Computer Stereo Vision for Autonomous Driving: Theory and Algorithms

Rui Fan, Li Wang, Mohammud Junaid Bocus, and Ioannis Pitas

Abstract As an important component of autonomous systems, autonomous car perception has had a big leap with recent advances in parallel computing architectures. With the use of tiny but full-feature embedded supercomputers, computer stereo vision has been prevalently applied in autonomous cars for depth perception. The two key aspects of computer stereo vision are speed and accuracy. They are both desirable but conflicting properties, as the algorithms with better disparity accuracy usually have higher computational complexity. Therefore, the main aim of developing a computer stereo vision algorithm for resource-limited hardware is to improve the trade-off between speed and accuracy. In this chapter, we introduce both the hardware and software aspects of computer stereo vision for autonomous car systems. Then, we discuss four autonomous car perception tasks, including (1) visual feature detection, description and matching, (2) 3D information acquisition, (3) object detection/recognition and (4) semantic image segmentation. The principles of computer stereo vision and parallel computing on multi-threading CPU and GPU architectures are then detailed.

R. Fan (✉)
Tongji University, Shanghai, China
e-mail: rfan@tongji.edu.cn

L. Wang
Continental Automotive Systems (Shanghai) Co., Ltd., Shanghai, China
e-mail: li.15.wang@continental-corporation.com

M. Junaid Bocus
University of Bristol, Bristol, England
e-mail: junaid.bocus@bristol.ac.uk

I. Pitas
Aristotle University of Thessaloniki, Thessaloniki, Greece
e-mail: pitas@csd.auth.gr

K. M. Hosny and A. Salah (eds.), *Recent Advances in Computer Vision Applications Using Parallel Processing*, Studies in Computational Intelligence 1073,
https://doi.org/10.1007/978-3-031-18735-3_3

1 Introduction

Autonomous car systems enable self-driving cars to navigate in complicated environments, without any intervention of human drivers. An example of autonomous car system architecture is illustrated in Fig. 1. Its hardware (HW) mainly includes: (1) car sensors, such as cameras, LIDARs, and Radars; and (2) car chassis, such as throttle, brake, and wheel. On the other hand, the software (SW) is comprised of four main functional modules [1]: (1) perception, (2) localization and mapping, (3) prediction and planning and (4) control. Computer stereo vision is an important component of the perception module. It enables self-driving cars to perceive environment in 3D.

This chapter first introduces the HW/SW aspects of an autonomous car system. Then, four autonomous car perception tasks are discussed, including: (1) visual feature detection, description and matching, (2) 3D information acquisition, (3) object detection/recognition and (4) semantic image segmentation. Finally, the principles of computer stereo vision and parallel computing on multi-threading CPU and GPU are detailed.

2 Autonomous Car System

2.1 Hardware

2.1.1 Car Sensors

The most commonly used car sensors include: (a) passive sensors, such as cameras; (b) active sensors, such as LIDARs, Radars and ultrasonic transceivers; and (c) other types of sensors, such as global positioning systems (GPS), inertial measurement unit (IMU), among others. When choosing them, we need to consider many different factors, such as sampling rate, field of view (FoV), accuracy, range, cost and overall system complexity.[1]

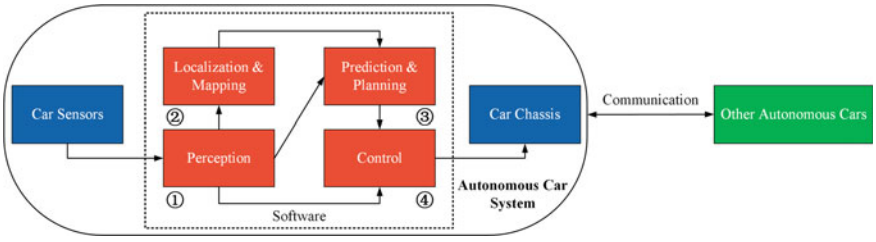

Fig. 1 Autonomous car system architecture

[1] https://autonomous-driving.org/2019/01/25/positioning-sensors-for-autonomous-vehicles.

Cameras capture 2D images, by collecting light reflected on 3D objects. Images captured from different views can be utilized to reconstruct the 3D driving scene geometry. Most autonomous car perception tasks, such as visual semantic driving scene segmentation and object detection/recognition, are developed for images. In Sect. 3, we provide readers with a comprehensive overview of these tasks. The perspective (or pinhole) camera model and the mathematical principles of multi-view geometry are discussed in Sect. 4. Acquired image quality is always subject to environmental conditions, such as weather and illumination. Therefore, the visual information fusion from other sensors is typically required for robust autonomous car perception.

LIDAR illuminates a target with pulsed laser light and measures the source-target distance, by analyzing the reflected laser pulses [2]. Due to its ability to generate highly accurate 3D driving scene geometry models, LIDARs are generally mounted on autonomous cars for depth perception. Current industrial autonomous car localization and mapping systems are generally based on LIDARs. Furthermore, Radars can measure both the range and radial velocity of an object, by transmitting an electromagnetic wave and analyzing its reflections [3]. Radars have already been established in the automotive industry, and they have been prevalently employed to enable intelligent vehicle advanced driver assistance system (ADAS) features, such as adaptive cruise control and autonomous emergency braking.[1] Similar to Radar, ultrasonic transceivers calculate the source-object distance, by measuring the time between transmitting an ultrasonic signal and receiving its echo [4]. Ultrasonic transceivers are commonly used for autonomous car localization and navigation.

In addition to the aforementioned passive and active sensors, GPS and IMU systems are commonly used to enhance autonomous car localization and mapping performance [1]. GPS can provide both time and geolocation information for autonomous cars. However, its signals can become very weak, when GPS reception is blocked by obstacles in GPS-denied regions, such as urban regions [5]. Hence, GPS and IMU information fusion is widely adopted to provide continuous autonomous car position and velocity information [1].

2.1.2 Car Chassis

Car chassis technologies, especially Drive-by-Wire (DbW), are required for building autonomous vehicles. DbW technology refers to the electronic systems that can replace traditional mechanical controls [6]. DbW systems can perform vehicle functions, which are traditionally achieved by mechanical linkages, through electrical or electro-mechanical systems. There are three main vehicle control systems that are commonly replaced with electronic controls: (1) throttle, (2) braking and (3) steering.

A Throttle-by-Wire (TbW) system helps accomplish vehicle propulsion via an electronic throttle, without any cables from the accelerator pedal to the engine throttle valve. In electric vehicles, TbW system controls the electric motors, by sensing accelerator pedal for a pressure (input) and sending signal to the power inverter modules. Compared to traditional hydraulic brakes, which provide braking effort,

by building hydraulic pressure in the brake lines, a Brake-by-Wire (BbW) system completely eliminates the need for hydraulics, by using electronic motors to activate calipers. Furthermore, in vehicles that are equipped with Steer-by-Wire (SbW) technology, there is no physical connection between the steering wheel and the tires. The control of wheels' direction is established through electric motor(s), which are actuated by electronic control units monitoring steering wheel inputs.

In comparison to traditional throttle systems, electronic throttle systems are much lighter, hence greatly reducing modern car weight. In addition, they are easier to service and tune, as a technician can simply connect a laptop to perform tuning. Moreover, an electronic control system allows more accurate control of throttle opening, compared to a cable control that stretches over time. Furthermore, since the steering wheel can be bypassed as an input device, safety can be improved by providing computer controlled intervention of vehicle controls with systems, such as Adaptive Cruise Control and Electronic Stability Control.

2.2 Software

Autonomous car perception module analyzes the raw data collected by car sensors (see Sect. 2.1.1) and outputs its understanding to the environment. This process is similar to human visual cognition. We discuss different autonomous car perception tasks in Sect. 3.

Perception module outputs are then used by other modules. The localization and mapping module not only estimates autonomous car location, but also constructs and updates the 3D environment map [7]. This topic has become very popular, since the concept of *Simultaneous Localization and Mapping (SLAM)* was introduced in 1986 [8].

Prediction and planning module first analyzes the motion patterns of other traffic agents and predicts their future trajectories. Such prediction outputs are then used to determine possible safe autonomous car navigation routes [9] using different path planning techniques, such as Dijkstra [10], A-star (or simply A*) [11], etc.

Finally, autonomous car control module sends appropriate commands to car controllers (see Sect. 2.1.2), based on its predicted trajectory and the estimated car state. This enables the autonomous car to follow the planned trajectory, as closely as possible. Traditional controllers, such as proportional-integral-derivative (PID) [12], linear-quadratic regulator (LQR) [13] and model predictive control (MPC) [14] are still the most commonly used ones in autonomous car control module.

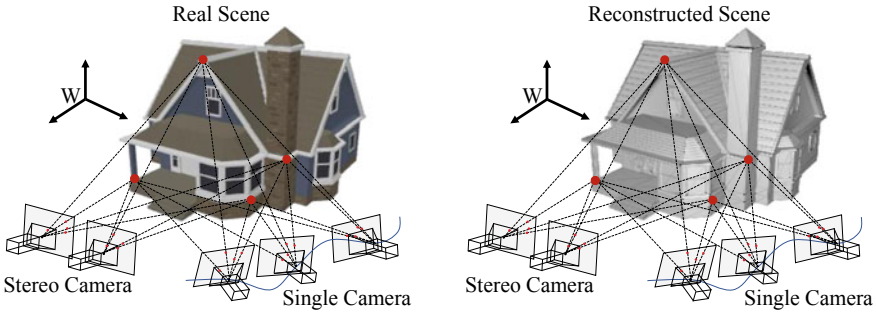

Fig. 2 3D scene reconstruction, where W presents the world coordinate system (WCS)

3 Autonomous Car Perception

The autonomous car perception module has four main functionalities:

1. visual feature detection, description and matching;
2. 3D information acquisition;
3. objection detection/recognition;
4. semantic image segmentation.

Visual feature detectors and descriptors have become very popular research topics in the computer vision and robotics communities. They have been applied in many application domains [15], such as image classification [16], 3D scene reconstruction [17], object recognition [18] and visual tracking [19]. The matched visual feature correspondences between two (or more) images can be utilized to establish image relationships [15]. The most well-known visual features are scale-invariant feature transform (SIFT) [20], speeded up robust feature (SURF) [21], oriented FAST and rotated BRIEF (ORB) [22], binary robust invariant scalable keypoints (BRISK) [23], and so forth.

The digital images captured by cameras are essentially 2D [24]. In order to extrapolate the 3D information from a given driving scene, images from multiple views are required [25]. These images can be captured using either a single moving camera [26] or an array of synchronized cameras, as shown in Fig. 2. The former is typically known as *structure from motion (SfM)* [27] or *optical flow* [26], while the latter is typically referred to as *stereo vision* or *binocular vision* (in case two cameras are used) [24]. SfM methods estimate both camera poses and the 3D points of interest from images captured from multiple views, which are linked by a collection of visual features. They also leverage bundle adjustment (BA) [28] technique to refine the estimated camera poses and 3D point locations, by minimizing a cost function known as total re-projection error [29]. Optical flow describes the motion of pixels between consecutive frames of a video sequence [29]. It is also regarded as an effective tool for dynamic object detection [26]. Stereo vision acquires depth information by finding the horizontal positional differences (disparities) of the visual feature

Fig. 3 Object detection/recognition

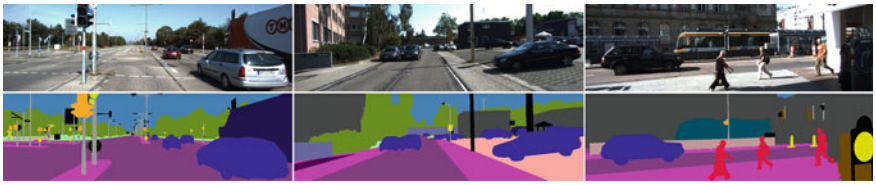

Fig. 4 Semantic image segmentation

correspondence pairs between two synchronously captured images. More details on computer stereo vision will be given in Sect. 4.3.

