromising quality. Additionally, processing in parallel on the pixel plane enables rapid low-level feature analysis and eliminates the need for time and energy-consuming long-distance data transfers from the sensor to an external processor. The sensor output may include only the essential feature data, such as the presence of an object or a set of object coordinates or features, thereby reducing the amount of hardware needed for further external image content analysis.

The KOVA1 camera system employs pixel cells that are connected within their immediate neighbourhood, allowing for direct information exchange during image analysis activities at the sensor level. Moreover, local memories integrated at the pixel level facilitate the storage of multiple full images or intermediate processing outcomes on the sensor plane. An FPGA chip is utilized to manage the program execution and I/O of the sensor-processor chip in the KOVA1 embedded camera system. As the control and I/O structures consume only a small fraction of the FPGA's resources, supplementary visual analysis operations can be implemented on the FPGA to enhance the on-chip sensor-level processing. The KEDE software environment from Kovilta is used to program and manage the camera, with programming and data I/O facilitated by an Ethernet connection, while direct CMOS-signal outputs bypass the network interface. Additionally, the camera's internal memory is capable of storing program code. It can operate autonomously without the need for a PC connection and transmit output data to an Ethernet-connected or directly controlled device.

1.5.2 Applications

Line detection: Santti et al. [98] proposed a line detection method by combining KOVA1 and FPGA for low-level and mid-level feature extraction respectively. This line detection method can be applied for industrial inspection and control applications with its performance of high-speed processing and low power consumption.

Seam and Spatter Tracking: In Lahdenoja et al. [24, 99] and Santti et al.'s work [100], KOVA1 is utilised for seam tracking for real-time robot path optimisation during a high power laser welding process. The reason to use this type of pixel-level sensor lies in the ability to control pixel-wise exposure periods facing significant intensity differences in a laser welding task. Hence, the effective binary feature points can be extracted on the focal plane and then this compressed information is sent to the FPGA for laser beam location extraction using Hough transform. With a frame rate over 1000 fps, the combination of in-sensor image pre-processing and FPGA-based Hough operation enables a real-time optical seam tracking for robot control. In addition, spatter can also be tracked in laser and manual arc welding [101] in extreme radiated light intensity conditions (Fig. 1.19).

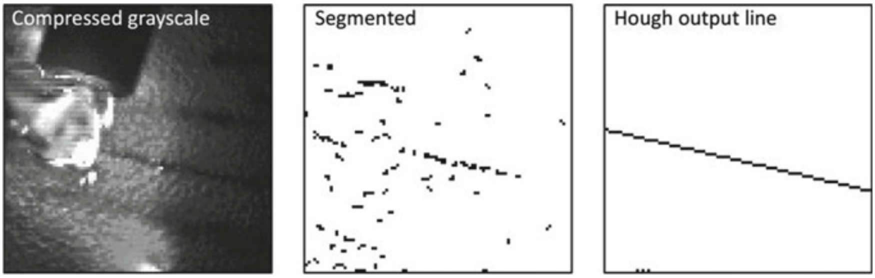


Fig. 1.19 Straight line extraction process using KOVA1. Left to right: figure captured with KOVA1, binary feature extraction using pixel-level process, estimated beam line with binary features using FPGA. Figure from [100]

1.6 Aistorm Mantis2

Mantis system is based on the event-driven charge domain for analogue signal processing without digitisation and provides an “always on” solution for analogue signal processing. One of the key features claimed by Aistorm is the noise cancelling techniques associated with the analogue signals. In addition, artificial intelligence can also be integrated into a chip for various applications. A 96 × 96 AI-in-Sensor machine learning SoC that is particularly well suited for classification jobs is the AIS-C100A Mants. It has a robust post wake-up ML capabilities as well as a configurable “always on” wake-up CNN engine that can be used to activate external micro-controllers when an object of interest is recognised. On a single monolithic device with the fewest possible external components, all necessary supporting circuitry is provided, including power management, timing, artificial intelligence, and communications in addition to a (up to) 40 mA LED driver enabling both linear and PWM control. An SPI port is used for communication. The on-board camera’s photos and videos as well as those transferred via the SPI connection can both be processed by Mantis. To provide the best contrast for AI calculations, the exposure time can be either internally or externally regulated. The AIS-C100A is housed in an OLGA package of 6.4 × 6.4 mm. The AI-in-Imager solutions that can directly take pixel data in its native charge form are AIStorm’s Mantis Family of AI-in-Imager processors. The end result is the only method in the world that can wake up a person, face, object, or action based on an image (Fig. 1.20).

There are several businesses that provide analogue AI solutions, but fundamental physical noise and bandwidth constraints prevent these products from being successful. The approach used by AIStorm is charge domain processing. This technology uses charge to create AI-in-Sensor processing opportunities for picture or audio improvement that simply cannot be realised through any other means. Other analogue methods’ noise and bandwidth restrictions are solved through charge domain processing, a revolution in processing. Over both analogue and digital solutions, charge domain solutions are preferable because they can immediately take IoT sensor data without the expense, power, or delay of digitization. The major applications proposed using the Mantis AI-in-Sensor chip cover motion tracking, gesture clas-

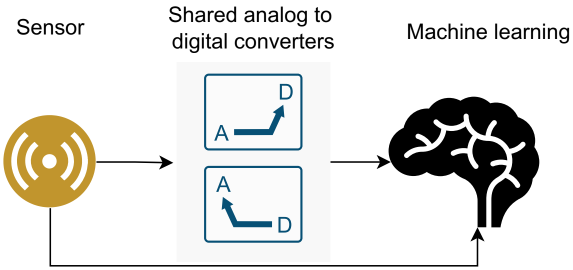


Fig. 1.20 AIS technology SKIPS digitization and moves directly to processing

sification, smart home, wearable devices (glasses, headsets), household appliances, gaming devices, automotive, and voice processes as claimed in their application website https://aistorm.ai/applications/.

1.7 Other In-Sensor Computing Devices

MIPA4k: MIPA4k [25] is an earlier sensory-level image processing sensor with similar architecture to KOVA1. A number of algorithms can be carried out based on the MIPA4k with the use of mixed signals. These algorithms mainly cover edge detection [102], local binary pattern [103], locally adaptive image sensing [104], space-dependent binary image processing [105], and object segmentation and tracking [106]. Similar early cellular vision chips also include ACE16K [107].

Memristor-Based Devices: Yao et al. [108] Memristor-based hardware is a platform to deploy the neural network using the programmable resistance within the integrated circuits mimicking the synaptic connections in a human brain [17, 108–111]. However, it integrates only storage and processing functions, which can be regarded as in-memory computing. Hence, signals should be input from sensors or other storage devices. They are thus usually integrated with other sensory systems for information processing. Lee et al. [19] take advantage of photo-diode and memristor crossbar for primary visual information process aiming to extract useful information from the input images. In-Sensor visual computing with memristor or new materials are currently not mature enough to support various practical applications for machine vision tasks. By far, the is few practical applications using memristor in-sensor computing devices.

Dynamic Vision Sensor (DVS): inivation [26] DVS produces data in the form of sparse contrast-change events that facilitate low-latency visual processing using external computational hardware [112–114]. These binary events are generated from in-sensor processing according to the brightness changes. Although the pixels in a DVS have a primitive in-sensor processing ability by binarising brightness changes, it achieves an ultra-high-speed response to the environment when working with external hardware computing units, enabling an enormous potential for robotics and computer vision in a challenging environment [115].

Other Emerging Sensor Devices: Mennel et al. [14] use a 2D semiconductor (WSe_2) photodiode array as the vision sensor, the photoresponsivity matrix to store the connecting weights of the neural network, where both supervised and unsupervised learning for classification are present. However, laser light and a set of optical systems are needed to project images onto the chip, which prevents it from having practical usage. Song et al. [116] proposed a CMOS-based PIP (Processing-in-Pixel) architecture where image convolution (8-bit weight configuration) can be performed as the image preprocessing before image data is read out. In addition, Datta et al. proposed the Processing-in-Pixel-in-memory paradigm where the first few convolutional layers of a CNN can be processed and the compressed data then sent to other near-sensor processors [117].

1.8 Conclusion

In-sensor visual computing is an emerging technology that enables efficient extraction of information solely on the sensing chip of the sensory system, facilitating signal collection, storage, and processing. Vision is the primary sense for human beings, and it accounts for a vast majority of information acquisition worldwide. Despite significant advances in computer vision using conventional camera vision systems and associated methodologies, various limitations, such as system latency and power consumption, persist. Image digitisation, storage, and transmission introduce latency, which is a bottleneck that impedes conventional machine vision systems from promptly responding to environmental changes. Additionally, traditional machine vision systems with loosely integrated sensors and processors have high energy costs, weight, and size, making them unsuitable for portable tasks.

To address the limitations of conventional machine vision systems, this chapter explores a new visual information processing scheme using a focal plane sensor processor (FPSP) that directly processes signals where they are collected, thereby avoiding issues with latency, power consumption, and size. The chapter focuses on mobile robotic control systems that utilize in-sensor computed results for multiple navigation research, and investigates novel parallel visual inference approaches, particularly machine learning-based algorithms, to extract higher-level information from analogue signals. To achieve this, a lightweight and high-speed binary convolutional neural network is proposed to categorize objects using efficient addition/subtraction and bit shifting operations. However, implementing neural networks on the focal plane is challenging due to hardware resource constraints and analogue noises. Therefore, this work proposes purely binarised convolutional neural networks with both binary weights and activations, trained with batch normalisation and adaptive threshold to binarise activations and alleviate noise. With this approach, only a small amount of extracted information is obtained, allowing for more efficient data transmission with less bandwidth, and enabling the establishment of an edge computing platform based on the neural network and PPA.

Furthermore, prior research has explored visual sensors for information collection but not for signal processing or motion control; however, this chapter investigates the direction of image processing on the sensing chip and servo motor control using the sensor's digested data directly. Thus, by merging in-sensor neural network inference and direct servo motor control, a sensory-motor system is presented. Moreover, with our proposed dynamic model swapping scheme, more sophisticated classification tasks than earlier work can be achieved. Lastly, a new in-sensor neural network architecture, fully convolutional neural networks, is presented for localisation and coarse segmentation tasks without using the fully connected layers. To deploy this new architecture of a three-layer neural network on the sensor, group convolution is introduced and implemented, with both binary weights and activations, making the fully convolutional neural network compact enough to be embedded on the sensor.

Acknowledgements This work is funded by National Natural Science Foundation of China (grant 62202285), EPSRC FARSCOPE CDT for future robotics and autonomous system, and China Scholarship Council (grant 201700260083).

References

1. Frisby JP, Stone JV (2010) Seeing: the computational approach to biological vision. MIT Press, Cambridge, MA
2. Lao J, Yan M, Tian B, Jiang C, Luo C, Xie Z, Zhu Q, Bao Z, Zhong N, Tang X et al (2022) Ultralow-power machine vision with self-powered sensor reservoir. Adv Sci 2106092
3. Wan T, Shao B, Ma S, Zhou Y, Li Q, Chai Y (2022) In-sensor computing: materials, devices, and integration technologies. Adv Mater 2203830
4. Interim report for the decadal plan for semiconductors (2020) Semiconductor research corporation and semiconductor industry association, pp 1–21
5. Pan W, Zheng J, Wang L, Luo Y (2022) A future perspective on in-sensor computing. Engineering
6. Zhou F, Chai Y (2020) Near-sensor and in-sensor computing. Nat Electr 3(11):664–671
7. Ielmini D, Wong H-SP (2018) In-memory computing with resistive switching devices. Nat Electr 1(6):333–343
8. Kaneda M (2013) Signal processing in the mammalian retina. J Nippon Med School 80(1):16–24
9. Zarándy Á (2011) Focal-plane sensor-processor chips. Springer Science & Business Media, New York City
10. Rodriguez-Vázquez Á, Dominguez-Castro R, Jiménez-Garrido F, Morillas S, Listán J, Alba L, Utrera C, Espejo S, Romay R (2008) The eye-ris cmos vision system. Analog circuit design. Springer, New York City, pp 15–32
11. https://kovilta.fi/technology/
12. Aistorm, Ai in sensor. Accessed 22 Nov 2021. https://aistorm.ai/
13. Zhang J, Dai S, Zhao Y, Zhang J, Huang J (2020) Recent progress in photonic synapses for neuromorphic systems. Adv Intell Syst 2(3):1900136
14. Mennel L, Symonowicz J, Wachter S, Polyushkin DK, Molina-Mendoza AJ, Mueller T (2020) Ultrafast machine vision with 2d material neural network image sensors. Nature 579(7797):62–66
15. Chen J, Carey SJ, Dudek P (2018) Scamp5d vision system and development framework. In: Proceedings of the 12th international conference on distributed smart cameras, pp 1–2
16. Dudek P, Hicks PJ (1999) An simd array of analogue microprocessors for early vision. In: Proceedings of the conference on postgraduate research in electronics, photonics and related fields (PREP'99), pp 359–362
17. Wang Y, Gong Y, Yang L, Xiong Z, Lv Z, Xing X, Zhou Y, Zhang B, Su C, Liao Q et al (2021) Mxene-zno memristor for multimodal in-sensor computing. Adv Funct Mater 31(21):2100144
18. Lyapunov N, Zheng XD, Yang K, Liu HM, Zhou K, Cai SH, Ho TL, Suen CH, Yang M, Zhao J et al (2022) A bifunctional memristor enables multiple neuromorphic computing applications. Adv Electr Mater 8(7):2101235
19. Lee D, Park M, Baek Y, Bae B, Heo J, Lee K (2022) In-sensor image memorization and encoding via optical neurons for bio-stimulus domain reduction towards visual cognitive processing
20. Pinkham R, Berkovich A, Zhang Z (2021) Near-sensor distributed dnn processing for augmented and virtual reality. IEEE J Emerg Selected Topics Circuits Syst 11(4):663–676
21. Sony, Intelligent vision sensors with ai processing functionality. Accessed 14 May 2020. https://www.sony.com/en/SonyInfo/News/Press/202005/20-037E/

22. Rodríguez-Vázquez A, Domínguez-Castro R, Jiménez-Garrido F, Morillas S (2010) A cmos vision system on-chip with multicore sensory processing architecture for image analysis above 1,000 f/s. In: Sensors, cameras, and systems for industrial/scientific applications XI, vol 7536. Spie, pp 213–223
23. Carey SJ, Lopich A, Barr DR, Wang B, Dudek P (2013) A 100,000 fps vision sensor with embedded 535gops, w 256× 256 simd processor array. In: 2013 symposium on VLSI circuits. IEEE, pp C182–C183
24. Lahdenoja O, Säntti T, Laiho M, Paasio A, Poikonen JK (2014) Seam tracking with adaptive image capture for fine-tuning of a high power laser welding process. In: Seventh international conference on machine vision (ICMV 2014), vol 9445. SPIE, pp 374–380
25. Poikonen J, Laiho M, Paasio A (2009) Mipa4k: a 64× 64 cell mixed-mode image processor array. In: 2009 IEEE international symposium on circuits and systems. IEEE, pp 1927–1930
26. inivation (2021) Dynamic vision platform. https://inivation.com/dvp/
27. Liu Y, Bose L, Greatwood C, Chen J, Fan R, Richardson T, Carey SJ, Dudek P, Mayol-Cuevas W (2021) Agile reactive navigation for a non-holonomic mobile robot using a pixel processor array. IET Image Process 1–10
28. Greatwood C, Bose L, Richardson T, Mayol-Cuevas W, Chen J, Carey SJ, Dudek P (2017) Tracking control of a uav with a parallel visual processor. In: 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 4248–4254
29. Greatwood C, Bose L, Richardson T, Mayol-Cuevas W, Chen J, Carey SJ, Dudek P (2018) Perspective correcting visual odometry for agile mavs using a pixel processor array. In: 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 987–994
30. McConville A, Bose L, Clarke R, Mayol-Cuevas W, Chen J, Greatwood C, Carey S, Dudek P, Richardson T (2020) Visual odometry using pixel processor arrays for unmanned aerial systems in gps denied environments. Front Robot AI 7
31. Martel JN, Mueller LK, Carey SJ, Dudek P, Wetzstein G (2020) Neural sensors: learning pixel exposures for hdr imaging and video compressive sensing with programmable sensors. IEEE Trans Pattern Anal Mach Intell 42(7):1642–1653
32. Bose L, Chen J, Carey SJ, Dudek P, Mayol-Cuevas W (2017) Visual odometry for pixel processor arrays. In: Proceedings of the IEEE international conference on computer vision, pp 4604–4612
33. Chen J, Carey SJ, Dudek P (2017) Feature extraction using a portable vision system. In: IEEE/RSJ international conference intelligent and robotic systems, Workshop vision-based Agile Auton. Navigation UAVs
34. Chua LO, Yang L (1988) Cellular neural networks: theory. IEEE Trans Circuits Syst 35(10):1257–1272
35. Roska T, Chua LO (1993) The cnn universal machine: an analogic array computer. IEEE Trans Circuits Syst II: Analog Digit Signal Process 40(3):163–173
36. Orovas C (2000) Cellular associative neural networks for pattern recognition. PhD dissertation, Citeseer
37. Rosin PL (2006) Training cellular automata for image processing. IEEE Trans Image Process 15(7):2076–2087
38. Torralba AB (1999) Analogue architectures for vision cellular neural networks and neuromorphic circuits. Doctorat thesis, Institute national Polytechnique Grenoble, Laboratory of Images and Signals
39. Dudek P (2019) Scamp-5: vision sensor with pixel parallel simd processor array. https://youtu.be/D3VcmkQiPR4
40. Liu Y, Bose L, Chen J, Carey SJ, Dudek P, Mayol-Cuevas W (2020) High-speed light-weight cnn inference via strided convolutions on a pixel processor array. In: The 31st British machine vision conference (BMVC 2020)
41. Carey SJ, Barr DR, Dudek P (2013) Low power high-performance smart camera system based on scamp vision sensor. J Syst Archit 59(10):889–899

42. Bose L, Chen J, Carey SJ, Dudek P, Mayol-Cuevas W (2019) A camera that cnns: towards embedded neural networks on pixel processor arrays. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 1335–1344
43. Debrunner T, Saeedi S, Kelly PH (2019) Auke: automatic kernel code generation for an analogue simd focal-plane sensor-processor array. ACM Trans Archit Code Optim (TACO) 15(4):1–26
44. Malekzadeh M, Clegg RG, Cavallaro A, Haddadi H (2020) Privacy and utility preserving sensor-data transformations. Pervasive Mobile Comput 63:101132. https://www.sciencedirect.com/science/article/pii/S1574119220300201
45. Lin W, Liang C, Wang JZ, Buyya R (2014) Bandwidth-aware divisible task scheduling for cloud computing. Softw: Pract Exp 44(2):163–174
46. Wang B, Dudek P (2013) Amber: adapting multi-resolution background extractor. In: 2013 IEEE international conference on image processing. IEEE, pp 3417–3421
47. Wang B, Dudek P (2014) A fast self-tuning background subtraction algorithm. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 395–398
48. Dudek P (2006) Adaptive sensing and image processing with a general-purpose pixel-parallel sensor, processor array integrated circuit. In: 2006 International workshop on computer architecture for machine perception and sensing. IEEE, pp 1–6
49. Dudek P, Vilariño DL (2006) A cellular active contours algorithm based on region evolution. In: 2006 10th international workshop on cellular neural networks and their applications. IEEE, pp 1–6
50. Alonso-Montes C, Vilarino D, Penedo M (2005) Cnn-based automatic retinal vascular tree extraction. In: 2005 9th international workshop on cellular neural networks and their applications. IEEE, pp 61–64
51. Wang B, Mroszczyk P, Dudek P (2014) A new method for fast skeletonization of binary images on cellular processor arrays. In: 2014 14th international workshop on cellular nanoscale networks and their applications (CNNA). IEEE, pp 1–2
52. Mroszczyk P, Dudek P (2012) Trigger-wave collision detecting asynchronous cellular logic array for fast image skeletonization. In: 2012 IEEE international symposium on circuits and systems (ISCAS). IEEE, pp 2653–2656
53. Razmjooei S, Dudek P (2010) Approximating euclidean distance transform with simple operations in cellular processor arrays. In: 2010 12th international workshop on cellular nanoscale networks and their applications (CNNA 2010). IEEE, pp 1–5
54. Martel JN, Müller LK, Carey SJ, Dudek P (2016) Parallel hdr tone mapping and auto-focus on a cellular processor array vision chip. In: 2016 IEEE international symposium on circuits and systems (ISCAS). IEEE, pp 1430–1433
55. Martel JN (2019) Unconventional processing with unconventional visual sensing: parallel, distributed and event based vision algorithms & systems. PhD dissertation, ETH Zurich
56. Carey SJ, Barr DR, Wang B, Lopich A, Dudek P (2012) Locating high speed multiple objects using a scamp-5 vision-chip. In: 2012 13th international workshop on cellular nanoscale networks and their applications. IEEE, pp 1–2
57. Barr DR, Carey SJ, Dudek P (2012) Low power multiple object tracking and counting using a scamp cellular processor array. In: 2012 13th international workshop on cellular nanoscale networks and their applications. IEEE, pp 1–2
58. Bose L, Dudek P, Chen J, Carey SJ, Mayol-Cuevas WW (2020) Fully embedding fast convolutional networks on pixel processor arrays. In: European conference on computer vision. Springer, pp 488–503
59. Liu Y, Bose L, Lu Y, Dudek P, Mayol-Cuevas W (2022) On-sensor binarized fully convolutional neural network with a pixel processor array. arXiv:2202.00836
60. Chen J, Liu Y, Carey SJ, Dudek P (2020) Proximity estimation using vision features computed on sensor. In: 2020 IEEE international conference on robotics and automation (ICRA). IEEE, pp 2689–2695
61. Wong MZ, Guillard B, Murai R, Saeedi S, Kelly PH (2020) Analognet: convolutional neural network inference on analog focal plane sensor processors. arXiv:2006.01765

62. Martel JN, Müller LK, Carey SJ, Müller J, Sandamirskaya Y, Dudek P (2017) Real-time depth from focus on a programmable focal plane processor. IEEE Trans Circuits Syst I: Regul Papers 65(3):925–934
63. Martel JN, Müller LK, Carey SJ, Dudek P (2017) High-speed depth from focus on a programmable vision chip using a focus tunable lens. In: 2017 IEEE international symposium on circuits and systems (ISCAS). IEEE, pp 1–4
64. Martel JN, Müller LK, Carey SJ, Müller J, Sandamirskaya Y, Dudek P (2017) Live demonstration: depth from focus on a focal plane processor using a focus tunable liquid lens. In: 2017 IEEE international symposium on circuits and systems (ISCAS). IEEE, pp 1–1
65. Murai R, Saeedi S, Kelly PH (2020) Bit-vo: visual odometry at 300 fps using binary features from the focal plane. arXiv:2004.11186
66. Stow E, Murai R, Saeedi S, Kelly PH (2021) Cain: automatic code generation for simultaneous convolutional kernels on focal-plane sensor-processors. arXiv:2101.08715
67. Wang B, Dudek P (2012) Coarse grain mapping method for image processing on fine grain cellular processor arrays. In: 2012 13th international workshop on cellular nanoscale networks and their applications. IEEE, pp 1–6
68. Martel JN, Sandamirskaya Y, Dudek P (2016) A demonstration of tracking using dynamic neural fields on a programmable vision chip. In: Proceedings of the 10th international conference on distributed smart camera, pp 212–213
69. Dudek P, Carey S (2006) General-purpose 128/spl times/128 simd processor array with integrated image sensor. Electr Lett 42(12):678–679
70. Martel JN, Müller LK, Carey SJ, Dudek P (2016) A real-time high dynamic range vision system with tone mapping for automotive applications. In: CNNA 2016; 15th international workshop on cellular nanoscale networks and their applications. VDE, pp 1–2
71. So HM, Martel JN, Wetzstein G, Dudek P (2022) Mantissacam: learning snapshot high-dynamic-range imaging with perceptually-based in-pixel irradiance encoding. In: 2022 IEEE international conference on computational photography (ICCP). IEEE, pp 1–12
72. Dudek P, Hülse M, Barr DR (2008) Cellular automata and non-static image processing for embodied robot systems on a massively parallel processor array. In: Automata-2008: theory and applications of cellular automata. Luniver Press, pp 504–510
73. Davison AJ (2018) Futuremapping: the computational structure of spatial ai systems. arXiv:1803.11288
74. Greatwood C, Bose L, Richardson T, Mayol-Cuevas W, Clarke R, Chen J, Carey SJ, Dudek P (2019) Towards drone racing with a pixel processor array. In: 11th international micro air vehicles, conferences and competitions, pp 73–79
75. Castillo-Elizalde H, Liu Y, Bose L, Mayol-Cuevas W (2021) Weighted node mapping and localisation on a pixel processor array. In: 2021 IEEE international conference on robotics and automation (ICRA). IEEE
76. Debrunner T, Saeedi S, Bose L, Davison AJ, Kelly PH (2019) Camera tracking on focal-plane sensor-processor arrays. In: Proceedings of the workshop on programmability and architectures for heterogeneous multicores (MULTIPROG), Vancouver, BC, Canada, vol 15
77. Klein G, Murray D (2007) Parallel tracking and mapping for small ar workspaces. In: 2007 6th IEEE and ACM international symposium on mixed and augmented reality. IEEE, pp 225–234
78. Liu Y, Bose L, Chen J, Fan R, Dudek P, Mayol-Cuevas W (2021) On-sensor cnn parallel computing with a pixel processor array. IEEE Trans Comput Imaging, manuscript
79. Bose L, Chen J, Carey SJ, Dudek P, Mayol-Cuevas W (2019) A camera that cnns: towards embedded neural networks on pixel processor arrays. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV)
80. Courbariaux M, Bengio Y, David J-P (2015) Binaryconnect: training deep neural networks with binary weights during propagations. In: Advances in neural information processing systems, pp 3123–3131
81. Liu Y, Chen J, Bose L, Dudek P, Mayol-Cuevas W (2021) Direct servo control from in-sensor cnn inference with a pixel processor array. In: 2021 IEEE international conference on robotics and automation (ICRA) Workshop: on and near-sensor vision processing, from photons to applications. IEEE

82. Liu Y, Chen J, Bose L, Dudek P, Mayol-Cuevas W (2021) Bringing a robot simulator to the scamp vision system. In: 2021 IEEE international conference on robotics and automation (ICRA) workshop: on and near-sensor vision processing, from photons to applications. IEEE
83. Guillard B (2019) Optimising convolutional neural networks for super fast inference on focal-plane sensor-processor arrays. PhD dissertation, Imperial College London
84. Wong M, Saeedi S, Kelly PH (2018) Analog vision-neural network inference acceleration using analog simd computation in the focal plane. PhD dissertation, Master's thesis, Imperial College London-Department of Computing
85. Liu Y (2022) On-sensor visual inference with a pixel processor array. PhD dissertation, University of Bristol
86. Wolfram S (1984) Cellular automata as models of complexity. Nature 311(5985):419–424
87. Mordvintsev A, Randazzo E, Niklasson E, Levin M (2020) Growing neural cellular automata. Distill 5(2):e23
88. Eye-RIS, Eye-ris v1.3 hardware description. Accessed 16 Sept 2022. https://imaging.teledyne-e2v.com/products/2d-cmos-image-sensors/eye-ris/
89. Sotoa LA, Morillasa S, Listána J, Jiméneza A, Arenab P, Patanéb L, De Fioreb S, Embedding the anafocus' eye-ris vision system in roving robots to enhance the action-oriented perception
90. Caballero-Garcia D, Jimenez-Marrufo A (2014) Visual routines for cognitive systems on the eye-ris platform. In: Spatial temporal patterns for action-oriented perception in roving robots II. Springer, pp 249–316
91. Soto LA, Morillas S, Listán J, Jiménez A, Arena P, Patané L, De Fiore S (2009) Embedding the anafocus' eye-ris vision system in roving robots to enhance the action-oriented perception. In: Bioengineered and bioinspired systems IV, vol 7365. SPIE, pp 80–90
92. Guzmán P, Díaz J, Agís R, Ros E (2010) Optical flow in a smart sensor based on hybrid analog-digital architecture. Sensors 10(4):2975–2994
93. Lucas BD, Kanade T et al (1981) An iterative image registration technique with an application to stereo vision. Vancouver, vol 81
94. Zarándy Á, Pencz B, Németh M, Zsedrovits T (2014) Implementation of visual navigation algorithms on the eye-ris 1.3 system. In: 2014 14th international workshop on cellular nanoscale networks and their applications (CNNA). IEEE, pp 1–2
95. Nicolosi L, Abt F, Tetzlaff R, Hofler H, Blug A, Carl D (2009) New cnn based algorithms for the full penetration hole extraction in laser welding processes. In: 2009 IEEE international symposium on circuits and systems. IEEE, pp 2713–2716
96. Abt F, Heider A, Weber R, Graf T, Blug A, Carl D, Höfler H, Nicolosi L, Tetzlaff R (2011) Camera based closed loop control for partial penetration welding of overlap joints. Phys Procedia 12:730–738
97. Fülöp T, Zarándy Á (2010) Bio-inspired looming object detector algorithm on the eye-ris focal plane-processor system. In: 2010 12th international workshop on cellular nanoscale networks and their applications (CNNA). IEEE, pp 1–5
98. Säntti T, Lahdenoja O, Paasio A, Laiho M, Poikonen J (2014) Line detection on fpga with parallel sensor-level segmentation. In: 2014 14th international workshop on cellular nanoscale networks and their applications (CNNA). IEEE, pp 1–2
99. Lahdenoja O, Säntti T, Poikonen J, Laiho M, Paasio A (2013) Characterizing spatters in laser welding of thick steel using motion flow analysis. In: Scandinavian conference on image analysis. Springer, pp 675–686
100. Säntti T, Poikonen JK, Lahdenoja O, Laiho M, Paasio A (2015) Online seam tracking for laser welding with a vision chip and fpga enabled camera system. In: 2015 IEEE international symposium on circuits and systems (ISCAS). IEEE, pp 1985–1988
101. Lahdenoja O, Säntti T, Laiho M, Poikonen J (2014) Spatter tracking in laser-and manual arc welding with sensor-level pre-processing
102. Poikonen J, Laiho M, Paasio A (2010) Anisotropie filtering with a resistive fuse network on the mipa4k processor array. In: 2010 12th international workshop on cellular nanoscale networks and their applications (CNNA). IEEE, pp 1–5

103. Lahdenoja O, Poikonen J, Laiho M (2010) Extracting local binary patterns with mipa4k vision processor. In: 2010 12th international workshop on cellular nanoscale networks and their applications (CNNA). IEEE, pp 1–5
104. Poikonen J, Laiho M, Paasio A (2009) Locally adaptive image sensing with the 64x64 cell mipa4k mixed-mode image processor array. In: 2009 European conference on circuit theory and design. IEEE, pp 93–96
105. Laiho M, Poikonen J, Paasio A (2009) Space-dependent binary image processing within a 64x64 mixed-mode array processor. In: 2009 European conference on circuit theory and design. IEEE, pp 189–192
106. Laiho M, Poikonen J, Paasio A (2010) Object segmentation and tracking with asynchronous grayscale and binary wave operations on the mipa4k. In: 2010 12th international workshop on cellular nanoscale Networks and their applications (CNNA). IEEE, pp 1–4
107. Linan G, Espejo S, Dominguez-Castro R, Rodriguez-Vázquez A (2002) Architectural and basic circuit considerations for a flexible 128× 128 mixed-signal simd vision chip. Analog Integr Signal Process 33(2):179–190
108. Yao P, Wu H, Gao B, Tang J, Zhang Q, Zhang W, Yang JJ, Qian H (2020) Fully hardware-implemented memristor convolutional neural network. Nature 577(7792):641–646
109. Wang T-Y, Meng J-L, Li Q-X, He Z-Y, Zhu H, Ji L, Sun Q-Q, Chen L, Zhang DW (2021) Reconfigurable optoelectronic memristor for in-sensor computing applications. Nano Energy 89:106291
110. Sun L, Wang Z, Jiang J, Kim Y, Joo B, Zheng S, Lee S, Yu WJ, Kong B-S, Yang H (2021) In-sensor reservoir computing for language learning via two-dimensional memristors. Sci Adv 7(20):eabg1455
111. Thomas A (2013) Memristor-based neural networks. J Phys D: Appl Phys 46(9):093001
112. Lungu IA, Liu S-C, Delbruck T (2019) Fast event-driven incremental learning of hand symbols. In: 2019 IEEE international conference on artificial intelligence circuits and systems (AICAS). IEEE, pp 25–28
113. Lungu I-A, Corradi F, Delbrück T (2017) Live demonstration: convolutional neural network driven by dynamic vision sensor playing roshambo. In: 2017 IEEE international symposium on circuits and systems (ISCAS). IEEE, pp 1–1
114. Linares-Barranco A, Rios-Navarro A, Tapiador-Morales R, Delbruck T (2019) Dynamic vision sensor integration on fpga-based cnn accelerators for high-speed visual classification. arXiv:1905.07419
115. Gallego G, Delbruck T, Orchard G, Bartolozzi C, Taba B, Censi A, Leutenegger S, Davison A, Conradt J, Daniilidis K et al (2019) Event-based vision: a survey. arXiv:1904.08405
116. Song R, Huang K, Wang Z, Shen H (2021) An ultra fast low power convolutional neural network image sensor with pixel-level computing. arXiv:2101.03308
117. Datta G, Kundu S, Yin Z, Lakkireddy RT, Beerel PA, Jacob AP, Jaiswal A (2022) P2m: a processing-in-pixel-in-memory paradigm for resource-constrained tinyml applications. arXiv:abs/2203.04737

Chapter 2
Environmental Perception Using Fish-Eye Cameras for Autonomous Driving

Yeqiang Qian, Ming Yang, and John M. Dolan

Abstract The fish-eye camera has become an indispensable sensor in autonomous driving. Due to the unique projection principle of the fish-eye camera, it provides a large field of view (FoV). Because of this special feature, the fish-eye camera has abundant applications in environmental perception and autonomous driving. However, many challenges still exist in the practical application of fish-eye cameras. In this chapter, typical fish-eye datasets, including real data and generated virtual data, are introduced. Then, the projection principle of the fish-eye camera and four classical fish-eye image representation models are given. By sorting and summarizing relevant research, we show various applications of fish-eye cameras in environmental perception, including semantic understanding and target detection. These works have designed various strategies to take advantage of fish-eye cameras and prevent image distortions, showing the broad application prospects of fish-eye cameras. The development trend of fish-eye cameras in autonomous driving is discussed.

2.1 Introduction

In the past decade, autonomous driving has rapidly developed. Both academia and industry are paying attention to this field. Self-driving cars are equipped with a variety of sensors, such as cameras, lidar, and radar. Cameras provide more semantic information than other sensors. The fish-eye camera is a unique type of camera that

Y. Qian
University of Michigan-Shanghai Jiao Tong University Joint Institute, Shanghai Jiao Tong University, Shanghai 200240, China
e-mail: qianyeqiang@sjtu.edu.cn

M. Yang (✉)
Key Laboratory of System Control and Information Processing, Department of Automation, Shanghai Jiao Tong University, Ministry of Education of China, Shanghai 200240, China
e-mail: mingyang@sjtu.edu.cn

J. M. Dolan
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
e-mail: jdolan@andrew.cmu.edu

R. Fan et al. (eds.), *Autonomous Driving Perception*, Advances in Computer Vision and Pattern Recognition, https://doi.org/10.1007/978-981-99-4287-9_2

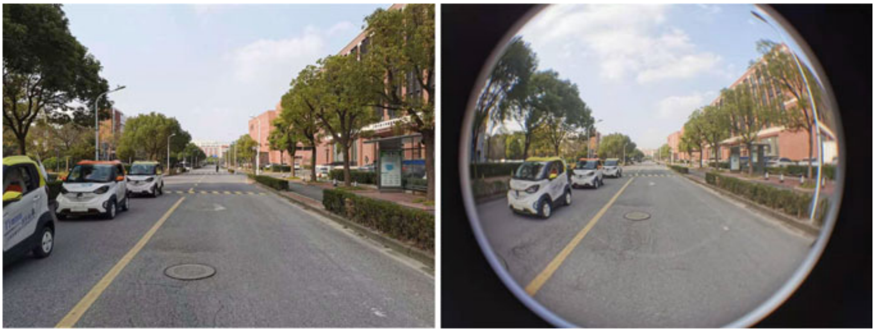


Fig. 2.1 Typical normal and fish-eye images taken in the same location

provides a larger field of view (FoV), up to 180°, than ordinary cameras. Therefore, fewer fish-eye cameras are needed to fully monitor their surroundings. Moreover, fish-eye cameras are indispensable sensors in autonomous driving and have a very large role in some scenarios.

Figure 2.1 shows a normal image and fisheye image; both were taken at the same location. Intuitively, the images appear quite different. The fish-eye image in Fig. 2.1b contains more scenery. In addition, serious distortion occurs in the fish-eye image, e.g., the straight street lamp in Fig. 2.1a becomes crooked in the fish-eye image. The same object in a normal image contains more pixels than in a fish-eye image, that is, fish-eye images lose information as the distance increases.

In short, fish-eye cameras have three advantages. Similar to fish-eye cameras, lidar sensors also have large FoVs [1–4]. However, cameras provide much richer semantic information. In addition, cameras have a higher cost performance [5], which is crucial for the industry. Moreover, these benefits apply to all camera sensors, not just fish-eye cameras.

Fish-eye cameras offer a wide FoV, usually up to 180°, which is crucial in autonomous driving as covering more blind spots helps improve safety.

The use of fish-eye cameras avoids the need for calibration among multiple cameras. To cover more space, ordinary cameras should be grouped. Therefore, for such a system, calibration is necessary, and calibration between two different sensors is complex. Inaccurate calibration will introduce unexpected noise in the system, leading to the failure of subsequent algorithms.

There are three challenges in applying fish-eye cameras. Deep learning methods have substantially become more popular in the field of computer vision, including environmental perception, where datasets have become particularly important. The use of sample-rich datasets contributes to the better performances of most algorithms. In recent years, many datasets for autonomous driving environment awareness, such as KITTI [6] and Cityscapes [7], have been published. However, these datasets contain normal images with a normal FoV. As fish-eye images and normal images appear completely different, normal images have difficulty benefitting from fish-eye-based

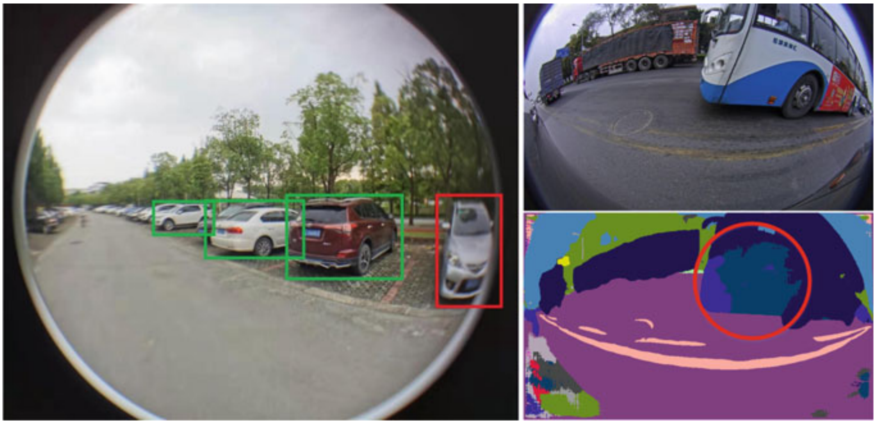


Fig. 2.2 False results caused by the distortion of the fish-eye image. The red bounding box represents false negative detection results, and the area in the red circle represents false positive segmentation results

algorithms. In conclusion, few significant fish-eye datasets have been formed, which severely limits the development of fish-eye-based algorithms.

This is the most significant problem of fish-eye cameras. In Fig. 2.1, street lights, cars and buildings appear in a completely different curved position from the real shape. Moreover, these distorted objects cause issues for most detectors as most detectors are trained on normal images. Figure 2.2a shows the object detection results obtained by using the faster region-based convolutional neural network (Faster R-CNN) [8]. Compared to other normal vehicles, the white vehicles with edges were so deformed that they were not accurately detected. Figure 2.2b shows the semantic segmentation results generated by the efficient residual-factorized CNN (ERFNet) [9, 10]. The areas in the red circles are segmentation errors caused by distortion.

As shown in Fig. 2.1, it is difficult for fish-eye cameras to clearly observe distant targets, which limits their application scenarios, that is fish-eye cameras can only be used for low-speed and short-range scenes. In these cases, the perception of distant objects is not urgently needed.

This chapter is organized as follows. The typical fish-eye image datasets are presented in Sect. 2.2. Different fish-eye projection models are introduced in Sect. 2.3. Various applications of fish-eye cameras in autonomous driving are organized in Sects. 2.4 and 2.5. Finally, the development tendency of fish-eye camera applications is discussed in Sect. 2.6.

2.2 Fish-Eye Image Datasets

High-quality, fish-eye datasets provide a comparison platform for various algorithms and can be utilized in the training process to improve the performance of the detector. Currently, three main methods are used to obtain fish-eye image data. Most works use real fish-eye cameras to manually collect real fish-eye datasets. Some works use simulators to generate virtual fish-eye images. Other works focus on generating fish-eye lens images from real perspective images. These real and virtual images promote research in this field.

2.2.1 Real Fish-Eye Datasets

We sorted the real fish-eye datasets in the field of automatic driving. Details of these datasets are shown in Table 2.1. The author uses different fish-eye camera settings to customize their fish-eye lens datasets, including different numbers of fish-eye lenses, image resolutions, numbers of datasets, and supported tasks.

Levi et al. [11, 12] collected a fish-eye dataset using a vehicle rearview camera to evaluate the application of rear-view pedestrian detection. This dataset is named the GM-ATCI rearview pedestrian dataset. This dataset contains 15 video sequences, and each session is obtained in different scenes on different dates. The dataset contains a total of 250 fragments, with a total duration of 76 min, and more than 200 K annotated pedestrian bounding boxes.

Oxford RobotCar [13], which was proposed by Maddern et al., is a large-scale dataset that focuses on the long-term autonomy of autonomous vehicles, mainly supporting positioning and mapping tasks. The dataset collects data under various weather conditions, including heavy rain, nighttime, direct sunlight, and snowfall, and has accumulated more than 1000 km of road data in one year. Due to the long data collection time, the appearances of many roads and buildings has changed, which is also a feature of this dataset.

Yogamani et al. [14] proposed the first fish-eye image dataset dedicated to intelligent vehicles. This dataset, which is named WoodScape, consists of four cameras covering 360° and a high-definition laser scanner, inertial measurement unit (IMU), and global navigation satellite system (GNSS). Annotations can be used for nine tasks, especially 3D object detection, depth estimation (superimposed on the front camera) and semantic segmentation. The semantic annotation of 40 classes at the instance level exceeds 10000 images, and the annotation of other tasks exceeds 100000 images. Rashed et al. further expanded the dataset in [15].

Liao et al. proposed the suburban dataset KITTI-360 [16]. Compared with the classic KITTI [6], KITTI-360 has two new fish-eye cameras and a larger amount of data. Compared with WoodScape [14], KITTI-360 provides temporal, coherent, semantic instance annotation and 3D annotation for 3D reasoning.

2.2.2 Simulator-Based Virtual Fish-Eye Datasets

In Table 2.1, we sorted the virtual fish-eye dataset generated by a simulator. Zhang et al. [17] proposed a multifield dataset for fish-eye lens images. The authors propose this dataset to analyse the impact of images from different FoVs on visual odometers.

Sekkat et al. [18] proposed a framework for generating omnidirectional images using images obtained in virtual environments. This dataset is named OmniScape. CARLA and GTAV simulators are utilized in this work. The generated dataset includes semantic segmentation data, depth maps, intrinsic camera parameters, and dynamic motorcycle parameters.

Sekkat et al. proposed the SynwoodScapes dataset [19], which was developed from Woodscape. As this is a virtual dataset, many of the shortcomings of WoodScape are addressed. For example, WoodScape cannot collect pixel-level, optical flow and depth or simultaneously annotate all four cameras, which can be easily implemented in SynwoodScape. The SynwoodScapes dataset contains 80k annotated images and supports many perceptual tasks.

Table 2.1 Real and virtual datasets of fish-eye images in intelligent vehicles

Datasets	Year	Real or virtual	Image information	Supported tasks
GM-ATCI [11, 12]	2015	Real	56K frames with 1280×800 resolutions	Pedestrian detection
Oxford RobotCar [13]	2017	Real	20M frames with 1024×1024 resolutions	IMU & GNSS & lidar
WoodScape [14]	2019	Real	100K frames with 1280×966 resolutions	2D/3D object detection Semantic/instance segmentation Motion segmentation IMU & GNSS & lidar
KITTI-360 [16]	2022	Real	150K frames with 1400×1400 resolutions	2D/3D object detection IMU & GNSS & lidar
Multi-FoV [17]	2016	Virtual	2.7K frames with 640×480 resolutions	Visual odometry
OmniScape [18]	2020	Virtual	1024×1024 resolutions	2D/3D object detection Semantic/instance segmentation Depth estimation Optical flow IMU & GNSS & lidar
SynWoodScape [19]	2022	Virtual	80K frames with 1280×966 resolutions	2D/3D object detection Semantic/instance segmentation Depth estimation Motion segmentation Optical flow Event camera IMU & GNSS & lidar

2.2.3 Fish-Eye Projection Model-Based Methods

Some methods use different fish-eye image projection models to generate images, as shown in Table 2.2. These methods can save time and avoid domain adaptation problems. Most methods use equidistant models to generate fish-eye images as these models are widely applied. These methods convert a normal perspective image to a fish-eye image according to the equidistant fish-eye projection model.

Qian et al. [20, 21] used a fixed focal length. Their approach can be extended to most datasets from a normal perspective. Using this method, enough fish-eye images can be generated to train fish-eye-based detectors. Other methods [23–27] use variable parameters to optimize the fixed focal length to achieve effective data enhancement with different degrees of freedom (DoF).

2.2.4 Comparison Between Real and Virtual Datasets

Real datasets have largely driven the development of fish-eye camera-based applications. Since it is time-consuming to collect and label fish-eye images, various virtual fish-eye datasets complement available training data at a low cost. However, model transformation-based methods cannot restore the FoV of a fish-eye camera, which is disadvantageous in research related to specific tasks. The data generated by a simulator exhibit the domain offset problem. We suggest that fish-eye cameras equipped on intelligent vehicles have a large role in low-speed and short-distance scenarios, and large-scale datasets are not available for such specific scenarios, e.g., parking lots. In addition, the combined dataset of fish-eye cameras and telephoto cameras is also expected as such a combination can cover a wider range of perception space.

Table 2.2 Methods for generating fish-eye images using fish-eye projection models

Reference	Author	Year	Description	Application
[20]	Qian et al.	2017	Fixed imaging plane	Pedestrian detection
[21]	Deng et al.	2017	Two focal lengths	Semantic segmentation
[22]	Deng et al.	2018	Surround-view images	Semantic segmentation
[23]	Blott et al.	2018	Six DoFs of data augmentation	Semantic segmentation
[24]	Sáez et al.	2018	Variable focal length	Semantic segmentation
[25]	Qian et al.	2019	Three DoFs of data augmentation	Pedestrian detection
[26]	Sáez et al.	2019	Variable focal length	Semantic segmentation
[27]	Ye et al.	2020	Seven DoFs of data augmentation	Semantic segmentation

2.3 Fish-Eye Camera Projection Principle

2.3.1 Four Classic Image Representation Models

A fish-eye camera provides a large FoV because of its unique projection model. The lens of a fish-eye camera is usually combined with multiple optical lenses. The incident ray passes through various refractions before mapping into the final image plane. The refraction of the incident ray enables a limited image plane to contain more of the scene of interest. This process contributes to the large FoV of a fish-eye camera.

The lens model of a fish-eye camera is simplified using Fig. 2.3. The projection process is divided into two parts. First, any point $P1(x_c, y_c, z_c)$ in reality is mapped into the imaging plane as $P2(x_e, y_e, z_e)$. θ is the angle between the incident ray and the optical axis. The imaging plane in a fish-eye camera is a hemisphere, whereas it is a flat plane in a normal camera. The focal length f is the radius of the hemisphere. Second, $P2$ is mapped to $P3(u, v)$ in the image plane. Different mapping models generate different values of l, which is the distance between $P3$ and the image centre. θ' is the angle between the final imaging point $P3$ and the optical axis. θ' is calculated from θ by using different projection models.

Four classical image representation models are available for fish-eye cameras, as shown in Fig. 2.4. Different representation models generate different fish-eye images.

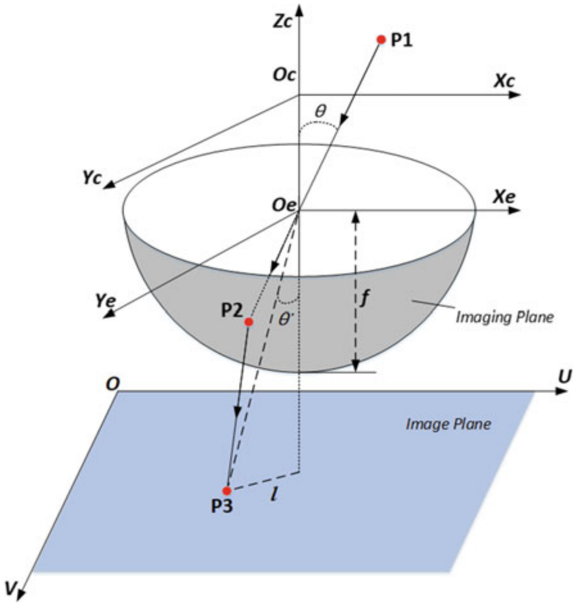


Fig. 2.3 Simplified lens model of a fish-eye camera

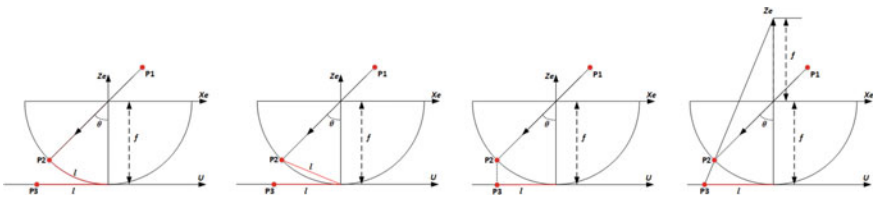


Fig. 2.4 Four image representation models of a fish-eye camera: (from left to right) equidistant, equisolid angle, erthographic and stereographic

- Equidistant:

$$l = f\theta \tag{2.1}$$

The distance l between $P3$ and the image centre is proportional to the angle θ between the incident ray and the optical axis. The proportionality coefficient is f, which is the focal length of the fish-eye camera. In this projection model, the largest FoV is 360°. This is the simplest projection model for a fish-eye camera.

- Equisolid angle:

$$l = 2f\sin(\theta/2) \tag{2.2}$$

Given the same θ, the equisolid angle model generates a shorter l than that of the equidistant model. The largest FoV of this model is also 360°.

- Orthographic:

$$l = f\sin(\theta) \tag{2.3}$$

The pixels in the final image become denser as the angle θ increases. The largest FoV of this model is 180°.

- Stereographic:

$$l = 2f\tan(\theta/2) \tag{2.4}$$

The stereographic projection model is also referred to as a conformal mapping model in geometry. The largest FoV approaches 360°.

Figure 2.5 shows a comparison of the four above projection models. The X axis is the angle θ, and the Y axis is the distance l/f. For a better comparison, the Y axis is divided by f.

We can also use a unified model [28] to approximate the four projection models described above:

$$l = fU(\theta) = f(k_1\theta^1 + k_2\theta^3 + k_3\theta^5 + k_4\theta^7 + k_5\theta^9) \tag{2.5}$$

k_{1-5} are parameters. Using this unified model as an example, the transformation from $P1$ in the space to the pixel $P3$ in the image is interpreted as:

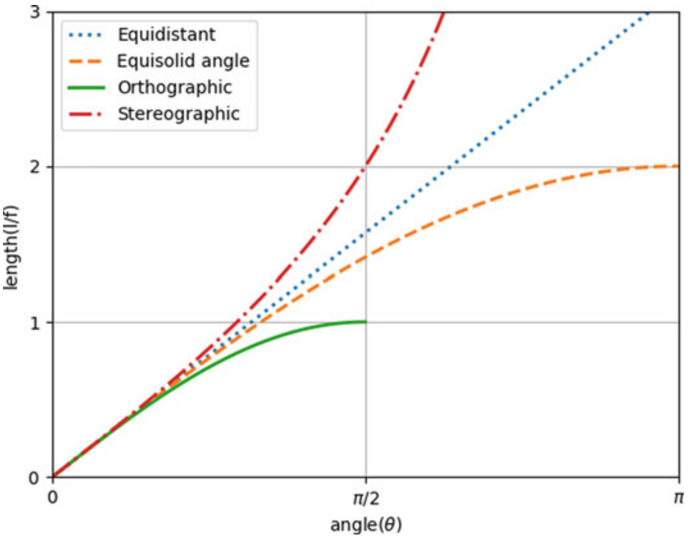


Fig. 2.5 Comparison among the four image representation models of fish-eye cameras

$$
\begin{aligned}
P2(x_e, y_e, z_e) &= \mathbf{R}\,P1(x_c, y_c, z_c) + \mathbf{T} \\
l^2 &= \left(\frac{x_e}{z_e}\right)^2 + \left(\frac{y_e}{z_e}\right)^2 \\
\theta &= \arctan(l) \\
\theta^* &= U(\theta) \\
u = \frac{f_x \theta^* x_e}{l z_e} + c_x \quad & v = \frac{f_y \theta^* y_e}{l z_e} + c_y
\end{aligned}
\tag{2.6}
$$

R and **T** are the rotation matrix and translation matrix from the space to the imaging plane, respectively. f_x and f_y are the focal lengths in different directions. They are also considered the proportionality coefficients. $P3(u, v)$ is the final pixel in the image. Since the origin of an image is not the image centre, c_x and c_y are used to translate the coordinates. According to Eqs. 2.5 and 2.6, $P1$ is easily mapped to $P3$. A final fish-eye image is thus generated.

2.3.2 Other Wide-Angle Camera Projection Principles

In addition to fish-eye cameras, other cameras, such as omnidirectional and panoramic cameras, support large-angle perception. These cameras are briefly introduced in this subsection.

The imaging steps of a wide-angle camera are similar to those of an ordinary camera. The main difference concerns the part of the lens that condenses light. To obtain a larger FoV, a wide-angle camera has three ways of condensing its lens [29].

The first projection principle is dioptric projection. The camera uses a lens to achieve imaging via the refraction of light. A typical camera is the fish-eye camera discussed in this chapter.

The second projection principle is catadioptric projection. The camera uses a standard camera and a mirror, which provides a 360-degree FoV in the horizontal plane and an FoV greater than 100° in the elevation direction. A typical camera is an omnidirectional camera [30].

The third projection principle is the polydioptric projection. The camera expands the FoV by combining multiple standard cameras. A typical camera is a panoramic camera.

2.4 Semantic Understanding

2.4.1 Semantic Segmentation in Fish-Eye Images

Semantic segmentation tasks are common in computer vision and intelligent transportation. Traditional semantic segmentation algorithms usually focus on traditional images. Because of the nonlinear distortion in fish-eye images, the traditional semantic segmentation algorithm is not competitive in fish-eye images. Therefore, many researchers have conducted corresponding studies, as shown in Table 2.3.

2.4.1.1 Adaptive Networks

Hanisch et al. [31] attempted to extract the passable region from the fish-eye image. The authors employed the mean shift segmentation algorithm to extract the superpixels of the original fish-eye image, divided the superpixels into three categories (ground, sky, and obstacle), and extracted the passable area from the ground.

Baek et al. [32] proposed a bottom net for detecting drivable areas. The network takes each vertical column of a given image as the input and classifies the pixel position of the bottom of the obstacle corresponding to that column. The union of all columns builds the drivable area or the curb area.

2.4.1.2 Data Augmentation

Blott et al. [23] tried to achieve an accurate semantic understanding of fish-eye images from another research direction. The authors utilized ordinary images to generate fish-eye images to expand their fish-eye image training set and achieved

Table 2.3 Semantic segmentation methods based on fish-eye images. $Ada.Net.$ represents an adaptive network, and $DataAug.$ represents data augmentation

Reference	Author	Year	Distortion solution	Solution description
[31]	Hanisch et al.	2017	Ada. Net.	Superpixels
[32]	Baek et al.	2018	Ada. Net.	Bottom-Net
[23]	Blott et al.	2018	Data Aug.	Rectilinear-to-fish-eye transformations
[33]	Zhou et al.	2019	Data Aug.	Skewing and gamma correction
[27]	Ye et al.	2020	Data Aug.	Seven-DOF augmentation
[21]	Deng et al.	2017	Ada. Net.+ Data Aug.	Overlapping pyramid pooling module + zoom augmentation
[24]	Saez et al.	2018	Ada. Net.+ Data Aug.	ERFNet + random flips and translations
[22]	Deng et al.	2018	Ada. Net.+ Data Aug.	Restricted deformable convolution + zoom augmentation
[26]	Saez et al.	2019	Ada. Net.+ Data Aug.	Optimized ERFNet + randomly chosen distortions
[34]	Playout et al.	2021	Ada. Net.+ Data Aug.	Adaptable deformable convolution + BlenDataset

data-enhanced effects. Their data enhancement approach consists of six distortion depth-first searches (DFSs), such as the rotation, pan, and zoom, thus effectively extending the fish-eye image training set. From the point of view of data, this method achieves good results in the semantic segmentation of fish-eye images. Ye et al. [27] followed the same idea and extended the data augmentation method to seven DOFs.

Zhou et al. [33] also explored data enhancement techniques, particularly tilt and gamma correction, from a real-world perspective to extend existing models. Their approach showed remarkable adaptability to changing lighting conditions and camera perspectives.

2.4.1.3 Adaptive Network + Data Augmentation

Deng et al. [21] proposed overlapping pyramid pooling to extract local, global, and pyramid context information to solve the complex scenes contained in language images. Based on the overlapping pyramid pool module, the researchers proposed the OPP-Net network, which achieves high-accuracy semantic segmentation for fish-eye images. Afterward, the authors proposed the restricted deformable convolution network [22] to further effectively solve the semantic segmentation problem caused by the distortion of fish-eye images.

Saez et al. [24] adapted ERFNet [9, 10] to solve the feature distortion problem of fish-eye images. They proposed a new fish-eye image dataset for semantic segmentation based on the CityScapes dataset. Based on ERFNet, researchers further proposed the ERFNetPSP network [26], which replaced the original ERFNet decoding layer with a manually designed pyramid pooling module, thereby improving the effect of semantic segmentation.

Playout et al. [34] leveraged the capabilities of deformable convolutional networks [35] to take into account nonlinear transformations by geometric fish-eye distortions. Furthermore, the authors proposed the adaptable deformable convolution, which adapts an existing convolution to the extrinsic deformations of a grid while preserving the intrinsic properties of the associated objects.

2.4.2 Semantic Segmentation in Omnidirectional or Panoramic Images

Since omnidirectional images and panoramic images support the understanding of 360-degree surroundings, their corresponding semantic segmentation methods are discussed here.

2.4.2.1 Adaptive Networks

Yang et al. [36] proposed a panoramic annular semantic segmentation (PASS) framework to perceive the entire surroundings of an image based on a compact panoramic annular lens system and an online panorama unfolding process. To consistently exploit the rich contextual cues contained in the unfolded panorama, they adapted the real-time ERFNetPSP [37] to predict semantically meaningful feature maps in different segments and fused them to achieve smooth and seamless panoramic scene parsing.

The work has been further advanced in multiple directions. In [36, 38] was extended with a detailed description of the proposed PASS framework and a PASS dataset to benchmark panoramic perception algorithms. In [39], the network architecture was replaced by the attention-connected SwaftNet. Moreover, the authors extended the PASS dataset by finely annotating more detail-critical classes, such as sidewalks and pedestrians. In [40], the authors proposed the omnisupervised, omnidirectional, semantic segmentation framework via multisource omnisupervised learning. The latest advancements include frameworks that focus on dimensionwise positional priors [41] and omnirange contextual dependencies [42].

2.4.2.2 Unsupervised Domain Adaptation

Ma et al. [43] used another idea to achieve panoramic semantic segmentation. The authors proposed an unsupervised domain adaptation method with an attention-augmented context exchange. Furthermore, a novel, densely annotated dataset, namely, DensePASS, was proposed. Zhang et al. [44] further extended this work with several domain adaptation modules, a detailed description of the DensePASS dataset, and an extended set of experiments and analyses.

2.4.3 Instance and Panoptic Segmentation

In addition to semantic segmentation, instance segmentation and panoptic segmentation have also been important image segmentation tasks in recent years. In this subsection, we briefly summarize the instance segmentation and panoptic segmentation methods that are applied to wide-angle images.

Dufour et al. [45] proposed a data augmentation method for instance segmentation in fish-eye images. Their method is similar to that of [21, 23, 24, 27]. The authors employed a Mask R-CNN [46] as the baseline and mixed fish-eye images and ordinary images for training, obtaining satisfactory segmentation results on both rectilinear and fish-eye images.

Jaus et al. [47] proposed a panoptic segmentation method for panoramic images. To overcome the lack of annotated panoramic images, the authors proposed a panoramic robust feature (PRF) framework that enables model training on standard pinhole images and transfers the learned features to a different domain. Using the proposed method, the authors managed to achieve significant improvements on their Wild Panoramic Panoptic Segmentation (WildPPS) dataset.

Petrovai et al. [48] proposed a complete process for developing a 360-degree, 2D, semantic environmental perception system using five cameras. The authors proposed deep learning-based, semantic virtual cameras that provide pixel-level, semantic, instance and panoptic information. The panoptic module yields increased segmentation accuracy and facilitates low-level fusion with 3D point clouds as part of the 3D perception system.

2.4.4 Analysis of Semantic Understanding

As a fine-grained task, image segmentation is more difficult than object detection. Thus, almost all the work is based on deep learning techniques. We discover that in semantic segmentation research involving fish-eye images, researchers try to develop a variety of advanced deep network techniques, which greatly promotes research in this field. In future research, it will be an exciting task to establish a unified semantic segmentation model for various image representation models.

2.5 Object Detection

In this section, methods of target detection and tracking using fish-eye images are introduced. This task is the most extensive use of fish-eye cameras. A single fish-eye camera can detect targets in a wider FoV, which is important for intelligent vehicle applications such as obstacle avoidance and vehicle tracking. However, the effect of severe image distortion on the detector is significant, as shown in Fig. 2.2a. Many approaches have been proposed to address this problem; they fall into two categories: the two-step approach and one-step approach.

In the two-step method, the input fish-eye image is processed into a general image representation model, thus reducing the image distortion to a certain extent. In the one-step method, the processing steps are directly performed on the original fish-eye image, and the results are directly displayed in the fish-eye image. Since the two-step method heavily relies on various image representation models, we introduce the classical image representation models that are commonly employed to correct fish-eye images.

2.5.1 Different Image Representation Models

2.5.1.1 Fish-Eye Model

The fish-eye model, which also known as the spherical model, is the main research object of this paper. Fish-eye projection involves four classical image representation models, which are discussed in Sect. 2.2. The vertical and horizontal angles of fish-eye projections are limited to 180°, and the resulting image can be placed in a circle. Therefore, the farther the line is from the centre of the image grid, the greater the curvature.

2.5.1.2 Rectilinear Model

The main advantage of rectilinear projection is that it maps all lines in three-dimensional space to lines on a two-dimensional grid. The result of a rectilinear projection is an image with a normal perspective, as shown in Fig. 2.1b. As rectilinear projection can produce a scene similar to a normal perspective image, it is widely utilized to correct fish-eye images.

The main drawback of rectilinear projection is that the perspective is greatly enhanced as the angle of view increases, which causes the objects at the edges of the resulting image to tilt. When using linear projection to correct a fish-eye image, the information at the edges of the image must be relinquished.

2.5.1.3 Cylindrical Model

Cylindrical projection can produce a complete, horizontal FoV of 180°, holding more information than linear projection. In addition, as cylindrical projection is more accurate than rectilinear projection in maintaining the target size, it is also widely employed to correct fish-eye images when a larger horizontal part of the scene needs to be maintained. However, cylindrical projection renders straight lines parallel to the line of sight of the observer as curves. Moreover, as the target approaches the north and south poles, the poles infinitely extend.

2.5.1.4 Equirectangular Model

An equirectangular projection is similar to a cylindrical projection. Equirectangular projection directly converts the longitude and latitude coordinates of a sphere into a grid of horizontal and vertical coordinates. Thus, equirectangular projection can achieve a 180-degree FoV for a given scene in the horizontal and vertical directions. However, significant distortion occurs in the vertical direction.

2.5.1.5 Mercator Model

The Mercator projection is closely related to cylindrical and isorectangular projections as it is a compromise between them. Compared to a cylindrical projection, the Mercator projection produces a minor vertical stretch and a larger available vertical angle, but its lines are more curved. This projection is widely applied to generate planar graphs.

2.5.1.6 Other Models

Several other image representation models, e.g., spherical, cube-map, icosahedron, tangent, hexagonal, and longitude-latitude models, are available. The spherical model projects 360-degree horizontal and vertical FoVs onto a spherical surface, and the fish-eye camera results can be compared to a hemispherical image representation [49]. The cube-map, icosahedron and tangent models are considered different image unfolding methods for spherical images. The cube-map model simplifies a spherical image into a cube and unfolds the six faces of this cube to form a 2D view [50]. The icosahedron model is an approximation of the spherical model that uses twenty equilateral triangles to simplify a sphere. The tangent model is based on the icosahedral model and further reduces the distortion induced by the spherical model [51]. The hexagonal model has advantages over traditional square arrangements in some image processing fields [52]. The longitude-latitude model is often employed to transform spherical images and panoramic images [53].

2.5.2 Two-Step Methods

In two-step methods, first, fish-eye images are rectified to normal images. Second, algorithms are applied to these normal images. Since two steps are involved in these algorithms, they are referred to as two-step methods. Two-step methods exhibit better scalability than one-step methods; i.e., most algorithms can be directly applied to rectified images without any alterations.

The rectification process is the transformation of different projection models. Notably, it is impossible to map a spherical image onto one flat surface without any distortion. Therefore, each projection has its own distortion while rectifying other distortions. Several commonly employed image representation models have been introduced in the previous subsection. The white lines are the longitudinal and latitudinal lines of the spherical world. Different image representation models are easily distinguished according to the distortion of these lines.

We organize the rectification methods developed since 2005 for fish-eye images in intelligent vehicles in Table 2.4. In most papers, fish-eye images are rectified by using the rectilinear model [11, 12, 54–59, 62–64]. The rectilinear model is exactly the normal perspective image model, as shown in Fig. 2.1a. All the straight lines remain straight in the rectilinear image, which is the way a human observes things. Afterward, normal algorithms are directly applied to the rectified image.

Some works choose other image representation models to rectify fish-eye images, e.g., the cylindrical model [60, 61, 65], equirectangular model [68], and Mercator model [67]. The common characteristic of these three projections is that they can keep a line straight in the vertical direction. The advantages of these three image representation models are described as follows: (1) It is easy to stitch multiple fish-eye images to form an omnidirectional image since the alignment in the vertical direction is accurate. Usually, modern intelligent vehicles are equipped with four fish-eye cameras, and the resultant omnidirectional image can be fused using such methods. (2) Traffic objects, especially pedestrians, are less distorted in the horizontal direction in the corrected image as pedestrians remain upright after rectification, making them easier for the detector to identify.

2.5.3 One-Step Methods

Some solutions design specialized methods to adapt to the distortion in fish-eye images. Since these methods can be directly applied to the original fish-eye images, they are referred to as one-step methods in this chapter. One-step methods omit complicated calibrations and rectification processes and can be directly applied to original fish-eye images. Moreover, two-step methods inevitably lose information during the rectification process, while one-step methods retain all information contained in the original fish-eye images.

Table 2.4 Two-step methods for object detection and tracking

Reference	Author	Year	Rectification model	Rectification method
[54]	Bertozzi et al.	2005	Rectilinear model	Lookup table
[55]	Wybo et al.	2007	Rectilinear model	IPM and probabilistic reasoning
[56]	Yang et al.	2008	Rectilinear model	IPM and background subtraction
[57]	Ma et al.	2009	Rectilinear model	Unknown
[58]	Lin et al.	2010	Rectilinear model	Backward mapping algorithm
[59]	Zhang et al.	2011	Rectilinear model	Unknown
[60]	Cheng et al.	2011	Cylindrical model	Coordinate transformation
[61]	Gressmann et al.	2011	Cylindrical model	Multicamera fusion
[62]	Tsuchiya et al.	2012	Rectilinear model	Inverse perspective mapping
[63]	Balisavira et al.	2012	Rectilinear model	Inverse perspective mapping
[64]	Broggi et al.	2014	Rectilinear model	Virtual views
[11]	Silberstein et al.	2014	Rectilinear model	Caltech Camera Calibration Toolbox
[12]	Levi et al.	2015	Rectilinear model	Unknown
[65]	Bertozzi et al.	2015	Cylindrical model	Camodocal[66]
[67]	Suhr et al.	2017	Mercator model	Unknown
[68]	Deng et al.	2017	Equirectangular model	Longitude-latitude method

We organize the one-step methods developed since 2011 for fish-eye images in intelligent vehicles in Table 2.5.

2.5.3.1 Distortion Adaptation

Bui et al. [75, 76] evaluated the effect of the deformable part model (DPM) algorithm on fish-eye images. Compared with the entire body, the limbs of the human body have less distortion in fish-eye images, and a method based on deformed components can adapt well to this characteristic. The researchers further reset the position of each component in their entire template, and the algorithm achieved ideal pedestrian detection performance.

Qian et al. [20] also made improvements based on the DPM algorithm. The researchers set a weight that could be learned for all root templates and component templates and that could adaptively learn the human features of the input fish-eye

Table 2.5 One-step methods for object detection and tracking

Reference	Author	Year	Distortion solution	Solution description
[69]	Molineros et al.	2012	Segmented processing	Residual flow
[70]	Cheng et al.	2012	Segmented processing	Parts-based method
[71]	Tadjine et al.	2013	Segmented processing	Multiple features and classifiers
[72]	Bui et al.	2013	Segmented processing	Histogram of oriented gradients (HOG) + support vector machine (SVM)
[73]	Kwon et al.	2014	Distortion adaptation	Background compensation
[74]	Liao et al.	2014	Distortion adaptation	Active contour model
[75]	Bui et al.	2014	Distortion adaptation	Deformable part model based pedestrian detector
[76]	Bui et al.	2014	Distortion adaptation	Deformable part model + ROI
[77]	Furnari et al.	2014	Segmented processing	Affine region detectors
[78]	Dooley et al.	2015	Segmented processing	Different strategies for different distances
[79]	Fremont et al.	2016	Data augmentation	Deformable part model + three-sensor fusion
[20]	Qian et al.	2017	Distortion adaptation	Adaptive deformable part model
[80]	Baek et al.	2018	Segmented processing	Multiple ROIs
[25]	Qian et al.	2019	Distortion adaptation	OSTN: oriented spatial transformer network
[81]	Yahiaoui et al.	2019	Distortion adaptation	FisheyeMODNet
[82]	Qian et al.	2020	Data augmentation	Adversary learning-based data augmentation
[83]	Li et al.	2020	Distortion adaptation	FisheyeDet
[84]	Wu et al.	2020	Distortion adaptation	Post-process of SiamRPN++
[85]	Zhao et al.	2021	Data augmentation	Transfer learning fish-eye images using for pedestrian re-identification
[15]	Rashed et al.	2021	Distortion adaptation	FisheyeYOLO
[86]	Plaut et al.	2021	Distortion adaptation	3D detection model adaptation via cylindrical images

image during the training process. The settings of these learnable weights greatly improve the ability of the algorithm to adapt to distorted targets. In addition, Qian et al. [25] continued to design the oriented spatial transformation network (OSTN), which can transform distorted pedestrian features into upright normal pedestrian features. This approach has achieved better accuracy in pedestrian detection tasks involving fish-eye images.

Yahiaoui et al. proposed FisheyeMODNet [81] to detect moving objects. The method is adapted from the ShuffleSeg network [87]. Li et al. proposed FisheyeDet [83] based on a self-study and contour-based object detector. The network adaptively extracted distortion features without prior information. Rashed et al. [15] adapted the You Only Look Once version 3 (YOLOv3) [88] model to output different representations. As in YOLOv3, object detection is performed at multiple scales. Plaut et al. proposed a 3D object detection method for fish-eye images [86]. This was the first approach to enable the use of existing 3D object detectors. The method was designed for perspective images and trained only on datasets containing perspective images.

2.5.3.2 Segmented Processing

Dooley et al. [78] devised different approaches according to the distance of the target vehicle as a vehicle has less distortion at greater distances in a fish-eye image and greater distortion up close. At a distance of 10-40 meters, the adaptive boosting (AdaBoost) classifier can be directly applied. For short distances, a wheel recognition method was proposed.

Baek et al. [80] discovered that the target in a fish-eye image has a larger distortion at the edge of the image and a smaller distortion at the centre of the image. Therefore, the researchers divided a whole fish-eye image into three ROIs; the middle part was directly detected by a traditional CNN because of its small feature distortion. For the left and right ROIs, a distorted network model was specifically trained.

2.5.3.3 Data Augmentation

Fremont et al. [79] divided a fish-eye image into seven regions corresponding to four different degrees of target distortion. The authors generated four corresponding training samples based on the imaging model of their fish-eye camera. Data augmentation is a very important step in machine algorithms. The training samples generated by this method greatly benefit the subsequent pedestrian detector. This approach has achieved good results in pedestrian detection based on fish-eye images and has a massive role in heavy machinery.

Qian et al. [25] proposed the projective model transformation algorithm, which generates fish-eye images from normal perspective images according to the imaging principle of the utilized fish-eye camera. This method can be directly applied to most normal datasets to generate numerous fish-eye training samples. These numerous samples provide ample room for the training of various algorithms based on deep learning. The authors further proposed a hard example mining method based on adversarial learning [82, 89]. Through adversarial training, this method can further excavate large distortions and difficult fish-eye images. These images are very significant with regards to long-tail scenes and significantly improve the performance of deep neural networks.

2.5.4 Analysis of Object Detection

In terms of early practical applications, the two-step approach has gained more attention because of its convenience. In recent years, with the development of deep learning technology, an increasing amount of work has been directly carried out on fish-eye images. We suggest that the current approach to fish-eye image detection tasks is too customized. Even the detection method based on the deep network has difficulty obtaining good results on perspective and fish-eye images. In future work, the most advanced deformable convolutional network structures [90–96] and domain adaptive methods should be developed in this field.

2.6 Summary and Prospects

In this paper, the fish-eye camera and its application in autonomous driving are introduced. First, we introduce the projection principle of the fish-eye camera and four classic fish-eye image representation models. Second, we present the current dataset of real and virtual fish-eye images. Last, we organized various development applications for fish-eye cameras in autonomous driving.

We present the following summary and prospect of the fish-eye camera in autonomous driving.

- Depth perception in fish-eye images is a valuable research task. Due to the maturity of depth reconstruction in normal perspective images, the corresponding technology can introduce depth perception to fish-eye images. Combined with AVM technology, depth perception can greatly help intelligent vehicles, for example, by providing collision warnings and automated valet parking.
- We suggest that the current approach to fish-eye image detection tasks is too customized. Even the detection method based on the deep network has difficulty obtaining good results on perspective and fish-eye images. In future work, the most advanced deformable convolutional network structures and domain adaptive methods should be developed in this field.
- Although researchers try to solve the semantic segmentation task with various new depth learning technologies, the unified representation of image features has not yet been established. In future research, it will be exciting to establish a unified semantic segmentation model for various image representation models.

- Compared with the common image dataset, the fish-eye image dataset is insufficient. Even existing fish-eye datasets are often geared towards conventional driving scenarios. We suggest that fish-eye cameras are most valuable for low-speed and short-range scenarios because of the lack of a scene-specific dataset.

References

1. Gao B, Pan Y, Li C, Geng S, Zhao H (2021) Are we hungry for 3d lidar data for semantic segmentation? A survey of datasets and methods, IEEE Trans Intell Transp Syst
2. Berrio JS, Shan M, Worrall S, Nebot E (2021) Camera-lidar integration: probabilistic sensor fusion for semantic mapping. IEEE Trans Intell Transp Syst
3. Peng K, Fei J, Yang K, Roitberg A, Zhang J, Bieder F, Heidenreich P, Stiller C, Stiefelhagen R (2022) Mass: multi-attentional semantic segmentation of lidar data for dense top-view understanding. IEEE Trans Intell Transp Syst
4. Feng D, Haase-Schütz C, Rosenbaum L, Hertlein H, Glaeser C, Timm F, Wiesbeck W, Dietmayer K (2020) Deep multi-modal object detection and semantic segmentation for autonomous driving: datasets, methods, and challenges. IEEE Trans Intell Transp Syst 22(3):1341–1360
5. David M (2017) Synthia: synthetic collection of imagery and annotations. IEEE Intell Transp Syst Mag 9(4):138–140
6. Geiger A, Lenz P, Urtasun R (2012) Are we ready for autonomous driving? The kitti vision benchmark suite. In: IEEE/CVF conference on conference on computer vision and pattern recognition (CVPR). IEEE, pp 3354–3361
7. Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S, Schiele B (2016) The cityscapes dataset for semantic urban scene understanding. In IEEE conference on computer vision and pattern recognition (CVPR), pp 3213–3223
8. Girshick R (2015) Fast r-cnn. In: IEEE international conference on computer vision (ICCV), pp 1440–1448
9. Romera E, Alvarez JM, Bergasa LM, Arroyo R (2017) Efficient convnet for real-time semantic segmentation. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 1789–1794
10. Romera E et al (2017) Erfnet: efficient residual factorized convnet for real-time semantic segmentation. IEEE Trans Intell Transp Syst 19(1):263–272
11. Silberstein S, Levi D, Kogan V, Gazit R (2014) Vision-based pedestrian detection for rear-view cameras. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 853–860
12. Levi D, Silberstein S (2015) Tracking and motion cues for rear-view pedestrian detection. In: IEEE international conference on intelligent transportation systems (ITSC). IEEE, pp 664–671
13. Maddern W, Pascoe G, Linegar C, Newman P (2017) 1 year, 1000 km: the oxford robotcar dataset. Int J Robot Res 36(1):3–15
14. Yogamani S, Hughes C, Horgan J, Sistu G, Varley P, O'Dea D, Uricár M, Milz S, Simon M, Amende K et al (2019) Woodscape: a multi-task, multi-camera fisheye dataset for autonomous driving. In: IEEE/CVF international conference on computer vision (ICCV), pp 9308–9318
15. Rashed H, Mohamed E, Sistu G, Kumar VR, Eising C, Sallab AE, Yogamani SK (2021) Generalized object detection on fisheye cameras for autonomous driving: dataset, representations and baseline. In: IEEE winter conference on applications of computer vision (WACV), pp 2271–2279
16. Liao Y, Xie J, Geiger A (2022) Kitti-360: a novel dataset and benchmarks for urban scene understanding in 2d and 3d. IEEE Trans Pattern Anal Mach Intell
17. Zhang Z, Rebecq H, Forster C, Scaramuzza D (2016) Benefit of large field-of-view cameras for visual odometry. In: IEEE international conference on robotics and automation (ICRA). IEEE, pp 801–808

18. Sekkat AR, Dupuis Y, Vasseur P, Honeine P (2020) The omniscape dataset. In: IEEE international conference on robotics and automation (ICRA), pp 1603–1608
19. Sekkat AR, Dupuis Y, Kumar VR, Rashed H, Yogamani S, Vasseur P, Honeine P (2022) Synwoodscape: synthetic surround-view fisheye camera dataset for autonomous driving. arXiv:2203.05056
20. Qian Y, Yang M, Wang C, Wang B (2017) Self-adapting part-based pedestrian detection using a fish-eye camera. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 33–38
21. Deng L, Yang M, Qian Y, Wang C, Wang B (2017) Cnn based semantic segmentation for urban traffic scenes using fisheye camera. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 231–236
22. Deng L, Yang M, Li H, Li T, Hu B, Wang C (2019) Restricted deformable convolution-based road scene semantic segmentation using surround view cameras. IEEE Trans Intell Transp Syst 21(10):4350–4362
23. Blott G, Takami M, Heipke C (2018) Semantic segmentation of fisheye images. In: European conference on computer vision (ECCV), pp 10–18
24. Sáez A, Bergasa LM, Romeral E, López E, Barea R, Sanz R (2018) Cnn-based fisheye image real-time semantic segmentation. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 1039–1044
25. Qian Y, Yang M, Zhao X, Wang C, Wang B (2019) Oriented spatial transformer network for pedestrian detection using fish-eye camera. IEEE Trans Multimed
26. Sáez Á, Bergasa LM, López-Guillén E, Romera E, Tradacete M, Gómez-Huélamo C, del Egido J (2019) Real-time semantic segmentation for fisheye urban driving images based on erfnet. Sensors 19(3):503
27. Ye Y, Yang K, Xiang K, Wang J, Wang K (2020) Universal semantic segmentation for fisheye urban driving images. In: IEEE international conference on systems, man, and cybernetics (SMC). IEEE, pp 648–655
28. Kannala J, Brandt SS (2006) A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. IEEE Trans Pattern Anal Mach Intell 28(8):1335–1340
29. Scaramuzza D (2021) Omnidirectional camera. Comput Vis
30. Dupuis Y, Savatier X, Ertaud J-Y, Vasseur P (2011) A direct approach for face detection on omnidirectional images. In: IEEE international symposium on robotic and sensors environments (ROSE). IEEE, pp 243–248
31. Hänisch S, Evangelio RH, Tadjine HH, Pätzold M (2017) Free-space detection with fish-eye cameras. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 135–140
32. Yeol Baek J, Veronica Chelu I, Iordache L, Paunescu V, Ryu H, Ghiuta A, Petreanu A, Soh Y, Leica A, Jeon B (2018) Scene understanding networks for autonomous driving based on around view monitoring system. In: IEEE conference on computer vision and pattern recognition workshops (CVPRW), pp 961–968
33. Zhou W, Zyner A, Worrall S, Nebot E (2019) Adapting semantic segmentation models for changes in illumination and camera perspective. IEEE Robot Autom Lett 4(2):461–468
34. Playout C, Ahmad O, Lecue F, Cheriet F (2021) Adaptable deformable convolutions for semantic segmentation of fisheye images in autonomous driving systems. arXiv:2102.10191
35. Dai J, Qi H, Xiong Y, Li Y, Zhang G, Hu H, Wei Y (2017) Deformable convolutional networks. In: IEEE international conference on computer vision (ICCV), pp 764–773
36. Yang K, Hu X, Bergasa LM, Romera E, Huang X, Sun D, Wang K (2019) Can we pass beyond the field of view? Panoramic annular semantic segmentation for real-world surrounding perception. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 446–453
37. Yang K, Bergasa LM, Romera E, Cheng R, Chen T, Wang K (2018) Unifying terrain awareness through real-time semantic segmentation. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 1033–1038
38. Yang K, Hu X, Bergasa LM, Romera E, Wang K (2019) Pass: panoramic annular semantic segmentation. IEEE Trans Intell Transp Syst 21(10):4171–4185
39. Yang K, Hu X, Chen H, Xiang K, Wang K, Stiefelhagen R (2020) Ds-pass: detail-sensitive panoramic annular semantic segmentation through swaftnet for surrounding sensing. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 457–464

40. Yang K, Hu X, Fang Y, Wang K, Stiefelhagen R (2020) Omnisupervised omnidirectional semantic segmentation. IEEE Trans Intell Transp Syst
41. Yang K, Hu X, Stiefelhagen R (2021) Is context-aware cnn ready for the surroundings? Panoramic semantic segmentation in the wild. IEEE Trans Image Process 30:1866–1881
42. Yang K, Zhang J, Reiß S, Hu X, Stiefelhagen R (2021) Capturing omni-range context for omnidirectional segmentation. In: IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 1376–1386
43. Ma C, Zhang J, Yang K, Roitberg A, Stiefelhagen R (2021) Densepass: dense panoramic semantic segmentation via unsupervised domain adaptation with attention-augmented context exchange. In: IEEE international conference on intelligent transportation systems (ITSC). IEEE, pp 2766–2772
44. Zhang J, Ma C, Yang K, Roitberg A, Peng K, Stiefelhagen R (2021) Transfer beyond the field of view: dense panoramic semantic segmentation via unsupervised domain adaptation. IEEE Trans Intell Transp Syst
45. Dufour R, Meurie C, Strauss C, Lezoray O (2020) Instance segmentation in fisheye images. In: International conference on image processing theory, tools and applications (IPTA). IEEE, pp 1–6
46. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask r-cnn. In: IEEE international conference on computer vision (ICCV), pp 2961–2969
47. Jaus A, Yang K, Stiefelhagen R (2021) Panoramic panoptic segmentation: towards complete surrounding understanding via unsupervised contrastive learning. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 1421–1427
48. Petrovai A, Nedevschi S (2022) Semantic cameras for 360-degree environment perception in automated urban driving. IEEE Trans Intell Transp Syst
49. Li S (2006) Full-view spherical image camera. In: International conference on pattern recognition (ICPR), vol 4. IEEE, pp 386–390
50. Han SW, Suh DY (2020) Piinet: a 360-degree panoramic image inpainting network using a cube map. arXiv:2010.16003
51. Eder M, Shvets M, Lim J, Frahm J-M (2020) Tangent images for mitigating spherical distortion. In: IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 12 426–12 434
52. Schlosser T, Friedrich M, Kowerko D (2019) Hexagonal image processing in the context of machine learning: conception of a biologically inspired hexagonal deep learning framework. In: IEEE international conference on machine learning and applications (ICMLA). IEEE, pp 1866–1873
53. Zhao Y, Xie H, Yao J, Chen E, Xu S (2017) An improved longitude-latitude mapping algorithm for fisheye image calibration. In: AOPC 2017: optical sensing and imaging technology and applications (AOPC), vol 10462. International Society for Optics and Photonics, p 104622U
54. Bertozzi M, Broggi A, Medici P, Porta P, Vitulli R (2005) Obstacle detection for start-inhibit and low speed driving. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 569–574
55. Wybo S, Bendahan R, Bougnoux S, Vestri C, Abad F, Kakinami T (2007) Improving backing-up manoeuvre safety with vision-based movement detection. IET Intel Transp Syst 1(2):150–158
56. Yang C, Hongo H, Tanimoto S (2008) A new approach for in-vehicle camera obstacle detection by ground movement compensation. In: IEEE international conference on intelligent transportation systems (ITSC). IEEE, pp 151–156
57. Ma G, Dwivedi M, Li R, Sun C, Kummert A (2009) A real-time rear view camera based obstacle detection. In: IEEE international conference on intelligent transportation systems (ITSC). IEEE, pp 1–6
58. Lin C-T, Shen T-K, Shou Y-W (2010) Construction of fisheye lens inverse perspective mapping model and its applications of obstacle detection. EURASIP J Adv Signal Process 2010:8
59. Yankun Z, Hong C, Weyrich N (2011) A single camera based rear obstacle detection system. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 485–490
60. Cheng G, Chen X (2011) A vehicle detection approach based on multi-features fusion in the fisheye images. In: International conference on computer research and development (ICCRD), vol 4. IEEE, pp 1–5

61. Gressmann M, Palm G, Löhlein O (2011) Surround view pedestrian detection using heterogeneous classifier cascades. In: IEEE international conference on intelligent transportation systems (ITSC). IEEE, pp 1317–1324
62. Tsuchiya C, Tanaka S, Furusho H, Nishida K, Kurita T (2012) Real-time vehicle detection using a single rear camera for a blind spot warning system. SAE Int J Passeng Cars-Electr Electric Syst 5(2012-01-0293):146–153
63. Balisavira V, Pandey V (2012) Real-time object detection by road plane segmentation technique for adas. In: International conference on signal image technology and internet based systems (SITIS). IEEE, pp 161–167
64. Broggi A, Cardarelli E, Cattani S, Medici P, Sabbatelli M (2014) Vehicle detection for autonomous parking using a soft-cascade adaboost classifier. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 912–917
65. Bertozzi M, Castangia L, Cattani S, Prioletti A, Versari P (2015) 360 detection and tracking algorithm of both pedestrian and vehicle using fisheye images. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 132–137
66. Heng L, Li B, Pollefeys M (2013) Camodocal: automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry. In: IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 1793–1800
67. Suhr JK, Jung HG (2017) Rearview camera-based backover warning system exploiting a combination of pose-specific pedestrian recognitions. IEEE Trans Intell Transp Syst 19(4):1122–1129
68. Deng F, Zhu X, Ren J (2017) Object detection on panoramic images based on deep learning. In: International conference on control, automation and robotics (ICCAR). IEEE, pp 375–380
69. Molineros J, Cheng SY, Owechko Y, Levi D, Zhang W (2012) Monocular rear-view obstacle detection using residual flow. In: European conference on computer vision (ECCV). Springer, pp 504–514
70. Cheng SY, Molineros J, Owechko Y, Levi D, Zhang W (2012) Parts-based object recognition seeded by frequency-tuned saliency for child detection in active safety. In: IEEE conference on intelligent transportation systems (ITSC). IEEE, pp 1155–1160
71. Tadjine H, Hess M, Karsten S (2013) Object detection and classification using a rear in-vehicle fisheye camera. In: FISITA world automotive congress. Springer, pp 519–528
72. Bui M, Fremont V, Boukerroui D, Letort P (2013) People detection in heavy machines applications. In: IEEE conference on cybernetics and intelligent systems (CIS). IEEE, pp 18–23
73. Kwon J, Kim D-S (2014) Detecting moving objects from slowly moving viewpoint for automotive rear view camera systems. In: IEEE international conference on consumer electronics (ICCE). IEEE, pp 51–52
74. Liao Y-S, Hsu L-Y (2014) Design and implementation of an image-based backing assistance system. In: IEEE international conference on intelligent transportation systems (ITSC). IEEE, pp 1898–1899
75. Bui M-T, Frémont V, Boukerroui D, Letort P (2014) Deformable parts model for people detection in heavy machines applications. In: International conference on control automation robotics vision (ICCARV). IEEE, pp 389–394
76. Bui M, Frémont V, Boukerroui D, Letort P (2014) Multi-sensors people detection system for heavy machines. In: IEEE international conference on intelligent transportation systems (ITSC). IEEE, pp 867–872
77. Furnari A, Farinella GM, Puglisi G, Bruna AR, Battiato S (2014) Affine region detectors on the fisheye domain. In: IEEE international conference on image processing (ICIP). IEEE, pp 5681–5685
78. Dooley D, McGinley B, Hughes C, Kilmartin L, Jones E, Glavin M (2015) A blind-zone detection method using a rear-mounted fisheye camera with combination of vehicle detection methods. IEEE Trans Intell Transp Syst 17(1):264–278
79. Fremont V, Bui M, Boukerroui D, Letort P (2016) Vision-based people detection system for heavy machine applications. Sensors 16(1):128

80. Baek I, Davies A, Yan G, Rajkumar RR (2018) Real-time detection, tracking, and classification of moving and stationary objects using multiple fisheye images. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 447–452
81. Yahiaoui M, Rashed H, Mariotti L, Sistu G, Clancy I, Yahiaoui L, Kumar VR, Yogamani S (2019) Fisheyemodnet: moving object detection on surround-view cameras for autonomous driving. arXiv:1908.11789
82. Qian Y, Yang M, Li H, Wang C, Wang B (2020) Adversarial training-based hard example mining for pedestrian detection in fish-eye images. IEEE Trans Intell Transp Syst 22(12):7688–7698
83. Li T, Tong G, Tang H, Li B, Chen B (2020) Fisheyedet: a self-study and contour-based object detector in fisheye images. IEEE Access 8:71 739–71 751
84. Wu Z, Wang M, Yin L, Sun W, Wang J, Wu H (2020) Vehicle re-id for surround-view camera system. arXiv:2006.16503
85. Zhao Z, Zhao Z, Wang S, Watta P, Murphey YL (2021) Pedestrian re-identification using a surround-view fisheye camera system. In: International joint conference on neural networks (IJCNN). IEEE, pp 1–8
86. Plaut E, Ben-Yaacov E, Shlomo BE (2021) 3d object detection from a single fisheye image without a single fisheye training image. In: IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW), pp 3654–3662
87. Gamal M, Siam M, Abdel-Razek M (2018) Shuffleseg: real-time semantic segmentation network. arXiv:1803.03816
88. Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. arXiv:abs/1804.02767
89. Qian Y, Yang M, Wang C, Wang B (2018) Pedestrian feature generation in fish-eye images via adversary. In: IEEE international conference on robotics and automation (ICRA). IEEE, pp 2007–2012
90. Khasanova R, Frossard P (2017) Graph-based classification of omnidirectional images. In: IEEE international conference on computer vision workshops (ICCVW), pp 869–878
91. Coors B, Condurache AP, Geiger A (2018) Spherenet: learning spherical representations for detection and classification in omnidirectional images. In: European conference on computer vision (ECCV), pp 518–533
92. Perraudin N, Defferrard M, Kacprzak T, Sgier R (2019) Deepsphere: efficient spherical convolutional neural network with healpix sampling for cosmological applications. Astron Comput 27:130–146
93. Su Y-C, Grauman K (2019) Kernel transformer networks for compact spherical convolution. In: IEEE/CVF conference on computer vision and pattern recognition, pp 9442–9451
94. Jeon Y, Kim J (2017) Active convolution: learning the shape of convolution for image classification. In: IEEE conference on computer vision and pattern recognition, pp 4201–4209
95. Henriques JF, Vedaldi A (2017) Warped convolutions: efficient invariance to spatial transformations. In: International conference on machine learning (ICML). PMLR, pp 1461–1469
96. Cohen TS, Geiger M, Köhler J, Welling M (2018) Spherical cnns. arXiv:1801.10130

Chapter 3
Stereo Matching: Fundamentals, State-of-the-Art, and Existing Challenges

Chuang-Wei Liu, Hengli Wang, Sicen Guo, Mohammud Junaid Bocus, Qijun Chen, and Rui Fan

Abstract Stereo matching is the process of generating dense correspondences in stereo images in order to create a disparity map for depth perception. Stereo matching is different from flow estimation task due to stereo rectification, which ensures that correspondences are always co-linear in a pair of stereo images. Stereo vision has become increasingly popular in mobile devices, such as autonomous cars and unmanned aerial vehicles, thanks to recent advances in full-feature embedded microcomputers. However, due to limited computing resources, there is a growing need for stereo matching algorithms that strike a balance between disparity estimation accuracy and efficiency. Challenges in this field include the lack of disparity ground truth, domain adaptation, and intractable areas such as occlusions. This chapter covers the fundamentals of stereopsis, including the perspective camera model and epipolar geometry, and reviews the most advanced stereo matching algorithms. It also explores

C.-W. Liu · S. Guo · Q. Chen · R. Fan (✉)
The Machine Intelligence and Autonomous Systems (MIAS) Group, the Robotics and Artificial Intelligence Laboratory (RAIL), State Key Laboratory of Intelligent Autonomous Systems, The Department of Control Science and Engineering, and Frontiers Science Center for Intelligent Autonomous Systems, Tongji University, Shanghai 201804, People's Republic of China
e-mail: rfan@tongji.edu.cn

C.-W. Liu
e-mail: cwliu@tongji.edu.cn

S. Guo
e-mail: guosicen@tongji.edu.cn

Q. Chen
e-mail: qjchen@tongji.edu.cn

H. Wang
The Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong SAR, China
e-mail: hwangdf@connect.ust.hk

M. J. Bocus
University of Bristol, Bristol, United Kingdom
e-mail: junaid.bocus@bristol.ac.uk

R. Fan et al. (eds.), *Autonomous Driving Perception*, Advances in Computer Vision and Pattern Recognition, https://doi.org/10.1007/978-981-99-4287-9_3

disparity confidence measures, disparity estimation evaluation metrics, and publicly available datasets and benchmarks, before summarizing the outstanding challenges in this field.

3.1 Introduction

Stereo matching is one of the most important tasks in computer vision, drawing the attention of many researchers for its wide range of applications [1], including autonomous driving [2–4], unmanned aerial vehicles [5], road condition assessment [6–10], mobile robot navigation [11, 12], and remote sensing [13]. In addition to the rapid development of stereo matching algorithms, there has been a parallel effort to create public benchmarks and datasets [14–17] that can serve as baselines for researchers to evaluate the effectiveness of their own stereo matching algorithms. With the advancement of machine learning techniques, stereo matching algorithms have been able to achieve exceptional accuracy on public benchmarks [18]. However, many challenges remain unresolved, hindering more accurate disparity estimation and wider application [19]. These challenges include disparity estimation in occluded regions [20] and unsupervised training [21].

This chapter aims to offer researchers interested in stereopsis a thorough introduction to the fundamentals of stereo matching, which includes the perspective camera model and the epipolar geometry of a stereo system. Section 3.4 covers a detailed discussion of the latest stereo matching algorithms, which are classified into two categories: explicit programming-based and deep learning-based. Then, in Sect. 3.5, we introduce the disparity confidence measure, which is a commonly used auxiliary task in stereo matching due to its ability to provide prior knowledge of the degree of difficulty in disparity estimation. In Sect. 3.6, we discuss the metrics used for evaluating the accuracy and efficiency of disparity estimation. We also introduce several publicly available stereo matching datasets and commonly-used benchmarks in Sect. 3.7. Finally, in Sect. 3.8, we summarize the four main challenges that currently exist in the stereo matching task. This chapter provides a comprehensive overview of the field of stereo matching and serves as a valuable resource for researchers conducting related studies.

3.2 One Camera

3.2.1 Perspective Camera Model

The perspective camera model projects observed points in the world onto its imaging plane through the camera center. As illustrated in Fig. 3.1, $\boldsymbol{o}^{\mathrm{C}}$ denotes the camera center; $\boldsymbol{\Pi}$ and $\hat{\boldsymbol{\Pi}}$ represent the imaging plane and the normalized imag-

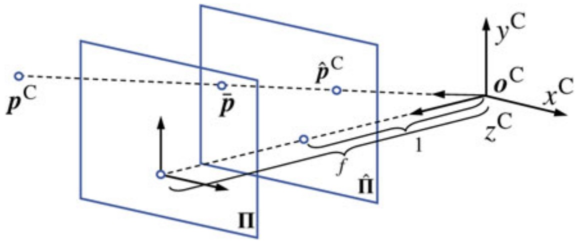


Fig. 3.1 Illustration of the perspective camera model, in which light travels in a straight line

ing plane, respectively; f represents the camera focal length. The optical axis is the ray originating at $\boldsymbol{o}^{\mathrm{C}}$ and passing perpendicularly through $\boldsymbol{\Pi}$. An observed 3D point $\boldsymbol{p}^{\mathrm{C}} = [x^{\mathrm{C}}, y^{\mathrm{C}}, z^{\mathrm{C}}]^{\top}$ in the camera coordinate system (CCS) is projected to $\bar{\boldsymbol{p}} = [x, y, f]^{\top}$ on $\boldsymbol{\Pi}$. The geometric relationship between $\boldsymbol{p}^{\mathrm{C}}$ and $\bar{\boldsymbol{p}}$ is as follows:

$$\frac{x^{\mathrm{C}}}{x} = \frac{y^{\mathrm{C}}}{y} = \frac{z^{\mathrm{C}}}{f}. \tag{3.1}$$

$\hat{\boldsymbol{p}}^{\mathrm{C}} = [\frac{x^{\mathrm{C}}}{z^{\mathrm{C}}}, \frac{y^{\mathrm{C}}}{z^{\mathrm{C}}}, 1]^{\top}$ is the normalized coordinates of $\boldsymbol{p}^{\mathrm{C}}$ on $\hat{\boldsymbol{\Pi}}$, and thus (3.1) can be rewritten as follows [22]:

$$\bar{\boldsymbol{p}} = f\hat{\boldsymbol{p}}^{\mathrm{C}} = \frac{f}{z^{\mathrm{C}}}\boldsymbol{p}^{\mathrm{C}}. \tag{3.2}$$

3.2.2 Intrinsic Matrix

In a perspective camera model, the relationship between point $\bar{\boldsymbol{p}} = [x, y, f]^{\top}$ on the image plane $\boldsymbol{\Pi}$ and its corresponding image pixel $\boldsymbol{p} = [u, v]^{\top}$ is given by [23]:

$$u - u_o = \alpha_x x, \quad v - v_o = \alpha_y y, \tag{3.3}$$

where the optical axis intersects the image plane at the image coordinates $[u_o, v_o]^{\top}$; α_x and α_y denote the effective size measured horizontally and vertically from the imaging plane $\boldsymbol{\Pi}$ to the image (in pixels per millimeter), respectively [22]. Plugging (3.2) into (3.3) yields:

$$u - u_o = \alpha_x f\frac{x^{\mathrm{C}}}{z^{\mathrm{C}}}, \quad v - v_o = \alpha_y f\frac{y^{\mathrm{C}}}{z^{\mathrm{C}}}. \tag{3.4}$$

Equation (3.4) can be rewritten as a matrix expression as follows:

$$\tilde{\boldsymbol{p}} = \frac{1}{z^{\mathrm{C}}}\boldsymbol{K}\boldsymbol{p}^{\mathrm{C}} = \frac{1}{z^{\mathrm{C}}}\begin{bmatrix} f_x & 0 & u_o \\ 0 & f_y & v_o \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x^{\mathrm{C}} \\ y^{\mathrm{C}} \\ z^{\mathrm{C}} \end{bmatrix}, \tag{3.5}$$

where $\boldsymbol{K}$ denotes the camera intrinsic matrix; $\tilde{\boldsymbol{p}} = [\boldsymbol{p}^\top, 1]^\top = [u, v, 1]^\top$ denotes the homogeneous coordinates of $\boldsymbol{p}$; $f_x = \alpha_x f$ and $f_y = \alpha_y f$. Plugging (3.5) into (3.2) yields:

$$\hat{\boldsymbol{p}}^{\mathrm{C}} = \boldsymbol{K}^{-1}\tilde{\boldsymbol{p}} = \frac{\bar{\boldsymbol{p}}}{f} = \frac{\boldsymbol{p}^{\mathrm{C}}}{z^{\mathrm{C}}}. \tag{3.6}$$

Therefore, all the 3D points on the ray originating at $\boldsymbol{o}^{\mathrm{C}}$ and passing through $\bar{\boldsymbol{p}}$ are projected on the image plane at $\bar{\boldsymbol{p}}$.

3.3 Two Cameras

3.3.1 Geometry of Multiple Images

3.3.1.1 Epipolar Geometry

The geometric constraint relationship between two perspective camera models is known as *epipolar geometry*, as illustrated in Fig. 3.2. Given two perspective camera models, $\boldsymbol{\Pi}_{\mathrm{L}}$ and $\boldsymbol{\Pi}_{\mathrm{R}}$ are the left and right image planes, respectively; $\boldsymbol{o}_{\mathrm{L}}^{\mathrm{C}}$ and $\boldsymbol{o}_{\mathrm{R}}^{\mathrm{C}}$ are the origins of the left camera coordinate system (LCCS) and the right camera coordinate system (RCCS) and the optic centers of the two camera models, respectively; $\boldsymbol{e}_{\mathrm{L}}$ and $\boldsymbol{e}_{\mathrm{R}}$ denote the left and right *epipolar line*, respectively. For a 3D point $\boldsymbol{p}^{\mathrm{W}}$ in the world coordinate system (WCS), its representations in the LCCS and RCCS are denoted by $\boldsymbol{p}_{\mathrm{L}}^{\mathrm{C}} = [x_{\mathrm{L}}^{\mathrm{C}}, y_{\mathrm{L}}^{\mathrm{C}}, z_{\mathrm{L}}^{\mathrm{C}}]^\top$ and $\boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}} = [x_{\mathrm{R}}^{\mathrm{C}}, y_{\mathrm{R}}^{\mathrm{C}}, z_{\mathrm{R}}^{\mathrm{C}}]^\top$, respectively. On the basis of 3D coordinate transformation in Appendix A, $\boldsymbol{p}_{\mathrm{L}}^{\mathrm{C}}$ can be transformed into $\boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}}$ using the rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times 3}$ and the translation vector $\boldsymbol{t} \in \mathbb{R}^{3\times 1}$ between the camera models:

$$\boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}} = \boldsymbol{R}\boldsymbol{p}_{\mathrm{L}}^{\mathrm{C}} + \boldsymbol{t}. \tag{3.7}$$

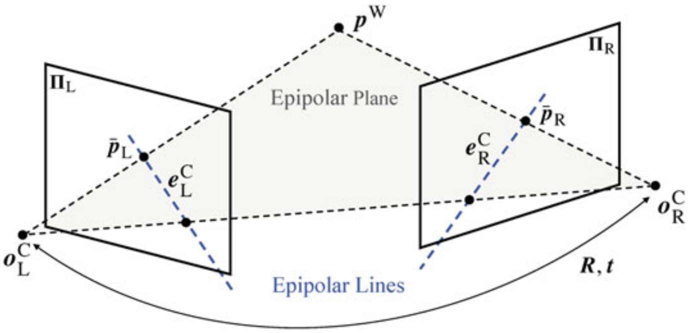

$\boldsymbol{p}^{\mathrm{W}}$
$\boldsymbol{\Pi}_{\mathrm{L}}$
$\boldsymbol{\Pi}_{\mathrm{R}}$
Epipolar Plane
$\bar{\boldsymbol{p}}_{\mathrm{L}}$
$\boldsymbol{e}_{\mathrm{L}}^{\mathrm{C}}$
$\bar{\boldsymbol{p}}_{\mathrm{R}}$
$\boldsymbol{e}_{\mathrm{R}}^{\mathrm{C}}$
$\boldsymbol{o}_{\mathrm{L}}^{\mathrm{C}}$
$\boldsymbol{o}_{\mathrm{R}}^{\mathrm{C}}$
Epipolar Lines
$\boldsymbol{R}, \boldsymbol{t}$

Fig. 3.2 Illustration of the epipolar geometry. The epipolar plane is uniquely defined by $\boldsymbol{o}_{\mathrm{L}}^{\mathrm{C}}$, $\boldsymbol{o}_{\mathrm{R}}^{\mathrm{C}}$, and $\boldsymbol{p}^{\mathrm{W}}$

The projection point of $\boldsymbol{p}^{\mathrm{W}}$ on Π_{L} and Π_{R} are $\bar{\boldsymbol{p}}_{\mathrm{L}} = [x_{\mathrm{L}}, y_{\mathrm{L}}, z_{\mathrm{L}}] = \frac{f_{\mathrm{L}}}{z_{\mathrm{L}}^{\mathrm{C}}} \boldsymbol{p}_{\mathrm{L}}^{\mathrm{C}}$ and $\bar{\boldsymbol{p}}_{\mathrm{R}} = [x_{\mathrm{R}}, y_{\mathrm{R}}, z_{\mathrm{R}}] = \frac{f_{\mathrm{R}}}{z_{\mathrm{R}}^{\mathrm{C}}} \boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}}$, respectively; f_{L} and f_{R} represent the focal lengths of the left and right cameras, respectively; The *epipolar plane* is defined by the camera centers $\boldsymbol{o}_{\mathrm{L}}^{\mathrm{C}}$, $\boldsymbol{o}_{\mathrm{R}}^{\mathrm{C}}$ and the 3D point $\boldsymbol{p}^{\mathrm{W}}$; The two epipolar lines represent the intersection of the epipolar plane and the two image planes Π_{L} and Π_{R}, respectively. Using (3.6), $\boldsymbol{p}_{\mathrm{L}}^{\mathrm{C}}$ and $\boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}}$ can be normalized as follows:

$$\hat{\boldsymbol{p}}_{\mathrm{L}}^{\mathrm{C}} = \frac{\boldsymbol{p}_{\mathrm{L}}^{\mathrm{C}}}{z_{\mathrm{L}}^{\mathrm{C}}} = \boldsymbol{K}_{\mathrm{L}}^{-1} \tilde{\boldsymbol{p}}_{\mathrm{L}}, \quad \hat{\boldsymbol{p}}_{\mathrm{R}}^{\mathrm{C}} = \frac{\boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}}}{z_{\mathrm{R}}^{\mathrm{C}}} = \boldsymbol{K}_{\mathrm{R}}^{-1} \tilde{\boldsymbol{p}}_{\mathrm{R}}, \tag{3.8}$$

where $\boldsymbol{K}_{\mathrm{L}}$ and $\boldsymbol{K}_{\mathrm{R}}$ are the left and right intrinsic matrices, respectively; $\tilde{\boldsymbol{p}}_{\mathrm{L}} = [\boldsymbol{p}_{\mathrm{L}}^{\top}, 1]^{\top} = [u_{\mathrm{L}}, v_{\mathrm{L}}, 1]^{\top}$ and $\tilde{\boldsymbol{p}}_{\mathrm{R}} = [\boldsymbol{p}_{\mathrm{R}}^{\top}, 1] = [u_{\mathrm{R}}, v_{\mathrm{R}}, 1]^{\top}$ are the homogeneous coordinates of the image pixels $\boldsymbol{p}_{\mathrm{L}}$ and $\boldsymbol{p}_{\mathrm{R}}$, respectively.

Considering an arbitrary 3D point $\boldsymbol{p}_0^{\mathrm{W}}$ on the ray that emanates from $\bar{\boldsymbol{p}}_{\mathrm{L}}$ and passes through $\boldsymbol{p}^{\mathrm{W}}$, it is always projected on Π_{L} at $\boldsymbol{p}_{\mathrm{L}}$ as discussed in Sect. 3.2.2. Additionally, its projection on Π_{R} is on the right epipolar line $\boldsymbol{e}_{\mathrm{R}}$, denoted by right image pixel $\boldsymbol{p}_{\mathrm{R}_0} = [u_{\mathrm{R}_0}, v_{\mathrm{R}_0}]^{\top}$. In the right image, the right epipolar line $\boldsymbol{e}_{\mathrm{R}}$ can be described by a vector $\boldsymbol{e}_{\mathrm{R}} = [a_{\mathrm{R}}, b_{\mathrm{R}}, c_{\mathrm{R}}]^{\top}$, which satisfies the following property:

$$\tilde{\boldsymbol{p}}_{\mathrm{R}_0}^{\top} \boldsymbol{e}_{\mathrm{R}} = [\boldsymbol{p}_{\mathrm{R}_0}^{\top}, 1] \boldsymbol{e}_{\mathrm{R}} = a_{\mathrm{R}} u_{\mathrm{R}_0} + b_{\mathrm{R}} v_{\mathrm{R}_0} + c_{\mathrm{R}} = 0, \tag{3.9}$$

where $\tilde{\boldsymbol{p}}_{\mathrm{R}_0} = [\boldsymbol{p}_{\mathrm{R}_0}^{\top}, 1]^{\top}$ represents the homogeneous coordinate of the image pixel $\boldsymbol{p}_{\mathrm{R}_0}$.

3.3.1.2 Fundamental Matrix

Fundamental matrix $\boldsymbol{F}$ is a 3 × 3 matrix that describes the epipolar geometry between stereo images. It corresponds an arbitrary image pixel $\boldsymbol{p}_{\mathrm{L}}$ in one view to a straight line in the other view. Multiplying both sides of (3.7) by $\boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}\top} [\boldsymbol{t}]_{\times}$ yields:

$$\boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}\top} [\boldsymbol{t}]_{\times} \boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}} = \boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}\top} [\boldsymbol{t}]_{\times} (\boldsymbol{R} \boldsymbol{p}_{\mathrm{L}}^{\mathrm{C}} + \boldsymbol{t}), \tag{3.10}$$

where $[\boldsymbol{t}]_{\times}$ denotes the *skew-symmetric* (or *antisymmetric*) matrix of $\boldsymbol{t}$ (see Appendix B for the definition and properties of a skew-symmetric matrix). According to (B.4), (3.10) can be rewritten as follows:

$$-\boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}\top} [\boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}}]_{\times} \boldsymbol{t} = \boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}\top} [\boldsymbol{t}]_{\times} \boldsymbol{R} \boldsymbol{p}_{\mathrm{L}}^{\mathrm{C}} + \boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}\top} [\boldsymbol{t}]_{\times} \boldsymbol{t}. \tag{3.11}$$

Applying (B.3) to (3.11) results in:

$$\boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}\top} [\boldsymbol{t}]_{\times} \boldsymbol{R} \boldsymbol{p}_{\mathrm{L}}^{\mathrm{C}} = 0. \tag{3.12}$$

Plugging (3.8) into (3.12) yields:

$$\tilde{\boldsymbol{p}}_{\mathrm{R}}^{\top}\boldsymbol{K}_{\mathrm{R}}{}^{-\top}[\boldsymbol{t}]_{\times}\boldsymbol{R}\boldsymbol{K}_{\mathrm{L}}{}^{-1}\tilde{\boldsymbol{p}}_{\mathrm{L}} = \tilde{\boldsymbol{p}}_{\mathrm{R}}^{\top}\boldsymbol{F}\tilde{\boldsymbol{p}}_{\mathrm{L}} = 0, \tag{3.13}$$

where the fundamental matrix $\boldsymbol{F}$ is defined as:

$$\boldsymbol{F} = \boldsymbol{K}_{\mathrm{R}}{}^{-\top}[\boldsymbol{t}]_{\times}\boldsymbol{R}\boldsymbol{K}_{\mathrm{L}}{}^{-1}. \tag{3.14}$$

According to (3.9), (3.13) can be rewritten as follows:

$$\tilde{\boldsymbol{p}}_{\mathrm{R}_0}^{\top}\boldsymbol{F}\tilde{\boldsymbol{p}}_{\mathrm{L}} = \tilde{\boldsymbol{p}}_{\mathrm{R}_0}^{\top}\boldsymbol{e}_{\mathrm{R}} = 0, \tag{3.15}$$

which indicates that for all points on the ray originating at $\boldsymbol{o}_{\mathrm{L}}^{\mathrm{C}}$ and passing through $\bar{\boldsymbol{p}}_{\mathrm{L}}$, their corresponding pixels in the right image plane form the right epipolar line $\boldsymbol{e}_{\mathrm{R}}$. $\boldsymbol{e}_{\mathrm{R}}$ is uniquely defined by $\boldsymbol{p}_{\mathrm{L}}$ as follows:

$$\boldsymbol{e}_{\mathrm{R}} = \boldsymbol{F}\tilde{\boldsymbol{p}}_{\mathrm{L}}. \tag{3.16}$$

3.3.1.3 Essential Matrix

Similar to the fundamental matrix, the *essential matrix* $\boldsymbol{E} \in \mathbb{R}^{3\times 3}$ depicts the point-line correspondence in the left and right normalized planes. An essential matrix can be considered as a special case of a fundamental matrix, that is, the intrinsic matrices $\boldsymbol{K}_{\mathrm{L}}$ and $\boldsymbol{K}_{\mathrm{R}}$ of two calibrated left and right cameras are already known. Thus (3.12) can be rewritten as follows:

$$\boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}\top}[\boldsymbol{t}]_{\times}\boldsymbol{R}\boldsymbol{p}_{\mathrm{L}}^{\mathrm{C}} = \boldsymbol{p}_{\mathrm{R}}^{\mathrm{C}\top}\boldsymbol{E}\boldsymbol{p}_{\mathrm{L}}^{\mathrm{C}} = 0, \tag{3.17}$$

where the essential matrix $\boldsymbol{E}$ is defined as:

$$\boldsymbol{E} = [\boldsymbol{t}]_{\times}\boldsymbol{R}. \tag{3.18}$$

Applying (3.8) to (3.17) yields:

$$\hat{\boldsymbol{p}}_{\mathrm{R}}^{\mathrm{C}\top}\boldsymbol{E}\hat{\boldsymbol{p}}_{\mathrm{L}}^{\mathrm{C}} = 0. \tag{3.19}$$

Similar to (3.16), $\boldsymbol{E}\hat{\boldsymbol{p}}_{\mathrm{L}}^{\mathrm{C}}$ denotes the parameters of the corresponding straight line in the normalized right image plane of the normalized left point $\hat{\boldsymbol{p}}_{\mathrm{L}}^{\mathrm{C}}$.

3.3.2 *Stereopsis*

3.3.2.1 Stereo Rectification

A pair of synchronized cameras can be used for 3D scene reconstruction, which involves determining the corresponding pixels between the left and right images. However, finding the corresponding pixel pairs on a 2D image space (optical flow estimation) is extremely time consuming. As discussed in Sect. 3.3.1.1, for a properly calibrated stereo vision system, $\boldsymbol{R}$, $\boldsymbol{t}$, $\boldsymbol{K}_{\mathrm{L}}$, and $\boldsymbol{K}_{\mathrm{R}}$ are already known, allowing the search range to be reduced to 1D along the epipolar line. In general, *stereo rectification* is applied as an image transformation process to align the two epipolar lines horizontally when the following relationship is satisfied:

$$\boldsymbol{R} = \boldsymbol{I}, \ \boldsymbol{t} = [T, 0, 0]. \tag{3.20}$$

Plugging (3.20) into (3.18) yields:

$$\boldsymbol{E} = [\boldsymbol{t}]_{\times}\boldsymbol{R} = \boldsymbol{t} \times \boldsymbol{R} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix}. \tag{3.21}$$

Considering a correspondence of normalized points $\hat{\boldsymbol{p}}_{\mathrm{L}}^{\mathrm{C}} = [x_{\mathrm{L}}, y_{\mathrm{L}}, 1]$ in the LCCS and $\hat{\boldsymbol{p}}_{\mathrm{R}}^{\mathrm{C}} = [y_{\mathrm{R}}, y_{\mathrm{R}}, 1]$ in the RCCS, applying (3.21) to (3.19) results in:

$$\hat{\boldsymbol{p}}_{\mathrm{R}}^{\mathrm{C}\top} \boldsymbol{E} \hat{\boldsymbol{p}}_{\mathrm{L}}^{\mathrm{C}} = [x_{\mathrm{L}}, y_{\mathrm{L}}, 1][0, -T, T \times y_{\mathrm{R}}]^{\top} = T \times (y_{\mathrm{R}} - y_{\mathrm{L}}) = 0, \tag{3.22}$$

which indicates that $\hat{\boldsymbol{p}}_{\mathrm{L}}^{\mathrm{C}}$ and $\hat{\boldsymbol{p}}_{\mathrm{R}}^{\mathrm{C}}$ have the same y coordinate, that is, the search range is further simplified to a horizontal row. The stereo rectification mainly involves the following steps [24]:

1. Orient the right camera with respect to the left camera using $\boldsymbol{R}$ to ensure the right image plane is parallel to the left image plane;
2. Rotate the right camera by $\boldsymbol{R}_{\mathrm{rect}}$ so that the left epipole is at infinity;
3. Reapply the same rotation to the right camera to restore the initial epipolar geometry;
4. Scale the left and right images to ensure an identical equivalent intrinsic matrix of the stereo cameras.

