Höhere Technische Bundeslehranstalt
Klagenfurt, Mössingerstraße

DIPLOMARBEIT

COMMEAN

eingereicht von: Luca Nachbar
Stefan Pisjak

Projektbetreuer: Dipl.–Ing. Benjamin Makula

Diese Diplomarbeit entspricht den Standards gemäß dem Leitfaden zur Umsetzung der Reife- und Diplomprüfung des BM|BF in der letztgültigen Fassung.

Klagenfurt, Mai 2022

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Unterschrift:

Luca Nachbar

Klagenfurt, 11. Mai 2022

Unterschrift:

Stefan Pisjak

Klagenfurt, 11. Mai 2022

Kurzfassung

Der Verkehrsfluss ist einer der wichtigsten, wenn nicht sogar der wichtigste Faktor in der modernen Stadtplanung. Üblicherweise wird er entweder manuell durch temporäre Messstationen oder durch sehr einfache Zählmechanismen analysiert. Genau hier soll diese Diplomarbeit ansetzen und diesen Prozess erleichtern.

Dies soll durch kleine Messstationen erfolgen, sogenannte Nodes. Diese werden an wichtigen Stellen in der Stadt platziert. Alle Nodes sind mit einer Kamera ausgestattet und werden überhöht befestigt, damit ein guter Überblick über die aktuelle Verkehrssituation gewährleistet ist. Mithilfe von künstlicher Intelligenz werden dann die einzelnen Fahrzeuge erkannt und durch die Unterstützung von maschinellem Lernen können die Fahrzeuge verfolgt werden.

Diese Fahrzeuge werden anschließend in Kategorien klassifiziert, wie z. B. Auto, Lkw oder Bus. Auch die Zeit, in der sich die Fahrzeuge im Sichtbereich der Kamera befinden, wird gespeichert. Datenschutz wird dadurch garantiert, dass alles lokal auf der Node verarbeitet wird und dadurch das Video niemals das Internet durchqueren muss.

Die gesammelten Daten werden dann von den Nodes über LTE/5G bzw. LoRa zu einem dedizierten Server gesendet. Dort werden die Daten dann analysiert und Statistiken, wie der Verkehrsfluss, werden generiert. Schließlich werden diese Daten auf einer Website den Bürger*Innen angezeigt.

Abstract

Traffic flow is the key information needed for city traffic planning. Traditionally, this data has to be gathered manually by humans. This thesis aims to automate this process.

This is done by mounting measuring stations, so-called Nodes, on key roads around the city. Those nodes are equipped with a camera and are placed on an elevated level to get a better overview of the traffic. With the help of artificial intelligence the individual vehicles are detected and afterwards tracked with machine learning.

Those vehicles are also classified into categories such-as cars, trucks or buses and the time, which is spent in the viewing angle of the camera is saved. Privacy is guaranteed by processing the video locally, so that the video stream never enters the internet.

Next the traffic data is sent by the nodes over LTE/5G or LoRa to a dedicated server. There the captured data is being analysed and statistics, such as the traffic flow, are generated. Finally, those statistics are displayed on a website available to the public.

Preface

Wir sind zwei Schüler*innen der HTL-Mössingerstraße und das ist unsere Diplomarbeit. Nach fünf Jahren Ausbildung wird unser Lehrgang in der HTL mit einer Diplomarbeit abgeschlossen. Hierfür haben wir uns ein Thema ausgesucht, bei welchem jede Person seine eigenen Stärken einbringen und sich in einem interessanten Themengebiet weiterbilden kann.

So ist in dieser Diplomarbeit ein großer Querschnitt vieler technischer Themen vorzufinden. Von hardwarenahen, telekommunikationstechnischen Themen, wie LoRa oder LTE und 5G, über Enterprise Frameworks, wie Spring, bis zur künstlichen Intelligenz mit maschinellem Lernen ist alles dabei.

Auch unserem Projektbetreuer Dipl.–Ing. Benjamin Makula gilt großer Dank, da er uns durch mögliche Lösungsansätze und viel Fachwissen weiterhelfen konnte.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

KAPITEL

1 Einleitung

Der Personen- und Lastverkehr auf Basis von fossilen Brennstoffen bewirkt eine immense lokale CO_2-Emission. Aber auch Fahrzeuge mit elektrischem Antrieb tragen bei aktuellem Strommix zu den Treibhausgasen und dadurch zur Klimakrise bei. Um den Stadtverkehr in Hinblick auf Abgase optimaler gestalten zu können, müssen unweigerlich Verkehrsmessungen durchgeführt werden. Dadurch können Hauptverkehrswege erschlossen werden und im weiteren Sinn können auf genau diesen, die öffentlichen Verkehrsmittel ausgebaut werden. Eine kostengünstige und mobile Messstation an Ampeln oder an anderen markanten Punkten in der Stadt würde eine enorme Erleichterung für eine zukünftige Stadtverkehrsplanung mit sich bringen. Genau an dieser Stelle soll die Diplomarbeit ansetzen.

1.1 Zielsetzung

Das Ziel dieser Diplomarbeit ist es, die in der Einleitung (1) beschriebene Ausgangslage zu verbessern. Konkret bedeutet das, dass sich in dem Gebiet, in welchem die Verkehrsanalyse durchgeführt werden soll, verschiedene Messstationen befinden. Diese sind mit einer Kamera und einem SBC (Single Board Computer) ausgestattet und sollen so gut wie möglich gegen Wind, Wetter und Vandalismus geschützt sein. Mithilfe der Kamera soll der SBC den Videostream aufnehmen und dann lokal in diesem Fahrzeuge erkennen. Zusätzlich soll es möglich sein, zwischen verschiedenen Fahrzeugklassen zu unterscheiden, um somit den Verkehr noch besser analysieren zu können.

Die gesammelten Daten sollen nun bei einer erfolgreichen Netzwerkverbindung an einen zentralen Server übertragen werden. Hierfür soll LoRa verwendet werden, aber es soll auch möglich sein, das Übertragungsmedium einfach zu wechseln. Dadurch ist es möglich, an einem Standort, an welchem kein LoRa-Netzwerk verfügbar ist, auf eine LTE/5G Verbindung umzuschalten.

Die erworbenen Daten sollen anschließend auf einem Server verarbeitet werden. Hierfür müssen diese in einer Datenbank gespeichert werden, da es möglich sein soll, den Verlauf des Verkehres über einen gewissen Zeitraum zu verfolgen. Außerdem werden diese Daten dann grafisch aufbereitet und über eine Website angezeigt.

1.2 Kooperation

Diese Diplomarbeit entsteht in Kooperation mit der Klagenfurt Mobil GmbH, welche für den öffentlichen Nahverkehr in Klagenfurt am Wörthersee zuständig ist. Ziel hierbei ist es, dass diese Messstationen in der Stadt Klagenfurt aufgebaut werden können und den Personen, welche für die Verkehrsplanung zuständig sind, dabei helfen die Busverbindungen zu optimieren.

KAPITEL

2 Grundlagen und Methoden

2.1 Stand der Technik

Zur Erkennung von Fahrzeugen gibt es derzeit unterschiedliche Möglichkeiten. So verwenden beispielsweise manche Städte Induktionsschleifen, um den Verkehr zu analysieren. Diese müssen aufwendig in der Fahrbahndecke eingelassen oder unter der Fahrbahn verlegt werden. Hierdurch kann gemessen werden, wie lange sich ein Fahrzeug über dieser befindet. Durch die Veränderung der Umgebung bedingt durch das Fahrzeug, ändert sich die Induktivität. Zusätzlich kann auch gemessen werden, wie lange sich das Fahrzeug über der Schleife befindet. Dadurch, dass die Geschwindigkeit auch (zumindest in etwa) bekannt ist, lässt sich durch die Änderung der Induktivität und der Dauer der Induktivitätsänderung der Fahrzeugtyp errechnen.

Diese Art von Messung kann gut auf Strecken, auf welchen der Verkehr frei fließen kann, eingesetzt werden. Wird diese Messvariante jedoch bei einer Ampel bzw. einem anderen Ort, an dem Fahrzeuge stehen können, eingesetzt, dann ist es viel schwieriger bzw. fast unmöglich den Fahrzeugtyp ausfindig zu machen. Auch die Anzahl der stehenden Fahrzeuge ist aktuell mit diesem Verfahren unmöglich herauszufinden. Zusätzlich ist es schwer festzustellen, wie sich der Verkehrsfluss verhält, da als einzige Referenz für ein Fahrzeug die Änderung der Induktivität und die Länge dient.

Ein weiterer Nachteil dieser Messmethode ist, dass Induktionsschleifen möglicherweise kleinere Fahrzeuge, wie Motorräder, Mopeds oder Fahrräder nicht gut erkennen, da diese das Magnetfeld nicht stark ändern. Hierbei können weitere Sensoren ein wenig Abhilfe schaffen. So könnten z. B. Lichtschranken oder Ultraschallsensoren verwendet werden, um alle Fahrzeuge zuverlässiger zu erkennen. Jedoch weisen auch diese Tücken auf. Weiters ist es mit einer Lichtschranke bzw. einem Ultraschallsensor schwierig zu unterscheiden, ob es nun ein Lkw mit Anhänger war oder zwei dicht hintereinander fahrende Pkws. Dies ist vor allem bei sich langsam fortbewegendem Verkehr eine besondere Herausforderung. [Ver21]. Um diese Herausforderung zu bewältigen, kann eine Kamera verwendet werden, was die elementare Ausgangslage dieser Diplomarbeit ist.

2.2 Lösungsansatz

Der Verkehr soll mittels Kamera analysiert werden und dadurch die Nachteile der zuvor genannten Technologien umgehen. Jedoch müssen diese Daten nicht nur gesammelt, sondern auch übertragen und dann für die Analyse aufbereitet werden.

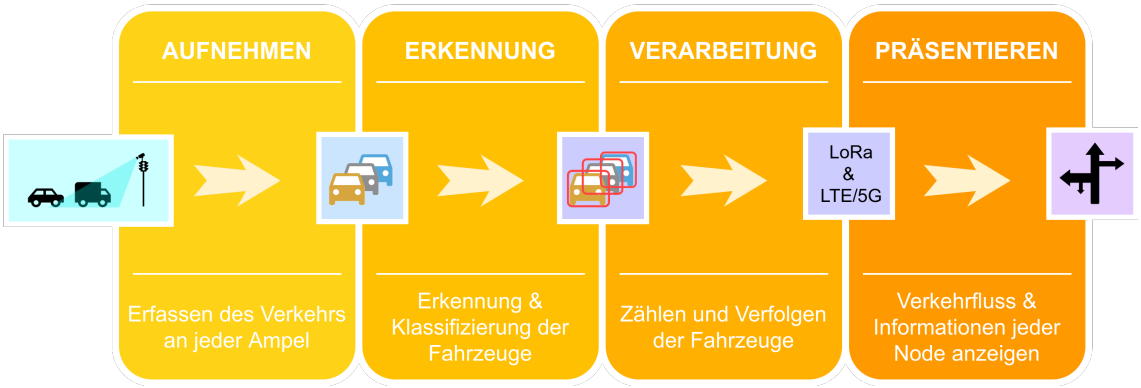


Abbildung 2.1: Übersicht über die Diplomarbeit

Diese Aufgabenbereiche werden nun auf die Teammitglieder aufgeteilt. So wird hardwaremäßig alles, was lokal auf jeder Messstationen passiert, von Stefan Pisjak bearbeitet, während für den Server (Backend und Frontend) Luca Nachbar zuständig ist

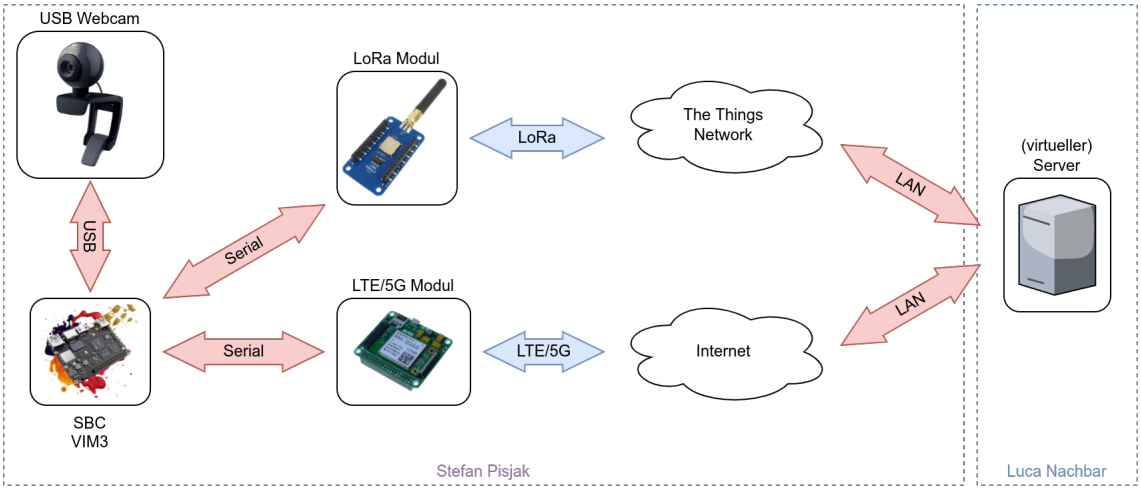


Abbildung 2.2: Aufteilung der Diplomarbeit – Hardware – Bilder von [ber21] [kha21d] [Har21] [Six21]

Für die Verarbeitung des Videostreams der Kamera ist Stefan Pisjak zuständig (Siehe Abschnitt 3.2). Das Erkennen, Klassifizieren und Zählen der Fahrzeuge soll lokal auf der Node erfolgen. Diese besteht aus einem SBC und einer Kamera. Die gesammelten Daten sollen dann über ein beliebiges Übertragungsmedium übertragen werden, weshalb verschiedene Datenaustauschmedien unterstützt werden sollen. In dieser Diplomarbeit wird der Fokus jedoch auf die Übertragung über LoRa bzw. LTE/5G gesetzt.

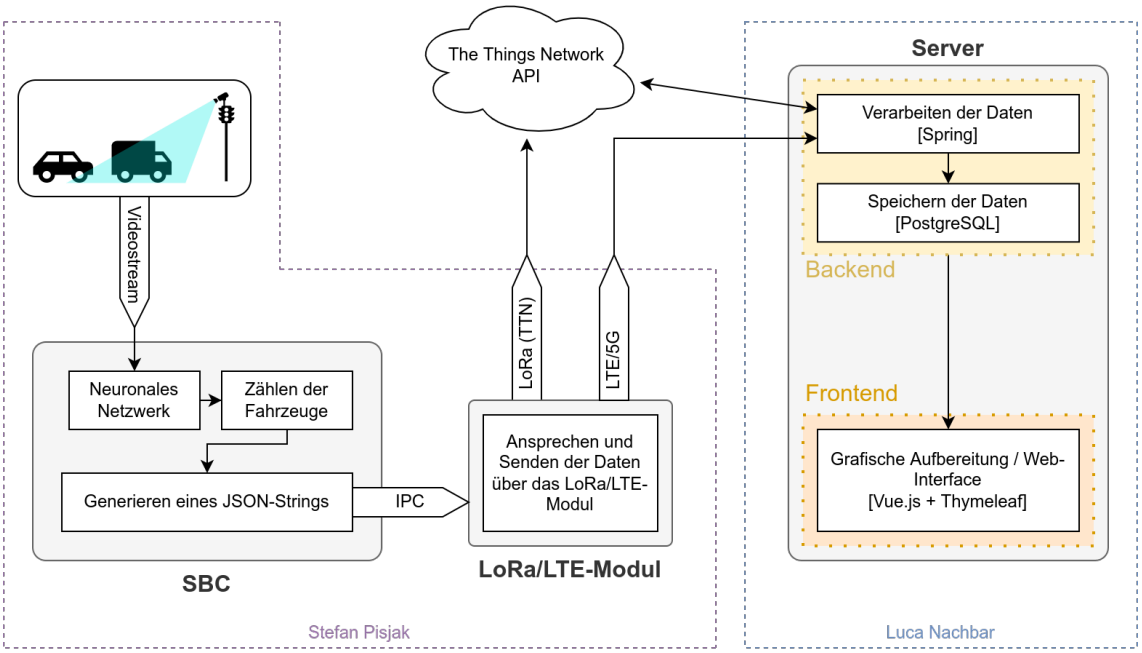


Abbildung 2.3: Aufteilung der Diplomarbeit - Software

Die gesammelten und übertragenen Daten werden anschließend vom Backend verarbeitet. In diesem müssen die jeweiligen Messdaten zu allen Ampeln gespeichert und verarbeitet werden. Damit die gesammelten Informationen auch einen Mehrwert bieten, sollen diese übersichtlich für alle Bürger*innen auf einer Website zugänglich gemacht werden. Hierfür wird als Framework *Vue.js* [You21] verwendet. Sowohl Back- und Frontend ist Aufgabe von Luca Nachbar und wird in Abschnitt 3.1 im Detail beschrieben.

KAPITEL

3 Individuelle Zielsetzung der Teammitglieder

3.1 Individuelle Zielsetzung des Teammitglieds Luca Nachbar

3.1.1 Aufgabenstellung

3.1.1.1 API für Datenaustausch

Da die Erkennung und Klassifizierung von Fahrzeugen bereits unmittelbar an der Ampel vorgenommen wird, müssen die Daten nur mehr übertragen werden. Um die Daten von der Ampel an das Server-Backend zu übertragen, wird eine einheitliche API definiert. Eine zusätzliche API ermöglicht es im Weiteren, Daten aus dem System auszulesen und den Nutzer*innen zur Verfügung zu stellen.

3.1.1.2 Verarbeiten der Daten

Nachdem die Daten der Fahrzeugerkennung & Klassifizierung an das Backend gesendet worden sind, müssen diese verarbeitet werden.
Dabei sind folgenden Werte ausschlaggebend:

- Anzahl der Fahrzeuge
- Aufenthalt an der Ampel
- Geschwindigkeit

Daraus soll folgendes berechnet werden:

- Durchschnittliche Wartezeit
- Rushhours im Tagesverkehr
- Durchschnittliche Geschwindigkeit

Die errechneten Werte werden anschließend in einer Datenbank gespeichert. So können diese jederzeit wieder abgerufen werden.

3.1.1.3 Darstellung der Verkehrsdaten

Der Nutzer*innen sollen über eine Webseite auf die Daten zugreifen können. Auf der Webseite werden alle Messpunkte angezeigt und die Nutzer*innen erhalten sofort die Information über den aktuellen Stand des Verkehrs bzw. die Verkehrssituation. Durch das Drücken auf den Messpunkt, werden die restlichen Daten in Detail angezeigt. Auf einem Graphen können anschließend die Rushhour und der Verkehrsfluss abgelesen werden.

3.1.2 Grundlagen und Methoden

3.1.2.1 Backend

3.1.2.2 Server-Standort

Als Erstes sollte die Frage geklärt werden, ob ein Cloud-Server oder ein Self-hosted Server verwendet werden soll. Da die Erkennung und Klassifizierung der Fahrzeuge bereits an der Ampel geschieht, ist hierfür keine starke Rechenleistung und keine hohe Datenrate nötig.

Als Entwicklungsserver wird daher ein Self-hosted Server verwendet. Grundbetriebssystem des Servers ist ProxMox VE. Die eigentlichen Anwendungen werden anschließend in einem isolierten Container ausgeführt.

3.1.2.3 Betriebssystem

Als Erstes sollte die Frage geklärt werden, ob ein Cloud-Server oder ein Self-hosted Server verwendet werden soll. Da die Erkennung und Klassifizierung der Fahrzeuge bereits an der Ampel geschieht, ist hierfür keine starke Rechenleistung und keine hohe Datenrate nötig.

3.1.2.3.1 Linux-Distribution

Zuletzt stellt sich die Frage welche Linux-Distribution verwendet werden sollte. Da diese Distribution nur in einem Docker-Container läuft, sollte diese so klein und minimalistisch wie möglich sein.

Zur Auswahl stehen folgende Distributionen:

- Alpine
- Debian-slim
- Arch

Alpine Linux ist eine Linux-Distribution, welche auf einen „kleinen Fußabdruck“ setzt, z. B.. verbraucht ein Alpine Container nur 8. MB. Es verwendet musl statt glibc was zu binary Inkompatibilitätsproblemen führt. Eine weitere Besonderheit von Alpine Linux ist das Init-System. Hierbei setzt es nicht, wie bei anderen Systemen üblich, auf SystemD, sondern auf openRC. Alpine wird hauptsächlich dort verwendet, wo kleine und sichere Linux Umgebungen gebraucht werden (z. B.: OpenWrt, postmarketOS,...).

Debian-slim ist eine minimalistische Version der Linuxdistribution Debian. Während Alpine Linux von Grund auf anders aufgebaut ist, werden bei Debian-slim nur bestimmte Programme und Bibliotheken nicht mitgeliefert.

Arch Linux wäre eine weitere Distribution die sehr minimalistisch ist, jedoch basiert es auf dem Rolling-Release Prinzip, welches nicht sinnvoll für eine Produktionsumgebung ist.

Da die Anwendung später als Docker-Image vertrieben werden soll, ist ein minimaler Fußabdruck des Betriebssystems von großem Vorteil. Ebenso sollte der Container schnell starten und einsatzbereit sein. Dies sind alles Eigenschaften von Alpine Linux und aus diesem Grund fiel die Wahl auf diese Distribution.

3.1.2.3.2 Programmiersprache

Grundsätzlich würden viele Programmiersprachen die Anforderungen für die Umsetzung erfüllen. Jedoch wird die Auswahl auf folgende beschränkt

- Python mit Django
- Java mit Spring

Diese limitierte Auswahl ist darauf rückzuführen, dass Python eine einfach zu lernende Sprache ist, die zudem sehr flexibel ist. Java wurde ausgewählt, da wir diese bereits in der Schule gelernt haben und wir bisher am häufigsten mit dieser Programmiersprache gearbeitet haben.

3.1.2.3.3 Framework

Zuletzt stellt sich noch die Frage, welches Framework Anwendung finden soll. Wie oben erwähnt, besteht die Auswahl zwischen Django oder Spring.

Django

Django ist ein Python-Framework, das einen schnellen und einfachen Weg bietet, einen Webservice zu erstellen. Django bezeichnet sich selbst als ein „Framework with batteries included“, das heißt, es beinhaltet bereits viele Funktionen für die andere Bibliotheken verwendet werden müssen (z. B. Django Admin Interface, Database, SQLlite3 etc.). Es kann unter anderem Benutzerauthentifizierung verwalten, Formulare erstellen und verarbeiten sowie Dateien empfangen und speichern. [sha20]

Spring

Spring ist ein Java-Framework, das ursprünglich entwickelt wurde, um den Umgang mit Enterprise JavaBeans zu erleichtern. Hierzu benutzt es Aspect-Oriented Programming (AOP), Plain Old Java Object (POJO) und dependency injection (DI). Das Framework bietet folgende Funktionen:

Das Framework bietet folgende Funktionen:

- #### 3.1.2.3.4 IoC container

  Dieser Container kümmert sich um den Lebenszyklus und die Abhängigkeiten von Objekten.

- #### 3.1.2.3.5 Data access framework

  Dieses Framework vereint unterschiedlichste Datenzugriffsmethoden in einem Framework(JDBC,Redis,MongoDBetc.

- #### 3.1.2.3.6 Spring MVC framework

  Durch dieses Framework wird das Erstellen von MVC architektur-basierten Webapplikationen ermöglicht.

- #### 3.1.2.3.7 Transaction management

  Dieses hilft dabei, Transaktionen durchzuführen. Dieses Framework ist nützlich, wenn Daten stets vor Verlust gesichert sein müssen, wie z. B. bei einem Stromausfall oder einem Systemabsturz. Dies geschieht, indem nach Abschluss der Transaktion die Daten sofort gespeichert und übernommen werden.

- #### 3.1.2.3.8 Spring Web Service

  Das Spring Web Service erleichtert die Übermittlung von Daten über das Internet. So werden eingehende XML bzw. Json Daten direkt in ein Objekt eingelesen.

- #### 3.1.2.3.9 JDBC abstraction layer

  Dieses Framework hilft beim Abfangen und Verarbeiten von Fehlern in der JDBC-Ebene.

- #### 3.1.2.3.10 Spring TestContext framework

  Dieses Framework hilft bei der Erstellung von Integration und Unit Testings.

[rom19]

3.1.2.3.11 Datenbank

Um die Daten zu speichern, wird eine Datenbank benötigt. Diese sollte flexibel genug sein, um gut mit Daten und mit Messdaten umgehen zu können.

MariaDB und InfluxDB

Zu Beginn wurden hierzu zwei Datenbanken verwendet: MariaDB, eine relationale Datenbank, und InfluxDB, eine zeitbasierte Datenbank. Jedoch hat sich während des Projektes bei der Datenbank InfluxDB Version 2 die Art der Datenspeicherung geändert. So konnte nun nur mehr ein Wert je Messung gespeichert werden, was dazu führt, dass für eine Messung drei Einträge erstellt werden müssen. Um die Vorgehensweise zu erleichtern, wurden Alternativen gesucht.

PostgreSQL mit TimeScaleDB

TimeScaleDB ist ein Plugin für PostgreSQL, welches erlaubt, Funktionen einer zeitorientierten Datenbank in PostgreSQL zu verwenden. So können wie bei InfluxDB Statistikdaten live abgerufen werden, ohne dass alle Daten analysiert werden müssen, was bei manchen Operationen viel Zeit spart.

3.1.2.4 Frontend

Damit alle gesammelten Daten übersichtlich dargestellt werden können, wird eine Webseite benötigt, welche eine Karte mit allen Nodes darstellt. Hierzu wird ein Framework benötigt, das eine Karte anzeigt und anschließend Punkte mit Informationen einzeichnet.

3.1.2.4.1 JavaScript

Mithilfe von JavaScript(JS) kann Code auf der Client-Seite ausgeführt werden. Somit können Operationen lokal am Rechner ausgeführt werden und müssen nicht am Server verarbeiten werden. So können Daten mittels einer API abgefragt und am Client dargestellt werden. Zahlreiche JS Bibliotheken ermöglichen es, schnell komplexe Webseiten zu erstellen.

Vue.js

Vue.js ist ein progressives JavaScript Framework, welches es ermöglicht, JS noch effektiver zu nutzen. So können Webseiten reaktiver gestaltet werden. Wenn sich zum Beispiel ein Wert im JavaScript verändert, wird dieser auch auf der ganzen Webseite geändert (Live-Updates).

Leaflet.js

Leaflet.js ist eine Open-Source-JavaScript-Library, welche es ermöglicht, ganz einfach eine mobiltelefonfreundliche, interaktive Karte zu erstellen. Der Ursprung dieser Karten ist frei zu wählen. In diesem Projekt wir OpenStreetMap(OSM) verwendet. OSM ist ein community driven Open-Source-Kartendienst.

Chart.js

Um den Verkehrsandrang zu visualisieren, wird ein Graph gezeichnet. Um diesen dynamisch und einfach zu zeichnen, wird die Chart.js Bibliotheken verwendet. Diese ist wie alle anderen Bibliotheken in diesem Projekt Open Source.

Axios.js

Axios ist ein `promise` basierter HTTP Client. Dieser kann sowohl im Browser als auch auf einem Node.js Server verwendet werden. Ein JavaScript-Promise ermöglicht es, mit Daten zu arbeiten, die am Zeitpunkt der Bearbeitung noch nicht vorhanden sind. Das Programm kann zum Beispiel weiter arbeiten, wenn eine API-Anfrage gemacht wird. Eine Promise hat drei Zustände:

- pending: Der Status während auf das Ergebnis gewartet wird.
- fulfilled: Wird ausgegeben, wenn die Daten ankommen.
- rejected: Wird ausgegeben, wenn die Daten nicht ankommen.

[Sar20]

3.1.2.4.2 TypeScript

TypeScript erweitert JavaScript mit statischen Datentypen und weiteren Hilfsmitteln, die es ermöglichen, Fehler in JS vorzubeugen.

3.1.2.4.3 Bootstrap

Bootstrap ermöglicht es, mit einem einfachen Workflow, interaktive Webseite zu erstellen. Dies geschieht mithilfe von HTML-Klassen. Man schreibt die gewünschte Eigenschaft in den HTML-Tag und Bootstrap kümmert sich im Hintergrund um die CSS-Formatierung.

3.1.2.4.4 SCSS

Syntactically Awesome Stylesheets kurz „SASS“ bzw. „SCSS“ ist eine Stylesheet-Sprache, welche es vereinfacht CSS zu schreiben. Es ermöglicht unter anderen das Anlegen von Variablen und Funktionen, welche im CSS-Code verwendet werden können. Zusätzlich wird das CSS nicht in eine Datei ,wie es normalerweise der Fall ist, geschrieben, sondern in kleinere SASS-Dateien, welche anschließend vom SCSS-Compiler zu einer optimierten CSS-Datei kompiliert werden.

3.1.2.5 GeoJson

GeoJson ist eine standardisierte JSON-Darstellung, welche für geografische Datenstrukturen geeignet ist. Eine GeoJson-Datei kann den normalen JSON-MIME-Type haben (`application/json`).

- Point: Ein Point besteht aus Koordinaten und symbolisiert einen einfachen Punkt auf der Karte
- LineString: Ein LineString ist eine Linie aus verschiedenen Koordinaten
- Polygon: Ein Polygon besteht aus den gegebenen Koordinaten
- MultiPoint: Dies ist eine Gruppe aus Punkten
- MultiLineString: Dies ist eine Gruppe aus verschiedenen Linien
- MultiPolygon: Dies ist eine Gruppe aus verschiedenen Polygonen
- GeometryCollection: Darunter versteht man eine Gruppe, die alle oben genannten Typen beinhalten kann

[But+16]

3.1.3 Realisierung

3.1.3.1 Entwicklungsumgebung Backend

Um mit der Entwicklung des Backends beginnen zu können, müssen zuerst einige Vorkehrungen getroffen werden. Als Erstes wird eine geeignete IDE benötigt mit welcher das Programm entwickelt wird. Für dieses Projekt wurde Eclipse und IntelliJ verwendet. In beiden Umgebungen lässt sich über den Add-on-Store eine Erweiterung für Spring herunterladen welche das Arbeiten mit Spring um einiges verbessert.

Um eine einfache Möglichkeit zu bieten, die Datenbank zu analysieren und zu bearbeiten wurde DBeaver verwendet, welches eine Open Source Alternative für die MySQL-Workbench ist. DBeaver erlaubt es die Daten aus der Datenbank zu lesen und diese tabellarisch dazustellen. Es können auch SQL-Queries getestet werden.

Die Datenbank wird als Docker Container geliefert. Mithilfe von docker-compose ist es möglich diese in Sekundenschnelle aufzurufen.

3.1.3.2 Entwicklungsumgebung Frontend

Als IDE für das Frontend wird VSCodium verwendet. VSCodium ist ein Fork von Visual Studio Code von Microsoft ohne jeglichen Telemetrie Code.

Vue.JS kann einfach über NPM installiert werden.

3.1.3.3 Backend – Setup

3.1.3.3.1 Maven

Maven ist ein Projektmanagement-Tool für Java, welches den Build-Prozess eines Projektes vereinfachen soll. Maven hilft auch bei der Verwaltung von Abhängigkeiten im Programm. Dies hat den Vorteil, dass man sich keine Gedanken machen muss, ob man eine Bibliothek installiert hat, da Maven diese automatisch herunterlädt und zum Projekt hinzufügt. Auch das Ausführen von JUnit-Test, funktioniert mit Maven.

3.1.3.3.2 Spring Boot Starter

Mithilfe von Spring Boot Starter kann ein Spring-Projekt erstellt werden und in eine IDE importiert werden.

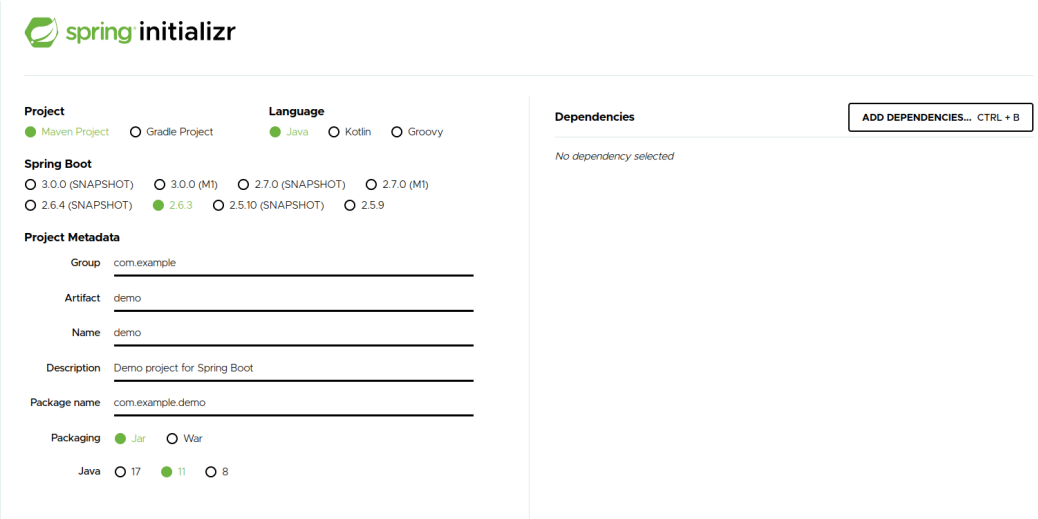


Abbildung 3.1: Spring Initializr

3.1.3.4 JPA und Hibernate

Mithilfe JPA und Hibernate wurde eine Verbindung zwischen den Objekten im Programm und einer Datenbank erzeugt. Hierbei hilft das Spring-Framework indem es mittels Auto Dependency Injection die Erstellung einer Verbindung sowie von Klassen vereinfacht.

Um die Konfiguration der Datenbankverbindung vorzunehmen, müssen die Verbindungseigenschaften in eine Konfigurationsdatei geschrieben werden.

```
spring:
  datasource:
    url: jdbc:postgresql://localhost:5432/commean_dev
    driver-class-name: org.postgresql.Driver
    username: user
    password: password
  jpa:
    show-sql: true
    database-platform: org.hibernate.dialect.PostgreSQLDialect
    hibernate:
      ddl-auto: create

corsserver: http://localhost:8081/
```

Anhand der Konfigurationsdatei kann Hibernate in Zusammenarbeit mit Spring eine Verbindung zur Datenbank herstellen und somit Daten austauschen. Um Daten jedoch zu speichern wird zuerst ein Java Objekt benötigt, welches mittels Java Annotation zu einer Hibernate Entity wird. Als Beispiel wird hier das `Node` Objekt genommen.

```java
@Entity
@Table(name = "nodes")

@EnableAutoConfiguration

public class Node {
  @Id
  @GenericGenerator(name = "uuid2", strategy = "uuid2")
  private UUID id;

  private String location;

  @ManyToOne

  @JoinColumn(name = "crossroad_id", referencedColumnName = "id")
  private Crossroad crossroad;

    //Getter and Setters
```

Mithilfe dieses Codes werden im Hintergrund SQL-Statements generiert, welche auf der Datenbank Tabellen und Abhängigkeiten zwischen den einzelnen Tabellen erstellen. Will man später auf die Daten zugreifen, benötigt man ein CRUD-Repository. Hierzu wird einfach eine Java-Klasse erstellt, die von `CrudRepository` erbt. Dieser wird einer Entity und dessen Id-Type übergeben, in diesen Fall eine UUID.

```java
@Repository
public interface NodeRepository extends CrudRepository<Node, UUID>
 ↪  {
  Iterable<Node> findAllByCrossroad(Crossroad crossroad);

  Iterable<Node> findAllWhereLocationNotNull();
}
```

Das CRUD-Repository beinhaltet nur die Grundmethoden zur Datenbearbeitung. Jedoch lässt sich der Umfang der Operationen leicht erweitern. So kann Spring anhand des Funktionsnamen ein dazugehöriges SQL-Statement generieren. Wie z. B. `Iterable<Node> findAllWhereLocationNotNull()`. Diese Funktion sucht in der Datenbank nach allen `Nodes`, in denen die Spalte `Position` undefiniert ist.

Nun kann bereits mit der Datenbank kommuniziert und Dateien gespeichert werden. Jedoch wir empfohlen, ein Servicelayer zwischen den Repository und den anderen Ebenen einzufügen, da die darunterliegende Ebene verändert werden kann, ohne, dass sich im ganzen Programm etwas ändert (SRP). Dies wird mit dem `NodeService` und `NodeServiceImpl` realisiert.

```java
public interface NodeService {

  Node addNode(Node tcn);
  List<Node> getAllNodes();
  Node getNodeById(UUID id);
  List<Node> getNodesByCrossroad(Crossroad c);
  List<Node> getAllNodesWhereLocatioNotNull();
  void deleteNodeById(UUID id);
}
```

```java
@Service
public class NodeServiceImpl implements NodeService {

  NodeRepository cameraNodeRepository;

  @Autowired
  public NodeServiceImpl(NodeRepository cameraNodeRepository) {
    this.cameraNodeRepository = cameraNodeRepository;
  }
  @Override
  public Node addNode(Node tcn) {
    return cameraNodeRepository.save(tcn);
  }
    //...
  @Override
  public void deleteNodeById(UUID id) {
    cameraNodeRepository.deleteById(id);

  }

}
```

Somit können nun Daten zwischen der Datenbank und dem Java-Programm ausgetauscht werden.

3.1.3.5 API-Controller

Um eine Kommunikation zwischen Backend, Frontend und den einzelnen Messstationen herzustellen, ist eine API notwendig. Mithilfe von Spring Boot Web kann diese einfach realisiert werden. Mittels der Java Annotation `@RestController` und `@GetMapping` kann eine Klasse in eine API-Schnittstelle verwandelt werden. Als Beispiel ist wieder die `Node`-Klasse verwendet worden.

```
@RestController
@RequestMapping("/api/v1/nodes")
public class NodeController {

  @GetMapping(value = "", produces = "application/json")
  @ResponseStatus(code = HttpStatus.OK)
  public Object getNode(@RequestParam("id") UUID uuid) {
    return NodeDto.convertToDto(tcns.getNodeById(uuid));
  }
    //...
}
```

Der oben angeführte Code gibt bei Aufruf der URL http://localhost:8081/api/v1/nodes?id=bba2752d-4d47-416b-96e6-fd875cbfe4ac das gewünschte `Node`-Objekt retour. Jedoch hat nun jeder Zugriff auf diesen Endpunkt. Bei manchen Operationen, wie beim Erstellen von Messdaten, könnte dies aber nicht gewollt sein. Hierzu muss eine Autorisierungs- und Authentifizierungsebene hinzugefügt werden.

3.1.3.6 Spring Security

Soll ein Endpunkt nur für bestimmte Benutzer*innen zugänglich gemacht werden, kann das mit Spring Security umgesetzt werden. Dieser Teil des Spring-Frameworks verwaltet Authentifizierung und Autorisierung in einer Spring-Applikation. In diesem Projekt wird eine Benutzerauthentifizierung benötigt.

3.1.3.6.1 Security Konfiguration

Die Security Konfiguration von Spring legt fest, welche Funktion bzw. welcher Endpunkt von Spring Security überprüft werden soll.

```
public class AuthConfig extends WebSecurityConfigurerAdapter {

  @Override
  protected void configure(HttpSecurity http) throws Exception {

    http.headers()
      .frameOptions().sameOrigin()
```

```
        .and()
        .csrf().disable()
        .authorizeRequests()
        .antMatchers("/api/v1/auth/**").permitAll()
        .antMatchers(HttpMethod.POST, "/api/v1/measurements/**").
        ↪  authenticated()
        .antMatchers(HttpMethod.PUT, "/api/v1/nodes").authenticated()
        .and()
        .sessionManagement().sessionCreationPolicy(
        ↪  SessionCreationPolicy.STATELESS)
        .and()
        .addFilterBefore(jwtAuthenticationFilter(),
        ↪  UsernamePasswordAuthenticationFilter.class);

    }
}
```

3.1.3.6.2 Benutzerauthentifizierung

Alle Nodes werden als ein Benutzer angesehen. Dieser Benutzer gehört zur Rolle `node`. Hiermit kann ihr Zugriff auf bestimmte Endpunkte zusätzlich blockiert werden. So kann zum Beispiel nur ein Benutzer der Rolle `admin` eine `Node` anlegen. Um diese Benutzer Rollen zuzuweisen, ist eine Datenbank erforderlich. Folgendes Diagramm zeigt das Schema:

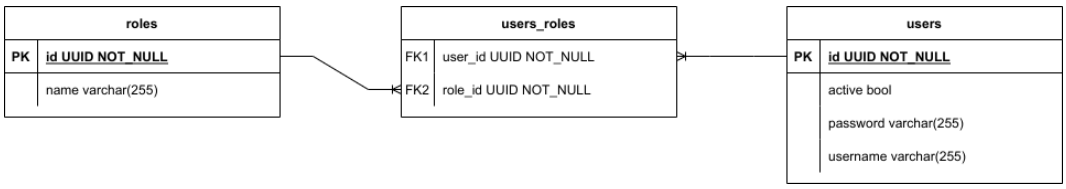


Abbildung 3.2: ER-Tabelle der Benutzer

Um diese Informationen über den Benutzer zu speichern, stellt Spring das `UserDetails` Interface bereit. Dies ermöglicht das Speichern von Nutzerdaten wie E-Mail-Adresse oder sonstige Informationen im von Spring bereit gestellten Benutzer Objekt – ohne dieses neu zu implementieren. Denn durch eine Neuimplementierung kann es zu Schwachstellen in Spring Security kommen. Diese Daten werden anschließend an das `Authentication` übergeben, wo diese für andere Operationen verwendet werden können.

```
public class UserPrincipal implements UserDetails {

```

```
3   private String username;
4   private String password;
5   private boolean enabled;
6   private Collection<? extends GrantedAuthority> authorities;
7
8   public static UserPrincipal userToPrincipal(User user) {
9     UserPrincipal userPrincipal = new UserPrincipal();
10    List<SimpleGrantedAuthority> authorities = user.getRoles().
      ↪  stream()
11        .map(role -> new SimpleGrantedAuthority("ROLE_" + role.
          ↪  getName())).collect(Collectors.toList());
12
13    userPrincipal.setUsername(user.getUsername());
14    userPrincipal.setPassword(user.getPassword());
15    userPrincipal.setEnabled(user.getActive());
16    userPrincipal.setAuthorities(authorities);
17    return userPrincipal;
18  }
19    //Getters and Setters
20 }
```

Der oben angeführte Code beinhaltet die Implementation in diesem Projekt. Da in diesem Projekt keine genaueren Daten wie E-Mail-Adressen oder sonstige Informationen benötigt werden, ist diese Klasse minimal gehalten.