Object detection/recognition refers to the recognition and localization of particular objects in images/videos [30]. It can be used in various autonomous car perception subtasks, such as pedestrian detection [31], vehicle detection [29], traffic sign detection [32], cyclist detection [33], etc., as shown in Fig. 3. Object detection approaches can be classified as either computer vision-based or machine/deep learning-based. The former typically consists of three steps [30]: (1) informative region selection (scanning the whole image by sliding windows with particular templates to produce candidate regions); (2) visual feature extraction, as discussed above; and (3) object classification (distinguishing a target object from all the other categories using a classifier). With recent advances in machine/deep learning, a large number of convolutional neural networks (CNNs) have been proposed to recognize objects from images/videos. Such CNN-based approaches have achieved very impressive results. The most popular ones include: regions with CNN features (R-CNN) [34], fast R-CNN [35], faster R-CNN [36], you only look once (YOLO) [37], YOLOv3 [38], YOLOv4 [39], etc.

Semantic image segmentation labels every pixel in the image with a given object class [40], such as lane marking, vehicle, collision-free space, or pedestrian, as illustrated in Fig. 4. The state-of-the-art semantic image segmentation approaches are mainly categorized into two main groups [41]: (1) single-modal and (2) data-fusion. The former typically segments RGB images with an encoder-decoder CNN architecture [42]. In recent years, many popular single-model semantic image segmentation algorithms, such as Fully Convolutional Network (FCN) [43], U-Net [44], SegNet [45], DeepLabv3+ [46], DenseASPP [47], DUpsampling [48], etc., have been proposed. Data-fusion semantic image segmentation approaches generally learn features from two different types of vision data [49], such as RGB and depth images

in FuseNet [50], RGB and surface normal images [51] in SNE-RoadSeg [40], RGB and transformed disparity [52, 53] images in AA-RTFNet [49], or RGB and thermal images in MFNet[54]. The learned feature maps are then fused to provide a better semantic prediction.

Please note: a given autonomous car perception application can always be solved by different types of techniques. For instance, lane marking detection can be formulated as a linear/quadratic/quadruplicate pattern recognition problem [55–57]. On the other hand, it can also be formulated as a semantic image segmentation problem and solved with CNNs.

4 Computer Stereo Vision

4.1 Preliminaries

1. Skew-symmetric matrix
In linear algebra, a *skew-symmetric matrix* $\mathbf{A}$ satisfies the following property: its transpose is identical to its negative, i.e., $\mathbf{A}^\top = -\mathbf{A}$. In 3D computer vision, the skew-symmetric matrix $[\mathbf{a}]_\times$ of a vector $\mathbf{a} = [a_1, a_2, a_3]^\top$ can be written as [25]:

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \tag{1}$$

A skew-symmetric matrix has two important properties:

$$\mathbf{a}^\top [\mathbf{a}]_\times = \mathbf{0}^\top, \quad [\mathbf{a}]_\times \mathbf{a} = \mathbf{0}, \tag{2}$$

where $\mathbf{0} = [0, 0, 0]^\top$ is a zero vector. Furthermore, the cross product of two vectors $\mathbf{a}$ and $\mathbf{b}$ can be formulated as a matrix multiplication process [25]:

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b} = -[\mathbf{b}]_\times \mathbf{a}. \tag{3}$$

These properties are generally used to simplify the equations related to vector cross-product.

2. Lie group SO(3) and SE(3)
A 3D point $\mathbf{x_1} = [x_1, y_1, z_1]^\top \in \mathbb{R}^{3\times 1}$ can be transformed into another 3D point $\mathbf{x_2} = [x_2, y_2, z_2]^\top \in \mathbb{R}^{3\times 1}$ using a rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times 3}$ and a translation vector $\mathbf{t} \in \mathbb{R}^{3\times 1}$:

$$\mathbf{x_2} = \mathbf{R}\mathbf{x_1} + \mathbf{t}. \tag{4}$$

$\mathbf{R}$ satisfies matrix orthogonality:

$$\mathbf{R}\mathbf{R}^\top = \mathbf{R}^\top \mathbf{R} = \mathbf{I} \text{ and } |\det(\mathbf{R})| = 1, \tag{5}$$

where $\mathbf{I}$ is an identity matrix and $\det(\mathbf{R})$ represents the determinant of $\mathbf{R}$. The group containing all rotation matrices is referred to as a *special orthogonal group* and is denoted as SO(3). $\tilde{\mathbf{x}}_1 = [\mathbf{x}_1{}^\top, 1]^\top$ and $\tilde{\mathbf{x}}_2 = [\mathbf{x}_2{}^\top, 1]^\top$, the homogeneous coordinates of $\mathbf{x}_1$ and $\mathbf{x}_2$, can be used to describe rotation and translation, as follows:

$$\tilde{\mathbf{x}}_2 = \mathbf{P}\tilde{\mathbf{x}}_1, \tag{6}$$

where

$$\mathbf{P} = \left[\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline \mathbf{0}^\top & 1 \end{array}\right], \tag{7}$$

$\mathbf{P}$ is a homogeneous transformation matrix.[2] The group containing all homogeneous transformation matrices is referred to as a *special Euclidean group* and is denoted as SE(3).

4.2 Multi-view Geometry

4.2.1 Perspective Camera Model

The perspective (or pinhole) camera model, as illustrated in Fig. 5, is the most common geometric camera model describing the relationship between a 3D point $\mathbf{p}^\mathrm{C} = [x^\mathrm{C}, y^\mathrm{C}, z^\mathrm{C}]^\top$ in the camera coordinate system (CCS) and its projection $\bar{\mathbf{p}} = [x, y, f]^\top$ on the image plane Π. $\mathbf{o}^\mathrm{C}$ is the camera center. The distance between Π and $\mathbf{o}^\mathrm{C}$ is the camera focal length f. $\hat{\mathbf{p}}^\mathrm{C} = [\frac{x^\mathrm{C}}{z^\mathrm{C}}, \frac{y^\mathrm{C}}{z^\mathrm{C}}, 1]^\top$ are the normalized coordinates of $\mathbf{p}^\mathrm{C} = [x^\mathrm{C}, y^\mathrm{C}, z^\mathrm{C}]^\top$. Optical axis is the ray originating from $\mathbf{o}^\mathrm{C}$ and passing perpendicularly through Π. The relationship between $\mathbf{p}^\mathrm{C}$ and $\bar{\mathbf{p}}$ is as follows [58]:

$$\bar{\mathbf{p}} = f\hat{\mathbf{p}}^\mathrm{C} = \frac{f}{z^\mathrm{C}}\mathbf{p}^\mathrm{C}. \tag{8}$$

4.2.2 Intrinsic Matrix

Since lens distortion does not exist in a perspective camera model, $\bar{\mathbf{p}} = [x, y, f]^\top$ on the image plane Π can be transformed into a pixel $\mathbf{p} = [u, v]^\top$ in the image using [24]:

$$u = u_o + s_x x, \quad v = v_o + s_y y, \tag{9}$$

[2] https://seas.upenn.edu/~meam620/slides/kinematicsI.pdf.

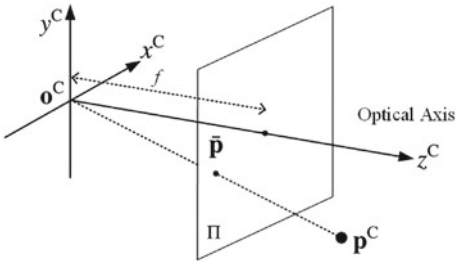

Fig. 5 Perspective camera model

where $\mathbf{p}_o = [u_o, v_o]^\top$ is the principal point; and s_x and s_y are the effective size measured (in pixels per millimeter) in the horizontal and vertical directions, respectively [58]. To simplify the expression of the camera intrinsic matrix $\mathbf{K}$, two notations $f_x = fs_x$ and $f_y = fs_y$ are introduced. u_o, v_o, f, s_x and s_y [25] are five camera intrinsic parameters. Combining (8) and (9), a 3D point $\mathbf{p}^C$ in the CCS can be transformed into a pixel $\mathbf{p}$ in the image using [59]:

$$\tilde{\mathbf{p}} = \frac{1}{z^C}\mathbf{K}\mathbf{p}^C = \frac{1}{z^C}\begin{bmatrix} f_x & 0 & u_o \\ 0 & f_y & v_o \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x^C \\ y^C \\ z^C \end{bmatrix}, \tag{10}$$

where $\tilde{\mathbf{p}} = [\mathbf{p}^\top, 1]^\top = [u, v, 1]^\top$ denotes the homogeneous coordinates of $\mathbf{p} = [u, v]^\top$. Plugging (10) into (8) results in:

$$\hat{\mathbf{p}}^C = \mathbf{K}^{-1}\tilde{\mathbf{p}} = \frac{\bar{\mathbf{p}}}{f} = \frac{\mathbf{p}^C}{z^C}. \tag{11}$$

Therefore, an arbitrary 3D point lying on the ray, which goes from $\mathbf{o}^C$ and through $\mathbf{p}^C$, is always projected at $\bar{\mathbf{p}}$ on the image plane.

4.2.3 Lens Distortion

In order to get better imaging results, a lens is usually installed in front of the camera [58]. However, this introduces image distortions. The optical aberration caused by the installed lens typically deforms the physically straight lines provided by projective geometry to curves in the images [60], as shown in Fig. 6a. We can observe in Fig. 6b that the bent checkerboard grids become straight when the lens distortion is corrected.

Lens distortion can be categorized into two main types: (1) radial distortion and (2) tangential distortion [24]. The presence of radial distortion is due to the fact that geometric lens shape affects straight lines. Tangential distortion occurs because the lens is not perfectly parallel to the image plane [58]. In practical experiments, the image geometry is affected by radial distortion to a much higher extent than by

Fig. 6 Distorted image correction: **a** original image; **b** corrected image

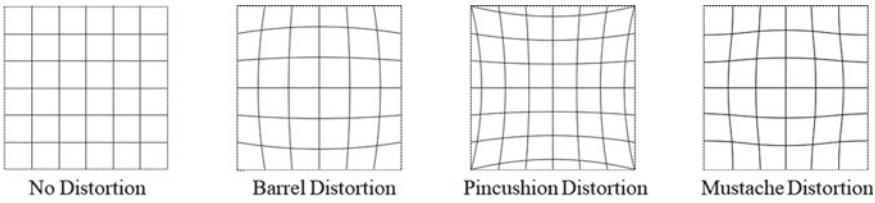

Fig. 7 Radial distortion types

tangential distortion. Therefore, the latter is sometimes neglected in the process of distorted image correction.

