FrubotAI

Entwicklung und Programmierung eines autonomen Roboters zur Fruchterkennung unter Verwendung von maschinellem Lernen

Von Adriano Carlucci und Leon Schaad
G2020D

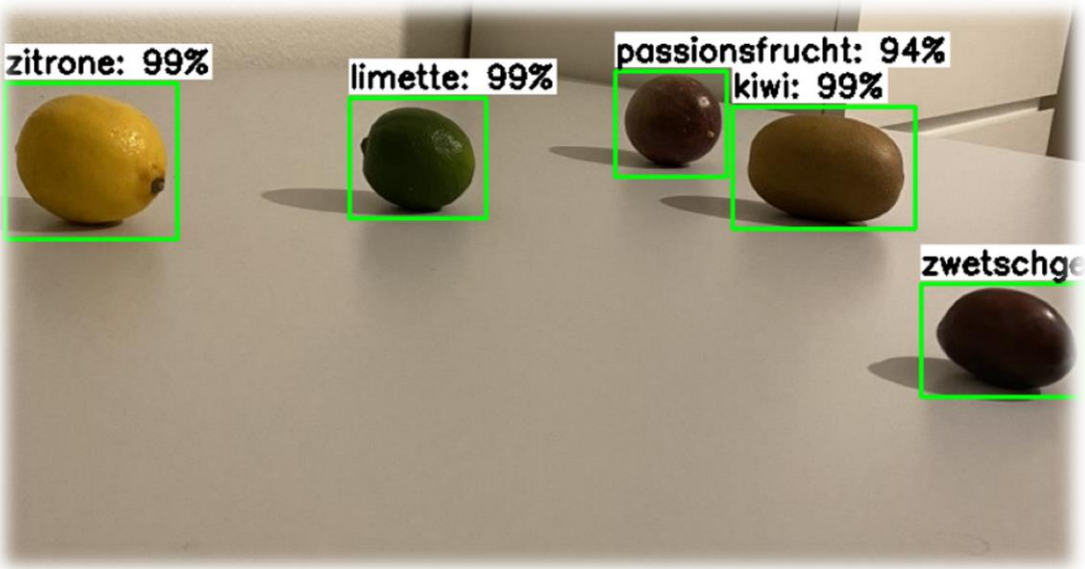

Abbildung 1: Titelbild1

Maturaarbeit

Neue Kantonschule Aarau

Betreuende Lehrperson: Nuhro Ego

Zweitbeurteilung: Dr. Stephen Weyeneth

12.10.2023

Abstract

Diese Maturaarbeit widmet sich dem Programmieren und Trainieren eines autonomen Fruchterkennungsroboters, welcher mit dem Namen "FrubotAI" getauft wurde. Unter Verwendung von TensorFlow wurde der Roboter mit dem Ziel entwickelt, die Fähigkeit zu erlernen, fünf verschiedene Früchte zu erkennen und zu unterscheiden. Durch die Verwendung von Sensoren, wie einer Kamera und einem Ultraschallsensor soll der Roboter in der Lage sein, in einem definierten Bereich, die Früchte selbständig zu finden, sich ihnen anzunähern und mittels Text-to-Speech Anweisungen zur Einsortierung der Früchte auszugeben. Das Herzstück dieser Maturaarbeit bildet das Machine-Learning-Modell, welches mithilfe von selbstaufgenommenen Bildern trainiert wurde, um eine präzise Erkennung der Früchte zu gewährleisten. Die Ergebnisse der Arbeit beinhalten eine Kombination aus Robotik, maschinellem Lernen und der Implementierung eines Programmcodes.

Vorwort

Wir beide sind von den Möglichkeiten und Entwicklungen des maschinellen Lernens und der Robotik begeistert. Deshalb war für uns relativ schnell klar, dass wir ein Projekt in diesen Themengebieten realisieren wollen.

Für die Aufgabe der Erkennung und Unterscheidung von Früchten haben wir uns entschieden, da dieses Projekt die grundlegenden Aspekte der Themengebiete Machine Learning, künstliche Intelligenz und Robotik aufgreift. Diese Gebiete finden in vielen Bereichen der Arbeitswelt Anwendung. Diese Vielseitigkeit und Relevanz hat von Anfang an unser Interesse geweckt.

Die Kombination aus der Steuerung eines Roboters und der Implementierung eines Machine Learning Modells durch einen Programmcode ermöglicht uns, sowohl auf der Hardware- als auch auf der Softwareebene zu arbeiten.

Die Arbeit wurde ohne grosses Vorwissen im Bereich des maschinellen Lernens und mit grundlegenden Programmierkenntnissen in Angriff genommen. Im Laufe der Arbeit konnten wir daher grosse Fortschritte in diesen Bereichen erzielen.

Durch die Arbeit konnten wir nicht nur unser technisches Wissen erweitern, sondern haben auch Fähigkeiten wie kritisches Denken, die Lösung komplexerer Probleme und die Zusammenarbeit im Team entwickelt.

Wir bedanken uns bei unserem Betreuer, Nuhro Ego, für seine wertvollen Ratschläge und für seine stetige Motivation, in Momenten, in denen nicht alles reibungslos verlief.

Des Weiteren bedanken wir uns bei EdjeElectronics für die hilfreichen Videos und die Bereitstellung der zahlreichen Anleitungen zur Erstellung eines Machine-Learning-Modells mit TensorFlow.

Inhaltsverzeichnis

Inhalt

1.0 Einleitung

Die Veröffentlichung des Chatbots "ChatGPT" hat eine enorme Aufmerksamkeit auf sich gezogen. Es stellt sich die Frage, inwiefern künstliche Intelligenz den Alltag der Zukunft beeinflussen wird.

Die schnelle Entwicklung dieser Gebiete ermöglicht nicht nur weite gesellschaftliche Veränderungen, sondern öffnet auch aufregende Möglichkeiten für Leute, die sich für ebendiese Gebiete informieren.

Die Konstruktionsidee dieser Maturaarbeit ist das Programmieren und Trainieren eines Roboters, der in der Lage ist, fünf verschiedene Früchte zu erkennen. Das Erkennen basiert auf einem Machine-Learning-Modell. Das Ziel ist, dass der Roboter in einem vordefinierten Testgelände die fünf verschiedenen Früchte eigenständig sucht, findet und ansteuert. Zudem soll der FrubotAI mithilfe von Text-to-Speech dazu auffordern, die Frucht an den zugehörigen Ort zu bringen.

Um dieses Ziel zu erreichen, müssen die drei Gebiete der Robotik, des maschinellen Lernens und der Informatik kombiniert werden.

Der Gebrauch der verschiedenen Roboterkomponenten gehört zum Thema der Robotik. Dabei werden verschiedene Sensoren, Motoren und eine Kamera verwendet. Diese sollen dem Roboter helfen, sich zu orientieren und die Früchte zu finden.

Damit sich der FrubotAI überhaupt an die gesuchten Früchte annähern kann, muss er zuerst erkennen, dass sich eine Frucht vor ihm befindet. Zur Erkennung der Früchte ist das Machine-Learning-Modell verantwortlich. Dieses wird mit selbst gemachten Fotos trainiert. Die Fotos selbst aufzunehmen, ist ein bewusster Entscheid, um ein besseres Gefühl für das Trainieren von Machine-Learning-Modellen zu erlangen.

Der Bereich der Informatik kommt in der Implementierung des Codes zum Zug. Der Programmcode kombiniert die beiden oberen Gebiete, da er die Roboterkomponenten in Verbindung mit dem Machine-Learning-Modell verwendet, um die Früchte finden, erkennen und mit ihnen interagieren zu können.

Diese Arbeit beantwortet die Hauptfrage, wie ein Fruchterkennungsroboter erstellt werden kann. Diese Frage teilt sich in untenstehende Teilfragen auf, welche vor allem im Methodenteil beantwortet werden.

Welche Hardware eignet sich zur Konstruktion eines Roboters, der Objekte erkennen soll?

Wie muss ein Machine-Learning-Modell trainiert werden, um in einer realen Umgebung zu funktionieren?

Welches Vorgehen muss gewählt werden, um einen funktionierenden Programmcode zu erstellen, der die Hardware und das trainierte Modell zu einem Roboter verbindet, welcher Objekte suchen, finden und erkennen soll.

2.0 Theoretische Grundlagen

2.1 Robotik und KI im Laufe der Zeit

Die Faszination für eigenständig Arbeitende, mechanische Konstruktionen reicht bis in die griechische Antike zurück.

Dabei stammen Ideen für Roboter oftmals aus dem Bereich der Mythologie. Ein Beispiel dafür, ist der eiserne Talos, der vom Schmiedegott Hephaistos geschmiedet wurde, mit der Aufgabe, die Insel Kreta zu bewachen. (ARTI - Autonomous Robot Technology GmbH, 2021)

Es gab aber auch bereits reale Ansätze für automatische Maschinen, wie automatische Theater und Musikapparate, welche auf den Heron von Alexandria zurückzuführen sind. (Wikipedia, 2023)

Die Idee von automatischen Vorrichtungen, die bestimmte Aufgaben ohne menschliches Beisteuern ausführen können, ist also bereits sehr früh entstanden.

Der heute weithin gebräuchliche Begriff "Roboter" entstand jedoch erst im 20. Jahrhundert. Der tschechische Schriftsteller Karel Čapek verwendete ihn erstmals in seinem 1920 veröffentlichten Theaterstück "R.U.R. (Rossum's Universal Robots)". Dabei werden künstliche, menschenähnliche Wesen dargestellt, die in Fabriken arbeiten und als "Roboter" bezeichnet werden.

Seitdem hat sich der Begriff "Roboter" in der Literatur, in Filmen, in der Technologie wie auch in der Alltagssprache weit verbreitet, um sich auf automatische Maschinen zu beziehen, die menschliche Aufgaben erledigen können.

Aus dem Begriff "Roboter" wurde dann auch der Begriff "Robotik" abgeleitet, welcher sich auf das Forschungs- und Anwendungsgebiet, das sich mit der Konstruktion, Programmierung, Steuerung und Anwendung von Robotern befasst, bezieht. (Damien, 2020)

Durch den Fortschritt der Technik hat sich die Robotik zu einem grossen Bereich der Wissenschaft entwickelt. Daher unterteilt sich die Robotik heutzutage in zahlreiche Unterkategorien, die breite Anwendungsmöglichkeiten in der heutigen Welt finden. (Wikipedia, 2023)

In den letzten Jahren hat sich vor allem der Bereich der Künstlichen Intelligenz stark etabliert. Die Künstliche Intelligenz bezieht sich auf die Entwicklung und Herstellung von Maschinen, um die Problemlösungs- und Entscheidungsfähigkeit des Menschen nachzuahmen. (IBM, kein Datum)

In dieser Arbeit werden insbesondere die Aspekte des maschinellen Lernens und des autonomen Fahrens im Rahmen der künstlichen Intelligenz behandelt.

2.2 Roboterkomponenten

Für den Bau eines Roboters braucht es verschiedene Komponenten. Es werden verschiedene Sensoren, Motoren und Erweiterungen benötigt, um dem Roboter zu ermöglichen, sich fortzubewegen. Der "FrubotAI" Roboter basiert auf einem Raspberry Pi. Ergänzt wird dieser durch das PiCar-X Roboterset von SunFounder, welches den Raspberry Pi zu einem Roboter komplettiert.

2.2.1 Raspberry Pi

Der Raspberry Pi ist ein Einplatinencomputer im Format einer Kreditkarte. Aufgrund seiner Grösse eignet er sich optimal für Projekte im Themenbereich der Robotik. Der gewählte Raspberry Pi 4B hat einen Broadcom BCM2711 Prozessor und acht Gigabyte Arbeitsspeicher. Er unterstützt eine WLAN-Verbindung und hat verschiedene Anschlüsse, wie vier USB-Ports und zwei micro-HDMI-Ports. (Wikipedia, 2023), (Raspberry Pi, kein Datum)

Es gibt auch die Möglichkeit, Komponenten über den verbauten 40-Pin General Purpose Input/Output (GPIO), im deutschen Allzweckeingabe/-ausgabe, anzusteuern. Die GPIOs werden verwendet, um digitale Signale zu leiten. (Wikipedia, 2023)

Der Raspberry Pi läuft mit dem Betriebssystem Raspberry Pi OS, welches auf Linux basiert. (Raspberry Pi OS, kein Datum)

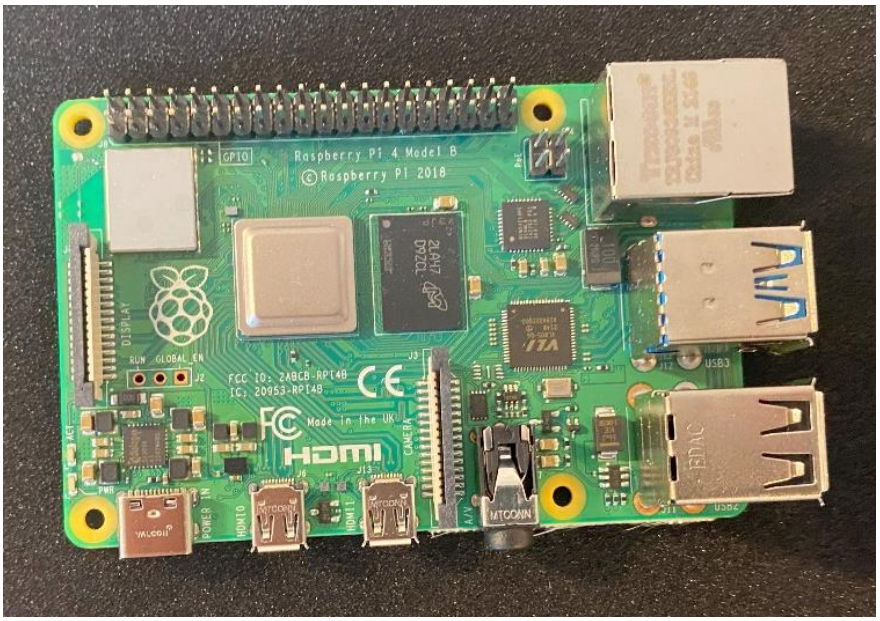

Abbildung 2: Raspberry Pi 4 (eigene Abbildung)

2.2.2 PiCar-X

PiCar-X ist ein Roboterset, das von einem Raspberry Pi gesteuert wird. Neben den Hardwarekomponenten, welche unten aufgeführt werden, kommt der PiCar-X mit einer Softwarekomponente. Diese besteht aus verschiedenen Beispielen, mit Hilfe derer man den Roboter testen und den Code erweitern kann. Somit bietet dieses Roboterset eine gute Option, um in das komplexe Thema der Robotik einzusteigen. (SunFounder, kein Datum)

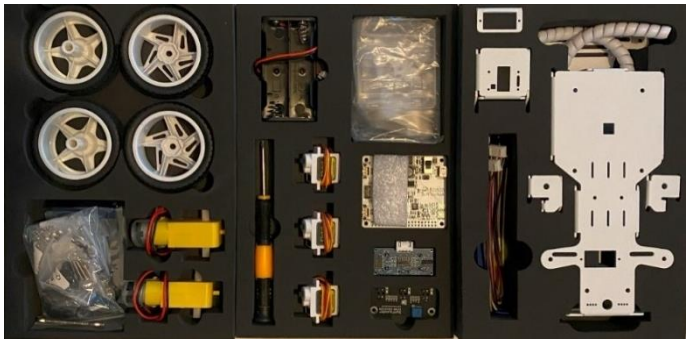

Abbildung 3:PiCar-X Komponenten (eigene Abbildung)

Chassis und Räder

Das Chassis und die Ränder bilden die Grundstruktur des Roboters. Am Chassis werden alle anderen Hardwarekomponenten des PiCar-X angebracht. Es besteht aus leichtem und robustem Aluminium. Die Räder sind aus Kunststoff und haben einen Plastikpneu. Dadurch sind sie leicht und haben eine gute Haftung.