After the stereo rectification, the two image planes become coplanar, and conjugate epipolar lines become collinear and parallel to the horizontal image axis [22], as illustrated in Fig. 3.3, where Π_{L}' and Π_{R}' denote the rectified image planes. Therefore, the process of finding corresponding pairs is further simplified to a 1D horizontal search problem.

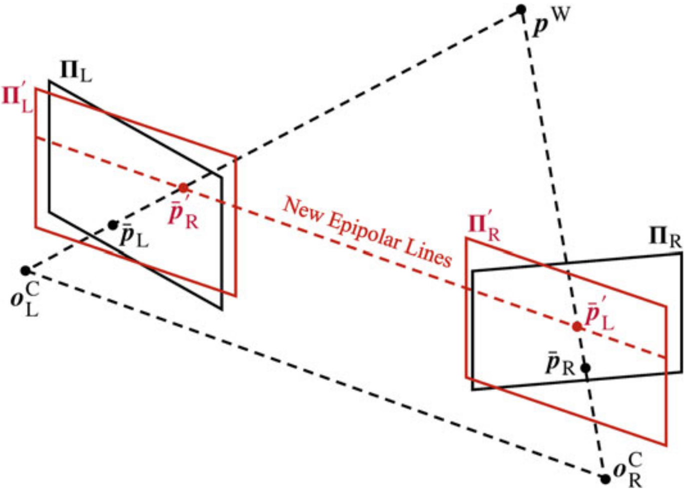


Fig. 3.3 Illustration of the stereo rectification. The left and right epipolar lines in a rectified stereo vision system are co-linear

3.3.2.2 Stereo Vision System

A properly rectified stereo vision system is depicted in Fig. 3.4, where the cameras are aligned in a parallel configuration. As a result of this alignment, the left and right cameras are considered identical, and their intrinsic matrices $\boldsymbol{K}_\mathrm{L}$ and $\boldsymbol{K}_\mathrm{R}$ are given by:

$$\boldsymbol{K}_\mathrm{L} = \boldsymbol{K}_\mathrm{R} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.23}$$

The length of the baseline of the stereo vision system is T_c, and the origin o^W of the WCS is positioned at the center of the baseline. The x, y and z-axis of the WCS are parallel to which of the LCCS and RCCS, and their x-axis are collinear. Hence, the rotation matrices between the three coordinate systems are both a third-order identity matrix $\boldsymbol{I}$, and the translation vectors from the WCS to the LCCS and the RCCS are $\boldsymbol{t}_\mathrm{L} = [-T_c/2, 0, 0]^\top$ and $\boldsymbol{t}_\mathrm{R} = [T_c/2, 0, 0]^\top$, respectively. Plugging (3.7) and (3.23) into (3.8) yields the following expressions:

$$\begin{aligned} x_\mathrm{L} &= f_x \frac{x^\mathrm{W} + T_c/2}{z^\mathrm{W}}, \quad y_\mathrm{L} = f_y \frac{y^\mathrm{W}}{z^\mathrm{W}}, \\ x_\mathrm{R} &= f_x \frac{x^\mathrm{W} - T_c/2}{z^\mathrm{W}}, \quad y_\mathrm{R} = f_y \frac{y^\mathrm{W}}{z^\mathrm{W}}. \end{aligned} \tag{3.24}$$

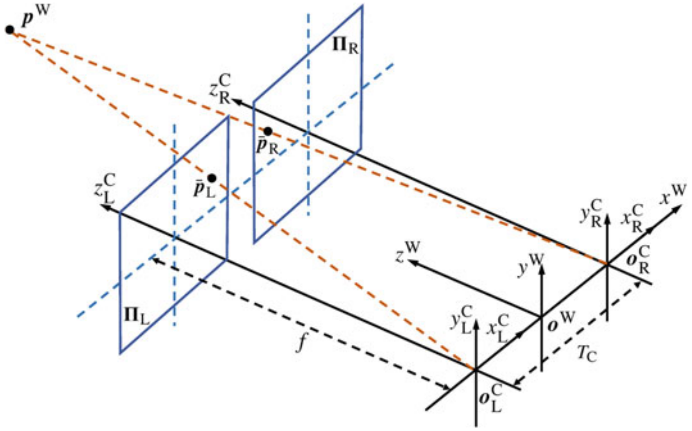


Fig. 3.4 Basic stereo vision system

For a 3D world point $\boldsymbol{p}^{\mathrm{W}} = [x^{\mathrm{W}}, y^{\mathrm{W}}, z^{\mathrm{W}}]^{\top}$, its corresponding pixel in the left and right images are denoted by $\boldsymbol{p}_{\mathrm{L}}$ and $\boldsymbol{p}_{\mathrm{R}}$, respectively. Applying (3.24) into (3.3) results in the coordinates of $\boldsymbol{p}_{\mathrm{L}}$ and $\boldsymbol{p}_{\mathrm{R}}$:

$$\boldsymbol{p}_{\mathrm{L}} = \begin{bmatrix} u_{\mathrm{L}} \\ v_{\mathrm{L}} \end{bmatrix} = \begin{bmatrix} f_x \frac{x^{\mathrm{W}}}{z^{\mathrm{W}}} + u_o + f_x \frac{T_c}{2z^{\mathrm{W}}} \\ f_y \frac{y^{\mathrm{W}}}{z^{\mathrm{W}}} + v_o \end{bmatrix}, \quad \boldsymbol{p}_{\mathrm{R}} = \begin{bmatrix} u_{\mathrm{R}} \\ v_{\mathrm{R}} \end{bmatrix} = \begin{bmatrix} f_x \frac{x^{\mathrm{W}}}{z^{\mathrm{W}}} + u_o - f_x \frac{T_c}{2z^{\mathrm{W}}} \\ f_y \frac{y^{\mathrm{W}}}{z^{\mathrm{W}}} + v_o \end{bmatrix}, \tag{3.25}$$

which further illustrates that $\hat{\boldsymbol{p}}_{\mathrm{L}}^{\mathrm{C}}$ and $\hat{\boldsymbol{p}}_{\mathrm{R}}^{\mathrm{C}}$ have the same y coordinate, as discussed in Sect. 3.3.2.1. Therefore, the disparity d (distance between co-row corresponding pixels) and depth z^{W} can be linked using:

$$d = u_{\mathrm{L}} - u_{\mathrm{R}} = f_x \frac{T_c}{z^{\mathrm{W}}}, \tag{3.26}$$

which indicates that d is inversely proportional to z^{W}. That is, when the 3D point $\boldsymbol{p}^{\mathrm{W}}$ lies away from the stereo vision system along the z-axis, its corresponding disparity d, the distance between u_{L} and u_{R}, is small.

3.4 Stereo Matching

With a properly rectified stereo vision system, the stereo matching task consists of finding the corresponding pairs of points ($\boldsymbol{p}_{\mathrm{L}}$ and $\boldsymbol{p}_{\mathrm{R}}$), and generating the dense disparity maps, as shown in Fig. 3.5. The performance of stereo matching algorithms is typically evaluated based on two main criterias: speed and accuracy. In general, there is a trade-off between speed and accuracy in stereo matching algo-

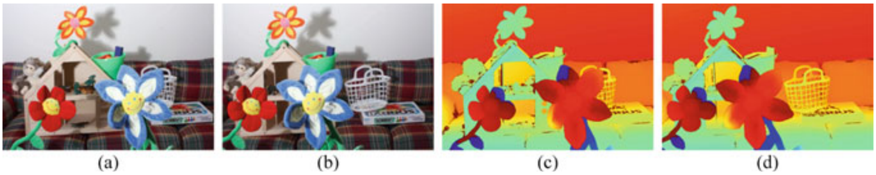


Fig. 3.5 **a** Left stereo image; **b** right stereo image; **c** left disparity map; **d** right disparity map

rithms. Well-designed algorithms tend to require more computations, resulting in a slower processing speed. The emphasis on speed or accuracy depends on the specific application of the stereo vision system [25]. For example, real-time performance is crucial for stereo vision systems used in autonomous driving [23], while accuracy is more important for indoor environment modeling [26]. Although advances in hardware are likely to reduce processing time in the future, improvements in algorithms and software are still significant.

Stereo matching algorithms can be divided into two classes: explicit programming-based and machine learning-based. The former considers disparity estimation as a local block matching problem or a global energy minimization problem [27], while the latter considers disparity estimation as a regression problem [28].

3.4.1 Explicit Programming-Based Stereo Matching Algorithms

Explicit programming-based stereo matching algorithms can be categorized into three types: local, global, and semi-global [29]. Local algorithms determine the disparity by finding the lowest cost or highest correlation, using a strategy called winner-takes-all (WTA). Global stereo matching algorithms formulate the problem of disparity estimation as a global energy minimization problem. This can be solved using Markov random fields (MRF)-based optimization algorithms [30] such as graph-cut (GC) [31] and dynamic programming (DP) [32]. Semi-global matching (SGM) [33, 34] approximates the MRF inference by performing cost aggregation along all directions in the image, which greatly improves the trade-off between the accuracy and efficiency of stereo matching [35]. Generally, the explicit programming-based stereo matching algorithms consist of four main steps: (1) cost computation, (2) cost aggregation, (3) disparity optimization and (4) disparity refinement [36].

3.4.1.1 Cost Computation

For all the pixels in the stereo images, each disparity candidate is associated with a matching cost. Generally, cost computation is used for evaluating the similarity of correspondence pairs, and a lower cost indicates a higher matching probability. Distance formulas including Manhattan distance and Euclidean distance are widely applied in cost computation, such as the absolute difference cost c_{AD} and the squared difference cost c_{SD}, which are defined as follows [37]:

$$\begin{aligned} c_{\mathrm{AD}}(u, v, d) &= \big|I_{\mathrm{L}}(u, v) - I_{\mathrm{R}}(u-d, v)\big|, \\ c_{\mathrm{SD}}(u, v, d) &= \big(I_{\mathrm{L}}(u, v) - I_{\mathrm{R}}(u-d, v)\big)^2, \end{aligned} \tag{3.27}$$

where d represents disparity; $I_{\mathrm{L}}(u, v)$, $I_{\mathrm{R}}(u-d, v)$ denote the pixel intensities of (u, v) in the left stereo image and $(u-d, v)$ in the right stereo image, respectively.

3.4.1.2 Cost Aggregation

In regions with weak texture or repetitive patterns, the effectiveness of the WTA strategy can be significantly impaired as there may be several correspondence pairs with similar costs. In order to avoid incorrect matches, c_{agg} aggregates matching cost by weighing the matching cost of a support region centered around the pixel $\boldsymbol{p} = [u, v]^\top$ [38]:

$$c_{\mathrm{agg}}(\boldsymbol{p}, d) = w(\boldsymbol{p}, d) * C(\boldsymbol{p}, d), \tag{3.28}$$

where w denotes the support region kernel; C refers to the matching costs or correlations of all the pixels within w. A commonly adopted method for computing aggregation cost c_{agg} is by performing a convolution between w and C.

Primal methods [27] use fixed square windows centered at each pixel as support regions, and all the elements in w are 1. In this way, the aggregation process becomes a uniform box filtering, and (3.27) can be reformulated as follows:

$$\begin{aligned} c_{\mathrm{SAD}}(u, v, d) &= \sum_{\boldsymbol{q} \in \mathscr{N}_{\mathbf{p}}} \big|I_{\mathrm{L}}(u, v) - I_{\mathrm{R}}(u-d, v)\big|, \\ c_{\mathrm{SSD}}(u, v, d) &= \sum_{\boldsymbol{q} \in \mathscr{N}_{\mathbf{p}}} \big(I_{\mathrm{L}}(u, v) - I_{\mathrm{R}}(u-d, v)\big)^2, \end{aligned} \tag{3.29}$$

where $\mathscr{N}_{\mathbf{p}}$ denotes the square window centered at $\boldsymbol{p} = [u, v]^\top$. Manhattan distance is used in SAD and SSD, which makes these methods computationally efficient but vulnerable to image intensity noise. In contrast, other more robust approaches such as normalized cross-correlation (NCC) have become prevalent in this process. NCC can be formulated as follows [27]:

$$c_{\mathrm{NCC}}(\boldsymbol{p}, d) = \frac{1}{n\sigma_{\mathrm{L}}\sigma_{\mathrm{R}}} \sum_{\boldsymbol{q}\in\mathscr{N}_{\mathrm{p}}} \Big(I_{\mathrm{L}}(\boldsymbol{q}) - \mu_{\mathrm{L}}\Big)\Big(I_{\mathrm{R}}\big(\boldsymbol{q} - \boldsymbol{d}\big) - \mu_{\mathrm{R}}\Big), \tag{3.30}$$

where

$$\sigma_{\mathrm{L}} = \sqrt{\sum_{\boldsymbol{q}\in\mathscr{N}_{\mathrm{p}}} \Big(I_{\mathrm{L}}(\boldsymbol{q}) - \mu_{\mathrm{L}}\Big)^2 / n}, \quad \sigma_{\mathrm{R}} = \sqrt{\sum_{\boldsymbol{q}\in\mathscr{N}_{\mathrm{p}}} \Big(I_{\mathrm{R}}(\boldsymbol{q} - \boldsymbol{d}) - \mu_{\mathrm{R}}\Big)^2 / n}; \tag{3.31}$$

$\boldsymbol{d} = [d, 0]^\top$; $\mu_{\mathrm{L,R}}$ and $\sigma_{\mathrm{L,R}}$ represent the average and standard deviation of the square windows of the stereo images, respectively; n is the number of pixels within the square window. The resulting $c_{\mathrm{NCC}} \in [-1, 1]$ signifies the similarity between the given pair of square windows. A higher c_{NCC} indicates a higher matching similarity.

In general, using a larger square window can help reduce uncertainty in disparity estimation, but it also significantly increases computation. The primal methods assume that pixels within fixed square windows share similar disparity, which is not always the case, particularly in regions with disparity discontinuities. Therefore, several adaptive cost aggregation strategies have been proposed to optimize support region generation and improve performance in these challenging regions. Consequently, numerous adaptive cost aggregation strategies have been proposed to optimize the support region generation. One such strategy [39] assigns weights to the pixels in the support region according to the color correlation. Fast bilateral stereo (FBS) [40] leverages a bilateral filter to aggregate the matching costs adaptively. FBS formulates the cost aggregation as follows:

$$c_{\mathrm{agg}}(\boldsymbol{p}, d) = \frac{\sum_{\boldsymbol{q}\in\mathscr{N}_{\mathrm{q}}} \omega_d(\boldsymbol{q})\omega_r(\boldsymbol{q})c(\boldsymbol{q}, d)}{\sum_{\boldsymbol{q}\in\mathscr{N}_{\mathrm{q}}} \omega_d(\boldsymbol{q})\omega_r(\boldsymbol{q})}, \tag{3.32}$$

where function ω_d is calculated from the spatial distance. Function ω_r reflects the color similarity.

3.4.1.3 Disparity Optimization

Unlike the WTA strategy in local algorithms, global algorithms determine the disparity by optimizing a global energy function, which typically has the following form [41]:

$$E(d) = E_{\mathrm{data}}(d) + E_{\mathrm{smooth}}(d), \tag{3.33}$$

where $E_{\mathrm{data}}(d)$ and $E_{\mathrm{smooth}}(d)$ represent a data term and a smooth term, respectively. The former denotes the global matching cost minimization, and the latter imposes

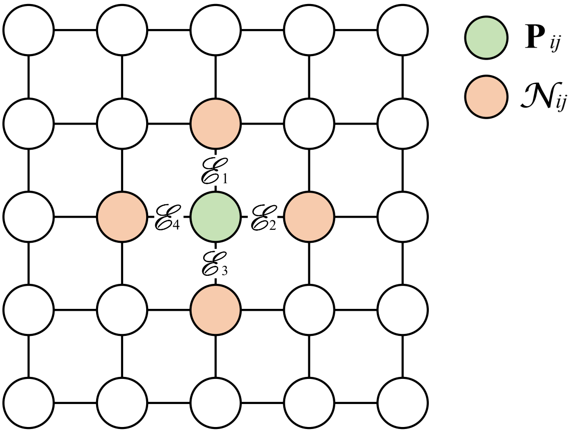


Fig. 3.6 An example of the MRF model in disparity optimization

constraints to reduce noise and errors in the cost volume, such as enforcing disparity smoothness and preserving image discontinuities [23].

Most global algorithms model the disparity optimization process with a MRF model. Figure 3.6 depicts a MRF model $\mathscr{G} = (\mathscr{P}, \mathscr{E})$, where $\mathscr{P} = \{\boldsymbol{p}_{11}, \boldsymbol{p}_{12}, \ldots, \boldsymbol{p}_{mn}\}$ denotes the vertices; $\mathscr{E} = \{(\boldsymbol{p}_{ij}, \boldsymbol{p}_{st}) \mid \boldsymbol{p}_{ij}, \boldsymbol{p}_{st} \in \mathscr{P}\}$ signifies the edges; and $\mathscr{N}_{ij} = \{\boldsymbol{q}_{1_{\boldsymbol{p}_{ij}}}, \boldsymbol{q}_{2_{\boldsymbol{p}_{ij}}}, \ldots, \boldsymbol{q}_{k_{\boldsymbol{p}_{ij}}} \mid \boldsymbol{q}_{\boldsymbol{p}_{ij}} \in \mathscr{P}\}$ represents a neighborhood system of $\boldsymbol{p}_{ij}$. Referring to the stereo matching task, all the vertices in $\mathscr{P}$ make up the $m \times n$ pixel disparity map. More specifically, $\boldsymbol{p}_{ij}$ denotes the vertex at (i, j) with a node value d_{ij} representing the disparity. In general, disparity nodes in the MRF model tend to have a stronger correlation with their neighboring nodes than other randomly chosen vertices. Therefore, (3.33) can be reformulated as follows:

$$E(\boldsymbol{p}) = \sum_{\boldsymbol{p}_{ij} \in \mathscr{P}} D(\boldsymbol{p}_{ij}, q_{\boldsymbol{p}_{ij}}) + \sum_{\boldsymbol{q}_{\boldsymbol{p}_{ij}} \in \mathscr{N}_{ij}} V(\boldsymbol{p}_{ij}, \boldsymbol{q}_{\boldsymbol{p}_{ij}}), \tag{3.34}$$

where $q_{\boldsymbol{p}_{ij}}$ represents image intensity differences; $D(\cdot)$ represents the matching cost or correlation and $V(\cdot)$ denotes the aggregation process inside a neighboring system, respectively.

Although global algorithms can achieve more accurate disparity results in textureless areas compared to local algorithms, the minimization of a global energy function which significantly increases computational requirements. Therefore, SGM [33] breaks down (3.34) into:

$$E(\boldsymbol{D}) = \sum_{\boldsymbol{p}} \left(\boldsymbol{c}(\boldsymbol{p}, d_{\boldsymbol{p}}) + \sum_{\boldsymbol{q} \in \mathscr{N}_{\mathbf{p}}} \lambda_1 \delta(|d_{\boldsymbol{p}} - d_{\boldsymbol{q}}| = 1) + \sum_{\boldsymbol{q} \in \mathscr{N}_{\mathbf{p}}} \lambda_2 \delta(|d_{\boldsymbol{p}} - d_{\boldsymbol{q}}| > 1) \right), \tag{3.35}$$

where $\boldsymbol{D}$ is the disparity map; $\boldsymbol{c}$ represents the cost volume; $\mathscr{N}_{\mathbf{p}}$ is the neighborhood system of pixel $\boldsymbol{p}$; λ_1 and λ_2 penalize its vicinities with different scales of disparity differences, i.e., one pixel and larger than one pixel, respectively; and $\delta(\cdot)$ takes on a value of 1 with a valid argument and 0 otherwise.

3.4.1.4 Disparity Refinement

At this point, the obtained disparity map usually has some intensity noise and holes. Therefore, post-processing steps are required for refinement. The left-right disparity consistency check (LRDCC) [27] generates both left and right disparity maps $\boldsymbol{D}_{\mathrm{L}}$ and $\boldsymbol{D}_{\mathrm{R}}$, and removes pixels with inconsistent disparities, which is defined as follows:

$$|\boldsymbol{D}_{\mathrm{L}}(u, v) - \boldsymbol{D}_{\mathrm{R}}(u - \boldsymbol{D}_{\mathrm{L}}(u, v), v)| > \sigma, \tag{3.36}$$

where σ is a manually set threshold and one pixel is commonly adopted. Subpixel enhancement increases the disparity image resolution by inserting a matching cost near the initial disparity [27]. Afterwards, a median filter [42] can be applied to fill the pixels that failed to match and the holes generated by LRDCC. Additionally, other methods such as robust plane fitting, intensity consistent, and locally consistent are also commonly adopted to improve the disparity accuracy. Nevertheless, the successive application of these methods is generally contingent upon the specific application needs and the stereo matching algorithm employed.

3.4.2 Machine Learning-Based Stereo Matching Algorithms

Convolutional neural networks (CNNs) have gained significant attention in the field of stereo matching for their ability in performing effective feature extraction and information aggregation. Consequently, they have become a popular choice for disparity estimation. These algorithms are primarily data-driven, requiring huge amounts of data for the CNNs to learn to perform stereo matching. CNN-based algorithms can be classified into two types: supervised and unsupervised. Supervised methods utilize disparity ground truth to guide the training and optimize CNN parameters [34, 43, 44], while unsupervised methods use geometric constraints between stereo images as supervision during training [45–47].

3.4.2.1 Supervised Approaches

Supervised stereo matching approaches can be broadly categorized into three groups: (1) learning better feature correspondences [43], (2) learning better regularization [34], and (3) learning dense disparity estimation in an end-to-end paradigm [44].

Specifically, the first group adopts the learned representations to calculate matching costs, which are subsequently subjected to cost aggregation and regularization methods for disparity estimation. For instance, Žbontar and LeCun [43] designed a

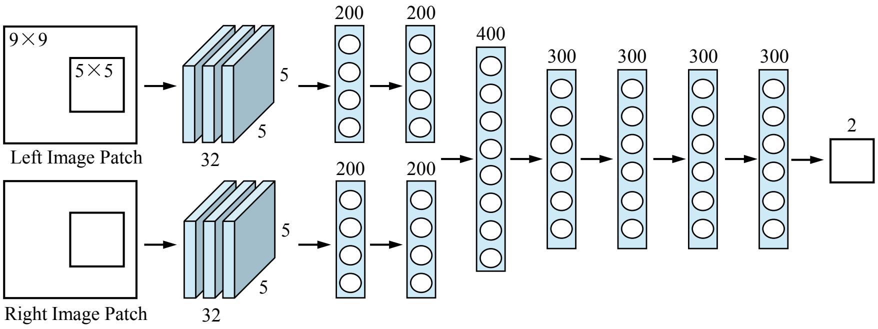


Fig. 3.7 The architecture of the CNN proposed in [43]

CNN to calculate patch-wise similarity scores, as shown in Fig. 3.7. It consists of a convolutional layer L_1 with 32 convolution kernels of size 5×5×1 and seven fully connected (FC) layers L_2–L_8 with 200, 300 or 400 neurons. Two 9×9 pixel gray-scale image patches are firstly input into a sequence of L_1, L_2 and L_3, respectively. Then, the generated 200-dimensional feature maps are concatenated into a 400-dimensional feature map, before passing through L_4–L_8. The output of layer L_8 is two real numbers, which is then fed into a softmax function to generate a distribution over two classes: (1) good match and (2) bad match. Finally, explicit programming-based cost aggregation and disparity optimization methods are deployed to generate the final disparity maps. Although these approaches achieved the state-of-the-art (SoTA) accuracy at that time, the employed explicit programming-based cost aggregation methods can produce wrong predictions in occluded or texture-less/reflective regions and therefore limit their stereo matching performance [48].

Therefore, some researchers have focused on the second category of approaches, which uses CNN to improve the cost aggregation step. Ǎkihito Seki and Marc Pollefeys [34] proposed SGM-Nets, which uses a 5 × 5-pixel gray-scale image patch as input and predicts SGM penalty parameters λ_1 and λ_2 with a CNN, as shown in Fig. 3.8. SGM-Nets consists of (1) two convolution layers, each followed by a rectified linear unit (ReLU) layer; (2) a concatenate layer to merge the two types of input; (3) two FC layers, followed by a ReLU layer and an exponential linear unit (ELU) layer, respectively; and (4) a constant layer for keeping SGM penalty values positive. Similar to SGM [33], the costs are then accumulated along four directions. In general, the outputs of SGM-Nets denote standard parameterization.

Recently, the third category of methods, i.e., end-to-end deep CNNs, has gained popularity because of their impressive performance in stereo matching tasks. For instance, GCNet [49] incorporated feature extraction (cost computation), cost aggregation and disparity optimization/refinement into a single end-to-end CNN model, as shown in Fig. 3.9. The cost volume construction method used in GCNet is different from previous methods. GCNet concatenates each feature vector with its corresponding vector extracted from the opposite stereo image across each disparity level, and packs these into a 4D volume. This approach allows GCNet to preserve more knowl-

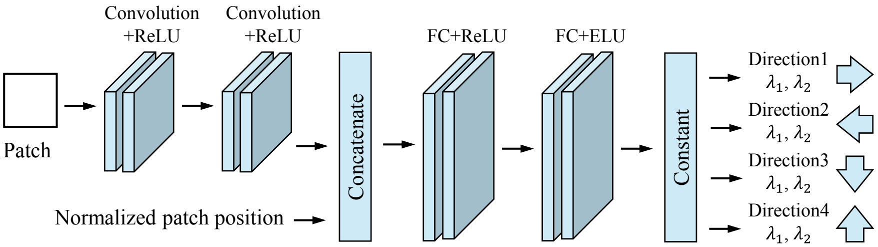


Fig. 3.8 The architecture of SGM-Nets [34]

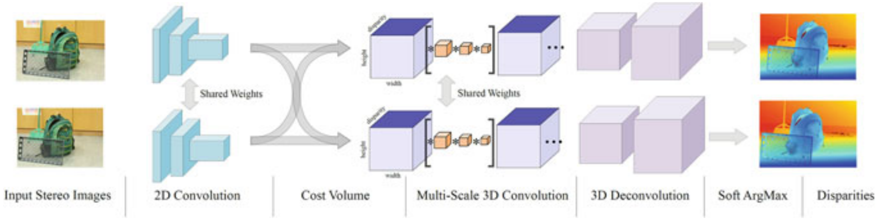


Fig. 3.9 The architecture of GCNet [49]

edge of the geometry of stereo vision, unlike previous methods. As a result, GCNet uses 3D convolution instead of 2D convolution for the disparity refinement process. Later on, Chang et al. [44] proposed Pyramid Stereo Matching Network (PSMNet), improving the feature extraction process with a spatial pyramid pooling module, and the disparity refinement process with a stacked hourglass 3D CNN, as shown in Fig. 3.10. The former enlarges the receptive fields and thus can aggregate more context information, while the latter regularizes the cost volume and further extends the regional support of context information in cost volume. In addition, Guo et al. [50] proposed group-wise correlation, which first divides the left and right features into groups and then computes correlation maps among each group. After that, the computed correlation maps are packed together for the following disparity estimation. Compared with previous cost volume construction methods by concatenation, group-wise correlation additionally introduces feature correlation into the cost volume, which avoids using more parameters to learn feature similarity in the disparity aggregation process.

In 2020, Cheng et al. [51] adopted neural architecture search techniques to obtain an effective network architecture for stereo matching. The stereo images are first processed by the Feature Net to generate visual features, which are then processed by the Matching Net to generate a 3D cost volume. The disparity map can finally be computed from the cost volume via the soft-argmin operation. Since the learnable parameters are only located in the Feature Net and the Matching Net. Cheng et al. [51] used neural architecture search techniques to select the optimal structures for them.

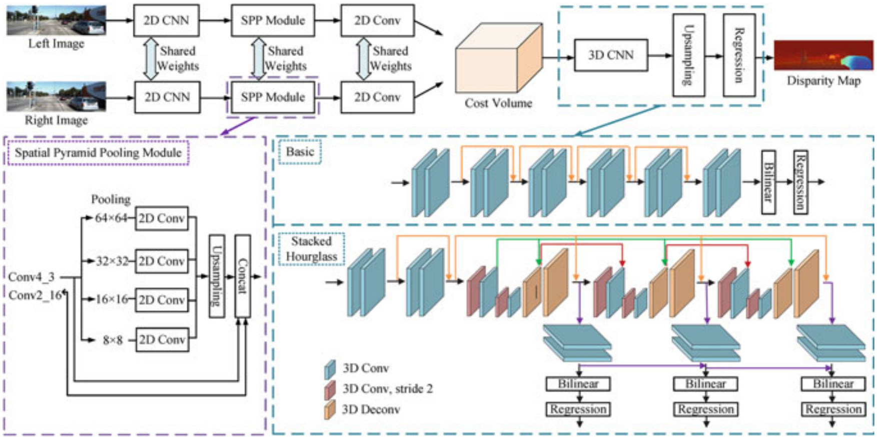


Fig. 3.10 The architecture of PSMNet [44]

Although the aforementioned end-to-end disparity estimation methods have achieved impressive results, they usually have a huge number of learnable parameters, resulting in a long inference time. To address this issue, some researchers have turned their focuses towards improving the inference speed of end-to-end methods for stereo matching. Specifically, Xu et al. [52] proposed a sparse point-based inter-scale cost aggregation module and a cross-scale cost aggregation module for efficient stereo matching. These two kinds of modules are lightweight and complementary, leading to an efficient and effective architecture for stereo matching. Inspired by RAFT [53], Wang et al. [45] presented OptStereo, which first builds multi-scale cost volumes and then iteratively updates disparity estimations at high resolution. This novel architecture can achieve a great trade-off between the accuracy and efficiency for stereo matching. The architecture of OptStereo is shown in Fig. 3.11. Later on, Lahav et al. proposed RAFT-Stereo [54], which inherits the gate recurrent unit (GRU) iteration architecture of [53], and includes a multilevel GRU to aggregate

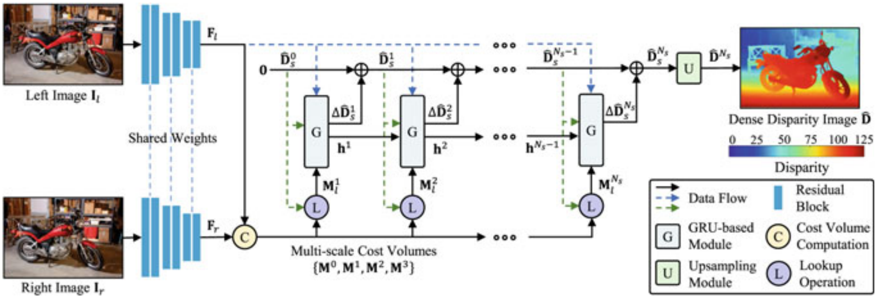


Fig. 3.11 The architecture of OptStereo [45]

feature maps at different resolutions. In 2022, Li et al. presented another recurrent structural network, CREStereo [55]. In CREStereo, the fixed-shape local search window used in previous works [45, 53, 54] is replaced with a deformable convolution operator [56] for constructing the cost volumes, which considerably improves the disparity estimation accuracy in occluded or texture-less areas. In addition, some researchers adopted the popular coarse-to-fine framework to balance the trade-off between the accuracy and efficiency for stereo matching [57–60].

3.4.2.2 Unsupervised Approaches

Although supervised approaches have achieved impressive results for disparity estimation, the collection of large amounts of disparity ground truth is time-consuming and requires significant labor. To address this issue, many researchers have focused on developing unsupervised stereo matching approaches to eliminate the need for disparity ground truth. The CNN architectures of unsupervised approaches are similar to those of supervised approaches, while the main difference lies in the network training process.

Specifically, Zhong et al. [46] proposed SsSMnet, which employs image warping error to drive the learning process, as shown in Fig. 3.12. Zhong et al. [46] leverages the following loss functions:

$$\mathcal{L} = \omega_p \left(\mathcal{L}_u^l + \mathcal{L}_u^r\right) + \omega_s \left(\mathcal{L}_s^l + \mathcal{L}_s^r\right) + \omega_c \left(\mathcal{L}_c^l + \mathcal{L}_c^r\right) + \omega_m \left(\mathcal{L}_m^l + \mathcal{L}_m^r\right), \quad (3.37)$$

where $\mathcal{L}_u^l$ and $\mathcal{L}_u^r$ denote the unary term; $\mathcal{L}_s^l$ and $\mathcal{L}_s^r$ denote the regularization term; $\mathcal{L}_c^l$ and $\mathcal{L}_c^r$ denote the consistency constraint term; $\mathcal{L}_m^l$ and $\mathcal{L}_m^r$ denote the maximization depth heuristic. The unary term is designed to minimize the discrepancy between the stereo images based on the disparity estimation, and has the following formulation:

$$\mathcal{L}_u^l\left(I_\mathrm{L}, I_\mathrm{L}'\right) = \frac{1}{N}\sum \lambda_1 \frac{1 - \mathrm{SSIM}\left(I_\mathrm{L}, I_\mathrm{L}'\right)}{2} + \lambda_2 \left|I_\mathrm{L} - I_\mathrm{L}'\right| + \lambda_3 \left|\nabla I_\mathrm{L} - \nabla I_\mathrm{L}'\right|, \quad (3.38)$$

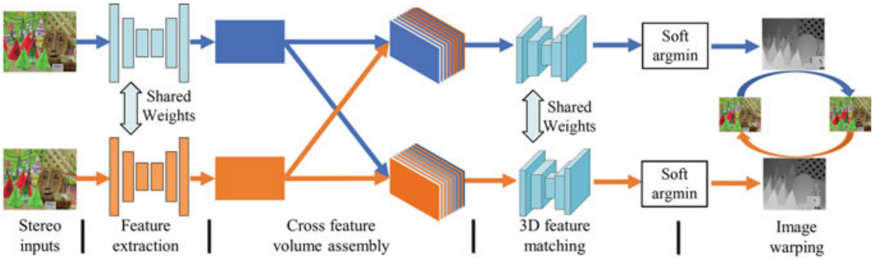


Fig. 3.12 The architecture of SsSMnet [46]

where N denotes the total number of pixels and I'_L denotes the reconstructed left image; SSIM$(\cdot)$ [61] measures the similarity between two images; λ_1, λ_2 and λ_3 are three weighting parameters for balancing the structural similarity, image appearance difference and image gradient difference. The regularization term is designed to make the disparity estimation locally smooth and is defined as follows:

$$\mathcal{L}_s^l = \frac{1}{N}\sum \left|\nabla_u^2 d_L\right| e^{-\left|\nabla_u^2 I_\mathrm{L}\right|} + \left|\nabla_v^2 d_L\right| e^{-\left|\nabla_v^2 I_\mathrm{L}\right|}. \tag{3.39}$$

The consistency constraint term is designed based on the concept of left-right consistency check, and has the following formulation:

$$\mathcal{L}_c^L = \left|I_\mathrm{L} - I''_\mathrm{L}\right|, \tag{3.40}$$

where I''_L is generated by warping the left image to the right view and warping back to the left image coordinate. Finally, the maximum depth heuristic minimizes the sum of all the disparities or maximizes the sum of all the depths, and has the following formulation:

$$\mathcal{L}_m^L = \frac{1}{N}\sum \left|d_L\right|. \tag{3.41}$$

These four terms work together to supervise the network training without the use of disparity ground truth, and are commonly used in subsequent proposed unsupervised stereo matching approaches.

Due to occlusion, certain regions in the left and right images are not commonly visible, leading to unreliable constraints for the networks' disparity estimation. This occurs as the unary term and the consistency constraint term cannot provide sufficient guidance in such areas. The unsupervised stereo matching approach presented by Zhao et al. [62] involves an random initialization of the network, followed by iterative updates of network parameters using left-right check. More specifically, in each iteration, a disparity map is predicted, which is then used to generate a confidence map based on left-right check. Pixels with confident disparity will be ignored in the next iteration. Li et al. [63] incorporated occlusion reasoning to improve the unsupervised stereo matching performance. In [63], an occlusion mask $\boldsymbol{O}$ is predicted before the disparity regression and is then used to ignore the $\mathcal{L}_u$ and $\mathcal{L}_c$ in the occluded area. The network architecture of [63] is illustrated in Fig. 3.13. Additionally, to avoid predicting an all-one occlusion map, an occlusion regularization loss is introduced, which is defined as follows:

$$\mathcal{L}_o = \frac{1}{N}\sum_{\boldsymbol{p}} \left|\boldsymbol{O}(\boldsymbol{p}) - \left(1 - e^{-|d_L(\boldsymbol{p}) - d_R(\boldsymbol{q})|}\right)\right|, \tag{3.42}$$

where $\boldsymbol{p}$ and $\boldsymbol{q}$ represent a pixel in the left disparity map and its corresponding pixel in the right disparity map, respectively. Joung et al. [64] utilized correspondence consistency between stereo images as a basis constraint for their approach, and

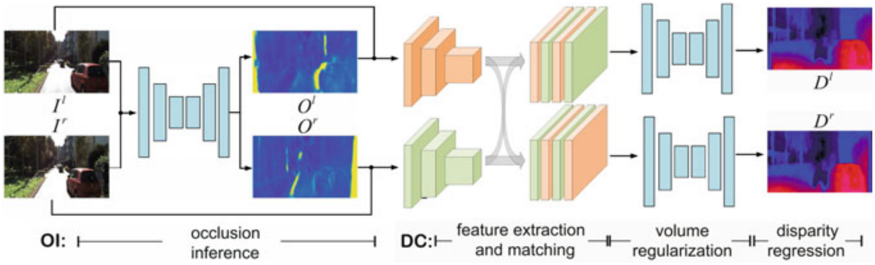


Fig. 3.13 The architecture of the occlusion-aware stereo matching network [63]

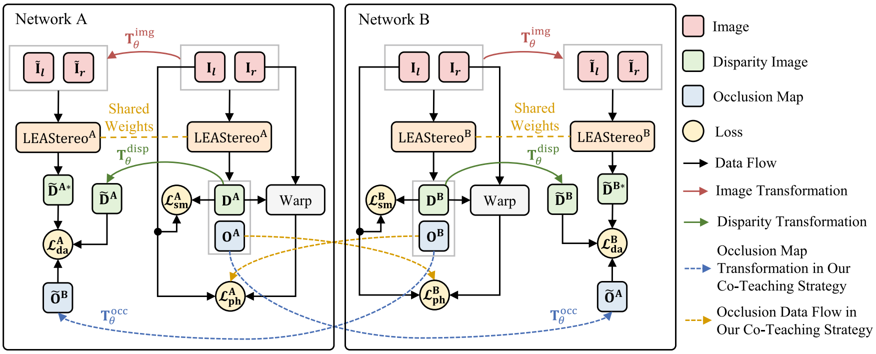


Fig. 3.14 The architecture of CoT-Stereo [47]

additionally developed a positive sample propagation scheme to enhance its stereo matching performance. In 2020, Liu et al. proposed Flow2Stereo [65], which leverages the geometric constraints presented in stereo videos to estimate disparity and optical flow simultaneously in an unsupervised manner.

However, the aforementioned unsupervised approaches still exhibit unstable performance in challenging areas, especially in regions with occlusions. This can be attributed to the sensitivity of a single network to outliers. To address this issue, Wang et al. [47] presented a co-teaching framework, where two networks teach each other about challenging regions in an unsupervised manner, as shown in Fig. 3.14. In CoT-Stereo [47], the adopted photometric loss $\mathcal{L}_{\text{ph}}$ is similar to the unary loss in (3.38), and the adopted smoothness loss is similar to the regularization loss in (3.39). To further improve the stereo matching accuracy, Wang et al. [47] adopted a data-augmentation loss, which has the following formulation [66]:

$$\mathcal{L}_{\text{da}} = \frac{\sum l\left(\left|\mathcal{S}(\widetilde{\boldsymbol{D}}) - \widetilde{\boldsymbol{D}}^*\right|\right) \cdot \mathcal{S}(\widetilde{\boldsymbol{O}})}{\sum \mathcal{S}(\widetilde{\boldsymbol{O}})},$$
$$l(x) = \begin{cases} x - 0.5, & x \geq 1 \\ x^2/2, & x < 1 \end{cases}, \tag{3.43}$$

where $\mathcal{S}(\cdot)$ denotes the stop-gradient; $\widetilde{\boldsymbol{D}}$, $\widetilde{\boldsymbol{D}}^{*}$ and $\widetilde{\boldsymbol{O}}$ denote the augmented samples. This data-augmentation paradigm can effectively enable the network to better handle occlusions. In addition, CoT-Stereo consists of a dynamic threshold selection scheme and an occlusion estimation swapping operation. The former ensures that the networks do not memorize possible outliers, while the latter enables the two networks to adaptively correct the inaccurate occlusion estimation. Inspired by [67], Fan et al. [68] proposed an occlusion-aware stereo matching algorithm based on confidence guided raw disparity fusion (CRD-Fusion). In CRD-Fusion, a correlation cost volume is firstly processed by five 3D convolutions to generate a raw disparity, which is then used to generate a raw occlusion map with consistency-based confidence measures (see Sect. 3.5.3). Afterwards, the raw disparity map and raw occlusion map are refined jointly using a hierarchical refinement module.

Another popular paradigm for unsupervised stereo matching is to generate reliable pseudo disparity ground truth for supervision. For example, Wang et al. [45] proposed OptStereo, where a pyramid voting module can generate reliable semi-dense disparity maps to supervise the CNN training. Despite recent advances in unsupervised stereo matching, there remains a significant performance gap between existing supervised and unsupervised approaches. As a result, we firmly believe that exploring more effective unsupervised training strategies is a promising direction for future research in this field.

3.5 Disparity Confidence Measures

In parallel with the rapid evolution of stereo matching algorithms, deciding the confidence of estimated disparity has also grown in popularity [69]. It is important to recognize that estimating the disparity confidence is not intended to predict potential disparity error margins. Instead, it serves as a metric for the probability that a stereo matching algorithm may fail in challenging situations such as occlusions and reflections. In other words, the disparity confidence map identifies areas that require extra attention for stereo matching.

Typically, the cost volume serves as the primary source of information for confidence measures. In this chapter, the cost volume is generated with NCC (3.30). For a left pixel $\boldsymbol{p} = [x, y]^{\top}$, its cost curve $c(\boldsymbol{p})$ is shown in Fig. 3.15, where $c_i(\boldsymbol{p})$ denotes the matching cost for disparity candidate i; $d_1(\boldsymbol{p})$, $d_2(\boldsymbol{p})$ and $d_{2m}(\boldsymbol{p})$ represent the disparity candidate possessing the minimum, the second minimum and the second smallest local minimum matching cost, respectively, and their corresponding matching costs are $c_{d_1}(\boldsymbol{p})$, $c_{d_2}(\boldsymbol{p})$ and $c_{d_{2m}}(\boldsymbol{p})$, respectively; $\boldsymbol{p}^r = [x - d_1(\boldsymbol{p}), y]^{\top}$ represents the corresponding right pixel of $\boldsymbol{p}$. However, most confidence measures only require a portion of the information in Fig. 3.15, and thus can be mainly grouped into cost-based, disparity-based, consistency-based and image-based. Example images are visualized in Fig. 3.16, where the first two columns show the different sources of information, and the last three columns present the different confidence measures.

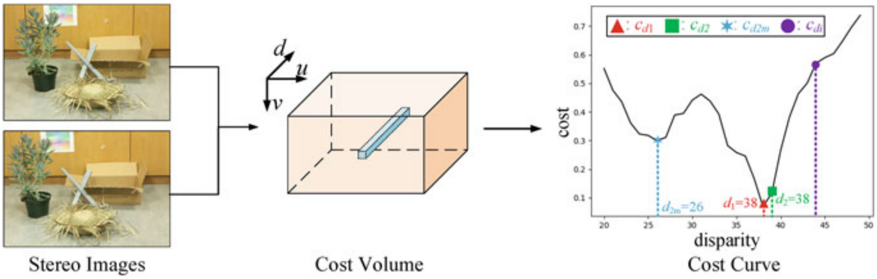


Fig. 3.15 Example of cost curve and illustrations of terms defined on the cost curve

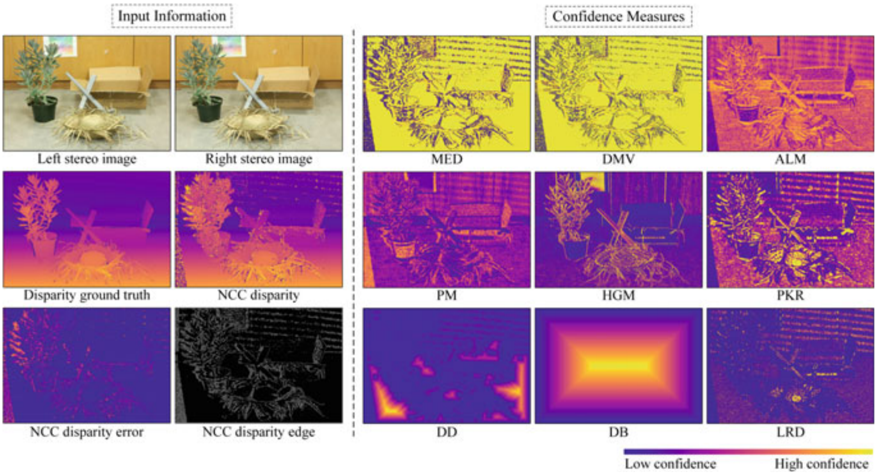


Fig. 3.16 Example images of confidence measures. The first two columns provide the sources of information, and the last three columns present the confidence measures

3.5.1 Cost-Based Confidence Measures

Cost-based confidence measures are obtained from the cost volume, mostly encoded by $c_{d_1}(\boldsymbol{p})$, $c_{d_2}(\boldsymbol{p})$ and $c_{d_{2m}}(\boldsymbol{p})$. In general, these confidence measures score a pixel with high confidence if it has a significantly lower $c_{d_1}(\boldsymbol{p})$, which indicates a strong match to its corresponding pixel in the other image.

1. The matching score metric (MSM) [70] directly uses the minimum cost $c_{d_1}(\boldsymbol{p})$ as a confidence measure, which is defined as follows:

$$\mathrm{MSM}(\boldsymbol{p}) = -c_{d_1}(\boldsymbol{p}). \tag{3.44}$$

2. Peak ratio (PKR) [70] is expressed by the ratio of $c_{d_{2m}}$ to c_{d_1}:

$$\mathrm{PKR}(\boldsymbol{p}) = \frac{c_{d_{2m}}(\boldsymbol{p})}{c_{d_1}(\boldsymbol{p})}. \tag{3.45}$$

3. The perturbation measure (PM) [71] considers the linearity of the cost curve, and is formulated as follows:

$$\mathrm{PM}(\boldsymbol{p}) = \sum_{i \neq d_1} e^{-\frac{\left[c_{d_1}(\boldsymbol{p}) - c_i(\boldsymbol{p})\right]^2}{s^2}}, \tag{3.46}$$

where s is a scaling parameter.

4. The attainable likelihood measure (ALM) [72] models the cost curve using a Gaussian distribution centered at $c_{d_1}(\boldsymbol{p})$:

$$\begin{aligned} \mathrm{ALM}(\boldsymbol{p}) &= \frac{1}{\sum_{i \in D} e^{-\frac{c_i(\boldsymbol{p})}{2\sigma(\boldsymbol{p})}}}, \\ \sigma(\boldsymbol{p}) &= \frac{\sum_{i \in D} (c_i(\boldsymbol{p}) - c_{d1}(\boldsymbol{p}))^2}{m}, \end{aligned} \tag{3.47}$$

where D represents the set of disparity candidates, and m denotes the number of disparity candidates.

3.5.2 Disparity-Based Confidence Measures

Disparity-based confidence measures use the disparity map generated with the WTA strategy from the cost volume as the input domain, without any additional cues from the cost volume.

1. The distance from discontinuities (DD) [73] metric derives confidence from the distance to a disparity discontinuity:

$$\mathrm{DD}(\boldsymbol{p}) = \min_{\boldsymbol{q} \in \hat{d}} |\boldsymbol{p} - \boldsymbol{q}|, \tag{3.48}$$

where $\hat{d}$ is obtained by applying a Canny edge detector [74] to the disparity map, as shown in row 3, column 2 of Fig. 3.16.

2. Disparity map variance (DMV) [71] measures the gradient over the disparity map:

$$\mathrm{DMV}(\boldsymbol{p}) = \left\| \nabla d_1(\boldsymbol{p}) \right\| . \tag{3.49}$$

3. Difference with median disparity (MED) [73] calculates the difference between $d_1(\boldsymbol{p})$ and the median disparity in the 5×5 windows centered at $\boldsymbol{p}$ in the disparity map.

3.5.3 Consistency-Based Confidence Measures

Consistency-based confidence measures evaluate the consistency between corresponding pixels in the stereo images based on epipolar geometry.

1. Left-right checking (LRC) [75] is a method that assumes that the disparity values at corresponding pixels in the left and right views should be consistent, except in the occluded regions. Therefore, the presence of inconsistent disparities signifies the difficulty of stereo matching, particularly in regions with occlusions. LRC is defined as follows:

$$\mathrm{LRC}(\boldsymbol{p}) = -|d_1(\boldsymbol{p}) - d_1^r(\boldsymbol{p}^r)|, \tag{3.50}$$

   where d_1^r represents the right disparity generated with the WTA strategy.
2. Left-right difference (LRD) [76] further measures the consistency of the minimum costs across the stereo images:

$$\mathrm{LRD}(\boldsymbol{p}) = \frac{c_{d2}(\boldsymbol{p}) - c_{d1}(\boldsymbol{p})}{|c_{d1}(\boldsymbol{p}) - c_{d1}^r(\boldsymbol{p}^r)|}. \tag{3.51}$$

3.5.4 Image-Based Confidence Measures

Image-based confidence measures consider only the stereo images as input.

1. Distance from border (DB) [73] calculates the distance between a pixel and its nearest image border, assuming that pixels close to the image border are more likely to be occluded or unseen in the other reference view. DB is defined as follows:

$$\mathrm{DB}(\boldsymbol{p}) = \min(x, y, W - x, H - y), \tag{3.52}$$

   where W and H represent the image width and height, respectively.

2. Horizontal gradient magnitude (HGM) [71] is based on the assumption that a higher image gradient indicates a richer texture, and thus a higher matching confidence. Similar to DMV, HGM can be formulated as follows:

$$\mathrm{HGM}(\boldsymbol{p}) = \|\nabla_x I_{\mathrm{L}}(\boldsymbol{p})\|, \tag{3.53}$$

   where $I_{\mathrm{L}}(\boldsymbol{p})$ represents the image intensity at pixel $\boldsymbol{p}$ in the left image.

3.6 Evaluation Metrics

The performance evaluation of a given stereo matching algorithm usually involves measuring both the accuracy and efficiency of the disparity estimation [25]. Commonly used metrics to evaluate the accuracy of an estimated disparity map are listed below [77]:

1. Root-mean-squared error (RMS), e_{RMS} is defined as follows [78]:

$$e_{\mathrm{RMS}} = \sqrt{\frac{1}{N}\sum_{\boldsymbol{p}\in\mathscr{P}} |\boldsymbol{D}_{\mathrm{E}}(\boldsymbol{p}) - \boldsymbol{D}_{\mathrm{G}}(\boldsymbol{p})|^2}, \tag{3.54}$$

   where $\boldsymbol{D}_{\mathrm{E}}$ and $\boldsymbol{D}_{\mathrm{G}}$ denote the estimated and ground truth disparity maps, respectively; $\mathscr{P}$ represents the set of disparities used for evaluation, and N denotes the size of $\mathscr{P}$.
2. End-point error (EPE), e_{EPE}, is expressed as the average disparity estimation error across all pixels [77]:

$$e_{\mathrm{EPE}} = \frac{1}{N}\sum_{\boldsymbol{p}\in\mathscr{P}} |\boldsymbol{D}_{\mathrm{E}}(\boldsymbol{p}) - \boldsymbol{D}_{\mathrm{G}}(\boldsymbol{p})|. \tag{3.55}$$

3. Percentage of error pixels (PEP), e_{PEP}, is given by [42]:

$$e_{\mathrm{PEP}} = \frac{1}{N}\sum_{\boldsymbol{p}\in\mathscr{P}} \delta\Big(|\boldsymbol{D}_{\mathrm{E}}(\boldsymbol{p}) - \boldsymbol{D}_{\mathrm{G}}(\boldsymbol{p})| > \delta_d\Big) \times 100\%, \tag{3.56}$$

   where δ_d represents the disparity evaluation tolerance.
4. D1-all, $e_{\text{D1-all}}$ takes into consideration the magnitude of disparity ground truth, and is expressed as [79]:

$$e_{\text{D1-all}} = \frac{1}{N}\sum_{\boldsymbol{p}\in\mathscr{P}} \delta\Big(|\boldsymbol{D}_{\mathrm{E}}(\boldsymbol{p}) - \boldsymbol{D}_{\mathrm{G}}(\boldsymbol{p})| > \max\{0.05 \times \boldsymbol{D}_{\mathrm{G}}(\boldsymbol{p}), 3\}\Big) \times 100\%. \tag{3.57}$$

5. The η error quantile represents the highest disparity error among the given percentage η of best matched pixels. For example, for $\eta = 50\%$, this is the median error.

In terms of evaluating the efficiency of disparity estimation, the most common approach is to measure the runtime processing per stereo image of the algorithm being evaluated. However, the runtime can vary depending on factors such as image resolution and disparity search range. Therefore, a more robust metric for evaluating stereo matching efficiency is Mde/s, which is given in millions of disparity evaluations per second [25]:

$$\mathrm{Mde}/s = \frac{u_{\max} v_{\max} d_{\max}}{t} 10^{-6}. \tag{3.58}$$

Similarly, time/MP and time/GD measure the runtime in megapixels and gigadisparities [80], respectively.

3.7 Public Datasets and Benchamarks

Open-access datasets and benchmarks have been developed in conjunction with the measurable progress in stereo matching algorithms. These datasets cover different application scenarios, such as driving scenes [14, 81], indoor and outdoor scenes [16, 38]. Table 3.1 summarizes the existing datasets, and we provide a detailed explanation of some of the most commonly used benchmarks to assist researchers in making their choices.

3.7.1 Middlebury Benchmark

The Middlebury dataset [15, 38, 42, 88] offers high-resolution stereo images of static indoor scenes and ground truth disparity maps with high accuracy, as depicted in Fig. 3.17. The disparity ground truth is generated using a structured light acquisition system, and Middlebury 2014 [15] attains a disparity accuracy of 0.2 pixels on most observed surfaces, including in half-occluded regions. Although only dozens of training samples are provided, its highly accurate ground truth disparity and online evaluation interface make it one of the most widely used benchmarks in stereo matching. Table 3.2 displays the benchmark results of the above-mentioned stereo matching algorithms on the Middlebury 2014 dataset [15].

Table 3.1 Critical attributes of commonly adopted publicly available datasets

Dataset	Size		Resolution	Source	Scene
	Training set	Testing set			
Middlebury 2014 [a] [15]	23	10	2964 × 2000	Real-world	Indoor
Sintel [b] [82]	1064	564	1024 × 436	Synthetic	Hybrid
NYU V2 [c] [83]	1449	–	640 × 480	Real-world	Indoor
KITTI Stereo 2012 [d] [81]	194	195	1226 × 379	Real-world	Driving
KITTI Stereo 2015 [e] [14]	200	200	1242 × 375	Real-world	Driving
Virtual KITTI [f] [84]	21260	–	1242 × 375	Synthetic	Driving
Virtual KITTI 2 [g] [85]	21260	–	1242 × 375	Synthetic	Driving
Scene Flow [h] [17]	34801	4248	960 × 540	Synthetic	Hybrid
ETH3D [i] [16]	27	20	940 × 490	Real-world	Synthetic
Falling Things [j] [86]	61500	–	960 × 540	Synthetic	Indoor
Driving Stereo [k] [87]	174437	7751	1762 × 800	Real-world	Indoor

[a] https://vision.middlebury.edu/stereo/data/scenes2014/
[b] http://sintel.is.tue.mpg.de/stereo
[c] https://cs.nyu.edu/ silberman/datasets/nyu_depth_v2.html
[d] https://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo
[e] https://www.cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=stereo
[f] https://europe.naverlabs.com/research/computer-vision/proxy-virtual-worlds-vkitti-1/
[g] https://europe.naverlabs.com/research/computer-vision/proxy-virtual-worlds-vkitti-2/
[h] https://lmb.informatik.uni-freiburg.de/resources/datasets/SceneFlowDatasets.en.html
[i] https://www.eth3d.net/low_res_two_view
[j] https://research.nvidia.com/publication/2018-06_falling-things-synthetic-dataset-3d-object-detection-and-pose-estimation
[k] https://drivingstereo-dataset.github.io/

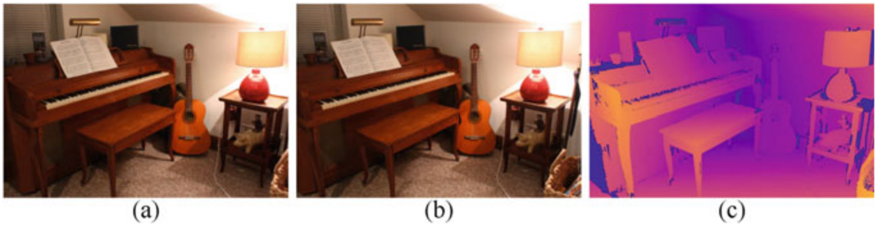


Fig. 3.17 Example images of middlebury 2014 dataset [15]. **a** Left stereo image; **b** right stereo image; **c** ground truth disparity map

Table 3.2 Online evaluation results of noc areas on Middlebury benchmark

Method	PEP (%) ↓		EPE (px) ↓	RMS (px) ↓	A50[a] (px) ↓	Time/MP (s) ↓	Time/GD (s) ↓
	$\delta_d = 1$	$\delta_d = 4$					
PSMNet [44]	63.9	23.5	6.68	19.4	1.94	2.62	32.2
GANet [48]	37.8	11.2	12.2	35.4	0.83	6.33	16.4
AANet [52]	25.5	10.8	6.37	23.5	0.56	2.48	7.07
LEAStereo [51]	20.8	2.75	1.43	8.11	0.53	2.53	7.27
AdaStereo [89]	29.5	6.35	2.22	10.2	0.65	0.13	0.38
HITNet [59]	13.3	3.81	1.71	9.97	0.40	0.11	0.29
RAFT-Stereo [54]	9.37	2.75	1.27	8.41	0.26	2.19	5.76
CREStereo [55]	8.25	2.04	1.15	7.70	0.28	0.77	2.22

[a] A50 represents 50% error quantile

3.7.2 KITTI Benchmark

The KITTI dataset [14, 81] was captured in rural and highway areas of Karlsruhe and has been commonly used in a variety of visul tasks [90], as illustrated in Fig. 3.18a, b. It comprises a rectified stereo vision system with two high-resolution color and grayscale video cameras mounted on a wagon, providing stereo images, and a Velodyne laser scanner to offer sparse ground truth disparity, as displayed in Fig. 3.18. The KITTI Stereo 2012 dataset [81] and KITTI Stereo 2015 dataset [14] both contain about 400 image pairs with accurate disparity ground truth. The KITTI dataset has been of great interest to a wide range of researchers in the field of autonomous driving due to sufficient samples for training CNNs.

Disparities for pixels that are visible in only one view due to occlusion cannot be directly obtained from stereo images, but they can be obtained from depth information (3.26). Therefore, the KITTI dataset provides both occluded (occ) and

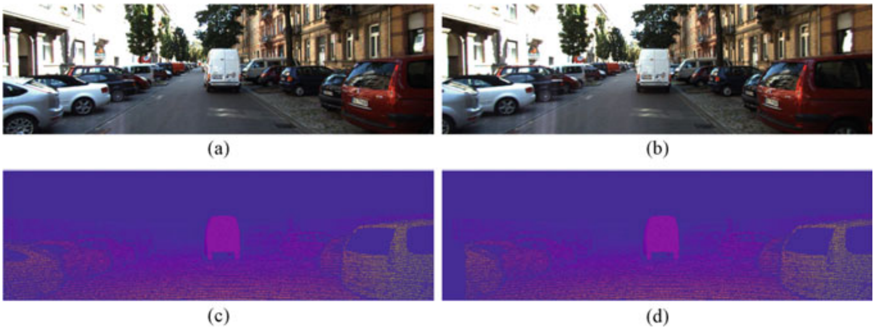


Fig. 3.18 Example images of KITTI Stereo 2015 dataset [14]. **a** Left stereo image; **b** right stereo image; **c** ground truth occluded (occ) disparity map; **d** ground truth non-occluded (noc) disparity map

Table 3.3 Online evaluation results of noc areas on KITTI Stereo 2012 dataset

Method	PEP (%) ↓				EPE (px) ↓	Runtime (s) ↓	Environment
	$\delta_d = 2$	$\delta_d = 3$	$\delta_d = 4$	$\delta_d = 5$			
GCnet [49]	2.71	1.77	–	1.12	0.6	0.90	Nvidia Titan Xp
PSMNet [44]	2.44	1.49	1.12	0.90	0.5	0.41	Nvidia Titan Xp
GANet [48]	1.89	1.19	0.91	0.76	0.4	1.80	GPU@2.5Ghz
GWCNet [50]	2.12	1.32	0.99	0.80	0.5	0.32	GPU@2.0Ghz
AANet [52]	2.30	1.55	1.20	0.98	0.4	0.06	NVIDIA V100
HITNet [59]	2.00	1.41	1.14	0.96	0.5	0.02	GPU@2.5Ghz
LEAStereo [51]	1.90	1.13	0.83	0.67	0.5	0.30	GPU@2.5Ghz
OptStereo [45]	1.91	1.20	0.92	0.77	0.4	0.10	GPU@2.5Ghz
SCV-Stereo [91]	2.01	1.27	0.97	0.81	0.5	0.08	GPU@2.5Ghz
CRD-Fusion [68]	5.07	3.35	2.66	2.26	0.9	0.02	GPU@2.5Ghz
CREStereo [55]	1.72	1.14	0.90	0.76	0.4	0.40	GPU@3.5Ghz

non-occluded (noc) ground truth disparity maps. The disparities of occluded areas are ignored in the latter, as shown in Fig. 3.18c, d. It is important to note that disparities in reflective areas such as glasses are not included. Generally, machine learning-based algorithms, which have greater generalization ability compared to explicit programming-based algorithms, are more likely to achieve better performance when using the occ ground truth disparity maps. The benchmark results for the

Table 3.4 Online evaluation results on KITTI Stereo 2015 dataset

Method	D1-all (%) ↓		Runtime (s) ↓	Environment
	occ areas	noc areas		
GCnet [49]	2.87	2.61	0.90	Nvidia Titan Xp
PSMNet [44]	2.32	2.14	0.41	Nvidia Titan Xp
GANet [48]	1.81	1.63	1.80	GPU@2.5Ghz
GWCNet [50]	2.11	1.92	0,32	GPU@2.0Ghz
AANet [52]	2.03	1.85	0.06	NVIDIA V100
HITNet [59]	1.98	1.74	0.02	GPU@2.5Ghz
LEAStereo [51]	1.65	1.51	0.30	GPU@2.5Ghz
OptStereo [45]	1.82	1.64	0.10	GPU@2.5Ghz
SCV-Stereo [91]	2.02	1.84	0.08	GPU@2.5Ghz
CRD-Fusion [68]	6.11	5.69	0.02	GPU@2.5Ghz
CREStereo [55]	1.69	1.54	0.41	GPU@3.5Ghz

Table 3.5 Online evaluation results of noc areas on ETH3D benchmark

Method	PEP (%) ↓		EPE (px) ↓	RMS (px) ↓	A50 (px) ↓	Runtime (s) ↓
	$\delta_d = 1$	$\delta_d = 4$				
PSMNet [44]	5.02	0.41	0.33	0.66	0.21	0.54
GANet [48]	6.22	0.55	0.36	0.75	0.21	1.00
GWCNet [50]	6.42	0.50	0.35	0.69	0.20	0.12
AANet [52]	5.01	0.75	0.31	0.68	0.16	1.26
AdaStereo [89]	3.09	0.20	0.24	0.44	0.15	0.40
HITNet [59]	2.79	0.19	0.20	0.46	0.10	0.02
RAFT-Stereo [54]	2.44	0.15	0.18	0.36	0.10	0.81
CREStereo [55]	0.98	0.10	0.13	0.28	0.09	–

KITTI Stereo 2012 dataset and KITTI Stereo 2015 dataset are presented in Tables 3.3 and 3.4, respectively.

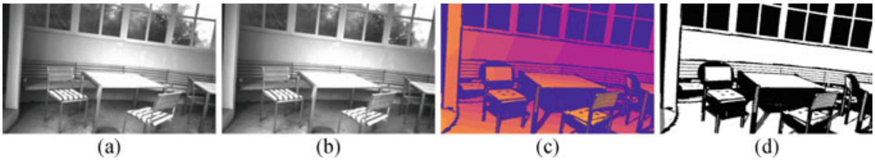


Fig. 3.19 Example images of eth3d benchmark [16]. **a** Left stereo image; **b** right stereo image; **c** ground truth disparity map; **d** noc mask

3.7.3 ETH3D Benchmark

The ETH3D benchmark [16] comprises grayscale stereo images of both indoor and outdoor static scenes, including 27 training image pairs and 20 test image pairs as depicted in Fig. 3.19. The high-precision laser scanner is utilized to generate the ground truth disparity maps. Furthermore, an occlusion mask is provided for its training image pairs, encompassing non-commonly visible areas and intractable areas such as glasses in the stereo matching task, as demonstrated in Fig. 3.19d. The benchmark results on the ETH3D dataset are presented in Table 3.5.

3.8 Existing Challenges

3.8.1 Unsupervised Training

Earlier research has indicated that CNN-based stereo matching algorithms can effectively solve disparity estimation [92]. However, such data-driven algorithms require a significant amount of disparity ground truth, which is challenging to obtain in real-world scenarios. Due to the absence of large-scale stereo datasets with disparity ground truth, it is challenging to generalize supervised stereo matching algorithms to novel settings [93]. Consequently, many researchers are now focusing on unsupervised stereo matching algorithms that predict depth maps directly from the intensity images, as discussed in Sect. 3.4.2.2. Although there have been significant advances in unsupervised stereo matching algorithms, there is still a considerable gap in disparity estimation accuracy compared to supervised stereo matching algorithms [18].

3.8.2 Domain Adaptation

Another approach to reduce the reliance on disparity ground truth is domain adaptation. These algorithms can generalize stereo matching knowledge learned from synthetic datasets [17, 94, 95] to unseen scenes. DSMNet [94] proposed the domain-invariant normalization (DN) method, which normalizes feature maps along both the spatial dimension and the channel axis to enhance the feature maps' invariance to domain differences. DSMNet also proposed a structure-preserving graph-based filter (SGF) to reduce sensitivity to local textures. Song et al. [89] further improved domain adaptation by performing image-level alignment, feature-level alignment, and output-space alignment using a color transfer method, a cost volume norm layer, and an auxiliary occlusion-aware task, respectively. In 2022, Zhang et al. [96] introduced the stereo selective whitening loss to better preserve feature consistency between domains.

3.8.3 Trade-Off Between Speed and Accuracy

Recent advances in parallel computing architectures and microcomputers have made stereo matching algorithms widely deployable on intelligent devices, such as autonomous cars. However, resource-limited hardware requires a balanced trade-off between speed and accuracy for stereo matching algorithms, which are often in opposition to each other [25]. Therefore, researchers are actively working on improving the real-time performance of stereo matching algorithms. For instance, AANet [52] replaces 3D convolutions commonly used in deep CNN-based algorithms and achieves a drastic reduction in the amount of calculation. Furthermore, in the widely adopted recurrent networks [45, 54, 55], the trade-off between accuracy and efficiency can be easily achieved with early stopping.

3.8.4 Intractable Areas

Middlebury 2002 dataset [42] has put extra emphasis on the texture-less areas and occluded areas for making the process of disparity estimation more difficult. In texture-less areas, a lack of visual features can result in pixel mismatches, which is typically addressed using the disparity smoothness constraint (3.39) [97, 98]. On the other hand, occluded areas pose a challenge due to the absence of corresponding information in the stereo images. To address this, CNN-based approaches and confidence measures discussed in Sect. 3.5 can be used to solve an auxiliary occlusion-aware task. Looser constraints are given within occluded areas during the training stage of stereo matching networks [21, 68, 89, 99]. Kim et al. [100] trained a cost aggregation network and a confidence estimation network in an adversarial structure to jointly learn disparity and confidence estimation. In 2022, [101] associated occlusions with disparity discontinuity and generated an occlusion mask from the disparity map using Sobel filter [102]. Additionally, [103] introduced an auxiliary semantic segmentation task to assist in guiding the network training for handling occlusion boundaries.

3.9 Summary

This chapter provided an overview of stereo vision and the stereo matching task. We discussed both explicit programming-based and machine learning-based stereo matching algorithms, as well as auxiliary disparity confidence measures. We also covered the metrics for evaluating disparity estimation and public stereo matching benchmarks. Finally, we summarized existing stereo matching challenges, providing researchers with valuable starting points for further studies.

Acknowledgements This work was supported by the National Key R&D Program of China under Grant 2020AAA0108100, the National Natural Science Foundation of China under Grant 62233013, and the Science and Technology Commission of Shanghai Municipal under Grant 22511104500.

Appendix

A Lie Group $SO(3)$

In the three-dimensional space, the coordinates of a 3D point in two coordinate systems are $\boldsymbol{x}_1 = [x_1, y_1, z_1]^\top$ and $\boldsymbol{x}_2 = [x_2, y_2, z_2]^\top \in \mathbb{R}^{3\times 1}$, respectively. $\boldsymbol{x}_1$ can be transformed into $\boldsymbol{x}_2$ using a rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times 3}$ and a translation vector $\boldsymbol{t} \in \mathbb{R}^{3\times 1}$:

$$\boldsymbol{x_2} = \boldsymbol{R}\boldsymbol{x_1} + \boldsymbol{t}, \tag{A.1}$$

where $\boldsymbol{R}$ satisfies orthogonality:

$$\boldsymbol{R}^\top = \boldsymbol{R}^{-1} \text{ and } |\det(\boldsymbol{R})| = 1, \tag{A.2}$$

where $\det(\boldsymbol{R})$ represents the determinant of $\boldsymbol{R}$. The subgroup of orthogonal matrices with $\det(\boldsymbol{R}) = +1$ is referred to as a *special orthogonal group* and is denoted as $SO(3)$.

B Skew-Symmetric Matrix

In linear algebra, a skew-symmetric matrix $\boldsymbol{A}$ satisfies the following property:

$$\boldsymbol{A} \text{ is a skew-symmetrix matrix} \Leftrightarrow \boldsymbol{A}^\top = -\boldsymbol{A}. \tag{B.1}$$

In 3D computer vision, the skew-symmetric matrix $[\boldsymbol{a}]_\times$ of a vector $\boldsymbol{a} = [a_1, a_2, a_3]$ is defined as [104]:

$$[\boldsymbol{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \tag{B.2}$$

The two properties of a skew-symmetric matrix are as follows:

$$\boldsymbol{a}^\top [\boldsymbol{a}]_\times = \boldsymbol{0}^\top, \ [\boldsymbol{a}]_\times \boldsymbol{a} = \boldsymbol{0}, \tag{B.3}$$

where $\boldsymbol{0} = [0, 0, 0]^\top$ is a zero vector. Furthermore, a skew-symmetric matrix can also represent cross product as matrix multiplication. Specifically, for two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, their cross product can be expressed as [104]:

$$\boldsymbol{a} \times \boldsymbol{b} = [\boldsymbol{a}]_\times \boldsymbol{b} = -[\boldsymbol{b}]_\times \boldsymbol{a}. \tag{B.4}$$

References

1. Zhou K et al (2020) Review of stereo matching algorithms based on deep learning. Comput Intell Neurosci 2020
2. Fan R et al (2022) Learning collision-free space detection from stereo images: homography matrix brings better data augmentation. IEEE/ASME Trans Mechatron 27(1):225–233
3. Wang H et al (2022) UnDAF: a general unsupervised domain adaptation framework for disparity or optical flow estimation. In: 2022 international conference on robotics and automation (ICRA). IEEE, pp 01–07
4. Ozgunalp U et al (2017) Multiple lane detection algorithm based on novel dense vanishing point estimation. IEEE Trans Intell Transp Syst 18(3):621–632
5. Duan R et al (2022) Stereo orientation prior for uav robust and accurate visual odometry. IEEE/ASME Trans Mechatron 27(5):3440–3450
6. Fan R et al (2020) Pothole detection based on disparity transformation and road surface modeling. IEEE Trans Image Process 29:897–908
7. Ma N et al (2022) Computer vision for road imaging and pothole detection: a state-of-the-art review of systems and algorithms. Transp Safety Environ 4(4):tdac026
8. Fan R et al (2021) Graph attention layer evolves semantic segmentation for road pothole detection: a benchmark and algorithms. IEEE Trans Image Process 30:8144–8154
9. Fan R, Liu M (2020) Road damage detection based on unsupervised disparity map segmentation. IEEE Trans Intell Transp Syst 21(11):4906–4911
10. Sicen G et al (2023) Road environment perception for safe and comfortable driving. Springer submitted for publication
11. Wang H et al (2022) Dynamic fusion module evolves drivable area and road anomaly detection: a benchmark and algorithms. IEEE Trans Cybern 52(10):10 750–10 760
12. Fan R et al (2020) SNE-RoadSeg: incorporating surface normal information into semantic segmentation for accurate freespace detection. In: Computer vision-ECCV (2020) 16th European conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX 16. Springer, pp 340–356
13. Shean DE et al (2016) An automated, open-source pipeline for mass production of digital elevation models (DEMs) from very-high-resolution commercial stereo satellite imagery. ISPRS J Photogramm Remote Sens 116:101–117
14. Menze M et al (2015) Object scene flow for autonomous vehicles. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 3061–3070

15. Scharstein D et al (2014) High-resolution stereo datasets with subpixel-accurate ground truth. In: German conference on pattern recognition (GVPR). Springer, pp 31–42
16. Schops T et al (2017) A multi-view stereo benchmark with high-resolution images and multi-camera videos. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 3260–3269
17. Mayer N et al (2016) A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 4040–4048
18. Hamid MS et al (2022) Stereo matching algorithm based on deep learning: a survey. J King Saud Univ-Comput Inf Sci 34(5):1663–1673
19. Bendig K et al (2022) Self-superflow: self-supervised scene flow prediction in stereo sequences. In: Proceedings of the IEEE international conference on image processing (ICIP). IEEE, pp 481–485
20. Lee M-J et al (2022) Refinement of inverse depth plane in textureless and occluded regions in a multiview stereo matching scheme. J Sens 2022
21. Gidaris S et al (2018) Unsupervised representation learning by predicting image rotations. arXiv:1803.07728
22. Trucco E et al (1998) Introductory techniques for 3-D computer vision, vol 201 . Prentice Hall Englewood Cliffs
23. Fan R et al (2023) Computer stereo vision for autonomous driving: theory and algorithms. In: Recent advances in computer vision applications using parallel processing. Springer, pp 41–70
24. Loop C et al (1999) Computing rectifying homographies for stereo vision. In: Proceedings of IEEE computer society conference on computer vision and pattern recognition (CVPR), vol 1. IEEE, pp 125–131
25. Tippetts B et al (2016) Review of stereo vision algorithms and their suitability for resource-limited systems. J Real-Time Image Proc 11(1):5–25
26. Ding J et al (2021) High-accuracy recognition and localization of moving targets in an indoor environment using binocular stereo vision. ISPRS Int J Geo Inf 10(4):234
27. Fan R et al (2018) Road surface 3D reconstruction based on dense subpixel disparity map estimation. IEEE Trans Image Process 27(6):3025–3035
28. Luo W et al (2016) Efficient deep learning for stereo matching. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 5695–5703
29. Hamzah RA et al (2016) Literature survey on stereo vision disparity map algorithms. J Sens 2016
30. Yamaguchi K et al (2012) Continuous markov random fields for robust stereo estimation. In: European conference on computer vision (ECCV). Springer, pp 45–58
31. Boykov Y et al (2001) Fast approximate energy minimization via graph cuts. IEEE Trans Pattern Anal Mach Intell 23(11):1222–1239
32. Brown MZ et al (2003) Advances in computational stereo. IEEE Trans Pattern Anal Mach Intell 25(8):993–1008
33. Hirschmuller H (2007) Stereo processing by semiglobal matching and mutual information. IEEE Trans Pattern Anal Mach Intell 30(2):328–341
34. Seki A et al (2017) SGM-Nets: semi-global matching with neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 231–240
35. Spangenberg R et al (2013) Weighted semi-global matching and center-symmetric census transform for robust driver assistance. In: International conference on computer analysis of images and patterns (CAIP). Springer, pp 34–41
36. Žbontar J et al (2016) Stereo matching by training a convolutional neural network to compare image patches. J Mach Learn Res 17(1):2287–2318
37. Hirschmuller H et al (2008) Evaluation of stereo matching costs on images with radiometric differences. IEEE Trans Pattern Anal Mach Intell 31(9):1582–1599
38. Scharstein D et al (2003) High-accuracy stereo depth maps using structured light. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition (CVPR), vol 1. IEEE, pp 195–202

39. Yang Q et al (2008) Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling. IEEE Trans Pattern Anal Mach Intell 31(3):492–504
40. Fan R et al (2019) Real-time dense stereo embedded in a UAV for road inspection. In: IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW), pp 535–543
41. Mattoccia S (2011) Stereo vision: algorithms and applications, vol 22. University of Bologna
42. Scharstein D et al (2002) A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int J Comput Vision 47(1):7–42
43. Zbontar J et al (2015) Computing the stereo matching cost with a convolutional neural network. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 1592–1599
44. Chang J-R et al (2018) Pyramid stereo matching network. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 5410–5418
45. Wang H et al (2021) PVStereo: pyramid voting module for end-to-end self-supervised stereo matching. IEEE Robot Autom Lett 6(3):4353–4360
46. Zhong Y et al (2017) Self-supervised learning for stereo matching with self-improving ability. arXiv:1709.00930
47. Wang H et al (2021) Co-teaching: an ark to unsupervised stereo matching. In: Proceedings of the IEEE international conference on image processing (ICIP). IEEE, pp 3328–3332
48. Zhang F et al (2019) GA-Net: guided aggregation net for end-to-end stereo matching. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 185–194
49. Kendall A et al (2017) End-to-end learning of geometry and context for deep stereo regression. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp 66–75
50. Guo X et al (2019) Group-wise correlation stereo network. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 3273–3282
51. Cheng X et al (2020) Hierarchical neural architecture search for deep stereo matching. Adv Neural Inf Process Syst 33:22 158–22 169
52. Xu H et al (2020) AANet: adaptive aggregation network for efficient stereo matching. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 1959–1968
53. Teed Z et al (2020) RAFT: recurrent all-pairs field transforms for optical flow. In: European conference on computer vision (ECCV). Springer, pp 402–419
54. Lipson L et al (2021) RAFT-stereo: multilevel recurrent field transforms for stereo matching. In: Proceedings of the IEEE international conference on 3D vision (3DV). IEEE, pp 218–227
55. Li J et al (2022) Practical stereo matching via cascaded recurrent network with adaptive correlation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 16 263–16 272
56. Zhu X et al (2019) Deformable ConvNets V2: more deformable, better results. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 9308–9316
57. Wang Y et al (2019) Anytime stereo image depth estimation on mobile devices. In: Proceedings of the IEEE international conference on robotics and automation (ICRA). IEEE, pp 5893–5900
58. Yee K et al (2020) Fast deep stereo with 2D convolutional processing of cost signatures. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision (WACV), pp 183–191
59. Tankovich V et al (2021) HitNet: hierarchical iterative tile refinement network for real-time stereo matching. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 14 362–14 372
60. Chen S et al (2023) Feature enhancement network for stereo matching. Image Vis Comput 131:104614
61. Wang Z et al (2004) Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 13(4):600–612
62. Zhou C et al (2017) Unsupervised learning of stereo matching. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp 1567–1575

63. Li A et al (2018) Occlusion aware stereo matching via cooperative unsupervised learning. In: Asian conference on computer vision (ACCV). Springer, pp 197–213
64. Joung S et al (2019) Unsupervised stereo matching using confidential correspondence consistency. IEEE Trans Intell Transp Syst 21(5):2190–2203
65. Liu P et al (2020) Flow2Stereo: effective self-supervised learning of optical flow and stereo matching. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 6648–6657
66. Liu L et al (2020) Learning by analogy: reliable supervision from transformations for unsupervised optical flow estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 6489–6498
67. Khamis S et al (2018) StereoNet: guided hierarchical refinement for real-time edge-aware depth prediction. In: European conference on computer vision (ECCV). Springer, pp 573–590
68. Fan X et al (2022) Occlusion-aware self-supervised stereo matching with confidence guided raw disparity fusion. In: Proceedings of the conference on robots and vision (CRV). IEEE, pp 132–139
69. Poggi M et al (2021) On the confidence of stereo matching in a deep-learning era: a quantitative evaluation. arXiv:2101.00431
70. Egnal G et al (2004) A stereo confidence metric using single view imagery with comparison to five alternative approaches. Image Vis Comput 22(12):943–957
71. Haeusler R et al (2013) Ensemble learning for confidence measures in stereo vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 305–312
72. Matthies L (1992) Stereo vision for planetary rovers: stochastic modeling to near real-time implementation. Int J Comput Vision 8(1):71–91
73. Spyropoulos A et al (2014) Learning to detect ground control points for improving the accuracy of stereo matching. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 1621–1628
74. Ding L et al (2001) On the canny edge detector. Pattern Recogn 34(3):721–725
75. Egnal G et al (2002) Detecting binocular half-occlusions: empirical comparisons of five approaches. IEEE Trans Pattern Anal Mach Intell 24(8):1127–1133
76. Hu X et al (2012) A quantitative evaluation of confidence measures for stereo vision. IEEE Trans Pattern Anal Mach Intell 34(11):2121–2133
77. Barron JL et al (1994) Performance of optical flow techniques. Int J Comput Vision 12:43–77
78. Szeliski R (1999) Prediction error as a quality metric for motion and stereo. In: Proceedings of the IEEE international conference on computer vision (ICCV), vol 2. IEEE, pp 781–788
79. Baker S et al (2011) A database and evaluation methodology for optical flow. Int J Comput Vis 92:1–31
80. Kalarot R et al (2011) Analysis of real-time stereo vision algorithms on gpu. In: International conference on image and vision computing New Zealand (IVCNZ), p 1
81. Geiger A et al (2012) Are we ready for autonomous driving? The kitti vision benchmark suite. In: Conference on computer vision and pattern recognition (CVPR)
82. Butler DJ et al (2012) A naturalistic open source movie for optical flow evaluation. In: European conference on computer vision (ECCV). Springer, pp 611–625
83. Silberman N et al (2012) Indoor segmentation and support inference from rgbd images. In: European conference on computer vision (ECCV). Springer, pp 746–760
84. Gaidon A et al (2016) Virtual worlds as proxy for multi-object tracking analysis. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 4340–4349
85. Cabon Y et al (2020) Virtual KITTI 2. arXiv:2001.10773
86. Tremblay J et al (2018) Falling things: a synthetic dataset for 3d object detection and pose estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops (CVPRW), pp 2038–2041
87. Yang G et al (2019) DrivingStereo: a large-scale dataset for stereo matching in autonomous driving scenarios. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 899–908

88. Hirschmuller H et al (2007) Evaluation of cost functions for stereo matching. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). IEEE, pp 1–8
89. Song X et al (2021) AdaStereo: a simple and efficient approach for adaptive stereo matching. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 10 328–10 337
90. Jingwei Y et al (2023) Semantic segmentation for autonomous driving. Springer submitted for publication
91. Wang H et al (2021) SCV-Stereo: learning stereo matching from a sparse cost volume. In: Proceedings of the IEEE international conference on image processing (ICIP). IEEE, pp 3203–3207
92. Dosovitskiy A et al (2015) FlowNet: learning optical flow with convolutional networks. In: Proceedings of the IEEE international conference on computer vision (CVPR), pp 2758–2766
93. Kuznietsov Y et al (2017) Semi-supervised deep learning for monocular depth map prediction. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 6647–6655
94. Zhang F et al (2020) Domain-invariant stereo matching networks. In: European conference on computer vision (ECCV) 2020. Springer, pp 420–439
95. Yang G et al (2019) Hierarchical deep stereo matching on high-resolution images. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 5515–5524
96. Zhang J et al (2022) Revisiting domain generalized stereo matching networks from a feature consistency perspective. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 13 001–13 011
97. Watson J et al (2021) The temporal opportunist: self-supervised multi-frame monocular depth. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 1164–1174
98. Shu C et al (2020) Feature-metric loss for self-supervised learning of depth and egomotion. In: European conference on computer vision (ECCV). Springer, pp 572–588
99. Godard C et al (2019) Digging into self-supervised monocular depth estimation. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 3828–3838
100. Kim S et al (2020) Adversarial confidence estimation networks for robust stereo matching. IEEE Trans Intell Transp Syst 22(11):6875–6889
101. Wu C-Y et al (2022) Toward practical monocular indoor depth estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 3814–3824
102. Kanopoulos N et al (1988) Design of an image edge detection filter using the sobel operator. IEEE J Solid-State Circuits 23(2):358–367
103. Liu H et al (2021) Pseudo supervised monocular depth estimation with teacher-student network. arXiv:2110.11545
104. Hartley R et al (2003) Multiple view geometry in computer vision. Cambridge University Press

Chapter 4
Semantic Segmentation for Autonomous Driving

Jingwei Yang, Sicen Guo, Mohammud Junaid Bocus, Qijun Chen, and Rui Fan

Abstract The task of semantic segmentation involves labeling each pixel in an image with its corresponding object class, which is achieved by clustering regions belonging to the same category using artificial intelligence. This is an important step from image processing to image analysis and has numerous applications in areas such as automatic driving, image enhancement, and 3D map reconstruction. With the emergence of deep learning, several sophisticated and efficient algorithms have been developed for this task. This chapter aims to review these methods, starting with a discussion of state-of-the-art semantic segmentation methods for both single modality and data fusion, emphasizing their contributions and significance in the field. Additionally, an overview of commonly used datasets is provided to assist researchers in selecting the appropriate dataset for their needs and goals. A comprehensive summary of evaluation metrics used to assess semantic segmentation results, along with corresponding benchmarks for a number of classic datasets, is also presented. Finally, practical applications of semantic segmentation in autonomous driving are explored, and conclusions are drawn on the current state of the art.

J. Yang · S. Guo · Q. Chen · R. Fan (✉)
The Robotics and Artificial Intelligence Laboratory (RAIL), State Key Laboratory of Intelligent Autonomous Systems, The Department of Control Science and Engineering and Frontiers Science Center for Intelligent Autonomous Systems, Tongji University, Shanghai 201804, People's Republic of China
e-mail: rfan@tongji.edu.cn

J. Yang
e-mail: jw_yang@tongji.edu.cn

S. Guo
e-mail: guosicen@tongji.edu.cn

Q. Chen
e-mail: qjchen@tongji.edu.cn

M. J. Bocus
University of Bristol, Bristol, United Kingdom
e-mail: junaid.bocus@bristol.ac.uk

R. Fan et al. (eds.), *Autonomous Driving Perception*, Advances in Computer Vision and Pattern Recognition, https://doi.org/10.1007/978-981-99-4287-9_4

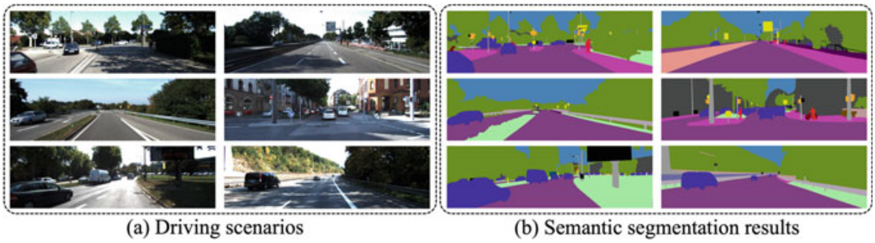
4.1 Introduction

Semantic segmentation is a critical task in computer vision that aims to partition an image into visually significant regions for further visual comprehension and analysis [1]. This high-level task is essential for complete scene understanding, as it is used in 2D images, videos, and even 3D or volumetric data. The importance of scene comprehension in computer vision is highlighted by the growing number of applications that rely on extracting knowledge from images, such as autonomous driving [2, 3], precision agriculture [4], geological testing [5], and medical treatment [6].

Since the first successful demonstration in the 1980s [7], significant progress has been made in the field of autonomous driving. This technology has immense potential benefits, including reducing traffic congestion and improving driving safety. Nevertheless, it is an arduous task to guarantee the reliability of autonomous driving. An autonomous car must be able to accurately identify, assess, and make decisions about road conditions and plan routes. Even a minor error in any of these steps can have serious consequences. To address this challenge, perception systems in autonomous driving must be accurate, robust, and capable of real-time processing. Accuracy guarantees reliable environmental information, robustness ensures proper functioning even in adverse weather conditions or sensor degradation, while real-time processing is crucial for high-speed driving.

Semantic segmentation is a critical technology for autonomous driving, as it enables the understanding of driving scenes (as shown in Fig. 4.1). Historically, researchers have used a range of traditional computer vision and machine learning techniques to address this issue. Semantic segmentation is essential for numerous applications, including scene interpretation and robot perception [1, 8, 9]. Initially, researchers employed random forest and conditional random field (CRF) methods to classify various semantic elements. However, the current mainstream techniques for semantic segmentation have shifted towards deep learning, resulting in significant improvements in segmentation performance [10–12]. In this section, we aim to provide a comprehensive overview of both single-modal and data fusion-based semantic segmentation.

Significant progress has been made in the field of single-modal image segmentation using deep learning-based semantic segmentation models, with exceptional


(a) Driving scenarios

(b) Semantic segmentation results

Fig. 4.1 Semantic segmentation used for autonomous driving

accuracy rates on widely used benchmarks. Early fusion involves combining conventional unimodal semantic segmentation networks and reusing existing models. For example, Gouprie et al. [13] adapted the RGB network in [14] to RGB-D semantic segmentation by concatenating input RGB and depth channels. Late fusion, on the other hand, aims to fuse multi-level features with different types through different streams [15–17]. For instance, [16] used two CNN models to extract features from RGB and depth images, and SVM classifiers to distinguish different semantic categories. [15] introduced a gated fusion layer that learns the respective contributions of high-level modality-specific features to enable more effective combination. Furthermore, hierarchical fusion allows the integration of multimodal features at different levels, as demonstrated in the works of [18, 19].

This chapter is structured as follows: Sect. 4.2 provides an overview of the latest segmentation algorithms and their characteristics, covering both single-modal and data fusion-based semantic segmentation. Section 4.3 provides a summary of the datasets and corresponding benchmarks utilized for semantic segmentation. In Sect. 4.4, evaluation metrics are discussed. In Sect. 4.5, various applications of semantic segmentation are explored. The chapter concludes with a discussion of current challenges in Sect. 4.6 and a summary in Sect. 4.7.

4.2 State of The Art

4.2.1 Single-Modal Networks

Pixel-level single-channel semantic segmentation is more challenging than image-level image classification and region-level object detection. Semantic segmentation is particularly crucial for a large number of real-world applications, such as autonomous driving [20, 21], surrounding sensing [22–24], and medical diagnosis [25, 26].

4.2.1.1 Single-Modal Convolutional Neural Networks

Fully Convolutional Networks

The introduction of the Fully Convolutional Network (FCN) [27] marked a significant advancement in end-to-end semantic segmentation. It has become the most advanced and robust framework for scenario analysis. In traditional Convolutional Neural Networks (CNNs), several fully connected layers are added after the final convolutional layer, which converts the feature map into a fixed-length feature vector. However, FCN can process input images of any size and upsample the final layer's feature map using a deconvolution layer. As a result, the output prediction for each pixel preserves the spatial information of the original input image, recovering its original size. This outperforms initial CNN methods such as R-CNN [28] and SDS [29]. However, the classification of individual pixels does not fully consider the

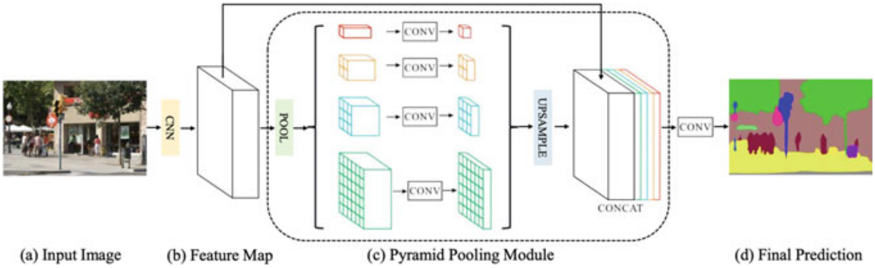


Fig. 4.2 PSPNet [32] architecture. Given an input image (**a**), it first uses CNN to get the feature map of the last convolutional layer (**b**), then a pyramid parsing module is applied to harvest different sub-region representations, followed by upsampling and concatenation layers to form the final feature representation, which carries both local and global context information in (**c**). Finally, the representation is fed into a convolution layer to get the final per-pixel prediction (**d**)

relationship between pixels, resulting in a lack of spatial consistency. This approach also increases the memory footprint and computational complexity. To address this, FastFCN [30] formulates the task of extracting high-resolution feature maps into a joint upsampling problem, reducing computational complexity by more than three times without sacrificing performance.

Pyramid Models

Recent semantic segmentation approaches have made significant improvements by incorporating pyramid-based multi-resolution representations to increase the receptive fields. However, traditional fully convolutional network (FCN) [31] models often suffer from misclassification of similar categories due to inadequate utilization of global information and contextual relationships across different receptive fields. The spatial pyramid pooling (SPP) [31] model addresses this limitation by obtaining global image-level features to improve the ability of FCN-based models to utilize contextual information. Nevertheless, to reduce the loss of contextual information within different sub-regions, [32] proposes a hierarchical global prior, known as the pyramid pooling module (PPM), as illustrated in Fig. 4.2. The PPM module combines the features of four different pyramid scales to capture contextual information of varying scales and sub-regions. Finally, the resulting representation is passed through a convolutional layer to produce the final per-pixel prediction.

The DeepLab model also addresses this challenge by using Atrous convolutions and Atrous Spatial Pyramid Pooling (ASPP) modules. Compared to traditional convolution, dilated convolution can obtain a bigger receptive field without sacrificing spatial resolution. This architecture has evolved over several generations:

DeepLabV1: Uses Atrous Convolution and Fully Connected CRF to control the resolution at which image features are computed.

DeepLabV2: Uses ASPP to consider objects at different scales and segment with much improved accuracy.

DeepLabV3: Apart from using Atrous Convolution, DeepLabV3 uses an improved

ASPP module with batch normalization and image-level features. Unlike its predecessors, DeepLabV3 does not incorporate the CRF (Conditional Random Field) method. The final feature map obtained from the network captures abundant semantic information, however, due to the pooling and convolution operations with striding used in the network backbone, it lacks the finer details of object boundaries.

DeepLabv3+ [33]: It improves upon DeepLabv3 by adding a decoder module to the architecture. While the encoder module encodes multi-scale contextual information using atrous convolution at multiple scales, the decoder module refines the segmentation results along object boundaries in a simple yet effective manner.

Encoder-Decoder Architectures

UNet [34] aims to enhance a standard convolutional network by adding a series of layers that gradually upsample the previous layers. This allows for the combination of high resolution features from the encoder with upsampled feature maps to fuse different levels of information and learn context information. By using a u-shaped architecture, the decoder can propagate context information to higher resolution layers and allow for better segmentation results.

The network structure LinkNet [35] is based on UNet's encoder-decoder structure. There is also a jump connection between the Encoder and Decoder, which allows the network to forget certain information during encoding and revisit it during decoding. Because the amount of information required by the network to decode and generate the image is relatively low, this reduces the number of parameters required by the network. Various operations can be used to implement the jump connection. Another advantage of using jump connections is that you can easily implement reverse gradient flow with the same connection. Tiramisu [36], another semantic segmentation algorithm, connects the two and sends the hidden encoder output to the corresponding decoder input.

The novelty of SegNet [37] lies in the way the decoder upsamples its lower-resolution input feature maps. Specifically, the pooling index computed in the maximum pooling step of the corresponding encoder is used by the decoder to perform nonlinear upsampling. Inverse pooling upsamples the feature maps, maintaining the integrity of high-frequency information while ignoring neighboring information for feature maps with lower resolution.

Unlike SegNet's symmetric encoder-encoder architecture, the ENet [38] architecture consists of a large encoder and a small decoder. ENet believes that the encoder should be able to manipulate lower resolution data, allowing it to process information and filter ideas. In contrast, the decoder is primarily responsible for sampling the output of the encoder and only needs to fine-tune the details.

High-Resolution Network (HRNet) [39] proposes a different approach to previous methods where high-resolution feature maps are recovered from low-resolution representations. Instead, HRNet suggests keeping high-resolution representations during the entire feature extraction and fusion process, avoiding loss of essential shape and bounding details that may occur with a high-to-low encoder. HRNet achieves high performance by using multi-resolution sub-networks in parallel and progressively and repeatedly fusing multi-scale information.

Prior approaches such as pooling-based [40, 41] and dilated convolution-based [42, 43] extensions, make non-adaptive use of homogeneous contextual dependencies for all image regions, which overlooks the variations in local representation and contextual dependencies for different categories.

Attention

Attention mechanism has become a widely used technique in various deep learning tasks, including natural language processing and image recognition. It is a crucial technique that deserves a thorough understanding in the field of deep learning. The inspiration for studying the Attention Mechanism comes from the human brain's ability to selectively focus on a region of interest from a visual signal. Humans learn to focus their attention on specific regions of an image, based on previously observed information, rather than processing all pixels of the entire image at once and adjusting the focus over time. This focus of attention enables humans to quickly select relevant information and disregard irrelevant information, similar to how the Attention Mechanism works in deep learning.

CNNs are limited in their ability to understand complex scenes due to the physical design of their convolutional kernel, which constrains feature information flow to a fixed-size local region. In order to address this limitation, Point-wise Spatial Attention Network (PSANet) [44] (Fig. 4.3) introduces adaptive attention masks to allow each location on the feature map to associate with other locations, thereby moderating the local neighborhood constraint. PSANet also employs bidirectional information propagation paths, which allow each location to aggregate information from all other locations to aid in its own prediction, while simultaneously allowing information from each location to be distributed globally to assist in the prediction of all other locations.

With the frequent use of attention mechanism in neural networks, non-local neural networks [45] obtain the attention mask by calculating the correlation matrix between each spatial point in the feature map. Figure 4.4a depicts the Non-Local structure. The block first computes the similarity between all positions. When the size of the input feature map is $C \times H \times W$, the computational complexity is $O(C \times H^2 \times W^2)$. Then matrix multiplication is performed to collect the influence of all locations on themselves, and the computational complexity of this step is also $O(C \times H^2 \times W^2)$. The high complexity brought by this matrix multiplication results in prohibitive com-

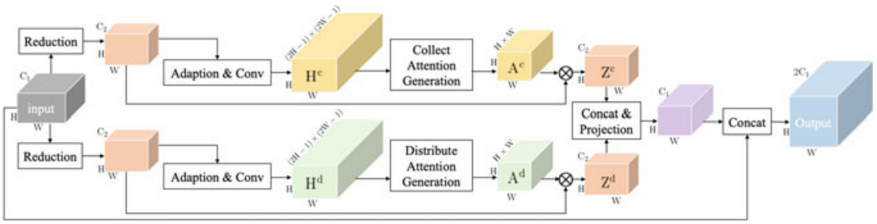


Fig. 4.3 Architecture of PSANet [44], consisting of two parallel branches

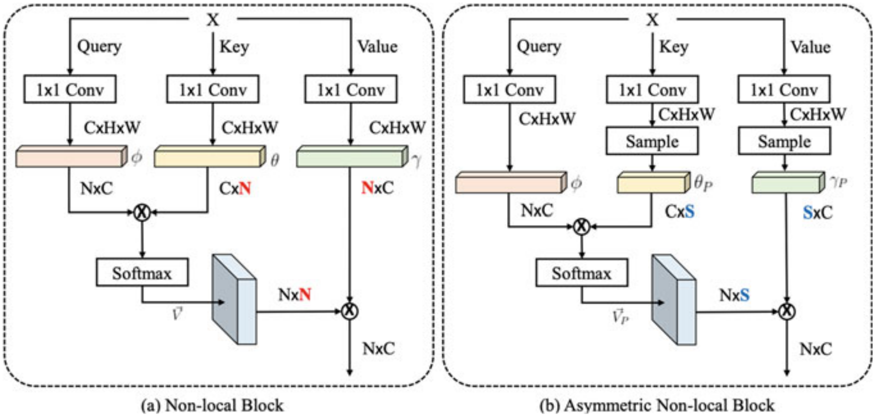


Fig. 4.4 **a** Architecture of a standard non-local block [45]; **b** the asymmetric non-local block [46]

putational costs and a massive GPU memory occupation. To address this issue, Asymmetric Non-local Neural Network (ANN) [46] selectively samples a small number of representative points from feature maps. Figure 4.4b shows the architecture of the ANN block, and it can be seen that the output size of the non-local blocks remains the same as long as the output of the key and value branches remains the same size. Therefore, the ANN block efficiently reduces the computational complexity by selecting a small number of representative points from the key and value branches, while maintaining comparable performance.

Non-local Net [45] computes attention using a whitened pairwise term and a unary term, which are tightly coupled, making it difficult to learn them separately. To address this limitation, Disentangled non-local block (DNLNet) [47] decouples these terms to facilitate their independent learning. Another method, Dual Attention Network (DANet) [48], encodes global context using a self-attention mechanism that incorporates both spatial and channel attention. Current methods such as PSANet [44] and DANet [48] use adaptive weights to calculate pair-wise similarity or learn pixel-wise attention maps. However, these approaches overlook the importance of global guidance from the global information extractor. In contrast to these works, Adaptive Pyramid Context Network (APCNet) [49] takes global-guided local affinity into account to estimate the degree of subregion contribution from local and global representations and exploits multi-scale representation with a feature pyramid.

Despite the superior performance of attention mechanisms over other methods, such as ASPP, PPM, large convolutional kernels, and stacked convolutional layers, the additional computational and GPU memory usage is often unaffordable. Therefore, some networks focus on reducing computational effort. Criss-Cross Network (CCNet) [50] aims to reduce the large computation budget introduced by full spatial attention. Another method, Interlaced Sparse Self-Attention Network (ISANet) [51], factors the dense affinity matrix as the product of two sparse affinity matrices, significantly reducing computation and memory complexity, particularly when processing high-resolution feature maps.

The attention-based methods have high computational complexity and require a large amount of GPU memory due to the generation of a large attention map. The generation and use of the attention map are computed with respect to all positions, which is a bottleneck. To address these issues, Expectation Maximization Attention (EMA) [52] proposes a novel attention-based method that uses the expectation maximization (EM) algorithm [53] to find a more compact basis set, significantly reducing computational complexity. Unlike previous methods that treat all pixels as reconstruction bases [44, 45], EMA finds a more efficient solution. EncNet (Context Encoding Module) [54] adds a class-dependent feature map to provide prior information about the scene, which "simplifies" the problem for the network. This approach differs from the standard attention mechanism.

Transformer Models

Transformers [56] rely on self-attention mechanisms and capture long-range dependencies among tokens in a sentence. In addition, transformers are highly parallelizable and suitable for training on large datasets, which has led to their success in natural language processing (NLP). This success has inspired the development of several computer vision methods that combine CNNs with self-attention mechanisms to tackle semantic segmentation problems [55].

The transformer-based OCR [57] approach uses the representation of the corresponding object class to characterize a pixel, which strengthens the pixel representation. Swin transformer [58] uses a variant of ViT [59], and proposes a hierarchical transformer whose representation is computed with shifted window. This hierarchical architecture has the flexibility to model at various scales and has different linear computational complexity with regard to image size. Segmenter [55] (Fig. 4.5) is also a transformer encoder-decoder architecture for semantic image segmentation. It relies on a ViT backbone and introduces a masked decoder inspired by

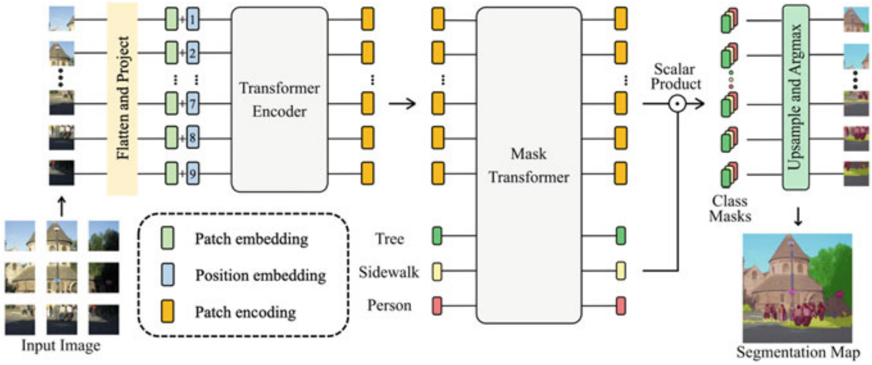


Fig. 4.5 Segmenter model [55]. The encoder module receives image patches, and the decoder module takes the input to a mask transformer to generate segmentation masks

DEtection TRansformer (DETR) [60]. Its architecture does not use convolutions but captures global context at every layer of the model during the encoding and decoding stages. SegFormer [61], a simple, efficient yet powerful semantic segmentation framework, unifies Transformers with lightweight multilayer perceptron (MLP) decoders. Twins [62] presents two powerful vision transformer backbones and dub them twin transformers: Twins-PCPVT and Twins-SVT. They find that interleaving local and global attention can produce impressive results, yet it comes with higher throughputs. ResNeSt [63] combines channel-wise attention with multi-path representation into a single unified Split-Attention block, which improves learned feature representations widely.

4.2.2 Data-Fusion Networks

Semantic segmentation is a popular computer vision task that involves partitioning an image into semantically meaningful regions. Typically, photometric data such as RGB is used for this task. However, structural information refers to the spatial relationships and arrangements of objects in the image, and it can enhance the performance and robustness of the segmentation algorithm due to its geometric property. To make the most of structural information in semantic segmentation tasks, data-fusion networks can combine different input visual information to create a richer representation of the image.

4.2.2.1 Structured Visual Information

- For humans to achieve stereo vision, perceiving depth information is crucial. In computer vision, images are acquired through cameras. However, in our everyday life, objects have three dimensions, whereas camera-captured images only contain two dimensions, resulting in the loss of one-dimensional information, which represents the relative distance of objects in space. Depth information in computer vision usually refers to the distance of each point in space relative to the camera, and the relative distance between the points can be easily calculated once the depth information is known.
- Transformed disparity information is utilized to differentiate potholes in the road. To accomplish this, a disparity map is subjected to a transformation process, whereby an energy function is minimized with respect to both the camera roll angle and the road disparity projection model [64].
- Thermal information is a visual representation that records the heat or temperature radiated by the object itself or its surroundings. An infrared (IR) camera is utilized to capture this data by using an infrared detector and an optical imaging lens to collect the infrared radiation energy distribution of the object in question. The resulting thermal image is then projected onto the photosensitive element of the IR detector, providing a detailed portrayal of the object's thermal profile.

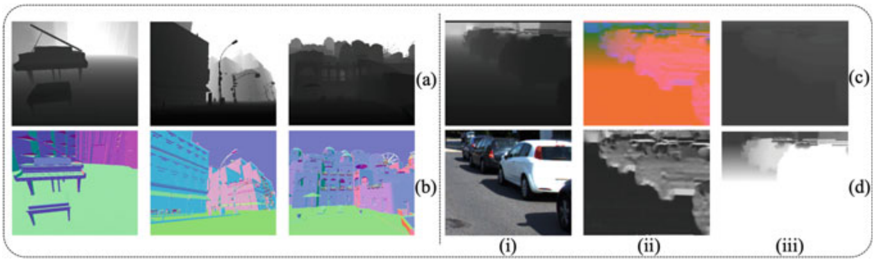


Fig. 4.6 Examples of structured visual information: **a** depth images, **b** surface normal images, (i)–(iii) in row **c** show depth image, HHA image, and height image, respectively. (i)–(iii) in row **d** show RGB image, surface normal angle image, and horizontal disparity image, respectively

- HHA information representation uses three parameters to capture depth images, namely the difference in horizontal position, the height above ground of each pixel, and the angle between the local surface normal and the estimated gravity direction. However, the HHA encoding method has limitations since it mainly emphasizes the interdependence between these parameters, while neglecting the unique contribution of each parameter to the overall representation (Fig. 4.6c, d).
- Surface normal information is a 3D spatial representation that represents the vector perpendicular to the given object. It is an important visual feature and is used in many computer vision applications [65, 66] (Fig. 4.6b).

4.2.2.2 RGB+X Data Fusion

Existing fusion methods fuse the structural visual information described in Sect. 4.2.2.1 with RGB information. Structural visual information captures geometric details such as distance, shape, and structure, which are not present in RGB images. These details provide a more precise understanding of the location and boundary of objects, aiding in semantic segmentation.

Recently, the combination of RGB images with depth information has been highly successful in the field of semantic segmentation, and FuseNet [18] is a typical example of an encoder-decoder architecture that fuses these inputs (as shown in Fig. 4.7). It uses two encoders, each with a VGG-16 [67] backbone, to extract features from the RGB and depth images separately. As the network delves deeper, the depth encoder's feature maps become incorporated into the RGB encoder. RedNet [68] takes this concept a step further by integrating multi-level features even more extensively along the top-down pathway.

RFNet [69] follows a similar approach to FuseNet by using two separate branches to extract features from RGB and depth images. The RGB branch is considered the primary branch in RFNet, while the depth branch is treated as a secondary branch. The two branches are combined using the attention feature complementary (AFC) module (as shown in Fig. 4.8). Furthermore, a spatial pyramid pooling module is implemented

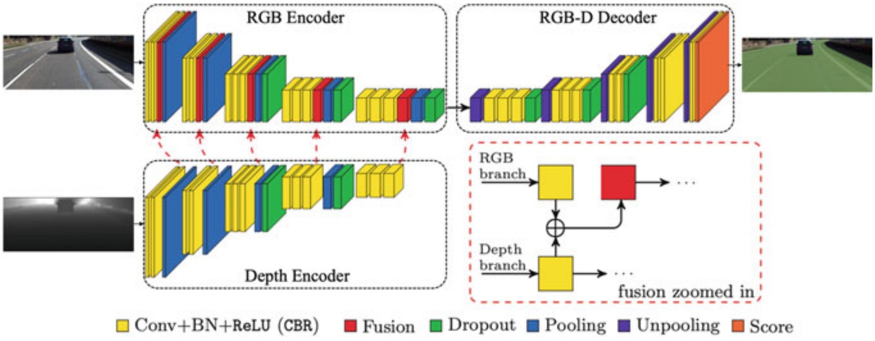


Fig. 4.7 Architecture of FuseNet [18], containing two branches to extract RGB and depth features, respectively

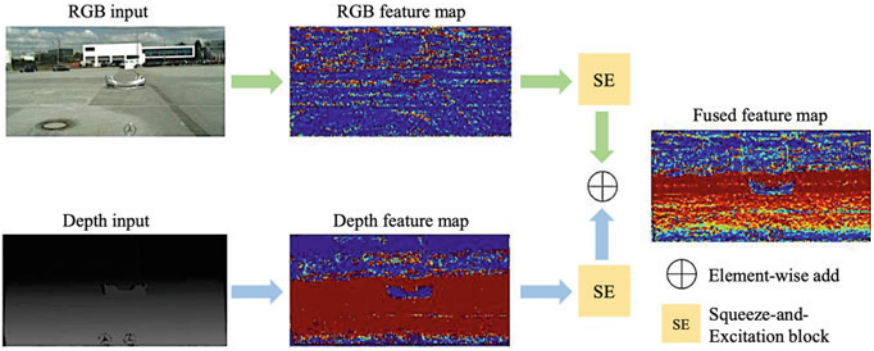


Fig. 4.8 Data fusion method in RFNet [69]

to aggregate the merged RGB-D features and produce feature maps incorporating information at multiple scales. The inclusion of a squeeze-and-excitation block in the AFC module, as proposed in [70], facilitates the acquisition of global knowledge to highlight significant channels and suppress less relevant ones. As a result, the AFC module can efficiently leverage informative features from both branches.

RFBNet [71] adds a novel stream to RGB and depth streams. These three streams are combined using approaches such as summation, concatenation, and self-supervised model adaptation (SSMA) block [72]. The resulting features are then passed through a decoder to generate the final predictions (as shown in Fig. 4.9). Specifically, RFBs are utilized in the upper layers to adeptly capture the interplay between the three streams. SSMA suggests a fusion approach that merges streams and multi-level features specific to different modalities.

Depth-aware CNN [73] utilizes depth-aware convolution and depth-aware average pooling, which can be integrated into the traditional segmentation framework to make full use of the depth information at low cost. Furthermore, in both the encoder and

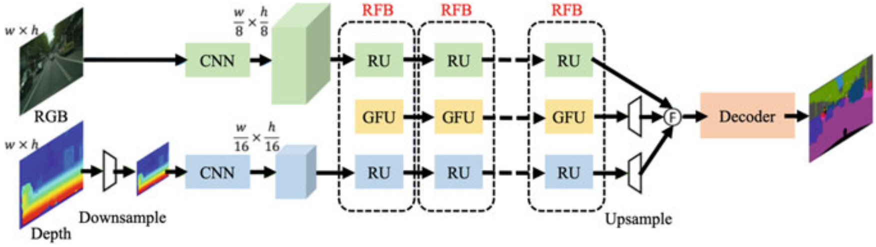


Fig. 4.9 Architecture of RFBNet [71]. Three streams: RGB stream (green), depth stream (blue), and interaction stream (yellow). F denotes the fused method

decoder, RGB and spectral image feature maps are processed separately by the depth-aware CNNs.

In addition to depth information, significant efforts have been directed towards integrating thermal and RGB images. The two types of visual information complement each other, as thermal images are proficient at representing diverse environments with varying illumination and encoding the structural details of the scene [74, 75]. The majority of previous research has split the encoder component of the semantic segmentation model into two branches of networks, extracting features from both RGB and thermal images, and then progressively merging the thermal characteristics with the RGB features based on the number of layers in the network. The following describes the current studies on fusion of RGB and thermal images.

MFNet [76] is an encoder-decoder structure data fusion network, featuring two encoders for extracting features from both RGB and thermal images (as shown in Fig. 4.10). To incorporate contextual information and extract features from each encoder, mini-inception modules utilize parallel convolutional and hole convolutional layers. In the fusion stage, the short-cut module concatenates the RGB and thermal features from each encoder at every scale before they are added to the output of the previous decoder layer, yielding the ultimate fused result. MFNet maximizes the complementarity between thermal image and RGB information for enhanced semantic segmentation in autonomous driving scenarios during the night.

RTFNet [77] is composed of three main components: an RGB encoder, a thermal encoder, and a decoder (as shown in Fig. 4.11). To extract the necessary features from the RGB and thermal images, the ResNet model is utilized as a feature extractor. The fusion process involves adding the extracted RGB and thermal features on a per-pixel basis to obtain the fused output. This fused output is then fed into the decoder to recover the feature map resolution and ultimately obtain the semantic segmentation result.

