Availability Analysis of Optical Sensors using Autonomous Driving Data

Verfügbarkeitsanalyse von optischen Sensoren auf Basis von Datensätzen des Autonomen Fahrens

Scientific work for obtaining the academic degree

Bachelor of Science (B.Sc.)

At the Department of Mechanical Engineering of Technical University Munich

Supervised by Prof. Dr.-Ing. Markus Lienkamp
Julian Kreibich, M.Sc.
Chair of Automotive Technology

Submitted by Joe Schmit
Korbinianstraße 18
80807 München

Submitted on September 13, 2021

Lehrstuhl für Fahrzeugtechnik
Fakultät für Maschinenwesen
Technische Universität München

Aufgabenstellung

Hochgenaue Kartendienste stellen für viele innovative Anwendungen im Fahrzeug eine unverzichtbare Grundlage dar. Dort werden neben statischen Straßeninformationen, wie Verkehrsschildern und Ampeln, auch dynamisch auftretende Ereignisse, z. B. Staus und Baustellen, den Kartendaten zugeordnet. Zusätzlich kann die Einspeisung aktueller Daten zur Beschaffenheit der Straßenoberfläche die Verkehrssicherheit und den Komfort weiter verbessern.

In Zukunft werden Fahrdynamiksysteme neben der fahrzeuginternen Sensorik zusätzlich auf Umfeldinformationen aus Kamera, Lidar, Radar und den genannten Kartendiensten zurückgreifen können. Während fahrzeuginterne Sensorik den aktuellen Zustand des Fahrzeugs, und der Straße mit hoher Zuverlässigkeit beschreibt, können Umfelddaten den zukünftigen Streckenzustand liefern. Dabei variiert die Verfügbarkeit und Güte der Preview-Daten je nach äußeren Randbedingungen wie Straßengeometrie, Sicht bzw. Wetter oder Verkehrsaufkommen.

Im Rahmen eines Forschungsprojekts soll demzufolge die Verfügbarkeit und Güte von optischen Sensoren auf Basis von Datensätzen des Autonome Fahrens untersucht werden. Folgende Arbeitspakete umfasst die zu vergebende Studienarbeit:

- Literaturrecherche und fundierte Darstellung des Stands der Wissenschaft Verfügbarkeit von optischen Sensoren
- Stand der Technik: Was haben FAS-Systeme für Genauigkeiten, Reichweiten und Verfügbarkeiten?
- Ableitung relevanter Umweltbedingungen, wie bspw. Kurven, vorausfahrende Fahrzeuge, Kuppen oder Senken und Witterung für die Verfügbarkeit der optischen Sensorik
- Analyse der Sensorverfügbarkeit innerhalb der Datensätze
- Kritische Diskussion und ausführliche Dokumentation der Ergebnisse

Die Ausarbeitung soll die einzelnen Arbeitsschritte in übersichtlicher Form dokumentieren. Der Kandidat/Die Kandidatin verpflichtet sich, die Studienarbeit selbständig durchzuführen und die von ihm verwendeten wissenschaftlichen Hilfsmittel anzugeben.

Die eingereichte Arbeit verbleibt als Prüfungsunterlage im Eigentum des Lehrstuhls.

Ausgabe: 15.03.2021 Abgabe: 13.09.2021

_____ _____

Prof. Dr.-Ing. M. Lienkamp Betreuer: Julian Kreibich, M. Sc.

Lehrstuhl für Fahrzeugtechnik
Fakultät für Maschinenwesen
Technische Universität München

Geheimhaltungsverpflichtung

Herr: **Schmit, Joe**

Gegenstand der Geheimhaltungsverpflichtung sind alle mündlichen, schriftlichen und digitalen Informationen und Materialien, die der Unterzeichner vom Lehrstuhl oder von Dritten im Rahmen seiner Tätigkeit am Lehrstuhl erhält. Dazu zählen vor allem Daten, Simulationswerkzeuge und Programmcode sowie Informationen zu Projekten, Prototypen und Produkten.

Der Unterzeichner verpflichtet sich, alle derartigen Informationen und Unterlagen, die ihm während seiner Tätigkeit am Lehrstuhl für Fahrzeugtechnik zugänglich werden, strikt vertraulich zu behandeln.

Er verpflichtet sich insbesondere:

- derartige Informationen betriebsintern zum Zwecke der Diskussion nur dann zu verwenden, wenn ein ihm erteilter Auftrag dies erfordert,
- keine derartigen Informationen ohne die vorherige schriftliche Zustimmung des Betreuers an Dritte weiterzuleiten,
- ohne Zustimmung eines Mitarbeiters keine Fotografien, Zeichnungen oder sonstige Darstellungen von Prototypen oder technischen Unterlagen hierzu anzufertigen,
- auf Anforderung des Lehrstuhls für Fahrzeugtechnik oder unaufgefordert spätestens bei seinem Ausscheiden aus dem Lehrstuhl für Fahrzeugtechnik alle Dokumente und Datenträger, die derartige Informationen enthalten, an den Lehrstuhl für Fahrzeugtechnik zurückzugeben.

Besondere Sorgfalt gilt im Umgang mit digitalen Daten:

- Für den Dateiaustausch dürfen keine Dienste verwendet werden, bei denen die Daten über einen Server im Ausland geleitet oder gespeichert werden (Es dürfen nur Dienste des LRZ genutzt werden (Lehrstuhllaufwerke, Sync&Share, GigaMove).
- Vertrauliche Informationen dürfen nur in verschlüsselter Form per E-Mail versendet werden.
- Nachrichten des geschäftlichen E-Mail Kontos, die vertrauliche Informationen enthalten, dürfen nicht an einen externen E-Mail Anbieter weitergeleitet werden.
- Die Kommunikation sollte nach Möglichkeit über die (my)TUM-Mailadresse erfolgen.

Die Verpflichtung zur Geheimhaltung endet nicht mit dem Ausscheiden aus dem Lehrstuhl für Fahrzeugtechnik, sondern bleibt 5 Jahre nach dem Zeitpunkt des Ausscheidens in vollem Umfang bestehen. Die eingereichte schriftliche Ausarbeitung darf der Unterzeichner nach Bekanntgabe der Note frei veröffentlichen.

Der Unterzeichner willigt ein, dass die Inhalte seiner Studienarbeit in darauf aufbauenden Studienarbeiten und Dissertationen mit der nötigen Kennzeichnung verwendet werden dürfen.

Datum: 15.03.2021

Unterschrift: 

Erklärung

Ich versichere hiermit, dass ich die von mir eingereichte Abschlussarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Garching, den 13.09.2021

Joe Schmit

Declaration of Consent, Open Source

Hereby I, **Joe, Schmit**, born on 19.03.1998, make the software I developed during my Bachelor's Thesis available to the Institute of Automotive Technology under the terms of the license below.

Garching, 15.03.2021

Joe Schmit, B. Sc.

Contents

Glossary

ACC	Adaptive Cruise Control
ADAS	Advanced Driving Assistance Systems
AV	Autonomous Vehicle
A2D2	Audi Autonomous Driving Dataset
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CV	Computer Vision
FOV	Field of View
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
KITTI	Karlsruhe Institute of Technology and Toyota Technical Institute
LIDAR	Light Detection and Ranging
NIR	Near Infrared
OpenCV	Open-source Computer Vision
OSM	OpenStreetMap
RGB	Red Green Blue color channels
ROI	Region of Interest
RTK	Real-time Kinematic Positioning
SAE	Society of Automotive Engineers
SLAM	Simultaneous Localization and Mapping
SWIR	Short Wave Infrared
TAI	International Atomic Time
TOF	Time of Flight
UNIX	Operating System developed in the 1970s
UTC	Coordinated Universal Time
V2X	Vehicle to Everything Communication
5G	Fifth Generation of Cellular Networks

Formula Symbols

Formula Symbol	Unit	Description
Δt	s	Time of Flight
B	m	Baseline
c_{x}	Pixel	Horizontal image center
c_y	Pixel	Vertical image center
d	m	Distance
d'	Pixel	Disparity
D	m	Depth
f	Pixel	Focal length
K_x	-	Horizontal Sobel operator
K_y	-	Vertical Sobel operator
n	-	Refractive index
v	m s^{-1}	Vehicle velocity

1 Introduction

1.1 Motivation

Mobility and the means by which it is accomplished have undergone multiple groundbreaking revolutions over time. From the invention of the automobile onwards, the pace of development and innovation in this space has continuously increased. From the introduction of the first safety features such as seatbelts, to the development of Advanced Driving Assistance Systems (ADAS) such as (adaptive) cruise control (ACC), to the advancements and widespread adoption of electric vehicles, the automotive industry has incessantly continued to make transportation and mobility safer, cleaner, and more efficient.

Currently, the next milestone transformation is underway in the automotive sector: fully autonomous vehicles (AV). The Society of Automotive Engineers (SAE) defines six different levels of autonomy ranging from 0 (no autonomous capacity) to 5 (fully autonomous) [1]. Currently AV are on the cusp of reaching level 4 or 5 autonomy that do not require any driver input. Fully driverless cars hold the potential to make transportation more secure by reducing accidents and traffic and by factoring out human errors which cause a majority of car crashes [2, p. 114].

To reach these levels of autonomy and to operate in a safe and satisfactory fashion, AV however need channels to perceive their environment and gain a comprehensive scene understanding under any conditions. The logical means to accomplish this task is to mimic human vision and its capabilities to quickly analyze and assess a scene, as well as gaining an understanding of position and distance of certain objects.

To this end, a host of sensors is employed. Specifically optical sensors are at the center of a comprehensive environmental perception. They include cameras, Light Detection and Ranging (LIDAR), and Radio Detection and Ranging (RADAR) sensors. The performance, reliability, and availability of them are however not always guaranteed. In fact, different factors affect them and the free space ahead of the vehicle. This work thus focuses on investigating the freely accessible space and the influences that affect it as well as the optical sensors used to detect it in the first place. Hence, gaining information about possible constrains of optical sensors and remedies for solving this uncertainty is paramount to safety.

1.2 Objective

The aim of the subsequent analysis is to investigate and assess circumstances that cause the performance of optical sensors to diminish. Various datasets representing real-world scenarios are thus analyzed for two purposes. First of all, the purpose is to obtain a measure of the detection range of their optical sensors as well as the free space they are able to perceive in

optimal conditions. Next, these results are explored more closely to filter out relevant constraints that hamper the performance, range, and degree of information that the engaged sensors can achieve.

One of the main objectives of this work is to establish a benchmark for the availability of optical sensors in real-world settings with varying environmental conditions such as adverse weather, dense traffic, and diverse roads. This can serve as a relevant basis for comparison when analyzing other datasets and their sensor setups.

In a supplementary step, this kind of information could then, in conjunction with precise Global Positioning Systems (GPS) and an internet or 5G cellular connection installed in the road vehicles, be used to update and enrich online map services. Other road users and vehicles could then access and retrieve this data to be alerted of anomalies in free space on their designated path due to construction sites, bad weather, or increased traffic volume for example. This information exchange between vehicles, the cloud, and other road users is called Vehicle to Everything or V2X communication and holds the potential to augment the autonomous capabilities of future AV, increase road safety, and enhance the comfort of passengers.

1.3 Structure

To set a foundation for the framework and achieve the desired results, this analysis is structured as displayed in Figure 1.1. In a first step the underpinnings of cameras, image processing, and common computer vision (CV) algorithms are laid out. An overview of other optical sensors and available datasets focused on autonomous driving is provided and an evaluation of them is carried out.

After a selection of datasets to investigate in this particular work, methods devised for free road detection and distance calculation using optical sensor data are presented. These approaches then allow for an evaluation of the availability of sensors in different conditions and different datasets.

Subsequently, the obtained results are presented and the influences that cause distortions in them are explored. A critical discussion and assessment of the used techniques, datasets, and results follows and includes possible shortcomings of either one.

The work concludes with an outlook on future applications and evaluations that could be achieved using this or a similar framework.

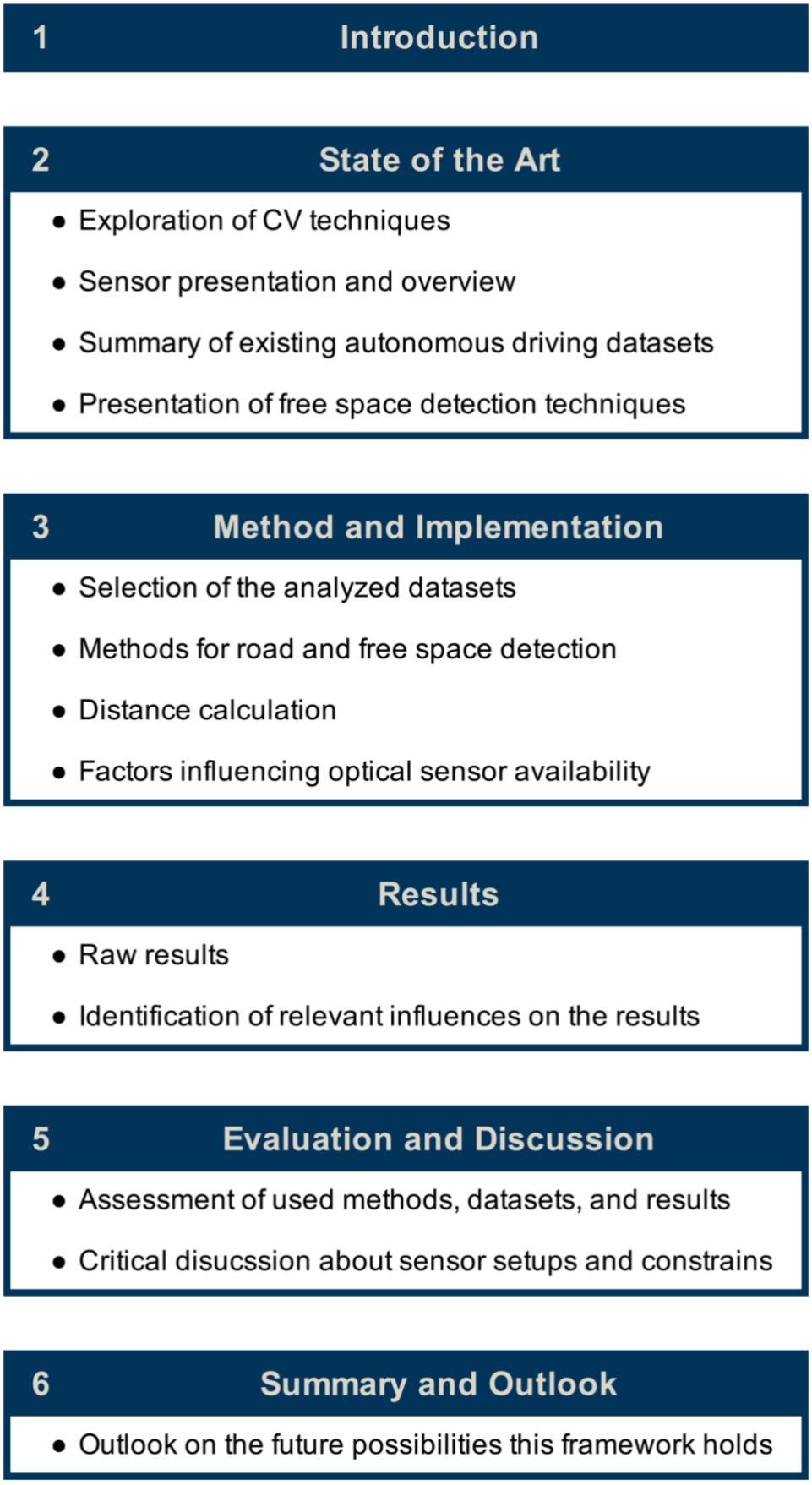

Figure 1.1: Structural overview of the covered topics in this work.

2 State of the Art

This chapter seeks to establish a foundation for the subsequent exploration of different datasets. To that end, it presents an overview of the mathematical foundations of imagery and Computer Vision techniques used to process images. In a next step the different sensors employed in the automotive context, their modes of operation, and their features are investigated. Based on this, existing datasets providing combinations of different sensor and image data are portrayed and assessed. The assessment takes into account which sensor setup was used, how vast the dataset is, and how the provided data can be useful for the task of free space and distance estimation. To conclude, different methods of sensor fusion used in the automotive context for tasks such as road detection and free space calculation are presented. Furthermore, existing methods used for distance estimation, up on which this work tries to build are explored.

2.1 Foundations of Imagery and Computer Vision

Thanks to their highly functional nervous system and the complex anatomy of their eyes, humans are exceptionally capable of quickly analyzing images [3, p. 19]. This ability is due to many body parts and concepts, some of which are explored in the following. Stereo vision, a concept explained in the upcoming chapter, which relies on humans having two eyes horizontally separated by a few centimeters, allows the brain to estimate the distance of objects as well as their relative position in a scene, i.e in front of or behind other objects [4, p. 301]. The retina with its cones and rods is able to transform incoming light sensations into neural signals which are then processed in the brain in order to create a picture from which shapes and colors are inferred. Human vision is thus one of the main gateways through which information about our surroundings reaches us [5, p. 77].

Thanks to its potential to extract and infer knowledge about the environment, the industry tries to implement this same concept of vision in cars to support their ADAS, make them safer in return, and future-proof them for an autonomous prospect [6, p. 18]. Therefore, most modern cars sold today, especially those with some degree of automated driving such as lane detection, come equipped with an onboard camera or even multi-camera systems to scan their surroundings. The captured images or videos are then processed in real time using CV techniques, to extract useful information. CV, or Machine Vision, is a vast field of computer science, which is used and developped to analyze images, interpret, and return useful information from them. However, since CV is an inverse problem (Figure 2.1), meaning it tries to infer the properties of a 3D object, here the causal factor, by evaluating a 2D image, here the observation, it is inherently challenging [7, p. 3146].

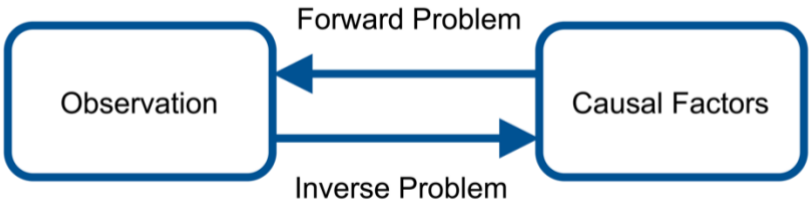

Figure 2.1: Visual description of an inverse problem

In the setting of autonomous vehicles this process of image recognition becomes ever more important. Real time CV based on camera images or videos is a necessary tool to inform a car about its surroundings. The information extraction using different techniques, highlighted in the following, helps AV as cyberphysical systems to become aware of their environment, including lane markings, adjacent cars, or other road obstacles. This awareness in turn allows the vehicle to take correct decisions and perform safe maneuvers.

Hence, the upcoming subchapters provide an overview of image processing and CV techniques employed in the context of AV and this thesis, in order to support the free space and distance estimation.

2.1.1 Mathematical Theory of Camera Images

Whereas humans can identify, without any further analysis or computation, objects, shapes, and colors of images simply by looking at the entirety of them through their eyes as highlighted above, this process looks very different when it comes to machines. Human vision does not require pixel-level examination of a picture in order to extract the contained information. Thanks to the introduction and development of CV since the late 1960s, modern machines are now able to interpret images up to a certain degree [8, p. 10]. In order to achieve this task aimed at the goal of full scene understanding images are stored in the memory of the machine as matrices of pixels. Generally, the ratio between the number of pixels and the size of an image determines its resolution and consequentially its level of detail [9, pp. 57–58] as seen in Figure 2.2.

Figure 2.2: Relationship between image size, pixel count and resolution. 1: Original image (1920x1208); 2: Resized image (910x572); 3: Resized image (455x286); 4: Resized image (228x143). Pictures from the A2D2 dataset by Audi [10]

Matrix Representation

Depending on the color space of the digital image, each pixel stores one or three values. As illustrated in Figure 2.3, in the case of a grayscale image, each pixel stores one intensity value, where higher values indicate a brighter spot. This yields a one-dimensional matrix representing the grayscale image. On the other hand, colorized images are most commonly stored in the RGB (Red, Green, Blue) format. In this instance, three different layers, embodying the three color channels, are overlaid to produce a colorized image. Hence, instead of holding only one intensity value, every pixel stores a vector of three intensity values, yielding a three-dimensional matrix. These intensity values usually have 8-bit or 1-Byte format, and correspondingly range from 0 to 255 in decimal notation, where 0 is the lowest and 255 the highest intensity. Consequently, in an RGB image there exist $256^3 = 16{,}777{,}216$ distinct color combinations. Because each pixel stores three of these values, this color representation is also known as 24-bit “true color” [11, p. 70].

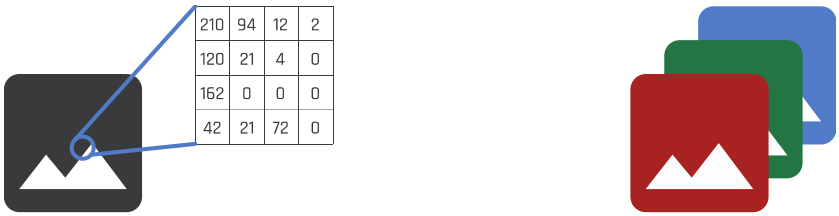

Figure 2.3: Comparison between grayscale and RGB Images [12, p. 8]

Pinhole Camera Model

In a next step, it is important to investigate the setup of cameras and how they produce digital images. A large number of modern camera systems are modelled with the so-called pinhole model [13, p. 1]. As stated previously, images imprint real 3D objects onto a 2D image plane. The task of CV is to extract information from this impression and gain an understanding of the surrounding scene. How exactly the camera achieves this conversion from 3D space to a 2D image plane, is determined by the camera matrix. This matrix holds a number of intrinsic parameters which are specific to each camera model. An exemplified representation of the pinhole model is shown in Figure 2.4. This model assumes that light rays originating from the 3D object traverse the ideally infinitesimal pinhole, without passing through a lens. These rays will go on to project an inversed virtual image of the object onto the image plane [14, p. 4].

F_c denotes the optical center of the pinhole camera from which the global coordinate system (X_c, Y_c, Z_c) originates, where Z_c is aligned with the optical axis perpendicular to the image plane. The image plane is located at a distance from F_c. This distance is known as the focal length (f_x, f_y) with pixels as unit, f_x the horizontal distance and f_y the vertical displacement. (X, Y, Z) are the coordinates which locate the 3D object, or in this case the large blue arrow, in global coordinates. Through specific matrix-vector operations described in Equations 2.1 and 2.2, the pinhole model now maps the position of this object in the real world to the image plane in pixel coordinates (u, v) by a perspective transformation.

$$sm' = s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = A[R|t]M' \tag{2.1}$$

where m' and M' are the coordinates of the point in pixel and real-world coordinates, s is a scaling factor and A the camera matrix holding the intrinsic parameters of focal length (f_x, f_y) and the pixel coordinates of the principal point (c_x, c_y) usually located in the image center.

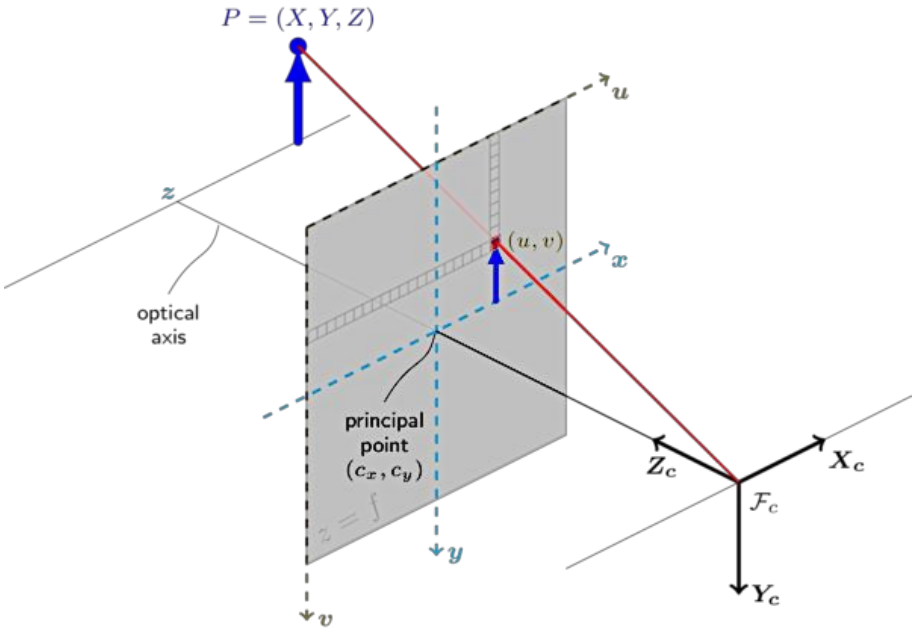

Figure 2.4: Pinhole camera model [15]

R represents the rotation matrix, which when appended with the translation vector t forms the joint rotation-translation matrix $[R|t]$ of extrinsic parameters. Extrinsic parameters contain information about the camera's position in global coordinates and as such $[R|t]$ describes the camera motion around a static scene and is used to translate the global coordinates (X, Y, Z) to the camera coordinates (x, y, z) (Eq. 2.2).

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t \tag{2.2}$$

It is important to note, however, that these equations describe a perfect pinhole camera model. Physically it is impossible to reduce the pinhole to the infinitely small size described above. This limitation in the aperture size causes each point in the image plane to be reached by rays from multiple points in the 3D space, thus interfering with the presented geometric properties. This interference in turn causes the image to blur significantly. Furthermore, although reducing the aperture size leads to crisper images, reducing its size too much causes darker images, because in total less light reaches the image plane. Thus, the trade-off between crispness and brightness arises in the pinhole camera model [16, pp. 2–3]. What possibilities exist to mitigate these unwelcome effects?

In order to deal with this issue and to deliver reliable images, modern cameras are mostly equipped with spherical lenses. These lenses are used to increase the number of light rays the image plane can receive without reducing its sharpness. Lenses refract all the inbound light rays emerging from one point P in 3D space in order to focus them to a single point P' on the camera film. This refraction process is governed by the refractive index n of the lens and is illustrated in

Figure 2.5. According to Snell's Law of refraction, the higher n, the more the incoming light rays are refracted. If the lens is well proportioned and positioned, this process is responsible for creating sharp and bright images at the same time, thus alleviating the trade-off caused by the pinhole model.

The described lens model uses the thin lens approximation which assumes that the angle θ of incoming rays on the lens is small enough to use that $\sin\theta = \theta$, because the curvature of the thin lens is small enough [16, p. 5]. In practice however, this assumption does not always hold, and to be able to use reliable camera images that best represent the 3D scene, one needs to account for the distortion thick lenses cause. Otherwise, when lenses suffer from small or large distortions, the ideal pinhole model might result in large errors in the digital image [17, p. 165]. These distortion errors can be remedied by calibrating the camera system accurately. This in turn guarantees a well-functioning camera system which delivers exact representations of the real 3D world in the form of images.

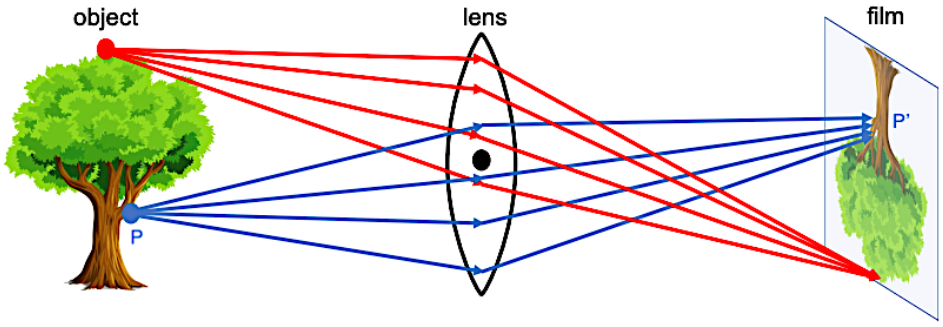

Figure 2.5: Lens refracting light rays from an object point to a single point on the camera film [16, p. 4]

Distortion and Calibration

In the following we always assume that the lenses are rotationally symmetric about their optical axis. Since the thin lens assumption does not hold in most realistic settings, the case of thick lenses has to be considered. Here, the angle θ at which light rays hit the lens can be elevated and the thin lens assumption of $\sin\theta = \theta$ is not valid anymore [18, pp. 159–170]. Additionally, thick lenses usually have small variations in their refractive index over their surface. This leads to a number of optical aberrations and errors in the formulas presented above. These include radial distortion, decentering distortion, and thin prism effect. Of these three types of distortion, the radial distortion constitutes by far the dominant factor [17, p. 165]. The thin prism effect leads to another type of image distortion: tangential distortion [19, p. 721]. Furthermore, since decentering and thin prism effect are respectively due to the misplacement and tilt of the lens, they can be resolved either by improving the camera hardware or by mathematically accounting for them with a translation vector and a rotation matrix [20, p. 607].

Radial distortion leads to the magnification or demagnification of the image as a function of the distance to the optical axis. As such the magnification at the edges is called pincushion distortion and the demagnification at the edges is called barrel distortion, both of which are illustrated in Figure 2.6. The pincushion effect usually occurs in older telephoto lenses, whereas the barrel distortion is predominant in fisheye and wide-angle lenses, which are often used in the automotive context in order to capture a wide field of view (FOV). One concrete ADAS application

that heavily relies on wide-angle camera lenses is 360° around view used for automated or assisted parking, where, thanks to the camera system, a bird's-eye view of the car is created [21].

Camera calibration is now used to counter the distortion effects and provide reliable images. Depending on whether the intrinsic and extrinsic camera parameters are known or unknown, it can be understood as a process to estimate or adjust these parameters. Hence, according to Forsyth and Ponce [14, pp. 22–29] camera calibration represents "an optimization process, where the discrepancy between the observed image features and their theoretical positions is minimized with respect to the camera's intrinsic and extrinsic parameters". This practice should be employed at all times, where vision systems or image processing come into play in order to guarantee transferability and accuracy of the method. The objective of calibration is thus, given a set of images, to approximate the components of the camera matrix A and the joint rotation-translation matrix $[R|t]$ introduced in Equation (2.1).

Figure 2.6: From left to right: Original undistorted square. Barrel distortion. Pincushion distortion. [22, p. 37]

The first model for treating distortion and thus calibrating cameras was introduced by A. Conrady in 1919, and later improved upon by D.C. Brown to build the widely used Brown-Conrady model for decentered distortion of lenses in 1966 [23, p. 58]. In the year 2000, Zhang presented in a seminal paper for 3D CV a flexible new technique for camera calibration, which has only few requirements [24, pp. 1330–1334]. It involves the following steps:

1. Taking at least two pictures of a planar pattern, such as the chess board seen in Figure 2.6, from different positions
2. Identify distinct feature points present in all the pictures
3. Estimate the intrinsic and extrinsic parameters
4. Refine the approximated parameters

In software practice, different CV libraries provide built-in functions for the calibration task. The open-source project OpenCV offers the widely used `cv2` package which features various useful functions highlighted in the following for the task of calibrating cameras and undistorting images [25].

When using a chessboard pattern, as is the case for most calibration tasks, it provides the `cv2.findChessboardCorners()` function to detect the corners of the board in each given image, which will be referred to as the image points. It further requires the positions of these corner points in 3D, referred to as the object points. In a next step these image and object points, along with one of the calibration images in grayscale figure as the input to the

`cv2.calibrateCamera()` function. The principle of this function is based upon the algorithm proposed by Zhang [26]. Hence, it returns the camera matrix, the distortion coefficients, as well as the rotation and translation vectors.

Now that the intrinsic and extrinsic parameters of the camera system are known or improved, the camera calibration process is completed. With these calculated parameters one can now turn the attention to the problem of image distortion and especially its solution. The `cv2` library also provides a function for this task. Based upon the calculated camera matrix and distortion coefficients `cv2.undistort()` takes in a raw, implying distorted, image and returns the undistorted result.

There also exist other software-based methods for the task of alleviating image distortion based on the principle of remapping. It consists of finding mapping functions in x and y direction from the distorted to the undistorted image and then using a remapping function. Concretely, `cv2.initUndistortRectifyMap()` uses the camera matrix and the distortion coefficients to compute a mapping function both in x and y direction. `cv2.remap()` takes these mappings and the distorted image to create an undistorted version of it. Both the `cv2.undistort()` and the remapping method deliver however the same results.

The step of camera calibration and especially treating image distortion, when working with already captured image data, is crucial to image processing. It allows reproducible results and enhances the accuracy of different CV algorithms used in safety relevant ADAS in the automotive sector. Now that the mathematical basics of digital images and the functioning of modern camera systems used in the automotive industry are covered, different image processing techniques relying on precise camera systems and images are introduced.

