1

Affordable and Reliable Autonomous Driving Through Modular Design

1.1 Introduction

In recent years, autonomous driving has become quite a popular topic in the research community as well as in industry, and even in the press, but besides the fact that it is exciting and revolutionary, why should we deploy autonomous vehicles? One reason is that ridesharing using clean-energy autonomous vehicles will completely revolutionize the transportation industry by reducing pollution and traffic problems, by improving safety, and by making our economy more efficient.

More specifically and starting with pollution reduction: there are about 260 million cars in the US today. If we were to convert all cars to clean-energy cars, we would reduce annual carbon emissions by 800 million tons, which would account for 13.3% of the US commitment to the Paris Agreement [1]. Also, with near-perfect scheduling, if ridesharing autonomous vehicles could be deployed, the number of cars could be reduced by 75% [2]. Consequently, these two changes combined have the potential to yield an annual reduction of 1 billion tons in carbon emission, an amount roughly equivalent to 20% of the US Commitment to the Paris Agreement.

As for safety improvement, human drivers have a crash rate of 4.2 accidents per million miles (PMM), while the current autonomous vehicle crash rate is 3.2 crashes PMM [3]. Yet, as the safety of autonomous vehicles continues to improve, if the autonomous vehicle crash rate PMM can be made to drop below 1, a whopping 30 000 lives could be saved annually in the US alone [4].

Lastly, consider the impact on the economy. Each ton of carbon emission has around a $220 impact on the US GDP. This means that $220 B could be saved annually by converting all vehicles to ride-sharing clean-energy autonomous vehicles [5]. Also, since the average cost per crash is about $30 000 in the US, by dropping the autonomous vehicle crash rate PMM to below 1, we could achieve another annual cost reduction of $300 B [6]. Therefore, in the US alone, the universal adoption of ride-sharing clean-energy autonomous vehicles could save as much as $520 B annually, which almost ties with the GDP of Sweden, one of the world's largest economies.

Nonetheless, the large-scale adoption of autonomous driving vehicles is now meeting with several barriers, including reliability, ethical and legal considerations, and, not least of

Engineering Autonomous Vehicles and Robots: The DragonFly Modular-based Approach,
First Edition. Shaoshan Liu.

which, affordability. What are the problems behind the building and deploying of autonomous vehicles and how can we solve them? Answering these questions demands that we first look at the underlying design.

1.2 High Cost of Autonomous Driving Technologies

In this section we break down the costs of existing autonomous driving systems, and demonstrate that the high costs of sensors, computing systems, and High-Definition (HD) maps are the major barriers of autonomous driving deployment [7] (Figure 1.1).

1.2.1 Sensing

The typical sensors used in autonomous driving include Global Navigation Satellite System (GNSS), Light Detection and Ranging (LiDAR), cameras, radar and sonar: *GNSS receivers*, especially those with real-time kinematic (RTK) capabilities, help autonomous vehicles localize themselves by updating global positions with at least meter-level accuracy. A high-end GNSS receiver for autonomous driving could cost well over $10 000.

LiDAR is normally used for the creation of HD maps, real-time localization, as well as obstacle avoidance. LiDAR works by bouncing a laser beam off of surfaces and measuring the reflection time to determine distance. LiDAR units suffer from two problems: first, they are extremely expensive (an autonomous driving grade LiDAR could cost over $80 000); secondly, they may not provide accurate measurements under bad weather conditions, such as heavy rain or fog.