FuseSeg [78] (as shown in Fig. 4.12) also consists of two encoders and a decoder. The two encoders use DenseNet [79] to extract RGB and thermal features respectively. The RGB encoder incorporates the corresponding thermal features with the RGB features using element summation. These fused feature maps are then further

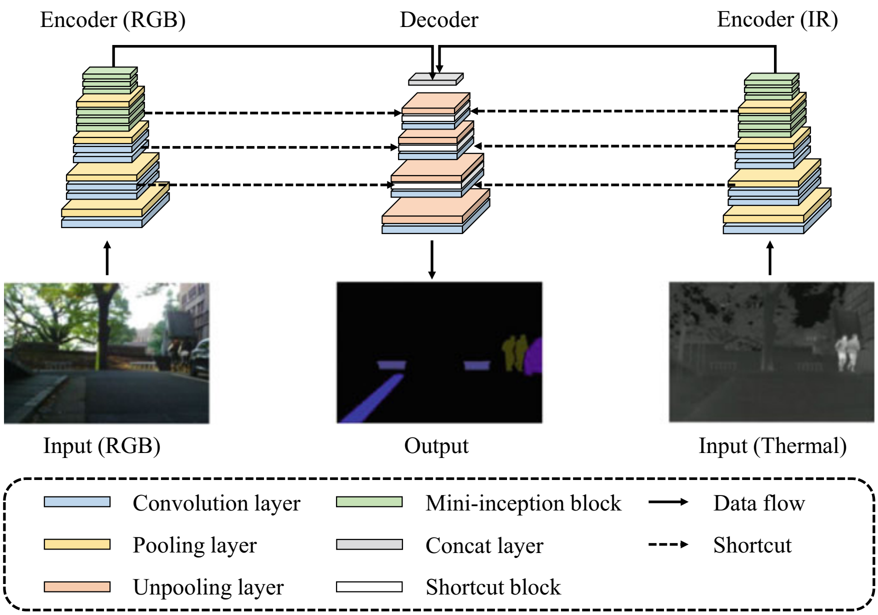


Fig. 4.10 Architecture of MFNet [76]. The output of RGB and IR encoders are fused to feed into the decoder

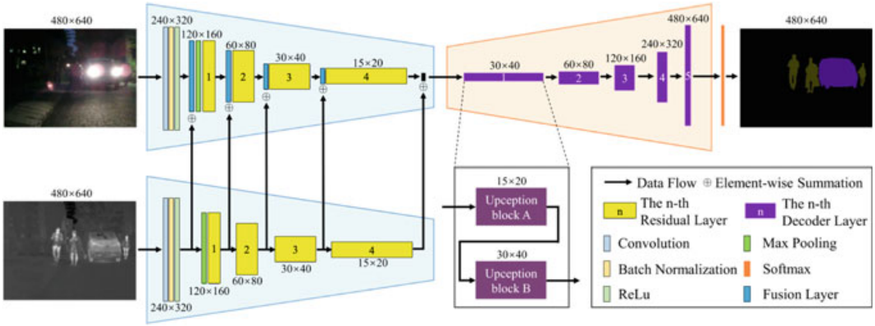


Fig. 4.11 Architecture of RTFNet [77], consisting of an RGB encoder, a thermal encoder (blue) and a decoder (orange)

combined with the decoder's corresponding feature maps using the tensor cascade method. Additionally, the bottom feature maps are directly copied to the decoder.

FEANet [80] comprises two components and aims to improve the comprehension of spatial information (as shown in Fig. 4.13). The first component of the network utilizes two encoders to extract RGB and thermal features separately. The output from each encoder layer is then fed into the feature-enhanced attention module (FEAM), which blends and refines the two features through weightage and fusion. This process enhances the features, allowing for effective capture and utilization of the comple-

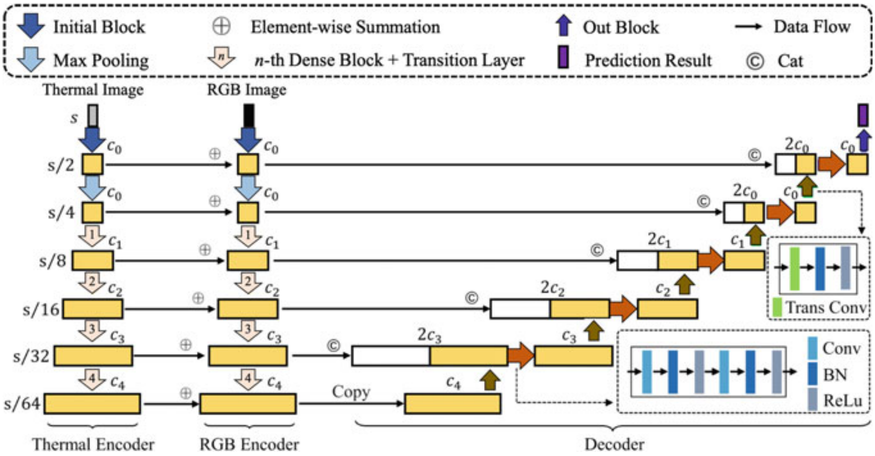


Fig. 4.12 Architecture of FuseSeg [78], consisting of an RGB encoder, a thermal encoder, and a decoder. It uses DenseNet as the backbone of the encoders

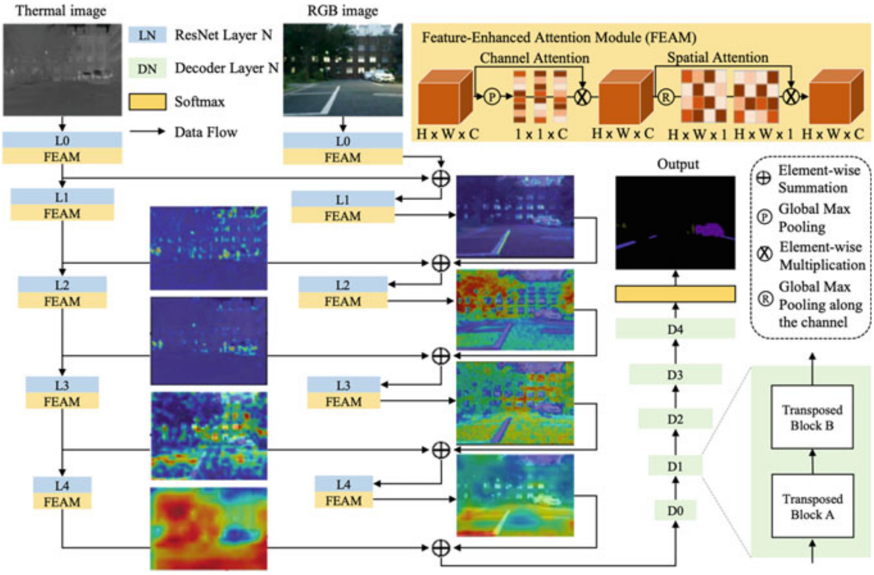


Fig. 4.13 Architecture of FEANet [80], consisting of a thermal stream, an RGB stream, and an output stream

mentary relationship between the two distinct features. The second component of the network uses a decoder to recover the resolution, which takes the enhanced fused features as input.

MFFENet [81] consists of two encoders, a feature fusion layer, and a multi-label supervision layer. It concatenates the multi-scale features with the features that contain global semantic information. To extract features, two parallel encoders based on DenseNet are situated in the top section. Feature fusion employs elementwise summation, with features at various levels being combined at the encoders' peak. The spatial attention mechanism module is used to improve fused features by focusing on foreground objects. The upgraded features are supplied to a multi-label supervision layer, which is responsible for semantic segmentation.

GMNet [82] classifies the extracted features into three groups that correspond to specific aspects of the visual content: fine-grained visual details, semantic information, and intermediate-level features. During the fusion process, GMNet integrates the low-level and high-level features separately. Specifically, for the low-level features, GMNet employs spatial and channel attention mechanisms to combine information across different feature maps. As for the high-level features, GMNet leverages multi-scale null convolution to capture fine-grained details while preserving the semantic richness of the information.

ABMDRNet [83] considers the differences between two types of visual information by mitigating their differences before fusing. Specifically, a bidirectional translation technique is employed to reduce feature differences between the images, followed by a weighting scheme to dynamically select the discriminative features for semantic segmentation.

CCAFFMNet [84] consists of two encoders and a decoder. The chief function of these encoders is to extract features from both RGB and thermal images, respectively. An innovative channel-coordinate attention feature-fusion module (CCAFFM) is deployed within the encoder to effectuate this. Specifically, the aforementioned module is responsible for extracting the RGB features and subsequently gauging the spatial correlation between each RGB and thermal feature. It then proceeds to fuse the feature maps, thereby generating fused features which are ultimately fed into the fusion layer of the RGB encoder. Finally, the fused features produced by each fusion layer are input into the decoder to enable the recovery of the resolution.

SNE-RoadSeg [86] proposed a data-fusion CNN architecture RoadSeg, which can extract and fuse features from both RGB images and the inferred surface normal information for accurate freespace detection. Based on it, SNE-RoadSeg+ [85] (Fig. 4.14) consists of SNE+ module for more accurate surface normal estimation, and a data-fusion DCNN (RoadSeg+) that can greatly minimize the trade-off between accuracy and efficiency with the use of deep supervision.

RDFNet [87] presents a fusion network that receives RGB and HHA [88] information as inputs and enhances the diverse visual information as well as the features across various levels. The HHA information is obtained via depth map coding.

These methods of fusing RGB and other visual information (Table 4.1) can combine the spatial structure features in the scene, thus boosting the performance of semantic segmentation. However, in the majority of the aforementioned approaches, the RGB and other visual features are first extracted separately before being fused, thereby limiting their ability to distinguish the differences and complementarities between different types of visual information.

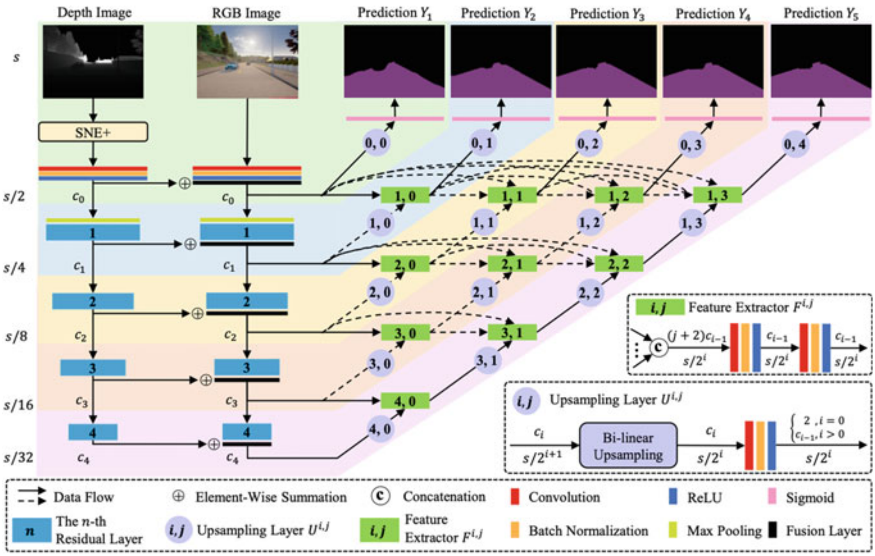


Fig. 4.14 Architecture of SNE-RoadSeg+ [85]. It consists of SNE+ (a module for achieving surface normal information), and RoadSeg+ (a data-fusion network)

Table 4.1 The results of semantic segmentation methods based on data fusion

Method	Year	Visual information	Dataset
FuseNet [18]	2017	RGB+Depth	SUNRGBD [89]
RedNet [68]	2018	RGB+Depth	SUNRGBD [89]
RFNet [69]	2020	RGB+Depth	Cityscapes [90], Lost and found [91]
RFBNet [71]	2019	RGB+Depth	ScanNet [92], Cityscapes [90]
Depth-aware CNN [73]	2018	RGB+Depth	NYU V2 [93], SUNRGBD [89], SID [94]
MFNet [76]	2017	RGB+Thermal	MFNet [76]
RTFNet [77]	2019	RGB+Thermal	MFNet [76]
FuseSeg [78]	2020	RGB+Thermal	MFNet [76]
FEANet [80]	2021	RGB+Thermal	MFNet [76]
MFFENet [81]	2021	RGB+Thermal	PST900 [95], MFNet [76]
GMNet [82]	2021	RGB+Thermal	PST900 [95], MFNet [76]
ABMDRNet [83]	2021	RGB+Thermal	MFNet [76]
CCAFFMNet [84]	2022	RGB+Thermal	RoadScene [96], MFNet [76],
SNE-RoadSeg [86]	2020	RGB+Surface normal	R2D Road [86], KITTI [97], SYNTHIA [98]
SNE-RoadSeg+ [85]	2021	RGB+Surface normal	KITTI [97], R2D road [86]
RDFNet [87]	2017	RGB+HHA	NYU V2 [93], SUNRGBD [89]

4.3 Public Datasets and Benchmarks

4.3.1 Public Datasets

In this section, we provide a summary of the datasets that have been used for training and testing in the context of semantic segmentation tasks. The datasets can be divided into two categories based on their data types: 2D datasets and 2.5/3D datasets. At the same time, for autonomous driving applications, models are often trained using synthetic data. Therefore, we also count the scenario types of the dataset in this section, including real scenarios and synthetic scenarios. Table 4.2 includes some classical datasets for semantic segmentation tasks.

4.3.1.1 2D Dataset

PASCAl VOC[1] [99] is a dataset for generic scenarios in semantic segmentation, consisting of 17125 images with a total of 20 categories (excluding background). For the semantic segmentation task, it provides 2913 images, of which 1464 were used for training with 3507 objects and 1449 were used for validation with 3422 objects. The test dataset is not publicly available.

PASCAL Context[2] [100] adds new annotated categories to PASCAL VOC 2010, increasing the number of categories to 540, with 59 commonly used categories and the others being re-labeled as background. The dataset includes 10103 images for training.

PASCAL Part[3] [101] provides additional annotations in PASCAL VOC 2010, and the components in each object are also annotated. This dataset includes the training dataset and validation dataset from PASCAL VOC, as well as 9637 labels for testing.

Stanford background dataset[4] [102] includes 715 images selected from the public datasets LabelMe [103], MSRC [104], PASCAL VOC 2007 [105], and Geometric Context [106]. The dataset is primarily targeted at outdoor scenarios with an image size of 320×240 pixels and includes at least one foreground object. The semantic and geometric labels were obtained through Amazon's Mechanical Turk (AMT) [107].

Semantic Boundaries Dataset (SBD)[5] [108] is a semantic boundary dataset that aggregates unannotated images from PASCAL VOC 2011 [109] and annotates 11355 images for semantic segmentation, 8498 of which were used for training and 2857 for testing. These annotations include the boundaries of each category and are based on category-level as well as instance-level information.

[1] http://host.robots.ox.ac.uk/pascal/VOC/voc2012/.

[2] https://cs.stanford.edu/~roozbeh/pascal-context/.

[3] http://roozbehm.info/pascal-parts/pascal-parts.html.

[4] http://dags.stanford.edu/projects/scenedataset.html.

[5] http://home.bharathh.info/pubs/codes/SBD/download.html.

Table 4.2 Classic datasets for semantic segmentation tasks

Dataset	Year	Classes	Synthetic/Real	Data	Samples	scenario
Stanford background [102]	2009	8	R	2D	715	Outdoor
SBD [108]	2011	21	R	2D	11355	Generic
PASCAL VOC [99]	2012	20	R	2D	≈3 k	Generic
PASCAL Context [100]	2014	540	R	2D	≈20 k	Generic
PASCAL Part [101]	2014	20	R	2D	≈20 k	Body part
COCO [110]	2014	≥ 80	R	2D	204721	Generic
NYU Depth V2 [93]	2012	894	R	2.5D	≈1.5 k	Indoor
SUN3D [127]	2013	–	R	2.5D	19640	Indoor
SUNRGBD [89]	2015	37	R	2.5D	10335	Indoor
ScanNet [92]	2017	21	R	2.5D/3D	1513	Generic
CamVid [111]	2008	32	R	2D	701	Driving
KITTI (road) [97]	2013	10	R	2D	579	Driving
KITTI (semantics) [114]	2018	11	R	2D	400	Driving
Virtual KITTI [115]	2016	–	S	2D	17 k	Driving
SYNTHIA [98]	2016	11	S	2D	13407	Driving
Cityscapes [90]	2016	33	R	2D	25 k	City
ADE20K [116, 117]	2017	150	R	2D	≥25 k	Generic
MVD [120]	2017	66	R	2D	25 k	City
ApolloScape [121]	2018	25	R	2D	140 k	Driving
Highway driving [122]	2019	10	R	2D	1200	Driving
WoodScape [123]	2019	40	R	2D	10 k	Driving
IDD [124]	2019	34	R	2D	10003	Driving
A2D2 [125]	2020	38	R	2D	≈41 k	Driving
IDDA [126]	2020	–	S	2D	1000 k	Driving

Microsoft Common Objects in Context (COCO)[6] [110] is a dataset built by Microsoft that includes tasks such as detection, key points, and segmentation. The dataset for the segmentation task is currently the largest semantic segmentation dataset in computer vision, containing over 80 objects, and consists of two separate releases in 2014, with 82783 training images, 40504 validation images, and 40775 test images. In the 2015 version, there are 165482 training images, 81208 validation images, and 81434 test images.

The Cambridge-driving Labeled Video Dataset (CamVid)[7] [111] is a semantic segmentation dataset for driving scenarios, which consists of urban road scenarios, including videos of five driving scenarios. There are more than 700 annotated images with a resolution of 960×720 and 32 objects. The dataset includes 368 images for training, 100 for validation, and 233 for testing.

KITTI[8] [112, 113] is a dataset for the evaluation of various computer vision tasks in autonomous driving scenarios, including real-world scenarios such as urban, rural, and highway. Semantic segmentation and instance segmentation tasks [114] consists of 200 training images with semantic segmentation annotations and 200 test images corresponding to stereo2015 and flow2015. Road semantic segmentation task [97] includes 289 images for training and 290 images for testing.

Virtual KITTI[9] [115] uses real-to-virtual world cloning to create a synthetic video dataset consisting of five videos obtained from KITTI clones. It comprises a total of 21260 high-resolution frames, and they are generated from five different virtual worlds of urban scenarios under different imaging and weather conditions. Virtual KITTI2 adds a stereo camera that makes the scenarios more realistic.

SYNTHetic Collection of Imagery and Annotations (SYNTHIA)[10] [98] is used to address the problem of semantic segmentation for scene understanding in driving scenarios and consists of a photorealistic framework of virtual cities rendered out. It includes town, city, and highway scenarios in different seasons and weather with 11 pixel-level labeling categories (road, sidewalk, void, sky, building, fence, car, sign, vegetation, pole, pedestrian, and cyclist). 13407 images are extracted from the video sequences for training.

Cityscapes[11] [90] is a dataset for pixel-level and instance-level semantic segmentation, captured from 50 city streets and includes scenarios from the spring, summer, and autumn seasons. It uses a stereo camera to acquire stereo visual video sequences, of which 5000 images are annotated with high-quality pixel-level annotations, and 20000 images are coarsely annotated. The scenarios in this dataset are more complex than previous datasets.

[6] https://cocodataset.org/#home.

[7] http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/.

[8] https://www.cvlibs.net/datasets/kitti/.

[9] https://europe.naverlabs.com/research/computer-vision.

[10] https://synthia-dataset.net/downloads/.

[11] https://www.cityscapes-dataset.com/.

ADE20K[12] [116, 117] consists of 27000 images with 3688 objects from the dataset SUN [118] and Places [119], all images are anonymized. 25574 images are labeled for training and 2000 images for validation.

The Mapillary Vistas (MVD)[13] [120] is a large-scene streetscape dataset consisting of 25000 high-resolution images with 66 objects, of which 18000 are used for training, 2000 for validation, and 5000 for testing. The images in the dataset are taken in different weather, season, and daytime conditions.

ApolloScapes[14] [121] publishes over 140000 frames of pixel-level semantically annotated images, with each image semantically annotated for each pixel. It captures hundreds of moving objects for scenarios in various traffic conditions, providing 8900 instance-level annotations for moving objects. Meanwhile, pose information is also annotated with centimeter-level accuracy. The annotation accuracy and richness of this dataset are much higher than those of KITTI and Cityscapes.

Highway Driving[15] [122] is a densely annotated dataset with ten objects for the semantic video segmentation task, which takes into account the relationship between neighbouring frames for each frame annotation. It consists of 20 sequences of 60 frames at a frame rate of 30Hz, 15 of which are for training and five for testing.

WoodScape[16] [123] is the first publicly available fisheye dataset for autonomous driving, with image acquisition by the four fisheye cameras on the vehicle. The dataset includes a total of 10000 images with semantic annotations for 40 categories and over 100000 annotated images for other tasks.

IDD[17] [124] is a new dataset for understanding road scenes in unstructured environments. The images in the dataset are collected by front-facing cameras in cars, it includes 10003 images labeled with 34 categories and 182 driving sequences on Indian roads. The images are mostly in 1080p resolution, with some images in 720p and other resolutions. The categories annotated add unique categories such as animals and autorickshaws compared to other datasets. Conditions such as weather and lighting are highly variable in this dataset.

Audi Autonomous Driving (A2D2)[18] [125] is a large autonomous driving dataset released by Audi, which contains RGB images and corresponding 3D point cloud data. The dataset uses six cameras and five LiDAR sensors to capture the surrounding environment of vehicles, collecting scenarios from highways, rural roads, and cities in southern Germany under different weather conditions. The dataset is annotated with 41227 frames of non-sequential data, all of which include semantic segmentation annotations and point cloud annotations, including 12497 frames of 3D bounding box annotations for objects in the field of view of the front camera. In addition, 392556 consecutive frames of unannotated sensor data are included.

[12] https://groups.csail.mit.edu/vision/datasets/ADE20K/index.html.

[13] https://research.mapillary.com/publication/iccv17a/.

[14] http://apolloscape.auto.

[15] https://sites.google.com/site/highwaydrivingdataset/.

[16] https://drive.google.com/drive/folders/1X5JOMEfVlaXfdNy24P8VA-jMs0yzf_HR.

[17] http://idd.insaan.iiit.ac.in/accounts/login/?next=/dataset/download/.

[18] https://www.a2d2.audi/a2d2/en.html.

ItalDesign Dataset (IDDA)[19] [126] is a large-scale synthetic dataset for semantic segmentation, consisting of over 100 different source visual domains. The dataset is used to solve the problem of domain transfer between training, and test data in various weather and viewpoint conditions in seven different types of cities.

4.3.1.2 2.5D/3D Dataset

NYU Depth V2[20] [93] consists of video sequences of indoor scenarios captured by RGB and depth cameras and includes 464 scenarios from three cities. In total, there are 1449 annotated RGB images and depth images containing 26 scenario categories. In addition, 407024 images are not annotated.

SUN3D[21] [127] is similar to NYU Depth V2 in that it consists of large-size RGB-D video data with eight annotated sequences, with scenarios in the dataset captured at different times of the day. Each frame in the sequence is accompanied by a semantic annotation and camera pose information. The dataset consists of 415 video sequences, captured in 41 different buildings in 254 different scenarios.

SUNRGBD[22] [89] is a collection of 10000 images from four RGB-D sensors, including images from NYU depth v2, Berkeley B3DO [128], and SUN3D. The dataset is densely annotated, including 146617 2D polygons and 58657 3D bounding boxes with accurate object orientations, as well as 3D room layouts and categories of the scenario.

ScanNet[23] [92] is an RGB-D video dataset containing 2.5 million views in more than 1.5 k scans. It annotates with 3D camera poses, surface reconstructions, and instance-level semantic segmentations. It consists of a total of 1513 images with 21 categories of objects, of which 1201 scenes are used for training and 312 scenes for testing.

4.3.2 Online Benchmarks

In this section, we list the benchmark results corresponding to two classical datasets for the semantic segmentation task. The evaluation metrics for semantic segmentation are based on the generic IoU (as shown in Sect. 4.4) and give four related metrics: IoU class, iIoU class, IoU category, and iIoU category. IoU class considers the accuracy of the segmentation results for the global scenario and evaluates the segmentation accuracy for the whole scenario. However, the global IoU measure is biased towards object instances covering large areas of the image and is not appli-

[19] https://idda-dataset.github.io/home/.

[20] https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html.

[21] http://sun3d.cs.princeton.edu/.

[22] http://rgbd.cs.princeton.edu/challenge.html.

[23] https://github.com/ScanNet/ScanNet.

cable to scenarios with strong scale variation. Therefore, iIoU class builds on the IoU class by segmenting the image scenario at the instance level and predicting the result. This metric focuses more on the segmentation accuracy of the algorithm across instances, with each instance in the scenario being segmented. Different instances in the same category are evaluated separately to obtain the final result. IoU is divided into ‘Class’ and ‘Category’ according to the granularity of the segmentation, with the subdivision class being divided into the coarse category for coarse granularity. The IoU category and iIoU category are evaluated according to the granularity of the segmentation.

4.3.2.1 Cityscapes

Table 4.3 shows semantic segmentation results in Cityscapes benchmark. This task involves predicting a per-pixel semantic labeling of the image without considering higher-level object instance or boundary information. It introduces a loss term that is aware of the boundaries for semantic segmentation, using an inverse-transformation network that effectively learns the extent of parametric transformations between predicted and target boundaries [90]. It achieves the highest IoU class, iIoU class, and iIoU category, while HMS Attention [129] method achieves the highest IoU category.

Table 4.3 Semantic segmentation results in Cityscapes benchmark, where the best results are in bold type

Method	IoU class (%)	iIoU class (%)	IoU category (%)	iIoU category (%)
InverseFormNet [130]	**85.8**	**72**	93.1	**85.6**
HMS Attention [129]	85.4	70.4	**93.2**	85.4
Naive-Student [131]	85.2	68.8	92.9	82.0
ViT-Adapter [132]	85.2	68.3	92.8	83.4
Wide-ResNets [133]	85.1	71.2	93.0	85.1
OCR-Seg [134]	84.5	65.9	92.7	83.9
Panoptic-DeepLab [135]	84.5	68.7	92.9	82.8
DCNAS [136]	84.3	68.5	92.7	84.6
EfficientPS [137]	84.2	65.2	92.5	83.5
Axial-DeepLab [138]	84.1	66.0	92.6	79.7

Table 4.4 Semantic segmentation results in KITTI benchmark, where the best results are in bold type

Method	IoU class (%)	iIoU class (%)	IoU category (%)	iIoU category (%)
WRP [139]	**76.44**	**50.92**	**89.63**	73.69
UJS_model [140]	75.11	47.71	89.53	**75.75**
VideoProp [141]	72.82	48.68	88.99	75.26
SN_DN161 [142]	68.89	40.45	87.06	67.93
MSeg1080_RVC [143]	62.64	31.62	86.59	68.05
Chroma UDA [144]	60.36	31.70	80.73	61.91
IfN-DomAdap [145]	59.50	30.28	81.57	61.91
SegStereo [146]	59.10	28.00	81.31	60.26
SGDepth [147]	53.04	24.36	78.65	55.95
SDNet [148]	51.14	17.74	79.62	50.45
APMoE_ROB [149]	47.96	17.86	78.11	49.17

4.3.2.2 KITTI

Table 4.4 shows the KITTI semantic segmentation benchmark. It consists of 200 semantically annotated trains as well as 200 test images corresponding to the KITTI Stereo and Flow Benchmark 2015. WRP [139] method learns to refine geometrically-warped labels and infuse them with learned semantic priors in a semi-supervised setting by leveraging cycle-consistency across time. It achieves the highest IoU class, iIoU class, and IoU category, while UJS_model [140] method achieves the highest iIoU category.

4.4 Evaluation Metrics

To ensure the usefulness and significance of semantic segmentation in this field, strict performance evaluation is necessary. Moreover, it is imperative that the evaluation of a system is carried out using established and widely recognized metrics that ensure equitable comparisons with existing methods. Moreover, various aspects of the system must be scrutinized to establish its soundness and usefulness, including execution time, memory usage, and accuracy. Depending on the system's purpose or context, certain metrics may carry greater significance than others; for instance, in a real-time application, execution speed may take precedence over accuracy to a certain extent. However, in the interest of scientific rigor, it is crucial to provide a comprehensive set of metrics for any proposed method.

Many evaluation criteria have been proposed and are frequently used to assess the accuracy of any kind of technique for semantic segmentation. The essence of semantic segmentation is the classification of pixels, the evaluation metrics of classification methods include confusion matrix, accuracy, precision, recall, and F1-score. The confusion matrix is composed of n_{TP} (true positive), n_{FP} (false positive), n_{FN} (false negative), and n_{TN} (true negative). The evaluation of semantic segmentation performance involves computing metrics such as pixel accuracy and Intersection over Union (IoU). We remark on the following notation details.

We denote the total number of all classes as m, where i represents the ith class.

- Pixel Accuracy (PA). PA is used to evaluate the percentage of correctly classified pixels. It is a simple metric that computes the ratio between the number of properly classified pixels and the total number of pixels. The formula for calculating PA is:

$$\mathrm{PA} = \frac{n_{\mathrm{TP}} + n_{\mathrm{TN}}}{n_{\mathrm{TP}} + n_{\mathrm{FP}} + n_{\mathrm{FN}} + n_{\mathrm{TN}}}. \tag{4.1}$$

- Class Pixel Accuracy (CPA). CPA represents the percentage of pixels that are actually classified as category i among all pixels predicted as category i:

$$\mathrm{CPA} = \frac{n_{\mathrm{TP}}}{n_{\mathrm{TP}} + n_{\mathrm{FP}}}. \tag{4.2}$$

- Mean Pixel Accuracy (MPA). MPA is calculated by dividing the sum of the correctly predicted pixels for each class by the sum of the total predicted pixels for that class. It is an improvement over PA and takes into account the accuracy for each individual class rather than just the overall accuracy. The formula for MPA is:

$$\mathrm{MPA} = \frac{\sum_{i=1}^{m} \mathrm{CPA}_i}{m}. \tag{4.3}$$

- Intersection over Union (IoU). IoU is a standard metric used to measure the degree of coincidence between pixel predictions and the ground truth, and to determine whether the pixel prediction is a positive or negative sample by comparing it with a threshold. It calculates the ratio between the intersection and the union of two sets, namely the ground truth and the predicted segmentation. This ratio can be expressed as the number of true positives (intersection) over the sum of true positives, false negatives, and false positives (union). IoU is calculated on a per-class basis:

$$\mathrm{IoU} = \frac{n_{\mathrm{TP}}}{n_{\mathrm{TP}} + n_{\mathrm{FP}} + n_{\mathrm{FN}}} \tag{4.4}$$

- Mean intersection over Union (MIoU). MIoU is a commonly used evaluation metric in semantic segmentation. It measures the average intersection over union (IoU) for all classes in a dataset. It is calculated by first computing IoU for each class and then taking the mean of all class IoUs. The formula for MIoU is as follows:

$$\text{MIoU} = \frac{\sum_{i=1}^{m} \text{IoU}_i}{m}. \tag{4.5}$$

- Frequency Weighted Intersection over Union (FWIoU). It is an improved version of MIoU, which takes into account the class imbalance issue that often occurs in semantic segmentation datasets. It assigns different weights to each class based on its appearance frequency, so that the evaluation metric is not dominated by the performance of the majority classes. FWIoU is calculated as follows:

$$\text{FWIoU} = \frac{n_{\text{TP}} + n_{\text{FN}}}{n_{\text{TP}} + n_{\text{FP}} + n_{\text{TN}} + n_{\text{FN}}} \times \frac{n_{\text{TP}}}{n_{\text{TP}} + n_{\text{FP}} + n_{\text{FN}}}. \tag{4.6}$$

- F1 score (also known as the Dice score). It is a metric used to evaluate the accuracy of binary classification models. It is calculated by balancing the precision and recall of the test sample, and is given by the following formula:

$$F_\beta = \frac{n_{\text{TP}}^2(1+\beta^2)}{(1+\beta^2)n_{\text{TP}}^2 + n_{\text{TP}}(\beta^2 n_{\text{FN}} + n_{\text{FP}})} \tag{4.7}$$

Generally, recall is considered as important as precision, so F1 score sets β to 1:

$$F_1 = \frac{2n_{\text{TP}}}{2n_{\text{TP}} + n_{\text{FP}} + n_{\text{FN}}}. \tag{4.8}$$

4.5 Specific Autonomous Driving Tasks

Semantic segmentation is utilized as a valuable tool for accomplishing specific tasks related to autonomous driving.

4.5.1 Freespace Detection

Freespace detection plays a crucial role in the visual perception of autonomous driving [150, 151]. It is essential to determine the drivable area for a vehicle to move safely. Freespace detection provides vital information for subsequent path planning, mainly to achieve road path planning and obstacle avoidance. However, environmental factors such as light conditions and extreme weather can affect the accuracy of freespace detection results obtained from cameras, LiDAR, and other sensors. Additionally, in congested road sections, surrounding vehicles may obstruct the view and affect the accuracy of freespace detection. Freespace detection is essentially a semantic segmentation problem that can be addressed through data fusion of different visual inputs. Figure 4.15 [86] shows the results of freespace detection.

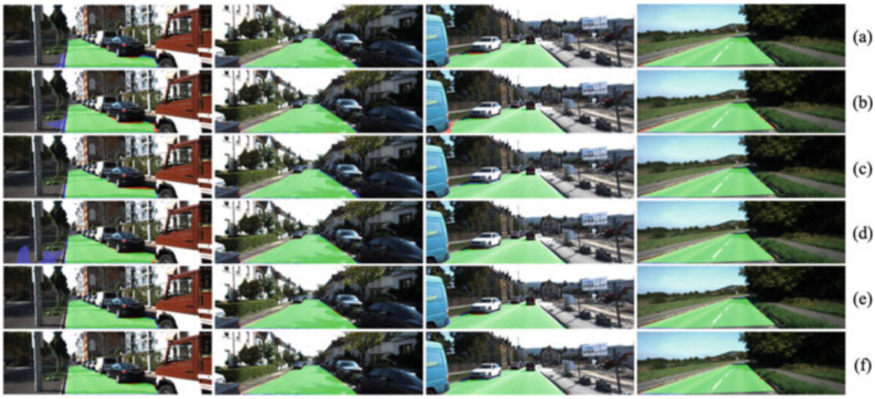


Fig. 4.15 Examples on the KITTI road benchmark, where rows **a**–**f** show the freespace detection results obtained by RBNet [152], TVFNet [153], LC-CRF [154], LidCamNet [155], RBANet [156] and SNE-RoadSeg, respectively. The true positive, false negative and false positive pixels are shown in green, red and blue, respectively

4.5.2 Road Defect Detection

Road pothole detection systems have become critical not only for smart urban road maintenance but also for autonomous driving [64, 157–160], with the rapid development of computers. While today's autonomous vehicles primarily focus on large target information such as pedestrians, traffic signs, and other vehicles, road defects also play a crucial role in ensuring ride quality, vehicle handling, fuel consumption, and tire wear. Sensing information about the size and shape of potholes can help self-driving cars drive smoothly, thereby improving driving comfort while protecting the vehicle [161, 162]. In recent years, semantic segmentation has been widely used for road defect detection [163]. Figure 4.16 [164] shows the results of semantic segmentation applications.

4.5.3 Road Anomaly Detection

The ability to detect road anomalies is crucial for ensuring the safety of autonomous vehicles [170]. Road anomalies refer to areas where there is a height difference from the surrounding freespace area, and detecting them allows the vehicle to make timely adjustments and avoid potential risks. Recent advances in deep learning technology have led to the development of several semantic segmentation methods that have proven effective in detecting road anomalies. Figure 4.17 [171] shows the results of semantic segmentation based on data fusion applied in road anomaly detection.

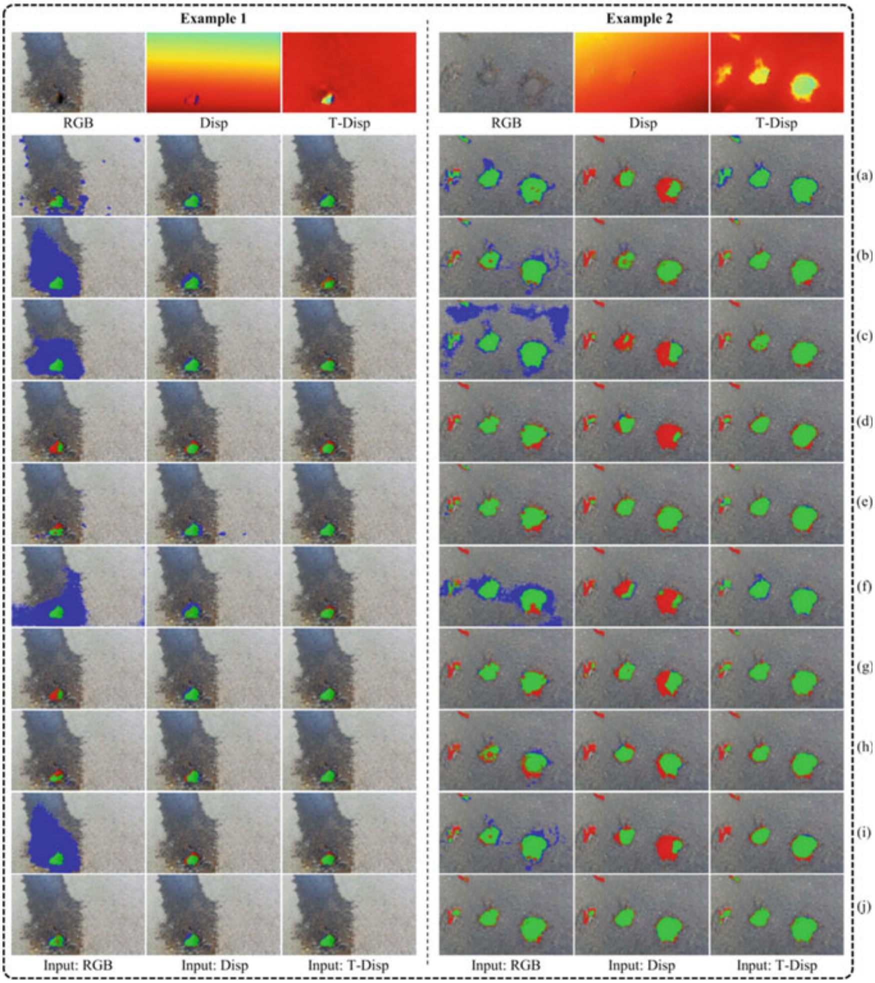


Fig. 4.16 Examples of the experimental results of SoTA CNNs: **a** FCN [27]; **b** U-Net [34]; **c** DenseASPP [165]; **d** DUpsampling [166]; **e** GSCNN [167]; **f** SegNet [37]; **g** DeepLabv3+ [33]; **h** PAN [168]; **i** ESPNet [169]; **j** GAL-DeepLabv3+ [164], where the true-positive, false-positive, and false-negative pixels are shown in green, blue and red, respectively

4.6 Existing Challenges

Multi-visual information fusion is particularly challenging for semantic segmentation tasks in autonomous driving because of its high requirements for accuracy, robustness, and real-time performance. Efficient processing of input information is a prerequisite for accurate perception of complex environments by driverless vehicles. In Sect. 4.2, we summarized the multi-visual information fusion network for semantic

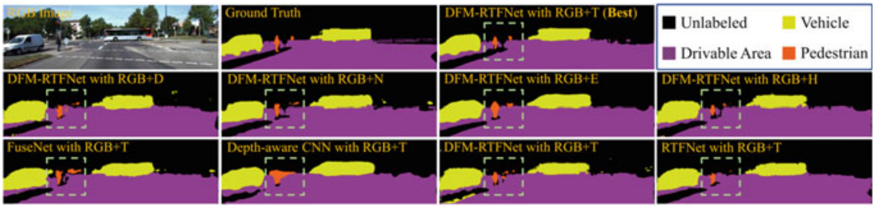


Fig. 4.17 Example of the experimental results on the KITTI semantic segmentation dataset. FuseNet [18], MFNet [76], Depth-aware CNN [73], RTFNet [77], and DFM-RTFNet [171] are all data-fusion networks. Significantly improved regions are marked with green dashed boxes

segmentation. Existing data fusion methods commonly combine RGB images with either thermal or depth images. Thermal images are particularly useful for detecting high-temperature regions, contours of targets, points or lines of sudden temperature changes, trends in temperature, and other related features. On the other hand, depth images are effective in capturing the physical structure of each object in the scene. However, these types of visual information are applicable only to specific scenes and scenarios. Currently, the fusion of RGB images with other visual information has received limited research attention. Therefore, selecting and acquiring the appropriate visual information based on the characteristics of the scenario remains a challenge. Due to its unique characteristics, such as color and texture, RGB data requires different network architectures and fusion methods compared to other types of visual information. While most existing fusion methods for RGB images involve overlaying and stitching, these approaches often fail to account for the spatial and temporal relationships between different types of visual information. As a result, fusing RGB images with other visual information is a significant challenge that requires the development of novel fusion methods. Furthermore, the computational costs and memory requirements associated with these fusion methods must be carefully considered to ensure that they are suitable for use in real-world applications.

4.7 Conclusion

In this chapter, we extensively explored semantic segmentation techniques for autonomous driving, covering both single-modal and data fusion approaches. We began by providing a comprehensive overview of semantic segmentation methods and then delved into the latest techniques for both scenarios, highlighting their significance and contributions to the field. We also presented common datasets that researchers can use to achieve their objectives and satisfy their requirements, along with evaluation metrics for assessing semantic segmentation performance and corresponding benchmarks for several classic datasets. Finally, we discussed various applications of autonomous driving and shared our perspective on the current state-of-the-art of semantic segmentation.

Acknowledgements This work was supported by the National Key R&D Program of China under Grant 2020AAA0108100, the National under Grant 62233013, and the Science and Technology Commission of Shanghai Municipal under Grant 22511104500.

References

1. Wang X-F, Huang D-S, Xu H (2010) An efficient local chan-vese model for image segmentation. Pattern Recognit 43(3):603–618
2. Ess A, Müller T, Grabner H, Van Gool L (2009) Segmentation-based urban traffic scene understanding. In: British machine vision conference (BMVC). Citeseer, p 2
3. Geiger A, Lenz P, Urtasun R (2012) Are we ready for autonomous driving? The kitti vision benchmark suite. In: 2012 IEEE conference on computer vision and pattern recognition (CVPR). IEEE, pp 3354–3361
4. Oberweger M, Wohlhart P, Lepetit V (2015) Hands deep in deep learning for hand pose estimation. arXiv:1502.06807
5. Yoon Y, Jeon H-G, Yoo D, Lee J-Y, So Kweon I (2015) Learning a deep convolutional network for light-field image super-resolution. In: Proceedings of the IEEE international conference on computer vision workshops (ICCV Workshop), pp 24–32
6. Wan J, Wang D, Hoi SCH, Wu P, Zhu J, Zhang Y, Li J (2014) Deep learning for content-based image retrieval: a comprehensive study. In: Proceedings of the 22nd ACM international conference on multimedia, pp 157–166
7. Dickmanns ED, Mysliwetz BD (1992) Recursive 3-D road and relative ego-state recognition. IEEE Trans Pattern Anal Mach Intell 14(02):199–213
8. Wang Y, Zhou Q, Liu J, Xiong J, Gao G, Wu X, Latecki LJ (2019) Lednet: a lightweight encoder-decoder network for real-time semantic segmentation. In: 2019 IEEE international conference on image processing (ICIP). IEEE, pp 1860–1864
9. Wang X, Huang D (2009) A novel density-based clustering framework by using level set method. IEEE Trans Knowl Data Eng 21(11):1515–1531
10. Huang D, Du J (2008) A constructive hybrid structure optimization methodology for radial basis probabilistic neural networks. IEEE Trans Neural Netw 19(12):2099–2115
11. Huang D (1999) Radial basis probabilistic neural networks: Model and application. Int J Pattern Recognit Artif Intell 13(07):1083–1101
12. Zhao Z-Q, Huang D-S, Sun B-Y (2004) Human face recognition based on multi-features using neural networks committee. Pattern Recognit Lett 25(12):1351–1358
13. Couprie C, Farabet C, Najman L, LeCun Y (2013) Indoor semantic segmentation using depth information: 1st international conference on learning representations, iclr 2013. In: 1st international conference on learning representations (ICLR)
14. Farabet C, Couprie C, Najman L, LeCun Y (2012) Learning hierarchical features for scene labeling. IEEE Trans Pattern Anal Mach Intell 35(8):1915–1929
15. Cheng Y, Cai R, Li Z, Zhao X, Huang K (2017) Locality-sensitive deconvolution networks with gated fusion for rgb-d indoor semantic segmentation. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), pp 3029–3037
16. Gupta S, Girshick R, Arbeláez P, Malik J (2014) Learning rich features from rgb-d images for object detection and segmentation. In: European conference on computer vision (ECCV). Springer, pp 345–360
17. Wang J, Wang Z, Tao D, See S, Wang G (2016) Learning common and specific features for rgb-d semantic segmentation with deconvolutional networks. In: European conference on computer vision (ECCV). Springer, pp 664–679
18. Hazirbas C, Ma L, Domokos C, Cremers D (2017) Fusenet: incorporating depth into semantic segmentation via fusion-based cnn architecture. In: Asian conference on computer vision (ACCV). Springer, pp 213–228

19. Song X, Herranz L, Jiang S (2017) Depth cnns for rgb-d scene recognition: learning from scratch better than transferring from rgb-cnns. In: Thirty-first AAAI conference on artificial intelligence
20. Sistu G, Leang I, Yogamani S (2019) Real-time joint object detection and semantic segmentation network for automated driving. arXiv:1901.03912
21. Siam M, Elkerdawy S, Jagersand M, Yogamani S (2017) Deep semantic segmentation for automated driving: taxonomy, roadmap and challenges. In: 2017 IEEE 20th international conference on intelligent transportation systems (ITSC). IEEE, pp 1–8
22. Pan B, Sun J, Leung HYT, Andonian A, Zhou B (2020) Cross-view semantic segmentation for sensing surroundings. IEEE Robot Autom Lett 5(3):4867–4873
23. Yang K, Hu X, Chen H, Xiang K, Wang K, Stiefelhagen R (2020) Ds-pass: detail-sensitive panoramic annular semantic segmentation through swaftnet for surrounding sensing. In: 2020 IEEE intelligent vehicles symposium (IV). IEEE, pp 457–464
24. Liu C-W, Wang H, Guo S, Junaid Bocus M, Chen Q, Fan R (2023) Stereo matching: fundamentals, state-of-the-art, and existing challenges. Springer, submitted for publication
25. Khan MZ, Gajendran MK, Lee Y, Khan MA (2021) Deep neural architectures for medical image semantic segmentation. IEEE Access 9:83 002–83 024
26. Alalwan N, Abozeid A, ElHabshy AA, Alzahrani A (2021) Efficient 3d deep learning model for medical image semantic segmentation. Alex Eng J 60(1):1231–1239
27. Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 3431–3440
28. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on Computer vision and pattern recognition (CVPR), pp 580–587
29. Hariharan B, Arbeláez P, Girshick R, Malik J (2014) Simultaneous detection and segmentation. In: European conference on computer vision (ECCV). Springer, pp 297–312
30. Wu H, Zhang J, Huang K, Liang K, Yu Y (2019) Fastfcn: rethinking dilated convolution in the backbone for semantic segmentation. arXiv:1903.11816
31. Lazebnik S, Schmid C, Ponce J (2006) Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: 2006 IEEE computer society conference on computer vision and pattern recognition (CVPR), vol 2. IEEE, pp 2169–2178
32. Zhao H, Shi J, Qi X, Wang X, Jia J (2017) Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 2881–2890
33. Chen L, Zhu Y, Papandreou G, Schroff F, Adam H (2018) Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV), pp 801–818
34. Ronneberger O, Fischer P, Brox T (2015) U-net: convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention (MICCAI). Springer, pp 234–241
35. Chaurasia A, Culurciello E (2017) Linknet: exploiting encoder representations for efficient semantic segmentation. In: 2017 IEEE visual communications and image processing (VCIP). IEEE, pp 1–4
36. Jégou S, Drozdzal M, Vazquez D, Romero A, Bengio Y (2017) The one hundred layers tiramisu: fully convolutional densenets for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops (CVPR Workshop), pp 11–19
37. Badrinarayanan V, Kendall A, Cipolla R (2017) Segnet: a deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans Pattern Anal Mach Intell 39(12):2481–2495
38. Paszke A, Chaurasia A, Kim S, Culurciello E (2016) Enet: a deep neural network architecture for real-time semantic segmentation. arXiv:1606.02147
39. Sun K, Xiao B, Liu D, Wang J (2019) Deep high-resolution representation learning for human pose estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 5693–5703

40. Florian L-C, Adam SH (2017) Rethinking atrous convolution for semantic image segmentation. In: Conference on computer vision and pattern recognition (CVPR), vol 6
41. Zhao H, Shi J, Qi X, Wang X, Jia J (2017) Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 2881–2890
42. Yu F, Koltun V (2015) Multi-scale context aggregation by dilated convolutions. arXiv:1511.07122
43. Liang-Chieh C, Papandreou G, Kokkinos I, Murphy K, Yuille A (2015) Semantic image segmentation with deep convolutional nets and fully connected crfs. In: International conference on learning representations (ICLR)
44. Zhao H, Zhang Y, Liu S, Shi J, Loy CC, Lin D, Jia J (2018) Psanet: point-wise spatial attention network for scene parsing. In: Proceedings of the European conference on computer vision (ECCV), pp 267–283
45. Wang X, Girshick R, Gupta A, He K (2018) Non-local neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 7794–7803
46. Zhu Z, Xu M, Bai S, Huang T, Bai X (2019) Asymmetric non-local neural networks for semantic segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 593–602
47. Yin M, Yao Z, Cao Y, Li X, Zhang Z, Lin S, Hu H (2020) Disentangled non-local neural networks. In: European conference on computer vision (ECCV). Springer, pp 191–207
48. Fu J, Liu J, Tian H, Li Y, Bao Y, Fang Z, Lu H (2019) Dual attention network for scene segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 3146–3154
49. He J, Deng Z, Zhou L, Wang Y, Qiao Y (2019) Adaptive pyramid context network for semantic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 7519–7528
50. Huang Z, Wang X, Huang L, Huang C, Wei Y, Liu W (2019) Ccnet: criss-cross attention for semantic segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 603–612
51. Huang L, Yuan Y, Guo J, Zhang C, Chen X, Wang J (2019) Interlaced sparse self-attention for semantic segmentation. arXiv:1907.12273
52. Li X, Zhong Z, Wu J, Yang Y, Lin Z, Liu H (2019) Expectation-maximization attention networks for semantic segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 9167–9176
53. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the em algorithm. J R Stat Soc: Ser B (Methodological) 39(1):1–22
54. Zhang H, Dana K, Shi J, Zhang Z, Wang X, Tyagi A, Agrawal A (2018) Context encoding for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 7151–7160
55. Strudel R, Garcia R, Laptev I, Schmid C (2021) Segmenter: transformer for semantic segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 7262–7272
56. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inf Process Syst 30
57. Yuan Y, Chen X, Chen X, Wang J (2019) Segmentation transformer: object-contextual representations for semantic segmentation. arXiv:1909.11065
58. Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, Guo B (2021) Swin transformer: hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 10 012–10 022
59. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S et al (2020) An image is worth 16x16 words: transformers for image recognition at scale. In: International conference on learning representations (ICLR)
60. NicolasCarion F, GabrielSynnaeve NU (2020) Alexanderkirillov, and sergeyzagoruyko. End-to-end object detection with transformers. In: Proceedings of the European conference on computer vision (ECCV)

61. Xie E, Wang W, Yu Z, Anandkumar A, Alvarez JM, Luo P (2021) Segformer: simple and efficient design for semantic segmentation with transformers. Adv Neural Inf Process Syst 34:12 077–12 090
62. Chu X, Tian Z, Wang Y, Zhang B, Ren H, Wei X, Xia H, Shen C (2021) Twins: revisiting the design of spatial attention in vision transformers. Adv Neural Inf Process Syst 34:9355–9366
63. Zhang H, Wu C, Zhang Z, Zhu Y, Lin H, Zhang Z, Sun Y, He T, Mueller J, Manmatha R et al (2022) Resnest: split-attention networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 2736–2746
64. Fan R, Ozgunalp U, Wang Y, Liu M, Pitas I (2022) Rethinking road surface 3-D reconstruction and pothole detection: from perspective transformation to disparity map segmentation. IEEE Trans Cybern 52(7):5799–5808
65. Ming N, Feng Y, Fan R (2022) SDA-SNE: spatial discontinuity-aware surface normal estimation via multi-directional dynamic programming. In: 2022 International conference on 3D vision (3DV), pp 486–494
66. Feng Y, Xue B, Liu M, Chen Q, Fan R (2023) D2NT: a high-performing depth-to-normal translator. In: 2023 IEEE international conference on robotics and automation (ICRA), pp 12360–12366
67. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
68. Jiang J, Zheng L, Luo F, Zhang Z (2018) Rednet: residual encoder-decoder network for indoor rgb-d semantic segmentation. arXiv:1806.01054
69. Sun L, Yang K, Hu X, Hu W, Wang K (2020) Real-time fusion network for rgb-d semantic segmentation incorporating unexpected obstacle detection for road-driving images. IEEE Robot Autom Lett 5(4):5558–5565
70. Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 7132–7141
71. Deng L, Yang M, Li T, He Y, Wang C (2019) Rfbnet: deep multimodal networks with residual fusion blocks for rgb-d semantic segmentation. arXiv:1907.00135
72. Valada A, Mohan R, Burgard W (2020) Self-supervised model adaptation for multimodal semantic segmentation. Int J Comput Vis 128(5):1239–1285
73. Wang W, Neumann U (2018) Depth-aware cnn for rgb-d segmentation. In: Proceedings of the European conference on computer vision (ECCV), pp 135–150
74. Lian Q, Lv F, Duan L, Gong B (2019) Constructing self-motivated pyramid curriculums for cross-domain semantic segmentation: a non-adversarial approach. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 6758–6767
75. Guo M, Wang Z, Yang N, Li Z, An T (2018) A multisensor multiclassifier hierarchical fusion model based on entropy weight for human activity recognition using wearable inertial sensors. IEEE Trans Hum-Mach Syst 49(1):105–111
76. Ha Q, Watanabe K, Karasawa T, Ushiku Y, Harada T (2017) Mfnet: towards real-time semantic segmentation for autonomous vehicles with multi-spectral scenes. In: 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 5108–5115
77. Sun Y, Zuo W, Liu M (2019) Rtfnet: Rgb-thermal fusion network for semantic segmentation of urban scenes. IEEE Robot Autom Lett 4(3):2576–2583
78. Sun Y, Zuo W, Yun P, Wang H, Liu M (2020) Fuseseg: semantic segmentation of urban scenes based on rgb and thermal data fusion. IEEE Trans Autom Sci Eng 18(3):1000–1011
79. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 4700–4708
80. Deng F, Feng H, Liang M, Wang H, Yang Y, Gao Y, Chen J, Hu J, Guo X, Lam TL (2021) Feanet: feature-enhanced attention network for rgb-thermal real-time semantic segmentation. In: 2021 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 4467–4473
81. Zhou W, Lin X, Lei J, Yu L, Hwang J-N (2021) Mffenet: multiscale feature fusion and enhancement network for rgb-thermal urban road scene parsing. IEEE Trans Multimed 24:2526–2538

82. Zhou W, Liu J, Lei J, Yu L, Hwang J-N (2021) Gmnet: graded-feature multilabel-learning network for rgb-thermal urban scene semantic segmentation. IEEE Trans Image Process 30:7790–7802
83. Zhang Q, Zhao S, Luo Y, Zhang D, Huang N, Han J (2021) Abmdrnet: adaptive-weighted bi-directional modality difference reduction network for rgb-t semantic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 2633–2642
84. Yi S, Li J, Liu X, Yuan X (2022) Ccaffmnet: dual-spectral semantic segmentation network with channel-coordinate attention feature fusion module. Neurocomputing 482:236–251
85. Wang H, Fan R, Cai P, Liu M (2021) SNE-Roadseg+: rethinking depth-normal translation and deep supervision for freespace detection. In: 2021 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 1140–1145
86. Fan R, Wang H, Cai P, Liu M (2020) SNE-RoadSeg: incorporating surface normal information into semantic segmentation for accurate freespace detection. In: Proceedings of the European conference on computer vision (ECCV). Springer, pp 340–356
87. Park S-J, Hong K-S, Lee S (2017) Rdfnet: Rgb-d multi-level residual feature fusion for indoor semantic segmentation. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp 4980–4989
88. Gupta S, Arbelaez P, Malik J (2013) Perceptual organization and recognition of indoor scenes from rgb-d images. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 564–571
89. Song S, Lichtenberg SP, Xiao J (2015) Sun rgb-d: a rgb-d scene understanding benchmark suite. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 567–576
90. Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S, Schiele B (2016) The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 3213–3223
91. Pinggera P, Ramos S, Gehrig S, Franke U, Rother C, Mester R (2016) Lost and found: detecting small road hazards for self-driving vehicles. In: 2016 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 1099–1106
92. Dai A, Chang AX, Savva M, Halber M, Funkhouser T, Nießner M (2017) Scannet: richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 5828–5839
93. Silberman N, Hoiem D, Kohli P, Fergus R (2012) Indoor segmentation and support inference from rgbd images. In: European conference on computer vision (ECCV). Springer, pp 746–760
94. Armeni I, Sax S, Zamir AR, Savarese S (2017) Joint 2d-3d-semantic data for indoor scene understanding. arXiv:1702.01105
95. Shivakumar SS, Rodrigues N, Zhou A, Miller ID, Kumar V, Taylor CJ (2020) Pst900: Rgb-thermal calibration, dataset and segmentation network. In: 2020 IEEE international conference on robotics and automation (ICRA). IEEE, pp 9441–9447
96. Xu H, Ma J, Le Z, Jiang J, Guo X (2020) Fusiondn: a unified densely connected network for image fusion. In: Proceedings of the thirty-fourth AAAI conference on artificial intelligence
97. Fritsch J, Kuehnl T, Geiger A (2013) A new performance measure and evaluation benchmark for road detection algorithms. In: International conference on intelligent transportation systems (ITSC)
98. Ros G, Sellart L, Materzynska J, Vazquez D, Lopez AM (2016) The synthia dataset: a large collection of synthetic images for semantic segmentation of urban scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 3234–3243
99. Everingham M, Winn J (2012) The pascal visual object classes challenge 2012 (voc2012) development kit. Pattern Anal Stat Model Comput Learn Tech Rep 2007:1–45
100. Mottaghi R, Chen X, Liu X, Cho N-G, Lee S-W, Fidler S, Urtasun R, Yuille A (2014) The role of context for object detection and semantic segmentation in the wild. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 891–898

101. Chen X, Mottaghi R, Liu X, Fidler S, Urtasun R, Yuille A (2014) Detect what you can: detecting and representing objects using holistic models and body parts. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 1971–1978
102. Gould S, Fulton R, Koller D (2009) Decomposing a scene into geometric and semantically consistent regions. In: 2009 IEEE 12th international conference on computer vision (ICCV). IEEE, pp 1–8
103. Russell BC, Torralba A, Murphy KP, Freeman WT (2008) Labelme: a database and web-based tool for image annotation. Int J Comput Vis 77(1):157–173
104. Criminisi A et al (2004) Microsoft research cambridge object recognition image database. http://research.microsoft.com/vision/cambridge/recognition
105. Everingham M, Van Gool L, Williams CK, Winn J, Zisserman A (2010) The pascal visual object classes (voc) challenge. Int J Comput Vis 88(2):303–338
106. Hoiem D, Efros AA, Hebert M (2007) Recovering surface layout from an image. Int J Comput Vis 75(1):151–172
107. Ipeirotis PG (2010) Analyzing the amazon mechanical turk marketplace. XRDS: Crossroads. ACM Mag Stud 17(2):16–21
108. Hariharan B, Arbeláez P, Bourdev L, Maji S, Malik J (2011) Semantic contours from inverse detectors. In: 2011 international conference on computer vision (ICCV). IEEE, pp 991–998
109. Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A (2011) The PASCAL visual object classes challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html
110. Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft coco: common objects in context. In: European conference on computer vision (ECCV). Springer, pp 740–755
111. Brostow GJ, Fauqueur J, Cipolla R (2009) Semantic object classes in video: a high-definition ground truth database. Pattern Recognit Lett 30(2):88–97
112. Kuutti S, Bowden R, Jin Y, Barber P, Fallah S (2020) A survey of deep learning applications to autonomous vehicle control. IEEE Trans Intell Transp Syst 22(2):712–733
113. Menze M, Geiger A (2015) Object scene flow for autonomous vehicles. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 3061–3070
114. Alhaija H, Mustikovela S, Mescheder L, Geiger A, Rother C (2018) Augmented reality meets computer vision: efficient data generation for urban driving scenes. Int J Comput Vis
115. Gaidon A, Wang Q, Cabon Y, Vig E (2016) Virtual worlds as proxy for multi-object tracking analysis. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 4340–4349
116. Zhou B, Zhao H, Puig X, Fidler S, Barriuso A, Torralba A (2017) Scene parsing through ade20k dataset. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 633–641
117. Zhou B, Zhao H, Puig X, Xiao T, Fidler S, Barriuso A, Torralba A (2019) Semantic understanding of scenes through the ade20k dataset. Int J Comput Vis 127(3):302–321
118. Xiao J, Hays J, Ehinger KA, Oliva A, Torralba A (2010) Sun database: large-scale scene recognition from abbey to zoo. In: 2010 IEEE computer society conference on computer vision and pattern recognition (CVPR). IEEE, pp 3485–3492
119. Zhou B, Lapedriza A, Xiao J, Torralba A, Oliva A (2014) Learning deep features for scene recognition using places database. Adv Neural Inf Process Syst 27
120. Neuhold G, Ollmann T, Rota Bulo S, Kontschieder P (2017) The mapillary vistas dataset for semantic understanding of street scenes. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp 4990–4999
121. Huang X, Cheng X, Geng Q, Cao B, Zhou D, Wang P, Lin Y, Yang R (2018) The apolloscape dataset for autonomous driving. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops (CVPR), pp 954–960
122. Kim B, Yim J, Kim J (2020) Highway driving dataset for semantic video segmentation. arXiv:2011.00674

123. Yogamani S, Hughes C, Horgan J, Sistu G, Varley P, O'Dea D, Uricár M, Milz S, Simon M, Amende K et al (2019) Woodscape: a multi-task, multi-camera fisheye dataset for autonomous driving. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 9308–9318
124. Varma G, Subramanian A, Namboodiri A, Chandraker M, Jawahar C (2019) Idd: a dataset for exploring problems of autonomous navigation in unconstrained environments. In: 2019 IEEE winter conference on applications of computer vision (WACV). IEEE, pp 1743–1751
125. Geyer J, Kassahun Y, Mahmudi M, Ricou X, Durgesh R, Chung AS, Hauswald L, Pham VH, Mühlegg M, Dorn S et al (2020) A2d2: audi autonomous driving dataset. arXiv:2004.06320
126. Alberti E, Tavera A, Masone C, Caputo B (2020) Idda: a large-scale multi-domain dataset for autonomous driving. IEEE Robot Autom Lett 5(4):5526–5533
127. Xiao J, Owens A, Torralba A (2013) Sun3d: a database of big spaces reconstructed using sfm and object labels. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp 1625–1632
128. Janoch A, Karayev S, Jia Y, Barron JT, Fritz M, Saenko K, Darrell T (2013) A category-level 3D object dataset: putting the kinect to work. In: Consumer depth cameras for computer vision. Springer, pp 141–165
129. Tao A, Sapra K, Catanzaro B (2020) Hierarchical multi-scale attention for semantic segmentation. arXiv:2005.10821
130. Borse S, Wang Y, Zhang Y, Porikli F (2021) Inverseform: a loss function for structured boundary-aware segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 5901–5911
131. Chen L-C, Lopes RG, Cheng B, Collins MD, Cubuk ED, Zoph B, Adam H, Shlens J (2020) Naive-student: leveraging semi-supervised learning in video sequences for urban scene segmentation. In: Computer vision-ECCV, (2020) 16th European conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16. Springer, pp 695–714
132. Chen Z, Duan Y, Wang W, He J, Lu T, Dai J, Qiao Y (2022) Vision transformer adapter for dense predictions. arXiv:2205.08534
133. Chen L-C, Wang H, Qiao S (2020) Scaling wide residual networks for panoptic segmentation. arXiv:2011.11675
134. Yuan Y, Chen X, Wang J (2020) Object-contextual representations for semantic segmentation. In: Computer vision-ECCV, (2020) 16th European conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16. Springer, pp 173–190
135. Cheng B, Collins MD, Zhu Y, Liu T, Huang TS, Adam H, Chen L-C (2020) Panoptic-deeplab: a simple, strong, and fast baseline for bottom-up panoptic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 12 475–12 485
136. Zhang X, Xu H, Mo H, Tan J, Yang C, Wang L, Ren W (2021) Dcnas: densely connected neural architecture search for semantic image segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 13 956–13 967
137. Mohan R, Valada A (2021) Efficientps: efficient panoptic segmentation. Int J Comput Vis (IJCV) 129(5):1551–1579
138. Wang H, Zhu Y, Green B, Adam H, Yuille A, Chen L-C (2020) Axial-deeplab: stand-alone axial-attention for panoptic segmentation. In: Computer vision-ECCV, (2020) 16th European conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV. Springer, pp 108–126
139. Ganeshan A, Vallet A, Kudo Y, Maeda S-I, Kerola T, Ambrus R, Park D, Gaidon A (2021) Warp-refine propagation: semi-supervised auto-labeling via cycle-consistency. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 15 499–15 509
140. Cai Y, Dai L, Wang H, Li Z (2021) Multi-target pan-class intrinsic relevance driven model for improving semantic segmentation in autonomous driving. IEEE Trans Image Process 30:9069–9084
141. Zhu Y, Sapra K, Reda FA, Shih KJ, Newsam S, Tao A, Catanzaro B (2019) Improving semantic segmentation via video propagation and label relaxation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 8856–8865

142. Bevandić P, Oršić M, Grubišić I, Šarić J, Šegvić S (2022) Multi-domain semantic segmentation with overlapping labels. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision (WACV), pp 2615–2624
143. Lambert J, Liu Z, Sener O, Hays J, Koltun V (2020) Mseg: a composite dataset for multi-domain semantic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 2879–2888
144. Erkent Ö, Laugier C (2020) Semantic segmentation with unsupervised domain adaptation under varying weather conditions for autonomous vehicles. IEEE Robot Autom Lett 5(2):3580–3587
145. Bolte J-A, Kamp M, Breuer A, Homoceanu S, Schlicht P, Huger F, Lipinski D, Fingscheidt T (2019) Unsupervised domain adaptation to improve image segmentation quality both in the source and target domain. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, p 0
146. Yang G, Zhao H, Shi J, Deng Z, Jia J (2018) Segstereo: exploiting semantic information for disparity estimation. In: Proceedings of the European conference on computer vision (ECCV), pp 636–651
147. Klingner M, Termöhlen J-A, Mikolajczyk J, Fingscheidt T (2020) Self-supervised monocular depth estimation: solving the dynamic object problem by semantic guidance. In: European conference on computer vision (ECCV). Springer, pp 582–600
148. Ochs M, Kretz A, Mester R (2019) Sdnet: semantically guided depth estimation network. In: German conference on pattern recognition (GCPR). Springer, pp 288–302
149. Kong S, Fowlkes C (2018) Pixel-wise attentional gating for parsimonious pixel labeling. arXiv:1805.01556
150. Ozgunalp U, Fan R, Ai X, Dahnoun N (2017) Multiple lane detection algorithm based on novel dense vanishing point estimation. IEEE Trans Intell Transp Syst 18(3):621–632
151. Fan R, Wang H, Cai P, Wu J, Bocus MJ, Qiao L, Liu M (2022) Learning collision-free space detection from stereo images: homography matrix brings better data augmentation. IEEE/ASME Trans Mechatron 27(1):225–233
152. Chen Z, Chen Z (2017) Rbnet: a deep neural network for unified road and road boundary detection. In: International conference on neural information processing. Springer, pp 677–687
153. Gu S, Zhang Y, Yang J, Alvarez JM, Kong H (2019) Two-view fusion based convolutional neural network for urban road detection. In: 2019 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 6144–6149
154. Gu S, Zhang Y, Tang J, Yang J, Kong H (2019) Road detection through crf based LiDAR-camera fusion. In: 2019 international conference on robotics and automation (ICRA). IEEE, pp 3832–3838
155. Caltagirone L, Bellone M, Svensson L, Wahde M (2019) LiDAR-camera fusion for road detection using fully convolutional neural networks. Robot Auton Syst 111:125–131
156. Sun J-Y, Kim S-W, Lee S-W, Kim Y-W, Ko S-J (2019) Reverse and boundary attention network for road segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision workshops, p 0
157. Fan R, Ai X, Dahnoun N (2018) Road surface 3D reconstruction based on dense subpixel disparity map estimation. IEEE Trans Image Process 27(6):3025–3035
158. Ma N, Fan J, Wang W, Wu J, Jiang Y, Xie L, Fan R (2022) Computer vision for road imaging and pothole detection: a state-of-the-art review of systems and algorithms. Transp Safety Environ 4(4):tdac026
159. Fan R, Liu M (2020) Road damage detection based on unsupervised disparity map segmentation. IEEE Trans Intell Transp Syst 21(11):4906–4911
160. Guo S, Jiang Y, Li J, Zhou D, Su S, Junaid Bocus M, Zhu X, Chen Q, Fan R (2023) Road environment perception for safe and comfortable driving. Springer, submitted for publication
161. Fan R, Ozgunalp U, Hosking B, Liu M, Pitas I (2020) Pothole detection based on disparity transformation and road surface modeling. IEEE Trans Image Process 29:897–908

162. Fan J, Bocus MJ, Hosking B, Wu R, Liu Y, Vityazev S, Fan R (2021) Multi-scale feature fusion: learning better semantic segmentation for road pothole detection. In: 2021 IEEE international conference on autonomous systems (ICAS). IEEE, pp 1–5
163. Fan R, Wang H, Bocus MJ, Liu M (2020) We learn better road pothole detection: from attention aggregation to adversarial domain adaptation. In: Computer vision-ECCV, (2020) Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16. Springer, pp 285–300
164. Fan R, Wang H, Wang Y, Liu M, Pitas I (2021) Graph attention layer evolves semantic segmentation for road pothole detection: a benchmark and algorithms. IEEE Trans Image Process 30:8144–8154
165. Yang M, Yu K, Zhang C, Li Z, Yang K (2018) Denseaspp for semantic segmentation in street scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 3684–3692
166. Tian Z, He T, Shen C, Yan Y (2019) Decoders matter for semantic segmentation: data-dependent decoding enables flexible feature aggregation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 3126–3135
167. Takikawa T, Acuna D, Jampani V, Fidler S (2019) Gated-scnn: gated shape cnns for semantic segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 5229–5238
168. Li H, Xiong P, An J, Wang L (2018) Pyramid attention network for semantic segmentation. arXiv:1805.10180
169. Mehta S, Rastegari M, Caspi A, Shapiro L, Hajishirzi H (2018) Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In: Proceedings of the European conference on computer vision (ECCV), pp 552–568
170. Wang H, Fan R, Sun Y, Liu M (2020) Applying surface normal information in drivable area and road anomaly detection for ground mobile robots. In: 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 2706–2711
171. Wang H, Fan R, Sun Y, Liu M (2022) Dynamic fusion module evolves drivable area and road anomaly detection: a benchmark and algorithms. IEEE Trans Cybern 52(10):10 750–10 760

Chapter 5
3D Object Detection in Autonomous Driving

Peng Yun, Yuxuan Liu, Xiaoyang Yan, Jiahang Li, Jiachen Wang, Lei Tai, Na Jin, Rui Fan, and Ming Liu

Abstract 3D object detection is an important perception module in autonomous driving systems. It recognizes sensor observations and predicts locations, sizes and orientations of key objects, which provides both semantic and spatial information for high-level decision making. In this chapter, we first introduce and analyze the properties of commonly used perceptual sensors in autonomous vehicles: cameras, LiDARs and RADARs. Then we define the research problem, detail the assumptions and introduce evaluation metrics of 3D object detection in the context of autonomous

Peng Yun, Yuxuan Liu, Xiaoyang Yan—equal contribution.
Rui Fan, Ming Liu—co-corresponding author.

P. Yun (✉) · Y. Liu · X. Yan · N. Jin · M. Liu
Hong Kong University of Science and Technology, Hong Kong, Hong Kong
e-mail: pyun@ust.hk

Y. Liu
e-mail: yliuhb@ust.hk

X. Yan
e-mail: xyanaq@ust.hk

N. Jin
e-mail: njinab@ust.hk

M. Liu
e-mail: eelium@ust.hk

J. Li · R. Fan
Tongji University, Shanghai, China
e-mail: 2230745@tongji.edu.cn

R. Fan
e-mail: rfan@tongji.edu.cn

J. Wang
Jilin University, Changchun, China
e-mail: wangjc2119@mails.jlu.edu.cn

L. Tai
Huawei Autonomous Driving Solutions, Shanghai, China
e-mail: ltai@connect.ust.hk

R. Fan et al. (eds.), *Autonomous Driving Perception*, Advances in Computer Vision
and Pattern Recognition, https://doi.org/10.1007/978-981-99-4287-9_5

driving. The main body reviews the state-of-the-art techniques and categorize them into camera-based, LiDAR-based, RADAR-based and multi-sensor fusion methods. For each method, we point out the main problems and their existing solutions. By analyzing the limitations of existing methods, we propose promising directions and open problems for future research.

5.1 Introduction

Perception is the primary component in an autonomous driving system. It takes charge of encoding the sensor readings to understand its surroundings. Accurate perception results act as the foundation of decision making. Object detection is one of the important problems in the perception of autonomous driving (Fig. 5.1). It allows vehicles to recognize and locate the key objects they are concerned about (like cars, pedestrians, and cyclists) in the 3D space from the sensor readings. Previously, object detection in autonomous driving was mostly implemented based on the images captured by vehicle-mounted cameras, called image-based object detection [1–4]. Compared to image-based object detection, 3D object detection is a more challenging problem. It is implemented based on various sensors, including cameras, LiDARs and RADARs. Its searching space is larger than image-based object detection due to the extra z-axis and the continuous coordinates. It requires more variables to estimate, including the z-axis position, height as well as the heading orientations. The extra estimation results of 3D object detection provide more information for decision-making in autonomous vehicles.

One challenge for 3D object detection arises from sensor limitations. Cameras suffer from foreshortening, flickering effects, and over-exposure problems; LiDARs and RADARs suffer from low-resolution and sparse data representations. Another challenge arises from environmental variations such as lighting and weather conditions. These factors significantly influence camera-based 3D object detection since the appearance information captured by cameras could be much different in various conditions. The density of the point clouds captured by LiDARs will be reduced by half on rainy days [5]. Besides, occlusion causes performance degradation of object

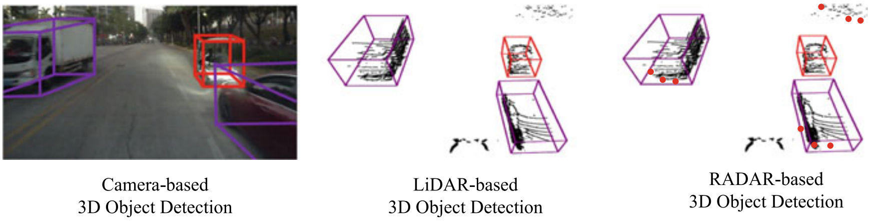


Fig. 5.1 3D object detection with various sensors. (The red dots in the RADAR-based 3D object detection sub-figure denote RADAR observations)

detection. When one object blocks the view of another, it results in partial or complete invisibility of the object.

There have been some survey papers published on 3D object detection in autonomous driving and its related fields [6–8]. However, they mainly focused on the camera- and the LiDAR-based methods and did not review the RADAR-based work. Recently, many state-of-the-art methods have emerged but none are covered in existing surveys. In this chapter, we consider the vehicle-mounted perceptual sensors: cameras, LiDARs, and RADARs, and discuss detection approaches based on uni-modal as well as multi-modal sensor observations. We also provide a more fine-grade classification framework for this problem, and point out crucial problems in each category to help researchers in this field. We claim the following contributions:

- This chapter provides a fine-grade classification framework for current 3D object detection methods in autonomous driving. It generally classifies the existing methods according to the data modal. Besides, it defines the crucial problems for each subcategory and further classifies the methods according to the solution they used.
- Compared to the earlier surveys, this chapter covers the most recent and advanced work. It provides the readers with an in-depth review of the state-of-the-art methods.
- This chapter proposes promising research directions in this field.

In this survey, we first introduce the background concepts and terminology of 3D object detection in Sect. 5.2. We classify the existing research work into camera-based methods, LiDAR-based methods, RADAR-based methods, and multi-sensor fusion methods according to the sensor modality in Sects. 5.3–5.6. We then propose promising research directions for 3D object detection in Sect. 5.7 and concludes this chapter in Sect. 5.8.

5.2 Background Concepts

In this section, we will first introduce the problem definition and assumptions of 3D object detection. Then we describe commonly used perceptual sensors, datasets as well as evaluation metrics.

5.2.1 Problem Definition and Assumptions

We define 3D object detection as a pattern recognition problem. The context of autonomous driving provides this problem with some assumptions which make it different from indoor or aerial applications. We denote $x \in X$ as an input, which can be a dense representation with shape $[W, H, C]$ like images, or a sparse representation with shape $[N, C]$. Let $y \in Y$ be a target, which is a set of 3D bounding boxes $\{B^i_{3D} = [cls, x_c, y_c, z_c, l, w, h, \theta] | i = 1, ..., n\}$, where cls is the class of the bounding box, $[x_c, y_c, z_c]^T$ is the box center, w, h, l denote the box sizes along the x,y,z-axes respectively (x and y axes define the ground plane in the right-handed coordinate system), and θ denotes the box rotation angle along the z-axis in range of $[0, \pi)$.

We denote $\mathcal{L}$ as a loss function, and $R[f]$ as the expected risk, i.e. $R[f] = \mathbb{E}_{x,y\sim P(X,Y)}[\mathcal{L}(f(x), y)]$, where $P(X, Y)$ is the true data distribution. We denote $f_{A(S)} : X \rightarrow Y$ as a model learned by a learning algorithm A using a training dataset $S := S_m := \{(x_i, y_i)\}^m_{i=1}$ of size m. We adopt $R_S[f]$ as the empirical risk of f as $R_S[f] = \frac{1}{m}\sum^m_{i=1} \mathcal{L}(f(x_i, y_i))$ with $S = \{(x_i, y_i)\}^m_{i=1}$. The goal of 3D object detection is defined as the minimization of the expected risk $R[f_{A(S)}]$. We typically minimize the non-computable expected risk $R[f_{A(S)}]$ by minimizing the computable empirical risk $R_S[f_{A(S)}]$.

The context of autonomous driving implies multiple assumptions for 3D object detection. Firstly, the semantic categories should be frequent and meaningful on the road, like vehicles, pedestrians, cyclists, barriers, etc. The objects like tables and books are ignored. Secondly, the scenes of interest are large-scale outdoor driving scenes across hundreds of meters. The indoor scenes are not considered. Thus it suffers less from stacking problems, like a pile of books stacked on a table. Thirdly, only the rotation along the z-axis is considered. It is a reasonable assumption since most objects on the road we are concerned about (like vehicles, pedestrians, and cyclists) have only yaw heading angles. Some works [9–11] also include ground assumptions whereby key objects are located on the same ground as the ego vehicle to simplify this problem.

5.2.2 Sensors

The commonly used perceptual sensors in autonomous driving include cameras, LiDARs and RADARs. In this section, we will discuss both advantages and limitations of each sensor in 3D object detection.

5.2.2.1 Cameras

Cameras are commonly used in our daily life. They are passive sensors similar to human vision and provide appearance information of their perception regions. An image captured from a camera is digitally represented in a dense array I having a dimension of $[H, W, 3]$, where H and W denote the height and width of the image. Each entry of I is an integer ranging from 0 to 255, representing the pixel intensity. The properties of cameras are listed in the following paragraphs.

Rich textures: Textures refer to information about the spatial arrangement of color or intensities in an image, such as edges, lines, and color patches. Images provide rich texture information of the surrounding environments.

Implicit geometrical shape information: Geometrical shape information is helpful for object detection, and such information is implicitly embedded inside the images and can be extracted by analyzing the contours of the patches or the relationship to their neighborhoods.

No depth information: RGB cameras provide no depth information. Even though stereo camera setups can recover the depth information under geometrical constraints, the extra computation for depth recovery is costly [13]. With the booming of deep learning, the depth information can be estimated in a relatively accurate and fast manner with GPU acceleration [14, 15]. However, deep neural networks suffer from bad generalization ability and require fine-tuning if the data distributions are changed. Current RGB-D cameras can provide depth information, but they have short detection range (less than 10 m), and most of them cannot be used in outdoor scenes due to the sunlight effects, which make them unsuitable for autonomous driving applications.

Long detection Range: A common RGB camera can capture the appearance of key objects like cars and pedestrians 200 m away.

The sensor limitations of cameras are derived from the above-mentioned properties. On one hand, cameras suffer from a lack of depth information. 3D object detection requires estimating the accurate locations, scales, and orientations in a 3D coordinate system. The depth information can be used to directly project each pixel on the image plane to the 3D space. Given the 3D coordinates of each pixel, the location of the key objects is much easier to estimate [16, 17]. On the other hand, rich texture information brings both benefits and limitations. They help distinguish key objects but make the captured image significantly different in various lighting or weather conditions.

5.2.2.2 LiDARs

LiDARs are active sensors and provide the position as well as the reflection intensity of each detected point. A point cloud retrieved from a LiDAR is commonly represented as a point list $P = \{p_i | i = 1, ..., n\}$ (Fig. 5.2b), where each p_i is a vector representing its coordinates in the continuous 3D space and its reflection intensity. The properties of LiDARs are listed in the following paragraphs.

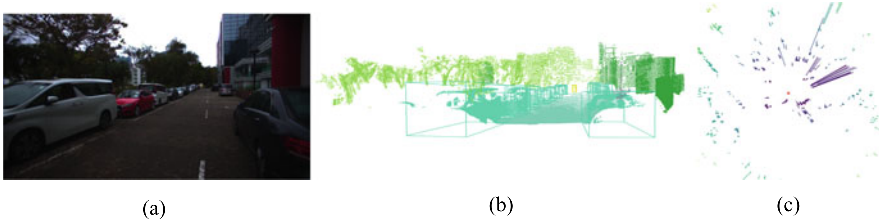

(a) (b) (c)

Fig. 5.2 A visualization of sensor data. **a** camera data; **b** LiDAR data; **c** RADAR data rendered from the nuScenes dataset [12]

Available spatial information: Each point is associated with its coordinates in the LiDAR frame, which helps estimate object size (dimension) and the six-degree-of-freedom (DoF) pose accurately.

Explicit structure and shape information: The points are naturally clustered and separated in a point cloud (Fig. 5.2b). The distribution of points in a local region preserves the structure and shape information.

Sparsity: There is a mass of no-point regions in the point cloud as shown in Fig. 5.2b. The points are distributed loosely in 3D space and are not arranged tightly side-by-side in a grid.

Unordered point list: A point cloud is represented as an unordered point list. The order of the points does not distinguish point clouds. For instance, if $P_1 = [p_a, p_b, p_c]$, $P_2 = [p_b, p_c, p_a]$, we still have $P_1 = P_2$.

Limited detection range: The detection range of LiDARs is always limited to under 100 m. The distant objects can only be detected with a few points.

The sensor limitations of LiDARs are derived from the above-mentioned properties. Firstly, it lacks appearance information and can only use geometrical information to recognize key objects. However, it is hard to recognize their semantic classes for those distant objects detected with only few points. Secondly, the sparsity and the unordered point list representation in the continuous space make point clouds unsuitable for convolution operations. The translation invariance of convolution operations has helped achieve state-of-the-art performance on image recognition tasks. Researchers have to convert the point clouds into a convolutional-friendly representation if they want to adopt convolution neural networks (CNNs) as feature extractors. Otherwise, new feature extractors need to be designed considering the properties above. Thirdly, the density of the point clouds captured by LiDARs might be reduced by half on rainy or snowy days, which may cause false negatives in extreme weather [5].

5.2.2.3 RADARs

In this chapter, we focus on automotive RADARs. The physical mechanisms of RADARs are similar to LiDARs, but they are implemented with microwaves instead of light waves. The on-chip processors of RADARs process the raw data, and finally, a point cloud can be retrieved from RADARs. Different from the point clouds obtained from LiDARs, they are in the 2D space and much sparser (Fig. 5.2c). Besides, the velocity of each point can be measured by RADARs through the Doppler effect. The properties of RADARs are similar to LiDARs except for the following points.

Low resolution: Because of the longer wavelength, the resolution of RADARs is lower than LiDARs. As a result, the point clouds of RADARs are much sparser than LiDARs' and are less precise in terms of localization.

Moderate detection range: The detection range of RADARs is longer than the LiDAR range but shorter than that of cameras. There are different types of RADARs suitable for different ranges: short-range RADARs (less than 30 m), moderate-range RADARs (less than 100 m), and long-range RADARs (less than 250 m).

Robust in extreme weathers: The point clouds of RADARs are much more robust in harsh environments (poor lighting and weather conditions, as well as extreme temperatures) than LiDARs [12], which allows it to be commonly used in blind-spot detection and collision avoidance.

The low resolution is one of the major limitations of RADARs in 3D object detection. It provides 2D locations of each point only (the 2D space defined by the x-axis and y-axis), which limits the possibility of height estimation. Besides, even though its detection range can be as high as 250 m, the automotive RADARs become unreliable at 10–15m when working in real-world scenes due to reflections from other objects [18]. The points of a RADAR projected onto a car 50m away can be less than 10 points [12]. Whether RADARs can accurately capture an object largely depends on its reflection strength. The reflections of radar signals from cars are always strong, while pedestrians and cyclists are smaller in size and have relatively few hard or metallic surfaces to reflect radar signals. For instance, a child standing next to a vehicle can be overlooked by a RADAR system. These are the reasons why 3D object detection algorithms based on automotive RADARs are uncommon.

5.2.2.4 Summary

The comparison between the properties of cameras, LiDARs, and RADARs are shown in Table 5.1. Due to the rich appearance or spatial information, cameras and LiDARs can be used independently for 3D object detection. There are numerous successful implementations solely based on camera or LiDAR data. In contrast, there are few 3D object detector based on automotive RADAR data only, since its data is too sparse and insufficient to estimate semantic classes.

Since different types of sensors have their pros and cons in different conditions, joint treatment of sensor data is essential for 3D object detection [12]. Multi-sensor fusion is a popular research direction in 3D object detection, which provides not only complementary but also redundancy in the face of sabotage, failures, adverse conditions and blind spots [19] (Fig. 5.3).

Table 5.1 Comparison between different types of sensors

	Cameras	LiDARs	RADARs
Information for object detection	Appearance	Spatial	Spatial & velocity
Data representations	Dense array	Unordered point list	Unordered point list
Robustness to extreme conditions	*	**	***
Detection range	$\geq 250m$	$\leq 100m$	$\leq 250m$

The number of '*'s represents its robustness level

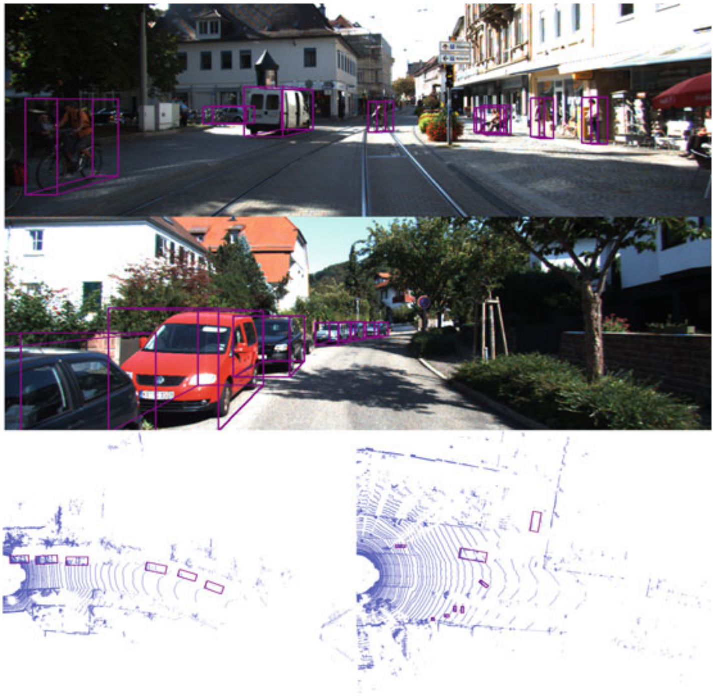


Fig. 5.3 Visualization rendered from the KITTI 3D object detection dataset [20]. The bottom row contains the BEV images of the two scenes in the upper rows

5.2.3 Public Datasets

Data is indispensable to supervised learning algorithms. In this section, we will introduce three representative 3D object detection datasets: KITTI [20], nuScenes [12] and Waymo [21] datasets.

5.2.3.1 KITTI Dataset

KITTI dataset is one of the most commonly used datasets in autonomous driving [20]. Its recording platform is a standard station wagon with two color and two grayscale PointGrey Flea2 video cameras, a Velodyne HDL-64E 3D laser scanner, a GPS/IMU localization unit with RTK correction signals [20]. All these cameras were rectified and well-calibrated with LiDAR and the GPS/IMU unit. The data collection scenes include well-structured highways, complex urban areas, and narrow countryside roads.

3D object detection is one of the benchmarks provided in the KITTI challenges. The dataset includes 7,481 training frames and 7,518 test frames, all of which come with sensor calibration information and annotated 3D boxes surrounding objects of interest. Along with the sensor data from LiDARs and cameras, a 3D object detection benchmark is provided for researchers to test their detection methods. Annotations are classified into three categories, namely easy, moderate, and hard cases, based on the size of objects, the degree of occlusion, and the level of truncation. The metrics that the KITTI dataset adopts consist of AP_{2D}, AP_{3D}, and AP_{BEV} for precision evaluation, and AOS for orientation evaluation, which will be discussed in Sect. 5.2.4.

Even though the KITTI dataset is well-known and popular, some limitations still exist. Firstly, all the measurements of KITTI were collected during the daytime and mostly under sunny conditions. It is hard to conduct research on the generalization ability and robustness against extreme conditions on the KITTI dataset. Secondly, RADARs were not considered in the KITTI recording platform, which limits the research output on RADAR-based 3D object detections. In addition, the class frequency is highly unbalanced: 75% car, 4% cyclist, and 15% pedestrian [22]. As a result, some 3D object detection algorithms were only tested on the car class due to insufficient samples of cyclists and pedestrians. Moreover, the majority of objects tend to have a dominant orientation, with their front facing towards the ego-vehicle.

5.2.3.2 NuScenes Dataset

The nuScenes dataset was specifically collected for object detection and tracking [12]. This dataset is designed for the driving context and contains sensory data from a fully autonomous vehicle, including 6 cameras, 5 RADARs, and 1 LiDAR, all with a complete 360-degree field of view. All the sensors are well-calibrated. They collected data using two Renault Zoe supermini electric cars equipped with an identical sensor layout, and the cars were driven in Boston and Singapore allowing researchers to evaluate their work across different geographic locations.

There are 40K frames annotated with 3D bounding boxes of 23 categories, including car, adult, child, barrier, and animal, which are in finer grain degrees compared to the KITTI dataset. On average, the nuScenes dataset has seven pedestrians and twenty vehicles per frame. Besides, they considered the effects of extreme conditions (nighttime and rainy days). The rainy and night frames account for 11.6% and 19.4% in the full 40k annotated frames. To evaluate the generalization capability of algorithms on such a dataset, they also collected data from different scene locations: Boston: 55%, Singapore-OneNorth: 21.5%, Singapore-Queenstown: 13.5%, Singapore-HollandVillage: 10%. They adopted AP_{BEV} and nuScenes detection score (NDS) as evaluation metrics. NDS is a new metric they proposed for evaluating the results on the nuScenes dataset. Half of the *NDS* is based on AP_{BEV}[1] while the remaining half of the measurements evaluates the accuracy of the detections based on box location, size, orientation, attributes, and velocity, calculated using their corresponding distances [12].

[1] It is noted that, different from the method calculating with intersection over union in the KITTI dataset, nuScenes calculate the AP_{BEV} with the 2D center distance on the ground plane.

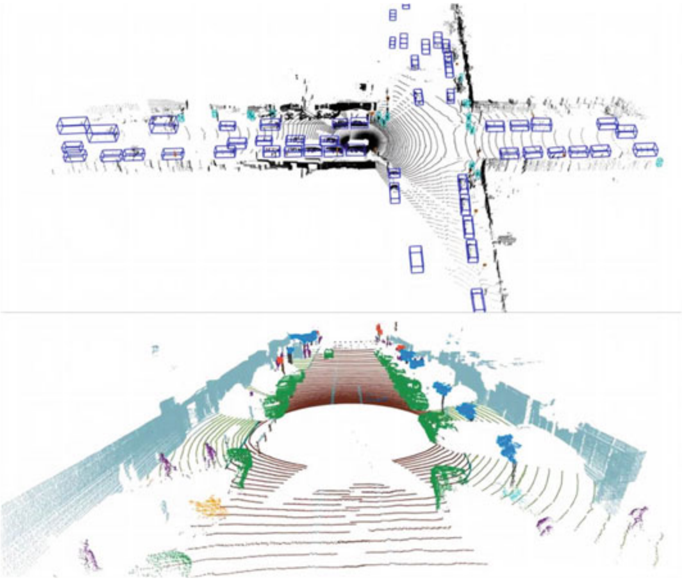


Fig. 5.4 Visualization rendered from Waymo Open Dataset [21]. The upper row is for object detection and tracking, while the lower row is for semantic segmentation

Compared to the KITTI dataset, the nuScenes dataset provides more data annotations covering more driving conditions. The available RADAR data will motivate researchers to contribute more on RADAR-based 3D object detection algorithms. The possible improvement from dataset aspects is two-fold. Firstly, more extreme conditions should be considered. There are some omitted conditions like snowy and stormy days. The ice layer formed on RADAR sensors may cause performance degradation, and the strong wind may cause physical vibration of sensors. Secondly, multi-LiDAR settings are also one type of common setups in autonomous vehicles, which is not considered in the current datasets (Fig. 5.4).

5.2.3.3 Waymo Open Dataset

Waymo dataset [21] is collected by the following sensors: 1 mid-range LiDAR, four short-range LiDARs, and five cameras (front and sides). The data is collected from various places, including Los Angeles, Detroit, Mountain View, San Francisco, Seattle, and Phoenix. This dataset also includes various environments, objects, and weather conditions.

The Waymo Open Dataset provides object detection and tracking capabilities with a collection of 12.6 million 3D bounding box labels, each with tracking IDs, for four object categories, namely vehicles, pedestrians, cyclists, and signs, on LiDAR data. Additionally, the dataset also contains 11.8 million 2D bounding box labels, also

Table 5.2 Comparison between KITTI [20], nuScenes [12], and Waymo [21] datasets

Dataset	# ann. frames	# 3D boxes	Night	Rain	Snow	Locations
KITTI	15k	200k	No	No	No	Karlsruhe
nuScenes	40k	1.4M	Yes	Yes	No	Boston, SG
Waymo	39k	12.6M	Yes	Yes	No	LA, Detroit, MV, SFO, Seattle, Phoenix

with tracking IDs, on camera data. These LiDAR labels are 3D 7-DoF bounding boxes with globally unique tracking IDs. The metrics used by Waymo Open Dataset include AP_{2D}, AP_{3D}, and APH for precision evaluation.

Being the most recent among these well-known open-source datasets, the Waymo dataset uses better sensors for data acquisition, resulting in denser LiDAR point clouds, better image data quality, and more objects in each frame. However, this is also a challenge for developers because larger data volumes bring more computational and reasoning pressure on processors.

5.2.3.4 Summary

The comparison between representative datasets is shown in Table 5.2. In summary, current datasets are sufficient to support the development of 3D object detection algorithms [11, 23–25] for basic classes (car, pedestrian, cyclist) in common scenes (sunny structured scenes). In the future, it is an urgent need to collect data in more challenging conditions, like snowy days, animal-on-the-road scenes.

5.2.4 Evaluation Metrics

In this section, we will discuss the metrics for 3D object detection evaluation. The evaluation metrics mainly contain two aspects: the precision metrics for localization and bounding box size evaluation (average precision), as well as the rotation similarity metrics for orientation evaluation (average orientation similarity).

5.2.4.1 Average Precision

For detection and classification tasks that output a probability y_i of the sample x_i belonging to positive samples in the given ground truth, recall rate is defined as the ratio of all positive samples ranked above a certain threshold t to the total number of positive samples:

$$r(t) = P(y_i \geq t | x_i \in C), \tag{5.1}$$

where C consists of all positive samples that exist in the ground truth. Precision is calculated as the ratio of all samples above the threshold t in the given ground truth to the total number of samples predicted above that threshold.

$$p(t) = P(x_i \in C | y_i \geq t). \tag{5.2}$$

By setting t, we can get a recall and precision rate accordingly, so that a precision-recall curve can be drawn by setting different $t \in [0, 1]$. The average precision (AP) is the area of the region below the precision-recall curve, which can be approximately computed as

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p_{interp}(r), \tag{5.3}$$

where r denotes the recall rate, and $p_{interp}(r) = \max_{\hat{r}:\hat{r} \geq r} p(\hat{r})$ is an interpolated alternative of the precision for a recall r.

At the very beginning, researchers adopted a similar AP in image detection, called AP_{2D}. For AP_{2D}, samples were considered as true positives, if the intersection over unions (IoUs) of the estimated and ground-truth 2D boxes on the image plane exceeds a specific value. It was recommended to combine AP_{2D} and AOS (which will be detailed in the next section) to evaluate the 3D object detection results in both locations, box size, and orientation. However, the effects of localization and bounding box size estimation cannot be decoupled by these two metrics [26].

AP_{2D} also suffers from the foreshortening effects that two objects with different sizes may project to the same 2D bounding box on the image plane. To solve this problem, some researchers computed the IoU between the bird's-eye-view (BEV) projection of the estimated and the ground-truth 3D boxes on the BEV image plane and proposed AP_{BEV}. The drawback of AP_{BEV} is that height error is not considered in the evaluation. Therefore, researchers computed AP_{3D} with the IoU computed between the estimated and ground-truth 3D boxes in the 3D space. Specifically, the IoU thresholds are 0.5 for pedestrians and cyclists and 0.7 for cars in the KITTI 3D object detection Benchmark [20].

In the nuScenes benchmark [12], the AP is specially computed according to the 2D center distance on the ground plane instead of IoUs. They claimed it decoupled the metric AP from object size and orientation and suited well to the evaluation of small objects like pedestrians. To decouple all the locations, scale, and orientation evaluations, they proposed average translation error, average scale error, and average orientation error [12]. They used a linear combination of AP and the average errors to indicate the thorough performance of the 3D detectors. They called this metric as called nuScenes detection score (*NDS*). In contrast, Waymo benchmark [21] proposed to weigh the true positives by their heading accuracy [21]. The weight scalar is defined as $\min(|\tilde{\theta} - \theta|, 2\pi - |\tilde{\theta} - \theta|)/\pi$, and this weighted average precision is called *APH*.

5.2.4.2 Average Orientation Similarity (AOS)

Similar to the definition of AP, average orientation similarity (AOS) is defined as

$$AOS = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \max_{\hat{r}: \hat{r} \geq r} s(\hat{r}), \tag{5.4}$$

where the orientation similarity $s \in [0, 1]$ at recall r is a normalized variant of the cosine similarity defined as

$$s(r) = \frac{1}{|D(r)|} \sum_{i \in D(r)} \frac{1 + \cos(\Delta_\theta^{(i)})}{2} \delta_i \tag{5.5}$$

where $D(r)$ refers to all object detections at a given recall rate r, while $\Delta_\theta^{(i)}$ represents the difference in angle between the estimated and ground truth orientation of detection i [20]. In order to penalize the cases where multiple detections correspond to a single object, δ_i is set to 1 if detection i overlaps with a ground truth bounding box by at least 50% in 2D, indicating that it has been assigned to that object. If the detection does not overlap sufficiently with any ground truth bounding boxes, δ_i is set to 0 to indicate that it has not been assigned. Through a similar exploration as AP, some researchers modified AOS by replacing the IoU computation method. Instead of computing IoUs on the image plane, they computed it in 3D space, resulting in a metric the metric called average heading similarity (AHS) [11].

5.2.4.3 Summary

Table 5.3 shows that $AP_{2D} + AOS$ and AP_{3D} are good choices for generally evaluating a 3D object detector. In practice, AP_{3D} is a harsher metric than $AP_{2D} + AOS$, since the methods perform lower than 10% in AP_{3D} even if they get more than 90% in $AP_{2D} + AOS$ [9, 10, 27].

Even though AP_{3D} can be used as a good metric for evaluating the general performance of a 3D object detector, it fails to decouple the effects of localization, scale, and orientation estimation. As a result, it is hard for researchers to analyze and trace back

Table 5.3 Comparison among different metrics for 3D object detection

Metrics	Loc. (x,y)	Loc. (z)	Scale (x,y)	Scale (z)	Orientation
AP_{2D}	Yes	Yes	Yes	Yes	No
$AP_{2D} + AOS$	Yes	Yes	Yes	Yes	Yes
AP_{BEV}	Yes	No	Yes	No	Yes
AP_{3D}	Yes	Yes	Yes	Yes	Yes

the reasons for the bad performance of their detectors. Promising solutions include the metrics such as *NDS* and the set of average errors we mentioned in Sect. 5.2.4.1. Both of them not only consider general performance but also decoupled performance in each aspect.

5.3 Camera-Based Methods

Images captured by cameras can be used for 3D object detection due to their rich textures, but camera-based object detection suffers from the lack of depth information which is greatly helpful for accurate spatial information estimation, including locations, scales, and orientations of 3D boxes. Therefore, one major riddle for camera-based methods is: **how to project 2D input to 3D and learn depth information from object supervision?**

We observe that the final feature representation significantly affects the architecture, training process, and computation complexity in camera-based methods. Depending on the output feature representation, we divide camera-based methods into (1) result-lifting and (2) feature-lifting methods.

5.3.1 Result-Lifting Methods

Result-lifting methods refer to models that are based on 2D features. These methods first make predictions on the image plane using the 2D features, then estimate depth information, and finally lift the 2D detections into the 3D space. A typical example is shown in Fig. 5.5. Depending on the method of obtaining depth information, we further group these methods into three categories: direct depth prediction, depth from keypoints, and depth from priors.

Direct Depth Prediction: The network structure for direct depth prediction is concise and straightforward. Thus, there exist numerous works have emerged to follow in this direction. In MonoPair [29], object locations and spatial constraints between matched object pairs are computed together using uncertainty-aware spatial constraints to optimize the predicted 3D locations of the objects. It utilizes the pair-wise spatial constraint to model the relationship between two neighboring objects. During the prediction, it imposes aleatoric uncertainty on the detector, formulates the predicted 3D locations and their pair-wise spatial information into a nonlinear least squares problem, and finally predicts the 3D results. On the other hand, Transformer [30] has shown great potential in computer vision, and researchers recently tried to apply it to 3D detection. However, it is challenging to apply the learned object query [31] to fully represent the object property in the image-based 3D detection task. The reason is that the size of objects at far and close distance fluctuates significantly due to the perspective projection. As a solution, MonoDTR [32] proposed the first transformer-based fusion module, which globally merges the image and depth

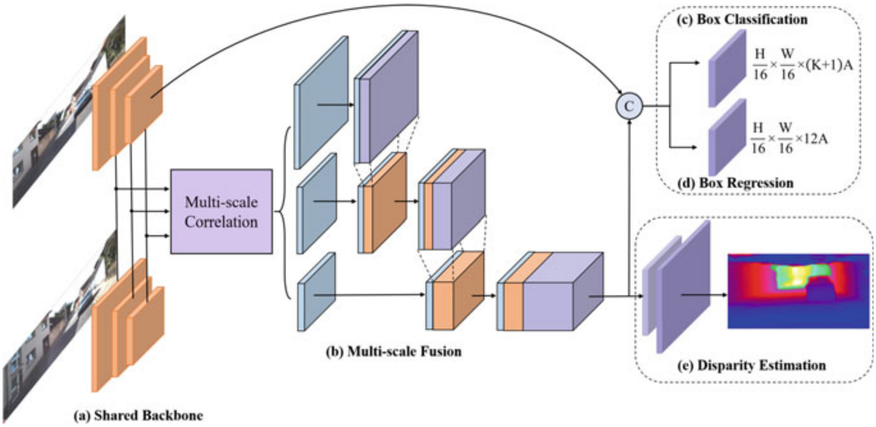


Fig. 5.5 YoloStereo3D architecture [28]. Backbones and multi-scale features both process features in 2D. 3D results are assembled from dense 2D predictions

information. The model employs a depth-aware feature enhancement (DFE) module that seeks to depth-aware features through implicit auxiliary supervision. These depth-aware features nicely assist the monocular 3D object detector, while preventing introducing high computational cost and inaccurate depth priors from using the off-the-shelf depth estimator.

In monocular 3D object detection tasks, getting an accurate depth estimate is always an ill-posed problem. To address the issue of inaccurate depth prior, the Depth-conditioned Dynamic Message Propagation (DDMP [33]) network presents a new method that utilizes a graph-based approach. This method utilizes contextual nodes in the image context, and predicts hybrid filter weights as well as affinities using aligned multi-scale depth features to propagate messages. In contrast, DD3D [34] proposes to use a wider range of data sources to address the problem of depth inaccuracy. The pseudo-lidar methods[35], have good scalability, while end-to-end detection networks [29] with simple structures and better generalization benefit less from large-scale depth data. DD3D aims to get the best of both worlds. It introduces a new 3D detection architecture that is fully convolutional and single-stage and effectively uses monocular depth estimation for pre-training. It leverages a large amount of unlabeled raw data, and its depth perception module benefits from pre-training on large labeled 2D detection datasets.

Depth from Keypoints: Keypoint-based methods usually make use of geometric constraints in the image for prediction. Stereo R-CNN [17] adopts the region-based framework and introduces another image to provide the spatial cues. Specifically, it simultaneously detects and associates objects from stereo pairs, uses the Stereo RPN module to produce left and right RoI proposals, then concatenates left-right RoI features to classify object categories and regress accurate 2D stereo boxes. Sparse constraints for 3D box estimation are developed using the results and keypoints. 3D box predictions are then developed based on projection relations between 3D box corners and 2D left-right boxes and keypoints. KM3D [36] predicts the projection of 9 keypoints for each object in the image using the center-net baseline, and it predicts depth purely from the minimization of the projection error between keypoint predictions and instance predictions. MonoFlex [37] focuses on using only the 3D height and the visual height of the object to predict depth, fully utilizing the knowledge of autonomous driving scenes.

Depth from Priors: As mentioned before, monocular methods highly dependent on depth estimation due to the lack of depth perception. Researchers have attempted to improve the accuracy of depth prediction by mining the information implied in images through various methods. Geometry projection is a technique that can be used to detect 3D objects from 2D images. This method estimates depth by using the object's height, which is a mathematical prior that can be incorporated into a deep learning model. M3D-RPN [38], GAC [39], and YoloStereo3D [28] collect depth priors from the training dataset to help define each anchor. However, the process of projection can result in the amplification of errors, causing the estimated height to be inaccurately reflected at the output depth. This amplification of error can lead to challenges in controlling depth inferences and can negatively impact training efficiency. To solve this problem, GUPNet [40] introduces a geometric uncertainty projection approach. It proposes a GUP module to obtain the geometry-guided uncertainty of the inferred depth, and adopts uncertainty theory to model the projection process under the probability framework.

5.3.2 Feature-Lifting Methods

Feature-lifting methods transform intermediate features into world coordinates and produce final predictions in BEV or 3D volumes. We further categorize these methods into pseudo-lidar methods and feature mapping methods.

Pseudo-Lidar Methods: Pseudo LiDAR methods, stemmed from [35], correspond to a framework that explicitly produces point cloud from images and applies LiDAR-based 3D detection methods to produce the detection output. Pseudo-LiDAR++ [41] uses sparse LiDAR signals to optimize the point cloud from the depth estimation network locally. Refined-MPL [42] demonstrates that point cloud generation quality does matter in 3D object detection performance. We point out that the above methods usually contain two separate modules trained independently. As a result, depth prediction networks cannot receive supervision signals from object-level supervision. To solve this problem, Pseudo-LiDAR E2E [43] develops a differentiable voxelization module and makes the entire network pipeline end-to-end trainable. Such methods are generally robust to camera parameter changes because they decouple depth prediction from 3D detection. However, the explicit point cloud prediction ignores the uncertainty and errors in depth prediction, and only the expectation of the predicted depth distribution is mapped to the downstream 3D detection part, making the pipeline sub-optimal. Further, CG-Stereo[44] incorporates uncertainty and confidence maps into 3D detection and achieves better detection accuracy.

Feature-Mapping Methods: Feature-mapping methods learn a mapping between 2D image features and 3D volume features. They learn a depth distribution for each image pixel and project the entire 3D volume in camera coordinates to the world coordinates [45–49]. Lift-Splat-Shoot [45] shows that mapping 2D features into 3D makes it easier to form a unified representation for surround-view cameras and other sensors. CaDNN [49] learns a depth distribution for each image pixel and lifts the features as a camera frustum. The frustum in camera coordinates is sampled into a feature volume in the world coordinates. LIGAStereo [48] benefits from sharing a similar feature space with LiDAR detection and applies LiDAR distillation to the 3D volume produced by the stereo network. BEVFusion[50] introduces a BEV Pooling module specifically designed for a parallel acceleration of the sampling process under the condition that the cameras have stable parameters. A common properties of these methods is that their feature mapping is driven by geometric constraints.

In contrast, some recent works adopt semantic-driven feature mapping, where semantics in the images will also affect the mapping between two features [51–53]. They achieve this with the cross-attention mechanism. PETR[51] directly uses transformers with carefully-designed positional embedding to achieve 2D-3D mapping. Furthermore, BEVFormer[52] applies deformable attention modules [53], where the reference points are geometrically determined, while the offsets are semantically determined, and takes advantage of both streams of methods.

5.3.3 Summary

Table 5.4 shows the state-of-the-art results for camera-based methods in the KITTI dataset. Both result-lifting methods and feature-lifting methods remain popular in academics and the industry. Recent result-lifting methods are fast and easy to deploy because of simple operators, and many of them are robust to changes in camera parameters or scenes. Feature-lifting methods are also popular because it is found that 3D features are easier to merge with multi-camera features, LiDAR features, and temporal features. The future developments of camera-based methods of both types are drawing growing interest for computer vision researchers and autonomous driving engineers.

5.4 LiDAR-Based Methods

The geometry information captured by LiDARs can be used for perception, and accurate spatial information is helpful for precise 3D object location. However, LiDARs suffer from sparsity and CNN-incompatible representations. Researchers who intend to percept key objects from LiDAR scans have to deal with the problem: **How to extract features from point clouds?**

Table 5.4 Comparison among the performances of camera-based methods based on the KITTI benchmark (test set)

Modality	Category	Implementation	Car (AP_{3D})			Time(s)
			Easy	Mod	Hard	
Stereo Cameras	Result-Lifting	3DOP [9]	6.6	5.1	4.1	3
		Stereo RCNN [17]	47.6	30.2	23.7	0.3
		YoloStereo3D [28]	65.8	40.7	30.0	0.08
	Feature-Lifting	Pseudo-LiDAR [35]	54.5	34.1	28.3	0.4
		Pseudo-LiDAR++ [41]	61.1	42.4	37.0	0.4
		Pseudo-LiDAR E2E [41]	64.8	43.9	39.0	0.4
		DSGN [47]	73.5	52.2	45.1	0.67
		LIGAStereo [48]	81.4	64.7	57.2	0.4
Mono Camera	Result-Lifting	Deep3D Box [27]	5.9	4.1	3.8	–
		MonoPair [29]	13.0	10.0	8.7	0.06
		M3DRPN [38]	14.8	9.7	7.4	0.16
		KM3D [36]	16.7	11.5	9.9	0.04
		DDMP [33]	19.7	12.8	9.8	0.18
		GAC [39]	21.6	13.2	9.9	0.05
		MonoDTR [32]	22.0	15.4	12.7	0.04
		GUPNet [40]	22.3	15.0	13.1	–
		DD3D [34]	23.2	16.3	14.2	–
	Feature-Lifting	Pseudo-LiDAR[35] [10]	10.8	7.5	6.1	0.1
		AM3D [54]	16.5	10.7	9.5	0.4
		RefinedMPL [42]	18.1	11.1	8.9	0.15
		CaDDN[49]	19.2	13.4	11.5	0.64

5.4.1 Quantization+CNN-Based Methods

One solution is to convert the input point clouds into convolutional-friendly representations and then apply CNNs to extract features. The conversion process is called quantization. Common quantization processes include projecting the point cloud into bird's-eye-view images [22, 55–57] or voxelizing the point cloud into a grid [23, 58–63]. The 3D spatial information is encoded into the grids by hand-crafted features, like point density, distance, occupancy, etc. After quantization, high-level features

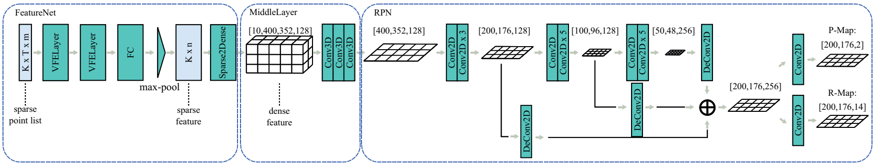


Fig. 5.6 VoxelNet architecture [23]. It consists of three main parts: FeatureNet (point-wise and voxel-wise feature transformation), MiddleLayer (3D dense convolution), and RPN (2D dense convolution)

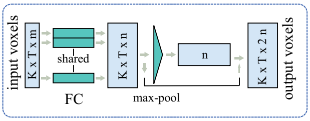


Fig. 5.7 VFELayer of VoxelNet [23]. The input is K voxels, within which there are T points with m feature channels. An MLP transforms the inputs into the feature space, and a max-pooling layer aggregates point-wise features into a global feature

can be extracted from the grid with CNNs and further used for classification and regression tasks.

However, the hand-crafted features in quantization are always sub-optimal to the visual tasks [56, 64]. Zhou et. al. proposed an end-to-end network for 3D object detection, called VoxelNet, where the voxel-wise features were learned from raw point clouds instead of hand-crafted by researchers [23]. The network architecture is shown in Fig. 5.6. To learn the voxel-wise features from point clouds, they first grouped the points according to their corresponding position in the grid. Then they applied a bunch of novel voxel feature encoding (VFE) layers to extract point-wise features from the points inside each voxel. The VFELayer contains multi-layer perceptrons (MLPs) and a max-pooling layer sequentially, as shown in Fig. 5.7. Finally, the point-wise features inside each voxel were aggregated into a voxel-wise feature with a max-pooling layer.

The MiddleLayer limited the running-time performance of VoxelNet and accounted for 50% computation of the whole network due to the 3D dense convolution operation. The grid and dense convolution in 3D are inefficient from both memory and computation aspects. Due to the sparsity of point clouds, the non-empty voxels only occupy $\leq$1% of the grid.[2] For dense 3D convolution, each voxel will incur addition-multiplication operations no matter if it is zero or not. Yan et al. adopted the spatially sparse convolution [65] to deal with the sparsity of point clouds [59]. As a result, the running-time performance of VoxelNet was improved by 4× (230 ms to 50 ms per frame).

One benefit of Quantization+CNN methods is that the success of vision tasks can be easily extended to LiDAR-based 3D object detection. Yin et al. claims

[2] If a point cloud is partitioned into a [10,400,352] dense grid, only around 5300 voxels are non-empty.

previous anchor-based detection unnecessarily increases the computational burden and tends to induce a huge number of false positives [60]. They extended the CenterNet [66, 67] from image-based 2D detection to LiDAR-based detection and proposed a center-based two-stage framework for 3D object detection. Compared to anchor-based parameterization, the center-based method reduces the search space since centers do not have an orientation dimension. Adopting auxiliary tasks to improve performance has achieved great success in vision tasks [1, 68, 69]. He et al. adopted the same idea and proposed to add auxiliary networks for estimating foreground points and centers to learn structure information [61]. Similarly, Ye et al. demonstrated that multi-task learning of segmentation and detection improves the overall performance of all these tasks [63].

Occlusion is a problem hindering both image- and LiDAR-based detection tasks. Xu et al. [70] explored the occlusion problem in 3D object detection and discussed three causes of occlusion and signal miss in LiDAR point clouds: external occlusion, signal miss, and self-occlusion. By considering these occlusion causes, they proposed BTCNet, which first estimates the occupancy of complete object shapes in the region affected by occlusion and signal miss, then applies a two-stage detector to integrate the occupancy information in generating proposals and finally refine the results. The limitation of their work is that it requires an additional occupancy map estimation network, and the computational effectiveness requires improvement.

The Quantization+CNN methods share two limitations. Firstly, information loss is induced in the stage of quantization. The accuracy highly depends on the quantization resolution. As resolution increases, information loss decreases, and positive samples are more distinguishable from negative ones. However, the improvement in accuracy is obtained at the cost of real-time performance. Secondly, the computational cost is not polynomial to the input size (i.e., the number of points in the point cloud). It is polynomial to the grid size determined by the size of RoI and the resolution.

5.4.2 Point-Based Methods

Quantization induces information loss, and results in unnecessarily voluminous data, and causes high computational costs when applied to large-scale scenes. To avoid quantization artifacts, some researchers designed special network structures to learn features from the input point clouds directly [24, 71–74].

Qi et al. [71] proposed PointNet for end-to-end learning representations directly from point clouds. They endowed PointNet with order invariance by applying a symmetric function. A symmetric function is a function that takes n vectors as input and produces a new vector that is invariant to the order of the input. Examples of symmetric functions include the $+$ and $\times$ operators. On account of the transformation invariance, they proposed T-Net, which was intended to align all input point clouds to a canonical space. A T-Net consists of MLPs and a max-pooling layer at the end and regresses the affine transformation 3×3 matrix. To force the estimated vector by

T-Net to coincide with an affine transformation matrix, they added a regularization term in their loss function:

$$L_{reg} = ||I - AA^T||^2, \tag{5.6}$$

where A is the matrix estimated by the T-Net, and I denotes the identity matrix.