Die Registrierung und Anmeldung der Benutzer erfolgt über die REST-Schnittstelle. Das Erstellen von Benutzern ist nur der Rolle `admin` gestattet. Das Begrenzen von Methoden auf bestimmte Gruppen geschieht mithilfe der `@Secured()` Annotation. Folgender Code zeigt die Methode zur Registrierung eines Nutzers.

```
1   @Secured("admin")
2   @PostMapping(value = "/signup", consumes = "application/json",
    ↪  produces = "application/text")
3   public ResponseEntity<?> registerUser(@RequestBody SignUpDto
    ↪  signUpDto) {
4
5     if (userRepository.existsByUsername(signUpDto.getUsername())) {
6       return new ResponseEntity<>("Username already taken!",
        ↪  HttpStatus.BAD_REQUEST);
7     }
```

```
8       User user = new User();
9       user.setUsername(signUpDto.getUsername());
10      user.setPassword(passwordEncoder.encode(signUpDto.getPassword
        ↪  ()));
11      user.setActive(true);
12
13          if (signUpDto.getRole()==null)
14      Role roles = roleRepository.findByName(RoleName.NODE);
15          else
16          Role roles = roleRepository.findByName(signUpDto.getRole
            ↪  ());
17          if (roles != null)
18      user.setRoles(Arrays.asList(roles));
19
20      userRepository.save(user);
21      return new ResponseEntity<>("User registered successfully",
        ↪  HttpStatus.CREATED);
22    }
23  }
```

Damit sich ein Benutzer anmelden kann und auch angemeldet bleibt, kommen verschiedenste Technologien zum Einsatz. In diesem Projekt ist es die JWT-Technologie. Sollte sich nun ein Benutzer anmelden, bekommt er vom Server einen JSON Web Token zurück.

JWT

JSON Web Token (kurz JWT) sind sogenannte Claims, welche benutzt werden, um Daten über einen sicheren Weg zu übertragen. Diese Claims können anschließend dazu verwendet werden, um einen Benutzer zu authentifizieren. Ein JWT besteht aus drei Teilen:

Dem Header, welcher Informationen über den Medientyp und die verwendeten kryptografischen Operationen enthält.

```
1   {
2   "alg": "HS512"
3   }
```

Aus dem Payload, welche die zu übertragene Informationen beinhaltet.

```
{
"sub": "6dc0c104-c681-4712-805a-5dbc06faf73d",
"iat": 1647645165,
"exp": 1663197165
}
```

Der letzte Teil ist die Signatur und besteht aus einem Hash der obigen Daten, der mit einem am Server vorhandenen Secret key verschlüsselt wird.

Zur Übertragung werden die JSON Daten mittels Base64 Codierung zu einer URL sicherem String konvertiert. Dieser wird im folgenden Format übertragen: `[header].[payload].[signature]` [nee21]

Die Vorteile gegenüber anderen Methoden besteht darin, dass ein JWT nicht auf dem Server gespeichert werden muss. Da er mit einer Signatur versehen ist, welche vom Server erstellt worden ist, kann dem Inhalt vertraut werden und es ist nicht nötig, in der Datenbank zu überprüfen, ob es sich wirklich um diesen Nutzer handelt. Dies hat wiederum den Nachteil, dass ein JWT, welcher ausgestellt worden ist, seine Gültigkeit erst verliert, wenn dieser abgelaufen ist. Hierfür muss man ein Ablaufdatum (kurz exp) in den Payload-Bereich schreiben. [JBS15]

Die Implementation im Backend erfolgt mittels Spring Security und jjwt – eine Bibliothek für die Erstellung und Überprüfung von JWT Token. Der unten angeführte Code hilft bei der Verwaltung von JWTs.

```
@Component
@Log4j2
public class JwtProvider {

  public String generateToken(Authentication authentication) {
    UserPrincipal userPrincipal = (UserPrincipal) authentication.
     ↪  getPrincipal();
    Timestamp expDate = Timestamp.from(Instant.now().plus(
     ↪  jwtExpirationInDays, ChronoUnit.DAYS));

    return Jwts.builder()
        .setSubject(userPrincipal.getUsername())
        .setIssuedAt(Date.from(Instant.now()))
        .setExpiration(expDate)
        .signWith(getSecretKey())
        .compact();
  }
}
```

Die hierdurch erstellten JWTs beinhalten den Benutzernamen zur Authentifizierung und ein Ablaufdatum sowie die Rolle des Benutzers. Hierdurch ist es nicht mehr nötig, eine Anfrage auf die Datenbank zu machen, da die Angaben in der Payload aufgrund der Signatur vertrauenswürdig sind.

3.1.3.7 MQTT und The Things Network

Um Daten zwischen Nodes und dem Server auszutauschen, kann auch MQTT verwendet. Hierzu wird The Things Network(kurz TTN) verwendet. Daten werden von den einzelnen Nodes an ein TTN-Gateway geschickt, welches anschließend die Daten an einen TTN-Server schickt. Das Server Backend kann sich nun via MQTT mit dem TTN-Server verbinden und bekommt die einzelnen Daten der Nodes.

3.1.3.7.1 MQTT Client

Der MQTT Client wird mithilfe der Paho MQTT Bibliothek implementiert. Diese Bibliothek bietet zwei Schnittstellen zur Kommunikation. Einen asynchronen und einen synchronen Client. Zweiteres ist jedoch nur ein Wrapper, welcher den asynchronen Client als einen synchronen darstellt.

Für dieses Projekt wurde die asynchrone Implementation verwendet. Im Gegensatz zum synchronen Client, bei dem das Programm auf eine MQTT-Nachricht wartet, erstellt der asynchrone Client einen eigenen Thread. Sollte nun eine Nachricht von MQTT empfangen werden, so wird eine Callback-Methode aufgerufen. Diese kann dann die MQTT-Nachricht verarbeiten.

Um eine Verbindung mit dem MQTT-Server herzustellen, muss der Client vorkonfiguriert werden. Da der MQTT-Server passwortgeschützt ist und nur verschlüsselte Verbindungen akzeptiert, müssen zuerst einige Voreinstellungen vorgenommen werden. Hierzu wurde eine Klasse erstellt, die das Verwalten der MQTT-Verbindung um einiges erleichtert.

```
1  public static void create() {
2      if (Client.asyncClient == null) {
3        try {
4          asyncClient = new MqttAsyncClient(props.getTtnUrl(), props.
            ↪  getClientId());
5          MqttConnectOptions connOpts = new MqttConnectOptions();
6          connOpts.setCleanSession(true);
7          connOpts.setAutomaticReconnect(true);
8          connOpts.setUserName("%s@ttn".formatted(props.getTtnAppId
            ↪  ()));
9          connOpts.setPassword(props.getTtnKey());
10         MqttToken token = (MqttToken) asyncClient.connect(connOpts
            ↪  );
11         token.waitForCompletion(5000);
12         log.info("MQTT connected");
13       } catch (MqttException e) {
```

```
14            log.error("MQTT connection failed \n{}", e.toString());
15        }}
16
```

Der gezeigte Code ist ein Ausschnitt der vorher genannten Klasse und initialisiert die Verbindung mit dem MQTT-Server. Die Daten, welche zur Anmeldung gebraucht werden, werden aus Umgebungsvariablen ausgelesen und im Programm verwendet.

MQTT stellt Daten unter Topics bereit. Will man die Daten von einem bestimmten Topic haben, so muss man dieses abonnieren. Der von TTN bereitgestellte MQTT-Server stellt die Daten unter folgenden Topic bereit : `v3/<Application Name>@ttn/devices/<Device Id>/up`. Der Applikationsname wird in der Konfigurationsdatei des Backends festgelegt, oder kann über eine Umgebungsvariable bereitgestellt werden. Die Geräte-Id wird bei der Erstellung einer Node angegeben. Aus diesen Daten wird anschließend der Pfad für das Topic gebaut und abonniert.

Nun wird das Backend bei jeder Aktualisierung der Daten am MQTT-Server darüber informiert und bekommt die neuen Daten. Da ein asynchroner Client die eingegangenen MQTT Nachrichten als Event ausgibt, wird eine Callback Klasse benötigt. Diese Klasse wird wie folgend implementiert:

```
1  public class WriteToLogCallback implements MqttCallback {
2    @Override
3    public void connectionLost(Throwable cause) {
4
5    }
6
7    @Override
8    public void messageArrived(String topic, MqttMessage message)
     ↪  throws Exception {
9
10   }
11
12   @Override
13   public void deliveryComplete(IMqttDeliveryToken token) {
14
15   }
16 }
```

In der `messageArrived` kann anschließend auf das Topic und die Nachricht zugegriffen werden. Anschließend wird der von TTN erhaltene JSON string auf ein Java Objekt mappen. Nach diesem Schritt können die Daten so behandelt werden, als würden sie über die REST-API gesendet worden sein.

3.1.3.8 Verarbeitung der Messdaten

Die von der Node erhaltenen Messdaten beinhalten die Anzahl der Fahrzeuge, deren Typ (PKW, LKW, Motorrad, Bus, etc.) sowie die Zeit, die diese gebraucht haben, um durch den Aufnahmebereich zu fahren. Diese Daten werden in einer Datenbank gespeichert.

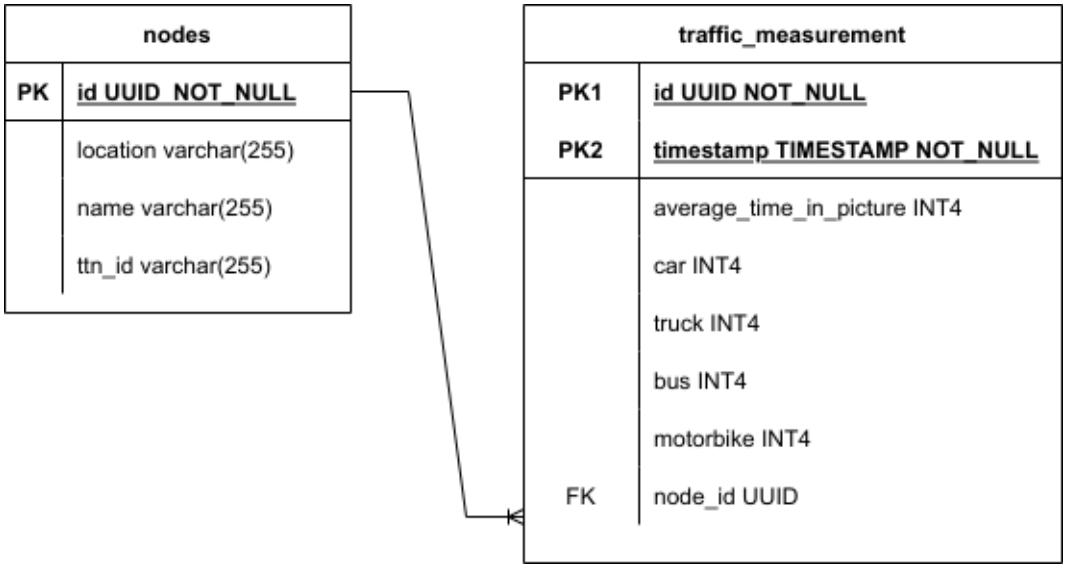


Abbildung 3.3: ER-Tabelle der Nodes und Measurements

Da die Daten nicht kontinuierlich übertragen werden, sondern nur dann, wenn eine Verbindung besteht und Fahrzeuge gezählt worden sind, muss der Zeitpunkt der Messung extra gespeichert werden. Will man nun diese Daten ausgeben, kommt es zu Lücken in der Statistik. Um diese Lücken zu füllen, gibt es in TimescaleDB die sogenannte `time_bucket` Funktion. Diese Funktion ermöglicht das Gruppieren von Messdaten nach einem angegebenen Intervall.

Zeit	Gerät	Temperatur
2020-01-01 00:00:00.000 +0100	3	33.08
2020-01-01 00:20:00.000 +0100	3	-34.57
2020-01-01 00:40:00.000 +0100	3	-6.03
2020-01-01 02:20:00.000 +0100	3	-22.06
2020-01-01 04:30:00.000 +0100	3	4.54
2020-01-01 04:40:00.000 +0100	3	23.72
2020-01-01 05:10:00.000 +0100	3	34.76
2020-01-01 06:10:00.000 +0100	3	-27.93
2020-01-01 06:20:00.000 +0100	3	-5.93
2020-01-01 01:20:00.000 +0100	2	-37.08
2020-01-01 01:50:00.000 +0100	2	22.59
2020-01-01 02:00:00.000 +0100	2	-34.16
2020-01-01 02:50:00.000 +0100	2	11.14
2020-01-01 03:20:00.000 +0100	2	10.36
2020-01-01 04:50:00.000 +0100	2	11.02
2020-01-01 05:00:00.000 +0100	2	-0.6

Tabelle 3.1: Normale Querry

Gerät	Zeitraum	Temperatur Druchschnitt
3	2020-01-01 01:00:00.000 +0100	1.18
2	2020-01-01 01:00:00.000 +0100	-2.38

Tabelle 3.2: time_bucket Querry

Wie in der oben angeführten Tabelle zu sehen ist, werden die Daten von einem Tag zusammengefasst und der Durchschnittswert an diesem Tag wird berechnet.

Eine weiter wichtige Funktion ist das Continuous Aggregate, dieses erlaubt bestimmte statistische Werte, wie zum Beispiel den Durchschnitt, zu berechnen und zu speichern. So kann dieser jederzeit abgegriffen werden, ohne das tausende Daten abgerufen werden müssen und mit ihnen der Durchschnitt berechnet werden muss. Sollten sich im Hintergrund Daten ändern, wird nur der Teil des Durchschnittes neu berechnet, in dem sich Daten verändert haben.

Mithilfe dieser zwei Funktionen kann die durchschnittliche Wartezeit an einer Ampel berechnet werden. Hierzu werden in einen Bereich von 15 Minuten von allen Fahrzeugen das durchschnittliche Aufkommen berechnet sowie die durchschnittliche Zeit im Bild abgespeichert. Dies ermöglicht, das für spätere Berechnungen nicht rund 900 Messdaten ausgewertet werden müssen, sondern nur ein einziger Messwert ausgewertet werden muss.

3.1.3.9 Darstellung der Messdaten

Nachdem die Messdaten ausgewertet worden sind, müssen diese visualisiert werden. Hierzu gibt es das Frontend, welches mittels Vue.Js und Leaflet die Livedaten auf einer Karte der Stadt visualisiert.

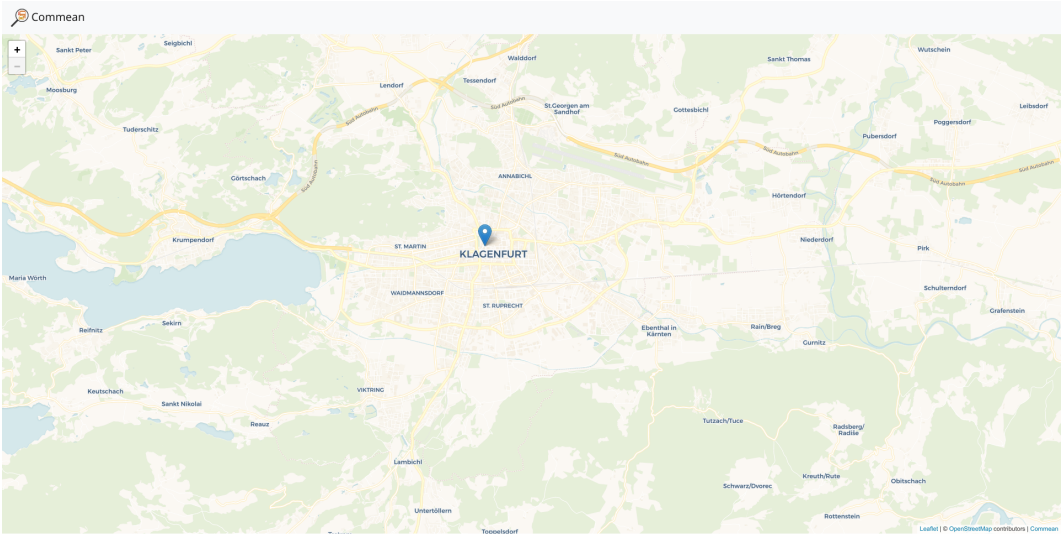


Abbildung 3.4: Webseite mit Karte

3.1.3.9.1 Veranschaulichung der Daten

Alle Nodes werden auf einer Karte der Stadt angezeigt und mittels eines Klicks auf den Marker wird ein Pop-up mit den Messdaten der Nodes angezeigt. Um die Daten für jeden verständlich auf der Karte anzuzeigen, werden diese mittels Piktogrammen visualisiert. Weiteres wird der Zeitverlust im Vergleich der vorherigen Tage angezeigt.

Abbildung 3.5: Pop-up mit Verkehrsdaten

Um die oben gezeigten Features zu implementieren, wird JavaScript und das Vue.JS-Framework benötigt. Folgender Teil befasst sich mit der Installation und Implementation der Funktionen.

3.1.3.10 Vue.js und Node.js

3.1.3.10.1 Setup der Umgebung

Um mit der Programmierung zu beginnen, wird ein Runtime-Environment benötigt. Hierzu muss Node.JS installiert werden. Unter Arch basierten Linux-Distributionen kann dies über folgenden Befehl installiert werden:

```
sudo pacman -S nodejs npm
```

Dieser Befehl installiert automatisch die neueste Version von NodeJS. Will man eine ältere Version verwenden, muss diese manuell nach installiert werden.

Zusätzlich wird das VueJS-Framework verwendet. Es gibt mehrere Wege VueJS zu installieren, in diesem Projekt wurde Vue CLI verwendet. Vue CLI sollte dann verwendet werden, wenn ein komplexeres Projekt geplant ist. Diese wird über npm installiert. Folgender Befehl muss ausgeführt werden, um Vue CLI global zu installieren:

```
npm install --global vue-cli
```

Nach der Initialisierung des Vue-Projektes wird eine Beispielseite geladen. Mit dem Befehl `npm run serve` kann diese Webseite ausgeführt werden. Vue verarbeitet nun das Projekt und stellt es über einen Webserver bereit.

```
DONE  Compiled successfully in 2801ms


  App running at:
  - Local:   http://localhost:8081/
  - Network: http://10.0.0.110:8081/

  Note that the development build is not optimized.
  To create a production build, run npm run build.
```

Abbildung 3.6: Konsolenausgabe

Wenn nun die oben angegebenen URL im Browser aufgerufen wird, kommt man auf das Standardprojekt von VueJS.

3.1.3.10.2 Installation von Libraries

Will man nun externe Bibliotheken zu seinem Projekt hinzufügen, kann man dies mithilfe von `npm` mit nur einem Befehl durchführen. Zuerst sucht man sich den Namen der Bibliothek heraus und installiert diese mit NPM:

```
npm install vue-leaflet
```

Um die Bibliothek verwenden zu können, muss diese nur am Anfang des Skriptes importiert werden:

```
import axios from "axios";
```

Will man eine normale JavaScript-Bibliothek auch in einer Vue-Komponente verwenden, so muss man diese separat in VueJS importieren.

```
1 import VueSweetalert2 from "vue-sweetalert2";
2 import "sweetalert2/dist/sweetalert2.min.css";
3 const app = createApp(App);
4
5 app.use(VueSweetalert2);
```

3.1.3.10.3 Vue

Im Gegensatz zu einer normalen Webseite wird der Quellcode nicht in HTML-Dateien geschrieben. Die einzige HTML Datei, welche vorhanden ist, ist die `index.html`. Diese beinhaltet lediglich die Grundstruktur der Webseite und nicht viel mehr. Alles, was in dieser Datei steht, ist statisch und lässt sich nicht von Vue verändern. Will man aber die Funktionen von Vue verwenden, so muss man eine Vue-Komponente einbinden. Standardmäßig heißt die Hauptkomponente `App`. Diese wird über ein `<div>` Element in die Webseite eingebunden. Der unten angeführte Code zeigt die `index.html` eines Standard Vue-Projektes.

```
<!DOCTYPE html>
<html lang="">
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content=
     ↪  "width=device-width,initial-scale=1.0" />
    <link rel="icon" href="<%= BASE_URL %>favicon.svg" />
    <title><%= htmlWebpackPlugin.options.title %></title>
  </head>
  <body>
    <noscript>
      <strong
        >This site doesn't work properly without JavaScript
         ↪  enabled.</strong>
    </noscript>
    <div id="app" class="mt-0"></div>
    <!-- built files will be auto injected -->
  </body>
</html>
```

Der Quellcode für die Vue-Komponente wird in einer `.vue` Datei geschrieben. Die oben genannt App-Komponente beinhaltet die gesamte Webseite. Dies hat den Vorteil, dass jedes Element dynamisch erstellt werden kann und so eine Seite ihre komplette Funktion ändern kann, ohne dass diese neu geladen werden muss.

```
<template>
  <TrafficMap />
</template>

<script>
```

```
6  import TrafficMap from "./components/TrafficMap.vue";
7
8  export default {
9    name: "App",
10   components: {
11     TrafficMap,
12   },
13 };
14 </script>
```

Der oben angeführte Code stellt ein Template dar, welches den Vue-Komponent `TrafficMap` beinhaltet. Diese Komponente zeigt die Karte der Stadt mit den einzelnen Nodes sowie die Verkehrsdaten an. Die `App.vue` Klasse, welche zurzeit nur die `TrafficMap`-Komponente beinhaltet, dient dazu, um später das Wechseln zwischen verschiedenen Komponenten zu ermöglichen, ohne dass die komplette Webseite umstrukturiert werden muss.

3.1.3.10.4 Leaflet.js

Um auf der Webseite eine Karte darzustellen, wird Leaflet verwendet. Es besteht die Möglichkeit, die Standard Vue-Bibliothek zu verwenden, jedoch gibt es eine Drittanbieterbibliothek, welche die Integration in Vue unterstützt. Mit dieser Bibliothek ist es dann möglich, Livedaten auf der Karte zu visualisieren

Leaflet und Tiles

Um auf der Karte etwas darzustellen, werden zunächst Daten benötigt. Hierzu verwendet Leaflet.js sogenannte Tiles, um diese zu zeichnen. Diese Tiles bestehen aus JPEG-Bildern, welche aneinandergereiht werden und somit eine Karte ergeben. Hierzu wird ein Tileserver benötigt. Da die Standardversion von OpenStreetMaps einen sehr hohen Detailgrat besitzt, wird ein anderer Tilesever verwendet. Dieser ist jedoch nicht für kommerzielle Verwendung geeignet. Um diese Karte später verwenden zu können, muss diese gekauft werden.

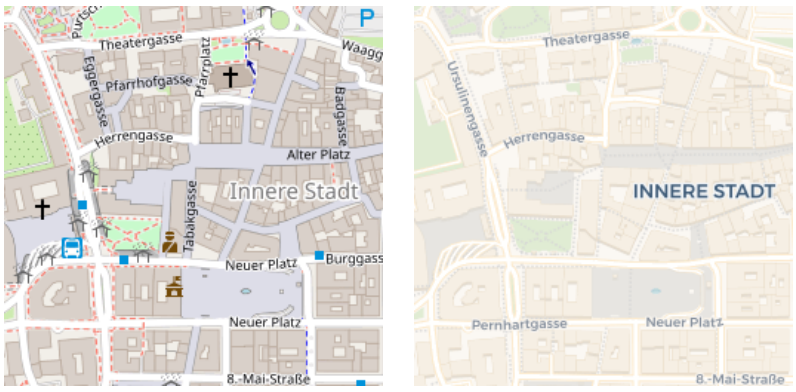


Abbildung 3.7: OSM-Style vs. Voyager-Style

GeoJson

Leaflet unterstützt das Einlesen und Verarbeiten von GeoJson-Daten. Diese werden anschließend von der Bibliothek verarbeitet und können weiter aufbereitet werden. Nachfolgend werden diese auf einem eigenen Layer geschrieben. So können diese nach Belieben eingeblendet werden.

Der unten angeführte Code zeigt den Abschnitt aus der Vue-Template-`TrafficMap`, welcher dafür zuständig ist, Daten vom Backend abzufragen und anschließend der Karte hinzuzufügen.

```
1   methods: {
2       async fetchGeoJson() {
3           try {
4               const {
5                   data
6               } = await GeoJsonRepo.get();
7               this.geojson = data;
8           } catch (error) {
9              //Error handling
10              }
11          }
12      },
13  }
```

Da dieser jedoch nur Punkte auf die Karte einfügen würde, welche keine Angaben über den jetzigen Status des Verkehres zeigen, muss noch eine weitere Funktion ausgeführt werden. Leaflet stellt hier die `onEacheFeature()`-Methode bereit, welche automatisch auf jedem Punkt aufgerufen wird.

```
1   onEachFeature(feature, layer) {
2       layer.on("click", async function () {
```

```
                const {
                    data
                } = await getData(feature.id);
                setTimeout(function () {
                    createApp(
                        Popup, {
                            data: data
                        }).mount(`#data`);
                }, 250);
            });
        },
```

Die Nutzung eines `onClick`-Ereignisses ermöglicht die Daten nur dann abzurufen, wann diese auch benötigt werden. So wird nur dann eine Anfrage an den Server gestartet, wenn der Benutzer mehr über diesen Messpunkt erfahren will.

Axios.js

Um die Messdaten vom Backend anzufordern, wird Axios verwendet. Zusätzlich wird mit Repositorien versucht, die Anfragen zu erleichtern und einfacher umzusetzen. Der folgende Code zeigt die Implementation der GeoJson-Anfrage mittels Axios.

```
import axios from "axios";

const baseDomain = "http://localhost:8080";
const baseURL = `${baseDomain}/api/v1`;

export default axios.create({
  baseURL,
});
```

Obiger Code stellt die Grundkonfiguration bereit, sollte sich der Server URL für das Backend ändern, so kann dieser hier zentral angepasst werden.

Folgender Code kümmert sich um die Anfrage der einzelnen Endpunkte der API. Genauer gesagt, holt sich dieser die GeoJson-Datei, welche alle Positionen von Messstationen beinhaltet.

```
import Repo from "./Repo";

```

```
const resource = "/nodes/geojson";
export default {
  get() {
    return Repo.get(`${resource}`);
  },
  getNode(nodeId) {
    return Repo.get(`${resource}/${nodeId}`);
  },
};
```

Der unten angeführte Code stellt die einzelnen Repositorien bereit. Dies ermöglicht das leichte Auswählen der einzelnen Endpunkte.

```
import GeoJsonRepo from "./GeoJsonRepo";
import NodeInfoRepo from "./NodeInfoRepo";

const repositories = {
  nodes: GeoJsonRepo,
  nodeInfo: NodeInfoRepo,
};
export const RepoFactory = {
  get: (name) => repositories[name],
};
```

3.2 Individuelle Zielsetzung des Teammitglieds Stefan Pisjak

3.2.1 Aufgabenstellung

3.2.1.1 Erkennen der Fahrzeuge

Die Aufgabe ist es, die einzelnen Fahrzeuge im ersten Schritt einmal zu erkennen. Dafür soll ein neuronales Netzwerk verwendet werden. Diese Erkennung soll bei jeglichen Wetter- & Umgebungsbedingungen zuverlässig funktionieren. Dabei ist es nicht wichtig, dass die Position des Objektes genau im Bild erkannt wird, sondern es reicht die Anzahl der Fahrzeuge im Bild zu erkennen.

3.2.1.2 Klassifizierung der Fahrzeuge

Nachdem das Erkennen der Fahrzeuge zuverlässig funktioniert, soll diese Erkennung um eine Klassifizierung erweitert werden. So sollen die Fahrzeuge dann in verschiedene Fahrzeugklassen gegliedert werden. Weiters soll es dann Kategorien für zumindest folgende Fahrzeugklassen geben:

- Pkws
- Lkws

Dies sollte im Idealfall alles lokal am Raspberry Pi bzw. einem stärkeren Einplatinencomputer ausgeführt werden. Zusätzlich könnte diesen Prozess ein Co-Prozessor (AI-Accelerator oder eine GPU), welcher die AI beschleunigen kann, unterstützen.

Falls es nicht möglich ist die Klassifizierung lokal durchzuführen, muss der Videostream so verarbeitet werden, dass dieser DSGVO-konform auf den Server übertragen werden kann. Dann würden die Daten am Server mit einer dedizierten GPU oder einem anderen Prozessor verarbeitet werden.

3.2.1.3 Sammeln weiterer Informationen

Zusätzlich sollen auch Statistiken dazu gesammelt werden, also wie z. B.:

- Wie lange steht ein Fahrzeug durchschnittlich an dieser Ampel?
- Wie lange braucht es, um die Ampel zu passieren?

Diese Daten werden dann gesammelt und an das Server-Backend geschickt.

3.2.1.4 Definieren einer API

Damit der Einplatinencomputer mit dem Server-Backend kommunizieren kann, muss vorher eine einheitliche API definiert werden.

Diese Daten werden dann mit LoRa bzw. LTE/5G übertragen.

3.2.2 Grundlagen und Methoden

3.2.2.1 Betriebssystem

Als Erstes musste das Betriebssystem für den Einplatinencomputer ausgesucht werden.

Als grobe Entscheidungsgrundlage kann einmal zwischen einer Linux-Distribution, Windows 10 on ARM und BSD unterschieden werden. Da der Einplatinencomputer nur den Videostream verarbeiten muss und deshalb auch keine grafische Umgebung benötigt wird, ist das Desktop-Environment egal, bzw. im Idealfall gibt es gar keines, um Speicherplatz zu sparen. Zusätzlich sind auch viele Frameworks, vor allem auf Debian-based Distributionen ausgelegt. Natürlich ist es auch möglich diese Frameworks auf ein anderes Betriebssystem mittels Portierung zu übertragen, jedoch ist es am einfachsten eine Debian-based Distribution zu nehmen.

Dadurch fallen die Optionen für sowohl Windows 10 on ARM als auch BSD weg und es muss eine (im besten Fall) Debian-based Distribution gesucht werden.

3.2.2.1.1 Linux-Distribution

Im Prinzip könnte eigentlich jede Linux-Distribution verwendet werden, jedoch wie bereits erwähnt, bietet es sich an, eine Debian-based Linux-Distribution zu verwenden. Beliebte Distributionen sind:

- Debian
- Ubuntu
- MX Linux
- Raspberry Pi OS (früher Raspbian)
- DietPi

Auch hier können die Distributionen nach dem gleichen Prinzip wie früher ausgesucht werden. Nachdem MX Linux eher eine Distribution für Desktop-PCs ist, kann diese auch gleich ausgeschlossen werden. Bei Ubuntu würde es eine eigene Variante ohne Desktop-Environment geben (Ubuntu Server). Jedoch ist Ubuntu größer als Debian und viele Features, welche Ubuntu bietet, werden gar nicht gebraucht.

Dadurch bleiben im Prinzip nur noch Debian selber, Raspberry Pi OS und DietPi übrig. Im Prinzip sind diese drei Optionen alle gute Betriebssysteme für unser Projekt, jedoch haben Raspberry Pi OS und DietPi noch einen Vorteil gegenüber Debian. Diese bieten nämlich out-of-the-box schon eine Möglichkeit einen Headless/remote install durchzuführen. Bei Debian ist es mit Modifikationen auch möglich (siehe philpagel/debian-headless (GitHub)), jedoch ist es bei Raspberry Pi OS und DietPi komfortabler, da es hier gleich funktioniert. Hier ist noch anzumerken, dass die Lite Variante von Raspberry Pi OS auch kein Desktop-Environment hat und dadurch kleiner ist.

Das einzige, was Raspberry Pi OS Lite und DietPi unterscheidet ist, dass DietPi extra darauf ausgelegt ist, schneller als Raspberry Pi OS Lite zu sein. Die offizielle Website [Kni21] gibt an, dass DietPi weniger RAM benötigt, weniger Prozesse laufen, das Image kleiner ist und noch viele weitere Vorteile gegenüber Raspberry Pi OS Lite bietet.

Als SBC wird bei unserer Diplomarbeit der Khadas VIM3 verwendet, da dieser einen eigenen Chip eingebaut hat, welcher neuronale Netzwerke beschleunigen kann [Kha21a]. Da dieser leider nicht alle Distributionen unterstützt, sondern nur spezielle, welche vom Hersteller bereitgestellt werden, muss Ubuntu schließlich doch verwendet werden.

3.2.2.2 Programmiersprache

Als Programmiersprache für die Fahrzeug-Erkennung- & Klassifizierung stehen im Prinzip viele Programmiersprachen offen. Zur gröberen Auswahl stehen folgende Programmiersprachen:

- Java/Kotlin
- C++
- Python

Diese Sprachen stehen zur Option, weil Java an der Schule unterrichtet wird und wir dadurch hier am meisten Erfahrung haben. Dann C++, weil es (wenn richtig programmiert) sehr schnell sein kann und Python, weil es relativ leicht zu programmieren ist und einen sehr großen Umfang an Programmbibliotheken bietet.

So gibt es zwar genug Machine-Learning/AI-Libraries für Java [Alh20], jedoch im Vergleich zu C++ bzw. Python ist die Anzahl überschaubar [Sah21]. Darüber hinaus gibt es für Python bekannte Libraries für AI/ML, wie PyTorch oder auch Tensorflow.

C++ Code kann performanter sein als vergleichbarer Python Code, wenn dieser richtig programmiert ist. Jedoch sind sowieso schon viel gebrauchte Funktionen/Libraries (numpy, ...) für Python in C/C++ implementiert und dadurch werden die Vorteile beider Welten kombiniert. Letztendlich wird bei der Diplomarbeit Python verwendet, aufgrund der zuvor genannten Vorteile.

3.2.2.3 Neuronale Netzwerke zum Erkennen von Objekten

Bevor auf einzelne neuronale Netzwerke eingegangen werden kann, ist es wichtig, dass diese untereinander vergleichen werden.

3.2.2.3.1 Was sind neuronale Netzwerke?

Neuronale Netzwerke sind, wie der Name schon sagt, eine Ansammlung von vielen (künstlichen) Neuronen. Ein künstliches Neuron kann sich so vorgestellt werden, dass es eine gewisse Anzahl an Ein- und Ausgängen hat. Jetzt hat das Neuron verschiedene Möglichkeiten, was es mit diesen Daten machen kann. So könnte es im Prinzip alle Werte zusammenzählen und dann durch eine beliebige Zahl dividieren oder auch einfach nur die Differenz zwischen dem größten und kleinsten Wert bilden (Das passiert aber so nicht, mehr dazu später.). Da so ein künstliches Neuron nicht viel alleine ausrichten kann, wird es in ein Netzwerk mit vielen anderen Neuronen gepackt. Hier ist eine beispielhafte Anordnung von Neuronen:

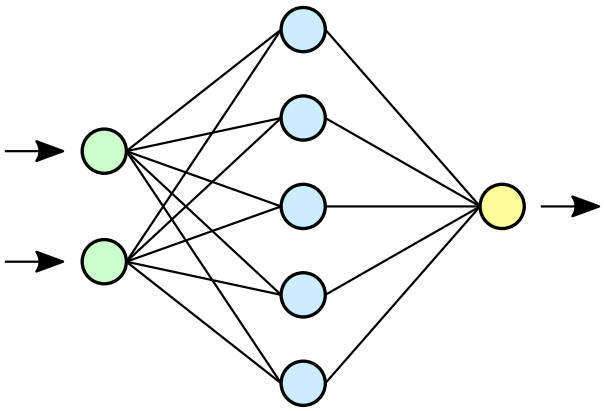


Abbildung 3.8: Einfaches neuronales Netzwerk [Dak06]

Die einzelnen Eingänge des Neurons haben verschiedene Gewichtungen, das bedeutet, dass ein bestimmtes anderes Neuron mehr oder weniger Einfluss auf das aktuelle Neuron hat. Diese Werte werden zusammengezählt und anhand einer Aktivierungsfunktion entscheidet das Neuron, ob es nun aktiviert ist oder nicht. Zusätzlich kann noch ein Schwellenwert (englisch bias) hinzuaddiert werden, welcher den Aktivierungspunkt verschiebt.

$$output = f(x) = \begin{cases} 1\ if\ \sum_{i=1}^{m} w_i \times x_i + b_i \geq 0 \\ 0\ if\ \sum_{i=1}^{m} w_i \times x_i + b_i < 0 \end{cases} \tag{3.1}$$

[IBM20]

In der Abbildung 3.8 kann die Zusammensetzung eines neuronalen Netzwerkes erkannt werden. Die zwei grünen Neuronen sind die Input-Nodes. Diese könnten bei einem Video dem roten und grünen Subpixel-Anteil eines Pixels eines Frames entsprechen. Danach gibt es eine gewisse Anzahl an sogenannten Hidden-Layer. In diesem Fall ist es nur ein Hidden-Layer mit fünf künstlichen Neuronen. Danach gibt es noch ein (gelbes) Neuron, welcher dem Output Layer entspricht. In diesem Fall ist es nur ein einzelnes Neuron. Dieses kann in diesem Fall nur einen einzelnen Wert angeben. Aber bei größeren neuronalen Netzwerken gibt es davon mehrere und dadurch kann die Klasse, die Position und die Wahrscheinlichkeit, dass sich an einer gewissen Stelle ein Objekt befindet, angegeben werden.

Auch die Anzahl der Input-Nodes muss gesteigert werden, um größere Bildbereiche abdecken zu können. Das Gleiche gilt auch für die Hidden-Layer, von denen es mehrere gibt. Im Hidden-Layer passiert auch das eigentliche Arbeiten des neuronalen Netzwerkes.

In folgender (Abbildung 3.9) wird das Ganze besser visualisiert:

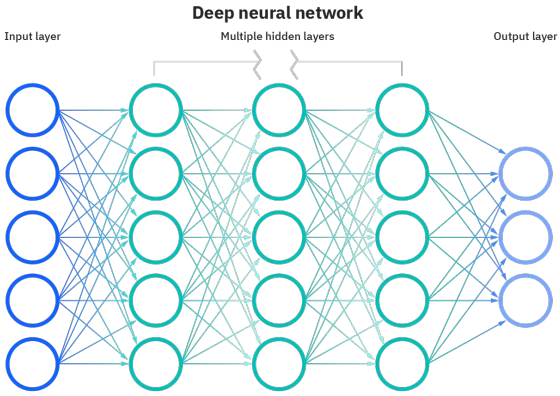


Abbildung 3.9: Tiefes neuronales Netzwerk. [IBM20]

Hier befinden sich fünf Input-Nodes, ein unbestimmt großer Hidden-Layer und drei Output-Nodes.

Für unsere Diplomarbeit bedeutet dies, dass am Eingang die einzelnen Frames des Videostream der Kamera anliegen und diese vom neuronalen Netzwerk verarbeitet wird. Am Ausgang wird dann angezeigt, wo sich zu welcher Wahrscheinlichkeit welches Fahrzeug befindet.

3.2.2.3.2 Wie lernt das neuronale Netzwerk?

Damit die Gewichtungen und den Bias bestimmt werden kann, muss das Netzwerk dies irgendwie herausfinden. Das macht ein neuronales Netzwerk normalerweise über sogenannte Backpropagation. Hier wird das Netzwerk vom Ausgang mit Daten versorgt. So werden hier gelabelte Daten (mehr dazu später) am Netzwerk angelegt und dadurch weiß das Netzwerk, was bei einem bestimmten Eingang ausgegeben werden soll.

Dies wird *supervised learning* genannt, da der Eingang und Ausgang schon bekannt sind [Art21]. Aber es gibt auch andere Varianten, wie ein neuronales Netzwerk lernen kann.

Damit nun das Netzwerk lernen kann, werden sogenannte gelabelte Daten bzw. Bilder benötigt. Viele gelabelte Bilder zusammen werden Datensatz genannt.

3.2.2.3.3 Was ist ein Datensatz?

Ein Datensatz sind Bilder, welche gelabelt sind. Das sind Bilder von Objekten, wobei diese dann von einem Menschen eingerahmt und klassifiziert wurden. Gelabelte Bilder sind Bilder von Objekten mit einer extra Datei, welche angibt, in welchem Bereich des Bildes sich Objekte befinden und welche Objekte sich dort befinden. Dieser Prozess ist sehr anstrengend und muss von einem Menschen gemacht werden. Jedoch gibt es schon fertige Datensätze, welche verwendet werden können. Die relevantesten Datensätze sind:

- COCO (Common Objects in Context) von Microsoft
- PASCAL VOC
- OID (Open Images Dataset) von Google
- ImageNet

Diese Datensätze sind relevant, weil mit diesen viele (Objekt-Erkennungs-) Frameworks trainiert und ausgetestet werden.

3.2.2.3.4 Auswahl des neuronalen Netzwerkes

Dies ist einer der elementarsten Teile meiner individuellen Zielsetzung bzw. auch der ganzen Diplomarbeit. Bei der Auswahl des neuronalen Netzwerkes gibt es viele Unterscheidungspunkte. Abbildung 3.10 zeigt eine Übersicht, wie ein neuronales Netzwerk aufgebaut sein kann.

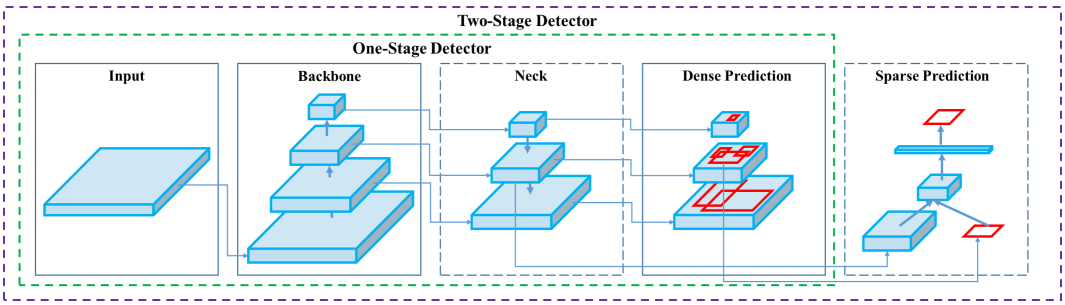


Abbildung 3.10: Object detector - Foto aus dem YOLOv4 Paper [BWL20]

Am Eingang befindet sich ein Bild bzw. auch nur ein Ausschnitt aus einem Bild. Hier ist noch anzumerken, dass hier, bei der Objekt-Erkennung meist nicht einfach ein sogenanntes *Feed-Forward* neuronales Netzwerk, welches bei Absatz 3.2.2.3.1 beschrieben wurde, ausreicht.

So wird oft ein sogenanntes Convolutional Neural Network, kurz CNN, verwendet (Aber nicht bei allen neuronalen Netzwerken zu Objekt). Dieses unterscheidet sich zum „normalen“ *Feed-Forward*

neuronalen Netzwerk, da hier die künstlichen Neuronen nicht linear untereinander angeordnet werden (Siehe Abbildung 3.9), sondern in einem 2D-Array (bzw. eigentlich einem 3D, mehr dazu später) angeordnet werden. Visuell würde dies Abbildung 3.11 entsprechen.

