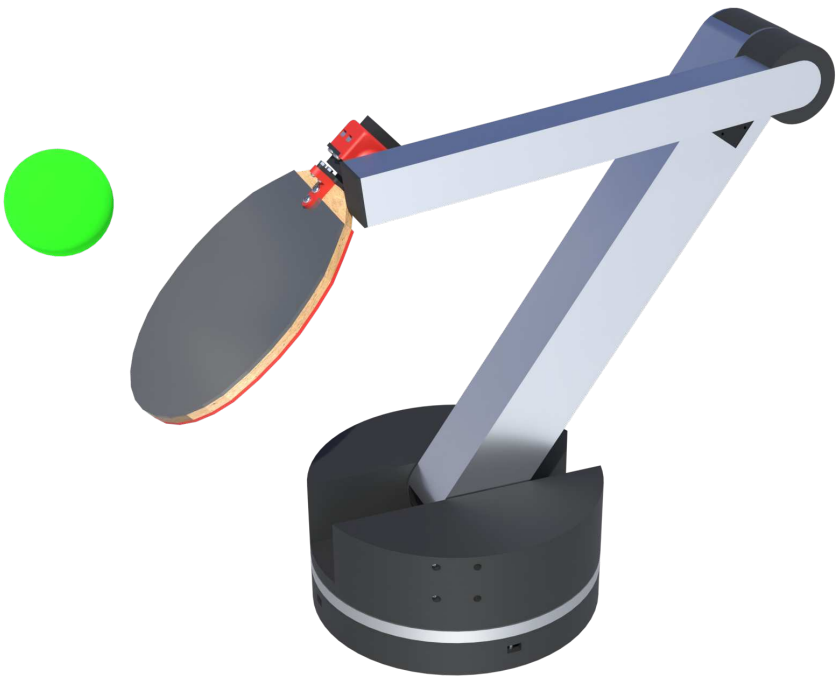
Wettbewerbsarbeit für Schweizer Jugend forscht

Entwicklung eines autonomen Roboterarm-Systems zum Pingpongspielen

Elias Bauer

Kollegium St. Fidelis Stans



Abstract

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung und dem Bau eines autonomen Roboterarm-Kamerasystems. Als Anwendungsbeispiele wurden das Fangen eines Balls mittels Netz und das direkte Retournieren eines Pingpongballs gewählt.

Das selbstentwickelte autonome System besteht aus drei Komponenten: (1) Roboterarm mit Pingpongschläger, (2) Kamerasystem zur dynamischen Bildaufnahme und (3) Computersoftware zur Erkennung und Lokalisation des Balls im dreidimensionalen Raum, der Vorhersage seiner Flugbahn sowie der Steuerung des Roboterarms.

Design und Bau des Roboterarms wurden mit einer Designsoftware und einem 3D-Drucker durchgeführt.

Die Komponenten des Systems wurden hinsichtlich der geplanten dynamischen Anwendungen optimiert, mit Fokus auf Schnelligkeit, Leichtigkeit und Robustheit. In der Arbeit wird die Auswahl der entsprechenden Bauteile erläutert, wie Motoren mit hohem Drehmoment und Kameras mit hoher Aufnahmefrequenz.

Für die beiden Anwendungen des Systems wurden zusätzliche Lösungen entwickelt und implementiert: So werden Methoden aufgezeigt, wie die Kameras mittels AprilTags ihre Position im Raum automatisch bestimmen, wie der Ball vor dessen Hintergrund erkannt wird, wie sich Messfehler der Kameras minimieren und wie sich zeitliche Abweichungen der Aufnahmen umgehen lassen. Ausserdem wird beschrieben, wie mit den erfassten Positionsdaten die Flugbahn des Balls vorhergesagt werden kann und es wird eine Methode gezeigt, wie man den optimalen Punkt ausrechnet, an dem der Ball durch den Roboterarm gefangen oder retourniert werden soll. Weiter beschäftigt sich die Arbeit mit der Berechnung einer optimalen Schlägerposition und Bewegungsabfolge, um den Ball mit einem am Arm montierten Pingpongschläger zurückzuspielen.

Geeignete Versuchsreihen zeigen die erfolgreiche Anwendung des Roboterarm-Systems sowohl beim Fangen eines Balls mit einer durchschnittlichen Fangquote von über 87%, als auch beim Pingpongspiel mit einer Rückspielquote von 90% und einer Spielsequenz von bis zu 20 Spielschlägen.

Zum Abschluss der Arbeit wird die multimediale Präsentation der Maturaarbeit mit Animationen und Echtzeit-Visualisierungen innerhalb Unreal Engine beschrieben.

Abbildung 1: https://youtube.com/watch?v=2rsmnFm0yxs

Danksagung

Die vorliegende Maturaarbeit wurde unter der Anleitung von **Herrn Urs Zellweger**, meinem Mentor und Physiklehrer, erstellt. Ihm danke ich für seine Anregungen und Hinweise, welche massgeblich zum Gelingen dieser Arbeit beigetragen haben.

Herrn **Fabian Lyck**, einem Betreuer bei der Schweizer Informatikolympiade[1], danke ich für seine Tipps bei der Auswahl geeigneter Bibliotheken und Techniken für das Kamerasystem.

Meinen Eltern, **Katja und Christian Bauer**, möchte ich für die Unterstützung beim Korrekturlesen der Maturaarbeit danken.

Herrn **Prof. Heinz Domeisen**, meinem Mentor im Rahmen des Schweizer Jugend forscht Wettbewerbs, möchte ich für die wertvollen Hinweise zur weiteren Charakterisierung und Evaluierung meines Roboterarm-Systems danken.

Inhaltsverzeichnis

1 Einführung **6**
1.1 Motivation . . . 6
1.2 Leitfragen . . . 7
1.3 Vorgehensweise . . . 7
1.4 Existierende Arbeiten . . . 8
1.4.1 „On Robotic Work-Space Sensing and Control“ . . . 8
1.4.2 „Cognitive Systems Table Tennis Robot“ . . . 8
1.4.3 „FORPHEUS“ . . . 8

2 Material und Methoden **9**
2.1 Bau des Roboterarms . . . 9
2.1.1 Bauanforderungen . . . 9
2.1.2 3D-Druck . . . 10
2.1.3 Basis . . . 10
2.1.4 Gelenk . . . 11
2.1.5 Hand . . . 11
2.1.6 Elektronik . . . 11
2.2 Kamerasystem . . . 12
2.2.1 Anforderungen . . . 12
2.2.2 Lochkamera-Modell . . . 13
2.2.3 Physische Kameras . . . 14
2.2.4 Ball Erkennungsroutine . . . 15
2.2.5 Flugbahn . . . 17
2.2.6 Kamera-Synchronisation . . . 17
2.2.7 Multithreading . . . 18
2.3 Steuerung . . . 19
2.3.1 System . . . 19
2.3.2 Inverse Kinematik . . . 19
2.3.3 Fangen . . . 20
2.3.4 Pingpong-Volleys . . . 21
2.3.5 Elastischer Stoss . . . 22
2.3.6 Umsetzung . . . 23
2.4 Material und Software . . . 24
2.4.1 Webcam . . . 24
2.4.2 Motoren . . . 24
2.4.3 Vierkantrohre . . . 24
2.4.4 OpenCV . . . 25
2.4.5 AprilTags . . . 25
2.4.6 Unreal Engine . . . 25
2.4.7 Fusion360 . . . 26
2.4.8 Blender . . . 26
2.4.9 Colmap . . . 26
2.5 Multimediale Präsentation in Unreal Engine . . . 27
2.5.1 Physikzimmer-Modell . . . 27

2.5.2 Roboterarm-Modell . . . 27
2.5.3 Präsentation und Animationen . . . 27
2.5.4 Kamera-Vorschau . . . 28
2.5.5 Elektronik-Visualisierung . . . 28

3 Ergebnisse **29**
3.1 Messungen . . . 29
3.1.1 Kamerasystem . . . 29
3.1.2 Pingpongspiel . . . 29
3.1.3 Fangfähigkeiten . . . 30
3.2 Produktentwicklung und Erreichen der Leitfragen . . . 30
3.3 Reflexion . . . 30
3.4 Weiterentwicklung . . . 31
3.5 Kosten . . . 31
3.6 Öffentliche Rezeption . . . 31

4 Anhang - Systemanalyse **32**
4.1 Design des Roboterarms . . . 32
4.2 Wahl der Motoren . . . 32
4.3 Vernachlässigung des Trägheitsmoments . . . 33
4.4 Interpolation der Ball-Erkennungen . . . 34
4.5 Einfluss der Versorgungspannung auf die Motorgeschwindigkeit . . . 34
4.6 Zuverlässigkeit der Stromquelle . . . 35

Kapitel 1

Einführung

1.1 Motivation

Autonome Systeme sind Geräte, Maschinen und Programme, die eigenständig handeln und Probleme lösen. Sie spielen schon heute im Leben von uns Menschen eine wichtige Rolle. Beispiele dafür sind Roboter, selbstfahrende Autos, Autopiloten oder KIs wie ChatGPT.

Autonome Robotersysteme werden auch in der Zukunft in den Bereichen Sport, Landwirtschaft, sowie der Pflege und medizinischen Versorgung von Menschen immer wichtiger werden. Kommerzielle Roboter und Kamerasysteme sind jedoch für viele Anwendungen zu teuer. In meiner Maturaarbeit zeige ich, dass dies auch deutlich günstiger geht.

Seit meiner Kindheit habe ich mich mit den Komponenten autonomer Systeme beschäftigt, sei es mit programmierbaren Robotern oder mit Algorithmen zum Lösen von Problemen. Im Rahmen meiner Maturaarbeit wollte ich deshalb ein autonomes System selbst entwickeln und mit einer konkreten Anwendung testen.

Meine Auswahl fiel daher auf ein autonomes Roboterarm-Kamerasystem, angewendet auf das Fangen oder das direkte Zurückspielen eines Pingpongballs. Pingpong ist bis heute eine meiner Lieblingssportarten. Daher lag es nahe, dies auch in meine Maturaarbeit einzubringen.

Die gewählte dynamische Anwendung stellt eine besondere Herausforderung, relativ zu einem Roboterarm-Einsatz in der Industrie, dar. Dort agiert ein Roboterarm typischerweise am ruhenden Objekt auf Basis von vorprogrammierten Befehlen, mit Fokus auf Genauigkeit und Wiederholbarkeit. Hier jedoch musste das System spezifisch für eine dynamische Anwendung optimiert werden, mit dem Schwerpunkt auf Schnelligkeit und Robustheit.

Das System umfasst drei Komponenten: (1) Roboterarm mit Pingpongschläger, (2) Kamerasystem zur dynamischen Bildaufnahme und (3) Computersoftware zur Erkennung und Lokalisation des Balls im dreidimensionalen Raum, der Vorhersage seiner Flugbahn sowie der Steuerung des Roboterarms.

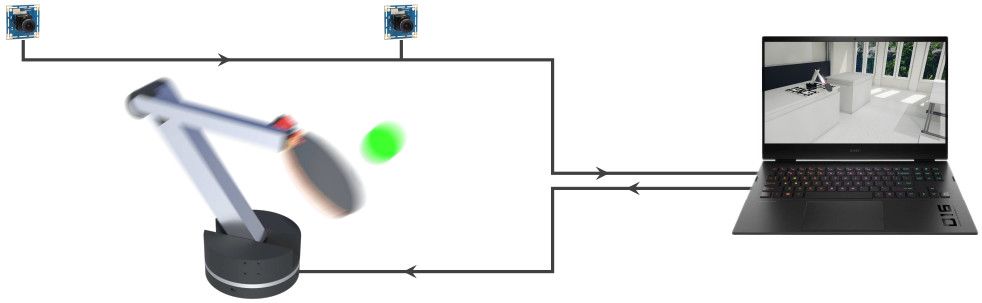


Abbildung 1.1: Schema der Komponenten[2] [3] [4]

Die Entwicklung dieser Komponenten und ihre Integration war das Ziel dieser Maturaarbeit und es ergaben sich somit die folgenden Leitfragen:

1.2 Leitfragen

- Wie baut man einen Roboterarm, der einen Pingpongball fangen und mit einem Tischtennisschläger direkt zurückspielen kann?
- Wie erstellt man ein Kamerasystem mit Webcams, welches die Flugbahn eines Balls erkennen und vorhersagen kann?
- Wie entwickelt man eine Software zur Integration und Steuerung beider Komponenten, damit das System autonom agieren kann?

1.3 Vorgehensweise

Nachdem ich die Idee hatte, einen Roboterarm zu entwickeln, der Pingpong spielen und Bälle fangen kann, stellte ich zunächst einige Berechnungen an, ob solch ein System überhaupt realisierbar ist und was die Anforderungen an das System und die einzelnen Komponenten wären. Des Weiteren führte ich Recherchen im Internet durch, hinsichtlich bisheriger Publikationen und existierender Beispiele vergleichbarer Systeme und deren einzelner Komponenten.

Im nächsten Schritt arbeitete ich ein erstes Konzept mit den notwendigen Kernfunktionalitäten der drei Komponenten aus. In einem iterativen Prozess mit den Arbeitsschritten: Detaillierung der Anforderungen, Design, Prototypenbau und Testen entwickelte ich zunächst den Roboterarm und das Kamerasystem und testete sie mit einer ersten rudimentären Software. Die einzelnen Bauteile wurden auf Basis ihrer Anforderungen und Spezifikationen über geeignete Onlinedienste bzw. eine Metallbaufirma bezogen.

Nachdem die Hardware-Komponenten etabliert waren, konnte die Software weiterentwickelt werden. Für die einzelnen Anwendungen des Ballfangens und Pingpongspielens mussten dann spezielle Lösungsansätze für Teilprobleme erarbeitet werden. Im Verlauf der nächsten Monate wurde die Software durch Testen und Ausprobieren optimiert, mit dem Ziel, das System schneller und genauer zu machen. Zur Beurteilung der Performance mussten entsprechende Kennzahlen wie Fangquote, Rückspielquote und Länge einer Spielsequenz definiert und verwendet werden.

1.4 Existierende Arbeiten

Im Folgenden werden einige bereits publizierte Projekte zum Thema Kameratracking, Roboterarm und Pingpongspiel vorgestellt. Die Arbeiten verwenden verschiedene Ball-Erkennungsverfahren, Roboterarme und Steuerungstechniken, die jedoch nicht im Rahmen dieser Maturarbeit Anwendung fanden.

1.4.1 „On Robotic Work-Space Sensing and Control“

In der Bachelorarbeit von Magnus Linderoth wird ein robustes Verfahren erklärt, um high-speed Bildanalysen durchzuführen, die variierende Beleuchtung und „Motion Blur“ bei der Erkennung eines fliegenden Balls berücksichtigen.[5] Es wird ebenfalls ein Kalman Filter beschrieben, der aus Ball-Erkennungen aus der Sicht zweier Kameraperspektiven, ohne Triangulation, die Flugbahn des Balls direkt generieren kann. Dies ist eine alternative Methode, um das Problem der Kamera-Synchronisation zu umgehen. Diese Technik wird benutzt, um einen Ball mithilfe eines industriellen Roboterarms mit hoher Genauigkeit zu fangen.

1.4.2 „Cognitive Systems Table Tennis Robot“

Diese Arbeit beschäftigt sich mit dem Pingpongspielen gegen einen industriellen Roboterarm.[6] Der Ball wird, wie in meiner Maturaarbeit, anhand von dessen Farbe und Form erkannt. Zusätzlich wird jedoch dessen Spin mithilfe des Logos auf dem Ball ermittelt. Die Flugbahn-Vorhersage wird auch wieder mit einem Kalman-Filter gemacht, welcher den Luftwiderstand des Balls berücksichtigt. In meiner Arbeit wende ich stattdessen Regressionsanalysen auf die ermittelten Ball-Punkte an, um die Ballbewegung mit Polynomen anzunähern. Ausserdem ist, im Gegensatz zu meiner Arbeit, die Schlägerbewegung nicht durch einen Algorithmus bestimmt. Stattdessen bringt der Roboter sich mit Reinforcement Learning die richtigen Bewegungsabfolgen selber bei.