Exploiting local structures has proven to be important for the success of convolutional architecture in visual tasks. PointNet can extract point-wise and global features, but the local features are hard to capture. In their following work [72], they proposed set-abstraction blocks based on PointNet to capture the local features of the point clouds, where each set-abstraction block sequentially samples center points, groups neighbor points and extracts features. Wang et al. [74] extended the spirit of graph neural networks to 3D point clouds to learn the local features from the point clouds which was based on the graph dynamically built with k-nearest neighbors (KNN). Li et al. [73] proposed a more efficient method to catch up with the local feature of point clouds. Instead of grouping the points with ball query methods and KNN methods like [72, 74], they learned the spatial distribution of the input point clouds with the self-organizing map in an unsupervised manner, so that the training time was largely reduced.

Shi et al. [75] extended PointNet architecture to 3D object detection and proposed PointRCNN. They adopted the PointNet++ network to implement foreground segmentation and point-wise bounding box regression. The estimated bounding boxes of the predicted foreground points were filtered out as 3D proposals, and the points inside each proposal were further used to refine the 3D bounding box. PointRCNN achieves better accuracy on the KITTI benchmark than VoxelNet on car class, which proves that the point-cloud-based learning representations can solve the information loss induced by quantization. Further, Qi et al. [76] adopted both PointNet architecture and hough voting to frame an end-to-end 3D object detection pipeline, called VoteNet. It estimates vote centers with the PointNet++ network and then clusters them into K groups. The K groups vote centers are further used for 3D bounding boxes regression. The above Point-based methods all consist of two stages, which have common limitations due to their slow inference time. Recently, Zhang et al. [77] proposed a one-stage point-based 3D object detector, and it runs at more than 80 FPS on the KITTI dataset. They claimed the bottleneck of point-based detectors is sampling resolution and proposed two learning-based instance-aware down-sampling strategies. With these two down-sampling strategies, their proposed approach largely improves the real-time performance of LiDAR-based object detectors with state-of-the-art accuracy.

5.4.3 Point-Voxel-Based Methods

To complement the shortcomings of Quantization+CNN and Point-based methods, some works combine them to balance accuracy and speed [78–80]. Yang et al. [79]

proposed a two-stage voxel-point-based 3D object detector. They discussed the problem where the classification score does not match the precision of location estimation, and proposed estimating IoU as an alternative to alleviate this problem. Another representative Point-Voxel-based method is HVPR [80], which is a one-stage detector and uses a memory module to augment point-based features, maintaining the efficiency of a single-stage method.

5.4.4 Summary

The performance of the methods mentioned above is compared in Table 5.5. The results are evaluated on the KITTI test set. It demonstrates that the 3D detection of small objects, like pedestrians and cyclists, is still an open problem. It is more challenging than vehicle detection because only few points are detected on these small objects, especially when they are far away. One possible solution is to utilize the help of other sensors like multi-LiDAR setups or LiDAR-camera setups. Besides, the research around Point-based 3D object detection is insufficient compared to quantization+CNN methods. It is a promising direction due to its excellent improvement in accuracy and there is substantial room for speed improvement.

5.5 RADAR-Based Methods

Similar to LiDARs, RADARs also provide spatial information on their perception fields. RADARs have already been widely used in automotive applications, such as collision avoidance and blind-spot detection, due to their robust performance in various weather and lighting conditions.

However, research of RADAR-based methods in 3D object detection is barely available. The reasons are: (1) the RADAR data is quite sparse due to the low resolution; (2) the automotive RADAR becomes unreliable at 10–15m when working in real-world scenes due to reflections from other objects; (3) the RADAR data is commonly in the 2D form, and height information is not available for 3D bounding box estimation. Due to the limited research work, we generally classified existing RADAR-based methods into *Spectrum-Image as Input* and *Point-Cloud as Input* methods.

Spectrum-Image as Input: The raw radar data contains the information of each location along a specific angle, which can be represented as a spectrum image. It is a dense representation similar to RGB images but contains spatial and reflection information. In the last decade, Bartsch et al. proposed a knowledge-based recognition system to detect pedestrians from the RADAR spectrum images [82]. They first segmented the spectrum images and handcrafted features from the patches. Then they classified the patches with a predefined empirical deterministic function. They claimed that it is hard to recognize pedestrians from solely RADAR data, especially

Table 5.5 Comparison between the performances of LiDAR-based methods based on the KITTI benchmark (test set)

Category	Method	Car(AP_{3D})			Ped(AP_{3D})			Cyc(AP_{3D})			Time(s)
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	
Q+CNN	PointPillars [57]	79.0	74.9	68.3	52.0	43.5	41.4	75.7	59.0	52.9	0.15
Q+CNN	VoxelNet [23]	77.5	65.1	57.7	39.5	33.7	31.5	61.2	48.4	44.4	0.2
Q+CNN	SECOND [59]	83.1	73.6	66.2	51.0	42.5	37.2	70.5	53.8	46.9	0.05
Q+CNN	SA-SSD [61]	88.8	79.8	74.2	–	–	–	–	–	–	0.4
Q+CNN	CIA-SSD [62]	89.6	80.3	72.9	–	–	–	–	-	–	0.031
Q+CNN	BTCNet [70]	90.6	82.9	78.1	–	-	-	82.8	68.7	61.8	–
Q+CNN	VoxelRCNN [81]	90.9	81.6	77.1	–	—	–	–	–	–	0.04
Point	PointRCNN [75]	85.9	75.7	68.3	49.4	41.7	38.6	73.9	59.6	53.5	0.1
Point	IA-SSD [77]	88.3	80.1	75.0	46.5	39.0	35.6	78.4	61.9	55.7	0.013
Voxel-Point	STD [79]	88.0	79.7	75.1	53.3	42.5	38.4	78.7	61.6	55.3	0.08
Voxel-Point	HVPR [80]	86.4	77.9	73.0	53.5	44.0	40.6	–	–	–	0.028
Voxel-Point	PVRCNN [78]	90.3	81.4	76.8	52.2	43.3	40.3	78.6	63.7	57.7	0.08

in partial-vision or occluded conditions. In recent years, deep learning models have been used as powerful feature extractors and classifiers for perceptual tasks. Kanil et al. first generated proposals from the spectrum images and classified them with a CNN model [83]. Their work can be further improved if it is solved with the help of R-CNN framework [2] or one-stage detection framework [3].

Point-Cloud as Input: As mentioned in Sect. 5.2.2.3, point clouds can be obtained from RADAR raw data with clustering preprocessing. Scheiner et al. [84] proposed a straightforward solution that separates the input RADAR point clouds into multiple clusters with DBSCAN and further classifies them with LSTMs. Schumann et al. [85] went further in this direction and extracted features from static and dynamic RADAR points with CNNs and memory-aware PointNets, respectively. Danzer et al. proposed to estimate rotated 2D bounding boxes (no height) from RADAR point clouds [86]. They treated the RADAR data similarly to the LiDAR's and adopted the PoineNet architectures [71, 72] to extract features and regress 2D bounding boxes. Similar to LiDAR-based object detection in Sect. 5.4, RADAR point clouds can be quantified into 2D images and CNN can be adopted to extract features and detect objects [85, 87]. In [88], they proposed a RADAR-based detection benchmark and compared YOLOv3 [4], PointPillars [57], PointNet++ [72] as well as their variances. Their comparison shows YOLOv3 and PointNet++ perform better than PointPillars variances.

Section Summary: Compared to camera-based and LiDAR-based object detection, RADAR-based object detection is underexplored due to its sparse data observation. In Sect. 5.6, we will introduce some RADAR-based object detectors which utilize RADAR data to improve LiDAR/camera-based detector performance.

5.6 Multi-sensor-fusion Methods

Multi-sensor fusion is a popular research direction in 3D object detection, which provides not only complementary but also redundancy in the face of sabotage, failures, adverse conditions and blind spots [19]. However, different sensors provide different data representations from which features are extracted with different models. Therefore, one important problem for multi-sensor-fusion methods is: **How to fuse multi-sensor data for perception?**

5.6.1 Feature Fusion

The feature vectors extracted from different types of sensors can be fused using concatenation or addition (Fig. 5.8 top). Chen et al. [89] proposed a two-stage network, called MV3D, to combine the camera and LiDAR data. With a Quantization+CNN pipeline, they quantized the point cloud into a BEV image and a front-view image, and extracted features with CNNs from both. In the first stage, they generated 3D proposals from the BEV image based on an RPN. In the second stage, they adopted multi-view ROI pooling to align the features from the BEV, front view, and RGB images. The features from different sensors were fused by concatenation and finally used to refine the proposal. A similar idea of fusing features on the image plane was also adopted by [11, 89] for camera-LiDAR fusion and adopted by [90] for camera-RADAR fusion.

In contrast to fusing pixel-wise features on the image plane, Sindaigi et al. [91] explored the performance of point-wise feature fusion in the 3D space and voxel-wise feature fusion in the grid space based on VoxelNet [23]. BEVFusion [50] uses two different pipelines to extract camera and LiDAR features, respectively. They introduce a BEV encoder to fuse the concatenated features. CenterFusion[92] proposed a frustum-based approach to complete representation fusion by projecting radar features onto the image plane. Objects in RADAR feature maps are associated by using information from the primary image bounding box. Then, they are projected into the image feature space and fused with the image feature. Decoder takes the extra velocity and depth information from the RADAR to generate prediction results.

5.6.2 Transformer Interaction Fusion

Vision transformer achieved good performance in 2D and 3D object detection in recent years. The architecture of transformer interaction fusion methods is shown in the middle row of Fig. 5.8. DeepFusion [93] uses the visual transformer to fuse features. PointPillars [57] is used as the feature extractor to obtain LiDAR features. The camera image serves as input to a 2D image feature extractor (such as ResNet) to obtain camera features. Then, the image feature is used as the key and value of the transformer decoder, and LiDAR feature is used to generate the query. The output of the decoder is concatenated with the LiDAR feature to complete feature fusion. Finally, it uses the voxel-based detection head to process the feature fusion to output the detection results.

LiDAR is hard to deal with object which is tiny and far away because of its low resolution compared to the camera. To address this problem, TransFusion [94] uses a dual transformer decoder. The first layer of the decoder generates initial prediction results by using the sparse 3D feature of point cloud queries. The query of the second layer fuses the initial result of the first layer to predict the final result on the image feature. Similar to TransFusion, DeepInteraction [95] introduces transformer encoders based on different modalities. In particular, the multi-modal predictive interaction layers take the image or the LiDAR representation as the input, the odd layer takes the camera representation, and even for LiDAR. Each layer boosts the perception strength of the corresponding modality.

A benefit of the unified representation is that there is no need to design a block for transforming feature space. MVDNet [96] can directly generate 2D proposals for both RADAR and LiDAR. The representations of the two modalities are then fused using the cross attention module. Transforming different modalities into a unified representation space is also a solution. UVTR [97] unifies feature representation in voxel space. They exploit image features to generate voxel-space representations of image features through simple convolutional layers. The modalities are then fused using knowledge transfer learning. Finally, transformer decoder is used to predict the result.

5.6.3 Cascade Pipeline

Cascade pipeline means that the results of one detector act as the inputs to the next detector, as shown in the bottom part of Fig. 5.8. Qi *et al.* [24] proposed Frustum-PointNet to fuse camera and LiDAR data for 3D object detection. They first generated 2D bounding boxes from only RGB images with an off-the-shelf image-based object detector. The results were projected into 3D space as frustums, the points inside which were used to refine 3D bounding boxes by PointNet architectures [71, 72] in the second stage. Some researches focus on camera-LiDAR fusion algorithms mainly using enhancement point cloud to do detection tasks using the image feature information. PointPainting and CenterPointV2 [60, 98] are classical examples of such algorithms. They all fuse the segmentation information to the LiDAR points. They use semantic segmentation to get objects. According to the segmentation results, the corresponding point clouds are classified, and a point cloud-based 3D object detection framework is applied to predict the results. For the latter, they use voxel-based method to process the LiDAR points. In the cascade pipeline, the features of different detectors can also be fused by concatenation or addition [99].

Even though these cascade methods achieve state-of-the-art performances on the KITTI benchmark [20], they suffer from the following limitations: 1) The accuracy is upper bounded by the early-level detectors. The false negatives will never be recovered if they are missed in the early stages. 2) If the early stage is an image-based detector, the detections will be vulnerable to appearance changes even if LiDAR data is included. 3) The long detection range of cameras is limited by LiDARs if there is no point detected by the LiDAR in the camera-detected region.

5.6.4 Section Summary

The methods mentioned above are evaluated on the KITTI test dataset and demonstrated in Table 5.6. One common drawback of the fusion methods above is that the fusion pipelines are fixed for a specific setup (1× camera + 1× LiDAR or 1× camera +1× RADAR). They can hardly be adapted to other settings. Due to various sensor setups on autonomous vehicles, one promising direction is to design a general multi-sensor fusion framework for 3D object detection so that it can flexibly scale up or down according to the sensor settings. It should be more accurate when more sensors are considered.

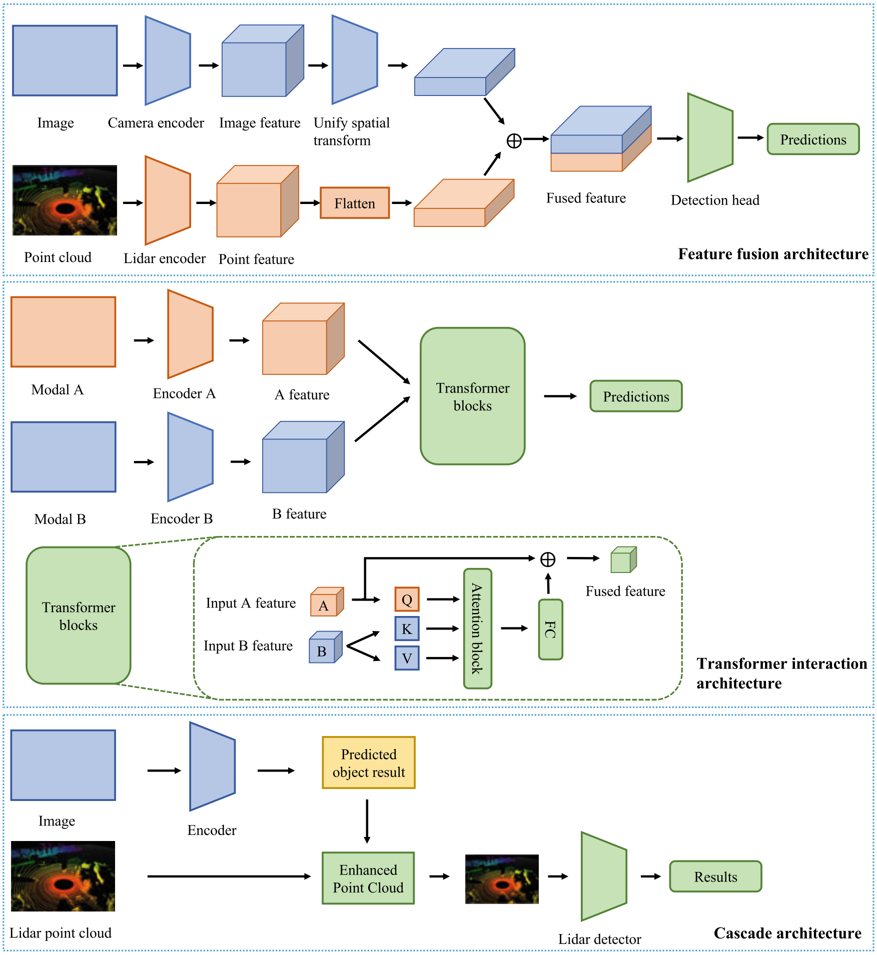


Fig. 5.8 Architectures of different multi-sensor-fusion-based methods. The top, middle, and bottom rows denote feature-fusion, transformer-based, and cascade architectures

5.7 Promising Directions

Besides continuously improving the accuracy and real-time performance, we have already introduced two open problems: RADAR-based 3D object detection methods in Sect. 5.5 and a flexible framework for multi-sensor-fusion 3D object detection in Sect. 5.6. In this section, we will introduce more interesting open problems in 3D object detection.

Table 5.6 Comparison between the performances of multi-sensor fusion methods. C, R, and L are short for camera, RADAR, and LiDAR, respectively

Solution	Implementation	Modality	KITTI ($AP_{Car,3D}$)			nuScenes		Time(s)
			Easy	Mod	Hard	NDS	mAP	
Feat.	MV3D [89]	C + L	71.2	62.6	56.5	–	–	0.4
Feat.	AVOD [11]	C + L	81.9	71.8	66.3	–	–	0.1
Feat.	MVX-Net [91]	C + L	83.2	72.7	65.2	–	-	0.06
Feat.	BEVFusion [50]	C + L	–	–	–	72.9	70.2	0.119
Feat.	CenterFusion [92]	R + L	–	–	-	44.9	32.6	–
Tran.	TransFusion [94]	C + L	–	–	–	71.7	68.9	0.266
Tran.	DeepInteraction [95]	C + L	–	–	–	76.3	75.6	0.204
Tran.	UVTR [97]	C + L	–	–	–	71.1	67.1	–
Cascade	F-PointNet [24]	C + L	81.2	70.4	62.2	–	–	0.2
Cascade	PointPainting [98]	C + L	92.4	88.1	83.3	46.4	33.9	0.016
Cascade	CenterPoint [60]	C + L	–	–	–	65.5	58.0	0.089

5.7.1 Extreme Conditions

The edge cases caused by extreme conditions are one of the major challenges in autonomous driving. As we claimed in Sect. 5.2.2, cameras and LiDARs are not robust to bad lighting conditions or extreme weather. The different appearance information in extreme weather and bad lighting conditions may cause a different data distribution from the training dataset for camera-based methods. Extreme weather, like rainy or snowy days, may worsen the quality of the point clouds of LiDAR scans.

One possible solution is to consider the data in extreme conditions when collecting the training data so that the data distribution will be similar when the 3D object detector is deployed in the real world. It can improve the performance of networks, but it is not the best option, since it is hard to enumerate all the extreme conditions in the real world. The unexpected conditions will put the passengers in a dangerous situation. Another solution is to adopt the multi-sensor setups since different sensor types have different failure modes during different conditions [12]. Especially, RADARs are famous for their robustness towards extreme conditions. Current multi-sensor fusion methods focus more on accuracy improvement than robustness improvement in extreme conditions, which needs more effort in the future. In addition, generative adversarial networks (GANs) [100], which have shown powerful ability, especially in domain adaptation in recent years, can be used to alleviate the difference between extreme conditions and normal conditions. For instance, the night-time image can be translated by GANs to daytime for 3D object detection. Similar works have gained attention in some visual tasks like place recognition [101, 102].

5.7.2 Uncertainty

Uncertainty is a natural part of any perception system's output. Knowing the confidence with which we can trust the 3D object detection output is crucial for decision making. To model the uncertainty of a perception output, a natural idea is to use the softmax output as the uncertainty. The higher output represents lower uncertainty.

Recently, some researchers modeled the perceptual uncertainty of a deep learning network output in a more detailed manner [103–105]. They classified the uncertainty into epistemic uncertainty which is caused by insufficient data, and aleatoric uncertainty which is caused by noisy annotation and model properties. In their work, they claimed that not only uncertainty was well modelled but also the accuracy of semantic segmentation networks was improved due to their consideration of uncertainty.

5.7.3 Summary

According to our discussion in this chapter, 3D object detection contains the following open problems:

How to improve the accuracy of 3D object detection, especially the objects in small sizes? For LiDAR- and RADAR-based methods, the cause of the bad accuracy when detecting small objects is due to the fact that there are few points detected by the perceptual sensors. One possible solution is to adopt more than one perceptual sensor, which provides ont only redundancy but also robustness. It could be the fusion of multiple LiDARs or RADARs, which naturally improve the number of points detected on the small objects. Besides, it could also be the fusion of different types of sensors, as we mentioned before, which also provides complimentary information in extreme cases.

How to implement 3D object detection based on RADAR data? As we mentioned in Sect. 5.5, current research of RADAR-related 3D object detection can be classified as point-cloud-based methods and spectral-image-based methods. However, all of them cannot estimate the properties of objects in the z-axis. It is limited by the sensor property, for instance, most of the current RADARs are 2D sensors. In the future, RADAR systems should be improved to provide 3D information as well.

How to design a flexible framework for multi-sensor fusion? As we mentioned in Sect. 5.6, current multi-sensor fusion methods are fixed and pre-designed for a specific sensor setup. The generalization to another setup is costly. A flexible framework of multi-sensor fusion 3D object detection is a promising research direction. One possible solution could be that all the sensors or groups of sensors can independently detect objects, and their results are fused by optimizing an objective function. More sensors provide more constraints and hence the results should be more accurate.

How to measure the robustness of 3D object detectors against extreme conditions? How to improve their robustness? A direct solution is to measure the difference in performance under common conditions and extreme conditions, like

$AP_{extreme}/AP_{common}$, where $AP_{extreme}$ and AP_{common} denote the average precision in the extreme and the common conditions. When the number of evaluation data is large enough, such a metric can measure the robustness of the algorithm in extreme conditions. One of the significant reasons for the bad performance under extreme conditions is sensor limitation. Therefore, the multi-sensor fusion methods could be a promising solution due to the availability of complementary information under different conditions.

How to model the uncertainty of 3D object detection? How to utilize the uncertain samples to enhance the performance of the detectors? As we mentioned in Sect. 5.7.2, the epistemic and aleatoric of [103, 104] can be extended to 3D object detection for measuring the uncertainty. One possible way to utilize the uncertain samples could include providing more weights to the uncertain samples in the training phase by adding a dynamic term in the loss function as demonstrated in [64, 106].

5.8 Conclusion

In this chapter, we reviewed the state-of-the-art 3D object detection methods in autonomous driving. We introduced the properties of three perceptual sensors in autonomous driving: cameras, LiDARs and RADARs, and analyzed their pros and cons for 3D object detection.

We reviewed the state-of-the-art 3D object detection methods in autonomous driving from the aspects of loss functions and architectures, and categorized them into camera-based, LiDAR-based, RADAR-based, and multi-sensor fusion methods. We introduced the problem definition, assumption, and evaluation metrics of 3D object detection in autonomous driving. The representative datasets were discussed, and their limitations were analyzed for future research. We pointed out the main problems and the existing solutions for each method. By analyzing the limitations of current solutions, we proposed some promising research directions and open problems. Quantitative results from the KITTI benchmark showed that 3D object detection is worth more effort, especially in small-size objects. Our analysis around autonomous driving system requirements showed that the robustness against extreme conditions, flexible multi-sensor fusion framework, and uncertainties in a perceptual system were all open problems in this field.

References

1. He K et al (2017) Mask R-CNN. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp 2961–2969
2. Girshick R (2015) Fast R-CNN. In: IEEE international conference on computer vision (ICCV), pp 1440–1448
3. Liu W et al (2016) SSD: single shot multibox detector. In: European conference on computer vision (ECCV). Springer, pp 21–37

4. Redmon J et al (2018) YOLOv3: an incremental improvement. Computing research repository (CoRR). https://arxiv.org/abs/1804.02767
5. Zhang C, et al (2018) Robust LIDAR localization for autonomous driving in rain. In: IEEE/RSJ international conference on intelligent robots and systems (IROS), pp 3409–3415
6. Arnold E et al (2019) A survey on 3D object detection methods for autonomous driving applications. IEEE Trans Intell Transp Syst (TITS) 3782–3795
7. Guo Y et al (2020) Deep learning for 3D point clouds: a survey. IEEE Trans Pattern Anal Mach Intell (TPAMI) 43(12):4338–4364
8. Alaba SY et al (2022) A survey on deep-learning-based LiDAR 3D object detection for autonomous driving. Sensors 22(24):9577. https://www.mdpi.com/1424-8220/22/24/9577
9. Chen X et al (2018) 3D object proposals using stereo imagery for accurate object class detection. IEEE Trans Pattern Anal Mach Intell (TPAMI) 40(5):1259–1272
10. Xiaozhi C et al (2016) Monocular 3D object detection for autonomous driving. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 2147–2156
11. Ku J et al (2018) Joint 3D proposal generation and object detection from view aggregation. In: IEEE/RSJ international conference on intelligent robots and systems (IROS), pp 1–8
12. Caesar H et al (2020) nuscenes: a multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 11 621–11 631
13. Fan R et al (2017) Real-time implementation of stereo vision based on optimised normalised cross-correlation and propagated search range on a GPU. In: IEEE international conference on imaging systems and techniques (IST), pp 1–6
14. Chang J-R et al (2018) Pyramid stereo matching network. In: IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 5410–5418
15. Mayer N et al (2016) A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 4040–4048
16. Xiang Y et al (2015) Data-driven 3D voxel patterns for object category recognition. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 1903–1911
17. Li P et al (2019) Stereo r-CNN based 3d object detection for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 7644–7652
18. Geronimo D et al (2010) Survey of pedestrian detection for advanced driver assistance systems. IEEE Trans Pattern Anal Mach Intell (TPAMI) 32(7):1239–1258
19. Kim J et al (2018) Robust camera lidar sensor fusion via deep gated information fusion network. In: IEEE intelligent vehicles symposium (IV), pp 1620–1625
20. Geiger A et al (2012) Are we ready for autonomous driving? The KITTI vision benchmark suite. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 3354–3361
21. Sun P et al (2020) Scalability in perception for autonomous driving: waymo open dataset. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 2446–2454
22. Simon M et al (2018) Complex-YOLO: an Euler-region-proposal for real-time 3D object detection on point clouds. In: European conference on computer vision (ECCV). Springer, pp 197–200
23. Zhou Y et al (2018) VoxelNet: end-to-end learning for point cloud based 3D object detection. In: IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 4490–4499
24. Qi CR et al (2018) Frustum point nets for 3D object detection from RGB-D data. In: IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 918–927
25. Liang M et al (2018) Deep continuous fusion for multi-sensor 3D object detection. In: Proceedings of the European conference on computer vision (ECCV), pp 641–656
26. Redondo-Cabrera CO (2016) Pose estimation errors, the ultimate diagnosis. In: European conference on computer vision (ECCV). Springer, pp 118–134
27. Mousavian A et al (2017) 3D bounding box estimation using deep learning and geometry. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 5632–5640

28. Liu Y et al (2021) YOLOStereo3D: a step back to 2D for efficient stereo 3D detection. In: International conference on robotics and automation (ICRA). IEEE, pp 13 018–13 024
29. Chen Y et al (2020) MonoPair: monocular 3D object detection using pairwise spatial relationships. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 12 093–12 102
30. Vaswani A et al (2017) Attention is all you need. In: Advances in neural information processing systems (NeurIPS), vol 30
31. Carion N et al (2020) End-to-end object detection with transformers. In: European conference on computer vision (ECCV). Springer, pp 213–229
32. Huang K-C et al (2022) MonoDTR: monocular 3D object detection with depth-aware transformer. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 4012–4021
33. Wang L et al (2021) Depth-conditioned dynamic message propagation for monocular 3D object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 454–463
34. Park D et al (2021) Is pseudo-lidar needed for monocular 3D object detection? In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 3142–3152
35. Wang Y et al (2018) Pseudo-LiDAR from visual depth estimation: bridging the gap in 3D object detection for autonomous driving. Computing research repository (CoRR), vol abs/1812.07179. https://arxiv.org/abs/1812.07179
36. Li P et al (2021) Monocular 3D detection with geometric constraints embedding and semi-supervised training. IEEE Robot Autom Lett (RAL) 6(3):5565–5572
37. Zhang Y et al (2021) Objects are different: flexible monocular 3D object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 3289–3298
38. Brazil G et al (2019) M3D-RPN: monocular 3D region proposal network for object detection. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp 9287–9296
39. Liu Y et al (2021) Ground-aware monocular 3D object detection for autonomous driving. IEEE Robot Autom Lett (RAL), pp 919–926
40. Lu Y et al (2021) Geometry uncertainty projection network for monocular 3D object detection. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 3111–3121
41. You Y et al (2019) Pseudo-LiDAR++: accurate depth for 3D object detection in autonomous driving. Computing research repository (CoRR). https://arxiv.org/abs/1906.06310
42. Vianney JMU et al (2019) RefinedMPL: refined monocular PseudoLiDAR for 3D object detection in autonomous driving. Computing research repository (CoRR). https://arxiv.org/abs/1911.09712
43. Qian R et al (2020) End-to-end pseudo-LiDAR for image-based 3D object detection. In: Conference on computer vision and pattern recognition (CVPR), pp 5881–5890
44. Li C et al (2020) Confidence guided stereo 3D object detection with split depth estimation. In: IEEE/RSJ international conference on intelligent robots and systems (IROS), pp 5776–5783
45. Philion J et al (2020) Lift, splat, shoot: encoding images from arbitrary camera rigs by implicitly unprojecting to 3D. In: Proceedings of the European conference on computer vision (ECCV). Springer, pp 194–210
46. Chen Y et al (2022) DSGN++: exploiting visual-spatial relation for stereo-based 3D detectors. IEEE Trans Pattern Anal Mach Intell (TPAMI) 1–14
47. Chen Y et al (2020) DSGN: deep stereo geometry network for 3D object detection. In: Conference on computer vision and pattern recognition (CVPR), pp 12 536–12 545
48. Guo X et al (2021) LIGA-Stereo: learning LiDAR geometry aware representations for stereo-based 3D detector. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 3153–3163
49. Reading C et al (2021) Categorical depth distribution network for monocular 3D object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 8555–8564

50. Liu Z et al (2022) BEVFusion: multi-task multi-sensor fusion with unified bird's-eye view representation. Computing research repository (CoRR). https://arxiv.org/abs/2205.13542
51. Liu Y et al (2022) Petr: position embedding transformation for multi-view 3d object detection. In: Proceedings of the European Conference on Computer Vision (ECCV). Springer, pp 531–548
52. Li Z et al (2022) Bevformer: learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In: Proceedings of the European conference on computer vision (ECCV). Springer, pp 1–18
53. Xia Z et al (2022) Vision transformer with deformable attention. Computing research repository (CoRR). https://arxiv.org/abs/2201.00520
54. Ma X et al (2019) Accurate monocular 3D object detection via color-embedded 3D reconstruction for autonomous driving. In: IEEE/CVF international conference on computer vision (ICCV), pp 6850–6859
55. Beltrán J et al (2018) BirdNet: a 3D object detection framework from LiDAR information. In: International conference on intelligent transportation systems (ITSC), pp 3517–3523
56. Yang B et al (2018) PIXOR: real-time 3D object detection from point clouds. In: IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 7652–7660
57. Lang AH et al (2019) Pointpillars: fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 12 697–12 705
58. Li B (2017) 3D fully convolutional network for vehicle detection in point cloud. In: IEEE/RSJ international conference on intelligent robots and systems (IROS), pp 1513–1518
59. Yan Y et al (2018) SECOND: sparsely embedded convolutional detection. Sensors 18(10):3337
60. Yin T et al (2021) Center-based 3D object detection and tracking. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 11 784–11 793
61. He C et al (2020) Structure aware single-stage 3D object detection from point cloud. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 11 873–11 882
62. Wu Z et al (2021) CIA-SSD: confident IoU-aware single-stage object detector from point cloud. Proc AAAI Conf Artif Intell (AAAI) 35(4):3555–3562
63. Ye D et al (2022) LidarMutliNet: unifying LiDAR semantic segmentation, 3D object detection, and panoptic segmentation in a single multi-task network. Computing research repository (CoRR). https://arxiv.org/abs/2206.11428
64. Lin T-Y et al (2017) Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp 2980–2988
65. Graham B et al (2017) Submanifold sparse convolutional networks. Computing research repository (CoRR). https://arxiv.org/abs/1706.01307
66. Zhou X et al (2020) Tracking objects as points. In: European conference on computer vision (ECCV). Springer, pp 474–490
67. Zhou X et al (2019) Objects as points. Computing research repository (CoRR). https://arxiv.org/abs/1904.07850
68. Teichmann M et al (2018) MultiNet: real-time joint semantic reasoning for autonomous driving. In: IEEE intelligent vehicles symposium (IV). IEEE, pp. 1013–1020
69. Gkioxari G et al (2019) Mesh R-CNN. In: Proceedings of the IEEE/CVF international conference on computer vision (CVPR), pp 9785–9795
70. Xu Q et al (2022) Behind the curtain: learning occluded shapes for 3D object detection. Proc AAAI Conf Artif Intell (AAAI) 36(3):2893–2901
71. Qi CR et al (2017) PointNet: deep learning on point sets for 3D classification and segmentation. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 77–85
72. Qi C et al (2017) PointNet++: deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems (NeurIPS), pp 5099–5108
73. Li J et al (2018) SO-Net: self-organizing network for point cloud analysis. In: IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 9397–9406

74. Wang Y et al (2019) Dynamic graph CNN for learning on point clouds. ACM Trans Graph (TOG) 38(5):1–12
75. Shi S et al (2019) PointRCNN: 3D object proposal generation and detection from point cloud. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 770–779
76. Qi CR et al (2019) Deep hough voting for 3d object detection in point clouds. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 9277–9286
77. Zhang Y et al (2022) Not all points are equal: learning highly efficient point-based detectors for 3D LiDAR point clouds. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 18 953–18 962
78. Shi S et al (2020) PV-RCNN: point-voxel feature set abstraction for 3D object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 10 529–10 538
79. Yang Z et al (2019) STD: sparse-to-dense 3D object detector for point cloud. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 1951–1960
80. Noh J et al (2021) HVPR: hybrid voxel-point representation for single-stage 3D object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 14 605–14 614
81. Deng J et al (2021) Voxel R-CNN: towards high performance voxel-based 3D object detection. Proc AAAI Conf Artif Intell (AAAI) 35(2):1201–1209
82. Bartsch A et al (2012) Pedestrian recognition using automotive radar sensors. Adv Radio Sci **10**(B.2), 45–55
83. Patel K et al (2019) Deep learning-based object classification on automotive radar spectra. In: IEEE radar conference (RadarConf), pp 1–6
84. Scheiner N et al (2020) Off-the-shelf sensor vs. experimental radar How much resolution is necessary in automotive radar classification?. In: International conference on information fusion (FUSION), pp 1–8
85. Schumann O et al (2019) Scene Understanding With Automotive Radar. IEEE Trans Intell Veh (TIV) 5(2):188–203
86. Danzer A et al (2019) 2d car detection in radar data with pointnets. In: IEEE intelligent transportation systems conference (ITSC). IEEE, pp 61–66
87. Dreher M et al (2020) Radar-based 2D car detection using deep neural networks. In: International conference on intelligent transportation systems (ITSC). IEEE, pp 1–8
88. Scheiner N et al (2021) Object detection for automotive radar point clouds - a comparison. AI Perspect 3(1):1–23
89. Chen X et al (2017) Multi-view 3D object detection network for autonomous driving. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 6526–6534
90. Zhang G et al (2019) Object detection and 3D estimation via an FMCW radar using a fully convolutional network. Computing research repository (CoRR). https://arxiv.org/abs/1902.05394
91. Sindagi VA et al (2019) Mvx-net: multimodal voxelnet for 3d object detection. In: International conference on robotics and automation (ICRA). IEEE, pp 7276–7282
92. Nabati R et al (2021) Center fusion: center-based radar and camera fusion for 3D object detection. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision (WACV), pp 1527–1536
93. Li Y et al (2022) DeepFusion: lidar-camera deep fusion for multi-modal 3D object detection. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp 17 182–17 191
94. Bai X et al (2022) TransFusion: robust LiDAR-camera fusion for 3D object detection with transformers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 1090–1099
95. Yang Z et al (2022) DeepInteraction: 3D object detection via modality interaction. Computing research repository (CoRR). https://arxiv.org/abs/2208.11112

96. Qian K et al (2021) Robust multimodal vehicle detection in foggy weather using complementary lidar and radar signals. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 444–453
97. Li Y et al (2022) Unifying voxel-based representation with transformer for 3d object detection. In: Advances in neural information processing systems (NeurIPS). https://openreview.net/forum?id=XA4ru9mfxTP
98. Xu S et al (2021) Fusion painting: multimodal fusion with adaptive attention for 3D object detection. In: IEEE international intelligent transportation systems conference (ITSC). IEEE, pp 3047–3054
99. Xu D et al (2018) Point fusion: deep sensor fusion for 3D bounding box estimation. In: IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 244–253
100. Goodfellow I et al (2014) Generative adversarial networks. In: Neural Information processing systems (NeurIPS), pp 2672–2680
101. Porav H et al (2018) Adversarial training for adverse conditions: robust metric localisation using appearance transfer. In: IEEE international conference on robotics and automation (ICRA), pp 1011–1018
102. Latif Y et al (2018) Addressing challenging place recognition tasks using generative adversarial networks. In: IEEE international conference on robotics and automation (ICRA), pp 2349–2355
103. Kendall A et al (2017) What uncertainties do we need in bayesian deep learning for computer vision? In: Advances in neural information processing systems (NeurIPS), pp 5574–5584
104. Kendall A et al (2018) Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 7482–7491
105. Yun P et al (2023) Laplace approximation based epistemic uncertainty estimation in 3D object detection. In: Conference on robot learning (CoRL). PMLR, pp 1125–1135
106. Yun P et al (2019) Focal Loss in 3D Object Detection. IEEE Robot Autom Lett (RAL) 4(2):1263–1270

Chapter 6
Collaborative 3D Object Detection

Siheng Chen and Yue Hu

Abstract 3D object detection is one of the most fundamental tasks of an autonomous system, which has made great progress recently. However, limited ability of individual vehicles results in the bottleneck of improvement of the 3D detection performance. To break through the limits of individual detection, collaborative 3D object detection has been proposed which enables agents to share information to perceive the environments beyond line-of-sight and field-of-view. Here, we provide an overview of the promising collaborative 3D object detection technology, including introducing the basic concepts and key challenges. We then focus on a fundamental tradeoff between communication cost and detection performance, and introduce the latest communication efficient collaborative 3D object detection method.

6.1 Introduction

When observing the world, each individual has a certain bias due to the limited field of view. The observation would be more holistic and robust when a group of individuals could collaborate and share information. In the parable of the blind men and an elephant, each blind man only feels a part of the elephant's body and perceives the elephant based on their limited observation. If they can fully collaborate with each other, they might be able to come closer to the truth. With decades of efforts on machine learning and computer vision, single-agent perception has made remarkable success in 2D/3D object detection, tracking and segmentation; however, it still suffers from a number of inevitable limitations due to an individual perspective, such as occlusion and long-range issues. Fortunately, with the fast development of

S. Chen (✉)
Shanghai Jiao Tong University and Shanghai AI Laboratory, Shanghai, China
e-mail: sihengc@sjtu.edu.cn

Y. Hu
Shanghai Jiao Tong University, Shanghai, China
e-mail: 18671129361@sjtu.edu.cn

R. Fan et al. (eds.), *Autonomous Driving Perception*, Advances in Computer Vision
and Pattern Recognition, https://doi.org/10.1007/978-981-99-4287-9_6

communication technologies, agents are able to share more information with each other, leading to an emerging field of collaborative perception.

Collaborative perception enables multiple agents to share complementary perceptual information with each other, promoting more holistic perception. It provides a new direction to fundamentally overcome a number of inevitable limitations of single-agent perception, such as occlusion and long-range issues. Related methods and systems are desperately needed in a broad range of real-world applications, such as vehicle-to-everything-communication-aided autonomous driving [1–3], multi-robot warehouse automation system [4, 5] and multi-UAVs (unmanned aerial vehicles) for search and rescue [6–8]. To realize collaborative perception, recent works have contributed high-quality datasets [9–11] and effective collaboration methods [2, 12–19].

Among many perception tasks, 3D object detection aims to determine the 3D location and the category of each foreground object in a given scene, which is fundamental for exploring the physical space [20–23]. 3D object detection has become one of the cornerstone techniques in most autonomous driving, robotics and video surveillance systems. Here we specifically consider collaborative 3D object detection for two reasons. First, most agents move and observe in the 3D space, and naturally, they should be able to understand 3D information and share them with each other. This is nontrivial because the 3D physical space is the foundation for multiple agents to collaborate. Essentially, occlusion and long-range issues do not occur without being aware of the 3D space. What an agent needs to understand is the physical space, instead of its own measurement. Second, the field of 3D object detection needs new directions. Current techniques based on an individual agent has become relatively mature. Most current progresses requires a huge amount of data and computational resources, instead of scientific thinking. Collaborative 3D object detection provides an orthogonal approach for exploring.

6.1.1 Significance

Scientific merits. Collaborative 3D object detection is not a simple extension of ordinary 3D object detection. In principle, collaborative 3D object detection can easily achieve what ordinary 3D object detection can never do, such as seeing through occlusion. This new task sits at the crux of communication, machine learning, computer vision, and robotics, and brings a series of new and challenging scientific questions: Given limited time and resources, who should an agent collaborate with? What information should an agent share to optimize the collaboration benefits? How to fuse heterogeneous information from multiple agents to improve the detection robustness? We might need to develop new information theories to seriously answer those questions. Researchers from different communities could ask a lot of diverse questions.

Engineering impacts. It is not difficult to connect collaborative 3D object detection to many ambitious engineering projects, such as autonomous driving, drone swarm and metaverse. Let us take autonomous driving as an example. Benefiting from the better building of communication infrastructure and developing communication technology such as V2X (Vehicle-to-Everything) communication, vehicles could exchange their messages in reliable manners, which enables the collaboration among them. Recent works [24, 25] have demonstrated that collaboration among vehicles could improve the accuracy of environmental perception as well as robustness and safety of transportation systems. In addition, collaborative 3D object detection can achieve what ordinary 3D object detection can do at a much lower cost. Autonomous driving vehicles are usually equipped with high-fidelity sensors to achieve reliable perception, which causes expensive cost. Is it a waste for a single vehicle to collect so much data all the time? Collaborative 3D detection can relax the harsh demand of perception equipments of individual vehicles.

6.1.2 Relations to Related Topics

The idea of collaborative 3D object detection is deeply rooted in collective intelligence, which is sometimes used synonymously with the wisdom of crowds. Technically, collaborative 3D object detection is new and related to multiple well established fields, especially multi-agent reinforcement learning. Multi-agent reinforcement learning is a sub-field of reinforcement learning and studies optimizing the decision-making process of multiple agents that coexist in a shared environment via reinforcement learning techniques [26–28]. Both multi-agent reinforcement learning and collaborative 3D object detection leverages collaboration or information sharing to improve the system performance. However, there are two distinct differences. First, multi-agent reinforcement learning emphasizes reinforcement learning techniques, while collaborative 3D object detection focuses on the task of 3D object detection. Second, multi-agent reinforcement learning considers information sharing at a level of decision making, where the perception needs are either trivial or secondary. In comparison, collaborative 3D object detection focuses on solving real-world perception issues. Intuitively, better perception enables better performances on all the subsequent tasks, including decision making. In future, it would be interesting to combine multi-agent reinforcement learning and collaborative 3D object detection to develop a multi-agent system with more comprehensive collaboration ability.

6.1.3 Category of Collaborative 3D Object Detection

The pipeline of 3D object detection is that raw data collected by an agent is firstly fed into encoder, and then the intermediate features output by encoder are decoded to output the final perception results. Depending on when the collaboration occurs in the pipeline, we can categorize the collaboration mode into four types, including early collaboration, intermediate collaboration, late collaboration and hybrid collaboration.

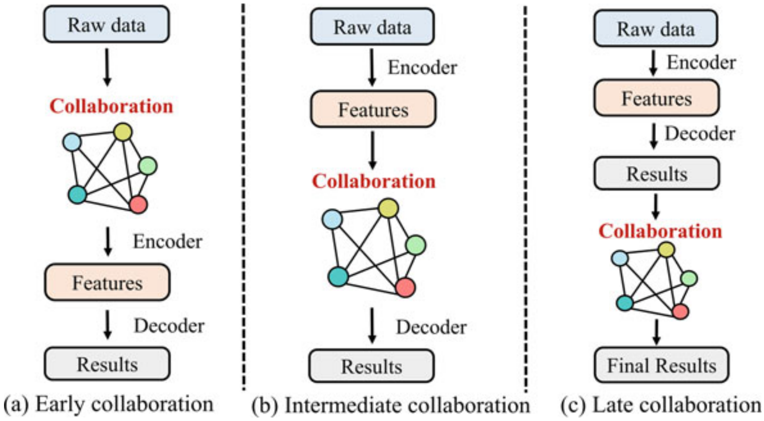


Fig. 6.1 Three types of collaboration

Early collaboration. Early collaboration conducts the collaboration in the input space, which shares raw sensory data between agents. It aggregates raw measurements from all the agents to promote a holistic perspective. Therefore, each agent could conduct following processing and finish detection based on the holistic perspective, see Fig. 6.1a, which can fundamentally solve the occlusion and long-range issues occurring in the single-agent 3D detection. References [29, 30] have adopted early collaboration mode and demonstrated its effectiveness with the help of rich information. However, sharing raw sensory data requires a lot of communication and easily congest the communication network with heavy data loads, which impedes its practical usage in most cases.

Late collaboration. Late collaboration conducts the collaboration in the output space, which promotes the fusion of the detection result output by each individual agent to achieve an refinement, see Fig. 6.1c. Reference [31] adopted late collaboration to develop a perception and localization system and dealt with latency and dropout of the communication link between the two agents. Reference [32] studied temporal and spatial alignment of the shared detected objects and proposed to use non-predicted sender state for the transformation and therefore to neglect the sender motion compensation. Although late collaboration is bandwidth-economic, it is very sensitive to the positioning error of the agents and suffers from high estimation errors and noise because of incomplete local observation.

Intermediate collaboration. Intermediate collaboration conducts the collaboration in the intermediate feature space. It enables the transmission of the intermediate features generated by each individual agent's prediction model. After fusion of these features, each agent decodes the fused features and produce the perception results, see Fig. 6.1b. Conceptually, we can squeeze representative information to those features, leading to economic communication bandwidth compared to early collaboration as well as upgraded perception ability compared to late collaboration. A lot of work [1, 33–36] agree with this idea and adopt intermediate collaboration and feature sharing.

In practice, the design of this collaboration strategy is algorithmically challenging from two aspects: (i) how to select the most beneficial and compact features from the raw measurements for transmission; and (ii) how to maximally fuse the other agents' features to enhance each agent's perception ability.

Hybrid collaboration. As mentioned above, each collaboration mode has its own advantages and disadvantages. Therefore, some work adopted hybrid collaboration which combines two or more collaboration modes to optimize the collaboration strategy. Reference [30] proposed to share high level information (late collaboration) where the sensor has high visibility and share low level information (early collaboration) where the visibility is poor. Their method is based on the observation that objects close to a sensor will have a high density of points and thus are more likely to be detected using a single sensor's observation. DiscoNet [2] leveraged a teacher model employing early collaboration to guide the training of student model employing intermediate collaboration. In reference stage, the communication bandwidth-consuming teacher model is discarded so that student model can keep superior performance with low communication bandwidth because it has learned the knowledge from the teacher model at the training stage.

6.2 Key Challenges

In this section, we consider two representative challenges in practical collaborative 3D object detection, including communication constraints and pose errors.

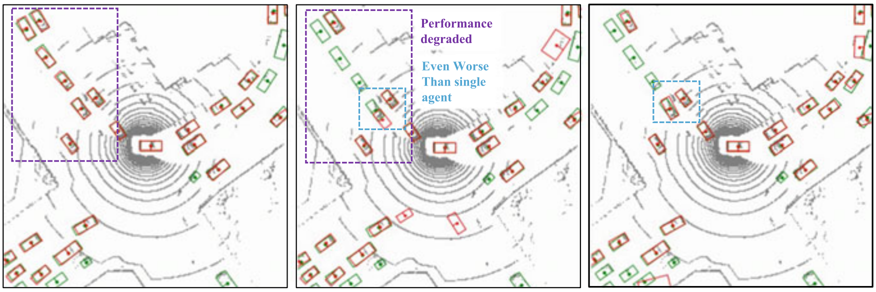
6.2.1 Communication Constraints

Collaborative 3D object detection requires a communication system to share information between agents. Vehicle-to-vehicle (V2V) communication can be implemented by two communication solutions, either IEEE 802.11p protocol or cellular network standards [37]. In IEEE 802.11p protocol, stations do not need to join a BSS (Basic Service Set) by operating in WAVE (Wireless Access in Vehicular Environment) mode, which reduces the connection setup overhead and suits vehicular safety applications well [38]. On the other hand, the fourth-generation cellular networks support LTE V2V standard development, supporting vehicular user equipments (VUEs) with low latency and highly reliable data transmission [39]. Compared to the 802.11p based V2V communication, it avoids channel congestion and collision induced by CSMA mechanism [40]. Though communication technology keeps developing for lower latency and better reliability, real-world communication systems are still far from ideal. They are always constrained by limited communication bandwidths, latency constraints or package dropout issues.

Intuitively, when more information are allowed to share between agents, each agent would collect more perceptual information and achieve better detection performances. However, bandwidth resources are always limited. Especially, when more agents are participated in the collaboration, available communication bandwidth for each agent is more limited. Thus, there is a fundamental tradeoff between the gain of detection ability and the expensive of communication bandwidth. This requires a collaboration strategy to select compact and critical messages for sharing, which cost less communication bandwidth, yet convey significant detection information [2, 41]. Currently, this fundamental tradeoff is one of the biggest challenges in collaborative 3D object detection.

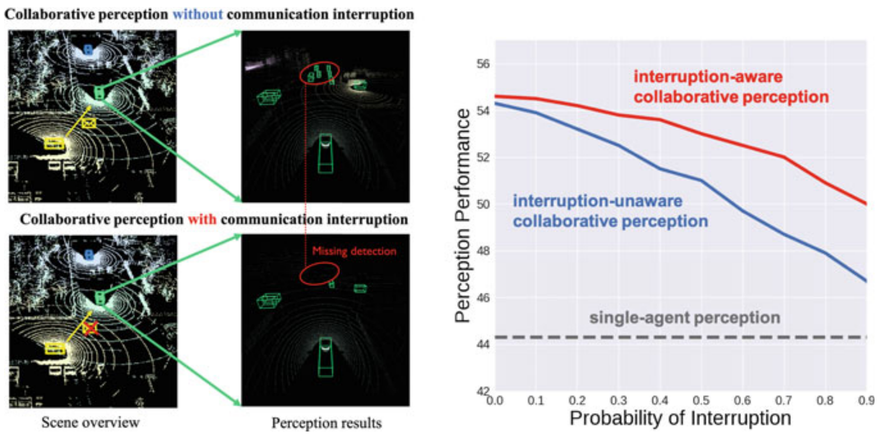
Communication latency is another fundamental issue. All the communication systems are designed to promote less latency, however, in practice, latency is inevitable and can cause huge troubles. In a real-time LTE-V2X communication system, the latency time is up to an average of (498 communication periods) + 131.30 ms. Besides, the varying latency times of various communication channels would cause severe time asynchronous issues. Experimentally, latency issue severely damages the collaborative 3D object detection system, resulting in even worse performance than single-agent 3D detection [16]. From Fig. 6.2, we see that: (i) the detected vehicles in the purple box in (a) with collaboration is missed in the (b); (ii) the correctly detected vehicles in the blue box in (c) are incorrect in (b). The reason is that the received collaborative data with latency represents the situation 1s ago, it misleads the detector to output boxes with significant deviation. This motivates us to consider a collaborative perception system robust to the inevitable communication latency.

Similarly, communication interruption also effects the robustness of a collaborative 3D object detection system. Random temporary interruption is one of the common communication problems caused by numerous environmental factors, including unstable communication channels and equipment failure. In this case, the communication link between each pair of two agents might be interrupted with a certain probability at each moment. This results in a dynamic, incomplete communication graph, which would severely degrade the collaboration performance and further affect the downstream tasks, such as tracking and trajectory prediction, causing a cascading failure; see Fig. 6.3. Figure 6.3a shows that the green vehicle can expand



(a) Collaboration without latency (b) Collaboration with 1s latency (c) Single-agent perception

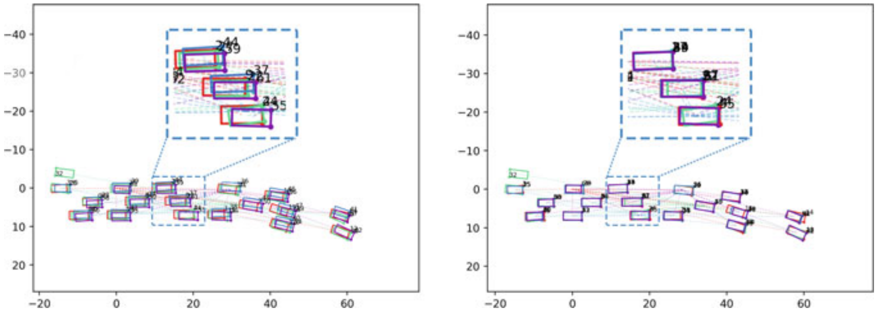
Fig. 6.2 Collaborative 3D detection. Red: *Detected*, green: *Ground truth*. Collaboration without considering latency could be even worse than no collaboration



(a) Interruption showcase

(b) Interruption affect

Fig. 6.3 Communication interruption issue and its affect



(a) Misaligned boxes

(b) Aligned boxes

Fig. 6.4 Pose errors could cause misaligned boxes after collaboration

its perception range to overcome occlusion and long-range issues and detect more objects by leveraging the supportive message sent by the yellow vehicle. However, when the communication between two vehicles is stochastically interrupted, the performance of collaborative perception becomes unstable: the detected objects sometimes appear, sometimes are missing, causing noisy inputs for the downstream tasks, such as tracking and trajectory prediction. Figure 6.3b shows the empirical performances of collaborative perception methods as a function of interruption probability on the V2X-Sim [2] dataset. We see that the performance are seriously degraded due to communication interruption. Fortunately, the proposed interruption aware collaborative perception framework (red curve) effectively alleviates the degradation (Fig. 6.4).

A diverse of practical communication constraints severely effect the overall performance of a collaborative 3D object detection system. However, related research works are clearly underexplored. The future development might require the knowledge and expertise from both communication and machine learning communities.

6.2.2 Pose Errors

In an autonomous system, the localization module is designed to find the ego position relative to the reference position. It consumes the real-time measurements from multiple sensors, such as LiDAR, IMU, GPS, odometer, cameras, as well as some geometric prior, such as maps. Because of running in the 3D space, the ego position of an agent is a 6DOF pose (translation and rotation). Precise localization of each agent is a foundation of multi-agent collaboration [42–46]. To share valid information with each other, multiple agents need precise poses to synchronize their individual data in a consistent spatial coordinate system. However, the 6 DoF pose estimated by each agent's localization module is not perfect in practice, causing annoying relative pose errors. Inaccurate poses would cause misalignment and inconsistency in collaboration, resulting in worse perception performance than single-agent perception [47].

To gain resistance to localization errors, previous works consider two main approaches: supervised training or robust network design. The first approach introduces additional supervision to empower the network being aware of the pose errors. For example, V2VNet (robust) [42] designs pose regression, global consistency and attention aggregation module to correct relative poses and concentrate on neighbor with less pose error; MASH [43] builds a similarity volume and explicitly learns the pixel to pixel correspondence to avoid using noisy pose in inference. The second approach focuses on designing robust frameworks or network architectures. For example, V2X-ViT [44] uses multi-scale window attention to capture features in various ranges; and FPV-RCNN [45] infers the semantic label of keypoints and finds correspondences between agents to correct relative poses.

6.3 Communication-Efficient Collaborative 3D Object Detection

In this section, we focus on studying the fundamental trade-off between detection performance and communication bandwidth. We introduce a communication-efficient collaborative 3D object detection method named `Where2comm`, which significantly reduces the communication cost without losing detection performance.

6.3.1 Problem Formulation

Consider N agents in the scene. Let $\mathcal{X}_i$ and $\mathcal{Y}_i$ be the observation and the perception supervision of the ith agent, respectively. The objective of collaborative perception is to achieve the maximized perception performance of all agents as a function of the total communication budge B and communication round K; that is,

$$\xi_\Phi(B,K) = \arg\max_{\theta,\mathcal{P}} \sum_{i=1}^{N} g\left(\Phi_\theta\left(\mathcal{X}_i, \{\mathcal{P}_{i\to j}^{(K)}\}_{j=1}^{N}\right), \mathcal{Y}_i\right), \qquad (6.1)$$

$$\text{subject to } \sum_{k=1}^{K}\sum_{i=1}^{N} |\mathcal{P}_{i\to j}^{(k)}| \le B,$$

where $g(\cdot,\cdot)$ is the perception evaluation metric, Φ is the perception network with trainable parameter θ, and $\mathcal{P}_{i\to j}^{(k)}$ is the message transmitted from the ith agent to the jth agent at the kth communication round. Note that (i) when $B = K = 0$, there is no collaboration and $\xi_\Phi(0,0)$ reflects the single-agent perception performance; (ii) through optimizing the communication strategy and the network parameter, collaborative perception should perform well consistently at any communication bandwidth or round; and (iii) we consider multi-round communication, where each agent serves as both a supporter (offering message to help others) and a requester (requesting messages from others).

To achieve a better balance between detection performance and communication bandwidth (6.1), previous works put forth solutions from several perspectives. For example, When2com [12] considers a handshake mechanism which selects the most relevant collaborators; V2VNet [1] considers end-to-end-learning-based source coding; and DiscoNet [2] uses 1D convolution to compress message. However, all previous works make a plausible assumption: once two agents collaborate, they are obligated to share perceptual information of all spatial areas *equally*. This unnecessary assumption can hugely waste the bandwidth as a large proportion of spatial areas may contain irrelevant information for perception task. Figure 6.5 illustrates such a spatial heterogeneity of perceptual information. Consider this safety-critical scenario, where the white car and the red car may collide due to occlusion. This collision could be avoided when the blue car can simply share a message about the red car's position to the white car. This idea brings a new dimension, which enables each agent to explore its spatial dimension and select spatially sparse, yet perceptually critical messages.

Figure 6.6 presents the comparisons on the communication graph with previous works. *Fully connected* versus *agent-level partially connected* versus ours *spatial-decouple partially connected* communication. *Fully connected* communication results in a large amount of bandwidth usage, growing on the order of $O(N^2)$, where N is the number of agents in a network. *Agent-level partially connected* communication prune irrelevant connections between agents while may erroneously sever the information connection. *Spatial-decouple partially connected* communication

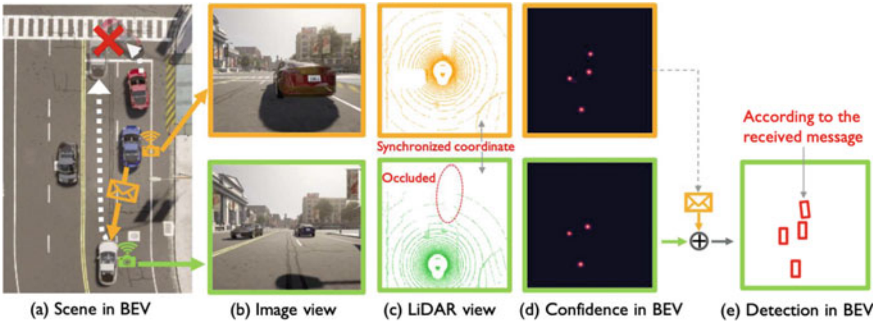


Fig. 6.5 Considering the precious communication bandwidth, each agent needs to speak to the point!

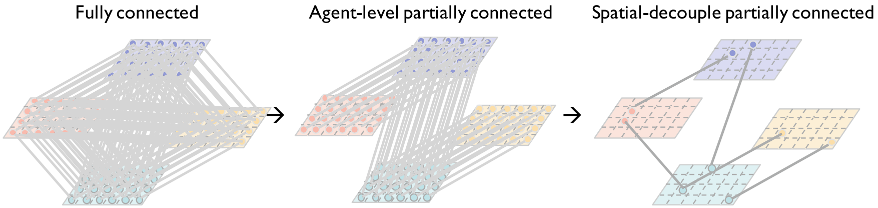

Fully connected
Agent-level partially connected
Spatial-decouple partially connected

Fig. 6.6 Spatial confidence-aware communication graph construction module. We spatially decouple the full feature map, and could flexibly involve the informative spatial areas in the communication. This *spatial-decouple partially connected* communication could further flexibly prune irrelevant connections per-location and is more bandwidth-efficient

could further flexibly prune irrelevant connections per-location and can substantially reduce the overall network complexity. Here we introduce `Where2comm`, which implements the spatial-decouple partially connected communication and adaptively selects where to communicate.

6.3.2 Mathematical Intuition

We now introduce `Where2comm` from a perspective of mathematical optimization. Mathematically, it is hard to obtain the global optimum of (6.1) due to hard constrains and non-differentialability of binary variables. Therefore, we introduce an auxiliary variable and decomposes the original problem into two sub-optimization problems, each one of which is easy to solve. To understand the details, let us consider a setting of fixed communication bandwidth and communication round, $K = 1$, $B = [B_1]$. Then, the optimization is

$$\max_{\theta,\mathcal{P}} \sum_{i=1}^{N} g\left(\Phi_\theta\left(\mathcal{X}_i, \{\mathcal{P}_{i\to j}\}_{j=1}^{N}\right), \mathcal{Y}_i\right), \tag{6.2}$$

$$\text{subject to} \sum_{i,j=1}^{N} |\mathcal{P}_{i\to j}| \le B_1.$$

Let the message sent from the ith agent to the jth agent be $\mathcal{P}_{i\to j} = \mathbf{M}_{i\to j} \odot \mathcal{F}_i$, where $\mathbf{M}_{i\to j} \in \{0,1\}^{H\times W}$ is a binary selection mask and $\mathcal{F}_i \in \mathbb{R}^{H\times W\times D}$ is the ith agent's feature map. Note that $\mathbf{M}_{i\to j}$ determines the message's spatial sparsity; that is, where to communicate. Then, the original optimization is equivalent to

$$\max_{\theta,\mathbf{M}} \sum_{i=1}^{N} g\left(\Phi_\theta\left(\mathcal{X}_i, \{\mathbf{M}_{i\to j}\}_{j=1}^{N}\right), \mathcal{Y}_i\right), \tag{6.3}$$

$$\text{subject to} \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} |\mathbf{M}_{i\to j}| \le b_1, \mathbf{M}_{i\to j} \in \{0,1\}^{H\times W},$$

where $\mathcal{F}_i$ can attribute to the network $\Phi_\theta(\cdot)$ and input data $\mathcal{X}_i$ and $b_1 = B_1/D$ with D the channel number of $\mathcal{F}_i$. Due to the binary constrains, it is hard to optimize (6.3) directly. Instead, we decompose (6.3) into two sub-optimization problems and optimize the binary selection matrix $\mathbf{M}_{i\to j}$ and the network parameters θ once at a time: (i) obtain a feasible binary selection matrix $\mathbf{M}_{i\to j}$ by optimizing a proxy constrained problem; (ii) given the feasible binary selection matrix $\mathbf{M}_{i\to j}$, optimize the perception network parameter θ. The constraint is satisfied in (i) and the perception goal is achieved in (ii). Specifically, two sub-optimization problems are

- **Obtain a feasible binary selection matrix $\mathbf{M}_{i\to j}$.** This essentially optimizes where to allocate the communication bandwidth. Intuitively, the spatial confidence reflects the perceptually critical level, so that those spatial regions with higher spatial confidence will provide more critical information to help the partners and should have a higher priority be selected.

Following this spirit, we consider a proxy constrained problem as follows,

$$\max_{\mathbf{M}} \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \mathbf{M}_{i\to j} \odot \mathbf{C}_i, \quad \text{s.t.} \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} |\mathbf{M}_{i\to j}| \le b_1, \mathbf{M}_{i\to j} \in \{0,1\}^{H\times W}, \tag{6.4}$$

where $\mathbf{C}_i$ is the spatial confidence map. Note that (i) even this optimization problem has hard constraints and non-differentialability of binary variables, it has an analytical solution that naturally satisfies all the constraints in (6.3); and (ii) even we cannot solve the original objective, this proxy objective still carries the similar idea to promote better, yet more compact perception. This solution is obtained by selecting those spatial regions whose corresponding elements in $\mathbf{M}$ rank top-b_1. The detailed steps of **selection function** are: (i) arrange the elements in the input matrix in descending order; (ii) given the communication budget constrain, decide the total number (b_1) of communication regions; (iii) set the spatial regions of $\mathbf{M}$, where elements rank in top-b_1 as the 1 and 0 verses.

• **Given the feasible binary selection matrix, optimize the network parameter** θ. This essentially optimizes the perception performance. The sub-problem is

$$\max_{\theta} \sum_{i=1}^{N} g\left(\Phi_{\theta}\left(\mathcal{X}_i, \{\mathbf{M}_{i\rightarrow j}\}_{j=1}^{N}\right), \mathcal{Y}_i\right). \tag{6.5}$$

This can be solved by standard **supervised learning**. For example, the perception evaluation metric $g(\cdot)$ can be evaluated by the detection loss calculated between detections and the ground-truth and the detection loss is optimized with an Adam optimizer. We thus get the optimized perception network parameter θ. Note that this sub-problem does not involve any constraints and is thus easy to optimize.

6.3.3 System Design

We now present the architecture implementation of `Where2comm`. Figure 6.7 demonstrates the overall system, including an observation encoder, a spatial confidence generator, the spatial confidence-aware communication module, the spatial confidence-aware message fusion module and a detection decoder. Among five modules, the proposed spatial confidence generator generates the spatial confidence map. Based on this spatial confidence map, the proposed spatial confidence-aware communication generates compact messages and sparse communication graphs to save communication bandwidth; and the proposed spatial confidence-aware message fusion module leverages informative spatial confidence priors to achieve better aggregation.

Observation encoder. The observation encoder extracts feature maps from the sensor data. `Where2comm` accepts single/multi-modality inputs, such as RGB

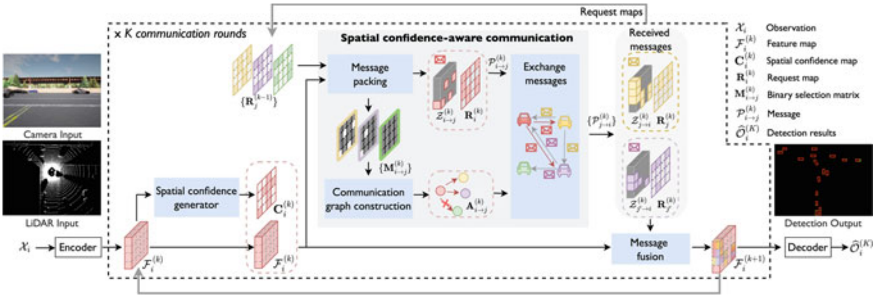

Request maps
× K communication rounds
Spatial confidence-aware communication
Camera Input
LiDAR Input
$\mathcal{X}_i$
Encoder
$\mathcal{F}_i^{(k)}$
Spatial confidence generator
$\mathbf{C}_i^{(k)}$
$\{\mathbf{R}_j^{(k-1)}\}$
Message packing
$\mathcal{Z}_{i\rightarrow j}^{(k)}$
$\mathbf{R}_i^{(k)}$
$\{\mathbf{M}_{i\rightarrow j}^{(k)}\}$
Communication graph construction
$\mathbf{A}_{i\rightarrow j}^{(k)}$
$\mathcal{P}_{i\rightarrow j}^{(k)}$
Exchange messages
$\{\mathcal{P}_{j\rightarrow i}^{(k)}\}$
Received messages
$\mathcal{Z}_{j\rightarrow i}^{(k)}$
$\mathbf{R}_j^{(k)}$
Message fusion
$\mathcal{F}_i^{(k+1)}$
Decoder
$\widehat{\mathcal{O}}_i^{(K)}$
Detection Output
$\mathcal{X}_i$ Observation
$\mathcal{F}_i^{(k)}$ Feature map
$\mathbf{C}_i^{(k)}$ Spatial confidence map
$\mathbf{R}_i^{(k)}$ Request map
$\mathbf{M}_{i\rightarrow j}^{(k)}$ Binary selection matrix
$\mathcal{P}_{i\rightarrow j}^{(k)}$ Message
$\widehat{\mathcal{O}}_i^{(K)}$ Detection results

Fig. 6.7 System overview. In `Where2comm`, spatial confidence generator enables the awareness of spatial heterogeneous of perceptual information, spatial confidence-aware communication enables efficient communication, and spatial confidence-aware message fusion boosts the performance

images and 3D point clouds. This work adopts the feature representations in bird's eye view (BEV), where all agents project their individual perceptual information to the same global coordinate system, avoiding complex coordinate transformations and supporting better shared cross-agent collaboration. For the ith agent, given its input $\mathcal{X}_i$, the feature map is

$$\mathcal{F}_i^{(0)} = \Phi_{\text{enc}}(\mathcal{X}_i) \in \mathbb{R}^{H \times W \times D},$$

where $\Phi_{\text{enc}}(\cdot)$ is the encoder, the superscript 0 reflects that the feature is obtained before communication and H, W, D are its height, weight and channel. All agents share the same BEV coordinate system. For the image input, $\Phi_{\text{enc}}(\cdot)$ is followed by a warping function that transforms the extracted feature from front-view to BEV. For 3D point cloud input, we discretize 3D points as a BEV map and $\Phi_{\text{enc}}(\cdot)$ extracts features in BEV. The extracted feature map is output to the spatial confidence generator and the message fusion module.

When the input sensor is a monocular camera, the depth is unknown and estimated. Instead of directly projecting 2D features to flat ground space, we consider lifting those features to 3D voxel space and then collapse them to the BEV. This design considers all the possible depths/altitudes, introducing flexibility in the projection, and mitigating the distortion effect caused by information loss in imaging. The detailed steps are: **(1)** Categorical Depth Distribution Network (CaDDN [22]), which is a recent and effective method to warp image feature to BEV feature, is applied to estimate the depth distribution for each image feature point. **(2)** Each feature point is wrapped from the 2D image space to the 3D physic space according to the known camera parameters. **(3)** The 3D voxel features are flattened to BEV features. Briefly, the warping function is unfolded as follows: for each image feature point locates at (u, v), given the estimated categorical depth d_i, and the known camera projection matrix $\mathbf{P} \in \mathbb{R}^{3\times4}$, 3D physical space coordinates $[x, y, z]^T$ is calculated conditioned on the image feature coordinates $[u, v, d_i]^T$ based on the projection function: $[u, v, d_i]^T = \mathbf{P} \cdot [x, y, z, 1]^T$.

Spatial confidence generator. The spatial confidence generator generates a spatial confidence map from the feature map of each agent. The spatial confidence map reflects the perceptually critical level of various spatial areas. Intuitively, for object detection task, the areas that contain objects are more critical than background areas. During collaboration, areas with objects could help recover the miss-detected objects due to the limited view; and background areas could be omitted to save the precious bandwidth. So we represent the spatial confidence map with the detection confidence map, where the area with high perceptually critical level is the area that contains an object with a high confidence score.

To implement, we use a detection decoder structure to produce the detection confidence map. Given the feature map at the kth communication round, $\mathcal{F}_i^{(k)}$, the corresponding spatial confidence map is

$$\mathbf{C}_i^{(k)} = \Phi_{\text{generator}}(\mathcal{F}_i^{(k)}) \in [0, 1]^{H \times W}, \tag{6.6}$$

where the generator $\Phi_{\text{generator}}(\cdot)$ follows a detection decoder. Since we consider multi-round collaboration, Where2comm iteratively updates the feature map by aggregating information from other agents. Once $\mathcal{F}_i^{(k)}$ is obtained, (6.6) is triggered to reflect the perceptually critical level at each spatial location. The proposed spatial confidence map answers a crucial question that was ignored by previous works: for each agent, information at which spatial area is worth sharing with others. By answering this, it provides a solid base for efficient communication and effective message fusion.

Spatial confidence-aware communication. With the guidance of spatial confidence maps, the proposed communication module packs compact messages with spatially sparse feature maps and transmits messages through a sparsely-connected communication graph. Most existing collaboration perception systems [1, 2, 48] considers full feature maps in the messages and fully-connected communication graphs. To reduce the communication bandwidth without affecting perception, we leverage the spatial confidence map to select the most informative spatial areas in the feature map (where to communicate) and decide the most beneficial collaboration partners (who to communicate).

Message packing determines what information should be included in the to-be-sent message. The proposed message includes: (i) a request map that indicates at which spatial areas the agent needs to know more; and (ii) a spatially sparse, yet perceptually critical feature map.

The request map of the ith agent is $\mathbf{R}_i^{(k)} = 1 - \mathbf{C}_i^{(k)} \in \mathbb{R}^{H \times W}$, negatively correlated with the spatial confidence map. The intuition is, for the locations with low confidence score, an agent is hard to tell if there is really no objects or it is just caused by the limited information (e.g. occlusion). Thus, the low confidence score indicates there could be missing information at that location. Requesting information at these locations from other agents could improve the current agent's detection accuracy.

The spatially sparse feature map are selected based on each agent's spatial confidence map and the received request maps from others. Specifically, a binary selection matrix is used to represent each location is selected or not, where 1 denotes selected, and 0 elsewhere. For the message sent from the ith agent to the jth agent at the kth communication round, the binary selection matrix is

$$\mathbf{M}_{i \to j}^{(k)} = \begin{cases} \Phi_{\text{select}}(\mathbf{C}_i^{(k)}) \in \{0, 1\}^{H \times W}, & k = 0; \\ \Phi_{\text{select}}(\mathbf{C}_i^{(k)} \odot \mathbf{R}_j^{(k-1)}), \in \{0, 1\}^{H \times W}, & k > 0; \end{cases} \tag{6.7}$$

where $\odot$ is the element-wise multiplication, $\mathbf{R}_j^{(k-1)}$ is the request map from the jth agent received at the previous round, $\Phi_{\text{select}}(\cdot)$ is the selection function which targets to select the most critical areas conditioned on the input matrix, which represents the critical level at the certain spatial location. We implement $\Phi_{\text{select}}(\cdot)$ by selecting the locations where the largest elements at in the given input matrix conditioned on the bandwidth limit; optionally, a Gaussian filter could be applied to filter out the outliers and introduce some context. In the initial communication round, each agent selects the most critical areas from its own perspective as the request maps from

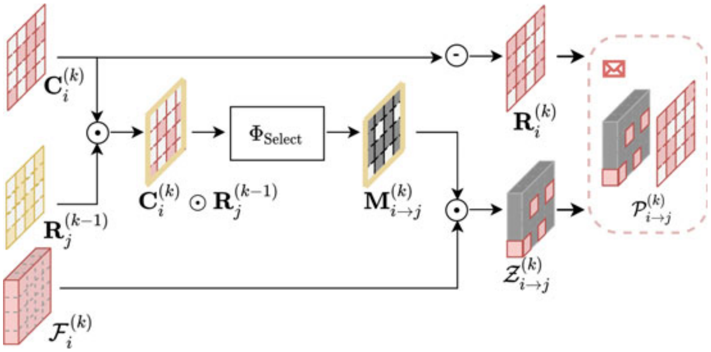

$\mathbf{C}_i^{(k)}$
$\mathbf{R}_j^{(k-1)}$
$\mathcal{F}_i^{(k)}$
$\mathbf{C}_i^{(k)} \odot \mathbf{R}_j^{(k-1)}$
Φ_{Select}
$\mathbf{M}_{i\to j}^{(k)}$
$\mathbf{R}_i^{(k)}$
$\mathcal{Z}_{i\to j}^{(k)}$
$\mathcal{P}_{i\to j}^{(k)}$

Fig. 6.8 Spatial confidence-aware message packing module. ⊙ denotes point-wise multiplication, ⊖ denotes point-wise minus by a matrix with the same shape as the input and filled with 1. Best viewed in color. Grey denotes the location being filled with zeros for the binary selection matrix $\mathbf{M}_{i\to j}^{(k)}$ and the feature map $\mathcal{Z}_{i\to j}^{(k)}$

other agents are not available yet; in the subsequent rounds, each agent also takes the partner's request into account, enabling more targeted communication. Then, the selected feature map is obtained as $\mathcal{Z}_{i\to j}^{(k)} = \mathbf{M}_{i\to j}^{(k)} \odot \mathcal{F}_i^{(k)} \in \mathbb{R}^{H\times W\times D}$, which provides spatially sparse, yet perceptually critical information.

Figure 6.8 presents the detail about the spatial confidence-aware message packing module. For the message from agent i to agent j at kth communication round, the module takes the spatial confidence map $\mathbf{C}_i^{(k)}$ of agent i and the request map $\mathbf{R}_j^{(k-1)}$ of agent j as input, and outputs the message $\mathcal{P}_{i\to j}^{(k)}$ including the masked feature map $\mathcal{Z}_{i\to j}^{(k)}$ and the request map of agent i.

Overall, the message sent from the ith agent to the jth agent at the kth communication round is $\mathcal{P}_{i\to j}^{(k)} = (\mathbf{R}_i^{(k)}, \mathcal{Z}_{i\to j}^{(k)})$. Note that (i) $\mathbf{R}_i^{(k)}$ provides spatial priors to request complementary information for the ith agent's need in the next round; the feature map $\mathcal{Z}_{i\to j}^{(k)}$ provides supportive information for the ith agent's need in the this round. They together enable mutually beneficial collaboration; (ii) since $\mathcal{Z}_{i\to j}^{(k)}$ is sparse, we only transmit non-zero features and corresponding indices, leading to low communication cost; and (iii) the sparsity of $\mathcal{Z}_{i\to j}^{(k)}$ is determined by the binary selection matrix, which dynamically allocates the communication budget at various spatial areas based on their perceptual critical level, adapting to various communication conditions.

Communication graph construction targets to identify when and who to communicate to avoid unnecessary communication that wastes the bandwidth. Most previous works [1, 2, 10] consider fully-connected communication graphs. When2com [12] proposes a handshake mechanism, which uses similar global features to match partners. This is hard to interpret because two agents, which have similar global features, do not necessarily need information from each other. Different from all previous works, we provide an explicit design rationale: the necessity of communication between the ith and the jth agents is simply measured by the overlap between the information that the ith agent has and the information that the jth agent needs. With the help of the spatial confidence map and the request map, we construct a more interpretable communication graph.

For the initial communication round, every agent in the system is not aware of other agents yet. To activate the collaboration, we construct a fully-connected communication graph. Every agent will broadcast its message to the rest of the system. For the subsequent communication rounds, we examine if the communication between agent i and agent j is necessary based on the maximum value of the binary selection matrix $\mathbf{M}_{i\to j}^{(k)}$, i.e. if there is at least one patch is activated, then we regard the connection is necessary. Formally, let $\mathbf{A}^{(k)}$ be the adjacency matrix of the communication graph at the kth communication round, whose (i, j)th element is

$$\mathbf{A}_{i,j}^{(k)} = \begin{cases} 1, & k = 0; \\ \max_{h\in\{0,1,..,H-1\},w\in\{0,1,...,W-1\}} \left(\mathbf{M}_{i\to j}^{(k)}\right)_{h,w} \in \{0, 1\}, & k > 0; \end{cases} \tag{6.8}$$

where h, w index the spatial area, reflecting message passing from the ith agent to the jth agent. Given this sparse communication graph, agents can exchange messages with selected partners.

Spatial Confidence-Aware Message Fusion. Spatial confidence-aware message fusion targets to augment the feature of each agent by aggregating the received messages from the other agents. To achieve this, we adopt a transformer architecture, which leverages multi-head attention to fuse the corresponding features from multiple agents at each individual spatial location. The key technical design is to include the spatial confidence maps of all the agents to promote cross-agent attention learning. The intuition is that, the spatial confidence map could explicitly reflect the perceptually critical level, providing a useful prior for attention learning.

Specifically, for the ith agent, after receiving the jth agent's message $\mathcal{P}_{j\to i}^{(k)}$, it could unpack to retrieve the feature map $\mathcal{Z}_{j\to i}^{(k)}$ and the spatial confidence map $\mathbf{C}_j^{(k)} = 1 - \mathbf{R}_j^{(k)}$. We also include the ego feature map in fusion and denote $\mathcal{Z}_{i\to i}^{(k)} = \mathcal{F}_i^{(k)}$ to make the formulation simple and consistent, where $\mathcal{Z}_{i\to i}^{(k)}$ might not be sparse. To fuse the features from the jth agent at the kth communication round, the cross-agent/ego attention weight for the ith agent is

$$\mathbf{W}_{j\to i}^{(k)} = \mathrm{MHA}_{\mathrm{W}}\left(\mathcal{F}_i^{(k)}, \mathcal{Z}_{j\to i}^{(k)}, \mathcal{Z}_{j\to i}^{(k)}\right) \odot \mathbf{C}_j^{(k)} \in \mathbb{R}^{H\times W}, \tag{6.9}$$

where $\mathrm{MHA}_{\mathrm{W}}(\cdot)$ is a multi-head attention applied at each individual spatial location, which outputs the scaled dot-product attention weight. Note that (i) the proposed spatial confidence maps contributes to the attention weight, as the features with higher perceptually critical level are more preferred in the feature aggregation; (ii) the cross-agent attention weight models the collaboration strength with a $H \times W$ spatial resolution, leading to more flexible information fusion at various spatial regions. Then, the feature map of the ith agent after fusing the messages in the kth communication round is

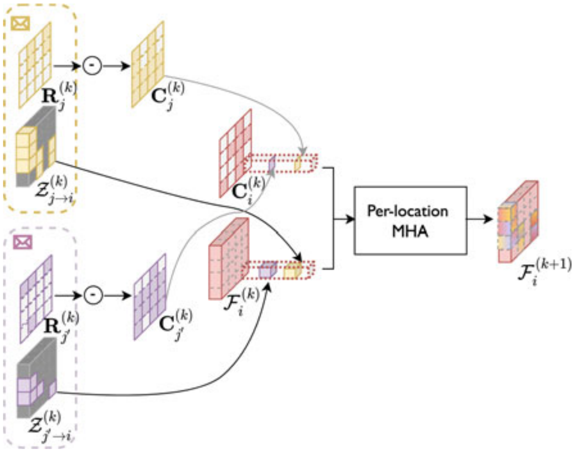


Fig. 6.9 Spatial confidence-aware message fusion module. Each agent attentively augments the features with the received messages at each location. And the per-location multi-head attention are separately operated at each location in parallel, it takes the features and the corresponding confidence scores as input, and outputs the augmented features

$$\mathcal{F}_i^{(k+1)} = \mathrm{FFN}\left(\sum_{j \in \mathcal{N}_i \bigcup \{i\}} \mathbf{W}_{j\to i}^{(k)} \odot \mathcal{Z}_{j\to i}^{(k)} \right) \in \mathbb{R}^{H\times W\times D}, \tag{6.10}$$

where FFN$(\cdot)$ is the feed-forward network and $\mathcal{N}_i$ is the neighbors of the ith agent defined in the communication graph $\mathbf{A}^{(k)}$. The fused feature $\mathcal{F}_i^{(k+1)}$ would serve as the ith agent's feature in the $(k+1)$th round. In the final round, we output $\mathcal{F}_i^{(k+1)}$ to the detection decoder to generate detections.

Figure 6.9 presents the detail about the spatial confidence-aware message fusion module. Given the received messages $\{\mathcal{P}_{j\to i}^{(k)}, j \in \mathcal{N}_i\}$, each agent i attentively augments the features with the received messages at each location. And the request map $\mathbf{R}_j^{(k)}$ in the received message is firstly decoded to the confidence map $\mathbf{C}_j^{(k)}$ via a point-wise minus. Then the per-location multi-head attention are separately operated at each location in parallel, it takes the features and the corresponding confidence scores as input, and outputs the augmented features.

Sensor positional encoding represents the physical distance between each agent's sensor and its observation. Sensor positional encoding is conditioned on the physical distance between the known sensor coordinates and each BEV gird's coordinate in the 3D physical space. It is introduced to provide spatial prior, as the smaller the sensing distance is, the clear the observation would be. Mathematically, similar to the position encoding in [49], our sensor positional encoding is given by $SPE_{(dis,2p)} = sin(dis/10000^{2p/D})$, $SPE_{(dis,2p+1)} = cos(dis/10000^{2p/D})$ where dis is the physical distance, p is the dimension, D is the total channel dimension of the BEV feature map, sin and cos denote the sine and cosine functions. The features are then summed up with the positional encoding of each location before inputting to the transformer.

Compared to existing fusion modules that do not use attention mechanism [1] or only use agent-level attentions [12], the per-location attention mechanism adopted by the proposed fusion emphasizes the location-specific feature interactions. It makes the feature fusion more targeted. Compared to the methods that also use the per-location attention-based fusion module [2, 10, 48], the proposed fusion module leverages multi-head attention with two extra priors, including spatial confidence map and sensing distances. Both assist attention learning to prefer high quality and critical features.

Detection decoder. The detection decoder decodes features into objects, including class and regression output. Given the feature map at the kth communication round $\mathcal{F}_i^{(k)}$, the detection decoder $\Phi_{\text{dec}}(\cdot)$ generate the detections of ith agent by

$$\widehat{\mathcal{O}}_i^{(k)} = \Phi_{\text{dec}}(\mathcal{F}_i^{(k)}) \in \mathbb{R}^{H \times W \times 7},$$

where each location of $\widehat{\mathcal{O}}_i^{(k)}$ represents a rotated box with class $(c, x, y, h, w, \cos\alpha, \sin\alpha)$, denoting class confidence, position, size and angle. The objects are the final output of the proposed collaborative perception system. Note that $\widehat{\mathcal{O}}_i^{(0)}$ denotes the detections without collaboration.

Training Details and Loss Functions. Algorithm 1 presents the pipeline of our multi-round spatial confidence-aware collaborative perception system. To train the overall system, we supervise two tasks: spatial confidence generation and object detection at each round. As mentioned before, the functionality of the spatial confidence generator is the same as the classification in the detection decoder. To promote parameter efficiency, our spatial confidence generator reuses the parameters of the detection decoder. For the multi-round settings, each round is supervised with one detection loss, the overall loss is $L = \sum_{k=0}^{K} \sum_{i}^{N} L_{\text{det}}\left(\widehat{\mathcal{O}}_i^{(k)}, \mathcal{O}_i\right)$, where $\mathcal{O}_i$ is the ith agent's ground-truth objects, L_{det} is the detection loss [50].

To adapt to multi-round communication and dynamic bandwidth, we train the model under various communication settings with curriculum learning strategy [51]. We first gradually increase the communication bandwidth and round; and then, randomly sample bandwidth and round to promote robustness. Through this training strategy, a single model can perform well at various communication conditions.

6.3.4 Experimental Results

Datasets. Our experiments covers four datasets, both real-world and simulation scenarios, two types of agents (cars and drones) and two types of sensors (LiDAR and cameras). Specifically, we conduct camera-only 3D object detection in the setting of V2X-communication aided autonomous driving on OPV2V dataset [10], camera-only 3D object detection in the setting of drone swarm on the proposed CoPerception-UAVs dataset, and LiDAR-based 3D object detection on DAIR-V2X dataset [11] and V2X-Sim dataset [9].

OPV2V. OPV2V [10] is a vehicle-to-vehicle collaborative perception dataset, co-simulated by OpenCDA [10] and Carla [52]. It includes 12K frames of 3D point clouds and RGB images with 230K annotated 3D boxes. The perception range is

Algorithm 1 Multi-round spatial confidence-aware collaborative perception system

```
Define $N$ as the number of agents , $K$ as communication round
# Initialization
for $i = 1, 2, \ldots, N$, do
    $\mathcal{F}_i^{(0)} = \Phi_{\mathrm{enc}}(\mathcal{X}_i) \in \mathbb{R}^{H \times W \times D}$    ▷ Extract intermediate feature
end for
for $k = 0, 1, \ldots, K-1$, do
    for $i = 1, 2, \ldots, N$, do # Each agent is computing individually
        $\mathbf{C}_i^{(k)} = \Phi_{\mathrm{generator}}(\mathcal{F}_i^{(k)}) \in \mathbb{R}^{H \times W}$    ▷ Generate spatial confidence map
        for $j = 1, 2, \ldots, N$, do
            # Message packing
            $\mathbf{R}_i^{(k)} = 1 - \mathbf{C}_i^{(k)} \in \mathbb{R}^{H \times W}$    ▷ Pack request map
            if $k = 0$ then
                $\mathbf{M}_{i \to j}^{(k)} = \Phi_{\mathrm{select}}(\mathbf{C}_i^{(k)}) \in \{0, 1\}^{H \times W}$    ▷ Select critical areas
            else
                $\mathbf{M}_{i \to j}^{(k)} = \Phi_{\mathrm{select}}(\mathbf{C}_i^{(k)} \odot \mathbf{R}_j^{(k-1)}) \in \{0, 1\}^{H \times W}$    ▷ Select requested areas
            end if
            $\mathcal{Z}_{i \to j}^{(k)} = \mathbf{M}_{i \to j}^{(k)} \odot \mathcal{F}_i^{(k)} \in \mathbb{R}^{H \times W \times D}$    ▷ Pack spatially sparse features
            # Communication graph learning
            if $k = 0$ then
                $\mathbf{A}_{i \to j}^{(k)} = 1$    ▷ Broadcast critical features and request
            else
                $\mathbf{A}_{i \to j}^{(k)} = \max_{h,w} \left(\mathbf{M}_{i \to j}^{(k)}\right)_{h,w} \in \{0, 1\}$    ▷ Communicate only when necessary
            end if
        end for
        # Communication
        Send $\mathcal{P}_{i \to j} = \left(\mathcal{Z}_{i \to j}^{(k)}, \mathbf{R}_i^{(k)}\right)$ to other agents
        Receive $\{\mathcal{P}_{j \to i} = \left(\mathcal{Z}_{j \to i}^{(k)}, \mathbf{R}_j^{(k)}\right), j \neq i\}$ from other agents
        # Message fusion
        $\mathcal{F}_i^{(k+1)} = f_{\mathrm{fuse}}\left(\mathcal{F}_i^{(k)}, \{(\mathcal{Z}_{j \to i}^{(k)}, \mathbf{R}_j^{(k)}), j = 1, 2, ..., N\}\right) \in \mathbb{R}^{H \times W \times D}$
    end for
    Store $\mathcal{F}_i^{(k+1)}$ and $\{\mathbf{R}_j^{(k)}, j \neq i\}$ for the next round
end for
$\mathcal{O}_i^{(K)} = \Phi_{\mathrm{dec}}(\mathcal{F}_i^{(K)})$    ▷ Output the final detections
```

40 m × 40 m. For camera-only 3D object detection task on OPV2V, we implement the detector following CADDN [22]. The input front-view image size is (416, 160). The front-view input feature map is transformed to BEV with resolution 0.5m/pixel.

V2X-Sim. V2X-Sim [9] is a vehicle-to-everything collaborative perception dataset, co-simulated by SUMO [53] and Carla, including 10K frames of 3D LiDAR point clouds and 501K 3D boxes. The perception range is 64 m × 64 m. For LiDAR-based 3D object detection task, our detector follows MotionNet [54]. We discretize 3D points into a BEV map with size (256, 256, 13) and the resolution is 0.4m/pixel in length and width, 0.25 m in height.

CoPerception-UAVs. To enrich the collaborative perception datasets, we consider the swarm of unmanned aerial vehicles (UAV) and propose a UAV-swarm-based collaborative perception dataset: CoPerception-UAVs, co-simulated by

AirSim [55] and Carla [52], including 131.9K aerial images and 1.94M 3D boxes. The perception range is 200 m × 350m. For the camera-only 3D object detection task on CoPerception-UAVs, our detector follows DVDET [8]. The input aerial image size is (800, 450). The aerial-view input feature map is transformed to BEV with the resolution of 0.25 m/pixel, and the size is (192, 352); see more details in Appendix.

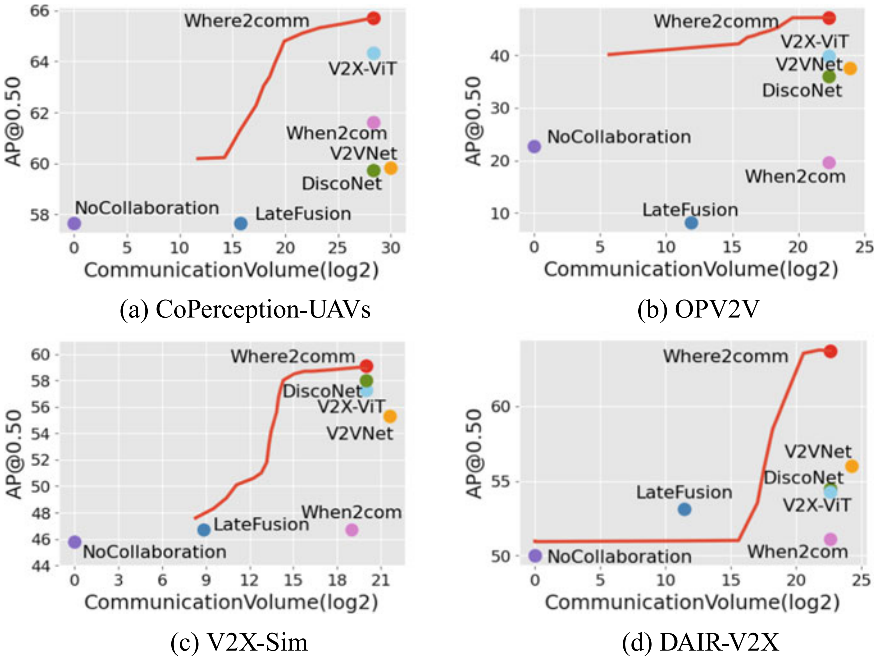
DAIR-V2X. DAIR-V2X [11] is the only public real-world collaborative perception dataset. Each sample contains two agents: a vehicle and an infrastructure, with 3D annotations. The perception range is 201.6 m × 80 m. Originally DAIR-V2X does not label objects outside the camera's view, we relabel all objects to cover 360-degree detection range. We complement several intermediate fusion-based baselines on DAIR-V2X to comprehensively validate our method on real data. For LiDAR-based 3D object detection task, our detector follows PointPillar [21]. We represent the field of view into a BEV map with size (200, 504, 64) and the resolution is 0.4 m/pixel in length and width.

Evaluation metrics. The detection results are evaluated by Average Precision (AP) at Intersection-over-Union (IoU) threshold of 0.50 and 0.70. The communication results count the message size by byte in log scale with base 2. To compare communication results straightforward and fair, we do not consider any extra data/feature/model compression. Mathematically for the selected sparse feature map $\mathcal{Z}_{i\to j}^{(k)} = \mathbf{M}_{i\to j}^{(k)} \odot \mathcal{F}_i^{(k)} \in \mathbb{R}^{H\times W\times D}$, the communication volume is

$$\log_2\left(|\mathbf{M}_{i\to j}^{(k)}| \times D \times 32/8\right), \tag{6.11}$$

where $|\cdot|$ denotes the L0 norm counting the non-zero elements in the binary selection matrix, this is, the total spatial girds need to be transmitted, and for each feature point D denotes the channel dimension, 32 is multiplied as float32 data type is used to represent each number, 8 is divided as the metric byte is used.

Benchmark comparison. Figure 6.10 compares the proposed `Where2comm` with the previous methods in terms of the trade-off between detection performance (AP@IoU=0.50) and communication bandwidth; also see exact values in Table 3 of Appendix. We consider single-agent detection without collaboration ($\widehat{\mathcal{O}}_i^{(0)}$), When2com [12], V2VNet [1], DiscoNet [2], V2X-ViT [48] and late fusion, where agents directly exchange the detected 3D boxes. The red curve comes from a single `Where2comm` model evaluated at varying bandwidths. We see that the proposed `Where2comm`: (i) achieves a far-more superior perception-communication trade-off across all the communication bandwidth choices and various collaborative perception tasks, including camera-only 3D object detection from aerial view and car front view, and LiDAR-based 3D object detection; (ii) achieves significant improvements over previous state-of-the-arts on both real-world (DAIR-V2X) and simulation scenarios, improves the SOTA performance by 7.7% on DAIR-V2X, 6.62% on CoPerception-UAVs, 25.81% on OPV2V, 1.9% on V2X-Sim; (iii) achieves the same detection performance of previous state-of-the-arts with extremely less communication volume: 5128 times less on CoPerception-UAVs, more than 100K times less on OPV2V, 55 times less on V2X-Sim, 105 times less on DAIR-V2X.



(a) CoPerception-UAVs

(b) OPV2V

(c) V2X-Sim

(d) DAIR-V2X

Fig. 6.10 `Where2comm` achieves consistently superior performance-bandwidth trade-off on all the three collaborative perception datasets, e.g., `Where2comm` achieves *5,000* times less communication volume and still outperforms When2com on CoPerception-UAVs dataset. The entire red curve comes from a single `Where2comm` model evaluated at varying bandwidths

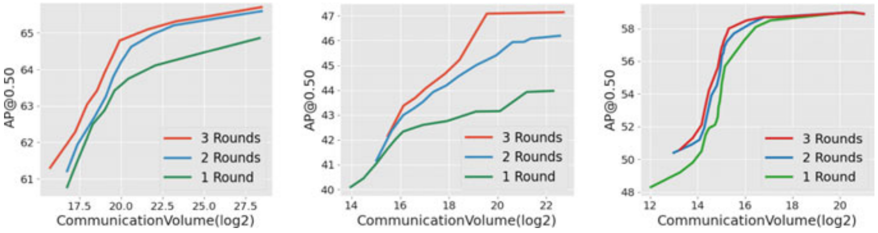
Table 6.1 presents the overall performance on the four datasets, CoPerception-UAVs, OPV2V [10], V2X-Sim1.0 [2] and DAIR-V2X [11]. For this LiDAR-based 3D object detection task, our detector follows PointPillar [21]. We see that `where2comm` consistently achieves significant improvements over previous methods on all the benchmarks.

Multi-round evaluation. Figure 6.11 presents the performances of `Where2comm` at communication rounds ranging from 1 to 3. Each curve comes from a single `Where2comm` model with a certain communication round evaluated at varying bandwidths. Results show that 1 communication round is good, more rounds are even better. Multi-round communication steadily improves the performance-bandwidth trade-off across all three datasets, reflecting its effectiveness and robustness. This encourages the agents to actively collaborate without worrying the performance degradation. This also validates that `Where2comm` can well work at various communication bandwidths and rounds.

Robustness to localization noise. We follow the localization noise setting in V2VNet and V2X-ViT (Gaussian noise with a mean of 0m and a standard deviation of 0m-0.6m) and conduct experiments on all the three datasets to validate the robustness against realistic localization noise. *Where2comm* is more robust to the

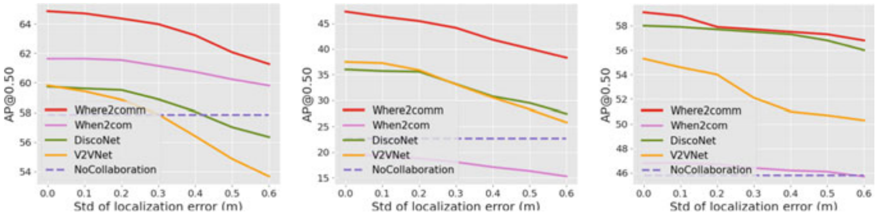
Table 6.1 Overall performance on CoPerception-UAVs, OPV2V, V2X-Sim and DAIR-V2X. Comm denotes the communication volume calculated with Eq. (6.11)

Dataset	CoPerception-UAVs		OPV2V		V2X-Sim1.0		DAIR-V2X	
Method/ Metric	Comm	AP@0.50/0.70	Comm	AP@0.50/0.70	Comm	AP@0.50	Comm	AP@0.50/0.70
No Collaboration	0.00	57.67/29.52	0.00	22.65/9.09	0.00	45.80	0.00	50.03/43.57
Late Fusion	15.77	53.12/37.88	11.87	8.24/3.84	8.83	46.70	11.45	53.12/37.88
When2com	28.37	61.63/33.55	22.28	19.69/8.29	20.00	46.70	22.62	51.12/36.17
V2VNet	29.95	59.82/33.14	23.87	37.47/14.67	21.58	55.30	24.21	56.01/42.25
V2X-ViT	28.37	59.12/41.57	22.28	39.82/16.43	20.00	57.30	22.62	54.26/43.35
DiscoNet	28.37	59.74/29.71	22.28	36.00/12.50	20.00	58.00	22.62	54.29/44.88
Where2comm	11.76	60.19/34.94	5.67	40.11/15.36	6.70	47.60	11.40	50.98/39.11
	14.27	60.23/34.93	15.49	42.15/16.09	8.29	49.10	15.58	51.01/39.10
	15.73	61.30/35.29	16.13	43.37/16.84	9.52	50.60	17.03	53.53/40.70
	17.96	63.04/36.10	17.04	44.07/17.15	10.41	51.80	17.53	55.84/42.44
	19.04	63.94/37.16	17.86	44.68/17.77	11.10	54.20	18.19	58.46/44.46
	21.62	65.10/38.98	18.43	45.23/18.02	12.27	56.60	20.56	63.54/48.78
	23.33	65.32/39.25	18.92	46.04/18.23	12.80	57.00	21.78	63.76/48.94
	25.31	65.46/39.27	18.92	46.04/18.23	13.98	58.90	22.35	63.71/48.89
	28.48	65.71/39.38	22.71	47.14/19.07	20.00	59.10	22.62	63.71/48.93



(a) CoPerception-UAVs (b) OPV2V (c) V2X-Sim

Fig. 6.11 More communication rounds continuously improve performance-bandwidth trade-off



(a) CoPerception-UAVs (b) OPV2V (c) V2X-Sim

Fig. 6.12 Robustness to localization error. Gaussian noise with zero mean and varying std is introduced. *Where2comm* consistently outperforms previous SOTAs and No Collaboration

localization noise than previous SOTAs. Figure 6.12 shows the detection performances as a function of localization noise level in CoPerception-UAVs, OPV2V and V2X-Sim datasets, respectively We see: (i) overall the collaborative perception performance degrades with the increasing localization noise, while *where2comm* outperforms previous SOTAs (When2com, V2VNet,DiscoNet) under all the localization noise. (ii) *where2comm* keeps being superior to *No Collaboration* while V2VNet fails when noise is over 0.4 m and DiscoNet fails when noise is over 0.5m on CoPerception-UAVs. The reasons are: (i) the powerful transformer architecture in fusion module attentively select the most suitable collaborative feature; (ii) the spatial confidence map helps filter out noisy features, these two designs work together to mitigate noise localization distortion effects.

Visualization of spatial confidence map. Figure 6.13 illustrates how `Where2comm` is empowered by the proposed spatial confidence map. In the scene, Drone 1's view is occluded by a tall building. With Drone 2's help, Drone 1 is able to detect through occlusion. Figure 6.13a–d shows Drone 1's observation, spatial confidence map (6.6), binary selection matrix (6.7), and ego attention weight (6.9). Figure 6.13f–h shows Drone 2's observation and message sent to Drone 1, including the request map (opposite of confidence map) and the sparse feature map, achieving efficient communication. Figure 6.13 (i) shows the attention weight for Drone 1 to fuse Drone 2's messages, which is sparse, yet highlights the objects' positions. Figure 6.13e, j compares the detection results before and after the collaboration with

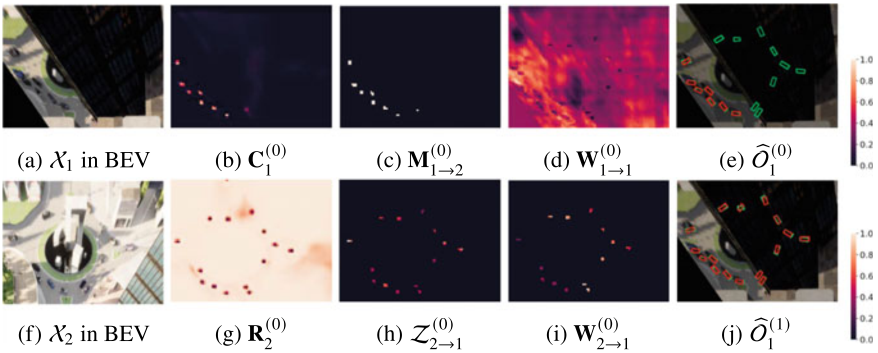

(a) $\mathcal{X}_1$ in BEV
(b) $\mathbf{C}_1^{(0)}$
(c) $\mathbf{M}_{1\rightarrow 2}^{(0)}$
(d) $\mathbf{W}_{1\rightarrow 1}^{(0)}$
(e) $\widehat{\mathcal{O}}_1^{(0)}$
(f) $\mathcal{X}_2$ in BEV
(g) $\mathbf{R}_2^{(0)}$
(h) $\mathcal{Z}_{2\rightarrow 1}^{(0)}$
(i) $\mathbf{W}_{2\rightarrow 1}^{(0)}$
(j) $\widehat{\mathcal{O}}_1^{(1)}$
1.0
0.8
0.6
0.4
0.2
0.0

Fig. 6.13 Visualization of collaboration between Drone 1 and Drone 2 on CoPerception-UAVs dataset, including spatial confidence map ($\mathbf{C}_1^{(0)}$), selection matrix ($\mathbf{M}_{1\rightarrow 2}^{(0)}$), message ($\{\mathbf{R}_2^{(0)}, \mathcal{Z}_{2\rightarrow 1}^{(0)}\}$) in the communication module, attention weight in the fusion module ($\mathbf{W}_{1\rightarrow 1}^{(0)}$,$\mathbf{W}_{2\rightarrow 1}^{(0)}$), and Drone 1's detection results before ($\widehat{\mathcal{O}}_1^{(0)}$) and after ($\widehat{\mathcal{O}}_1^{(1)}$) collaboration. Green and red boxes denote ground-truth and detection, respectively. The objects occluded by a tall building can be detected through transmitting spatially sparse, yet perceptually critical message

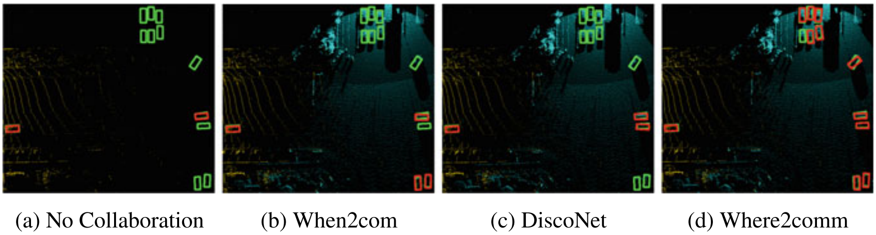
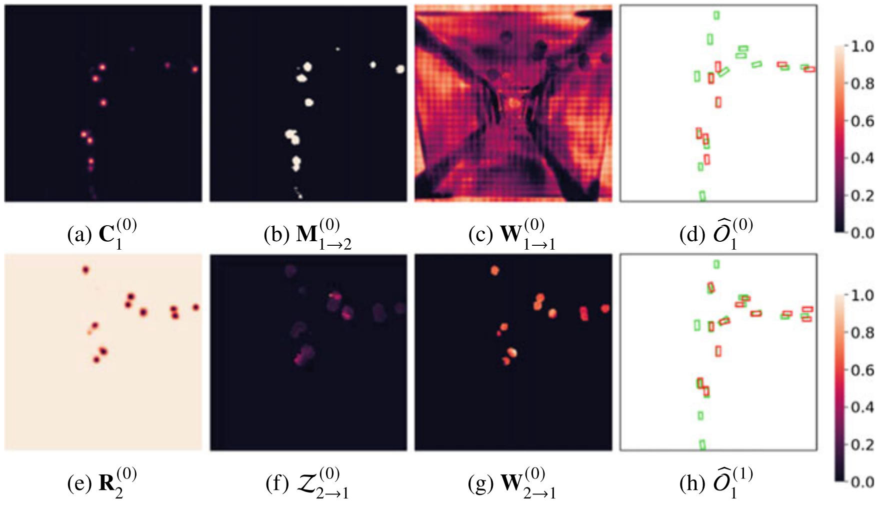


Fig. 6.14 `Where2comm` qualitatively outperforms When2com and DiscoNet in DAIR-V2X dataset. Green and red boxes denote ground-truth and detection, respectively. Yellow and blue denote the point clouds collected from vehicle and infrastructure, respectively

Drone 2. We see that the proposed spatial confidence map contributes to spatially sparse, yet perceptually critical message, which effectively helps Drone 1 detect occluded objects.

Visualization of detection results. Figure 6.14 shows that compared to *No Collaboration*, *When2com* and *DiscoNet*, `Where2comm` is able to achieves more complete and accurate detection results. The reason is that *When2com* employs a scalar to denote the agent-to-agent attention, which cannot distinguish which spatial area is more informative; *DiscoNet* employs a MLP-based fusion weight learning, which cannot well capture the complex collaboration attention; while `Where2comm` can zoom in to critical spatial areas in a cell-level resolution and leverage the spatial confidence map and sensing distances as priors to achieve more comprehensive fusion.

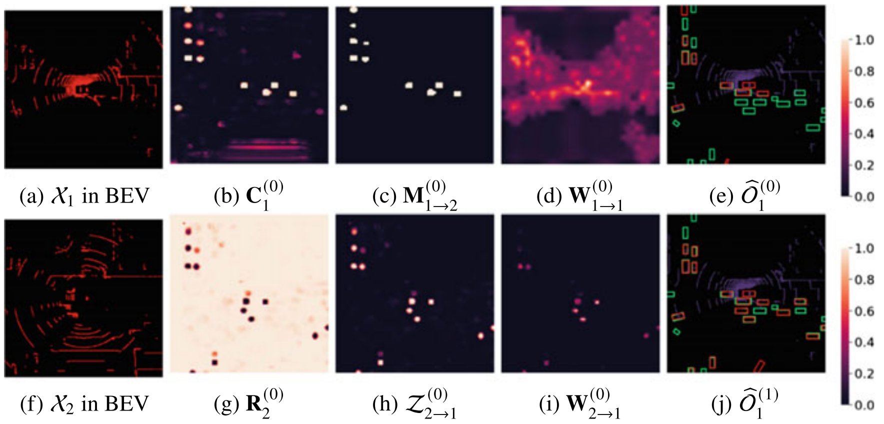
Visualization of collaboration in OPV2V and V2X-Sim. Figures 6.15 and 6.16 illustrates how `Where2comm` is empowered by the proposed spatial confidence map on OPV2V and V2X-Sim dataset. In the scene, with Vehicle 2's help, Vehicle 1 is



(a) $\mathbf{C}_1^{(0)}$ (b) $\mathbf{M}_{1\to2}^{(0)}$ (c) $\mathbf{W}_{1\to1}^{(0)}$ (d) $\widehat{\mathcal{O}}_1^{(0)}$

(e) $\mathbf{R}_2^{(0)}$ (f) $\mathcal{Z}_{2\to1}^{(0)}$ (g) $\mathbf{W}_{2\to1}^{(0)}$ (h) $\widehat{\mathcal{O}}_1^{(1)}$

Fig. 6.15 Visualization of collaboration between Vehicle 1 and Vehicle 2 on OPV2V dataset, including spatial confidence map ($\mathbf{C}_1^{(0)}$), selection matrix ($\mathbf{M}_{1\to2}^{(0)}$), message ($\{\mathbf{R}_2^{(0)}, \mathcal{Z}_{2\to1}^{(0)}\}$) in the communication module, attention weight in the fusion module ($\mathbf{W}_{1\to1}^{(0)}$,$\mathbf{W}_{2\to1}^{(0)}$), and Vehicle 1's detection results before ($\widehat{\mathcal{O}}_1^{(0)}$) and after ($\widehat{\mathcal{O}}_1^{(1)}$) collaboration. Green and red boxes denote ground-truth and detection, respectively. The objects occluded can be detected through transmitting spatially sparse, yet perceptually critical message



(a) $\mathcal{X}_1$ in BEV (b) $\mathbf{C}_1^{(0)}$ (c) $\mathbf{M}_{1\to2}^{(0)}$ (d) $\mathbf{W}_{1\to1}^{(0)}$ (e) $\widehat{\mathcal{O}}_1^{(0)}$

(f) $\mathcal{X}_2$ in BEV (g) $\mathbf{R}_2^{(0)}$ (h) $\mathcal{Z}_{2\to1}^{(0)}$ (i) $\mathbf{W}_{2\to1}^{(0)}$ (j) $\widehat{\mathcal{O}}_1^{(1)}$

Fig. 6.16 Visualization of collaboration between Vehicle 1 and Vehicle 2 on V2X-Sim dataset, including spatial confidence map ($\mathbf{C}_1^{(0)}$), selection matrix ($\mathbf{M}_{1\to2}^{(0)}$), message ($\{\mathbf{R}_2^{(0)}, \mathcal{Z}_{2\to1}^{(0)}\}$) in the communication module, attention weight in the fusion module ($\mathbf{W}_{1\to1}^{(0)}$,$\mathbf{W}_{2\to1}^{(0)}$), and Drone 1's detection results before ($\widehat{\mathcal{O}}_1^{(0)}$) and after ($\widehat{\mathcal{O}}_1^{(1)}$) collaboration. Green and red boxes denote ground-truth and detection, respectively. The objects occluded by a tall building can be detected through transmitting spatially sparse, yet perceptually critical message

able to detect the missed objects in the single view. Figure 6.15a–d shows Vehicle 1's spatial confidence map, binary selection matrix, ego attention weight, and the detection results by its own observation. Figure 6.15e–f shows Vehicle 2's message sent to Drone 1, including the request map (opposite of confidence map) and the sparse feature map, achieving efficient communication. Figure 6.15g shows the attention weight for Vehicle 1 to fuse Vehicle 2's messages, which is sparse, yet highlights the objects' positions. Figure 6.15d, h compares the detection results before and after the collaboration with Vehicle 2. We see that the proposed spatial confidence map contributes to spatially sparse, yet perceptually critical message, which effectively helps Vehicle 1 detect occluded objects.

6.3.5 Ablation Studies

Effect of Gaussian filter in perceptually critical area selection. Figure 6.17 compares two versions of the selection matrix (6.7) with and without Gaussian filter. We see that applying Gaussian filter improves the overall performance. The reason is that: (i) Gaussian filter could help filter out the outliers in the input map, selecting more robust critical regions; (ii) it considers the context, benefiting the independent feature selection at each certain location by providing more information.

Effect of components in spatial confidence-aware message fusion. Table 6.2 assesses the effectiveness of the proposed fusion with two priors. We see that: (i) per-location multi-head attention (MHA) outperforms the vanilla attention by 10.84% on OPV2V on AP@0.50, because MHA leverages information from multiple heads, better capturing cross-agent attention; and (ii) As two informative priors, both sensing position encoding (SPE) and spatial confidence map (SCM) can consistently improve the performance. Especially, the version with all three designs improves the detection performance by 22.06% on OPV2V on AP@0.50.

Fig. 6.17 Applying Gaussian filter improves performance

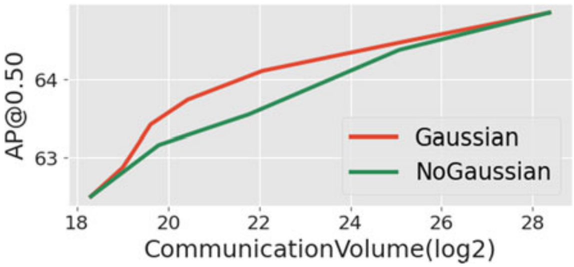


Table 6.2 Fusion component ablation study. Multi-head attention (MHA), sensor positional encoding (SPE) and spatial confidence map (SCM) all improves the performances. Results are reported in AP@0.50/AP@0.70

MHA	SPE	SCM	OPV2V	CoPerception-UAVs	V2X-Sim
			34.96/13.92	63.48/44.23	51.2/45.7
✓			38.75/13.28	63.99/44.46	57.3/50.8
✓	✓		39.82/16.43	64.34/46.86	59.1/52.0
✓	✓	✓	**47.30/19.30**	**64.83/47.62**	**59.1/52.2**

6.3.6 Further Thoughts on Communication Efficiency

In the above context, we have demonstrated the effectiveness of `Where2comm` in balancing communication cost and detection ability. The key of `Where2comm` is to explore the sparsity in the spatial dimension. In future, we can further explore the sparsity along the feature dimension and the temporal dimension. From the feature aspect, many feature channels could be redundant and can be significantly compressed. From the temporal aspect, it is probably unnecessary to communicate all the time because perceptual information in consecutive time stamps are too similar to bring news. Therefore, it could be interesting to determine critical time stamps, which would further significantly reduces the communication cost.

6.4 Chapter at a Glance

This chapter introduces an emerging field of collaborative 3D object detection. It enables multiple agents to share complementary detection information with each other and provides a new direction to fundamentally overcome occlusion and long-range issues in single-agent 3D detection. We then highlight several key challenges in collaborative 3D object detection, including communication constraints, pose errors, heterogeneous devices and security issues. We next focus on a fundamental trade-off between detection performance and communication bandwidth and introduce a latest communication-efficient collaborative 3D detection method.