Cameras are mostly used for object recognition and tracking tasks, such as lane detection, traffic light detection, and pedestrian detection. Existing implementations usually mount multiple cameras around the vehicle to detect, recognize, and track objects. However, an important drawback of camera sensors is that the data they provide may not be reliable

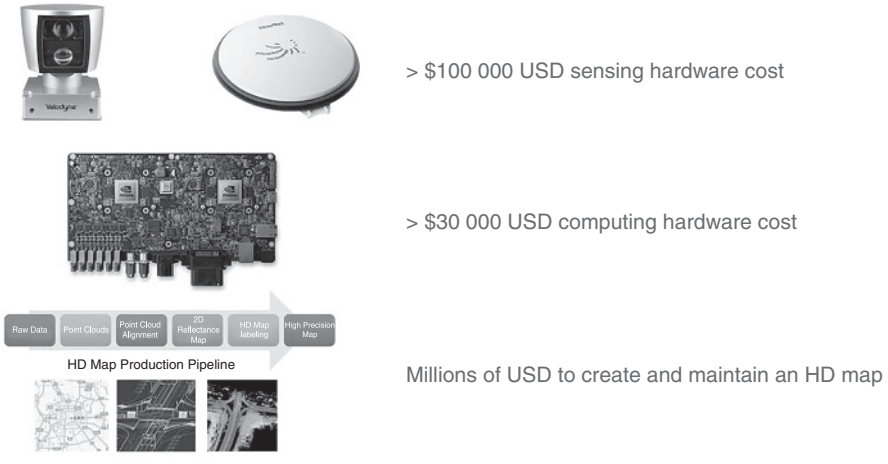

Figure 1.1 Cost breakdown of existing autonomous driving solutions.

under bad weather conditions and that their sheer amount creates high computational demands. Note that these cameras usually run at 60 Hz, and, when combined, can generate over 1 GB of raw data per second.

Radar and sonar: The radar and sonar subsystems are used as the last line of defense in obstacle avoidance. The data generated by radar and sonar show the distance from the nearest object in front of the vehicle's path. Note that a major advantage of radar is that it works under all weather conditions. Sonar usually covers a range of 0–10 m whereas radar covers a range of 3–150 m. Combined, these sensors cost less than $5000.

1.2.2 HD Map Creation and Maintenance

Traditional digital maps are usually generated from satellite imagery and have meter-level accuracy. Although this accuracy is sufficient for human drivers, autonomous vehicles demand maps with higher accuracy for lane-level information. Therefore, HD maps are needed for autonomous driving.

Just as with traditional digital maps, HD maps have many layers of information. At the bottom layer, instead of using satellite imagery, a grid map is generated by raw LiDAR data, with a grid granularity of about 5 cm by 5 cm. This grid basically records elevation and reflection information of the environment in each cell. As the autonomous vehicles are moving and collecting new LiDAR scans, they perform self-localization by performing a real time comparison of the new LiDAR scans against the grid map with initial position estimates provided by GNSS [8].

On top of the grid layer, there are several layers of semantic information. For instance, lane information is added to the grid map to allow autonomous vehicles to determine whether they are on the correct lane when moving. On top of the lane information, traffic sign labels are added to notify the autonomous vehicles of the local speed limit, whether traffic lights are nearby, etc. This gives an additional layer of protection in case the sensors on the autonomous vehicles fail to catch the signs.

Traditional digital maps have a refresh cycle of 6–12 months. However, to make sure the HD maps contain the most up-to-date information, the refresh cycle for HD maps should be shortened to no more than one week. As a result, operating, generating, and maintaining HD maps can cost upwards of millions of dollars per year for a mid-size city.

1.2.3 Computing Systems

The planning and control algorithms and the object recognition and tracking algorithms have very different behavioral characteristics which call for different kinds of processors. HD maps, on the other hand, stress the memory [9]. Therefore, it is imperative to design a computing hardware system which addresses these demands, all within limited computing resources and power budget. For instance, as indicated in [9], an early design of an autonomous driving computing system was equipped with an Intel® Xeon E5 processor and four to eight Nvidia® K80 graphics processing unit (GPU) accelerators, connected with a Peripheral Component Interconnect-E (PCI-E) bus. At its peak, the whole system, while capable of delivering 64.5 Tera Operations Per Second (TOPS), consumed about 3000 W, consequently generating an enormous

amount of heat. Also, at a cost of $30 000, the whole solution would be unaffordable (and unacceptable) to the average consumer.