1.4.3 „FORPHEUS“

FORPHEUS, ein von der japanischen Elektronikfirma Omron entwickelter Tischtennisroboter, besitzt eine Vielzahl von Trackingkameras und Bewegungssensoren, um nicht nur den Ball zu verfolgen, sondern auch die Bewegungen des Spielers zu analysieren.[7] Damit kann der Roboter nicht bloss auf kompetitivem Level Tischtennis spielen, sondern soll auch dem Spieler Rückmeldungen zu dessen Technik geben können. Der Roboter ist oberhalb des Tisches angebracht und steuert den Schläger mit mehreren Stäben, womit die bewegte Masse minimiert wird.

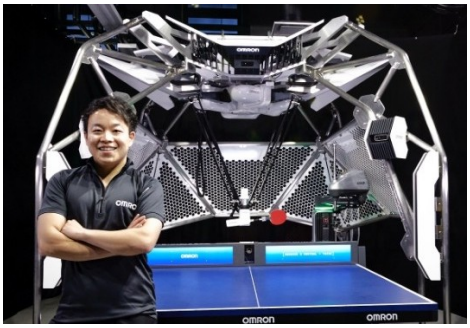


Abbildung 1.2: FORPHEUS Gen.7 mit dessen Hauptentwickler[8]

Kapitel 2

Material und Methoden

2.1 Bau des Roboterarms

2.1.1 Bauanforderungen

- **Fünf Freiheitsgrade**

  Das Ende des Roboterarms sollte jeden Punkt des Raums innerhalb einer, durch die Armlänge definierten, Halbkugel erreichen können. Dafür sind drei Motoren nötig, zwei in der Basis und ein weiterer am Gelenk zwischen Ober- und Unterarm. Dies ist vergleichbar mit dem menschlichen Arm, welcher an der Schulter ein Kugelgelenk und am Ellbogen ein Scharniergelenk besitzt. Dabei hat das Schultergelenk beim Menschen noch eine zusätzliche Drehachse. Um den Schläger oder den Ring beliebig ausrichten zu können, sind zwei weitere Motoren notwendig. Einer für die Rollachse und einer für die Nickachse (Auf- und Abwärtsbewegung). Die menschliche Hand besitzt auch hier einen Freiheitsgrad mehr, und zwar die Seitwärtsbewegung um die Gierachse.

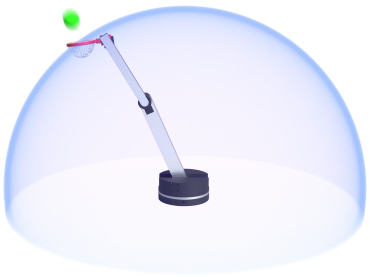


Abbildung 2.1: Reichweite des Armes[4]

- **Schnelligkeit**

  Der Roboterarm sollte sich schnell genug bewegen können, um den Ball zu erreichen. Ein Ball, der aus 2 m zugeworfen wird, ist etwa 1 s lang in der Luft. Nach Berücksichtigung der Latenz und der Zeit, bis die Flugbahn erkannt wird, bleiben dem Arm etwa 0.7 s Zeit, um sich an jede beliebige Zielposition zu bewegen. Von der Ausgangsposition in der Mitte muss sich kein Motor weiter als 90° drehen, um dies zu erreichen. Die Motoren sollten also alle eine Mindestgeschwindigkeit von ca. 130 $\frac{\text{deg}}{\text{s}}$ haben.

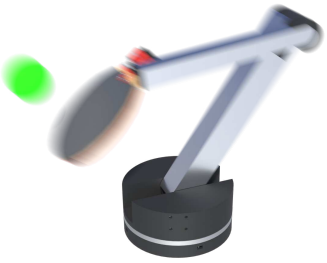


Abbildung 2.2: Arm-Bewegung[4]

- **Genauigkeit**

  Der Ring zum Fangen des Balls hat einen Durchmesser von 9 cm, und der Ball einen Durchmesser von 6 cm. Daraus ergibt sich eine Toleranz von 1.5 cm, damit der Ball den Ring beim Fangen nicht streift. Bei einer Armlänge von 70 cm entspricht dies einer Motorgenauigkeit von ca. 1.25°. Um Pingpong spielen zu können, reicht diese Genauigkeit auch aus, da der Pingpongschläger ja grösser als der Ring ist.

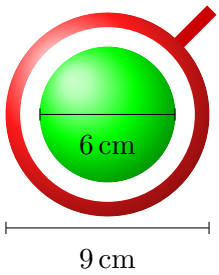


Abbildung 2.3: Ballgrösse relativ zum Reifen[9]

2.1.2 3D-Druck

Für das Design und den Bau des Roboterarms wurde die Designsoftware Fusion 360 und der 3D-Drucker Ender 5 Pro verwendet. Die meisten Arbeitsschritte konnten damit zuhause durchgeführt werden, was den Design-Prozess sehr erleichterte. So war es möglich, einzelne Teile schnell zu testen und weiter zu optimieren. Auch konnten beispielsweise verschiedene Iterationen der Hand solange ausprobiert werden, bis das Ergebnis zufriedenstellend war. Der 3D-Drucker Ender 5 Pro ist mit einem Preis von 400 CHF ein Einsteigermodell. Er hat eine XY-Genauigkeit von 0.2 mm, welche sehr hilfreich für kleinere Bauteile und Details ist, beispielsweise die der Hand. Die eher grosse Buildplate (22 cm × 22 cm) war notwendig, um die Basis im Ganzen drucken zu können.

Abbildung 2.4: Ender 5 Pro[10]

2.1.3 Basis

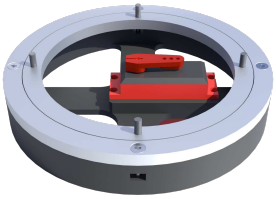


Abbildung 2.5: Untere Basis[4]

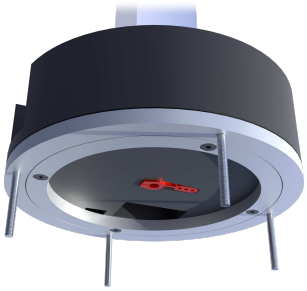


Abbildung 2.6: Basis von unten[4]

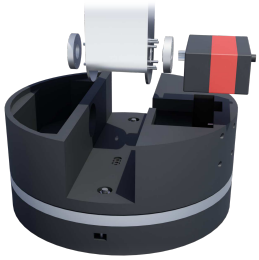


Abbildung 2.7: Obere Basis[4]

Die Basis wurde in mehreren Iterationen hinsichtlich Drehgeschwindigkeit, Drehmoment und flexiblem Kabelverlauf optimiert. Das finale Design besteht aus einer oberen und einer unteren Hälfte. Um beide Hälften zu verbinden, befindet sich zwischen ihnen eine Drehscheibe mit Kugellager, welche das reibungslose Drehen ermöglicht. Die untere Basis ist mit vier langen Schrauben durch die Drehscheibe hindurch an die Grundplatte geschraubt. Der Motor in der unteren Hälfte ist, so wie alle anderen Motoren, ein Servo. Er ist mit einem Servoarm an der oberen Hälfe fixiert, um diese zu drehen. Die obere Hälfte der Basis ist mit vier Schrauben am inneren Ring der Drehplatte befestigt. In ihr befindet sich der Motor für den „Oberarm“, sowie die Elektronik für die Steuerung der Motoren. Der Motor für den Oberarm sitzt in dem oberen Teil, wo er festgeschraubt ist. In der gegenüberliegenden Hälfte ist Platz für die Elektronik, um die Motoren des Arms zu steuern. Wie in Abbildung 2.7 zu erkennen, ist am Oberarm der Servo befestigt. Für eine robuste Fixierung des Arms war es notwendig, diesen auf der gegenüberliegenden Seite zusätzlich mit einem Befestigungsring zu stabilisieren. Durch das Loch im Befestigungsring konnten die Kabel für die oberen Motoren mit der Elektronik verbunden werden.

2.1.4 Gelenk

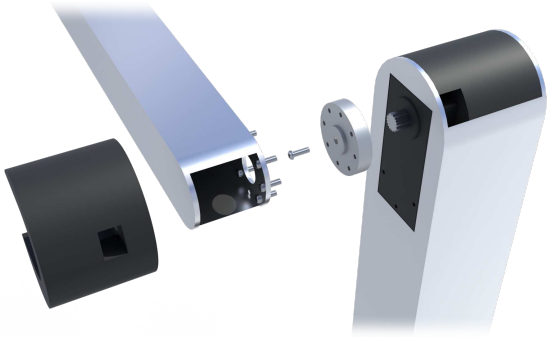


Abbildung 2.8: Oberarm-Gelenk[4]

Das Gelenk zwischen Ober- und Unterarm ist sehr simpel, dank der Entscheidung, die Rohrgrösse des Oberarms so zu wählen, dass der Motor exakt hineinpasst. An diesen ist ein runder Servo-Arm montiert, der gleiche Typ wie der in der Basis. Am Servo-Arm ist der Unterarm mit sechs Schrauben befestigt. Die Abdeckungen sind hauptsächlich ästhetischer Natur, nur die am Unterarm stützt das Gelenk und entlastet dadurch zusätzlich die Achse des Servo-Motors.

2.1.5 Hand

Die „Hand“ des Roboterarms ermöglicht eine Funktionalisierung des Systems für die jeweilige Anwendung. Im vorliegenden Fall ist die Hand entweder ein Ring zum Fangen des Balls, oder ein Pingpongschläger, um einen Ball zurückzuspielen. Die Hand wurde in mehreren Entwicklungsschritten hinsichtlich Stabilität und Schlagkraft optimiert. Das finale Design benutzt zwei gleiche Motoren für die beiden Freiheitsgrade. Wie in Abbildung 2.9 zu sehen, ist im Vierkantrohr, aus dem der Unterarm besteht, ein Servomotor festgeschraubt. An diesem ist eine Vorrichtung für den zweiten Servomotor befestigt und an diesem wiederum sind zwei L-förmige Halterungen angebracht. Das Werkzeug, welches der Arm bewegen soll, kann an dieser zweiteiligen Halterung festgeschraubt werden (siehe Abbildung 2.10 und 2.11).

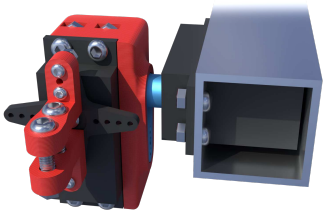


Abbildung 2.9: Hand[4]

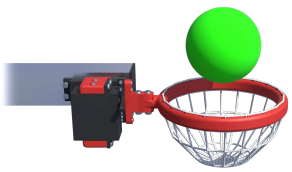


Abbildung 2.10: Hand mit Netz[4]

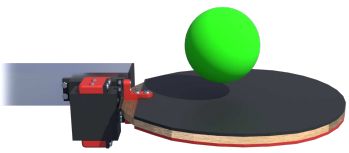


Abbildung 2.11: Hand mit Schläger[4]

2.1.6 Elektronik

Zur Steuerung der Motoren befindet sich ein Arduino-Nano in der Basis, welcher gleichzeitig mit dem Computer verbunden ist. Die fünf Servomotoren haben je drei Kabel: eines für Strom, eines für Erdung und eines für die Positionsdaten. Der Motor in der Basis, sowie die beiden Motoren für Ober- und Unterarm, benötigen 12 V, wobei die Motoren in der Hand 6 V brauchen. Diese 12 V liefert eine verstellbare Labor-Stromquelle. Um 12 V zu 6 V umzuwandeln, befindet sich in der Basis ein sogenannter „buck converter“. Es handelt sich dabei um einen kleinen, verstellbaren Abwärtswandler. Die Erdung des Arduinos ist mit der Erdung der Stromquelle und allen Erdungen der Motoren verbunden, und die Daten-Eingänge der Motoren sind jeweils mit einem digitalen Pin des Arduinos verbunden.

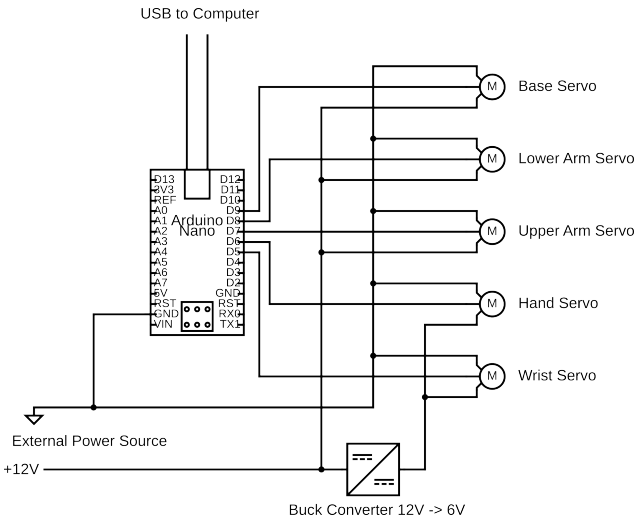


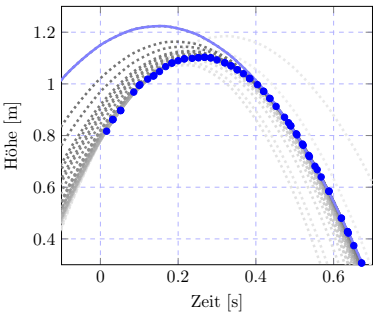
Abbildung 2.12: Schaltplan der Elektronik[11]

2.2 Kamerasystem

2.2.1 Anforderungen

- **Stereo Kamerasystem:** Für die Lokalisation des Balls im dreidimensionalen Raum werden zwei Kameras benötigt. Die Positionen der Kameras links und rechts hinten, oberhalb des Roboterarms, wurden so gewählt, dass die Kameras den Bewegungsablauf störungsfrei aufnehmen können und weit genug voneinander entfernt sind, um gleichzeitig eine dreidimensionale Lokalisation des Balls zu ermöglichen. Auch haben die Kameras somit freie Sicht auf die AprilTags, um deren Position bestimmen zu können.

- **Genauigkeit:** Einzelne Messungen dürfen etwas abweichen, da sich kleine Abweichungen durch die Parameterschätzung der Flugbahnparabel ausgleichen. Ausserdem müssen die Messungen am Anfang der Flugbahn noch nicht so genau sein, da sie zu diesem Zeitpunkt nur eine ungefähre Abfangposition liefern müssen, sodass der Roboterarm beginnen kann, sich in Bewegung zu setzen. Die Fang-Position kann dann verfeinert werden, sobald weitere Messungen eintreffen. Jedoch sollte die schlussendliche Vorhersage nicht mehr als die Hälfte des Balldurchmessers von der wahren Position abweichen, da ansonsten ein konsistentes Fangen des Balls schwierig werden würde.



Ermittelte Flugbahn eines Balls[9]

- **Erkennungsgeschwindigkeit:** Der Ball wird dem Roboterarm aus einer Distanz von ca. 2 m zugeworfen. Die Flugdauer beträgt ungefähr 1 s. Es dauert ungefähr 0.2 s bis die Kameras genügend Bilder aufnehmen, um eine erste Näherung der Flugbahn zu berechnen. Gleichzeitig ergibt sich eine Latenz von ca. 0.1 s im ganzen System. Somit verbleiben ca. 0.7 s Zeit, um den Arm zu bewegen. Die 0.2 s bis zur Erkennung sind sehr kurz, aber der Roboterarm kann sich mit einer vorläufigen Messung schon an die vorausgesagte Fang-Position hinbewegen. Sobald weitere Messungen eintreffen, kann diese laufend verbessert werden.

- **Flexibilität:** Die Kameras sollten bewegt werden können und danach automatisch ihre Position aktualisieren. Dies wird mithilfe der AprilTags erreicht. Diese Tags auf einem Bild zu erkennen, ist aber - relativ zum Erkennen eines Balls - rechenintensiv. Die Erkennungsrate ist davon abhängig, wie stark sich die Position der Kamera seit der letzten Messung verändert hat. Wenn die Position fast gleich bleibt, aktualisiert sie sich erst etwa eine Sekunde später wieder. Verändert sich die Position aber stark, wird gleich in der nächsten Frame (16 ms später) eine weitere Positionsbestimmung durchgeführt. Um Messungenauigkeiten auszugleichen, ist die zur Triangulation des Balls verwendete Kamera-Position über die letzten zehn Messungen gemittelt.

