INSTITUTE FOR NETWORKED SOLUTIONS

OST Ostschweizer Fachhochschule

JetBot

Autonomes und fortlaufendes maschinelles Lernen

Studienarbeit

Studiengang Informatik

OST – Ostschweizer Fachhochschule

Campus Rapperswil-Jona

Herbstsemester 2020

Autor(en):	Aaron Studer, Benjamin Peter, Yanick Rek
Betreuer:	Prof. Dr.-Ing. Andreas Rinkel, Marc Sommerhalder
Projektpartner:	INS, Institute for Networked Solutions

Abbildungsverzeichnis

Tabellenverzeichnis

Inhaltsverzeichnis

1. Abstract

Das Ziel ist es, informatikinteressierten Personen einen Einblick in die Informatik, mit Fokus auf die Künstliche Intelligenz, zu ermöglichen. Der Roboter soll sich, nach der Umsetzung des Folgeprojekts, autonom durch zusammenhängende Räume bewegen können und nach einer Person oder einem Objekt suchen.

Die Arbeit ist in die folgenden drei Teilgebiete unterteilt: Hardware Evaluation, Objekt Detektion und Pfadfindung.

Für die Evaluation der Hardware wird der zur Verfügung gestellte JetBot Roboter von SparkFun als Referenz verwendet. Das Hauptproblem mit diesem JetBot ist die unzureichende Stromversorgung was dazu führt, dass der Jetson Nano in einem Power-Saving-Mode betrieben werden muss. Ein selbst zusammengestellter JetBot entspricht am ehesten den Anforderungen gemäss Nvidia. Abgesehen von der Stromversorgung kann auch die WLAN-Konnektivität verbessert werden und die GPIO-Pins sind so noch unbelegt.

Bei der Object-Detektion auf dem Jetson Nano muss stark auf die Perfomance geachtet werden. Drei Algorithmen (Faster R-CNN, YOLO v3 und SSD) werden miteinander verglichen. SSD Mobilenet-V2 konnte bei der Evaluation für diese Applikation überzeugen.

Die Pfadfindung wird algorithmisch, mithilfe von zwei Ultraschallsensoren und den Wänden der Räume, gelöst. Falls ein Hindernis beim Abfahren der Wände auftritt, wird es mithilfe der Ultraschallsensoren umfahren. Falls das Hindernis beim direkten Anfahren des Ziels auftritt, wird ein schlankes Kollisionsdetektions-Modell verwendet.

2. Management Summary

2.1 Ausgangslage

In der hier vorliegenden Arbeit wird beschrieben, wie man einen Roboter dazu einsetzen kann, informatikinteressierten Personen einen Einblick in die Informatik, mit Fokus auf die Künstliche Intelligenz, zu ermöglichen. Die eigentliche Umsetzung des hier Beschriebenen, wird in einer Folgearbeit stattfinden.

Um zu Beginn der Arbeit bereits ein Gefühl für die Hardware und deren Möglichkeiten entwickeln zu können, wurde ein Roboter von SparkFun zur Verfügung gestellt. Das Herzstück des Roboters ist ein Jetson Nano, welcher für Machine Learning ausgelegt ist.

Die Arbeit umfasst die Definition der Folgearbeit, inklusiv Ausbaustufen für allfällige weiterführende Arbeiten, sowie die Evaluation der Hardware, des Object-Detection-Algorithmus und die Ausarbeitung eines Algorithmus, der es einem Roboter ermöglicht einen geschlossenen Raum vollständig abzufahren. Durch diese Technologien kann eine Art Rescue-Roboter entwickelt werden, der nach einer Person oder einem Objekt in mehreren zusammenhängenden Räumen selbstständig und autonom sucht.

2.2 Vorgehen, Technologien

In den ersten Wochen wurde der Roboter getestet und versucht mehrere kleine experimentelle Teile zu implementieren, um ein Gefühl für die Hardware zu entwickeln.

Zudem wurden Ideen gesammelt, was man in der Folgearbeit alles implementieren kann. Diese Ideen wurden dann mit Marc Sommerhalder und Prof. Dr.-Ing. Andreas Rinkel aussortiert und priorisiert, so dass man ein klares Bild erhielt, was der Roboter schlussendlich tun wird und in welcher Reihenfolge die Teilschritte implementiert werden.

Da bereits in den ersten Wochen mehrere Probleme mit der Hardware auftraten, wurde die Evaluation der Hardware ebenfalls zu einem wichtigen Teil der Arbeit. Dazu wurde der bereits vorliegende Roboter mit mehreren ähnlichen Robotern, die alle den Jetson Nano als Recheneinheit besitzen, verglichen. Das Hauptproblem des Roboters, den wir zu Beginn erhielten, ist dass die Stromversorgung für den Jetson Nano nicht ausreicht und der Jetson Nano darum in einem Power-Saving-Mode betrieben werden muss. Ein weiteres Problem ist, dass das WLAN, welches mittels eines USB-Dongles zur Verfügung gestellt wird, nicht zuverlässig funktioniert. Ausserdem sind die GPIO-Pins bereits durch ein Extension-Board belegt, das für den Zweck dieser Arbeit viel zu gross ist. Durch dieses Board können weitere Komponenten, wie zum Beispiel Ultraschall-Sensoren nur mittels eines Workarounds angeschlossen werden. Da dies den Umfang der Arbeit einschränkt, werden bessere Lösungen dafür gefunden.

Für die Objekterkennung werden die drei bekanntesten Detektionsalgorithmen evaluiert, um dadurch eine optimale Lösung für die Echtzeit Objekterkennung zu finden.

Damit sich der Roboter selbstständig durch die Räumlichkeiten bewegen kann, wird ein Algorithmus zur Pfadfindung evaluiert und implementiert. Falls Hindernisse beim Erkunden auftreten, muss der Roboter diese erkennen und ausweichen können. Da die Objekt-Detektion den Roboter bereits stark auslastet, wird eine leistungsschonende Lösung gefunden.

2.3 Ergebnisse

Das Ziel der Folgearbeit ist es den Roboter in einem Raum platzieren zu können und er autonom ein bestimmtes Objekt in diesem Raum finden kann.

Als erste Erweiterungsstufe soll es möglich sein das Kollisions-Erkennungs-Modell mittels Userinput oder mittels Ultraschallsensor fortlaufend zu verbessern.
In der zweiten Erweiterungsstufe soll der Roboter per Sprachsteuerung kontrollierbar sein. Zum Beispiel soll es möglich sein, dem Roboter mündlich zu sagen, nach welchem Objekt er suchen soll. Eine weitere Erweiterung ist die Interkommunikation zwischen mehreren Robotern, um zum Beispiel den Raum schneller abzufahren oder die Detektions-Modelle miteinander abgleichen zu können.

Statt eines fertigen Roboters wird ein selbst zusammengestellter JetBot verwendet. Dieser wurde anhand der „Bill of Materials“[1] von NVIDIA zusammengestellt.
Die Stromversorgung erreicht noch immer nicht die empfohlenen 4 Ampère. Jedoch werden 3 Ampère erreicht, was ausreichen wird, um den Jetson Nano auf voller Leistung laufen zu lassen, da wenig zusätzliche Komponenten über den Jetson Nano gespiesen werden. Um die Zuverlässigkeit des WLAN sicherzustellen, wird statt eines USB-Dongles eine WLAN-Karte, zusammen mit zwei Antennen verwendet. Die Kamera wird ausgetauscht mit einer, die einen etwas grösseren Weitwinkel besitzt.
Das Gehäuse kann 3D-gedruckt werden. Dies ist kostengünstig und ermöglicht Anpassungen am Gehäuse, um die nötigen Sensoren korrekt befestigen und ausrichten zu können.

Für die Objekt-Detektion wird SSD Mobilenet-V2 verwendet, welches sich für die Echtzeit-Objekterkennung eignet. Zudem ermöglicht das Modell eine Geschwindigkeitsoptimierung sowie eine Verringerung des Batterieverbrauchs im Vergleich zu anderen Modellen.

Die Pfadfindung wird algorithmisch mithilfe von zwei Ultraschallsensoren gelöst. Der Roboter wird sich anhand der Wände orientieren und diesen folgen, bis er die gesamten Räumlichkeiten erfasst hat. Falls ein Hindernis während dem Abfahren der Wände erkannt wird, wird das Ausweichmanöver mithilfe der Ultraschallsensoren durchgeführt. Falls der Roboter sich bereits auf dem direkten Weg zum Ziel befindet, wird ein schlankes Kollisionsdetektions-Modell verwendet.

[1] https://jetbot.org/master/bill_of_materials.html

2.4 Ausblick

Die hier vorliegenden theoretischen Analysen und Implementationen können in einer folgenden Bachelorarbeit umgesetzt werden. Die Bestellung der Hardware wurde bereits in Auftrag gegeben. Einzelne Teile müssen noch 3D-gedruckt werden und dann kann der Roboter zusammengebaut werden und die Entwicklung beginnen.

3. Einleitung

In der hier vorliegenden Arbeit werden anhand eines programmierbaren Roboters die Möglichkeiten moderner Einsatzfelder der Informatik, mit Fokus auf die Künstliche Intelligenz, für angehende Studierende aufgezeigt. Ausgehend von einer Analyse der verschiedenen verfügbaren Technologien wie handelsüblichen Robotersystemen und verfügbaren Software-Bibliotheken wird anschliessend eine Anforderungsspezifikation für ein nachfolgendes Projekt, eine mögliche Bachelorarbeit, erarbeitet. Der Fokus liegt hierbei auf Komponenten, die preiswert sind und auf Technologien, die auch ohne «High-End» Grafikkarten und enormer RAM Kapazität auskommen.

Nach dieser und der anschliessenden zukünftigen Arbeit ist es möglich zu demonstrieren, wie sich ein Roboter ohne menschliche Interaktion autonom bewegen und Probleme lösen kann. Diese Visualisierung von Problemen im Bereich der künstlichen Intelligenz wird es Informatik interessierte Personenermöglichen, sich mehr mit dem Thema vertraut machen.

Inhalte der Analyse sind:

- Anforderungsspezifikation der benötigten Hardware und Roboter Selektion
- Evaluation und Studium von drei verschiedenen Object Detection Algorithmen (Faster R-CNN, YOLO v3 und SSD)
- Evaluation und Visualisierung der algorithmischen Pfadfindung
- Evaluation der Implementation für die Hindernis-Erkennung

Eine Demonstration, wie sich der Roboter nach der Umsetzung bewegen könnte und wie die Object Detection Algorithmen implementiert sind, kann man über die eigenentwickelte Web-App[2] testen.

[2] https://algo-vis-robot.azurewebsites.net/

4. Das weiterführende Projekt

In diesem Kapitel werden Ideen für die anschliessende Bachelorarbeit aufgezeigt. Die Implementation wird in drei verschiedene Ausbaustufen aufgeteilt und dazu wird jeweilig erklärt, warum die darin enthaltenen Ziele eine passende Ergänzung für das hier vorliegende Projekt sind.

Die Entwicklung des gesamten Projekts ist nicht Teil dieser Arbeit. In dieser Vorarbeit wird evaluiert, was alles möglich ist und welche Hard- und Software-Komponenten sich für das weiterführende Projekt eignen.

4.1 Übersicht Funktionalität und Ausbaustufen

Die folgende Tabelle stellt die drei Ausbaustufen der zukünftigen Arbeit und die dazugehörige Funktionalität als Übersicht dar. Die zukünftige Implementation soll in der gleichen, nachfolgenden Reihenfolge geschehen.

Tabelle 1: Übersicht Funktionalität und Ausbaustufen

<table>
<tr><th>Name der Stufe:</th><th>Beschreibung der Funktionalität:</th></tr>
<tr><td>Stufe 1: Grund-funktionalität</td><td>
<table>
<tr><th>Ziel</th><th>Notwendige Komponenten</th></tr>
<tr><td>Priorität 1:
Erkennen eines Objekts oder einer Person</td><td>• WLAN-Verbindung
• Kamera
• Trainiertes Modell zur Erkennung von Objekten
• Anzeigemöglichkeit</td></tr>
<tr><td>Priorität 2:
Anfahren des Ziels</td><td>• Motoren
• Eingabemöglichkeit, damit Benutzer zwischen verschiedenen Objekten auswählen kann</td></tr>
<tr><td colspan="2">Priorität 3:
Abfahren von zusammenhängenden Räumen
<table>
<tr><th>Name des Arbeitspakets</th><th>Notwendige Komponenten</th></tr>
<tr><td>Arbeitspaket 1: Räume abfahren</td><td>• Ultraschallsensor seitlich und vorne</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>Arbeitspaket 2: Ziel anfahren</td><td>• Modell zur Kollisionserkennung</td></tr>
</table>
</td></tr>
<tr><td>Stufe 2: Fortlaufendes Lernen</td><td>
<table>
<tr><td>Ziel</td><td>Notwendige Komponenten</td></tr>
<tr><td>Priorität 4: Fortlaufendes Lernen durch neue Situationen und Userinput</td><td>• Modell zur Kollisionserkennung erweitern
<table>
<tr><td>Name des Arbeitspakets</td><td>Notwendige Komponenten</td></tr>
<tr><td>Arbeitspaket 1: Controller</td><td>• Userinput, zum Beispiel durch Controller</td></tr>
<tr><td>Arbeitspaket 2: Ultraschallsensoren</td><td>• Ultraschallsensoren zum autonomen fortlaufenden Lernen</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Stufe 3: Ideen für zukünftige Projekte</td><td>
<table>
<tr><td>Ziel</td><td>Notwendige Komponenten</td></tr>
<tr><td>Priorität 5: Sprachsteuerung</td><td>• Mikrofon</td></tr>
<tr><td>Priorität 6: Interkommunikation zwischen zwei bis n Robotern</td><td>• Kommunikationsprotokoll
• Aufgabenteilung
• Modellerweiterung</td></tr>
</table>
</td></tr>
</table>

4.2 Stufe 1: Grundfunktionalität

Die Stufe 1 bringt grundlegende Machine Learning Elemente in das Projekt, wie die Objekt- und Personen-Erkennung sowie ein Modell zur Detektion von Kollisionen. Um diese bereits gut recherchierten wissenschaftlichen Themen interessant darzustellen, ist das Ziel der ersten Stufe, dass der Roboter sich selber autonom durch zusammenhängende Räume bewegen kann. Er muss beim Erkunden dieser Räume nach einem Objekt suchen, das zu Beginn definiert wurde. Der Roboter wird beim Erkunden der Räumlichkeiten von zwei Ultraschallsensoren unterstützt.

