SCOUT

Ein Projekt von Jacob Utermöhlen und Narindar Singh

Kurzfassung

SCOUT ist ein selbstentwickelter Roboterhund, der in der Lage ist, sich über Fernsteuerung zu bewegen sowie mit Hilfe von Sensoren sein Umfeld wahrzunehmen und Hindernisse zu erkennen und seine Bewegung entsprechend anzupassen.

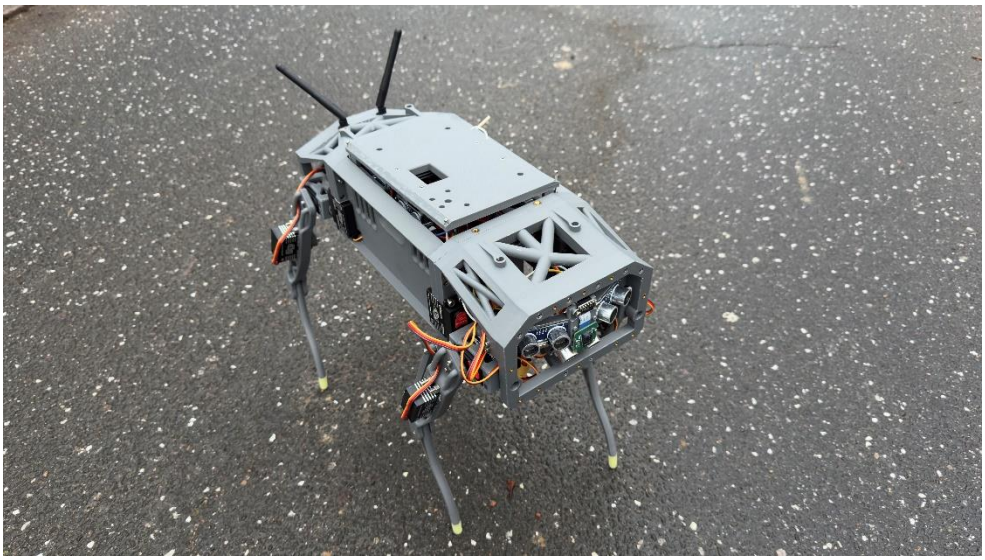
Perspektivisch soll SCOUT so weiterentwickelt werden, dass er sehbehinderten Menschen beim Navigieren helfen und sie bei alltäglichen Situationen unterstützen kann.

Alle dafür notwendigen Sensoren und Hardware sind im Design von SCOUT bereits angelegt.

Wir haben den Roboter selbst entworfen, gebaut und alle Steuerungsprogramme selbst programmiert

Zentrale Themen unseres Projektes sind:

- Bewegung des Hundes
- mit Kameras und Sensoren die Umgebung wahrnehmen
- Veränderung in der Umgebung wahrnehmen und automatisch auf diese reagieren
- Eigenständiges Design des Roboterhundes, Bau aus selbst 3D gedruckten Bauteilen sowie handelsübliche Einzelkomponenten (z.B. Prozessor), eigenständig programmierte Software für Steuerung und Betrieb
- Design und Ausstattung die perspektivisch auch autonome Navigation ermöglichen



Inhalt

1. Einführung

1.1 Ideenfindung

Wir kamen auf unsere Idee einen Roboterhund zu bauen bei der Teilnahme and der Jugend-Forscht AG am Gymnasium Hochrad. Ein Teilnehmer unseres Teams hat bereits Erfahrung mit Robotik, so entschieden wir uns einen Roboter zu bauen.

Wir bekamen mit wie Nachbarn von uns, sie sind blind, mitten auf der Straße gingen, da es der einfachste Weg für sie ist einem bestimmten Weg zu folgen - nicht gerade sicher.

So beschlossen wir einen Roboter zu bauen, der so aufgebaut ist, dass er in zukünftiger Weiterentwicklung das Navigieren sehbehinderter Personen erleichtern soll.

1.2 Ideen zur Umsetzung

Um die Kosten des Projektes möglichst gering zu halten, entschieden wir weit verbreitete Bauteile zu verwenden und meiste nicht-elektronischen Bauteile mit dem 3D-Drucker selbst herzustellen.

Danach muss die Bewegung des Roboters geplant werden und alles mit Software zu einem kompletten Roboter zusammengefügt werden.

2. Vorgang, Methode, Material

2.1 Design des Roboters

Für das Model und Design des Roboters entschieden wir alle Teile von Grund auf selbst zu entwickeln, um die gesamte Kontrolle über das Projekt zu haben.

Wir fingen an mit Stift und Papier grobe Skizzen des Hundes zu erstellen, um Ideen zum Design zu finden. Die nächsten zwei Wochen verbrachten wir damit, mit dem CAD (Computer-Aided Design) Programm Solidworks ein exaktes 3D Modell des Roboters zu erstellen.

Davor erstellten wir Kriterien, die die Mechanik erfüllen muss.

- Alle Teile müssen so geformt sein, dass sie einfach mit einem 3D-Drucker von zu Hause hergestellt werden können, um die Kosten des Roboters niedrig zu halten.
- Der Roboter sollte genug Platz für große Batterien bieten, um eine möglichst lange Batterielaufzeit zu ermöglichen.

- Jedes Teil sollte einfach zu erreichen sein, um spätere Aufrüstungen einfach zu machen.
- Platz für alle möglichen Sensoren und Kameras muss gegeben sein.