Radial distortion

Radial distortion mainly includes (1) barrel distortion, (2) pincushion distortion and (3) mustache distortion, as illustrated in Fig. 7. It can be observed that (a) radial distortions are symmetric about the image center and (b) straight lines are no longer preserved. In barrel distortion, the image magnification decreases with the distance from the optical axis (lines curve outwards). In contrast, the pincushion distortion pinches the image (lines curve inwards). Mustache distortion is a mixture of the above two distortion types. It starts out as barrel distortion close to the optical axis and gradually turns into pincushion distortion close to image periphery. Barrel distortion is commonly applied in fish-eye lenses to produce wide-angle/panoramic images, while pincushion distortion is often associated with telephoto/zoom lenses. Radial distortions can be corrected using[3]:

$$\begin{aligned} x_{\text{undist}} &= x_{\text{dist}}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6), \\ y_{\text{undist}} &= y_{\text{dist}}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6), \end{aligned} \tag{12}$$

[3] https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html.

where the corrected point will be $\mathbf{p}_{\text{undist}} = [x_{\text{undist}}, y_{\text{undist}}]^\top$; ${r^2} = {x_{\text{dist}}}^2 + {y_{\text{dist}}}^2$; $x_{\text{dist}} = \frac{x^{\text{C}}}{z^{\text{C}}} = \frac{u-u_o}{f_x}$ and $y_{\text{dist}} = \frac{y^{\text{C}}}{z^{\text{C}}} = \frac{v-v_o}{f_y}$[4] can be obtained from the distorted image. k_1, k_2 and k_3 are three intrinsic parameters used for radial distortion correction. They can be estimated using a collection of images containing a planar checkerboard pattern.

Tangential distortion

Similar to radial distortion, tangential distortion can also be corrected using:

$$\begin{aligned} x_{\text{undist}} &= x_{\text{dist}} + \left[2p_1 x_{\text{dist}} y_{\text{dist}} + p_2(r^2 + 2{x_{\text{dist}}}^2)\right], \\ y_{\text{undist}} &= y_{\text{dist}} + \left[p_1(r^2 + 2{y_{\text{dist}}}^2) + 2p_2 x_{\text{dist}} y_{\text{dist}}\right], \end{aligned} \tag{13}$$

where p_1 and p_2 are two intrinsic parameters, which can also be estimated using a collection of images containing a planar checkerboard pattern.

4.2.4 Epipolar Geometry

The generic geometry of stereo vision is known as *epipolar geometry*. An example of the epipolar geometry is shown in Fig. 8. Π_{L} and Π_{R} represent the left and right image planes, respectively. $\mathbf{o}_{\text{L}}^{\text{C}}$ and $\mathbf{o}_{\text{R}}^{\text{C}}$ denote the origins of the left camera coordinate system (LCCS) and the right camera coordinate system (RCCS), respectively. The 3D point $\mathbf{p}^{\text{W}} = [x^{\text{W}}, y^{\text{W}}, z^{\text{W}}]^\top$ in the WCS, is projected at $\bar{\mathbf{p}_{\text{L}}} = [x_{\text{L}}, y_{\text{L}}, f_{\text{L}}]^\top$ on Π_{L} and at $\bar{\mathbf{p}_{\text{R}}} = [x_{\text{R}}, y_{\text{R}}, f_{\text{R}}]^\top$ on Π_{R}, respectively. f_{L} and f_{R} are the focal lengths of the left and right cameras, respectively; The representations of $\mathbf{p}^{\text{W}}$ in the LCCS and RCCS are $\mathbf{p}_{\text{L}}^{\text{C}} = [x_{\text{L}}^{\text{C}}, y_{\text{L}}^{\text{C}}, z_{\text{L}}^{\text{C}}]^\top = \frac{z_{\text{L}}^{\text{C}}}{f_{\text{L}}}\bar{\mathbf{p}_{\text{L}}}$ and $\mathbf{p}_{\text{R}}^{\text{C}} = [x_{\text{R}}^{\text{C}}, y_{\text{R}}^{\text{C}}, z_{\text{R}}^{\text{C}}]^\top = \frac{z_{\text{R}}^{\text{C}}}{f_{\text{R}}}\bar{\mathbf{p}_{\text{R}}}$, respectively. According to (4), $\mathbf{p}_{\text{L}}^{\text{C}}$ can be transformed into $\mathbf{p}_{\text{R}}^{\text{C}}$ using:

$$\mathbf{p}_{\text{R}}^{\text{C}} = \mathbf{R}\mathbf{p}_{\text{L}}^{\text{C}} + \mathbf{t}, \tag{14}$$

where $\mathbf{R} \in \mathbb{R}^{3\times3}$ is a rotation matrix and $\mathbf{t} \in \mathbb{R}^{3\times1}$ is a translation vector. $\mathbf{e}_{\text{L}}^{\text{C}}$ and $\mathbf{e}_{\text{R}}^{\text{C}}$ denote the left and right *epipoles*, respectively. The *epipolar plane* is uniquely defined by $\mathbf{o}_{\text{L}}^{\text{C}}$, $\mathbf{o}_{\text{R}}^{\text{C}}$ and $\mathbf{p}^{\text{W}}$. It intersects Π_{L} and Π_{R} giving rise to two *epipolar lines*, as shown in Fig. 8. Using (11), $\mathbf{p}_{\text{L}}^{\text{C}}$ and $\mathbf{p}_{\text{R}}^{\text{C}}$ can be normalized using:

$$\hat{\mathbf{p}}_{\text{L}}^{\text{C}} = \frac{\mathbf{p}_{\text{L}}^{\text{C}}}{z_{\text{L}}^{\text{C}}} = {\mathbf{K}_{\text{L}}}^{-1}\tilde{\mathbf{p}}_{\text{L}}, \quad \hat{\mathbf{p}}_{\text{R}}^{\text{C}} = \frac{\mathbf{p}_{\text{R}}^{\text{C}}}{Z_{\text{R}}^{\text{C}}} = {\mathbf{K}_{\text{R}}}^{-1}\tilde{\mathbf{p}}_{\text{R}}, \tag{15}$$

[4] https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html.

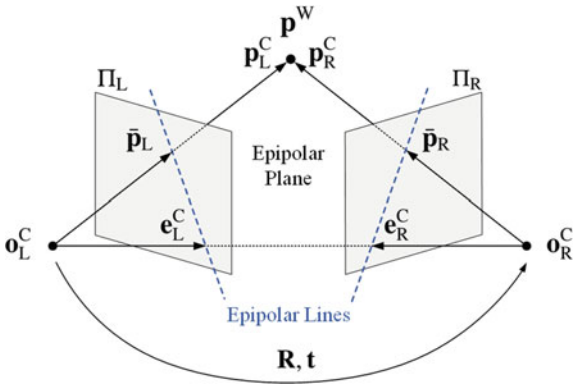

Fig. 8 Epipolar geometry

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

4.2.5 Essential Matrix

Essential matrix $\mathbf{E} \in \mathbb{R}^{3\times3}$ was first introduced by Longuet-Higgins in 1981 [61]. A simple way of introducing the defining equation of $\mathbf{E}$ is to multiply both sides of (14) by $\mathbf{p}_{\mathrm{R}}^{\mathrm{C}\top}[\mathbf{t}]_{\times}$:

$$\mathbf{p}_{\mathrm{R}}^{\mathrm{C}\top}[\mathbf{t}]_{\times}\mathbf{p}_{\mathrm{R}}^{\mathrm{C}} = \mathbf{p}_{\mathrm{R}}^{\mathrm{C}\top}[\mathbf{t}]_{\times}(\mathbf{R}\mathbf{p}_{\mathrm{L}}^{\mathrm{C}} + \mathbf{t}). \tag{16}$$

According to (3), (16) can be rewritten as follows:

$$-\mathbf{p}_{\mathrm{R}}^{\mathrm{C}\top}[\mathbf{p}_{\mathrm{R}}^{\mathrm{C}}]_{\times}\mathbf{t} = \mathbf{p}_{\mathrm{R}}^{\mathrm{C}\top}[\mathbf{t}]_{\times}\mathbf{R}\mathbf{p}_{\mathrm{L}}^{\mathrm{C}} + \mathbf{p}_{\mathrm{R}}^{\mathrm{C}\top}[\mathbf{t}]_{\times}\mathbf{t}. \tag{17}$$

Applying (2) to (17) yields:

$$\mathbf{p}_{\mathrm{R}}^{\mathrm{C}\top}[\mathbf{t}]_{\times}\mathbf{R}\mathbf{p}_{\mathrm{L}}^{\mathrm{C}} = \mathbf{p}_{\mathrm{R}}^{\mathrm{C}\top}\mathbf{E}\mathbf{p}_{\mathrm{L}}^{\mathrm{C}} = 0, \tag{18}$$

The essential matrix $\mathbf{E}$ is defined by:

$$\mathbf{E} = [\mathbf{t}]_{\times}\mathbf{R} \tag{19}$$

Plugging (15) into (18) results in:

$$\hat{\mathbf{p}}_{\mathrm{R}}^{\mathrm{C}\top}\mathbf{E}\hat{\mathbf{p}}_{\mathrm{L}}^{\mathrm{C}} = 0, \tag{20}$$

which depicts the relationship between each pair of normalized points $\hat{\mathbf{p}}_{\mathrm{R}}^{\mathrm{C}}$ and $\hat{\mathbf{p}}_{\mathrm{L}}^{\mathrm{C}}$ lying on the same epipolar plane. It is important to note here that $\mathbf{E}$ has five degrees of freedom: both $\mathbf{R}$ and $\mathbf{t}$ have three degrees of freedom, but the overall scale ambiguity causes the degrees of freedom to be reduced by one [25]. Hence, in theory, $\mathbf{E}$ can be estimated with at least five pairs of $\mathbf{p}_{\mathrm{L}}^{\mathrm{C}}$ and $\mathbf{p}_{\mathrm{R}}^{\mathrm{C}}$. However, due to the non-linearity of $\mathbf{E}$, its estimation using five pairs of correspondences is always intractable. Therefore, $\mathbf{E}$ is commonly estimated with at least eight pairs of $\mathbf{p}_{\mathrm{L}}^{\mathrm{C}}$ and $\mathbf{p}_{\mathrm{R}}^{\mathrm{C}}$ [58], as discussed in Sect. 4.2.6.

4.2.6 Fundamental Matrix

As introduced in Sect. 4.2.5, the essential matrix creates a link between a given pair of corresponding 3D points in the LCCS and RCCS. When the intrinsic matrices of the two cameras in a stereo rig are unknown, the relationship between each pair of corresponding 2D image pixels $\mathbf{p}_{\mathrm{L}} = [u_{\mathrm{L}}, v_{\mathrm{L}}]^{\top}$ and $\mathbf{p}_{\mathrm{R}} = [u_{\mathrm{R}}, v_{\mathrm{R}}]^{\top}$ can be established, using the *fundamental matrix* $\mathbf{F} \in \mathbb{R}^{3\times3}$. It can be considered as a generalization of $\mathbf{E}$, where the assumption of calibrated cameras is removed [25]. Applying (15) to (20) yields:

$$\tilde{\mathbf{p}}_{\mathrm{R}}^{\top} {\mathbf{K}_{\mathrm{R}}}^{-\top} \mathbf{E} {\mathbf{K}_{\mathrm{L}}}^{-1} \tilde{\mathbf{p}}_{\mathrm{L}} = \tilde{\mathbf{p}}_{\mathrm{R}}^{\top} \mathbf{F} \tilde{\mathbf{p}}_{\mathrm{L}} = 0, \tag{21}$$

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

$$\mathbf{F} = {\mathbf{K}_{\mathrm{R}}}^{-\top} \mathbf{E} {\mathbf{K}_{\mathrm{L}}}^{-1}. \tag{22}$$

$\mathbf{F}$ has seven degrees of freedom: a 3×3 homogeneous matrix has eight independent ratios, as there are nine entries, but the common scaling is not significant. However, $\mathbf{F}$ also satisfies the constraint $\det(\mathbf{F}) = 0$, which removes one degree of freedom [25]. The most commonly used algorithm to estimate $\mathbf{E}$ and $\mathbf{F}$ is "eight-point algorithm" (EPA), which was introduced by Hartley in 1997 [62]. This algorithm is based on the scale invariance of $\mathbf{E}$ and $\mathbf{F}$: $\lambda_{\mathbf{E}} {\mathbf{p}_{\mathrm{R}}^{\mathrm{C}}}^{\top} \mathbf{E} \mathbf{p}_{\mathrm{L}}^{\mathrm{C}} = 0$ and $\lambda_{\mathbf{F}} \tilde{\mathbf{p}}_{\mathrm{R}}^{\top} \mathbf{F} \tilde{\mathbf{p}}_{\mathrm{L}} = 0$, where $\lambda_{\mathbf{E}}, \lambda_{\mathbf{F}} \neq 0$. By setting one element in $\mathbf{E}$ and $\mathbf{F}$ to 1, eight unknown elements still need to be estimated. This can be done using at least eight correspondence pairs. If the intrinsic matrices $\mathbf{K}_{\mathrm{L}}$ and $\mathbf{K}_{\mathrm{R}}$ of the two cameras are known, the EPA only needs to be carried out once to estimate either $\mathbf{E}$ or $\mathbf{F}$, because the other one can be easily worked out using (21).

