Challenges in Developing Software for Autonomous Driving

Bachelor's Thesis in Informatics
at the Department of Informatics of the Technical University of Munich.

Michael Grad

Challenges in Developing Software for Autonomous Driving

Herausforderungen in der Entwicklung von Software für autonomes Fahren

Bachelor's Thesis in Informatics

at the Department of Informatics of the Technical University of Munich.

Supervisor	Univ.-Prof. Dr. Thomas Huckle
Advisor	Dr. rer. nat. Tobias Neckel
Submitted by	Michael Grad
Submitted on	14.12.2018

Appendix I

Declaration

I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, 14.12.2018, Signature

Contents

1. Introduction

In this paper, various challenges and possible solutions will be explained and assessed. Starting at a general description and goal of autonomous driving, a form of formalizing the extent of automation in a vehicle is explained, namely the SAE Levels zero through five. After a short summary of the history of driver assistance systems that depicts the difference between multiple advanced driver assistance systems and an autonomous driving system is given, the technical structure is portrayed.

Delimiting the properties of an embedded system to regular computers, the computational network of ECUs in a vehicle is shown. Non functional requirements to the system are illustrated by comparing transformational to reactive systems, as well as event and time triggered systems.
In order to structure the software- (and some hardware-) components, a functional architecture view is followed, which was originally shown in [1]. The perception chapter features topics like:

Sensing, in which camera-, radar-, lidar- and ultrasonic-sensor are depicted along with their properties, as well as strengths and weaknesses. The sensing section will be finished by comparing the equipment of current vehicles.

In **Sensor Fusion** two architectures will be presented as well as some methods, of which the ”Kalman Filter” will be explained in more detail.

Localization shows a few methods that offer themselves to accurately determine the current position of the vehicle.

Semantic understanding will cover, how machine learning can be used to better understand the content of a scene, which can be seen as an ”interpretation” of the sensor data.

Two different kinds of **world models** will be differentiated as well as the role of ”Local Dynamic Maps” which end the perception part of the system.

In Decision and Control, two approaches for **trajectory generation** will be shown as well as a few other components that are necessary or useful to incorporate. Concluding the technical challenges, the soft- and hardware architecture of (autonomous) cars, including ECU communications with various bus systems and AUTOSAR, will be covered.
Throughout the chapter about technical challenges, there will be examples of cases where the autonomous system failed in a particular task. These accidents have to some part occurred in testing phases as well as in open beta programs of autonomous systems.

Chapter four will concentrate on the development process of software for autonomous vehicles and will pick up the standardized concept of AUTOSAR, where chapter three ended and

portrait its benefits for collaboration between companies, giving an exemplary list of firms that work on autonomous technology, that are not necessarily associated with the automotive industry. The assessment of the development process itself will cover separation of concerns, standardized demands for autonomous software, development strategies like the V-Model or Scrum and will finally give an insight into testing and quality management.

Finally a short coverage on the legal regulations of autonomous vehicles in different areas should be given, along with some ethical problems that arise when developing an autonomous system which need to be solved or in same way even have already been assessed.

2. Taxonomy of Autonomous vehicles

2.1. Definition of Autonomous Vehicles

A fully autonomous vehicle is defined to be capable of maneuvering itself from a given starting point to a destination under any roadway or environmental conditions, without any interaction by the driver. (Derived from [2]). This includes the vehicle controlling the longitudinal as well as the lateral movement of the vehicle but also autonomous solutions for when sensors are incapacitated or covered in dirt.
Dozens of companies like BMW, Audi, Volkswagen, Volvo, Daimler or Bosch are currently working on making autonomous vehicles ready to be suitable for the mass. However, there are also other players active in this particular area, which one might not immediately associate with the automotive industry. These include amongst others Waymo, a subsidiary of Alphabet, formerly known as the Google self-driving car project and the peer to peer ride sharing platform Uber. But also Apple, Samsung and Huawei have confirmed to be researching and developing technology for autonomous driving. (Infos regarding companies, See: [3] and [4])
One thing to keep in mind is, that autonomous driving is not exclusively comprised of self driving cars, but includes many other types of vehicles like buses, trucks or even motorbikes and (electric-) bicycles as is worked on by the MIT (See [5]). It is thus important to clarify which vehicle is point of subject in order to prevent misunderstandings. While this paper concentrates on cars, most concepts can be transfered to trucks, etc..

2.2. Levels of Autonomous Driving

Before looking at the different logical components of an autonomous car, it makes sense to first make yourself familiar with different degrees of automation in vehicles. Information in this section about different SAE Levels is taken from [2]. The standard SAE J3016 by the formerly known Society of Automotive Engineers (now: SAE) defines six automation levels, that have become a popular means of measure for classifying the capabilities of vehicles. The norm was designed in ”Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems”.
The scale begins with **SAE Level 0** which equivalences a vehicle with no automation except for rudimentary systems like an anti-lock braking system (ABS) or electric stability control (ESP). Subsequently the human driver is the instance perceiving the environment and performing all driving functions in every direction.
A vehicle on **SAE Level 1** however contains driver assistance for either lateral or longitudinal control. An example for this level includes adaptive cruise control systems that regulate the speed of a vehicle to conform to a given speed limit but also slow the vehicle down when a slower vehicle is detected or a driver ahead slows down. Thus it is an automation for longitu-

dinal acceleration. Driver assists for lateral movement often include lane departure warning systems which are sometimes capable of taking action into the steering of the car to keep it inside the lane boundaries when the driver is not paying attention.
A combination of lateral and longitudinal assistance is defined in **SAE Level 2**. Consequently on this level the driver is assisted by two systems, regulating both lateral and longitudinal movement of the vehicle. It differs form Level 1 in that both steering and acceleration is executed by the system whereas Level 1 is defined to only have one, lateral or longitudinal control automated. Still it is only considered an assistant instead of an automated driving system, since the driver must perform any other actions involved in the driving task. Still the system relies on the driver monitoring its actions and being able to take control over the vehicle at any point in time. Thus the driver must still be aware of the environment and situation around the vehicle.
From **SAE Level 3** onwards, the automated driving system itself monitors the environment to an extent, where there is no need for the driver to supervise its decisions. Nevertheless, this level still includes and relies on the driver as a fallback solution for when the system encounters situations which it can not resolve itself. Thus at this point the vehicle is not yet capable of maneuvering itself in every condition, but only in certain environments like highways or outside of cities.
SAE Level 4 extends the scenarios in which the vehicle can perform the driving tasks on its own, but also manages to continue when the driver does not respond to a take over request. This means that the vehicle has various methods of resolving situations which are not addressed by the human driver. These include overly complex traffic situations, new or unknown environments but also problems like incapacitated sensors due to impurities.
Fully autonomous driving, as defined in **SAE Level 5** is achieved, when the automated driving system can perform all driving tasks at any environmental condition as well as every kind of road and traffic situation, that also a human driver could cope with.

Besides SAE's definition as described above, there are further attempts to formalize the extent of automation in an automated vehicle, one of which was published by the German "Federal Road Research Institute", with the main difference being that Level 5 as described by the SAE is not included (See: [6]).

2.3. History and latest developments in the field

Information from this section are taken from [7].
From the SAE levels described above, it is noticeable that the scale depicts not only autonomous driving, but multiple extents of automation in a vehicle. Thus a car with an adaptive cruise control system is not commonly described as "autonomous" but rather "equipped with driver assistance". There might still remain the question, whether a vehicle with two Level 1 systems (regulating lateral and longitudinal movements) might be considered Level 5 right away and not only Level 2. In order to depict the most important differences between driver

assists and an autonomous driving system, it makes sense to look at the functionality of these system bottom up (from SAE Level 0 to 5).
Before 2003 most road cars were mainly equipped with driver assists like ABS or ESP. These systems were often of mechanical nature or were controlled by a small electronic control unit (ECU). However, since safety in traffic is a major concern, companies started developing advanced systems to protect both humans inside as well as around the vehicle. Thus assists like adaptive cruise control or lane departure assistants came to be. Most of the times however, these were modular systems that were designed to be implemented in vehicles independent of one another. This meant, that every assist had its own sensors which (in the worst case) implied a 1:1 relationship between sensors and assists.
Since vehicles are increasingly equipped with high tech features like such, it makes no longer sense to design systems that each come with their own sensors. Thus it has become popular to have the systems share the sensors built into the car for a more economic use. This means that every assist can choose to grab information from a subset of sensors the car is equipped with. Therein lies a shift from a 1:1 to a n:m relationship between sensors and system which use them.
Latest developments and research in autonomous driving however lead to another redesign of sensor usage in vehicles. Since systems now do not only fulfill one specific task, like lane keeping, a new overall approach was followed. Instead of systems accessing the sensors, the sensor-data is used to generate a world map, that depicts the situation around the vehicle. Data sources therefor might include, radar-, ultrasound- or lidar-sensors, cameras, high resolution maps, etc.. The there-out resulting world model is then source of data for all kinds of driving assists, but also a more overall image of the vehicle's surroundings, as is needed for autonomous driving systems that fall under SAE Levels 3 through 5.

3. Logical and Technical Components of Autonomous Vehicles

Embedded Systems

Source of information for embedded systems: [7]. Since each computational element of an autonomous vehicle is strictly fitted to handle one specific task, they differ from general purpose computers, as they can be classified as an ”Embedded System”. In order to be referred to as such, a computational system needs to fulfill various criteria. Thus an ”Embedded System” is often defined as the following. (Definition taken from [7]). Definition 3:

> ”A combination of computer hardware and software, designed to perform a certain, well defined function, often associated with a hard time limit, whereat the hardware represents the system-context (i.e. the physical environment) of the reactive software.”
>
> — [7]

Embedded Systems often follow a common chain of steps:

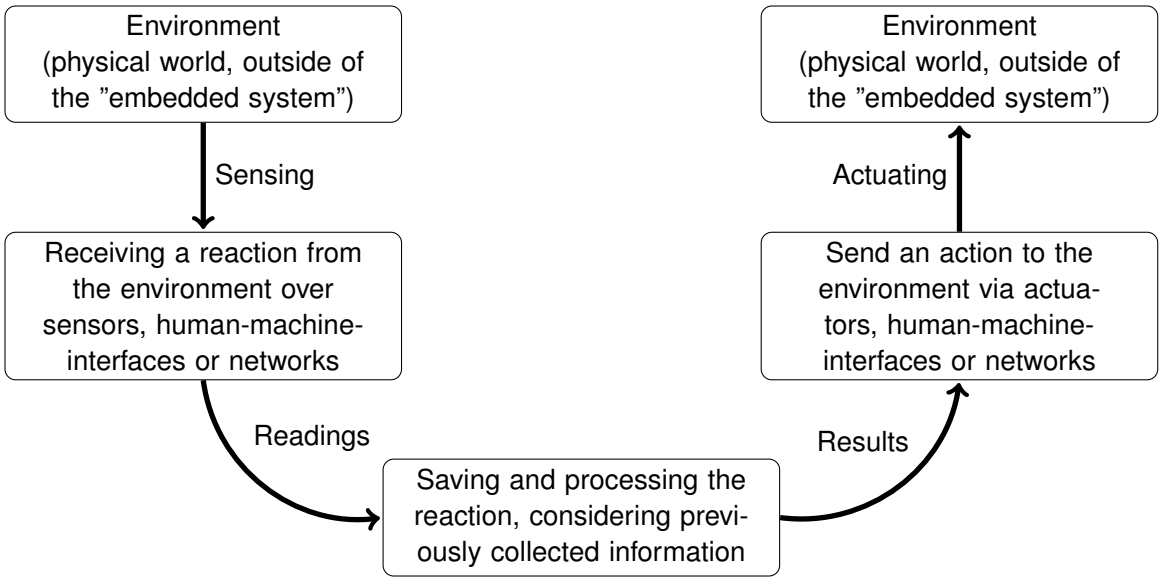

Figure 1 Functionality of an ”Embedded System”

Since nearly all cars consist of a big network of Electronic Control Units (ECUs), all specialized on covering a certain task, many of which need to be finished before a given point in time, one can say, that a modern car is a collection of multiple dozens of ”Embedded Systems”.

An airbag system for example includes acceleration sensors, some of which are located on the ECU of the airbag, others are spread throughout the vehicle to be able to detect impacts at multiple positions. Other components include the ECU itself, which stores the program that evaluates the sensor readings and decides whether or not the airbag should be deployed. When necessary, a nylon bag is inflated by a gas-generator. When assessing whether this system is an ”Embedded System” you can map these components to the ones described in definition 3. Thus the ECU equates to the computer hardware, but so do the acceleration sensors as well as the actuation unit, whereas the Software is running on the ECU. The system does in fact serve a certain, well defined function that under all circumstances should be executed within a certain timespan. The system context of the hardware is indeed represented in form of the actuator unit.
To continue the example above for the functionality of an ”Embedded System” as depicted in Figure 1, the acceleration sensors monitor the environment. These information are transmitted to the ECU which then evaluates these readings, maybe compares them to previous states to detect whether a hard impact has happened and might also save various variables for future calculations. Depending on the result of this step, the ECU might send a signal to the actuator unit to deploy, which then would equal to an intervention in the environment that might in some cases influence the values of the next sensor readings.

Transformational and reactive Systems

Informations from this section are taken from: [7]. Most computer programs are what is called a ”Transformational System”. Such a system takes a certain input data, process the input and terminates, delivering the desired output for the input data. However this concept, though easier to realize, is not the optimal solution for realizing a system that needs to constantly react to impulses from both the driver and the environment. That is why most of the time not transformational but reactive systems are used for tasks that need to regulate specific system parameters. The functionality of a reactive system can be depicted by altering Figure 1 slightly, which results in Figure 2:

The chart in Figure 2 illustrates an important aspect of a reactive system. It does not terminate, instead it runs in an infinite loop, processing sensor readings and user inputs, resulting in an output for both user and actors in the system. Other characteristics include: concurrency, failure resistant, propriety and an exact time behavior. Due to this trademarks, reactive systems are a popular choice for embedded-, real time-, communication-systems and many more.
Additionally, Figure 2 shows a driver, that can also provide input data, e.g. for steering or acceleration. In order to supply the driver with information about the environment, the system

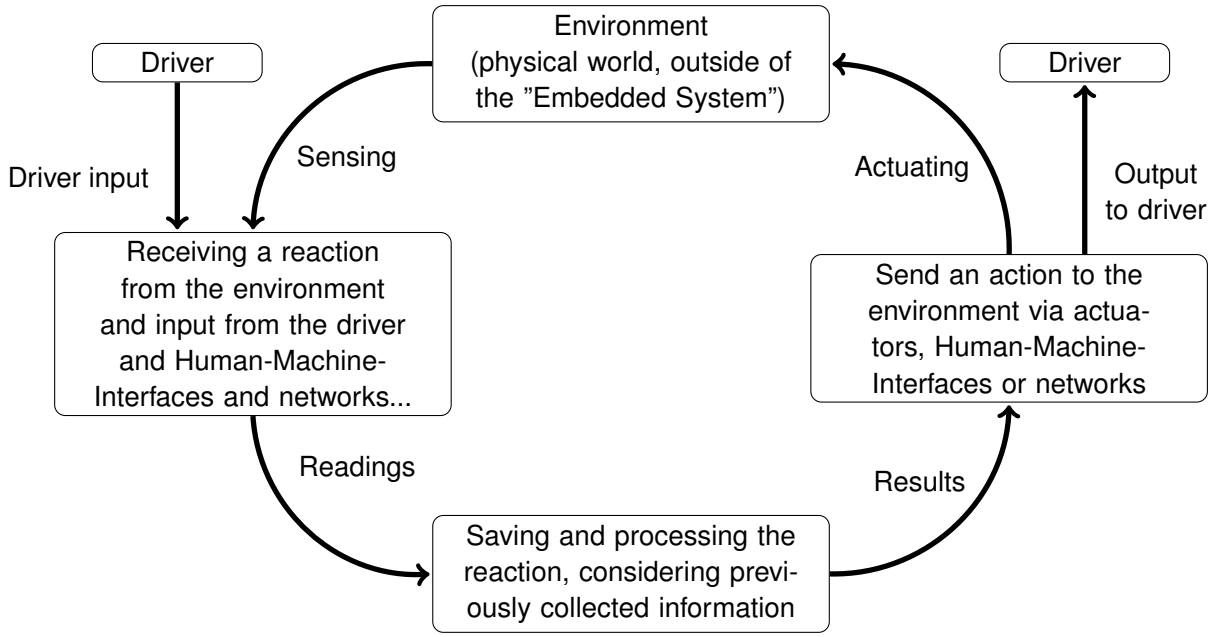

Figure 2 Functionality of a ”Reactive System”

gives an output to the driver as well. This might happen in form of a speedometer, haptic feedback via the steering wheel, etc..

Event- and time-triggered Systems

As timing is a critical factor of safety relevant software, there are two major approaches for architecture designs that have different effects on how reliable a system is concerning timing. When a task is particularly to have finished before a given point in time, you refer to the latest point in time when a result is relevant as its deadline. In order to guarantee that the time for a task stays within this bound, you might consider to choose an operating system that is time triggered rather than event triggered, like most general purpose operating systems.
Most operating systems that are not time critical are **Event triggered**. This means that they react to events like a mouse click or a sensor reading. When such an event takes place, an interrupt reaches the CPU of the system, causing it to start its routine for the specific situation, which means they are driven by interrupts. This approach is simple, works well and is widely used. However this method requires a relatively big overhead, as it is susceptible to fail when it is hit with a high load in a short time, thus when multiple interrupts occur shortly after another. Such a high load might lead to violations of the real time requirement, since the hardware might not be powerful enough to compute every interrupt in time. However some application need absolute certainty that the computer program has produced a result at a given time or else it might lead to situations that can endanger the people in and around the car. Since you can not guarantee the timing with event triggered systems, they are mostly used for use cases where soft real time is required, like window lifters or central locking.
Time triggered systems on the other hand can guarantee that a result is present at a given point in time, as these operating systems reserve resources for each process that might

need to be executed. Instead of event driven interrupts, a clock generates interrupts every x milliseconds. When a clock interrupt is registered, sensor data is checked and actors are talked to if needed. Obviously a well considered x is crucial for the functionality of the system, since it needs to be small enough to react to scenarios quickly, but also big enough to grant the system sufficient time to address problems when needed. Also you might choose to only consider specific sensors only every other or every n-th interrupt to save computational power. But then, again, you need to make sure that you do not observe the sensor too rarely. All in all, event triggered reacts faster but might fail with high loads, time triggered on the other hand reacts slower to events but can handle higher loads better. The choice of event triggered for soft real-time and time triggered for hard real time requirements seems optimal. However, the application dependent use of the two approaches is subject of controversy. Source of information of this section: [7]

Functional architecture view

In order to get a more detailed understanding of the logical (software-)components of an autonomous vehicle, it makes sense to categorize them into predefined predicaments. Since an autonomous vehicle as a whole acts like one reactive system, you can follow the loop from sensing to actuation and visit all functional components of the autonomous driving system. There are many ways to divide the many logical components into classes, however following the chain of events of one loop, it makes sense to have the three main categories: **Perception**, **Decision and Control** as well as **Vehicle Platform Manipulation**. This functional architecture view (FAV) was taken from Sagar Behere's and Marting Törngern's paper, called: A functional architecture for autonomous driving. (See: [1])

Perception, which is responsible of creating a model of the environment of the vehicle using sensors, fusing and interpreting their data as well as localize the vehicle as exactly as possible, thus they further divided perception into: sensing, sensor fusion, localization, semantic understanding and world model.

Decision and Control, by help of the world model, is designed to analyze the situation the vehicle is in, find the right way as well as a fitting trajectory (or rather a set of realistic trajectories to choose from) and derive necessary actions to realize this trajectory. Subcomponents of decision and control are: trajectory generation, energy management, diagnostics and fault management and reactive control.

Finally, **vehicle platform manipulation** is responsible to execute the actions that decision and control has decided are necessary. Furthermore it is unique to the vehicle and its kinematic models in order to make every component as reusable as possible, taking separation of concerns into account. It was subdivided into: platform stabilization, passive safety and trajectory execution.

It should be noted, that due to the nature of an engineered system, there might also exist architectures, that follow different strategies to archive the same goal. However, the functional architecture described by [1] has been chosen to represent a frame for the description of the technical and logical components, as it implements the widely used ”sense, plan, act”

philosophy, but also depicts a very intuitive flow of events.
In the following, each component will be taken a look at and some technologies for its realization will be illustrated. Also for some components, accidents will be depicted that are somewhat related to a bug, error or possible weakness of it.