1.3 Achieving Affordability and Reliability

Many major autonomous driving companies, such as Waymo, Baidu, and Uber, and several others are engaged in a competition to design and deploy the ultimate ubiquitous autonomous vehicle which can operate reliably and affordably, even in the most extreme environments. Yet, we have just seen that the cost for all sensors could be over $100 000, with the cost for the computing system another $30 000, resulting in an extremely high cost for each vehicle: a demo autonomous vehicle can easily cost over $800 000 [10]. Further, beyond the unit cost, it is still unclear how the operational costs for HD map creation and maintenance will be covered.

In addition, even with the most advanced sensors, having autonomous vehicles coexist with human-driven vehicles in complex traffic conditions remains a dicey proposition. As a result, unless we can significantly drop the costs of sensors, computing systems, and HD maps, as well as dramatically improve localization, perception, and decision-making algorithms in the next few years, autonomous driving will not be universally adopted.

Addressing these problems, a reliable autonomous vehicle has been developed by us and for low-speed scenarios, such as university campuses, industrial parks, and areas with limited traffic [11,12]. This approach starts with low speed to ensure safety, thus allowing immediate deployment. Then, with technology improvements and with the benefit of accumulated experience, high-speed scenarios will be envisioned, ultimately having the vehicle's performance equal that of a human driver in any driving scenario. The keys to enable affordability and reliability include using sensor fusion, modular design, and high-precision visual maps (HPVMs).

1.3.1 Sensor Fusion

Using LiDAR for localization or perception is extremely expensive and may not be reliable. To achieve affordability and reliability, multiple affordable sensors (cameras, GNSS receivers, wheel encoders, radars, and sonars) can be used to synergistically fuse their data. Not only do these sensors each have their own characteristics, drawbacks, and advantages but they complement each other such that when one fails or otherwise malfunctions, others can immediately take over to ensure system reliability. With this sensor fusion approach, sensor costs are limited to under $2000.

The localization subsystem relies on GNSS receivers to provide an initial localization with sub-meter-level accuracy. Visual odometry can further improve the localization accuracy down to the decimeter level. In addition, wheel encoders can be used to track the vehicles' movements in case of GNSS receiver and camera failures. Note that visual odometry deduces position changes by examining the overlaps between two frames. However, when a sudden motion is applied to the vehicle, such as a sharp turn, it is possible that visual odometry will fail to maintain localization due to the lack of overlapping regions between two consecutive frames.

The active perception subsystem seeks to assist the vehicle in understanding its environment. Based on this understanding and a combination of computer vision and of millimeter wave (mmWave) radars to detect and track static or moving objects within a 50 m range, the vehicle can make action decisions to ensure a smooth and safe trip. With stereo vision, not only can objects including pedestrians and moving vehicles be easily recognized but the distance to these detected objects can be accurately pinpointed as well. In addition, mmWave radars can also detect and track fast-moving objects and their distances under all weather conditions.

The passive perception subsystem aims to detect any immediate danger and acts as the last line of defense of the vehicle. It covers the near field, i.e. a range of 0–5 m around the vehicle. This is achieved by a combination of mmWave radars and sonars. Radars are very good moving object detectors and sonars are very good static object detectors. Depending on the current vehicle speed, when something is detected within the near field, different policies are put into place to ensure the safety of the vehicle.

1.3.2 Modular Design

In the recent past, designs of autonomous driving computing systems have tended to be costly but affordable computing solutions are possible [9]. This has been made possible by the application of modular design principles which push computing to the sensor end so as to reduce the computing demands on the main computing units. Indeed, a quad-camera module such as the DragonFly sensor module [11] alone can generate image data at a rate of 400 Mbps. If all the sensor data were transferred to the main computing unit, it would require this computing unit to be extremely complex, with many consequences in terms of reliability, power, cost, etc.