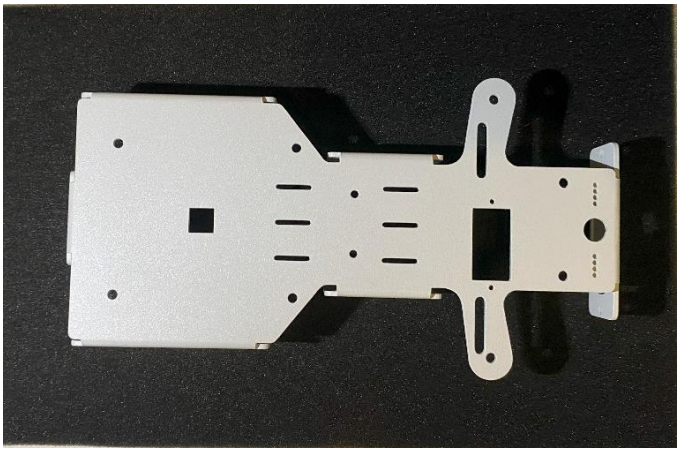

Abbildung 4: Chassis (eigene Abbildung)

Motoren

Im Roboterset von PiCar-X sind zwei Arten von Motoren eingebaut. Zwei Gleichstrom-Getriebemotoren und drei Servomotoren. (EducaTec AG, 2023)

Gleichstrom-Getriebemotoren

Die Gleichstrom-Getriebemotoren werden im PiCar-X zum Antrieb der Hinterräder verwendet. Die eingebauten Motoren sind sogenannte TT Motoren. Die Motoren erhalten ihren Namen von der charakteristischen Form. (rot eingezeichnet)

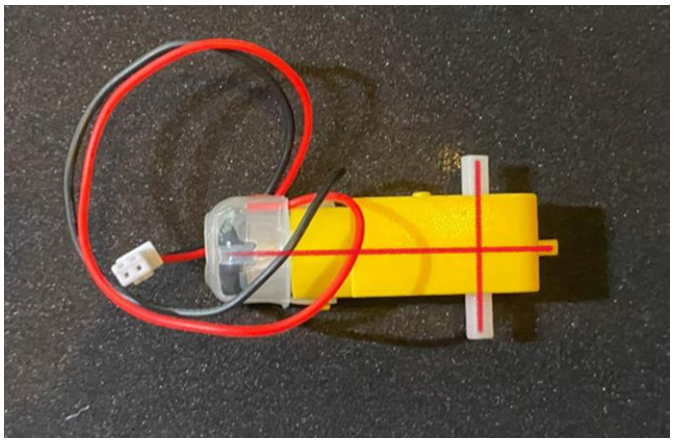

Abbildung 5: TT-Motor (eigene Abbildung)

Ein Gleichstrommotor ist ein Elektromotor, der durch eine Gleichstromquelle betrieben wird. Der Wechselstrommotor hingegen verwendet Wechselstrom. Dabei wird elektrische Energie in mechanische umgewandelt. (simpleclub, 2015)

Der verbaute Gleichstrom-Getriebemotor kombiniert einen Elektromotor mit einem Getriebe. Ein Getriebe besteht aus Zahnrädern, welche dafür sorgen, dass das Drehmoment und die Drehzahl modifiziert werden können. Das ermöglicht es, die Ausgangsleistung des Motors an die spezifischen Anforderungen einer Anwendung anzupassen (OSZ-KFZ, 2021)

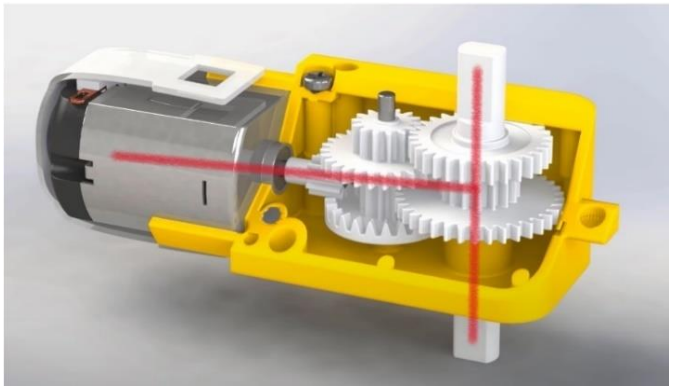

Abbildung 6: TT-Motor schematisch (MRMS-WORKSHOP, 2020)

Servomotoren

Der PiCar-X ist mit drei Servomotoren ausgestattet. Einer von ihnen steuert die Vorderräder und ermöglicht so die präzise Kontrolle über die Fahrtrichtung. Die beiden anderen Servomotoren dienen zur Ausrichtung der Kamera. Die Nutzung von Servomotoren ermöglicht dem PiCar-X präzises Bewegen und Ausrichten.

Abbildung 7: Servomotor (SunFounder, 2018)

Ultraschallsensor

Das Ultraschallmodul besteht aus zwei Sensoren: Einer sendet das Ultraschallsignal, der andere empfängt es. Wenn die ausgestrahlten Ultraschallsignale auf ein Objekt treffen, werden die Schallwellen reflektiert und vom anderen Sensor aufgenommen. Das Modul misst die Zeit zwischen dem Ausstrahlen und dem Empfangen des Signals und berechnet damit die Distanz zum Objekt. (Honerlage, 2021)

$$Entfernung = \frac{Zeit \cdot Schallgeschwindigkeit}{2}$$

Abbildung 8: Formel Ultraschallsensor (eigene Abbildung)

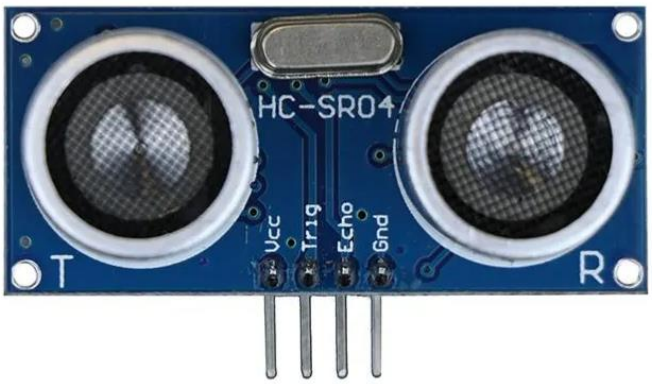

Abbildung 9: Ultraschallsensor (rs-online, 2023)

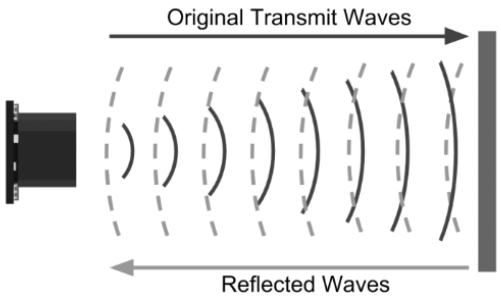

Abbildung 10: Funktionsweise Ultraschallsensor (exptech, 2018)

Grayscale Modul

Das Grayscale-Modul von SunFounder ist am Unterboden des Chassis des PiCar-X fixiert. Es besteht aus drei Infrarot-Lichtschrankensensoren (SO, S1,S2), die in einer Linie angeordnet sind. Diese emittieren ein Infrarotlicht, welches je nach Oberfläche unterschiedlich stark reflektiert wird. So ermittelt das Grayscale-Modul, welcher Untergrund sich direkt unter dem Roboter, links vom Roboter und rechts vom Roboter befindet. Der PiCar-X kann mithilfe des Grayscale-Moduls so etwa einer Linie folgen. (SunFounder, kein Datum)

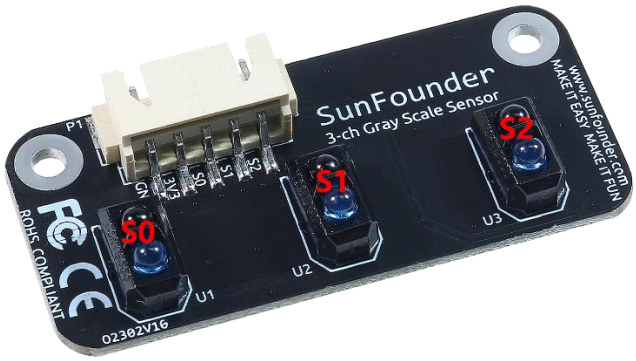

Abbildung 11: Grayscale-Modul (SunFounder, 2023)

Lautsprecher

Das PiCar-X Roboterset enthält einen kleinen Monolautsprecher, der auf dem Robot-Hat montiert ist. Dieser wird zur Text-to-Speech Ausgabe verwendet. Der Widerstand des Lautsprechers beträgt 8 Ohm und die Leistung 1 Watt. (SunFounder, kein Datum)

Abbildung 12: Lautsprecher allgemein (distrelec, 2023)

Kamera Modul

Die Raspberry Pi V2 Kamera ist eine kleine Kamera, welche direkt am Raspberry Pi angeschlossen wird. Sie hat eine Auflösung von 3280 × 2464 Pixel, was 8 Megapixel entspricht. Es ist möglich, mit der Kamera Bilder und Videos aufzunehmen. (Raspberry Pi, kein Datum)

Abbildung 13: Raspberry Pi V2 Kamera (Raspberry Pi, 2023)

Stromversorgung

Der PiCar-X wird grundsätzlich von Batterien betrieben. Wenn diese angeschlossen sind und der Schalter auf dem Robot-Hat aktiviert ist, werden sowohl der PiCar-X als auch der Raspberry Pi mit Strom versorgt. Es ist weiterhin möglich, den Raspberry Pi und die Kamera über den USB-C-Anschluss mit Strom zu versorgen. Dies ermöglicht eine flexible Stromversorgung, je nach Anwendungszweck.

Alternativ ist es möglich, ein Labornetzgerät an den Kontakten der Batterien anzuschliessen. Somit lässt sich der Raspberry Pi inklusive Robot-Hat mit Strom versorgen. Der Vorteil gegenüber den Batterien ist, dass das Labornetzgerät nicht aufgeladen werden muss, dafür ist der Roboter mit den Batterien mobiler.

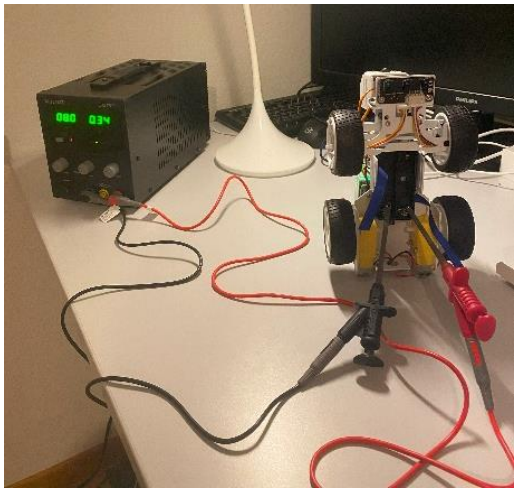

Abbildung 14: Labornetzgerät (eigene Abbildung)

Robot-Hat

Auflistung der Komponenten:

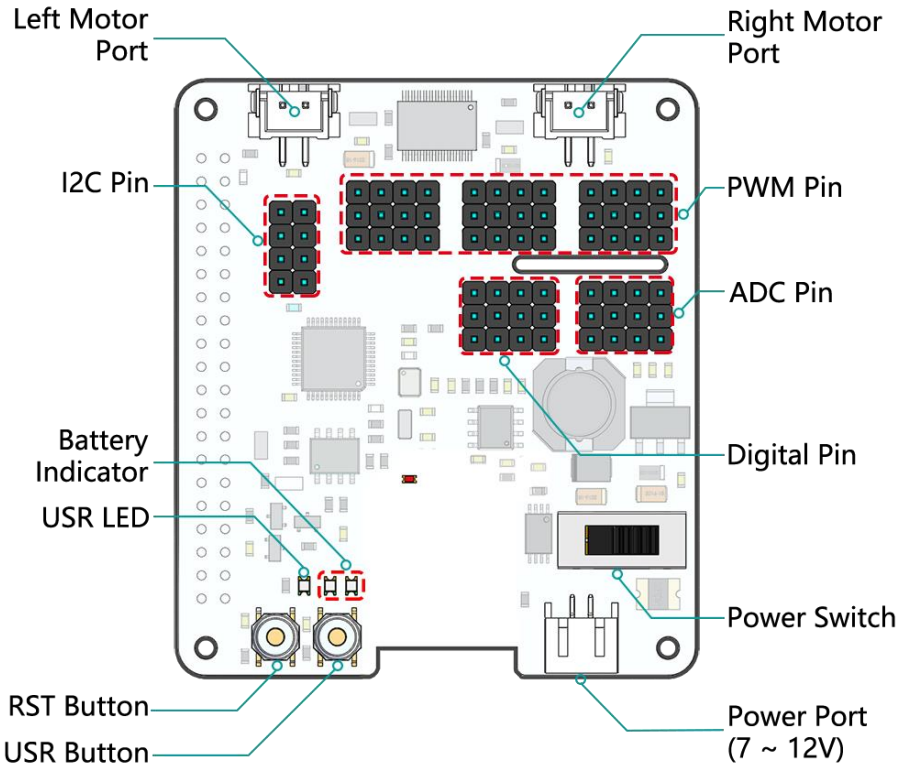

Abbildung 15: Robot-Hat (SunFounder, 2023)

Der Robot-Hat ist eine Erweiterungsplatine für den Raspberry Pi. Er wird über die GPIO-Pins des Raspberry Pis angeschlossen. Der Robot-Hat ermöglicht dem Raspberry Pi die Kommunikation mit den Komponenten des PiCar-X Robotersets.

Folgende Komponenten sind im Robot-Hat integriert: (SunFounder, kein Datum)

- Links/Rechts Motor Port: In diesen beiden Anschlüssen werden jeweils der linke und rechte Antriebsmotor angeschlossen.
- I2C-Pin: Dieser Pin ermöglicht die Kommunikation zwischen Raspberry Pi und Robot-Hat.
- PWM-Pin: Von diesen Pins aus werden die Motoren angesteuert, indem beispielsweise eine Spannung übertragen wird. (Wikipedia, 2022)
- ADC-Pin: Über diesen Pin erfolgt die Steuerung des Grayscale-Moduls. Die analogen Signale des Grayscale-Moduls werden in digitale Werte umgewandelt. (Wikipedia, 2022)
- Digital-Pin (GPIO): Von diesen Pins aus wird der unter anderem der Ultraschallsensor angesteuert. Die digital Pins haben nur zwei Zustände (0 oder 1). Das heisst, dass sie entweder ein oder aus sind. Sie übertragen nur digitale Signale. (Wikipedia, 2023)
- Batterie-Indikator: Der Batterie-Indikator verfügt über zwei grüne LED-Lämpchen. Beide leuchten auf, wenn die Spannung über 7.8V liegt. Zwischen 6.7V und 7.8V leuchtet nur ein Lämpchen. Bei einer tieferen Spannung sind beide aus. So lässt sich einfach überprüfen, ob der PiCar-X ausreichend Strom erhält.

- USR-LED: Dieses LED-Lämpchen ist konfigurierbar. Es ist möglich, über den Programmcode dieses Lämpchen nach seinen Wünschen anzusteuern. Eine Ausgabe von 1 schaltet das Lämpchen ein, eine Ausgabe von 0 schaltet es aus.
- RST-Button: Ein kurzes Drücken des RST Buttons führt zum Zurücksetzen des laufenden Programms. Langes Drücken trennt die Bluetooth-Verbindung des Raspberry Pis mit dem verbundenen Gerät.
- USR-Button: Diese Taste ist gleich wie das USR-LED-Lämpchen noch ohne vordefinierten Nutzen. Der Benutzer kann über diesen Button frei verfügen.
- Power-Switch: Der Power-Switch ist der Schalter, um die Stromversorgung des Robot-Hats ein- und auszuschalten.
- Power-Port: Der Power Port ist der Anschluss, welcher den Robot-Hat und Raspberry Pi mit Strom versorgt. Die Spannung dieses Ports liegt zwischen 7V und 12V.

2.3 Machine Learning

Maschinelles Lernen ist ein Teilbereich der künstlichen Intelligenz. Auf Grundlage von Daten und Algorithmen wird der menschliche Lernprozess imitiert und dabei seine Genauigkeit Stück für Stück verbessert.

Es wird also nicht versucht, dass der Programmcode, die Lösung des Problems ausdrückt. Stattdessen wird der Programmcode als ein lernendes System definiert, welches durch die Analyse von verschiedenen Beispielen (Daten) eigenständig eine funktionierende Lösung erarbeiten soll. Dieser Ansatz ermöglicht es Computern, aus ihren Erfahrungen zu lernen und nicht nur Aufgaben auszuführen, für welche sie explizit programmiert wurden. (Wuttke, kein Datum)

Ein Beispielproblem für ein Machine Learning Modell ist die Erkennung von Früchten. Ein Computerprogramm soll durch die Eingabe von Daten, in diesem Falle einem Bild einer Zitrone, automatisch erkennen, um welche Frucht es sich handelt. Die Ausgabe lautet folgendermassen: "Zitrone". Zudem kann das Modell auch die Wahrscheinlichkeit dieser Zuordnung angeben, beispielsweise "zitrone: 99%“

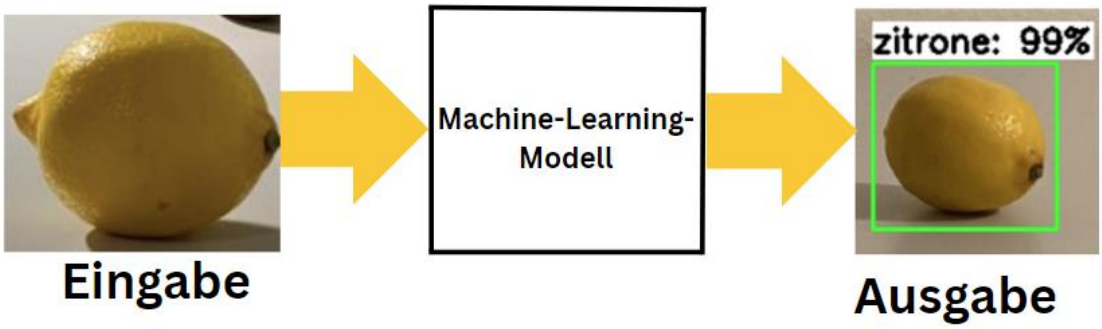

Abbildung 16: Machine-Learning-Modell (eigene Abbildung)

2.3.1 Überwachtes maschinelles Lernen

Für verwendetes Beispielproblem eignet sich das überwachte maschinelle Lernen.