Ein konstruiertes Raum-Modell aus Karton ermöglicht es, die Grundfunktionalität in einer sicheren Umgebung darzustellen und zu präsentieren. Zusätzlich ist es möglich, in dieser Umgebung verschiedene Testfälle und Szenarien zu testen.

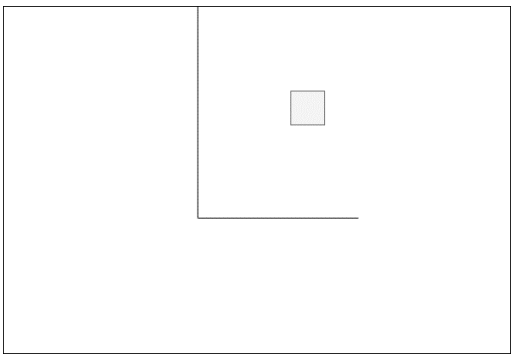


Abbildung 1: Grundfunktionalität, Raum-Modell

Anhand dieser ersten Ausbaustufen kann gezeigt werden, dass eine eher kostengünstige Plattform genug Performance hat, verschiedene Arbeiten gleichzeitig zu erledigen. Zusätzlich stellt es eine gute Kombination zwischen Machine Learning (Objekt / Personen- und Kollisionserkennung) und algorithmischen Wegfindungsproblemen dar.

4.2.1 Erkennen eines Objekts oder Person

Das Teilproblem «Erkennen eines Objekts» ist ein bekanntes Machine Learning Problem.
Dadurch, dass die Objekterkennung nicht auf einem vollwertigen Computer, sondern auf dem Roboter ausgeführt wird, muss stark auf den Speicherverbrauch und andere Hardware Limitierungen geachtet werden, was zu möglichen Optimierungsproblemen führen könnte.

4.2.2 Direktes Anfahren eines Ziels falls erkannt

Das Ziel ist, dass der Roboter das Objekt bereits im Blick hat und er dieses direkt anfahren kann, ohne ausweichen zu müssen. Zudem muss es möglich sein, die Geschwindigkeit des Roboters einfach anpassen zu können. Ein mögliches Problem könnte sein, dass der Roboter mit dem Objekt kollidiert und nicht davor anhält. Das wäre jedoch für die Ausbaustufe 1.2 noch absolut in Ordnung. Damit der Benutzer zwischen mehreren möglichen Objekten auswählen kann, erhält der Endbenutzer in Jupyter Notebook eine Dropdown-Liste mit den möglichen Objekten.

4.2.3 Algorithmus, um das Abfahren komplexer Räume zu ermöglichen

Die Priorität 1.3 wird in zwei Arbeitspakete unterteilt, die dann zu einem guten Suchalgorithmus zusammengesetzt werden.

Räume abfahren, so dass jeder Ort einmal von der Kamera erfasst wird

Der Roboter wird sich anhand der Wände orientieren und sich so durch alle zusammenhängenden Räume bewegen. Durch die Weitwinkel-Kamera wird dadurch jeder Ort einmal erfasst. Dieses Problem wird algorithmisch Mithilfe von zwei Ultraschallsensoren gelöst.

Zielobjekt anfahren und dabei, wenn nötig, einem Hindernis ausweichen

Direkt auf ein Objekt zu zufahren ist bis zu dieser Priorität bereits implementiert. Nun muss der Roboter auch lernen, einem Hindernis wie zum Beispiel einer Säule auszuweichen.
Die Kollisionsdetektion wird hier nicht nur mittels Ultraschallsensoren, sondern auch mittels eines trainierten Kollisions-Detektions-Modells, welches die Kamera als Sensor verwendet, gewährleistet. Die Kamera muss verwendet werden, da die Ultraschallsensoren nicht alle Hindernisse erkennen können.

Sobald ein Hindernis erkennt wird, richtet sich der Roboter so aus, dass sich das Hindernis rechts von ihm befindet und vom Ultraschallsensor detektiert wird.
Dann fährt der Roboter wieder im Standard-Modus weiter. Das heisst, er fährt dem Hindernis nach, bis er das Zielobjekt sehen kann. Dadurch kann man sicherstellen, dass man das Hindernis in jedem Fall umfahren kann. So kann gewährleistet werden, dass das Objekt sicher wiedergefunden wird, obwohl man es kurzzeitig aufgrund des veränderten Winkels nicht mehr detektieren kann.

4.3 Stufe 2: Fortlaufendes Lernen

Die Stufe zwei wird die Komplexität des Roboters noch einmal deutlich erhöhen. Ab dieser Stufe wird nicht nur ein vortrainiertes Modell genutzt, sondern wenn nötig wird dieses auch erweitert, damit seine Bibliothek zur Problemlösung immer grösser und genauer wird.
Die Erweiterung eines bestehenden Modells ist zurzeit technisch nicht möglich (Catastrophic interference [1]), da man jedoch noch die ursprünglichen Daten zur Erzeugung des Modells zur Verfügung hat, kann man mit deren Hilfe das Modell neu berechnen und das alte Modell zu einem geeigneten Zeitpunkt mit dem Neuen ersetzen.

4.3.1 Bestätigung von möglichen Kollisionen

Diese Priorität wird in zwei Arbeitspakete unterteilt, die zwar das gleiche Ziel verfolgen, dies jedoch mit komplett unterschiedlichen Mitteln erreichen wollen.

Bestätigung von möglichen Kollisionen durch Controller

Das Ziel des ersten Arbeitspaket wird sein, dass ein vortrainiertes Kollisionsdetektion-Modell während dem Einsatz verbessert werden kann.
Konkret bedeutet das, dass wenn der Roboter sich nicht sicher ist, ob er demnächst in ein Hindernis fährt, er den Benutzer fragen kann, wie seine Einschätzung ist. Dieser kann per Knopfdruck entscheiden, was

passiert. Nach dem Entscheid wird der Roboter ein Foto aufnehmen und dieses wird dann in das bereits vortrainierte Modell aufgenommen.

Bestätigung von möglichen Kollisionen durch Ultraschall-Sensor

Bei diesem zweiten Arbeitspaket wird das gleiche Ziel wie zuvor verfolgt, nur ist hier keine User Interaktion notwendig. Nach dem Detektieren wird wieder ein Foto aufgenommen und so das Modell verbessert. Die Implementation wird sich sicherlich ähnlich gestalten wie innerhalb der vorherigen Priorität, nur muss hier das Zusammenspiel mit dem Ultraschallsensor sichergestellt werden.

4.4 Stufe 3: Ideen für zukünftige Projekte

In der dritten und letzten Stufe werden Implementationsideen für zukünftige Arbeiten gesammelt. Das Spektrum der Künstlichen Intelligenz sowie deren Möglichkeiten sind sehr gross. Daher wäre es interessant zu sehen, wie man eine Sprachsteuerung für den Roboter implementieren könnte oder wie mehrere Roboter (zwei bis n) miteinander kommunizieren könnten, um sich gegenseitig Arbeit abzunehmen. Des Weiteren wäre ein Austausch von unterschiedlich trainierten Modellen optimal, da die Roboter so bedeutend schneller viel mehr neue Situationen lernen können.

5. Evaluation

5.1 Roboter

In diesem Abschnitt wird evaluiert, welches Roboter-Kit sich am besten für die Arbeit eignet. Als Orientierung werden die von NVIDIA vorgeschlagenen Roboter-Kits [2] verwendet.
Zu Beginn der Arbeit wurde ein Kit von SparkFun als Grundlage zur Verfügung gestellt. Dies führte jedoch früh zu vielen verschiedenen Problemen, bis der Roboter letztendlich aus unbekannten Gründen nicht mehr startete. In diesem Unterkapitel werden nun die positiven und negativen Aspekte des Kits beschrieben und dann versucht, einen besseren Roboter anhand der Spezifikationen zu finden. Zusätzlich muss noch erwähnt werden, dass die Evaluation anhand der Spezifikation und Produktbewertung nicht zwingend zu einem perfekten Ergebnis führen. Dennoch wurde versucht, ein möglichst guter Vergleich zwischen den Kits zu erzielen, ohne die Kits bestellen und ausprobieren zu müssen, was sehr zeit- und kostenintensiv wäre.

5.1.1 SparkFun Roboter

In diesem Teilabschnitt wird das SparkFun-Kit genau untersucht. Die Einschätzung des SparkFun Roboters wird als Referenz gebraucht, um zu entscheiden, ob ein anderer Roboter besser oder schlechter für die weiterführende Arbeit geeignet ist.

Der SparkFun-JetBot besteht aus einem NVIDIA Jetson Nano Developer Kit, welches speziell für Künstliche Intelligenz und Robotik ausgelegt ist. Ebenso besitzt er zwei Motoren (und zwei Räder), eine Kamera, ein kleines Informationsdisplay, einen USB WLAN-Adapter sowie zwei optionale Ultraschallsensoren. Mittels SD-Karte wird das Betriebssystem Ubuntu 18.04.2 LTS eingesetzt und als Programmiersprache wird Python 3 in Jupyter Notebooks verwendet.

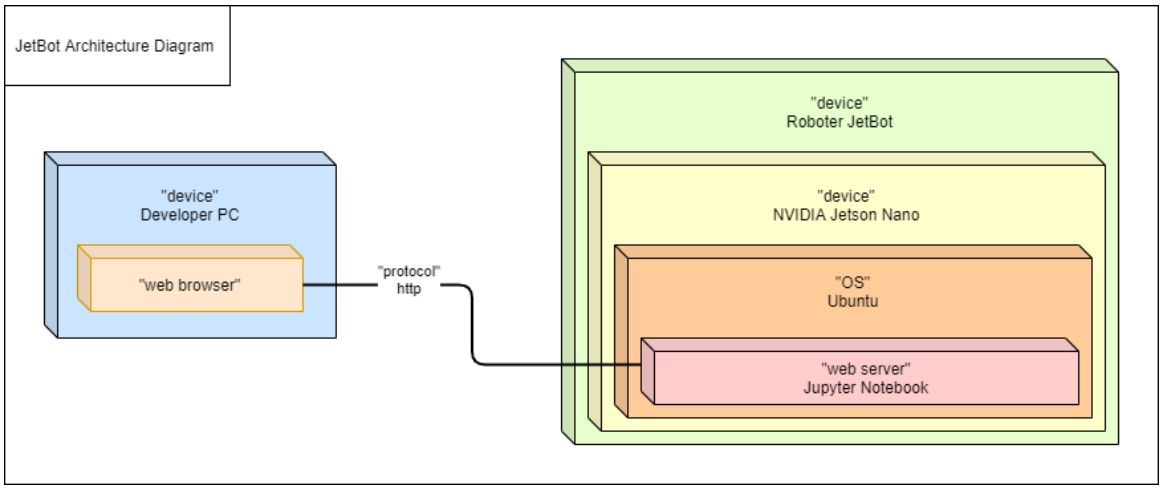


Abbildung 2: Komponenten, Architektur Diagramm

Jetson Nano

Das Kernstück des Roboter-Kits ist ein Jetson Nano Developer Kit. Zwei SparkFun-Kits liegen bereits zu Beginn des Projekts zu Testzwecken vor. Das eine ist ein Jetson Nano Developer Kit A02 (ältere Version) und das andere ist ein Jetson Nano Developer Kit B01 (neuere Version). In diesem Artikel [3] sieht man die Unterschiede zwischen den beiden Versionen.
Kurz zusammengefasst gibt es keine Veränderung in der Leistung der Boards, jedoch ist es in der neueren Version möglich, eine zweite Kamera anzuschliessen.

Peripherie

Der Jetson Nano muss hardwaremässig mit den Motoren, dem WLAN-Adapter, einem Display und der Kamera interagieren. Gespiesen wird er durch eine Powerbank mittels zwei Micro-USB-Kabeln. Die erste Speisung ist für das Board selbst, die zweite für die Motoren.

Qwiic-Board (I2C-Schnittstelle für Display und Motoren)

Um mit den Motoren und dem Display zu interagieren, wird ein von SparkFun zur Verfügung gestelltes Qwiic-Board benötigt. Dieses Qwiic-Board besetzt den gesamten GPIO-Header des Jetson Nano und stellt vier I2C-Anschlüsse zur Verfügung. Dadurch, dass I2C ein serielles Protokoll ist, ist es jedoch auch möglich, mehr als vier Geräte anzusteuern (theoretisch unendlich viele). Wenn man jedoch mehrere Geräte seriell miteinander verbindet, kann die Kommunikation mit einem bestimmten Gerät etwas länger dauern.
Der Nachteil des Qwiic-Boards ist, dass der gesamte GPIO-Header[3] belegt wird und man deshalb keine einzelnen Pins, beispielsweise für die Verwendung eines Ultraschallsensors, zusätzlich gebrauchen kann.