3.1. Perception

3.1.1. Sensing

Sensing, as the name suggests, is responsible to perceive the environment around the vehicle. The obvious source of data for this task, are various types of sensors. It is important to outline the differences between sensing and perceiving as sensing refers to only gathering ”raw numbers”-data while perception already includes a semantic analysis or interpretation of the sensor data. (See: [1])
However in the following, a selection of most commonly used sensors for autonomous or assisted driving should be introduced along with their features and weaknesses.

Challenge: Perception
Challenges in the area of perception are:

- Depicting the surrounding environment accurately, in order to generate a world model that is as similar to the real world as possible.
  - Detecting every relevant object, if possible.
  - Retrieving as accurate position data of these objects as possible, correlating the data of multiple sensors, if needed.
  - Classify relevant objects in the scene correctly.
  - (Optional) Estimate a realistic future movement pattern for dynamic objects in the scene.
  - Generate a world model that contains all relevant information as a data source for following layers (like Decision and Control)
- Ensuring to have environmental information available at all time, thus for all weather scenarios, bad visibility, etc..
- Covering a reasonably big area.

Cameras

One of the most basic sensors used in autonomous vehicles are cameras. As they provide a visual representation of the environment, they have been used in a magnitude of advanced driver assist systems for detecting objects like surrounding cars, speed limit signs and pedestrians. Installed in strategic places around the vehicle, cameras offer the possibility to monitor 360° of vision at all time which would be impossible for one human driver. Most cameras these days use complementary metal-oxide semiconductor sensors (CMOS) to capture an image in high definition with roughly one or two megapixels (Details concerning cameras taken from [8]).
An important technicality of the camera system is its dynamic range. The dynamic range, regarding imaging, describes the correlation of the brightest and darkest pixel in the image. It is most commonly expressed in dB, while most modern camera systems for automotive use have around 120dB. Cameras with 130dB or an even higher dynamic range offer better image quality as portions of the picture, that differ greatly in brightness, do not blow out, resulting in a loss of information (For dynamic range and signal to noise ratio in cameras in the whole chapter, see: [9]). A classic example where dynamic range is important, is when driving out (or into) a tunnel, since the tunnel itself is comparatively dark in contrast to the outside. Equation (1) shows how the dynamic range d in dB is related to the ratio between N_{min} and N_{max}, the maximum and minimum contrast between two pixels (the largest possible signal from one pixel and the noise floor).

$$d = 20 \cdot log(\frac{N_{Max}}{N_{Min}}) \tag{1}$$

$$SNR = \frac{\mu_{Signal}}{\sigma_{Background}} \tag{2}$$

$$f = \frac{l}{d} \tag{3}$$

$$1 \geq e \cdot fps \tag{4}$$

(Equations 1 taken from: [9], equation 2 taken from: [10]. Equations 3 and 4 are common definitions)
Another important specification is the signal to noise ratio (SNR). It quantifies how sensitive the image sensor is. Just as the dynamic range, it is usually given in form of a decibel number or just as a ratio as in equation (2). The equation for the SNR takes the average pixel signal value μ_{signal} and the standard deviation $\sigma_{background}$ of the background.
Camera sensors may have a SNR of 1 at 30 frames per second. The number of frames captured per second is important here, since it indirectly influences the SNR. Naturally, the noise of an image is inversely proportional to the amount of light that hits the sensor. Means of controlling the light which the sensor is exposed to include: **aperture** of the lens, **shutter speed** and **ISO**.
The **shutter speed** is usually expressed in fractions of seconds (e.g. $\frac{1}{50}sec$ or $\frac{1}{200}sec$). The

exposure duration is directly proportional to the amount of light that hits the sensor. To pick up the example above, 30 frames per second would equate to a maximum exposure of $\frac{1}{30}sec$, since equation (4) (exposure duration e and frames per second fps) obviously needs to be fulfilled. Choosing a higher frame rate like 60 fps(implying a shorter exposure time of maximum $\frac{1}{60}sec$) results in an image half as bright as at 30 fps, however moving subjects are less susceptible to motion blur, which is particularly important for cameras recording the environment outside the moving vehicle. Choosing a higher exposure time has inverse consequences.
The **aperture** (equation (3)) f of the lens describes the ratio between the focus length l and the diameter d of the aperture, which is a circular opening with a fixed or variable radius. The aperture is usually measured in F-Stops, like 1.4 or 2.8. A lower F-stop number allows for more light to hit the sensor, but also introduces a smaller depth of field, which means that a smaller portion of the image is in focus than at an F-Stop of 16.
Finally, the ISO determines the sensitivity of the sensor, with ISO 100 meaning that the sensor is less sensitive, however when enough light hits the sensor to expose the image correctly, less noise is visible. A higher ISO introduces higher noise in the image, but also requires less light.
A rule of thumb amongst photographers says, that when shooting outside on a sunny day, an aperture of 16, an ISO of 200 and a shutter speed of $\frac{1}{200}sec$ results in a well exposed image. However the task is to have a camera system, that delivers a useful image for ideally all situations, like driving directly into the sun as well as through a tunnel. Since this problem is present in a magnitude of applications like consumer cameras, smart phones, etc. automatic algorithms for exposure are well researched and work mostly reliably.
The recorded images are preprocessed on device and compressed in H.264 before they are sent to a central unit that may perform further operations.
Cameras may be used in conjunction with one another to gather three dimensional data (See: [8]).
Especially important for autonomous vehicles are forward facing cameras. These can be divided into medium- and high- range, with medium cameras being utilized for cross traffic detection, pedestrian recognition and emergency braking concerning the vehicles ahead. The main difference between cameras of different ranges is the horizontal field of view, which can reach from 35° for high range to 70° to 120° for medium range cameras (See: [8]). Figure 3 shows a lately released camera system including three different focal lengths, one fisheye lens for a high horizontal field of view, one for medium range and a third telephoto lens for longer distances. (Image and infos concerning this product taken from: [11])

Figure 3 A camera system designed by ZF and Mobileye for use in vehicles. Realeased in 2018, it is primarily targeted for advanced driver assistance systems as well as semi automated driving functions. The three different shadings visualize the fields of views of each lens. (Image taken from [11])

Solution: Camera-Sensors

Advantages of camera sensors:

+ Detailed information of the environment (Multiple mega-pixels of data).
+ Easy to interpret for humans.
+ Due to universal use cases also outside of the automotive domain, camera technology, algorithms for exposure, etc. are well researched.
+ Many variations available: (non-)color, infrared, monocular, stereo, etc.
+ Combinations of multiple cameras with different focal lengths can cover a wide variety of distances.
+ Comparatively cheap.

Disadvantages of camera sensors:

- Limited dynamic range can lead to information loss, especially in high contrast scenes.
- Remarkably worse performance at night.
- Produces a big amount of data. Processing may require powerful hardware and high bandwidth links.
- Susceptible to fail under harsh environmental situations like dirt on the lens, rain, fog, etc.

See: [9], [12] and [8]

Radar Sensor

Firstly, the commonly used word **Radar** is actually an abbreviation for **Ra**dio **D**etection **A**nd **R**anging, which already gives a clue to how it works. Since it played a major role in the second world war, radars were thoroughly researched and thus can provide a wide range of information. Depending on the details of the system, it may provide data like direction, height, distance, course and speed of an object (See [13]).
The information in the following paragraph was taken from [13] and [14].
One of the simplest versions of radar is used for **distance measurement**. A transmitter generates short electric impulses (usually less then a millisecond apart), which is then turned into a electromagnetic wave using a radar antenna. The speed of the wave approximately equals the speed of light (c), however in practical use cases do not take place in a vacuum. When the wave eventually hits an object like a plane or a ship it is reflected, just like a sound wave, and produces an echo. Just like you can hear an echo when yelling into a canyon, the electromagnetic wave is reflected and detectable at the radio antenna from which it originated from. A duplexer usually switches the antenna between send and receive mode to protect the hardware from the high powered impulses that are sent. As soon as the echo is detected by the radio antenna, where the signal is also demodulated, the round trip time of the impulse ($\triangle t$) is known and the distance d can be calculated using the simple formula (5). Formula 5 is already adjusted to only account for the distance between the antenna and the detected object, which is necessary since the round trip time would deliver twice the distance.

$$d = \frac{c \cdot \triangle t}{2} \tag{5}$$

In order to derive the **speed of an object**, one could store its distance to the antenna, using the method described above and than match a sensor reading at a later point to the same object. This way, knowing the distance traveled and the time interval, that it took the object to travel that distance, you can calculate its mean speed.
The information in the following two paragraphs is taken from [15].
Still, most radars in vehicles use either the pulse-Doppler approach or an implementation of continuous wave frequency modulation (CWFM). Devices utilizing the **pulse-Doppler** technique, have one antenna, that alternates between sending out a high frequency pulse and receiving the echo produced by objects in the environment. Sending out a wave, it can detect speed more accurately, taking use of the Doppler effect. The Doppler effect describes the elongation or buckling of an emitted signal when the distance between sender and receiver changes. Thus when a (stationary) radar antenna emits a wave, of which it knows its frequency, the wave travels through space with that exact frequency until it hits an obstacle and is reflected. Assuming the object is traveling towards the antenna, the frequency that is reflected and received by the radar antenna has a higher frequency than the wave it sent out (See figure 4). Accordingly, when the objects travels away from the antenna, the reflected wave has a longer wavelength than the sent one. This way you can gather the distance and

relative motion of an object.

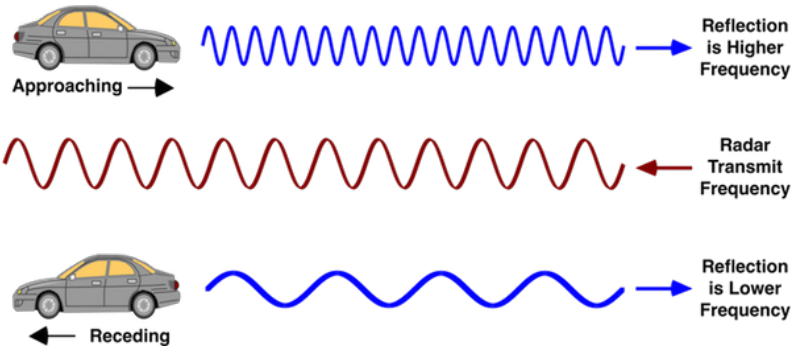

Figure 4 A depiction of the Doppler effect. A vehicle driving towards the sender of the wave reflects the radar wave at a higher frequency then the original wave. You could imagine a 1 hz wave reflected by a wall would be at 1 hz again, since the duration between two maximum amplitudes hitting the wall is one second. When the object now travels towards the sender of the wave, it reflects a frequency >1, since the time between two highs hitting the object is now reduced to <1 second. (Image taken from [1])

A radar that uses **continuous wave frequency modulation (CWFM)** has two dedicated antennas for sending and receiving. One antenna always sends a continuous carrier wave at a frequency that changes over time. The other antenna constantly receives the echo of that wave.

Both of the previously mentioned approaches can sense similar data, however differ slightly in hardware and evaluation logic.

An important aspect of radar sensors is their resolution. Paper [16] shows, that mobile sensors operating at 76-77 GHz are capable of producing a very usable image of objects ahead. Depending on the region that is supposed to be monitored by the radar, one can modify a radar system to deliver a long range image of up to 200 meters in a 10 $^\circ$ arc or a 30° arc with a range of 30 meters (See [15]). Since pulse-Doppler and CWFM have different approaches, each has its own characteristics. For example, it is more difficult for a CWFM radar to detect multiple objects than for a pulse radar (See [15]). Thus a combination of the two approaches is suggested by [15] that accounts for one of each kind and to dedicate the CWFM for long range and pulse-Doppler for short range coverage. Figure 5 gives an example of how processed radar readings look like.

[1] `https://www.quora.com/A-plane-is-moving-in-a-circular-path-around-the-transmitter-of-the-radar-Will-there-be-Doppler-shift-detected-in-the-radar`)

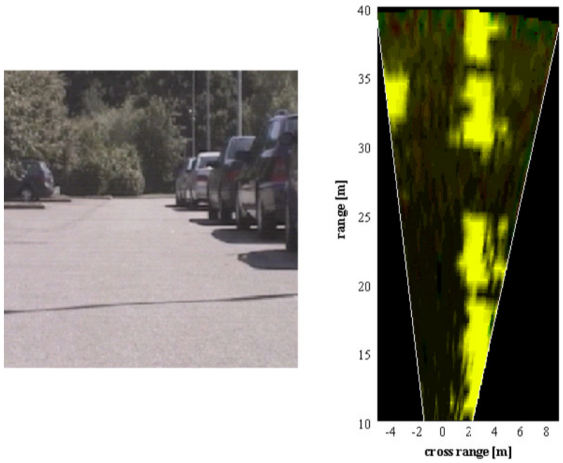

Figure 5 This graphic shows how the radar sensor detects objects in the environment opposed to a photograph of the scene. Looking at the sensor data on the right, you can see that the parked cars, from a top down point of view, are marked as yellow. Furthermore, the radar visualizes a detail, that was not clearly visible in the picture, which is a free parking spot in between tow cars. See [16]

Solution: Radar-Sensors

Advantages of radar sensors:

+ Can collect a wide variety of data (position, speed, heading, etc.)
+ Can detect relative speed of other vehicles accurately.
+ Is not impacted by poor visibility, bad weather conditions or lack of light (at night, in a tunnel).
+ Can be used for all kinds of ranges (long-, medium- and short -range).

Disadvantages of radar sensors:

- Not color sensitive.
- No especially high resolution (but also not excessively bad).
- Not particularly cheap.

See: [12], [15]

Lidar Sensor

The information in the Lidar sections are taken from [17] and [18]. As you might expect from the names being quiet similar, Lidar works almost like Radar, but instead of sending radio waves, it emits quick laser light signals, as often as 150.000 times per seconds. **Lidar** stands for **Li**ght **D**etection **A**nd **R**anging. Just as radio waves, the light bounces off of objects and the returning signal is recorded and processed. Lidar sensors usually only measure the time it

takes the signal to travel trough space and return to the sensor, in order to derive the distance between the sensor and the obstacle. Additionally it is particularly good at determining the size of an obstacle, thanks to its outstanding resolution compared to radar. Due to that and being able to generate a three dimensional map, is the reason, why most companies think of the Lidar sensor as an essential part of an autonomous' cars suite of sensors.
The sensors usually work within "near-infrared, visible (but not really visible), and UV spectrum" (See [18]). The region in which information is collected can change to benefit resolution in a particular image section. Spinning mirrors are often used to redirect the laser beam to increase resolution. Resolutions of 0.25 degrees of angular resolution are not uncommon. Figure 6 shows the inner mechanics of a Lidar sensor.

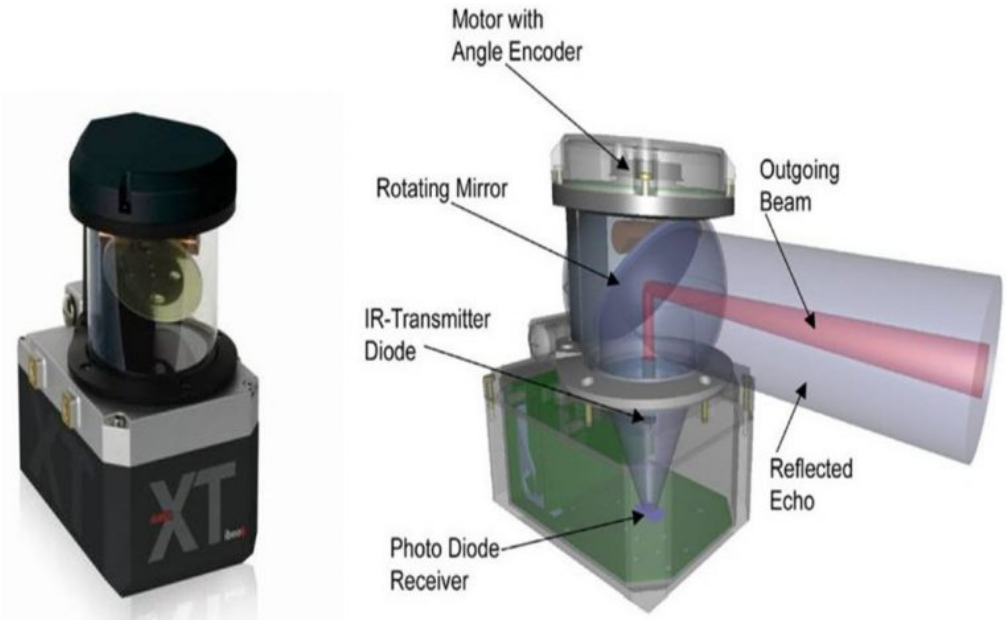

Figure 6 This graphic shows the components of a Lidar sensor. You can easily see the two light beams, the red one indicates the generated laser pulse that leaves the sensor, whereas the bigger, blue beam represents the reflected echo. Also you can spot the components like the rotating mirror and its motor, which are responsible for redirecting the laser to enhance resolution. (Picture taken from [19])

Processing the time it takes the light to leave the sensor, bounce back and reach the sensor again, the distance can be derived, similarly to radar. The result is an array of distance information that can be visualized in an image, as in figure 7.

An important advantage of Lidar over Radar is its higher resolution. The company Waymo stated, that its Lidar system can detect pedestrians and can even tell what direction they are facing and thus may enhance the accuracy of predictions of their probable trajectories. Furthermore the system is allegedly able to detect gestures e.g. by cyclists around 200 meters away at the car's full speed (See [17]). Figure 8 compares an image generated from Radar- to one by a Lidar-sensor.
Furthermore it is worse than radar for detecting the speed of other traffic participants as well

Figure 7 The data shown in the upper image represents the reflectivity data. The bottom visualization contains distance information, where darker means closer and ligher means farther. (Picture taken from [18])

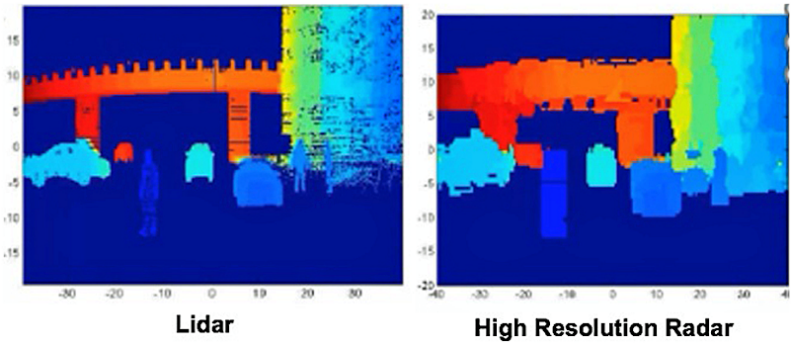

Figure 8 The two images show the same scene. The left image is product of a Lidar sensor, whereas the right image is taken from a high resolution radar system. It is clearly visible, that Lidar supports a much higher resolution, resulting in greater details in the image. (Picture taken from [17])

as in short range areas.

Solution: Lidar-Sensors
Advantages of Lidar sensors:

+ High resolution.
+ Good for long range.
+ Usable in all lighting conditions.

Disadvantages of radar sensors:

- Not color sensitive.
- Quality decreases in bad weather conditions like fog, snow, etc.
- Expensive (At the time: declining price, still multiple thousand dollars).
- Comparatively large sensor.
- Not as good for speed detection as Radar

See: [17], [18] and [12].

Ultrasonic Sensor

Finally, ultrasonic sensors will be covered. Ultrasonic sensors have been used in vehicles for many years, as they are frequently used in driver assistance systems for parking. In this type of application, ultrasonic sensors are used to measure the distance between the car and the nearest obstacle, like walls or fences and forward this information in form of visual or audible feedback (This paragraph: see [20]).
In fact Ultrasonic sensors are almost exclusively used to measure the distance to an obstacle. Still, they also have the ability to derive information about the material of an object. Also they can be used to measure speeds utilizing the same Doppler-Shift effect as Radar and Lidar do , however only for lower speeds due to a comparatively low range. Common ranges are from 0.15 to 5.5 meters (This paragraph: see [21])

But there are even more similarities between ultrasonic sensors and Radar / Lidar. For its main purpose, distance measurement, just like the previously mentioned systems, it measures the time of flight of a sound wave. Therefor it emits a sound wave, which may be in the range from 30kHz up to 5 MHz. Such high frequencies are not audible for humans, who can perceive 20-20kHz sound waves. [20] states formula (6) for calculating the distance between the sensor and an object, where k is a constant near 0.5 to account for the fact that the signals travels twice the distance (to the object and the echo back), however they account for slight derivations from 0.5 to allow a more precise reading, having calibrated the sensor

Figure 9 A line of ultrasonic sensors manufactured by Bosch. Their detection range ranges from 15 centimeters to 5.5 meters. The dimension of the senors is about 44 by 26 millimeters. (Picture taken from [21])

and setting k. T_f is the measured time of flight of the signal, v_s is the speed of sound. However sound, rather than light or radio waves, has a substantially different speed in different temperatures, since it is a vibration of matter (See [22]). [20] suggests formula (7) for picking more exact speed of sound depending on the temperature T in Kelvin. The emitted audio signals range from 30kHz to 5MHz. Higher frequencies benefit resolution but also increase the cost of the sensor, also higher frequencies come with a higher sound attenuation in the air. Lower frequencies diminish scattering problems at the cost of a lower resolution. Also, the high frequencies of the sound waves allow for a high directivity, with the frequency directly influencing the distance and directivity (See [22]). The information in the paragraph above is extracted from [20].

$$d = k \cdot T_f \cdot v_s \tag{6}$$

$$v_s = 20.055 \cdot \sqrt{T} \tag{7}$$

Taking 20°C to calculate the speed of sound at that temperature, you can see that the (approximately) resulting 343 meters per second are by a magnitude lower than the speed of light (approx. $3 \cdot 10^8$ meters per second). On one side this can be seen as an advantage, since you can use slow electronics to process the sensor data (See [22]), however it also means, that it can not operate at such high speeds as Radar and Lidar, which signals travel at the speed of light.

