

Robotik

Ostfalia Hochschule für angewandte Wissenschaften

Sawyer Lösungen zur Bewältigung des Bamboleo Brettspiels

Marcus Willner, 70483065

Jendrik Heitmann, 70471407

10.01.2023

Inhalt

Aufbau und allg. Auftrag.....	3
Bamboleo	3
Sawyer	3
Planung:.....	4
Umsetzung:.....	5
Bildverarbeitung.....	5
Roboter	8

Aufbau und allg. Auftrag

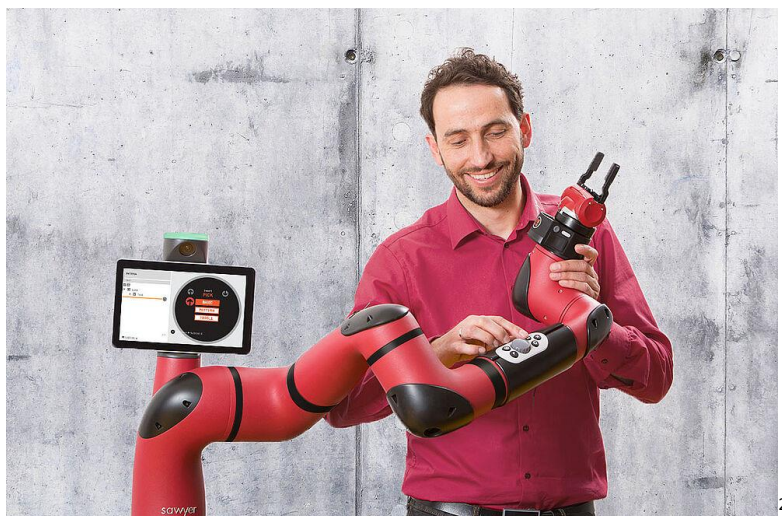
Bamboleo

Bei dem Brettspiel Bamboleo¹ handelt es sich um ein Planungs-Balancierspiel bei dem verschiedene Bausteine auf einer runden Spielfläche abgelegt sind, welche dabei auf einem zentralen Stützpunkt balanciert. Ziel des Spiels ist es, möglichst viele dieser Steine von der Brettoberfläche zu entfernen, ohne, dass diese dabei das Gleichgewicht verliert und umkippt.

Sawyer

Sawyer stellt einen einarmigen kollaborativen Roboter dar, kurz Cobot, der mit sieben Freiheitsgraden ausgestattet ist. Dazu kommen zwei interne Kameras, welche einerseits in der Kopfeinheit und andererseits im sogenannten Cuff verbaut sind.

Des Weiteren bietet Sawyer eine eigene Entwicklungsumgebung namens Intera Studio für simple Anwendungen, sowie ein eigenes SDK zur Entwicklung eigener codebasierter Anwendungen.



Im Rahmen des Projekts soll Sawyer zwei Aufgaben erfüllen. Einerseits werden durch die Kamera die benötigten Bildmaterialien zur Verarbeitung erfasst. Andererseits sollen durch den Roboterarm die Bewegung der Spielelemente wahrgenommen werden.

¹ https://www.zoch-verlag.com/zoch_en/categories/skill-games/bamboleo-601120100-en.html

² <https://www.rethinkrobotics.com/de/sawyer>

Planung:

