School of Engineering

Centre for Artificial Intelligence

Bachelorarbeit Computer Science

# Autonome Schlauchführung als Beispiel der robotischen Handhabung verformbarer linearer Objekte


| | | |
|---|---|---|
| **Autorinnen** | | Theresa Valentina Neubert<br>Felicitas Farah Baur |
| **Hauptbetreuung** | | Prof. Dr. Alisa Rupenyan-Vasileva |
| **Nebenbetreuung** | | Diego Bolliger |
| **Datum** | | 05.06.2026 |

# Zusammenfassung

Verformbare lineare Objekte wie Schläuche oder Kabel bestehen aus elastischen, weichen Materialien. Dies führt zu einem komplexen, nicht linearen Verhalten, das ihre Modellierung und robotische Handhabung erschwert. Die zuverlässige robotische Handhabung solcher Objekte ist daher trotz hoher industrieller Relevanz nach wie vor Gegenstand aktueller Forschung.

Eine mögliche Anwendung der Robotik in Bezug auf verformbare Objekte stellt das Verlegen und Verbinden von Schläuchen und Kabeln dar. Das Ziel dieser Arbeit liegt darin, einen Schlauch von einem Roboterarm durch ein Labyrinth manövrieren zu lassen. Damit soll belegt werden, dass Roboter auch im Umgang mit elastischen und verformbaren Schläuchen Anwendung finden können.

Der Versuchsaufbau besteht aus einem kollaborativen FANUC CR-7iA/L Roboterarm, einer Intel RealSense D435iF Stereokamera, einem Bota Systems Kraft-Moment-Sensor sowie einem Robotiq Hand-E Greifer. Ein Labyrinth aus 35 Holzsäulen, welche auf einer Spanplatte festgeschraubt wurden, bildet eine kontrollierte Versuchsumgebung, in der typische Problemstellungen bei der Handhabung von Schläuchen gezielt herbeigeführt und untersucht werden können. Als Versuchsobjekte während der Entwicklung dienen zunächst zwei hohle, weiche Silikonschläuche unterschiedlicher Länge. Ein massiver und dadurch weniger flexibler Silikonschlauch wird anschliessend eingesetzt, um die Generalisierbarkeit des Systems zu überprüfen. Alle Schläuche werden hierbei an einem Ende mit zwei farblich unterschiedlichen Ringen aus Isolierband gekennzeichnet, um eine eindeutige Orientierung festzulegen. Dieses Ende wird im Folgenden als Schlauchanfang bezeichnet.

Für die Erkennung des Schlauchanfangs wurde eine Software entwickelt, die beide Isolierbandringe über HSV-Farbsegmentierung lokalisiert und damit deren Position sowie Orientierung bestimmen kann. Die Koordinatensystemkalibrierung stützt sich auf alle 35 Säulen des Labyrinths und kombiniert eine starre SVD-Transformation mit einer nicht-starren Thin-Plate-Spline-Korrektur. Die Pfadplanung erfolgt mittels eines angepassten A-Star-Algorithmus, der kollisionsfreie Wege durch das Labyrinth bevorzugt, die möglichst weit von den Säulen entfernt verlaufen. Bleibt der Schlauch im Labyrinth stecken, so wird die daraus resultierende Überschreitung der Zugkräfte mittels Sensor am Greifer registriert. Dies löst eine Programmsequenz aus, die den Schlauch skelettiert, seinen Verlauf durch das Labyrinth bestimmt und die Säulen identifiziert, an denen der Schlauch hängen geblieben ist. Anschliessend wird der Schlauch durch gezielte Greif- und Zugbewegungen befreit.

Die Evaluation zeigt, dass der Anfang des Schlauches unter Laborbedingungen in 100 % der Fälle korrekt erkannt werden konnte. Die Säulen, an denen der Schlauch hängen geblieben war, wurden in 73,3 % der Extremsituationen korrekt erkannt, wobei fehlerhafte Bézier-Interpolationen beim Schliessen von Lücken im Skelett als Hauptfehlerursache identifiziert wurden. Das System zeigt, dass eine autonome robotische Schlauchführung grundsätzlich möglich ist und bildet die Basis für weitere Forschung im Bereich der automatisierten Verlegung von Schläuchen und Kabeln.

# Abstract

Deformable linear objects such as hoses and cables respond to forces and friction in ways that are difficult to model and predict. This results from their elasticity and material compliance, which lead to complex non-linear behavior. Consequently, robotic manipulation of such objects remains an open research challenge despite its high industrial relevance.

One potential application of robotics involving deformable objects is the routing and connection of hoses and cables. The objective of this thesis was to enable a robotic arm to guide a hose through a maze. This serves to demonstrate that robots can also be applied to tasks involving elastic and deformable hoses.

The experimental setup consists of a collaborative FANUC CR-7iA/L robotic arm, an Intel RealSense D435iF stereo camera, a Bota Systems force-torque sensor, and a Robotiq Hand-E gripper. A maze comprising 35 wooden posts mounted on a particle board provides a controlled experimental environment in which typical challenges associated with hose manipulation can be deliberately reproduced and investigated. During development, two hollow silicone hoses of different lengths were used as test objects. A solid and stiffer silicone hose was subsequently employed to evaluate the generalizability of the developed system. For all hoses, one end was defined as the hose tip and marked using two differently colored rings of electrical tape.

To detect the hose tip, software was developed that localizes the two tape rings using HSV color segmentation and thereby determines their position and orientation. Coordinate system calibration is based on all 35 maze posts and combines a rigid singular value decomposition transformation with a non-rigid thin plate spline correction. Path planning is performed using a modified A-star search algorithm that favors collision-free routes through the maze while maximizing the distance from surrounding posts. If the hose becomes stuck within the maze, the resulting increase in pulling force is detected by the sensor mounted on the gripper. This triggers a recovery procedure that skeletonizes the hose, determines its path through the maze, and identifies the posts at which the hose has caught. The hose is then released through targeted grasping and pulling motions at those locations.

The evaluation shows that the hose tip could be detected correctly in 100% of cases under laboratory conditions. The posts at which the hose had caught were correctly identified in 73.3% of extreme test scenarios, with inaccurate Bézier interpolation during skeleton gap closure being identified as the primary source of error. The results demonstrate that autonomous robotic hose routing is fundamentally feasible and provide a foundation for further research into the automated installation of hoses and cables.

# Vorwort

Die vorliegende Arbeit haben wir, Felicitas Farah Baur und Theresa Valentina Neubert, im Rahmen unserer Bachelorarbeit für den Bachelor of Science in Informatik an der Zürcher Hochschule für Angewandte Wissenschaften verfasst. Die gewählte Aufgabenstellung des Centre for Artificial Intelligence sowie die Dokumentation haben wir gemeinsam und zu gleichen Teilen bearbeitet. Alle Bestandteile des Systems haben wir kollaborativ entwickelt. Sie sind das Ergebnis gemeinsamer Diskussionen, Experimente und iterativer Verfeinerung. Im Rahmen einer thematischen Vertiefung hat sich Felicitas auf den Kraft-Sensor und die Navigation des Roboters konzentriert, Theresa auf die Bilderkennung.

Schon vor dem Studium haben wir uns beide in der Freizeit mit kleineren Elektronikprojekten beschäftigt. Theresa baut in ihrer Freizeit unter anderem Custom-Gameboys, Felicitas hat ein System entwickelt, das ihre Pflanzen automatisch giesst. Über das Studium hinweg ist bei uns beiden der Wunsch gewachsen, über unsere rein softwareseitigen Fähigkeiten hinauszuwachsen und ein Projekt umzusetzen, in dem Software direkt mit der Hardware zusammenspielt, weshalb wir uns für ein Robotikprojekt entschieden haben.

Ein besonderer Dank gilt Diego Bollinger, der diese Arbeit betreut und uns bei allen technischen Fragen sowie konzeptionellen Überlegungen unterstützt hat. Ebenso danken wir Prof. Dr. Alisa Rupenyan-Vasileva für das entgegengebrachte Vertrauen und die Unterstützung während des gesamten Projektverlaufs. Nicht zuletzt geht ein herzlicher Dank an alle Personen in unserem persönlichen Umfeld, die durch Korrekturlesen des Berichts, kritisches Mitdenken und stetigen Rückhalt zum Gelingen dieser Arbeit beigetragen haben.

# Inhaltsverzeichnis

1 Einleitung . . . 1
1.1 Ausgangslage . . . 1
1.2 Problemstellung . . . 1
1.2.1 Offizielle Aufgabenstellung . . . 1
1.2.2 Eigene Zielsetzung und Aufgabenwahl . . . 2
1.2.3 Aufbau der Arbeit . . . 2
2 Theoretische Grundlagen . . . 3
2.1 Industrieroboter . . . 3
2.1.1 Koordinatensysteme . . . 3
2.1.2 Homogene Koordinaten . . . 3
2.1.3 Homogene Transformationsmatrizen . . . 4
2.1.4 Euler-Winkel und Rotationsdarstellung . . . 4
2.1.5 Kinematik . . . 4
2.1.6 FANUC Remote Motion Interface . . . 5
2.2 Computer Vision in der Robotik . . . 5
2.2.1 Lochkameramodell und intrinsische Parameter . . . 5
2.2.2 Tiefenkamera . . . 5
2.2.3 Pose-Schätzung mit solvePnP . . . 5
2.2.4 Augmented Reality University of Córdoba (ArUco)-Marker . . . 6
2.2.5 Farbbasierte Bildsegmentierung . . . 6
2.2.6 Konturanalyse und Ellipsen-Fitting . . . 6
2.2.7 Skelettierung . . . 7
2.2.8 Kamera-Roboter-Kalibrierung . . . 8
2.3 Pfadplanung . . . 9
2.3.1 Occupancy Grid . . . 9
2.3.2 Distance Transform . . . 9
2.3.3 A-Star-Algorithmus . . . 9
2.3.4 Pfadvereinfachung nach Ramer-Douglas-Peucker . . . 9
2.4 Manipulation deformierbarer Objekte . . . 10
2.4.1 Deformierbare lineare Objekte . . . 10
2.4.2 Greifstrategie und Kraftüberwachung . . . 10
2.5 Kraftmessung und -überwachung . . . 10
2.5.1 Kraft-Moment-Sensorik . . . 10
2.5.2 Schwellenwert-basierte Überlasterkennung . . . 10
3 Vorgehen und Methoden . . . 11
3.1 Versuchsaufbau . . . 11
3.1.1 Experimenteller Aufbau . . . 11
3.1.2 Hardwarekomponenten . . . 12
3.1.3 Softwarekomponenten . . . 15
3.2 Durchführung . . . 16
3.2.1 Teilprojekt 1: Erkennen des Schlauchs . . . 16
3.2.2 Teilprojekt 2: Kurzer Schlauch durch das Labyrinth . . . 17
3.2.3 Teilprojekt 3: Langer Schlauch mit Feststecken-Erkennung . . . 18
3.3 Lösungsbeschreibung . . . 20
3.3.1 Top-Level-Architektur . . . 20
3.3.2 Module im Schlauchprojekt . . . 21

3.4 Detailbeschreibung der Module . . . 23
3.4.1 Kalibrierung des Koordinatensystems . . . 23
3.4.2 Hauptablauf . . . 24
3.4.3 Erkennen des Schlauchs . . . 26
3.4.4 Pfadplanung . . . 27
3.4.5 Roboteraktionen . . . 30
3.4.6 Kraftüberwachung . . . 32
3.4.7 Verfahren zum Befreien des Schlauchs . . . 33
3.4.8 Safe-Grasp . . . 37
4 Experimentelle Resultate . . . 39
4.1 Schlaucherkennung . . . 39
4.1.1 Resultate . . . 39
4.2 Kraft-Sensor Schwellenwert . . . 41
4.2.1 Resultate . . . 42
4.3 Optimale Schrittlänge . . . 43
4.3.1 Resultate . . . 44
4.4 Erkennung des feststeckenden Schlauchs . . . 45
4.4.1 Resultate . . . 45
4.4.2 Vereinfachung auf Tier 3 Near . . . 48
4.5 Systemtest . . . 49
4.5.1 Resultate . . . 50
5 Diskussion und Ausblick . . . 51
5.1 Zielerreichung . . . 51
5.2 Grenzen des Systems . . . 52
5.2.1 Skelett-Rekonstruktion bei der Problemstellen-Erkennung . . . 52
5.2.2 Abhängigkeit vom Schlauchmatrial . . . 52
5.2.3 Restriktionen des Versuchsaufbaus . . . 52
5.2.4 Dreidimensionale Schlauchrekonstruktion . . . 53
5.3 Ungelöste Problemstellungen . . . 53
5.3.1 Schlauchschleifen . . . 53
5.3.2 Tageslichtempfindlichkeit . . . 53
5.4 Verbesserungsmöglichkeiten . . . 53
5.4.1 Skelett-Rekonstruktion . . . 53
5.4.2 Lern-basierte Erkennung der Problemstelle . . . 53
5.4.3 Farbiger Referenzring . . . 54
5.4.4 Parameter auf Material abstimmen . . . 54
5.4.5 Vermeidung von Singularitäten . . . 54
5.4.6 Detektionsverfahren . . . 54
5.5 Industrielle Anwendungsmöglichkeiten . . . 55
Literaturverzeichnis . . . 56
Glossar . . . 59
Abbildungsverzeichnis . . . 62
Tabellenverzeichnis . . . 63
A Nutzungsdeklaration künstlicher Intelligenz . . . 64
B Projektbeschreibung . . . 65
C Quellcode-Ablage . . . 67

# 1 Einleitung

Die robotische Handhabung von Deformable Linear Objects (DLOs) wie Schläuchen und Kabeln gilt als offene Forschungsfrage. Das automatisierte Arbeiten mit starren Werkstücken ist in der Industrie seit Jahrzehnten etabliert, vergleichbare Verfahren für verformbare Objekte fehlen jedoch. Die vorliegende Arbeit nähert sich diesem Feld an einem konkreten Versuchsaufbau.

## 1.1 Ausgangslage

Die robotische Handhabung starrer Körper ist heute weitgehend erforscht und in zahlreichen industriellen Anwendungen breit etabliert. Deutlich anspruchsvoller gestaltet sich hingegen die Manipulation verformbarer Objekte, die nach wie vor ein aktives Forschungsfeld mit zahlreichen offenen Fragestellungen darstellt. Typische Untersuchungsgegenstände sind die Handhabung von DLOs wie Seilen oder Kabeln [1], [2], [3], das Falten und Manipulieren textiler Materialien [4] sowie Anwendungen in der robotergestützten Chirurgie, etwa das Anheben von Gewebe [5]. Im Unterschied zu starren Körpern erfordert die Handhabung solcher Objekte eine fortgeschrittene visuelle und haptische Wahrnehmung, ein nachgiebiges Regelverhalten sowie ein fundiertes Verständnis des physikalischen Verhaltens der verwendeten Materialien [5], [6].

Aktuelle Übersichtsarbeiten zeigen, dass sich die Forschung zu DLOs im letzten Jahrzehnt stark ausdifferenziert hat und mittlerweile ein breites Spektrum an Modellierungs-, Simulations- und Wahrnehmungsverfahren vorliegt [7]. Gleichzeitig ist die Automatisierung in industriellen Anwendungen mit DLOs wie Schläuchen, Kabeln oder Kabelbäumen trotz hoher wirtschaftlicher Relevanz bisher wenig ausgereift [6].

Am Centre for Artificial Intelligence (CAI) der Zürcher Hochschule für Angewandte Wissenschaften (ZHAW) School of Engineering untersucht die Industrial Artificial Intelligence Group (IAI) ebensolche Problemstellungen mit dem Ziel, die Fähigkeiten robotischer Systeme für industrielle Einsatzszenarien weiterzuentwickeln. Im konkreten Kontext der industriellen Montage sind Verschlauchungs- und Verkabelungsaufgaben von besonderer Bedeutung, da sie den direkten Umgang mit DLOs beinhalten und sich mit heutigen robotischen Systemen nur schwer automatisieren lassen. Die vorliegende Arbeit legt eine Grundlage für weitere Forschungsvorhaben, indem ein robotischer Versuchsaufbau sowie ein System zur autonomen Ausführung dieser Aufgaben entwickelt, implementiert und erprobt werden, um Erkenntnisse über Herausforderungen und Lösungsansätze bei der Handhabung verformbarer Objekte zu gewinnen.

## 1.2 Problemstellung

Nachfolgend wird dargelegt, wie sich die offizielle Aufgabenstellung von der konkreten Zielsetzung unterscheidet. Dazu werden die Arbeitspakete vorgestellt, die Zielsetzung ausgeführt und der Aufbau der Arbeit beschrieben.

### 1.2.1 Offizielle Aufgabenstellung

Das primäre Ziel dieser Arbeit bestand darin, einen industriellen Roboterarm zu befähigen, eine für die Verschlauchung in der Montage typische Aufgabe autonom auszuführen. Dazu wurde eine konkrete Aufgabe gewählt, ein geeigneter experimenteller Aufbau konstruiert und ein passender Schlauch sowie zusätzliches Material bestimmt. Mithilfe einer Kamera und Algorithmen aus dem Gebiet der Computer Vision (CV) wurde ein Wahrnehmungssystem entwickelt, das Schlauch und weitere Objekte im Arbeitsraum lokalisiert. Auf Basis eines bestehenden Python-Treibers wurde anschliessend ein Programm für den Roboterarm entwickelt, das den Schlauch greift und die gewählte Aufgabe ausführt.

Das Projekt gliedert sich gemäss Aufgabenstellung in folgende Arbeitspakete:

1. **Einarbeitung in den Roboter:** Sicherer und selbstständiger Umgang mit dem Roboterarm sowohl über das physische Bedienpanel als auch softwareseitig.
2. **Entwurf des Versuchsaufbaus:** Festlegung der zu automatisierenden Aufgabe, Auswahl von Schlauch und Zusatzmaterial, Definition des Versuchslayouts sowie der Kriterien für Erfolg und Misserfolg.
3. **Wahrnehmung von Schlauch und Zusatzobjekten:** Mit der zur Verfügung stehenden Kamera werden der Schlauch und weitere relevante Objekte erfasst und in das Koordinatensystem des Roboters überführt. Dies kann sowohl mit einfachen Strategien aus dem Bereich CV auf Basis von Markern als auch mit leistungsfähigeren Modellen des maschinellen Lernens wie You Only Look Once (YOLO) [8] umgesetzt werden.
4. **Roboterprogramm:** Auf Grundlage der räumlichen Informationen aus dem Wahrnehmungssystem wird ein Programm entwickelt, das den Schlauch an einer geeigneten Stelle greift und die gewählte Aufgabe ausführt. Ziel ist ein tragfähiger erster Lösungsansatz mit vollständigem Ausführungsversuch.
5. **Optimierung:** Da eine erfolgreiche Ausführung aufgrund der Verformbarkeit des Schlauchs erfahrungsgemäss selten unmittelbar gelingt, werden Arbeitsablauf, Roboterbewegungen und Wahrnehmungssystem iterativ verfeinert, um die Erfolgsquote zu erhöhen.
6. **Evaluation:** Abschliessend wird die Leistung der Methode quantitativ und qualitativ bewertet. Zusätzlich werden potenzielle Fehlerquellen dokumentiert und mögliche Gegenmassnahmen aufgezeigt.
7. **Dokumentation:** Ergebnisse, Vorgehen und Reflexion werden im vorliegenden Bericht festgehalten.

Die offizielle Aufgabenstellung des CAI ist im Anhang B dieser Arbeit beigefügt.

### 1.2.2 Eigene Zielsetzung und Aufgabenwahl

Aufbauend auf der offiziellen Aufgabenstellung wurden folgende konkrete Ziele erarbeitet:

1. Der Roboterarm soll den Schlauch in einem definierten Arbeitsbereich präzise und sicher greifen können.
2. Der Roboterarm soll einen kurzen Schlauch präzise und sicher in das definierte Ziel manövrieren können.
3. Der Roboterarm soll einen langen Schlauch navigieren können und erkennen, wenn dieser im Arbeitsbereich feststeckt. Ein Feststecken des Schlauches soll er autonom lösen können.

Im Fokus steht dabei die prototypische Realisierung eines funktionsfähigen Gesamtsystems, nicht die Entwicklung neuer wissenschaftlicher Methoden zur Manipulation verformbarer Objekte.

### 1.2.3 Aufbau der Arbeit

Das Kapitel 2 führt in die für das Projekt relevanten theoretischen Grundlagen ein. Im Kapitel 3 wird das methodische Vorgehen, insbesondere der Versuchsaufbau sowie die gewählten Mess- und Steuerungsansätze, beschrieben. Kapitel 4 stellt die erzielten Resultate der einzelnen Arbeitspakete dar. Kapitel 5 diskutiert diese Resultate bezüglich Aussagekraft und Relevanz und gibt einen Ausblick auf weiterführende Arbeiten. Anschliessend folgen das Literaturverzeichnis, das Glossar mit den verwendeten Fachbegriffen und Abkürzungen sowie das Abbildungs- und Tabellenverzeichnis. Im Anhang folgen die Nutzungsdeklaration der eingesetzten KI-Werkzeuge, die offizielle Projektbeschreibung des CAI sowie die Quellcode-Ablage mit Hinweisen zum entstandenen Repository.

# 2 Theoretische Grundlagen

Die in dieser Arbeit umgesetzte Lösung greift auf Bausteine aus klassischer Robotik, Computer Vision und algorithmischer Pfadplanung zurück. Aus jedem dieser Felder werden nur jene Aspekte aufgegriffen, die für das Verständnis der späteren Implementierung relevant sind.

## 2.1 Industrieroboter

Ein **Industrieroboter** ist gemäss ISO 8373 ein automatisch gesteuerter, programmierbarer, multifunktionaler Manipulator mit drei oder mehr Achsen, der für industrielle Aufgaben eingesetzt wird [9]. Im Jahr 2024 wurden weltweit 542’000 Industrieroboter neu installiert, mehr als doppelt so viele wie zehn Jahre zuvor [10].

Typische Anwendungen sind [11]:
- Schweissen
- Lackieren
- Bestücken
- Verpacken

Ein **Knickarmroboter**, ein spezifischer Industrieroboter oder auch serieller Manipulator genannt, besteht aus einer Folge starrer Glieder, die durch motorisierte Drehgelenke verbunden sind. Die Glieder bilden eine offene kinematische Kette vom ortsfesten Sockel bis zum Endeffektor. Mit sechs Drehgelenken verfügt der Roboter über sechs Degrees of Freedom (DoF), was ausreicht, um den Endeffektor in einer beliebigen Position und Orientierung innerhalb des Arbeitsraums zu positionieren [12].

### 2.1.1 Koordinatensysteme

Zur mathematischen Beschreibung der Roboterbewegung wird jedem Glied ein **kartesisches Koordinatensystem** zugeordnet. Von praktischer Bedeutung sind dabei folgende Koordinatensysteme:

- Das **Weltkoordinatensystem** dient als raumfestes Referenzsystem, relativ zu dem alle anderen Koordinatensystem beschrieben werden.
- Das **Basiskoordinatensystem** ist am Roboterfuss fixiert und bildet den Ausgangspunkt der kinematischen Kette.
- Das **Werkzeugkoordinatensystem** ist am Endeffektor angebracht und beschreibt dessen Lage und Orientierung im Raum.
- Der **Tool Center Point (TCP)** bezeichnet den definierten Arbeitspunkt des Werkzeugs. Alle Bewegungsbefehle beziehen sich auf diesen Punkt.

Diese Koordinatensysteme und ihre Benennung folgen der in der Robotik üblichen Konvention [12].

### 2.1.2 Homogene Koordinaten

In der klassischen kartesischen Darstellung lassen sich Translation und Rotation nicht einheitlich als Matrixmultiplikation ausdrücken. **Homogene Koordinaten** lösen dieses Problem, indem jedem Punkt eine zusätzliche Koordinate $w = 1$ hinzugefügt wird:

$$\tilde{p} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Damit lässt sich eine kombinierte Rotation-Translation als einzige Matrixmultiplikation schreiben, was die in Kapitel 2.1.3 beschriebene Darstellung erst ermöglicht [13].

### 2.1.3 Homogene Transformationsmatrizen

Um die relative Lage zweier Koordinatensystem zueinander zu beschreiben, müssen sowohl Rotation als auch Translation berücksichtigt werden. Eine Rotationsmatrix $R \in SO(3)$ beschreibt die Orientierung eines Frames relativ zu einem anderen, während ein Translationsvektor $p \in \mathbb{R}^3$ die Position des Ursprungs angibt [12]. Beide Grössen werden in einer **homogenen Transformationsmatrix** $T \in SE(3)$ zusammengefasst:

$$T = \begin{pmatrix} R & p \\ 0^T & 1 \end{pmatrix}$$

Der wesentliche Vorteil dieser Darstellung liegt darin, dass die Verkettung mehrerer aufeinanderfolgender Transformationen durch einfache Matrixmultiplikation ausgedrückt werden kann [14]. Diese Darstellung kommt in den folgenden Abschnitten sowohl für die Beschreibung der Roboterkinematik im Kapitel 2.1.5 als auch für die Kamera-Roboter-Kalibrierung im Kapitel 2.2.8 zum Einsatz.

### 2.1.4 Euler-Winkel und Rotationsdarstellung

Eine Rotation im dreidimensionalen Raum lässt sich durch drei aufeinanderfolgende Drehungen um die Koordinatenachsen beschreiben. Diese drei Winkel werden als **Euler-Winkel** bezeichnet, wobei die Reihenfolge der Drehachsen (Konvention) festgelegt sein muss. Ein bekanntes Problem dieser Darstellung ist der **Gimbal Lock**:

Wird der mittlere Drehwinkel $\pm 90°$, fallen erste und dritte Drehachse zusammen und ein DoF geht verloren [12].

Eine alternative Darstellung bietet die **Rodrigues-Rotation**. Sie beschreibt eine Rotation kompakt durch einen einzelnen Vektor $\vec{r}$, dessen Richtung die Drehachse $\hat{k}$ und dessen Betrag den Drehwinkel $\theta$ angibt:

$$R = I\cos\theta + (1-\cos\theta)\hat{k}\hat{k}^T + \left[\hat{k}\right]_\times \sin\theta$$

Diese kompakte, singularitätsfreie Darstellung eignet sich besonders für die Optimierung von Pose-Schätzungen und wird daher in der CV bevorzugt eingesetzt [12].

### 2.1.5 Kinematik

Ein Roboter mit $n$ Gelenken bildet eine offene kinematische Kette aus starren Gliedern. Die Vorwärtskinematik beschreibt die Pose des Endeffektors als Verkettung der einzelnen Gelenktransformationen [12]:

$$T = T_1 \cdot T_2 ... T_n$$

Die inverse Kinematik löst das umgekehrte Problem: Zu einer gegebenen Zielpose werden die nötigen Gelenkwinkel berechnet. Diese Lösung ist im Allgemeinen nicht eindeutig, da ein sechsachsiger Roboter dieselbe Pose in mehreren Konfigurationen erreichen kann. Bei FANUC-Robotern werden diese durch drei Flags unterschieden: Flip/No-Flip, Up/Down und Front/Back [15].

Die Menge aller erreichbaren TCP-Positionen wird als Arbeitsraum bezeichnet und ist durch Gelenklimits und Gliedlängen begrenzt [12]. In dieser Arbeit wird die inverse Kinematik nicht selbst implementiert. Die Steuerung empfängt kartesische Zielposen, löst die inverse Kinematik intern und führt die Bewegung als Gelenk- oder Linearbewegung aus.

### 2.1.6 FANUC Remote Motion Interface

Die Kommunikation zwischen dem Steuerungsprogramm und der FANUC-Steuerung erfolgt über das Remote Motion Interface (RMI), eine Transmission Control Protocol / Internet Protocol (TCP/IP)-basierte Application Programming Interface (API), über die JavaScript Object Notation (JSON)-kodierte Befehle gesendet und empfangen werden. Das RMI unterscheidet asynchrone Bewegungsbefehle, die in einem Buffer der Steuerung eingereiht werden, und synchrone Abfragen mit sofortiger Antwort. Eine ausführliche Beschreibung des Protokolls findet sich in der Bachelorarbeit von G. Bortolin [16].

## 2.2 Computer Vision in der Robotik

In diesem Abschnitt wird die Theorie der bildverarbeitungsbasierten Verfahren beschrieben, die in dieser Arbeit zur Erkennung und Lokalisierung des Schlauchs sowie zur Kalibrierung zwischen Kamera und Roboter eingesetzt werden.

