KI-basierte dynamische Montageplanung und Demontageplanung von ähnlichen Produkten

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

David Timmermann

Tag der mündlichen Prüfung: 24.04.2026

1. Referent/Referentin: Prof. Dr. Rüdiger Dillmann
2. Referent/Referentin: Prof. Dr. Jürgen Fleischer

Danksagung

Als ich während meines Bachelorstudiums am KIT die Möglichkeit hatte, verschiedenste Robotikvorlesungen zu besuchen, wurde mein Interesse an diesem Themenfeld geweckt. Über Bachelor und Master hinweg begleitete mich Robotik in verschiedensten Vorlesungen, Praktika und Abschlussarbeiten. Als ich dann die Möglichkeit erhielt, mich im Rahmen meiner Masterarbeit tiefer mit Hardware- und Softwarefragestellungen rund um das Thema robotische Montage auseinanderzusetzen, legte dies den Grundstein für meine weitere Forschung.

Die vorliegende Arbeit ist in meiner Zeit als wissenschaftlicher Mitarbeiter und stellv. Abteilungsleiter der Abteilung Interaktive Diagnose- und Servicesysteme (IDS) am FZI Forschungszentrum Informatik entstanden. Meine Arbeit am FZI hat mir ermöglicht, meine wissenschaftliche Neugier durch spannende Projekte, interessante Konferenzen und eine tolle Zusammenarbeit mit verschiedensten Kollegen zu befriedigen.

Bei Herrn Prof. Dr.-Ing. Rüdiger Dillmann möchte ich mich besonders für seine wissenschaftliche Betreuung und das Interesse an dieser Arbeit bedanken. Die vielen Diskussionen, ob im Rahmen der Klausurtagung oder bei unseren Promotionsgesprächen, haben mir geholfen, das Thema gezielt weiterzuentwickeln und das Ziel der Arbeit im Blick zu behalten. Auch möchte ich an dieser Stelle Prof. Dr.-Ing. Jürgen Fleischer vom Institut für Produktionstechnik am KIT für die Übernahme des Korreferats danken.

Ein großer Dank geht auch an meinen ehemaligen Abteilungsleiter Prof. Arne Rönnau sowie meine aktuellen Abteilungsleiter Georg Heppner und Tristan Schnell. Dank euch hatte ich die Möglichkeit, viele verschiedene Projekte zu bearbeiten und mich in verschiedenen Projektrollen zu probieren. Auch meinen jetzigen und ehemaligen IDS-Kollegen möchte ich danken. Viele lange Abende und arbeitsreiche Wochenenden waren vor allem dank euch nur halb so schlimm. An dieser Stelle sei besonders Johannes Mangler, Carsten Plasberg, Jakob Weinland, Friedrich Graaf, Marvin Große Besselmann, Felix Exner, Stefan Scherzinger, Pascal Becker und Timothee Büttner für erinnerungswürdige Dienstreisen und die gemeinsame Arbeit an spannenden Projekten gedankt.

Ohne die tatkräftige Unterstützung von vielen Studierenden im Rahmen ihrer Abschlussarbeiten oder Hiwistellen wäre diese Arbeit nicht möglich gewesen. Mein Dank geht daher an Anastasiia Maklashevskikh, Claudius Kocher, Dominik Heid, Felix Schemenz, Florian Pohl, Jakob Weise, Lennart Nachtigal, Nicolas Kessler und Thomas Friedel, die alle mit ihrer Unterstützung zu meiner Forschung beigetragen haben.

Ein riesiger Dank geht auch an meine Familie. Bei meinen Eltern Ulrich und Ursula möchte ich mich bedanken, weil sie mein Technikinteresse geweckt und gefördert sowie mir den Weg in das Studium ermöglicht haben. Für ein offenes Ohr und ihre ansteckende Zuversicht möchte ich an dieser Stelle meiner Schwester Lea danken. Meinem Onkel Hartmut gilt mein Dank, dass er mich in meinem Technikinteresse sowohl in meiner Jugend als auch während meines Studiums bestärkt hat. Mikrocontroller wären für mich ohne seine Hilfe vermutlich weiterhin ein Buch mit sieben Siegeln. Für die vielen tiefgehenden Diskussionen über das wissenschaftliche Arbeiten und die vielen Hinweise für Verbesserungspotenzial geht mein Dank an meine Schwiegermutter Roswitha. Auch meinem Freund Adrian möchte ich an dieser Stelle danken; ohne die Freundschaft mit dir und die tatkräftige Unterstützung beim Lernen im Studium wäre ich heute vermutlich nicht an diesem Punkt im Leben. Die Erinnerungen an die Zeit in unserer gemeinsamen WG in Grötzingen sind einige der besten, die ich habe.

Widmen möchte ich diese Arbeit meiner Frau Magdalena. Über all die Jahre hat sie mich in meinem Promotionsbestreben bestärkt und motiviert. Magdalena, ohne deine vielfältige Unterstützung wäre diese Arbeit nicht möglich gewesen. Sei es nun beim Rückenfreihalten im Haushalt, dem offenen Ohr, wenn es mal wieder schwierig und frustrierend war, oder auch einfach deine liebevolle Art - all das hat einen unschätzbaren Beitrag für diese Arbeit geleistet. Gerade seitdem unsere Tochter Laura Teil unseres Lebens ist, merke ich, was du für eine wertvolle Stütze für mich bist.

Abschließend will ich meiner Tochter Laura Danke sagen und mich für die Nachmittage und Abende entschuldigen, an denen ich nicht mit dir spielen konnte. Ich verspreche, dass wir die verpassten Spielstunden alle nachholen werden.

Pfinztal, im Februar 2026 David Timmermann

Abstract (English Version)

Automation and robotics have become indispensable in many areas of modern life. Particularly in industrial environments, they are integral components of numerous processes and workflows. However, there are various domains and situations where automation reaches its limits due to a lack of flexibility. In assembly, fluctuating availability of materials and machinery causes issues in the execution of statically planned assembly sequences. In recycling and disassembly, on the other hand, product degradation and general variant diversity increase the need for developing corresponding adaptive capabilities.

This thesis presents a concept for AI-based dynamic planning of assembly and disassembly processes for similar products. It aims to provide greater flexibility within these domains. To this end, AI-based approaches for solving assembly sequence planning are developed and adapted. By leveraging state-of-the-art methods such as Large Language Models in conjunction with detailed assembly information derived from CAD data, assembly processes can be planned automatically. With the introduction of the concept of assembly affordances, the individual action possibilities of a component can be systematically captured and addressed during the subsequent planning process. In a plan refinement process following sequence generation, these assembly affordances are utilized by a hierarchical semantic planner to translate the identified component sequences into concrete, high-level assembly plans. The resulting plan can subsequently serve as dynamic input for an intelligent robot cell that executes the actual assembly process.

Finally, the developed concepts and implementations are evaluated and compared through experiments using both real-world and synthetic data.

As this thesis demonstrates, the intelligent application of current AI methods allows for the flexibilization of previously static assembly workflows. Furthermore, concepts such as assembly affordances enable a transparent adaptation of sequences, allowing the resulting plans to serve as a solid foundation for subsequent automated execution.

Kurzfassung

Automatisierung und Robotik sind aus vielen heutigen Lebensbereichen nicht mehr wegzudenken. Besonders im Industrieumfeld gehören sie zu integralen Bestandteilen vieler Prozesse und Abläufe. Gleichwohl gibt es diverse Bereiche und Situationen in denen die Automatisierung aufgrund fehlender Flexibilität in ihre Grenzen kommt. In der Montage sorgen veränderte Verfügbarkeiten von Material und Maschinen für Probleme bei der Ausführung von statisch geplanten Montageabläufen. Im Recycling und der Demontage hingegen steigt durch Produktveränderungen sowie der allgemeine Variantenvielfalt der Bedarf entsprechende Anpassungsfähigkeiten zu entwickeln.

In dieser Arbeit wird ein Konzept für die KI-basierte dynamische Planung von Montage- und Demontageabläufen für ähnliche Produkte vorgelegt. Es zielt darauf eine größere Flexibilität in diesen Bereichen bereitzustellen. Dazu werden KI-basierte Ansätze für die Lösung der Montagesequenzplanung entwickelt und adaptiert. Durch aktuelle Methoden wie Large Language Modelle in Verbindung mit detaillierten Baugruppeninformationen aus CAD-Daten können automatisiert Montagevorgänge geplant werden. Mit der Einführung des Konzeptes der Montageaffordanz können die einzelnen Verwendungsmöglichkeiten eines Bauteils systematisch erfasst und im weiteren Planungsverlauf adressiert werden. In einem nach der Sequenzgenerierung angesiedelten Planverfeinerungsprozess, werden die entsprechenden Montageaffordanzen von einem hierarchischen semantischen Planer verwendet, um die gefundenen Bauteilsequenzen in konkrete Montagepläne auf einem hohen Abstraktionsniveau zu überführen. Der so erzeugte Plan kann anschließend als dynamische Eingabe für eine intelligente Roboterzelle genutzt werden, die den eigentlichen Montageprozess durchführt.

Die entwickelten Konzepte und Implementierungen werden abschließend mit Experimenten auf realen und künstlichen Daten evaluiert und verglichen.

Wie die vorliegende Arbeit zeigt, können durch den intelligenten Einsatz von heute existierenden KI-Verfahren bisher statische Prozessabläufe in der Montage flexibilisiert werden. Zudem hinaus ermöglichen Konzepte wie die Montageaffordanzen eine nachvollziehbare Anpassung von Sequenzen, um die resultierenden Pläne als Grundlage für eine spätere automatisierte Ausführung zu verwenden.

Verwendung von Large Language Models In dieser Arbeit wurden textgenerierende Künstliche Intelligenzen (Large Language Models), darunter ChatGPT und GitHub Copilot, unterstützend eingesetzt. Der Einsatz beschränkte sich auf die sprachliche Korrektur (Grammatik, Orthografie) sowie die stilistische Überarbeitung von selbst verfassten Textpassagen. Es wurden keine KI-Tools zur Generierung von wissenschaftlichen Inhalten, Thesen oder Forschungsergebnissen verwendet. Die Verantwortung für alle Inhalte liegt vollständig beim Verfasser.

Inhaltsverzeichnis

1. Einführung

Die robotische Montage sowie die robotische Demontage sind nicht nur stark wachsende Themenfelder, sondern zugleich entscheidende Faktoren für aktuelle und zukünftige Entwicklungen in der Fertigung und der Kreislaufwirtschaft. Industrieroboter und auch leistungsstarke Serviceroboter finden heute bereits Verwendung, um schnelle und repetitive Manipulationsaufgaben, wie z. B. das Handhaben von schweren und unhandlichen Bauteilen und Baugruppen, zu automatisieren. Derzeit sind solche Systeme überwiegend noch statisch programmiert, sodass bei Veränderungen - unabhängig vom Umfeld oder dem zu bearbeitenden Objekt - eine Anpassung der Programmierung notwendig ist. Insbesondere für grundlegende Programmabläufe, die steuern, in welcher Reihenfolge ein Produkt zusammengesetzt oder zerlegt werden soll, ist es notwendig, dass ein Roboterprogrammierer das Programm und ggf. die Robotertrajektorien adaptiert. Damit verbunden sind unerwünschte Stillstandszeiten bis zum Abschluss der erforderlichen Anpassung. Zudem spielt eine Rolle, ob und wann die entsprechend qualifizierte Fachkraft verfügbar ist. Eine Lösung stellen automatisierte Planungssysteme dar, welche Veränderungen, etwa neue Objektvarianten, erkennen und auf Basis der neuen Modelle die notwendigen Änderungen implementieren. Besonders der Einsatz von künstlicher Intelligenz (KI) eröffnet in diesem Zusammenhang ein interessantes, aktives und zukunftsorientiertes Forschungsfeld im Bereich der Robotik.

1.1. Motivation

Die Montageplanung im Zusammenhang mit Robotik ist bereits seit langer Zeit ein aktives Forschungsfeld. Erste Ansätze versuchten, das Problem mit graphenbasierten Methoden wie Liaison-Graphen [1] zu lösen, während spätere Ansätze das Planungsproblem als Optimierungsproblem interpretierten und mit Optimierungsverfahren wie Partikelschwarmoptimierung [2] eine Lösung erzeugten. Frühe Durchbrüche konnte die Montageplanung und die damit einhergehende Automatisierung etwa im Bereich der Fahrzeugmontage oder der generellen Fertigung verzeichnen. Diese Szenarien basieren alle auf einer Gemeinsamkeit: Sie sind statisch und vorhersagbar. Die Entwicklungen der letzten Jahre haben allerdings gezeigt, dass diese Sichtweise nicht mehr als gegeben angenommen werden kann. Verschiedene Ereignisse der letzten fünf Jahre zeigen dies deutlich: Die COVID-19-Pandemie, die Blockade des Suezkanals im Jahr 2021, der Ukraine-Konflikt oder aktuell die zunehmend unzureichende Verfügbarkeit von

Speicherbausteinen aufgrund der hohen Nachfrage durch die KI-Industrie haben einen massiven Einfluss auf die Verfügbarkeit von Bauteilen. Innerhalb kürzester Zeit sind Komponenten nicht mehr erhältlich und müssen durch andere ersetzt werden, was Anpassungen an Produkten und deren Zusammenbau zur Folge hat. Diese Anpassungen manuell zu realisieren, ist vor dem Hintergrund des Fachkräftemangels herausfordernd. Darüber hinaus sind die bisher etablierten Methoden in der Regel mit langen Laufzeiten verbunden. Künstliche Intelligenz kann an diesen Punkten unterstützen, indem der zeitaufwendige Teil, das Training der KI, bereits vorab auf Beispielen von ehemaligen Montagen erfolgt. Die eigentliche Montageplanung durch die KI benötigt verhältnismäßig wenig Zeit und kann durch die Generalisierung während des Trainings mit geänderten Bauteilen umgehen.

Eine ähnliche Situation ergibt sich im Kontext von Kreislaufwirtschaft und Demontage als einem sehr entscheidenden Bereich, wenn es um die Themen Nachhaltigkeit und Souveränität bei Rohstoffen geht. Besonders um die höherwertigen „Rs“ der Kreislaufwirtschaft - Wiederverwenden (Reuse), Reparieren (Repair) und Generalüberholen (Remanufacture) - zu adressieren, ist eine möglichst vollständige und zerstörungsfreie Zerlegung notwendig. Im Vergleich zur Montage ändert sich hier jedoch die Grundannahme, denn in diesem Fall kommt die Varianz bei dem zu zerlegenden Produkt primär durch die Produktalterung und Variantenvielfalt zustande. Es gibt viele Produkte in sehr ähnlicher Form von verschiedenen Herstellern, auch ihr Aufbau ähnelt sich. Oftmals wird bei der Zerlegung erst kurz vor der Ausführung des eigentlichen Demontageprozesses bekannt, welches Produkt genau zerlegt wird. Manuelle Programmierung oder Ansätze mit langer Rechenzeit sind in solchen Fällen nicht sinnvoll anwendbar, sodass bisher nicht selten auf manuelle Zerlegung durch Menschen zurückgegriffen wird. Für eine sinnvolle Umsetzung des Kreislaufgedankens ist eine Automatisierung unumgänglich. KI kann mit ähnlichen Prinzipien wie bei der Montage verwendet werden, um den Planungsprozess unterstützend schneller und flexibler zu gestalten. Die Relevanz dieses Forschungsfeldes lässt sich auch an den verschiedenen existierenden Sonderforschungsbereichen erkennen. So beschäftigt sich am Karlsruher Institut für Technologie der *SFB 1574 - Kreislauffabrik für das ewige Produkt* [3] mit verschiedensten Fragestellungen rund um die Überführung von gebrauchten Produkten in aktuelle Produktgenerationen.

1.2. Problemstellung und Forschungsfrage

Im Kontext der Planung von Montage und Demontage liegt der Fokus der vorliegenden Arbeit auf der Generierung von symbolischen High-Level-Plänen für die Montage oder Demontage von ähnlichen Produkten. Die Themenstellung wird differenziert in drei Teilbereiche:

- **Sequenzplanung:** Um ausgehend von einem gegebenen Produkt eine mögliche Sequenz für die korrekte Montage oder Zerlegung zu bestimmen, muss

aus allen möglichen Kombinationen der vorkommenden Bauteile eine korrekte Reihenfolge gefunden werden, welche mindestens kollisionsfrei ist. Häufig gibt es auch weitere Anforderungen, welche die Effizienz des Prozesses erhöhen. Beispiele dazu sind die Reduktion von Werkzeugwechseln oder die Reduktion der Länge der Robotertrajektorien. Das System betrachtet hierfür theoretisch $n!$ mögliche Lösungen, wobei n der Anzahl der Bauteile entspricht. Für jeden einzelnen Schritt in einer potenziellen Reihenfolge ist zu prüfen und sicherzustellen, dass die zuvor definierten Kriterien eingehalten werden. Da diese Überprüfungen mit klassischen Ansätzen, etwa Simulationen, in Summe sehr langwierig sind, soll das System dieser Arbeit die Überprüfungen nicht explizit durchführen müssen.

- **Aufgabenplanung:** Ausgehend von einer gefundenen Reihenfolge, in der die Bauteile zu handhaben sind, ist eine zusätzliche Definition der korrekten Manipulation (Füge- oder Zerlegeoperation) für jeden einzelnen Schritt in der Reihenfolge notwendig. Die korrekte Manipulationsaktion ist von verschiedensten Faktoren abhängig. So entscheidet beispielsweise ein Übermaß darüber, ob ein Bauteil in ein anderes hineingepresst werden muss oder hineingleiten kann. Auch eine Schraube interagiert über den Verlauf ihrer Montage unterschiedlich mit verschiedenen Bauteilen (Durchstecken durch eine Unterlegscheibe vs. Einschrauben in ein Gewinde). Im Rahmen dieser Arbeit soll das System imstande sein, für jede Bauteilkombination und Füge-/Zerlegesituation die korrekte Manipulationsaktion herzuleiten.
- **Komplexe Modelldaten:** Beim Einsatz von KI ist die mögliche Komplexität der Eingangsdaten häufig limitiert. Mächtige Sprachmodelle verarbeiten beispielsweise nur Textinformationen, Bilderkennungsverfahren nutzen Kamera- oder Laserscannerdaten, welche in einem 2D- oder 3D-Array hinterlegt sind. Viele für eine Baugruppe relevante Parameter sind häufig in komplexen Datenformaten, meist CAD, hinterlegt. Um Parameter wie Materialeigenschaften oder Dichte zu beachten, ist es daher für den in der Arbeit verwendeten Ansatz notwendig, auch komplexe Datenstrukturen sinnvoll einzubinden.

Ausgehend von den einzelnen Problemfeldern ergibt sich folgendes Forschungsziel:

> **Forschungsziel 1.** Diese Arbeit zielt darauf ab, einen Ansatz zu entwickeln, der unter Einbindung komplexer Produktdaten für eine neue Variante eines existierenden Produktes eine mögliche Montage- oder Demontagesequenz generiert. Die auf diese Weise generierte Sequenz soll anschließend durch konkrete Manipulationsanweisungen in jedem Sequenzschritt erweitert werden, und zwar so, dass die Sequenz in einen symbolischen High-Level-Plan überführt wird, der anschließend von einer intelligenten Automatisierungszelle genutzt werden kann.

Hieraus ergeben sich nachstehende konkrete Forschungsfragen:

> **Forschungsfrage 1.** In welchem Format und mit welcher Methodik können komplexe Produktdaten einem KI-System zugänglich gemacht werden?

> **Forschungsfrage 2.** Wie sieht eine KI-Architektur aus, welche eine Montage- bzw. Demontagesequenz generieren kann, und wie gut generalisiert ein solcher Ansatz?

> **Forschungsfrage 3.** Mit welcher Methodik ist es möglich, eine Sequenz ohne Manipulationsinformationen in einen symbolischen High-Level-Plan zu überführen?

Das Forschungsziel und die Forschungsfragen werden unter folgenden Annahmen bearbeitet:

- Für eine gegebene Baugruppe existiert eine entsprechende CAD-Datei oder eine gleichwertige Informationsquelle.
- Bei den betrachteten Baugruppen handelt es sich um neue Varianten oder leicht angepasste Varianten von bereits behandelten Baugruppen.
- Die im symbolischen Plan aufgeführten Füge- oder Zerlegeoperationen sind für die nachgelagerte automatisierte Roboterzelle umsetzbar.
- Die generierte Sequenz muss physikalisch realisierbar sein, jedoch nicht optimal unter beliebigen Kriterien. Sie kann den Input für mögliche Optimierungsverfahren darstellen.
- Demontage und Montage werden im Rahmen dieser Arbeit als inverses Problem betrachtet. Betrachtete Baugruppen haben daher die Eigenschaft, dass sie durch zerstörungsfreie Demontageoperationen zerlegt werden können.

1.3. Konzept der Arbeit

In dieser Arbeit werden Montage- und Demontageplanung als zwei Teilbereiche bearbeitet. Das Gesamtkonzept beschreibt ein System, das ausgehend von Informationen über eine aktuell vorliegende Variante einer Baugruppe zunächst eine mögliche Sequenz für die Montage bzw. Demontage definiert und anschließend die Sequenz zu einem symbolischen Plan verfeinert, der als Liste von Handlungsanweisungen für eine intelligente Roboterzelle verwendet werden kann. Das Konzept ist in Abbildung 1.1 dargestellt.

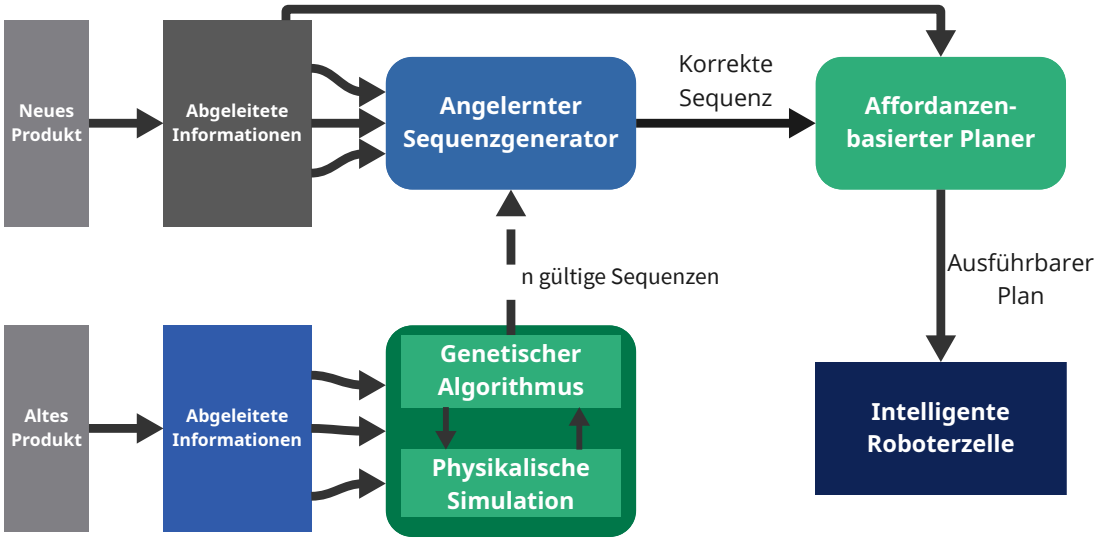


Abbildung 1.1.: Generelles Konzept des entwickelten Frameworks.

Zu Beginn des Prozesses greift das System auf die Informationen der in der Vergangenheit behandelten Baugruppen zu. Daraus werden mittels klassischer Ansätze gültige optimierte Montage- oder Demontagesequenzen abgeleitet, mit dem Ziel, einen Datensatz für das Training der KI zu erstellen. Dieser Schritt kann übersprungen werden, wenn ein für die Baugruppenklasse existierender Datensatz zur Verfügung steht. Die im Datensatz enthaltenen Sequenzen sollen gewissen Qualitätskriterien genügen, sodass die KI im Folgeschritt auf möglichst optimalen Beispielen trainiert werden kann.

Nach erfolgreicher Erstellung eines Trainingsdatensatzes wird dieser genutzt, um eine KI auf diesem Datensatz zu trainieren. Dabei fließen auch die Informationen über die aktuelle Baugruppe, welche in Form einer CAD-Datei oder eines ähnlichen Konstruktes vorliegen, mit ein. Es werden dabei unterschiedliche KI-Ansätze verglichen und hinsichtlich ihrer Eignung für die Thematik eingeschätzt. In der Inferenz generiert die KI für die gegebene Baugruppe eine gültige Sequenz, wobei „gültig“ primär auf den Aspekt der Kollisionsfreiheit abzielt.

Ausgehend von der so generierten Sequenz erfolgt abschließend die Überführung in einen symbolischen Plan. Dazu wird das Konzept der Affordanzen aufgegriffen und zu Montageaffordanzen erweitert. Ein KI-basiertes Bildverarbeitungssystem segmentiert die Affordanzen der einzelnen Bauteile. In der Folge werden diese Informationen von einem semantischen Planer mit den Informationen aus den CAD-Daten und der generierten Sequenz kombiniert, um pro Sequenzschritt die korrekte Manipulationsaktion abzuleiten. Der so erzeugte symbolische Plan kann abschließend einer intelligenten Roboterzelle für die Ausführung der Montage bzw. Demontage zur Verfügung gestellt werden.

1.4. Wissenschaftlicher Beitrag

In dieser Arbeit wird ein ganzheitliches Framework für die Generierung von symbolischen High-Level-Plänen für die Montage- und Demontageplanung vorgestellt. Das entwickelte Konzept erlaubt eine dynamische Planerstellung unter Berücksichtigung von komplexem Produktwissen. Dies ermöglicht eine Anpassung von Manipulationsabläufen ohne lange Stillstandszeiten. Der Hauptbeitrag dieser Forschung ist:

- Ein Framework für die automatische Umwandlung komplexer Produktdaten in Form von CAD-Daten in einen Wissensgraphen (engl. Knowledge Graph KG). Ein Wissensgraph stellt eine semantische Datenbankstruktur dar, die Wissen als Netzwerk aus Entitäten (wie Personen, Orten oder Konzepten) und deren Beziehungen zueinander speichert, um Daten in einen maschinenlesbaren Kontext zu setzen. Dabei legt das Framework einen starken Fokus auf die automatisierte Verarbeitung der CAD-Daten, sodass manuelle Eingriffe in den Prozess minimiert werden. Durch die Aufbereitung der Informationen in einem Wissensgraphen liegen diese in einem Format vor, welches einer künstlichen Intelligenz durch Serialisierung auf einfachem Weg zugänglich gemacht werden kann. Ein weiterer Vorteil der Aufbereitung stellt die übersichtliche und verständliche Darstellung der Informationen dar, sodass unvollständige Ausgangsinformationen zielgerichtet ergänzt werden können. Zusätzlich kann dieses Konzept als Basis für die Integration von anderen komplexen Informationsquellen dienen, um sie einem KI-System wie einem Large Language Model zugänglich zu machen.

- Ein Ansatz für die dynamische Erstellung von Montage- und Demontagesequenzen hinsichtlich neuer Varianten bekannter Baugruppen. Das umfangreiche, aber für spezielle Anwendungsfälle meist unzureichende Grundlagenwissen wird durch die spezifischen Baugruppeninformationen aus dem Wissensgraphen ergänzt und verfeinert. Durch diese zusätzliche Informationsquelle zur Laufzeit ist es möglich, ohne spezifisches Fine-Tuning für verschiedene Baugruppen Sequenzen zu erzeugen. Um die Korrektheit der

vorgeschlagenen Sequenzen sicherzustellen, wird die KI durch generalisierte Validierungsalgorithmen und eine auf Vision Language Models basierte Grounding-Komponente ergänzt, deren Feedback vom Large Language Model berücksichtigt wird.

- Ein affordanzbasierter Ansatz für die Überführung von Sequenzen in konkrete High-Level-Pläne, welche auch Manipulationsanweisungen für jeden Sequenzschritt enthalten. Durch die Ableitung der benötigten Affordanzklassen auf Basis der DIN 8580 berücksichtigt das Konzept alle real vorkommenden Fügeoperationen. Die komplexe und aufwändige händische Segmentierung der Bauteile anhand der einzelnen Affordanzklassen kann das Konzept durch einen Bild-basierten KI-Ansatz automatisieren. Die gewonnenen Informationen werden durch einen semantischen Planer weiterverarbeitet und als Basis für die Ableitung der korrekten Manipulationsaktion verwendet. Der semantische Planer stellt dabei durch eine klar definierte und überprüfbare Ableitungslogik sicher, dass die Zuweisung der Aktionen zu jedem Sequenzschritt korrekt ist.

Die Machbarkeit des vorgeschlagenen Konzeptes wird durch unterschiedliche Analysen und Experimente anhand von verschiedenen Baugruppen gestützt. Die Arbeit soll den Einsatz von KI im Kontext von Montage- und Demontageplanung fördern, indem sie eine mögliche technische Repräsentation von Produktwissen in Verbindung mit aktuellen KI-Methoden aufzeigt.

1.5. Aufbau der Arbeit

Die Abhandlung ist wie folgt aufgebaut:

Kapitel 2 führt Grundlagen und Begriffe zu dieser Arbeit ein. Zunächst wird das Themenfeld der Fertigungstechnik mit einem Fokus auf die Aspekte der Montageplanung und der Verbindungstechnik dargestellt. Anschließend wird der Bereich des maschinellen Lernens vorgestellt. Der Fokus liegt auf neuronalen Netzen und deren Ausprägung als Large Language Model bzw. Vision Language Model. Das Kapitel schließt ab mit einer Vorstellung des Konzeptes des Wissensgraphen und dem Konzept der Affordanzen.

Kapitel 3 untersucht den aktuellen Stand der Technik. Es fokussiert sich dabei auf die relevanten Themen der Montageplanung und die Fähigkeiten aktueller KI-Methoden.

Kapitel 4 führt in das entwickelte Konzept der Arbeit ein. Die Teilbereiche des Konzeptes werden kurz erläutert und Anknüpfungspunkte zu artverwandten Themen aufgezeigt.

Kapitel 5 beschreibt detailliert das Vorgehen bei der Montagesequenzerstellung. Zunächst werden die Aspekte der Trainingsdatengenerierung, welche über einen genetischen Algorithmus gelöst wurden, erläutert. Anschließend erfolgt eine detaillierte Vorstellung zweier entwickelter KI-Ansätze für die Generierung von Montagesequenzen. Zunächst wird der Sequenz-zu-Sequenz-Ansatz vorgestellt. Abschließend findet die Vorstellung des zweiten Ansatzes statt, welcher einen Wissensgraphen und ein Large Language Model für die Sequenzerstellung nutzt.

Kapitel 6 konzentriert sich auf die Überführung der Sequenz in einen High-Level-Plan durch die Verwendung von Montageaffordanzen. Hierzu wird in einem ersten Schritt das Konzept der Montageaffordanzen skizziert. Auf dieser Basis wird ein Ansatz für die automatisierte Segmentierung von Bauteilen anhand der Montageaffordanzen vorgestellt. Das Kapitel schließt ab mit einer detaillierten Vorstellung des semantischen Planers für die Generierung der korrekten Fügeoperationen.

Kapitel 7 stellt die ausführliche Evaluation aller erarbeiteten Konzepte vor, welche in dieser Arbeit vorgeschlagen wurden. Die Zuverlässigkeit und Genauigkeit der Vorhersage der KI-Komponenten für die Sequenzerstellung sowie deren Laufzeitverhalten wird anhand verschiedener Baugruppen untersucht. Hierbei werden sowohl künstlich erzeugte Baugruppen als auch Baugruppen aus realen Anwendungen berücksichtigt. Basierend auf den generierten Sequenzen wird anschließend die Überführung in High-Level-Pläne anhand entsprechender Baugruppen evaluiert. Die Diskussion zu den Ergebnissen der Untersuchung schließt das Kapitel ab.

Kapitel 8 fasst die wesentlichen Erkenntnisse der Arbeit zusammen und gibt einen Ausblick auf Forschungsperspektiven.

2. Grundlagen zur KI-basierten Montageplanung

2.1. Fertigungstechnik

Fertigungstechnik ist eine Wissensgebiet im Maschinenbau und den damit verbundenen Bereichen. Sie befasst sich grundlegend mit der Herstellung von festen Körpern (Werkstücke, Bauteile und Produkte) durch entsprechende Fertigungsverfahren. Weitere Aspekte, die im Rahmen der Fertigungstechnik betrachtet werden sind die Planung, Organisation und Realisierung von Produktionsprozessen [4].

2.1.1. Montagetechnik

Die Montagetechnik ist ein zentraler Teilbereich der Fertigungstechnik. Sie beschäftigt sich mit der systematischen Verbindung von Einzelteilen zu komplexeren Baugruppen. Montageprozess basiere hauptsächlich auf den fünf Grundfunktionen Fügen, Handhaben, Prüfen, Justieren und Sonderoperationen, wobei das Fügen die dauerhafte oder lösbare Verbindung von Komponenten realisiert. Neben den prozesstechnischen Abläufen umfasst das Feld die Systemtechnik, welche die physische Umsetzung durch Montageanlagen sowie deren Verkettung mittels Förder- und Zuführtechnik beschreibt.

Ergänzend dazu bildet die Montageplanung den notwendigen organisatorischen Rahmen, dartunter fallen die Erstellung von Vorranggraphen und die Layoutplanung der Produktionsumgebungen. Ein wesentlicher Aspekt ist die montagegerechte Produktgestaltung (engl. Design for Assembly), welche bereits in der Konstruktionsphase darauf fokussiert ist, die spätere Montagekomplexität durch strukturierte Bauteilgestaltung zu minimieren. Des Weiteren sind qualitätssichernde Maßnahmen wie das Prüfen und Justieren relevant, um die Funktionalität des Endprodukts innerhalb definierter Toleranzen zu gewährleisten. Somit integriert die Montagetechnik operative Prozessschritte, anlagentechnische Realisierungen und planerische Strategien zu einem ganzheitlichen Produktionssystem [5].

2.1.2. Demontagetechnik

Demontagetechnik befasst sich mit der systematischen Zerlegung technischer Erzeugnisse in ihre Einzelteile, um eine Rückführung in den Material- oder Produktkreislauf zu ermöglichen. Sie ist die technologische Basis für die Kreislaufwirtschaft, indem sie Prozesse zur Wiederverwendung (engl. Re-Use) und Verwertung (engl. Recycling) sowie zur Schadstoffentnahme bereitstellt. Die operativen Grundfunktionen orientieren sich an der Montagetechnik, wobei insbesondere das Identifizieren, Zerlegen sowie das Handhaben und Sortieren von zerlegten oder teilzerlegten Komponenten im Vordergrund stehen. Ein wesentlicher Unterscheid zur Montage ist die Differenzierung in zerstörungsfreie und zerstörende Demontageverfahren, jeweils abhängig vom angestrebten Verwertungsziel. Eine besondere Herausforderung in der Demontagetechnik stellt die Unsicherheit über den Produktzustand dar, da Verschleiß oder Deformationen die Automatisierbarkeit im Vergleich zur Montage erheblich komplizieren. Daher erfordert die Demontagetechnik intelligente flexible Systeme und robuste Sensorik, um auf variierende Produkteigenschaften reagieren zu können. Darüber hinaus arbeitet die Demontagetechnik oftmals nur auf unvollständigen Modellen oder Informationen, denn nicht immer die ursprünglichen Konstruktionsdaten zugänglich sind. Ein weiterer Aufgabenbereich in diesem Feld ist die demontagegerechten Konstruktion (engl. Design for Disassembly), um bereits in der Entwicklungsphase das Produkt für das Ende seines Lebenszyklus zu optimieren [6].

2.1.3. Montage- und Demontageplanung

Montageplanung (engl. Assembly Planning AP) und Demontageplanung (engl. Disassembly Planning DP) sind zwei eng miteinander verwandte Wissensgebiete.

Montageplanung beschreibt die Aufgabe der Generierung eines Montageplans unter Berücksichtigung von Eigenschaften wie Geometrien, Abhängigkeiten zwischen Komponenten und verfügbaren Fertigungsressourcen [7]. Hierbei sind einige Eigenschaften zwingend zu berücksichtigen, etwa die Geometrien zur Sicherstellung von kollisionsfreien Montagevorgängen, während andere Faktoren primär für die Optimierung der Montagepläne relevant sind wie z.B. die Betrachtung von redundanten Fertigungsressourcen für die Reduktion der Montagezeit. Montageplanung ist der Problemklasse NP-schwer zuzuordnen und lässt sich hauptsächlich in drei Unterkategorien differenzieren: Montagereihenfolgenplanung (engl. Assembly Sequence Planning ASP), Montagelinienaufteilung (engl. Assembly Line Balancing ALB) und Montagepfadplanung (engl. Assembly Path Planning APP) [8].

Im weiteren Verlauf der Arbeit wird im Zusammenhang mit Montage und Demontage folgende Terminologie für zu montierende oder zerlegende Objekte verwendet:

- Bauteile: Atomare Komponente eines Produktes, z.B. Schrauben, Unterlegscheiben, Metallprofile. Bauteile sind einzigartig, daher werden sie stets nur an einer Position eingesetzt. Gleichwohl kann es mehrere Bauteile des gleichen Typs geben.
- Unterbaugruppe: Sie stellen eine optionale Zwischenunterteilung eines Produktes dar. Mehrere Bauteile bilden zusammen eine Unterbaugruppe, beispielsweise kann der Motor eines Autos als Unterbaugruppe des Autos betrachtet werden. Eine Unterbaugruppe kann dabei aus einer beliebigen Kombination von Unterbaugruppen und Bauteilen aufgebaut sein. So bilden Zylinderkopf und Kurbelgehäuse zwei Unterbaugruppen der Baugruppe Verbrennungsmotor. An Unterbaugruppen gilt die Anforderung, dass die darin vorkommenden Bauteil bzw. Unterbaugruppen in einem logischen Zusammenhang zueinanderstehen müssen.
- Baugruppe: Die Baugruppe bildet das Gesamtprodukt aller in ihr enthaltenen Bauteile. Ein Beispiel hierfür ist ein Fahrzeug.
- Komponenten: Komponenten werden in dieser Arbeit als Sammelbegriff für eine beliebige Mengen aus Bauteilen und Unterbaugruppen genutzt.

Montagereihenfolgenplanung

Bei der Montagereihenfolgenplanung (ASP) wird für eine gegebene Baugruppe eine Reihe von kollisionsfreien Operationen $o_1, ..., o_n$ gesucht, die zu einer Montage der Teile $p_1, ..., p_n$ führt. Das Ergebnis des ASP ist ein symbolischer Plan, dessen Granularität abhängig vom Einsatzszenario ist. So kann es in einem Fall sinnvoll sein, einen sehr abstrakten Plan zu generieren, welcher besonders Fügeoperationen als abstraktes Konstrukt betrachtet, beispielsweise *Schrauben*, in einem anderen Fall kann eine feinere Unterteilung notwendig sein. Dementsprechend kann *Schrauben* unterteilt werden in: Schraube aus Materiallager entnehmen, Schraube an Fügeposition ansetzen, mit Schraubtool Fügevorgang durchführen.

Im Kontext der vorliegenden Arbeit wird ASP in zwei Teilschritte unterteilt. Zunächst wird die Montagesequenz gesucht. Hierbei handelt es sich um die reine Abfolge, in welcher die Bauteile montiert werden müssen. Bei der Suche nach einer möglichen Montagesequenz wird die Annahme getroffen, dass zu der entsprechenden Reihenfolge korrekte Operationen existieren und den Schritten zugeordnet werden können. Diese Zuordnung erfolgt in einem zweiten Schritt der Montageplanverfeinerung. Sie hat das Ziel, die korrekten Fügeoperationen für jeden der einzelnen Sequenzschritte zu bestimmen. Die Granularität der Fügeoperationen ist dabei nicht festgelegt und kann szenarioabhängig gewählt werden.

Ein einfaches Szenario für die Anwendung von ASP ist in Abbildung 2.1 dargestellt. Die Montage einer Radaufhängung muss in diesem einfachen Szenario realisiert werden. Hier ergibt die Suche nach einer möglichen Montagesequenz,

(a) CAD-Rendering einer Baugruppe einer Radaufhängung.

(b) CAD Rendering der einzelenen Bauteile einer Radaufhängung, bestehend aus Federung (links), den beiden Rädern (unten) und der Achse (oben).

Abbildung 2.1.: Baugruppe einer Radaufhängung und ihre einzelnen Bauteile

dass zunächst die Aufhängung (p_1) mit der Achse (p_2) durch Aufstecken (o_1) verbunden wird, bevor die Räder (p_3 und p_4) durch Aufpressen (o_2 und o_3) mit der Achse verbunden werden.

ASP ist ein Kombinationsproblem und in die Problemklasse NP-schwer einzuordnen. Dieses liegt an dem faktoriellen Wachstums des Lösungsraums mit steigender Anzahl der zu montierenden Komponenten [9]. Hierbei ist die ausschlaggebende Problematik in der Suche einer gültigen Montagesequenz und nicht die Überführung in einen Montageplan. Das führt dazu, dass die einfachste Umsetzung des ASP, welche bei der Suche nach einer gültigen Montagesequenz alle Bauteilekombinationen prüft, eine Aufwand von $O(n!)$ hat.

Die geläufigste Darstellung für das ASP-Problem ist als gerichteter Graph $D = (P, C)$, wobei die Knoten P jeweils ein Bauteil der Baugruppe repräsentieren und die Kanten C eine Beziehung zwischen zwei Bauteilen repräsentiert. Bilden die Kanten eine Vorrangbeziehung ab, wird von einem Vorranggraph gesprochen. Mit diesem lassen sich mögliche Montagereihenfolgen darstellen. Da es bei komplexen Baugruppen mit vielen Bauteilen mehrere gültige Montagesequenzen in Hinblick auf die kollisionsfreie Ausführung und physikalische Machbarkeit gibt, werden zusätzliche Zielfunktionen definiert, die bei der Auswahl der korrekten Sequenz berücksichtigt werden [10]. Diese überführt das ASP-Problem in ein Optimierungsproblem. Häufige Zielfunktionen an denen die Optimierung durchgeführt wird sind: Minimierung der Ausführungszeit der Montage, Minimierung

der Kosten der Montage oder Minimierung der Anzahl der notwendigen Werkzeugwechsel. Die konkreten Zielfunktionen sind stets Szenario abhängig, wobei auch Kombinationen aus verschiedenen Zielfunktionen möglich sind. Ansätze für ASP sind primär in vier Unterkategorien aufteilbar:

- Traditionelle Ansätze: Diese umfassen graphenbasierte Methoden wie AND/OR-Graphen [11] und Liaison-Graphen [1] sowie matrixbasierte Methoden wie Assembly Incidence Matrices [12].

- Mathematische Ansätze: Für die Verarbeitung durch mathematische Ansätze, wird das ASP-Problem durch mathematische Modelle formuliert, hierbei werden Fügeoperationen als Variablen dargestellt und Vorrangbedingungen, sowie Ressourcenbeschränkungen als Nebenbedingungen umgesetzt [13]. Zur Lösung finden Ansätze wie Integer Programming [14], Mixed-Integer Linear Programming [15] oder Constraint Programming [16] Anwendung.

- Heuristik Ansätze: Da aufgrund der NP-schweren Problematik exakte rechnerische Lösungen oft nicht praktikabel sind, kommen häufig auch heuristische Methoden, wie Greedy-Algorithmen [17] oder lokale Suchmethoden wie Simulated Annealing [18] zum Einsatz.

- Soft Computing Ansätze: Durch steigernde Rechenleistung in den vergangenen Jahrzenten und Fortschritte im Bereich Maschinelles Lernen (ML) und KI sind verschiedenste Soft-Computering Ansätze entstanden. Diese verwenden approximierte Modelle und Unschärfe, um komplexe reale Probleme - im Vergleich zu traditionellen exakten Methoden - in kürzerer Zeit zu lösen und zeitgleich einen breiteren Suchraum zu explorieren [13]. Beispiele für diese Ansätze sind Genetische Algorithmen [19], Ant Colony Optimization [20] und Partikelschwarmoptimierung [2]. Weitere Ansätze, die zu dieser Kategorie gehören, werden werden im Stand der Technik 3 erläutert.

Montagelinienaufteilung

Die Montagelinienaufteilung (ALB) befasst sich mit der Aufteilung aller Fügeoperationen für eine Baugruppe in eine Menge von n atomaren Manipulationen und deren Zuweisung zu m Montagezellen / Montagestationen [21]. Das Ziel ist es, diese Aufteilung unter Berücksichtigung von Vorrangbeschränkungen zwischen den einzelnen Manipulationen und einer annährend gleichen Zeitdauer an jeder Montagestation zu finden. Auch beim ALB handelt es sich um ein Optimierungsproblem. Hierfür kommen häufig die gleichen Soft-Computing-Methoden zum Einsatz, z.B. genetischen Algorithmen [22].

Montagepfadplanung

Die Montagepfadplanung (APP) kann als Erweiterung der durch die Montagereihenfolgenplanung gefundenen Lösung aufgefasst werden, da indem für jeden Teilschritt des Montageplans eine kollisionsfreie Trajektorie für das Fügen der Bauteile $p_1, ..., p_n$ gefunden werden muss. Basierend auf dem Beispiel aus Abbildung 2.1, bedeutet dies im konkreten Fall, dass durch den ASP bestimmt wird, dass die Achse durch die Aufhängung gesteckt wird und anschließend die Räder aufgepresst werden. Anschließend definiert der Planer für das APP die kollisionsfreie Trajektorie für jede Operation. Für das Einsetzen der Achse würde der APP einen kollisionsfreien Pfad generieren, der definiert, wie sich die Achse von ihrem initialen Lagerort bewegt, sich der Aufhängung im korrekten Winkel und Position nähert und durch die zylindrische Öffnung geführt wird, bis sie ihre finale Montagekonfiguration erreicht. Häufig betrachtet APP für die Generierung der Trajektorien nicht nur die eigentliche Baugruppe, sondern auch die Umgebung / Montagzelle.

Ansätze für das APP sind vielfältig und lassen sich grob in die nachstehenden fünf Kategorien unterteilen:

- Graphenbasierte Ansätze: Sind ähnlich ausgelegt wie bei der Montagereihenfolgenplanung, zu berücksichtigende Bauteile, Umgebungsobjekte etc. und deren Beziehungen zueinander werden in einer Graphenstruktur repräsentiert. Beispiele sind Blocking Graphs [23], AND/OR Graphen [24], Visiblility Graphs [25] und Contact State Graphs [26].
- Rasterbasierte Ansätze: Unterteilen den Konfigurationsraum / Arbeitsraum in ein uniformes zwei- oder dreidimensionales reguläres Raster. Diesen diskreten Suchraum explorieren die Ansätze lokal, um geeignete Trajektorien zu finden. Beispiele hierfür sind A* [27], Potenzialfelder [28], Maximum Clearance Methods [29].
- Stichprobenbasierte Ansätze: Abstrahieren den Suchraum durch Erzeugen von zufälligen kollisionsfreien Konfigurationen, welche anschließend zu einem Graphen verbunden werden. Im Vergleich zu rasterbasierten Ansätzen arbeiten sie im kontinuierlichen Raum und sind so besonders in großen Arbeitsräumen recheneffizienter. Beispiele sind Ansätze wie Probalistic Roadmaps [30] und Rapidly-exploring Random Trees [31].
- Ansätze zur Raumzerlegung: Ansätze in diesem Bereich verwenden eine Divide-and-Conquer-Strategie indem der Arbeitsraum durch Unterteilung in kleinere Unterräume zerlegt wird, in welchen die Lösung gesucht wird. Eine Stärke dieser Ansätze stellt die hohe Parallelisierbarkeit dar. Beispiele für die Ansätze sind Swept Volumes [32] und Cell Decomposition [33].
- Interaktive Ansätze: Binden den Menschen durch Techniken in den Planungsprozess und der Verifikation von Pfaden ein wie z.B. wie Virtual-Reality oder haptische Schnittstellen [34].

Demontageplanung

Demontageplanung betrachtet viele deckungsgleiche Themen wie die Montageplanung. Unter der Voraussetzung der zerstörungsfreien Demontage, der Abwesenheit von Manipulationen die Baugruppe oder Bauteile beschädigen z.B durch Sägen oder Fräsen, können viele Montagemethoden invertiert auf die Demontage angewandt werden. Die Demontageplanung ist auch für die Montageplanung relevant, da für das Finden einer ersten möglichen Montagesequenz, welche danach optimiert wird, unter anderem das Montage durch Demontage (engl. Assembly by Disassembly) Prinzip [35] angwandt wird. Hierbei werden Bewegung und Vorrangstatus der Demontagezustände verwendet. Das führt dazu, dass der Lösungsraum für die Montagereihenfolgenplanung beschränkt werden kann.

2.1.4. Verbindungstechnik

Unter Verbindungstechnik werden Methoden für das Zusammensetzen von Baugruppen aus Bauteilen verstanden [36]. Fügen stellt einen zentralen Teil dieser Methoden dar und bezeichnet den Verbindungsvorgang der Komponenten (Bauteile, Unterbaugruppen) miteinander. Unterteilt werden die Verbindungsvorgänge nach den physikalischen Prinzipien Kraftschluss, Formschluss und Stoffschluss.

Kraftschluss

Kraftschlüssige Verbindungen verhindern durch Haftreibung die Bewegung der beteiligten Verbindungspartner. Wenn die Lastkraft die Haftreibung überschreitet, löst sich die Verbindung. Ein typisches Beispiel für diese Art von Verbindungen stellen Schraubverbindungen dar.

Formschluss

Ein Formschluss entsteht, wenn zwei Bauteile ineinandergreifen. Dann entsteht die Verbindung durch den Kontakt von zwei Wirkflächen. Eines der Bauteile verhindert somit die Bewegung des anderen in mindestens einer Richtung. Besonders bei Holz wird häufig Formschluss verwendet, etwa bei der Verbindung von Holzparkett durch die Nut-Feder-Verbindung.

Stoffschluss

Stoffschluss stellt eine Sonderverbindung dar. Bei ihr haften die Bauteile durch atomare oder molekulare Kräfte aneinander. Diese Verbindung erfordert das Hinzufügen einer zusätzlichen Komponente zur Verbindung, entweder in Form von Energie oder in Form von zusätzlichem Material. Die Verbindung ist im Vergleich

zu den vorangegangenen in der Regel nicht zerstörungsfrei zu lösen. Konkrete Techniken, die auf diesem Prinzip beruhen sind Klebe-, Löt- und Schweißverbindungen.

Fügeoperationen

Fertigungsverfahren sind nach der Norm DIN 8580 [37] in verschiedene Kategorie unterteilt. Dazu zählen:

- Urformen: Schaffen einer geometrischen Form aus formlosen Stoffen.
- Umformen: Gezielte Änderung der Form eines festen Körpers durch Deformation.
- Trennen: Gezielte Änderung der Form eines festen Körpers durch Materialabtrag.
- Fügen: Verbindung von zwei oder mehr Komponenten.
- Beschichten: Aufbringen einer fest haftenden Schicht eines formlosen Stoffes auf der Oberfläche eines festen Körpers.
- Stoffeigenschaften ändern: Änderung der Stoffeigenschaft eines festen Körpers durch z.B. Einbringen von Stoffteilchen.

Relevant für die vorliegende Arbeit ist insbesondere die Kategorie Fügen. In dieser Kategorie sind verschiedene Fügeoperationen enthalten. Sie werden nachstehend benannt:

Zusammensetzen:
Durch Schwerkraft (Reibung), Formschluss, Federkraft oder einer Kombination dieser Kräfte werden beim Zusammensetzen die Komponenten verbunden. Hierbei unterscheidet die Norm zwischen Auflegen, Aufsetzen, Einhängen, Einlegen, Einrenken, federnd Einspreizen, Ineinanderschieben und Schichten.

Füllen:
Fügen durch Füllen umfasst alle Prozesse, die dazu dienen, breiige, dampfförmige, flüssige, gasförmige, körnige, pastenförmige, pluverige Stoffe oder kleine Körper in hohle oder poröse Körper einzubringen. Ein Beispiel hierfür ist das Befüllen von Druckbehältern mit einem Gas.

Anpressen / Einpressen:
Anpressen bzw. Einpressen bezeichnet das Fügen durch elastisches Verformen. Das unbeabsichtigte Lösen dieser Verbindung wird durch einen Kraftschluss verhindert. Verfahren die unter diese Kategorie fallen sind: Klemmen, Klammern, Nageln, Schrauben und Verkeilen.

Fügen durch Urformen:
Für das Urformen wird mit formlosen Zusatzstoffen wie Pulver oder Schmelze

gearbeitet, welche die Verbindung zwischen den Komponenten herstellen. Beispiele für solche Verfahren sind: Ausgießen, Einbetten, Vergießen und Ummanteln.

Fügen durch Umformen:
Fügen durch Umformen bezeichnet die Verbindung von Komponenten entweder durch direkte Umformung oder durch Hilfsfügeteile, welche umgeformt werden. Verfahren in diesem Bereich sind: Biegen, Engen, Falzen, Kerben, Nieten, Spreizen und Weiten.

Fügen durch Schweißen:
Definiert wird Schweißen als stoffschlüssige Verbindung von artgleichen Werkstoffen, meist Metalle oder Thermoplasten, mit oder ohne Zugabe eines artgleichen Werkstoffs. Die Verbindungsfestigkeit beruht ausschließlich auf den Kohäsionskräften. Unterschieden wird primär zwischen Schmelzschweißverfahren und Pressschweißverfahren. Erstere Stellen die Verbindung durch das Erstarren der in schmelzflüssigen Zustand überführten Komponenten her. Beim Pressschweißen kommt kein Zusatzstoff zum Einsatz, die Komponenten werden durch große Kraft und meist unter Zusatz zusätzlicher Energie aufeinander gepresst.

Fügen durch Löten:
In der Norm ist Löten definiert als stoffschlüssige nicht lösbare Verbindung von in der Regel metallischen Werkstoffen mit Hilfe eines nicht artgleichen Zusatzmetalls. Beim Löten wird mit Temperaturen gearbeitet, die unter der Schmelztemperatur der Fügeteile liegt und nur das Zusatzmaterial verflüssigt.

Fügen durch Kleben:
Nach Norm ist Kleben definiert als das Fügen von Komponenten unter Verwendung eines Klebstoffes, d.h. eines nichtmetallischen Werkstoffes, der die Komponenten durch Flächenhaftung und innere Festigkeit verbindet. Hiermit ist Adhäsion und Kohäsion gemeint.

2.1.5. CAD und STEP Modelle

Computer Aided Desing (CAD) bezeichnet den Einsatz von Computertechnologie für die Erstellung, Analyse, Änderung und Optimierung einer Konstruktion. Zum Einsatz kommt CAD in verschiedensten Bereichen von der Architektur, über die Fertigung bis hin zur Medizintechnik und wird verwendet, um technische Modelle und Zeichnungen zu erstellen. Abbildung 2.2 zeigt das Rendering einer mit CAD konstruierten Baugruppe. Ein großer Vorteil von CAD ist das Darstellen der entworfenen Konstruktion in 3D (aus verschiedenen Richtungen) sowie die Möglichkeit des Simulierens von physikalischen Eigenschaften basierend auf Aspekten wie Elastizität oder Festigkeit der Objekte. CAD bietet die Chance die Entwicklungszeit gegenüber manuellen Methoden zu verkürzen und dient dadurch auch der Kostenersparnis. Der Einsatz von CAD trägt gleicherweise zur Fehlervermeidung im Konstruktionsprozess bei. Ein weiter Vorteil

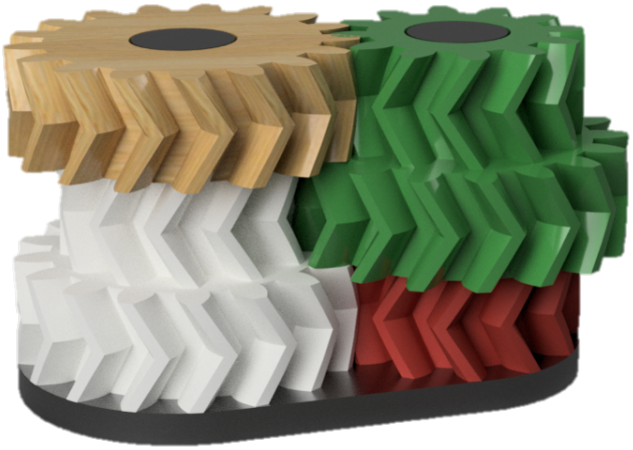


Abbildung 2.2.: Rendering einer exemplarischen Baugruppe bestehend aus mehreren Zahnrädern.

ist, dass alle relevanten Informationen in einer CAD-Datei hinterlegt und somit gebündelt sind. CAD-Daten dienen ferner als direkte Grundlage für weiterführende Prozesse, insbesondere für die computergestützte Fertigung (CAM) oder den 3D-Druck. In der Regel wird eine Konstruktion in einer CAD-Umgebung eines SW-Anbieters erstellt. Allerdings ist das in der Software erzeugte Datenformat nicht immer direkt lesbar und nach internen Standards des SW-Anbieters aufgebaut. Dadurch ist das Arbeiten mit den mit den CAD-Daten außerhalb der Software des entsprechenden Anbieters erschwert.

Als Lösung für diese Herausforderung existiert die ISO 10303 (Standard for Exchange of Product model data, kurz STEP) [38]. Diese Norm definiert einen plattformunabhängigen Formatstandard für CAD-Dateien. Durch die Überführung in eine STEP-Datei wird die Konstruktion in ein fest vorgegebenes Format gebracht, welches anschließend in CAD-/CAM-Software verschiedener Hersteller genutzt werden kann. Entsprechende Schnittstellen zum Einlesen bzw. Exportieren von STEP-Dateien sind daher bei fast allen Herstellern integriert. Die Norm gibt dabei nicht nur vor, wie die Geometrie der Konstruktion zu speichern ist, sondern auch welche möglichen weiteren produktbezogenen Daten hinterlegt werden können und in welcher Form. Durch den Aufbau als Klartextstruktur ist eine STEP-Datei einfach zu lesen und definierte Anwendungsprotokolle geben vor, welche genauen Daten in der STEP-Datei zu erwarten sind und wie diese strukturiert sind. Aktuell weit verbreitet ist das AP214 [39], welches Zugriff auf die Eigenschaften wie 3D-Geometrie, Farben, Layer, Annotationen, geometrische Dimensionen und Toleranzen ermöglicht. Neure Standards wie AP242 [40] bieten zusätzliche Möglichkeiten wie die umfassende Unterstützung von modellbasierten 3D-Konstruktionen, sind jedoch noch nicht so weit verbreitet. Aufgrund der Verbreitung und der ausreichenden Informationsdichte im AP214 basieren die Implementierungen der vorliegenden Arbeit auf diesen Voraussetzungen.

2.2. Machinelles Lernen

Maschinelles Lernen (ML) bezeichnet Methoden, die ein Aufgabenfeld durch den Einsatz von Lernalgorithmen auf Basis von Daten erlernen und mit diesen erworbenen Mustern Vorhersagen über neue / unbekannte Muster treffen können. [41].

Ansätze für das maschinelle Lernen lassen sich primär in 4 Kategorien einteilen:

- Überwachtes Lernen (engl. Supervised Learning): Hierbei lernt das Modell auf Basis von Trainingsdaten, die bereits gelabelt sind und daher das korrekte Ergebnis enthalten. Der Algorithmus wird auf dem korrekten Input-Output-Paar trainiert und lernt entsprechende Zusammenhänge. Während der Ausführung (Inferenz) kann das Ergebnis für einen neuen Input vorhergesagt werden. Aufgaben die mit solchen Ansätzen gelöst werden sind häufig Klassifizierungsaufgaben bzw. Regressionsaufgaben.

- Unüberwachtes Lernen (engl. Unsupervised Learning): Hierbei lernt der Algorithmus auf Basis von ungelabelten Trainingsdaten. Muster, Strukturen oder Zusammenhänge müssen selbstständig in den Daten erkannt und ggf. gruppiert werden. Aufgaben die mit solchen Ansätzen gelöst werden, sind häufig Clustering oder Anomalieerkennung.

- Teilüberwachtes Lernen (engl. Semi-Supervised Learning): Dieses Format stellt eine Mischform dar, bei welcher eine kleine Menge gelabelter Daten mit einer großen Menge ungelabelter Daten kombiniert wird. Besonders in Szenarien, in denen das Labeln der Daten besonders aufwendig oder teuer ist, etwa wenn dieses händisch durch Menschen passieren muss, sind solche Ansätze nützlich. Anwendung finden derartige Verfahren bei der medizinischen Bildanalyse, in denen nur eine kleine Anzahl von Röntgenbildern von Ärzten befundet werden aber eine große Masse an unbefundeten Bildern verfügbar ist.

- Reinforcement Learning (RL): Ansätze in dieser Kategorien basieren auf dem Prinzip von Versuch und Irrtum. Ein Agent interagiert mit einer Umgebung und erhält auf der Basis seiner Handlungen positives oder negatives Feedback. Ziel des Agenten ist es das positive Feedback zu maximieren. Auf diese Weise lernt der Agent das optimale Verhalten für die entsprechende Situation. RL Ansätze finden in den verschiedensten Bereichen Anwendung, beispielsweise bei der Reglung von Laufverhalten für mehrbeinige Robotersysteme [42].