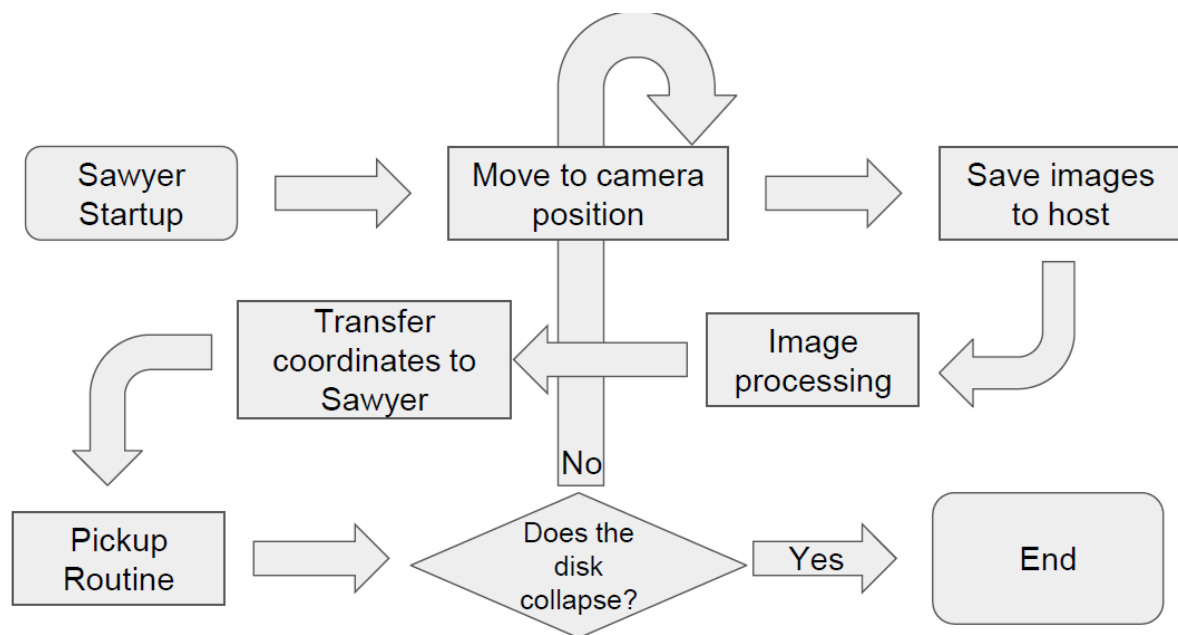


Abb. 1: System Flowchart

Neben dem Training der Bilderkennung sollten im anschließenden Spielverhalten verschiedene Punkte seitens Sawyer angelaufen werden. Dazu sollten einerseits ein Punkt zur Kameraaufnahme gehören, sowie andererseits die verschiedenen Grasp-Approaches.

Im Bestfall wurde so das Ziel gesetzt ein System zu entwickeln, welches neben vordefinierten Positionen auch flexible Koordinaten anlaufen kann.

Unabhängig von der Entwicklung der Bildverarbeitung durch den Taiwanesischen Teil des Teams, sollte also ein eigenes ROS Paket³ entwickelt werden, welches in der Lage ist, ein Set an Koordinaten, welches wiederum aus der Bildverarbeitung übergeben wurde, zu verarbeiten und anzulaufen.

Da das interne Intera Studio nicht in der Lage ist, ein Programm mit Parametern zur Laufzeit darzustellen, ist demnach der Ansatz über das SDK gewählt worden.

³ <http://wiki.ros.org/ROS/Tutorials/CreatingPackage>

Umsetzung:

Bildverarbeitung

Damit der Sawyer sich überhaupt im Raum orientieren kann und in der Lage ist die einzelnen Spielsteine vom Spielfeld zuerkennen, wird eine Bilderkennung benötigt. Ziel dieses Teils ist es, die einzelnen Spielsteine auf dem Feld zu identifizieren, das Massezentrum zu bestimmen und damit einen potenziellen Spielstein zu identifizieren, der entfernt werden kann, ohne das Spielfeld aus dem Gleichgewicht zu bringen. Um dieses Ziel zu erreichen, wurde ein entsprechender Prozess gestaltet, welcher in Abbildung 2 dargestellt ist.