2.2.2 Lochkamera-Modell

Das Lochkamera-Modell ist eine vereinfachte Darstellung der Funktionsweise einer Kamera. Sie besteht aus einem kleinen Loch (der Lochblende), durch das das Licht eintritt und auf der gegenüberliegenden Seite eines abgedunkelten Kastens ein umgekehrtes Bild erzeugt. Die Brennweite ist der Abstand zwischen der Lochblende und der Bildebene und bestimmt das Sichtfeld (Field of View) der Kamera. Die Bildebene ist der Ort, an dem das Bild entsteht. Am Brennpunkt laufen die Lichtstrahlen zusammen, hier also bei der Lochblende. Da die Kamera keine Linse hat, ist das entstandene Bild nicht verzerrt.[12]

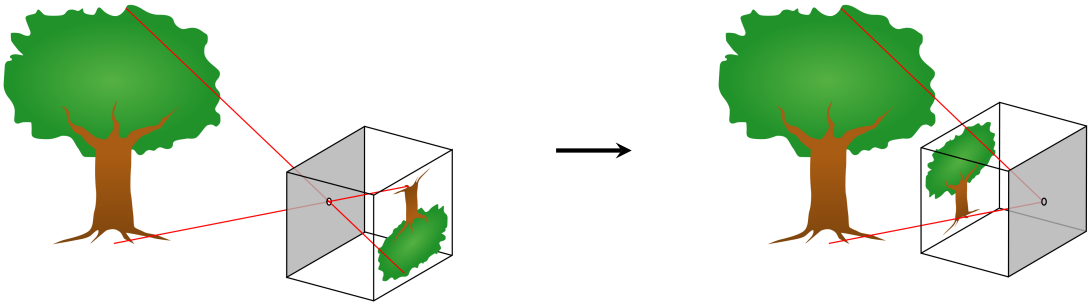


Abbildung 2.13: Lochkamera, welche einen Baum aufnimmt[13]

Man kann nun die Bildebene nach vorne spiegeln, bei gleichbleibendem Abstand zum Brennpunkt. Meiner Meinung nach ist diese Variante des Modells etwas intuitiver, als wenn sich die Bildebene hinter dem Brennpunkt befindet. Die Projektion eines Punkts ist immer noch dort auf der Bildebene, wo die Gerade, die durch den Punkt und den Brennpunkt geht, diese schneidet.

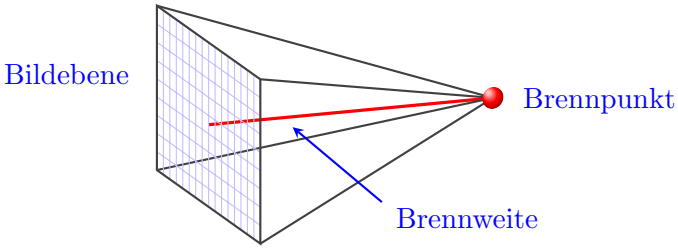


Abbildung 2.14: Lochkamera-Modell[9]

In der realen Welt treffen Lichtstrahlen aus der Umgebung auf den Sensor einer Kamera. In der Computergrafik macht man aber stattdessen das Ganze rückwärts. Es werden „Sichtstrahlen“ aus der Kamera herausgesendet (ein Strahl pro Pixel) und dann wird die Farbe des getroffenen Objekts auf den Pixel übertragen. Dies macht beispielsweise das Programm Blender, um photorealistische Bilder zu erstellen. Dabei kann der Strahl auch von Objekten reflektiert werden, um Spiegelungen und andere Effekte zu erzeugen. Dieser Prozess wird auch Ray-Tracing genannt.[14]

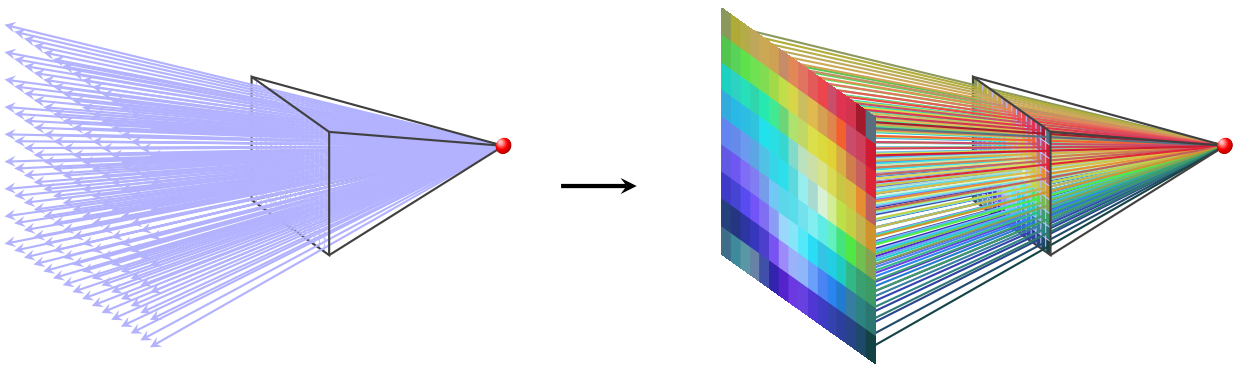


Abbildung 2.15: Lochkamera-Modell mit „Sichtstrahlen“, welche die Farbe des getroffenen Objekts aufnehmen[9]

2.2.3 Physische Kameras

Abbildung 2.16: ELP-Kamera[2]

Intrinsische Parameter
Die intrinsischen Parameter sind von Kamera zu Kamera verschieden, verändern sich aber nicht, ausser die Kamera hat einen optischen Zoom, bei dem sich das Objektiv bewegt. Die wichtigsten Parameter sind die Auflösung der Kamera sowie deren Brennweite. Die Verzerrungskoeffizienten der Linse werden aber bei einer stärker ausgeprägten Verzerrung ebenfalls gebraucht, um den Ball richtig triangulieren zu können.[15]

Abbildung 2.17: Basisplatte[4]

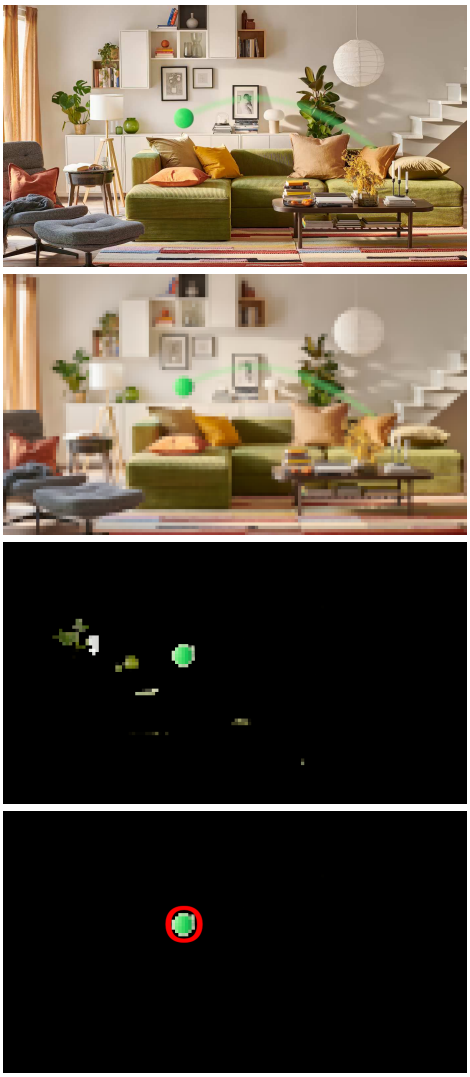
Extrinsische Parameter
Extrinsische Parameter beschreiben die Position und Rotation einer Kamera in der dreidimensionalen Welt. Die AprilTags ermöglichen eine automatische und ständig aktualisierte Bestimmung der extrinsischen Parameter, relativ zum Roboterarm.[15]

Verzerrung
Die meisten Kameralinsen verzerren das aufgenommene Bild etwas. Das extremste Beispiel hierfür sind Fischaugenlinsen von Überwachungskameras. Bei tonnenförmiger Verzerrung werden gerade Linien nach aussen, während sie bei kissenförmiger Verzerrung nach innen gebogen werden. Diese Verzerrungsarten haben somit Auswirkungen auf die Form der Objekte im Bild. Würde diese Verzerrung nicht vor der Ballerkennung rückgängig gemacht werden, könnte es zu erheblichen Messfehlern führen, insbesondere, wenn sich der Ball zu nahe am Bildrand befindet.[15][16] Die verwendeten Webcams haben einen leichten Fischaugen-Effekt, weisen also eine tonnenförmige Verzerrung auf. Um diese wieder aus dem Bild herauszurechnen, werden die Verzerrungskoeffizienten der intrinsischen Parameter verwendet. Die OpenCV-Funktion `undistort` entzerrt mit diesen Koeffizienten das aufgenommene Bild, um es für die nächsten Schritte wieder zu korrigieren.

Abbildung 2.18: Die zwei Verzerrungsarten, Originalbild:[17]

2.2.4 Ball Erkennungsroutine

Abbildung 2.19: Verarbeitungsschritte, Originalbild:[17]



1. Schritt: Bild einlesen
Die Webcam nimmt mit 720p bei 60 fps auf und schickt die Bilder im MJPG-Format an den Computer, der sie dann wieder dekomprimiert und mithilfe der Verzerrungskoeffizienten entzerrt.

2. Schritt: Downscaling
Um die Verarbeitungsgeschwindigkeit der nächsten Schritte zu beschleunigen, wird die Auflösung des Bilds in beiden Dimensionen um den Faktor 0.5 verringert. Dadurch müssen nur $\frac{1}{4}$ der Pixel verarbeitet werden, ohne die Genauigkeit dabei zu sehr zu beeinträchtigen.

3. Schritt: Farb- und Hintergrundfilter
Es werden Pixel herausgefiltert, deren Farbe anders ist als die des Balls. Damit Objekte im Hintergrund, die von gleicher Farbe wie der Ball sind, nicht fälschlicherweise als dieser erkannt werden, müssen ebenfalls Pixel herausgefiltert werden, deren Farbe sich in der letzten Zeit nicht verändert hat.

4. Schritt: Blob-Erkennung
Es werden Gruppen benachbarter Pixel erkannt, die eine Mindestgrösse erfüllen und einigermassen rund sind. Das Zentrum der grössten Gruppe ist dann die 2D-Projektion des Balls, mit deren Hilfe die 3D-Position trianguliert wird.

5. Schritt: Triangulation
Bei einer Lochkamera liegen: Ein Punkt, seine Projektion auf der Bildebene sowie der Brennpunkt der Kamera *alle* auf einer Geraden. Dort, wo sich die Ball-Geraden der beiden Kameras im 3D-Raum schneiden, befindet sich der Ball.[18] Dafür wird die OpenCV-Funktion `triangulatePoints` verwendet, welche die 2D-Projektionen des Balls und die Projektions-Matrizen (die Matrix der intrinsischen Parameter multipliziert mit der Matrix der extrinsischen Parameter) der beiden Kameras als Parameter nimmt und daraus die 3D-Position des Balls bestimmt.

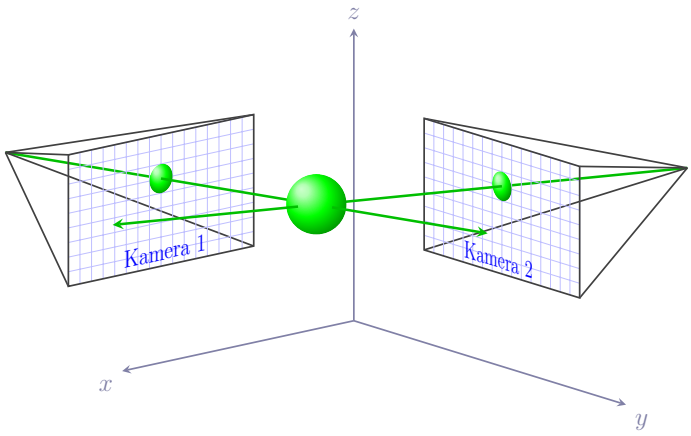


Abbildung 2.20: Triangulation der 3D-Position aus zwei Projektionen[9]

Code Beispiel: Blob Erkennung

```
// Hier werden die benötigten Bibliotheken angegeben:
#include <opencv2/core.hpp>
#include <opencv2/features2d/features2d.hpp>
#include <bits/stdc++.h>
using namespace cv;
using namespace std;

// Die Funktion "detect_ball" nimmt das gefilterte Bild als Argument und
// gibt den Standort des Balls auf dem Bild zurück, falls er zu sehen ist.
optional<Point2f> detect_ball(Mat &threshold_image)
{
        SimpleBlobDetector::Params blob_params;

        // nur Blobs, die convex genug sind
        blob_params.filterByConvexity = true;
        blob_params.minConvexity = 0.6;
        blob_params.maxConvexity = 1;

        // nur Blobs, die mindestens 1000 Pixel und maximal 50000 Pixel gross sind
        blob_params.filterByArea = true;
        blob_params.minArea = 1000;
        blob_params.maxArea = 50000;

        // nur Blobs, die kreisförmig genug sind
        blob_params.filterByCircularity = false;
        blob_params.minCircularity = 0.6;
        blob_params.maxCircularity = 1;

        // ein Blob Detector wird mit den angegebenen Parametern erstellt
        SimpleBlobDetector *blob_detector = SimpleBlobDetector::create(blob_params);

        // der Blob Detector wird auf dem Bild laufen gelassen und füllt die Liste "blobs"
        vector<KeyPoint> blobs;
        blob_detector->detect(threshold_image, blobs);

        // wird nur ausgeführt, falls mindestens ein Kandidat gefunden wurde
        if (!blobs.empty())
        {
                // die erkannten Blobs werden nach Grösse sortiert
                sort(blobs.begin(), blobs.end(), [](const KeyPoint &a, const KeyPoint &b)
                        { return a.size > b.size; });

                // der Standort des grössten Blobs wird aus der Funktion zurückgegeben
                return blobs[0].pt;
        }

        // falls kein Ball gefunden wurde, wird nichts zurückgegeben
        return {};
}
```

2.2.5 Flugbahn

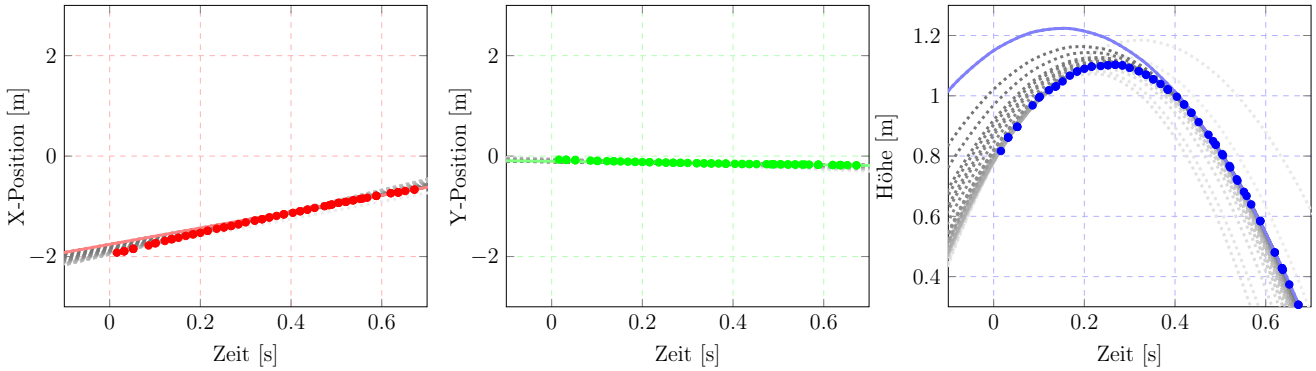


Abbildung 2.21: Mit dem Kamerasystem aufgenommene Flugbahn[9]

Die Methode der kleinsten Quadrate (least-squares) ist ein Verfahren zur Annäherung einer Polynomfunktion anhand einer gegebenen Datenmenge, bei der mehr Datenpunkte vorhanden sind als Unbekannte, die Lösung also überbestimmt ist. Dabei werden die Unbekannten so gewählt, dass die Summe der Abweichungen im Quadrat möglichst klein ist.[19]
Das Kamerasystem liefert die 3D-Koordinaten des Balls, aus denen dieses Verfahren die Funktionen bestimmen kann. Für die X- und Y-Koordinaten des Balls können Polynome ersten Grades verwendet werden, da der Luftwiderstand, vor allem bei den späteren Messungen, vernachlässigbar ist. Die Z-Koordinate muss jedoch durch ein Polynom zweiten Grades repräsentiert werden, da die Erdbeschleunigung auf den Ball wirkt. Ballbewegungen, die nicht Teil einer Flugbahn sind, können leicht herausgefiltert werden. Bei denen ist die Abweichung von den Polynomen sehr gross und die zweite Ableitung der Z-Koordinate ist weit von $-9.81\,\frac{\mathrm{m}}{\mathrm{s}^2}$ entfernt.
Der Ball wird auch besser gefangen, wenn nur die letzten 30 Messungen für die Bestimmung der Funktionsparameter verwendet werden. Ansonsten wird das Ergebnis zu sehr durch die ersten Messungen verfälscht, welche eine viel höhere Messungenauigkeit aufweisen als die letzten, bei denen der Ball schon deutlich näher war. Die Fangposition kann somit bis kurz vor Eintreffen des Balls kontinuierlich verfeinert werden.