2.1.2 Stereo Vision

Stereoscopic or simply stereo Vision is a method to extract pixel-level depth information from a set of 2D images by triangulation. It relies on a number of horizontally displaced cameras, usually two, and uses a similar approach to the human vision with two eyes delivering two slightly offset images. It is a passive sensing method, meaning it measures the distance of objects by receiving information about their relative position and structure in the images [27, p. 2].

Even though the majority of AV and cars providing some level of autonomous navigation use lasers as their primary sensor [28, p. 211], a stereo setup is often used in parallel. Compared to laser sensors such as LIDAR, it provides a very cost-effective alternative and support for object detection and distance measurements [29, p. 5826]. Furthermore, stereo cameras are much more efficiently packaged than their laser-based counterparts thus setting them up for a more flexible and widespread deployment on AV.

The flow chart in Figure 2.7 illustrates the different stages of stereoscopic vision. These three steps necessary to extract 3D depth information and to create a detailed depth map from a set of 2D images as well as the setup of a stereo camera system are explored in the following. Throughout this work only stereo systems with two technically identical cameras are considered. Furthermore, a parallel camera setup, yielding parallel image planes, is assumed and thus considerably simplifies the depth calculation.

Figure 2.7: Flow chart of Stereo Vision method

In a conventional stereo system, two calibrated cameras (LC and RC) individually take one snapshot of the same scene at the exact same time. The lenses of these two cameras are horizontally distanced by the baseline B as shown in Figure 2.8. θ_0 is the horizontal angle of view of both cameras. In a next step, it is important that once these images are taken, they are undistorted as described in the previous subchapter. This ensures accuracy throughout the stereo application and guarantees the correctness of the disparity map and thus of the depth estimation. Likewise, this warrants the stability of the subsequent algorithms because they operate on a pixel level and as a result are highly sensitive to distortion [30, p. 9].

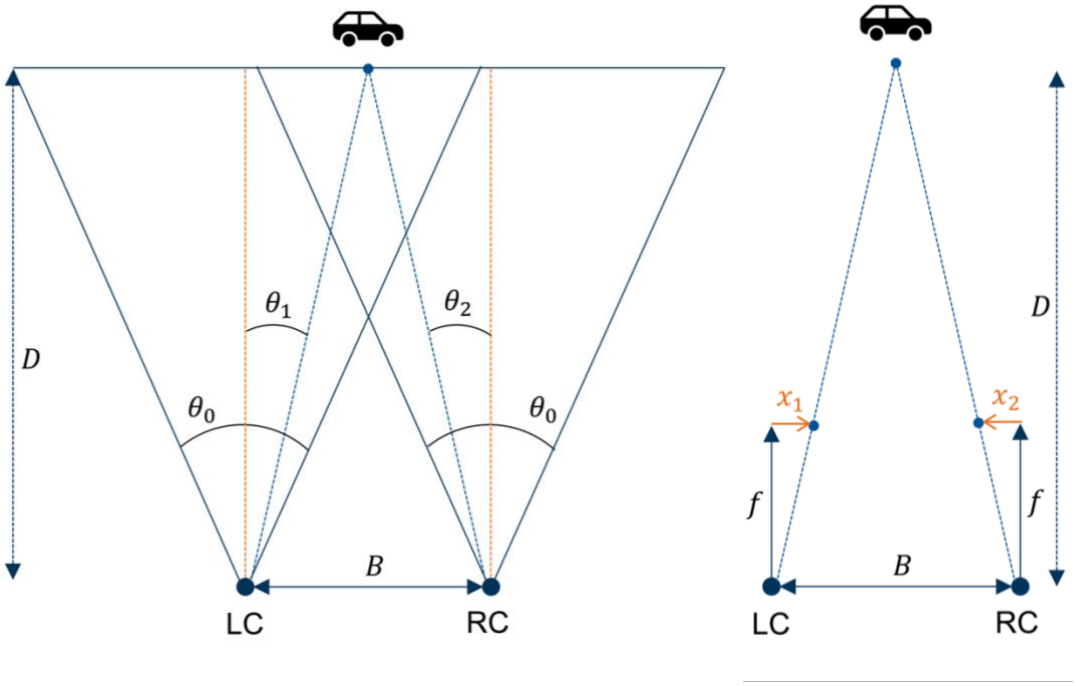

Figure 2.8: Setup the stereo cameras and values used for depth calculation.

The image capture step yields two slightly shifted images of the identical scene. However, now the most complicated and computationally expensive part of stereo vision lies ahead: matching [30, p. 2]. It consists of identifying the same objects, shapes, or color instances in both of the available images on a very small scale and matching them to one another. In a first step different image preprocessing techniques are applied in order to reduce the computational cost of pixel-level operations [31, p. 238]. One way to achieve this efficiency gain is to reduce the resolution of both images to a manageable level to significantly decrease the number of individual pixels. Additionally, if the cameras originally capture RGB images, another means to improve the computational complexity is the conversion to the grayscale color space, which roughly divides the complexity by three. Both compression techniques do not meaningfully interfere with the accuracy of the final result [30, p. 2].

Although the images depict the same scene, they are still taken from slightly varying camera positions thus leaving some pixels without a match in the other image. These pixels are then filtered out and a new region of interest (ROI) considering only the pixels with a match in the neighbor image are kept. There exist a number of matching algorithms, but most rely on some kind of feature extraction as a first measure, where high-level attributes such as cars, pedestrians, and buildings in automotive applications are detected. Afterwards, a pixel-level matching algorithm has to solve the correspondence problem, meaning to find a unique mapping between the points of both images. Fortunately, because the image data has already been undistorted, this correspondence problem becomes a lot less complex. That is to say, the correspondence problem boils down to search for a matching pixel along the same horizontal scanline, the epipolar line, on which the original pixel is found in the other image [30, p. 4]. Bhatti [30, pp. 5–7] identifies six distinct properties or restrictions of the correspondence problem:

1. Similarity: matched pixels should have similar attributes such as color, shape, and size
2. Uniqueness: every pixel has one unique match in the second image
3. Positional order: if two features appear in a given order in the first image, they have to appear in the same order in the second image
4. Disparity continuity: changes in the disparity values are generally smooth
5. Structural relations: identical geometrical features between matched pixels
6. Epipolar restriction: matched pixels lie on the same epipolar line

The different horizontal coordinates of the matched pixels are due to the slight shift of perspective between both cameras, because they are separated by the baseline B. This difference in horizontal coordinates in the respective images is known as the disparity d' and calculates to $x_1 - x_2$. The disparity d' is one of the key values from the stereo vision method and is used to create disparity maps as in Figure 2.9 and crucially to compute the depth D of image points. Considering the values, the stereo setup presented in Figure 2.8, and that we assumed a parallel setup, we can now compute the depth D in m as follows:

$$D = \frac{Bf}{d'} = \frac{Bf}{x_1 - x_2} \tag{2.3}$$

where f is the focal length of the cameras in pixels.

We observe that the depth is thus inversely proportional to the disparity value. This in turn can lead to errors in the proximity of the cameras, where disparity values can demonstrate some discontinuities. Furthermore, once disparity values become very small and approach 0 (at very distanced points), even small errors in the disparity value, lead to large errors in the real-world depth, reducing the precise range of a stereoscopic vision setup [34, pp. 4–5].

As such stereo vision provides a cheap and flexible alternative or support to the more widely deployed active sensors for distance estimation such as LIDAR or RADAR. Furthermore, a stereo system delivers more robust performance and stability in tricky visibility conditions experienced during night-time driving or in poor weather conditions [33, p. 34]. Thanks to the improvements in modern hardware, the moderate computational cost of stereo vision systems

is easily overcome enabling real-time applications. Therefore, stereo setups could play an important role in the autonomous close-range navigation of AV.

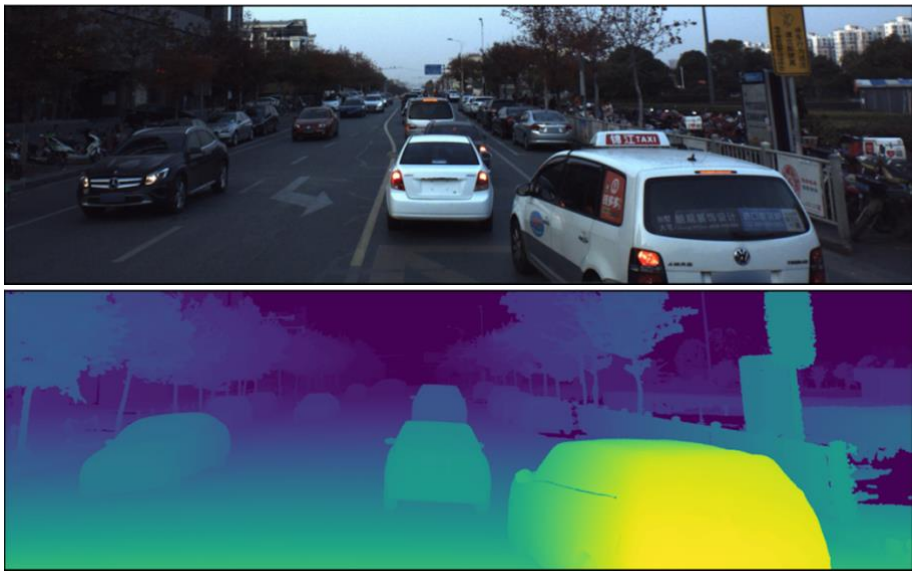

Figure 2.9: RGB image at the top and the corresponding disparity map at the bottom. Pictures from the ApolloScape dataset by Baidu [34].

2.1.3 Color Spaces

Most modern digital cameras now capture color images. These images are most commonly represented on electronic devices and personal computers using the RGB additive color model which is based on the Young-Helmholtz theory of trichromatic color vision. In this instance, every pixel of an image is assigned three color values, denoting the intensity of each color, i.e. red, green, and blue. These intensity values are usually 8-bit format and thus range anywhere from 0 to 255, where the vector $(0, 0, 0)$ would represent the black color and $(255, 255, 255)$ the white color. The choice for the red, green, and blue color components is derived from the physiology of the human eye. The photoreceptive cone cells are maximally stimulated when exposed to large differences in wavelength, which is provided when using RGB [35, pp. 93–94]. This in turn makes for easy color recognition. Grayscale images with only one intensity value are obtained by taking the average of the three RGB color values.

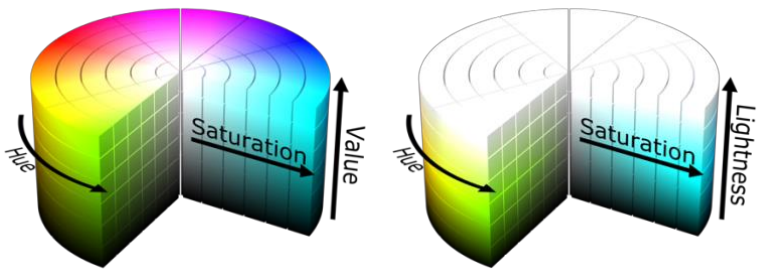

Figure 2.10: HSV color cylinder on the left and HSL color cylinder on the right

Apart from grayscale images there also exist the HSV (**H**ue **S**aturation **V**alue) and HSL (**H**ue **S**aturation **L**ightness) color spaces. These color spaces allow for easier feature extraction based on object colors. In fact, the hue value alone categorizes a color. Subsequently, the S and V

values only account for the color intensity and its darkness or lightness respectively which can be observed in Figure 2.10. As such, converting RGB images to one of these color models lends itself very well to detect and filter out certain colors in images. It can for instance be used to detect lane markings and thus support real-time road detection algorithms for autonomous driving. Indeed, Zhou et al. [36, p. 661] remark that such a conversion increases the stability of their road detection algorithm thanks to enhanced lighting conditions. Figure 2.11 depicts an image in RGB format and its counterpart in the HSV color space. We can observe that indeed, features such as the road surface and its markings which are illuminated differently in the RGB image and therefore appear quite different to the human eye, are marked in the same color in the HSV image. This property can be suitably exploited when the tasks of image segmentation, i.e. classifying pixels to distinct categories, or lane detection based on road markings arise.

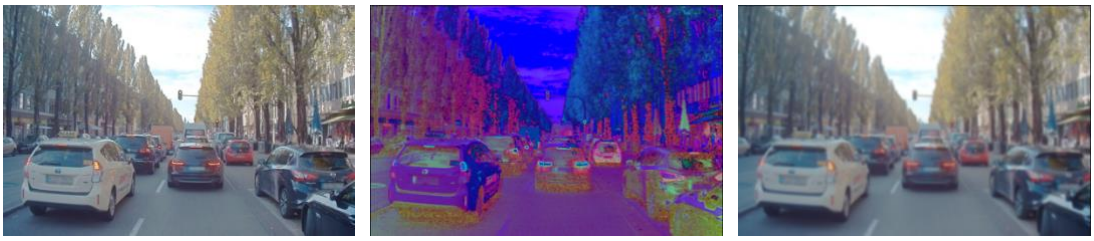

Figure 2.11: From left to right then top to bottom: RGB image, HSV image, Blurred Image. Pictures from A2D2 dataset by Audi [10].

2.1.4 Gaussian Blurring

Modern cameras used in the automotive sector achieve extremely high resolutions. In order to process these large images in real-time, ADAS applications such as road detection need to reduce the detail level of them. In fact, the Canny edge detection algorithm presented in the following relies heavily on smoothed inputs, since it is based on derivatives [37, p. 1405]. The effect of blurring on the Canny algorithm can be noticed in Figure 2.12, where the non-blurred image is much noisier than its blurred counterpart.

Gaussian blurring is a preferred image pre-processing technique to denoise images and make them more manageable. It removes unnecessary details and smooths the input image A by convoluting it with a gaussian filter kernel $H_{i,j}$ of size $2k+1 \times 2k+1$ and σ the standard deviation of the Gaussian distribution as seen in Equation 2.4.

$$A' = H_{i,j} * A \tag{2.4}$$

where $H_{i,j} = \frac{1}{2\pi\sigma^2} e^{-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}}$, and $1 \leq i,j \leq (2k+1)$

The smoothed result of such a convolution with standard deviation of $\sigma = 1$ and a kernel size of 7×7 can be observed in Figure 2.11. A less detailed and denoised image reduces the computational complexity of subsequent CV algorithms such as edge detection and segmentation. As a matter of fact, it is often the first step and provides a solid basis for the task of image segmentation [38, pp. 393–395], which will be explored in an upcoming subchapter.

2.1.5 Canny Algorithm

In 1986, John Canny introduced one of the most important image processing algorithms of its time which is still widely in use today [39]. The Canny algorithm is a type of edge detection algorithm, which when applied to images returns the edges of the objects present in the frame. It reduces these objects to their contours and is therefore extremely useful for detecting them during image analysis. In the automotive and especially the autonomous space it is, in conjunction with other powerful algorithms, employed to detect obstacles on the road and warn the driver or instruct the AV to perform a maneuver. Furthermore, it can be used in the context of road and lane detection, a vital component for full autonomy [40].

Canny [39, p. 680] postulated three performance criteria his novel algorithm should fulfill:

1. Good detection, meaning a low probability of missing real edge points as well as a low probability of falsely identifying non-edge points
2. Good localization, such that edge points should be close to the center of the real edge
3. One single response for a single edge

Since the algorithm sits at the beginning of most other feature detection techniques which depend on it [40, p. 2120], it is crucial that it delivers satisfactory and stable results every time. Instabilities may lead to disruptions in the operation of AV and therefore to safety-critical problems. Hence, in the following a step-by-step guide for edge detection based on the Canny algorithm is presented. This implementation of the Canny algorithm builds upon the more light-weight Sobel edge detection algorithm and thus yields a more detailed and precise result [41, p. 135].

First off, in order to satisfy the first performance criterion, the noise level of the considered image needs to be reduced by blurring it, as high-resolution images contain too many details leading to an elevated probability of falsely detected edge points. The implication of this phenomenon and the remedial blurring can be observed in Figure 2.12. A common blurring technique is the Gaussian blurring introduced above. The kernel size needs to be adjusted to the initial image dimensions in order to deliver satisfying results.

Furthermore, if the image at hand is an RGB scale image, as is usually the case with modern cameras, it needs to be transformed to the grayscale space. The Canny edge detector can only operate with one intensity value per pixel. This does however not negatively impact the outcome of the edge detection in terms of quality and stability. If anything, this transformation aids in stabilizing the algorithm.

Next comes the core idea behind the Canny approach: the gradient calculation. In fact, the algorithm flags a pixel or a series of pixels as an edge if their gray intensity values and those of their neighboring pixels differ sharply. Since we are interested in real world traffic images, edges can occur in all directions. For the calculation of the approximate derivatives in x and y direction the Sobel operators K_x and K_y are used respectively. They are 3×3 convolution kernels yielding the intensity derivatives of every pixel when convolved with the matrix of the blurred, grayscale image A as seen in Equations 2.5 and 2.6.

$$G_x = K_x * A' = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A' \tag{2.5}$$

$$G_y = K_y * A' = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A' \tag{2.6}$$

The matrices G_x and G_y contain the approximated horizontal and vertical derivatives for every original pixel of A'. From these derivatives we can calculate the edge strength G as well as the direction θ of the edge as follows:

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.7}$$

$$\theta = arctan\frac{G_y}{G_x} \tag{2.8}$$

The newly acquired matrix G contains the edge information of the input image A. Depending on the magnitude of the individual values $g_{i,j} \in G$ some more prominent edges are thick and other more subdued edges will present a thinner shape.

To mitigate this issue and obtain a more concise edge image with uniformly thin edges, the application of the non-maxima suppression lends itself well. It consists of iterating through the gradient intensity matrix G and finding the pixels with the maximum edge value in edge direction.

The previous step delivers an edge image with equally broad edges, but their intensity values still vary as a function of the initial color and illumination in the RGB image. Since one looks for an edge image with clear and distinguishable edges, every detected edge should be relevant and as a result have the same intensity. Therefore, a double thresholding method is applied to categorize pixels as strong, weak, or irrelevant. Pixels with an intensity greater than the upper threshold are flagged as strong, those bounded by the upper and lower threshold are weak, and those below the lower threshold are discarded.

The pixel classification is now used to track all the relevant edges and connect them, if necessary. This final step is called hysteresis. The approach is straightforward: if a pixel is marked as weak, we investigate its immediate vicinity. If one neighboring pixel is strong, the weak pixel is processed as strong. In the other case it is discarded. As a result, we only retain the relevant contours and obtain a satisfying result as observed in Figure 2.12.

Figure 2.12: From left to right: Original RGB image, Canny edge image without preceding blurring, Canny edge image with preceding blurring. Pictures from the A2D2 dataset by Audi [10].

In the software practice the OpenCV library offers a simple function to apply the canny edge detector to any grayscale image. Its `cv2.canny()` function takes in a preferably blurred, grayscale image as well as two threshold values for the double thresholding and returns a binary edge image such as the one in Figure 2.12.

The traditional Canny algorithm as described above is still widely used today in practice. However, Rong et al. [42] note that it is highly sensitive to noise as a result of its gradient calculation and its fixed thresholding values. They propose alternatives based on adaptive threshold selection methods to stabilize the algorithm and obtain satisfactory results in every iteration, which is critical for reliable ADAS.

2.1.6 Image Segmentation

Image segmentation is a technique to divide and partition an image on a pixel level into parts, called segments [43, p. 809]. It is one of the most promising techniques for visual scene understanding and as such it can be of great value to the development of AV as it allows for very precise obstacle localization and hence also avoidance [44, p. 669]. But this great performance comes at a high computational cost. In fact, considering the different learning problems from visual information, i.e. image classification as well as object localization and detection, it sits at the top of the food chain in terms of extracted information but also in terms of complexity [45, p. 162] [46, p. 1].

Hence, it became one of the defining problems of CV in recent years and various deep learning techniques mostly based on an encoder-decoder pair have been developed as a consequence. A trained encoder CNN detects high-level features such as cars or pedestrians. This classification is later on enhanced with information from a decoder which assigns a category or segment to each pixel [47, p. 3].

Mathematically it can be described as follows [48, pp. 420–421]:

Let I be the set of image pixels. After segmentation a set of n different regions $\{S_1, S_2, \dots, S_n\}$ is obtained with the resulting properties:

- $\bigcup_{i=1}^{n} S_i = I, \quad$ with $S_i \cap S_j = \emptyset$, for $i \neq j$
- S_i is a connected region

In the following three concrete segmentation tasks of importance to autonomous driving are presented.

Foreground Segmentation

Foreground segmentation is the most low-level task of image segmentation. It focuses on assigning each pixel to one of two categories: the foreground or the background of an image. Depending on the application domain, different techniques for foreground recognition are used. As Ortego et al. [49] elaborate that unsupervised (detection of spatio-temporally relevant objects), semi-supervised (pursue initially segmented objects), and supervised approaches exist.

In the automotive sphere, focus lies on discerning moving objects from stationary objects. Accentuating moving objects to the foreground mask enables AV to detect other traffic participants and supports the task of autonomous navigation. Since this is a rather low-level method, real-time applications of foreground segmentation are possible [50], which allow traffic tracking methods. Figure 2.13 portrays an image of the ApolloScape dataset [34] and its corresponding foreground mask. One observes that, indeed the moving parts of the image, in this case other cars are part of the foreground masks, thus achieving the described goal.

Figure 2.13: RGB image of the ApolloScape dataset and its corresponding Foreground mask. Pictures from the ApolloScape dataset by Baidu [34].

Semantic Segmentation

Semantic image segmentation can be understood as the labelling of pixels to pre-defined semantic classes [45, p. 162]. After this process of ascribing semantic labels to each and every pixel of the image at hand, a colored mask is laid on top of the image. Hence, every different color of this mask codes for a specific class as can be observed in Figure 2.14.

In an automotive setting, common semantic classes include cars, pedestrians, the road surface, or road signs. Such a detailed segmentation of the scene has great potential to support autonomous driving, by providing good visual scene understanding. Thanks to modern computer architectures, some semantic segmentation methods can be achieved in real-time furthering its importance.

Semantic segmentation provides however also benefits for the subsequent data analysis of the recorded scenes. It allows for a low effort evaluation of traffic circumstances and provides a basis to investigate the free road surface in front of the car. Such information can then be used to enhance map information or inform an AV of tricky road and traffic conditions ahead of its arrival.

Figure 2.14: RGB Image and its corresponding semantic segmentation mask. Different types of cars are colored in different shades of red. Pictures from the A2D2 dataset by Audi [10].

Instance Segmentation

Whereas in semantic segmentation, pixels of objects of the same semantic class are colored the same way, instance segmentation attributes different labels to multiple instances of the same class. It can thus be considered as a combination of semantic segmentation and object detection at the same time [51]. It requires to assign objects a certain semantic class, but further to detect the contours of every object to distinguish it from different objects of the same class. The computational complexity is therefore also increased [52, p. 7] and instance segmentation does not lend itself as well to real-time applications as a consequence.

2.2 Sensor Overview

Today there are a little over one billion cars on the road worldwide. The World Economic Forum predicts that this number will more than double in the years leading up to 2040 [53]. This increase will consequently fuel a surge in traffic load and thus also road accidents. In fact, in 2015 over 1.3 million deaths were due to traffic injuries as per the World Health Organization. As most of today's accidents can still be attributed to human error [54, p. 178], the advent of autonomous vehicles is poised to improve traffic safety and prevent fatal accidents to an absolute minimum – a goal called "Vision Zero" [55, p. 14].

However, in order to achieve these goals AV need to execute safe and well-informed decisions in a reliable fashion. For this matter a host of sensors and ADAS are put to use in every autonomous or partly autonomous car to guarantee a safe operation. Thankfully the cost of safety-critical sensors sharply decreased over recent years and as a result 92.7% of new cars sold in the United States (as of May 2018) are now equipped with at least one ADAS feature, such as emergency breaking or lane keep assist [56, pp. 7–13].

The described ADAS features all heavily rely on visual sensors of some sort. They can be thought of as the indispensable virtual eyes, in analogy to human vision of an automated vehicle, which detect obstacles to avoid, find drivable paths in the near surrounding, and sense approaching traffic dangers. Furthermore, in order to make decisions and control the AV's movement, information about its speed, acceleration, both in radial and axial directions, and its GPS location are needed. In most AV or partly autonomous cars, these are computed in real-time and stored in another type of sensor: the inertial measurement unit or IMU. The IMU is thus crucial to the task of Simultaneous Localization and Mapping or SLAM.

Visual sensors can be divided in two categories: passive and active sensors. Cameras and stereo cameras can be considered as passive, because they merely receive information in the form of incoming light rays and form an image based on them. Active sensors on the other hand mostly rely on time of flight (TOF) information. Examples are LIDAR and RADAR sensors which emit different waves, record the time it took for these waves to be reflected and redirected onto them, and subsequently calculate the position of the reflected point. Kovacic et al. [57, p. 26] further elaborate that a combination of active and passive sensors (e.g. a camera system combined with a LIDAR sensor) enhances the systems adaptability to changing environments and therefore increases the system's reliability.

To guarantee an advantageous interaction between these different sensors, sensor fusion is of high importance [58, p. 3]. Since different sensors might experience availability issues in different conditions or at different detection ranges, sensor fusion can mitigate this problem by using information from multiple sensors, thus reducing the dependence on a single one. An overview of the autonomous navigation process is presented in Figure 2.15, where one can observe the different sensor information being processed in a parallel fashion.

In the following, different optical and SLAM sensors are presented and evaluated based on their advantages and disadvantages. A comparison between them is also provided at the end of this Subchapter in Table 2.1.

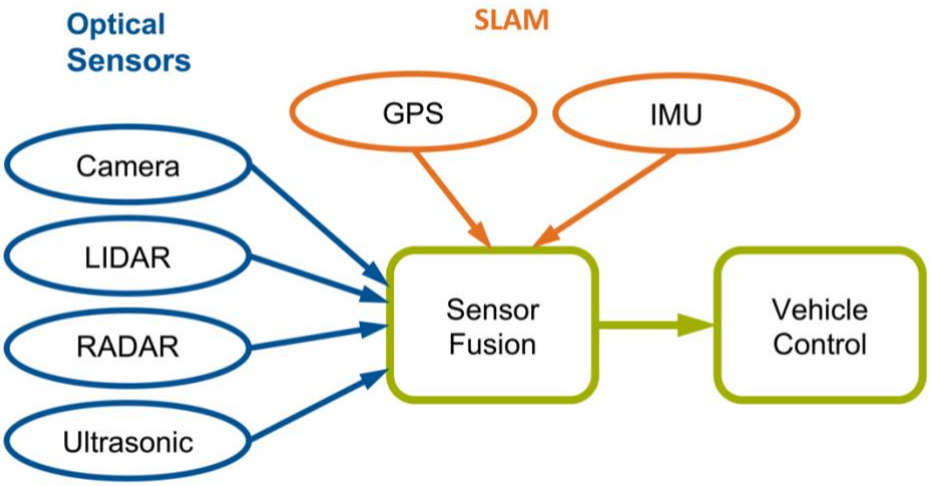

Figure 2.15: Overview of the Sensor Fusion in an AV

2.2.1 LIDAR

LIDAR stands for **Li**ght **D**etection **a**nd **R**anging, and sensors of this type are an optical analog to well-known RADAR sensors. These sensors are widely used in AV, but since their cost is quite elevated, the most precise and sophisticated systems are currently only employed in test and development vehicles [54, p. 183]. It is a sensor, that actively emits radiation and collects its subsequent reflection from objects in its FOV.

Commonly, near infrared (NIR) radiation at 850-950 nm or short-wave infrared (SWIR) at 1550 nm is used to guarantee eye safety as well as a significant range [61, pp. 53–54], [62]. In fact, the shorter the used wavelength, the more energy the light beam carries, and as a result the larger the range of LIDAR scanners, which ranges from 100 m up to 400 m [61, p. 266]. Two different modes of operation exist: a pulse-based and a phase-based method. Pulse-based setups send out short (1-10ns) individual laser pulses, whereas phase-based systems use a continuous wave modulation [62, p. 7]. In either case, the distance d of a reflected target from the LIDAR sensor is calculated on the basis of its TOF Δt as described in Equation 2.9.

$$d = \frac{c}{n}\frac{\Delta t}{2} \tag{2.9}$$

where c is the speed of light and n the refractive index of the light travelling medium

As such, LIDAR sensors are especially suited for extracting 3D depth information from a scene and gain a high-quality spatio-temporal knowledge of the environment. This spatial information is used to create dense and precise (error magnitude of 3-15 mm as per [63]) point clouds, an example of which can be observed in Figure 2.16.

LIDAR data in the form of point clouds is employed in real-time for emergency breaking and collision avoidance mechanisms. Furthermore, it can inform the AV of the weather conditions based on the average detection range of its signal. In rainy or foggy conditions, the signals may be reflected by raindrops or aerosols and thus the detection range decreases significantly [61]. Moreover, bad weather conditions with lots of precipitation can reduce the stability of the detection and large variability of detection range is observed. LIDAR information is also useful in

the posterior analysis of different autonomous datasets. It delivers precise depth information and can be used for object classification purposes in conjunction with camera data.

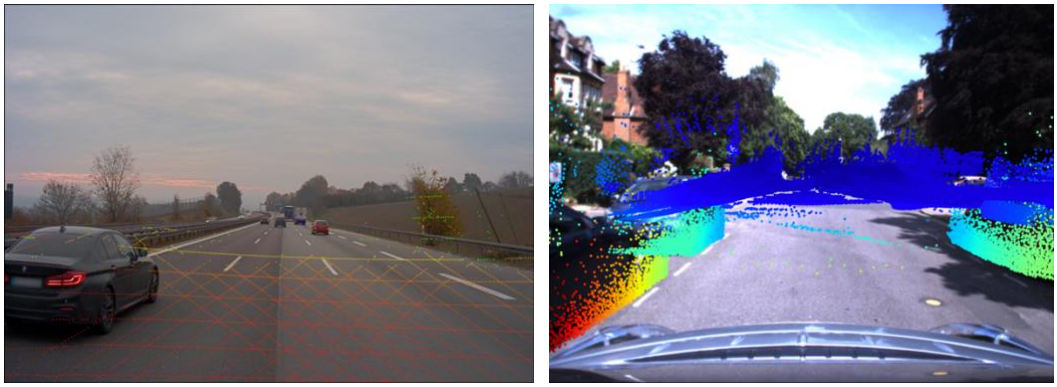

Figure 2.16: LIDAR point clouds projected into corresponding images of the A2D2 dataset on the left and the Oxford RobotCar dataset[64]

LIDAR systems also provide a very wide horizontal coverage. In the case of revolving scanners, the horizontal FOV is 360°. Furthermore, they represent a good support for camera sensors at nighttime or dark conditions. As it gets more difficult to extract useful information from underexposed pictures, the LIDAR sensor is only partly restricted at larger distances due to the lower reflectivity of dark surfaces. One advantage for LIDAR scanners during the night consists in the fact that its signals do not interfere with solar radiation [65, p. 26].

Currently Velodyne laser scanners are the most popular among AV development projects [59, p. 51], [60]. Similarly, well-known datasets such as KITTI [66], and A2D2 [10] use LIDAR scanners from the Velodyne brand, because of their high precision, 360° FOV, and long range. Even though their cost has come down considerably over the past years, these sensors remain too expensive for widespread commercial use.

2.2.2 RADAR

RADAR, short for **Ra**dio **D**etection **a**nd **R**anging, is another type of visual sensor deployed to support ADAS methods in AV. Similar to the aforementioned LIDAR unit, RADAR scanners are also part of the active sensors. Contrary to LIDAR it sends out millimeter radio waves from a radio frequency generator. Reflected radio waves from targets are then picked up by the patch antenna, amplified, and denoised. Based on the elapsed time between sending waves out and receiving them, a direct distance measurement is carried out.

Even though radio waves cannot resolve the shape and texture of targets, they have other advantages over NIR and SWIR radiation. They are not, or only marginally, affected by tricky environmental conditions such as rain, fog, and high luminosity [68, p. 392]. In addition to this, radio waves experience a slight frequency shift when being reflected by a moving object such as other road participants due to the Doppler effect. The Doppler effect is described in Equation 2.10, from which the velocity v of the moving object can be directly computed.

$$f_r = f_t \left(\frac{1 + \frac{v}{c}}{1 - \frac{v}{c}} \right) \tag{2.10}$$

where c is the speed of light, f_r the received shifted frequency, and f_t the original frequency of the radio waves

The combination of the spatial localization and the direct calculation of object velocities makes RADAR the obvious choice for ACC [54, p. 182]. This ADAS is widely distributed among modern cars, not only because of its relative ease of implementation, but also because RADAR sensors only cost a fraction of a corresponding LIDAR scanner [69, p. 864]. Moreover, RADAR technology can be adapted, by changing the wavelength, to focus on shorter range, such as in parking situations, or in medium to long range for ACC [70, p. 23]. It is however less suited for static object detection and is more prone to interference, which can cause precision issues.