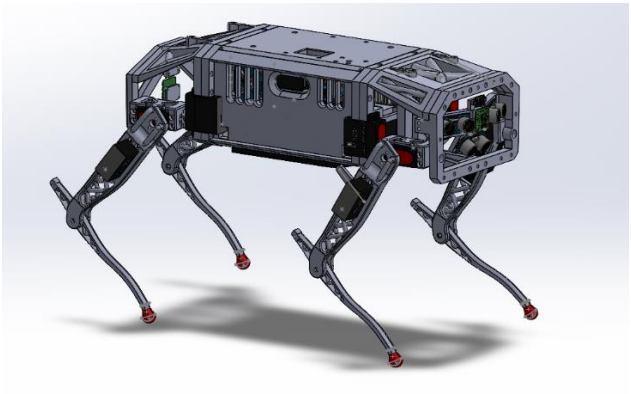


Abbildung 1, 3D Modell von Scout, Solidworks 2025

2.1.2 Die Beine

Das wichtige eines navigierenden Roboters ist die Fortbewegung, in diesem Fall also die Beine.

Sie müssen robust sein, um Stürze und kraftvolle Fremdeinwirkungen standzuhalten. Beim 3D-Drucken ist es wichtig auf die Druckorientierung zu achten. Fürs Drucken wird jedes Teil, welche aus einem STL-Mesh bestehen, von einem Programm in viele Schichten „zerschnitten“, die dann von 3D-Drucker übereinander gedruckt werden.

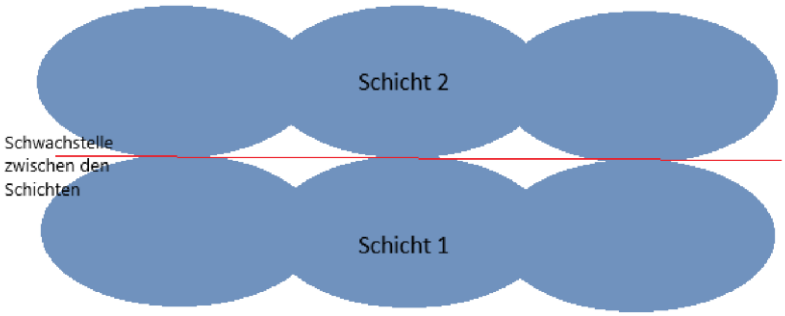


Abbildung 2, Druckschichten und Schwachpunkte, Jacob U.

Die Übergänge zwischen den Druckschichten sind Schwachpunkte in dem jeweiligen Teil. Wir mussten also darauf achten, dass alle Teile der Beine vom Drucker horizontal und nicht vertikal gedruckt werden, um das Bein widerstandsfähiger zu machen.

Jedes Bein wird von drei Servomotoren angetrieben, diese werden häufig beim Modellbau benutzt. Wir entschieden uns Servos mit einem Drehmoment von 40kg/cm zu nutzen, um das Gewicht des Hundes halten zu können.

Jeder Motor wiegt um die 70g. Für den Unterschenkel eines Beines entschieden wir uns dazu den Motor nicht an der Rotationsachse des Kniegelenkes zu montieren, sondern die Masse des Motors möglichst nahe an das Obere Gelenk zu befestigen, um das

Trägheitsmoment des Beines zu verringern. So kann sich das Bein schneller und dabei mit weniger Drehmoment der Motoren bewegen..

Wir entschieden uns den Beinen eine Krümmung zu geben, damit sie elastischer sind und Stürze usw. besser Abfangen können

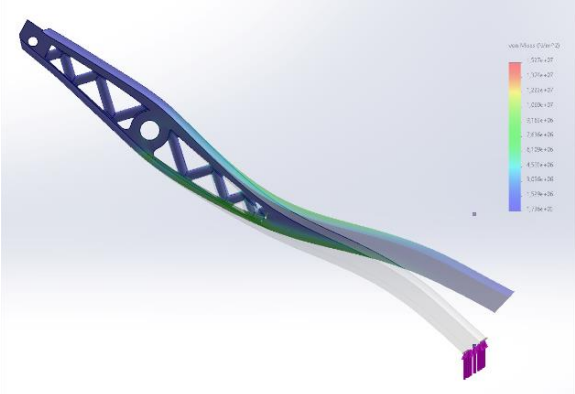


Abbildung 3, Stresstest unter 10N Belastung (gebogenes Bein), Solidworks Simulation 2025

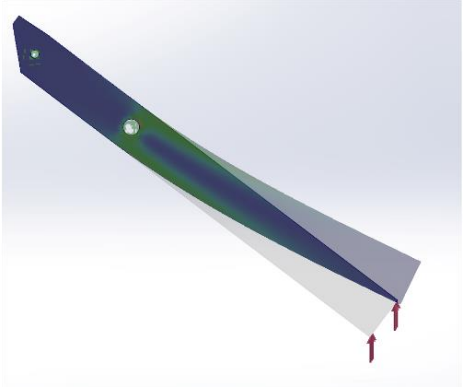


Abbildung 4, Stresstest unter 10N Belastung (gerades Bein), Solidworks Simulation 2025

Am Anfang haben wir die Füße auch mit dem 3D-Drucker gedruckt. Für den ganzen Roboter wurde ausschließlich PLA benutzt, weil es sehr fest und günstig ist. Noch dazu ist es sehr einfach zu drucken, hat allerdings auch einen Nachteil. Wir haben festgestellt, dass die harten Plastikenden der Beine keinerlei Haftung hatten, und auf allen Oberflächen sind. Wenn sich die Beine bewegten, die Füße aber nur über den Untergrund rutschten, bewegte sich der Hund nicht fort.

Die Lösung: Wir mussten die Füße aus einem weichen Gummi anfertigen. Unser erster Prototyp waren halbierte Flummis, die mit Sekundenkleber and die Beine geklebt wurden.