Eventuell wäre es möglich, Pins auf Kosten einer oder zwei I2C-Anschlüssen am Qwiic-Board umzukonfigurieren. Das kann jedoch nicht mit Sicherheit gesagt werden, ohne es getestet zu haben.
Wenn es möglich ist, können die Pins auf dem Qwiic-Board einfach abgefangen werden, um zum Beispiel die Pins eines Ultraschallsensors dort anzulöten.
Wenn es nicht möglich ist, muss ein I2C-fähiger Ultraschall-Sensor gefunden werden. Dies garantiert aber nicht, dass es mit dem Qwiic-Board funktioniert und müsste zusätzlich getestet werden.

Das Qwiic-Board nimmt dem Benutzer/innen mit dem richtigen SD-Karten-Image von SparkFun einen grossen Teil der Arbeit ab, da man nicht manuell einzelne Pins ansteuern muss. Es führt jedoch zu Komplikationen, wenn man zum Beispiel ein Image von NVIDIA benutzen möchte. Dann kann das Ansteuern der Peripherien zu einem sehr mühsamen und langwierigen Task werden.

Display

Das Display ist sehr hilfreich, da es einem die jeweilige IP-Adresse anzeigt, damit man sich mit dem Roboter per JupyterLab verbinden kann. Ohne Display müsste man den Roboter jedes Mal an einen Monitor

[3] Stiftleiste für Zugang zu Input-Output Pins

anschliessen, um herauszufinden, ob der Roboter überhaupt mit dem WLAN verbunden ist.
Das Display funktioniert ausgezeichnet und wird automatisch angesteuert, wenn man ein SD-Karten-Image von SparkFun verwendet.

WLAN-Adapter

Der WLAN-Adapter von Edimax ist im JetBot-Kit enthalten. Dennoch funktioniert er leider nicht wie gewünscht auf Anhieb. Mit folgendem Workaround [4] funktioniert der WLAN-Adapter zwar mit «normalen» WPA 2 Netzwerken, jedoch ist es dennoch unmöglich, ihn mit WPA 2 Enterprise Netzwerken zu verbinden, um eine stabile Verbindung aufrecht zu erhalten.
Damit der Workaround richtig funktioniert, muss ein veraltetes SD-Karten-Image von SparkFun verwendet werden, welches auf dieser Seite [5] unter «Troubleshooting» zu finden ist. Mit den neueren Versionen des Images funktioniert das WLAN nicht.

Um den Roboter auch an der Fachhochschule OST verwenden zu können, kann ein Hotspot via Smartphone genutzt werden.

Kamera

Die Kamera wird direkt an einem kompatiblen Header (J5) auf dem Jetson Nano angeschlossen und ruft keine Probleme hervor. Positiv an der verwendeten Kamera ist vor allem der weite Winkel von 145°, wodurch man mehr von der Umgebung im Blick hat, ohne sich drehen zu müssen [6].

Stromversorgung der Motoren

Ein weiteres kleines Board wird benötigt, um die Motoren über die Powerbank speisen zu können. Das ist gut, weil der Jetson Nano dadurch nur die Informationen zu diesem Board schicken muss und die gesamte Leistung vom Batteriepack kommt.
Ein Motor benötigt unter Belastung mit 5V einen Strom von ungefähr 1.3A. Da zwei Motoren gebraucht werden, wird mit einem maximalen Stromverbrauch von 2.6A gerechnet.
Um die volle Leistung der Motoren ausnützen zu können, wird ein 3A-Port benötigt, welcher bei dieser Powerbank jedoch bereits für den Jetson Nano gebraucht wird. Dadurch, dass man den Motoren weniger Strom zur Verfügung stellt, als sie eigentlich brauchen würden, reduziert man die Kraft, mit welcher sie sich drehen können. Ob die Belastung der Motoren eine Auswirkung auf die Powerbank oder im schlimmsten Fall auf den 3A-Port der Powerbank hat, kann nicht abschliessend beantwortet werden [7].

Powerbank

Die zur Verfügung gestellte Powerbank ist unpraktisch, weil man sie nicht gleichzeitig laden und verwenden kann. Ausserdem sind sich selbst die Entwickler bei SparkFun nicht sicher, ob sie der Powerbank vertrauen sollen, was eine grosse Unsicherheit darstellt.
So schreiben sie im Datenblatt: «Please be aware that the ability to run multiple neural networks in parallel

may only be possible with a full 5V 4A power supply.» [8].
Die oben genannte «5V 4A power supply» kann jedoch nicht über die Powerbank bezogen werden, sondern benötigt ein zusätzliches Netzteil. Die Powerbank liefert bei 5V maximal 3A. Da es für die fortführende Arbeit fast unausweichlich ist, mehrere Neuronale Netzwerke gleichzeitig laufen zu lassen, könnte die Powerbank tatsächlich der ausschlaggebende Punkt sein, auf den SparkFun JetBot zu verzichten.
Auf dem von SparkFun zur Verfügung gestellten SD-Karten-Image ist ein 5W-Mode, welcher immer aktiviert werden soll, wenn der Roboter an der Powerbank betrieben wird. Da der Roboter bei einer zukünftigen Arbeit durchgehend ausschliesslich von der Powerbank gespiesen wird, bedeutet dies eine starke Einschränkung der Funktionalität, beziehungsweise der Rechenleistung des Jetson Nano.

Es stellt sich jedoch die Frage, weshalb eine Stromversorgung von 5V und 4A empfohlen wird, wenn der Jetson Nano doch nur 10W benötigt, was also 5V und 2A entsprechen würde. In diesem Artikel [9] wird beschrieben, dass das Nano Modul allein 10W benötigt. Sobald man jedoch Peripherien anschliesst, wie zum Beispiel die Kamera, das Qwiic-Board, das Display, den WLAN-Adapter oder die Ultraschallsensoren, wird schnell sehr viel mehr Strom benötigt. Mit 4A funktioniert laut Online-Bewertungen alles problemlos. Ob der Jetson Nano auch mit 3A auf voller Leistung funktioniert, muss ausprobiert werden und benötigt viele Langzeittests und somit Zeit. Um jedoch eine genauere Einschätzung der Situation zu machen, werden hier die maximale Leistungsaufnahme der wichtigsten verwendeten Peripherien aufgelistet:

Tabelle 2: Übersicht Leistungsaufnahme verwendeter Peripherien

Peripherie	**Menge**	**Maximale Stromaufnahme**	**Spannung**	**Resultierende Leistung**
Kamera	1	160mA	3V	480mW
WLAN-Adapter	1	N/A	N/A	N/A
OLED-Display	1	78µA	3V	2.34mW
Ultraschall-Sensor	2	15mA	5V	75mW
Qwiic-Board	1	N/A	N/A	N/A
Motortreiber	2	25µA	3V	0.75µW

[10], [11], [12], [13]

Insgesamt brauchen die Peripherien, zu welchen man Daten zur Berechnung aus den Spezifikationen finden kann, eine Leistung von 632mW. Das heisst, wenn der 3A-Output stabil ist, bleiben also noch ca. 4.35W für die restlichen Peripherien übrig.

Die Frage, ob der 3A-Output stabil bleibt, auch wenn vom anderen Output für die Motoren eine grosse Leistung benötigt wird, bleibt jedoch offen und kann mit der verfügbaren Datenlage nicht geklärt werden.

Auf die Kapazität der Batterie wird nicht geachtet, da der Roboter jeweils nur kurz für Präsentationen gebraucht wird und danach wieder geladen werden kann. Eine Powerbank, die eine relativ kleine Kapazität von 10000mAh besitzt, kann den Roboter mit einer durchschnittlichen Belastung von 5A, was einer sehr hohen Rechen- und Fahrleistung entspricht, für 2h betreiben.

Übersicht Vorteile

- Mit Hilfe von Jupyter kann man mitverfolgen, was der Roboter «sieht» und wie er entscheidet
- Qwiic-Board erleichtert Ansteuerung des Displays und der Motoren
- Display funktioniert gut und ist informativ
- WLAN funktioniert nach Workaround zuverlässig mit «normalen» WPA2 Netzwerken
- Eigenständige und ausreichende Stromversorgung der Motoren
- Kamera funktioniert zuverlässig und Weitwinkel-Optik ist ein grosser Vorteil

Übersicht Nachteile

- Qwiic-Board besetzt gesamten GPIO-Header → Einschränkung bei Ansteuerung weiterer Sensoren
- Qwiic-Board und WLAN-Adapter erfordern ein veraltetes SparkFun-Image → weniger Flexibilität bei der Auswahl → keine Optimierung beim SD-Karten-Image
- WLAN-Adapter ist eine Bastelei und funktioniert nicht mit WPA2 Enterprise Netzwerken (eduroam)
- Der Roboter hat eine grosse Ausschwenkung beim Kurvenfahren (durch die lange Powerbank). Dadurch kann man weniger nah den Wänden entlangfahren.
- Powerbank
  - Kann nicht gleichzeitig geladen und verwendet werden
  - Schränkt die Performance des Jetson Nano eventuell ein und damit auch die möglichen Funktionalitäten eines zukünftigen Projektes

5.1.1.1 Anforderungen an einen besseren Roboter

Aufgrund der Evaluation des SparkFun JetBot werden hier die Anforderungen an einen perfekten Roboter definiert. Mit den hier erläuterten Anforderungen kann entschieden werden, ob ein nachfolgender Roboter für eine weiterführende Arbeit geeignet ist.

Tabelle 3: Priorisierte Anforderungen an Roboter

1. Priorität:	• Leistung muss vergleichbar oder besser sein als Jetson Nano
	• Leistung der Recheneinheit darf nicht eingeschränkt werden
	  • Nano Modul braucht eine Leistung von bis zu 10W
	  • Bei 5V muss die Stromversorgung also 2A ausschliesslich für das Nano Modul liefern können
	  • Stromversorgung soll bestenfalls 4A liefern**,** um einen gewissen Spielraum sicherstellen zu können
	• Separate Stromversorgung für Motoren
	• Informelles Display für IP-Adresse
	• Zuverlässiger WLAN-Adapter
	• Weitwinkel Kamera mit zuverlässiger Ansteuerung

2. Priorität:	• WLAN funktioniert mit WPA2 Enterprise Netzwerken
	  • Somit kann der Roboter ohne zusätzlichen Mehraufwand im OST («eduroam») Netzwerk integriert werden
	• Nicht vollbesetzte GPIO-Pins
	  • Ansteuerung von zusätzlichen Sensoren wird dadurch deutlich vereinfacht
	• Freie Wahl des Images, keine Gebundenheit an Hersteller
	• Roboter kann gleichzeitig geladen und programmiert / angesteuert werden
	• Wenderadius klein, um nahes Fahren von Kurven an Wänden zu ermöglichen

5.1.2 Waveshare Roboter

Der Waveshare Roboter wird offiziell von NVIDIA empfohlen.

Im User Guide [14] von Waveshare wird direkt auf das gleiche SD-Karten-Image verwiesen, auf welches auch beim SparkFun-Roboter verwiesen wird. Auch der Rest des User Guides ist sehr ähnlich, wie der von SparkFun. Das muss nicht unbedingt positiv oder negativ sein, denn beim SparkFun-JetBot geben die Images nur Probleme, weil der WLAN-Adapter nicht damit kompatibel ist. Da beim Waveshare-JetBot eine andere WLAN-Hardware verwendet wird, ist es möglich, dass hier alles wie beschrieben funktioniert.
Ob es jedoch zu Komplikationen kommen würde, wenn man ein eigenes, perfekt auf das Projekt abgestimmtes Image verwenden würde, kann zum jetzigen Zeitpunkt nicht mit Sicherheit beantwortet werden.

Stromversorgung

Einer der wichtigsten Punkte in der Prioritätenliste ist, dass der Jetson Nano nicht durch die Stromversorgung eingeschränkt wird.

Das Jetson Nano Modul benötigt bei 5V 2A. Damit der Jetson Nano mit Sicherheit zuverlässig funktioniert, soll er mit 4A gespiesen werden. Die Motoren werden separat gespiesen. Die Stromversorgung wird beim Waveshare-JetBot durch drei seriell-geschaltete wiederaufladbare 18650 Batterien sichergestellt. Jede Batterie läuft mit 3.7V. Das heisst, wenn die Batterien vollgeladen sind, kommen sie zusammen auf ungefähr 11.1V. Wie man aus dieser Seite [15] entnehmen kann, kann so eine Batterie bis zu 4A liefern. Daraus ergibt sich eine theoretisch maximale Leistung von 44.4W auf der Eingangsseite des Spannungswandlers. Aus dem von Waveshare erstellten Schema [16] geht hervor, dass die Umwandlung auf 5V mittels dem Baustein APW7313 erfolgt. Aus dem Datenblatt [17] des APW7313 geht hervor, dass bei genügender Eingangsleistung ein Ausgangs-Strom von bis zu 3A gezogen werden kann. Das heisst, der Jetson Nano kann mit dieser Stromversorgung wegen dem von Waveshare eingesetzten Spannungswandler mit maximal 3A betrieben werden. Da die Motoren für eine Spannung zwischen 3V-6V ausgelegt sind, muss davon ausgegangen werden, dass die Motoren ebenfalls an der 5V-Speisung hängen. Das bedeutet, von den 3A muss man noch den Strom abziehen, welcher die Motoren benötigen. Dieser Strom liegt im Extremfall bei bis zu 2.4A. Dadurch würde im schlimmsten Fall der Jetson Nano sogar abstürzen, weil er kurzzeitig nicht mehr genügend Leistung erhält.

Diese Stromversorgung ist aus oben genannten Gründen unzureichend und noch schlechter als die vom SparkFun-JetBot.