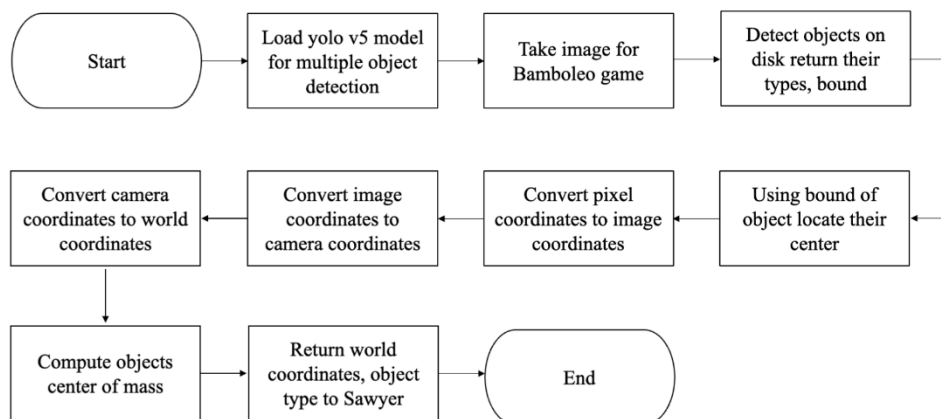


Abb. 2: Vision Part Flowchart

Die Entwicklung hierfür wurde über Python realisiert. Für die eigentliche Bilderkennung wurde das Framework YOLOv5 genutzt. YOLOv5 ist ein Objekterkennungsmodell, welches durch Ultralytics im Sommer 2020 veröffentlicht wurde. Möchte man eigene individuelle Objekte erkennen, wie es in diesem Project der Fall ist, muss das Modell mit entsprechenden Daten trainiert werden.

Im Spiel gibt es verschiedenste Objektformen mit unterschiedlichen Gewichten, die zuverlässig erkannt werden müssen.

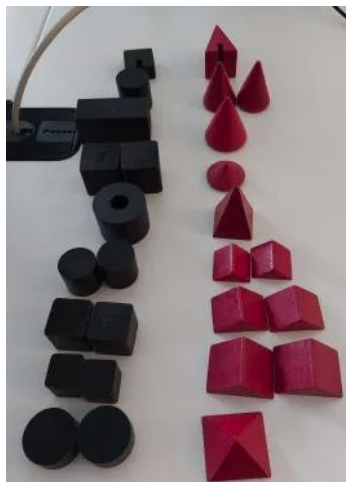


Abb. 3: Bild aller Objekte in Bamboleo

Um eine große Anzahl an Beispielbildern zum Training des Objekterkennungsmodell zu generieren, wurden verschiedensten Ausgangssituationen in einem 3D Blender Modell erstellt. Insgesamt wurden 100 Bilder mit Prismen, Pyramiden, Würfeln und Kegeln wie in Abbildung 4 generiert. Die einzelnen Objekte wurden dann von YOLOv5 gelabelt und das Modell Schritt für Schritt verbessert.

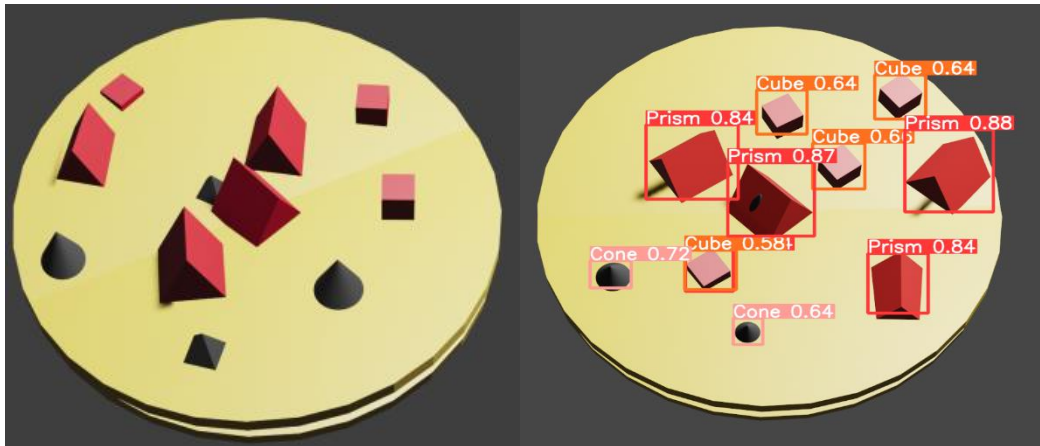


Abb. 4: 3D Modell von Bamboleo mit und ohne YOLOv5 Label

Die Ergebnisse werden am Ende jedes Durchlaufs aktualisiert. Nach den 100 Durchläufen zeigen die Ergebnisse, dass das Model bereits relativ zuverlässig arbeitet und für den Einsatz bereit ist.