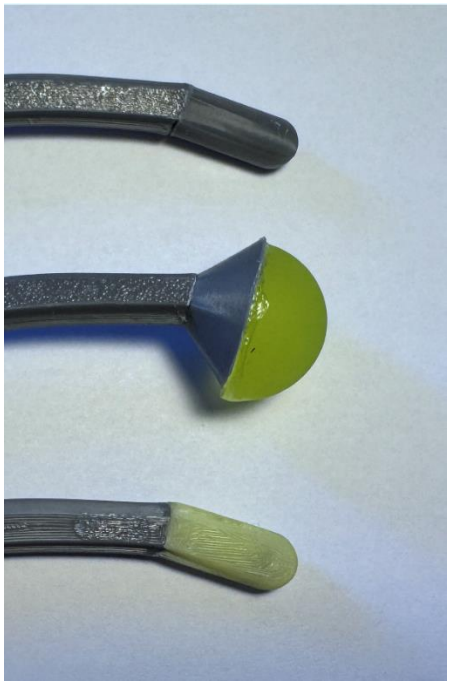


Abbildung 5, Prototypen der Füße, Jacob U.

Nicht nur sieht das nicht gut aus, war aber auch nicht funktional. Die Flummis in der Hälfte zu zerschneiden war schwierig und zeitaufwendig. Außerdem hatten sie nicht genug Haftung und fielen zu einfach von den Beinen ab.

Unsere finale Idee war, dass wir die Füße aus Silikon selbst herzustellen. Mit dem 3D-Drucker haben wir Schalen gedruckt, in die wir flüssiges Silikon gießen konnte, und so die Beine mit einer Gummischicht umschließen konnten. Dies hat sehr gut funktioniert. Die Beine haben am Ende einen griffigen Gummiteil, der der Form des Beines folgt und

2.2 Elektronik und Hardware

Für die Steuerung des Roboters brauchen wir einen Computer an Board. Weil wir in einer weiteren Projektphase planen, dass der Roboter ein Kamerasystem bekommt und eventuell KI oder Neuronale Netzwerke nutzt, benötigen wir einen Leistungsfähigen und kleinen Computer. Am Anfang nutzten wir einen Raspberry Pi 4 b, weil der günstig und einfach zu benutzen ist. Bei Test mit einem selbstgeschriebenem Programm zur Objekterkennung stellten wir fest, dass diese nur 3 bis 4 Bilder pro Sekunde verarbeiten kann.

Deshalb stiegen wir auf einen NVIDIA Jetson Orin Nano um, ein sehr ähnlicher Computer mit sehr viel mehr Leistung

Dankenswerterweise wurde uns der Jetson kostenlos von MyBotShop.de zur Verfügung gestellt, nachdem wir dem Unternehmen unser Projekt erläutert hatten.

Jedes der vier Beine wird mit jeweils drei Servos bewegt. Jeder der Servos hat ein Drehmoment von 40kg/cm bei einer Spannung von 7,4V. Mit den drei Motoren hat jedes Bein drei Freiheitsgrade.

Der Roboter besitzt fünf HC-SR04 Ultraschallsensoren, die Distanzen messen. Außerdem sind zwei Laser Distanzsensoren verbaut, jeweils einer vorne und hinten. Mit ihnen können z.B. Wände erkannt werden. Eine Inertiale Messeinheit (ICM-20948) misst Beschleunigung in drei Richtungen und Drehraten in allen drei Achsen. Mit Hilfe eines Kalman-Filters werden Rotationswinkel ermittelt.

Alle Sensoren sind mit einem ESP-32 verbunden. Ein ESP-32 ist ein sehr kostengünstiger Mikrokontroller mit dem Sensoren sehr einfach auszulesen sind.

Der ESP32 kommuniziert mit dem UART-Serial Protokoll mit dem Jetson und sendet die Sensordaten.

2.3 Kinematik

Um Gehbewegungen mit den Beinen auszuführen, muss sich der Fuß jeden Beines frei in einem 3D Koordinatensystem bewegen können. Die Servo-Motoren lassen sich nur mit Winkeln steuern.

Um das Ende des Fußes millimetergenau im 3D-Raum bewegen zu können muss inversive Kinematik benutzt werden.

Gegeben sind drei Werte, X, Y und Z in einem virtuellen Koordinatensystem. Die Werte geben die Position des Fußes relativ zur der des ersten Gelenkes (0|0|0). Anhand dieser

Werte werden alle Winkel der Gelenke berechnet, sodass der Fuß relativ vom Startpunkt X, Y und Z Millimeter weit entfernt ist.

Hierfür wurde einfache Trigonometrie mit Sinus- und Kosinussatz angewendet.

Am Ende erhält man drei Formeln, bei denen wir bei eingesetzten X, Y und Z Werten drei Winkel erhalten. Bewegt man das zugehörige Gelenk zu dem Winkel, befindet sich der Fuß im 3D-Raum relativ zum Startpunkt X, Y und Z Millimeter entfernt.

Anschließend wurde jedes Bein mithilfe der Python-Library Matplotlib visualisiert.