4.2.7 Homography Matrix

An arbitrary 3D point $\mathbf{p}^{\mathrm{W}} = [x^{\mathrm{W}}, y^{\mathrm{W}}, z^{\mathrm{W}}]^{\top}$ lying on a planar surface satisfies:

$$\mathbf{n}^{\top} \mathbf{p}^{\mathrm{W}} + b = 0, \tag{23}$$

where $\mathbf{n} = [n_x, n_y, n_z]^\top$ is the normal vector of the planar surface. Its corresponding pixels $\mathbf{p}_\mathrm{L} = [u_\mathrm{L}, v_\mathrm{L}]^\top$ and $\mathbf{p}_\mathrm{R} = [u_\mathrm{R}, v_\mathrm{R}]^\top$ in the left and right images, respectively, can be linked by a *homography matrix* $\mathbf{H} \in \mathbb{R}^{3\times3}$. The expression of the planar surface can be rearranged as follows:

$$-\mathbf{n}^\top \mathbf{p}^\mathrm{W}/b = 1. \tag{24}$$

Assuming that $\mathbf{p}_\mathrm{L}^\mathrm{C} = \mathbf{p}^\mathrm{W}$ and plugging (24) and (15) into (14) results in:

$$\mathbf{p}_\mathrm{R}^\mathrm{C} = \mathbf{R}\mathbf{p}_\mathrm{L}^\mathrm{C} - \frac{1}{b}\mathbf{t}\mathbf{n}^\top \mathbf{p}_\mathrm{L}^\mathrm{C} = \left(\mathbf{R} - \frac{1}{b}\mathbf{t}\mathbf{n}^\top\right) z_\mathrm{L}^\mathrm{C} \mathbf{K}_\mathrm{L}{}^{-1}\tilde{\mathbf{p}}_\mathrm{L} = z_\mathrm{R}^\mathrm{C}\mathbf{K}_\mathrm{R}{}^{-1}\tilde{\mathbf{p}}_\mathrm{R} \tag{25}$$

Therefore, $\tilde{\mathbf{p}}_\mathrm{L}$ and $\tilde{\mathbf{p}}_\mathrm{R}$ can be linked using:

$$\tilde{\mathbf{p}}_\mathrm{R} = \frac{z_\mathrm{L}^\mathrm{C}}{z_\mathrm{R}^\mathrm{C}}\mathbf{K}_\mathrm{R}\left(\mathbf{R} - \frac{1}{b}\mathbf{t}\mathbf{n}^\top\right)\mathbf{K}_\mathrm{L}{}^{-1}\tilde{\mathbf{p}}_\mathrm{L} = \mathbf{H}\tilde{\mathbf{p}}_\mathrm{L}. \tag{26}$$

The homography matrix $\mathbf{H}$ is generally used to distinguish obstacles from a planar surface [63]. For a well-calibrated stereo vision system, $\mathbf{R}$, $\mathbf{t}$, $\mathbf{K}_\mathrm{L}$ as well as $\mathbf{K}_\mathrm{R}$ are already known, and z_L^C is typically equal to z_R^C. Thus, $\mathbf{H}$ only relates to $\mathbf{n}$ and b, and it can be estimated with at least four pairs of correspondences $\mathbf{p}_\mathrm{L}$ and $\mathbf{p}_\mathrm{R}$ [63].

4.3 Stereopsis

4.3.1 Stereo Rectification

3D scene geometry reconstruction with a pair of synchronized cameras is based on determining pairs of correspondence pixels between the left and right images. For an uncalibrated stereo rig, finding the correspondence pairs is a 2D search process (optical flow estimation), which is extremely computationally intensive. If the stereo rig is calibrated, 1D search should be performed along the epipolar lines. An image transformation process, referred to as *stereo rectification*, is always performed beforehand to reduce the dimension of the correspondence pair search. The stereo rectification consists of four main steps [58]:

1. Rotate the left camera by $\mathbf{R}_\mathrm{rect}$ so that the left image plane is parallel to the vector $\mathbf{t}$;
2. Apply the same rotation to the right camera to recover the original epipolar geometry;
3. Rotate the right camera by $\mathbf{R}^{-1}$;
4. Adjust the left and right image scales by allocating an identical intrinsic matrix to both cameras.

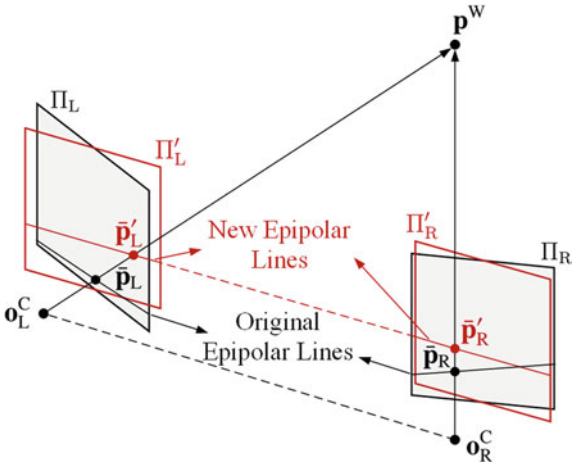

Fig. 9 Stereo rectification

After the stereo rectification, the left and right images appear as if they were taken by a pair of parallel cameras with the same intrinsic parameters, as shown in Fig. 9, where Π_{L} and Π_{R} are the original image planes; Π'_{L} and Π'_{R} are the rectified image planes. Also, each pair of conjugate epipolar lines become collinear and parallel to the horizontal image axis [58]. Hence, determining the correspondence pairs is simplified to a 1D search problem.

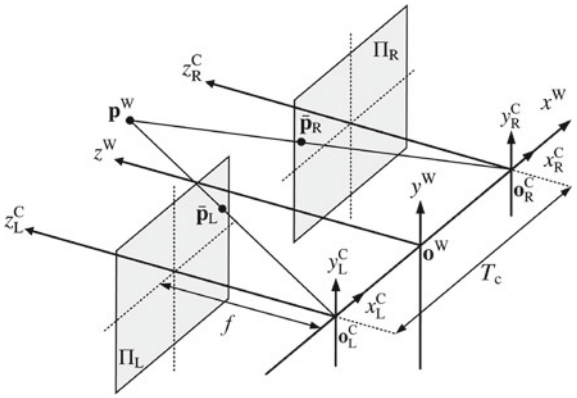

Fig. 10 Basic stereo vision system. $\mathbf{p}^{\mathrm{W}}$ can be transformed to $\bar{\mathbf{p}}^{\mathrm{C}}_{\mathrm{L}}$ and $\bar{\mathbf{p}}^{\mathrm{C}}_{\mathrm{R}}$ using (29)

4.3.2 Stereo Vision System

A well-rectified stereo vision system is illustrated in Fig. 10, which can be regarded as a special epipolar geometry introduced in Sect. 4.2.4, where the left and right cameras are perfectly parallel to each other. $x_{\mathrm{L}}^{\mathrm{C}}$ and $x_{\mathrm{R}}^{\mathrm{C}}$ axes are collinear. $\mathbf{o}_{\mathrm{L}}^{\mathrm{C}}$ and $\mathbf{o}_{\mathrm{L}}^{\mathrm{C}}$ are the left and right camera optical centers, respectively. The baseline of the stereo rig T_c, is defined as the distance between $\mathbf{o}_{\mathrm{L}}^{\mathrm{C}}$ and $\mathbf{o}_{\mathrm{R}}^{\mathrm{C}}$. The intrinsic matrices of the left and right cameras are given by:

$$\mathbf{K}_{\mathrm{L}} = \mathbf{K}_{\mathrm{R}} = \mathbf{K} = \begin{bmatrix} f & 0 & u_o \\ 0 & f & v_o \\ 0 & 0 & 1 \end{bmatrix}, \tag{27}$$

respectively. Let $\mathbf{p}^{\mathrm{W}} = [x^{\mathrm{W}}, y^{\mathrm{W}}, z^{\mathrm{W}}]^\top$ be a 3D point of interest in the WCS. Its representations in the LCCS and RCCS are $\mathbf{p}_{\mathrm{L}}^{\mathrm{C}} = [x_{\mathrm{L}}^{\mathrm{C}}, y_{\mathrm{L}}^{\mathrm{C}}, z_{\mathrm{L}}^{\mathrm{C}}]^\top$ and $\mathbf{p}_{\mathrm{R}}^{\mathrm{C}} = [x_{\mathrm{R}}^{\mathrm{C}}, y_{\mathrm{R}}^{\mathrm{C}}, z_{\mathrm{R}}^{\mathrm{C}}]^\top$, respectively. Since the left and right cameras are considered to be exactly the same in a well-rectified stereo vision system, s_x and s_y in (9) are simply set to 1 and $f_x = f_y = f$. $\mathbf{p}^{\mathrm{W}}$ is projected on Π_{L} and Π_{R} at $\bar{\mathbf{p}}_{\mathrm{L}} = [x_{\mathrm{L}}, y_{\mathrm{L}}, f]^\top$ and $\bar{\mathbf{p}}_{\mathrm{R}} = [x_{\mathrm{R}}, y_{\mathrm{R}}, f]^\top$, respectively. $\mathbf{o}^{\mathrm{W}}$, the origin of the WCS, is at the center of the line segment $L = \{t\mathbf{o}_{\mathrm{L}}^{\mathrm{C}} + (1-t)\mathbf{o}_{\mathrm{R}}^{\mathrm{C}} \mid t \in [0, 1]\}$. z^{W} axis is parallel to the camera optical axes and perpendicular to Π_{L} and Π_{R}. Therefore, an arbitrary point $\mathbf{p}^{\mathrm{W}}$ in the WCS can be transformed to $\mathbf{p}_{\mathrm{L}}^{\mathrm{C}}$ and $\mathbf{p}_{\mathrm{R}}^{\mathrm{C}}$ using:

$$\mathbf{p}_{\mathrm{L}}^{\mathrm{C}} = \mathbf{I}\mathbf{p}^{\mathrm{W}} + \mathbf{t}_{\mathrm{L}}, \quad \mathbf{p}_{\mathrm{R}}^{\mathrm{C}} = \mathbf{I}\mathbf{p}^{\mathrm{W}} + \mathbf{t}_{\mathrm{R}}, \tag{28}$$

where $\mathbf{t}_{\mathrm{L}} = [\frac{T_c}{2}, 0, 0]^\top$ and $\mathbf{t}_{\mathrm{R}} = [-\frac{T_c}{2}, 0, 0]^\top$; Applying (11) and (15) to (28) results in the following expressions:

$$\begin{aligned} x_{\mathrm{L}} &= f\frac{x^{\mathrm{W}} + T_c/2}{z^{\mathrm{W}}}, \quad y_{\mathrm{L}} = f\frac{y^{\mathrm{W}}}{z^{\mathrm{W}}}, \\ x_{\mathrm{R}} &= f\frac{x^{\mathrm{W}} - T_c/2}{z^{\mathrm{W}}}, \quad y_{\mathrm{R}} = f\frac{y^{\mathrm{W}}}{z^{\mathrm{W}}}. \end{aligned} \tag{29}$$