Display

Ein Display ist vorhanden und standardmässig wird auf diesem die IP-Adresse angezeigt. Um den Jetson Nano mit dem Display zu verbinden, wird kein zusätzliches Board benötigt. Das Display befindet sich direkt auf dem Board, auf welchem sich auch die Batterie und der Motorentreiber befinden. Dieses Board und der Jetson Nano werden mit 6 Kabeln miteinander verbunden.

WLAN

Hier wird eine andere Lösung verwendet, um dem Jetson Nano die Möglichkeit zu geben, ohne Kabel zu kommunizieren. Statt einem USB-Dongle wird ein WLAN-Karte [18] zusammen mit zwei Antennen [19] gebraucht. Diese Karte ermöglicht, wie auch der USB-Dongle, keine Verbindung zu WPA2 Enterprise Netzwerken.

Durch Recherchen in Foren konnten aber keine Probleme mit dieser WLAN-Karte gefunden werden. Dies ist ein signifikanter Unterschied, da unzählige Entwickler Probleme mit dem USB-Dongle von SparkFun haben und ein Workaround zwingend erforderlich ist, damit der USB-Dongle zuverlässig funktioniert.

Kamera

Die Kamera [19] des Waveshare-JetBots hat sogar noch einen weiteren Winkel als der vom SparkFun-JetBot und zwar entspricht der horizontale Winkel dieser Kamera 160°.

Die Auflösung entspricht 3280x2464 Pixel. Dies ist ausreichend für den Roboter.

5.1.3 Silicon Highway Roboter

Der JetBot, welchen man von Silicon Highway kaufen kann, unterscheidet sich von den anderen Robotern, weil er nicht alles zur Verfügung stellt. Das Gehäuse sowie die Halterung für die Kamera müssen noch selbst 3D-gedruckt werden.

Silicon Highway orientiert sich sehr stark an der «Bill of Materials» [20] von NVIDIA.

Stromversorgung

Zur Stromversorgung wurde die von NVIDIA vorgeschlagene Powerbank verwendet.

Im Kapitel «NVIDIA-designed open-source JetBot» findet man eine genauere Beschreibung der Powerbank und deren Leistung.

Display

Es wird das von NVIDIA vorgeschlagene Display eingesetzt. Dies ist das gleiche Display, das auch beim Waveshare-JetBot verbaut ist. Dadurch kann erwartet werden, dass keine Probleme mit dem Display auftreten werden.

WLAN

Um eine zuverlässige WLAN-Verbindung sicherzustellen wurde die von NVIDIA vorgeschlagene Option gewählt, dass man eine WLAN-Karte zusammen mit zwei Antennen verbaut.

Mehr dazu im Kapitel «NVIDIA-designed open-source JetBot».

Kamera

Es wird die Kamera «RPi Camera V2» verwendet. Diese ist eine von drei zur Verfügung stehenden Optionen in der JetBot-Anleitung von NVIDIA.

Mehr dazu steht im Kapitel «NVIDIA-designed open-source JetBot».

5.1.4 NVIDIA-designed open-source JetBot

Es ist kein Zufall, dass alle Roboter-Kits, die einen Jetson Nano verwenden, JetBot heissen. NVIDIA stellt nämlich eine JetBot-Anleitung [21] zur Verfügung, die es einem ermöglicht, alle Teile einzeln zu kaufen und sich das Roboter-Gehäuse selbst zu drucken. Nun haben es dem Kunden mehrere Drittparteien, wie SparkFun und Waveshare, einfacher gemacht, indem sie das Gehäuse inklusive aller benötigten Teile in einem Kit verkaufen. NVIDIA hat jedoch mehrere Optionen zur Verfügung gestellt, wie man zum Beispiel die WLAN-Konnektivität oder die Stromversorgung umsetzen kann. Zudem haben sich die Drittparteien einige Freiheiten genommen und diese Anleitung nach Belieben angepasst. Darum sind dann sehr ähnliche Roboter entstanden, welche sich aber in ihrer Zusammensetzung trotzdem stark unterscheiden.

Es ist nach wie vor möglich, sich einen eigenen JetBot zusammenzustellen und sich das Gehäuse selbst zu drucken. In diesem Abschnitt werden die verschiedenen von NVIDIA vorgeschlagenen Optionen einander gegenübergestellt und der bestmögliche NVIDIA-JetBot zusammengestellt.

Stromversorgung

NVIDIA schlägt folgende Powerbank [22] vor, um die Stromversorgung für den JetBot sicherzustellen. Die Powerbank hat zwei USB-Ports, die beide für einen Strom von 3A ausgelegt sind.

Der eine Port wird für die Motoren benutzt. Da beide Motoren zusammen maximal 2.6A benötigen kann man beide Motoren auf voller Leistung benutzen, ohne Angst haben zu müssen, dass man die Spezifikationen der Powerbank überschreitet. Dadurch kann man sicher sein, dass die Belastung der Motoren keinen Einfluss auf den zweiten Port hat.

Am zweiten Port wird der Jetson Nano angeschlossen. Um sicherzustellen, dass der Jetson Nano jederzeit mit der maximalen Leistung laufen kann, benötigte man bei 5V 4A. Somit ist dieser zweite Port gleich leistungsstark wie der des SparkFun-Roboters. Trotzdem wird diese Powerbank besser eingeschätzt als die von SparkFun, weil die Spannung bestimmt nicht einbricht, wenn die Motoren stark belastet werden. Dies ist bei der SparkFun-Powerbank nicht mit Sicherheit der Fall. Zusätzlich wird das Display über den Motorentreiber gespiesen, statt wie beim SparkFun-JetBot über den Jetson Nano.

Display

Es wird das Display verwendet, welches auch beim Waveshare-JetBot und beim Silicon-Highway-JetBot eingesetzt wird. Es wird erwartet, dass das problemlos funktioniert und die IP-Adresse mithilfe des NVIDIA-SD-Karten-Image automatisch angezeigt wird.

WLAN

Um eine WLAN-Verbindung herzustellen, stellt NVIDIA zwei Optionen zur Verfügung.

Option 1: Wifi-Karte + Antenne

Diese Variante wird vom Waveshare-JetBot eins zu eins übernommen.

Mittels Google-Suche kann kein Hinweis dafür gefunden werden, dass diese Variante zu Problemen führt. Das kann man jedoch erst mit Sicherheit sagen, wenn man die Variante selbst getestet hat.

Mit WPA2 Enterprise Netzwerken (zum Beispiel «eduroam») kann man sich gemäss Spezifikation [18] jedoch nicht verbinden.

Option 2: USB-Dongle

Diese Variante wurde von SparkFun umgesetzt. Sie haben jedoch einen anderen USB-Dongle verwendet, als von NVIDIA vorgeschlagen wird. Eine Begründung für diese Wahl kann nicht gefunden werden.
Es ist bekannt, dass der USB-Dongle von SparkFun zu vielen Problemen führt, welche mit einem Workaround jedoch weitgehend gelöst werden können.

Der erste USB-Dongle [23], der von NVIDIA empfohlen wird, ist ein kleiner WLAN-Adapter. Der zweite von NVIDIA empfohlene USB-Dongle [24] ist ein WLAN-Adapter mit einer leistungsstarken Antenne. Diese Antenne ermöglicht eine leistungsstärkere WLAN-Verbindung. Dadurch wird es für den Roboter möglich, sich weiter vom WLAN-Hotspot zu entfernen.

Um den ersten Adapter auf dem Jetson Nano zum Laufen zu bringen, muss ebenfalls ein Workaround [25] verwendet werden. Diese Lösung scheint jedoch anhand von Kommentaren nicht immer zu funktionieren. Ausserdem gibt es allgemein sehr wenige Suchergebnisse zu der Kombination aus diesem Adapter und dem Jetson Nano.
Für den zweiten Adapter wurde mittels Google-Suche kein Eintrag für ein mögliches Fehlverhalten gefunden. Das bedeutet jedoch nicht unbedingt, dass nie Fehler auftreten.

Keiner der beiden von NVIDIA empfohlenen WLAN-Adapter unterstützen eine Verbindung mit WPA2 Enterprise Netzwerken.

Tabelle 4: Vergleich WLAN-Optionen

Kriterien	**USB-Dongle**	**USB-Dongle mit Antenne**	**Option WLAN-Karte**
WLAN zuverlässig	eventuell nach Workaround	unbekannt	Ja, wahrscheinlich
Signalstärke	Mittel	Stark	
Bluetooth	Nein		Ja

Kamera

NVIDIA stellt drei Optionen zur Verfügung, welche Kameras man verwenden kann.

Option 1 (Standard): Leopard Imaging Camera

Die Standard-Option ist dieselbe Kamera [6], welche auch von SparkFun verwendet wird. Die Kamera hat ein Weitwinkel von 145° und eine Auflösung von 3280x2464 Pixel bei 21 Bildern pro Sekunde.

Option 2: IMX219-160 Camera

Bei der zweiten Option wird die gleiche Kamera [26] verwendet, die auch von Waveshare verbaut wird. Diese Kamera überzeugt mit einem Weitwinkel von 160° und einer Auflösung von maximal 3280x2464 Pixel. Die Kamera kann mit 30 Bildern pro Sekunde mit einer Auflösung von 2560×2048 Pixel verwendet werden.

Option 3: RPi Camera V2 + Lens Attachment

Die dritte Option besteht aus der Kamera [27], die sonst für den Raspberry Pi verwendet wird und einem Weitwinkel-Aufsatz [28], der einen Winkel von 160° ermöglicht.

Die Kamera filmt mit einer Auflösung von 1920x1080 Pixel bei 30 Bildern pro Sekunde. Dies ist eine deutliche Verschlechterung gegenüber den ersten beiden Optionen.

Tabelle Kamera-NVIDIA-JetBot

Tabelle 5: Vergleich Kameras

Kriterien	**Leopard Imaging Camera**	**IMX219-160 Camera**	**RPi Camera + Lens Attachment**
wird eingesetzt von	SparkFun	Waveshare	Raspberry Pi
Winkel	145°	160°	
Maximum Image Transfer Rate	8MP * 21fps = 168MP/s	5MP * 30fps = 150MP/s	2MP * 30fps = 60MP/s

Tabelle Roboter Vergleichsmatrix

Tabelle 6: Roboter Vergleichsmatrix

Pr.	Kriterien	SparkFun-JetBot	Waveshare- JetBot	NVIDIA- JetBot	Silicon-Highway- JetBot
Hoch	Recheneinheit	Jetson Nano			
Hoch	Stromversorgung gesamt	5V * (2A + 3A) = 25W	5V * 3A = 15W	5V * (3A + 3A) = 30W	
Hoch	- Stromversorgung Motoren	5V * 2A = 10W	Gesamtleistung geteilt, je nach Gebrauch	5V * 3A = 15W	
Hoch	- Stromversorgung Jetson Nano	5V * 3A = 15W		5V * 3A = 15W	
Hoch	Stromversorgung der Motoren läuft nicht über Rechen-einheit	Ja			
Hoch	Display (zeigt IP-Adresse)	vorhanden			
Hoch	WLAN zuverlässig	Ja, nach Workaround	Ja, wahrscheinlich	siehe Tabelle WLAN-NVIDIA-JetBot	Ja, wahrscheinlich
Hoch	Kamera zuverlässig	Ja		siehe Tabelle Kamera-NVI-DIA-JetBot	Ja, wahrscheinlich
Hoch	Kamera Winkel	145°	160°		160°
Hoch	Kamera Maximum Image Transfer Rate	8MP * 21fps = 169MP/s	5MP * 30fps = 157MP/s		2MP * 30fps = 60MP/s
Mittel	WLAN läuft mit WPA2 Enterprise	Nein			
Mittel	Freie GPIO-Pins, um zusätzliche Sensoren anzusteuern	Eventuell nach Workaround	Ja		
Mittel	Image kann frei gewählt werden	Nein (wegen Qwiic)	Ja		
Mittel	Roboter kann während dem Laden verwendet werden	Nein			
Mittel	Ausschwenkung beim Fahren von Kurven	Gross	Klein	Mittel	

5.1.5 Entscheid

Um sicherzugehen, dass das weiterführende Projekt reibungslos funktioniert, wird ein selbst zusammengestellter JetBot im Rahmen der von NVIDIA zur Verfügung gestellten Komponenten [20], verwendet.

Eine vollständige Liste mit den verwendeten Komponenten ist im Abschnitt «Bill of Materials» zu finden.

Stromversorgung

Zur Stromversorgung wird, die von NVIDIA vorgeschlagene Powerbank verwendet, da NVIDIA diese als einzige Option auflistet und diese Option im Vergleich mit den Stromversorgungen der anderen JetBots besser abgeschnitten hat.

Display

Es wird das von NVIDIA vorgeschlagene Display verwendet, da es das einzige von NVIDIA verwendete Display ist und online keine Hinweise dazu gefunden werden können, dass dieses Display zu unerwünschten Fehlern führen könnte.

WLAN

Um eine zuverlässige WLAN-Verbindung sicherzustellen, wird eine WLAN-Karte [29] zusammen mit 2 WLAN-Antennen [30] verwendet.

Diese Variante wurde gewählt, weil die USB-Dongle-Variante zu unzuverlässig erscheint.

Kamera

Es wird die von Waveshare eingesetzte Kamera IMX219-160 [26] verwendet. Diese überzeugt durch ihren weiten Winkel und schnelle Übertragungsrate.

Bestellung

Bill of Materials

In der folgenden Tabelle sind alle benötigten Komponenten aufgelistet. Die jeweiligen Anbieter sind jeweils die von NVIDIA vorgeschlagenen Links, mit Ausnahme vom Jetson Nano. Ausserdem musste der Preis vom Jetson Nano von 99$ auf 175$ angehoben werden.

Die Tabelle dient dazu, die Kosten eines selbst zusammengestellten JetBot ungefähr abschätzen zu können.