Beim überwachten maschinellen Lernen wird von gekennzeichneten Datensätzen Gebrauch gemacht, um einen Algorithmus zu trainieren, der in der Lage ist, Objekte zu erkennen und zu unterscheiden.

Einfacher ausgedrückt, werden gelabelte Bilder, beispielsweise ein Bild einer Zitrone, mit dem Titel "Zitrone" als Eingabe genutzt, um das Modell zu trainieren. Wenn nun ein anderes Bild einer Zitrone in das trainierte Modell eingegeben wird, soll dieses ebenfalls als Zitrone erkannt und ausgegeben werden. (Wuttke, kein Datum)

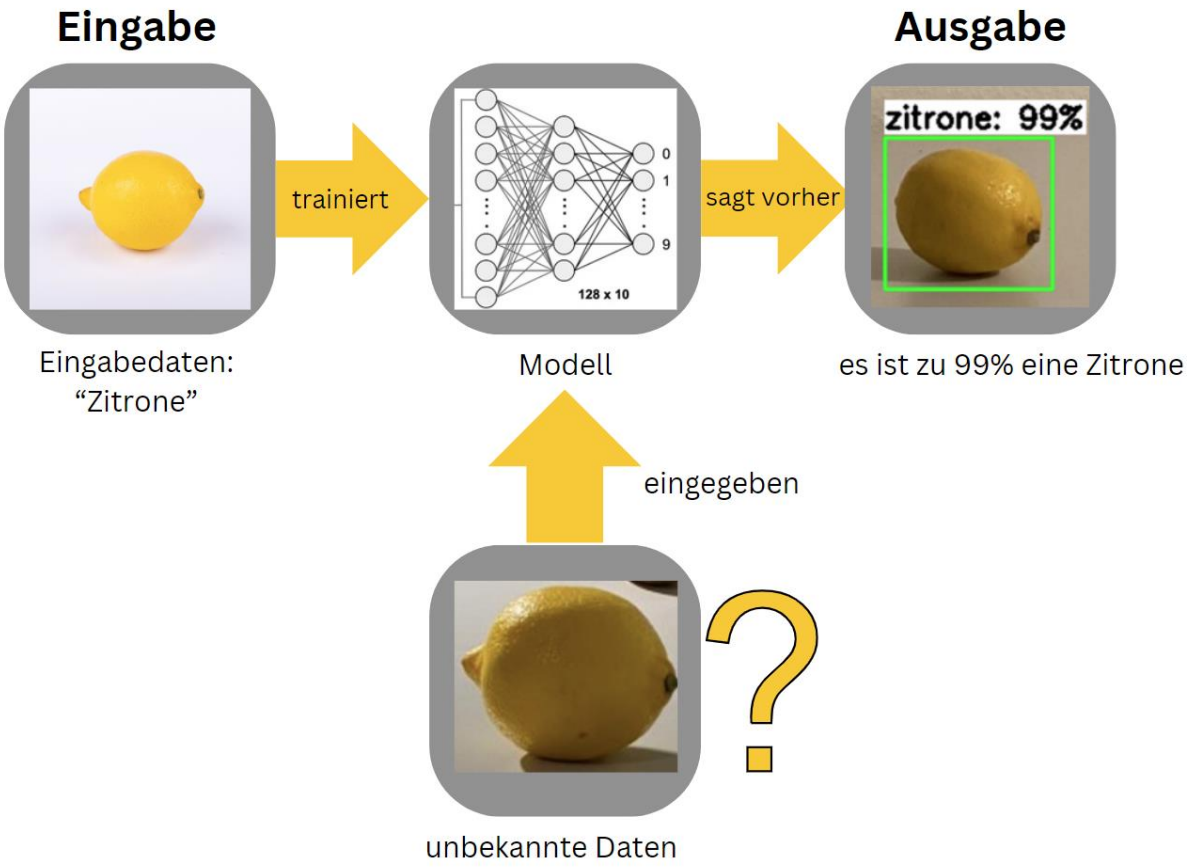

Abbildung 17: Überwachtes maschinelles Lernen (eigene Abbildung)

Neben dem überwachten maschinellen Lernen existieren noch die Ansätze des unüberwachten, des teilüberwachten und des verstärken maschinellen Lernens.

Da diese in dieser Arbeit jedoch keine Anwendung finden, werden diese nicht weiter erklärt, sind aber unter der folgenden Quelle bei Interesse aufrufbar:

Machine Learning: Algorithmen, Methoden und Beispiele (datasolut.com)

2.3.2 Funktionsweise Machine-Learning-Modell

In diesem Abschnitt wird erläutert, wie ein Machine Learning Modell funktioniert. Um eine genaue Erkennung der Objekte zu gewährleisten, muss ein Datensatz mit ausreichend Bildern, von den zu erkennenden Objekten, vorhanden sein.

Dieser Datensatz wird in 3 Teile unterteilt.

Tabelle 1: Aufteilung Datensatz

Trainingsdatensatz	**Validierungsdatensatz**	**Testdatensatz**
70-80% der Bilder	10-15% der Bilder	10-15% der Bilder

Beim Trainingsdatensatz handelt es sich um die Bilder, die verwendet werden, um das Modell zu trainieren.

Während des Trainings passt das Modell seine internen Gewichtungen an. Das heisst, das Modell passt sich an, um Zusammenhänge, zwischen den einzelnen Bildern zu erkennen und die Früchte voneinander zu unterscheiden. Es wird versucht, aus Mustern in den verwendeten Beispielen ein Rezept zu erstellen. (Wuttke, kein Datum)

Nach dem Training des Modells wird das Modell mit dem Validierungsdatensatz getestet. Durch diesen Schritt kann die Leistung des Modells mithilfe von Daten überprüft werden, welche in der Testphase nicht verwendet wurden.

Dank des Validierungsdatensatzes können Überanpassungen, welche in der Trainingsphase entstanden sind, eruiert werden.

Nachdem das Model mithilfe des Validierungsdatensatzes optimiert wurde, wird seine endgültige Leistung mithilfe des Testdatensatzes evaluiert.

(Jared Wilber, kein Datum)

2.3.3 künstliches neuronales Netz (KNN)

Ein künstliches neuronales Netzwerk ist ein Modell, welches von der Funktionsweise des menschlichen Gehirns inspiriert ist. In höheren Organismen, wie dem Menschen, findet die Signalübertragung über Nervenzellen statt, welche über Synapsen miteinander verbunden sind.

Die Verbindungen zwischen den einzelnen Nervenzellen sind flexibel und werden stärker, je häufiger sie genutzt werden.

Künstliche neuronale Netzwerke ahmen diese natürlichen neuronalen Netzwerke mithilfe von Algorithmen nach. Sie werden dazu verwendet, komplexe Muster und Zusammenhänge in Daten zu erkennen. (Novustat, 2020)

Die Grundbausteine eines künstlichen neuronalen Netzwerkes sind die Neuronen, welche in verschiedenen Schichten angeordnet sind. Die Neuronen sind über sogenannte Kanten miteinander verbunden. Über diese Kanten läuft der Informationsfluss zwischen den einzelnen Neuronen.

Die Informationsweitergabe erfolgt in drei verschiedenen Schichten: der Eingabeschicht (Input Layer), der verborgenen Schicht (Hidden Layer) und der Ausgabeschicht (Output Layer). Dabei sind immer alle Neuronen einer Schicht mit allen Neuronen der nächsten Schicht verbunden.

In der Eingabeschicht werden Informationen aus der Aussenwelt durch die Neuronen aufgenommen und über die Kanten an die verborgene Schicht weitergeleitet.

Die verborgene Schicht kann aus mehreren Schichten bestehen und ist der Ort, an dem die eigentliche Verarbeitung der Informationen stattfindet. Hier werden komplexe Muster erkannt und Zusammenhänge zwischen den Daten hergestellt.

In der Ausgabeschicht werden die verarbeiteten Informationen schlussendlich als Ergebnis durch die Neuronen ausgegeben.

(studyflix, 2022)

Funktionsweise künstliches neuronales Netz

Damit ein künstliches neuronales Netz in der Lage ist Objekte zu unterscheiden, müssen erstmals Bilder dieser Objekte dem neuronalen Netz zugänglich gemacht werden. Wir als Menschen erkennen auf einem Bild eines Objektes beispielsweise direkt die Farbe und Form des Objektes. Da ein Computer mit diesen Begriffen jedoch nichts anfangen kann, bestimmt er beispielsweise die Farbwerte der Bilder. Diese Farbwerte können dann als Input in das KNN verwendet werden, wo sie von den Eingabeneuronen aufgenommen werden.

Von der Eingabeschicht werden die Information über Kanten weitergeleitet. Jede Verbindung (Kante) zwischen Neuronen ist gewichtet. Wenn die Eingabewerte über die Kanten an das nächste Neuron weitergeleitet werden, werden die Gewichtungen auf die Eingabewerte angewendet. Genauer gesagt, wird der Eingabewert mit dem Wert der Gewichtung multipliziert. Somit erhalten wir die gewichteten Eingaben.

Wenn mehrere Kanten mit verschiedenen Eingabewerten auf ein einzelnes Neuron führen, werden die gewichteten Eingabewerte addiert und man erhält die gewichtete Summe.

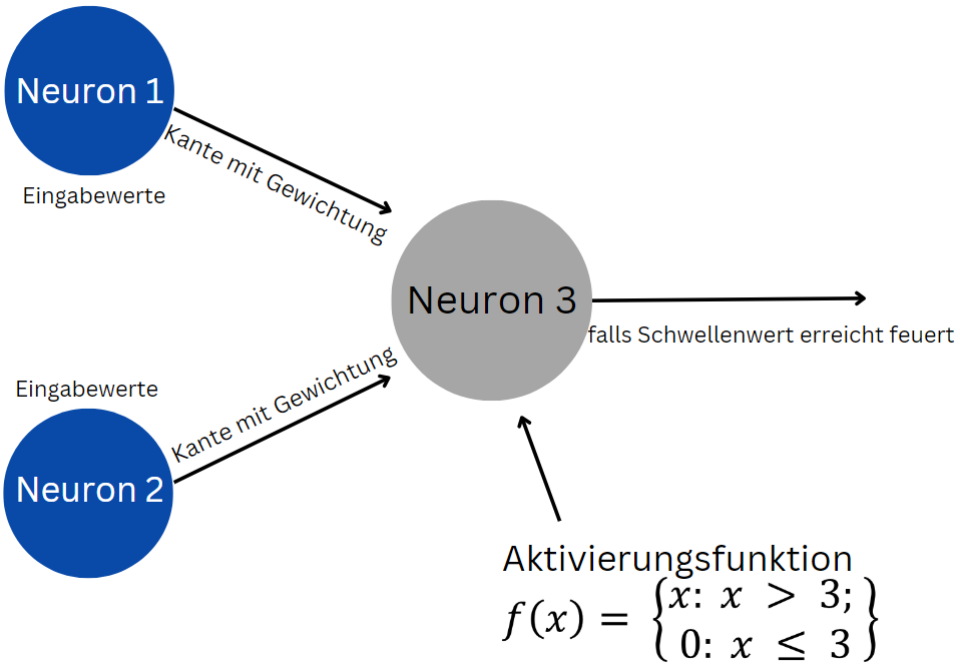

Abbildung 18: Funktionsweise KNN (eigene Abbildung)

Die gewichtete Summe durchläuft dann eine Aktivierungsfunktion, die bestimmt, ob die gewichtete Summe gross genug ist, damit der Informationsimpuls weitergeleitet wird.

Eine häufige Aktivierungsfunktion bilden Schwellenwertfunktionen.

Im obigen Beispiel würde der Informationsimpuls nur weitergeleitet werden, wenn die gewichtete Summe grösser als 3 ist. Dieser Prozess wird so oft wiederholt, bis ein Wert in der Ausgabeschicht als Ergebnis ausgegeben wird. (studyflix, 2022)

2.3.4 Convolutional Neural Network (CNN)

Convolutional Neural Networks sind eine Art künstlicher neuronaler Netzwerke, die speziell für die Bildverarbeitung geeignet sind.

Im Gegensatz zu herkömmlichen neuronalen Netzwerken, bei denen jeder Pixel als eigenständige Eingabe behandelt wird, setzen CNNs spezielle Schichten ein, um Merkmale in Bildern effizient zu identifizieren und zu erfassen.

CNNs bestehen in der Regel aus 3 Grundkomponenten: Convolutional Layers, Pooling Layers und Fully Connected Layers

Die Convolutional Layers identifizieren und extrahieren wichtige Merkmale in den Eingabedaten, wie Formen, Linien und Kanten.

Die Pooling Layers komprimieren die Eingaben aus den Convolutional Layers. Dadurch wird der Speicherbedarf und die Rechenzeit des Modells gesenkt.

(DataScientist, 2023)

Schlussendlich nutzen die Fully Connected Layers die erkannten Merkmale, um eine endgültige Erkennung und Unterscheidung der Objekte zu ermöglichen. (Litzel, 2019)

2.4 Programmierung

2.4.1 Programmcode

Ein Programmcode ist eine Abfolge von Anweisungen, die in einer bestimmten Programmiersprache geschrieben sind. Die Anweisungen beschreiben die spezifischen Aktionen, welche ausgeführt werden müssen, um eine Aufgabe zu erledigen. Der Programmcode des FrubotAIs ist in der Programmiersprache Python geschrieben.

2.4.2 Objektorientierte Programmiersprache

Python ist eine objektorientierte Programmiersprache. Das heisst, dass es eine Struktur von Klassen und Objekten gibt. (Dalwigk, 2021)

Klassen

Klassen dienen als Vorlagen für Objekte und fungieren als Baupläne für diese. Sie definieren die Eigenschaften und Methoden, die ein Objekt einer bestimmten Klasse besitzt. Unter Methoden versteht man Aktionen, welche auf dem Objekt ausgeführt werden können.

Objekte

Objekte werden von der Klasse instanziiert, sie verwenden also die Klassen als Vorlage. Die Objekte haben spezifische Werte für die Eigenschaften der Klasse. (Wikipedia, 2022)

Ein anschauliches Beispiel zur Erklärung des Verhältnisses zwischen Klassen und Objekten ist ein Auto. Die Klasse "Auto" legt etwa die Eigenschaften Marke, Modell, Farbe und Anzahl Türen fest.

Das Objekt "Auto1" ist ein Fiat 500 mit der Farbe Rot und zwei Türen.

Das Objekt "Auto2" ist ein Mercedes E 200 d mit der Farbe Grau und vier Türen.

2.4.3 Methoden und Funktionen

Entscheidend, um den Überblick zu behalten, ist das Gliedern des Codes. Ansonsten erhält man einen langen und unübersichtlichen Programmcode. Ein Mittel zur Verbesserung der Struktur ist das Einsetzen von Methoden. Diese können ausgelagert werden. Das heisst, dass die Methode unter ihrem Namen im Code aufgerufen werden kann, ohne dass der ganze Inhalt erneut codiert werden muss. Methoden sind also Sammlungen von Befehlen. Den Vorteil von Methoden sieht man in diesem Beispiel der Klasse Matura23Utils: (w3schools, kein Datum)

Code zum Aufrufen der Methode:

```
objectInfoList = Matura23Utils.getDetectedObjectInfoList(img, results, CAMERA_WIDTH, CAMERA_HEIGHT)
```

Abbildung 19: Code für Aufruf der Methode

Code in der Methode:

```
@staticmethod
def getDetectedObjectInfoList(img,results,cameraWidth,cameraHeight):
    objectInfoList = []
    excludedFruitList = ['zitrone']
        # in results ist die Liste der erkannten Objekte
    if (len(results) > 0 and results[0] is not None):
        # wir gehen über alle Elemente in der Liste und erstellen ein Objekt mit den Koordinaten und anderen Infos
        for i in range(len(results)):
            eachObjectInfo = {}
            eachObject = results[i]
            # Zitrone ausklammern
            currentLabel = Matura23Utils.labels_map[eachObject['class_id']]
            if currentLabel not in excludedFruitList:
                # Convert the bounding box figures from relative coordinates
                # to absolute coordinates based on the original resolution
                ymin, xmin, ymax, xmax = eachObject['bounding_box']
                xmin = int(xmin * cameraWidth)
                xmax = int(xmax * cameraWidth)
                ymin = int(ymin * cameraHeight)
                ymax = int(ymax * cameraHeight)
                #print("[xmin: {} | xmax: {} | ymin: {} | ymax {}]".format(xmin,xmax,ymin,ymax))
                # Nullpunkt ist oben links im Quadrat
                try:
                    eachObjectInfo['x'] = xmin # x-coordinate
                    eachObjectInfo['y'] = ymin # y-coordinate
                    eachObjectInfo['width'] = xmax-xmin # width
                    eachObjectInfo['height'] = ymax-ymin # height
                    eachObjectInfo['class_id'] = int(eachObject['class_id'])  # object class_id default: 0
                    eachObjectInfo['label'] = Matura23Utils.labels_map[eachObject['class_id']]  # object label default 'None'
                    eachObjectInfo['count'] = len(results) # number found objects default: 0
                except:
                    print("somethiing went wrong!")
                # Objektinfo der Liste hinzufügen
                objectInfoList.append(eachObjectInfo)
    # else:
        # print('no object found')
    return objectInfoList
```

Abbildung 20: Code in Methode (eigene Abbildung)

Funktionen

Auch Funktionen können helfen, einen Programmcode übersichtlicher zu machen. Wenn eine Funktion definiert ist, kann man diese aufrufen, ohne den ganzen Code noch einmal niederzuschreiben. Im Gegensatz zur Methode ist die Funktion keiner spezifischen Klasse zugeordnet.