Our approach is more practical: it entails breaking the functional units into modules and having each module perform as much computing as possible. This makes for a reduction in the burden on the main computing system and a simplification in its design, with consequently higher reliability. More specifically, a GPU SoM (System on Module) is embedded into the DragonFly module to extract features from the raw images. Then, only the extracted features are sent to the main computing unit, reducing the data transfer rate a 1000-fold. Applying the same design principles to the GNSS receiver subsystem and the radar subsystem reduces the cost of the whole computing system to less than $2000.

1.3.3 Extending Existing Digital Maps

Creating and maintaining HD maps is another important component of deployment costs. Crowd-sourcing the data for creating HD maps has been proposed. However, this would require vehicles with LiDAR units, and we have already seen that LiDARs are extremely expensive and thus not ready for large-scale deployment. On the other hand, crowd-sourcing visual data is a very practical solution as many cars today are already equipped with cameras.

Hence, instead of building HD maps from scratch, our philosophy is to enhance existing digital maps with visual information to achieve decimeter-level accuracy. These are

called HPVMs. To effectively help with vehicle localization, HPVMs consists of multiple layers:

1. The bottom layer can be any of the existing digital maps, such as Open Street Map; this bottom layer has a resolution of about 1 m.
2. The second layer is the ground feature layer. It records the visual features from the road surfaces to improve mapping resolution to the decimeter level. The ground feature layer is particularly useful when in crowded city environments where the surroundings are filled with other vehicles and pedestrians.
3. The third layer is the spatial feature layer, which records the visual features from the environments; this provides more visual features compared with the ground feature layer. It also has a mapping resolution at the decimeter level. The spatial feature layer is particularly useful in less-crowded open environments such as the countryside.
4. The fourth layer is the semantic layer, which contains lane labels, traffic light and traffic sign labels, etc. The semantic layer aids vehicles in making planning decisions such as routing.

1.4 Modular Design

Before we go into the details of the rest of this book, let us briefly go over the modular design methodology and introduce each module. Hopefully with this introduction, readers will be able to easily follow the contents of this book.

Figure 1.2 shows a DragonFly Pod [13], a low-speed autonomous passenger pod built utilizing the modular design methodology described in this book. This vehicle consists of multiple components, a RTK GNSS module for localization, a DragonFly computer vision module for localization (using visual inertial odometry technology) and active perception,

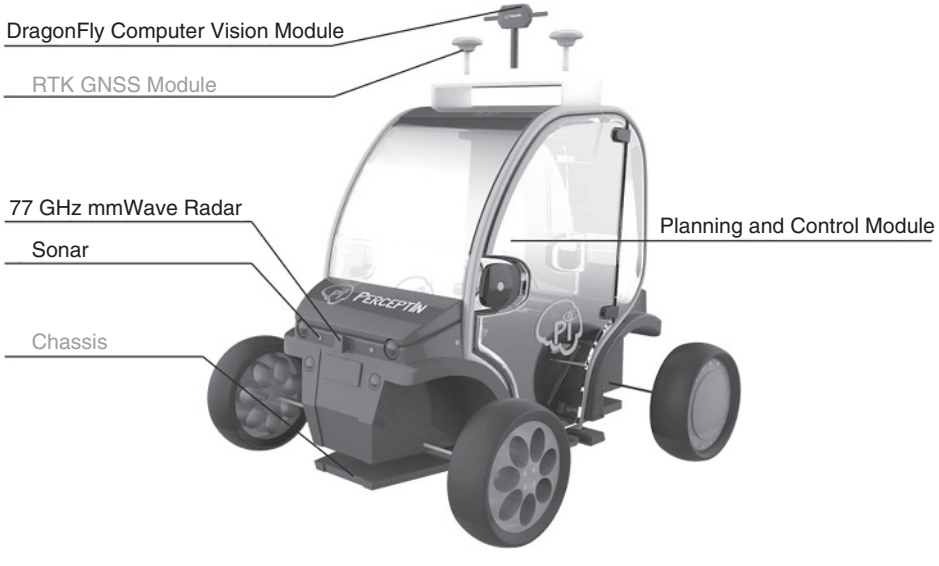

Figure 1.2 Modular design of a DragonFly Pod.