References

1. Wang TH, Manivasagam S, Liang M, Yang B, Zeng W, Urtasun R (2020) V2vnet: vehicle-to-vehicle communication for joint perception and prediction. In: European conference on computer vision. Springer, pp 605–621
2. Li Y, Ren S, Wu P, Chen S, Feng C, Zhang W (2021) Learning distilled collaboration graph for multi-agent perception. In: Advances in neural information processing systems, vol 34

3. Chen S, Liu B, Feng C, Vallespi-Gonzalez C, Wellington CK (2021) 3d point cloud processing and learning for autonomous driving: Impacting map creation, localization, and perception. IEEE Signal Process Mag 38:68–86
4. Li Z, Barenji AV, Jiang J, Zhong RY, Xu G (2020) A mechanism for scheduling multi robot intelligent warehouse system face with dynamic demand. J Intell Manuf 31(2):469–480
5. Zaccaria M, Giorgini M, Monica R, Aleotti J (2021) Multi-robot multiple camera people detection and tracking in automated warehouses. In: 2021 IEEE 19th international conference on industrial informatics (INDIN). IEEE, pp 1–6
6. Scherer J, Yahyanejad S, Hayat S, Yanmaz E, Andre T, Khan A, Vukadinovic V, Bettstetter C, Hellwagner H, Rinner B (2015) An autonomous multi-UAV system for search and rescue. In: Proceedings of the first workshop on micro aerial vehicle networks, systems, and applications for civilian use, pp 33–38
7. Alotaibi ET, Alqefari SS, Koubaa A (2019) Lsar: Multi-UAV collaboration for search and rescue missions. IEEE Access 7:55817–55832
8. Hu Y, Fang S, Xie W, Chen S (2023) Aerial monocular 3d object detection. IEEE Robot Autom Lett 8(4):1959–1966
9. Li Y, An Z, Wang Z, Zhong Y, Chen S, Feng C (2022) V2X-Sim: a virtual collaborative perception dataset for autonomous driving. IEEE Robot Autom Lett 7
10. Xu R, Xiang H, Xia X, Han X, Li J, Ma J (2022) OPV2V: an open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication. In: ICRAD
11. Yu H, Luo Y, Shu M, Huo Y, Yang Z, Shi Y, Guo Z, Li H, Hu X, Yuan J, Nie Z et al (2022) DAIR-V2X: a large-scale dataset for vehicle-infrastructure cooperative 3d object detection. In: In proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)
12. Liu YC, Tian J, Glaser N, Kira Z (2020) When2com: multi-agent perception via communication graph grouping. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4106–4115
13. Liu YC, Tian J, Ma CY, Glaser N, Kuo CW, Kira Z (2020) Who2com: collaborative perception via learnable handshake communication. In: 2020 IEEE international conference on robotics and automation (ICRA). IEEE, pp 6876–6883
14. Zhou Y, Xiao J, Zhou Y, Loianno G (2022) Multi-robot collaborative perception with graph neural networks. IEEE Robot Autom Lett
15. Arnold E, Dianati M, de Temple R (2022) Cooperative perception for 3d object detection in driving scenarios using infrastructure sensors. IEEE Trans Intell Transp Syst 23:1852–1864
16. Lei Z, Ren S, Hu Y, Zhang W, Chen S (2022) Latency-aware collaborative perception. In: ECCV
17. Li Y, Zhang J, Ma D, Wang Y, Feng C (2022) Multi-robot scene completion: towards task-agnostic collaborative perception. In: Conference on robot learning (CoRL). PMLR
18. Runsheng X, Zhengzhong T, Xiang H, Shao W, Zhou B Ma J (2022) CoBEVT: cooperative bird's eye view semantic segmentation with sparse transformers. In: CoRL
19. Su S, Li Y, He S, Han S, Feng C, Ding C, Miao F (2022) Uncertainty quantification of collaborative detection for self-driving
20. Chen S, Liu B, Feng C, Vallespi-Gonzalez C, Wellington C (2021) 3d point cloud processing and learning for autonomous driving: impacting map creation, localization, and perception. IEEE Signal Process Mag 38(1):68–86
21. Lang AH, Vora S, Caesar H, Zhou L, Yang J, Beijbom O (2019) Pointpillars: fast encoders for object detection from point clouds. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 12689–12697
22. Reading C, Harakeh A, Chae J, Waslander SL (2021) Categorical depth distribution network for monocular 3d object detection. In: CVPR
23. Huang J, Huang G (2022) Bevdet4d: exploit temporal cues in multi-camera 3d object detection. abs/2203.17054
24. Schiegg FA, Llatser I, Bischoff D, Volk G (2021) Collective perception: a safety perspective. Sensors, 21(1):159

25. Shan M, Narula K, Wong YF, Worrall S, Khan M, Alexander P, Nebot E (2021) Demonstrations of cooperative perception: safety and robustness in connected and automated vehicle operations. Sensors 21(1):200
26. Lazaridou A, Peysakhovich A, Baroni M (2017) Multi-agent cooperation and the emergence of (natural) language. In: 5th international conference on learning representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference track proceedings. OpenReview.net
27. Leibo JZ, Zambaldi V, Lanctot M, Marecki J, Graepel T (2017) Multi-agent reinforcement learning in sequential social dilemmas. In: Larson K, Winikoff M, Das S, Durfee EH (eds) Proceedings of the 16th conference on autonomous agents and multiagent systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017. ACM, pp 464–473
28. Hernandez-Leal P, Kartal B, Taylor ME (2019) A survey and critique of multiagent deep reinforcement learning. Auton Agents Multi Agent Syst 33(6):750–797
29. Chen Q, Tang S, Yang Q, Fu S (2019) Cooper: cooperative perception for connected autonomous vehicles based on 3d point clouds. In: 2019 IEEE 39th international conference on distributed computing systems (ICDCS). IEEE, pp 514–524
30. Arnold E, Dianati M, de Temple R, Fallah S (2020) Cooperative perception for 3d object detection in driving scenarios using infrastructure sensors. IEEE Trans Intell Transp Syst
31. Miller A, Rim K, Chopra P, Kelkar P, Likhachev M (2020) Cooperative perception and localization for cooperative driving. In: 2020 IEEE international conference on robotics and automation (ICRA). IEEE, pp 1256–1262
32. Allig C, Wanielik G (2019) Alignment of perception information for cooperative perception. In: 2019 IEEE intelligent vehicles symposium (IV). IEEE, pp 1849–1854
33. Chen Q, Ma X, Tang S, Guo J, Yang Q, Fu S (2019) F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds. In: *P*roceedings of the 4th ACM/IEEE symposium on edge computing, pp 88–100
34. Emad Marvasti E, Raftari A, Emad Marvasti A, Fallah YP (2020) Bandwidth-adaptive feature sharing for cooperative lidar object detection. In: 2020 IEEE 3rd connected and automated vehicles symposium (CAVS). IEEE, pp 1–7
35. Emad Marvasti E, Raftari A, Emad Marvasti A, Fallah YP, Guo R, Lu H (2020) Cooperative lidar object detection via feature sharing in deep networks. In: 2020 IEEE 92nd vehicular technology conference (VTC2020-Fall). IEEE, pp 1–7
36. Marvasti EE, Raftari A, Marvasti AE, Fallah YP, Guo R, Lu H (2020) Feature sharing and integration for cooperative cognition and perception with volumetric sensors. arXiv:2011.08317
37. Mei J, Zheng K, Zhao L, Teng Y, Wang X (2018) A latency and reliability guaranteed resource allocation scheme for lte v2v communication systems. IEEE Trans Wireless Commun 17:3850–3860
38. Jiang D, Delgrossi L (2008) IEEE 802.11p: Towards an international standard for wireless access in vehicular environments. In: VTC Spring 2008 - IEEE vehicular technology conference, pp 2036–2040
39. Araniti G, Campolo C, Condoluci M, Iera A, Molinaro A (2013) Lte for vehicular networking: a survey. IEEE Commun Mag 51:148–157
40. Lei L, Kuang Y, Cheng N, Shen X, Zhong Z, Lin C (2016) Delay-optimal dynamic mode selection and resource allocation in device-to-device communications-part i: Optimal policy. IEEE Trans Veh Technol 65:3474–3490
41. Hu Y, Fang S, Lei Z, Zhong Y, Chen S (2022) Where2comm: Communication-efficient collaborative perception via spatial confidence maps. In: Advances in neural information processing systems
42. Vadivelu N, Ren M, Tu J, Wang J, Urtasun R (2021) Learning to communicate and correct pose errors. In: Kober J, Ramos F, Tomlin C (eds) Proceedings of the 2020 conference on robot learning, volume 155 of *P*roceedings of machine learning research. PMLR, pp 1195–1210. Accessed from 16–18 Nov 2021
43. Glaser N, Liu YC, Tian J, Kira Z (2021) Overcoming obstructions via bandwidth-limited multi-agent spatial handshaking. In: 2021 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 2406–2413

44. Xu R, Xiang H, Tu Z, Xia X, Yang MH, Ma J (2022) V2X-ViT: Vehicle-to-everything cooperative perception with vision transformer. In: Proceedings of the European conference on computer vision (ECCV)
45. Yuan Y, Cheng H, Sester M (2022) Keypoints-based deep feature fusion for cooperative vehicle detection of autonomous driving. IEEE Robot Autom Lett 7(2):3054–3061
46. Yuan Y, Sester M (2022) Leveraging dynamic objects for relative localization correction in a connected autonomous vehicle network. arXiv:2205.09418
47. Wang TH, Manivasagam S, Liang M, Yang B, Zeng W, Urtasun R (2020) V2vnet: vehicle-to-vehicle communication for joint perception and prediction. In: ECCV (2), Lecture notes in computer science, vol 12347. Springer, pp 605–621
48. Xu R, Xiang H, Tu Z, Xia X, Yang MH, Ma J (2022) V2X-ViT: vehicle-to-everything cooperative perception with vision transformer. In: ECCV
49. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, vol 30
50. Zhou X, Wang D, Krähenbühl P (2019) Objects as points. arXiv:1904.07850
51. Bengio Y, Louradour J, Collobert R, Weston J (2009) Curriculum learning. In: Proceedings of the 26th annual international conference on machine learning, pp 41–48
52. Dosovitskiy A, Ros G, Codevilla F, Lopez A, Koltun V (2017) Carla: an open urban driving simulator. In: Conference on robot learning. PMLR, pp 1–16
53. Krajzewicz D, Erdmann J, Behrisch M, Bieker L (2012) Recent development and applications of sumo-simulation of urban mobility. Int J Adv Syst Meas 5(3&4)
54. Wu P, Chen S, Metaxas DN (2020) Motionnet: joint perception and motion prediction for autonomous driving based on bird's eye view maps. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 11382–11392
55. Shah S, Dey D, Lovett C, Kapoor A (2018) Airsim: high-fidelity visual and physical simulation for autonomous vehicles. In: Field and service robotics. Springer, pp 621–635

Chapter 7
Enabling Robust SLAM for Mobile Robots with Sensor Fusion

Jianhao Jiao, Xiangcheng Hu, Xupeng Xie, Jin Wu, Hexiang Wei, Lu Fan, and Ming Liu

Abstract Simultaneous Localization and Mapping (SLAM) is a fundamental problem in robotics. Over the past three decades, researchers have made significant progress in solving the probabilistic SLAM problem by presenting various theoretical frameworks, efficient solvers, and complete systems. As the development of

This work was supported by Guangdong Basic and Applied Basic Research Foundation, under project 2021B1515120032, Foshan-HKUST Project no. FSUST20-SHCIRI06C, and Project of Hetao Shenzhen Hong Kong Science and Technology Innovation Cooperation Zone (HZQB-KCZYB-2020083), awarded to Prof. Ming Liu.

M. Liu (✉)
Robotics and Autonomous Systems, The Hong Kong University of Science and Technology, Nansha, Guangzhou 511400, Guangdong, China
e-mail: eelium@ust.hk

J. Jiao · X. Hu · X. Xie · J. Wu · H. Wei · M. Liu
Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, China
e-mail: jjiao@connect.ust.hk

X. Hu
e-mail: xhubd@connect.ust.hk

X. Xie
e-mail: xxieak@connect.ust.hk

J. Wu
e-mail: jwucp@connect.ust.hk

H. Wei
e-mail: hweiak@connect.ust.hk

M. Liu
HKUST Shenzhen-Hong Kong Collaborative Innovation Research Institute, Futian, Shenzhen, China

L. Fan
The Hong Kong Polytechnic University, Hong Kong, China
e-mail: cslfan@comp.polyu.edu.hk

R. Fan et al. (eds.), *Autonomous Driving Perception*, Advances in Computer Vision and Pattern Recognition, https://doi.org/10.1007/978-981-99-4287-9_7

autonomous robots (i.e., self-driving cars, legged robots) continues, SLAM systems have become increasingly popular for large-scale real-world applications. The evolution of SLAM is also often propelled by the emergence of new sensors or sensor combinations. This chapter provides an introduction to the commonly used sensors in mobile robots, followed by a comprehensive review of several classic SLAM systems from a modern perspective. Additionally, this chapter presents a real-world case of constructing a multi-sensor system and a challenging SLAM dataset, offering a valuable tutorial for researchers in developing their research platforms. Overall, this chapter aims to provide readers with a comprehensive guide to learn sensor fusion from theory to practice completely.

7.1 Introduction

7.1.1 Background

Benefitting from the progress in mechatronics, sensory technology, and artificial intelligence (AI), the evolution of autonomous robots in recent years is surprising. Many types of mobile robots, including service robots, drones, and self-driving cars, have moved beyond the environments of warehouses and manufacture plants into widespread deployment in industry and society, gradually changing our way of living or working. Quadrupedal robots such as the SpotMini[1] developed by *Boston Dynamics* have been used for surveillance, releasing laborers from repeated and dangerous works. Drones have been applied in racing [22], cinematography [43], and rescue [128] tasks, which highly differs from their initial stage [53]. The rapid development of self-driving cars [4, 39, 65, 100] is also amazing. These autonomous vehicles have been widely used in robotaxi, logistics, and environmental inspection. Figure 7.1 show some robots in our daily life.

It is important to acknowledge that the development of highly autonomous robots is a complex and challenging endeavor. There are numerous technical, ethical, and regulatory challenges that need to be addressed before we can fully realize the poten-

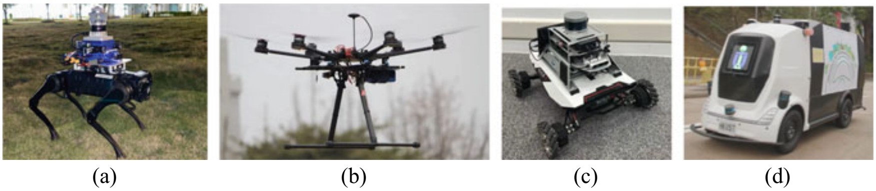


Fig. 7.1 Examples of robot including **a** the quadrupedal robot, **b** the drone, **c** the mobile robot, and **d** the autonomous vehicle for the last-mile delivery [65]

[1] https://www.bostondynamics.com/spot.

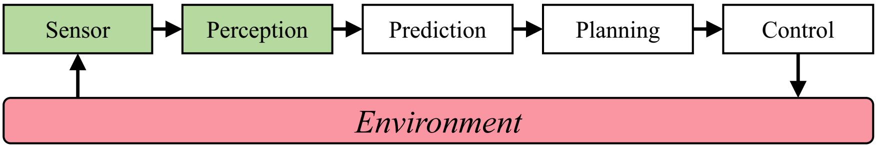


Fig. 7.2 Framework of a typical robotic system [97]. Modules marked in green (sensor and perception) are focused in this chapter

tial of this technology. Therefore, it is clear that there are still many works to be done in order to achieve highly autonomous robots. The current technologies have shown that navigation in a 2D indoor environment with a robot equipped with wheel encoders and a laser scanner has been addressed well, enabling the wide application of home cleaning robots. However, it is still challenging for robots to perform tasks without human intervention given higher performance requirements. For example, existing robots cannot conduct repeated inspection missions stably in environments where several dynamic agents such as a human crowd are present. This application case arises at least two issues, while existing technologies do not have clear solutions: *(1)* Dynamic environments tend to validate the general assumptions of existing localization and mapping systems which utilize static landmarks to estimate the ego-motion. *(2)* The pre-planned trajectory may be occluded by new-placed objects or moving agents. Robots should be capable of real-time recognizing and predicting agents' motion as well as generating a new path to avoid collision.

Sensing and **Understanding** the world of robots is very complex, with challenges spanning from calibration and simultaneous localization and mapping (SLAM) to scene understanding. A typical robotic system, as shown in Fig. 7.2, consists of multiple modules, with the *Sensor* and *Perception* modules addressing many of the problems related to "sensing" and "understanding". These modules play a fundamental role in higher-level navigation tasks such as decision-making [90] and path planning [19, 21, 98].

7.1.2 Summary of this Chapter

The field of sensor fusion is a rapidly evolving area, due to the significant progress of mobile robots and the commercial availability of sensors. Therefore, there is always a lack of literature to introduce the latest sensors and practical application examples from the implementation perspective.

In this chapter, we provide a broad overview of sensors and discuss the necessity of sensor fusion to enable robust perception in challenging environments. This part also reviews related works of modern SLAM systems. To enrich the content and make the chapter more interesting, we also introduce a detailed application of building a multi-senosr research platform and a challenging SLAM dataset.

7.1.3 Organization

The rest of this chapter is organized into the following sections:

- Section 7.2: We broadly summarize features of major sensors that are commonly used in mobile robots.
- Section 7.3: We introduce related works of SLAM systems ranging from LiDAR-based and vision-based approaches.
- Section 7.4: We provide a practical example of multi-sensor SLAM dataset, in which many implementation details are included.

7.2 Sensors

All sensors have a limited precision and probabilities to fail in some scenarios. Therefore, multi-sensor fusion is the key to robust perception. Different sensors can complement each other, and thus the system's perception capability is enhanced with sensor fusion. Regarding sensors' characteristics, it is useful to put sensors into two categories: **interoceptive** and **exteroceptive** sensors [3]. Their definitions are given as follow:

- **Interoceptive**: being stimuli arising within the body.
- **Exteroceptive**: being activated by stimuli received by an organism from outside.

Typical interoceptive sensors are the accelerometer (measures translational acceleration), gyroscope (measures angular rate), and rotary encoder (measures angular rate). Typical exteroceptive sensors are camera and time-of-flight transmitter/receiver (e.g., RGB-D camera, Radio Detection And Ranging (RaDAR), Light Detection And Ranging (LiDAR), and Global Navigation Satellite System (GNSS) transmitter/receiver). We briefly review the main features and related works of these sensors in the following sections.

7.2.1 Interoceptive Sensors

7.2.1.1 IMU

An Inertial Measurement Unit (IMU) is an electronic device that is composed of accelerometers, gyroscopes, and sometimes magnetometers to measure a body's acceleration and angular rate (see Fig. 7.3). IMUs are typically used to maneuver aircraft (an attitude and heading reference system), including unmanned aerial vehicles (UAVs), among many others, and spacecraft, including satellites and landers. The key advantage is that IMUs are seldom affected by external environments.

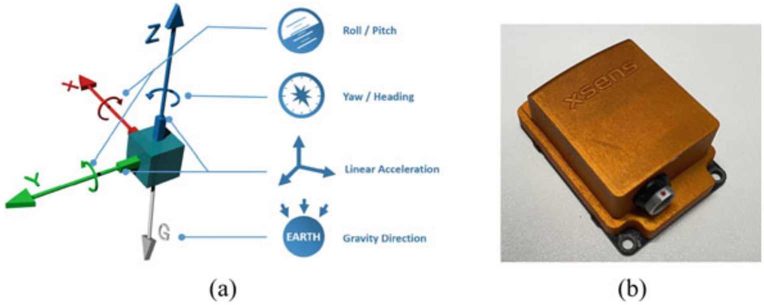
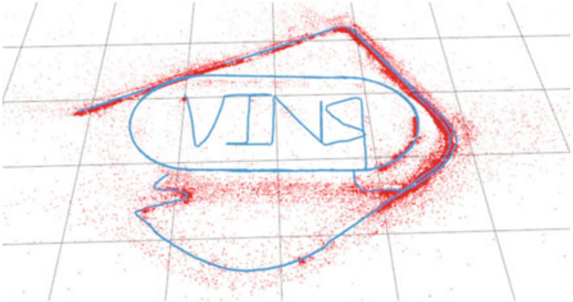


Fig. 7.3 **a** The principle of IMU. **b** An IMU product: Xsens IMU [113]

Fig. 7.4 An example of VINS-Mono's results [78]: (blue) trajectory; (red) 3D landmarks



The commonly used IMUs on robots are the microelectromechanical (MEMS) IMUs, which are both cheap and tiny while keeping with promising performance.

A major disadvantage of using IMUs is that they typically suffer from accumulated error. The navigation system is continually integrating accelerometer data twice to compute the position, and gyroscope data once to track the orientation. The output high-frequency gyroscope and accelerometer measurements suffer from a Gaussian noise and bias [26]. Any errors, however small, are accumulated, yielding drift over time. Small errors in measuring acceleration and angular velocity may be integrated into large errors in position or orientation, existing as systematic errors [131]. Thus, the position needs to be periodically corrected with the input of another navigation system. Consequently, inertial sensors are inaccurate and unsuitable for positioning applications over an extended period of time and are usually utilized to supplement other navigation systems.

As emphasized by Barfoot [3], "In most cases, the best state estimation concepts make use of both interoceptive and exteroceptive measurements." IMUs are often fused with cameras [78] and LiDARs [113] in the Extended Kalman Filter (EKF) or optimization framework. The VINS-Mono [78] is one of the most popular visual-inertial solutions, and an example result is shown in Fig. 7.4. Besides pose estimation and localization, IMU-centric sensor systems have been demonstrated with several novel applications, including object tracking [23, 81], semantic mapping [87], and temporal calibration [82].

7.2.1.2 GNSS

The Global Navigation Satellite System (GNSS) refers to a constellation of satellites providing space signals transmitting positioning and timing data to GNSS receivers. The receivers then use this data to determine location. By definition, GNSS provides global coverage. Through the receiver, any device can use the GNSS to locate its global position. Existing GNSS includes the China's BeiDou navigation satellite system (BeiDou), European Union's Galileo positioning system (GALILEO), the USA's NAVSTAR global positioning system (GPS), Russia's global navigation satellite system (GLONASS), and India's NavIC and Japan's Quasi-Zenith regional satellite system (QZSS) [25].

Like the IMU, tiny GNSS devices have been widely embedded into consumer-level devices such as phones and electronic wristbands. But the GNSS has several limitations: *(1)* hard to initialize (needs enough satellites to be found); *(2)* noisy and inaccurate measurements ($\geq 10cm$); and *(3)* low frequency (typically $1 - 100$Hz). This is because, in practical applications, the GNSS commonly suffers from multiple sources of errors and influences, including the message transmission delay through the atmospheric lay, the reflection of signals on multiple surfaces (e.g., localization among buildings), ephemeris errors, and the uncertainties on the satellite's position [105]. Therefore, the ideal case is to use the GNSS in outdoor open-field environments. In recent years, differential GNSS (DGPS) and real-time kinematic GNSS (RTK-GNSS) were also developed to enhance the GNSS's accuracy and allow for localization within the order of decimeters or even centimeters in well-conditioned environments. Figure 7.5 illustrates the principle of GNSS, multilateration measurement of GNSS, and a RTK-GNSS product.

Fusing GNSS signals with IMU measurements via. the EKF is a popular solution, and therefore becomes the Inertial Navigation System (INS). Robotic researchers also explored the potential of the GNSS on camera- or LiDAR-based systems and thus achieved the hybrid indoor-outdoor navigation systems [16, 120]. The global position provided by the GNSS is important to correct accumulative drift.

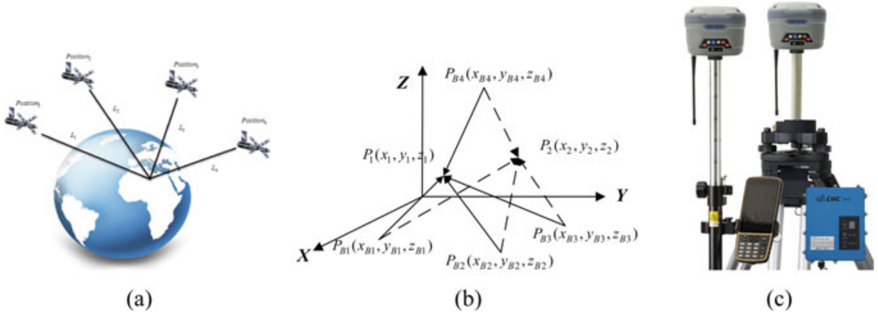

Z
$P_{B4}(x_{B4}, y_{B4}, z_{B4})$
$P_1(x_1, y_1, z_1)$
$P_2(x_2, y_2, z_2)$
Y
$P_{B3}(x_{B3}, y_{B3}, z_{B3})$
X
$P_{B1}(x_{B1}, y_{B1}, z_{B1})$
$P_{B2}(x_{B2}, y_{B2}, z_{B2})$
(a)
(b)
(c)

Fig. 7.5 Measurement system: **a** Principle of GNSS, **b** multilateration measurement, and **c** A RTK-GNSS product

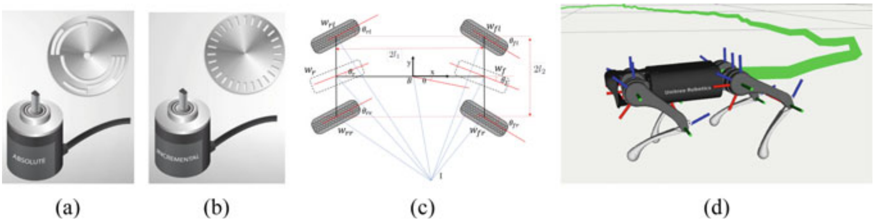


Fig. 7.6 **a** The absolute encoder. **b** The incremental encoder. **c** The wheel odometry. **d** The leg odometry

7.2.1.3 Rotary Encoder

The rotary encoder measures the absolute angular position (e.g., joints for legged robots) or relative angular position (e.g., wheels for ground vehicles) of robots' shaft. The output of encoder is continually processed into odometry, such as leg odometry for quadrupedal robots [112] and wheel odometry for ground vehicles [55, 108], as shown in Fig. 7.6.

Regarding SLAM, the integrated odometry provides a lower bound of error to a state estimator in degenerate environments, such as long tunnels and open space. Several teams in the DARPA subterranean challenge have verified the effectiveness of using encoders: the stability and robustness of a SLAM framework have been significantly improved in challenging scenarios [49, 52, 86]. Rotary encoders are also used to enforce the robustness of zero velocity update (ZUPT) [50, 107]. Furthermore, the encoders offer scale information, which is desirable for the monocular camera. An existing work has shown that encoder data is incorporated with a monocular visual odometry [108] on a ground vehicle.

However, rotary encoders including the leg odometer and wheel odometer suffer from accumulated drifts with the existence of contact points slippage which needs to be modeled. Another error source arises from intrinsics (such as the leg length and wheel radius) that change over time due to factors such as the increasing robot payload and non-rigid deformation. Recent research has shown that online kinematic calibration can significantly improve the accuracy of SLAM in online applications [55, 112].

7.2.2 Exteroceptive Sensors

7.2.2.1 Camera

Cameras are designed to imitate the output of the human visual system. The front-end optics capture light emitted or reflected by an object in the 3D world through the optical center and project it onto the camera's 2D imaging plane. The intensities of light are linearly transformed and restored as an image, with each element referred to as a *pixel*. These images are incredibly rich in texture and provide both spatial and qualitative information, which has captured the attention of researchers studying *computer vision problems* [1, 29].

For robots to interact with the world effectively, they often require 3D perception capabilities. Camera calibration is the first challenge that needs to be addressed, and its objective is to estimate the geometric model that describes the camera projection process. Intrinsic calibration estimates the *focal length*, *pixel origin*, and *distortion parameters* of a camera, while extrinsic calibration computes the relative rotation and translation among multiple cameras. In practical applications, both intrinsics and extrinsics are pre-computed using marker-based methods [121]—by moving a checkerboard with sufficient motion in front of the cameras—and sometimes are optimized online (known as online calibration).

Visual odometry (VO) or visual SLAM (VSLAM) algorithms are categorized based on how they process images. There are three categories: feature-based (indirect) methods, direct methods, and hybrid methods.

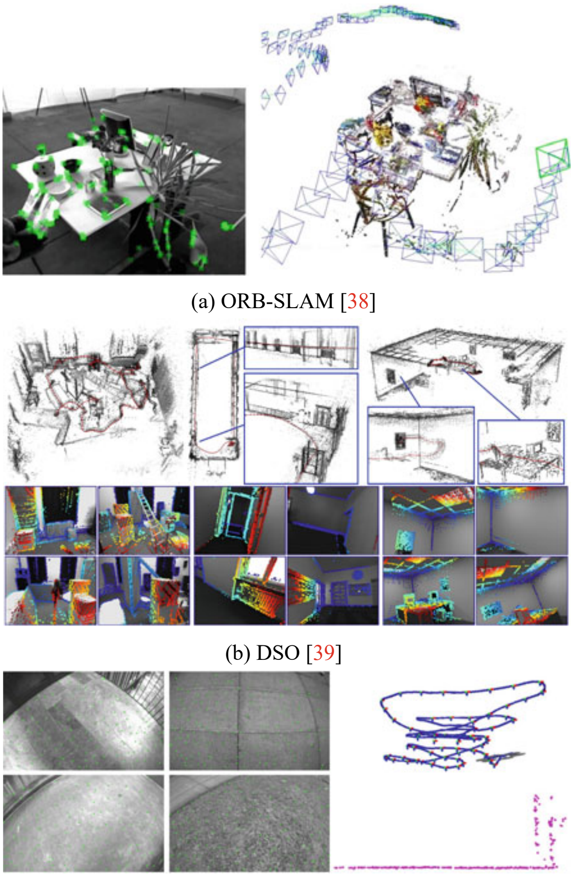
- Typical feature-based SLAM frameworks are the ORB-SLAM series [15, 71, 72], which utilize ORB features [88] with robust descriptors to improve short and mid-term data associations. They also construct a covisibility graph to manage keyframes, and performs loop closure and relocalization with the DBoW2 library [37]. Example of ORB-SLAM's results are shown in Fig. 7.7a.
- One typical direct method for visual odometry is the Direct Sparse Odometry (DSO) [24]. DSO directly processes raw sensor measurements to estimate the geometry. It minimizes the *photometric error* that computes the intensity difference between corresponding pixels, instead of the *reprojection error* in feature-based methods. The direct method skips the keypoint extraction and matching step, which saves a high constant cost per frame. However, it may fail in large viewpoint changes, and optimizing the photometric residual easily gets stuck in a local minimum. Example of DSO's results are shown in Fig. 7.7b.
- Semidirect VO (SVO) [28] belongs to the hybrid domain, which sparse image alignment (direct manner) to estimate frame-to-frame motion and bundle adjustment (indirect manner) to refine the geometry and camera poses. Examples of SVO's results are shown in Fig. 7.7c.

Learning-based approaches open up a new door for designing novel visual systems in recent years [9, 69, 130]. For example, semantics from images [10] can provide object types and properties, which offer depth prior and strong association cues.

The minimum setup of VSLAM is using a monocular camera, but the absolute scale is missing. An initialization phase based on physical-based prior is required to establish a map with sufficient points. In practice, since epipolar constraints can acquire the depth information [1], stereo or multi-camera solutions are more popular. Furthermore, RGB-D cameras that exploit the time-of-flight technique to obtain depth offer another option directly, as demonstrated in applications [73, 99].

But traditional cameras are easily affected in situations of high dynamics, low texture or structure distinctiveness, and challenging illumination conditions (see Fig. 7.8). Novel solutions are proposed:

Fig. 7.7 Results of classical VSLAM systems



(a) ORB-SLAM [38]

(b) DSO [39]

(c) SVO [40]

- One direction is to fuse cameras with the IMU, leading to the visual-inertial system that the IMU helps to resolve the high dynamics and textureless issues [15, 57, 78, 87]. Alternatively, cameras can be fused with the LiDAR, in which the absolute depth information eliminates the scale ambiguity problem [114, 125, 132].
- Another direction is to develop novel cameras that capture images with different techniques, such as the bio-inspired event cameras. Different from traditional frame cameras which capture intensity images at a fixed rate, event cameras asynchronously capture the per-pixel *intensity changes* and output a stream of *events*. Each event is encoded with information, including the triggered time, pixel localization, and the sign of the intensity change. As summarized in [35], event cameras have high temporal resolution (Âμs-level), high dynamic range (140 dB vs. 60 dB of frame cameras), and low power consumption. These characteristics enable vent cameras to have great potential for several computer vision and robotic tasks, e.g., high-speed motion estimation [12, 34, 46], feature tracking [54], and high dynamic

(a) Motion blur

(b) High illumination

Fig. 7.8 Open challenges in traditional computer vision [123]: **a** motion blur and **b** high illumination

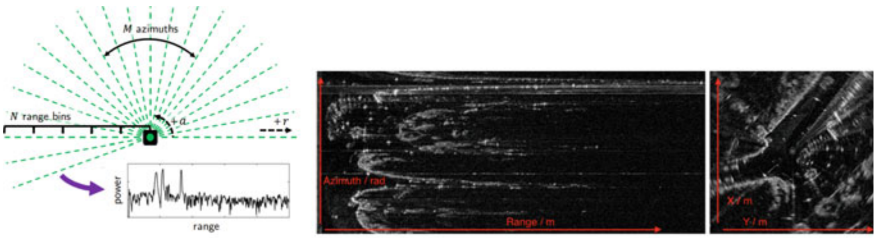
range perception [75, 84], which are commonly difficult to frame cameras. However, research on common vision problems with event cameras is preliminary. This is because event cameras work in a fundamentally different way from frame cameras. Existing vision-based solutions still lack a good way to handle events. Novel event-based algorithms must be investigated.

7.2.2.2 RaDAR

RaDAR sensors emit and receive radio waves to determine objects' distance, angle, and velocity. The important advantage of the RaDAR is its reliability against extreme conditions such as rain, snow, dust, fog, or direct sunlight. Since the technology is rapidly becoming affordable and efficient, RaDARs are currently accessible to modern self-driving cars.

We take the Frequency-Modulated Continuous-Wave (FMCW) RaDAR[2] as an example. The FMCW RaDAR is a special RaDAR that provides a 360°-view of the scenes with several desirable features: high reliability, high resolution, and long-range. The RaDAR data can be viewed as a 2D image that can be processed with vision-based techniques, as shown in Fig. 7.9. Research on FMCW RaDARs has been active in recent years [44]. Cen et al. [17, 18] present a RaDAR-only odometry pipeline with efficient keypoint extraction and graph-based matching. The graph matching approach is robust to false-positive key points. Hong et al. [42] propose the complete graph-based RaDAR SLAM system: *RaDAR-SLAM* that online manages the map points, detects loop candidates, and optimizes pose graphs to correct drift. Burnett et al. [13] provide solutions to compensate the motion distortion and Doppler effect in radar odometry, while this was often neglected by previous research. Directly aligning two consecutive RaDAR frames offer another option to solve the odometry

[2] https://navtechradar.com/explore/fmcw-radar.



(a) (b)

Fig. 7.9 **a** Bird-eye view of a FMCW scanning radar [17]. **b** Example sensor data from the Navtech CTS350-X radar [4]

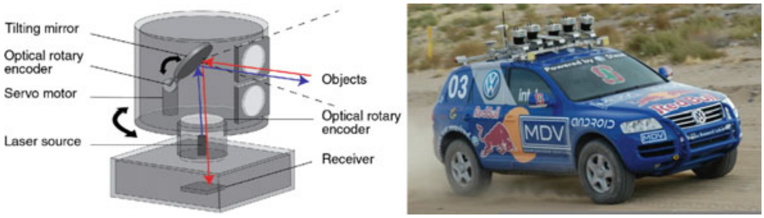
problem. An example is demonstrated in [76] which exploits the phase correlation. Regarding the object detection and place recognition tasks, FMCW RaDARs are also feasible, as presented in [11, 32, 101, 103, 116]. Available open datasets aush as Oxford RaDAR RobotCar dataset [4] and MulRan dataset [51] further spur research in this area.

RaDARs and LiDARs have similar working principles, but each uses different wavelengths of light. The longer wavelength used by RaDAR does not allow the detection of small objects. It is also challenging to determine the object's category (e.g., pedestrian or cars) from RaDARs. A complete navigation system should make the complementary strengths of RaDARs and LiDARs, as shown in [41].

7.2.2.3 LiDAR

LiDAR sensors emit and receive lasers (an amplified light with a short wavelength) to determine the distance of an object. The intensity of the laser partially reflects the surface materials of the targeted object. This property can be used to recognize special targets such as the road border. Figure 7.10a illustrates the working principle of LiDARs. Traditional mechanical scanning LiDARs (e.g., Velodyne HDL-64E) commonly align multiple lasers vertically and physically rotate them at a high rate (around 3600°–7200° per second) to obtain a 360° horizontal field of view. This can significantly enhance the perceptual view compared with single-beam LiDARs. Due to their active nature in measuring distance, LiDARs are robust to external weather and light conditions and very reliable in terms of measurement accuracy. As a drawback, these sensors contain large moving parts (like the rotating emitter), which leaves room for mechanical errors. Moreover, some inaccuracies can also exist due to the collision materials. For example, black objects can absorb much of the emitted light may be generating no-measurements, and windows or translucent materials may produce several reflections. Figure 7.10a illustrates the working principle of a classical mechanical LiDAR.

The application of autonomous driving accelerates the development of LiDAR technology. A pivotal occurrence should be the success of Stanley [102] that is



(a) Working principle (b) Stanley

Fig. 7.10 **a** The working principle of a traditional mechanical LiDAR. **b** The Stanley autonomous vehicle is equipped with multiple LiDARs [102]

(a) Velodyne (b) Ouster (c) Livox (d) Mid-360

Fig. 7.11 Four types of commonly used LiDARs

the first autonomous vehicle completed the DARPA Grand Challenge in 2005 (see Fig. 7.10b). Stanley replied on LiDARs in 3D mapping and obstacle detection. This event has motivated researchers and engineers to improve both hardware and algorithms of LiDARs. In the past ten years, the complexity and cost of producing a LiDAR quadratically increased along with the number of used lasers. However, now, it is no longer an issue. More and more companies, such as *Velodyne*, *Ouster*, *Luminar*, *Robosense*, *Livox*, have joined the LiDAR market in recent years (see Fig. 7.11). Some of them resort to the solid-state design to eliminate the moving mechanical parts. The decreasing weight, size, and price make LiDARs gradually available in robotic platforms. Most of self-driving cars adopt LiDARs as their standard setup. LiDARs are also popular to mobile robots [61] and quadrupedal robots [56].

7.3 SLAM

SLAM has been at the core of research in the field of robot navigation for more than 30 years since it was proposed. Currently, SLAM has been widely used in many applications such as service robots, autonomous driving, virtual/augmented reality. The goal of SLAM is to estimate a robot's states while simultaneously constructing a model (also called map) of the world from sensory data [14]. A robot *state* consists of quantities, such as the pose (position and orientation) that describe the robot's motion over time. A state also includes the robot's velocity, sensor biases,

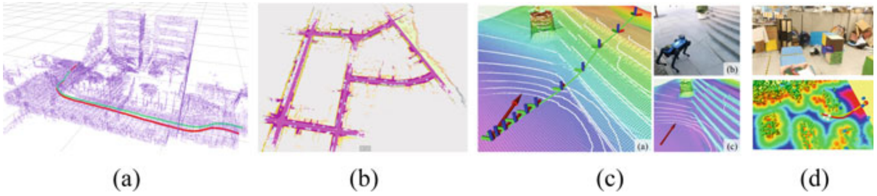


Fig. 7.12 Examples of SLAM outputs. Robot's trajectories are shown as red or green lines in **a** and **c**. Maps are represented in different formats: **a** an aggregation of 3D LiDAR points [48], **b** dense semantic mesh map, **c** elevation map [111], and **d** Euclidean Signed Distance Field (ESDF) which is used for path planning (yellow and red lines) [40]

kinematics, and calibration parameters, depending on the problems' complexity. A *map* is a representation of the environment in which the robot operates. It can be expressed in numerous formats, e.g., landmarks' positions, object bounding boxes, grid maps, which is application-dependent. Figure 7.12 shows some examples of SLAM's results, including estimated trajectories and maps.

7.3.1 Architecture of SLAM

A pictorial representation of a typical SLAM system is given in Fig. 7.13. The architecture of a SLAM system includes two main components: the *front end* and *back end*. The front end abstracts features from sensor data that are amenable for estimation. For instance, in some VSLAM systems, the front end extracts the pixel location of key points from images which are easy to model with the back end. The front end is also in charge of associating each measurement to a specific landmark in the environment. This process is also called *data association*. The front end might also provide an initial guess for the variables in optimization (e.g., initialize landmarks' position from multiple views). In contrast, the back end performs state estimation

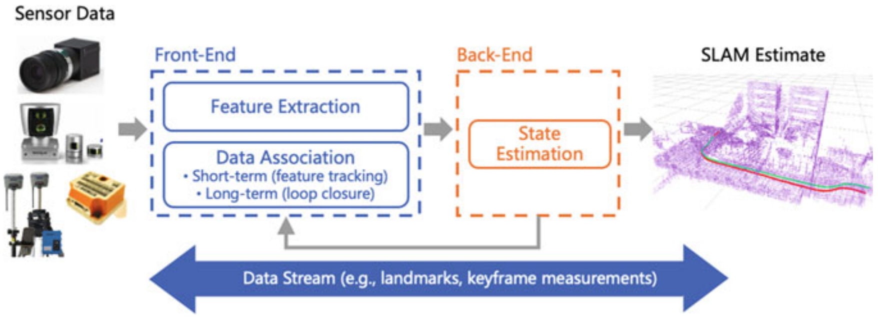

Sensor Data
Front-End
Feature Extraction
Data Association
• Short-term (feature tracking)
• Long-term (loop closure)
Back-End
State Estimation
SLAM Estimate
Data Stream (e.g., landmarks, keyframe measurements)

Fig. 7.13 Front end and back end in a typical SLAM system

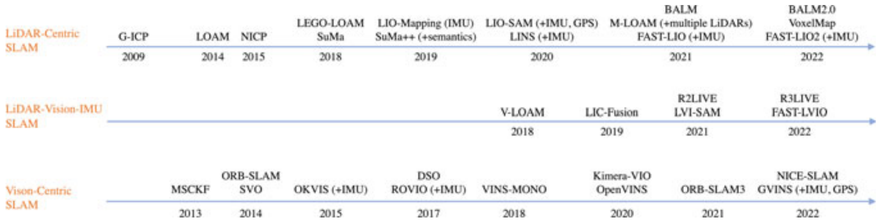


Fig. 7.14 Summary of related work of modern SLAM systems

on the abstracted data produced by the front end. There are several classical state estimation methods [3]: Maximum A Posteriori (MAP) estimation, Gaussian Filters (e.g., the Kalman Filter), and Non-parametric Filters (e.g., Particle Filter).

The data stream management is also important for SLAM. The data association module needs frequent operations of data, which becomes one of the bottlenecks of SLAM. For short-term data association, we need to retrieve and associate corresponding features between current frames and recently consecutive frames. For long-term data association, we need to associate new measurements to older landmarks. Features or landmarks can be stored and managed with the tree-based structure (e.g., the KDTree [6] and its incremental version [110]), hash map, covisibility graph, to support efficient data query, addition, and removal.

7.3.2 *Challenges of SLAM*

The major challenges faced by modern SLAM algorithms are summarized as follows:

- **Degeneracy**: Various scenarios might degrade sensor measurements. For instance, scenes such as tunnels and corridors cannot constrain the LiDAR's motion in several degree of freedom (DoF). Scenes such as low light and intense motion are detrimental to the quality of images. Thus, SLAM methods based on multi-sensor fusion are becoming more and more important.
- **Changing Scenes**: In dynamic environments, several elements are moving during the repeated traverse of a robot. The more time of the traverse, the more deformed the elements will be (e.g., pedestrians, cars, roads under construction). This deformation will lead to wrong data associations, making SLAM accuracy drop or even fail.
- **Scalability**: Applications including exploration for environmental monitoring or large-scale precision agriculture, SLAM may encounter issues of significant grow-

ing of computational time and memory footprint. Due to limited resources of a robot, the computational and memory complexity should be bouned for large-scale SLAM.

7.3.3 Modern SLAM Systems

There are extensive SLAM systems (Fig. 7.14). Here we focus on modern SLAM systems which are commonly used in mobile robots.

7.3.3.1 LiDAR-Based SLAM

Since the DRAPA Challenge in 2006, 3D LiDAR has been widely used in autonomous driving. The LiDAR has become the prevalent choice for deploying robot mapping and positioning algorithms due to its good stable performance under extreme lighting conditions.

LiDAR-Only Systems

Early 3D LiDAR SLAM algorithms often use the iterative closest point (ICP) algorithm [2] to estimate relative transformation between consecutive frames. ICP variants including the point-to-plane ICP and point-to-line ICP have been proposed to improve the original cost function and accelerate the computation.

Segal et al. put forward the [91] that combines point-to-point and point-to-plane metrics. It uses plane normal vectors to assign weights to each error point of the objective function of the optimization problem model to reduce the influence of outliers and initial values. Serafin et al. proposed the NICP [92] that eliminates false matches based on distance, curvature, and normal vectors, and optimizes the normal vectors when optimizing transformations. Magnusson et al. proposed a probabilistic matching algorithm based on the normal distribution assumption, namely the NDT (normal distribution transformation) algorithm [68]. It formulates the cost function from the probabilistic perspective instead of the metric distance, by assuming that the point cloud falls in each grid conforms to a normal distribution, so as to construct the probability distribution function of each grid.

In recent years, Yokozuka et al. presented the LiTAMIN [117] that adopts the F-norm and regularized covariance matrix to normalize the loss function. They further proposed the LiTAMIN2 [118] that computes the symmetric KL-Divergence. The formulated cost function includes not only the distance between points but also the difference between the shape of the distribution. Jiao et al. proposed the first multi-LiDAR SLAM system (called M-LOAM) [48]. The increasing LiDARs significantly enlarge the robot's field of view.

Maintaining the global consistency of a LiDAR SLAM system is an important problem. Liu et al. presented the BALM [64] algorithm to introduce idea of bundle

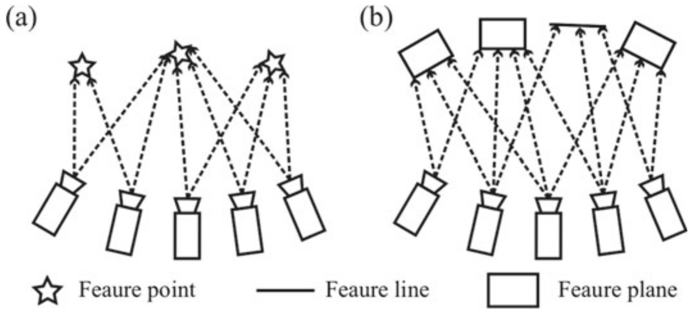


Fig. 7.15 Comparison of BA formulations: **a** visual BA constrains feature points from locating at the same point; **b** LiDAR BA [64] constrains feature points from lie on the same edge or plane

adjustment (BA) into LiDAR SLAM, as shown in Fig. 7.15. With the closed-form representation of planar features, BALM optimizes poses and feature variables simultaneously, so as to reducing the accumulative drift of the map. They further proposed the BALM 2.0 [66] to encode all raw point clouds associated with the corresponding feature through a compact set of parameters and derive the second-order closed-form derivative of BA. Also focusing on the multi-view registration problem, Huang et al. proposed a different objective function and designed a set of voxel-based multi-view registration pipelines to achieve better precision than BALM.

There are also LiDAR-only systems that contribute to novel representations of point clouds and maps. SuMa [5] was presented to project the 3D laser point cloud into a 2D depth map, which calculates the normal vector of points in parallel with a GPU. It also represents the point cloud with a set of surfels that are associated with normal vectors and uncertainties. Park et al. [77] also utilized the probabilistic surfel fusion to address the LiDAR mapping problem. By considering the measurement noise caused by the beam direction and distance, the uncertainty of the position and normal of each surfel is modeled. Yuan et al. proposed the VoxelMap [119], which expresses the map with a grid of adaptive size, models the uncertainty caused by pose error and plane point measurement noise, deduces its propagation process in the system, and finally achieves a higher map consistency.

LiDAR SLAM is also driven by new types of LiDARs that have different scanning patterns from mechanical LiDARs. Livox-LOAM [62] explicitly designed feature extraction and continuous point cloud correction for the Livox solid-state LiDAR. LILI-OM [59] integrated the Livox LiDAR with IMU measurements, which presents a sliding-window state estimator based on hierarchical keyframes.

Combining point clouds with semantics to enhance the accuracy and robustness of SLAM is becoming a popular solution. Chen et al. proposed the SuMa++ [20] where the semantics-aware ICP was presented. The semantic feature provides higher-level information such as the types of objects and is beneficial to inlier matching (e.g., removing dynamic objects).

IMU- and GNSS-Aid LiDAR-Based SLAM

In 2014, Zhang et al. proposed the LiDAR Odometry and Mapping (LOAM) [124] system, which defined the characteristics of the curvature of a point, and extracted plane points, and edge points through the curvature distinction. It uses these associated features to estimate the scan-to-scan and scan-to-map transform. LOAM can be optionally coupled with high-frequency IMU measurements to help predict the pose of the next frame. This type of fused method is also regarded as a loosely coupled approach. Due its great performance in both indoor and outdoor scenarios, the improved algorithms based on LOAM have gradually become mainstream in following years.

Shan et al. proposed the LEGO-LOAM [93] to explicitly utilize ground points in optimization. IMU measurements are fused in a similar manner to LOAM. Ye et al. proposed the first open-source tightly-coupled LiDAR-inertial odometry (LIO): LIO-Mapping [113]. LIO-Mapping jointly optimizes the LiDAR and IMU biases within a sliding window. On the basis of LEGO-LOAM, the improved LIO-SAM [95] introduces pre-integration constraints into the factor graph formulation. Besides the MAP estimation, Kalman Filters are also widely used in several LIO systems. Qin et al. [80] proposed a tightly-coupled LIO system based on the error state Kalman Filter, named LINS, which has achieved high operating efficiency and accuracy in scenes such as ports and wharves. Xu et al. proposed the FAST-LIO [109] and FAST-LIO2 [110]. Two contributions are presented. First, they designed an iKD-Tree data structure to support online data addition and deletion. Second, they advanced a manifold iterative error Kalman filter, which greatly reduce the data dimension. In most scenarios, FAST-LIO2's accuracy, robustness, and efficiency have reached the state-of-the-art level.

Several LiDAR-based SLAM systems are aimed at outdoor scenes and combined with GNSS to eliminate the accumulative drift. LIO-SAM introduces the GPS factor into the backend in a loosely coupled form to achieve large-scale low-drift mapping, as shown in Fig. 7.16. Another algorithm [7] designs a novel factor graph and introduce carrier phase and pseudo-range factors to eliminate the mapping drift.

7.3.3.2 Vision-Based SLAM

The camera is another popular senosr for mobile robots, and thus the vision-based SLAM is very active in the SLAM research field. But the pure visual odometry (VO) is hard to achieve stable positioning and mapping due to factors such as lighting and motion blur. Generally, it can be used in combination with a high-frequency IMU to significantly improve these problems. Compared with the pure VO, VIO allows the state estimation under intense motion and texture-less environments.

Existing VIO systems are mainly divided into two categories: Kalman filter-based and MAP-based approaches. Algorithms based on the Kalman filter include MSCKF [58], ROVIO [8], Open-VINS [38], while OKVIS [57], VINS-Mono [78], and Kimera-VIO [87] are MAP-based methods. ROVIO combines with IMU mea-

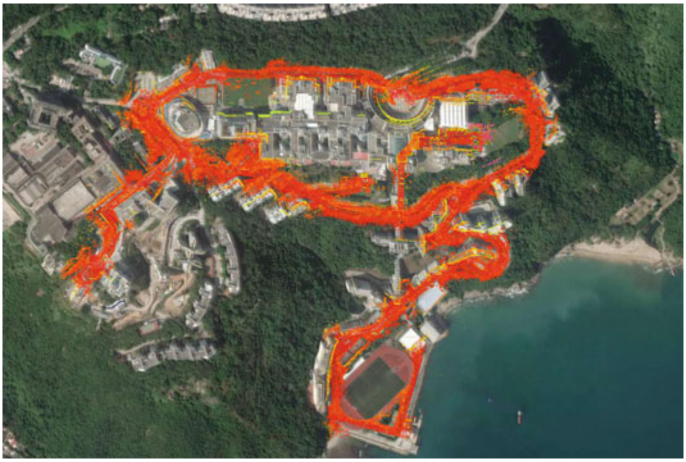


Fig. 7.16 The point cloud map is constructed by the LIO-SAM integrated with GNSS measurements, which is aligned on the google map

surements to track the optical flow of the pixels in the image block near the feature points, and minimize the photometric error. It can stably estimate the robot's position even when there are few feature points. On the other hand, VINS-Mono combines all IMU and image measurements within a fixed-size sliding window to optimize the camera pose. VINS-Mono also includes a bag-of-words loop closure detection module, which can perform 4-DOF loop optimization. ORB-SLAM [71], a representative of the feature-based method, is based on ORB features [88]. Its enhanced version: ORB-SLAM3 [15] has features including supporting more camera modalities: stereo and fisheye cameras, tightly coupled VIO, and multi-map.

As mentioned in Sect. 7.2.2.1, event cameras indicate a new research direction. Regarding the event-based VO/SLAM, recent works can be categorized according to three complexity axes: *(1)* problem dimensionality: from handling the localization subproblem [33] to solving the complete tracking-and-mapping problem with the need to estimate depth [83]; *(2)* type of motion: from constrained motions such as pure rotation or planar motion [36] to arbitrary 6-DoF motions [46, 127]; *(3)* type of scenes: from artificial patterns [70] to natural scenes [85].

7.3.3.3 LiDAR-Visual-IMU-Based SLAM

Another interesting series of SLAM systems should be based on the combination of LiDAR, visual, and inertial measurements. Such systems are called LVI systems. Current LVI systems mainly focuses on two issues: *(1)* Robustness: vision-based constraints can helps LIO to avoid degeneration, while LIO can significantly enlarge VO's scalability. *(2)* Data enhancement: LiDAR measurements offer sparse depth values of visual features, while images can be used to colorize point clouds.

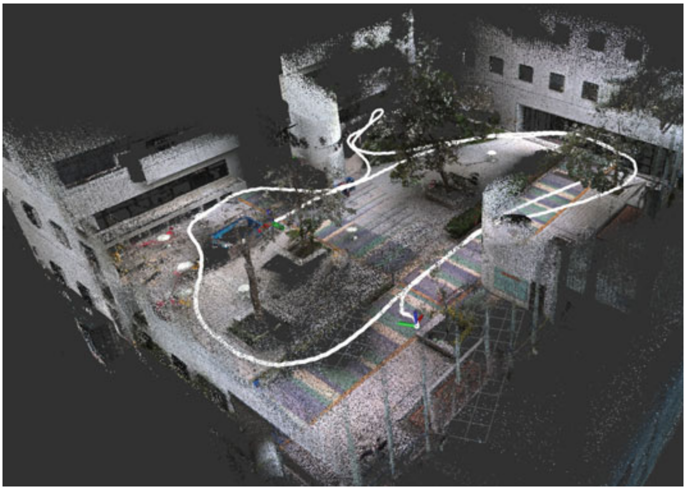


Fig. 7.17 R3LIVE's results: estimated trajectory (white) and colorized point cloud

V-LOAM [122] associates the visual features with the LiDAR measurements. The visual pose serves as the prior the LOAM algorithm. LVI-SAM [96] combines VINS-Mono [78] and LIO-SAM [95] as well as adds a scene degradation detection module on the basis of VLOAM, realizing seamless switching between the LIO and VIO system. LIC-Fusion [132] is an MSCKF-based solution that combines IMU, camera, and LiDAR measurements in a sliding window to optimize the robot pose. R2LIVE [60] is based on the ESIKF (error-state iterated Kalman Filter) method, which extracts sparse visual features and points cloud features. It minimizes the reprojection error within the sliding window. Its follow-up work: R3LIVE [63] followed the semi-dense framework and formulates the photometric error in VIO. R3LIVE also proposed to colorize the map gathered by LiDAR point clouds and updated points' color with the Bayesian filter. An example of R3LIVE's results is shown in Fig. 7.17. In contrast, FAST-LIVO [126] does not extract visual features, while directly using image blocks of map-associated points to minimize photometric errors.

7.4 Application

This section introduces a practical example of creating a multi-sensor SLAM dataset. Details including sensor specifications, calibration techniques, data collection, and evaluation of some SLAM systems are presented. We believe that these content can benefit to readers by providing sufficient implementation details.

7.4.1 Motivation

The high-quality open dataset, which is the collection of multi-sensor data and provide a suite of benchmark tools, significantly contributes to current progress of multi-sensor SLAM. On one hand, these datasets can waive inhibitive requirements on budget and manpower, such as system integration calibration, and field operations. On the other hand, they investigate advantages and limitations of current SLAM solutions, and elaboratedly design practical but challenging sequences [104]. Benefiting from these efforts, researchers can easily develop, validate, and rank their algorithms with others, thus acclerating the breakthroughs. However, existing datasets were mostly collected with a single data collection platform or simplified sensor configuration. Researchers may only utilize limited sensors to develop algorithms to bet fit the datasets, but cannot generalize it well to other scenarios. We consider that a desirable dataset should fullfill the following four requirements.

1. Various sensors are required in the dataset, encouraging researchers to explore novel approaches to use them.
2. Algorithm evaluation should be fairly conducted on various mobile robots. These robots perform different motion patterns that may challenge several SLAM algorithms' assumptions.
3. Sequences have to cover from room-scale (meter-level) to large-scale (kilometer-level) environments to evaluate algorithms' scalability. Large-scale sequences might exist several environmental changes, e.g., moving objects and structural changes.
4. Ground-truth trajectories and 3D maps are required to evaluate algorithms' localization and surface reconstruction accuracy, respectively.

We are motivated to propose the **FusionPortable Benchmark**,[3] a novel multi-sensor dataset with a set of sequences from diverse environments [47]. Compared with previous datasets [4, 89, 106, 115, 129], three new features are emphasized.

First, we advance a *portable* and *versatile* multi-sensor device that is elaborately manufactured. Two RGB frame cameras are mounted on the left and right side. Two event cameras are also similarly mounted. One 128-beam mechanical LiDAR is installed at the middle of the device. One high-frequency and high-precision IMU is mounted below the LiDAR, while one RTK-GPS is installed on the top of the LiDAR. All these sensors are mounted on the same rigid aluminum-alloy-based parts. The complete device has its own clock synchronization unit, processor, and battery, thus self-contained. Since its size, weight, and extensibility are satisfying, we advance that it would be a plug-and-play support to various mobile robots.

Second, we install the sensor rig on various platforms ranging from the handheld mode with a gimbal stabilizer, a quadruped robot, and an autonomous vehicle in performing distinguishable motion for the dataset construction. Various structured or semi-structured environments on the campus, including the lab, garden, canteen,

[3] https://ram-lab.com/file/site/fusionportable/dataset/fusionportable.

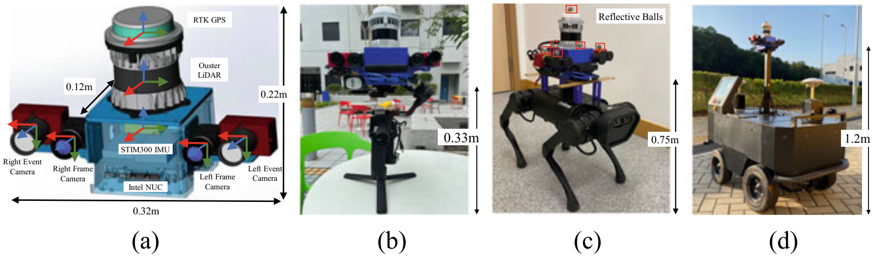


Fig. 7.18 The multi-sensor device and data collection platform

corridor, escalator, and outdoor road, are examined. The collected sequences might present several environmental changes caused by external light, moving objects, and scene texture. These issues are challenging to SLAM algorithms.

Third, besides ground-truth poses, we also provide ground-truth maps of most indoor sequences. We benchmarked several state-of-the-art (SOTA) SLAM systems, including two vision-based methods and four LiDAR-based approaches.

7.4.2 System Overview

Here, we introduce sensor specifications and how to calibrate the presented multi-sensor device. Figure 7.18 shows how the handheld device is mounted on three data collection platforms.

7.4.2.1 Sensor Configuration

Sensors' characteristics can be found in Table 7.1. Asynchronous events are assembled into individual packages and published at a fixed rate. We provide both the ROS bag format and raw data format for researchers. We use the Intel NUC to run sensor drivers, attach timestamps of sensor messages, and record messages into ROS bags on the Ubuntu system. The PC uses an i7 processor, 1TB solid-state drive (SSD), and 64GB DDR4 memory.

7.4.2.2 3D LiDARs

We configure the OS1–128[4] with a 45° FOV as the LiDAR to provide accurate measurements of surrounding environments (around 2.62 million points per second). This LiDAR enjoys three advantages that attract us to select. First, it has an internal

[4] https://ouster.com/zh-cn/products/scanning-lidar/os1-sensor.

Table 7.1 Sensors and characteristics

Sensor	Characteristics
3D LiDAR	OS1–128, 120 m range@**10** Hz; FOV: 45°vert., 360°horiz Image: 1028 × 128@**10** Hz IMU: ICM20948@**100**Hz, 9-axis MEMS, intrinsic calibrated
Frame Camera	Stereo color cameras: 2 FILR BFS-U3-31S4C Resolution: 1024 × 768, global shutter@**20** Hz; FOV: 66.5°vert., 82.9°horiz
Event Camera	Stereo color event cameras: 2 DAVIS346 Resolution: 346 × 240; FOV: 67°vert., 83°horiz IMU: MPU6150@**1000**Hz, 6-axis MEMS, intrinsic calibrated
IMU	STIM300@**200** Hz, Bias Instability 0.3°$/h$, Allan Var. @25°C
GPS	ZED-F9P RTK-GPS@**10** Hz, 4 concurrent GNSS, L1/L2/L5 RTK

synchronized IMU that outputs 100 Hz linear accelerations and angular velocities. Second, it provides additional properties that are encoded into depth images, signal images, and ambient images of surroundings. These images are perfectly correlated with point clouds. Especially, the ambient images resemble visible light images of the same scenes, which open opportunities of applying image-based methods to process range data efficiently, as demonstrated in [94]. Third, its size, weight, and affordable price are encouraging to different types of mobile robot [74, 107].

7.4.2.3 Stereo Frame Cameras

Two FILR BFS-U3-31S4C global-shutter color cameras[5] are mounted at two sides on the system, facing directly forward. They are synchronized by an external trigger and capture high-resolution images at 20 fps. Their exposure time is set as fixed values to minimize the relative latency. Our experiments show that the average difference in timestamps of these images is below $1ms$.

7.4.2.4 Stereo Event Cameras

We configure two event cameras which have a 346 × 260 resolution and an internal high-rate IMU output. With known extrinsics, the stereo event camera offers spatio-temporal constraints for semi-dense mapping. Event cameras are synchronized using the trigger signal generated from the left camera (*master*) to deliver sync pulses to the right (*slave*) through an external wire.[6] But there is no way to synchronize the image acquisition ($\approx 15ms$ offset). To suppress the LiDAR's laser light, both cameras are

[5] https://www.flir.eu/products/blackfly-s-usb3.

[6] https://inivation.gitlab.io/dv/dv-docs/docs/external-camera-sync.

equipped with additional infrared filters. For indoor sequences, we manually set and fix the APS exposures, which helps to minimize the latency between cameras.

7.4.2.5 Inertial Measurement Unit

A tactical-grade STIM300 IMU that is rigidly mounted below the LiDAR is employed as the main inertial sensor of the system. It features a high update rate (200Hz) but noisy and drifting measurements.

7.4.2.6 Global Positioning System

We additionally install a ZED-F9P RTK-GPS device on the top of the LiDAR. In outdoor scenes, the GPS is activated and provides accurate latitude, longitude, and altitude readings. But it may sometimes become unstable due to buildings' occlusion.

7.4.3 Sensor Calibration

We should carefully calibrate *intrinsics* of individual sensors, *extrinsics*, and overall time latency between sensors in advance. We set the coordinate system of the STIM300 IMU as the body frame, and use the estimated spatio-temporal parameters to fuse with other sensors.

7.4.3.1 Clock Synchronization

We use an Field Programmable Gate Array (FPGA) to generate an external signal trigger to synchronize clocks of all sensors. This can guarantee data collection across multiple sensors with minimum latency. In GPS-enabled environments, the FPGA receives a pulse-per-second (PPS) signal from the GPS and outputs 200, 20, 10Hz signal to the IMU, cameras, and LiDAR respectively. To enable that the time synchronization still works in GPS-denied scenes, the FPGA switches to use its internal clock.

7.4.3.2 Stereo Camera Calibration

Intrinsics and extrinsics of our stereo frame and event cameras are estimated using the off-the-shelf Matlab calibration toolbox.[7] We move the sensor suite before a checkerboard to collect a sequence of images. To avoid motion blur and ensure

[7] https://www.mathworks.com/help/vision/camera-calibration.html.

sufficient constraints, the motion is slow and contains significant transformation. We evenly sample images as the calibration data, and manually remove outliers if their reprojection errors are high. According to lens used on cameras, we use the pinhole camera and radial-tangential distortion model. The Kalibr [30] also offers toolkits for camera calibration. But Kalibr's results have higher reprojection errors than those of Matlab, and thus are not adopted.

7.4.3.3 Camera-IMU Extrinsic Calibration

The intrinsics of IMUs can be calibrated using the open source Allen derivation tool that estimates the noisy density and random walk for gyroscope and accelerometer measurements. After that, the spatial and temporal parameters of a camera w.r.t. an IMU can be obtained by the Kalibr that estimates in a full batch optimization using splines to model poses of the system. Our system consists of 4 IMUs: STIM300, ICM20948 in the OS1 LiDAR, and two MPU6050 in the DAVIS346 event cameras. Thus, we calibrate intrinsics of all these IMUs, and estimate extrinsics of these sensor pairs: ⟨STIM300, frame cameras⟩, ⟨STIM300, event cameras⟩, ⟨left MPU6050, left DAVIS346⟩, and ⟨right MPU6050, right DAVIS346⟩. The extrinsics from the ICM20948 to the LiDAR are provided by the manufacturer.

7.4.3.4 Camera-LiDAR Extrinsic Calibration

Based on initial extrinsics provided by the CAD model, we further refine the relative transformation from the left frame/event camera to the LiDAR. The checkerboard is utilized as the calibration target that provides distinctive corners and boundaries for data association. We proposed the LCE-Calib [45] to address the calibration problem. We extract outer corners of the board from point clouds and images. The extrinsics are optimized by minimizing distance of all corresponding corners.

7.4.3.5 Remark

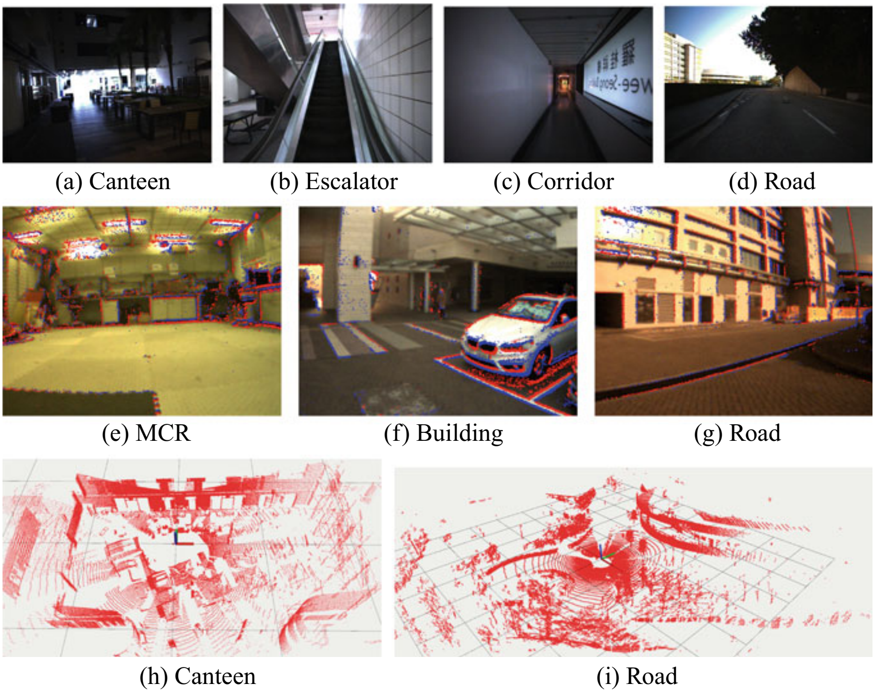
Life-long sensor calibration is always a challenging problem [67]. We provide the best estimates of parameters using the above methods, but we cannot guarantee that they are accurate for a specific traversal. We try our best to solidify the mechanical structure and external perturbation. We encourage readers to make our estimates as initial values and investigate novel calibration approaches for long-term extrinsics estimation.

7.4.4 Dataset Description

7.4.4.1 Sequences

The collected sequences should cover various environments, lighting conditions, motion patterns, dynamic objects, etc. Figure 7.19 illustrates sample sensor data. We categorize major characteristics of our collected sequences as follows:

1. **Location**: Environmental locations are divided into indoors and outdoors. GPS signal is available but sometimes unstable in outdoor environments.
2. **Structure**: Structured environments can mainly be explained using geometric primitives (e.g., offices or buildings), while semi-structured environments have both geometric and complex elements like trees and sundries. Scenarios like narrow corridors are structured but may cause state estimators.
3. **Lighting Condition**: Frame cameras are sensitive to external lighting conditions. Both weak and strong light may raise challenges to visual processing algorithms.
4. **Appearance**: Texture-rich scenes facilitate visual algorithms to extract stable features (e.g., points and lines), while textureless may negatively affect the performance. Also, many events are triggered in texture-rich scenes.



(a) Canteen (b) Escalator (c) Corridor (d) Road

(e) MCR (f) Building (g) Road

(h) Canteen (i) Road

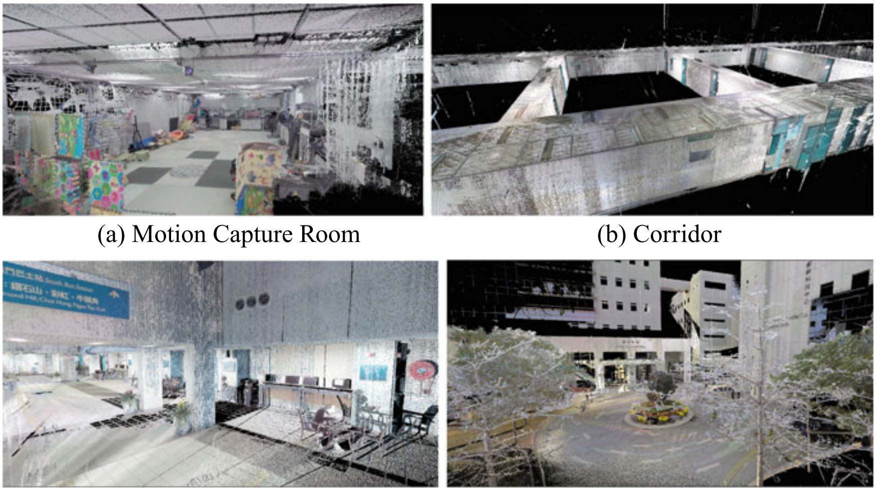
Fig. 7.19 Sample sensor measurements. **a**–**d** Images captured by the frame camera. **e**–**f** Images augmented by positive events (red) and negative events (blue). **h**–**i** 3D point clouds of the LiDAR

5. **Motion Pattern**: Slow, normal, and fast motion may be performed. Regarding mounted platforms, the handheld device performs arbitrary 6-DoF and jerky motions, the device installed on a gimbal stabilizer conducts 6-DoF but stable motions, the quadruped robot mostly performs planar but jerky motions. In contrast, the vehicle performs planar movements at a constant speed.

7.4.4.2 Groundtruth Generation

Most sequences provide ground-truth poses for algorithm evaluation. In several indoor scenes, we also provide ground-truth maps of surrounding environments. The ground truth generation is detailed as follows:

- **Ground-truth maps**: In small- or middle-scale environments, we use the Leica BLK360 laser scanner[8] to record the structure's high-resolution colorized 3D dense map with millimeter accuracy from multiple locations. This map can evaluate the surface reconstruction accuracy. Figure 7.20 visualizes three examples.
- **Ground-truth poses**: In the motion capture room, we use the OptiTrack[9] to measure the pose of the center of reflective balls at 120Hz with millimeter accuracy. The OptiTrack is directly connected with the same PC to record poses to minimize



(a) Motion Capture Room

(b) Corridor

(c) Inside the Building

(d) Outside the Building

Fig. 7.20 Ground-truth point cloud in color of the motion capture room, corridor, and building scenario. Point cloud data was recorded by the Leica BLK360 laser scanner. They are used to generate trajectory groundtruth and evaluate algorithms' reconstruction accuracy

[8] https://leica-geosystems.com/products/laser-scanners/scanners/blk360.

[9] https://optitrack.com.

the time latency. The extrinsics from the balls' center to the body frame of the sensor rig are solved by the hand-eye calibration approach [31]. In middle-scale environments that are covered by the ground-truth maps, we register the current frame with the map to estimate LiDAR's poses as the ground-truth trajectory. The initial pose for the registration is obtained with a LIO algorithm. In outdoor environments, we fuse the RTK GPS signal with LiDAR-inertial measurements to obtain accuracy trajectories based on the LIO-SAM [95].

7.4.5 Evaluation

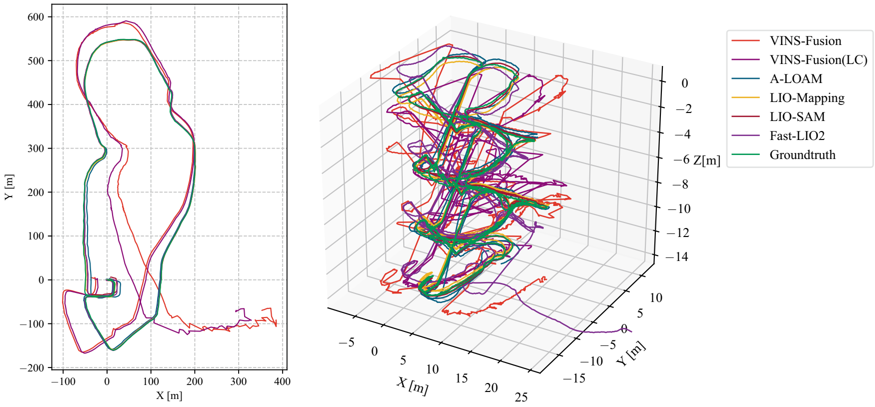
We use this dataset to benchmark some state-of-the-art (SOTA) SLAM systems. Here, we evaluate several open-source systems with different sensor combinations and methodologies: VINS-Fusion (IMU+stereo frame cameras) [79], ESVO (stereo event cameras) [127], A-LOAM (LiDAR-only) [124], LIO-Mapping (IMU+LiDAR) [113], LIO-SAM (IMU+LiDAR) [95], and FAST-LIO2 (IMU+LiDAR) [110]. We calculate the mean absolute trajectory error (ATE) of estimated trajectories w.r.t. the ground truth [123]. For LiDAR-based systems, we also report the mapping accuracy on two sequences by calculating the mean point-to-point error of algorithms' maps w.r.t. the ground-truth maps.[10].

The quantitative localization results are reported in Table 7.2. "LC" indicates that the loop closure module is used. "×" means that algorithms fail to finish the sequence. ESVO's results are not shown here since it cannot finish all sequences. It requires events to be continuously triggered to generate reliable time surface maps for camera tracking. But all these sequences contain textureless scenarios or static motion. In general, LiDAR-based methods outperform vision-based methods, especially on middle- or large-scale sequences. VINS-Fusion and FAST-LIO2 fail in some cases since they cannot initialize well at the beginning of the sequence. Without the aid of the IMU, A-LOAM cannot handle jerky and rapid motion and thus performs poorly on two MCR sequences and all sequences on the quadruped robot. Although FAST-LIO2 has a superior real-time performance based on the Kalman filter, it sometimes has unstable results on several sequences. **escalator_day** and **MCR_fast_01**. Surprisingly, LIO-SAM performs well on all quadruped robot-based sequences, even at fast and jerky motion with large rotation. But two sequences are still challenging to all vision- and LiDAR-methods: corridor day and stair day, where environments are commonly texture-less and structureless. Moreover, we also evaluate the mapping quality of A-LOAM and LIO-SAM on the **corridor_day** and **garden_day** sequences. Trajectory results on two sequences are visualized in Fig. 7.21. The distance map is in Fig. 7.22. Especially for the corridor mapping, A-LOAM's map has a large drift on the z-axis.

[10] https://github.com/mp3guy/SurfReg.

Table 7.2 Localization accuracy

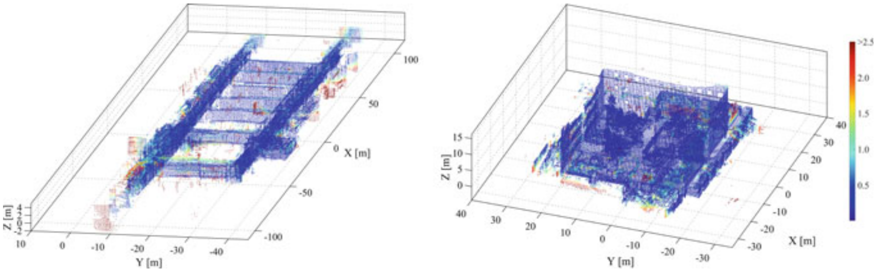
Platform	Sequence	VINS-Fusion (LC)	A-LOAM	LIO-Mapping	LIO-SAM	FAST-LIO2
Handheld	canteen_night	0.409	0.067	0.097	**0.063**	0.071
	canteen_day	0.691	0.057	0.088	**0.053**	0.057
	garden_night	0.328	0.567	0.242	0.254	**0.205**
	garden_day	0.518	0.528	0.097	0.069	**0.068**
	corridor_day	1.807	**0.416**	1.755	0.594	1.563
	escalator_day	2.127	0.981	0.346	**0.207**	4.193
	building_day	12.861	1.580	0.916	0.222	**0.146**
	MCR_slow	×	0.087	**0.042**	0.063	0.114
	MCR_normal	0.168	0.328	**0.052**	0.082	0.121
	MCR_fast	×	0.416	**0.099**	0.117	×
Quad. Robot	MCR_slow_00	0.096	0.120	0.032	**0.023**	0.047
	MCR_slow_01	0.081	0.054	**0.030**	**0.030**	0.051
	MCR_normal_00	0.094	0.492	0.093	**0.042**	0.127
	MCR_normal_01	0.086	0.635	0.390	**0.040**	0.068
	MCR_fast_00	0.264	4.601	2.405	**0.052**	0.408
	MCR_fast_01	0.130	8.264	2.210	**0.066**	1.495
Apollo	campus_road_day	77.528	5.707	4.122	7.364	**4.080**



(a) Campus_road_day

(b) Escalator_day

Fig. 7.21 Trajectories of the algorithms on two sequences: campus_road_day and escalator_day w.r.t. the ground truth



(a) Corridor_day

(b) Garden_day

Fig. 7.22 Evaluation of **a** A-LOAM's and **b** LIO-SAM's mapping accuracy

7.5 Conclusion

In conclusion, this chapter provides a comprehensive overview of the sensors commonly used in mobile robots, as well as the related works of modern SLAM systems. The chapter begins with an introduction to the preliminaries of sensors and then delves into key methodologies, major challenges, and LiDAR-based and vision-based approaches in SLAM. Furthermore, the chapter provides detailed steps for creating a multi-sensor open dataset, including practical considerations and techniques. Overall, this chapter serves as a valuable resource for those seeking to understand the fundamentals of sensors and SLAM, as well as for those looking to build their own multi-sensor open dataset.

References

1. Andrew AM (2001) Multiple view geometry in computer vision. Kybernetes
2. Arun KS, Huang TS, Blostein SD (1987) Least-squares fitting of two 3-d point sets. IEEE Trans Pattern Anal Mach Intell 5:698–700
3. Barfoot TD (2017) State estimation for robotics. Cambridge University Press
4. Barnes D, Gadd M, Murcutt P, Newman P, Posner I (2020) The oxford radar robotcar dataset: a radar extension to the oxford robot car dataset. In: 2020 IEEE international conference on robotics and automation (ICRA). IEEE, pp 6433–6438
5. Behley J, Stachniss C (2018) Efficient surfel-based slam using 3d laser range data in urban environments. Robot: Sci Syst
6. Bentley JL (1975) Multidimensional binary search trees used for associative searching. Commun ACM 18(9):509–517
7. Beuchert J, Camurri M, Fallon M (2022) Factor graph fusion of raw GNSS sensing with IMU and lidar for precise robot localization without a base station. arXiv:2209.14649
8. Bloesch M, Burri M, Omari S, Hutter M, Siegwart R (2017) Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback. Int J Robot Res 36(10):1053–1072
9. Bloesch M, Czarnowski J, Clark R, Leutenegger S, Davison AJ (2018) Codeslam-learning a compact, optimisable representation for dense visual slam. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2560–2568

10. Bowman SL, Atanasov N, Daniilidis K, Pappas GJ (2017) Probabilistic data association for semantic slam. In: IEEE international conference on robotics and automation (ICRA). IEEE, pp 1722–1729
11. Broome M, Gadd M, De Martini D, Newman P (2020) On the road: Route proposal from radar self-supervised by fuzzy lidar traversability. AI 1(4):558–585
12. Bryner S, Gallego G, Rebecq H, Scaramuzza D (2019) Event-based, direct camera tracking from a photometric 3d map using nonlinear optimization. In: 2019 international conference on robotics and automation (ICRA). IEEE, pp 325–331
13. Burnett K, Schoellig AP, Barfoot TD (2021) Do we need to compensate for motion distortion and doppler effects in spinning radar navigation? IEEE Robot Autom Lett 6(2):771–778
14. Cadena C, Carlone L, Carrillo H, Latif Y, Scaramuzza D, Neira J, Reid I, Leonard JJ (2016) Past, present, and future of simultaneous localization and mapping: toward the robust-perception age. IEEE Trans Robot 32(6):1309–1332
15. Campos C, Elvira R, Rodríguez JJG, Montiel JM, Tardós JD (2021) Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. IEEE Trans Robot
16. Cao S, Lu X, Shen S (2022) Gvins: tightly coupled GNSS-visual-inertial fusion for smooth and consistent state estimation. IEEE Trans Robot 38(4):2004–2021
17. Cen SH, Newman P (2018) Precise ego-motion estimation with millimeter-wave radar under diverse and challenging conditions. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, pp 6045–6052
18. Cen SH, Newman P (2019) Radar-only ego-motion estimation in difficult settings via graph matching. In: 2019 international conference on robotics and automation (ICRA). IEEE, pp 298–304
19. Cheng J, Chen Y, Zhang Q, Gan L, Liu C, Liu M (2022) Real-time trajectory planning for autonomous driving with gaussian process and incremental refinement. In: 2022 international conference on robotics and automation (ICRA). IEEE, pp 8999–9005
20. Chen X, Milioto A, Palazzolo E, Giguere P, Behley J, Stachniss C (2019) Suma++: Efficient lidar-based semantic slam. In: 2019 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 4530–4537
21. Chen Y, Xin R, Cheng J, Zhang Q, Mei X, Liu M, Wang L (2022) Efficient speed planning for autonomous driving in dynamic environment with interaction point model. IEEE Robot Autom Lett 7(4):11 839–11 846
22. Delmerico J, Cieslewski T, Rebecq H, Faessler M, Scaramuzza D (2019) Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset. In: 2019 international conference on robotics and automation (ICRA). IEEE, pp 6713–6719
23. Eckenhoff K, Yang Y, Geneva P, Huang G (2019) Tightly-coupled visual-inertial localization and 3-d rigid-body target tracking. IEEE Robot Autom Lett 4(2):1541–1548
24. Engel J, Koltun V, Cremers D (2017) Direct sparse odometry. IEEE Trans Pattern Anal Mach Intell 40(3):611–625
25. EUSPA (2021) What is GNSS. https://www.euspa.europa.eu/european-space/eu-space-programme/what-gnss
26. Forster C, Carlone L, Dellaert F, Scaramuzza D (2016) On-manifold preintegration for real-time visual-inertial odometry. IEEE Trans Robot 33(1):1–21
27. Forster C, Pizzoli M, Scaramuzza D (2014) Svo: fast semi-direct monocular visual odometry. In: IEEE international conference on robotics and automation (ICRA). IEEE, pp 15–22
28. Forster C, Zhang Z, Gassner M, Werlberger M, Scaramuzza D (2016) Svo: semidirect visual odometry for monocular and multicamera systems. IEEE Trans Robot 33(2):249–265
29. Forsyth D, Ponce J (2011) Computer vision: a modern approach. Prentice Hall
30. Furgale P, Rehder J, Siegwart R (2013) Unified temporal and spatial calibration for multi-sensor systems. In: 2013 IEEE/RSJ international conference on intelligent robots and systems. IEEE, pp 1280–1286
31. Furrer F, Fehr M, Novkovic T, Sommer H, Gilitschenski I, Siegwart R (2018) Evaluation of combined time-offset estimation and hand-eye calibration on robotic datasets. In: Field and service robotics: results of the 11th international conference. Springer, pp 145–159

32. Gadd M, De Martini D, Newman P (2021) Unsupervised place recognition with deep embedding learning over radar videos (2021). arXiv:2106.06703
33. Gallego G, Lund JE, Mueggler E, Rebecq H, Delbruck T, Scaramuzza D (2017) Event-based, 6-dof camera tracking from photometric depth maps. IEEE Trans Pattern Anal Mach Intell 40(10):2402–2412
34. Gallego G, Lund JE, Mueggler E, Rebecq H, Delbruck T, Scaramuzza D (2017) Event-based, 6-dof camera tracking from photometric depth maps. IEEE Trans Pattern Anal Mach Intell 40(10):2402–2412
35. Gallego G, Delbrück T, Orchard G, Bartolozzi C, Taba B, Censi A, Leutenegger S, Davison AJ, Conradt J, Daniilidis K, Scaramuzza D (2020) Event-based vision: a survey. IEEE Trans Pattern Anal Mach Intell 1–1
36. Gallego G, Rebecq H, Scaramuzza D (2018) A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3867–3876
37. Gálvez-López D, Tardos JD (2012) Bags of binary words for fast place recognition in image sequences. IEEE Trans Robot 28(5):1188–1197
38. Geneva P, Eckenhoff K, Lee W, Yang Y, Huang G (2020) Openvins: a research platform for visual-inertial estimation. In: 2020 IEEE international conference on robotics and automation (ICRA). IEEE, pp 4666–4672
39. Geyer J, Kassahun Y, Mahmudi M, Ricou X, Durgesh R, Chung AS, Hauswald L, Pham VH, Mühlegg M, Dorn S et al (2020) A2d2: audi autonomous driving dataset. arXiv:2004.06320
40. Han L, Gao F, Zhou B, Shen S (2019) Fiesta: fast incremental Euclidean distance fields for online motion planning of aerial robots. In: IEEE/RSJ international conference on intelligent robots and systems (IROS), pp 4423–4430
41. Heng L (2020) Automatic targetless extrinsic calibration of multiple 3d lidars and radars. In: 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 10 669–10 675
42. Hong Z, Petillot Y, Wang S (2020) Radarslam: radar based large-scale slam in all weathers. In: 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 5164–5170
43. Huang C, Gao F, Pan J, Yang Z, Qiu W, Chen P, Yang X, Shen S, Cheng K-T (2018) Act: An autonomous drone cinematography system for action scenes. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, pp 7039–7046
44. ICRA (2021) Radar in robotics. https://sites.google.com/view/radar-robotics/home
45. Jiao J, Chen F, Wei H, Wu J, Liu M (2023) Lce-calib: automatic lidar-frame/event camera extrinsic calibration with a globally optimal solution. arXiv:2303.09825
46. Jiao J, Huang H, Li L, He Z, Zhu Y, Liu M (2021) Comparing representations in tracking for event camera-based slam. In: Proceedings of the IEEE/cvf conference on computer vision and pattern recognition, pp 1369–1376
47. Jiao J, Wei H, Hu T, Hu X, Zhu Y, He Z, Wu J, Yu J, Xie X, Huang H et al (2022) Fusionportable: a multi-sensor campus-scene dataset for evaluation of localization and mapping accuracy on diverse platforms. In: 2022 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 3851–3856
48. Jiao J, Ye H, Zhu Y, Liu M (2021) Robust odometry and mapping for multi-lidar systems with online extrinsic calibration. IEEE Trans Robot
49. Khattak S, Nguyen H, Mascarich F, Dang T, Alexis K (2020) Complementary multi–modal sensor fusion for resilient robot pose estimation in subterranean environments. In: 2020 International conference on unmanned aircraft systems (ICUAS). IEEE, pp 1024–1029
50. Kilic C, Ohi N, Gu Y, Gross JN (2021) Slip-based autonomous zupt through gaussian process to improve planetary rover localization. IEEE Robot Autom Lett 6(3):4782–4789
51. Kim G, Park YS, Cho Y, Jeong J, Kim A (2020) Mulran: multimodal range dataset for urban place recognition. In: 2020 IEEE international conference on robotics and automation (ICRA). IEEE, pp 6246–6253

52. Kubelka V, Vaidis M, Pomerleau F (2022) Gravity-constrained point cloud registration. arXiv:2203.13799
53. Kumar V, Michael N (2012) Opportunities and challenges with autonomous micro aerial vehicles. The International Journal of Robotics Research 31(11):1279–1291
54. Le Gentil C, Tschopp F, Alzugaray I, Vidal-Calleja T, Siegwart R, Nieto J (2020) Idol: a framework for imu-dvs odometry using lines. In: 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 5863–5870
55. Lee W, Eckenhoff K, Yang Y, Geneva P, Huang G (2020) Visual-inertial-wheel odometry with online calibration. In: 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 4559–4566
56. Lee J, Hwangbo J, Wellhausen L, Koltun V, Hutter M (2020) Learning quadrupedal locomotion over challenging terrain. Sci Robot 5(47)
57. Leutenegger S, Lynen S, Bosse M, Siegwart R, Furgale P (2015) Keyframe-based visual-inertial odometry using nonlinear optimization. Int J Robot Res 34(3):314–334
58. Li M, Mourikis AI (2013) High-precision, consistent ekf-based visual-inertial odometry. Int J Robot Res 32(6):690–711
59. Li K, Li M, Hanebeck UD (2021) Towards high-performance solid-state-lidar-inertial odometry and mapping. IEEE Robot Autom Lett 6(3):5167–5174
60. Lin J, Zheng C, Xu W, Zhang F (2021) R2live: a robust, real-time, lidar-inertial-visual tightly-coupled state estimator and mapping. IEEE Robot Autom Lett 6(4):7469–7476
61. Lin J, Liu X, Zhang F (2020) A decentralized framework for simultaneous calibration, localization and mapping with multiple lidars. arXiv:2007.01483
62. Lin J, Zhang F (2020) Loam livox: a fast, robust, high-precision lidar odometry and mapping package for lidars of small FOV. In: 2020 ieee international conference on robotics and automation (ICRA). IEEE, pp 3126–3131
63. Lin J, Zhang F (2022) R3live: a robust, real-time, RGB-colored, lidar-inertial-visual tightly-coupled state estimation and mapping package. In: 2022 international conference on robotics and automation (ICRA). IEEE, pp 10 672–10 678
64. Liu Z, Zhang F (2021) Balm: bundle adjustment for lidar mapping. IEEE Robot Autom Lett 6(2):3184–3191
65. Liu T, hai Liao Q, Gan L, Ma F, Cheng J, Xie X, Wang Z, Chen Y, Zhu Y, Zhang S et al (2021) The role of the hercules autonomous vehicle during the covid-19 pandemic: an autonomous logistic vehicle for contactless goods transportation
66. Liu Z, Liu X, Zhang F (2022) Efficient and consistent bundle adjustment on lidar point clouds. arXiv:2209.08854
67. Maddern W, Pascoe G, Linegar C, Newman P (2017) 1 year, 1000 km: the oxford robotcar dataset. Int J Robot Res 36(1):3–15
68. Magnusson M, Andreasson H, Nuchter A, Lilienthal AJ (2009) Appearance-based loop detection from 3d laser data using the normal distributions transform. In: 2009 IEEE international conference on robotics and automation. IEEE, pp 23–28
69. McCormac J, Handa A, Davison A, Leutenegger S (2017) Semanticfusion: dense 3d semantic mapping with convolutional neural networks. In: 2017 IEEE international conference on robotics and automation (ICRA). IEEE, pp 4628–4635
70. Mueggler E, Gallego G, Scaramuzza D (2015) Continuous-time trajectory estimation for event-based vision sensors, Technical Report
71. Mur-Artal R, Montiel JMM, Tardos JD (2015) Orb-slam: a versatile and accurate monocular slam system. IEEE Trans Robot 31(5):1147–1163
72. Mur-Artal R, Tardós JD (2017) Orb-slam2: an open-source slam system for monocular, stereo, and RGB-d cameras. IEEE Trans Robot 33(5):1255–1262
73. Newcombe RA, Izadi S, Hilliges O, Molyneaux D, Kim D, Davison AJ, Kohi P, Shotton J, Hodges S, Fitzgibbon A (2011) Kinectfusion: real-time dense surface mapping and tracking. In: 10th IEEE international symposium on mixed and augmented reality. IEEE, pp 127–136
74. Nguyen TM, Yuan S, Cao M, Lyu Y, Nguyen TH, Xie L (2021) Ntu viral: a visual-inertial-ranging-lidar dataset, from an aerial vehicle viewpoint. Int J Robot Res 02783649211052312

75. Pan L, Scheerlinck C, Yu X, Hartley R, Liu M, Dai Y (2019) Bringing a blurry frame alive at high frame-rate with an event camera. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 6820–6829
76. Park YS, Shin YS, Kim A (2020) Pharao: direct radar odometry using phase correlation. In: 2020 IEEE international conference on robotics and automation (ICRA). IEEE, pp 2617–2623
77. Park C, Moghadam P, Williams JL, Kim S, Sridharan S, Fookes C (2021) Elasticity meets continuous-time: Map-centric dense 3d lidar slam. IEEE Trans Robot 38(2):978–997
78. Qin T, Li P, Shen S (2018) Vins-mono: a robust and versatile monocular visual-inertial state estimator. IEEE Trans Robot 34(4):1004–1020
79. Qin T, Pan J, Cao S, Shen S (2019) A general optimization-based framework for local odometry estimation with multiple sensors. arXiv:1901.03638
80. Qin C, Ye H, Pranata CE, Han J, Liu M (2020) Lins: A lidar-inerital state estimator for robust and fast navigation
81. Qiu K, Qin T, Gao W, Shen S (2019) Tracking 3-d motion of dynamic objects using monocular visual-inertial sensing. IEEE Trans Robot 35(4):799–816
82. Qiu K, Qin T, Pan J, Liu S, Shen S (2020) Real-time temporal and rotational calibration of heterogeneous sensors using motion correlation analysis. IEEE Trans Robot
83. Rebecq H, Horstschäfer T, Gallego G, Scaramuzza D (2016) Evo: a geometric approach to event-based 6-dof parallel tracking and mapping in real time. IEEE Robot Autom Lett 2(2):593–600
84. Rebecq H, Ranftl R, Koltun V, Scaramuzza D (2019) High speed and high dynamic range video with an event camera. IEEE Trans Pattern Anal Mach Intell 43(6):1964–1980
85. Rebecq H, Ranftl R, Koltun V, Scaramuzza D (2019) Events-to-video: bringing modern computer vision to event cameras. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 3857–3866
86. Reinke A, Palieri M, Morrell B, Chang Y, Ebadi K, Carlone L, Agha-Mohammadi AA (2022) Locus 2.0: robust and computationally efficient lidar odometry for real-time 3d mapping. IEEE Robot Autom Lett
87. Rosinol A, Abate M, Chang Y, Carlone L (2020) Kimera: an open-source library for real-time metric-semantic localization and mapping. In: 2020 IEEE international conference on robotics and automation (ICRA). IEEE, pp 1689–1696
88. Rublee E, Rabaud V, Konolige K, Bradski G (2011) Orb: an efficient alternative to sift or surf. In: International conference on computer vision, pp 2564–2571
89. Schubert D, Goll T, Demmel N, Usenko V, Stückler J, Cremers D (2018) The tum vi benchmark for evaluating visual-inertial odometry. In: 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 1680–1687
90. Schwarting W, Alonso-Mora J, Rus D (2018) Planning and decision-making for autonomous vehicles. Robot Auton Syst Ann Rev Control
91. Segal A, Haehnel D, Thrun S (2009) Generalized-ICP. In: Robotics: science and systems, vol 2, Issue 4. Seattle, WA, p 435
92. Serafin J, Grisetti G (2015) Nicp: dense normal based point cloud registration. In: 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 742–749
93. Shan T, Englot B (2018) ‘Lego-loam: lightweight and ground-optimized lidar odometry and mapping on variable terrain. In: 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 4758–4765
94. Shan T, Englot B, Duarte F, Ratti C, Rus D (2021) Robust place recognition using an imaging lidar. In: 2021 IEEE international conference on robotics and automation (ICRA). IEEE, pp 5469–5475
95. Shan T, Englot B, Meyers D, Wang W, Ratti C, Rus D (2020) Lio-sam: tightly-coupled lidar inertial odometry via smoothing and mapping. In: 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 5135–5142
96. Shan T, Englot B, Ratti C, Rus D (2021) Lvi-sam: tightly-coupled lidar-visual-inertial odometry via smoothing and mapping. arXiv:2104.10831

97. Siegwart R, Nourbakhsh IR, Scaramuzza D (2011) Introduction to autonomous mobile robots. MIT Press
98. Song H, Ding W, Chen Y, Shen S, Wang MY, Chen Q (2020) Pip: planning-informed trajectory prediction for autonomous driving. In: European conference on computer vision. Springer, pp 598–614
99. Sturm J, Engelhard N, Endres F, Burgard W, Cremers D (2012) A benchmark for the evaluation of RGB-d slam systems. In: IEEE/RSJ international conference on intelligent robots and systems. IEEE, pp 573–580
100. Sun P, Kretzschmar H, Dotiwalla X, Chouard A, Patnaik V, Tsui P, Guo J, Zhou Y, Chai Y, Caine B et al (2020) Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2446–2454
101. Tang TY, De Martini D, Newman P (2021) Get to the point: learning lidar place recognition and metric localisation using overhead imagery
102. Thrun S, Montemerlo M, Dahlkamp H, Stavens D, Aron A, Diebel J, Fong P, Gale J, Halpenny M, Hoffmann G et al (2006) Stanley: the robot that won the darpa grand challenge. J Field Robot 23(9):661–692
103. Wang Y, Jiang Z, Gao X, Hwang JN, Xing G, Liu H (2021) Rodnet: radar object detection using cross-modal supervision. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision, pp 504–513
104. Wang W, Zhu D, Wang X, Hu Y, Qiu Y, Wang C, Hu Y, Kapoor A, Scherer S (2020) Tartanair: a dataset to push the limits of visual slam. In: 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 4909–4916
105. Weiffenbach G (2019) Tropospheric and ionospheric propagation effects on satellite radio-doppler geodesy. In: Electromagnetic distance measurement. University of Toronto Press, pp 339–352
106. Wen W, Zhou Y, Zhang G, Fahandezh-Saadi S, Bai X, Zhan W, Tomizuka M, Hsu LT, Urbanloco: a full sensor suite dataset for mapping and localization in urban scenes. In: 2020 IEEE international conference on robotics and automation (ICRA). IEEE, pp 2310–2316
107. Wisth D, Camurri M, Fallon M (2022) Vilens: visual, inertial, lidar, and leg odometry for all-terrain legged robots. IEEE Trans Robot
108. Wu KJ, Guo CX, Georgiou G, Roumeliotis SI (2017) Vins on wheels. In: 2017 IEEE international conference on robotics and automation (ICRA). IEEE, pp 5155–5162
109. Xu W, Zhang F (2021) Fast-lio: a fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter. IEEE Robot Autom Lett 6(2):3317–3324
110. Xu W, Cai Y, He D, Lin J, Zhang F (2022) Fast-lio2: fast direct lidar-inertial odometry. IEEE Trans Robot 38(4):2053–2073
111. Yang B, Zhang Q, Geng R, Wang L, Liu M (2022) Real-time neural dense elevation mapping for urban terrain with uncertainty estimations. IEEE Robot Autom Lett 8(2):696–703
112. Yang S, Choset H, Manchester Z (2022) Online kinematic calibration for legged robots. IEEE Robot Autom Lett 7(3):8178–8185
113. Ye H, Chen Y, Liu M (2019) Tightly coupled 3d lidar inertial odometry and mapping. In: 2019 IEEE international conference on robotics and automation (ICRA)
114. Ye H, Huang H, Liu M (2020) 'Monocular direct sparse localization in a prior 3d surfel map. In: 2020 IEEE international conference on robotics and automation (ICRA). IEEE, pp 8892–8898
115. Yin J, Li A, Li T, Yu W, Zou D (2021) M2dgr: a multi-sensor and multi-scenario slam dataset for ground robots. IEEE Robot Autom Lett 7(2):2266–2273
116. Yin H, Wang Y, Xiong R (2021) Improved radar localization on lidar maps using shared embedding. arXiv:2106.10000
117. Yokozuka M, Koide K, Oishi S, Banno A (2020) Litamin: lidar-based tracking and mapping by stabilized ICP for geometry approximation with normal distributions. In: 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 5143–5150

118. Yokozuka M, Koide K, Oishi S, Banno A (2021) Litamin2: ultra light lidar-based slam using geometric approximation applied with kl-divergence. In: 2021 IEEE international conference on robotics and automation (ICRA). IEEE, pp 11 619–11 625
119. Yuan C, Xu W, Liu X, Hong X, Zhang F (2022) Efficient and probabilistic adaptive voxel mapping for accurate online lidar odometry. IEEE Robot Autom Lett 7(3):8518–8525
120. Yu Y, Gao W, Liu C, Shen S, Liu M (2019) A gps-aided omnidirectional visual-inertial state estimator in ubiquitous environments. In: 2019 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 7750–7755
121. Zhang Z (2000) A flexible new technique for camera calibration. IEEE Trans Pattern Anal Mach Intell 22(11):1330–1334
122. Zhang J, Singh S (2018) Laser-visual-inertial odometry and mapping with high robustness and low drift. J Field Robot 35(8):1242–1264
123. Zhang Z, Scaramuzza D (2018) A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In: 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 7244–7251
124. Zhang J, Singh S (2014) Loam: lidar odometry and mapping in real-time. In: Robotics: Science and Systems, vol 2, p 9
125. Zhang J, Singh S (2015) Visual-lidar odometry and mapping: low-drift, robust, and fast. In: 2015 IEEE international conference on robotics and automation (ICRA). IEEE, pp 2174–2181
126. Zheng C, Zhu Q, Xu W, Liu X, Guo Q, Zhang F (2022) Fast-livo: fast and tightly-coupled sparse-direct lidar-inertial-visual odometry. In: 2022 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 4003–4009
127. Zhou Y, Gallego G, Shen S (2021) Event-based stereo visual odometry. IEEE Trans Robot 37(5):1433–1450
128. Zhou X, Zhu J, Zhou H, Xu C, Gao F (2021) Ego-swarm: a fully autonomous and decentralized quadrotor swarm system in cluttered environments. In: 2021 IEEE international conference on robotics and automation (ICRA). IEEE, pp 4101–4107
129. Zhu AZ, Thakur D, Özaslan T, Pfrommer B, Kumar V, Daniilidis K (2018) The multivehicle stereo event camera dataset: an event camera dataset for 3d perception. IEEE Robot Autom Lett 3(3):2032–2039
130. Zhu Z, Peng S, Larsson V, Xu W, Bao H, Cui Z, Oswald MR, Pollefeys M (2022) Nice-slam: neural implicit scalable encoding for slam. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 12 786–12 796
131. Zou Z, Li L, Hu X, Zhu Y, Xue B, Wu J, Liu M (2023) Robust equipment-free calibration of low-cost inertial measurement units. IEEE Trans Instrum Meas
132. Zuo X, Geneva P, Lee W, Liu Y, Huang G (2019) Lic-fusion: lidar-inertial-camera odometry. In: 2013 IEEE/RSJ international conference on intelligent robots and systems. IEEE

Chapter 8
Visual SLAM for Texture-Less Environment

Yanchao Dong, Yuhao Liu, and Sixiong Xu

Abstract Recent researches on vision-based self-localization have catalyzed versatile and reliable real-time Visual Simultaneous Localization and Mapping (VSLAM) systems. However, retrieving ground truth, estimating calibration parameters and annotating useful labels all require cumbersome human labor. Moreover, there are lots of object instances in the environments while traditional mapping modules can only estimate 3D information of isolated sparse or semi-dense feature points. To meet the gap between the above requirements, we present a VSLAM method based on a synthetic dataset which can effectively utilize texture-less object instances. We also propose several new evaluation criteria that can fully take advantage of ground truth and annotations from synthetic datasets. The proposed Visual SLAM method includes newly designed feature extraction, matching, localization and mapping modules, which jointly use object features and point features to estimate camera 6-Degrees Of Freedom (6-DOF) poses and do richer map construction. Experiments are conducted using the proposed datasets and criteria with several state-of-the-art VSLAM methods to demonstrate the functionality of our datasets. Owing to the object feature fusion in the co-visibility graph, it can conducts scale aware bundle adjustments to reduce accumulated errors. The advantages of proposed Visual SLAM method are demonstrated through experiments conducted both on synthetic datasets and real-world datasets.

Keywords Synthetic dataset · Visual SLAM · Evaluation criteria · Localization · Mapping · Object pose

Y. Dong · Y. Liu · S. Xu (✉)
College of Electronic and Information Engineering, Tongji University,
Shanghai 201804, People's Republic of China
e-mail: xsx1910635@tongji.edu.cn

Y. Dong
e-mail: dongyanchao@tongji.edu.cn

Y. Liu
e-mail: 2211271@tongji.edu.cn

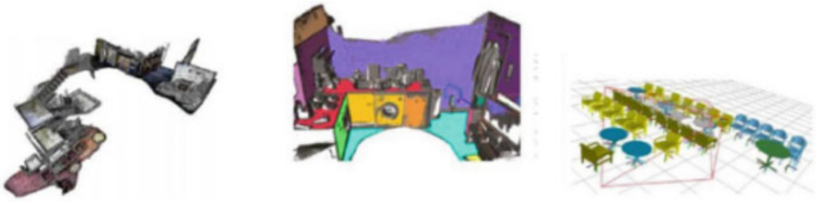
R. Fan et al. (eds.), *Autonomous Driving Perception*, Advances in Computer Vision
and Pattern Recognition, https://doi.org/10.1007/978-981-99-4287-9_8

8.1 Introduction

With the rapid development of computer vision and the low cost of visual sensors, Visual Simultaneous Localization and Mapping (VSLAM) methods have been paid special attention for localization and navigation applications such as unmanned cars in intelligent transportation industry and automated guided vehicles (AGV) in logistics industry, and have developed into a relatively well-established theoretical system.

VSLAM system only uses image sensors (cameras) for map construction and self-positioning. According to the number and type of image sensors, VSLAM can be divided into monocular camera SLAM, binocular (stereo) camera SLAM, RGBD camera SLAM, multi-camera system SLAM, etc; According to the density of image features used in low-level processing, it can be divided into sparse SLAM, semi-dense SLAM and dense SLAM systems, which are shown in Fig. 8.1.

The chapter is organized as follows. Section 8.2 details current state-of-the-art VSLAM algorithms using lines, planes and objects as features. Section 8.3 shows the VSLAM datasets which include real-world recorded datasets and computer graphic based methods. Section 8.4 illustrates the details of the implementation and design of this dataset, and provides the theoretical basis for the subsequent experiments. Moreover, this section details our VSLAM pipeline from the perspective of modules in the proposed VSLAM pipeline and proposes our parameterization pipeline on keyobjects extracted from actual images. Section 8.5 proposes a renovated VSLAM testing pipeline and the corresponding test results and presents experiments on localization accuracy and map construction performance. Sections 8.6 and 8.7 conclude this chapter and discusses future developments of the proposed VSLAM methods.



(a) Dense VSLAM system map

(b) Map created by VSLAM system using plane and surface features

(c) Map created by VSLAM system using feature markers

Fig. 8.1 Typical dense maps. **a** Dense map of staircase room obtained from dense point cloud data after fusion, rendering and artistic processing. **b** Dense interior map based on plane and surface panel. **c** Dense conference room map built with feature markers

8.2 State-of-the-Art Visual Simultaneous Localization and Mapping Algorithms

Most modern simultaneous localization and mapping (SLAM) methods aim to recover camera trajectory and 3D reconstruction of surroundings from a sequence of images in real-time. Various research fields including autonomous driving, robotics and augmented reality use SLAM as their base infrastructure in navigation and guidance. In the vast variety of SLAM methods, methods that use one or several image sensors (i.e.Visual SLAM (VSLAM)) have been paid special attention; in recent years, researchers have proposed several reliable, versatile and accurate VSLAM systems [1–4]. These modern VSLAM methods mostly consist by three modules:

1. **Feature extraction** module generates and manages a group of image features from the image;
2. **Localization** (i.e.*camera tracking*) module estimates camera poses from a group of images by feature matching;
3. **Mapping** (i.e.*map construction*) module builds up, manages and adjusts an organized form of local or global 3D map in various ways, including loop correction.

The feature extraction module is the key functionality that VSLAM can accommodate to environmental preconditions. Typical VSLAM methods exploit point features in this module since they are easy to describe and manage; there also exist works that use alternative features, mostly because these features are better at describing that particular kind of environment. For instance, in texture-less environments like building hallways, the quantity of available feature points may be insufficient and the repeatability of image features may downgrade the match quality, which influences the overall localization accuracy of the VSLAM system. However, these texture-less environments are widely distributed in GPS-denied areas where VSLAM methods are most needed, which poses challenges for VSLAM systems. Features suitable for texture-less environments should be able to preserve as much structural information as possible with limited quantity. Three kinds of features are relatively common: lines (edges), planes and objects.

8.2.1 Lines and Planes

Lines and planes are welcoming VSLAM features as man-made structures satisfying Manhattan world assumption, which helps simplify both the feature sampling process and feature optimization process. The enhanced description of structural information embedded in line and plane features allow these systems achieve better mapping and localization accuracy in texture-less environments.

Works like REBVO [5] use direct methods on edge pixels to achieve edge-mapping in real time. Similar methods are used in works [6, 7]. Those two works both use the combination of image intensity and one other image description technique in direct image alignment: the former work uses the combination of image intensity and geometry error between pixels and edges, and the latter work uses weighted sum of image intensity error and image distance field error. PL-SVO [8] uses sparse image alignment both between point and line features to retrieve camera poses which is an extension of the famous SVO [2]. Some works [9, 10] use a different approach: the extracted lines (edges) are matched like point features using specific line matcher, which allows authors to perform mature techniques like bundle adjustment from point-feature VSLAM on line features. Besides the above works that try to adopt lines as VSLAM features, we've also seen works [11, 12] that estimate camera poses using Manhattan World Assumption which fully take advantage of special characters of man-made environments. Usually these methods achieve better localization accuracy as compared to non-constrained methods.

Typical plane-based VSLAM systems [13, 14] use planes sampled from point clouds of RGBD camera as features for tracking or mapping. These methods either utilize local or global depth regularization like work [15] by Salas-Moreno et al. or use direct methods like CPA-SLAM [13], or the combination of direct methods with geometric optimization of 3D planes like works [14, 16]. Liwicki et al. provides a method [17] that uses monocular image without depth sensors to detect and formulate planes on keyframe. The method can operate in large-scale as compared to those small-scale VSLAM systems mentioned above. Experiments proposed in this work show that it has better robustness than LSD-SLAM, but with higher local depth noise and it can only operate in environments in which most of the surfaces are planar.

As we have discussed above, these methods have advantages both on camera localization and map construction, but some of the existing methods treat lines and planes as special features extraction tools useful in man-made environments, which cannot fully utilize the structural and semantic information processed in these features. Some methods can use the structural information embedded in certain environments, but they only focus on achieving accurate localization in low-speed small-scale environments. Besides, the robustness and performance of the state-of-the-art line feature detectors and feature descriptors cannot match the performance of point feature detectors and feature descriptors, which will influence the overall performance of VSLAM system.

8.2.2 Objects

VSLAM methods that use objects as features are often referred as **object-oriented VSLAM** or **object-level VSLAM** [18–20]. One of the earliest works that focuses on object-level mapping is SLAM++ [21], this method uses the reconstruction mechanism of the famous KinectFusion to establish models of objects, and proposes a complete framework for model-based object matching, important object insertion, pose estimation and object-level graph optimization. Several methods share similar topology, for instance, work [22] also uses pre-integrated or pre-defined models for object tracking; this work aims for establishing a point cloud map of camera's surroundings with labeled object identity. To achieve this, this work combines output of two different CNNs to jointly detect objects and estimate object poses, where the pre-integrated object model is used. Another approach is using special representation or parameterization of objects, normally using output results from state-of-the-art object detector. A series of works about QuadricSLAM [18, 19] represent objects in 9-DOF quadric form; similar methods have been found in works [23, 24] from Hosseinzadeh et al. which have extended quadric representation of objects onto a unified representation that contains objects, planes and 3D points, and have introduced joint optimization method that operates on local structural information and semantic information at the same time. CubeSLAM [25] from Yang and Scherer proposes a method to actively match 9-DOF quadric represented objects with possible poses from a limited range.

Current works on object-level VSLAM mostly focus on parameterization of important objects using various kinds of techniques; these methods either rely on the outputs of object detection networks or using reconstructed object models, thus they are better in describing common objects in ordinary environments. We haven't discovered works that pay special attention to those texture-less, yet very organized objects, which are very important landmarks in driving scenarios.

A wide group of reliable and versatile VSLAM[1] methods that use one or several image sensors [1, 2, 26, 27] or use the fusion of cameras and other sensors [28–31] have been widely studied and applied. There are several kinds of information needed in the development of VSLAM systems which should be contained in that dataset: camera poses, calibration parameters, annotations, images, distortion simulation.

In recent years, using special software [32] to simulate various aspects of vehicles has already been a common practice both in academic research and modern automobile industry. By taking advantage of the flourishing computer graphic industry, we can generate large amount of photorealistic synthetic images, accompanied by enough information for wide-ranged evaluation and validation tasks in VSLAM systems.

8.3 The VSLAM Dataset

This section illustrates the VSLAM datasets, and justifies the need for scientific research and the feasibility of the proposed solution.

[1] Many works that claim themselves as "Visual Odometry" possess many features similar to SLAM system; therefore we refer both VO and Visual SLAM as "VSLAM" in the following paper.

8.3.1 Real-World Recorded Datasets

Nowadays, images from most publicly available datasets for use in Computer Vision topics are retrieved by camera recording in real world. The ground truths are captured and synchronized with the camera by other instruments. For VSLAM datasets or benchmark systems, the four kinds of most widely used information are camera images, IMU data, camera poses and frame depth maps. The first two are often used as VSLAM input, while the last two are often used as ground truth.

8.3.1.1 Images and IMU Data

Numerous VSLAM datasets supply stereo camera image data [33–35] synchronized with external sensors; there also exist datasets that only provide monocular data [36]. Some works [37, 38] provide fisheye camera image data or omnidirectional camera image data, but only work [33] provides fisheye stereo pairs. For retrieving IMU data, most existing methods either utilize IMU modules [34, 39, 40] or use bundled IMU in smart phones or tablets [41] or the combination of both [38].The IMU data frames are recorded at higher rate as compared to camera images, and the state-of-the-art synchronization techniques that can sync the timestamp of camera frames and IMU frames can be found in [38].

8.3.1.2 Camera Positions

State-of-the-art SLAM datasets use devices like GNSS positioning systems [34, 35, 40, 42] or industrial motion capture systems [33, 37, 43] to retrieve ground truth camera 6-DOF poses. Using IMU integration results modified by exterior position fixes [41] is also a widely-used approach.

8.3.1.3 Frame Depth Maps

Retrieving accurate 3D information of pixels on the frame from real-world scenes is a difficult task and only a few existing works carry such information. Typical works like [34, 40, 42] use laser scanner (or LIDAR) to retrieve 3D information of camera's surroundings; there also exists another way that uses a multi-station [43] to get camera position and point cloud at the same time. However, these two approaches share the same drawback that the instruments are expensive and the application scopes are limited.

As we analyzed above, typical real-world scene datasets require much labor to retrieve data, and the amount of available information is limited. Also, even if researchers find a way to retrieve enough information using different instruments with neglectable measurement errors (in VSLAM research), errors will still be introduced during alignment and calibration parameter estimation. A detailed comparison regarding the information provided by some publicly available datasets can be found in Table 8.1.

8.3.2 Computer Graphic Based Methods

8.3.2.1 Rendered Datasets

Rendering photorealistic images based on virtually established real-world scenes for training and testing various computer vision algorithms like semantic segmentation, has proved to be an ideal way both for accuracy and efficiency [45]. Recent advances in generative adversarial network on traffic vision research suggest that using artificial scenes to train neural network helps enhance accuracy on object detection and semantic segmentation [46].

8.3.2.2 Simulators

Using powerful computer graphic engines such as Unreal Engine[2] or Unity3D,[3] researchers can build up simulators that simulate how objects move and react with certain scenes. Typical open-source works like AirSim [47] and UnrealCV [48] can simulate different scenarios with high definition images and multiple sensor outputs; besides, it can simulate how objects react when you enter control commands, thus enabling researchers to test a large robot control system consisting of localization, route planning and motion control at the same time.

Compared to all the present works mentioned above, our proposed work focuses more on providing essential environmental factors that effect how VSLAM operates. Detailed comparison of data entries provided between different datasets and simulators is illustrated in Table 8.1.