2.4.4 Rückgabebefehl "return"

Mit dem Befehl "return", werden Werte aus Funktionen und Methoden zurückgegeben. Wenn ein "return" ausgegeben wird, springt das Programm aus der laufenden Funktion. (w3schools, kein Datum)

Im angehängten Beispiel gibt der FrubotAI zurück, dass er eine Frucht gefunden hat sowie die Informationen über die Frucht. Um das System stabiler zu gestalten, muss die Frucht mehrmals gesehen werden, bevor die Informationen ausgegeben werden.

```
foundCount = foundCount + 1
if foundCount >= maxFoundCount: # erst wenn es mehrmals gefunden wurde
    print("[doSearchFruit] object found:{}".format(foundObjectInfo))

    words = ["I see a {}".format(foundObjectInfo['label'])]
    if (Matura23Utils.speakInGerman):
        words = ["Ich sehe eine {}".format(foundObjectInfo['label'])]
    Matura23Utils.speakOut(words, Matura23Utils.speakInGerman)

    return True, foundObjectInfo
```

Abbildung 21: Befehl "return" (eigene Abbildung)

2.4.5 Befehl "break"

Ein "break" wird verwendet, um eine Schlaufe vorzeitig zu beenden. In diesem Beispiel aus dem Programmcode des FrubotAIs, beendet der Roboter die Annäherung an die gefundene Frucht, sobald er ein Objekt in gewünschter Nähe misst. (w3schools, kein Datum)

```
if directionX == 'forward':
    # Echosensor ansteuern um Stueckweise an Objekt zu gelangen
    targetDistance = 20
    distance = round(px.ultrasonic.read(), 2)
    if distance <= targetDistance:
        nearFruit = True
        print('in front of object')
        break
```

Abbildung 22: Befehl "break" (eigene Abbildung)

2.4.6 Verzweigungen

Einfache Bedingungsprüfung: if-else

Eine if-Verzweigung ermöglicht es, eine Anweisung nur unter bestimmten Bedingungen auszuführen. Wenn diese Bedingung erfüllt ist, wird der zugehörige Codeblock ausgeführt. Andernfalls wird er übersprungen. Falls eine alternative Anweisung ausgeführt werden soll, wenn die Bedingung nicht erfüllt ist, kann ein else-Zweig definiert werden. (w3schools, kein Datum)

In diesem Fall wird beispielsweise die Variable "foundCount" nur erhöht, wenn die Frucht gefunden wurde. Ansonsten wird die "notFoundCount" Variable um eins erhöht.

```
if found == True:
    foundCount = foundCount + 1
else:
    notFoundCount = notFoundCount + 1
```

Abbildung 23: if-else Bedingung (eigene Abbildung)

Komplexe Bedingungsprüfung: if-elif-else

If-elif-else-Verzweigungen bieten eine erweiterte Möglichkeit, mehrere Bedingungen zu prüfen. Wenn die Bedingung im if-Zweig nicht erfüllt ist, geht der Code zum elif-Zweig weiter. Wenn die elif-Bedingung zutrifft, wird der entsprechende Codeblock ausgeführt. Ansonsten wird der else-Zweig ausgeführt. Es lassen sich beliebig viele elif-Verzweigungen hinzufügen, um mehr Abfragen einzubauen. (w3schools, kein Datum)

Dieses Beispiel einer if-elif-else Verzweigung steuert, welche Ausrichtung die Vorderräder des Roboters annehmen sollen. Je nachdem ,wie der Winkel zum gefundenen Objekt ist, richtet der Roboter den Servomotor der Vorderräder aus.

```
if angleToObject > 5:
    angleToObject = 20
elif angleToObject < -5:
    angleToObject = -20
else:
    directionX = 'forward'
```

Abbildung 24: if-elif-else Bedingung (eigene Abbildung)

2.4.7 Schlaufen

Schlaufen werden für wiederkehrende Anweisungen verwendet. Dabei wird zwischen while- und for-Schlaufen unterschieden. Bei for-Schlaufen wird festgelegt, wie oft über die Liste iteriert werden soll. Im Gegensatz dazu wiederholen sich while-Schlaufen, solange die Bedingung wahr ist. (w3schools, kein Datum)

In diesem Beispiel einer for-Schlaufe wird der Winkel der Vorderradsteuerung des Roboters von rechts nach links gedreht. Hierzu wird der Winkel mit minus 1 addiert, bis er auf -35 eingestellt ist.

```
for angle in range(35, -35, -1):
    px.set_dir_servo_angle(angle)
    time.sleep(0.01)
```

Abbildung 25: for-Schlaufe (eigene Abbildung)

Bei diesem Beispiel einer while-Schleife wartet der Roboter so lange, bis die Frucht vor ihm entfernt wird. Wenn die Frucht nicht mehr von dem Ultraschallmodul wahrgenommen wird, ändert sich die Variable "FruitInFront" auf "False", sodass der Programmcode nicht mehr in die while-Schleife kommt.

```
# Mit Echosensor warten, bis Objekt entfernt wird
fruitInFront = True
targetDistance = 30
fruitNotInFront = 0
while fruitInFront == True:
    distance = round(px.ultrasonic.read(), 2)
    time.sleep(0.1)
    if distance > targetDistance:
        fruitNotInFront = fruitNotInFront + 1

    if fruitNotInFront > 5:
        fruitInFront = False
```

Abbildung 26: while-Schlaufe (eigene Abbildung)

2.5 Git und GitHub

Git und GitHub sind zwei eng miteinander verwandte Werkzeuge und bilden zusammen die Grundlage für eine effiziente Zusammenarbeit in der Softwareentwicklung. Die genauen Funktionsweisen der beiden Hilfsmittel werden nachfolgend erläutert:

2.5.1 Git

Git ist ein Versionskontrollsystem, das dazu dient, Änderungen in Dateien zu verfolgen und die Zusammenarbeit zwischen mehreren Personen zu koordinieren. Git ist ein verteiltes Versionskontrollsystem. Das bedeutet, dass Git die Daten nicht nur auf einem zentralen Server speichert. Stattdessen klont jeder Mitarbeitende eine Kopie eines Repositories, welches eine Sammlung von Dateien ist. Somit hat jeder Mitarbeitende den gesamten Verlauf des Projektes auf seinem Computer. (Gaba, 2023)

Git hilft also dabei, einen Überblick über die Änderungen an einem Projekt zu behalten. Macht man etwa einen fatalen Fehler, der nicht mehr so einfach gelöst werden kann, so kann man einfach eine Version zurückgehen und einfacher herausfinden, was diesen Fehler verursacht hat.

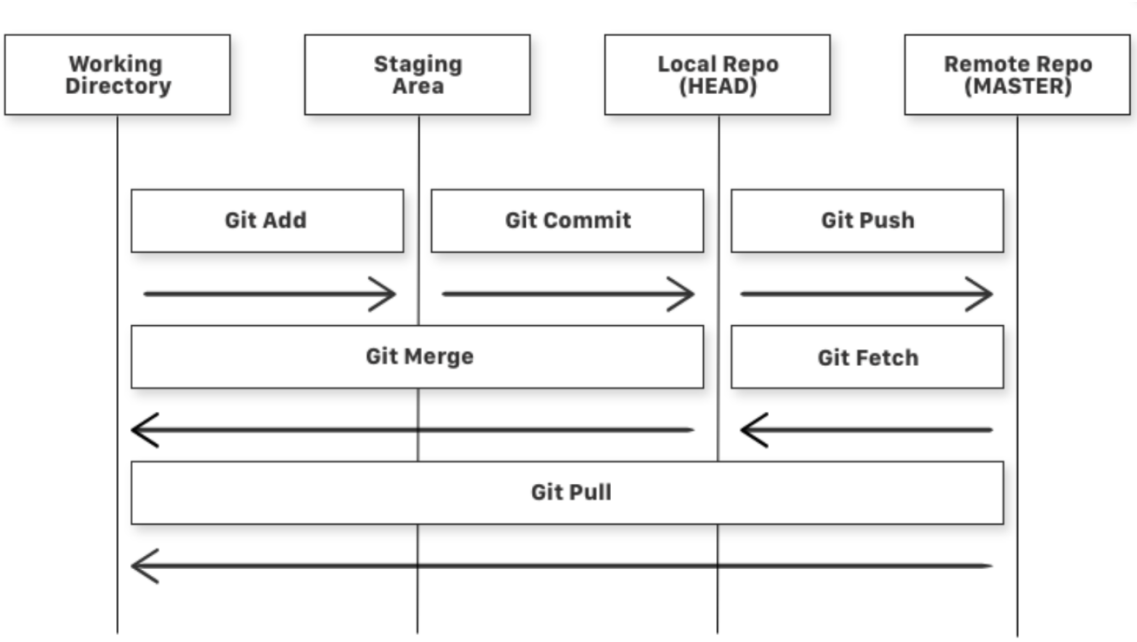

Diagram of a simple Git Workflow

Abbildung 27: Git Workflow (Venkatesan, 2019)

2.5.2 Git Workflow

(Venkatesan, 2019)

Working-Directory

Das Working-Directory ist der Ausgangspunkt. Hier werden Dateien erstellt, bearbeitet und getestet. Dieses Verzeichnis enthält den aktuellen Stand des Projektes.

Add

Nachdem Änderung im Working-Directory vorgenommen wurden, muss entschieden werden, welche dieser Änderungen für das Projekt relevant sind. Die relevanten Änderungen werden mit dem Befehl "Add" in die Staging-Area übernommen.

Staging-Area

In der Staging-Area können die Änderungen vor der endgültigen Speicherung im lokalen Repository nochmals überprüft werden.

Commit

Mit dem Commit werden die Änderungen ins lokale Repository übernommen. Jeder Commit enthält eine Commit-Nachricht, in der beschrieben wird, was geändert wurde. Dies ermöglicht eine klare Nachverfolgung des Projektverlaufs.

Local-Repository

Im Local-Repository werden alle Commits gespeichert, die im Verlauf des Projekts erstellt wurden. Das Local-Repository dient als Archiv und ermöglicht es, zu früheren Versionen zurückzukehren und diese einzusehen.

Push

Um die Zusammenarbeit zu erleichtern, kann man mit dem Befehl "push", die commiteten Daten auf ein externes Repository hochladen. Das externe Repository befindet sich dann auf einer Plattform wie Github und dient als zentraler Speicherort für das Projekt. Von dort aus können alle Berechtigten auf das Projekt zugreifen.

Fetch

Mit dem "Fetchen“ können die Änderungen aus dem externen Repository abgerufen und mit dem lokalen Repository verglichen werden.

Merge

Nachdem die gefetchten Änderungen überprüft wurden, kann man sie mit dem Befehl "Merge“ in den eigenen Arbeitsbereich übernehmen.

Pull
Der Befehl „Pull“ kombiniert das "Fetchen" und "Mergen".

2.5.2 GitHub
GitHub ist eine Plattform, die auf dem oben beschrieben Git-Versionisierungssystem basiert. Dies externen Repositorys können über GitHub verwaltet werden.

3.0 Methoden

In diesem Kapitel wird Entwicklungsprozess des Projektes dargestellt. Die Entwicklung unseres FrubotAIs basiert nebst dem PiCar-X-Kit von SunFounder (SunFounder, kein Datum), auch auf den Anleitungen und Überlegungen von EdjeElectronics. (EdjeElectronics, kein Datum)

3.1 Raspberry Pi

Der Raspberry Pi 4b wurde als "Gehirn" des Roboters ausgewählt. Die Wahl ist auf verschiedene Gründe zurückzuführen.

Einerseits ist der Raspberry Pi relativ klein und findet somit auf vielen Roboter-Chassis Platz.

Ausserdem bietet er trotz seines kostengünstigen Preises eine grosse Palette an Möglichkeiten und Anwendungsgebieten. Die Unterstützung von Programmen wie TensorFlow und OpenCV, welche zur Erkennung von Objekten notwendig sind, stellt sich als weiteres entscheidendes Merkmal dar. Zusätzlich gibt es zum Thema Raspberry Pi eine aktive Community mit vielen Tutorials und Foren, welche sich im Verlauf der Arbeit als hilfreich herausgestellt haben.

3.2 PiCar-X

Als Grundlage des Roboters wurde sich für das Roboterset „PiCar-X“ entschieden.

Die Auswahl für den PiCar-X ist auf folgende Faktoren zurückzuführen. Erstens bietet das PiCar-X Roboterset eine breite Ausstattung an Sensoren, welche gut zur Erkennung und Unterscheidung von Objekten geeignet sind.

Ausserdem stellt der Anbieter SunFounder viele Beispiele gratis zur Verfügung. Diese Beispiele eignen sich gut als Orientierungshilfe für Einsteiger in die Robotik.

3.3 Datensammlung

Ein grosser Datensatz ist die Grundlage eines Machine-Learning-Modells. Daher ist es von grosser Bedeutung, dass ein stabiles Fundament aus Trainingsdateien vorhanden ist. In diesem Projekt wurden etwa 750 Bilder verwendet (im Anhang). Da sich als Ziel gesetzt wurde, eigene Bilder zu verwenden, war der erste Schritt die Aufnahme der Bilder. Dabei gibt es einige Dinge zu beachten. Folgend werden diese kurz erläutert:

3.3.1 Verschiedene Hintergründe
Es ist wichtig, dass Bilder in verschiedenen Umgebungen, mit unterschiedlichen Hintergründen aufgenommen werden. Dadurch wird dem Modell ermöglicht, dass es sich an verschiedene Szenarien anpassen kann.

3.3.2 Negative und positive Beispiele
Ein weiterer Aspekt, der berücksichtigt werden sollte, ist, dass auf den Bildern der Trainingsdaten einerseits die Objekte enthalten sind, welche später erkannt werden sollen (Positive). Andererseits sollen in diesen Bildern aber auch Objekte enthalten sein, welche nicht klassifiziert und erkannt werden sollen. (Negative)

Diese Methode stellt sicher, dass das Machine-Learning-Modell in der Lage ist, zwischen relevanten und irrelevanten Merkmalen in den Bildern zu unterscheiden, was die Genauigkeit der Erkennung erhöht.

3.3.3 Unterschiedliche Belichtung
Zudem muss beachtet werden, dass die Bilder unter variierenden Lichtverhältnissen aufgenommen werden. Dies ermöglicht eine zuverlässigere Erkennung der Früchte, unabhängig von den aktuellen Lichtverhältnissen. Daher sind die Bilder in dieser Arbeit unter verschiedenen Lichtquellen aufgenommen, darunter Sonnenlicht und einer dimmbaren Lampe.

3.3.4 Bilder mit Raspberry Pi Kamera und Handy
Da eine grosse Anzahl an Bildern geschossen werden muss, sollte eine handliche Kamera gewählt werden. In diesem Projekt wurde für den Grossteil der Fotos eine iPhone 11 Pro Kamera verwendet. Zusätzlich wurden Fotos mit der Raspberry Pi Kamera V2 ins Modell integriert, sodass das Modell bereits mit Fotos der Kamera trainiert wird, mit welcher später die Objekte erkannt werden sollen.

3.4 Datenverarbeitung

Da das Machine-Learning-Modell in dieser Arbeit mithilfe von überwachtem Lernen trainiert wird, müssen die Daten (Bilder) vor ihrer Integration in das Modell gelabelt werden. (siehe Kapitel 2.3.1 Überwachtes maschinelles Lernen)

3.4.1 Labeln

Für das Labeln der Bilder wurde das Tool LabelImg verwendet. Mithilfe von LabelImg können rechteckige Boxen um die einzelnen Früchte gezeichnet und mit ihrem entsprechenden Label (Namen) versehen.

Beim Labeln ist zu beachten, dass die Boxen so eng wie möglich um die jeweilige Frucht gezeichnet werden. Somit wird erreicht, dass der Anteil Frucht pro Fläche möglichst gross ist und somit ein möglichst kleiner Anteil an Hintergrund oder anderen Objekten enthalten ist, welche das Modell verfälschen können.

Zudem ist wichtig, dass "schlechte Bilder“ des Datensatzes gelöscht werden. Unter "schlechten Bilder" versteht man Aufnahmen, auf denen eine Frucht nur teilweise sichtbar ist, oder Früchte sich gegenseitig überlappen. Solche Bilder können zu fehlerhaften Ergebnissen führen und sollten daher aus dem Datensatz entfernt werden.