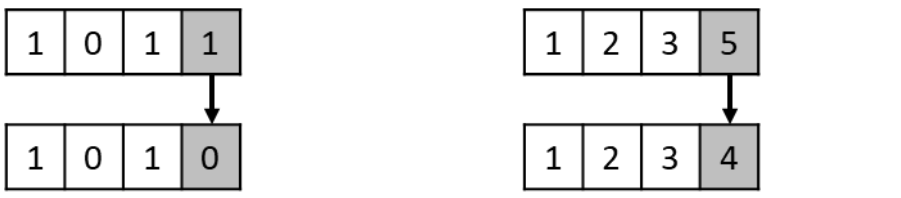
Anzumerken ist, dass sich nicht alle Verfahren direkt in die vier Kategorien einordnen lassen, da die Übergänge fließend sind.

2.2.1. Genetische Algorithmen

Das Konzept der genetischen Algorithmen (GA) ist primär den evolutionären Algorithmen zuzuordnen und an das Prinzip der genetischen Vererbung in der Biologie angelehnt. Durch die Entwicklung über mehrere Generationen hinweg werden Fähigkeiten und Eigenschaften von einzelnen Individuen, die sich als gut erweisen, an die nachfolgenden Generationen weitergegeben und optimiert.

Bei einem GA repräsentiert ein Individuum einen einzelnen, konkreten Lösungskandidaten. Innerhalb des Individuums stellt ein Gen den kleinsten Baustein dar, der für einzelne Paramater oder spezifische Eigenschaften der Lösung steht. Die Population fasst die Gesamtheit aller Individuen zusammen, die zu einem spezifischen Zeitpunkt existieren und miteinander verglichen werden. Als Generation werden in diesem Zusammenhang die einzelnen Iterationsschritte bezeichnet. In diesem Kontext wird die aktuelle Population bewertet und in eine nachfolgende Population überführt. Zentrale Mechaniken bei einem GA sind der Aufbau der Population, die Fitnessbewertung der einzelnen Individuen und die Kombination von guten Individuen für die Folgepopulation.

Für die Besetzung der initialen Population wird in einem GA gewöhnlich die einzelnen Individuen zufällig generiert. Dadurch kann der Lösungsraum großflächig abgedeckt werden. Für die Folgepopulationen wird basierend auf den Fitnesswerten der aktuellen Population eine Auswahl an geeigneten Elternkandidaten getroffen. Die Auswahl kann dabei nach verschiedenen Prinzipien erfolgen erfolgen. Auf diesem Weg ist eine Selektierung der Individuen mit den höchsten Fitnesswerten, eine nach Fitness gewichtete Zufallsauswahl oder die Auswahl von Repräsentanten von Untergruppen der Population möglich [43]. Um eine Weiterentwicklung der Folgepopulation zu erreichen, werden die selektierten Individuen adaptiert oder miteinander kombiniert, um neue Individuen zu erzeugen. Für die Adaption wird in den meisten Fällen der Mutations-Operator verwendet. Er modifiziert ein Individuum, indem er ein zufälliges Gen verändert wird. Wird eine Genrepräsentation durch Bits gewählt, so kann diese durch ein Bit-Flip erreicht werden. Bei einer anderen Repräsentation, z.B. einer Alphabets-basierten Darstellung, wird ein zufälliges zulässiges Zeichen aus dem Lösungsraum gewählt. Eine Veranschaulichung des Vorgehens ist in Abbildung 2.3 dargestellt.



(a) Mutation mit Hilfe einer Bit-Umschaltung.

(b) Mutation durch Einfügen eines zufälligen Zeichens.

Abbildung 2.3.: Schema von Mutations Operationen

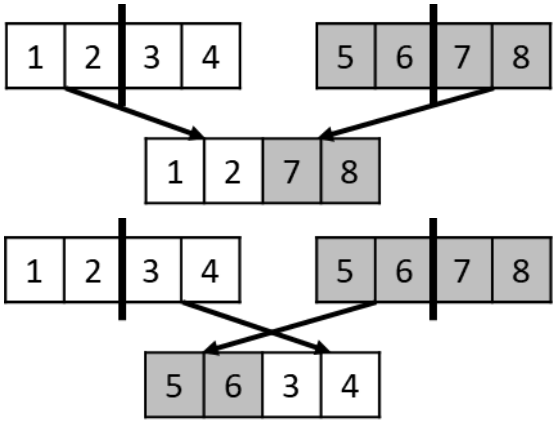


Abbildung 2.4.: Single-Point-Crossover-Operator: Ein Index wird gewählt, der die Genome in zwei Teilbereiche unterteilt. Diese werden mit dem jeweils ergänzenden Bereich des anderen Genom kombiniert um zwei neue Nachkommen zu zeugen.

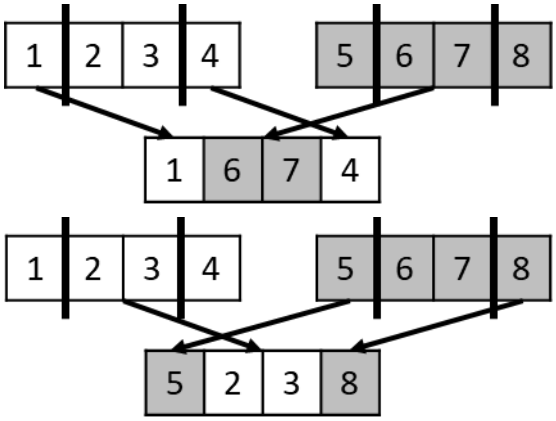


Abbildung 2.5.: Two-Point-Crossover-Operator: Ein Minimum und ein Maximum Index werden ermittelt. Alle Zeichen zwischen diesen Indices werden zwischen den Elterngenomen ausgetauscht.

Bei der Kombination von zwei Individuen findet häufig der Crossover-Operator Anwendung. Dieser kombiniert zwei Elternindividuen, indem eine zufällig ausgewählte Teilsequenz der Lösung eines Elternteiles in das andere Elternteil kopiert wird. Für diesen Vorgang gibt es sowohl die Möglichkeit die Single-point-crossover (vgl. Abbildung 2.4) oder die Two-point-crossover Methodik (vgl. Abbildung 2.5) zu nutzen. der ersten Variante wird eine einzelne Trennstelle im Gencode gewählt, anschließend findet eine kreuzweise Kombination statt und erzeugt zwei unterschiedliche Nachkommen. Beim Two-point-crossover werden zwei Stellen im Genom gewählt. Diese begrenzen den auszutauschenden Bereich. Die einzelnen Teilabschnitte werden anschließend ebenfalls kreuzweise in das jeweils andere Elternteil eingefügt.

Der Ablauf der Fitnessbewertung, Elternauswahl und Generierung einer neuen

Generation wiederholt sich kontinuierlich und zwar so lange, bis ein Abbruchkriterium erreicht wird. Dieses Abbruchkriterium ist frei wählbar. Gängige Methoden beenden den Prozess nachdem die Fitness einen Schwellwert überschreitet oder es zu einer Stagnation der Fitness kommt.

2.2.2. Neuronale Netze

Künstliche neuronale Netze stellen etablierte Werkzeuge zur Entwicklung und Unterstützung lernender Systeme in zahlreichen Anwendungen dar. Insbesondere in der Bildverarbeitung haben sie bedeutende Fortschritte bei Klassifikations- und Segmentierungsaufgaben erzielt. Aus mathematischer Sicht bestehen sie aus Kompositionen von Funktionen. Jeder Funktion wird eine Schicht zugewiesen, wobei jede Schicht eine definierbare Anzahl an Parametern besitzt, die als Vektoren angeordnet sind. Jedes Element eines solchen Vektors wird als Neuron bezeichnet. Insgesamt definiert die Anzahl der Schichten die Tiefe und die Anzahl der Neuronen pro Schicht die Breite des Netzes.

Eine für diese Arbeit relevante Kategorie sind die sogenannten vorwärtsgerichteten neuronalen Netze (engl. Feedforward-Netze), da hier die Parameter einer Schicht ausschließlich von den Ausgaben der vorherigen Schicht abhängen. Es finden keine Rückkopplungen zu vorangegangenen Schichten statt. In vorwärtsgerichteten neuronalen Netze erhält ein Neuron eine Anzahl gewichteter Eingaben aus der Vorgängerschicht. Der Ausgang des Neurons berechnet sich aus der Summe dieser gewichteten Eingänge und einem Bias-Term (Schwellenwert). Dieser bietet die Möglichkeit, das Verhalten des Neurons unabhängig vom Input zu verschieben. Für die nachfolgenden Neuronen stellt der Ausgang dieses Neurons wiederum einen möglichen Input dar.

Um den Ausgang eines Neurons auf einen bestimmten Wertebereich - häufig das Intervall $[0, 1]$ - zu beschränken und nichtlineare Zusammenhänge modellieren zu können, werden auf die gewichteten Summen Aktivierungsfunktionen angewandt. Häufig eingesetzt werden hierfür die lineare Funktion, die Sigmoid-Funktion und die Rectified Linear Unit (ReLU). Eine Darstellung dieser Funktionen ist in Abbildung 2.6 zu sehen. Die lineare Aktivierungsfunktion findet vor allem bei der letzten Schicht eines Regressionsproblems Anwendung, da dort der gesamte Wertebereich abgebildet werden soll. Sigmoid und Softmax werden bevorzugt bei Klassifizierungsproblemen verwendet, da sie Wahrscheinlichkeiten für die zugehörigen Klassen ausgeben. Bei binärer Klassifikation findet vor allem die Sigmoid-Funktion Anwendung, bei mehr als zwei Klassen ist Softmax die bevorzugte Wahl. In den verdeckten Schichten (engl. Hidden Layers) wird häufig die nichtlineare ReLU-Funktion eingesetzt. Diese unterbindet negative Werte (setzt sie auf Null), verhält sich bei positiven Werten jedoch linear, was das Training tiefer Netze erleichtert. Für alle genannten Aktivierungsfunktionen ist es möglich, einen Gradienten zu definieren, was für das Training mittels Backpropagation essenziell ist.

Name	Funktion
Linear	$f(x) = x$
Sigmoid	$f(x) = \frac{e^x}{e^x+1}$
Softmax	$f(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$
ReLU	$f(x) = \max(0, x)$

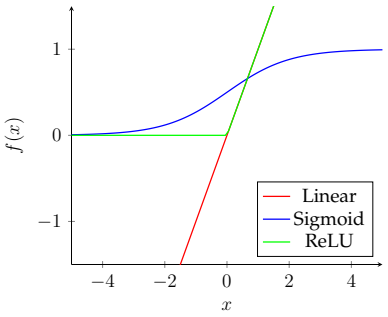


Abbildung 2.6.: Aktivierungsfunktionen: Linear, Sigmoid, Softmax und ReLU.

Fehlerfunktionen

Für das Training eines neuronalen Netzes müssen die Gewichte der Verbindungen iterativ angepasst werden, um den Fehler zu minimieren. Die geläufigste Methode hierfür ist die Backpropagation [41]. Diese nutzt den Gradienten der Fehlerfunktion bezüglich der Gewichte und berechnet, ausgehend von der Ausgabeschicht rückwärts durch das Netz, die notwendigen Anpassungen. Je näher eine Schicht an der Eingabeschicht liegt, desto kleiner kann der Effekt des Fehlersignals auf die Gewichte werden (Vanishing Gradient Problem). Für die Definition der Fehlerfunktion (engl. Loss Function) gibt es verschiedene Möglichkeiten, die je nach Problemstellung gewählt werden. Im Rahmen dieser Arbeit sind besonders Klassifizierungs- und Regressionsprobleme relevant. Bei überwachten Lernansätzen erfolgt die Berechnung über den numerischen Vergleich der Soll-Werte (Ground Truth) und der Ist-Werte (Prädiktion). Gängig sind vor allem folgende Funktionen:

- **L1-Verlust (Mean Absolute Error):** Für Regressionsprobleme ist der L1-Verlust [44] (oft auch als L1-Loss bezeichnet) eine häufig genutzte Fehlerfunktion. Ausgehend von n vorhergesagten Werten und n Soll-Werten berechnet die Funktion den Betrag der Differenz zwischen den einzelnen Wertepaaren. Diese Differenzbeträge werden aufsummiert und durch die Anzahl n geteilt. Formal ergibt sich für n Soll-Werte y_i und Vorhersagen y_i^p der mittlere absolute Fehler (*Mean Absolute Error*, MAE) nach folgender Gleichung:

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - y_i^p|}{n} \tag{2.1}$$

- **Kreuzentropie:** Die Kreuzentropie (engl. *Cross Entropy*) findet häufig bei Klassifizierungsproblemen Anwendung. Als Maß für die Abweichung zwischen zwei Wahrscheinlichkeitsverteilungen P (Soll) und Q (Ist) einer Zu-

fallsvariablen X (mit Zielmenge Ω) berechnet sie sich wie folgt:

$$H(P,Q) = -\sum_{x\in\Omega} P(X=x)\cdot\log(Q(X=x)) \tag{2.2}$$

Im Kontext des maschinellen Lernens muss dies adaptiert werden, um die Verteilung der Soll-Werte mit der Verteilung der generierten Ist-Werte zu vergleichen. Bei einer binären Klassifikation wird hierfür die binäre Kreuzentropie (*Binary Cross Entropy*, L_{bce}) [45] verwendet, welche sich für M Trainingsdatenpunkte mit dem m-ten Soll-Wert y_m und dem m-ten Ist-Wert y_m^p wie folgt berechnet:

$$L_{bce} = -\frac{1}{M}\cdot\sum_{m=1}^{M}\left[y_m\cdot\log(y_m^p) + (1-y_m)\cdot\log(1-y_m^p)\right] \tag{2.3}$$

In vielen Klassifizierungsproblemen existieren jedoch mehr als zwei Klassen. Hierfür ist eine Erweiterung auf die kategorische Kreuzentropie (*Categorical Cross Entropy*, L_{cce}) [45] notwendig:

$$L_{cce} = -\frac{1}{M}\cdot\sum_{k=1}^{K}\sum_{m=1}^{M} y_{k,m}\cdot\log(y_{k,m}^p) \tag{2.4}$$

Hierbei ist K die Anzahl der Klassen, $y_{k,m}$ der m-te Soll-Wert der k-ten Klasse (meist One-Hot-kodiert) und $y_{k,m}^p$ die vorhergesagte Wahrscheinlichkeit für die k-te Klasse des m-ten Datensatzes.

- **Focal-Loss:** Ein Nachteil der Kreuzentropie ist die implizite Annahme, dass alle Klassen gleich gewichtet oder gleich schwer zu erlernen sind. In realen Szenarien (z. B. bei starkem Klassenungleichgewicht) ist diese Annahme oft nicht haltbar. Um dem entgegenzuwirken, werden einfach zu klassifizierende Beispiele in der Fehlerfunktion weniger stark gewichtet, sodass sich das Training auf die „schwierigen" Fälle konzentriert. Dies erfordert eine Adaption der Kreuzentropie und ist für den binären Focal-Loss (L_{bfl}) [46] wie folgt definiert:

$$L_{bfl} = -\frac{1}{M}\cdot\sum_{m=1}^{M}\left[(1-y_m^p)^\gamma\cdot y_m\cdot\log(y_m^p) + (y_m^p)^\gamma\cdot(1-y_m)\cdot\log(1-y_m^p)\right] \tag{2.5}$$

Durch den zusätzlichen Faktor $(1-y_m^p)^\gamma$ bzw. $(y_m^p)^\gamma$ wird der Beitrag zum Fehler reduziert, wenn die Vorhersage bereits sehr sicher (nahe 0 oder 1) ist. Für den Fokussierungsparameter hat sich in der Literatur der Wert $\gamma = 2$ etabliert. Auch für die kategorische Kreuzentropie existiert eine entsprechende Erweiterung zum Focal-Loss [46].

Schichten in neuronalen Netzen

Neben einer passend gewählten Fehlerfunktion ist auch die Architektur, also die Verkettung der Schichten, ausschlaggebend für die Leistungsfähigkeit eines neuronalen Netzes. Gängig sind hierfür zwei Varianten: vollständig verbundene Schichten (engl. *Dense Layers* oder *Fully Connected Layers*) und Faltungsschichten (engl. *Convolutional Layers*).

Vollständig verbundene Schichten zeichnen sich dadurch aus, dass jedes Neuron mit allen Neuronen der vorherigen Schicht verbunden ist. Hierdurch können im Lernprozess Zusammenhänge zwischen beliebigen Merkmalen (engl. *Features*) hergestellt werden. Zur Definition sind lediglich die Anzahl der Neuronen sowie die Aktivierungsfunktion erforderlich. In Szenarien mit hochdimensionalen Eingabedaten, wie etwa bei der Verarbeitung hochaufgelöster Bilder, hat dieses Konzept jedoch Nachteile: Die Anzahl der Parameter explodiert, was das Training verlangsamt und die Gefahr von Overfitting erhöht.

In der Bildverarbeitung werden daher vor allem Faltungsschichten [47] eingesetzt. Sie besitzen deutlich weniger Parameter, da sie das Prinzip der lokalen Konnektivität und der gemeinsamen Gewichtsnutzung (engl. Weight Sharing) verfolgen. Für das semantische Verständnis von Bildern sind primär Informationen aus der lokalen Nachbarschaft eines Pixels relevant. Dies wird durch eine Filterstruktur realisiert, die mathematisch einer diskreten Faltung entspricht. Für ein Pixel $I(y, x)$ und einen quadratischen Faltungskern (Kernel) k der Größe $n \times n$ (mit ungeradem n und Zentrum a) berechnet sich der gefaltete Wert $I^*(y, x)$ wie folgt:

$$I^*(y, x) = \sum_{i=1}^{n} \sum_{j=1}^{n} I(y - i + a, x - j + a) \cdot k(i, j) \tag{2.6}$$

Das Ergebnis dieser Operation wird als Merkmalskarte (engl. *Feature Map*) bezeichnet und repräsentiert extrahierte Muster des Eingangsbildes, wie z. B. Kanten oder Texturen. Neben der Effizienz ist die Translationsinvarianz ein entscheidender Vorteil gegenüber vollständig verbundenen Schichten. In der Bildverarbeitung werden primär drei Arten der Faltung unterschieden:

- **Standardfaltung (engl. Standard Convolution)** [48]: Hierbei gleitet der Filterkern schrittweise über die Eingabedaten und berechnet durch Multiplikation und Addition lokale Merkmale. Oft wird dabei die räumliche Dimension durch eine Schrittweite (engl. Stride) > 1 verringert.
- **Transponierte Faltung (engl. Transposed Convolution)** [48]: Sie dient primär der Hochskalierung der Eingabedaten, etwa in Decodern von Segmentierungsnetzwerken. Dabei werden die Werte der Eingabe auf einen größeren Ausgabebereich projiziert, wodurch die räumliche Auflösung erhöht wird.

- **Dilatierte Faltung (engl. Dilated Convolution)** [48]: Der hierbei eingesetzte Filterkern lässt gezielt Lücken zwischen seinen Einträgen (Dilationsrate > 1). Dadurch wird das rezeptive Feld - der Bereich des Eingabebildes, den das Neuron „sieht" - vergrößert, ohne die Anzahl der Parameter oder die Rechenlast signifikant zu erhöhen.

Merkmalsextraktoren / Basisnetze (Backbones)

Ein in der Bildverarbeitung notwendiger erster Schritt ist die Extraktion relevanter Merkmale aus den sensorischen Eingangsbildern bzw. Bildfolgen. Hierfür werden meist explizite Subnetze verwendet, die als Merkmalsextraktoren (engl. *Feature Extractors*) oder Basisnetze (engl. *Basisnetz / Backbones*) bezeichnet werden. Sie bilden den initialen Teil der gesamten Netzarchitektur und transformieren die rohen Bilddaten in abstrakte Merkmalsvektoren, die anschließend für Aufgaben wie Klassifikation und Regression weiterverarbeitet werden.

Ein etabliertes Basisnetz stellt das Residual Neural Network (ResNet) [49] dar. Im Vergleich zu vorherigen Architekturen führt das ResNet sogenannte Identitätsverbindungen (engl. *Identity Shortcuts*) ein, welche die traditionelle Kette von Faltungsschichten überbrücken. Definiert werden kann dies für eine Eingabe X und die Abbildung $F(X)$, welche eine beliebige Anzahl an Schichten zwischen Eingabe und Ausgabe darstellt, als:

$$Y = F(X) + X.$$

Bekannt ist diese Struktur auch als Residualblock (engl. *Residual Block*). Sie ermöglicht es, bereits gelernte Merkmale ungefiltert an tiefere Schichten weiterzugeben. Hierdurch wird das Training tieferer Netze vereinfacht, da nachfolgende Schichten redundante Informationen nicht neu lernen müssen (Vermeidung des *Vanishing Gradient*-Problems).

Architektonisch besteht das ResNet aus einer Abfolge dieser Residualblöcke, die sich aus einer 1×1-Faltungsschicht, gefolgt von einer 3×3-Faltungsschicht und einer erneuten 1×1-Faltungsschicht sowie ReLU-Aktivierungsfunktionen zusammensetzen. Der Verarbeitungsprozess erfolgt dabei in fünf Stufen. Zunächst wird die räumliche Auflösung der Merkmalskarten sukzessive halbiert, während ihre Anzahl von 64 auf 2048 steigt. Hierdurch werden die Differenzierbarkeit und Aussagekraft der Merkmale gesteigert. In der Praxis kommen Erweiterungen des ResNet zum Einsatz. Zwei der am häufigsten verwendeten sind das `ResNet50` und das `ResNet101`. Sie unterscheiden sich vom Standard-ResNet und untereinander primär durch die Anzahl der Residualblöcke in der vierten Verarbeitungsstufe. Tabelle 2.1 stellt die Varianten gegenüber.

Eine Alternative zum ResNet stellt das DenseNet [50] dar. Dieses Verfahren kann die Anzahl der Merkmalskarten, welche eine neue Schicht hinzufügen muss, reduzieren, da in sogenannten Dense-Blöcken die Merkmalskarten aller vorange-

Auflösungs-stufe	**Ausgabe-dimension**	**ResNet50** (50 Schichten)	**ResNet101** (101 Schichten)
C1	256×256	7×7, 64, Stride 2	
C2	128×128	3×3 Max-Pooling, Stride 2	
		$\begin{pmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{pmatrix} \times 3$	
C3	64×64	$\begin{pmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{pmatrix} \times 4$	
C4	32×32	$\begin{pmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{pmatrix} \times 6$	$\begin{pmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{pmatrix} \times 23$
C5	16×16	$\begin{pmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{pmatrix} \times 3$	

Tabelle 2.1.: Aufbau der ResNet-Architektur für eine Eingabedimension von 512×512. Die Schichten sind Faltungsschichten (Conv-Layer), sofern nicht anders benannt.

gangenen Schichten konkateniert werden und so den späteren Schichten direkt zur Verfügung stehen.

Im Kontext von Basisnetzen wird häufig auch die Thematik des Transfer-Learnings behandelt. Eine der Herausforderungen besteht darin, dass für viele Problemstellungen kein ausreichend großer Datensatz zur Verfügung steht, der es ermöglicht, die Netze von Grund auf zu trainieren. Zur Lösung wird auf Basisnetze zurückgegriffen, die auf umfangreichen Datensätzen wie ImageNet [51] vortrainiert wurden. Dies geschieht unter der Annahme, dass elementare Merkmale in Bildern universeller Natur sind und daher auch für die aktuelle Problemstellung Relevanz besitzen. Basierend auf den vortrainierten Gewichten kann der Merkmalsextraktor mit einem deutlich kleineren Datensatz für die spezifische Aufgabe nachtrainiert (feinjustiert) werden, wodurch relevante Merkmale schneller und effizienter erlernt werden.

2.2.3. Segmentierung

Die semantische Segmentierung stellt eine Erweiterung der Bildklassifikation dar. Ziel ist es, jedem Pixel im Eingangsbild eine konkrete Klasse zuzuordnen, um semantisch zusammengehörige Bildbereiche zu erkennen. Die semantische Segmentierung gliedert sich methodisch in drei Phasen:

1. **Herunterskalierung (engl. Downsampling):** Extraktion und Komprimierung von Merkmalen.

2. **Hochskalierung (engl. Upsampling):** Wiederherstellung der ursprünglichen räumlichen Auflösung.

3. **Pixelweise Klassifikation:** Anwendung einer Softmax-Aktivierungsfunktion auf der Ausgabeschicht.

Da während des Downsamplings räumliche Informationen verloren gehen, existieren verschiedene Ansätze, um diese wiederherzustellen. Weit verbreitet sind Sprungverbindungen (engl. *Skip Connections*) [52]. Sie leiten Merkmalskarten aus früheren Schichten des Encoders direkt an die korrespondierenden Schichten des Decoders weiter, wo sie addiert oder konkateniert werden. Dem Netzwerk wird damit ermöglicht, sowohl lokale Details als auch den globalen Kontext zu berücksichtigen. Dies hat einen positiven Effekt auf die Segmentierungsqualität sowohl kleiner als auch großer Objekte.

Für die semantische Segmentierung werden häufig Architekturen nach dem Encoder-Decoder-Prinzip eingesetzt. Der Encoder, oftmals umgesetzt als Basisnetz wie das ResNet, reduziert zunächst die Dimensionen und extrahiert abstrakte Merkmale, während der Decoder diese Merkmale wieder auf die ursprüngliche Bildgröße hochskaliert. Eine bekannte Implementierung dieser Architektur ist das SegNet [53]. Es zeichnet sich durch einen symmetrischen Aufbau aus. Die im Encoder ermittelten Pooling-Indizes werden genutzt, um im Decoder eine präzise Rekonstruktion der räumlichen Struktur durchzuführen.

2.2.4. Objekterkennung

Im Bereich der Robotik und Automatisierung ist die maschinelle Objekterkennung ein zentraler Aspekt, da sie einem System ermöglicht, autonom mit seiner Umwelt zu interagieren. Charakteristisch für die Objekterkennung sind zwei Teilprobleme: einerseits die Klassifizierung der Objekte im Bild, andererseits deren Lokalisierung. Meist ist gefordert, dass Ansätze mehrere Objekte in einem Bild erkennen müssen, wobei diese auch aus unterschiedlichen Objektklassen stammen können. Die Objektklassifizierung ist ein Standard-Klassifizierungsproblem, während sich die Objektlokalisierung als Regressionsproblem definieren lässt. In der Regel werden die Lokalisierungen durch Begrenzungsrahmen / Bounding Box (engl. *Bounding Boxes*) gekennzeichnet, die um die entsprechenden Objekte im Bild gelegt werden. Diese sind meist achsenparallel, sodass sie nur aus horizontalen und vertikalen Kanten bestehen.

Auch bei der Objekterkennung kommen Faltungsnetze (engl. Convolutional Neural Networks, CNN) zum Einsatz. Sie extrahieren zunächst die für die zu erkennenden Objekte relevanten Merkmale. Die erzeugten Merkmalskarten werden anschließend in Subbereiche unterteilt, auf denen die Objektlokalisierung ausgeführt wird. Das Netz berechnet für alle Subbereiche des Bildes die Wahrscheinlichkeit für das Auftreten eines Objektes. Des Weiteren ermittelt das Netz für je-

den Subbereich und jede Objektklasse die Wahrscheinlichkeit, dass der Bereich ein Objekt dieser spezifischen Klasse enthält.

Basierend auf dieser grundlegenden Funktionsweise haben sich historisch zwei dominierende Architektur-Prinzipien etabliert, die sich im Kompromiss zwischen Genauigkeit und Inferenzgeschwindigkeit unterscheiden: zweistufige und einstufige Detektoren.

Zweistufige Detektoren, die meist aus der Familie der regionsbasierten Faltungsnetze (Region-based Convolutional Neural Networks, R-CNN) [54] stammen, trennen das Problem explizit auf. Ein erstes Netz, das sogenannte *Region Proposal Network* (RPN), generiert eine Menge potenziell relevanter Bereiche (Regionsvorschläge), die mit hoher Wahrscheinlichkeit ein Objekt enthalten – unabhängig von dessen Klasse. Ein zweites Netz bewertet anschließend diese spezifischen Ausschnitte, klassifiziert sie und verfeinert die Koordinaten der Bounding Boxes. Dieser Ansatz erzielt häufig die höchste Erkennungsgenauigkeit, ist jedoch aufgrund des seriellen Aufbaus mit längeren Laufzeiten verbunden.

Im Gegensatz dazu verzichten einstufige Detektoren, wie die YOLO-Architektur [55], auf den separaten Schritt der Regionsvorschläge. Stattdessen wird das Problem als ein einziges Regressionsproblem betrachtet, bei dem das Eingabebild typischerweise in ein festes Raster unterteilt wird. Für jede Rasterzelle prädiziert das Netz Bounding Boxes und Klassenzugehörigkeiten. Um die Varianz verschiedener Objektformen besser abzubilden, greifen viele Netze auf Ankerboxen (engl. *Anchor Boxes*) zurück. Dies sind vordefinierte Referenzrahmen mit unterschiedlichen Seitenverhältnissen und Skalierungen. Das Netz lernt dabei nicht die absoluten Koordinaten, sondern lediglich die notwendigen Verschiebungen (Offsets) und Skalierungen, um eine Ankerbox / Anchor Box optimal an das tatsächliche Objekt anzupassen.

2.2.5. Large Language Models

Sprachmodellierung (engl. *Language Modeling*, LM) hat zum Ziel, das Verständnis von natürlicher Sprache durch Computer zu verbessern, indem die Wahrscheinlichkeit möglicher Wortfolgen bestimmt wird. Sprachmodellierung stellt damit eine der zentralen Komponenten der maschinellen Verarbeitung natürlicher Sprache (engl. *Natural Language Processing*, NLP) dar. Dieses Forschungsfeld fokussiert sich darauf, Computer zu befähigen, menschliche Sprache zu verstehen, zu interpretieren und zu generieren. Sprachmodellierung differenziert sich in vier Bereiche [56]:

- **Statistische Sprachmodelle** (engl. *Statistical Language Models*, SLMs):
  Statistische Sprachmodelle sind in den 1990er Jahren entstanden. Sie verwenden ein n-Gramm-Modell basierend auf statistischen Methoden zur Bestimmung wahrscheinlicher Wortfolgen. n-Gramm-Modelle sind allerdings

besonders stark vom „Fluch der Dimensionalität“ (engl. *Curse of Dimensionality*) betroffen. Damit ist gemeint, dass die Anzahl der möglichen n-Gramm-Kombinationen exponentiell ansteigt, je größer die Sequenzlänge wird. Dies führt dazu, dass die meisten Wortsequenzen in einem realen Trainingsdatensatz nie vorkommen. Die Daten sind wenig aussagekräftig und das Modell hat Schwierigkeiten, zuverlässige Wahrscheinlichkeiten zu schätzen.

- **Neuronale Sprachmodelle** (engl. *Neural Language Models*, NLMs):
  Neuronale Sprachmodelle umgehen das Problem der statistischen Sprachmodelle, indem sie neuronale Netzwerke einsetzen, um die statistischen Muster und Wahrscheinlichkeiten von Wortsequenzen zu lernen. Durch die Darstellung von Wörtern in Vektoren (engl. Embeddings) können neuronale Sprachmodelle semantische Ähnlichkeiten und Abhängigkeiten im Kontext besser erfassen. Bei den neuronalen Sprachmodellen stellt sich allerdings die Herausforderung, dass sie einen deutlich höheren Bedarf an Rechen- und Datenressourcen haben und zum Halluzinieren neigen können.

- **Vortrainierte Sprachmodelle** (engl. *Pre-trained Language Models*, PLMs):
  Vortrainierte Sprachmodelle werden im Vergleich zu neuronalen Sprachmodellen auf allgemeinen Datensätzen vortrainiert (*pre-trained*) und dann ggf. für einen spezifischen Fall nachtrainiert (*fine-tuning* bzw. Feinabstimmung). Durch diesen Ansatz sind vortrainierte Sprachmodelle deutlich besser auf verschiedene Sprachaufgaben übertragbar und die Feinabstimmung erfordert eine beträchtlich kleinere Datenmenge im Vergleich zu neuronalen Sprachmodellen. Das Trainieren auf einem großen allgemeinen Datensatz setzt jedoch eine hohe Ressourcenkapazität voraus.

- **Large Language Models** (LLMs):
  Large Language Models gehören zu den neusten Entwicklungen auf diesem Gebiet. Sie beinhalten hunderte Milliarden von Parametern und können Aspekte wie das Lernen der Aufgabe während der Inferenz (*In-Context Learning*) und komplexe Argumentationen realisieren. Dadurch sind sie keine aufgabenspezifischen Systeme mehr, sondern können allgemeine Aufgaben lösen, ohne auf ein spezifisches Szenario nachtrainiert werden zu müssen.

Modelle, die auf einem breiten Datensatz trainiert werden, bevor eine aufgabenspezifische Adaption stattfindet, werden als *Foundation Models* bezeichnet. LLMs bilden eine große Subkategorie hiervon [57]. Auch *Vision Language Models* (VLMs) sind den Foundation Models als relevante Subkategorie zugeordnet. Im Folgenden werden Foundation Models als Synonym für ein vortrainiertes Netzwerk verwendet, unabhängig von der Zuordnung zu LLMs oder VLMs.

Aktuelle LLM-Architekturen basieren primär auf der Transformer-Architektur. Diese verwendet den *Attention*-Mechanismus anstelle von rekurrenten oder Faltungsmechanismen und ermöglicht eine deutlich stärkere Parallelisierung, sodass Milliarden von Parametern eingebunden werden können. Das Hauptmerk-

mal davon ist der *Self-Attention*-Mechanismus. Er berechnet kontextbezogene Darstellungen durch *Query-Key-Value*-Interaktionen, wodurch jedes Token alle Positionen in der Sequenz berücksichtigen kann. Auf diese Weise sind weitreichende Abhängigkeiten effektiv erfassbar bei gleichzeitig guter Recheneffizienz. Das ist eine wesentliche Voraussetzung für das Training von großen Sprachmodellen. Grundlegend kann zwischen drei Transformer-Architekturen unterschieden werden [58]:

- Encoder-Architekturen
- Decoder-Architekturen
- Encoder-Decoder-Architekturen

Abbildung 2.7 skizziert die drei verschiedenen Ansätze.

Reine Encoder-Modelle, wie das bekannte BERT-Modell [59], verarbeiten Texte bidirektional. Sie können maskierte Wörter vorhersagen und den Kontext aus beiden Richtungen, sowohl links als auch rechts vom maskierten Wort, simultan berücksichtigen. Durch gestapelte Transformer-Encoder-Layer, die den bidirektionalen *Self-Attention*-Mechanismus verwenden, erzeugen die Modelle kontextualisierte Repräsentationen des Inputtextes. Primär werden diese eingesetzt, wenn ein tiefgreifendes Verständnis des Inputs benötigt wird. Hierzu zählen Applikationen wie Textklassifikation sowie die Erkennung benannter Entitäten und extraktive Frage-Antwort-Systeme [60].

Reine Decoder-Modelle, wie die Modelle der GPT-Familie, verwenden einen autoregressiven Ansatz. Dieser bestimmt ein potenziell nächstes Token basierend auf allen bisher verwendeten Tokens in der aktuellen Sequenz, indem gestapelte Transformer-Decoder-Layer mit einem *Masked Self-Attention*-Mechanismus verwendet werden. Beim *Masked Self-Attention*-Mechanismus handelt es sich um ein Konzept, in dem jedes Token in einer Sequenz nur Informationen über vorangegangene Tokens erhält, sodass keine zukünftigen Tokens im Training oder

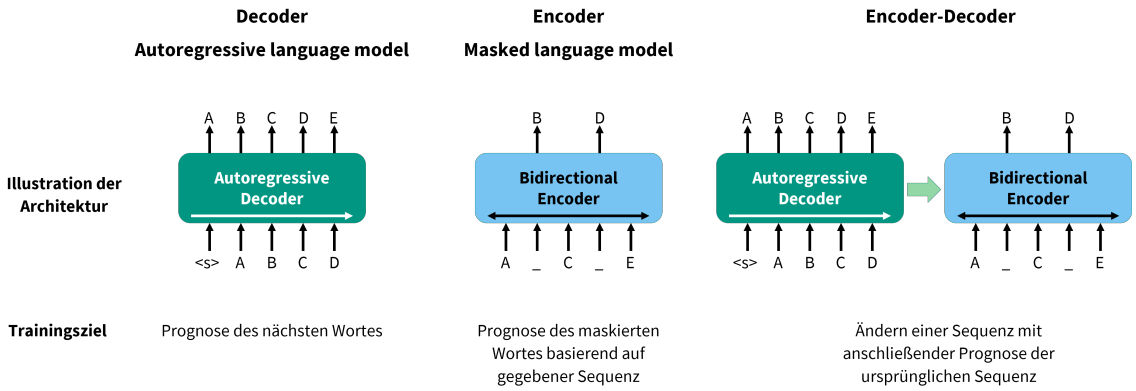


Abbildung 2.7.: Gegenüberstellung der drei grundlegenden Transformer-Architekturen (Encoder, Decoder, Encoder-Decoder).

der späteren Ausführung berücksichtigt werden. Diese Modelle eignen sich besonders für den Einsatz in Applikationen wie Textvervollständigung, kreatives Schreiben und Freitext-Antworten auf Fragen. Besonders neuere Modelle, die auf immer umfangreicheren Datensätzen trainiert werden, sind inzwischen durch Prompt / Prompt Engineering-basierte Ansätze anstelle von *Fine-Tuning* in der Lage, vielfältige allgemeine Fähigkeiten anzubieten [60].

Encoder-Decoder-Modelle wie BART [61] und T5 kombinieren die beiden Architekturen. Eine Eingangssequenz wird zunächst durch den Encoder verarbeitet und anschließend wird die Ausgabesequenz durch den Decoder erzeugt. Diese vielseitigen Modelle sind in der Regel für Applikationen der Sequenzrekonstruktion vortrainiert. Dazu kommen unterschiedliche Verfremdungstechniken (*corruption techniques*) zum Einsatz, beispielsweise *Sentence Permutation* oder *Text Infilling* [60]. Encoder-Decoder-Modelle sind dadurch besonders für Sequenz-zu-Sequenz-Anwendungen (*sequence-to-sequence*, Seq2Seq) geeignet. Bei ihnen kommt es sowohl auf ein Verständnis des Inputs an als auch auf eine Erzeugung von kohärenten Ausgaben. Beispiele hierfür sind Übersetzer, Textzusammenfassungen oder der Transfer auf einen anderen Schreibstil.

Um die Performance von Transformer-basierten LLMs zu bestimmen bzw. zu steigern, werden Skalierungsgesetze verwendet. Zwei Hauptansätze für die Schätzung der Leistungsskalierung finden dabei Verwendung:

- Das von Kaplan et al. [62] definierte Skalierungsgesetz etabliert eine Potenzgesetz-Beziehung (*engl. power-law*). Diese besagt, dass sich die Leistung verbessert, wenn Modellparameter, Trainingsdatenmenge und verfügbare Rechenleistung zunehmen. Es besteht die Tendenz, dass die Skalierung der Modellgröße der Skalierung der Datengröße vorzuziehen ist.

- Das Chinchilla-Skalierungsgesetz [63] baut auf dem Ansatz von Kaplan auf und verfeinert ihn. Demnach sollten für eine optimale Leistung sowohl die Modellgröße als auch die Trainingsdatenmenge in ungefähr gleichem Maße erhöht werden.

Neben der allgemeinen Leistungsverbesserung existieren auch Ansätze, ein LLM an ein aufgabenspezifisches Problem anzupassen und so dessen Performance zu steigern. Die beiden hauptsächlichen Strategien dazu sind *Prompting* und *Fine-Tuning*.

- **Prompting:** Unter einem Prompt wird die Eingabe an ein generatives KI-Modell verstanden. Sie hat das Ziel, die Ausgabe des Modells zu steuern. Prompting-Strategien nehmen keine direkte Anpassung an dem zugrundeliegenden Modell bzw. dessen Gewichten vor, sondern verfeinern die Eingabe für das Netzwerk. Es werden vor allem aufgabenspezifische Anpassungen am Input vorgenommen [64]. Eine genauere Aufschlüsselung von möglichen Prompting-Strategien ist in Abschnitt 2.2.7 einzusehen.

- **Fine-Tuning:** Im Vergleich zu Prompting verändert *Fine-Tuning* das Modell grundlegend, indem es sich auf die Anpassung der Gewichte des Netzwerks fokussiert. Beim *Fine-Tuning* wird das grundlegend trainierte Netzwerk mit einem kleinen aufgabenspezifischen Datensatz nachtrainiert. So können Muster und Merkmale des vortrainierten Modells auf eine neue Aufgabe übertragen werden, was zu einer besseren Performance bei dieser konkreten Aufgabe führt [65].

2.2.6. Vision Language Models

Während klassische LLMs allein einen textbasierten Input verarbeiten können, haben Entwicklungen in den letzten Jahren das Konzept der sprachbasierten Foundation Models erweitert, um auch weitere Eingabeinformationen zuzulassen. *Vision Language Models* (VLMs) sind eine dieser Erweiterungen, indem sie Bilder als zusätzlichen Input neben dem reinen Text zulassen. Die Ausgabe eines VLM ist weiterhin sprachbasiert.

Die drei zentralen Bestandteile der VLM-Architektur sind der *Vision Encoder*, der *Text Encoder* und der *Text Decoder*. Für eine Verarbeitung von einem visuellen Input wird der Vision Encoder verwendet, um das Bild in Einbettungsmerkmale (*Embedding Features*) zu projizieren. Der Text Encoder übernimmt die gleiche Aufgabe für den textuellen Input, indem er Textsequenzen in den Einbettungsraum (*Embedding Space*) projiziert. Final wird dann der Text Decoder verwendet, um aus den kodierten Repräsentationen einen entsprechenden textuellen Output zu generieren.

Bild-Encoder

Bild-Encoder (engl. *Vision Encoder*) werden in der Regel nach einem von zwei Prinzipien vortrainiert: durch multimodale Ansätze (z. B. Training auf Bild-Text-Paaren zur Erfassung von sprachlich-visuellen Zusammenhängen) oder unimodale Ansätze (z. B. das Training auf annotierten großen Datensätzen wie etwa ImageNet). Beispiele für unimodale Netzwerke sind ResNet [66] und Vision Transformers (ViTs) [67], während CLIP [68] zu den multimodal vortrainierten Ansätzen zählt. Aktuelle VLMs integrieren häufig vortrainierte Vision Encoder, anstatt sie von Grund auf zu trainieren, da diese eine robuste visuelle Repräsentation bieten und beim *Transfer Learning* hocheffizient sind.

Text-Encoder

Die Funktionsweise von Text Encodern ist ähnlich der Verarbeitung bei Vision Encodern. Tokenisierte Inputs, in diesem Fall Textsequenzen, werden in den Einbettungsraum projiziert. Ältere multimodale Modelle wie CLIP, BLIP [69] und

ALIGN [70] nutzten sowohl einen dedizierten Text Encoder als auch einen Vision Encoder. Durch *Contrastive Learning* sind sie in der Lage, die Modalitäten in einem gemeinsamen latenten Raum auszurichten.

Aktuellere VLMs, wie beispielsweise LLaVA [71], nutzen keinen separaten Text Encoder, sondern verwenden ein LLM. Im Falle von LLaVA handelt es sich hierbei um LLaMA [72], um das Sprachverständnis zu gewährleisten.

Text-Decoder

Durch den Text Decoder können schlussendlich kohärente textuelle Antworten generiert werden, die sowohl visuelles als auch textuelles Verständnis mit einbeziehen. Der Text Decoder verwendet hierfür hauptsächlich LLMs als Textgeneratoren und nutzt kodierte visuelle Merkmale, um kontextreichen Output zu generieren. Beispiele für diesen Ansatz stellen GPT-4V [73], Flamingo [74] und Kosmos-2 [75] dar. Sie verwenden typischerweise Mechanismen zur visuellen Projektion, die es den mächtigen Language Decodern ermöglichen, visuelle Informationen effektiv zu integrieren [66].

Bei VLMs, welche von Grund auf trainiert werden, ist in der Regel ein separater Text Decoder notwendig. Modelle, die jedoch LLMs als Backbone einsetzen, verwenden häufig den bereits im LLM vorhandenen Decoder wieder.

Das Konzept des *Cross-Attention*-Mechanismus wird häufig als Interaktionsschicht zwischen den drei vorgestellten Komponenten genutzt. Dieser ermöglicht es den textuellen und visuellen Tokens, sich gegenseitig während der Verarbeitung zu beeinflussen. Da Vision Encoder und Text Encoder separate Darstellungen für jede Modalität erstellen, wird durch den *Cross-Attention-Layer* ein *Attention Score* modalitätsunabhängig berechnet. Dieser ermöglicht es textuellen Tokens, die visuellen Tokens zu berücksichtigen und umgekehrt [76].

Besonders für den Text Decoder ist dies relevant, da es dem Generierungsprozess ermöglicht, sich während der Erzeugung der Textausgabe dynamisch auf relevante visuelle Merkmale zu fokussieren. Modelle wie Flamingo und VisualBERT [77] setzen diesen expliziten *Cross-Attention-Layer* ein, um stärkere multimodale Interaktion während der Enkodierung und Dekodierung zu ermöglichen.

Der *Cross-Attention*-Mechanismus stellt allerdings nur eine mögliche Option dar. Einige Modelle, beispielsweise CLIP, adressieren diese Thematik auf andere Art und Weise. Eine Option, wie sie auch in CLIP verwendet wird, ist, die separaten Einbettungen von Bildern und Text direkt in einem gemeinsamen Einbettungsraum durch *Contrastive Learning* [78] in Beziehung zu setzen. Im Trainingsprozess zielt dies darauf ab, dass zueinander gehörige Bild-Text-Paare ähnliche Einbettungen aufweisen, während unabhängige Paare durch diesen Mechanismus voneinander separiert werden.

Für ein besseres Verständnis kann das Szenario der Radmontage in Abbildung 2.1 genutzt werden. CLIP erstellt in diesem Fall ein Embedding für das Bild und ein

separates Embedding für die verschiedenen Text-Labels wie Rad, Achse, Bremse. CLIP weist korrekten Labels, beispielsweise Rad und Achse, deren Embeddings nahe am Bild-Embedding liegen, hohe Ähnlichkeitswerte zu. Inkorrekte Labels, wie beispielsweise Bremse, hätten hingegen Embeddings, die im gemeinsamen Raum weit entfernt sind.

Kontrastives Lernen (engl. *Contrastive Learning*) ist ein effektiver Ansatz für Applikationen wie Bild-Text-Suche oder *Zero-Shot*-Klassifikation. Er hat jedoch nicht die detaillierte Interaktionsfähigkeit, welche der *Cross-Attention*-Ansatz bietet und die für komplexere multimodale Schlussfolgerungsaufgaben (engl. *Reasoning Tasks*) verwendet werden kann.

VLMs bieten durch die Einbindung von visuellem Input zusätzlich zum textuellen Input einige Vorteile. Allerdings sehen sie sich aktuell noch mit einigen grundlegenden Schwächen und Herausforderungen konfrontiert, die ihre Robustheit und Einsatzmöglichkeit einschränken:

- **Halluzinationen:** Wie auch LLMs neigen VLMs zu Halluzinationen, was ein signifikantes Problem darstellt. Modelle generieren auf den ersten Blick plausible Ausgaben, die aber faktisch falsche Beschreibungen visueller Informationen, Objekte, Attribute oder Bezeichnungen beinhalten, welche in den Eingangsdaten nicht vorhanden sind [79].

- **Ausrichtungs-Probleme:** Durch das fehlerhafte Ausrichten (engl. *Align*) von textuellen und visuellen Repräsentationen können Schwierigkeiten auftreten. Dadurch bedingt kann ein fehlerhaftes oder inkonsistentes multimodales Verständnis entstehen, durch das das Modell textuelle Beschreibungen nicht korrekt mit den entsprechenden visuellen Informationen verbindet [80].

- **Datenknappheit:** Da die Qualität der Modelle grundlegend von der Qualität der Trainingsdaten bestimmt wird, werden hochwertige Bild-Text-Paare benötigt, die korrekt annotiert sind. Durch die hohe Anforderung an die Menge der Trainingsdaten besteht das Risiko, dass Modelle möglicherweise Schwierigkeiten mit domänenspezifischen Inhalten oder unzureichend repräsentierten visuellen Konzepten haben [66].

- **Rechenanforderungen:** Das Training von großen Modellen auf großen multimodalen Datensätzen erfordert enorme Rechenkapazitäten. Diese Ressourcen können nur wenige Einrichtungen zur Verfügung stellen. Wird ein Modell auf unzureichenden Rechenkapazitäten trainiert, verschärfen sich die Probleme hinsichtlich des Verständnisses der Daten, und das Erreichen einer robusten Leistung über mehrere Domänen hinweg ist schwer realisierbar [66].

2.2.7. Prompting-Techniken

Bei einem Prompt handelt es sich um eine textuelle Anweisung, die an ein sprachbasiertes Foundation Model übergeben wird. Die Anweisung konfiguriert das Modell, sodass die Funktionen des Modells präzisiert und verbessert werden. Ein Prompt kann die von einem LLM erzeugten Antworten eingrenzen, indem er durch Guidelines spezifische Regeln und Parameter für die Interaktion mit dem Modell etabliert. Konkret definiert ein Prompt den konversationellen Rahmen für das Modell und spezifiziert, welche Informationen von besonderer Relevanz sind und in welcher Form, ferner mit welchem Inhalt eine Ausgabe erzeugt werden soll.

Prompt Engineering beschreibt in diesem Kontext den Prozess, sprachbasierte Foundation Models mittels gut gewählter Prompts zu programmieren und zu steuern. Nachstehende Ziele werden verfolgt [81]:

- **Verbesserung der Schlussfolgerungs- und Logikfähigkeiten:** Dazu müssen Prompting-Strategien entwickelt werden, die Modelle Schritt für Schritt durch strukturierte logische Prozesse führen. Dies ermöglicht das Lösen von komplexen Problemen, mathematisches Verständnis und Multi-Hop-Inferenzaufgaben, die eine bewusste Analyse und systematisches Denken erfordern.
- **Reduzierung von Halluzinationen:** Zur Reduktion von Halluzinationen wird versucht, durch *Retrieval-Augmented*-Ansätze und Verifikationsmechanismen die Antwort des Modells in faktenbasierten Informationen zu verankern (*Grounding*). Das sorgt für eine Minimierung der Generierung von falschen, irreführenden oder erfundenen Inhalten, denn die Antworten werden an verlässlichen externen Informationsquellen, z. B. dem Prompt, festgemacht.
- **Transfer auf neue Aufgaben:** Durch Techniken wie *Zero-Shot-Prompting* oder *Few-Shot-Prompting* ist es möglich, das grundlegende vortrainierte Wissen der Foundation Models in neue Domänen zu übertragen. Die Notwendigkeit von umfangreichen zusätzlichen Trainingsdurchläufen kann dadurch umgangen werden und erlaubt dennoch eine Anpassung auf neue Problemstellungen.
- **Verbesserung von Konsistenz und Kohärenz:** Dieses Ziel soll sicherstellen, dass Modellausgaben über die gesamte Interaktion mit dem Modell hinweg einen korrekten logischen Fluss aufweisen. Hierfür werden Ansätze wie *Contrastive Learning* verwendet, die das Modell unterstützen, korrekte und inkorrekte Denkmuster zu unterscheiden.

Bei einem Prompt wird konkret zwischen zwei Varianten [82] unterschieden: *Cloze Prompts* und *Prefix Prompts*. *Cloze Prompts* werden verwendet, wenn das Modell an einer oder mehreren Stellen Lücken innerhalb des vorgegebenen Textes füllen soll. Sie sind daher besonders für *Masked Language Models* (MLMs) wie das

Basic Prompt Engineering	Advanced Prompt Engineering	Prompt Optimization
Instructions	Chain-of-Thought	Textual Gradients
Role Prompting	Least-to-Most Prompting	Continuous Prompts
Triple Quotes Separation	Retrieval Augmentation	Constrained Answer Space
Few-Shot Prompting	Tree-of-Thoughts	
	Self-Consistency	

Tabelle 2.2.: Überblick über verschiedene Prompting-Techniken.

BERT-Modell geeignet, da sie starke Übereinstimmungen mit dem Format im *Pretraining Task* aufweisen. *Prefix Prompts* fokussieren sich auf die Weiterführung von gegebenen Textpräfixen. Sie eignen sich besonders für autoregressive LLMs wie z. B. GPT, da sie der von links nach rechts verlaufenden Generierung des Outputs der Modelle entsprechen.

Neben dieser grundlegenden Unterscheidung des Typs werden im *Prompt Engineering* systematische Ansätze zur Erstellung von Prompt-Vorlagen [83] erarbeitet. Tabelle 2.2 stellt eine Übersicht zu verschiedenen Prompting-Techniken und den Kategorien dar, denen sie zugeordnet werden können. Die variierenden Techniken kommen in unterschiedlichen Bereichen des gesamten Spektrums vom manuellen *Template Engineering* bis hin zu automatisierten *Discovery*-Methoden vor. Manuelles Engineering ermöglicht die präziseste Kontrolle von Promptstrukturen und ermöglicht die Einbindung von domänenspezifischem Wissen. Allerdings ist es auch mit einem erheblichen zeitlichen Mehraufwand verbunden, da die Prompts händisch erstellt werden und möglicherweise nicht die optimale Formulierung des Prompts gewährleistet ist. Zur Lösung dieser Herausforderungen hat sich das automatisierte Lernen von Prompt-Vorlagen (engl. *Template Learning*) durch zwei verschiedene Methoden [82] etabliert:

1. **Diskrete Prompt-Suche:** Hierbei handelt es sich um ein automatisches Verfahren, das gezielt nach konkreten, lesbaren Wörtern sucht, die als Eingabe ein KI-Modell dazu bringen, eine bestimmte Aufgabe so gut wie möglich zu lösen.

2. **Kontinuierliche Prompt-Optimierung:** Diese Methode arbeitet direkt im Einbettungsraum, indem eine Optimierung der kontinuierlichen Vektordarstellungen durchgeführt wird. Der Ansatz eliminiert die Beschränkung, dass Prompts interpretierbaren natürlichsprachlichen Sequenzen entsprechen müssen, und kann potenziell effektivere, für Menschen jedoch weniger verständliche Prompt-Repräsentationen finden.

Manuelles *Template Engineering* ermöglicht die Implementierung einiger grundlegender Prompting-Techniken, die auf menschlicher Intuition bezüglich effektiver Aufgabenformulierung basieren. Dazu zählen:

- **Zero-Shot-Prompting:** Hier wird eine Anweisung ohne jegliche Beispiele erstellt.

- **One-Shot- und Few-Shot-Prompting:** Es inkludiert ausgewählte relevante Beispiele in die Anweisung.

- **Role-based Prompting:** Es beschreibt die Rolle, die das Modell bei der Anweisung übernehmen soll.

Aufbauend auf den benannten grundlegenden Prompting-Techniken sind fortschrittlichere Techniken entwickelt worden. Sie sollen die Schlussfolgerungs- und Problemlösungsfähigkeiten von LLMs durch strukturelle Führung verbessern. Zu diesen Techniken gehören u. a.:

- **Chain-of-Thought (CoT) Prompting:** Diese Technik veranlasst Modelle, Zwischenschritte in ihrem Denkprozess zu generieren. Es entsteht so eine Argumentationskette, die logische Sprünge in der Ausgabe des Modells minimiert.

- **Self-Consistency:** Diese Methodik stellt eine Erweiterung des CoT-Ansatzes dar. Sie regt das Modell an, mehrere Lösungspfade zu generieren und die konsistenteste Antwort über diverse Wege hinweg zu wählen. Hierdurch wird die Konvergenzvalidierung verbessert, wodurch sich die Zuverlässigkeit erhöht.

- **Least-to-Most Prompting:** Hier wird der Ansatz der hierarchischen Problemdekomposition genutzt, indem komplexe Aufgaben in einfachere Teilprobleme zerlegt werden, welche sequenziell gelöst werden sollen. Bei einem holistischen Modell können derartige Aufgabenstellungen nicht korrekt bearbeitet werden.

- **Tree-of-Thoughts (ToT):** Der ToT-Ansatz erweitert die Fähigkeit des Modells, Schlussfolgerungen abzuwägen. Dazu werden mehrere Denkpfade parallel exploriert. Das Modell betrachtet verschiedene Lösungsstrategien, die es bei Bedarf auch schrittweise zurückverfolgen kann, und bewertet Teilbereiche im Entscheidungsbaum.

- **Retrieval-Augmented Generation (RAG):** Es handelt sich hier um einen Ansatz zur Minimierung der Halluzinationsproblematik, indem externe Wissensquellen direkt in den Prompting-Prozess eingebunden werden. Relevante, aus Datenbanken und Dokumenten abrufbare Fakten werden konkateniert, um die Antwort des Modells mit verifizierbaren Daten zu belegen.

- **Prompt Optimization:** Unter *Prompt Optimization* wird die systematische Verfeinerung von Prompt-Formulierungen durch automatisierte Methoden verstanden. Hierzu gehören gradientenbasierte Ansätze, Black-Box-Optimierungstechniken und textuelle Gradientenmethoden. Sie generieren die natürlichsprachliche Beschreibung von Fehlern in Prompts, um iterative Verbesserungen vorzunehmen.

Werden die beschriebenen Prompting-Techniken mit traditionellem *Fine-Tuning* verglichen, kann festgestellt werden, dass einer der Hauptunterschiede in ihrem Einfluss auf die Parameter des Modells besteht. Dies beeinflusst maßgeblich die möglichen Anwendungsszenarien der beiden Prinzipien.

Beim klassischen *Fine-Tuning* ohne Prompting wird eine direkte Parameteroptimierung vorgenommen, indem alle oder einige Parameter des vortrainierten Modells mittels Gradienten aus den Trainingssamples für das *Fine-Tuning* aktualisiert werden. Dies bietet große Vorteile in Szenarien, bei denen eine hohe Datenverfügbarkeit vorliegt, d. h. ein ausreichend großer Trainingsdatensatz für das neue Szenario erzeugt werden kann, weil sie die optimale Anpassung an die spezifische Aufgabe ermöglicht. In Fällen von *Few-Shot*-Szenarien kann dies jedoch auch ein Nachteil sein, da *Fine-Tuning* zu einem *Overfitting* führen kann. Zusätzlich kann es dazu führen, dass initial im Modell vorhandene Fähigkeiten verlernt werden.

Prompting-Methoden haben im Vergleich dazu Vorteile, wenn die vollständige Wissensbasis des vortrainierten Modells erhalten bleiben soll oder auch in Fällen von *Zero-Shot*-Szenarien. Sie können auf leistungsschwächeren Systemen genutzt werden, die zwar eine Modellinferenz ermöglichen, aber kein Training der Modelle. Durch fortgeschrittene Prompting-Techniken sind auch Anwendungen im Bereich von *Few-Shot*-Szenarien möglich sowie eine Verbesserung der Ergebnisse im Vergleich zu den grundlegenden Prompting-Techniken. Einer der gravierendsten Nachteile von Prompting-Ansätzen ist der erhebliche Engineering-Aufwand, der benötigt wird, um eine hohe Antwortgüte zu erhalten. Auch sind sie stark abhängig von der sorgfältigen und exakten Erstellung der Vorlagen.

2.3. Wissensgraph

Bei einem Wissensgraphen (engl. *Knowledge Graph*, KG) handelt es sich um eine strukturierte Repräsentation von Informationen, die als gerichteter Graph organisiert sind. Hierbei stellen die Knoten des Graphen die Entitäten (z. B. reale Objekte, aber auch abstrakte Konzepte) und die Kanten die semantische Beziehung zwischen den Entitäten dar. Einen fundamentalen Baustein eines Wissensgraphen stellt ein Tripel dar, bestehend aus (Subjekt, Prädikat, Objekt). Ein Beispiel hierfür ist `(Deckel, teil_von, Gehäuse)`. Durch diese Struktur kann Wissen faktenbasiert in einer maschinenlesebaren Form erfasst werden.

Wissensgraphen können in vier Hauptkategorien [84] differenziert werden, wobei für die Einordnung die Art der gespeicherten Informationen betrachtet wird.

- **Enzyklopädischer Wissensgraph:** Diese speichern allgemeines Weltwissen durch die Integration von Informationen aus unterschiedlichen Quellen. Beispiele eines solchen Wissensgraphen sind Wikidata [85] und Freebase [86], die breite faktische Informationen über Entitäten und deren Beziehungen abdecken.

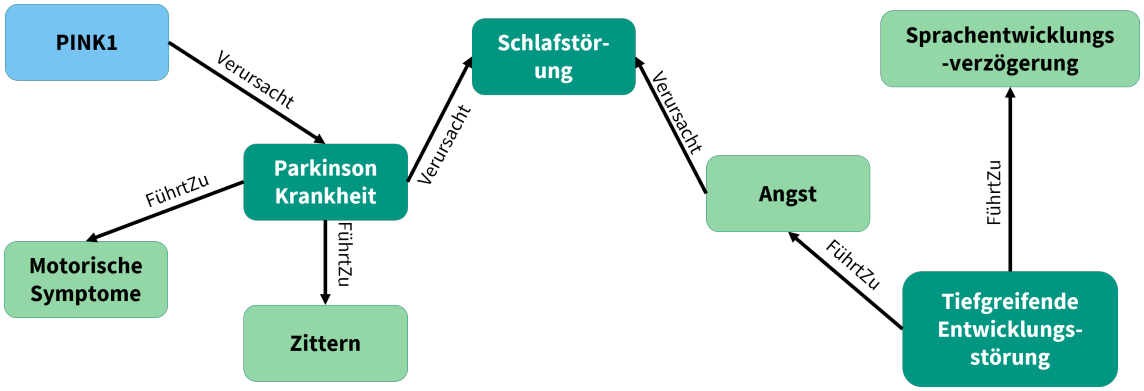


Abbildung 2.8.: Vereinfachte Darstellung eines domänenspezifischen Wissensgraphen, welcher die Domäne Schlafstörung umfasst.