Solution: Ultrasonic-Sensors

Advantages of Ultrasonic sensors:

+ Cheap.
+ Small size.
+ Robust to various weather conditions.
+ Works under any lighting condition.

Disadvantages of radar sensors:

- Not color sensitive.
- Can only measure low speeds.
- Limited range.

See: [22], [20] and [12].

Conclusion: Sensors
As demanded at the beginning of the perception chapter, the following challenges can be addressed by a reasonable choice of sensors:

- Depict the surrounding environment accurately (both accurate and correct data).
- Ensure to have environmental data at all time as well as for all scenarios as possible (day/night, clear, fog, snow, dust,...).
- Cover a reasonably big area.

The following table summarizes what sensor can be used for which application.

Sensor Type	Works at	Weather robust	Range	Resolution
Camera	Day / Worse at night	No	short - long	high
Radar Sensor	Day & Night	Yes	short - long	medium
Lidar Sensor	Day & Night	To some extent	long	high
Ultrasonic Sensor	Day & Night	Yes	short	low

Thus a combination of different senor types that ensures the aforementioned demands, is highly recommended and contributes to a more reliable system in the form of redundancy.

See: [12] and sources from the sections above.

Classification of Sensors

This section contains information taken from [1]. The senors mentioned in the previous sections can all be classified as external sensors, as they are used to monitor the environment around the car rather than collecting data of the vehicle itself. Second of which are sometimes referred to as internal sensors. [1] furthermore distinguish between senors at a high ”Technology Readiness Level” (TRL) and low TRL.
At **lower TRLs**, which, according to [1] are common in experimental vehicles, it is standard to receive and process the raw sensor data itself. Tasks that than still need to be done are ”filtering, estimation, fusion, association and possible classification operations” (See [1] p.2 Section 2.1).

As opposed to those sensors, these of a **high TRL**, often sold by automotive vendors, include a processing unit and return information on a higher level. Thus such a sensor, rather than reporting raw readings from the sensor, may return predefined objects, like pedestrians, cars, obstacles along with specific data (e.g. position, speed, heading, etc.). Integrating a high TRL sensor into a vehicle's environment brings several advantages. First of all, the sensor system can be treated as a "Black Box", thus no inner workings need to be known in order to use the sensor. However some of these systems still allow for extraction of the raw data. Secondly, since the return types are rather high level, sensor readings need not be run through the sensor fusion (and might even skip semantic understanding). According to [1], these types of sensors are commonly found in late stage prototypes where the main focus seems to be at "failure probabilities, confidence levels of output data, and common mode failures with other sensors" (See [1] p.2 Section 2.1).
As a comparison of the two approaches, [1] states, that if processing of raw sensor data is part of the overall system, "details tend to be more transparent, although that does not necessarily make the analysis easier" (See [1] p.3 Section 2.1).

Typical Equipment of current Vehicles

Finally, a few examples of how current vehicles and their suite of sensors, should be given and compared.

Waymo (Google) Autonomous Vehicle

Information in the following paragraph is extracted from [23] and [24]. Waymo uses Lidar, Radar and cameras, all in full 360° of vision. According to Google, the range of this sensor suite covers a distance of around 300 meters. The Lidar system used, is developed by Waymo itself and one of their vehicles usually includes three different types of Lidar: one short range to detect objects directly around it, one high resolution mid range and one long range. Being the only sensor that is sensitive to color, Waymo also includes a "high resolution" camera (or vision System as they call it). It includes multiple sets of cameras that are optimized for long range and should also work at low light scenarios. A radar system ensures environmental data even at bad weather conditions. All of the aforementioned sensors are designed to deliver a full 360° of coverage. Additionally, the following supplemental sensors are installed:

- Audio sensor: in order to detect sirens of emergency vehicles.
- GPS sensor: to support the vehicles localization capabilities.

Figure 10 shows the sensor suite around the car. Additionally, Figure 11 leads to a website containing a 360° video by Waymo in which the sensor arrangement is visualized.

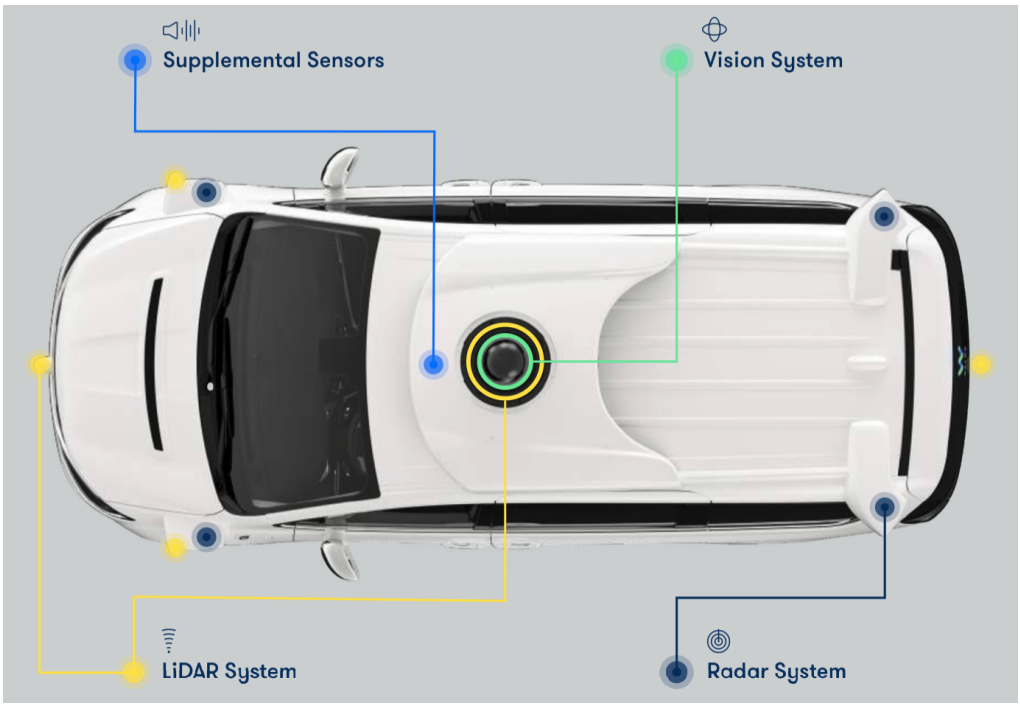

Figure 10 A visualization of the sensors used by Waymo's test vehicles. (Picture taken from [23])

Link: `https://waymo.com/tech/`

Figure 11 This site by Waymo contains a 360° video depicting the sensor suite of their vehicles.

Tesla Model S

In contrast to Waymo's vehicle, the Tesla Model S has been for sale since 2012. Tesla is often taken into consideration when talking about autonomous vehicle, since they are heavily promoting their advanced driver assistance system for lateral and longitudinal control, in combination with an advanced collision avoidance system as their so called "Autopilot". Therefore, Tesla has been harshly criticized, since the name suggests an SAE Level 3 autonomous system, whereas the manual describes the "autopilot" rather as an assist, that requires constant monitoring. That often leads to drivers neglecting their supervision tasks and even distract themselves; some of which even brag about the feature, uploading videos where the driver and passengers play games or watch a movie during the ride, with the autopilot active. Though it should be stated, that the car requests the driver to every once in a while hold the steering wheel, it allows for multiple notifications to be ignored and as demonstrated by a several sources, the system can be as easily tricked as placing an orange onto the steering wheel.

Information about the senors are from the official autopilot website (See [25]). Regarding the sensors array of a Model S, it contains 8 cameras spread throughout the vehicle (See Figure 12), three of them are forward facing, one wide 120° camera (range: 60 meters), one

medium or ”Main” camera (range: 150 meters) and one narrow, long range camera (range: 250 meters). Symmetrically at the right and left side of the car, there are ”Forward Looking Side Cameras” (90°, range: 80 meters) in order to observe vehicles entering the ego vehicles lane and offers additional safety at intersections. Furthermore, also installed on both sides of the car are two ”Rearward Looking Side Cameras” that cover blind spots and are important for lane change maneuvers. Finally a rear view camera is used for backing up or parking, but also to observe the traffic behind the car.
A forward facing radar sensor delivers data for up to 160 meters ahead, also at bad visibility. Finally, ultrasonic sensors cover a complete area of 8 meters around the car.

Figure 12 A visualization of the sensors and their ranges of a Tesla Model S. (Picture taken from [25])

The hardware of the ”autopilot” has changed multiple times since the first edition of the Model S, the one summarized above is the most current one.
What stands out is the lack of a Lidar sensor. In fact, founder of Tesla, Elon Musk has stated, that he does not think that Lidar is going to be a long term solution:

> “In my view, it[Lidar]’s a crutch that will drive companies to a local maximum that they will find very hard to get out of. [...] Perhaps I am wrong, and I will look like a fool. But I am quite certain that I am not.”
>
> —Elon Musk [26]

[27] seems to have spotted a Tesla that was equipped with additional sensor hardware which is very likely a Lidar sensor, it is thus very likely that tests were done on Tesla’s side regarding Lidar technology.
It is quite striking that Tesla’s ”autopilot” system is mainly relying on cameras, even more so since Mobileye, a company that supported Tesla developing the vision based system has quit

its collaboration due to too many accidents while the autopilot was engaged. Since then, Tesla has developed its own platform ”Tesla Vision”, based on a Nvidia Drive PX2 onbord computer. (Informations of this paragraph were taken from [26])

Audi A8

On the way of bringing the first car with official SAE Level 3 autonomous driving system, Audi designed the new A8 to take over driving tasks as long as the vehicle is on a highway and the traffic flows below 60 km/h. According to Audi, it would then be the first car, that would allow the driver to be distracted from the driving task. Latest developments in the US however indicate, that the A8 model for the US might not be equipped with Level 3 automation for legal reasons. Nevertheless Audi claims the technical readiness to do so and thus it is valid to include an overview of the sensors into this comparison. (Information by Audi: [28] and by [29])

Figure 13 shows all the different sensors that come with the Audi A8 in order to fulfill a Level 3 autonomous driving task. Firstly Audi's design features 4 cameras that contribute to a 360° field of vision; they are mounted in the back, the radiator grill and by both side mirrors respectively. A fifth camera is located at the upper center of the windshield. Ultrasonic sensors are located at both ends of the vehicle, four mid range radar systems are placed at the four corners of the vehicle. Finally a long range radar is installed in the radiator grill next to a Lidar sensor. According to Audi, the A8 is thus the first car to feature a lidar sensor. (See: [30])

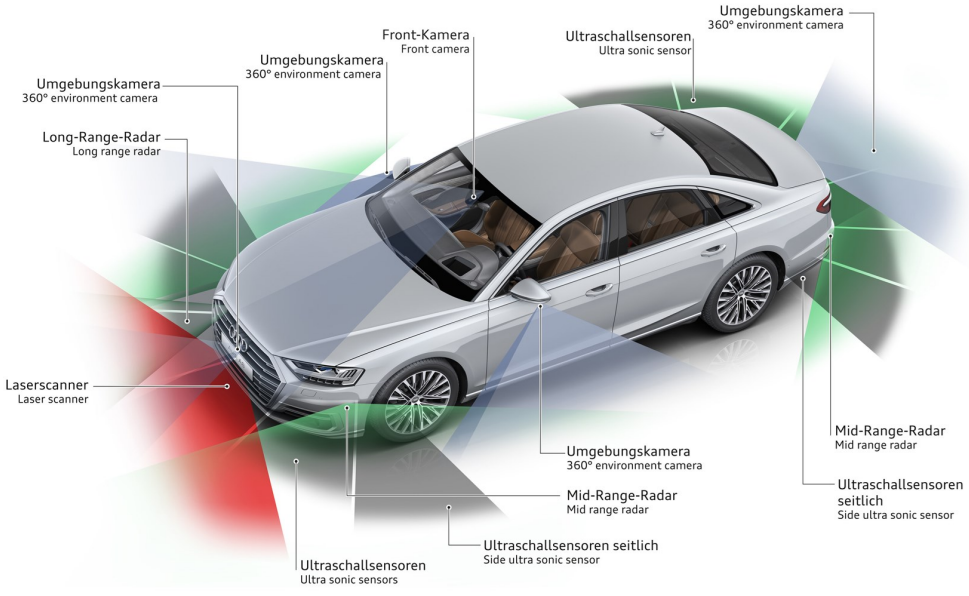

Figure 13 A visualization of the sensors of the Audi A8. (Picture taken from [30])

All in all, it shows, that there are indeed noticeably different approaches for the choice of sensors and their placement, however many similarities can be found like the 360° camera systems that are featured on all vehicles demonstrated above. Radar also seems to have become a sensor that is indispensable for autonomous driving, as well as ultrasonic sensors. Also Lidar seems to be a popular choice to complete the vehicle's suite of sensors. However, it is going to be critical for sensors that are used to fulfill demanded criteria to be permitted for use in autonomous vehicles as it represents a highly safety relevant topic. In the following, one instance is demonstrated how sensors (and processing of their data) might lead to unwanted effects.

3.1.2. Accidents of autonomous vehicles with stationary obstacles

The informations regarding the accidents and their cause, are taken from the sources: [31], [32], [33] and [34].

There have been numerous occasions where autonomous vehicles have collided with stationary objects. On of these incidents occurred on Friday, January 22nd, 2018, when a Tesla Model S rear-ended a stationary fire truck at approximately 105 km/h (65 mph). The car was driving on the Interstate 405, according to the driver in autonomous mode. The Tesla was following a pickup truck which suddenly changed lane to avoid a parked fire truck ahead. The Model S however did not slow down behind the stationary fire truck and caused a collision. Regarding the severity of the accident, the fire truck had some damage to its rear end, whereas the car's front end was destroyed. Thankfully, due to the comparatively big crumple zone, no one was injured.

An almost identical accident happened just months after, on May 11th, 2018. Just like the previous accident, a Tesla rear ended a stationary fire truck at more than 95 km/h (60 mph). The driver has been hospitalized, after her Tesla ran over a red light and collided with the stationary fire truck.

Still, the list of these types of accidents does not end here. In the same month, on May 29th, 2018, a Tesla driving in "Autopilot" mode crashed with yet another stationary emergency vehicle. This time, the Tesla was driving in Laguna Beach, California, when it collided with a stationary police car. Both cars seemed to have suffered from heavy damage.

Thus stationary vehicles seems to be a prominent weak spot of Tesla's "Autopilot" feature. In fact an excerpt out of the Model S's Owner's manual warns:

> "Traffic-Aware Cruise Control cannot detect all objects and may not brake/decelerate for stationary vehicles or objects, especially in situations when you are driving over 50 mph (80 km/h) and in situations where a vehicle you are following moves out of your driving path and a stationary vehicle or object is in front of you. Always pay attention to the road ahead and stay prepared to take immediate corrective action. Depending on Traffic-Aware Cruise Control to avoid a collision can result in serious injury or death. In addition, Traffic-Aware Cruise Control may react to

vehicles or objects that either do not exist or are not in the lane of travel, causing Model S to slow down unnecessarily or inappropriately."

— [35]

This shows that the aforementioned accidents are known problems of Tesla's Model S.
However Tesla is not the only car manufacturer struggling with this exact problem. As reported, Volvo's semi autonomous "Pilot Assist" system as well as most other cars that are equipped with an adaptive cruise control system.
To see why stationary vehicles seem to be a problem, researchers have reconstructed this kind of accidents and found out, that this behavior appears, when the adaptive cruise control of the automated car is set to keep a given speed and then detects a car in front for which it has to slow down. When the preceding vehicle now changes lane (eg. to avoid a stationary vehicle such as a parked fire truck or police car), the automated vehicle accelerates to again reach the previously set speed. This is when the stationary vehicle is ignored and a collision is barely avoidable. Additionally the reason why the stationary vehicle is often overlooked in such situation is, that these systems are designed to ignore certain readings.
In order to minimize the amount of false positives (the times where the car initiates a braking even though there is no reason to do so) specific algorithms decide which stationary objects need to be minded. Since the streets are filled with stationary objects like posts, guard rails, traffic signs next to the street or hanging from above, a fine balance is required to decide whether a stationary object is an obstacle which you should brake for or if it is no threat for the car.
Since occasional emergency brakings without an obvious reason might impose an even bigger danger of being rear ended on a highway with a significant speed delta, it is important to let the sensors ignore certain objects.

3.1.3. Sensor Fusion

Since the main goal is to build a world model that resembles the real environment as accurately as possible, the data, which is collected by the sensors, needs to be evaluated and finally combined in the world model. This represents one key advantage of a world model: all the data can be retrieved from a centralized point. In contrast: advanced driver assistance systems often do not access one central source of information, but only communicate with the one or multiple sensors, that deliver information relevant for that task (See n:m relation in 2.3).
Further processing of the sensor data in the model proposed by [1], is done in the **Sensor Fusion** part of the system. Since an active world model, allowing a magnitude of information extraction is the main goal, a process called sensor fusion is necessary. Since at this point the only information collected are values from independent sensors, every sensing component sends its data to a central computing unit, which then uses all of the information and correlates them in order to improve the quality of the measurements. This process is trivial for humans, since it lies in our nature to interpret our senses. Humans are thereby able to

locate an engine sound in a three dimensional room; when we then move our head and see a car, we instinctively know that the vehicle is the origin of the sound. In the previous example, humans combine the information of two senses (hearing and seeing), however it is the task of sensor fusion to determine a link between a car that is visible on a reading of a camera image and an obstacle detected by the radar sensor. Establishing a logical connection between multiple sensor reading is task of the so called senor fusion. Since computers lack the kind of instinctive behavior compared to humans, code needs to be designed that acts in a way to achieve the same result. Therefor multiple approaches and techniques exist, some of which are presented in the following.
Many information of the sensor fusion chapter are taken from [36], which provides both, a good starting point to the field of sensor data fusion but also covers various models and methods.

Terminology

There have been many attempts to find a fitting terminology for the system that is realized in this logical component. Thus terms like ”sensor fusion”, ”data fusion”, ”information fusion”, ”multi-sensor data fusion” and ”multi-sensor integration” are used in literature to describe different techniques to that are designed to centralize the data from multiple sensors (See [36]). A definition of the term sensor fusion might look like the one stated by [36].

> ”Sensor Fusion is the combining of sensory data or data derived from sensory data such that the resulting information is in some sense better than would be possible when these sources were used individually.”
>
> — [36]

Next to the definition of sensor fusion, terms like direct and indirect fusions are used by G. T. McKee (See [36]), where direct fusion refers to a combination of data produced by heterogeneous and homogeneous sensors, soft sensors and previous sensor readings. Indirect fusion instead uses a priori information and inputs by a human.

Challenge: Sensor Data Fusion
Demands for the sensor fusion system are:

- Cleaning the input data to respect important objects and neglect useless information reliably, which benefits runtime performance.
- Correlate data from multiple sensors that refer to the same real world object.
- Track objects over a period of multiple iterations in order to gather even more information like speed, trajectory which may also contribute to the classification of the object (pedestrian, cyclist, car, animal, etc.).
- Ensure a fast enough design that runs in real time to provide as up to date information as possible.

Categorization of Sensor Fusion approaches

Sensor fusion systems can roughly be divided into two groups, C^3I, which stands for *command, control, communications, and intelligence* and embedded real-time applications. A classification of the fusion level is often handled by a low-, medium- and high-level scale which can be interpreted as such:

Low-Level: combines multiple sources of raw sensor data to generate a new data with higher entropy than the original numbers.

Medium-Level: higher level information like image features (edges, corners, textures) are utilized and fused into a feature map.