Based on the techniques mentioned above, we introduce "Tongji Computer Graphic" (*TCG*) dataset, a synthetic dataset for VSLAM evaluation. This dataset contains multiple sequences in different indoor environments with more than 15k images in total and the render pipeline is shown in Fig. 8.2.

Using the data mentioned above, we also propose an evaluation framework for testing the performance of VSLAM systems. For researchers and developers, except for trivial usages like using special data from TCG dataset on various computer vision research topics or using the framework to evaluate VSLAM algorithms, the TCG dataset can be particularly suitable for two kinds of tasks in the development of a versatile and robust VSLAM system:

[2] https://www.unrealengine.com/.

[3] https://unity3d.com/.

Table 8.1 Comparison of some publicly available datasets. In this chart, “Depth Raw” represents depth from noisy sensors like RGBD camera or ultrasonic radar; “Depth truth” represents depth from precise sensors like Lidar or rendered by software. The I-Net dataset does not have object poses as ground truth, but with object bounding boxes, so it is marked as “×”

Type	Real	Real	Real	Real	Real	Real	Render	Render	Real&Render	Render
Dataset	EuRoC [43]	KITTI [42]	TUM [36]	OxIOD [37]	TUMVI [33]	ADVIO [41]	I-Net [44]	SYN [45]	VISMA [22]	Proposed
Environment in/outdoor	In	Out	In/Out	In	In/Out	In	In	Out	In	In
Low-texture environment	√	×	√	√	√	√	×	×	×	√
Motion blur	√	×	×	×	×	×	×	√	×	√
Depth of field effect	×	×	×	×	×	×	×	×	×	√
Fisheye	×	×	√	√	√	√	√	×	×	√
Stereo pairs	√	√	×	×	√	×	√	×	×	√
IMU raw	√	√	×	√	√	√	√	×	√	×
Depth raw	×	×	×	×	×	√	√	×	×	√
Depth truth	√	√	×	×	×	×	√	√	×	√
Edge map	×	×	×	×	×	×	×	×	×	√
Optical flow truth	×	×	×	×	×	×	√	√	×	√
Segmentation truth	×	×	×	×	×	×	√	√	×	√
Objects truth	×	√	×	×	×	×	×	×	√	√

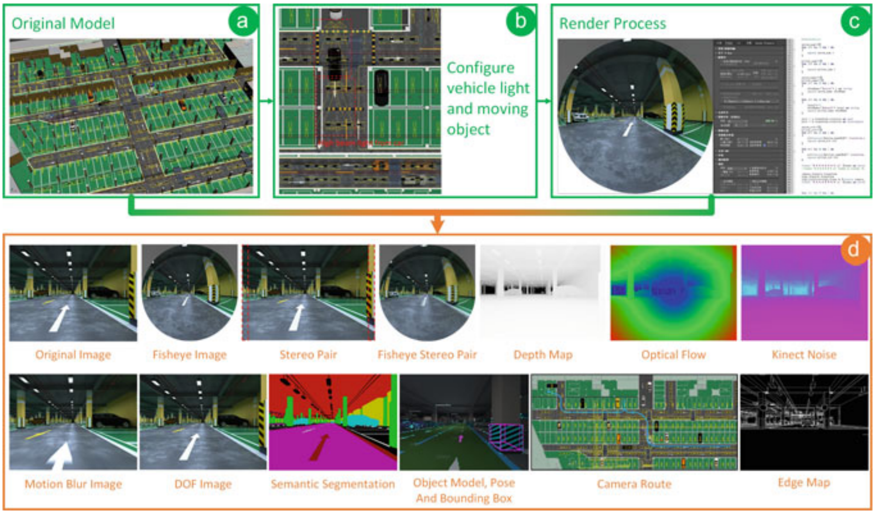
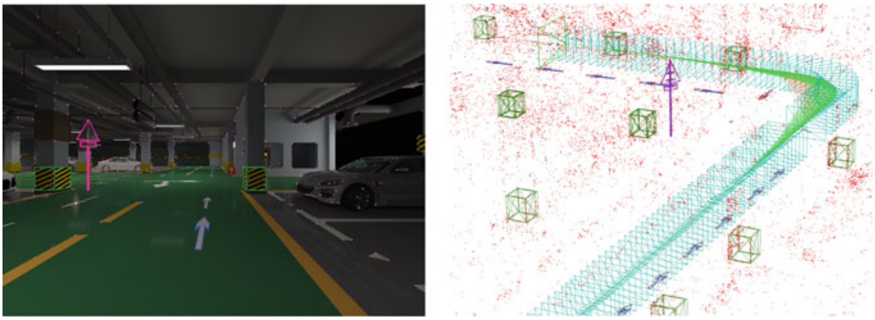


Fig. 8.2 Pipeline of proposed work. **a** We establish 3D model of underground garage as our environment. **b** We configure moving patterns of moving vehicles and challenging high beam light from vehicles on the scene. **c** We render several elements and retrieve information in the box below using mechanism in 3DS-MAX and V-Ray. **d** Examples for all supported ground truth data in our dataset

1. **Algorithm validation.** For instance, researchers can use this dataset to validate VSLAM algorithms in special environments or under special preconditions. By taking advantage of the vast variety of resources from the 3ds-max community, researchers can retrieve images and ground truth in some special environments without much human labor. Also, when trying to determine the superiorities and inferiorities of feature selection techniques.
2. **Debugging VSLAM algorithms.** As modern VSLAM systems are often composed of several modules with different functionality, and one module may malfunction by interacting with other modules, and this is difficult to discover. By substituting information estimated by VSLAM system into ground truth data from the dataset (e.g.using ground truth optical flow to replace 2D feature matching results), we can eliminate the effect of interaction among different modules, which helps us debug this algorithm.

Moreover, we propose in this work a monocular object-oriented VSLAM method that focuses mainly on utilizing texture-less object instances such as traffic signs to accurately localize camera itself and reconstruct camera's surroundings. By matching the projection of rasterized CAD model on the image to the image's distance field, we estimate the transformation that maps raster points from object coordinate system to frame coordinate system. Utilizing these geometric information, the VSLAM system builds and maintains a global VSLAM map which contains both point features and object features, which allows the system to establish point-object joint covisibility



(a) One selected frame showing matched mapobjects in purple and matched mappoints in cyan

(b) Drawing of reconstructed point-object joint map

Fig. 8.3 Demonstration of the proposed VSLAM system. **a** One frame from the VSLAM system. The image used in evaluating this VSLAM system is rendered using Computer Graphics (CG) model of an underground garage. The lines with different colors are image projection of rasterized object models. We use purple to represent traffic signs, green to represent recognizable pillars and blue to represent arrows. **b** The 3D view of the reconstructed map in VSLAM system. The colors of objects on the scene are identical to colors from **a**. We use green lines between keyframes (boxes in cyan) to represent point covisibilities, orange lines to represent object covisibilities

graph containing keyframe associations established by both the common keypoints and keyobjects. We also conduct experiments that compare the localization accuracy and map construction performance of the proposed method with state-of-the-art VSLAM methods both on synthetic datasets (for analytical performance evaluation) and real-world datasets (for functionality validation). The effects of the proposed VSLAM method running on one exampled photorealistic synthetic dataset can be found in Fig. 8.3, and the effects of the proposed VSLAM method on one exampled real-world dataset can be found in Fig. 8.18.

The pipeline of the proposed VSLAM method is shown in Fig. 8.4. The main contributions of this work are listed as follows:

1. A large and rich large-scale dataset with enough information for validating, evaluating and debugging VSLAM systems, especially large amount of data in challenging low texture environments.
2. A new VSLAM evaluation method with detailed rating criteria for mapping block of VSLAM. The criteria are listed below:

    a. **Reconstruction Error (RE)** is the average error of world positions of 3D features on keyframes across entire global map;
    b. **Reconstruction Error Standard Deviation (RE_STD)** is the average standard deviation of **Reconstruction Error (RE)** across entire global map;
    c. **Cruciality of Features (CF)** measures the percentage of features lying on static area of the surroundings;

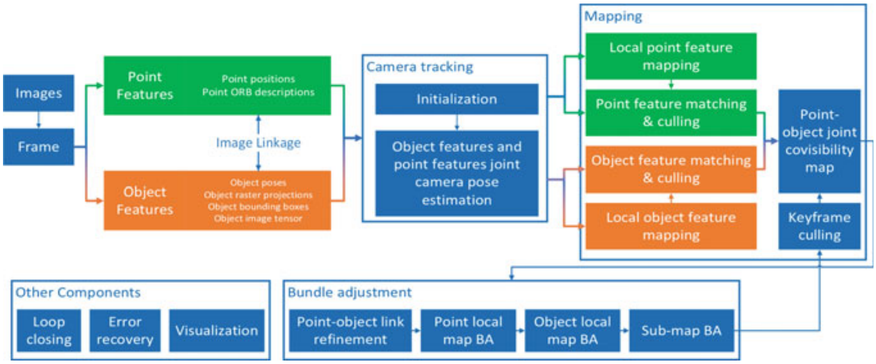


Fig. 8.4 Pipeline of proposed work. We mark components related to point features in green, components related to object features in orange, and components that need information from both in blue

  d. **Significance of Features (SF)** measures how well features sampled by VSLAM describe general shape of the objects and structures in the surroundings.
  e. **Outlier Removal Capability (ORC)** measures VSLAM's capability to remove outliers sampled from moving objects.

3. Tests of current state-of-the-art VSLAM systems with different topology on camera tracking and mapping based on proposed dataset and rating criteria, which validate both the usability and functionality of the proposed rating criteria and the proposed datasets.
4. We propose in this paper an object-oriented VSLAM method which consists of several modules:

  a. Feature extraction module which detects and describes point and object features;
  b. Camera localization module which jointly uses point and object features to estimate camera 6-DOF pose;
  c. Map construction module which is able to construct and maintain a point-object joint covisibility map.

5. Traditional way of establishing covisibility map consists of associating two keyframes with common image features. The proposed method can associate two keyframes with shared object observations, which extends the capability of connecting two keyframes to being able to associate two distant keyframes that do not share common image features. Using the extra constraints provided by these associations, the proposed system can conduct scale aware bundle adjustments to revise the accumulated error.
6. We provide details on how we convert CAD models of keyobjects on the image into the form recognizable by VSLAM system, how to detect keyobjects and

describe their image appearance, how we estimate object poses and match related objects under this particular form.

8.4 Pipeline of the VSLAM Algorithms

8.4.1 Details of Datasets

The proposed pipeline in Fig. 8.2 illustrates how we build models and retrieve useful ground truth. We further detail how we implement this pipeline in the following parts of this section.

8.4.1.1 Data Specification

As we have stated in Sect. 8.2, all sequences in our dataset are obtained based on simulated motion of a vehicle moving in a large virtual underground garage model. We focus on using garage models to render sequences in this dataset for three reasons:

1. Videos or image sequences recorded in garages often contain the following distinguishable features that usual environments cannot provide:
    a. Large number of texture-less images.
    b. Large number of straight lines and planes.
    c. The image appearance and image features are often repetitive: two very different places may actually have the same image appearance.
2. Driving a vehicle into one parking lot often needs continuously rotative motion, which is challenging for monocular VSLAM system.
3. So far we have not discovered datasets that contain images recorded (rendered) in garages or in environments with similar distinguishable features like garages.

We use 3DS-Max plugin MadCar[4] to offer animations of virtual wheeled vehicles. We also use this software to simulate motion of moving vehicle set as obstacles in the scene. In Fig. 8.5, we demonstrate paths of camera of four example sequences from the proposed dataset. All the scene models used for rendering the sequences are built in Autodesk 3DS-MAX[5] and images are rendered by V-Ray[6] industrial off-line renderer. V-ray uses path-tracing techniques to simulate realistic global illumination in order to render high-quality photorealistic images. All sequences are rendered based on simulated motion of a virtual car exploring the scene, on which head a front-looking camera with viewing angle of approximately 90 degrees is attached.

[4] https://rendering.ru/madcar.html.

[5] https://www.autodesk.com/products/3ds-max/overview.

[6] https://www.chaosgroup.com/vray/3ds-max.

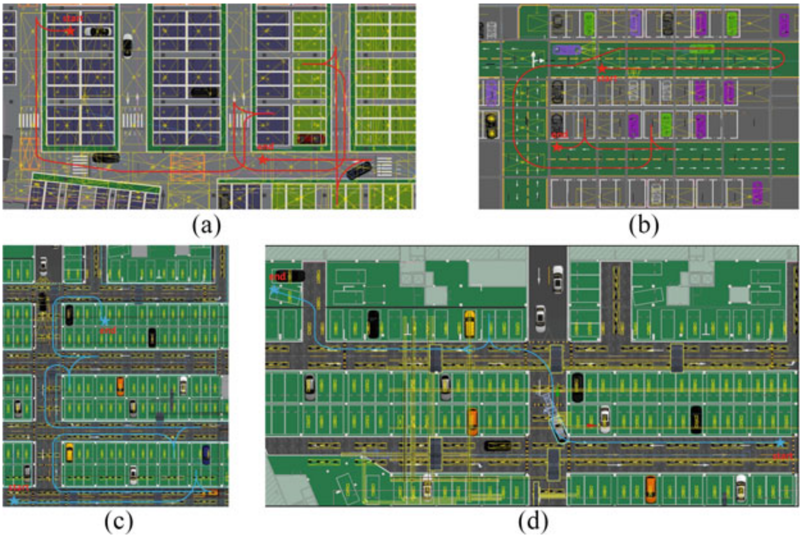


Fig. 8.5 **a**–**d** are top view of routes from four selected sequences from the proposed dataset. **a**, **d** The two sequences are pure static environments. **b** This sequence contains several high-beam lights but no moving vehicle. **c** This sequence contains one moving vehicle and several high-beam lights

Considering the render pipeline shown in Fig. 8.4, each sequence possesses the following contents:

1. Camera 6-DOF pose for each frame in world coordinate system.
2. Already undistorted wide-angle images and fisheye images.
3. Bounding boxes for significant objects on each frame, and these objects' relative transformation in camera coordinate system.
4. Depth for each pixel in each frame.
5. Semantic/instance segmentation for each frame.
6. Edge map of object borders for each frame.
7. Optical flow from frame F_N to the next frame F_{N+1}.
8. Camera intrinsic parameters both for pinhole camera model and fisheye camera model.
9. Stereo image pairs both for wide-angle images and fisheye images.
10. Images with motion blur effect and depth-of-field effect.

8.4.1.2 Semantic Segmentation and Depth Maps

The ground truth edge maps, semantic segmentation results and depth maps are rendered at the same time with the image, yielding accurate annotations per-pixel. We also generate noisy depth by simulating a real Kinect mechanism using the method from work [49]. By using these accurate annotations along with camera 6-DOF poses, we can generate accurate scene flow between two successive frames $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$ using relative pose between two frames:

$$p_2 = \pi(R, t, \pi^{-1}(p_1, d_1)), \tag{8.1}$$

where $\boldsymbol{p_2}$ is the corresponding pixel position of pixel $\boldsymbol{p_1}$ from $\boldsymbol{F}_1$ in $\boldsymbol{F}_2$, $\boldsymbol{d_1}$ is the depth of pixel in that frame, $\boldsymbol{R}, \boldsymbol{t}$ denote relative transformation between two frames, and $\boldsymbol{\pi}()$ denotes the projection function for this particular camera model. However, motions of the camera will introduce changes on relative viewing position between objects, which make the visibility of a particular pixel not only determined by field-of-view of the camera, but also by possible shading from other pixels. To resolve this issue, we check if the depth of pixel $\boldsymbol{p_2}$ rendered in $\boldsymbol{F}_2$ is smaller than the depth estimated by relative motion; smaller depth means the pixel $\boldsymbol{p_2}$ is shaded by pixels from another object.

For simulated fisheye camera images, we use spherical camera model [50] to calculate 3D information of pixels with theoretically calculated camera intrinsic parameters; we also provide rendered images with pre-integrated checkerboards, allowing camera calibration using other techniques or camera models.

8.4.1.3 Rendering Sequences

The luminance of the virtual scene is controlled by pre-integrated lights, and the light in the environments of every datasets has even distribution. To simulate high-beam light of the vehicle, we insert lights with high luminance and parallel beams on corresponding places of vehicle headlights.

The four sequences shown in Fig. 8.5 contain about 16k images with total length of about 1km. All images have a resolution of 1280 × 960 and horizontal field of view of roughly 89.4 degrees. Detailed statistics of the four datasets can be found in Table 8.2. The main differences between the four sequences (besides the trivial differences like image number) are:

1. In sequence A, the floor of underground garage will reflect light, and no moving vehicle exists.
2. In sequence B, there are several high-beam lights and large variation in brightness of global illumination.
3. In sequence C, there are moving vehicles and several high-beam lights.
4. In sequence D, there are no high-beam lights brightness and there are no light reflections.

All sequences are rendered on 3 workstations equipped with 16-core AMD Ryzen Threadripper 1950× Processor and 64 GB RAM in parallel. For fast validation of algorithms, we suggest using lower subdivision number: the image quality can still be preserved for overall impression but the rendering time for each image can be

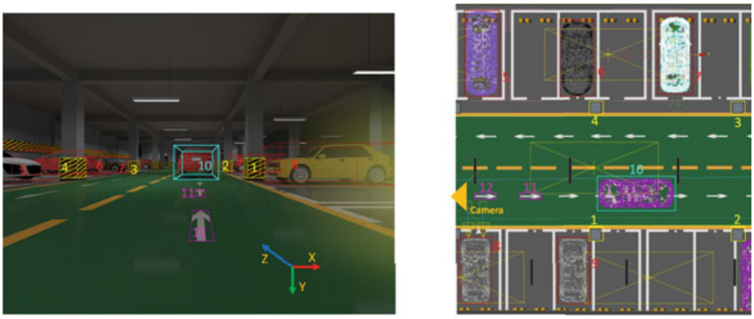
Table 8.2 Statistics of the four sequences. The A~D here corresponds to the A~D from Fig. 8.5

Dataset	A	B	C	D
Length of vehicle route	501 m	163 m	252 m	101 m
Number of images	6000	4000	4600	1300
Number of labeled vehicles	4	18	16	12
Number of labeled moving vehicles	×	×	2	×
Number of labeled traffic signs	18	74	30	13
Number of labeled pillar or structure	19	23	50	25
Number of labeled special characters	51	×	×	×
Number of labeled speed bumps	×	×	7	5

shortened to less than 60 s, which enables us to render a test sequence containing about 1500 images on the workstation mentioned above in roughly 1 day.

8.4.1.4 Key Objects

Our virtual garage models include the most important environmental elements in garages, such as a large variety of different pavements, lane markings, piping and tubing, different kinds of lights, traffic signs, parking lots, moving and static vehicles and speed bumps that force the car to slow down. The property that the virtual models are scalable means we can insert any wanted high-precision models into the model and generate useful poses, labels and annotations without extra effort. These specially inserted models are referred to as **keyobjects**. We divide important keyobjects on the scene into two categories: **signs** and **subjects**. Signs are noticeable marks or symbols in the scene. All signs will stay reasonably still for a relatively long time, which indicates they are static objects for mapping mechanism on VSLAM methods. Subjects are objects that are noticeable but cannot stay still for a long time. They might move after the camera has left the scene, thus introduce semi-static scenes when the VSLAM system try to use stored map of previously visited area to locate itself; or they are moving when the camera sees the object. Vehicles and human beings in most cases are subjects. Our dataset stores object information for each keyobject seen in each frame including instance segmentation results that indicate the class of the object, 3D bounding boxes for the object, precise object CAD model and whether the object is static or not. A vivid description of how keyobjects are labeled can be found in Fig. 8.6.



(a) Labeled keyobjects in one frame (b) Top view of the labeled objects

Index	1	2	3	4	5	6	7	8	9	10	11	12
X (m)	2.27	2.27	-5.23	-5.23	-8.14	-8.36	-8.26	4.39	4.78	-0.13	0.020	0.019
Y (m)	1.55	1.55	1.55	1.55	-6.64	-0.074	-0.029	-0.00068	-0.070	-0.070	0.65	0.66
Z (m)	9.07	17.47	17.47	9.07	7.95	13.76	19.10	7.71	18.56	2.27	5.18	2.54

(c) Coordinates of objects' center. The coordinate system are shown in (a)

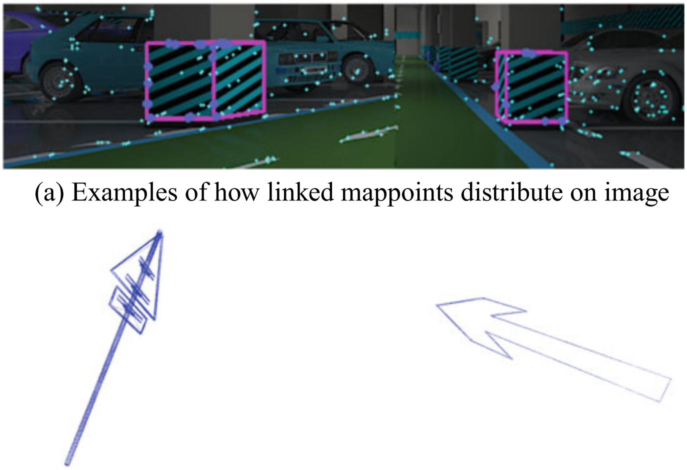
Fig. 8.6 We use red bounding boxes to label static cars and cyan bounding boxes to label moving cars, which are subjects; purple bounding boxes to label traffic signs (arrows in this frame) and yellow bounding boxes to label pillars and special objects, which are signs. The numbers on **a**, **b** are indexes for subjects and signs and are used in **c**

8.4.2 Visual SLAM Based on Keyobjects

We build our pipeline of object-oriented VSLAM based on previous work ORB_SLAM2 [3]. Similar to ORB_SLAM2, we divide our VSLAM system into several individual parts running in parallel threads: localization module for camera 6-DOF pose estimation, map construction module for establishing and managing an organized map representation, loop closing module for correcting loop, and several other blocks for error recovery and visualization. We further detail our VSLAM architecture in the following chapter.

8.4.2.1 Feature Extraction

Point features are detected using Features from Accelerated Segment Test(FAST) and described using ORB; object features are detected using random forest classification techniques [51]. We use rasterized model to describe keyobjects in VSLAM system which can be easily retrieved from object CAD model, and we sample **raster points** from these lines to finalize the parameterization process: the rasterized model consists of a group of discrete 3D points. We also borrow the idea of mappoints from [3] and establish point-object image links between mapobjects and mappoints: if image position of the observed mappoint is located on one mapobject's visible edges, the mappoint is considered linked to the mapobject. A vivid description of how features distribute on one typical frame can be found in Fig. 8.3; demonstration for linked



(a) Examples of how linked mappoints distribute on image

(b) Examples of projected objects using rasterized model

Fig. 8.7 **a** We use gray points to represent linked points and small blue dots to represent unlinked mappoints, the purple lines represent outline of the pillar in the image. **b** Demonstration of two objects: a traffic sign on the left and a guiding arrow on the right. Note that we have enlarged and rotated the two objects to achieve better viewing impression. Blue dots represent visible raster points

mappoints on the mapobject can be found in Fig. 8.7; demonstration of the rasterized objects can be found in Fig. 8.7.

8.4.2.2 Localization

Suppose that we already have a group of $\boldsymbol{n_o}$ keyobjects $\boldsymbol{O_i} \in \theta$ matched with a group of mapobjects $\boldsymbol{O_i^M} \in \theta$ and a group of $\boldsymbol{n_p}$ keypoints $\boldsymbol{p_i^m} \in \upsilon$ matched with a group of mappoints $\boldsymbol{P_i^M} \in \Upsilon$ on one frame, we can estimate the camera's 6-DOF pose $\boldsymbol{\nu_{rc}}$ and $\boldsymbol{s}$ at this frame by minimizing a sum of weighted reprojection error across all matched features:

$$\nu_{rc}, s = \arg\min \left(\sum_{i=1}^{n_p} \left(e_{p,i} \right) + \mu * \sum_{j=1}^{n_o} (e_{o,j}) \right), \tag{8.2}$$

μ represents weight of object reprojection error and will be estimated using object pose score. We define $\boldsymbol{\nu_{rc}}$ here as transformation that maps feature 3D position in world coordinate system to camera coordinate system.

In Eq. 1.2, error term $\boldsymbol{e_{p,i}}$ represents the error of mappoint reprojection, whereby we use same definition from ORB_SLAM2, and error term $\boldsymbol{e_{o,j}}$ represents the error of reprojecting one raster point from the object to the frame. We assume there exist

$\boldsymbol{n_r}$ raster points from one object, therefore by using wrap function $p = \omega(P, \nu_{rc})$ that maps 3D raster points to actual image plane, we get the disclosed form of object reprojection error as the sum of $\boldsymbol{n_r}$ raster points' reprojection error:

$$e_o = \sum_{k=1}^{n_r} (\omega(s * P_k, \nu_{rc} * \xi_w) - p_k), \tag{8.3}$$

where $\boldsymbol{P_k}$ is the 3D raster point from mapobject, $\boldsymbol{p_k}$ is the matched raster projection position on the image, $\boldsymbol{\xi_w}$ is the object pose that transforms raster point position from object coordinate system to world coordinate system, namely object-to-world transformation. This transformation can be calculated by combining the object-to-frame transformation and corresponding camera pose, and will be further optimized in the mapping module. Using the visibility buffer, we can get common visible points between two objects, which can be viewed as matched raster points between two objects.

8.4.2.3 Map Construction and Bundle Adjustment

Map construction module inserts new keyframes and new landmarks (mapobjects and mappoints in this work) into the map, possesses these keyframes and landmarks by constraints introduced by covisibility map, and performs bundle adjustments to revise errors in map. We mainly follow the keyframe and landmark insertion policy introduced in ORB_SLAM2 that inserts keyframes and features frequently and culls redundant ones. The organized map after culling are represented as a point-object joint covisibility map. Each node in this map is a regular keyframe, and an edge (i.e.association) between two keyframes exists if they observe enough common mappoints or share common mapobject observation. The mappoint associations link keyframes using both geometric and appearance-based constraints, while the mapobject associations link keyframes with geometric similarity and instance semantic information. The mapobject associations overcome the drawback that mappoint associations are influenced by image appearance; as soon as the detection and classification of landmarks are stable, we can build associations between keyframes that share the same mapobject observation but with different image appearance, which is demonstrated in Fig. 8.8.

After the above optimization, we will conduct a full bundle adjustment within extended sub-map to further adjust mappoints in these keyframes. Note that the optimization process mentioned above is limited within the extended sub-map, as we believe constraints calculated by object sub-map cannot truly influence the entire map; the full map optimization should be done by loop closure module.

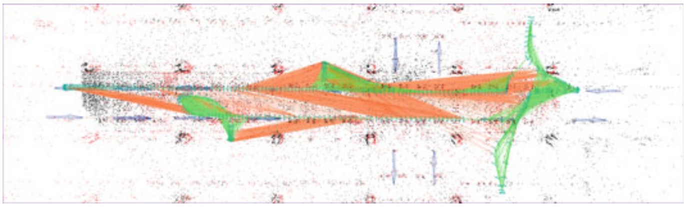


Fig. 8.8 Factor graph created by the proposed system running on one sequence in our synthetic dataset. We use orange lines to represent keyframe covisible associations created by object feature matching, and green lines to represent keyframe covisible associations created by point feature matching. The cyan squares are keyframes, the black and red dots are point features and the arrows and markers are object features

8.4.3 Keyobject Parameterization

As it has been stated in Sect. 8.4.1, the rasterized model of objects uses a group of organized lines exported from object CAD model to reveal actual object model, which perfectly suits our need as we are mostly dealing with texture-less objects. Besides, since most traffic signs are standardized and dimension-constrained, we can generate accurate line models using these fixed dimensional numbers, which can also be used for keyobject parameterization. One typical example of how rasterized object are represented on an actual image can be found in Fig. 8.7. We further detail how the proposed VSLAM system parameterizes raster models, and how to use this model to estimate object poses and match object instances, in the following sections.

8.4.3.1 Object Rasterization

The lines from CAD models will be sampled into raster points; curved edges will be divided into continuous small line segments and dealt with similar methods. For one straight line edge, starting from the start point $\boldsymbol{P_{start}}$ (i.e.$\boldsymbol{P_0}$), the current raster point $\boldsymbol{P_{i+1}}$ is calculated using its predecessor $\boldsymbol{P_i}$ ($\boldsymbol{i}$ starts from 0) using the following equation:

$$P_{i+1} = P_i + \tau(P_i) * dP \tag{8.4}$$

$\boldsymbol{\tau(P_i)}$ is the tangential direction vector of the edge starting from $\boldsymbol{P_{start}}$ and ending at $\boldsymbol{P_{end}}$, and $\boldsymbol{dP}$ is the "basic" sampling step which is defined as 1mm. During the tracking and map optimization process, we randomly take one point from three raster points to reduce reductant computation cost. We also calculate a visibility buffer that indicates which sampled raster points are visible under current object pose using the method from work [52].

8.4.3.2 Object Pose Estimation

The purpose of object pose estimation is estimating object pose by building object's CAD model and only one frame (with object fully or partly projected on that frame). In this work, we propose a light-weighted pipeline for estimating object pose (i.e.object-to-frame transformation). The pipeline for object pose estimation is demonstrated in Fig. 8.9. As shown in the figure, after we have determined existence of the object using classification tree technique [51], we first estimate an initial guess of the object pose with relatively large error; then we use the re-weighted distance transform based technique from the previous work D^{2}CO [53] to refine the initial guess and retrieve precise object poses. The pipeline requires only rasterized object model and one single calibrated image, thus providing object pose estimation for each frame individually.

After finishing pose refinement, we calculate score of object pose using the average weight from last iteration of optimization process of all visible pixels, which will be used to calculate the weight μ as shown in Eq. (8.2).

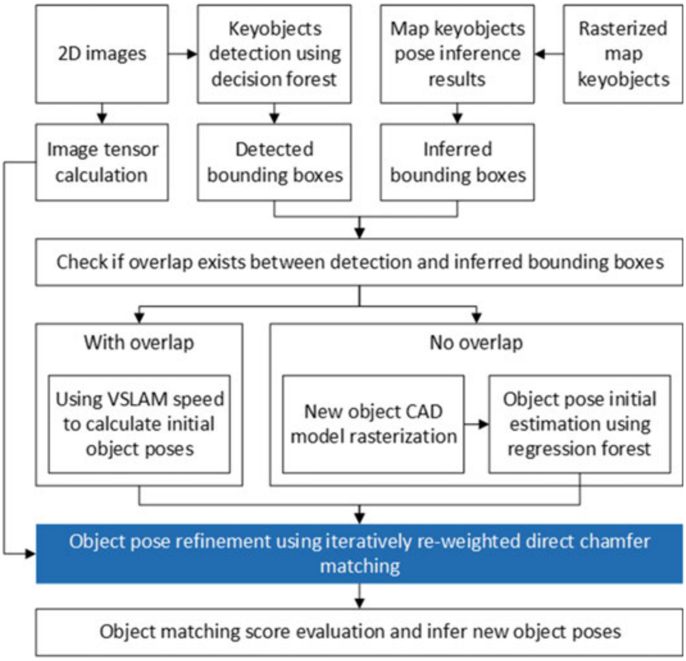

2D images
Keyobjects detection using decision forest
Map keyobjects pose inference results
Rasterized map keyobjects
Image tensor calculation
Detected bounding boxes
Inferred bounding boxes
Check if overlap exists between detection and inferred bounding boxes
With overlap
Using VSLAM speed to calculate initial object poses
No overlap
New object CAD model rasterization
Object pose initial estimation using regression forest
Object pose refinement using iteratively re-weighted direct chamfer matching
Object matching score evaluation and infer new object poses

Fig. 8.9 Pipeline of object pose estimation

8.4.3.3 Object Pose Estimation

For one keyobject $\boldsymbol{O}$ with know object pose $\boldsymbol{\xi}$ in F_1 and a group of n projected raster points $\boldsymbol{p_i} \in \psi$, the purpose of object matching is to find matched object from a group of keyobject (or mapobject) candidates θ located in F_2. However, texture-less objects (e.g.traffic signs) are often repetitive on image appearance, thus we cannot use unique descriptors to distinguish each individual object, and the range of searching for match among a group of candidates must be limited. To use this matching method, we assume here that objects are mostly individual instances with no overlap in 3D space. The proposed matching algorithm takes parts of strategy from semi-dense matching algorithms [1, 4]. The steps are as follows:

1. We downsample the point group ψ to form a new raster point group $\boldsymbol{q} \in \chi$, by default we randomly take one point from three raster points. Then we calculate epipolar line $\boldsymbol{L_{epi}}$ of the selected raster point $\boldsymbol{q}$ in F_2.
2. Every object candidate $\boldsymbol{O_{can}}$ that $\boldsymbol{L_{epi}}$ cut across is examined using the following three criteria:

   a. Whether the candidate and $\boldsymbol{O}$ have same object type;
   b. Whether the cross points fall on same edge;
   c. Whether the pixel intensity SSD error [4] between $\boldsymbol{q}$ and the cross points group ψ_{cross} falls within certain range.

   The number of matched points from $\boldsymbol{q}$ that satisfy all the three criteria is individually counted for each object candidate $\boldsymbol{O_{can}}$. The candidate that possesses more than 70% of the raster points from χ is the matched object of $\boldsymbol{O}$.

Comparing with the previous matching methods which mostly rely on image appearance or object feature descriptors, our method provides invariance on image appearance repetitiveness and do not rely on neural networks which are not computational cost-effective for objects. We demonstrate the object matching process between two distant frames with different image features but sharing same object observation in Fig. 8.10. Examples of how this method performs in real-world environments can also be found in Fig. 8.17.

8.5 Experimental Results

8.5.1 Evaluation of VSLAM Algorithms

The property that the proposed dataset contains photorealistic images and zero-error ground truth, makes the dataset a suitable platform for testing various SLAM algorithms. By fully taking advantage of the variety of generated data, we propose here a renovated VSLAM testing pipeline and the corresponding test results.

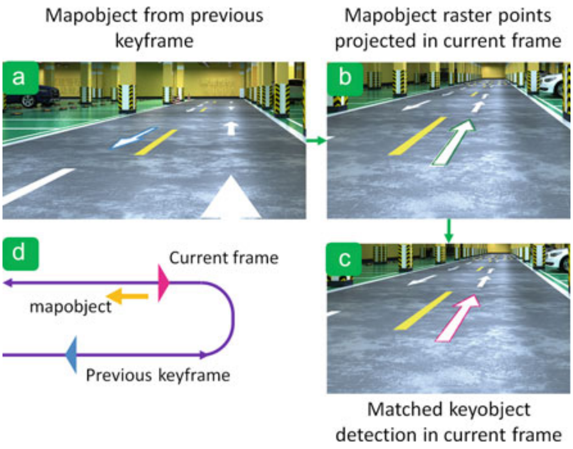

Mapobject from previous keyframe
Mapobject raster points projected in current frame
a
b
d
Current frame
mapobject
Previous keyframe
c
Matched keyobject detection in current frame

Fig. 8.10 Demonstration of object matching process. We use blue in **a** to represent projection of raster points from one mapobject from θ on one previous keyframe and purple in **c** to represent projection of raster points from one keyobject observed on current frame. Green in **b** are raster points projected from mapobject in the map to this frame; we use these projections in helping us to link the mapobject in **a** with the keyobject in **c**. Demonstration for the relative position between previous keyframe and current frame can be found in **d**. Note that the purple line in **d** are camera route simplified for demonstration

8.5.1.1 Evaluation Metrics

Current VSLAM evaluation frameworks mostly focus on camera localization error evaluation. For state-of-the-art VSLAM systems, the local map consists of a group of connected keyframes and the corresponding 3D features on these keyframes. Nevertheless, most VSLAM systems treat the mapping module as the auxiliary of camera localization, and the typical VSLAM map carries several flaws compared to state-of-the-art 3D reconstruction methods. We introduce several evaluation metrics in the following parts of this section that focus on evaluating how well VSLAM systems reconstruct their surroundings.

A. Localization Error

We use the Absolute Position Error (APE) and Relative Position Error (RPE) to describe localization error of camera positions. The APE is defined as:

$$E_{APE} = \frac{1}{m} \sum_{j=0}^{m} (\| \ (P_{ct} - \Delta * P_c) \ \|), \tag{8.5}$$

where m is the total number of keyframes generated by the VSLAM system, $\boldsymbol{P_c}$ is the world position of camera center, $\boldsymbol{P_{ct}}$ is the ground truth world position of camera center. $\boldsymbol{\Delta}$ is the rigid body transformation that map features from image coordinate system to real-world coordinate system estimated by methods in [54].

The RPE is defined as:

$$E_{RPE} = \frac{1}{m}\sum_{j=0}^{m}(\| (T_{ct} - \Delta * T_c) \|). \tag{8.6}$$

We define the $\boldsymbol{T_c}$ as the relative position change between the two successive frames F_1 and F_2, and $\boldsymbol{T_{ct}}$ as the ground truth relative position. As the camera goes 6-DOF in the coordinate system, the change of RPE may not strictly follow the same varying pattern with APE; there is a possibility that the overall average RPE will be relatively low, but the accumulation of small errors and drifts on one dimension will lead to relatively large overall APE.

B. Reconstruction Error

We define errors of VSLAM depth map using average error across entire map:

$$E_{re} = \frac{1}{m}\sum_{j=0}^{m}(\frac{1}{n_j}\sum_{i=0}^{n_j}(d_{error,i,j})), \tag{8.7}$$

where m is the total number of keyframes generated by the SLAM system, n_j is the total number of 3D features observed in keyframe j. The $\boldsymbol{E_{re}}$ can thus be defined as Reconstruction Error (RE). Utilizing same rigid body transformation $\boldsymbol{\Delta}$ the error term $\boldsymbol{d_{error,i,j}}$ is defined as:

$$d_{error} = \frac{\| (P_t - \Delta * P) \|}{\| P_t \|}, \tag{8.8}$$

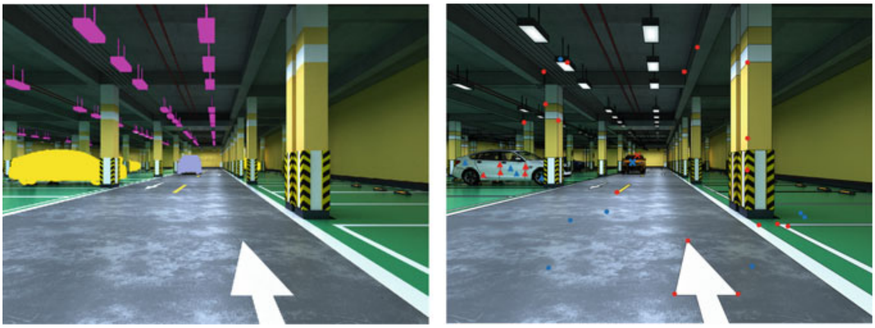
where $\boldsymbol{P}$ is the world position of one 3D feature estimated by Visual SLAM system, $\boldsymbol{P_t}$ is the ground truth world position. To measure the amount of variation in depth map, we calculate the average standard deviation $\boldsymbol{S_{re\_std}}$ across entire map utilizing error term defined in Eq. 8.8 using the following equation:

$$S_{re\_std} = \frac{1}{m}\sum_{j=0}^{m}(d_{error,j}). \tag{8.9}$$

C. Cruciality of Features

Most VSLAM systems can only use static features to build up local 3D map, since features sampled from moving objects or semi-static objects will affect the tracking accuracy. To best describe how feature distribution affects the operation of VSLAM systems, we further divide the cruciality of camera's surroundings into five grades:

1. **Static structures** are the fixed signs and structures, faculties or establishments that will not change in reasonably long time.
2. **Appearance variable structures** are structures or establishments that stay still in general, but the shape or features will change due to exterior forces.
3. **Semi-static objects** will stay reasonably still for some time, but will disappear or change their positions after they disappear from camera's field of view.



(a) Labeled different level of feature cruciality

(b) Illustration of different features wit different significance and different cruciality

Fig. 8.11 **a** Unmarked area: static structure. Purple: appearance variable structure. Lights will change its illuminance unexpectedly, the feature distribution will change, but the general position of the lights remains the same all the time. Yellow: semi-static objects. These objects will stay reasonably still for some time, but its future moving pattern is unpredictable. Gray: moving objects. We use a moving vehicle as example for moving objects here. **b** Red markers are features with high significance; green marks are features with low significance. Marks "×" are features lying on moving objects; marks "•" are features lying on static structures and variable structures; marks "▲" are features lying on variable still objects

4. **Moving objects** are objects moving right now. The difference between moving objects and semi-static objects is that the moving objects move within camera's field of view, while the semi-static objects will not move when seen by the camera.
5. **Irrelevant objects** are objects which are faraway from the camera, thus will stay still on the image even when the camera is moving, like clouds in the sky or faraway large buildings.

We define the cruciality of features in VSLAM map using the average weighted grade cross the entire map:

$$S_{ce} = \frac{1}{m}\sum_{i=0}^{m}(3 * P_{1i} + 2 * P_{2i} + 1 * P_{3i} + 0 * (P_{4i} + P_{5i})), \qquad (8.10)$$

where m is the total number of keyframes generated by the SLAM system, P_1, P_2, P_3, P_4 and P_5 represent percentage of features falling within static structure area, variable structure area, variable still object area, movable object area and irrelevant object area in this keyframe respectively. We believe features on "static structure" are the best since both their 3D positions and image features will stay unchanged. A vivid illustration for features with different level of cruciality can be found in Fig. 8.11.

D. Significance of Features

For VSLAM methods that use any kinds of image features, simply using every sampled features is obviously infeasible. Many VSLAM systems use different methods to choose suitable feature for camera tracking and local mapping.

Table 8.3 Online camera pose error results. The error is calculated by comparing between ground truth and camera 6-DOF poses retrieved directly from real-time SLAM operation. We use "ORB" to represent ORB-SLAM2

Dataset	Error type	DSO	ORB	SVO2.0	DeepTAM
TCG_1	APE (mm)	**428.5**	586.3	2030.0	1021.5
	RPE (mm)	**3.751**	9.429	17.55	4.356
TCG_2	APE (mm)	1216.0	**117.4**	4560.4	3035.6
	RPE (mm)	**3.657**	4.601	44.06	7.056
TCG_3	APE (mm)	1578.0	**656.8**	2650.0	2970.0
	RPE (mm)	**7.174**	11.19	32.20	9.511
TCG_4	APE (mm)	5021.0	**381.5**	3630.2	2137.0
	RPE (mm)	25.40	**4.995**	34.53	15.66
TCG_5	APE (mm)	**175.1**	354.7	6946.1	4048.0
	RPE (mm)	**1.363**	9.793	25.57	10.46
TCG_6	APE (mm)	**113.8**	934.9	13389	7601.0
	RPE (mm)	**1.159**	3.740	60.28	16.11
TCG_7	APE (mm)	**5286.0**	6416.0	14549	196478
	RPE (mm)	**10.48**	26.46	64.11	41.41

To make the most of the power of the local map in describing surroundings, features that best describe the general shape of the surroundings are obviously the best. Thus, we define the "significance" of features in VSLAM map to describe how many features lie on borders of different objects. The "significance" can be defined as:

$$S_{se} = \frac{1}{m}\sum_{i=0}^{m}\frac{B_i}{N_i}, \tag{8.11}$$

where m is the total number of keyframes generated by the SLAM system, N_i is the total number of observed features in current keyframe, B_i is number of features lying on "borders". The "borders" are defined as organized high-gradient area on the image (i.e.edges formulated by changes in texture, color, surface normal, etc). We define features whose image distance to nearest borders is lower than 3px as pixels on "borders". A vivid description of how feature distribution affects their significance is illustrated in Fig. 8.11; note that all the "borders" are drawn in black lines in Fig. 8.11.

E. Outlier Removal Capability

Features lying on moving objects are treated as outliers for VSLAM systems and must be removed. Most VSLAM methods use methods utilizing feature reprojection error to classify outliers; work [55] uses a combination of geometry methods and CNN classification methods to remove outliers on moving objects.

As the capability to remove outlier is crucial, we use labeled area of moving objects (i.e.vehicles, pedestrians, etc) in our dataset to test how well the outliers can be removed. We define the following equation to rate this capability:

$$S_{orc} = \frac{1}{m} \sum_{i=0}^{m} (1 - \frac{O_i}{N_i}), \tag{8.12}$$

where m is the total number of keyframes generated by the SLAM system, N_i is the total number of observed features on moving objects in one keyframe, O_i is the number of features remaining in final map. The definition indicates that the higher the S_{orc} is, the better is the VSLAM's ability to remove outliers.

8.5.1.2 Evaluation Results

We test four selected state-of-the-art VSLAM methods in this category following the scheme: the ORB_SLAM2 [3], the DSO [1], the DeepTAM [56] and the SVO [2]. Note that DeepTAM needs depth image as input, therefore we use the simulated Kinect noise as shown in Fig. 8.2.

A. Localization
We test accuracy of localization using 7 rendered sequences, denoted by TCG_1 to TCG_7 respectively. The results for camera tracking are shown in Table 8.3; the comparison of camera absolute positions estimated by VSLAM systems on sequence TCG_1 is shown in Fig. 8.12; we also compare the APE and ratio of APE to camera's travelling distance in Fig. 8.13.

As demonstrated in the results, the DSO carries lowest error on sequence TCG_1, TCG_5, TCG_6 and TCG_7, but its unstable initialization methods will also lead to relatively large overall errors on TCG_4. Note that the above results are retrieved under certain preconditions provided by TCG dataset; thus the results are complementary for test results under other preconditions or with datasets recorded with different technologies.

B. Mapping
We test the feature distribution and reconstruction error of ORB_SLAM2, DSO and SVO using error terms. The results are shown in Table 8.4. Note that as the deep learning based methods often do not have entities that are considered as "features" in those traditional methods, we do not test any deep learning based methods in this part.

The three systems can sample most of their features on static structures; however, the DSO can sample more features on significant edges and borders, since the system uses image gradient as the main standard for sampling features. The ORB_SLAM2 and SVO sample features evenly on the scene, thus a large portion of the features lies on roads but not significant borders, objects or signs. All the three methods have relatively large depth reconstruction errors, but DSO has lowest depth map derivation, suggesting that the depth map generated by DSO has less "peak error

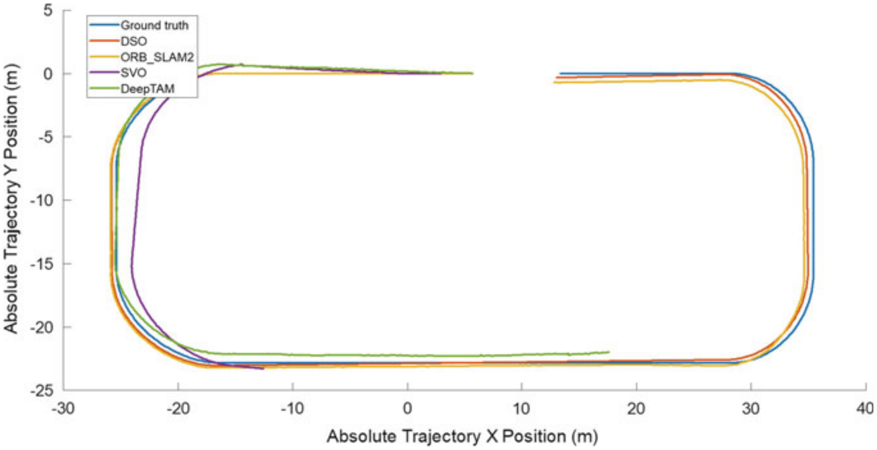


Fig. 8.12 Demonstration of camera absolute position estimated by VSLAM using TCG_1 dataset. The SVO and DeepTAM fail to finish exploring the entire sequence, therefore this figure only demonstrate part of the track calculated by the two systems

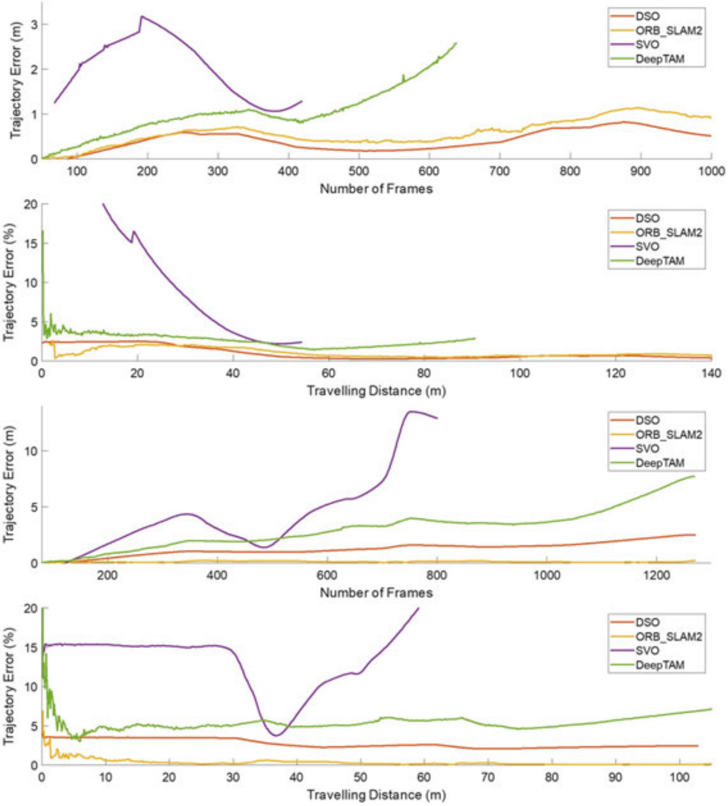
Table 8.4 Online local depth map error results. The error is calculated by comparing between ground truth and keyframe 3D features retrieved directly from real-time SLAM operation

Dataset	TCG_1				TCG_5				
Error	RE (%)	RE_STD	CF	SF (%)	RE (%)	RE_STD	CF	SF (%)	ORC
DSO	22.01	0.3869	2.835	74.15	20.86	0.3828	2.565	76.31	0.7358
ORB	20.27	0.5608	2.724	69.72	19.77	0.5506	2.427	70.03	0.9354
SVO2.0	16.78	0.5305	2.756	71.30	19.68	0.5597	2.387	71.03	0.7227

value”. This is probably achieved by the joint optimization mechanism in DSO that uses a sliding window to optimize depth and camera pose jointly. The ORB_SLAM2 has best capability of removing outliers, thanks to its advanced local map bundle adjustment methods.

8.5.2 Experiments on Synthetic and Real-World Datasets

We test the proposed VSLAM system on a workstation equipped with overclocked Core I7 CPU and 32GB memory. Section 8.5.2.1 presents tests and experiments on synthetic datasets; Sect. 8.5.2.2 presents tests on real-world datasets. Synthetic datasets can provide high-quality photorealistic images and accurate ground truth for end-to-end VSLAM evaluation, and we use these datasets to analyse in detail how the proposed VSLAM system performs; we cannot construct real-world dataset with accurate ground truth, therefore we only use real-world datasets for validating



Test results of APE to frame number of the sequence on TCG_1

Test results of ratio of APE to camera's travelling distance on TCG_1

Test results of APE to frame number of the sequence on TCG_2

Test results of ratio of APE to camera's travelling distance on TCG_2

Fig. 8.13 Demonstration of comparison of localization errors on TCG_1 and TCG_2. We demonstrate figure of APE and ratio of APE to camera's travelling distance. Note that SVO cannot finish exploring the two sequences and DeepTAM cannot finish exploring TCG_1, therefore we only demonstrate part of the results

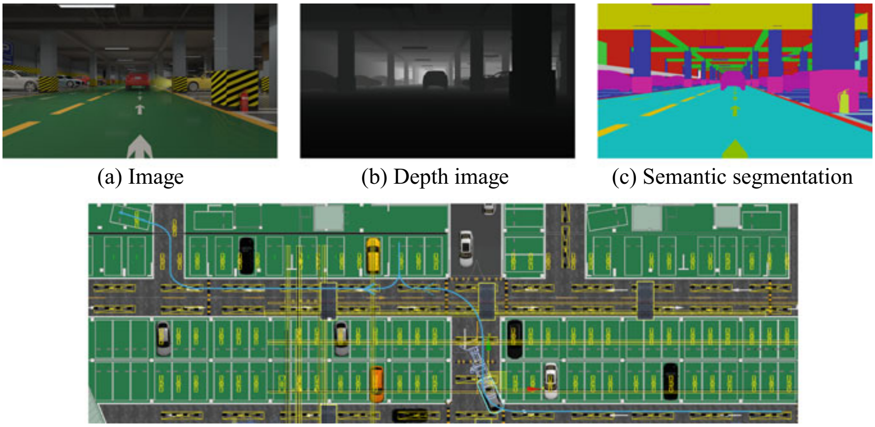
the functionalities of the proposed VSLAM system, especially validate how the joint point-object covisibility map performs on these real-world sequences.

8.5.2.1 Evaluation Results on Synthetic Datasets

In this section we demonstrate evaluation results on synthetic datasets.

We set the vehicle to explore in a virtual large-scale building model with camera mounted on the vehicle's front. The test set used in this work contains 7 different test sequences with different environmental preconditions and different routes. An example of our synthetic dataset can be found in Fig. 8.14.

The results of camera localization accuracy experiments in pure static environments (i.e.$SYN_1 \sim SYN_5$) are shown in Table 8.5, the results of camera localization



(a) Image (b) Depth image (c) Semantic segmentation

(d) Camera route from topview of the virtual model

Fig. 8.14 Examples of the synthetic dataset. **a**, **b**, **c** Examples of three kinds of information that can be automatically generated when rendering the synthetic dataset. **d** Topview of our virtual underground garage model in 3DS-MAX software. The blue line represents camera trajectory from one of our test sequences; the white car on the blue line is the vehicle where the camera is mounted

Table 8.5 Localization accuracy comparison in purely static environments. Note that we cannot guarantee that the three systems can be initialized at the same frame, thus we select common frames that all the three systems can stably work. To reduce the influence of the inevitable random factors, for instance, the DSO's initialization process may fail when testing SYN_2, we use the average result for each sequence from more than five successful runs

Dataset	Error Type	DSO	ORB [3]	Proposed
SYN_1	APE (mm)	1152.51	1231.25	**863.37**
	RED (%)	8.03	**3.77**	3.84
SYN_2	APE (mm)	1363.96	1325.54	**1211.04**
	RED (%)	8.34	6.59	**5.51**
SYN_3	APE (mm)	**255.02**	695.13	436.05
	RED (%)	2.63	3.03	**1.38**
SYN_4	APE (mm)	**426.86**	440.92	494.98
	RED (%)	3.12	2.54	**2.49**
SYN_5	APE (mm)	405.88	2000.98	**199.18**
	RED (%)	1.46	9.35	**1.46**

Table 8.6 Localization accuracy comparison on sequences with moving objects. Note that similar to Table 8.5, we select common frames that all the three systems can stably work, and use the average result for each sequence from more than five successful runs

Dataset	Error type	DSO	ORB [3]	Proposed
SYN_6	APE (mm)	3913.82	1263.96	**934.9**
	RED (%)	11.34	6.96	**5.51**
SYN_7	APE (mm)	11344.8	**97.37**	98.04
	RED (%)	35.64	2.41	**2.19**

Table 8.7 Results comparing connection rate R_{CON} of the proposed work ("This") with work [3] ("ORB"). Similar to results from Table 8.5, we use the average results from several runs, and use common parts from the sequence in which both ORB and our proposed system stably work

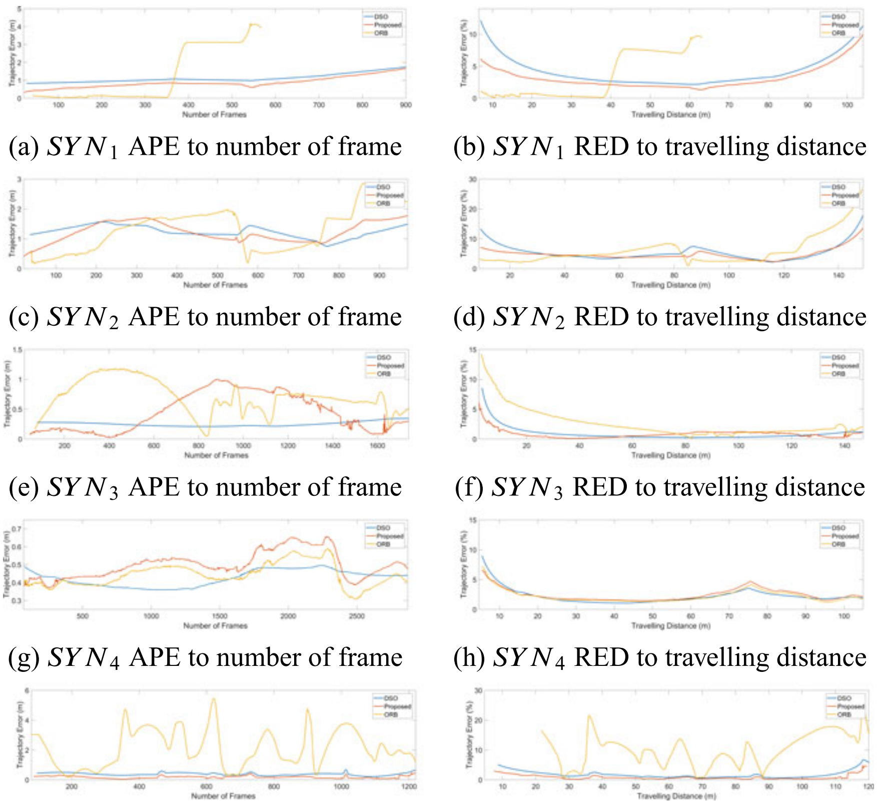
Dataset	SYN_1 (%)	SYN_2 (%)	SYN_3 (%)	SYN_4 (%)	SYN_5 (%)
This	49.81	85.46	83.26	55.40	51.84
ORB	37.12	52.61	74.71	51.18	38.50

accuracy experiments in environments with moving objects (i.e.$SYN_6 \sim SYN_7$) are shown in Table 8.6, and the comparison of APE and RED relative to frame number and travelling distance are shown in Fig. 8.15.

The proposed system has better absolute tracking accuracy in 3 out of 5 cases compared to DSO and ORB_SLAM2, but as can be observed from from Fig. 8.15c and e, the proposed system and ORB_SLAM2 clearly show larger variation in APE. The DSO owns a unique architecture in which a large sliding window is used to optimize all active keyframes; the positioning error of 3D features that are most concerned with the tracking process is averaged and accumulated, given DSO's better error consistence in every one of the first five sequences. The average magnitude of RED could characterize the accumulation of scale drift, and it's clear that the proposed system has better scale drift resilience than the other, thanks to the submap's constraints which are used to actively rectify scale drifts.

The localization accuracy for DSO in sequence SYN_6 and SYN_7, as shown in Table 8.6, is much worse than the other two systems. The two sequences are specially designed to test the outlier removal functionality of the VSLAM system; in the two sequences, there will always be one large moving vehicle, which may occupy about 30% of the actual image. For DSO, we have noticed that a large portion of the outliers are still preserved in the system, which becomes false 3D mappoints in the map, and in return influences the accuracy of camera localization. The proposed system and ORB_SLAM2 share the same outlier removal mechanism and can withstand detrimental moving objects on image; besides, if there are not enough static feature points available on the frame, the proposed system can still localize itself using at least one mapobject on the scene, which extends its functionality in scenes with moving objects.

The proposed system features a novel map construction mechanism, which can associate two keyframes which observe the same mapobjects even when the two



(a) SYN_1 APE to number of frame (b) SYN_1 RED to travelling distance

(c) SYN_2 APE to number of frame (d) SYN_2 RED to travelling distance

(e) SYN_3 APE to number of frame (f) SYN_3 RED to travelling distance

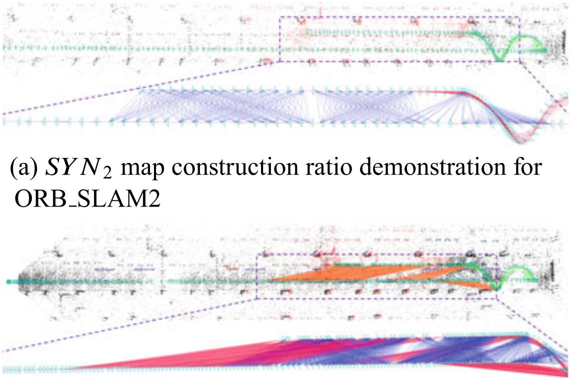
(g) SYN_4 APE to number of frame (h) SYN_4 RED to travelling distance

(i) SYN_5 APE to number of frame (j) SYN_5 RED to travelling distance

Fig. 8.15 Error charts comparing three VSLAM systems using $SYN_1 \sim SYN_5$. We use ORB to represent work [3]. Note that ORB_SLAM2 cannot finish exploring SYN_1, therefore we only demonstrate parts on which ORB_SLAM2 can stably work

frames do not share the same image features. The results comparing the proposed method with ORB_SLAM2 on connection rate are shown in Table 8.7. We also demonstrate the factor graph of the proposed system and ORB_SLAM2 running on sequence SYN_2 in Fig. 8.16. Parts of the factor graph from Fig. 8.16 are slightly enlarged to demonstrate how well the factor graph built by the two systems "cover" the theoretical associations.

The proposed method clearly performs better in establishing covisible associations when compared with ORB_SLAM2: At the start of the sequence, frames share the same image appearance will also share the same object observation, thus the covisibility map built by point features and the covisibility map built by object features largely coincide, making the superiority of the proposed system insignificant. In the middle of the sequence, the ORB_SLAM2 could only establish associations between keyframe *KF* and its "vertical frames", the horizontal frames will not be included, as



(a) SYN_2 map construction ratio demonstration for ORB_SLAM2

(b) SYN_2 map construction ratio demonstration for the proposed system

Fig. 8.16 Map construction results comparison on SYN_2 between ORB_SLAM2 and the proposed system. **a**, **b** The upper figure is the actual map construction results in VSLAM system. The orange lines represent associations established by matching object instances in factor graph, and the green lines are associations introduced by matching image feature. We slightly enlarged the part of map in purple box to provide better view of how the connections are distributed, and the enlarged results are shown in the lower figure. In the lower figure, the lines are theoretical associations (the two frames view common image parts) calculated when rendering the synthetic dataset. Fuchsia lines are associations that have been "activated" by VSLAM system, while blue lines are associations that have not been "activated". Note that we cannot guarantee that the two systems will select same keyframes, therefore the theoretical connections are calculated individually for the two systems, which means the theoretical connections of the two VSLAM systems will not be the same

there exists a large variation between actual image features of *KF* and its horizontal frames (the "vertical" and "horizontal" is relative to the camera's moving direction). For the proposed system, the enhanced mapping module can build associations both between horizontal frames and vertical frames, which makes the connection ratio of this system obviously better than ORB_SLAM2. Besides, the APE and RED of the sequence are lower when comparing with DSO and ORB_SLAM2 (see Table 8.5), owing to the scale aware bundle adjustments based on the extra associations. Note that Fig. 8.8 also provides an example regarding the capability of our system to establish covisibility map, Fig. 8.10 provides a demonstration of the object matching process.

8.5.2.2 Evaluation Results on Real-World Datasets

We test the proposed VSLAM system on two real-world sequences: the sequence used in Fig. 8.17 is recorded in a large underground garage; the sequence used in Fig. 8.18 is recorded on real-world road. Both the Figs. 8.17 and 8.18 illustrate the joint covisibility graph constructed by 3D point features and rasterized object models features, which implies the feature extraction, tracking and mapping modules can work effectively in real-world environments. The point-object joint covisibility map can establish connections between frames without enough common point features, and is the key functionality that helps the proposed method reduce accumulated error. Part (D) of Fig. 8.17 is the actual trajectory of the camera, and the red circle is used to show that the camera moves to the same place when going back; part (B) and part (C) of Fig. 8.17 are camera trajectories recovered by the proposed method and ORB_SLAM2 respectively. The result of ORB_SLAM2 obviously contains larger accumulated error than that of the proposed method. The orange lines in part (B) of Fig. 8.17 are object connections between keyframes observing the same object; while the result of ORB_SLAM2 does not contain such connections between these keyframes, as shown in part (C) of Fig. 8.17. Figure 8.18 is another experiment of the proposed method running on actual outdoor environments; Fig. 8.18 illustrates that both the traffic sign objects and the point features can be effectively extracted under various environmental preconditions and be utilized to construct the joint covisibility map in the proposed VSLAM system.

8.6 Conclusion

In this chapter, we present a fully synthetic dataset "TCG" and the evaluation framework that uses rendered photorealistic images from "TCG" to evaluate VSLAM system. We use models in 3DS-Max and several supplementary software tools to setup and render a large database of sequences. By fully taking advantage of the render elements in V-Ray software, we can generate large amount of information regarding the rating and debugging of a VSLAM system. Our dataset can provide many kinds of information when compared with other state-of-the-art VSLAM datasets, as shown in Table 8.1. We also propose several new criteria for rating performance of VSLAM based on environmental information, which extends VSLAM evaluation framework from only testing camera positions to evaluating camera tracking and mapping jointly. Based on these high-accuracy zero-error ground truth, we test four popular VSLAM systems on our dataset to demonstrate the functionality of the framework and the dataset.

Although techniques and methods proposed in this chapter are useful, we still believe that both the proposed dataset and evaluation framework need improvement. There are mainly two future topics for extending and perfecting the proposed work:

1. We will add more sequences in different environments. In this work we focus on sequences in underground garages; however, the framework proposed for VSLAM evaluation are not limited to sequences in underground garages. We will update two kinds of sequences in the future:
    a. Sequences in outdoor environments that include climate change, luminance variation and simulated traffic actions;
    b. Sequences that are not based on planar vehicle motion.

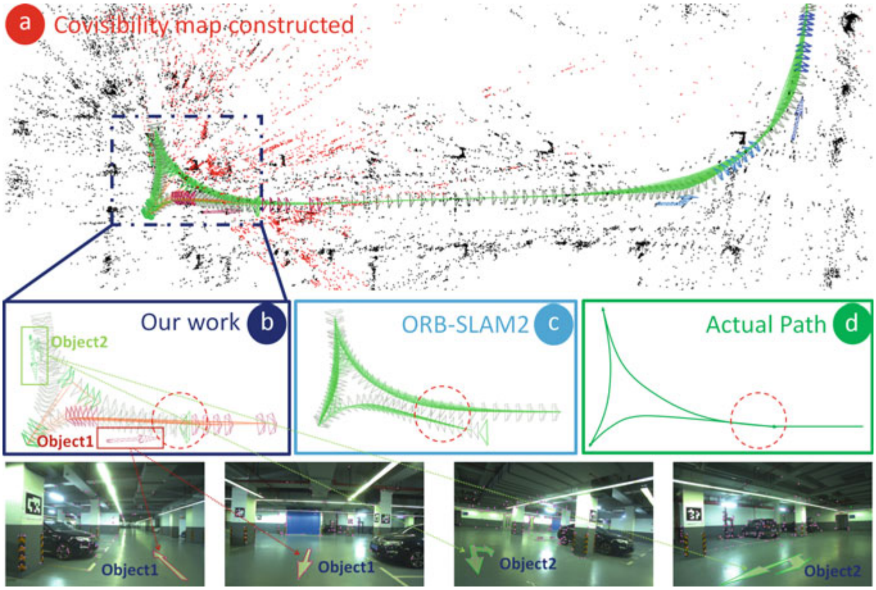


Fig. 8.17 Map reconstruction results of the proposed method in real-world underground garage sequences. We demonstrate the joint covisibility map constructed by the proposed method in **a**. We also slightly enlarge one part of the covisibility map in **b**, where orange lines are object connections constructed by the proposed method; blue lines are the actual path of the camera's motion. Frames that observe the same mapobject are marked with same color of that mapobject: red keyframes observe object1 and green keyframes observe object2. We select two keyframes that observe object1 and two keyframes that observe object2 as an example of how objects are distributed on the image. **c** is the map reconstruction results of the original version of ORB_SLAM2. We also present the actual path of the camera's motion in **d**. The red circle marks the terminating location of the camera when it moves either forward or backward. Note that all the paths in this figure are not precise and can only be treated as a brief impression of how the camera moves

2. Using information provided by the dataset, we will continue to study how different environmental factors influence the performance of VSLAM systems. We will not only focus on the trivial factors such as texture of objects or global luminance, but also on structures and layouts that influence the distribution and organization of 3D features.

The method of using computer graphic methods to generate VSLAM data has encountered huge development in recent years, thanks to the development of realistic rendering technology. Although CG-based methods have the natural superiority that it can generate zero-error ground truth and labels, the fact that it needs human intervention means we cannot simulate every possible contingency in real world. Thus, the key factor on generating realistic data for VSLAM evaluation is simulating object behavior precisely. To solve this problem, we believe there are mainly two ways:

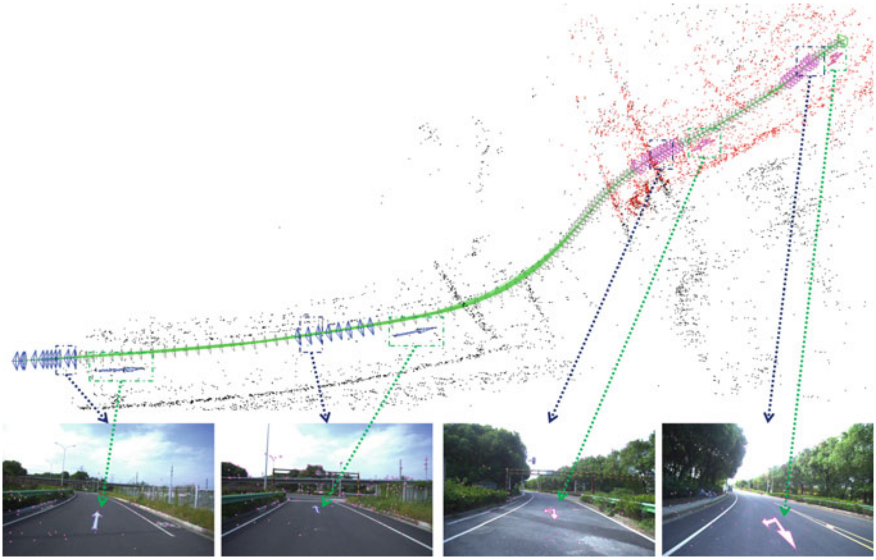


Fig. 8.18 Demonstration of results of the proposed method running on sequences retrieved from real-world roads. Points and objects are expressed using the same manner as in Fig. 8.17. Four keyframes are selected from corresponding parts marked on the map for demonstration

1. Establish a large database that contains large number of possible behaviors of different objects (e.g.pedestrians, vehicles, animals, etc) on the scene.
2. Developing methods that can extract object behaviors from videos and pictures on the internet.

Using the above two approaches, researchers can generate a wide variety of different datasets for different computer vision research areas. In recent years we have seen works like [57] that can generate labeled synthetic datasets automatically, but it cannot always resolve occlusions or intersections correctly, and thus still needs to be improved.

8.7 Discussion

This chapter proposed a method on how to apply texture-less objects with known CAD model (e.g.traffic signs) in VSLAM architecture. Our method mainly consists of two parts: the particular VSLAM architecture using objects as landmarks and the parameterization process that characterize mapobjects suitable for VSLAM operation. The object-oriented VSLAM extends the classical architecture of ORB_SLAM2 to include object features based camera tracking and local mapping module. The object based local mapping module allows us to enhance the map build functionality and retrieve covisibility associations based on common structural information. We further deploy this enhanced covisibility graph to enable several unique bundle adjustment methods which will help reduce accumulated tracking error. The work also presents a complete pipeline on: (A) generating rasterized model based on CAD model; (B) keyobject pose estimation by matching object model to image distance field; (C) keyobject matching mechanism that can compute object matching without the usage of any CNN outputs. In the experiments that follow, we evaluate the localization performance and mapping performance of the proposed system; we prove that the proposed VSLAM architecture can provide comparable localization performance with state-of-the-art VSLAM algorithms with lower accumulated scale drift; we have also proved that by extending the architecture of ORB_SLAM2, we can build up better factor graph in helping operation of other modules in VSLAM.

While we have proved that the proposed system can achieve better overall performance than state-of-the-art VSLAM methods on our synthetic dataset, there are still several areas that can be improved to perfect the proposed work.

1. Currently both the object detection method and object initial pose estimation method utilize classification tree (or regression tree) method, whose robustness and accuracy are not as good as state-of-the-art deep learning based methods. Future implementation of this method will fuse better detection and initial pose estimation methods.
2. The proposed method focuses on using the keyobjects as landmarks for precise camera localization and pay little attention to enhance the accuracy of 3D positions of landmarks. While the accuracy of mapobject poses can be guaranteed by the object pose estimation process, the accuracy of mappoints is similar to ORB_SLAM2. The objects on the scene have provided geometric constraints to these mappoints, utilizing these constraints to enhance the accuracy of 3D positions of sparse feature points will be one of the topics of our future work.
3. While most researchers focus on creating vision-based self-localization methods that are better at exploring an unknown scene, the actual industrial applications have proposed demands on using existed map in localization. These pre-built maps do not only contain 3D information like point cloud but also process rich semantic information like keyobjects (signs, markers etc, with labels, poses and 3D information) and structural information (annotated 3D lines and planes, lane lines, structural lines). To fully take advantage of these annotated information, our system needs to be extended in a future work.

References

1. Engel J, Koltun V, Cremers D (2018) Direct sparse odometry. IEEE Trans Pattern Anal Mach Intell 40(3):611–625
2. Forster C, Zhang Z, Gassner M, Werlberger M, Scaramuzza D (2017) Svo: semidirect visual odometry for monocular and multicamera systems. IEEE Trans Rob 33(2):249–265

3. Mur-Artal R, Tardós JD (2017) Orb-slam2: an open-source slam system for monocular, stereo, and rgb-d cameras. IEEE Trans Rob 33(5):1255–1262
4. Engel J, Schöps T, Cremers D (2014) Lsd-slam: large-scale direct monocular slam. In: European conference on computer vision. Springer, pp 834–849
5. Jose Tarrio J, Pedre S (2015) Realtime edge-based visual odometry for a monocular camera. In: Proceedings of the IEEE international conference on computer vision. IEEE, pp 702–710
6. Yang S, Scherer S (2017) Direct monocular odometry using points and lines. In: 2017 IEEE international conference on robotics and automation (ICRA). IEEE, pp 3871–3877
7. Wang X, Dong W, Zhou M, Li R, Zha H (2016) Edge enhanced direct visual odometry. In: BMVC
8. Gomez-Ojeda R, Briales J, Gonzalez-Jimenez J (2016) Pl-svo: Semi-direct monocular visual odometry by combining points and line segments. In: 2016 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 4211–4216
9. Maity S, Saha A, Bhowmick B (2017) Edge slam: edge points based monocular visual slam. In: Proceedings of the IEEE international conference on computer vision, pp 2408–2417
10. Gomez-Ojeda R, Zuñiga-Noël D, Moreno F-A, Scaramuzza D, Gonzalez-Jimenez J (2017) Pl-slam: a stereo slam system through the combination of points and line segments. arXiv preprint arXiv:1705.09479
11. Zhou H, Zou D, Pei L, Ying R, Liu P, Yu W (2015) Structslam: visual slam with building structure lines. IEEE Trans Veh Technol 64(4):1364–1375
12. Li H, Yao J, Bazin J-C, Lu X, Xing Y, Liu K (2018) A monocular slam system leveraging structural regularity in manhattan world. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, pp 2518–2525
13. Ma L, Kerl C, Stückler J, Cremers D (2016) Cpa-slam: consistent plane-model alignment for direct rgb-d slam. In: 2016 IEEE international conference on robotics and automation (ICRA). IEEE, pp 1285–1291
14. Hsiao M, Westman E, Zhang G, Kaess M (2017) Keyframe-based dense planar slam. In: 2017 IEEE international conference on robotics and automation (ICRA). IEEE, pp 5110–5117
15. Salas-Moreno RF, Glocken B, Kelly PH, Davison AJ (2014) Dense planar slam. In: IEEE international symposium on mixed and augmented reality (ISMAR). IEEE 2014:157–164
16. Hsiao M, Westman E, Kaess M (2018) Dense planar-inertial slam with structural constraints. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, pp 6521–6528
17. Liwicki S, Zach C, Miksik O, Torr PH (2016) Coarse-to-fine planar regularization for dense monocular depth estimation. In: European conference on computer vision. Springer, pp 458–474
18. Nicholson L, Milford M, Sünderhauf N (2019) Quadricslam: dual quadrics from object detections as landmarks in object-oriented slam. IEEE Robot Autom Lett 4(1):1–8
19. Jablonsky N, Milford M, Sünderhauf N (2018) An orientation factor for object-oriented slam. arXiv preprint arXiv:1809.06977
20. McCormac J, Clark R, Bloesch M, Davison A, Leutenegger S (2018) Fusion++: volumetric object-level slam. In: 2018 international conference on 3D vision (3DV). IEEE, pp 32–41
21. Salas-Moreno RF, Newcombe RA, Strasdat H, Kelly PH, Davison AJ (2013) Slam++: simultaneous localisation and mapping at the level of objects. Proceedings of the IEEE conference on computer vision and pattern recognition 2013:1352–1359
22. Fei X, Soatto S (2018) Visual-inertial object detection and mapping. In: Proceedings of the European conference on computer vision (ECCV), pp 301–317
23. Hosseinzadeh M, Li K, Latif Y, Reid I (2018) Real-time monocular object-model aware sparse slam. arXiv preprint arXiv:1809.09149
24. Hosseinzadeh M, Latif Y, Pham T, Suenderhauf N, Reid I (2018) Towards semantic slam: points, planes and objects. arXiv preprint arXiv:1804.09111
25. Yang S, Scherer S (2018) Cubeslam: monocular 3d object detection and slam without prior models. arXiv preprint arXiv:1806.00557
26. Li R, Wang S, Long Z, Gu D (2018) Undeepvo: monocular visual odometry through unsupervised deep learning. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, pp 7286–7291

27. Pumarola A, Vakhitov A, Agudo A, Sanfeliu A, Moreno-Noguer F (2017) Pl-slam: real-time monocular visual slam with points and lines. In: 2017 IEEE international conference on robotics and automation (ICRA). IEEE, pp 4503–4508
28. Qin T, Li P, Shen S (2018) Vins-mono: a robust and versatile monocular visual-inertial state estimator. IEEE Trans Rob 34(4):1004–1020
29. Zhang J, Singh S (2018) Laser-visual-inertial odometry and mapping with high robustness and low drift. Journal of Field Robotics 35(8):1242–1264
30. Zhang J, Kaess M, Singh S (2017) A real-time method for depth enhanced visual odometry. Auton Robot 41(1):31–43
31. Li S-P, Zhang T, Gao X, Wang D, Xian Y (2019) Semi-direct monocular visual and visual-inertial slam with loop closure detection. Robot Auton Syst 112:201–210
32. López A, Villalonga G, Sellart L, Ros G, Vázquez D, Xu J, Marín J, Mozafari A (2017) Training my car to see using virtual worlds. Image Vis Comput 68:08
33. Schubert D, Goll T, Demmel N, Usenko V, Stückler J, Cremers D (2018) The tum vi benchmark for evaluating visual-inertial odometry. arXiv preprint arXiv:1804.06120
34. Carlevaris-Bianco N, Ushani AK, Eustice RM (2016) University of michigan north campus long-term vision and lidar dataset. Int J Robot Res 35(9):1023–1035
35. Miller M, Chung S-J, Hutchinson S (2018) The visual-inertial canoe dataset. Int J Robot Res 37(1):13–20
36. Engel J, Usenko V, Cremers D (2016) A photometrically calibrated benchmark for monocular visual odometry. arXiv preprint arXiv:1607.02555
37. Chen C, Zhao P, Lu CX, Wang W, Markham A, Trigoni N (2018) Oxiod: The dataset for deep inertial odometry. arXiv preprint arXiv:1809.07491
38. Pfrommer B, Sanket N, Daniilidis K, Cleveland J (2017) Penncosyvio: A challenging visual inertial odometry benchmark. In: 2017 IEEE international conference on robotics and automation (ICRA). IEEE, pp 3847–3854
39. Blanco-Claraco J-L, Moreno-Dueñas F, González-Jiménez J (2014) The málaga urban dataset: High-rate stereo and lidar in a realistic urban scenario. Int J Robot Res 33(2):207–214
40. Maddern W, Pascoe G, Linegar C, Newman P (2017) 1 year, 1000 km: the oxford robotcar dataset. Int J Robot Res 36(1):3–15
41. Cortés S, Solin A, Rahtu E, Kannala J (2018) Advio: an authentic dataset for visual-inertial odometry. In: Proceedings of the European conference on computer vision (ECCV), pp 419–434
42. Geiger A, Lenz P, Stiller C, Urtasun R (2013) Vision meets robotics: the kitti dataset. Int J Robot Res 32(11):1231–1237
43. Burri M, Nikolic J, Gohl P, Schneider T, Rehder J, Omari S, Achtelik MW, Siegwart R (2016) The euroc micro aerial vehicle datasets. Int J Robot Res 35(10):1157–1163
44. Li W, Saeedi S, McCormac J, Clark R, Tzoumanikas D, Ye Q, Huang Y, Tang R, Leutenegger S (2018) Interiornet: mega-scale multi-sensor photo-realistic indoor scenes dataset. In: British machine vision conference (BMVC)
45. Ros G, Sellart L, Materzynska J, Vazquez D, Lopez AM (2016) The synthia dataset: a large collection of synthetic images for semantic segmentation of urban scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3234–3243
46. Li X, Wang K, Tian Y, Yan L, Deng F, Wang F-Y (2018) The paralleleye dataset: a large collection of virtual images for traffic vision research. IEEE Trans Intell Transp Syst 99:1–13
47. Shah S, Dey D, Lovett C, Kapoor A (2018) Airsim: high-fidelity visual and physical simulation for autonomous vehicles. In: Field and service robotics. Springer, pp 621–635
48. Qiu W, Zhong F, Zhang Y, Qiao S, Xiao Z, Kim TS, Wang Y (2017) Unrealcv: virtual worlds for computer vision. In: Proceedings of the 2017 ACM on multimedia conference. ACM, pp 1221–1224
49. Handa A, Whelan T, McDonald J, Davison AJ (2014) A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In: IEEE international conference on robotics and automation (ICRA). IEEE, 1524–1531

50. Maye J, Furgale P, Siegwart R (2013) Self-supervised calibration for robotic systems. In: 2013 IEEE intelligent vehicles symposium (IV). IEEE, pp 473–480
51. Murphy KP (2012) Machine learning: a probabilistic perspective. MIT Press
52. Muñoz E, Konishi Y, Murino V, Del Bue A (2016) Fast 6d pose estimation for texture-less objects from a single rgb image. In: 2016 IEEE international conference on robotics and automation (ICRA). IEEE, pp 5623–5630
53. Imperoli M, Pretto A (2015) D^2CO: fast and robust registration of 3d textureless objects using the directional chamfer distance. In: International conference on computer vision systems. Springer, pp 316–328
54. Sturm J, Engelhard N, Endres F, Burgard W, Cremers D (2012) A benchmark for the evaluation of rgb-d slam systems. In: 2012 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 573–580
55. Bescós B, Fácil JM, Civera J, Neira J (2018) Dynslam: tracking, mapping and inpainting in dynamic scenes. arXiv preprint arXiv:1806.05620
56. Zhou H, Ummenhofer B, Brox T (2018) Deeptam: deep tracking and mapping. In: European conference on computer vision (ECCV)
57. Kar A, Prakash A, Liu M-Y, Cameracci E, Yuan J, Rusiniak M, Acuna D, Torralba A, Fidler S (2019) Meta-sim: learning to generate synthetic datasets. arXiv preprint arXiv:1904.11621

Chapter 9
Multi-task Perception for Autonomous Driving

Xiaodan Liang, Xiwen Liang, and Hang Xu

Abstract Multi-task learning has become a popular paradigm to tackle multiple tasks simultaneously with less inference time and computation resources. Recently, many self-supervised pre-training methods have been proposed and they have achieved impressive performance on a range of computer vision tasks. However, their generalization ability to multi-task scenarios is yet to be explored. Besides, most multi-task algorithms are designed for specific tasks usually not within the scope of autonomous driving, which makes it difficult to compare state-of-the-art multi-task learning methods in autonomous driving. In this chapter, we divide the multi-task perception into 2D perception and 3D perception in autonomous driving. For 2D perception, we extensively investigate the transfer ability of various self-supervised methods and reproduce multiple popular multi-task methods. Then we introduce a simple and effective pretrain-adapt-finetune paradigm for multi-task learning and a novel adapter named LV-Adapter which reuses powerful knowledge from the Contrastive Language-Image Pre-training (CLIP) model pre-trained on image-text pairs. We further present an effective multi-task framework for autonomous driving, GT-Prompt, which learns general prompts and generates task-specific prompts to guide the model to capture task-invariant and task-specific information. For 3D perception, we investigate both multi-modality fusion and multi-task learning, and introduce an effective multi-level gradient calibration learning framework across tasks and modalities during optimization.

X. Liang (✉) · X. Liang
Shenzhen Campus of Sun Yat-Sen University, Shenzhen, China
e-mail: liangxd9@mail.sysu.edu.cn

X. Liang
e-mail: liangxw29@mail2.sysu.edu.cn

H. Xu
Huawei Noah's Ark Lab, Shanghai, China
e-mail: xu.hang@huawei.com

R. Fan et al. (eds.), *Autonomous Driving Perception*, Advances in Computer Vision
and Pattern Recognition, https://doi.org/10.1007/978-981-99-4287-9_9

9.1 Introduction

Multi-task learning is a challenging problem in computer vision and has become an effective and low-resource paradigm in autonomous driving [49, 50, 113]. It is imperative for an autonomous vehicle to understand and solve multiple perception tasks, e.g., detecting surrounding cars and pedestrians, predicting road affordability for driving, and locating the lanes to perform suitable driving actions. Solving multiple tasks jointly can reduce the training and inference time to a great extent, and also enforces the model to learn more generalizable representations [17, 50]. In spite of efficiency, a unified architecture should enhance the robustness of the autonomous driving system by learning the relationships between multiple heterogeneous tasks [50, 93, 105].

Some recent works have attempted to apply unified training on multiple tasks in autonomous training. [46] trains per-pixel depth prediction, semantic segmentation, and instance segmentation in a single model. [42] introduces an extra traffic light classifier to learn different traffic patterns following traffic light changes. [12] learns object detection and depth prediction together to identify dangerous traffic scenes. However, these works differ in task types, evaluation matrix, and dataset, making it hard to compare their performances. For example, most of them are developed based on dense prediction [4, 112] and natural language understanding [16, 99], rather than being tailored for more common perception tasks for autonomous driving, thus these methods may produce poor results when applied to a self-driving system. Therefore, there is an increasing demand for a thorough evaluation of existing multi-task learning methods covering common tasks in autonomous driving.

In this chapter, we divide multi-task perception into 2D perception and 3D perception. For 2D perception, we investigate effective adaption for multi-task learning, and further propose general and task-specific prompts. For 3D perception, we unify multiple modalities to perform multi-task learning.

9.1.1 2D Perception

It is crucial for multi-task learning models to obtain universal features transferred from existing state-of-the-art off-the-shelf pre-trained models. Recently, many latest self-supervised pre-training methods have achieved great potential when transferred to various types of vision tasks [102, 108, 110] under the pretrain-finetune paradigm. Despite the impressive performance, their transferability to multi-task learning scenarios is yet to be explored. We argue that the joint learning of multiple heterogeneous tasks will introduce a multitude of challenges to train a unified model and it is often not the case that multi-task learning is of universal benefit. Firstly, the popular pretrain-finetune paradigm may result in worse performance in multi-task learning due to the misalignment of objectives between pre-training and fine-tuning [29, 35] since most supervised and self-supervised pre-training methods are designed for specific objectives or tasks [9, 10, 108, 110]. Secondly, the performance of multi-task learning relies on many non-trivial factors, e.g., model architectures, data augmentations, hyperparameters, and convergence properties [90, 92]. The specialized techniques catered for a specific type of architecture or task cannot be readily applied to a universal architecture since they are prone to fail during generalization. Moreover, due to the labor-intensive process of data annotation, it is hard to collect complete annotations of different granularities for all tasks, which further complicates the situation. We focus on heterogeneous multi-task learning on partially labeled data (HMPL) under the realistic scenarios of autonomous driving.

9.1.1.1 Effective Adaptation for Multi-task Learning

We first delve into the representation learning stage of HMPL to reveal the performance degradation of different pre-training methods. We thoroughly examine a wide range of pre-training methods including supervised pre-training (pretrained on ImageNet [88]), classification-oriented methods (e.g., SimCLR [9], MoCo [34]), detection-oriented methods (e.g., DetCo [108]), segmentation-oriented methods (e.g., DenseCL [102]), and vision-language pre-training methods (e.g., CLIP [82]) on three fine-grained tasks on the large-scale driving dataset BDD100K [118], i.e., object detection, semantic segmentation, and drivable area segmentation. Surprisingly, the performance of these methods varies greatly under the same training protocol, especially on the dense prediction task, which suggests that the misalignment in the pretrain-finetune paradigm can lead to prominent performance degradation and there exists much room for improvement.

Given that ‘universal’ pre-training is still unsolved and redesigning the resource-intensive pre-training scheme comes with great computation overhead, **LV-Adapter** [61] develops a general approach that can fully harness the knowledge from the off-the-shelf pretrained models and make them amenable to multi-task scenarios via efficient adaptation. The inspiration of LV-Adapter comes from the recent progress of prompt-based learning in Natural Language Processing (NLP) [26, 67, 82, 123], where the language model pretrained on massive amounts of raw text can be adapted to new scenarios with high efficiency by introducing hand-crafted or learnable prompts. Following this philosophy, LV-Adapter first introduces a simple but yet very effective pretrain-adapt-finetune paradigm for multi-task transfer learning as a substitute for the dominating pretrain-finetune paradigm in computer vision. Concretely, during the adapt stage, learnable multi-scale adapters with a small amount of parameters are tuned with frozen random initialized task-specific heads to dynamically adapt the knowledge from the pretrained models under the multi-task objectives. Results show that the adapt stage mitigates the gap between pre-training and fine-tuning and significantly improves the overall performance across different pretrained models while being very effective and introducing no extra training overhead.

Apart from the supervised and self-supervised pre-training, Contrastive Language-Image Pre-training (CLIP) [82] manages to learn high-quality visual representation from an enormous amount of noisy image-text pairs. CLIP achieves breakthrough performance in zero-shot image recognition, which indicates that it has great potential for improving the generalization capability and robustness of multi-task learning. To this end, LV-Adapter further excavates the linguistic knowledge from CLIP and enhances the visual representations in the multi-task scenarios. To maximize the correspondence between the dense features and the class-specific concepts, LV-Adapter first learns task-specific prompts in an end-to-end manner. Then LV-Adapter models the language-to-vision adaptation function based on the cross-attention mechanism [97] to incorporate language priors in visual features.

The pretrain-adapt-finetune paradigm significantly improves the overall performance of different off-the-shelf pre-training methods. Furthermore, LV-Adapter can serve as an effective complementary approach for multi-task learning, which brings consistent performance gains on three heterogeneous tasks simultaneously.

9.1.1.2 General and Task-Specific Prompts for Multi-task Learning

We provide a systematic study of present Multi-Task Learning (MTL) methods on large-scale driving dataset BDD100K [118]. Specifically, we find that task scheduling [62] is better than zeroing loss [106], but worse than pseudo labeling [29] on most tasks. Interestingly, in task-balancing methods, uncertainty [46] produces satisfactory results on most tasks, while MGDA [89] only performs well on lane detection. This indicates that negative transfer [17], which is a phenomenon whereby increasing the performance of a model on one task will hurt the performance on another task with different needs, is common among these approaches.

To mitigate the negative transfer problem, we introduce the general visual prompts and task-specific prompts (henceforth referred to as **GT-Prompt**) based on the following motivations: (1) Since the model has to solve multiple tasks simultaneously, we construct lightweight prompt blocks to learn more general and transferable representation. Blending the learned task-invariant knowledge can enhance the effectiveness of the multi-task model; (2) Given the visual clues of each task, the model can extract task-related information from the pre-trained model. Inspired by prompting in natural language processing [55, 71, 121], we leverage exemplars to generate task-specific prompts by considering that the visual clues should represent the specific task to some extent, and give hints for learning task-specific information; (3) Since different tasks may require different receptive fields during training, we introduce multi-scale window attention [85] for the task-specific fusion module which integrates the task-specific prompt and visual representation, further enhancing the multi-task model on most tasks. Extensive experiments show that models equipped with GT-Prompt improve their counterparts and surpass single-task baselines by a large margin (e.g., +3.6% mAP, +2.8% mIoU, +2.1% mIoU, +0.7% IoU in object detection, semantic segmentation, drivable area segmentation, and lane detection, respectively, under the disjoint-balance data split setting.).

9.1.2 3D Perception

3D perception task plays an important role in autonomous driving. Previous works mostly focus on single modality [24, 46, 56, 57, 72, 73, 81, 109, 115, 116] and different perception tasks are separated into individual models [2, 11, 52, 54, 117]. It is desirable to leverage complementary modalities to produce robust predictions and integrate multiple tasks within a model for the sake of computation budget. For instance, with the development of hardware, it is affordable to deploy both LiDAR and camera on a car, which are responsible to provide spatial information and semantic information. Integrating semantic-complementary vision tasks within a framework would greatly facilitate the deployment of real-world applications [5].

Recent advances have stayed tuned for multi-modality fusion [59, 75] and multi-task learning [57, 119] in the applications of 3D autonomous driving scenarios. Meanwhile, it is of great interest to unify multi-modality fusion and multi-task learning within a framework. However, we cannot expect that dumping all the individual components into one framework will function smoothly. Thus, we build up a competitive baseline based on BEVFusion [75], which takes as input both the point cloud and image, serving two complementary vision tasks: 3D detection (foreground) and map segmentation (background). However, we observe the severe issues of modality bias and task conflict: (a) different tasks prefer specific modality, e.g., 3D detection relies on spatial information provided by LiDAR sensor while segmentation task relies more on image inputs. (b) adding a new task will degrade the performance of both tasks.

From the perspective of optimization, we investigate the potential gradient imbalance that occurs during end-to-end training in a hierarchical view. First, we study the gradients which are produced by different task heads and are applied to update the parameters of the shared backbone. We observe that simply summing up these raw gradients to update the shared backbone would damage the performance of both tasks, suggesting an imbalance between them. Empirical findings prove that there is a great discrepancy between the gradient magnitudes w.r.t. the task objectives. Second, we inspect the gradients produced in the intra-gradient layer, which is to be separated into successive modality branches. Given a trained baseline, we visualize the gradient distributions of different modality branches and find great imbalance in their magnitudes. We further calculate the task accuracy by dropping one of the modalities to measure the modality bias. Our findings align with the theoretical analysis of [101], which suggests that the point cloud and image branches are suffering from the imbalanced convergence rate w.r.t. the downstream tasks.

By taking into account the findings discussed above, we propose a model to uni**f**y m**u**lti-modality mu**l**ti-task 3D perception via mu**l**ti-level gradi**e**nt calib**r**ation, dubbed as ***Fuller***. Specifically, we devise the multi-level gradient calibration, comprised of inter-gradient and intra-gradient calibration, to address the associated issues. In terms of the task conflict, we find that the task with lower gradient magnitude would be overwhelmed by another task with higher gradient magnitude. Thus, we propose to calibrate the gradients of different task losses at the backbone. Since the gradient would be manipulated at the layer level, this technique is referred to as **inter-gradient** calibration. Regarding modality bias, we expect the different modalities to update and converge at the same pace. Hence, before the gradients are separated into the modality branches, we calibrate their magnitudes to the same level, which is performed in the intra-gradient layer internally and thus called **intra-gradient** calibration.

On top of the gradient calibration, we introduce two lightweight heads for our tasks. These two heads are both transformer-based query heads. With our specially designed initialization methods, they can generate fine-grained results with just one decoder layer and thus can save much more parameters than dense heads.

9.2 Related Work

9.2.1 Visual Perception for Autonomous Driving

Autonomous driving relies on a perception system to gather information and understand the environment. Visual perception, as the most similar sensing modality to humans, provides high-resolution images that satisfy almost all tasks required for autonomous driving. Some of the tasks have long been studied beyond autonomous driving scenarios. Chen et al. [8] predict 2D object detection from images while Semantic FPN [48] performs semantic segmentation and Lanenet [103] implements lane detection respectively using visual inputs. Though these models are designed for different tasks, they all adopt the backbone-header architecture, some of which even share the same backbone structure like ResNet [37] or transformer [22]. Running independent models for perception tasks separately is a waste of time and computation resources, thus calling for the development of a unified perception system.

LiDAR and image are the two most powerful and widely used modalities in the area of autonomous driving. Multimodal fusion has been well-studied to boost the performance of 3D object detection tasks [2, 11, 59, 98, 117]. Multi-task networks of 3D perception also arouse significant interest in the autonomous driving community. These multi-task studies are limited to uni-modal network architectures, either with a LiDAR backbone [24, 46, 115] or an image backbone [57, 72, 73, 81, 109]. MMF [58] works on depth completion and object detection with both camera and LiDAR inputs, but depth estimation only works as an auxiliary head and only object detection was evaluated. BEVFusion [75] is the first multimodal network to perform object detection and map segmentation simultaneously. However, BEVFusion [75] focuses on single task and network acceleration, and only provides two pieces of joint training results. Our proposed method is the first multimodal multitask network, and we evaluate each task and analyze them from multi-modality and multi-task learning perspectives.

9.2.2 Multi-task Learning

Multi-task methods are mainly divided into two categories [96], network architecture improvement [33, 76, 87, 111] and optimization methods [13, 66, 69]. Multi-task learning jointly trains shared parameters on multiple tasks, mining latent information among them to improve efficiency and accuracy. Famous multi-task learning models include Mask R-CNN [36], which applies Faster R-CNN [86] as the backbone and conducts instance segmentation and object detection at the same time. Other methods like Eigen et al. [23] address depth prediction, surface normal estimation, semantic labeling tasks, and MultiNet [94] provide prediction on classification, detection, and semantic segmentation tasks within a single model. YOLOP [105] leverages CSPDarknet as the backbone, which branches out three task-specific heads for object detection, drivable area segmentation, and lane detection prediction. Standley et al. [90] and Christopher et al. [25] improve the previous multi-task training schema by grouping proper tasks together rather than naively training all tasks together.

The goal of multi-task optimization methods is to balance the loss weights of different tasks to prevent one task from overwhelming another during training. DWA [69] adjusts the loss weights based on the rate at which the task-specific losses change, but it requires that the loss magnitudes are balanced beforehand. Gradnorm [13] balances the loss weights automatically by stimulating the task-specific gradients to be of similar magnitude. IMTL [66] optimizes the training process by guaranteeing the aggregated gradient has equal projections onto individual tasks. Yet they have not been studied in the domain of multi-modality multi-task learning. In this chapter, we focus on developing general and effective approaches for multi-task learning in autonomous driving scenarios.

9.2.3 Multimodal Learning

Multimodal learning is increasingly used to improve the performance of certain tasks, such as action recognition [27, 43, 45], visual question answering [1, 41], and perception tasks in autonomous driving [2, 59, 75]. Most multi-modality research focuses on the network structure, such as concatenation, convolution or gated fusion in the middle, or later part of the network [40, 47, 79]. Few studies [80, 100] concentrate on multimodal optimization methods during the training process. [100] proposes the overfitting-to-generalization ratio (OGR) to quantize the significance of overfitting and try to solve it with Gradient Blending. It designs modal heads for each task and it is thus difficult to expand to a multi-task network. OGM-GE [80] try to solve the optimization imbalance problem by dynamically adjusting the gradients of different modalities. Since it separates parameters of different modalities in the linear classification head, it is hard to generalize to other complicated task heads. Differently, our method can be used in networks with any task head as long as the network has modal-specific parameters.

9.2.4 Pre-training Methods

A dominating paradigm in computer vision is to pretrain on a large scale of data, e.g., ImageNet [88], then finetune on target tasks with usually less training data. Recently, researchers are interested in learning visual representations without human supervision. We roughly divide them into three categories, namely, classification-oriented, detection-oriented, and segmentation-oriented methods. Classification-oriented methods typically rely on contrastive learning and online clustering, e.g., SimCLR [9], MoCo [10, 34], and BYOL [32]. Detection-oriented methods like DetCo [108] are specially designed for object detection by conducting contrastive learning between the global image and local patches. Segmentation-oriented methods like PixPro [110] further work on pixel-level correspondence to benefit dense prediction downstream tasks. We show that these methods are sub-optimal for multi-task learning, and we focus on improving the performance of these off-the-shelf methods instead of redesigning the computation-intensive pre-training stage.

9.2.5 Prompt-Based Learning

Prompt-based learning [38, 68, 104] is put forward to bridge the gap between pre-training and model tuning in the field of natural language processing. GPT-3 [3] first designs various text prompts according to the property of tasks and treats the downstream task as a masked language modeling problem. Meanwhile, other approaches like [55, 71, 121] train learnable continuous prompts in the embedding space of the model and achieve competitive performance compared to finetuning. Recently, CLIP [82], which is trained on multi-modality vision-language pairs data, achieved impressive performance on zero-shot image classification by injecting visual categories into the text input as a prompt. Subsequent works [26, 114, 123] further tunes CLIP with learnable soft prompts by few-shot supervisions. In the field of computer vision, prompt tuning is introduced by injecting learnable vectors in the input space [44] or inserting lightweight blocks to learn prompts [77]. However, these approaches are tailored for solving downstream tasks independently, and are inapplicable to heterogeneous multi-task learning. In this work, we design general and task-specific prompts to learn task-invariant and task-specific knowledge for heterogeneous multi-task learning.

9.3 2D Perception

9.3.1 Empirical Study

9.3.1.1 Preliminaries on Multi-task Learning

Multi-task Architectures

Multi-task learning (MTL) architectures apply parameter sharing to learn shared information between different tasks. MTL architectures can be divided into encoder-focused architectures [28, 69, 76, 87] and decoder-focused ones [95, 111, 120] according to parameter sharing scope. Encoder-focused architectures can be further categorized into hard and soft parameter sharing. In this chapter, we select the hard parameter-sharing structure as our backbone due to its simplicity and stability. Parameters are only shared in the encoder part of the model followed by task-specific heads. As Fig. 9.1 shows, the image inputs first go through the shared encoder, and then the feature map is fed into different heads to produce corresponding predictions. The multi-task methods can be divided into three types as in [60].

Task Scheduling

Task scheduling is the process of choosing which task or tasks to train on at each training step. Some scheduling methods arrange the task orders during the training process in a fixed order like Round-Robin [118], while others may sample tasks following specific distributions [63], like Uniform sampler and Weighted sampler. Specifically, Uniform sampler samples tasks from a uniform distribution and Weighted sampler samples tasks with weight proportional to the number of training epochs of each task. We test the above three task scheduling methods in our investigation and compare their performances.

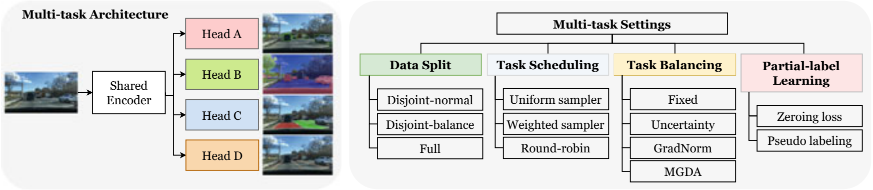


Fig. 9.1 The multi-task architecture and settings in our investigation. We follow the common multi-task architecture where each task shares the same encoder and has its specific head. The multi-task settings focus on three types of task scheduling, four task balancing methods, and two partial-label learning techniques and cover three common data split settings

Task Balancing

Task balancing is designed to deal with the gradients between tasks for the shared parameters in the network. When dealing with multiple tasks, the shared parameters are likely to be dominated by the one with a large gradient magnitude or confused by conflicting gradients. It is intuitive to apply weights over these gradients to balance among tasks, and several methods have been proposed, including (1) Fixed weighting, which fixes all loss weights during training; (2) Uncertainty weighting [46], introducing the task-dependent Homoscedastic uncertainty as the basis for weighting losses by maximizing the Gaussian likelihood of the uncertainty; (3) GradNorm [13], calculating the product of $L2$ norm of task gradient and the relative inverse learning rate as the indicator of the task learning pace, and then setting task weights to minimize the learning pace difference among tasks to balance the training process; (4) MGDA [89], treating the Multi-Task Learning problem as a multi-objective optimization problem by using multiple gradient descent algorithm [18]; (5) ParetoMTL [65], which finds a solution called Pareto optimal solution where all tasks losses can decrease without increasing the loss on other tasks.

Learning on Partial Labels

Image segmentation task requires annotations of labels to every pixel of the image, which is time-consuming and it is thus hard to get enough annotations. To deal with the missing annotation problem, two different methods are introduced, including Zeroing loss [51, 107] and Pseudo labeling [29]. Zeroing loss [51, 107] simply zero losses for a particular task if the input image does not have the corresponding annotation. Pseudo labeling [29] first trains a teacher model on fully labeled data. Then the teacher model is used to label the missing annotations to create a multitask pseudo-labeled dataset.

9.3.1.2 Multi-task Setup

We focus on four major tasks in autonomous driving, i.e., object detection, semantic segmentation, drivable area segmentation, and lane detection. We choose the large-scale BDD100K [118] dataset as the main dataset, which contains diverse heterogeneous tasks. Examples of each task are shown in Fig. 9.1. We choose the hard-parameter sharing structure for efficiency and select the state-of-the-art detector and segmentation head to build up our model.

Encoder

The encoder consists of a backbone network and a neck network. We choose the Swin transformer [74] as the backbone to extract features of the input image. The output of the backbone is denoted as $\{C_2, C_3, C_4, C_5\}$. Then we adopt Feature Pyramid Network (FPN) [64] module for the neck network to fuse features generated by the backbone. The output of the neck is denoted as $\{P_2, P_3, P_4, P_5\}$.

Detection Head

For the detection task, we choose Sparse R-CNN [91] as the detection head. Sparse R-CNN directly generates N_{detect} of candidates from the output of the last layer of the Feature Pyramid Network. According to the object number distribution in the BDD100k dataset, we decide to set the candidate number $N_{detect} = 300$. Set prediction loss is applied to find the optimal bipartite matching between predictions and ground truth objects.

Segmentation Head

For segmentation-based tasks, we choose state-of-the-art segmentors, e.g., MaskFormer [14] and Semantic FPN [48], as the segmentation head. MaskFormer or Semantic FPN takes in multi-layer output from Feature Pyramid Network $\{P_2, P_3, P_4, P_5\}$. Features from each layer are up-sampled to $1/4$ scale and summed element-wise. Then this merged feature map is again upsampled by a factor of 4, followed by softmax to produce the classification score for every pixel at the original resolution.

Optimization

Our multi-task loss contains specific parts for different task heads. For object detection, we adopt the same objective as in Sparse R-CNN [91]. For segmentation-based tasks, i.e., semantic segmentation, drivable area segmentation, and lane detection, we employ the same loss as in MaskFormer [14] or Semantic FPN [48]. Therefore, the total loss for the multi-task model is formulated as follows:

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_{\text{det}} + \lambda_2 \mathcal{L}_{\text{sem}} + \lambda_3 \mathcal{L}_{\text{driv}} + \lambda_4 \mathcal{L}_{\text{lane}}, \tag{9.1}$$

where $\mathcal{L}_{\text{det}}$, $\mathcal{L}_{\text{sem}}$, $\mathcal{L}_{\text{driv}}$, $\mathcal{L}_{\text{lane}}$ represent objectives for object detection, semantic segmentation, drivable area segmentation, and lane detection, respectively. λ_1, λ_2, λ_3, and λ_4 stand for weighting factors for different parts.

9.3.2 Dataset

In BDD100K [118], 70k training images are labeled for object detection, lane detection, and drivable area segmentation, and only 7k training images are labeled for semantic segmentation. These two sets are not disjoint and about 3k images have complete annotations for all three tasks. As is described in [29], data annotation is one of the biggest challenges for training a multi-task model. In real-world scenarios, it is unrealistic to obtain complete types of annotations for all input images, especially in the traffic scene where fine-grained understanding is required. In this chapter, we are interested in the performance of multi-task learning with different quantities of annotations. Thus, we mainly consider the following scenarios that correspond to three different levels of annotation scarcity.

9.3.2.1 Disjoint-Normal Setting

Under the same annotation effort or budget, the quantity of labels decreases with the increase in complexity of labeling a task. Hence, in this setting, we consider a realistic scenario where each input image is labelled to only one task, namely, the annotations of each task are disjoint, and the number of labelled images for each task is in decreasing order as follows: drivable area segmentation (20k), object detection (10k), semantic segmentation (7k), lane detection (20k).

9.3.2.2 Disjoint-Balance Setting

We further consider a more difficult setting with the lowest quantity of annotations that corresponds to the scenarios of scarce annotations. There are 21k images in this setting and each task has 7k labeled images that are not overlapped with other tasks.

9.3.2.3 Full Setting

Full setting refers to experimenting on all available annotations on ∼74k images in BDD100K and can be used to analyse the upper bound of different methods.

9.3.3 Pretrain-Finetune for Multi-task Learning

In this section, we extensively investigate the performance of different types of pre-training methods when transferring to multi-task scenarios, including supervised pre-training, classification/detection/segmentation-oriented methods, and vision-language pre-training methods.

9.3.3.1 Revisiting Pre-training Methods

Supervised Pre-training

We select the weights pretrained on the ImageNet, which is the most widely adopted pretrained model in the past decade.

Classification-Oriented Methods

Currently, the state-of-the-art classification-oriented pre-training methods are mainly based on contrastive learning and online clustering. During the pre-training phase, they produce image-level prediction using global features and minimize the distance between positive pairs while pushing the representations of negative pairs apart. This eliminates the need for computation-intensive generation steps in previous generative methods [20, 21]. However, they may lack spatial sensitivity for fine-grained tasks since the spatial details are not considered in their formulation. We select MoCo-v1 [34], MoCo-v2 [34], SimCLR [9], SwAV [7], BYOL [32] for comparison.

Detection-Oriented Methods

DetCo [108] is specially designed for object detection by enforcing contrastive learning between global images and local patches with multi-level supervision. It achieves good performance on object detection while maintaining competitive classification accuracy.

Segmentation-Oriented Methods

Different from the aforementioned ones, this type of methods pursue pixel-level self-supervised learning for learning dense feature representations. For example, PixPro [110] proposes a pixel-to-propagation consistency task, where two asymmetric pipelines are utilized to obtain positive pixel pairs. We select DenseCL [102] for comparison. We download self-supervised pretrained models from MMSelfSup repository.[1]

9.3.3.2 Comparisons of Pre-training Methods

We report the performance of the aforementioned methods in Table 9.1, and come to the following observations:

- Many methods encounter substantial degradation on pixel-level segmentation tasks. For semantic segmentation, MoCo-v1/v2 and DenseCL achieve the worst mIoU. For drivable area segmentation, MoCo-v1/v2, DenseCL, and CLIP have

[1] https://github.com/open-mmlab/mmselfsup.

Table 9.1 Comparisons of different paradigms under the Disjoint-normal setting with ResNet-50 backbone. Orange color indicates the results of the novel *pretrain-adapt-finetune* paradigm, while others are results of conventional *pretrain-finetune* paradigm

		Semantic Seg.		Drivable Seg.		Object Detection		
Type	Model	mIoU	pACC	mIoU	pACC	mAP	AP50	AP75
Classification-oriented	MoCo-v1 [34]	17.8	48.6	70.8	92.0	25.8	49.5	23.1
		59.2	93.2	83.6	96.9	25.9	50.0	23.0
	MoCo-v2 [10]	10.3	19.8	73.4	93.5	26.0	50.1	23.2
		61.2	93.4	83.8	96.9	26.1	50.4	23.4
	SimCLR [9]	60.3	93.3	83.5	96.8	25.4	48.9	22.5
		60.1	93.2	83.5	96.8	25.2	48.9	22.3
	SwAV [7]	45.9	71.1	82.0	96.5	25.6	49.1	23.0
		61.1	93.3	83.1	96.7	25.6	49.3	23.1
	BYOL [32]	59.2	90.2	75.6	93.9	25.9	49.8	23.4
		61.7	93.4	83.5	96.8	25.7	49.4	23.1
Detection-oriented	DetCo [108]	38.1	58.5	83.2	96.7	25.9	49.7	22.9
		61.0	93.4	83.7	96.9	26.2	50.3	23.3
Segmentation-oriented	DenseCL [102]	20.0	40.0	73.7	93.7	26.1	50.3	23.5
		60.7	93.3	83.9	96.9	26.3	50.3	23.7
Vision-language	CLIP [82]	54.5	91.1	74.1	93.1	26.5	50.7	23.8
		61.0	93.2	83.4	96.8	26.3	50.5	23.5

the worst performance. We hypothesize that the architecture of task-specific heads and the domain gap cause between pretraining and finetuning degradation.

- Only SimCLR achieves decent performance on all three tasks.
- The pre-training paradigm does not seem to have an explicit correlation with the downstream performance, and even the task-oriented design in pre-training does not guarantee the transfer performance on the corresponding task type. For example, the segmentation oriented DenseCL achieves sub-optimal performance on two segmentation tasks.

9.3.4 Effective Adaptation for Multi-task Learning

9.3.4.1 Pretrain, Adapt, Then Finetune

The state-of-the-art pre-training methods mentioned above perform well when transferred to a single task such as image classification and object detection. However, most of them cannot achieve good performance simultaneously in multi-task learning, which is referred to as degraded transferring performance. We attribute the

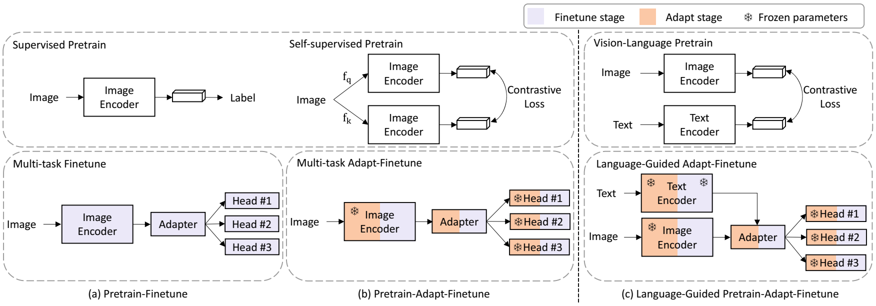


Fig. 9.2 Comparisons of the conventional *pretrain-finetune* paradigm and the novel *pretrain-adapt-finetune* paradigm. The language-guided *pretrain-adapt-finetune* paradigm further incorporates language priors into multiple downstream tasks

degraded transferring performance of the state-of-the-art pre-training methods to two critical factors, i.e., optimization gap and architectural gap. Firstly, the high specialization of the pre-training methods and the heterogeneity of downstream tasks result in the optimization gap in the classic *pretrain-finetune* paradigm. To be specific, the objective for pre-training is usually a type of constastive loss (e.g., InfoNCE [78]) while the fine-tuning stage is supervised by a weighted sum of several task-specific losses. Secondly, we notice that many models in Table 9.1 are pretrained in a model with convolutional head, e.g., Fast R-CNN detector [30] and convolution-based projection head. In contrast, we adopt the transformer-based MaskFormer head for segmentation tasks. The distinctions between convolution and transformer lied in the inductive bias (local v.s. global) and the internal representation structure [83] and we conjecture this plays a non-negligible role in transferring pre-trained models.

We draw inspiration from the recent progress of prompt-based learning, where prompt tuning has emerged as a new alternative to fine-tuning. For example, P-Tuning v2 [70] matches the fine-tuning performance by only tuning learnable prompts while freezing the large-scale language model. Following this philosophy, we propose a simple but yet effective *pretrain-adapt-finetune* paradigm for the effective adaptation of those off-the-shelf pretrained models without redesigning the resource-intensive pre-training stage.

During the *adaptation* stage, we are given the pretrained model weights (e.g., ResNet-50) inherited from the *pre-training* stage. We aim to transform the model via a small amount of learnable parameters to adapt the knowledge of the pretrained weights towards multi-task scenarios. To this end, we freeze the random initialized task-specific heads and the backbone, while tuning the parameters of FPN (feature pyramid network) supervised by the multi-task loss function in Eq. 9.1. We compare different schemes for multi-task learning in Fig. 9.2. This *adaptation* stage characterizes a few critical design choices: (1) This stage bridges the gap between *pre-training* and *fine-tuning* by including the pretrained weights and multi-task objectives simultaneously and the frozen backbone prevents the pretrained weights from being spoiled before the *finetune* stage. (2) As opposed to the single-layer prompt/adapter in P-

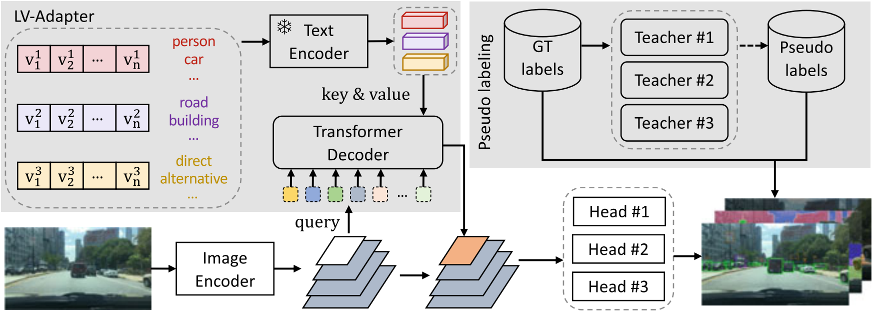


Fig. 9.3 An overview of LV-Adapter. We first train three specialized teacher on labeled data to generate pseudo labels for each task. The multi-task model is then trained on both ground-truth and pseudo labels under the language-guided *pretrain-adapt-finetune* paradigm

Tuning [53] and CLIP-Adapter [26], the FPN acts as a multi-scale adapter that are endowed with greater capacity and provides semantically stronger features for fine-grained downstream tasks. We have experimented with more sophisticated adapters (e.g., scale-aware adapters, lightweight transformer) and observed marginally better results. Hence, we opt for the original architecture above. We experimentally verify that our *pretrain-adapt-finetune* paradigm significantly improves the performance and stability of different kinds of pre-training methods without increasing the total training costs.