- **Wissensgraph für Alltagswissen:** Diese fokussieren sich auf alltägliche Konzepte, Objekte und Ereignisse, die solches Wissen repräsentieren, das von Menschen typischerweise implizit verstanden wird. Beispiele hierfür sind ConceptNet [87], ein Wissensgraph, welcher Computer dabei unterstützt, Wortbedeutungen zu verstehen, oder ATOMIC [88], das kausale Beziehungen zwischen Ereignissen modelliert.
- **Domänenspezifische Wissensgraphen:** Diese repräsentieren spezielle Fachgebiete, z. B. Chemie, Mathematik oder Informatik. Sie bieten kleinere, aber dafür präzisere und zuverlässigere Informationen und Zusammenhänge innerhalb einer spezialisierten Domäne. Ein Beispiel im medizinischen Bereich ist UMLS [89]. Abbildung 2.8 zeigt einen domänenspezifischen Wissensgraphen im medizinischen Kontext.
- **Multimodale Wissensgraphen:** Während die zuvor benannten Kategorien reine Textinformationen speichern, gehen die multimodalen Wissensgraphen darüber hinaus und integrieren verschiedene Informationstypen wie Bilder, Töne oder Videos. Das ermöglicht Anwendungen wie visuelle Fragebeantwortung und modalitätsübergreifendes Schließen (*Cross-Modal Reasoning*). IMGpedia [90] ist ein Beispiel für ein solches System, welches Bild- und Textinformationen kombiniert.

Die Verbindung von LLMs und Wissensgraphen ist ein aktives Forschungsfeld. Es bietet viel Potenzial, da die komplementären Stärken beider Technologien verbunden werden können. Für diese Verbindung haben sich drei verschiedene Richtungen etabliert: Wissensgraph-gestützte LLMs (engl. *KG-enhanced LLMs*), LLM-angereicherte Wissensgraphen (engl. *LLM-augmented KGs*) und die Synergie von LLMs und Wissensgraphen (engl. *Synergized LLMs and KGs*).

Wissensgraph-gestützte LLMs [84] adressieren kritische Limitationen von LLMs, z. B. Halluzinationen, mangelnde Interpretierbarkeit und unzureichendes Domänenwissen, indem das strukturierte Faktenwissen aus dem Wissensgraphen eingebunden wird. Es haben sich dazu drei verschiedene Vorgehensmodelle etabliert:

1. **Wissensgraph-gestütztes Training** (engl. *KG-enhanced pre-training*): Integration der Wissensgraphen in die Trainingsphase (Pre-Training). Dies kann erreicht werden, indem Trainingsziele modifiziert werden, um Entitäts-Relations-Informationen einzubinden. Beispiele hierfür sind ERNIE [91] oder KEPLER [92]. Es ist ebenfalls möglich, das Wissen direkt in die Modelleingabe mittels strukturierter Repräsentationen einzubinden oder auch *Instruction-Tuning* zu verwenden. Dadurch wird dem LLM ein besseres Verständnis der Wissensgraphen-Struktur vermittelt.

2. **Wissensgraph-gestützte Inferenz** (engl. *KG-enhanced inference*): Hierbei wird der Wissensgraph erst zur Ausführungszeit eingebunden. Es existiert eine Trennung zwischen dem zusätzlichen Wissen des Wissensgraphen und dem grundlegenden Wissen des Modells. Der Wissensgraph wird in diesem Fall für dynamische Informationsupdates verwendet, etwa durch *Retrieval-Augmented*-Ansätze wie RAG, die relevante Informationen während der Ausgabegenerierung abrufen, oder durch Wissensgraph-Prompting-Techniken, welche das strukturierte Wissen des Wissensgraphen in eine Textsequenz umwandeln, die für die Anfrage an das Modell genutzt werden kann.

3. **Wissensgraph-gestützte Interpretierbarkeit** (engl. *KG-enhanced interpretability*): In diesem Vorgehen werden die Informationen des Wissensgraphen genutzt, um zu prüfen oder zu analysieren, was das LLM gelernt bzw. verstanden hat. Dadurch ergeben sich tiefere Einblicke in die interne Wissensrepräsentation des LLM und seinen Argumentationsprozess.

LLM-angereicherte Wissensgraphen [84] bezeichnet den Prozess, der das Sprachverständnis von LLMs nutzt, um verschiedene Aufgaben des Wissensgraphen zu verbessern. Folgende Beispiele dazu:

- Verwendung von LLMs als Text-Encoder für die bessere Repräsentation von Entitäten und Relationen.

- Vervollständigung und Konstruktion von Wissensgraphen.

- Als Möglichkeit, den Wissensgraphen zur Textgenerierung zu nutzen bzw. für die Fragebeantwortung mit dem Ziel, Lücken zwischen natürlicher Sprache und strukturiertem Wissen zu schließen.

Synergie von LLMs und Wissensgraphen [84] bezieht sich auf ein einheitliches Framework, in dem Wissensgraph und LLM als gleichwertige Komponenten genutzt werden. Das Framework kombiniert das implizite Wissen von LLMs mit dem expliziten strukturierten Wissen von Wissensgraphen durch synergetische Wissensrepräsentationen. Dadurch wird ein bidirektionales Schlussfolgern möglich, bei dem sich Wissensgraph und LLM gegenseitig in ihren Fähigkeiten ergänzen, um ein umfassendes Verständnis und präzisere Schlussfolgerung von Aufgaben und Antworten zu erreichen.

2.4. Affordanzen

Affordanz / Affordance stellen ein von J.J. Gibson [93] definiertes Konzept für die Beschreibung der wahrgenommenen und tatsächlichen Handlungsmöglichkeiten und Objektfunktionen dar, die ein Objekt oder eine Umgebung einer Person anbietet. Ähnliche Konzepte stellen der von Koffka definierte Forderungscharakter [94] bzw. der von Lewin definierte Aufforderungscharakter [95] bereit. Im Vergleich zu diesen beiden Konzepten sind Affordanzen eines Objektes jedoch nicht abhängig von dem konkreten Bedürfnis, welches der Verwender der Objekte hat, und somit unveränderlich. Als Beispiel bietet der Griff einer Schere immer die Aktion „in die Hand nehmen“ und die Scherenblätter die Aktion „schneiden“ an. Auch die Sitzfläche eines Stuhles bietet immer die Affordanzen „setzen“ und „darauf stehen“ an. Hierdurch kann die Menge der möglichen Affordanzen für ein Objekt deutlich größer sein als beim Konzept von Lewin oder Koffka; auch Mehrdeutigkeiten sind möglich. Welche Affordanz konkret verwendet wird, entscheidet somit der Nutzer individuell.
Besondere Beachtung finden Affordanzen u. a. beim Design von Umgebungen [96], Produkten [97] und Interfaces [98]. Das Konzept wird hier genutzt, um ein spezifisches Nutzerverhalten zu fördern oder gewisse Funktionalitäten einfach verständlich umzusetzen.
Auch in der Robotik finden Affordanzen in verschiedenen Bereichen Anwendung. So werden Robotersystemen mittels Affordanzen Informationen ihrer Umgebung vermittelt, beispielsweise welche Bereiche eines Objekts gegriffen oder an welchen Stellen Objekte abgestellt werden können [99]. Darauf aufbauend ist es ebenfalls möglich, Affordanzen für die konkrete Trajektorienplanung zu nutzen [100]. Des Weiteren unterstützen Affordanzen in der Robotik die Abstraktion von Handlungen auf ähnliche Objekte [101].

3. Stand der Forschung

Im folgenden Kapitel werden verschiedene Ansätze zu relevanten Themenbereichen des in dieser Arbeit entwickelten Konzeptes vorgestellt. Zunächst werden Ansätze zur Erweiterung von Montage- und Demontageplanung, etwa durch moderne, KI-basierte Planungsmethodiken, betrachtet. Anschließend erfolgt eine Analyse von Methoden zur Integration von CAD-Systemen und Foundation Models. Im weiteren Verlauf werden Ansätze untersucht, die Foundation Models für die robotische Manipulation einsetzen. Berücksichtigung finden ebenfalls Verfahren für die Affordanzdetektion sowie die Nutzung von Affordanzen im Planungsprozess. Wie die Auseinandersetzung mit den verschiedenen theoretischen Zugängen und Ansätzen deutlich macht, zeigt sich eine übergreifende Forschungslücke. Diese wird analysiert und reflektiert mit dem Ziel, ein Konzept zu entwickeln, das geeignet erscheint, das Forschungsdesiderat zu beheben.

3.1. Fortschritte in der automatischen Montageplanung

Der Fokus der Arbeit liegt besonders auf der automatisierten Montageplanung bzw. der zerstörungsfreien automatisierten Demontageplanung. Entsprechend befasst sich der folgende Abschnitt mit den aktuellen Fortschritten auf diesen Gebieten und erweitert die in Kapitel 3 erläuterten Methoden. Schwerpunkte bilden das ASP und APP. Zunächst beschreibt Abschnitt 3.1.1 Ansätze zur Verwendung von KI-basierten Verfahren für die Lösung des Planungsproblems. Daran anschließend wird in Abschnitt 3.1.2 untersucht, welche Ansätze existieren, um Kollisionen und physikalische Randbedingungen bei ASP- und APP-Problemen zu berücksichtigen. Abschließend fasst Abschnitt 3.1.3 Lösungen für die automatische Integration von CAD-Informationen in den Planungsprozess zusammen.

3.1.1. Planungsmethoden

Eine Vielzahl verschiedener Ansätze zur Integration von KI-Methoden in das Montageplanungsproblem ist über die Jahre entwickelt und untersucht worden.

Die vielversprechendsten Ansätze lassen sich primär in die drei Forschungsrichtungen Graph Learning, Reinforcement Learning und Foundation Models unterteilen.

Graph Learning im Rahmen von Montage- bzw. Demontageplanung verfolgt im Kern die Idee, durch die Verwendung von ML-Methoden die Relationen innerhalb eines Graphen, z. B. die Bauteilabhängigkeiten, direkt zu erfassen. Diese Relationen werden automatisiert auf Embedding-Vektoren abgebildet, ohne dass komplexe topologische Informationen vorab händisch vereinfacht oder extrahiert werden müssen. Durch die Verwendung von Deep-Learning-Methoden identifizieren und extrahieren Graph-Learning-Verfahren relevante Grapheigenschaften und transformieren sie in kontinuierliche Vektorrepräsentationen, die für Anwendungen wie Kantenprädiktion oder Knotenklassifikation geeignet sind [102]. Im Rahmen von AP und DP existiert eine Vielzahl von Anwendungsmöglichkeiten für Graph Learning. Im Folgenden werden vier Ansätze vorgestellt, welche aktuelle Beispiele für die Verwendung von Graphrepräsentationen im Planungskontext darstellen.

Ma et al. [103] stellen ein heterogenes Graph-Transformer-Framework vor, welches Montageregeln erlernt, indem es verschiedene Informationen als unterschiedliche Arten von Kanten und Knoten repräsentiert. Knoten können entweder Bauteile oder Verbindungspunkte sein. Kanten können entweder kinematische Beschränkungen zwischen Bauteilen, Verbindungen zwischen Slots (welche Abstände oder Winkel repräsentieren) oder Verbindungen zwischen Bauteilen und ihren zugehörigen Slot-Knoten darstellen. Der Ansatz generiert einen Ground-Truth-Datensatz, indem vollständige Baugruppen schrittweise in Zwischenzustände zerlegt werden und markiert wird, welche Verbindungen in jedem Zustand möglich sind. Der Datensatz wird anschließend verwendet, um einen Graph Transformer zu trainieren, der die Wahrscheinlichkeiten für alle möglichen nächsten Zustände vorhersagt. Basierend auf diesen Vorhersagen wählt ein Algorithmus den nächsten logischen Zustand aus und validiert die Wahl durch einen CAD-Solver. Dies ermöglicht die Ableitung des spezifischen Montageschrittes, der erfolgen muss, um den gewählten Zustand zu realisieren und sich einem Zielzustand zu nähern.

Im Gegensatz zum heterogenen Graphansatz von Ma et al. präsentieren Huang et al. [104] ein dynamisches Graph-Learning-Framework für die 3D-Bauteilmontage. Dieses kann Vorhersagen treffen, wie jedes Bauteil im Raum positioniert und orientiert werden sollte. In diesem Ansatz wird jedes Bauteil als ein Knoten im Graphen realisiert, und mittels neuronaler Netze wird die 6-DoF-Pose bestimmt, die jedes Bauteil in der finalen Baugruppe einnehmen muss. Hierzu wird das Netz mit verschiedenen Montagekonfigurationen trainiert, wobei unterschiedliche Zwischenzustände betrachtet werden. Das Netz lernt darauf basierend, die notwendigen Transformationen zu definieren, die ein Bauteil aus seiner aktuellen Lage in die Endlage in der finalen Baugruppe bewegen.

Einen weiteren graphbasierten Ansatz stellen Attad et al. [105] mit dem Framework *GRACE* vor. Vergleichbar mit dem Konzept von Ma et al. werden hier Bau-

teile und ihre Oberflächen als Knoten repräsentiert, während Kanten die geometrischen Beziehungen zwischen den Oberflächen erfassen. Durch die Verwendung von Graph Attention Networks erfasst das Framework den aktuellen Montagezustand und leitet daraus eine Wahrscheinlichkeitsverteilung ab, die angibt, welches Bauteil als Nächstes gefügt werden sollte. Anschließend wird ein Entscheidungsbaum mit Tiefensuche genutzt, um vollständige Montagesequenzen zu generieren, wobei Backtracking-Mechanismen integriert sind, falls nicht realisierbare Montagepfade auftreten. Im Gegensatz zu früheren Ansätzen konzentriert sich *GRACE* auf die Bestimmung der zeitlichen Abfolge von Fügeoperationen unter expliziter Berücksichtigung der Einschränkungen von Robotersystemen und Machbarkeitsprüfungen.

Einen weiteren graphbasierten Ansatz präsentieren Hansjosten et al. [106]. Dieser fokussiert sich auf die Demontage und betrachtet auch nicht-zerstörungsfreie Demontageprozesse. Kernziel des Ansatzes ist die automatische Generierung von Demontagegraphen aus 3D-Modellen. Dabei werden in einem ersten Schritt geometrische Kontakte analysiert und geometrische Abhängigkeiten identifiziert. Diese werden anschließend durch weitere Nicht-Geometrie-Verbindungen (Joints), z. B. Schrauben, Sicherungsringe und Klebstoffe, ergänzt. Aus den Informationen modelliert das System zunächst einen ungerichteten Graphen. Dieser wird mittels gerichteter Traversierung in einen gerichteten Zustandsgraphen überführt. Hierbei repräsentiert jede Kante einen möglichen Demontageschritt. Mittels Dijkstra-Algorithmus und Kostenmetrik (z. B. Zeit oder Zerstörungsgrad) wird anschließend eine optimale Demontagesequenz für ein spezifisches Demontageziel berechnet.

Graph-Learning-Ansätze für die Montage- und Demontageplanung zeigen, wie die Repräsentation von Baugruppen als heterogene Graphen neuronalen Netzen und Algorithmen ermöglicht, komplexe Beziehungen zwischen Bauteilen, Randbedingungen und Montagezuständen zu erlernen. Während diese KI-Methoden die Nutzung struktureller Informationen mittels Graphrepräsentationen möglich machen, erfordern sie typischerweise große Datensätze mit gelabelten Beispielen. Dem entgegen stehen Reinforcement-Learning-Ansätze, die Montagestrategien durch Trial-and-Error-Interaktionen mit der Umgebung erlernen können.

Reinforcement Learning (RL) nutzt einen Agenten, der seinen Umgebungszustand wahrnimmt, Aktionen ausführt und Belohnungen basierend auf seiner Aufgabenerfüllung erhält. Die Policy des Agenten, welche die Aktionsauswahl bestimmt, wird durch iterative Trainingsepisoden entwickelt, in denen Belohnungen oder Bestrafungen die Verhaltensoptimierung in Richtung der Maximierung der kumulativen Belohnung steuern. Deep Reinforcement Learning (DRL) erweitert diesen Ansatz durch die Einbindung tiefer neuronaler Netze als Funktionsapproximatoren, um komplexe Zustandsrepräsentationen und Aktionsräume zu approximieren. Dies ermöglicht ein effektives Training durch zyklische Agenten-Umgebungs-Interaktionen. Dabei generiert jede Episode Daten, die zur Anpassung der Netzwerkparameter und zur Verbesserung der Policy genutzt werden.

Bei der Anwendung von RL-Methoden auf das AP können drei Forschungsbereiche basierend auf dem Design der Belohnungsfunktion und den Implementierungsstrategien unterschieden werden [107].

Der erste Forschungsbereich konzentriert sich auf die zeitbasierte Optimierung. Hier lernen die Agenten, die Montagezeit durch direktes zeitliches Feedback zu minimieren. So normalisieren beispielsweise Neves et al. [108] die Dauer von Montageschritten zwischen empirisch ermittelten Minimal- und Maximalgrenzen und integrieren Bestrafungen für Werkzeugwechsel, um die Sequenzoptimierung für eine aus neun Teilen bestehende Montage eines Flugzeugspielzeugs zu steuern. Ähnlich dazu verwenden Winter et al. [109] schrittbasierte Belohnungen, bei denen jede Aktion eine negative Bestrafung erhält, während eine erfolgreiche Bauteilmontage positives Feedback liefert. Dies fördert effektiv kürzere Montagepfade.

Der zweite Bereich behandelt zustandsbasierte Belohnungen, bei denen Agenten Feedback basierend auf dem Erreichen gültiger Montagekonfigurationen oder der Vermeidung ungültiger Zustände erhalten. Zhao et al. [110] implementieren ein binäres Belohnungssystem (+1 für korrekte vollständige Montage, -1 andernfalls). Kombiniert wird dieses mit Faltungsnetzen, die sowohl Bilder des Montagezustands als auch Bauteilbilder verarbeiten, um optimale Montagesequenzen zu bestimmen. Antonelli et al. [111] erweitern diesen Ansatz durch die Einbeziehung von Mensch-Roboter-Kollaborationsszenarien, wobei das Erreichen vordefinierter Montagezustände positive Belohnungen auslöst. Die Anpassungsfähigkeit an Änderungen durch menschliche Bediener bleibt hierbei erhalten.

Der dritte Forschungsbereich adressiert die Optimierung roboterbezogener Randbedingungen unter Berücksichtigung physikalischer Einschränkungen und des Energieverbrauchs in automatisierten Montageszenarien. Yin et al. [112] demonstrieren diesen Ansatz durch das Training von Multi-Roboter-Systemen zur Montage von Fachwerkstrukturen bei gleichzeitiger Optimierung von Montagezeit, Energieverbrauch und struktureller Stabilität durch gewichtete Belohnungsfunktionen. Dabei werden die Roboterpositionierung, Montageaktionen und die Erfüllung dynamischer Randbedingungen berücksichtigt.

RL-Ansätze haben ihre Effektivität beim Erlernen von Montagestrategien durch Interaktion mit der Umgebung unter Beweis gestellt, erfordern allerdings umfangreiche Trainingsepisoden und Rechenressourcen, um optimale Policies zu entwickeln. Foundation Models stellen ein alternatives Paradigma dar, das vorab trainiertes Wissen aus riesigen Datensätzen nutzt und potenziell die Notwendigkeit für aufgabenspezifisches Training in Montageplanungsszenarien eliminiert. Da die robotische Montage als eine Unterkategorie der robotischen Manipulation betrachtet werden kann - da Manipulationsaktionen verwendet werden, um Produkte zu montieren -, lassen sich viele in Kapitel 3.3 hervorgehobene Konzepte auf Montageanwendungsfälle übertragen. Eine genaue Vorstellung von relevanten aktuellen Ansätzen ist daher in diesem Kapitel zu finden.

Eine erhebliche Herausforderung bei der Nutzung von KI-basierten Ansätzen in AP- und DP-Szenarien stellt die effiziente Einbindung der KI in der Anwendung, insbesondere in bestehende Anwendungen, dar. Frisch et al. [113] adressieren diese Problematik in ihrer Arbeit mit einem modularen Deployment-Konzept auf Basis des Robot Operating Systems (ROS). In ihrem Ansatz verwenden sie eine Low-Code-Pipeline, die komplexe ML-Modelle in spezialisierte ROS-Nodes für die Datenerfassung, Vorverarbeitung und Inferenz separiert. Durch die Aufteilung in einzelne Komponenten wird eine Modularität erreicht. Sie ermöglicht es, KI-basierte Methoden, beispielsweise Autoencoder für die Anomalieerkennung oder Condition Monitoring, flexibel in bestehende robotische Systeme zu integrieren. Dies reduziert die Notwendigkeit, die gesamte Steuerungssoftware neu strukturieren zu müssen.

3.1.2. Physiksimulation und Kollisionserkennung

Ein weiteres eng mit den Themen Montage- und Demontageplanung verbundenes Forschungsfeld ist die Kollisionsdetektion bzw. Kollisionsvorhersage sowie die generelle Thematik der Physiksimulation. Hierbei ist eine Kollision definiert als Interferenz von Teilen, bei der zwei oder mehr Objekte gleichzeitig denselben räumlichen Bereich beanspruchen [114].

Ben Said et al. [115] adressieren die Kollisionsidentifikation durch einen systematischen zweistufigen Ansatz, der zunächst primäre Fügepfade unter Vermeidung von Kollisionen der Bauteile untereinander generiert und die Trajektorien anschließend modifiziert, um statische Umgebungshindernisse zu vermeiden. Diese Methode verwendet eine inkrementelle Kollisionserkennung entlang der geplanten Trajektorien und identifiziert kritische Kollisionsparameter, sogenannte *Final Free Points* und *Initial Free Points*, um systematisch um erkannte Kollisionsstellen im Arbeitsraum zu navigieren. In diesem Zusammenhang repräsentieren Final Free Points den letzten kollisionsfreien Punkt in der generierten Trajektorie vor einem Hindernis, während Initial Free Points den ersten kollisionsfreien Punkt in der generierten Trajektorie nach einem Hindernis bezeichnen.

Das Framework *Assemble Them All* von Tian et al. [114] erzeugt die Kollisionsvorhersage durch eine physikbasierte Simulation, welche Signed Distance Fields (SDF) für eine präzise geometrische Interferenzdetektion verwendet. SDFs sind mathematische Funktionen, die jeden Punkt im 3D-Raum auf seinen kürzesten Abstand zur Oberfläche eines Objekts abbilden, wobei negative Werte Punkte innerhalb des Objekts und positive Werte Punkte außerhalb anzeigen. Dies ermöglicht eine effiziente und genaue Kollisionserkennung durch Prüfung, wann der Abstand zwischen Objekten negativ wird. Der Ansatz nutzt diskrete Kräfte in einer Physiksimulation, um kollisionsfreie Zustände vorherzusagen. Dadurch können komplexe Geometrien und besonders Engstellen, die herausfordernd für traditionelle geometrische Kollisionserkennungsmethoden sind, effektiv gehand-

habt werden, während gleichzeitig die Recheneffizienz für Baugruppen mit vielen Teilen gewahrt bleibt.

Mit dem *ASAP*-Framework erweitern Tian et al. [116] ihren vorherigen Ansatz durch die Integration von sowohl Kollisionsidentifikation als auch physikalischer Zustandsvorhersage in einem kombinierten Framework. Dieses kombiniert SDF-Kollisionserkennung mit strafbasierten Kontaktmodellen und einer Analyse der Stabilität in Bezug auf die Gravitation. *ASAP* verwendet einen Demontage-Baumsuchalgorithmus, der entweder geometrische Heuristiken oder GNNs zur Steuerung der Teileauswahl nutzt, während eine physikbasierte Simulation eingesetzt wird, um sowohl kollisionsfreie Pfade als auch die Stabilität mittels Halteplänen zu überprüfen. Haltepläne spezifizieren, welche Bauteile physisch gehalten werden müssen (z. B. durch robotische Greifer), um keine Instabilitäten oder nicht realisierbaren Szenarien während der Montage, etwa durch freischwebende Bauteile, zu erzeugen. Somit detektiert das System nicht nur geometrische Kollisionen, sondern sagt auch physikalisch mögliche Montagezustände unter Berücksichtigung von Schwerkraft, Reibung und Halteanforderungen vorher. Ferner wird sichergestellt, dass kollisionsfreie Montagesequenzen unter realen physikalischen Randbedingungen stabil und ausführbar bleiben.

Die bisher beschriebenen Ansätze betrachten primär die Aufgabenstellung, wie mögliche Kollisionen vorhergesagt werden können oder auch, wie sich einzelne Montageschritte auf die Baugruppe auswirken. Prozessschritte wie additive Fügeprozesse (z. B. 3D-Druck) oder auch zerspanende Prozesse sind durch diese Ansätze jedoch nicht abbildbar, aber häufiger Bestandteil moderner Fertigung. Für derartige Prozessschritte existiert ebenfalls eine Vielzahl an Prozesssimulationen, die jedoch nicht direkt in die bisherigen Ansätze integriert werden können. Im Framework *PyBullet Industrial* von Baumgärtner et al. [117] wird genau diese Herausforderung thematisiert. Das Framework erweitert die von verschiedenen Robotersimulationen genutzte Physik-Engine PyBullet um eine Prozesssimulationskomponente. So ist es möglich, nicht nur kinematische Ketten, sondern auch physikalische Effekte wie Prozesskräfte, Materialabtrag oder Materialauftrag zu simulieren. Dies ist eine unerlässliche Voraussetzung, wenn Montage- oder Demontagevorgänge simuliert oder validiert werden müssen, bei denen die Werkzeuginteraktion mit den Bauteilen relevant ist.

Die praktische Anwendung und die Notwendigkeit einer solchen physikbasierten Validierung in einem adaptiven Kontext zeigen Hansjosten et al. [118] am Beispiel der autonomen Zerlegung von alten Elektromotoren. Da Altprodukte häufig im Laufe ihrer Verwendungszeit geometrische Abweichungen oder Defekte entwickeln, sind die vorliegenden Produkte nicht exakt deckungsgleich mit den ursprünglichen CAD-Daten. Der Ansatz von Hansjosten nutzt an dieser Stelle Sensorik und die Simulation zur Echtzeit-Validierung von Prozessalternativen. Scheitert eine geplante zerstörungsfreie Zerlegeoperation, erlaubt die Kopplung der Planungsebene mit der physikalischen Simulation die autonome Generierung und Prüfung von alternativen Strategien, wie beispielsweise den Wechsel

zu einem (teil-)zerstörenden Prozess. Dieser geschlossene Kreislauf aus Wahrnehmung, physikalischer Simulation und adaptiver Planung ermöglicht es, die Unsicherheiten bei der Handhabung gebrauchter Komponenten prozesssicher zu beherrschen.

3.1.3. CAD-basierte Ansätze

Im Rahmen von AP und DP ist eine Vielzahl verschiedener Informationsquellen für Kollisionsinformationen und Randbedingungen vorhanden. CAD-Daten stellen in diesem Kontext die am häufigsten vorliegende Datenquelle dar. Sie sind größtenteils auch die Datenquelle mit den meisten bzw. detailliertesten Informationen. Entsprechend existieren verschiedene Ansätze zur Nutzung von CAD-Informationen für AP und DP. Zahlreiche Ansätze sind in diesem Zusammenhang einer von zwei Kategorien zuordenbar: die Extraktion geometrischer Randbedingungen und die Nutzung der Metadaten.

Ein Konzept zur Extraktion geometrischer Randbedingungen aus CAD-Daten legen Tariki et al. [119] vor. Sie führen einen Algorithmus ein, der systematisch die 3D-Geometrie analysiert, um zwei kritische Arten räumlicher Beziehungen automatisch zu extrahieren. Erstens werden interferenzfreie Matrizen berechnet, indem Kollisionsszenarien systematisch analysiert werden, bei denen jedes Teil virtuell von externen Positionen entlang der Koordinatenrichtungen $\pm x, y, z$ in seine finale Montageposition bewegt wird. Diese geometrische Analyse bestimmt, ob Bauteil A nach Bauteil B ohne Kollision montiert werden kann, und etabliert binäre Kompatibilitätsbeziehungen, die machbare Montagesequenzen definieren. Zweitens werden Fügebeziehungen extrahiert, indem geometrische Merkmale analysiert werden, um Male-Female-Teilebeziehungen zu identifizieren (z. B. das Erkennen von Löchern, die das Einfügen entsprechender Stifte oder Bolzen erfordern). Ein genetischer Algorithmus nutzt diese extrahierten geometrischen Randbedingungen anschließend als Fitnessfunktionen, um Montagesequenzen zu generieren, die sowohl die Kollisionsvermeidung als auch eine bevorzugte Einfügereihenfolge berücksichtigen (z. B. Sicherstellung, dass Female-Teile mit Löchern positioniert sind, bevor Male-Teile eingefügt werden).

Kiyokawa et al. [120] erweitern diesen Ansatz der Extraktion geometrischer Randbedingungen über die statische räumliche Analyse hinaus hin zur dynamischen physikalischen Simulation. Im Vergleich zum Ansatz von Tariki et al., der fixe geometrische Beziehungen aus CAD-Modellen extrahiert, simuliert das Framework den tatsächlichen Montageprozess in Bewegung. Das System analysiert, wie sich Bauteile bei der Montage verhalten. Dazu prüft es Kontaktflächen, berechnet die nötigen Kräfte für das Zusammenfügen und sagt mögliche Unsicherheiten beim Kontakt voraus. Die CAD-Geometrie dient als Eingabe für eine *Contingent Contact-Exploring Rapidly-exploring Random Trees* genannte Trajektorienplanung. Diese generiert Robotertrajektorien, welche sowohl die freie Bewegung im Raum als auch kontaktintensive Interaktionen zwischen Bauteilen

berücksichtigen. Anstatt das CAD-Modell als statische Quelle von Randbedingungen zu behandeln, implementiert dieser Ansatz eine dynamische Simulation, in der die geometrischen Informationen das Fundament für eine physikbasierte Bewegungsplanung bilden, welche die reale Montagedynamik berücksichtigt.

Während die vorangegangenen Ansätze geometrische Randbedingungen direkt für die Planung nutzen, verwenden Thomas et al. [121] diese als Vorwissen, um das Erlernen einer RL-Policy zu steuern. Das System extrahiert zunächst die Informationen zur geometrischen Machbarkeit aus CAD-Modellen, um Bewegungspläne zu generieren, und nutzt diese Pläne dann als Referenztrajektorien für das Training einer Policy eines Roboteragenten, die die komplexe, kontaktintensive Dynamik während der tatsächlichen Montageausführung handhaben kann.

In die zweite Kategorie fallen Ansätze, die CAD-Daten als kodierte Design- bzw. Produktbeschreibung und nicht bloß als geometrische Randbedingungen behandeln. Das *Auto-Assembly*-Framework von Chervinskii et al. [122] kann dieser Kategorie zugeordnet werden, da es die CAD-Informationen als eine vollständige Spezifikation behandelt. Das System analysiert die Designdateien, um automatisch Montagesequenzen abzuleiten, Prozessstücklisten zu generieren und Robotersteuerungscode zu erstellen, indem das CAD-Modell so behandelt wird, als enthielte es implizite Montageanweisungen, die dekodiert und ausgeführt werden können. Dieser Ansatz geht davon aus, dass die Struktur, wie Teile in der CAD-Umgebung entworfen und organisiert sind, die vom Konstrukteur beabsichtigte Montagemethodik widerspiegelt.

Der Ansatz *JoinABLE* von Willis et al. [123] führt dieses Konzept noch weiter, indem die in CAD-Dateien eingebetteten parametrischen Verbindungsinformationen als schwache Überwachung für maschinelles Lernen genutzt werden. Anstatt die Geometrie zu analysieren, um auf Randbedingungen zu schließen, werden die expliziten Verbindungsdefinitionen, Montagegraphen und parametrischen Beziehungen verwendet, die Konstrukteure in CAD-Systemen definieren. Dieser Ansatz nutzt, dass parametrische CAD-Dateien reichhaltige semantische Informationen darüber enthalten, wie Bauteile verbunden werden sollen. Das sind Informationen, die weit über das hinausgehen, was allein aus der Geometrie abgeleitet werden kann. Das System lernt so, Verbindungstypen und Montagekonfigurationen vorherzusagen, indem es die expliziten Annotationen des Konstrukteurs über Teilebeziehungen versteht. Ein solches Konzept setzt jedoch zwingend voraus, dass entsprechende Informationen auch in den CAD-Daten enthalten sind.

3.2. Methoden der Interaktion zwischen CAD und Foundation Models

In den letzten Jahren haben besonders Foundation Models diverse neue Anwendungsbereiche für KI erschlossen. Auch im Kontext von Montage und Demonta-

ge ermöglichen diese neuen Interaktionen etwa mit den CAD-Daten. Dabei können die CAD-Daten sowohl als Datenquelle als auch als Generierungsziel verwendet werden, sodass CAD-Modelle durch natürlichsprachliche Anweisungen generiert werden.

3.2.1. Generierung von CAD-Modellen durch Foundation Models

Insgesamt betrachtet zeigt sich, dass in den vergangenen Jahren verschiedenartige Ansätze entwickelt wurden, die eine Generierung von CAD-Modellen durch Foundation Models ermöglichen. Häufig erzeugen diese jedoch eine Zwischenrepräsentation und nicht direkt 3D-CAD-Modelle oder 2D-CAD-Zeichnungen. Ein möglicher Grund hierfür liegt in der Schwierigkeit aktueller Modelle, präzise 3D- oder 2D-Ausgaben zu liefern [124]. Eine Übersichtsarbeit von Zhang et al. [124] macht deutlich, dass die derzeit häufigsten Anwendungen die CAD-Code-Generierung, die Textgenerierung und die parametrische CAD-Generierung umfassen.

CAD-Code-Generierung bezieht sich auf die Verwendung sprachbasierter Foundation Models zur Erzeugung von ausführbarem Code, der zur Erstellung von 3D-CAD-Modellen oder zur Durchführung CAD-bezogener Aufgaben genutzt werden kann. Mögliche Ausgabeformate sind Python- und C++-Skripte, SQL-Abfragen oder softwarespezifischer CAD-Code. Der Ansatz der CAD-Code-Generierung nutzt das Pre-Training von Foundation Models auf sehr großen Code-Datenbanken, um automatisch funktionalen Programmcode aus natürlichsprachlichen Beschreibungen oder anderen Eingaben zu produzieren und anschließend die gewünschten CAD-Ausgaben zu erzeugen.

Bei der Herangehensweise von Li et al. [125] werden mehrere verschiedene Datenformate, darunter textuelle Beschreibungen, Bilder und Ground-Truth-3D- Formen zur Generierung des Programmcodes genutzt. Die 3D-Formen werden als Bilder gerendert. Deshalb besteht die Eingabe entweder nur aus Text oder aus zusätzlichen Bildern verschiedener Typen. Zusätzlich ist ein Debugger integriert, der den erzeugten Code iterativ verfeinert, bis eine erfolgreiche Validierung stattgefunden hat. Ein kontraintuitives Ergebnis ihres Ansatzes ist, dass das verwendete VLM GPT-4V mit reiner Texteingabe am besten abschnitt und multimodale Eingaben im Durchschnitt sowohl bei der Parsing-Rate als auch bei der geometrischen Genauigkeit übertraf. Hierbei beschreibt die Parsing-Rate den Prozentsatz der generierten CAD-Programmcodes, die ohne Syntaxfehler erfolgreich ausgeführt werden konnten, um eine 3D-Form zu erzeugen. Die geometrische Genauigkeit fasst zusammen, wie genau die generierte 3D-Form mit der beabsichtigten Ground-Truth-Form übereinstimmt, gemessen als das Verhältnis des überlappenden Volumens zum gesamten kombinierten Volumen beider Formen.

Ein Konzept für die Code-Generierung unter Verwendung von Python wird von Mallis et al. [126] präsentiert. Ihr Framework stellt einem VLM-Planer eine mul-

timodale Benutzeranfrage als Kontext zur Verfügung. Der Planer stellt eine Vorgehensweise bereit, einen Plan, eine natürlichsprachliche Beschreibung des geplanten nächsten Ausführungsschrittes sowie eine Aktion als Python-Code, der den Plan ausführt, zu erzeugen. Die Aktion wird anschließend in einem Python-Interpreter ausgeführt, der in FreeCAD integriert ist. Die Ausgabe wird als Kontext an das VLM zurückgegeben. Sie kann entweder Text oder Renderings des aktuellen Zustands des CAD-Objekts sein.

Im Gegensatz zu Code-Generierungsaufgaben konzentriert sich die parametrische CAD-Generierung auf die Beschreibung von CAD-Dateien durch parametrische Daten, die häufig in JSON-Dateien gespeichert werden [124]. Der Ansatz von You et al. [127] verwendet das GPT-4V-Modell zur Vorhersage der CAD-Struktur, die aus diskreten Befehlstypen wie Linien, Bögen, Kreisen und semantischen Teilelabels besteht. Ein separates Transformer-Netzwerk sagt die kontinuierlichen numerischen Parameter, Koordinaten, Winkel und Abstände für jeden Befehl voraus. Diese Faktorisierung nutzt das visuelle Verständnis von GPT-4V für die strukturelle Zerlegung; ein spezialisiertes neuronales Netz wird für die präzise geometrische Parametervorhersage eingesetzt.

Xu et al. [128] stellen ein Framework bereit, das ein Large Language Model (LLM) verwendet und neben textuellen Beschreibungen auch Multi-View-Bilder und eine Punktwolke durch modalitätsspezifische Encoder leitet. Die Ausgabe ist eine Sequenz parametrischer CAD-Befehle, z. B. Skizzen- und Extrusionsoperationen mit geometrischen Parametern, die in Standard-CAD-Software ausgeführt werden können, um ein editierbares 3D-Modell zu generieren.

Die dritte gängige Nutzung von LLMs für die CAD-Erstellung ist die Textgenerierung. Hierbei ist das Hauptziel, die Nutzer in ihrem Konstruktionsprozess zu unterstützen. Als Beispiel stellen Liu et al. [129] ein Framework bereit, bei dem Nutzer ihre anfängliche Konstruktionsidee übermitteln können. Basierend darauf generiert GPT-3 Prompt-Vorschläge, von denen die Nutzer einige auswählen können, welche dann in einen editierbaren Prompt umformuliert werden. Dieser Prompt wird schließlich an DALL-E (ein von OpenAI entwickeltes KI-Bildgenerierungsmodell) übergeben, das ein Bild erstellt. Auf der Grundlage dieses Bildes kann der Nutzer dann Änderungen anfordern, bis das Bild dem Verständnis des Nutzers von den ursprünglichen Designabsichten entspricht.

3.2.2. Nutzung von CAD-Modellen als Eingabe für Foundation Models

Anstelle CAD-Modelle zu generieren, können diese auch als Eingabe für Foundation Models bzw. deren Prompts dienen. Dabei können verschiedene Datenformate zum Einsatz kommen, einschließlich Renderings (Li et al. [125]) oder Punktwolken (Xu et al. [128]).

Diese Konzepte werden durch ein Framework von Kienle et al. [130] erweitert, das sich in zwei Teile differenziert: *SegCAD* und *QueryCAD*. Das übergeordnete Ziel dieses Frameworks ist die präzise Beantwortung von Fragen zu CAD-Dateien. Dieses erfolgt durch das *QueryCAD*-System und ist nur durch die zuvor von der *SegCAD*-Komponente durchgeführte Analyse möglich. *SegCAD* bearbeitet die Herausforderung, spezifische CAD-Teile aus Textbeschreibungen zu identifizieren. Dabei wird das 3D-Modell aus mehreren Blickwinkeln gerendert. Zudem werden 2D-Vision-Modelle angewendet, um Segmentierungsmasken zu generieren. Diese Segmentierungsmasken heben hervor, welche Pixel im gerenderten Bild zu Teilen gehören, die der gegebenen Textbeschreibung entsprechen. Sie werden anschließend in den 3D-Raum zurückprojiziert, wobei eine geometrische Adjazenzfilterung sicherstellt, dass nur tatsächlich verbundene Flächen ausgewählt werden. Dieser Filterprozess beginnt bei der am nächsten zur Kamera gelegenen Fläche und wählt rekursiv nur jene maskierten Flächen aus, die im 3D-Modell physikalisch benachbart sind. Dadurch wird die Einbeziehung nicht zusammengehöriger Flächen, die in der 2D-Ansicht lediglich verbunden erscheinen, verhindert. Dieser Ansatz ermöglicht eine Open-Vocabulary-Teilesegmentierung unter Nutzung vorab trainierter 2D-Modelle ohne spezialisierte 3D-CAD-Datensätze zu erfordern. Open-Vocabulary-Teilesegmentierung bezeichnet die Fähigkeit, spezifische Teile oder Komponenten eines 3D-Objekts, z. B. Löcher, Wellen oder Zahnräder in einem CAD-Modell, basierend auf natürlichsprachlichen Beschreibungen zu identifizieren und zu segmentieren, ohne dass das Modell auf einen vordefinierten, festen Satz von Teilekategorien trainiert werden muss.

Eine andere Lösung zur Bereitstellung von CAD-Informationen für ein LLM wird im *FlexCAD*-Framework von Zhang et al. [131] präsentiert. Ziel ist hier die Modifikation von CAD-Modellen basierend auf Benutzereingaben. Im Unterschied zu den bereits vorgestellten Ansätzen wird das CAD-Modell in strukturiertem Text als Eingabe für das LLM transformiert. *FlexCAD* konvertiert jedes geometrische Element in textuelle Token, wobei Kurventypen (Linie, Bogen, Kreis) direkt als Text repräsentiert werden und numerische Koordinaten von Dezimalzahlen in textuelle Token umgewandelt werden. Jede hierarchische Ebene (Kurve, Schleife, Fläche, Skizze, Extrusion) wird mit speziellen „End“-Token markiert, und Token aus feineren Ebenen werden konkateniert, um Repräsentationen gröberer Ebenen zu bilden. Dies erzeugt eine vollständige textuelle Beschreibung des CAD-Modells, die alle geometrischen und hierarchischen Informationen bewahrt und gleichzeitig von LLMs verarbeitet werden kann.

3.3. Methoden zur Nutzung von Foundation Models für die robotische Manipulation

In Abschnitt 3.1.1 wurden bereits KI-basierte Planungsmethoden vorgestellt. Da Foundation Models jedoch aufgrund ihres breiten Fähigkeitenspektrums, das über reine Montage- und Demontageaufgaben hinausgeht, besonders relevant sind,

werden diese in diesem Abschnitt gesondert betrachtet. Zunächst erläutert das folgende Unterkapitel die grundlegenden Methodiken in diesem Kontext. Die darauffolgenden Abschnitte konzentrieren sich auf Ansätze, die spezifische Herausforderungen hieraus adressieren.

3.3.1. Vision-Language-Action-Modelle in der Robotik

Die im Folgenden beschriebenen Ansätze lassen sich als Vision-Language-Action-Modelle (VLA) klassifizieren. Sie ermöglichen Robotern, visuelle Szenen und natürlichsprachliche Befehle zu verstehen und gleichzeitig entsprechende physische Aktionen auszuführen, indem sie über Foundation Models eine Brücke zwischen High-Level-Instruktionen des Menschen und der Low-Level-Robotersteuerung schlagen [132].

Eine verbindende Methode einer Überbrückung von menschlichen Instruktionen und robotischer Manipulation ist *SayCan* von Ahn et al. [133]. Dieser Ansatz ermöglicht durch die Berücksichtigung von robotischen Affordanzen das Grounding von LLMs. Hierzu werden die Planungsfähigkeiten von LLMs mit erlernten Wertefunktionen, welche die Machbarkeit von Fähigkeiten abschätzen, kombiniert. Robotische Affordanzen repräsentieren die Menge an Aktionen, die ein Roboter unter Berücksichtigung seines aktuellen Zustands und der Umgebungsbedingungen physikalisch ausführen kann. Diese werden durch erlernte Wertefunktionen quantifiziert, die die Wahrscheinlichkeit einer erfolgreichen Ausführung vorhersagen. Das *SayCan*-Framework berechnet einen gemeinsamen Wahrscheinlichkeitswert für jede potenzielle Aktion, indem es die bedingte Wahrscheinlichkeit des LLMs $P(Skill|Instruktion)$ mit einer Affordanzfunktion kombiniert, welche die Machbarkeit anhand der aktuellen Beobachtung schätzt. Dieser multiplikative Bewertungsmechanismus stellt sicher, dass ausgewählte Aktionen sowohl semantisch angemessen als auch physikalisch realisierbar sind.

Im Gegensatz zum Ansatz von *SayCan*, der LLMs über Schnittstellen mit externen Affordanzfunktionen verbindet, integriert *PaLM-E* von Driess et al. [134] kontinuierliche Sensormodalitäten direkt in den Embedding-Space des LLMs und schafft so eine End-to-End Embodied Multimodal Architecture. Das Framework integriert visuelle Beobachtungen, Zustandsschätzungen und andere Sensordaten als multimodale Token und kombiniert sie mit Text-Token zu multimodalen Sätzen. Dies ermöglicht dem LLM, linguistische und perzeptuelle Informationen innerhalb eines einheitlichen Repräsentationsrahmens zu verarbeiten. In Fällen wie der Szenenbeschreibung oder dem Grounded Question Answering stellt die Ausgabe direkt die Lösung der Aufgabe dar, während das Framework bei Planungs- oder Steuerungsaufgaben eine Sequenz von Skills basierend auf einer vordefinierten Menge möglicher Fähigkeiten generiert. *PaLM-E* ist in einen iterativen Algorithmus eingebunden, bei dem die Low-Level-Skills ausgeführt werden und *PaLM-E* basierend auf neuen Beobachtungen die Sequenz bei Bedarf neu plant.

Das *RT-2*-Modell von Zitkovich et al. [135] demonstriert, wie auf Web-Daten trainierte VLMs für die Robotersteuerung adaptiert werden können, indem Aktionen als Text-Token repräsentiert werden. Das ermöglicht den Transfer von Internetwissen auf die Robotik. *RT-2* baut auf vorab trainierten VLMs, PaLI-X und PaLM-E, auf, indem diese sowohl auf multimodalen Web-Daten als auch auf Datensätzen von Robotertrajektorien nachtrainiert werden. Dabei werden Roboteraktionen tokenisiert und innerhalb des bestehenden Vokabulars des Modells als natürlicher Sprachtext behandelt. Dieser Ansatz unterscheidet sich sowohl von *SayCan* als auch von *PaLM-E*, da er direkt Low-Level-Steuerbefehle anstelle von High-Level-Instruktionen ausgibt. Dadurch entstehen VLA-Modelle, die innerhalb einer einzigen Architektur zwischen Sprachaufgaben und Robotersteuerung wechseln können. Die Methodik nutzt das durch Training im Internet-Maßstab erworbene semantische Wissen, um die Generalisierung auf neue Objekte, Hintergründe und Umgebungen zu verbessern, die in den robotischen Trainingsdaten nicht vorhanden waren.

Aufbauend auf diesen multimodalen Ansätzen führt *VIMA* von Jiang et al. [136] ein vereinheitlichtes Prompting-Paradigma ein, das diverse Robotermanipulationsaufgaben durch strukturierte multimodale Prompts als Sequenzmodellierungsproblem behandelt. Während *PaLM-E* die direkte Integration kontinuierlicher Sensormodalitäten in LLMs demonstriert, fokussiert sich *VIMA* auf ein systematisches, multimodales Prompt-Design als universelle Schnittstelle für die Aufgabenspezifikation. Dies ermöglicht einem einzelnen Modell, Aufgaben wie die Manipulation von Objekten unter Berücksichtigung von Randbedingungen oder visuelles Schlussfolgern durch sorgfältig strukturierte Prompt-Templates zu bewältigen. Die Architektur nutzt ein Transformer-basiertes Encoder-Decoder-Design, bei dem ein vorab trainiertes T5-LLM diese multimodalen Prompts kodiert und ein Roboter-Controller autoregressiv Aktionen basierend auf den bereitgestellten Informationen generiert. *VIMA* verfolgt einen objektzentrierten Ansatz, bei dem Mask R-CNN zur Identifikation individueller Objekte in einer Szene verwendet wird, die anschließend separat durch einen Vision Transformer verarbeitet werden.

Während diese Ansätze verschiedene Strategien zur Integration von Sprachverständnis und Robotersteuerung aufzeigen, stoßen sie oft an Grenzen bezüglich der Cross-Embodiment-Generalisierung. Dieser Aspekt wird im Kontext des *Octo*-Frameworks von Team et al. [137] aufgegriffen. Cross-Embodiment-Generalisierung bezeichnet die Fähigkeit eines Modells, erlernte Fähigkeiten und Wissen über verschiedene Roboter-Hardwarekonfigurationen hinweg zu transferieren. *Octo* verwendet eine Transformer-First-Architektur mit drei Schlüsselkomponenten. Zunächst konvertieren Input-Tokenizer verschiedene Modalitäten in ein einheitliches Format, sodass Sprachinstruktionen zu Text-Token werden, während Kamerabilder in Sequenzen umgewandelt und von Transformatoren weiter verarbeitet werden. Anschließend verarbeitet ein Transformer-Backbone diese gemischten Sequenzen aus visuellen und linguistischen Informationen. Schließlich generieren Aktionsvorhersagenetzwerke aus diesen verarbeiteten Informationen Sequenzen von Roboterbewegungen. Dieses modulare Design ermöglicht eine

effiziente Anpassung an neue Robotersetups durch das Hinzufügen neuer Ein- oder Ausgabekomponenten, wobei das eigentliche Reasoning-Modell unverändert bleibt.

3.3.2. Methoden zum Grounding im Kontext von robotischer Manipulation

Der Begriff Grounding beschreibt die Fähigkeit eines sprachbasierten Foundation Models, ein Wort, für das eine Repräsentation erlernt wurde, mit seinem korrespondierenden Gegenstück in der realen Welt zu verbinden. Ein Beispiel dazu: Während Sprachmodelle aus Texten lernen können, dass „Norden“ und „Süden“ Gegensätze sind, haben sie keinen Zugriff auf die verankerte Bedeutung dieser Wörter – also die tatsächlichen räumlichen Richtungen, in die man sich in der physischen Welt bewegen müsste, wenn die Anweisung „gehe nach Norden“ erfolgt [138]. Dies ist für VLA-Modelle relevant, da beispielsweise physikalische Einschränkungen wie mögliche Kollisionen korrekt identifiziert werden müssen.

Ein Konzept, das ein Grounding ähnlich dem *SayCan*-Ansatz ermöglicht, ist *Grounded Decoding* von Huang et al. [139]. Im Gegensatz zu *SayCan* untersucht dieser Ansatz mehrere Arten von Grounding-Funktionen, anstatt sich ausschließlich auf mittels Reinforcement Learning trainierte Wertefunktionen zu verlassen. Es werden drei generelle Techniken zur Generierung von Grounding-Funktionen vorgeschlagen, die als spezifische Mechanismen über verschiedene Domänen hinweg implementiert werden. Diese Funktionen werden während des Token-Level-Decodings angewendet, um partielle Textsequenzen zu evaluieren. Das ermöglicht es dem System, physikalisch unmögliche, unsichere oder nicht bevorzugte Aktionen herauszufiltern. Die semantische Kohärenz und die Fähigkeiten zur langfristigen Planung des LLMs bleiben dabei erhalten.

Einen anderen Ansatz zum Grounding repräsentiert *SayPlan* von Rana et al. [140], der 3D-Szenengraphen (3DSG) zur Repräsentation der Umgebung nutzt. Ziel dieses Frameworks ist es, basierend auf einem 3DSG und einer natürlichsprachlichen Aufgabeninstruktion langfristige Pläne für einen mobilen Manipulator zu generieren und auszuführen. Hierbei werden zwei Stufen unterschieden. Zuerst erfolgt eine semantische Suche, um der Herausforderung zu begegnen, dass 3DSGs großer Umgebungen aufgrund von Token-Limitierungen zu umfangreich für die direkte Verarbeitung durch ein LLM sein können. Zudem erfordern die meisten Aufgaben nur eine kleine Teilmenge der gesamten Umgebungsinformationen. Daher kollabiert *SayPlan* zunächst den vollständigen 3D-Szenengraphen, um nur dessen oberste Hierarchieebene anzuzeigen, was die Token-Anzahl um ca. 80 % reduziert. Das LLM navigiert dann durch diese komprimierte Repräsentation mittels `expand`- und `contract`-API-Aufrufen, um relevante Bereiche selektiv zu explorieren. Dies wird durch In-Context-Learning-Beispiele und Chain-of-Thought-Prompting geleitet um zu bestimmen, welche Knoten manipuliert

werden sollen. Dieser Prozess dauert an, bis ein aufgabenspezifischer Teilgraph identifiziert ist, der nur die notwendigen Entitäten für die gegebene Instruktion enthält. Die zweite Stufe wird als iteratives Neuplanen bezeichnet. Sie nimmt den aufgabenspezifischen Teilgraphen und die Instruktion, um eine Sequenz von High-Level-Navigations- und Manipulationsaktionen zu generieren. Damit Halluzinationen des LLMs verhindert werden, werden zwei Mechanismen angewendet. Erstens werden Pfadplanungsaufgaben an Algorithmen wie Dijkstra delegiert. Zweitens nutzt das System die Selbstreflexionsfähigkeiten des LLMs, indem ein Szenengraph-Simulator verwendet wird, um generierte Pläne gegen Umgebungs-Constraints zu validieren. Bei Verletzungen liefert dieser ein textuelles Feedback, dass das LLM dazu veranlasst, seinen Plan iterativ zu verfeinern, bis eine machbare Lösung erreicht ist.

Während *SayPlan* das Grounding durch strukturierte räumliche Repräsentationen und hierarchisches Szenenverständnis adressiert, konzentriert es sich primär auf geometrische und räumliche Beziehungen zwischen Objekten und Orten. Eine erfolgreiche robotische Manipulation erfordert jedoch oft nicht nur das Wissen darüber, wo sich Objekte befinden und wie sie zu erreichen sind, sondern auch, woraus sie bestehen und wie sie gehandhabt werden müssen. Dieser Bedarf an einem Verständnis physikalischer Eigenschaften motiviert einen komplementären Ansatz von Gao et al. [141], der vorschlägt, physikalische Objektkonzepte direkt in VLMs für die robotische Manipulation zu integrieren. Dieses Konzept basiert auf dem Datensatz *PHYSOBJECTS*, der 39,6 Tsd. Crowd-Sourced- und 417 Tsd. automatisierte Annotationen physikalischer Konzepte für Bilder realer Haushaltsgegenstände enthält. Die physikalischen Konzepte umfassen Masse, Material, Verformbarkeit und Zerbrechlichkeit. Dieser Datensatz wird anschließend verwendet, um InstructBLIP nachzutrainieren. InstructBLIP ist ein VLM, das einen visuellen Encoder mit Googles FlanT5-XXL LLM kombiniert, um sowohl Bilder als auch Textinstruktionen zu verarbeiten. Das VLM wird trainiert, indem es Ja/Nein-Fragen zu physikalischen Eigenschaften von Objekten in Bildern beantwortet. Danach wird das resultierende Modell in einen LLM-basierten Roboterplaner integriert, bei dem das LLM das fein abgestimmte VLM zu physikalischen Konzepten von Objekten in einer Szene befragt, bevor Manipulationspläne generiert werden.

Die zuvor beschriebenen Ansätze befassen sich primär mit statischen Repräsentationen. Der *SeeDo*-Ansatz von Wang et al. [142] analysiert hingegen die Möglichkeit, Sprache mit zeitlichen Aktionssequenzen zu verbinden, in der Form wie sie in menschlichen Demonstrationsvideos ablaufen. Die Herausforderung besteht darin, Sprache auf dynamische Prozesse zu grounden, bei denen sich räumliche Beziehungen im Zeitverlauf durch sequenzielle Manipulationsaktionen entwickeln. Dieses temporale Grounding-Problem erfordert, dass das System die Objektkonsistenz über Video-Frames hinweg aufrechterhält und versteht, wie Aktionen kausal mit sich ändernden räumlichen Konfigurationen verbunden sind, eine Herausforderung, die statische Ansätze nicht adressieren können. Zur Realisierung nutzt *SeeDo* drei Module: ein Keyframe-Auswahlmodul, ein visuelles Wahrnehmungsmodul und ein VLM-Reasoning-Modul. Zuerst werden die ein-

zelnen Frames des Demonstrationsvideos an das Keyframe-Auswahlmodul übergeben. Dieses adressiert Beschränkungen der VLM-Kontextlänge, indem es Heuristiken zur Handgeschwindigkeit nutzt, um kritische Frames zu identifizieren. Anschließend werden die ausgewählten Frames an das visuelle Wahrnehmungsmodul weitergeleitet, das einen Open-Vocabulary-Objektdetektor verwendet, um Bounding Boxes für sichtbare Objekte im ersten Frame zu identifizieren. Diese Bounding Boxes dienen als Input-Prompts für das Segment Anything Model 2 (SAM 2), einem Videosegmentierungsmodell, das Objekte über mehrere Frames hinweg verfolgt. Schließlich nutzt das VLM-Reasoning-Modul CoT-Prompting mit GPT-4o, um Referenzobjekte zu identifizieren und Task-Planning-Schritte als finale Ausgabe zu generieren.

Während die vorherigen Ansätze das Grounding auf räumliche Beziehungen, physikalische Eigenschaften oder zeitliche Sequenzen fokussieren, thematisiert *VLM-MSGraph* von Li et al. [143] domänenspezifisches Grounding im Kontext der industriellen robotischen Montage. Im Gegensatz zu universellen Mechanismen führt *VLM-MSGraph* eine spezialisierte Lösung ein, die Montageinstruktionen in strukturiertem Fertigungswissen grounded. Dies wird umgesetzt durch die Kombination von High-Level-Montagesequenzlernen mittels mehrerer VLM-Agenten mit einem Low-Level-3D-Raumgedächtnis für geometrie-bewusste Manipulation. Es wird ein multi-hierarchischer Szenengraph erstellt. Auf der hohen Ebene nutzen mehrere VLM-Agenten CoT-Reasoning und Retrieval-Augmented Generation (RAG), um visuelle Eingaben zu analysieren und Montagesequenzen als Triplets in der Form *Objekt1, Montagebeziehung, Objekt2*, in einem dreistufigen Prozess zu generieren. Parallel dazu erfasst das System auf der niedrigen Ebene räumliche 3D-Beziehungen durch RGB-D-Beobachtungen mittels Open-Vocabulary-Perzeption, erstellt ein 3D-Objektinstanz-Gedächtnis durch Aggregation von Punktwolken und ermöglicht geometrie-bewusste Manipulation durch Berechnung optimaler 6D-Posen. Der Roboter führt die Montageaufgaben aus, indem er die sequenziellen Instruktionen aus dem MSGraph in präzise Aktionen übersetzt und so die Brücke zwischen High-Level-Aufgabenplanung und Low-Level-Geometriemanipulation schlägt.

3.3.3. Methoden zur Integration von Neuplanungsansätzen für die robotische Manipulation

Eine weitere wichtige Fähigkeit für ein System beim Einsatz in der Robotik ist die Fähigkeit, auf Veränderungen in der Umgebung oder die Verfügbarkeit neuer Informationen zu reagieren. Im Kontext von Foundation Models wird dies als Neuplanung (engl. *Replanning*) bezeichnet. Im folgenden Abschnitt werden verschiedene Ansätze für die Realisierung von Replanning vorgestellt.

Ein Konzept zur Nutzung von visuellem Feedback für die Anpassung von Langzeitplänen ist *VILA* von Hu et al. [144]. Es nimmt sowohl eine visuelle Beobachtung als auch High-Level-Sprachinstruktionen als Eingabe und generiert eine

Sequenz von Textaktionen für robotische Manipulationsaufgaben, unter der Annahme, dass alle notwendigen primitiven Skills verfügbar sind. Als VLM wird GPT-4V von OpenAI verwendet, welches für jeden Schritt die aktuelle visuelle Beobachtung, die Sprachinstruktion und die Historie der ausgeführten Aktionen erhält. Basierend auf diesen Informationen wird durch CoT-Reasoning ein Schritt-für-Schritt-Plan generiert. Anschließend wird der erste Schritt ausgewählt und durch den entsprechenden primitiven Skill ausgeführt, woraufhin die Aktion zur Historie der beendeten Pläne hinzugefügt wird. *VILA* implementiert Neuplanung durch einen kontinuierlichen geschlossenen Regelkreis, bei dem das System nach Ausführung jedes Schrittes eine neue visuelle Beobachtung erfasst und das VLM mit den aktualisierten Informationen erneut abfragt. Dieser schrittweise Ansatz ermöglicht es dem VLM, sowohl als Szenenbeschreiber als auch als Erfolgsdetektor zu fungieren. Dadurch kann der Roboter seine Strategie dynamisch auf Basis von Echtzeit-visuellem Feedback anpassen.