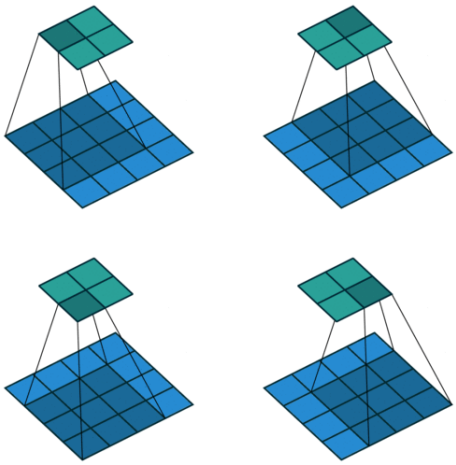


Abbildung 3.11: No padding, no strides [DV16]

Das blaue, untere 2D-Array entspricht hierbei dem Input-Image. In diesem Fall besteht das Bild nur aus 4x4 Pixel. Anstatt, dass die Pixel in einer Linie nebeneinander angeordnet werden, werden sie wie hier zu sehen (Abbildung 3.11) angeordnet. Dadurch können räumliche Abhängigkeiten besser vom neuronalen Netzwerk verarbeitet werden [Sah18].

Das grüne, obere 2D-Array ist der Kernel (wird auch oft Filter genannt oder mit K abgekürzt) und entspricht dem ersten Convolutional Layer. Dies ist, um es mit einem Feed-Forward neuronalen Netzwerk zu vergleichen, der erste Hidden-Layer. Wie auch beim Feed-Forward neuronalen Netzwerk kann es hier auch wieder mehrere Convolutional Layer geben. Außerdem muss dieser Aufbau dreimal passieren, für die roten, blauen und grünen Subpixel. Visuell würde dies Abbildung 3.12 entsprechen.

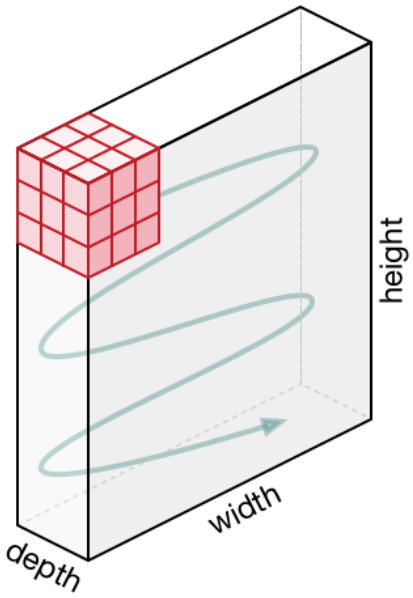


Abbildung 3.12: Movement of the Kernel [Sah18]

Der Kernel

Der Kernel funktioniert so, dass dieser gewisse Gewichtungen hat, dabei sind unterschiedliche Gewichtungen unterschiedlich gut beim Erkennen verschiedener Merkmale. So ist z. B. der Kernel, welcher bei Abbildung 3.13 verwendet wird, auf die Erkennung vertikaler Kanten spezialisiert.

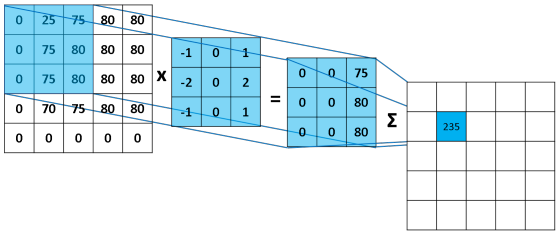


Abbildung 3.13: Verwenden des Kernels [Rob18]

Erwähnenswert ist noch, dass es verschieden Arten von Kernel gibt, so ist die Abbildung 3.11 ein Beispiel für einen Kernel ohne Padding und ohne Strides. Wenn ein Padding zu einem Kernel hinzugefügt wird, dann bedeutet das, dass um das Input-Image eine gewisse Anzahl an „leere“ Pixel hinzugefügt werden. Am besten kann dies auch anhand einer Visualisierung erkannt werden (Abbildung 3.14).

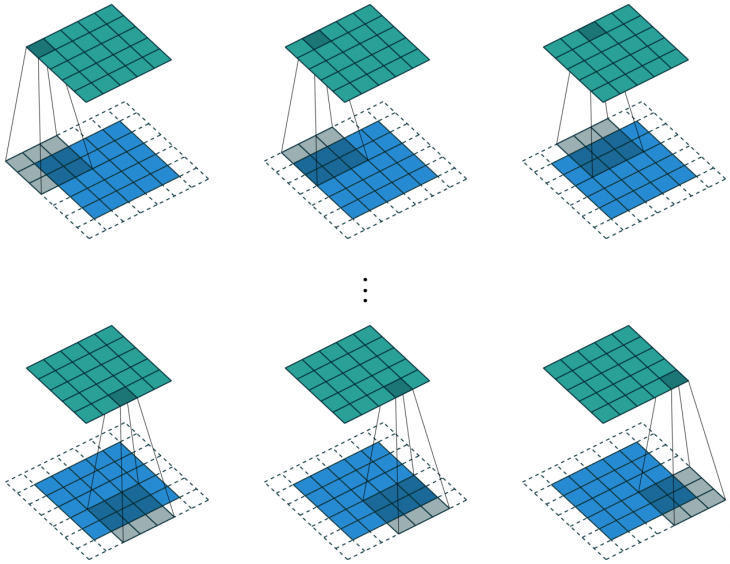


Abbildung 3.14: Padding, no strides [DV16]

Hier ist aufgrund des Paddings der Kernel auch genau gleich groß, wie das eigentliche Bild.

Wird nun noch mehr Padding hinzugefügt, sodass die äußerste Node nur noch einen „echten“ Pixel hat und die anderen Padding sind, dann wird dies Full Padding genannt, siehe Abbildung 3.15.

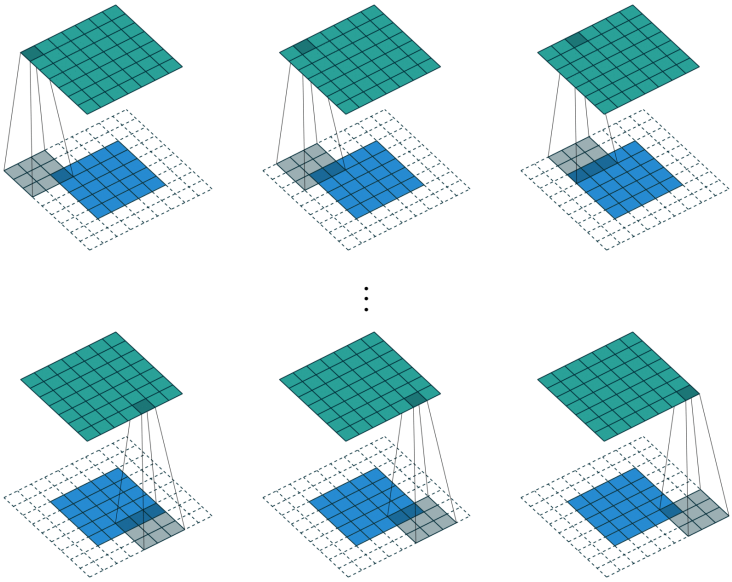


Abbildung 3.15: Full Padding, no strides [DV16]

Dabei ist es noch nennenswert, dass hier der Convolution Layer größer ist als der eigentliche Input-Layer. Schließlich gibt es dann noch Strides, welche besagen, ob nun eine Reihe & Zeile ausgelassen werden soll oder nicht. In diesem Fall Abbildung 3.16 wird das Input-Image nicht einfach auf einen 3x3 Kernel gebracht, sondern sogar auf einen 2x2 Kernel.

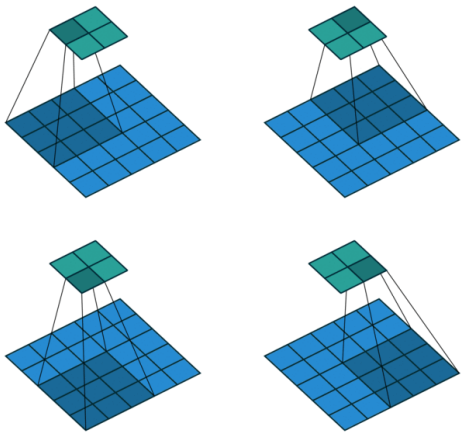


Abbildung 3.16: No padding, strides [DV16]

Schließlich kann das Padding mit den Strides auch kombiniert werden. In diesem Fall Abbildung 3.17 „gleichen sich Padding und Strides aus“ und der Kernel ist gleich groß, wie wenn weder Padding noch Strides verwendet werden würden.

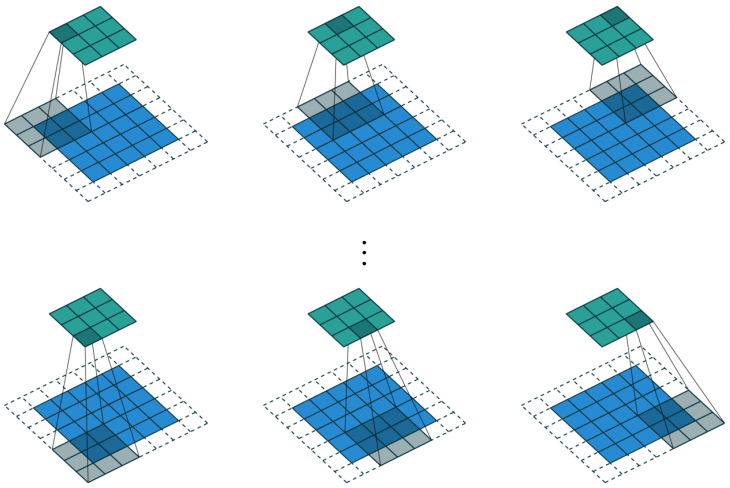


Abbildung 3.17: Padding, strides [DV16]

Es gibt aber auch viele andere Kombinationen und Möglichkeiten, wie z. B. Dilated Convolution, wo dann nicht alle Werte vom Input-Image genommen werden, sondern nur in einem gewissen Muster. Außerdem kann das Padding auch „odd sein“, wodurch sich dann das Auslassen einer Reihe nicht mehr symmetrisch ausgeht. Hierdurch wird auf einer Seite das Padding genommen und auf der Gegenüberliegenden nicht. Diese Bilder und weitere Information können hier gefunden werden [DV16].

Pooling Layer

Auf den Convolutional Layer folgt dann der Pooling Layer. Dieser ist auch wieder zuständig für die Erkennung bestimmter Merkmale, welche aber unabhängig von der Position und Rotation sind [Sah18]. Er wird verwendet um die Höhe und Breite des Layers zu reduzieren. So kann mit dem Wert 2 für

Strides die Bildgröße halbiert werden [Sta21]. Im Prinzip ist der Pooling Layer ganz ähnlich aufgebaut wie der Convolutional Layer, jedoch hat dieser keine fixe Matrix, welche bestimmt, welche Daten in den nächsten Layer kommen, sondern hierbei hat gibt es zwei Varianten, wie das neuronale Netzwerk vorgehen kann.

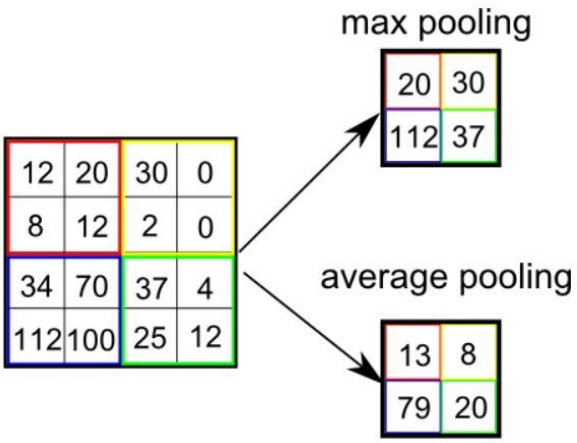


Abbildung 3.18: Max und Average Pooling. Foto von [Sah18]

Einmal kann Max Pooling verwendet werden, was einfach den Maximalwert aus den ausgewählten Neuronen nimmt. Zusätzlich hat Max Pooling noch den Vorteil, dass eine Art Rauschreduktion [Sah18] durchgeführt wird, weil immer nur der Maximalwert genommen wird.

Average Pooling bildet, wie der Name schon verrät, den Mittelwert aus allen Neuronen (Es werden alle Werte zusammengezählt und durch die Anzahl dividiert.). In diesem Fall z. B. für das obere, rechte, rote Rechteck:

$$(12 + 20 + 8 + 12)/4 = 13$$

Meist wird Max Pooling bevorzugt, weil es besser Features, wie Kanten, erkennt, da nicht einfach ein Mittelwert gebildet wird, sondern immer der größte Wert genommen wird. Zusätzlich wirkt es wie ein Filter, welcher das Rauschen reduzieren kann. [Sah18] [Sta21]

Den Unterschied zwischen Max Pooling und Average Pooling kann auch anhand eines Bildes (in diesem Fall des Commean Logos) sichtbar gemacht werden:

Abbildung 3.19: Average Pooling (128x128x3 → 64x64x3)

Abbildung 3.20: Max Pooling (128x128x3 → 64x64x3)

Hier wird ein Bild mit einer Auflösung von 128x128 Pixel (mit drei Channel [RGB]) auf 64x64 Pixel herunterskaliert. Dabei tritt beim Average Pooling kein, sogenannter, Treppeneffekt auf. Beim Average Pooling werden vier Pixel immer zusammengefasst und der Durchschnitt davon genommen. Dadurch sieht das herunterskalierte Bild noch immer relativ gut aus, da immer die Nachbarpixel berücksichtigt wurden. Dies ist vorteilhaft, wenn ein Bild von einer zu großen Auflösung auf eine kleine Auflösung skaliert werden soll, ohne das es sofort auffällt. Jedoch für das neuronale Netzwerk ist es von Nachteil, da hier die Kanten eben *nicht* auffallen.

Im Vergleich hat das Bild, welches mit Max Pooling herunterskaliert wurde, eine klar sichtbare Kante (Es hat zwar noch immer antialiast Kanten, aber dies liegt daran, dass bereits das Originalbild antialiast war.). Wenn die Auflösung noch stärker verringert wird, dann ist der Effekt noch deutlicher, wie es Abbildung 3.22 beweist.

Abbildung 3.21: Average Pooling (2048x2048x3 → 64x64x3)

Abbildung 3.22: Max Pooling (2048x2048x3 → 64x64x3)

Dadurch, dass die Kanten und andere Features am Bild bei Max Pooling viel besser herausstechen, wird dies bei neuronalen Netzwerken auch bevorzugt.

Backbone

Das Backbone besteht aus einer Kombination des Pooling Layers Abschnitt 3.2.2.3.4 und dem Convolution Layer Abschnitt 3.2.2.3.4. Diese zwei Layer bauen dann gemeinsam das Backbone auf und

ermöglichen dem neuronalen Netzwerk erst das Erkennen von Objekten.

Wenn nun das neuronale Netzwerk ein Objekt erkennen soll, erkennt es als erstes gleich das ganze Fahrzeug und geht dann immer weiter ins Detail. Das liegt daran, weil beim Trainieren über Backpropagation das neuronale Netzwerk von „hinten“ alles lernen muss; also sucht es zuerst passende Kanten, welche es dann in Strukturen, wie Kreise, Rechtecke, etc., danach in Windschutzscheiben, Lichter und dann final in Fahrzeuge gruppieren kann. [Rob18]

Für dieses Backbone kann z. B. ResNet [He+15], die Weiterentwicklung ResNeXt [Xie+16] oder DetNet [Li+18] verwendet werden. [BWL20] Aber es gibt hier auch viele andere Backbones, welche jeweils immer entweder auf etwas spezialisiert sind oder einfach neue state-of-the-art Algorithmen/-Lösungen implementieren. Challenges dabei sind, dass das Backbone sowohl kleinste Details, als auch große prominente Objekte erkennt.

Im Prinzip können damit schon Objekte erkannt werden, jedoch gibt es auch Erweiterungen, um die Erkennung zu verbessern. So gibt es nach dem Backbone noch ein sogenanntes *Neck* (Siehe Abbildung 3.10). Auch wieder für dieses Neck gibt es verschiedenste Möglichkeiten, wie z. B. Feature Pyramid Network (kurz FPN) [Lin+16] [BWL20] oder eine Weiterentwicklung davon PANet [BWL20] [Liu+18].

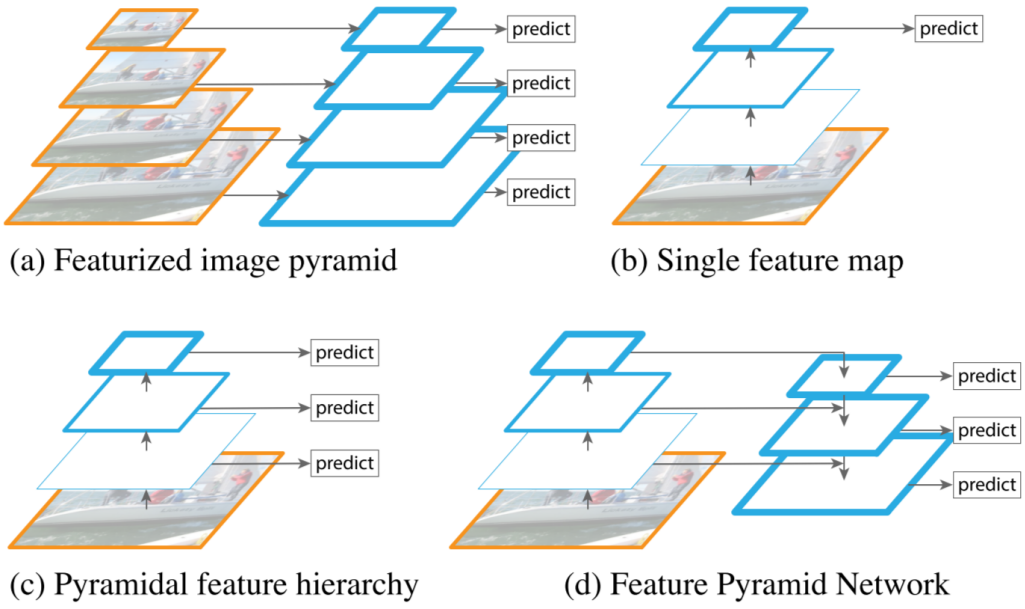


Abbildung 3.23: Mögliche Architekturen. Foto von [Lin+16]

Hier werden verschiedene Architekturen angezeigt (Die linke „Pyramide“ entspricht dabei dem Backbone.).

Ohne dieses Neck würde Architektur b verwendet werden. Dies war auch lange Zeit der Standard für CNNs Convolutional Neural Network, da andere Strukturen zu viel Performance benötigten und damit das ganze neuronale Netzwerk verlangsamten [Lin+16]. Deshalb setzen auch neuronale Netzwerke, wie Fast und Faster R-CNN standardmäßig nicht auf diese featurized Bilder Pyramiden.

Option (c) wird von unter anderem auch Single Shot MultiBox Detector (kurz SSD) verwendet. Anzumerken ist noch, dass SSD ein Detektor ist und somit den Head (Dense Prediction) auch noch

dabei hat [BWL20]. Jedoch wird hier immer nur von jedem neuen Layer die Prediction gemacht und dabei werden die Ergebnisse der anderen Layer nicht verwendet. [Lin+16] [Liu+15]

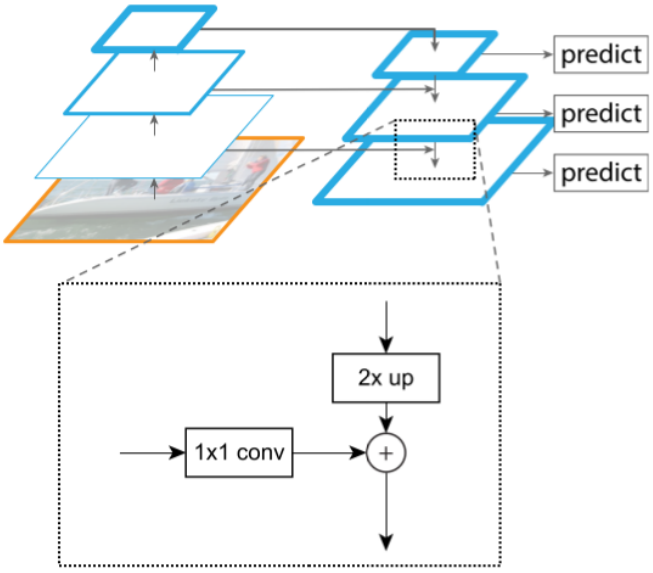


Abbildung 3.24: Architektur von FPN. Foto von [Lin+16]

Bei FPN wird hier jedoch sowohl vom höheren Layer die Prediction genommen und diese mit den Ergebnissen vom Backbone kombiniert. Bei FPN wird der linke Schritt (Backbone im YOLO-Paper [BWL20]) in diesem Bild Bottom-Up Pathway und der rechte Teil Top-Down Pathway. Im Top-Down Pathway werden die einzelnen Layer, sogenannte Maps, immer mit einem Faktor von 2 hochskaliert [Lin+16]. Dadurch können die Ergebnisse von den Bildern mit höherer Auflösung mit denen kleinerer Auflösung kombiniert werden, was zu einer besseren Erkennung führt [Lin+16].

Nach dem Backbone und dem Neck kommt jetzt der Head (Siehe Abbildung Abbildung 3.10). Dieser besteht entweder aus einer Dense Prediction und einer Sparse Prediction und wird daher auch Two-Stage Detector genannt. Beispiele dafür sind Faster R-CNN (Region based CNNs) oder R-FCN (Region-based Fully Convolutional Network) [BWL20].

Jedoch kann es auch sein, dass das neuronale Netzwerk nur eine Dense Prediction hat. Dann wird dieses neuronale Netzwerk One-Stage Detector genannt. Prominente Beispiele dafür sind YOLO (You Only Look Once), RPN (Region Proposal Network) oder SSD. [BWL20]

3.2.3 Realisierung

3.2.3.1 Inbetriebnahme des VIM3

Als Erstes muss das Betriebssystem installiert werden. Hierfür gibt es zwei Möglichkeiten am VIM 3, entweder das OS wird auf der eMMC oder auf einer microSD-Karte installiert.

3.2.3.1.1 Installieren des OS auf die eMMC

Um das Betriebssystem auf die eMMC zu installieren, wird ein zusätzliches Tool, welches vom Hersteller (Khadas) zur Verfügung gestellt wird, benötigt. Unter Linux wird ein CLI-Tool zur Verfügung gestellt und unter Windows eines mit einer GUI. [Kha21b]

Dieses muss installiert werden und dann sollte es möglich sein, dieses Tool über das Startmenü zu starten:

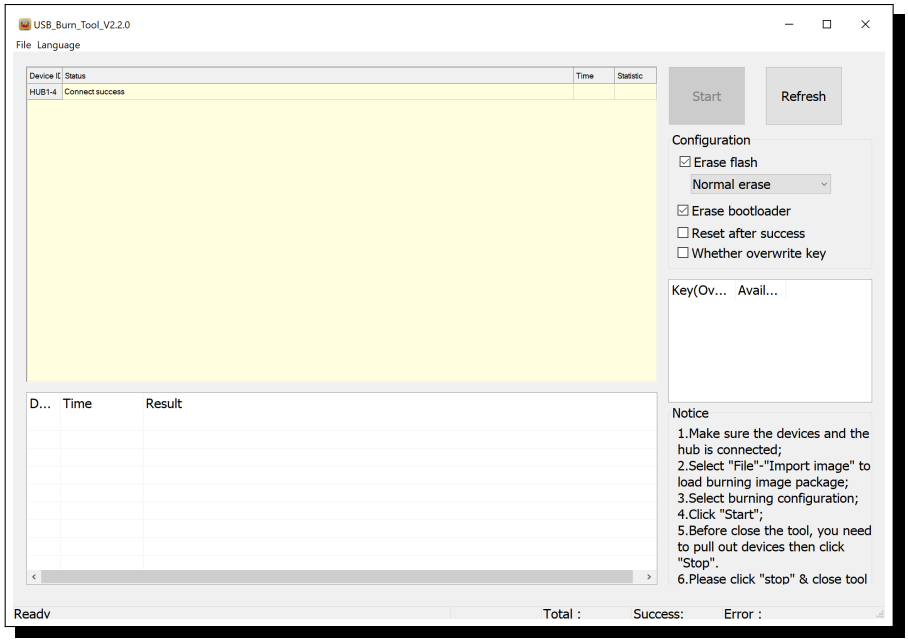


Abbildung 3.25: Öffnen des Tools

Nun muss der VIM3 mit dem Computer per USB-C Kabel verbunden werden. Bei erfolgreicher Verbindung sollte der VIM3 nun im Programm aufgelistet sein (Siehe Abbildung 3.25).

Jetzt muss das richtige Image ausgewählt werden, welches auf die eMMC geflasht werden soll. Dieses Image kann mit einem wiederum anderen Tool generiert werden, oder es kann einfach ein fertiges Image von der Website heruntergeladen werden.

Dieses Image kann einfach über *File* $\rightarrow$ *Import Image* ausgewählt werden:

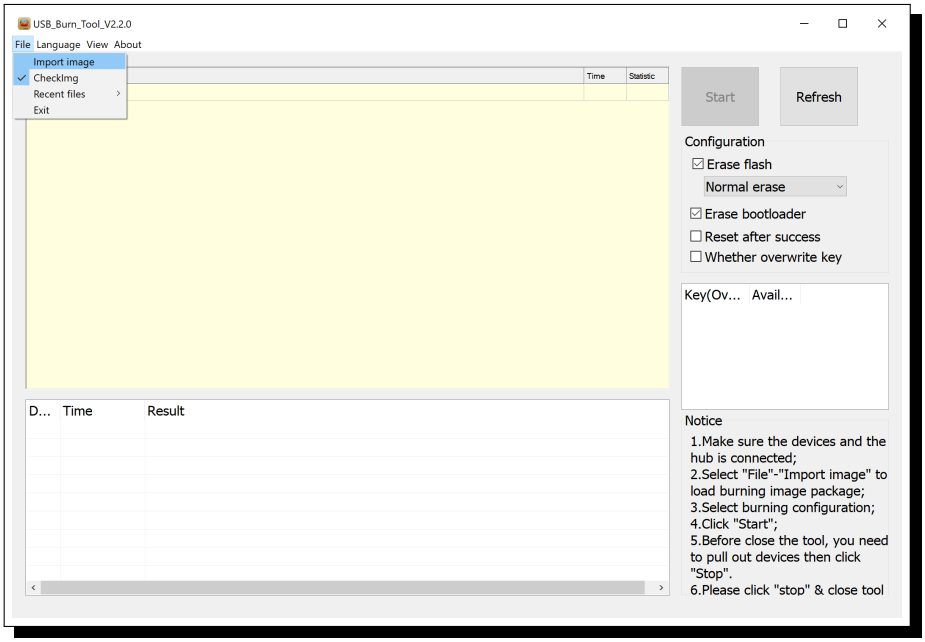


Abbildung 3.26: Auswählen des Images

Daraufhin wird das Image automatisch überprüft:

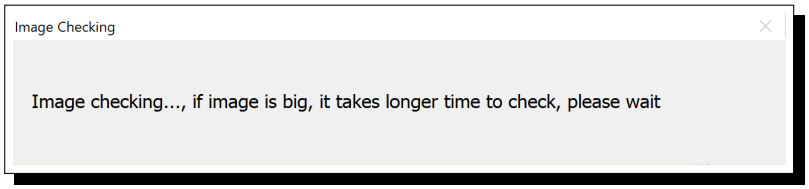


Abbildung 3.27: Auswählen des Images

Um den Flash-Vorgang zu starten, muss dann nur noch der *Start*-Knopf gedrückt werden.

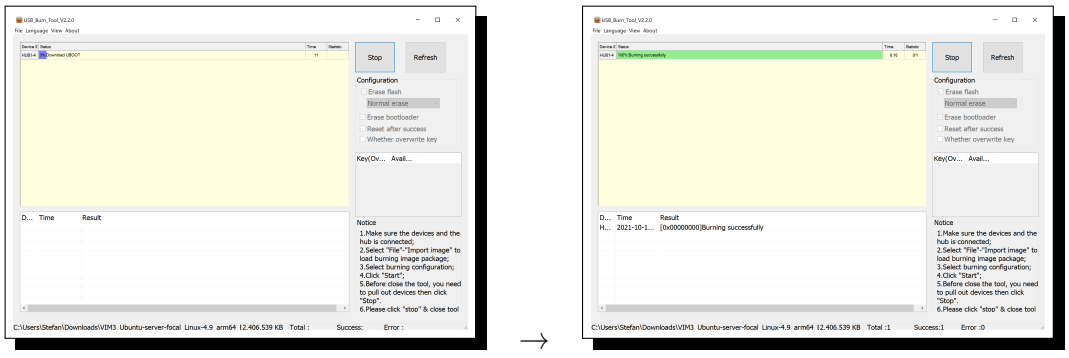


Abbildung 3.28: Flash-Vorgang

Nun kann der VIM3 vom Computer getrennt und verwendet werden.

3.2.3.1.2 VNC-Server einrichten

Damit nun am VIM3 etwas entwickelt werden kann, würde im Prinzip nur eine SSH-Verbindung ausreichen, damit der Code darauf übertragen werden kann. Jedoch da die Erkennung von Fahrzeugen eine visuelle Aufgabe ist, ist es wichtig, dass das Ergebnis der Erkennung gleichzeitig mit dem Videostream angezeigt wird. Dies wird nur für die Entwicklung benötigt und später soll dies alles auf einem headless System laufen. Anstatt jedes Mal einen Monitor anzuschließen ist es komfortabler, sich einfach über einen Laptop mit VNC zu dem VIM zu verbinden.

Als VNC-Server kann im Prinzip ein beliebiger VNC-Server ausgewählt werden. In diesem Fall wird der TightVNC (Server) benutzt.

Dieser kann über folgendes Kommando mit dem Paketmanagement-System `apt` installiert werden.

```
1  # apt install tightvncserver
```

Beim Verbinden mit dem VNC-Server wird automatisch eine bestimmte Datei, welche die grafische Umgebung initialisieren soll, ausgeführt (`~/.vnc/xstartup`).

Falls bei der Distribution standardmäßig keine Desktop-Umgebung dabei ist, muss auch diese noch installiert werden. Hier gibt es verschiedenste Desktop-Umgebungen, wobei manche rechenaufwändiger sind und andere den Fokus eher auf eine schlanke Umgebung ohne viel Ressourcenverbrauch setzen.

Hier entscheiden wir uns für LXDE [LXD22], was eine schlanke Desktop-Umgebung ist, um die Performance für das eigentliche Programm zu sparen. Auch LXDE kann wieder über `apt` installiert werden.

```
1  # apt install lxde
```

Nun muss nur noch die `xstartup`-Datei so bearbeitet werden, dass diese automatisch die Desktop-Umgebung startet.

Listing 1 `xstartup`-Datei zum Initialisieren der Desktop-Umgebung

```
1 #!/bin/sh
2
3 unset SESSION_MANAGER
4 unset DBUS_SESSION_BUS_ADDRESS
5
6 [ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
7
8 exec startlxde
```

Hier werden zuerst die zwei Umgebungsvariablen gelöscht und dann die `Xresources`, falls vorhanden, geladen. Zu guter Letzt wird dann die Desktop-Umgebung mit `exec startlxde` gestartet.

Schließlich muss der VNC-Server auch noch gestartet werden, damit eine Verbindung zu diesem möglich ist. Da dieser VNC-Server nach jedem Neustart gestartet werden muss, kann dieses Kommando auch in ein Skript gepackt werden, damit nicht immer wieder erneut das Kommando ausgeführt werden muss. Zusätzlich kann noch eine beliebige Auflösung übergeben werden, mit welcher der VNC-Server dann läuft.

Listing 2 Skript zum Starten des VNC-Servers

```
1 #!/bin/bash
2
3 vncserver -geometry 1860x1020
```

Listing 3 Ausgabe des Skripts

```
$ ./vncAtHome.sh

New 'X' desktop is Khadas:1

Starting applications specified in /home/khadas/.vnc/xstartup
Log file is /home/khadas/.vnc/Khadas:1.log
```

VNC-Verbindung herstellen

Nach dem Ausführen des Skripts (Listing 2) muss jetzt nur noch die Verbindung vom Client hergestellt werden. Dies kann auch wieder über einen beliebigen Client erfolgen, aber in diesem Fall wird TigerVNC verwendet.

Wenn das Programm gestartet wird, muss einfach nur die IP eingeben werden, gefolgt von einem Doppelpunkt. Der Doppelpunkt gibt an, welche Instanz genommen werden soll. So ist es möglich, gleichzeitig mehrere VNC-Instanzen für verschiedene Benutzer*innen offen zu haben. Weiters ist es möglich, eine weitere Instanz zu starten, indem das Skript einfach öfters ausgeführt wird und dann bekommt jede neue Instanz eine neue Ziffer.

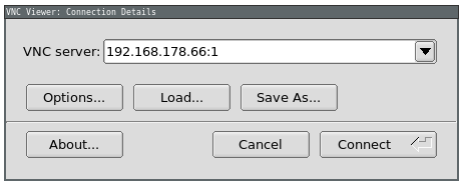


Abbildung 3.29: Eintragen der IP des VNC-Servers

Danach muss, wenn ein Passwort vergeben wurde, dieses eingegeben werden.

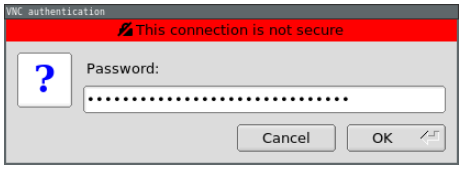


Abbildung 3.30: Eingeben des Passworts für den VNC-Server

Schließlich sollte dann die Verbindung hergestellt werden und es wird der Desktop angezeigt.

Abbildung 3.31: Erfolgreiche VNC-Verbindung

3.2.3.2 SSHFS Verbindung herstellen

SSHFS ist die Abkürzung für SSH Filesystem und ermöglicht den Austausch von Daten über SSH. Um SSHFS unter Windows zu benutzen, muss es zuerst installiert werden. Dafür kann der aktuellste Build von GitHub (billziss-gh/sshfs-win[Zis21]) installiert werden oder der Package Manager Chocolatey [Web21] kann verwendet werden.

3.2.3.2.1 Herstellung der Verbindung unter Windows

Um die Verbindung herzustellen, kann entweder die CLI oder GUI verwendet werden.

Verbindung über die GUI herstellen

Um die Verbindung über die GUI herzustellen, muss zuerst der Windows-Explorer geöffnet werden und dann zu „Dieser PC“ navigiert werden.

Danach kann das Netzlaufwerk eingebunden werden, indem unter dem Menüband unter dem Tab „Computer“ „Netzlaufwerk verbinden“ ausgewählt wird.

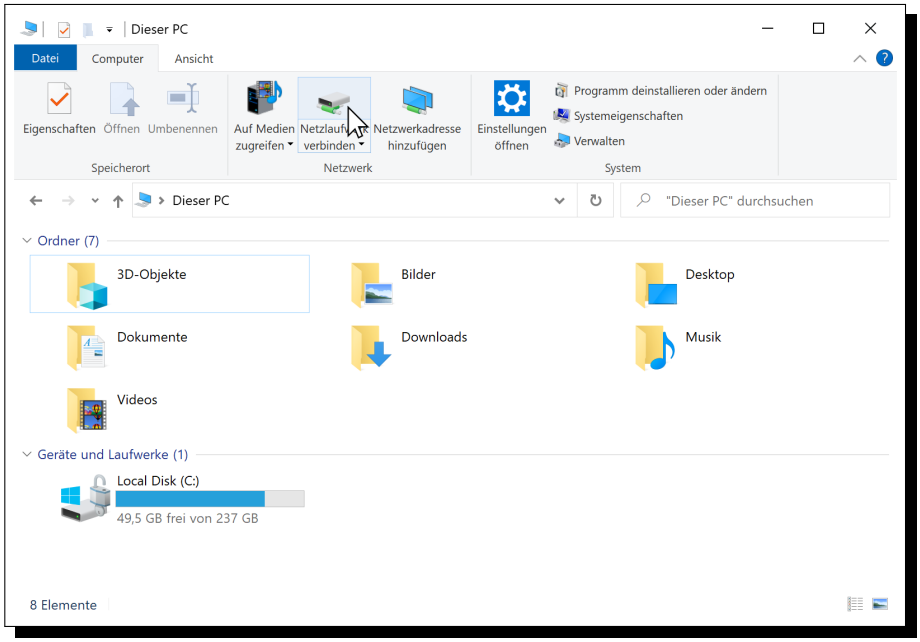


Abbildung 3.32: Öffnen des Fensters um das Netzlaufwerk zu verbinden

Darauf öffnet sich ein neues Fenster. Der Pfad muss einem der folgenden Syntaxen [Zis21] entsprechen:

```
1  \\sshfs\[LOCUSER=]\REMUSER@HOST[!PORT][\PATH]
2  \\sshfs.r\[LOCUSER=]REMUSER@HOST[!PORT][\PATH]
3  \\sshfs.k\[LOCUSER=]REMUSER@HOST[!PORT][\PATH]
4  \\sshfs.kr\[LOCUSER=]REMUSER@HOST[!PORT][\PATH]
```

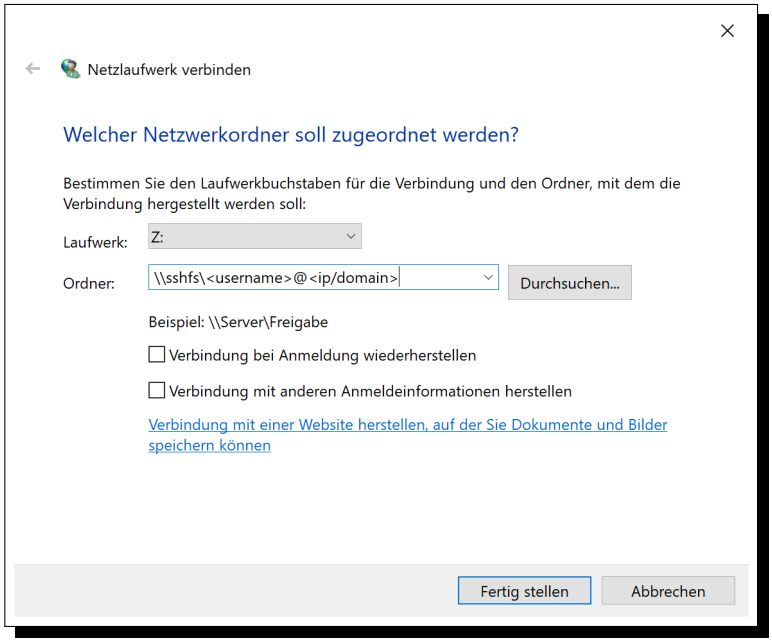


Abbildung 3.33: Konfigurieren der Parameter für das Netzlaufwerk

Mehr dazu auf GitHub (https://github.com/billziss-gh/sshfs-win).

Nun sollte sich ein Pop-up öffnen, in welchem die Anmeldedaten eingetragen werden sollen:

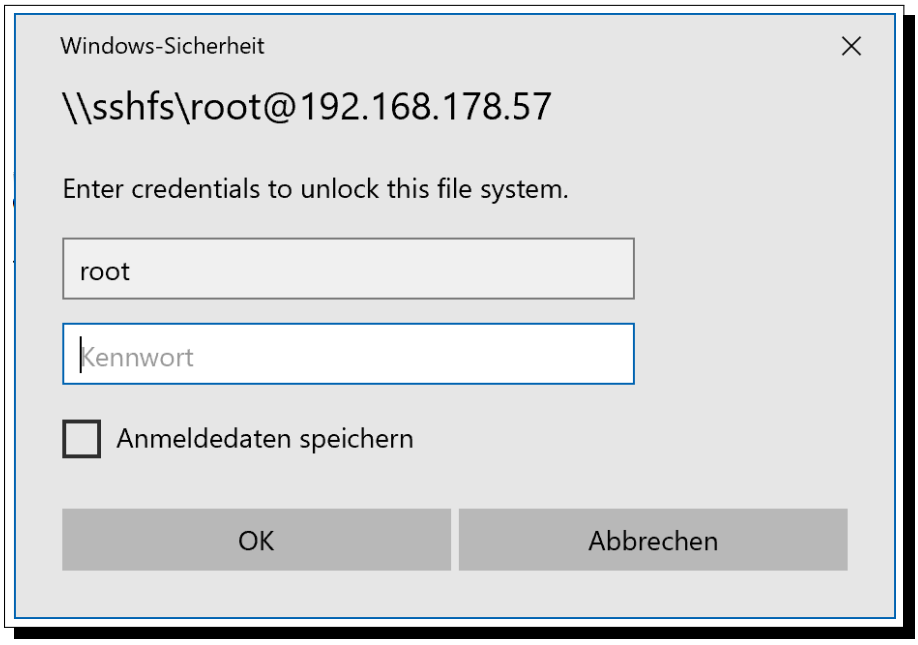


Abbildung 3.34: Anmeldefenster

Nach erfolgreicher Anmeldung sollte das Netzlaufwerk wieder unter „Dieser PC“ erscheinen:

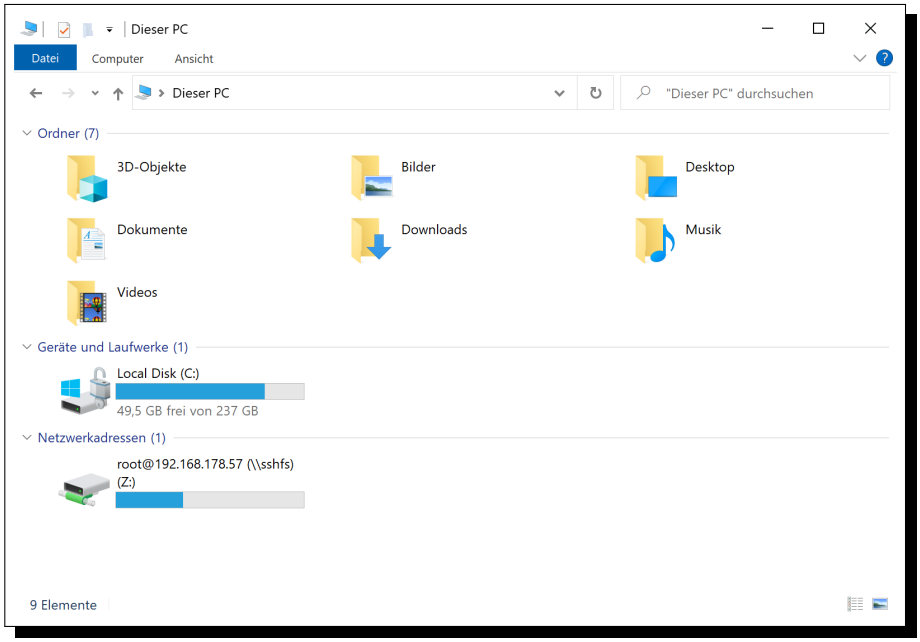


Abbildung 3.35: Erfolgreiche Einbindung des Netzlaufwerks

Über die CLI

Es ist auch möglich die Verbindung über die PowerShell bzw. CMD herzustellen.

```
1  PS> net use Z: \\sshfs\root@192.168.178.57
```

3.2.3.2.2 Troubleshooting

Es kann auch sein, dass die Verbindung nicht hergestellt werden kann. In der GUI wird dabei kein wirklicher Errorcode angezeigt, nur dass die Verbindung nicht hergestellt werden konnte. In der CLI wird ein Error ausgegeben.