Tabelle 7: Bill of Material

Komponente	Anzahl	Kosten pro Stück [$]	Kosten gesamt [$]	Anbieter	Bemerkung
Jetson Nano	1	175,00	175,00	Distrelec	-
Micro SD-Karte	1	13,99	13,99	Amazon	64GB

Motor	2	5,90	11,80	Amazon	TT-Form
Motorentreiber	1	19,95	19,95	Amazon	-
Caster Ball	1	6,30	6,30	Amazon	-
Stromversorgung	1	15,99	15,99	Amazon	2x 5V/3A
USB-Kabel Pack	1	6,99	6,99	Amazon	angewinkelt
PiOLED display	1	14,95	14,95	Amazon	-
PiOLED header	1	5,95	5,95	Amazon	-
Chassis	1	0,00	0,00	STL-File	3D-Druck
Kamera-Halter	1	0,00	0,00	STL-File	3D-Druck
Kamera	1	29,90	29,90	Amazon	-
WLAN-Karte	1	18,95	18,95	Amazon	-
WLAN-Antenne	2	5,06	10,12	Arrow	-
Rad	2	5,00	10,00	Adafruit	-
Caster base	1	0,00	0,00	STL-File	3D-Druck
Caster shroud	1	0,00	0,00	STL-File	3D-Druck
Doppelseitiger Kleber	2	0,07	0,14	Amazon	-
Schrauben M2	20	0,07	1,40	Amazon	-
Schrauben M3	4	0,12	0,48	Amazon	-
Muttern M3	4	0,06	0,24	Amazon	-
Jumper-Kabel	4	0,03	0,12	Amazon	-
Gesamt:			342,27		

Gemäss dieser Berechnung kostet ein selbst zusammengestellter JetBot ungefähr 350$. In diesen 350$ ist der 3D-Druck nicht enthalten.

Die Kosten des JetBot ohne 3D-Druck und ohne Jetson Nano belaufen sich auf ungefähr 175$.

Der SparkFun-JetBot kostet ohne Jetson Nano 244 CHF und der Waveshare-JetBot kostet ohne Jetson Nano 123$.
Der Bausatz von Silicon Highway kostet ohne 3D-Teile und Jetson Nano 255 CHF.

Alternative

Damit man nicht alles einzeln einkaufen muss und um Zeit zu sparen, kann man einfach einen fertig zusammengestellten Bausatz von Silicon Highway bestellen.
Dieser kommt jedoch mit der falschen Kamera. Folglich muss diese zusätzlich bestellt werden.

Kosten: Silicon-Highway-Bausatz (255CHF) + Kamera (30CHF) = 285CHF
[31], [26]

Diese Kosten beinhalten noch keine 3D-Teile und keinen Jetson Nano.

5.2 Object Detection

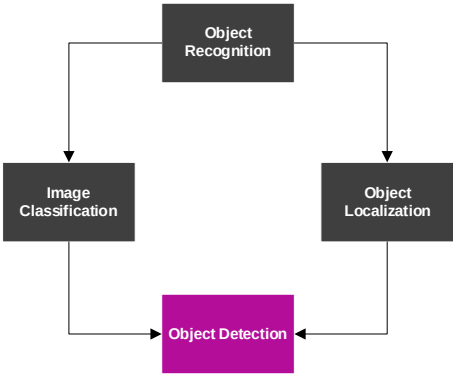


Abbildung 3: Object Detection Schema

Für die Auswahl des besten Object Detection Algorithmus sind drei der momentan erfolgreichsten Algorithmen evaluiert worden:

- Faster R-CNN
- YOLO v3
- SSD

In den folgenden Kapiteln wird aufgezeigt, wie diese grundlegend funktionieren und wie sie nach den Faktoren Performance und Accuracy (mAP) evaluiert worden sind.

5.2.1 Implementierung

Zur Überprüfung der Machbarkeit und Handhabung der verschiedenen Lösungen sind diese in «Google Colab» implementiert. Programmiert ist alles in Python 3 mit der Verwendung des TensorFlow Frameworks (für Faster R-CNN und SSD) und Darknet (für YOLOv3).

Alle verwendeten Implementierungen geben die Möglichkeit, die Modelle weiter zu trainieren. Dies lässt die Option offen, das Model auf selbst definierte Objekte zu trainieren.

Für die Implementierung wurden Modelle verwendet, welche mit einem grossen Object Detection, Segmentation und captioning Datenset namens COCO (Common Objects in Context) trainiert wurden [32].

Die Algorithmen sind alle für das Jetson (NVIDIA TensorRT) Framework optimierbar und grössten teils auch schon optimiert verfügbar [33].

5.2.2 Faster R-CNN

Faster R-CNN ist ein Object Detection Algorithmus, welcher 2015 von einem Team bei «Microsoft Research» entwickelt wurde. Anders als seine Vorgänger (R-CNN & Fast R-CNN) verwendet er nicht mehr einen «selective search Algorithmus», sondern einen Algorithmus, welcher ein separates Netzwerk «region proposal» lernen lässt. Dieses Netzwerk, das RPN, bringt die Vorschlagszeit von ca. 2 Sekunden auf ca. 10 Millisekunden herunter im Vergleich zu den vorherigen Versionen. Die «region proposals» werden dann mit Hilfe des «RoI Pooling Layers» reshaped. Dies wird dann wiederum verwendet, um das Bild innerhalb der «proposed region» zu klassifizieren und die Versatzwerte der Begrenzungsrahmen vorherzusagen [34], [35], [36], [37], [38].

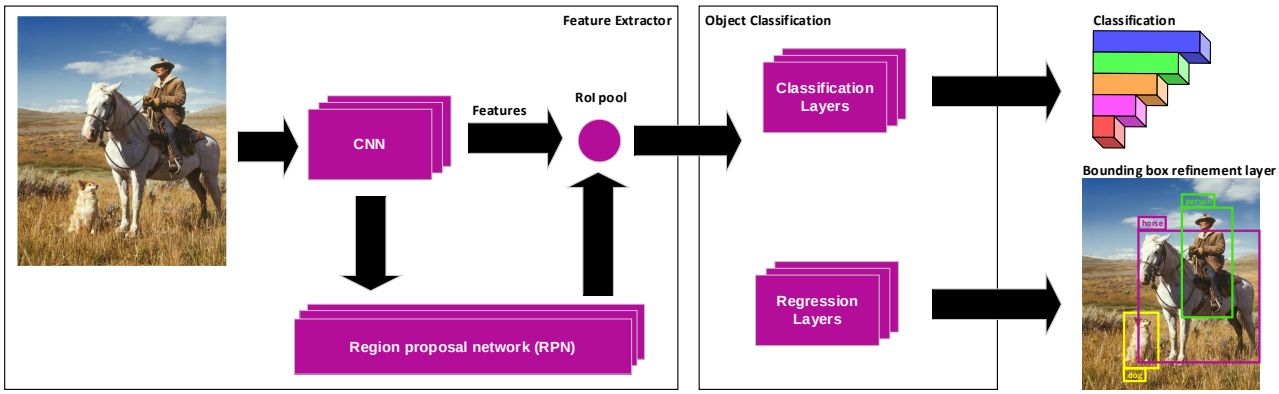


Abbildung 4: Faster R-CNN - Prozess

5.2.3 YOLO v3

YOLO (You Only Look Once), ist im Gegensatz zu Faster R-CNN ein Algorithmus basierend auf Regression. Das bedeutet, dass der Algorithmus anstatt des Selektierens von interessanten Teilen eines Bildes, Klassen und Begrenzungsrahmen für das ganze Bild in einem Durchlauf des Algorithmus prognostiziert. Der Algorithmus splittet das Bild in Zellen. Diese sind wiederum dafür verantwortlich, die Begrenzungsrahmen vorherzusagen. Danach werden mittels einem vorhergesagten «pc-Wert» die Boxen mit tiefer Objektwahrscheinlichkeit entfernt. Begrenzungsrahmen mit der grössten gemeinsamen Fläche werden durch «Non-Max-Suppression» zusammengefasst [36], [37], [39], [40], [41], [42], [43].

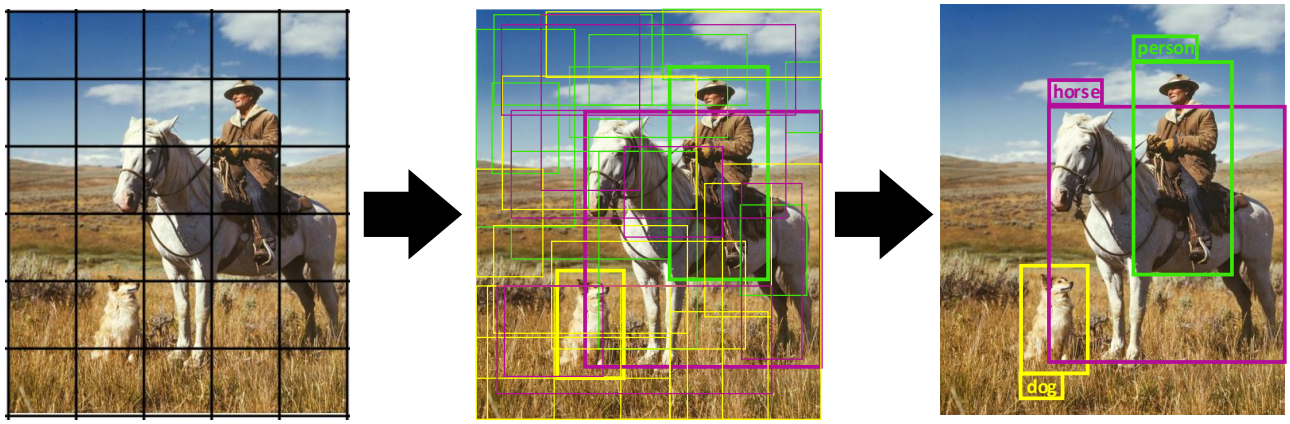


Abbildung 5: YOLO v3 - Prozess

5.2.4 SSD

SSD (Single Shot MultiBox Detector) basiert ebenfalls auf Regression (Single Shot). Zusätzlich verwendet es eine MultiBox Technik für die Begrenzungsrahmenregression und einen sogenannten Detector für die Klassifizierung der Objekte. SSD nutzt ein «single stage object detection network», welches Vorhersagen anhand von Multiskalen-Features zusammenführt. Auf einem Bild wird ein Deep-Learning-CNN ausgeführt,

um Vorhersagen aus mehreren Feature-Maps zu erstellen. Der Detector sammelt und decodiert Vorhersagen dann, um Begrenzungsrahmen zu generieren [37], [44], [45], [46], [47].

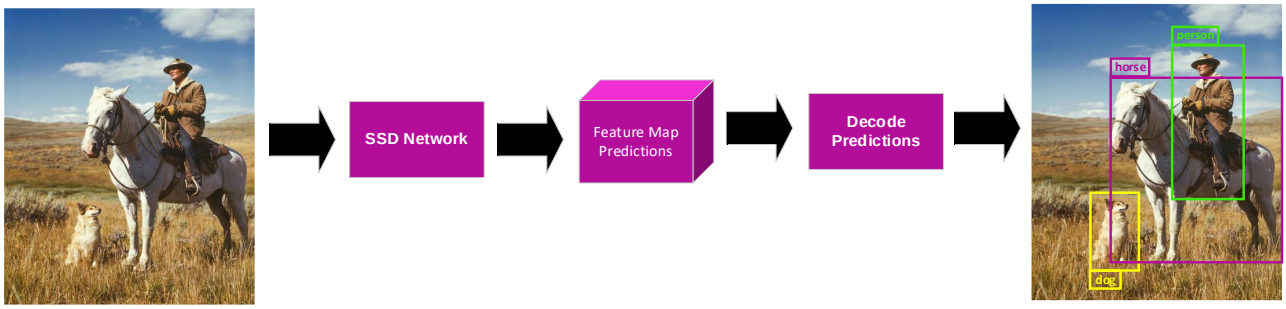


Abbildung 6: SSD - Prozess

5.2.5 Mathematische Grundlagen

Um die mAP (mean Average Precision) für unsere Tests berechnen zu können, benötigen wir die folgenden mathematischen Grundlagen:

Precision misst, wie akkurat die Vorhersage ist (Anteil der richtig vorhergesagten positiven Ergebnisse zu allen vorhergesagten positiven Ergebnissen).

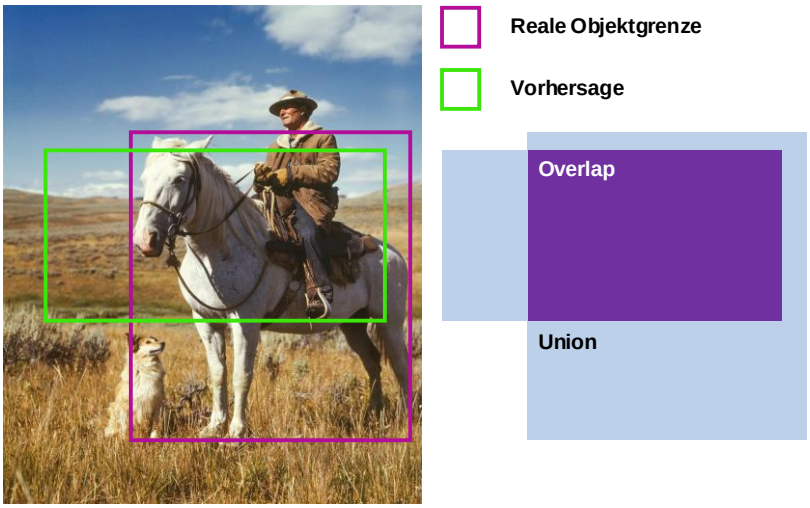
$$Precision = \frac{TP}{TP + FP}$$

Recall misst, wie vollständig alle positiven Ergebnisse gefunden werden (Anteil der richtig positiv klassifizierten Ergebnisse zu allen tatsächlich positiven Ergebnissen).

$$Recall = \frac{TP}{TP + FN}$$

True positive (TP): Auf einem Bild ist ein Pferd. Es wird getestet ob ein Pferd erkannt wird und es wird tatsächlich erkannt, somit bekommt man TP zurück.
True negativ (TN): wäre hingegen: Wenn es auf dem Bild kein Pferd hätte. Es wird getestet, ob ein Pferd erkannt wird und man bekommt die Antwort «kein Pferd vorhanden» als TN.
False positive (FP): Ist das gleiche wie TP einfach in Bezug auf falsche Werte.
False negative (FN): Ist das gleiche wie TN einfach in Bezug auf falsche Werte.