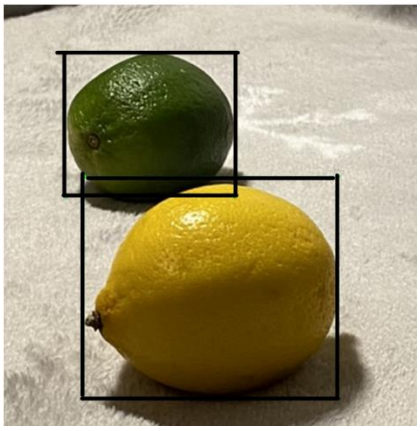

Abbildung 28: überlappende Rahmen (eigene Abbildung)

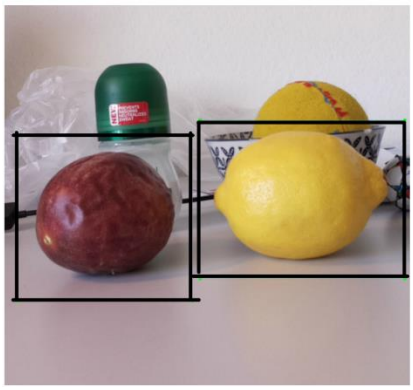

Abbildung 29:ungenau gelabeltes Bild (eigene Abbildung)

3.5 Testen des Objekterkennungsmodells

3.5.1 Genauigkeit des Modells

Im verwendeten Colab-Notizbuch werden die Modelle zweimal getestet. Einmal wird das Modell visuell mit einer Ausgabe trainiert, um einen Eindruck davon zu verschaffen, wie das Modell funktioniert.

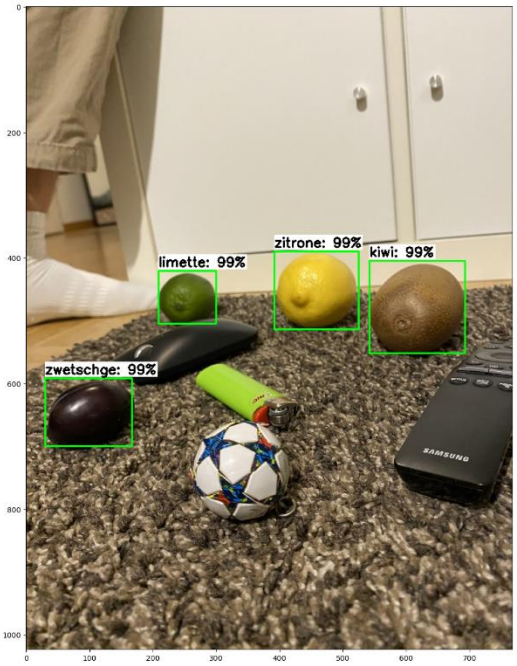

Abbildung 30: Genauigkeit des Modells (eigene Abbildung)

Im zweiten Schritt werden die Bilder des Testdatensatzes verwendet, um die Genauigkeit des Modells zu berechnen. Diese Ergebnisse werden in der Darstellung der Ergebnisse gezeigt.

Dazu wird der mAP-Wert verwendet. Der Mean-Average-Precision-Wert drückt aus, wie genau das Modell ein Objekt erkennt. Je höher der mAP-Wert, desto präziser und zuverlässiger erkennt das Modell die bestimmten Objekte.

Um die Genauigkeit in einer Zahl auszudrücken, werden zwei Faktoren zusammengesetzt. Einerseits wird die Anzahl der korrekt erkannten Objekte berücksichtigt und andererseits die Genauigkeit der Boxen, welche um die gefundenen Objekte platziert werden.

Faktor 1: Korrekt erkannte Objekte:

Bei der Quantifizierung der Treffer wird nach vier Fällen unterschieden. Es gibt zwei Arten von korrekten Treffern und zwei Arten von falschen Treffern. Ein korrekter Treffer liegt vor, wenn das Modell ein Objekt markiert, wo sich eines befindet. Als korrekter Treffer zählt ebenfalls, wenn das Modell keine Objekte markiert, wo sich keine gesuchten Objekte befinden. Die zwei möglichen Typen von falschen Treffern sind, wenn ein

gesuchtes Objekt markiert wird, obwohl keines vorhanden ist oder wenn ein gesuchtes Objekt vorhanden ist, das Modell es aber nicht sieht.

Faktor 2: Genauigkeit der Box:

Der zweite Faktor zur Bestimmung des mAP-Wertes ist die Genauigkeit, mit der das Modell die Boxen um die gesuchten Objekte zeichnet. Üblicherweise wird ein Schwellenwert festgesetzt, bei dem ein Objekt als korrekt umrahmt gilt. Dieser liegt beispielsweise bei 50%.

3.5.2 Testumgebung

Als Testumgebung des Roboters wurde ein umgekehrtes Tischtuch verwendet, dessen Unterseite weiss ist und eine gute Haftung für den Roboter bietet. Der weisse Untergrund ermöglicht dem Roboter, die Früchte einfacher zu erkennen, weil das weisse Material einen starken Kontrast zu den Früchten bildet.

Zudem wurde der Roboter unter verschiedenen Lichtbedingungen getestet, um herauszufinden, in welchem Licht der Roboter die Früchte am genausten erkennt. Analog zum Erstellen des Datensatzes wurde dazu das natürliche Sonnenlicht sowie eine dimmbare Lampe verwendet.

3.5.3 Test physische Erkennung

Die physische Erkennung wurde unter Verwendung von drei verschiedenen Kameras und zwei Betriebssystemen getestet.

Die getesteten Kameras umfassen die Raspberry Pi V2 und V3 Kameras, die auf dem PiCar-X Roboterset montiert werden und am Raspberry Pi angeschlossen sind.

Für die Pi V2 Kamera konnten sowohl das ältere Betriebssystem "Buster" sowie das neuere Betriebssystem "Bullseye" getestet werden. Die neuere Kamera Pi V3 konnte nur mit dem Betriebssystem "Bullseye" getestet werden, da sie mit dem älteren Betriebssystem "Buster" inkompatibel ist. Zusätzlich wurde das Modell auch unter Verwendung der Kamera eines Surface Pro 7 getestet.

3.6 Erstellung eines Machine Learning Modells

3.6.1 TensorFlow und Colab

TensorFlow

Um den Datensatz zu trainieren, braucht man eine Plattform für maschinelles Lernen. Eine Plattform, welche sich gut dafür eignet, ist TensorFlow von Google. TensorFlow ermöglicht das Training von komplexen Machine Learning Modellen und hat eine aktive Community. Der Begriff "Tensor" leitet sich von den mehrdimensionalen Tensoren (Datenstrukturen) ab, die verwendet werden, um Berechnungen beim Training neuronaler Netze durchzuführen.

Colab

Durch ebendiese aktive Community von TensorFlow ist es möglich, ein Colab Notebook von EdjeElectronics als Vorlage zu verwenden, für das Trainieren des Datensatzes. Ein Colab Notebook ist ein interaktives Skript, welches ermöglicht, Python Code auszuführen.

Unter folgenden Quellen ist das Colab Notebook zu finden.

Matura23_Train_TFLite2_Object_Detction_Model.ipynb - Colaboratory (google.com)

Colab ist ein Service von Google, welcher temporär Rechner mit hoher GPU-Leistung zur Verfügung stellt. Auf diesen Rechnern lässt sich das GPU-aufwendige Trainieren von Daten für Machine-Learning-Modelle effizient umsetzen.

3.6.2 Datensatzaufteilung

Wie in der Theorie zum Machine Learning erwähnt (siehe Kapitel 2.3.2 Funktionsweise Machine-Learning-Modell), werden die Bilder in drei Sätze aufgeteilt. Die drei Rubriken sind Trainieren, Validieren und Testen. Der Trainingssatz wird zum Trainieren verwendet, der Validierungssatz dient dem Zwischencheck während des Trainings, und der Testsatz wird am Ende verwendet, um die Leistungen des Modells zu quantifizieren.

3.6.3 Training

Im Trainingsprozess wird das neuronale Netz zur Objekterkennung trainiert. Um die Dauer und Intensität des Trainings anzupassen, stehen zwei Parameter zur Verfügung: die Anzahl der Schritte und die Grösse der Trainingseinheiten. Die Anzahl der Schritte legt fest, wie oft das Modell die gesamten Trainingseinheiten durchläuft. Die Grösse der Trainingseinheiten bestimmt, wie viele Dateien bei jeder Durchführung verarbeitet werden. Standardmässig sind dies 40‘000 Schritte mit einer Trainingsgrösse von 16.

Um das Modell, auf die Erkennung von Früchten zu trainieren, muss ein Trainingsmodell verwendet werden. Das "SSD-MobileNet-V2-FPNLite" eignet sich bestens für das Trainieren eines Modells zur Erkennung von Früchten. Dieses Convolutional Neural Network (CNN) Modell ist so ausgelegt, dass es mehrere Objekte auf einem Foto erkennen kann. Ausserdem ist dieses Modell geeignet für eine Anwendung auf Rechnern mit begrenzten Ressourcen, beispielsweise einem Raspberry Pi.

3.6.4 TensorFlow Modell zu TensorFlow Lite komprimieren

Nachdem das Modell trainiert wurde, muss es von einem TensorFlow Modell zu einem TensorFlow Lite Modell komprimiert werden. TensorFlow Lite Modelle sind besser geeignet, um auf Geräten wie dem Raspberry Pi abgespielt zu werden, da das Modell so angepasst wird, dass es weniger Speicher benötigt und die Ausführung schneller läuft.

3.6.5 Erweiterung Coral-USB-Accelerator

Der Coral-USB-Accelerator von Google beschleunigt die FPS des Raspberry Pis. FPS steht für Frames per Second und meint die Anzahl Bilder, die pro Sekunde verarbeitet werden. Mithilfe dieser Erweiterung kann die Zuverlässigkeit des Modells erheblich erhöht werden, da mehr Bilder in der gleichen Zeit verarbeitet werden können. (Coral, 2020)

Abbildung 31: Coral-USB-Accelerator (Coral, 2023)

3.7 Verbindungsmethoden Raspberry Pi und Windows

Die Verbindung des Raspberry Pis und dem Windows-Rechner ermöglicht eine einfachere Steuerung des Raspberry Pis. In dem folgenden Abschnitt wird erläutert, welche Programme dazu verwendet wurden und welche Vorteile diese Programme mit sich bringen.

3.7.1 SSH

SSH ermöglicht die Einrichtung einer direkten und sicheren Verbindung in einem Netzwerk zwischen zwei Computern, im Falle dieses Projekts zwischen einem Windows und einem Raspberry Pi.

SSH funktioniert nach dem Prinzip des Client-Server-Modells. Dies bedeutet, dass der Raspberry Pi die Rolle des Servers einnimmt. Somit stellt er dem Client Dienste bereit. Der Windows-Rechner, der die Rolle des Clients einnimmt, kann auf diese Dienste zugreifen. So ist es möglich, mit dem Windowsrechner direkt auf die Kommandozeile des Raspberry Pis zuzugreifen.

Die Verbindung mit dem Raspberry Pi erfolgt über die Kommandozeile des Windowsrechners. Im Falle dieses Projekts funktioniert die Verbindung wie folgt:

SSH pi@raspberrypi

SSH = Verbindungstyp

Pi= Username

raspberrypi = Gerät

Nach Eingabe des eingestellten Passworts steht die Verbindung. Sobald die Verbindung erfolgreich ist, können Befehle vom Windows-Rechner an den Raspberry Pi gesendet werden, um verschiedene Aufgaben auszuführen. (Loshin, 2021) (Ionos, 2022)

3.7.2 VNC

VNC ist eine plattformübergreifende Bildschirmfreigabetechnologie, die entwickelt wurde, um die Fernsteuerung anderer Computer zu ermöglichen. Ähnlich wie SSH wird VNC in dieser Arbeit verwendet, um den Raspberry Pi von einem Windows-Computer aus zu steuern.

Einer der Hauptvorteile von VNC liegt darin, dass es Benutzern die Möglichkeit gibt, auf die grafische Benutzeroberfläche (GUI) des Raspberry Pis zuzugreifen.

Dies erleichtert insbesondere für Leute, welche weniger mit der Kommandozeile vertraut sind, die Steuerung und Verwaltung des Raspberry Pis erheblich.

3.7.3 Sambashare

Durch Sambashare ist es möglich, direkt vom Windows-Rechner auf das Filesystem des Raspberry Pis zuzugreifen. Dies ermöglicht das einfache Übertragen von Dateien auf den Raspberry Pi. Somit können beispielsweise die TensorFlow Lite Modelle vom Windowsrechner auf den Raspberry Pi übertragen werden.

3.7.4 Git(Hub)

Mithilfe der Plattform GitHub können verschiedene Version abgespeichert werden. Um die Versionen gemäss dem Git-Versionierungssystem überarbeiten zu können, ist ein Git-Client erforderlich. In diesem Projekt wird auf dem Raspberry Pi der integrierte Git-Client von Visual Studio Code verwendet und auf dem Windowsrechner der Git-Client "Sourcetree".

3.8 Programmcode

Der Programmcode dieses Projektes ist in Python geschrieben. In folgendem Kapitel wird der Weg von der Grundidee bis zum fertigen Programmcode beschrieben.

3.8.1 Wissenserwerb Programmieren

Module von SunFounder

Die verschiedenen Module enthalten Beispiele, welche von Sunfounder zur Verfügung gestellt werden. Diese helfen, sich eine Vorstellung zu verschaffen, wie die verschiedenen Komponenten des Roboters funktionieren und zusammenspielen. Die Module sind in drei Gruppen eingeteilt, namentlich "PiCar-X", "Robot-Hat" und "Vilib".

Im Modul "PiCar-X" befinden sich die Beispiele zur allgemeinen Steuerung des Roboters. Darin enthalten ist das Bewegen des Roboters, die Verwendung des Ultraschall- und Grayscale Moduls, sowie die Wiedergabe von Text-to-Speech.

Im "Robot-Hat" Modul ist ersichtlich, wie der Robot-Hat die Komponenten des Robotersets ansteuert und die Informationen der Komponenten an den Raspberry Pi übermittelt.

Das Modul "Vilib" beschreibt die Möglichkeiten der Kamera. Beispielsweise wird gezeigt, wie basierend auf OpenCV und TensorFlow Lite Runtime, ein Machine Learning Modell zur Erkennung von Objekten verwendet werden kann.

Durch die Module von SunFounder ist es möglich, auch mit bescheidenen Programmierkenntnissen, ein Basiswissen über die Programmierung und Steuerung von Sensoren und Motoren zu erlangen.

Internetrecherche zur Implementierung von spezifischen Anweisungen

Beim Programmieren eines eigenen Programmcodes kann es trotz solider Grundkenntnisse vorkommen, dass man unsicher ist, wie man einen Gedanken im Code umsetzt oder wie die Syntax einer bestimmten Anweisung definiert ist. Oft ist es hilfreich, dazu im Internet zu recherchieren. Besonders bewährt hat sich hier die Webseite https://www.w3schools.com/. W3Schools bietet eine thematisch geordnete Ablage in verschiedenen Programmiersprachen.

3.8.2 Grundidee des Programmes

Um ein grösseres Projekt umzusetzen, ist es hilfreich, sich zuerst einen Überblick zu verschaffen und eine klare Vorgehensweise festzulegen. Dies bewährt sich auch beim Implementieren eines Programmcodes. Im Falle dieses Projektes soll der PiCar-X Roboter mithilfe eines selbst trainierten Objekterkennungsmodell Früchte erkennen und mit ihnen interagieren.

Anschliessend ist es wichtig, die erforderlichen Schritte zu bestimmen, um dieses Ziel zu erreichen. Eine Möglichkeit dafür ist es, mit einer grafischen Skizze zu arbeiten. So können die einzelnen Schritte klar ersichtlich und verständlich dargestellt werden.