Systemfehler 67

Um diesen Fehler zu beheben, kann der Fehlerlösung auf GitHub gefolgt werden: sshfs-win#279

Es muss im Prinzip nur der Dienst „WinFsp.Launcher“ über den Task-Manager gestartet werden:

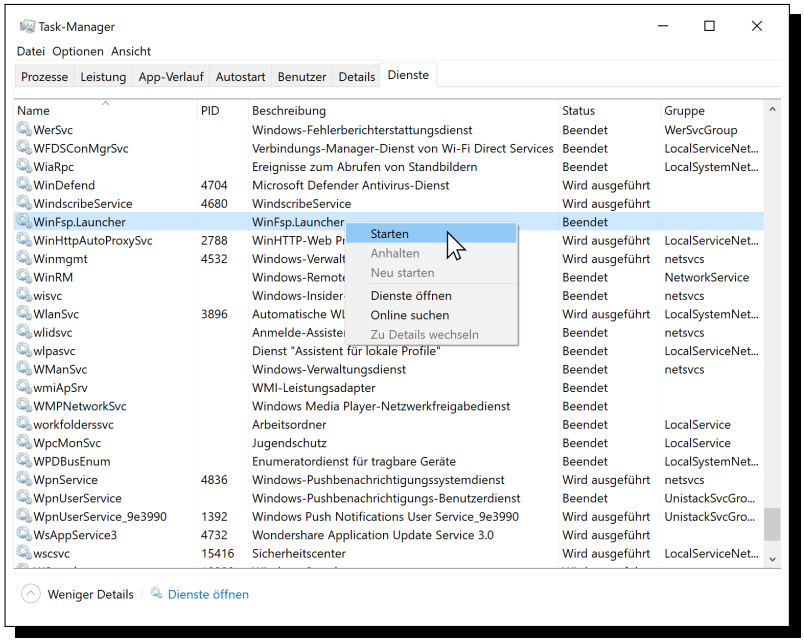


Abbildung 3.36: Windows File System Proxy Service starten

3.2.3.2.3 Herstellung der Verbindung unter Linux

Dies ist relativ ähnlich, wie unter Windows per CLI Abschnitt 3.2.3.2.1. Jedoch muss unter Linux zuerst der Ordner, welcher für die SSHFS-Verbindung verwendet wird, erstellt werden.

Hierfür kann im Prinzip jeder Ordner verwendet werden, jedoch bietet es sich an, die Konventionen einzuhalten. Es gibt zwei Orte, welche meist zum Mounten verwendet werden. Einmal */mnt* und */run/media* (bzw. früher nur */media*). Ersteres wird für temporäre Mounts verwendet und Letzteres für Medien, welche entfernt werden können, wie z. B. Floppy-Discs, CDs oder USB-Sticks. [Rus04]

Da *thunar*, ein File-Manager, Geräte, welche unter */run/media* anzeigt und durch einen Knopf-Druck „auswerfen“ kann, wird hier dieser Ort bevorzugt. Dadurch kann die Verbindung einfach über die GUI wieder getrennt werden. Jedoch gilt auch dasselbe Prinzip, falls der Ordner */mnt* Ordner verwendet werden soll.

Zuerst sollte sich der gewünschte Ordnername überlegt werden. Dieser wird dann in einer Umgebungsvariable temporär gespeichert, damit diese Kommandos leichter zum Kopieren und Einfügen sind. In diesem Fall wird der Name *raspi1* verwendet.

```
1 $ export name="raspi1"
```

Nun muss der gewünschte Ordner erstellt werden. Dafür könnten Root-Rechte benötigt werden, deshalb wird *sudo* verwendet. Der Parameter *-p* erzeugt automatisch die ganze Ordnerstruktur. *$USER* und *$name* werden automatisch durch den Namen des aktuellen Benutzerkontos und durch den Namen, welcher zuvor definiert wurde, ersetzt.

```
1 # mkdir /run/media/$USER/$name -p
```

Da nun der Ordner mit dem *root-Benutzer*in* erstellt wurde, hat ein normaler User-Account keine Berechtigungen auf den gerade erstellten Ordner. Deshalb müssen die Berechtigungen des Ordners abgeändert werden oder der Account, welcher den Ordner besitzt. Mit folgendem Kommando wird der Account, welcher den Ordner besitzt auf den/die aktuelle*n Benutzer*in abgeändert.

```
# chown $USER:$USER /run/media/$USER/$name
```

Hierbei steht der Wert vor dem Doppelpunkt für den/die Benutzer*in, welche*m der Ordner gehört. Der Wert danach gibt die zugehörige Gruppe an. Hierbei werden beide Werte, auf den/die aktuelle*n Benutzer*in abgeändert.

Schließlich kann mit folgendem Kommando ein beliebiger Remote-Host mit diesem Ordner „verbunden“ werden.

```
$ sshfs <Benutzername des Remote-Hosts>@$<IP/Domain des
 ↪  Remote-Hosts>:/ /run/media/$USER/$name
```

Hierbei wird das Root-Verzeichnis des Remote-Hosts unter */run/media/$USER/$name* eingebaut. Ob alles funktioniert hat kann überprüft werden, indem der Inhalt des Ordners einfach angezeigt wird:

```
$ ls /run/media/$USER/$name -la
```

```
lrwxrwxrwx 1 root root     7 Aug 12 15:31 bin -> usr/bin
drwxr-xr-x 1 root root  3584 Jan  1  1970 boot
drwxr-xr-x 1 root root  3780 Oct 12 21:17 dev
[...]
drwxrwxrwt 1 root root   220 Oct 17 16:17 tmp
drwxr-xr-x 1 root root  4096 Aug 12 15:31 usr
drwxr-xr-x 1 root root  4096 Sep 29 15:58 var
```

3.2.3.3 Aktivieren der Kamera

Um die Kamera bzw. das CSI (Camera Serial Interface) zu aktivieren, muss zuerst die Konfiguration für DietPi geöffnet werden. Dies kann durch folgendes Kommando erfolgen:

```
$ dietpi-config
```

Darauf muss im Menü zu *1. Display Options → 8. RPi Camera* navigieren.

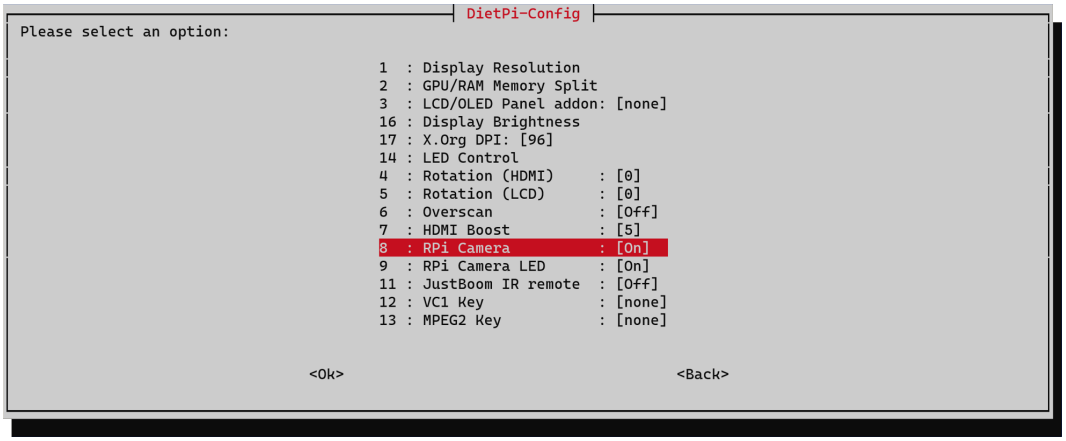


Abbildung 3.37: DietPi-Config

Durch Drücken der Eingabetaste kann dadurch die Kamera aktiviert werden und daraufhin muss der SBC neu gestartet werden.

Abbildung 3.38: Aufforderung zum Reboot

Nach dem Neustart sollten sich unter */dev/video** neue „Dateien“ befinden. Um dies zu überprüfen, können alle passenden Dateien wie folgt angezeigt werden.

```
$ ls /dev/video* -la
```

```
crw-rw---- 1 root video 81, 8 Oct  7 15:17 /dev/video0
crw-rw---- 1 root video 81, 2 Oct  7 15:17 /dev/video10
[...]
crw-rw---- 1 root video 81, 7 Oct  7 15:17 /dev/video18
```

Aus der Ausgabe kann geschlossen werden, dass es einige Devices gibt, jedoch haben nur *root* und die Gruppe *video* darauf Lese- & Schreibrechte. Nun sollte die Webcam nicht nur von *root* verwendet werden können, sondern auch von eine*m Benutzer*in ohne Root-Rechte.

Um dieses Problem zu lösen, können entweder die Berechtigungen der Datei so geändert werden, dass alle User*innen darauf Zugriff haben, oder dass alle User*innen, welche darauf Zugriff haben sollen, in die Gruppe *video* hinzugefügt werden.

Um Ersteres zu erreichen, kann mit *chmod* die Berechtigungen auf eine Datei geändert werden. Nun kann allen anderen, außer *root* & der Gruppe *video*, die Berechtigungen wie folgt gegeben werden:

```
# chmod o+rw /dev/video*
```

Jedoch sollte die zweite Variante bevorzugt werden, da hierdurch feiner kontrolliert werden kann, welche*r User*in auf die Kamera Zugriff hat. Das Hinzufügen eines Accounts zu einer Gruppe kann mittels *usermod* erfolgen. So wird durch folgendes Kommando *dietpi* zu der Gruppe *video* hinzugefügt:

```
# usermod -a -G video dietpi
```

Die Option *-G* bewirkt, dass der/die User*in zu genau den angegebenen Gruppen hinzugefügt wird, jedoch werden Gruppen, in welchen der/die User*in davor Mitglied war, entfernt, wenn sie nicht mit angegeben werden. Um dies zu verhindern, kann die Option *-a* hinzugefügt werden, wodurch die Gruppen nur hinzugefügt und nicht entfernt werden (append).

Um die Gruppen zu aktualisieren, muss der/die Benutzer*in abgemeldet und wieder angemeldet werden bzw. muss der VNC-Server neu gestartet werden. Bei TigerVNC kann dies über folgendes Kommando erfolgen:

```
# systemctl restart vncserver.service
```

Um sich nun den Videostream anzusehen, benötigt es ein weiteres Programm, nämlich *cheese*. Damit können Fotos/Videos von Webcams oder anderen Geräten, welche als Kamera erkannt werden, aufgenommen werden. Hier wird es nur zur Funktionsüberprüfung der Kamera verwendet.

```
# apt install cheese
```

Dann kann *cheese* entweder über die CLI oder über das Startmenü des jeweiligen Desktop Environments gestartet werden.

```
$ cheese
```

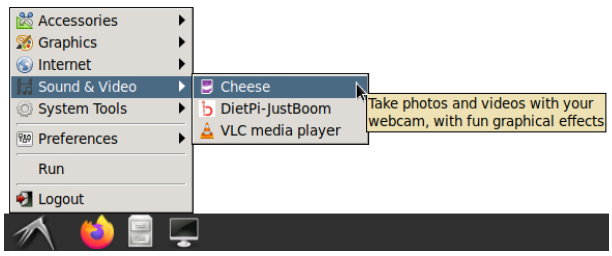


Abbildung 3.39: Starten von *cheese* in LXDE.

Nun startet das Programm und im Idealfall wird auch gleich der Videostream der Kamera angezeigt. Jedoch kann es auch sein, dass standardmäßig eine falsche Kamera ausgewählt ist. Um diese zu wechseln, muss diese in den Einstellungen geändert werden. Hierfür müssen diese über den Knopf mit den drei Balken dann unter Preferences geöffnet werden.

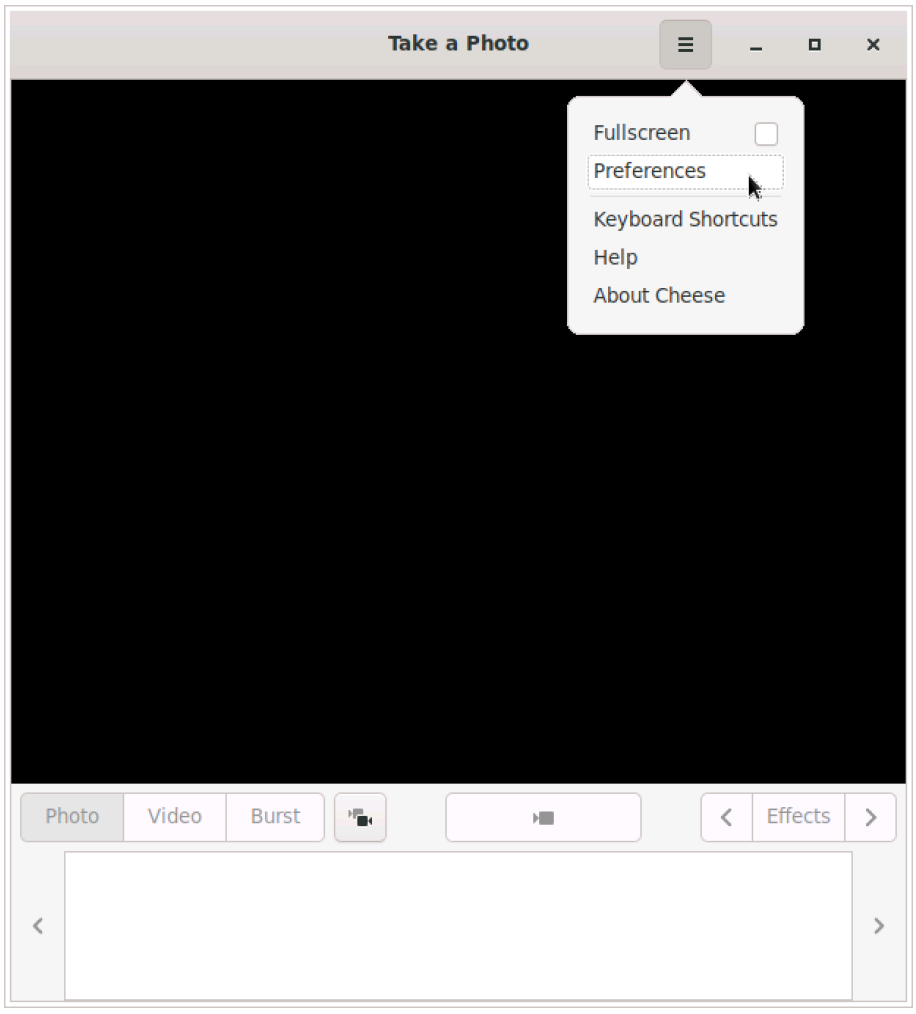


Abbildung 3.40: Öffnen der Einstellungen in *cheese*

Hier muss nun das richtige Device ausgewählt werden. Wichtig ist hier auch die Auflösung!

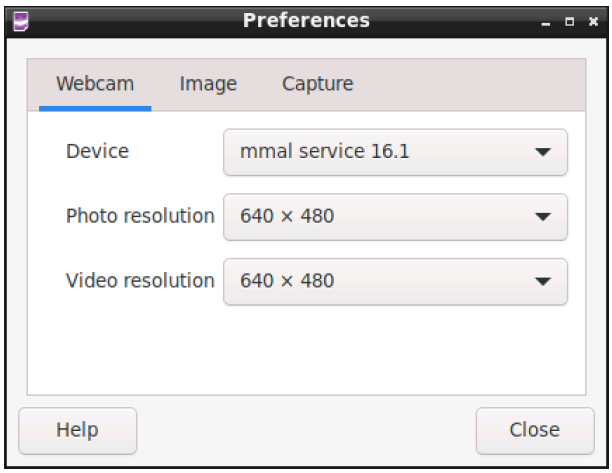
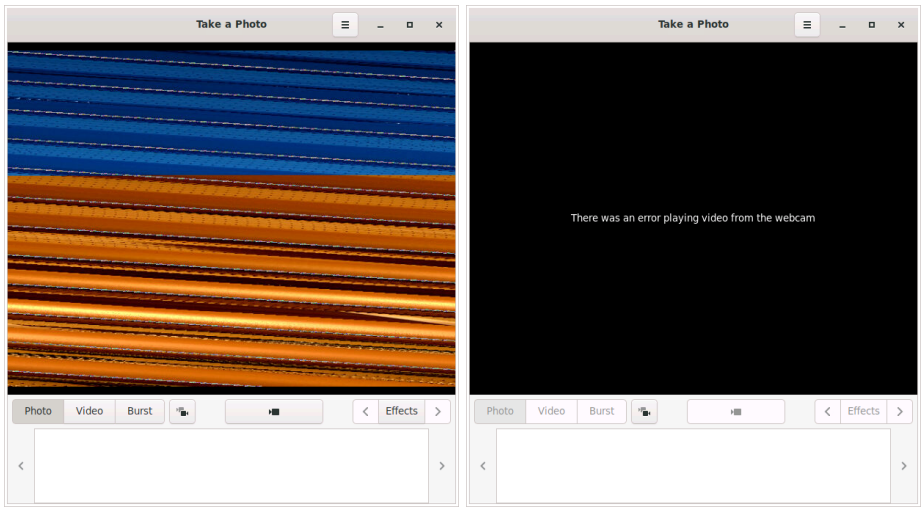


Abbildung 3.41: Einstellungen in *cheese*

Dabei gilt zu beachten, dass nicht eine zu hohe Auflösung für die Kamera verwendet wird, da es sonst zu einer Art soft-brick der Kamera führen kann. Siehe Abbildung 3.42b. Außerdem muss auch auf das Seitenverhältnis geachtet werden, da bei bestimmten Auflösungen ansonsten das Bild „nicht richtig zusammengesetzt wird“. Siehe Abbildung 3.42a.



(a) Falsches Seitenverhältnis **(b)** Zu hohe Auflösung (2560x1920)

Abbildung 3.42: Falsche Einstellungen der Auflösung

Falls ein falsches Seitenverhältnis ausgewählt wurde, reicht es einfach, die Einstellungen wieder zu öffnen. Bei einer zu hohen Auflösung musste meist *cheese* bzw. der ganze SBC neu gestartet werden.

Außerdem ist es zu erwähnen, dass mit einem Raspberry Pi 3B+ bei einer Auflösung von 1280x960 eine Bildrate von nur etwa 1 FPS (Frame[s] per Second) zu erwarten ist. Währenddessen bei einer Auflösung von 640x480 schon um die 20 bis 30 FPS zu erwarten sind. Beim Raspberry Pi 4 sieht es ähnlich aus.

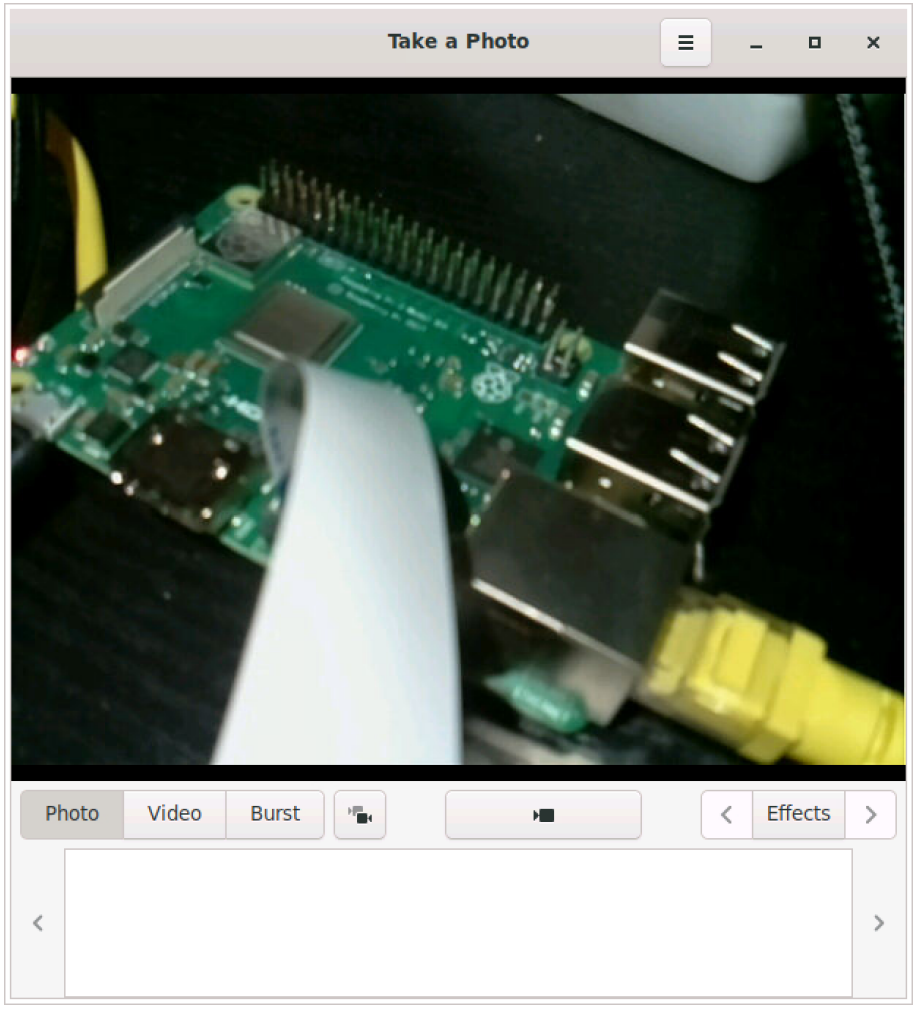


Abbildung 3.43: Anzeigen des Videostreams in *cheese*

Wenn alles richtig eingestellt wurde, sollte der Videostream in diesem Fenster angezeigt werden. Damit ist nun die Funktionsfähigkeit der Kamera und des SBCs überprüft.

3.2.3.4 Neuronales Netzwerk

Zur Erkennung der Fahrzeuge wird ein neuronales Netzwerk verwendet. Welche Unterschiede es dabei gibt, wird bereits in Unterunterabschnitt 3.2.2.3 besprochen. Da zu Beginn die Hardware, also der VIM3 noch nicht verfügbar war, mussten die neuronalen Netzwerke am Anfang auf einem Desktop-Rechner ausgeführt werden. Wie zu erwarten, erzielt dieser viel höhere Bildraten bzw. genauere und zuverlässigere Ergebnisse, als der „kleine“ Einplatinencomputer.

3.2.3.4.1 Ausprobieren von YOLO mit dem Darknet-Framework

Es gibt viele verschiedene neuronale Netzwerke und verschiedenste Frameworks, welche verwendet werden können, um das neuronale Netzwerk zu benutzen. Aber wie unter Unterunterabschnitt 3.2.2.3 bereits besprochen wurde, wird bei unserer Diplomarbeit YOLO verwendet, das es sich hierbei um

einen One-Stage Detector handelt und dadurch performanter auf langsamer Hardware funktionieren kann.

Bei YOLO gibt es bereits mehrere Versionen. Im Prinzip gibt es größere Unterschiede bis zur 3. Version, ab welcher dann nur kleine, inkrementelle Änderungen vorgenommen wurden. So hat YOLOv4 nur sogenannte „Bag of Freebies“, welche nicht mehr Rechenaufwand bedeuten, aber trotzdem die Performance des neuronalen Netzwerkes verbessern. Und „Bag of Specials“, welche im Prinzip das Model genauer machen, jedoch mit mehr Rechenaufwand verbunden sind [Mai21] [BWL20].

Und bei YOLOv5 ist sich die Community noch nicht sicher, ob es überhaupt diesen Namen verdient hab und nicht einfach eine Implementation von YOLOv3 für PyTorch (ein anderes Framework für maschinelles Lernen) ist [Mai21]. Außerdem wurde zu YOLOv5, im Vergleich zu den anderen, noch kein Paper veröffentlicht [Mai21] [eer20].

Anfänglich wurden vor allem YOLOv3 und YOLOv4 ausgetestet, weil diese beiden neuralen Netzwerke später Verwendung finden würden. Hierfür wurde das Darknet-Framework von pjreddie [pjr21] verwendet.

Darknet - einzelnes Bild

Hier sind die ersten Ergebnisse beim Benutzen dieses Frameworks:

Input File	Output Image	Network	Options [Tabelle 3.4]	Device/CPU/GPU	Time [ms]	FPS
Abbildung 3.44a	Abbildung 3.45a	YoloV3	0	RPi 4	65,570.25	0.02
			0	Ryzen 1800x	5,802.82	0.17
			2	RTX 2080 [Ryzen 1800x]	425.45	2.35
	Abbildung 3.45b	YoloV4	0	RPi 4	137,569.75	0.01
			0	Ryzen 1800x	13,863.24	0.07
			2	RTX 2080 [Ryzen 1800x]	750.20	1.33
Abbildung 3.44b	Abbildung 3.45c	YoloV3	0	RPi 4	65,527.37	0.02
			2	RTX 2080 [Ryzen 1800x]	708.79	1.41
	Abbildung 3.45d	YoloV4	0	RPi 4	137,409.04	0.01
			2	RTX 2080 [Ryzen 1800x]	721.03	1.39
Abbildung 3.44c	Abbildung 3.45e	YoloV3	0	RPi 4	65,588.77	0.02
			1		23,152.91	0.04
			2	RTX 2080 [Ryzen 1800x]	403.20	2.48
			3		500.71	2.00
			4		417.96	2.39
	Abbildung 3.45f	YoloV4	0	RPi 4	137,479.43	0.01
			1		68,230.64	0.01

	2	RTX 2080 [Ryzen 1800x]	423.11	2.36
	3		414.46	2.41
	4		417.16	2.40

Tabelle 3.3: Performance Vergleich bei einem einzelnen Bild

Option	Description
0	Default
1	OPENMP (Open Multi-Processing)
2	GPU Support
3	Explicit GPU Support for RTX 2080
4	Explicit GPU Support for RTX 2080 and C++ Support

Tabelle 3.4: Optionen für Darknet (Bilder)

Hierbei werden die Gewichtungen, welcher auf https://pjreddie.com/darknet/yolo/ gefunden werden, genommen.

Leider haben hier fast alle Ergebnisse eine sehr niedrige Bildwiederholungsrate und nur wenn Darknet mit einer GPU, in diesem Fall einer Nvidia RTX 2080, beschleunigt wird, kann eine Bildwiederholungsrate erreicht werden, welche verwendbar ist. Außerdem ist auch erkennbar, dass die Auflösung (Abbildung 3.44a > Abbildung 3.44b > Abbildung 3.44c) fast keine Rolle spielt und die Performance hier noch leicht erhöht wird. Auch spezielles Optimieren auf eine bestimmte GPU ändert auch kaum was an der Bildwiederholungsrate.

Standardmäßig wird am Raspberry Pi nur ein Kern verwendet, aber selbst mit Multithreading braucht dieser für einen Frame fast eine Minute (siehe Abbildung 3.44c mit ~68,23 s pro Frame). Leider sind auch selbst Desktop-CPUs, wie der AMD Ryzen 1800x, nicht schnell genug um Darknet performant auszuführen.

Als Testbilder wurden folgende Bilder (Abbildung 3.44) verwendet. (Testbild 3 (Abbildung 3.44c) ist einfach nur Testbild 2 (Abbildung 3.44b) auf 1280x720 skaliert.)

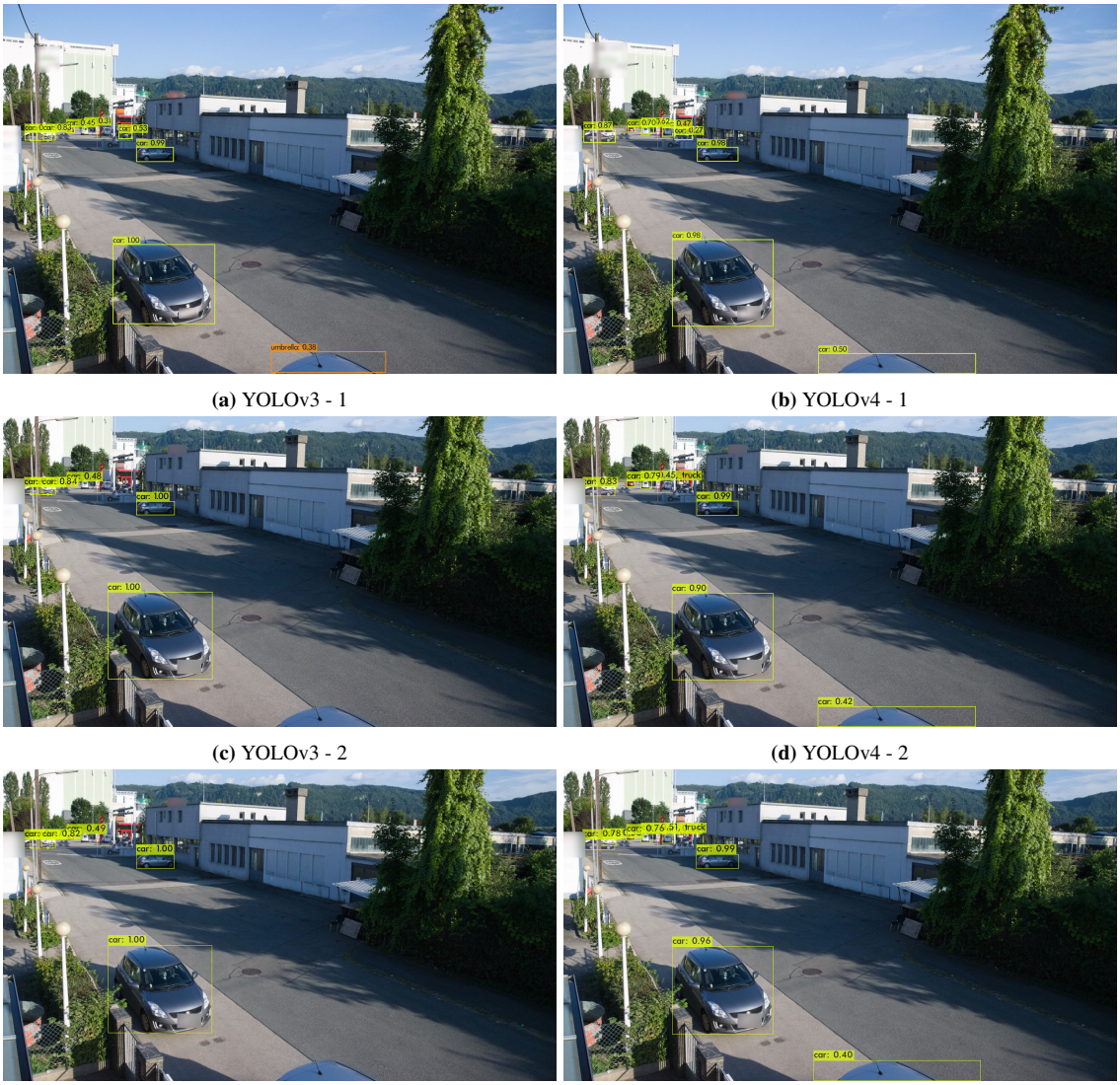
(a) Testbild 1 (6008x4008)

(b) Testbild 2 (1920x1080)

(c) Testbild 3 (1280x720)

Abbildung 3.44: Testbilder

Diese Abbildungen werden YOLO als Inputbilder übergeben.



(a) YOLOv3 - 1

(b) YOLOv4 - 1

(c) YOLOv3 - 2

(d) YOLOv4 - 2

(e) YOLOv3 - 3

(f) YOLOv4 - 3

Abbildung 3.45: Ergebnisse des neuronalen Netzwerks

Unabhängig auf welcher Hardware Darknet mit YOLO ausgeführt wird, es liefert immer dasselbe Ergebnis. Alle erkannten Objekte werden eingerahmt und die Box wird mit Objekttyp und Wahrscheinlichkeit beschriftet. Der erkannte Objekttyp kann Werte, wie z. B. „car“, für Auto, oder „truck“ für einen Lkw, annehmen. Welche Klassen es hierbei gibt, hängt davon ab, womit das neuronale Netzwerk trainiert wurde, was in diesem Fall mit dem COCO Dataset passiert ist (Mehr dazu unter Absatz 3.2.2.3.3). Die Ziffer danach bedeutet, wie sicher sich das neuronale Netzwerk ist, dass es sich hierbei um ein Objekt handelt. So bedeutet 0.98 eine 98 % Wahrscheinlichkeit, dass genau hier ein Objekt existiert. Die Bilder hier wurden im Nachhinein noch einmal bearbeitet, um Kennzeichen und andere ortsbezogene Informationsquellen unscharf zu machen.

Bei den Ergebnissen (Abbildung 3.45) kann erkannt werden, dass YOLOv3 bei dem 1. Bild mit hoher Auflösung (Abbildung 3.45a) bereits sehr gut die Fahrzeuge identifizieren kann. Der einzige, sehr ungewöhnliche Fehler, welcher hierbei auftritt, ist, dass das Fahrzeug am unteren Rand des Bildes als „umbrella“, also Regenschirm, erkannt wird. Ansonsten werden auch die relativ kleinen Fahrzeuge im Hintergrund auch noch immer sehr gut erkannt.

YOLOv4 (Abbildung 3.45b) erkennt hierbei noch mehr Fahrzeuge und diese genauer und teilweise auch mit einer noch höheren Sicherheit. Auch der Regenschirm verschwindet.

Beim Testbild 2 (Abbildung 3.44b) mit einer niedrigeren Auflösung werden dann schon weniger Fahrzeuge erkannt. Jedoch bleibt die Erkennung mit YOLOv3 (Abbildung 3.45c) noch immer sehr gut, aber die Fahrzeuge im Hintergrund werden kaum mehr bzw. etwas falsch erkannt. Allerdings wird das Fahrzeug in der Mitte des Bildes noch immer gut erkannt, was dann später auch das wichtigste ist.

YOLOv4 (Abbildung 3.45d) erkennt weiterhin erfolgreich einige Fahrzeuge im Hintergrund und selbst das Dach vom Fahrzeug am unteren Rand des Bildes wird noch immer als Fahrzeug erkannt.

Beim Testbild 3 (Abbildung 3.44c) ändert sich sowohl bei YOLOv3 (Abbildung 3.45e) als auch bei Abbildung 3.45f kaum etwas. Es werden noch immer die gleichen Fahrzeuge erkannt, jedoch reduziert sich hier nur die Wahrscheinlichkeit.

Darknet - Video

Bis jetzt wurde alles nur bei einem einzelnen Bild bzw. Frame ausprobiert, um vor allem zu Erkennen, wie gut das neuronale Netzwerk die Fahrzeuge erkennt. Aber um die eigentliche Bildwiederholungsrate zu bestimmen, wird ein Video benötigt. Hierfür werden wieder zwei Testvideos verwendet, einmal ein 1080p Video (1920x1080) und einmal ein 720p Video (1280x720). Beide Videos haben den gleichen Inhalt, jedoch haben diese unterschiedliche Auflösungen.

Input File	Network	Options [Tabelle 3.6]	Display Video	Device/CPU/GPU	Average FPS
Testvideo 1	YoloV3	0		RPi 4	0.00
		1	✓	Ryzen 1800x	1.30
		2	✓	RTX 2080 [Ryzen 1800x]	57.80
					83.80
	YoloV4	0		RPi 4	0.00
		1	✓	Ryzen 1800x	0.60
		2	✓	RTX 2080 [Ryzen 1800x]	44.30
					47.00
Testvideo 2	YoloV3	0		RPi 4	0.00
		1	✓	Ryzen 1800x	1.30
		2	✓	RTX 2080 [Ryzen 1800x]	77.10
					92.80
	YoloV4	0		RPi 4	0.00
		1	✓	Ryzen 1800x	0.60
		2	✓	RTX 2080 [Ryzen 1800x]	44.20
					47.00

Tabelle 3.5: Performance Vergleich bei einem Video

Option	Description

0	OPENMP + C++ Support
1	OPENMP + AVX + C++ Support
2	Explicit GPU Support for RTX 2080 and C++ Support

Tabelle 3.6: Optionen für Darknet (Video)

Hier ist nun die Performance schon um einiges besser, als bei dem einzelnen Frame (Tabelle 3.3). Es kann mithilfe einer dedizierten Grafikkarte schon eine Bildwiederholungsrate von 85 bis 90 Bildern pro Sekunde mit YOLOv3 oder ~45 Bilder pro Sekunde mit YOLOv4 erreicht werden. Ein Desktop-Prozessor kann zwar schon einen Frame pro Sekunde mit YOLOv3 verarbeiten, aber wirklich Realtime-Verarbeitung ist damit nur sehr begrenzt möglich. Wie zu erwarten kann der Raspberry Pi leider wieder nur sehr langsam die Frames verarbeiten und erzielt damit überall ~0 Bilder pro Sekunde.

Genau deshalb muss auf einem Einplatinencomputer der ganze Prozess mit einem eigenen Chip beschleunigt werden, da sonst keine Realtime Verarbeitung möglich ist. Dementsprechend wird in unserer Diplomarbeit auch der Khadas VIM3 verwendet.

3.2.3.4.2 NPU des VIM3 verwenden

Für die Realtime Verarbeitung des Videos wird ein VIM3 von Khadas verwendet. Dieser besitzt nicht nur einen stärkeren Prozessor als der Raspberry Pi 4, sondern auch eine dedizierte NPU (Neural Processing Unit) [Ras22a] [kha21d].

Diese NPU wird nicht automatisch von allen Frameworks verwendet und muss extra angesprochen werden. Hierbei gibt es vom Hersteller Khadas einige Möglichkeiten, diese anzusprechen.

Einerseits gibt es ein sogenanntes (AML-NPU-)SDK, wobei es nicht wirklich ein SDK ist. Im Wahrheit ist es eigentlich nur Programmcode, welcher abgeändert werden kann/muss. Dieser ist in C++ geschrieben [kha21a] [Kha21b].

Es gibt auch Support für Tengine, was sich vor allem auf embedded Devices spezialisiert [OPE21]. Jedoch sind diese Programmstückchen in C++ geschrieben und es gibt auch nicht wirklich ein SDK dafür, sondern es existiert wieder nur Programmcode [kha21c].

Aus diesen Gründen wird KSNN verwendet, da dies eine wirkliche Bibliothek ist, welche einfach in Python verwendet werden kann [kha21b].

KSNN

Im Repository für KSNN gibt es schon eine fertige Python Wheel Datei, welche nur noch installiert werden muss. Dieser Teil ist sehr gut im oben genannten Repository dokumentiert. Außerdem befinden sich auch Beispielprogramme im Repository.

So muss im Prinzip nur zuerst KSNN installiert werden. Dies erfolgt mit pip wie folgt:

```
1  $ pip3 install ksnn/ksnn-1.2-py3-none-any.whl
```

Nun ist KSNN auch schon installiert und die Beispielprogramme können ausgeführt werden. Unter dem Ordner `examples/darknet` [kha21b] befindet sich zusätzlich ein YOLOv3 Netzwerk, welches sich im Dateiformat, welches KSNN benötigt, befindet.

Ein Blick in die `README.md` zeigt dann auch schon, wie das Testprogramm ausgeführt werden kann.

```
$ python3 yolov3-picture.py --model ./models/VIM3/yolov3.nb
  ↪  --library ./libs/libnn_yolov3.so --picture ./data/1080p.bm
```

Nun müssen nur noch die Parameter abgeändert werden und voilà schon wird das Testbild 3 Abbildung 3.44c verwendet.

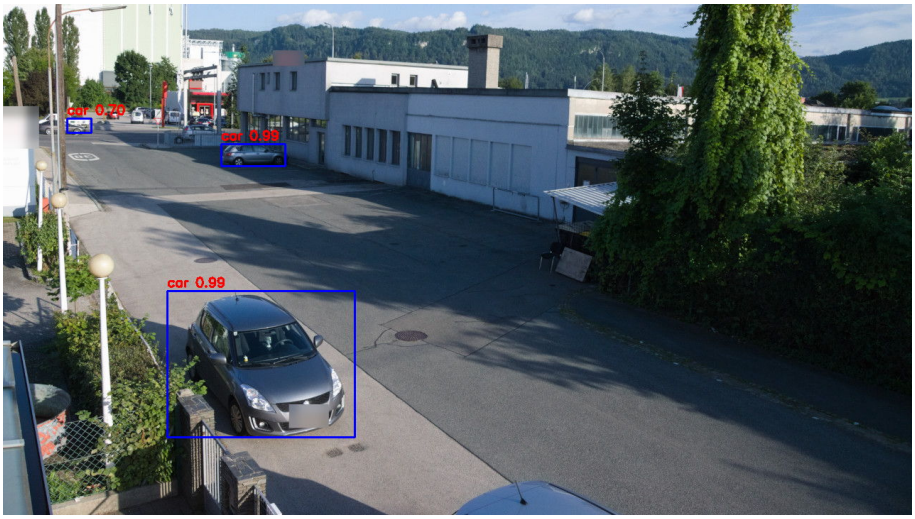


Abbildung 3.46: Testbild 3 - KSNN - YOLOv3

Wie mit freiem Auge erkannt werden kann, werden nun nicht mehr alle Fahrzeuge erkannt, sondern nur noch drei Fahrzeuge. Jedoch kann erkannt werden, dass es sich noch immer um YOLOv3 Derivate, wie YOLOv3Tiny handelt, da im Hintergrund das kleinste Objekt noch immer gleich falsch erkannt wird, wie bei Abbildung 3.45e. YOLOv3Tiny ist eine „abgespeckte“ Variante von „vollem“ YOLOv3, welche kleiner ist und performanter läuft.

3.2.3.5 Projekt Setup

Da dieser Teil der Diplomarbeit in Python geschrieben wird, muss hierfür eine Struktur gefunden werden. Theoretisch ist es mit Python möglich, das ganze Projekt in `.py`-Datei zu packen. Jedoch ist es von Vorteil, wenn das Projekt modular aufgebaut wird, damit es später leichter zum Verwalten ist.

3.2.3.5.1 Verwalten der Python-Pakete

Da das Programm auch externe Python-Pakete benötigt, muss das ein Manager verwalten. Das Packaging von Python ist noch immer eine Baustelle von Python, da es viele Möglichkeiten gibt, aber sich derzeit noch nicht so wirklich eine durchgesetzt hat. Oft ist es möglich bei den jeweiligen Linux-Distributionen das Python-Paket über das jeweilige Paketmanagement-System zu installieren.