Aufbauend auf *VILA*s Ansatz des visuellen Feedback-basierten Neuplanens erweitert *REPLAN* von Skreta et al. [145] das Konzept durch die Einbindung autonomer Verifikationsmechanismen und mehrstufigen Reasonings. Dies ermöglicht eine robustere Handhabung komplexer Langzeitaufgaben, die sowohl strategische High-Level-Planung als auch Low-Level-Ausführungsüberwachung erfordern. Das *REPLAN*-Framework nutzt fünf verschiedene Module, die High-Level- sowie Low-Level-Planung, Umgebungswahrnehmung und Bewegungssteuerung abdecken. Zunächst konvertiert der High-Level-LLM-Planer die nutzerspezifizierte Aufgabe in eine Liste von Teilaufgaben. Ein zusätzliches VLM wird für das Grounding eingesetzt, indem es spezifische visuelle Fragen des High-Level-Planers zu Objektzuständen beantwortet. Diese Aufgabenteilung hebt die rein sprachlichen Reasoning-Fähigkeiten des LLMs hervor, während die Bildinterpretationsfähigkeit des VLMs zur Wahrnehmung der physischen Welt genutzt wird. Die High-Level-Teilaufgaben werden im weiteren Verlauf von einem Low-Level-LLM-Planer in konkrete, natürlichsprachlich formulierte Bewegungspläne konvertiert. Diese werden in Belohnungsfunktionen übersetzt, welche die gewünschte Roboterbewegung repräsentieren und vom Bewegungscontroller zur Generierung tatsächlicher Aktionen genutzt werden. Ein Verifizierer prüft schließlich die Ausgaben beider Planer und des Wahrnehmungs-Moduls. Neuplanung wird in diesem Framework durch ein mehrstufiges Fehlererkennungs- und Wiederherstellungssystem realisiert: Scheitert die Ausführung oder erkennt das VLM Fehler aus visuellen Beobachtungen, generiert der High-Level-Planer Korrekturmaßnahmen. Sollte das Gesamtziel nach Abschluss aller Teilaufgaben und Neuplanungs-Versuche nicht erreicht sein, reflektiert der High-Level-Planer über vergangene Erfahrungen und schlägt einen komplett neuen Plan vor. Dabei wird das Gedächtnis an fehlgeschlagene Versuche genutzt, um repetitive Fehler zu vermeiden.

Während *REPLAN* hierarchische Neuplanungsfähigkeiten demonstriert, ist es belastet durch die Limitierungen im räumlichen Verständnis des VLMs sowie in der Konsistenz des LLMs und der LLM-VLM-Interaktion [145]. Um diese Herausforderungen zu bewältigen, führt *ReplanVLM* von Mei et al. [146] einen An-

satz der doppelten Fehlerkorrektur auf Basis von GPT-4V ein. Das Framework verwendet einen *Decision Bot* zur Erstellung initialer Aufgabenpläne und zugehörigem Ausführungscode, der anschließend von einem *Inner Bot* auf korrekte Formatierung, Konsistenz und logische Stichhaltigkeit geprüft wird. Dieser interne Fehlerkorrekturmechanismus verhindert Halluzinationen, indem er dem Decision Bot eine Rückmeldung für das Neuplanen gibt, sofern Fehler erkannt werden. Nach erfolgreicher Verifikation und Codeausführung validiert ein *External Bot* die Aufgabenerfüllung durch Vergleich der aktuellen Umgebung mit dem Initialzustand und löst eine Neuplanung aus, falls Anforderungen unerfüllt sind. Dieser externe Mechanismus handhabt spezifisch plötzliche Umweltveränderungen oder Systemfehler bereits während der Ausführung.

3.4. Affordanzen

In der Robotik sind Affordanzen ein wertvolles Instrument zur Erfassung von Handhabungsmöglichkeiten. Neben Planern, die auf Basis dieser Informationen agieren, gewinnen besonders Systeme an Bedeutung, welche die Affordanzen direkt aus Bilddaten extrahieren. Dies ist insbesondere in Szenarien essenziell, in denen die Interaktionsmöglichkeiten der Umgebung nicht vorab bekannt sind.

3.4.1. Detektion von Affordanzen

Um einem Robotersystem mögliche Interaktionen zwischen dem Roboter und Objekten oder auch zwischen den Objekten untereinander klar erkennbar zu machen, sind Affordanzen eine Möglichkeit. Aus Bilddaten lassen sich diese automatisiert, etwa durch den Einsatz von KI, ableiten. Auf diese Weise lässt sich eine autonome Planungs- und Verarbeitungskette aufbauen. Im Folgenden werden verschiedene Verfahren zur Extraktion von Affordanzen aus Bilddaten vorgestellt.

Do et al. [147] präsentieren das *AffordanceNet*, einen Ansatz, um mittels eines CNNs eine Objekt- und Affordanz-Erkennung für Haushaltsgegenstände durchzuführen. Die Architektur des CNNs basiert auf Mask R-CNN. So wird neben dem RPN und dem RoI-Alignment auch der Zweig zur Objekterkennung übernommen. Als Backbone wird hier VGG-16 [148] verwendet. Ähnlich der Instanzsegmentierung im Mask R-CNN, wird hier die semantische Segmentierung zur Affordanzerkennung auf den Region Proposals des RPNs durchgeführt. Um eine feinere Segmentierung zu ermöglichen, werden die 7×7-Feature Maps der Region Proposals in drei Schritten hochskaliert. Auf den hochskalierten 224×224-Feature Maps wird dann mittels Softmax-Aktivierung eine Segmentierung in die Affordance-Klassen erzeugt.

Auch Nguyen et al. [149] verwenden einen regionsbasierten Ansatz. Sie verwenden in diesem Fall R-FCN [150], um Region Proposals zu generieren. Auf diese Region Proposals schließt sich dann VGG-16 zur Affordanz-Erkennung an. Dabei wird kein Upsampling durchgeführt. Als Nachverarbeitung folgt auf die Segmentierung ein Dense Conditional Random Fields (Dense CRF) [151]. CRFs lernen Zusammenhänge zwischen Pixeln und den zugewiesenen Labels. Daraus können semantische Zusammenhänge zwischen ähnlichen Pixeln hergestellt werden. So wird die Segmentierung des CNNs insbesondere an Objektkanten eindeutiger.

Ebenfalls von Nguyen et al. [152] wird ein anderes CNN präsentiert. Es erkennt wie das *AffordanceNet* Affordanzen für Haushaltsgegenstände. Dieses Netz arbeitet im Gegensatz zum *AffordanceNet* auf RGB-D-Bilddaten. Dazu wird der UMD-Datensatz verwendet. Das Netz ist als symmetrische Encoder-Decoder-Architektur ohne Skip Connections konzipiert. Der Encoder wird durch VGG-16 implementiert. Der Decoder verwendet Upsampling-Schichten. Untersucht wird hier auch die Kodierung der Tiefeninformation. Hier wird das Tiefenbild wahlweise direkt als Graustufenbild oder als HHA-Kodierung in das Netz eingespeist. Die HHA-Kodierung verwendet die Tiefenbilder, um Tiefenunterschiede in horizontaler Bildrichtung zu beschreiben. Des Weiteren beschreibt die HHA-Kodierung die Höhe über dem Boden und der Winkel zwischen den Oberflächennormalen und der Gravitationsrichtung. Die Ergebnisse zeigen allerdings, dass die HHA-Kodierung keinen bedeutenden Vorteil erbringt.

Das Framework von Zhao et al. [153] setzt die Affordanzendetektion mit einem anderen Konzept um. Das vorgeschlagene Netz arbeitet als Encoder-Decoder-Architektur. Dabei wird im Decoder ein Modul eingebaut, welches Zusammenhänge zwischen Affordanzen und Objektklassen erlernt. Dafür werden die Affordanzklassen angegeben, aus denen sich die Objektklassen zusammensetzen. So werden die Affordanzklassen abhängig von den erkannten Objektklassen unterschiedlich stark gewichtet.

Abschließend präsentieren Sawatzky et al. [154] einen Ansatz, der auf Datensätzen arbeiten kann, die nur mit Keypoints annotiert sind. Keypoints bezeichnen punktuelle Bereiche des Objektes. Diese werden mit den entsprechenden Affordanzen annotiert. Netze, die auf solchen schwach annotierten Datensätzen segmentieren sollen, müssen den Objektbereichen auf Basis dieser Keypoints die passenden Affordanzen zuordnen. In ihrem Ansatz wird das ResNet101 bzw. VGG-16 zur Segmentierung verwendet. Die Genauigkeit der Vorhersagen ist im Vergleich zu vollständig annotierten Bildern geringer. Allerdings ist die Erstellung des Datensatzes effizienter.

3.4.2. Affordanzen in der robotischen Montage und Demontage

Nach der Detektion der Affordanzen können diese im Anschluss verwendet werden, um verschiedenste Aufgabenstellungen im Bereich der Montage und Demontageplanung bzw. der Ausführung zu unterstützen. Der folgende Abschnitt stellt hierzu relevante Arbeiten vor.

In der Robotermontage spielen Affordanzen eine zentrale Rolle, die oft mit der von semantischen Merkmalen vergleichbar ist. Beßler et al. [155] stellen für die Planung von Montageschritten einen Ansatz vor, der darauf abzielt, Montagepläne unter direkter Verwendung von gesammeltem Montagewissen zu generieren. Zu diesem Zweck wird das gesamte Wissen als Ontologie in der *Web Ontology Language (OWL)* modelliert, wobei die Wissensbasis *KNOWROB* als Informationsquelle dient. Die erweiterte Ontologie beschreibt mechanische Komponenten, die notwendigen Montageverbindungen sowie die zugehörigen Affordanzen, welche speziell im Kontext von Montage- und Greifvorgängen betrachtet werden. Durch die Integration eines Open-World-Reasoning-Ansatzes in diese Ontologie befähigen Beßler et al. das System dazu, logische Schlüsse zu ziehen und Montageschritte selbst dann erfolgreich zu planen, wenn das verfügbare Wissen über die Umgebung unvollständig ist.

Ebenfalls auf Basis von OWL präsentieren Xi et al. [156] in ihrer Arbeit einen Ansatz, der primär darauf ausgelegt ist, die hohe Komplexität von Planungsprozessen in der Montage zu reduzieren. Die Kernidee liegt hierbei in der automatischen Erkennung und Definition von Unterbaugruppen. Das vorgeschlagene Framework geht dabei in mehreren Schritten vor. Zunächst wird ein semantisches Modell der Baugruppe erstellt, in dem die Beziehungen zwischen den einzelnen Komponenten ontologisch abgebildet werden. Anschließend werden Unterbaugruppen identifiziert, indem die semantischen Beziehungen zwischen den Montagekomponenten sowie deren nicht-geometrische Attribute, wie etwa Material oder Funktion, analysiert werden. Um diese Informationen nutzbar zu machen, kommt ein Clustering-Algorithmus zum Einsatz, der die Komponenten logisch gruppiert und so die Planungsebene für den Roboter vereinfacht.

Ein auf Bilddaten basiertes Vorgehen wird von Isume et al. [157] mit ihrem Ansatz zur Planung von Montagesequenzen vorgelegt. Ausgangspunkt ist hierbei ein vollständiges 3D-Modell der finalen Baugruppe, aus dem zunächst eine Liste aller benötigten Komponenten extrahiert wird. Für jede dieser Komponenten werden im Anschluss die spezifische Form sowie die zugehörigen Affordanzen algorithmisch bestimmt. In der realen Ausführungsphase wird die Roboterszene dann nach potenziellen Bauteilen durchsucht, die für den Montageprozess infrage kommen. Die finale Auswahl der Komponenten erfolgt dabei durch einen Abgleich der detektierten Affordanzen und der visuellen Ähnlichkeit zu den Referenzteilen aus dem ursprünglichen 3D-Modell.

Ergänzend zu den reinen Montageansätzen untersuchen Vongbunyong et al. [158] den gezielten Einsatz von Affordanzen im Bereich der kognitiven Demontageautomatisierung. Das Framework ist darauf ausgelegt, unterschiedliche Verbindungselemente wie Schrauben, Clips oder Nieten automatisiert zu erkennen und voneinander zu unterscheiden. Anstatt starrer Ablaufpläne werden Affordanzen genutzt, um dynamisch zu entscheiden, welche spezifische Demontageaktion, beispielsweise das Drehen eines Schraubendrehers oder das Ansetzen eines Hebels, für eine erkannte Komponente physikalisch am plausibelsten ist. Dies ermöglicht dem Roboter eine hohe Flexibilität, da er so auch in der Lage ist, mit Produkten zu interagieren, für die keine expliziten CAD-Daten oder Demontageanleitungen vorliegen.

Die Arbeit von Chu et al. [159] konzentriert sich auf den Bereich der sensorgestützten Montage und zeichnet sich durch den Einsatz von Deep Learning für die Detektion von sogenannten Füge-Affordanzen aus. Im Gegensatz zu klassischen Verfahren werden keine vordefinierten CAD-Modelle verwendet, sondern das System ist darauf trainiert, direkt aus visuellen Daten die geometrischen Merkmale zweier Bauteile zu lernen, die eine Passung ermöglichen. Das neuronale Netz identifiziert dabei relevante Bereiche an den Objekten, die eine erfolgreiche Montage implizieren. Das erlaubt dem Roboter, Fügestellen auch in unstrukturierten Umgebungen robuster zu lokalisieren und die Bahnplanung für den Fügeprozess adaptiv an die visuelle Wahrnehmung anzupassen.

Einen anderen Schwerpunkt setzen Myers et al. [160], indem sie die Rolle von Werkzeug-Affordanzen bei der Ausführung von Manipulationsaufgaben thematisieren. Ihr Ansatz analysiert die geometrischen Eigenschaften von Bauteilen im Arbeitsraum, um deren Eignung als Werkzeug für eine bestimmte Fügeoperation zu bewerten. So wird beispielsweise erkannt, ob ein Bauteil einen greifbaren Stiel und ein schweres Ende besitzt und somit auch faktisch als Hammer genutzt werden kann. Diese semantische Analyse befähigt das Robotersystem dazu, tatsächliche, aber auch improvisierte Werkzeuge in der Umgebung zu erkennen oder aus einem Set von Werkzeugen dasjenige auszuwählen, das basierend auf seinen funktionalen Eigenschaften am besten für den aktuellen Fügeschritt geeignet ist.

Auch generative KI-Modelle werden bereits im Kontext von Affordanzen eingesetzt. So integrieren Li et al. [161] generative KI-Modelle, um die Lücke zwischen sprachlichen Anweisungen und physischen Handlungen zu schließen. Das Chain-of-Affordance Vision-Language-Action (CoA-VLA) Framework nutzt große Vision-Language-Modelle, um komplexe und teils vage Montageanweisungen in eine Sequenz konkreter Affordanz-Schritte zu zerlegen. Das System lernt hierbei, eine Kette aus Bauteil-, Greif- und Bewegungs-Affordanzen zu bilden, die logisch aufeinander aufbauen. Dadurch ist der Roboter in der Lage, abstrakte Befehle wie „Montiere das Rad“ in kollisionsfreie Bewegungspfade zu übersetzen, ohne dass diese explizit vorprogrammiert werden müssen.

Um besonders im Rahmen der Mensch-Roboter-Kollaboration die in Hinblick auf Effizienz notwendige Flexibilität zu ermöglichen, zielt der Ansatz von Wang

et al. [162] darauf ab, durch den Einsatz von affordanzbasiertem Schlussfolgern (engl. *Affordance Reasoning*) eine dynamische Aufgabenübernahme mittels Roboter zu ermöglichen. Per Überprüfung der notwendigen Werkzeug-Affordanzen des aktuellen Bauteils kann das System erkennen, ob das Werkzeug momentan verfügbar ist oder durch den Menschen belegt ist. Dies ermöglicht, Demontagesequenzen flexibel und in Echtzeit anzupassen. Aufgrund der semantischen Verknüpfung von Bauteileigenschaften mit den aktuell verfügbaren Werkzeug-Affordanzen kann das System alternative Lösungswege berechnen. Dadurch steigert sich nicht nur die Effizienz des gesamten Arbeitsprozesses, da es zu weniger Stillstandszeiten des Roboters kommt, sondern es macht den Vorgang auch deutlich resilienter gegenüber unvorhergesehenen Störungen oder Änderungen im Arbeitsablauf.

3.5. Identifikation des Forschungsdesiderats

In diesem Abschnitt werden themenbezogene Forschungsbedarfe auf der Grundlage der aktuellen Forschungslage benannt. Dazu werden Forschungsarbeiten aus den für diese Arbeit relevanten Bereichen zusammengefasst und analysiert. Dadurch wird einerseits deutlich, welche Themenbereiche intensiv untersucht worden sind, anderseits zeigt sich, wo ein eher unvollständiger Wissensstand besteht. Die daraus resultierenden Erkenntnisse werden konsolidiert mit dem Ziel, Forschungslücken aufzuweisen und zu bearbeiten. Damit wird zugleich ein Beitrag zur Entwicklung neuer Konzepte geleistet. Die Grundlage für ein solches neuartiges Konzept wird in Kapitel 4 vorgestellt.

In der automatischen Montage- und Demontageplanung bieten KI-basierte Planungsmethoden durch spezialisierte algorithmische Fähigkeiten oftmals effizientere Lösungen als traditionelle Ansätze. Graph-Learning-Methoden zeichnen sich dadurch aus, strukturelle Beziehungen zwischen Bauteilen zu erfassen und latente Montageregeln aus heterogenen Repräsentationen zu erlernen. Reinforcement-Learning-Ansätze demonstrieren Anpassungsfähigkeit, indem sie optimale Montagestrategien durch Interaktion mit der Umgebung und belohnungsbasierte Optimierung erlernen. Auch Foundation Models zeigen vielversprechendes Potenzial, vorab trainiertes Wissen aus großen Datensätzen zu nutzen, um die Montage- und Demontageplanung ohne umfangreiches aufgabenspezifisches Training zu ermöglichen. Ebenso hat die Forschung im Bereich der Physiksimulation und Kollisionsvalidierung robuste Frameworks etabliert, die geeignet sind, die Montagemachbarkeit durch systematische Kollisionserkennung und Modellierung physikalischer Randbedingungen zu ermöglichen. CAD-basierte Ansätze stellen hochentwickelte Methoden dar, die physikalische und geometrische Randbedingungen extrahieren können und gleichzeitig auf parametrische Konstruktionsinformationen zugreifen.

Aufbauend auf diesen CAD-basierten Ansätzen demonstrieren Konzepte zur Interaktion zwischen CAD und Foundation Models verschiedene Möglichkeiten.

Zum einen gelingt es, Foundation Models für die Erstellung von CAD-Daten durch Code-Generierung, parametrische CAD-Generierung und Unterstützung des Designprozesses einzusetzen. Zum anderen ermöglicht die Nutzung von CAD-Modellen als Eingabe für Foundation Models Methoden wie Frage-Antwort-Systeme, Bauteilsegmentierung und eine strukturierte textuelle Repräsentation geometrischer Informationen zu realisieren.

Ergänzend zu diesen CAD- und Foundation-Model-Schnittstellen haben Foundation Models für die robotische Manipulation umfassende Frameworks etabliert, die mehrere kritische Bereiche abdecken. VLA-Modelle überbrücken erfolgreich die Kluft zwischen High-Level-Instruktionen des Menschen und Low-Level-Robotersteuerung durch verschiedene architektonische Ansätze, vom affordanzbasierten Grounding bis zur multimodalen Integration. Grounding-Methoden stellen das Bewusstsein für physikalische Randbedingungen durch Wertefunktionen, Szenengraphen, Verständnis physikalischer Eigenschaften und zeitliche Aktionssequenzen sicher. Neuplanungs-Ansätze bieten robuste Mechanismen zur Handhabung von Umgebungsänderungen und Ausführungsfehlern durch visuelles Feedback, hierarchische Fehlerbehebung und prädiktive Planrevision.

Die Detektion von Affordanzen findet primär im Haushaltsbereich statt und hat dort vorkommende Gegenstände im Fokus. Ansätze nutzen dabei neben klassischen CNN-basierten Methoden inzwischen auch Encoder-Decoder-Konzepte bzw. Transformer. Um die Genauigkeit und Zuverlässigkeit der Detektionen zu erhöhen, wird zum Training an dieser Stelle häufig auf vollständig annotierte Daten zurückgegriffen.

Für die Verwertung der Affordanzinformationen in Montage- und Demontageszenarien greifen Systeme u. a. auf Ontologien und bildbasierte Verfahren zurück. Hierbei zeigen Affordanzen einen Weg für eine größere Flexibilisierung etwa beim Lösen von Verbindungselementen oder Finden von alternativen Montageschritten oder Werkzeugen auf. Unter Verwendung von generativen KI-basierten Verfahren sind Affordanzen auch für Aufgaben wie Planverfeinerung, von einem symbolischen Plan in einen robotischen Handlungsplan, verwendbar.

Die Analyse der Domänen offenbart deutliche Forschungsdesiderate in den Bereichen von Schnittstellen. Während jedes Feld anspruchsvolle individuelle Ansätze entwickelt hat, besteht ein erkennbarer Mangel an einem vereinheitlichten Framework, das die Montageplanung, CAD-Integration und die Fähigkeiten von Foundation Models sowie Affordanzen systematisch in einem einzigen kohärenten System kombiniert. Aktuellen Ansätzen in der robotischen Manipulation fehlt die Nutzung von CAD-Daten als Grounding für Instruktionen von Foundation Models. Des Weiteren verwendet kein bestehendes Framework Wissensgraphen, um CAD-Daten für Large Language Models verfügbar zu machen, trotz des signifikanten Potenzials, solche strukturierten Repräsentationen für die Extraktion von geometrischen Randbedingungen als auch für die Einbindung der weiteren Konstruktionsinformationen zu verwenden. Schließlich konzentrieren sich existierende Affordanz-Ansätze häufig auf einen Informationszugewinn zur

Laufzeit oder erfordern detaillierte Vorabinformationen wie einfache Montageanweisungen, statt bereits im initialen Planungsprozess auf die geometrischen und semantischen Informationen der Bauteile zuzugreifen.

Diese identifizierte Lücke motiviert die Entwicklung eines ganzheitlichen Frameworks, welches für die Erzeugung einer Bauteilsequenz ein Dual-Model-Verifikationsframework verwendet und anschließend Affordanzen für eine Überführung der Sequenz in einen symbolischen Plan nutzen kann.

4. Ansatz für einen generativen Montageplanungsprozess

Die Diskussion zum aktuellen Stand der Forschung hat deutlich gezeigt, dass in verschiedenen Ansätzen bereits unterschiedliche Arten von künstlicher Intelligenz für die Planung von Montage und Demontage verwendet werden. Demnach bestehen die größeren Herausforderungen in der Verwendung von komplexen Eingangsdaten. Im Folgenden wird das Konzept zu dieser Arbeit erläutert, wobei eines der Hauptziele ist, komplexe Eingangsdaten direkt für die Planung mit KI-Methoden zu nutzen. Auf die zentralen Bestandteile des Konzeptes wird nachfolgend detailliert eingegangen.

4.1. Einleitung

Das hier vorgeschlagene Konzept umspannt den gesamten Planungsprozess bis zur Generierung eines symbolischen Plans für die Montage oder Demontage von Baugruppen. Konkret bedeutet dies, dass zu Beginn ein Modell der zu manipulierenden Baugruppe in das System eingegeben wird und dieses in eine Füge-/Zerlegesequenz mit dazugehörigen Robotertätigkeiten transformiert wird. Die entsprechende Sequenz ist dabei physikalisch realisierbar, da weder Durchdringungen noch Kollisionen zugelassen werden. Die Sequenz ist jedoch nicht auf ein bestimmtes Szenario oder einen bestimmten Aspekt hin optimiert, z. B. die minimale Anzahl von Werkzeugwechseln. Das entstandene Konzept ist in Abbildung 4.1 dargestellt und beinhaltet vier zentrale Teile.

4.2. Datengenerierung auf Basis von vorherigen Baugruppen

Eine Herausforderung beim Einsatz von Methoden der KI stellt die zur Verfügung stehende Datenmenge dar. Diese muss je nach eingesetzter KI-Methodik zum einen umfangreich genug für ein erfolgreiches generalisierendes Training sein, zum anderen alle für das Problem relevanten Daten enthalten. Auch wenn für die Montage einige wenige Datensätze öffentlich verfügbar sind, waren diese im Rahmen der Arbeit aufgrund ihrer Nachteile nicht geeignet. Beispielsweise

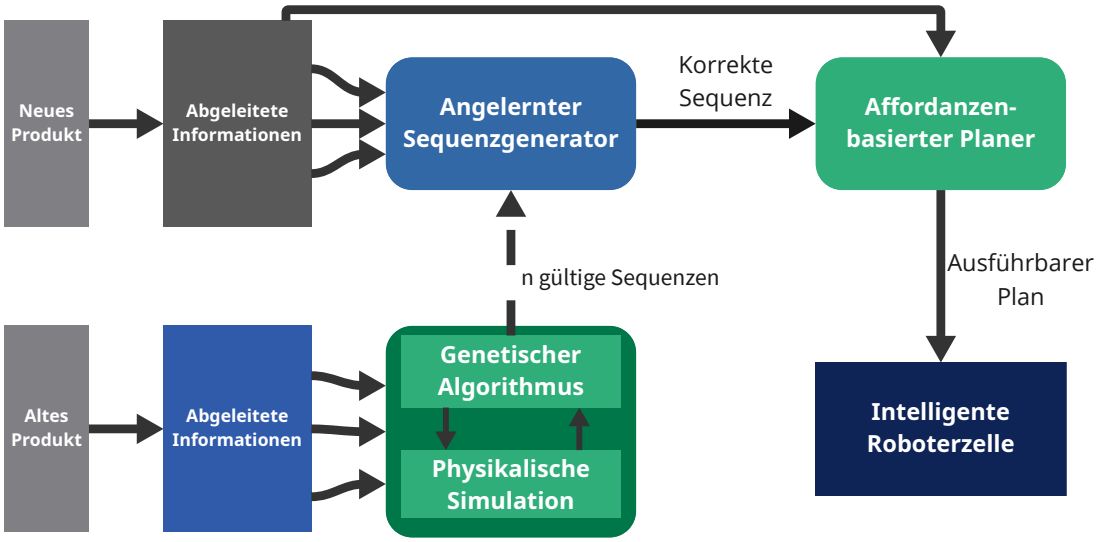


Abbildung 4.1.: Generelles Konzept des entwickelten Frameworks.

war der Umfang der Datensätze häufig sehr gering (meist 10–100 Baugruppen pro Klasse) oder die Daten waren stark vorverarbeitet, etwa durch ein vereinfachtes bzw. reduziertes Oberflächenmodell, sodass sie keine ausreichende Informationstiefe besaßen.

Aus diesen Gründen ist der erste Teil des Konzeptes ein Ansatz zur automatisierten Generierung von Montagesequenzen für eine Baugruppe. Diese Komponente basiert auf einem adaptierten genetischen Algorithmus. Hierzu wird ein Oberflächenmodell, welches aus einem CAD-Modell abgeleitet werden kann, als Ausgangspunkt gewählt. Der genetische Algorithmus optimiert die einzelnen Individuen auf Basis der Kriterien Durchdringungsfreiheit, Lokalität und Kippstabilität. Anders als klassische genetische Algorithmen generiert dieser Ansatz dabei nicht nur einen optimalen Kandidaten, sondern zielt darauf ab, eine große Anzahl an möglichen Montagesequenzen für eine gegebene Baugruppe zu erzeugen.

Da diese Funktionskomponente ausschließlich zur Generierung von Trainingsdaten eingesetzt wird, ist sie nicht unmittelbar Bestandteil der Planungsprozesskette. Es wäre an dieser Stelle wünschenswert, wenn diese Daten bereits durch den späteren Anwender zur Verfügung gestellt werden könnten. Auch wenn diese Komponente nur sekundär Teil der Planungsprozesskette ist, wurde untersucht, wie insbesondere die Überprüfungsalgorithmen, die in den genetischen Algorithmus integriert sind, als zusätzliche Absicherungskomponente genutzt werden könnten. Für eine genauere Erläuterung des möglichen Vorgehens wird an dieser Stelle auf [Plasberg et al., 2023] verwiesen.

4.3. Bestimmung korrekter Sequenzen

Ein wichtiger Schwerpunkt des Planungskonzeptes ist der Sequenzgenerator. Anders als klassische Sequenzgeneratoren, welche auf kombinatorischen Ansätzen beruhen, wird in dem vorliegenden Konzept ein KI-basierter Sequenzgenerator verwendet. Ein zentraler Aspekt ist dabei, verschiedene KI-Ansätze auf den Montagekontext anzuwenden und zu evaluieren, welche Vor- und Nachteile die jeweiligen Ansätze bieten. Folgende Ansätze wurden hierzu untersucht und verglichen:

- **Sequenzbestimmung mithilfe eines Sequenz-zu-Sequenz-Ansatzes:** In diesem Ansatz findet die Sequenzbestimmung mithilfe eines Transformer-basierten, angepassten Sequenz-zu-Sequenz-Netzwerkes (engl. *Sequence-to-Sequence*, Seq2Seq) statt. Hierzu wird die Baugruppe zunächst voxelisiert und anschließend eine mögliche Teilreihenfolge durch das Netzwerk ermittelt. Hierbei beeinflussen vor allem die in der Architektur integrierten *Gated Recurrent Units* (GRU) die Verallgemeinerungsfähigkeit des Netzwerkes. Im Vergleich zu einem nicht angepassten Sequenz-zu-Sequenz-Ansatz ermöglicht die umgesetzte Lösung, mit mehreren gleichartigen Bauteilen zu arbeiten. Zusätzlich wird pro Sequenzschritt nicht nur die Geometrie des zu verwendenden Bauteils erzeugt, sondern auch eine zuvor festgelegte eindeutige ID verwendet, sodass eine direkte Zuordnung von Sequenzschritt zum konkreten Bauteil und der Bauteilposition möglich ist.
- **Sequenzbestimmung mithilfe eines Graph Transformers:** Die Generierung einer Sequenz mittels einer Graph-Transformer-Architektur ist ein weiterer untersuchter Ansatz. Hierbei stellen die einzelnen Bauteile der Baugruppe die Knoten des Graphen dar, die Kanten repräsentieren mögliche Fügeoperationen zwischen einzelnen Bauteilen. Bei der Inferenz prädiziert das Modell basierend auf den Knoten die für die Montage relevanten Kanten und legt fest, in welcher Reihenfolge der Graph durchlaufen werden muss. Im Vergleich zum Sequenz-zu-Sequenz-Ansatz benötigt der Graph Transformer vorab keine Transformation der Eingangsdaten (wie Voxelisierung). Dadurch kann dieser auch mit hochaufgelösten Modellen arbeiten.
- **Sequenzbestimmung mithilfe einer LLM/VLM-Kombination:** Auch Large-Language-Modelle werden auf ihre Anwendbarkeit im Rahmen von Montageplanung hin untersucht. Besonderer Fokus liegt auf der Verwendung von komplexen Informationen, wie sie in einem CAD-Modell vorliegen. Um die Informationen dem Netzwerk verfügbar zu machen, werden sie in einen Wissensgraphen übertragen, der anschließend vom Netzwerk verarbeitet werden kann. Aufgrund des umfangreichen Basiswissens von LLMs und ihrer Schlussfolgerungs-Fähigkeiten generieren diese nicht nur eine mögliche Bauteilsequenz, sondern auch eine Aufschlüsselung der notwendigen Manipu-lations- und Fügeoperationen pro Bearbeitungsschritt. Um Halluzinationen des Netzwerkes zu reduzieren, werden zusätzlich klassische Validierungsregeln sowie VLMs zum Grounding eingesetzt.

- **Sequenzoptimierung auf Basis von Pareto:** Eine allgemeingültige, optimale Montage- oder Demontagesequenz für eine Baugruppe mittels KI zu generieren, ist aufgrund der Vielzahl an zu beachtenden Faktoren oftmals nicht möglich. Je nach eingesetzter KI und den zugrundeliegenden Trainingsdaten variieren die Schwerpunkte. So kann es beispielsweise sein, dass sich das eine Netzwerk stärker auf eine möglichst optimale Stabilität der Baugruppe fokussiert, während ein weiteres einen stärkeren Fokus auf die Lokalität von Montageschritten legt. Vor diesem Hintergrund wurde untersucht, ob sich die empfohlenen Sequenzschritte der verschiedenen Netzwerke kombinieren lassen, um sowohl eine höhere Güte der generierten Sequenz zu erreichen als auch die Zuverlässigkeit der kombinierten Sequenz im Vergleich zu den Sequenzen der einzelnen Netze zu erhöhen. Hierzu wird ein Pareto-Optimierer angewandt.

Große Aufmerksamkeit wurde bei den betrachteten Ansätzen besonders auf den Sequenz-zu-Sequenz-Ansatz und den LLM-VLM-Ansatz gelegt. Dementsprechend werden diese beiden Techniken detailliert im folgenden Kapitel vorgestellt und diskutiert. Für eine genauere Erläuterung des Graph-Transformer-Ansatzes wird auf [Timmermann et al., 2024a] verwiesen. Für eine genauere Vorstellung der Sequenzoptimierung sind Details in [Timmermann et al., 2024b] veröffentlicht.

4.4. Handlungsplanung mit Affordanzen

Zur Überführung in einen symbolischen Plan ist neben der reinen Sequenzgenerierung auch die Zuweisung von notwendigen Fügeoperationen für jeden Sequenzschritt erforderlich, wie sie beispielsweise nach der DIN 8593 – „Fertigungsverfahren Fügen“ [163] definiert sind. Da abhängig davon, welcher Teil des zu verwendenden Bauteils gefügt wird, eine andere Fügeoperation genutzt werden muss, ist es notwendig, in jedem Arbeitsschritt konkret die zu fügenden Bauteile und ihre Kontaktstellen zu prüfen.

Zur Definition der Beziehung zwischen Bauteilbereich/Teilgeometrie und entsprechender Fügeoperation wird das Konzept der Affordanzen verwendet und in Form von Montageaffordanzen auf den konkreten Anwendungsfall Montage übertragen. Dadurch ist es auch möglich, zulässige Affordanzpaare zu definieren, welche die theoretisch möglichen Fügeoperationen für zwei sich berührende Bauteile darstellen. Die möglichen Affordanzen und die gültigen Affordanzpaare sind direkt aus der DIN 8593 abgeleitet.

Da ein händisches Markieren der gültigen Affordanzen auf jedem Bauteil sehr zeitintensiv ist, wird das System durch eine KI-basierte Affordanzdetektion ergänzt. Diese verwendet ein CNN-basiertes, gelerntes tiefes neuronales Netzwerk, um auf den Kameradaten der Bauteile die gültigen Affordanzen zu segmentieren. Dazu wird sowohl ein Single-Stream-Ansatz als auch ein Multi-Stream-Ansatz

untersucht. Der Multi-Stream-Ansatz berücksichtigt insbesondere Kanteninformationen stärker, sodass feinere Bauteilstrukturen, wie etwa Gewinde, besser erkannt und ausgewertet werden können.

Zur Auswertung der erkannten Montageaffordanzen wird anschließend ein hierarchisch gegliederter Planer verwendet. Dieser greift neben den erkannten Montageaffordanzen sowohl auf die zuvor generierte Montagesequenz als auch auf Geometrieinformationen aus den Bauteilmodellen zu. Nach einer Reduktion der zu betrachtenden Affordanzpaare – durch die Untersuchung auf gültige Kontaktstellen – leitet der Planer die potenziellen Fügeoperationen ab und priorisiert diese. Die am höchsten priorisierte Fügeoperation stellt die korrekte Fügeoperation dar. Durch eine nachgelagerte Analyse aller ausgewählten Fügeoperationen kann das System zusätzlich bestimmen, ob in bestimmten Prozessschritten ein stoffschlüssiger Fügeprozess notwendig ist, um Bauteile final in Position zu bringen. Abschließend wird der Montageplan sowohl in graphischer als auch in serialisierter Form für die Weiterverarbeitung durch eine intelligente Roboterzelle ausgegeben.

4.5. Intelligente Roboterzelle

Eine intelligente Roboterzelle ermöglicht die Automatisierung komplexer Montageprozesse, die über die Fähigkeiten klassischer Fertigungszellen hinausgehen. Durch den Einsatz von hochauflösender und schneller Sensorik, wie beispielsweise Kameras und taktilen Sensoren, kann ein Roboter eine Montageszene wahrnehmen und autonom auf neue oder sich ändernde Produkte dynamisch reagieren. Ihre zentralen Fähigkeiten sind diejenigen zur Kognition und Adaptivität. Sie werden durch KI-basierte Bildverarbeitungssysteme, dynamische kollisionsfreie Pfadplaner und integrierte Robotercodegeneratoren realisiert. Dadurch wird der Roboterzelle ermöglicht, selbstständig Entscheidungen zu treffen, einen neuen Montageplan umzusetzen und Prozessparameter in Echtzeit anzupassen. Auf diese Weise kann eine durchgängig hohe Qualität gewährleistet werden. Eine weitere essenzielle Eigenschaft ist die Fähigkeit zur Mensch-Roboter-Kollaboration (MRK). Sie ermöglicht eine enge und sichere Interaktion mit menschlichen Fachkräften in geteilten Arbeitsbereichen, um die jeweiligen Stärken optimal zu kombinieren, etwa bei der Handhabung komplexer Bauteile.

In dieser Untersuchung sind intelligente Roboterzellen kein Hauptfokus, da sie besonders die Weiterverarbeitung und Ausführung des High-Level-Plans betrachten. Im Rahmen der durchgeführten Forschung wurden hierzu primär zwei Fragestellungen untersucht.

Obwohl moderne KI-basierte Bildverarbeitungssysteme in den vergangenen Jahren qualitativ hochwertigere Ergebnisse erzielen konnten, sind sie bei der exakten 3D-Bauteillokalisation noch nicht vergleichbar mit klassischen Ansätzen der

Bildverarbeitung, welche Techniken wie Modellabgleiche verwenden. Eine große Herausforderung bei diesen Ansätzen ist jedoch deren lange Ausführungszeit, die dann auftritt, wenn der Modellabgleich auf einen großen Arbeitsraum angewandt werden muss. Um diese Problematik zu adressieren, wurde ein hybrider Vision-Ansatz konzipiert und evaluiert. Er besteht aus zwei getrennten Komponenten: einem Mask-RCNN-basierten Objektklassifikator und Lokalisator, welcher für die im Arbeitsraum klassifizierten Objekte eine entsprechende Region-of-Interest definiert, und einer klassischen 3D-Modell-Matching-Komponente für die exakte Bauteillokalisation. Zur Steigerung der Performance wird das Matching ausschließlich in der definierten Region-of-Interest durchgeführt. Zusätzlich kann durch die zuvor erfolgte Klassifizierung das zu matchende Modell bereits eingegrenzt werden. So steigern sich zusätzlich die Robustheit und Ausführungsgeschwindigkeit, da ausschließlich dieses Modell gematcht werden muss. Für eine genauere Vorstellung des hybriden Ansatzes wird auf [Timmermann et al., 2021a] verwiesen.

Besonders MRK-Anwendungen profitieren von einer schnellen und präzisen Lokalisation, erlaubt diese doch das dynamische Hinzuziehen eines Roboters zu einem Montagevorgang. Eine Herausforderung stellt allerdings die Übermittlung des aktuellen Zustands der Montage dar. Wird der Montageprozess nicht dauerhaft überwacht, ist es dem System nur eingeschränkt möglich, aus dem Kamerabild den aktuellen Zustand der Montage abzuleiten, da beispielsweise Bauteile im Inneren einer Baugruppe für das Kamerasystem möglicherweise verdeckt sind. Aus der Betrachtung der nicht verbauten Bauteile kann dann eine Lösung abgeleitet werden. Hierbei untersucht das System, welche Schritte im Montageplan bereits durchgeführt sein könnten, und führt bei Mehrdeutigkeiten, welche durch gleiche Bauteile auftreten können, eine logische Analyse durch. Details zu diesem Ansatz sind in [Timmermann et al., 2020] dargestellt.

4.6. Demontage

Die in den folgenden Kapiteln detailliert vorgestellten Ansätze beziehen sich auf verschiedene Montageszenarien. Die angewandten Techniken für die Sequenzgenerierung sowie für die Generierung des symbolischen Plans lassen sich auch auf den Demontagekontext anwenden. Ein Montagevorgang kann ebenso als inverser Demontagevorgang gesehen werden, vorausgesetzt, es sind keine stoffschlüssigen Verbindungen vorhanden, sodass keine dauerhaften Verbindungen betrachtet werden müssen.

4.7. Zusammenfassung und Fazit: Ansatz für einen generativen Montageplanungsprozess

Das Kapitel präsentiert das Gesamtkonzept des Montageplanungsprozesses. Das System transformiert Produktmodelle automatisiert in ausführbare symbolische Pläne für Roboterzellen. Um dem Mangel an Trainingsdaten zu begegnen, wird zunächst ein Modul vorgestellt, das mittels eines genetischen Algorithmus synthetische, physikalisch valide Montagesequenzen aus CAD-Daten generiert.

Kern des Frameworks ist die Sequenzbestimmung, für die verschiedene KI-Ansätze evaluiert werden, darunter Sequenz-zu-Sequenz-Modelle auf Voxelbasis sowie die Kombination von Large-Language-Modellen mit Wissensgraphen. Zur Überführung der reinen Sequenz in einen Handlungsplan führt das Kapitel das Konzept der Montageaffordanzen ein. Hierbei detektiert ein KI-gestütztes Visionsystem funktionale Geometrieeigenschaften, woraufhin ein semantischer Planer die korrekten Fügeoperationen gemäß DIN 8593 ableitet. Abschließend wird die Integration in eine intelligente Roboterzelle skizziert, die mittels hybrider Bildverarbeitung die Pläne adaptiv ausführt. Das Konzept ist zudem unter bestimmten Voraussetzungen auf Demontageprozesse übertragbar.

5. Sequenzplanung mit Hilfe von generativer KI

Dieses Kapitel befasst sich mit der Generierung von physikalisch umsetzbaren Montagesequenzen durch KI-gestützte Verfahren. Zunächst wird der verwendete Ansatz vorgestellt, mit dem ein Trainingsdatensatz für die KI-Ansätze erzeugt wird. Bei diesem Ansatz handelt es sich um eine Adaption eines genetischen Algorithmus. Anschließend wird ein Verfahren präsentiert, welches mithilfe eines Sequenz-zu-Sequenz-Encoder-Decoder-Ansatzes 3D-Repräsentationen von Baugruppen lernen und deren Montagereihenfolge auf ähnliche Baugruppen übertragen kann. Abschließend wird ein Framework-Konzept vorgestellt, das Wissensgraphen und Foundation Models kombiniert, um detaillierte Baugruppeninformationen mit einem Large-Language-Model auszuwerten und daraus eine Montagesequenz abzuleiten.

5.1. Trainingsdatengenerierung

Bei überwachten und unüberwachten Lernansätzen ist ein großer Datensatz mit qualitativ hochwertigen Daten unverzichtbar. Daraus können solche Verfahren Prinzipien und Gesetzmäßigkeiten lernen und diese anschließend auf neue bzw. ähnliche Problemstellungen übertragen. Da im Rahmen dieser Arbeit unterschiedliche KI-Verfahren betrachtet werden, ist ein entsprechender Datensatz notwendig. Im Montagekontext existieren jedoch keine frei verfügbaren, aufbereiteten Datensätze, die eine ausreichende Menge an Daten enthalten. Daher wird zunächst ein Datensatz erzeugt und ein entsprechendes Werkzeug entwickelt.

Grundlage des Erzeugungswerkzeug ist ein genetischer Algorithmus (GA), wie in Kapitel 2.2.1 beschrieben. Dieser generiert durch Crossover- und Mutations-Opera-toren mögliche Montagesequenzen und bewertet sie auf Basis der Kriterien: Kollisionsfreiheit, Lokalität und Stabilität. Im Vergleich zu einem klassischen genetischen Algorithmus ist das Ziel dieses Werkzeugs nicht nur, eine möglichst optimale Lösung zu generieren, sondern eine größere Menge an validen Lösungen bereitzustellen. Dies ist hilfreich, um einem lernenden Ansatz aufzuzeigen, dass verschiedene Lösungen zu einem korrekten Montageergebnis führen können.

Für die Generierung von Trainingsdaten durch das vorgeschlagene Werkzeug muss die Baugruppe einige Randbedingungen erfüllen:

- Die Baugruppe muss als STEP-Datei vorliegen.
- In der STEP-Datei ist die Baugruppe bereits korrekt orientiert (Schwerkraft in $-Z$-Richtung bzw. entgegen der Aufbau-Richtung).
- Bauteile müssen auf einer linearen Trajektorie an ihre Zielposition bewegt werden können.
- Die Fügerichtung der Bauteile muss parallel zu einer der drei Hauptachsen liegen.
- Die Baugruppe enthält keine stoffschlüssigen Fügeoperationen, sodass sie durch die inverse Montagesequenz zerlegt werden kann.

5.1.1. Notwendige Vorverarbeitung von CAD-Daten

Damit die CAD-Daten der Baugruppe von einem Planer verwendet werden können, müssen sie geeignet vorverarbeitet werden.

In einem ersten Schritt wird durch das System geprüft, ob alle in der STEP-Datei definierten Bauteile eine eindeutige ID besitzen. Sollte dies nicht der Fall sein, werden die IDs der Bauteile adaptiert. Dieser Schritt ist essenziell, da ohne eindeutige IDs eine spätere Zuordnung der Bauteile zu den einzelnen Montageschritten nicht mehr möglich ist.

Anschließend werden alle Bauteile darauf geprüft, ob sich ihre Geometrie mit der eines anderen Bauteils überschneidet. Dies kann u. a. vorkommen, wenn ein Bauteil mittels Einpressen gefügt wird. Das Übermaß ist in diesen Fällen zwar korrekt, verhindert jedoch die Prüfung von Kriterien wie beispielsweise der Kollisionsfreiheit. Da der Datensatz ausschließlich dazu dient, mögliche Montagesequenzen zu erlernen, nicht aber die dafür notwendigen Fügeoperationen, werden die Bauteile mit Übermaß bereinigt, indem die Geometrie des kleineren Bauteils entsprechend verkleinert wird.

Nach der Bereinigung findet eine Extraktion der Liaisondaten statt. Diese beinhalten für jedes Bauteilpaar folgende Informationen: eine Liste von Berührungspunkten zwischen den Bauteilen und die relative Transformation zwischen ihnen. Im Fall von sich berührenden Bauteilflächen, wodurch theoretisch unendlich viele Berührungspunkte existieren, wird die Berührungsfläche durch einen Punkt approximiert, der im geometrischen Zentrum der Fläche liegt. Sollte die Liste keinen Eintrag beinhalten, so berühren sich die Bauteile nicht.

5.1.2. Initialisierung von Populationen

Für die Initialisierung einer Population wird ein Ansatz gewählt, der mindestens ein Individuum mit einer korrekten Montagesequenz erzeugt und die restlichen Individuen der Ausgangspopulation mit zufällig generierten Sequenzen besetzt.

Dieses Vorgehen gewährleistet, dass der Algorithmus bereits in der ersten Population zielgerichtet Lösungen weiterentwickelt. Im Vergleich zur Initialisierung aller Individuen mit validen Montagesequenzen steigt zwar das Risiko, dass in der ersten Population nur sehr wenige korrekte Sequenzen enthalten sind; demgegenüber ist die Generierung vieler korrekter Montagesequenzen jedoch sehr zeitaufwendig, und viele dieser Individuen würden nicht effektiv zur Folgepopulation beitragen.

Um das Individuum mit der korrekten Montagesequenz zu erzeugen, wird ein Dekonstruktionsansatz verwendet. Unter der Annahme, dass die Demontage in inverser Reihenfolge zur Montage erfolgen kann, wird nach einer möglichen Zerlegungssequenz gesucht. Zunächst wird ein Startbauteil bestimmt. Dazu wird eine gewichtete Zufallsauswahl durchgeführt. Die Anzahl der Liaisons stellt in diesem Fall den Gewichtungsfaktor dar. Bauteile mit weniger Liaisons, also Bauteile, die weniger Verbindungen zu anderen Bauteilen haben, werden hierbei höher gewichtet und somit mit höherer Wahrscheinlichkeit ausgewählt. Anschließend versucht das System, das Bauteil kollisionsfrei aus der Baugruppe zu entfernen. Gelingt dies, gilt der Schritt als physikalisch machbar und wird als korrekt angenommen. Dieser Prozess wird dann für die verbleibenden Bauteile in der Baugruppe wiederholt, bis eine vollständige Lösung gefunden wurde. Durch dieses Vorgehen kann innerhalb der Laufzeit von $O(n!)$ eine Lösung gefunden werden.

5.1.3. Fitnessberechnung

Nach der Generierung einer Population erfolgt stets die Fitnessberechnung. Dazu werden drei Bewertungsfunktionen miteinander kombiniert, sodass sich die Fitness jedes Individuums G wie folgt berechnet:

$$f(G) = c_{kollision} \cdot f_{kollision}(G) + c_{stabil} \cdot f_{stabil}(G) + c_{lokal} \cdot f_{lokal}(G) \tag{5.1}$$

Hierbei sind $f_{kollision}$, f_{stabil} und f_{lokal} die einzelnen Bewertungsfunktionen, die mit ihren jeweiligen Konstanten $c_{kollision}$, c_{stabil} und c_{lokal} gewichtet aufsummiert werden und die Gesamtfitnessfunktion bilden.

Kollisionsberechnung
Für die Berechnung des Fitnesswertes bezüglich der Kollisionsfreiheit muss geprüft werden, ob ein Bauteil während des Fügevorgangs mit einem anderen Bauteil kollidiert. Zur schnelleren Analyse wird die kontinuierliche Fügetrajektorie diskretisiert, wobei die Größe der Diskretisierungsschritte frei wählbar ist. Konkret wird die Fitness über folgende Gleichung berechnet:

$$f_{kollision}(G) = \begin{cases} 1 & \text{wenn } \forall g \in G : \exists P_{g,O(g,G)} \\ \frac{|V|}{c \cdot |G|} & \text{sonst, wobei } V = \{g \in G \mid v(g,G) = 1\} \end{cases} \tag{5.2}$$

Hierbei gilt:

$$v(g, G) = \begin{cases} 1 & \text{wenn } \exists P_{g,O(g,G)} \\ 0 & \text{sonst} \end{cases} \tag{5.3}$$

Der Wert der Fitnessfunktion nimmt somit 1 an, wenn über alle Montageschritte hinweg keine Kollisionen auftreten. Andernfalls entspricht der Wert der Fitnessfunktion dem Anteil der aufeinanderfolgenden gültigen Montageschritte an der Gesamtzahl der Montageschritte, dividiert durch einen Faktor c.

Stabilitätsberechnung
Die Berechnung der Stabilität der Baugruppe erfolgt über die Betrachtung der Kippstabilität. Dies stellt eine vereinfachte Betrachtung dar, ist jedoch ein ausreichendes Vorgehen, da keine Unterscheidung zwischen verschiedenen Fügeoperationen getroffen wird. Eine weitere Vereinfachung ist die Annahme einer identischen Dichte für alle Bauteile. Pro Schritt der Montagesequenz prüft der Algorithmus, ob sich der verändernde Schwerpunkt durch das Fügen des Bauteils g auf die Stabilität der Baugruppe auswirkt. Folgende Schritte finden statt:

1. Berechnung des Gesamtschwerpunktes nach dem Fügen des neuen Bauteils.

2. Berechnung der Aufstandspunkte der bis dahin gefügten Baugruppe.

3. Ableitung des konvexen Aufstandspolygons (Konvexe Hülle) aus den Aufstandspunkten.

4. Projektion des Schwerpunktes in die Ebene des Aufstandspolygons.

Liegt der projizierte Schwerpunkt innerhalb des Aufstandspolygons, so gilt der aktuelle Montagezustand als stabil. Wie in Gleichung 5.4 und 5.5 beschrieben, ergibt sich die Stabilität wie folgt:

$$s(g, G) = \begin{cases} 1 & \text{wenn } x_s(g) \text{ über } x_s(O(g, G)) \text{ liegt} \\ 0 & \text{sonst} \end{cases} \tag{5.4}$$

$$f_{stabil}(G) = \begin{cases} 1 & \text{wenn } \forall g \in G : s(g, G) = 1 \\ \frac{1}{|\overline{S}|} & \text{sonst, wobei } \overline{S} = \{g \in G \mid s(g, G) = 0\} \end{cases} \tag{5.5}$$

Lokalitätsberechnung
Zur Reduktion von Trajektorienlängen wird zusätzlich ein Lokalitätskriterium als weiterer Fitnesswert genutzt. Dieser ermöglicht es, Montagesequenzen zu bevorzugen, in denen Bauteile mit geringem räumlichem Abstand in aufeinanderfolgenden Schritten gefügt werden. Für die Berechnung wird dementsprechend der Abstand des aktuellen Bauteils zu dem zuvor gefügten Bauteil betrachtet. Die Abstände werden summiert und anhand der Anzahl der Bauteile normiert.

Hieraus ergibt sich folgende Gleichung:

$$f_{lokal}(G) = 1/\sum_{g \in G} (x_s(g) - x_s(O(g, G))) \tag{5.6}$$

5.1.4. Generierung einer neuen Population

Zur Generierung einer neuen Population ist es zunächst notwendig, Individuen als potenzielle Elternkandidaten auszuwählen. Die ausgewählten Kandidaten werden anschließend durch die Anwendung des Mutations- und Crossover-Operators kombiniert. Auf diese Weise lassen sich neue Individuen erzeugen. Um für die Anwendung im Montagekontext geeignet zu sein, müssen beide Operatoren adaptiert werden.

Elternauswahl
Im klassischen GA-Konzept ist vorgesehen, dass Individuen mit einer hohen Fitness sich über mehrere Generationen behaupten können und die Nachkommen in neuen Generationen ausschließlich auf diesen Vorläufern basieren. Dies hat zur Folge, dass Individuen unter einem gewissen Fitnessschwellenwert für die Generierung der Nachfolgegeneration nicht weiter betrachtet werden. Als Konsequenz daraus ergibt sich eine schnelle Verbesserung der Fitness, allerdings auch eine höhere Anfälligkeit, in lokalen Maxima zu verharren.

Um diese Thematik entsprechend zu adressieren, wird der Fitnessschwellenwert nicht als ausschließliches Kriterium für die Elternauswahl genutzt. Ergänzend kommt eine probabilistische Methode zum Einsatz, bei der jedes Individuum eine Wahrscheinlichkeit proportional zu seiner Fitness besitzt. Mithilfe der daraus entstehenden Wahrscheinlichkeitsverteilung werden die Eltern zufällig gewählt und eine nachfolgende Generation erzeugt.

Mutations-Operator
Der klassische Mutationsoperator tauscht ein einzelnes Gen gegen ein beliebiges anderes Gen aus. Im Montagekontext würde dies zu fehlerhaften Montagesequenzen führen, da dasselbe Bauteil mehrfach vorkommen könnte. Dementsprechend muss der Mutations-Operator angepasst werden. Dies geschieht durch die

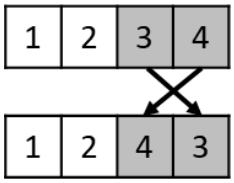


Abbildung 5.1.: Die Swap-Mutation, bei der keine trivialen Gendefekte mehr auftreten können.

Adaption hin zu einer *Swap-Mutation*. Im Vergleich zum klassischen Ansatz werden zwei Bauteile innerhalb der Montagesequenz getauscht, sodass weiterhin sichergestellt ist, dass jedes Bauteil einzigartig bleibt. Ein Beispiel dazu ist in Abbildung 5.1 dargestellt.

Crossover-Operator
In seiner klassischen Form ist auch der Crossover-Operator nur bedingt geeignet, da er vermehrt ungültige Montagesequenzen erzeugt. Aufgrund dieses Umstands ist für den Montagekontext der *Two-Point-Crossover* dem *Single-Point- Crossover* vorzuziehen, da er besser mit festen Reihenfolge-Bedingungen umgehen kann. Gleichwohl erzeugt auch der Two-Point-Crossover aufgrund seiner Funktionsweise möglicherweise Montagesequenzen, welche mehrere gleiche Bauteile enthalten. Entsprechend wird folgende Anpassung durchgeführt:

1. **Generierung:** Der normale Two-Point-Crossover wird ausgeführt.
2. **Bereinigung:** Das Kind-Genom wird auf Duplikate geprüft. Dabei hat der neu eingefügte Abschnitt Vorrang; Duplikate werden aus dem restlichen Genom entfernt.
3. **Reparatur (Auffüllen):** Fehlende Bauteile werden wieder eingefügt.
    - Position: Vor oder nach dem ausgetauschten Abschnitt.
    - Reihenfolge: Die fehlenden Teile werden in der Reihenfolge eingefügt, in der sie im Spender-Elternteil vorkamen. Dies erhöht die Wahrscheinlichkeit, dass die logischen Abhängigkeiten der Montage eingehalten werden.

Abbildung 5.2 veranschaulicht dieses Prinzip anhand eines einfachen Beispiels.

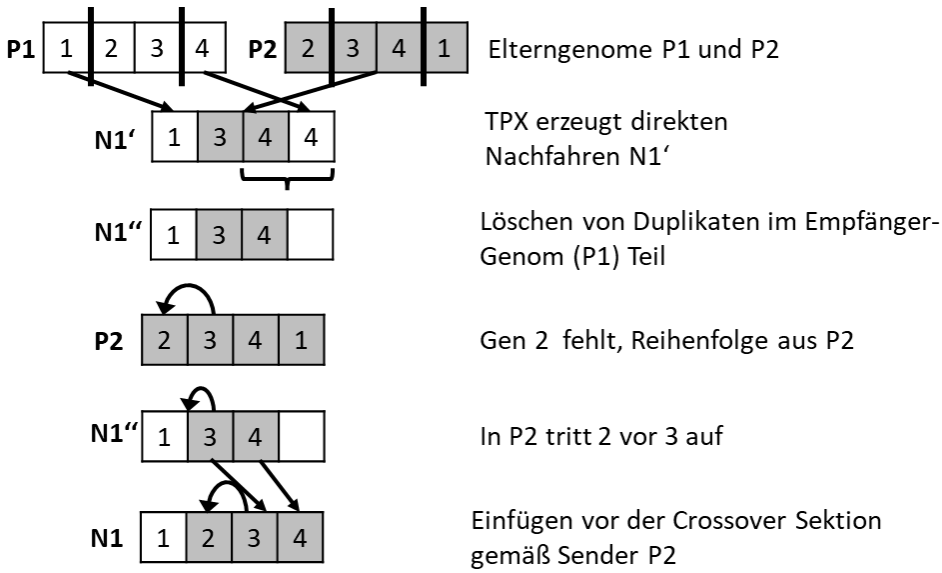


Abbildung 5.2.: Der selbstheilende TPX-Operator.

5.1.5. Abbruchkriterium

Da ein genetischer Algorithmus zyklisch durchgeführt wird, ist ein Abbruchkriterium notwendig, das die Ausführungszyklen beschränkt. Im Rahmen dieses Konzepts existieren zwei Abbruchkriterien: die maximale Laufzeit und die Konvergenz der Individuen.

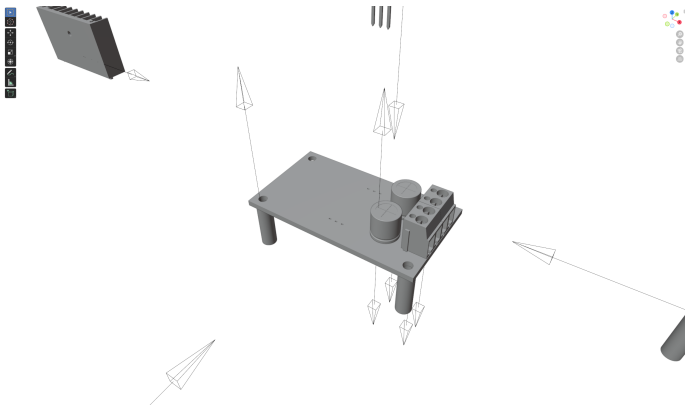
Die Begrenzung der maximalen Laufzeit sorgt vor allem für die Verhinderung von Endlosschleifen in den Fällen, in denen der Algorithmus keine initiale Lösung finden kann. Dieses Kriterium dient daher primär der Absicherung.

Das Konvergenzkriterium dient dazu festzustellen, wenn bei den Individuen über mehrere Generationen hinweg keine Verbesserung stattfindet. Hierbei gilt, dass Konvergenz erreicht ist, wenn sich das Individuum mit der höchsten Fitness über x Generationen nicht mehr ändert. Wird die Generierung durch das Konvergenzkriterium terminiert, wird anschließend die Menge der n Individuen mit dem höchsten Fitnesswert ausgegeben.