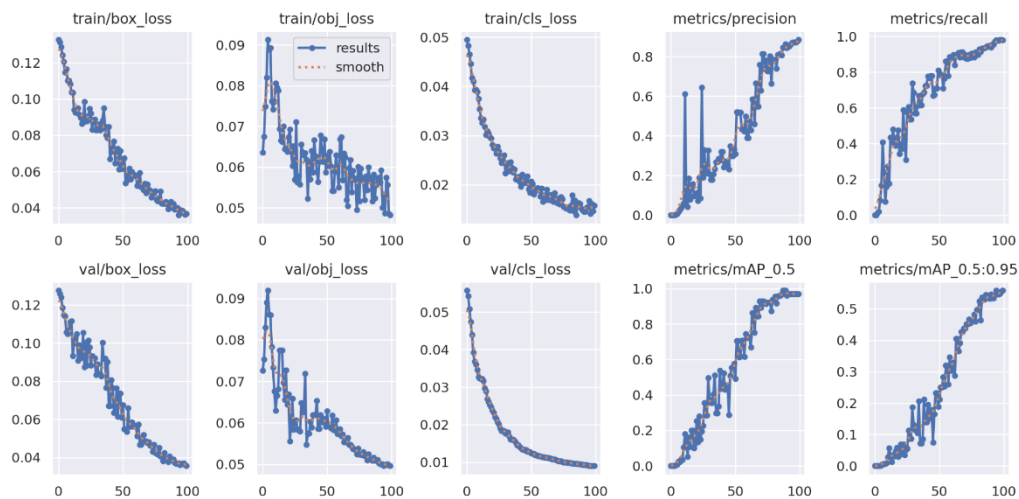


Abb. 5: Ergebnis nach 100 Trainingsbildern

Im nächsten Schritt gilt es, die Kamera des Sawyers zu kalibrieren, damit eine entsprechende Orientierung möglich ist und Koordinaten zur Berechnung des Massezentrums übergeben werden können. Zur Bestimmung der intrinsischen Kameraparameter wird eine Testfeldkalibrierung durchgeführt. Hierfür werden 10 bis 20 Bilder eines Schachbrettmusters mit der Kamera des Sayers aufgenommen und zur Kalibrierung genutzt.

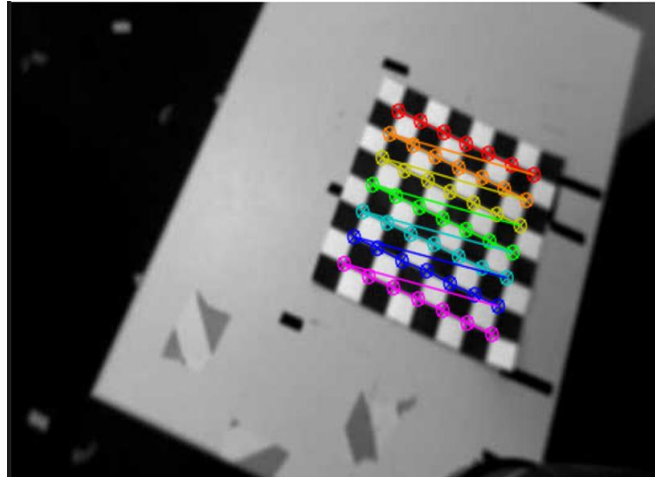


Abb. 6: Testbild mit durchgelaufener Kalibrierung

Mit der erfolgten Kalibrierung ist es nun möglich, die Pixelkoordinaten eines Bildes in Koordinaten in der realen Welt umzurechnen.