Applying (29) to (9) yields the following expressions:

$$\mathbf{p}_{\mathrm{L}} = \begin{bmatrix} u_{\mathrm{L}} \\ v_{\mathrm{L}} \end{bmatrix} = \begin{bmatrix} f\frac{x^{\mathrm{W}}}{z^{\mathrm{W}}} + u_o + f\frac{T_c}{2z^{\mathrm{W}}} \\ f\frac{y^{\mathrm{W}}}{z^{\mathrm{W}}} + v_o \end{bmatrix}, \quad \mathbf{p}_{\mathrm{R}} = \begin{bmatrix} u_{\mathrm{R}} \\ v_{\mathrm{R}} \end{bmatrix} = \begin{bmatrix} f\frac{x^{\mathrm{W}}}{z^{\mathrm{W}}} + u_o - f\frac{T_c}{2z^{\mathrm{W}}} \\ f\frac{y^{\mathrm{W}}}{z^{\mathrm{W}}} + v_o \end{bmatrix}. \tag{30}$$

The relationship between the so-called disparity d and depth z^{W} is as follows [24]:

$$d = u_{\mathrm{L}} - u_{\mathrm{R}} = f\frac{T_c}{z^{\mathrm{W}}}. \tag{31}$$

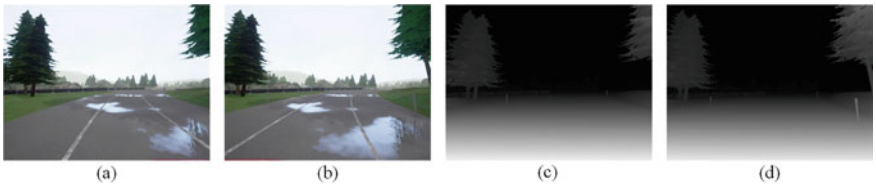

Fig. 11 **a** left image, **b** right image, **c** left disparity image $\mathbf{D}_{\mathrm{L}}$ and (d) right disparity image $\mathbf{D}_{\mathrm{R}}$

It can be observed that d is inversely proportional to z^{W}. Therefore, for a distant 3D point $\mathbf{p}^{\mathrm{W}}$, $\mathbf{p}_{\mathrm{L}}$ and $\mathbf{p}_{\mathrm{R}}$ are close to each other. On the other hand, when $\mathbf{p}^{\mathrm{W}}$ lies near the stereo camera rig, the position difference between $\mathbf{p}_{\mathrm{L}}$ and $\mathbf{p}_{\mathrm{R}}$ is large. Therefore, disparity estimation can be regarded as a task of (1) finding the correspondence ($\mathbf{p}_{\mathrm{L}}$ and $\mathbf{p}_{\mathrm{R}}$) pairs, which are on the same image row, on the left and right images and (2) producing two disparity images $\mathbf{D}_{\mathrm{L}}$ and $\mathbf{D}_{\mathrm{R}}$, as shown in Fig. 11.

4.3.3 Disparity Estimation

The two key aspects of computer stereo vision are speed and accuracy [64]. Over the past two decades, a lot of research has been carried out to improve disparity estimation accuracy while reducing computational complexity. However, the stereo vision algorithms designed to achieve better disparity accuracy typically have higher computational complexity [24]. Hence, speed and accuracy are two desirable but conflicting properties. It is very challenging to achieve both of them simultaneously [64].

In general, the main motivation of designing a stereo vision algorithm is to improve the trade-off between speed and accuracy. In most circumstances, a desirable trade-off entirely depends on the target application [64]. For instance, a real-time performance is required for stereo vision systems employed in autonomous driving, because other systems, such as data-fusion semantic driving scene segmentation, usually take up only a small portion of the processing time, and can be easily implemented in real-time if the 3D information is available [24]. Although stereo vision execution time can definitely be reduced with future HW advances, algorithm and SW improvements are also very important [64].

State-of-the-art stereo vision algorithms can be classified as either computer vision-based or machine/deep learning-based. The former typically formulates disparity estimation as a local block matching problem or a global energy minimization problem [63], while the latter basically considers disparity estimation as a regression problem [65].

Computer vision-based stereo vision algorithms

Computer vision-based disparity estimation algorithms are categorized as: (1) local, (2) global and (3) semi-global [66]. Local algorithms simply select an image block from the left image and match it with a series of image blocks selected from the right image. Optimal disparity estimation corresponds to either the lowest difference costs or the highest correlation costs. Global algorithms translate disparity estimation into a probability maximization problem or an energy minimization problem [67], which can be solved using Markov random field (MRF)-based optimization algorithms [68]. Semi-global matching (SGM) [69] approximates MRF inferences by performing cost aggregation along all image directions, which greatly improves both disparity estimation accuracy and efficiency. Generally, a computer vision-based disparity estimation algorithm consists of four main steps: (1) cost computation, (2) cost aggregation, (3) disparity optimization and (4) disparity refinement [70].

1. Cost Computation

Disparity d is a random variable with N possible discrete states, each of them being associated with a matching cost c. The two most commonly used pixel-wise matching costs are the absolute difference (AD) cost c_{AD} and the squared difference (SD) cost c_{SD} [70]. Since the left and right images are typically in gray-scale format, c_{AD} and c_{SD} can be computed using [71]:

$$\begin{aligned} c_{\mathrm{AD}}(\mathbf{p}, d) &= \left| i_{\mathrm{L}}(\mathbf{p}) - i_{\mathrm{R}}(\mathbf{p} - \mathbf{d}) \right|, \\ c_{\mathrm{SD}}(\mathbf{p}, d) &= \left(i_{\mathrm{L}}(\mathbf{p}) - i_{\mathrm{R}}(\mathbf{p} - \mathbf{d}) \right)^2, \end{aligned} \tag{32}$$

where $\mathbf{d} = [d, 0]^\top$, $i_{\mathrm{L}}(\mathbf{p})$ denotes the pixel intensity of $\mathbf{p} = [u, v]^\top$ in the left image and $i_{\mathrm{R}}(\mathbf{p} - \mathbf{d})$ represents the pixel intensity of $\mathbf{p} - \mathbf{d} = [u - d, v]^\top$ in the right image.

2. Cost Aggregation

In order to minimize incorrect matches, pixel-wise difference costs are often aggregated over all pixels within a support region [66]:

$$c_{\mathrm{agg}}(\mathbf{p}, d) = w(\mathbf{p}, d) * C(\mathbf{p}, d), \tag{33}$$

where the center of the support region is at $\mathbf{p} = [u, v]^\top$. The corresponding disparity is d. c_{agg} denotes the aggregated cost. w is a kernel that represents the support region. C represents a neighborhood system containing the pixel-wise matching costs of all pixels within the support region. c_{agg} can be obtained by performing a convolution between w and C. A large support region can help reduce disparity optimization uncertainties, but also increase the algorithm execution time significantly.

Since the support regions are always rectangular blocks, these algorithms are also known as *stereo block matching* [63]. When the convolution process is a uniform box filtering (all the elements in w are (1), the aggregations of c_{AD} and c_{SD} are referred to as the sum of absolute difference (SAD) and the sum of squared difference (SSD), respectively, which can be written as [24]:

$$\begin{aligned} c_{\mathrm{SAD}}(\mathbf{p}, d) &= \sum_{\mathbf{q}\in\mathscr{N}_{\mathbf{p}}} \left| i_{\mathrm{L}}(\mathbf{q}) - i_{\mathrm{R}}(\mathbf{q}-\mathbf{d}) \right|, \\ c_{\mathrm{SSD}}(\mathbf{p}, d) &= \sum_{\mathbf{q}\in\mathscr{N}_{\mathbf{p}}} \left(i_{\mathrm{L}}(\mathbf{q}) - i_{\mathrm{R}}(\mathbf{q}-\mathbf{d}) \right)^2, \end{aligned} \tag{34}$$

where $\mathscr{N}_{\mathbf{p}}$ is the support region (or neighborhood system) of $\mathbf{p}$. Although the SAD and the SSD are computationally efficient, they are very sensitive to image intensity noise. In this regard, some other cost or similarity functions, such as the normalized

$$c_{\mathrm{NCC}}(\mathbf{p}, d) = \frac{1}{n\sigma_{\mathrm{L}}\sigma_{\mathrm{R}}} \sum_{\mathbf{q}\in\mathscr{N}_{\mathbf{p}}} \Big(i_{\mathrm{L}}(\mathbf{q}) - \mu_{\mathrm{L}} \Big) \Big(i_{\mathrm{R}}(\mathbf{q}-\mathbf{d}) - \mu_{\mathrm{R}} \Big), \tag{35}$$

where

$$\sigma_{\mathrm{L}} = \sqrt{\sum_{\mathbf{q}\in\mathscr{N}_{\mathbf{p}}} \Big(i_{\mathrm{L}}(\mathbf{q}) - \mu_{\mathrm{L}} \Big)^2 / n}, \quad \sigma_{\mathrm{R}} = \sqrt{\sum_{\mathbf{q}\in\mathscr{N}_{\mathbf{p}}} \Big(i_{\mathrm{R}}(\mathbf{q}-\mathbf{d}) - \mu_{\mathrm{R}} \Big)^2 / n}, \tag{36}$$

μ_{L} and μ_{R} represent the means of the pixel intensities within the left and right image block, respectively. σ_{L} and σ_{R} denote the standard deviations of the left and right image block, respectively. n represents the number of pixels within each image blocks. The NCC cost $c_{\mathrm{NCC}} \in [-1, 1]$ reflects the similarity between the given pair of left and right image blocks. A higher c_{NCC} corresponds to a better block matching.

In addition to the cost aggregation via uniform box filtering, many adaptive cost aggregation strategies have been proposed to improve disparity accuracy. One of the most famous algorithms is fast bilateral stereo (FBS) [59, 72], which uses a bilateral filter to aggregate the matching costs adaptively. A general expression of cost aggregation in FBS is as follows:

$$c_{\mathrm{agg}}(\mathbf{p}, d) = \frac{\sum_{\mathbf{q}\in\mathscr{N}_{\mathbf{q}}} \omega_d(\mathbf{q})\omega_r(\mathbf{q})c(\mathbf{q}, d)}{\sum_{\mathbf{q}\in\mathscr{N}_{\mathbf{q}}} \omega_d(\mathbf{q})\omega_r(\mathbf{q})}, \tag{37}$$

where functions ω_d and ω_r are based on spatial distance and color similarity, respectively [24]. The costs c within a rectangular block are aggregated adaptively to produce c_{agg}.