IoU (Intersection over Union) misst den Overlap zwischen zwei Flächen. Dies wird hier verwendet, um zu messen, wie fest die Vorhersage mit der Realen Objektgrenze überlappt. Mit einem vordefinierten IoU kann dann beispielsweise bei einer Klassifizierung entschieden werden, ob eine Vorhersage «true positive» oder «false positive» ist. (Beispielsweise IoU > 0.5 → true positive || IoU < 0.5 → false positive)



$$IoU = \frac{Overlap\ Fläche}{Union\ Fläche}$$

Abbildung 7: IoU

IP (Interpolated Precision) definiert den höchsten Precision-Wert für einen bestimmten Recall-Level.

$$\boldsymbol{P}\text{interp(r)} = \max_{r' \geq r} p(r')$$

mAP (durchschnittlicher Präzision Mittelwert) wird mit der 11 Punkte Interpolation Technik berechnetet (an 11 unterschiedlichen Recall-Leveln).

$$mAP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \boldsymbol{P}\text{interp(r)}$$

[48], [49], [50], [51], [52]

5.2.6 Vergleich

Es ist sehr schwer, einen fairen Vergleich zwischen unterschiedlichen Objekterkennungsalgorithmen zu machen. Wichtig ist die Balance zwischen Performance und Genauigkeit, speziell in Bezug auf Echtzeit Programme.

Die Tabelle zeigt die Unterschiede der Algorithmen hinsichtlich des mAP. Die Daten wurden einer Studie des «Facebook AI Researches» und einer der «University of Washington» entnommen [43], [53].
Die Tests wurde mit der COCO test-dev durchgeführt.

Tabelle 8: mAP und AP Auswertung

	$mAP_{(.50:05:.95)}$	$AP_{(IoU=0.5)}$	$AP_{(IoU=0.75)}$	mAP_S	mAP_M	mAP_L
Faster R-CNN-512	36.2	59.1	39.0	18.2	39.0	48.2
SSD-513	31.2	50.4	33.3	10.2	34.5	49.8
YOLOv3-608	33.0	57.9	34.4	18.3	35.4	41.9

*Die Zahlen hinter den Algorithmen, also beispielsweise SSD-**513,** steht für die Inputgrösse des Bildes, hier 513x513.*
mAP_S = % mAP für kleine Objekte: Fläche < 32^2 , mAP_M = % mAP für mittlere Objekte: 32^2 < Fläche < 96^2 , mAP_L = % mAP für grosse Objekte: Fläche > 96^2

Erklärung

In COCO gibt es mehr kleine Objekte als große Objekte. Spezifisch: Ungefähr 41% der Objekte sind klein (Fläche <32^2), 34% sind mittel (32^2 <Fläche <96^2) und 24% sind gross (Fläche> 96^2). Die Fläche wird als Anzahl der Pixel in der Segmentierungsmaske gemessen [54].
mAP_S steht also zum Beispiel für den mAP, welcher bei einem Test spezifisch auf kleine Objekte entstand.

Die nächste Tabelle zeigt den mAP im Verhältnis zur durchschnittlichen Berechnungszeit mit einem IoU von 0.5, dieser wurde von uns auch bei früheren Tests für die Echtzeit-Objekt-Erkennung verwendet. Die Daten in den Tabellen sind denselben Studien entnommen wie die in der letzten.
Für diesen Vergleich wurden zusätzlich noch andere Bild Inputgrössen aufgelistet.

Tabelle 9: Speed / Accuracy Tradeoff

	$AP_{(IoU=0.5)}$	**Zeit (*ms*)**
Faster R-CNN-512	59.1	172.0
SSD-321	45.4	61.0
SSD-513	50.4	125.0
YOLOv3-320	51.5	22.0
YOLOv3-608	57.9	51.0

Aus den vorherigen Daten entnehmen wir folgende Vor- und Nachteile.

Tabelle 10: Vor- und Nachteile Algorithmen

	Vorteil	**Nachteil**
Faster R-CNN	RPN-Methode erlaubt es, die Objekterkennung nahezu in Echtzeit zu erfolgen.	Auch mit der Effizienz des Algorithmus ist er nicht schnell genug, um in Programmen verwendet zu werden, welche Echtzeit benötigen.
YOLO v3	Die Lokalisierung von Objekten ist sehr effizient und ermöglicht somit die Verwendung in Echtzeit-Programmen.	Zeigt Probleme beim Erkennen von kleinen Objekten, was jedoch bei den anderen Algorithmen auch der Fall ist.

SSD	Durch die Verwendung eines einzelnen Netzwerks ist die Lokalisierung von Objekten im Vergleich zu Faster R-CNN schneller und ist somit ebenfalls für Verwendung in Echtzeit-Programmen geeignet.	Die Genauigkeit bei der Erkennung von Objekten ist im Vergleich zu Faster R-CNN schlechter.

Faster R-CNN wird auf Basis der Erkenntnisse aus den Evaluierungen ausgeschlossen. Der Algorithmus gibt zwar im Vergleich die akkuratesten Resultate, ist für das weiterführende Projekt jedoch zu langsam. Dazu kommt, dass er zwei Modelle für die Object Detection verwendet, was dazu führt, dass mehr Rechenleistung benötigt wird. Dies könnte bei einer eventuell zukünftigen Implementierung einer Sprachsteuerung oder anderen Modulen sehr schnell dazu führen, dass man an die Leistungsgrenzen des Roboters stösst [43], [38], [55], [47], [56], [57].

5.2.7 Entscheid

YOLOv3 liegt im Geschwindigkeits-/Zeitvergleich klar vorne. Hier muss jedoch auch berücksichtigt werden, dass sich die Inputgrössen der Bilder, welche vom Algorithmus angenommen werden, unterscheiden.

Wenn man die Algorithmen bezüglich der Coding-Dokumentation vergleicht, so ist SSD nach unseren Nachforschungen ganz klar besser dokumentiert und läuft sauber mit dem TensorFlow Framework.

Bei weiteren Forschungen spezifisch auf den Jetson Nano bezogen, sind wir auf das Detection Model SSD Mobilenet V2 von TensorFlow Google gestossen. Dieses Modell muss durch die Verwendung von «Depthwise Separable Convolutions» ca. 9-mal weniger Arbeit leisten wie vergleichbare neuronale Netzwerke. Zudem nutzt es das Modell «Residual Connection». Dies ermöglicht es, Gradienten direkt durch ein Netzwerk fliessen zu lassen, ohne dabei nicht-lineare Aktivierungsfunktionen zu durchlaufen.

Das Modell ermöglicht somit eine Geschwindigkeitsoptimierung sowie eine Verringerung des Batterieverbrauchs im Vergleich zu anderen Modellen.

Durch die Möglichkeit, dass Echtzeit-Objekterkennung mit SSD möglich ist und andere Beispielsprojekte diesen Algorithmus bereits effizient auf dem Jetson Nano verwenden, ging der SSD Algorithmus als Sieger aus dieser Evaluation für die hier definierten Vorgaben heraus [44], [55], [58].

5.3 Algorithmische Pfadfindung in einem Raum

Da sich der Roboter für Präsentationszwecke in einem Modell aus zusammenhängenden Räumen bewegt, muss ein Algorithmus zur Pfadfindung implementiert werden. Klassische Algorithmen wie der A*-, Dijkstra- oder Bellman-Ford-Algorithmus kommen für das erste Durchlaufen der Räume nicht in Frage, da sie zu Beginn Informationen über den ganzen Raum besitzen müssen.

5.3.1 Wände abfahren, bis Ziel erkannt

Dieser Algorithmus funktioniert in jedem abgeschlossenen Raum und in zusammenhängenden Räumen, auch wenn man im Voraus keinerlei Informationen hat, wie dieser aussieht. Der Roboter fährt bei diesem Algorithmus der Wand nach, bis er das gesuchte Objekt irgendwo sichten kann. Wenn er das Objekt erkannt hat, fährt er direkt darauf zu und weicht dabei, wenn nötig, Hindernissen aus.

Mit unserer eigenentwickelten Web-App[4] kann man eine solche Lösungsvariante visualisieren. Mit der App können verschiedene Raumkonstellation getestet werden und es wird bei jedem Schritt aufgezeigt, was die für den Roboter wichtigen Detektionen sind. Wie diese Hindernisse oder Wände detektiert werden, wird im nächsten Kapitel behandelt. Die Applikation wurde in JavaScript geschrieben und kann für eine spätere Implementation als Vorlage[5] genutzt werden. Die Anleitung, wie Tests in der App durchgeführt werden können, findet man unter dieser URL: «https://algo-vis-robot.azurewebsites.net/readme».

Tabelle 11: Web-App, Visualisierung der algorithmischen Pfadfindung

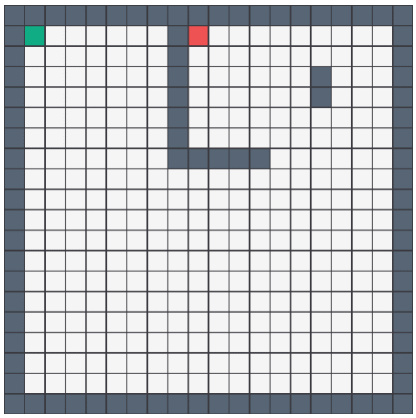


Abbildung 8: Web-App, theoretische Darstellung der Testumgebung

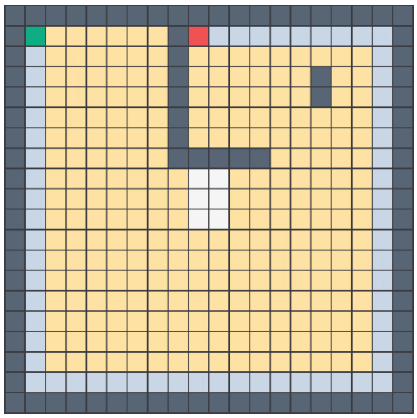


Abbildung 9: Web-App, theoretische Darstellung der Testumgebung nach Durchlauf des Algorithmus

[4] https://algo-vis-robot.azurewebsites.net/
[5] https://github.com/yV11/algo-vis-robot

Diese Variante eignet sich hervorragend für das Finden von Objekten in geschlossenen Räumen, solange diese nicht zu gross sind. Dies würde nämlich dazu führen, dass es nicht mehr möglich wäre, ein Objekt, das sich in der Mitte des Raumes befindet, von einer Wand aus zu erkennen. Das heisst, das Ziel im Raum muss durch die Weitwinkel-Kamera erfasst werden können.

Um den schnellsten Weg nach dem Erkunden der Räume zu finden, kann ein Pfadfindungsalgorithmus wie der Dijkstra Algorithmus (da es keine negativen Kantengewichte gibt) verwendet werden. Dadurch kann der Roboter auf dem Rückweg zum Startpunkt den «Shortest Path» verwenden. Dies ist aber eine optionale Erweiterung, die in einem zukünftigen Projekt nicht mit oberster Priorität behandelt wird. Eine mögliche Implementation kann aber in der Web-App mit dem Befehl */dijkstra* nach dem Erkunden und Finden des Ziels getestet werden.

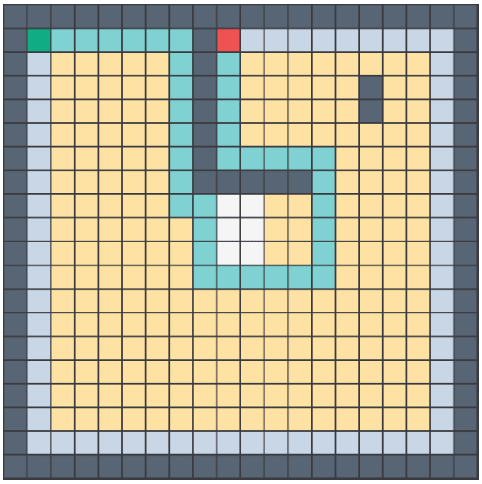


Abbildung 10: Web-App, theoretische Darstellung der Testumgebung mit Befehl /dijkstra und Shortest Path

Der Algorithmus verwendet die sechs weissen Felder in der Mitte nicht, da sie nie von einer Kamera erfasst wurden. Trotzdem würde der Roboter beim Verwenden des «Shortest Path» 24 Felder einsparen und dadurch deutlich schneller sein.

Abbildung 11: QR-Code mit URL zur eigenentwickelter Web-App

5.4 Hindernis-Erkennung

5.4.1 Lösungsvariante Ultraschall

Eine Lösung mithilfe von zwei Ultraschallsensoren ist sehr effizient, da keine weiteren, grösseren Hardware-Ressourcen gebraucht werden. Dies ist ein riesigerer Vorteil, da die Ressourcen so oder so schon stark begrenzt sind. Die Ultraschallsensoren werden am Roboter so platziert, dass sie die Distanz nach vorne sowie zu einer Seite (rechts) messen können. Durch diese gewonnenen Informationen ist es möglich, dass sich der Roboter anhand der zusammenhängenden Wände orientiert und so durch alle Räume bewegen kann. Um die Implementation der Ultraschallsensoren möglichst ressourcenschonend zu halten, wird dafür kein Modell antrainiert. Stattdessen werden die Messungen rein algorithmisch ausgewertet.