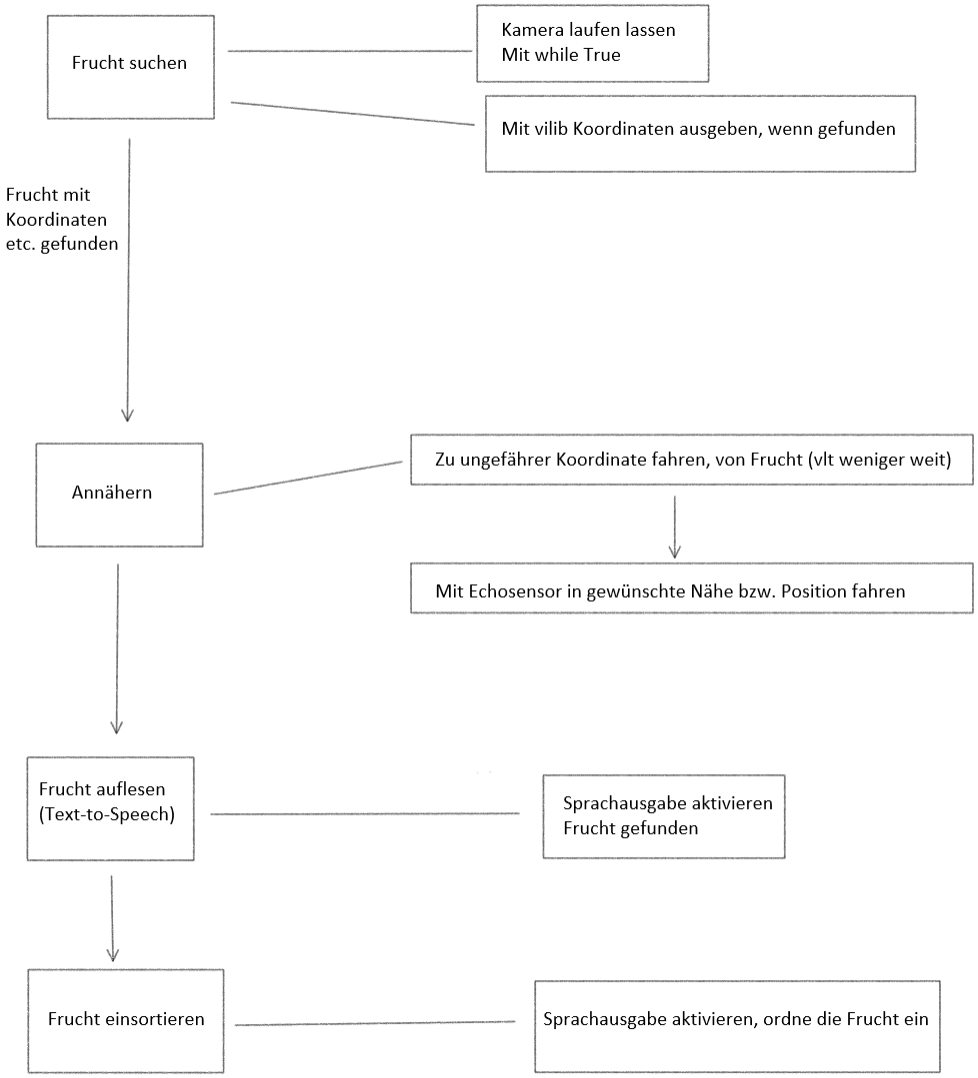

Abbildung 32: Grundidee Programmcode (eigene Abbildung)

3.8.3 Flowchart

Dem Ziel eines Programmcodes, kommt man einen Schritt näher, indem man die Grundidee zu einem Flowchart umstrukturiert. Ein Flowchart ist eine grafische Darstellung eines Programmcodes, die den Ablauf auf einen Blick einsehbar macht.

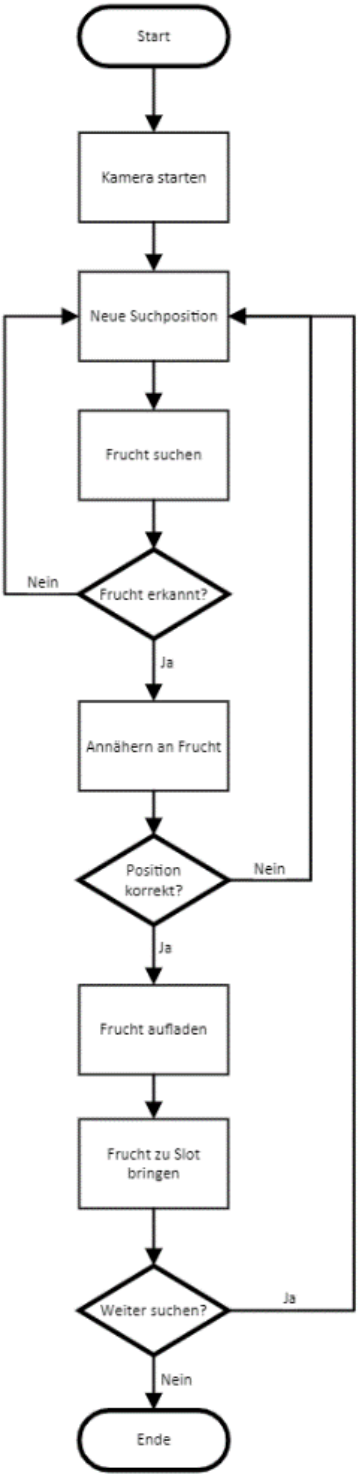

Abbildung 33: Flowchart (eigene Abbildung)

Beim aufgeführten Flowchart werden drei verschiedene Symbole verwendet.

Das Rechteck symbolisiert einen Aktionsblock, der eine Anweisung ausführt.

Die Raute stellt hingegen eine Verzweigung dar. Hier wird eine Entscheidung aufgrund einer Bedingung getroffen.

Die Ovale markieren den Start und das Ende.

Diese Struktur zeigt, wo im Programmcode später Bedingungen eingebaut werden müssen. So kann durch das Verwenden eines Flowcharts der Programmablauf klar dargestellt werden.

3.8.4 Pseudocode

Um den Übergang vom Flowchart zum Programmcode zu erleichtern, kann man den Code zunächst als sogenannten Pseudocode verfassen. Pseudocode ist der Versuch, einen Programmcode in eigenen Worten zu beschreiben. Dieser Zwischenschritt über den Pseudocodes, ist besonders hilfreich für weniger routinierten Programmierer, da er es ermöglicht, den genauen Ablauf des Codes niederzuschreiben, ohne jeden Befehl in einer spezifischen Programmiersprache kennen zu müssen.

Am besten lässt sich der Pseudocode anhand eines Beispiels erklären. Dazu wird der Block "neue Suchposition" aus dem Flowchart in Pseudocode umgeschrieben:

If Roboter sieht eine Frucht?

If Ist es die unterste Frucht?

Wenn ja, soll er diese übernehmen

Ausgeschrieben heisst dies: Es soll geschaut werden, ob die gesehene Frucht die unterste im Bild ist. Obiger Pseudocode ist äquivalent zu folgendem Programmcode in Python:

```
if len(objectInfoList) > 0:
    for i in range(len(objectInfoList)):
        eachObject = objectInfoList[i]
        if eachObject['y'] > foundYCoord and eachObject['label'] == fruitLabel
            foundYCoord = eachObject['y']
            foundObjectInfo = eachObject
```

Abbildung 34: Programmcode des Pseudocodes (eigene Abbildung)

3.8.5 Programmieren in Python

Mithilfe der oben genannten Vorstufen wird nun der Programmcode in der Programmiersprache Python geschrieben.

Code-Editor: Visual Studio Code

Beim Schreiben eines Programmcodes hilft ein Code-Editor. In diesem Projekt wurde der Code-Editor Visual Studio Code von Microsoft verwendet. Code-Editoren bringen mehrere Vorteile mit sich. Einerseits gestalten sie den Code übersichtlicher, mit verschiedenen Funktionen wie Syntaxhervorhebung, Zeilennummerierung oder Codevervollständigung. Andererseits kann der Code durch Debugging besser analysiert werden. Des Weiteren kann der Code mit dem integrierten Git-Client verwaltet werden.

Es ist hilfreich, den Code des Hauptskriptes auszulagern. Im Programm des FrubotAIs wurde dazu die Klasse "Matura23Utils" erstellt, welche den Code der einzelnen Hauptblöcke und weitere Hilfsmethoden enthält. Der Code des Hauptskriptes enthält durch dieses Auslagern nur die Hauptblöcke des Programmes, analog zum Flowchart (siehe Kapitel 3.8.3 Flowchart).

```
while countFound < maxHits:
    img = Vilib.detect_obj_parameter['object_img']
    results = Vilib.detect_obj_parameter['object_results']
    objectInfoList = Matura23Utils.getDetectedObjectInfoList(img, results, CAMERA_WIDTH, CAMERA_HEIGHT)

    # beim ersten mal veraendern wir die Position nicht, an Ort und Stelle suchen
    if countRun > 0:
        Matura23Utils.doGoInNewPosition(px,objectInfoList)

    found,foundObjectInfo = Matura23Utils.doSearchFruits(px,objectInfoList,CAMERA_WIDTH,CAMERA_HEIGHT)
    if found == True:
        nearFruit = Matura23Utils.doGoCloserToFruit(px, foundObjectInfo, CAMERA_WIDTH, CAMERA_HEIGHT)

        # Nur wenn die Frucht gefunden wird und er sich in Position bringen kann
        # bringt er die Frucht in den Slot
        if nearFruit == True:
            countFound = countFound + 1
            Matura23Utils.doPickUpFruit(px,foundObjectInfo)
            Matura23Utils.doSortInFruit(px,foundObjectInfo)
        else:
            countNotFound = countNotFound + 1

    else:
        print('no fruit in this position')

    countRun = countRun + 1

print("countFound:{} | counNotFound:{} | countRun:{}".format(countFound, countNotFound, countRun))

Matura23Utils.doEnd()
```

Abbildung 35: Code Hauptskriptes (eigene Abbildung)

Diese Blöcke wurden nach dem Verfahren von "Trial & Error" implementiert. Das heisst, es wurde ein erster Versuch codiert, danach getestet und falls nötig verbessert.

Um das Prinzip des finalen Programmcodes erklären zu können, eignet sich das Flussdiagramm (siehe Kapitel 3.8.3 Flowchart) Aus diesem Flussdiagramm werden die wichtigsten Blöcke ausgewählt, um ein Bild des Programmcodes aufzuzeigen.

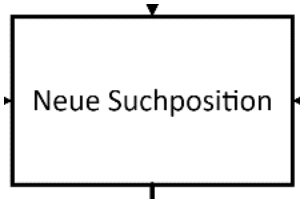

Abbildung 36: Neue Suchposition (eigene Abbildung)

In Flowchart Block "Neue Suchposition"

Im Code Methode "doGoInNewPosition"

In diesem Abschnitt soll sich der FrubotAI-Roboter neu positionieren, falls er keine Frucht findet. Dazu schlägt er die Räder nach rechts ein, fährt rückwärts, schlägt die Räder nach links ein und fährt nach vorne. Anschliessend stellt er die Räder wieder gerade.

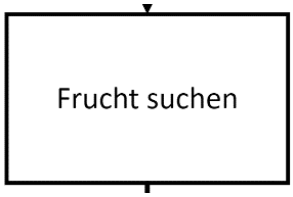

Abbildung 37: Frucht suchen (eigene Abbildung)

In Flowchart Block "Frucht Suchen"

Im Code Methode "doSearchFruits"

In diesem Codeabschnitt soll der Roboter überprüfen, ob sich eine Frucht an seiner aktuellen Position befindet. Um sicherzustellen, dass der Roboter keine falschen Früchte aufgrund von fehlinterpretierten Bildern erkennt, muss die Frucht auf mehreren Bildern bestätigt werden. Analog dazu gibt der Roboter nicht auf, wenn er eine Frucht, die er zuvor gesehen hat, auf einem Bild nicht wiedererkennt. Auf diese Weise kann die Erkennung der Objekte stabiler und zuverlässiger gestalten werden.

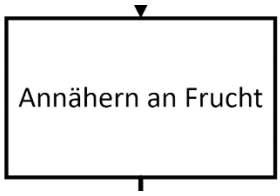

Abbildung 38: Annäherung an Frucht (eigene Abbildung)

In Flowchart Block "Annäherung an Frucht"

Im Code Methode "doGoCloserToFruit"

In diesem Programmteil nähert sich der Roboter an die gefundene Frucht an. Dazu liest er die X-Koordinaten der gefundenen Frucht aus. Liegt die Frucht nicht direkt vor ihm, dreht er die Vorderräder in die entgegengesetzte Richtung zur Frucht und fährt rückwärts, um seine Position zu korrigieren. Wenn sich die Frucht vor ihm befindet, nähert er sich in kleinen Schritten, bis der Ultraschallsensor die Frucht in der gewünschten Nähe identifiziert. Um zu verhindern, dass der Roboter sich endlos ausrichtet, toleriert er eine Abweichung von 5° nach links und rechts, als geradeaus.

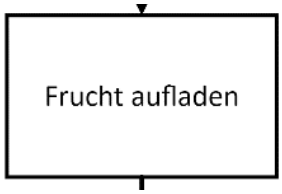

Abbildung 39: Aufladen der Frucht (eigene Abbildung)

In Flowchart Block "Frucht aufladen"

Im Code Methode "doPickUpFruit"

Der Roboter befindet sich vor der Frucht. Er gibt mithilfe von Text-to-Speech wieder, welche Frucht er gefunden hat. Anschliessend wartet er vor der Frucht, bis sie aufgehoben wird. Solange der Ultraschallsensor die Frucht vor dem Roboter erfasst, bleibt er in seiner Position. Sobald die Frucht aufgenommen wird, bedankt er sich für das Auflesen.

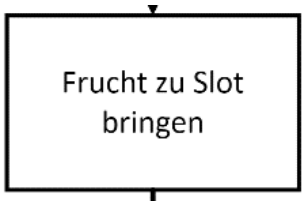

Abbildung 40: bringen zu Slot (eigene Abbildung)

In Flowchart Block "Frucht zu Slot bringen"

Im Code Methode "doSortInFruit"

Als Letztes bittet der Roboter darum, die Frucht an den zugehörigen Ort einzusortieren.

Die obigen Schritte sind die entscheidenden Schritte des Programmcodes für den FrubotAI. Unter folgenden Quellen lässt sich der vollständige Code im GitHub einsehen. Das Hauptprogramm ist unter dem File matura23-robot.py zu finden, während die ausgelagerten Methoden und Funktionen unter matura23utils.py aufrufbar sind.

https://github.com/Adrigit04/matura23-mypicar-x

4.0 Darstellung der Ergebnisse

In diesem Kapitel werden die in der Methode erarbeiteten Ergebnisse dargestellt. Im ersten Teil werden dabei die Ergebnisse der verschiedenen Durchgänge des Machine-Learning-Modells in Form einer Tabelle aufgezeigt. Im zweiten Teil wird die Erkennung mit den verschiedenen Kameras und Betriebssystemen, sowie mit dem Coral-USB-Accelerator dargestellt. Im letzten Teil folgen Bilder und Videos des Roboters, wie er die Zielaufgabe erledigt.

4.1 Ergebnisse des Machine Learning Modells

In den folgenden Tabellen wird die mAP der verschiedenen Runs mit variierenden Voraussetzungen dargestellt. Dabei werden einige Bedingungen aus der Datensatzerstellung und Datensatzverarbeitung aufgegriffen. Die Spalten der "Handykamera" sowie die der "Raspberry Pi V2 Kamera" beziehen sich auf die verwendete Kamera beim Aufnehmen der Bilder. "genau gelabelt", sagt aus, ob der anfangs ungenau gelabelte oder der später überarbeitete Datensatz verwendet wurde. Die Spalte "Negative" gibt an, ob ausschliesslich die zu erkennenden Objekte in den Bildern des verwendeten Datensatzes zu sehen sind oder ob auch andere Objekte in den Bildern enthalten sind.

Tabelle 2: Vergleich der Läufe des Modelles

Run	Charakter	Handy-kamera	Raspberry Pi V2 Ka-mera	Genau gelabelt	Nega-tive	mAP durch-schnittlich
1	Bilder in Zimmer mit Kunstlicht, mit variieren-dem Hinter-grund	Ja	Nein	Nein	Nein	67.43%
2	Bilder unter Sonnenlicht mit weissem Hinter-grund	Ja	Nein	Nein	Nein	74.61%
3	Bilder aus Run 1 und Run 2 kombiniert	Ja	Nein	Nein	Nein	78.68%
4	Bilder aus Run 1 und 2 mit wei-teren Bildern mit negativen	Ja	Nein	Nein	Ja	73.46%
5	Bilder mit Rasp-berry Pi V2 Ka-mera unter Kunstlicht	Nein	Ja	Ja		93.74%
6	Bilder aus Run 4,5 kombiniert	Ja	Ja	Ja	Ja	88.80%

In folgender Tabelle werden neben der durchschnittlichen mAP auch die mAPs der einzelnen Früchte dargestellt.

Tabelle 3: Vergleich mAPs

Run	mAP durch-schnittlich	mAP Zitrone	mAP Limette	mAP Zwetschge	mAP Kiwi	mAP Passions-frucht
1	67.43%	64.85%	67.43%	52.65%	75.29%	69.57%
2	74.61%	76.16%	75.27%	66.53%	76.44%	78.64%
3	78.68%	79.53%	81.08%	72.89%	75.25%	84.67%
4	73.46%	73.46%	68.57%	65.47%	76.72%	78.13%
5	93.74%	98.05%	94.27%	91.00%	96.47%	88.89%
6	88.80%	90.94%	90.27%	80.37%	92.05%	90.39%

4.2 Vergleich Kameras und Betriebssysteme

Im folgenden Abschnitt werden die Resultate der verschiedenen Kameras dargestellt.

Die Tests führen zu verschiedenen Ergebnissen in Bezug auf die Leistungsfähigkeit der verschiedenen Kameras und Betriebssystemen.