3. Disparity Optimization

The local algorithms simply select the disparities that correspond to the lowest difference costs or the highest correlation costs as the best disparities in a Winner-Take-All (WTA) way.

Unlike WTA applied in the local algorithms, matching costs from neighboring pixels are also taken into account in the global algorithms, e.g., graph cuts (GC) [73] and belief propagation (BP) [74]. The MRF is a commonly used graphical model in such algorithms. An example of the MRF model is depicted in Fig. 12. The graph $\mathscr{G} =$

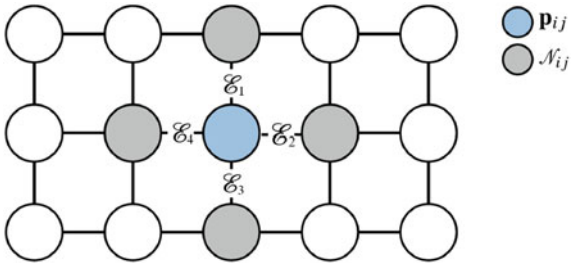

Fig. 12 MRF model

$(\mathscr{P}, \mathscr{E})$ is a set of vertices $\mathscr{P}$ connected by edges $\mathscr{E}$, where $\mathscr{P} = \{\mathbf{p}_{11}, \mathbf{p}_{12}, \ldots, \mathbf{p}_{mn}\}$ and $\mathscr{E} = \{(\mathbf{p}_{ij}, \mathbf{p}_{st}) \mid \mathbf{p}_{ij}, \mathbf{p}_{st} \in \mathscr{P}\}$. Two edges sharing one common vertex are called a pair of adjacent edges [75]. Since the MRF is considered to be undirected, $(\mathbf{p}_{ij}, \mathbf{p}_{st})$ and $(\mathbf{p}_{st}, \mathbf{p}_{ij})$ refer to the same edge here. $\mathscr{N}_{ij} = \{\mathbf{q}_{1_{\mathbf{p}_{ij}}}, \mathbf{q}_{2_{\mathbf{p}_{ij}}}, \ldots, \mathbf{q}_{k_{\mathbf{p}_{ij}}} \mid \mathbf{q}_{\mathbf{p}_{ij}} \in \mathscr{P}\}$ is a neighborhood system of $\mathbf{p}_{ij}$.

For stereo vision problems, $\mathscr{P}$ is a $m \times n$ pixel disparity image and $\mathbf{p}_{ij}$ is a graph vertex (or node) at the site of (i, j) with a disparity node value d_{ij}. Because the consideration of more candidates usually makes true disparity inference intractable, only the neighbors adjacent to $\mathbf{p}_{ij}$ are considered for stereo matching [68], in a pairwise MRF fashion, as the disparity of node $\mathbf{p}_{ij}$ tends to have a strong correlation with its vicinities, while it is linked implicitly to any other random nodes in the disparity map. The joint MRF probability can be written as [68]:

$$P(\mathbf{p}, q) = \prod_{\mathbf{p}_{ij} \in \mathscr{P}} \Phi(\mathbf{p}_{ij}, q_{\mathbf{p}_{ij}}) \prod_{\mathbf{q}_{\mathbf{p}_{ij}} \in \mathscr{N}_{ij}} \Psi(\mathbf{p}_{ij}, \mathbf{q}_{\mathbf{p}_{ij}}), \tag{38}$$

where $q_{\mathbf{p}_{ij}}$ represents image intensity differences, $\Phi(\cdot)$ expresses the compatibility between possible disparities and the corresponding image intensity differences, while $\Psi(\cdot)$ expresses the compatibility between $\mathbf{p}_{ij}$ and its neighborhood system. Now, the aim of finding the best disparity is equivalent to maximizing $P(\mathbf{p}, q)$ in (38), by formulating it as an energy function [59]:

$$E(\mathbf{p}) = \sum_{\mathbf{p}_{ij} \in \mathscr{P}} D(\mathbf{p}_{ij}, q_{\mathbf{p}_{ij}}) + \sum_{\mathbf{q}_{\mathbf{p}_{ij}} \in \mathscr{N}_{ij}} V(\mathbf{p}_{ij}, \mathbf{q}_{\mathbf{p}_{ij}}), \tag{39}$$

where $D(\cdot)$ and $V(\cdot)$ are two energy functions. $D(\cdot)$ corresponds to the matching cost and $V(\cdot)$ determines the aggregation from the neighbors. In the MRF model, the method to formulate an adaptive $V(\cdot)$ is important, because image intensity in discontinuous areas usually varies greatly from that of its neighbors [24]. However, the process of minimizing (39) results in high computational complexities, rendering real-time performance challenging. Therefore, SGM [69] breaks down (39) into:

$$E(\mathbf{D}) = \sum_{\mathbf{p}} \left(c(\mathbf{p}, d_{\mathbf{p}}) + \sum_{\mathbf{q} \in \mathcal{N}_{\mathbf{p}}} \lambda_1 \delta(|d_{\mathbf{p}} - d_{\mathbf{q}}| = 1) + \sum_{\mathbf{q} \in \mathcal{N}_{\mathbf{p}}} \lambda_2 \delta(|d_{\mathbf{p}} - d_{\mathbf{q}}| > 1) \right), \tag{40}$$

where $\mathbf{D}$ is the disparity image, c is the matching cost, $\mathbf{q}$ is a pixel in the neighborhood system $\mathcal{N}_{\mathbf{p}}$ of $\mathbf{p}$. λ_1 penalizes the neighboring pixels with small disparity differences, i.e., one pixel; λ_2 penalizes the neighboring pixels with large disparity differences, i.e., larger than one pixel. $\delta(\cdot)$ returns 1 if its argument is true and 0 otherwise.

4. Disparity Refinement

Disparity refinement usually involves several post-processing steps, such as the left-and-right disparity consistency check (LRDCC), subpixel enhancement and weighted median filtering [76]. The LRDCC can remove most of the occluded areas, which are only visible in one of the left/right image [63]. In addition, a disparity error larger than one pixel may result in a non-negligible 3D geometry reconstruction error [63]. Therefore, subpixel enhancement provides an easy way to increase disparity image resolution by simply interpolating the matching costs around the initial disparity [76]. Moreover, a median filter can be applied to the disparity image to fill the holes and remove the incorrect matches [76]. However, the above disparity refinement algorithms are not always necessary and the sequential use of these steps depends entirely on the chosen algorithm and application needs.

Machine/deep learning-based stereo vision algorithms

With recent advances in machine/deep learning, CNNs have been prevalently used for disparity estimation. For instance, Žbontar and LeCun [77] utilized a CNN to compute patch-wise similarity scores, as shown in Fig. 13. It consists of a convolutional layer L_1 and seven fully-connected layers L_2–L_8. The inputs to this CNN are two 9×9-pixel gray-scale image patches. L_1 consists of 32 convolution kernels of size $5\times5\times1$. L_2 and L_3 have 200 neurons each. After L_3, the two 200-dimensional feature vectors are concatenated into a 400-dimensional vector and passed through L_4-L_7 layers. Layer L_8 maps L_7 output into two real numbers, which are then fed through a softmax function to produce a distribution over the two classes: a) good match and b) bad match. Finally, they utilize computer vision-based cost aggregation and disparity optimization/refinement techniques to produce the final disparity images. Although this method has achieved the state-of-the-art accuracy, it is limited by the employed matching cost aggregation technique and can produce wrong predictions in occluded or texture-less/reflective regions [78].

In this regard, some researchers have leveraged CNNs to improve computer vision-based cost aggregation step. SGM-Nets [79] is one of the most well-known methods of this type. Its main contribution is a CNN-based technique for predicting SGM penalty parameters λ_1 and λ_2 in (40) [69], as illustrated in Fig. 14. A 5×5-pixel gray-scale image patch and its normalized position are used as the CNN inputs. It has (a) two convolution layers, each followed by a rectified linear unit (ReLU) layer; (b) a concatenate layer for merging the two types of inputs; (c) two fully connected

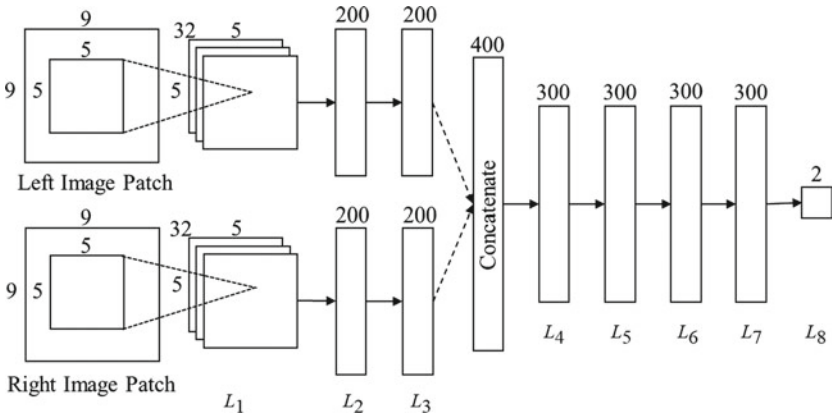

Fig. 13 The architecture of the CNN proposed in [77] for stereo matching

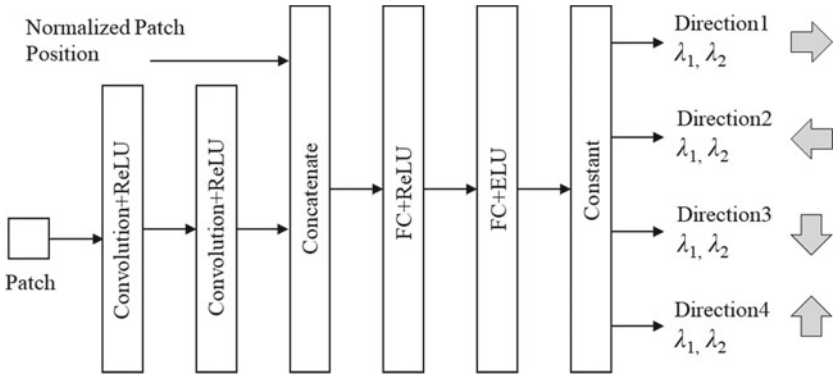

Fig. 14 SGM-Nets [79] architecture

(FC) layers of size 128 each, followed by a ReLU layer and an exponential linear unit (ELU); (d) a constant layer to keep SGM penalty values positive. The costs can then be accumulated along four directions. The CNN output values correspond to standard parameterization.

Recently, end-to-end deep CNNs have become very popular. For example, Mayer et al. [80] created three large synthetic datasets[5] (FlyingThings3D, Driving and Monkaa) and proposed a CNN named DispNet for dense disparity estimation. Later on, Pang et al. [81] proposed a two-stage cascade CNN for disparity estimation. Its the first stage enhances DispNet [80] by equipping it with extra up-convolution modules and the second stage rectifies the disparity initialized by the first stage and generates residual signals across multiple scales. Furthermore, GCNet [82] incorporated feature extraction (cost computation), cost aggregation and disparity optimiza-

[5] https://lmb.informatik.uni-freiburg.de/resources/datasets/SceneFlowDatasets.en.html.

tion/refinement into a single end-to-end CNN model, and it achieved the state-of-the-art accuracy on the FlyingThings3D benchmark [80] as well as the KITTI stereo 2012 and 2015 benchmarks [83–85]. In 2018, Chang et al. [86] proposed Pyramid Stereo Matching Network (PSMNet), consisting of two modules: (a) spatial pyramid pooling and (b) 3D CNN. The former aggregates the context of different scales and locations, while the latter regularizes the cost volume. Unlike PSMNet [86], guided aggregation net (GANet) [78] replaces the widely used 3D CNN with two novel layers: a semi-global aggregation layer and a local guided aggregation layer, which help save a lot of memory and computational cost.