2.2.6 Kamera-Synchronisation

Damit der Ball genau trianguliert werden kann (aus den 2D-Projektionen die 3D-Position bestimmen), müssen die Aufnahmen der beiden Kameras möglichst gleichzeitig gemacht werden. Es gibt aber leider keine Möglichkeit, die Kameras zu synchronisieren oder zu steuern, wann sie ein Bild aufnehmen sollen. Beide Kameras nehmen mit einer Bildrate von 60 fps (Frames per Second) auf. Daraus ergibt sich eine zeitliche Versetzung von ca. 16 ms zwischen zwei Aufnahmen. Im ungünstigsten Fall liegen also 8 ms zwischen den Aufnahmen beider Kameras.

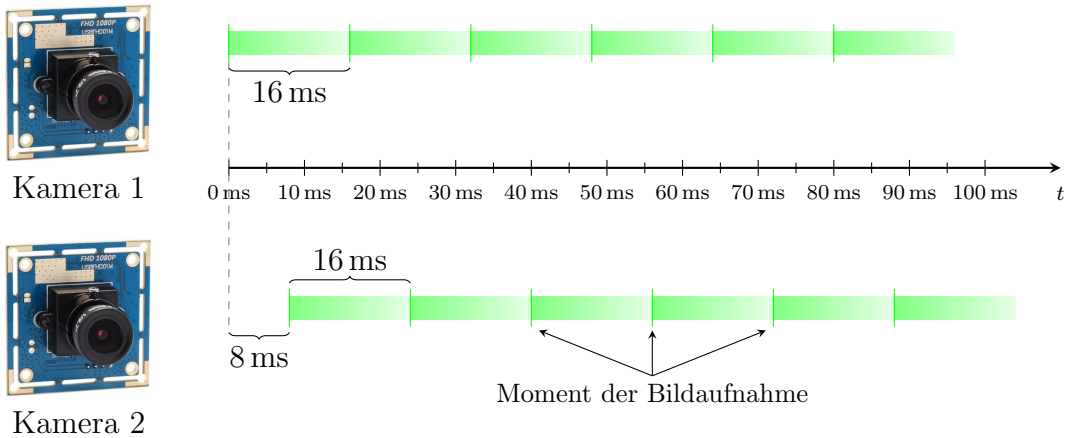


Abbildung 2.22: Kamera-Synchronisierung[20]

Das Problem der nicht vorhandenen Kamera-Synchronisation kann jedoch weitgehend umgangen werden: Von den Kameras kann der genaue Zeitpunkt, an dem ein Bild aufgenommen wurde, verlangt werden. Somit können Lücken zwischen Aufnahmen rechnerisch gefüllt werden. Machte beispielsweise eine der Kameras bereits zwei Aufnahmen, eine bei 0 ms und eine bei 16 ms, während die zweite Kamera erst eine Aufnahme bei 8 ms machte, kann bei der ersten Kamera zwischen den letzten beiden Positionen linear interpoliert werden, um die entsprechende Position bei 8 ms abzuschätzen. Diese kann dann benutzt werden, um die 3D-Position des Balls bei 8 ms zu triangulieren. Da der Ball sich nicht in einer geraden Linie bewegt, ist diese Position nicht perfekt, aber ausreichend für ein zufriedenstellendes Ergebnis.

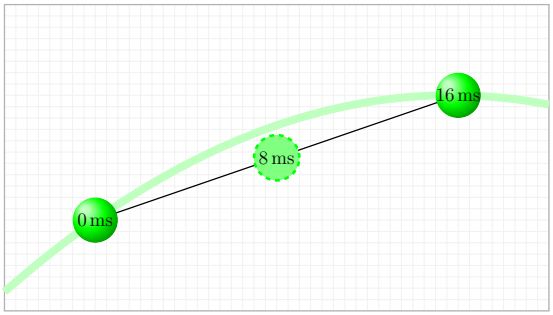


Abbildung 2.23: Interpolierte Ballposition[9]

2.2.7 Multithreading

Moderne Prozessoren können mehrere Programmteile, sogenannte Threads, gleichzeitig ausführen. Dies kommt z.B. dann zum Einsatz, wenn man während des Musikhörens gleichzeitig ein Dokument bearbeitet.
Im Fall vom Trackingsystem ist diese Fähigkeit auch sehr nützlich. Beide Kameras haben ihren eigenen Thread, um die Bilder zu dekomprimieren und zu bearbeiten. Das ist viel schneller, als wenn die Bilder nacheinander bearbeitet werden würden. Die erkannten 2D-Projektionen des Balls auf den Bildern werden dann zu einem weiteren Thread gesendet, der die Flugbahn anhand des letzten Bilds erneuert. Die neue Flugbahn wird dann zu einem letzten Thread weitergereicht, der die Position des Roboterarms neu berechnet und diese dann an ihn übergibt. Dank des Multithreadings müssen diese vier Teile des Programms nie aufeinander warten, und das nächste Bild kann bereits bearbeitet werden, während die Position des Roboterarms noch aktualisiert wird.

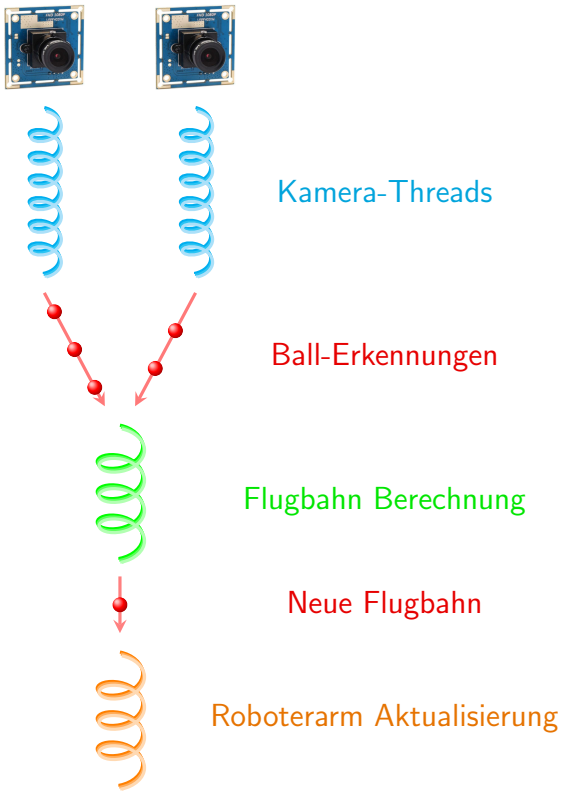


Abbildung 2.24: Thread-Synchronisierung[20]

2.3 Steuerung

2.3.1 System

Das System besteht aus einem Roboterarm, zwei Kameras und einem externen Computer. Die Kameras schicken die Aufnahmen an den Computer, welcher sie auswertet und die neue Position des Roboterarms berechnet. Die gewünschten Motorpositionen werden an den Arduino weitergeleitet, welcher sich in der Basis des Roboterarms befindet. Dieser gibt dann die Befehle an die Motoren weiter. Der Roboterarm selbst besitzt also keinerlei Rechnungsfähigkeit - alles wird durch den externen Computer berechnet und gesteuert.

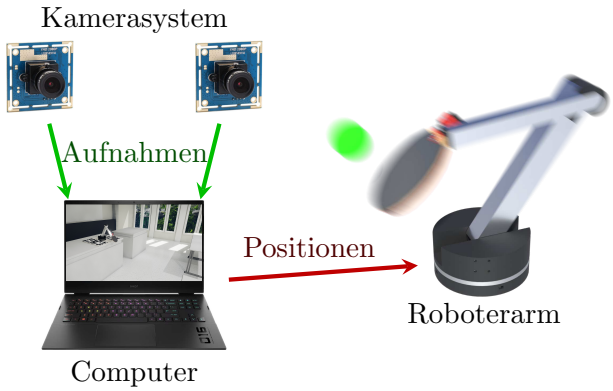


Abbildung 2.25: Schema des Systems[9][4][3][2]

2.3.2 Inverse Kinematik

Um das Ende des Roboterarms an eine gewünschte Position und Ausrichtung zu bringen, müssen die richtigen Gelenkwinkel des Roboters ausgerechnet werden. Dieser Prozess wird inverse Kinematik genannt.[21] Da sich das Ende des Arms immer auf einer Ebene bewegt, die durch den Mittelpunkt der Basis geht, kann diese durch Rotation der Basis in den richtigen Winkel gebracht werden, sodass der Zielpunkt auf ihr liegt. Der Rest der Berechnungen für die Winkel des Arms läuft nur noch auf dieser Ebene ab. Da es sich jetzt lediglich um zwei Dimensionen handelt, sind die nächsten Schritte deutlich vereinfacht worden.

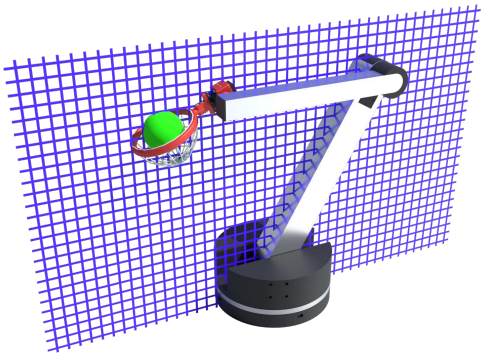


Abbildung 2.26: Arm-Ebene[4]

Die Ausrichtung des Korbs soll dem Einfallswinkel des Balls entsprechen. Jetzt kann vom Abfangpunkt P_3 aus, mit der Korblänge d, der Punkt P_2 berechnet werden. Der Ursprung des Arms befindet sich am Punkt P_0, der Abstand von P_2 dorthin sei c. Die Länge des Ober- und des Unterarms sei a und b. Hier handelt es sich also um ein Dreieck mit drei bekannten Seitenlängen (a, b, c) und drei unbekannten Winkeln (α, β, γ). Dadurch kann man mit dem Kosinussatz die drei unbekannten Winkel berechnen und mit dem arctan den Winkel ε relativ zur Ebene. Der Winkel des Oberarms, relativ zur Basis, ist jetzt $\beta + \varepsilon$, und der des Unterarms, relativ zum Oberarm, ist γ. Der Winkel der Hand, relativ zum Unterarm, entspricht dem Winkel zwischen $\overrightarrow{P_1P_2}$ und $\overrightarrow{P_2P_3}$, also $\varphi = \arccos(\frac{\overrightarrow{P_1P_2} \cdot \overrightarrow{P_2P_3}}{|\overrightarrow{P_1P_2}||\overrightarrow{P_2P_3}|})$.

$$\alpha = \arccos\left(\frac{b^2+c^2-a^2}{2 \cdot b \cdot c}\right)$$

$$\beta = \arccos\left(\frac{a^2+c^2-b^2}{2 \cdot a \cdot c}\right)$$

$$\gamma = \arccos\left(\frac{a^2+b^2-c^2}{2 \cdot a \cdot b}\right)$$

$$\varepsilon = \arctan\left(\frac{P_2.y-P_0.y}{P_2.x-P_0.x}\right)$$

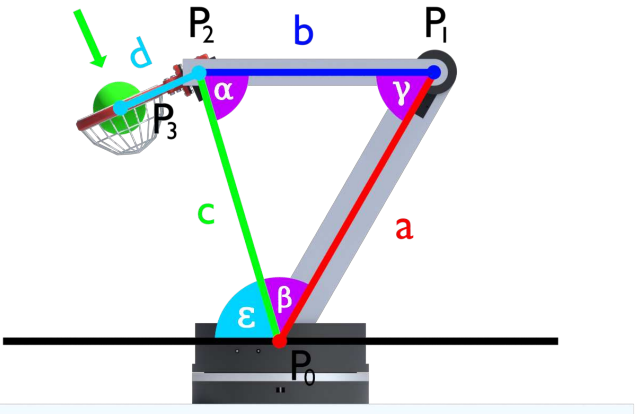


Abbildung 2.27: Armlängen und -winkel[4]

2.3.3 Fangen

Die Zeit, die der Roboterarm braucht, um eine bestimmte Position zu erreichen, ist das Maximum der Zeit, die alle Motoren von der jetzigen Position bis zur neuen Position brauchen. Also in anderen Worten: Der Motor, der von der jetzigen Position am längsten braucht, bestimmt die Gesamtzeit der Bewegungsabfolge. Die Position der Motoren kann dabei mit der inversen Kinematik berechnet werden. Es wird also die Motorenposition für jeden Punkt entlang der Flugbahn ausgerechnet, sowie die Zeit, die der Arm dorthin braucht. Bei der besten Position ist die Differenz zwischen Ankunftszeit des Balls und Ankunftszeit des Roboterarms maximal. Um den Prozess zu optimieren, werden nur die Punkte innerhalb der Reichweite des Roboterarms in Betracht gezogen.

Dafür kann die Reichweite des Arms als Kugel dargestellt werden. Dies sind die Formeln für eine Kugel am Punkt (x_0, y_0, z_0) und mit dem Radius r:

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2$$

Dies ist die Formel für die Flugbahn des Balls:

$$\begin{aligned} x &= p_x + t \cdot v_x \\ y &= p_y + t \cdot v_y \\ z &= a \cdot t^2 + b \cdot t + c \end{aligned}$$

Jetzt können die Werte für x, y, z, welche durch t und den Konstanten definiert sind, in die Formel für die Kugel eingesetzt und nach t aufgelöst werden, um die Schnittpunkte der Parabel mit der Kugel zu bestimmen. Die Anzahl Schnittpunkte entspricht der Anzahl reellen Lösungen für t. Dabei gibt es zwischen 0 und vier Schnittpunkte.

$$(p_x + t \cdot v_x - x_0)^2 + (p_y + t \cdot v_y - y_0)^2 + (a \cdot t^2 + b \cdot t + c - z_0)^2 = r^2$$

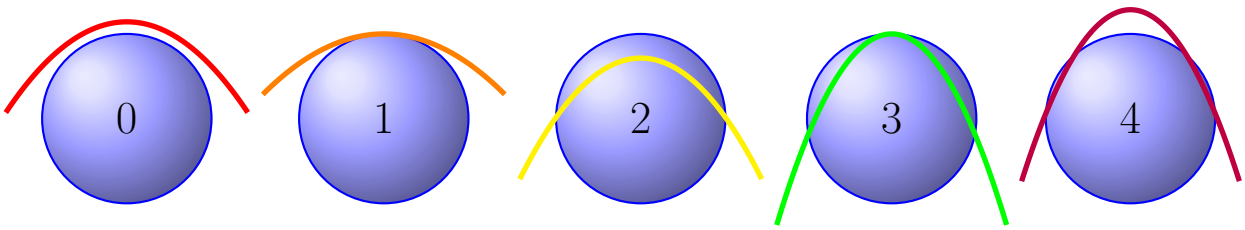


Abbildung 2.28: Kugel-Schnittpunkte[9]

2.3.4 Pingpong-Volleys

Um einen Pingpongball zurückzuspielen, reicht es nicht, nur den Schläger hinzuhalten. Ansonsten würde der Ball zu langsam zurückfliegen - auch muss sich der Arm dem Ball entgegenbewegen. Der Arm wartet etwas unterhalb der Abfangposition, bis der Ball nahe genug ist, und bewegt dann den Schläger nach oben, kurz bevor der Ball ankommt. Der Ball soll, egal wo er abgefangen wird, immer an den gleichen Punkt zurückgespielt werden (1.4 m in Richtung des Spielers und 0.8 m von der Basis aus nach oben).
Die benötigte Geschwindigkeit, um bei einem bestimmten Anfangswinkel das Ziel zu erreichen, kann mit der untenstehenden Formel ausgerechnet werden. Der optimale Winkel befindet sich am Minimum der Funktion, also dort, wo die Ableitung gleich 0 ist. Dieser Punkt wird mithilfe des Gradientenverfahrens gefunden, da die explizite Formel für die Ableitung zu komplex ist.