2.2.3 Ultrasonic Sensor

Ultrasonic sensors, like RADAR and LIDAR also reside in the category of active sensors. Their principle for object detection and ranging similarly relies on actively sending out ultrasonic waves and gathering their reflections. Pulses of ultrasonic waves at a frequency ranging from 20 kHz up to several MHz are emitted by piezoelectric resonant transducers. Reflections are collected by a receiving transducer and analyzed based on their TOF and wave pattern. The distance d of the detected objects is then calculated using Equation 2.11.

$$d = c\frac{\Delta t}{2} \tag{2.11}$$

where c is the speed of sound in air and Δt the TOF.

Ultrasonic sensors show a good resilience to bad visibility conditions as Babak et al. [71, p. 4] point out in their work. However, due to interference from other ultrasound waves and obstacles made of wave altering materials, the accuracy and range of ultrasonic sensors is quite limited [72, p. 232]. Furthermore, their precision suffers under high vehicle speeds [73, p. 143]. Hence, and because their good detection range is restricted to only several meters, they are employed in close range ADAS such as park assist technology, where RADAR and LIDAR sensors have poorer resolution. To increase the detection and recognition accuracy, approaches using a combination of ultrasonic sensors and artificial neural networks are developed [74].

2.2.4 Inertial Navigation System

Self-localization and motion understanding is one of the central principles for safe navigation of AV. Not only does this involve the object detection and ranging of the direct environment provided by optical sensors, but also the monitoring of physical quantities of the car. Additionally, AV heavily rely on high precision GPS data to be able to find drivable paths when other sensors are not available [54, p. 182].

The inertial measurement unit or IMU is equipped with a host of different sensors to fulfill the task of measuring and calculating important physical quantities such as speed, lateral acceleration, longitudinal acceleration, steering angle, yaw, pitch and roll of the vehicle. Common setups include three gyroscopes and three accelerometers and achieve good temporal resolution as high as 100 Hz [76, p. 8], [77, p. 572]. IMUs need to be well calibrated in order to achieve high accuracy. In fact, biases and the drift of gyroscopes accumulate in the IMU due to

a misalignment with the frame of reference, which can lead to faulty measurements and thus to compromising the AV navigation process [76, p. 1].

A Global Navigation Satellite System or GNSS sensor provides the necessary positioning data. The most common example of such a GNSS is GPS. GPS receivers use signals sent by multiple satellites to determine the longitude, latitude, and altitude of the car for positioning in a global frame of reference. Temporal resolution is however much lower [76, p. 572] and the time delay between two signals in combination with the position data can be used to compute an average vehicle velocity. To improve GPS data and correct common errors linked to it, real-time kinematic positioning or RTK techniques which rely on a fixed base station and the vehicle as its rover are used. This brings error magnitudes down to centimeter levels [77].

Together the IMU and GPS sensors make up the Inertial Navigation System or INS. The combination of both allows for more precise measurements and remedies the misalignment problems faced by the IMU. There exist methods employing Kalman filters to produce positive feedback loops and reduce the INS errors even further, since high precision is of critical importance to the navigation process of AV [78].

Table 2.1: Comparison of widely employed optical sensors

Sensor	Application	Benefits	Shortcomings
Stereo vision	Object detection and localization	Depth information Low cost	Poor performance in bad visibility conditions Computationally expensive
LIDAR	Object detection and localization ACC	High precision High range	High cost Reduced performance in bad visibility conditions
RADAR	Moving Object localization ACC	Low cost Resilient to bad visibility conditions Object velocity information	Prone to interference Low recognition of static objects
Ultrasonic	Park assist	Low cost Resilient to bad visibility conditions	Low range Prone to interference Increased variance with speed

2.3 Available Datasets

Fully autonomous driving is a notoriously difficult task. Since its humble beginnings in the 1980s [79], a lot of development and progress has been made to shift fully AV from the unimaginable into the possible realm. Autonomous mobility represents an opportunity to drastically change

and improve traffic in a myriad of ways. It holds solutions to reduce congestion, thanks to precise GPS navigation and vehicle to vehicle communication [80]. Furthermore, autonomous mobility is poised to increase road safety. Thanks to multi-modal sensor setups, AV gain a detailed scene understanding of their immediate surroundings and are mostly able to take action in a faster, more reliable, and more rational fashion than their human counterparts. In addition to these benefits, driverless mobility will enable transport to become much less carbon intensive on account of smarter navigation techniques [81].

Today, mostly legal barriers stand in the way of deploying and operating commercially available AV. In fact, even though autonomous technology has come so far as to operate and function reliably well in most conditions, traffic can pose very complex settings with a lot of different categories and moving parts such as displayed in Figure 2.17. For driverless cars to be adopted both legally and by consumers, they need to be able to cope with such difficult scenarios in an accurate, robust and real-time manner [82, p. 1341].

The advancements made up to today and those needed for a fully driverless future are largely thanks to datasets recorded by research teams all over the world. The open-source nature of them allows every developer to use them, work on them, and contribute to them as a result. They can act as benchmarks for new algorithms in sensor fusion or object detection thanks to ground truth data [64], [83]. Moreover, most of them provide a learning and testing basis for deep learning methods used in tasks of scene understanding such as categorization, segmentation, and behavior analysis.

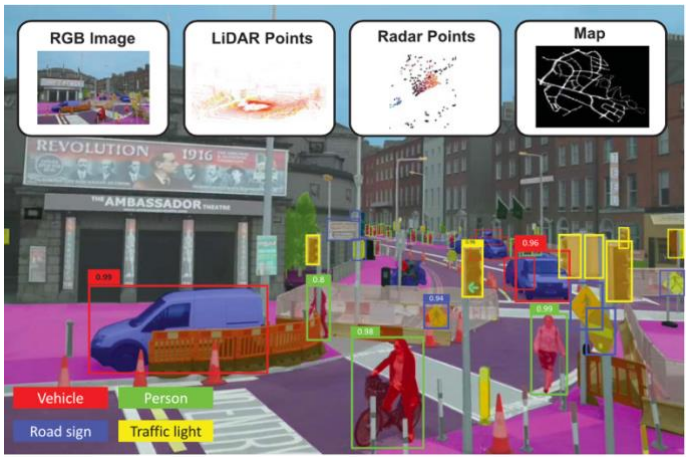

Figure 2.17: RGB image with overlaid semantic mask from the Mapillary Vistas Dataset [46] fused with LIDAR and Radar information to produce bounding boxes for the different objects [82, p. 1341]. It displays a very complex scene with a lot of different classes and plenty moving diverse movement patterns.

This open availability and the competition character [66], [84], [85] most dataset platforms offer for specific AV tasks have allowed the software development behind driverless vehicles to flourish in recent years. In the following, some of the most used and largest datasets for autonomous driving are thus presented and evaluated based on their sensor setups, size, and information they provide.

2.3.1 Presentation

A*3D

The A*3D dataset [83] is multi-modal and uses two cameras as well as a LIDAR sensor. Specifically, two Chameleon3 color cameras from PointGrey are combined with a 64 channel Velodyne 3D LIDAR, which offers a reliable detection range from 0.9 m up to 100 m. Carrying these optical sensors is the A*Star autonomous vehicle, depicted in Figure 2.18.

The A*3D data has been recorded in the whole Singapore area (Figure 2.18) and covers a wide range of traffic, road (urban, highway, car parks, etc.), and lighting conditions. Recording was done in the afternoon and night of a wet season month (March 2018) and of a dry season month (July 2018) in driving speed ranges from 40 km/h to 70 km/h. In total, over 39,000 frames were recorded over this period and annotated with over 230,000 3D bounding boxes spanning seven different classes.

The standout features of this dataset are its high scene and weather diversity and image density. As such, 30% of the frames were recorded at night and the dataset is partly comprised of heavily occluded frames. As a result, it has a high object detection density of 5.9 detections per frame.

Figure 2.18: Left: A map of Singapore with the roads recorded by A*3D in blue and those driven by nuScences in red. Right: A*3D recording vehicle and sensor setup.

A2D2

The A2D2 dataset [10] has been recorded by Audi in an effort to further their AV development. It uses a setup of six 2.3 MP color cameras in addition to five Velodyne VLP-16 LIDAR scanners for a full 360° coverage. Furthermore, an IMU combined with GPS is employed to gather SLAM information of the test vehicle – an Audi Q7 e-tron.

The over 392,556 images, with corresponding LIDAR and INS data, were mostly recorded in urban settings in the German cities of Gaimersheim, Ingolstadt, and Munich. In addition to these RGB images, over 41,000 semantically labelled images are provided spanning 38 classes. For nearly 12,500 of these semantically labelled frames, 3D bounding boxes are provided.

The extremely detailed segmentation as well as the 360-degree LIDAR and camera coverage are the standout features of this large dataset.

ApolloScape

The ApolloScape dataset [84] has been recorded by a research team of Baidu Inc.. It uses two VUX-1HA LIDAR scanners with a 360° FOV and a range of 1.2 m up to 420 m in combination with a VMX-CS6 stereo camera setup and an IMU with GPS sensors.

It covered a distance of around 1,000 km at an average speed of 30 km/h in 10 Chinese cities. It consists of over 143,000 video frames which were semantically annotated using 24 different classes.

One exceptional feature of this dataset is its lane marking segmentation, where different lane markings are identified and aid the AV to navigate safely.

Cityscapes

The Cityscapes [86] dataset was developed with the aim of better urban scene understanding in conjunction with the Daimler AG R&D, the TU Darmstadt, MPI Informatics, and the TU Dresden. It uses a single stereo camera setup with two cameras, recording images at a frequency of 17 Hz.

It was recorded in over 50 cities, mostly in Germany and in some neighboring countries. For the scene parsing and segmentation 5,000 frames were selected for fine grained results and 20,000 frames are provided with coarse pixel-level annotations.

Cityscapes offers a varied urban environment with high object density and very detailed segmentation.

H3D

The Honda Research Institute 3D dataset or H3D [87] was recorded using one Velodyne HDL-64E 3D LIDAR scanner, aggregated with three PointGrey Grasshopper3 video cameras and an Automotive Dynamic Motion Analyzer with GPS.

In total, 27,721 frames were collected and annotated with 3D bounding boxes. The vehicle covered large urban parts of the Bay Area, including San Francisco and Mountain View.

KAIST Multi-Spectral

The KAIST Multi-Spectral dataset [88] uses a similar sensor setup to the A*3D and H3D datasets including a Velodyne HDL-32E LIDAR, an OXTS INS, and two PointGrey color cameras. In addition to this it however also employs a FLIR thermal camera for increased accuracy in the stereo matching algorithm

KITTI

The KITTI dataset [66] was recorded by a research team of the Karlsruhe Institute of Technology (KIT) in conjunction with the Toyota Technological Institute. It is one of the pioneering autonomous driving datasets and the first one to use sensor fusion. In fact, it employs a setup of four PointGrey cameras (two grayscale and two color cameras) to form a stereoscopic setup, a Velodyne HDL-64E LIDAR, as well as an OXTS INS.

It was mostly recorded in the urban and suburban parts of Karlsruhe, with some sequences recorded on the KIT campus during September and October of 2011. The total size of the initial dataset is about 25,000 annotated frames and also includes 3D bounding box annotations.

The KITTI website hosts a number of competitive leaderboards in different categories such as segmentation, stereo matching, and scene flow.

nuScenes

The nuTonomy Scenes or nuScenes dataset [89] is the first autonomous dataset that uses the full sensor suite. It employs six cameras, one LIDAR sensor, five RADAR sensors as well as an INS with RTK for full 360° scene coverage.

It was recorded in urban settings in Singapore (see Figure 2.18) and Boston and covers a distance of about 240 km at an average speed of 16 km/h. It depicts 1,000 scenes of about 20 seconds and annotates them with bounding boxes.

Oxford RobotCar

The Oxford RobotCar dataset [64], [67] is a relatively new one that also employs the whole available sensor suite to improve AV software. A Nissan Leaf was equipped with six cameras, two SICK LMS-151 2D LIDAR scanners, one 3D LIDAR and an INS system with RTK correction.

In total, over 1,000 km were covered over the duration of a whole year including all types of weather and lighting conditions. This resulted in over 20,000,000 images obtained in and around central Oxford in the United Kingdom.

It represents a good source of data for similar driving roads in vastly different environmental conditions and therefore allows to analyze how the availability of sensors is affected by external weather and lighting.

Waymo Open Dataset

The Waymo Open Dataset [85] has been generated in collaboration with Google. It uses a sensor setup of five LIDAR scanners as well as five cameras for full scene coverage.

In total, 1,150 scenes of about 20 seconds were recorded, yielding more than 230,000 annotated LIDAR and camera frames. It was mostly recorded in the Bay Area, but a large geographical diversity was paramount for the development team.

2.3.2 Summary and Overview

A general trend one can observe when analyzing the different openly available driving datasets is the usage of at least two sensors. Most commonly cameras are therefore combined with either LIDAR or RADAR sensors to achieve better object detection and above all get a direct distance reading from signals to better localize obstacles.

Furthermore, one notices a trend to use LIDAR sensors over their radio-based counterparts. In the development and research vehicles described above precision is of high importance and the one-time cost for installing LIDAR sensors is thus justified. However, since these scanners are still highly expensive, only few commercial vehicles are equipped with them today [60], [90, p. 3].

Table 2.2: Comparison of well-known autonomous datasets

Name	Sensor						Annotation			Traffic			Diversity			Frames
	C	S	T	R	L	I	S	B	SF	U	H	R	D	N	W	(k)
A*3D	x				x			x		x	x	x	x	x	x	39
A2D2	x				x	x	x	x		x	x		x		x	41
ApolloScape	x	x			x	x	x	x		x	x		x	x	x	143
Cityscapes	x	x					x			x			x			25
H3D	x				x	x		x		x			x			27
KAIST	x	x	x		x	x		x		x		x	x	x		95
KITTI	x	x			x	x	x	x	x	x	x	x	x			25
nuScenes	x	x		x	x	x		x		x		x	x	x	x	40
Oxford RobotCar	x	x		x	x	x		x		x			x	x	x	20,000
Waymo	x				x	x		x		x	x	x	x	x	x	200

Legend: C: Camera, S: Stereo, T: Thermal Camera, R: RADAR, L: LIDAR, I: INS, S: 3D Semantic Labels, B: Bounding boxes, SF: Scene Flow, U: Urban, H: Highway, R: Rural and Residential, D: Day, N: Night, W: Bad weather

2.4 Methods used in the Automotive Context

The SAE is one of the leading authorities when it comes to defining and monitoring the levels of automation AV can reach. Specifically, it outlines six different categories of automation for passenger cars, which are displayed in Figure 2.19. To get to Level 4 or 5 autonomy AV heavily rely on sensor data which is used in various types of ADAS. These systems must operate in a reliable fashion to ensure safe operation and deployment of AV. In the following, two of the most basic but also most crucial methods in ADAS are presented and evaluated: lane detection and free space estimation. In fact, without the current developments in sensor and ADAS technology the leap to autonomy would not be possible.

2.4.1 Lane Detection

For enabling a safe operation of driverless cars, road or lane detection is one of the crucial issues. It is leveraged in several partly autonomous tasks such as ACC, lane departure warning, and lane centering, and also represents a cornerstone of fully automated vehicles.

In some easy scenarios such as highway roads, the task is relatively easy and modern approaches achieve great accuracy. Road surface, geometry, and environmental conditions are however prone to change. This high variability greatly increases the complexity of this important task, hence significantly reducing the stability and accuracy to unsatisfactory levels.

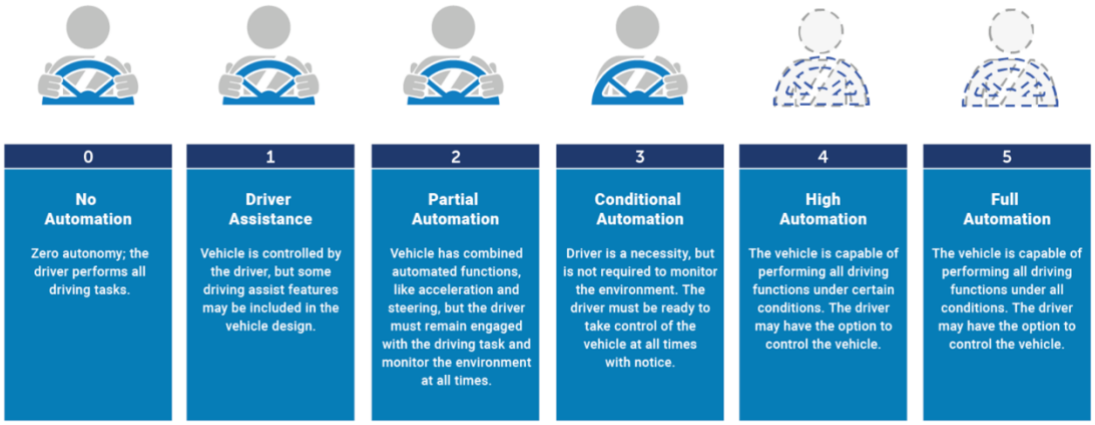

Figure 2.19: Automation levels as defined by the SAE [91, p. 50]

Therefore, to improve lane detection algorithms, the DARPA Grand Challenge and Urban Challenge have both sparked a great amount of research and development in this field since the beginning of the 21st century [92], [93]. Yearly or bi-yearly competitions have allowed research teams all over the world to test their work and push the envelope regarding AV and as such also lane detection systems. In recent years, different approaches or combinations of methods have crystalized, which will be presented in the following.

The most obvious and widespread method for lane detection relies on a monocular camera setup. Bar Hillel et al. [94, p. 5] identify two main reasons for this. On the one hand, visual data, such as lane markings or road boundaries, are specifically designed to provide clear visual information to a driver. Hence, using a camera to detect and interpret these cues using machine vision is very straightforward. On the other hand, cameras are today still the most cost-effective tool for such a task, and they are therefore already widely used in commercially available cars delivering vast amounts of data for improvements as a result.

Feature extraction and lane localization is achieved using a variety of well-known techniques. Most algorithms rely on some kind of edge detection (e.g. Canny edge detector), and use powerful filters, such as Hough transforms [95], to detect lines and curves in order to support the edge information and to retrieve knowledge about road curvature which can pose problems [96, p. 16]. Methods solely based on cameras, however lack in adaptive power and can be disturbed by lighting conditions.

Currently, novel methods employing LIDAR technology are being developed to enhance lane detection. In fact, the range information it provides [97, p. 1] and its resilience against low lighting conditions are of vital importance to stabilize lane detection algorithms and obtain satisfactory results. Furthermore, since most LIDAR scanners not only provide spatial information, but also reflection intensities, LIDAR based methods are able to reliably detect lane markings [98] and estimate ground roughness to define road edges [99].

However, when the road surface is covered in snow or no lane markings are present, both visual approaches using cameras and LIDAR scanners fail. To still be able to navigate autonomously lane detection using GPS data in combination with the vehicle's dynamic information in form of the IMU are employed. For this type of lane localization to work, very precise ($< 1\,m$) position data is required. Techniques such as RTK might be able to fulfill this requirement aptly. Moreover, highly detailed map information about the current road environment and condition is of utmost importance to this type of navigation [94, p. 6].

In conclusion, sensor fusion of the above-mentioned sensors will behave in the most stable and reliable way. It can withstand difficult lighting and weather conditions and the GPS and map information will support some level of autonomy even in the most adverse circumstances. An overview of this fusion process for lane detection can be seen in the flow chart of Figure 2.20.

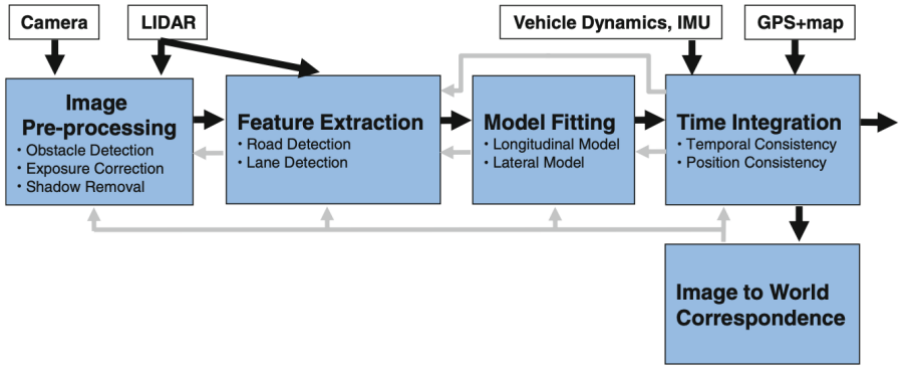

Figure 2.20: Flow chart of a sensor-fusion-based lane detection method [94, p. 7]

2.4.2 Free Space Estimation

In addition to lane detection, which is a crucial task for autonomous or semi-autonomous vehicles, there also exist different or complementary methods for the navigation of AV. One of the used methods is the free space estimation discussed below. As Yao et al. [100, p. 1] put it, it answers the most basic, but also the most pressing question for autonomous driving: what is the drivable free space that can be reached by the automobile by passing obstacles without collision?

Identifying this area usually comes down to an image segmentation task, returning an image with two categories, as can be observed in Figure 2.21: a reachable green road area and an occluded or blocked red area. It originally relied on stochastic or deterministic occupancy grids as introduced in [101], to determine a pixel's vacancy and classify it accordingly.

Whereas most lane detection algorithms provide a notion of where the car sits in the lane and how this lane flows, focusing on the immediate surroundings, free space estimation tries to get a broader overview of the drivable surroundings. As such it can propose new corridors for AV to reach [102] and can achieve an enhanced path planning.

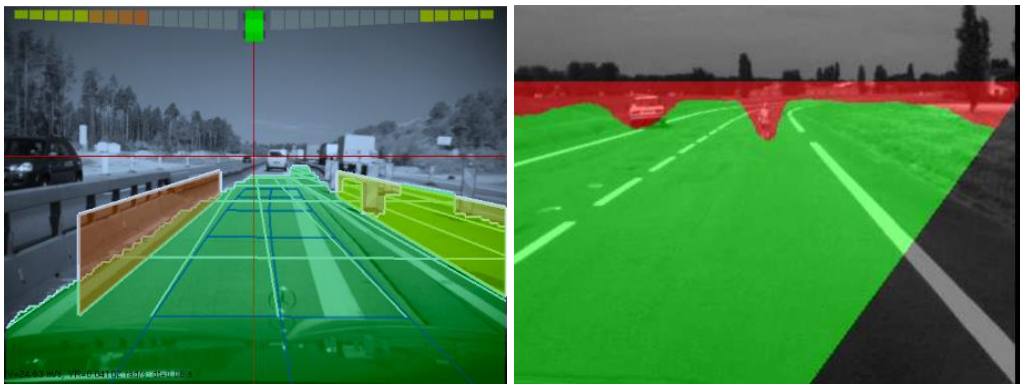

Figure 2.21: Free Space Segmentation: green area represents the drivable area, whereas red denotes obstacles. Left visualization by [103, p. 192] and right is a result of [104, p. 8]

Fast and reliable free space estimation has the potential to critically support ADAS applications such as ACC. Furthermore, since it gives an extensive view and segmentation of the surroundings, it can give AV more flexibility in their navigation in difficult traffic situations to avoid obstacles without collision. Moreover, on-line map services enriched with information about the free space in static and car-free conditions as proposed by Diego et al. [105] hold the capacity to considerably improve an AV's perception of its environment and help to guide it in tough conditions. Their concept relies on manually or semi-automatically segmenting the road in a traffic-free reference video. This reference video is available on-line, where AV can access and overlay them in a real-time fashion on their current recording to mitigate the visibility blocking effect of traffic.

2.5 Related Work

Since the beginning of the 21st century, commercial cars are being equipped with a host of different sensors. This trend grew stronger with the introduction of ADAS being able to partly automate vehicles and taking control of the car. Most driver assistance systems rely on the information provided by various sensors in real-time [106, p. 144], such as cameras, laser scanners, or GPS signals. The availability or reliability of sensor data is however not guaranteed in tricky environmental conditions. This has large implications for road safety in general, as more traffic and/or bad weather drastically increase the crash likelihood [107]. The following chapter provides an overview of work done to analyze how different sensors are affected by traffic density, road geometry, or bad weather conditions.

To ascertain how the performance of optical sensors is corrupted by tough conditions, one straightforward method consists of investigating the visibility or viewing distance and comparing them to favorable settings.

Pomerleau [108] defines an estimated visibility index based on good daytime visibility to investigate the reduction in visibility during nighttime, difficult lighting conditions, and bad weather conditions. The visibility index estimation exploits the RALPH (Rapidly Adapting Lateral Position Handler) system, which is able to find and track arbitrary road features and measuring the attenuation contrast between them. His work proved that indeed visibility of the car's camera during dense fog, morning glare, or during the night was reduced by 70 - 90%. This poses a problem to camera systems in AV since the safe and stable navigation depends strongly on the visibility and on the consistency of the visual data.

Using a stereo camera setup, Pavlic [33, pp. 34–36] demonstrated that the visibility distance of the cameras dropped to about 50 m from the original 250 m in clear weather, when dense fog is present during daytime, but increasingly so during the dawn. Along with drowsiness and speed it is thus no surprise that reduced visibility is one of the main factors driving the increased accident rate and intensity during nighttime, especially when fog is present [109, p. 310]. Hence, Hautière et al. [110] and Tarel et al. [111] have respectively developed methods for better fog detection and importantly for enhancing the quality and contrast of images taken during foggy conditions.

Laser scanners such as LIDAR also suffer from reduced range when confronted with adverse lighting or precipitation. In rainy, foggy, or snowy conditions little water droplets in the air may refract, or totally reflect the laser beams issued by the LIDAR scanner, thus preventing them from reaching real obstacles such as other cars in the distance. In fact, both Kutila et al. [61] and

Hadj-Bachir et al. [112] show that with dense fog or heavy snow, the maximal and mean detection range of LIDAR scanners sharply decreases. This has implications for ADAS applications relying on this data such as ACC, but also on the real-time object detection based on LIDAR, which enables AV to brake in time to avoid obstacles.

The performance of object detection algorithms based on camera images and deep learning techniques also strongly depends on image quality and thus on weather conditions. Michaelis et al. [113, p. 1] showed that corrupted images can lead to a decrease of 30 – 60% in object detection and classification when compared to optimal conditions. They propose a simple data augmentation trick in stylizing training images to increase the algorithms robustness in the wake of corrupted image data.

The free and reachable space is obviously a lot more restricted in dense traffic environments. Hence, the importance for fast and reliable algorithms which allow the AV to navigate safely and flexibly even in these conditions. For the task of free space estimation, a lot of research is focused on using a stereo camera setup and applying stable v- and u-disparity algorithms to detect obstacles in the free space [103], [114], [115]. These algorithms flag pixels of objects that protrude from the (assumed) flat road surface because they have larger disparities. To find the whole free space based on these few earmarked pixels, k-nearest-neighbor (K-NN) and Support Vector Machine (SVM) methods are employed [116, p. 816]. Kemsaram et al. [117] have proposed a very complete approach with an integrated framework of three deep neural networks to combine the tasks of lane detection, obstacle detection, and free space estimation (Figure 2.22). The main conclusion from their work lies in the fact, that indeed a pipeline to simultaneously fulfill the tasks of lane detection, obstacle detection, and free space estimation can be achieved relatively fast (56 ms) on today's computing architectures. However, such latencies are too great for real-time ADAS applications and can thus only be used in low-speed ADAS, as the authors stated.

Figure 2.22: Top: Road and Ego-Lane detection on the left. Obstacle detection and annotation on the right. Bottom: Combination of obstacle detection, road detection, and freely drivable space [117, p. 264]

To overcome the difficulties, the various sensors face, a rich sensor suite with sensor data fusion represents a good solution. It increases the sensor availability and most importantly its reliability, thus enabling a stable autonomous driving process. This work thus focuses on analyzing datasets with a host of optical sensors to combine the approach of free space estimation or road detection with depth information from laser scanners and GPS data to get a more complete impression of various driving environments. The results of this will then serve as a basis to analyze the impact of bad weather or dense traffic conditions on the free space and the maximal distance of it in front of the vehicle.

3 Method and Implementation

To analyze the sensor availability, the free space in front of the vehicle, and detrimental effects on sensor performance, the following implementation has been devised. In a first step, the detailed selection process for datasets to be worked on based on their sensor setup, diversity, and provided information is laid out. Even though this work only focuses on four comprehensive datasets, the nature of the implementation allows for alternative data to be easily included and analyzed. From this starting point, methods for lane and free space or road detection are presented depending on the dataset at hand. Once this free space is detected, the distance calculation relying either on LIDAR, stereo disparity maps, or AI-generated depth maps is displayed. During execution of the algorithm, so-called blocking factors are identified for every frame, yielding a measure of view occlusion.

In a first step of post-processing in datasets where INS information is provided, an investigation of the influence of the road type and location on the free visibility is performed. Furthermore, thanks to the timestamps of the respective frames and the diverse recording conditions of the datasets, the effect of nighttime driving and bad weather conditions is explored.

The post-processing can thus be used to improve map services by adding information about visibility conditions depending on the weather conditions, the time of day, and as a result also on the expected traffic situation at the time of driving. Such richly detailed maps can assist AV in their navigation and path-planning process, when other sensors are limited [118]–[120].

Figure 3.10 provides an overview of the processing pipeline for this work and shows the different steps necessary for this analysis.

3.1 Dataset Selection

To perform a free space estimation, including the crucial depth information and GPS data, the selection of the right datasets is essential. Based on their sensor setup and the post-processing after the capture, the myriad of autonomous driving datasets all provide different data points. For this analysis at least two requirements need to be fulfilled by them. Depth information of some form, must be available. On top of that visual information from driving images is required. Hence, datasets such as Mapillary Vistas are not apt for this task, since it only offers camera data without any depth or distance estimation.

The contribution of semantic segmentations as well as INS data (including GPS signals) is an additional bonus for this analysis, since it enables a detailed understanding of blocking factors and the influence of the vehicle's position on its free space ahead. Furthermore, diverse driving environments during the data acquisition are desirable to analyze the sensor availability in difficult environments.

In the following, the four selected datasets are presented in detail and explanations as to why they were selected are laid out. Table 3.1 provides a summary of the qualities and features, wich explain the choice of these datasets.

3.1.1 A2D2

As presented in chapter 2.3, the A2D2 dataset [10] has been recorded by a research team of Audi in and around the German cities of Gaimersheim, Ingolstadt, and Munich. The carrying vehicle, an Audi Q7 e-tron, was equipped with a complete multimodal sensor suite, comprised of five 360° LIDAR sensors with a range of up to 120 m and an accuracy of ±3 cm, six surround cameras covering the whole surroundings of the vehicle, a GNSS receiver, and an automotive bus gateway. The bus gateway serves as the origin of the global frame of reference and accomplishes two central tasks: it collects all the information about the vehicle's state and driver input and timestamps all the sensor data in UTC (Coordinated Universal Time) format, synchronized to a time master as seen in the hardware setup in Figure 3.1.

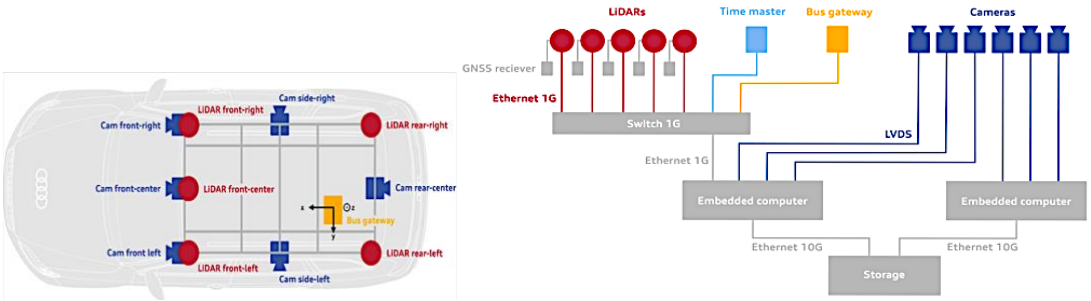

Figure 3.1: Sensor setup from a top view on the left and hardware connections of the different parts of the sensor suite on the right.