High-Level: ”combines decisions from several experts.” (See: [36] p. 7) These might include: fuzzy logic, statistics, etc.

Operating at a much slower rate (multiple seconds to minutes), C^3I fusion is not suitable for real time applications.

JDL Fusion architecture

Infos from the JDL sections are taken from [36]. One fusion architecture that has been proposed in 1985 is referred to as the US Joint Directors of Laboratories (JDL). It was developed under the Department of Defense and describes a model with different levels of processing as well as some entities to support the sensor fusion task.
The main components, which are located inside the ”Data Fusion Domain”, are five Levels of processing, that have no particular chain of execution and might even run concurrently. Also a part of the ”Data Fusion Domain” is a Database Management System which contributes to the seamless flow of information. On opposing ends of the model stand the sources (i.e.

sensors, database knowledge, human input, etc.) and the ”Man-Machine Interaction” as an output. All components are connected to a bus that provides a means of communication.

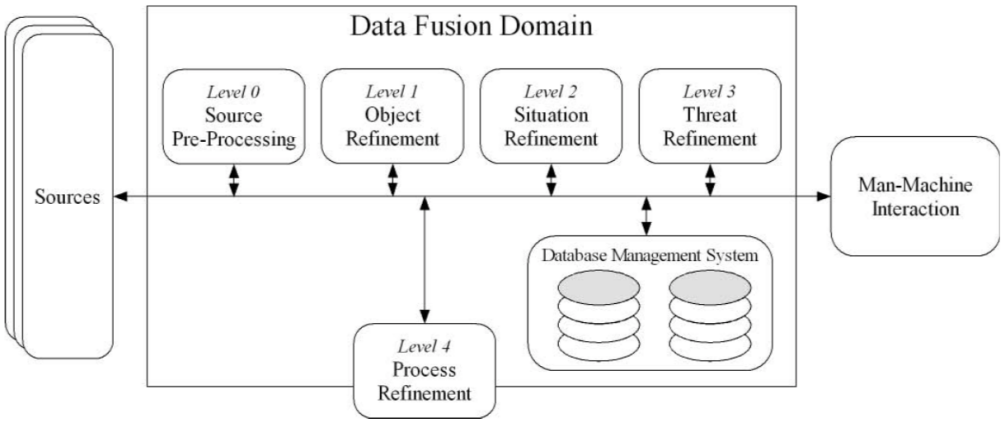

Figure 14 An overview of the JDL architecture. Five levels of data processing inside the ”Data Fusion Domain” can work concurrently, communicating via a common bus. Graphic taken from [36].

A short description of each entity in the JDL Architecture:

Level 0: Source preprocessing firstly analyzes the data and optimizes it in a way that reduces the load for future processing.

Level 1: Object refinement aligns the data to adhere to a given unit of measurement and correlates sensor data that refers to the same real world object.

Level 2: Situation refinement tries to identify a higher level situation of the environment by analyzing the relationship of objects and events.

Level 3: Threat refinement tries to determine vulnerabilities and opportunities based on a priori information and predictions.

Level 4: Process refinement monitors meta data like performance to ensure it adheres to demands imposed e.g by real time constraints

Waterfall Model

According to [36] the waterfall model is an alternative to the JDL architecture since it is suitable for real time embedded systems. It is in many aspects similar to the JDL architecture, however concentrates a bit more on lower level processing. Figure 15 shows the structure of the waterfall model.

Parallels to the JDL architecture are:

- Sensing/Signal processing $\leftrightarrow$ Level 0

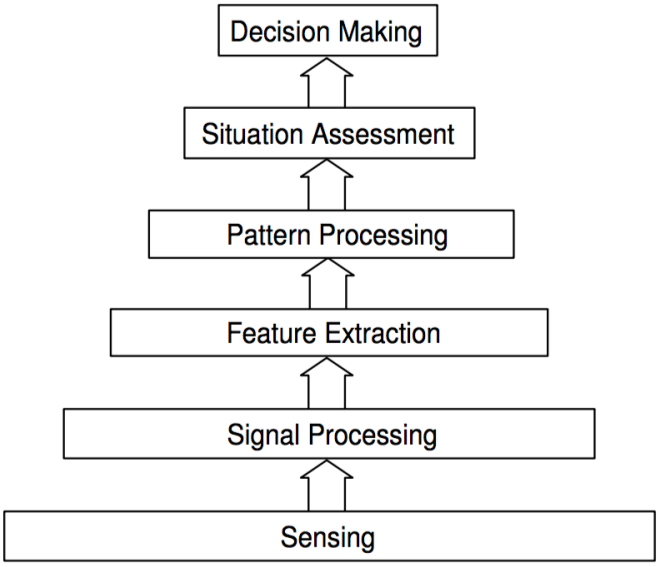

Figure 15 An overview of the waterfall model. The ”waterfall” structure is clearly visible, as information only flow in on direction. Graphic taken from [36].

- Feature extraction/pattern processing $\leftrightarrow$ Level 1
- Situation assessment $\leftrightarrow$ Level 2
- Decision Making $\leftrightarrow$ Level 3

One of the most striking difference is the lack of any feedback data flow. The usage of this model is thus lower.

Method: Kalman Filter

(Information and equations taken from [36])
Shifting from the architecture of a sensor fusion system to a method that is used in processing the data collected by the sensors, the Kalman filter is a popular choice. It is used in many applications, one of which is calculating an optimal estimate for values that are either not directly measurable or which readings are noisy. It also allows for the fusion of multiple sensors into one data vector, containing the internal state, provided all dependencies are linear.
A classic Kalman filter is described by two formulas 8 and 9. It should be noted, that it uses discrete time steps, which will here be referred to as k.

When applying the Filter, an estimation for the next time step is generated. This might be with the help of an underlying physical model in order to get an estimation that is as close to the real value as possible. This physical model is then manifested in the matrices A, B and C. The variables for the noise as in w and v are mostly modeled as Gaussian white noise $\sim N(0, Q)$ and $\sim N(0, R)$ respectively.

$$\vec{x}_{k+1} = A \cdot \vec{x}_k + B \cdot \vec{u}_k + w \tag{8}$$

$$\vec{y}_k = C \cdot \vec{x}_k + v \tag{9}$$

where: $\vec{x}_{k+1}$is the state of the system at the time t+1.

- A is a non singular state transition matrix.
- $\vec{x}_k$ is the state of the system at the time k.
- B is a matrix that determines how sensor readings at time k affect the systems state vector $\vec{x}_{k+1}$
- $\vec{u}_k$ is the input (e.g. command for the system) at time k.
- w is a number that represents the noise of the system.
- $\vec{y}_k$ is the observation of the system including noise. The elements of this vector are noisy sensor readings at the time t.
- C is a matrix that determines the effect of $\vec{x}_k$ on the internal state.
- v is the noise of the measurement.

What the Kalman filter now does, is that that it takes the latest internal state and uses the inputs that were made to the system in order to generate a prediction $\hat{\vec{x}}_{k+1|k}$ to how the internal state looks like at the next time step. Note, that all variables with a hat ($\hat{x}$) are predictions. Since this estimation is prone to an error, a covariance matrix P is calculated, modeling the possible deviation from the estimate.

$$P_{k+1|k} = A \cdot P_{k|k} \cdot A^T + Q \tag{10}$$

where: $P_{k|k}$is the final covariance matrix of the last time step (See formula 13).

In the next step, the so called Kalman gain is calculated, which will later be used as a weight factor to correct the estimation.

$$K_{k+1} = \frac{P_{k+1|k} \cdot C^T}{C \cdot P_{k+1|k} C^T + R} \tag{11}$$

You can see, that the Kalman gain is directly influenced by the covariance of v, the noise of the sensor readings. As a consequence, if the covariance is high, the denominator gets big-

ger, thus the Kalman gain is smaller. Since this value influences by what factor the difference between estimation and sensor reading is corrected, it only seems logical, that values with a higher noise are not as much accounted for as those with lower noise, thus lower covariance values.
When now the new reading of the sensors are available ($\vec{y}_{k+1}$), the estimation can be updated:

$$\hat{\vec{x}}_{k+1|k+1} = \hat{\vec{x}}_{k+1|k} + K_{k+1} \cdot (\vec{y}_{k+1} - C \cdot \hat{\vec{x}}_{k+1|k}) \tag{12}$$

As visible in equation 12, the updated state is the initial estimation plus the difference between the estimation and the actual sensor readings. Again picking up the role of the Kalman gain, if the sensor readings come with little noise, the data delivered by the sensors are weighted more heavily, in contrast to sensor data that has a high noise level.
The updated estimation again has an error covariance matrix, which is the new $P_{k+1|k+1}$ in 13.

$$P_{k+1|k+1} = (I - K_{k+1} \cdot C) P_{k+1|k} (I - K_{k+1} \cdot C)^T + K_{k+1} \cdot R \cdot K_{k+1}^T \tag{13}$$

Further methods and functionality of sensor fusion

Another method that can be used in the sensor fusion part of the system, is **Fuzzy logic**.
Fuzzy logic differs from regular logic in the main aspect, that it is not limited to discrete values of true ”1” and false ”0” but allows for a variable to be ”0.4”. An example in the automotive domain for that would be a vehicle that drives in front of the ego-car. A conventional logic could have a boolean variable ”enoughDistance” to describe whether there is enough distance between the two cars and another boolean variable ”brake” that is ”1” or true when brakes need to be applied. This would imply, that car always will initiate a full braking, once the ”enoughDistance” value is ”0” or false. Fuzzy Logic on the other hand might set the ”enoughDistance” value to ”0.2”, which would mean that there is still pretty much room between the cars, and could therefor set the brakes to ”0.2” instead of putting full brake pressure on.

Other methods include: **Statistical classification**, **logical connections** and **rule based** sensor fusion (See [37]).

As you can see, sensor fusion is an important task, which has a decisive part in the overall

system, since it processes and provides most of the data that is then used to construct the world model, upon which most calculations of the Decision and Control section rely. Therefor an error in this part of the system is especially serious, as you can see in the following accident.

3.1.4. Tesla Accident with semitrailer

Information concerning the following accident and its causes taken from: [38] , [39] and [34].
On Monday, May 7th in 2016 Joshua Brown, owner of a Tesla Model S, was killed during an accident which occurred, while the ”Autopilot” feature was activated. He was driving east-bound on highway US Route 27A near Willston, Florida. At the location of the accident, the road has four lanes, two in each direction. The collision took place at an intersection, where the road NE 140th Court meets the highway approximately perpendicular.
When the Model S was approaching the intersection, a truck that was headed westbound, took a left turn to get onto NE 140th Court, thus temporarily blocking the road. Despite the ”Autopilot” controlling the vehicle, the car crashed into the right side of the trailer. Due to the higher ride hight of the trailer compared to the much lower Tesla, the car went trough under it, tearing its roof off. The car continued driving for 277 meters when it eventually came to a stop colliding with a pole. Joshua Brown, driver of the Tesla was killed during the accident, the driver of the Truck left the scene uninjured.
The data recovered from the Model S showed, that the ”Autopilot” feature was active during the accident, the speed at the time of impact was just short of 120 km/h (74 mph) at a posted speed limit of 105 km/h (65 mph).
Despite the vehicle crashing, the investigation report by the National Highway Traffic Safety Administration (NHTSA) did not detect any fault in the automated driving system. Tesla’s so called ”Autopilot” consists of three main components: an adaptive cruise control system, regulating the speed of the car and keeping a safe distance to the vehicle in front (called TACC). Secondly the auto-steer system keeps the car within the bounds within the lane markings. Finally Tesla’s ”Autopilot’ also includes a lane change assist, that automatically performs a lane change when the driver orders it to do so. Subsequently, longitudinal control is taken over by the TACC, whereas latitudinal steering is provided by the auto-steer system. This package of assists makes the Model S conform with the definition of a Level 2 automated vehicle according to SAE Levels depicted in 2.2.
Nevertheless the ”Autopilot” system additionally comes with an automatic emergency braking (EBS) assist, which should avoid accidents, even if the three main components fail to work reliably. The reason why the EBS did not prevent the crash has to do with the detection and classification of vehicles respectively obstacles, thus one might blame a faulty sensor fusion component for this accident.
The company that developed the software for detecting vehicles, Mobileye, stated that the system was not designed for such a situation. Mobileye uses deep learning neural networks in their EyeQ SOC that is used in the Model S to detect vehicles. Nevertheless, it has only been trained to detect rear ends of vehicles only. Since the Tesla only perceived the truck from its side, as it was making a left turn and stood perpendicular to the car.

Since this system is designed to detect relevant traffic participants by analyzing various senor data, it falls under the category of sensor fusion, performing object association and tracking.

3.1.5. Localization

The localization component has the task of determining the position of the vehicle on a global scale. Where the dynamic world model tries to represent and depict the car's position on a local scale, the localization system works on a higher level, positioning the car in a larger context. Still, it might communicate with the sensor fusion component, in order to determine the position of the vehicle more accurately.

Challenge: Localization
Challenges in the area of Localization are:

- Determine the position of the vehicle in a global context as accurately as possible.
- Enhance the available data by using algorithms that take additional data into account.

Map matching

One technique that can be used in order to retrieve the vehicle's position as precisely as possible is called **map matching**. One of the most commonly used technologies to get positional data is the U.S owned "Global Positioning System" (GPS). According to official information[1] the error of the system itself is $\leq$ 7.8 meters in 95% of cases. Since this is not especially detailed, map matching is used in order to more accurately determine the vehicle's position.
The basic function of map matching takes a series of GPS data measurements and enhance their accuracy by matching them to the positions of a previously recorded map of roads. Take a highway for an example; the error of a single GPS measurement would not easily allow to determine, on which lane the vehicle is traveling. However, taking multiple data points in an ordered list, allows a map matching algorithm to match the (inaccurate) GPS readings with the accurate road map data. Popular map matching APIs are: Google Roads, Mapbox Map Matching API and the open source graphhopper routing (See [40]).
Since this process is used in nearly all vehicle GPS navigational systems, which have been in use for many years, this technique is comparatively sophisticated.
The position of obstacles, that were found by the previous components, may also be transformed into the map coordinate system. This also opens up the possibility to share information about obstacles with other traffic participants and on a higher level contribute to traffic jam information and bypass recommendations.

[1] `https://www.gps.gov`

Solution: Map Matching

Advantages of Map Matching:

+ Delivers a more detailed information about the vehicle's position.
+ Opportunity to share detailed information about obstacles.

Disadvantages of Map Matching:

- Requires high definition maps which require much hard drive space to store (might be dynamically loaded over cellular network).
- Matches the position of the vehicle to a prerecorded centerline of the road, which may just be enough detail needed for a global scale, but not for the autonomous driving task.

(Visual) Landmark recognition

Another approach that would allow a greater accuracy in positioning, is the tracking of (visual) landmarks in the environment. This might be done by observing features in the visible area around the vehicle, like lane markings or special signs. Realizations can be done in multiple ways.

[41] for example uses the vehicle's Lidar Sensor to detect the curbs of the road and matches them with the ones provided in a digital map. They furthermore use two Kalman filters to refine the position data. The first Kalman filter therefor takes input data from the "internal navigation system" (INS) and a "low cost GPS". So far, the result is not influenced by the Lidar's curb information. The second Kalman filter however takes the result of the first filter as well as the processed Lidar data, containing the information about the position of the curbs to finalize the result.

A visual approach is described by [42] who trained an optical character recognition system (OCR) that recognizes various markings on the road (such as arrows, speed limits, etc). Provided the position of these markings is accurately saved in a digital map, an algorithm can associate the position of the markings as perceived by the vehicle's sensors with the precise information from the map.

Regarding this technology, the German Department for traffic and digital infrastructure (BMVI) has dedicated some sections of highways, namely on the "Autobahn" A9 and A93 to serve as a "digital testing field autobahn" which should help the local car manufacturers to test their technology under real world circumstances. This project includes a high speed, near 5G data transmission along the test site, advanced radar and camera systems along the road but also new, special roadsigns, that have been installed, distanced about 2.5 kilometers apart. These can serve as such landmarks to improve localization. These signs, as shown in Figure 16 show a pattern, especially suitable to be tracked by cameras.

Figure 16 These signs are put up along a section of the autobahn A9 in order to support accurate positioning of vehicles (Picture taken from [43])

Solution: (Visual) Landmark recognition

Advantages of (Visual) Landmark recognition:

+ Delivers a more detailed information about the vehicle's position.
+ Can compute the real position of the vehicle instead of a prerecorded centerline.
+ Detection of road-markings is most probably a part of semantic understanding anyway.

Disadvantages of (Visual) Landmark recognition:

- Requires high definition maps which require much hard drive space to store (might be dynamically loaded over cellular network).
- Dependent on the currency of the map data and the landmarks in real world.
- Adds even more information to the massive data size of high definition maps by the need for landmark data.

High Definition Maps

It shows that digital maps with a high resolution are a key element of localizing a vehicle. The information that comes with these maps is furthermore not restricted to the position of a road and also the distinct lanes, but they will likely hold data like: roadsigns, areas which are safe to drive on, positions of traffic lights, speed limit info, stop lines, specific lane markings and visual landmarks for positional tracking (as described above) and many more.
Obviously this information, that needs to be extremely detailed to allow for an accurate representation of the environment, generates a huge amount of data, that needs to be available

to the vehicle's autonomous driving system when it needs to drive itself through that particular area. Since it seems unpractical at this point in time to save a snapshot of the global road system on the vehicle, some form of caching needs to be implemented. One approach would be to only store the most commonly visited areas on vehicle. However it is also conceivable, that the high definition map data is streamed to the vehicles from servers along the road. In this scenario servers, which are located along the road could stream information about the environment of x kilometers of radius. Thus strategically placed servers, providing information of the immediate surroundings, would need to cover all areas in order to offer service to autonomous vehicles in every possible location. The transfer would require a reliable communication between the vehicles and the servers, which would greatly benefit of the 5G mobile network that is planned to be available in the future. Although a proprietary means of communication is imaginable, it is probably more likely for this technology to rely on cellular network (like Tesla etc. are already using).

Solution: High Definition Maps

Advantages of High Definition Maps:

+ Offers precise data for various algorithm that enhance the quality of positional data immensely.
+ Functions as a global coordinate system that can serve as a common communication port and therefor allows for better communication between vehicles.

Disadvantages of High Definition Maps:

- Generate a huge amount of data due to the high degree of accuracy that is required (+ potential information about landmarks).
- Generation of high definition maps are more costly since exact measurements are required.
- Maps need to be kept up to date and efficiently updated to the vehicles.

3.1.6. Semantic Understanding

In order to finalize the information packed into the world model and offer advanced functionality like estimating predictions of how entities around the vehicle (like other cars, trucks, cyclists, pedestrians, etc.) might move in the near future. [1] describes this part of the system as "the component in which the balance shifts from sensing to perception" (See [1] p. 3), meaning that up to this point the main objective was to capture accurate information about the environment whereas from now on, the focus lies on interpreting this data in order to get its semantic content. In other words: up to this point, components tried to capture the environment, but this component tries to understand what happens in the environment. Tasks in order to do so include: classifying objects in the scene, annotating them with an entity that the system knows and might also have physical models that describe their behavior, thus generating a prediction for its future trajectory, speed, etc.. Furthermore it is used to detect

ground planes (road surfaces), find the road's geometry and determine which areas are safe to drive on. Several of these tasks were already mentioned in previous components, like drivable areas, which might be shared with multiple cars, which indicates that components, that offer themselves to support the task of another component, can work together in order to improve the quality and performance of the whole system.

Challenge: Semantic understanding
Demands for Semantic understanding are:

- Classify objects in the scene and match them with entities which the system knows.
- Predict possible future trajectories, behavior, etc..
- Detect ground planes, road geometry and drivable areas.