Although the aforementioned CNN-based disparity estimation methods have achieved compelling results, they usually have a huge number of learnable parameters, resulting in a long processing time. Therefore, current state-of-the-art CNN-based disparity estimation algorithms have hardly been put into practical uses in autonomous driving. We believe these methods will be applied in more real-world applications, with future advances in embedded computing HW.

4.3.4 Performance Evaluation

As discussed above, disparity estimation speed and accuracy are two key properties and they are always pitted against each other. Therefore, the performance evaluation of a given stereo vision algorithm usually involves both of these two properties [64].

The following two metrics are commonly used to evaluate the accuracy of an estimated disparity image [87]:

1. Root mean squared (RMS) error e_{RMS}:

$$e_{\text{RMS}} = \sqrt{\frac{1}{N}\sum_{\mathbf{p}\in\mathscr{P}} |\mathbf{D}_{\text{E}}(\mathbf{p}) - \mathbf{D}_{\text{G}}(\mathbf{p})|^2}, \tag{41}$$

2. Percentage of error pixels (PEP) e_{PEP} (tolerance: δ_d pixels):

$$e_{\text{PEP}} = \frac{1}{N}\sum_{\mathbf{p}\in\mathscr{P}} \delta\Big(|\mathbf{D}_{\text{E}}(\mathbf{p}) - \mathbf{D}_{\text{G}}(\mathbf{p})| > \delta_d\Big) \times 100\%, \tag{42}$$

where $\mathbf{D}_{\text{E}}$ and $\mathbf{D}_{\text{G}}$ represent the estimated and ground truth disparity images, respectively; N denotes the total number of disparities used for evaluation; δ_d represents the disparity evaluation tolerance.

Additionally, a general way to depict the efficiency of an algorithm is given in millions of disparity evaluations per second Mde/s [64] as follows:

$$\text{Mde}/s = \frac{u_{\max} v_{\max} d_{\max}}{t} 10^{-6}. \tag{43}$$

However, the speed of a disparity estimation algorithm typically varies across different platforms, and it can be greatly boosted by exploiting the parallel computing architecture.

5 Heterogeneous Computing

Heterogeneous computing systems use multiple types of processors or cores. In the past, heterogeneous computing meant that different instruction-set architectures (ISAs) had to be handled differently, while modern heterogeneous system architecture (HSA) systems allow users to utilize multiple processor types. A typical HSA system consists of two different types of processors: (1) a multi-threading central processing unit (CPU) and (2) a graphics processing unit (GPU) [88], which are connected by a peripheral component interconnect (PCI) express bus. The CPU's memory management unit (MMU) and the GPU's input/output memory management unit (IOMMU) comply with the HSA HW specifications. CPU runs the operating system and performs traditional serial computing tasks, while GPU performs 3D graphics rendering and CNN training.

5.1 Multi-threading CPU

The application programming interface (API) Open Multi-Processing (OpenMP) is typically used to break a serial code into independent chunks for parallel processing. It supports multi-platform shared-memory multiprocessing programming in C/C++ and Fortran [89]. An explicit parallelism programming model, typically known as a fork-join model, is illustrated in Fig. 15, where the compiler instructs a section of the serial code to run in parallel. The master thread (serial execution on one core) forks a number of slave threads. The tasks are divided to run in parallel amongst the slave threads on multiple cores. Synchronization waits until all slave threads finish their allocated tasks. Finally, the slave threads join together at a subsequent point and resume sequential execution.

5.2 GPU

GPUs have been extensively used in computer vision and deep learning to accelerate the computationally intensive but parallelly-efficient processing and CNN training. Compared with a CPU, which consists of a low number of cores optimized for sequentially serial processing, GPU has a highly parallel architecture which is composed of hundreds or thousands of light GPU cores to handle multiple tasks concurrently.

A typical GPU architecture is shown in Fig. 16, which consists of N streaming multiprocessors (SMs) with M streaming processors (SPs) on each of them. The single instruction multiple data (SIMD) architecture allows the SPs on the same SM to execute the same instruction but process different data at each clock cycle. The device has its own dynamic random access memory (DRAM) which consists of global memory, constant memory and texture memory. DRAM can communicate with the host memory via the graphical/memory controller hub (GMCH) and the I/O controller hub (ICH), which are also known as the Intel northbridge and the Intel

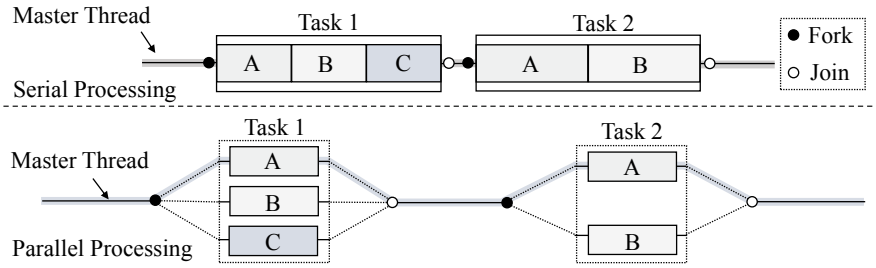

Fig. 15 Serial processing versus parallel processing

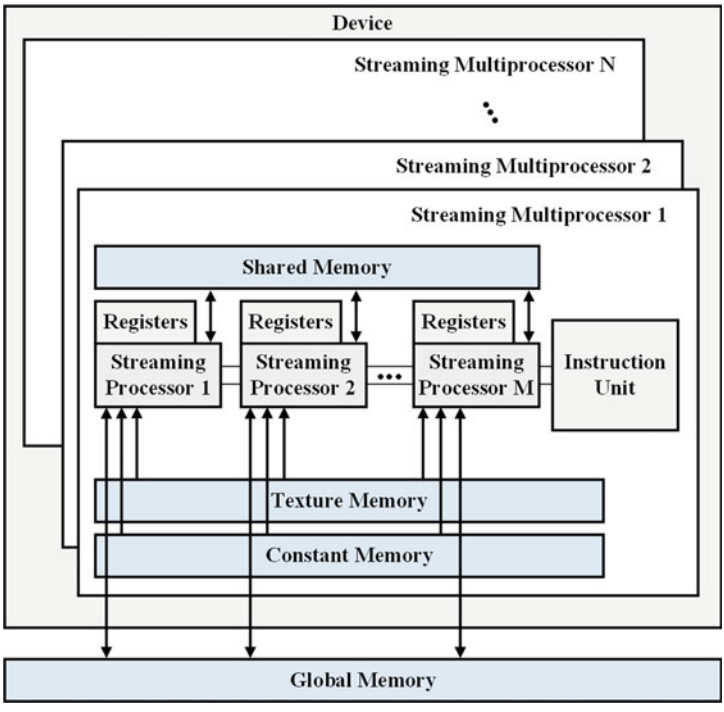

Fig. 16 GPU architecture [24]

southbridge, respectively. Each SM has four types of on-chip memories: register, shared memory, constant cache and texture cache. Since they are on-chip memories, the constant cache and texture cache are utilized to speed up data fetching from the constant memory and texture memory, respectively. Due to the fact that the shared memory is small, it is used for the duration of processing a block. The register is only visible to the thread.

In CUDA C programming, the threads are grouped into a set of 3D thread blocks which are then organized as a 3D grid. The kernels are defined on the host using the CUDA C programming language. Then, the host issues the commands that submit the kernels to devices for execution. Only one kernel can be executed at a given time. Once a thread block is distributed to an SM, the threads are divided into groups of 32 parallel threads which are executed by SPs. Each group of 32 parallel threads is known as a warp. Therefore, the size of a thread block is usually chosen as a multiple of 32 to ensure efficient data processing.

6 Summary

In this chapter, we first introduced the autonomous car system, from both HW aspect (car sensors and car chassis) and SW aspect (perception, localization and mapping, prediction and planning, and control). Particularly, we introduced the autonomous car perception module, which has four main functionalities: (1) visual feature detection, description and matching, (2) 3D information acquisition, (3) object detection/recognition and (4) semantic image segmentation. Later on, we provided readers with the preliminaries for the epipolar geometry and introduced computer stereo vision from theory to algorithms. Finally, heterogeneous computing architecture, consisting of a multi-threading CPU and a GPU, was introduced.

Acknowledgements This work was supported by the National Key R&D Program of China, under grant No. 2020AAA0108100, awarded to Prof. Qijun Chen. This work was also supported by the Fundamental Research Funds for the Central Universities, under projects No. 22120220184, No. 22120220214, and No. 2022-5-YB-08, awarded to Prof. Rui Fan. This work has also received partial funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871479 (AERIALCORE).

References

1. R. Fan, J. Jiao, H. Ye, Y. Yu, I. Pitas, M. Liu, Key ingredients of self-driving cars, in *27th European Signal Processing Conference (EUSIPCO) 2019, Satellite Workshop: Signal Processing, Computer Vision and Deep Learning for Autonomous Systems* (2019), pp. 1–5
2. Lidar–light detection and ranging–is a remote sensing method used to examine the surface of the earth, in *NOAA. Archived from the Original on*, vol. 4 (2013)
3. T. Bureau, Radar definition, in *Public Works and Government Services Canada* (2013)
4. W.J. Westerveld, Silicon photonic micro-ring resonators to sense strain and ultrasound (2014)