Concerning the diversity of recording conditions, the data was recorded during a period in 2018 stretching from June to December. All of the frames were captured during the daytime in mostly sunny or slightly cloudy conditions. During the later phase of the data acquisition in the fall of 2018, some recordings were obtained in rainy or foggy conditions as seen in Figure 3.2. Furthermore, even though no nighttime recordings are available, the dataset exhibits a broad diversity in traffic conditions with frames taken on the highway, country roads, but also in city center of Munich during rush hour (Figure 3.3), which exemplifies highly complex and crowded traffic situations. Before working with the provided images, they need to be undistorted, for which the very comprehensive tutorial [121], offered by the A2D2 team, provides a guide.

Figure 3.2: Diversity of the A2D2 dataset: Rainy and foggy conditions are recorded in addition to clear weather conditions such as in Figure 3.3.

In total, over 390,000 frames with corresponding LIDAR and bus data (including GPS coordinates) were captured, of which nearly 12,500 were annotated with 3D bounding boxes as well as with 2D semantic segmentation – these are the frames this work will focus on. Here lies one of the main benefits of this dataset: not only does it deliver high resolution RGB images and the corresponding depth information thanks to its LIDAR scanners, but it also provides richly detailed semantic masks. This segmentation allows for easy filtration of the free road surface ahead, as well as for classifying LIDAR points to the different semantic classes as displayed in Figure 3.3.

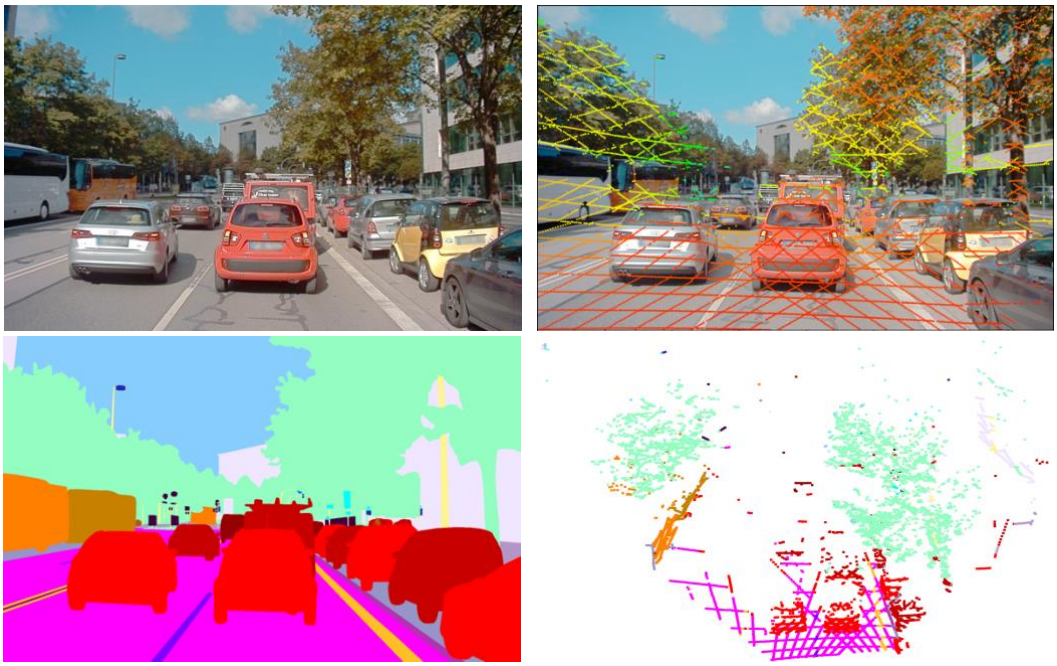

Figure 3.3: Dense traffic situation during the Munich rush hour. Top: RGB image with its corresponding LIDAR point cloud, with the distance of points coding for the color of them in the point cloud. Bottom: Corresponding semantic image. Point cloud in Bird's Eye View with points colored with respect to the semantic class they belong to.

The classification of LIDAR points based on the semantic mask will later on be leveraged to compute the distance of the free road space ahead of the vehicle. Based on this distance information combined with the GPS coordinates and vehicle data, we can infer a measure for the free space and the sensor availability at this point. Similarly, the semantic images will be used to detect and classify different blocking factors and the level of occlusion for each frame.

The LIDAR data is stored in a `.npz` format and can be loaded into Python using the `numpy.load()` function. It is then provided in a dictionary format containing the following data points for every reflection and frame: 3D points in global coordinates, reflectance values, pixel coordinates ('row' and 'col') in the image, which were obtained using Equation 2.1, depth and distance values in m. This format makes the LIDAR data very useable.

The only drawback of this dataset manifests itself in the LIDAR data. In fact, the point cloud shows an elevated density at low range, but this density decreases sharply with increased distance, such that only few or no points are detected with a distance greater than 40 m. Hence, a method for estimating the distance of far-out road pixels, based on their pixel height in the image is introduced.

3.1.2 ApolloScape

The ApolloScape dataset [84] is one of the largest openly available driving datasets there is. The Baidu Inc. research team used two LIDAR scanners with a range of up to 420 m and a precision ranging from 3 to 5 mm, two extremely high resolution (9 MP) stereo cameras, and a GNSS antenna with a position accuracy of 20 to 50 mm mounted on top of a Toyota Land Cruiser carrying vehicle. In total, over 143,000 frames were annotated on a pixel level and showcase the high object density of the dataset: between 12.7 and 38.1 vehicles were detected on average in every frame, compared to 4.1 vehicles per frame in the foundational KITTI dataset.

This is partly explained by the locations of recording. In fact, the ApolloScape dataset was recorded in ten Chinese cities which all experience dense traffic. Concerning the weather and lighting conditions, it shows however only little diversity: nearly all of the frames were collected during the day in clear or lightly overcast weather. Though, for our evaluation, this lack of diversity is no deal breaker in this case, which will become clear in the following.

In fact, since the dataset is so vast and employs a whole sensor suite, it is divided into different subsets: Scene Parsing, Car Instance, Lane Segmentation, Self Localization, Trajectory, Detection/Tracking, Stereo, Inpainting. We focus our work on the Stereo and Scene Parsing subsets, which do not rely on weather sensitive sensors such as LIDAR but provide detailed disparity or depth maps.

ApolloScape Stereo

The stereo part of the ApolloScape dataset provides the following visual data: RGB images of the left and right camera of the stereo setup, background and foreground masks for every scene, as well as crucially ground truth disparity maps for each frame. This “ground truth has been acquired by accumulating 3D point clouds from LIDAR and fitting 3D CAD models to individually moving cars” [124]. The ground truth is available for around 4,200 frames, which were mostly recorded in a period from December 2017 to March 2018.

The disparity maps for this dataset are very detailed and every pixel is assigned a disparity value, as can be observed in Figure 3.4. From this disparity, the real-world distance of each pixel from the vehicle can be easily computed with just a few additional values, namely the baseline and the focal length of the camera system, as is laid out in Equation 2.3.

In combination with a road detection algorithm, presented in Subchapter 3.2, the free road space and the depth of it can thus readily be computed. Figure 3.4 displays a typical frame from this dataset, which is delivered in an undistorted way, in that it captures dense traffic situations with heavy occlusions and changing lighting conditions. These factors all make for a challenging environment for AV, and thus qualify for an analysis in this work.

Furthermore, the foreground masks, which mark all the vehicles in the image as foreground, allow for a computationally cheap method for identifying occlusions and the degree of visibility reduction, due to traffic. The methodology behind this will be discussed in subchapter 3.4.

The only downside of this dataset lies in the fact that, although GPS coordinates were recorded during the data acquisition, they are not provided as part of the stereo subset. However, due to the relative uniformity of the road situations in this dataset, this does not pose an insurmountable problem for this analysis – we qualify most of the driven roads as highways or country roads with relatively wide lanes.

Another motivation, apart from the pixel-level depth annotation, to use the stereo dataset, is to showcase another approach for depth and distance calculation in AV apart from the expensive LIDAR option. Furthermore, it is interesting to analyze the results on a basis of stability, when compared to the LIDAR based methods, especially in challenging environments.

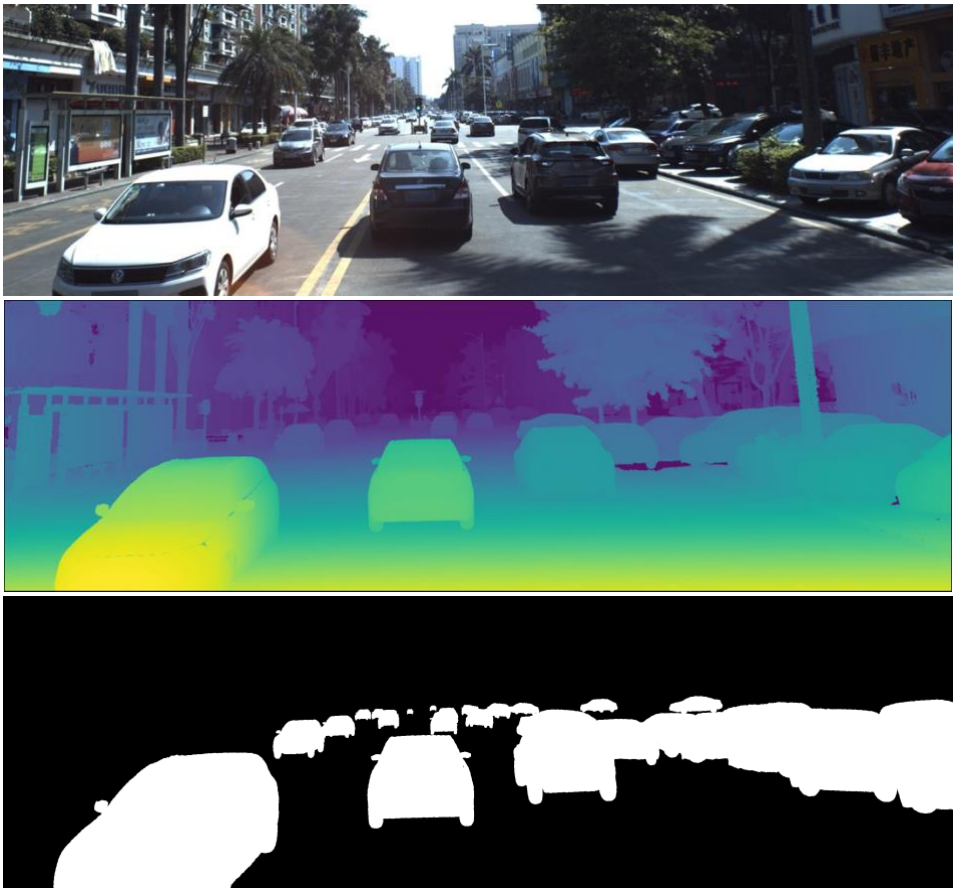

Figure 3.4: RGB image of the ApolloScape Stereo dataset with its corresponding disparity map and foreground mask

ApolloScape Scene Parsing

Another part of the vast ApolloScape dataset is dedicated to a better environmental perception and scene understanding. To that end, around 143,000 undistorted frames, (i.e. the whole dataset) have been annotated on a pixel level [125]. There exist however differences: some of these images have been semantically segmented, meaning every object of the same class is colored in the same way. Others have undergone an instance segmentation, assigning different colors to different instances of objects of the same class. To streamline the analysis with the similar data structure of the A2D2 dataset, this work focuses on the semantically segmented dataset. This will leave over 11,000 frames, of which every fifth is evaluated. This selection is due to the high frequency of image capture and reduces the runtime considerably, without losing much information.

The research team specifies that images were taken at intervals of one meter of the track the vehicle has travelled, yielding a total driving distance of around 140 km, across different Chinese cities. The segmentation method employed, classifies pixels into 25 different categories, such as cars, bicycles, trucks or the road surface. An example of this can be observed in Figure 3.5.

This segmentation, similarly to the A2D2 dataset, allows for a straightforward filtration of the free road surface ahead. Moreover, we can exploit the semantic images to identify and quantify the different blocking factors in a detailed way, which in turn allows us to analyze the level and variance of visibility conditions.

Conveniently, included in the scene parsing subset of ApolloScape are pixel-level depth images for static background at mm-level accuracy. Each pixel of such a depth image is assigned a depth value in an `unsigned short int` format, and an example of it can be observed in Figure 3.5. To obtain the absolute depth D in meters, this value needs to be transformed to a `double` format and divided by 200. This comes down to the following operation, described in Equation 3.1:

$$D = \frac{double(depth)}{200} = \frac{255\ depth}{200} \tag{3.1}$$

where $depth$ is in unsigned short int format.

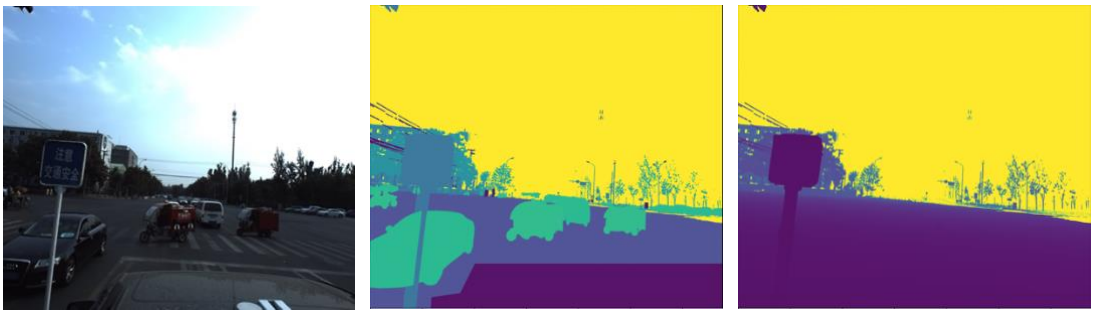

Figure 3.5: RGB image of the Scene Parsing subset, corresponding semantic mask and corresponding depth image.

Thus, estimating the free space, and calculating the depth of free space in front of the vehicle can be achieved in a very precise way. This allows us to draw conclusions on the availability and usability of camera data in dense traffic situations with a high moving object density.

3.1.3 KITTI

The KITTI dataset [66] has been recorded in a cooperation between the Karlsruhe Institute of Technology (KIT) and the Toyota Technical Institute in the months of September and October in 2011. It represents the first multimodal sensor dataset and combines visual camera data with LIDAR point clouds, and INS (GPS+IMU) data, as portrayed in the hardware setup of Figure 3.6.

In total, the raw data is composed of around 25,000 frames captured at a frequency of 1 Hz, and similarly to the ApolloScape dataset it is divided into different subsets: Stereo, Flow, Scene Flow, Depth, Odometry, Object, Tracking, Road, Semantics. However, most of these subsets are very small, including the stereo and semantic parts, with 200 test and 200 training images each. They are therefore not particularly suited for this work but are intended to serve as a basis to develop new software for stereo matching and semantic segmentation algorithms. We thus focus on the raw (synced and rectified) dataset which contains undistorted grayscale and RGB images, 3D Velodyne point clouds with around 100,000 points per frame, INS data, 3D object tracklets and crucially calibration files for coordinate transformation from one sensor to the other. An

exemplary frame, regarding the environment and lighting conditions, including the very dense LIDAR point cloud projected into the image is presented in Figure 3.7.

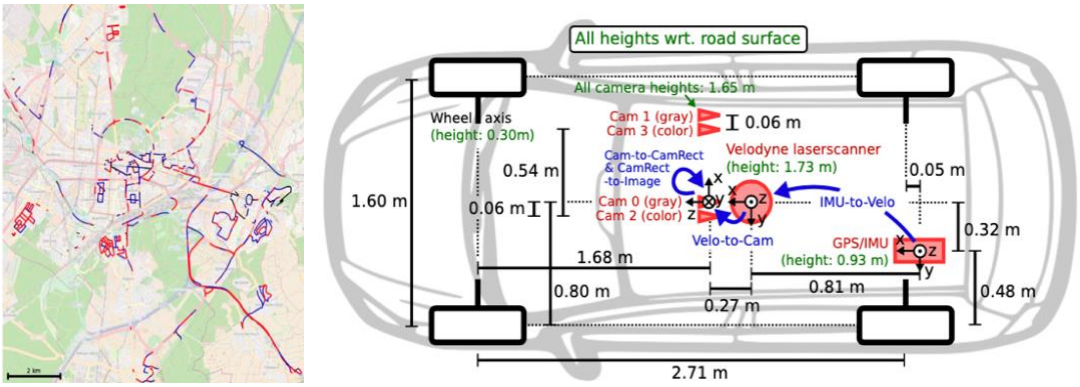

Figure 3.6: Left: Map of Karlsruhe and the travelled roads of the KITTI research vehicle. Right: Sensor setup for the research vehicle with corresponding coordinate systems [66, p. 2].

The LIDAR data in the KITTI dataset comes in a binary `.bin` format and can be loaded into Python using the `numpy.fromfile()` function. It contains only the 3D points in global coordinates, up to a range of 120 m and an accuracy of 2 cm, and the reflectance values of each point in the pointcloud. To streamline the analysis of various datasets, the `prepare_data()` function returns a dictionary with the same content as in the A2D2 case. For that matter, the KITTI team supplies the `project_velo_to_cam2()` function which transforms global coordinates of LIDAR points to camera coordinates of camera 2 based on the calibration files which hold the translation and rotation matrices, as well as the intrinsic and extrinsic parameters of each camera.

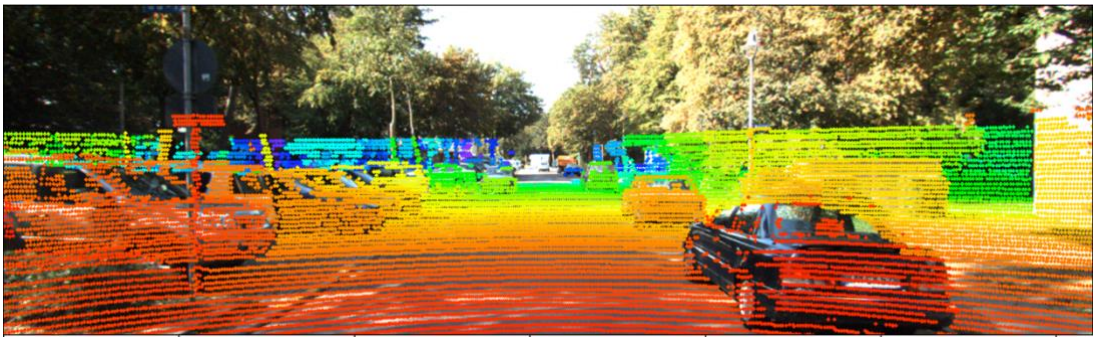

Figure 3.7: Exemplary KITTI frame with projected LIDAR point cloud. The points are color coded based on their distance from the vehicle, red meaning closer and blue meaning farther away.

Concerning the diversity of the dataset, KITTI offers frames which were mostly captured in similar conditions: sunny, relatively low traffic, and in and around the streets of Karlsruhe (Figure 3.6 and Figure 3.7). No nighttime or bad weather instances were recorded. The KITTI dataset is still included in this work as it is the pioneering dataset in this space and represents a benchmark for nearly all autonomous driving datasets today. Thus, it serves as a good basis for comparison.

After the road detection algorithm has been performed, the LIDAR points reflected by the free road surface are filtered out and their distance is calculated based on the LIDAR data. The

LIDAR setup of the KITTI dataset does not suffer as badly from a point cloud density decrease with increasing range as the A2D2 setup.

An OXTS unit is utilized for IMU registration and receiving GPS signals. Thanks to RTK it achieves a resolution of around 2 cm or 0.1°. This precise data is used in the post-processing of this work, to get a measure of visibility based on the different road types.

The KITTI datasets also includes a number of static frames which are dedicated for the detection of persons. These are not relevant for this evaluation and are therefore discarded, resulting in a total of over 23,000 analyzed frames, corresponding to nearly 40 minutes of recordings.

3.1.4 Oxford RobotCar

The Oxford RobotCar dataset [64] was assembled in period of a little over one year from 6 May 2014 to 13 December 2015 and covered over 1,000 km in central Oxford, UK. In fact, this dataset is particular in that it covered the same 10 km stretch of road, displayed in Figure 3.8 in over 100 recordings. The aim of this lies in analyzing the effect different weather and lighting conditions have on the sensor setup of the test vehicle. Since it is one of the largest (20,000,000 video frames, over 23 TB of data in total) and most diverse (driving conditions include large portions of nighttime, rain, and/or snow) datasets there is, it lends itself very well for this work to analyze the optical sensor availability in difficult conditions.

Due to this vast size, a subset of different recordings was selected for this analysis. To that end, recordings showing varying conditions are analyzed to distinguish different factors reducing the availability of optical sensors. For reference some clear daylight subsets are investigated along with recordings in tricky conditions such as dusk, dawn, night, or precipitation. This leaves over 90,000 frames in total, of which every tenth is retained and analyzed because of the high frequency. Hence, little information is lost, but runtime of the algorithm is considerably improved.

Figure 3.8: Top: Traveled route of the research vehicle. Green routes indicate good GPS signal and red marks faulty GPS readings or pick-up. Bottom: Sensor setup of the research vehicle [64, p. 2].

The research vehicle, a Nissan Leaf, was equipped with a full sensor suite composed of six cameras (three individual cameras and a stereo camera with three lenses), two 2D LIDAR scanners, one 3D LIDAR scanner and an INS system for self-localization. An overview of this sensor setup is provided in Figure 3.8 and the development kit for this dataset includes an extensive calibration method.

Each traversal can be downloaded individually and is subdivided into chunks of recordings covering a duration of around six minutes. For this work, only the images from the center of the stereo camera (Bumblebee XB3) and the LIDAR scans of the 3D LIDAR are used. These images were captured at a frequency of 16 Hz and are stored as GBRG Bayer images and can be demosaiced to RGB images using the `load_image()` function.

Thanks to the provided labels, indicating driving conditions such as rainy, sun, or poor GPS, one can mainly focus on comparing different parts of the whole dataset under varying conditions. As such we utilize the LIDAR data in combination with a road detection algorithm to estimate the free space ahead of the car. An exemplary image with its corresponding pointcloud generated by the `project_laser_into_camera()` function of the development kit can be observed in Figure 3.9. Of note is that this LIDAR scanner does not suffer too strong from reduced density at elevated distances, which facilitates the distance calculation at long range.

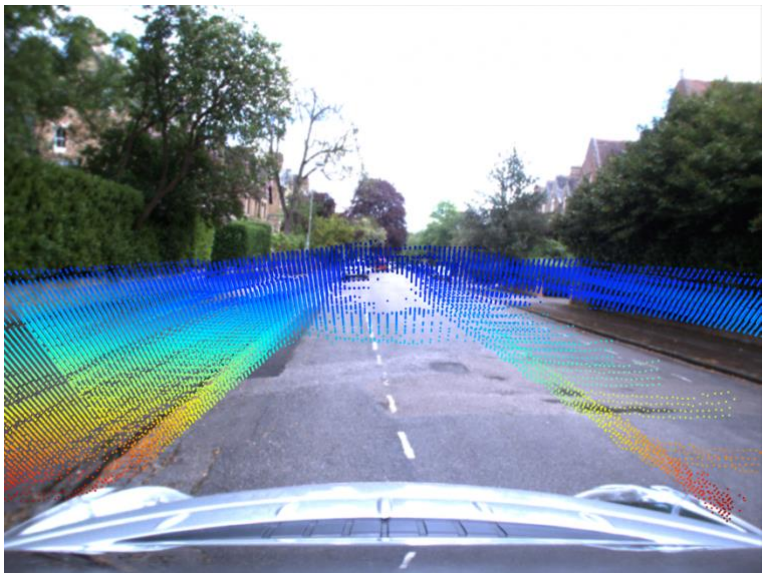

Figure 3.9: Exemplary RobotCar frame with projected LIDAR point cloud. The points are color coded based on their distance from the vehicle, red meaning closer and blue meaning farther away.

GPS coordinates, along with vehicle state data such as speed, acceleration, or roll, are provided for every frame and allow a detailed analysis of free space depending on the vehicle state and position.

Table 3.1: Overview of the selected datasets for this analysis

Name	Depth Information	Semantic Images	Advantages	Disadvantages
A2D2	LIDAR	Yes, 38 classes	Semantic images	LIDAR range
ApolloScape Stereo	Disparity Maps	Yes, foreground and background	Disparity maps	No GPS
ApolloScape Scene Parsing	Depth Maps	Yes, 24 classes	Semantic images, depth maps	No GPS
KITTI	LIDAR	No	Good benchmark	Small LIDAR range
Oxford RobotCar	LIDAR	No	Large, diverse weather	Same driving route

3.2 Road Detection

Based on the data each analyzed dataset offers, different approaches for the (free) road detection ahead of the vehicle are required. We differentiate methods relying purely on image data, on image and LIDAR data, and on semantic images, to cater to the details of every considered dataset.

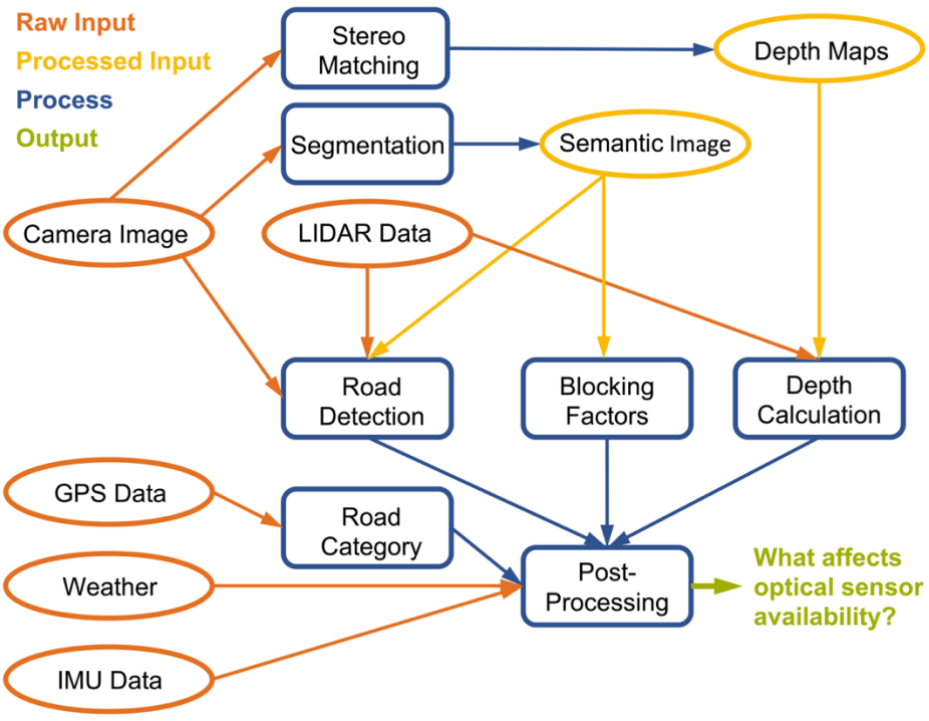

Figure 3.10: Pipeline of the devised method.

3.2.1 Road Detection Algorithm

Road Detection based on Image Data only

Thanks to the developments of CV in recent years, the feature extraction from images has been considerably improved. Hence, crucial tasks to autonomy such as road detection based on high resolution video frames have become feasible in real-time – a feature which is used in a lot of ADAS (e.g. lane departure warning, ACC) in modern cars [126], [127].

The approach presented in the following is an adaptation of [128], and focuses on image data only. In a first step, a ROI is defined, to speed up the process and reduce unnecessary computations. In fact, for this analysis the definition of a rectangular ROI (Figure 3.11) reduces the number of pixels by nearly 80% and thus significantly speeds up the subsequent algorithm, while keeping the road area in the picture.

Figure 3.11: Original frames at the left and their corresponding ROI from KITTI (top) and Apolloscape Stereo (bottom) at the right.

In a next step, different algorithms are applied to this ROI. First, the ROI undergoes Gaussian blurring as described in Eq. 2.4 to smoothen the image and reduce the noise. Depending on the dataset, the image resolution changes, which is why variable kernel sizes are used. Consequently, the higher the resolution, the larger the kernel size should be. Hence for the 4 MP images of the ApolloScape Stereo dataset, a kernel size of 9×9 delivers satisfactory smoothing, whereas KITTI and Oxford RobotCar images, with a resolution of 0.5 MP and 1.2 MP respectively, are sufficiently denoised when using a kernel size of 5×5.

Subsequently, a Canny edge detector is employed to return a binary image containing the meaningful edges in the image. Since the image has been cropped to the ROI, these mainly include road lane markings and road/surrounding edges, which can be leveraged for the road detection (Figure 3.12).

To stabilize the algorithm, road markings are also retrieved by making use of color filtration. To this end, since most lane markings are either yellow or white, the algorithm focuses on these two colors. Even though it is possible to filter out colors in the RGB color space, it is not efficient since one has to take into account the three color channels and compare each of them. Therefore, the ROI is transformed from the RGB color space into the HSV or HSL color space. The hue channel can thus be used as a single parameter for filtering out colors such as yellow. Moreover,

the white color can be easily filtered out based on the lightness and saturation values. In fact, considering Figure 2.10 white is characterized by high lightness and relatively low saturation. The results of the preprocessing for the road detection can be observed in Figure 3.12.

Figure 3.12: From left to right: Smoothed ROI, Canny edge image, Binary ROI after filtration of white color

To conclude the road detection and obtain an image region which corresponds to the road, the different preprocessed ROI are combined. First, a corridor is defined. It tapers from the bottom of the ROI, where the road in the image is wider, to the top, where the road narrows down in the image plane. The results of the color filtration and the canny image are overlaid using the `logic_and()` function to identify common features inside this corridor, which serves as a basis for the detection algorithm. The corridor is then traversed from top to bottom and from the middle to the sides to find the first detected feature and mark it as lane contours. The finished results of this algorithm can be observed in Figure 3.13, where the original frame is overlaid with the ROI and the detected road in violet. This method is used for the ApolloScape Stereo dataset which only offers (stereo) camera data, and no segmentation or LIDAR information.

Figure 3.13: Exemplary results of the road detection algorithm based on image data only. These images have been produced by the `overlay_road_on_image()` function (see Appendix).

The presented algorithm does however have its limits. Even though it is very robust at close range, as can be observed in the presented results (Figure 3.13), the quality of the detection deteriorates at larger distances. This is mainly due to two factors: curvature and contrast attenuation. Due to the strict definition of the corridor, the algorithm is not well suited to factor in large curvatures of the road. While this does not affect its performance at close range, it does at greater distances, where some roads show curvature or at intersections where roads make 90 degree turns. Furthermore, since the Canny algorithm relies on the contrast between

neighboring pixels, the contrast attenuation at greater distance [108, p. 908] hampers its performance and thus also the road detection algorithm presented above.

Thus, for datasets where LIDAR point clouds are available, image data is combined with LIDAR data to make the algorithm more robust and reduce faulty detections at large distances or large curvatures. This approach is laid out in the following subchapter.

Road Detection Based on a Combination of Image and LIDAR Data

Since the position of the relevant LIDAR scanner in the global coordinate system is known, this information can be leveraged to filter out only those points which were reflected from the road surface. The concept behind this method is the flat road assumption, which is widely used for free space and obstacle detection [99, p. 2], [129, pp. 1343–1345]. It assumes that most of the freely drivable space in front of the vehicle is flat, thus LIDAR points with a global position above the road either belong to moving obstacles or are an indication that the road bends and static objects appear in front of the vehicle.

A straightforward realization of this concept consists in filtering out all the LIDAR points which were reflected from objects at an elevation of more than 50 cm from the road. Combining this approach with the method discussed for camera data only comes down to retaining only those LIDAR points at low elevation (flat road surface) and inside the corridor as defined above. This yields more stable results, as observed in Figure 3.14. Another advantage of this approach can be found in the fact that instead of only the pixel coordinates of the road surface, one immediately also obtains the necessary distance information of these points.

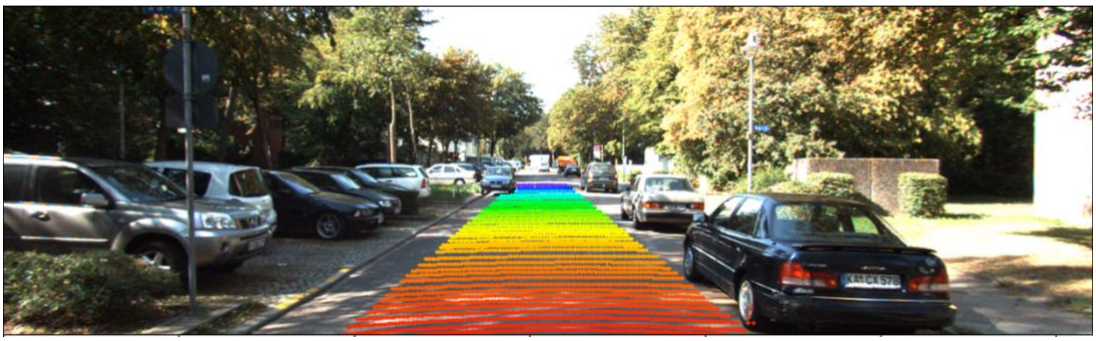

Figure 3.14: Result of the combined road detection approach relying on camera and LIDAR data. For reference, Figure 3.7 shows the identical frame, but with the complete pointcloud.