Nun können die Objekterkennung und die Umrechnung in Weltkoordinaten zusammengefügt werden, um die Position der einzelnen Spielsteine auf dem Spielfeld zu bestimmen. Abbildung 7 zeigt die erkannten Objekte und die Umrechnung der Koordinaten.

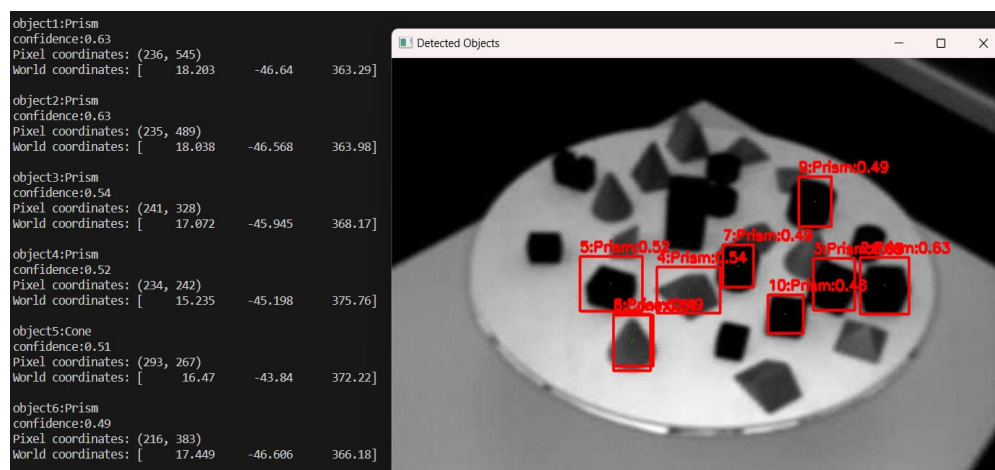


Abb. 7: Aufnahme mit Objekterkennung und Positionsbestimmung

Auf Grundlage der Objekterkennung und der Positionsbestimmung wird im letzten Schritt das Massezentrum berechnet, um einen Kandidaten zum Entfernen zu bestimmen. Dies geschieht in der in Abbildung 8 gezeigten Reihenfolge.

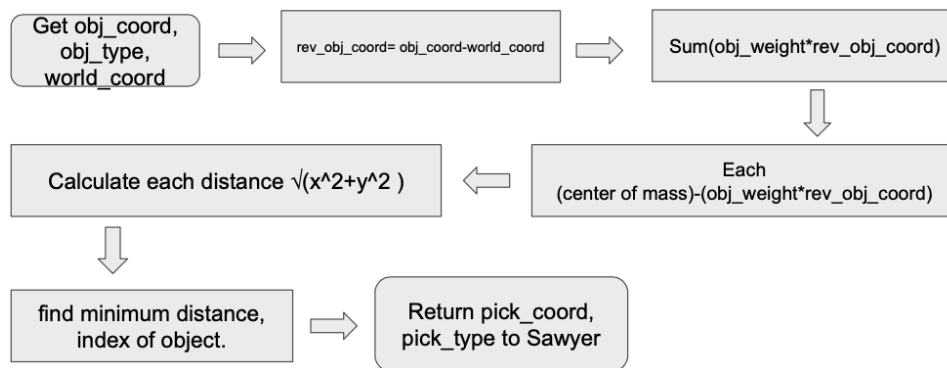


Abb. 8: Bestimmung des Massezentrums

All diese Informationen können nun an den Roboterpart übergeben werden, um die eigentliche Manipulation (Greifen eines Objekts) durchzuführen.

Roboter

Wie oben erwähnt, erlaubt die Erstellung eigener ROS-Pakete die Ausführung von Python Skripten über die Host-Maschine. Dazu werden diese hier auf einem Laptop über Fernsteuerung auf dem Sawyer Robot gestartet.