9.3.4.2 Language-to-Vision Adapter

From the perspective of transfer learning, CLIP can serve as a complementary scheme to enhance the *pretrain-adapt-finetune* paradigm, since it can comprehend the concepts in natural language as well as the correspondence between visual and linguistic features. We note that the result of CLIP in Table 9.1 only reuses the weights of its image encoder and discards the text encoder. Therefore, we take a step further to explicitly exploit the knowledge in the full CLIP model.

Basically, the CLIP model excels in aligning the visual and language embeddings and some works [84, 122] claim that the textual features generated by CLIP have meaningful correspondence to the semantic regions in an image. Therefore, we pursue underpinning the compatibility between the semantic concepts of each task and the image features in order to generate semantically stronger context for downstream tasks. We outline the resulting model, named LV-Adapter, in Fig. 9.3, and elaborate on each component in what follows.

Pixel-Class Correspondence

The CLIP model adopts ResNet [37] and BERT [19] as the encoders for image and text, respectively. Formally, we denote the last output feature maps from ResNet stages $\{C_i\}_{i=2}^{5}$ as $\{\mathbf{x}_i\}_{i=2}^{5}$. In the CLIP model, attentional pooling and L_2 normalization are applied to $\mathbf{x}_5 \in \mathbb{R}^{H_5 W_5 \times C_5}$ to produce the global image feature $\hat{\mathbf{I}}_e \in \mathbb{R}^{1 \times C_5}$ for zero-shot inference, which can be formulated as follows:

$$[\hat{\mathbf{I}}_e, \hat{\mathbf{x}}_5] = \text{L2\_NORM}(\text{MHSA}(\text{GAP}(\mathbf{x}_5) \oplus \mathbf{x}_5)), \tag{9.2}$$

where GAP$(\cdot)$ denotes global average pooling, MHSA$(\cdot)$ denotes multi-head self-attention [97], and $\oplus$ denotes concatenation operation. To extract textual features of each class, class-specific prompts are constructed via a prompt-generating function and fed into a text encoder and we denote the normalized output features for N classes as $\hat{\mathbf{T}}_e \in \mathbb{R}^{N \times C}$, which can be formulated as follows:

$$\hat{\mathbf{T}}_e = \text{L2\_NORM}(\text{TE}(\text{Gen}(\{\mathbf{n}_i\}_{i=1}^{N}))), \tag{9.3}$$

where Gen$(\cdot)$ is the generator function for class-specific prompts, TE is text encoder, and $\{\mathbf{n}_i\}_{i=1}^{N}$ is the embeddings of class names. Though $\hat{\mathbf{T}}_e$ and $\hat{\mathbf{I}}_e$ are well aligned in the image-text contrastive pre-training, they do not preserve any spatial details and are not readily applicable for downstream fine-grained tasks. In contrast, the multi-level features output by FPN are semantically rich in the spatial dimension, but are not directly aligned in the pre-training stage. Therefore, we pursue to learn task-specific prompts and propose a language-to-vision adapter to enhance the pixel-class correspondence.

Learning Task-Specific Prompts

In the original implementation of CLIP, the prompt-generating function Gen$(\cdot)$ outputs hand-crafted prompts using the predefined template, e.g., "`a photo of a [CLS]`." However, the performance is highly sensitive to the form of the template [82, 123], which may be sub-optimal when transferring to heterogeneous downstream tasks. Hence, we follow CoOp [123] to use learnable textual contexts in prompting for each task. In spite of the inconsistent output formats of three tasks (4-D box v.s. dense output), they both need to determine the category of boxes or pixels, and we incorporate the class names of each task in prompting. The task-specific prompting can be formulated as follows:

$$\hat{\mathcal{T}}_{e,i} = \text{L2\_NORM}(\text{TE}(\text{Gen}(\mathbf{v}_i \oplus \mathbf{n}_i))), \tag{9.4}$$

where $\mathbf{n}_i$ refers to the embedding of class name belonging to a specific task, and $\mathbf{v}_i$ is the learnable contexts.

Learning Language-to-Vision Adapter

To align the textural features and the dense FPN features, we propose a Language-to-Vision Adapter to incorporate the language priors into the visual features. Formally, we denote the last feature map of P_5 as $\mathbf{z}_5 \in \mathbb{R}^{H_5 W_5 \times C}$, and we aim to learn an adapter function $\mathcal{A}_{L\rightarrow V}$ to generate language-aware context for downstream tasks. We utilize the cross-attention mechanism in Transformer decoder [97] for Language-to-Vision adaptation:

$$\mathcal{A}_{L\rightarrow V}(\hat{\mathcal{T}}_e, \mathbf{z}_5) = \text{TransDecoder}(q = \mathbf{z}_5, k\&v = \hat{\mathcal{T}}_e), \tag{9.5}$$

where the q, k, v stands for query, key, and value, respectively. A single linear fully connected layer is used to adjust the channel number of $\hat{\mathcal{T}}_e$ and we omit it in Eq. 9.5 for simplicity. We denote the output of Equation 9.5 as $\tilde{\mathbf{z}}_5 \in \mathbb{R}^{H_5 W_5 \times C}$ and we simply substitute $\mathbf{z}_5$ with $\tilde{\mathbf{z}}_5$ and leave the task-specific head design unchanged.

9.3.5 GT-Prompt

The key to multi-task learning is to learn general and explicit representations among tasks, and establish relationships between them. Therefore, a good multi-task learning framework should take full advantage of task-agnostic and task-specific knowledge, and guide the model to learn better representations. To this end, we introduce our proposed GT-Prompt, which consists of three components: (1) general prompts (G-Prompt) generated by lightweight prompt blocks to learn task-invariant knowledge; (2) task-specific prompts (T-Prompt) produced by pre-trained image encoder to encode task-specific information; (3) a task-specific fusion module to integrate the visual representation and task-specific prompts.

9.3.5.1 General Prompt

The general prompt is learnable task-invariant knowledge for all tasks. To sufficiently exploit semantic information from the pre-trained backbone, we construct multiple layer-wise prompt blocks (denoted as G-Prompt blocks), which are plugged into the backbone and they extract task-agnostic information. The illustration of G-Prompt is shown in the bottom left of Fig. 9.4.

Specifically, layer-wise G-Prompt blocks are inserted into each layer of the transformer backbone. Given an input image, the backbone produces intermediate layer-wise feature maps after self-attention modules, which are defined as $\{Attn^1, Attn^2, ..., Attn^i\}$. Layer-wise G-Prompt blocks are represented as $\{G^1, G^2, ..., G^i\}$. We suppose that the input feature maps for each transformer layer are

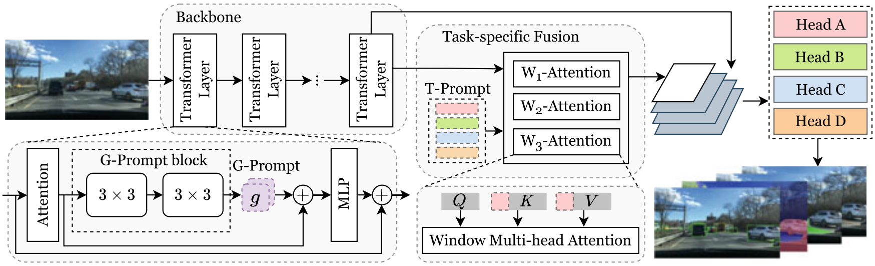


Fig. 9.4 The architecture of the proposed GT-Prompt consists of (1) G-Prompt generated by lightweight prompt blocks to learn task-invariant knowledge; (2) T-Prompt produced by a pre-trained image encoder to encode task-specific information; and (3) a task-specific fusion module to integrate the visual representation and task-specific prompts

$\{x^1, x^2, ..., x^i\}$, then the feature maps output from the attention layer can be formulated as:

$$y^i = Attn^i(x^i), \tag{9.6}$$

in which y^i represents intermediate feature maps after the attention layer. G-Prompt blocks further generate G-Prompt:

$$g^i = G^i(y^i), \tag{9.7}$$

where g^i represents layer-wise general prompt, which is blended with the intermediate feature maps y^i:

$$x^{i+1} = x^i + MLP(y^i + \beta g^i) \tag{9.8}$$

where β denotes a learnable parameter to balance two terms, and MLP is a neural network with two hidden layers and a GELU nonlinearity [39]. For efficiency and effectiveness, the G-Prompt block is built with two 3 × 3 depthwise convolutional layers [15].

9.3.5.2 Task-Specific Prompt

The overview of our task-specific prompts is shown in Fig. 9.5. Task-specific prompts should encode task-specific information which helps the model understand tasks better, thus we leverage exemplars to generate T-Prompt.

Here we adopt an image encoder pre-trained on ImageNet classification to generate task-specific prompts. For visual perception, the ground-truth annotations provide hints of shapes and sizes of different objects, motivating the model to choose the best window size and mix multi-scale information specifically. For object detection, given an image I and its annotated bounding boxes $R = \{v_1, v_2, ..., v_n\}$, task-specific prompt aims to mark the image regions with class-specific markers. Here we use the

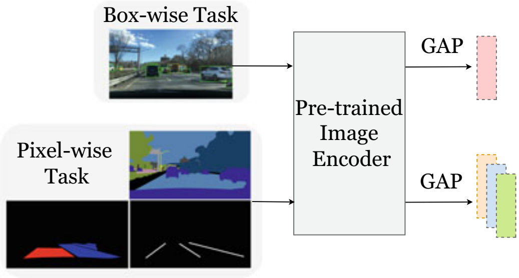


Fig. 9.5 Details of task-specific prompts. For the box-wise task, we leverage exemplars with colored bounding boxes to generate the task-specific prompt. For pixel-wise tasks, we adopt selected annotated masks to produce prompts

colored bounding boxes to mark all objects uniquely as in visualization. We choose the image encoder whose architecture is the same as the backbone and denote the last output feature map from the image encoder C_5 as x_5. After selecting the visualization map, we pass it into C_5 and apply a Global Average Pooling (GAP) on it to get the task-specific prompt:

$$p = \text{GAP}(x_5) \in \mathbb{R}^{1\times 1\times D}. \tag{9.9}$$

For pixel-wise tasks (i.e., semantic segmentation, drivable area segmentation, and lane detection), we extract features of corresponding ground-truth masks of exemplars by Eq. 9.9. In this way, we get task prompts p_{det}, p_{sem}, p_{driv}, p_{lane} for object detection, semantic segmentation, drivable area segmentation and lane detection, respectively.

9.3.5.3 Task-Specific Fusion

The key idea of our task-specific fusion module is to prepend the related task-specific prompt to the key and the value of the window multi-head attention at each transformer layer. Since the fixed window size in the multi-head attention cannot capture diverse multi-scale information for different tasks, we introduce multi-scale windows to perform multi-scale window multi-head attention as in Fig. 9.4.

Here we suppose that the task-specific fusion module contains h heads and n_{win} scales of windows. We first evenly divide the input feature maps $\hat{x}$ into n_{win} groups:

$$\{\hat{y}_i\} = \text{Split}(\hat{x}) \in \mathbb{R}^{\frac{h}{n_{win}}\times H\times W\times \frac{D}{h}}, i = 1, ..., n_{win}, \tag{9.10}$$

where $\hat{y}_i$ indicates intermediate feature maps. Then we apply multi-head attention layers with different window sizes to different groups. Specifically, given a group of input feature maps $\hat{y}_i$, the query Q_i, key K_i, and value V_i are learned by linear projections:

$$\begin{aligned} Q_i &= \hat{y}_i W_q, \\ K_i &= \hat{y}_i W_k, \\ V_i &= \hat{y}_i W_v, \end{aligned} \tag{9.11}$$

where W_q, W_k, and W_v indicate learnable parameters for query, key, and value, respectively. To learn task-specific information for different tasks, we prepend specific T-Prompt p to key and value respectively, and obtain a new key or value:

$$\begin{aligned} K_i' &= p \oplus K_i, \\ V_i' &= p \oplus V_i, \end{aligned} \tag{9.12}$$

where K_i' and V_i' represent new key and value, and $\oplus$ is concatenation. Then we apply window multi-head attention with multi-scale windows to get task-specific information:

$$O_i = \mathrm{MLP}(\mathrm{W}_i\text{-Attention}(Q_i, K_i', V_i')), \tag{9.13}$$

where O_i denotes the output feature, and W$_i$-Attention represents window multi-head attention [74]. We concatenate multi-scale output feature maps and get the task-specific feature O as follows:

$$O = O_1 \oplus O_2 \oplus ... \oplus O_i, i = 1, ..., n_{win}, \tag{9.14}$$

In this way, we get task-specific features O_{det}, O_{sem}, O_{driv}, O_{lane} for object detection, semantic segmentation, drivable area segmentation and lane detection, respectively. These features are then processed by subsequent FPN neck and task-specific heads, which predict results for different tasks.

9.4 3D Perception

9.4.1 Dataset

nuScenes [5] is a multi-sensor datasets with diverse annotations to support multiple tasks such as detection, tracking and especially BEV map segmentation which is usually absent in other datasets. There are 28,130 training samples and 6,019 validation samples, each comprising one 32-beam LiDAR scan and 6 multi-view images. Regarding 3D detection, there are 10 foreground categories, and the mean Average Precision (mAP) and nuScenes Detection Score (NDS) are used to evaluate the performance. As for map segmentation, the model is required to segment 6 background categories in BEV view, which is measured by the mean Intersection over Union (mIoU).

9.4.2 *Fuller*

In this section, we introduce the Fuller, a framework that unifies the multi-modality multi-task 3D perception in autonomous driving scenarios. Fuller aims to mitigate the problem of modality bias and task conflict during the end-to-end training, which is accomplished by gradient calibration in a hierarchical way. Regarding the network architecture, we introduce a light-weight design for the task heads, named Fuller-det and Fuller-seg.

9.4.2.1 Network Architecture

As shown in Fig. 9.6, our proposed Fuller extracts features from both LiDAR point cloud and images, then transforms them into a unified bird's-eye view (BEV) representation. It relies on VoxelNet [124] as LiDAR backbone and Swin-T [74] as image backbone. As for image features from multi-view cameras, we adopt same strategy as LSS [81] to project them onto BEV. We adopt the modality fusion strategy where the features of two branches, f^{img} and f^{lid}, are first concatenated and then fed into the fusion block:

$$f^{fuse} = \mathtt{conv}(f^{lid} \oplus f^{img}), \tag{9.15}$$

where $\mathtt{conv}$ is the modal fusion block and $\oplus$ is concatenation. f^{fuse} is then connected to task-specific heads.

The detection head Fuller-det is in DETR [6] style with object queries. Given the fusion feature f^{fuse}, Fuller-det initializes the queries using an auxiliary heatmap head according to TransFusion [2]. Specifically, we sort out the top-k candidates

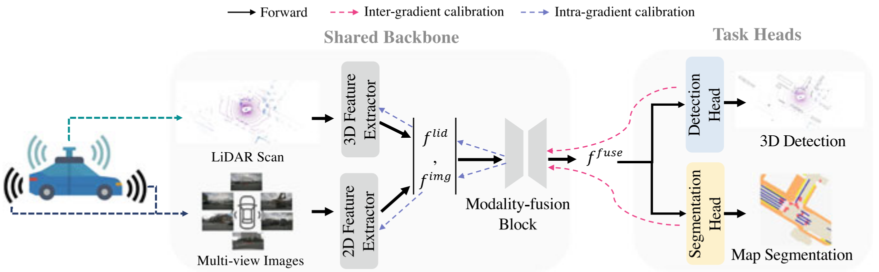


Fig. 9.6 Framework of the Fuller. Generally, Fuller takes as input the LiDAR scan and multi-view images and predicts two tasks: 3D detection and map segmentation. Fuller introduces multi-level gradient calibration to deal with the problems of task conflict and modality bias during optimization: (i) The gradients, produced by the task heads and applied on the shared backbone, will be calibrated on the last layer of the backbone before it is further back-propagated, namely, inter-gradient calibration (pink dashed line). (ii) When it comes to the subsequent modality branches of the shared backbone, the gradient magnitudes will be calibrated again to the same level within the intra-gradient layer, referred to as intra-gradient calibration (blue dashed line)

from the local maxima positions of heatmaps as object queries ranked by confidence scores.

Fuller-seg is also a query based semantic segmentation head with segmentation queries. We firstly transform BEV feature f^{fuse} to output shape feature $F \in \mathbb{R}^{H \times W \times C}$. Query feature is produced by projecting the one-hot category vector. Then the transformer decoder layer use initialized queries and transformed BEV feature F to get mask embeddings $M \in \mathbb{R}^{N \times C}$. Finally, binary mask prediction $S \in \mathbb{R}^{N \times H \times W}$ is obtained via a dot product between M and F, followed by a sigmoid activation.

Both Fuller-det and Fuller-seg have only one transformer decoder layer and can achieve comparable results as the state-of-the-art methods. Since the task heads are computationally effective, we focus on the optimization of the shared backbone.

9.4.2.2 Multi-level Gradient Calibration

We now introduce the multi-level gradient calibration. First, it will calibrate the gradient between tasks via inter-gradient calibration. When it comes to the subsequent modality branches of the backbone, the gradient will be calibrated again by intra-gradient calibration.

Inter-gradient Calibration for Task Conflict

By definition, the gradients will be propagated from the task heads to the shared backbone. Without any regularization, multi-task learning would simply sum up the individual gradients for backbone update. Since the gradients of the downstream tasks usually exhibit great distinction, this naive manner will inevitably result in task conflict. For example, an objective with low gradient magnitude would be overwhelmed by another one with high gradient magnitude. Therefore, existing works [13, 66, 69, 96] propose to *manipulate* the gradients to interfere with the optimization process.

Following this philosophy, we visualize the gradient distribution of the two tasks to inspect the inferior performance. Specifically, we compute the ratio of $L2$ norm between the gradients computed by raw individual losses:

$$\gamma_{\text{task}} = \frac{||\nabla_{\texttt{shared\_L}} \mathcal{L}_{\text{Det}}||}{||\nabla_{\texttt{shared\_L}} \mathcal{L}_{\text{Seg}}||}, \tag{9.16}$$

where ∇ denotes gradient computation operator, $\mathcal{L}_{\text{Det}}$ and $\mathcal{L}_{\text{Seg}}$ are the output losses for 3D detection and map segmentation, respectively. Usually, the shared backbone gradients computed by different task losses will be used to measure task characteristics. To save computation time, we chose the last layer of shared backbone as the `shared_L`. Thus, γ_{task} is a metric that reflects the gradient discrepancy.

As we might notice in Fig. 9.7, the γ_{task} between the two tasks is very large. Given this finding, we infer that the emergence of task conflict is probably because the

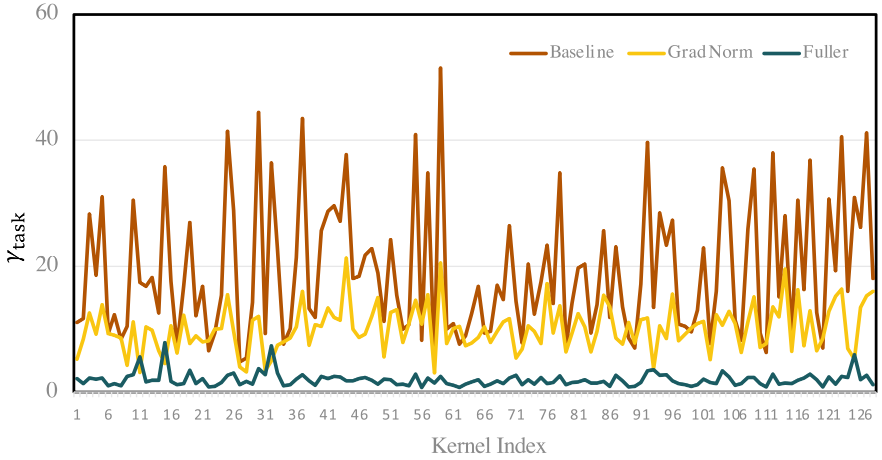


Fig. 9.7 We visualize the γ_{task} (9.16) in the first layer of the modality-fusion block. The gradient tensors are unfolded along the first axis. It is easy to observe that the gradient magnitude of seg loss dramatically lags behind that of the det loss. We apply the proposed Fuller and compare it with GradNorm [13]. We find that our method is able to balance the gradients from two tasks. Importantly, our method yields a more stable and lower γ_{task}

gradients of segmentation task are overwhelmed by those of detection task. Inspired by the loss weighting methods [13, 66, 69, 96], we balance the gradients of different tasks by balancing their loss weights. At each iteration, we obtain the gradients corresponding to individual loss on the last layer of the shared backbone. These gradients are utilized to derive the new loss weights. Then the aggregated loss is applied to calibrate the whole network gradients. We evaluate existing literature and choose the IMTL_G [66] as the technique for this purpose given its superior performance.

Intra-Gradient Calibration for Modality Bias

We have analyzed the impact of different task objectives on the backbone holistically. Another complicated situation arises when optimizing modality branches. During experiments, we notice the issue of modality bias which undermines the assumption that multiple modalities can collaboratively support the downstream tasks. This phenomenon is also known as semantic inconsistency [31] and modality imbalance [101].

The first layer of modality fusion block is denoted as intra-gradient layer, parameterized by θ^F. It comprises of two parts θ^F_{lid} and θ^F_{img}, which respectively represent the parameters directly connected with LiDAR and image backbones during back propagation. Let H denote the modality branches, where θ^H_{lid} and θ^H_{img} represent the parameters of the LiDAR and image branches, respectively. According to the chain rule, the gradient for a certain modality branch is defined as:

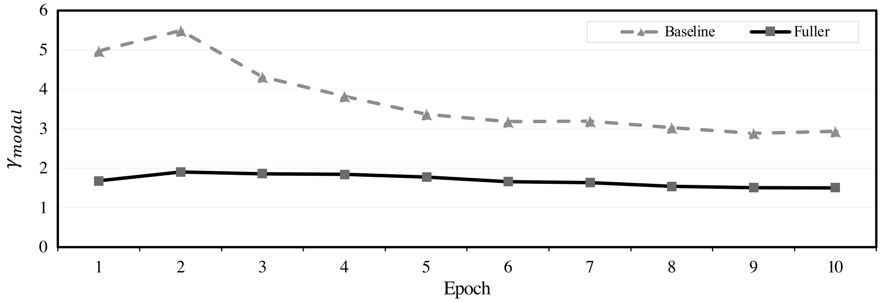


Fig. 9.8 Compared with baseline, Fuller has a lower γ_{modal} (9.18), meaning that two modalities can be learned in a balanced manner

$$G_{mod} = \frac{\partial \mathcal{L}}{\partial \theta^H_{mod}} = \frac{\partial \mathcal{L}}{\partial \theta^F_{mod}} \cdot \frac{\partial \theta^F_{mod}}{\partial \theta^H_{mod}}, \tag{9.17}$$

where $mod = \{lid, img\}$, and G_{lid} and G_{img} denote the gradients of the two modality branches. According to Eq. (9.17), $\nabla\theta^F_{lid} = \frac{\partial \mathcal{L}}{\partial \theta^F_{lid}}$ would carry out the updating message from the task heads to the LiDAR branch. The same applies for the image branch. image branch.

Regarding the term $\nabla_{\theta^F}\mathcal{L}$, this gradient corresponds to the optimization process whereby the intra-gradient layer coordinately fuses the two modalities to adapt the downstream tasks. Therefore, we use $\nabla\theta^F_{lid}$ and $\nabla\theta^F_{img}$ within the intra-gradient layer to establish the connection between two modality branches. As they are to be separated into different branches, we consider their relative magnitude during end-to-end training:

$$\gamma_{\text{modal}} = \frac{||\nabla\theta^F_{lid}||}{||\nabla\theta^F_{img}||}. \tag{9.18}$$

The result in Fig. 9.8 shows that in most of the time, $||\nabla\theta^F_{lid}||$ would surpass $||\nabla\theta^F_{img}||$, which means the LiDAR and image branches receive uneven attention from the downstream tasks.

To solve this problem, we propose to calibrate the gradients between two branches, i.e., G_{lid} and G_{img}. In practice, we gate the one with greater magnitude to slow down its pace, making the tasks pay balanced attention to both modalities. At t step, we obtain the gating factors as follows:

$$\begin{aligned} w^t_{lid} &= \sigma(||\nabla\theta^F_{lid}{}^t||, ||\nabla\theta^F_{img}{}^t||) \in (0, 1], \\ w^t_{img} &= \sigma(||\nabla\theta^F_{img}{}^t||, ||\nabla\theta^F_{lid}{}^t||) \in (0, 1], \end{aligned} \tag{9.19}$$

$$\sigma(x, y) = \mathbf{1}_{\frac{x}{y}>1}(1 - \mathtt{tanh}(\alpha \cdot \frac{x}{y})) + \mathbf{1}_{\frac{x}{y}<=1}, \tag{9.20}$$

where $\sigma(\cdot, \cdot)$ is a composition function conditioned by the indicator function and it is used to measure a paired input. α is a weighting factor. The gating factors in Eq. (9.19) are further smoothed by momentum update with coefficient m to stabilize the training. Then the calibrated gradient will be backpropagated to the associated branch:

$$w^t_{mod} = m \cdot w^{t-1}_{mod} + (1 - m) \cdot w^t_{mod}, \tag{9.21}$$

$$G^t_{mod} = w^t_{mod} \cdot G^t_{mod}. \tag{9.22}$$

We refer to this technique as intra-gradient calibration which is performed between modalities.

9.4.2.3 Fuller: The Blueprint

We have presented the inter-gradient and intra-gradient calibration in the hierarchical view. They are proposed to optimize the whole backbone and the associated modality branches, respectively. At each updating step, we first calculate the gradients w.r.t. the two objectives in the last layer of the shared backbone. The two gradients are calibrated to alleviate the problem of task conflict, where a pair of weights are derived. After applying the weights on the raw losses, we obtain the calibrated gradient of the total loss on the intra-gradient layer, $\nabla\theta^F_{lid}$ and $\nabla\theta^F_{img}$. To mitigate the issue of modality bias, we utilize them to calibrate the gradients of corresponding branches.

9.5 Experiments

9.5.1 2D Perception

9.5.1.1 LV-Adapter

Experiment Settings

We focus on three tasks, i.e., object detection, semantic segmentation, and drivable area segmentation, on the driving dataset BDD100K. The default parameters are as follows: The epoch is fixed as 36, syncBN is on, learning rate is set to 2.5×10^{-5}, and weight decay is 1×10^{-4}. The image scale is 1280×(720, 600). No other data augmentation is adopted. We adopt the AdamW optimizer with a warmup iteration of 1000 and the warmup factor is 0.01. For CLIP adaptation, the continuous prompts are prepended to the class, and the length of prompts is 16. During the adapt stage, backbone and heads are frozen, and the learning rate is set to 2.5×10^{-4}. The number of layers of transformer in language-to-vision adaptation is 3.

Pretrain-adapt-finetune Paradigm

We conduct experiments on popular self-supervised pretrained models. As shown in Table 9.1, when training with the conventional pretrain-finetune paradigm, the performances of state-of-the-art self-supervised methods are unstable, especially in semantic segmentation and drivable area segmentation. For example, MoCo-v2 only achieves 10.3 mIoU on semantic segmentation. In contrast, equipped with pretrain-adapt-finetune paradigm, MoCo-v2's performance is improved by a large margin in semantic segmentation (+50.9 in mIoU). MoCo-v1's performance is improved on drivable area segmentation by 12.8 mIoU. However, performances of pretrain-adapt-finetune paradigm are lower in the multi-task tuned pretrained type. We argue that these models are trained on more data and similar tasks, hence only finetuning them can yield good results. Most experiments show that the pretrain-adapt-finetune paradigm can better utilize knowledge from pretrained models.

Comparison with Baselines

We present results of single-task baselines and multi-task models under three settings in Table 9.2. The results show that our methods perform better on all metrics. In object detection and semantic segmentation, our method improves single-task baselines by 5$\sim$6 in mAP or mIoU, and further surpasses pseudo labeling greatly by 1$\sim$2 mAP or mIoU. Therefore, we conclude that language-guided pretrain-adapt-finetune paradigm can reduce the gap between the pretraining and downstream tasks and better utilize vision-language knowledge to solve multi-task problems.

9.5.1.2 Multi-task Methods on BDD100K

We study performances of popular existing multi-task methods under three settings on BDD100K, and we consider four heterogeneous tasks, namely, object detection, semantic segmentation, drivable area segmentation, and lane detection.

Partial-label Learning

As shown in Table 9.3, pseudo labeling [29] can improve performances especially in object detection and semantic segmentation compared with zeroing loss [106]. However, the improvement in drivable area segmentation and lane detection is not obvious, since some noisy pseudo labels may hamper the training process.

Table 9.2 Results of single-task baselines and multi-task models with ResNet-50 backbone. SS and DA respectively correspond to semantic segmentation and drivable area segmentation. – indicates inapplicable

Setting	Method	mIoU (SS)	mIoU (DA)	mAP	AP50	AP75
Full	MaskFormer [14]	57.1	–	–	–	–
	MaskFormer [14]	–	83.9	–	–	–
	Sparse R-CNN [91]	–	–	29.4	55.8	26.4
	Self-training [29]	61.8	84.4	30.1	56.6	27.6
	LV-Adapter (Ours)	**63.1**	**84.9**	**31.1**	**58.2**	**28.4**
Disjoint-balance	MaskFormer [14]	57.1	–	–	–	–
	MaskFormer [14]	–	78.1	–	–	–
	Sparse R-CNN [91]	–	–	18.6	37.8	15.6
	Self-training [29]	59.4	80.3	22.4	44.1	19.6
	LV-Adapter (Ours)	**61.8**	**80.6**	**24.6**	**47.4**	**21.9**
Disjoint-normal	MaskFormer [14]	57.1	–	–	–	–
	MaskFormer [14]	–	82.0	–	–	–
	Sparse R-CNN [91]	–	–	20.9	41.9	17.8
	Self-training [29]	60.3	83.1	24.9	48.1	22.2
	LV-Adapter (Ours)	**62.2**	**83.7**	**26.4**	**50.5**	**23.7**

Task Scheduling

Here we compare task scheduling methods on disjoint-normal settings. As shown in Table 9.3, three task sampling methods (i.e., Uniform sampler [62], Weighted sampler [62] and Round-robin [62]) perform better than Zeroing loss [106] by a large margin on segmentation-based tasks, but get worse in object detection. We hypothesize that training one task per step may lead to forgetting to some extent.

Table 9.3 Comparisons of popular task scheduling strategies and partial-label learning methods under the disjoint-normal setting

Methods	mAP	AP50	AP75	mIoU (SS)	mIoU (DA)	IoU (LD)
Zeroing loss [106]	31.1	54.3	30.2	55.7	88.0	22.2
Uniform sampler [62]	30.1	52.8	29.0	60.6	88.6	23.4
Weighted sampler [62]	29.3	51.9	28.7	58.5	**88.9**	**23.8**
Round-robin [62]	30.2	53.1	29.7	**61.0**	88.7	23.5
Pseudo labeling [29]	**32.6**	**54.6**	**32.3**	59.7	88.2	23.0

Task Balancing

We choose pseudo labeling as the baseline since task balancing methods are more suitable in settings with complete labels. Fixed [29] denotes fixed loss weights for all tasks during training. As shown in Table 9.4, Uncertainty performs better than Fixed on semantic segmentation and drivable area segmentation under the disjoint-normal settings, while performances of other approaches (i.e., GradNorm and MGDA) degrade significantly. Especially, GradNorm uses the last shared layer of weights to compute gradient norm, thus we adopt the last layer of P_5 in the neck. Interestingly, MGDA achieves the best result on lane detection, indicating that it suffers from the heavy negative transfer.

For efficiency and effectiveness, we choose pseudo labeling with fixed loss weights as our baseline, which achieves competitive performance compared with other complicated multi-task methods, to verify the effectiveness of GT-Prompt.

Table 9.4 Comparisons of popular task balancing strategies with pseudo labels under the disjoint-normal setting

Methods	mAP	AP50	AP75	mIoU (SS)	mIoU (DA)	IoU (LD)
Fixed [29]	**32.6**	**54.6**	**32.3**	59.7	88.2	23.0
Uncertainty [46]	32.2	54.1	31.5	**59.8**	**88.6**	23.8
GradNorm [13]	25.9	43.2	26.1	39.2	39.6	3.7
MGDA [89]	25.9	44.6	26.0	50.1	85.4	**25.2**

9.5.1.3 GT-Prompt

Main Results

As shown in Table 9.5, our GT-Prompt surpasses the baseline consistently on almost all metrics in all three settings. We conclude that GT-Prompt can learn task-invariant and task-specific knowledge during training, and further improve performance. We also adopt the trained GT-Prompt model as the pre-trained model and finetune on a single task to test the generalization of GT-Prompt. During finetuning, we only use the pre-trained model to initialize the backbone and remove pre-trained heads. Note that we do not use pseudo labels during finetuning. As shown in Table 9.5, GT-Prompt (FT) performs better than GT-Prompt on all segmentation-based tasks by a large margin in all settings, validating the excellent generalization capability of GT-Prompt.

Ablation Study

We conduct all ablation studies under the disjoint-balance setting for efficiency.

Module Components

We present detailed comparisons on each module to validate our GT-Prompt, as in Table 9.6. Equipped with G-Prompt, the model achieves better results on all tasks (#1 vs. #2), confirming that G-Prompt can learn general information which is useful for all tasks. When we introduce the task-specific fusion module to integrate T-Prompt and the visual representation, the performance is improved on both semantic segmentation and lane detection. After combining all components (#3), the model shows a superior performance overall. FT indicates further finetuning pre-trained GT-Prompt's backbone on single tasks, and this achieves the best results on all tasks.

G-Promt Block

We further investigate different designs for the G-Prompt block. For efficiency, we choose lightweight modules, i.e., MLP [39], SE [40], and depthwise convolution [15], to construct G-Prompt block. As shown in Table 9.7, depthwise convolution achieves superior performance. Note that these experiments were conducted without T-Prompt.

Task-Specific Fusion

We conduct an ablation study to verify the effectiveness of multi-scale window size in the task-specific fusion module as in Table 9.8. We can observe that when the task-specific fusion module is equipped with multi-scale windows, it can learn more effective multi-scale task-specific information.

Table 9.5 Comparison with single-task and multi-task learning baselines under different settings

Setting	Methods	mAP	AP50	AP75	mIoU (SS)	mIoU (DA)	IoU (LD)
Full	Sparse R-CNN [91]	36.5	61.5	36.1	–	–	–
	Semantic FPN [48]	–	–	–	59.8	–	–
	Semantic FPN [48]	–	–	–	–	89.1	–
	Semantic FPN [48]	–	–	–	–	–	25.9
	Pseudo labeling [29]	36.3	61.6	36.1	60.9	89.3	23.8
	GT-Prompt	**36.5**	**62.0**	**36.2**	61.7	89.3	23.9
	GT-Prompt (FT)	36.3	61.5	35.9	**63.5**	**89.4**	**26.0**
Disjoint-normal	Sparse R-CNN [91]	28.8	50.4	28.0	–	–	–
	Semantic FPN [48]	–	–	–	59.8	–	–
	Semantic FPN [48]	–	–	–	–	87.8	–
	Semantic FPN [48]	–	–	–	–	–	25.2
	Pseudo labeling [29]	32.6	54.6	32.3	59.7	88.2	23.0
	GT-Prompt	**32.7**	54.7	**32.3**	60.8	88.2	23.3
	GT-Prompt (FT)	31.8	**55.1**	31.4	**62.6**	**88.7**	**25.7**
Disjoint-balance	Sparse R-CNN [91]	28.1	49.2	26.7	–	–	–
	Semantic FPN [48]	–	–	–	59.8	–	–
	Semantic FPN [48]	–	–	–	–	85.5	–
	Semantic FPN [48]	–	–	–	–	–	23.7
	Pseudo labeling [29]	31.3	52.8	30.8	60.2	87.0	22.2
	GT-Prompt	**31.7**	53.6	**31.3**	60.8	87.0	22.4
	GT-Prompt (FT)	30.9	**53.7**	30.1	**62.6**	**87.6**	**24.4**

Table 9.6 Ablation study of our proposed GT-Prompt under the disjoint-balance setting. Fusion denotes the task-specific fusion module with task-specific prompts

#	G-Prompt	Fusion	mIoU (SS)	mIoU (DA)	IoU (LD)
1	✗	✗	60.2	87.0	22.2
2	✓	✗	60.3	87.2	22.3
3	✓	✓	60.8	87.0	22.4
FT	✓	✓	**62.6**	**87.6**	**24.4**

Table 9.7 Comparisons of multi-task models equipped with different designs of G-Prompt blocks

Design	mIoU (SS)	mIoU (DA)	IoU (LD)
MLP	59.8	86.7	21.9
SE	59.9	87.0	**22.3**
Depthwise conv	**60.3**	**87.2**	**22.3**

Table 9.8 Comparison of fixed and multi-scale window size in the task-specific fusion module

Window	mAP	mIoU (SS)	mIoU (DA)	IoU (LD)
Fixed	31.1	60.3	86.4	22.3
Multi-scale	**31.7**	**60.8**	**87.0**	**22.4**

Table 9.9 Comparison of fixed and trainable T-Prompt under the disjoint-balance setting

Trainable	mAP	mIoU (SS)	mIoU (DA)	IoU (LD)
✗	31.0	60.3	86.8	**22.4**
✓	**31.7**	**60.8**	**87.0**	**22.4**

Task-Specific Prompt

Since we obtain task-specific prompts from the pre-trained image encoder, there are two ways to treat them during training: fixed prompts or trainable ones. As shown in Table 9.9, trainable task prompts boost the performance by a margin, indicating that trainable prompts motivate the model to learn effective task-specific information based on training samples.

9.5.2 3D Perception

9.5.2.1 Fuller

We compare the Fuller with current state-of-the-art methods and report the result on nuScenes validation set in Table 9.10. We list each model's modality and group them by task setting. We adopt the competitive baseline (i.e., penultimate row) based on BEVFusion [75]. Given hardware capacity of V100 GPU, the voxel size is set to 0.1m for multi-task learning. For fair comparison, the single-task models Fuller-det and Fuller-seg using voxel size of 0.1m are set as the upper bounds for 3D detection and map segmentation.

As shown in Table 9.10, the performance of multi-task baseline drops by a large margin compared to the upper bounds. Particularly, the mIoU of segmentation task drops drastically from 62.3% to 44.0%, which discourages the multi-task applications in autonomous driving scenario. The degradation mainly comes from task conflict, which is resulted from the notorious negative transfer. In theory, each objective has

Table 9.10 Comparison with benchmark. The upper two sub-tables are single task results while the bottom one is multi-task result. 'L' and 'C' represent LiDAR and Camera, respectively. We treat single task result as our upper bound because multi-task will generally decrease the performance. Baseline means Fuller is naively trained where detection loss and segmentation loss are set to 1:1. '–' means inapplicable. † means the multi-task result in BEVFusion [75]. ‡ means re-implementation result in BEVFusion [75]. 'share' means multi-task heads share one BEV encoder to process the fused multimodal feature. 'sep' means task heads have separate encoders

	Modality	VoxelSize	LiDAR	Image	mAP (%)↑	NDS↑	mIoU (%)↑
3D detection							
BEVFormer [57]	C	–	–	ResNet101 [37]	41.6	51.7	–
CenterPoint [116]	L	0.075	VoxelNet	–	59.6	66.8	–
MVP‡ [117]	C+L	0.075	VoxelNet	DLA-34	66.1	70.0	–
TransFusion [2]	C+L	0.075	VoxelNet	DLA-34	67.5	71.3	–
BEVFusion [75]	C+L	0.075	VoxelNet	Swin-T	68.5	71.4	–
Fuller-det	C+L	0.075	VoxelNet	Swin-T	**67.6**	**71.3**	–
Fuller-det (upper bound)	C+L	0.1	VoxelNet	Swin-T	**62.1**	**66.6**	–
BEV map segmentation							
LSS‡ [81]	C	–	–	EfficientNet-B0	–	–	44.4
CenterPoint‡ [116]	L	0.1	VoxelNet	–	–	–	48.6
BEVFusion [75]	C+L	0.1	VoxelNet	Swin-T	–	–	62.7
Fuller-seg (upper bound)	C+L	0.1	VoxelNet	Swin-T	–	–	**62.3**
3D detection + BEV map segmentation							
BEVFusion† [75] (share)	C+L	0.1	VoxelNet	Swin-T	–	69.7	54.0
BEVFusion† [75] (sep)	C+L	0.1	VoxelNet	Swin-T	–	69.9	58.4
Baseline(share)	C+L	0.1	VoxelNet	Swin-T	59.1	65.0	44.0
Fuller(share)	C+L	0.1	VoxelNet	Swin-T	**60.5**	**65.3**	**58.4**

its own local minima given a parameter space. The conflict arises when jointly optimizing multiple objectives without regularization. On the other hand, the proposed model Fuller can bridge the gap between the single-task model and the multi-task variant, as it improves mIoU from 44.0% to 58.4% in map segmentation and increases the mAP from 59.1% to 60.5% in 3D detection.

9.6 Conclusion

In this chapter, we thoroughly investigate the multi-task learning in both 2D perception and 3D perception for autonomous driving.

9.6.1 2D Perception

9.6.1.1 LV-Adapter

We first reveal the unsuitability of state-of-the-art self-supervised models under the multi-task setting. To reduce the gap between the pre-training and fine-tuning stage, we propose a simple but yet highly efficient *pretrain-adapt-finetune* paradigm, which boosts the performances of most self-supervised pretrained models by a large margin. We further excavate the complementarity of vision-language pre-training in multi-task learning. We introduce the LV-Adapter [61], which incorporates language priors via learning task-specific prompts and excavating corresponding visual information. We further conduct extensive experiments to demonstrate the effectiveness of our proposed method.

9.6.1.2 GT-Prompt

We first provide an in-depth analysis of popular multi-task learning methods under the realistic scenarios of self-driving, which covers four common perception tasks, namely, object detection, semantic segmentation, drivable area segmentation, and lane detection. We find that existing methods cannot solve all tasks satisfactorily due to the negative transfer. To mitigate the negative transfer issue, we propose the general and task-specific prompt (GT-Prompt), which decouples task-invariant and task-specific knowledge. We further introduce multi-scale window attention to blend task-specific prompts and learn task-specific information effectively. Experimental results show that GT-Prompt can improve both single-task and multi-task baselines on the large-scale driving dataset BDD100K.

9.6.2 3D Perception

9.6.2.1 Fuller

We introduce the Fuller model, which unifies multi-modality multi-task learning for 3D perception tasks, and we point out the major problems such as modality bias and task conflict. To cope with the problems, we propose multi-level gradient calibration to guide the learning process of the model. Specifically, the inter-gradient calibration will balance the gradients w.r.t. downstream tasks on the last layer of the shared backbone. Before being separated into different branches, the magnitude of these gradients will be calibrated again within the intra-gradient layer. Through comprehensive experiments, we demonstrate the effectiveness of our Fuller model.

References

1. Antol S, Agrawal A, Lu J, Mitchell M, Batra D, Zitnick C.L, Parikh D (2015) VQA: visual question answering. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 2425–2433
2. Bai X, Hu Z, Zhu X, Huang Q, Chen Y, Fu H, Tai C.L (2022) Transfusion: robust lidar-camera fusion for 3d object detection with transformers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 1090–1099
3. Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler DM, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I, Amodei D (2020) Language models are few-shot learners. Advances in neural information processing systems (NeurIPS) 33:1877–1901
4. Brüggemann D, Kanakis M, Obukhov A, Georgoulis S, Van Gool L (2021) Exploring relational context for multi-task dense prediction. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 15869–15878
5. Caesar H, Bankiti V, Lang A.H, Vora S, Liong V.E, Xu Q, Krishnan A, Pan Y, Baldan G, Beijbom O (2020) Nuscenes: a multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 11621–11631
6. Carion N, Massa F, Synnaeve G, Usunier N, Kirillov A, Zagoruyko S (2020) End-to-end object detection with transformers. In: Proceedings of European conference on computer vision, pp 213–229
7. Caron M, Misra I, Mairal J, Goyal P, Bojanowski P, Joulin A (2020) Unsupervised learning of visual features by contrasting cluster assignments. Adv Neural Inf Process Syst (NeurIPS) 33:9912–9924
8. Chen R, Ai H, Shang C, Chen L, Zhuang Z (2019) Learning lightweight pedestrian detector with hierarchical knowledge distillation. In: 2019 IEEE international conference on image processing (ICIP). IEEE, pp 1645–1649
9. Chen T, Kornblith S, Norouzi M, Hinton G (2020) A simple framework for contrastive learning of visual representations. In: International conference on machine learning (ICML), pp 1597–1607
10. Chen X, Fan H, Girshick R, He K (2020) Improved baselines with momentum contrastive learning. arXiv:2003.04297
11. Chen X, Zhang T, Wang Y, Wang Y, Zhao H (2022) Futr3d: a unified sensor fusion framework for 3d detection. arXiv preprint arXiv:2203.10642

12. Chen Y, Zhao D, Lv L, Zhang Q (2018) Multi-task learning for dangerous object detection in autonomous driving. Inf Sci 432:559–571
13. Chen Z, Badrinarayanan V, Lee CY, Rabinovich A (2018) Gradnorm: gradient normalization for adaptive loss balancing in deep multitask networks. In: International conference on machine learning, pp 794–803. PMLR
14. Cheng B, Schwing A, Kirillov A (2021) Per-pixel classification is not all you need for semantic segmentation. Advances Neural Inf Process Syst (NeurIPS) 34
15. Chollet F (2017) Xception: deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1251–1258
16. Clark K, Luong MT, Khandelwal U, Manning CD, Le QV (2019) Bam! born-again multi-task networks for natural language understanding. arXiv preprint arXiv:1907.04829
17. Crawshaw M (2020) Multi-task learning with deep neural networks: a survey. arXiv preprint arXiv:2009.09796
18. Désidéri JA (2012) Multiple-gradient descent algorithm (mgda) for multiobjective optimization. Comptes Rendus Math 350(5–6):313–318
19. Devlin J, Chang MW, Lee K, Toutanova K (2019) Bert: pre-training of deep bidirectional transformers for language understanding. arXiv arXiv:1810.04805
20. Donahue J, Krähenbühl P, Darrell T (2017) Adversarial feature learning. arXiv arXiv:1605.09782
21. Donahue J, Simonyan K (2019) Large scale adversarial representation learning. In: NeurIPS
22. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, et al (2020) An image is worth 16x16 words: transformers for image recognition at scale. arXiv preprint arXiv:2010.11929
23. Eigen D, Fergus R (2015) Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp 2650–2658
24. Feng D, Zhou Y, Xu C, Tomizuka M, Zhan W (2021) A simple and efficient multi-task network for 3d object detection and road understanding. In: 2021 IEEE/RSJ international conference on intelligent robots and systems, pp 7067–7074
25. Fifty C, Amid E, Zhao Z, Yu T, Anil R, Finn C (2021) Efficiently identifying task groupings for multi-task learning. Adv Neural Inf Process Syst (NeurIPS) 34
26. Gao P, Geng S, Zhang R, Ma T, Fang R, Zhang Y, Li H, Qiao Y (2021) Clip-adapter: better vision-language models with feature adapters. arXiv preprint arXiv:2110.04544
27. Gao R, Oh TH, Grauman K, Torresani L (2020) Listen to look: action recognition by previewing audio. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 10457–10467
28. Gao Y, Ma J, Zhao M, Liu W, Yuille AL (2019) Nddr-cnn: layerwise feature fusing in multitask cnns by neural discriminative dimensionality reduction. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 3205–3214
29. Ghiasi G, Zoph B, Cubuk ED, Le QV, Lin TY (2021) Multi-task self-training for learning general representations. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 8856–8865
30. Girshick RB (2015) Fast r-cnn. In: 2015 IEEE international conference on computer vision (ICCV), pp 1440–1448
31. Goel S, Bansal H, Bhatia S, Rossi RA, Vinay V, Grover A (2022) Cyclip: cyclic contrastive language-image pretraining. arXiv preprint arXiv:2205.14459
32. Grill JB, Strub F, Altché F, Tallec C, Richemond P, Buchatskaya E, Doersch C, Avila Pires B, Guo Z, Gheshlaghi Azar M, Piot B, kavukcuoglu k, Munos R, Valko M, (2020) Bootstrap your own latent: a new approach to self-supervised learning. Adv Neural Inf Process Syst (NeurIPS) 33:21271–21284
33. Guo P, Lee CY, Ulbricht D (2020) Learning to branch for multi-task learning. In: International conference on machine learning, pp 3854–3863
34. He K, Fan H, Wu Y, Xie S, Girshick R (2020) Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 9729–9738

35. He K, Girshick RB, Dollár P (2019) Rethinking imagenet pre-training. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 4917–4926
36. He K, Gkioxari G, Dollar P, Girshick R (2017) Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp 2961–2969
37. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
38. He Y, Zheng S, Tay Y, Gupta J, Du Y, Aribandi V, Zhao Z, Li Y, Chen Z, Metzler D et al (2022) Hyperprompt: prompt-based task-conditioning of transformers. In: International conference on machine learning, pp 8678–8690. PMLR
39. Hendrycks D, Gimpel K (2016) Bridging nonlinearities and stochastic regularizers with gaussian error linear units
40. Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7132–7141
41. Ilievski I, Feng J (2017) Multimodal learning and reasoning for visual question answering. Adv Neural Inf Process Syst 30:551–562
42. Ishihara K, Kanervisto A, Miura J, Hautamaki V (2021) Multi-task learning with attention for end-to-end autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2902–2911
43. Ismail AA, Hasan M, Ishtiaq F (2020) Improving multimodal accuracy through modality pre-training and attention. arXiv preprint arXiv:2011.06102
44. Jia M, Tang L, Chen BC, Cardie C, Belongie S, Hariharan B, Lim SN (2022) Visual prompt tuning. arXiv preprint arXiv:2203.12119
45. Kazakos E, Nagrani A, Zisserman A, Damen D (2019) Epic-fusion: audio-visual temporal binding for egocentric action recognition. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 5492–5501
46. Kendall A, Gal Y, Cipolla R (2018) Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 7482–7491
47. Kiela D, Grave E, Joulin A, Mikolov T (2018) Efficient large-scale multi-modal classification. In: Proceedings of the AAAI conference on artificial intelligence, pp 5198–5204
48. Kirillov A, Girshick R, He K, Dollár P (2019) Panoptic feature pyramid networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 6399–6408
49. Kirillov A, He K, Girshick R, Rother C, Dollár P (2019) Panoptic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 9404–9413
50. Kokkinos I (2017) Ubernet: training a 'universal' convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 6129–6138
51. Kokkinos I (2017) Ubernet: training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 6129–6138
52. Lang AH, Vora S, Caesar H, Zhou L, Yang J, Beijbom O (2019) Pointpillars: fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 12697–12705
53. Lester B, Al-Rfou R, Constant N (2021) The power of scale for parameter-efficient prompt tuning. In: EMNLP
54. Li Q, Wang Y, Wang Y, Zhao H (2022) Hdmapnet: an online HD map construction and evaluation framework. In: 2022 international conference on robotics and automation, pp 4628–4634. https://doi.org/10.1109/ICRA46639.2022.9812383
55. Li XL, Liang P (2021) Prefix-tuning: optimizing continuous prompts for generation. arXiv preprint arXiv:2101.00190
56. Li Y, Ge Z, Yu G, Yang J, Wang Z, Shi Y, Sun J, Li Z (2022) Bevdepth: acquisition of reliable depth for multi-view 3d object detection. arXiv preprint arXiv:2206.10092

57. Li Z, Wang W, Li H, Xie E, Sima C, Lu T, Qiao Y, Dai J (2022) Bevformer: learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In: Proceedings of the European conference on computer vision, pp 1–18. https://doi.org/10.1007/978-3-031-20077-9_1
58. Liang M, Yang B, Chen Y, Hu R, Urtasun R (2019) Multi-task multi-sensor fusion for 3d object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 7345–7353
59. Liang T, Xie H, Yu K, Xia Z, Lin Z, Wang Y, Tang T, Wang B, Tang Z (2022) BEVFusion: a simple and robust LiDAR-camera fusion framework. arXiv preprint arXiv:2205.13790
60. Liang X, Niu M, Han J, Xu H, Xu C, Liang X (2023) Visual exemplar driven task-prompting for unified perception in autonomous driving. arXiv preprint arXiv:2303.01788
61. Liang X, Wu Y, Han J, Xu H, Xu C, Liang X (2022) Effective adaptation in multi-task co-training for unified autonomous driving. arXiv preprint arXiv:2209.08953
62. Likhosherstov V, Arnab A, Choromanski K, Lucic M, Tay Y, Weller A, Dehghani M (2021) Polyvit: co-training vision transformers on images, videos and audio. arXiv preprint arXiv:2111.12993
63. Likhosherstov V, Arnab A, Choromanski K, Lucic M, Tay Y, Weller A, Dehghani M (2021) Polyvit: co-training vision transformers on images, videos and audio. arXiv preprint arXiv:2111.12993
64. Lin TY, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2117–2125
65. Lin X, Zhen HL, Li Z, Zhang QF, Kwong S (2019) Pareto multi-task learning. Adv Neural Inf Process Syst 32
66. Liu L, Li Y, Kuang Z, Xue J, Chen Y, Yang W, Liao Q, Zhang W (2021) Towards impartial multi-task learning. In: International conference on learning representations
67. Liu P, Yuan W, Fu J, Jiang Z, Hayashi H, Neubig G (2021) Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. arXiv preprint arXiv:2107.13586
68. Liu P, Yuan W, Fu J, Jiang Z, Hayashi H, Neubig G (2021) Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. arXiv preprint arXiv:2107.13586
69. Liu S, Johns E, Davison AJ (2019) End-to-end multi-task learning with attention. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 1871–1880
70. Liu X, Ji K, Fu Y, Du Z, Yang Z, Tang J (2021) P-tuning v2: prompt tuning can be comparable to fine-tuning universally across scales and tasks. arXiv arXiv:2110.07602
71. Liu X, Zheng Y, Du Z, Ding M, Qian Y, Yang Z, Tang J (2021) Gpt understands, too. arXiv preprint arXiv:2103.10385
72. Liu Y, Wang T, Zhang X, Sun J (2022) Petr: position embedding transformation for multi-view 3d object detection. arXiv preprint arXiv:2203.05625
73. Liu Y, Yan J, Jia F, Li S, Gao Q, Wang T, Zhang X, Sun J (2022) Petrv2: a unified framework for 3d perception from multi-camera images. arXiv preprint arXiv:2206.01256
74. Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, Guo B (2021) Swin transformer: hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 10012–10022
75. Liu Z, Tang H, Amini A, Yang X, Mao H, Rus D, Han S (2022) Bevfusion: multi-task multi-sensor fusion with unified bird's-eye view representation. arXiv preprint arXiv:2205.13542
76. Misra I, Shrivastava A, Gupta A, Hebert M (2016) Cross-stitch networks for multi-task learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3994–4003
77. Nie X, Ni B, Chang J, Meng G, Huo C, Zhang Z, Xiang S, Tian Q, Pan C (2022) Pro-tuning: unified prompt tuning for vision tasks. arXiv preprint arXiv:2207.14381
78. van den Oord A, Li Y, Vinyals O (2019) Representation learning with contrastive predictive coding. arXiv:1807.03748

79. Owens A, Efros AA (2018) Audio-visual scene analysis with self-supervised multisensory features. In: Proceedings of the European conference on computer vision, pp 631–648
80. Peng X, Wei Y, Deng A, Wang D, Hu D (2022) Balanced multimodal learning via on-the-fly gradient modulation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 8238–8247
81. Philion J, Fidler S (2020) Lift, splat, shoot: encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In: Proceedings of European conference on computer vision, pp 194–210
82. Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J, Krueger G, Sutskever I (2021) Learning transferable visual models from natural language supervision. In: International conference on machine learning (ICML), pp 8748–8763
83. Raghu M, Unterthiner T, Kornblith S, Zhang C, Dosovitskiy A (2021) Do vision transformers see like convolutional neural networks? arXiv arXiv:2108.08810
84. Rao Y, Zhao W, Chen G, Tang Y, Zhu Z, Huang G, Zhou J, Lu J (2022) Denseclip: language-guided dense prediction with context-aware prompting. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 18082–18091
85. Ren P, Li C, Wang G, Xiao Y, Chang QDXLX (2022) Beyond fixation: dynamic window visual transformer. arXiv preprint arXiv:2203.12856
86. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: towards real-time object detection with region proposal networks. Adv Neural Inf Process Syst (NeurIPS) 28
87. Ruder S, Bingel J, Augenstein I, Søgaard A (2019) Latent multi-task architecture learning. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 4822–4829
88. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein MS, Berg AC, Fei-Fei L (2015) Imagenet large scale visual recognition challenge. Int J Comput Vis (IJCV) 115(3):211–252
89. Sener O, Koltun V (2018) Multi-task learning as multi-objective optimization. Adv Neural Inf Process Syst 31
90. Standley TS, Zamir AR, Chen D, Guibas LJ, Malik J, Savarese S (2020) Which tasks should be learned together in multi-task learning? In: International conference on machine learning (ICML), pp 9120–9132
91. Sun P, Zhang R, Jiang Y, Kong T, Xu C, Zhan W, Tomizuka M, Li L, Yuan Z, Wang C, Luo P (2021) Sparse r-cnn: end-to-end object detection with learnable proposals. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 14454–14463
92. Sun X, Panda R, Feris RS (2020) Adashare: learning what to share for efficient deep multi-task learning. Adv Neural Inf Process Syst (NeurIPS) 33:8728–8740
93. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow IJ, Fergus R (2013) Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199
94. Teichmann M, Weber M, Zoellner M, Cipolla R, Urtasun R (2018) Multinet: real-time joint semantic reasoning for autonomous driving. In: IEEE intelligent vehicles symposium (IV), pp 1013–1020
95. Vandenhende S, Georgoulis S, Gool LV (2020) Mti-net: multi-scale task interaction networks for multi-task learning. In: European conference on computer vision. Springer, pp 527–543
96. Vandenhende S, Georgoulis S, Van Gansbeke W, Proesmans M, Dai D, Van Gool L (2021) Multi-task learning for dense prediction tasks: a survey. IEEE Trans Pattern Anal Mach Intell
97. Vaswani A, Shazeer NM, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. Adv Neural Inf Process Syst (NeurIPS) 30
98. Vora S, Lang AH, Helou B, Beijbom O (2020) Pointpainting: sequential fusion for 3d object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4604–4612
99. Wang A, Singh A, Michael J, Hill F, Levy O, Bowman SR (2018) Glue: a multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461

100. Wang W, Tran D, Feiszli M (2020) What makes training multi-modal classification networks hard? In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 12695–12705
101. Wang X, Ke B, Li X, Liu F, Zhang M, Liang X, Xiao Q (2022) Modality-balanced embedding for video retrieval. In: Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval, pp 2578–2582
102. Wang X, Zhang R, Shen C, Kong T, Li L (2021) Dense contrastive learning for self-supervised visual pre-training. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 3023–3032
103. Wang Z, Ren W, Qiu Q (2018) Lanenet: real-time lane detection networks for autonomous driving. arXiv preprint arXiv:1807.01726
104. Wang Z, Zhang Z, Ebrahimi S, Sun R, Zhang H, Lee CY, Ren X, Su G, Perot V, Dy J et al (2022) Dualprompt: complementary prompting for rehearsal-free continual learning. arXiv preprint arXiv:2204.04799
105. Wu D, Liao M, Zhang W, Wang X (2021) Yolop: you only look once for panoptic driving perception. arXiv preprint arXiv:2108.11250
106. Xiao T, Liu Y, Zhou B, Jiang Y, Sun J (2018) Unified perceptual parsing for scene understanding. In: European conference on computer vision (ECCV), pp 418–434
107. Xiao T, Liu Y, Zhou B, Jiang Y, Sun J (2018) Unified perceptual parsing for scene understanding. In: Proceedings of the European conference on computer vision (ECCV), pp 418–434
108. Xie E, Ding J, Wang W, Zhan X, Xu H, Li Z, Luo P (2021) Detco: unsupervised contrastive learning for object detection. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 8392–8401
109. Xie E, Yu Z, Zhou D, Philion J, Anandkumar A, Fidler S, Luo P, Alvarez JM (2022) M^{2}bev: multi-camera joint 3d detection and segmentation with unified birds-eye view representation. arXiv preprint arXiv:2204.05088
110. Xie Z, Lin Y, Zhang Z, Cao Y, Lin S, Hu H (2021) Propagate yourself: exploring pixel-level consistency for unsupervised visual representation learning. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 16679–16688
111. Xu D, Ouyang W, Wang X, Sebe N (2018) Pad-net: multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 675–684
112. Xu Y, Li X, Yuan H, Yang Y, Zhang J, Tong Y, Zhang L, Tao D (2022) Multi-task learning with multi-query transformer for dense prediction. arXiv preprint arXiv:2205.14354
113. Yang Z, Zhang Y, Yu J, Cai J, Luo J (2018) End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perceptions. In: 24th international conference on pattern recognition (ICPR), pp 2289–2294
114. Yao Y, Zhang A, Zhang Z, Liu Z, Chua TS, Sun M (2021) Cpt: colorful prompt tuning for pre-trained vision-language models. arXiv preprint arXiv:2109.11797
115. Ye D, Zhou Z, Chen W, Xie Y, Wang Y, Wang P, Foroosh H (2022) Lidarmultinet: towards a unified multi-task network for lidar perception. arXiv preprint arXiv:2209.09385
116. Yin T, Zhou X, Krahenbuhl P (2021) Center-based 3d object detection and tracking. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 11784–11793
117. Yin T, Zhou X, Krähenbühl P (2021) Multimodal virtual point 3d detection. Adv Neural Inf Process Syst 34:16494–16507
118. Yu F, Chen H, Wang X, Xian W, Chen Y, Liu F, Madhavan V, Darrell T (2020) Bdd100k: a diverse driving dataset for heterogeneous multitask learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2636–2645
119. Zhang Y, Zhu Z, Zheng W, Huang J, Huang G, Zhou J, Lu J (2022) Beverse: unified perception and prediction in birds-eye-view for vision-centric autonomous driving. arXiv preprint arXiv:2205.09743. https://doi.org/10.48550/arXiv.2205.09743
120. Zhang Z, Cui Z, Xu C, Jie Z, Li X, Yang J (2018) Joint task-recursive learning for semantic segmentation and depth estimation. In: Proceedings of the European conference on computer vision (ECCV), pp 235–251

121. Zhong Z, Friedman D, Chen D (2021) Factual probing is [mask]: learning versus learning to recall. arXiv preprint arXiv:2104.05240
122. Zhou C, Loy CC, Dai B (2021) Denseclip: extract free dense labels from clip. arXiv arXiv:2112.01071
123. Zhou K, Yang J, Loy CC, Liu Z (2021) Learning to prompt for vision-language models. arXiv preprint arXiv:2109.01134
124. Zhou Y, Tuzel O (2018) Voxelnet: end-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4490–4499

Chapter 10
Bird's Eye View Perception for Autonomous Driving

Jiayuan Du, Shuai Su, Rui Fan, and Qijun Chen

Abstract Bird's eye view (BEV) perception refers to transforming a perspective view into a bird's eye view and performing perception tasks such as 3D detection, map segmentation, and motion prediction. Due to its inherent advantages in representing 3D space, fusing multi-modal data, facilitating decision making, and aiding in path planning, BEV perception has garnered significant attention from both academia and industry. In this chapter, we present an overview of BEV perception, covering its definition, categorization, benefits, and mathematical foundations. We then provide a comprehensive review of the current state-of-the-art research, datasets, evaluation metrics, and industrial applications. In the end, we outline several existing challenges and present our own conclusions regarding BEV perception.

10.1 Introduction

Bird's eye view (BEV) perception involves learning representations in the bird's eye view space and performing perception tasks, such as 3D detection [1], map segmentation [2], and motion prediction [3]. With the sensor settings of autonomous vehicles becoming increasingly diverse and complex, there is a great demand for a unified representation. Thanks to its inherent advantages in 3D space representation,

J. Du · S. Su · R. Fan · Q. Chen (✉)
The Robotics and Artificial Intelligence Laboratory (RAIL), The Department of Control Science and Engineering, Frontiers Science Center for Intelligent Autonomous Systems, and State Key Laboratory of Intelligent Autonomous Systems, Tongji University, Shanghai 201804, People's Republic of China
e-mail: qjchen@tongji.edu.cn

J. Du
e-mail: dujiayuan@tongji.edu.cn

S. Su
e-mail: sushuai@tongji.edu.cn

R. Fan
e-mail: rfan@tongji.edu.cn

R. Fan et al. (eds.), *Autonomous Driving Perception*, Advances in Computer Vision and Pattern Recognition, https://doi.org/10.1007/978-981-99-4287-9_10

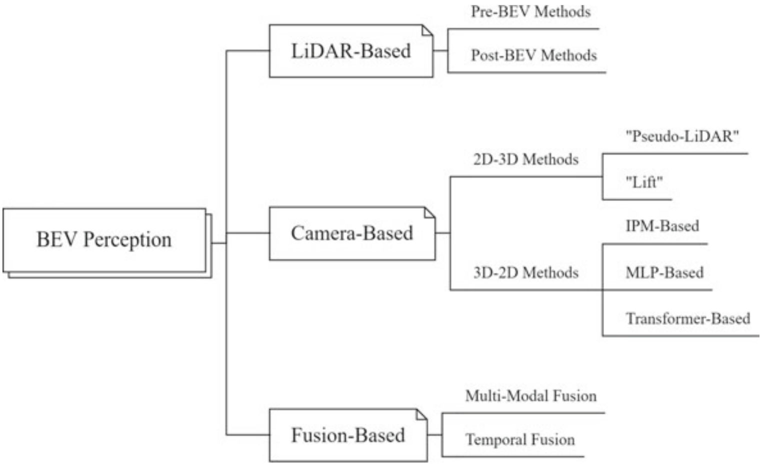


Fig. 10.1 Taxonomy of BEV perception methods

multi-modal fusion, decision making, and path planning, BEV perception has been attracting more and more attention from both academia and industry.

According to the different modalities of input data, a taxonomy of BEV perception methods can be yielded as illustrated in Fig. 10.1: LiDAR-based methods, camera-based methods, and fusion-based methods. Conventional LiDAR-based methods with their BEV-like point clouds have shown great success in 3D detection and semantic segmentation tasks [4]. However, perception using only perspective view or multi-view (vision-centric methods) remains a significant challenge.

In modern autonomous driving, decision-making and motion planning modules depend on multiple perception and prediction modules to gather sufficient environmental information. The perception task not only detects dynamic objects but also identifies static elements such as road boundaries, crosswalks, lane lines, and road signs. Meanwhile, the prediction task requires the system to deduce the motion trend of other dynamic objects, providing a basis for decision-making and path planning to avoid collisions.

Currently, in the industry, research on perception and prediction algorithms based solely on vision typically only focuses on a single sub-problem in the overall process. For example, researchers may focus on 3D object detection, map segmentation, object tracking, or motion prediction, and then fuse the perception results of different networks through pre-fusion or post-fusion methods. While this approach enables problem decomposition and facilitates independent academic research, it results in multiple submodules stacked in a linear structure when building the overall system. However, this serial architecture has several significant drawbacks:

1. The errors of upstream modules are continuously transmitted downstream, which can significantly affect the performance of downstream tasks when sub-problems

are studied independently. In these cases, the ground truth is typically used as input, which can result in cumulative errors impacting downstream performance.

2. In a serial architecture, repeated operations such as feature extraction and dimension conversion in Convolution Neural Networks (CNNs) can lead to redundant calculations, which are not conducive to improving the overall efficiency of the system.
3. The temporal information cannot be fully utilized. On the one hand, the temporal information can be leveraged as a supplement to the spatial information to better detect the blocked object at the current moment and provide more reference information for the location of the object. On the other hand, temporal information can help judge the motion state of the object. In the absence of temporal information, the method based on pure vision can hardly effectively judge the motion speed of the object.

Different from traditional serial architecture, the BEV scheme utilizes multiple sensors (such as camera, LiDAR, RADAR, IMU, GPS) and convert multi-modal information to a unified BEV space for various downstream perception tasks. Such a scheme can provide a unified representation space for autonomous driving perception [5] and can accomplish multiple perception tasks in parallel. Specifically, the advantages of BEV perception are reflected in the following aspects:

1. Intuitive and friendly for subsequent modules. BEV perception provides a holistic view of the surrounding environment, enabling the system to detect obstacles and road elements that might not be visible from the vehicle's current position, which is beneficial to downstream prediction and planning modules.
2. Fusion-friendly.

   a. BEV perception provides a consistent coordinate system for all sensors, simplifying the fusion of data from multiple sources and enabling better object association and tracking.
   b. Temporal fusion is easier to realize. In the BEV space, temporal information can be easily fused via ego-motion of the autonomous vehicle.

3. Easier end-to-end optimization. In traditional perception tasks, recognition, tracking, and prediction operate more like a "serial system", where errors in the upstream of the system are propagated downstream, resulting in error accumulation. However, in the BEV space, perception and prediction occur in a unified space, allowing for direct end-to-end optimization through neural networks and generating "parallel" results, thereby avoiding error accumulation. Additionally, this approach can significantly reduce the impact of algorithm logic, enabling the perceptual network to learn from data in a self-driven manner and achieve better functional iterations.

This chapter is organized as follows: Sect. 10.1 is an overall introduction of BEV perception, including conception, taxonomy and advantages. Section 10.2 introduces mathematical fundamentals of inverse perspective mapping, which is of vital importance in BEV perception. Sections 10.3, 10.4 and 10.5 introduces LiDAR-based, camera-based and fusion-based methodologies of BEV perception, respectively. Section 10.8 shows nowadays industrial applications of BEV perception. In Sects. 10.9 and 10.10 we discuss the existing challenges and Future Development of BEV perception.

10.2 BEV Fundamentals

Generally, cameras mounted on a vehicle suffer from a significant perspective effect [6–8], as illustrated in Fig. 10.2a. This perspective effect can result in the driver's inability to accurately perceive distance, making advanced image processing or analysis challenging. As a result, it is necessary to generate a bird's eye view using perspective transformation., as shown in Fig. 10.2 [9].

Perspective mapping can transform points in 3D space to the image space, while the inverse problem of projecting image pixels back to 3D space is considered an ill-posed problem. To address this challenge, the pioneer work of Inverse Perspective Mapping (IPM) [10] introduced a constraint that the inversely mapped points should lie on the ground plane. This constraint allows the mapping to be described via a homography matrix, enabling the solution of the ill-posed problem. The geometry of perspective mapping and inverse transformation is shown in Fig. 10.3. Mathematically, IPM is a linear mapping in homogeneous coordinates.

10.2.1 Perspective Mapping

For an ideal pinhole camera model, we state two definitions for camera coordinate system $\boldsymbol{C} = \{\mathbf{x_C}, \mathbf{y_C}, \mathbf{z_C}\}$ and an image plane $\boldsymbol{I} = \{\mathbf{u}, \mathbf{v}\}$ at distance f (focal length) from the origin parallel to $\mathbf{x_C}$ and $\mathbf{y_C}$. A perspective mapping is described by $\mathscr{P}_B : \mathbb{R}^3 \mapsto \mathbb{R}^2$:

$$\mathbf{p} \mapsto \mathbf{p}' = \begin{bmatrix} u \\ v \end{bmatrix} = \frac{-f}{(\mathbf{p} \cdot \mathbf{z_C})} \cdot \begin{bmatrix} (\mathbf{p} \cdot \mathbf{x_C}) \\ (\mathbf{p} \cdot \mathbf{y_C}) \end{bmatrix}, \tag{10.1}$$

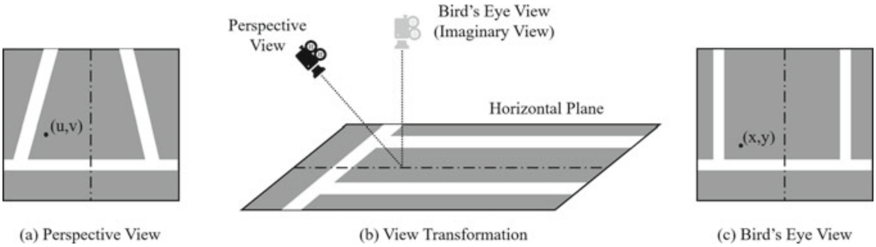


Fig. 10.2 Illustration of perspective transformation in a parking lot scene [9]

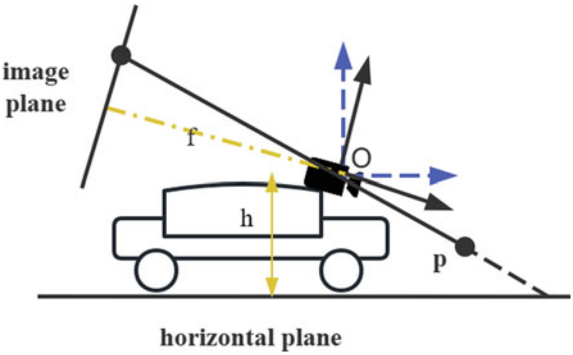


Fig. 10.3 Geometry of inverse perspective mapping [10]. Nodal point O is the center of projection and the common origin of the camera frame and the world frame. Solid black arrow $\mathbf{z_C}$, $\mathbf{y_C}$ axes of the camera coordinate system, and the Z-axis is in the same direction as the camera. h is the height of the nodal point O and f is the focal length of the camera. Dotted blue arrow $\mathbf{y_W}$, $\mathbf{z_W}$ axes of the world coordinate system, the Y-axis is in the same direction as the vehicle. The horizontal axes $\mathbf{x_C}$ and $\mathbf{x_W}$ are perpendicular to the paper plane. $\mathbf{p}$ is a point in 3D space; $\mathbf{p'_I}$, $\mathbf{p'_H}$ denote its projections into the image and the horizontal plane; $\tilde{\mathbf{p}}$ means homogeneous representation of $\mathbf{p'}$

where $(\cdot)$ denotes the inner product. All 3D points on the ray projected to $p' = [u, v]^\top$ is given by $\tilde{\mathbf{p}}$:

$$\mathbf{p'} \mapsto \tilde{\mathbf{p}} = \begin{bmatrix} \lambda u \\ \lambda v \\ -\lambda f \end{bmatrix}, \quad for \quad \lambda \in \mathbb{R}, \tag{10.2}$$

where $\tilde{\mathbf{p}}$ is the homogeneous representation of $\mathbf{p'}$ in the coordinate frame $\boldsymbol{C}$. Substitute (10.2) into (10.1), we have $\mathscr{P}(\tilde{\mathbf{p}}) = \mathbf{p'}$ for all $\lambda \neq 0$.

10.2.2 Inverse Perspective Mapping

Denote a world coordinate system as $\boldsymbol{W} = \{\mathbf{x_W}, \mathbf{y_W}, \mathbf{z_W}\}$. The horizontal plane is spanned by $\mathbf{x_W}$ and $\mathbf{y_W}$ and the upward direction ia pointed by $\mathbf{z_W}$. Suppose there is a center of projection O at height h from the horizontal plane and its focal length is f from the image plane. By assuming camera frame and world frame share the common origin O, then the coordinate transformation from the camera to the world is described by an orthogonal matrix $\boldsymbol{Q}$. In (10.3), $\boldsymbol{Q}$ is represented by the column vectors $\mathbf{x'}$, $\mathbf{y'}$ and $\mathbf{z'}$, the matrix $\boldsymbol{Q}$ is also known as the extrinsic matrix.

Under the assumption that the back-projected points fall on the ground, IPM is actually finding the correspondence between a point $\mathbf{p'_I}$ in the image plane and a point $\mathbf{p'_H}$ in the horizontal plane. Here, the horizontal ground plane is spanned by $\mathbf{x_W}$, $\mathbf{y_W}$ in the world frame. In homogeneous coordinates, this is a linear mapping of a point $\tilde{\mathbf{p_I}}$ to a point $\tilde{\mathbf{p_H}}$, characterized by the orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{3\times3}$:

$$\begin{gathered}\tilde{\mathscr{Q}}:\mathbb{R}^3 \mapsto \mathbb{R}^3; \quad \tilde{\mathbf{p}_\mathbf{H}} = Q \cdot \tilde{\mathbf{p}_\mathbf{I}} \\ \begin{bmatrix} \mu X \\ \mu Y \\ -\mu h \end{bmatrix} = \begin{bmatrix} x'_1 \; y'_1 \; z'_1 \\ x'_2 \; y'_2 \; z'_2 \\ x'_3 \; y'_3 \; z'_3 \end{bmatrix} \cdot \begin{bmatrix} \lambda u \\ \lambda v \\ -\lambda f \end{bmatrix}. \end{gathered} \tag{10.3}$$

Here, the expression of camera frame axes $\mathbf{x_C}$, $\mathbf{y_C}$, $\mathbf{z_C}$ in the world frame is denoted by the matrix $\boldsymbol{Q}$, which is composed out of x'_1, ..., z'_3. Then it projects $\tilde{\mathbf{p}_\mathbf{H}}$ onto the horizontal plane using (10.1) and yields the inverse perspective mapping $\mathscr{Q}$:

$$\begin{gathered}\mathscr{Q}:\mathbb{R}^2 \mapsto \mathbb{R}^2; \quad \mathbf{p}'_\mathbf{H} = \mathscr{Q}\,(\mathbf{p_I}) \\ \begin{bmatrix} X \\ Y \end{bmatrix} = \frac{-h}{x'_3 u + y'_3 v - z'_3 f} \cdot \begin{bmatrix} x'_1 u + y'_1 v - z'_1 f \\ x'_2 u + y'_2 v - z'_2 f \end{bmatrix}. \end{gathered} \tag{10.4}$$

10.3 LiDAR-Based BEV Perception

LiDAR sensors provide depth information, which can lead to better object recognition performance compared to visual-based sensors. However, LiDAR lacks semantic information such as texture or color, and the point clouds may be sparse in distant areas where information is missing. The general pipeline of LiDAR-based BEV perception is illustrated in Fig. 10.4. It takes point cloud as input and performs feature extraction and view transformation to construct BEV feature maps. Common detection heads are used to generate 3D prediction results. LiDAR-based methods can be classified into pre-BEV methods and post-BEV methods based on the order of feature extraction and BEV transformation, as shown in Fig. 10.4.

10.3.1 Pre-BEV Methods

Pre-BEV methods extract features before the BEV transformation using either point or voxel representations. Point-based methods process the raw LiDAR point cloud, which has significant consumption of computing and storage resources. In contrast, voxel-based methods voxelize the point cloud into discrete grids by discretizing the continuous 3D coordinate, providing a more efficient representation. To extract

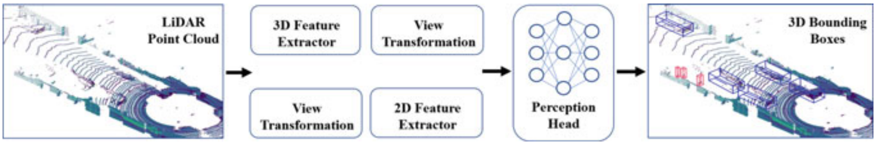


Fig. 10.4 General pipeline of LiDAR-based BEV perception [11]

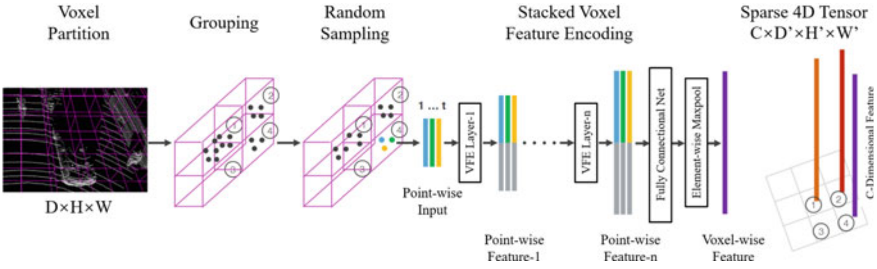


Fig. 10.5 Feature extraction of VoxelNet architecture [14]

point cloud features from the voxel, 3D convolution or 3D sparse convolution [12, 13] are commonly utilized. The majority of cutting-edge techniques typically undertake feature extraction using 3D sparse convolution. The height axis is then densified and compressed to form at the 3D voxel characteristics as a 2D tensor in BEV space.

As shown in Fig. 10.5. VoxelNet [14] firstly partitions the 3D space into voxels, generates voxel-wise features by maxpooling of all point-wise features, and represents the space as a sparse 4D tensor. Then, a 3D convolution is applied to aggregate spatial context in the tensor. Finally, it transforms the tensor into BEV implicitly, and generates the 3D bounding boxes with a region proposal network (RPN). SECOND [15] introduces sparse convolution in the processing of voxel representation, which greatly accelerate the training and inference.

Comparing with anchor-based bounding box, center-based representation has several advantages:

- Points have no intrinsic direction. Therefore, the search space of the target detector is reduced, which allowing backbone to learn the rotation invariance of objects and their variance with respect to rotation.
- It is also helpful to simplify downstream tasks such as tracking.A point object's trajectory is its path across space and time. The network can then integrate the continuous frames and forecast the relative offset between them with ease.
- Point-based feature extraction facilitates researchers to design a faster and more effective two-stage refinement module.

As the representative work, CenterPoint [16] becomes a baseline method for 3D detection due to its excellent design of two-stage center-based detector (Fig. 10.6).

To learn more discriminative features from LiDAR point cloud, PV-RCNN [17] combines voxel and point branches to generate 3D proposals and refinement respectively. To help the backbone network learn structure-aware features, SA-SSD [18] designs an auxiliary network that combines the foreground segmentation and center estimation tasks. Voxel Region of Interest (RoI) Pooling is adopted by Voxel R-CNN [19] to aggregate construction data from voxel-wise features, and it achieves comparable accuracy to point-based approaches. With the concept of dynamic graph, Object DGCNN [20] remodels the detection problem as information transmission process in the BEV space. To aggregate large 3D context, VoTr [21] introduces an attention

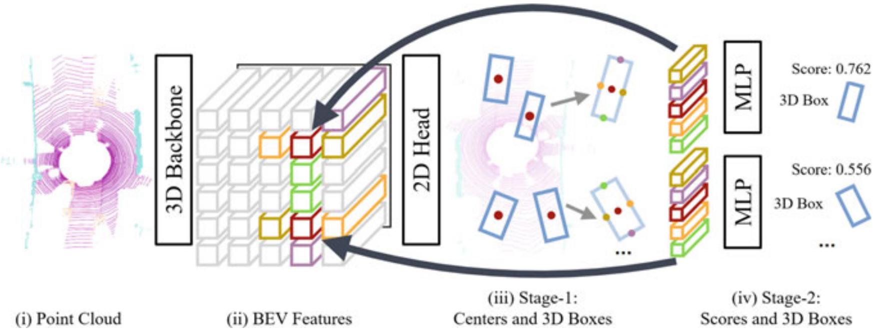


Fig. 10.6 Overview of CenterPoint framework [16]

mechanism on numerous voxels. SST [22] follows the idea of shifting windows in Swin Transformer [23], and divides the voxelized point cloud space into windows. It applies sparse regional attention and window shifting to avoid information loss by downsampling. AFDetV2 [24] introduces a keypoint supervision as auxiliary task and adopts a multi-task head to build a single-stage anchor-free network.

10.3.2 Post-BEV Methods

Another technical route of LiDAR-based BEV perception is to transform the LiDAR point cloud into BEV space first, and then carry out 2D feature extraction.

In order to fuse the front image with LiDAR point cloud and its front view, MV3D [25] first transforms LiDAR point cloud into BEV space (Fig. 10.7). It generates 3D object proposals in BEV space then project them back to three views. For each view,

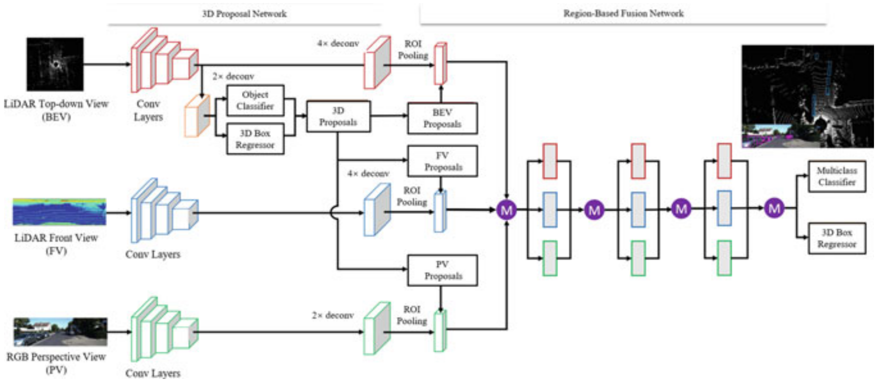


Fig. 10.7 Multi-view 3D object detection network (MV3D) [25]

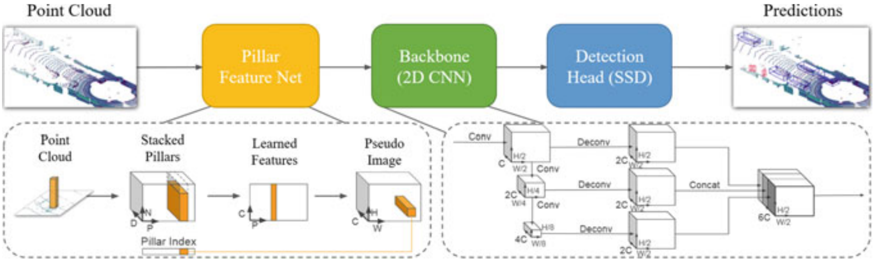


Fig. 10.8 PointPillar overview [32]

ROI pooling is used to extract region-wise features and these features are fused by A deep fusion network. The fused features are then used to predict 3D bounding boxes and object class jointly. The BEV representation of LiDAR point cloud is widely adopted in other works [26–31] as well. The term "pillar" refers to a unique kind of voxel with limitless height, which is first introduced in PointPillars [32]. To learn a representation of points in pillars, PointPillars utilizes a condense version of the PointNet [33]. Following that, standard 2D convolutional networks and detection heads are applied to process the encoded features. Although the PointPillar does not perform as well as other state-of-the-art (SOTA) methods, it and its variants are highly efficient, making them appropriate for industrial applications (Fig. 10.8).

10.4 Camera-Based BEV Perception

Comparing with LiDAR point clouds, 2D images do not naturally preserve accurate range information. Hence the core issue of camera-based BEV perception is to learn the depth from images explicitly or implicitly. According to the methods for transformation from a perspective view (PV) into a BEV, the camera-based BEV perception can be classified into two categories: 2D-3D methods and 3D-2D methods. The former "lifts" 2D features to 3D space via reconstructing depth information from 2D features, and the latter utilizes 3D-2D projection mapping to encode 2D features with 3D information (Fig. 10.9).

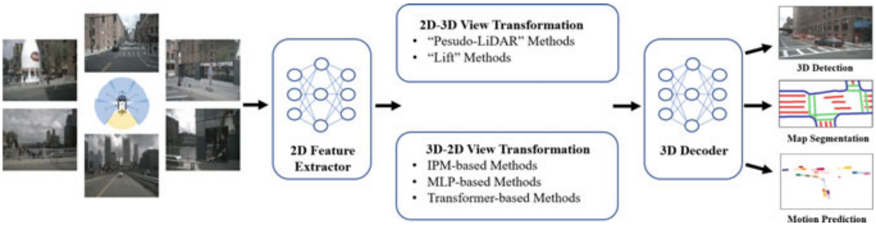


Fig. 10.9 General pipeline of camera-based BEV perception

10.4.1 2D-3D Methods

Since depth information is inaccessible for monocular cameras, 2D-3D methods construct the view transformation by predicting the depth. Currently, there are two sets of mainstream methods. One set is the "pseudo-LiDAR" method, which predicts a dense depth map, and is represented by the Pseudo-LiDAR family [34, 35]. The other set is the "lift" method, which predicts depth distribution and is represented by LSS (Lift, Splat, Shoot) [36] and its variants.

10.4.1.1 "Pseudo-LiDAR" Methods

As the name implies, Pseudo-LiDAR [34] estimates pixel-wise depth maps from stereo or monocular images and converts depth maps into pseudo-LiDAR point clouds, which can be direct input for 3D detectors designed for LiDARs. The pipeline of Pseudo-LiDAR is shown in Fig. 10.10, which is a straightforward way that can easily integrates mature experience of SOTA monocular depth estimation and LiDAR-based 3D detection methods. The accuracy of monocular depth estimation network is insufficient, while stereo network can provide better depth estimation and already has excellent performance in perception for autonomous driving. Therefore, Pseudo-LiDAR++ [35] improves the accuracy of depth estimation of faraway objects by adopting the stereo network. In AM3D [37], the pseudo-LiDAR point clouds is enhanced by corresponding RGB features. PatchNet [38] owes the effectiveness of the "pseudo-LiDAR" method to the coordinate system transformation rather than the data representation itself. However, the "Pseudo-LiDAR" methods have several problems. The performance of detection is heavily rely on the accuracy of depth estimation [39]. And the pixel-wise ground truth in outside scenes are difficult to obtain. Moreover, such methods suffer from data leakage and generalization problems. Researches show that the data leakage of KITTI depth benchmark causes the performance gap between validation and testing sets [34, 40]. To allow the pipeline can be trained in an end-to-end manner, E2E Pseudo-LiDAR [41], as illustrated in Fig. 10.11, introduces a Change-of-Representation module. The module improves generalization performance by alleviating the gradient cut-off between 3D object detection stage and depth estimation stage.

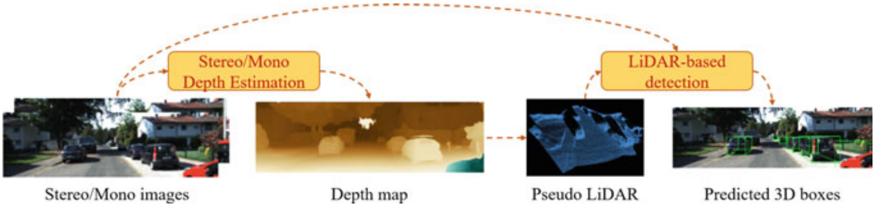


Fig. 10.10 Pipeline of Pseudo-LiDAR [34]

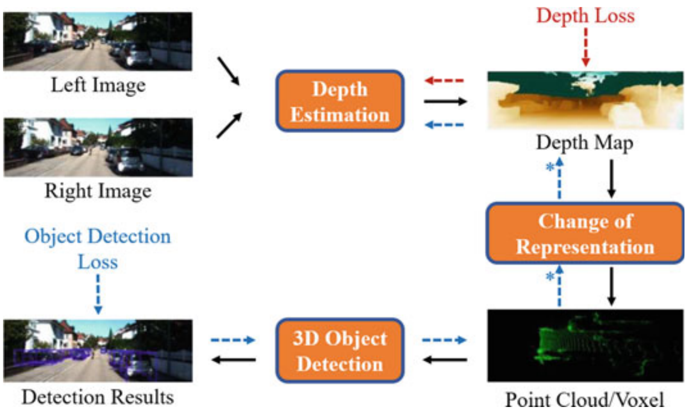


Fig. 10.11 The architecture of E2E Pseudo-LiDAR [41]

10.4.1.2 "Lift" Methods

Rather than predicting dense depth map, "Lift" paradigm predicts the depth distribution instead. LSS [36] is representative of this approach (Fig. 10.12). For each pixel, it predicts a context vector $\mathbf{c} \in \mathbb{R}^C$ and a categorical distribution over depth $\alpha \in \Delta^{D-1}$, and the corresponding 3D feature is determined by their outer product $\alpha \cdot \mathbf{c}$. In other words, the 2D features are "lifted" via depth distribution to 3D space, then the downstream perception task can be performed following SOTA LiDAR-based approaches. However, there are several drawbacks of such methods: (1) the depth distribution estimated is discrete and sparse; (2) the boundaries of objects are difficult to handle. A lot of works [42–47] follow this LSS paradigm as well. To improve the accuracy of depth distribution, CaDDN [42] uses depth ground truth derived from LiDAR point cloud as supervision (Fig. 10.13). BEVDepth [47] also uses LiDAR points as depth supervision. In addition, the intrinsic and extrinsic parameters of camera are introduced as the depth estimation prior, and the input features are adjusted in a SE-like manner. In BEVDepth, the predictions of depth and context are separated, and additional Resnet blocks are used to increase discrimination. FIERY [43] using EfficientNet [48] to extract features and predict depth distribution in the same way with LSS [36]. BEVFusion [46] adds LiDAR branch (VoxelNet [14]) onto LSS, and fuses camera features and LiDAR features in BEV space following Transfusion [49] paradigm. BEVDet [44] and its temporal enhenced version [45] follow the similar paradigm to learn an implicit view transformation (Figs. 10.14 and 10.15).

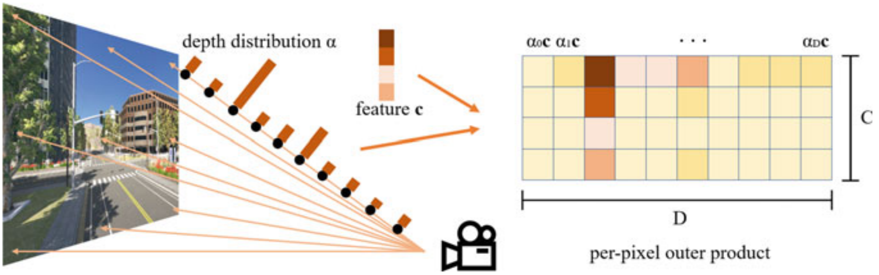


Fig. 10.12 The "lift" step of LSS [36]

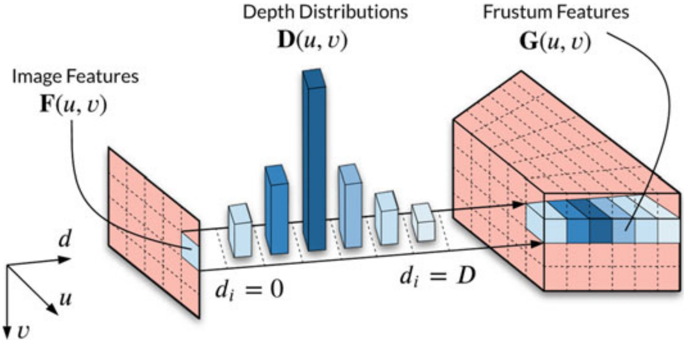


Fig. 10.13 Construction of frustum features in CaDDN [42]

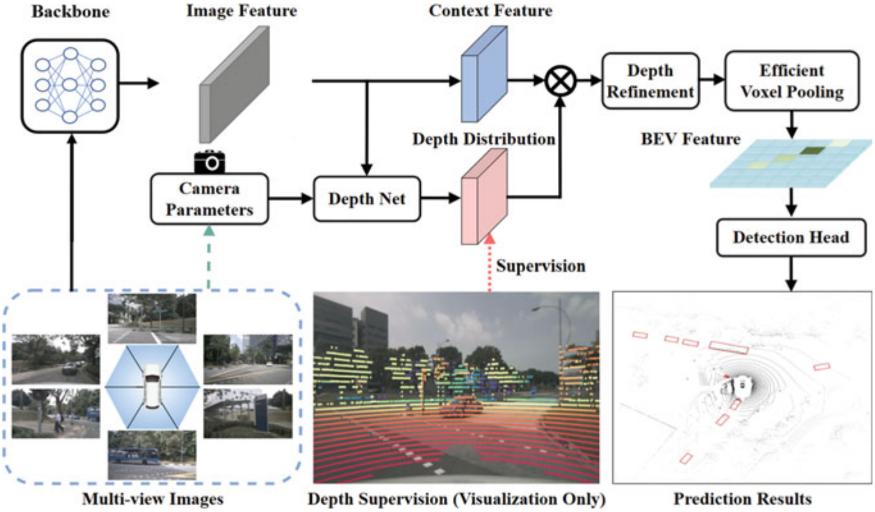


Fig. 10.14 Framework of BEVDepth [47]

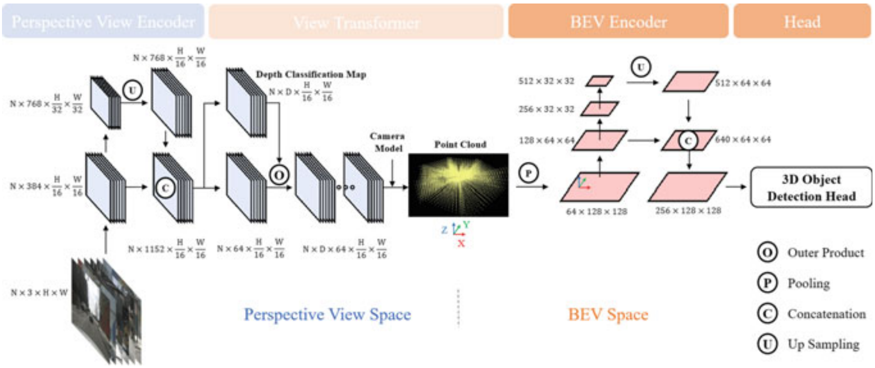

Perspective View Encoder
View Transformer
BEV Encoder
Head
$N \times 768 \times \frac{H}{32} \times \frac{W}{32}$
$N \times 768 \times \frac{H}{16} \times \frac{W}{16}$
$N \times 384 \times \frac{H}{16} \times \frac{W}{16}$
$N \times 1152 \times \frac{H}{16} \times \frac{W}{16}$
$N \times 64 \times \frac{H}{16} \times \frac{W}{16}$
$N \times D \times 64 \times \frac{H}{16} \times \frac{W}{16}$
$N \times 3 \times H \times W$
Depth Classification Map
$N \times D \times \frac{H}{16} \times \frac{W}{16}$
Camera Model
Point Cloud
Z Y X
U
C
O
P
512 × 32 × 32
256 × 32 × 32
128 × 64 × 64
512 × 64 × 64
640 × 64 × 64
64 × 128 × 128
256 × 128 × 128
3D Object Detection Head
Perspective View Space
BEV Space
O Outer Product
P Pooling
C Concatenation
U Up Sampling

Fig. 10.15 The framework of the BEVDet [44] paradigm

10.4.2 3D-2D Methods

For 3D-2D methods, the 3D information is reconstructed by encoding 2D features to 3D space via 3D prior such as 3D-2D projection mapping. Geometry-based methods are introduced by Inverse Perspective Mapping. For pure network-based methods, MLP and transformer are two main branches, Some works index local features according to 3D-2D projection such as explicit BEV feature modeling [50, 51], or DETR3D [1] and its derivants [52]. Other methods utilize implicit 3D positional encoding [53] to avoid direct detph estimation.

10.4.2.1 IPM-Based Methods

Inverse Perspective Mapping (IPM) is the pioneer work of 3D-2D methods. Due to the lack of depth information caused by perspective mapping, projecting the pixels on image back to 3D space is ill-posed. Mallot et al. present IPM [10] to solve this ill-posed problem by adding constraint that the inversely mapped points lie on the ground plane, which means that the mapping can be described via a homography matrix. The detailed foundation of mathematics is introduced in Sect. 10.2. In practical applications, the homography matrix can be mathematically derived from the calibration parameters of camera settings, such as intrinsic and extrinsic matrices.

Abbas and Zisserman [54] use CNNs to extract 2D features and estimate the vertical vanishing points and horizontal vanishing lines to determine the homography matrix. After the PV-BEV transformation by IPM, many downstream perception tasks can be done in the BEV space, such as optical flow estimation, object detection, map segmentation, object tracking, path planning, etc. VPOE [55] adopts YOLOv3 [56] as the detection head to estimate the object pose on BEV. Palazzi et al. [57] maps detections from PV to BEV occupancy grid map based on synthetic dataset. In practice, the intrinsic and extrinsic parameters generally unknown or the extrinsic parameters may change due to the bumpy road. To overcome this issue, TrafCam3D [58] introduces a dual-view network architecture to perform a more robust homography mapping.

IPM-based methods have good interpretability of the PV-BEV transformation. However, there are distorted areas where the objects are above the ground plane. To reduce the distortion, a series of subsequent works [50, 58–66] explore the semantic information or aid the domain gap between PV and BEV using GANs [67].

10.4.2.2 MLP-Based Methods

IPM-based methods are explicitly based on physical reality and strong mathematical assumptions. On the contrary, MLP-based methods leverage the geometry of camera. In MLP-based methods, the multilayer perceptron (MLP) constructs a universal approximate mapping function of view transformation in a data-driven manner.

Adopting MLP to perform 3D-2D feature projection is initially introduced by OFT [68], which projects 2D features to 3D voxel space (Fig. 10.16). Rather than predict the depth distribution, OFT scatters the same image feature along the ray from nodal point to a specific 3D point, in other word, it assumes that the depth distribution is uniform. Such assumption works in flat roads but fails in undulating roads. VED [69] introduces a variational encoder-decoder (VED) architecture with an MLP bottleneck to convert PV image to semantic BEV occupancy grid map end-to-end. VPN [70] utilizes a two-layer MLP to perform the view transformation. FishingNet [71] follows the same pattern with VPN and fuses LiDAR and RADAR data in a late fusion manner for downstream tasks. To overcome the difficulties including occlusion, lack of depth information and small objects, STA-ST [72] and PON [73] make use of Feature Pyramid Networks (FPNs) [74] to extract multi-scale image features, and compress the features along height axis and expand along dpeth aixs to leverage the vertical context, as shown in Fig. 10.17. Leveraging the vertical context via column-wise compression makes the network model more robust to occlusion and inaccessible

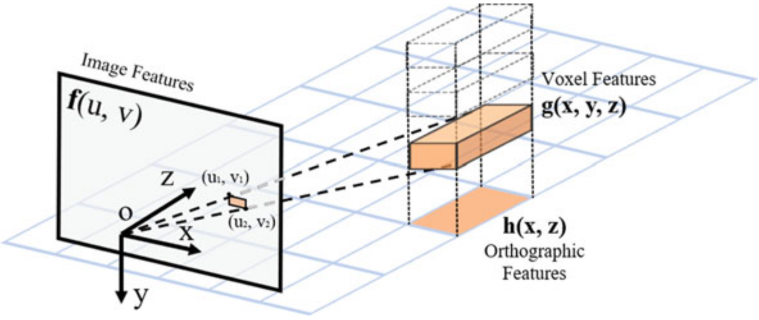


Fig. 10.16 Orthographic Feature Transform (OFT) [68]. Orthographic features are generated by feature accumulation and compression

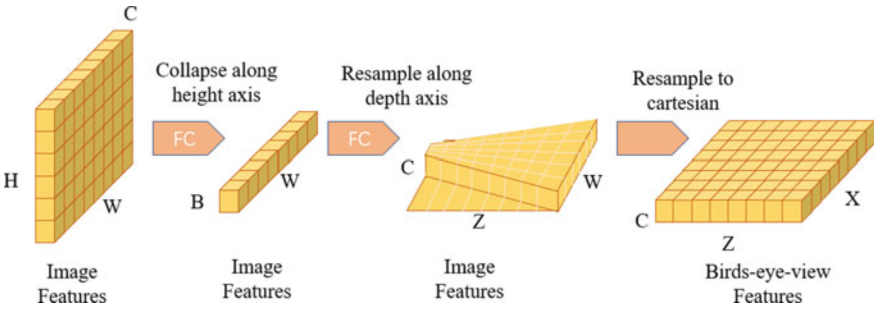


Fig. 10.17 PON [73]: features extracted by FPN are collapsed along height axis and expanded along depth axis

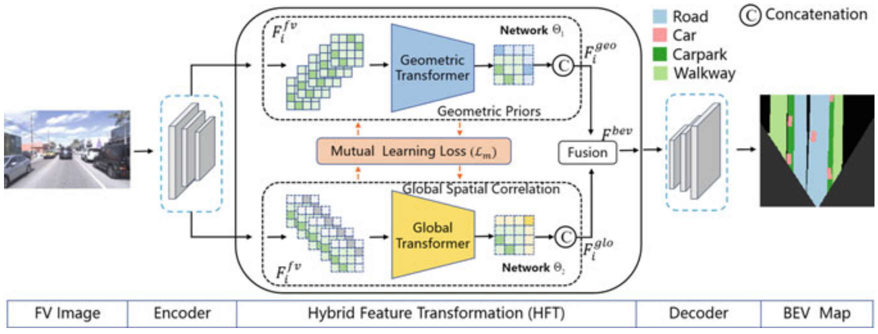


Fig. 10.18 An overview architecture of HFT [77]

depth information. It is also widely adopted and promoted in the transformer-based methods, which is discussed in Sect. 10.4.2.3.

To improve the accuracy of vectorized HD map, HDMapNet [75] uses an extra MLP to build bidirectional projection between PV and BEV to check whether the features are correctly mapped. Inspired by the back projection, PYVA [76] puts forward a bidirectional self-supervision scheme with stronger attention-based BEV feature. HFT [77] combines a camera model-based branch with a camera model-free branch via a hybrid feature transformation to utilize geometric prior and capture global context respectively (Fig. 10.18).

10.4.2.3 Transformer-Based Methods

With the development of attention mechanism and vision transformer (ViT), a lot of BEV perception works utilize transformer architecture to model the 3D-2D view transformation in an implicit way. On the strength of the cross attention, transformer is more data-dependent, thus has more powerful spatio-temporal representation but

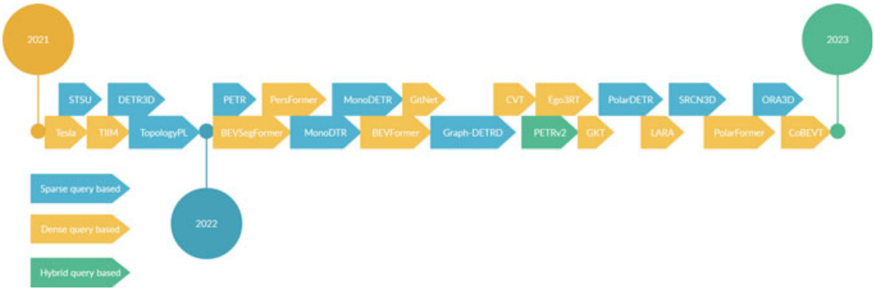


Fig. 10.19 Chronological overview of transformer-based methods [78]

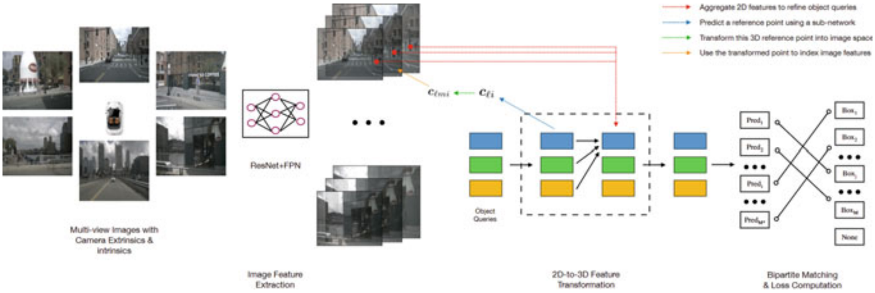

Aggregate 2D features to refine object queries
Predict a reference point using a sub-network
Transform this 3D reference point into image space
Use the transformed point to index image features
Multi-view Images with Camera Extrinsics & Intrinsics
ResNet+FPN
Image Feature Extraction
$c_{\ell mi}$
$c_{\ell i}$
Object Queries
2D-to-3D Feature Transformation
Pred_1
Pred_2
Pred_i
Pred_{M^*}
Box_1
Box_2
Box_i
Box_M
None
Bipartite Matching & Loss Computation

Fig. 10.20 Overview of DETR3D [1]

is difficult to train. Designing effective positional encoding and constructing appropriate queries are still challenging.

Based on the granularity of learnable queries in the transformer decoder, the transformer-based methods can be divided into three categories: sparse query-based, dense query-based and hybrid query-based [78]. A chronological overview of transformer-based methods is illustrated in Fig. 10.19. In terms of specific BEV perception downstream tasks, sparse queries are commonly used for detection tasks and dense queries are commonly used for segmentation tasks.

Sparse query-based methods are represented by the "DETR" family [1, 41, 53, 79–84]. As shown in Fig. 10.20, DETR3D [1] takes multi-view images as input and adopts ResNet and FPN to extract 2D features. After defining a set of sparse object queries, each one is translated into a 3D reference point. By reprojecting the 3D reference point into the image space, 2D features are sampled in the meantime to refine the object queries. In the end, it employs a set-to-set loss and produces predictions for each query. In order for sparse queries to interact with 2D features directly in the vanill cross attention, PETR [53] encodes 3D positional embedding derived from camera settings into 2D multi-view features, which considerably streamlines the feature sampling procedure in DETR3D (Fig. 10.21). The follow-up study PETRv2 [81] makes use of the temporal information by extending the 3D positional embedding from spatial domain to temporal domain. Graph-DETR3D [82] makes use of graph

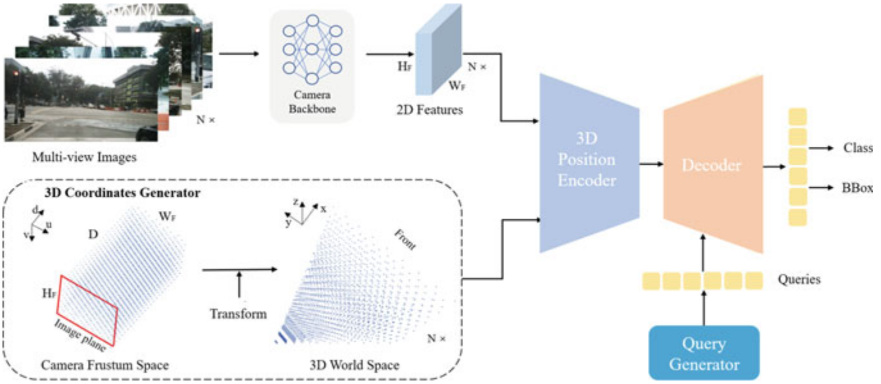


Fig. 10.21 The architecture of the proposed PETR paradigm [53]

structure learning, which enhances the inadequate feature aggregation of DETR3D and improves the performance in the overlap regions. ORA3D [83] utilizes stereo disparity supervision and adversarial training. PolarDETR [84] reformulates the 3D detection task in the polar coordinate system, which takes full advantage of the symmetry of multiple views. SRCN3D [85] designs the first two-stage 3D-2D surround-view camera 3D detection approach in a fully-convolutional architecture based on SparseRCNN [86]. It replaces global attention with local dynamic instance interaction head to get lower computation cost.

Dense queries are pre-allocated with corresponding positions in 3D space or BEV space for dense query-based approaches. The quantity of dense queries (spatial resolution) is typically larger than the quantity of sparse queries in sparse query-based algorithms. For a variety of downstream tasks, including 3D detection, motion prediction and map segmentation, the interaction between image features and dense queries can lead to the dense BEV representation.

Tesla [87] takes advantage of positional encoding and context summary to generate dense queries in the BEV space, and performs the view transformation using the vanilla cross attention between image features in multiple views and dense BEV queries, without considering the camera parameters. CVT [2] constructs a camera-aware positional embedding derived from calibrated parameters of surround-view cameras. Then through a succession of cross attention layers, it learns a BEV positional embedding that aggregate cross-view information (Fig. 10.22).

To balance the memory consumption and model scalability, Persformer [51] and BEVSegFormer [88] adopt deformable attention [89] in the view transformation module for 3D lane detection and BEV segmentation, respectively. BEVFormer [5] takes advantage of the deformable attention to interact dense BEV queries with multi-view features. Current queries and history queries are also interacted via deformable attention to leverage temporal information. Inspired by the ray tracing perspective, Ego3RT [90], as shown in Fig. 10.23, learns ego 3D representation from 2D image features making use of deformable attention and executes multiple downstream tasks.

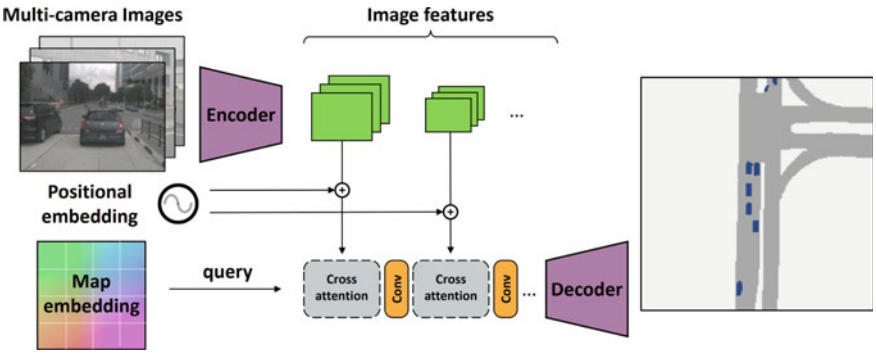


Fig. 10.22 The pipeline of CVT [2]

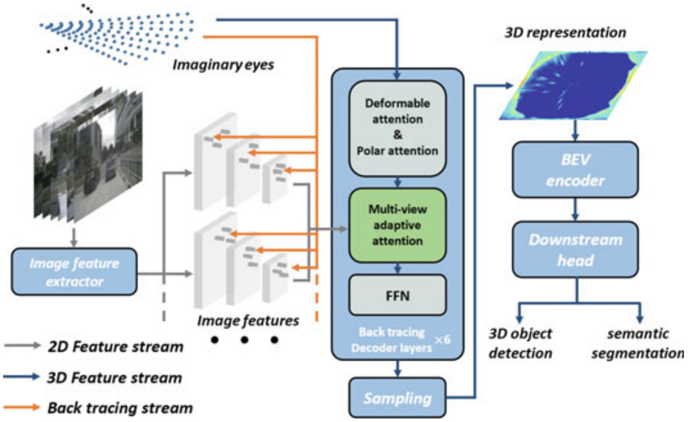


Fig. 10.23 Ego3RT pipeline [90]

CoBEVT [91] introduces fused axial attention (FAX), a novel attention variant, which can effectively aggregate features in both local and global agent or camera views.

The geometric priors are used by GKT [93] to guide the transformer to concentrate on discriminative regions and unfolds kernel features to produce BEV representation. TIIM [92] treats 1–1 correspondence between each image column and BEV polar ray as a sequence-to-sequence translation, as shown in Fig. 10.24. Such geometric constraint avoids the dense cross attention between 2D image features and BEV queries, and makes it more suitable for the transformer-style architecture. GitNet [94] obtains coarse pre-aligned BEV features by a geometry-guided pre-alignment module and refines the features via a ray-based transformer like TIIM. PolarFormer [95] extends such ray-based or column-wise transformer from a single camera to multiple surrounding cameras. Instead of learning a quadratic “all-to-all” correspondence between features in multi-camera space and BEV space, LaRa [96] uses a moderately fixed size latent space to control computation and memory consumption of the view transformation.

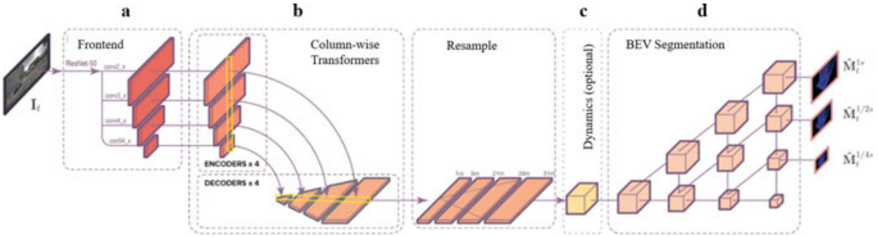


Fig. 10.24 Architecture of TIIM framework [92]. **a** Construct spatial representations in the image-plane. **b** Transform image-plane representations to BEV. **c** Construct spatiotemporal representation in BEV-plane (optional) . **d** Semantically segmentation BEV representation

10.5 Fusion-Based BEV Perception

Sensor fusion in autonomous driving is mainly about the fusion of images and point clouds (either LiDAR or RADAR). Previous methods conduct fusion in data-level [97, 98] or feature-level [25, 99–102], which rely either on calibration or directly using high-dimension features. As we have introduced in previous sections, BEV space provides a unified and convenient representation for LiDAR, camera and RADAR. Meanwhile, temporal information can be easily fused in the BEV space via ego-motion of vehicle. With the concern of robustness and consistency, a lot of works conduct BEV perception in a fusion-based way.

10.5.1 Multi-modal Fusion

Some methods operate feature fusion in 3D space [33, 89, 103–106]. As Fig. 10.25 shows, UVTR converts image and point cloud to modality-specific voxel space and fuses the features with a voxel encoder. Some other methods perform feature fusion in BEV space [46, 71, 107]. As a representative work, BEVFusion [46] extracts multi-modal features from LiDAR and multi-view cameras, the features are transformed into a shared BEV space efficiently and fused with a fully-convolutional BEV encoder. Then the encoded features are decoded by task-specific heads to support different perceptual tasks (Fig. 10.26). Besides fusion in 3D space or BEV space, there is a kind of methods conduct fusion via general queries [49, 52]. As shown in Fig. 10.27, FUTR3D [52] designs a query-based Modality-Agnostic Feature Sampler (MAFS) to extract features from all available modalities according to the 3D reference point of each query, which is easily adaptable to all sensor configurations and combinations.