Image Recognition

One of the fields that has developed most in the last few years, which is essential for realizing the development of an autonomous vehicle, is image recognition. Since it is a complex task to design an algorithm that is able to reliably detect a wide variety of different objects which themselves come in many forms of appearance, it has taken quite some time to develop a system that does so.
The most commonly used technique to classify objects in a picture is using **machine learning**. One example is Tesla, whose "Vision" system is based on a **deep neural network**, which has been trained to detect and classify important elements of a scene. The exact inner mechanics of these systems is too complex to be covered here in more detail, however it is built up like this: a neural network as it is used in computer vision system consists of so called neurons, which have multiple weighted inputs, a bias and a transfer- as well as an activation function that leads to an output (See [44]). Multiple neurons may form what is referred to as a layer which is important for a special kind of these networks, called "Feedforward neural network". Here information may only flow in one direction, thus the output of the neurons of one layer can only be inputs to the neurons in the next layer. These Feedforward neural networks have three basic layers: input-, hidden (there may be 0 to many)- and output-layer. A Feedforward neural network that has many hidden layers is referred to as a deep neural network. (Informations taken from [44], also review for further information regarding neural networks).
The reason why machine learning and neural networks have experienced a rapid growth in the last years is the computational power of GPUs (graphics processing units). In contrast to CPUs which feature fewer complex cores, optimized for single core performance, GPUs can have hunderds of cores and thousands of hardware threads, that allow for many concurrent operations. Machine learning algorithms can take advantage of this architecture, which means that training a model can be done much faster. This is needed, because the data that is needed to train a machine learning model (referred to as training dataset) often is of

immense size.
Information by the German car manufacturer BMW says, that up to 50 Peta-bytes of training data ($5 \cdot 10^{16}$ Bytes) are used to reach satisfying results. This amount of data equals about 10.000 hours of recorded driving including camera, radar, laser and vehicle information but most importantly labels of what the desired output should be (e.g. when developing a system that should recognize relevant objects in the scene, provide labels with bounding boxes and descriptions regarding the class of the object). This collection of data is processed in a data center with the goal to end up with a trained model, which can be loaded onto the vehicle and to take over that task it was designed for. The resulting model is then supposed to be able to perform its tasks for situations (within a certain margin) that were not part of the training dataset (e.g. identify an object as a car, even though the exact model was not part of the training data set). As stated by BMW, the resulting model has about ten hidden layers of neurons and has about 50 Mega-byte ($5 \cdot 10^6 Byte$). Information from the last paragraph taken from [7].

Further Deep Learning Applications in Vehicles

In the example described above, neural networks can be used to detect relevant objects in the scene. However in autonomous driving, there are many more situations, where machine learning can be used.
One of these tasks is called **semantic image segmentation**, which, instead of assigning bounding boxes around objects, marks different areas of the frame on a higher level. These might include an area marked as grass, road, trees or parking lot. An example of such a semantic segmentation is shown in figure 17.

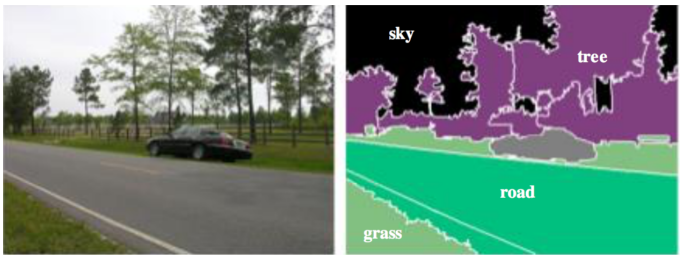

Figure 17 A visualization of semantic segmentation, the different colors layered over the original image show different regions that are of importance. (Image taken from [44])

While the approach described above only takes one frame at a time to detect objects in the scene, it might also be of great value to use machine learning in order to detect common patterns in traffic on a higher level. Thus a machine learning model might be trained to take in multiple seconds of video in order to detect procedures like lane changes, overtakes, etc.. Obviously these phases needed to be labeled in the training dataset. A combination of the two approaches (object- and maneuver-detection) is also imaginable.

Solution: Machine Learning
Advantages of Machine Learning:

+ No need to develop a complex algorithm for vision tasks, instead provide neural network framework and training data.
+ Performance can improve over time by collecting data that is again used to refine the model.

Disadvantages of Machine Learning:

- In order to generate a reliable model, excessive amount of labeled (!) training data is needed.
- Training a model in reasonable time requires heavy computational power.
- Despite a large training data set, the model might still not deliver the desirable output in every situation.

See: [44] and [7].

This component also includes some form of implementation that models the behavior of other traffic participants (buses, other cars, pedestrian, etc.) in order to correctly react to situations that human drivers can do instinctively. An example therefor might be to conclude whether a pedestrian might cross the road or to judge whether another car yields. The following case shows, how important it is for autonomous vehicles to have a correct model for the behavior of other human drivers.

3.1.7. Google Autonomous Vehicle Accident

Information regarding this accident are taken from: [45] and [34]. On Sunday, February 14th in 2016, one of Google's autonomous test vehicles (often referred as Google AVs) collided with a Bus at a four lane intersection. The report by Google states, that their autonomous cars are often being tested on El Camino Real, a three lane boulevard in the city of Mountain View. This site was chosen, because it covers a wide variety of scenarios like traffic lights, intersections and busy traffic. At the place of the accident, an intersection with Castro Street, El Camino Real has four lanes. The rightmost lane is reserved to right turning drivers, the two middle lanes to cross the intersection in order to continue on El Camino Real as well as a lane for left turning drivers. The rightmost lane however is wide enough to accommodate for two vehicles standing side by side, thus the vehicles on the left may cross the intersection to continue driving on El Camino Real, vehicles on the right side are the ones making a right turn. Google reports, that:

> "Most of the time it makes sense to drive in the middle of a lane. But when you're

> teeing up a right-hand turn in a lane wide enough to handle two streams of traffic, annoyed traffic stacks up behind you."
>
> — [45]

Google further explains, that it has added a feature to the autonomous vehicle that would implement a social norm, in that the car stays on the right side of the lane, to allow other vehicles to pass on the left.

> "It's vital for us to develop advanced skills that respect not just the letter of the traffic code but the spirit of the road."
>
> — [45]

A few weeks after implementing this new skill however, one of Google's Lexus AVs has suffered an accident with a bus.
When approaching the aforementioned intersection, the car was about to make a right hand turn onto Castro Street and thus it got in the proper lane. As the vehicle was programmed to do, it hugged the right side of the rightmost lane as it is wide enough to allow two vehicles to drive side by side. The right side of the lane however was blocked by sand bags. Having detected the obstacle, the autonomous vehicle came to a stop. After waiting for a few cars to pass, it crept towards the center of the lane to and while doing so, collided with an approaching bus with a speed of 15mph which caused body damage to the car. According to Google's report, its autonomous vehicle had detected the bus, but assumed it would yield to the AV since it was ahead of the bus. Thus the vehicle simply misjudged the behavior of the bus driver. The test driver who had supervised the decisions of the AV also saw the bus approaching from behind, but he also assumed that the bus driver would yield and thus allowed to car to proceed. Thus the accident might as well have occurred with two human drivers.
The fault in this particular case lies within the "semantic understanding" component of the vehicle, since its responsibilities includes the prediction of likely future behaviors of the vehicle. To be more exact, the component within semantic understanding that was not judging the bus correctly was the "human model". This part of semantic understanding is designed to model and predict the actions of human driver as precisely as possible. Therein lies a substantial challenge for autonomous vehicles as they need to cope with non autonomous and sometimes unreasonable, human drivers.
No person has been injured and there was only body damage to the vehicles. Google has modified its software in a way that the car now assumes that large vehicles like buses or trucks are not as likely to yield than smaller vehicles.

3.1.8. World Model

Finally, all information that was gathered and processed in the previous components is stored and represented in a so called World Model. This collection of information holds all the data of the environment which the ego vehicle perceived.

Challenge: World Model
Challenges of the World Model are:

- Holding as much information about the environment as possible.
- Runtime performance that allows for real time lookups on data from the model.
- (Optional) Advanced information (like trajectories or future position estimations of objects).

Therein it mostly only models the external environment (surroundings) and not the state of the cars internals. While a combination of both internal and external model into one is possible, they exist rarely, since implementations that separate the two states are more optimized. Other differences in the kind of world model lead to a distinction between **active -** and **passive - world models**.

Active and Passive World Model

Information about active and passive world models are taken from [1]. A **passive world model** is in a way passive, that it does not include any semantic information about the environment. It might provide an array of objects around the vehicle, along with their positions and speeds, however this data is not analyzed in order to classify objects in any way. Not knowing to what class of traffic participant a sensed object belongs to, it is also not possible to take physical models into account that offer an estimation of future behavior.

Solution: Passive World Model
Advantages of a Passive World Model:

+ Less computation required, since sensor data is not semantically analyzed.
+ Less complex system, since semantic components need not be implemented.

Disadvantages of a Passive World Model:

- Lack of semantic information.
- No estimation for future behavior as in trajectory, speed, position, etc..

An active world model on the other hand includes all the previously mentioned information,

which the passive model does not offer. This brings the advantage that accessing components in decision and control might query an estimation of the state of an object at a future point in time.

Solution: Active World Model
Advantages of an Active World Model:

+ Classification of objects in the scene.
+ Kinematic information about objects in the scene.
+ Estimation of future behavior.

Disadvantages of an Active World Model:

- Analysis of scene requires computational power.
- Physical, kinematic models need to be implemented for a variety of object classes.

This component often is referred to by different names, depending on the source. Waymo for example describes their world model as a ”Three dimesional Map”. Furthermore, Waymo uses an active world model, estimating possible behaviors for every object in the scene.

Local Dynamic Maps

Informations regarding local dynamic maps from [46]. In order to enhance the capability of efficient communications between vehicles, one attempt to create a standard for sharing information about the environment is called **Local Dynamic Map**. The main goal of this structure is to provide current, highly accurate traffic information of all relevant objects in the environment. Such type of map consists of four layers which differ from one another in the frequency in which updates occur, thus the higher the layer is, the more dynamic its informations are.
The first and lowest layer is static and holds map information as do navigational systems or standard, digital maps.
The second layer then includes elements that enhance the quality of localization in form of permanent structures like bridges, lane markings, calibration signs (as has the A9 (see figure 16)) that are used as reference and calibration points.
The third layer contains dynamic information, meaning information of the current state of the street. These range from weather information like fog, danger of aquaplaning, iced roads over broken down vehicles to traffic density.
The fourth and highest layer is highly dynamic and can be updated within seconds. Informations within include the state of traffic lights, the speed of the traffic flow. Also information about emergency vehicles and their trajectory might be shared here in order to more efficiently form a rescue line.

3.1.9. Uber: wrong classification of traffic participant

Information regarding this accidents were taken from: [47] and [34]. Being responsible for kinematic and dynamic model of the surrounding environment, the ”active world model” is a crucial data source for finding a safe trajectory for both, the ego vehicle and its surroundings. What can happen when the kinematics of a traffic participant is estimated incorrectly, can be observed at the incident which was caused by Uber's autonomous vehicle.

On March 18th, 2018, a pedestrian, pushing her bike over the road has been killed by the self driving Volvo XC90. Even though the vehicle was equipped with multiple cameras, radar-, navigation- and even a modern LIDAR-Sensor, the woman was hit by the car and died of her injuries. Shortly before the accident, the vehicle has been driving autonomously for 19 consecutive minutes and approached the pedestrian at a speed of around 70 km/h (43 mph). Just about six seconds before the impact, the radar and LIDAR sensors recognized the woman whereupon the car's sensor fusion system associated the reading as an unknown object. After more iterations, the system classified the victim as a vehicle and finally as a bicycle, as she was pushing one over the street. At this point the world model component started to generate a predictions of future paths of the recognized object. As it has gotten the information that the woman was allegedly a cyclist rather than a pedestrian, a future travel path was predicted, that would be typical of a person riding a bike. Since riding a bike and walking a bike over the street differ greatly regarding kinematic properties, the path was subsequently wrong. 1.3 seconds before the impact, the system recognized, it needed to initiate an emergency braking to avoid colliding with the woman and her bike. However, even though the Volvo was equipped with an emergency brake feature, it did not stop the vehicle, since Uber explained that emergency braking in disabled by default when the vehicle is in autonomous drive mode in order to reduce false positives and risk being rear ended. This decision shows parallels to the neglecting of readings like Tesla did resulting in collisions with stationary obstacles. Thus there was no emergency braking and the Uber test car hit the pedestrian at around 62 km/h (39 mph).

Subsequently, this was firstly an error of the sensor fusion component, as it is its task to classify readings as real world entities like bikes, pedestrians, cars, trucks, etc.. However, since this wrong information was fed into the active world model, it did not predict a accurate enough path for the rest of the system to generate a safe trajectory.

3.2. Decision and Control

3.2.1. Trajectory Generation

Now that the vehicle has a three dimensional -depending on the implementation (taking saved, past events and predictions for possible future trajectories of other objects into account) even four dimensional- understanding of its surroundings, the perception part of the system is done. The main task from here on is to generate a trajectory, that is optimized for predefined attributes, like efficiency. A (dynamic) trajectory is defined as a path in space, containing information about time, which an object follows. It thus can be characterized by a function $\vec{f}(t) = (x(t), y(t), z(t))$, where $f(t)$ returns the object's positional coordinates in $\Re^3$

at a given time t.
There exist many different algorithms that have the goal to generate a safe trajectory for a given traffic scene, two of which are described in the following. The algorithms that are going to be covered in this paper, have been chosen due to their different nature, one relying on a start and end point to generate a polynomial that describes the trajectory, while the other one uses interpolation.

Motion planning using a conformal spatiotemporal lattice

One possible approach for a motion planning algorithm is proposed by [48], which will be explained in the following. It overlays a lattice over the scene and then uses the discrete points to connect them and calculate a smooth connection between them.
Firstly, a grid (so called lattice) is generated around the center of the lane in which the ego vehicle is currently in. The center line is described by a sampled function $[x(s)y(s)\theta(s)\kappa(s)]$ where s is the arc length. The points of the grid are then defined in $p(s,l) = [x(s,l)y(s,l)\theta(s,l)\kappa(s,l)]$ where l describes the lateral offset from the centerline and κ the curvature of the lane so that $\theta(s,l) = \int_0^S \kappa(s,l)$. Using specific formulas for each of the elements of $p(s,l)$, a discrete lattice (i,j) is generated, where a positive and negative l generates lattice points on the left and the right side of the centerline respectively.

In order to generate a path between two discrete points on the lattice, [48] uses a cubic polynomial spiral. This means that for a cubic function κ (the curvature of the road) coefficients can be determined, that connect any two points. Using a simple kinematic model, a function is established, that describes the endpoint of the path, where κ and θ can be computed in a finite number of steps, however the x and y of the endpoint require for two Fresnal integrals to be solved numerically, for which the authors of [48] use the Simpons method. Following the gradient of the Jacobian matrix of the endpoint state vector, the distance between the endpoint x_p and the desired destination point x_{dest} is minimized. [48] calculates the Jacobian "using a symbolic differentiation with respect to the parameters of the expression used to calculate the endpoint" (See [48], p. 3). Newton's method is applied to the Jacobian matrix iteratively until the error $x_{dest} - x_p$ is sufficiently small.
The result, after multiple iterations, is a set of paths, that are possible for the vehicle to follow and represent a collection from which to choose the most suiting path, which properties can be chosen by a meaningful cost function. Since there is not yet a time dimension to this paths, the start time and velocity is taken and a constant acceleration is added over the length of the path.
Further, [48] applies techniques to improve the runtime performance of the algorithm. While a state lattice is a proven choice for non changing environments, one should not simply add time and velocity, since it contributes to a rapid growth in the size of the lattice. One of the ways, it was addressed in the paper was a reduction of discrete speed values.

Solution: Conformal spatiotemporal lattice
Advantages of Conformal spatiotemporal lattice (as proposed by [48]):

+ Generates a trajectory that can avoid static and dynamic obstacles.
+ Can be accelerated by a GPU.

Disadvantages of Conformal spatiotemporal lattice (as proposed by [48]):

- Does not account for traffic elements (like signals, speed limits, etc.)
- Limited acceleration profile due to reduced speed values.
- Does not account for gentle acceleration.
- At low speeds, the maneuverability of the vehicle can be limited, when constructing paths using a cubic polynomial spiral.
- Expects an end point as a destination that first needs to be determined.

See: [48]

Action annotated trajectory generation using cubic interpolation

Another approach, that takes traffic elements, such as traffic lights and speed limits into consideration is proposed by [49]. They divide responsibilities into Cartography, Mission Control, Behavior and Controller. Cartography translates to the perception part of the system as in chapter 3.1, the controller takes over tasks, that will be depicted in chapter 3.3. Mission Control on the other hand is responsible for tracking target points, whereas Behavior does the trajectory generation itself.
They use a modified environment map that includes Akima splines. Furthermore, a ”Trajectory Data Format” is introduced that includes three layers, first of which stores the trajectory as a spline. The second one consists of a graph, who’s edges contain information about like speed limits or track width. One spline may span over multiple edges of this layer, depending on the distance that is supposed to be processed ahead. The final layer is called ”action track” and as the name suggests, it stores information about the action that is necessary at the respective position (e.g. traffic light, lane change, give way, etc.). This format is supposed to contain all the information of the final trajectory that is supposed to be executed. Figure 18 visualizes, how these layers might look like.

From here on, the actual trajectory is generated. Since the algorithm also accounts for obstacles, a mechanism for swerving around them is added afterwards.
The method used to calculate a trajectory, is to refine a comparatively rough graph in multiple

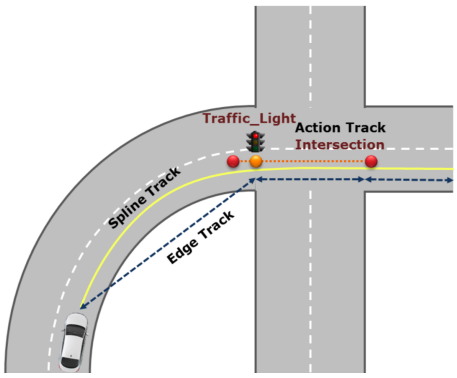

Figure 18 In this graphic from [49], you can see how one scene is represented in the three different layers of the ”Trajectory Data Format”. The spline is shown in yellow, the edge track as a dotted line and action track elements, in this case a traffic light. (Graphic taken from [49])

steps to end up with a smooth travel path. The fist step and the roughest version of the path is a so called **MacroPlan**. Its graph like structure allows for an easy depth search, starting from the current position of the ego vehicle. The distance that is planned ahead, is defined as $d_{plan} = max(v \cdot d_f, d_{min})$ with v being the speed of the vehicle. An example for a ”Macro Plan” is shown in figure 19. Depending on the overall destination of the vehicle, one of the depicted directions can be chosen and refined.

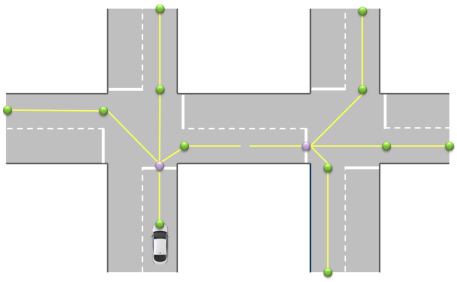

Figure 19 This macro plan from [49] shows the graph like structure of the ”Macro Plan”. The car shows the current position of the ego vehicle. Furthermore, the vertices are differentiated between junctions (violet) and normal vertices. (Graphic taken from [49])

This ”Macro Plan” is then sub-sampled into points, keeping a common distance. This step is called **Point Sampled Plan**. These sub-sampled points follow the spline, that is kept in the digital map, which usually equates to the center line of the road. If obstacles or other factors are recognized, that forbid the following of the predefined spline, the points of this plan might be relocated, in order to avoid a collision but still advance towards the desired direction (See figure 20). It is meant to later be used as a ”support pillar” for the spline interpolation. [49] also stresses, that, since at this point the ”Macro Plan” is fully calculated, its vertices and edges can be fed into the ”Edge Track” (second layer of the final trajectory information) concurrently. The same holds true for the ”Action Track”, since ”static actions are added when stationary events are encountered during the iteration.” (See [49], p. 3).

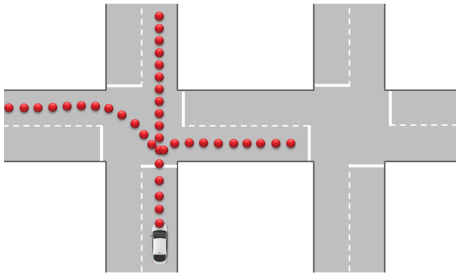

Figure 20 This macro plan from [49] shows the graph like structure of the "Macro Plan". The car shows the current position of the ego vehicle. Furthermore, the vertices are differentiated between junctions (violet) and normal vertices. (Graphic taken from [49])

Finally a spline is generated using cubic polynomial interpolation and tested whether the vehicle is able to perform it. Obstacles are avoided by additional, alternative trajectories that are generated and rated by a cost function. [49] differentiates between "swerve-" and "lateral shift" maneuvers, each with respective parameters to describe the maneuver. These alternative routes originate in different point sampled plans and are each evaluated by closer inspection after realizing them in a "Micro Plan". Figure 21 how the finished spline looks like.