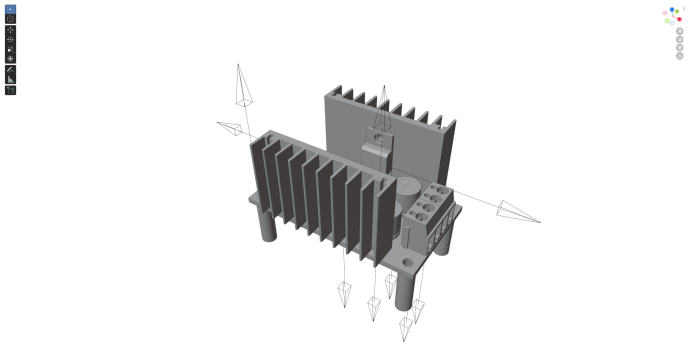
5.1.6. Optimierung des Ansatzes

Die initiale Umsetzung des Konzepts erfolgte mittels FreeCAD [164], da dieses über eine gut nutzbare Python-API [165] verfügt und direkt auf den STEP-Dateien arbeiten kann. Ein großer Nachteil des Systems ist jedoch die geringe Rechenperformance, insbesondere bei Kollisionsberechnungen und den dafür verwendeten Booleschen Operatoren. Dies hatte zur Folge, dass die Berechnung der Montagesequenzen für eine Baugruppe mittlerer Komplexität (20–50 Bauteile) mehrere Stunden dauerte, bis eine Menge von ca. 300 validen Montagesequenzen generiert war. Eine Ursachenanalyse zeigte, dass die hohe Modellauflösung des CAD-Formats einen erheblichen Rechenaufwand verursacht, gleichzeitig aber eine Genauigkeit bietet, die in wenigen Fällen benötigt wird. Zur Reduktion der Laufzeit wurde daher die Portierung des Ansatzes in ein Werkzeug durchgeführt, das Blender verwendet.

Blender verarbeitet im Vergleich zu FreeCAD keine direkten CAD-Modelle, sondern Polygonmodelle. Diese bestehen aus Vertices (Eckpunkten), welche die Geometrie definieren. Durch das Zusammenfassen benachbarter Vertices lassen sich die Modellgeometrien deutlich vereinfachen und so insbesondere Kollisionsberechnungen beschleunigen. Des Weiteren wurden Verbesserungen der Performance erreicht, indem in einem ersten Schritt automatisch die spätere Fügerichtung eines Bauteils bestimmt wird. Auf Basis dieser Richtung findet dann eine Projektion der relevanten Bauteile in eine 2D-Ebene statt, was eine schnelle Vorabentscheidung ermöglicht, ob es zwischen zwei Bauteilen überhaupt zu einer Kollision kommen kann. Das Ergebnis dieser Analyse wird in einer Liste gespeichert, auf die in jedem Iterationsschritt des genetischen Algorithmus zurückgegriffen wird.



(a) Ausgangsposition aller Bauteile, bevor sie über die finale Fügetrajektorie gefügt werden.



(b) Vollständig montierte Leiterplatine.

Abbildung 5.3.: Grafische Visualisierung einer möglichen Montagesequenz für eine Leiterplatine.

Da durch die Vereinfachung der Modellgeometrie Details verloren gehen können, sind insbesondere Bauteile mit feiner Geometrie, z. B. eine Schraube, problematisch, da je nach gewählter Vereinfachung Teile der Geometrie ggf. komplett entfernt werden. Zur Umgehung dieses Problems wird ein Skalierungsschritt eingefügt. Dieser prüft zu Beginn des Prozesses, ob alle Bauteile eine Mindestgröße aufweisen. Ist dies nicht der Fall, wird das kleinste Bauteil ausgewählt und auf die entsprechende Mindestgröße hochskaliert. Der ermittelte Skalierungsfaktor wird dann auf alle weiteren Bauteile angewandt. Dieses Verfahren funktioniert für eine Vielzahl der Baugruppen zufriedenstellend. Problematisch ist es nur bei Baugruppen, die eine große Spreizung der Bauteilgrößen aufweisen, z. B. M1,5- und M50-Schrauben in derselben Baugruppe. Das so angepasste Tool erlaubt eine deutlich schnellere Generierung von Trainingsdaten sowie die Abdeckung einer großen Anzahl von Modellvarianten. Um die Pläne auch für einen menschlichen Nutzer schnell erkennbar zu machen, wurde zusätzlich eine grafische Visualisierung umgesetzt; diese ist ebenfalls in Abbildung 5.3 dargestellt.

5.2. Montagesequenzgenerierung mittels eines Sequenz-zu-Sequenz-Ansatzes

Ausgehend von dem zuvor erstellten Datensatz kann nun ein Ansatz realisiert werden, der Montagesequenzen für ähnliche Baugruppen generieren kann. „Ähnlich“ ist in diesem Kontext definiert als verschiedene Varianten einer grundlegenden Baugruppe. Der folgende Ansatz soll es ermöglichen, aus vorangegangenen Montagen die wichtigsten Kriterien zu erlernen und diese anschließend auf eine neue Variante des gleichen Typs anzuwenden. Umgesetzt wird dieses Konzept mithilfe eines Sequenz-zu-Sequenz-Encoder-Decoder-Ansatzes (Seq2Seq). Bei diesem handelt es sich um eine Weiterentwicklung des *PQ-Net* von Wu et al. [166].

Basierend auf einem 3D-Modell der Baugruppe, in dem die Bauteile einzeln modelliert sind, erfolgt zunächst eine Voxelierung des Modells und der einzelnen Komponenten. Anschließend verarbeitet ein Autoencoder die voxelierten Komponenten und transformiert deren Geometrie in einen niedrigdimensionalen Merkmalsvektor. Eine korrekte Montagesequenz der Bauteile wird dann durch den Sequenz-zu-Sequenz-Encoder-Decoder generiert, indem die Komponenten zunächst in einen latenten Raum transformiert werden, bevor die Sequenz auf Basis der Bauteilformen, ihrer räumlichen Position und Größe sowie ihrer eindeutigen Bauteil-ID abgeleitet wird. Im Vergleich zum *PQ-Net* kann dieser Ansatz auch Montagesequenzen für Baugruppen mit gleichartigen Bauteilen (beispielsweise mehrere Schrauben des gleichen Typs) verarbeiten, was für eine Verwendung mit realen Baugruppen zwingend erforderlich ist.

5.2.1. Anforderungen und Vorverarbeitung

Damit Baugruppen auf der Basis dieses Ansatzes verarbeitet werden können, müssen sie spezifische Anforderungen erfüllen:

- Es werden ausschließlich physikalisch mögliche Baugruppen betrachtet, die kollisionsfrei montiert werden können.
- Alle Bauteile einer Baugruppe müssen vollständig modelliert sein.
- Eine Baugruppe darf keine sich überschneidenden Bauteile enthalten. Sollte es zwischen den Bauteilen Überschneidungen geben, etwa durch ein Übermaß bei einem Passstift, werden diese mit dem gleichen Ansatz wie beim genetischen Algorithmus aufgelöst.
- Die Bauteile der Baugruppe benötigen ein ähnliches Größenverhältnis; andernfalls könnten Bauteile bei der Vorverarbeitung fälschlicherweise entfernt werden.
- Bauteile müssen eine eindeutige Bauteil-ID besitzen.

Baugruppen, welche diese Anforderungen erfüllen, werden in einem ersten Schritt vorverarbeitet. Hierzu werden die Bauteile sowie die gesamte Baugruppe inklusive aller Objekte auf ein $64 \times 64 \times 64$ Raster voxeliert. Dies vereinfacht die Geometrie der Modelle und Bauteile stark, mit der Konsequenz, dass feine Details, etwa Gewindegänge, verloren gehen. Sind erhebliche Größenunterschiede vorhanden, z. B. eine kleine M1-Schraube an einem Fahrzeug, würden die kleinen Bauteile entfernt werden, da sie nicht von Artefakten unterschieden werden können, welche durch die Voxelierung entstehen. Daraus ergibt sich die Anforderung der ähnlichen Größenverhältnisse.

5.2.2. Autoencoder

Der Autoencoder ist die erste Komponente des verwendeten Netzwerks. Er wird initial auf die Geometrie aller später vorkommenden Bauteile trainiert und transformiert die geometrischen Repräsentationen in eine Darstellung im latenten Vektorraum, die anschließend vom Netzwerk weiterverwendet werden kann. Die Architektur des Autoencoders ist vergleichbar mit jener des *PQ-Net*. Sie verwendet eine Kombination aus einer 3D-Faltungsschicht (engl. *3D Convolutional Layer*), einer 3D-Batch-Normalisierungsschicht (engl. *3D Batch Normalization Layer*) und abschließend einer Leaky-ReLU-Schicht als Aktivierungsfunktion. Für das Encoding werden fünf dieser Schichtkombinationen sequenziell hintereinander geschaltet, bevor die Transformation in den latenten Vektorraum durch eine finale 3D-Faltungsschicht und eine sich anschließende Sigmoid-Aktivierungs-funktion abgeschlossen wird.

Für das Training des Encoders wird zusätzlich ein entsprechender Decoder benötigt. Dieser verfügt über eine simplere Architektur: Er besteht aus sechs vollständig verbundenen Schichten, wobei jede Schicht eine eigene Leaky-ReLU-Aktivierungsfunktion besitzt. Durch den Decoder kann anschließend die latente Darstellung der Bauteile in die geometrische Darstellung zurücküberführt werden. Die Architektur des Autoencoders ist in Abbildung 5.4 dargestellt.

Um sicherzustellen, dass der Autoencoder alle notwendigen Merkmale bei der Überführung von der geometrischen in die latente Darstellung berücksichtigt, wird im Training die Geometrie der Bauteile durch den Encoder in den latenten Vektorraum transformiert und diese Darstellung anschließend durch den Decoder rekonstruiert. Jegliche Abweichungen werden durch das gesamte Netzwerk (Decoder und Encoder) zurückpropagiert und die Netzwerkgewichte angepasst, bis zum Ende des Trainingsprozesses eine vollständige Rekonstruktion jedes Bauteils möglich ist.

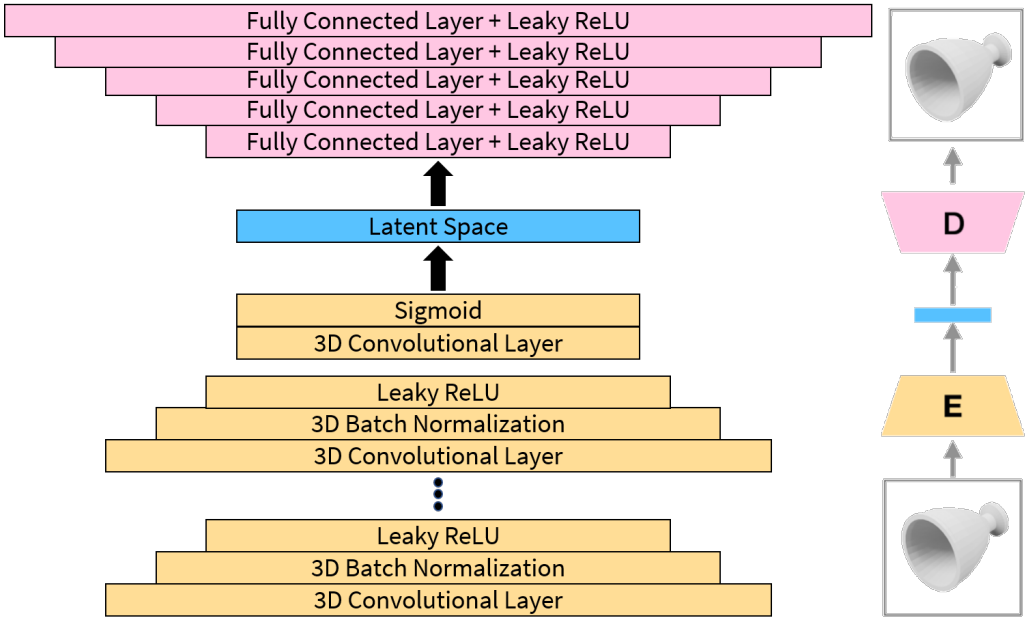


Abbildung 5.4.: Die Struktur des Autoencoders [Timmermann et al., 2024d].

5.2.3. Sequenz-zu-Sequenz-Encoder-Decoder

Die komprimierte Darstellung der Bauteile und der Baugruppe wird nun für die Generierung der Montagesequenz durch das Seq2Seq-Encoder-Decoder-Netzwerk verwendet. Hierzu nutzt das Netzwerk *Gated Recurrent Units* (GRU) [167].

GRUs sind darauf ausgelegt, mit sequenziellen Daten zu arbeiten, ohne dass Probleme mit verschwindenden Gradienten auftreten. Die beiden Hauptbestandteile einer GRU sind der verborgene Zustand (engl. *Hidden State*) und die Tore (engl. *Gates*). Der verborgene Zustand stellt eine Art Gedächtnisvektor dar. Er speichert Informationen aus vorangegangenen Zeitschritten und macht sie für den aktuellen Zeitschritt verfügbar. Dies ermöglicht dem Netzwerk, Eingaben nicht isoliert, sondern im Kontext der Historie zu interpretieren. Die Tore steuern, welche Informationen beibehalten und welche verworfen werden. Hierzu erhält jede GRU-Zelle zwei Eingaben: die aktuelle Eingabe x_t und den verborgenen Zustand des vorherigen Schrittes h_{t-1}. Die Tore werden unterschieden in Reset-Tor (engl. *Reset Gate*) und Update-Tor (engl. *Update Gate*). Während das Reset-Tor entscheidet, wie viel vom vergangenen Wissen ignoriert werden soll, steuert das Update-Tor, wie stark der neue Zustand vom alten Zustand abhängt. Die konkrete Funktionsweise einer GRU ist wie folgt:

1. **Filterung:** Durch das Reset-Tor wird bestimmt, welche Informationen aus dem verborgenen Zustand vergessen werden können.

2. **Generierung eines Kandidaten:** Basierend auf der aktuellen Eingabe und dem gefilterten alten Zustand wird ein neuer Kandidat für den verborgenen

Zustand berechnet. Dieser enthält die neuen Informationen, die potenziell gespeichert werden sollen.

3. **Update:** Das Update-Tor trifft die endgültige Entscheidung über den neuen verborgenen Zustand, indem es den alten Zustand mit dem neuen Kandidaten kombiniert.

Die Ausgabe ist der neue verborgene Zustand, der an den nächsten Zeitschritt weitergereicht wird. Hierdurch lernt das GRU, wichtige Informationen über lange Zeiträume (Sequenzlängen) zu bewahren und irrelevante Informationen direkt zu verwerfen.

Der Encoder ist als bidirektionales, gestapeltes rekurrentes neuronales Netzwerk aufgebaut, welches die GRU-Zellen zur Verarbeitung nutzt. Am Ende der Verarbeitung werden die finalen verborgenen Zustände miteinander verkettet. Das Ergebnis ist ein latenter Vektor h_z von fester Größe, der als komprimierte Repräsentation der gesamten Baugruppe dient und an den Decoder übergeben wird. Auch der Decoder ist als gestapeltes RNN aufgebaut und verwendet GRUs. Aus den rekonstruierten Sequenzen lassen sich direkt die Bauteil-IDs ablesen. So entfällt die Geometrie-Vorhersage, die im PQ-Net notwendig ist. Gleichzeitig kann durch die direkte Ausgabe der Bauteil-IDs sicher mit mehreren Bauteilen gleicher Geometrie umgegangen und ein Mehrschicht-Perzeptron (engl. *Multi-Layer Perceptron*, MLP) eingespart werden. Die Architektur des Sequenz-zu-Sequenz-Encoder-Decoders ist in Abbildung 5.5 dargestellt.

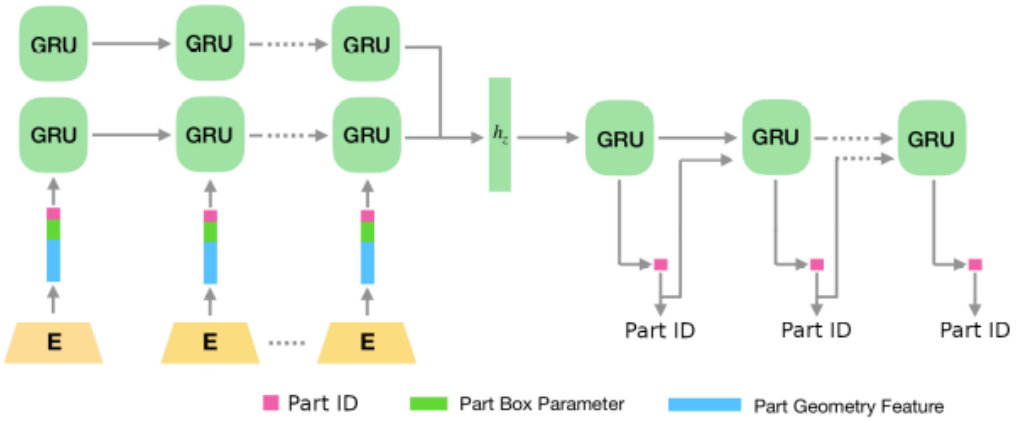


Abbildung 5.5.: Die um die Bauteil-ID erweiterte Struktur der Sequenz-zu-Sequenz-Encoder-Decoder-Architektur [Timmermann et al., 2024d].

5.3. LLM basierte Sequenzgenerierung

Während der zuvor vorgestellte Sequenz-zu-Sequenz-Ansatz eine Montagereihenfolge ausschließlich basierend auf den 3D-Modellen der Bauteile erstellt, bieten Ansätze wie LLMs oder VLMs die Möglichkeit, auch komplexere Wissensbasen einzubinden. Mit diesen Ansätzen ist es möglich, über die reine Sequenzgenerierung hinauszugehen und mögliche Fügeoperationen oder Manipulationssequenzen durch die Netzwerke generieren zu lassen. Nachfolgend wird ein Ansatz vorgestellt, welcher ein LLM für die Generierung eines möglichen Montageplans nutzt und dabei zur Minimierung von Halluzinationen ein VLM als Grounding-Komponente einbindet. Als Informationsquelle nutzt dieser Ansatz eine vollständige CAD-Datei.

Um den Planungsprozess umzusetzen, werden in diesem Ansatz zunächst durch ein LLM eine realisierbare Montagesequenz und anschließend eine mögliche Manipulationssequenz generiert. Hierbei definiert eine Manipulationssequenz nicht ausschließlich die Fügeoperationen für jeden Schritt der Montagesequenz, sondern erweitert diese um zusätzliche robotische Manipulationsschritte, beispielsweise das Greifen eines Bauteils. Für die Sequenzgenerierung erhält das LLM zunächst eine Prompt-Vorlage. Diese definiert die Aufgabe und zusätzlich einen Wissensgraphen, welcher die geometrischen und weitere Informationen enthält, die für die Montage relevant sind. Aus diesen Informationen generiert das LLM eine erste Montagesequenz, die anschließend durch einen konfigurierbaren Validierungsalgorithmus und das VLM als optionale Grounding-Komponente geprüft wird. Der Validierungsalgorithmus kontrolliert zunächst die formale Korrektheit der Ausgabe. Bei einem Nichtbestehen dieser Prüfung wird entsprechendes Feedback an das LLM geschickt. Eine formal korrekte Antwort wird anschließend durch das VLM geprüft. Nach erfolgreicher Generierung der Montagesequenz erfolgt im zweiten Schritt die Generierung der Manipulationssequenz; hierzu wird neben den bereits existierenden Informationen auch eine Liste der möglichen Fähigkeiten (Manipulationsliste) berücksichtigt. Die generierte Manipulationssequenz wird anschließend ebenfalls mit einem Validierungsalgorithmus geprüft. In diesem Fall erfolgt jedoch kein Einsatz des VLM. Der gesamte Ablauf ist in Abbildung 5.6 dargestellt.

5.3.1. Aufbereitung der Eingangsdaten

Der Ansatz verwendet zwei bzw. für das Finden der korrekten Manipulationssequenz drei verschiedene, für ihren Einsatz aufbereitete Datenquellen. Diese Aufbereitung kann dynamisch angepasst werden, ohne dass Änderungen am Ansatz selbst notwendig sind. Dadurch wird es allein durch die Anpassung der Aufbereitung möglich, eine Adaption für neue Montageszenarien durchzuführen. Folgende Eingangsdaten sind erforderlich:

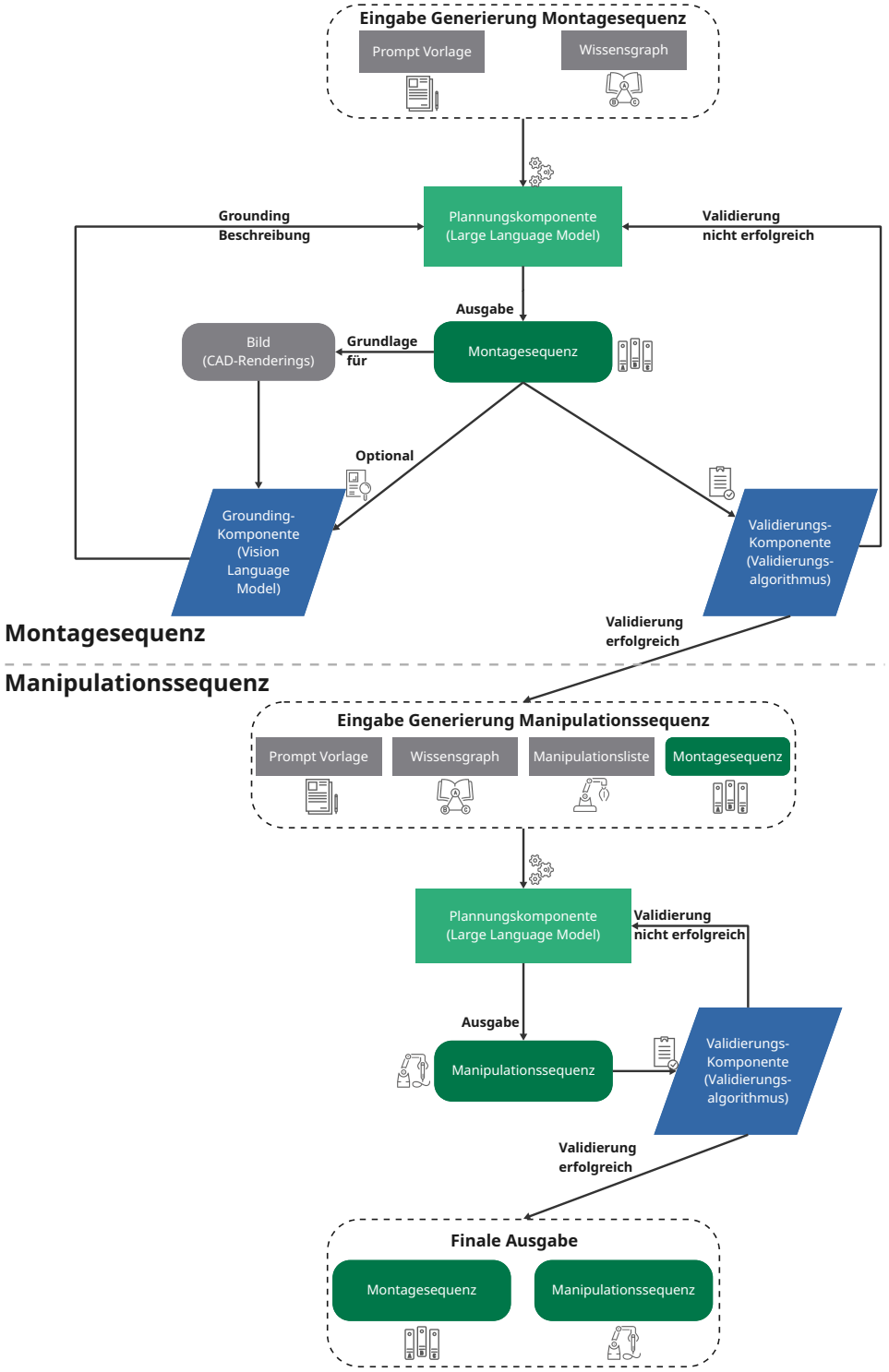


Abbildung 5.6.: Konzept zur Generierung von Montage- und Manipulationssequenzen durch ein LLM auf Basis von CAD-Daten.

- **Wissensgraph:** Der Wissensgraph enthält primär die geometrischen und weitere für die Montage relevante Informationen. Sie werden aus den CAD-Daten, in diesem Fall einer STEP-Datei, extrahiert. Zudem stellen sie die strukturelle Basis für die Montage dar und bilden Aspekte wie Bauteilbeziehungen, Einschränkungen und die räumliche Konfiguration für einen konkreten Anwendungsfall ab.

- **Prompt-Vorlage:** Die Prompt-Vorlage stellt das Verbindungsinterface zwischen den in natürlicher Sprache formulierten Anweisungen und dem LLM dar. Sie unterstützt die Generierung durch das LLM mittels Vorgabe von Strukturen. Dabei können die Vorlagen sowohl szenariospezifisch als auch generalisiert sein. Während die spezifische Vorlage relevant für sehr spezielle Montagesequenzen ist, kann die generalisierte Vorlage notwendige Anpassungen bei einem Szenarienwechsel reduzieren. Durch diese mögliche Varianz im Vorlagendesign wird ein effizienter Einsatz des Ansatzes über verschieden komplexe Montageszenarien hinweg ermöglicht.

- **Manipulationsliste:** Die Manipulationsliste stellt eine Auflistung aller verfügbaren Aktionen dar, welche die Roboter oder die Montagezelle während der Ausführung anbieten. Durch diese Liste werden die physischen Fähigkeiten und Randbedingungen der Robotersysteme für das jeweilige Montageszenario spezifiziert.

Konzept des Wissensgraphen

Der Wissensgraph stellt in diesem Ansatz den Hauptinput für das Framework dar. Er besteht aus zwei grundlegenden Komponenten: Knoten und Kanten. Knoten repräsentieren hierbei einzelne Bauteile oder Unterbaugruppen, während Kanten die möglichen Beziehungen darstellen.

Da Knoten ein Bauteil, eine Kombination von Bauteilen, Unterbaugruppen und die vollständige Baugruppe repräsentieren können, speichern sie die spezifischen Informationen der jeweiligen Formate. Diese individuellen Eigenschaften des repräsentierten Objektes können beschreibende, geometrische, materialbezogene oder graphenbezogene Eigenschaften sein. Eine Übersicht zu allen im Wissensgraphen repräsentierten Informationen ist in Tabelle 5.1 dargestellt.

Die Kanten des Wissensgraphen werden in diesem Ansatz für die Darstellung von Kollisions- und Montagebeziehungen verwendet. Hierfür werden zwei unterschiedliche Kantentypen definiert, die jeweils spezifischen Beziehungstypen zwischen Knoten entsprechen.

Kollisionskanten (engl. Collision Edges) erfassen räumliche Interferenzbeziehungen zwischen zwei Knoten. Aus ihnen leitet sich ab, ob sich Bauteile oder Baugruppen berühren. Da Kollisionen die beiden Objekte betreffen, die durch

Typ	Eigenschaftsname	Beschreibung
Deskriptive Eigenschaften	Eindeutige Identifikationsnummer	Eindeutige ID jedes Objektes, welche dem LLM ermöglicht, zwischen Bauteilen des gleichen Typs (z. B. mehrere gleiche Schrauben) zu unterscheiden.
	Name	Name des Objektes; ermöglicht dem LLM, das Objekt besser zu verstehen und gibt Hinweise auf die Form bzw. die Funktion im Montagekontext.
	Beschreibung	Kurze textuelle Beschreibung des Objektes; ermöglicht dem LLM, das Objekt besser zu verstehen und gibt Hinweise auf die Form bzw. die Funktion im Montagekontext.
Geometrische Eigenschaften	Bounding Box	Definiert einen rechteckigen 3D-Raum, der das Objekt vollständig umschließt. Sie ermöglicht dem LLM eine geometrische Abschätzung des räumlichen Bedarfs.
	Volumen	Informationen über das Volumen des Objektes.
	Massenschwerpunkt	Informationen über den Massenschwerpunkt des Knotens, relativ zum Ursprung des Objektes.
	Abmessungen	Repräsentieren die tatsächlichen Ausdehnungsmaße des Objekts (im Gegensatz zur Bounding Box, die auch Positionsdaten enthält).
Materialeigenschaften	Dichte	Dichte des Objektes.
	Mechanische Eigenschaften	Mechanische Eigenschaften des Objektes, z. B. Elastizität.
	Thermische Eigenschaften	Thermische Eigenschaften des Objektes, aus welchen sich z. B. Fügeoperationen (Schweißen möglich?) ableiten lassen.
Graphenbezogene Eigenschaften	Elternknoten	Die eindeutige ID des Elternknotens. Gibt dem LLM Hilfestellung, die Rolle des Knotens in der Graphenstruktur zu verstehen, indem sie montagebezogene hierarchische Informationen hervorhebt.
	Kinderknoten	Die eindeutigen IDs der Kinderknoten. Gibt dem LLM Hilfestellung, die Rolle des Knotens in der Graphenstruktur zu verstehen, indem sie montagebezogene hierarchische Informationen hervorhebt.

Tabelle 5.1: Im Wissensgraphen gespeicherte Eigenschaften.

die Knoten repräsentiert werden, sind diese Kanten im Graphen ungerichtet. Jede Kollisionskante speichert verschiedene Informationen. Zunächst wird gespeichert, ob überhaupt eine Kollision zwischen den Objekten in zwei Knoten vorliegt. Ist das der Fall, werden zusätzlich die Kontaktpunkte bzw. das Zentrum der Kontaktfläche gespeichert. Sollte keine Kollision zwischen den beiden Objekten vorliegen, speichert die Kante den minimalen Abstand zwischen den Knoten. Während für eine rein theoretische Betrachtung von möglichen Montagesequenzen diese Information keinen Mehrwert bietet, ist sie u. a. bei einer späteren Realisierung mittels einer Roboterzelle relevant, da sie Aussagen darüber erlaubt, ob ein ausreichender Freiraum für die robotische Manipulation der Objekte während des Montagevorgangs vorliegt. Kollisionskanten müssen nur zwischen Knoten derselben Baugruppenebene (engl. Subassembly level) bestimmt werden.

Montagekanten (engl. Assembly Edges) definieren im Gegensatz zu Kollisionskanten eine hierarchische Beziehung zwischen Knoten auf unterschiedlichen Montageebenen. Montageebenen stellen die verschiedenen Granularitäten innerhalb der Baugruppe dar:

- Die niedrigste Ebene besteht aus einzelnen Bauteilen.
- Die Zwischenebene enthält stetig größer werdende Unterbaugruppen.
- Die höchste Ebene repräsentiert die vollständige Baugruppe.

Montagekanten spezifizieren entsprechend, welche Bauteile oder unvollständige Unterbaugruppen die nächsthöhere Unterbaugruppe oder Baugruppe bilden. Montagekanten sind dabei gerichtet und zeigen von den einzelnen Komponenten jeweils auf den zugehörigen Knoten der nächsthöheren Ebene.

Zur Veranschaulichung der verschiedenen Kantentypen ist ein einfaches Szenario in Abbildung 5.7 dargestellt. Da es keine direkte Unterbaugruppe gibt, zeigen in diesem Szenario alle Montagekanten (in Schwarz dargestellt) auf die vollständige Baugruppe. Es existieren somit nur zwei Montageebenen: einzelne Bauteile und die vollständige Baugruppe. Die Kollisionskanten verbinden die einzelnen Bauteile der untersten Montageebene. Grüne Kanten verdeutlichen hierbei Bauteile, die keine direkten Kontaktpunkte aufweisen, und rote Kanten zeigen die Existenz von Kontaktpunkten zwischen zwei Bauteilen an.

Technische Umsetzung des Wissensgraphen

Zum Transfer der CAD-Daten in einen Wissensgraphen wird zunächst die Annahme getroffen, dass das für die Übergabe der Daten genutzte STEP-Format dem Standard AP 214 entspricht. Durch den Standard ist klar definiert, wie essenzielle Informationen wie die 3D-Geometrie, Baugruppenstruktur und geometrische Bemaßungen abgelegt sind, was die automatisierte Verarbeitung ermöglicht. Um auf die entsprechenden Informationen zuzugreifen, wird auf die OpenCASCADE-Bibliothek (OCC) zurückgegriffen. Eine weitere Annahme ist, dass

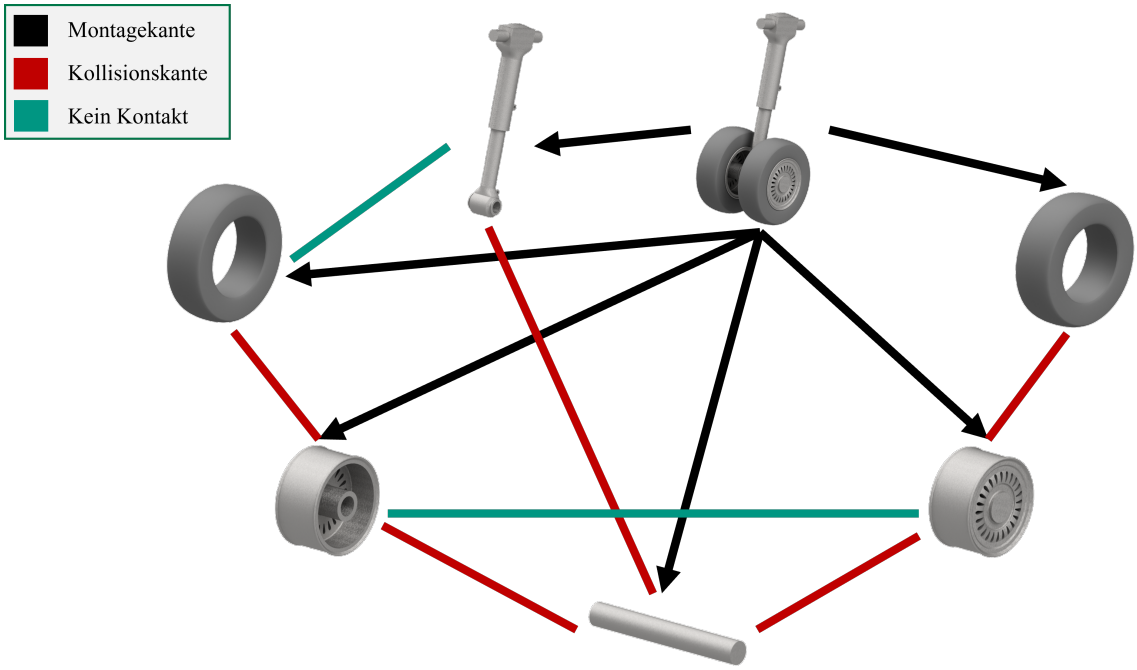


Abbildung 5.7.: Beispielhafte Darstellung der Kantenstruktur des Wissensgraphen für eine Baugruppe eines Flugzeugfahrwerks. Die Montagekanten zeigen vom übergeordneten Knoten zu seinen untergeordneten Knoten, während Kollisionskanten ungerichtet sind.

pro STEP-Datei nur eine Gesamtbaugruppe existiert, also alle Bauteile in einer Beziehung zueinander stehen.

Als zentrales Prinzip für die Datenverarbeitung nutzt OCC *Shapes*, die direkt mit den Entitäten in der STEP-Datei korrespondieren. Die *Shapes* bilden eine hierarchische Struktur, die von *Vertices* bis hin zu *Solids* reicht. Durch *Compounds* wird die Struktur der Baugruppe nachgebildet, indem mehrere *Solids* durch sie zusammengefasst werden und so als eine Unterbaugruppe aufgefasst werden können. Da in einer STEP-Datei die finale Baugruppe jeweils das gesamte Objekt repräsentiert, dient der höchste *Compound* als Wurzelknoten für die Struktur des zu erzeugenden Wissensgraphen.

Der Ablauf der Informationsextraktion aus der STEP-Datei erfolgt nach einem systematischen Ansatz:

1. **Identifikation der Labels:** *Labels* stellen Verwaltungsobjekte für Metainformationen dar und beinhalten Attribute wie Materialeigenschaften oder Komponentennamen. Sie liegen in strukturierter Form abrufbar in der STEP-Datei vor.

2. **Ableiten des Wurzelknotens:** Es wird die *OCC Shape* identifiziert, die den Wurzelknoten darstellt. Hierzu prüft das System, welche *Shape* keine Eltern-Beziehung, also keinen übergeordneten Knoten hat.

3. **Ableitung der Knoten der Graphenstruktur:** Ausgehend von der als Wurzelknoten identifizierten *Shape* erfolgt die rekursive Ableitung der *Child Shapes*. Diese werden mit den entsprechenden Labels verknüpft und bilden die Knoten des Graphen. Hierbei kann die Rekursionstiefe initial angepasst werden, sodass Unterbaugruppen, welche bereits vormontiert zugeliefert werden, nicht tiefergehend analysiert werden müssen. Implementiert werden die Knoten als *Node*-Klasse. Sie kombiniert die Informationen der *Labels* und der *Shapes*. Aus den *Labels* werden die Metadaten, wie Materialeigenschaften und die Eltern-Kind-Beziehungen verwendet, während auf Basis der *Shapes* die Berechnung der geometrischen Eigenschaften durchgeführt wird. Dies sind die Bounding Boxes sowie weitere topologische Attribute wie Volumen, Masse und Dimensionen. Für eine einfachere Handhabung der geometrischen Informationen werden diese in einer *Topology*-Klasse gespeichert.

4. **Ableitung der Kanten der Graphenstruktur:** Auf Basis der zuvor bereits extrahierten *Labels* können die Montagekanten direkt ausgelesen werden. Für die Extraktion der Kollisionskanten werden jedoch zusätzliche Berechnungen durchgeführt. Aus Gründen der Recheneffizienz wird die Extraktion nicht direkt auf den in OCC definierten *Shapes* durchgeführt, sondern auf Mesh-Objekten der *Shapes*. Die Mesh-Objekte approximieren die *Shapes* durch eine Darstellung aus Vertices und dreieckigen Polygonen. Für eine effiziente Berechnung wird anschließend die *Flexible Collision Library* (FCL) verwendet, welche die Objekte in einer *Bounding Volume Hierarchy* (BVH) darstellt. In dieser Struktur können effizient mögliche Kollisionen zwischen Bauteilen einer Baugruppe berechnet werden. Hierzu wird zunächst pro Bauteil geprüft, ob die anderen Bauteile über oder unter einer Toleranzschwelle im Hinblick auf die Entfernung zwischen den Bauteilen liegen. Bauteile mit einer Entfernung unter der Toleranzschwelle gelten als kollidierend, Bauteile über der Toleranzschwelle als nicht kollidierend. Die Toleranzschwelle ist notwendig, da durch die Approximation der Shapes Ungenauigkeiten entstehen. Diese Toleranzschwelle beträgt aktuell einen Millimeter und kann jederzeit angepasst werden.

5. **Serialisierung der Graphenstruktur:** Zur Verarbeitung der Graphenstruktur durch das LLM ist eine Serialisierung in ein für das LLM einfach zu verarbeitendes Datenformat notwendig. Als Datenformat kommen hierzu verschiedene Formate in Betracht: *XML*, *GraphML* oder *JavaScript Object Notation (JSON)*. Verwendet wird final das Datenformat *JSON*, da es durch eine kompaktere Darstellung weniger Tokens bei der Eingabe in das LLM benötigt, ohne zeitgleich die Lesbarkeit für den Menschen einzuschränken. Für die Serialisierung werden in der *JSON*-Struktur die Knoten und Kanten mit ihren jeweiligen Attributen (z. B. IDs, geometrische Eigenschaften und Relationen) als Schlüssel-Wert-Paare (*key-value pairs*) gespeichert.

Konzept der Prompt-Vorlage

Die Prompt-Vorlage gibt die eigentliche Montageaufgabe vor und kontextualisiert die szenariospezifischen Eingabeinformationen. Wie in Kapitel 2.2.7 definiert, besteht ein Prompt aus einer textuellen Anweisung, die an ein sprachbasiertes Foundation Model gesendet wird. Das hat zur Folge, dass Fähigkeiten, welche das vortrainierte Modell besitzt, auf eine neue Aufgabe übertragen werden können. Die Prompt-Vorlage fungiert hierbei als Grundgerüst von Anweisungen, das mit szenariospezifischen Informationen, z. B. dem Wissensgraphen, kombiniert wird, um den Prompt zu generieren, der schlussendlich durch das LLM verarbeitet wird.

Die inhärente Problemkomplexität und die daraus resultierende Promptlänge erfordern den Einsatz eines autoregressiven Modells. Entsprechend nutzen alle Prompts das Prefix-Prompt-Format. Dieses Format ermöglicht eine autoregressive Fortführung eines initialen Textes. Die Prompt-Vorlagen erfüllen zwei primäre Funktionen: die Definition des Kontextes und der Aufgabe für das Foundation Model sowie die Erläuterung der anwendungsfallspezifischen Informationen.

Definition des Kontextes und Aufgabe:
Die Definition des Kontextes, robotische Montageanwendung, ist über die gesamte Ausführungszeit für alle eingesetzten Modelle konsistent, sodass der Kontext über alle Vorlagen gleich definiert werden kann. Die Aufgabe hingegen ist abhängig vom aktuellen Zeitpunkt in der Ausführung. Beim Einsatz in Kombination mit dem LLM bedeutet dies die Interpretation von szenariospezifischen Informationen, wie dem Wissensgraphen oder der Antwort des VLMs, sowie die Generierung der Montagesequenz und der Manipulationssequenz. In Verbindung mit dem VLM besteht die Aufgabe primär in der Beschreibung, der Analyse und der Interpretation von CAD-Renderings.

Der Klassifikation von Chen et al. [168] folgend werden grundlegende Prompt-Engineering-Methoden verwendet, um die Kontext- und Aufgabendefinition zu verbessern. Für die Etablierung des Kontextes wird rollenbasiertes Prompting eingesetzt, indem dem Foundation Model eine spezifische Expertenrolle zugewiesen wird. Auch werden klare und präzise Anweisungen verwendet, um präzise und spezifische Antworten der Modelle zu generieren. Zusätzlich unterstützt eine klare Promptstruktur die Erfassung durch das Modell. Hierzu werden verschiedene Separatoren wie dreifache Anführungszeichen, Hashtags und Abschnittsüberschriften verwendet, um zwischen Aufgabenbeschreibung, Kontexterläuterung und den szenariospezifischen Informationen zu differenzieren.

Erläuterung der szenariospezifischen Informationen:
Diese Fähigkeit wird durch Abschnitte in der Prompt-Vorlage realisiert, welche die szenariospezifischen Daten, insbesondere das Datenformat, beschreiben. Während LLMs in früheren Arbeiten bereits zeigten, dass sie Graphenstrukturen verstehen können, sind die Beschreibungen der Bedeutung und Funktion von Knoten und Kanten notwendig, da Foundation Models nicht auf den für Montagean-

wendungen typischen Formaten der Wissensgraphen trainiert wurden. Zur Minimierung von szenariospezifischen Anpassungen werden allgemeinere Beschreibungen der Eigenschaften gegenüber hochspezialisierten, anwendungsfallspezifischen Beschreibungen bevorzugt.

Neben Aufgaben-, Kontext- und Eingabebeschreibung können Beispielantworten helfen, die Ausgabe der Foundation Models zu steuern. Dies wird als One-Shot-Prompting bezeichnet, wenn ein Beispiel genutzt wird, oder Few-Shot-Prompting, wenn mehrere Beispiele genutzt werden.

Im Sinne einer allgemeineren Beschreibung werden hierbei wiedererkennbare Szenarien verwendet, etwa einfache Baugruppen. Dies kann dabei unterstützen, eine fehlerhafte Generalisierung zu vermeiden, die bei zu speziellen Szenarien auftreten kann. So können fehlerhafte Ausgaben minimiert werden. Durch die Wahl eines allgemeingültigen Beispiels wird daher für die Foundation Models sichergestellt, dass der Fokus auf dem strukturierten Vorgehen bei der Montageplanung liegt, anstelle von domänenspezifischen Details, die möglicherweise nicht auf das aktuelle Szenario übertragen werden können.

Da in diesem Ansatz sowohl LLMs als auch VLMs eingesetzt werden und das LLM unterschiedliche Aufgaben im Rahmen eines Durchlaufes ausführt, werden verschiedene unabhängige Prompt-Vorlagen erzeugt. Durch die automatische Integration der anwendungsspezifischen Informationen in die jeweilige Vorlage, welche dann die finale Eingabe für das Foundation Model generiert, wird eine effiziente Wiederverwendbarkeit der Vorlagen ermöglicht. So ist der Transfer in andere Montageszenarien möglich, während weiterhin die Option besteht, szenariospezifische Anforderungen und Einschränkungen zu integrieren.

Technische Umsetzung der Prompt-Vorlagen

Alle Prompt-Vorlagen sind im *JSON*-Format erstellt und enthalten Sektionen für Aufgabendefinition, Kontext und Erläuterungen zu den für das LLM nicht geläufigen Datenformaten, insbesondere dem Wissensgraphen und der Manipulationsliste. Spezielle Anweisungen in den Vorlagen wirken bekannten Schwachstellen der Modelle entgegen. Hierbei wird ein vereinfachtes rollenbasiertes Prompting eingesetzt, welches dem LLM hilft zu verstehen, dass es Montageplanung für die robotische Montage durchführt. Ein größerer Fokus liegt jedoch auf der Kontextualisierung. Für Aufgaben, die eine speziell formatierte Ausgabe benötigen, werden durch Few-Shot-Prompting anwendungsfallunabhängige Beispiele integriert. Sie bieten dem System eine Orientierung, ohne dass falsche schlussfolgernde Analogien zum aktuell vorliegenden Montageproblem provoziert werden. Das Framework selbst steuert während der Ausführung dann dynamisch die Auswahl und Komposition der geeigneten Vorlagen und Informationen für den jeweiligen Ausführungsschritt.

Das Konzept der Manipulationsliste

Ein weiterer anwendungsfallspezifischer Input ist die Manipulationsliste. Sie wird ausschließlich im Generierungsschritt der Manipulationssequenz genutzt und beeinflusst die Erstellung der grundlegenden Montagesequenz nicht. Sie definiert alle Manipulationsaktionen, welche der Roboterzelle zur Durchführung der Montage zur Verfügung stehen, und dient damit als Beschränkung des Aktionsraumes auf vorhandene Fähigkeiten. Da das Ziel der Arbeit darin liegt, final einen symbolischen Montageplan zu generieren, ist die Manipulationsliste bewusst auf einem hohen Abstraktionsniveau angelegt. Dadurch wird ermöglicht, sie in einem einfach menschenlesbaren Format zu realisieren, sodass sie mit Informationen wie: „`Führe Trajektorie zu < Pose 1 > aus`“ oder „`Greife < Objekt 1 >`“ gefüllt werden kann. An dieser Stelle wird davon ausgegangen, dass diese Funktionen intern weiter in ihre Low-Level-Befehle unterteilt werden, z. B. die Regelung auf spezifische Gelenkwinkel. Durch die gewählte Abstraktion wird das LLM darin bestärkt, sich auf die Auswahl der korrekten Manipulationen sowie einer logischen Anordnung zu fokussieren. Die Einbindung von Low-Level-Befehlen oder die automatische Erstellung der Manipulationsliste - diese wird aktuell händisch durch den Anwender erzeugt - stellen Potenziale für zukünftige Entwicklungen dar. Durch die Manipulationsliste wird das LLM darin unterstützt, realistische Manipulationssequenzen zu generieren und die Wahrscheinlichkeit von Halluzinationen und/oder die fehlerhafte Anordnung von Manipulationssequenzen zu reduzieren. Da viele Montageszenarien mit komplexen Baugruppen nicht mit einem einzigen Robotersystem realisierbar sind, ist das Konzept der Manipulationsliste explizit darauf ausgelegt, verschiedene Robotersysteme abzudecken. Dazu wird pro Robotersystem eine eigene Manipulationsliste erstellt, die ausschließlich die Fähigkeiten des jeweiligen Systems beschreibt. Das LLM berücksichtigt anschließend alle verfügbaren Listen. Dadurch wird es möglich, die spezialisierten Funktionen verschiedener Roboter, beispielsweise Schwerlastroboter und Roboter mit Werkzeugwechselsystemen, zu berücksichtigen. Gleichzeitig stellen die Listen sicher, dass die generierten Manipulationssequenzen die operativen Beschränkungen und Fähigkeiten jedes Roboters beachten.

Für eine einheitliche Datenverarbeitung wird auch die Manipulationsliste als JSON-Format dem LLM zur Verfügung gestellt. Hierbei besteht die Option, die möglichen Manipulationsfähigkeiten und ihre Zuordnung zu einem entsprechenden Robotersystem entweder automatisiert abzuleiten oder, sollten diese Informationen nicht vorhanden sein, die Liste händisch zu erstellen. Wird ein Szenario betrachtet, in dem mehrere Robotersysteme vorhanden sind, so besitzt, wie bereits beschrieben, jedes Robotersystem eine eigene Manipulationsliste, die referenziert wird.

5.3.2. Montagesequenzgenerierung

An der Generierung der Montagesequenz sind neben den Eingangsdaten in Form von Wissensgraph und Prompt-Vorlage das Planungsmodul (welches auf das LLM zugreift), der Validierungsalgorithmus sowie die Grounding-Komponente beteiligt, wobei die zuletzt genannte Komponente das VLM integriert.

Planungskomponente

Bei der Planungskomponente handelt es sich um die zentrale Komponente der Sequenzgenerierung. Sie greift direkt auf das LLM zu, um strukturierte Informationen zu interpretieren und durch iterative Verfeinerung realisierbare Montagesequenzen zu generieren. Jegliche Ausgabe wird in der Planungskomponente erzeugt bzw. auch auf Basis von Feedback des Validierungsalgorithmus und der Grounding-Komponente angepasst. Für die Realisierung der Planungskomponente sind sowohl LLMs als auch VLMs als mögliche Technik untersucht worden. Bewertet wurden die beiden Techniken vor allem anhand der folgenden Anforderungen: Die Fähigkeit, Information aus verschiedenen komplexen Quellen zu synthetisieren; die Handhabung eines umfangreichen und über den Verlauf der Ausführung wachsenden Kontexts sowie das Erzeugen einer strukturierten Ausgabe. Die aktuelle Forschung zeigt, dass die Stärken von LLMs diese Anforderungen besser erfüllen, während VLMs Vorteile beim Einsatz als Grounding-Komponente aufweisen.

Die primären Funktionen der Planungskomponente lassen sich unterteilen in die Informationsverarbeitung und die Sequenzgenerierung.

Im Rahmen der Informationsverarbeitung ist die Komponente dafür zuständig, Eingabequellen zu interpretieren und zu synthetisieren sowie relevante Informationen aus Prompts verschiedener Komplexität zu filtern. Im initialen Durchlauf muss sie den Kontext, die Aufgabe und die Informationen aus dem Wissensgraphen extrahieren. Im Zusammenhang mit dem Wissensgraphen bedeutet dies vor allem, dass ein grundlegendes Verständnis für die Graphenstruktur und die darin definierten Beziehungen, also die Montageabhängigkeiten und die Kollisionsabhängigkeiten, durch die Komponente möglich ist. Da die Planungskomponente iterativ zu einer möglichen Lösung finden soll, ist in den nachfolgenden Aufrufen besonders die Berücksichtigung von Feedback relevant. Nach der initialen Sequenzgenerierung erhält die Planung Feedback. Entweder durch die Validierungskomponente bezüglich der formalen Korrektheit der bisherigen Ausgabe oder durch die Grounding-Komponenten, die eine Beschreibung des aktuellen Status der Montage liefert. An dieser Stelle ist es die Aufgabe des in der Planungskomponente enthaltenen LLMs, dieses Feedback zu interpretieren und mit den bereits vorhandenen Informationen zu kombinieren, um die Sequenz ggf. zu verbessern. Hierbei stellt das Kontextmanagement eine besondere Herausforderung dar, die vor allem über mehrere Verbesserungszyklen hinweg zum Tragen kommt. Durch das reine Anhängen von neuen Informationen an den Prompt

wächst dessen Länge auf ein Niveau, welches den Rechenaufwand und die Komplexität der Aufgabe stark erhöht. Daher ist eine intelligente Filterung der Informationen notwendig, um zwischen für den Kontext relevanten und redundanten Informationen zu unterscheiden.

Bei der Aufgabe der Sequenzgenerierung generiert die Komponente aus den verarbeiteten Daten eine mögliche Montagesequenz. Nach der Definition von Ghandi et al. [7] ist diese definiert als das Finden einer möglichen Sequenz, welche sich als eine Aneinanderreihung von kollisionsfreien Montageschritten darstellen lässt und dessen Ausführung zur Montage des finalen Produktes führt. Da die Montagesequenz nur Angaben über die Reihenfolge macht, in welcher die Bauteile zu verwenden sind, jedoch keine Informationen über das Zustandekommen (wie?) enthält, wird bei dieser Betrachtung angenommen, dass es für jeden Montageschritt eine mögliche Manipulationssequenz gibt, die diesen Schritt realisieren kann. Abbildung 5.8 verdeutlicht den Unterschied zwischen einer realisierbaren und einer nicht realisierbaren Montagesequenz. Hierbei ist bei der nicht realisierbaren Sequenz die Anforderung der physikalischen Korrektheit verletzt. Ein kritischer Aspekt bei der Sequenzgenerierung ist die Einhaltung eines fest vorgegebenen Formates der Ausgabe. Durch Einhaltung des vorgegebenen Formates ist sichergestellt, dass die generierte Sequenz anschließend durch die anderen Komponenten verarbeitet und geprüft werden kann, etwa durch den Validierungsalgorithmus oder für die Generierung der Renderings für die visuelle Verifikation. Das ist allerdings nur möglich, wenn in einer Ausgabe alle Angaben konsistent sind und sich die Daten automatisiert extrahieren lassen.

Durch die Umsetzung der Planungskomponente als LLM ist das mögliche Ein- und Ausgabeformat auf Text begrenzt. Daraus ergibt sich, dass für den Wissensgraphen eine Überführung in ein Textformat notwendig ist und dieses den möglichen Informationsverlust minimieren sollte (siehe dazu auch 5.3.1). Durch das Textformat der Ausgabe ist für sie ebenfalls eine Weiterverarbeitung notwendig, damit die Pipeline automatisiert ausgeführt werden kann. Eine solche Weiterverarbeitung gilt insbesondere bei der Überführung der einzelnen Montageschritte in ausführbare OCC-Operationen, sodass ein Rendering der entsprechenden Montagezustände möglich ist. Somit kann die Grounding-Komponente genutzt werden.

Durch die Anforderung, große Eingabe-Prompts einerseits mit vielen Informationen und anderseits eine sich ändernde Ausgabe zu handhaben, ist die Wahl der möglichen LLM-Architekturen eingeschränkt:

- **Encoder-only-Modelle** sind nicht verwendbar. Obwohl sie das standardisierte Format einhalten, könnten sich für diese Architektur zu große Variationen in der Anzahl und den Typen der Montageschritte für verschiedene Modelle ergeben.
- **Encoder-Decoder-Modelle** sind ebenfalls nicht verwendbar. Auch wenn sie Vorteile durch den Encoder in der Vorverarbeitung der Eingabe haben können, ist die sehr beschränkte Größe des verarbeitbaren Kontexts ein Aus-

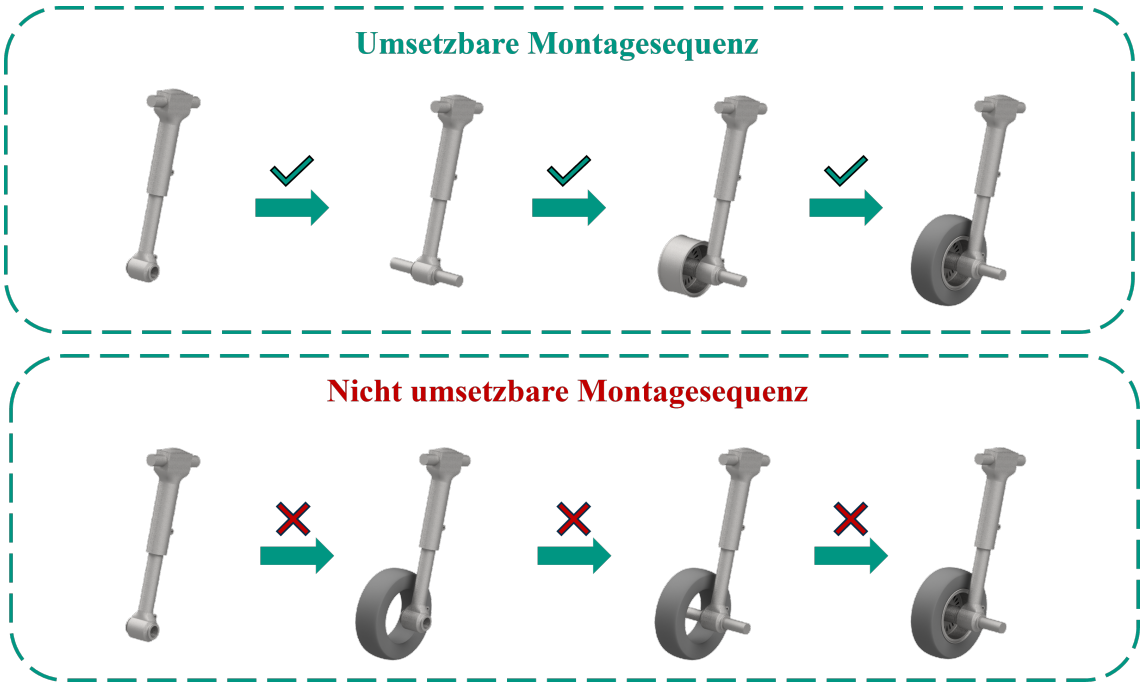


Abbildung 5.8.: Vergleich von realisierbaren und nicht realisierbaren Montagesequenzen für ein Flugzeugfahrwerk. Die untere Reihenfolge ist unmöglich, da der Reifen frei schwebend montiert wird, was keine physikalisch stabile Lösung darstellt.

schlusskriterium. Diese Beschränkung entsteht durch den höheren Speicherbedarf, durch die encoded Repräsentationen und die Decoder States. Aktuelle große Encoder-Decoder-Modelle, beispielsweise GLM-130B [169], können nur einen Prompt-Input von knapp über 2000 Tokens verarbeiten.

- **Decoder-only-Modelle** sind in der Lage, mit großen Prompt-Eingabe-Längen zu arbeiten. Modelle wie Qwen3 [170] können über 30.000 Eingabezeichen verarbeiten und variable Ausgabeformate realisieren.

Technische Umsetzung der Planungskomponente

Zur technischen Umsetzung der Planungskomponente für die Sequenzgenerierung greift das System umfassend auf die durch das Framework zur Verfügung gestellte Verarbeitungspipeline zurück. Da im Rahmen dieser Arbeit kein Finetuning der verwendeten Foundation Models aufgrund der geringen zur Verfügung stehenden Datenmenge sowie der nutzbaren Trainingshardware umsetzbar ist, konzentriert sich die technische Umsetzung vor allem auf die gezielte Reduktion von Aufgabenkomplexität und Informationsmenge, die das Modell während der Inferenz verarbeiten muss.

Zur Reduktion der generellen Komplexität der Planungsaufgabe wird dem LLM initial nicht die vollständige Baugruppe zur Verfügung gestellt, sondern die in

der Gesamtbaugruppe enthaltenen Unterbaugruppen. Die Informationen über die existierenden Unterbaugruppen können direkt aus dem Wissensgraphen abgeleitet werden, indem geprüft wird, welche Bauteile/Komponenten jeweils den gleichen Elternknoten haben. Durch die Unterteilung der Aufgabe in Unterbaugruppen wird die Komplexität der zu lösenden Planungsaufgabe für das LLM zunächst deutlich verkleinert und es müssen nur die dafür relevanten Knoten und Kanten aus dem Wissensgraphen serialisiert und in den Eingabeprompt eingefügt werden. Dadurch reduziert sich die Promptlänge.

Als zusätzliche Maßnahme zur Komplexitätsreduzierung wird die initiale Lösungsfindung und die Aufbereitung der Lösung in zwei separate Prompts unterteilt. Bei der Verarbeitung des ersten Prompts muss das LLM zunächst alle verfügbaren Informationen interpretieren und darauf basierend eine Lösung formulieren. Durch den zweiten Prompt erfolgt dann die Transformation der gefundenen Lösung in das gewünschte Ausgabeformat. Durch diesen Ansatz werden die Fähigkeiten der Chain-of-Thought-Modelle besser genutzt, da nicht nur die finale formatierte Ausgabe betrachtet wird, sondern auch die Gedankenkette, welche zur Lösung geführt hat.

Eine weitere Maßnahme zur Reduzierung der Datenmenge durch die Verarbeitung von LLMs ist, dass nur die Kollisionskanten in den Prompt übernommen werden, die auch tatsächlich eine Kollision zwischen den beiden beteiligten Knoten anzeigen. Näherungsbeziehungen zwischen sich nicht berührenden Komponenten können trotz der fehlenden Kanteninformationen vom LLM weiterhin über die verfügbaren Bounding-Box-Informationen abgeleitet werden. Durch diese Maßnahme ist es möglich, die Promptlänge zu reduzieren.