So können unter Ubuntu mithilfe von `apt`, das Paketmanagement-System von Debian und Derivate, verschiedene Python-Pakete installiert werden. So können Pakte, wie z. B. `numpy`, welches performanteres Arbeiten mit Arrays ermöglicht, mithilfe von `apt` installiert werden.

Dies kann mit folgendem Kommando erfolgen:

```
# apt install python-numpy
```

Aber auch andere Paketmanagement-Systeme, wie `pacman`, für Arch Linux und Derivate, bieten die Möglichkeit an, die Pakete global fürs Betriebssystem zu installieren:

```
# pacman python-numpy -SP
```

Wenn nun ein Programm für eine Distribution ausgeliefert werden soll, dann sollte dieser Weg bevorzugt werden, da alle anderen installierten Programme dies auf diesen Weg überprüfen und dadurch alles pro Distribution einheitlich geregelt ist. Außerdem existiert noch der Vorteil, dass alles auch nur einmal installiert werden muss und dann von allen Programmen, welche dies benötigen, einfach verwendet werden kann.

Wenn nun jedoch eine Distribution verwendet wird, wessen Paketmanagement-System das gewünschte Python-Paket nicht anbietet, dann gibt es mehrere Wege, wie fortgefahren werden kann. Einmal gibt es den Weg, dass das gewünschte Paket von einem selber für die Distribution gewartet wird, jedoch ist dies einiges an Aufwand und solange es nicht so viele Programme benötigt wird, eher unrealistisch. Weiters gibt es die Möglichkeit, dass ein Paket-Manager für Python verwendet wird. Hier gibt es im Prinzip zwei Möglichkeiten: `pip` oder `Conda`.

`Conda` wurde ursprünglich für Python gebaut, aber unterstützt mittlerweile schon mehrere Programmiersprachen, wie R, Ruby, Scala, Java, JavaScript und noch viele mehr [Ana22]. `Pip` auf der anderen Seite wurde exklusiv nur für Python gebaut und hat einen minimalistischen Ansatz, während `Conda` versucht alles (auch mehrere Programmiersprachen) in einem Manager zu lösen [pan22]. Da `pip` den minimalistischeren Ansatz hat, mehr Pakete unterstützt [pan22], von der PyPA (Python Packaging Authority) empfohlen wird [The21] und nicht alle Funktionen von `Conda` benötigt werden, wird `pip` verwendet.

Hierbei ist es aber auch wichtig, wie die Pakete installiert werden. Im Prinzip können mit Root-Rechten alle Pakete mit `pip` global, also unter dem ganzen Betriebssystem verfügbar, installiert werden. Dies sollte vermieden werden, da diese installierten Pakete mit den Paketen der jeweiligen Linux-Distribution im Konflikt stehen können, was dann zu weiteren Problemen führen kann.

Nun ist es möglich die Pakete pro Benutzer*in zu installieren, jedoch hat auch dies Nachteile. Es wird zwar das Problem mit dem Konflikt zwischen den Paketen vom Paketmanagement-System des Betriebssystems umgangen. Jedoch gibt es noch immer das Problem, dass es nicht möglich ist zwei unterschiedliche Versionen eines Pakets zu installieren. Dies kann zu Problemen führen, wenn unterschiedliche Versionen von den einzelnen Programmen benötigt werden. Außerdem kann schnell die Übersicht verloren gehen, welche Pakete jetzt zu welchem Programm gehören, da ja einfach immer alle Pakete zur Verfügung stehen.

Genau hierfür gibt es virtuelle Umgebungen in Python. (Leider) gibt es auch hier wieder mehrere Wege zum Ziel. So kann mit `venv` wie folgt eine virtuelle Umgebung erstellt werden [Arc21]:

```
$ python -m venv Testumgebung
```

Hiermit wird im aktuellen Ordner eine virtuelle Umgebung erstellt, welche den Namen „Testumgebung“ trägt. Alternativ kann auch mit `virtualenv` die gleiche Umgebung mit diesem Kommando erstellt werden [Arc21]:

```
$ virtualenv Testumgebung
```

Nun wurde die virtuelle Umgebung „nur“ erstellt. Um diese jedoch zu verwenden, muss vorher eine gewisse Datei in der aktuellen Shell mit `source` ausgeführt werden.

```
$ source Testumgebung/bin/activate
```

Nun ist die Shell in der virtuellen Umgebung. Der Zugriff auf alle Dateien bleibt gleich, nur jetzt ist es so, als wären noch keine Python-Pakete installiert und alles befindet sich in einer reinen Umgebung.

Um diesen ganzen Aufwand etwas zu reduzieren, gibt es auch `pipenv`. Hier muss nur zu dem gewünschten Ordner navigiert werden und dann das folgende Kommando ausgeführt werden:

```
$ pipenv shell
```

Dies kombiniert die zwei Kommandos von zuvor und erstellt automatisch die virtuelle Umgebung und aktiviert diese auch gleich.

```
Creating a virtualenv for this project...
Pipfile: /home/stefan/Downloads/Testumgebung/Pipfile
Using /usr/local/bin/python (3.10.1) to create virtualenv...
Creating virtual environment...
[...]

Successfully created virtual environment!
Virtualenv location:
 ↪  /home/stefan/.local/share/virtualenvs/Testumgebung-qZ-X46rQ
Creating a Pipfile for this project...
Launching subshell in virtual environment...
. /home/stefan/.local/share/virtualenvs/Testumgebung-qZ-X46rQ
 ↪  /bin/activate

$ (Testumgebung-qZ-X46rQ)
```

Einen (praktischen) Unterschied gibt es jedoch, nämlich wie dem Output des Programmes entnommen werden kann, werden die Dateien, nicht wie bei `virtualenv` (ohne Optionen), im gleichen Ordner, sondern in einem eigenen Ordner unter dem `XDG_DATA_HOME` [Arc22] erstellt.

3.2.3.6 Optischer Fluss

Nachdem das Erkennen der Fahrzeuge erfolgreich ist, ist der nächste Schritt das Nachverfolgen der Fahrzeuge über mehrere Frames. Das neuronale Netzwerk kann nur die Fahrzeuge erkennen, hat aber kein Wissen, was im vorherigen Frame vorhanden war.

Um es möglich zu machen, Fahrzeuge über mehrere Frames hinweg zuordnen zu können, kann der optische Fluss verwendet werden. Hier wir jeder Frame mit dem nächsten Frame verglichen und dadurch bestimmt, wie sich die Objekte im Bild verhalten.

Es gibt im Prinzip zwei Varianten des optischen Flusses. Einmal gibt es den Dense Optical Flow, bei welchem alle Pixel verfolgt werden [Kuk21]. Ein prominenter Vertreter hier ist *Farneback*, wobei dieser in OpenCV schon implementiert ist.

Da der Dense Optical Flow jeden Pixel verfolgt, benötigt dieser auch einiges an Leistung.

Die zweite Variante ist der Sparse Optical Flow, bei welchem zuerst markante Punkte aus dem Bild extrahiert werden, welche dann verfolgt werden. Diese Variante benötigt nicht so viel Leistung wie der Dense Optical Flow. Ein Vertreter dieser Variante ist *Lucas-Kanade*, welcher auch bereits in OpenCV implementiert ist.

Bei dieser Implementation (Listing 4) wird Farneback verwendet, da hier alle Pixel verfolgt werden. Als Wrapper für alle Funktionen gibt es die Klasse `OpticalFlow`.

Listing 4 Optischer Fluss in Python

```python
class OpticalFlow:
    def init(self, frame: np.ndarray) -> None:
        """Initializes Optical Flow"""
        self.previous = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
        self.hsv = np.zeros_like(frame)
        self.hsv[..., 1] = 255

    def generate_flow(self, img: np.ndarray) -> None:
        self.next = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
        self.flow = cv.calcOpticalFlowFarneback(
            self.previous, self.next, None, 1 / 2, 4, 30, 3, 5, 1.1
            ↪ , 0
        )
        self.previous = self.next
```

Hier wird der optische Fluss initialisiert, indem diesem der erste Frame des Videos übergeben wird. Dann wird das Bild in Grauwerte mit `cv.cvtColor(frame, cv.COLOR_BGR2GRAY)` umgewandelt und in `self.previous` gespeichert.

Mit der Methode `generate_flow` wird der nächste Frame übergeben und mithilfe von OpenCV und der Methode `calcOpticalFlowFarneback()` wird der optische Fluss berechnet.

Listing 5 Parameter von `calcOpticalFlowFarneback()` [Ope21]

```
calcOpticalFlowFarneback (  InputArray          prev,
                            InputArray          next,
                            InputOutputArray    flow,
                            double              pyr_scale,
                            int                 levels,
                            int                 winsize,
                            int                 iterations,
                            int                 poly_n,
                            double              poly_sigma,
                            int                 flags
                         )
```

Der Methode `calcOpticalFlowFarneback()` müssen diese Parameter übergeben werden. Für `pyr_scale` wird der Standardwert von `1/2` verwendet, wobei sowohl die `winsize` als auch `iterations` wurde so angepasst, dass der optische Fluss noch immer alles gut erkennt, aber die Performance gesteigert wird. Schließlich wird dann der aktuelle Frame als alter Frame unter `self.previous` gespeichert, damit beim nächsten Frame auch hier wieder der optische Fluss angewendet werden kann.

Um sich nur den optischen Fluss anzeigen zu lassen, kann mit folgender Methode auch ein Bild generiert werden:

Listing 6 Bild aus optischen Fluss generieren.

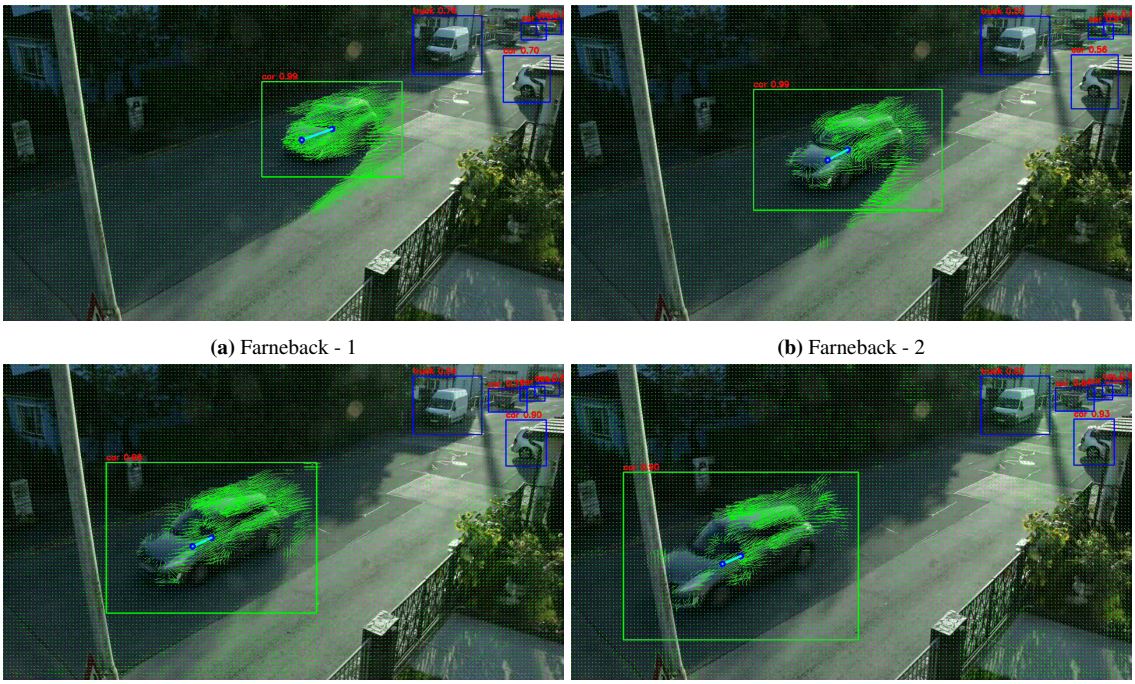
```
1     def generate_img(self) -> cv.cvtColor:
2         mag, ang = cv.cartToPolar(self.flow[..., 0], self.flow[...,
          ↪  1])
3         self.hsv[..., 0] = ang * 180 / np.pi / 2
4         self.hsv[..., 2] = cv.normalize(mag, None, 0, 255, cv.
          ↪  NORM_MINMAX)
5
6         return cv.cvtColor(self.hsv, cv.COLOR_HSV2BGR)
```

Hier kann, je nach Farbton, die Stärke und Richtung des optischen Flusses erkannt werden. Alternativ kann der optische Fluss auch so visualisiert werden, indem ein Änderungsvektor zwischen der Position im vorherigen Frame und des aktuellen Frames eingezeichnet wird. Jedoch kann dieser Vektor auch nicht an allen Pixeln eingezeichnet werden, da sonst nur noch Vektoren sichtbar wären und das Bild nicht mehr. Um den optischen Fluss so zu visualisieren, gibt es die Methode `add_vectors_to_img()`.

Listing 7 Bild aus optischen Fluss generieren.

```
1  def add_vectors_to_img(self, img: np.ndarray, spacing=8) ->
   ↪ None:
2      img_height = img.shape[0]
3      img_width = img.shape[1]
4
5      y = 0
6      for flow_x in self.flow:
7          x = 0
8          if y % spacing == 0:
9              for flow_y in flow_x:
10                 if x % spacing == 0:
11                     mag_x = flow_y[0]
12                     mag_y = flow_y[1]
13
14                     end_x = int(x + mag_x)
15                     end_y = int(y + mag_y)
16                     end_x = end_x if 0 <= end_x and end_x <=
                        ↪ img_width else x
17                     end_y = end_y if 0 <= end_y and end_y <=
                        ↪ img_height else y
18                     cv.line(img, (x, y), (end_x, end_y), (0,
                        ↪ 255, 0), 1)
19                 x += 1
20         y += 1
```

Wenn dieses Video nun angezeigt wird, sehen die Frames wie folgt aus:



(a) Farneback - 1

(b) Farneback - 2

(c) Farneback - 3

(d) Farneback - 4

Abbildung 3.47: Farneback

Nun kann der optische Fluss, welcher sich in einem Objekt befindet, welches von YOLO erkannt wurde, zusammengezählt werden und dadurch kann die nächste Position vorhergesagt werden.

Listing 8 Voraussichtliche zukünftige Position vorhersagen

```
    def get_direction_of_rect(self, box: Box, fps: int) ->
     ↪  OpticalFlowValue:
        """Given a specific portion of the frame, this function
           calculates the optical flow of this area.

        :param box: Area on which the optical flow should be
                    calculated
        :type box: Box
        :param fps: Frames per Second
        :type fps: int
        :return: An object which encapsulates the optical flow
        :rtype: OpticalFlowValue
        """
        left, top, right, bottom = box.get_values()
        cropped_flow = self.flow[top:bottom, left:right, :] / fps

        flow_sum = np.sum(np.sum(cropped_flow, axis=0), axis=0)
        std_x = np.std(cropped_flow[:, :, 0])
        std_y = np.std(cropped_flow[:, :, 1])

        return OpticalFlowValue(flow_sum[0], flow_sum[1]), (std_x,
         ↪  std_y)
```

Hier wird einmal die Summe und die Standardabweichung bestimmt. Die Summe wird benötigt, um die nächste Position des Fahrzeuges vorherzusagen. Die Standardabweichung wird gebraucht, um zu entscheiden, ob sich das Fahrzeug nun wirklich bewegt (kleine Standardabweichung) oder ob es nur eine Störung ist (große Standardabweichung). Es kann nämlich sein, dass sich ein Fahrzeug vor ein anderes Fahrzeug bewegt und die Rechtecke dadurch überlappen. Damit beide Fahrzeuge nun nicht als „in Bewegung“ erkannt werden, wird auf die Standardabweichung geachtet und erst wenn diese einen gewissen Schwellenwert erreicht, dann wird das Fahrzeug erkannt. Dies ist bei den Bildern durch die grüne bzw. blaue Umrahmung dargestellt.

So werden Fahrzeuge, welche grün umrahmt sind, als „in Bewegung“ erkannt, währenddessen Fahrzeuge, welche blau umrahmt sind, als stehend markiert sind.

Wenn jetzt nur der aus den Berechnungen resultierende Vektor angezeigt wird, dann sieht es wie folgt aus Abbildung 3.48.

Abbildung 3.48: Optischer Fluss in Arbeit

Ein blauer Kreis befindet sich immer in der Mitte des erkannten Rechtecks, während der zweite blaue Kreis die nächste Position vorhergesagt. Wenn die einzelnen Frames übereinandergelegt werden, kann erkannt werden, dass die Vorhersage immer relativ gut mit der nächsten Position übereinstimmt Abbildung 3.49. (Leider wurde die Kamera in der Hand gehalten, wodurch nun das ganze Bild verschwommen wirkt, jedoch funktioniert der Algorithmus trotzdem.)

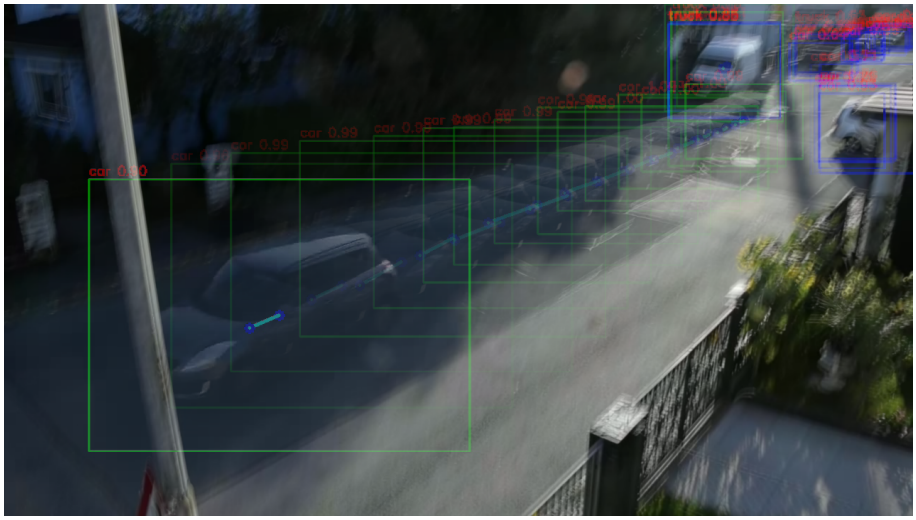
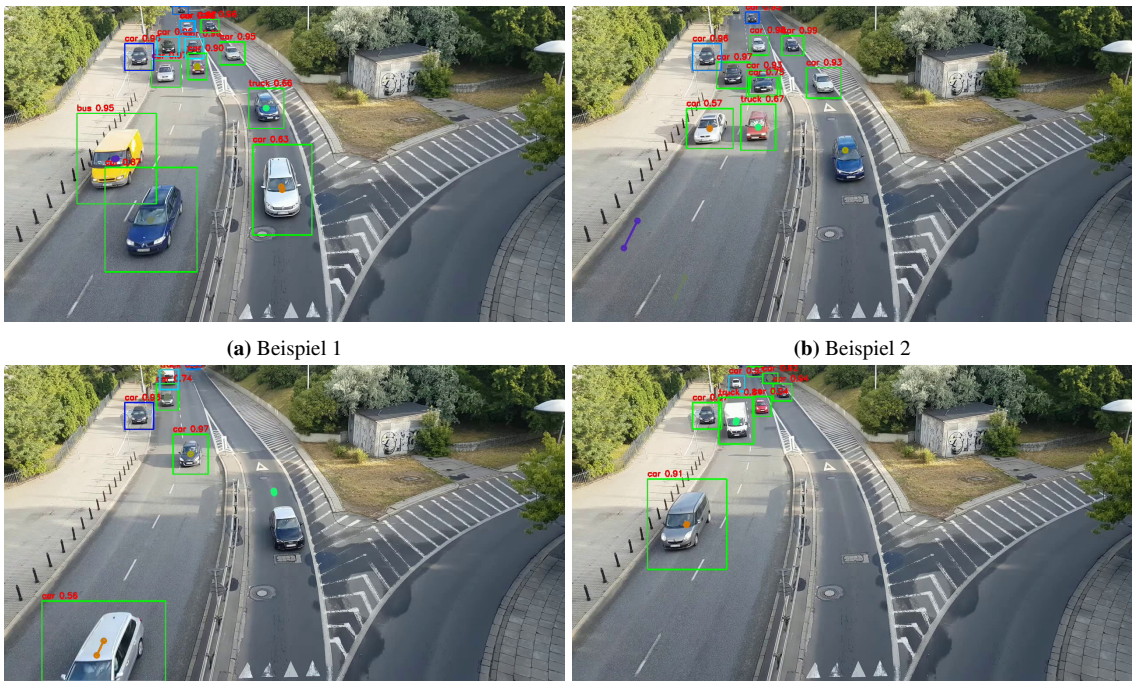


Abbildung 3.49: Optischer Fluss in Arbeit (Überlagert)

Hier funktioniert der optische Fluss ganz gut, da das Video eine (relativ) hohe Bildwiederholungsrate von 5 Bildern pro Sekunde hat. Außerdem ist es auch von Vorteil, dass sich hier nur ein Fahrzeug bewegt, welches sich prominent im Bild befindet. Dadurch kann Farneback relativ gut erkennen, wohin sich bestimmte Pixel bewegt haben.

Wenn jetzt jedoch ein anderes Video genommen wird [Maj18], sind die Ergebnisse schon anders.



(a) Beispiel 1 **(b)** Beispiel 2

(c) Beispiel 3 **(d)** Beispiel 4

Abbildung 3.50: Algorithmus funktioniert nicht so gut [Maj18]

Hier kann erkannt werden, dass der optische Fluss einige Probleme hat. So ist anscheinend das Video so enkodiert, dass der optische Fluss irgendwelche Artefakte erkennt. Das hat zur Folge, dass die Änderungsvektoren in die falsche Richtung zeigen und damit für die Positionsvorhersage der Autos eine verfälschte Richtung angeben. Aus diesem Grund ist die vorhergesagte Position des Fahrzeuges manchmal genau gegenüberliegend von der Richtung, in welche sich das Fahrzeug eigentlich bewegt. Weiters erschwert die geringe Bildwiederholungsrate die Erkennung (~3 Bilder pro Sekunde). Dadurch bewegen sie die einzelnen Fahrzeuge zwischen den Frames noch weiter und es fällt Farneback noch schwerer diese Fahrzeuge zu verfolgen. Dies könnte durch eine bessere Konfiguration der Parameter Listing 5 verbessert werden, jedoch würde dies auch wieder die Verarbeitungsgeschwindigkeit reduzieren, was sich wiederum negativ auf die Bildwiederholungsrate und weiters auf das Verfolgen auswirken würde.

Bei manchen Frames ist die Erkennung akzeptabel, wie bei Abbildung 3.50a erkannt werden kann. Hier werden alle Fahrzeuge, welche sich bewegen, grün umrahmt und es wird auch ein Vektor in die richtige Richtung gezeichnet. Jedoch gibt es auch bei der Berechnung der Vektorlänge ein Problem, wie bei dem Video [Maj18] gut erkennbar ist. So entsprechen ein paar Pixel im Hintergrund eine größere Distanz, als die gleiche Anzahl an Pixel im Vordergrund. Dies wird jedoch nicht wirklich berücksichtigt und resultiert darin, dass die Vorhersage für weit entfernt Fahrzeuge zu groß oder für Fahrzeuge nahe bei der Kamera zu klein ist. Aber bis auf diese Schwierigkeit spielt hier auch das neuronale Netzwerk mit und es werden erfolgreich alle Fahrzeuge erkannt.

Jedoch schon beim zweiten Beispiel Abbildung 3.50b können die Probleme schon erkannt werden, wenn das neuronale Netzwerk einmal ein Fahrzeug nicht erkennt, bzw. doppelt. Hier wird zwar ein Vektor gezeichnet, nur nachdem die Länge von diesem nicht stimmt, kann die Position auch eher schlecht vorhergesagt werden. Außerdem ist ein weiters Problem, dass aufgrund der größeren Abstände, in welchen sich die Fahrzeuge zwischen den Frames bewegen, dass dadurch der optische Fluss irritiert wird. Auch das Encoding/Rauschen des Videos tut dem nicht gut und dadurch wird bei diesem Bild der violette Vektor, der sich in der linken unteren Ecke des Videos befindet auch in die falsche

Richtung gezeichnet, wobei das eigentliche Fahrzeug schon längst aus dem Bild verschwunden ist.

Weiters sind auch beim dritten Beispiel Abbildung 3.50c dieselben Probleme zu vermerken. So wird hier auch das Fahrzeug in der Mitte nicht mehr vom neuronale Netzwerk erkannt und da die Länge des Vektors nicht passt, kann die Position auch nicht gut weiter verfolgt werden.

Schließlich gab es sogar beim Überprüfen der Standardabweichung Probleme mit der Zuverlässigkeit. So wurden stehende Fahrzeuge als fälschlicherweise „in Bewegung“ erkannt. Dies kann bei Beispiel 4 (Abbildung 3.50d) erkannt werden, da das stehende Fahrzeug im Hintergrund fehlerhaft grün umrahmt wird und dadurch als „in Bewegung“ erkannt wird.

Da das Anpassen der Parameter keine Option ist, da dies die Verarbeitungsgeschwindigkeit negativ beeinflussen würde, musste nun nach Alternativen gesucht werden. Als Option würde stehen, dass anstatt einem Dense Optical Flow, wie *Farneback*, ein Sparse Optical Flow, wie *Lucas-Kanade* verwendet werden würde.

Jedoch wurde dann gleich beschlossen, dass der nächste Schritt zum *Feature Matching* gemacht werden soll, da für *Lucas-Kanade* auch zuerst Punkte zum Verfolgen gesucht werden müssen, welche dann später am nächsten Frame versucht zugeordnet zu werden. Jedoch bietet *Feature Matching* zusätzlich noch den Vorteil, dass diese Keypoints dann an den Server geschickt werden könnten und somit das Verfolgen des Verkehrs über mehrere Nodes möglich wäre.

3.2.3.7 Feature Matching

Das Prinzip passiert darauf, dass zuerst gewisse markante Punkte aus einem Bild extrahiert werden. Diese werden Features genannt. Um solche Features zu finden, muss zuerst ein Feature Detector verwendet werden.

Hierfür gibt es verschiedene Feature Detector wie:

- SIFT - Scale-invariant feature transform
- SURF - Speeded up robust features
- ORB - Oriented FAST and rotated BRIEF

SIFT und SURF sind patentiert. Im Gegensatz ist ORB nicht patentiert und ist sogar auch schneller und benötigt weniger Ressourcen [Dub22]. ORB wurde von OpenCV Labs entwickelt und ist im Prinzip eine Weiterentwicklung von FAST und BRIEF. Features, welche nur mit FAST erkannt wurden, sind nicht skalierungs- und rotationsunabhängig. Deshalb wurden Eigenschaften von BRIEF hinzugefügt und dadurch ist nun ORB auch skalierungs- und rotationsunabhängig [Dub22].

Nun müssen diese Features nur noch zwischen zwei Bildern, bzw. in diesem Fall Frames, zugeordnet werden. Hierfür gibt es in der OpenCV Bibliothek zwei Matcher.

- BFMatcher - Brute-Force Matcher
- FLANN based Matcher - Fast Library for Approximate Nearest Neighbors

Wie der Name schon verrät, ist der BFMatcher zwar nicht der beste Matcher im Hinblick auf Performance, jedoch erzielt dieser die besten Ergebnisse. FLANN ist optimiert auf nearest neighbor search in großen Datensätzen [Ope22].

Abbildung 3.51: SIFT mit FLANN [Maj18]

Hier werden alle gefunden Features mit einem kleinen (bunten) Kreis markiert. Auf der linken Seite befindet sich der vorherige und auf der rechten der nächste Frame. Bei Abbildung 3.51 werden alle Features angezeigt und die Matches zwischen beiden Frames mit einer Linie markiert.

Außerdem ist es auch gut zu erkennen, dass SIFT vor allem beim Graffiti im Hintergrund viele Features erkennt und im nächsten Frame auch erfolgreich zuordnen kann (mithilfe von FLANN). Auch der Übergang zwischen Scheinwerfer und der Karosserie der Autos ist für SIFT oft ein Feature wert. Da SIFT leider ein Patent besitzt, verwenden wir es nicht weiter und setzen deshalb auf die patent-freie Alternative ORB:

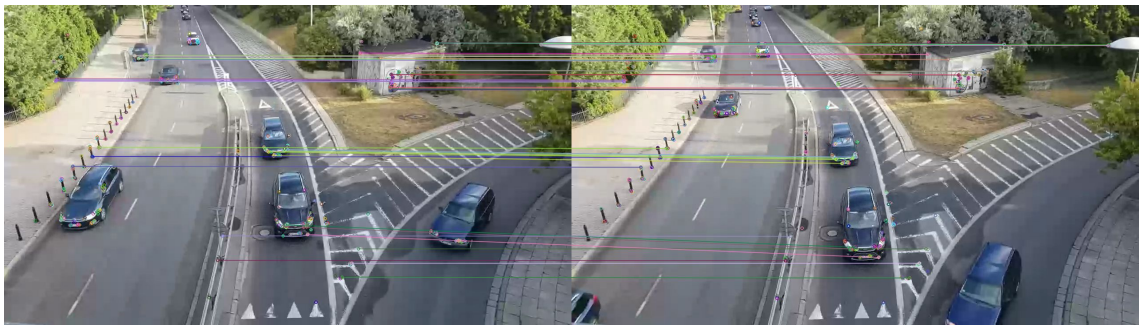


Abbildung 3.52: ORB mit BFMatcher [Maj18]

In Abbildung 3.52 werden zur besseren Übersicht nur die besten Features eingezeichnet. Im Prinzip passiert hier das gleiche wie bei SIFT und ist sogar etwas performanter. So ist auch hier das Graffiti, da es einen starken Kontrast zwischen den weißen und dunklen Flächen bietet, sehr interessant.

Da es ziemlich sperrig ist, immer zwei aufeinanderfolgende Frames nebeneinander anzuzeigen, gibt es einen anderen Weg die zugeordneten Features darzustellen. So werden bei Abbildung 3.53 die zugeordneten Features einfach von der Position, an welcher die Features im letzten Frame waren, zu dem aktuellen Frame eingezeichnet. Dies sollte im Idealfall dazu führen, dass alle Fahrzeuge eine Spur mit Linien hinter sich haben. Wird ein Fahrzeug fälschlicherweise einem anderen Fahrzeug zugeordnet, werden die Linien zu dem anderen (falschen) Fahrzeugen gezogen.

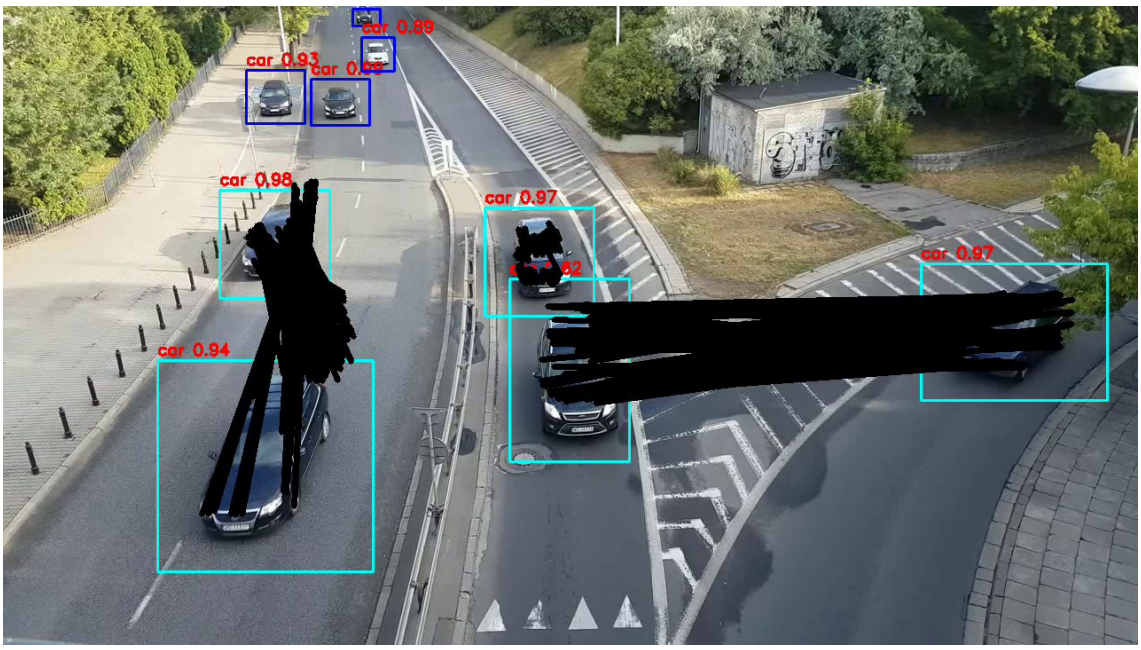


Abbildung 3.53: ORB ohne Gewichtung der Position

In Abbildung 3.53 wird zuerst mithilfe des neuronalen Netzwerkes (YOLO) die Position des Fahrzeuges bestimmt und das Bild an dieser Stelle zugeschnitten. In diesem Bildausschnitt sucht dann ORB nach Features. Diese werden mit den Features vom vorherigen Frame verglichen und dann wird Linie zwischen zwei Features eingezeichnet, welche zusammen gehören.

Leider kann hier auch erkannt werden, dass dies nicht perfekt funktioniert. So werden z. B. auf der linken Seite die zwei Fahrzeuge miteinander vertauscht. Dies kann daran erkannt werden, dass das Fahrzeug, welches sich näher am unteren Rand des Bildes befindet, die Vektoren nach „hinten“ verbunden hat. Das andere Fahrzeug zeigt mit dessen Vektoren genau auf die Position, an welcher sich das ursprüngliche Fahrzeug im vorherigen Frame befunden hat.

Allerdings kann es auch besser funktionieren, wie am oberen Fahrzeug in der Mitte (97 % Wahrscheinlichkeit) beobachtet werden kann. Hier verbinden nur kurze Vektoren die Matches miteinander. Dies bringt dann schon den ersten Optimierungsschritt, um das Zuordnen zu verbessern.

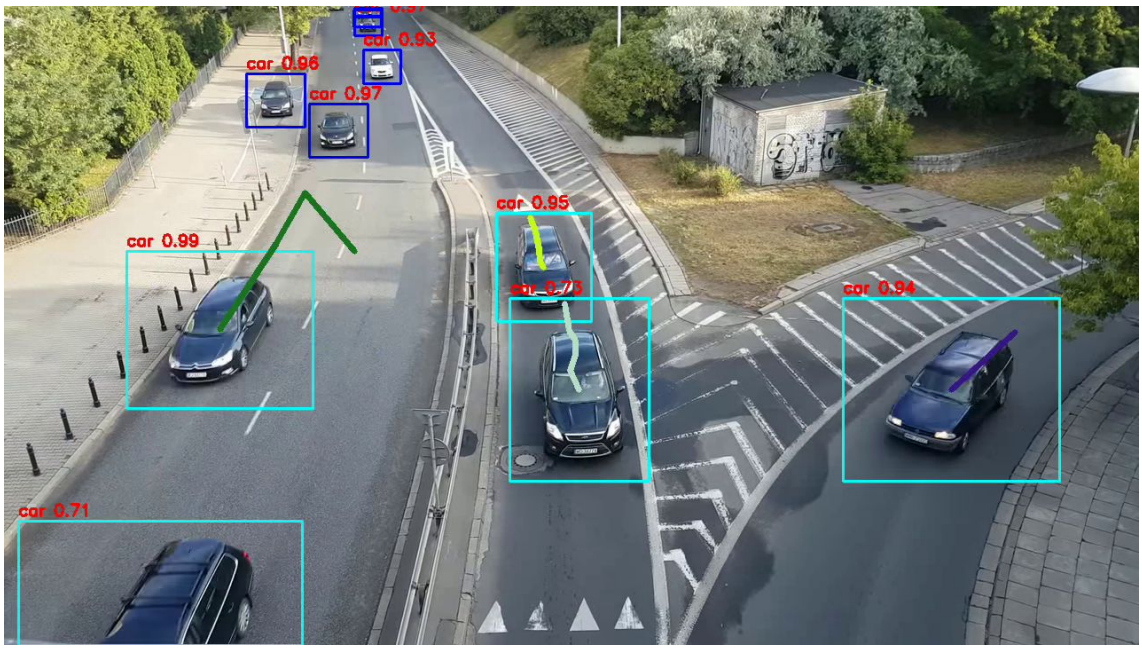


Abbildung 3.54: ORB mit Gewichtung der Position

In Abbildung 3.54 wird die Position auch mitberücksichtigt. Sowohl die Distanz, welche die einzelnen Features voneinander entfernt sind, aber auch die absolute Position des Fahrzeuges im Frame.

Hier wird dann auch zusätzlich nur noch die Position mithilfe einer Linie dargestellt. Wie zu sehen ist, funktioniert diese Gewichtung der Werte schon relativ gut. Aber wie auch zu erkennen ist, beim Fahrzeug am linken Bildrand ist ein „Knick“ in der Positionsverfolgung. Dies liegt daran, dass hier das Fahrzeug nicht als neues Fahrzeug erkannt wurde, sondern dass es einem anderen Fahrzeug zugeordnet wurde. Der Grund dafür liegt an der Gewichtung. In diesem Fall überwiegt die Gewichtung des Feature Matching gegenüber der Position. Dadurch wird das Fahrzeug dem anderen Fahrzeug, welches das Bild verlässt, zugeordnet, anstatt als neues Fahrzeug erkannt zu werden.

Bei den anderen Fahrzeugen hat das Ganze jedoch ganz gut geklappt.

Listing 9 Gewichtung der Fahrzeuge [Pis21]

```
    def is_this_me(self, vehicle_detected: Vehicle, matches: tuple
    ↪ ):
        if type(vehicle_detected) != type(self):
            self.logger.info("Invalid type!")
            return sys.maxsize

        wrong_class_penalty = 1
        if vehicle_detected.get_object_class() != self.
        ↪ get_object_class():
            self.logger.debug(
                "Invalid object class! %s - %s",
                vehicle_detected.get_object_class(),
                self.get_object_class(),
            )
            wrong_class_penalty = 1.5

        my_center = self.get_center()
        their_center = vehicle_detected.get_center()

        diff = tuple(map(lambda i, j: i - j, my_center,
        ↪ their_center))
        length = np.linalg.norm(diff)

        distance_sum = 0
        for match in matches:
            distance_sum += match.distance

        return (
            distance_sum * wrong_class_penalty * length * 7 / (len(
            ↪ matches) * 10000)
        ) ** 2
```

Jedes Fahrzeug besitzt eine `is_this_me`-Methode, welche im Prinzip einen Score zurückgibt, wie gut diese zwei Fahrzeuge zusammenpassen. Je geringer der Wert, desto besser passt es.

Zuerst wird überprüft, ob es sich bei beiden Objekten um den gleichen Typ (`Vehicle`) handelt. Ansonsten wird der maximale Zahlenwert zurückgegeben.

Danach wird davon ausgegangen, dass beide Fahrzeuge vom neuronalen Netzwerk die gleiche Klasse bekommen haben, deshalb wird auch `wrong_class_penalty` auf 0 `0` gesetzt. Falls jedoch beide Typen nicht miteinander übereinstimmen, dann wird dies mit einer Erhöhung des Wertes um 50 % bestraft, was nicht so hoch ist. Dies liegt nämlich daran, dass das neuronale Netzwerk gerne Fahrzeuge einmal als `car` und dann im darauffolgenden Frame als `truck` identifiziert. Würde nun die Änderung des Typs zu hoch bestraft werden, dann würde dies als ein neues Fahrzeug erkannt werden.

Danach wird der Mittelpunkt beider Fahrzeuge bestimmt und der Abstand berechnet. Schließlich werden dann die Distanzen der einzelnen Features zusammengezählt.

Aus diesen zwei bzw. drei Werten wird dann der Score berechnet.

$$m\text{...Matches} \tag{3.2}$$
$$\vec{v_1}\text{... Ortsvektor zu Fahrzeug 1} \tag{3.3}$$
$$\vec{v_2}\text{... Ortsvektor zu Fahrzeug 2} \tag{3.4}$$
$$c\text{... Bestrafung für falschen Fahrzeugtyp} \tag{3.5}$$

$$score = (\bar{m} * c * (\vec{v_1} - \vec{v_2}) * \frac{7}{10000})^2 \tag{3.6}$$

3.2.3.8 Regression

Damit nicht nur die Differenz zwischen aktuelle und der Position im vorherigen Frame berücksichtigt wird, kann mithilfe von einer Regression die nächste Position vorhergesagt werden. Hierfür wird als unabhängige Variable die Zeit betrachtet. X und Y-Koordinaten sind dann die abhängigen Variablen.

Ein erster Versuch war es, mithilfe von einer Taylor-Reihenentwicklung die Funktion zu bestimmen.

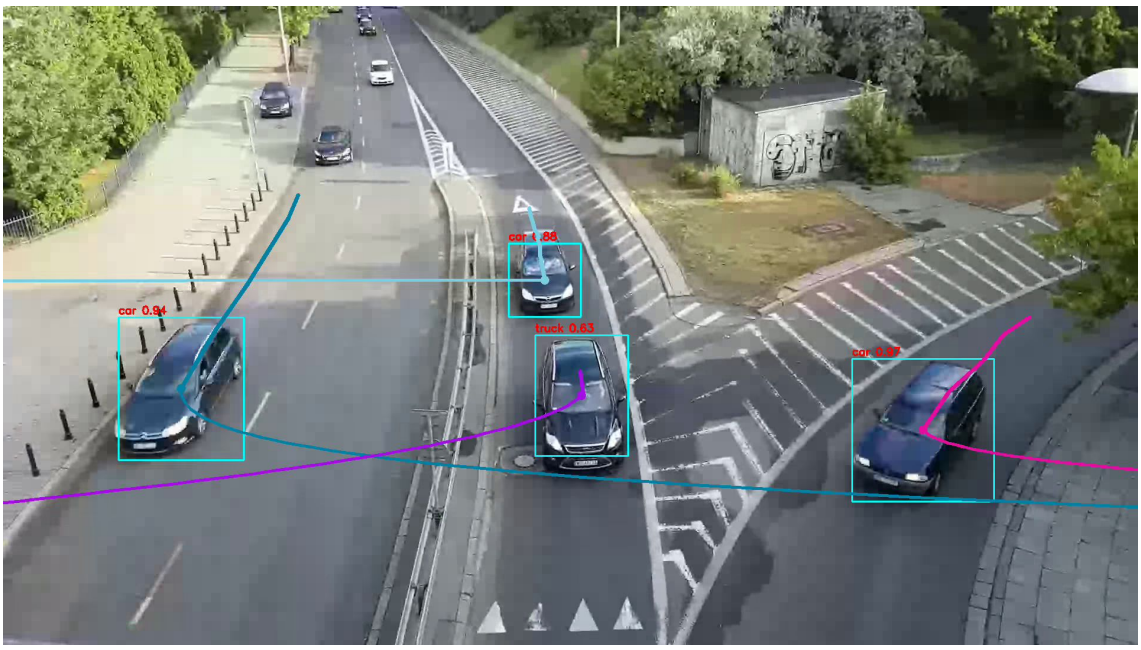


Abbildung 3.55: Taylor-Reihenentwicklung

Wie jedoch in Abbildung 3.55 erkannt werden kann, versucht die resultierende Funktion alle Punkte „genau zu treffen“. Dadurch bekommt die Funktion sehr schnell eine sehr hohe Ordnung und die Vorhersage bricht gleich in eine Richtung aus.