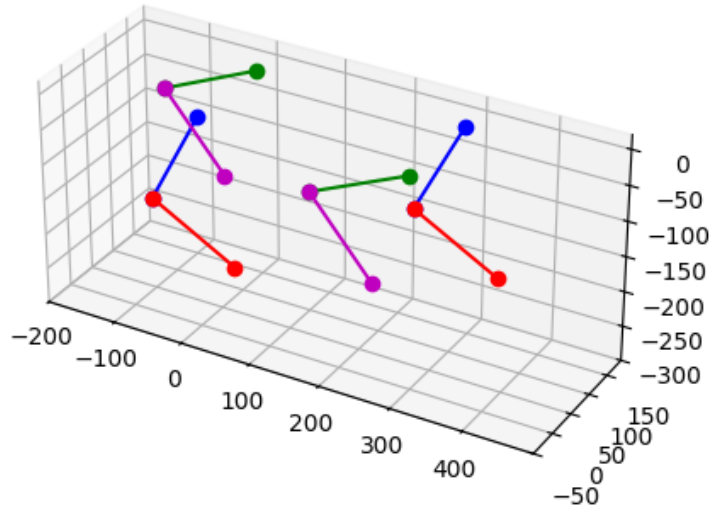


Abbildung 6, Visualisierung der Beine, Python Matplotlib

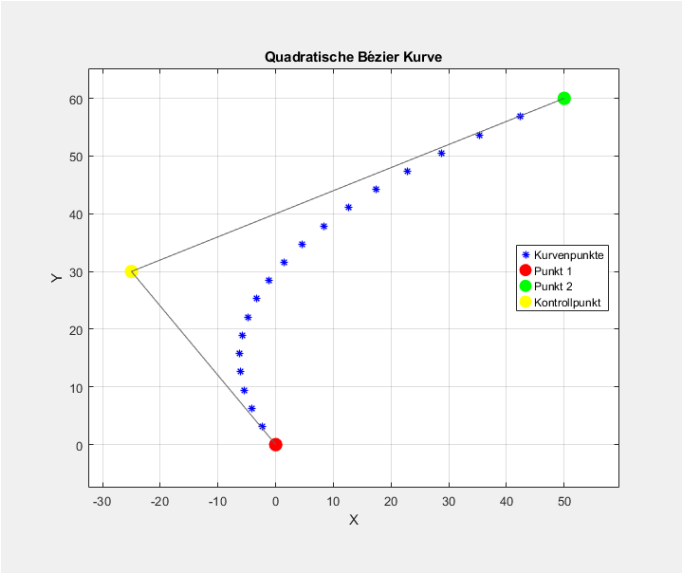
Wir kennen den Winkel von jedem Gelenk. Mit weiterer Trigonometrie können wir Start- und Endpunkt jedes Beinsegments im 3D Raum berechnen. Mit Matplotlib konnten wir dann zwischen den beiden Punkten eine Linie ziehen und darstellen.

Die Visualisierung hat uns bei der Erstellung der Gehbewegung und der Synchronisation der Beine geholfen.

2.4 Gehbewegung

Mit der Kinematik ausgerechnet, kann sich jeder Fuß nun präzise im 3D-Raum bewegen. Damit sich der Roboter nach vorne bewegen kann, mussten wir zuerst eine Gehbewegung erstellen. Unsere erste Idee war es, jedem Bein vier Punkte in Form eines Trapezes zu geben. Das funktioniert zwar, allerdings sind die Bewegungen sehr zackig und abrupt. Vor allem gegen Ende der Bodenberührung muss das sich Bein schnell in die komplett andere Richtung bewegen. Schon nach wenigen Durchläufen würden die Motoren aufgrund der schnellen Geschwindigkeits- und Richtungsänderung Schaden nehmen. Wir haben nach Funktionen gesucht, die eine „weiche“ Kurve erstellen können.

Wir entschieden uns den Anfang und das Ende der Bodenberührung mit einer „Quadratischen Bézier-Kurve“ zu vollenden. Eine Quadratische Bézier-Kurve erstellt zwischen zwei Punkten P1 und P2 t-viele Punkte, deren Position von einem Kontrollpunkt gelenkt wird.



Eine zweite Kurve wird gespiegelt und flache Punkte hinzugefügt. Die Punkte auf Höhe 0 sind die, wo der Fuß jeweils den Boden berührt und den Roboter nach vorne „drückt“. Die Distanz der gespiegelten Bezier-Kurven ist die Schrittweite des Hundes und ist variabel.

Abbildung 7, Quadratische Bézierkurve, MATLAB R2024b

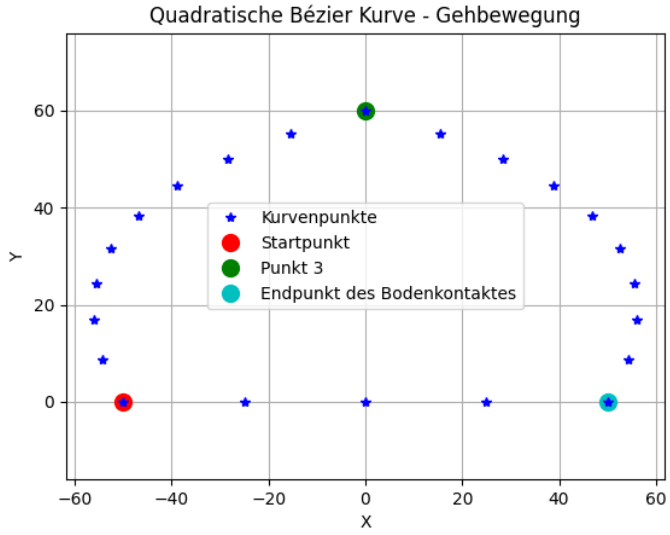


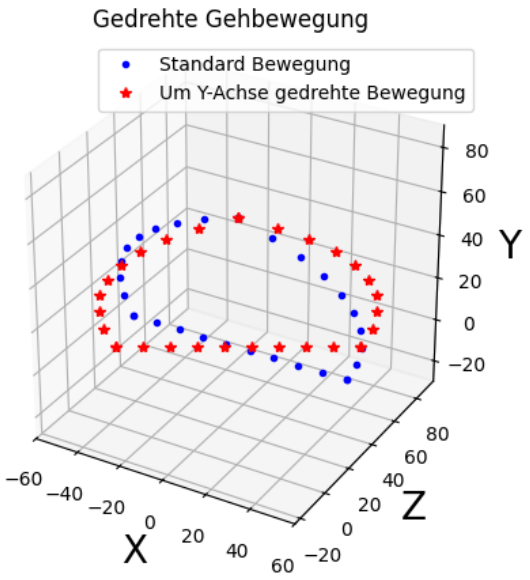
Abbildung 8, Gehbewegung mit 100mm Bodenkontakt, Python Matplotlib

2.4.2 Kurven

Damit der Roboter Kurven gehen kann, muss die Gehbewegung bei den Vorderbeinen gedreht werden. Jeder Punkt in der Gehbewegung befindet sich in einem 3D-Koordinatensystem. Wenn der Roboter eine Kurve gehen soll, berechnen wir für jeden Punkt in der Gehbewegung neue Koordinaten, die um den Winkel α um die Y-Achse gedreht sind. Dabei erhalten wir anhand von simplem Kalkulationen und den Ursprünglichen X und Z Werten neue X und neu Z Werte:

$$X_{neu} = \cos\alpha \times X_{alt}$$

$$Z_{neu} = \sin\alpha \times X_{alt} + Z_{alt}$$



Nachdem wir die Punkte nun drehen konnten, musste jeder Punkt für jedes Bein neu berechnet werden. Dank unseres Jetson Nanos (und weil dies einfache Rechnungen sind) geschieht dies in Echtzeit.