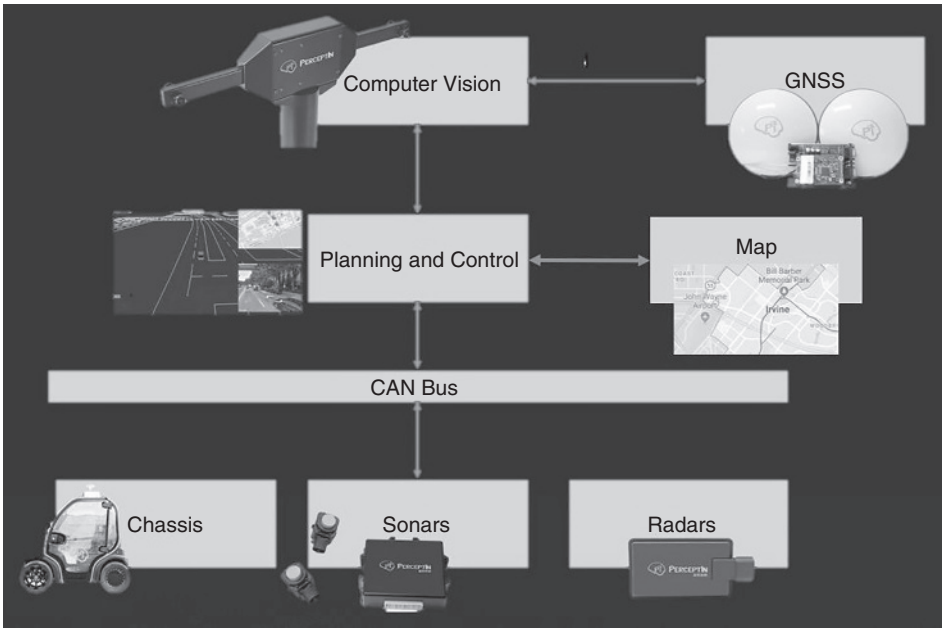

Figure 1.3 Modular design architecture.

a mmWave radar and a sonar for passive perception, a planning and control module for real-time planning, and a chassis module. Figure 1.3 shows the architecture diagram of this design and shows how the modules interact with each other.

1.4.1 Communication System

First, to connect different modules to form a working system, a reliable communication system is needed. The Controller Area Network (CAN) bus is the most widely used in-vehicle communication network today due to its simplicity, and it can be used to connect Electronic Control Units (ECUs), sensors, and other components to enable communication with each other. Before going into the details of other components, readers should first understand how the CAN bus works.

1.4.2 Chassis

The traditional vehicle chassis utilizes mechanical control, such as mechanical cables, hydraulic pressure, and other ways of providing a driver with direct, physical control over the speed or direction of a vehicle.

However, for autonomous driving to work, we need a drive-by-wire-ready chassis such that the chassis can apply electronic controls to activate the brakes, control the steering, and operate other mechanical systems. Specifically, the chassis module provides the basic application program interfaces for the planning and control module, such that the planning and control module can perform steer, throttle, and brake actions to make sure that the vehicle travels on the planned trajectory.

1.4.3 mmWave Radar and Sonar for Passive Perception

For mid-range obstacle detection, we can apply 77 GHz mmWave radar such that the planning and control module can make decisions when obstacles are detected. Similarly, sonars cover near-range obstacles and act as the very last line of defense; once sonars detect an obstacle, they directly signal the chassis to stop to minimize risks of an accident.

mmWave radar and sonar sensors can be combined and used for passive perception. By passive perception, we mean that when obstacles are detected, the raw data are not fed to the planning and control module for decision making. Instead, the raw data are directly sent to the chassis through the CAN bus for quick decision making. In this case, a simple decision module is implemented in the chassis to stop the vehicle when an obstacle is detected within a short range.