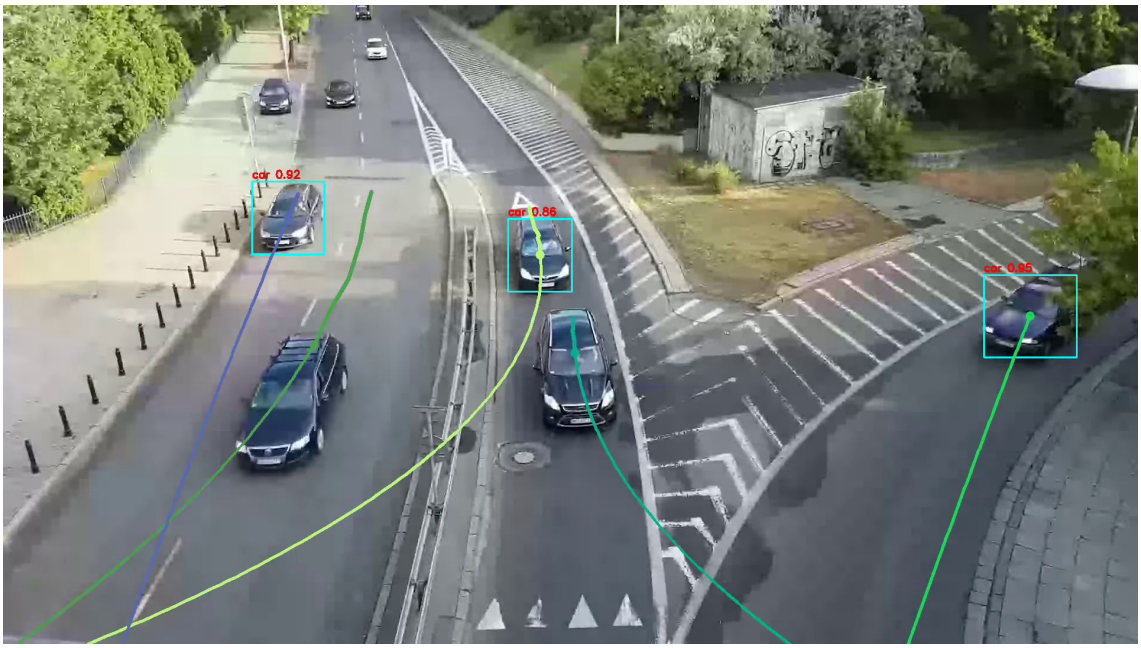


Abbildung 3.56: Lineare Regression 2. Grades

Zum Vergleich, wenn eine lineare Regression verwendet wird, dann kann der Grad der Polynomfunktion angegeben werden. Dadurch werden zwar nicht alle Punkte „genau getroffen“, jedoch steigt die Ordnung der Funktion auch nur auf das angegebene Niveau.

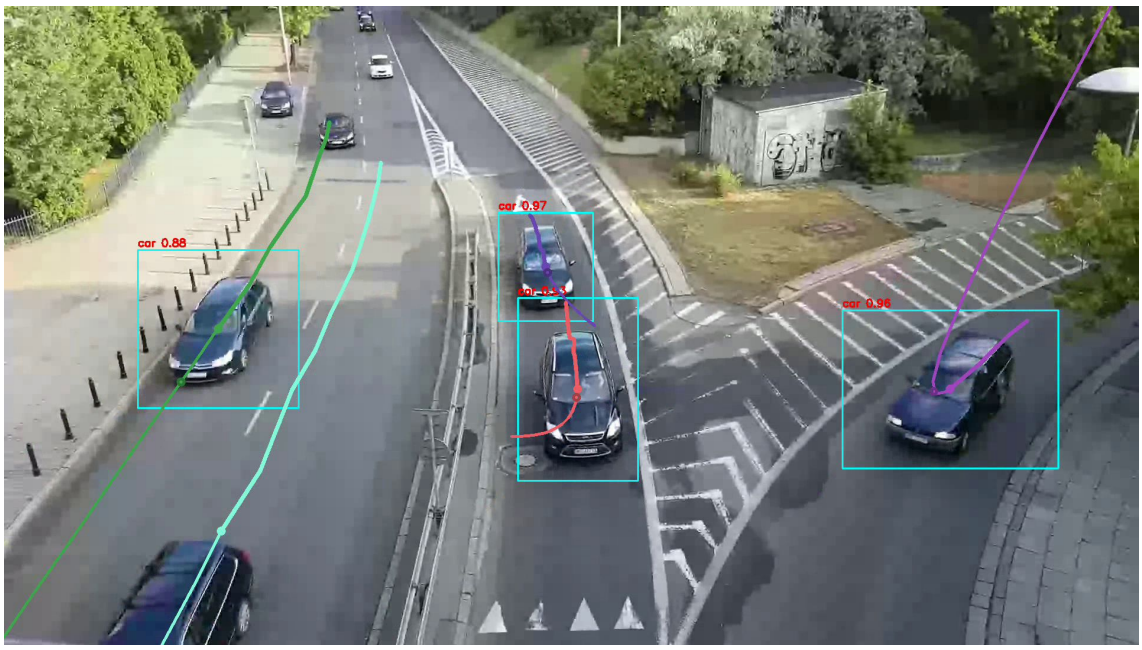


Abbildung 3.57: Lineare Regression 3. Grades

Im Prinzip würde auch eine Polynomfunktion 2. Grades (Abbildung 3.56) reichen, aber um komplexere Verhalten, wie das Beschleunigen oder auf einer Kreuzung abzubiegen, zu beschreiben, ist es von Vorteil, wenneine Polynomfunktion 3. Grades zu verwenden. Diese bricht zwar auch leichter aus, aber weil es nur darum geht, die Position im Nahbereich zu berechnen, reicht dies aus.

Listing 10 Klasse zum Vorhersagen der Position [Pis22a]

```
1  import numpy as np
2  from numpy.polynomial import Polynomial
3
4  class MultipleRegression:
5      def __init__(self, x: np.array, y: np.array, time_now: np.array
       ↪  ):
6          self.model_x = Polynomial.fit(time_now, x, 3, domain=[0,
           ↪  3000])
7          self.model_y = Polynomial.fit(time_now, y, 3, domain=[0,
           ↪  3000])
8
9      def get_x(self, time_now: np.array):
10         return self.model_x(time_now)
11
12     def get_y(self, time_now: np.array):
13         return self.model_y(time_now)
```

Für die Polynomfunktion stellt bereits `numpy` eine geeignete Klasse (`Polynomial`) bereit. Diese wird hier nur in einen Wrapper gepackt, damit sowohl X als auch Y-Koordinaten sich gesammelt in einem Objekt befinden.

Außerdem wurde auch der Scoring-Algorithmus etwas abgeändert:

Listing 11 Gewichtung der Fahrzeuge mit Regression

```
    def is_this_me(self, vehicle_detected: Vehicle, matches: tuple
    ↪ ):
        ...
        predicted_next_time = TimePassed.get_next_time()
        their_center = vehicle_detected.get_center()
        my_predicted_center = (
            self.multiple_regression.get_x(predicted_next_time),
            self.multiple_regression.get_y(predicted_next_time),
        )

        diff = tuple(map(lambda i, j: i - j, my_predicted_center,
        ↪ their_center))
        length = np.linalg.norm(diff)
        ...
        return (
            distance_sum * wrong_class_penalty * length / (len(
            ↪ matches) * 1000)
        ) ** 2
```

Hier wird im Vergleich zu (Code: Listing 9 - Formel: Gleichung 3.6) nun nicht mehr die aktuelle, sondern die vorhergesagte Position verwendet. Außerdem wurde der Faktor auch etwas angepasst.

m...Matches (3.7)

$\vec{v_{pred}}$... Vorhergesagte Position (3.8)

$\vec{v_o}$... Ortsvektor zum anderem Fahrzeug (3.9)

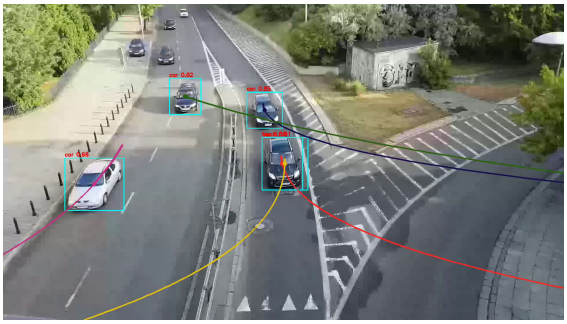
c... Bestrafung für falschen Fahrzeugtyp (3.10)

$$score = (\bar{m} * c * (\vec{v_{pred}} - \vec{v_o}) * \frac{1}{1000})^2 \tag{3.11}$$

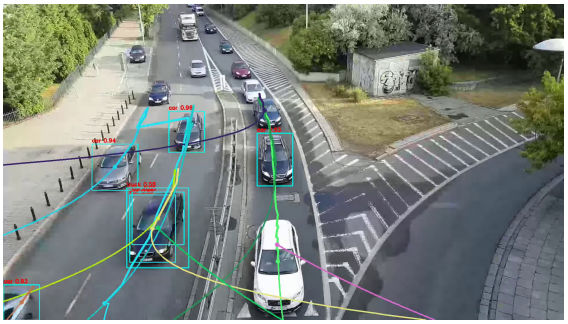
3.2.3.9 Filtern der Objekte

Ein neuronales Netzwerk erkennt ein Objekt nicht nur eindeutig an einer einzigen Stelle und das war es. In Wahrheit werden sehr viele Objekte in jedem einzelnen Frame erkannt. So erkennt das neuronale Netzwerk bei einem Fahrzeug in Wahrheit mehrere Fahrzeuge gleichzeitig. Diese haben alle leicht

unterschiedliche Wahrscheinlichkeit und leicht veränderte Positionen. Nun ist es Aufgabe der Nachverarbeitung das richtige Fahrzeug herauszufiltern. Um diese redundanten Boxen zu entfernen und dadurch ein akkurateres Zählen zu ermöglichen, gibt es die Non-maximum suppression, kurz NMS [Mue21].



(a) Doppeltes Erkennen eines Fahrzeuges



(b) Dreifaches Erkennen eines Fahrzeuges

Um diese Berechnung durchzuführen, wurde bisher die Implementation von Khadas verwendet, da diese für die allgemeine Objekt-Erkennung ausgelegt ist.

Listing 12 Implementation aus Beispiel von Khadas [kha21b]

```
    @staticmethod
    def _nms_boxes(boxes, scores):
        x = boxes[:, 0]
        y = boxes[:, 1]
        w = boxes[:, 2]
        h = boxes[:, 3]

        areas = w * h
        order = scores.argsort()[::-1]

        keep = []
        while order.size > 0:
            i = order[0]
            keep.append(i)

            xx1 = np.maximum(x[i], x[order[1:]])
            yy1 = np.maximum(y[i], y[order[1:]])
            xx2 = np.minimum(x[i] + w[i], x[order[1:]] + w[order[1
             ↪  :]])
            yy2 = np.minimum(y[i] + h[i], y[order[1:]] + h[order[1
             ↪  :]])

            w1 = np.maximum(0.0, xx2 - xx1 + 0.00001)
            h1 = np.maximum(0.0, yy2 - yy1 + 0.00001)
            inter = w1 * h1

            ovr = inter / (areas[i] + areas[order[1:]] - inter)
            inds = np.where(ovr <= YoloPostProcessing.NMS_THRESH)[0
             ↪  ]
            order = order[inds + 1]
        keep = np.array(keep)
        return keep
```

Dies kann auch anhand dieser Visualisierung (Abbildung 3.59) erkannt werden.

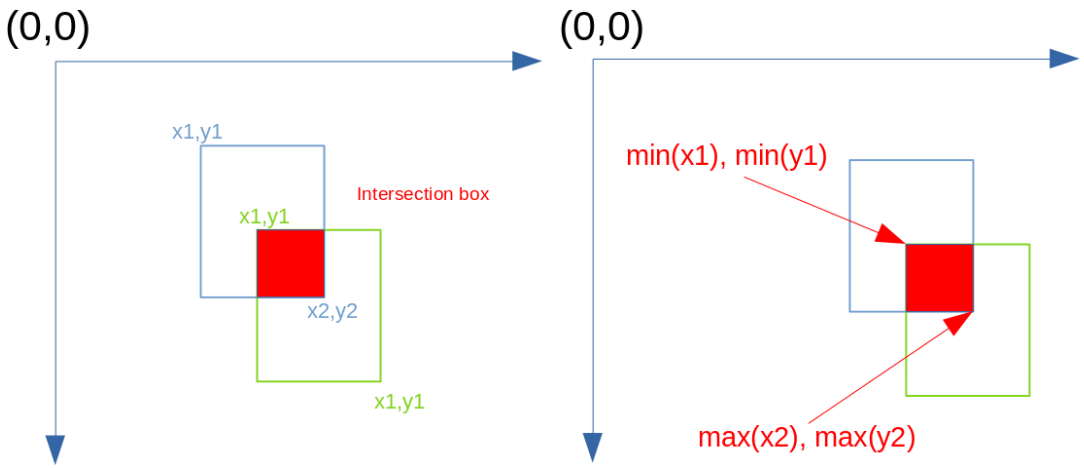


Abbildung 3.59: Algorithmus für NMS [Mue21]

Mit dem Algorithmus von Khadas wird hier die Fläche, in welcher sich die zwei Rechtecke überschneiden, berechnet. Mit `np.where(...)` können dann die Indexe der Rechtecke bestimmt werden, welche einen gewissen Grenzwert überschneiden.

Dieser Grenzwert ist deshalb die einzige Möglichkeit, die Fahrzeuge in diesem Aspekt zu filtern. Jedoch liegt das eigentliche Problem nicht in dem NMS Algorithmus selber, sondern wird es bei der Implementation von Khadas immer nur pro Klasse durchgeführt. Deshalb wird bei Abbildung 3.58a auch gleichzeitig ein Bus und ein Auto erkannt.

Listing 13 NMS pro Klasse [kha21b]

```
def yolov3_post_process(input_data):
    ...
    nboxes, nclasses, nscores = [], [], []
    for c in set(classes):
        inds = np.where(classes == c)
        b = boxes[inds]
        c = classes[inds]
        s = scores[inds]

        keep = nms_boxes(b, s)

        nboxes.append(b[keep])
        nclasses.append(c[keep])
        nscores.append(s[keep])

    ...
    boxes = np.concatenate(nboxes)
    classes = np.concatenate(nclasses)
    scores = np.concatenate(nscores)

    return boxes, classes, scores
```

Dies kann einfach gelöst werden, indem diese Methode nur einmal für alle Objekte ausgeführt wird, anstatt es für jede Klasse zu machen. Zusätzlich sollte es auch etwas die Performance verbessern, da diese Methode nur einmal pro Frame aufgerufen werden muss.

Weiters gibt es noch das Problem, dass manche Fahrzeuge einem falschen Typen zugeordnet werden:

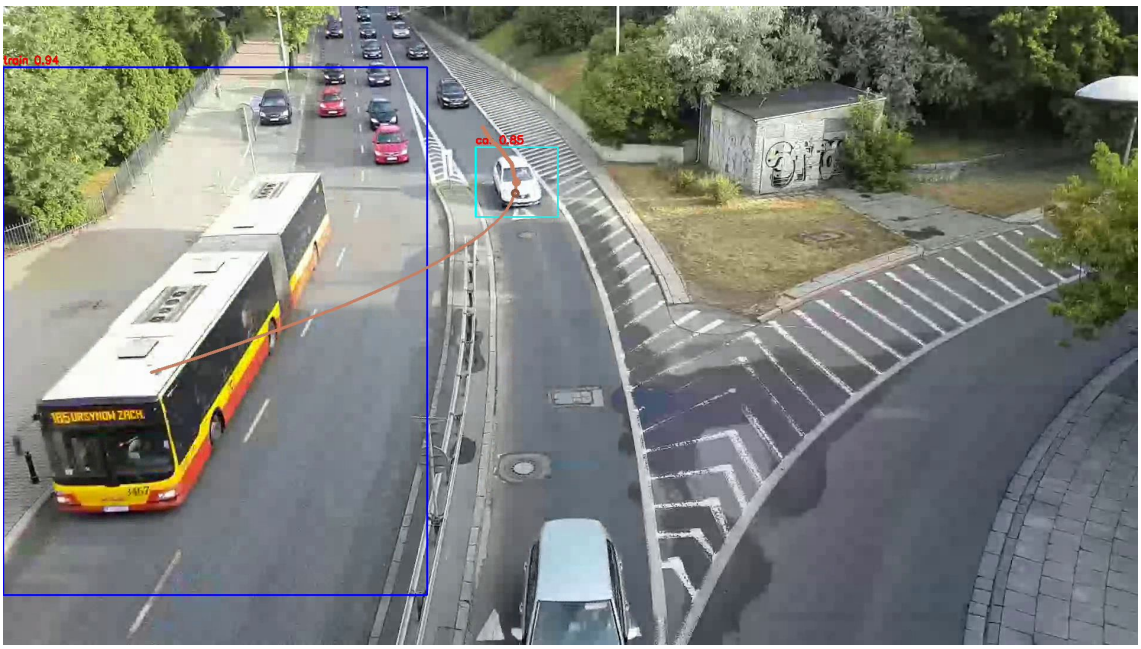


Abbildung 3.60: Bus als Zug zugeordnet

Um solche falschen Erkennungen filtern zu können, werden Objekte, welche nicht einem gewissen Objekttyp entsprechen, gleich ignoriert.

Listing 14 Filtern der Boxen [Pis22b]

```
    @staticmethod
    def _filter_boxes(boxes, box_confidences, box_class_probs):
        box_scores = box_confidences * box_class_probs
        box_classes = np.argmax(box_scores, axis=-1)
        box_class_scores = np.max(box_scores, axis=-1)
        pos = np.where(
            np.logical_and(
                box_class_scores >= YoloPostProcessing.OBJ_THRESH,
                (box_classes == 2)  # car
                | (box_classes == 3)  # motorbike
                | (box_classes == 5)  # bus
                | (box_classes == 7),  # truck
            )
        )

        boxes = boxes[pos]
        classes = box_classes[pos]
        scores = box_class_scores[pos]

        return boxes, classes, scores
```

Hier (Listing 14) werden zuerst alle Objekte herausgefiltert, welche nicht zu einem gewissen Wert den benötigten `box_class_scores` haben. Und danach werden alle Objekte, welchen nicht den Typ `car`, `motorbike`, `bus` oder `truck` haben, herausgefiltert.

Dies beschleunigt dann die Weiterverarbeitung noch einmal und dadurch kann die Bildwiederholungsrate etwas gesteigert werden.

3.2.3.10 Analyse der Performance

Die Performance bzw. die Bildwiederholungsrate ist essenziell für die erfolgreiche Erkennung der Fahrzeuge. Dies liegt daran, dass je größer die Bildwiederholungsrate ist, desto kleiner ist der Abstand der Fahrzeuge zwischen den Frames. Durch den kleineren Abstand werden dann die verschiedene Algorithmen genauer, da die Position nur in unmittelbarer Nähe vorhergesagt werden muss. Dadurch werden die Resultate besser. Im Prinzip kann jedes Python-Programm mit diesem Profiler ausgeführt werden und dabei werden die Daten in `~/output.pstats` gespeichert:

```
$ python3 -m cProfile \
    -o ~/output.pstats \
```

```
3       python/commean.py
```

Mit `gprof2dot` können diese Daten dann in eine übersichtliche Grafik umgewandelt werden.

```
1   $ gprof2dot -f pstats ~/output.pstats | dot -Tpng -o
    ↪  profiling.png
```

Wenn dies nun gemacht wird, dann wird folgende Grafik erstellt:

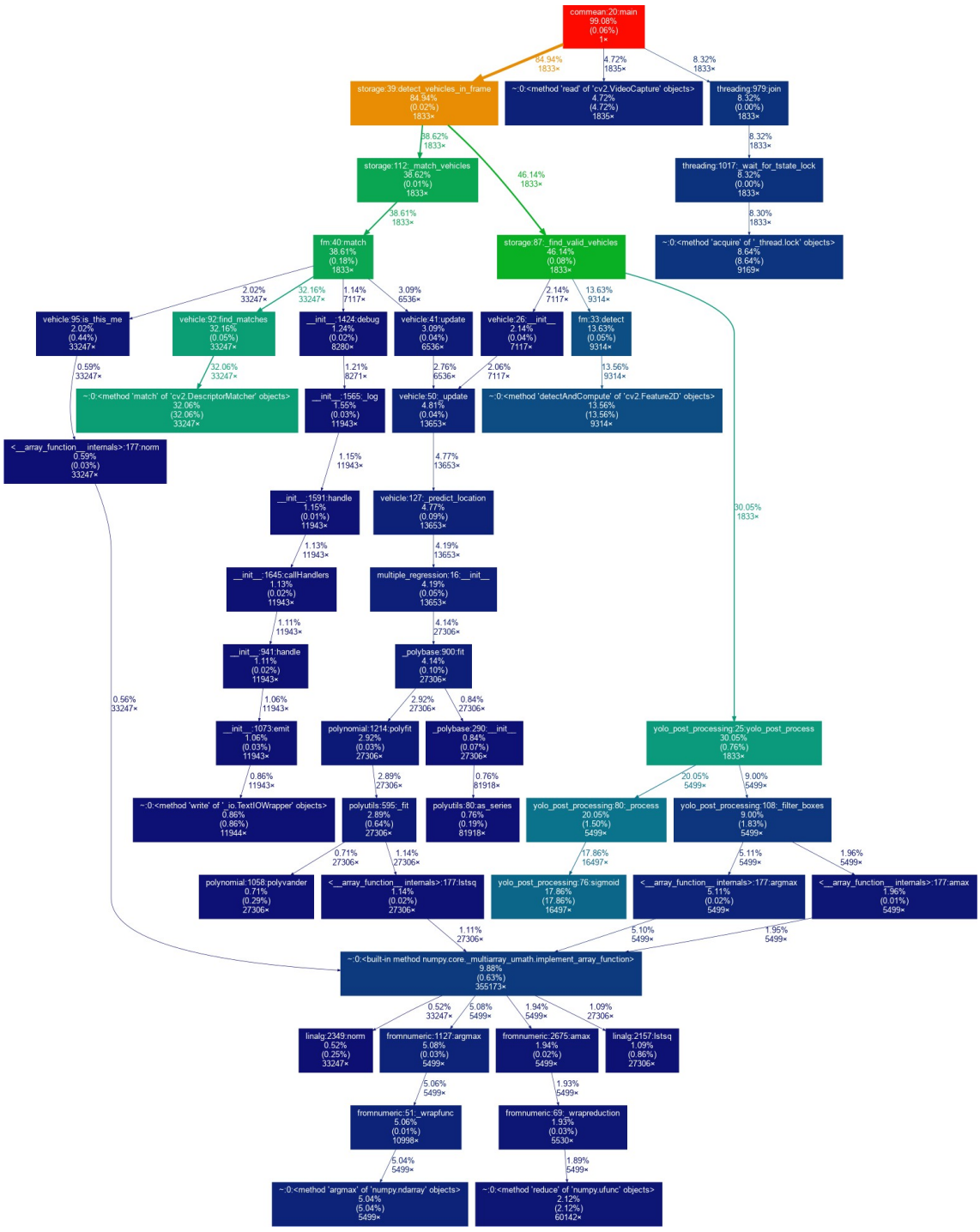


Abbildung 3.61: Profiling des Programmes

Hier kann erkannt werden, dass viel Zeit in das Feature Matching investiert wird. Aber auch das Erkennen der Features mithilfe von ORB benötigt 13,56 % der ausgeführten Zeit.

Weiters benötigt auch die Sigmoid-Funktion für YOLO einiges an Performance.

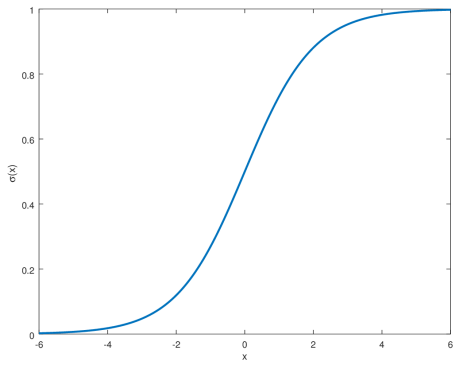


Abbildung 3.62: Sigmoid-Funktion [Hvi18]

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.12}$$

Diese wird als Aktivierungsfunktion verwendet. Dadurch können die Werte aus dem neuronalen Netzwerk so umgewandelt werden, dass sich diese in einem Bereich zwischen 0 und 1 befinden.

Da dies jedoch sehr rechenaufwändig ist und die Werte danach sowieso gefiltert werden, kann diese Funktionen optimiert werden.

Listing 15 Sigmoid Implementationen [Pis22b]

```
1     @staticmethod
2     def sigmoid(x):
3         return 1 / (1 + np.exp(-x))
4
5     @staticmethod
6     def faster_sigmoid(x):
7         return np.where(x > 1, 1 / (1 + np.exp(-x)), 0)
```

Bei der `faster_sigmoid()` Methode werden einfach Werte, welche unter 1 liegen, gleich ignoriert und dadurch kann dort die Berechnung eingespart werden. Jedoch kann diese neue Methode nicht überall verwendet werden. So wird die Position der Box noch weiterhin mit der normalen Sigmoid-Funktion bestimmt. Aber da Objekte unter einer gewissen Wahrscheinlichkeit bei der Klasse und der Wahrscheinlichkeit des Auftretens später gefiltert werden, können für beide die schnellere Sigmoid-Funktion verwendet werden.

Listing 16 Verwendung von `sigmoid` und `faster_sigmoid` [Pis22b]

```
    @staticmethod
    def _process(input, mask, anchors):
        anchors = [anchors[i] for i in mask]
        grid_h, grid_w = map(int, input.shape[0:2])

        box_confidence = YoloPostProcessing.faster_sigmoid(input[
        ↪  ..., 4])
        box_confidence = np.expand_dims(box_confidence, axis=-1)

        box_class_probs = YoloPostProcessing.faster_sigmoid(input[
        ↪  ..., 5:])

        box_xy = YoloPostProcessing.sigmoid(input[..., :2])
        box_wh = np.exp(input[..., 2:4])
        box_wh = box_wh * anchors
        ...
```

3.2.3.11 Schaltung und Board

Hiermit war die Erkennung und Klassifizierung der Fahrzeuge abgeschlossen. Jedoch müssen die gesammelten Daten auch an den Server übertragen werden. Die Übertragung funktioniert sowohl über die WiFi/Ethernetschnittstelle aber auch über LoRa.

Damit das LoRa-Modul auch angesprochen werden kann, wird eine Adapterplatine zwischen dem VIM und dem Modul benötigt.

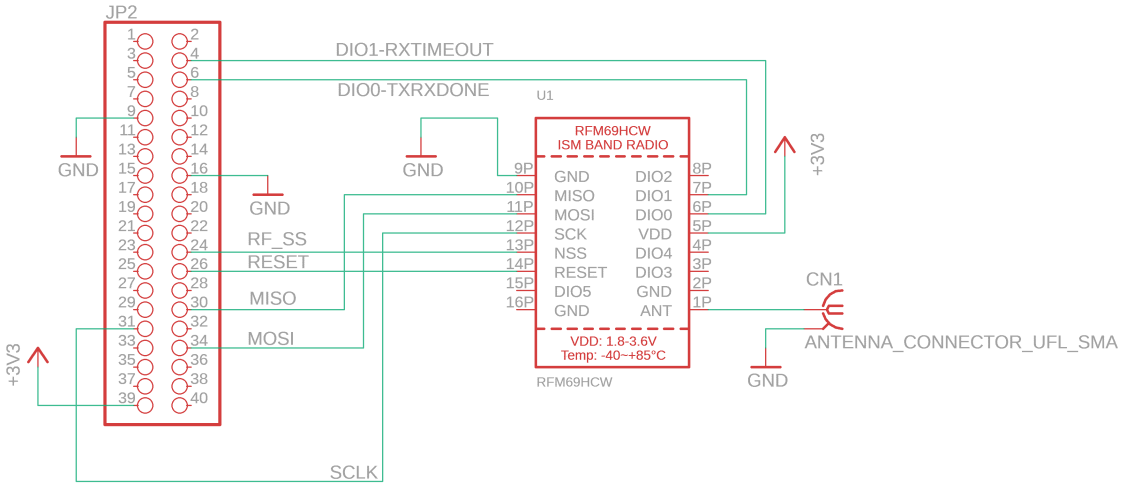


Abbildung 3.63: Schaltplan

Im Prinzip wird einfach das Pinout vom VIM verwendet und das LoRa-Modul angeschlossen. Dafür werden Pins für SPI, Versorgungsspannung, Masse und zwei GPIO-Pins für Signale vom LoRa-Modul benötigt. Außerdem muss die Antenne auch noch angeschlossen werden.

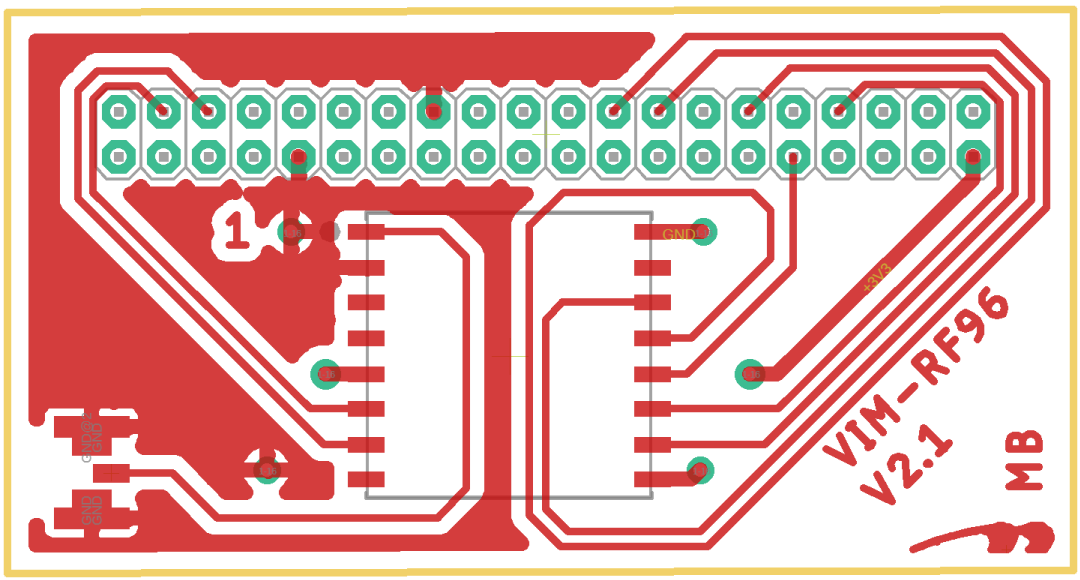


Abbildung 3.64: Board

Zum Ansprechen des LoRa-Moduls wird die LMIC-Library (LoraWAN-MAC-in-C) **??** verwendet, welche vom Arduino-Framework für den VIM geported wurde.

KAPITEL

4 Zusammenfassung

4.1 Projektmanagement

4.1.1 Scrum

Scrum ist ein Vorgehensmodell in der agilen Softwareentwicklung, welches für das Projektmanagement verwendet wird. Bei Scrum wird in kleinen Teams an einem Projekt bzw. Projektbereich gearbeitet. Die gesamte Arbeit wird zuerst einmal in grobe Teile unterteilt, sogenannte Epics. Dies würden in unserer Diplomarbeit dem Frontend, Backend, der Bilderkennung etc. entsprechen.

Weiters werden dann für die Epics (User) Stories erstellt. Diese sollen neue Funktionen für die jeweiligen Epics sein, wobei das ganze vom Endkunden betrachtet wird. Dadurch soll es den Entwicklern*innen ermöglicht werden, sich gezielt an konkrete Wünschen zu orientierten, anstatt die ganze Zeit an einem großen Epic zu arbeiten.

Um diese User Story zu erreichen, müssen einzelne Tasks/Issues erstellt werden, welche dann schließlich von den Entwickler*innen bearbeitet werden. Sind dann bei einer User Story alle Tasks abgeschlossen, kann auch die User Story als abgeschlossen betrachtet werden. Dasselbe gilt auch analog für Epics. Weiters wird der Projektzeitraum in Sprints aufgeteilt. Diese Sprints sind meist eine oder zwei Wochen lang. Pro Sprint werden dann Tasks zugeordnet, welche in dem gewünschten Zeitraum abgearbeitet werden sollen. Damit gut eingeschätzt werden kann, wie viele Tasks abgearbeitet werden können, muss der Arbeitsaufwand für jeden Task bestimmt werden.

Am Ende jedes Sprints wird der abgeschlossene Sprint betrachtet und es wird versucht, den ganzen Prozess weiter zu optimieren. So kann dann nach jedem Sprint besser eingeschätzt werden, was alles in einem Sprint umgesetzt werden kann. Dadurch kann immer besser vorhergesagt werden, wann welche Funktion implementiert werden kann.

4.1.2 Iterationen

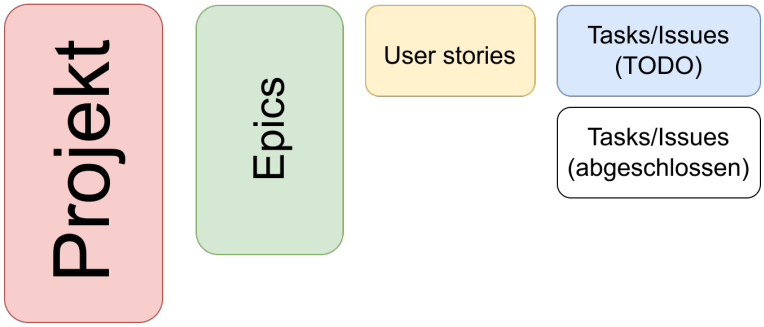


Abbildung 4.1: Milestone Story Board – Legende

4.1.2.1 Iteration 1

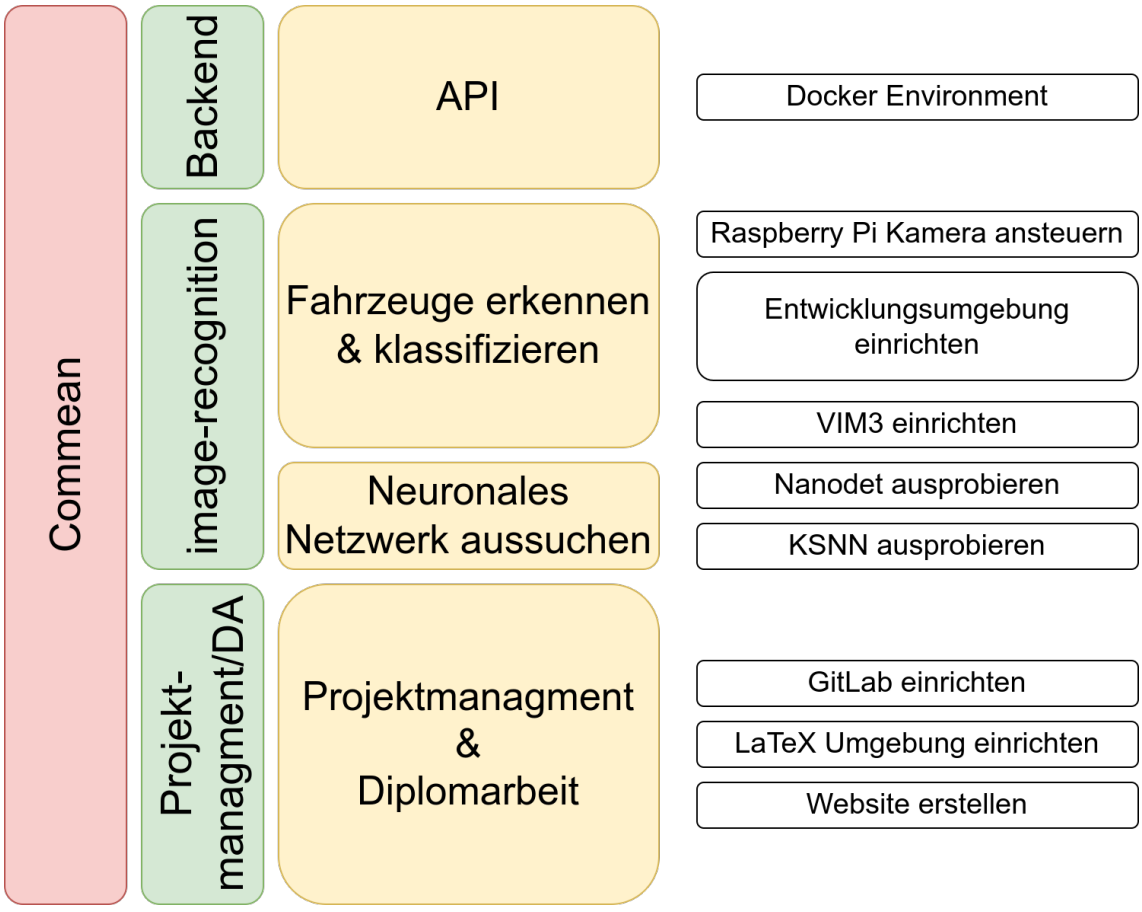


Abbildung 4.2: Milestone Story Board – Ende Iteration 1

In der ersten Iteration wurde das ganze Projekt erst langsam eingerichtet. So wurden hier teilweise Frameworks für die Bilderkennung ausprobiert, um diese untereinander zu vergleichen. Ebenso mussten allgemeine Tätigkeiten, welche zur Diplomarbeit gehören, durchgeführt werden. So wurde GitLab

und die dazugehörigen Labels für die einzelnen Tasks erstellt. Da diese Diplomarbeit auch in LaTeX geschrieben wird, musste diese Umgebung bei den einzelnen Teammitgliedern eingerichtet werden. Auch weitere elementare Aufgaben, wie das Ansprechen der Raspberry Pi Kamera bzw. die Docker-Umgebung für das Backend, wurden in dieser Iteration behandelt.

4.1.2.2 Iteration 2

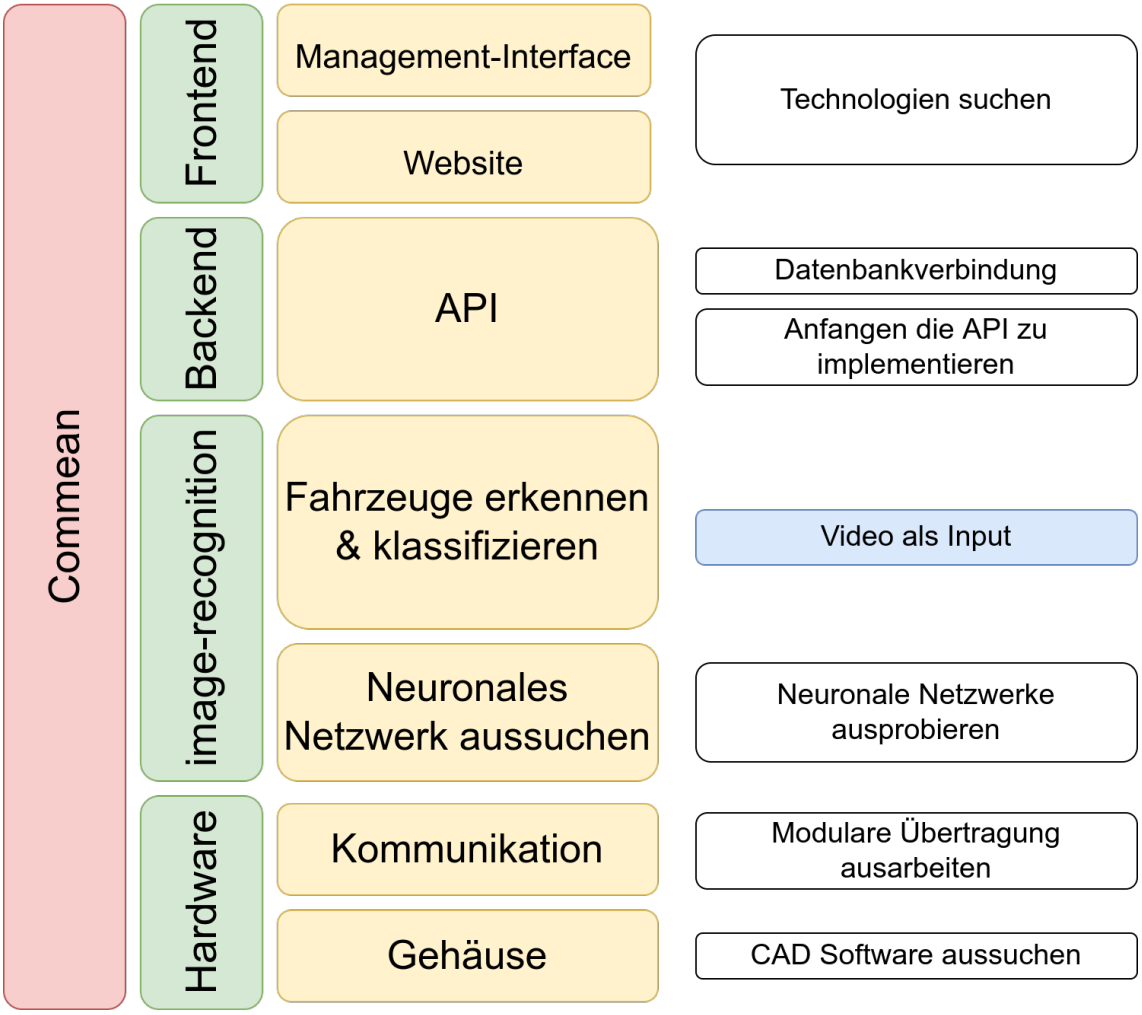


Abbildung 4.3: Milestone Story Board – Ende Iteration 2

In dieser Iteration wurden die Grundsteine für die weitere Entwicklung des Projekts gelegt. So wurden zuerst allgemeine Technologien, welche verwendet werden können, gesucht, wie das Framework für die Bilderkennung oder das CAD-Programm für das Gehäuse. Auch die Verbindung zwischen dem Backend und der Datenbank funktioniert nun.