Abbildung 2.29: Zurückspielen des Balls[4]

$$v(\alpha) = \sqrt{\frac{ax^2}{2\cos(\alpha)^2(y - \tan(\alpha) * x)}},\ v'(\alpha) \stackrel{!}{=} 0$$

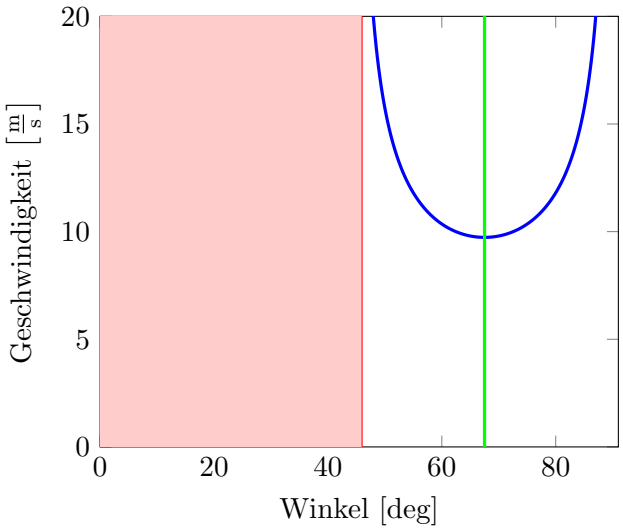


Abbildung 2.30: Für den Punkt (4 m, 4 m) benötigte Geschwindigkeit je nach Winkel[9]

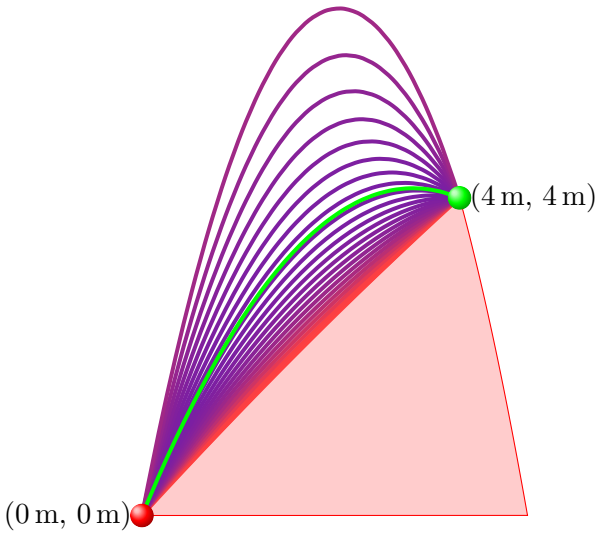


Abbildung 2.31: Verschiedene Flugbahnen, rötere haben eine grössere Anfangsgeschwindigkeit, die grüne ist optimal[9]

2.3.5 Elastischer Stoss

Dies ist die Formel für den elastischen Stoss, mit Vektoren ausgedrückt[22]:

$$\mathbf{v}_1' = \mathbf{v}_1 - \frac{2m_2}{m_1+m_2} \frac{\langle \mathbf{v}_1-\mathbf{v}_2, \mathbf{x}_1-\mathbf{x}_2 \rangle}{\|\mathbf{x}_1-\mathbf{x}_2\|^2} (\mathbf{x}_1-\mathbf{x}_2)$$

Die Winkelklammern $\langle\rangle$ stehen für das Skalarprodukt der Vektoren. Die Formel für die Endgeschwindigkeit des zweiten Objektes ist dieselbe, aber mit vertauschten Indizes.

Im Fall von Pingpong kann man den Term aber deutlich vereinfachen. Die Masse des Balls, relativ zum Schläger, ist vernachlässigbar, man kann also in der Formel annehmen, dass $m_2 = \infty$. Dadurch fällt $\frac{2m_2}{m_1+m_2}$ weg und es bleibt nur noch 2. Für das Ergebnis ist die Länge von $\mathbf{x}_1 - \mathbf{x}_2$ vernachlässigbar, nur die Richtung ist wichtig und in diesem Fall ist diese der Normalvektor des Schlägers. Man kann also in der Formel $\mathbf{x}_1 - \mathbf{x}_2$ mit $\mathbf{N}$ ersetzen. $\|\mathbf{x}_1 - \mathbf{x}_2\|^2$ verschwindet, da die Länge des Normalvektors 1 beträgt. Da das Skalarprodukt distributiv ist, kann man ausserdem das Minus in ein Plus umwandeln und dafür $\mathbf{v}_1 - \mathbf{v}_2$ mit $\mathbf{v}_2 - \mathbf{v}_1$ ersetzen.

$$\mathbf{v}_1' = \mathbf{v}_1 + 2 \langle \mathbf{v}_2 - \mathbf{v}_1, \mathbf{N} \rangle \mathbf{N}$$

Wenn man $\mathbf{v}_1$ auf die andere Seite des Gleichheitszeichens setzt, wird deutlich, dass der Normalvektor die gleiche Richtung wie $\mathbf{v}_1' - \mathbf{v}_1$ hat, da $2 \langle \mathbf{v}_2 - \mathbf{v}_1, \mathbf{N} \rangle$ nur eine Konstante ist. Die Länge des Normalvektors ist aber sowieso 1, nur die Richtung ist wichtig, um den Schläger korrekt auszurichten.

$$\underline{\underline{\mathbf{N} = \frac{\mathbf{v}_1'-\mathbf{v}_1}{\|\mathbf{v}_1'-\mathbf{v}_1\|}}}$$

Jetzt fehlt nur noch die Berechnung der Geschwindigkeit des Schlägers. Wenn beim Skalarprodukt $\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{c}$ einer der beiden Vektoren und der Skalar gegeben sind (bsp. $\mathbf{a}$ und $\mathbf{c}$), gibt es für den zweiten Vektor ($\mathbf{b}$) unendlich viele Lösungen. In diesem Anwendungsfall macht es aber am meisten Sinn, den Vektor zu wählen, bei dem $\mathbf{v}_2$ die kleinste Länge hat, da der Roboterarm somit den Ball am weitesten spielen kann. Da das Skalarprodukt mit $\mathbf{N}$ gebildet wird, wird das Ergebnis am stärksten beeinflusst, wenn sowohl $\mathbf{v}_2$ als auch $\mathbf{N}$ dieselbe Richtung aufweisen.

$$\underline{\underline{\mathbf{v}_2 = \mathbf{N} \left(\frac{\|\mathbf{v}_1'-\mathbf{v}_1\|}{2} - \langle \mathbf{N}, -\mathbf{v}_1 \rangle \right)}}$$

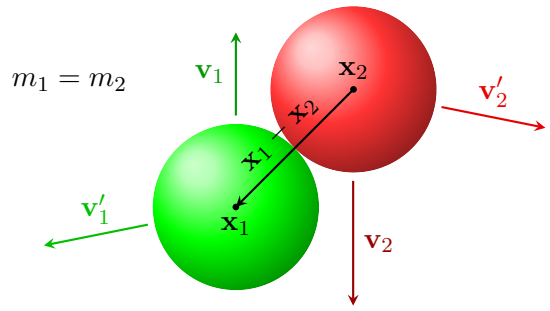


Abbildung 2.32: Elastischer Stoss[9]

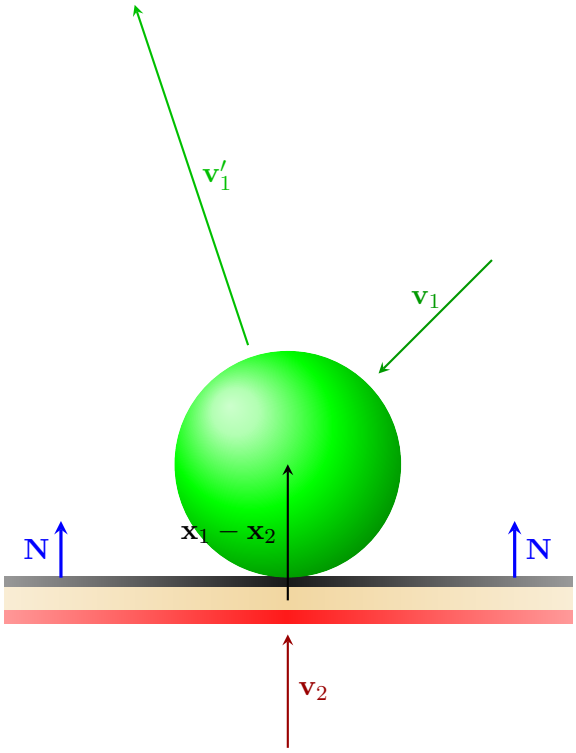


Abbildung 2.33: Elastischer Stoss am Schläger[9]

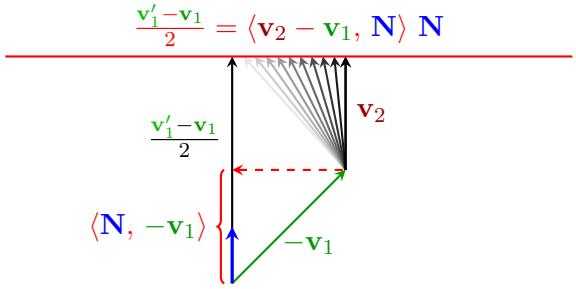


Abbildung 2.34: Optimales $\mathbf{v}_2$[9]

2.3.6 Umsetzung

Jetzt weiss der Roboterarm, wie der Schläger gehalten und in welche Richtung er bewegt werden soll. Die Frage ist nur noch, mit welcher Geschwindigkeit er sich bewegen soll.
In der Realität geht Energie bei einer Kollision verloren, sie ist nie perfekt elastisch. Die Stosszahl k gibt an, wie elastisch ein Stoss ist.[23]

$$k = \frac{\langle \mathbf{N}, \mathbf{v}_2' - \mathbf{v}_1' \rangle}{\langle \mathbf{N}, \mathbf{v}_1 - \mathbf{v}_2 \rangle}$$

- $k = 0 \rightarrow$ komplett plastischer (unelastischer) Stoss
- $k = 1 \rightarrow$ komplett elastischer Stoss

In den Formeln für den elastischen Stoss müsste man überall 2 mit $(k + 1)$ ersetzen, um die korrigierten Werte zu erhalten. Man kann aber jetzt sehen, dass $\mathbf{N}$ nicht von der Stosszahl abhängt und daher die Richtung des effizientesten $\mathbf{v}_2$ ebenso nicht. Schlussendlich wurde einfachheitshalber entschieden, den Arm so schnell wie möglich in Richtung von $\mathbf{v}_2$ zu bewegen und dessen Länge einfach zu vernachlässigen.

Der Arm wartet 15 cm in Richtung von $\mathbf{v}_2$ unter der Abfangposition, bis der Ball 0.2 s von ihr entfernt ist. Sobald der Ball näher als 0.2 s ist, bewegt sich der Arm so schnell wie möglich nach oben und trifft den Ball auf dem Weg.
Wichtig ist aber, dass sich der Schläger auf einer geraden Linie bewegt. Dafür dürfen sich allerdings nicht alle Motoren des Roboterarms mit maximaler Geschwindigkeit bewegen. Der Motor für den Oberarm muss sich normalerweise in diesen 0.2 s viel weniger weit bewegen als der für den Unterarm. Daher kann man nicht einfach die Position der Motoren an der Zielposition ausrechnen und sie dann dorthin bewegen. Der Roboterarm zerlegt stattdessen den Pfad in einige Punkte und visiert diese dann im richtigen Zeitabstand einen nach dem anderen an.
Es stellte sich heraus, dass sich der Arm nicht ausreichend schnell bewegen kann, um den Ball weit genug zurückzuspielen. Damit der Ball zusätzlich an Geschwindigkeit gewinnt, wartet der Schläger leicht nach unten gewinkelt. Beim Aufprall wird er dann schnell nach oben zurückgedreht. Dadurch bewegt sich der Teil des Schlägers, auf den der Ball auftrifft, deutlich schneller.
Nachdem der Ball getroffen wurde, bewegt sich der Roboterarm wieder an die Ausgangsposition in der Mitte zurück, um für den vom Gegenspieler zurückgespielten Ball bereit zu sein.

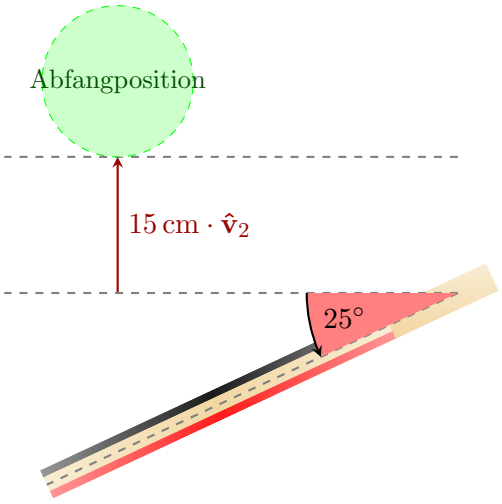


Abbildung 2.35: Schläger Bewegung[9]

2.4 Material und Software

2.4.1 Webcam

Abbildung 2.36: Webcam[2]

Intrinsische Parameter

- Auflösung = 1280 px × 720 px
- Brennweite = 1363 px
- Verzerrungskoeffizienten: {-0.4317, 0.2098, -0.0069, 0.0070}
- Bildfrequenz = 60 fps

2.4.2 Motoren

Basis-Motor

- $\omega \approx 180\,\frac{\text{deg}}{\text{s}}$ (Bei 12 V)
- $M \approx 15\,\text{N m}$ (Bei 12 V)
- Drehbereich: 180° (255°)

Abbildung 2.37: Basis-Motor[24]

Arm-Motor

- $\omega \approx 180\,\frac{\text{deg}}{\text{s}}$ (Bei 12 V)
- $M \approx 15\,\text{N m}$ (Bei 12 V)
- Drehbereich: 255°

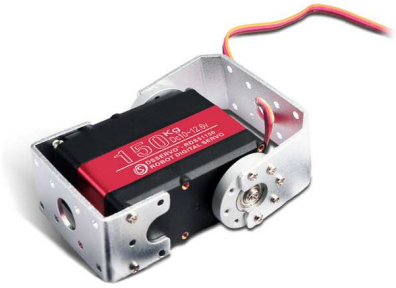


Abbildung 2.38: Arm-Motor[25]

Hand-Motor

- $\omega \approx 420\,\frac{\text{deg}}{\text{s}}$ (Bei 6 V)
- $M \approx 1.5\,\text{N m}$ (Bei 6 V)
- Drehbereich: 180°

Abbildung 2.39: Hand-Motor[26]

2.4.3 Vierkantrohre

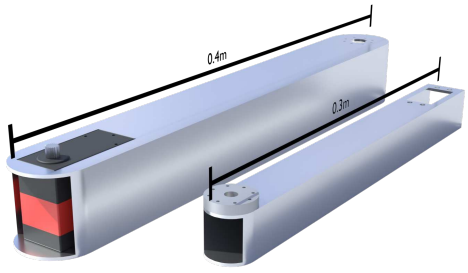


Abbildung 2.40: Rohre[4]

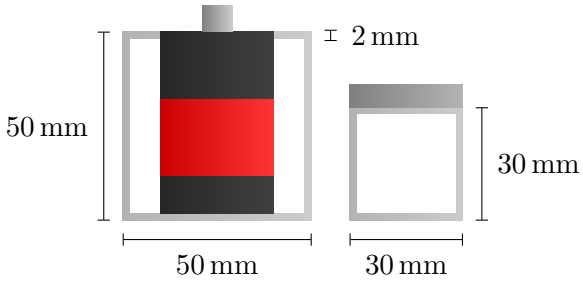


Abbildung 2.41: Querschnitt der Rohre[9]

Die Wahl der Rohre war eine wichtige Entscheidung für den Bau des Roboterarms. Das Material Aluminium wurde wegen seiner günstigen Eigenschaften (einfach zu bearbeiten, erschwinglich und leicht) gewählt. Die Grösse des Oberarms hat sich aus der Höhe des Motors ergeben. Wenn das Rohr eine Höhe von 50 mm hat und eine Wanddicke von 2 mm, passt der Motor mit einer Höhe von 48 mm perfekt hinein und kann auf der Rückseite festgeschraubt werden. Der Unterarm musste nicht ganz so gross sein, da dort ein kleinerer Motor eingebaut wurde. Die runden Servo-Arme haben einen Durchmesser von 30 mm, also wurde die Grösse des Unterarms dementsprechend ausgewählt. Die Längen der Arme wurden eher arbiträr bestimmt. Der Oberarm hat eine Länge von 0.4 m und der Unterarm eine Länge von 0.3 m. Die effektive Länge, also der Abstand zwischen den Gelenken, beträgt jedoch beim Oberarm 0.35 m und beim Unterarm 0.265 m.