### 2.2.1 Lochkameramodell und intrinsische Parameter

Das Lochkameramodell beschreibt die Projektion eines 3D-Punktes $P = (X, Y, Z)^T$ auf den 2D-Bildpunkt $P' = (u, v)^T$. In homogenen Koordinaten lässt sich diese Projektion als Matrixmultiplikation ausdrücken [17]:

$$P' = K[I \mid \mathbf{0}]P$$

Dabei ist $K$ die Kameramatrix mit den Brennweiten $f_x$, $f_y$ und dem Bildhauptpunkt $c_x$, $c_y$ [12]:

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

Mit einem bekannten Tiefenwert $d$ lässt sich die Projektion umkehren, wobei $u$ und $v$ die Pixelkoordinaten des detektierten Bildpunkts bezeichnen:

$$X_c = (u - c_x) \cdot \frac{d}{f_x}, \quad Y_c = (v - c_y) \cdot \frac{d}{f_y}$$

Reale Linsenverzerrungen werden durch Verzeichnungskoeffizienten modelliert, die bei der in Kapitel 2.2.3 beschriebenen Poseberechnung intern von Open Source Computer Vision Library (OpenCV) berücksichtigt werden.

### 2.2.2 Tiefenkamera

Eine Tiefenkamera liefert neben dem Farbbild einen Tiefenwert pro Pixel. Bei Stereo-Infrarot-Kameras wird die Tiefe durch Triangulation aus der Disparität zweier versetzter Sensorbilder berechnet [12]. Das Tiefenbild wird pixelweise mit dem Farbbild ausgerichtet und ermöglicht in Kombination mit der Kameramatrix aus Kapitel 2.2.1 die Umrechnung jedes Bildpunkts in eine 3D-Position. Diese Umrechnung kommt in dieser Arbeit sowohl beim Kapitel 3.4.3 als auch beim Kapitel 3.4.7.1 zum Einsatz.

### 2.2.3 Pose-Schätzung mit solvePnP

Das Perspective-n-Point (PnP)-Problem besteht darin, aus $n$ bekannten 3D-Punkten eines Objekts und deren 2D-Projektionen im Kamerabild die Pose des Objekts relativ zur Kamera zu bestimmen [12]. Die Lösung liefert einen Rotationsvektor $\vec{r}$ in der in Kapitel 2.1.4 beschriebenen Rodrigues-Darstellung und einen Translationsvektor $t \in \mathbb{R}^3$. Daraus lässt sich eine homogene Transformationsmatrix, wie in Kapitel 2.1.3, aufbauen, die das Objekt im Kamerakoordinatensystem verortet.

In dieser Arbeit wird `cv2.solvePnP` [18] mit dem Infinitesimal Plane-based Pose Estimation (IPPE)-Algorithmus [19] eingesetzt, der für quadratische planare Marker optimiert ist. Als Eingabe dienen die vier Ecken des Markers in Bild- und Objektkoordinaten sowie die Kameramatrix und die Verzeichnungskoeffizienten. Dieses Verfahren kommt ausschliesslich bei der Berechnung der Säulenpositionen durch Marker im Rahmen der Kalibrierung im Kapitel 2.2.8 zum Einsatz.

### 2.2.4 ArUco-Marker

ArUco-Marker sind quadratische Fiducial-Marker mit einem binären Gitter, deren Bitmuster eine eindeutige ID codieren [20]. Die Erkennung erfolgt über Schwellenwertbildung, Konturdetektion und Perspektivkorrektur des Vierecks, woraus die ID ausgelesen wird. Aus den vier Ecken des erkannten Markers wird mittels solvePnP, beschrieben in Kapitel 2.2.3, die Pose im Kamerakoordinatensystem berechnet.

Ein bekannter Nachteil planarer Marker ist ihre Anfälligkeit auf gekrümmten Oberflächen. Die quadratische Geometrie wird durch die Krümmung verzerrt, was die Erkennung ab moderaten Blickwinkeln stark beeinträchtigt [21]. In dieser Arbeit werden ArUco-Marker abschliessend nur noch auf den flachen Säulenoberseiten zur Kalibrierung eingesetzt (Kapitel 3.4.1), wo diese Einschränkung nicht auftritt.

### 2.2.5 Farbbasierte Bildsegmentierung

Die farbbasierte Segmentierung dient dazu, Objekte anhand ihrer Farbe vom Hintergrund zu trennen. In dieser Arbeit wird sie sowohl für die Erkennung des Schlauchanfangs im Kapitel 2.2.6, als auch für die Segmentierung des Schlauchkörpers bei der Erkennung, an welcher Stelle der Schlauch feststeckt eingesetzt, welche in Kapitel 2.2.7 beschrieben wird.

#### 2.2.5.1 HSV-Farbraum

Der Hue, Saturation, Value (HSV)-Farbraum trennt Farbton (H), Sättigung (S) und Helligkeit (V) in separate Kanäle. Diese Trennung macht die Farbsegmentierung robuster gegenüber Beleuchtungsschwankungen, Schatten und Reflexionen als eine Segmentierung im Red, Green, Blue (RGB)-Farbraum [22]. Durch Definition eines Wertebereichs für H, S und V lässt sich ein bestimmter Farbbereich als binäre Maske segmentieren [23]. In dieser Arbeit werden auf diese Weise sowohl der Schlauchkörper als auch die farbigen Ringmarkierungen am Schlauchanfang erkannt.

#### 2.2.5.2 Morphologische Operationen

Morphologische Operationen verarbeiten binäre Masken anhand eines Strukturelements. Opening (Erosion gefolgt von Dilation) entfernt kleine Rauschpixel, Closing (Dilation gefolgt von Erosion) schliesst Lücken in der Maske [24]. In dieser Arbeit wird nach jeder HSV-Segmentierung ein Opening und Closing mit einem elliptischen Strukturelement angewendet, um eine saubere Schlauchmaske zu erhalten.

### 2.2.6 Konturanalyse und Ellipsen-Fitting

Aus einer binären Maske lassen sich durch Konturextraktion die Umrisse segmentierter Objekte als Punktfolgen gewinnen [25]. Die Zirkularität einer Kontur beschreibt, wie kreisförmig sie ist:

$$C = \frac{4\pi A}{P^2}$$

Dabei bezeichnen $A$ die Fläche und $P$ den Umfang der Kontur; ein perfekter Kreis ergibt $C = 1$. Durch Ellipsen-Fitting wird an eine Kontur mit mindestens fünf Punkten eine Ellipse im Least-Squares-Verfahren angepasst, deren Mittelpunkt subpixelgenau bestimmt ist [26]. Da sich ein kreisförmiges Merkmal unter perspektivischer Projektion als Ellipse abbildet, bleibt die Form auch bei schräger Betrachtung segmentierbar [21]. In dieser Arbeit werden auf diese Weise die beiden Ringmarkierungen am Schlauchanfang lokalisiert.

### 2.2.7 Skelettierung

Die Skelettierung reduziert eine binäre Maske auf eine ein Pixel breite Mittellinie, die die topologische Struktur der ursprünglichen Form bewahrt. Das Verfahren entfernt schrittweise Randpixel, solange die Form und der Zusammenhang der Struktur erhalten bleiben. Dadurch bleibt am Ende nur die Mittellinie (das Skelett) übrig [27].

#### 2.2.7.1 Kubische Bézier-Kurven

Eine kubische Bézier-Kurve interpoliert zwischen zwei Endpunkten $P_0$ und $P_3$ über zwei Kontrollpunkte $P_1$ und $P_2$, die die Tangentenrichtung an den Endpunkten bestimmen:

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3, \quad t \in [0,1]$$

Die Kurve beginnt in $P_0$ mit Tangente $P_1 - P_0$ und endet in $P_3$ mit Tangente $P_3 - P_2$. In dieser Arbeit werden Bézier-Kurven eingesetzt, um verdeckungsbedingte Lücken im Skelett zu überbrücken. Die Kontrollpunkte werden dabei aus der lokalen Tangentenrichtung der Skelett-Fragmente an den jeweiligen Endpunkten abgeleitet.

#### 2.2.7.2 Breitensuche

Die Breadth-First Search (BFS) ist ein Algorithmus zur Graph-Traversierung, der ausgehend von einem Startknoten alle erreichbaren Knoten schichtweise besucht. Eine Warteschlange (Queue) verwaltet die noch zu besuchenden Knoten. In jedem Schritt wird der vorderste Knoten entnommen, seine unbesuchten Nachbarn werden mit Distanz $d + 1$ markiert und hinten an die Queue angefügt:

$$d(v) = d(u) + 1, \quad \forall v \in N(u)$$

Dabei bezeichnet $N(u)$ die Nachbarmenge des aktuellen Knotens $u$, wobei bereits besuchte sowie bereits in der Queue befindliche Knoten ausgeschlossen werden. Die BFS garantiert, dass jeder Knoten mit der kürzesten Distanz zum Startknoten erreicht wird. In dieser Arbeit wird die BFS auf dem Skelett des Schlauchs ausgehend vom erkannten Schlauchanfang ausgeführt. Das Skelett liefert zwar die Mittellinie des Schlauchs, kodiert jedoch keine Information darüber, welcher Punkt näher am definierten Anfang und welcher näher am Ende liegt. Die Distanzkarte der BFS ordnet jedem Skelett-Pixel seinen Abstand vom Schlauchanfang zu und macht diese Reihenfolge explizit auswertbar.

### 2.2.8 Kamera-Roboter-Kalibrierung

Die Kamera-Roboter-Kalibrierung bestimmt die Transformation zwischen dem Kamerakoordinatensystem und dem User Frame 5 (UF5). In dieser Arbeit erfolgt die Kalibrierung zweistufig: Eine starre Transformation via Singulärwertzerlegung (SVD) und eine nicht-starre Korrektur via Thin-Plate-Spline (TPS).

#### 2.2.8.1 Starre Transformation via SVD

Gegeben seien $n$ Korrespondenzpaare aus Kamerakoordinaten $\{s_i\}$ und UF5-Koordinaten $\{d_i\}$. Gesucht ist die starre Transformation (Rotation $R$ und Translation $t$), die die Summe der quadratischen Abstände zwischen transformierten Quell- und Zielpunkten minimiert [28]. Dazu werden beide Punktmengen durch Subtraktion ihres jeweiligen Schwerpunkts zentriert, sodass der Mittelpunkt jeder Punktmenge anschliessend im Ursprung liegt:

$$\overline{s} = \frac{1}{n}\sum_{i=1}^{n} s_i, \quad \overline{d} = \frac{1}{n}\sum_{i=1}^{n} d_i$$

Aus den zentrierten Punkten wird die Kreuzkovarianzmatrix berechnet:

$$H = \sum_{i=1}^{n} (s_i - \overline{s})\left(d_i - \overline{d}\right)^T$$

Die SVD von $H$ liefert $H = U\Lambda V^T$, woraus sich Rotation und Translation ergeben:

$$R = VU^T, \quad t = \overline{d} - R\overline{s}$$

Falls $\det(R) < 0$, wird die letzte Spalte von $V$ negiert, um eine Spiegelung auszuschliessen.

#### 2.2.8.2 Nicht-starre Korrektur via TPS

Ein TPS interpoliert glatt zwischen verstreut liegenden Stützpunkten mit minimaler Biege-Energie [29]. Die Interpolationsfunktion hat die Form:

$$f(x) = a_0 + a^T x + \sum_{i=1}^{N} w_i \varphi(\|x - x_i\|)$$

Dabei bezeichnet $\varphi(r) = r^2 \ln(r)$ die radiale Basisfunktion im zweidimensionalen Fall, $x_i$ die Stützstellen, $w_i$ die zugehörigen Gewichte und $a_0, a$ den affinen Anteil. In dieser Arbeit werden die nach der starren Transformation verbleibenden Restfehler an den Säulenpositionen als Stützstellen verwendet. An diesen Stellen verschwindet der Fehler exakt, dazwischen wird glatt interpoliert.

## 2.3 Pfadplanung

Die Pfadplanung bestimmt einen kollisionsfreien Weg von einem Start- zu einem Zielpunkt durch eine Umgebung mit Hindernissen. In dieser Arbeit wird die Arbeitsfläche des Labyrinths als zweidimensionales Gitter in endlich viele, gleich grosse Zellen aufgeteilt, auf dem ein gewichteter Suchalgorithmus den optimalen Pfad berechnet.

### 2.3.1 Occupancy Grid

Ein Occupancy Grid unterteilt die Umgebung in gleichmässige Zellen, die jeweils als frei oder belegt markiert werden [30]. In dieser Arbeit wird die Arbeitsfläche in ein Raster mit fester Zellgrösse unterteilt. Jede Säule wird mit einem Sicherheitsradius aufgebläht und die entsprechenden Zellen als Hindernis markiert.

### 2.3.2 Distance Transform

Die Distance Transform berechnet für jede freie Zelle eines Occupancy Grids den euklidischen Abstand zum nächsten Hindernis [31]. In dieser Arbeit dient sie als Grundlage für die Kostenfunktion des Suchalgorithmus: Zellen nahe an Hindernissen erhalten höhere Kosten, sodass der resultierende Pfad bevorzugt in der Korridormitte verläuft.

### 2.3.3 A-Star-Algorithmus

Der A-Star-Algorithmus (A*) ist ein informierter Suchalgorithmus, der den kürzesten Pfad in einem gewichteten Graphen findet [32]. Er kombiniert die bisherigen Kosten $g(n)$ vom Start zum aktuellen Knoten mit einer Heuristik $h(n)$, die die Restkosten zum Ziel schätzt. Die resultierende Bewertungsfunktion $f(n)$ gibt die geschätzten Gesamtkosten des Pfades durch Knoten $n$ an:

$$f(n) = g(n) + h(n)$$

Ist $h(n)$ zulässig (überschätzt die tatsächlichen Restkosten nie), garantiert der A* die optimale Lösung. In dieser Arbeit wird die euklidische Distanz als Heuristik verwendet und die Kostenfunktion $g(n)$ um einen Aufschlag aus der Distance Transform erweitert, der Pfade nahe an Säulen verteuert.

### 2.3.4 Pfadvereinfachung nach Ramer-Douglas-Peucker

Der Ramer-Douglas-Peucker (RDP)-Algorithmus reduziert die Anzahl der Punkte eines Pfades, indem er rekursiv Punkte entfernt, deren Abstand zur Verbindungslinie zwischen Start und Ende unterhalb eines Schwellenwerts $\varepsilon$ liegt [33]. Das Ergebnis ist ein vereinfachter Pfad, der nur noch die wesentlichen Eckpunkte enthält.

## 2.4 Manipulation deformierbarer Objekte

Die Manipulation starrer Körper mit Robotern ist ein weitgehend gelöstes Problem. Die Handhabung deformierbarer Objekte stellt hingegen eine offene Forschungsfrage dar, da sich deren Form während der Manipulation verändert und die Planung entsprechend angepasst werden muss [12].

### 2.4.1 Deformierbare lineare Objekte

DLO sind Objekte, deren Ausdehnung in einer Dimension deutlich grösser ist als in den beiden anderen, wie Kabel, Schläuche, Seile oder Drähte. Ihre Manipulation ist besonders anspruchsvoll, da sie unendlich viele DoF besitzen und ihr Verformungsverhalten von Materialeigenschaften wie Steifigkeit, Elastizität und Reibung abhängt [7]. In dieser Arbeit wird der zu handhabende Schlauch als DLO betrachtet.

### 2.4.2 Greifstrategie und Kraftüberwachung

Beim Greifen eines DLO muss die Greifkraft so gewählt werden, dass das Objekt sicher gehalten wird, ohne es zu beschädigen oder übermässig zu verformen. Bei kraftgeregelten Greifern wird dazu ein Kraftlimit vorgegeben, bei dessen Erreichen der Greifer das weitere Schliessen beendet [12]. Zusätzlich kann ein externer Kraft-Moment-Sensor eingesetzt werden, um während der Manipulation Kontaktkräfte zu überwachen und auf unerwartete Widerstände zu reagieren.

## 2.5 Kraftmessung und -überwachung

Die Kraftmessung ermöglicht es dem Roboter, während der Manipulation Kontaktkräfte zu erfassen und auf unerwartete Widerstände zu reagieren [12].

### 2.5.1 Kraft-Moment-Sensorik

Ein Kraft-Moment-Sensor misst die auf ihn wirkenden Kräfte und Drehmomente in bis zu sechs DoF ($F_x$, $F_y$, $F_z$, $T_x$, $T_y$, $T_z$). In der Robotik wird er typischerweise zwischen Roboter-Werkzeugflansch (Flansch) und Endeffektor montiert, um Kontaktkräfte während der Manipulation zu erfassen [12].

### 2.5.2 Schwellenwert-basierte Überlasterkennung

Bei der schwellenwertbasierten Überlasterkennung wird zunächst im Ruhezustand eine Baseline der Kraftwerte aufgenommen. Während des Betriebs wird die Abweichung der aktuellen Messwerte von dieser Baseline überwacht. Überschreitet die Abweichung einen definierten Schwellenwert, wird dies als Überlast gewertet:

$$|F_{\text{mess}} - F_{\text{idle}}| > F_{\text{schwelle}}$$

In dieser Arbeit wird dieses Prinzip eingesetzt, um während der Navigation festzustellen, ob der Schlauch an einer Säule fest hängt.

# 3 Vorgehen und Methoden

Aus den allgemeinen Konzepten des vorigen Kapitels wird in diesem Schritt eine konkrete Lösung. Beschrieben werden Versuchsaufbau und eingesetzte Hardware, die Architektur der entwickelten Software sowie die einzelnen Iterationsschritte, die im Projektverlauf zu den finalen Entscheidungen geführt haben.

## 3.1 Versuchsaufbau

Um die Komplexität eines DLO abbilden zu können, muss es Wege und Hindernisse geben, durch welche der Roboterarm manövrieren kann und die zudem auch verdeutlichen welche Problemstellungen sich durch das Handhaben eines DLO ergeben können. Des Weiteren muss der Aufbau durch ein Kamerasichtfeld abgedeckt werden um den Versuchsaufbau und die Position des Schlauches an den Roboter zu übergeben. Dies hat zu der Entscheidung geführt ein Labyrinth aus Säulen auf einem Holzbrett zu entwerfen, durch welches der Roboterarm den Schlauch führen muss.

### 3.1.1 Experimenteller Aufbau

Wie Abbildung 1 zeigt, besteht das Labyrinth aus einer Spanplatte mit den Abmessungen 72 x 72 cm und ist auf dem Unterbauwagen des Roboterarms fest montiert. Auf der Spanplatte sind 10 cm hohe Holzsäulen mit Schrauben montiert, jede mit einem Durchmesser von 2 cm. Die Säulen sind in einem Rechteck angeordnet das theoretisch aus 6 x 6 Säulen besteht, welche jeweils 10 cm Abstand zueinander haben. Praktisch fehlt die erste Säule, da diese den definierten Nullpunkt des Systems darstellt. Der Nullpunkt des Systems ist das definierte Ziel, zu dem der Schlauchanfang geführt werden muss, während der Roboterarm durch das Labyrinth aus 35 Säulen manövriert. Jede Säule hat einen ID-Marker, welcher auf die Oberseite der Säule aufgeklebt wurde, um dessen Position eindeutig zu definieren. Der Versuchsaufbau wird mit einer Kamera, welche an einem Stativ gegenüber dem Roboterarm auf der Höhe von 1 Meter befestigt ist, gefilmt. Der Roboterarm selbst hat einen Kraft-Sensor sowie einen Greifer montiert. Die in Abbildung 2 abgebildeten Schläuche, die in diesem Versuch den DLO entsprechen, bestehen aus weichem Silikon und sind hohl. Beide Schläuche haben einen Durchmesser von 1,2 cm und haben am definierten Anfang eine Markierung aus blauem Isolierband sowie darunter angebracht nochmals eine mit weissem Isolierband angebrachte Referenz-Markierung. Diese werden später zum Erkennen des Anfangs und der Ausrichtung der Schläuche benötigt.



Abbildung 1: Versuchsaufbau

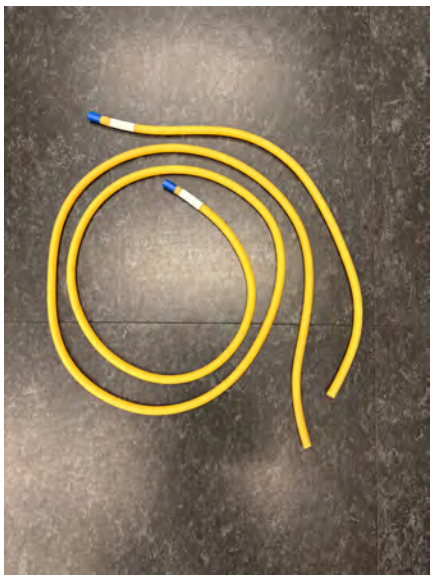


Abbildung 2: Kurzer und langer gelber Schlauch aus weichem Silikon

### 3.1.2 Hardwarekomponenten

Im Folgenden werden die Hardwarekomponenten des Versuchs näher betrachtet und im Detail deren Einfluss und Mehrwert für den Versuch aufgezeigt, sowie deren technische Grenzen.

#### 3.1.2.1 Roboter

Der in diesem Versuch eingesetzte Roboter, dargestellt in Abbildung 3, ist ein FANUC CR-7iA/L, ein kompakter, kollaborativer Industrieroboter mit sechs Rotationsachsen. Er hat ein Eigengewicht von 55 kg und verfügt über eine maximale Nutzlast von 7 kg am Flansch. Seine Reichweite beträgt 911 mm und er hat eine Wiederholgenauigkeit von ±0,01 mm [34]. Der Roboter ist nach ISO 10218-1 als kollaborativer Roboter zertifiziert [35] und kann ohne Schutzeinhausung neben Menschen betrieben werden, da er bei Kontakt automatisch stoppt. Die Steuerung erfolgt über eine FANUC R-30iB Mate Plus Einheit. Der Roboter war bereits auf einem stabilen Unterbauwagen mit Aluminiumprofilen und Bremseinrichtung vormontiert.

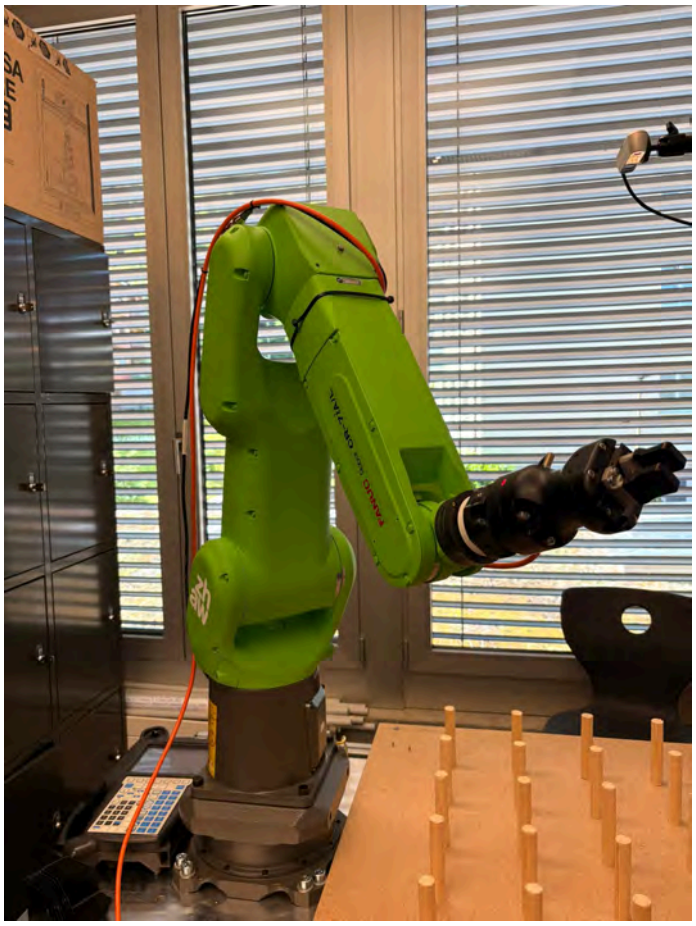


Abbildung 3: FANUC Robot CR-7iA/L

#### 3.1.2.2 Greifer

Als Endeffektor dient der in Abbildung 4 abgebildete Hand-E Greifer der Firma Robotiq. Der Greifer kommuniziert über eine serielle RS485-Schnittstelle und wird über ein externes 24V/1A-Netzteil mit Strom versorgt [36]. Für die Montage an den FANUC-Flansch wurde eine Adapterplatte konstruiert und 3D-gedruckt, um die unterschiedlichen ISO-Flanschmuster zu verbinden.




Abbildung 4: ROBOTIQ Hand-E Greifer

#### 3.1.2.3 Kraft-Sensor

Zwischen Flansch und Greifer ist ein in Abbildung 5 gezeigter Kraft-Moment-Sensor vom Typ Bota Systems BFT-SENS-IND2-B4 montiert. Der Sensor kommuniziert über eine serielle Schnittstelle, dem Bota Binary Protokoll, und misst unter anderem die auf den Greifer wirkenden Kräfte in X-, Y- und Z-Richtung [37].



Abbildung 5: Kraft-Moment-Sensor Bota Systems BFT-SENS-IND2-B4

#### 3.1.2.4 Kamera

Zur visuellen Erfassung des Versuchsaufbaus dient die in Abbildung 6 abgebildete Intel RealSense D435iF Stereokamera. Diese liefert sowohl RGB-Farbbilder, die für die Farbsegmentierung in den HSV-Farbraum konvertiert werden, als auch Tiefeninformationen über ein Stereo-Infrarot-Sensorpaar [38]. Die Kamera ist an einem Stativ auf einer Höhe von circa 1 Meter über dem Labyrinth befestigt und blickt senkrecht auf die Arbeitsfläche hinunter. Die Tiefenmessung ermöglicht es, die zweidimensionalen Pixelkoordinaten erkannter Objekte in dreidimensionale Raumpositionen umzurechnen. Aufgrund der begrenzten Höhenauflösung bei der gegebenen Montagehöhe eignet sich der Tiefensensor jedoch nur eingeschränkt für die präzise Bestimmung der Z-Koordinate auf der Arbeitsfläche. Die Höhe wurde gewählt damit der Roboterarm sich frei auf der Fläche bewegen kann, ohne mit der montierten Kamera zu kollidieren.

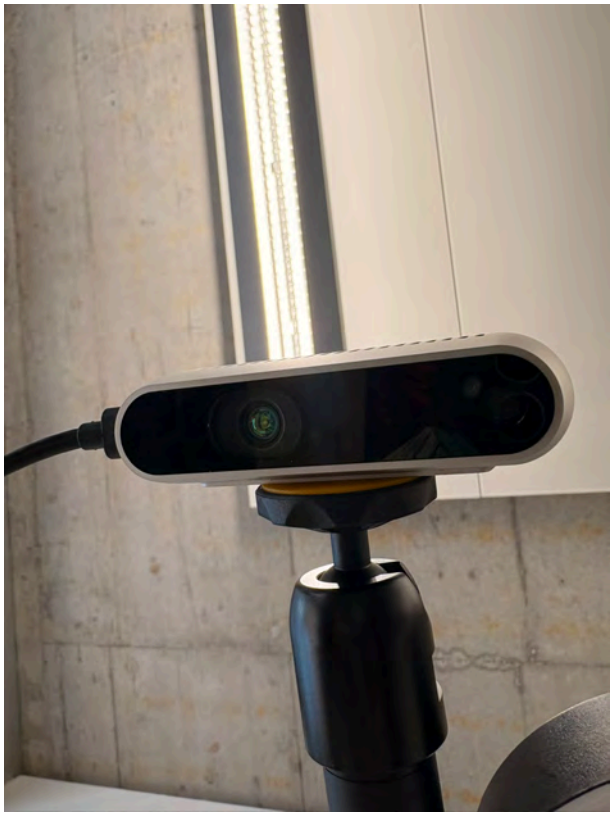


Abbildung 6: RealSense D435iF montiert an ein Meter hohem Stativ

### 3.1.3 Softwarekomponenten

Im Folgenden werden die im Projekt verwendeten Softwarekomponenten und externen Bibliotheken beschrieben. Die gesamte Projektlogik wurde in Python implementiert, wobei sowohl eigens entwickelte Module als auch etablierte Open-Source-Bibliotheken zum Einsatz kommen.

#### 3.1.3.1 FANUC Remote Motion Interface

Die Kommunikation mit dem Roboter erfolgt über das FANUC RMI, eine TCP/IP-basierte API, die als Softwareoption (R912) auf der Steuerung aktiviert ist. Als Grundlage dient der im Vorgängerprojekt entwickelte Python-Driver [16], über den JSON-kodierte Befehle per Socket an die Steuerung übermittelt sowie Positionen angefahren, Register gelesen und geschrieben und Systeminformationen abgefragt werden können. Im Rahmen dieser Arbeit wurde die Bibliothek strukturell bereinigt und um ein Modul zur Anbindung des Kraft-Sensors erweitert.

#### 3.1.3.2 Intel RealSense SDK

Die Ansteuerung der Kamera erfolgt über das Intel RealSense Software Development Kit (SDK) 2.0 `pyrealsense2`. Das SDK wird verwendet, um die Farb- und Tiefenstreams der Kamera zu starten, die intrinsischen Kameraparameter auszulesen und die Tiefendaten mit den Farbbildern zu synchronisieren.