4.1.2.3 Iteration 3

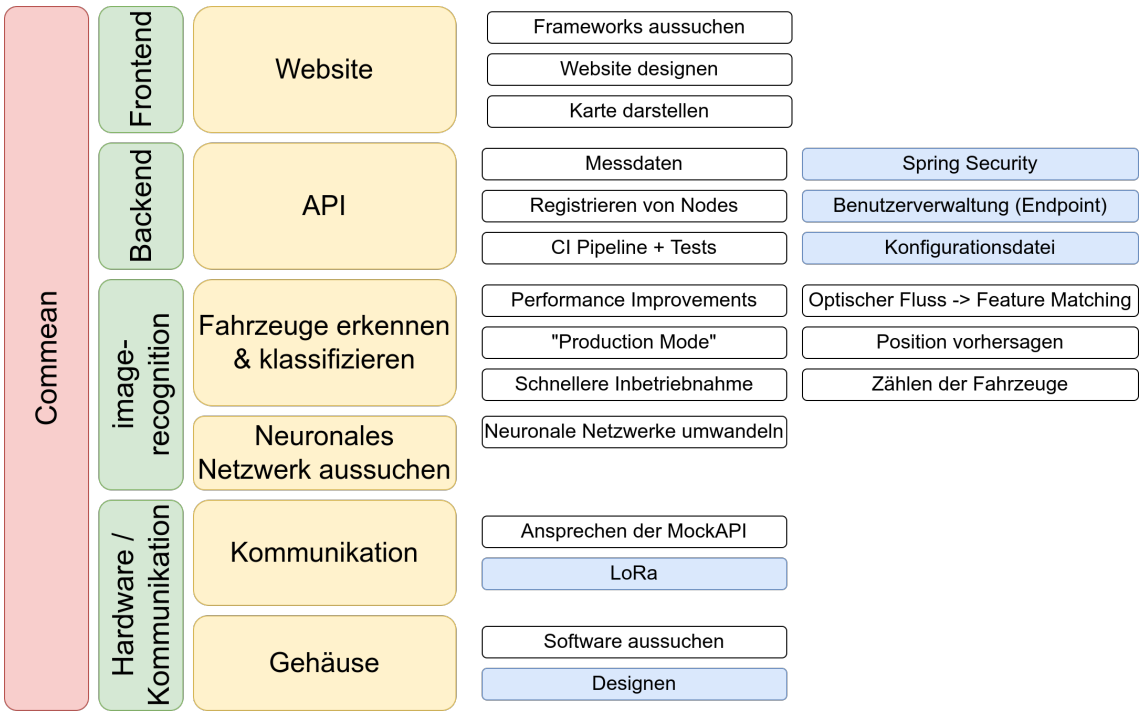


Abbildung 4.4: Milestone Story Board – Ende Iteration 3

In Iteration 3 wurde dann ein großer Teil der Diplomarbeit erledigt. So bekamen hier alle Teile des Projektes das erste Mal eine wirkliche Funktion und konnten auch unabhängig voneinander verwendet werden. Das Frontend kann nun angezeigt werden, das Backend funktioniert bereits zu einem großen Teil und die Bilderkennung kann schon Fahrzeuge in einer gewissen Toleranz erfolgreich zählen. Auch bei der Hardware und Kommunikation gibt es Fortschritte, da nun bereits eine Test-API angesprochen werden kann und Fusion/EAGLE soll für die Hardware verwendet werden.

4.1.2.4 Iteration 4

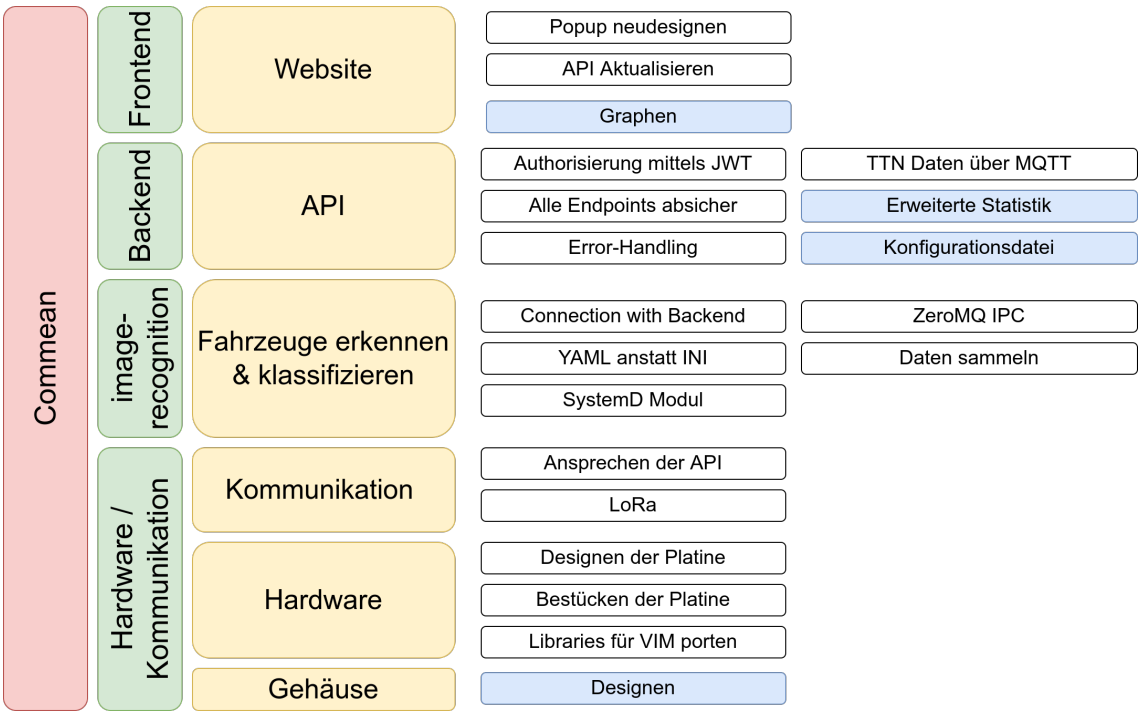


Abbildung 4.5: Milestone Story Board – Ende Iteration 4

In der vorletzten Iteration ging es darum, das ganze Projekt nun endlich langsam zusammenzustellen. So ist nun JWT im Backend implementiert, was es der Bilderkennung ermöglicht, mittels REST-API die Daten an den Server zu schicken. Außerdem ist nun die Verbindung zwischen dem LoRa-Modul und dem SBC hergestellt, wodurch auch eine Anbindung mittels *The Things Network* möglich war. Damit auch das Backend mit dem *The Things Network* kommunizieren kann, wird ein MQTT-Server verwendet, welcher auch bereitgestellt wird. Dadurch können die Daten aus dem *The Things Network* mittels MQTT-Server abgefangen werden.

4.1.2.5 Übersicht

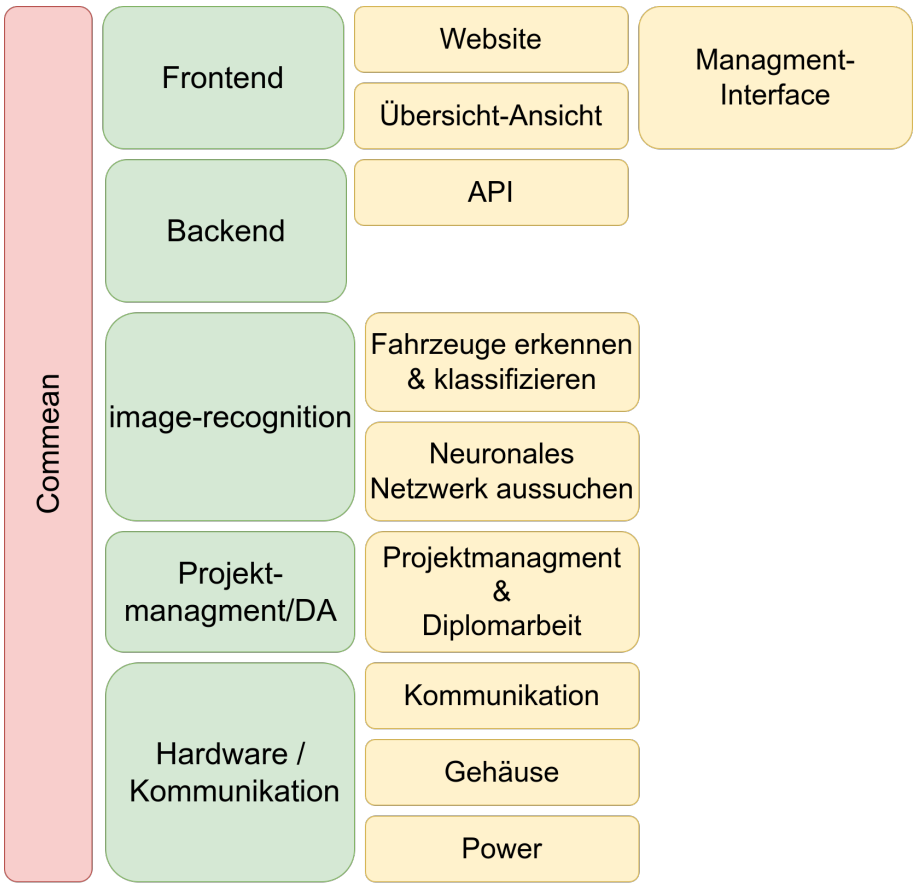


Abbildung 4.6: Milestone Story Board – Übersicht

4.2 Inbetriebnahme

Um den Prototyp in Betrieb zu nehmen, muss nur das Adapterboard mit dem LoRa-Modul auf den VIM gesteckt werden. Zusätzlich bietet es sich noch an, eine Antenne mit SMA-Anschluss anzuschließen, um weitere Distanzen bis zum nächsten LoRa-Gateway zu ermöglichen.

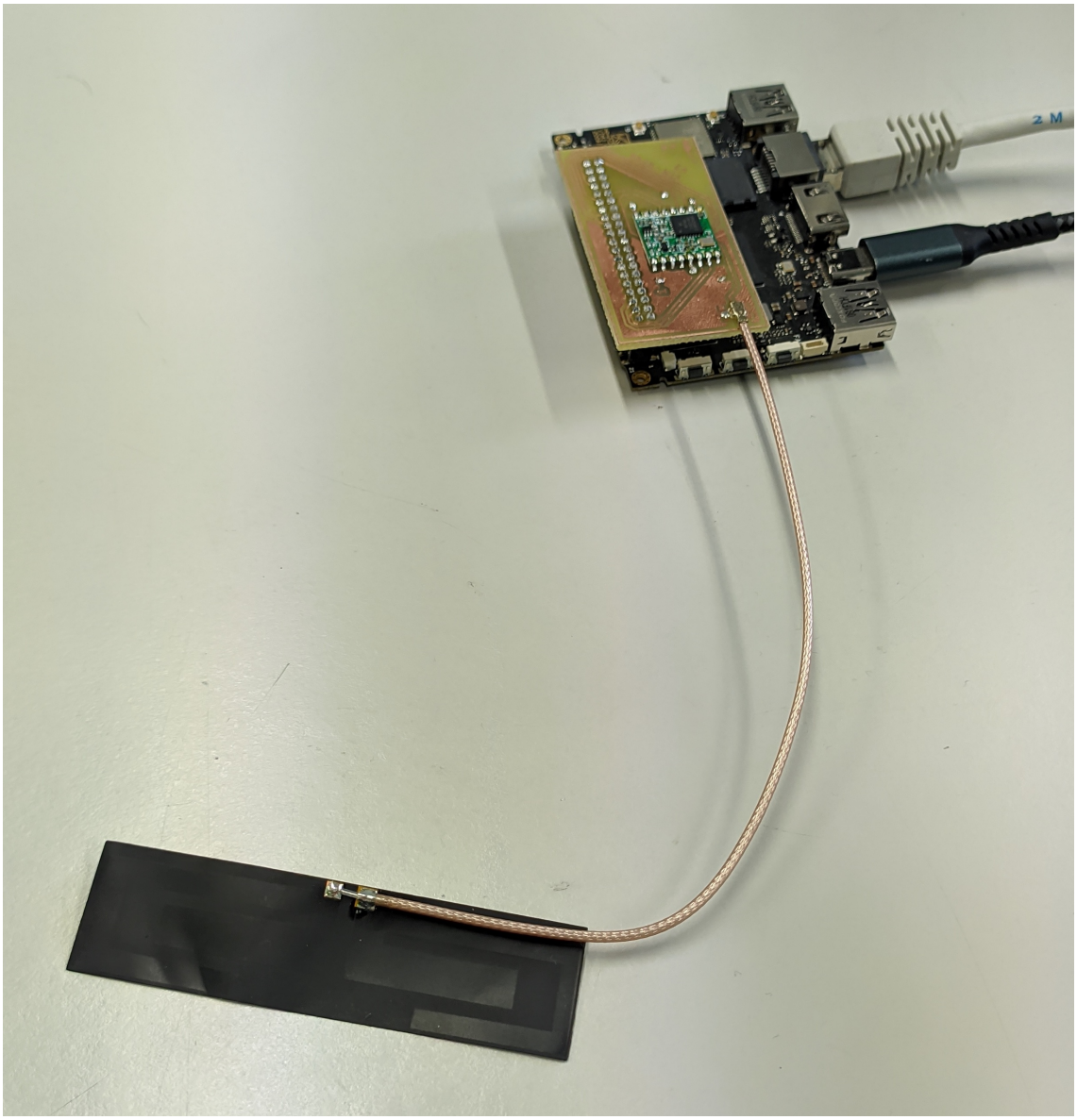


Abbildung 4.7: Prototyp (mit Antenne)

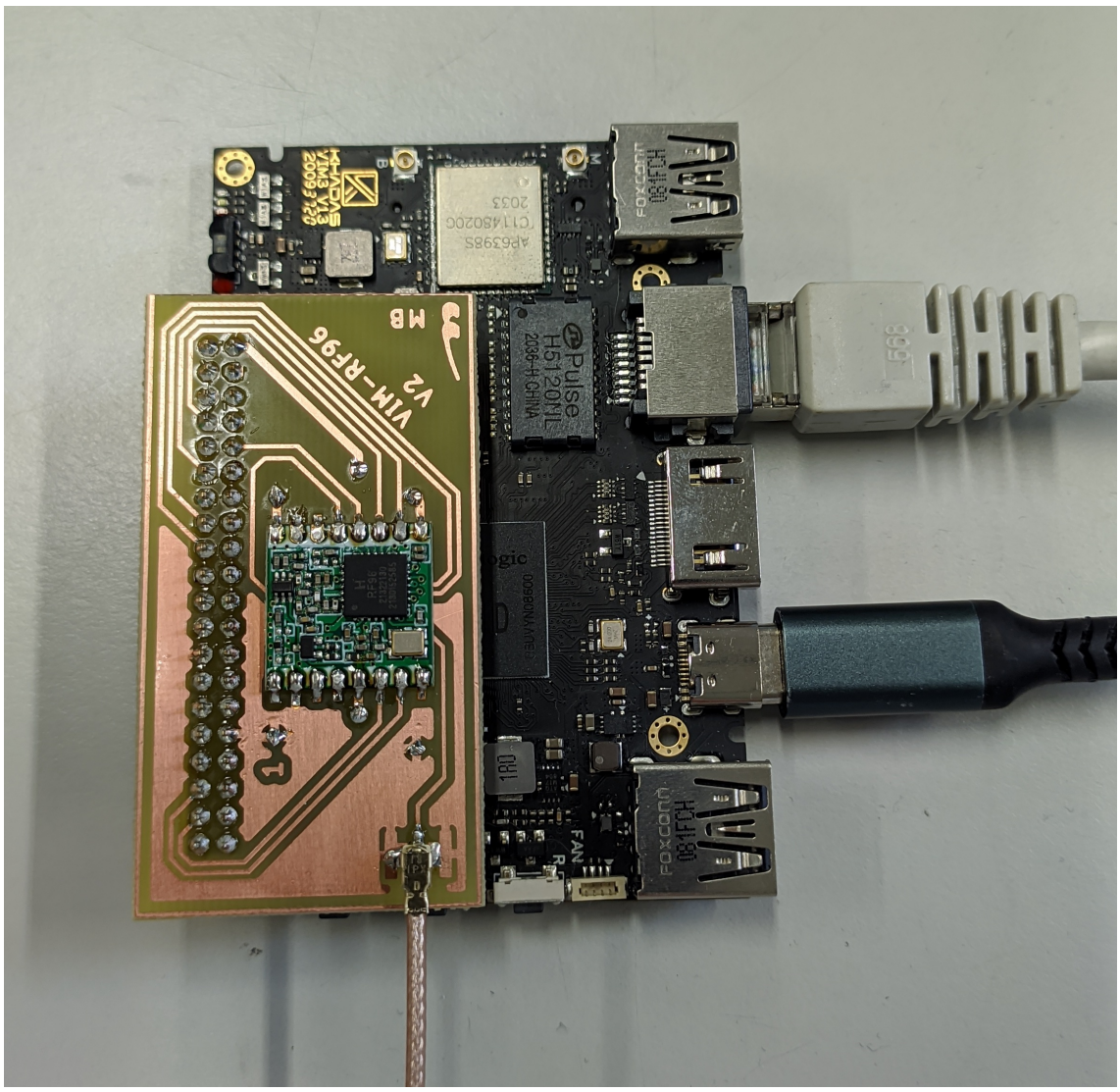


Abbildung 4.8: Prototyp (ohne Antenne)

Nach der Inbetriebnahme des Prototyps sendet dieser automatisch Daten über das Internet bzw. LoRa an das Backend. Auf dem folgenden Foto ist ersichtlich, wie die Daten an das Things Network geschickt und vom Server verarbeitet werden. Anschließend werden die Daten in der Datenbank gespeichert und können über die Webseite abgerufen werden. Wie auf folgenden Bild zu sehen ist.

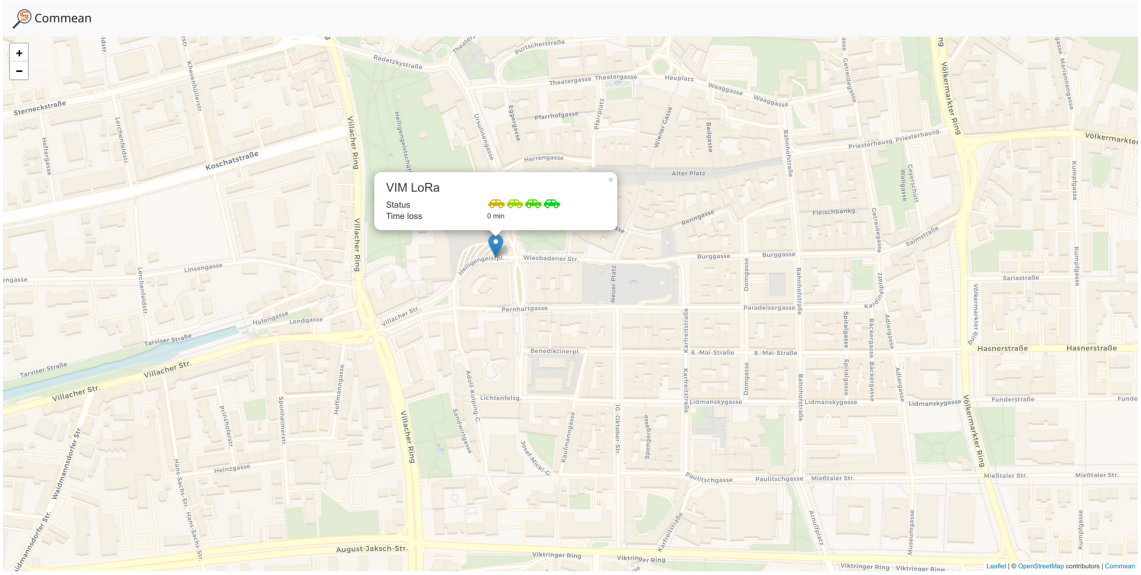


Abbildung 4.9: Webseite mit Pop-up

4.3 Arbeitszeitnachweis

4.3.1 Luca Nachbar

Datum	Anfang	Ende	Beschreibung	Dauer in h	In der Schule
14/07/21	09:30:00	11:00:00	Magistrat kontaktieren	1.50	
14/07/21	11:00:00	12:30:00	Termin mit Sandra Wassermann	1.50	
20/07/21	06:30:00	09:00:00	ÖVP kontaktieren; Einen Termin ausmachen	2.50	
20/07/21	11:30:00	13:00:00	Termin mit Dr. Julia Löschnig	1.50	
28/07/21	17:00:00	22:00:00	GitLab Struktur besprechen und einrichten	5.00	
08/09/21	08:00:00	11:10:00	Einführung in GitLab und Besprechung der Herangehensweise der DA	3.17	
14/09/21	19:00:00	22:30:00	Diplomarbeit Aufgabenstellung	3.50	
15/09/21	16:00:00	21:00:00	DiplomarbeitGrundlagen und Methoden	5.00	
16/09/21	13:04:59	16:40:00	Sync mit Projektbetreuer	3.58	x
21/09/21	07:50:00	11:25:00	Sync mit Team	3.58	x

22/09/21	13:04:59	16:40:00	Leaflet.js und GeoJson testen	3.58	x
29/09/21	18:35:00	21:19:00	Statische Webseite mit leaflet	2.73	
30/09/21	13:05:00	16:40:00	Verarbeitung von GeoJson Daten	3.58	x
05/10/21	07:50:00	11:25:00	Anzeigen von Daten auf Karte	3.58	x
07/10/21	13:05:00	16:40:00	Laden der Karte verbessern	3.58	x
12/10/21	07:50:00	11:25:00	Spring Projekt aufsetzen	3.58	x
14/10/21	13:05:00	16:40:00	JPA Entitys erstellen für Nodes und Kreuzungen	3.58	x
19/10/21	07:50:00	11:25:00	Datenbank verbindung mittels JPA und Spring Data	3.58	x
21/10/21	13:05:00	16:40:00	Verbindung mit InfluxDB	3.58	x
26/10/21	20:02:00	23:59:00	Wechsle der Datenbank von MariaDB und InfluxDB auf TimescaleDB	3.95	
27/10/21	19:30:00	23:24:00	Testen von TimescaleDB	3.90	
28/10/21	19:20:00	23:59:00	Fehlerbehebung von Fehlern entstanden durch Datenbankwechsel	4.65	
03/11/21	17:00:00	21:00:00	Service Objekte für Datenbank	4.00	
03/11/21	13:04:59	16:20:00	Objekt für Messdaten erstellen	3.25	x
06/11/21	14:30:00	21:34:44	Junit tests für Services und Repositories	7.08	
16/11/21	07:50:00	11:25:00	Präsentation Iteration 1 Vorbereitung	3.58	x
16/11/21	18:09:57	20:00:00	Präsentation Iteration 1 Vorbereitung	1.83	
18/11/21	13:05:00	16:40:00	Präsentation	3.58	x
23/11/21	07:50:00	11:25:00	Präsentation	3.58	x
25/11/21	13:05:00	16:40:00	Spring MVC Endpoints Theorie	3.58	x
30/11/21	07:50:00	11:25:00	API für Kamera Nodes auf Frontend	3.58	x
02/12/21	13:04:59	16:40:00	API Dokumenation und Error handling	3.58	x

06/12/21	16:30:00	21:30:00	Integrieren von Leaflet mit VueJS	5.00	
07/12/21	07:50:00	11:25:00	Einlesen von GeoJson in Frontend	3.58	x
09/12/21	13:04:59	16:40:00	Über Tileserver informieren,dynamische popupss Leaflet	3.58	x
11/12/21	14:20:00	18:00:00	Tileserver aufsetzten und Testen	3.67	
12/12/21	09:40:00	11:55:00	CartoCSS mit Mapnik	2.25	
13/12/21	18:00:00	19:40:00	Leaflet dynamische Popups	1.67	
14/12/21	07:50:00	11:25:00	Popups Leaflet,Messdaten API verbessern	3.58	x
15/12/21	18:40:00	23:00:00	Popups Leaflet,Messdaten über API abfragen	4.33	
16/12/21	13:04:59	16:40:00	Aufbau des Frontends in VueJs	3.58	
21/12/21	7:50:00	11:25:00	Einbindung von Leaflet in VueJs	3.58	
27/12/21	13:32:00	15:00:00	Error handling bei API Fehlern	1.47	
28/12/21	14:00:00	18:35:00	Popups für Fehler	4.58	
05/01/22	17:32:00	23:34:00	Live Daten aus der Datenbank auf Karte anzeigen	6.03	
06/01/22	13:00:00	18:00:00	Optimierung der API anfragen	5.00	
11/01/22	7:50:00	11:25:00	Spring Security Theorie	3.58	x
13/01/22	13:04:59	16:40:00	Ausarbeitung Nodes und Apikeys	3.58	x
17/01/22	9:30:00	13:34:00	Implementierung der API-Key generation, Spring Security Filter	4.07	
18/01/22	7:50:00	11:25:00	Implementierung von API Key abfrage	3.58	x
20/01/22	13:04:59	16:40:00	Frontend Umstieg auf Axios	3.58	x
22/01/22	17:30:00	22:45:00	Spring CORS für Endpoints, Konfigurationsdatei	5.25	
25/01/22	7:50:00	11:25:00	Spring Program über Dateien Konfigurierbar machen	3.58	x
27/01/22	13:49:59	16:40:00	Umgebungsvariablen als Konfiguration	2.83	x

29/01/22	14:00:00	17:30:00	Programm als Docker-Container	3.50	
01/02/22	7:50:00	11:25:00	API Key für einzelne Benutzer	3.58	x
03/02/22	13:05:00	16:40:00	Entfernung von API-Keys	3.58	x
05/02/22	14:00:00	17:00:00	Spring Security und JWT Theorie	3.00	
08/02/22	07:50:00	11:25:00	Verbesserung der Integration-Tests und Anmelde-Flow	3.58	x
09/02/22	18:00:00	20:15:00	Benutzer Modell erarbeiten	2.25	
10/02/22	13:05:00	16:40:00	Implementierung von Benutzern	3.58	x
12/02/22	13:20:00	18:00:00	Filter und Security Config	4.67	
15/02/22	08:30:00	14:25:00	JWT Generation und Signierung	5.92	
16/02/22	14:00:00	16:35:00	Änderung der Endpoints	2.58	
19/02/22	17:00:00	22:15:00	Bugfixes in Security	5.25	
22/02/22	07:50:00	11:25:00	Finalisierung JWT Security	3.58	x
22/02/22	17:00:00	20:00:00	Integration Test mit JWT, verbesserung der API	3.00	
23/02/22	19:00:00	21:10:00	Merge JWT in main, Statistische erhebung	2.17	
24/02/22	13:05:00	16:40:00	Id von Measurement aus jwt auslesen	3.58	x
27/02/22	18:00:00	21:00:00	API genauer Dokumentieren	3.00	
01/03/22	07:50:00	10:20:00	Null Pointer Exeption bei API beheben	2.50	x
03/03/22	13:05:00	16:40:00	TimeScaleDB Continius agregate demo	3.58	x
05/03/22	16:00:00	19:00:00	Timescale DoW m für statistik	3.00	
08/03/22	07:50:00	13:05:00	TTN API, umstieg auf MQTT	5.25	
10/03/22	13:05:00	16:40:00	Installation von Messstation	3.58	x
15/03/22	07:50:00	14:05:00	Einbindugn von TTN via MQTT	6.25	x
17/03/22	13:05:00	16:40:00	Itregations Test Protyp ? Backend (Internet)	3.58	x

19/03/22	14:30:00	23:59:00	Fehlerkorrektur der API	6.25	
22/03/22	07:50:00	14:05:00	Itregations Test Protyp ? Backend (LoRa)	3.58	x
24/03/22	13:05:00	16:40:00	Itregations Test Protyp ? Backend (LoRa)	6.25	x
27/03/22	14:00:00	23:59:00	Fehlerkorrektur LoRa/MQTT	3.58	
29/03/22	07:50:00	14:05:00	Deployment von Back- und Frontend auf Server	6.25	x
31/03/22	13:05:00	16:40:00	CORS Fehler beheben	3.58	x
			Gesamtarbeitszeit (Ist)	308.47	152.75
			Soll	320.00	160.00
			Offen (Ist-Soll)	11.53	7.25

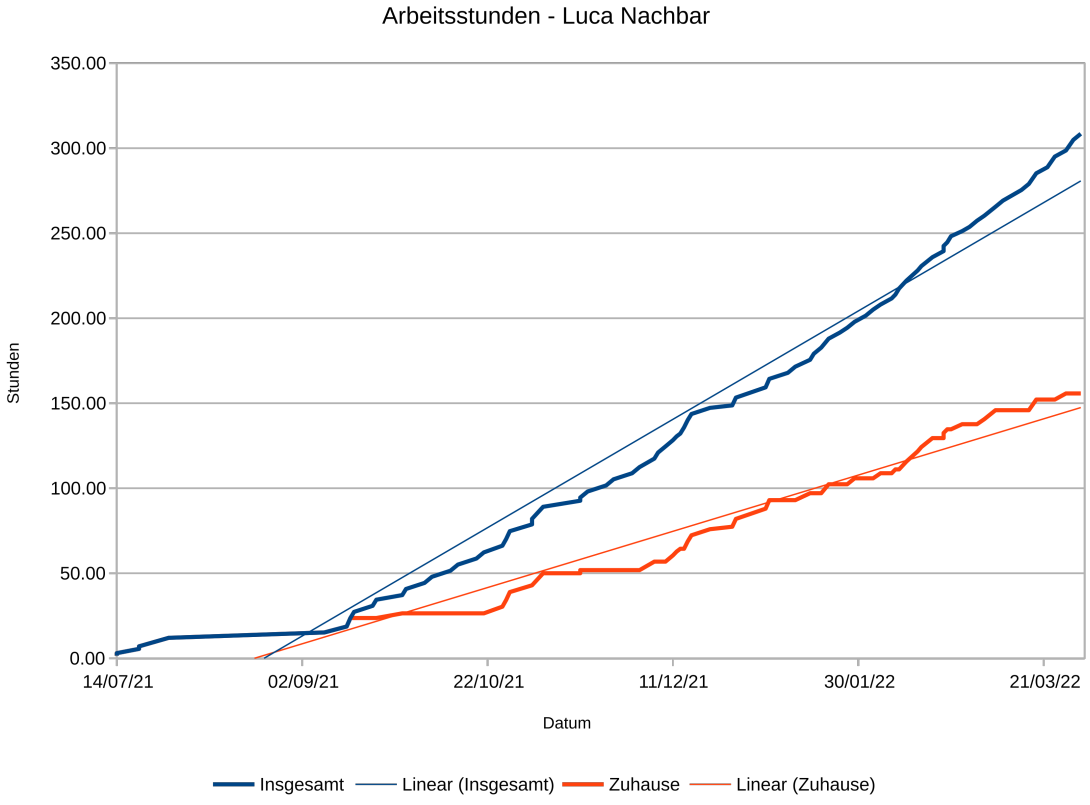


Abbildung 4.10: Arbeitsstundenübersicht - Luca Nachbar

4.3.2 Stefan Pisjak

Datum	Anfang	Ende	Beschreibung	Dauer in h	In der Schule
14/07/21	09:30:00	11:00:00	Magistrat kontaktieren	1.50	
14/07/21	11:00:00	12:30:00	Termin mit Sandra Wassermann	1.50	

20/07/21	06:30:00	09:00:00	ÖVP kontaktieren; Einen Termin ausmachen; ?	2.50
20/07/21	09:00:00	10:00:00	Grafiken und Diagramme erstellen	1.00
20/07/21	11:30:00	13:00:00	Termin mit Dr. Julia Löschnig	1.50
28/07/21	17:00:00	22:00:00	GitLab Struktur besprechen und einrichten	5.00
30/07/21	11:00:00	12:00:00	Dateien auf GitLab anstatt Nextcloud bereitstellen	1.00
05/08/21	08:00:00	08:15:00	Präsentationsvorlage suchen	0.25
12/08/21	18:30:00	19:15:00	Präsentation polishen	0.75
12/08/21	21:30:00	22:10:00	Alte Aktivitäten in GitLab nachtragen	0.67
17/08/21	17:00:00	19:00:00	Sich über Neuronale Netzwerke informieren	2.00
17/08/21	19:00:00	00:00:00	Sich über die DSGVO informieren	5.00
03/09/21	21:00:00	00:00:00	YOLO ausprobieren	3.00
05/09/21	11:00:00	13:20:00	Dokumente ausfüllen	2.33
05/09/21	18:00:00	20:00:00	YOLOv4 ausprobieren	2.00
05/09/21	20:00:00	22:30:00	Raspberry PI aufsetzen; YOLO auf diesem Ausprobieren	2.50
05/09/21	22:30:00	00:00:00	YOLO mit GPU	1.50
06/09/21	11:00:00	13:45:00	YOLO optimieren; YOLO mit niedriger Auflösung	2.75
06/09/21	18:30:00	21:00:00	YOLO für Videos (& niedriger Auflösung)	2.50
07/09/21	11:00:00	13:00:00	LaTeX Vorlage suchen und importieren & Editor suchen	2.00
07/09/21	13:00:00	19:30:00	LaTeX Fehler bei der Vorlage ausbessern und allgemein ausfüllen	6.50
07/09/21	22:00:00	22:40:00	Alte Meetings nachtragen	0.67
08/09/21	08:30:00	12:45:00	Einführung in GitLab und Besprechung der Herangehensweise der DA	4.25

08/09/21	19:45:00	22:45:00	JabRef anschauen; Wie man damit Sachen in LaTeX referenzieren kann	3.00	
09/09/21	18:45:00	19:45:00	Glossaries in LaTeX Vorlage fixen	1.00	
10/09/21	10:15:00	10:30:00	Wikipage für Glossaries erstellen	0.25	
10/09/21	14:45:00	16:30:00	Alternativen zu YOLO suchen	1.75	
10/09/21	16:30:00	16:40:00	Weiteres Vorgehen besprechen (weiter YOLO oder etwas anderes ausprobieren)	0.17	
10/09/21	16:40:00	17:20:00	Alte Issues lablen & Gitlab Gruppe aufräumen	0.67	
10/09/21	17:20:00	17:30:00	Raspberry PI aufsetzen	0.17	
11/09/21	21:40:00	00:00:00	Nanodet & ncnn ausprobieren	2.33	
12/09/21	15:00:00	22:30:00	Diplomarbeit: Aufgabenstellung & Anfang der Grundlagen und Methoden	7.50	
13/09/21	14:50:00	19:00:00	Diplomarbeit: Erweitern der Neuronalen Netzwerke	4.17	
14/09/21	13:20:00	14:10:00	Luca Nachbar LaTeX erklären	0.83	
14/09/21	14:10:00	18:00:00	Diplomarbeit: Backbone des Neuronalen Netzwerkes	3.83	
15/09/21	14:00:00	14:34:59	Maximilian Pezdirnik: LaTeX erklären	0.58	
15/09/21	14:34:59	17:34:59	Diplomarbeit: Neck/Head eines Neuronalen Netzwerks beschreiben	3.00	
16/09/21	13:04:59	16:40:00	Sync mit Projektbetreuer	3.58	x
18/09/21	12:40:00	13:00:00	Grundlagen zur API klären	0.33	
21/09/21	07:50:00	11:25:00	Sync mit Team	3.58	x
23/09/21	13:04:59	16:40:00	Projektstrukturierung mit Scrum	3.58	x

23/09/21	20:20:00	21:20:00	Versuchen die GitLab Projekte zu schulinternen Instanz zu migrieren	1.00	
23/09/21	21:20:00	22:30:00	Nachtragen der Laboreinheiten und Sheet überarbeiten	1.17	
28/09/21	07:50:00	09:35:00	Genehmigung überarbeiten	1.75	x
28/09/21	09:35:00	09:50:00	Sprints definieren	0.25	x
28/09/21	09:50:00	11:25:00	Präsentation vorbereiten	1.58	x
28/09/21	14:14:00	15:15:00	Scoped issues und änderung der Issue States	1.02	
28/09/21	15:15:00	15:55:00	API definieren	0.67	
28/09/21	15:30:00	15:45:00	Präsentation erweitern	0.25	
28/09/21	22:50:00	23:20:00	Flow diagram hinzufügen	0.50	
29/09/21	13:04:59	16:25:00	Nanodet mit PyTorch ausprobieren	3.33	x
29/09/21	18:00:00	21:50:00	Nanodet mit PyTorch ausprobieren	3.83	
04/10/21	15:00:00	17:00:00	User Stories erstellen	2.00	
05/10/21	07:47:00	10:26:00	Prepare Sprints	2.65	x
05/10/21	10:26:00	11:25:00	VIM3 vs Google Coral anschauen	0.98	x
06/10/21	13:04:59	16:25:00	Raspberry Pi Kamera	3.33	x
06/10/21	17:15:00	18:00:00	Projekte zu schulinterner Instanz migrieren	0.75	
06/10/21	19:45:00	21:00:00	Raspberry Pi Kamera	1.25	
07/10/21	13:04:59	16:25:00		3.33	x
07/10/21	19:00:00	21:00:00	Dokumentation über Rpi Kamera	2.00	
08/10/21	06:50:00	07:00:00	LaTeX Styling	0.17	
10/10/21	12:05:00	13:00:00	RPi Kamera am Rpi 4	0.92	
12/10/21	07:50:00	11:10:00	NCNN auf aarch64 (Versuchen)	3.33	x
12/10/21	18:55:00	20:41:00	NCNN auf aarch64 (Erfolg); NCNN Python mit Desktop CPU & GPU	1.77	
13/10/21	13:04:59	16:25:00	Test Programm für Rpi Kamera	3.33	x
13/10/21	18:00:00	18:40:00		0.67	x
14/10/21	13:04:59	16:04:59	KMG Meeting	3.00	x

14/10/21	16:04:59	16:40:00	User Stories relinken	0.58	
15/10/21	14:45:00	16:00:00	Doku über SSHFS	1.25	
17/10/21	14:50:00	18:30:00		3.67	
18/10/21	15:00:00	16:40:00	Einleitung / 1. Kapitel schreiben	1.67	
18/10/21	16:40:00	18:05:00	Diagramme überarbeiten	1.42	
19/10/21	07:50:00	11:10:00	Setup VIM3	3.33	
19/10/21	16:00:00	18:25:00		2.42	
20/10/21	13:30:00	13:36:00	Document Setting Up of VIM3	0.10	
20/10/21	16:40:00	17:17:00	Flow diagram hinzufügen (SW)	0.62	
21/10/21	13:04:59	16:25:00	Trying out NPU for VIM3	3.33	x
27/10/21	10:30:00	12:00:00	Grundlagen und Methoden schreiben	1.50	
28/10/21	07:10:00	09:30:00	CI Pipeline for thesis	2.33	
28/10/21	15:00:00	16:00:00	Document Setting Up of VIM3	1.00	
31/10/21	13:00:00	16:30:00	Setup VIM3	3.50	
01/11/21	17:05:00	21:30:00		4.42	
02/11/21	09:15:00	12:30:00		3.25	
04/11/21	13:04:59	16:25:00		3.33	x
08/11/21	11:25:00	12:15:00	Ethische Bedenken	0.83	
09/11/21	07:50:00	11:10:00	Projekt-Website erstellen	3.33	x
09/11/21	18:10:00	19:15:00	KSNN - Clean up Code	1.08	
10/11/21	13:04:59	16:25:00	KSNN ? Frames	3.33	
11/11/21	13:04:59	14:00:00	Website feinschliff	0.92	x
11/11/21	14:00:00	16:25:00	KSNN ? Fix bugs	2.42	x
14/11/21	07:55:00	10:45:00	KSNN ? Webcam	2.83	
16/11/21	07:50:00	11:10:00	Scripts erstellen um schneller VIM in der Schule zu benutzen	3.33	x
16/11/21	17:45:00	18:45:00	Präsentation für 2. Iteration vorbei	1.00	
16/11/21	18:45:00	19:35:00	Flow Diagramme updaten	0.83	
16/11/21	19:35:00	22:25:00	Präsentation + Iteration 3 vorbereiten	2.83	
17/11/21	01:00:00	01:30:00		0.50	
17/11/21	13:04:59	14:25:00	SystemD Modul für VIM3	1.33	
17/11/21	14:25:00	16:25:00	Flow Diagramme	2.00	

18/11/21	19:10:00	19:55:00	KSNN ? Video File (& blocked wegen IP-Cam)	0.75	
18/11/21	13:04:59	16:04:59	Präsentationen	3.00	x
18/11/21	15:35:00	16:20:00	Poster als Vektorgrafik	0.75	x
18/11/21	17:00:00	17:15:00		0.25	
20/11/21	11:25:00	12:15:00	Code Cleanup	0.83	
20/11/21	14:50:00	15:30:00		0.67	
21/11/21	12:25:00	13:15:00	Investigate instability issues	0.83	
21/11/21	13:15:00	13:49:59	Update KSNN	0.58	
21/11/21	14:15:00	16:25:00	YoloV3 für KSNN builden	2.17	
23/11/21	07:50:00	09:30:00	innovation<AT>school	1.67	x
23/11/21	09:30:00	09:45:00	Poster fixen	0.25	x
23/11/21	09:45:00	09:55:00	Flow Diagram	0.17	x
23/11/21	09:55:00	11:10:00	Investigate instability issues	1.25	x
23/11/21	13:49:59	14:20:00		0.50	
24/11/21	13:04:59	15:30:00		2.42	
24/11/21	13:04:59	16:25:00	Brainstorming ? Lucas Kanade Optischer Flow	3.33	x
30/11/21	07:50:00	11:10:00	Optischen Flow anzeigen	3.33	x
30/11/21	13:45:00	15:40:00	Optischen Fluss analysieren	1.92	
30/11/21	16:00:00	16:20:00	Poster zu Repository hinzufügen	0.33	
30/11/21	16:49:59	19:45:00	Algorithmus zur Erkennung implementieren	2.92	
01/12/21	09:30:00	10:00:00		0.50	
02/12/21	13:55:00	14:40:00	Vehicle Klasse erstellen und einbinden	0.75	
03/12/21	11:00:00	12:19:59	Nächste Position vorhersehen	1.33	
07/12/21	07:50:00	11:10:00	Farneback optimieren	3.33	x
09/12/21	13:04:59	13:45:00	Threads	0.67	x
09/12/21	13:45:00	16:25:00	Fahrzeuge tracken	2.67	x
09/12/21	20:00:00	23:40:00		3.67	
11/12/21	14:50:00	18:30:00		3.67	
11/12/21	22:00:00	23:20:00	"Production Mode" hinzufügen	1.33	
13/12/21	18:00:00	20:00:00	Daten für Backend generieren	2.00	