The majority of recent research relies on single-agent systems, which struggle to handle occlusions and recognize far-off objects in complicated traffic situations. This problem can be addressed because to the advancement of Vehicle-to-Vehicle (V2V) communication technologies, which broadcast sensor data to other adjacent

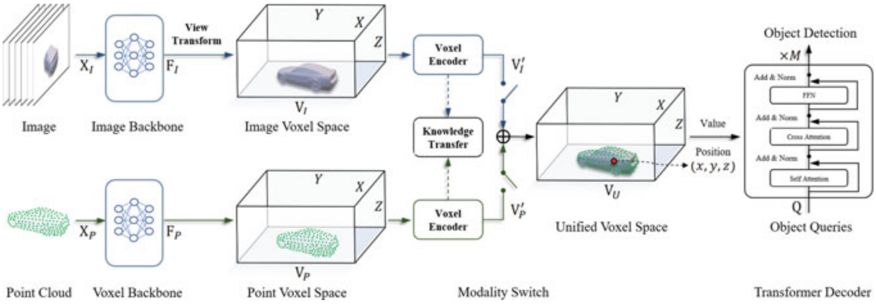


Fig. 10.25 The framework of UVTR with multi-modality inputs [103]

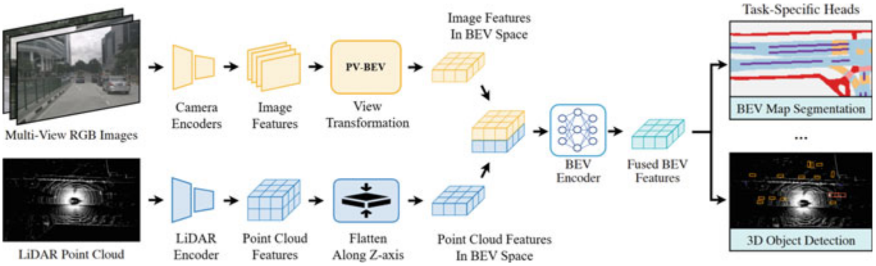


Fig. 10.26 The architecture of BEVFusion [46]

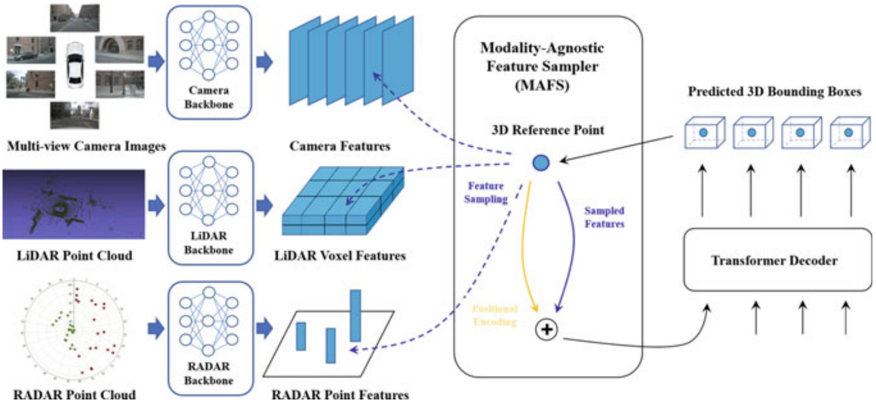


Fig. 10.27 Overview of FUTR3D [52]

autonomous cars to provide alternative perspectives of the same scene. Inspired by v2v communication technology, CoBEVT [91] develops a multi-agent multi-camera perception architecture that can cooperatively produce BEV map. However, it has only been validated on simulated datasets [108]. Most methods rely on object-ground intersections (for example, shadows) as context for depth reasoning [109]. However, these methods become unreliable for distant objects. Besides vehicle-to-vehicle communication, object-to-object relationship is also studied. Reference [110] proposes to localize object depth by comparing objects to each other and perform message-passing across a graph of the objects via a graph neural network.

10.5.2 Temporal Fusion

To alleviate the occlusion and estimate the motion of objects, temporal fusion is an effective way for robust BEV perception system. BEVDet4D [45] fuses the feature maps with sparial alignment and concatenation. A similar concatenation-based approach has also been used in other works, including TIIM [92], FIERY [43], and PolarFormer [95]. Conerning the object motion and ego-motion, BEVFormer [5], PETRv2 [81] and UniFormer [111] utilize attention module to fuse temporal information from either previous BEV feature maps or previous frames, to more effectively create associations of the same objects in different timestamps (Fig. 10.28).

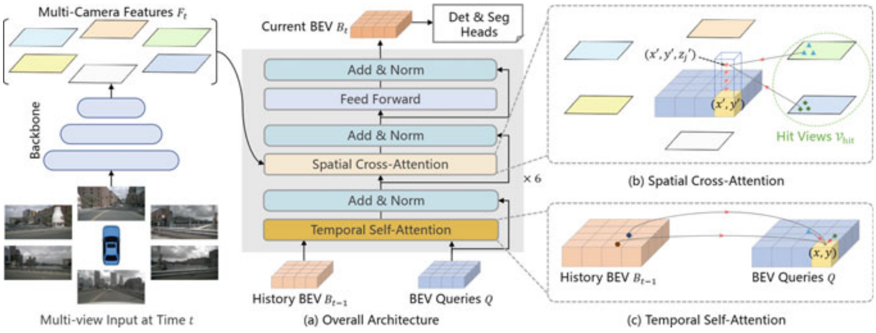


Fig. 10.28 Overall architecture of BEVFormer [5]

10.6 Datasets

KITTI [112], nuScenes [113] and Waymo Open Dataset (WOD) [114] are the most influential benchmarks for BEV perception. KITTI is a well-known benchmark for multiple autonomous driving tasks, such as stereo, optical flow, visual odometry, object detection and 3D tracking [112]. It contains 3712, 3769 and 7518 images for training, validation, and testing, respectively, also with corresponding LiDAR point clouds. The evaluation consists of 3D object detection and BEV evaluation. And there are three levels of difficulty according to objects' size, occlusion and truncation. The nuScenes dataset is a public large-scale dataset for autonomous driving developed by the team at Motional [113]. There are 1000 driving scenes in Boston and Singapore, two cities that are known for their dense traffic and highly challenging driving situations. Among them, 850 scenes are used for training and validation, 150 scenes are used for testing, with duration of 20 seconds each. The sensors include 6 surrounding cameras, 1 LiDAR and 5 RADARs. The resolution of image is 1600 × 900 pixels. In the meantime, corresponding CAN-bus data and high-definition map (HD Map) are available to investigate the help of numerous inputs. In the training, validation, and testing sets of the Waymo Open Dataset [114], there are 798, 202, and 80 video sequences, respectively. Each sequence consists of 5 LiDARs and 5 views (side left, front left, front, front right and side right), the image resolution is 1920 × 1280 pixels or 1920 × 886 pixels.

Besides the datasets mentioned above, more benchmarks such as Argoverse, Lyft, KITTI-360, H3D, are also applicable to for BEV perception. Table 10.1 provides a summary of the detailed information for these benchmarks.

Table 10.1 Datasets for BEV Perception

Dataset	Views	Scenes	Scans	Images	Det.	Seg.	Classes	Night/rain	Stereo
KITTI 3D [112]	1	–	15 K	15 K	80 K	–	8 (3)	✗/✗	✓
nuScenes [113]	6	1,000	390 K	1.4 M	1.4 M	40 K	23 (10)	✓/✓	✗
Waymo Open Dataset [114]	5	1,150	230 K	12 M	12 M	50 K	4 (3)	✓/✓	✗
Argoverse V1 [115]	7	113	22 K	490 K	993 K	–	15	✓/✓	✓
Argoverse V2 [116]	7	1000	150 K	2.7 M	–	–	26	✓/✓	✓
Lyft L5 [117]	6	366	46 K	260 K	1.3 M	–	9	✗/✗	✗
H3D [118]	3	160	27 K	83 K	1.1 M	–	8	✗/✗	✗
Cityscapes 3D [119]	1	–	–	5 K	40 K	–	8 (6)	✓/✓	✓
KITTI 360 [120]	4	11	80 K	320 K	68 K	80 K	19	✗/✗	✓

KITTI 3D: https://www.cvlibs.net/datasets/kitti/
nuScenes: https://www.nuscenes.org/nuscenes
Waymo Open Dataset: https://waymo.com/open/
Argoverse v1: https://www.argoverse.org/av1.html
Argoverse v2: https://www.argoverse.org/av2.html
Lyft L5: https://www.woven-planet.global/en/data/perception-dataset
H3D: https://usa.honda-ri.com/H3D
Cityscapes 3D: https://www.cityscapes-dataset.com
KITTI 360: https://www.cvlibs.net/datasets/kitti-360/

10.7 Evaluation Metrics

The most commonly used evaluation metric for BEV detection is average precision (AP), which refers to the area under the precision-recall curve. In order to calculate AP, the Intersection-over-Union (IoU) is used to measure the difference between predictions and ground truth (annotations, labels). The definition of IoU between prediction bounding box and ground truth is given by (10.5):

$$IoU = \frac{area \quad of \quad overlap}{area \quad of \quad union}. \tag{10.5}$$

If the IoU is beyond the threshold, the prediction is seen as True Positive (TP). Otherwise, the result is treated as False Positive. False Negtive (FN) and True Negtive (TN) can be defined in the same way. Thus, Precision and Recall can be calculated as follows:

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{Predictions}, \tag{10.6}$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{GroundTruth}. \tag{10.7}$$

Using the interpolated values, AP can be calculated by (10.8):

$$AP = \frac{1}{|\mathbb{R}|} \sum_{r \in \mathbb{R}} p_{interp}(r), \tag{10.8}$$

where $\mathbb{R}$ denotes the set of all recall positions, the interpolation function $p_{interp}(\cdot)$ is defined as: $p_{interp} = max_{r':r' \geq r} p(r')$. Mean average precision (mAP) is the average of APs of different classes or difficulty levels.

As for evaluation metrics for BEV segmentation, the IoU for each class and mIoU over all classes are the most frequently adopted.

Besides the common metrics, there are dataset-specific metrics.

- Average Orientation Similarity (AOS):

$$AOS = \frac{1}{|\mathbb{R}|} \sum_{r \in \mathbb{R}} \max_{r':r' \geq r} c(r'), \tag{10.9}$$

  where the c(r) denotes orientation similarity, which is a normalized variant of the cosine similarity.
- Average Translation Error (ATE) calculates the Euclidean distance between predicted object center and ground truth on the 2D ground plane.
- Average Scale Error (ASE) calculates the 3D IoU error $(1 - IoU)$ after performing the alignment of translation and orientation.

- Average Orientation Error (AOE) refers to the smallest yaw angle bias between the predictions and labels.
- Average Velocity Error (AVE) is defined as the L2 norm of 2D velocity differences.
- Average Attribute Error (AAE) indicates one minus attribute classification accuracy.
- nuScenes Detection Score (NDS) is the combination of the mean AP and the mean TP over all catergories:

$$NDS = \frac{1}{10}\left[5 \cdot mAP + \sum_{k=1}^{5}(1 - \min(1, mTP_k))\right]. \tag{10.10}$$

- Average Precision weighted by Heading (APH) takes account of heading angel while calculating the AP metric.
- Longitudinal Error Tolerant 3D Average Precision (LET-3D-AP) is designed to be more tolerant with respect to depth estimation errors.

$$LET - 3D - AP = \int_0^1 p(r)dr, \tag{10.11}$$

  where $p(r)$ is the precision value at recall r.
- Longitudinal Affinity Weighted LET-3D-APL (LET-3D-APL) penalizes the predictions that do not overlap with any ground truth.

$$LET - 3D - APL = \int_0^1 p_L(r)dr = \int_0^1 \bar{a}_l \cdot p(r)dr, \tag{10.12}$$

  where $p_L(r)$ indicates the longitudinal affinity weighted precision, the precision value at recall r is denoted by $p(r)$, and the factor $\bar{a}_l$ means the average longitudinal affinity of all matched predictions treated as TP.

10.8 Industrial Applications

In the industry, BEV perception has been on the rise in recent years. The system-level architecture design for BEV perception is discussed in this section. The various BEV perception architectures put forth by companies worldwide are summarized in Fig. 10.29. The industrial BEV perception architectures usually consist of data preprocessing, feature extractor, PV-BEV transformation, fusion module and task-specific prediction head. Each module is elaborated in details below.

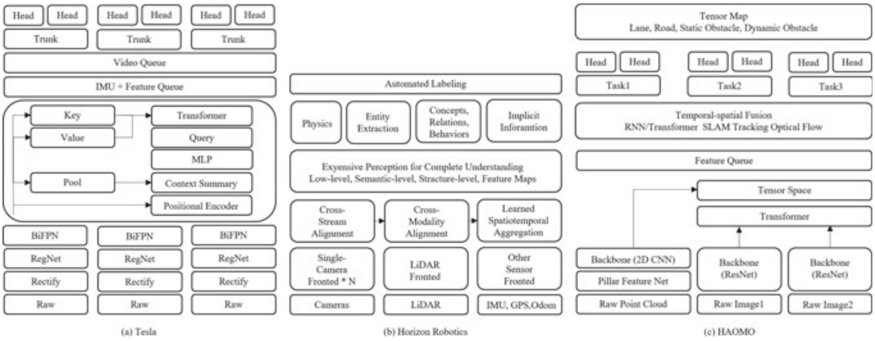


Fig. 10.29 BEV architecture comparison across industrial corporations [87, 121, 122]

10.8.1 Data Preprocessing

Several data modalities, including camera, LiDAR, RADAR, IMU, and GPS, are supported by BEV-based perception algorithms at present. The primary perception sensors for autonomous driving are cameras and LiDAR. Several products, such as those made by Tesla [87], PhiGent [123], and Mobileye [124], only use cameras as input sensors. The others use a variety of camera and LiDAR combinations, for example, Horizon [121] and HAOMO [122]. Be aware that sensor fusion designs frequently use IMU and GPS signals [87, 121, 122]. For examples, see Tesla, Horizon, and HAOMO. In the data preprocessing, numerous sensors are usually calibrated and synchronized.

10.8.2 Feature Extraction

A backbone network and a neck network frequently make up the feature extractor module, which transforms raw data into useful feature representations. Several feature extractor combinations exist for the backbone network and neck network. As an illustration, the image backbone network can be RegNet [125] in Tesla and ResNet [126] in HAOMO. The neck network could be FPN [74] adopted by HAOMO, BiFPN [127] utilized by Tesla, etc. The voxel-based option from Mobileye or the pillar-based choice from HAOMO are both ideal candidates for the backbone in terms of point cloud input.

10.8.3 PV-BEV Transformation and Fusion

In industry, there are mainly four approaches to perform PV-BEV transformation:

1. Fixed IPM. Projecting PV features into BEV space can be achieved by a fixed view transformation based on the assumption that the ground is flat. The ground plane is handled well by fixed IPM projection. Nonetheless, it is sensitive to road flatness and vehicle jolting.
2. Adaptive IPM. The extrinsic parameters of self-driving vehicles are used by adaptive IPM, which projects features to BEV in accordance with these parameters. Adaptive IPM still makes the flat ground assumption despite being robust to vehicle pose.
3. Transformer-based view transformation. Transforming PV features into BEV space using a dense transformer is known as transformer-based view transformation. Tesla, Horizon, and HAOMO have all broadly adopted this data-driven transformation since it works without making any assumptions first.
4. ViDAR (Pseudo-LiDAR). This term is first proposed by Waymo and Mobileye concurrently at different venues [124, 128] in early 2018, to describe the method of projecting PV features from visual inputs into BEV space, resembling the representation form of LiDAR point cloud. Therefore, point cloud-based techniques can be applied to obtain BEV features. Recently, there have been numerous ViDAR applications, including those from Tesla, Mobileye, Waymo, Toyota, [87, 124, 128–130], etc.

In the industrial world, ViDAR and transformer choices predominate. And the multi-modal fusion or spatiotemporal fusion are often adopted as a auxiliary of the view transformation module.

10.8.4 Perception Heads

The multi-head design is widely adopted in industrial BEV perception architecture as well. All prediction results such as motion prediction, map segmentation and 3D bounding boxes are decoded from BEV feature space since BEV feature is a unified representation that aggregates multi-modal information. In some designs, prediction results in PV are also decoded from the associated PV features. According to Horizon Robotics [121], the prediction results can be classified into four categories: (a) Low-level results are related to physics constrains, such as optical flow, depth, normal vector, etc. (b) Semantic-level results include entity extractions, for example, vehicle bounding boxes, laneline segmentation, etc. (c) Structure-level results represent relationship between objects, including object tracking, motion prediction, etc. (d) Implicit information such as feature map of auxiliary task.

10.9 Existing Challenges

The processing of raw point cloud in LiDAR-based methods requires high memory consumption and computational cost. However, converted representations such as voxel and pillar lose information to varying degrees. Generating a balanced trade-off between performance and efficiency becomes a vital challenge for LiDAR-based BEV perception.

For camera-based methods, the core issue is to reconstruct 3D information from 2D images. In 2D-3D methods, the effort is devoted to estimating depth or depth distribution. However, "Pseudo-LiDAR" methods can not estimate accurate depth as real LiDAR, while "Lift" methods do not perform well at object boundaries and far away distance. Thus a better depth estimation approach is in urgent need to be explored. In 3D-2D methods, recent studies devote to complementing geometric into network approaches. IPM-based methods have good interpretability but can not avoid distortions above the ground plane. MLP-based methods are theoretically a universal approximator [131], but not fully explored in the depth, occlusion and multi-view. Transformer-based methods have best performance at present but still need to further investigate better queries, spatiotemporal fusion, network lightweight and polar parameterization. For network training, effective ground truth annotation for the BEV grid is still pending.

In general, the existing challenges of BEV perception are:

- Proposing a better representation of point cloud;
- Designing a better depth estimator for 2D images;
- Achieving better generalization ability and robustness;
- Obtaining the truth value annotation in the BEV grid conveniently;
- Trading-off between performance and efficiency in applications.

10.10 Conclusions

In the traditional approach, various perception tasks such as 3D object detection, obstacle instance segmentation, lane line segmentation, and trajectory prediction are separated from each other. This separation makes the autonomous driving algorithm need to use multiple sub-modules in series, which greatly increases the development and maintenance cost of the whole system. BEV perception allows these perception tasks to be implemented within a single algorithmic framework, significantly reducing the need for manpower. Considering the advantages of BEV mentioned above, many research institutions and major car companies are promoting the implementation of BEV solutions. Corresponding BEV algorithms have been proposed based on different combinations of sensor input layers, basic tasks, and product scenarios. While it is certain that BEV perception algorithm can integrate the features of multiple sensors and improve the accuracy of perception and prediction, its potential to be the ultimate solution for autonomous driving is uncertain and depends on various factors.

Acknowledgements This work was supported by the National Key R&D Program of China under Grant 2020AAA0108100, the National Natural Science Foundation of China under Grant 62233013, and the Science and Technology Commission of Shanghai Municipal under Grant 22511104500.

References

1. Wang Y et al (2022) DETR3D: 3D object detection from multi-view images via 3D-to-2D queries. In: Conference on robot learning (CoRL). PMLR, pp 180–191
2. Zhou B, Krähenbühl P (2022) Cross-view transformers for real-time map-view semantic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 13760–13769
3. Zhang Y et al (2022) Beverse: unified perception and prediction in birds-eye-view for vision-centric autonomous driving. Comput Res Repos (CoRR). arXiv:2205.09743
4. Yulan G et al (2020) Deep learning for 3D point clouds: a survey. IEEE Trans Pattern Anal Mach Intell 43(12):4338–4364
5. Li Z et al (2022) BEVFormer: learning Bird's-eye-view representation from multi-camera images via spatiotemporal transformers. Comput Res Repos (CoRR). arXiv:2203.17270
6. Rui F et al (2018) Road surface 3D reconstruction based on dense subpixel disparity map estimation. IEEE Trans Image Process 27(6):3025–3035
7. Rui F et al (2021) Learning collision-free space detection from stereo images: homography matrix brings better data augmentation. IEEE/ASME Trans Mechatron 27(1):225–233
8. Rui F et al (2021) Rethinking road surface 3-d reconstruction and pothole detection: from perspective transformation to disparity map segmentation. IEEE Trans Cybern 52(7):5799–5808
9. Luo L-B et al (2010) Low-cost implementation of bird's-eye view system for camera-on-vehicle. In: ICCE 2010 - 2010 digest of technical papers international conference on consumer electronics (ICCE), pp 311–312
10. Mallot HA et al (1991) Inverse perspective mapping simplifies optical flow computation and obstacle detection. Biol Cybern 64(3):177–185
11. Li H et al (2022) Delving into the devils of bird's-eye-view perception: a review, evaluation and recipe. Comput Res Repos (CoRR). arXiv:2209.05324
12. Graham B (2014) Spatially-sparse convolutional neural networks. Comput Res Repos (CoRR). arXiv:1409.6070
13. Choy C et al (2019) 4D spatio-temporal convnets: Minkowski convolutional neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 3075–3084
14. Zhou Y, Tuzel O (2018) VoxelNet: end-to-end learning for point cloud based 3D object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 4490–4499
15. Yan Y et al (2018) SECOND: Sparsely embedded convolutional detection. Sensors 18(10):3337
16. Yin T et al (2021) Center-based 3D object detection and tracking. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 11784–11793
17. Shi S et al (2020) PV-RCNN: point-voxel feature set abstraction for 3d object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 10529–10538

18. Chenhang He et al (2020) Structure aware single-stage 3D object detection from point cloud. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 11873–11882
19. Jiajun D et al (2021) Voxel R-CNN: towards high performance voxel-based 3D object detection. Proceedings of the AAAI conference on artificial intelligence (AAAI) 35:1201–1209
20. Wang Y, Solomon JM (2021) Object DGCNN: 3D object detection using dynamic graphs. Adv Neural Inf Process Syst (NeurIPS) 34:20745–20758
21. Mao J et al (2021) Voxel transformer for 3D object detection. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 3164–3173
22. Fan L et al (2022) Embracing single stride 3D object detector with sparse transformer. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 8458–8468
23. Liu Z et al (2021) Swin transformer: hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 10012–10022
24. Hu Y et al (2022) AFDetV2: rethinking the necessity of the second stage for object detection from point clouds. Proceedings of the AAAI conference on artificial intelligence (AAAI) 36:969–979
25. Chen X et al (2017) Multi-view 3D object detection network for autonomous driving. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 1907–1915
26. Yang B et al (2018) Pixor: real-time 3D object detection from point clouds. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 7652–7660
27. Yang B et al (2018) HDNet: exploiting HD maps for 3D object detection. In: Conference on robot learning (CoRL). PMLR, pp 146–155
28. Beltrán J et al (2018) BirdNet: a 3D object detection framework from LiDAR information. In: 2018 21st international conference on intelligent transportation systems (ITSC). IEEE, pp 3517–3523
29. Yiming Z et al (2018) RT3D: real-time 3-D vehicle detection in LiDAR point cloud for autonomous driving. IEEE Robot Autom Lett (RAL) 3(4):3434–3440
30. Ali W et al (2018) YOLO3D: end-to-end real-time 3D oriented object bounding box detection from LiDAR point cloud. In: Proceedings of the European conference on computer vision (ECCV) workshops
31. Simony M et al (2018) Complex-YOLO: an Euler-region-proposal for real-time 3D object detection on point clouds. In: Proceedings of the European conference on computer vision (ECCV) workshops
32. Lang AH et al (2019) PointPillars: fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 12697–12705
33. Qi CR et al (2017) PointNet: Deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 652–660
34. Wang Y et al (2019) Pseudo-LiDAR from visual depth estimation: bridging the gap in 3D object detection for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 8445–8453
35. You Y et al (2019) Pseudo-LiDAR++: accurate depth for 3D object detection in autonomous driving. Comput Res Repos (CoRR). arXiv:1906.06310
36. Philion J, Fidler S (2020) Lift, splat, shoot: encoding images from arbitrary camera rigs by implicitly unprojecting to 3D. In: European conference on computer vision (ECCV). Springer, pp 194–210
37. Ma X et al (2019) Accurate monocular 3D object detection via color-embedded 3D reconstruction for autonomous driving. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 6851–6860

38. Ma X et al (2020) Rethinking Pseudo-LiDAR representation. In: European conference on computer vision (ECCV). Springer, pp 311–327
39. Lian Q et al (2022) Exploring geometric consistency for monocular 3D object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 1685–1694
40. Simonelli A et al (2021) Are we missing confidence in Pseudo-LiDAR methods for monocular 3D object detection? In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 3225–3233
41. Qian R et al (2020) End-to-end Pseudo-LiDAR for image-based 3D object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 5881–5890
42. Reading C et al (2021) Categorical depth distribution network for monocular 3D object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 8555–8564
43. Hu A et al (2021) FIERY: future instance prediction in Bird's-Eye view from surround monocular cameras. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 15273–15282
44. Huang J et al (2021) BEVDet: high-performance multi-camera 3D object detection in bird-eye-view. Comput Res Repos (CoRR). arXiv:2112.11790
45. Huang J, Huang G (2022) BEVDet4D: exploit temporal cues in multi-camera 3D object detection. Comput Res Repos (CoRR). arXiv:2203.17054
46. Liu Z et al (2022) BEVFusion: multi-task multi-sensor fusion with unified Bird's-eye view representation. Comput Res Repos (CoRR). arXiv:2205.13542
47. Li Y et al (2022) BEVDepth: acquisition of reliable depth for multi-view 3D object detection. Comput Res Repos (CoRR). arXiv:2206.10092
48. Tan M, Le Q (2019) EfficientNet: rethinking model scaling for convolutional neural networks. In: International conference on machine learning (ICML). PMLR, pp 6105–6114
49. Bai X et al (2022) TransFusion: robust LiDAR-camera fusion for 3D object detection with transformers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 1090–1099
50. Garnett N et al (2019) 3D-LaneNet: end-to-end 3D multiple lane detection. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 2921–2930
51. Chen L et al (2022) PersFormer: 3D lane detection via perspective transformer and the OpenLane benchmark. Comput Res Repos (CoRR). arXiv:2203.11089
52. Chen X et al (2022) FUTR3D: a unified sensor fusion framework for 3D detection. Comput Res Repos (CoRR). arXiv:2203.10642
53. Liu Y et al (2022) PETR: position embedding transformation for multi-view 3D object detection. Comput Res Repos (CoRR). arXiv:2203.05625
54. Abbas SA, Zisserman A (2019) A geometric approach to obtain a bird's eye view from an image. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV) workshops
55. Kim Y, Kum D (2019) Deep learning based vehicle position and orientation estimation via inverse perspective mapping image. In: 2019 IEEE intelligent vehicles symposium (IV). IEEE, pp 317–323
56. Redmon J, Farhadi A (2018) YOLOv3: an incremental improvement. Comput Res Repos (CoRR). arXiv:1804.02767
57. Palazzi A et al (2017) Learning to map vehicles into bird's eye view. In: International conference on image analysis and processing (ICIAP). Springer, pp 233–243
58. Loukkal A et al (2021) Driving among flatmobiles: Bird-eye-view occupancy grids from a monocular camera for holistic trajectory planning. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision (WACV), pp 51–60
59. Can YB et al (2022) Understanding bird's-eye view of road semantics using an onboard camera. IEEE Robot Autom Lett (RAL) 7(2):3302–3309

60. Reiher L et al (2020) A sim2real deep learning approach for the transformation of images from multiple vehicle-mounted cameras to a semantically segmented image in bird's eye view. In: 2020 IEEE 23rd international conference on intelligent transportation systems (ITSC). IEEE, pp 1–7
61. Hou Y et al (2020) Multiview detection with feature perspective transformation. In: European conference on computer vision (ECCV). Springer, pp 1–18
62. Gu JA (2020) Homography loss for monocular 3D object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 1080–1089
63. Zhu X et al (2018) Generative adversarial frontal view to bird view synthesis. In: 2018 International conference on 3d vision (3DV). IEEE
64. Srivastava S et al (2020) Learning 2D to 3D lifting for object detection in 3D for autonomous vehicles. In: 2019 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 4504–4511
65. Mani K et al (2020) Monolayout: amodal scene layout from a single image. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision (WACV), pp 1689–1697
66. Bruls T et al (2019) The right (angled) perspective: improving the understanding of road scenes using boosted inverse perspective mapping. In: 2019 IEEE intelligent vehicles symposium (IV). IEEE, pp 302–309
67. Antonia C et al (2018) Generative adversarial networks: an overview. IEEE Signal Process Mag 35(1):53–65
68. Roddick T et al (2018) Orthographic feature transform for monocular 3d object detection. Comput Res Repos (CoRR). arXiv:1811.08188
69. Lu C et al (2019) Monocular semantic occupancy grid mapping with convolutional variational encoder-decoder networks. IEEE Robot Autom Lett (RAL) 4(2):445–452
70. Bowen P et al (2020) Cross-view semantic segmentation for sensing surroundings. IEEE Robot Autom Lett (RAL) 5(3):4867–4873
71. Hendy N et al (2020) Fishing net: future inference of semantic heatmaps in grids. Comput Res Repos (CoRR). arXiv:2006.09917
72. Saha A et al (2021) Enabling spatio-temporal aggregation in birds-eye-view vehicle estimation. In: 2021 IEEE international conference on robotics and automation (ICRA). IEEE, pp 5133–5139
73. Roddick T, Cipolla R (2020) Predicting semantic map representations from images using pyramid occupancy networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 11138–11147
74. Lin T-Y et al (2017) Feature pyramid networks for object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 2117–2125
75. Li Q et al (2022) HDMapNet: an online HD map construction and evaluation framework. In: 2022 international conference on robotics and automation (ICRA). IEEE, pp 4628–4634
76. Yang W et al (2021) Projecting your view attentively: monocular road scene layout estimation via cross-view transformation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 15536–15545
77. Zou J et al (2022) HFT: lifting perspective representations via hybrid feature transformation. Comput Res Repos (CoRR). arXiv:2204.05068, 2022
78. Ma Y et al (2022) Vision-centric BEV perception: a survey. Comput Res Repos (CoRR). arXiv:2208.02797
79. Can YB et al (2021) Structured bird's-eye-view traffic scene understanding from onboard images. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 15661–15670
80. Can YB et al (2022) Topology preserving local road network estimation from single onboard camera image. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 17263–17272
81. Liu Y et al (2022) PETRv2: a unified framework for 3D perception from multi-camera images. Comput Res Repos (CoRR). arXiv:2206.01256

82. Chen Z et al (2022) Graph-DETR3D: rethinking overlapping regions for multi-view 3D object detection. Comput Res Repos (CoRR). arXiv:2204.11582
83. Roh W et al (2022) ORA3D: overlap region aware multi-view 3D object detection. Comput Res Repos (CoRR). arXiv:2207.00865
84. Chen S et al (2022) Polar parametrization for vision-based surround-view 3D detection. Comput Res Repos (CoRR). arXiv:2206.10965
85. Shi Y et al (2022) SRCN3D: Sparse R-CNN 3D surround-view camera object detection and tracking for autonomous driving. Comput Res Repos (CoRR). arXiv:2206.14451
86. Sun P et al (2021) Sparse R-CNN: end-to-end object detection with learnable proposals. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 14454–14463
87. Tesla. tesla ai day 2022. https://www.youtube.com/watch?v=j0z4FweCy4M Accessed August, 2021
88. Peng L et al (2023) BEVSegFormer: Bird's eye view semantic segmentation from arbitrary camera rigs. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision (WACV), pp 5935–5943
89. Zhu X et al (2020) Deformable DETR: Deformable transformers for end-to-end object detection. Comput Res Repos (CoRR). arXiv:2010.04159
90. Lu J et al (2020) Learning ego 3D Representation as Ray Tracing. Comput Res Repos (CoRR). arXiv:2206.04042, 2022
91. Xu R et al (2022) CoBEVT: cooperative bird's eye view semantic segmentation with sparse transformers. Comput Res Repos (CoRR). arXiv:2207.02202
92. Saha A et al (2022) Translating images into maps. In: 2022 international conference on robotics and automation (ICRA). IEEE, pp 9200–9206
93. Chen S et al (2022) Efficient and robust 2D-to-BEV representation learning via geometry-guided Kernel transformer. Comput Res Repos (CoRR). arXiv:2206.04584
94. Gong S et al (2022) GitNet: geometric prior-based transformation for birds-eye-view segmentation. In: 17th European conference computer vision-ECCV 2022, Part I, Tel Aviv, Israel, October 23–27, 2022, Proceedings. Springer, pp 396–411
95. Jiang Y et al (2022) PolarFormer: multi-camera 3D object detection with polar transformers. Comput Res Repos (CoRR). arXiv:2206.15398
96. Bartoccioni F et al (2022) LaRa: latents and rays for multi-camera bird's-eye-view semantic segmentation. Comput Res Repos (CoRR). arXiv:2206.13294
97. Vora S et al (2020) PointPainting: sequential fusion for 3D object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 4604–4612
98. Wang C et al (2021) PointAugmenting: cross-modal augmentation for 3D object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 11794–11803
99. Piergiovanni AJ et al (2021) 4D-Net for learned multi-modal alignment. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 15435–15445
100. Xu D et al (2018) PointFusion: deep sensor fusion for 3D bounding box estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 244–253
101. Liang M et al (2018) Deep continuous fusion for multi-sensor 3D object detection. In: Proceedings of the European conference on computer vision (ECCV), pp 641–656
102. Ku J et al (2018) Joint 3D proposal generation and object detection from view aggregation. In: 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 1–8
103. Li Y et al (2022) Unifying voxel-based representation with transformer for 3D object detection. Comput Res Repos (CoRR). arXiv:2206.00630
104. Chen Z et al (2022) AutoAlign: pixel-instance feature aggregation for multi-modal 3D object detection. Comput Res Repos (CoRR). arXiv:2201.06493
105. Chen Z et al (2022) AutoAlignV2: deformable feature aggregation for dynamic multi-modal 3d object detection. Comput Res Repos (CoRR). arXiv:2207.10316

106. Nabati R, Qi H (2021) CenterFusion: center-based Radar and camera fusion for 3D object detection. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision (WACV), pp 1527–1536
107. Harley AW et al (2022) Simple-BEV: what really matters for multi-sensor BEV perception? . Comput Res Repos (CoRR). arXiv:2206.07959
108. Dosovitskiy A et al (2017) CARLA: an open urban driving simulator. In: Conference on robot learning (CoRL). PMLR, pp 1–16
109. van Dijk T, de Croon G (2019) How do neural networks see depth in single images? In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 2183–2191
110. Saha A et al (2022) "The pedestrian next to the lamppost" adaptive object graphs for better instantaneous mapping. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 19528–19537
111. Qin Z et al (2022) UniFormer: unified multi-view fusion transformer for spatial-temporal representation in bird's-eye-view. Comput Res Repos (CoRR). arXiv:2207.08536
112. Andreas G et al (2013) Vision meets robotics: the kitti dataset. Int J Robot Res 32(11):1231–1237
113. Caesar H et al (2020) nuScenes: a multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 11621–11631
114. Sun P et al (2020) Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 2446–2454
115. Chang M-F et al (2019) Argoverse: 3D tracking and forecasting with rich maps. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 8748–8757
116. Wilson B et al (2021) Argoverse 2: next generation datasets for self-driving perception and forecasting. Comput Res Repos (CoRR). arXiv:2301.00493
117. Houston J et al (2021) One thousand and one hours: self-driving motion prediction dataset. In: Conference on robot learning (CoRL). PMLR, pp 409–418
118. Patil A et al (2019) The H3D dataset for full-surround 3D multi-object detection and tracking in crowded urban scenes. In: 2019 international conference on robotics and automation (ICRA). IEEE, pp 9552–9557
119. Gählert N et al (2020) Cityscapes 3D: dataset and benchmark for 9 dof vehicle detection. Comput Res Repos (CoRR). arXiv:2006.07864
120. Liao Y et al (2022) Kitti-360: a novel dataset and benchmarks for urban scene understanding in 2D and 3D. IEEE Trans Pattern Anal Mach Intell
121. Robotics H (2020) Horizon algorithms. https://en.horizon.ai/products/applications-algorithms/ Accessed 2022
122. HAOMO. haomo ai day. https://www.bilibili.com/video/BV1Wr4y1H7W7 Accessed 2022
123. PhiGent. Phigent: Technical roadmap. https://43.132.128.84/coreTechnology Accessed 2022
124. Mobileye (2020) Ces 2020 by mobileye. https://youtu.be/HPWGFzqd7pI Accessed 2020
125. Radosavovic I et al (2020) Designing network design spaces. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 10428–10436
126. He K et al (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 770–778
127. Tan M et al (2020) EfficientDet: scalable and efficient object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 10781–10790
128. Waymo (2020) Drago anguelov - machine learning for autonomous driving at scale. https://youtu.be/BV4EXwlb3yo Accessed 2020
129. Zhou T et al (2017) Unsupervised learning of depth and ego-motion from video. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 1851–1858

130. Gordon A et al (2019) Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp 8977–8986
131. Kim T, Adalı T (2003) Approximation by fully complex multilayer perceptrons. Neural Comput 15(7):1641–1666

Chapter 11
Road Environment Perception for Safe and Comfortable Driving

Sicen Guo, Yu Jiang, Jiahang Li, Dacheng Zhou, Shuai Su, Mohammud Junaid Bocus, Xingyi Zhu, Qijun Chen, and Rui Fan

Abstract With the ongoing evolution of autonomous driving technology, road environment perception systems have become a significant focus of research. However, there is currently a paucity of comprehensive survey articles that provide a systematic overview of state-of-the-art (SoTA) computer vision techniques and vibration methods for road defect detection, particularly with regards to deep learning methods. This chapter aims to fill this gap by describing the sensing technologies for vision-based

S. Guo · J. Li · D. Zhou · S. Su · Q. Chen · R. Fan (✉)
The Robotics and Artificial Intelligence Laboratory (RAIL), The Department of Control Science and Engineering, Frontiers Science Center for Intelligent Autonomous Systems, and State Key Laboratory of Intelligent Autonomous Systems, Tongji University, Shanghai 201804, People's Republic of China
e-mail: rfan@tongji.edu.cn

S. Guo
e-mail: guosicen@tongji.edu.cn

J. Li
e-mail: 2230745@tongji.edu.cn

D. Zhou
e-mail: zhoudacheng20@tongji.edu.cn

S. Su
e-mail: sushuai@tongji.edu.cn

Q. Chen
e-mail: qjchen@tongji.edu.cn

Y. Jiang
CTO Office, ClearMotion, Inc. Billerica, MA 01821, USA
e-mail: yjiang@clearmotion.com

M. J. Bocus
University of Bristol, Bristol, United Kingdom
e-mail: junaid.bocus@bristol.ac.uk

X. Zhu
The Department of Roads and Airport Engineering and Key Laboratory of Road and Traffic Engineering of Ministry of Education, Tongji University, Shanghai 200092,
People's Republic of China
e-mail: zhuxingyi66@tongji.edu.cn

R. Fan et al. (eds.), *Autonomous Driving Perception*, Advances in Computer Vision
and Pattern Recognition, https://doi.org/10.1007/978-981-99-4287-9_11

and vibration-based road environment data acquisition, summarizing several public datasets for pothole and crack detection, and providing a comprehensive review of SoTA road defect detection algorithms. Additionally, this chapter also discusses the core competencies of autonomous vehicle software systems, such as planning and control. Finally, we offer a glimpse into the future of autonomous driving systems, envisioning the integration of both vision and motion sensors. We believe that this survey will act as a helpful guide for advancing road defect detection technology, providing strategic advice and practical guidance to those involved in developing such systems.

11.1 Introduction

Over the past few decades, the global vehicle population has significantly increased, resulting in greater pressure on road surfaces and a corresponding rise in road defects. These harsh road conditions not only damage vehicles but also impede smooth driving and contribute to traffic accidents. In fact, one-third of the approximately 33,000 yearly crashes are attributed to poor road conditions [1]. A study by The Pacific Institute for Research and Evaluation, which analyzed crash data from various government agencies over 18 months, found that road issues such as potholes and icy conditions on highways are responsible for over 42,000 fatalities annually [2].

Road defects such as potholes can significantly increase the risk of accidents and vehicle repair costs, particularly in poor or substandard conditions. The impact of hitting a pothole can cause damage to the vehicle's suspension and can also result in the driver losing control of the vehicle. When a vehicle's tyres encounter a pothole, the forces acting on the tyres become imbalanced, leading to weight shifting to the lower tyres and causing the vehicle to tilt. As the tyre strikes the pothole's edge, a concentrated force impacts the tyre directly, which can result in tyre deformation, breakage, and even rim bending. These impacts not only cause damage to the vehicle, but also interfere with the driving experience, making it difficult to drive the vehicle in a straight line.

Autonomous driving, also known as robot-assisted driving, has garnered significant research attention in recent years. The utilization of autonomous vehicles is projected to have a crucial impact on the future of urban transportation systems by enhancing safety, productivity, accessibility, road efficiency, and environmental sustainability. The pursuit of fully autonomous vehicles has instigated intense competition among leading companies, including Google, Toyota, and Ford, to advance their respective autonomous robotic vehicle concepts [4–6]. Autonomous ground vehicles have been recognized for their prospects to provide crucial benefits to society, such as reducing the frequency of road accidents caused by human factors, optimizing fuel consumption, and enhancing driving convenience.

Despite the significant milestone that autonomous vehicles represent in the advancement of intelligent automation, concerns about their safety and reliability continue to be major issues. Autonomous vehicles require robust detection of their surroundings and interpretation of a wide range of contextual information to avoid collisions, particularly from a safety perspective. Road environment perception capabilities play a crucial role in ensuring the safe deployment of autonomous vehicles. For example, ClearMotion [7] has developed a cloud software called RoadMotion. The collection and analysis of road surface mapping data by autonomous vehicles enable the vehicle subsystems to make informed decisions in real-time. This data facilitates safer control of steering, stopping, starting, and absorbing shocks based on the actual road conditions. This requires the vehicle sensors to recognize various road defects, as well as road signs such as lane lines, speed humps, and pedestrian crossing. In the contemporary realm of autonomous driving, the detection and localization of roadway defects have garnered paramount importance in ensuring safe and efficient transportation. To this end, sophisticated algorithms have been developed, which predominantly employ a global mapping approach, harnessing the power of Global Positioning Systems (GPSs), Light Detection and Ranging (LIDARs), and cameras. Such algorithms meticulously detect and precisely localize a given defect within the vehicle's coordinate system, subsequently transforming this information into global coordinates. Leveraging this accurate spatial information, the algorithm scrutinizes the global map and identifies the optimal path for the vehicle, guaranteeing smooth navigation on the road [8, 9].

In the last two decades, the field of road perception has witnessed a remarkable transformation, marked by the advent of advanced acquisition and detection techniques. Yet, most recent surveys in this field do not comprehensively cover the SoTA advancements in computer vision, including 3-D point cloud modeling, segmentation, and machine/deep learning. Furthermore, the relatively nascent field of vibration-based road detection remains largely unexplored. In light of this, we present a meticulous and comprehensive review of the SoTA road perception systems, discussing both computer vision-based and vibration-based road defect detection algorithms. To facilitate a comprehensive understanding of this domain, we begin by providing a systematic overview of available systems and methods, as illustrated in Fig. 11.1. We begin by providing a comprehensive introduction to vision-based road imaging methods (see Sect. 11.2.1), followed by an introduction of vibration-based road environment acquisition techniques (see Sect. 11.2.2). We then introduce the road defect datasets obtained through the aforementioned methodologies (see Sect. 11.3). To ensure a clear and coherent presentation, we categorize road defect detection algorithms into two main groups: vision-based (see Sect. 11.4.1) and vibration-based (see Sect. 11.4.2). Subsequently, we describe how road information can be utilized in autonomous driving systems for vehicle planning and control (see Sect. 11.5). Finally, we conduct a critical evaluation of the limitations of existing methods and explore future research directions in this fast-evolving field (see Sect. 11.6). Our comprehensive survey promises to equip researchers and practitioners alike with a nuanced and in-depth understanding of the latest developments in road perception systems.

11.2 Sensing Technologies for Road Environment Perception

11.2.1 Vision Sensors

11.2.1.1 2-D Imaging

The collection of road data through 2-D imaging methods, such as digital imaging or digital image acquisition, was first initiated in the early 1990s [10, 11]. However, the limitations of 2-D images make it difficult to effectively depict the physical structure of roads [12]. During that time, cameras and range sensors were commonly used as the primary sensing devices for acquiring road information. Nonetheless, image segmentation algorithms that are applied to gray-scale/RGB road defect images can

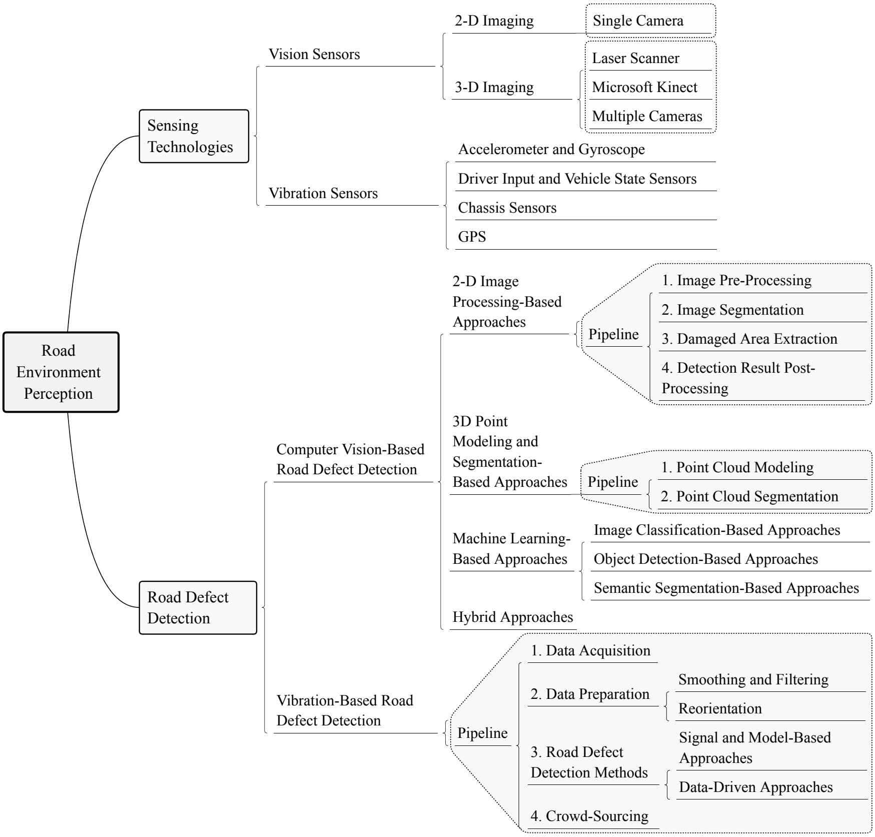

Road Environment Perception
Sensing Technologies
Vision Sensors
2-D Imaging
Single Camera
3-D Imaging
Laser Scanner
Microsoft Kinect
Multiple Cameras
Vibration Sensors
Accelerometer and Gyroscope
Driver Input and Vehicle State Sensors
Chassis Sensors
GPS
Road Defect Detection
Computer Vision-Based Road Defect Detection
2-D Image Processing-Based Approaches
Pipeline
1. Image Pre-Processing
2. Image Segmentation
3. Damaged Area Extraction
4. Detection Result Post-Processing
3D Point Modeling and Segmentation-Based Approaches
Pipeline
1. Point Cloud Modeling
2. Point Cloud Segmentation
Machine Learning-Based Approaches
Image Classification-Based Approaches
Object Detection-Based Approaches
Semantic Segmentation-Based Approaches
Hybrid Approaches
Vibration-Based Road Defect Detection
Pipeline
1. Data Acquisition
2. Data Preparation
Smoothing and Filtering
Reorientation
3. Road Defect Detection Methods
Signal and Model-Based Approaches
Data-Driven Approaches
4. Crowd-Sourcing

Fig. 11.1 A systematic overview of road environment perception systems created based on our previous taxonomy introduced in [3]

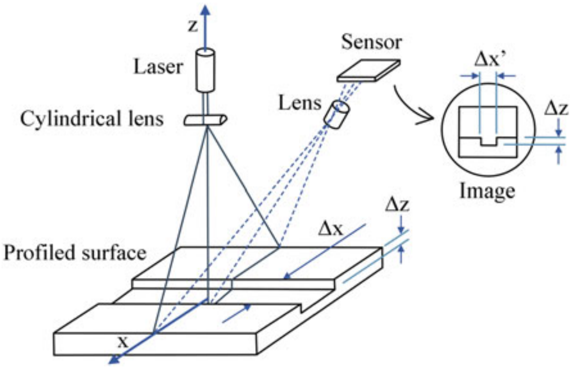


Fig. 11.2 Laser triangulation [19]. By placing the sensor at a predetermined distance from the laser's light source, the reflection angle of the laser illumination can be calculated, enabling the derivation of precise 3-D information about the road surface

be easily influenced by a wide range of factors, including unfavorable illumination conditions [13]. To overcome these limitations and simultaneously enhance the accuracy of road defect detection, researchers have increasingly turned to 3-D imaging technologies [12, 14, 15].

11.2.1.2 3-D Imaging

The first successful use of 3-D imaging methods to collect road surface data was in 1997 [16]. This section aims to provide a comprehensive discussion of the contemporary 3-D imaging technologies that are currently being employed for the purpose of acquiring road data.

Laser scanning is a widely adopted imaging method for precise 3-D road surface data acquisition [17]. Nonetheless, it requires laser sensors to be mounted on dedicated detection vehicles like the Georgia Institute of Technology Sensing Vehicle [18]. As a result, this method is not widely used due to the high cost of equipment acquisition and long-term maintenance expenses [15] (Fig. 11.2).

Microsoft Kinect [13] was originally developed for motion sensing games on the Xbox-360 platform, but have since been utilized for a variety of other applications. These sensors are equipped with various components such as an infrared (IR) sensor/camera, an IR emitter, an RGB camera, accelerometers, a tilt motor for motion tracking capabilities, and microphones [17]. Microsoft Kinect is a ideal technology for road surface imaging when installed on a vehicle, with a working range of 800–4000 mm. Previous studies have used Kinect sensors for 3-D road data acquisition and defect detection [13, 20, 21]. However, the reliability of Kinect sensors can be challenged by the IR saturation issue, particularly when exposed to direct sunlight [22] (Fig. 11.3).

Multiple images captured from different views can also be used to reconstruct the 3-D geometry of a road surface [12, 24]. This technique is based on the theory of *multi-view geometry* [25]. The initial proposal for an automatic 3-D reconstruction and modeling system for road signs was introduced in [26]. The authors employed

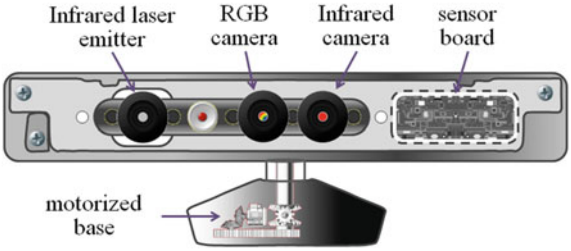


Fig. 11.3 Microsoft Kinect sensor [23]

a multi-view constrained 3-D reconstruction approach that incorporates geometric constraints on the shape of the road signs to obtain an optimal 3-D silhouette, achieving centimeter-level accuracy.

11.2.2 Vibration Sensors

Vibration-based road defect detection methods utilize non-vision-based sensors, such as inertial sensors (e.g., accelerometer and gyroscope sensors), vehicle speed, vehicle suspension states, drivers' inputs, etc. Vibration-based road defect detection methods have gained popularity due to their cost-effectiveness and relatively light computational burden. Despite being tied to vehicle dynamics and potentially affected by vehicle parameters, these methods provide more robust measurements as they are less influenced by weather or lighting conditions.

One of the earlier attempts at using mobile inertial sensors to detect potholes was conducted by a research group in CASIL at MIT in 2008. In [28], Eriksson et al. created a system known as *pothole patrol* that used inertial sensors and GPS sensors from mobile devices deployed to a fleet of seven taxis in the Boston area. The orientation of each mobile sensor was fixed in the vehicle once installed. Therefore, a one-time calibration can be performed for each vehicle right after sensor installation. In [29], researchers at Microsoft Research India presented *Nericell*, an add-on system that can be carried together with smartphones by users in the normal course. Since the orientation of the mobile sensors may vary when placed inside a vehicle, a practical reorientation approach was developed to virtually align the smartphone's three axes with body frame of the car. In 2011, Perttunen et al. collected accelerometer data at 38Hz and GPS data at 1Hz from a Nokia N95 8GB mobile phone, and they were able to detect road defects by training and deploying an SVM detector [30]. Later, with the increasing programmability and sensor accessibility of smartphones, researchers started developing applications directly on smartphones to detect potholes and other types of road defects using built-in smartphone sensors (see Table 11.1 for a list of possible sensors or signals that can be used for road defect detection). Android smartphones with accelerometers were used in [31] for real-time pothole detection. Wu et al. [32] developed a customized Android application and the smartphone can be mounted without being fixed to the back seat. In [33], the authors proposed a

Table 11.1 Different types of smartphone sensor data for road defect detection

Signal/sensor name	Type	Unit	Description
Acceleration	Measured	m/s^2	Raw accelerometer measurements
Gyroscope	Measured	deg/s	Raw gyroscope measurements
Linear acceleration	Derived	m/s^2	Accelerometer measurements excluding gravity
Magnetometer	Measured	μT	Ambient geomagnetic field measurements
Gravity	Derived	m/s^2	Gravitational acceleration
Rotation	Derived	rad	Orientation of the sensor
GPS	Measured	deg	Location information of the sensor

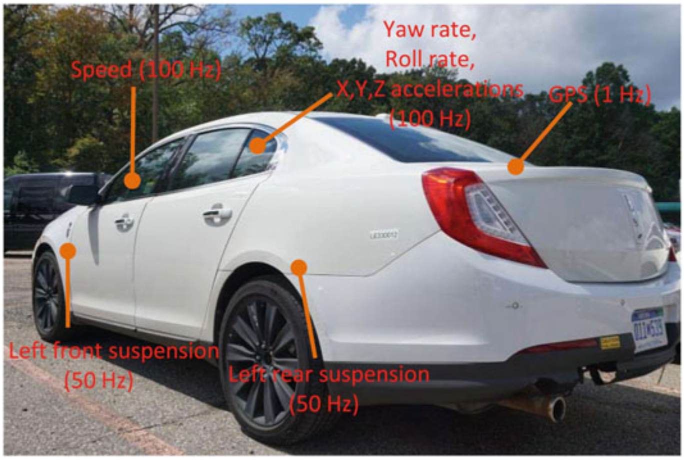


Fig. 11.4 In-vehicle sensors for vibration signals collection [27]

model called *smart-patrolling* on Android phones to crowd-source data for pothole detection. A system called *Wolverine* was introduced in [34] and was implemented on a Google Nexus S. A road monitoring system was developed using a Nokia Lumia 820 mobile device to detect road defects from measured gyroscope data around gravity rotation and accelerometer sensor data [35]. Recently, Rajput et al. placed smartphones in public transportation buses with dense connectivity to achieve large-scale road monitoring [36]. While most research studies rely on accelerometer sensing as the primary indicator for road defects, the *RoadMonitor* system [35] proposed a cross-validation methodology for the detection and severity assessment of road bumps using gyroscope sensors.

Although smartphones can be used as vibration sensors, vehicle built-in sensors generally provide richer content. In [37], researchers constructed a signal set con-

Table 11.2 Pothole dataset

Pothole dataset	Sensors	Color/gray	Dataset size	Resolution
[43]	GoPro Hero 3 + camera	Color	628	2,760 × 3,680
[44]	–	Color	3,227 (2,475 for training and 752 for testing)	720 × 1,280
[45]	Smartphone mounted on a car	Color	9,053	600 × 600
[22]	ZED stereo camera	Color	67	800 × 1,312
[46]	ZED stereo camera	Color	600 (240 for training, 180 for validation and 180 for testing)	400 × 400
UDTIRI	Collected from the Internet or taken manually	Color	1,000 (600 for training, 100 for validation and 300 for testing)	Variety of resolutions

taining eighteen different signals for road surface defect detection. The signal set includes measurements from the Inertial Measurement Unit (IMU) located at the center of gravity of the vehicle, as well as measurements from the shock absorber and spindle signals. Similarly, in [27], vehicle measurements, including suspension position reading, were collected to conduct state estimation based on Jump-Diffusion Processes (JDPs). The sensor data collection setup utilized by [27] is presented in Fig. 11.4.

11.3 Public Datasets

Data plays a crucial role in deep learning algorithms. While there are many large-scale datasets available for general-purpose object detection, the small number and size of existing datasets have become a bottleneck for the development of defect detection systems. This section provides an overview of common road imaging datasets, as well as pothole and crack datasets, which are summarized in Tables 11.2 and 11.3.

11.3.1 Road Imaging Datasets

Road scene imaging datasets are crucial for road defect detection. While datasets such as KITTI [38] and Cityscape [39], which are widely used for computer vision

Table 11.3 Crack dataset

Crack dataset	Sensors	Color/gray	Dataset size	Resolution
[48]	Nine CMOS line camera	Gray	38,000	6,144 × 1,024
[49]	Aviiva SM2 CL 1010 Camera	Color	–	1,024 × 1,714
[50]	DSLR camera (Nikon D5200)	Color	332	277 images with 4,928 × 3,264 and 55 images with 5,888 × 3,584
[51]	16 MP Nikon digital camera	Color	56,092	256 × 256,64 × 64
[52]	Collected from the Internet or taken manually	Color	200	Variety of resolutions
[53]	Digital camera	Color	1,328	4,032 × 3,016
[54]		Color	600	224 × 144,976 × 640

tasks, also include road scenes, they provide less information about the road surface and are primarily focused on street scenes. Specifically, there are datasets dedicated to road surface defect detection, such as those used for 3D reconstruction of road surfaces [12, 15].

For the road surface 3-D reconstruction, [12] creates three datasets[1] (91 stereo image pairs). Datasets 1 and 2 focus on road scenes, while dataset 3 assists researchers in evaluating with sample models. Each dataset includes both uncalibrated and calibrated left and right images. The calibrated images are serve as estimation of the disparity map, while the uncalibrated images and calibration parameters are used to reconstruct the 3-D road geometry. Additionally, they have created a road pothole 3-D geometry reconstruction dataset[2] [15]. The accuracy of the road pothole 3-D geometry reconstructed using stereo vision technology is 2.23 mm.

Toronto-3-D[3] [40] is a sizable dataset of urban outdoor point cloud collected by a Mobile Laser Scanning (MLS) system in Toronto, Canada. It contains approximately 78.3 million points and covers around 1 km of road.

The RadarScenes dataset[4] [41] consists of recordings collected from a measurement vehicle equipped with four automotive radar sensors and a documentary camera facing the front. It includes point cloud data collected by the radar sensors and seman-

[1] https://github.com/ruirangerfan/road_surface_3d_reconstruction_datasets.

[2] https://github.com/ruirangerfan/rethinking_road_reconstruction_pothole_detection.

[3] https://www.kaggle.com/datasets/priteshraj10/point-cloud-lidar-toronto-3d.

[4] https://www.kaggle.com/datasets/aleksandrdubrovin/the-radarscenes-data-set.

tic annotations on a point-wise level. The dataset was recorded in Ulm, Germany over a period of three years from 2016 to 2018 and spans over 4 hours in length.

RoadSaW [42] is a recently introduced dataset that aims to facilitate the estimation of road surface and wetness. The dataset contains 720,000 bird's-eye-view patches captured on a test track, high-resolution videos, and accurate synchronized water film height measurements. The dataset covers various surface types such as asphalt, concrete, and cobblestone. The FLIR Blackfly 3.2 MP, 30 fps camera is mounted behind the truck's windscreen at a height of 2.66 meters. The primary dataset consists of 250 videos recorded on five different days and is split almost evenly among the training (70%), validation (20%), and test (10%) sets.

11.3.2 Pothole Datasets

[45] presented a sizable road defect dataset,[5] containing 9,053 color road images with a resolution of 600 × 600 pixels collected in Japan. The images, which show 15,435 road defects, were captured using a smartphone installed on a car in diverse weather and lighting conditions.

A comprehensive dataset[6] for instance-level pothole detection has been compiled by Rath et al. [43]. The dataset comprises a training set, a test set, and an annotation CSV file. The training set includes 2,658 color images of pothole-free roads and 1,119 color images of roads with potholes, while the test set contains 628 color images. These images, with a resolution of 2,760 × 3,680 pixels, were taken by a GoPro Hero 3+ camera, with a 0.5-s time-lapse mode, while the car moved at a 40 km/h average speed and scanned the road surface. The dataset size is approximately 2.70 GB.

The first multi-modal road pothole detection dataset[7] to facilitate the development and evaluation of more advanced and accurate pothole detection algorithms is presented in [22]. This dataset includes 55 groups of data, consisting of (1) RGB images, (2) subpixel disparity images, (3) transformed disparity images, and (4) pixel-level pothole annotations, with an image resolution of 800 × 1,312 pixels. Additionally, the Pothole-600 dataset[8] [46] provides two types of vision sensor data: color images and transformed disparity images. The transformed disparity images are obtained by applying the disparity transformation algorithm [47] to dense subpixel disparity images estimated using the stereo matching algorithm [12].

UDTIRI dataset[9] focuses on object detection, semantic segmentation, and instance segmentation tasks in the field of pothole detection. The dataset consists of images collected from online search engines as well as a collection of real images captured

[5] https://github.com/sekilab/RoadDamageDetector.

[6] https://kaggle.com/sovitrath/road-pothole-images-for-pothole-detection.

[7] https://github.com/ruirangerfan/stereo_pothole_datasets.

[8] https://sites.google.com/view/pothole-600.

[9] https://www.udtiri.com/.

in China to increase the diversity of image sources. The potholes in the UDTIRI dataset vary in scale, with the smallest pothole covering 0.3% of the image area and the largest covering 92%. On average, potholes cover 11.2% of the image area. The dataset also provides pixel-level statistics, with the average pothole bounding box having a width of 500 pixels, a height of 248 pixels, and an average area of 224,892 pixels.

11.3.3 Crack Datasets

Cha et al. proposed a crack dataset[50], which was obtained using a hand-held DSLR camera (Nikon D5200) in a complex building at the University of Manitoba. The dataset comprises 332 raw images, including 277 images with a resolution of 4,928 × 3,264 pixels and 55 images with a resolution of 5,888 × 3,584 pixels. The objects in the images were located at distances ranging from approximately 1.0 to 1.5 m, although some images were taken at distances below 0.1 m for testing purposes. Additionally, the lighting intensities in the images were significantly different. The dataset was divided into training and validation subsets, consisting of 277 images, and a testing subset consisting of 55 images.

Zhang et al. presented a dataset [51] consisting of 56,092 images of manually annotated concrete bridge decks that include cracks of varying widths, ranging from as narrow as 0.06 mm to as wide as 25 mm. The images have a size of 256 × 256 pixels and are subdivided into smaller sub-images with dimensions of 64 × 64 pixels to improve detection accuracy. From an initial pool of 4,800 images, 4,300 images are partitioned into 17,200 sub-images, and blurry images or those with corner cracks are excluded. As a result, a total of 16,789 images are used as the dataset for this study. Moreover, to evaluate the classifier's generalization ability, 500 bridge deck images are randomly combined to form 20 images with a size of 1,280 × 1,280 pixels for testing.

Choi et al. introduced a dataset [52] comprising of 200 digital images sourced from the internet or captured manually. Each image was captured under distinct conditions such as varying distances, lighting intensity, field of view (FOV), and image quality. The images have spatial dimensions between 513 and 1,920 pixels in any axis. The minimum size of the images is 513 × 513 pixels, while the maximum size is approximately that of high definition images, which is 1,920 × 1,080 pixels.

A digital single lens reflex camera was used to capture a total of 409 photos (4,032 × 3,016 pixels) in a tunnel in Huzhou, Zhejiang Province, China, under various lighting circumstances [53]. Since training convolutional neural networks (CNNs) on large images may cause GPU memory overflow, the large images were cropped to a size of 512 × 512 pixels. The resulting dataset contains 919 cropped images, which were randomly divided into a training set and a test set with a ratio of 4:1. The training set was used to optimize the model parameters, and the test set was used to evaluate the model's accuracy.

11.4 Road Defect Detection

11.4.1 Computer Vision-Based Road Defect Detection

An overview of the taxonomy of cutting-edge computer vision-based road defect detection algorithms is presented in Fig. 11.1. Classical 2-D image processing-based algorithms employ explicit programming to process road RGB or disparity/depth images, involving various techniques such as image enhancement, compression, transformation, and segmentation [55]. On the other hand, segmentation-based methods and 3-D road point cloud modeling involve fitting geometric models, such as planar or quadratic surfaces, to the road point cloud and then segmenting the cloud by comparing observed and fitted surfaces. Machine/deep learning-based algorithms utilize techniques such as image classification, object recognition, or semantic segmentation to detect road potholes. Hybrid methods combine two or more methods to increase overall performance. Recent studies have shown that deep learning approaches hold potential for road defect detection [56–58].

11.4.1.1 2-D Image Processing-Based Approaches

The classical 2-D image processing techniques have been widely studied for road defect detection, and they generally follow four fundamental steps: (1) image preprocessing, (2) image segmentation, (3) defect extraction, and (4) post-processing. Ryu et al. [59] proposed a classical 2-D image processing technique that uses a histogram shape-based thresholding technique [60] to binarize the grayscale road data. The segmented images then undergo post-processing operations to eliminate noise, including median filter and morphology operations. The median filter smooths the image while preserving edges, and morphology operation helps in reducing redundant noise. Finally, the histogram of pixel intensity is evaluated to identify the regions of interest containing road defects.

Researchers have developed various image segmentation algorithms for detecting road defects using depth/disparity images, which have led to higher accuracy in intelligent road condition assessment systems [13, 18, 61]. For instance, [13] used the Microsoft Kinect sensor to capture depth images of pavements, which were segmented using the wavelet transform algorithm [62]. Tsai et al. introduced a method [18] that utilizes a quite precise laser scanner placed on a road inspection vehicle to identify road defects via depth images. Initially, a high-pass filter is applied to the depth images to make the depth values of undamaged road pixels more similar. The processed depth images are then segmented using the watershed method [63] to detect road defects. Similarly, Fan et al. processed dense road disparity images using the disparity transformation algorithm [61], which made road defects more distinct. They then applied a histogram-based thresholding technique to the converted disparity images to identify road faults.

11.4.1.2 3-D Point Modeling and Segmentation-Based Approaches

In recent years, 3-D point clouds have shown great promise for road pothole detection and segmentation. The techniques developed for processing 3-D road point clouds generally follow two procedures [64, 65]. Firstly, the 3-D point cloud is interpolated into an explicit geometric model, typically a planar or quadratic surface, in order to obtain a structured representation of the road surface. Secondly, the 3-D point cloud is segmented by comparing it with the geometric interpolation model, which enables the identification of road potholes with greater accuracy and efficiency [64, 65].

An example of utilizing the 3-D point cloud technique for road defect detection and segmentation is shown in [64]. This method employs least-squares fitting to approximate the dense 3-D road point clouds with quadratic surfaces, and calculates the elevation difference between the observed surfaces and the fitted surfaces to extract road defects (potholes) effectively.

The accuracy of least-squares fitting will be significantly impaired by the outliers in input data [22]. To address this challenge, [66] introduced the robust least-squares approximation with bi-square weights for modeling road point clouds. Reference [47] further improved the robustness of quadratic surface fitting by employing the random sample consensus (RANSAC) algorithm [67]. In addition, [22, 68] improved the accuracy of road defect detection by introducing surface normal information into the quadratic surface fitting procedure. These techniques have demonstrated their effectiveness in accurately modeling road surfaces and detecting road defects with a high level of precision.

Compared to other approaches, 3-D point cloud modeling and segmentation methods are relatively uncommon, mainly because real-world roads are often highly irregular and uneven, making these techniques sometimes impractical.

11.4.1.3 Machine Learning-Based Approaches

The year 2012 marked a significant turning point in the field of machine learning, as neural network-based methods surpassed traditional hand-designed feature approaches in the ImageNet classification competition. Since then, the development of machine learning methods has progressed rapidly, resulting in many advanced algorithms for tasks such as image classification, object detection, and semantic segmentation, which consistently outperform traditional methods. The use of deep learning methods has become standard practice in many fields due to their ability to provide broader coverage and higher performance for solving more complex problems. In this section, we will review road defect detection applications of excellent learning-based methods, including image classification, object detection, and semantic segmentation.

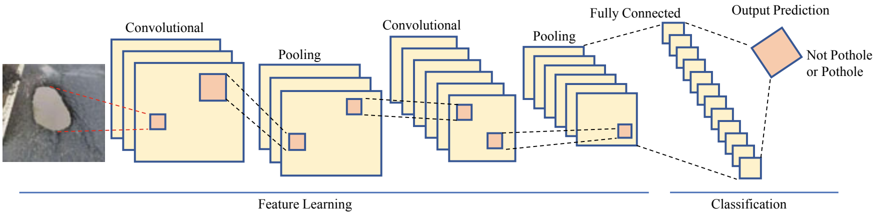


Fig. 11.5 Architecture of image classification progress [69]

Image Classification Approaches

The increasing availability of large training datasets and computational resources has facilitated the extensive use of Deep Convolutional Neural Networks (DCNNs) in detecting road defects. Compared to classical support vector machine (SVM) approaches, DCNNs can learn more abstract and hierarchical visual features, resulting in better detection performance [61]. Various DCNN-based approaches have been proposed for road defect detection [69–72], and they have been extensively studied [69, 71]. A study by Pereira et al. [69] presented a DCNN architecture consisting of four convolutional-pooling layers and a fully connected (FC) layer. The model was able to accurately classify defect and non-defect images in road data obtained from Timor-Leste. Similarly, Ye et al. [17] proposed a DCNN for road pothole classification [71]. The network consists of a pre-pooling layer, three convolutional-pooling layers, a sigmoid layer, and two fully connected layers. The pre-pooling layer is tailored to eliminate features that are irrelevant to road potholes, resulting in improved classification performance of road images. The experimental results demonstrate that the network can effectively identify road defects in different lighting conditions.

As shown in Fig. 11.5, image classification involves two steps: generating a feature map and performing feature learning through convolutional networks. The final output layer is a fully connected layer that flattens the input received from preceding layers and provides the result. Different classification networks vary in the architecture of the feature learning network used. For example, Bhatia et al. [72] proposed a DCNN based on the widely used residual network architecture [73]. Their proposed model was tested on thermal road images collected in challenging weather conditions and outperformed previous methods [59, 70, 74] in accurately classifying road images into pothole and non-pothole images.

Compared to pothole classification, road crack detection has received more attention using classification networks, mainly due to the ability of CNNs to learn discriminative visual features without explicit feature engineering [75]. For instance, [76] introduced a CNN for road crack detection that takes RGB road images as input. The learned visual features are then mapped to a scalar value using a fully connected layer, representing the probability of the input image containing road cracks. Similarly, [56] proposed a deep CNN to classify road images as containing or not

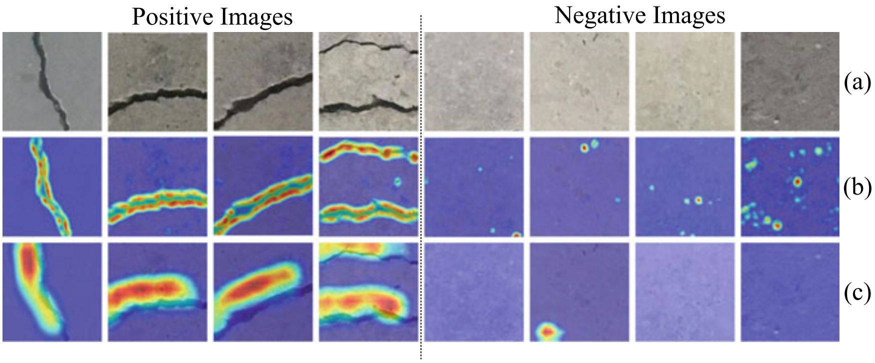


Fig. 11.6 Examples of Grad-CAM++ [79] results [80]: **a** road crack images, **b** the class activation maps of ResNet-50 [81], **c** the class activation maps of SENet [82]. The warmer the color is, the more attention the CNN pays

containing road cracks. Besides image classification, an image thresholding method was applied to segment the road images for pixel-level road crack detection.

A comprehensive evaluation of 30 SoTA CNNs was conducted by [77] for the purpose of road crack detection. The results of the experiments suggest that image classification is not a significantly challenging task for road crack detection, and the performance of deep CNNs in this area is highly similar. Furthermore, it was found that only 10,000 images are needed to train a well-performing CNN for this task. However, pre-trained CNNs often perform poorly on additional test sets, making unsupervised domain adaptation (UDA) [78] an important area of research that requires more attention. Additionally, observable artificial intelligence (AI) algorithms like Grad-CAM++ [79], have become essential for applications involving image classification to better understand CNN decision-making. Figure 11.6 shows an example of Grad-CAM++ results.

Object Detection Approaches

In 2015, Girshick introduced an advanced object detection algorithm named Fast R-CNN [83], which builds upon the R-CNN approach. This method employs CNNs to directly analyze the input image, producing a Convolutional Feature Map (CFM). Then, selective search is applied to identify region proposals from the CFM, and the proposals are reshaped to a fixed size using a Region of Interest (RoI) pooling layer before being fed into a fully connected layer. Finally, a softmax layer predicts the classes of the region proposals. However, the reliance of R-CNN [84] and Fast R-CNN [83] on selective search to generate region proposals is computationally expensive. To tackle this problem, Ren et al. presented Faster R-CNN [85], which introduces a region proposal network to learn the region proposals. Compared to R-CNN, Faster R-CNN is 250 times faster, and 11 times faster than Fast R-CNN.

Unlike the R-CNN series, the YOLO series takes a distinct approach to object detection. YOLOv1 [86] views the issue of object detection as a regression problem and divides the input image into an $S \times S$ grid, where each grid cell predicts B bounding boxes with their corresponding class probabilities and offsets. However, YOLOv1 [86] suffers from a high number of localization errors and relatively low recall. To address these issues, YOLOv2 [87] introduces several improvements, including the addition of batch normalization to all convolutional layers, fine-tuning the classification network at full resolution, and replacing the fully connected layers with anchor boxes for predicting bounding boxes.

Regarding the implementation of object detectors in road defect detection, several studies have utilized popular algorithms to detect road defects. For instance, Suong et al. [88] trained a YOLOv2 [87] model to recognize road potholes, while [89] utilized a Faster R-CNN [85] to accomplish the same goal. Reference [90] employed three kinds of different YOLOv3 [91] architectures for detecting road defects, while [92] introduced YOLOv3 [91] to detect road potholes from RGB images obtained by a car-mounted smartphone. The proposed algorithm has been implemented on a Raspberry Pi 2 model B using TensorFlow, and the reported results show a mean average precision (mAP) of 68.83% and an inference time of 10 ms. Moreover, in a recent study[10] [93], the performances of YOLOv2 [87] and Mask R-CNN [94] for road defect detection have been compared and analyzed. However, these methods use object detection networks that are not specifically designed for the task of road defect detection and are limited to providing instance-level predictions. Consequently, in recent years, semantic image segmentation has emerged as a more desirable technique for detecting road defects, as it can provide pixel-level predictions. It should be noted that Dhiman et al. (2019) compared the performances of YOLOv2 and Mask R-CNN for road defect detection, but these methods still lack the specificity of semantic image segmentation.

Semantic Segmentation Approaches

Semantic segmentation is an image processing technique for labeling each pixel in an image with a corresponding class [95]. In recent years, semantic segmentation has gained significant popularity for road defect detection [96]. Cutting-edge semantic image segmentation networks can be broadly classified into two categories based on the number of encoders used: (1) single-modal [57] segmentation networks that use only one kind of vision sensor data, such as RGB or depth images, and (2) data-fusion segmentation networks that use multiple types of vision sensor data, such as color and transformed disparity images [97] or color and surface normal information [98]. Both types of networks have been employed by researchers for the detection of road defects and defects, and they all have achieved successful outcomes.

Recent years have seen significant progress in the development of state-of-the-art solutions for semantic segmentation of road cracks. For instance, Choi et al. proposed

[10] https://vimeo.com/337886918.

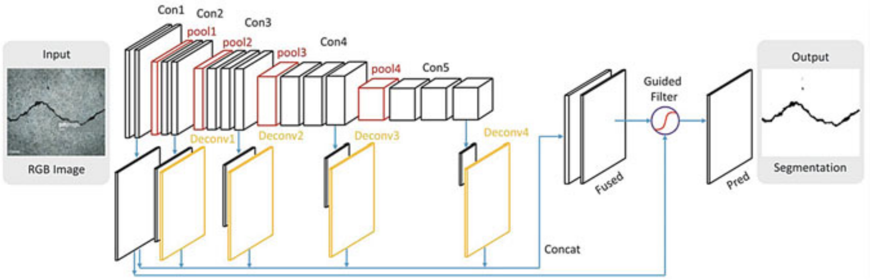


Fig. 11.7 DeepCrack architecture [58] (with permission for resue)

an approach for the segmentation of concrete cracks called SDDNet [52], which uses a combination of standard convolutions, Densely connected Separable convolution (DenSep), and a modified Atrous Spatial Pyramid Pooling (ASPP) module. Similarly, Liu et al. introduced DeepCrack [58], which is a robust CNN-based model that utilizes a deep hierarchical feature learning architecture. As illustrated in Fig. 11.7, DeepCrack avoids using fully connected layers and instead inserts side-output layers after the convolutional layers. Deep supervision is applied at every side-output layer to generate a final fused output that can capture features at different scales and levels. Finally, guided filtering is used to refine the resulting fused prediction with the first side-output layer.

In a recent study, Wang et al. [97] proposed a data-fusion convolutional neural network (CNN) for detecting road defects. The authors conducted a comprehensive evaluation of different modalities of visual features, including RGB images, disparity images, surface standard images [99], elevation images, HHA images, and transformed disparity images [61]. The experimental results showed that the transformed disparity image is the most informative visual feature for road defect detection. The implications of their findings are significant for improving the effectiveness of road defect detection techniques.

11.4.1.4 Hybrid Approaches

Hybrid approaches for road defect detection usually involve the integration of multiple algorithms mentioned earlier. By combining different methods, the system can achieve greater robustness and produce more accurate detection results.

Jog et al. proposed a hybrid approach for road pothole detection in 2012 [100]. The proposed method combines 2-D object recognition and 3-D road geometry reconstruction to accurately detect road potholes. High definition camera video frames are initially utilized for road pothole recognition, while sparse 3-D road geometry reconstruction is concurrently performed using the same video. The combination of these two modalities allows for accurate detection of road potholes and reduces

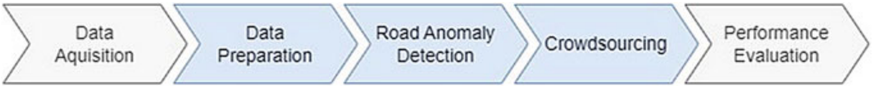


Fig. 11.8 Typical vibration-based road defect detection workflow

the number of falsely detected potholes. This hybrid method provides an effective solution for road defect detection.

In 2017, Kang et al. presented a hybrid road pothole detection system that utilizes 2-D LiDAR data and RGB road images for automated detection [101]. This system overcomes the limitations caused by electromagnetic waves and poor road conditions, and benefits from the complementary strengths of these two types of data. The 2-D LiDAR data provides road profile information, while the RGB road images offer road texture information. Two LiDARs are used to obtain accurate and large road area coverage. By combining the advantages of different vision data sources, this hybrid system is able to enhance the overall road pothole detection accuracy.

In 2019, [22] also proposed a hybrid network framework, which employs a hybrid framework that combines 2-D image processing and 3-D road surface modeling algorithms to enhance the detection of road potholes. Initially, potential undamaged road areas are extracted by applying disparity transformation (in Sect. 11.4.1.2) and Otsu's thresholding [102]. Subsequently, a quadratic surface is fitted to the original disparity image using the RANSAC algorithm to improve the robustness of surface fitting. The difference between the actual and fitted disparity images is then analyzed to detect potholes. This approach shows promising results in terms of improving road defect detection accuracy.

11.4.2 Vibration-Based Road Defect Detection

As discussed in Sect. 11.2.2, vibration-based techniques utilize non-vision sensors, such as inertial sensors, to gather vehicle data, which is then analyzed using detection algorithms to identify road defects. Vibration-based methods are generally more cost-effective than computer vision-based solutions, and have gained popularity among research groups in recent years. However, it is important to note that accurately estimating the dimensions of road defects is more challenging with vibration-based approaches compared to vision-based methods.

Typically, the workflow of road defect detection approaches follows the steps depicted in Fig. 11.8. It should be noted that the first step, i.e., data acquisition approaches, has been discussed in Sect. 11.2.2. The last step, performance evaluation, can employ any generic method and is not specific to vibration-based methods. Hence, our discussion here will concentrate on the three intermediate steps, namely, data preparation, road defect detection, and crowd-sourcing of detection results.

11.4.2.1 Data Preparation

Like other machine learning workflows, it is important to preprocess the raw data collected from mobile and built-in vehicle sensors to obtain a curated dataset before analysis. To ensure precise road defect detection, the raw sensor data typically undergoes smoothing and filtering. Additionally, if mobile sensors such as smartphones are employed, it is important to virtually re-orientate the sensor data to correspond with the vehicle's body frame.

Smoothing and Filtering

Smoothing and filtering of raw signals can be achieved by performing convolution in the time domain with finite impulse response (FIR) filters, such as a moving-average filter. Also, one can pass the raw signals through infinite impulse response (IIR) filters which are usually designed in the frequency domain. In both cases, low-pass filters (e.g., [103, 104]) and/or high-pass filters (e.g., [28]) are typically used to remove the high-frequency sensor noise and/or the low-frequency bias, respectively. Band-pass filters are frequently employed to isolate signal content within a specific frequency range. The speed signal is often utilized to accept or reject windows for further processing. Windows where the vehicle is moving slowly or not moving at all are commonly discarded to eliminate events like door slams [28]. When speed can only be inferred from GPS sensors, smoothing and estimation methods like Kalman filters are used to enhance speed estimation and remove outliers [30]. In particular, accelerometer measurements can be combined with GPS to overcome time-to-first-fix (TTFF) delays [105], and a technique called speed-dependency removal is used to segment the data. Speed-dependency removal is also addressed in [30].

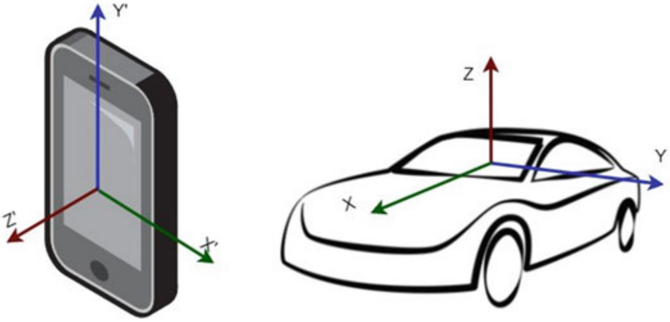


Fig. 11.9 Smartphone sensor coordinate and vehicle body coordinate

Reorientation

Another common practice in data pre-processing is reorientation, which is referred to as the process of reorienting sensor data values from a device coordinate system to the vehicle body coordinate system (see Fig. 11.9). This challenge exists in almost all smartphone-based solutions since the displacement of a smartphone in a vehicle can be arbitrary. The development of *Nericell* by Mohan et al. involved extensive investigation into this issue [29]. They realized that while it is theoretically possible to infer the angles of rotation about each axis, such a framework yields multi-factor trigonometric equations that are complex to solve. Alternatively, they resorted to an equivalent framework based on Euler angles. Under this framework, a pre-rotation about Z and a tilt angle about Y can be estimated from a short window of data collected from the steady motion of the vehicle. Then, a post-rotation angle, again about Z, can be derived from the braking and acceleration of a vehicle traveling in a straight line. These three Euler angles can easily represent any orientation, and this practical reorientation approach has been quickly adopted and extended by other research studies (e.g., [33, 106]). More reorientation strategies can be found in [107].

11.4.2.2 Road Defect Detection Methods

In the literature, there are various methodologies reported for road defect detection using vibration signals. Generally, these algorithms can be divided into two categories: signal-based and data-driven approaches. The signal-based methods aim at filtering or estimating signals with physical meanings, and identifying outliers by imposing metrics and thresholds on them, whereas the data-driven approaches focus more on extracting distinctive information from the data and training machine learning classifiers to detect road defects.

Signal and Model-Based Approaches

The researchers at CSAIL, MIT, developed *Pothole Patrol* [28] which utilized a threshold on Z-peak (vertical peak acceleration) along with longitudinal acceleration and vehicle speed to filter potential defects. They trained a simple parameterized model using five different types of manually selected road events. The system reported a 0.2% false positive rate for pothole detection. The *Nericell* system presented by Microsoft Research India utilized Windows phones carried by users to perform rich sensing. The authors used empirical calibration to determine parameters and thresholds for detecting bumps and defects based on vertical acceleration. The system was evaluated using data collected from Bangalore and Seattle, and reported promising results.

In [31], the authors explored four different metrics on accelerometer sensor data for pothole detection. The first three metrics focused on the peak, standard deviation, and changing rate of the vertical acceleration. The fourth metric, called G-ZERO, detected the moment when the acceleration measurements in all three axes are close to zero. This was based on the assumption that when a vehicle enters a pothole, it experiences a temporary free fall. It is noteworthy that reorientation was not required with the G-ZERO approach. The authors reported a high detection accuracy of up to 90% using these simple metrics.

In 2015, Li et al. analyzed pothole characteristics, and broke down the dynamics of hitting a pothole into different stages [108]. A multi-phase dynamic model was created as a switching system with three modes to capture the pothole-hitting response. They compared the results with F-Tire in a simulation environment and demonstrated that their model is capable of generating similar results and trends as in F-Tire. A detection algorithm was proposed by recursively computing the confidence of each mode with Bayesian estimation and Unscented Kalman Filtering. In [27], a state estimation framework was developed to detect road defects. These measurements from multiple vehicle state sensors were studied with both a full-vehicle model and half-model, and an optimal observer based on jump-diffusion processes (JDPs) was constructed to estimate known state and input signals, such as the road velocity input. In this work, it has also been demonstrated that the JDP-base estimator works better than a Kalman filter when jumps, such as potholes, cracks and bumps. With the estimated states, a threshold-based road defect detection algorithm was developed, and a false negative rate of 1.78% was reported.

Data-Driven Approaches

Data-driven methods involve training machine learning models using either extracted features or raw sensor signals to produce results close to labeled ground truth. A variety of signal-processing techniques can be applied to extract features from rich sensor data. In the time domain, features can be derived from statistics such as standard deviation, mean, entropy, variance, root mean square, median, integral square, and range [109]. Another method used for feature extraction in the time domain is dynamic time warping (DTW), which is primarily employed in research on speech recognition. DTW calculates the degree of similarity between two datasets by comparing incoming signal data to established templates. The similarity can then be used as a feature or compared against a threshold for detection purposes. In [33], DTW was compared to other machine learning methods such as support vector machines (SVMs), hidden-Markov models (HMMs), and general neural networks. In the frequency domain, the power spectral density (PSD) of vibration signals reveals useful insights that could be used to distinguish different road conditions.

Features can also be extracted in the wavelet domain (see, for example, [105, 110, 111]). In her PhD thesis [110], Griffiths carefully analyzed Mexican Hat, Mortlet, Haar, as well as Daubechies 6 and 10. She came to the conclusion that road vehicle vibrations may be adequately analyzed using the Mortlet wavelet as well as the Daubechies 6 and 10 wavelets. Seraj et al. utilised wavelet decomposition analysis for signal processing of inertial sensor signals from smartphones. In [111], Li et al. applied continuous wavelet transform (CWT) to extract features from sensor data and were able to estimate the length of potholes by converting wavelet scales to physical scales. Additionally, Fourier transform and wavelet transform can be used to generate virtual images from 1D sensor data, making them compatible with convolutional neural networks (CNNs) [112].

As for classification, numerous machine learning techniques have been applied to detect road defects from vibration data. In [32], classifiers such as SVM, linear regression (LR), and random forest (RF) were trained and used for prediction. They reported that RF had the best performance as compared to other classifiers, and a precision of 88.5% was achieved. In [105], SVM was utilized in a real-time multi-class detector that achieved consistent detection accuracy of approximately 90% for severe road defects, regardless of vehicle type or road location. Du et al. [103] used an improved Gaussian model for road-surface recognition and the k-nearest neighbor (kNN) method for classification. Their system operated at 400Hz for data sampling, and a detection accuracy rate of 96.03% was reported. In [104], the authors utilized a $C4.5$ decision tree algorithm to classify sensor data from smartphones for detecting road conditions such as potholes and bumps. They developed an Android application called *RoadSense* and evaluated its performance with experimental data, which showed a consistent detection rate of around 98.6%. In [113], the authors proposed a two-step method. First, a random forest filter was applied to quickly distinguish normal windows from defective windows. Then, data windows with different lengths were analyzed using DTW together with the kNN to determine the type of defects. Self Organizing Map (SOM) was applied in [36] to handle data collected from buses.

Road surface defect detection has also been approached using deep learning techniques, such as deep feed-forward networks (DFN), recurrent neural networks (RNNs), CNNs, and long-short-term memories (LSTMs). One advantage of deep learning techniques is that they can directly process the raw data without requiring any feature extraction process. However, when using CNNs for 1D signals, such as accelerometer sensor data, it is necessary to transform the data into a 2D virtual image, which can be achieved using various methods. In [114], various deep learning models such as CNN, LSTM, and reservoir computing models were examined for road surface defect detection. In [112], Baldini et al. proposed using short-time Fourier Transform (STFT) and continuous wavelet transform (CWT) on inertial sensors to generate virtual images for CNNs to classify road defects, achieving an accuracy of 97.2%. Luo et al. compared DFNs, CNNs, and RNNs in [37], using a range of sensor data, including suspension states, and found that RNNs outperformed DFNs and CNNs while using fewer model parameters.

Seraj et al. proposed a method to detect road defects by analyzing driver behaviors in [115], which is different from previous studies that focused on the detection of different types of pavements and/or types of defects. They were able to reliably detect swerves for analyzing driver behavior by studying features of the road curves and using a simple machine learning technique. Signal analysis was performed using Stationary Wavelet Transformation (SWT) decomposition. In 2017, a similar approach of detecting road defects from the behavior of vehicles was presented in [116], where the sum-of-squares (SOS) polynomial was used as a metric to evaluate the normality of vehicle behavior.

11.4.2.3 Crowd-Sourcing

When it comes to road defect detection without vision sensors, aggregating detection results from different drives or crowd-sourced data can be more challenging compared to vision-based solutions, as low-cost GPS sensors are usually the only source of location information. In [28], a distance grid-based spatial clustering algorithm was introduced to remove false positives that were unlikely to occur at the exact same location. A similar spatial clustering method was implemented in [115] to reduce smartphone GPS error, enabling precise aggregation of swerving behavior at the same location. In 2016, [117] proposed a clustering algorithm that incorporates Mahalanobis distance to quantify the similarity between newly reported and existing clusters. This method can effectively localize isolated defects and compress information for densely distributed ones.

11.5 Planning and Control

Planning and control are crucial aspects of autonomous vehicles that help ensure safe and efficient navigation. Planning refers to the process of determining a safe and efficient path for the vehicle to follow, taking into account its surroundings, environmental conditions, and any constraints such as traffic laws and regulations. The vehicle's planning system considers factors such as traffic patterns, road conditions, and the position and movement of other objects in the environment to create a map of the driving space. Based on this information, the vehicle's onboard computer generates a path that minimizes risk and optimizes efficiency and updates it continuously as the vehicle moves and its surroundings change. Control, on the other hand, refers to the processes that ensure the vehicle follows the planned path. The control system of an autonomous vehicle receives information from various sensors and actuators, such as cameras, lidar, radar, and GPS, to understand its position, speed, and orientation. Based on this information, the control system makes decisions about how to control the vehicle's movement, including steering, accelerating, and braking. The control system is also responsible for monitoring the vehicle's performance and making any necessary adjustments to keep it on track and ensure it is operating safely.

Different from other planning layers, a motion planner is a component of the planning system in an autonomous vehicle that is responsible for generating smooth and safe vehicle trajectories in the environment. The motion planner takes into account the vehicle's surroundings, including other vehicles, road conditions, and any constraints such as traffic laws, to generate a feasible and safe path for the vehicle to follow (see [118, 119], and references therein). However, the majority of recently reported motion planning techniques are 2D-based and only the vehicle's in-plane (longitudinal and lateral) motions will take into account. Previous studies on autonomous vehicles have investigated the potential of utilizing preview information of upcoming terrain, such as road grade, for optimizing their performance (e.g., [120–122]). However, these studies did not address how vertical trajectories can be planned to improve the comfort and safety of autonomous vehicles. In contrast, active suspension technology, which is a suspension system that can adjust in real-time based on road conditions and driving dynamics, has been extensively studied (e.g., [123–125]). When applied to autonomous vehicles, active suspension technology can enhance the driving experience by improving ride comfort, stability and control, safety, and sensor performance.

In [126], a new framework referred to as XYZ motion planning was introduced by Jiang et al. . It is shown that with the knowledge of road surface defects, such as potholes or large dips, as well as the ability to actively control the vertical motion of an autonomous vehicle, a 3-D path can be planned for the vehicle, and can be executed by speed, steering, and active suspension control systems. It was also shown via simulation that this new planning and control framework provides increased ride comfort and safety.

It has been reported that Tesla's 2022.20 update includes a new feature that allows a vehicle to predict rough roads and raise the suspension accordingly for Model S and Model X vehicles, using data collected from other Teslas. This feature has been seen as a possible step towards the pothole-avoiding feature that Tesla CEO Elon Musk promised in 2020 [127]. However, unlike active suspension systems, this feature is based on predictive modeling and does not involve real-time adjustments to the suspension system. In contrast, Ford developed a semi-active suspension system for their 2017 Fusion V6 Sport that has a feedback control-based system installed to lessen the discomfort experienced by drivers when traveling on a road with potholes [128].

11.6 Existing Challenges and Future Insights

In the era before the advent of deep learning in 2012, the primary research focus in the field of road perception were classical 2-D image processing-based methods. However, these explicit programming-based methods are computationally demanding and sensitive to environmental factors, particularly changes weather in and illumination conditions [13]. Additionally, the irregular shapes of road defects make it infeasible to rely on the geometric assumptions made in these methods. To overcome these limitations, 3D point cloud modeling and segmentation-based techniques were introduced in 2013, which greatly improved the accuracy of pothole detection [64]. However, these approaches have a limited field of view, assuming a single-frame 3D road point cloud forms a planar or quadratic surface. Despite efforts to enhance the robustness of road point cloud modeling, such as using the RANSAC algorithm [22], these methods require many parameters to guarantee satisfactory performance, making them challenging to apply to new scenarios.

Autonomous driving perception systems require detailed geometric information about road defects and obstacles, such as width, depth, and volume. As a result, hybrid methods that combine 3-D road reconstruction and semantic segmentation have become a promising area of research. While deep learning networks have shown excellent performance, they require large amounts of annotated data, making unsupervised or self-supervised stereo matching algorithms an attractive alternative for road surface 3-D reconstruction. Furthermore, the development of more comprehensive datasets for road perception and innovative data fusion methods, such as those proposed by [97, 98, 129], are essential for further progress in this field.

In recent years, advancements in computing power and the decreasing cost of sensing and computing technologies have led to significant progress in the research of autonomous driving systems. As a result, various companies are actively researching related technologies. Advanced driver assistance systems, such as Autopilot [130] and Super Cruise [131], integrate traffic-aware cruise control and lane-keeping assist systems to improve safety and reduce driver workload. Autopilot, as implemented in Tesla cars, adjusts the vehicle's speed to match surrounding traffic and centers the vehicle in a clearly marked lane. Similarly, Super Cruise includes adaptive cruise control that maintains a safe distance from the vehicle ahead and a lane-keeping system that keeps the vehicle centered in the lane, even around curves. Additionally, Super Cruise employs automatic emergency braking to help prevent collisions. Overall, these features enhance safety by reducing driver workload and assisting in potentially dangerous situations.

Although significant progress has been made toward fully autonomous vehicles, there is still a long way to go before driverless cars become the norm. According to a poll by Partners for Automated Vehicle Education (PAVE) [132], Americans remain skeptical of current AV technology, but are slightly more optimistic about the availability of safe autonomous driving technology in the future. Nearly three in four Americans believe that AV technology is not ready for widespread use. Additionally, 48% of Americans say they would never ride in a taxi or ride-share vehicle that was being driven autonomously. However, 58% believe that safe AVs will be available in ten years, while 20% believe that they will never be safe. The industry needs to work on gaining the trust and confidence of drivers before autonomous driving cars can become mainstream.

11.7 Conclusion

Researchers are highly engaged in road environment perception systems as autonomous driving technology advances. However, there is currently a lack of comprehensive surveys on the latest computer vision and vibration-based defect detection techniques, particularly those involving deep learning methods. We first discussed sensing technologies for acquiring data from vision-based and vibration-based sensors. Following that, we summarized several public datasets about potholes and cracks. Then, a detailed analysis of SoTA road defect detection algorithms was presented. Computer vision-based methods include (1) 2-D image processing-based approaches, (2) 3-D point modeling and segmentation, and (3) deep learning-based approaches. Road defect detection methods based on vibration were also extensively discussed. This chapter also discussed the other core competencies of autonomous vehicle software systems: Planning and Control. Finally, we envisage that future autonomous driving systems will combine vision and motion sensors.

Acknowledgements This work was supported by the National Key R&D Program of China under Grant 2020AAA0108100, the National Natural Science Foundation of China under Grant 62233013, and the Science and Technology Commission of Shanghai Municipal under Grant 22511104500.

References

1. The pothole facts. https://www.pothole.info/the-facts/
2. Accidents and injuries caused by bad road conditions. https://www.gregcolemanlaw.com/bad-road-damages-and-effects.html
3. Ma N et al (2022) Computer vision for road imaging and pothole detection: a state-of-the-art review of systems and algorithms. Transp Safety Environ 4(4):tdac026
4. Winn J, Shotton J (2006) The layout consistent random field for recognizing and segmenting partially occluded objects. In: 2006 IEEE computer society conference on computer vision and pattern recognition (CVPR), vol 1. IEEE, pp 37–44
5. Pandey G et al (2011) Ford campus vision and LIDAR data set. Int J Robot Res 30(13):1543–1552
6. Korosec K (2027) Toyota is betting on this startup to drive its self driving car plans forward. https://fortune.com/2017/09/27/toyota-self-driving-car-luminar/, November 2027
7. Proactive chassis controls powered by road surface fingerprinting software. https://www.clearmotion.com/roadmotion
8. Roberts J, Corke P (2020) Obstacle detection for a mining vehicle using a 2D laser. Proceedings of the Australian conference on robotics and automation 2000:185–190
9. Lakhotia A et al (2006) CajunBot: architecture and algorithms. J Field Robot 23(8):555–578
10. Mahler DS et al (1991) Pavement distress analysis using image processing techniques. Comput-Aided Civil Infrastruct Eng 6(1):1–14
11. Koutsopoulos HN, Downey A (1993) Primitive-based classification of pavement cracking images. J Transp Eng 119(3):402–418
12. Fan R et al (2018) Road surface 3D reconstruction based on dense subpixel disparity map estimation. IEEE Trans Image Process 27(6):3025–3035
13. Jahanshahi MR et al (2013) Unsupervised approach for autonomous pavement-defect detection and quantification using an inexpensive depth sensor. J Comput Civ Eng 27(6):743–754
14. Fan R et al (2019) Real-time dense stereo embedded in a UAV for road inspection. In: 2019 IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW). IEEE Computer Society, pp 535–543
15. Fan R et al (2022) Rethinking road surface 3-D reconstruction and pothole detection: from perspective transformation to disparity map segmentation. IEEE Trans Cybern 52(7):5799–5808
16. Laurent J et al (1997) Road surface inspection using laser scanners adapted for the high precision 3D measurements of large flat surfaces. In: Proceedings international conference on recent advances in 3-D digital imaging and modeling (Cat. No. 97TB100134). IEEE, pp 303–310
17. Mathavan S et al (2015) A review of three-dimensional imaging technologies for pavement distress detection and measurements. IEEE Trans Intell Transp Syst 16(5):2353–2362
18. Tsai Y, Chatterjee A (2018) Pothole detection and classification using 3d technology and watershed method. J Comput Civ Eng 32(2):04017078
19. Guo S et al (2022) Digital transformation for intelligent road condition assessment. In: Intelligent systems in digital transformation: theory and applications. Springer, pp 511–533

20. Joubert D et al (2011) Pothole tagging system. In: The robotics and mechatronics conference of South Africa, CSIR International Conference Centre, Pretoria, pp 23–25
21. Moazzam I et al (2013) Metrology and visualization of potholes using the microsoft kinect sensor. In: 16th international IEEE conference on intelligent transportation systems (ITSC 2013). IEEE, pp 1284–1291
22. Fan R et al (2019) Pothole detection based on disparity transformation and road surface modeling. IEEE Trans Image Process 29:897–908
23. Warade S et al (2012) Real-time detection and tracking with Kinect. In: International conference on computer and information technology, pp 86–89
24. Liu C-W et al (2023) Stereo matching: fundamentals, state-of-the-art, and existing challenges. Springer
25. Andrew AM (2021) Multiple view geometry in computer vision. Kybernetes
26. Soheilian B et al (2013) Detection and 3D reconstruction of traffic signs from multiple view color images. ISPRS J Photogramm Remote Sens 77:1–20
27. Li Z et al (2017) Optimal state estimation for systems driven by jump-diffusion process with application to road anomaly detection. IEEE Trans Control Syst Technol 25(5):1634–1643
28. Eriksson J et al (2008) The pothole patrol: using a mobile sensor network for road surface monitoring. In: Proceedings of the 6th international conference on mobile systems, applications, and services, Breckenridge, pp 29–39
29. Mohan P et al (2008) Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In: Proceedings of the 6th ACM conference on embedded network sensor systems, Raleigh, NC, November 2008, pp 323–336
30. Perttunen M et al (2011) Distributed road surface condition monitoring using mobile phones. In: Ubiquitous intelligence and computing. Springer, pp 64–78
31. Mednis A et al (2011) Real time pothole detection using android smartphones with accelerometers. In: 2011 international conference on distributed computing in sensor systems and workshops (DCOSS), Barcelona, Spain, 2011, pp 1–6
32. Wu C et al (2020) An automated machine-learning approach for road pothole detection using smartphone sensor data. Sensors 20(19)
33. Singh G et al (2017) Smart patrolling: an efficient road surface monitoring using smartphone sensors and crowdsourcing. Pervasive Mob Comput 40:71–88
34. Bhoraskar R et al (2012) Wolverine: traffic and road condition estimation using smartphone sensors. In: 2012 fourth international conference on communication systems and networks (COMSNETS 2012), Bangalore, India, January 2012, pp 1–6
35. Mohamed A (2015) Roadmonitor: An intelligent road surface condition monitoring system. In: Proceedings of the 7th IEEE international conference intelligent systems IS'2014, September 24–26, 2014, Warsaw, Poland, Volume 2: tools, architectures, systems, applications. Springer, pp 377–387
36. Rajput P et al (2022) Road condition monitoring using unsupervised learning based bus trajectory processing. Multimodal Transp 1(4):100041
37. Luo D et al (2020) Road anomaly detection through deep learning approaches. IEEE Access 8:117 390–117 404
38. Geiger A et al (2013) Vision meets robotics: the KITTI dataset. Int J Robot Res 32(11):1231–1237
39. Cordts M et al (2016) The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 3213–3223
40. Raj P et al (2020) Point cloud LIDAR (Toronto 3D). https://www.kaggle.com/datasets/priteshraj10/point-cloud-lidar-toronto-3d
41. Dubrovin A et al (2020) The radarscenes data set. https://www.kaggle.com/datasets/aleksandrdubrovin/the-radarscenes-data-set
42. Cordes K et al (2022) RoadSaW: a large-scale dataset for camera-based road surface and wetness estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 4440–4449

43. Rath SR (2020) Road pothole images for pothole detection. https://www.kaggle.com/datasets/sovitrath/road-pothole-images-for-pothole-detection, September 2020
44. Bombay YI (2021) Semantic segmentation datasets of indian roads. https://www.kaggle.com/datasets/eyantraiit/semantic-segmentation-datasets-of-indian-roads, November 2021
45. Maeda H et al (2018) Road damage detection using deep neural networks with images captured through a smartphone. Comput Res Repos (CoRR). arXiv:1801.09454
46. Fan R (2020) We learn better road pothole detection: from attention aggregation to adversarial domain adaptation. In: Computer vision-ECCV 2020 workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16. Springer, pp 285–300
47. Fan R et al (2018) A novel disparity transformation algorithm for road segmentation. Inf Process Lett 140:18–24
48. Zhang W et al (2014) Automatic crack detection and classification method for subway tunnel safety monitoring. Sensors 14(10):19 307–19 328
49. Medina R et al (2017) Crack detection in concrete tunnels using a gabor filter invariant to rotation. Sensors 17(7):1670
50. Cha Y et al (2017) Deep learning-based crack damage detection using convolutional neural networks. Comput-Aided Civil Infrastruct Eng 32(5):361–378
51. Maguire M et al (2018) SDNET2018: a concrete crack image dataset for machine learning applications
52. Choi W, Cha Y-J (2019) SDDNet: Real-time crack segmentation. IEEE Trans Industr Electron 67(9):8016–8025
53. Ren Y et al (2020) Image-based concrete crack detection in tunnels using deep fully convolutional networks. Constr Build Mater 234:117367
54. Chen F, Jahanshahi MR (2020) ARF-Crack: rotation invariant deep fully convolutional network for pixel-level crack detection. Mach Vis Appl 31(6):47
55. Koch C et al (2015) A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure. Adv Eng Inform 29(2):196–210
56. Fan R (2019) Road crack detection using deep convolutional neural network and adaptive thresholding. In: IEEE intelligent vehicles symposium (IV). IEEE, pp 474–479
57. Fan R et al (2022) Learning collision-free space detection from stereo images: homography matrix brings better data augmentation. IEEE/ASME Trans Mechatron 27(1):225–233
58. Liu Y et al (2019) DeepCrack: a deep hierarchical feature learning architecture for crack segmentation. Neurocomputing 338:139–153
59. Ryu S et al (2015) Image-based pothole detection system for ITS service and road management system. Math Prob Eng 2015
60. Koch C, Brilakis I (2011) Pothole detection in asphalt pavement images. Adv Eng Inform 25(3):507–515
61. Fan R, Liu M (2020) Road damage detection based on unsupervised disparity map segmentation. IEEE Trans Intell Transp Syst 21(11):4906–4911
62. Beylkin G et al (2009) Fast wavelet transforms and numerical algorithms. In: Fundamental papers in wavelet theory. Princeton University Press, pp 741–783
63. Najman L, Schmitt M (1994) Watershed of a continuous function. Signal Process 38(1):99–112
64. Zhang Z (2013) Advanced stereo vision disparity calculation and obstacle analysis for intelligent vehicles. PhD dissertation, University of Bristol
65. Wu R et al (2021) Scale-adaptive road pothole detection and tracking from 3D point clouds. In: 2021 IEEE international conference on imaging systems and techniques (IST). IEEE, pp 1–5
66. Li Y et al (2018) Road pothole detection system based on stereo vision. In: NAECON 2018-IEEE national aerospace and electronics conference. IEEE, pp 292–297
67. Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM 24(6):381–395
68. Ozgunalp U (2016) Vision based lane detection for intelligent vehicles. PhD dissertation, University of Bristol

69. Pereira V et al (2018) A deep learning-based approach for road pothole detection in timor leste. In: 2018 IEEE international conference on service operations and logistics, and informatics (SOLI). IEEE, pp 279–284
70. An KE et al (2018) Detecting a pothole using deep convolutional neural network models for an adaptive shock observing in a vehicle driving. In: 2018 IEEE international conference on consumer electronics (ICCE). IEEE, pp 1–2
71. Ye W et al (2021) Convolutional neural network for pothole detection in asphalt pavement. Road Mater Pavement Design 22(1):42–58
72. Bhatia Y et al (2022) Convolutional neural networks based potholes detection using thermal imaging. J King Saud University-Comput Inf Sci 34(3):578–588
73. He K et al (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 770–778
74. Hoang N (2018) An artificial intelligence method for asphalt pavement pothole detection using least squares support vector machine and neural network with steerable filter-based feature extraction. Adv Civil Eng 2018
75. LeCun Y et al (2015) Deep learning. Nature 521(7553):436–444
76. Zhang L (2016) Road crack detection using deep convolutional neural network. In: IEEE international conference on image processing (ICIP). IEEE, pp 3708–3712
77. Fan J et al (2021) Deep convolutional neural networks for road crack detection: qualitative and quantitative comparisons. In: 2021 IEEE international conference on imaging systems and techniques (IST). IEEE, pp 1–6
78. Hoffman J et al (2018) Cycada: cycle-consistent adversarial domain adaptation. In: International Conference on Machine Learning. PMLR, pp 1989–1998
79. Chattopadhay A et al (2018) Grad-CAM++: generalized gradient-based visual explanations for deep convolutional networks. In: 2018 IEEE winter conference on applications of computer vision (WACV). IEEE, pp 839–847
80. Fan R et al (2023) Computer-aided road inspection: systems and algorithms. In: Recent advances in computer vision applications using parallel processing. Springer, pp 13–39
81. He K et al (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 770–778
82. Hu J et al (2018) Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 7132–7141
83. Girshick R (2015) Fast R-CNN. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp 1440–1448
84. Girshick R et al (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) 2014:580–587
85. Ren S et al (2015) Faster R-CNN: towards real-time object detection with region proposal networks. Adv Neural Inf Process Syst (NeurIPS) 28:91–99
86. Redmon J et al (2016) You only look once: Unified, real-time object detection. Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) 2016:779–788
87. Redmon J, Farhadi A (2017) YOLO9000: better, faster, stronger. Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) 2017:7263–7271
88. Suong LK et al (2018) Detection of potholes using a deep convolutional neural network. J Univers Comput Sci 24(9):1244–1257
89. Wang W et al (2018) Road damage detection and classification with faster R-CNN. In: 2018 IEEE international conference on big data (big data). IEEE, pp 5220–5223
90. Ukhwah EN (2019) Asphalt pavement pothole detection using deep learning method based on YOLO neural network. In: International seminar on intelligent technology and its applications (ISITIA). IEEE, pp 35–40
91. Redmon J, Farhadi A (2018) YOLOv3: an incremental improvement. Comput Res Repos (CoRR). arXiv:1804.02767
92. Camilleri N, Gatt T (2020) Detecting road potholes using computer vision techniques. In: 2020 IEEE 16th international conference on intelligent computer communication and processing (ICCP). IEEE, pp 343–350

93. Dhiman A, Klette R (2019) Pothole detection using computer vision and learning. IEEE Trans Intell Transp Syst 21(8):3536–3550
94. He K et al (2017) Mask R-CNN. Proceedings of the IEEE international conference on computer vision (ICCV) 2017:2961–2969
95. Fan R et al (2023) Computer stereo vision for autonomous driving: Theory and algorithms. In: Recent Advances in Computer Vision Applications Using Parallel Processing. Springer International Publishing, 2023, pp 41–70
96. Yang J et al (2023) Semantic segmentation for autonomous driving. Springer
97. Wang H et al (2021) Dynamic fusion module evolves drivable area and road anomaly detection: a benchmark and algorithms. IEEE Trans Cybern 52(10):10 750–10 760
98. Fan R et al (2020) SNE-RoadSeg: incorporating surface normal information into semantic segmentation for accurate freespace detection. In: European conference on computer vision (ECCV). Springer, pp 340–356
99. Fan R et al (2021) Three-filters-to-normal: An accurate and ultrafast surface normal estimator. IEEE Robot Autom Lett 6(3):5405–5412
100. Jog G et al (2012) Pothole properties measurement through visual 2D recognition and 3d reconstruction. Comput Civil Eng 2012:553–560
101. Kang B-H, Choi S-I (2017) Pothole detection system using 2D LiDAR and camera. In: 2017 ninth international conference on ubiquitous and future networks (ICUFN). IEEE, pp 744–746
102. Otsu N (1979) A threshold selection method from gray-level histograms. IEEE Trans Syst Man Cybern 9(1):62–66
103. Du R et al (2020) Abnormal road surface recognition based on smartphone acceleration sensor. Sensors 20(2):451
104. Allouch A et al (2017) Roadsense: Smartphone application to estimate road conditions using accelerometer and gyroscope. IEEE Sens J 17(13):4231–4238
105. Seraj F et al (2016) RoADS: a road pavement monitoring system for anomaly detection using smart phones. Big data analytics in the social and ubiquitous context. Springer International Publishing, Cham, pp 128–146
106. Sattar S et al (2021) Developing a near real-time road surface anomaly detection approach for road surface monitoring. Measurement 185:109990
107. Carlos MR et al (2016) Evaluating reorientation strategies for accelerometer data from smartphones for ITS applications. Ubiquitous computing and ambient intelligence. Springer International Publishing, Cham, pp 407–418
108. Li Z et al (2015) Road anomaly estimation: model based pothole detection. American control conference (ACC) 2015:1315–1320
109. Martinez-Ríos EA et al (2022) A review of road surface anomaly detection and classification systems based on vibration-based techniques. Appl Sci 12(19)
110. Griffiths KR (2012) An improved method for simulation of vehicle vibration using a journey database and wavelet analysis for the pre-distribution testing of packaging. PhD dissertation, University of Bath, Bath
111. Li X et al (2019) Embracing crowdsensing: an enhanced mobile sensing solution for road anomaly detection. ISPRS Int J Geo Inf 8(9):412
112. Baldini G et al (2020) On the application of time frequency convolutional neural networks to road anomalies' identification with accelerometers and gyroscopes. Sensors 20(22):6425
113. Zheng Z et al (2020) A fused method of machine learning and dynamic time warping for road anomalies detection. IEEE Trans Intell Transp Syst 23(2):827–839
114. Varona B et al (2020) A deep learning approach to automatic road surface monitoring and pothole detection. Pers Ubiquit Comput 24:519–534
115. Seraj F et al (2015) A smartphone based method to enhance road pavement anomaly detection by analyzing the driver behavior. In: Adjunct proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing and proceedings of the 2015 ACM international symposium on wearable computers, Osaka, Japan, September 2015, pp 1169–1177

116. Shu D et al (2017) A sum-of-squares polynomial approach for road anomaly detection using vehicle sensor measurements. In: Dynamic systems and control conference, vol 58288, Tysons, VA, USA, November 2017, p V002T17A004
117. Li Z et al (2016) A new clustering algorithm for processing gps-based road anomaly reports with a mahalanobis distance. IEEE Trans Intell Transp Syst 18(7):1980–1988
118. Paden B et al (2016) A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Trans. Intell. Veh. 1(1):33–55
119. Claussmann L et al (2020) A review of motion planning for highway autonomous driving. IEEE Trans Intell Transp Syst 21(5):1826–1848
120. Fröberg A (2008) Efficient simulation and optimal control for vehicle propulsion. PhD dissertation, Department of Electrical Engineering, Linköping University
121. Ward JW et al (2022) A method of optimal control for class 8 vehicle platoons over hilly terrain. J Dyn Syst Meas Contr 144(1):1–18
122. Sciarretta A et al (2015) Optimal ecodriving control: energy-efficient driving of road vehicles as an optimal control problem. IEEE Comput Sci Eng 35(5):71–90
123. Tseng HE et al (2015) State of the art survey: active and semi-active suspension control. Veh Syst Dyn 53(7):1034–1062
124. Anderson Z et al (2020) Self-driving vehicle with integrated active suspension. USA Patent US10 828 953B2, Nov 10, 2020
125. Anderson ZM et al (2020) Active vehicle suspension system. USA Patent US9 702 349B2, July 11, 2017
126. Jiang Y et al (2023) On XYZ-motion planning for autonomous vehicles with active suspension systems. In: American control conference. San Diego, SD
127. Teslas can now use adaptive suspension to automatically smooth the ride over rough roads. https://www.carscoops.com/2022/07/teslas-can-now-use-adaptive-suspension-to-automatically-smooth-the-ride-over-rough-roads. Accessed: 2023-02-13
128. Technology in 2017 ford fusion helps protect against pothole damage. https://www.cbsnews.com/detroit/news/technology-in-2017-ford-fusion-helps-protect-against-pothole-damage/, Accessed: 2023-02-13
129. Wang H et al (2020) Applying surface normal information in drivable area and road anomaly detection for ground mobile robots. In: 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 2706–2711
130. Souman J et al (2021) Human factors guidelines report 2: driver support systems overview. TNO Human Factors Research Institute
131. What Do I Need To Use Super Cruise? https://www.cadillac.com/ownership/vehicle-technology/super-cruise
132. PAVE Poll: Americans wary of AVs but say education and experience with technology can build trust. https://pavecampaign.org/pave-poll-americans-wary-of-avs-but-say-education-and-experience-with-technology-can-build-trust/