Nachteile

Um die Ultraschallsensoren korrekt einsetzen zu können, muss man in einer optimierten Umgebung arbeiten. Das heisst, man geht bei dieser Lösungsvariante davon aus, dass man immer eine durchgehende Wand an der Seite und vor dem Roboter hat.
Diese Variante eignet sich darum sehr gut in einer modellierten Testumgebung. Sie ist jedoch so nicht einsetzbar, wenn man den Roboter zum Beispiel einfach auf dem Tisch platzieren will. Dann würde der Roboter nämlich die Tischkante nicht als «Wand» erkennen und vom Tisch fallen.

5.4.2 Lösungsvariante Kamera

Eine andere Variante ist, auf weitere Sensoren zu verzichten und die Detektion von Objekten, Distanzen und Hindernissen alle mit der gleichen Kamera zu ermöglichen. Mithilfe der Kamera können auch Tischkanten erkannt werden.

Objekt-Detektion

Es ist bekannt, dass die Objekt-Detektion auf dem Jetson Nano gut funktioniert. Das Objekt darf dafür jedoch nicht zu weit entfernt sein. Die Objekt-Detektion benötigt sehr viel Leistung und lastet den Jetson Nano schon fast komplett aus.

Distanz-Detektion

Die Distanz muss nicht genau gemessen werden. Der Roboter muss nur wissen, wann und in welche Richtung er sich drehen muss, um den Rand des Raumes seitlich abzufahren. Dies ist möglich, indem man ein neues Modell anlernt. Diese Detektion kann verwendet werden, um den seitlichen Abstand zu einer Wand oder zu einem Abgrund, wie zum Beispiel einer Tischkante, zu wahren.

Hindernis-Detektion

Um einen Raum vollständig abfahren zu können, muss es möglich sein, die Wände oder Tischkanten seitlich abzufahren. Es existieren bereits mehrere Implementationen von JetBots, die einer Linie folgen. Das

Abfahren einer Wand oder Tischkante folgt dem gleichen Prinzip und kann mit einem selbst trainierten Modell umgesetzt werden.

Alle Detektionen zusammen

Der Jetson Nano ist also durchaus fähig, jeden einzelnen der oben genannten Detektionen umzusetzen. Das grosse Problem ist jedoch, dass der Jetson Nano bereits mit der Zielobjekt-Detektion fast komplett ausgelastet ist und es langfristig nicht möglich ist, noch zwei weitere Detektionen parallel laufen zu lassen.

Deshalb scheidet diese Lösungsvariante in der Konstellation mit einem Jetson Nano aus.

5.4.3 Entscheid

Um den gesamten Raum abfahren zu können und das Objekt finden zu können, ist die Implementation mit den Ultraschallsensoren am sinnvollsten. Diese Implementation ist zudem ressourcenfreundlich und performant.

Sobald der Roboter das Objekt erkennt und direkt darauf zufahren kann, werden die Ultraschallsensoren nicht mehr benötigt. Dafür wird ab diesem Moment ein schlankes Kollisionsdetektions-Modell verwendet, das, wie auch die Objekt-Detektion, über die Kamera funktioniert. Dies muss so gelöst werden, weil auf dem Weg zum Objekt noch Hindernisse auftreten könnten. Es ist möglich, dass der Roboter zum Beispiel nur mit einem Rad in ein Hindernis fahren würde und ein solches Objekt würde von einem zentrierten Ultraschallsensor nicht erkannt werden.
Mit dem Jetson Nano ist es möglich, die Objekt-Detektion und eine Kollisions-Detektion kurzzeitig parallel laufen zu lassen. Um die Kollisions-Detektion so performant wie möglich zu machen, kann das Modell gleich im Karton-Modell antrainiert werden.

Tabelle 12: Kollisions-Detektionen basierend auf momentaner Situation

Situation	**Verwendete Kollisions-Detektion**
Kein Objekt erkannt	Ultraschallsensoren (vorne & rechts)
Objekt erkannt	Kamera mit performanter Kollisions-Detektion

6. Literaturverzeichnis

[1] «Catastrophic interference,» 22 10 2020. [Online]. Available: https://en.wikipedia.org/wiki/Catastrophic_interference. [Zugriff am 7 12 2020].

[2] NVIDIA, «nvidia,» NVIDIA, 20 10 2020. [Online]. Available: https://www.nvidia.com/de-de/autonomous-machines/embedded-systems/jetbot-ai-robot-kit/. [Zugriff am 20 10 2020].

[3] L. Jackson, «arducam,» ArduCam, 17 3 2020. [Online]. Available: https://www.arducam.com/nvidia-jetson-nano-b01-update-dual-camera/. [Zugriff am 20 10 2020].

[4] eleccelerator, «Eleccelerator,» 9 6 2019. [Online]. Available: http://eleccelerator.com/wiki/index.php?title=Jetson_Nano_Edimax_EW-7611ULB. [Zugriff am 10 11 2020].

[5] EVAN_DOUBLE_U, «learn.sparkfun.com,» SparkFun, 23 2 2018. [Online]. Available: https://learn.sparkfun.com/tutorials/assembly-guide-for-sparkfun-jetbot-ai-kit-v20/4-software-setup-guide-from-nvidia. [Zugriff am 20 10 2020].

[6] LeopardImaging, «leopardimaging.com,» LeopardImaging, 13 5 2019. [Online]. Available: https://www.leopardimaging.com/product/nvidia-jetson-cameras/nvidia_nano_mipi_camera_kits/li-imx219-mipi-ff-nano/li-imx219-mipi-ff-nano-h136/. [Zugriff am 20 10 2020].

[7] AdaFruit, «adafruit.com,» AdaFruit, 3 2 2019. [Online]. Available: https://www.adafruit.com/product/3777. [Zugriff am 20 10 2020].

[8] SparkFun, «distrelec.ch,» SparkFun, 17 5 2019. [Online]. Available: https://www.distrelec.ch/Web/Downloads/_t/ds/KIT-15437_eng_tds.pdf. [Zugriff am 20 10 2020].

[9] kangalow, «jetsonhacks.com,» JetsonHacks, 10 4 2019. [Online]. Available: https://www.jetsonhacks.com/2019/04/10/jetson-nano-use-more-power/. [Zugriff am 20 10 2020].

[10] LeopardImaging, «leopardimaging.com,» LeopardImaging, 18 10 2019. [Online]. Available: https://www.leopardimaging.com/uploads/LI-IMX219-MIPI-FF-NANO_SPEC.pdf. [Zugriff am 20 10 2020].

[11] SparkFun, «cdn.sparkfun.com,» SparkFun, 1 1 2009. [Online]. Available: https://cdn.sparkfun.com/assets/learn_tutorials/3/0/8/SSD1306.pdf. [Zugriff am 20 10 2020].

[12] E. J. Morgan, «pdf1.alldatasheet.com,» SparkFun, 16 11 2014. [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/1132203/ETC2/HC-SR04.html. [Zugriff am 20 10 2020].

[13] Toshiba, «cdn-shop.adafruit.com,» Toshiba, 1 11 2012. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/TB6612FNG_datasheet_en_20121101.pdf. [Zugriff am 20 10 2020].

[14] Waveshare, «waveshare.com,» Waveshare, 31 7 2020. [Online]. Available: https://www.waveshare.com/wiki/JetBot_AI_Kit. [Zugriff am 20 10 2020].

[15] NKON, «eu.nkon.nl,» NKON, 13 5 2010. [Online]. Available: https://eu.nkon.nl/rechargeable/li-ion/18650-size/nitecore-18650-nl1826-2600mah-4a.html. [Zugriff am 20 10 2020].

[16] Waveshare, «waveshare.net,» Waveshaer, 28 2 2020. [Online]. Available: https://www.waveshare.net/w/upload/1/16/JetBot.pdf. [Zugriff am 20 10 2020].

[17] ANPED, «pdf.alldatasheet.com,» ANPED, 6 4 2013. [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/548536/ANPEC/APW7313.html. [Zugriff am 20 10 2020].

[18] Intel, «intel.com,» Intel, 1 1 2016. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/dual-band-wireless-ac-8265-brief.pdf. [Zugriff am 20 10 2020].

[19] molex, «molex.com,» molex, 27 8 2020. [Online]. Available: https://www.molex.com/webdocs/datasheets/pdf/en-us/2042811100_ANTENNAS.pdf. [Zugriff am 20 10 2020].

[20] NVIDIA, «jetbot.org,» NVIDIA, 17 5 2019. [Online]. Available: https://jetbot.org/master/bill_of_materials.html. [Zugriff am 20 10 2020].

[21] NVIDIA, «jetbot.org,» NVIDIA, 16 5 2019. [Online]. Available: https://jetbot.org/master/getting_started.html. [Zugriff am 20 10 2020].

[22] amazon, «amazon.com,» INIU, 8 9 2018. [Online]. Available: https://www.amazon.com/gp/product/B07H6LB4J4/. [Zugriff am 2020 10 25].

[23] TP-Link, «tp-link.com,» TP-Link, 1 1 2020. [Online]. Available: https://www.tp-link.com/us/home-networking/usb-adapter/archer-t2u-nano/#overview. [Zugriff am 20 10 2020].

[24] TP-Link, «static.tp-link.com,» TP-Link, 1 1 2019. [Online]. Available: https://static.tp-link.com/2020/202010/20201019/Archer%20T2U%20Plus(EU&US)1.0%20Datasheet.pdf . [Zugriff am 10 20 2020].

[25] A. Batista, «gist.github.com,» 1 1 2019. [Online]. Available: https://gist.github.com/allanbatista/708db0a17f03ccfb9b70275eebdb2f5b. [Zugriff am 20 10 2020].

[26] Amazon, «amazon.com,» Waveshare, 17 6 2019. [Online]. Available: https://www.amazon.com/dp/B07T43K7LC/ref=cm_sw_su_dp. [Zugriff am 20 10 2020].

[27] Amazon, «amazon.com,» Raspberry Pi, 25 4 2016. [Online]. Available: https://www.amazon.com/Raspberry-Pi-Camera-Module-Megapixel/dp/B01ER2SKFS/. [Zugriff am 20 10 2020].

[28] Amazon, «amazon.com,» MakerFocus, 17 9 2018. [Online]. Available: https://www.amazon.com/dp/B07HF81BVL/ref=cm_sw_su_dp?th=1. [Zugriff am 20 10 2020].

[29] Amazon, «amazon.com,» Intel, 14 1 2017. [Online]. Available: https://www.amazon.com/Intel-Dual-Band-Wireless-Ac-8265/dp/B01MZA1AB2/. [Zugriff am 20 10 2020].

[30] Arrow, «arrow.com,» Arrow, 1 1 2020. [Online]. Available: https://www.arrow.com/en/products/2042811100/molex. [Zugriff am 10 20 2020].

[31] Siliconhighway, «siliconhighwaydirect.co.uk,» Siliconhighway, 1 1 2020. [Online]. Available: https://www.siliconhighwaydirect.co.uk/product-p/jetbot-kit.htm. [Zugriff am 20 10 2020].

[32] coco, «Coco,» 2020. [Online]. Available: https://cocodataset.org/. [Zugriff am 13 11 2020].

[33] NVIDIA, «nvidia,» 24 9 2020. [Online]. Available: https://ngc.nvidia.com/catalog/models/nvidia:tlt_pretrained_object_detection. [Zugriff am 7 12 2020].

[34] D. Parthasarathy, «Athelas,» 22 4 2017. [Online]. Available: https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4. [Zugriff am 9 11 2020].

[35] S. Ananth, «towards data science,» 9 8 2019. [Online]. Available: https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46. [Zugriff am 9 11 2020].

[36] J. Brownlee, «Machinelearningmastery,» 22 5 2019. [Online]. Available: https://machinelearningmastery.com/object-recognition-with-deep-learning/.

[37] D. Mwiti, «Heartbeat,» 18 7 2019. [Online]. Available: https://heartbeat.fritz.ai/a-2019-guide-to-object-detection-9509987954c3. [Zugriff am 17 11 2020].

[38] S. Ren, K. He, R. Girshick und J. Su, «arxiv,» 6 1 2016. [Online]. Available: https://arxiv.org/pdf/1506.01497.pdf. [Zugriff am 20 11 2020].

[39] J. Świeżewski, «Appsilon,» 22 5 2020. [Online]. Available: https://appsilon.com/object-detection-yolo-algorithm/. [Zugriff am 10 11 2020].

[40] Appsilon, «Appsilon,» 17 9 2020. [Online]. Available: https://appsilon.com/pp-yolo-object-detection/. [Zugriff am 10 11 2020].

[41] M. Rajput, «Medium,» 13 6 2020. [Online]. Available: https://medium.com/towards-artificial-intelligence/yolo-v5-is-here-custom-object-detection-tutorial-with-yolo-v5-12666ee1774e.

[42] J. Hui, «Medium,» 18 3 2018. [Online]. Available: https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088. [Zugriff am 16 11 2020].

[43] J. Redmon und A. Farhadi, 2017. [Online]. Available: https://pjreddie.com/media/files/papers/YOLOv3.pdf. [Zugriff am 19 11 2020].

[44] O. Elisha, «Heartbeat,» 17 9 2020. [Online]. Available: https://heartbeat.fritz.ai/real-time-object-detection-using-ssd-mobilenet-v2-on-video-streams-3bfc1577399c. [Zugriff am 13 11 2020].