Im Rahmen der geplanten Durchführung müssen dazu regelmäßig Kamerabilder gemacht werden vom Spielfeld. Diese sollen vom Host-Gerät an die Bildverarbeitung weitergeleitet werden. In diesem Szenario ist die Weiterleitung jedoch noch manuell geschehen.

```

def take_image():
    rp = intera_interface.RobotParams()
    rospy.init_node('camera_display', anonymous=True)
    camera = intera_interface.Cameras()
    camera.start_streaming('right_hand_camera')
    rospy.loginfo("Camera Stream running. Cancel with Ctrl+C")
    rospy.spin
  
```

Abb. 9: Beispielcode des Kamerastreams

Die Kamera leitet nun den gesamten Kamerastream an den Laptop weiter, auf dem die Bilder festgehalten werden.

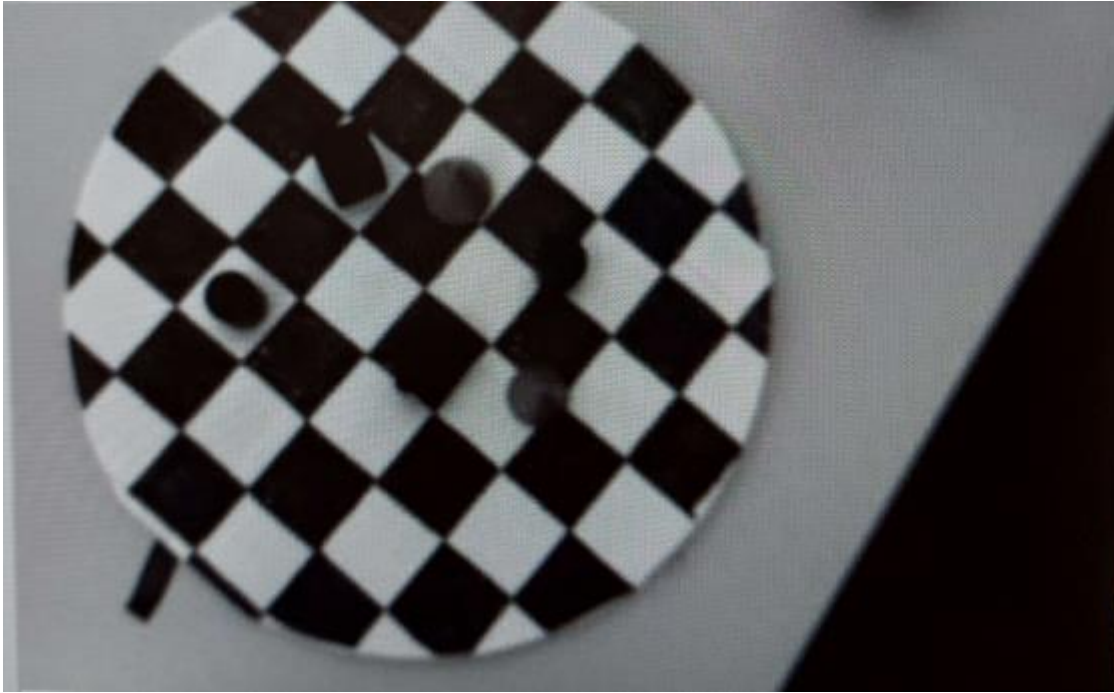


Abb. 10: Graustufenbild der Armkamera

An dieser Stelle wird bereits sichtbar, dass die Identifikation der einzelnen Spielsteine ein recht großes Hindernis darstellen wird, da die Grau-Skalierbarkeit der Armkamera die Farben ausnimmt, und sich zweitens schwarze Spielsteine schlecht vom teilweise schwarzen Hintergrund abheben. Zukünftig wäre eine angehbare Lösung entweder die Farbe der Spielsteine zu ändern, oder im Rahmen der Nachbearbeitung die Exposure anzupassen.

Da die Bildweitergabe momentan eh manuell abläuft, ist die Bewegung des Roboters im Folgenden allerdings mit Mockdaten durchgeführt wurde, um so das Thema der Bildnachbearbeitung zu umgehen.