#### 3.1.3.3 OpenCV & ArUco

Für die Bildverarbeitung wird OpenCV (opencv-python) eingesetzt. Die integrierte ArUco-Bibliothek ermöglicht die Erkennung und Identifikation der auf den Säulen angebrachten Marker. Mittels der Funktion `solvePnP()` wird aus der bekannten Markergrösse und deren Abbildung im Kamerabild die räumliche Pose relativ zur Kamera berechnet. Darüber hinaus wird OpenCV für die HSV-basierte Farberkennung des Schlauchs verwendet.

#### 3.1.3.4 Bota Systems Driver

Zur Kommunikation mit dem Kraft-Sensor wird der offizielle Bota Systems Python-Driver `bota-driver` verwendet. Dieser liest die Kraftwerte über das Bota Binary Protokoll via serielle Schnittstelle aus.

#### 3.1.3.5 scikit-image

Für die Ermittlung der Schlauchmittellinie wird die Funktion `skeletonize()` aus der Bibliothek `scikit-image` eingesetzt. Ausgangspunkt ist die mittels HSV-Farbmasking erzeugte binäre Maske des Schlauchs. Diese Maske weist an Stellen, an denen Säulen den Schlauch verdecken, Unterbrechungen auf. Durch die Skelettierung wird die Maske auf eine ein Pixel breite Mittellinie reduziert, wobei die Lücken überbrückt werden. Aus dem resultierenden Skelett werden anschliessend Richtungsvektoren berechnet, die zum Schlauchanfang zeigen.Diese Vektoren bestimmen die Schlauchorientierung. Daraus lassen sich die korrekten Greifposen berechnen und die Säulen erkennen, an denen der Schlauch feststeckt.

#### 3.1.3.6 Weitere Libraries

`NumPy` bildet die Grundlage für nahezu alle numerischen Operationen im Projekt, darunter Matrizenoperationen, Koordinatentransformationen und die Handhabung von Bilddaten als Arrays. `SciPy` wird für die Konvertierung zwischen Rotationsmatrizen, Euler-Winkeln und Quaternionen in der Kamera-Poseberechnung eingesetzt. Die Steuerung des Greifers erfolgt über das Modbus-RTU-Protokoll via serieller Schnittstelle, wofür die Bibliothek `pymodbus` verwendet wird.

## 3.2 Durchführung

In diesem Abschnitt wird die praktische Durchführung des Projekts beschrieben. Das Projekt wurde in drei Teilprojekte unterteilt. In den folgenden Unterkapiteln wird jeweils einmal das Ziel des jeweiligen Teilprojekts erklärt. Danach werden verschiedene Varianten oder Entscheidungen aufgezeigt, welche im Rahmen der Iterationsschritte ausprobiert wurden.

### 3.2.1 Teilprojekt 1: Erkennen des Schlauchs

Das Erkennen des Schlauchs stellt den ersten und grundlegenden Schritt der Schlauchführung dar. Der Anfang des Schlauchs muss zuverlässig erkannt werden und seine Ausrichtung muss bestimmt werden können.

#### 3.2.1.1 Lösungsansätze: Erkennen des Schlauchs

Für die Erkennung des Schlauchanfangs wurden im Laufe des Projekts mehrere Ansätze evaluiert:

1. **ArUco-basiert:** Ein erster Ansatz basierte auf einem kleinen ArUco-Marker (ID 1, 10 mm), der direkt auf den Schlauch geklebt wurde. Dieser erwies sich jedoch als unzuverlässig da, wegen der geringen Markergrösse und der Kamerahöhe von einem Meter, die Erkennung teilweise mehrere Minuten gedauert hat. Zudem knickte der Marker nach jedem Greifvorgang ein, wodurch er für weitere Erkennungsiterationen unbrauchbar wurde. Die zylindrische Oberfläche des Schlauchs verschärfte das Problem zusätzlich, da die flache Quadratgeometrie eines ArUco-Markers auf einer gekrümmten Fläche verzerrt wird und die Erkennung bereits ab moderaten Blickwinkeln stark beeinträchtigt wird.
2. **HSV-Farbmaske am Schlauchanfang:** Im Rahmen eines weiteren Ansatzes wurde am Schlauchanfang eine Markierung, in einer vom Schlauchkörper abweichenden Farbe, angebracht und diese über eine HSV-Farbmaske segmentiert. Aus dem Konturschwerpunkt der Markierung liess sich zwar eine Position des Anfangs ableiten, jedoch keine Orientierung. Eine einzelne Farbregion liefert nur einen einzelnen Bezugspunkt. Um jedoch eine Richtung zu ermitteln, in welcher präzise gegriffen werden soll, werden mindestens zwei unterscheidbare Punkte benötigt, zwischen denen ein Vektor aufgespannt werden kann. Die für eine korrekte Greifpose erforderliche Ausrichtung des Schlauchs liess sich so nicht zuverlässig ermitteln. Ein Greifen allein über die Position genügt nicht in diesem Versuchsaufbau.
3. **Random Sample Consensus (RANSAC):** Bei RANSAC erwies sich die Kombination aus grosser Kameradistanz und den Säulen als problematisch. Die jeweils 10 cm hohen Säulen haben im Punktwolkenbild ein zu hohes Rauschen im Verhältnis zum maximal 1 cm hohen Schlauch erzeugt, sodass dieser nicht zuverlässig vom Hintergrund separiert werden konnte. Ausserdem war die Auflösung der Kamera dafür auch grundsätzlich nicht hoch genug.
4. **YOLO:** YOLO wurde aufgrund des zeitlichen Aufwands verworfen, da die Erstellung eines projektspezifischen Datensatzes und das anschliessende Training des Modells im Rahmen dieser Arbeit nicht realisierbar war.

Aus den verworfenen Ansätzen ergaben sich zwei konkrete Anforderungen an eine robustere Lösung.

- Erstens muss die Markierung sowohl eine Position als auch eine Orientierung liefern, was über eine einzelne Farbregion nicht möglich ist, da sich aus einem einzelnen Punkt keine Richtung ableiten lässt.
- Zweitens muss die Markierung dem wiederholten Greifen standhalten, ohne wie ein aufgeklebter planarer Marker einzuknicken und dadurch unbrauchbar zu werden.

Die Umsetzung dieser Lösung wird im Kapitel 3.4.3 beschrieben.

### 3.2.2 Teilprojekt 2: Kurzer Schlauch durch das Labyrinth

Im zweiten Teilprojekt wird ein kurzer Schlauch durch das Labyrinth geführt. Aufgrund der geringen Länge ist sichergestellt, dass der Schlauch im Inneren des Labyrinths nicht hängen bleiben kann. Damit lässt sich der grundlegende Ablauf von der Erkennung über die Pfadplanung bis zum Ablegen am Zielpunkt isoliert validieren, ohne dass komplexere Effekte wie Steckenbleiben berücksichtigt werden müssen.

#### 3.2.2.1 Lösungsansätze: Kalibrierung des Koordinatensystems

Damit ein im Kamerabild erkannter Greifpunkt vom Roboter exakt angefahren werden kann, müssen Kamerakoordinatensystem und UF5 aufeinander abgestimmt sein. Die Kalibrierung erfolgt einmalig.

Ein erster Ansatz definierte den Nullpunkt des Systems über einen einzelnen, grossen ArUco-Marker, der lose in den Ursprung des Labyrinths gelegt wurde und das gesamte Kamerakoordinatensystem aufspannte. Dies erwies sich als unpraktisch, da der Marker dauerhaft im Sichtfeld der Kamera liegen musste. Zudem war er nicht von Anfang an fest fixiert, sodass seine Position nicht reproduzierbar war. Weil das gesamte Koordinatensystem auf dieser einzelnen Referenz beruhte, setzte sich deren Ungenauigkeit über die gesamte Arbeitsfläche fort, sodass der Roboter regelmässig danebengriff und mit den Säulen kollidierte.

Da die Säulen des Labyrinths ohnehin fest auf der Platte verschraubt sind, wird die Kalibrierung stattdessen auf alle 35 Säulen gestützt. Der Nullpunkt selbst trägt damit keinen Marker mehr, sondern wird rechnerisch aus den festen Säulenpositionen abgeleitet.

Die technische Umsetzung der Kalibrierung ist in Kapitel 3.4.1 beschrieben. Dazu gehören die Erkennung der Säulenpositionen durch Kamera und Roboter sowie die Korrektur des verbleibenden Restfehlers.

#### 3.2.2.2 Lösungsansätze: Greifen des Schlauchs

Beim Greifen des Schlauchs musste zuerst das folgende Problem gelöst werden: In der Implementation des Vorgängerprojektes basierte das Greifen neben einer Maximalkraft auch auf Positionen. Es musste angegeben werden bis zu welchem Abstand die Greifzangen zusammen fahren sollen. Da der Schlauch aus hohlem, weichem Silikon besteht und beim Zugreifen nachgibt, hatte das Schliessen auf eine feste Greiferposition keinen zuverlässigen Halt gegeben. Gelöst wurde das Problem, indem der Greifer vollständig schliesst, wobei der Robotiq Hand-E die Greifkraft eigenständig auf einen festen Maximalwert begrenzt und das Schliessen beendet, sobald dieser erreicht ist. Der Schlauch wird nun also zusammengequetscht bis dieser Maximalwert erreicht ist und so unabhängig von seinem Materialwiderstand mit gleichbleibender Kraft gegriffen, ohne beschädigt zu werden.

#### 3.2.2.3 Lösungsansätze: Navigation durch das Labyrinth

Ausgehend vom Greifpunkt berechnet der Pfadplaner mit dem A* einen kollisionsfreien Weg zum Nullpunkt. Die Implementation dieses Programmabschnittes hat schon bei der ersten Iteration gut funktioniert und die Lösung wird in Kapitel 3.4.4 beschrieben.

### 3.2.3 Teilprojekt 3: Langer Schlauch mit Feststecken-Erkennung

Im dritten Teilprojekt wird ein langer Schlauch durch das Labyrinth geführt. Anders als beim kurzen Schlauch kann dieser zwischen den Säulen stecken bleiben. Dieses Teilprojekt erweitert den Ablauf um das Erkennen eines feststeckenden Schlauchs und das anschliessende Befreien.

#### 3.2.3.1 Lösungsansätze: Erkennen des Feststeckens

Um zu erkennen, ob der Schlauch feststeckt, wurde zwischen Flansch und Greifer ein Kraft-Moment-Sensor eingebaut. Bleibt der Schlauch während der Navigation an einer Säule hängen, steigt die auf den Greifer wirkende Zugkraft an. Überschreitet diese einen definierten Schwellenwert gegenüber dem zuvor kalibrierten Ruhezustand, wird dies als Feststecken gewertet und die laufende Navigation unterbrochen.

Ein erster Ansatz sah vor, die Roboterbewegung bei Überschreiten des Schwellenwerts sofort abzubrechen. Dies erwies sich jedoch als nicht umsetzbar, da der Bewegungsbuffer der Robotersteuerung sich während der Programmausführung nicht zuverlässig leeren liess, was einen Neustart erfordert hätte. Stattdessen wird die Navigation in kurze Wegabschnitte unterteilt, zwischen denen die Kraft geprüft wird. Wird die Schwelle überschritten, bricht die Bewegung nach dem aktuellen Abschnitt kontrolliert ab. Das Kapitel 3.4.7 erklärt diese Lösung im Detail.

#### 3.2.3.2 Lösungsansätze: Lokalisieren der Problemstelle

Steckt der Schlauch fest, muss bestimmt werden, an welcher Säule er hängen geblieben ist. Diese Stelle wird im Folgenden als **Problemstelle** bezeichnet. Da ein langer Schlauch an mehreren Säulen gleichzeitig hängen bleiben kann, genügt es nicht, eine einzelne Problemstelle zu finden. Alle betroffenen Säulen müssen entlang des Schlauchs in ihrer Reihenfolge bekannt sein, damit zuerst die hinterste, also die vom Kopf des Schlauches am weitesten entfernte, gelöst werden kann.

Einzelne Problemstellen liessen sich rasch erkennen, die eigentliche Schwierigkeit lag in ihrer Reihenfolge. Ein erster Ansatz bestimmte nur den Schlauchanfang über seine Markierung. Damit war zwar der Startpunkt bekannt, der weitere Verlauf des Schlauchs und damit die Reihenfolge der betroffenen Säulen jedoch nicht. In einem zweiten Ansatz wurde deshalb der gesamte Schlauch skelettiert und vom Anfang aus durchlaufen, sodass sich für jeden Punkt seine Lage relativ zum Anfang ergibt und die festsitzenden Säulen entlang des Verlaufs geordnet werden können.

Beim Ausprobieren zeigte sich allerdings, dass eine reine Lücken-Suche im Schlauch nicht in allen Situationen funktioniert. Durch das Zurückspicken des Schlauchs verschwinden teilweise die Lücken, an denen eine Problemstelle erkannt wird, und ein Feststecken ist grundsätzlich auch ganz ohne sichtbare Lücke möglich. Daraus sind im Laufe des Projekts mehrere Erkennungsstrategien entstanden, die nacheinander ausprobiert wurden und im Folgenden als **Tiers** bezeichnet werden:

1. **Tier 1, Lücken-Suche:** Im gemittelten Schlauchbild wurden Stellen gesucht, an denen direkt bei einer Säule eine vollständige Lücke im Schlauchverlauf sichtbar ist. Das funktioniert, wenn der Schlauch sichtbar an einer Säule abknickt, aber genau das ist nach einem Zurückspicken oft nicht mehr der Fall.
2. **Tier 2, Säulen-Sampling:** Da bei einer angepressten Säulenkante die HSV-Maske an dieser Stelle «flackert», wurden gezielt drei Punkte links, rechts und mittig direkt an der Säule abgetastet. War die Mitte deutlich schwächer bedeckt als die seitlichen Punkte, wurde die Säule als Problemstelle gewertet. Das hat für Situationen, in denen der Schlauch ganz an einer Säule anliegt, sauber funktioniert, war aber sehr empfindlich auf die HSV-Schwellen und die genaue Lage der Sample-Punkte.
3. **Tier 3, Skelett-Proximity (Near):** Statt nach Lücken oder einem Druckmuster zu suchen wurde geprüft, ob das Schlauch-Skelett irgendwo näher als 20 mm an einer Säule vorbeiläuft. Das hat den Vorteil, dass es weder eine sichtbare Lücke noch eine exakte Säulenkante braucht, der Schlauch muss nur in der Nähe der Säule liegen.

4. **Tier 4, Fallback:** Greift keine der vorherigen Stufen, wird eine Säule mit genug Schlauch dahinter gewählt, welche am weitesten vom Kopf des Schlauches entfernt liegt. Dadurch kann von ganz hinten mehr Schlauch ins Labyrinth nachgegeben werden, was oft dazu führt, dass sich potenzielle Problemstellen im Inneren des Labyrinths entspannen können.

Welche dieser Stufen sich in der Praxis bewähren und warum das vollständige Tiersystem verworfen und sich für ein System lediglich aus Tier 3 und Fallback entschieden wurde, wird in Kapitel 4.4 durch konkrete Messungen erklärt. Der detaillierte Ablauf der finalen Implementierung wird in Kapitel 3.4.7 beschrieben.

#### 3.2.3.3 Lösungsansätze: Befreien des Schlauchs

Steht der Greifpunkt an der zu lösenden Problemstelle fest, greift der Roboter den Schlauch dort und schiebt ihn entlang des Korridors ein Stück weiter ins Labyrinth nach, sodass die Spannung nachlässt. Diese grundsätzliche Idee hat nach etwas Fine-Tuning gut funktioniert, weshalb hierfür keine weiteren Ideen ausprobiert wurden. Die Umsetzung wird in Kapitel 3.4.7 beschrieben.

Beim Ausprobieren landete der Schlauchanfang nach dem Loslassen jedoch immer wieder ausserhalb des Labyrinths oder zu nahe an einer Säule und war dadurch nicht mehr greifbar oder erkennbar. Aus diesem Problem entstand einerseits eine Greifpunkt-Prüfung, die jeden Greifpunkt vor dem Anfahren auf Erreichbarkeit kontrolliert und ihn bei Bedarf verschiebt welche in Kapitel 3.4.8.1 beschrieben wird, und andererseits eine Fallback-Methode, die den Kopf des Schlauchs auch dann erkennt, wenn er nicht ganz im Sichtfeld liegt, beschrieben in Kapitel 3.4.8.2.

## 3.3 Lösungsbeschreibung

Die im Projektverlauf entstandene Software gliedert sich in zwei Schichten. Eine hardwareunabhängige Abstraktionsebene bildet das Fundament, darüber liegt die projektspezifische Anwendungslogik. Diese Trennung erwies sich als notwendig, weil sowohl Hardware-Treiber als auch einzelne Algorithmen im Lauf der Iterationen mehrfach ausgetauscht werden mussten.

### 3.3.1 Top-Level-Architektur

Auf oberster Ebene gliedert sich das Projekt in zwei Schichten, die Anwendungslogik und eine darunterliegende Abstraktionsschicht. Innerhalb der Abstraktionsschicht stehen vier voneinander unabhängige Pakete nebeneinander, die jeweils ein Hardwaresystem abstrahieren. Die Abbildung 7 zeigt die beiden Schichten und ihre Abhängigkeiten.

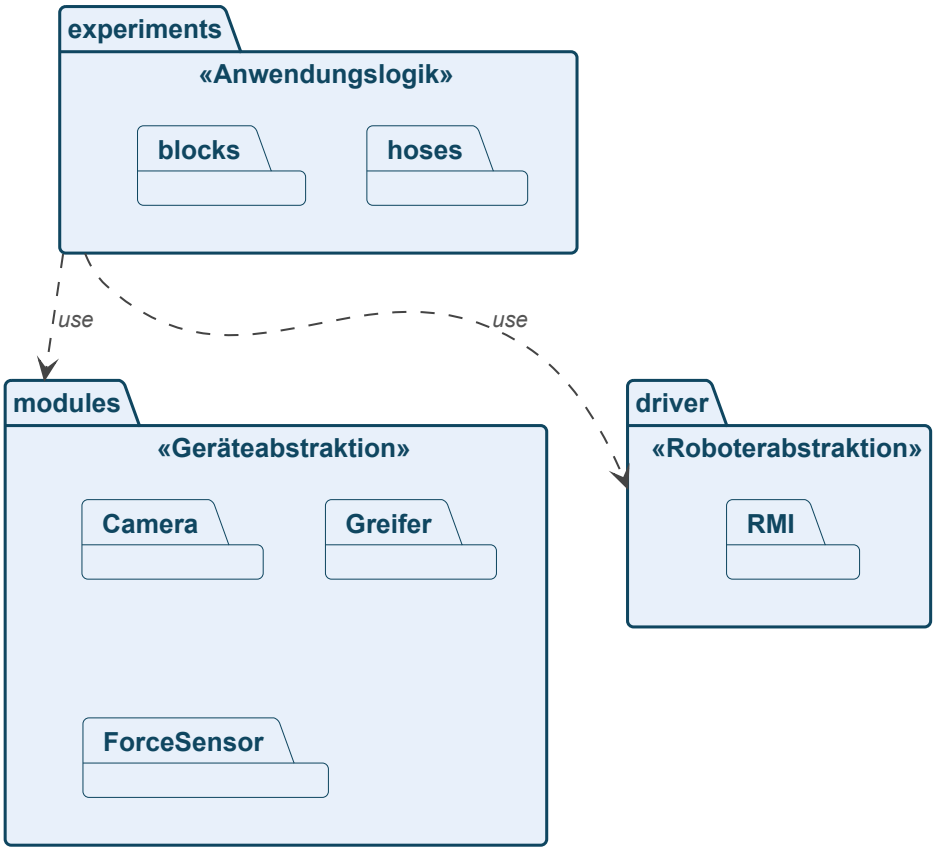



Abbildung 7: Top-Level-Architektur

Die Anwendungslogik liegt im Ordner `experiments` und enthält die konkreten Projekte. Neben dem Projekt `hoses` ist auch das Vorgängerprojekt `blocks` darin angesiedelt. Sie orchestriert den jeweiligen Versuchsablauf und greift dafür auf die Abstraktionen zu.

Die Abstraktionsschicht umfasst vier eigenständige Klassen, die jeweils einSubsystem kapseln:

1. **Camera** abstrahiert die Intel RealSense Kamera und liest Farb- sowie Tiefenstreams über das Intel RealSense SDK (`pyrealsense2`) aus.
2. **Greifer** steuert den Robotiq Hand-E Greifer über das Modbus-RTU-Protokoll auf einer seriellen RS485-Schnittstelle (`pymodbus`).
3. **ForceSensor** liest den Bota-Kraftsensor über das Bota Binary Protokoll auf einer seriellen Schnittstelle aus (`bota-driver`).
4. **RMI** kommuniziert mit den Roboter per TCP/IP-Socket und tauscht RMI-Befehle als JSON aus.

Jede dieser Klassen kommuniziert direkt mit ihrer Gegenstelle und ist von den übrigen Abstraktionen unabhängig. Die zugrunde liegenden Hersteller-Bibliotheken und Wire-Level-Protokolle bilden konzeptuell eine dritte Ebene unterhalb dieser Schicht, sie werden durch die Klassen vollständig verborgen und sind im Diagramm bewusst nicht dargestellt.

Die Aufteilung dieser vier Abstraktionen auf die Ordner `modules` und `driver` ist rein organisatorisch und folgt keiner architektonischen Trennung. `Camera`, `Greifer` und `ForceSensor` sind je in einer einzelnen Datei umgesetzt und teilen sich daher den Ordner `modules`. Die RMI-Abstraktion besteht hingegen aus mehreren Dateien, neben `RMI_Class` gehören `RMI_Command_Library`, `RMI_Datatypes`, `RMI_Enums` und `rmi_error_codes` dazu, und liegt deshalb in einem eigenen Ordner `driver`. Die Implementierung der RMI-Abstraktion stammt aus dem Vorgängerprojekt [16] und wurde im Rahmen dieser Arbeit nur strukturell bereinigt.

### 3.3.2 Module im Schlauchprojekt

Als grundliegende Architektur wurde ein Orchestrator-Muster gewählt. Die Klasse `HoseMazeApp` in `app.py` instanziiert die übrigen Module, hält die zugehörigen Objekte und ruft sie in der definierten Reihenfolge auf. Die projektbezogenen Module kapseln oder erweitern die in der Abstraktionsschicht bereits gekapselten Geräte erneut. `RobotActions` umschliesst `RMI` und `Greifer`, `VisionSystem` umschliesst `Camera`, und die Klasse `ForceSensor` erweitert die gleichnamige Basis-Klasse aus `modules` per Vererbung. Die Abbildung 8 zeigt die Module von `experiments/hoses` und ihre Zuordnung zu den darunterliegenden Basis-Abstraktionen.

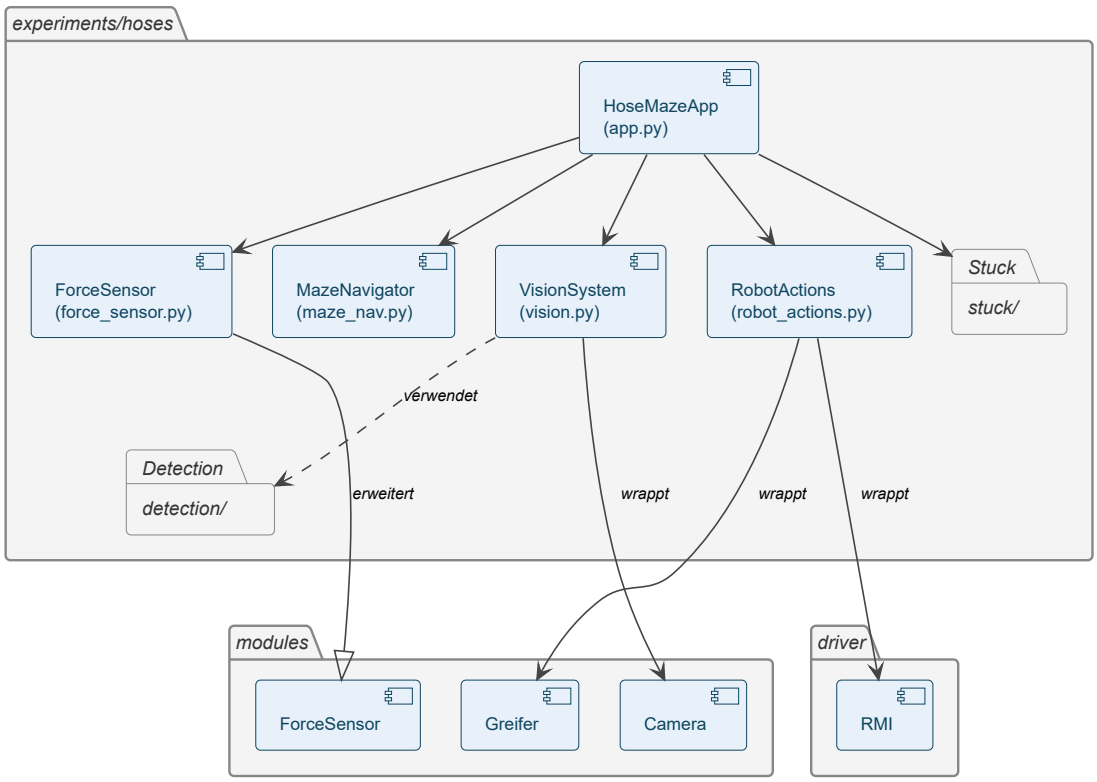



Abbildung 8: Modulübersicht innerhalb `experiments/hoses`

Diese zweite Schicht ist eine bewusste Designentscheidung. Ursprünglich war die Idee, alle Funktionen direkt in die allgemeinen Abstraktionsklassen aufzunehmen, damit andere Projekte sie unverändert weiterverwenden können. Auch das Vorgängerprojekt `blocks` hatte den Roboter mit einer eigenen Klasse `RoboterSystem_Class` ein weiteres Mal abstrahiert. Beim Versuch, deren Funktionen für die Schlauchführung zu generalisieren, stellte sich jedoch heraus, dass praktisch nichts wiederverwendbar war, weil die enthaltene Logik vollständig block-spezifisch ist. Eine immer weiter wachsende allgemeine Klasse wäre die Folge gewesen, gefüllt mit Methoden, die ausserhalb des jeweiligen Projekts niemals zum Einsatz kommen. Stattdessen wurden die benötigten Bewegungs- und Erkennungs-Funktionen in eigene Wrapper-Klassen innerhalb von `experiments/hoses` ausgelagert. Innerhalb eines Projekts können sie so von allen Modulen einheitlich genutzt werden, ohne die wiederverwendbaren Basis-Abstraktionen mit Logik aufzublähen, die nur von einem Projekt verwendet werden kann.

Konkret übernehmen die Wrapper folgende Aufgaben:

1. `RobotActions` ergänzt `RMI` und `Greifer` um Bewegungssequenzen für den das Navigieren im Labyrinth sowie um die verwendeten Greif-Parameter.
2. `VisionSystem` kombiniert `Camera` mit einem austauschbaren Detektor aus dem Paket `detection/`, lädt die kalibrierte Koordinatentransformation auf die Kamera und stellt eine `detect_hose()`-Methode mit Timeout und optionaler Live-Vorschau bereit.
3. Die Klasse `ForceSensor` in `force_sensor.py` erbt von der Basis-Klasse und ergänzt die Methode `is_overloaded()`, die Fx und Fy gegen die kalibrierte Idle-Baseline und einen Schwellenwert vergleicht.

Daneben enthält `experiments/hoses` drei Module ohne direkten Bezug zu einer Basis-Abstraktion: `MazeNavigator` für die A*-Pfadplanung, das Paket `detection/` mit den austauschbaren Erkennungsalgorithmen und das Paket `stuck/` mit der Recovery-Logik.

## 3.4 Detailbeschreibung der Module

Die folgenden Module bilden gemeinsam einen Durchgang der Schlauchführung. Sie werden in der Reihenfolge beschrieben, in der sie zur Laufzeit zum Zug kommen. Den Anfang macht die einmalige Kalibrierung des Koordinatensystems.

### 3.4.1 Kalibrierung des Koordinatensystems

Bevor der Roboter den Schlauch greifen kann, müssen das Kamerakoordinatensystem und das UF5 aufeinander abgestimmt werden. Dieser Schritt wird einmalig durchgeführt und muss nur wiederholt werden, wenn die Kamera physisch bewegt wird.

Im ersten Schritt wird das Skript `scan_pillars.py` ausgeführt. Die Kamera erkennt die auf den Säulen aufgeklebten ArUco-Marker (IDs 3-37, 10 mm) und berechnet mittels `solvePnP()` die dreidimensionale Position jedes Markers im Kamerakoordinatensystem. Um eine stabile Messung zu gewährleisten, wird jede Säulenposition über 30 aufeinanderfolgende Frames gemittelt. Das Ergebnis wird als Rohdaten im Kamerakoordinatensystem in der Datei `pillar_positions.json` gespeichert.