[45] S.-H. Tsang, «towards data science,» 2018 11 2018. [Online]. Available: https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11. [Zugriff am 13 11 2020].

[46] MathWorks, «MathWorks,» 2020. [Online]. Available: https://ch.mathworks.com/help/vision/ug/getting-started-with-ssd.html. [Zugriff am 16 11 2020].

[47] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu und A. C. Berg, «arxiv,» 29 12 2016. [Online]. Available: https://arxiv.org/pdf/1512.02325.pdf. [Zugriff am 20 11 2020].

[48] R. Khandelwal, «towards data science,» 6 1 2020. [Online]. Available: https://towardsdatascience.com/evaluating-performance-of-an-object-detection-model-137a349c517b. [Zugriff am 17 11 2020].

[49] J. Hui, «Medium,» 7 3 2018. [Online]. Available: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173. [Zugriff am 19 11 20].

[50] Saracus, «Saracus,» 14 11 2018. [Online]. Available: https://www.saracus.com/blog/performance-metriken-klassifikation-2-2/. [Zugriff am 19 11 2020].

[51] S. Chauhan, «Blogspot,» 05 10 2020. [Online]. Available: https://themechcom.blogspot.com/2020/10/understanding-mean-average-precision.html. [Zugriff am 20 11 2020].

[52] W. Koehrsen, «towards data science,» 3 3 2018. [Online]. Available: https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c.

[53] T.-Y. Lin, P. Goyal, R. Girshick, K. He und P. Dollar, «arxiv,» 7 2 2018. [Online]. Available: https://arxiv.org/pdf/1708.02002.pdf. [Zugriff am 23 11 2020].

[54] coco, «cocodataset,» 2020. [Online]. Available: https://cocodataset.org/#detection-eval. [Zugriff am 24 11 2020].

[55] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto und H. Adam, «arxiv,» 17 4 2017. [Online]. Available: https://arxiv.org/pdf/1704.04861.pdf.

[56] J. Hui, «Medium,» 28 3 2018. [Online]. Available: https://jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359. [Zugriff am 20 11 2020].

[57] Sanchez, «iop,» 2019. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/844/1/012024/pdf. [Zugriff am 20 11 2020].

[58] D. Franklin, «Nvidia,» 31 1 2020. [Online]. Available: https://news.developer.nvidia.com/realtime-object-detection-in-10-lines-of-python-on-jetson-nano/. [Zugriff am 24 11 2020].

7. Anhang

7.1 Aufgabenstellung

OST
Ostschweizer
Fachhochschule

Aufgabenstellung

JetBot - Autonomes und fortlaufendes maschinelles Lernen

Trotz grossen Fortschritten in weiten Bereichen der künstlichen Intelligenz sind im Bereich der angewandten künstlichen Intelligenz Themen der Autonomie und Adaption weiterhin nur wenig thematisiert. Ein allgemeiner Einsatz von Robotern für alltägliche Aufgaben ist allerdings erst möglich, wenn diese in der Lage sind, sich autonom und selbständig an Situationen anzupassen und neue Inhalte zu erlernen. Ein Pflegeroboter zum Beispiel muss sich, wenn möglich, selbständig an die Umgebung anpassen und Verhaltensweisen einzelner Patienten erlernen und interpretieren können. Es stellt sich also die Frage, wie einfache bis komplexe sensorische kognitive Leistungen auf vorhandenen Roboter-Plattformen implementiert werden können.

In dieser Arbeit sollen die Grenzen des Jetson Nano in Bezug auf autonomes Lernen als Grundlage für eine weiterführende Arbeit evaluiert werden. Es soll aufgezeigt werden, wo die Grenzen der Möglichkeiten für einen Rescue-Roboter liegen. In diesem Bezug sollen Roboter wie beispielsweise der JetBot und andere Modelle evaluiert werden. Des Weiteren soll auf die Objekterkennung und weitere benötigte Machine Learning Module vertieft eingegangen werden, um eine optimale Grundlage für eine mögliche Umsetzung zu erreichen.

Rapperswil 2020-12-10,

Prof. Dr.-Ing. Andreas Rinkel	
Marc Sommerhalder	

Namen Unterschriften

Aufgabenstellung Seite 1

Abbildung 12: Anhang, Aufgabenstellung

7.2 Eigenständigkeitserklärung

OST
Ostschweizer
Fachhochschule

Eigenständigkeitserklärung

Erklärung

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.
- dass wir keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt haben.

Rapperswil 2020-12-10,

Aaron Studer	
Benjamin Peter	
Yanick Rek	

Namen Unterschriften

Eigenständigkeitserklärung Seite 1

Abbildung 13: Anhang, Eigenständigkeitserklärung

7.3 Einverständniserklärung

OST
Ostschweizer
Fachhochschule

Einverständniserklärung Publikation auf eprints.hsr.ch

☑ SA
O BA

Titel der Arbeit:	JetBot - Autonomes und fortlaufendes maschinelles Lernen
Team:	Aaron Studer, Benjamin Peter, Yanick Rek
Betreuer:	Prof. Dr.-Ing. Andreas Rinkel, Marc Sommerhalder

Wir sind mit der Publikation unserer Arbeit auf eprints.hsr.ch einverstanden, sofern für diese Arbeit keine Geheimhaltungsvereinbarung unterzeichnet wurde.
Nach Bekanntgabe der Note haben wir die Möglichkeit innert 14 Tagen Einsprache zu erheben und das Einverständnis zur Publikation der Arbeit auf eprints.hsr.ch zurückzuziehen. In diesem Falle wird nur der Abstract publiziert.

Rapperswil 2020-12-10,

Aaron Studer	
Benjamin Peter	
Yanick Rek	

Namen Unterschriften

Einverständniserklärung Publikation auf eprints.hsr.ch Seite 1

Abbildung 14: Anhang, Einverständniserklärung

7.4 Vereinbarung über Urheber- und Nutzungsrechte

OST Ostschweizer Fachhochschule

Vereinbarung über Urheber- und Nutzungsrechte

1. Vereinbarung

1.1.1 Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der SA «JetBot - Autonomes und fortlaufendes maschinelles Lernen» von Aaron Studer, Benjamin Peter und Yanick Rek unter der Betreuung von Prof. Dr.-Ing. Andreas Rinkel und Marc Sommerhalder geregelt.

1.1.2 Urheberrecht

Die Urheberrechte stehen der Studenten zu.

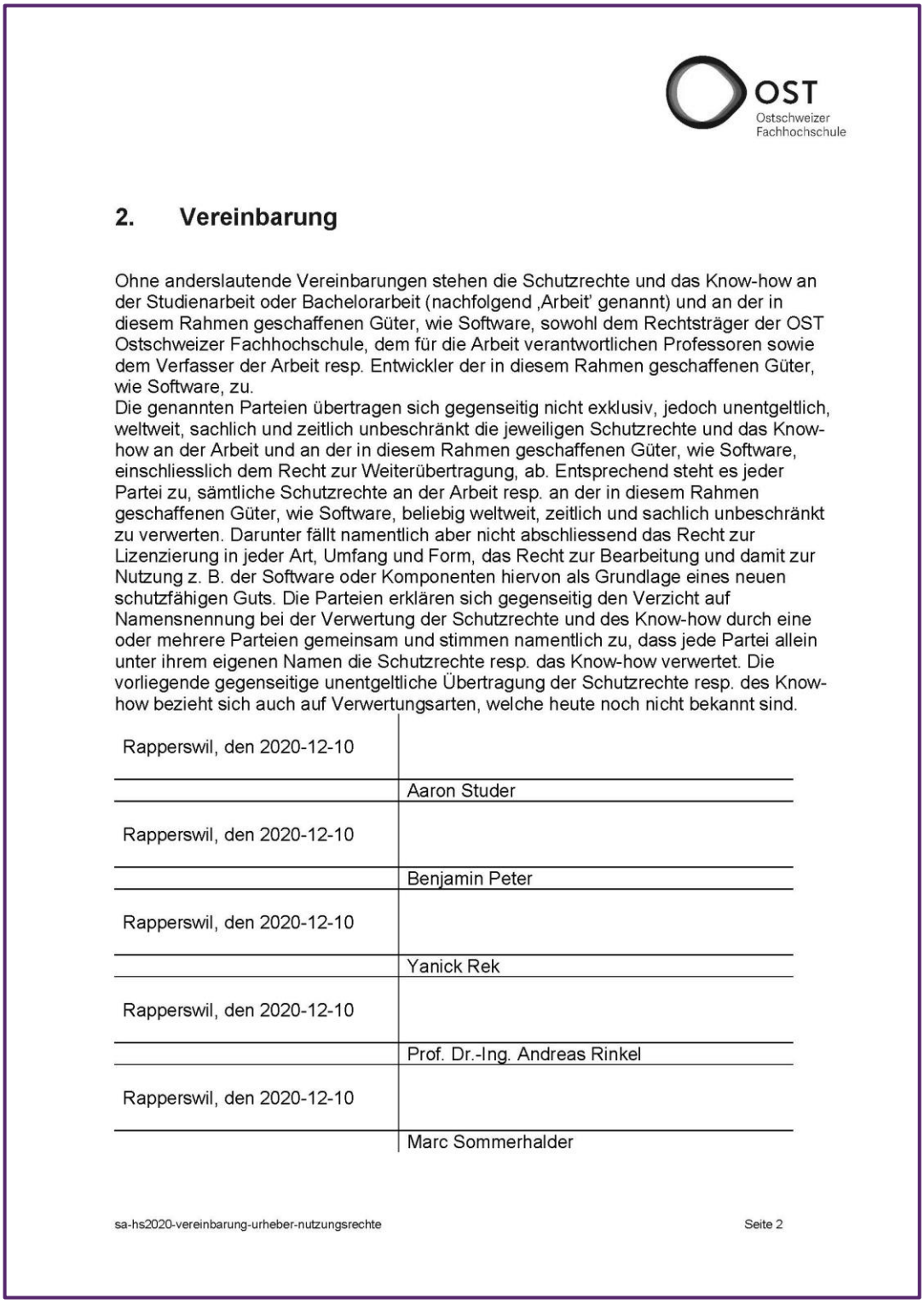
1.1.3 Verwendung

Die Ergebnisse der Arbeit dürfen sowohl von der Studentin / dem Studenten, von der OST wie vom «INS, Institute for Networked Solutions» nach Abschluss der Arbeit verwendet und weiterentwickelt werden

Rapperswil, den 2020-12-10	
	Aaron Studer
Rapperswil, den 2020-12-10	
	Benjamin Peter
Rapperswil, den 2020-12-10	
	Yanick Rek
Rapperswil, den 2020-12-10	
	Prof. Dr.-Ing. Andreas Rinkel
Rapperswil, den 2020-12-10	
	Marc Sommerhalder

sa-hs2020-vereinbarung-urheber-nutzungsrechte Seite 1

Abbildung 15: Anhang, Vereinbarung über Urheber - und Nutzungsrechte Seite 1



OST
Ostschweizer
Fachhochschule

2. Vereinbarung

Ohne anderslautende Vereinbarungen stehen die Schutzrechte und das Know-how an der Studienarbeit oder Bachelorarbeit (nachfolgend ‚Arbeit' genannt) und an der in diesem Rahmen geschaffenen Güter, wie Software, sowohl dem Rechtsträger der OST Ostschweizer Fachhochschule, dem für die Arbeit verantwortlichen Professoren sowie dem Verfasser der Arbeit resp. Entwickler der in diesem Rahmen geschaffenen Güter, wie Software, zu.
Die genannten Parteien übertragen sich gegenseitig nicht exklusiv, jedoch unentgeltlich, weltweit, sachlich und zeitlich unbeschränkt die jeweiligen Schutzrechte und das Know-how an der Arbeit und an der in diesem Rahmen geschaffenen Güter, wie Software, einschliesslich dem Recht zur Weiterübertragung, ab. Entsprechend steht es jeder Partei zu, sämtliche Schutzrechte an der Arbeit resp. an der in diesem Rahmen geschaffenen Güter, wie Software, beliebig weltweit, zeitlich und sachlich unbeschränkt zu verwerten. Darunter fällt namentlich aber nicht abschliessend das Recht zur Lizenzierung in jeder Art, Umfang und Form, das Recht zur Bearbeitung und damit zur Nutzung z. B. der Software oder Komponenten hiervon als Grundlage eines neuen schutzfähigen Guts. Die Parteien erklären sich gegenseitig den Verzicht auf Namensnennung bei der Verwertung der Schutzrechte und des Know-how durch eine oder mehrere Parteien gemeinsam und stimmen namentlich zu, dass jede Partei allein unter ihrem eigenen Namen die Schutzrechte resp. das Know-how verwertet. Die vorliegende gegenseitige unentgeltliche Übertragung der Schutzrechte resp. des Know-how bezieht sich auch auf Verwertungsarten, welche heute noch nicht bekannt sind.

Rapperswil, den 2020-12-10	
	Aaron Studer
Rapperswil, den 2020-12-10	
	Benjamin Peter
Rapperswil, den 2020-12-10	
	Yanick Rek
Rapperswil, den 2020-12-10	
	Prof. Dr.-Ing. Andreas Rinkel
Rapperswil, den 2020-12-10	
	Marc Sommerhalder

sa-hs2020-vereinbarung-urheber-nutzungsrechte Seite 2

Abbildung 16: Anhang, Vereinbarung über Urheber - und Nutzungsrechte Seite 2

ins
INSTITUTE FOR
NETWORKED SOLUTIONS

OST
Ostschweizer
Fachhochschule

Impressum

Datum
18. Dezember 2020

Aaron Studer, Benjamin Peter, Yanick Rek

OST – Ostschweizer Fachhochschule
Informatik

Oberseestrasse 10, Postfach
8640 Rapperswil, Schweiz

ost.ch