Wenn wir dem Roboter den Befehl erteilen, er solle nach links gehen, dann werden die Wegpunkte für die vorderen Beine neu berechnet.

Abbildung 9, Um Y-Achse gedrehte Gehbewegung, Python Matplotlib

Aus der Perspektive von oben auf den Bodenkontakt jedes Beins, sieht das Kurvenlaufen ungefähr so aus. Der Winkel der Vorderbeine variiert je nach Winkel α.

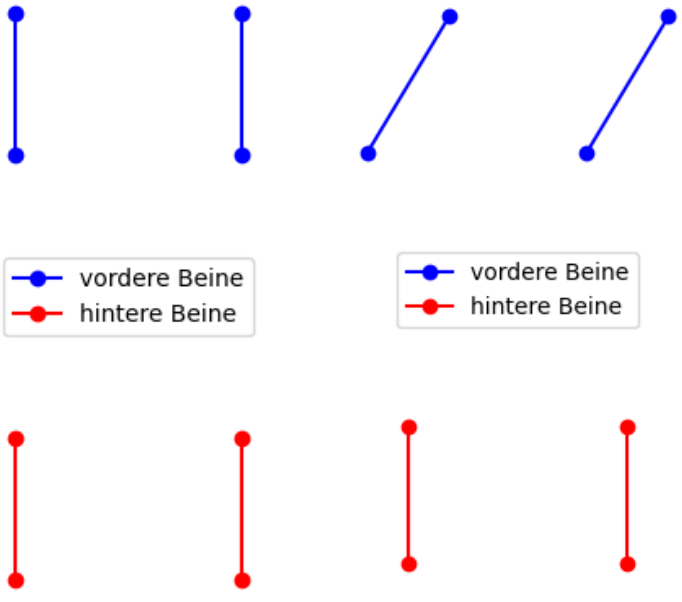


Abbildung 11, Bodenkontakt beim Geradegehen, Python Matplotlib

Abbildung 10, Bodenkontakt beim Kurvengehen, Python Matplotlib

Beim Gehen ist es wichtig, dass immer zwei, gegenüberliegende Beine Bodenkontakt haben, damit der Roboter stabil ist.

Die Bodenkontakte sind so getaktet:

- Vorne links mit hinten rechts
- Vorne rechts mit hinten links

2.5 Kontrolle und Software

Die Softwareentwicklung ist der aufwendigste Teil unseres Projektes. Wir entschieden uns sie Schritt für Schritt anzugehen. Das Endziel unseres Projektes ist die autonome Navigation, komplett isoliert von externen Computern.

Um dieses Ziel schneller zu erreichen, ließen wir den autonomen Aspekt unseres Projektes erstmal aus und konzentrierten uns auf die wesentlichen Aspekte der Steuerung von einem Benutzter mithilfe eines GUIs (Graphical User Interface).

Die gesamte Software lässt sich aktuell in drei Bereiche aufteilen:

1. Code auf dem Mikrocontroller (ESP32)
2. Code auf dem Jetson Orin Nano (unserem On-Board Computer)
3. Grafische Benutzeroberfläche.

2.5.2 Code des Mikrokontrollers

Unser Roboter besitzt viele Sensoren. Diese müssen ausgelesen und deren Daten teilweise verarbeitet werden. Wir nutzen hierfür den ESP32, für den bereits viele Code-Libraries existieren, die die Auslesung einfacher machen.

Wir benutzen eine Inertiale Messeinheit die Beschleunigung misst. Mithilfe eines Kalman-Filters werden Gyroskop- und Beschleunigungsdaten kombiniert und zukünftige Daten vorhergesagt. Mithilfe dieser Mathematischen Methode erhalten wir präzise Neigungswinkel in X- und Y-Achse.

Der Code beinhaltet eine Initialisierungsphase, in der alle Sensoren initialisiert und geprüft werden. Darauf folgt eine nicht endende Schleife. In der Schleife werden alle Sensoren ausgelesen, formatiert und in einen String gepackt. Der String (Zeichenfolge) wird mit Hilfe des UART-Protokolls an den Jetson Orin Nano übertragen. Der Vorgang wiederholt sich so lange, bis der Roboter ausgeschaltet ist.

2.5.3 Firmware des Jetson

Die Firmware des Jetson haben wir in Python geschrieben. Sie ist in verschiedene Teile aufgebaut, um das Debuggen und Wartung zu vereinfachen.

Der Roboter ist aktuell noch nicht autonom und kommuniziert noch mit einer Benutzerfläche auf einem externen Laptop. Wir benutzen Threading in unserem Code. Kurz gesagt ist der Code in Threads aufgeteilt. Threads sind Codeabschnitte, die parallel und asynchron auf jeweils einem CPU-Kern laufen.

Am Anfang startet der Jetson einen WLAN-Hotspot, mit dem sich der Computer mit der Benutzeroberfläche verbindet. Wir hosten gleichzeitig auch zwei TCP-Server in dem vom

Jetson erstellten Netzwerk. Die Beiden TCP Verbindungen erlauben der Benutzeroberfläche Daten mit hoher Bandbreite zu empfangen und zu senden. Danach wird dauerhaft nach eintreffenden Befehlen gesucht. Trifft ein Befehl ein, wird dieser individuell verarbeitet.