2.4.4 OpenCV

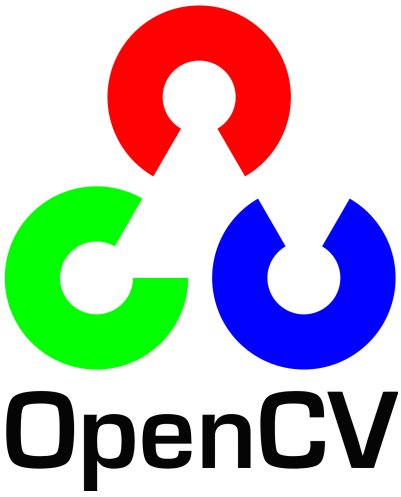


Abbildung 2.42: OpenCV Logo[27]

OpenCV ist eine Open-Source Bibliothek für Computer-Vision maschinelles Lernen. Die Bibliothek bietet eine breite Auswahl an Funktionen und Werkzeugen für Bild- und Videoverarbeitung, Objekterkennung, Merkmalsextraktion, Kamerakalibrierung und vieles mehr. Die Bibliothek umfasst über 2500 optimierte Algorithmen.[28] OpenCV ist in C++ geschrieben und eignete sich daher optimal für mein Projekt. OpenCV-Werkzeuge wurden für mehrere Arbeitsschritte verwendet, wie das Einlesen und Dekomprimieren der Frames der Webcams, die Erkennung des Balls auf den Bildern oder die 3D-Triangulation der Position des Balls.

2.4.5 AprilTags

Abbildung 2.43: Roboter mit AprilTag ausgestattet[29]

AprilTags ist ein visuelles Referenzsystem für die Robotikforschung. Es funktioniert ähnlich wie ein Barcode, speichert eine kleine Menge an Informationen (Tag-ID) und ermöglicht gleichzeitig eine einfache und genaue 6D (x, y, z, roll, pitch, yaw) Posenabschätzung des Tags.[30]

2.4.6 Unreal Engine

Abbildung 2.44: Landschaft in Unreal Engine[31]

Unreal Engine ist eine bekannte Game-Engine, die von Epic Games erstellt wurde und Entwicklern eine breite Auswahl an Tools bereitstellt, um hochwertige und realistische Spiele zu erstellen. Die Engine ist aber auch für andere Anwendungsbereiche gut geeignet, sie findet beispielsweise Einsatz in der Architektur oder Filmindustrie.[32] Da sie in C++ geschrieben ist, war sie leicht zu integrieren. Die Präsentation läuft in der Engine und es kann, wie bei einer PowerPoint-Präsentation, zwischen verschiedenen Szenen geschaltet werden. Einige davon haben einfache LaTeX-Folien, andere haben Animationen und Echtzeit-Visualisierungen vom Tracking-System.

2.4.7 Fusion360

Abbildung 2.45: Fusion 360 Splashscreen[33]

Fusion360 ist ein weit verbreitetes, cloudbasiertes CAD-Programm, das von Autodesk entwickelt wurde. Es wird von Designern und Ingenieuren verwendet, um präzise 3D-Modelle und technische Zeichnungen zu erstellen und deren Verhalten unter Stress zu simulieren. Das Programm enthält auch Funktionen, um Toolpaths und G-Codes für CNC-Maschinen, Laser-Cutter und 3D-Drucker zu erstellen. Im Rahmen dieser Arbeit wurde die Gratis-Version verwendet, welche die 3D-Modell- und Toolpath-Funktionen enthält, um ein 3D-Modell des Roboterarms zu produzieren. Aus diesem 3D-Modell generierte ich dann die G-Codes, um die entsprechenden Teile mit dem 3D-Drucker herzustellen.

2.4.8 Blender

Abbildung 2.46: Blender-Logo[34]

Blender ist eine kostenlose Open-Source 3D-Designsoftware, die zum Entwerfen und Animieren von digitalen Modellen, Figuren und visuellen Effekten verwendet wird. Sie kann auch für die Videobearbeitung und andere Multimedia-Projekte benutzt werden. Sie ist bei Künstlern und Designern aufgrund ihrer Vielseitigkeit sehr beliebt. Blender wurde eingesetzt, um aus dem in Fusion360 erstellten Modell des Roboterarms Abbildungen zu erstellen, die in der Dokumentation sowie der Präsentation vorkommen.

2.4.9 Colmap

Abbildung 2.47: Sparse-Modell des Stadtzentrums von Rom, aus 21K Fotos mit COLMAP erstellt.[35]

COLMAP (`https://colmap.github.io`) ist eine universelle Structure-from-Motion (SfM)- und Multi-View Stereo (MVS)-Pipeline mit einer grafischen und einer Befehlszeilen-Schnittstelle. Sie bietet eine breite Palette von Funktionen für die Rekonstruktion von 3D-Modellen aus Bildsequenzen.[36]
Bei der Erstellung eines 3D-Modells aus einer Bildersequenz kann die Bibliothek nebenbei die intrinsischen Parameter (Brennweite und Verzerrungskoeffizienten) der benutzten Kameras herausfinden. Diese Funktion von COLMAP nutzte ich, um mit einer Bildsequenz, bestehend aus ca. 1000 Bildern, die Kameras zu kalibrieren.

2.5 Multimediale Präsentation in Unreal Engine

Zur verbesserten didaktischen Präsentation der Arbeit und ihrer Ergebnisse wurden einige Teile in räumlichen Modellen und Animationen dargestellt. Im Folgenden sind einige Elemente und deren Entwicklung erläutert.

2.5.1 Physikzimmer-Modell

Das Physikzimmer wurde auf Basis eigener Messungen mithilfe des Programms Blender nachgebaut und zu Unreal Engine exportiert. Die in der Präsentation eingebauten Animationen des Roboterarms und Visualisierungen des Trackingsystems haben somit diesen Raum als Hintergrund.

Abbildung 2.48: Modell des Physikzimmers[37]

2.5.2 Roboterarm-Modell

Das Modell des Roboterarms ist dort im Zimmer platziert, wo dieser auch während der Präsentation aufgestellt sein wird. Das Modell war bereits in Fusion 360 erstellt, musste also nur noch in Unreal Engine importiert werden. Das Modell war bei der Fehlersuche ausgesprochen hilfreich, da Bewegungsabläufe bereits an diesem ausgetestet werden konnten. Ausserdem kann bei einer Positionsänderung überprüft werden, ob Teile des Roboterarms mit Objekten zusammenstossen würden. Sollte dies der Fall sein, würde die Position nicht auf den Roboterarm übernommen werden.

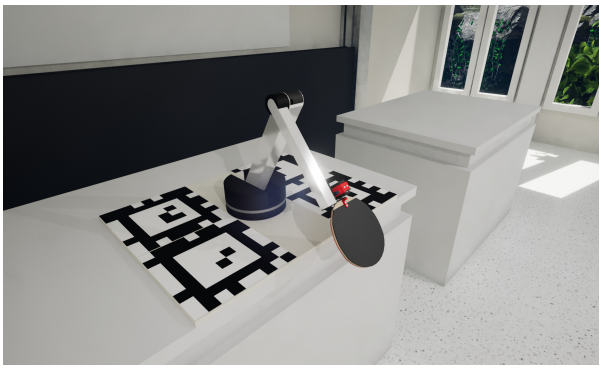


Abbildung 2.49: Modell des Roboterarms[37]

2.5.3 Präsentation und Animationen

Unreal Engine hat ein Szenen-System, welches es ermöglicht, eine Hauptszene mit dem Physikzimmer zu erstellen und dann einige Unterszenen mit dem Roboterarm, sowie den Animationen und Visualisierungen. Beim Drücken der PgUp / PgDn Tasten kann die nächste oder vorherige Unterszene geladen und die aktuell angezeigte entfernt werden. Sollte als nächstes eine Folie gezeigt werden, wird nur die alte Szene entfernt und die Folie auf dem Bildschirm gezeigt. Bei Animationen passiert dasselbe. Sie sind in Abschnitte eingeteilt und werden am Ende eines Abschnitts pausiert. Dann kann entweder die → Taste gedrückt werden, um die Animation weiterlaufen zu lassen, oder die ← Taste, um den letzten Abschnitt der Animation wieder rückwärts abzuspielen.

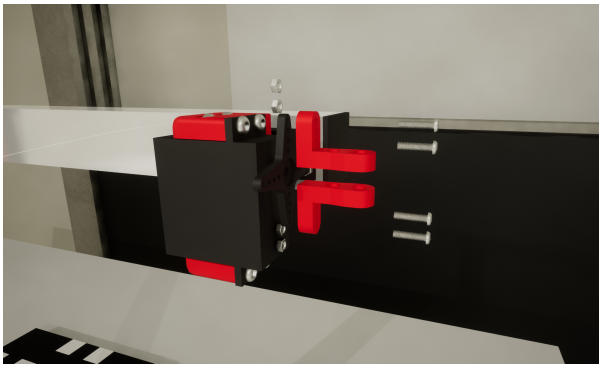


Abbildung 2.50: Animierter Aufbau der Hand[37]

2.5.4 Kamera-Vorschau

In Unreal-Engine werden beide Kameras genau dort angezeigt, wo sie sich auch in Realität befinden. Wenn der Benutzer eine der Kameras auswählt, wird eine Vorschau davon abgespielt, was die Kamera momentan aufnimmt. Dabei kann man sich die HSV (Hue, Saturation, Value) Kanäle des Bilds einzeln anzeigen lassen. Farb- und Hintergrundfilter können ebenfalls an- oder ausgeschaltet werden.

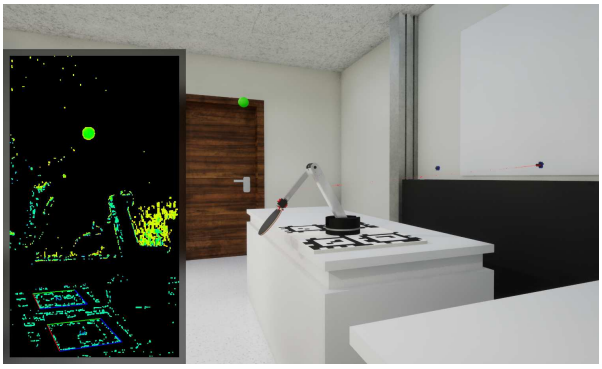


Abbildung 2.51: Kamera-Vorschau in Unreal Engine[37]

2.5.5 Elektronik-Visualisierung

Um die Elektronik im Basisgehäuse zeigen zu können, waren zwei Dinge notwendig: die Modelle der Komponenten sowie die Kabel. Die ersteren waren online erhältlich und konnten nach einigen kleineren Modifikationen in Unreal Engine importiert werden. Die Kabel anzeigen zu lassen, war etwas arbeitsintensiver. Glücklicherweise hat aber Unreal Engine ein Spline-System, welches es ermöglicht, 3D-Modelle entlang Bézierkurven zu verzerren. Die Endpunkte der Kurve werden mit jedem Frame aktualisiert, damit sie sich am richtigen Ort befinden, selbst wenn sich die Elektronik bewegt.

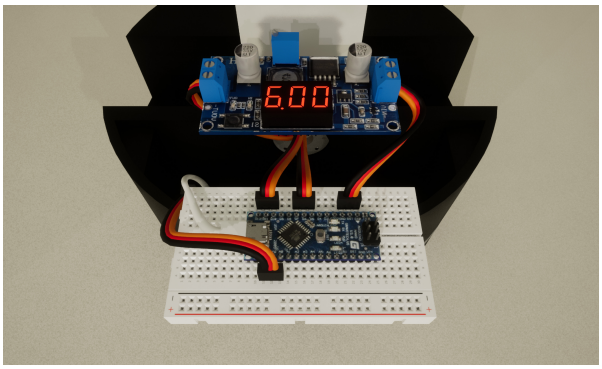


Abbildung 2.52: Elektronik der Basis[37]

Kapitel 3

Ergebnisse

3.1 Messungen

3.1.1 Kamerasystem

Um die Genauigkeit des Kamerasystems und die Präzision der Vorhersage der Flugbahn zu veranschaulichen, kann die erwartete Fangposition im Verlaufe eines Wurfs und ihre Veränderung dargestellt werden. Die Fangposition wird jedes Mal aktualisiert, sobald eine neue Ball-Position vom Kamerasystem erkannt wird. In der Grafik rechts wurde die Abweichung der Fangpositionen zu der schlussendlich verwendeten, zusammen mit dem jeweiligen Zeitpunkt dargestellt. Wie man sehen kann, ist der Abstand der Position ca. 0.1 s nach Abwurf etwa 5 cm. Das heisst, der Roboterarm kann bereits zu diesem Zeitpunkt in diese Richtung bewegt werden. In den letzten 0.2 s vor dem Aufprall ist die Abweichung bereits unter 1 cm und die Position des Arms kann noch verfeinert werden.

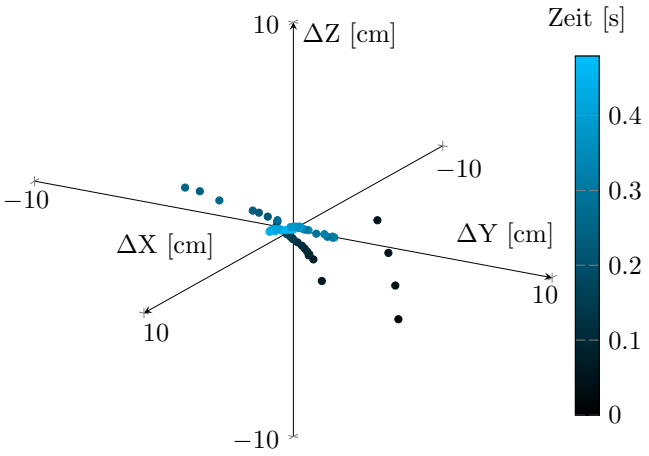


Abbildung 3.1: Abweichung der Fangposition im Verlaufe eines Ballwurfs[9]

3.1.2 Pingpongspiel

Um die Pingpong-Fähigkeiten des Roboterarms zu testen, wurde eine Messung der Länge einzelner Ballwechsel durchgeführt. Es wurde dabei gezählt, wie oft der Roboterarm den Ball berührt. Hier wurde der Pingpongball dem Roboterarm aus einer Distanz von 2 m zugeworfen und dann, ohne die Füsse zu bewegen, wurde der Ballwechsel so lange wie möglich fortgeführt. Dabei wurde dem Roboterarm immer an variierenden Stellen zugespielt. Es wurden 16 aufeinanderfolgende Ballwechsel durchgeführt und ihre Länge dokumentiert. Der Roboterarm schaffte durchschnittlich 9.75 Ballberührungen, was einer Rückspielquote von 89.7% entspricht.