Fotos mit Raspberry Pi V2 Kamera und Betriebssystem Buster:

Ohne Coral-USB-Accelerator

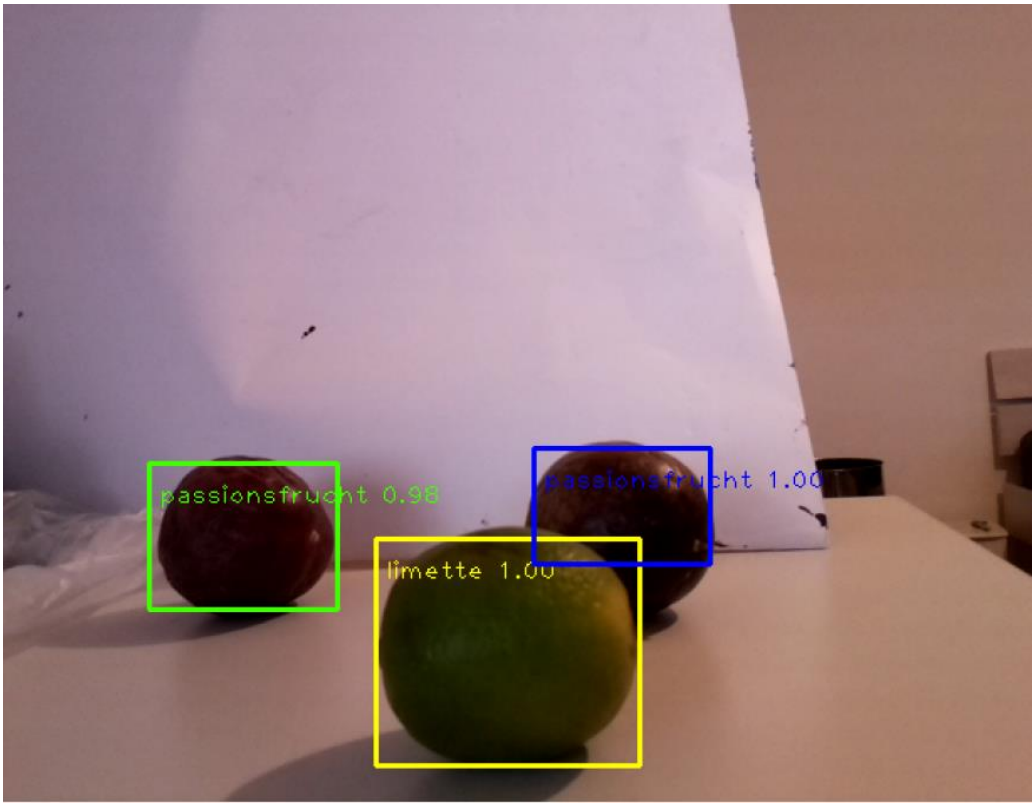

Abbildung 41: Raspberry Pi ohne Coral (eigene Abbildung)

Mit Coral-USB-Accelerator

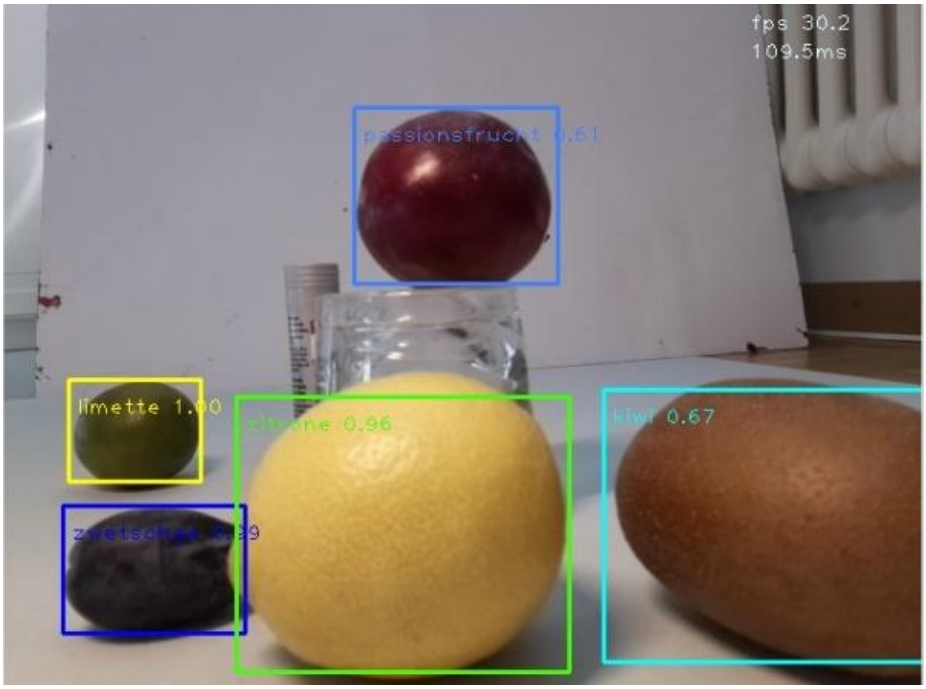

Abbildung 42: Raspberry Pi mit Coral (eigene Abbildung)

Foto mit Microsoft Surface Pro 7 Kamera und Betriebssystem Windows:

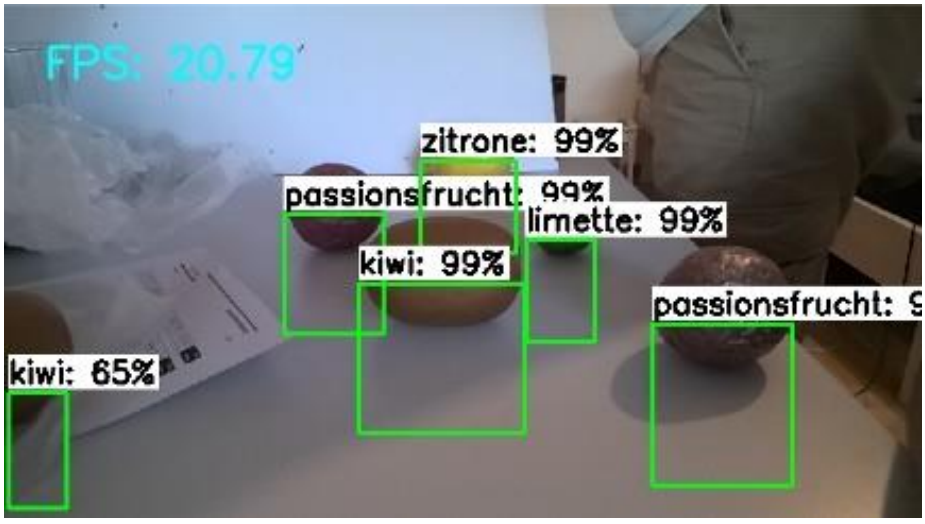

Abbildung 43: Bild Testlauf Windows (eigene Abbildung)

4.3 Bilder und Videosammlung Roboter

Folgende Bilder zeigen das Endprodukt des FrubotAIs:

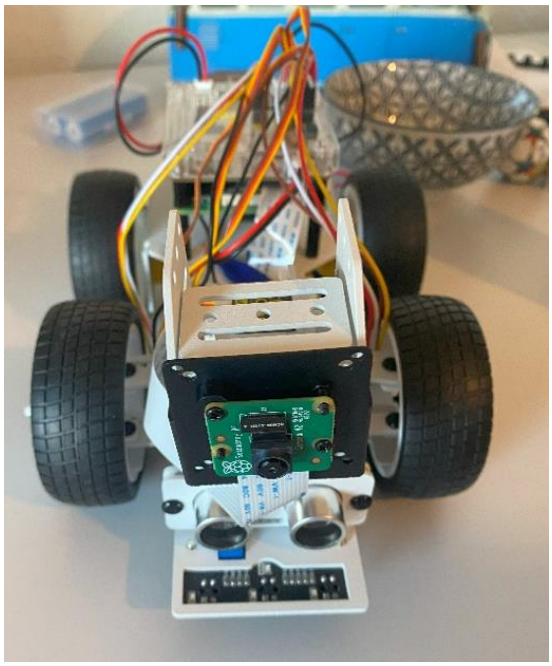

Abbildung 44: PiCar-X von vorne (eigene Abbildung)

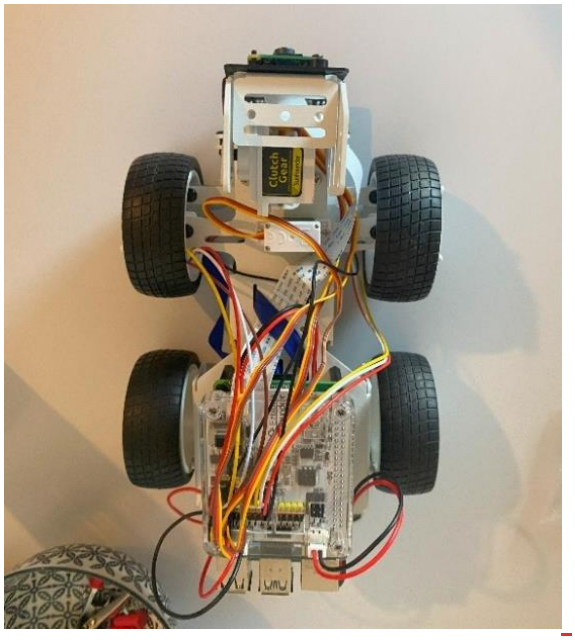

Abbildung 45: PiCar-X von oben (eigene Abbildung)

Der FrubotAI ist in der Lage, unter gewissen Voraussetzungen, alle fünf Früchte zu erkennen. Von vier der fünf Früchte (Zwetschge, Limette, Passionsfrucht, Kiwi) konnten Videos gedreht werden, in welchen der Roboter seine Aufgabe korrekt erfolgreich ausführt. Lediglich bei der Zitrone konnte kein erfolgreicher Lauf festgehalten werden.

Unter folgenden Quellen ist der YouTube-Kanal mit den verschiedenen Videos zu finden, indem der FrubotAI seine vorgesehene Aufgabe erfolgreich erfüllt.

https://www.youtube.com/@Leon____le/videos

5.0 Diskussion der Ergebnisse

Im folgenden Abschnitt erfolgt eine Analyse und kritische Betrachtung der Ergebnisse, welche für den Programmcode, den Roboter und das Machine-Learning-Modell erzielt wurden. Dabei wird auf Schwachstellen eingegangen, welche während den Tests identifiziert wurden. Zudem werden konstruktive Verbesserungsvorschläge präsentiert.

5.1 Ursprüngliche Zielsetzung

Zu Beginn des Projekts wurde sich als Ziel gesetzt, dass der Roboter die Früchte auch an den zugehörigen Ort transportieren kann. An Stelle der Text-to-Speech Ausgaben, sollte der Roboter, durch eine Vorrichtung wie einen Greifarm, die Früchte an einen vorbestimmten Ort platzieren. Dieses Vorhaben erwies sich allerdings als zu umfangreich, um ebenfalls in dieses Projekt eingebaut zu werden.

Das Einsortieren der Früchte wird deswegen vom Roboter delegiert, anstatt dass er es selbst ausführt. Deshalb ist aus dem vorbestimmten Ort, an den der Roboter die Früchte ursprünglich bringen sollte, ein fiktiver Ort geworden.

5.2 Verbesserungs- und Erweiterungsvorschläge

5.2.1 Datensatz

Für eine genauere Erkennung der Früchte ist es notwendig, den Datensatz zu vergrössern. Ein grösserer Datensatz ermöglicht dem Modell, die Merkmale der einzelnen Früchte besser zu erlernen, was zu einer präziseren Erkennung der Früchte führt.

5.2.2 Mehr Negative einbauen

Ausserdem ist es wichtig, mehr Früchte mit Negativen in den Datensatz aufzunehmen. Dies ist notwendig, da das Modell sonst schnell Früchte an Orten erkennt, wo sich gar keine Frucht befindet. Durch das Einbeziehen von negativen Bildern kann das Modell besser zwischen relevanten und irrelevanten Merkmalen unterscheiden.

5.2.3 Automatisierter Labelprozess

Zudem könnte ein maschinelles Lernmodell eingesetzt werden, um die Bilder des Datensatzes automatisch zu labeln. Auf diese Weise kann die Verarbeitung des Datensatzes deutlich effizienter gestaltet werden und dadurch die Menge an Bildern erhöht.

5.2.4 Andere Komponenten

Eine weitere Optimierungsmöglichkeit, um ein genaueres Erkennen der Früchte zu erreichen, liegt darin, eine andere Wahl der Komponenten zu treffen.

Obwohl der PiCar-X eine gute Wahl für Einsteiger in die Robotik ist und bereits relativ komplexe Projekte ermöglicht, reichen die Raspberry Pi V2 Kamera und V3 Kamera meistens nicht aus, um eine konstante Erkennung der Früchte zu garantieren.

Das grösste Problem ist das Annähern an die Frucht. Bei diesem Prozess ist die Erkennung der Früchte sehr inkonstant, wodurch der Roboter manchmal eine gefundene Frucht aus den Augen verliert.

Bei einem Projekt zur Objekterkennung wäre es sinnvoll, eine leistungsstärkere Kamera zu verwenden. Dadurch könnte die physische Fruchterkennung deutlich gesteigert werden. Alternativ kann das PiCar-X-Kit durch den Coral-USB-Accelerator erweitert werden. Dieser steigert die Bilderrate pro Sekunde, somit können mehr Daten pro Zeit verarbeitet werden, wodurch der Roboter die Objekte zuverlässiger erkennt. Mit dieser Erweiterung eignet sich auch das PiCar-X Roboterset zu Aufgaben im Themengebiet der Objekterkennung.

5.2.5 Stabileres Programm

Um die suboptimale Hardware des PiCar-X-Robotersets auszukorrigieren, ist ein gutes Programm nötig. Dieses kann Ungenauigkeiten der Roboterkomponenten ausgleichen und den Roboter seine Aufgabe zuverlässiger erledigen lassen. Das aktuelle Programm für den FrubotAI kaschiert bereits einige Schwächen des Roboters, allerdings könnte durch zusätzlichem Aufwand die Zuverlässigkeit gesteigert werden.

5.2.6 Verwendung von Kunstfrüchten

Ausserdem wäre die Verwendung von Kunstfrüchten sinnvoll. Diese haben den Vorteil, dass die äusserliche Form der Früchte gleich bleibt, da Kunstfrüchte keiner natürlichen Veränderungen ausgesetzt sind.

5.3 Vergleich mit anderen Arbeiten

5.3.1 Coco-Labels vs. eigenes Modell

Coco steht für Common Objects in Context und ist ein weitverbreiteter Datensatz zur Erkennung von verschiedenen Objekten wie beispielsweise Tastaturen, Tassen oder Autos. Der Datensatz enthält 80 Objektkategorien und wurde mit rund 153'000 Bildern trainiert. Folglich ca. 1900 Bilder pro Kategorie (Shah, 2023)

In einem Versuch wurde das Coco-Labels-Modell ebenfalls über die Raspberry Pi V2 Kamera mit dem Betriebssystem Buster getestet (ohne Coral-USB-Accelerator) und erzielte keine guten Leistungen. Die Ergebnisse des Tests zeigen, dass mit der Raspberry Pi V2 Kamera relativ willkürlich Objekte erkannt werden wie auf folgendem Bild zu erkennen ist.

Abbildung 46: Versuch Coco-Label (eigene Abbildung)

Daraus ist schlusszufolgern, dass die anfänglichen Probleme dieser Arbeit beim Erkennen der Früchte nicht im Modell selbst lagen, sondern vielmehr an einer zu schwachen Kamera, die aber aufgrund der Kompatibilität zum PiCar-X und dem Raspberry Pi erforderlich war. Diesen Schwächen der Kamera konnte allerdings entgegengewirkt werden, indem der oben erwähnte Coral-USB-Accelerator verwendet wurde. So konnte trotz allem ein gutes Resultat erzielt werden.

5.4 Fazit

5.4.1 Ende gut, alles gut

Anfangs erkannte das Modell lediglich die Passionsfrucht und Limette (siehe Kapitel 4.2 Vergleich Kameras und Betriebssysteme). Lange konnten nur erfolgreiche Läufe mit der Passionsfrucht erzielt werden. Mit den anderen Früchten haderte das Modell. Erst mit der Erweiterung durch den Coral-USB-Accelerator konnten alle Früchte erkannt werden.

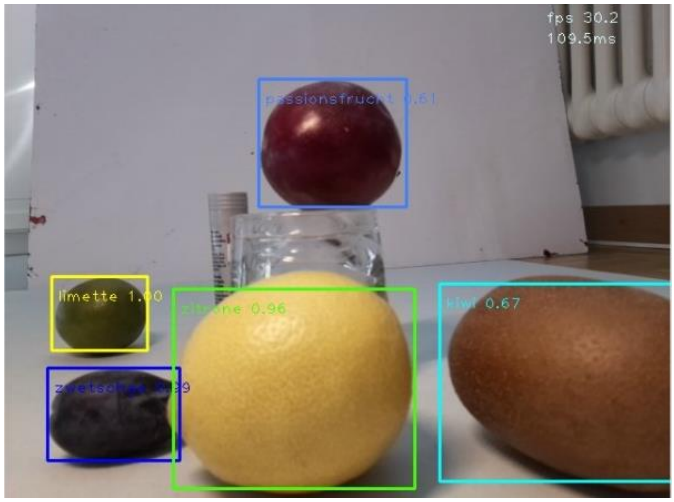

Abbildung 47 Erkennung aller Früchte mit Coral (eigene Abbildung)

Den Hinweis zur Problemlösung lieferte der Versuch, das Machine-Learning-Modell auf dem Windows-Rechner laufen zu lassen. Dadurch wurde festgestellt, dass das Modell sehr gut funktioniert und die anfänglichen Probleme, bei der Erkennung von Früchten, in den Unterschieden zwischen Raspberry Pi und Surface Pro 7 liegen mussten.