14/12/21	07:50:00	11:10:00	Programm dokumentieren	3.33	x
14/12/21	13:49:59	16:00:00	pylint einrichten	2.17	
14/12/21	16:00:00	16:10:00	Abstürzen des Programmes beheben	0.17	
15/12/21	13:04:59	16:25:00	Nachholen der Dokumentation abschließen	3.33	
16/12/21	13:05:00	16:25:00	Programm bei neuem Video optimieren	3.33	x
16/12/21	20:15:00	23:55:00		3.67	
17/12/21	21:15:00	23:00:00	Feature matching anstatt Optischen Fluss	1.75	
18/12/21	13:05:00	18:25:00		5.33	
18/12/21	22:20:00	24:00:00	Feature matching fix using wrong object	1.67	
19/12/21	09:20:00	12:20:00	Feature matching Vektoren zeichnen	3.00	
19/12/21	18:40:00	20:00:00	Feature matching "scoring algorithm" verbessern	1.33	
20/12/21	17:15:00	18:15:00		1.00	
20/12/21	18:15:00	23:45:00	KSNN auf 1.2 updaten (Versuchen)	5.50	
21/12/21	07:50:00	08:10:00	Popup designen	0.33	
21/12/21	08:10:00	11:10:00	KSNN NN versuchen zu konvertieren	3.00	
21/12/21	15:00:00	16:30:00		1.50	
22/12/21	13:05:00	16:25:00		3.33	
22/12/21	18:00:00	19:00:00		1.00	
24/12/21	10:35:00	13:35:00		3.00	
26/12/21	10:25:00	10:40:00	Scoring Algorithmus verbessern	0.25	
26/12/21	11:00:00	16:10:00	Regression verwenden um die voraussichtliche Position vorherzusagen	5.17	
27/12/21	10:10:00	10:50:00	Technik fürs Leben anmelden	0.67	
27/12/21	10:50:00	12:30:00	Geschwindigkeit des Fahrzeuges berücksichtigen	1.67	
28/12/21	07:50:00	15:00:00	Regression nicht X per Y, sondern X & Y per Zeit	7.17	
29/12/21	08:50:00	10:50:00	Code optimieren	2.00	
31/12/21	10:30:00	10:50:00	KSNN Updaten	0.33	

31/12/21	10:50:00	13:20:00	Convert tool weiter debuggen	2.50	
03/01/22	10:30:00	10:50:00		0.33	
03/01/22	10:45:00	13:00:00	Glossaries fixen	2.25	
03/01/22	13:00:00	13:40:00	Dokumentation über Neuronale Netzwerke	0.67	
03/01/22	15:00:00	16:30:00	Tabelle in LaTeX übertragen	1.50	
04/01/22	16:30:00	17:50:00	Geschichte von Yolo beschreiben	1.33	
04/01/22	18:15:00	18:30:00	Python Dataclasses	0.25	
05/01/22	10:25:00	10:35:00	Bei "Technik fürs Leben" anmelden	0.17	
05/01/22	12:40:00	16:45:00	Darknet dokumentieren	4.08	
06/01/22	10:25:00	12:30:00	KSNN dokumentieren	2.08	
06/01/22	13:55:00	15:40:00	Python Pakete dokumentieren	1.75	
07/01/22	10:25:00	12:00:00	VNC-Server dokumentieren	1.58	
07/01/22	15:40:00	16:20:00	Optischen Fluss dokumentieren	0.67	
08/01/22	10:15:00	12:35:00	Programmcode für Optischen Fluss dokumentieren	2.33	
09/01/22	11:45:00	13:20:00	Optischen Fluss dokumentieren	1.58	
11/01/22	07:50:00	11:10:00	Algorithmus zum Tracken verbessern	3.33	x
11/01/22	23:45:00	00:00:00		0.25	
12/01/22	13:05:00	16:25:00		3.33	
12/01/22	22:15:00	23:30:00		1.25	
16/01/22	18:45:00	19:25:00		0.67	
16/01/22	19:25:00	19:55:00	Jugend Innovativ Anmeldung vorbereiten	0.50	
16/01/22	19:55:00	21:15:00	Einführung ins Front- & Backend	1.33	
17/01/22	08:10:00	10:40:00	Datenbank und Skripts fürs Backend	2.50	
17/01/22	21:00:00	23:20:00		0.00	
18/01/22	07:50:00	11:10:00	CI/CD Pipeline	3.33	x
19/01/22	13:05:00	16:25:00		3.33	
19/01/22	21:30:00	23:00:00	Fahrzeuge zählen	1.50	
20/01/22	13:05:00	14:05:00	Algorithmus zum Tracken verbessern	1.00	x
20/01/22	14:05:00	16:25:00	Autos am Rande des Bilder ignorieren	2.33	x

21/01/22	15:20:00	21:00:00	Various Algorithm Improvements	5.67	
22/01/22	07:00:00	08:00:00	Pezdirnik Teil von DA mit main mergen	1.00	
22/01/22	09:50:00	11:00:00	Kurzfassung, Abstract, ? schreiben	1.17	
23/01/22	12:30:00	13:30:00	Feature matching dokumentieren	1.00	
23/01/22	16:00:00	17:00:00	Minted für Code Formattierung	1.00	
23/01/22	17:00:00	21:00:00	Feature matching weiter dokumentieren	4.00	
23/01/22	21:40:00	22:40:00	Neuen Scoring Algorithmus dokumentieren und DA abgeben	1.00	
24/01/22	21:35:00	23:15:00	Postprocessing überarbeiten	1.67	
25/01/22	07:30:00	07:50:00	Verbessern des Tracken	0.33	
25/01/22	07:50:00	11:00:00		3.17	x
25/01/22	11:00:00	11:20:00	Python Dataclasses	0.33	
25/01/22	20:00:00	20:30:00		0.50	
26/01/22	07:20:00	07:35:00	Fix logger	0.25	
26/01/22	07:35:00	09:45:00	Markov Chains anschauen	2.17	
27/01/22	13:05:00	16:25:00		3.33	x
29/01/22	07:05:00	07:55:00	Config vom USB-Stick einlesen	0.83	
29/01/22	13:00:00	16:30:00	Tracking Linie hinzufügen	3.50	
30/01/22	11:10:00	12:00:00	Fix Vehicle Counter	0.83	
31/01/22	20:00:00	21:00:00	Präsentation für 3. Iteration vorbereiten	1.00	
01/02/22	07:50:00	10:50:00	Präsentation	3.00	x
01/02/22	10:50:00	11:15:00	Tracking Linie hinzufügen	0.42	x
01/02/22	16:50:00	21:50:00	Code optimieren	5.00	
02/02/22	13:05:00	14:20:00	Falsches Zählen ausbessern	1.25	
02/02/22	14:20:00	16:25:00	Zählalgorithmus verbessern	2.08	
03/02/22	13:05:00	15:05:00	LoRa Modul ansprechen	2.00	x
03/02/22	15:05:00	16:25:00	Winkel besser berechnen	1.33	x
03/02/22	17:35:00	17:50:00		0.25	

03/02/22	18:00:00	21:00:00	Mit LoRa TTN versuchen anzusprechen	3.00	
04/02/22	07:50:00	09:10:00	Node unter Windows aufsetzen	1.33	
05/02/22	08:00:00	09:25:00	LoRaWan versuchen aufzusetzen	1.42	
06/02/22	07:15:00	07:30:00	Präsentation hochladen	0.25	
08/02/22	07:50:00	10:20:00	Verbindung mit TTN herstellen	2.50	x
08/02/22	10:20:00	11:15:00	TTN Api ansprechen	0.92	x
08/02/22	15:10:00	16:00:00	Fehlende Hardware bestellen	0.83	
08/02/22	16:00:00	16:40:00	Skripts fixen	0.67	
08/02/22	16:45:00	17:15:00	Frontend aufsetzen	0.50	
09/02/22	14:55:00	18:30:00	Popup in Component auslagern (versuchen)	3.58	
10/02/22	11:35:00	13:05:00	Popup wirklich in Component auslagern	1.50	
10/02/22	13:05:00	15:00:00	Popup designen	1.92	x
10/02/22	15:00:00	16:40:00	Board designen	1.67	x
11/02/22	12:35:00	14:20:00	Popup ? Grafik automatisch zuschneiden	1.75	
12/02/22	09:30:00	10:30:00	Platine bestellen	1.00	
12/02/22	13:35:00	13:50:00	Platine abbestellen	0.25	
12/02/22	13:50:00	16:35:00	Library für VIM versuchen zu kompilen	2.75	
13/02/22	16:45:00	18:00:00	Verbindung zwischen VIM und LoRa Modul versuchen herzustellen	1.25	
14/02/22	11:25:00	14:00:00		2.58	
15/02/22	07:50:00	12:00:00		4.17	
15/02/22	16:00:00	18:30:00		2.50	
17/02/22	12:00:00	15:15:00	Anfangen des Portes der LMIC auf den VIM	3.25	
17/02/22	15:15:00	21:35:00	LMIC von MCC für VIM porten	6.33	
18/02/22	12:00:00	15:30:00		3.50	
22/02/22	12:00:00	14:05:00		2.08	
23/02/22	08:20:00	09:25:00		1.08	
23/02/22	13:00:00	18:00:00		5.00	
23/02/22	18:00:00	19:25:00	Referenzen fixen	1.42	
23/02/22	22:30:00	23:30:00	Fehler aus DA ausbessern	1.00	
24/02/22	08:00:00	12:30:00		4.50	

24/02/22	13:00:00	19:30:00	ZeroMQ: Verbindung zwischen LMIC und Commean	6.50	
25/02/22	08:00:00	09:00:00		1.00	
26/02/22	14:30:00	20:30:00	ZeroMQ: Protokoll erweitern	6.00	
27/02/22	13:35:00	19:35:00	ZeroMQ: LMIC Verbindung dynamisch erkennen	6.00	
28/02/22	11:30:00	15:30:00	Projektbericht für Jugend Innovativ schreiben	4.00	
28/02/22	16:30:00	20:40:00	Daten generieren	4.17	
28/02/22	23:00:00	23:35:00	YAML durch INI ersetzen (Config)	0.58	
01/03/22	07:50:00	09:20:00		1.50	
01/03/22	09:20:00	11:30:00	Kommunikation über lokales Interface ermöglichen	2.17	
01/03/22	14:25:00	16:20:00	Thesis überarbeiten	1.92	
02/03/22	07:50:00	12:00:00	Python Programm installieren und SystemD Modul dafür schreiben	4.17	
02/03/22	12:00:00	12:25:00	Für LMIC SystemD Modul schreiben	0.42	
02/03/22	14:15:00	15:15:00	Grafiken aktualisieren	1.00	
02/03/22	15:15:00	16:00:00	URLs zu Literaturverzeichnis hinzufügen	0.75	
02/03/22	16:40:00	21:00:00	Thesis: Arbeitszeitnachweis (versuchen) einzufügen	4.33	
02/03/22	22:00:00	23:30:00	Projektbericht für Jugend Innovativ finalisieren	1.50	
03/03/22	13:00:00	17:00:00	Board designen	4.00	
04/03/22	09:25:00	09:35:00	Platine beschriften	0.17	
08/03/22	07:50:00	09:50:00	TTN in der Schule austesten	2.00	x
08/03/22	09:50:00	09:55:00	Fehler bei LMIC ausbessern	0.08	x
08/03/22	09:55:00	11:25:00	Platine bestellen	1.50	
09/03/22	13:05:00	16:30:00	Gehäuse anfangen zu designen	3.42	
10/03/22	18:30:00	19:30:00	Python Memory Leak debuggen	1.00	
11/03/22	14:40:00	17:10:00		2.50	

12/03/22	19:15:00	19:45:00		0.50	
13/03/22	06:55:00	07:15:00		0.33	
13/03/22	07:15:00	08:15:00	Testprogramm für Sample LoRa Daten	1.00	
13/03/22	14:00:00	15:00:00	Python Memory Leak (temp.) fixen	1.00	
14/03/22	08:00:00	08:05:00	Platine abholen	0.08	
14/03/22	14:40:00	15:15:00	Memory Leak Alternativen ausprobieren	0.58	
14/03/22	15:15:00	15:30:00	Testdaten für LoRa generieren	0.25	
14/03/22	15:50:00	17:40:00	Präsentation für Ende 4. Sprint vorbereiten	1.83	
14/03/22	19:10:00	20:00:00	API Kommunikation besprechen	0.83	
15/03/22	07:50:00	11:25:00	Prototyp Inbetriebnahme	3.58	x
15/03/22	15:40:00	17:20:00	Platine (auf Fehler) analysieren	1.67	
15/03/22	19:30:00	20:30:00	Milestone Story Board in DA einfügen/nachzeichnen	1.00	
16/03/22	12:00:00	13:05:00	Bei Prototyp nach Fehler suchen	1.08	
16/03/22	13:05:00	14:20:00		1.25	
16/03/22	14:20:00	15:30:00	Kommunikation mit Backend	1.17	
16/03/22	15:30:00	16:40:00	Tracking Lines über Config-File hinzufügen	1.17	
17/03/22	11:50:00	12:20:00	Präsentation überarbeiten	0.50	
17/03/22	13:05:00	14:45:00	Tracking Lines über Config-File hinzufügen	1.67	x
17/03/22	14:45:00	16:25:00	Präsentation Ende 4. Iteration	1.67	x
17/03/22	21:50:00	22:40:00	Print-Statements aufräumen	0.83	
17/03/22	22:40:00	23:45:00	Netzwerkverbindung widerstandsfähiger machen	1.08	
19/03/22	07:20:00	08:25:00	Speicherplatz am VIM frei machen	1.08	
20/03/22	10:00:00	13:20:00	Argumente übersichtlicher parsen	3.33	
20/03/22	13:45:00	15:40:00	Zusammenfassung schreiben	1.92	

20/03/22	21:50:00	22:45:00	Arbeitsaufzeichnung aktualisieren	0.92	
21/03/22	19:25:00	19:30:00	EAGLE Files auf Gitlab hochladen	0.08	
21/03/22	19:30:00	19:40:00	Wettbewerb Dateien auf Gitlab hochladen	0.17	
21/03/22	20:00:00	20:20:00	Diagramme für neue Statistikberechnung zeichnen	0.33	
22/03/22	07:40:00	07:55:00	Story fürs Tracken der Fahrzeuge auf einzelnen Trackinglinien erstellen	0.25	
22/03/22	08:00:00	08:10:00	Netzwerkverbindung auf eigenem Thread	0.17	
22/03/22	08:55:00	09:20:00	KMG Präsentation vorbereiten	0.42	x
23/03/22	07:20:00	07:25:00	Payload definieren (API v2)	0.08	
23/03/22	13:05:00	13:20:00	Payload übertragen	0.25	
23/03/22	13:20:00	14:20:00	Netzwerk multithreaden: Analyse	1.00	
23/03/22	14:20:00	16:25:00	Threads: Memory Leak analysieren	2.08	
24/03/22	13:05:00	16:00:00	KMG Meeting	2.92	x
24/03/22	16:00:00	16:40:00	Produkte zusammensuchen	0.67	x
24/03/22	17:00:00	17:15:00	Gewerbe Anmeldung nachlesen	0.25	
25/03/22	13:05:00	15:05:00	E-Mail und Produktliste erstellen und senden	2.00	
27/03/22	20:30:00	21:00:00	Python Threads debuggen	0.50	
28/03/22	13:00:00	13:25:00	Bei FH Kärnten Wettbewerb teilnehmen	0.42	
28/03/22	13:25:00	13:45:00	Infos zu KWF Förderantrag recherchieren	0.33	
30/03/22	06:55:00	07:05:00	Cbridge-Verbindung automatisch wiederherstellen	0.17	
30/03/22	11:45:00	12:15:00	Open-Source Unternehmen ? möglicher Businessplan	0.50	
30/03/22	13:05:00	13:15:00	netidee: Anmeldung	0.17	
30/03/22	13:15:00	15:30:00	Open-Source Finanzierungen suchen	2.25	

			Experts Group		
			OpenSource		
30/03/22	15:30:00	16:40:00	kontaktieren	1.17	
31/03/22	13:05:00	14:35:00	Kamerabild flippen	1.50	x
			Diplomarbeit		
31/03/22	14:35:00	16:40:00	überarbeiten	2.08	x
			Diplomarbeit		
31/03/22	17:25:00	18:00:00	überarbeiten	0.58	
			Dynamische		
31/03/22	18:00:00	19:00:00	Erkennung der Webcam	1.00	
			Diplomarbeit		
02/04/22	11:00:00	13:45:00	überarbeiten	2.75	
			Gesamtarbeitszeit (Ist)	590.38	128.88
			Soll	320.00	160.00
			Offen (Ist-Soll)	−270.38	31.12

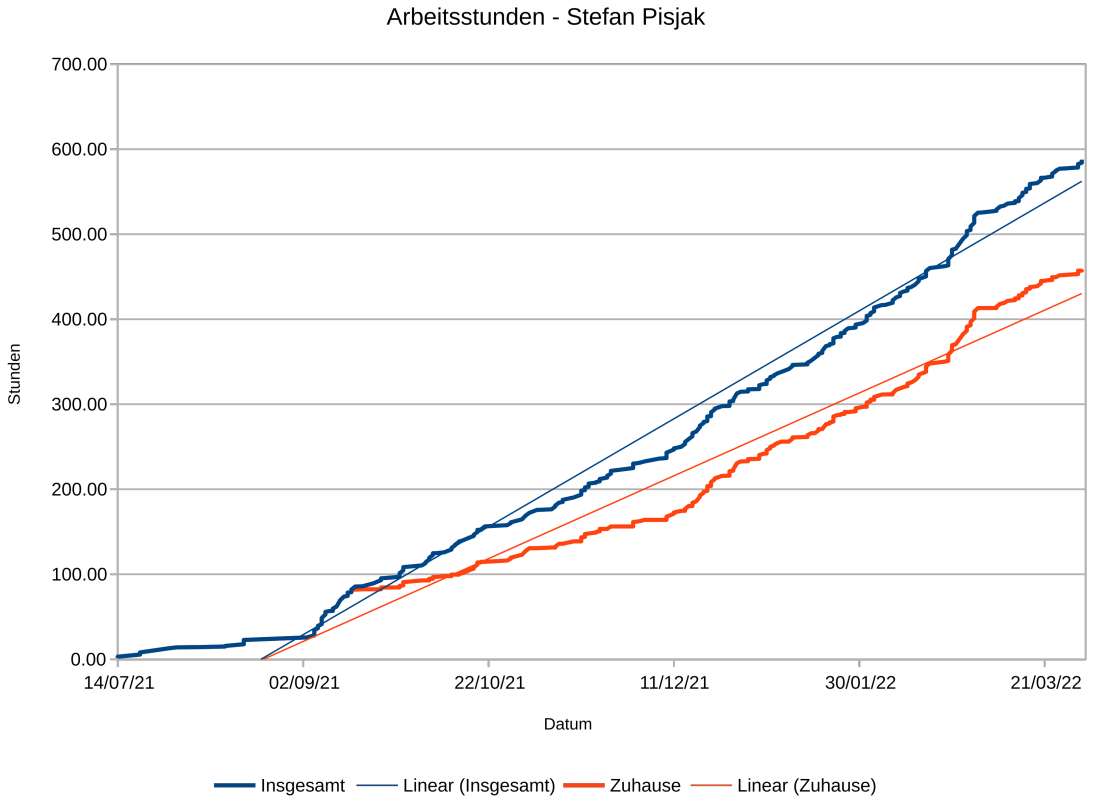


Abbildung 4.11: Arbeitsstundenübersicht - Stefan Pisjak

4.4 Wettbewerbe

Da die Diplomarbeit innovative Aspekte, wie die verschiedenen Algorithmen, welche zur Zählung der Fahrzeuge verwendet werden, hat, wurde diese bei einigen Wettbewerben eingereicht. Hierbei musste

abhängig vom Wettbewerb ein Projektbericht geschrieben werden, welcher das Projekt zusammenfassen soll. (Stand: 3. April 2022)

4.4.1 innovation@school

Das Projekt wurde beim *innovation@school* (Ideenwettbewerb 2021/2022) eingereicht. Leider hat die Diplomarbeit es nicht in die Phase 2 geschafft.

4.4.2 Jugend Innovativ

Hier wurde das Projekt für die 35. Runde qualifiziert.

4.4.3 Technik fürs Leben-Preis 2022

Hier wurde das Projekt ebenfalls eingereicht.

4.4.4 Maturaprojektwettbewerb - FH Kärnten

Auch hier wurde das Projekt eingereicht.

ANHANG

A Appendix

Glossar

AI AI (englisch für Artificial Intelligence) zu Deutsch "Künstliche Intelligenz", ist die Verarbeitung von Daten mit einem intelligenten Verhalten. Oft wird zum Trainieren Machine Learning verwendet.. 33

API Eine API, (englisch für Application Programming Interface), zu Deutsch "Programmierschnittstelle", ist eine Schnittstelle zwischen zwei oder mehreren Programmen, welche über diese Daten hin- und herschicken.. 33

CLI Die CLI (englisch für Command Line Interface) ist ein Weg, wie Benutzer*innen einen Computer bedienen können. Hier wird ein Text bzw. Kommando in ein Terminal geschrieben und vom Computer verarbeitet. Der Gegenspieler dazu ist die GUI, bei welcher vieles/alles mit einer Maus einfach auszuwählen ist.. 132

CRUD CRUD kurz für Create, Read, Update, Delete beschreibt die 4 essenziellen Befehle von SQL Operationen.. 15

CSI Das Camera Serial Interface, kurz CSI, ist eine Schnittstelle, welche die Kommunikation zwischen Kamera und einem Prozessor ermöglicht.. 56

DSGVO Die Datenschutz-Grundverordnung, kurz DSGVO (eng. GDPR), ist eine Verordnung der EU, welche Regeln zur Verarbeitung persönlicher Daten vorgibt.. 33

eMMC eMMC (englisch für Embedded Multi Media Card) ist ein Halbleiterspeicher, welcher intern (embedded) in einem Gerät eingebaut ist. Alternativen sind entweder SSDs oder UFS Speicher, welche höhere Geschwindigkeiten anbieten.. 45

EU Die Europäische Union, kurz EU, ist eine Union verschiedener Mitgliedstaaten.. 131

GPU Eine GPU (englisch für Graphics Processing Unit) oder auch Grafikprozessor genannt, ist ein Prozessor, welcher auf das Verarbeiten grafischer Daten spezialisiert ist.. 33, 132

GUI Die GUI (englisch für Graphical User Interface) oder zu Deutsch auch grafische Benutzeroberfläche ermöglicht es Benutzer*innen die einfache Bedienung eines Computers mittels grafischer Symbole.. 46, 131

headless Ein headless System (zu Deutsch „kopfloses“ System) ist meist ein Computer, welcher als Server verwendet wird und über keinen Monitor oder sonstige grafische Ausgabe verfügt.. 48

Hibernate Hibernate ist eine Implementation von JPA und erlaubt die einfache Kombination von OOP und Relativen Datenbanken.. 14

JPA JPA kurz für Java Persistence API, ist eine Java Spezifikation zur Verbindung von objektorientierter Programmierung zu relationalen Datenbanken.. 14

MIME MIME (Multipurpose Internet Mail Extensions) gibt die Art und das Format einer Datei an [FB96]. 12

musl musl ist eine C-Standart Bibliothek ähnlich, wie glibc, jedoch versucht musl so lightweight wie möglich zu sein. 8

NPU Eine NPU (englisch für Neural Processing Unit) ist ein Chip, welcher spezialisiert ist maschinelles Lernen und dadurch neuronale Netzwerke zu beschleunigen. Dies ist derzeit ein beliebter Weg auf Einplatinencomputern neurale Netzwerke zu beschleunigen, da keine dedizierte GPU verwendet wird.. 67

pip Pip ist ein/der Package Installer für Python. Er wird von der PyPA (Python Packaging Authority) empfohlen [The21], wobei es auch Alternativen, wie Conda [Ana22] gibt.. 67

ProxMox VE Proxmox VE ist eine komplette Open Source-Virtualisierungsplattform für Server. Es kombiniert KVM- und Container-basierte Virtualisierung und verwaltet virtuelle Maschinen, Container, Storage, virtuelle Netzwerke und Hochverfügbarkeits-Cluster übersichtlich über die zentrale Managementoberfläche.. 8

PyTorch PyTorch ist ein Framework für maschinelles Lernen für Python und wird hauptsächlich von Facebook entwickelt.. 62

Rolling-Release Rolling-Release (zu Deutsch: „laufende Aktualisierung“) bedeutet, dass ein Betriebssystem durchgehend neue Versionen erhaltet, sobald Aktualisierungen verfügbar sind.. 9

Root-Rechte Mit Root-Rechten zu arbeiten (in einer *Unix-Umgebung) bedeutet, dass die Person, welche damit arbeitet, volle Rechte auf alle Dateien im System hat.. 55, 57

SBC Ein SBC (englisch für Single Board Computer) ist ein Einplatinencomputer. Hier befinden sich alle Komponenten, welcher der Computer zum Funktionieren braucht auf einer einzigen Platine. Prominente Vertreter sind hier z. B. der Raspberry Pi oder ein ASUS Tinker Board.. 1, 4

SRP SRP (englisch für single-responsibility principle) besagt, dass eine Klasse nur einen Grund haben muss, weshalb sie umgeschrieben werden darf.. 16

SSH SSH (englisch für Secure Shell) ermöglicht den sicheren (verschlüsselten) Zugriff auf entfernte Server. Meist wird es verwendet, um über die CLI etwas durch Fernwartung zu konfigurieren. Es ist ähnlich wie Telnet, jedoch verschlüsselt.. 51, 132

SSHFS SSHFS steht für SSH File System und ist ein Protokoll, welches es ermöglicht Daten über eine SSH Verbindung zu übertragen. Dies bietet den Vorteil, dass nicht zusätzlich noch ein FTP oder anderer Server zu Datenübertragung aufgesetzt werden muss.. 51

Tensorflow Tensorflow ist ein Framework, welches vor allem für maschinelles Lernen verwendet wird. Es wird hauptsächlich von Google entwickelt.. 35

UUID UUID kurz für Universally Unique Identifier welches eine 128-Bit lange Zahl ist die normal als Hexadezimalzahl dargestellt wird. Wegen ihrer Größe und Art der Generation wird sie als eindeutig gehandhabt.. 15

YOLO YOLO steht für “You only look once” und ist ein neuronales Netzwerk, welches vor allem auch auf real-time Objekt-Erkennung und Klassifizierung ausgelegt ist.. 62

Literatur

[19] *WIN-Total*. 5. Dez. 2019. URL: https://www.wintotal.de/was-ist-ein-hub/#:~:text=Ein%20Hub%20(Englisch%20f%C3%BCr%20%E2%80%9ENabe%E2%80%9C%20oder%20%E2%80%9EKnotenpunkt%E2%80%9C)%20ist,Hubs%20arbeiten%20auf%20Schicht%201%20(Bit%C3%BCbertragungsschicht)%20des%20OSI-Modells..

[21] 5. Okt. 2021. URL: https://www.lora-wan.de/lora-gateway/.

[22a] *DSL-Ratgeber*. 23. Jan. 2022. URL: http://www.dsl-ratgeber.net/wp-content/uploads/2012/08/lte-land-grafik.jpg.

[22b] *Firmware Definition und Begriffserklärung*. 23. Jan. 2022. URL: https://www.it-service24.com/lexikon/f/firmware/.

[22c] *LoRa/GPS HAT Manual Guide*. 22. Jan. 2022.

[22d] *What is a Downlink?* 23. Jan. 2022. URL: https://www.techopedia.com/definition/5055/downlink.

[AEQ18] AEQ-WEB. *Zukunft LoRa und LoRaWAN - Grundlagen, IoT Vernetzung, Reichweiten und Anwendungen*. Hrsg. von AEQ-WEB. 10. Apr. 2018. URL: https://www.youtube.com/watch?v=9cs25G3G7CY.

[Alh20] M. Alhamid. „Machine Learning in Java“. In: *towards data science* (13. Okt. 2020).

[Ana22] Anaconda, Inc. *Conda.io*. Hrsg. von Conda Team. 6. Jan. 2022. URL: https://docs.conda.io/en/latest/.

[And16] Andi. *LoRa, LoraWAN; noch ein neuer Funkstandard*. 10. Apr. 2016. URL: http://wp.andreas.bieri.name/myblog/2016/04/10/lora-lorawan-noch-ein-neuer-funkstandard/.

[Arc21] Arch Linux Communtiy. *Python/Virtual environment*. 5. Dez. 2021. URL: https://wiki.archlinux.org/title/Python/Virtual_environment#Pipenv.

[Arc22] Arch Linux Communtiy. *XDG Base Directory*. 5. Jan. 2022. URL: https://wiki.archlinux.org/title/XDG_Base_Directory.

[Art21] Artifical Intelligence in Medicine, Hrsg. *Is deep learning supervised or unsupervised?* 11. Feb. 2021. URL: https://ai-med.io/ac-observations/is-deep-learning-supervised-or-unsupervised/.

[ber21] berrybase. *Raspberry Pi Camera Module 8MP v2.1*. 18. Okt. 2021. URL: https://shop.hardwario.com/lora-module/.

[Bjö17] Björk. *LoRa Single-Channel Gateway*. 23. Juli 2017. URL: https://www.bjoerns-techblog.de/wp-content/uploads/2017/07/IMG_7164.jpg.

[But+16] H. Butler u. a. *The GeoJSON Format*. Aug. 2016. DOI: 10.17487/RFC7946. URL: https://www.rfc-editor.org/info/rfc7946.

[BWL20] A. Bochkovskiy, C.-Y. Wang und H.-Y. M. Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV].

[Cak20] T. Cake. *Cake Defi*. 20. Jan. 2020. URL: https://support.cakedefi.com/hc/de/articles/360035526972-Was-ist-ein-Node.

[Cor21] Coral. *Coral Products*. Hrsg. von Coral. 21. Sep. 2021. URL: https://coral.ai/products/.

[cor22] coreeletronics, Hrsg. *LattePanda*. 23. Jan. 2022. URL: https://core-electronics.com.au/lattepanda-4g-64gb-the-most-powerful-win10-dev-board.html.

[Dak06] M. Dake. *Neural network*. Hrsg. von W. Commons. 28. Nov. 2006. URL: https://commons.wikimedia.org/wiki/File:Neural_network.svg.

[dpa18] dpa. *t-online*. 20. Sep. 2018. URL: https://www.t-online.de/digital/smartphone/id_84483392/netzausbau-mobilfunkanbieter-nehmen-hunderte-neue-lte-masten-in-betrieb.html.

[Dub22] A. Dubey. „ORB Feature Detection in Python OpenCV“. In: *CodeSpeedy* (23. Jan. 2022).

[DV16] V. Dumoulin und F. Visin. *A guide to convolution arithmetic for deep learning*. 23. März 2016. arXiv: 1603.07285 [stat.ML].

[Eck16] M. Eckert. „ComputerWeekly.de“. In: *Node (Netzwerkknoten)* (7. Jan. 2016).

[eer20] eeric. *ultralytics/yolov5 - Issue 1333 (Where is yolov5 paper?)* 10. Nov. 2020. URL: https://github.com/ultralytics/yolov5/issues/1333.

[FB96] N. Freed und D. N. S. Borenstein. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. Nov. 1996. DOI: 10.17487/RFC2045. URL: https://www.rfc-editor.org/info/rfc2045.

[Fre19] J. Freeman. „InfoWorld“. In: *What is JSON? A better format for data exchange* (25. Okt. 2019).

[Fri22] Fritzbox. *Fritzbox*. 23. Jan. 2022. URL: https://maxwireless.de/wp-content/uploads/2018/10/avm-6890-v2-lte-bands.jpg.

[Fun20] B. Funk. „HotHardware“. In: *Google Coral Dev Board Mini SBC Brings Raspberry Pi-Sized AI Computing to the Edge* (17. Okt. 2020).

[Gil] A. S. Gillis. „IoT Agenda“. In: *What is internet of things (IoT)?* ().

[goe20] goelaparna1520. „Convolutional Neural Network (CNN) in Machine Learning“. In: *GeeksforGeeks* (28. Dez. 2020).

[Har21] Hardwario. *LoRa Module*. 18. Okt. 2021. URL: https://www.khadas.com/vim3.

[He+15] K. He u. a. *Deep Residual Learning for Image Recognition*. 10. Dez. 2015. arXiv: 1512.03385 [cs.CV].

[Hvi18] Hvidberrrg. *The sigmoid function (a.k.a. the logistic function) and its derivative*. 5. Juli 2018. URL: https://hvidberrrg.github.io/deep_learning/activation_functions/sigmoid_function_and_derivative.html.

[IBM20] IBMCloudEducation. *Neural Networks*. Hrsg. von IBMCloudLearn. 17. Aug. 2020. URL: https://www.ibm.com/cloud/learn/neural-networks.

[JBS15] M. Jones, J. Bradley und N. Sakimura. *JSON Web Token (JWT)*. Mai 2015. DOI: 10.17487/RFC7519. URL: https://www.rfc-editor.org/info/rfc7519.

[Kha21a] Khadas. *Khadas VIM3*. Hrsg. von Khadas. 21. Sep. 2021. URL: https://www.khadas.com/product-page/vim3.

[Kha21b] Khadas. *VIM3 - Documentation*. 28. Okt. 2021. URL: https://docs.khadas.com/linux/vim3/#VIM3.

[kha21a] khadas. *khadas/aml_npu_app*. 7. Dez. 2021. URL: https://github.com/khadas/aml_npu_app.

[kha21b] khadas. *khadas/ksnn*. 20. Dez. 2021. URL: https://github.com/khadas/ksnn.

[kha21c] khadas. *khadas/tengine_khadas_app*. 7. Dez. 2021. URL: https://github.com/khadas/tengine_khadas_app.

[kha21d] khadas. *VIM 3*. 18. Okt. 2021. URL: https://www.berrybase.de/raspberry-pi/raspberry-pi-computer/kameras/raspberry-pi-camera-module-8mp-v2.1?c=341.

[Kni21] D. Knight. *DietPi OS stats and comparison*. 12. Sep. 2021. URL: https://dietpi.com/stats.html.

[Kno20] KnowHow. *RAM(Arbeitsspeicher): Was steckt dahinter?* 28. Sep. 2020. URL: https://www.ionos.de/digitalguide/server/knowhow/was-ist-ein-arbeitsspeicher/.

[Kom21] E. Kompendium. *Elektronik Kompendium*. 27. Sep. 2021. URL: https://www.elektronik-kompendium.de/sites/kom/2203171.htm.

[Kuk21] M. Kuklin. „Optical Flow in OpenCV (C++/Python)“. In: *LearnOpenCV* (4. Jan. 2021).

[Lat21a] LattePanda. *LattePanda Products*. Hrsg. von LattePanda. 21. Sep. 2021. URL: http://www.lattepanda.com/products.

[Lat21b] LattePandaTeam. *Solutions for Your Intelligent Factory with LattePanda*. 22. Jan. 2021. URL: https://www.lattepanda.com/blog-3283.html.

[lea22] learncisco. *learncisco*. 23. Jan. 2022. URL: https://www.learncisco.net/courses/icnd-1/lan-connections/functions-of-routing.html.

[Ley21] Leytala. *CSI-Kamera*. Hrsg. von Amazon. 27. Sep. 2021. URL: https://www.amazon.de/CSI-Schnittstellenkameramodul-Millionen-Raspberry-Komponenten-Mini-Kamera-Videomodul/dp/B08QCQFDJX/ref=sr_1_2?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=MIPI-CSI%20raspberry&qid=1632216205&sr=8-2.

[Li+18] Z. Li u. a. *DetNet: A Backbone network for Object Detection*. 17. Apr. 2018. arXiv: 1804.06215 [cs.CV]. URL: https://arxiv.org/pdf/1804.06215v1.pdf.

[Lin+16] T.-Y. Lin u. a. *Feature Pyramid Networks for Object Detection*. 9. Dez. 2016. arXiv: 1612.03144 [cs.CV].

[Liu+15] W. Liu u. a. *SSD: Single Shot MultiBox Detector*. 8. Dez. 2015. DOI: 10.1007/978-3-319-46448-0_2. arXiv: 1512.02325 [cs.CV]. URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[Liu+18] S. Liu u. a. *Path Aggregation Network for Instance Segmentation*. 5. März 2018. arXiv: 1803.01534 [cs.CV].

[LXD22] LXDE Contributors. *LXDE - Desktop environment for all*. 7. Jan. 2022. URL: https://www.lxde.org/.

[Mai21] G. Maindola. *A Brief History of YOLO Object Detection Models From YOLOv1 to YOLOv5*. Hrsg. von machinelearningknowledge. 27. Aug. 2021. URL: https://machinelearningknowledge.ai/a-brief-history-of-yolo-object-detection-models/.

[Maj18] K. Majek. *4K Road traffic video for object detection and tracking - free download now!* 16. Juli 2018. URL: https://www.youtube.com/watch?v=MNn9qKG2UFI&list=PLcQZGj9lFR7y5WikozDSrdk6UCtAnM9mB&index=3.

[mcc22] mcci-catena. *arduino-lmic*. 27. Jan. 2022. URL: https://github.com/mcci-catena/arduino-lmic.

[Mue21] V. Mueller. „Non-maximum suppression“. In: *towards data science* (30. Sep. 2021).

[nee21] aman neekhara. „JSON web token | JWT“. In: (21. Dez. 2021).

[OPE21] OPEN AI LAB. *Tengine Docs*. 6. Aug. 2021. URL: https://github.com/OAID/Tengine.

[Ope21] OpenCV. *Object Tracking - Function Documentation*. 25. Dez. 2021. URL: https://docs.opencv.org/4.5.5/dc/d6b/group__video__track.html#ga5d10ebbd59fe09c5f650289ec0ece5af.

[Ope22] OpenCV. *Feature Matching*. 23. Jan. 2022. URL: https://docs.opencv.org/4.x/dc/dc3/tutorial_py_matcher.html.

[pan22] pankaj0323. „Conda vs Pip: Choosing your Python package managerChoosing Your Python Package Manager Conda Vs Pip“. In: *AskPython* (6. Jan. 2022).

[Pis21] S. Pisjak. *image-recognition - vehicle.py*. 26. Dez. 2021. URL: https://git.htl-klu.at/da-2122-5chel/pnp-commean/image-recognition/-/blob/21384a5881da3c8bdd73c81990aa1d1eb4514f27/KSNN/Darknet/python/vehicle/vehicle.py.

[Pis22a] S. Pisjak. *image-recognition - multiple_regression.py*. 13. Jan. 2022.

[Pis22b] S. Pisjak. *image-recognition - yolo_post_processing.py*. 21. Jan. 2022. URL: https://git.htl-klu.at/da-2122-5chel/pnp-commean/image-recognition/-/blob/21384a5881da3c8bdd73c81990aa1d1eb4514f27/KSNN/Darknet/python/vehicle/vehicle.py.

[pjr21] pjreddie. *pjreddie/darknet*. 21. Dez. 2021. URL: https://github.com/pjreddie/darknet.

[Ras21] Raspberry. *Raspberry Products*. Hrsg. von Raspberry. 21. Sep. 2021. URL: https://www.raspberrypi.org/products/.

[Ras22a] RaspberryPi.com. *Raspberry Pi 4 Tech Specs*. 6. Jan. 2022. URL: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/.

[Ras22b] Raspi.TV, Hrsg. *Raspberry Pi 3 Model B launches today*. 23. Jan. 2022. URL: http://raspi.tv/2016/raspberry-pi-3-model-b-launches-today-64-bit-quad-a53-1-2-ghz-bcm2837.

[RF18] J. Redmon und A. Farhadi. *YOLOv3: An Incremental Improvement*. 8. Apr. 2018. arXiv: 1804.02767 [cs.CV].

[Rob18] R. Robinson. *mlnotebook.github.io*. 23. Juli 2018. URL: https://github.com/mlnotebook/mlnotebook.github.io.

[Rog22] Rogon. *Rogon*. 23. Jan. 2022. URL: https://www.rogon.de/images/wlan1.jpg.

[rom19] rominVaghani. „Introduction to Spring Framework“. In: *GeeksforGeeks* (5. Dez. 2019).

[Rus04] C. Y. Rusty Russell Daniel Quinlan. *Filesystem Hierarchy Standard Group*. 2004. URL: https://www.pathname.com/fhs/pub/fhs-2.3.html.

[Sah18] S. Saha. „A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way“. In: *towards data science* (15. Dez. 2018).

[Sah21] A. Sahu. *Top 8 Programming Languages for Artificial Intelligence*. 12. Mai 2021. URL: https://www.westagilelabs.com/blog/top-8-programming-languages-for-artificial-intelligence/.

[Sam22] Samsung. *Samsung*. 23. Jan. 2022. URL: https://images.samsung.com/is/image/samsung/p5/ch/explore/samsung-within/innovation/5g-evolution/20190911-Samsung-5G-Infografik-1G-Desktop.png.

[Sar20] J. J. " Sarjeant. *Axios Introduction*. 2020. DOI: https://axios-http.com/docs/intro.

[sha20] ujjwal sharma. „Django Introduction and Installation“. In: *GeeksforGeeks* (26. Feb. 2020).

[Six21] Sixfab. *Raspberry Pi 3G-4G/LTE Base Shield V2*. 18. Okt. 2021. URL: https://sixfab.com/product/raspberry-pi-3g-4glte-base-shield-v2/.

[Sta21] C. Stats. „Understanding Max Pooling and Average Pooling. Which one to use and Why?“ In: *Medium* (22. März 2021).

[Sti21] J. H. Stiftung. *Induktionsschleifen im Straßenverkehr*. 27. Okt. 2021. URL: https://www.leifiphysik.de/elektrizitaetslehre/elektromagnetische-induktion/ausblick/induktionsschleifen-im-strassenverkehr.

[Tel22] Telekom. *Telekom*. 23. Jan. 2022. URL: https://www.telekom.com/en/company/details/the-nine-most-important-facts-about-lte-606446#:~:text=Overall,%2098%%20of%20Germany%E2%80%99s%20population%20now%20has%20LTE,Mbit/s.%20GSM%20supports%20a%20maximum%20of%20220%20Kbit/s..

[The21] The pip developers. *pip*. 22. Okt. 2021. URL: https://pypi.org/project/pip/.

[Ver21] S. Verkehrssysteme. *Sicherheit bei automatischen Schranken*. 27. Okt. 2021. URL: https://www.stadlmayr.at/schranken_sicherheitselemente.html.

[Ver22] Verizon. *Verizon*. 23. Jan. 2022. URL: https://www.verizon.com/info/definitions/wifi/.

[Web21] C. Website. *Chocolatey Software - The Package Manager for Windows*. [Online; accessed 2021-10-15]. 2021. URL: https://chocolatey.org/.

[Xie+16] S. Xie u. a. *Aggregated Residual Transformations for Deep Neural Networks*. 16. Nov. 2016. arXiv: 1611.05431 [cs.CV].

[You21] E. You. *Vue.js - The Progressive JavaScript Framework*. 27. Okt. 2021. URL: https://vuejs.org/.

[Zis21] B. Zissimopoulo. *billziss-gh/sshfs-win*. 8. Juni 2021. URL: https://github.com/billziss-gh/sshfs-win.