The main reason for this design is that when obstacles are detected in close range, we want to stop the vehicle as soon as possible instead of going through the complete decision pipeline. This is the best way to guarantee the safety of passengers as well as pedestrians.

1.4.4 GNSS for Localization

The GNSS system is a natural choice for vehicle localization, especially with RTK capability, GNSS systems can achieve very high localization accuracy. GNSS provides detailed localization information such as latitude, longitude, altitude, as well as vehicle heading. Nonetheless, GNSS accuracy suffers when there are buildings and trees blocking an open sky, leading to multipath problems. Hence, we cannot solely rely on GNSS for localization.

1.4.5 Computer Vision for Active Perception and Localization

Computer vision can be utilized for both localization and active perception. For localization, we can rely on visual simultaneous localization and mapping (VSLAM) technologies to achieve accurate real-time vehicle locations. However, VSLAM usually suffers from cumulative errors such that the longer the distance the vehicle travels, the higher the localization error. Fortunately, by fusing VSLAM and GNSS localizations, we can achieve high accuracy under different conditions, because GNSS can be used as the group-truth data when it is not blocked, and VSLAM can provide high accuracy when GNSS is blocked.

In addition, computer vision can be used for active perception as well. Using stereo vision, we can extract spatial or depth information of different objects; using deep learning techniques, we can extract semantic information of different objects. By fusing spatial and semantic information, we can detect objects of interest, such as pedestrians and cars, as well as getting their distance to the current vehicle.

1.4.6 Planning and Control

The planning and control module receives inputs from perception and localization modules, and generates decisions in real time. Usually, different behaviors are defined for a planning and control module and under different conditions, one behavior is chosen.

A typical planning and control system has the following architecture: first, as the user enters the destination, the *routing module* checks the map for road network information and generates a route. Then the route is fed to the *behavioral planning module*, which checks the traffic rules to generate motion specifications. Next, the generated route along with motion specifications are passed down to the *motion planner*, which combines real-time perception and localization information to generate trajectories. Finally, the generated trajectories are passed down to the *control system*, which reactively corrects errors in the execution of the planned motions.

1.4.7 Mapping

A mapping module provides essential geographical information, such as lane configurations and static obstacle information, to the planning and control module. In order to generate real-time motion plans, the planning and control module can combine perception inputs, which detect dynamic obstacles in real time, localization inputs, which generate real-time vehicle poses, and mapping inputs, which capture road geometry and static obstacles.

Currently, fully autonomous vehicles use high definition 3D maps. Such high precision maps are extremely complex and contain a trillion bytes of data to represent not only lanes and roads but also semantic and locations of 3D landmarks in the real world. With HD maps, autonomous vehicles are able to localize themselves and navigate in the mapped area.

1.5 The Rest of the Book

In the previous sections we have introduced the proposed modular design approach for building autonomous vehicles and robots. In the rest of the book, we will delve into these topics, and present the details of each module as well as how to integrate these modules to enable a fully functioning autonomous vehicle or robot.

The first part of the book consists of Chapters 2–8, in which we introduce each module, including communication systems, chassis technologies, passive perception technologies, localization with RTK GNSS, computer vision for perception and localization, planning and control, as well as mapping technologies.

- Chapter 2: In-Vehicle Communication Systems
- Chapter 3: Chassis Technologies for Autonomous Robots and Vehicles
- Chapter 4: Passive Perception with Sonar and mmWave Radar
- Chapter 5: Localization with RTK GNSS
- Chapter 6: Computer Vision for Perception and Localization
- Chapter 7: Planning and Control
- Chapter 8: Mapping

The second part of the book consists of Chapters 9 and 10, in which we present two interesting case studies: the first one is about applying the modular design to build low-speed

autonomous vehicles; and the second one is about how NASA builds its space robotic explorer using a modular design approach.