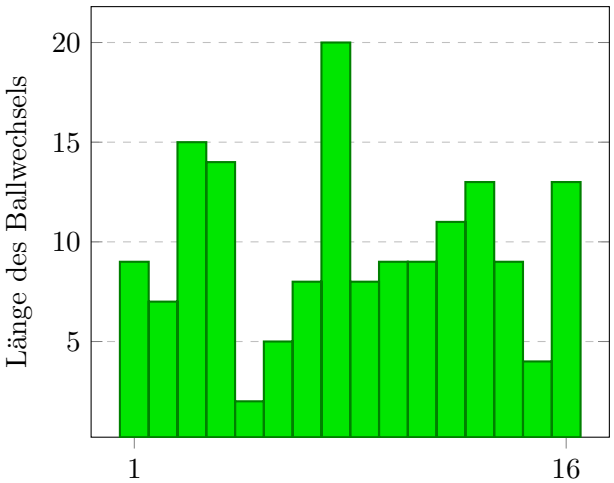


Abbildung 3.2: Aufeinanderfolgende Ballwechsel und ihre Länge[9]

3.1.3 Fangfähigkeiten

Um die Fang-Fähigkeiten des Roboterarms zu beziffern, wurde eine Messung der Fangquote durchgeführt. Dabei warf man dem Roboterarm einen Schaumstoffball mit einem Durchmesser von 6 cm aus einer Distanz von 2 m zu. Der Durchmesser des Fangrings betrug dabei 9 cm. Die Würfe wurden in drei Gruppen unterteilt, basierend darauf, wie weit der Roboterarm sich bewegen musste. Die Fangquote wurde in diesen drei Kategorien separat dokumentiert. Ab einer Basisdrehung von mehr als 30° gehört der Wurf zu der entsprechenden linken oder rechten äusseren Kategorie. Es wurden 30 Würfe pro Kategorie durchgeführt. Dabei ist nicht überraschend, dass die Fangquote in der mittleren Kategorie leicht höher ist, da sich der Roboterarm hier nicht so weit bewegen musste.

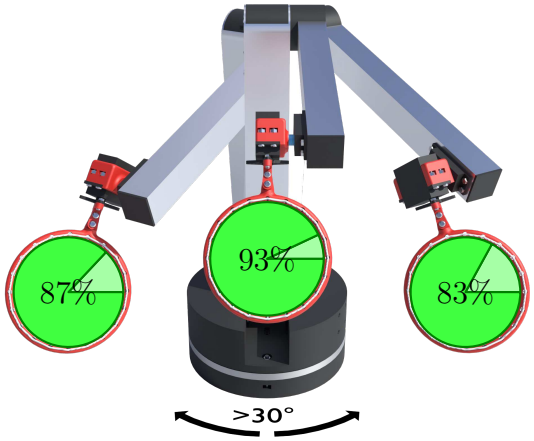


Abbildung 3.3: Fangquote in den Kategorien[9][4]

3.2 Produktentwicklung und Erreichen der Leitfragen

Wie an den dargestellten Ergebnissen zu sehen ist, wurden der Roboterarm und das Kamerasystem erfolgreich entwickelt und die in den Leitfragen gesetzten Ziele erreicht.

Der Roboterarm kann mit guter Wiederholbarkeit lange Ballwechsel aufrecht erhalten. Die Messungen werden jedoch auch durch die Spielart und Spielfähigkeit des Gegenspielers beeinflusst.

Auch beim Fangen eines Balls mit einem Netz konnte eine gute Fangquote erzielt werden.

Das Kamerasystem ermöglicht dies mit der Flugbahnvorhersage, die auch schon nach nur wenigen Messungen eine gute Abschätzung der Fangposition liefert, die dann noch mit der Zeit verbessert wird.

3.3 Reflexion

Der Roboterarm wurde zunächst nicht ausschliesslich für das Pingpongspiel entwickelt. Beim ersten Entwurf des Roboterarms konnte sich die Basis beliebig weit drehen, was die Kabelführung eher verkomplizierte. Auch jetzt hat der Roboterarm noch eine recht grosse Range of Motion, welche nicht notwendig wäre. So kann er auch Punkte hinter sich erreichen.

Bei einigen Bauelementen des Roboterarms wurde zu sehr auf die Ästhetik geachtet, was sich für die Funktionalität als eher nachteilig erwies. Beispielsweise wurde das Rohr für den Oberarm in einer Grösse gewählt, wo der Motor für den Unterarm perfekt hineinpasst und so optisch deutlich weniger sichtbar ist. Auch Unreal Engine wurde teilweise aus ästhetischen Gründen gewählt, da das Programm einige sehr gutaussehende Render-Features hat. Andere Engines wären jedoch effizienter gewesen, und darüber hinaus hätten sich eigene Bibliotheken leichter integrieren lassen.

Verschiedene Testfunktionen, wie beispielsweise die Aufnahme der letzten Flugbahn und die dazu ausgeführten Roboterarmbewegungen, wurden erst gegen Ende des Entwicklungsprozesses programmiert. Diese Funktionen hätten aber schon vorher das Optimieren des Roboterarms deutlich vereinfacht und im Nachhinein hätte es eine Priorität sein sollen, diese so früh wie möglich zu implementieren.

3.4 Weiterentwicklung

Die im Rahmen dieser Maturaarbeit entwickelte Version eines Roboterarm- und Kamerasystems konnte alle definierten Ziele erreichen. Gleichzeitig ergeben sich zahlreiche Ideen und Möglichkeiten der Weiterentwicklung des Systems:

- Den Roboterarm explizit für das Pingpongspielen weiter zu optimieren, auch für einen Einsatz an der Tischtennisplatte.
- Erkennen der Position des Gegenspielers und damit Ermöglichung variabel definierter Ziele zum Anspielen.
- Automatische Erkennung der Abweichung des retournierten Balls relativ zum Ziel mithilfe der Kameras.

3.5 Kosten

Die Materialkosten des Produkts halten sich mit CHF 300 in Grenzen (ohne Computer für die Steuerung). Die Motoren und Webcams, welche für zukünftige Projekte wiederverwendet werden können, machen dabei über 70% der Materialkosten aus.

Objekt	**Preis [CHF]**
ELP-Kamera × 2	85.70
Basis Servo Motor	38.73
Arm Servo Motor × 2	73.12
Hand Servo Motor × 2	10.10
Metallrohre	20.00
Drehscheibe	16.56
Basis Servo Arm	2.02
Buck Converter	0.73
Arduino Nano (AliExpress)	3.46
Kabel	7.82
3D-Drucker Filament	14.78
Pressspanplatte	11.98
Pingpong Schläger	5.02
Total	**290.02**

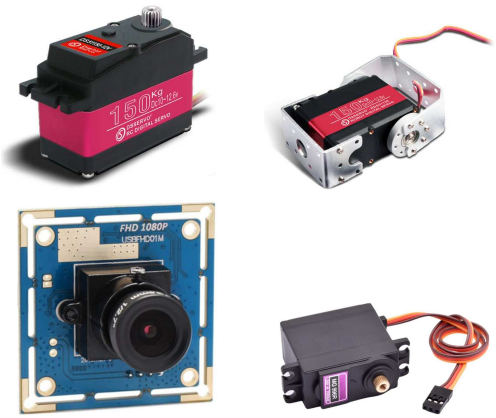


Abbildung 3.4: Übersicht der wichtigsten Bauelemente[24] [25] [2] [38]

3.6 Öffentliche Rezeption

Abbildung 3.5: Twitter Post des Arduino Accounts[39]

Um die öffentliche Rezeption zu testen, wurde ein Video des Roboterarms auf Reddit hochgeladen.[40] Obwohl es eine erste Version der Software war und der Roboterarm zu diesem Zeitpunkt nur 10 Ballwechsel erreichte, erzielte das Video ca. 40k Aufrufe und ca. 700 Upvotes, und das nach nur einer Woche. In der Beschreibung des Videos wurde erwähnt, dass ein Arduino zur Steuerung der Motoren des Roboterarms eingesetzt wurde. Daraufhin postete das offizielle Arduino-Konto das Video auf Twitter. Da die Firma natürlich eine deutlich grössere Reichweite hat, erreichte das Video nach einer Woche etwa 225k Aufrufe und ca. 2000 Likes. Der Arduino-Account stellt sonst auch Videos anderer Arduino-Projekte online, diese bekommen aber durchschnittlich deutlich weniger Aufrufe, typischerweise nur etwa 20k. Demnach erfuhr das Projekt in den sozialen Medien eine äusserst positive Resonanz.

Kapitel 4

Anhang - Systemanalyse

4.1 Design des Roboterarms

Für das Basisdesign wurde ein klassischer Roboterarm-Aufbau gewählt, da zu Beginn des Maturaarbeitsprojekts verschiedene Anwendungsmöglichkeiten des Robterarms in Erwägung gezogen wurden, beispielsweise das Sortieren von Objekten auf einem Tisch. Der Roboterarm sollte daher flexibel eingesetzt werden können. Erst im Verlauf des Projekts wurde die Arbeit auf das Tischtennisspielen fokussiert. Durch das klassische Design des Roboterarms ergeben sich für das Tischtennisspielen gewisse Limitationen, da es für das Spielen auf einer Tischtennisplatte nicht geeignet erscheint. Dafür wäre eher eine freihängende Konstruktion notwendig wie in 1.4.3 zu sehen. Es wurde deshalb die Entscheidung gefällt, das Tischtennisspielen auf das freie Volley Spielen zu beschränken, da sich dabei alle wesentlichen Anforderungen und Funktionen ebenfalls gut testen lassen.

Abbildung 4.1: Arm-Design[4]

4.2 Wahl der Motoren

Aufgrund der Anforderungen, welche sich aus der Anwendung im dynamischen Anwendungsbereich wie dem Tischtennisspielen ergeben, wurden als Motoren Servomotoren für den Roboterarm gewählt. Diese sind im Gegensatz zu Alternativen wie z.B. Schrittmotoren zwar ungenauer, dafür aber für den gleichen Preis schneller und stärker, sowie leichter zu steuern. Die gewählten Servomotoren erwiesen sich als passend für die gestellte Aufgabe. Es hat sich gezeigt, dass der Roboterarm eine hohe Dynamik beim Spiel erreichen kann. Die Bewegungen sind schnell und doch ausreichend genau für das Spiel. Gleichzeitig ergab sich im Verlaufe der Zeit ein gewisser Verschleiss der Motoren, was sich insbesondere an den inneren Zahnrädern zeigte. Für ein vollentwickeltes Produkt müssten die Motoren verbessert werden und auf eine längere Lebensdauer optimiert werden.

Abbildung 4.2: Motor des Unter- und Oberarmes[25]

4.3 Vernachlässigung des Trägheitsmoments

Beim Bewegen des Roboterarmes von einer Position zu einer anderen wären theoretisch mehrere Bewegungsabläufe möglich. Bei der letztlich gewählten Vorgehensweise werden einfach alle Motoren so schnell wie möglich zur neuen Position hin bewegt. Es könnte aber auch schneller sein, den Arm zuerst etwas einzuziehen, um sein Trägheitsmoment zu verringern, danach die Basis zu bewegen, und erst dann den Rest des Arms zu strecken und an die neue Position zu bewegen. Um die Auswirkung der Armstellung auf die Bewegungzeit der Basis zu messen, wurden bei einer 180°-Drehung die Geschwindigkeit im Laufe der Zeit betrachtet. Dafür wurde ein Streifen Papier mit einem Schwarz-Weiss Muster auf die Basis geklebt. Dann konnte mit einem Lichtsensor und einer LED, welche beide an einen Raspberry-Pi Pico Mikrocontroller angeschlossen wurden, die Zeit zwischen den Helligkeitsänderungen gemessen werden und aus diesen Messungen die Winkelgeschwindigkeit der Basis berechnet werden. Die Messung wurde dann sowohl im Uhrzeigersinn als auch im Gegenuhrzeigersinn für drei Arm-Stellungen durchgeführt. Einmal war der Arm in gestreckter und vollkommen vertikaler Stellung, dem optimalen Fall. Dann war der Arm in gesteckter und vollkommen horizontaler Stellung, dem ungünstigsten Fall. Eine letzte Messung wurde dann durchgeführt, bei welcher der Arm in der Ruheposition, in welcher der Arm beim Pingpongspielen wartet, gedreht wurde. Bei dieser hat der untere Arm einen Winkel von 60° zur Horizontalen und der obere Arm zum unteren einen von 70°.

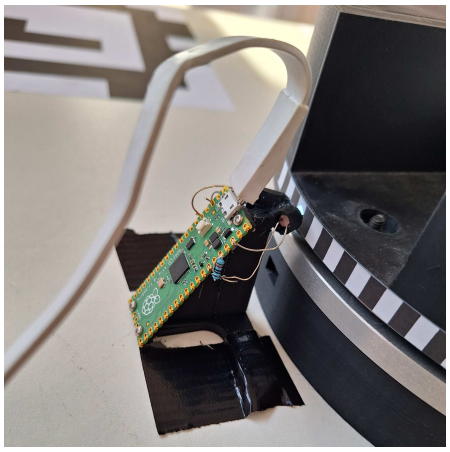


Abbildung 4.3: Versuchsaufbau[11]

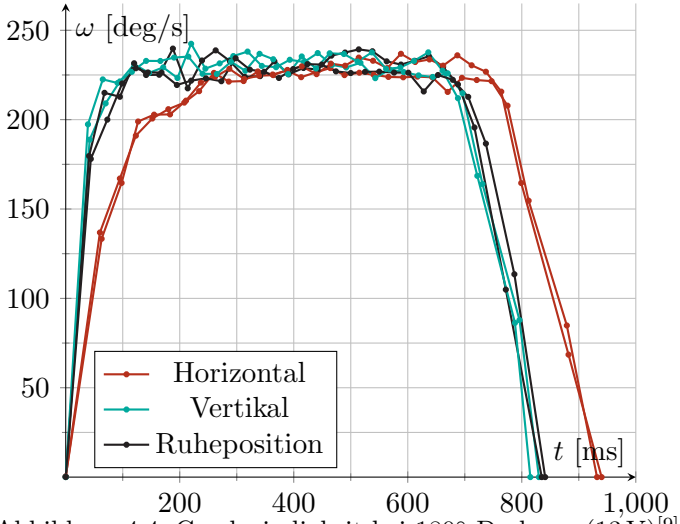


Abbildung 4.4: Geschwindigkeit bei 180° Drehung (13 V)[9]

Abbildung 4.5: Arm-Ausrichtungen: Vertikal, Horizontal, Ruheposition[4]

Die Frage, ob es beim Bewegen des Roboterarms besser ist, den Arm zunächst etwas einzuziehen, zu drehen, und danach wieder auszufahren, ist somit geklärt. Der geringe Zeitunterschied von 0.1 s im ungünstigsten Fall deutet darauf hin, dass es besser ist, selbst die gestreckte horizontale Stellung zu belassen, da das Einziehen und Ausfahren wohl mehr Zeit in Anspruch nehmen würde. Im realistischeren Fall, wo der Arm am Anfang in der gewinkelten Ruheposition ist, ist der Zeitunterschied praktisch ganz vernachlässigbar. Meine Annahme der gleichzeitigen direkten Anpassung von Streckung und Rotation scheint daher die optimale zu sein.