This combined approach is used in datasets where camera and LIDAR hardware is present, specifically these are the KITTI and Oxford RobotCar datasets.

3.2.2 Road Detection Based on Semantic Images

In the cases of the A2D2 and ApolloScape Scene Parsing datasets, the road or free space detection does not require a dedicated road detection algorithm, such as the one described above. The advantage of these datasets lies in the contribution of semantic images provided for every frame in the dataset. Every pixel of these semantically segmented pictures is thus attributed to a specific class and colored accordingly.

Free road detection thus comes down to filtering out all the road pixels based on their color which can be achieved with the `filter_road()` function, which takes a semantic image of the scene and returns a binary image of the road. The pixel coordinates (u, v) of the road pixels can then be obtained by the `get_road_pixels()` function which takes in the binary image of the filtered out road. The binary image is generated by retaining the road pixels of the semantic image based on their color and coloring them in white (or 255) while the rest of the image is turned black (or 0). The result of this operation can be observed in Figure 3.15 for exemplary frames of the A2D2 and ApolloScape Scene Parsing datasets.

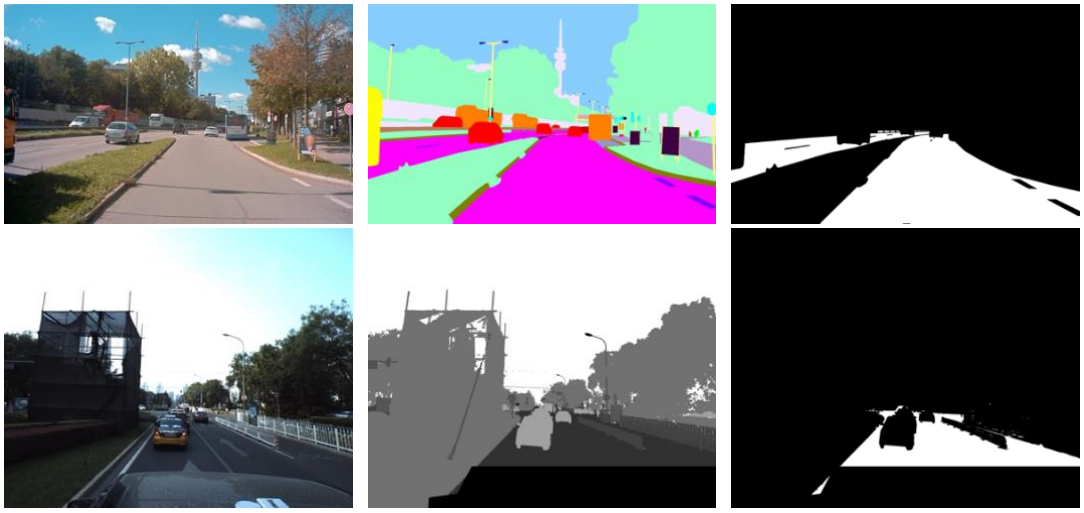

Figure 3.15: Binary road images based on the semantic masks provided in the A2D2 (top) and ApolloScape Scene Parsing (bottom) datasets.

3.3 Distance Calculation

Now that the free space ahead of the car has been identified thanks to the road detection or filtration algorithms, the next step consists in getting a measure for the depth or distance of this free space. Getting a quantitative measure for the maximal distance of the freely reachable space is of high importance for AV. In fact, the autonomous navigation process is not only guided by lane detection, self-localization, and mapping techniques, but it also depends on the free distance ahead. Information about this can help the AV to adapt its speed, warn the driver, or alert other drivers thanks to updated online map services.

Furthermore, for datasets where LIDAR scanners are employed to get distance information about the surroundings of the car, a depth calculation and analysis of the point clouds can offer a measure for the sensor availability in bad weather or lighting conditions. The depth of the free space of the car, also provides information about the level of occlusions and visibility blockage, something discussed later in this work.

3.3.1 Datasets with LIDAR

In this analysis, we focus on datasets providing some depth or distance information for every frame. Methods employing LIDAR scanners are very popular among research teams, because

of their high precision and range. In this case, the datasets using LIDAR scanners are the ones from Audi (A2D2), KITTI, and Oxford (RobotCar).

The method to get a distance measure for the free road pixels, which we identified in the previous step, is straightforward when using LIDAR scanners. The point clouds are mapped from their global 3D coordinates in meters to image coordinates in pixels using the intrinsic and extrinsic parameters of the camera system as described in Eq. 2.1. The points reflected from identified road pixels are filtered out and their distance in meters is read out.

Thanks to its relatively unrestricted range, this already represents the last step for the Oxford RobotCar frames. However, as observed in Figure 3.3, the point cloud captured by the LIDAR of the A2D2 research vehicle experiences a sharp decrease in density at an increased distance. Furthermore, the KITTI LIDAR scanner also mostly captures points at relatively close range of around 30 - 40 m. Hence, a large number of identified road pixels are not considered in the distance calculation as described above in this case.

Furthermore, the inverse projection, meaning the calculation of global coordinates in meters from pixel values, is a very hard problem which can analytically only be solved when each pixel contains a depth value as is the case in RGBD (RGB and Depth) images. In fact, deep learning methods to tackle this notoriously hard problem exist [130]–[132], but they require a lot of training and are therefore not considered in this work.

To still acquire accurate distance estimates for every free road pixel in the segmented frames, one can leverage the relationship between the vertical pixel coordinate and the real-world distance. Hence, the introduction of a regression model based on the 3D distance of the retained LIDAR points which were reflected from the free road surface and their corresponding vertical pixel coordinate. With this model we can henceforth predict the real-world distance of identified road pixels as a function of their pixel height.

This regression method performs well, when comparing it to known benchmark distances such as the spacing between lane markings on German highways. Since the regression only takes in the pixel height as a predictor, as most roads are relatively straight and follow the form of a tapering corridor, the model also holds well for scenarios such as in the left part of Figure 3.16. However, the regression performance deteriorates rapidly when only few LIDAR points are present, when the road twists sharply, or in situations where a large part of the image width is covered by road as is the case on intersections. Frames with only few LIDAR points and sparse point clouds and with large horizontal sections of free road surface, where the regression model fails, are displayed on the middle and right images of Figure 3.16.

Figure 3.16: Frames from the A2D2 dataset. Left: Regression holds well. Middle and right: regression model fails.

Furthermore, to get a measure on the robustness of this method and generally to obtain more stable and comparable results, different percentiles of the maximally detected or predicted distances are selected. This shields the algorithm against eventual large outliers and allows

meaningful comparisons between the different datasets, thus permitting conclusions about the effect of the environment in which the data was captured on the free distance.

3.3.2 Datasets with Depth or Disparity Maps

In instances where the dataset includes annotated depth or disparity maps, as is the case for the ApolloScape datasets, depth calculation is straightforward. First the road must be detected or filtered as described in subchapter 3.2. Afterwards, the depth of each identified road pixel is either directly read out as described in Equation 3.1 if the depth information is stored in a depth map or calculated as described in Equation 2.3 if a disparity map is considered.

Again, different percentiles of the maximally detected depths are stored for better comparisons and to get a measure of the level of occlusion in each frame.

3.4 Blocking Factors

Obtaining a measure for the level of occlusion in real-world scenarios the considered datasets display is important for the development and application of autonomous navigation. It can be used in real-time when camera images and other optical sensors indicate to the vehicle that the visibility ahead is low due to other traffic or static objects, which in turn can trigger a warning to the passengers or driver. Alerting the driver and the AV about a high level of occlusion can thus be critical to safety.

The method this work focuses on is centered around a different approach. The goal lies in updating online map services from which the AV can retrieve information on expected traffic or obstruction while driving. Based on the time of day (e.g. rush hour) and the position and type of street (e.g. city street or motorway) different levels of occlusion are observed. When map services are enriched with such information, they can become a powerful tool for AV to rely on in dense traffic situations or bad weather, either by adjusting the driving style or by notifying the driver to take over.

For identifying levels of visibility obstruction, the devised method makes use of the different semantic images offered in the A2D2 and ApolloScape datasets. The relevant blocking factors identified in this analysis are cars, trucks and buses, and pedestrians and bikes. Furthermore, frames where other obstructions significantly block the visibility are also considered. In a first step, the relevant blocking factors are filtered out based on their semantic class color and a binary image is produced (Figure 3.17 and Figure 2.13).

Once the binary image is generated, the proportion of theoretical road pixels obstructed by blocking factors is calculated. To this end, the semantic image is traversed from the bottom to the top and from the left road edge to the right road edge. Once the proportion of car, truck, or pedestrian pixels with relation to the road width in the considered pixel row exceeds 50%, the algorithm stops, marks the frame as blocked, gives the category of the blocking factor, and also returns the row (and column) in which this blocking factor was detected (see `blocking()` function in Appendix).

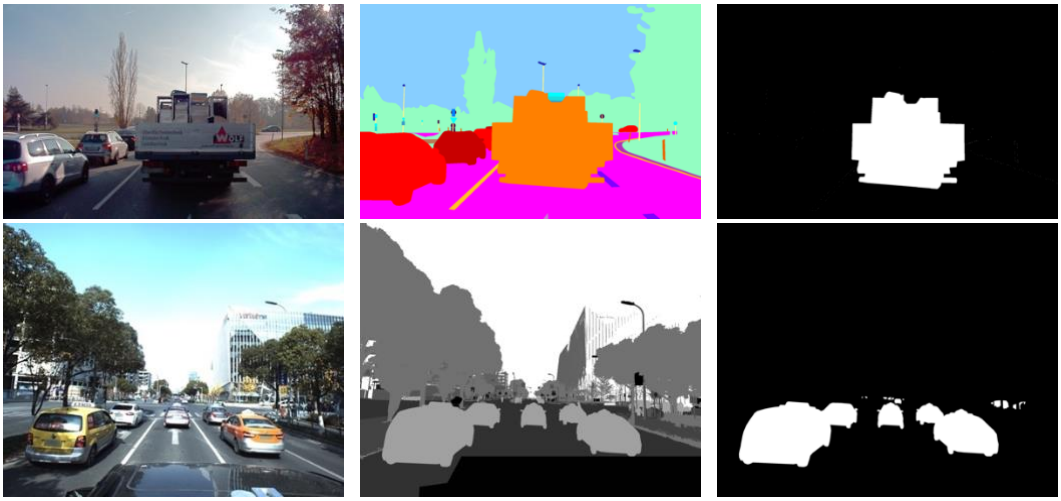

Figure 3.17: Exemplary frames qualified as “blocked”. On top the visibility is blocked by a truck, whereas the visibility in the bottom frame is obstructed by cars.

The algorithm can stop after the first blocked row, since the blocking factors in most cases take up a large proportion of the image above this first row, as can be seen in Figure 3.17. Depending on the row or pixel height at which the occlusion has been located a level of obstruction is defined. The lower the row, where an obstruction is localized, the higher the degree of obstruction. This work differentiates between three levels of visibility obstruction:

- Low: most of the space ahead of the vehicle is free, but the maximal distance is not detected, since a blocking factor occurs at relatively long range. ADAS technologies such as ACC are alerted, but no direct breaking action is required.
- Medium: Visibility is blocked at relatively close range already. If the ego-vehicle is still moving, breaking action is required. The bottom frame of Figure 3.17 is exemplary for this case.
- High: There is close to no free visibility and space directly ahead of the car and the optical sensors of the ego-vehicle are highly constrained. The upper frame of Figure 3.17 exemplifies such a situation.

Depending, if the frame is indeed obstructed or not, the distance up to the blocking factor is retained or not. For the calculation of this distance, the regression method is used with the row of the obstruction as input in the case of the A2D2 dataset. In the setting of the ApolloScape semantic dataset the row of the obstruction as well as the middle of the road are given as input for the depth image and the depth at these coordinates is preserved.

Information about blocking factors and the visibility obstructions they cause is used in the post-processing step to analyze to what degree optical sensors are constrained by them and also to estimate the proportion of frames where the AV then can no longer rely on its LIDAR or camera data, since they can only deliver a low degree of information.

3.5 Post Processing

The final step of the analysis consists in identifying the meaningful factors affecting optical sensor availability and as a result the free space and free viewing distance ahead of the vehicle. To this end, different datapoints provided with the visual image and LIDAR data are investigated. Specifically, the points in time, the weather conditions and crucially also the GPS position are investigated.

Consequently, in a first stage this information needs to be extracted and attributed to the corresponding frame. During the data acquisition, the INS (IMU and GPS) timestamps and logs its data into an on-board storage unit along with the images and/or LIDAR point clouds. The formats `(.json`, `.csv` or `.txt` files) and parameters which are saved vary for each dataset. Common parameters include the current vehicle speed, acceleration, roll, pitch, yaw, the timestamp (in UNIX, UTC, or TAI format) as well as GPS coordinates (longitude and latitude) in degrees. To access the files in which this data is stored and extract the correct parameters for every frame the `get_metaparameters()` function can be used. Additionally, the Oxford RobotCar dataset also provides tags characterizing the conditions during recording such as “night”, “sun”, “rain”, “dusk” or “snow”.

The focus in this analysis lies on the point in time, the GPS coordinates, and the weather conditions. Hence, the time is converted from its encoded format into a 24 hour time of day format: hh:mm:ss. Frames which were recorded during the rush hour, meaning between 7:00 and 9:00 or between 16:00 and 18:00 are flagged as taken during times of elevated traffic. This allows for a comparison of the depth of free space and thus sensor availability and reliability for dense traffic scenarios and normal, relatively low traffic.

GPS coordinates are in turn used for two purposes. First of all, they can be employed to enrich map services with traffic and visibility information. Additionally, they are leveraged to estimate the effect of the type of road on free space. Specifically, they serve as the input for an online tool developed by the Chair of Automotive Engineering at the Faculty of Mechanical Engineering at TUM which returns the type of road for GPS coordinates in the whole of Europe. The `get_road_category()` function then accesses the API out of the program and extracts this information for every frame [133].

This information is retrieved from the OpenStreetMap (OSM) database [134]. It classifies GPS coordinates into different categories such as ‘highway’, ‘amenity’, ‘historic’, or ‘cycleway’. In this analysis we only focus on frames which were correctly localized on a ‘highway’. This road category is subdivided further into road types upon which an investigation of the free space is performed. This is aimed at answering the question of how optical sensors are affected by different road types such as ‘motorways’, ‘trunk roads’, ‘primary roads, or ‘secondary roads’ for example.

Another interesting examination is the effect that lighting and weather conditions have on the optical sensor availability. For that matter, the high diversity in environmental conditions ranging from snow, rain, and dense fog to nighttime and dusk driving in the A2D2, ApolloScape, and Oxford RobotCar datasets provides a good basis for further investigation. Especially the LIDAR based datasets (A2D2 and RobotCar) can be sensitive to precipitation, thus reducing the availability and reliability of the LIDAR scanner for autonomous navigation.

4 Results

The following chapter presents the results obtained when applying the implementation laid out in Chapter 3 on the considered datasets. Estimations for the free space and the free distance detected by the optical sensor suite are explored for each frame of the datasets. In a first stage the raw results of how far optical sensors can reliably perceive their surroundings are presented and compared for each dataset. In a following step the results are sliced based on different factors to analyze their influence on sensor performance and general visibility. These factors include traffic and weather conditions as well as the road type as explained in Subchapter 3.5.

These breakdowns of the results help distinguish the relevant influences on optical sensors and lay open the particularities of the different datasets. This in turn can be leveraged to evaluate the various sensor setups and establish a real-world comparison between their performance and shortcomings.

4.1 Raw Results

In this first section of the chapter presenting the results, raw and unrefined findings are shown and explained. To this end, the free distances of the different datasets are compared. Furthermore, the sensitivities and drop-offs from maximally detected distances to various percentiles of these detections are analyzed and a choice for which measure to use across the board is justified.

The raw analysis of the datasets concludes in the formation of dataframes containing all the relevant information which was available by the proposed data points. Depending on the latter, the final results include free road distances, the timestamps of each analyzed frame, relevant blocking factors, as well as GPS coordinates. Based on this raw information, the initial step of the post-processing consists in finding out the OSM road categories detailing where the data was recorded. Since this analysis focuses on real-world scenarios, where most of the driving is done on public roads, every frame that has not been located to be on a “highway” category (OSM) is discarded.

However, due to minor GPS errors, or the unavailability of GPS coordinates in some recordings because the signals of some satellites did not reach the GNSS antenna of the vehicle, a considerable part of frames were mistakenly disregarded. Nevertheless, this guarantees a good basis for comparison between the datasets by discarding unrepresentative frames. Subsequently, the OSM road type is determined for an ensuing investigation of the influence that road types and geometries have on the free space ahead of vehicles, which will be discussed in Subchapter 4.2.3.

Furthermore, the timestamp in TAI, UNIX, or UTC format is converted to a datetime format and added to the dataframe for two reasons: identifying and marking frames that were recorded

during the rush hour with increased traffic and marking frames that were acquired during difficult lighting conditions such as night, dusk, or dawn or in difficult weather conditions which are more likely to appear in autumn and winter months. This categorization will also be used in the following subchapters to investigate the sensitivity and availability of optical sensors in complicated visibility conditions.

Free Distance

Different levels of free distances are stored in the final dataframes to analyze the sensitivity of distance measurements and determine a good basis for comparison between the different datasets. Hence, percentiles ranging in increments of 5% from 75 to the maximum free detected road distance are retained. The identical approach is taken with the calculated free distances in the cases of A2D2 and KITTI.

As can be observed in Figure 4.1, a sharp drop-off in detected distance occurs when comparing the maximum with the lower percentiles. This considerable decrease is largely due to outlier points detected by the LIDAR scanner of the A2D2 research vehicle, which is relatively constrained in its range as can be seen in the plot displaying the 99th percentile detected distance (Figure 4.1).

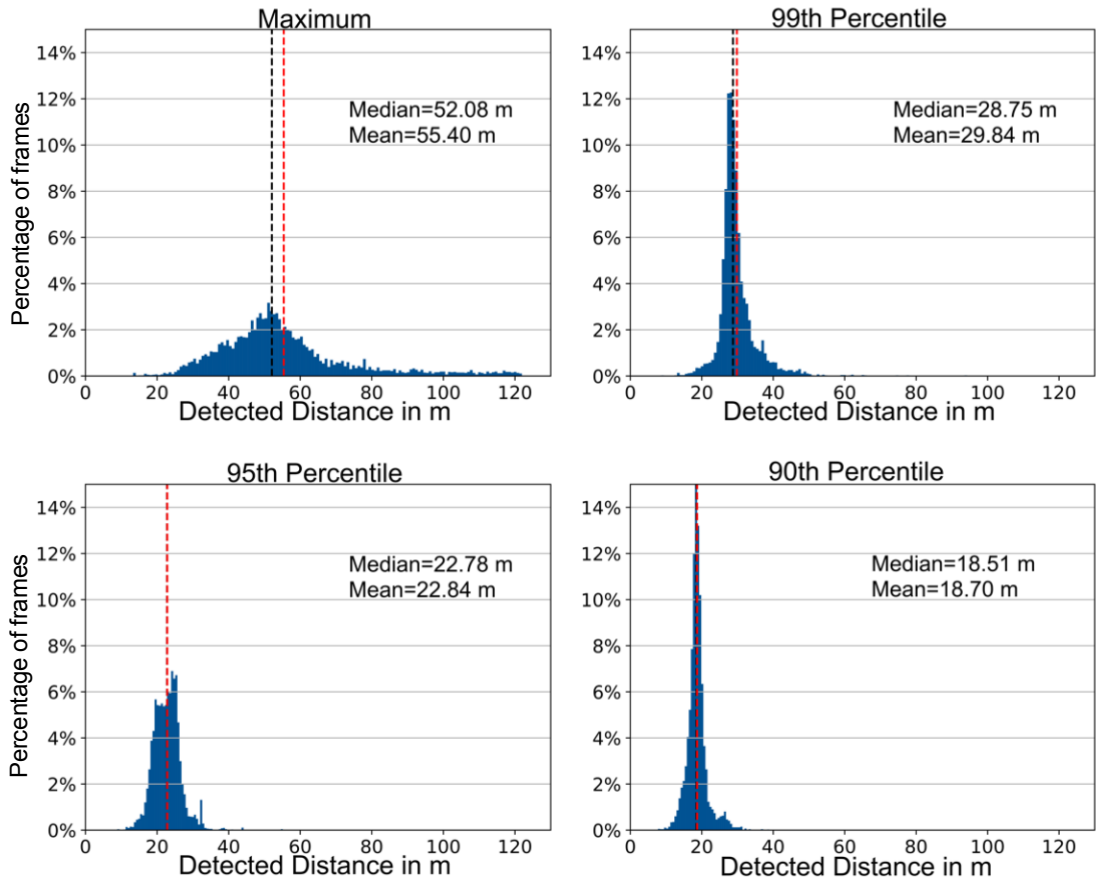

Figure 4.1: Drop-off from maximal distance to lower percentiles of detected free distances for the A2D2 dataset.

Figure 4.1 showcases that the maximally detected road distance by the LIDAR sensor employed in the A2D2 is caused by very few outliers, less than 1% of the pointcloud in fact. This is

demonstrated in the fact that the median of the 99th percentile of detected distances is nearly 45% lower than the median of the maximal distance. Furthermore, a greater concentration of detected free distances around the 20 to 25 m marks is observable for the 99th, 95th, and 90th percentiles. The nature of these outliers also means high fluctuations independent of the actual free visibility and thus the maximally detected free distance is not particularly well suited for comparison, because it is largely dependent on the type of LIDAR employed.

Hence, one notices a maximal range of 120 m in the case of the A2D2 LIDAR, whereas the one employed on the KITTI research vehicle reaches a real-world range of only 80 m (Figure 4.2). In fact, a similar pattern emerges for the KITTI dataset. However, the drop-off between the maximally detected distance and the 99th, 95th, and 90th percentiles is not as pronounced.

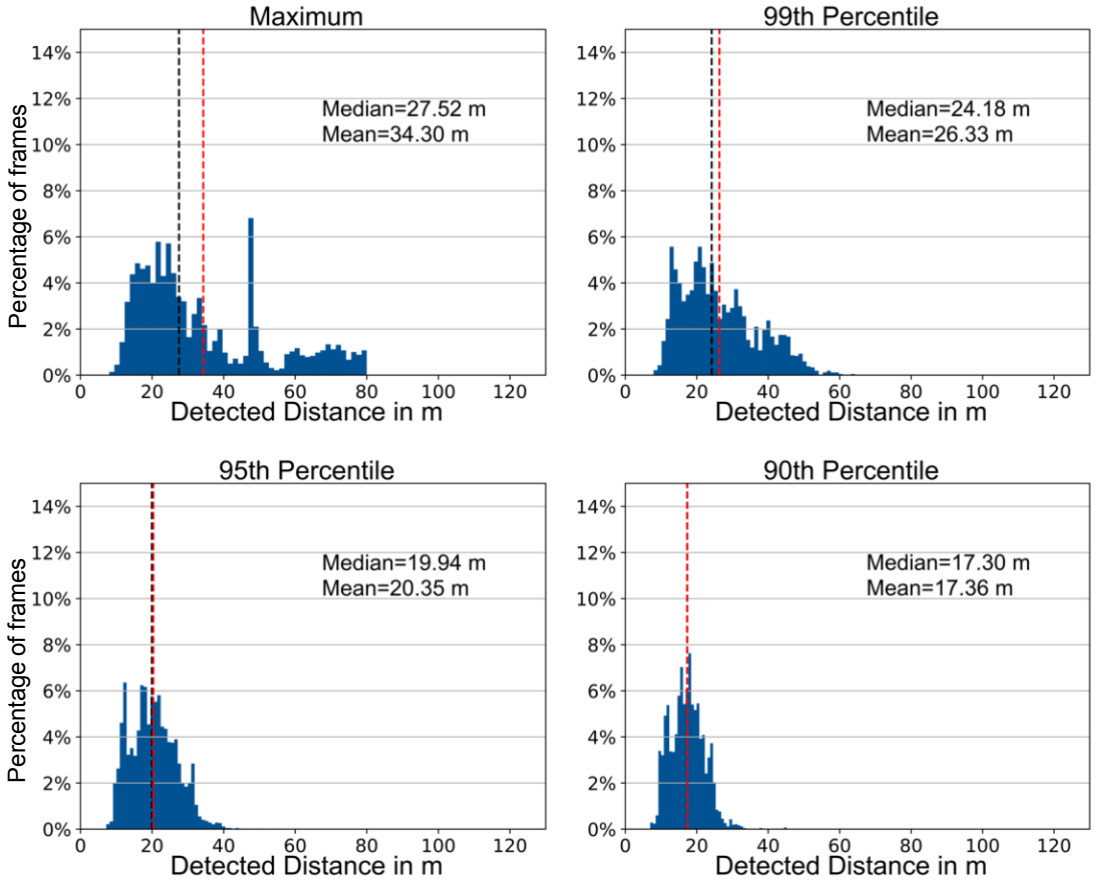

Figure 4.2: Drop-off from maximal distance to lower percentiles of detected free distances for the KITTI dataset.

This phenomenon is due to the greater pointcloud density in case of the KITTI LIDAR scanner, when compared to the one employed in A2D2 (Figure 3.3 and Figure 3.7). Moreover, this increased point cloud density explains why the detected free distances do not show a similarly high concentration around the median values as observed in the A2D2 dataset, but rather a broader distribution around the median.

However as emphasized in Chapter 3, especially in Subchapter 3.3, both the LIDAR sensors employed for the A2D2 and KITTI data acquisition are - apart from their outliers at the maximum - relatively restrained in their range and leave large parts of free road ahead undetected. This is why, to get a reasonable basis for comparison across all datasets, this work resorts to using the calculated unobstructed distances for these two data sources, which will be presented in the

following. The calculated free distances are obtained by predicting the distance of the highest (in pixel coordinates) free road pixels, based on a regression model between the distances and vertical pixel coordinates of LIDAR points.

First, the filtering out of the few frames where the regression failed, meaning it predicted negative or inexplicably large free distances, is performed to get rid of wrongfully distorted results. Such failures occur in settings of very sparse point clouds or when the road does not follow a relatively straight line. Furthermore, to get more stable distributions different percentiles of the calculated distances from the 75th up to the maximum are collected. The subsequent results are found for the A2D2 (Figure 4.3) and KITTI (Figure 4.4) datasets.

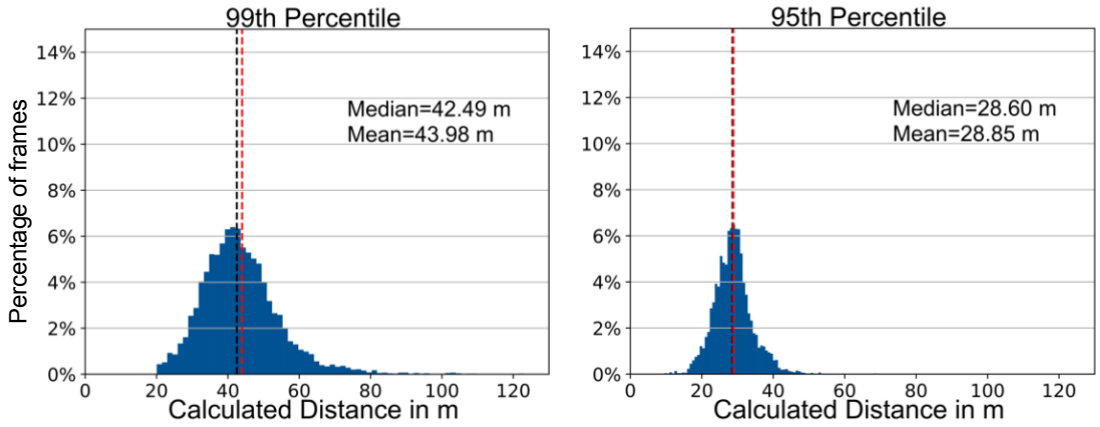

Figure 4.3: Calculated free distances for the A2D2 dataset.

When comparing the 99th percentile of the calculated free distance (Figure 4.3) with the 99th percentile of the detected free distance (Figure 4.1) we observe a large increase in the median free distance. This goes to show, that indeed the LIDAR scanner used for the data acquisition in the A2D2 dataset is constrained and leaves large parts of free road undetected. In fact, a considerable increase of 13.74 m is observable in the median free distance. A similarly large increase is observed for the calculated distance in the KITTI dataset.

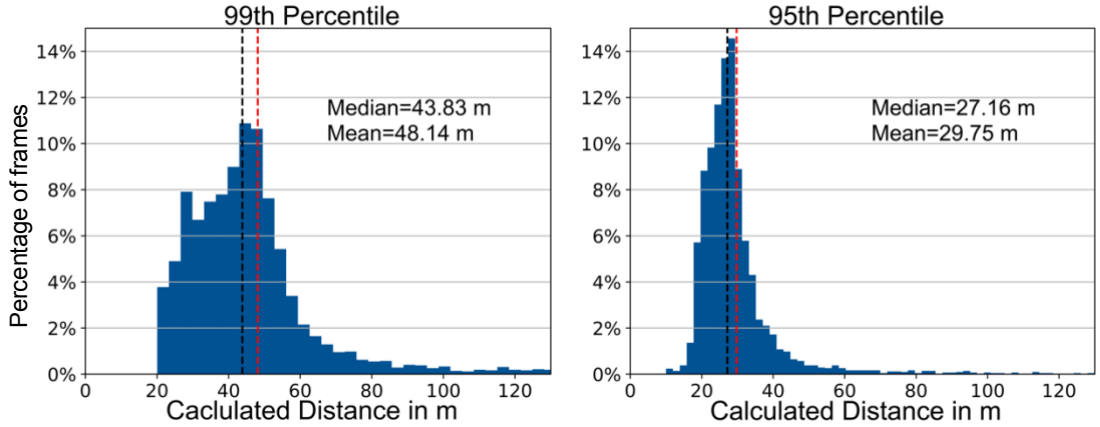

Figure 4.4: Calculated free distances for the KITTI dataset.

Again, one can observe a large increase in free distance when comparing the calculated distance with the detected distance in the case of KITTI. As such, the LIDAR left 19.65 m of freely accessible space in front of the vehicle undetected, when comparing the 99th percentiles of the

calculated and detected distances for KITTI. Furthermore, Figure 4.4 proves that the road detection algorithm works well for the frames of KITTI. Since the A2D2 and KITTI datasets have been recorded in similar driving environments (in and around larger German cities, mostly in good weather conditions), the free road distances should be comparable. Since A2D2 provides detailed semantic images, it serves as a precise benchmark for the distance calculation using the regression model. Hence, obtaining similar results for KITTI where the road has been detected by a dedicated algorithm, confirms the good performance of the road detection algorithm.

Thus, moving on, for KITTI and A2D2, only the 99th percentile of the calculated distance is used. This not only guarantees that the maximum of free space is indeed detected, but also ensures a meaningful comparison between the different datasets. Moreover, the lower percentiles are not considered in the analysis. Even though they serve to control that the algorithm works correctly and in a predictable way, they do not offer a lot of information, since the free distances closely ahead of the vehicle are similar for most frames.

The final LIDAR based dataset is the Oxford RobotCar dataset. It is particular, compared to A2D2 and KITTI, in the fact that its LIDAR setup is not restricted in terms of range. Hence, no distance calculation based on the detected road surface is needed and the analysis is performed on the detected distances.

In fact, even though the median of maximally detected distances over the different frames is not significantly higher in the RobotCar dataset, when compared to A2D2, observing the lower percentiles (e.g. 99th) one notices that no sharp drop-off occurs (Figure 4.5). This is partly due to the higher point cloud density but also indicates that this LIDAR sensor reliably detects points at a large distance, thus ruling out that the furthest detected free distances are caused by outliers.