Bei dem Befehl Kurven zu gehen, werden alle Wegpunkte jeder Beinbewegung neu berechnet. Je nach Einstellung besteht ein Gehzyklus aus 20 bis 50 Wegpunkten. Würde man dem Bein sagen, jedem dieser Wegpunkte zu folgen, wäre die Bewegung sehr „eckig“.

Mit Hilfe eines Interpolationsalgorithmus werden 10 bis 100 Punkte nach linearem Schema berechnet und zwischen die ursprünglichen Punkte gesetzt.

2.5.4 Grafische Benutzeroberfläche

Um die Steuerung des Roboters zu vereinfachen, erstellten wir eine grafische Benutzeroberfläche (GUI). Das GUI dient als visuelle Brücke zwischen Mensch und Maschine und ermöglicht eine unkomplizierte Steuerung und Überwachung des Roboters. Durch die übersichtliche Darstellung von Informationen, wie beispielsweise dem aktuellen Status des Roboters, dem Batteriestand oder den empfangenen Befehlen. Es erlaubt dem Nutzer, den Roboter ohne komplexe Programmierkenntnisse zu bedienen und fördert so eine effektive und benutzerfreundliche Mensch-Maschine-Kommunikation.

Die Benutzeroberfläche ist in C# mit Hilfe der Windows Presentation Foundation programmiert. Aktuell werden keine komplexen Berechnungen in dem Programm selbst ausgeführt. Alle Berechnungen benötigt für die Bewegung des Roboters werden lokal auf dem Jetson Orin Nano durchgeführt und das GUI dient lediglich als Anzeige von Statusinformationen und der Befehlseingabe.

Abbildung 12, Erste Version der Benutzeroberfläche, C# WPF

Die Benutzeroberfläche haben wir möglichst schlicht gehalten und nur die einfachen Befehle beinhaltet. Das Aussehen zu „verbessern“ hätte uns viel Zeit gekostet und war nicht unsere Priorität.

3. Versuche zur Gehbewegung

Die ersten Gehversuche von SCOUT schlugen fehl. Der Roboter hat sich entweder nicht wirklich fortbewegen oder ist umgefallen. Zu diesem Zeitpunkt war die Gehbewegung statisch im Code festgelegt, d.h., dass alle Parameter wichtig für das Gehen (Schrittweite, Bodenabstand, Abstand der Füße, usw.) am Anfang von uns vorgegeben wurden.

In über 50 Testversuchen, bei denen wir viel mögliche Parameterkombinationen getestet haben, gelang es uns nicht ein stabiles Gehen des Roboters zu erzielen. Wir stellten fest, dass der Roboter jedes Mal instabil ist und nach wenigen Schritten umfällt, er starke und unvorhersehbare Kurven läuft oder sich kaum fortbewegt – wobei das Umfallen nach wenigen Schritten unser größtes Problem war.

3.2 Anpassung der Gehparameter durch Sensordaten

Mit der eingebauten Inertialen Messeinheit und einem Kalman-Filter können wir den Neigungswinkel von SCOUT in allen drei Achsen messen.

Mit einem PID-Kontroller wird die horizontale Distanz der Beine vom Nullpunkt (manuell gesetzter Punkt) und die Höhe der Füße vom Körper variiert.

Durch Echtzeitkorrektur wird verhindert, dass SCOUT sich zu stark neigt und eventuell umkippt. Dazu war ein aufwendiges manuelles stimmen der PID-Werte nötig.

4. Perspektive und zukünftige Weiterentwicklung

Perspektivisch soll SCOUT so weiterentwickelt werden, dass er autonom navigieren und sich bewegen kann, um so als Unterstützung für blinde Menschen zu funktionieren.

Dazu sind alle notwendigen Elemente im Design von SCOUT bereits berücksichtigt und angelegt, bedürfen aber noch der funktionalen Weiterentwicklung.

Insbesondere sind das die Entwicklung eines Kamerasystems und eines Neuronalen Netzes.

4.2 Reinforcement Learning

Um das Gehen zu verbessern wäre ein Ansatz mit Reinforcement Learning sinnvoll.

Dabei simulieren wir den Roboter in dem Programm Isaac Sim, wo er sich sehr ähnlich wie in der realen Welt verhält. Mit einem komplexem Neuronalem Netzwerk kann der Robot darauf hin von selbst lernen, wie er sich am besten fortbewegt.

Dabei geben wir dem Roboter einen Zielpunkt, zu dem er kommen muss. Mit einer Belohnungsfunktion wird die Distanz zum Ziel bewertet (je näher am Ziel, desto höher die Punkte). Durch diese Rückmeldung lernt das Netzwerk, dass die Bewegungen des Roboters gut oder eben schlecht gewesen sind. Eine Bewegung mit einer hohen Punktzahl (nahe am Ziel) hilft dem Netzwerk weitere und bessere Bewegungen zu erstellen.

Aktuell sind wir noch nicht so weit das Neuronale Netzwerk zu trainieren. Die Komplexität und vielen Faktoren des Systems macht die Aufgabe äußerst komplex.