Um die benötigte Tokenlänge möglichst gering zu halten und dem LLM nur die Informationen zur Verfügung zu stellen, welche es für die aktuelle Aufgabe benötigt, umfasst der Eingabeprompt je nach aktuellem Zustand der Ausführung stets nur ausgewählte Informationen. In jedem Prompt sind zunächst die spezifischen Informationen der Baugruppe/Unterbaugruppe enthalten, die aktuell zu betrachten sind, da sie als Entscheidungsgrundlage für den Planungsprozess fungieren. Während des Formatierungsschrittes enthält der Prompt zusätzliche Informationen über die notwendige Formatierung. Nach der Validierung ist der Prompt erweitert durch die gefundene Lösung sowie das Ergebnis des Validierungsalgorithmus. Im Grounding-Schritt erhält das LLM die Beschreibung der durch das VLM erkannten Szene sowie eine Beschreibung der Aufgabenstellung an das VLM. Abbildung 5.9 stellt diesen Ansatz schematisch dar.

Validierungskomponente

Die Aufgabe der Validierungskomponente besteht darin, die von der Planungskomponente generierten Montagesequenzen durch algorithmische Methoden zu analysieren und auf ihre formale Korrektheit hin zu prüfen. Das Ergebnis wird

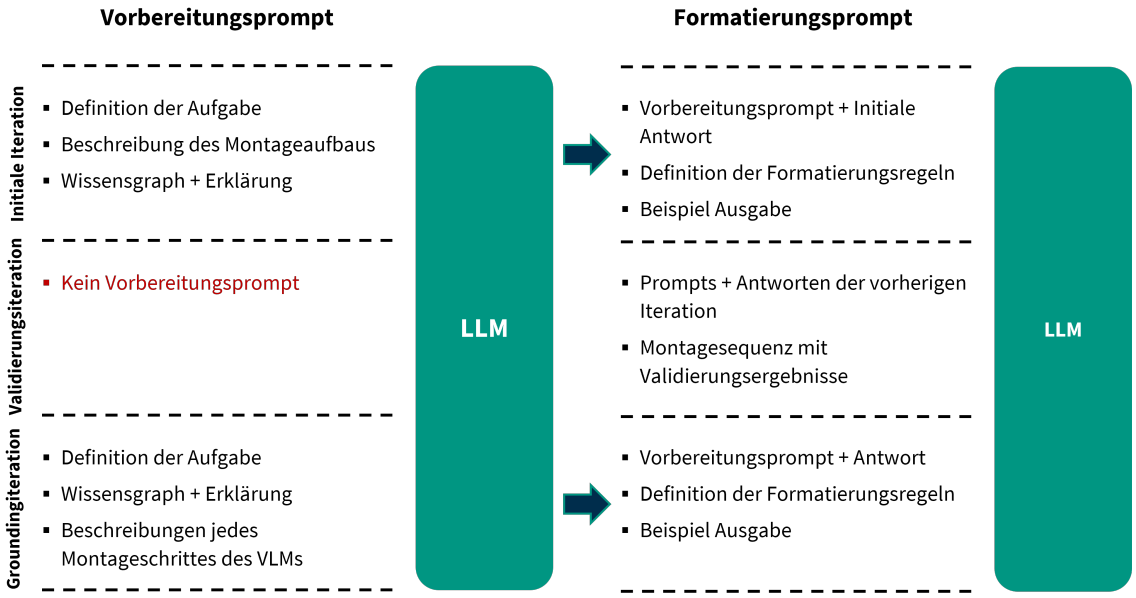


Abbildung 5.9.: Schematische Darstellung der Informationen, die in die zwei LLM-Aufrufe der Planungskomponente eingehen. Durch eine gezielte Auswahl der Informationen für den Prompt wird die Anzahl der benötigten Tokens minimiert.

in der Form von detailliertem Feedback an die Planungskomponente zurückgespiegelt, sodass eine iterative Verfeinerung und Verbesserung der generierten Sequenz ermöglicht wird. Hierbei nimmt die Komponente sowohl eine Validierung als auch eine Verifikation der Sequenz vor, da sie sowohl die strukturelle Korrektheit überprüft als auch die inhaltliche Korrektheit der Montagesequenz. Validierung bezeichnet in diesem Kontext die inhaltliche Korrektheit der generierten Sequenz in Bezug auf die Aufgabenstellung und der darin enthaltenen spezifischen Anforderungen. Verifikation bezieht sich hingegen auf die formale Korrektheit der generierten Ausgabe in Bezug auf vorgegebene Formatierung, Regeln und Strukturen [171].

Bei der Prüfung der Montagesequenz betrachtet der Algorithmus in der Validierungskomponente drei verschiedene Fehlerkategorien: Verifikation der formalen Korrektheit, Korrektheit der Montagelogik und Konsistenz des Wissensgraphen. Die Überprüfung aller drei Kategorien erfolgt dabei unabhängig voneinander, sodass sowohl eine sequenzielle als auch eine parallele Überprüfung erfolgen kann. Eine sequenzielle Überprüfung ist jedoch in vielen Fällen die robustere Variante, da Fehler in einer der Fehlerkategorien Folgefehler in anderen Kategorien hervorrufen können. Sollten alle drei Fehlerkategorien ohne Auffälligkeiten geprüft werden, so gilt der Algorithmus als erfolgreich durchlaufen.

Bei der Prüfung der formalen Korrektheit prüft der Algorithmus die Vollständigkeit der Sequenz, die interne Konsistenz und die Einhaltung der korrekten Formatierungs- und Syntaxvorgaben aus der Prompt-Vorlage. Diese Überprüfung erkennt zuverlässig unvollständige Ausgaben, wenn z. B. das LLM nicht

in der Lage ist, eine vollständige Montagesequenz zu generieren. Auch Rechtschreibfehler in der Ausgabe werden erkannt. So können die Fehlinterpretation von Bauteilnamen und deren Beschreibungen verhindert werden.

Um die Korrektheit der Montagelogik sicherzustellen, analysiert der Algorithmus die vorliegende Sequenz in Bezug auf die Einhaltung der grundlegenden Montageprinzipien und die Berücksichtigung von physikalischen Einschränkungen. Dazu wird untersucht, ob logische Abhängigkeiten in der Sequenz berücksichtigt und die physikalischen Beziehungen zwischen Bauteilen korrekt sind. So können Fehler wie die falsche Positionierung von zwei Bauteilen zueinander erkannt und adressiert werden.

Die Konsistenz des Wissensgraphen stellt eine essenzielle Machbarkeitsprüfung der vorliegenden Sequenz dar. In diesem Schritt validiert der Algorithmus, ob alle im Wissensgraphen definierten Bauteile in der Sequenz verwendet werden und ob die pro Sequenzschritt montierten Bauteile auch wirklich physikalischen Kontakt zueinander haben. Hierzu nutzt der Algorithmus geometrische, räumliche und relationale Informationen, die aus dem CAD-Modell extrahiert werden können. Hierdurch kann auch geprüft werden, ob verbaute Komponenten sich über den Verlauf der Sequenz nicht durch Halluzinationen des LLMs bewegen, sondern an den korrekten Positionen verbleiben.

Eine entscheidende Herausforderung im Zusammenhang mit der Validierungskomponente stellt die Variabilität der vom LLM generierten Ausgabe dar. Diese kann über verschiedene Ausführungszyklen semantisch korrekt, aber syntaktisch unterschiedlich sein. Die Fehlerquelle kann in der generativen Art der LLMs begründet sein. Da es aufwendig ist, Validierungschecks zu entwickeln, die eine absolute Garantie bezüglich der Realisierbarkeit einer Sequenz bieten, ist das Ziel der Validierungskomponente vor allem, kategorisch unmögliche Lösungen zu identifizieren, im Sinne einer Vorfilterung der Lösungen. Auf dieser reduzierten Lösungsmenge baut anschließend das in der Architektur nachgelagerte Grounding auf.

Technische Umsetzung der Validierungskomponente

Bei der Validierungskomponente wird eine algorithmische Überprüfung eingesetzt anstelle von anderen Validierungstechniken, wie beispielsweise der Verwendung eines separaten LLMs für die Analyse. Diese Entscheidung wurde getroffen, da der algorithmische Ansatz diverse Vorteile gegenüber einer LLM-basierten Lösung als Validierungskomponente besitzt:

- **Geringere Rechenintensität:** Das Durchlaufen des Algorithmus benötigt weniger Rechenressourcen und ist schneller abgeschlossen, da ausschließlich deterministische Regeln abgeprüft statt Inferenzen durchgeführt werden.

- **Spezifisches Feedback:** Der algorithmische Ansatz generiert automatisch spezifisches Feedback durch die Identifikation von Fehlerarten und Fehlerursprüngen, während die Ausgaben des LLMs ohne spezialisierte Prompt-Vorlage generischer ausfallen können.
- **Dynamische Anpassbarkeit:** Die verschiedenen Überprüfungen können dynamisch und unabhängig voneinander angepasst werden. Das ermöglicht besonders in der Testphase eine schnelle Adaption.
- **Nachvollziehbarkeit:** Die Fehler, die zu einem Nichtbestehen der Prüfung führen, können klar nachvollzogen werden und erlauben eine systematische Leistungsanalyse. Bei einem LLM kann die Korrektheit bzw. die Nachvollziehbarkeit einer Ausgabe nicht garantiert werden.

Bevor die Ausgabe des LLMs durch den Algorithmus geprüft werden muss, ist es notwendig, den relevanten Abschnitt mit der Montagesequenz in der Ausgabe zu identifizieren. Dies ist vor allem bei Chain-of-Thought (CoT) Modellen eine Herausforderung, da diese neben ihrer eigentlichen Lösung auch die gesamte Herleitungskette in der Ausgabe mitliefern. Da CoT-Modelle ihre Ausgabe stets so strukturieren, dass die Herleitungskette vor der finalen Antwort angeführt wird, ist eine Bottom-up-Strategie für die Extraktion der Montagesequenz möglich. Zunächst prüft das System die finale Zeile der Ausgabe auf den Bezeichner „`Assembly sequence formatted successfully`“, da dies in der Prompt-Vorlage als Abschluss einer Montagesequenz definiert ist. Anschließend wird jede Zeile der Ausgabe von diesem Punkt aufsteigend geprüft, bis die erste Zeile gefunden wird, die mit einer „`1`“ beginnt. Diese ist als Starttoken in der Prompt-Vorlage definiert und zeigt an, dass jegliche Ausgabe über dieser Zeile nicht mehr zur Montagesequenz gehört und nicht weiter betrachtet werden muss. Innerhalb der so ausgewählten Ausgabe werden für eine effiziente Prüfung alle notwendigen Informationen extrahiert, sodass pro Montageschritt die IDs der beteiligten Knoten/Bauteile identifiziert werden. Der Extraktionsprozess implementiert dabei mehrere Fallback-Lösungen, um auf verschiedene Formatierungsfehler durch das LLM, z. B. Schreibfehler, zu reagieren und diese an das LLM rückzumelden. Neben der Überprüfung auf einfache Formatierungsfehler wird im Rahmen der Überprüfung der formalen Korrektheit auch kontrolliert, ob die extrahierte Sequenz sich mit einer der in der Prompt-Vorlage angegebenen Strukturen deckt. Dies findet durch Prüfung mittels umfangreicher regulärer Ausdrücke (Regex) statt und stellt sicher, dass die Struktur für die weitere automatisierte Verarbeitung eingehalten und geeignet ist.

Bei der Überprüfung der Einhaltung der korrekten Montagelogik sind insbesondere Erreichbarkeitsanalysen für die Endlage der Bauteile notwendig, um u. a. zu überprüfen, ob ein Bauteil kollisionsfrei an die Montagestelle bewegt werden kann. Hierzu werden die bereits montierten Bauteile mittels der $TopoDS\_Shape$-Klasse aus Open CASCADE in ein fusioniertes Oberflächenmodell zusammengeführt und anschließend das zu betrachtende Bauteil von seiner Ausgangslage zu seiner endgültigen Montageposition bewegt. Wird bei dieser Bewegung festgestellt, dass das Bauteil die bereits montierten Komponenten berührt bzw. in der

Endlage von diesen vollständig umschlossen wird, so gilt das Bauteil in dieser Form als nicht montierbar.

Für den Abgleich mit dem Wissensgraphen wird pro Montageschritt geprüft, ob die im Schritt enthaltenen Bauteile korrespondierende Knoten im Wissensgraphen haben und ob zwischen den Knoten Kollisionskanten existieren, sodass diese in ihrer Endlage miteinander Kontakt aufweisen. Auch wird in diesem Schritt überprüft, ob sich die beteiligten Knoten alle auf der gleichen Montageebene befinden. Auf diese Weise wird die Einhaltung des Prinzips der Unterbaugruppen sichergestellt. Nach Prüfung der individuellen Montageschritte erfolgt die Kontrolle, ob alle Knoten im Wissensgraphen verarbeitet wurden und ob einer dieser Knoten mehrfach verwendet wurde, also ein Bauteil mehrfach in der Montage verwendet wird.

Das Ergebnis der Überprüfungen wird konsolidiert und um entsprechendes spezifisches Feedback erweitert, z. B. über nicht verwendete Bauteile. Anschließend wird dieses als Textprompt an das LLM zurückgegeben.

Da es sich bei der Validierungskomponente nicht um ein neuronales Netz, sondern um einen klassischen Algorithmus handelt, kann dieser lokal auf dem System ausgeführt werden und erfordert keine leistungsstarke GPU. Dadurch ist es möglich, die GPU-Ressourcen für das LLM und VLM bereitzuhalten, was ein zusätzliches Entladen und späteres Neuladen der Modelle aus und in den VRAM verhindert.

Grounding-Komponente

Als dritte Komponente im Prozess der Montagesequenzgenerierung übernimmt die Grounding-Komponente die Aufgabe der visuellen Verifikation. Dies erfolgt durch die automatisierte Interpretation von CAD-Renderings der verschiedenen Montagezustände und ermöglicht so neben der Validierungskomponente ein zusätzliches Grounding. Dieser duale Grounding-Mechanismus ermöglicht es dem LLM, visuelle Gegebenheiten gegen das aus dem Wissensgraphen abgeleitete Verständnis abzugleichen. Durch dieses Vorgehen werden Diskrepanzen zwischen der aktuellen Montagesequenz und der korrekten Logik im Wissensgraphen deutlicher sichtbar und ermöglichen so einen besseren Verfeinerungsprozess.

Wie im Stand der Technik dargestellt, sind Grounding-Techniken essenziell für die Generierung von realisierbaren Lösungen für die robotische Montage. Multimodale Grounding-Ansätze, wie beispielsweise Grounded Decoding [172], haben ihre Effektivität bei der Reduktion von Halluzinationen in Foundation Models durch komplementäre Verifikationsquellen bewiesen. Mit der Grounding-Komponente wird dies durch die Integration einer visuellen Beschreibung des aktuellen Montagezustandes unterstützt, da sie der Planungskomponente eine

alternative Grounding-Perspektive ermöglicht. Dies kann solche räumlichen Beziehungen aufdecken, die übersehen wurden, oder Einschränkungen sichtbar machen, die während der initialen Interpretation der Informationen aus dem Wissensgraphen falsch aufgefasst wurden.

Bei der Ausführung der Grounding-Komponente werden Informationen zu den einzelnen Montageschritten automatisiert aus den LLM-Antworten extrahiert. Pro Montageschritt wird ein individuelles CAD-Rendering generiert, um eine fokussierte Bewertung der im Montageschritt beschriebenen Aktionen zu ermöglichen. Dies ermöglicht dem VLM der Grounding-Komponente, sich ausschließlich auf die Realisierbarkeit und die räumlichen Beziehungen in diesem Schritt zu konzentrieren.

Zur Auswertung der generierten Renderings wird das VLM im aktuellen Ansatz ausschließlich als beschreibende Komponente eingesetzt. Es generiert detaillierte visuelle Beschreibungen der Montageschritte, ohne eine Bewertung der Realisierbarkeit vorzunehmen. Die erzeugten Beschreibungen werden anschließend von der Planungskomponente interpretiert, welche die visuellen Informationen analysieren muss, um potenzielle Inkonsistenzen oder Fehler zu identifizieren. Durch dieses Vorgehen wird eine strikte Trennung zwischen Beschreibungs- und Interpretationsaufgaben eingehalten, was die Komplexität des VLMs reduziert, jedoch einen zusätzlichen Verarbeitungsschritt in der Planungskomponente erfordert. Eine Alternative stellt die direkte Interpretation der Informationen im VLM dar. Dies erhöht jedoch die Aufgabenkomplexität für das VLM und steigert das Risiko von Halluzinationen. Erste Versuche haben gezeigt, dass die zum Zeitpunkt der Arbeit verfügbaren VLMs diese gesteigerten Anforderungen bezüglich der Komplexität nicht erfüllen konnten, sodass dieser Ansatz nicht weiterverfolgt wurde.

Technische Umsetzung der Grounding-Komponente

Aus denselben Gründen wie bei der Planungskomponente erfolgt auch bei der Grounding-Komponente kein Finetuning, sondern der Einsatz von vortrainierten Foundation Models sowie die Verbesserung der Ausgabe durch Prompting-Techniken.

Eine der entscheidenden Funktionen für die korrekte Ausführung der Grounding-Komponente stellt die Generierung der CAD-Renderings der einzelnen Montagezustände dar. Hierzu wird die OCC-Schnittstelle verwendet und analog zum Vorgehen bei der Erreichbarkeitsanalyse in der Validierungskomponente werden die montierten Bauteile in einer $TopoDS_{Shape}$ fusioniert. Pro Renderingdurchlauf werden mehrere Kameraperspektiven generiert, um dem VLM eine maximale Informationsmenge zu liefern. Im Fokus der Renderings ist dabei immer die Kontaktfläche zwischen dem neu montierten Bauteil und der bis dahin montierten Baugruppe. Für jeden Montageschritt erstellt das System Renderings aus zwei

(a) Isometrische Ansicht der Positionierung der Federung des Flugzeugfahrwerks.

(b) Detailansicht der Befestigung eines Reifens auf der Achse.

Abbildung 5.10.: Beispiel CAD-Renderings einer Flugzeugfahrwerksmontage. Rosafarben eingefärbt sind zuvor positionierte Bauteile, während gelb eingefärbte Bauteile mit cyanfarbenen Konturen aktuell gefügt werden. Diese beiden Ansichten verdeutlichen beispielhaft die visuellen Informationen, welche das VLM als Eingabe erhält.

verschiedenen Kameraperspektiven: eine isometrische Ansicht und eine Nahaufnahme. Für eine einfachere Interpretierung durch das VLM werden die Renderings anschließend aufbereitet. Hierzu werden Hauptkomponenten in den Farben Braun, Gelb und Rosa eingefärbt und die Konturen mit dickeren Linien in Schwarz, Cyan und Grün hervorgehoben, sodass eine klare Kantendarstellung vor einem hellgrauen Hintergrund gewährleistet ist. Um dem VLM über die Bilder zu vermitteln, welche der Bauteile für den aktuellen Montageschritt am relevantesten sind, werden Bauteile, die keinen Bezug zu dem aktuellen Montageschritt haben, in Braun mit schwarzen Konturen gerendert. Relevante, aber bereits in vorherigen Montageschritten vorkommende Bauteile werden in Rosa mit grünen Konturen gerendert. Das Bauteil des aktuellen Montageschrittes wird in Gelb mit Konturen in Cyan dargestellt. Das Ziel der Verwendung unterschiedlicher Farben besteht darin, einen maximalen Kontrast zwischen Bauteilen zu erreichen und so die Erkennung zu erleichtern. Ein Beispiel für generierte Renderings ist in Abbildung 5.10 dargestellt. Erste Evaluationen haben gezeigt, dass aktuelle VLMs mit zwei Renderings die besten Ergebnisse liefern. Weitere Kameraperspektiven bieten zwar das Potenzial der Steigerung der Informationsmenge, haben aber in den Tests gezeigt, dass sie die aktuellen VLMs überfordern und zu schlechteren Ergebnissen führen.

Nach erfolgreicher Generierung der Renderings erfolgt deren Verarbeitung durch das VLM. Dieses identifiziert zunächst die in den Renderings vorhandenen Farben und bestimmt dann, welches Objekt cyanfarbene Kanten besitzt. Der Fokus auf die cyanfarbenen Konturen statt auf die gelben Flächen des Objektes ist in ersten Tests begründet, die zeigen, dass diese eine höhere Erkennungsrate, ins-

besondere für Kleinteile wie Schrauben, aufweisen. Der verwendete Prompt beauftragt das VLM anschließend, das gefundene Objekt genauer zu beschreiben, indem eine Beschreibung der Form und Pose des Objektes angefragt wird. Hierbei wird explizit auf eine Teileidentifikation durch das VLM verzichtet, da dies besonders bei spezielleren Bauteilen zu fehlerhaften Ausgaben führt. Anschließend wird die Methodik auf die rosafarbenen Objekte angewandt, um ein Gesamtverständnis für die für den aktuellen Montageschritt relevanten Objekte zu erzeugen.

Die erkannten Formen und Posen werden anschließend in eine Gesamtantwort zusammengeführt und zur Interpretation an das LLM übergeben.

5.3.3. Manipulationssequenz

Ausgehend von einer gefundenen Montagesequenz wird durch das System anschließend eine zugehörige Manipulationssequenz generiert. Diese besteht im Vergleich zum vorherigen Planungsschritt aus nur zwei statt drei Komponenten: der Planungskomponente und der Validierungskomponente. Beide sind in ihrer grundlegenden Idee ähnlich zu den Komponenten aus der Montagesequenzplanung, wurden jedoch für die Manipulationsplanung adaptiert. Der Wegfall der noch im vorherigen Schritt eingebundenen Grounding-Komponente hat mehrere Gründe. Einer der Hauptgründe ist die Art der auftretenden Fehler. Erste Evaluationen haben gezeigt, dass die überwiegende Fehlerquelle bei der Generierung der Manipulationssequenz in Inkonsistenzen in der Formatierung liegt. Dieser Aspekt kann bereits durch die Validierungskomponente adressiert werden. Ein weiterer Punkt ist, dass bereits bei der Generierung der Montagesequenz ein Grounding stattgefunden hat und dies implizit durch die Montagesequenz an diesen Schritt weitergegeben wird. Die Montagesequenz integriert bereits ein räumliches und geometrisches Verständnis, welches sowohl aus der Wissensgraphen-Analyse als auch – sofern aktiviert – aus der visuellen Verifikation durch die Grounding-Komponente der vorherigen Stufe stammt. Dieser geerbte Kontext bietet ein ausreichendes Grounding-Fundament für die Generierung angemessener Manipulationssequenzen, ohne dass weitere visuelle Verifikationsschritte notwendig sind. Zusätzlich war auch die Optimierung der Recheneffizienz ein Grund. Wie zuvor erläutert, bieten VLMs zwar ein hilfreiches zusätzliches Grounding für die Planungskomponente, ihre aufwendigen Inferenzen erzeugen jedoch einen substanziellen Rechenaufwand (computational overhead).

Planungskomponente

Wie in der vorherigen Planungskomponente, ist auch in der Komponente für die Manipulationsplanung ein LLM integriert. Hierbei ist es jedoch nicht zwingend

erforderlich, dass es sich bei dem gewählten LLM um das gleiche wie zuvor handelt. Durch die große Nähe zwischen Sequenzplanungs- und Manipulationsplanungsproblem in Bezug auf ihre Anforderungen an die Schlussfolgerungsfähigkeiten der verwendeten LLMs ist prinzipiell zu erwarten, dass ein LLM, welches in der Sequenzplanung gute Ergebnisse erzeugen kann, auch geeignet für die Manipulationsplanung ist. Weitere Gründe, die gegen ein zusätzliches LLM sprechen, sind die geringere Recheneffizienz durch zusätzliche Speicheranforderungen und längere Ausführungszeit durch das Laden eines zweiten Modells. Eine wesentliche Designentscheidung bei der Planungskomponente ist, dass sie die zu verarbeitende Montagesequenz nicht weiter anpassen kann. Dies gilt auch, sollte festgestellt werden, dass keine mögliche Manipulationssequenz hierzu erzeugt werden kann. Das Ziel dieser Planungskomponente besteht darin, entsprechend einer gegebenen Montagesequenz eine mögliche Manipulationssequenz zu generieren und keine zusätzliche Validierungskomponente für den vorherigen Schritt darzustellen. Dementsprechend wird für diese Komponente die Annahme getroffen, dass sie ausschließlich mit korrekten Montagesequenzen ausgeführt wird. Dies ermöglicht bei der Evaluation eine Analyse der Qualität der Schlussfolgerungen über mehrere Bereiche hinweg.

Damit das verwendete LLM in die Lage versetzt wird, eine gültige Manipulationssequenz zu generieren, greift das System nicht nur auf den Wissensgraphen und eine bereitgestellte Prompt-Vorlage zu, sondern auch auf die im vorherigen Schritt generierte Montagesequenz und eine Manipulationsliste. Beide neuen Eingabequellen werden dazu in einer adaptierten Prompt-Vorlage integriert, welche als Eingabe für das LLM dient und einen umfassenden Kontext für die Generierung der Manipulationsliste bereitstellt. Dieses Vorgehen vereinfacht auf der Implementierungsebene die Übergabe der Informationen an das LLM, da hier auf die gleichen Techniken wie bei der Sequenzplanung zurückgegriffen werden kann.

Als Ausgabe wird durch das LLM eine Manipulationssequenz in natürlicher Sprache und nicht als Robotercode generiert. Dabei repräsentiert die Sequenz einen symbolischen Plan, enthält somit für spezifische Montageschritte eine Zuweisung von Manipulationen, die sich nach den in der Manipulationsliste aufgezählten richten. Falls es sich um ein Multi-Roboter-Setup handelt, erzeugt es auch eine konkrete Zuweisung von Manipulationen zu den jeweiligen Robotern. Die Syntax der Ausgabe passt sich bei der Beschreibung innerhalb der Manipulationsliste an; eine Pick-and-Place-Fähigkeit kann also sowohl sehr vereinfacht als „`Bewege Objekt X von Position Y zu Position Z`“ dargestellt werden, aber auch feiner aufgegliedert als „`Öffne Greifer um Wert x`“, „`Bewege Greifer zu Position Y`“, „`Schließe Greifer um Wert z`“ etc. Durch dieses Vorgehen ist es möglich, die Komplexität des finalen symbolischen Montageplans in einem gewissen Rahmen zu adaptieren. Abbildung 5.11 zeigt beispielhaft für zwei Robotersysteme, welche Manipulationsfähigkeiten diese anbieten könnten.

[1] Bildquelle Schweißroboter: www.evsint.com (Zugegriffen am 16.02.2026)

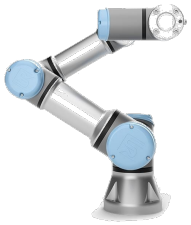
Schweißroboter	**Serviceroboter mit flexiblem Arbeitswerkzeug**
	
Mögliche Manipulationen: 1. Zur Zielposition <Objekt_1> (Knoten_xy) bewegen 2. Zielposition auf <Objekt_1> (Knoten_xy) setzen 3. Schweißvorgang zwischen <Objekt_1> (Knoten_xy) und <Objekt_2> (Knoten_yz) ausführen	**Mögliche Manipulationen:** 1. Zur Zielposition <Objekt_1> (Knoten_xy) bewegen 2. Zielposition auf <Objekt_1> (Knoten_xy) setzen 3. Arbeitswerkzeug von <vorheriges_Werkzeug> zu <nächstes_Werkzeug> wechseln 4. <Objekt_1> (Knoten_xy) greifen 5. < Objekt_1> (Knoten_xy) an Zielposition (Knoten_yz) platzieren 6. < Objekt_1> (Knoten_xy) auf Arbeitsfläche positionieren 7. < Objekt_1> (Knoten_xy) in <Objekt_2> (Knoten_yz) schrauben 8. < Objekt_1> (Knoten_xy) in < Objekt_2> (Knoten_yz) einsetzen 9. <Objekt_1> (Knoten_xy) in <Objekt_2> (Knoten_yz) einpressen

Abbildung 5.11.: Beispielhafte Auflistung der Manipulationsfähigkeiten zweier Robotersysteme. Aufgrund des fest installierten Werkzeugs ist das Aktionsspektrum des Schweißroboters im Vergleich zum zweiten System eingeschränkt.[1]

Die Entscheidung durch das System, nicht den direkten Robotercode, sondern einen symbolischen Plan zu generieren, der für Menschen lesbar ist, lässt sich durch folgende Aspekte begründen:

- Das für Menschen interpretierbare Format macht es einfacher, die Ausgabe zu prüfen und falls notwendig, zu korrigieren.
- Das angedachte Framework kann unabhängig von der Roboterzelle verwendet werden. Das ist vor allem während der aktiven Entwicklungsphase der Roboterzelle relevant, da sich zu diesem Zeitpunkt die grundlegenden Robotersysteme und meist auch deren Robotercodeformate ändern.
- Das Framework zielt auf die Fragestellung ab, ob Foundation Models in der Lage sind, ein Montageplanungsproblem zu lösen. Für die Beantwortung dieser Frage ist ein symbolischer Plan eine ausreichende Antwort.
- Es existieren bereits Arbeiten, welche aus einem symbolischen Plan ausführbaren Robotercode generieren [173].

Beim Vergleich von Montagesequenzplanung und Manipulationssequenzplanung kommt es bezüglich der Komplexität zu einer Verschiebung. Bei der Montagesequenzplanung liegt die Komplexität besonders in der großen Menge an Informationen und der Notwendigkeit, diese zu interpretieren. So muss das LLM in

diesem Fall beispielsweise die komplette Struktur des Wissensgraphen analysieren. Bei der Manipulationssequenzplanung wird die Komplexität der Informationsmenge pro Montageschritt deutlich reduziert. Bereits in der Eingabe ist durch die Montagesequenz bekannt, welche Bauteile betrachtet werden müssen, sodass sich das System nur auf die dafür relevanten Informationen aus dem Wissensgraphen fokussieren kann. Da ein Montageschritt meistens nicht aus nur einer Manipulation besteht, müssen bei der Manipulationsplanung pro Schritt eine größere Anzahl an korrekt aufeinanderfolgenden Manipulationen betrachtet werden. Das bedeutet auch eine höhere Komplexität der Formatierungsanforderungen für die Ausgabe. Besonders im Hinblick auf die Validierung des symbolischen Plans und einer möglichen nachgelagerten automatisierten Ausführung sind die Formatierungsanforderungen zentral.

Technische Umsetzung der Planungskomponente

Die Planungskomponente der Manipulationssequenzgenerierung unterscheidet sich im Hinblick auf die technische Umsetzung nur geringfügig von der Planungskomponente der Sequenzplanung. Aus Gründen der Wiederverwendbarkeit sowie der Reduktion des Bedarfs an zusätzlichen Rechenressourcen wird das gleiche LLM eingesetzt wie zuvor. Auch das Inferenzkonzept wird in der Form beibehalten, dass in einer ersten Ausführung die Eingangsdaten analysiert werden, bevor in einem zweiten Schritt die Sequenz in ein korrektes Format transformiert wird.

Unterschiede zwischen den beiden Planungskomponenten bestehen primär in den eingebundenen Informationsquellen. Hier erhält die Planungskomponente der Manipulationssequenz nicht nur den Wissensgraphen und eine entsprechende Prompt-Vorlage, sondern auch die zuvor generierte Montagesequenz und die Manipulationsliste. Durch den Aufbau der Manipulationsliste im JSON-Format gestaltet sich dies analog zur Einbindung des Wissensgraphen.

Über die Manipulationsliste wird zusätzlich spezifiziert, ob die vorliegende Montagesequenz auf eine Roboterzelle mit nur einem oder mehreren Robotern und Werkzeugen angewandt wird. Hierfür besteht die Manipulationsliste aus Unterlisten, je eine Unterliste pro Robotertyp. Zusätzlich können in der Prompt-Vorlage Informationen über die verwendeten Roboter hinterlegt sein, z. B. Traglast, Reichweite etc. Sind diese Daten nicht enthalten, so nimmt das System gleichartige Roboter an und geht davon aus, dass die Roboter alle Bauteile in Bezug auf Erreichbarkeit und Traglast handhaben können.

Eine weitere Unterscheidung zwischen Planung der Montagesequenz und Manipulationssequenz stellt die Abkehr vom schrittweisen Vorgehen dar. Anders als zuvor ist eine voneinander unabhängige Betrachtung der einzelnen Montageschritte nicht möglich, da spätere Montageschritte Einfluss darauf haben können,

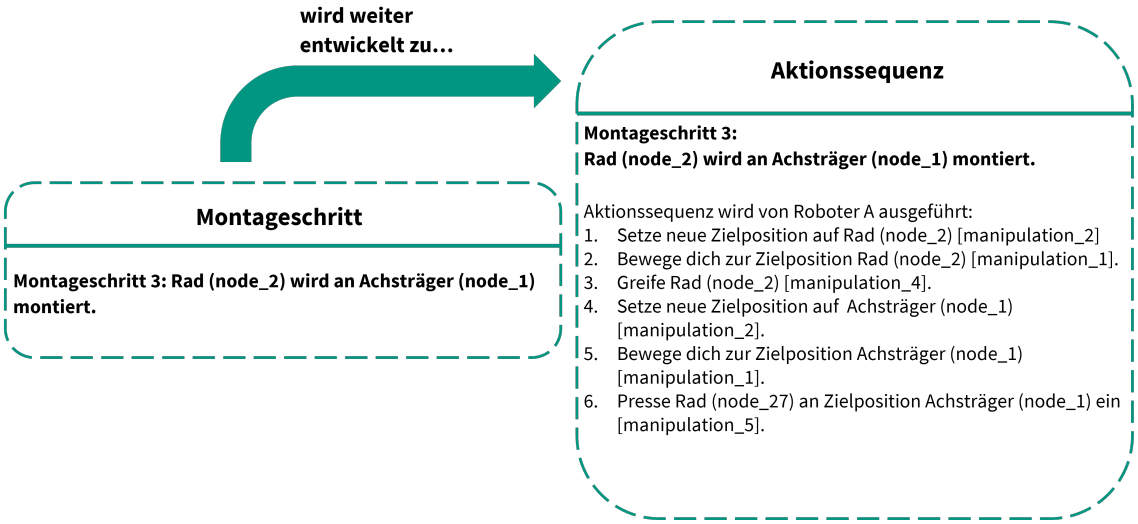


Abbildung 5.12.: Veranschaulichung der Transformation eines Montageschrittes in eine Manipulationssequenz, am Beispiel der Radmontage auf einer Achse aus Abbildung 5.10b.

wie ein Bauteil in einem vorherigen Montageschritt zu handhaben ist. Entsprechend wird eine Gesamtbetrachtung aller Montagesequenzen und der dafür notwendigen Montageschritte durchgeführt.

Im Formatierungsschritt wird durch die Prompt-Vorlage eine strikte Strukturierung der Manipulationssequenz und der darin enthaltenen notwendigen Manipulationsfolgen für jeden Fügeschritt erzwungen. Jede Manipulationsfolge ist nach der folgenden Struktur aufgebaut: Identifikation des konkreten Fügeschritts, Zuweisung zu einem konkreten Roboter, gefolgt von den konkreten Manipulationsaufgaben für jeden Roboter. Der Zuweisungsblock wiederholt sich anschließend für jeden benötigten Roboter. Ein Beispiel für eine entsprechend strukturierte Manipulationssequenz ist in Abbildung 5.12 dargestellt.

Verglichen mit der Montagesequenz stellt die Manipulationssequenz eine deutlich größere Herausforderung in Bezug auf die verfügbare Tokenlänge der Ausgabe dar. Bei Baugruppen mit vielen Bauteilen bzw. komplexen Fügeoperationen besteht eine Chance, dass die generierte Sequenz die mögliche Tokenlänge überschreitet. Dieser Fall kann automatisch detektiert werden und wird durch einen Zusatzmechanismus abgefangen, welcher die bis zur Tokengrenze erzeugte Ausgabe zwischenspeichert und dem LLM als zusätzliche Eingabe zur Verfügung stellt. So wird das LLM befähigt, die Generierung an der vorherigen Abbruchstelle fortzusetzen.

Validierungskomponente

Die Validierungskomponente ist in ihrer grundlegenden Funktion und Struktur deckungsgleich zu der Validierungskomponente für die Montagesequenzgene-

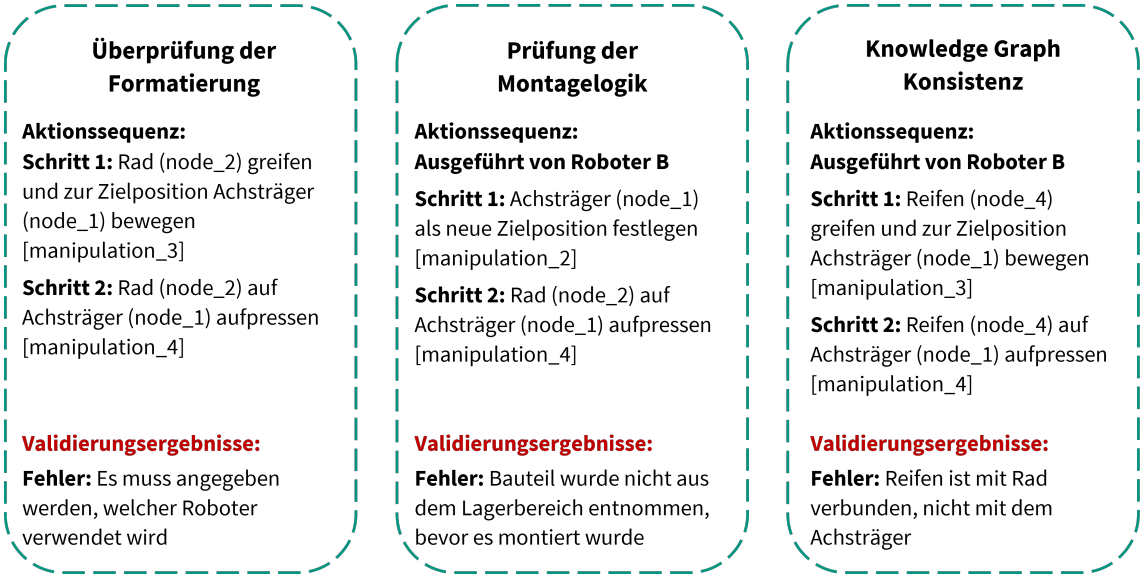


Abbildung 5.13.: Beispiele für Validierungsfehler bei der Manipulationssequenz eines Radmontageschritts, die die drei Validierungskategorien veranschaulichen: Bei der formalen Korrektheitsprüfung wird die Formatkonformität überprüft, bei der Prüfung der Montagelogik wird die operative Durchführbarkeit überprüft und bei der Konsistenzprüfung des Wissensgraphen werden die räumlichen und montagetechnischen Beziehungen validiert.

rierung. Sie weist jedoch einige wesentliche Unterschiede in Bezug auf ihre Anforderungen auf. Zu diesen zählen die im Vergleich zur Montagesequenz erhöhte Länge der finalen Ausgabe durch die Unterteilung der einzelnen Montageschritte, die Integration von Beschränkungen durch die Manipulationsliste sowie der Wegfall der nachgelagerten Grounding-Komponente.

Wie bei der Validierung der Montagesequenz, werden vom Validierungsalgorithmus erneut die drei Kategorien betrachtet: Verifikation der formalen Korrektheit, Bewertung der logischen Korrektheit und Prüfung der Konsistenz des Wissensgraphen. Hierbei verschiebt sich allerdings der Fokus auf die Kategorie der Bewertung der formalen Korrektheit. Dies ist in der größeren Komplexität der einzelnen Montageschritte begründet, da sie nun mehrere Manipulationsschritte enthalten können, was die Länge der Ausgabe im Vergleich zur zugrundeliegenden Montagesequenz substanziell erhöht. Die erhöhte Länge der Ausgabe korreliert direkt mit einem gesteigerten Potenzial für Formatierungsfehler, Syntaxabweichungen und strukturellen Inkonsistenzen. Das hat Einfluss auf eine nachgelagerte Verarbeitung. Zeitgleich sinkt die Relevanz der Konsistenzvalidierung des Wissensgraphen, da nicht die vollständige Graphenstruktur, sondern nur Teilbereiche des Graphen für individuelle Montageschritte relevant sind. Dies hat eine reduzierte Wahrscheinlichkeit von graphenbezogenen Fehlinterpretationen und Inkonsistenzen zur Folge. Abbildung 5.13 skizziert exemplarische Prüfungen für alle drei Kategorien.

Die Verifikation der formalen Korrektheit ist vom Ansatz her grundsätzlich deckungsgleich zu jener bei der Validierung der Montagesequenz. Durch die geänderte Aufgabenstellung sind jedoch einige Aspekte angepasst. Unter anderem beinhaltet die Verifikation die Verifizierung, dass zu jedem Montageschritt eine zugehörige Manipulationssequenz generiert wurde, sowie die Prüfung, dass nur solche Manipulationen gewählt sind, die auch in der Manipulationsliste vorkommen. Daraus ergibt sich, dass ein zusätzlicher Input die Manipulationsliste als Informationsquelle berücksichtigen und auswerten muss.

Für die Bewertung der logischen Korrektheit untersucht der Algorithmus die semantische Kohärenz von Manipulationssequenzen und deren Einhaltung von fundamentalen operativen Prinzipien. Hierzu zählen:

- Prüfung, dass Fügeoperationen nur an Positionen an der Baugruppe ausgeführt werden und beispielsweise keine Schraubaktionen im leeren Raum durchgeführt werden.
- Prüfung auf eine schlüssige Manipulationssequenz pro Montageschritt; beispielsweise müssen korrekte Werkzeugwechsel durchgeführt werden, sollte ein Roboter mehrere Werkzeuge verwenden können.
- Prüfung auf die korrekte Roboterauswahl bei einem Multi-Roboter-Setup.

Die Validierung der Konsistenz des Wissensgraphen übernimmt die gleiche Rolle wie im Schritt zuvor. Sie dient als Prüfung zwischen den vorgeschlagenen Manipulationssequenzen und den im Wissensgraphen kodierten strukturellen Informationen. Exemplarische Prüfungen umfassen die Verifizierung, dass die korrekten Montagemanipulationen für einen bestimmten Bauteiltyp verwendet werden, und die Bestätigung, dass Manipulationssequenzen die zwischen Komponenten definierten räumlichen Beziehungen und Einschränkungen berücksichtigen. Bei der technischen Umsetzung der Validierungskomponente können zu einem Großteil die Ansätze der Validierungskomponente für die Montagesequenzgenerierung genutzt werden. Den einzigen signifikanten Unterschied stellt der Wechsel von einem Bottom-up-Analyseansatz auf einen Top-down-Ansatz dar.

5.3.4. Umsetzung des Frameworks

Die Umsetzung des Frameworks erfolgt als eine Verarbeitungspipeline und ist für die Orchestrierung der sequenziellen Ablaufsteuerung verantwortlich. Ihre Hauptaufgaben gliedern sich in die korrekte Ausführung der Framework-Komponenten sowie das Informationsmanagement, welches sicherstellt, dass jede Komponente Zugriff auf die anwendungsfall- und phasenspezifischen Daten hat. Aufgrund der erheblichen Modellgröße und der damit verbundenen Rechenanforderungen werden die Foundation Models nicht lokal, sondern remote auf einem GPU-Cluster ausgeführt. Dies erfordert eine Infrastruktur für den Datentransfer via SSH und die Verwaltung von einzelnen Inferenzausführungen. Zusätzlich obliegt der Pipeline die Überwachung von Abbruchkriterien sowie

die Erfassung von Zeit- und Token-Performance, um neben der Lösungsfindung auch quantitative Leistungsindikatoren bereitzustellen. Der Aufbau und Datenfluss des entwickelten Frameworks ist in Abbildung 5.14 dargestellt.

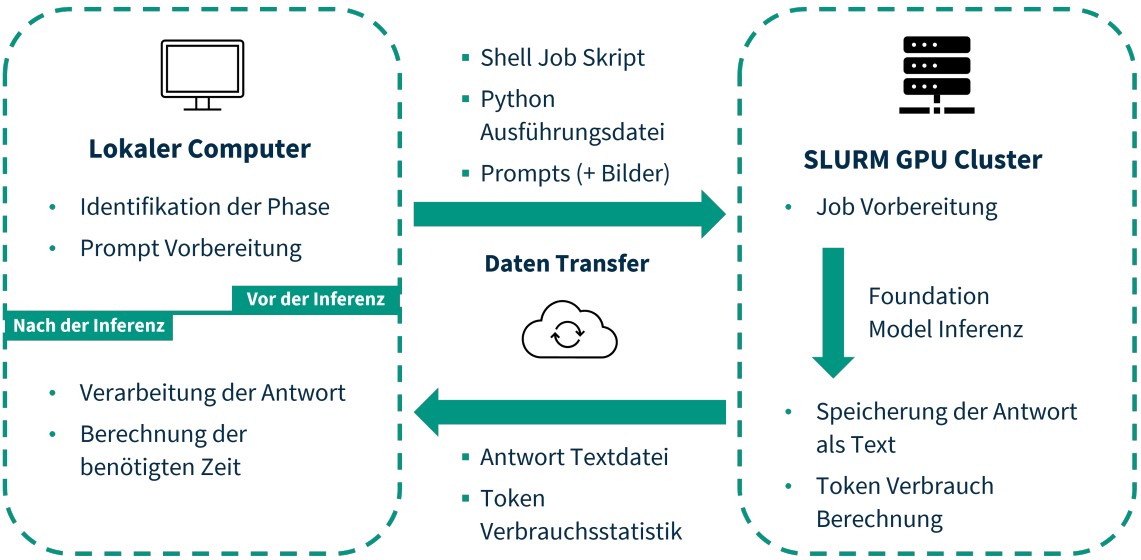


Abbildung 5.14.: Schematische Darstellung des implementierten Frameworks und des Datenflusses. Nach der lokalen Dateivorbereitung erfolgt der Transfer mittels SSH. Auf dem SLURM-Cluster steuert ein Shell-Job-Skript die Ressourcenzuweisung (GPUs) und das Laden der Modelle, woraufhin das Python-Skript die LLM-Inferenz ausführt. Abschließend werden die generierten Antworten und der ermittelte Token-Verbrauch zurück an den lokalen Computer übermittelt.

Informationsmanagement und Komponentenselektion

Die Steuerung der Pipeline erfolgt über eine Konfigurationsdatei. Diese definiert mittels Flags und Konfigurationsparametern den Programmfluss, wodurch eine flexible Anpassung der beteiligten Komponenten und der verschiedenen Phasen ermöglicht wird. Die Aktivierung spezifischer Komponenten wird durch Laufzeit-Flags kontrolliert, die festlegen, welche Feedback-Mechanismen (z. B. Validierung oder Grounding) während der Generierung von Montage- oder Manipulationssequenzen instanziiert werden.

Der Fortschritt innerhalb der Phasen wird durch verschachtelte Schleifen gesteuert. Das ermöglicht die Umsetzung flexibler Terminierungsstrategien, die von festen Iterationszahlen bis hin zu erfolgsbasierten Ansätzen reichen. Die Pipeline nutzt dabei eine bedingte Logik, um die korrekte Sequenzierung zu erzwingen: Erfolgreiche Validierungsergebnisse triggern automatisch den Übergang zu nachfolgenden Komponenten, während Validierungsfehler das Feedback unmit-

telbar zurück an die Planungskomponente leiten und verbleibende Grounding-Stufen überspringen, um kaskadierende Fehler zu vermeiden.

Der Informationsfluss zwischen den Komponenten wird durch strukturierte Datenintegration realisiert. Ausgaben der Validierung und Feedback des Groundings werden automatisch mit vorherigen Modellantworten und anwendungsfallspezifischen Eingaben fusioniert, bevor sie in kohärente Prompts für den nächsten Iterationszyklus formatiert werden. Hierbei unterscheidet die Verarbeitungspipeline strikt zwischen temporärer Datenspeicherung während der Laufzeit und permanenter Datenspeicherung für die nachgelagerte Auswertung. Permanente Daten, wie strukturierte Informationen oder Modellantworten, werden im JSON- oder Textformat gespeichert. Temporär verwaltet werden hingegen exekutionsspezifische Daten, die nach der Verarbeitung obsolet werden (z. B. veraltete Validierungsergebnisse) oder die formatspezifische Informationen enthalten, die nicht verlustfrei in Text konvertiert werden können. Ein Beispiel hierfür sind die geometrischen Repräsentationen von Knoten im $TopoDS_{Shape}$-Format. Diese existieren als Laufzeitvariablen und werden nach Abschluss der Ausführung freigegeben, um die Speichernutzung zu optimieren.

Zur Steigerung der Recheneffizienz implementiert das System einen Filtermechanismus für validierte Sequenzteile. Erfolgreich validierte Montage- und Manipulationssequenzen von vollständigen Unterbaugruppen werden temporär von nachfolgenden Validierungszyklen ausgeschlossen. Diese Unterbaugruppen bleiben zwar für das Grounding zugänglich, werden jedoch in reinen Validierungsrunden umgangen, um redundante Verarbeitungsschritte zu reduzieren. Sobald jedoch eine Grounding-Iteration durchgeführt wird, werden alle Unterbaugruppen erneut für die Validierung freigegeben, da das visuelle Feedback Modifikationen an jedem Schritt der Sequenz hervorrufen kann.

Eng verknüpft mit der Effizienzsteigerung sind die implementierten Terminierungskriterien, die sowohl erfolgsgetriebene als auch Schwellenwert-Iterationsstrategien abdecken. Drei Hauptkriterien wurden integriert:

1. **Maximale Iterationen:** Konfigurierbare Zähler für jeden Iterationstyp (Grounding oder Validierung) begrenzen die Laufzeit.
2. **Erfolgreiche Validierung:** Die Filterung aller Unterbaugruppen erlaubt eine Terminierung, sobald das gesamte Produkt fehlerfrei validiert wurde.
3. **Stagnation der Lösung:** Ein zusätzliches Kriterium beendet den Prozess, wenn die Planungskomponente trotz Fehlermeldung identische Sequenzen zurückgibt. Endlosschleifen werden damit verhindert.

Zusätzlich integriert das Framework umfassende Tracking-Funktionen. Das Zeit-Monitoring erfasst Zeitstempel zu Beginn und Ende jeder Komponentenausführung, inklusive der Wissensgraph-Erstellung und Datentransfers. Dies liefert Benchmarking-Daten für den Vergleich mit manuellen Planungsmethoden. Das Token-Tracking ermöglicht zudem eine vergleichende Analyse des zusätzlichen Rechenaufwands verschiedener Foundation Models.

5.4. Zusammenfassung und Fazit Sequenzplanung mithilfe von generativer KI

Dieses Kapitel widmet sich der komplexen Herausforderung, physikalisch valide Montagesequenzen automatisiert zu generieren, wobei der Mangel an öffentlich zugänglichen Trainingsdaten im industriellen Kontext ein zentrales Hindernis darstellt. Zur Lösung dieses Problems wird zunächst ein genetischer Algorithmus entwickelt, der basierend auf CAD-Daten synthetische, aber valide Montagesequenzen erzeugt. Ein besonderer Fokus liegt hierbei auf einem Dekonstruktionsansatz, der Montageprozesse invers betrachtet, um von Beginn an gültige Lösungen für das Training zu garantieren.

Auf dieser Datengrundlage wird ein Voxel-basiertes Sequenz-zu-Sequenz-Netzwerk trainiert, das geometrische Ähnlichkeiten nutzt, um Montageprinzipien auf neue Baugruppen zu übertragen. Des Weiteren wird ein neuartiges Framework vorgestellt, das die semantischen Fähigkeiten von LLMs mit der strukturierten Logik von Wissensgraphen verknüpft. Der Wissensgraph repräsentiert hierbei die Baugruppenstruktur und räumliche Relationen, die das LLM zur Planung nutzt. Der Ansatz generiert zunächst eine abstrakte Montagesequenz und reichert diese im zweiten Schritt um die notwendigen Manipulationsaktionen an, wobei spezifische Parameter der eingesetzten Robotersysteme berücksichtigt werden. Da generative Sprachmodelle zu Halluzinationen neigen, integriert das System einen robusten, dualen Verifikationsmechanismus. Dieser besteht aus einer physikalischen Validierung durch Prüfalgorithmen und einer visuellen Überprüfung (Grounding) mittels VLMs, die generierte Renderings der Montageschritte bewerten. Dieser Ansatz stellt sicher, dass die vom Sprachmodell vorgeschlagenen Pläne nicht nur logisch klingen, sondern in einer realen Roboterzelle tatsächlich kollisionsfrei ausführbar sind.

6. Montagesequenzverfeinerung durch den Einsatz von Montageaffordanzen

Allein auf Basis einer reinen Teilesequenz, die keine weitergehenden Informationen über den konkreten Prozessschritt beinhaltet, ist es einem automatisierten System nicht ohne Weiteres möglich, eine Montage auszuführen. Dies liegt daran, dass die erforderliche Manipulationsaktion nicht in jedem Arbeitsschritt identisch ist. Dementsprechend muss eine Montagesequenz, um in einen nutzbaren Montageplan überführt zu werden, um konkrete Manipulationen pro Arbeitsschritt erweitert werden. Dies kann auf unterschiedliche Weise erfolgen, wobei von der manuellen Informationsergänzung durch den Menschen über interaktive Systeme bis hin zu vollautomatisierten Ansätzen verschiedene Lösungen realisierbar sind.

Im folgenden Kapitel wird der in dieser Arbeit entwickelte und evaluierte Ansatz zur automatisierten Bestimmung von Fügeoperationen auf Basis von Montageaffordanzen beschrieben. Dieser Ansatz zielt insbesondere darauf ab, die notwendige menschliche Interaktion so weit wie möglich zu reduzieren oder gänzlich zu vermeiden, um den Einsatz in autonomen und vollautomatisierten Systemen zu ermöglichen.

6.1. Affordanzen im Montagekontext

Eine große Herausforderung bei der Bestimmung einer Montageaktion für einen konkreten Arbeitsschritt stellt die Vielseitigkeit der Bauteile dar. So ist es möglich, dass eine Schraube zunächst durch ein oder mehrere Bauteile gesteckt wird, bevor sie final in eine Gewindebohrung eingeschraubt wird. Dies hat zur Folge, dass pro Bauteiltyp keine feste Montageaktion definiert werden kann. Welche Aktion in jedem Arbeitsschritt erforderlich ist, ergibt sich primär aus der Geometrie der beteiligten Bauteile, insbesondere in den Bereichen der Kontaktstellen. Für die Bestimmung der möglichen Montageaktion, die einem bestimmten Bereich der Bauteilgeometrie zugeordnet werden kann, wird das Konzept der Affordanzen genutzt.

Im Rahmen dieser Arbeit wird das Prinzip der Affordanzen aufgegriffen und daraus das Konzept der *Montageaffordanzen* abgeleitet. Im Vergleich zum allgemeinen Affordanz-Konzept sind Montageaffordanzen auf die Situation der zerstörungsfreien Montage und Demontage eingeschränkt. Hierdurch wird der Suchraum, aus dem ein System eine mögliche Aktion ableiten muss, signifikant reduziert. Des Weiteren werden Montageaffordanzen so definiert, dass ein geometrischer Bereich eines Objektes jeweils nur einen konkreten Affordanztyp anbieten kann. Beide Einschränkungen erleichtern durch die Reduktion von Mehrdeutigkeiten die Umsetzung einer automatisierten Affordanzerkennung. Im Vergleich zu den Konzepten des Forderungs- bzw. Aufforderungscharakters bleiben die Affordanzen eines Objektes jedoch unveränderlich. Dies ermöglicht es, trotz einer initial unklaren Montagesituation eindeutige Affordanzen zu bestimmen.

Die betrachteten Montageaffordanzen beziehen sich primär auf die Montage mechanischer Baugruppen wie Getriebe oder Gehäuse. Grundlage für die Definition bilden die Fügeoperationen der DIN 8593 – „Fertigungsverfahren Fügen“ [163]. Das entwickelte Konzept ist jedoch so modular gestaltet, dass es um weitere Montageaktionen (z. B. für die Elektronikbestückung) erweitert werden kann. Zur Bestimmung der relevanten Fügeoperationen führt die Norm folgende Grundkategorien auf:

- Zusammensetzen – Gruppe 4.1
- Füllen – Gruppe 4.2
- Anpressen, Einpressen – Gruppe 4.3
- Urformen – Gruppe 4.4
- Umformen – Gruppe 4.5
- Schweißen – Gruppe 4.6
- Löten – Gruppe 4.7
- Kleben – Gruppe 4.8

Von besonderer Relevaz für die Arbeit sind die Gruppen *4.1* und *4.3*, welche form- und kraftschlüssige Montagetechniken beschreiben sowie die Gruppe *4.6 – 4.8*, welche stoffschlüssige Fügetechniken aufführen, da sie die gängigsten Montagetechniken umfassen. Basierend auf diesen fünf Gruppen werden im weiteren Verlauf die Montageaktionen, welche in Tabelle 6.1 aufgelistet sind, berücksichtigt. In Tabelle 6.2 werden hierfür die Bezeichnungen der entsprechenden Fügeoperationen definiert.

Aus den verschiedenen Montageoperationen werden in einem nächsten Schritt die entsprechenden Montageaffordanzen abgeleitet. Die meisten der Fügeoperationen erfordern dabei das vorhandensein von zwei Montageaffordanzen, beispielsweise ist für die Fügeoperation Schrauben eine *SCREW_INNER* Affordanz

Gruppe	Untergruppe	Definition
4.1. Zusammen-setzen	4.1.1. Auflegen, Aufsetzen, Schichten	Fügen zusammenpassender Teile unter Nutzung der Schwerkraft, im allgemeinen in Verbindung mit Formschluss.
	4.1.3. Ineinanderschieben	Fügen, bei dem das eine Fügeteil in das andere oder über das andere geschoben wird.
	4.1.6. Federnd Einspreizen	Fügen durch vorheriges elastisches Verformen, damit das Fügeteil nach dem Einlegen oder Aufschieben und anschließendem Rückfedern durch Formschluss gehalten wird.
4.3. Anpressen, Einpressen	4.3.1. Schrauben	Fügen durch Anpressen mittels selbsthemmenden Gewindes.
	4.3.4. Fügen durch Pressverbindung, davon 4.3.4.1. Fügen durch Einpressen, Verstiften	Fügen durch Ineinanderschieben eines Innenteils und eines Außenteils, wobei zwischen beiden ein Übermaß besteht
4.6. Schweißen 4.7. Löten 4.8. Kleben	Stoffschlüssige Verbindungen, werden gleich behandelt, ohne Unterteilung.	Schweißen: Fügen unter Einfluss von Wärme und/oder Druck. An den Fügestellen werden die Werkstücke aufgeschmolzen. Löten: Fügen durch Schmelzen von Lot. Die Fügestellen werden erwärmt, aber nicht geschmolzen. Kleben: Fügen mit Klebstoff

Tabelle 6.1.: Ausgewählte Fügeoperationen nach DIN 8593 - Teile 1, 3, 6-8 [163].

Fügeoperation	Bezeichnung
Federnd Einspreizen	*ELASTIC*
Auflegen	*LAYUP*
Stoffschlüssige Verbindung	*MATERIAL*
Einpressen	*PRESS*
Schrauben	*SCREW*
Ineinanderschieben	*TELESCOPE*

Tabelle 6.2.: Bezeichnungen für die Fügeoperationen.

auf dem einen und eine *SCREW_OUTER* Affordanz auf dem anderen Bauteil notwendig. Eine Definition aller Affordanzen sowie die Zuordnung zu den spezifischen Fügeoperationen ist in Tabelle 6.3 dargestellt. In der Definition ist zu erkennen, dass aus machen Montageaffordanzen mehrere Fügeoperationen abgeleitet werden können. In diesen Fällen muss das spätere System zusätzliche Informationen über die Baugruppe berücksichtigen, z.B. ob zwischen zwei Bauteilen ein Übermaß besteht um eine korrekte Montageaktion zu bestimmen.

Affordanz	**Definition**	**Fügeoperation**
SCREW_OUTER	Außengewinde (z. B. Schraube)	*SCREW*
SCREW_INNER	Innengewinde (z. B. Bohrung)	*SCREW*
PLUG_OUTER	Innenteil (Stift) eines Steck-/Presskontaktes. Glatte Oberfläche, konstanter Querschnitt.	*TELESCOPE, PRESS*
PLUG_INNER	Außenteil (Bohrung) eines Steck-/Presskontaktes. Empfänger für $PLUG\_OUTER$.	*TELESCOPE, PRESS*
PLUG_INTERLOCKING_OUTER	Innenteil eines formschlüssigen Kontaktes. Querschnitt vergrößert sich entlang der Länge.	*ELASTIC*
PLUG_INTERLOCKING_INNER	Passung eines formschlüssigen Kontaktes. Querschnitt vergrößert sich entlang der Tiefe.	*ELASTIC*
LAYUP	Flache oder gekrümmte Oberfläche ohne spezifische oben genannte Merkmale.	*LAYUP, MATERIAL*

Tabelle 6.3.: Definition der Montageaffordanzen.

6.2. Detektion von Montageaffordanzen

Die Bestimmung der Affordanzen eines Bauteils kann auf unterschiedliche Art und Weise erfolgen. Dadurch besteht die Möglichkeit diese Informationen vorab zu definieren und in Form einer Informationsdatenbank dem System zur Verfügung zu stellen. Es gibt jedoch auch die Option, die vorhanden Affordanzen erst zur Laufzeit zu detektieren, dieses kann sowohl eine durch den Menschen händisch hinzugefügte Information sein als auch eine auf Basis von Visiondaten abgeleitete. Um zur Laufzeit das System möglichst autonom agieren lassen zu können, wurde in der Arbeit das automatisierte Ableiten der Affordanzen auf Basis von Visioninformationen weiterverfolgt. Auch möglich wäre die Einbindung einer externen Datenbank. Allerdings hätte diese sehr umfangreich sein müssen, um alle im Alltag vorkommenden Bauteile und Abwandlungen abzudecken. Eine entsprechende Datenbank existierte zum Zeitpunkt der Arbeit noch nicht.