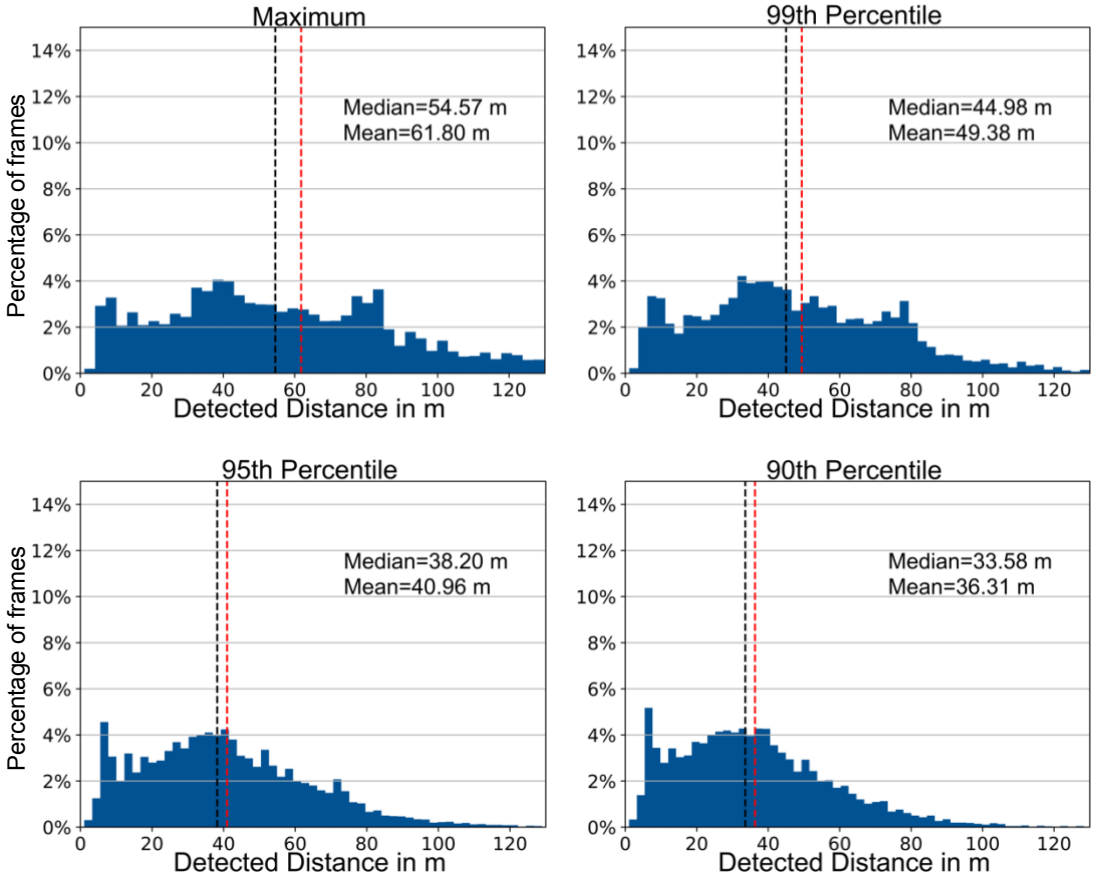

Figure 4.5: Detected free distances for the Oxford RobotCar dataset.

To ensure good comparability and meaningful results, the following data explorations rely on the 99th percentile of detected free distances. An interesting finding one notices when observing Figure 4.5 is the broad distribution of free distance. No concentration, as is the case for A2D2 and KITTI, around the mean or median occurs, but wide ranges of free distance are noticed. Since the data acquisition for Oxford RobotCar took place on the same route repeatedly, these large differences are not due to significantly different road types. Therefore, this distribution and the factors causing it will be investigated in Subchapter 4.2.

This concludes the LIDAR based datasets and subsequently the results for the ApolloScape datasets with disparity and depth maps are presented. Figure 4.6 shows the detected free road distances for the ApolloScape Scene Parsing dataset which provides semantic masks, where filtering out the free road pixels is facilitated, and detailed depth masks for each corresponding frame.

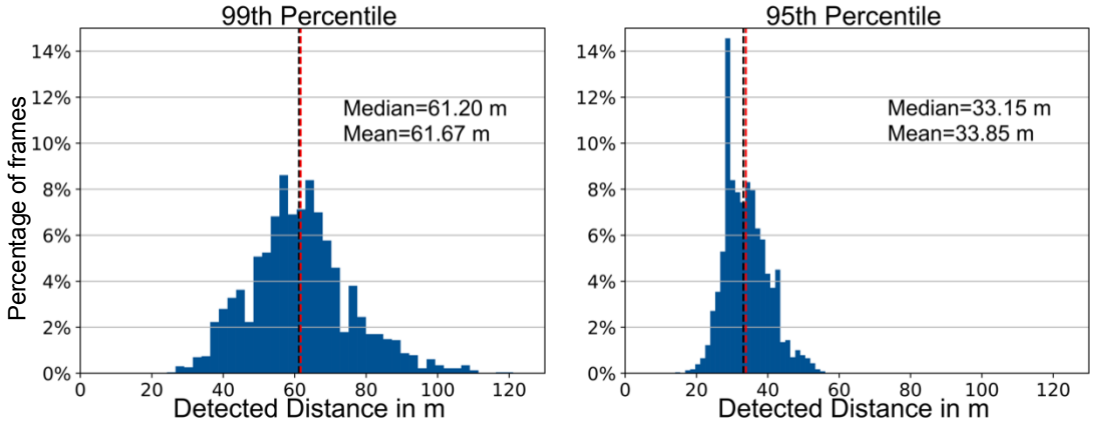

Figure 4.6: Detected free distances for the ApolloScape Scene Parsing dataset.

In this case, the maximally detected free distance is not regarded, since it nearly always showed the maximum depth that is possible if depth maps are stored in the 8-bit `short int` format which results in 325.125 m (Eq. 3.1). This is likely due to small errors on the edge between the road and the horizon and will be further discussed in Chapter 5. Hence, the analysis resorts to using the 99th percentile, which shows a relatively broad, but stable distribution and a median detected depth of 61.20 m.

The ApolloScape Scene Parsing dataset also represents a type of benchmark for this work. On the one hand it provides semantic images with pixel level accuracy, which significantly facilitates the road detection and makes the results of this task highly accurate. On the other hand, it also offers detailed depth maps, which represent a ground truth. For the free road distance calculation these two features allow for a highly precise analysis.

The stereoscopic subset of the ApolloScape dataset shows an interesting outcome, as presented in Figure 4.7. In fact, the detected free road distance is much lower than is the case for the other datasets. This is particularly curious, since ApolloScape Stereo was recorded in the same places as the large Scene Parsing dataset and partly covers the same frames.

This discrepancy between the other datasets and the ApolloScape Stereo dataset could be caused by different factors. One possible explanation lies in the fact, that disparity maps experience a degradation of accuracy at higher distances as laid out in Subchapter 2.1.2. Another reason for these unusual results could be the road detection algorithm. In fact, the

images of the Apolloscape dataset show relatively low illumination and might cause reduced performance of the algorithm. This will be further discussed in Chapter 5.

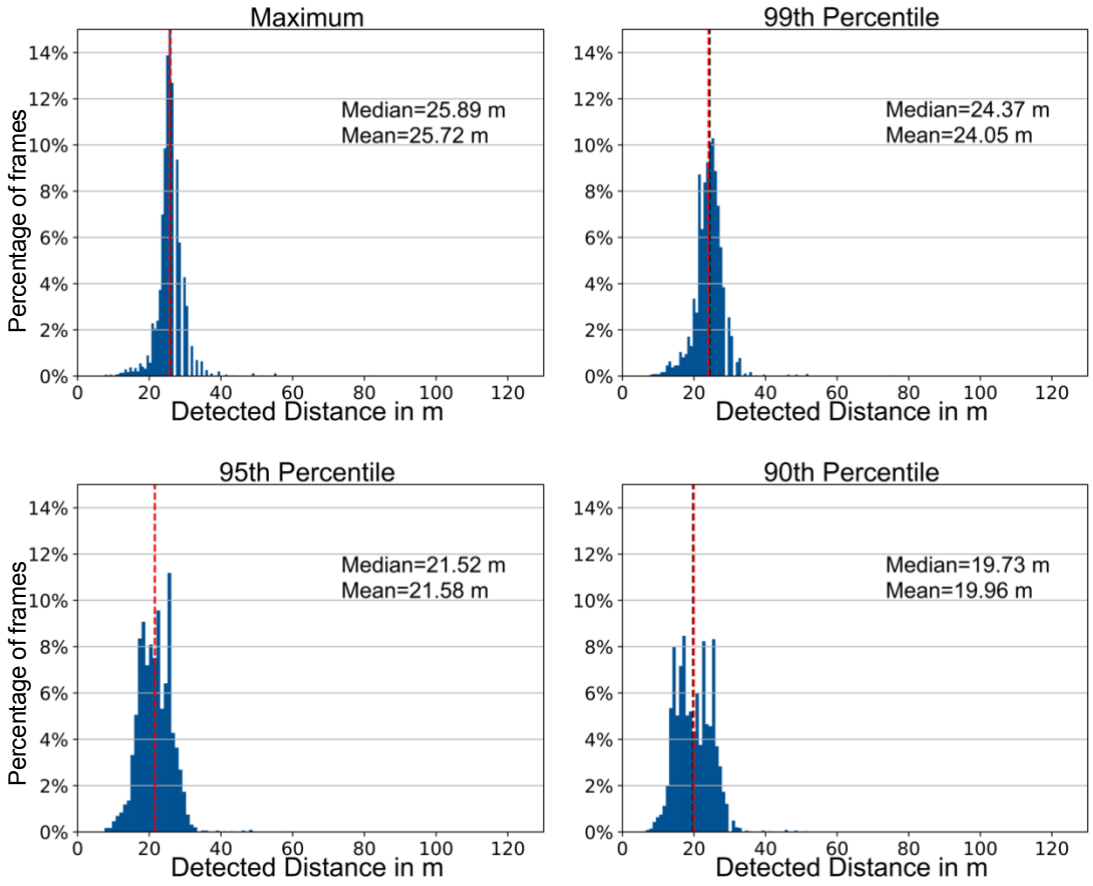

Figure 4.7: Detected free distances for the ApolloScape Stereo dataset.

One other thing to notice when observing Figure 4.7 is the high concentration of detected free distances around the mean/median. This might indicate that, even though stereo cameras are restricted in their maximally detected range, they provide more stable results at shorter distances. While this behavior makes this dataset less interesting for the analysis at hand, it can be leveraged in ADAS applications such as ACC and in AV in general to provide a reliable and cost-effective method for information extraction at close range.

Table 4.1 provides an overview of the relevant raw results for every analyzed dataset. It contains information about the free distances, the number of retained frames after filtering out faulty frames, the variability in driving conditions, as well as figures about the number of blocked frames.

In the following the raw results in form of the free distance ahead of the vehicle are used to qualify frames as presenting unobstructed views. For that matter, different frames from each dataset are investigated and judged to be free of obstruction. Subsequently the detected or calculated 99th percentile free distance, depending on the dataset, is analyzed.

Table 4.1: Overview of the most important raw results of the analyzed datasets

Name	Median free distance (99th Percentile) in m	Number of frames	Night in %	Bad weather in %	Rush hour in %	Blocked in %		
						L	M	H
A2D2	42.49*	4401	0	24.6	25.2	35.7	14.4	9.7
Apolloscape Scene Parsing	61.20	2015	0	0	95.8	64.1	10.9	5.1
Apolloscape Stereo	24.37	3460	0	0	63.3	14.9	0	0
KITTI	43.83*	4182	0	0	0	-	-	-
Oxford RobotCar	44.98	8135	22.1	22.1	56.5	-	-	-

* Calculated distance, Legend: L: Low, M: Medium, H: High

One can thus conclude that frames can be considered free of visibility obstructions if a certain free distance is detected. Hence for frames recorded in the city center or on country roads, a free distance above 80 m is required. However, when vehicles travel at a higher speed as is the case on the motorway, more free distance is required so that a frame can be considered free and no ADAS needs to be alerted. Here, a threshold of 100 m is set. Figure 4.8 holds examples of images with unrestricted visibility for the city center (Oxford RobotCar), the motorway (A2D2), and a country road (KITTI). Free distances of 83.37, 101.58, and 92.47 m have been obtained respectively. Given this free distance and the fact, that no blocking factors have been detected, these images can be considered to offer free visibility.

Figure 4.8: Exemplary frames of free visibility across the datasets and across various road types.

4.2 Influences on the Results

The raw results provide a measure for the free space that lies ahead of the car on average. They allow first conclusions about the free distance that is needed so that one can consider the visibility free and optical sensors unconstrained, by analyzing different frames (Figure 4.8). This already builds a good basis for the subsequent analysis.

In fact, in this subchapter the different influences affecting the results and crucially the optical sensors are more closely investigated. Conclusions about factors such as traffic, weather, and road conditions can thus be drawn. Using the GPS data provided in certain datasets, deductions about the role that the type of road plays are also made. This data can in turn be used to update online map services and alert AV when travelling in conditions exhibiting bad weather or reduced visibility.

The results are thus sliced according to the traffic conditions and resulting blocking factors, the weather and lighting conditions in which frames were gathered, and the type of road where the datasets were recorded.

4.2.1 Traffic and Blocking Factors

A steady trend which ought to continue during the 21st century is the continuous increase in road traffic. This increase is partly due to passenger cars and partly due to the surge in freight transport across the road systems of the world [135, p. 66]. Increased traffic not only causes a rise in car accidents but also amplifies the complexity of scenarios encountered by AV.

Hence, obtaining measures for the extent of traffic and specifically the blocking factors causing visibility reduction is paramount for the future development of AV. In this analysis, the `blocking()` function uses the semantic images provided by the A2D2 and ApolloScape datasets to identify these blocking factors and to qualify the degree of visibility obstruction caused by them, based on the vertical pixel coordinate where the blocking factor has been identified.

The results of this operation can be observed in Figure 4.9. It shows the distribution of relevant blocking factors (cars, trucks and buses, and pedestrians, bicycles and bikes) across the two datasets along with the degree of blockage (low, medium, high) they cause. For the widespread adoption of AV, the interaction between driverless vehicles and pedestrians or cyclists needs to provide a high level of safety and reliability as modern cities will move away from being car-centered to being centered around their inhabitants thus producing more vehicle-pedestrian contacts.

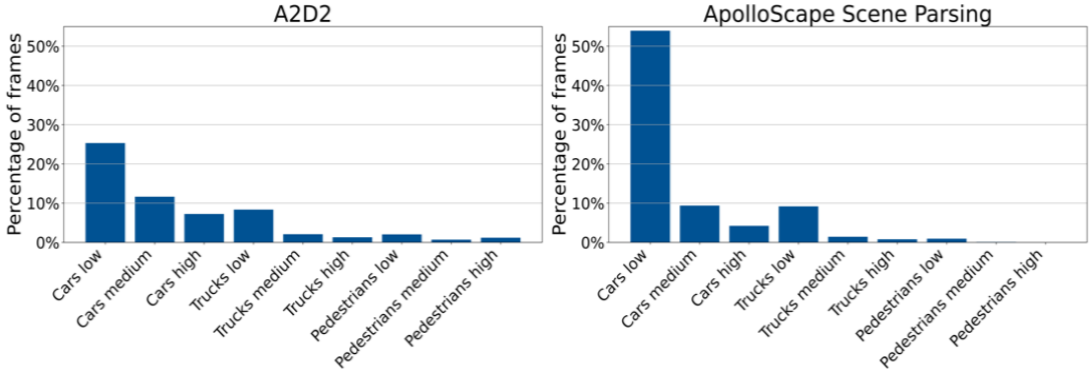

Figure 4.9: Distribution of common blocking factors for A2D2 and ApolloScape Scene Parsing.

Two interesting differences between the datasets arise when looking at Figure 4.9. On the one hand, the ApolloScape frames experience visibility blocking a lot more commonly than is the case for A2D2. Considering Table 4.1 this discrepancy can be explained by the fact, that nearly all (95.8%) of the frames captured by the ApolloScape research vehicle were taken during rush hour, whereas this proportion is a lot lower for A2D2 (25.2%). Moreover, the settings in which the images of these two datasets were acquired differ substantially. Audi covered lots of miles on the motorway but also on secondary and tertiary roads (OSM) (see Figure 4.18), which are similar to country roads with reduced traffic. Baidu, however recorded their ApolloScape dataset in large Chinese cities with large motorways and primary roads with increased traffic volume and thus more cars blocking the free visibility (Figure 3.4 and Figure 3.11).

On the other hand, A2D2 frames are more prone to have their visibility reduced due to pedestrians and bicycles than is the case for ApolloScape. This is again due to the different environments in which the respective research vehicles circulated. Audi also covered parts of the German city centers where lots of pedestrians and cyclist are usually present, something that is only marginally present in the Apolloscape datasets. A2D2 thus offers more diverse driving environments and consequently its results paint a picture that is more closely related to real-world scenarios.

As laid out in Subchapter 3.4, once a blocking factor is identified, the devised method retains the distance up to the latter as the actual free distance. This ensures that for frames that are heavily occluded but show a small and, in this case, irrelevant spot of road at a larger distance, the free space the car can actually reach is correctly retained. A comparison between the free distances of the medium or highly blocked and free or lowly blocked frames is provided in Figure 4.10.

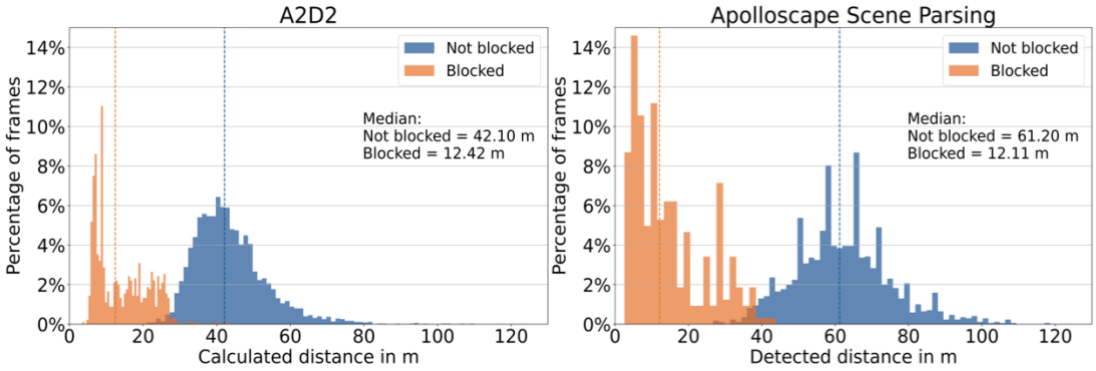

Figure 4.10: Influence of blocking factors on the free space ahead of the vehicle. Analysis based on the semantic images provided in the A2D2 and ApolloScape Scene Parsing datasets.

The histograms in Figure 4.10 clearly exemplify how strongly the free road distance is constrained and affected by strong traffic. A reduction of close to 30 m of free space is thus observable for both cases. Such a steep decline in freely accessible space has implications for ADAS such as ACC or emergency breaking, which should be immediately alerted to step in.

For the blocked frames one can observe that the median distance until a considerable blocking factor is reached is very similar for both datasets. A median free distance of only 12 m is detected or calculated in such a scenario. The free space and crucially the information camera images can provide in a scene like this is thus extremely limited and the AV need other sensors or indeed the driver to guide itself through dense traffic scenarios. Figure 4.11 provides a sequence of frames where the research vehicle enters a dense traffic situation. The devised algorithm returns diminishing distances until the first blocking factor of 24.40, 16.49, 15.48, 13.47, 12.30, 9.94,

9.25, 9.22, and 9.02 m for the frames in order of appearance. This example showcases that the algorithm works in a stable manner and is able to detect blocking factors and calculate the correct distance up to them.

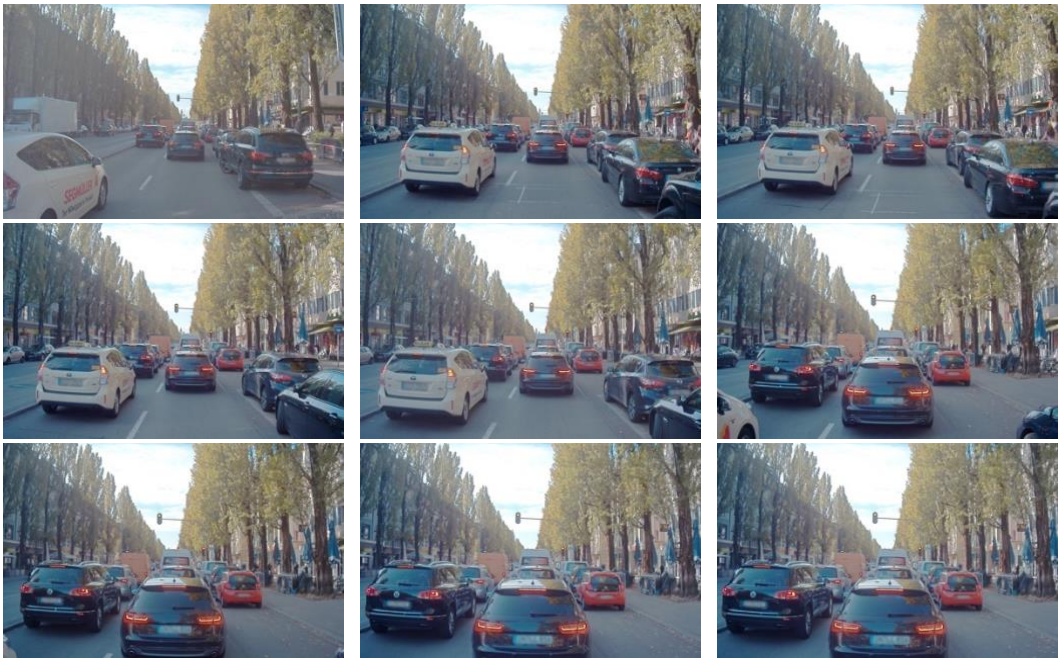

Figure 4.11: Traffic sequence exemplifying that the algorithm works in an expected way and catches frames whose visibility is heavily occluded.

On the other hand, when considering the free road distances detected by the LIDAR scanner (99^{th} percentile) from the A2D2 dataset, it does not seem to be affected by high traffic volume (Figure 4.12). In fact, its median detected distance even slightly increases in such settings. This behavior can be explained by the fact that a LIDAR scanner rotates and collects points from its whole horizontal FOV (360° in this specific case). As a consequence, it is able to detect even small patches of free road surface which lies ahead of the actual blocking factor.

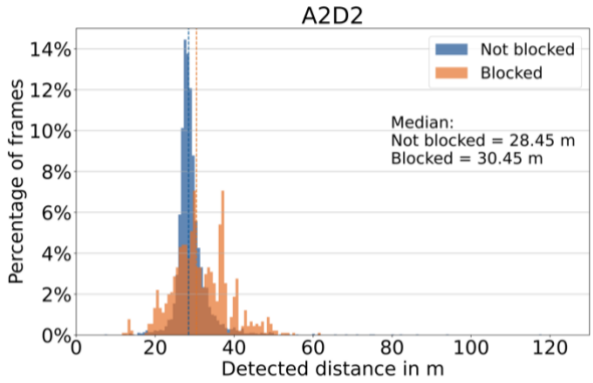

Figure 4.12: Free road distance detected by the LIDAR scanner in A2D2 in the case of identified blocked or free frames.

As a result, this measure is not very useful to the autonomous navigation process since it provides a false estimate of the freely accessible space. To remedy this problem, either the method described above which relies on semantic images or a built-in object detection program which directly uses LIDAR data can be used to fulfill the task correctly.

Considering the relatively broad distribution of detected free distances for RobotCar, one factor for explaining the partly very small free distances lies in the acquisition of frames at the beginning and the end of each recording of RobotCar. In fact, the research vehicle starts and arrives at the identical location which is in a closed parking space with close to no free space as a consequence (Figure 4.13). Furthermore, some recordings were taken during rush hour on narrow British roads with a lot of traffic (Figure 4.13), which again constrains the LIDAR availability and detected free distances.

Figure 4.13: Exemplary frames from Oxford RobotCar highlighting different factors that diminish the detected free distance. 8.91 m and 6.19 m of detected free distance on the left and right, respectively.

4.2.2 Weather and Lighting Conditions

For AV to operate reliably and to be consequentially adopted by the public, they need to perform and be able to navigate under varying environmental conditions. As most autonomous driving datasets have been recorded in clear and sunny conditions during the day (e.g. KITTI, ApolloScape) more data is needed to evaluate the performance of sensors and thus of AV in bad weather and tricky lighting conditions.

In this analysis both A2D2 and Oxford RobotCar have a portion of frames recorded during bad weather including rain, fog, and snow (Table 4.1). Furthermore, the RobotCar research vehicle also travelled for parts of its recording during nighttime, which differentiates it from the other datasets and makes it interesting for the following exploration.

Weather

In a first step, the influence that precipitation has on the sensor availability is analyzed. To this end frames such as the ones in Figure 4.14 are considered. The left image shows a frame with a wet road surface shortly after it rained, whereas the right image displays a scenario with strong fog and a wet road surface. In both cases one notices the relative sparsity of the point cloud when compared to images from the same dataset in clear conditions with an overlaid point cloud (Figure 2.16 and Figure 3.3).

This decrease in density is the first indication that the LIDAR scanner suffers performance decrease in bad weather conditions. It decreases the ability of the sensor to reliably detect obstacles and thus reduces the capability with which AV can navigate safely under such conditions.

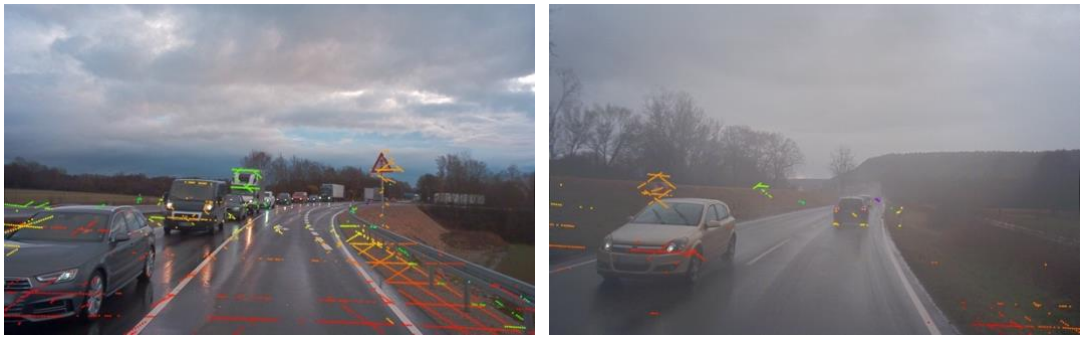

Figure 4.14: Decrease in point cloud density in poor weather conditions for A2D2 LIDAR.

This sparsity of the 3D point cloud has further consequences. In fact, to be a reliable source for the visual scene understanding, the high-frequency LIDAR scans need to be consistent across frames displaying a similar scene. However, given weather conditions with rainfall or dense fog, this consistency is not given anymore, and large standard deviations are observed (Figure 4.15). This is due to the fact, that in conditions with small weather droplets hanging in the air, LIDAR signals are often prematurely refracted or completely reflected by them, before reaching their designated targets as explained in Section 2.2.1. Hence, the variance of LIDAR points and distances strongly increases in bad weather conditions which reduces its reliability and thus its availability for safe navigation.

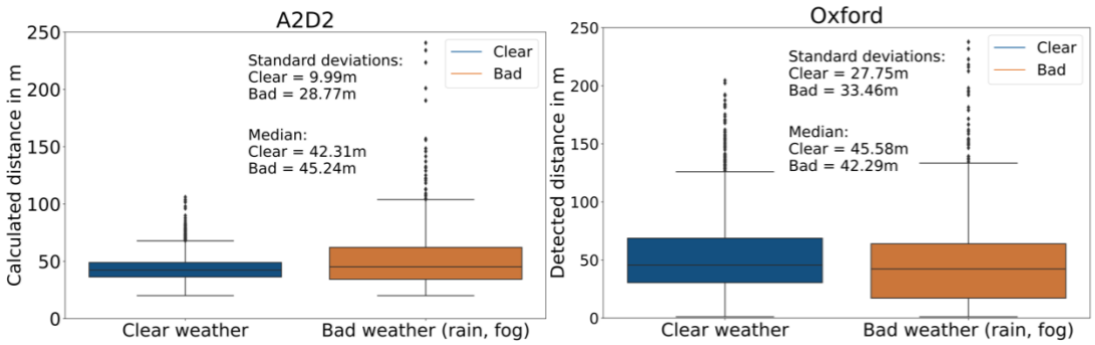

Figure 4.15: Influence of weather conditions including precipitation on the free distance in the A2D2 and Oxford RobotCar datasets.

Interestingly, the median detected, or calculated distances do not seem to be affected by bad weather conditions. This behavior might however be due to outliers at the top which distort the actual results.

Furthermore, even though the increase in standard deviation is considerable in the case of the Oxford RobotCar dataset, it is significantly higher when observing the boxplot of A2D2. One difference between A2D2 and Oxford RobotCar explaining this discrepancy lies in the fact, that LIDAR scanners of different makes have been employed. The LIDAR scanner in the RobotCar research vehicle generally catches points at larger distances but this increase in range also comes with an increased variance, even in clear conditions. A2D2 point clouds are on the other hand much more ordered and restricted in range in clear conditions, which can be seen when comparing the images in Figure 2.16.

Hence, to conclude one can state that depending on the LIDAR make, the sensor is more or less deeply affected by bad weather conditions. When the conditions worsen and the standard

deviation of detected distances increases, the LIDAR scanner is not well suited to inform the AV reliably about its surroundings and other sensors must fill that gap to be able to maneuver the vehicle.

Lighting

Thanks to the fact, that parts of the RobotCar dataset have been recorded during dusk and night, a comparison between the performance and availability of the LIDAR during the day and during the nighttime or in tricky lighting conditions can be performed. To that end the raw results have been sliced in two subsets according to the time of day at which the frames were captured, to obtain one part of frames acquired during the night and another part during the day.

Figure 4.16 provides an exemplary frame that was captured during dusk. Moreover, the weather conditions were not favorable in this case, as rain drops on the camera lens cause distortions. This, in combination with the lights of the car in front and the streetlights creates a difficult image setting. In these conditions, the performance of algorithms relying on camera images, such as obstacle detection and classification, is sharply reduced as is laid out in Subchapter 2.1 as well as in Table 2.1.

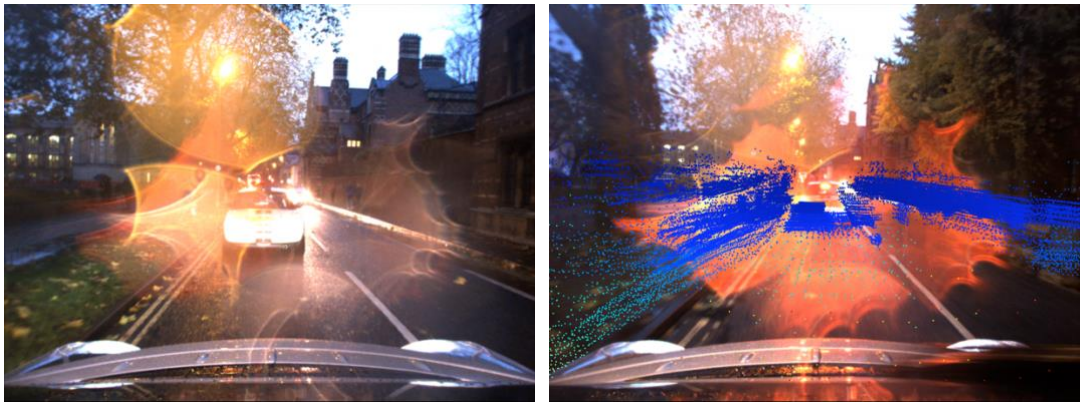

Figure 4.16: Exemplary RobotCar frame captured during dusk with its overlaid point cloud on the right.

In contrast to images which were taken under precipitation, no decrease in point cloud density is observable in this case. However, when looking at Figure 4.17, one notices a significant decrease in median detected free distance of nearly 13 m as well as an elevated standard deviation. The higher standard deviation is explicable by the fact, that there exists an overlap in the nighttime driving and the driving during bad weather conditions in the analyzed subset of RobotCar. The lower detection range is however a trait that is exclusive to the frames recorded at night or dusk and is cause for concern when developing and operating AV.

As touched upon in Section 2.2.1, this range reduction is caused by the lower reflectivity of surfaces at night and at larger distances. For nighttime driving both camera and LIDAR sensors are thus significantly constrained in their ability and availability. The reduction in image quality due to lower exposure at night or foggy conditions also hampers the crucial road detection algorithm which sits at the top of this analysis.