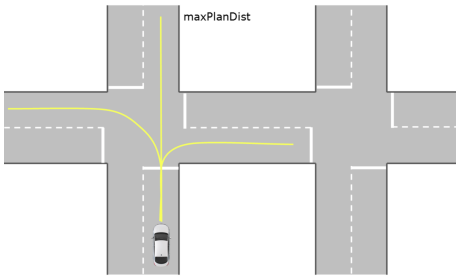

Figure 21 Having been interpolated, the splines can be added to the set of possible routes. (Graphic taken from [49])

Solution: Action annotated trajectory generation using cubic interpolation

Advantages of an Action annotated trajectory generation using cubic interpolation (as proposed by [49]:

+ Takes "Actions" like traffic lights and many other events into consideration.
+ Avoids static and dynamic obstacles.

Disadvantages of an Action annotated trajectory generation using cubic interpolation (as proposed by [49]:

- Requires high definition digital maps.
- Relies on templates for lane changes and a predefined list of "Actions". While this seems to work in practice, it might lack the flexibility of other approaches.

See: [49]

3.2.2. Reactive Control / Emergency Mechanisms

While the generation of a feasible trajectory might resort to an active world model, taking multiple likely future predictions into account, it is still possible for unforeseen events to suddenly occur, like a person stepping onto the street out of a previous blind spot. A human driver would act reflexively in such a situation to avoid an accident. Thus adding dedicated systems that can quickly react to such situations makes sense.
One might think that the autonomous driving system should be able to handle such occasions, however according to [1], the loop of such an emergency reaction system is ”at least an order of magnitude faster than the nominal system loop”. This advantage of these emergency backup components come from the fact that they only evaluate the data of a few sensors and do not need to perform nearly as many computations as the big system loop. Still both components run in parallel, while the emergency obviously has the power to override the output signals of the autonomous driving loop e.g. in order initiate an emergency braking.
When the nominal system loop runs at sufficiently high frequency, it might also be possible to get rid of the dedicated emergency system. Nevertheless functional redundancy is an argument to still keep such a system, since it is not too big of an imposition.

Solution: Redundant emergency mechanisms
Advantages of Redundant emergency mechanisms:

+ Run at a much higher frequency than the nominal autonomous driving system, which allows for a faster reaction in case of an unforeseen event.
+ These systems already exist, as they are sold today in similar form as driver assistance systems.
+ Add functional redundancy to the system, which is a desirable attribute in a safety relevant system.

Disadvantages of Redundant emergency mechanisms:

- Still require additional computational time.
- Since they may override the nominal output of the big loop, false positives must be kept at a minimum.

See: [1].

3.2.3. Further Possible Components of Decision and Control

The paper of [1] furthermore lists even more functionalities that fall under ”Decision and Control” which are not going to be covered in detail in this paper, however are worth mentioning (since these components are taken from [1], so is the info in the following two sections.):

Energy Management

This part is of particularly important role in electric or hybrid vehicles, since they suffer more from a higher energy consumption than regular gas engines. Car manufacturers of electric vehicles are struggling to extend the range of their cars in order to minimize the inconvenience of a longer charging time compared to gas engines, and thus utilize every possible way to make a vehicle more efficient. Next to weight and drag reduction, this component might be of value as it can be used to optimize the efficiency by determining optimal acceleration profiles, but also when to just let the car roll or activate regenerative braking. [1] also mentions, that the powerful computational units, required for autonomous driving impose another energy consumer in the system. Due to the high number of energy consumers in the system, this component may also communicate with ACs, entertainment systems or lights.

Diagnostics and Fault Management

Logging, Diagnostics and Fault Management is part of every bigger and well designed system and thus is compulsory for a complex architecture like an autonomous car. [1] however also stresses that is in this case, it also is responsible to monitor the overall system in order to detect anomalies, but also asses when to initiate a handover to the driver.

3.3. Vehicle Platform Manipulation

The remaining part in the sense-plan-act loop is the final execution of the calculated trajectory. Since determining the values for acceleration, steering wheel position and feeding them to the actors is a rather simple task, as it solely must be able to follow a predetermined path, in the paragraph ”Trajectory Execution” the focus will be on giving some examples for common communication buses, used in vehicles.

3.3.1. Trajectory execution

As a result of the ongoing integration of computer components into vehicles, the concept of ”Drive by Wire” has come up and has been implemented to various extent. The main idea is to eliminate physical connections between steering input devices like the steering wheel or the accelerator and replace it with a computational system, consisting of sensors, an evaluation logic and actors. In such a system, the sensors pick up on the inputs of the driver, digitizes them and feeds them to the evaluation logic that might use or even alter this data as desired before it sends output signals to the actors that then execute the command. In the case of ”Drive by Wires” these actors often already exist in some form, e.g. the steering in form of power steering or shifting with an automatic gearbox.
While ”throttle by wire” is common in modern cars, more safety relevant systems most of the times rely on a physical connection like braking. ”Steer by wire” is still also uncommon at this point in time, with many brands choosing hybrid systems (For drive-by-wire, see [50]).

3.3.2. Inter ECU bus-systems

However, since the computer architecture of a vehicle has evolved to be very decentralized, with the number of electronic control units (ECUs) steadily increasing, the need for reliable means of communication between ECUs has resulted in multiple types being developed, each optimized for its application. The main bus types, that are used, are: CAN, MOST, FlexRay and Ethernet (The following depiction of bus systems in vehicle contains information from [51]).

CAN (Controller Area Network) is a worldwide accepted standardized bus mostly used for the drive-train, chassis and body. Its short data length of 8 Bytes at only 100+ messages per seconds makes it unsuitable for tasks that require a high bandwidth, as CAN can only provide around 10kBit/s to 1 MBit/s. Messages are sent as a broadcast, so every component connected to the bus can pick which message and data is important for it. Typical lengths of a CAN bus lie between 40 meters and reach up to 1km (But: the length of the cable impacts the data transmission rate) (See [7]). However, it features a high electromagnetic compatibility (EMC). It was first introduced by Bosh in 1990 (See: [51]).

FlexRay is one of the latest standards, as it was introduced by BMW in 2007. At the time, it solved various problems with CAN, since the old bus in combination with the ”OSEK” standard did not satisfy demands in the area of determinism, fault tolerance and data rate (See [7]). FlexRay solved these problems by featuring a higher data rate of up to 10 Mbits/s (when used in an ”active Star” typology), determinism, fault tolerance (e.g. in form of error detection). It actually consists of two buses, both of which can transmit with up to 10Mbits/s (Connection to second channel is optional) (See [7]). As a transmission medium, copper or optical fibers are used. Typical use cases include steering and braking (See: [51]), since its fault tolerance mechanisms allow for use at safety relevant tasks. However, it is not mostly used, since development costs are still high. Being integrated in more vehicles, it will render the FlexRay the most cost effective solution.

MOST (Media Oriented System Transport) , as the name suggests, is mostly used for media and infotainment tasks. It uses optical fiber exclusively (resulting in a great EMC), arranged in a circle typography (See [7]). Since the transmission time of max. $\frac{n}{2}$ (with n being the number of nodes in the circle) drops down to max. $n - 1$ when only one node fails, it is not suitable for safety relevant tasks. However, it is optimized by design for transmitting streaming data, as well as package- and control-data. Hence its applications are of infotainment nature, such as: GPS, radio, video streaming (e.g. backup camera(s)), etc.(See: [51]). Data rates reach up to 150MBits/s.

Ethernet is one of the most commonly used networking means, thus it has also found its was into the automotive domain. With its currently very little usage (1%), it is expected, that it will quickly grow to 40% in the next two years (See: [51]). Its high data rates of

(currently) up to 10GBits/s come at exactly the right time to support the amount of data that is needed for autonomous driving tasks. The reason for the low usage of Ethernet in vehicles, is mainly the comparatively high cost of the shielded cables, that are necessary to provide the demanded EMC in order to deliver an acceptable fault tolerance on a physical level. Twisted pair cabled at 100MBits/s have been used more often for media oriented tasks.

3.3.3. AUTOSAR

However not only hardware buses have emerged to a industry standard, but also the software. Well developed system architectures keep a ”separation of concerns” (SoC) in mind. This principle basically describes, that the system as a whole is divided into smaller components, each with its unique function. This allows for distinct parts of the software to be replaced with another implementation and also makes re-use of software easier. Since there are many players in the automotive industry, it is prone to many different software architectures and coding standards, which adds another challenge to overcome. In order to reduce these variations, many companies from different tiers of the automotive industry have collectively founded a standard that is supposed to minimize this issue. Its product was the so called **”AUTOSAR”** (**AUT**omotive **O**pen **S**ystem **AR**chitecture. Their main goal is to ”improve complexity management of integrated E/E architectures through increased reuse and exchangeability of SW modules between OEMs and suppliers” ([7] lecture 9, p. 5). It was first developed by its nine ”core partners”: BMW, Bosh, Continental, Ford, Daimler, PSA, General Motors, Toyota and Volkswagen. (The following information about AUTOSAR platforms are taken from [7])

Classic Platform

At this point in time, there are two important platforms, designed by the AUTOSAR association, first of which is called **Classic Platform**. This one is the standard platform which is used by many real time systems in vehicles. It offers hard real time with deterministic behavior, supports most of the aforementioned bus systems and some few security mechanisms. With its layered structure, it allows for software to be interchanged between systems, as it no longer needs to be written for one specific ECU. The layers are depicted in figure 22 (Graphic is taken from [2]).

The **Microcontroller Abstraction Layer** is intended to provide an interface to higher layers that is independent of the microcontroller. Its implementation is dependent on the microcontroller, as it includes proprietary drivers.

The **ECU Abstraction Layer** has a similar task as the Microcontroller Abstraction Layer, as its main purpose is to make higher software layers independent of the ECU hardware,

[2] https://mgc-images.imgix.net/embedded-software/autosarmcal-E0C74E5B.png?q=80&w=1920&fit=max)

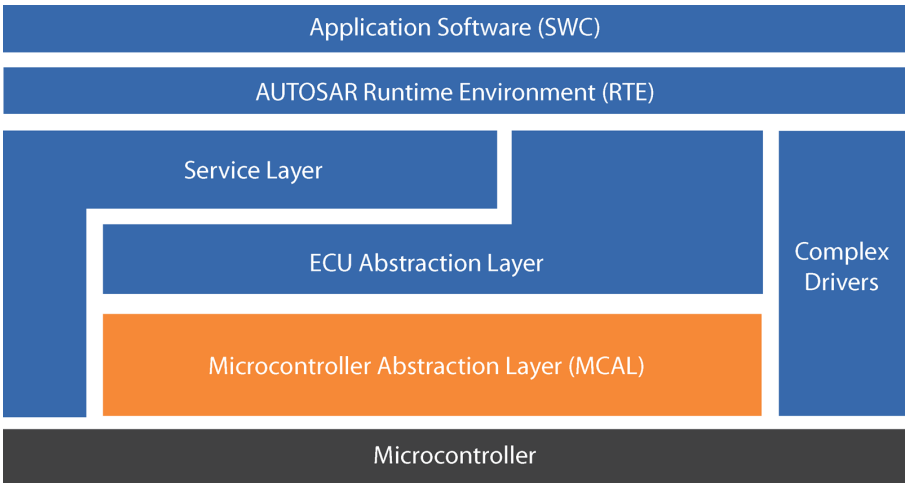

Figure 22 This figure shows the different layers of the AUTOSAR Classic platform architecture. Picture taken from [2]

that is used. It stores drivers for external devices and thus makes interaction with peripherals independent of their location in the system and connection (like port/connector type).

Representing the highest level of the basic software, the **Service Layer** provides basic services, like: OS functionality, network communication within the whole vehicle, memory management, diagnostics, etc.. As suggested by figure 22 it is mostly independent of the microcontroller and the ECU hardware.

In order to allow for communication between the runtime environment and the microcontroller, the **Complex Drivers** layer connects both of these layers. This communication path is useful, when implementing functionalities that are not accounted for in AUTOSAR, or those, that have very strict timing boundaries. This layer can be microcontroller, ECU and application dependent.

The **Runtime Environment** is the direct interface to the application software. From here on upwards, the architecture shifts from layered to component style. These components can communicate with each other, even with other ECUs via the runtime environment. This layer makes the application software independent from the system, that it is deployed on, which allows for great reuseability, whereas the component style accounts for separation of concerns.

Adaptive Platform

However, technological progress and always changing requirements, made it necessary to extend the classic platform, in order to satisfy the demands, that arise with future features. Hence, AUTOSAR has announced, that the **Adaptive Platform** will be available to use (series start expected in 2019). Its main improvement is the support for high performance hardware, like SoCs and GPUs, which, as mentioned before, accelerate many computations that are needed for autonomous driving. It also relies on Ethernet rather than CAN or FlexRay and

features more security features like IPsec.

These measures simplify not only the development process, but also the collaboration between companies, OEMs and suppliers.

4. Development of Software for autonomous Vehicles

The development process itself is another challenge that needs to be resolved when working on software projects in the field of autonomous driving. Common regulations, like AUTOSAR help reduce boundaries, both within a company and between multiple collaboration partners, such as suppliers.
The automotive industry heavily relies on suppliers, as the software development tasks are greatly outsourced. This means that when a vehicle is planned, requirements are analyzed, possible technologies assessed, which results in a specification. This specification is then implemented by the so called **Tier 1 supplier**, which is responsible for implementing an isolated part of the overall system. Suppliers of Tier 1 are subsequently called **Tier 2**, and so on. In this context, the automotive manufacturer that delegates work to the "Tier 1" supplier is called a **Original Equipment Manufacturer** or **OEM**. This delegation structure, in combination with the software focus of autonomous driving systems, offers an optimal position for companies, which are usually not known for developing automotive systems (This paragraph contains information from [7]).

4.1. Companies working on autonomous Software

With that being said, the companies that lead in the development of autonomous driving systems are mostly traditional car manufacturers. Figure 23 (Source: see [1]) gives a sense of scale of the development of different companies, measured by filed patents that were filed by the respective company. Apart from the car manufacturers, Bosh and Continental appear as two companies that have been producing vehicle parts for many years. At the end however, Google takes tenth place as the first party who, in the past, has not been in the automotive sector to that extent. It is also noteworthy, that more than half of the companies in the top ten, are located in Germany, with Audi leading as the highest ranked car manufacturer on place two, behind Bosch.

However since filed patents do not describe the exact progress of development, some companies researching autonomous driving that are otherwise rather associated with different fields and their role will briefly be covered. Please keep in mind that this is no exhaustive list, but rather a selection by public popularity, size of the company, autonomously driven miles, etc..

Waymo (Alphabet) was originally a project by Google with the goal to advance technol-

[1] IW Köln. Number of autonomous driving patent filings worldwide between 2010 and July 2017, by main company. `https://www-statista-com.eaccess.ub.tum.de/statistics/623636/autonomous-driving-global-patent-fillings-by-company/` (accessed 11/13/18, 10:35 AM)

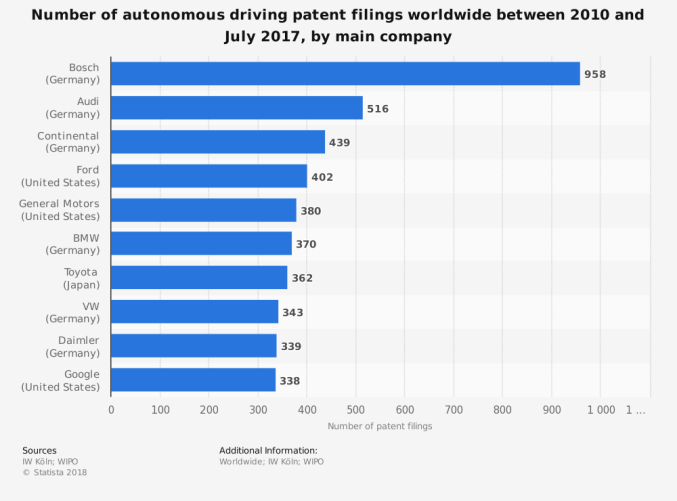

Figure 23 This figure orders different companies regarding their filed patents in the field of autonomous driving. Graphic taken from [4.1]

ogy for autonomous driving. By now, they have become a company of their own, under Alphabet, which is also Google's parent company. As of October 2018 they reached an impressive 10 million miles driven by their autonomous cars, according to own details "largely on complex city streets" (See: [52]), which come alongside 2.7 billion miles that were simulated. Having started on Toyota's Prius cars, the company later added multiple Lexus RX450h's. In 2015, Waymo has introduced a concept vehicle, called the "Firefly", which key features were fully autonomous capabilities to the extent, that it lacked any steering wheel and pedals. When the Chrysler Pacifica was added to the fleet, it was the first mass-produced vehicle that had a fully integrated hardware equipment, designed by Waymo. These exact vehicles were then driving on public roads by 2017 for testing purposes, with no driver in them. In 2018 Waymo promised a collaboration with Jaguar (See: Figure 24, picture taken from: [2]), including 20.000 cars of the model I-PACE, to be modified to drive fully autonomously. (Information about Waymo taken from: `https://waymo.com/` ([52]) and sub-pages)

The Chinese company **Huwaei** has lately announced a collaboration with Audi to develop a Level 4 autonomous vehicle. A prototype of an Audi Q7 was showcased at October 11th 2018 the Huawei Connect 2018, after having agreed on a strategic plan for the joint venture in July 2018. Next to an extended sensor suite, the SUV also houses the so called "Huawei's Mobile Data Center" as a computational unit (Information taken from [53]).

However, Huawei is not the only mobile phone manufacturer that researches autonomous driving. **Samsung** as well has confirmed to be working on technology for autonomous driving (See: [4]).

There are many indications suggesting, that **Apple** is also researching autonomous systems. While being very secretive about it, legal documents exist, such as DMV approvals

[2] `https://storage.googleapis.com/sdc-prod/v1/press/WaymoIPACE-Front.jpg`

Figure 24 The electric Jaguar I-PACE with Waymos sensor suite, with fully autonomous capabilities. (Picture taken from [2])

for 66 cars to test autonomous software or obligatory accident reports for occasions where self driving cars are involved, are linked to apple (See [3]). It is yet unclear, whether the company plans on designing a complete vehicle or only the autonomous system. Apple CEO Tim Cook vaguely confirmed these assumptions.

> "In terms of the autonomous systems, what we've said is we are very focused on autonomous systems from a core technology point of view. We do have a large project going and are making a big investment in this. From our point of view, autonomy is sort of the mother of all AI projects, and the autonomous systems can be used in a variety of ways, and a vehicle is only one. But there are many different areas of it, and I don't want to go any further with that."
>
> —Tim Cook [4]

4.2. Software Development Process

Next to the technical aspect of an autonomous system, the development process itself is another challenge that needs to be agreed on, even before the first code is written. The specific structures of the automotive industry with many OEM - Tier 1 relationships and the overall complexity of the whole systems already point toward a set of standards and development models which might suite this instance best. Other regulations are of legal nature and are required to be fulfilled in order to offer the system for sale, at the end of the development.

[3] https://www.cnbc.com/2018/08/31/apple-reports-first-autonomous-vehicle-collision-in-california.html
[4] https://www.cnbc.com/2017/08/01/apple-earnings-call-ceo-tim-cook-hints-at-self-driving-car-project.html

Challenge: Development process
Challenges of the Development process are:

- Overall Goal: Support the software development in a way that it results in a finished product.
- Plan-ability: You want to be able to put a date on when the software development is finished. This includes deadlines for certain milestones, etc.
- Agile reactions: Be able to react to changes in the requirements, used technology, delayed parts, etc.
- Re-usability: Develop the software in a way that allows for easy exchange of implementations and makes it possible to re-use the implementation in another project.
- Re-use code when possible to save time. Certain conditions may require a new implementation (e.g. a more efficient one).

These demands might not all be completely fulfilled, since some of them contradict each other in some way, like plan-ability and agile development. Nevertheless, in the following, some models and coding styles are depicted that try to tackle these problems.

4.2.1. Separation of Concerns

The concept of **separation of concerns** describes a programming style, that mainly affects the architecture of the software. By organizing code structures in a way, that each functionality can be treated as a standalone black box, makes it easy for programmers to implement the functionality. ”Black box” in this case means, that you only need to know the API of the black box in order to implement it.
As an example: imagine you want to add some sort of encryption to your software. With the black box approach, you only need to add the encryption component to your project, and, in order to use it, fall back on a few functions like generateKey(), encrypt(message: m, key k) and decrypt(cipher: c, key: k). This allows you to integrate the encryption component, without knowing its inner workings. You can treat it as a ”black box”.

However separation of concerns mostly requires you to divide your code into components, each responsible for a specific functionality. This means, that the components are interchangeable and have low coupling between them and the application software. This increases the re-usability of the components and easy replacement of the component, without having to alter the code in the other components, but with minimal changes in the application software.

In practice, these components are located inside a container, that takes over certain tasks for

the components. The container itself, has the OS abstraction layer underneath it and utilizes it for its tasks. Since components only include clearly separated, technical aspects and no vehicle specific information, and the hardware that lies underneath is heavily abstracted, they can be easily re-used. (Last two paragraphs: See [7]).