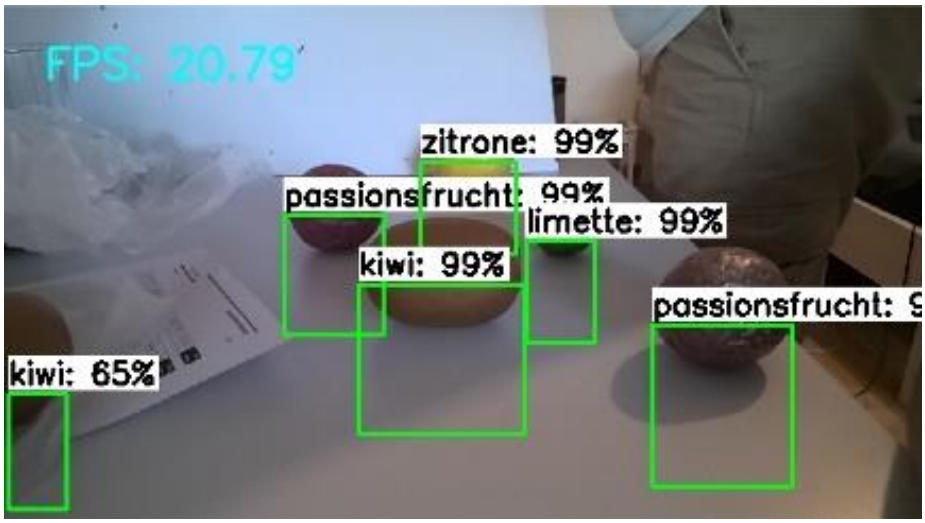

Abbildung 48: Erkennung aller Früchte mit Windo (eigene Abbildung)

Da ein zentraler Unterschied der Rechner ist, wie viele Bilder pro Sekunde sie verarbeiten können, liegt der Coral-USB-Accelerator als Problemlöser auf der Hand.

5.4.2 "Problemfall Zitrone"

Die Arbeit ist gut gelungenen und das Ziel, alle Früchte zu erkennen wurde erreicht. Zusätzlich konnten alle Früchte, ausser die Zitrone, in einem erfolgreichen Versuch festgehalten werden. Es ist klar, dass die Genauigkeit des Machine-Learning-Modells, mit zusätzlichen Daten zum Trainieren, gesteigert werden könnte. Besonders die Erkennung der Zitrone könnte noch verbessert werden mit zusätzlichen Daten.

Allerdings reicht die Genauigkeit dieses Machine-Learning-Modells für den Zweck dieser Arbeit völlig aus.

5.4.3 Quintessenz der Arbeit

Das Projekt hat sehr viel Freude bereitet. Die Erfahrungen, welche mit diesem Projekt gemacht wurden, sind äusserst umfangreich. Durch die Arbeit ist es gelungen ein fundiertes Wissen in den Themengebieten der Robotik, des maschinellen Lernens und der Informatik zu erlangen. Aufgrund der fehlenden Erfahrung in diesen Gebieten vor der Arbeit, war es schwer den Umfang dieses Projektes einzuschätzen. Trotzdem ist es gelungen, eine erfolgreiche technische Produktion zu kreieren.

6.0 Zusammenfassung

Ziel dieser Maturaarbeit war die Entwicklung eines Raspberry Pi basierten Roboters, der in der Lage ist, fünf verschiedene Früchte mithilfe von maschinellem Lernen zu erkennen und anzusteuern. Sobald die Frucht angesteuert wird, erfolgt die Ausgabe des Fruchtnamens mittels Text-to-Speech. Beim Entfernen der Frucht, soll der Roboter wiederum, mittels Text-to-Speech, den Befehl geben, die Frucht an ihren zugehörigen Ort zu bringen.

Die Basis dieser Arbeit bildet das Erstellen eines eigenen Datensatzes, um ein Machine-Learning-Modell zu trainieren. Dieser Datensatz besteht ausschliesslich aus selbst aufgenommen Fotos der Früchte. Die Bilder wurden unter verschiedenen Lichtverhältnissen, vor verschiedenen Hintergründen und mit unterschiedlichen Kameras aufgenommen, um das Modell möglichst robust zu gestalten. Zudem wurden auch negative Objekte einbezogen, um die Zuverlässigkeit weiter zu verbessern.

Durch das Training des Modells mithilfe des Datensatzes konnte schlussendlich ein mAP-Wert von 88.8% erzielt werden.

Mit einem funktionierenden Modell allein kann aber noch keine Frucht angesteuert werden. Daher wurde das Roboterset PiCar-X als Grundlage gewählt und mit dem Raspberry Pi 4b ergänzt. Der Raspberry Pi fungiert als Gehirn des Roboters.

Mithilfe des Programms Visual Studio Code wurde in Python ein Programmcode entwickelt, der es dem Roboter ermöglicht, die Früchte in einem definierten Bereich zu finden und sich ihnen anzunähern. Sobald die Frucht gefunden und angesteuert wird, kann mithilfe von Text-to-Speech der Name der Frucht ausgegeben werden.

Vor die grössten Herausforderungen stellte uns die Raspberry Pi V2 Kamera. Diese erzielte ohne die Erweiterung des Coral-USB-Accelerators nur moderate Ergebnisse. Erst durch die erwähnte Erweiterung ist gelungen, gute Resultate zu erzielen.

7.0 Quellenverzeichnisse

7.1 Literaturverzeichnis

ARTI - Autonomous Robot Technology GmbH. (15. 01 2021). *Eine kurze Geschichte der Robotik – Anfänge bis zur Gegenwart*. Abgerufen am 3. 10 2023 von https://morethandigital.info/eine-kurze-geschichte-der-robotik-anfaenge-bis-zur-gegenwart/

Coral. (2020). *Coral*. Abgerufen am 01. 09 2023 von https://coral.ai/docs/accelerator/get-started#requirements

Dalwigk, F. (Regisseur). (2021). *Was ist Objektorientierte Programmierung?* [Kinofilm]. Abgerufen am 03. 10 2023 von https://www.youtube.com/watch?v=c6RrcEvIix0

Damien, L. (03. 05 2020). *Where does the word 'robot' come from?* Abgerufen am 3. 10 2023 von https://www.sciencefocus.com/future-technology/where-does-the-word-robot-come-from

DataScientist. (17. 05 2023). *Convolutional Neural Network (CNN): Alles, was Du wissen solltest*. Abgerufen am 03. 09 2023 von https://datascientest.com/de/convolutional-neural-network-2

EdjeElectronics. (kein Datum). *EdjeElectronics Youtube*. Abgerufen am 10. 10 2023 von https://www.youtube.com/@EdjeElectronics

EducaTec AG. (03. 10 2023). *DC Gearbox Motor - "TT Motor" - 200RPM - 3 to 6VDC*. Abgerufen am 03. 10 2023 von https://educatec.ch/tectools/bauteile/3185/dc-gearbox-motor-tt-motor-200rpm-3-to-6vdc

Gaba, I. (03. 02 2023). *simplilearn*. Abgerufen am 07. 10 2023 von https://www.simplilearn.com/tutorials/git-tutorial/what-is-github

Honerlage, L. (19. 03 2021). *Ultraschall Abstandssensor HC-SR04*. Abgerufen am 03. 10 2023 von https://wiki.hshl.de/wiki/index.php/Ultraschall_Abstandssensor_HC-SR04#:~:text=Auf%20dem%20HC%2DSR04%20befinden,in%20ein%20elektrisches%20Signal%20um

IBM. (kein Datum). *Was ist künstliche Intelligenz?* Abgerufen am 03. 10 2023 von https://www.ibm.com/de-de

Ionos. (23. 06 2022). *Ionos.de*. Abgerufen am 04. 10 2023 von https://www.ionos.de/digitalguide/server/tools/secure-shell-ssh/

Jared Wilber, B. W. (kein Datum). *The Importance of Data Splitting*. Abgerufen am 03. 10 2023 von https://mlu-explain.github.io/train-test-validation/

Litzel, N. (25. 02 2019). *Was ist ein Convolutional Neural Network?* Abgerufen am 04. 10 2023 von https://www.bigdata-insider.de/was-ist-ein-convolutional-neural-network-a-801246/

Loshin, P. (02 2021). *Secure Shell (SSH)*. (ComputerWeekly.de, Produzent) Abgerufen am 04. 10 2023 von https://www.computerweekly.com/de/definition/Secure-Shell-SSH

Novustat. (03. 08 2020). *Künstliches neuronales Netz einfach erklärt: Lernen im Data Mining*. Abgerufen am 03. 10 2023 von https://novustat.com/statistik-blog/kuenstliches-neuronales-netz-einfach-erklaert.html

OSZ-KFZ (Regisseur). (2021). *Grundlage Zahnradgetriebe* [Kinofilm]. Abgerufen am 03. 10 2023 von https://youtu.be/zrPj9tS-bH4?si=-XROkES8e8l_34dK

Raspberry Pi OS. (kein Datum). *Raspberry Pi OS*. Abgerufen am 03. 10 2023 von https://www.raspberrypi.com/software/

Raspberry Pi. (kein Datum). *Raspberry Pi Camera Module 2*. Abgerufen am 03. 10 2023 von https://www.raspberrypi.com/products/camera-module-v2/

Raspberry Pi. (kein Datum). *Raspberry Pi Camera Module 2*. Abgerufen am 03. 10 2023 von https://www.raspberrypi.com/products/camera-module-v2/

Shah, D. (19. 01 2023). *v7labs*. Abgerufen am 02. 10 2023 von https://www.v7labs.com/blog/coco-dataset-guide

simpleclub, p. (Regisseur). (2015). *Wie funktioniert ein Gleichstrommotor?* [Kinofilm]. YouTube. Abgerufen am 03. 10 2023 von https://www.youtube.com/watch?v=6kJuuPxo4Q0

studyflix. (25. 01 2022). *Neuronale Netze*. Abgerufen am 03. 10 2023 von https://studyflix.de/informatik/neuronale-netze-4297

SunFounder. (kein Datum). *Grayscale Module*. Abgerufen am 03. 10 2023 von https://docs.sunfounder.com/projects/pico-4wd-v2/en/latest/module/grayscale_intro.html

SunFounder. (kein Datum). *Hardware Introduction*. Abgerufen am 03. 10 2023 von https://docs.sunfounder.com/projects/robot-hat-v4/en/latest/hardware_introduction.html

SunFounder. (kein Datum). *PiCar-X*. Abgerufen am 03. 10 2023 von https://docs.sunfounder.com/projects/picar-x/en/latest/

SunFounder. (kein Datum). *Robot HAT*. Abgerufen am 04. 10 2023 von https://docs.sunfounder.com/projects/picar-x/en/latest/about_robot_hat.html

SunFounder. (kein Datum). *SunFounder About us*. Abgerufen am 10. 10 2023 von https://www.sunfounder.com/pages/about-us

Venkatesan, G. (05. 01 2019). *freeCodeCamp*. Abgerufen am 07. 10 2023 von https://www.freecodecamp.org/news/learn-the-basics-of-git-in-under-10-minutes-da548267cc91/

w3schools. (kein Datum). *w3schools - python*. Abgerufen am 06. 10 2023 von https://www.w3schools.com/python/default.asp

Wikipedia. (14. 10 2022). *Analog-Digital-Umsetzer*. Abgerufen am 04. 10 2023 von https://de.wikipedia.org/wiki/Analog-Digital-Umsetzer

Wikipedia. (24. 10 2022). *Objektorientierte Programmierung*. Abgerufen am 06. 10 2023 von https://de.wikipedia.org/wiki/Objektorientierte_Programmierung

Wikipedia. (25. 10 2022). *Pulsdauermodulation*. Abgerufen am 4. 10 2023 von https://de.wikipedia.org/wiki/Pulsdauermodulation

Wikipedia. (13. 05 2023). *GPIO*. Abgerufen am 03. 10 2023 von https://de.wikipedia.org/wiki/GPIO

Wikipedia. (13. 5 2023). *GPIO*. Abgerufen am 04. 10 2023 von https://de.wikipedia.org/wiki/GPIO

Wikipedia. (10. 10 2023). *Raspberry Pi*. Abgerufen am 03. 10 2023 von https://de.wikipedia.org/wiki/Raspberry_Pi

Wikipedia. (27. 07 2023). *Robotik*. Abgerufen am 03. 10 2023 von https://de.wikipedia.org/wiki/Robotik

Wikipedia. (27. 07 2023). *Robotik*. Abgerufen am 04. 10 2023 von https://de.wikipedia.org/wiki/Robotik

Wuttke, L. (kein Datum). Abgerufen am 03. 10 2023 von Machine Learning: Definition: https://datasolut.com/was-ist-machine-learning/

Wuttke, L. (kein Datum). *Machine Learning: Definition, Algorithmen, Methoden und Beispiele*. Abgerufen am 03. 10 2023 von https://datasolut.com/was-ist-machine-learning/#Wie-funktioniert-Machine-Learning

Wuttke, L. (kein Datum). *Machine Learning: Definition, Algorithmen, Methoden und Beispiele*. Abgerufen am 03. 10 2023 von https://datasolut.com/was-ist-machine-learning/

Wuttke, L. (kein Datum). *Training-, Validierung- und Testdatensatz*. Abgerufen am 03. 10 2023 von https://datasolut.com/wiki/trainingsdaten-und-testdaten-machine-learning/

7.2 Abbildungsverzeichnis

7.3 Tabellenverzeichnis

Anhang

Antiplagiatserklärung

Umgang mit Plagiaten

Was gilt als Plagiat?

Unter einem Plagiat ist die ganze oder teilweise Übernahme eines fremden Werks ohne Angabe der Quelle und des Urhebers bzw. der Urheberin zu verstehen. Das Plagiat ist eine Verletzung des Urheberrechts. Kürzere Passagen eines fremden Werkes dürfen zitiert werden. Dies setzt aber eine Kennzeichnung des Zitats und eine Angabe der Quelle voraus. Folgende Handlungen stellen ein Plagiat im weiteren Sinne dar:

a) Die Verfasserin bzw. der Verfasser reicht ein Werk, das von einer anderen Person auf Auftrag erstellt wurde, unter ihrem bzw. seinem Namen ein.

b) Die Verfasserin bzw. der Verfasser reicht ein fremdes Werk unter ihrem bzw. seinem Namen ein.

c) Die Verfasserin bzw. der Verfasser übernimmt Textteile oder Aussagen von fremden Quellen, ohne die Herkunft mit einem Beleg kenntlich zu machen. Dazu gehört namentlich auch das Verwenden von Textteilen aus dem Internet ohne Quellenangabe und das Übersetzen fremdsprachiger Texte oder Teile von fremdsprachigen Texten ohne Quellenangabe.

d) Die Verfasserin bzw. der Verfasser übernimmt Textteile oder Aussagen von fremden Quellen und nimmt leichte Textanpassungen und -umstellungen vor (Paraphrasieren), ohne die Herkunft mit einem Beleg kenntlich zu machen.

Unredlichkeiten und ihre Folgen

Beim Aufdecken von Unredlichkeiten wird je nach Ausmass unterschiedlich vorgegangen:

- Wenige vergessene Zitierangaben, fehlerhaftes Anwenden der Grundregeln des Zitierens und Ähnliches wird mit entsprechenden Punktabzügen bei der Bewertung der Arbeit sanktioniert.
- Die im Sinne von c) und d) als Plagiat identifizierten Textteile oder Aussagen werden - mit der entsprechenden Konsequenz für die Bewertung der Arbeit - nicht korrigiert. Zudem können disziplinarische Massnahmen ergriffen werden.
- Übersteigen die im Sinne von c) und d) identifizierten Textteile oder Aussagen 50% des Umfangs einer Arbeit, wird die Arbeit mit der Note 1 bewertet und hat zwingend eine disziplinarische Massnahme zur Folge. Bei den Abschlussarbeiten entfällt die mündliche Präsentation.
- Ein Plagiat im Sinne von a) und b) wird mit der Note 1 bewertet und hat zwingend eine disziplinarische Massnahme zur Folge. Bei den Abschlussarbeiten entfällt die mündliche Präsentation.

Antiplagiatserklärung

Ich erkläre hiermit, dass diese Arbeit nicht abgeschrieben, kopiert, übersetzt oder über das Internet heruntergeladen wurde, die Inhalte dieser Arbeit, auch nicht in Teilen, aus anderen Quellen übernommen worden sind, ohne dass diese korrekt ausgewiesen wurden, der Quellennachweis korrekt angebracht und vollständig aufgeführt ist und die dargestellten Daten und Resultate selber erhoben und korrekt wiedergegeben wurden.

Datum:

Name: Leon Schaad Abt.: G2020D Unterschrift: L. Schaad

Name: Adriano Carlucci Abt.: G2020D Unterschrift: A. Carlucci

Name: _____ Abt.: _____ Unterschrift: _____

Name: _____ Abt.: _____ Unterschrift: _____

Sammlung der wichtigsten Quellen

Colab Notebook

Matura23_Train_TFLite2_Object_Detction_Model.ipynb - Colaboratory (google.com)

Programmcode

https://github.com/Adrigit04/matura23-mypicar-x

Videos FrubotAI

https://www.youtube.com/@Leon____le/videos

Datensatz zum Trainieren des Modells

https://github.com/Adrigit04/matura23-ml-training