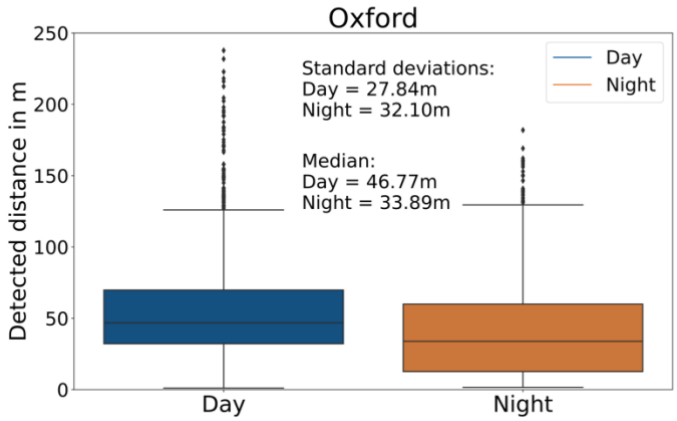

Figure 4.17: Influence of lighting conditions on the free distance in the Oxford RobotCar dataset.

4.2.3 Road Type

In a final step of post-processing, the effect that the type of road has on the free space is investigated for the datasets where precise GPS data is available (A2D2 and KITTI). To this end the GPS coordinates corresponding to every frame are used in combination with the `get_road_category()` to get the OSM road category and crucially the type on which these frames were recorded.

Unfortunately, the GPS coordinates provided in the Oxford RobotCar dataset are not precise enough for this kind of analysis. In fact, given that most of the recordings are located in the city with a dense road network, the OSM database could not match the GPS coordinates to a precise road type.

Hence, for this investigation, only the A2D2 and KITTI datasets are analyzed, since ApolloScape does not provide any GPS information. As laid out in Subchapter 3.5 only the frames which can be located on the OSM road category “highway” are retained. Figure 4.18 presents the distribution of road types (with “highway” as category) for both German datasets.

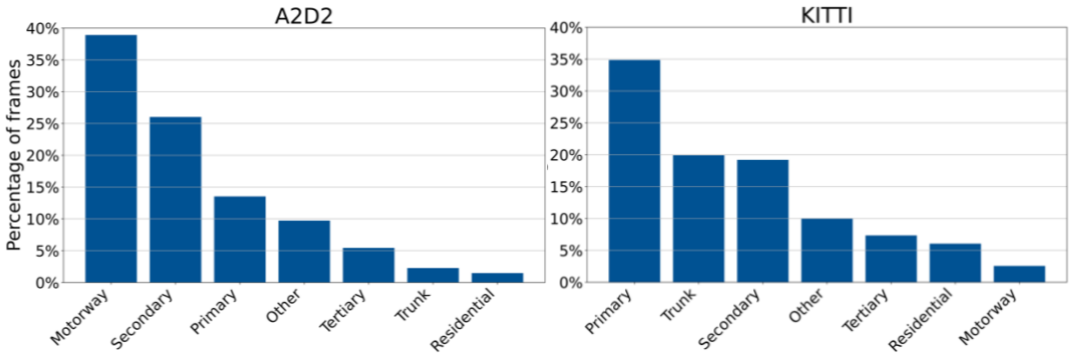

Figure 4.18: Comparison between the OSM road type distribution among the A2D2 and KITTI datasets, which provide detailed GPS data.

This part of the analysis is divided into two sections: the influence of the road type on the whole detected free space and afterwards it looks into how the road type plays into the free space up to the first blocking factor.

In a first step, the effect the type of road has on the calculated free distance is investigated. Figure 4.19 provides the relevant histograms for the A2D2 dataset. Here, the free space of frames captured on the motorway is compared to those that were acquired on primary and secondary roads as they are the most common road types in A2D2.

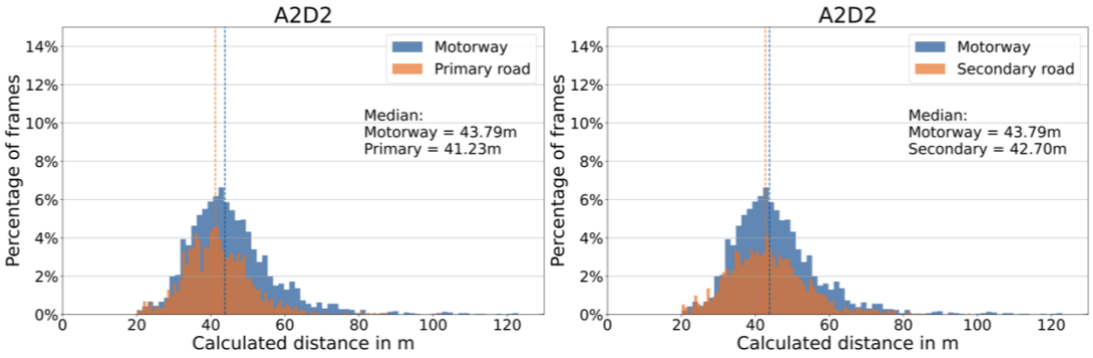

Figure 4.19: Calculated free distances for motorways, primary and secondary roads for A2D2.

A similar pattern can be observed when looking at the same data for the KITTI dataset. In fact, observing Figure 4.19 and Figure 4.20, the road type does not seem to have a significant effect on the free distance apart from the difference in free distance when comparing the motorways with the secondary roads of the KITTI dataset. The discrepancy, when comparing this to the same plot for A2D2, is probably occurring because of the inner city driving in KITTI. Some recordings of the dataset were done in the pedestrian zone in Karlsruhe, with lots of pedestrians blocking the visibility and reducing the free space, thus altering the results for secondary roads in this case. Furthermore, these roads present more curved road geometries which affects the method for calculation and road detection and thus furthers this decrease.

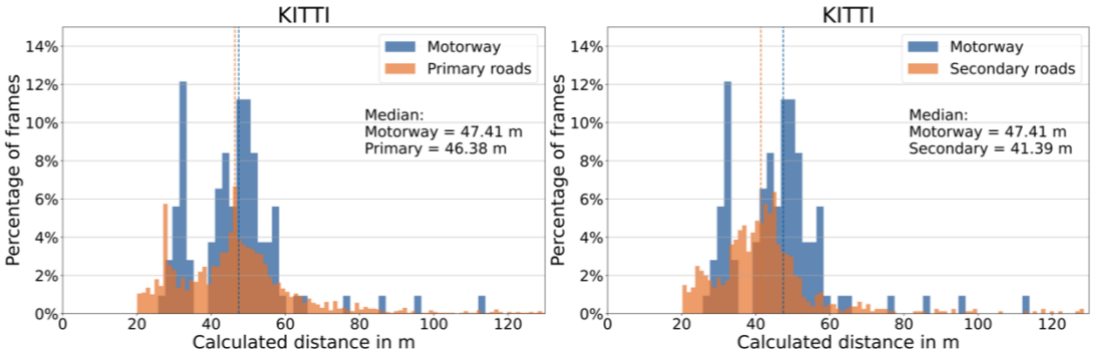

Figure 4.20: Calculated free distances for motorways, primary and secondary roads for KITTI.

For the second part of the analysis, a look at the distance until a blocking factor is reached is taken. This step ought to give a more accurate picture of the real data since the amount and shape of traffic varies across different road types. Because blocking factors were only investigated for datasets with accompanying semantic images, this step is limited on the A2D2 dataset.

This investigation yields a clear distinction between the different types of road (Figure 4.21). In fact, when comparing the median free distance of frames on the motorway to the one on primary or secondary roads the visibility distance is reduced by over 6.50 or 9.50 m respectively. This

significant decrease in free distance is largely due to the different traffic environments and driving styles on these different roads.

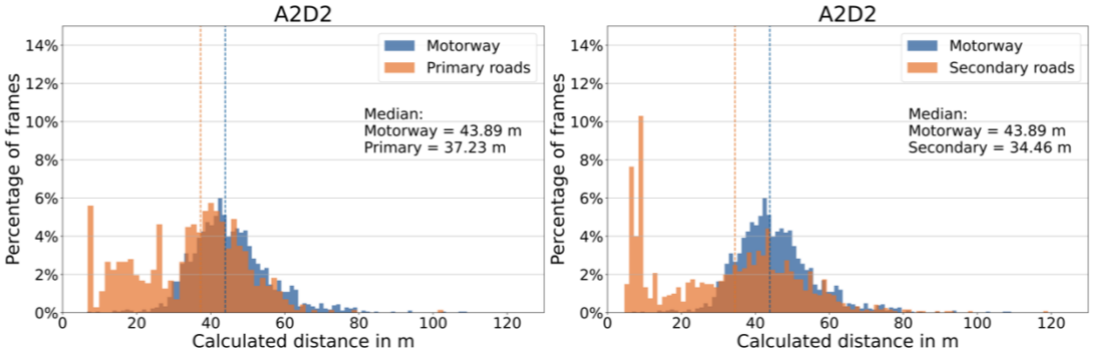

Figure 4.21: Influence of road types on the free distance up to the first blocking factor for A2D2.

As such, when driving on the motorway, vehicles usually keep a larger safety distance between each other in consequence to the increased travel velocity. This distance is only decreased when experiencing dense traffic on the motorway which was not encountered during the recording of the dataset which becomes evident when observing that the median distance until the first blocking factor (43.89 m) is nearly identical to the median of the calculated free distance (43.79 m). This indicates that close to no significant blocking factors have been detected on the motorway.

The situation is different, however, when it comes to the smaller primary and secondary roads. These types of road, often located in and around larger cities experience dense traffic on a regular basis. Hence, more blocking factors are detected and thus decreased free space is available. Hence, the type of road is, along with the blocking factors, a central influence on the free space ahead of the vehicle.

5 Evaluation and Discussion

Following the chapter presenting the obtained results, this part of the work evaluates the used methods and datasets and remarks possible shortcomings of them. These can in turn be used to improve the algorithm and extend it to more datasets. Furthermore, impacts on the results are assessed based on their relevance for the development of optimized online map services. The chapter concludes with a discussion about various sensor setups and their performance in adverse conditions.

5.1 Method and Datasets

Considering the processing pipeline of the implemented method in Figure 3.10, the basic task consists in analyzing different frames by detecting the free road ahead of the vehicle, getting a real-world distance measure for the free road pixels, and possibly identify relevant factors which constrain this free road distance and thus the optical sensors with which the research vehicles were equipped. During the actual analysis, different shortcomings of the method and especially the used datasets have presented themselves.

Since the implemented method is closely interlinked with the dataset and the types of data it provides, both instances are assessed in parallel and evaluated according to different steps of the exploration process.

5.1.1 Road Detection

Depending on the available sensor data for each dataset, different methods for road detection have been devised (see Subchapter 3.2). Some show however better accuracy and stability than others, which, since the road detection sits at the beginning of the analysis, ultimately affects the results and the usefulness of them considerably. In this section, the strengths and weaknesses of the different algorithms are investigated.

When using the ApolloScape Stereo dataset, a road detection algorithm based on camera images only has been applied. It is based on detecting lane markings by color filtering and edge recognition and defines a tapering road corridor originating from the car and reaching up to the horizon. It is thus very sensitive to the image quality and requires the presence of road markings for a successful detection. Furthermore, it only works well in scenes with a relatively straight road geometry. The ApolloScape Stereo images suffer however from relatively low illumination which reduces the contrast of lane markings compared to the asphalt of the road and thus hampers the Canny edge detection, which is crucial to this algorithm. As a result, the lane detection does not always provide satisfactory results as can be seen in Figure 5.1, which is a scenario of no road markings, a non-straight road ahead and relatively low illumination.

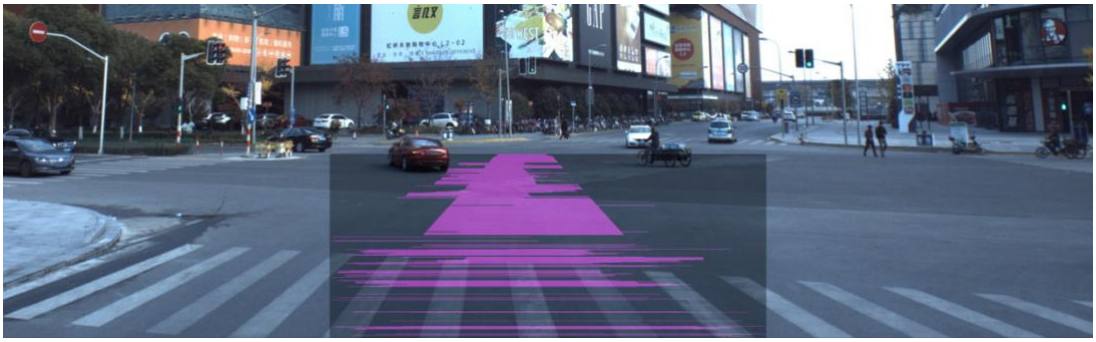

Figure 5.1: Exemplary frame of the ApolloScape Stereo dataset where the road detection algorithm fails.

Hence for datasets which, in addition to camera images, also offer LIDAR data, the road detection method is slightly altered. As laid out in Subchapter 3.2, since the global position of the LIDAR scanner is known, the flat road assumption can be leveraged to filter out points which are too high above the road surface to actually be part of it. This filtration delivers, in combination with the road detection based on image data only, a more stable and precise result. It is not only more resilient against the absence of road markings, but also against bad lighting and provides good results even for frames recorded during the night, as seen in the analysis of the Oxford RobotCar dataset. It does however also suffer from bad detection when it is confronted with roads that show significant curvature or no clear road ahead, such as in Figure 5.1.

Consequently, datasets which offer complementary semantically segmented images provide the best and most accurate results. They are not restricted by bad weather conditions and gather information about the road curvature as well. Hence, A2D2 and ApolloScape Scene Parsing serve as a sort of benchmark in this regard, and they also present the most useful results.

5.1.2 Distance Calculation

The free distance calculation is based on the road detection and in this analysis, three different distance information formats were utilized: LIDAR, stereoscopic vision with corresponding disparity map, and manually annotated depth maps. The differences between the datasets are explored in the following and the shortcomings of each of them are laid open.

As explained in Subchapters 2.2 and 2.3, LIDAR scanners are the go-to method for depth estimation in most research vehicles, because of their high range and precision. In the case of the A2D2 and KITTI datasets, which both rely on Velodyne LIDAR sensors, the range was relatively restricted. This in turn led to the implementation of a regression to find out the distance of free road sections that are farther away from the vehicle. Due to this, two problems arose. First, even though the method works well, some precision is lost when compared to LIDAR scanners that do not suffer from range restriction such as the one in Oxford RobotCar. Moreover, in conditions where the road detection failed (KITTI) or the point cloud is very sparse in bad weather conditions (A2D2) the regression also produces distorted results with unreasonable distances. These frames then have to be discarded and cannot serve in the analysis. This again emphasizes the importance of good road detection. Such problems do not occur for Oxford RobotCar frames, since the LIDAR scanner reliably detects points of over 100 m distance, thus resulting in lower error rates and fewer discarded frames.

On the other hand, ApolloScape provides disparity and depth maps for their Stereo and Scene Parsing subsets, respectively. These differ in the fact that disparity maps based on stereo images can be produced in real-time during the operation of the vehicle, as explained in Section 2.1.2, whereas the detailed depth maps have been produced after capture.

However, as seen in Figure 4.7, the results for the stereo vision-based dataset turned out surprisingly lower than expected. This is curious, since the same frames are also part of the much more precise Scene Parsing subset, which does not show this behavior. This effect is likely due to two separate factors. First of all, the road detection algorithm for the stereo subset (which does not provide semantic images) does not always work in a satisfactory manner (Figure 5.1), which does negatively influence and distort the results. Furthermore, as pointed out in Section 2.1.2, disparity maps suffer from small errors at large distances, which in turn causes large errors in the real-world distance, since they are inversely proportional. Both of these factors lead to the lesser median free distance detected in this dataset when compared to the Scene Parsing subset.

This indicates that even though a stereo camera setup is very cost effective and relatively easy to implement, it struggles to paint an accurate picture of the scene at larger distances. It does however offer a stable mechanism to detect objects and free space (e.g. by using a v-disparity algorithm as laid out in Subchapter 2.5) and get a distance measure accordingly at a closer range.

The manually annotated depth maps of the ApolloScape Scene Parsing dataset represent the gold standard for an analysis of this type. They provide real-world distance measurements for every pixel in the scene and thus represent a straightforward way for analyses of this type. In combination with the segmented images, they deliver the most precise results across the investigated datasets.

These depth maps are however computationally complex to obtain and are thus not suited for real-time operation. Furthermore, there seems to be some lost precision at the very end of the free road surface, where small pixel errors in the depth maps or semantic images cause the distance calculation to fail. They are however overcome, by using the lower percentiles of the free distance.

Generally, the free distance up to the first blocking factor (Figure 4.10) carries the most useful and meaningful information on free space and optical sensor availability. To that end, A2D2 and ApolloScape Scene Parsing, offer the best basis in the form of their semantically segmented images. This is a major shortcoming of the KITTI and RobotCar datasets for this analysis and emphasizes the importance of semantic masks once more.

5.1.3 Post-Processing

During the post-processing step of the analysis, different factors that influence the results are identified. To that end GPS coordinates are used to determine the road category and type on which the frames have been recorded. This is however only possible for the A2D2 and KITTI datasets. ApolloScape does not provide GPS coordinates in their download packages which reduces the level of detail in its analysis. The RobotCar dataset on the other hand does deliver coordinates from its INS unit at regularly spaced intervals. They are however not precise enough to get accurate road matching using the OSM database in the dense road network of central Oxford. Thus, no analysis of the influence that road types have on the results could be accomplished for the ApolloScape datasets and Oxford RobotCar.

To conclude this critical assessment of the datasets and the implemented methods, it is important to note, that even though some datasets provide more or less detailed information, the variety in scenes and sensors allows for a representative investigation and serves as a solid basis for further dataset exploration. Moreover, the depth maps in the ApolloScape Scene Parsing dataset serve as a good benchmark for the other distance measuring sensors and allow a conclusion about their relative performance.

5.2 Influences on the Sensor Availability

As observed in Subchapter 4.2, the raw results are affected to a varying degree by different factors. The analyzed influences are the blocking factors, weather and lighting conditions, and the type of road. These affect the free distance and sensor availability in different ways however, which are further discussed in the following.

It turns out that adverse weather conditions do in fact not significantly reduce the median detection range of the LIDAR scanners. However, this interesting behavior is provoked by a small number of large outliers that do not represent the true scene situation. As a result, one observes large standard deviations in detection distances which reduces the reliability of LIDAR sensors in these conditions significantly.

Lighting conditions also play a big role when it comes to LIDAR performance. In fact, when analyzing the frames captured at night for the Oxford RobotCar dataset to those acquired during the day, a considerable decrease in detection range is observable. In addition to this, the standard deviation also increased which renders the information LIDAR sensors gather during nighttime or bad lighting conditions less useful to the operation and navigation of AV.

The most significant effect on optical sensors and their free range is however ascribable to traffic and the blocking factors (Figure 4.10 and Figure 4.11). In such scenarios other types of information such as precise GPS coordinates in combination with online map services are required for the AV or ADAS to function accordingly as optical sensors cannot provide useful scene understanding.

Considering the types of road, a clear correlation between the rush hour traffic in form of blocking factors and the type of road could be observed (Figure 4.21). When analyzing the free distance up to the first blocking factor caused by an obstacle, one notices that motorways are less affected by visibility reduction due to traffic than primary and secondary roads.

5.3 Critical Discussion about Sensor Setups

After this analysis, a critical discussion about different sensor setups, their strengths, and weaknesses with relation to this method is of interest. The goal of this discussion consists in defining a setup that is resilient against adverse environmental conditions as well as dense traffic to guarantee a stable operation of various ADAS used in autonomous technology.

As discussed above, the image capture is paramount and lays a first step for subsequent algorithms such as road and obstacle detection. Hence an optimal setup of optical sensors must include a camera system with at least one instance oriented in driving direction.

To get a distance measure of obstacles ahead and inform ADAS applications such as ACC to perform certain maneuvers, additional sensors are needed. Optimally a LIDAR scanner with a range of over 100 m and dense scene coverage is employed. A large range is required in combination with a dense point cloud to optimize scene understanding in the immediate range, but crucially also at larger distances, which is critical to safe autonomous navigation.

However, LIDAR scanners suffer performance decreases in adverse weather and challenging lighting conditions (Figure 4.14, Figure 4.15, and Figure 4.17). Hence, in order to obtain a resilient and reliable sensor system, alternative supports for these scenarios are needed. To that end, a RADAR sensor or a more cost and size effective stereoscopic camera system could be used. They both offer good stability in every weather condition (Table 2.1), particularly at shorter range, and can thus support the LIDAR scanner.

Real-world road scenarios also regularly include high traffic volume and diminished visibility (Figure 4.11 and Figure 4.13), where purely optical sensors, such as cameras, LIDAR, or RADAR cannot provide any useful information to help navigate the vehicle safely. In such scenarios SLAM applications can still offer some degree of autonomy. They require however very precise GPS data to locate the vehicle in its surroundings. Additionally, a connection to the internet or cloud could be of vital importance for the secure and dependable navigation of AV. In fact, online map services which are updated in real-time by other vehicles, can hold useful information about upcoming traffic, unusual road obstacles such as construction sites, or hold information about weather conditions at upcoming locations. The AV can thus process this information to increase its degree of autonomy, the safety of its operation, as well as the comfort of its passengers. This particular step is further discussed in Chapter 6.

As discussed in Subchapter 5.1, the combination of semantic images and depth maps yield the most precise results. However, both of these data sources are computationally complex to obtain and are not yet possible in real-time, as explained in Section 2.1.6. Therefore, alternative real-time methods to detect and locate obstacles are required. To this end, one can leverage the LIDAR sensor, whose point clouds deliver enough information for obstacle detection, given that the point cloud is dense enough (Table 2.1).

6 Summary and Outlook

6.1 Summary

Along the preceding chapters a method for measuring the optical sensor availability as well as the free space has been implemented and the results introduced. An introduction and comparison between optical sensors based on their strengths and shortcomings has been carried out. Common CV techniques which support the image processing and lane detection algorithms have been presented.

Results obtained by using different distance and free space calculation methods showed a good comparability between the diverse datasets which were analyzed. These methods have been validated (Figure 4.8, Figure 4.11, and Figure 4.16) and assessed based on their shortcomings in Subchapter 5.1. Subsequently, important factors which constrain the optical sensors in range or in general performance have been investigated and the distance up to the first blocking factors has been identified as the most important measure for visibility, free space, and sensor availability.

The acquired and analyzed data can be used and exploited in different ways. In a first step, the method and results can be used to establish a benchmark for the free space estimation with a distance measure included across various new datasets. On the other hand, when using such a method on board of an operating vehicle, data about visibility, free space, or anomalies on the road can be uploaded to online map services (Figure 6.1). This enriches the map environment with important information, which can be accessed by other vehicles for various purposes: pre-alerting ADAS or the passengers of the vehicle about traffic conditions, improve the safety of the autonomous navigation, and thus increase the comfort of the passengers. Moreover, as Retallack et al. [136, p. 1] note, real-time information about traffic conditions can alleviate “accident-prone conditions before they can fully develop”.

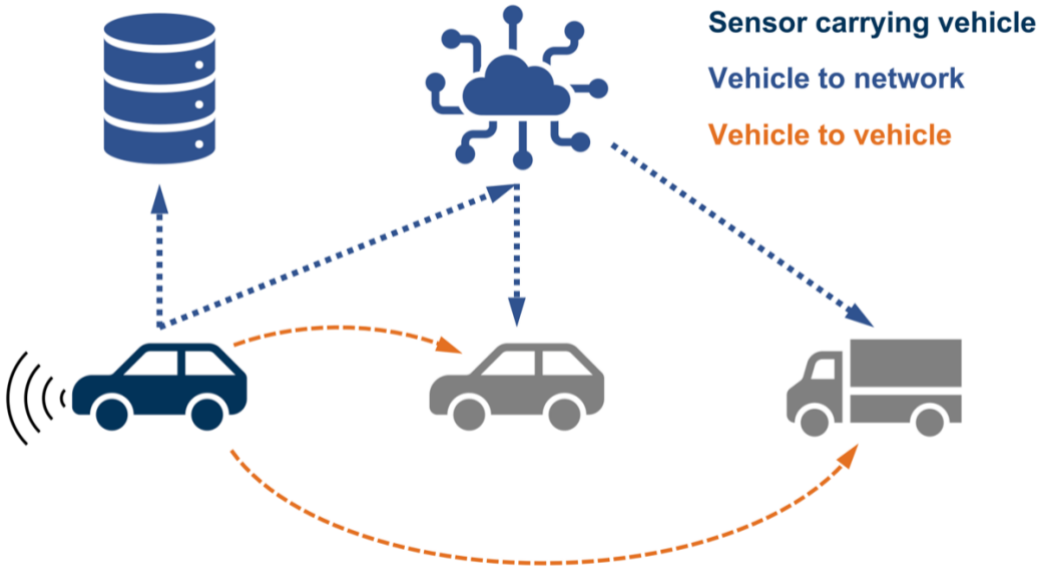

Figure 6.1: V2X communication, where the vehicle in front shares information about its surroundings with the network, online map services, and directly with other traffic members.

In addition to this, recent developments in 5G communication technology [137, pp. 972–973], allow for more connected vehicles and thus for faster and more reliable V2X communication, where a vehicle equipped with a full sensor suite can inform other cars, the network, or smart road infrastructure about its surroundings and possible anomalies (Figure 6.1). This in turn enhances the performance of AV and facilitates data acquisition from traffic.

6.2 Outlook

Considering the devised method, the question of what can be achieved with it in the upcoming future arises. Thanks to the architecture of the code, it can be easily adapted to different approaches regarding the problem of optical sensor availability and free space calculation.

In a first step, more datasets could be investigated to confirm the findings and maybe discover new information and influences during the post-processing. In fact, all the datasets presented in Subchapter 2.3, as well as a myriad of other autonomous driving datasets, can be investigated using the provided framework, given that they at least provide some kind of depth information acquired by optical sensors in combination with raw or augmented camera images.

Complementary to the point above, a different kind of sensor setup can be investigated. In fact, for the few datasets that provide both LIDAR and RADAR data, it could be interesting to compare the results and subsequently draw conclusions about the strengths and shortcomings of each sensor. Moreover, for a computationally more lightweight analysis, LIDAR data could be used for obstacle detection instead of camera images, thus speeding the algorithm up considerably.

As discussed above, one notices that semantic images allow for the most precise and meaningful results in this regard. They are however computationally expensive to obtain and are thus currently not suited for real-time applications during the operation of AV. However, the continuous advancements in computing and processing power may make this feat possible within the next few years which could represent a game changing breakthrough for the navigation of autonomous cars in general. Furthermore, methods for real-time semantic segmentation relying on RADAR signals are currently being developed [138].

Another factor that may be highly relevant in restricting the information optical sensors are able to provide is road geometry. In fact, turns, large gradients, or strong elevation changes pose a challenge to optical sensors, because they are unable to overcome solid obstacles with their signals. For this kind of exploration, semantic images come in very useful again since they capture the curvature of the road well.

List of Figures

List of Tables

Bibliography

[1] SAE International, “Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles,” *SAE International*, vol. J3016_202104, Apr. 2021.

[2] A. Haghi, D. Ketabi, M. Ghanbari, and H. Rajabi, “Assessment of human errors in driving accidents; Analysis of the causes based on aberrant behaviors,” *Life Science Journal*, vol. 11, no. 9, 2014.

[3] M. Ramamurthy and V. Lakshminarayanan, “Human Vision and Perception,” in *Handbook of Advanced Lighting Technology*, Cham: Springer International Publishing, 2015. doi: 10.1007/978-3-319-00295-8_46-1.

[4] D. Marr and T. Poggio, “A computational theory of human stereo vision,” *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 204, no. 1156, May 1979, doi: 10.1098/rspb.1979.0029.

[5] C. V. Raman, “The role of the retina in vision,” *Proceedings of the Indian Academy of Sciences - Section B*, vol. 56, pp. 77–87, Aug. 1962.

[6] V. K. Kukkala, J. Tunnell, S. Pasricha, and T. Bradley, “Advanced Driver-Assistance Systems: A Path Toward Autonomous Vehicles,” *IEEE Consumer Electronics Magazine*, vol. 7, no. 5, Sep. 2018, doi: 10.1109/MCE.2018.2828440.

[7] Z. Pizlo, “Perception viewed as an inverse problem,” *Vision Research*, vol. 41, no. 24, Nov. 2001, doi: 10.1016/S0042-6989(01)00173-0.

[8] R. Szelisky, *Computer Vision Algorithms and Applications (Texts in Computer Science)*. 2010.

[9] W. Gonzalez and R. E. Woods, *Eddins, Digital image processing using MATLAB*. 2004.

[10] J. Geyer *et al.*, “A2D2: Audi Autonomous Driving Dataset,” Apr. 2020, Accessed: Aug. 05, 2021. [Online]. Available: https://arxiv.org/abs/2004.06320

[11] A. C. Barkans, “Color recovery: true-color 8-bit interactive graphics,” *IEEE Computer Graphics and Applications*, vol. 17, no. 1, 1997, doi: 10.1109/38.576860.

[12] L. Schöbel, “Proposal of a segmentation based lane detection in urban environments,” Munich, 2018.

[13] K. M. Dawson-Howe and D. Vernon, “Simple pinhole camera calibration,” *International Journal of Imaging Systems and Technology*, vol. 5, no. 1, 1994, doi: 10.1002/ima.1850050102.

[14] D. Forsyth and J. Ponce, *Computer vision: a modern approach*. 2003.

[15] OpenCV, “Camera Calibration and 3D Reconstruction.”

[16] K. Hata and S. Savarese, “Camera Models,” in *CS231A Course Notes*, Stanford, 2021, pp. 1–17.

[17] A. Wang, T. Qiu, and L. Shao, “A Simple Method of Radial Distortion Correction with Centre of Distortion Estimation,” *J Math Imaging Vis*, vol. 35, pp. 165–172, 2009, doi: 10.1007/s10851-009-0162-1.

[18] E. Hecht, *Optics / Eugene Hecht.* 2017.

[19] F. E. Washer, “Prism Effect, Camera Tipping, and Tangential Distortion,” Washington D.C., Sep. 1957.

[20] J. Wang, F. Shi, J. Zhang, and Y. Liu, “A new calibration model of camera lens distortion,” *Pattern Recognition*, vol. 41, no. 2, Feb. 2008, doi: 10.1016/j.patcog.2007.06.012.

[21] A. S. Makarov and M. V. Bolsunovskaya, “The 360° around view system for large vehicles, the methods of calibration and removal of barrel distortion for omnidirectional cameras,” *AIST*, 2016.

[22] F. AlQahtani, J. Banks, V. Chandran, and J. Zhang, “Detection and Tracking of Faces in 3D Using a Stereo Camera Arrangements,” *International Journal of Machine Learning and Computing*, vol. 9, no. 1, 2019, doi: 10.18178/ijmlc.2019.9.1.762.

[23] T. A. Clarke and J. G. Fryer, “The Development of Camera Calibration Methods and Models,” *The Photogrammetric Record*, vol. 16, no. 91, Apr. 1998, doi: 10.1111/0031-868X.00113.

[24] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, 2000, doi: 10.1109/34.888718.

[25] OpenCV, “Camera Calibration,” *OpenCV*, Apr. 02, 2021.

[26] K. Sadekar and S. Mallick, “Camera Calibration using OpenCV,” *Learn OpenCV*, Feb. 25, 2020.

[27] A. Zaarane, I. Slimani, W. al Okaishi, I. Atouf, and A. Hamdoun, “Distance measurement system for autonomous vehicles using stereo camera,” *Array*, vol. 5, Mar. 2020, doi: 10.1016/j.array.2020.100016.

[28] C. C. T. Mendes and D. F. Wolf, “Stereo-Based Autonomous Navigation and Obstacle Avoidance*,” *IFAC Proceedings Volumes*, vol. 46, no. 10, Jun. 2013, doi: 10.3182/20130626-3-AU-2035.00045.

[29] Y. Kim, J. Jeong, and A. Kim, “Stereo Camera Localization in 3D LiDAR Maps,” Oct. 2018. doi: 10.1109/IROS.2018.8594362.

[30] A. Batthi, *Stereo Vision*. InTech, 2008. doi: 10.5772/89.

[31] Y. D. Salman, K. R. Ku-mahamud, and E. Kamioka, “Distance Measurement for Self-Driving Cars Using Stereo Camera,” *Proceeding of the 6Th International Conference of Computing & Informations*, no. 105, 2017.