Sollten die Bilder in der Theorie verarbeitet sein, werden dazu je nach Wunsch entweder Weltkoordinaten oder Matrizen zurückgegeben, welche dann als Parameter an den Roboter übergeben werden.

In der Ausführung sieht dies wie folgt aus.

```
positions = {
    'start' : (0.2, -0.3, 0.2, 0.7, 0.4, 0.4, 0.4),
```

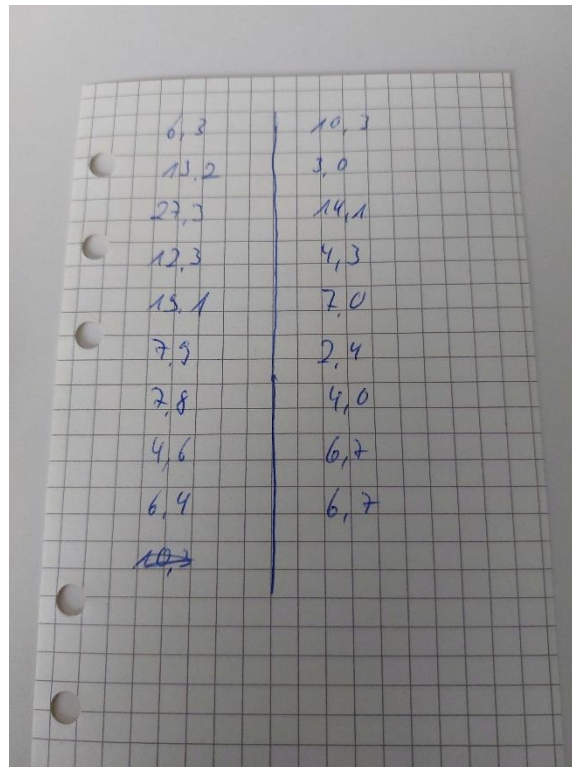
Abb. 11: Winkelgelenkeinstellung zur Positionseinnahme

Positionen sind für die vordefinierten direkt im Code angelegt. Hier werden die verschiedenen Winkel, die jedes Gelenk einnehmen muss direkt gespeichert, sodass diese Punkte im Code verwendet werden können.

In Sachen Spielsteine werden diese Winkel als Parameter der Bewegung übergeben, sodass der Roboter in der Lage ist, spontane Positionen anzusteuern, etwas, das rein mit der Nutzung von Intera nicht möglich erschien.

Allerdings musste festgestellt werden, dass die interne Lösung der Matrizen teilweise zu bisher unbekannten Fehlern führte, sodass aus manchen Positionen manche andere nicht korrekt angesteuert werden könnten. Deshalb wurde das Videomaterial der dazugehörigen Position mit Intera gefertigt aus Gründen der Vorstellbarkeit.

Des Weiteren ist bisher keine Funktion enthalten, welche das Spiel korrekt nach vorgegebenen Regeln spielt. Zwar ist der Roboter nun „fast“ in der Lage die Spielsteine korrekt zu entfernen, jedoch fehlt die Auswertung der Gewichtsverteilung anhand der Spielplattform. Dazu wäre in der Zukunft der Nutzen einer externen Kamera denkbar, die seitlich des Spielbretts dessen Neigung erfasst und dies zur Berechnung teilt.



A photograph of a piece of graph paper with handwritten data in blue ink. The data is organized into two columns separated by a vertical line. The left column contains 11 values, and the right column contains 10 values. The last value in the left column is crossed out and replaced with an arrow pointing to the right.

6,3	10,3
13,2	3,0
23,3	14,1
12,3	4,3
15,1	7,0
7,9	2,4
7,8	4,0
4,6	6,7
6,4	6,7
10,3 →	

Abb. 12: Massetabelle der Spielsteine

Ansatz hier war kurzzeitig das Gewicht der Steine zu erfassen, um möglichst diese Varianz zu eliminieren. Allerdings weicht das Gewicht der Steine stark voneinander ab, sodass dies weiterhin mit einbezogen werden muss.