Solution: Separation of Concerns
Advantages of separation of concerns:

+ Re-use saves time, money, human resources, etc..
+ Exchanging components is much easier.
+ Integration of components is simplified, since they can be treated as black boxes.
+ Allows for a modular system architecture.
+ Predefined API makes collaboration between companies and Tier - suppliers easier.
+ Results in an easy-to-understand code structure (improved readability).

Disadvantages of separation of concerns:

- Needs to be kept in mind when designing the system architecture.

See: [7]

4.2.2. Standardized Demands for Automotive Software

Besides practicality and effectiveness, safety must not be overlooked and thus plays a key role in the development process. So much so that there exist specific legal norms and standards that must be adhered to during the software development. These are mostly captured by ISO and DIN norms. Some of them, which have to do with the development process will be explained.

DIN EN ISO 8402 / **ISO 9000** are about quality in a general sense, thus it is not explicitly designed for the automotive industry. Still it is used to ensure quality. The older ISO 8402 has been replaced by the newer ISO 9000 (which meanwhile has also been replaced by 9001), however 8402 is still often used as a guideline to quality. It describes quality as: ”The entirety of characteristics of a unit regarding its suitability to fulfill defined and assumed requirements”, whereas ISO 9000 redefined quality to be: ”The degree to which a unit fulfills inherent requirements” (both definitions directly translated from: [54]). ISO 9000 furthermore includes definitions for quality management, customer, interested parties, errors and requirements. It contains the definitions of these terms as well as methods that support the development process in various stages.

ISO 26262 is specifically designed for automotive applications as it addresses safety rel-

evant systems of road vehicles. Some important points are, that software need to be developed with best available technology, otherwise liability claims may be raised against the developers. Safety standards help to capture the best available technology. Also, ISO 26262 define frame conditions for all phases of the life-cycle, depending on the safety-relevance of the system, that need to be considered for each process. Therefor, so called ”Automotive Safety Integrity Levels” (ASIL) are defined to have some kind of classification for the systems: (Information of ISO26262 taken and translated from [7] and [54])

ASIL A-D The different ASIL levels describe, how safety relevant a system is. There exist four levels, A through D, where D is the highest level. The classification of systems is done considering potential severity when malfunctioning, probability of failure and controllability in case of a fault. (Information of ASIL levles taken and translated from [7])

Solution: Standardized Demands
Advantages of Standardized Demands:

+ Helps to structure the development process, as predefined frame conditions need to be adhered to.
+ Enhances focus on safety relevant features and prevents errors in the development process.
+ Provides a standardized protocol that all manufacturers must obey. This forces a comparable measure of safety awareness.

Disadvantages of standardized Demands:

- Requires attention when designing and executing the development process.
- Might restrict companies in some ways by forcing them to stay within the predefined limits.

See: [7]

4.2.3. Development Strategy

Development strategies are of great importance that need to be chosen with care as they heavily influence the flexibility and progression of the development process. A project without a predefined strategy is nowadays unimaginable. There exist many different ways to organize the development process, each of which has its own advantages and drawbacks, thus it is necessary to asses, which aspects are important for a specific project. The following selection is just a short description of some of the most commonly used models, some of which are considered outdated and not flexible, others on the other hand are comparatively new and designed for agile development. Since these models are very common and descriptions can be found easily, they will be described only shortly, but will rather be assessed by their usefulness for software development in an automotive context.

Waterfall Model

The waterfall model is the oldest of the models which are covered here. In some way it can be seen as a ”legacy” model as it lacks many benefits of more modern approaches. When choosing the waterfall model as your software life cycle, you start with requirements-engineering, followed by -analysis, afterwards you take care of system- and object design. Finally, after the implementation, you run unit-, integration- and system-tests, to validate your software. (Phases taken from [55]). Even though this model allows for planning deadlines and milestones, it lacks many other important aspects, like agility and the first outcomes are rather produced at the end of the project.

V-Model

The V-Model can be seen as an augmentation of the waterfall model in which some of its weaknesses are tackled. They differ in a way, that all steps from implementation on forwards are already defined before the implementation. This means that unit-tests are written when working on the object design, integration-tests during system design, system-tests during the requirements analysis and the requirements engineering includes the generation of acceptance-tests. This way, you start with writing tests before implementing the software and then validate it. Advantages hereby are regular results / deliverables in each phase while keeping the plan-ability of the waterfall model.
This model is commonly used in the automotive industry due to the aforementioned properties. The heavy OEM and supplier structure results in a V-Model, in which the OEM writes a requirement analysis and a specification for what a given system should be able and how it needs to function. All the next steps from software requirement analysis to software system tests are taken over by Tier 1-n suppliers, while the final integration- and acceptance-test is done by the OEM. (Information regarding this model taken from [7])
This way, the plan-ability is given, which is required for a project, which has a very long development time of many years. Also the cooperation with suppliers can be handled nicely. The only drawback might be, that this model is not particularly agile and thus cannot react quickly to changing requirements or technology.

Scrum-Model

The most agile one under these three model is the Scrum-Model. In contrast to the V- and waterfall model, it organizes the projects in so called sprints instead of phases. At the beginning, the requirements are assessed and collected in a product backlog, which then contains all the functionality, which the finished product should have. Then, during development, a subset of requirements from the product is selected and collected in a sprint backlog. All requirements in the sprint backlog are implemented during a so called sprint, that spans over

a predefined time, most of the times 2 weeks to one month. In the so called daily scrum meeting, every developer on the project explains his or her progress and possible problems or blockages.
Advantages of this model are: high agility, since you can react quickly to changes between sprints. Also, deliverables are generated after each sprint. It lacks a certain degree of plan-ability in favor of agility, which marks one reason, why the V-Model is still more popular in the automotive industry than the scrum model is. (Infos concerning scrum taken from [55])

Solution: Development strategies
Advantages of Development strategies:

+ Practically un-dispensable in modern software development processes.
+ Introduces a structure to the development process.
+ Structure and organization in the project helps to avoid errors and helps to keep track of the project status.

Disadvantages of Development strategies:

- Concrete model to use must be chosen with care.
- Additional organizational overhead.

See: [7], [55]

4.3. Quality Management and Testing

Having taken a look at ways to organize a project to minimize errors, another very important step needs to be discussed. Information in this chapter 4.3 is taken from [7].

4.3.1. Verification and Validation

Obviously, a crucial part of developing automotive and most of all safety relevant software, is testing. Both verification and validation are of decisive role when realizing a project in a scope of a vehicle. These two terms differ in an important aspect. While verification asks the question: ”Am I implementing the component correctly”, validation instead checks, whether: ”Am I implementing the correct component”. Second of which questions should already be carefully assessed when planing the project.
The remaining verification of the system finally needs to be covered by concrete software testing. This is practically mandatory, as some norms depend on a security rating, which take failure rate and severity in case of an error into account, for which tests may be taken into consideration. Furthermore it is in the interest of the OEM to test produced software

and the overall system exhaustively, in order to deliver not only a working, but also an easy and pleasant to use product, with respect to quality. Different kinds of testing and how car manufacturers make this process more efficient, will be explained in the following.

4.3.2. Different Kinds of Testing

[7] differentiates between four mayor kinds of test cases:

- Functional testing (e.g. checking functional requirements)
- Compatibility testing (e.g. checking for interoperability)
- Reliability testing (e.g. checking behavior in case of fault injections)
- Performance testing (e.g. checking timing constraints)

There are many methods of testing software, that fall under these categories. They can be divided into ”Black Box” and ”White Box” testing. The concept of a black box has already been described in 4.2.1. In this instance, it refers to whether or not you analyze the code itself (e.g. check all paths that are possible in an algorithm) or you treat the tested component as a black box and check if the desired functionality is achieved based on different inputs. Classical testing methods in the ”black box” category are: creating equivalence classes, setting up a cause and effect graph or error guessing. Examples for white box tests are: model based test case generation, data flow analysis and control flow graph analysis.
Exemplary, model based test case generation will be taken a look at (Infos taken from [7]). Starting from a comparatively rough model in addition to a test case specification, that contains detailed information on how the test cases will look like, possible test cases are determined and instantiated. These test cases are then turned into test vectors, which are used as input data for the implementation that is supposed to be tested. Comparing expected results with those of the implementation give information about the correctness of the system. In model based testing, the test case specification might demand, that every state, every transition, every condition and every class of paths is checked at least once. However if this deems to be unpractical due to a extensive amount of states, transitions, etc. you might benefit from so called ”Modified Condition Decision Coverage” (MCDC). The guideline ”Software Considerations in Airborne Systems and Equipment Certification” (RTCA/DO-178B) defines statement coverage in the following way (Citation taken from [7]):

> ”Every condition in a decision in the program has taken all possible outcomes at least once, every decision in the program has taken all possible outcomes at least once, and each condition in a decision has been shown to independently affect that decision’s outcome. A condition is shown to independently affect a decision’s outcome by varying just that condition while holding fixed all other possible conditions.”
>
> —[7]

This way, it is possible to test the influence of every single condition in the system in a computational effective manner.

Another possibility to completely ensure that the required functionality is the so called ”formal verification”, which is recommended for safety relevant systems (Infos from [7]).

Finally formal, static code analysis helps to reduce the complexity of the code as it often includes coding style guidelines from companies. These can be checked automatically by software tools and ensure a uniform code style as well as readability. Compulsory code reviews make sure that a high code quality is kept and that every developer is familiar with the code written by colleagues.

4.3.3. Tools for Software Testing

As in nearly all parts of production, many parts of the development process can and have been automated or optimized in a way that allows an easier and more effective work-flow. As mentioned above, testing is a particularly important area especially for safety relevant software as in autonomous driving systems. Thus, the testing procedure of this application domain is going to be taken a look at, as described by [7].

The testing process itself can be divided into seven steps. **Planning** the test, creating a testing **specification** and test **implementation** are the first three tasks that are required before running the test. The following steps are the **execution** of the test, **documenting** the results and **concluding** the test. Finally the functional safety undergoes a **verification**.

Due to the commonly used waterfall model, which comes with a successive implementation of Tier 1 deliverables, there are various levels of tests to choose from, depending on factors like cost in any form (time, money, effort), availability of required components, etc.. Some exemplary levels are:

Unit Testing

Unit testing is the most fundamental form of software testing, as it requires very little effort. Aside from the test itself, which needs to be set up with the seven steps as described above, it requires very little additional efforts, since it only tests the functionality of one component. It thus can help to verify, that a tested unit fulfills the desired internal behavior, however it does not necessarily assure its compatibility with other components, once it is integrated into a bigger context. Inputs from other components may be mocked (simulated) in order to run the test, even when surrounding components are not implemented yet.
Besides unit tests (on a virtual platform) themselves, integration testing and overall ECU test-

ing (on real ECU hardware) can also be seen as unit testing, as they all test the functionality of one unit.

Integration Testing

Whereas Unit testing checks each component on its own, integration testing has the goal to make sure, that components communicate correctly with each other. So it tests whether interfaces are implemented correctly. This test may also only test a coherent subsystem, i.e. multiple components, that work together to implement a specific functionality. This might still happen on a virtual platform, which is simpler to integrate into an automatic continuous integration procedure, however it can introduce another source of errors, when simulating hardware components like a communication bus.

Lab-Car Testing

A Lab-Car is a static, non drivable collection of real hardware components that are connected as they will be in the final vehicle. This is a compromise between a real testing environment and practicality, since it is not necessary to conduct on the street test runs. It often is the first integration test of the whole system and makes it easy to test compatibility of components. Also, it gives a first insight into production procedures and cable harnesses.
These Lab-Cars can often be run in a virtually dynamic mode, which allows for a range of input parameters (throttle, brake actuation, chosen gear, etc.) and offers easy ways to read output variables, like temperature signals, etc..

Prototype Vehicle

A prototype vehicle test the whole system in a real world environment. Obviously this level of testing requires concrete implementations of components and is rather costly regarding time and effort. On these test runs, predefined checklists are processed. Often multiple prototype vehicles are in use to accelerate the testing process, but also to gather statistical information as well as dispersion. Obviously errors are monitored and logged as they are directly communicated with developers, often on a daily basis.

5. Legal situations in various countries

Despite the research of past years in the field of autonomous driving, many countries have not yet adapted laws to accommodate for self driving cars. Different legislative regulations in countries all over the world and slow adaption of this technology often makes it impossible for car manufacturers to (officially) sell their cars as ”autonomous”. Some manufacturers choose to sell their systems as a ”beta” feature without any promises about the performance, like Tesla, which, despite the fact that they named their autonomous system ”Autopilot” does not allow the driver to be distracted from the traffic situation and can not be made liable for accidents caused under an active autopilot. But also the legal situation for car manufacturers is important as in regulations for testing their systems on the open roads. The following should be a comparison of the current (as of December 2018) legal regulations regarding these aspects between selected countries.

5.1. Means of Measure: The Autonomous Vehicle Readiness Index (AVRI) by KPMG

In order to compare the readiness of multiple countries for autonomous vehicles, KPMG has developed a methodology that takes a set of factors into account to produce a total score that makes it easy to rank and compare countries. The total score is based on four main ”pillars”: **policy and legislation**, **technology and innovation**, **infrastructure** and **consumer acceptance**. Each of these categories feature a set of judgment criteria, which results in one rank for each of the pillars. The total score is simply the sum of all pillar scores. While it takes more factors into consideration than just the legal situation in the respective country, as is depicted in this chapter, it still gives an interesting insight into the status quo around the world. In this chapter however, only the legislative pillar will be considered. The whole index for further information can be found in [56]. The ranking based on policy and legislation can be seen in figure 25.

5.2. Netherlands

Having the best total score in the AVRI and the third best policy and legislation score, the Netherlands represent a model for many other countries. Furthermore, they occupy place four in technology and innovation, place one in infrastructure and come in second in consumer acceptance (For AVRI ratings, see [56]).
The dutch government has recognized the potential in autonomous vehicles and has thus loosened its regulations regarding testing of autonomous vehicles on public roads. This offers manufacturers the ability to apply for a permit which allows them to test their autonomous

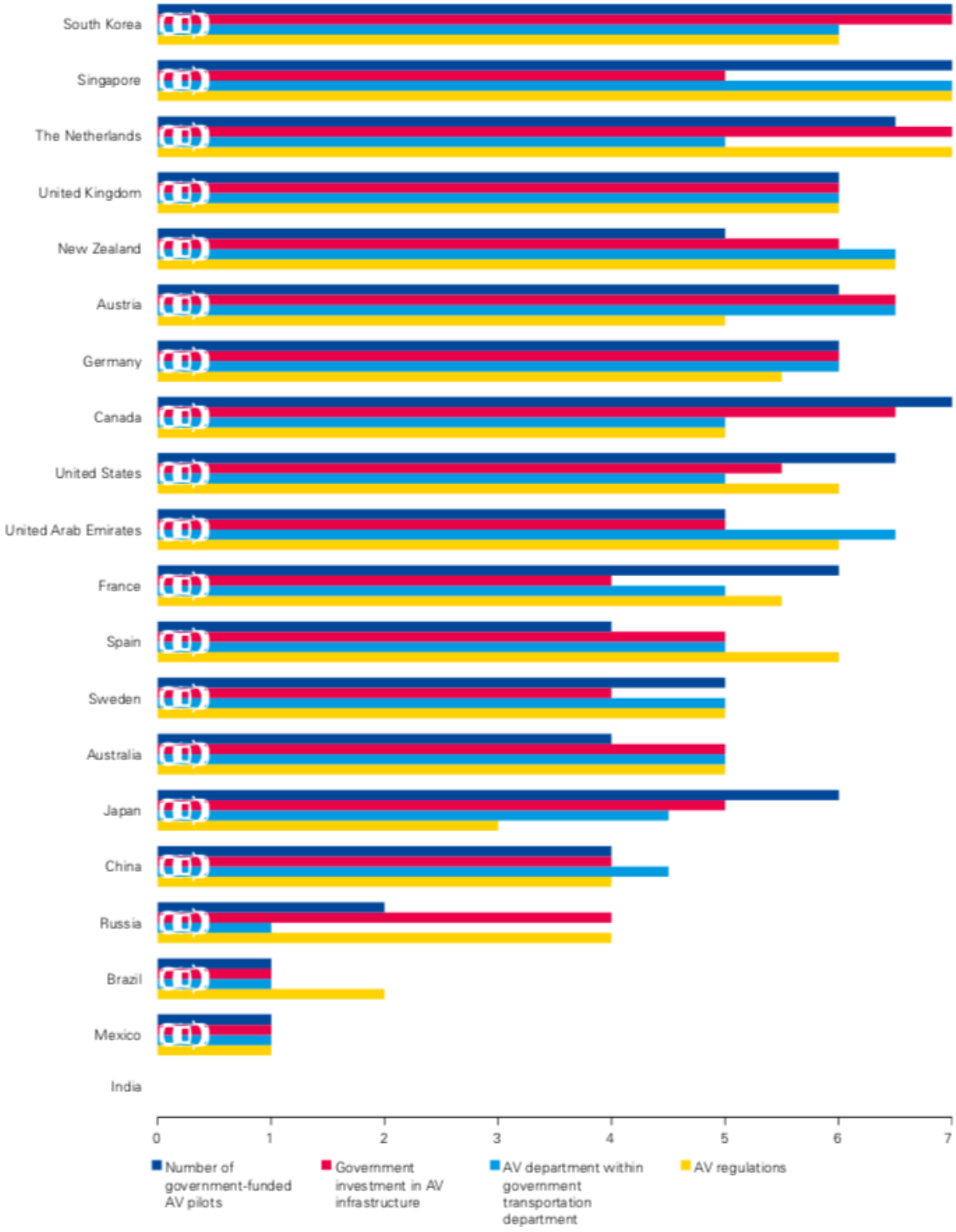

Figure 25 The rank of the AVRI based on policy and legislation. Graphic taken from [56].

vehicles on public roads, even without a driver in the car. However, regulations demand a driver for emergency takeovers who can control the car remotely. Having proven to the Dutch vehicle authority, that their tests are indeed safe, companies may drive their test vehicles on public roads (Information regarding public road tests in the Netherlands taken from [57])

5.3. Germany

Since Germany is home to many car manufacturers and not a minor force in the autonomous vehicle industry, its laws accommodate for self driving vehicles. In 2017, laws were updated in order to account for autonomous vehicles. Nevertheless, driver less cars are still not allowed. Regulations limit the application of autonomous cars to the following points (see: [58]):

- Automated driving systems may only be used as intended. A system that is developed to be used on highways only, the manufacturer must make sure, it can not be used elsewhere.
- When an automated driving system has taken over control of the vehicle, the driver may shift his attention to other matters, like operating his phone or watching a movie, however he must stay ready to take over control when asked to do so by the system and when the driver sees need to intervene in order to avoid an accident.
- The driver is still liable for everything the vehicle does when in an automated driving mode. This means that the manufacturer can not be made accountable for the damage that the car caused when driving automatically.
- A ”black box”, similar to those in airplanes, needs to record at all times whether the driver or the autonomous system was controlling the vehicle.

Despite these changes another regulation, ECE R 79, forbids automatic intervening into the steering of the vehicle, when its speed exceeds 10 km/h (plus 2 km/h tolerance). This hurdle is obviously too strict for a real world use of autonomous vehicles (Information regarding ECE R 79, see: [59]).

5.4. United States

Information in the following sector is taken from [60]. The legal regulations regarding autonomous vehicles in the US differs from state to state. While some states have adapted their law to allow companies to test their cars, others have denied them. Previously, the legislation of the automotive domain in the US has been organized in that the federal government was responsible for laws regarding the vehicle, whereas the states had power over the regulations of the drivers. However, an autonomous car is in some way both. Nevertheless, the US House of Representatives has restricted states to create unreasonable constraints against autonomous cars by their ”Self drive Act”.

California regulations concerning autonomous vehicles were taken from [61] and [62]. To take one state in the US as an example, which has rather liberal regulations, see California. With many tech companies in the region and the ”Silicon Valley”, it offers a great opportunity for manufacturers to test their vehicles in the real world. As of April, companies can apply for

three different types of permissions: test with a safety driver on board, test without a safety driver on board and deployment. Furthermore, in October 2017, California allowed vehicles without steering wheel or pedals onto public streets. There are however certain aspects that need to be fulfilled by the (as of 29th November 2018) 61 permit holders.
Firstly, they need to file so called ”Autonomous Vehicle Collision Reports”, within 10 days after an accident. Secondly, so called ”Autonomous Vehicle Disengagement Reports” must be submitted every year to the DMV that contain a summary of disengagements of the autonomous driving system during testing.