[32] T. S. Hsu and T. C. Wang, “An improvement stereo vision images processing for object distance measurement,” *International Journal of Automation and Smart Technology*, vol. 5, no. 2, 2015, doi: 10.5875/ausmt.v5i2.460.

[33] A. Naveen and N. Bandaru, “Obstacle detection using stereo vision for self-driving cars,” *Stanford*, 2016.

[34] H. Fradi and J.-L. Dugelay, “Improved depth map estimation in stereo vision,” Feb. 2011. doi: 10.1117/12.872544.

[35] M. Pavlic, “Kamerabasierte Nebeldetektion und Sichtweitenschätzung im Fahrzeug,” Munich, 2014.

[36] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, “The ApolloScape Open Dataset for Autonomous Driving and Its Application,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, Oct. 2020, doi: 10.1109/TPAMI.2019.2926463.

[37] M. H. Rowe, “Trichromatic Color Vision in Primates,” *Physiology*, vol. 17, no. 3, Jun. 2002, doi: 10.1152/nips.01376.2001.

[38] H. Zhou and H. Wang, “Vision-based lane detection and tracking for driver assistance systems: A survey,” Nov. 2017. doi: 10.1109/ICCIS.2017.8274856.

[39] E. Akbari Sekehravani, E. Babulak, and M. Masoodi, “Implementing canny edge detection algorithm for noisy image,” *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 4, Aug. 2020, doi: 10.11591/eei.v9i4.1837.

[40] E. S. Gedraite and M. Hadad, “Investigation on the effect of a Gaussian Blur in image filtering and segmentation,” 2011.

[41] J. Canny, “A Computational Approach to Edge Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, Nov. 1986, doi: 10.1109/TPAMI.1986.4767851.

[42] X. Yan and Y. Li, “A method of lane edge detection based on Canny algorithm,” Oct. 2017. doi: 10.1109/CAC.2017.8243122.

[43] M. Joshi and A. Vyas, “Comparison of Canny edge detector with Sobel and Prewitt edge detector using different image formats,” *International Journal of Engineering Research & Technology*, no. 1, 2014.

[44] W. Rong, Z. Li, W. Zhang, and L. Sun, “An improved Canny edge detection algorithm,” Aug. 2014. doi: 10.1109/ICMA.2014.6885761.

[45] D. Kaur and Y. Kaur, “Various Image Segmentation Techniques: A Review,” *International Journal of Computer Science and Mobile Computing*, vol. 3, no. 5, 2014.

[46] Z. Zhang, S. Fidler, and R. Urtasun, “Instance-level segmentation for autonomous driving with deep densely connected MRFs,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-December. doi: 10.1109/CVPR.2016.79.

[47] Ç. Kaymak and A. Uçar, “A brief survey and an application of semantic image segmentation for autonomous driving,” in *Smart Innovation, Systems and Technologies*, vol. 136, 2019. doi: 10.1007/978-3-030-11479-4_9.

[48] G. Neuhold, T. Ollmann, S. R. Bulo, and P. Kontschieder, “The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, vol. 2017-October. doi: 10.1109/ICCV.2017.534.

[49] M. Treml *et al.*, “Speeding up Semantic Segmentation for Autonomous Driving,” *NIPS 2016 workshop MLITS*, no. Nips, 2016.

[50] S. Raut, M. Raghuvanshi, R. Dharaskar, and A. Raut, “Image segmentation - A state-of-art survey for prediction,” 2009. doi: 10.1109/ICACC.2009.78.

[51] D. Ortego, J. C. Sanmiguel, and J. M. Martinez, “Hierarchical Improvement of Foreground Segmentation Masks in Background Subtraction,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 6, Jun. 2019, doi: 10.1109/TCSVT.2018.2851440.

[52] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 1999, doi: 10.1109/cvpr.1999.784637.

[53] A. M. Hafiz and G. M. Bhat, “A survey on instance segmentation: state of the art,” *International Journal of Multimedia Information Retrieval*, vol. 9, no. 3, 2020, doi: 10.1007/s13735-020-00195-x.

[54] B. de Brabandere, D. Neven, and L. van Gool, “Semantic Instance Segmentation for Autonomous Driving,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2017, vol. 2017-July. doi: 10.1109/CVPRW.2017.66.

[55] M. N. Smith, “The number of cars worldwide is set to double by 2040,” *World Economic Forum*, Apr. 2016. https://www.weforum.org/agenda/2016/04/the-number-of-cars-worldwide-is-set-to-double-by-2040 (accessed Aug. 05, 2021).

[56] J. Z. Varghese and R. G. Boone, “Overview of Autonomous Vehicle Sensors and Systems,” *International Conference on Operations Excellence and Service Engineering*, 2015.

[57] J. R. Reagan and M. Singh, “Automotive Evolution,” in *Management 4.0: Cases and Methods for the 4th Industrial Revolution*, Singapore: Springer Singapore, 2020. doi: 10.1007/978-981-15-6751-3_2.

[58] AAA, “Advanced Driver Assistance Technology Names,” *American Automobile Association*, no. January, 2019.

[59] K. Kovačić, E. Ivanjko, and H. Gold, “Computer Vision Systems in Road Vehicles: A Review,” 2014. doi: 10.20532/ccvw.2013.0002.

[60] J. van Brummelen, M. O’Brien, D. Gruyer, and H. Najjaran, “Autonomous vehicle perception: The technology of today and tomorrow,” *Transportation Research Part C: Emerging Technologies*, vol. 89, Apr. 2018, doi: 10.1016/j.trc.2018.02.012.

[61] Y. Li and J. Ibanez-Guzman, “Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems,” *IEEE Signal Processing Magazine*, vol. 37, no. 4, Jul. 2020, doi: 10.1109/MSP.2020.2973615.

[62] M. E. Warren, “Automotive LIDAR Technology,” Jun. 2019. doi: 10.23919/VLSIC.2019.8777993.

[63] M. Kutila, P. Pyykonen, W. Ritter, O. Sawade, and B. Schaufele, “Automotive LIDAR sensor development scenarios for harsh weather conditions,” Nov. 2016. doi: 10.1109/ITSC.2016.7795565.

[64] B. Pradhan and M. Ibrahim Sameen, *Laser Scanning Systems in Highway and Safety Assessment*, vol. 7. 2020.

[65] G. Vosselman and H.-G. Maas, *Airborne and Terrestrial Laser Scanning*. 2010.

[66] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, “1 year, 1000 km: The Oxford RobotCar dataset,” *The International Journal of Robotics Research*, vol. 36, no. 1, Jan. 2017, doi: 10.1177/0278364916679498.

[67] P. Radecki, M. Campbell, and K. Matzen, “All Weather Perception: Joint Data Association, Tracking, and Classification for Autonomous Ground Vehicles,” May 2016, Accessed: Aug. 05, 2021. [Online]. Available: https://arxiv.org/abs/1605.02196

[68] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets Robotics: The KITTI Dataset,” *International Journal of Robotics Research (IJRR)*, 2013.

[69] X. Zhang, M. Zhou, P. Qiu, Y. Huang, and J. Li, “Radar and vision fusion for the real-time obstacle detection and identification,” *Industrial Robot*, vol. 46, no. 3, 2019, doi: 10.1108/IR-06-2018-0113.

[70] E. Ward and J. Folkesson, “Vehicle localization with low cost radar sensors,” Jun. 2016. doi: 10.1109/IVS.2016.7535489.

[71] S. M. Patole, M. Torlak, D. Wang, and M. Ali, “Automotive radars: A review of signal processing techniques,” *IEEE Signal Processing Magazine*, vol. 34, no. 2, Mar. 2017, doi: 10.1109/MSP.2016.2628914.

[72] S. J. Babak, S. A. Hussain, B. Karakas, and S. Cetin, “Control of autonomous ground vehicles: A brief technical review,” in *IOP Conference Series: Materials Science and Engineering*, 2017, vol. 224, no. 1. doi: 10.1088/1757-899X/224/1/012029.

[73] B. S. Lim, S. L. Keoh, and V. L. L. Thing, “Autonomous vehicle ultrasonic sensor vulnerability and impact assessment,” Feb. 2018. doi: 10.1109/WF-IoT.2018.8355132.

[74] A. Carullo and M. Parvis, “An ultrasonic sensor for distance measurement in automotive applications,” *IEEE Sensors Journal*, vol. 1, no. 2, 2001, doi: 10.1109/JSEN.2001.936931.

[75] E. N. Budisusila, B. Arifin, S. A. D. Prasetyowati, B. Y. Suprapto, and Z. Nawawi, “Artificial Neural Network Algorithm for Autonomous Vehicle Ultrasonic Multi-Sensor System,” Aug. 2020. doi: 10.1109/EECCIS49483.2020.9263459.

[76] X. Guang, Y. Gao, H. Leung, P. Liu, and G. Li, “An Autonomous Vehicle Navigation System Based on Inertial and Visual Sensors,” *Sensors*, vol. 18, no. 9, Sep. 2018, doi: 10.3390/s18092952.

[77] S. Sukkarieh, E. M. Nebot, and H. F. Durrant-Whyte, “A high integrity IMU/GPS navigation loop for autonomous land vehicle applications,” *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, Jun. 1999, doi: 10.1109/70.768189.

[78] L. Wonniger, “Introduction to Network RTK,” *WaSoft*, Jun. 16, 2008. http://www.wasoft.de/e/iagwg451/intro/introduction.html (accessed Aug. 07, 2021).

[79] X. Xia *et al.*, “Estimation on IMU yaw misalignment by fusing information of automotive onboard sensors,” *Mechanical Systems and Signal Processing*, vol. 162, Jan. 2022, doi: 10.1016/j.ymssp.2021.107993.

[80] E. D. Dickmanns and B. D. Mysliwetz, “Recursive 3-D road and relative ego-state recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, 1992, doi: 10.1109/34.121789.

[81] T. Möller, A. Padhi, D. Pinner, and A. Tschiesner, “The future of mobility is at our doorstep,” *McKinsey & Company*, Dec. 2019, Accessed: Aug. 08, 2021. [Online]. Available:

https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/the-future-of-mobility-is-at-our-doorstep#

[82] "Autonomous Driving," *Roland Berger*, 2015. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwi4sPHJg6HyAhWLgf0HHZkNAiMQFnoECAYQAw&url=https%3A%2F%2Fwww.rolandberger.com%2Fpublications%2Fpublication_pdf%2Froland_berger_tab_autonomous_driving.pdf&usg=AOvVaw2xn2HXndjG9ZbBMmTU1WkK (accessed Aug. 08, 2021).

[83] D. Feng *et al.*, "Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, Mar. 2021, doi: 10.1109/TITS.2020.2972974.

[84] Q.-H. Pham *et al.*, "A*3D Dataset: Towards Autonomous Driving in Challenging Environments," May 2020. doi: 10.1109/ICRA40945.2020.9197385.

[85] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The ApolloScape Open Dataset for Autonomous Driving and Its Application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, Oct. 2020, doi: 10.1109/TPAMI.2019.2926463.

[86] P. Sun *et al.*, "Scalability in Perception for Autonomous Driving: Waymo Open Dataset," Dec. 2019, Accessed: Aug. 08, 2021. [Online]. Available: https://arxiv.org/abs/1912.04838

[87] M. Cordts *et al.*, "The Cityscapes Dataset for Semantic Urban Scene Understanding," Apr. 2016. [Online]. Available: www.cityscapes-dataset.net

[88] A. Patil, S. Malla, H. Gang, and Y. T. Chen, "The H3D dataset for full-surround 3D multi-object detection and tracking in crowded urban scenes," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2019, vol. 2019-May. doi: 10.1109/ICRA.2019.8793925.

[89] Y. Choi *et al.*, "KAIST Multi-Spectral Day/Night Data Set for Autonomous and Assisted Driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, Mar. 2018, doi: 10.1109/TITS.2018.2791533.

[90] H. Caesar *et al.*, "nuScenes: A multimodal dataset for autonomous driving," Mar. 2019, Accessed: Aug. 08, 2021. [Online]. Available: https://arxiv.org/abs/1903.11027

[91] D. I. P. Barnes, M. Gadd, P. Murcutt, P. Newman, and I. Posner, "The Oxford Radar RobotCar Dataset: A Radar Extension to the Oxford RobotCar Dataset," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020, Accessed: Aug. 05, 2021. [Online]. Available: https://arxiv.org/abs/1909.01300

[92] N. Jayaweera, N. Rajatheva, and M. Latva-aho, "Autonomous Driving without a Burden: View from Outside with Elevated LiDAR," Apr. 2019. doi: 10.1109/VTCSpring.2019.8746507.

[93] R. J. Harrington, C. Senatore, J. M. Scanlon, and R. M. Yee, "The role of infrastructure in an automated vehicle future," *Bridge*, vol. 48, no. 2, 2018.

[94] B. Salesky, "A Decade after DARPA: Our View on the State of the Art in Self-Driving Cars," *Medium - Self Driven*, 2017.

[95] A. Bacha *et al.*, “Odin: Team VictorTango’s entry in the DARPA Urban Challenge,” *Journal of Field Robotics*, vol. 25, no. 8, 2008, doi: 10.1002/rob.20248.

[96] A. Bar Hillel, R. Lerner, D. Levi, and G. Raz, “Recent progress in road and lane detection: A survey,” *Machine Vision and Applications*, vol. 25, no. 3. 2014. doi: 10.1007/s00138-011-0404-2.

[97] R. O. Duda and P. E. Hart, “Use of the Hough Transformation to Detect Lines and Curves in Pictures,” *Communications of the ACM*, vol. 15, no. 1, 1972, doi: 10.1145/361237.361242.

[98] Z. W. Kim, “Robust lane detection and tracking in challenging scenarios,” in *IEEE Transactions on Intelligent Transportation Systems*, 2008, vol. 9, no. 1. doi: 10.1109/TITS.2007.908582.

[99] F. Xu, L. Chen, J. Lou, and M. Ren, “A real-time road detection method based on reorganized lidar data,” *PLoS ONE*, vol. 14, no. 4, 2019, doi: 10.1371/journal.pone.0215159.

[100] S. Kammel and B. Pitzer, “Lidar-based lane marker detection and mapping,” 2008. doi: 10.1109/IVS.2008.4621318.

[101] L. B. Cremean and R. M. Murray, “Model-based estimation of off-highway road geometry using single-axis LADAR and inertial sensing,” 2006. doi: 10.1109/ROBOT.2006.1641945.

[102] J. Yao, S. Ramalingam, Y. Taguchi, Y. Miki, and R. Urtasun, “Estimating drivable collision-free space from monocular video,” 2015. doi: 10.1109/WACV.2015.62.

[103] A. Elfes, “Sonar-based real-world mapping and navigation,” *IEEE Journal on Robotics and Automation*, vol. 3, no. 3, Jun. 1987, doi: 10.1109/JRA.1987.1087096.

[104] R. JIANG, R. KLETTE, T. VAUDREY, and S. WANG, “CORRIDOR DETECTION AND TRACKING FOR VISION-BASED DRIVER ASSISTANCE SYSTEM,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 25, no. 02, Mar. 2011, doi: 10.1142/S0218001411008567.

[105] H. Badino, R. Mester, T. Vaudrey, and U. Franke, “Stereo-based free space computation in complex traffic scenarios,” 2008. doi: 10.1109/SSIAI.2008.4512317.

[106] N. Soquet and M. Perrollaz, “Free space estimation for autonomous navigation,” *International Conference on Computer Vision Systems*, no. Icvs, 2007.

[107] F. Diego, J. M. Álvarez, J. Serrat, and A. M. López, “Vision-based road detection via on-line video registration,” 2010. doi: 10.1109/ITSC.2010.5624998.

[108] A. Ziebinski, R. Cupek, H. Erdogan, and S. Waechter, “A Survey of ADAS Technologies for the Future Perspective of Sensor Fusion,” 2016. doi: 10.1007/978-3-319-45246-3_13.

[109] A. Theofilatos and G. Yannis, “A review of the effect of traffic and weather characteristics on road safety,” *Accident Analysis & Prevention*, vol. 72, Nov. 2014, doi: 10.1016/j.aap.2014.06.017.

[110] D. Pomerleau, “Visibility estimation from a moving vehicle using the RALPH vision system,” 1997. doi: 10.1109/ITSC.1997.660594.

[111] R. Gallen, A. Cord, N. Hautiere, E. Dumont, and D. Aubert, “Nighttime Visibility Analysis and Estimation Method in the Presence of Dense Fog,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, Feb. 2015, doi: 10.1109/TITS.2014.2331177.

[112] N. Hautière, J.-P. Tarel, H. Halmaoui, R. Brémond, and D. Aubert, “Enhanced fog detection and free-space segmentation for car navigation,” *Machine Vision and Applications*, vol. 25, no. 3, Apr. 2014, doi: 10.1007/s00138-011-0383-3.

[113] J.-P. Tarel, N. Hautiere, A. Cord, D. Gruyer, and H. Halmaoui, “Improved visibility of road scene images under heterogeneous fog,” Jun. 2010. doi: 10.1109/IVS.2010.5548128.

[114] M. Hadj-Bachir and P. de Souza, “LIDAR sensor simulation in adverse weather condition for driving assistance development,” *HAL Archives - Ouvertes*, no. December, 2019.

[115] C. Michaelis *et al.*, “Benchmarking Robustness in Object Detection: Autonomous Driving when Winter is Coming,” Jul. 2019, Accessed: Aug. 14, 2021. [Online]. Available: https://arxiv.org/abs/1907.07484

[116] A. Wedel, H. Badino, C. Rabe, H. Loose, U. Franke, and D. Cremers, “B-spline modeling of road surfaces with an application to free-space estimation,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 4, 2009, doi: 10.1109/TITS.2009.2027223.

[117] K. Y. Lee, G. Y. Song, J. M. Park, and J. W. Lee, “Stereo vision enabling fast estimation of free space on traffic roads for autonomous navigation,” *International Journal of Automotive Technology*, vol. 16, no. 1, 2015, doi: 10.1007/s12239-015-0012-7.

[118] G. Avi, M. Devy, and A. Mar, “Lane Extraction and Tracking for Robot Navigation in Agricultural Applications,” *Advanced Robotics*, 2003.

[119] N. Kemsaram, A. Das, and G. Dubbelman, “An Integrated Framework for Autonomous Driving: Object Detection, Lane Detection, and Free Space Detection,” Jul. 2019. doi: 10.1109/WorldS4.2019.8904020.

[120] S. Krome *et al.*, “Workshop on Navigating Autonomous Cars,” Sep. 2017. doi: 10.1145/3131726.3131731.

[121] F. Ulbrich, S. S. Rotter, and R. Rojas, “Adapting to the Traffic Swarm,” in *Robotic Systems*, IGI Global, 2020. doi: 10.4018/978-1-7998-1754-3.ch067.

[122] K. Jo and M. Sunwoo, “Generation of a Precise Roadway Map for Autonomous Cars,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 3, Jun. 2014, doi: 10.1109/TITS.2013.2291395.

[123] A2D2, “Tutorial.” https://www.a2d2.audi/a2d2/en/tutorial.html (accessed Aug. 17, 2021).

[124] “ApolloScape.” http://apolloscape.auto/stereo.html (accessed Aug. 13, 2021).

[125] X. Huang *et al.*, “The ApolloScape Dataset for Autonomous Driving,” Jun. 2018. doi: 10.1109/CVPRW.2018.00141.

[126] J. Son, H. Yoo, S. Kim, and K. Sohn, “Real-time illumination invariant lane detection for lane departure warning system,” *Expert Systems with Applications*, vol. 42, no. 4, Mar. 2015, doi: 10.1016/j.eswa.2014.10.024.

[127] R. Chapuis, F. Marmoiton, R. Aufrere, F. Collange, and J. Derutin, “Road detection and vehicles tracking by vision for an on-board ACC system in the VELAC vehicle,” 2000. doi: 10.1109/IFIC.2000.859833.

[128] F. Girstl, “Selbstfahrende Rennfahrzeuge: Detektion von Fahrspuren und Spurvermessung,” Munich, 2018.

[129] L. Chen, J. Yang, and H. Kong, “Lidar-histogram for fast road and obstacle detection,” May 2017. doi: 10.1109/ICRA.2017.7989159.

[130] C. Godard, O. mac Aodha, M. Firman, and G. Brostow, “Digging into self-supervised monocular depth estimation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, vol. 2019-October. doi: 10.1109/ICCV.2019.00393.

[131] C. Godard, O. mac Aodha, and G. J. Brostow, “Unsupervised monocular depth estimation with left-right consistency,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, vol. 2017-January. doi: 10.1109/CVPR.2017.699.

[132] G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett, “Deep Learning Techniques for Inverse Problems in Imaging,” *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, 2020, doi: 10.1109/jsait.2020.2991563.

[133] L. Adenaw, J. Kreibich, M. Wittmann, L. Merkle, A. Waclaw, and M. Lienkamp, “MAGIS – A Geographic Information System for Mobility Data Analysis,” Oct. 2019. doi: 10.1109/ITSC.2019.8917054.

[134] “OpenStreetMap.” www.openstreetmap.org/copyright (accessed Aug. 31, 2021).

[135] “Road traffic, vehicles and networks,” in *Environment at a Glance 2013: OECD Indicators*, Paris: OECD Publishing, 2013. doi: 10.1787/9789264185715-20-en.

[136] A. E. Retallack and B. Ostendorf, “Current Understanding of the Effects of Congestion on Traffic Accidents,” *International Journal of Environmental Research and Public Health*, vol. 16, no. 18, Sep. 2019, doi: 10.3390/ijerph16183400.

[137] S. A. AbdelHakeem, A. A. Hady, and H. Kim, “Optimizing 5G in V2X Communications,” in *Research Anthology on Developing and Optimizing 5G Networks and the Impact on Society*, IGI Global, 2021. doi: 10.4018/978-1-7998-7708-0.ch041.

[138] R. Prophet, G. Li, C. Sturm, and M. Vossiek, “Semantic Segmentation on Automotive Radar Maps,” Jun. 2019. doi: 10.1109/IVS.2019.8813808.

Appendix

A Content of Memory Card

Table A.1: Folder Structure of Memory Card

Folder	Content
1_Bachelor_Thesis	Bachelor Thesis in Word and PDF format and all used images and visualizations.
2_Code	Python scripts used in the analysis: evaluation.py and visualizations.py along with utils scripts containing helpful functions.
3_Literature	All the used sources for this work in PDF format.

B Code

1. Python Implementations

Code B.1: Preparation of LIDAR data from different datasets

```
def prepare_data(depth_filename, calib, dataset, img_nbr=None):

    #lidar_points = nx4 (x, y, z, reflectance)
    if dataset == "KITTI":
        lidar = load_velo_scan(depth_filename)
        img_filename =
extract_image_file_name_from_depth_file_name(depth_filename,
dataset="KITTI")
        img = load_image(img_filename, dataset)
        img_height = img.shape[0]
        img_width = img.shape[1]
        points = lidar[:, 0:3]
        reflectance = lidar[:, 3]

        road_points = points[points[:, 2] < -1] #since the LIDAR
scanner sits at an elevation of 1.73m above ground
        road_points = road_points[np.abs(road_points[:, 1]) <
np.percentile(np.abs(road_points[:, 1]), 80)] # we are interested in
the corridor in front of the car, not the region to the sides
        depth = points[:, 0]
        road_depth = road_points[:, 0]
        distance = np.linalg.norm(points, axis=1)
        road_distance = np.linalg.norm(road_points, axis=1)

        # projection matrix (project from velo2cam2)
        proj_velo2cam2 = project_velo_to_cam2(calib)
        pixel_coord = project_to_image(points.transpose(),
proj_velo2cam2)
        road_pixel_coord = project_to_image(road_points.transpose(),
proj_velo2cam2)

        # Filter out the Lidar Points on the image
        inds = np.where((pixel_coord[0, :] < img_width) &
(pixel_coord[0, :] >= 0) & (pixel_coord[1, :] < img_height) &
(pixel_coord[1, :] >= 0) & (points[:, 0] > 0))[0]

        road_inds = np.where((road_pixel_coord[0, :] < img_width) &
(road_pixel_coord[0, :] >= 0) & (road_pixel_coord[1, :] <
img_height) & (road_pixel_coord[1, :] >= 0) &(road_points[:, 0] >
0))[0]

        pixel_coord = pixel_coord[:, inds]
        pixel_coord = np.transpose(pixel_coord)
        road_pixel_coord = road_pixel_coord[:, road_inds]
        road_pixel_coord = np.transpose(road_pixel_coord)
        points = points[inds, :]
        road_points = road_points[road_inds, :]
        depth = depth[inds]
        road_depth = road_depth[road_inds]
        distance = distance[inds]
        road_distance = road_distance[road_inds]
        reflectance = reflectance[inds]
        row = pixel_coord[:, 1]
        road_row = road_pixel_coord[:, 1]
        col = pixel_coord[:, 0]
        road_col = road_pixel_coord[:, 0]

```

```
58
59          lidar_dict = {'points': road_points, 'reflectance':
60  reflectance, 'row': road_row, 'col': road_col, 'depth': road_depth,
61  'distance': road_distance, 'all_points': points, 'all_row': row,
62  'all_col': col, 'all_depth': depth, 'all_distance': distance}
63
64          return lidar_dict
65
66
67      elif dataset == "Oxford":
68          # lidar: XYZI pointcloud from the binary Velodyne
69  data Nx4
70          date = depth_filename.split('/')[6]
71          lidar_dir = depth_filename.split('/')
72          lidar_dir = lidar_dir[:len(lidar_dir)-1]
73          lidar_dir = '/'.join(lidar_dir)
74          pose_file =
75  extract_ins_folder_name_from_depth_file_name(depth_filename)
76          extrinsic = depth_filename.split('/')
77          extrinsic = extrinsic[:len(extrinsic)-4]
78          extrinsic = '/'.join(extrinsic)
79          extrinsic = extrinsic + '/extrinsics'
80          time = depth_filename.split('.')[0]
81          time = time.split('/')[-1]
82          time = np.int(time)
83          img_dir = depth_filename.split('/')
84          img_dir = img_dir[:len(img_dir)-4]
85          img_dir = '/'.join(img_dir)
86          img_dir = img_dir + '/Camera/' + date + '/stereo/centre'
87          models = "/Volumes/Extreme SSD/Bachelorarbeit/Oxford/camera-
88  models"
89          points, reflectance = build_pointcloud(lidar_dir, pose_file,
90  extrinsic, start_time=time-1e7, end_time=time+1e7, origin_time=time)
91
92          points = np.array(points)
93          points = points[:3, :]
94          points = np.transpose(points)
95          points = points[points[:, 2] > -2]
96          points = points[np.abs(points[:, 1]) <
97  np.percentile(np.abs(points[:, 1]), 5)]
98
99          #get pixels from matrix in project laser into camera
100         pixels, depth = project_laser_into_camera(img_dir,
101 lidar_dir, pose_file, models, extrinsic, img_nbr, show_ptcld=False,
102 road=True)
103         pixels = np.array(pixels)
104         row = pixels[1, :]
105         col = pixels[0, :]
106         distance = np.linalg.norm(points, axis=1)
107
108         lidar_dict = {'points': points, 'reflectance': reflectance,
109 'row': row, 'col': col, 'depth': depth, 'distance': distance}
110
111         return lidar_dict
```

Code B.2: Calculation of road distances for the ApolloScape dataset

```
1   def calculate_road_distance(semantic_img, depth_img, dataset):
2       if dataset == "Apolloscape_stereo":
3           # Intrinsic Camera Parameters
4           baseline = 0.36  # baseline is the distance between the 2
5   stereo cameras
6           f = 2301.3147  # focal length in pixels
7
8           road_pixels = get_road_pixels(semantic_img, depth_img,
9   dataset)
10          u = road_pixels[:, 0]
```

```
            v = road_pixels[:, 1]
            distance = []
            depth = []
            for i in range(len(u) - 1):
                z = ((f * baseline) / depth_img[u[i], v[i]])
                x = (z * u[i])/f
                y = (z * v[i])/f
                dist = np.sqrt(z**2 + x**2 + y**2)
                depth.append(z)
                distance.append(dist)
            depth = np.asarray(depth)
            distance = np.asarray(distance)

            return depth, distance, road_pixels

        elif dataset=="Apolloscape_semantic":
            road_pixels = get_road_pixels(semantic_img, depth_img,
dataset)
            u = road_pixels[:, 0]
            v = road_pixels[:, 1]
            distance = []
            for i in range(len(u)-1):
                dist = 255.0*(depth_img[u[i], v[i]]/200.0)
                distance.append(dist)

            return 0, distance, road_pixels
```

Code B.3: Overlays the detected road on top of the corresponding RGB image

```
def overlay_road_on_image(img, dataset):

    output = img
    roi = roi_r(img, dataset)
    roi = gauss(roi, dataset)
    canny_edge = canny(roi)
    white_line = filter_white_hls_binary(roi)
    yellow_line = filter_yellow_hls_binary(roi)
    semantic_image = det_line_1lane_init(canny_edge, white_line,
yellow_line, dataset)

    if dataset=="KITTI":
        roi_factor_y = 0.5
        roi_factor_x = 0.3
    elif dataset == "Apolloscape_stereo":
        roi_factor_y = 0.5
        roi_factor_x = 0.3
    elif dataset=="Oxford":
        roi_factor_y = 0.45
        roi_factor_x = 0.3

    factor_y = int(roi_factor_y * img.shape[0])
    factor_x = int(roi_factor_x * img.shape[1])

    output[factor_y:, factor_x:output.shape[1]-factor_x, :] = 0.6 *
img[factor_y:, factor_x:img.shape[1]-factor_x, :] + 0.4 *
semantic_image

    return output
```

Code B.4: Obtain the road category and type from GPS coordinates

```
def get_road_category(longitude, latitude):

    url = "http://gis.ftm.mw.tum.de/reverse?coordinates=[" +
str(longitude) + "," + str(latitude) + "]"
    response = urlopen(url)
    data = json.loads(response.read())

    category = data['features'][0]['properties']['category']
    location = data['features'][0]['properties'] ['display_name']
    type = data['features'][0]['properties']["type"]

    return category, location, type
```

Code B.5: Function to detect blocking factors in semantic images.

```
def blocking(semantic_image, img, dataset):
    l, r = get_road_boundaries(semantic_image, img, dataset)
    l = np.array(sorted(l, key=lambda x: x[0], reverse=True))
    r = np.array(sorted(r, key=lambda x: x[0], reverse=True))
    bin_cars = filter_cars(semantic_image, dataset)
    bin_trucks = filter_trucks(semantic_image, dataset)
    bin_ped_bc = filter_ped_bc(semantic_image, dataset)

    #Traverse image from bottom to top until blocking factor has
been reached
    for i in range(l.shape[0]):
        left = l[i, 1] # Interested in Ego Lane
        right = r[i, 1]
        road_width = right - left
        row = l[i, 0]
        car = np.sum(np.array(bin_cars[row, left:right] == 255))
        truck = np.sum(np.array(bin_trucks[row, left:right] == 255))
        ped = np.sum(np.array(bin_ped_bc[row, left:right] == 255))
        if road_width > 0:
            if dataset=="A2D2":
                if car/road_width > 0.5:
                    return [1, 0, 0], row
                    break
                elif truck/road_width > 0.5:
                    return [0, 1, 0], row
                    break
                elif ped/road_width > 0.5:
                    return [0, 0, 1], row
                    break
                else:
                    if i == l.shape[0] - 1:
                        return [0, 0, 0], 0, 0
                    else:
                        continue
            elif dataset=="Apolloscape_semantic":
                if car/road_width > 0.5:
                    return [1, 0, 0], row, int(road_width/2)
                    break
                elif truck/road_width > 0.5:
                    return [0, 1, 0], row, int(road_width/2)
                    break
                elif ped/road_width > 0.5:
                    return [0, 0, 1], row, int(road_width/2)
                    break
                else:
                    if i==l.shape[0]-1:
                        return [0, 0, 0], 0, 0
                    else:
                        continue
        else:
            continue
```

Table B.1: Used Python libraries

Library	Usage
NumPy	Mathematical functions for array-based calculations, filtering, and data storage.
pandas	Building and working with the dataframes containing the final results.
Open3D	Building and visualizing point clouds from raw LIDAR data.
OpenCV	Working with image data.
matplotlib	Working with image data and creating plots explaining the results.
seaborn	Creating plots.

2. Query for Road Type based on GPS Coordinates

- Establish a connection to the TUM network, by VPN for example.
- Access the API hosted by the FTM Chair at the following url: `http://gis.ftm.mw.tum.de/reverse?coordinates=[longitude, latitude]`, where longitude and latitude need to be provided in decimal degree format.
- The API uses the OSM database and returns the address of the input, the corresponding category and road type.