Im zweiten Schritt wird das Skript `calibrate_origin.py` ausgeführt. Der Roboter wird manuell per Teach-Pendant zu mindestens vier Säulen gefahren, wobei jeweils die zugehörige Säulen-ID eingegeben wird; das System liest die aktuelle Roboterposition automatisch aus. Aus den so entstehenden Korrespondenzpaaren zwischen Kamera- und UF5-Koordinate wird per SVD eine 2D-Rigid-Body-Transformation berechnet, welche Rotation und Translation zwischen den beiden Koordinatensystemen beschreibt. Der Ursprung des Labyrinths wird nicht angefahren, sondern aus den gemessenen Positionen der Säulen ID 3 und ID 8 über einen fixen geometrischen Offset abgeleitet. Die resultierende Transformationsmatrix wird als `origin_transform.npy` gespeichert.

In einem dritten Schritt wird der nach der starren Transformation verbleibende Restfehler durch eine nicht-starre Korrektur ausgeglichen. Dazu werden die starr transformierten Säulenpositionen und die zugehörigen Roboter-Messpunkte als Stützstellen eines in Kapitel 2.2.8.2 beschriebenen TPS verwendet und in der Datei `origin_correction.npz` gespeichert. Beim Setzen von `origin_transform` auf der Kamera wird diese Korrektur automatisch geladen und auf die XY-Komponenten jeder bereits starr transformierten Position angewendet.

Wurde die Kamera verschoben genügt es, `scan_pillars.py` erneut auszuführen und anschliessend `calibrate_origin.py` mit der schnelleren Rekalibrierung zu starten. Die SVD wird dann aus den bereits gespeicherten Roboter-Messpunkten und den aktualisierten Kamerakoordinaten neu berechnet.

Beim Start lädt `main.py` die Datei `origin_transform.npy` und bricht den Programmablauf ab, falls diese nicht vorhanden ist.

### 3.4.2 Hauptablauf

In `app.py` ist der Hauptablauf des Versuchs implementiert. Von hier aus werden beim Start alle Subsysteme initiiert und verbunden, alle weiteren Schritte ausgeführt und am Ende sämtliche Verbindungen geschlossen. Die Abbildung 9 gibt einen Überblick über diesen Ablauf.

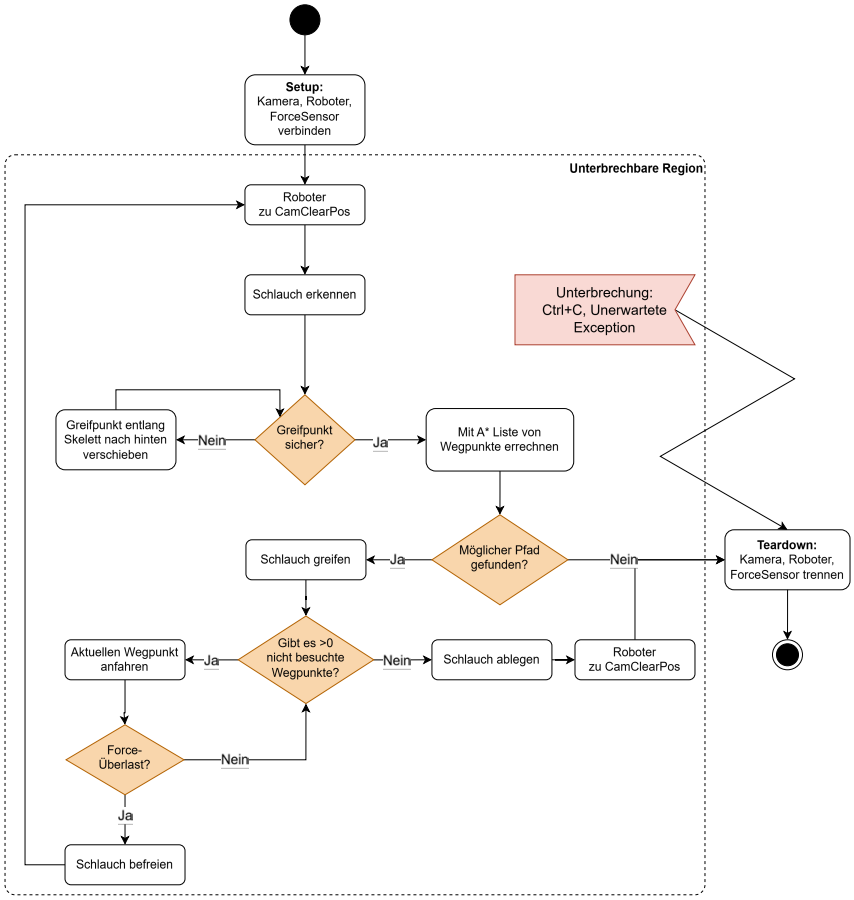



Abbildung 9: Hauptloop eines Schlauchführungszyklus

**Setup und Teardown:** Beim Start werden Kamera, Roboter und Kraftsensor verbunden sowie die Ruhe-Baseline des Kraftsensors in der aktuellen Pose kalibriert. Anschliessend fährt der Roboter in eine vordefinierte Ausgangsposition, aus der die Kamera ungehinderte Sicht auf das Labyrinth hat. Nach dem Beenden des Programmes werden alle Verbindungen in umgekehrter Reihenfolge geschlossen um zu verhindern, dass beispielsweise beim nächsten Startup der Buffer des Roboters noch voll ist oder sonstige Probleme entstehen.

**Schlauchführungszyklus:** Der Hauptloop führt pro Iteration einmal die in Abbildung 9 aufgezeigten Schritte in vorgegebener Reihenfolge aus. Auf die genauere Implementierung dieser Schritte wird in den folgenden Kapiteln näher eingegangen. Zum Überblick hier die Verweise:

1. Zu Beginn wird in einem Loop so lange nach einem Schlauchanfang gesucht, bis ein stabiler Greifpunkt erkannt werden konnte. Die reguläre Erkennung wird in Kapitel 3.4.3 erklärt, wenn dort Fehler auftreten greift Kapitel 3.4.8.2.
2. Für jeden Greifpunkt wird einmal überprüft, ob der Roboter dort mit einem Pfeiler kollidieren würde, was in Kapitel 3.4.8.1 beschrieben wird.
3. Vom Greifpunkt aus wird zum 0,0 Punkt im Koordinatensystem ein kollisionsfreier Pfad durch das Labyrinth berechnet. Falls keiner gefunden werden kann, beendet das Programm. Dieser Weg wird in jeweils 1 cm lange Wegabschnitte aufgeteilt, wie in Kapitel 3.4.4 ersichtlich.

4. Danach greift der Roboter den Schlauch und navigiert ihn Wegpunkt für Wegpunkt durch das Labyrinth, was das Kapitel 3.4.5 näher beschreibt.
5. Erkennt der Kraftmonitor während der Navigation eine Überlast wie in Kapitel 3.4.6 beschrieben, wird das in Kapitel 3.4.7 beschriebene Verfahren ausgelöst. Nach erfolgreichem Befreien des Schlauchs, beginnt der Hauptablauf nochmals.
6. Sobald alle Wegabschnitte angefahren wurden, wird der Schlauch am Ziel abgelegt, der Roboter fährt zurück in die Ausgangsposition und der Ablauf endet regulär.

**Unterbrechungen:** Wie in Abbildung 9 dargestellt, ist der gesamte Greifzyklus von einer unterbrechbaren Region umschlossen. Ein Ctrl+C oder eine unerwartete Exception führt jederzeit direkt zum Teardown, um auch dann ein sauberes Herunterfahren sicherzustellen.

**Debug-Flags:** Für Entwicklung und Diagnose stellt `debug.py` verschiedene Debug-Flags bereit. Alle Flags sind standardmässig deaktiviert und können vor dem Programmstart gesetzt werden:

| | |
|---|---|
| `interactive_confirm` | Pausiert den Ablauf vor jedem Schritt und wartet auf eine Tastatureingabe. Dabei kann mit `q` abgebrochen werden, was zum sauberen Teardown führt. |
| `detector_overlay` | Gibt das Kamerabild mit eingezeichneter Detection-Visualisierung kontinuierlich auf dem Bildschirm aus. |
| `detector_confirm` | Zeigt nach erfolgreicher Erkennung ein Vorschaufenster mit dem Debug-Frame des Detektors und wartet vor dem Weiterfahren auf eine Tastatureingabe. |
| `path_visualization` | Öffnet nach der Pfadplanung ein Bild mit dem berechneten A*-Pfad, den Säulen und zeigt auch die verbotenen Gebiete. |
| `analyzer_panels` | Zeigt während dem Befreien des Schlauchs auf vier beschriebenen Bild-Panels was aktuell erkannt wird. |
| `force_log` | Protokolliert die Kraftmesswerte von Fx und Fy während der Navigation auf der Konsole. |

### 3.4.3 Erkennen des Schlauchs

Die Schlauch-Erkennung folgt einem Strategy-Pattern, damit sich der konkrete Erkennungsalgorithmus leicht austauschen lässt, ohne dafür den Hauptablauf in `app.py` anpassen zu müssen. Dies hat sich schon im Rahmen dieser Arbeit als sehr nützlich erwiesen, da verschiedene Detektionsalgorithem ausprobiert wurden. `vision.py` koppelt Kamera und einen Detektor, der das Interface `HoseDetectorBase` implementiert. Die Abbildung 10 zeigt die beteiligten Klassen.

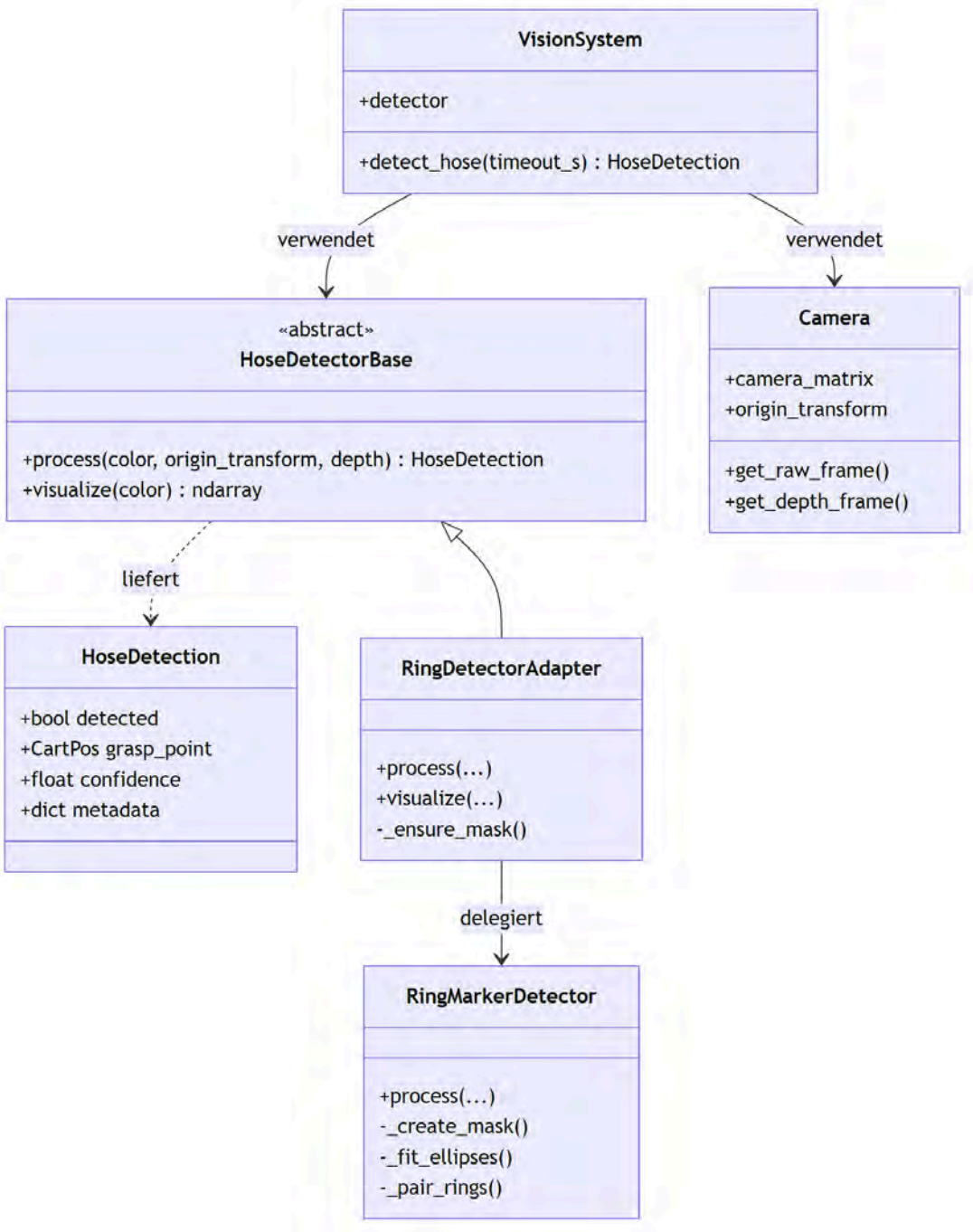



Abbildung 10: Klassendiagramm von Vision und Detection

`VisionSystem` lädt beim Aufbau die Kalibrierungsmatrix `origin_transform.npy` und stellt sie der Kamera bereit. Die Methode `detect_hose()` wartet in einer Schleife auf Frames, ruft pro Frame die `process()`-Methode des Detektors auf und gibt das erste Ergebnis zurück, dessen Confidence einen definierten Schwellenwert überschreitet.

Der konkrete Detektor in dieser Arbeit ist der `RingDetectorAdapter`, welcher aus den in Kapitel 3.2.1 genannten Anforderungen entstanden ist.

Als konzeptionelle Inspiration diente die Arbeit zu CylinderTag [21], welche formal belegt, weshalb planare Marker auf zylindrischen Oberflächen versagen: Da deren Erkennungsalgorithmus auf der Quadratstruktur des Markers beruht, führt bereits eine moderate Krümmung der Oberfläche zu einer Deformation dieser Struktur und damit zu Erkennungsausfällen ab moderaten Blickwinkeln. Als robustere Alternative weist die Arbeit auf kreisförmige Merkmale hin, deren quadratische Form unter perspektivischer Projektion erhalten bleibt, wodurch sich der markierte Bereich zuverlässig segmentieren lässt. Der CylinderTag-Algorithmus selbst codiert die Marker-ID jedoch über das Doppelverhältnis als projektive Invariante und ist für die hier benötigte Aufgabe, welche die blosse Bestimmung von Position und Ausrichtung eines einzelnen Schlauchanfangs ermitteln muss, deutlich überdimensioniert. Zudem ist die zugehörige Bibliothek ausschliesslich in C++ verfügbar und das Einbinden in die bestehende Python-Architektur wurde verworfen.

Jedoch wurde aufbauend auf der genannten Erkenntnis zu kreisförmigen Merkmalen eine eigene, deutlich einfachere Lösung entwickelt, die beide oben genannten Anforderungen erfüllt. Der Ansatz nutzt einen blauen und einen weissen Ring aus Isolierband, die am Schlauchanfang angebracht sind. Mittels HSV-Farbmasken werden die beiden Ringe im Kamerabild segmentiert und über Ellipsen-Fitting mithilfe der Methode `cv2.fitEllipse()` jeweils in ihrem Subpixel-Mittelpunkt lokalisiert. Der Greifpunkt ergibt sich als Mittelpunkt zwischen den beiden Ringzentren, die Orientierung des Schlauchs aus dem Richtungsvektor zwischen zwei Bezugspunkten. Diese liefern damit genau die Richtungsinformation, die einer einzelnen Farbmaske fehlte. Gleichzeitig sind die aufgeklebten Ringe unempfindlich gegenüber dem Einknicken beim Greifen. Da sich ein Ring zudem auch bei schräger Betrachtung weiterhin als Ellipse abbildet und zuverlässig segmentierbar bleibt, ist der Ansatz robust gegenüber der zylindrischen Schlauchgeometrie. In der praktischen Erprobung erwies sich diese Lösung gegenüber den zuvor evaluierten Ansätzen als deutlich zuverlässiger.

### 3.4.4 Pfadplanung

`MazeNavigator` berechnet einen kollisionsfreien Pfad vom Greifpunkt zum Ziel durch das Säulenlabyrinth. Dieses Script besteht aus den folgenden zwei Phasen:

**Initialisierung `init()`:** Von `app.py` aus wird zu Beginn einmalig die Initialisierungsfunktion dieses Scripts aufgerufen, welches folgende zwei Aktionen ausführt:

1. Einerseits werden die Säulenpositionen aus `pillar_positions.json` geladen und über die Arbeitsfläche ein Raster mit einer Zellgrösse von 5 mm gelegt, in dem jede Zelle entweder frei oder ein Hindernis ist. Eine Zellgrösse von zum Beispiel 10 cm wäre zu grob gewesen, da der Hindernisbereich `OBSTACLE_RADIUS_MM = 35 mm` um jede Säule damit nicht präzise abgebildet werden kann und `_center_in_corridors()` Wegpunkte mit einer Genauigkeit von `CORRIDOR_SHIFT_EPS_MM = 0.5 mm` verschiebt, was eine Zellgrösse von 5 mm voraussetzt. Bei einem Säulenabstand von zum Beispiel 10 cm würde eine einzige 10 cm-Zelle den gesamten Korridor abdecken, was eine sinnvolle Pfadberechnung in dieser Implementation ausschliesst. Damit der Roboter nicht in eine Säule fährt, wird um jede Säule ein Hindernisbereich von 35 mm markiert, was der halben Greiferbreite plus etwas Sicherheitsabstand entspricht. Des Weiteren wurden einzelne Korridore statisch als gesperrt hinterlegt, da der Roboter sich dort in einer Singularität befinden würde.

2. Zusätzlich wird aus dem Gitter eine Distance-Map berechnet, die für jede freie Zelle den Abstand zum nächsten Hindernis speichert. Sie ist die Grundlage dafür, dass der A* später Pfade bevorzugt, die möglichst weit von den Säulen entfernt verlaufen.

**Pfadberechnung `plan()`:** Die Methode erhält einen Start- und einen Zielpunkt und durchläuft die in Abbildung 11 gezeigten Schritte:

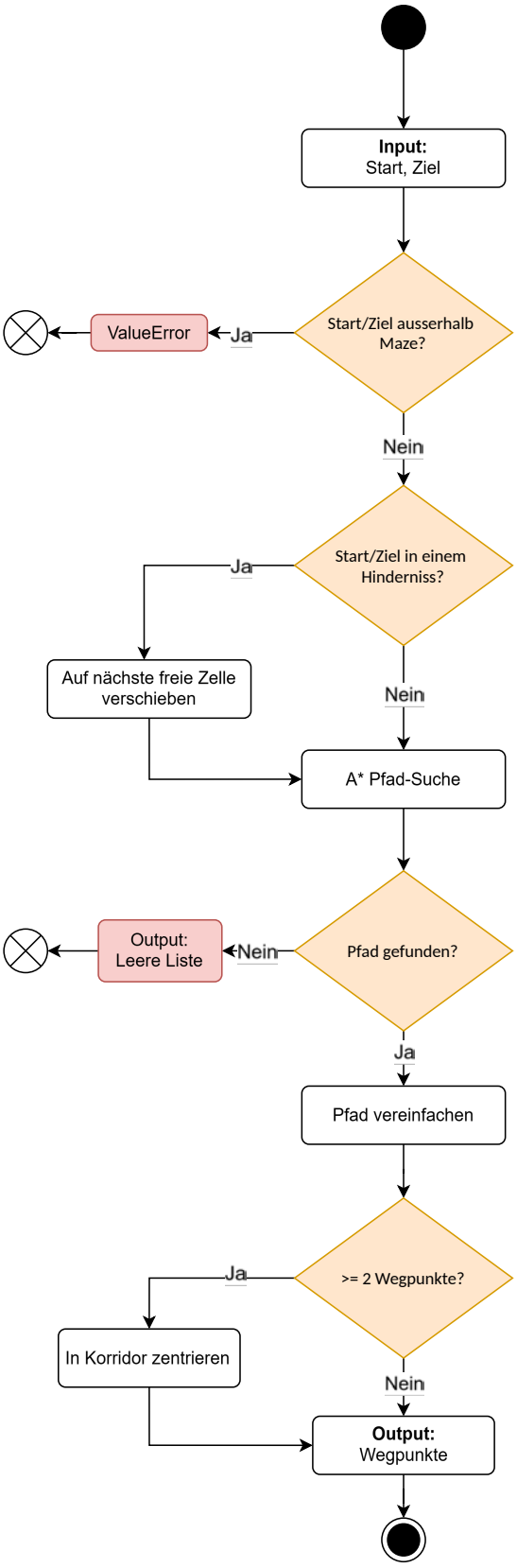



Abbildung 11: Ablauf der Pfadplanung

1. Liegt der Start- oder Zielpunkt ausserhalb des Labyrinths, wird ein `ValueError` geworfen und die Berechnung bricht ab.
2. Liegt einer der Punkte zwar im Labyrinth, aber innerhalb einer Säulenzone, wird er auf die nächste freie Zelle verschoben.
3. Anschliessend sucht der A* den günstigsten Pfad. Findet der Algorithmus keinen Weg, wird eine leere Liste zurückgegeben und der Ablauf endet.
4. Der gefundene Pfad wird mit `_simplify()` von allen Zwischenpunkten befreit, die ohnehin auf einer geraden Linie liegen, sodass nur die Eckpunkte übrig bleiben.
5. Bleiben mindestens zwei Wegpunkte übrig, schiebt `_center_in_corridors()` die geraden Streckenabschnitte zusätzlich exakt in die Mitte zwischen die beiden nächsten Säulenreihen.

**A*-Variation:** Die Suche kombiniert zwei Mechanismen. Erstens betritt der A* die Gefahrenzonen um die Säulen nicht, da diese Zellen im Gitter hart als Hindernis markiert sind. Zweitens sucht der A* in seiner klassischen Form den kürzesten Pfad. Dieser führt in diesem Versuch oft knapp an den Säulenrändern entlang, was riskant ist, weil der Schlauch dort leicht hängen bleibt. Aus diesem Grund wurde die Kostenfunktion um einen Aufschlag erweitert, der umso grösser ausfällt, je näher eine Zelle an einer Säule liegt. Dieser Aufschlag stammt aus der zuvor berechneten Distance-Map. Zellen weit weg von Hindernissen verursachen so kaum Zusatzkosten, Zellen direkt am Säulenrand hingegen sind teuer. Dadurch hält die Variation des A* den Pfad besonders in Kurven von den Säulen weg und nimmt dafür bewusst einen etwas längeren Weg in Kauf.

Das Ergebnis ist eine Liste von Wegpunkten im UF5, die direkt an die Roboteraktionen übergeben wird. Die Abbildung 12 zeigt einen so berechneten Pfad als Debug-Ausgabe, mit den Säulen samt Hindernisbereich und dem geglätteten Weg vom Greifpunkt zum Ziel:

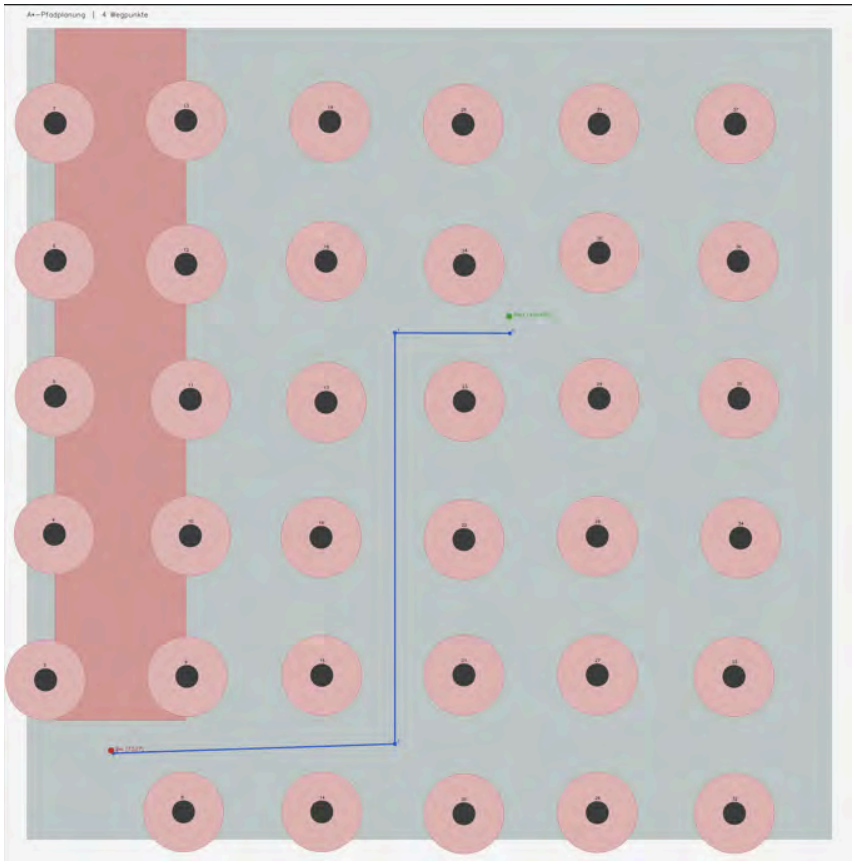


Abbildung 12: Beispielhaft berechneter Pfad (Debug-Ausgabe)

### 3.4.5 Roboteraktionen

Die Klasse `RobotActions` bündelt alle Bewegungen von Roboter und Greifer, die benötigt werden um sicher im Labyrinth navigieren zu können. Beim Initialisieren wird zuerst der Greifer per Modbus verbunden und aktiviert. Anschliessend wird die Verbindung zum Roboter über RMI aufgebaut und der Userframe sowie das Toolframe gesetzt.

Nach aussen stellt die Klasse mehrere einfache Methoden bereit, die jeweils eine konkrete Bewegung oder Aktion kapseln, sodass der restliche Ablauf nicht direkt mit den RMI-Funktionen arbeiten muss:

- `move_above(pos)` und `move_down(pos)` fahren linear über beziehungsweise auf einen vorgegebenen Greifpunkt mit fixer Sicherheits- bzw. Greifhöhe. Mit `move_linear(pos)` kann eine beliebige Pose linear angefahren werden.
- `grip()` schliesst den Greifer mit 80 N Klemmkraft, `release()` öffnet ihn wieder auf 50 mm.
- `lift()` hebt den Greifer auf eine definierte sichere Höhe an. `lift_safe()` versucht im Fehlerfall dasselbe, räumt vorher aber zusätzlich den RMI-Buffer und laufende Bewegungen auf, damit der Roboter wieder in einen definierten Zustand kommt.
- `go_to_safe_height()` fährt den Roboter senkrecht auf die sichere Höhe. Mit `go_to_cam_clear_pos()` wird per Gelenkbewegung eine feste Pose angefahren, aus der die Kamera freie Sicht auf das Labyrinth hat.
- `move_to(step)` führt einen einzelnen Schritt des geplanten Schlauchführungszyklus aus. Reine Greifer-Drehungen werden dabei bewusst langsamer gefahren als Translationen, da diese Bewegungen für den Schlauch kritischer sind.
- `read_position()`, `reset()` und `disconnect()` lesen die aktuelle Pose aus, setzen den Roboter nach einem Fehler zurück beziehungsweise trennen Roboter und Greifer am Ende wieder sauber.

Die Navigation durch das Labyrinth wird vollständig von dieser Klasse abgewickelt. `build_navigation_poses(waypoints, grasp_pose)` erzeugt aus dem im Kapitel 3.4.4 geplanten Wegpunktpfad die komplette Bewegungssequenz, die anschliessend Schritt für Schritt ausgeführt wird. Die Hauptschleife prüft zwischen den Schritten lediglich die in Kapitel 3.4.6 beschriebene Kraftüberwachung und kann so direkt auf erhöhte Kräfte reagieren. Der Aufbau der Bewegungssequenz ist in Abbildung 13 dargestellt.