Um aus den Daten eines Bildverarbeitungssystems die entsprechenden Affordanzen abzuleiten wurde ein Ansatz realisiert, welcher zunächst auf den Bilddaten eine Objektlokalisation durchführt und anschließend eine Affordanzerkennung auf den detektierten Bauteilen ausführt. Da beim Einsatz eines solchen Systems eine Vielzahl an verschiedenen Varianten eines Bauteiltyps zu erwarten ist, nutzt das System KI-Verfahren, um eine generalisiertes Verständnis für Affordanzen zu erlenen.

6.2.1. Montageaffordanzen Trainingsdatensatz

Für das Erlernen eines generalisierten Verständnisses von Affordanzen mit überwachten Lernverfahren benötigt das KI System einen ausreichend großen und diversen Trainingsdatensatz. Der Datensatz besteht dabei sowohl aus synthetischen als auch realen Bilddaten. Zur Erstellung dieser synthetischen Trainingsdaten wurde das Programm Blender [174] in Kombination mit STL-Modellen verschiedener Bauteile verwendet. Die Bilder der realen Objekte wurden mit einer Intel RealSense D435 Stereo-Kamera aufgezeichnet und anschließend ebenfalls in Blender annotiert.

In einem ersten Schritt wurde hierzu die zuvor definierten Montagaffordanzen in Segmentierungsklassen übertragen. Diese beinhaltet neben der Montageaffordanz auch eine eindeutige Zuordnung zu einer Texturfarbe, welche sowohl im Datensatz als auch im späteren KI-System genutzt wird, um Bauteilbereichen eine Affordanzklasse zuzuordnen.

Für die Generierung von synthetischen Trainingsdaten wird ein zweistufiges Verfahren verwendet. Zunächst werden die entsprechenden Bauteilmodelle geladen und eine Segmentierungsmaske für jedes Bauteil erzeugt. Hierfür stehen zwei Techniken zur Verfügung: Texture Painting und Vertex Painting.

> **Texture Painting** stellt die aufwendigere Technik dar, da die Segmentierungsmaske in eine seperate Texturdatei auslagert wird. Diese muss anschließend mittels UV-Mapping [175] auf die 3D Geometrie projiziert werden. Durch die Trennung von Textur und Geometrie ist es möglich eine beliebig exakte und detaillierte Affordanzensegmentierung durchzuführen. Das ist vorallem dann vorteilhaft, wenn das Bauteilmodell nur aus einer geringen Anzahl an Polygonen besteht und dadurch Details, z.B. Gewinde, nicht deutlich ausgeprägt sind.
>
> **Vertex Painting** definiert im Gegensatz dazu die zu Segmentierungsmaske direkt an den einzelnen Vertices des 3D-Modells. Hierdurch ist der Detaillierungsgrad der Segmentierung direkt abhängig von der Anzahl der Vertices. Ein späteres Mapping ist bei dieser Technik jedoch nicht notwendig.

Nach der Definition aller im Bauteil vorkommenden Montageaffordanzen durch die Segmentierungsmaske wird in einem zweiten Schritt das Modell gerendert. Um eine möglichst große Menge an Trainingsdaten pro Bauteil zu extrahieren, werden hierzu mehrere virtuelle Kameras erzeugt, welche aus unterschiedlichen Betrachtungswinkeln die Szene und das Bauteil betrachten und rendern. Zudem wird die Bauteilposition variiert, indem das Bauteil nicht an einem fixen Punkt in der Szene erzeugt, sondern sowohl die Position als auch die Orientierung zufällig bestimmt wird. Weitere Parameter mit denen die Szene variiert werden kann, stellen die Hintergrundtextur, die Komposition mit anderen Bauteilen als auch die Beleuchtungs- und Schatteneinstellungen dar.

Für die Erweiterung des Datensatzes hinsichtlich realer Bilddaten werden hierzu zunächst Aufnahmen von ausgewählten Bauteilen mit der RealSense Kamera gemacht. Hierbei werden von Bauteilen mehrere Aufnahmen mit unterschiedlichen Orientierungen und Positionen in der Szene angefertigt. Auf diesen werden anschließend die verschiedenen Affordanzklassen händisch annotiert. Da dieser Prozess sehr zeitaufwendig ist und für jedes einzelne Bauteil pro Bild durchgeführt werden muss, ist nicht möglich einen beliebig großen Datansatz zu annotiren. Zur Generierung eines ausreichend umfangreichen Datensatzes wird im Anschluss Daten Augmentation angewandt. Hierdurch ist es möglich die Informationsmenge, welche aus jeder einzelnen Aufnahme gewonnen werden kann, zu maximieren. Zum Einsatz kamen sowohl Augmentation der Farbwerte, beispielsweise Sättigung und Helligkeit sowie Anwendung von künstlichen Translationen und Rotationen. Dadurch wurde es möglich, aus 175 realen Aufnahmen 2000 Bilder in den Datensatz einzufügen.

Zunächst wurden aus den RGB-D Daten der realen Kamera auch die Tiefenbildinformationen verwendet. Das Ziel war feinere Geometrie wie etwa Gewinde deutlicher zu detektieren. Da die Farbbilder in einer Auflösung von 1920×1090 aufgenommen wurden und die Aufnahmen des Tiefenbildsensors nur eine Auflösung von 480 x 848 aufweisen, fand eine Hochskalierung der Tiefenbilder statt und eine anschließende Ausrichtung zum Farbbild. Die beiden Bilder wurden anschließend kombiniert und als RGB-D Bild abgespeichert. Aufgrund der ungenügenden Qualität der Tiefenbilddaten der RealSense Kamera wurde dieses Verfahren jedoch im weiteren Verlauf verworfen.

6.2.2. KI-basierte Affordanzdetektion

Zur Erkennung der Bauteile und der Bestimmung der Montageaffordanzen werden zwei neuronale Netzwerke hintereinandergeschaltet. Als Eingabe dient ein Bild der Szene, in welcher die Bauteile zu erkennen sind. Das erste Netzwerk dient der reinen Objekterkennung und generiert entsprechende relevanter Bildbereich (engl. Region of Interest). Diese werden anschließend von einem zweiten Netz verwendet um Montageaffordanzen zu segmentieren. Eine grafische Darstellung der umgesetzten Netzwerkarchitektur und der implementierten Teilnetze kann dem Anhang A entnommen werden.

Das Backbone nutzt in seiner ursprünglichen Variante ein ResNet50 oder ResNet101. Sie extrahieren die Merkmale der Eingangsbilder in fünf Teilschritten, wie in Kapitel 2.2.2 erläutert. In jedem Teilschritt wird die Dimension der Merkmalskarten halbiert und die Anzahl der Merkmalskarten verdoppelt. Hierdurch entstehen aussagekräftige Merkmale für die Klassifizierung. Auch wenn das verwendete Backbone standardmäßig ein ResNet50 verwendet, ist die Anbindung an das restliche Netzwerk so flexibel gestaltet, dass auch andere Ansätze, beispielsweise der DINO Vision Transformer von Meta, verwendet werden können.

Nach der Verarbeitung durch das ResNet-Backbone teilt sich das Netzwerk in zwei Zweige auf: einen für die Objekterkennung und einen für die Affordanz-Erkennung. Der Objekterkennungszweig nutzt einen Feature-Pyramid-Network-Decoder (FPN), um unterschiedliche Objektgrößen zu erkennen. Das FPN verwendet die Auflösungsstufen C2 bis C5 des ResNet-Backbones und besteht aus vier Decoder-Schichten. Abbildung A.1 zeigt dazu die Architektur des FPN-ResNet-Backbones.

In jeder Schicht des FPN-Decoder werden die Merkmalskarten des Backbones zunächst mit einer 1×1 Faltungsschicht auf 256 Merkmalskarten gefiltert. Diese werden anschließend mit einer Hochskalierungsschicht (engl. Upsampling Layer) mit Schrittweise 2 auf die nächstgrößere Auflösungsstufe hochskaliert und elementweise aufaddiert. Die vier resultierenden Auflösungsstufen P5 bis P2 werden abschließend mit einer 3×3 Faltungsschicht gefiltert. Für die fünfte Auflösungsstufe P6 wird zusätzlich ein Max-Pooling mit einer Schrittweite von 2 auf P5 angewendet.

Für die Bestimmung möglicher relevanter Bildbereich wird ein Region-Proposal-Netzwerk, vergleichbar mit Mask R-CNN[176] und Faster R-CNN [54] verwendet. Im Gegensatz zur Implementierung im Faster R-CNN werden in dieser Implementierung ausschließlich Seitenverhältnisse definiert und keine direkten Ankergrößen. Die verschiedenen Objektgrößen werden durch die Auflösungsstufen des FPN-Backbones abgedeckt, wobei jeder FPN-Stufe eine spezifische Ankergröße zugeordnet wird. Das Region-Proposal-Netzwerk wird auf alle FPN-Schichten P2 bis P6 angewendet und erzeugt für jede Auflösungsstufe Regionsvorschläge (RoIs) sowie deren Objektwahrscheinlichkeiten. Im Anschluss filtert ein Vorschlagsschicht (engl. Proposal Layer) relevante Bildbereiche, welche eine hohe Überschneidung mit anderen relevante Bildbereichen aufweisen oder nur eine geringe Objektwahrscheinlichkeit haben.

Ähnlich wie in Mask R-CNN, wird RoI-Alignment verwendet, um die Regionsvorschläge präzise zu verarbeiten. Da verschiedene RoI-Größen durch unterschiedliche FPN-Stufen erzeugt wurden, kommt Pyramid-RoI-Alignment zum Einsatz. Die passende FPN-Stufe für jedes RoI wird nach der Gleichung 1 in [177] bestimmt, die die RoI-Größe berücksichtigt.

Der Bounding-Box-Zweig für die Objekterkennung verarbeitet abschließend die RoI-Merkmalskarten mit zwei Faltungsschichtenzu einem Merkmalsvektor der Größe 1024. Die erste Schicht filtert ohne Padding mit einem 7×7 Faltungskernel, das zweite verwendet einen 1x1 Kernel. Beide Schichten werden mit Batch Normalization und ReLU-Aktivierung ergänzt. Die Objektklassifizierung wird durch eine vollständig verbundene Schicht mit Softmax-Aktivierungsfunktion realisiert, wohingegen für die Objektlokalisierung eine entsprechende Schicht mit linearer Aktivierung zum Einsatz kommt.

Die Affordanzerkennung wird als separate Encoder-Decoder-Architektur mit Sprungverbindungen implementiert, die sich direkt nach dem ResNet-Backbone vom Rest des Netzes abspaltet. Dabei fungieren die Auflösungsstufen C1 bis C5

als Sprungverbindungen für den Affordanz-Zweig. Diese Architektur vereint die Vorteile semantisch starker Merkmale niedriger Auflösung mit präzisen räumlichen Informationen.

Der Decoder ist symmetrisch zu den Auflösungsstufen der Sprungverbindungen aufgebaut und skaliert die Merkmalskarten auf die halbe Auflösung des Eingabebildes hoch. In der Basisvariante erfolgt die Hochskalierung mittels transponierter Faltung. Jede Decoder-Schicht besteht aus zwei 3×3 Faltungsschichten mit einer Schrittweite von eins und ReLU-Aktivierung. Zur Regularisierung können optional Dropout-Schichten sowie Batch-Normalisierung integriert werden.

Die Schichten der transponierten Faltung (engl. Transposed Convolution) verwenden ebenfalls 3×3-Filterkerne mit einer Schrittweite von 2. Hierdurch wird die räumliche Auflösung der Merkmalskarten in jeder Schicht verdoppelt, während sich ihre Anzahl halbiert. Die Merkmalskarten der Sprungverbindungen werden anschließend mit den Ausgaben der transponierten Faltung konkateniert. Die finale Affordanzsegmentierung erfolgt durch eine 1×1-Faltungsschicht mit anschließender Softmax-Aktivierung auf den Merkmalskarten der letzten Decoder-Schicht.

Ein wesentlicher Aspekt des Region-Proposal-Netzwerk ist, dass dieses auf die Bestimmung spezifischer Klassen trainiert werden muss. Im Rahmen des umgesetzten Konzepts wird hierfür der Weiche-L1-Loss verwendet. Diese Methode zeichnet sich dadurch aus, dass sie eine Kombination aus dem mittleren absoluten Fehler (L1-Loss) und der mittleren quadratischen Abweichung (L2-Loss) darstellt. Dazu wird zunächst die Differenz zwischen Soll- und Ist-Wert berechnet. Für Differenzen, deren Betrag größer oder gleich eins ist, kommt der L1-Loss zur Anwendung. Bei Beträgen kleiner als eins wird hingegen der L2-Loss genutzt. Dies bietet den Vorteil, dass die Robustheit des L1-Loss gegenüber Ausreißern (großen Werten) mit der Stabilität des L2-Loss bei kleinen Abweichungen kombiniert wird. Basierend auf diesem Prinzip wird die Fehlerfunktion L_{rpnreg} formuliert. In dieser bezeichnen y_i die Soll-Werte (Translation und Skalierung der N Ankerboxen) und y_i^p die Ist-Werte der N Regionsvorschläge:

$$L_{rpnreg} = \begin{cases} \frac{1}{N} \cdot \sum_{i=1}^{N} 0.5 \cdot (y_i - y_i^p)^2, & |x| < 1 \\ \frac{1}{N} \cdot \sum_{i=1}^{N} |y_i - y_i^p| - 0.5, & sonst \end{cases} . \tag{6.1}$$

Zum Training der Objektwahrscheinlichkeiten der Regionsvorschläge wird die kategorische Kreuzentropie L_{rpncls} verwendet. Die Gesamtfehlerfunktion des Region-Proposal-Netzwerk ist anschließend über folgende Gleichung definiert:

$$L_{rpn} = w_{rpnreg} \cdot L_{rpnreg} + w_{rpncls} \cdot L_{rpncls} \tag{6.2}$$

In dieser bezeichnen w_{rpnreg} und w_{rpncls} anpassbare Hyperparameter für die Gewichtung der Fehlerfunktion.

In der Objekterkennung finden ebenfalls zwei Fehlerfunktionen Anwendung: eine für die Objektlokalisierung und eine für die Klassifizierung. Für die Objektlokalisierung wird erneut auf den Weichen-L1-Loss zurückgegriffen, in diesem Fall als L_{objreg} bezeichnet. Für die Objektklassifizierung wird, wie zuvor, die kategorische Kreuzentropie genutzt (L_{objcls}). Die Gesamtfehlerfunktion L_{obj} für die Objekterkennung setzt sich daher wie folgt zusammen:

$$L_{obj} = w_{objreg} \cdot L_{objreg} + w_{objcls} \cdot L_{objcls} \tag{6.3}$$

Auch hier fungieren w_{objreg} und w_{objcls} als Hyperparameter zur Gewichtung der Fehleranteile.

Für den Affordanzerkennungs-Zweig wird als Verlustfunktion der Focal-Loss L_{aff} verwendet. Die Wahl dieser Funktion ist in der Beschaffenheit des Datensatzes begründet, welcher keine gleichmäßige Klassenverteilung aufweist. Insbesondere die Hintergrundklasse ist im Vergleich zu den eigentlichen Affordanzklassen stark dominant. Der Einsatz des Focal-Loss ermöglicht es, schwer klassifizierbare Beispiele stärker zu gewichten und so dem Ungleichgewicht entgegenzuwirken. Über den Hyperparameter w_{aff} kann auch dieser Fehleranteil im Gesamtnetz gewichtet werden.

Die Gesamtfehlerfunktion L_{all} für das gesamte Netzwerk ergibt sich entsprechend aus der Summe der gewichteten Komponenten:

$$\begin{aligned} L_{all} = {} & w_{rpnreg} \cdot L_{rpnreg} + w_{rpncls} \cdot L_{rpncls} + w_{objreg} \cdot L_{objreg} \\ & + w_{objcls} \cdot L_{objcls} + w_{aff} \cdot L_{aff} \end{aligned} \tag{6.4}$$

Zur Evaluation der Vorhersagegüte kommen verschiedene Metriken zum Einsatz. Von besonderer Relevanz sind hierbei die Objektlokalisierung und die Affordanzerkennung. Dementsprechend wird für die Regression im Region-Proposal-Netzwerk sowie für die Objektlokalisierung der Jaccard-Koeffizient (Intersection over Union, IoU) herangezogen. Dieser stellt ein relatives Maß für die Überlappung zwischen der Bounding-Box des Soll-Werts (engl. Ground Truth) und der vorhergesagten Bounding-Box dar.

Im Rahmen der Affordanzerkennung werden zwei Metriken genutzt. Mittels IoU wird die Überlappung der korrekten Segmentierung mit der Vorhersage für jede Affordanzklasse verglichen. Ergänzend wird der F1-Score verwendet, welcher das harmonische Mittel aus Genauigkeit (engl. Precision) und Trefferquote (engl. Recall) bildet. Dies ermöglicht neben der Beurteilung der Vorhersagegenauigkeit auch eine Aussage über die Robustheit der Segmentierung gegenüber Fehlklassifikationen.

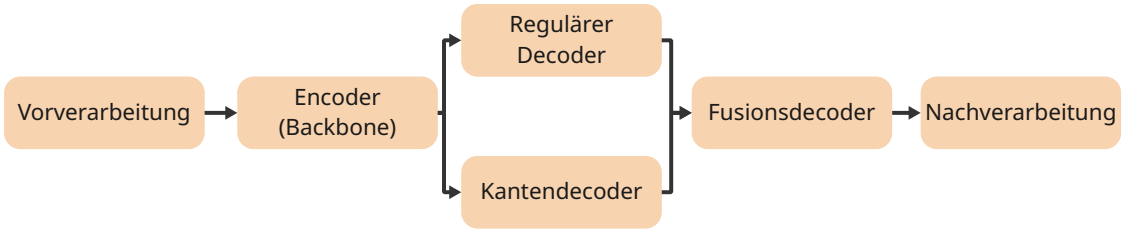


Abbildung 6.1.: Erweiterte Architektur der Affordanzdetektion mit zwei parallelen Zweigen und einem kombinierenden Fusionsdecoder.

6.2.3. Multistream Erweiterung

Der bisher vorgestellte Ansatz für die Erkennung der Montageaffordanzen verfolgt eine Single-Stream-Architektur. Dabei werden alle in den Eingangsdaten enthaltenen Merkmale (Form, Farbe, Texturen) durch ein einziges neuronales Netz verarbeitet. Ein solcher Ansatz stellt in der Regel einen Kompromiss zwischen der notwendigen Netzgröße und der erreichbaren Genauigkeit bzw. Erfolgsrate dar.

Im Fall von Montageaffordanzen ist es für ein solches Netzwerk besonders herausfordernd, feine Strukturen, wie z.,B. Gewinde, zuverlässig zu erkennen. Um diese Einschränkung zu adressieren, wird der Decoder des Single-Stream-Ansatzes nach dem Vorbild des Gated-SCNN [178] um zwei Komponenten erweitert: einen speziellen Kantendecoder (engl. Edge-Decoder) und einen nachgeschalteten Fusion-Decoder. Die so angepasste Struktur ist in Abbildung 6.1 dargestellt.

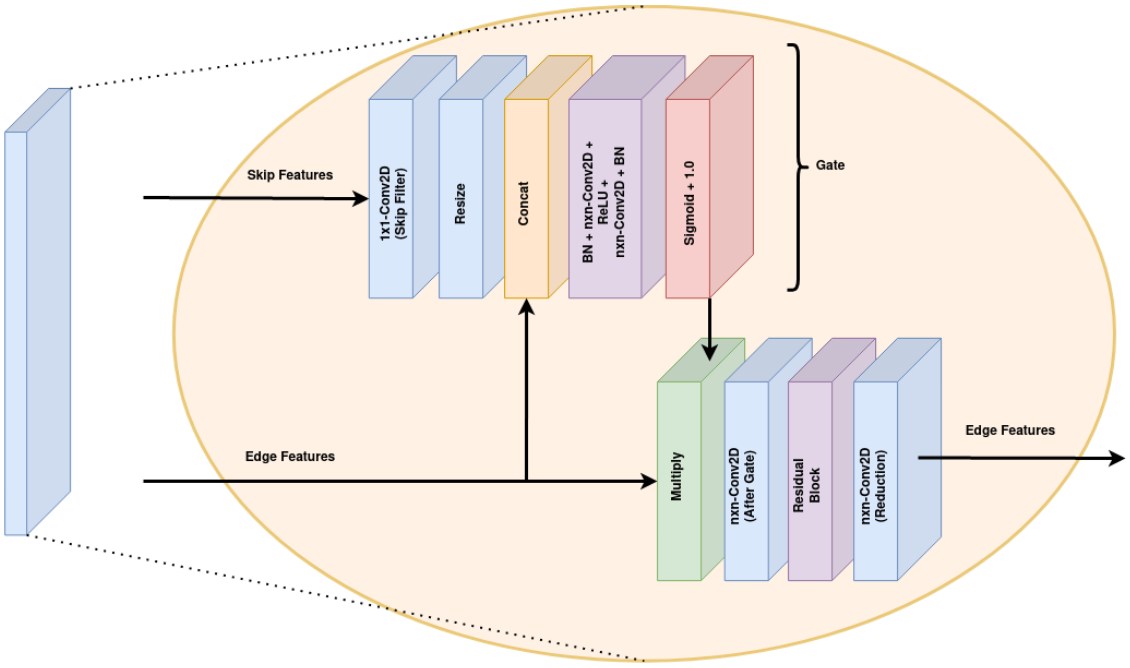


Abbildung 6.2.: Architektur des Kantendecoder.

Kantendecoder: Der hinzugefügte Kantendecoder, dargestellt in Abbildung 6.2, zielt ausschließlich auf die Erkennung von Kanten ab – insbesondere von feinen Strukturen wie Gewinden. Das zentrale Element des Kantendecoders bildet das Edge-Gate. Dieses Modul filtert die Merkmale mithilfe einer Aufmerksamkeitskarte (engl. Attention Map). Hierzu greift es auf zwei unterschiedliche Merkmalsvektoren zu:

- Gefilterte Sprung-Merkmale (engl. Filtered Skip Features): Diese Merkmale stammen aus der ersten Schicht des Encoders und weisen daher eine hohe Ähnlichkeit zum Eingangsbild auf. Sie enthalten somit detaillierte Kanten- und Konturinformationen. Um die Kantendetektion weiter zu verbessern, werden die Sprung-Merkmale zunächst durch einen Residual-Block gefiltert und anschließend durch eine $n \times n$-Faltungsschicht (Conv2D) reduziert.
- Kantenmerkmale (engl. Edge Features): Diese Merkmale wurden bereits im Decoder verarbeitet. Sie repräsentieren daher die vom Decoder erlernten, relevanten Kanteninformationen.

Die Aufmerksamkeitskarte verarbeitet die beiden Merkmalstypen, indem diese zunächst konkateniert und anschließend durch eine $n \times n$-Faltungsschicht sowie eine nachfolgende ReLU-Aktivierung geleitet werden. Abschließend wird die Ausgabe durch eine Sigmoid-Funktion skaliert.

Fusionsdecoder (Fusion Decoder): Da der reguläre Decoder und der Kantendecoder zwei unterschiedliche Merkmalstypen erzeugen - semantische Merkmale und Kantenmerkmale - müssen diese anschließend zusammengeführt werden. Dies geschieht durch einen Fusionsdecoder, dessen Aufbau in Abbildung 6.3 dargestellt ist.

Im ersten Schritt konkateniert der Fusionsdecoder die beiden Merkmalsvektoren, bevor eine sequenzielle Verarbeitung durch mehrere Faltungsschichten er-

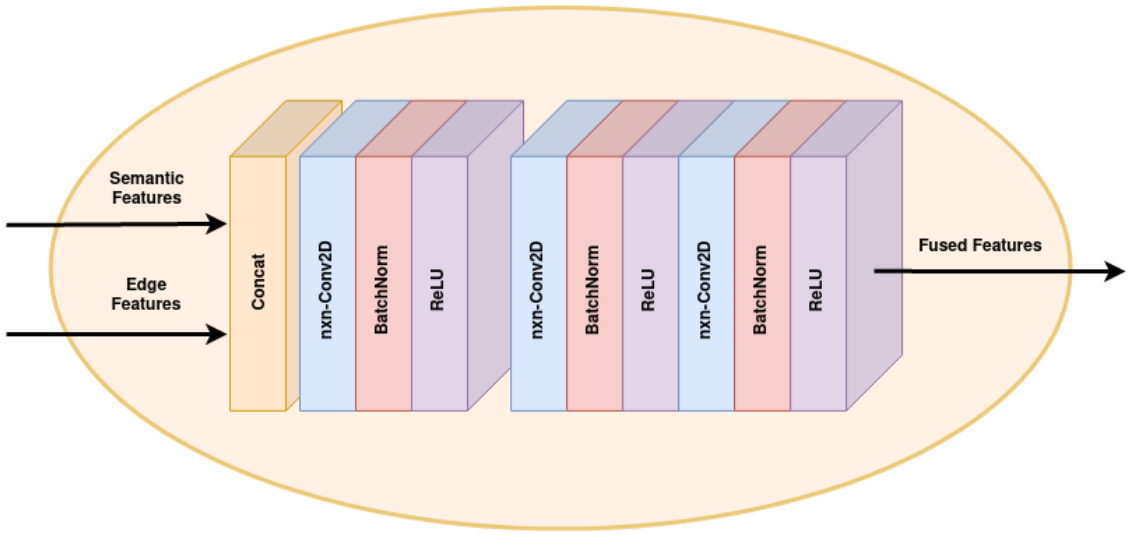


Abbildung 6.3.: Architektur des Fusion Decoder.

folgt. Jede dieser Schichten setzt sich aus einer $n \times n$-Faltung (Conv2D), gefolgt von einer Batch-Normalisierung und einer ReLU-Aktivierungsfunktion zusammen. Das Resultat des Fusionsdecoders wird als fusionierte Merkmale (engl. Fused Features) bezeichnet. Diese werden anschließend analog zum Single-Stream-Modell weiterverwendet, um die Montageaffordanzen zu identifizieren.

6.3. Montageplanung auf Basis von Montageaffordanzen

Für die Ableitung von konkreten Montageaktionen in jedem Schritt der Montagesequenz wird ein semantischer Planer verwendet. Dieser kombiniert die detektierten Affordanzen mit der zuvor berechneten durchführbaren Montagesequenz. Hierbei werden auch vorherige und nachgelagerte Montageschritte berücksichtigt, da sie Einfluss auf die zu betrachtenden Montageschritte haben. Durch diese Betrachtung ist es auch möglich stoffschlüssige Verbindungen zu

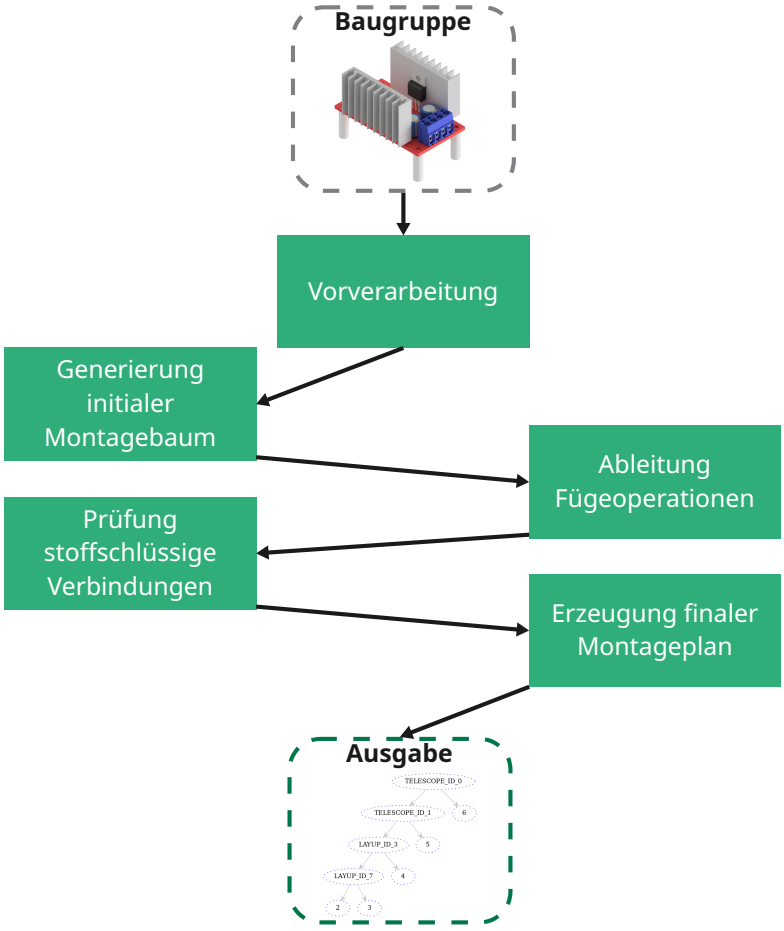


Abbildung 6.4.: Aufbau des Planers.

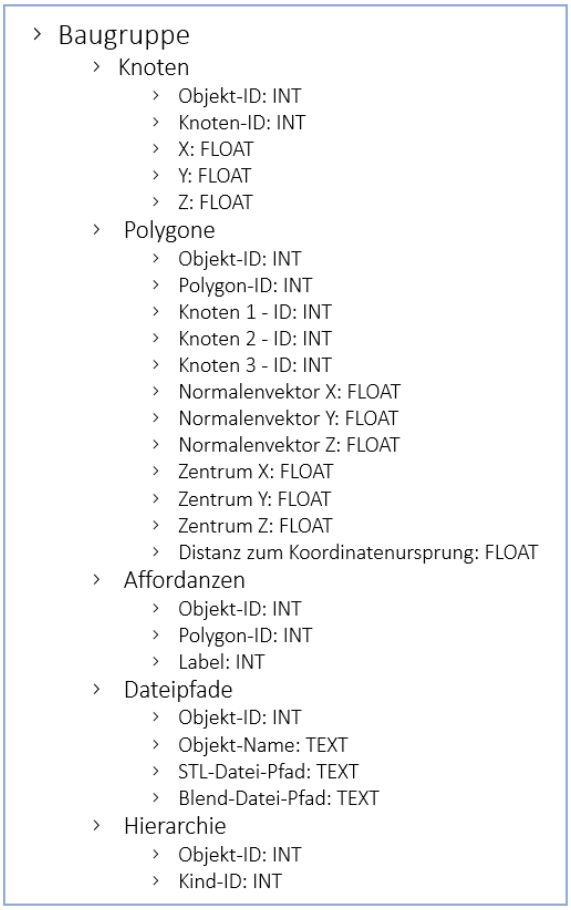


Abbildung 6.5.: Das vom semantischen Planer verwendete Informationsmodell, welches aus den CAD Daten extrahiert wird.

berücksichtigen. Die abgeleiteten Montageaktionen entsprechen dabei den in Abschnitt 6.1 definierten. Als zusätzliche Anforderung gilt, dass kein Bauteil lose, also ausschließlich über die Operation LAYUP, mit der Baugruppe verbunden sein darf.

Zur Bestimmung aller notwendigen Montageaktionen führt das System die in Abbildung 6.4 dargestellten Schritte aus. Diese werden im folgendenden genauer beschrieben.

Vorverarbeitung
Die Planungsgrundlage für den semantischen Planer stellen konkret das 3D-Modell der Baugruppe, die Montagesequenz sowie die identifizierten Affordanzen der einzelnen Bauteile dar. Hierbei liegt das 3D-Modell als STEP-Datei und das die identifizierten Affordanzen in Form eines Blender-Szenen-File vor. Eine Mehrzahl der in den Daten enthaltenen Informationen liegen in einer komplexen Datenstruktur vor oder sind in ihrem initialen Format für das spätere System nicht direkt verarbeitbar. Um dieser Problematik zu begegnen, findet eine Vorverarbeitung der Daten statt, welches die notwendigen Informationen aus den ursprünglichen Datenformaten extrahiert und für den einfachen Zugriff in einer

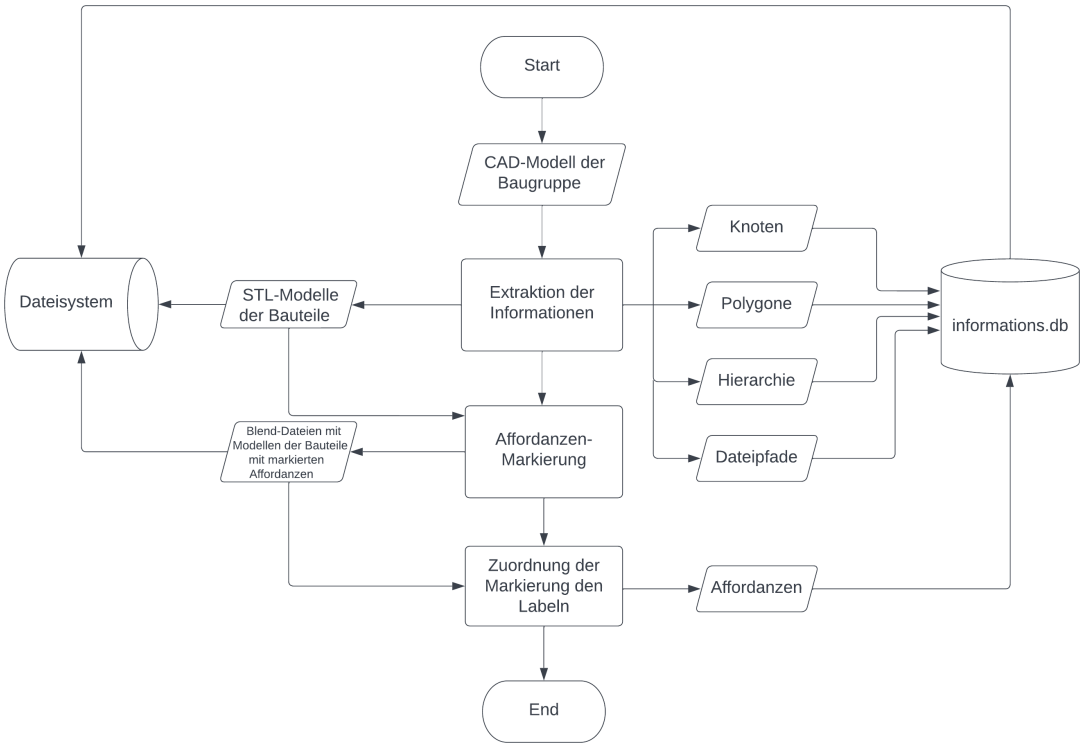


Abbildung 6.6.: Ablauf der Datenextraktion zur Erstellung des Informationsmodells.

Datenbank speichert. Die zu extrahierenden Informationen und deren Datentyp sind in Abbildung 6.5 dargestellt. Die Datenextraktion erfolgt dabei nach dem in Abbildung 6.6 aufgezeigten Ablauf. In einem ersten Schritt extrahiert das System aus dem vorliegenden STEP-Datei u.a. Informationen über die Zugehörigkeit der Bauteile zu Unterbaugruppen, das jeweilige Oberflächenmodell der einzelnen Komponenten, sowie deren Endpose. Auch die Informationen über die Zuordnung der Affordanzen zu den einzelnen Bauteilen wird in die Datenbank übertragen. Sollte das in Abschnitt 6.2.2 beschriebene KI-System zur Detektion der Affordanzen nicht für die vorliegende Baugruppe nutzbar sein, kann an dieser Stelle auch auf ein manuelles Markierungssystem vergleichbar zu dem im Abschnitt 6.2.1 beschriebenen zurückgegriffen werden.

Generierung des initialen Montagebaums
Im ersten Schritt übernimmt das System die erhaltene Montagesequenz und überführt diese in eine Montagebaumdarstellung. Hierbei bilden die Blätter des Baumes die einzelnen Bauteile ab und die Knoten stellen den jeweiligen Montageschritt dar. Entsprechend lassen sich folgende Verhältnisse abbilden:

- Bauteil wird an Bauteil montiert. Hierdurch entsteht eine (Unter-)Baugruppe.
- Bauteil wird an Baugruppe montiert.
- Bauteil wird an Unterbaugruppe montiert.

- Unterbaugruppe wird an zweite Unterbaugruppe montiert.
- Unterbaugruppe wird an (Haupt-)Baugruppe montiert.

Initial sind alle Knoten im Baum mit Platzhalter *DEFAULT* markiert, welcher in den folgenden Schritten durch die korrekte Fügeoperationen ersetzt wird.

Ableiten der Montageoperation
Zur Ableitung der potenziellen Fügeoperationen verwendet der Planer das in Abbildung 6.7 skizierte Logikschema. Um die korrekte Fügeoperation abzuleiten, nutzt das System Affordanzen-Paare. Dabei ist zu beachten, dass pro infrage kommenden Affordanzen-Paares *(affordanz_1, affordanz_2)* sowohl das direkte Paar als auch das inverse Paar *(affordanz_2, affordanz_1)* betrachtet wird. Diese bidirektionale Betrachtung ermöglich eine vollständige Erfassung aller möglichen Affordanzen-Interaktion zwischen zwei Bauteilen.

Bei Affordanzen-Paaren wird unterschieden zwischen gültigen Paaren, aus welchen sich eine Fügeoperation ableiten lässt, und ungültigen Paaren, in denen

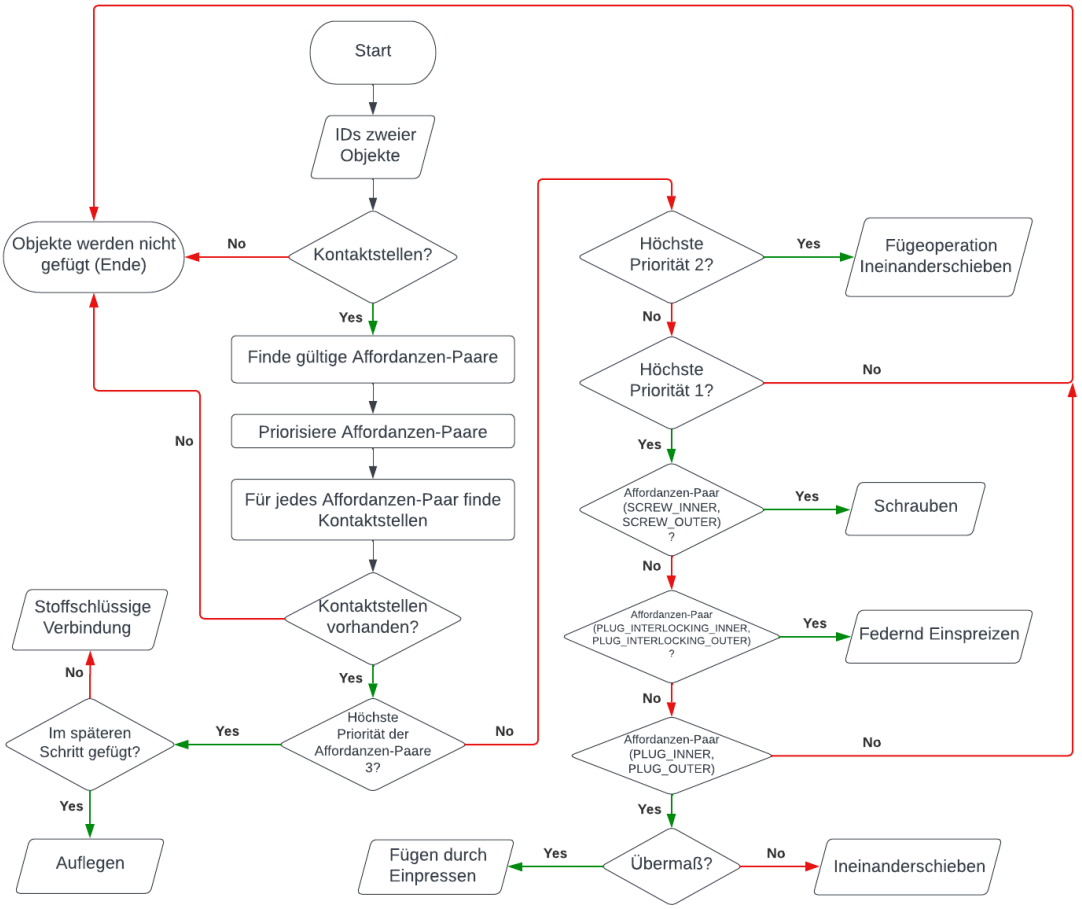


Abbildung 6.7.: Logik zur Ableitung der korrekten Fügeoperation auf Basis der detektierten gülitgen Affordanzenpaare

Affordanzen-Paar	Priorität
LAYUP, LAYUP	3
SCREW_OUTER, PLUG_INNER	2
PLUG_OUTER, PLUG_INNER	1
SCREW_OUTER, SCREW_INNER	1
PLUG_INTERLOCKING_OUTER, PLUG_INTERLOCKING_INNER	1

Tabelle 6.4.: Prioritäten der Affordanzen-Paare.

die Affordanzen keine Gemeinsamkeiten haben. Gültige Paare werden mit einer spezifischen Priorität versehen, die dessen Relevanz für die Ableitung der Fügeoperation festlegen. Eine Aufstellung aller als gültig betrachteten Paarungen und die zugehörige Priorität ist in Tabelle 6.4 zu finden, wird 1 als die höchste Priorität gewertet und 3 als die geringste. An den Beispielbauteilen in Abbildung 6.8 lässt sich erkennen, dass als Affordanzen-Paare *(LAYUP, LAYUP)* und *(PLUG_INNER, PLUG_OUTER)* für die Kontaktstellen gültig sind. Jedoch ist das Paar *(PLUG_INNER, PLUG_OUTER)* aufgrund seiner spezifischen Funktionalität für die Montageaktion ausschlaggebend. Ein zentraler Grundsatz der Methodik liegt in der Annahme von gleichartigen Verbindungen bei multiplen Fügestellen. Diese gilt sowohl für Situationen, in denen ein Bauteil an einer Fügestelle mehrere Bauteile gleichzeitig verbindet (ein Beispiel hierfür ist ein Bolzen der durch die Bohrungen zweier Platten geführt wird), als auch für mehrfache Verbindungen zwischen denselben Bauteilen. Im Beispiel in Abbildung 6.8 sind beide Noppen über *(PLUG_INNER, PLUG_OUTER)* definiert.

In der Ausführung analysiert die Logik zunächst zwei Bauteile darauf, ob diese Kontaktstellen miteinander besitzen. Sollte dieses nicht der Fall sein, also das Bauteil-Paar räumlich voneinander getrennt sein, wird es nicht weiter betrachtet.

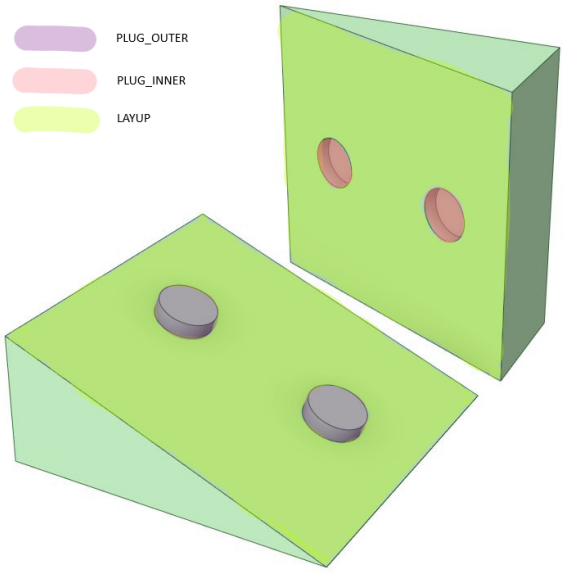


Abbildung 6.8.: Beispiel von zwei Bauteilen mit markierten Affordanzen.

Im anderen Fall bestimmt das System zunächst, ob sich durch die Kontaktstellen gültige Affordanzen-Paare ergeben, und speichert diese mit ihrer entsprechenden Priorität ab. Diese Analyse wird für alle Bauteilepaarungen, welche sich aus dem zuvor aufgestellten Montagebaum ergeben durchgeführt.

Nach Abschluss der Kontaktstellenanalyse erfolgt die eigentliche Ableitung der Fügeoperationen durch einen mehrstufigen Auswahlprozess. Bei jedem Bauteil-Paar, für welches die ein gültiges Affordanzen-Paar identifiziert werden konnte, wird das Paar mit der höchsten Priorität berücksichtigt.

- Priorität 3: Bei der dieser Prioritätsstufe wird automatisch die Fügeoperation *LAYUP* abgeleitet, diese entspricht typischerweise Auflege- und Positionierungsoperationen.
- Priorität 2: Bei Bauteil-Paaren mit der höchsten Priorität 2 wird die Montageaktion *TELESCOPE* abgeleitet.
- Priorität 1: Bei Priorität 1 existiert theoretisch die Möglichkeit das verschiedene Typen von Affordanzen-Paaren existieren. Da jedoch gleichartige Fügestellen angenommen werden, kann pro Bauteil-Paar nur ein Typ von Affordanzen-Paar der Priorität 1 auftreten. Möglich sind entsprechend folgende Fälle (inkl. ihrer Inverse):
  - Affordanzen-Paar *(SCREW_OUTER, SCREW_INNER)*: In diesem Fall kann direkt die Fügeoperation *SCREW*, also eine Schraubverbindung abgeleitet werden.
  - Affordanzen-Paar *(PLUG_INTERLOCKING_OUTER, PLUG_INTERLOCKING_INNER)*: Bei dieser Paarung wird die Montageaktion *ELASTIC* abgeleitet, also eine elastische Verbindung beispielsweise über einen Rastmechanismus.
  - Affordanzen-Paar *(PLUG_ OUTER, PLUG_INNER)*: Tritt dieser Fall auf, prüft das System zunächst beide Bauteile, ob ein Übermaß vorliegt. Ist die Überprüfung positiv, so wird die Montageoperation *PRESS* abgeleitet, andernfalls wird die Montagetätigkeit *TELESCOPE* zugewiesen.

Prüfung auf stoffschlüssigen Verbindungen
Nachdem initial alle Fügeoperationen für alle Montageschritte abgeleitet wurden, erfolgt abschließend eine Validierung des semantischen Montagebaumes. Dabei werden alle Knoten, die mit der Montageaktion LAYUP klassifiziert sind, geprüft.

Ziel der Überprüfung besteht darin, festzustellen, ob die im Montageschritt gefügten Bauteile nachfolgend durch eine der mechanischen Verbindungsoperationen (*ELASTIC*, *PRESS*, *SCREW* oder *TELESCOPE*) mit anderen Komponenten fixiert werden. Wird bei der Analyse festgestellt, dass bestimmte Bauteile ausschließlich durch die LAYUP-Operation positioniert sind, handelt es sich um eine lose Komponente und sie gilt daher als nicht montiert. In diesen Fällen erfolgt eine automatische Korrektur der Fügeoperation von *LAYUP* zu Material, da

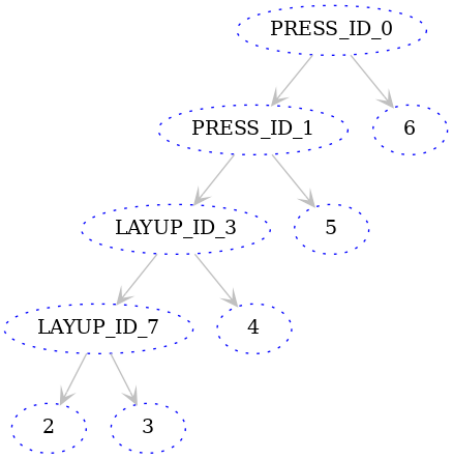


Abbildung 6.9.: Beispiel eines finalen Montagebaums.

nur durch eine stoffschlüssige Verbindung wie Kleben oder Schweißen eine feste Fixierung des Bauteils an die Baugruppe möglich ist und so die strukturelle Integrität der Baugruppe gewährleistet werden kann.

Hierdurch ist sichergestellt, dass der semantische Montagebaum einen realistischen und praktisch umsetzbaren Montageplan repräsentiert, bei dem alle Komponenten korrekt fixiert sind.

Erzeugung von Montageplan / Visualisierung des finalen Montagebaums
Final werden die abgeleiteten Montageaktionen in den zu Beginn erzeugten Montagebaum eingetragen, so dass diese den *DEFAULT* Operator ersetzen, vgl. Abbildung 6.9. Der erzeugte Baum kann anschließend visualisiert werden, was vor allem für einen menschlichen Anwender eine einfache Übersicht über den Montageplan ermöglicht. Für eine maschinelle Weiterverarbeitung wird der Ergebnis-Montagebaum in serialisierter Form für die Textgenerierung verwendet. Ein Beispiel eines simplen Montageplans ist in Abbildung 6.10 zu sehen.

tree.dot × instruction.txt ×

```
1. Das Bauteil mit der Objekt-ID 2 und dem Namen "2_Platte1" wird mit dem Bauteil mit
der Objekt-ID 3 und dem Namen "3_Platte2" durch die Fügeoperation "Auflegen"
zusammengesetzt.
2. Die Untergruppe aus dem Schritt 1 wird mit dem Bauteil mit der Objekt-ID 4 und dem
Namen "4_Platte3" durch die Fügeoperation "Auflegen" zusammengesetzt.
3. Die Untergruppe aus dem Schritt 2 wird mit dem Bauteil mit der Objekt-ID 5 und dem
Namen "5_Stift1" durch die Fügeoperation "Fuegen durch Einspressen" zusammengesetzt.
4. Die Untergruppe aus dem Schritt 3 wird mit dem Bauteil mit der Objekt-ID 6 und dem
Namen "6_Stift2" durch die Fügeoperation "Fuegen durch Einspressen" zusammengesetzt.
```

Abbildung 6.10.: Textuelle Form des Montageplans.

6.4. Zusammenfassung und Fazit Montageaffordanzen

Dieses Kapitel fokussiert sich auf die notwendige Überführung der zuvor generierten abstrakten Montagesequenz in einen konkreten, roboter-ausführbaren Handlungsplan. Zentrales Element ist hierbei das Konzept der *Montageaffordanzen*, welches psychologische Wahrnehmungsmodelle auf die technische Montage überträgt. Es beschreibt, wie spezifische geometrische Merkmale von Bauteilen (z. B. Gewinde, Bohrungen oder Rastnasen) bestimmte Fügeoperationen gemäß der Norm DIN 8593 implizieren.

Um diese Merkmale in unstrukturierten Umgebungen zuverlässig zu erkennen, wird ein Bildverarbeitungssystem auf Basis von Faltungsnetzen (z. B. Mask R-CNN) entwickelt. Zur Überwindung der Datenknappheit beim Training dieser Netze kommt ein hybrider Ansatz zum Einsatz: Ein synthetischer Datensatz, erzeugt durch eine automatisierte Pipeline in Blender, wird mit realen Aufnahmen kombiniert. Dies steigert die Robustheit der Erkennung gegenüber Lichtverhältnissen und Texturen signifikant. Ein darauf aufbauender hierarchischer semantischer Planer fusioniert die visuellen Affordanz-Informationen mit der vorgegebenen Montagesequenz und den CAD-Daten. Durch das Verbindung korrespondierender Affordanz-Paare (z. B. Schraube und Mutter) leitet der Planer vollautomatisch die korrekten Fügeverfahren und die notwendigen Werkzeuge ab. Das System ist zudem intelligent genug, stoffschlüssige Verbindungen wie Kleben vorzuschlagen, wenn keine mechanische Arretierung erkannt wird. Das Ergebnis ist ein detaillierter Montagebaum, der alle Parameter für die physische Robotersteuerung enthält.

7. Experimente und Diskussion

In diesem Kapitel werden die verschiedenen Experimente vorgestellt, die zur Evaluation der entwickelten Konzepte durchgeführt wurden, und deren Ergebnisse diskutiert.

Zunächst liegt der Fokus auf Experimenten im Zusammenhang mit der Montagesequenzgenerierung. In Vorversuchen mit unterschiedlichen Baugruppen wird die korrekte Funktionsweise des genetischen Algorithmus evaluiert, bevor mit den daraus erzeugten Datensätzen der Sequenz-zu-Sequenz-Ansatz untersucht wird. Abschließend erfolgt die Evaluation des LLM-basierten Ansatzes unter Verwendung eines dedizierten Experimentsatzes.

Nach der Untersuchung der verschiedenen Verfahren zur Montagesequenzgenerierung werden zudem die Ansätze zur Montageaffordanzdetektion sowie die Montageplanverfeinerung analysiert.

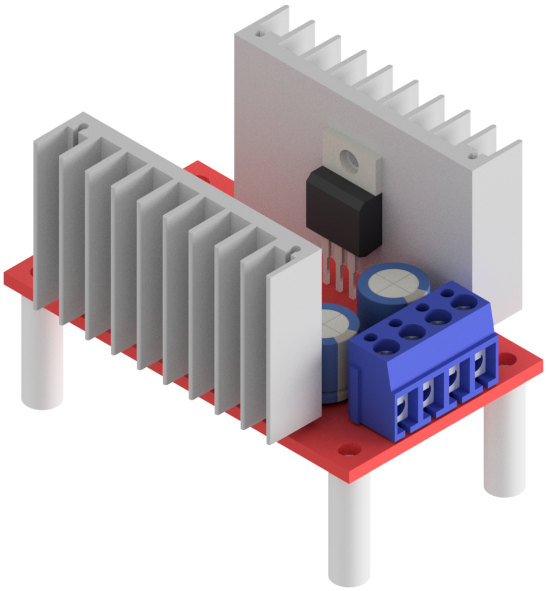
7.1. Genetischer Algorithmus

Bei den Experimenten zum genetischen Algorithmus stehen drei Aspekte im Fokus: Leistungsfähigkeit, Robustheit und Effizienz des angepassten genetischen Algorithmus. Die Evaluation zielt darauf ab, die Auswirkungen verschiedener Parameter und Modifikationen auf die Qualität der erzeugten Montagesequenzen zu quantifizieren und das Verhalten bei verschiedenen realitätsnahen Baugruppen zu untersuchen.

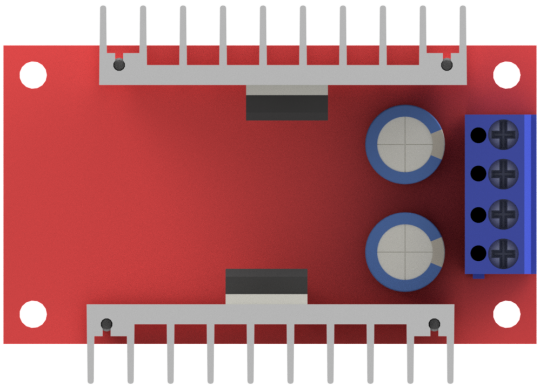
7.1.1. Datensatz und Evaluationsumgebung

Die Evaluation erfolgt anhand von vier verschiedenen Datensatzklassen, die unterschiedliche Komplexitätsgrade und Montageanforderungen repräsentieren. Einige der verwendeten Modelle wurden modifiziert, um Anforderungen wie geradlinige, achsenparallele Fügetrajektorien oder die korrekte Orientierung der Baugruppe (Schwerkraft in negativer x-Richtung) sicherzustellen.

Die erste Datensatzklasse besteht aus 15 verschieden bestückten Leiterplatinen (vgl. Abbildung 7.1) variierender Komplexität. Die Anzahl der Bauteile pro Baugruppe liegt zwischen 9 und 17. Eine Besonderheit ist das Fehlen von Unterbaugruppen. Die Bauteile weisen primär Fügetrajektorien entlang der x-Achse auf



(a) Seitenansicht der Leiterplatine.

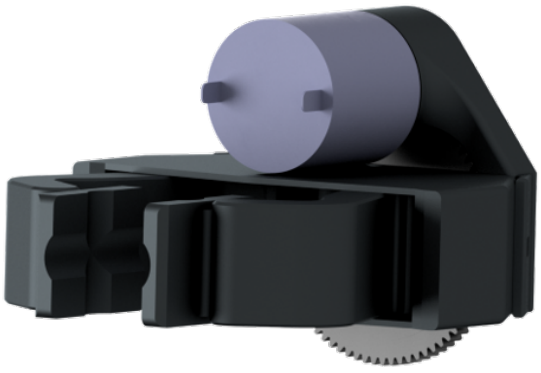


(b) Draufsicht auf die Leiterplatine.

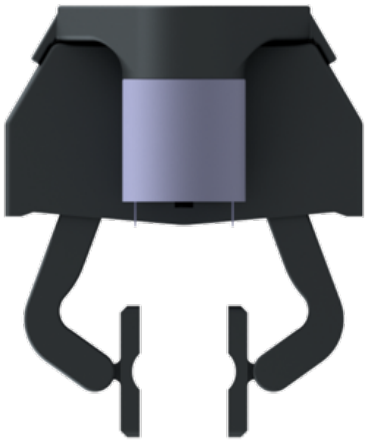
Abbildung 7.1.: Eine Variante der Leiterplatinen aus zwei verschiedenen Perspektiven.

und werden größtenteils senkrecht auf die Platine aufgebracht. Diese Platinen stellen für den genetischen Algorithmus eine vergleichsweise einfache Problemstellung dar, da nur wenige räumliche Abhängigkeiten bestehen und somit eine hohe Anzahl valider Montagesequenzen existiert.

Für die zweite Datensatzklasse werden Varianten einfacher mechanischer Greifer verwendet (vgl. Abbildung 7.2). Die neun betrachteten Modelle umfassen 6 bis 16 Bauteile. Die Komplexität ist höher als bei den Leiterplatinen, da die Kompo-

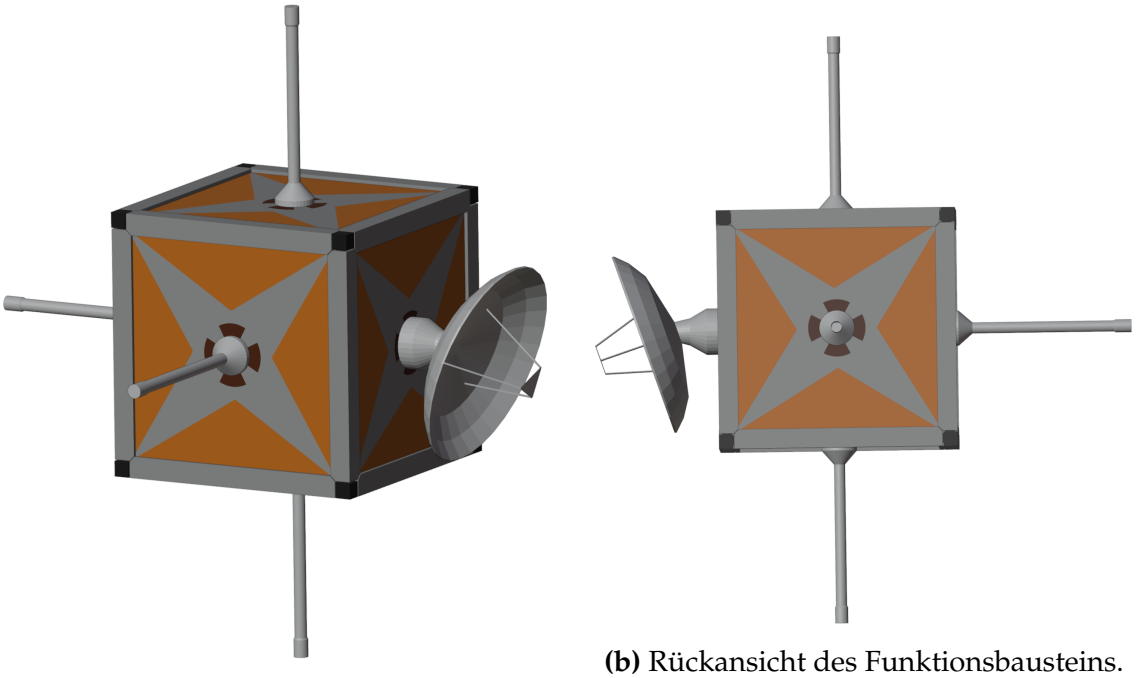


(a) Seitenansicht des Greifers.



(b) Draufsicht auf den Greifer.

Abbildung 7.2.: Eine Variante des mechanischen Greifers aus zwei verschiedenen Perspektiven.



(a) Frontansicht des Funktionsbausteins.

(b) Rückansicht des Funktionsbausteins.

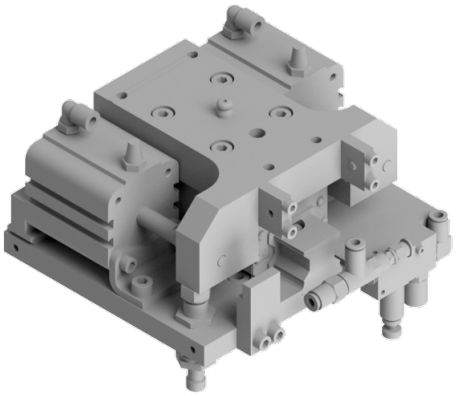
Abbildung 7.3.: Variante eines Mockups eines Funktionsbausteins aus zwei Perspektiven.

nenten stärker ineinandergreifen. Da der genetische Algorithmus Materialverformungen (z. B. Silikon-Greifflächen) nicht nativ verarbeiten kann, wurden Greiferbacke und flexible Fläche als ein gemeinsames Bauteil modelliert. Aufgrund der variierenden Fügerichtungen dient dieser Datensatz zur Beurteilung der Erkennung von Montageabhängigkeiten bei moderater Bauteilanzahl.