- Chapter 9: Building the DragonFly Pod and Bus
- Chapter 10: Enabling Commercial Autonomous Space Robotic Explorers

From our practical experiences, the capabilities of autonomous vehicles and robots are often constrained by limited onboard computing power. Therefore, in the final part of the book, we delve into state-of-the-art approaches in building edge computing systems for autonomous vehicles and robots. We will cover onboard edge computing design, vehicle-to-everything infrastructure, as well as autonomous vehicle security.

- Chapter 11: Edge Computing for Autonomous Vehicles
- Chapter 12: Innovations on the Vehicle-to-Everything Infrastructure
- Chapter 13: Vehicular Edge Security

1.6 Open Source Projects Used in this Book

As you can see, an autonomous driving system is a highly complex system that integrates many technology pieces and modules. Hence, it is infeasible and inefficient to build everything from scratch. Hence, we have referred to many open source projects throughout the book to help readers to build their own autonomous driving systems. Also, throughout the book we have used PerceptIn's autonomous driving software stack to demonstrate the idea of modular design. The open source projects used in this book are listed below:

- *CANopenNode* [14]: This is free and open source CANopen Stack is for CAN bus communication.
- *Open Source Car Control* [15]: This is an assemblage of software and hardware designs that enable computer control of modern cars in order to facilitate the development of autonomous vehicle technology. It is a modular and stable way of using software to interface with a vehicle's communications network and control systems.
- *OpenCaret* [16]: This is an open source Level-3 Highway autopilot system for Kia Soul EV.
- *NtripCaster* [17]: A GNSS NTRIP (Networked Transport of RTCM via Internet Protocol) Caster takes GNSS data from one or more data stream sources (Base Stations referred to as NTRIP Servers) and provides these data to one or more end users (often called rovers), the NTRIP Clients. If you need to send data to more than one client at a time, or have more than one data stream, you will need a Caster.
- GPSD *(GPS Daemon)* [18]: This is a service daemon that monitors one or more GNSS receivers attached to a host computer through serial or USB ports, making all data on the location/course/velocity of the sensors available to be queried on Transmission Control Protocol port 2947 of the host computer. With GPSD, multiple location-aware client applications can share access to supported sensors without contention or loss of data. Also, GPSD responds to queries with a format that is substantially easier to parse than the NMEA 0183 emitted by most GNSS receivers.

- *Kalibr* [19]: This is a toolbox that solves the following calibration problems:
  - Multiple camera calibration: intrinsic and extrinsic calibration of a camera system with non-globally shared overlapping fields of view.
  - Visual-inertial calibration (camera-IMU): spatial and temporal calibration of an IMU with respect to a camera system.
  - Rolling shutter camera calibration: full intrinsic calibration (projection, distortion, and shutter parameters) of rolling shutter cameras.
- *OpenCV* [20]: OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.
- *ORB-SLAM2* [21]: This is a real-time SLAM library for Monocular, Stereo and RGB-D cameras that computes the camera trajectory and a sparse 3D reconstruction. It is able to detect loops and relocalize the camera in real time.
- *libELAS* [22]: This is a cross-platform C++ library with MATLAB wrappers for computing disparity maps of large images. Input is a rectified grayscale stereo image pair of the same size. Output is the corresponding disparity maps.
- *Mask R-CNN* [23]: This is a deep learning model for object detection and instance segmentation on Keras and TensorFlow.
- *Baidu Apollo* [24]: Apollo is a high performance, flexible architecture which accelerates the development, testing, and deployment of autonomous vehicles.
- *OpenStreetMap* [25]: This is a collaborative project to create a free editable map of the world. The geodata underlying the map are considered the primary output of the project. The creation and growth of OpenStreetMap has been motivated by restrictions on use or availability of map data across much of the world, and the advent of inexpensive portable satellite navigation devices.