5. N. Samama, *Global Positioning: Technologies and Performance*, vol. 7 (Wiley, 2008)
6. S. Liu, Chassis technologies for autonomous robots and vehicles (2019)
7. M.U.M. Bhutta, M. Kuse, R. Fan, Y. Liu, M. Liu, Loop-box: multiagent direct slam triggered by single loop closure for large-scale mapping. IEEE Trans. Cybern. **52**(6), 5088–5097 (2022)
8. R.C. Smith, P. Cheeseman, On the representation and estimation of spatial uncertainty. Int. J. Robot. Res. **5**(4), 56–68 (1986)
9. C. Katrakazas, M. Quddus, W.-H. Chen, L. Deka, Real-time motion planning methods for autonomous on-road driving: state-of-the-art and future research directions. Transp. Res. Part C: Emerg. Technol. **60**, 416–442 (2015)
10. T.H. Cormen, Section 24.3: Dijkstra's algorithm, in *Introduction to Algorithms* (2001), pp. 595–601
11. D. Delling, P. Sanders, D. Schultes, D. Wagner, Engineering route planning algorithms, in *Algorithmics of Large and Complex Networks* (Springer, 2009), pp. 117–139
12. M. Willis, Proportional-integral-derivative control, in *Department of Chemical and Process Engineering University of Newcastle* (1999)
13. G.C. Goodwin, S.F. Graebe, M.E. Salgado et al., *Control System Design* (Prentice Hall, Upper Saddle River, NJ, 2001)
14. C.E. Garcia, D.M. Prett, M. Morari, Model predictive control: theory and practice-a survey. Automatica **25**(3), 335–348 (1989)
15. M. Hassaballah, A.A. Abdelmgeid, H.A. Alshazly, Image features detection, description and matching, in *Image Feature Detectors and Descriptors* (Springer, 2016), pp. 11–45
16. S. Liu, X. Bai, Discriminative features for image classification and retrieval. Pattern Recogn. Lett. **33**(6), 744–751 (2012)
17. P. Moreels, P. Perona, Evaluation of features detectors and descriptors based on 3d objects. Int. J. Comput. Vision **73**(3), 263–284 (2007)
18. P. Dollar, C. Wojek, B. Schiele, P. Perona, Pedestrian detection: an evaluation of the state of the art. IEEE Trans. Pattern Anal. Mach. Intell. **34**(4), 743–761 (2011)
19. M. Danelljan, G. Häger, F.S. Khan, M. Felsberg, Discriminative scale space tracking. IEEE Trans. Pattern Anal. Mach. Intell. **39**(8), 1561–1575 (2016)
20. D.G. Lowe, Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
21. H. Bay, T. Tuytelaars, L. Van Gool, Surf: speeded up robust features, in *ECCV* (Springer, 2006), pp. 404–417
22. E. Rublee, V. Rabaud, K. Konolige, G. Bradski, Orb: an efficient alternative to sift or surf, in *International Conference on Computer Vision* (IEEE, 2011), pp. 2564–2571
23. S. Leutenegger, M. Chli, R.Y. Siegwart, Brisk: binary robust invariant scalable keypoints, in *International Conference on Computer Vision* (IEEE, 2011), pp. 2548–2555
24. R. Fan, Real-time computer stereo vision for automotive applications, Ph.D. dissertation, University of Bristol (2018)
25. R. Hartley, A. Zisserman, *Multiple view Geometry in Computer Vision* (Cambridge University Press, 2003)
26. H. Wang, R. Fan, M. Liu, CoT-AMFlow: adaptive modulation network with co-teaching strategy for unsupervised optical flow estimation, in *Conference on Robot Learning* (PMLR, 2021), pp. 143–155
27. S. Ullman, The interpretation of structure from motion. Proc. R. Soc. Lond. Ser. B. Biol. Sci. **203**(1153), 405–426 (1979)
28. B. Triggs, P.F. McLauchlan, R.I. Hartley, A.W. Fitzgibbon, Bundle adjustment-a modern synthesis, in *International Workshop on Vision Algorithms* (Springer, 1999), pp. 298–372
29. H. Wang, Y. Liu, H. Huang, Y. Pan, W. Yu, J. Jiang, D. Lyu, M.J. Bocus, M. Liu, I. Pitas et al., ATG-PVD: ticketing parking violations on a drone, in *European Conference on Computer Vision Workshops* (Springer, 2020), pp. 541–557
30. Z.-Q. Zhao, P. Zheng, S.-T. Xu, X. Wu, Object detection with deep learning: a review. IEEE Trans. Neural Netw. Learn. Syst. **30**(11), 3212–3232 (2019)

31. D. Wang, C. Devin, Q.-Z. Cai, F. Yu, T. Darrell, Deep object-centric policies for autonomous driving, in *2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019), pp. 8853–8859
32. A. Mogelmose, M.M. Trivedi, T.B. Moeslund, Vision-based traffic sign detection and analysis for intelligent driver assistance systems: perspectives and survey. IEEE Trans. Intell. Transp. Syst. **13**(4), 1484–1497 (2012)
33. B. Wu, F. Iandola, P.H. Jin, K. Keutzer, Squeezedet: unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2017), pp. 129–137
34. R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 580–587
35. R. Girshick, Fast R-CNN, in *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1440–1448
36. S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, in *Advances in Neural Information Processing Systems* (2015), pp. 91–99
37. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 779–788
38. J. Redmon, A. Farhadi, Yolov3: an incremental improvement (2018)
39. A. Bochkovskiy, C.-Y. Wang, H.-Y. M. Liao, Yolov4: optimal speed and accuracy of object detection (2020)
40. R. Fan, H. Wang, P. Cai, M. Liu, SNE-RoadSeg: incorporating surface normal information into semantic segmentation for accurate freespace detection, in *European Conference on Computer Vision* (Springer, 2020), pp. 340–356
41. H. Wang, R. Fan, Y. Sun, M. Liu, 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, 2020), pp. 2706–2711
42. R. Fan, H. Wang, P. Cai, J. Wu, M.J. Bocus, L. Qiao, M. Liu, Learning collision-free space detection from stereo images: homography matrix brings better data augmentation. IEEE/ASME Trans. Mechatron. **27**(1), 225–233 (2021)
43. J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3431–3440
44. O. Ronneberger, P. Fischer, T. Brox, U-net: convolutional networks for biomedical image segmentation, in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (Springer, 2015), pp. 234–241
45. V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: a deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **39**(12), 2481–2495 (2017)
46. L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with Atrous separable convolution for semantic image segmentation, in *ECCV* (2018), pp. 801–818
47. M. Yang, K. Yu, C. Zhang, Z. Li, K. Yang, Denseaspp for semantic segmentation in street scenes, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 3684–3692
48. Z. Tian, T. He, C. Shen, Y. Yan, Decoders matter for semantic segmentation: data-dependent decoding enables flexible feature aggregation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 3126–3135
49. R. Fan, H. Wang, M.J. Bocus, M. Liu, We learn better road pothole detection: from attention aggregation to adversarial domain adaptation, in *European Conference on Computer Vision Workshops* (Springer, 2020), pp. 285–300

50. C. Hazirbas, L. Ma, C. Domokos, D. Cremers, Fusenet: incorporating depth into semantic segmentation via fusion-based CNN architecture, in *Asian Conference on Computer Vision* (Springer, 2016), pp. 213–228
51. R. Fan, H. Wang, B. Xue, H. Huang, Y. Wang, M. Liu, I. Pitas, Three-filters-to-normal: an accurate and ultrafast surface normal estimator. IEEE Robot. Autom. Lett. **6**(3), 5405–5412 (2021)
52. R. Fan, U. Ozgunalp, B. Hosking, M. Liu, I. Pitas, Pothole detection based on disparity transformation and road surface modeling. IEEE Trans. Image Process. **29**, 897–908 (2019)
53. R. Fan, M. Liu, Road damage detection based on unsupervised disparity map segmentation. IEEE Trans. Intell. Transp. Syst. **21**(11), 4906–4911 (2019)
54. Q. Ha, K. Watanabe, T. Karasawa, Y. Ushiku, T. Harada, 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, 2017), pp. 5108–5115
55. U. Ozgunalp, R. Fan, X. Ai, N. Dahnoun, Multiple lane detection algorithm based on novel dense vanishing point estimation. IEEE Trans. Intell. Transp. Syst. **18**(3), 621–632 (2016)
56. R. Fan, N. Dahnoun, Real-time stereo vision-based lane detection system. Meas. Sci. Technol. **29**(7), 074005 (2018)
57. J. Jiao, R. Fan, H. Ma, M. Liu, Using dp towards a shortest path problem-related application, in *2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019), pp. 8669–8675
58. E. Trucco, A. Verri, *Introductory Techniques for 3-D Computer Vision*, vol. 201 (Prentice Hall Englewood Cliffs, 1998)
59. R. Fan, J. Jiao, J. Pan, H. Huang, S. Shen, M. Liu, 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, 2019), pp. 535–543
60. Z. Zhang, A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. **22**(11), 1330–1334 (2000)
61. H.C. Longuet-Higgins, A computer algorithm for reconstructing a scene from two projections. Nature **293**(5828), 133–135 (1981)
62. R.I. Hartley, In defense of the eight-point algorithm. IEEE Trans. Pattern Anal. Mach. Intell. **19**(6), 580–593 (1997)
63. R. Fan, X. Ai, N. Dahnoun, Road surface 3d reconstruction based on dense subpixel disparity map estimation. IEEE Trans. Image Process. **27**(6), 3025–3035 (2018)
64. B. Tippetts, D.J. Lee, K. Lillywhite, J. Archibald, Review of stereo vision algorithms and their suitability for resource-limited systems. J. Real-Time Image Proc. **11**(1), 5–25 (2016)
65. W. Luo, A.G. Schwing, R. Urtasun, Efficient deep learning for stereo matching, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 5695–5703
66. D. Scharstein, R. Szeliski, High-accuracy stereo depth maps using structured light, in *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1 (IEEE, 2003), p. I
67. M.G. Mozerov, J. van de Weijer, Accurate stereo matching by two-step energy minimization. IEEE Trans. Image Process. **24**(3), 1153–1163 (2015)
68. M. F. Tappen, W.T. Freeman, Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters," in *Null* (IEEE, 2003), p. 900
69. H. Hirschmuller, Stereo processing by semiglobal matching and mutual information. IEEE Trans. Pattern Anal. Mach. Intell. **30**(2), 328–341 (2007)
70. J. Žbontar, Y. LeCun, Stereo matching by training a convolutional neural network to compare image patches. J. Mach. Learn. Res. **17**(1), 2287–2318 (2016)
71. H. Hirschmuller, D. Scharstein, Evaluation of stereo matching costs on images with radiometric differences. IEEE Trans. Pattern Anal. Mach. Intell. **31**(9), 1582–1599 (2008)
72. Q. Yang, L. Wang, R. Yang, H. Stewénius, D. Nistér, Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling. IEEE Trans. Pattern Anal. Mach. Intell. **31**(3), 492–504 (2008)

73. Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23**(11), 1222–1239 (2001)
74. A.T. Ihler, A.S. Willsky et al., Loopy belief propagation: Convergence and effects of message errors. J. Mach. Learn. Res. **6**(May), 905–936 (2005)
75. A. Blake, P. Kohli, C. Rother, *Markov Random Fields for Vision and Image Processing* (Mit Press, 2011)
76. D. Scharstein, R. Szeliski, A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Comput. Vision **47**(1–3), 7–42 (2002)
77. J. Žbontar, Y. LeCun, Computing the stereo matching cost with a convolutional neural network, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1592–1599
78. F. Zhang, V. Prisacariu, R. Yang, P.H. Torr, Ga-net: guided aggregation net for end-to-end stereo matching, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 185–194
79. A. Seki, M. Pollefeys, SGM-nets: semi-global matching with neural networks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 231–240
80. N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, T. Brox, 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* (2016), pp. 4040–4048
81. J. Pang, W. Sun, J.S. Ren, C. Yang, Q. Yan, Cascade residual learning: a two-stage convolutional neural network for stereo matching, in *Proceedings of the IEEE International Conference on Computer Vision Workshops* (2017), pp. 887–895
82. A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, A. Bry, End-to-end learning of geometry and context for deep stereo regression, in *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 66–75
83. A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? the Kitti vision benchmark suite, in *2012 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2012), pp. 3354–3361
84. M. Menze, C. Heipke, A. Geiger, Joint 3d estimation of vehicles and scene flow. ISPRS Ann. Photogramm., Remote. Sens. Spat. Inf. Sci. **2** (2015)
85. C. Menze, Moritz; Heipke, A. Geiger, Object scene flow. ISPRS J. Photogramm. Remote. Sens. **140**, 60–76 (2018)
86. J.-R. Chang, Y.-S. Chen, Pyramid stereo matching network, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 5410–5418
87. J.L. Barron, D.J. Fleet, S.S. Beauchemin, Performance of optical flow techniques. Int. J. Comput. Vis. **12**(1), 43–77 (1994)
88. S. Mittal, J.S. Vetter, A survey of CPU-GPU heterogeneous computing techniques. ACM Comput. Surv. (CSUR) **47**(4), 1–35 (2015)
89. H. Jin, D. Jespersen, P. Mehrotra, R. Biswas, L. Huang, B. Chapman, High performance computing using MPI and OpenMP on multi-core parallel systems. Parallel Comput. **37**(9), 562–575 (2011)