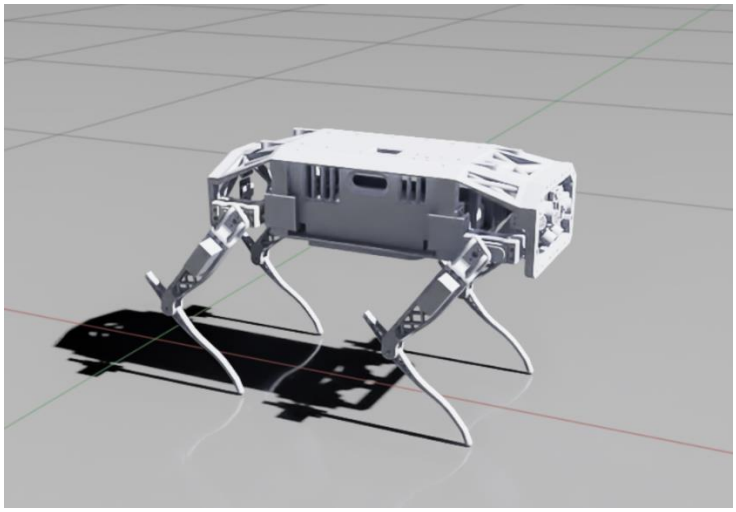


Abbildung 13, SCOUT ein einer simulierten Umgebung, Isaac Sim

4.3 Kamerasystem

Das Kamerasystem ist aktuell noch nicht voll funktionstüchtig und nicht in der Robotersteuerung integriert

SCOUT benutzt eine kleine, nach vorne gerichtete Kamera. Das Kamerabild wird dann vom Jetson verarbeitet.

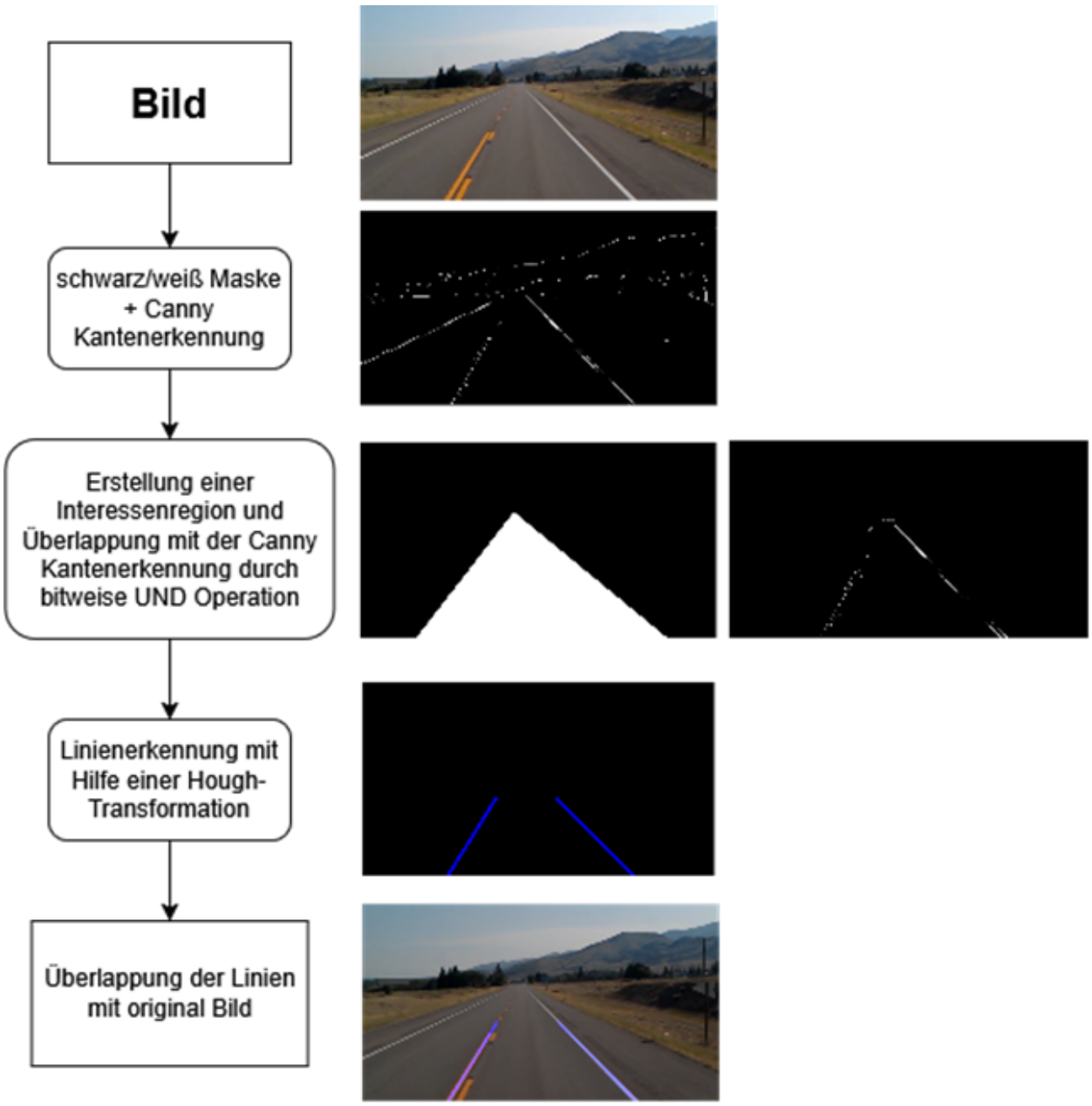


Abbildung 14, Visualisierung des Prozesses zur Linienerkennung, Jacob U.

Die oben abgebildeten Bilder sind Beispiele. Die Kameraperspektive wird im Betrieb anders sein und der Code dementsprechend angepasst.

Anhand der Linien aus der Linienerkennung können wir eine Mitte berechnen, der der Hund folgen soll. Unser oben gezeigte, vereinfachter Algorithmus funktioniert auch mit einem Video, oder im Betrieb mit dem Echt-zeit Kamerabild. Dafür haben wir eine Schleife erstellt, die bei jedem Durchlauf das nächste Bild eines Kamerabildes nimmt und die Linien neu ermittelt.