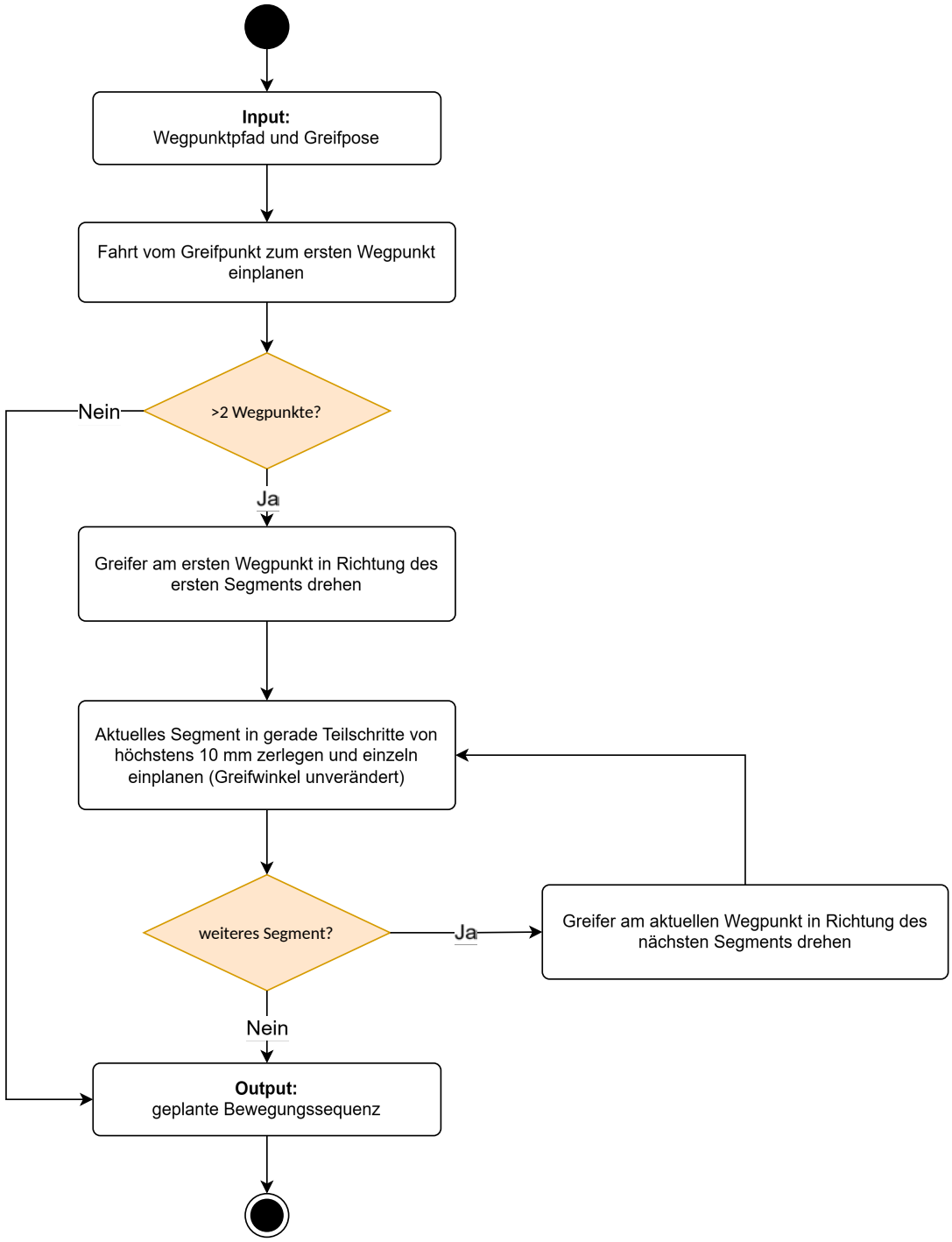



Abbildung 13: Ablauf vom Erstellen der Roboter-Bewegungssequenzen

Die Berechnung erfolgt in drei Phasen:

1. Zuerst wird die Anfahrt vom Greifpunkt zum ersten Wegpunkt geplant. Diese Bewegung erfolgt leicht erhöht auf einer Z-Höhe von 18 mm statt 13 mm, damit der Schlauch beim Verfahren nicht über den Boden schleift. Der Greifwinkel bleibt dabei zunächst unverändert
2. Anschliessend wird der Greifer am ersten Wegpunkt so gedreht, dass er parallel zum ersten Pfadsegment ausgerichtet ist, wodurch der Schlauchkopf in Fahrtrichtung.
3. Danach folgt die eigentliche Navigation entlang des Pfads. Jedes Segment wird dabei in mehrere gerade Teilschritte mit maximal 10 mm Länge zerlegt. An jedem neuen Wegpunkt wird der Greifer zusätzlich wieder auf die Richtung des nächsten Segments ausgerichtet.

### 3.4.6 Kraftüberwachung

Die Kraftüberwachung dient dazu, zu erkennen, ob der Schlauch während dem Navigieren an einer Säule stecken geblieben ist.

Die im Rahmen dieser Arbeit ergänzte Basis-Klasse `modules.ForceSensor` kapselt die Kommunikation mit dem Bota-Sensor und stellt eine vereinfachte API zur Verfügung:

- `connect()` instanziiert den `BotaDriver` anhand der Konfigurationsdatei `bota_config.json`, konfiguriert den Sensor und aktiviert ihn. Die Methode liefert zurück, ob die Initialisierung erfolgreich war. Mit `disconnect()` wird der Sensor wieder deaktiviert und die verwendeten Ressourcen des Treibers werden freigegeben.
- `get_force()` und `get_torque()` liefern den zuletzt verfügbaren Kraft- beziehungsweise Drehmomentvektor als Tripel `(Fx, Fy, Fz)` in N bzw. `(Tx, Ty, Tz)` in Nm zurück. Intern verwenden beide Methoden die non-blocking Methode `read_frame()`. Laut Dokumentation arbeitet der Treiber dabei mit einem Single-Slot-Puffer, der immer den aktuellsten Sensor-Frame enthält [39]. Dadurch ist kein zusätzlicher Lese-Thread oder Locking nötig.
- `get_temperature()` liefert analog die aktuelle Sensortemperatur in °C.
- `get_force_magnitude()` stellt eine vereinfachte Methode bereit, um direkt den Betrag `|F|` des Kraftvektors zu erhalten.
- `calibrate_idle(samples=100)` mittelt mehrere Messwerte im Ruhezustand und berechnet daraus eine Baseline für `Fx` und `Fy`. Im Gegensatz zu den anderen Methoden, wird hier bewusst `read_frame_blocking()` verwendet, damit jeder Messwert garantiert aus einem neuen Sensor-Frame stammt und nicht mehrfach derselbe Pufferinhalt verwendet wird.

Die im Projekt erstellte Unterklasse `experiments.hoses.ForceSensor` ergänzt zusätzlich die Methode `is_overloaded()`. Diese liefert `True`, sobald sich `Fx` oder `Fy` um mehr als `FORCE_THRESHOLD_N` = 5 N von der zuvor bestimmten Idle-Baseline unterscheiden.

### 3.4.7 Verfahren zum Befreien des Schlauchs

Nach dem Anschlagen des Kraftsensors, wenn der Schlauch stecken geblieben ist, wird die Bewegung abgebrochen und das Paket `stuck/` aufgerufen. Dieses bestimmt, an welcher Säule der Schlauch feststeckt und wo er gegriffen werden muss, und führt anschliessend eine korrigierende Greif- und Zugbewegung aus.

Die Abbildung 14 zeigt die vorhanden Komponenten. Den Einstieg bildet `Recovery`, aufgerufen von `HoseMazeApp`, welches den gesamten Ablauf koordiniert. Für die Bildauswertung nutzt es den `Analyzer`, der pro Kamerabild ein `AnalysisResult` mit der erkannten Problemstelle und einem vorgeschlagenen Greifpunkt liefert. `SafeGrasp` liegt zwar im selben Ordner und taucht deshalb im Klassendiagramm auf, ist inhaltlich aber unabhängig vom Befreien des Schlauchs und wird darum im Kapitel 3.4.8 beschrieben.

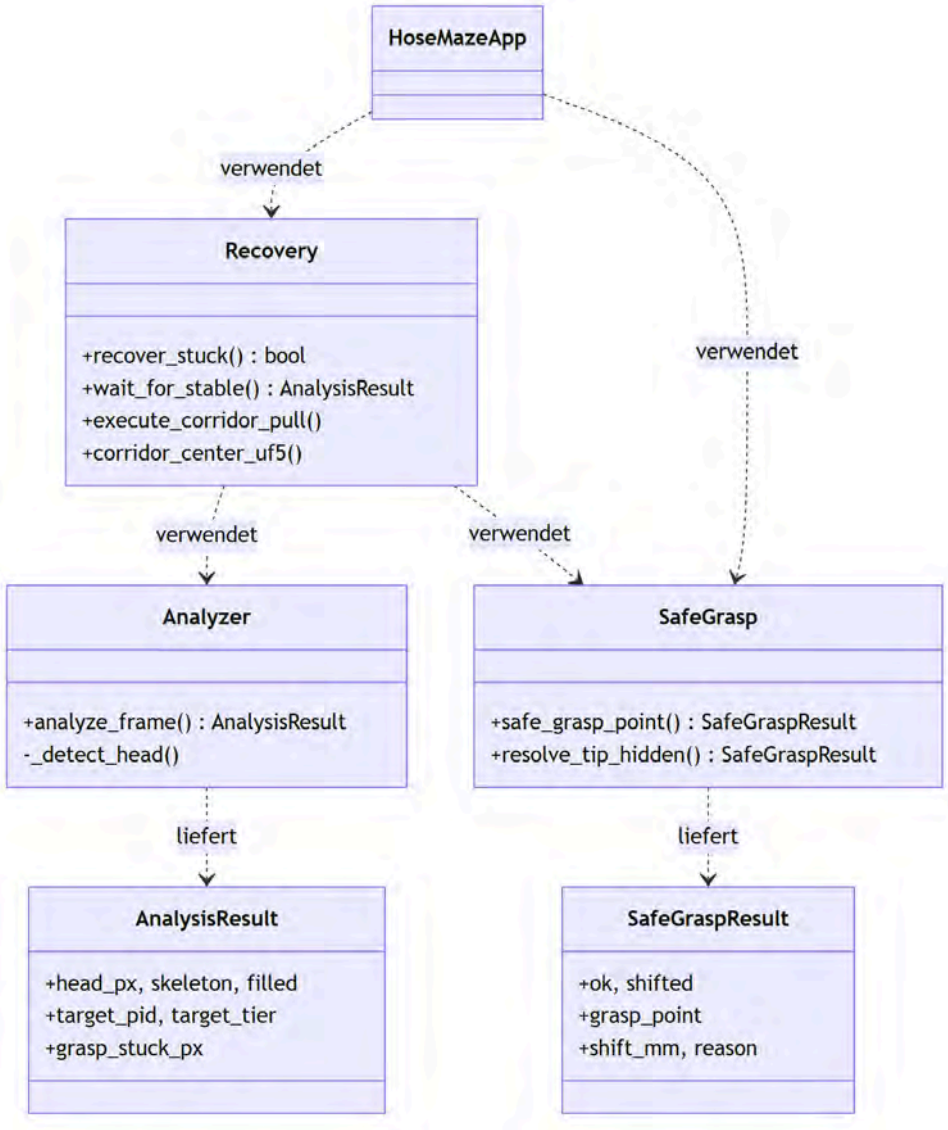



Abbildung 14: Komponenten der Schlauch-Befreiung und ihr Zusammenspiel

Der Befreiungsablauf besteht aus zwei Teilen, dem Erkennen und dem Befreien. Zunächst werden aus dem Kamerabild Problemstelle und Greifpunkt bestimmt. Anschliessend wird der Greifpunkt angefahren und der Schlauch gelöst.

### 3.4.7.1 Lokalisieren der Problemstelle

Bevor der Schlauch befreit werden kann, muss zuerst herausgefunden werden, an welcher Säule er feststeckt. Diese Erkennung wurde grundsätzlich basierend auf der HSV-Maske des Schlauchs in Kombination mit den bekannten Positionen der Säulen umgesetzt. Ziel ist es, die Problemstelle am Schlauch zu bestimmen und daraus einen geeigneten Greifpunkt für die Befreiung abzuleiten. Um die Erkennung robuster gegenüber Bildrauschen zu machen, werden die jeweils letzten zwei Schlauchmasken in einem Buffer gespeichert und gemittelt, bevor sie weiterverarbeitet werden. Die Abbildung 15 zeigt den Ablauf des Erkennungsprozesses.

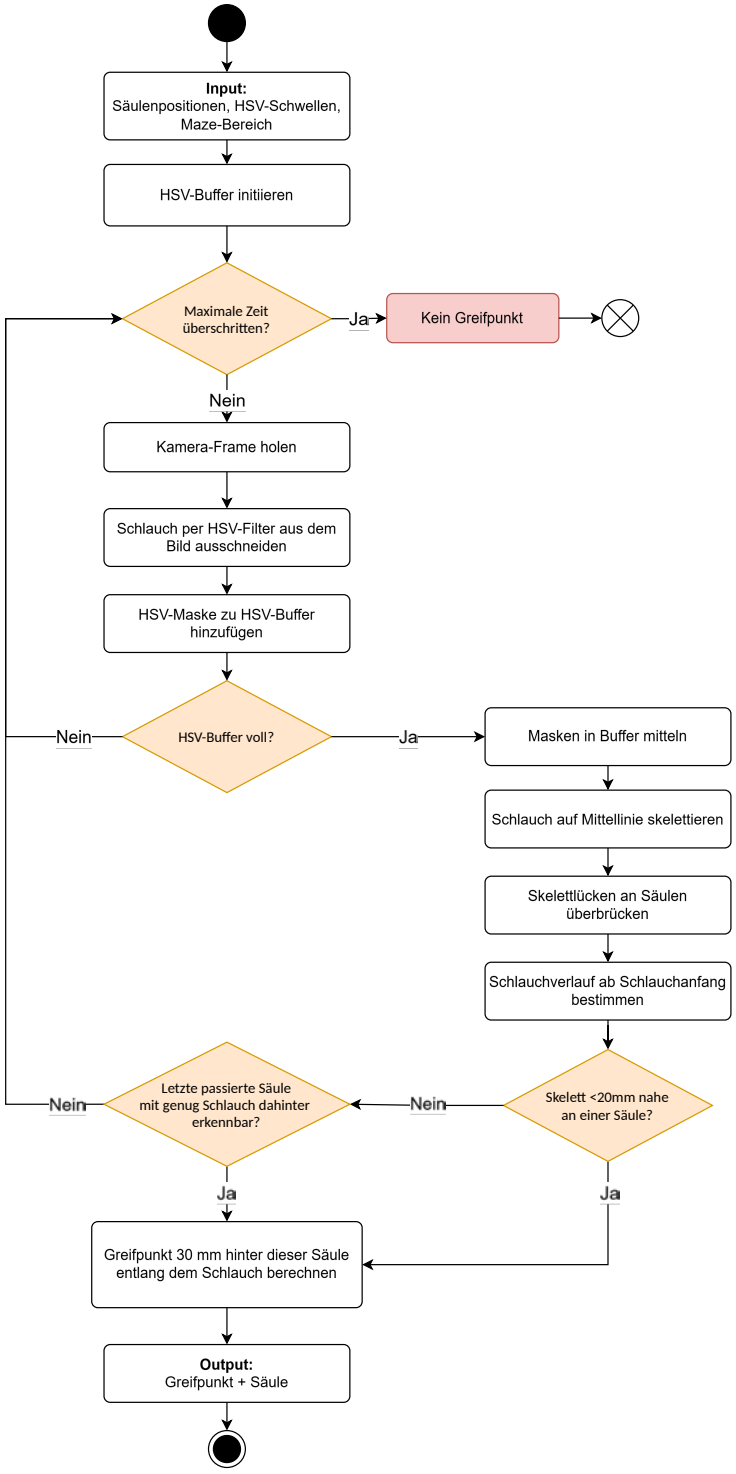



Abbildung 15: Gesamtablauf der Lokalisierung der Problemstellen

Pro Frame wird der gesamte Schlauchkörper im Kamerabild über eine HSV-Farbmaske segmentiert. Eine einzelne Maske ist dabei rein binär, jedes Pixel ist also entweder Schlauch oder Hintergrund. Sobald zwei solche Masken im Buffer liegen, werden sie gemittelt und es wird mit dieser gemittelten Maske weitergearbeitet. Aus den 0/1-Werten entstehen so Grauwerte zwischen 0 und 1: Ein Pixel, das in beiden Frames als Schlauch erkannt wurde, erhält den Wert 1, eines das nur in einem der Frames erkannt wurde den Wert 0,5. Der Grauwert gibt also an, wie zuverlässig eine Stelle über die Frames hinweg bedeckt war, und glättet zugleich das Bildrauschen der Einzelmasken. Durch Skelettierung wird die gemittelte Maske auf eine ein Pixel breite Mittellinie reduziert. Da die Säulen den Schlauch teilweise verdecken, weist sie an diesen Stellen Unterbrechungen auf, die überbrückt werden, sodass eine durchgehende Linie entsteht. Vom erkannten Schlauchanfang aus wird das Skelett anschliessend mit einer BFS durchlaufen und jedem Pixel sein Abstand zum Anfang zugeordnet. Dadurch ist an jeder Stelle des Schlauches der Richtungsvektor zurück zum Anfang eindeutig festgelegt, und für jeden Punkt lässt sich seine Lage relativ zum Anfang sowie die Orientierung des Schlauchverlaufs im Versuchsaufbau bestimmen.

Anhand des Verlaufs und der bekannten Säulenpositionen wird in zwei Schritten nach einer Problemstelle gesucht. Zuerst wird geprüft, ob das Schlauchskelett näher als 20 mm an einer Säule vorbeiläuft. Trifft das zu, gilt diese Säule als Problemstelle. Typisch ist das, wenn der Schlauch seitlich an einer Säule anliegt.

Findet sich keine solche Säule, wird auf einen Fallback zurückgegriffen. Dessen Idee ist, ganz hinten am Schlauch zu greifen und von dort mehr Material ins Labyrinth nachzugeben, sodass sich das eigentliche Feststecken weiter vorne oft von selbst entspannt. Dazu sucht der Fallback die Säule, die dem Schlauch entlang am weitesten vom Kopf entfernt liegt und hinter der noch ein möglicher Greifpunkt vorhanden ist. In der Praxis ist dies meistens die zweit-hinterste Säule, da die hinterste zum Rand des Labyrinths gehört und potenzielle Greifpunkte sonst ausserhalb des definierten Bereichs lägen. Nur wenn sich der komplette Schlauch im Labyrinth befindet, wird tatsächlich die hinterste Säule gewählt. Liefert auch der Fallback keine Säule, beginnt der Ablauf im nächsten Frame von vorne.

In beiden Fällen liegt der Greifpunkt zunächst 30 mm hinter der Problemstelle, gemessen entlang des Schlauches Richtung Schlauchende. Dieser Greifpunkt wird anschliessend durch den in Kapitel 3.4.8.1 beschriebenen Algorithmus überprüft und wenn nötig verschoben.

#### 3.4.7.2 Befreien des Schlauchs

Sobald ein stabiler Greifpunkt feststeht, muss immer noch das Feststecken gelöst werden. Grundsätzlich wird etwas hinter dem Ort wo der Schlauch feststeckt angesetzt und in die Richtung gezogen, in die der Schlauch verläuft, damit er nachgibt und sich von der Säule löst. Dafür werden Greif- und Zugbewegung benötigt, welche wie folgt berechnet werden:

- Die **Greifer-Rotation** ergibt sich aus der Tangente des Schlauchskeletts am Greifpunkt, sodass der Greifer quer zum Schlauchverlauf zugreift.
- Die **Zugrichtung** folgt dem Schlauchverlauf in Richtung Schlauchanfang und wird auf die nächste der vier Achsenrichtungen gerundet, damit der Zug parallel zu einem Korridor des Labyrinths verläuft.
- Die **Korridor-Mitte** wird aus dem Mittelpunkt der nächsten Säulen links und rechts vom Greifpunkt berechnet.

Daraus ergibt sich der Weg, der vom Greifpunkt zur Korridor-Mitte und dann 100 mm in Zugrichtung führt. In der Abbildung 16 ist ein Screenshot vom Debug-Output eines solchen berechneten Wegs zu sehen.

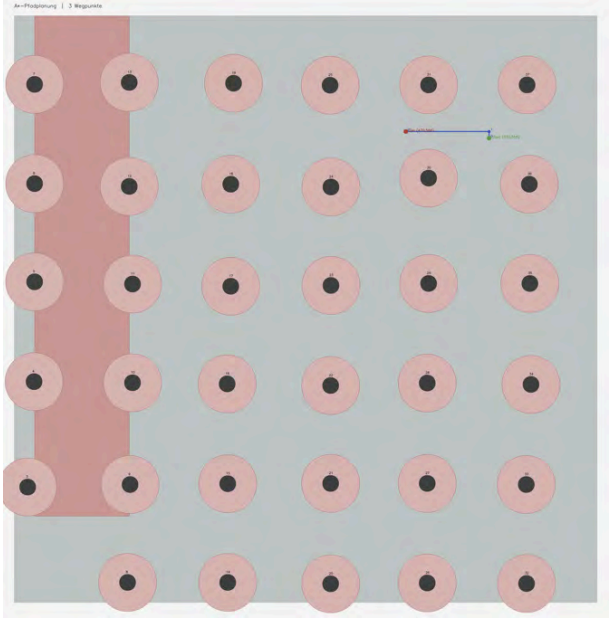


Abbildung 16: Debug-Output aus Top-Down-Ansicht mit Greifpunkt und Zugziel

Bevor der Roboter losfährt, durchläuft der erkannte Greifpunkt noch einmal die Safe-Grasp-Prüfung, die ihn bei Bedarf geringfügig verschiebt. Anschliessend fährt der Roboter über den Greifpunkt, senkt ab, greift dort den Schlauch, und führt dann das Befreiungsmanöver aus. Nach dem Loslassen und Anheben kehrt der Ablauf in den Hauptloop zurück, wo die Schlauch-Erkennung von Neuem beginnt.

### 3.4.8 Safe-Grasp

Die Klasse `Safe-Grasp` stellt Methoden zur Validierung von Greifpunkten und zum Schätzen des Schlauchanfangs bereit, wenn dieser nicht direkt im Kamerabild erkennbar ist.

#### 3.4.8.1 Greifpunkt absichern

Ist der blaue Ring sichtbar und damit der Greifpunkt bekannt, kommt die Methode `safe_grasp_point()` zum Einsatz, deren Ablauf die Abbildung 17 zeigt.

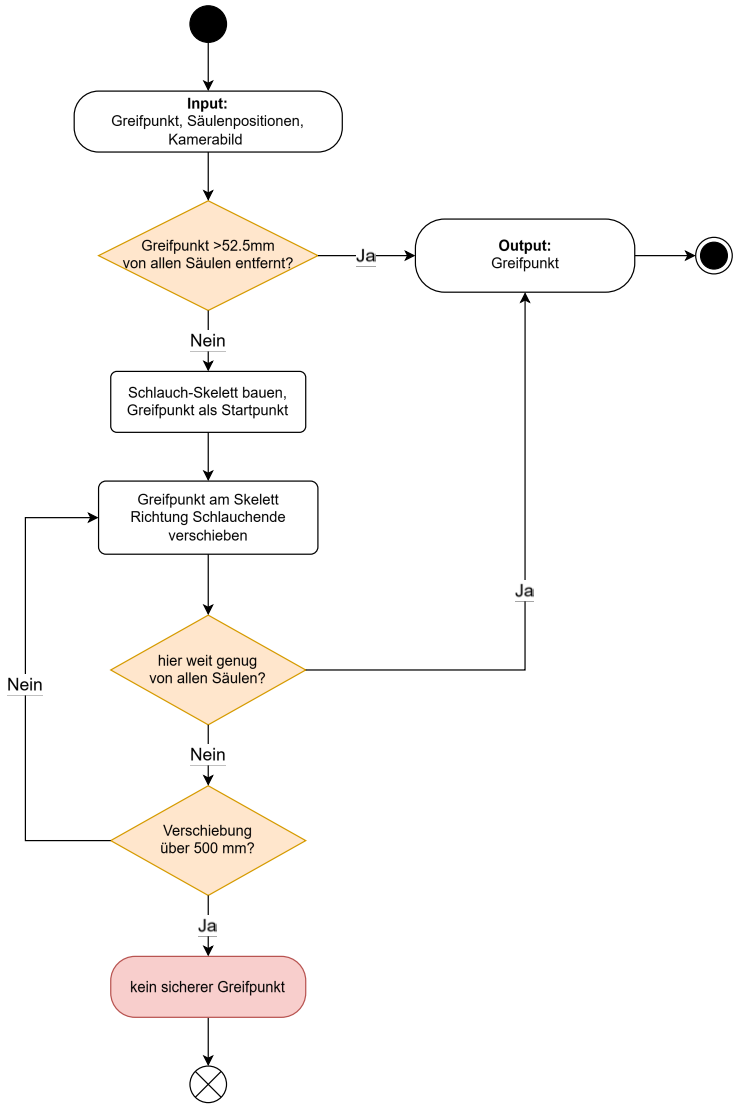



Abbildung 17: Ablauf des Finden eines sicheren Greifpunkts

1. Zuerst wird geprüft, ob der Greifpunkt mindestens 52,5 mm von jeder Säule entfernt liegt, damit der Greifer dort kollisionsfrei aufsetzen kann. Dieser Mindestabstand setzt sich aus dem halben Greifer-Durchmesser (37,5 mm), dem halben Säulen-Durchmesser (10 mm) und einer Sicherheitsmarge von 5 mm zusammen.
2. Ist der Abstand gross genug, wird der Punkt direkt zurückgegeben.
3. Andernfalls wird er schrittweise am Schlauchskelett Richtung Schlauchende verschoben, bis eine freie Stelle gefunden wird oder die maximale Verschiebung von 500 mm überschritten ist.
4. An der gefundenen Stelle wird zusätzlich die Greiferdrehung aus der Tangente des Schlauchskeletts bestimmt, damit der Greifer quer zum Schlauch zugreift.

#### 3.4.8.2 Verdeckter Schlauchanfang

`resolve_tip_hidden()` kommt zum Einsatz, wenn der blaue Ring nicht erkannt werden kann, weil er beispielsweise hinter einer Säule oder ausserhalb des Bildausschnitts liegt.

In diesem Fall wird der Schlauchanfang aus dem Skelett des Schlauches abgeleitet, die Abbildung 18 zeigt diesen Ablauf.

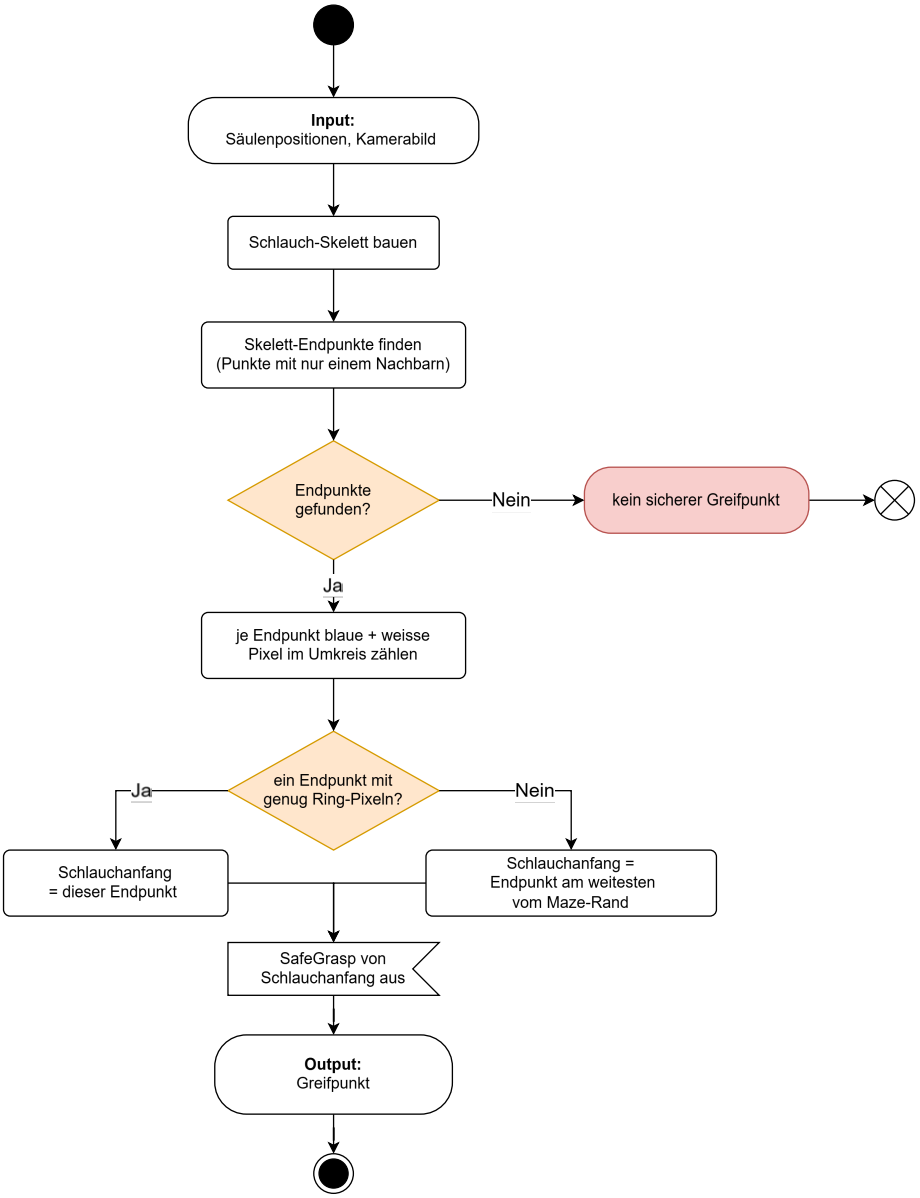



Abbildung 18: Ablauf der Schlauchanfang-Ermittlung bei verdecktem Ring

1. Zuerst wird das Skelett des Schlauches aufgebaut und alle Skelett-Endpunkte ermittelt, also Pixel mit nur einem Nachbarn.
2. Für jeden Endpunkt werden im Bild in einem Umkreis von 40 Pixeln blaue und weisse Pixel gezählt. Der Endpunkt mit den meisten solcher Ring-Pixel gilt als Schlauchanfang, da dort der teilweise verdeckte blaue oder weisse Ring vermutet wird.
3. Liefert kein Endpunkt genug Ring-Pixel, dient der am weitesten vom Rand des Labyrinths entfernte Endpunkt als Fallback, da der Schlauchanfang im Innern des Labyrinths erwartet wird.
4. Von diesem Punkt aus wird anschliessend die gleiche Verschiebe-Logik wie in Kapitel 3.4.8.1 durchlaufen, bis ein sicherer Greifpunkt gefunden ist.