4.4 Interpolation der Ball-Erkennungen

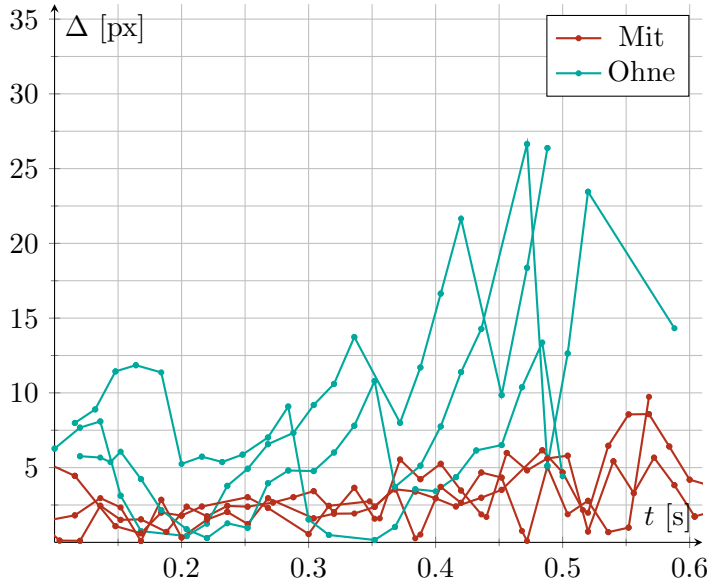


Abbildung 4.6: Reprojektionsfehler mit und ohne Interpolation[9]

Wie im Kapitel 2.2.4 der Arbeit dargestellt, wurde die Position des Balls durch Triangulation der Kameraaufnahmen bestimmt. Zur Beurteilung der Genauigkeit und Qualität dieser Positionsbestimmung des Balls wird die bestimmte 3D-Position auf die beiden Kamerabilder zurück projiziert und mit den ursprünglichen 2D-Positionen verglichen.
Um zu veranschaulichen, wie viel die Interpolation der Ball-Erkennungen (Kapitel 2.2.6) beim Tracken des Balles ausmacht, wurden die Reprojektionsfehler im Verlaufe der Zeit betrachtet. Wie in der Grafik zu sehen, ist der Fehler deutlich grösser wenn nicht interpoliert wurde. Die zeitlichen Schwankungen können damit erklärt werden, dass während der Flugbahn der Ball auf dem Bild sich verschieden schnell bewegt. In einer weiterentwickelten Version des Systems könnten Kameras mit einem externen Auslöser verwendet werden, welche somit synchronisiert aufnehmen könnten. Diese sind aber in der Regel deutlich teurer, weshalb sie hier nicht verwendet wurden.

4.5 Einfluss der Versorgungspannung auf die Motorgeschwindigkeit

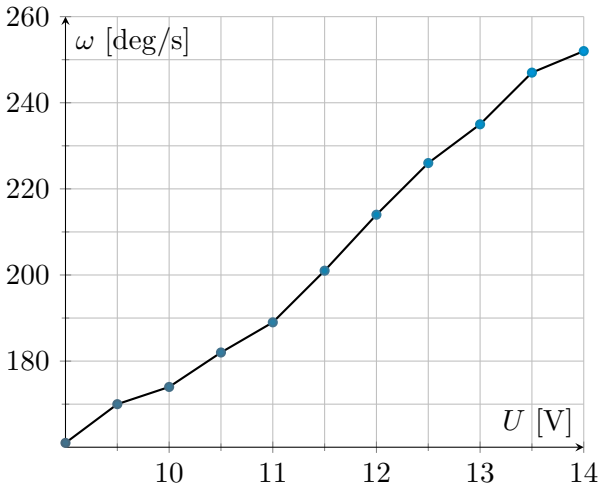


Abbildung 4.7: Abhängigkeit zwischen Spannung und durchschnittlicher Basisgeschwindigkeit[9]

Um den Einfluss der Versorgungspannung auf die Motorgeschwindigkeit zu messen, wurde der gleiche Versuchsaufbau wie in Abschnitt 4.3 verwendet. Die Basis des Roboterarmes wurde bei jeder Versorgungsspannung beschleunigt und dann dessen durchschnittliche Drehgeschwindigkeit gemessen. Dabei war der Arm in vertikaler Stellung. Dies wurde bei jeder Versorgungsspannung wiederholt.
Es zeigt sich eine proportionale Abhängigkeit der Drehgeschwindigkeit von der Versorgungsspannung. Eine adequate Kontrolle der Versorgungsspannung ist für die meisten Anwendungen also wichtig.

4.6 Zuverlässigkeit der Stromquelle

Da die Motorleistung bei geringerer Versorgungsspannung abnimmt, sollte die Stromquelle diese möglichst Konstant halten, auch unter Last. Um also dessen Zuverlässigkeit zu testen, habe ich einen Raspberry-Pi Pico Mikrocontroller Parallel mit dem Roboterarm an die Stromquelle angeschlossen. Da aber der Mikrocontroller mit 3.3 V betrieben wird, die Stromquelle dem Roboterarm jedoch bis zu 13 V liefert, musste die Spannung mit einigen Widerständen, welche einen Spannungsteiler bildeten, verringert werden. Der Mikrocontroller konnte somit die verringerte Spannung jede Millisekunde messen, diese wieder auf die originale Spannung hochrechnen, und dann an den Computer schicken.

Die Spannung wurde während einer 180° Drehung der Basis gemessen, während der Arm in der angewinkelten Ruheposition war (drei Messreihen). Die vom Mikrocontroller gelieferte Daten sind unten in der Abbildung zusammen mit der entsprechenden Winkelgeschwindigkeit zu dem Zeitpunkt zu sehen. Dabei war die Stromquelle auf 13 V eingestellt. Wie zu sehen, geht die Spannung ca. 0.1 s lang mehr als 1 V unter die Zielspannung. Während der Drehbewegung bleibt sie auch etwas niedriger. Wenn der Motor dann wieder "bremst", steigt die Spannung sogar ein halbes Volt über die Zielspannung.

Diese Spannungsunterschiede verursachen wahrscheinlich eine leicht längere Beschleunigungszeit des Arms. Auch noch zu beachten ist, falls ein weiteres Gerät an diese Stromquelle angeschlossen wäre, müsste dieses mit leichten Spannungsschwankungen rechnen. Diese Effekte könnten wahrscheinlich mit einem Kondensator ausreichender Kapazität oder einer besseren Stromquelle verhindert werden.

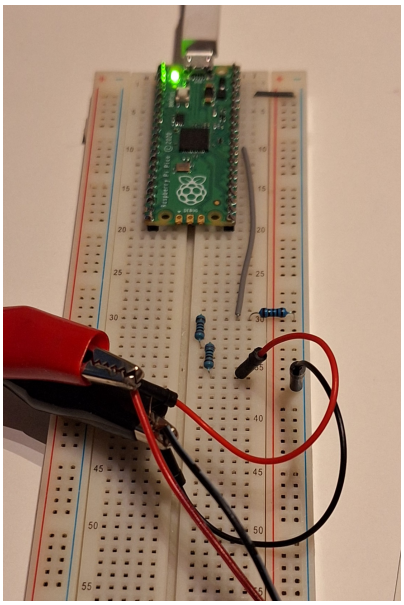


Abbildung 4.8: Versuchsaufbau[11]

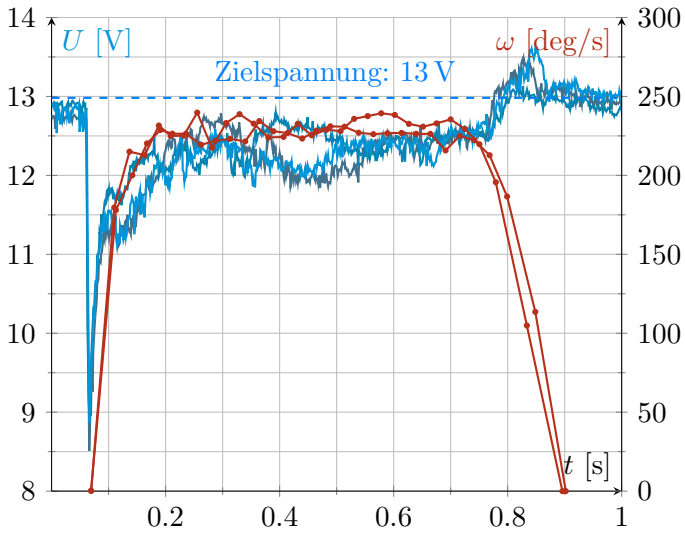


Abbildung 4.9: Spannung der Stromquelle bei 180° Drehung (3 Messreihen in blau/grün) zusammen mit der Winkelgeschwindigkeit (2 Messreihen in rot) im Zeitverlauf.[9]

Literaturverzeichnis

[1] *Swiss Olympiad in Informatics*. URL: https://soi.ch/ (besucht am 28. 09. 2023).

[5] Magnus Linderoth. "On Robotic Work-Space Sensing and Control". Department of Automatic Control, Lund Institute of Technology, 2013. URL: https://lup.lub.lu.se/search/files/3819777/4124507.pdf.

[6] Jonas Tebbe u. a. *Table Tennis Robot — University of Tübingen*. URL: https://uni-tuebingen.de/en/132097 (besucht am 29. 09. 2023).

[7] *FORPHEUS — Technology — OMRON Global*. URL: https://www.omron.com/global/en/technology/information/forpheus/ (besucht am 29. 09. 2023).

[12] Wikipedia. *Lochkamera — Wikipedia, die freie Enzyklopädie*. URL: https://de.wikipedia.org/w/index.php?title=Lochkamera&oldid=226076150 (besucht am 01. 10. 2023).

[14] Wikipedia. *Raytracing — Wikipedia, die freie Enzyklopädie*. URL: https://de.wikipedia.org/w/index.php?title=Raytracing&oldid=236313842 (besucht am 01. 10. 2023).

[15] Wikipedia. *Kamerakalibrierung — Wikipedia, die freie Enzyklopädie*. URL: https://de.wikipedia.org/w/index.php?title=Kamerakalibrierung&oldid=232703876 (besucht am 01. 10. 2023).

[16] *Verzeichnung — Wikipedia, die freie Enzyklopädie*. URL: https://de.wikipedia.org/w/index.php?title=Verzeichnung&oldid=230628646 (besucht am 19. 06. 2023).

[18] Wikipedia. *Triangulation (computer vision) — Wikipedia, The Free Encyclopedia*. URL: https://en.wikipedia.org/w/index.php?title=Triangulation_(computer_vision)&oldid=1155138545 (besucht am 01. 10. 2023).

[19] *Methode der kleinsten Quadrate — Wikipedia, die freie Enzyklopädie*. URL: https://de.wikipedia.org/w/index.php?title=Methode_der_kleinsten_Quadrate&oldid=233758126 (besucht am 20. 05. 2023).

[21] Wikipedia. *Inverse Kinematik — Wikipedia, die freie Enzyklopädie*. 2022. URL: https://de.wikipedia.org/w/index.php?title=Inverse_Kinematik&oldid=223316443 (besucht am 02. 05. 2023).

[22] Wikipedia contributors. *Elastic collision — Wikipedia, The Free Encyclopedia*. URL: https://en.wikipedia.org/w/index.php?title=Elastic_collision&oldid=1142868158 (besucht am 22. 04. 2023).

[23] Wikipedia contributors. *Inelastic collision — Wikipedia, The Free Encyclopedia*. URL: https://en.wikipedia.org/w/index.php?title=Inelastic_collision&oldid=1137666995 (besucht am 22. 04. 2023).

[28] *OpenCV - Open Computer Vision Library*. URL: https://opencv.org/ (besucht am 01. 06. 2023).

[30] *AprilRobotics/apriltag: AprilTag is a visual fiducial system popular for robotics research*. 2018. URL: https://roboticsknowledgebase.com/wiki/sensing/apriltags/ (besucht am 23. 03. 2023).

[32] *The most powerful real-time 3D creation tool - Unreal Engine*. URL: https://unrealengine.com (besucht am 01. 10. 2023).

[36] Johannes L. Schoenberger. *COLMAP – COLMAP 3.8-dev documentation*. URL: https://colmap.github.io/ (besucht am 21. 03. 2023).

[40] *Robot Arm that can play ping-pong and catch balls : robotics.* URL: https://www.reddit.com/r/robotics/comments/13j6hm7/robot_arm_that_can_play_pingpong_and_catch_balls/ (besucht am 22. 05. 2023).

Bildverzeichnis

[2] *Full HD USB Camera Module 1080P USB2.0 OV2710 Color Sensor Support MJPEG with 3.6MM Lens [ELP-USBFHD01M-L36]...* URL: https://http://www.elpcctv.com/full-hd-usb-camera-module-1080p-usb20-ov2710-color-sensor-support-mjpeg-with-36mm-lens-p-203.html (besucht am 08. 03. 2023).

[3] *HP Omen 16-c0850nz RTX 3070 (16.10", AMD Ryzen 7 5800H, 16 GB, 1000 GB, CH) - digitec.* URL: https://www.digitec.ch/en/s1/product/hp-omen-16-c0850nz-rtx-3070-1610-amd-ryzen-7-5800h-16-gb-1000-gb-ch-notebooks-16087341 (besucht am 01. 10. 2023).

[4] Elias Bauer. *Selbsterstellt mit Blender.* 2023.

[8] *FORPHEUS — Technology — OMRON Global.* URL: https://www.omron.com/global/en/technology/information/forpheus/ (besucht am 29. 09. 2023).

[9] Elias Bauer. *Selbsterstellt mit TikZ.* 2023.

[10] *Creality Ender 5 Pro — All the Best Features of 2019.* URL: https://www.diyelectronics.co.za/store/printers/2897-creality-ender-5-pro-3d-printer.html (besucht am 26. 04. 2023).

[11] Elias Bauer. *Selbsterstellt.* 2023.

[13] *Pinhole camera - Wikipedia.* URL: https://en.wikipedia.org/wiki/Pinhole_camera#/media/File:Pinhole-camera.svg (besucht am 16. 03. 2023).

[17] *A retro living room fit for today - IKEA Switzerland.* URL: https://www.ikea.com/ch/en/rooms/living-room/gallery/a-mid-century-modern-living-room-for-modern-times-pubbc584e70 (besucht am 08. 03. 2023).

[20] *Selbsterstellt mit TikZ, beinhaltet Bild von ELP-Kamera.* URL: https://http://www.elpcctv.com/full-hd-usb-camera-module-1080p-usb20-ov2710-color-sensor-support-mjpeg-with-36mm-lens-p-203.html (besucht am 08. 03. 2023).

[24] *Baja hohe drehmoment servo 60kg DS5160 HV Digital Servo für 1/5 Redcat HPI Baja 5B SS RC servo Auto kompatibel SAVOX 0236 LOSI XL 5T—Teile & Zubehör— - AliExpress.* URL: https://de.aliexpress.com/item/32921844198.html?spm=a2g0o.order_list.order_list_main.103.21ef5c5f8iZWFC (besucht am 08. 03. 2023).

[25] *1 stücke servo 60kg hohe drehmoment Roboter servo RDS5160 SSG für roboter DIY digitale servo arduino servo grosse servo—servo accessories—servo discservo motor cnc machine - AliExpress.* URL: https://de.aliexpress.com/item/725115447.html?spm=a2g0o.productlist.main.3.49811b04aD6Ce5 (besucht am 08. 03. 2023).

[26] *Ba180/360 grad SG90 9g Micro Servos für RC Roboter Hubschrauber Flugzeug Steuert Auto Boot— — - AliExpress.* URL: https://de.aliexpress.com/item/1005004573642720.html?spm=a2g0o.productlist.main.3.60139513VzX3q (besucht am 08. 03. 2023).

[27] *OpenCV - Wikipedia.* URL: https://en.wikipedia.org/wiki/OpenCV (besucht am 19. 03. 2023).

[29] *AprilTag.* URL: https://april.eecs.umich.edu/software/apriltag (besucht am 23. 03. 2023).

[31] *The First Look screenshot image - Unreal Engine 5 - Indie DB.* URL: https://www.indiedb.com/engines/unreal-engine-5/images/the-first-look-screenshot (besucht am 31. 03. 2023).

[33] *Fusion 360 Full Package Training - Codeo.* URL: https://codeo.com.tr/fusion-360-full-package-training/ (besucht am 04. 04. 2023).

[34] *Logo Blender - Blender (software) - Wikipedia*. URL: https://en.wikipedia.org/wiki/Blender_(software)#/media/File:Logo_Blender.svg (besucht am 04. 04. 2023).

[35] Johannes L. Schoenberger. *COLMAP – COLMAP 3.8-dev documentation*. URL: https://colmap.github.io/ (besucht am 21. 03. 2023).

[37] Elias Bauer. *Selbsterstellt mit Unreal Engine*. 2023.

[38] *2/4 stücke mg996r mg 996 Lenk drehung rc arduino Servomotor 180 360 Metall getriebe für rc Auto LKW Hubschrauber Roboter Zubehör— — - AliExpress*. URL: https://de.aliexpress.com/item/1005004560891847.html?spm=a2g0o.order_list.order_list_main.53.90815c5fiB9cfi (besucht am 08. 03. 2023).

[39] *This Arduino-based robot can volley pretty well!* URL: https://twitter.com/arduino/status/1658820535151652867 (besucht am 22. 05. 2023).