References

1 U.S. Department of Energy (2017). Emissions from Hybrid and Plug-In Electric Vehicles. https://www.afdc.energy.gov/vehicles/electric_emissions.php (accessed 1 December 2017).

2 MIT CSAIL (2016). Study: carpooling apps could reduce taxi traffic 75%. https://www.csail.mit.edu/news/study-carpooling-apps-could-reduce-taxi-traffic-75 (accessed 1 December 2017).

3 VirginiaTech (2017). Automated vehicle crash rate comparison using naturalistic data. https://www.vtti.vt.edu/featured/?p=422 (accessed 1 December 2017).

4 U.S. Department of Transportation (2016). U.S. Driving Tops 3.1 Trillion Miles in 2015. https://www.fhwa.dot.gov/pressroom/fhwa1607.cfm (accessed 1 December 2017).

5 Moore, F.C. and Diaz, D.B. (2015). Temperature impacts on economic growth warrant stringent mitigation policy. *Nature Climate Change* 5 (2): 127–131.

6 New York State Department of Transportation (2016). Average Accident Costs. https://www.dot.ny.gov/divisions/operating/osss/highway-repository/39D1F023EC4400C6E0530A3DFC0700C6 (accessed 1 December 2017).

7 Liu, S., Li, L., Tang, J. et al. (2017). *Creating Autonomous Vehicle Systems*, Synthesis Lectures on Computer Science, vol. 6, 1–186. Morgan & Claypool Publishers.

8 Liu, S., Tang, J., Wang, C. et al. (2017). A unified cloud platform for autonomous driving. *Computer* (12): 42–49.

9 Liu, S., Tang, J., Zhang, Z., and Gaudiot, J.-L. (2017). Computer architectures for autonomous driving. *Computer* 50 (8): 18–25.

10 AutonomousStuff (2017). Lincoln MKZ Platform. https://autonomoustuff.com/product/lincoln-mkz (accessed 1 October 2018).

11 YouTube (2018). PerceptIn DragonFly Sensor Module https://www.youtube.com/watch?v=WQUGB-IqbgQ&feature=youtu.be (accessed 1 October 2018).

12 Vega, P. (2018). UC Irvine grad works to make a self-driving car costing under $10,000. *Los Angeles Times*. http://www.latimes.com/socal/daily-pilot/news/tn-dpt-me-driverless-cars-20180105-story.html (accessed 8 January 2018).

13 PerceptIn (2017). PerceptIn DragonFly Pod. https://www.perceptin.io/products (accessed 1 October 2019).

14 GitHub (2019). CANopenNode. https://github.com/CANopenNode/CANopenNode (accessed 1 October 2019).

15 GitHub (2019). Open Source Car Control. https://github.com/PolySync/oscc (accessed 1 October 2019).

16 GitHub (2019). OpenCaret. October 2019, https://github.com/frk2/opencaret (accessed 1 October 2019).

17 GitHub (2019). NtripCaster. https://github.com/nunojpg/ntripcaster (accessed 1 October 2019).

18 gpsd (2019). gpsd – a GPS sevice daemon. https://gpsd.gitlab.io/gpsd/index.html (accessed 1 October 2019).

19 GitHub (2019). Kalibr. https://github.com/ethz-asl/kalibr (accessed 1 October 2019).

20 OpenCV (2019). OpenCV. https://opencv.org (accessed 1 October 2019).

21 GitHub (2019). ORB-SLAM2. https://github.com/raulmur/ORB_SLAM2 (accessed 1 October 2019).

22 Geiger, A. (2019). libELAS. http://www.cvlibs.net/software/libelas (accessed 1 October 2019).

23 GitHub (2019). Mask R-CNN. https://github.com/matterport/Mask_RCNN (accessed 1 October 2019).

24 GitHub (2019). Baidu Apollo. https://github.com/ApolloAuto/apollo (accessed 1 October 2019).

25 GitHub (2019). OpenStreetMap. https://github.com/openstreetmap (accessed 1 October 2019).