# 4 Experimentelle Resultate

Die zuvor beschriebenen Subsysteme werden nun quantitativ überprüft. Untersucht werden Schlauch-erkennung, Kraftschwellenwert, Schrittlänge und die Erkennung des feststeckenden Schlauchs in je einem eigenen Experiment unter kontrollierten Bedingungen. Abschliessend bewertet ein Systemtest das Zusammenspiel aller Komponenten unter realen Bedingungen.

## 4.1 Schlaucherkennung

In diesem Experiment wurden drei Erkennungsverfahren für den Schlauchanfang hinsichtlich Erkennungsrate, Verarbeitungszeit und Positionsgenauigkeit unter drei verschiedenen Lichtbedingungen verglichen. Getestet wurden der **Ring-Detektor** aus dem Kapitel 3.4.3, ein **ArUco-Marker**-basierter Detektor sowie ein einfacher **HSV-Blob-Detektor** der lediglich die blaue HSV-Maske ohne Formanalyse erkennt. Die drei Lichtszenarien waren Deckenlicht (kontrolliert, diffus), Versuchslampenlicht (gerichtet, intensiver) und Tageslicht (unkontrolliert, Sonneneinstrahlung). Pro Methode und Szenario wurden 20 Frames ausgewertet.

### 4.1.1 Resultate

Als Referenzwert für die Positionsgenauigkeit diente die per Teach-Pendant manuell angefahrene Greifposition des Schlauchanfangs ($X = 312.5$ mm, $Y = 430.5$ mm, $R = -2.68°$). Die gemessenen Mittelwerte der drei Detektoren werden relativ zu diesem Wert bewertet.

| **Methode** | **Lichtszenario** | **Rate %** | **Zeit (ms)** | $\overline{X}$ **(mm)** | $\sigma_X$ | $\overline{Y}$ **(mm)** | $\sigma_Y$ | $\overline{R}$ **(°)** | $\sigma_R$ | **conf** |
|---|---|---|---|---|---|---|---|---|---|---|
| Ring-Detektor | Deckenlicht | 100.0 | 107.6 | 312.61 | 0.06 | 430.71 | 0.70 | −2.70 | 0.18 | 1.00 |
| Ring-Detektor | Versuchslampe | 100.0 | 220.2 | 314.47 | 0.08 | 431.57 | 0.61 | −2.62 | 0.19 | 1.00 |
| Ring-Detektor | Tageslicht | **5.0** | 179.9 | 313.75 | 0.00 | 431.31 | 0.00 | −0.38 | 0.00 | 1.00 |
| ArUco ID 3 | Deckenlicht | 40.0 | 4603.9 | 311.56 | 0.00 | 413.75 | 0.14 | −4.44 | 1.09 | 1.00 |
| ArUco ID 3 | Versuchslampe | 90.0 | 12279.8 | 311.57 | 0.01 | 413.92 | 0.22 | −77.06 | 105.7 | 1.00 |
| ArUco ID 3 | Tageslicht | 90.0 | 4406.3 | 312.98 | 0.01 | 413.47 | 0.17 | −1.96 | 83.25 | 1.00 |
| HSV-Blob (Blau) | Deckenlicht | 100.0 | 157.4 | 313.39 | 0.10 | 413.65 | 0.39 | 0.00 | 0.00 | 0.195 |
| HSV-Blob (Blau) | Versuchslampe | 100.0 | 125.3 | 313.50 | 0.10 | 414.33 | 0.28 | 0.00 | 0.00 | 0.172 |
| HSV-Blob (Blau) | Tageslicht | 100.0 | 145.5 | 311.84 | 0.08 | 415.17 | 0.34 | 0.00 | 0.00 | 0.186 |

Tabelle 1: Detektions-Benchmarking der drei Erkennungsverfahren unter drei Lichtszenarien

Die in der Tabelle 1 zusammengefassten Ergebnisse zeigen drei klar unterschiedliche Profile.

Der **Ring-Detektor** erzielt unter Decken- und Versuchslampenlicht eine Erkennungsrate von 100 % bei gleichzeitig sehr geringer Positionsstreuung ($\sigma_X < 0.1$ mm, $\sigma_Y < 0.7$ mm) und einer stabilen Rotationsschätzung ($\sigma_R < 0.2°$). Die Verarbeitungszeit liegt zwischen 108 ms und 220 ms. Unter Tageslicht bricht die Rate auf 5 % ein. Die Ursache liegt nicht in der HSV-Segmentierung als solcher, der HSV-Blob-Detektor ist unter denselben Bedingungen zu 100 % stabil, sondern spezifisch im weissen Referenzring. Dessen HSV-Maske ist definiert als hohe Helligkeit ($V \geq 200$) bei niedriger Sättigung ($S \leq 78$), unabhängig vom Farbton. Nur mit Tageslicht war nicht mehr die Helligkeit des weissen HSV-Wertes gewährleistet. Der blaue Ring bleibt durch seinen spezifischen Hue-Bereich (H: 81-142) auch bei Tageslicht selektiv. Ein Ersetzen des weissen durch einen zweiten farbigen Ring (z.B. rot oder grün) würde die Tageslichtempfindlichkeit beseitigen, da Hue-basierte Masken gegenüber Helligkeitsschwankungen robuster sind wie in Kapitel 2.2.5.1 beschrieben.

Der **HSV-Blob-Detektor** erreicht in allen drei Lichtszenarien 100 % Erkennungsrate und ist damit das robusteste der drei Verfahren gegenüber Lichtveränderungen. Das liegt daran, dass hier spezifisch die blaue Farbe getestet wurde und nicht der weisse Marker am Schlauch. Die Positionsgenauigkeit ist daher vergleichbar mit dem Ring-Detektor ($\sigma_X \leq 0.1$ mm, $\sigma_Y \leq 0.4$ mm). Der wesentliche Nachteil ist das vollständige Fehlen einer Rotationsschätzung ($\overline{R} = 0°$, $\sigma_R = 0°$ in allen Szenarien), da ein einfacher Flächenschwerpunkt keine Orientierungsinformation liefert. Die Konfidenzwerte von 0,17-0,20 spiegeln die niedrige geometrische Selektivität wider.

Der **ArUco-Detektor** ist für den Echtzeitbetrieb ungeeignet. Die mittlere Verarbeitungszeit beträgt 4,4-12,3 Sekunden pro Frame, was um den Faktor 20-60 über den anderen Verfahren liegt. Die Ursache dafür ist die erschöpfende Suche nach dem Marker in jedem Frame ohne Tracking. Zusätzlich ist die Rotationsschätzung unter Versuchslampen- und Tageslicht instabil ($\sigma_R = 83°$-$106°$), vermutlich durch Reflexionen auf dem Marker, die die Eckpunktdetektion stören. Die Erkennungsrate von nur 40 % unter Deckenlicht zeigt zudem eine unerwartete Empfindlichkeit gegenüber diesem Szenario.

**Schlussfolgerung:** Für den Versuchsaufbau dieser Arbeit wurde der **Ring-Detektor** gewählt. Da die Versuche in einem Labor mit kontrollierter Beleuchtung durchgeführt werden, tritt das Tageslichtproblem in der Praxis nicht auf. Gegenüber dem HSV-Blob-Detektor bietet der Ring-Detektor den entscheidenden Vorteil einer zuverlässigen Rotationsschätzung, die für die korrekte Orientierung des Endeffektors beim Greifen des Schlauchanfangs erforderlich ist.

## 4.2 Kraft-Sensor Schwellenwert

Um den optimalen Schwellenwert für den Kraft-Sensor herauszufinden wurde der in der Abbildung 19 gezeigte Versuch aufgebaut:

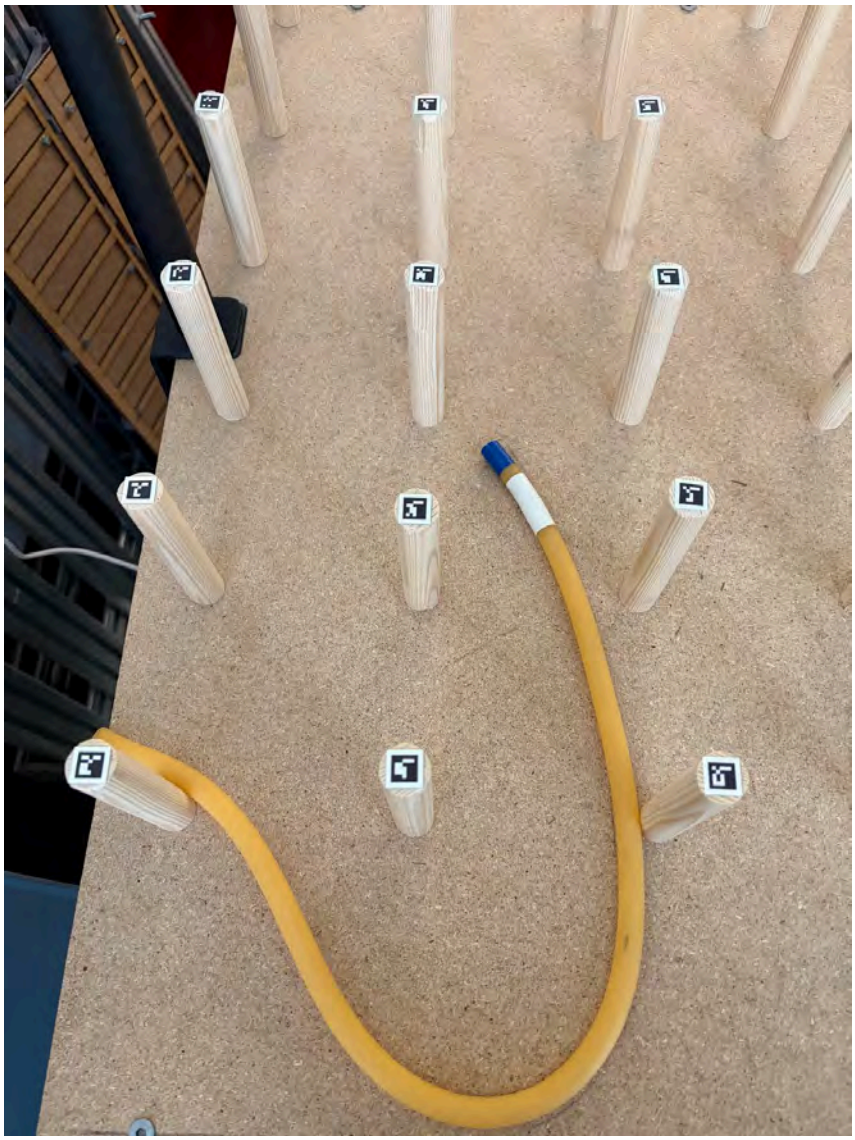


Abbildung 19: Schlauch links neben dem Tisch befestigt und in Position für Experiment

In diesem Experiment wurde der Schlauch an einem Ort befestigt, um einen Pfeiler geschlauft und an einem definierten Punkt abgelegt. Mit einer sehr tiefen Schrittlänge von 5 mm musste der Roboter den Schlauch greifen und so lange durch das Labyrinth navigieren, bis der Kraft-Sensor ausschlug. Anschliessend hielt der Roboter still und die Länge des gespannten Schlauchs wurde gemessen.

Als Referenzwert diente eine zuvor manuell ermittelte Schlauchlänge. In einem Vorversuch wurde der Schlauch von Hand durch denselben Aufbau geführt und in dem Moment gestoppt, in dem als Mensch deutlich spürbar war, dass ein weiteres Ziehen zum Hängenbleiben führen würde. In dieser Position war der Schlauch **45 cm** gestreckt, was dementsprechend als Referenzwert diente.

### 4.2.1 Resultate

Es wurden zehn Schwellenwerte von 1 N bis 10 N getestet, jeder Schwellenwert wurde dreimal wiederholt. Bei den drei tiefsten Schwellenwerten löste der Sensor aus, bevor der Schlauch überhaupt nennenswert gespannt werden konnte, sodass keine sinnvolle Schlauchlänge gemessen werden konnte. Bei Schwellenwert 1 N reagierte der Sensor unmittelbar nach dem Greifen, bei den Schwellenwerten 2 N und 3 N nach wenigen Schritten und vor jeglicher Schlauchspannung. Ab Schwellenwert 4 N konnte der Roboter den Schlauch tatsächlich um das Hindernis führen, bis der Sensor an einer realen Spannung auslöste.

| Schwellenwert [N] | Versuch 1 | Versuch 2 | Versuch 3 | Mittelwert [cm] |
|---|---|---|---|---|
| 1 | direkt | direkt | direkt | – |
| 2 | 2 Schritte | 7 Schritte | 8 Schritte | – |
| 3 | 3 Schritte | 5 Schritte | 7 Schritte | – |
| 4 | 44.2 cm | 43.7 cm | 43.5 cm | 43.80 |
| 5 | 45.6 cm | 45.3 cm | 45.2 cm | **45.37** |
| 6 | 46.5 cm | 46.2 cm | 46.7 cm | 46.47 |
| 7 | 47.4 cm | 47.1 cm | 47.5 cm | 47.33 |
| 8 | 48.0 cm | 48.2 cm | 47.9 cm | 48.03 |
| 9 | 50.5 cm | 49.5 cm | 50.0 cm | 50.00 |
| 10 | 51.5 cm | 50.9 cm | 51.2 cm | 51.20 |

Tabelle 2: Schlauchlänge beim Auslösen des Kraft-Sensors pro Schwellenwert

Die Messreihe in der Tabelle 2 zeigt einen klaren monotonen Zusammenhang zwischen Schwellenwert und Schlauchlänge. Mit jedem zusätzlichen Newton dehnt sich der Schlauch um durchschnittlich etwa 1,2 cm weiter, bevor der Sensor anschlägt. Die Streuung innerhalb einer Wiederholungsserie bleibt mit maximal 0,5 cm sehr gering, was auf ein gut reproduzierbares Auslöseverhalten hindeutet.

**Schlussfolgerung:** Der zuvor definierte Optimalwert von 45 cm wird am besten durch den Schwellenwert von **5 N** abgebildet, dessen Mittelwert mit 45,37 cm nur 0,37 cm über dem Zielwert liegt. Schwellenwert 4 N würde den Schlauch mit 43,8 cm im Mittel zu früh als feststeckend melden, Schwellenwert 6 N hingegen erlaubt mit 46,5 cm bereits ein leichtes Überdehnen über den Zielwert hinaus. Für die nachfolgenden Experimente wird der Kraft-Sensor entsprechend auf 5 N gesetzt.

## 4.3 Optimale Schrittlänge

Das Ziel dieses Experimentes war es, den Zusammenhang zwischen der Schrittlänge, welche der Roboter absolviert, und dem Zurückspicken des Schlauches herauszufinden. Nach dem Anschlagen des Force-Sensors fährt der Roboter den aktuellen Wegabschnitt fertig, bevor er den Schlauch loslässt und mit der Befreiung beginnt. Dementsprechend wurde die Hypothese aufgestellt, dass ein längerer Wegabschnitt zu mehr Zurückspicken führt, weil der Schlauch noch lange nach der Erkennung weiter gespannt wird.

Im Gegensatz zum vorherigen Experiment wurde der Schlauch nicht befestigt, sondern lediglich so im Labyrinth platziert, dass er beim Durchfahren zuverlässig und reproduzierbar an derselben Stelle feststeckt. So lässt sich für jede Schrittlänge unter realitätsnahen Bedingungen messen, wie stark der Schlauch nach dem Auslösen des Kraft-Sensors zurückspickt.

Der Versuch wurde in zwei Konfigurationen durchgeführt, die sich in der Lage des Schlauchs relativ zum Roboterpfad unterscheiden. Die Abbildung 20 und die Abbildung 21 zeigen die jeweiligen Ausgangspositionen, von denen aus der Roboter den Schlauch in Konfiguration 1 gerade entlang seines Pfads und in Konfiguration 2 diagonal durch das Labyrinth spannt.

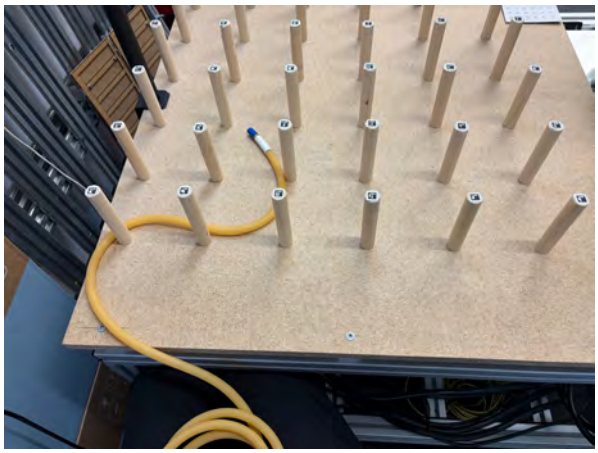


Abbildung 20: Konfiguration 1: Ausgangsposition für die gerade Schlauchführung

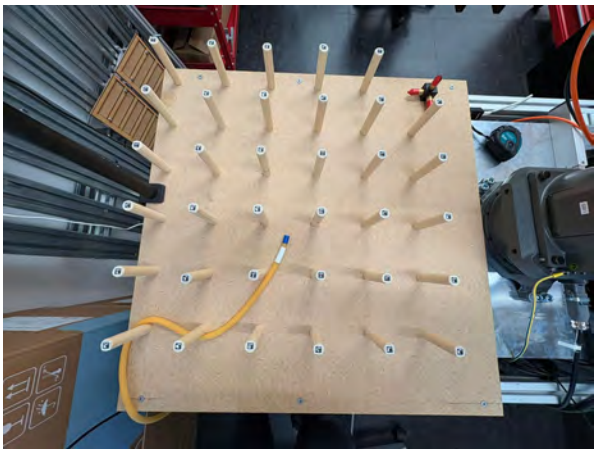


Abbildung 21: Konfiguration 2: Ausgangsposition für die diagonale Schlauchführung

In beiden Konfigurationen wurden sechs Schrittlängen zwischen 5 mm und 50 mm getestet, jede mit fünf Wiederholungen. Erfasst wurden für jeden Versuch die Anzahl absolvierter Schritte bis zum Auslösen des Kraft-Sensors (A.S.) sowie die anschliessend gemessene Länge des zurückgespickten Schlauchstücks in Zentimetern.

### 4.3.1 Resultate

Die Messreihen beider Konfigurationen sind in der Tabelle 3 und der Tabelle 4 festgehalten.

| Schrittlänge [mm] | Versuch 1 | | Versuch 2 | | Versuch 3 | | Versuch 4 | | Versuch 5 | | Mittelwert | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A.S. | cm | A.S. | cm | A.S. | cm | A.S. | cm | A.S. | cm | A.S. | cm |
| 5 | 14 | 5.5 | 15 | 5.6 | 14 | 5.0 | 14 | 4.6 | 16 | 5.6 | 14.6 | 5.26 |
| 10 | 8 | 5.1 | 9 | 7.6 | 8 | 5.9 | 9 | 7.4 | 8 | 6.3 | 8.4 | 6.46 |
| 20 | 5 | 10.5 | 4 | 5.7 | 5 | 10.8 | 4 | 6.2 | 4 | 5.9 | 4.4 | 7.82 |
| 30 | 3 | 8.6 | 3 | 9.9 | 4 | 15.7 | 3 | 8.3 | 3 | 10.6 | 3.2 | 10.62 |
| 40 | 2 | 5.4 | 2 | 5.5 | 2 | 5.6 | 2 | 5.7 | 2 | 5.6 | 2.0 | 5.56 |
| 50 | 3 | 23.6 | 3 | 25.0 | 3 | 13.4 | 3 | 53.0 | 3 | 13.8 | 3.0 | 25.76 |

Tabelle 3: Ergebnisse mit Konfiguration 1: Schlauch gerade entlang des Roboterpfads

| Schrittlänge [mm] | Versuch 1 | | Versuch 2 | | Versuch 3 | | Versuch 4 | | Versuch 5 | | Mittelwert | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A.S. | cm | A.S. | cm | A.S. | cm | A.S. | cm | A.S. | cm | A.S. | cm |
| 5 | 14 | 9.1 | 14 | 8.6 | 15 | 9.4 | 14 | 8.8 | 15 | 9.5 | 14.4 | 9.08 |
| 10 | 8 | 6.7 | 9 | 6.5 | 8 | 9.8 | 8 | 9.4 | 8 | 9.9 | 8.2 | 8.46 |
| 20 | 4 | 10.1 | 5 | 9.0 | 4 | 9.5 | 4 | 10.3 | 4 | 9.8 | 4.2 | 9.74 |
| 30 | 3 | 10.4 | 3 | 11.6 | 3 | 10.4 | 3 | 11.0 | 3 | 10.2 | 3.0 | 10.72 |
| 40 | 3 | 14.7 | 2 | 8.5 | 3 | 21.3 | 2 | 13.4 | 2 | 11.5 | 2.4 | 13.88 |
| 50 | 2 | 12.3 | 2 | 11.6 | 2 | 12.8 | 3 | 18.9 | 2 | 13.1 | 2.2 | 13.74 |

Tabelle 4: Ergebnisse mit Konfiguration 2: Schlauch diagonal durch das Labyrinth gespannt

In beiden Konfigurationen bestätigte sich die Hypothese, dass eine grössere Schrittlänge tendenziell zu einem stärkeren Zurückspicken des Schlauchs führt. Gleichzeitig sinkt die Anzahl absolvierter Schritte (A.S.), weil der Roboter pro Iteration einen längeren Abschnitt fährt, bevor der Kraft-Sensor erneut geprüft wird.

Besonders auffällig war in Konfiguration 1, dass das Zurückspicken meist dann relativ klein blieb, wenn die gefahrene Gesamtstrecke bis zur Erkennung ungefähr bei 80 mm lag. Das sieht man gut bei 10 mm Schrittlänge: Die Durchläufe mit 8 Schritten (80 mm) lagen mit rund 5-6 cm deutlich tiefer als jene mit 9 Schritten (90 mm), welche bereits über 7 cm erreichten. Ein ähnliches Verhalten zeigte sich auch bei 20 mm und 40 mm. Die 5-mm-Reihe passt allerdings nicht ganz in dieses Muster und lässt sich aktuell nicht eindeutig erklären.

Bei 50 mm wurde das Verhalten deutlich unruhiger. In Konfiguration 1 schwankten die Werte stark zwischen 13,4 cm und 53,0 cm. Je grösser die Schrittlänge ist, desto stärker hängt das Resultat davon ab, an welcher Stelle innerhalb eines Schritts der Sensor genau auslöst.

In Konfiguration 2 verlief der Anstieg gleichmässiger. Insgesamt waren die Werte etwas höher als bei der geraden Schlauchführung, dafür aber deutlich stabiler und mit weniger Streuung zwischen den Wiederholungen.

**Schlussfolgerung:** Für den späteren Einsatz wurde schliesslich eine Schrittlänge von **10 mm** gewählt. Damit bleibt das Zurückspicken zuverlässig unter 10 cm, gleichzeitig ist der Roboter aber deutlich schneller unterwegs als mit 5 mm Schrittlänge.

## 4.4 Erkennung des feststeckenden Schlauchs

In diesem Experiment wurde die Erkennung des Feststeckens in zwei Schritten evaluiert. Zunächst wurde das vollständige Tiersystem über 15 verschiedene Schlauchsituationen getestet, um die Verteilung der eingesetzten Stufen und deren Trefferquoten zu erfassen. Auf Basis dieser Ergebnisse wurde anschliessend untersucht, ob sich das System durch den alleinigen Einsatz der dominantesten Stufe vereinfachen lässt, ohne die Erkennungsqualität zu verschlechtern. Die vier Tiers und deren Funktionsweise sind in Kapitel 3.2.3.2 beschrieben.

Der Schlauch wurde in 15 verschiedene Situationen gebracht und die Erkennung live im Debug-Fenster überwacht.

### 4.4.1 Resultate

Wie die Tabelle 5 zeigt, wurde der Schlauch in 10 der 15 Situationen korrekt erkannt und gegriffen, was einer Trefferquote von **66,7 %** entspricht. Da Erkennung und Greifen in jedem getesteten Fall denselben Ausgang hatten, führt ein Erkennungsfehler unmittelbar zu einem falschen Greifpunkt.

| Situation | Tier 1 | Tier 2 | Tier 3 | Tier 4 | Korrekt erkannt? | Greifen korrekt? |
|---|---|---|---|---|---|---|
| 1 | X | | | | Ja | Ja |
| 2 | X | | | | Ja | Ja |
| 3 | X | | | | Ja | Ja |
| 4 | | | X | | Ja | Ja |
| 5 | | | X | | Ja | Ja |
| 6 | | | | X | Ja | Ja |
| 7 | | X | | | Ja | Ja |
| 8 | | | X | | Ja | Ja |
| 9 | | | X | | Nein, Bézier | Nein |
| 10 | | | X | | Nein, near zu dominant | Nein |
| 11 | X | | | | Ja | Ja |
| 12 | | | X | | Ja | Ja |
| 13 | X | | | | Nein, Loop | Nein |
| 14 | X | | | | Nein, Bézier | Nein |
| 15 | | | X | | Nein, Bézier | Nein |

Tabelle 5: Erkennung der Problemstelle über 15 Schlauchsituationen

In der Tabelle 6 fällt auf, dass Tier 1 und Tier 3 mit sechs beziehungsweise sieben Einsätzen beide sehr oft zum Zug kamen. Tier 2 und Tier 4 kamen je nur einmal zum Zug, ihre 100 %-Trefferquote ist daher wenig aussagekräftig.

| Tier | Eingesetzt | Korrekt | Falsch | Trefferquote |
|---|---|---|---|---|
| Tier 1 | 6 | 4 | 2 | 66,7 % |
| Tier 2 | 1 | 1 | 0 | 100,0 % |
| Tier 3 | 7 | 4 | 3 | 57,1 % |
| Tier 4 | 1 | 1 | 0 | 100,0 % |
| **Total** | **15** | **10** | **5** | **66.7 %** |

Tabelle 6: Trefferquote pro Tier-Variante

Bei den fünf Fehlerkennungen lassen sich drei Ursachenklassen unterscheiden. Um diese visuell besser erklären zu können, wurden Screenshots der Debug-Fenster dieser Fälle angefügt. Jedes Fenster ist viergeteilt: oben links das Kamerabild mit überlagertem Schlauchverlauf, oben rechts die reine HSV-Maske, unten links die betroffene Säule (gelber Kreis) überlagert mit der HSV Maske und der berechneten Greifposition und unten rechts das überbrückte Skelett mit Säulen mit Schlauchanfang, Bézierkurven und dem resultierenden Greifpunkt (rot) und dem Schlauchanfang (türkises Kreuz).