Aktuell arbeiten wir an einer Objekterkennung. SCOUT wird in der Lage sein Straßenschilder, Ampeln, Treppen und Autos zu erkennen und darauf

Entscheidungen zu treffen. Steht eine Ampel auf Rot, so bleibt --- stehen und gibt dies and den Nutzer weiter. Mit Hilfe der Open Source Library OpenCV und der YOLO-Architektur von Ultralytics lässt sich so etwas mit moderatem Aufwand umsetzen.

Das Kamerasystem ist auch der Grund gewesen, warum wir uns für einen NVIDIA Jetson Orin Nano anstelle des Raspberry Pis entschieden haben. Die Videoverarbeitung benötigt viel Rechenleistung, um das Kamerabild häufig genug pro Sekunde zu verarbeiten.

Wir nutzen aktuell eine Kamera mit einem schmalen Sichtfeld. Eine Kamera mit einem breiterem Sichtfeld wäre eine gute Verbesserung des Kamerasystems und wird Teil einer nächsten Projektphase sein.

4.4 Neuronale Netzwerke

Damit der Roboter auf bestimmte Situationen reagieren kann muss diesem beigebracht werden was wann zu tun ist. Mithilfe eines Neuronalen Netzes und unterstütztem Lernen wird dies möglich sein.

Ein Neuronales Netzwerk was eigenständig Entscheidungen trifft wäre eine sehr gute Ergänzung zu unserem Projekt, wird aber Aufgrund der Komplexität der Umgebungen nicht ganz einfach umzusetzen sein.

Beim unterstütztem Lernen von so einem neuronalem Netz würden wir dem Roboter eine Umgebung (in unserem Fall könnte dies eine Kreuzung mit roten Ampeln, einem Fahrradweg und parkenden oder fahrenden Autos sein) und eine gewollte Reaktion oder Handlung. Anhand von vielen solcher Datensätzen lernt ein Algorithmus ein bestimmtes Verhaltensmuster und kann bei einer neuen unbekannten Umgebung Ähnlichkeiten zu den Trainingsdaten finden und ähnliche Entscheidungen, wie die von uns vorgegebenen, treffen.

5. Fazit und Pläne

Die Entwicklung eines funktionierenden Roboterhundes war eine komplexe Aufgabe, die umfassendes Wissen in verschiedenen technischen Disziplinen wie Mechanik, Elektronik und Softwareentwicklung erforderte.

In diesem Projekt haben wir uns dieser Herausforderung mit dem Ziel gestellt, eine solide Basis für zukünftige Entwicklungen zu schaffen. Wir haben den gesamten Entwicklungsprozess – von der ersten Konzeption bis zur finalen Umsetzung – eigenständig durchgeführt. Unser Fokus lag darauf, einen Roboterhund zu konstruieren, der nicht nur als Plattform für weitere Projekte dient, sondern auch unsere spezifischen Designvorstellungen und technischen Anforderungen erfüllt. Dieser Roboterhund soll somit als vielseitige Grundlage für nachfolgende Entwicklung in Bereichen wie autonomer Navigation, Interaktion mit der Umwelt und fortgeschrittener Robotik dienen.

Wichtig für uns war die vollständige Eigenentwicklung aller mechanischen Komponenten. D.h. Sämtliche Bauteile, vom Rahmen über die Beine bis hin zu den Gelenken, wurden von uns selbst designt und anschließend im 3D-Druckverfahren hergestellt. Diese Vorgehensweise ermöglichte uns eine hohe Flexibilität bei der Gestaltung und die präzise Umsetzung unserer Vorstellungen.

Eine besondere Herausforderung stellte die Entwicklung der Beinbewegung dar. Hier galt es, innovative Lösungsansätze zu finden, um eine realistische und gleichzeitig funktionale Fortbewegung zu ermöglichen. Durch intensive Auseinandersetzung mit der Thematik, verschiedenen Tests und kreative Lösungsfindung konnten wir schließlich eine zufriedenstellende Umsetzung erreichen. Dabei spielten Aspekte wie die Koordination der einzelnen Gelenke, die Stabilität des Gangbildes und die Anpassung der Bewegungsparameter eine entscheidende Rolle.

Die Verknüpfung der einzelnen elektronischen Komponenten, wie Motoren, Sensoren und Mikrokontroller, sowie die Implementierung der Steuerung erfolgten nach unseren eigenen Vorstellungen. Wir entwickelten eigene Schaltpläne, löteten die Komponenten auf Platinen und programmierten die Teile, die für die Ansteuerung der Motoren und die Verarbeitung der Sensordaten verantwortlich ist.

Wir haben es geschafft einen eigenen Roboterhund komplett eigenständig zu entwickeln, wobei alle Aspekte aus eigener Arbeit stammen und keine Hilfe von außenstehenden Personen erfolgte.

Pläne:

Obwohl der Roboterhund bereits funktionsfähig ist, sehen wir noch Entwicklungspotenzial. Unsere zukünftigen Pläne konzentrieren sich auf folgende Bereiche:

- Natürlichere Bewegungen: Wir wollen die Beinbewegungen verfeinern und natürlichere Gangarten implementieren, um die Fortbewegung flüssiger und effizienter zu gestalten.
- Systemintegration: Die Abstimmung zwischen Mechanik, Elektronik und Software soll optimiert werden, um die Gesamtperformance und Zuverlässigkeit zu steigern.
- Entwicklung unseres Kamerasystems: Dazu zählt die Erkennung von Straßen, Wegen, Ampeln, Personen, Schildern usw..
- Neuronales Netzwerk: Die Integration eines neuronalen Netzwerks kann dem Roboterhund ermöglichen seine Umgebung besser wahrzunehmen, aus Erfahrungen zu lernen und autonom zu agieren.

Diese Weiterentwicklungen sollen die Funktionalität, Leistungsfähigkeit und Autonomie des Roboterhundes ermöglichen und eine solide Basis für zukünftige Forschungsarbeiten bilden.