6. Ethical Aspects of Autonomous Driving

Finally, the ethics involved in the development of autonomous vehicles are far from trivial. It sure represents an impressive technological advancement to develop a self driving car, however the extent to which technology will be able to decide on safety relevant matters, which might result in physical harm of humans, or animals for that matter, should carefully be assessed. Due to the topicality of the subject, some of these ethical dilemmas have already been addressed by ethics committees.

One specific example that is often used as a dilemma of autonomous vehicles, is how they should act when an accident is unavoidable. Several institutions have studied this scenario trying to find guidelines that are ethical. One of these institutions is the German Federal Ministry of Transport, who has published a report by an ethics commission, titled ”Automated and Connected driving” (See [63]). It contains 20 rules that explain what autonomous and connected vehicles may do and how they should decide in emergency situations. A subset of these rules will be summarized in the following:

Rule 2: Enhancing safety in traffic by protecting every participant, is the number one priority and must not be overridden by any other ”utilitarian consideration”(See [63] p.6, point 2). Autonomous driving systems should not be licensed if it does not diminish potential harm.

Rule 5: The automated driving system needs to be built in a way that it does not maneuver itself into a dangerous situation and if it encounters one, avoid accidents whenever possible. This includes the aforementioned dilemma of choosing between two evils.

Rule 7: In case of an unavoidable accident, while all technical mechanisms to avoid an incident have taken place, the life of humans has the highest priority. Thus, the system must, as technology possible, be programmed in a way to save the life of humans over damage to animals or objects.

Rule 9: When encountering an unavoidable accident that leaves a potential option for the system to choose on what traffic participant will be headed towards, it must not take personal features, such as ”age, gender, physical or mental constitution” (See [63] p.7, point 9) into account. It is also forbidden to initiate a weighting of victims against each other. However a mechanism that reduces the amount of persons injured can be seen as ethical.

Rule 14: Attacks against the automated driving system, especially its electronic infrastructure must not possess such severity that they influence the people’s confidence in this technology.

Rule 19: The autonomous vehicle must on its own switch to a safe state when it encounters hazardous situations.

As you can see, many thought has already gone into the ethical aspects of autonomous driving.

MIT researchers have published the results of a survey (See [64]), that asked 1928 participants how self driving cars should act in various scenarios, which results will be assessed later in this chapter. Within [64], the authors capture another important aspect, namely that:

> Manufacturers and regulators will need to accomplish three potentially incompatible objectives: being consistent, not causing public outrage, and not discouraging buyers.
>
> —[64]

It captures an intuitively obvious conflict: car manufacturers naturally want to make their vehicle as safe for the passengers as possible. This car would subsequently have a distinct advantage on the market as safety is one of the key concerns of customers. Therefor companies might choose an algorithm that rather spares the passengers from any harm, even if more pedestrians or other traffic participants will be injured in an accident. In order to avoid this scenario, legislative means would be needed to prohibit such an implementation. The survey stresses, that these events might occur only rarely, however will increase, once autonomous vehicles become more common on the roads.

The survey [64] showed, that most people think an autonomous car should sacrifice its passengers, in order to save a group of pedestrians. 76% of participants stated it would be moral to kill one passenger in the vehicle to save ten pedestrians. When asked to rate two approaches from 0 (save the passengers) to 100 (minimize number of victims), many participants tended towards the second option, with an average of 85. Another question showed, that the more pedestrians can be saved, by sacrificing the driver, the more people advocated for the car killing the driver. However, to come back to the advantage of manufacturers selling passenger-protective cars: participants preferred vehicles on the street, that sacrificed the driver, however they did not want to buy one themselves, instead for them they would choose a vehicle that protects the driver. Finally [64] comes to the conclusion, that regulations might be ”necessary but also counterproductive”.

All in all, the time until autonomous vehicles are fully ready for market can and should be used to raise awareness for the dilemmas imposed by this technology. Thereby legislation will be forced to adapt laws and regulations to avoid a split into two classes of autonomous cars, where the ”passenger saving feature” in form of a different ethical algorithm is sold as an expensive add on to maximize profit.

Although people tend to agree that everyone would be better off if AVs were utilitarian (in the sense of minimizing the number of casualties on the road), these same people have a personal incentive to ride in AVs that will protect them at all costs. Accordingly, if both self-protective and utilitarian AVs were allowed on the market, few people would be willing to ride in utilitarian AVs, even though they would prefer others to do so.

—[64]

Acronyms

A

ABS - Anti Blocking System

AC - Air Conditioning

API - Application Programming Interface

AV - Autonomous Vehicle

C

CMOS - Complementary Metal-Oxide Semiconductor

CPU - Central Processing Unit

CWFM - Continuous Wave Frequency Modulation

D

DMV - Department of Motor Vehicles

E

EBS - Emergency Braking System

ECU - Electronic Control Unit

Ego vehicle - Vehicle which includes the autonomous system

ESP - Electronic Stability Program

F

FAV - Functional Architecture View

FPS - Frames per Second

G

GPS - Global Positioning System

GPU - Graphics Processing Unit

K

KPMG - Financial Audit Company; named after their founders: Piet Klynveld, William Barclay Peat, James Marwick and Reinhard Goerdeler

N

NHTSA - National Highway Traffic Safety Administration (US)

O

OEM - Original Equipment Manufacturer

OS - Operating System

S

SAE - Society of Automotive Engineers

SNR - Signal to Noise Ratio

SOC - System on a Chip

T

TACC - Traffic Aware Cruise Control (Tesla)

TRL - Technological Readiness Level

U

UV - Ultraviolet

Bibliography

[1] S. Behere and M. Törngren, “A Functional Architecture for Autonomous Driving,” in *Proc. First Int. Work. Automot. Softw. Archit. - WASA ’15*, 2015. [Online]. Available: https://sagar.se/files/wasa2015.pdf

[2] S. international, “U.S. Department of Transportation’s New Policy on Automated Vehicles Adopts SAE International’s Levels of Automation for Defining Driving Automation in On-Road Motor Vehicles,” p. 2, 2016. [Online]. Available: https://web.archive.org/web/20170903105244/https://www.sae.org/misc/pdfs/automated_driving.pdf

[3] D. Muoio, “RANKED: The 18 companies most likely to get self-driving cars on the road first,” 2017. [Online]. Available: https://www.businessinsider.de/the-companies-most-likely-to-get-driverless-cars-on-the-road-first-2017-4?op=1

[4] CBInsight, “46 Corporations Working On Autonomous Vehicles,” 2018. [Online]. Available: https://www.cbinsights.com/research/autonomous-driverless-vehicles-corporations-list/

[5] D. Silverberg, “MIT autonomous bicycle,” 2018. [Online]. Available: https://motherboard.vice.com/en_us/article/qvwxev/mit

[6] Bundesanstalt für Straßenwesen, “Rechtsfolgen zunehmender Fahrzeugautomatisierung,” Bundesanstalt für Straßenwesen, Tech. Rep., 2012. [Online]. Available: http://www.bast.de/DE/Publikationen/Foko/Downloads/2012-11.pdf?__blob=publicationFile

[7] R. Stolle, R. Asmus, T. Stauner, and C. Dr. Salzmann, “TUM Lecture - Automotive Software Engineering,” Munich, 2018.

[8] R. Gert and U. Voelzke, “Three Sensor Types Drive Autonomous Vehicles,” 2017. [Online]. Available: https://www.sensorsmag.com/components/three-sensor-types-drive-autonomous-vehicles

[9] G. Alper, “Dynamic range (DNR) and signal to noise ratio for CCD and CMOS image sensors,” 2013. [Online]. Available: https://www.adimec.com/dynamic-range-dnr-and-signal-to-noise-ratio-snr-for-ccd-and-cmos-image-sensors/

[10] “Signal-to-noise ratio (imaging),” 2018. [Online]. Available: https://en.wikipedia.org/wiki/Signal-to-noise_ratio_(imaging)

[11] B. Newton, “Premium-Kamerasystem mit drei Objektiven für teilautomatisiertes Fahren,” 2016. [Online]. Available: https://www.all-electronics.de/premium-kamerasystem-mit-drei-objektiven-fuer-teilautomatisiertes-fahren/

[12] W. T. Buller, “Benchmarking Sensors for Vehicle Computer Vision Systems,” 2017. [Online]. Available: http://mtri.org/automotivebenchmark.html

[13] C. Wolff and K. Hoel, “Radar Basics Radar Principle.” [Online]. Available: http://www.radartutorial.eu/01.basics/RadarPrinciple.en.html#this

[14] “Radar.” [Online]. Available: https://de.wikipedia.org/wiki/Radar

[15] Digi-Key’s European Editors, “Radar Sensing for Driverless Vehicles,” 2016. [Online]. Available: https://www.digikey.com/en/articles/techzone/2016/nov/radar-sensing-for-driverless-vehicles

[16] R. Schneider and J. Wenger, “High resolution radar for automobile applications,” *Adv. Radio Sci.*, 2003. [Online]. Available: https://www.researchgate.net/publication/26438456_High_resolution_radar_for_automobile_applications

[17] A. Neal, “LiDAR vs. RADAR,” 2018. [Online]. Available: https://www.sensorsmag.com/components/lidar-vs-radar

[18] D. Kohanbash, “LIDAR vs RADAR: A Detailed Comparison,” 2017. [Online]. Available: http://robotsforroboticists.com/lidar-vs-radar/

[19] T. Agarwal, “LIDAR System (Light Detection And Ranging) Working and Applications.” [Online]. Available: https://www.elprocus.com/lidar-light-detection-and-ranging-working-application/

[20] A. Carullo and M. Parvis, “An ultrasonic sensor for distance measurement in automotive applications,” *IEEE Sens. J.*, 2001.

[21] Bosch, “Bosch Mobility Solution most sensitive ultrasonic system on the market for highly precise parking and maneuvering Used in these systems,” 2018. [Online]. Available: https://www.bosch-mobility-solutions.com/en/products-and-services/passenger-cars-and-light-commercial-vehicles/driver-assistance-systems/construction-zone-assist/ultrasonic-sensor/

[22] H. Munenori, “An introduction to ultrasonic sensors for vehicle parking,” 2010. [Online]. Available: http://www.newelectronics.co.uk/electronics-technology/an-introduction-to-ultrasonic-sensors-for-vehicle-parking/24966/

[23] Waymo, “On the Road to Fully Self-Driving.” [Online]. Available: https://storage.googleapis.com/sdc-prod/v1/safety-report/SafetyReport2018.pdf

[24] V. Koifman, “Waymo Self-Driving Car Relies on 5 LiDARs and 1 Surround-View Camera,” 2017. [Online]. Available: http://image-sensors-world.blogspot.com/2017/10/waymo-self-driving-car-relies-on-5.html

[25] Tesla, “Tesla Autopilot,” 2018. [Online]. Available: https://www.tesla.com/autopilot?redirect=no

[26] A. J. Hawkins, “Elon Musk still doesn’t think LIDAR is necessary for fully driverless cars,” 2018. [Online]. Available: https://www.theverge.com/2018/2/7/16988628/elon-musk-lidar-self-driving-car-tesla

[27] C. Thompson, “Tesla Model S with new sensors Business Insider,” 2016. [Online]. Available: http://uk.businessinsider.com/tesla-model-s-with-new-sensors-photos-2016-3?IR=T

[28] Audi, “Der neue Audi A8-hochautomatisiertes Fahren auf Level 3,” 2017. [Online]. Available: https://www.audi-mediacenter.com/de/per-autopilot-richtung-zukunft-die-audi-vision-vom-autonomen-fahren-9305/der-neue-audi-a8-hochautomatisiertes-fahren-auf-level-3-9307

[29] C. Paukert, “Why the 2019 Audi A8 won’t get Level 3 partial automation in the US,” 2018. [Online]. Available: https://www.cnet.com/roadshow/news/2019-audi-a8-level-3-traffic-jam-pilot-self-driving-automation-not-for-us/

[30] P. McNamara, “How did Audi make the first car with Level 3 autonomy,” 2017. [Online]. Available: https://www.carmagazine.co.uk/car-news/tech/audi-a3-level-3-autonomy-how-did-they-get-it-to-market/

[31] J. Gilboy, “Utah Tesla Driver Turns on Autopilot, Rams Fire Truck at 60 MPH-The Drive,” Tech. Rep., 2018. [Online]. Available: http://www.thedrive.com/news/20848/utah-tesla-driver-turns-on-autopilot-rams-fire-truck-at-60-mph

[32] ——, “Officials Find Cause of Tesla Autopilot Crash Into Fire Truck,” Tech. Rep., 2018. [Online]. Available: http://www.thedrive.com/news/20912/cause-of-tesla-autopilot-crash-into-fire-truck-cause-determined-report

[33] T. Lee, “Why emergency braking systems sometimes hit parked cars and lane dividers,” *Ars Tech.*, 2018. [Online]. Available: https://arstechnica.com/cars/2018/06/why-emergency-braking-systems-sometimes-hit-parked-cars-and-lane-dividers/

[34] T. Univ. Prof. Dr. Huckle and T. Dr. Neckel, *Bits and Bugs: A Scientific and Historical Review on Software Failures in Computational Science*. SIAM series on Software, Environments, and Tools, 2019.

[35] Tesla, *Model S owner's manual*, 2018. [Online]. Available: https://www.tesla.com/sites/default/files/model_s_owners_manual_north_america_en_us.pdf

[36] W. Elmenreich, "An Introduction to Sensor Fusion," Institut für Technische Informatik Vienna University of Technology, Austria, Tech. Rep., 2002. [Online]. Available: https://www.researchgate.net/profile/Wilfried_Elmenreich/publication/267771481_An_Introduction_to_Sensor_Fusion/links/55d2e45908ae0a3417222dd9/An-Introduction-to-Sensor-Fusion.pdf

[37] Wikipedia, "Sensordatenfusion," 2018. [Online]. Available: https://de.wikipedia.org/wiki/Sensordatenfusion

[38] NTSB, "Collision Between a Car Operating With Automated Vehicle Control Systems and a Tractor-Semitrailer Truck," Tech. Rep., 2017. [Online]. Available: https://www.ntsb.gov/investigations/AccidentReports/Reports/HAR1702.pdf

[39] Tesla, "MODEL S SOFTWARE RELEASE NOTES v7.1," Tech. Rep., 2016. [Online]. Available: https://www.tesla.com/sites/default/files/pdfs/release_notes/tesla_model_s_software_7_1.pdf

[40] R. Kruger, G. Simeonov, F. Beck, and T. Ertl, "Visual Interactive Map Matching," University of Duisburg-Essen, Duisburg-Essen, Tech. Rep. 6, 2018. [Online]. Available: https://www.vis.wiwi.uni-due.de/uploads/tx_itochairt3/publications/pacificvis18-tvcg_map_matching.pdf

[41] L. Wang, Y. Zhang, and J. Wang, "Map-Based Localization Method for Autonomous Vehicles Using 3D-LIDAR," Department of Control Science and Engineering, Tongji University, Shanghai, Tech. Rep., 2017. [Online]. Available: https://www.researchgate.net/profile/Yihuan_Zhang/publication/319881329_Map-Based_Localization_Method_for_Autonomous_Vehicles_Using_3D-LIDAR/links/59c02555458515e9cfd54de8/Map-Based-Localization-Method-for-Autonomous-Vehicles-Using-3D-LIDAR.pdf

[42] M. Schreiber, F. Poggenhans, and C. Stiller, "Detecting symbols on road surface for mapping and localization using OCR," in *2014 17th IEEE Int. Conf. Intell. Transp. Syst. ITSC 2014*, 2014. [Online]. Available: http://www.mrt.kit.edu/z/publ/download/2014/SchreiberPoggenhansStiller2014itsc.pdf

[43] J. Peter, “Digitales Testfeld Autobahn,” 2018. [Online]. Available: https://adac-blog.de/digitales-testfeld-autobahn/

[44] Ç. Kaymak and A. Uçar, “A Brief Survey and an Application of Semantic Image Segmentation for Autonomous Driving,” Firat University, Mechatronics Eng. Dept., Elazig, Turkey, Tech. Rep. [Online]. Available: https://arxiv.org/pdf/1808.08413.pdf

[45] Google, “Google Self-Driving Car Project Monthly Report,” Google, Tech. Rep., 2016. [Online]. Available: https://static.googleusercontent.com/media/www.google.com/de//selfdrivingcar/files/reports/report-0216.pdf

[46] Mcc, “Local Dynamic Map.” [Online]. Available: http://www.mobile-car-communication.de/index.php?ptitle=ldm

[47] NTSB, “Preliminary Report - Highway HWY18MH010,” 2018. [Online]. Available: https://www.ntsb.gov/investigations/AccidentReports/Pages/HWY18MH010-prelim.aspx

[48] M. McNaughton, C. Urmson, J. M. Dolan, and J.-w. Lee, “Motion planning for autonomous driving with a conformal spatiotemporal lattice,” in *2011 IEEE Int. Conf. Robot. Autom.*, vol. 1. IEEE, may 2011, pp. 4889–4895. [Online]. Available: http://ieeexplore.ieee.org/document/5980223/

[49] M. Wang, T. Ganjineh, and R. Rojas, “Action annotated trajectory generation for autonomous maneuvers on structured road networks,” *ICARA 2011 - Proc. 5th Int. Conf. Autom. Robot. Appl.*, pp. 67–72, 2011.

[50] J. Laukkonen, “What Is Drive-by-Wire Technology?” 2018. [Online]. Available: https://www.lifewire.com/what-is-drive-by-wire-534825

[51] A. Sawant, S. Dr. Lenina, and J. Dhananjay, “CAN , FlexRay , MOST versus Ethernet for Vehicular,” *Int. J. Innov. Adv. Comput. Sci.*, vol. 7, no. 4, pp. 336–339, 2018.

[52] WAYMO, “Technology-We’re building a safer driver that is always alert and never distracted.” [Online]. Available: https://waymo.com/tech/

[53] Huawei, “Huawei and Audi Announce Joint Innovation in L4 Automatic Driving Solutions by Industry,” 2018. [Online]. Available: https://www.huawei.com/en/press-events/news/2018/10/Huawei-Audi-L4-Automatic-Driving

[54] Springer, “Qualitaetsmanagement – Begriffe, Grundprinzipien, Anwendung im Gesundheitswesen,” in *PsyQM. Springer*. Berlin, Heidelberg: Springer, 2008, ch. A.1.

[55] B. Prof. Dr. Bruegge, “TUM Lecture - Introdction to Software Engineering,” 2016.

[56] KPMG, “Autonomous Vehicles Readiness Index: Assessing countries’ openness and preparedness for autonomous vehicles,” p. 60, 2018. [Online]. Available: https://assets.kpmg.com/content/dam/kpmg/tw/pdf/2018/03/KPMG-Autonomous-Vehicle-Readiness-Index.pdf

[57] Government of the Netherlands, “Mobility , public transport and road safety Self-driving vehicles Vehicles are increasingly equipped with automatic features , such as automatic parking , adaptive cruise control and stop and go control systems . These automatic features are rapidly d.” [Online]. Available: https://www.government.nl/topics/mobility-public-transport-and-road-safety/self-driving-vehicles

[58] M. Wacket, T. Escritt, and T. Davis, “Germany adopts self-driving vehicles law,” 2017. [Online]. Available: https://www.reuters.com/article/us-germany-autos-self-driving/germany-adopts-self-driving-vehicles-law-idUSKBN1881HY

[59] M. Schubert, “Revision of the Road Traf c Act (Almost) Paves the Way for Automated Driving in Germany,” 2017. [Online]. Available: http://de.genre.com/knowledge/blog/revision-of-the-road-traffic-act-almost-paves-the-way-for-automated-driving-in-germany-en.html

[60] B. Smith, “Regulations for Autonomous Cars,” 2018. [Online]. Available: https://www.azom.com/article.aspx?ArticleID=15668

[61] A. J. Hawkins, “California green lights fully driverless cars for testing on public roads,” *The Verge*, 2018. [Online]. Available: https://www.theverge.com/2018/2/26/17054000/self-driving-car-california-dmv-regulations

[62] DMV California, “Testing of Autonomous Vehicles with a Driver,” 2018. [Online]. Available: https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/testing

[63] Federal Ministry of Transport and Digital Infrastructure, “Ethics Commission automated and connected driving,” Tech. Rep., 2017. [Online]. Available: https://www.bmvi.de/SharedDocs/EN/publications/report-ethics-commission-automated-and-connected-driving.pdf?__blob=publicationFile

[64] J.-F. Bonnefon, A. Shariff, and I. Rahwan, “The social dilemma of autonomous vehicles,” *Science (80-.).*, vol. 352, no. 6293, pp. 1573–1576, jun 2016. [Online]. Available: http://www.sciencemag.org/cgi/doi/10.1126/science.aaf2654