- **Bézier-Überlappung (3 Fälle, Situationen 9, 14, 15):** Beim Schliessen der Lücken im Schlauch-Skelett wurden zwischen den Endpunkten der Segmente kubische Bézier-Kurven gezogen. In diesen drei Situationen überlappten sich die berechneten Kurven, sodass die rekonstruierte Skelett-Topologie nicht mehr der tatsächlichen Schlauchführung entsprach. Die Abbildung 22 zeigt die zusammengezogene Schlaufe und die in die falsche Richtung laufenden Vektoren unten rechts:

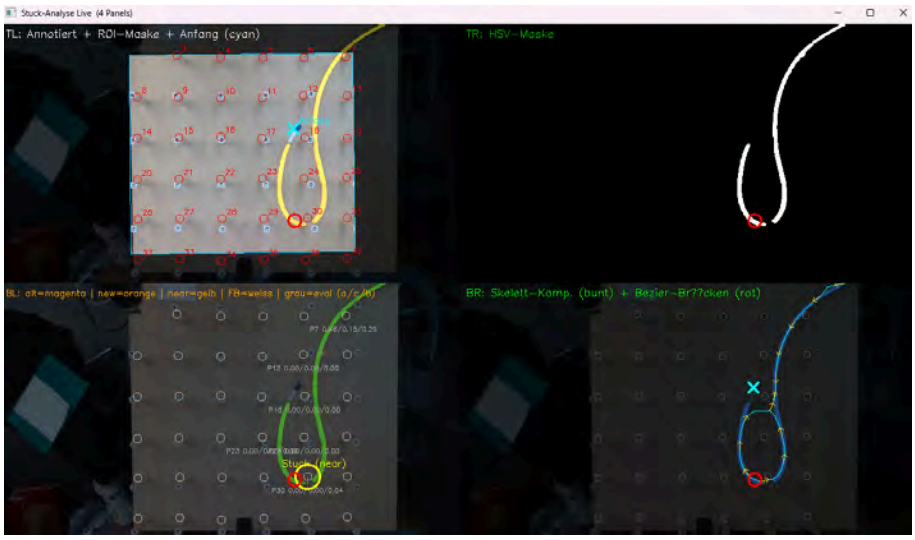



Abbildung 22: Situation 9: Der überbrückte Bogen verfälscht die Skelett-Topologie

- **Near zu dominant (1 Fall, Situation 10):** Es wurde via Tier 3 die Säule 23 erkannt, obwohl der Schlauch in dieser Situation tatsächlich an einem anderen Säule weiter vorne Richtung Kopf hängen geblieben war. Die Distanzschwelle, mit welcher in Tier 3 gesucht wird, war hier vermutlich zu eng gewählt. Die Abbildung 23 zeigt den fälschlich erkannten Punkt, und auch das eigentliche Loch in der Maske ist sichtbar:

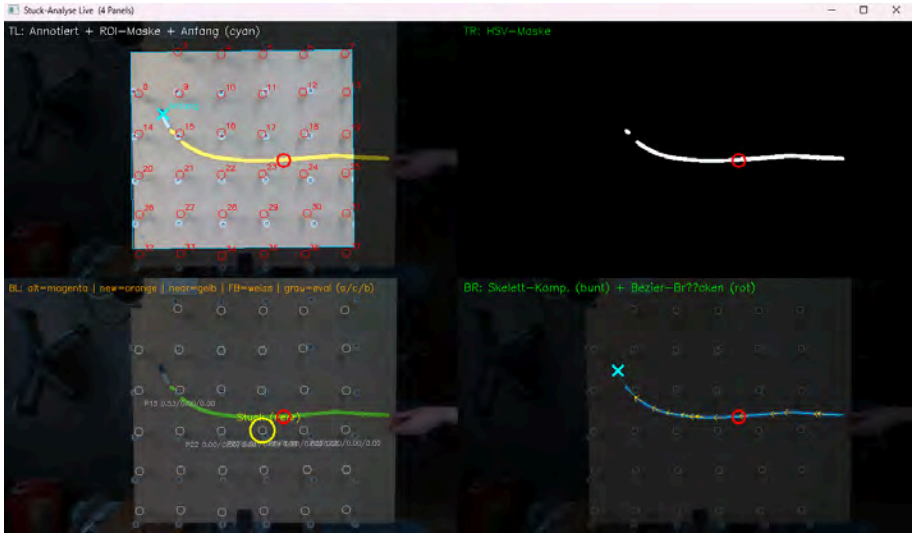



Abbildung 23: Situation 10: Tier 3 erkennt nicht vorhandene Problemstelle

- **Loop (1 Fall, Situation 13):** Der Schlauch bildete eine geschlossene Schleife. Dies führt zu Fehlern beim Aufbauen der Bézier-Kurven, und zudem lässt sich nicht feststellen, wo der Schlauch oben oder unten durchläuft. Die Abbildung 24 zeigt diesen Fehler unten rechts:

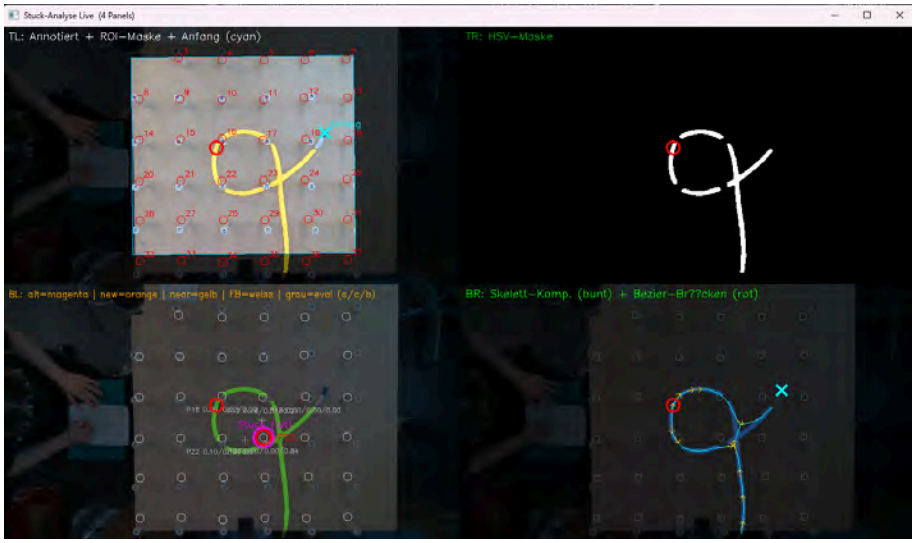



Abbildung 24: Situation 13: Die geschlossene Schleife verfälscht die Skelett-Topologie

Bei der Bewertung der Trefferquote muss berücksichtigt werden, dass die 15 Situationen bewusst nicht repräsentativ für den normalen Betrieb gewählt wurden. Ziel war es, möglichst viele schwierige oder seltene Problemstellungen zu provozieren, um Schwächen des Verfahrens sichtbar zu machen. Einige der gezeigten Fehlerfälle, insbesondere die geschlossene Schleife, traten während der übrigen Systemtests praktisch nie auf. Die effektive Trefferquote im normalen Betrieb dürfte daher deutlich höher liegen als die hier gemessenen 66,7 %.

### 4.4.2 Vereinfachung auf Tier 3 Near

Das vollständige Tiersystem benötigt 30-45 Sekunden pro Situation, da Tier 1 und Tier 2 jeweils erst vollständig durchlaufen werden müssen, bevor Tier 3 zum Zug kommt. Da Tier 3 in der ersten Evaluation am häufigsten erfolgreich war, lag die Hypothese nahe, dass Tier 3 die Aufgaben der vorgelagerten Stufen mit abdeckt und sich das System dadurch sowohl beschleunigen als auch vereinfachen lässt. Zur Überprüfung wurden zwei vereinfachte Konfigurationen mit denselben 15 Situationen wie im vorherigen Experiment getestet.

In **Konfiguration A** wurden Tier 1, Tier 2 und Tier 4 deaktiviert, sodass nur noch Tier 3 (Near) zur Erkennung verwendet wurde. Aufbauend auf den Erkenntnissen aus Konfiguration A wurde in **Konfiguration B** der Suchradius von Tier 3 leicht reduziert, um die in der ersten Evaluation beobachtete Über-Detektion (Situation 10, „Near zu dominant«) zu vermeiden, und gleichzeitig Tier 4 als Fallback wieder aktiviert.

#### 4.4.2.1 Resultate

Die Erkennungszeit sank in beiden Konfigurationen wie erwartet stark auf 1-10 Sekunden pro Situation. Die Trefferquoten unterscheiden sich jedoch, wie Tabelle 7 zeigt:

| Konfiguration | Erkennungszeit pro Situation | Trefferquote |
|---|---|---|
| Vollständiges Tiersystem (Tier 1–4) | 30–45 s | 66,7 % (10/15) |
| Konfiguration A: Tier 3 allein | 1–10 s | 60,0 % (9/15) |
| Konfiguration B: Tier 3 (kleinerer Radius) + Tier 4 | 1–10 s | 73,3 % (11/15) |

Tabelle 7: Vergleich von vollständigem Tiersystem und zwei vereinfachten Konfigurationen

Mit **Konfiguration A** wurden 9 von 15 Situationen korrekt erkannt (60,0 %), was leicht unter dem ursprünglichen Tiersystem liegt. Nicht korrekt erkannt wurden die Situationen 6, 9, 10, 13, 14 und 15. Situation 6 ist dabei ein typischer Fallback-Fall, in dem der Schlauch nicht entlang einer Säulenkante anliegt und damit für die Skelett-Proximity unsichtbar bleibt.

Mit **Konfiguration B** wurden 11 von 15 Situationen korrekt erkannt (73,3 %). Situation 6 wurde vom Fallback korrekt aufgegriffen und Situation 10 vom angepassten Near-Radius richtig zugeordnet. Die verbleibenden Fehler (Situationen 9, 13, 14, 15) gehen auf Bézier-Überlappungen beziehungsweise den Loop-Sonderfall zurück und treten unabhängig vom eingesetzten Tier auf.

**Schlussfolgerung:** Konfiguration B liefert sowohl die kürzeste Erkennungszeit als auch die höchste Trefferquote und wurde deshalb für den weiteren Betrieb gewählt. Der Fallback bleibt notwendig, weil Tier 3 allein Situationen ohne Skelett-Nähe zu einer Säule nicht abdeckt. Die verbleibenden Fehlerquellen der Bézier-Überlappungen und dem Loop sind allgemeine Fehler in der Skelett-Rekonstruktion und unabhängig von der gewählten Erkennungsstufe.

## 4.5 Systemtest

In diesem letzten Experiment wurde eine Ausgangssituation definiert und mit drei verschiedenen Schläuchen der gesamte Ablauf 20 Mal durchlaufen. Dabei wurden Fehlentscheidungen, Fehler, Zeitlänge und ob der Roboter ins Ziel gekommen ist notiert. Der Versuchsaufbau wurde mit den verschiedenen Schläuchen analog zur Ausgangssituation in Kapitel 4.2 und Kapitel 4.3 initiiert, um das gesamte System zu testen.

Folgende drei Schläuche, Schlauch A und B davon zu sehen in der Abbildung 2 und der Referenzschlauch in der Abbildung 25, wurden ausprobiert:

1. Schlauch A: Ein 70 cm langer, gelber und hohler Schlauch aus weichem Silikon
2. Schlauch B: Ein 255 cm langer, gelber und hohler Schlauch aus weichem Silikon
3. Referenz Schlauch: Ein 103 cm langer, grauer und massiver Schlauch aus hartem Silikon

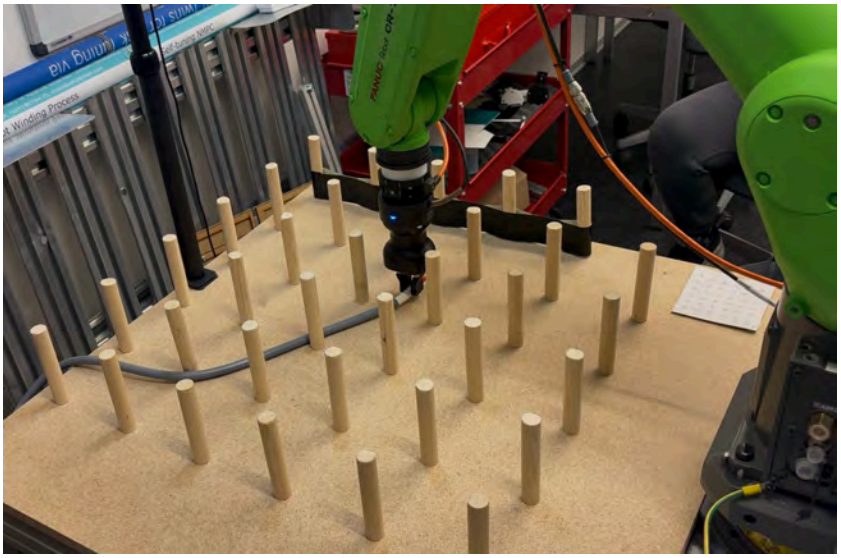


Abbildung 25: Grauer Referenzschlauch im Versuchsaufbau

Für die beiden gelben Schläuche konnten sowohl der in Kapitel 4.2 ermittelte Kraftschwellenwert von 5 N als auch die HSV-Farbgrenzen unverändert übernommen werden. Für den Referenz-Schlauch mussten dagegen beide neu kalibriert werden. Der Kraftschwellenwert wurde deutlich angehoben, weil der härtere und massive Schlauch bereits beim normalen Ziehen mehr Kraft auf den Greifer übertrug als der ursprüngliche Wert zuliess. Zusätzlich mussten die HSV-Farbgrenzen für das graue Schlauchmaterial neu bestimmt werden.

### 4.5.1 Resultate

Die in Tabelle 8 aufgeführten Ergebnisse zeigen, dass der Ring-Detektor als Erkennungsverfahren für den Schlauchanfang in diesem Versuchsaufbau grundsätzlich zuverlässig funktioniert.

| Schlauch | Erkennung Schlauchanfang | Aufgetretene Feststeckungen | Ziel erreicht | Befreiungsversuche | Korrekte Erkennungen |
|---|---|---|---|---|---|
| Schlauch A | 100 % | 0 | 100 % | n.b. | n.b. |
| Schlauch B | 100 % | 51 | 70 % | 45 | 88 % |
| Referenz-Schlauch | 95 % | 5 | 90 % | 2 | 40 % |

Tabelle 8: Systemtest-Vergleich der drei Versuchsschläuche über je 20 Durchläufe

Bei Schlauch A und B wurde eine Erkennungsrate von 100 % erreicht. Beim Referenz-Schlauch liegt diese mit 95 % leicht darunter. Dies erklärt sich dadurch, dass der blaue Startmarker zu nah am HSV-Wertebereich des grauen Schlauchkörpers lag und somit, auf einen roten Marker ausgewichen werden musste. Dieser konnte kurzfristig nicht mit derselben Präzision kalibriert werden, was die geringfügige Abweichung erklärt.

**Schlauch A** blieb in keinem der 20 Durchläufe stecken. Der kurze Schlauch erzeugt schlicht nicht genug Reibung an den Säulen, um den definierten Kraftschwellenwert von 5 N zu überschreiten. Ein Befreiungsversuch war daher nie notwendig und der Roboter erreichte in allen 20 Fällen das Ziel. Dieser Schlauch bestätigt die grundlegende Funktionsfähigkeit des Gesamtsystems, ist jedoch für eine aussagekräftige Evaluation des Befreiungsablaufs nicht geeignet.

**Schlauch B** blieb in den 20 Durchläufen insgesamt 51 Mal stecken. In 88 % dieser Fälle (45/51) konnte das System die Säule, an welcher der Schlauch hängt, korrekt identifizieren. Danach konnte der Roboter dort greifen und durch das Nachführen von zusätzlichem Schlauch in das Labyrinth den nötigen Spielraum für die weitere Navigation schaffen. 70 % der Durchläufe endeten im definierten Ziel. Die verbleibenden 12 % der Fehler lassen sich auf die zwei folgenden Ursachen zurückführen:

- Erstens führten enge Schlauchverläufe zu Überlappungen der berechneten Bézier-Kurven, was eine fehlerhafte Rekonstruktion der Schlauchtopologie und daraus folgend einen falschen Greifpunkt verursachte.
- Zweitens befand sich der Schlauch in einzelnen Situationen ausserhalb der kinematischen Reichweite des Roboters, sodass dieser in eine Singularität geriet.

Der **Referenz-Schlauch** erreichte trotz seiner abweichenden Materialeigenschaften in 90 % der Durchläufe das Ziel und blieb nur 5 Mal stecken. Sein höheres Gewicht, die tiefere Elastizität und die geringere Länge sorgen dafür, dass er glatter an den Säulen vorbeigleitet, statt sich zu verhaken. Die Erkennungsrate fällt mit 40 % (2/5) jedoch deutlich tiefer aus als bei Schlauch B. Da Grau im HSV-Farbraum keine definierte Sättigung und keinen eindeutigen Farbton besitzt, kann die HSV-Segmentierung den Schlauchkörper bei der Erkennung der Problemstelle nicht zuverlässig modellieren, was zu einer fehlerhaften Skelettrekonstruktion führt.

**Schlussfolgerung:** Die Versuche zeigen, dass ein DLO-Handhabungssystem auf die spezifischen Eigenschaften des zu handhabenden Schlauchs abgestimmt sein muss. Bereits geringe Abweichungen in Länge, Gewicht, Elastizität oder Farbe erfordern Anpassungen der HSV-Parameter, des Kraftschwellenwerts sowie gegebenenfalls des Befreiungsablaufs. Ein direkter Transfer eines für einen Schlauchtyp optimierten Systems auf einen anderen ist ohne Nachkalibrierung nicht verlässlich möglich.

# 5 Diskussion und Ausblick

Wie in Kapitel 1 dargelegt, ist die Automatisierung der Handhabung verformbarer Objekte trotz hoher wirtschaftlicher Relevanz nach wie vor ein offenes Forschungsfeld. Industrielle Verschlauchungs- und Verkabelungsaufgaben lassen sich mit heutigen robotischen Systemen nur schwer automatisieren, da DLOs wie Schläuche oder Kabel ein komplexes, schwer vorhersehbares physikalisches Verhalten zeigen und gleichzeitig fortgeschrittene visuelle sowie haptische Wahrnehmung erfordern.

Ziel dieser Arbeit war es, in diesem Feld eine Grundlage für weitere Forschungsvorhaben zu legen. Ein physisches Labyrinth sollte aufgebaut, ein Wahrnehmungssystem entwickelt und ein Programm auf Basis der Bachelorarbeit von G. Bortolin [16] implementiert werden, das einen Schlauch autonom durch genanntes Labyrinth führt. Dazu gehörten auch das Erkennen und autonome Befreien eines feststeckenden Schlauchs.

## 5.1 Zielerreichung

In der Tabelle 9 werden die drei Ziele bewertet, die zu Beginn dieser Arbeit in Kapitel 1.2.1 definiert wurden.

| **Ziel** | **Ergebnis** |
|---|---|
| **Ziel 1:**<br>Schlauch zuverlässig greifen | ✓ Vollständig erreicht<br>Der Ring-Detektor erzielt im Systemtest im Kapitel 4.5 unter Laborbedingungen eine Erkennungsrate von 100 % (für den Referenz-Schlauch mit angepasstem Marker 95 %). Die Safe-Grasp-Prüfung verhindert Kollisionen des Greifers mit den Säulen zuverlässig. |
| **Ziel 2:**<br>Kurzen Schlauch manövrieren | ✓ Vollständig erreicht<br>Die in Kapitel 3.4.4 beschriebene Pfadplanung mittels A* berechnet kollisionsfreie Pfade reproduzierbar. Und auch die Greif- und Ablagesequenz, beschrieben in Kapitel 3.2.2, laufen stabil. |
| **Ziel 3:**<br>Langen Schlauch manövrieren mit Erkennung des Feststeckens und autonomer Befreiung | ✓ Vollständig erreicht<br>Das Feststecken des Schlauchs wird durch den Kraft-Sensor zuverlässig erkannt. Im Systemtest im Kapitel 4.5 wurde der Schlauch in 88 % der Fälle (45/51) erfolgreich befreit, und der Roboter erreichte in 70 % der Durchläufe das definierte Ziel. |

Tabelle 9: Bewertung der drei Projektziele

Das System stellt einen funktionierenden Proof-of-Concept für die autonome robotische Schlauchführung dar. Die grundlegenden Mechanismen sind validiert:

- Erkennung des Schlauchanfangs
- Pfadplanung
- Kraftüberwachung
- Befreiung des feststeckenden Schlauchs

Neben der Zielerreichung im Ganzen sind einzelne Teilsysteme besonders positiv hervorzuheben. Der Ring-Detektor hat sich als robuste und verlässliche Lösung für die Erkennung von DLOs erwiesen: Die Kombination aus zwei verschiedenfarbigen Ringen liefert sowohl Position als auch Orientierung des Schlauchanfangs und ist gegenüber dem wiederholten Greifen unempfindlich.

Die Kalibrierung des Koordinatensystems aus Kapitel 3.4.1 stützt sich auf alle 35 Säulen des Labyrinths und ist damit unabhängig von einem einzelnen Referenzmarker. Sie lässt sich bei Bedarf schnell wiederholen und liefert stabile Ergebnisse.

Die in Kapitel 3.4.3 beschriebene Austauschbarkeit des Detektors durch das gewählte Strategy-Pattern hat sich bereits im Projektverlauf bewährt, da verschiedene Erkennungsalgorithmen ohne Anpassung des Hauptablaufs evaluiert werden konnten.

Auch der Versuchsaufbau selbst hat seinen Zweck erfüllt. Das Säulenlabyrinth veranschaulicht die typischen Problemstellungen bei der Handhabung von DLOs auf anschauliche Weise und hat das System wiederholt vor reale Herausforderungen gestellt.

## 5.2 Grenzen des Systems

Das entwickelte System erfüllt die gestellten Ziele unter Laborbedingungen. Dabei sind Einschränkungen zutage gekommen, die den Einsatzbereich in seiner aktuellen Form begrenzen und bei einer Weiterentwicklung adressiert werden müssen.

### 5.2.1 Skelett-Rekonstruktion bei der Problemstellen-Erkennung

Die grösste Einschränkung liegt in der Robustheit der Problemstellen-Erkennung. Wie in Kapitel 4.4 gezeigt, erreicht das finale System mit Tier 3 als primärer Erkennungsstufe und Tier 4 als Fallback eine Trefferquote von 73,3 % über die 15 bewusst provozierten Extremsituationen. Im Systemtest, beschrieben in Kapitel 4.5, unter realen Bedingungen lag die Erfolgsquote mit 88 % (45/51) nochmals höher, was bestätigt, dass die Fehlerfälle des isolierten Tests hauptsächlich seltene Sonderkonstellationen abbilden. Das strukturelle Defizit der Bézier-Interpolation bleibt dennoch bestehen:

Drei der vier Fehlerkennungen sind auf dieselbe Ursache zurückzuführen: die kubische Bézier-Interpolation beim Schliessen von Lücken im Schlauch-Skelett. Treten mehrere Lücken zu nah beieinander auf, können sich die berechneten Kurven topologisch überschneiden. Die daraus resultierende Skelett-Geometrie entspricht nicht mehr dem tatsächlichen Schlauchverlauf, und die nachfolgende BFS-basierte Reihenfolge der Säulen stützt sich auf eine fehlerhafte Grundlage. Dies ist kein Problem, das sich durch Anpassung eines Schwellenwerts beheben lässt, sondern ein strukturelles Defizit der rein geometrischen Interpolation ohne Physikmodell des Schlauchs.

### 5.2.2 Abhängigkeit vom Schlauchmatrial

Alle kalibrierten Parameter des Systems wurden auf den gelben Testschlauch abgestimmt: der Kraftschwellenwert von 5 N, die Schrittlänge von 10 mm sowie die HSV-Farbgrenzen der Schlaucherkennung. Schläuche aus anderem Material mit abweichendem Reibungskoeffizienten, anderer Elastizität oder anderer Farbe erfordern eine vollständige Neukalibrierung dieser Parameter. Der Systemtest im Kapitel 4.5 mit dem grauen Schlauch hat dies im Ansatz bestätigt.

### 5.2.3 Restriktionen des Versuchsaufbaus

Dass der Versuchsaufbau die typischen Problemstellungen bei der Handhabung von DLOs gut abbildet, wurde bereits in Kapitel 5.1 festgehalten. Dieselbe Komplexität des Aufbaus hat jedoch auch dazu geführt, dass für den sicheren Betrieb des Roboters eine Vielzahl von Regeln notwendig war. Bestimmte Korridore mussten statisch als unpassierbar hinterlegt werden, da der Roboter dort in Singularitäten gerät. Singularitäten stellten dabei eine wiederkehrende Herausforderung dar, die sich nicht in jeder Situation vollständig im Voraus planen liess und erst durch praktisches Erproben sichtbar wurde. Diese Einschränkungen begrenzen die Menge der möglichen Schlauchpfade und damit die Szenarien, in denen das System produktiv eingesetzt werden kann. Für weiterführende Arbeiten empfiehlt es sich, die Erreichbarkeit aller relevanten Arbeitsbereiche bereits in der Planungsphase des Versuchsaufbaus systematisch zu prüfen.

### 5.2.4 Dreidimensionale Schlauchrekonstruktion

Im Rahmen der Arbeit wurde RANSAC als Ansatz zur Erkennung des Schlauchs über die Tiefendaten der RealSense-Kamera evaluiert. Der Ansatz scheiterte daran, dass die Säulen des Labyrinths in der Punktwolke erhebliches Rauschen erzeugen. Da die Säulen mit 10 cm deutlich höher sind als der Schlauch mit maximal 1.2 cm Durchmesser, liess sich kein verlässlicher Punktwolken-Ausschnitt des Schlauchs isolieren. Für einen Versuchsaufbau ohne vertikale Hindernisse oder mit einer anderen Kamerapositionierung könnte RANSAC ein vielversprechender Ansatz für eine vollständige dreidimensionale Schlauchrekonstruktion sein.

## 5.3 Ungelöste Problemstellungen

Nicht alle im Projektverlauf identifizierten Probleme konnten innerhalb des Rahmens dieser Arbeit gelöst werden. Die folgenden Problemstellungen bestehen im aktuellen System weiterhin.

### 5.3.1 Schlauchschleifen

Bildet der Schlauch eine geschlossene Schleife, lässt sich aus dem zweidimensionalen Kamerabild nicht erkennen, welcher der sich kreuzenden Schlauchstränge oben und welcher unten verläuft. Damit ist auch der tatsächliche Verlauf des Schlauchs nicht eindeutig bestimmbar und das System wählt in diesem Fall einen falschen Greifpunkt.

### 5.3.2 Tageslichtempfindlichkeit

Der Ring-Detektor erreicht unter Tageslicht eine Erkennungsrate von 5 %. Die Ursache liegt nicht im Erkennungsverfahren als solchem, sondern spezifisch im weissen Referenzring. Die weisse HSV-Maske basiert auf hoher Helligkeit bei niedriger Sättigung und reagiert damit empfindlich auf Beleuchtungsschwankungen. Ein Ersetzen des weissen durch einen zweiten farbigen Ring würde dieses Problem beheben, da Hue-basierte Masken gegenüber wechselnden Lichtverhältnissen robuster sind. Da alle Versuche unter kontrollierter Laborbeleuchtung durchgeführt wurden, hatte die Tageslichtempfindlichkeit keinen Einfluss auf die Resultate.

## 5.4 Verbesserungsmöglichkeiten

Die in Kapitel 5.2.1 und Kapitel 5.3 beschriebenen Probleme legen konkrete Ansatzpunkte für Verbesserungen nahe.

### 5.4.1 Skelett-Rekonstruktion

Der grösste Hebel für eine höhere Trefferquote der Problemstellen-Erkennung liegt in der Lückenrekonstruktion des Schlauch-Skeletts. Die kubische Bézier-Interpolation ist anfällig für topologische Fehler bei mehreren gleichzeitigen Lücken. Ein vielversprechender Ansatz wäre der Einsatz eines physikbasierten Schlauchmodells, etwa eines Cosserat-Stab-Modells oder eines Elastic-Spline, das den Schlauchverlauf an Lückenstellen anhand der physikalischen Eigenschaften des Materials approximiert statt rein geometrisch zu interpolieren.

### 5.4.2 Lern-basierte Erkennung der Problemstelle

Ergänzend zur regelbasierten Erkennung könnte ein Machine-Learning-Modell trainiert werden, das aus dem Kamerabild und den bekannten Säulenpositionen direkt lernt, in welchen Situationen der Schlauch an welcher Säule hängt und welche Lösungsbewegung am wirksamsten ist. Der im Projektverlauf aufgebaute Datensatz an Feststeck-Situationen könnte als Grundlage für ein solches Training dienen. Ein solches Modell wäre potenziell robuster gegenüber den Sonderfällen, die die aktuelle regelbasierte Erkennung nicht abdeckt, insbesondere Schlauchschleifen und Bézier-Überlappungen.

### 5.4.3 Farbiger Referenzring

Das Ersetzen des weissen Rings durch einen zweiten farbigen Ring, etwa in Rot oder Grün, würde die Tageslichtempfindlichkeit des Ring-Detektors beheben. Hue-basierte Masken sind gegenüber wechselnden Lichtverhältnissen inhärent robuster als helligkeitsbasierte und würden den Detektor auch ausserhalb kontrollierter Laborumgebungen einsetzbar machen.

### 5.4.4 Parameter auf Material abstimmen

Kraftschwellenwert, Schrittlänge und HSV-Farbgrenzen sind aktuell fest auf den gelben Testschlauch kalibriert. Eine Kalibrierungsroutine, die diese Parameter automatisch auf ein neues Schlauchmaterial abstimmt, würde den Aufwand bei einem Materialwechsel erheblich reduzieren und die Wiederverwendbarkeit des Systems verbessern.

### 5.4.5 Vermeidung von Singularitäten

Im Systemtest in Kapitel 4.5 geriet der Roboter wiederholt in Singularitäten, weil der berechnete Greifpunkt am Rand der kinematischen Reichweite lag. Im Laufe der Arbeit zeigte sich, dass die Ursache häufig in der Berechnung der Greifer-Rotation lag. Aktuell wird die Drehung auf eine von vier festen Achsrichtungen (0°, 90°, 180°, 270°) gerundet, was bei ungünstiger Ausgangsorientierung zu Drehungen von fast einer vollen Umdrehung führt, obwohl eine kleine relative Korrektur ausgereicht hätte. Eine Delta-basierte Rotation, die vom aktuellen Greiferwinkel ausgeht und nur die nötige Differenz fährt, würde diese überflüssigen Drehbewegungen vermeiden und damit eine häufige Ursache für Singularitäten beseitigen.

Ergänzend dazu wäre eine sauberere Modellierung des erreichbaren Arbeitsraums sinnvoll. Aktuell wurde lediglich eine einzelne Korridorreihe statisch als unerreichbar hinterlegt, in der bekannte Singularitäten auftreten. Eine explizite Bounding-Box des sicher erreichbaren Bereichs oder alternativ das Abfangen von Motion-Limit-Errors zur Laufzeit mit anschliessender Neuberechnung eines alternativen Greifpunkts würde das System robuster gegenüber bisher nicht vorgesehenen Posen machen.

### 5.4.6 Detektionsverfahren

Basierend auf den Ergebnissen des Detektions-Benchmarkings aus Kapitel 4.1 lassen sich für weiterführende Arbeiten konkrete Empfehlungen ableiten. Eine ArUco-basierte Segmentierung des Schlauchs ist aufgrund der langen Verarbeitungszeiten und der instabilen Rotationsschätzung unter wechselnden Lichtverhältnissen nicht zu empfehlen. Der Ring-Detektor hingegen hat sich als robuste und rechen effiziente Lösung für die Erkennung des Schlauchanfangs bewährt und ist für diese Aufgabe anderen Verfahren vorzuziehen.

Für die Erkennung der Problemstelle könnte ein lern-basierter Ansatz der aktuellen regelbasierten Erkennung überlegen sein. Steht ein repräsentativer Datensatz an Feststeck-Situationen zur Verfügung, bietet sich YOLO als Segmentierungsmodell an, das direkt aus dem Kamerabild lernt, an welcher Säule der Schlauch hängt. Das CylinderTag-Verfahren [21], welches kreisförmige Merkmale auf zylindrischen Oberflächen nutzt und damit die bekannten Schwächen planarer Marker umgeht, wäre zudem ein interessanter Ansatz für eine robustere markerbasierte Erkennung des Schlauchanfangs.

## 5.5 Industrielle Anwendungsmöglichkeiten

Die in dieser Arbeit entwickelten Teilsysteme adressieren Problemstellungen, die in industriellen Verschlauchungs- und Verkabelungsaufgaben regelmässig auftreten. Die kraft-sensorbasierte Erkennung des Feststeckens, die visuelle Greifpunktbestimmung an DLOs sowie die autonome Befreiung sind Bausteine, die sich grundsätzlich auf reale Montageszenarien übertragen lassen, etwa das automatisierte Verlegen von Kühlschläuchen in der Automobilindustrie, das Einziehen von Kabelbäumen in Kabelkanälen oder das Führen von Hydraulikleitungen in beengten Bauräumen.

Neben dem in dieser Arbeit gewählten Ansatz, den Schlauch am Boden entlang durch das Labyrinth zu ziehen, wurden zwei weitere Strategien diskutiert, aber nicht umgesetzt. Beim Heben über Säulen würde der Roboter den Schlauch anheben, über eine Säule legen und auf der anderen Seite absenken. Dies würde das Hängenbleiben am Boden vermeiden, erfordert jedoch eine präzise Höhenkontrolle und ein Verständnis der dreidimensionalen Schlauchgeometrie unter Schwerkrafteinfluss. Beim Wurf-Ansatz würde der Schlauch in der Luft um Säulen geworfen, was kürzere Zykluszeiten ermöglichen könnte, aber dynamische Bewegungsplanung und ein Modell des Schlauchfluges voraussetzt. Der gewählte Ansatz ist von den dreien derjenige, der am nächsten an einer industriellen Umsetzung liegt, da er reproduzierbar, rechen effizient und mit etablierten CV-Methoden umsetzbar ist.

Diese Arbeit legt damit eine Grundlage, auf der die IAI am CAI der ZHAW weiteraufbauen kann. Ein naheliegender nächster Schritt wäre die Ablösung der regelbasierten Problemstellen-Erkennung durch einen lern-basierten Ansatz, der aus gesammelten Feststeck-Situationen selbst lernt, an welcher Säule der Schlauch hängt und welche Lösungsbewegung am wirksamsten ist. Darüber hinaus könnte die Aufgabe erweitert werden: Statt den Schlauch lediglich durch das Labyrinth zu führen, könnte der Roboter ihn abschliessend an einer Verschraubung befestigen und damit eine vollständige Verschlauchungsaufgabe autonom ausführen.

# Literaturverzeichnis


[1] D. Seita *u. a.*, «Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks», in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China: IEEE, 2021, S. 4568–4575. doi: 10.1109/ICRA48506.2021.9561391.

[2] P. Sundaresan *u. a.*, «Learning Rope Manipulation Policies Using Dense Object Descriptors Trained on Synthetic Depth Data», in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, Frankreich: IEEE, 2020, S. 9411–9418. doi: 10.1109/ICRA40945.2020.9197121.

[3] F. Liu, E. Su, J. Lu, M. Li, und M. C. Yip, «Robotic Manipulation of Deformable Rope-Like Objects Using Differentiable Compliant Position-Based Dynamics», *IEEE Robotics and Automation Letters*, Bd. 8, Nr. 7, S. 3964–3971, 2023, doi: 10.1109/LRA.2023.3264766.

[4] B. Balaguer und S. Carpin, «Combining imitation and reinforcement learning to fold deformable planar objects», in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA: IEEE, 2011, S. 1405–1412. doi: 10.1109/IROS.2011.6094992.

[5] P. M. Scheikl, E. Tagliabue, B. Gyenes, M. Wagner, D. Dall'Alba, und P. Fiorini, «Sim-to-Real Transfer for Visual Reinforcement Learning of Deformable Object Manipulation for Robot-Assisted Surgery», *IEEE Robotics and Automation Letters*, Bd. 8, Nr. 2, S. 560–567, 2023, doi: 10.1109/LRA.2022.3227873.

[6] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, und Y. Mezouar, «Robotic manipulation and sensing of deformable objects in domestic and industrial applications: A survey», *The International Journal of Robotics Research*, Bd. 37, Nr. 7, S. 688–716, 2018, doi: 10.1177/0278364918779698.

[7] A. Caporali, I. Cuiral-Zueco, G. López-Nicolás, und G. Palli, «Robotic perception and manipulation of deformable linear objects: A survey», *The International Journal of Robotics Research*, 2026, doi: 10.1177/02783649261432253.

[8] J. Redmon, S. Divvala, R. Girshick, und A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection», in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, 2016, S. 779–788. doi: 10.1109/CVPR.2016.91.

[9] International Organization for Standardization, «ISO 8373:2021: Robotics – Vocabulary». Zugegriffen: 20. Mai 2026. [Online]. Verfügbar unter: https://www.iso.org/standard/75539.html

[10] International Federation of Robotics, «World Robotics 2025 – Industrial Robots», Frankfurt am Main, Deutschland, technical report, 2025. Zugegriffen: 20. Mai 2026. [Online]. Verfügbar unter: https://ifr.org/worldrobotics/report-2025

[11] Encyclopaedia Britannica, «Automation – Robots in Manufacturing». Zugegriffen: 24. Mai 2026. [Online]. Verfügbar unter: https://www.britannica.com/technology/automation/Robots-in-manufacturing

[12] B. Siciliano, L. Villani, G. Oriolo, und A. De Luca, *Foundations of Robotics*. in Advanced Textbooks in Control and Signal Processing. Cham, Schweiz: Springer Nature Switzerland AG, 2025. doi: 10.1007/978-3-031-85523-8.

[13] Matheretter, «Homogene Koordinaten». Zugegriffen: 3. Juni 2026. [Online]. Verfügbar unter: https://www.matheretter.de/wiki/homogene-koordinaten

[14] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 4. Aufl. Hoboken, NJ, USA: Pearson, 2021.

[15] FANUC Switzerland GmbH, «HandlingTool Grundkurs». Biel, Schweiz.

[16] G. Bortolin, «Fanuc Python-Driver», unveröffentlicht, Bachelorarbeit, Winterthur, Schweiz, 2025.

[17] K. Hata und S. Savarese, «CS231A Course Notes 1: Camera Models». Zugegriffen: 3. Juni 2026. [Online]. Verfügbar unter: https://web.stanford.edu/class/cs231a/course_notes/01-camera-models.pdf

[18] OpenCV, «Camera Calibration and 3D Reconstruction: solvePnP». Zugegriffen: 24. Mai 2026. [Online]. Verfügbar unter: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html

[19] T. Collins und A. Bartoli, «Infinitesimal Plane-Based Pose Estimation», *International Journal of Computer Vision*, Bd. 109, Nr. 3, S. 252–286, 2014, doi: 10.1007/s11263-014-0725-5.

[20] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, und M. J. Marín-Jiménez, «Automatic Generation and Detection of Highly Reliable Fiducial Markers Under Occlusion», *Pattern Recognition*, Bd. 47, Nr. 6, S. 2280–2292, 2014, doi: 10.1016/j.patcog.2014.01.005.

[21] S. Wang, M. Zhu, Y. Hu, D. Li, F. Yuan, und J. Yu, «CylinderTag: An Accurate and Flexible Marker for Cylinder-Shape Objects Pose Estimation Based on Projective Invariants», *IEEE Transactions on Visualization and Computer Graphics*, Bd. 30, Nr. 12, S. 7486–7499, 2024, doi: 10.1109/TVCG.2024.3350901.

[22] H.-C. Kang, H.-N. Han, H.-C. Bae, M.-G. Kim, J.-Y. Son, und Y.-K. Kim, «HSV Color-Space-Based Automated Object Localization for Robot Grasping without Prior Knowledge», *Applied Sciences*, Bd. 11, Nr. 16, S. 7593, 2021, doi: 10.3390/app11167593.

[23] OpenCV, «Changing Colorspaces». Zugegriffen: 24. Mai 2026. [Online]. Verfügbar unter: https://docs.opencv.org/4.x/df/d9d/tutorial_py_colorspaces.html

[24] OpenCV, «Morphological Transformations». Zugegriffen: 24. Mai 2026. [Online]. Verfügbar unter: https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html

[25] OpenCV, «Contours: Getting Started». Zugegriffen: 24. Mai 2026. [Online]. Verfügbar unter: https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html

[26] OpenCV, «Contour Features». Zugegriffen: 24. Mai 2026. [Online]. Verfügbar unter: https://docs.opencv.org/4.x/dd/d49/tutorial_py_contour_features.html

[27] scikit-image, «skimage.morphology.skeletonize». Zugegriffen: 24. Mai 2026. [Online]. Verfügbar unter: https://scikit-image.org/docs/stable/api/skimage.morphology.html

[28] K. S. Arun, T. S. Huang, und S. D. Blostein, «Least-Squares Fitting of Two 3-D Point Sets», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Nr. 5, S. 698–700, 1987, doi: 10.1109/TPAMI.1987.4767965.

[29] SciPy, «scipy.interpolate.RBFInterpolator». Zugegriffen: 24. Mai 2026. [Online]. Verfügbar unter: https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.RBFInterpolator.html

[30] A. Elfes, «Using Occupancy Grids for Mobile Robot Perception and Navigation», *Computer*, Bd. 22, Nr. 6, S. 46–57, 1989, doi: 10.1109/2.30720.

[31] OpenCV, «distanceTransform». Zugegriffen: 24. Mai 2026. [Online]. Verfügbar unter: https://docs.opencv.org/4.x/d7/d1b/group__imgproc__misc.html

[32] P. E. Hart, N. J. Nilsson, und B. Raphael, «A Formal Basis for the Heuristic Determination of Minimum Cost Paths», *IEEE Transactions on Systems Science and Cybernetics*, Bd. 4, Nr. 2, S. 100–107, 1968, doi: 10.1109/TSSC.1968.300136.

[33] U. Ramer, «An Iterative Procedure for the Polygonal Approximation of Plane Curves», *Computer Graphics and Image Processing*, Bd. 1, Nr. 3, S. 244–256, 1972, doi: 10.1016/S0146-664X(72)80017-0.

[34] FANUC America Corporation, «CR-7iA/L Collaborative Robot – Product Specifications». Zugegriffen: 3. Juni 2026. [Online]. Verfügbar unter: https://www.fanucamerica.com/products/robots/series/collaborative-robot/cr-7ia-l-cobot

[35] International Organization for Standardization, «ISO 10218-1:2025: Robotics – Safety requirements – Part 1: Industrial robots». Zugegriffen: 3. Juni 2026. [Online]. Verfügbar unter: https://www.iso.org/standard/73933.html

[36] Robotiq Inc., «Hand-E Adaptive Gripper – Instruction Manual». Zugegriffen: 3. Juni 2026. [Online]. Verfügbar unter: https://assets.robotiq.com/website-assets/support_documents/document/Hand-E_Manual_UniversalRobots_PDF_20191219.pdf

[37] Bota Systems AG, «SensONE 6-axis Force Torque Sensor (BFT-SENS-IND2-B4)». Zugegriffen: 3. Juni 2026. [Online]. Verfügbar unter: https://www.botasys.com/force-torque-sensors/sensone

[38] Intel Corporation, «Intel RealSense Depth Camera D435i – Product Specifications». Zugegriffen: 3. Juni 2026. [Online]. Verfügbar unter: https://www.intel.com/content/www/us/en/products/sku/190004/intel-realsense-depth-camera-d435if/specifications.html

[39] Bota Systems AG, «Bota FT Stack Documentation – Driver Concept». Zugegriffen: 25. Mai 2026. [Online]. Verfügbar unter: https://code.botasys.com/en/gen_a/layer1/driver/driver_concept.html

# Glossar

| Abkürzung | Ausgeschrieben | Beschreibung | Seiten |
|---|---|---|---|
| A* | A-Star-Algorithmus | Heuristischer Suchalgorithmus zur Pfadplanung in gewichteten Graphen, der effizient den kostengünstigsten Pfad bestimmt. | 9, 17, 22, 25, 27, 29, 51 |
| API | Application Programming Interface | Programmierschnittstelle, die die Kommunikation zwischen Softwarekomponenten ermöglicht. | 5, 15, 32 |
| ArUco | Augmented Reality University of Córdoba | Quadratische Markierungen mit eindeutigen Mustern zur Positions- und Orientierungsschätzung von Objekten im Kamerabild. | 5, 6, 15, 16, 17, 23, 39, 40, 54 |
| BFS | Breadth-First Search | Graphsuchalgorithmus, der Knoten schichtweise besucht und in ungewichteten Graphen kürzeste Pfade findet. | 7, 35, 52 |
| CAI | Centre for Artificial Intelligence | Institut der ZHAW School of Engineering, das Forschung und Lehre im Bereich künstliche Intelligenz bündelt. | 1, 2, 55 |
| CV | Computer Vision | Teilgebiet der Informatik zur automatischen Analyse und Interpretation visueller Informationen aus Bildern und Videos. | 1, 2, 4, 55 |
| DLO | Deformable Linear Object | Verformbares lineares Objekt wie ein Kabel, Seil oder Schlauch. Aufgrund seiner hohen Verformbarkeit ist die robotische Handhabung anspruchsvoll. | 1, 10, 11, 50, 51, 52, 55 |
| DoF | Degree of Freedom | Anzahl unabhängiger Bewegungsrichtungen eines Systems. Industrieroboter besitzen typischerweise sechs oder mehr Freiheitsgrade. | 3, 4, 10 |
| Flansch | Roboter-Werkzeugflansch | Genormte Anschlussplatte am Ende des Roboterarms zur Montage von Werkzeugen und Sensoren. | 10, 12, 13, 18 |

| Abkürzung | Ausgeschrieben | Beschreibung | Seiten |
|---|---|---|---|
| HSV | Hue, Saturation, Value | Farbraummodell, das Farben durch Farbton, Sättigung und Helligkeit beschreibt. Wird häufig für Farbsegmentierung verwendet. | 6, 14, 15, 16, 18, 27, 34, 35, 39, 40, 46, 49, 50, 52, 53, 54 |
| IAI | Industrial Artificial Intelligence Group | Forschungsgruppe am CAI Centre for Artificial Intelligence der ZHAW School of Engineering, welche die vorliegende Arbeit betreut. | 1, 55 |
| IPPE | Infinitesimal Plane-based Pose Estimation | Verfahren zur präzisen Poseschätzung planarer Objekte anhand ihrer Projektion im Kamerabild. | 6 |
| JSON | JavaScript Object Notation | Leichtgewichtiges textbasiertes Format zur strukturierten Speicherung und Übertragung von Daten. | 5, 15, 20 |
| OpenCV | Open Source Computer Vision Library | Open-Source-Bibliothek für Computer Vision und Bildverarbeitung mit Funktionen für Kamerakalibrierung, Objekterkennung und Bildanalyse. | 5, 15 |
| PnP | Perspective-n-Point | Verfahren zur Bestimmung der Position und Orientierung eines Objekts relativ zur Kamera anhand bekannter 3D-Punkte und ihrer 2D-Projektionen. | 5 |
| RANSAC | Random Sample Consensus | Robuster Schätzalgorithmus zur Bestimmung eines Modells aus verrauschten Daten unter gleichzeitiger Erkennung von Ausreissern. | 16, 53 |
| RDP | Ramer-Douglas-Peucker | Algorithmus zur Vereinfachung von Linienzügen, der überflüssige Punkte entfernt und den Verlauf weitgehend beibehält. | 9 |
| RGB | Red, Green, Blue | Additives Farbmodell, das Farben durch die Grundfarben Rot, Grün und Blau beschreibt. | 6, 14 |

| Abkürzung | Ausgeschrieben | Beschreibung | Seiten |
|---|---|---|---|
| RMI | Remote Motion Interface | FANUC-Schnittstelle zur Steuerung und Überwachung von Roboterbewegungen über ein Netzwerk. | 5, 15, 20, 21, 30 |
| SDK | Software Development Kit | Sammlung von Werkzeugen, Bibliotheken und Dokumentation zur Entwicklung von Software für eine bestimmte Plattform. | 15, 20 |
| SVD | Singulärwertzerlegung | Verfahren der linearen Algebra zur Zerlegung einer Matrix in drei Matrizen. Wird unter anderem für Approximationen und Koordinatentransformationen verwendet. | 8, 23 |
| TCP | Tool Center Point | Bezugspunkt eines Roboterwerkzeugs, auf den sich Positions- und Orientierungsangaben beziehen. | 3, 4 |
| TCP/IP | Transmission Control Protocol / Internet Protocol | Netzwerkprotokoll-Suite für die Adressierung, Übertragung und zuverlässige Zustellung von Daten. | 5, 15, 20 |
| TPS | Thin-Plate-Spline | Verfahren zur glatten Interpolation zwischen Stützpunkten, das auch nichtlineare Verformungen modellieren kann. | 8, 23 |
| UF5 | User Frame 5 | Benutzerdefiniertes Koordinatensystem mit Index 5 in der FANUC-Steuerung. Es dient in dieser Arbeit als Referenzkoordinatensystem. | 8, 17, 23, 29 |
| YOLO | You Only Look Once | Familie von Echtzeit-Objekterkennungsmodellen zur gleichzeitigen Detektion und Klassifikation von Objekten in Bildern. | 2, 16, 54 |
| ZHAW | Zürcher Hochschule für Angewandte Wissenschaften | Hochschule mit mehreren Standorten in Zürich und Winterthur, an der diese Arbeit verfasst wird. | 1, 55 |

# Abbildungsverzeichnis

Abbildung 1 Versuchsaufbau . . . 11
Abbildung 2 Kurzer und langer gelber Schlauch aus weichem Silikon . . . 11
Abbildung 3 FANUC Robot CR-7iA/L . . . 12
Abbildung 4 ROBOTIQ Hand-E Greifer . . . 13
Abbildung 5 Kraft-Moment-Sensor Bota Systems BFT-SENS-IND2-B4 . . . 13
Abbildung 6 RealSense D435iF montiert an ein Meter hohem Stativ . . . 14
Abbildung 7 Top-Level-Architektur . . . 20
Abbildung 8 Modulübersicht innerhalb `experiments/hoses` . . . 21
Abbildung 9 Hauptloop eines Schlauchführungszyklus . . . 24
Abbildung 10 Klassendiagramm von Vision und Detection . . . 26
Abbildung 11 Ablauf der Pfadplanung . . . 28
Abbildung 12 Beispielhaft berechneter Pfad (Debug-Ausgabe) . . . 29
Abbildung 13 Ablauf vom Erstellen der Roboter-Bewegungssequenzen . . . 31
Abbildung 14 Komponenten der Schlauch-Befreiung und ihr Zusammenspiel . . . 33
Abbildung 15 Gesamtablauf der Lokalisierung der Problemstellen . . . 34
Abbildung 16 Debug-Output aus Top-Down-Ansicht mit Greifpunkt und Zugziel . . . 36
Abbildung 17 Ablauf des Finden eines sicheren Greifpunkts . . . 37
Abbildung 18 Ablauf der Schlauchanfang-Ermittlung bei verdecktem Ring . . . 38
Abbildung 19 Schlauch links neben dem Tisch befestigt und in Position für Experiment . . . 41
Abbildung 20 Konfiguration 1: Ausgangsposition für die gerade Schlauchführung . . . 43
Abbildung 21 Konfiguration 2: Ausgangsposition für die diagonale Schlauchführung . . . 43
Abbildung 22 Situation 9: Der überbrückte Bogen verfälscht die Skelett-Topologie . . . 46
Abbildung 23 Situation 10: Tier 3 erkennt nicht vorhandene Problemstelle . . . 46
Abbildung 24 Situation 13: Die geschlossene Schleife verfälscht die Skelett-Topologie . . . 47
Abbildung 25 Grauer Referenzschlauch im Versuchsaufbau . . . 49

# Tabellenverzeichnis

Tabelle 1 Detektions-Benchmarking der drei Erkennungsverfahren unter drei Lichtszenarien . . . . . 39
Tabelle 2 Schlauchlänge beim Auslösen des Kraft-Sensors pro Schwellenwert . . . . . 42
Tabelle 3 Ergebnisse mit Konfiguration 1: Schlauch gerade entlang des Roboterpfads . . . . . 44
Tabelle 4 Ergebnisse mit Konfiguration 2: Schlauch diagonal durch das Labyrinth gespannt . . . . . 44
Tabelle 5 Erkennung der Problemstelle über 15 Schlauchsituationen . . . . . 45
Tabelle 6 Trefferquote pro Tier-Variante . . . . . 45
Tabelle 7 Vergleich von vollständigem Tiersystem und zwei vereinfachten Konfigurationen . . . . . 48
Tabelle 8 Systemtest-Vergleich der drei Versuchsschläuche über je 20 Durchläufe . . . . . 50
Tabelle 9 Bewertung der drei Projektziele . . . . . 51
Tabelle 10 Während der Erstellung der Arbeit eingesetzte KI-Werkzeuge und ihre Verwendungszwecke . . . . . 64

# A Nutzungsdeklaration künstlicher Intelligenz

Im Zuge der Erstellung dieser Bachelorarbeit kamen die in Tabelle 10 aufgeführten KI-basierten Werkzeuge an verschiedenen Stellen unterstützend zum Einsatz. Sämtliche damit erzeugten Inhalte wurden vor der Übernahme kritisch geprüft, inhaltlich validiert und bei Bedarf überarbeitet. Die zugrundeliegenden Konzepte, Analysen und Schlussfolgerungen wurden vollständig von den Autorinnen entwickelt.

| **KI-Werkzeug** | **Verwendungszweck** |
|---|---|
| Claude | Code-Vervollständigung und -Generierung sowie Unterstützung beim Verfassen und Korrigieren von Textabschnitten der Arbeit. |
| ChatGPT | Umformulierung von Textpassagen und englische Übersetzung einzelner Abschnitte, insbesondere des Abstracts. |
| Google Search AI Summary | KI-generierte Zusammenfassungen, die bei der regulären Google-Websuche automatisch eingeblendet werden. |

Tabelle 10: Während der Erstellung der Arbeit eingesetzte KI-Werkzeuge und ihre Verwendungszwecke

# B Projektbeschreibung

January 2026

## Robotic Tube Handling

Students Felicitas Farah Baur, Theresa Valentina Neubert
Supervisors Dr. Alisa Rupenyan, Diego Bolliger

### Positioning of the thesis

While research and technologies for manipulating rigid bodies with robots are mature, the handling of deformable objects is an ongoing area of research with many open questions. Example topics include manipulating linear deformable objects such as ropes [1], [2], [3], folding and handling cloth [4] and applications in robotic surgery like lifting tissue. [5]. To cope with the challenges of deformation, the robotic system requires advanced perception, compliance, and understanding of the physical behavior of materials. At the industrial AI group, we are investigating these problems with the goal of advancing the capabilities of robotic systems in industrial settings.

In the context of industrial assembly, tubing and cabling are important tasks involving deformable objects that are difficult to automate with robots. The goal of this project is to lay a foundation for further research in this area by developing a robotic setup for autonomously handling tasks in tubing such as connecting tubes to fittings.

### Objectives

The main objective of this project is to enable an industrial robot arm to autonomously perform a task required for tubing in assembly, such as connecting and disconnecting a hose from a fitting. To this end, the students will decide on a task, construct an experimental setup and chose an appropriate hose and additional material. Using a camera and computer vision algorithms, the students should develop a perception system to localize the hose and the other objects. Building on a previously developed Python driver or by using ROS drivers, the students should then write a program for the robot arm to grasp the hose and perform the task.

### Plan

The steps and corresponding success criteria are as follows:

1. Familiarization with Robot: The students need to get comfortable using the robotic arm independently both with the physical control pad and through software.
2. Design experiment setup. This means the students have to decide the following things:
   - Decide on a task to automate.
   - Choose a hose and select additonal material.
   - Decide on experiment layout.
   - Define a measure of success and failure.
3. Perception of hose and additonal objects: Using the provided camera, the hose and any other objects need to be perceived and put in context with the robot's coordinates. This can be accomplished with basic computer vision strategies such as markers or by using more advanced machine learning models like YOLO.

4. Robot program: The students then need to develop a program that performs the task. Using the spatial information from the perception system, the robot will have to grab the hose at an appropriate location and manipulate it to fulfill the task. The goal is that the robot makes a proper attempt at the task.

5. Refinement: Due to the deformability of the hose, it is unlikely that a successful execution of the task is easily performed. Hence, a refinement and optimization of the method is needed. Here the students will enhance the workflow, movements of the robot, and perception system to improve task completion.

6. Evaluation: In the evaluation phase, the students should evaluate how well their method works. Further, they should document what problems can arise during the task execution and what might be done to deal with them.

7. Documentation: Finally, the students should document their work in a report.

## References

[1] D. Seita et al., "Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China: IEEE, May 30, 2021, pp. 4568–4575, ISBN: 978-1-72819-077-8. DOI: 10.1109/ICRA48506.2021.9561391. Accessed: Apr. 19, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/9561391/.

[2] P. Sundaresan et al., "Learning Rope Manipulation Policies Using Dense Object Descriptors Trained on Synthetic Depth Data," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France: IEEE, May 2020, pp. 9411–9418, ISBN: 978-1-72817-395-5. DOI: 10.1109/ICRA40945.2020.9197121. Accessed: Apr. 19, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/9197121/.

[3] F. Liu, E. Su, J. Lu, M. Li, and M. C. Yip, "Robotic Manipulation of Deformable Rope-Like Objects Using Differentiable Compliant Position-Based Dynamics," *IEEE Robot. Autom. Lett.*, vol. 8, no. 7, pp. 3964–3971, Jul. 2023, ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2023.3264766. Accessed: Apr. 19, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10093017/.

[4] B. Balaguer and S. Carpin, "Combining imitation and reinforcement learning to fold deformable planar objects," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA: IEEE, Sep. 2011, pp. 1405–1412, ISBN: 978-1-61284-456-5 978-1-61284-454-1 978-1-61284-455-8. DOI: 10.1109/IROS.2011.6094992. Accessed: Oct. 3, 2024. [Online]. Available: http://ieeexplore.ieee.org/document/6094992/.

[5] P. M. Scheikl et al., "Sim-to-Real Transfer for Visual Reinforcement Learning of Deformable Object Manipulation for Robot-Assisted Surgery," *IEEE Robot. Autom. Lett.*, vol. 8, no. 2, pp. 560–567, Feb. 2023, ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2022.3227873. Accessed: Apr. 19, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/9976185/.

# C Quellcode-Ablage

Der im Rahmen dieser Arbeit entstandene Quellcode ist im ZHAW-internen GitHub-Repository **Fanuc_Driver** auf dem Branch **BA_Neubethe_Baufel01** abgelegt und unter folgender Adresse erreichbar:

https://github.zhaw.ch/boie/Fanuc_Driver/tree/BA_Neubethe_Baufel01

Neu hinzugekommen sind insbesondere:

- `experiments/hoses/`, das vollständige Projekt der autonomen Schlauchführung,
- `modules/ForceSensor_Class.py`, ein generischer Treiber für den Bota Kraft-Moment-Sensor.

Im obersten Verzeichnis des Repositorys befindet sich zudem eine neue README-Datei. Sie erläutert die Ordnerstruktur, beschreibt den Aufbau des Projekts und dient nachfolgenden Studierenden als Einstiegspunkt für die Einarbeitung und Weiterentwicklung.