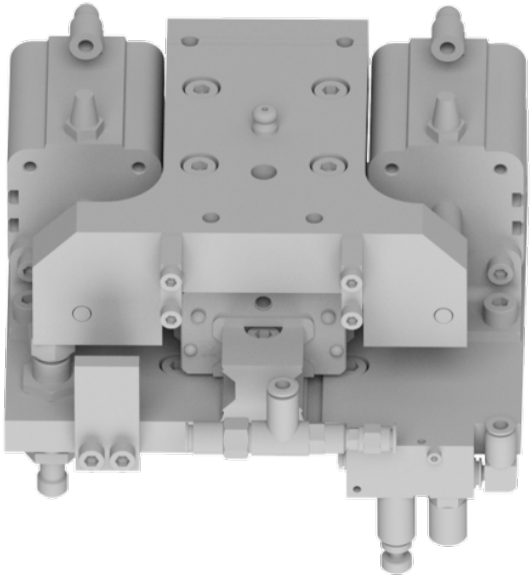
Der dritte Datensatz umfasst 20 Varianten eines Funktionsbausteins für modulare Satelliten (vgl. Abbildung 7.3). Es handelt sich um vereinfachte Mockups mit 11 bis 19 Bauteilen. Die würfelförmige Grundstruktur ist identisch, während variierende Zusatzkomponenten (Antennen, Schubdüsen, Batterien, Tanks) die Komplexität erhöhen. Viele Bauteile müssen montiert werden, bevor das Gehäuse geschlossen wird, was restriktive Reihenfolgen erzwingt.

Als vierte Klasse dient eine Baugruppe aus dem Sondermaschinenbau mit 54 bis 66 Einzelteilen (vgl. Abbildung 7.4). Die Montagesequenz unterliegt sehr restriktiven Bedingungen. Zur Verarbeitung durch den genetischen Algorithmus wurden Komponenten wie Pneumatikzylinder als fertige Unterbaugruppen (verschmolzene Geometrien) behandelt. Dennoch stellt dieser Datensatz einen hochkomplexen, realen Anwendungsfall dar.

Die Experimente wurden auf einem System mit Intel Core i7-7700HQ, 16 GB RAM und einer NVIDIA GeForce GTX 1080 (8 GB VRAM) durchgeführt. Evaluiert wurde der in 5.1.6 vorgestellte optimierte GA-Ansatz auf Blender-Basis.



(a) Seitenansicht der Sonderbaugruppe.



(b) Draufsicht auf die Sonderbaugruppe.

Abbildung 7.4.: Die Sonderbaugruppe aus zwei verschiedenen Perspektiven.

7.1.2. Durchgeführte Experimente und Ergebnisse

Die Evaluation untersucht verschiedene Aspekte des genetischen Algorithmus. Als einflussreiche Parameter wurden die Anzahl der Elternindividuen (Varianzübertragung) sowie die Populationsgröße (Abdeckung des Suchraums) identifiziert. Getestet wurden Kombinationen aus Elternanzahl (2, 4, 8) und initialer Populationsgröße (10, 20, 40).

Zunächst wurde die Fitnessentwicklung über die Generationen analysiert. Es zeigte sich, dass eine Konvergenz über alle Datensätze hinweg bei 50 Generationen erreicht wird. Eine weitere Erhöhung führte zu keiner signifikanten Verbesserung. Während Greifer und Funktionsbausteine bereits nach ca. 40 Generationen konvergierten, erreichte die stark restriktierte Sonderbaugruppe diesen Zustand noch früher.

Ein Schwerpunkt lag auf der Initialisierungsstrategie. Der entwickelte deterministische Dekonstruktionsansatz, der mindestens eine valide Sequenz garantiert, wurde gegen eine rein zufällige Initialisierung evaluiert. Das Ergebnis bestätigt den Vorteil des neuen Ansatzes bei komplexen Modellen: Während bei Platinen und Greifern beide Ansätze ähnliche Ergebnisse lieferten, scheiterte die Zufallsinitialisierung bei den komplexen Baugruppen vollständig an der Findung einer validen Sequenz.

Die Variabilität der Individuen wurde ebenfalls geprüft, um Redundanz in den Populationen auszuschließen. Trotz der Optimierung nahm die Kollisionsrate (Notwendigkeit der Anpassung neu erzeugter Individuen) über die Zeit nicht signifikant zu, was auf eine stabile Diversität hindeutet. Dass Bestlösungen oft

Datensatzklasse	Initialisierung (Ø)	Minimale Gesamtlaufzeit	Maximale Gesamtlaufzeit	Durchschnittliche Gesamtlaufzeit
Leiterplatinen	3,223 s	45,265 s	96,383 s	59,409 s
Greifer	4,017 s	72,247 s	117,957 s	87,335 s
Satellitenmodul	16,315 s	110,598 s	308,654 s	119,707 s
Sonderbaugruppe	811,230 s	5251,554 s	8658,744 s	6955,149 s

Tabelle 7.1.: Übersicht der Laufzeiten des genetischen Algorithmus, unterteilt in Initialisierung und Gesamtlaufzeit.

erst im letzten Drittel der Generationen gefunden wurden, belegt die Effektivität des Prozesses über die gesamte Laufzeit.

Hinsichtlich der Ausführungszeit wurde evaluiert, bis zu welcher Komplexität der genetischen Algorithmus für eine Online-Verwendung geeignet ist (Zielschwelle: < 2 Min.). Bei einer Elternanzahl von 4 und einer Population von 40 zeigt Tabelle 7.1 einen deutlichen Laufzeitsprung bei der Sonderbaugruppe aufgrund der hohen Bauteilanzahl. Platinen und Greifer blieben stets unter zwei Minuten. Bei den Satellitenmodulen lag der Mittelwert knapp unter der Schwelle, wobei einzelne Varianten diese überschritten.

7.1.3. Diskussion und Einordnung der Ergebnisse

Die Evaluation belegt, dass das System für unterschiedliche Komplexitätsgrade autonom gültige Montagesequenzen erzeugt. Die intelligente Initialisierung ist hierbei essenziell für komplexe Baugruppen. Die Ausführungszeit ist bei hochkomplexen Modellen für dynamische Online-Szenarien zu hoch. Da der genetischen Algorithmus jedoch primär zur Generierung von Trainingsdaten für KI-Modelle dient, die vor dem Einsatz trainiert werden, ist die Laufzeit im produktiven Betrieb zweitrangig. Robustheit und Varianz der erzeugten Daten stehen hier im Vordergrund.

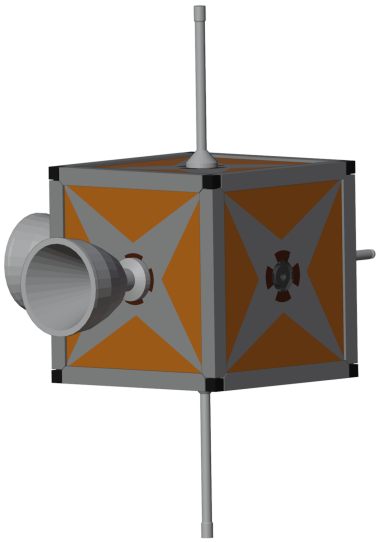
7.2. Sequenz-zu-Sequenz-basierte Montagesequenzgenerierung

Für die Experimente im Zusammenhang mit dem Sequenz-zu-Sequenz-Ansatz wird auf die Baugruppe der Funktionsbausteine für modulare Satelliten zurückgegriffen, die bereits beim genetischen Algorithmus für Experimente genutzt wurde. Im Rahmen der Experimente sollte sowohl evaluiert werden wie es um die Generalisierungsfähigkeit des Netzes bestellt ist, als auch wie es sich in Hinblick auf die Laufzeit gegenüber dem genetischen Algorithmus verhält.

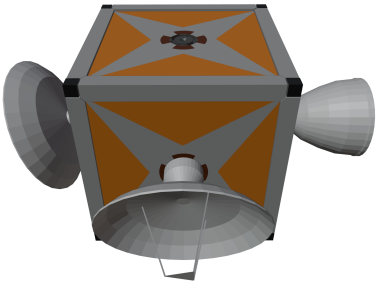
7.2.1. Datensatz und Evaluationsumgebung

Der im Rahmen der Evaluation des genetischen Algorithmus betrachteten Datensatz ist mit nur 20 Varianten nicht direkt als Trainings- bzw. Testdatensatz verwendbar, da der Umfang auch für einfache neuronale Netze zu gering ist. Aus diesem Grund muss der Datensatz zunächst erweitert werden. Das Ziel der Erweiterung ist, den Datensatz auf 2000 Varianten zu erweitern. Die bisherigen Funktionsbausteine sind händisch konstruiert worden. Ein solches Vorgehen ist allerdings für die angestrebte Zielmenge in der Umsetzung nicht realisierbar. Daher werden die fehlenden Funktionsbausteine künstlich generiert.

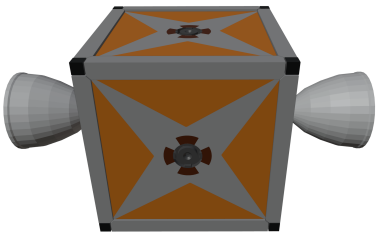
Für die Generierung wird zunächst ein entsprechendes Generierungswerkzeug in Blender implementiert. Das Werkzeug kann hierfür auf die zuvor konstruierten Bauteile der Funktionsbausteine zugreifen. Da das betrachtete Mockup des



(a) Eine Variante eines Funktionsbausteins mit zwei Schubdüsen und mehreren Antennen.



(b) Eine Variante eines Funktionsbausteins mit Schubdüse und Parabolantennen.



(c) Eine Variante eines Funktionsbausteins mit zwei Schubdüsen.

Abbildung 7.5.: Verschiedene durch den Generierungsansatz erzeugte künstliche Varianten eines Funktionsbausteins.

Satelliten ausschließlich aus einer Kombination der konstruierten Bauteile besteht, ist es möglich mit einfach logischen Regeln die Generierung durchzuführen. Die logischen Regeln stellen sicher, dass alle notwendigen Bauteile: Stützstruktur, Gehäuse und on-board Computer verwendet werden und eine zulässige Kombination von optionalen Bauteilen zur Anwendung kommt. Bei der Konzipierung des Mockups wurde darauf geachtet, dass alle gültigen Bauteilkombinationen auch zu gültigen Montagesequenzen führen. Dementsprechend sind die implementierten logischen Regeln ausreichend, um verwendbare Varianten zu erzeugen. Abbildung 7.5 zeigt mehrere auf diese Weise erzeugten Varianten.

Nach erfolgreicher Generierung der Varianten wird der genetische Algorithmus verwendet, um für die Varianten Montagesequenzen zu erzeugen. Auch wenn bei Nutzung des Sequenz-zu-Sequenz-Ansatzes primär die Generierung von gültigen Sequenzen im Fokus steht, so ist es doch das Ziel, die KI auf möglichst optimalen Trainingsdaten zu trainieren. Unter Berücksichtigung der Annahme, dass bei mehreren ähnlichen Varianten einer Baugruppe gewisse zu bevorzugende Teilsequenzen existieren, sollte die KI in der Lage sein diese implizit mitzulernen und anzuwenden. Aus diesem Grund wird nicht die erste gültige Lösung, die für eine Baugruppe gefunden wird in den Trainingsdatensatz aufgenommen, sondern das Ergebnis der Optimierung durch den genetischen Algorithmus.

Die in der späteren Inferenz generierte Montagesequenz soll abschließend auf dem Demonstrator, - dargestellt in Abbildung 7.6 -, ist von einem Universal Robot UR10e montiert werden. Voraussetzung dafür ist, dass die Roboterbewegungen, die bei der Montage ausgeführt werden, keine Kollisionen mit den bereits gefügten Bauteilen hervorrufen und für die Kinematik des Roboterarmes realisierbar sind. Der verwendete genetische Algorithmus prüft zwar die Kollisionsfreiheit, betrachtet hierfür aber nur die Bauteile selbst, nicht aber den Greifer oder Roboterarm. Deshalb sind nicht alle generierten Sequenzen möglich und geeignet für das Training der KI. Um diese Problematik zu adressieren, werden alle

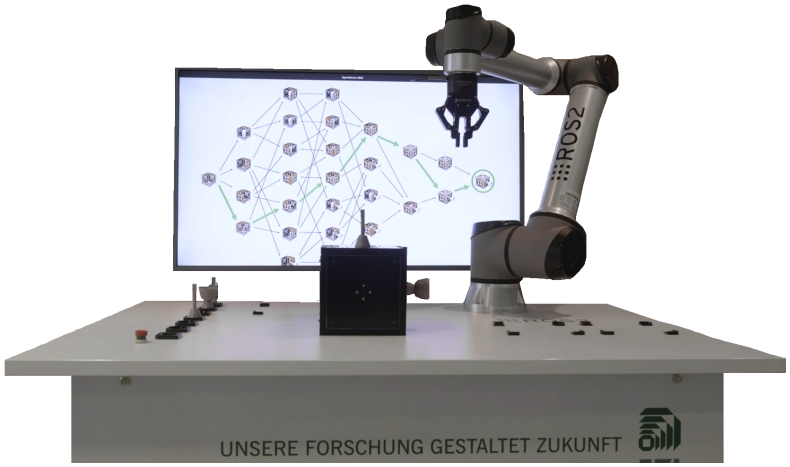


Abbildung 7.6.: Demonstratoraufbau für die Montage der Mockups der Funktionsbausteine.

durch den genetische Algorithmus generierten Sequenzen auf ihre Verwendbarkeit im Demonstrator geprüft. Hierzu wird die Software MovieIt2 in Verbindung mit ROS2 verwendet. Bei MoveIt2 handelt es sich um eine Planungssoftware für Robotertrajektorien, die in einem gegebenen Umgebungsmodel kollisionsfreie Robotertrajektorien generieren kann. Für jeden in einer Sequenz enthaltenen Teilschritt wird entsprechend geprüft, ob eine kollisionsfreie Trajektorie für den Roboter von einem Aufnahmepunkt der Bauteile neben der Baugruppe bis zum entsprechenden Montagepunkt für das Bauteil in der Baugruppe gefunden werden kann. Ist das der Fall für alle Teilschritte, wird die Sequenz als geeignet für den Demonstrator gewertet und in den Datensatz übernommen. Andernfalls wird die Sequenz verworfen.

Durch diese Prüfung wurden 556 der 2000 Sequenzen als nicht möglich bewertet und entfernt. Die verbleibenden 1444 Varianten einschließlich ihrer dazugehörigen Montagesequenzen stellen jedoch einen ausreichend großen Datensatz dar; es müssen keine zusätzlichen Varianten erzeugt werden müssen.

Bevor der so erzeugte Datensatz für das Training der KI verwendet werden kann, müssen die Modelle der Varianten in ein für das neuronale Netz verarbeitbares Format überführt werden. Dieses ist im Falle vom umgesetzten Sequenz-zu-Sequenz Netz eine Darstellung in einer Voxel-Karte (engl. Voxel-Map). Sie hat die Dimension $64 \times 64 \times 64 \times n + 1$. Hierbei entspricht n der Anzahl der Bauteile und die zusätzliche Tiefe $+1$ ist notwendig, da auch die Gesamtbaugruppe in der Voxel-Karte eingetragen wird. Für die Voxelisierung können existierende Werkzeuge genutzt werden. Jedes Bauteil wird in seiner Endlage einzeln voxelisiert als Darstellung in einer $64 \times 64 \times 64$ Voxel-Karte ausgegeben und in der nächsten freien $n + 1$ Dimension der Gesamtvoxel-Karte gespeichert.

Nach erfolgreicher Konvertierung jeder Baugruppe in eine Voxel-Karte wird die Aufteilung in den Training-, Validation- und Testdatensatz durchgeführt. Die initiale Aufteilung hierfür war 60-20-20. Während der Evaluation zeigte sich jedoch, dass diese Aufteilung nicht die optimalen Ergebnisse erzielen konnte. Sie wurden schließlich erreicht mit einer Aufteilung von ca. 70-25-5 erreicht. Konkret bedeutet dieses für die finale Aufteilung:

- Trainingsdatensatz: 1010 Baugruppen,
- Validierungsdatensatz: 361 Baugruppen,
- Testdatensatz: 73 Baugruppen.

Für die Experimente verwendet wurde ein Rechnersystem mit einem Intel Core i9-9900K Prozessor, 64 GB Arbeitsspeicher und einer Nvidia GeForce RTX 4090 mit 24GB VRAM.

7.2.2. Durchgeführte Experimente und Ergebnisse

Im Rahmen der Evaluation erfolgt zunächst das Training der KI. Das zugrundeliegende Konzept sowie die Architektur wurden bereits in Kapitel 5.2 vorgestellt. Für das Training wird die Architektur, bestehend aus Autoencoder und Sequenz-zu-Sequenz-Modell, entsprechend konfiguriert.

Der Autoencoder ist für die Komprimierung und Rekonstruktion der Eingangsdaten zuständig und wird zunächst parametrisiert. Der Encoder, konzipiert als Faltungsnetz, besteht aus fünf Schichten mit je 32 Filtern. Die extrahierten wesentlichen Merkmale eines Objektes werden in einen niedrigdimensionalen Latenter Raum / Latent Space der Dimension 128 projiziert. Der anschließende Decoder ist als mehrschichtiges Perzeptron mit sechs Schichten implementiert, die jeweils 128 Neuronen umfassen. Der Sequenz-zu-Sequenz-Teil ist für die serielle Anordnung der Bauteile verantwortlich und nutzt intern ein rekurrentes neuronales Netz mit einer Hidden-State-Dimension von 256.

Das Training des Modells erstreckt sich über 500 Epochen. In jeder Epoche verarbeitet das Netzwerk den gesamten Datensatz einmal, aufgeteilt in Batches. Die Batch-Größe beträgt 40, sodass die Gewichte nach der Fehlerberechnung von jeweils 40 Beispielen aktualisiert werden. Die Lerngeschwindigkeit wird durch die Lernrate bestimmt, welche mit $5 \cdot 10^{-4}$ initialisiert ist. Um gegen Ende des Trainings eine stabile Konvergenz zu gewährleisten, kommt ein Lernraten-Scheduler zum Einsatz. Dieser verringert die Lernrate exponentiell mit einem Faktor von $0,999$ alle 300 Schritte. Für das Training des Sequenz-zu-Sequenz-Modells wird die Technik des *Teacher Forcing* angewandt. Dabei erhält das Netz während des Trainings teilweise die Ground-Truth-Ausgabe als Eingabe anstatt der eigenen Vorhersage. Der Einfluss des Teacher Forcing wird über die Zeit schrittweise mit einem Faktor von $0,999$ reduziert.

Als Verlustfunktion / Loss Function für das Stopp-Signal dient die binäre Kreuzentropie, die mit einer Gewichtung von $0,01$ in die Gesamtfehlerfunktion eingeht.

Nach erfolgreichem Training erfolgt die Evaluation des Netzes. Hierfür wird die Anzahl der korrekten Sequenzvorhersagen für die verschiedenen Baugruppen analysiert. Dazu werden die Vorhersagen des Modells mittels der zuvor beschriebenen Simulation auf ihre Ausführbarkeit am Demonstrator geprüft. Es findet

Datensatz	Datenanzahl	Korrekt generierte Sequenzen	Inkorrekt generierte Sequenzen	Quote
Training	1010	1008	2	99,8 %
Test	361	360	1	99,7 %
Evaluation	73	73	0	100 %

Tabelle 7.2.: Ergebnisse der Vorhersage des Sequenz-zu-Sequenz-basierten Sequenzgenerators.

	KI	Genetischer Algorithmus
Durchschnitt	0.0184 s	16.315 s
Schnellster Durchlauf	0.014 s	11.081 s
Längster Durchlauf	0.034 s	29.711 s

Tabelle 7.3.: Vergleich zwischen Sequenz-zu-Sequenz-Ansatz und genetischem Algorithmus bezüglich der Laufzeit bis zur Erstellung einer ersten gültigen Montagesequenz.

somit nicht nur eine Kontrolle hinsichtlich der physikalisch möglichen Montagesequenzen statt, sondern auch bezüglich der tatsächlichen Ausführbarkeit. Die Ergebnisse sind Tabelle 7.2 zu entnehmen. Insgesamt generiert das Netzwerk sowohl für den Trainings- als auch für den Test- und Evaluationsdatensatz in nahezu 100 % der Fälle korrekte Montagesequenzen. Bei genauerer Untersuchung der wenigen Fehlerfälle zeigt sich, dass im Trainingsdatensatz eine Sequenz zwar physikalisch möglich, jedoch für den Roboter nicht ausführbar war. Bei den beiden anderen Fehlerfällen konnte keine physikalische Korrektheit festgestellt werden.

Im zweiten Teil der Evaluation wird die Inferenzzeit des Modells mit der Laufzeit des genetischen Algorithmus verglichen (siehe Tabelle 7.3). Anzumerken ist, dass beim genetischen Algorithmus lediglich die Zeit bis zur Generierung der ersten validen Lösung (Initialisierung der ersten Generation) betrachtet wird. Eine Berücksichtigung des gesamten Optimierungsprozesses würde zu deutlich längeren Laufzeiten führen. Die Generierung durch das KI-Modell ist in allen drei betrachteten Metriken um den Faktor ca. 800 schneller. Nicht in den Zeiten enthalten sind die notwendige Datenvorverarbeitung sowie die Zeiten für das Laden des Modells und den Datentransfer.

7.2.3. Diskussion und Einordnung der Ergebnisse

Die Evaluation des Sequenz-zu-Sequenz-basierten Modells unterstreicht das Potenzial dieses Ansatzes. Trotz der begrenzten Datengrundlage ist die Vorhersagegüte hoch, und die geringe Laufzeit ermöglicht den Einsatz in automatisierten Montagesystemen. Kritisch zu betrachten sind jedoch die erforderliche Vorverarbeitung und deren Konsequenzen. Die geringe Auflösung der Voxel-Karte führt zu einem signifikanten Detailverlust sowohl bei den einzelnen Bauteilen als auch bei der Gesamtbaugruppe. Dieser Informationsverlust ist bereits bei den untersuchten Baugruppen, in denen alle Komponenten ähnliche Dimensionen aufweisen, erheblich (vgl. Abbildung 7.7). Bei Baugruppen mit stärkeren Größenunterschieden hat dies jedoch gravierende Auswirkungen: Werden beispielsweise große Objekte wie ein Fahrzeug betrachtet und auf das $64 \times 64 \times 64$-Voxel-Raster skaliert, sind kleine Komponenten (wie eine M5-Schraube) bei gleichem Skalierungsfaktor nicht mehr darstellbar. Dementsprechend beschränkt sich die Anwendbarkeit dieses Ansatzes auf Szenarien mit homogenen Bauteilgrößen.

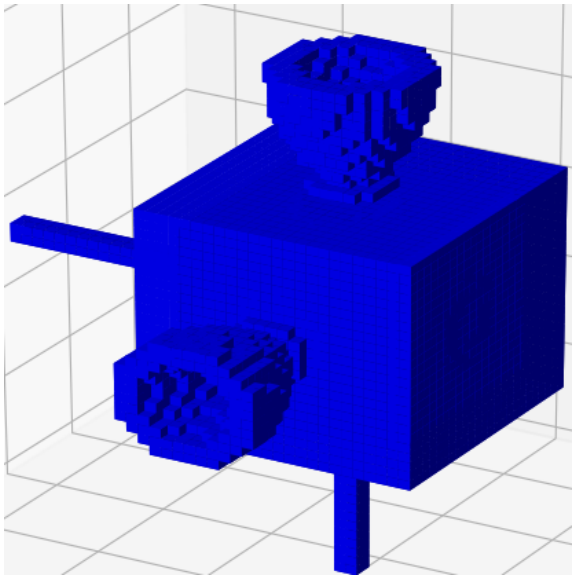


Abbildung 7.7.: Beispiel eines voxelisierten Funktionsbausteins.

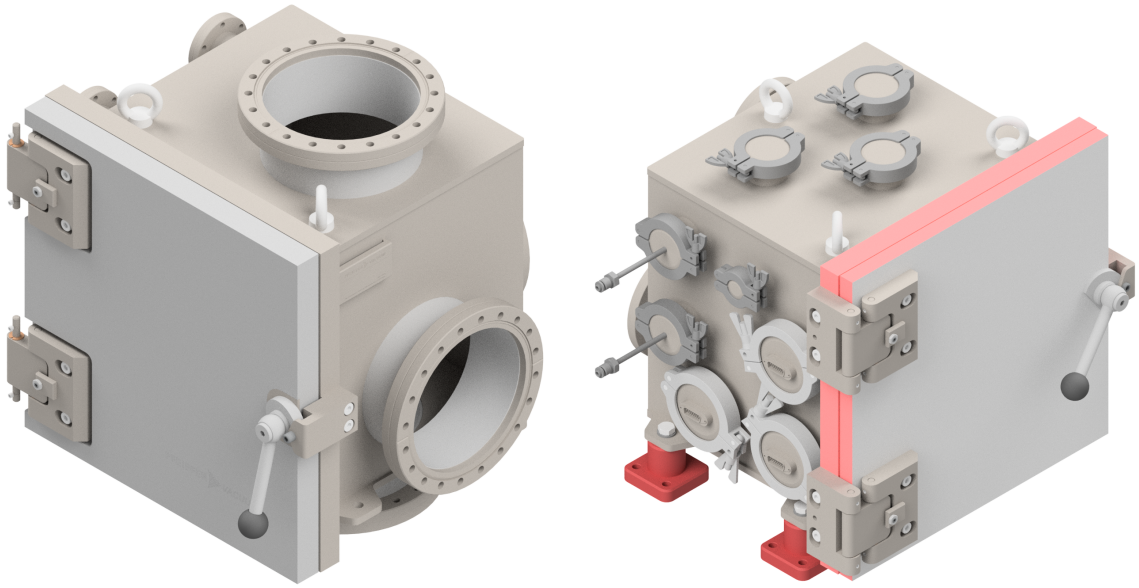
7.3. LLM-basierte Montagesequenzgenerierung

Die Evaluation der LLM-basierten Montagesequenzgenerierung gliedert sich in mehrere Schritte. Zunächst wurden zwei unterschiedliche Datensätze erstellt: der eine bestehend aus verschiedenen Varianten einer Vakuumkammer, der andere aus diversen Flugzeugfahrwerken. Ziel der ersten Experimente war es, die generelle Machbarkeit der automatisierten Montageplanung mittels Large Language Models und Vision-Language Models nachzuweisen. Anschließend lag der Fokus darauf, die Leistungsgrenzen der aktuellen Modelle, den Einfluss der Datenqualität sowie die Effizienz verschiedener Feedback-Mechanismen quantitativ und qualitativ zu bewerten.

7.3.1. Datensatz und Evaluationsumgebung

Um die Generalisierungsfähigkeit und Robustheit des Systems zu untersuchen, wurden zwei Datensätze mit grundlegend unterschiedlichen Charakteristika ausgewählt.

Der Vakuumkammer-Datensatz umfasst 70 verschiedene Produktvarianten, die auf zwei standardisierten Hochvakuumkammer-Ausführungen basieren. Bei einer Hochvakuumkammer handelt es sich um einen meist kubischen Hohlkörper, der über mehrere Zu- und Abgänge, sogenannte Ports, verfügt. An diesen Ports werden Komponenten wie Sensoren oder Vakuumpumpen angeschlossen. Neben den Ports besitzt eine Kammer in der Regel eine Tür sowie bei Bedarf ein oder mehrere Sichtfenster. Um das erforderliche Vakuum zu gewährleisten, werden alle dauerhaften Verbindungen – beispielsweise die Kammerwände oder



(a) Variante der Vakuumkammer mit großen Zugängen, ohne Standfüße.

(b) Variante der Vakuumkammer mit einer Vielzahl kleinerer Zugänge und Standfüßen.

Abbildung 7.8.: CAD-Renderings zweier Varianten der Vakuumkammer.

die Verbindungen zwischen Port und Kammerwand – zunächst geheftet und anschließend vakuumdicht verschweißt. Für alle temporären Verbindungen, etwa an der Tür, werden Dichtungen eingesetzt. Das gewählte Dichtmaterial ist abhängig vom späteren Einsatzszenario und reicht von Metalldichtungen bis hin zu Silikondichtungen. Bei den beschriebenen Vakuumkammern handelt es sich um reale Industrieprodukte. Ausgehend von den Basisvarianten mit kubischem Grundkörper werden die standardisierten Kammern gemäß spezifischer Kundenanforderungen modifiziert. Diese Anpassungen können die Wahl der Dichtungen, die Anzahl und Position der Ports sowie den Einbau zusätzlicher Sichtfenster oder Sonderkomponenten umfassen. Die Grundkörper aller für die Experimente verwendeten Vakuumkammern weisen je nach Ausgangsvariante eine Kantenlänge von $300\,\mathrm{mm}$ oder $500\,\mathrm{mm}$ auf. Abhängig von den exakten Anforderungen ergeben sich Baugruppen mit einer Bauteilanzahl zwischen 15 und 40 Einzelteilen, die vielfache Abhängigkeiten untereinander aufweisen. Insgesamt liegen 70 verschiedene Kammervarianten vor, von denen exemplarische Ausführungen in Abbildung 7.8 dargestellt sind.

Für alle betrachteten Varianten liegen die Geometriedaten als STEP-Dateien vor. Diese sind hierarchisch in Unterbaugruppen gegliedert, wobei zwei Hierarchiestufen unterschieden werden:

- Stufe 1: Dies ist die oberste Montageebene. Alle Bauteile und Unterbaugruppen auf dieser Stufe fügen sich zur vollständigen Baugruppe zusammen. Auf dieser Ebene finden in der Regel keine Schweißprozesse mehr

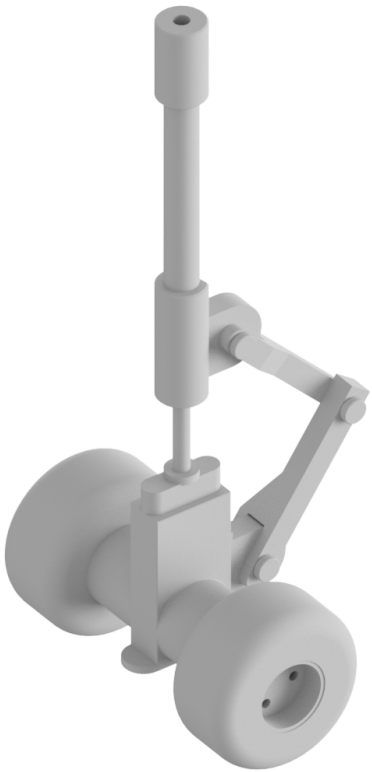
statt.

- Stufe 2: Auf dieser Ebene entstehen die entsprechenden Unterbaugruppen. Beispiele hierfür sind das Schweißen des Grundkörpers oder das Anbringen von Typenschildern an Kammerwänden. Auf dieser Ebene finden primär die Schweißprozesse statt.

Theoretisch wären weitere Hierarchiestufen denkbar, in den vorliegenden Kammern sind jedoch nur die ersten beiden Stufen relevant.

Da das LLM im zweiten Inferenzschritt auch eine Manipulationssequenz für die Baugruppe generiert, ist für die Evaluation eine möglichst breite Variation der notwendigen Fügeoperationen essenziell. In dieser Hinsicht bietet der Vakuumkammer-Datensatz eine Vielzahl an erforderlichen Fügetechniken: Um eine Kammer vollständig zu montieren, sind die Aktionen Schweißen, Schrauben, Einpressen und Einsetzen notwendig.

Als zweiter Datensatz wurde, basierend auf den von Lupinetti et al. [179] veröffentlichten Baugruppen, ein Datensatz von Flugzeugfahrwerken erstellt. Dieser umfasst sechs Varianten, die sich im Gegensatz zum Vakuumkammer-Datensatz stark voneinander unterscheiden. Beispiele für die untersuchten Fahrwerke sind



(a) Eine konstruktiv einfache Variante des Flugzeugfahrwerks.

(b) Eine Variante des Fahrwerks mit komplexer Federung.

Abbildung 7.9.: CAD-Renderings zweier Varianten des Flugzeugfahrwerks.

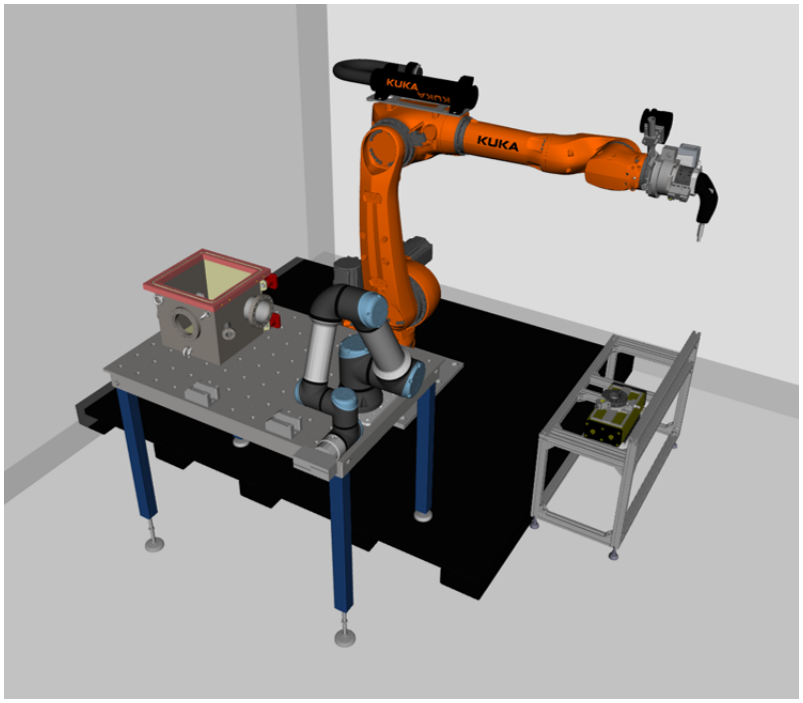


Abbildung 7.10.: Darstellung der Automatisierungszelle mit zwei Robotern. Diese Zelle wurde im Projekt GANResilRob aufgebaut und für die Montage der Hochvakuumkammern verwendet [180].

in Abbildung 7.9 zu sehen. Die Bauteilanzahl in diesen Baugruppen variiert zwischen 5 und 26. Auch hier existieren diverse Abhängigkeiten der Bauteile untereinander, wodurch die Anzahl der gültigen Montagesequenzen limitiert ist. Im Vergleich zu den Vakuumkammern sind nicht alle Baugruppen vollständig ausmodelliert; in einigen Fällen fehlen explizite Verbindungselemente wie Schrauben und Bolzen, deren Vorhandensein vom Modell implizit angenommen werden muss. Im Gegensatz zur Vakuumkammer existieren in diesem Datensatz keine Gleichteile. Der Aufbau der Fahrwerke besteht typischerweise aus einer Radbasis, einem Dämpfungssystem und Stabilisierungselementen.

Für die Erstellung der Manipulationssequenz berücksichtigt das System eine Roboterzelle mit zwei Robotern: einem großen Industrieroboter für das Handling schwerer Bauteile und Werkzeuge sowie einem Serviceroboter, der primär Andienungsaufgaben übernimmt. Ein Beispiel dieses Aufbaus ist in Abbildung 7.10 dargestellt.

Als Hardwareplattform für das LLM und VLM dient ein GPU-Cluster mit einer NVIDIA A100 Tensor-Core-GPU (80 GB VRAM) und 512 GB Arbeitsspeicher. Diese Ausstattung ermöglicht es, die Modelle im Arbeitsspeicher vorzuhalten und innerhalb kürzester Zeit zu laden. Da es sich bei dem GPU-Cluster um eine geteilte Ressource handelt, wird der Zugriff auf die GPU zwischen mehreren Nutzern synchronisiert. Dies kann zu Wartezeiten (Job Queuing) führen, was bei der Betrachtung der Gesamtlaufzeit des Ansatzes zu berücksichtigen ist.

	Initial	Validierung	Grounding
Korrekte Montagesequenzen (in %)			
Hierarchiestufe 1	8,00	17,00	17,00
Hierarchiestufe 2	91,00	100,00	91,00
Korrekte Montageschritte (in %)			
Hierarchiestufe 1	88,96	92,08	91,00
Hierarchiestufe 2	95,00	100,00	99,00

Tabelle 7.4.: Erfolgsrate der Montagesequenzgenerierung unterteilt in Hierarchiestufe 1 und 2.

7.3.2. Durchgeführte Experimente und Ergebnisse

Für die ersten Experimente wird zusätzlich zu den beiden bereits erläuterten Datensätzen für das LLM das QwQ-Modell genutzt. Die verwendete Variante des QwQ-Modells besitzt 32 Milliarden Parameter und gehört zur Klasse der Chain-of-Thought-Modelle (CoT). Für das VLM wird das Qwen2.5-VL-Modell eingesetzt.

In einem ersten Experiment wird mit einer nur minimal spezifizierten Prompt-Vorlage untersucht, wie hoch die Out-of-the-box-Performanz des Ansatzes ist. Bei diesen Experimenten werden für die Generierung der Montagesequenz sowohl der Validierungsalgorithmus als auch die Grounding-Komponente verwendet. Als erstes Proof-of-Concept-Experiment werden ausschließlich 12 Varianten der Vakuumkammer untersucht, die verhältnismäßig wenige Änderungen im Vergleich zu den Standardkammern aufweisen. Die Beurteilung, ob eine für diese Varianten erzeugte Sequenz korrekt ist, erfolgt manuell. Die Ergebnisse dieser ersten Testreihe sind in Tabelle 7.4 dargestellt.

Betrachtet man ausschließlich die Erfolgsquote der Generierung vollständig korrekter Sequenzen, so sind die Ergebnisse stark verbesserungswürdig. Nach der initialen Generierung waren lediglich 8 % der Montagesequenzen korrekt, und auch durch die Integration von Feedback aus dem Validierungsalgorithmus sowie der Grounding-Komponente konnte dieser Wert nur auf 17 % gesteigert werden. Eine detaillierte Analyse der generierten Sequenzen zeigt jedoch, dass über 90 % der einzelnen Montageschritte pro Sequenz korrekt sind. Um eine fehlerhafte Sequenz in eine valide Sequenz zu überführen, sind im Durchschnitt weniger als vier Änderungen notwendig.

Eine genauere Analyse der auftretenden Fehler (vgl. Abbildung 7.11) zeigt zum einen, dass fast alle Fehler im Zusammenhang mit der Hierarchiestufe 1 stehen, und zum anderen, dass in über 50 % der Fehlerfälle das problematische Bauteil die Türdichtung ist. Für die Unterbaugruppen bedeutet dies, dass in nahezu allen Fällen eine korrekte Montagesequenz gefunden werden kann. Bei der Türdichtung ergibt sich reproduzierbar die Problematik, dass das LLM die Dichtung erst

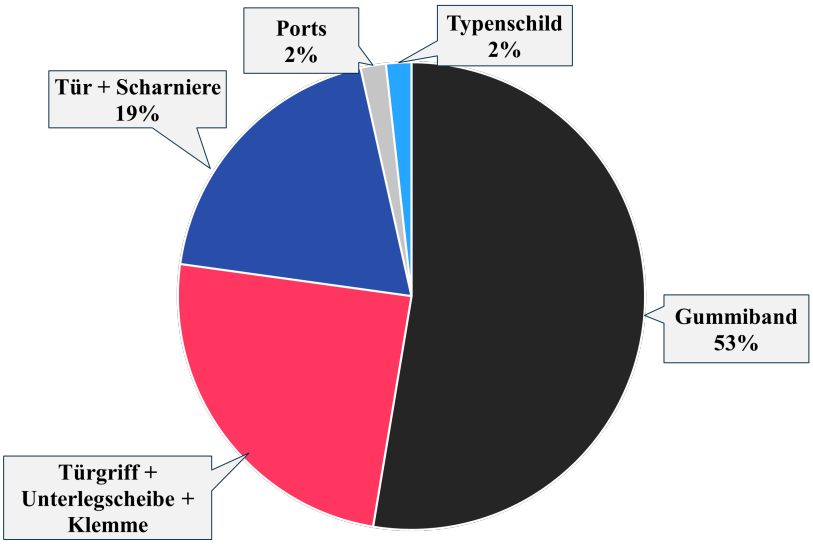


Abbildung 7.11.: Verteilung der aufgetretenen Fehlerarten während der Montagesequenzgenerierung.

nach der Tür montieren will. Im konkreten Fall der Vakuumkammer ist dies technisch nicht möglich, da die Türdichtung während der Montage als Abstandshalter zwischen Grundkörper und Tür dient. Dieser Sachverhalt spiegelt sich zwar im Wissensgraphen wider, wird jedoch vom LLM ignoriert.

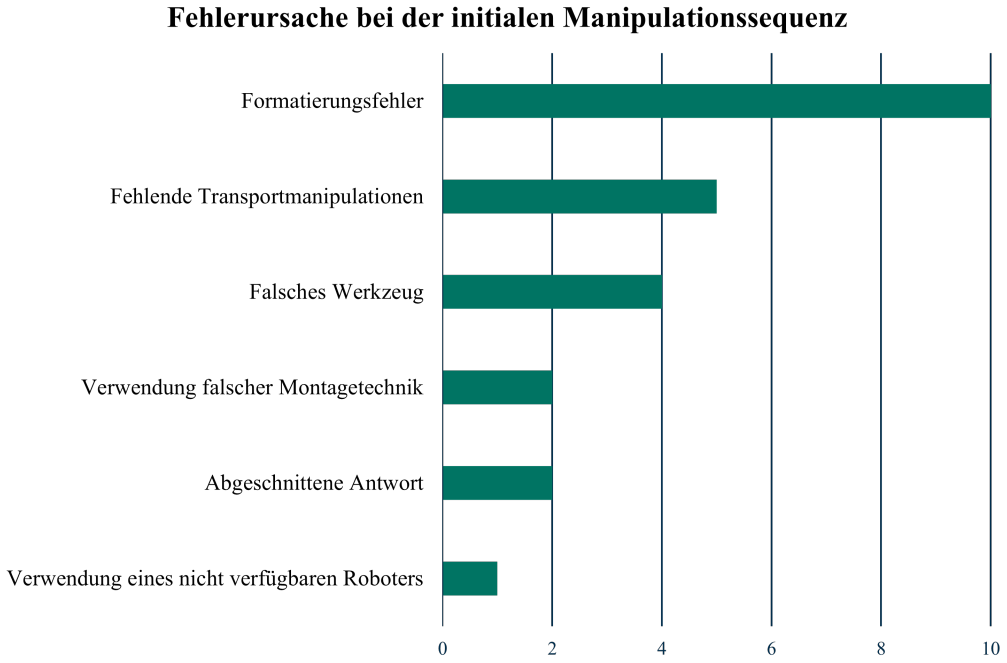


Abbildung 7.12.: Verteilung der aufgetretenen Fehlerarten bei der Generierung der Manipulationssequenz.

Vergleicht man den Einfluss der Validierungs- und der Grounding-Komponente, wird deutlich, dass der Validierungsalgorithmus zu Ergebnisverbesserungen führt, während die Grounding-Komponente eine Stagnation der Performanz bewirkt oder diese in einigen Fällen sogar verschlechtert. Eine differenzierte Untersuchung dieses Sachverhalts verdeutlicht, dass die Interpretation der Renderings durch das LLM häufig Fehler wiedereinführt, die zuvor durch das Feedback des Validierungsalgorithmus eliminiert wurden.

Für die Generierung einer Manipulationssequenz ergibt sich ein leicht abweichendes Bild. Werden die auftretenden Fehler im Kontext der Manipulationssequenz evaluiert (vgl. Abbildung 7.12), zeigt sich, dass am häufigsten Formatierungsprobleme vorliegen. Dies tritt in 10 von 24 ausgewerteten Sequenzen auf. Darüber hinaus traten verhältnismäßig häufig zwei weitere Probleme auf: Zum einen werden Bauteile nicht zunächst aus dem Materiallager entnommen, sondern direkt gefügt. Zum anderen wird, obwohl die korrekte Fügeoperation bestimmt wurde, ein falsches Werkzeug für den Fügeschritt ausgewählt. Die Ursache für Formatierungsprobleme liegt vorwiegend im Prompt-Template, während der Ursprung der beiden anderen Probleme primär in einer unzureichenden Validierungskomponente zu suchen ist.

Basierend auf den Evaluationsergebnissen der ersten Experimente wurden Optimierungen in drei Kernbereichen des Frameworks vorgenommen. So wurde die Prompt-Vorlage angepasst und die darin enthaltenen Anweisungen zur Formatierung präzisiert. Ebenso wurde im Wissensgraphen sichergestellt, dass jeder Knoten eine aussagekräftige Beschreibung enthält, mit dem Ziel, Fehlannahmen durch das LLM - etwa bei der Montage der Türdichtung - zu minimieren. Abschließend fanden Anpassungen in der Validierungskomponente für die Manipulationssequenz statt, indem Logikabfragen für die zuvor identifizierten Fehlerklassen eingeführt wurden.

Zur Überprüfung, ob die Änderungen die Qualität des Frameworks messbar steigern, wird eine zweite Experimentreihe durchgeführt. Neben den bereits getesteten Baugruppen werden nun auch fünf Baugruppen höherer Komplexität untersucht. Diese Vakuumkammern weisen eine deutlich größere Anzahl baugruppenspezifischer Bauteile auf. Die Ergebnisse der erneuten Experimente sind in

	Initial	**Validierung**	**Grounding**
Korrekte Montagesequenzen (in %)			
Hierarchiestufe 1	12,50	62,50	37,50
Hierarchiestufe 2	71,00	66,67	71,43
Korrekte Montageschritte (in %)			
Hierarchiestufe 1	90,89	96,92	91,52
Hierarchiestufe 2	87,62	77,78	91,52

Tabelle 7.5.: Erfolgsrate der Montagesequenzgenerierung nach den Anpassungen unterteilt in Hierarchiestufe 1 und 2.

	Initial	Validierung	Grounding
Korrekte Montagesequenzen (in %)			
Hierarchiestufe 1	0,00	80,00	60,00
Hierarchiestufe 2	75,00	66,67	75,00
Korrekte Montageschritte (in %)			
Hierarchiestufe 1	65,67	93,55	91,95
Hierarchiestufe 2	88,94	96,97	91,95

Tabelle 7.6.: Erfolgsrate der Montagesequenzgenerierung für die komplexeren Baugruppen unterteilt in Hierarchiestufe 1 und 2.

Tabelle 7.5 für die ursprünglichen Baugruppen und in Tabelle 7.6 für die zusätzlich hinzugefügten Varianten dargestellt.

Die durchgeführten Anpassungen führten zu einer deutlichen Steigerung der Anzahl korrekter vollständiger Montagesequenzen. Auffällig ist hierbei, dass sich der Anteil korrekter einzelner Montageschritte auf fast 97 % erhöht, sich jedoch speziell nach dem initialen Durchlauf des LLMs eine Verschlechterung der anteiligen korrekten Montageschritte für die Hierarchiestufe 2 ergibt. Dies konnte auf die Auswirkungen einer einzelnen Baugruppe eingegrenzt werden und wird daher als Anomalie betrachtet.

Vergleicht man die komplexeren Baugruppen mit den einfacheren, so fällt auf, dass die Sequenzgenerierung für die komplexen Kammern leicht bessere Ergebnisse liefert (80 % vollständig korrekte Montagesequenzen gegenüber 62 %). Eine mögliche Erklärung liegt in der geringeren Anzahl an Abhängigkeiten der zusätzlichen Bauteile in den komplexen Baugruppen. Bei diesen handelt es sich oft um Komponenten, die lediglich Kontakt zu einem einzigen anderen Bauteil oder einer Unterbaugruppe haben.

Auch die Laufzeit des Frameworks war Gegenstand der Untersuchung. In diesem Kontext ist die Laufzeit definiert als Zeitspanne zwischen dem Start des ersten Inferenzdurchlaufs und der finalen Ausgabe der Manipulationssequenz. Sie liegt im Mittel über alle getesteten Baugruppen bei 203 Minuten.

In beiden Experimentreihen ist auffällig, dass die Grounding-Komponente zu keiner Verbesserung der Generierung führt und bei den komplexeren Baugruppen tendenziell sogar eine Verschlechterung hervorruft. Daher wird in einem weiteren Evaluationsschritt geprüft, ob sich die Resultate bei Deaktivierung der Grounding-Komponente und gleichzeitiger wiederholter Ausführung (bis zu 10-mal) der Validierungskomponente verändern. Die Ergebnisse für diese Testreihe sind in Abbildung 7.13 dargestellt.

Es ist zu erkennen, dass eine ausschließliche Nutzung der Validierungskomponente zu den besten Gesamtergebnissen führt. Alle betrachteten Standardfälle und drei der fünf komplexeren Vakuumkammern konnten bis zu einer vollständig fehlerfreien Montagesequenz iteriert werden. Auch die Anzahl der antei-

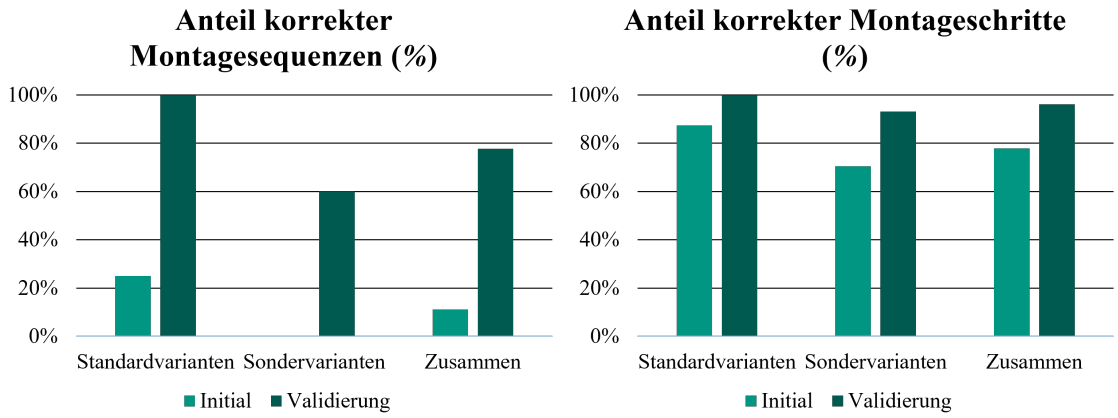


Abbildung 7.13.: Vergleich der Erfolgsrate mit und ohne Einbindung des VLMs als Grounding-Komponente.

lig korrekten Montageschritte konnte gesteigert werden; hier sind durchschnittlich 33 Schritte pro Sequenz korrekt. Weitere positive Effekte zeigen sich bei der Laufzeit des Frameworks: Durch den Wegfall von Modellwechseln, Rendering-Generierung und VLM-Inferenz konnte die Laufzeit auf im Mittel 117 Minuten reduziert werden.

Problematisch an einem Ansatz ohne zusätzliches Grounding kann jedoch sein, dass die Validierung zu False-Positives führt. In einem gesonderten Experiment wurde untersucht, welche Auswirkungen ein initial fehlerhafter Wissensgraph hat. Dazu wurde im Wissensgraphen einer der komplexen Baugruppen eine Kollisionskante entfernt. Dies führte im Ablauf zu dem Problem, dass eine für den Schritt korrekt ausgewählte Schraube durch den Validierungsansatz als falsch gewertet wurde, da laut Wissensgraph kein Kontakt zwischen der Schraube und dem entsprechenden Bauteil bestand. Bei späterer wiederholter Ausführung mit aktivierter Grounding-Komponente konnte dieses Problem durch das VLM erkannt und auf Basis des Feedbacks durch das LLM eine korrekte Montagesequenz erzeugt werden.

Hinsichtlich der praktischen Anwendbarkeit des Frameworks sind neben der Korrektheit auch Laufzeit und notwendige Rechenkapazität entscheidend. Unter diesen Aspekten wird in den folgenden Experimenten untersucht, ob der Wechsel auf ein Modell ohne Chain-of-Thought oder auf ein deutlich kleineres, lokal ausführbares CoT-Modell Vorteile bringt. Als Modell ohne Chain-of-Thought wird das Qwen3-Modell (32 Mrd. Parameter) untersucht, sowie das Modell DeepSeek-R1-Distill-Qwen. Bei letzterem handelt es sich um ein Chain-of-Thought-Modell mit 14 Mrd. Parametern, das auf einer NVIDIA GeForce RTX 4090 und somit einem lokalen Rechnersystem ausgeführt werden kann. Für die Untersuchung wurden die vorherigen Baugruppen erneut verwendet und die Durchläufe ohne VLM ausgeführt, sodass nur die LLM-Performanz betrachtet wird. Die Ergebnisse sind in Abbildung 7.14 dargestellt (alle Tests fanden zur Vergleichbarkeit weiterhin auf dem GPU-Cluster statt).

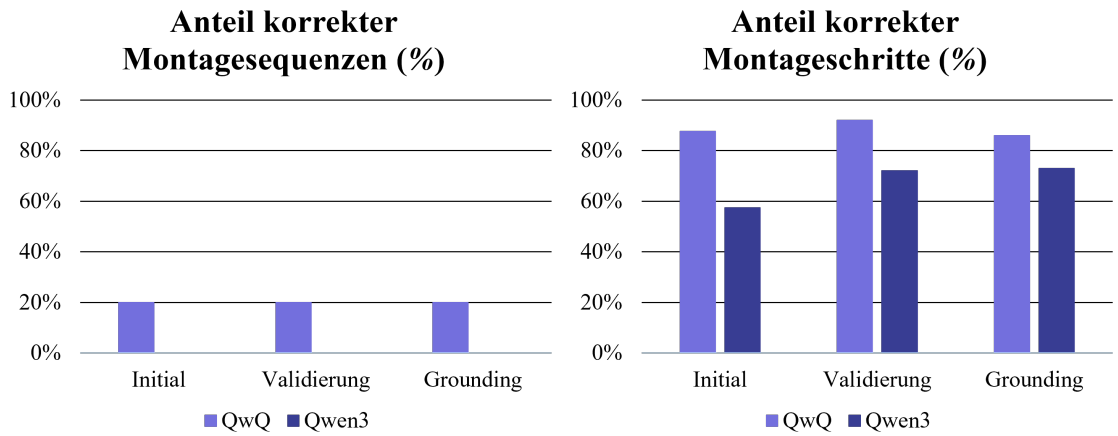


Abbildung 7.14.: Vergleich der Erfolgsrate zwischen dem QwQ-Modell und dem alternativen Qwen3-Modell.

Es ist deutlich erkennbar, dass beide Modelle keine adäquate Alternative darstellen. Das Qwen3-Modell konnte für die Standardkammern keine einzige vollständig korrekte Lösung generieren. Eine Analyse zeigt, dass das Modell Schwierigkeiten hat, das Validierungsfeedback umzusetzen; Fehler blieben über mehrere Generationen bestehen. Besonders bei komplexeren Baugruppen treten grundlegende Formatierungsfehler, primär bei den Knoten-IDs, auf. Die Laufzeit sinkt durch den Wegfall der Herleitungskette erwartungsgemäß auf 88 Minuten. Die Ergebnisse des DeepSeek-Modells sind noch problematischer. In keinem Fall konnten korrekte vollständige Sequenzen erzeugt werden, und der Anteil korrekter Montageschritte beträgt lediglich 4 %. Auffällig ist eine starke Tendenz zu repetitiven Schleifen: In 40 % der Fälle wiederholt das Modell den gleichen Montageschritt so lange, bis das Token-Limit erreicht ist, wodurch keine vollständige Ausgabe erzeugt wird. Folglich ist die Laufzeit mit im Mittel 190 Minuten sogar länger als beim QwQ-Modell.

Die finale Experimentreihe beschäftigte sich mit der Übertragbarkeit und Generalisierungsfähigkeit des Frameworks auf andere Baugruppen. Hierzu wurden die zuvor beschriebenen Varianten der Flugzeugfahrwerke verwendet. Die verwendeten Prompt-Vorlagen wurden nur minimal adaptiert, um die geänderte Aufgabenstellung (Montage eines Fahrwerks vs. Vakuumkammer) zu berücksichtigen. Entsprechend ist die einzige für diese Experimente vollständig geänderte Eingangsinformation der spezifische Wissensgraph. Die Ergebnisse dieser Testreihe sind in Tabelle 7.7 dargestellt.

Während die reine Betrachtung der Zahlen den Schluss nahelegt, dass der Ansatz ungeeignet ist (nur ca. 17 % vollständig korrekte Sequenzen), zeigt eine differenzierte Analyse, dass das Framework in der Lage ist, für alle betrachteten Baugruppen logisch korrekte Montagesequenzen (Bauteilreihenfolge) bereits nach dem initialen LLM-Durchlauf zu erzeugen. Dies lässt darauf schließen, dass sowohl die Extraktion des Wissensgraphen als auch die Prompt-Vorlage generali-

	Initial	Validierung	Grounding
Korrekte Montagesequenzen (in %)			
Hierarchiestufe 1	10,70	16,67	0,00
Hierarchiestufe 2	90.33	100,00	100,00
Korrekte Montageschritte (in %)			
Hierarchiestufe 1	77,23	80,25	91,00
Hierarchiestufe 2	95,00	100,00	100,00

Tabelle 7.7.: Erfolgsrate der Montagesequenzgenerierung für die Varianten des Flugzeugfahrwerks unterteilt in Hierarchiestufe 1 und 2.

sierbar sind. In der Validierungsphase konnten jedoch viele Fehler nicht im gleichen Umfang adressiert werden wie bei der Vakuumkammer. Insbesondere die Überprüfung auf verbundene Bauteile schlug regelmäßig fehl und führte zu fehlerhaftem Feedback. Auch die Formatierung der ausgegebenen Montagesequenz war in mehreren Fällen inkorrekt. Da jedoch die verwendeten Bauteil-IDs und die Reihenfolge korrekt waren, war es dennoch in allen Fällen möglich, korrekte Manipulationssequenzen zu definieren. Die Laufzeit des Frameworks verkürzte sich bei diesen Baugruppen auf durchschnittlich 121 Minuten, was primär auf die reduzierte Bauteilanzahl (im Mittel 12,5 Bauteile) zurückzuführen ist.

7.3.3. Diskussion und Einordnung der Ergebnisse

Wie die Evaluation zeigt ist das Framework in der Lage, selbst hochkomplexe Baugruppen zu bewältigen. Die Stärken liegen in der Nutzung des Wissensgraphen als effektives Grounding-Instrument, das Halluzinationen minimiert und in der robusten Generierung von Aktionssequenzen auch die Multi-Roboter-Szenarien korrekt handhabt.

Die größten Herausforderungen sind die lange Laufzeit des Frameworks sowie die Abhängigkeit von leistungsstarken Chain-of-Thought-Modellen. Die Validierungskomponente hat sich sich als die effektiveste Methode zur Qualitätsverbesserung erwiesen. Sie muss allerdings vorsichtig konfiguriert werden, um nicht zu spezifiziert für einen Datensatz zu sein. Das könnte die Transferierbarkeit auf andere Datensätze reduzieren. Problematisch ist zudem die Verwendung eines VLMs als Grounding-Komponente. Auch wenn in spezifischen Fällen Vorteile dadurch zu erreichen sind, ist die allgemeine Performanz ausbaufähig und ihr Einsatz mit einem hohen zusätzlichen Rechenaufwand verbunden.

Insgesamt muss limitierend berücksichtigt werden, dass aktuell in kurzen Abständen neue fähigere LLM und VLM Modelle veröffentlicht werden. Dementsprechend sind die in dieser Evaluation gewonnenen Erkenntnisse eher als Momentaufnahmen zu verstehen.

7.4. Affordanzendetektion

Die Evaluation des entwickelten Ansatzes zur kombinierten Affordanz-Erkennung und Objektlokalisierung im Montagekontext basiert auf einer systematischen Untersuchung der einzelnen Teilkomponenten sowie des Gesamtsystems. Zunächst wird die verwendete Datengrundlage und die technische Infrastruktur beschrieben, da sie die Rahmenbedingungen der Ergebnisse maßgeblich beeinflussen. Anschließend werden die spezifischen Experimente detailliert vorgestellt.

7.4.1. Datensatz und Evaluationsumgebung

Wie bereits im vorherigen Kapitel angesprochen, existieren für die spezifische Problemstellung der Erkennung von Montageaffordanzen bei industriellen Bauteilen keine öffentlich zugänglichen Datensätze. Aus diesem Grund wird im Rahmen der Evaluation ein eigener Datensatz erstellt. Dieser umfasst 175 RGB-D-Bilder, die mit einer Intel RealSense D435 Stereokamera aufgenommen wurden. Die Aufnahmen zeigen verschiedenste gängige Bauteile aus dem Maschinenbau, darunter Schrauben, Muttern, Unterlegscheiben, Profile und Platten. Die Entscheidung, keinen rein synthetischen Datensatz zu erzeugen, welcher z. B. ausschließlich CAD-Renderings verwendet, wurde bewusst getroffen. Leitidee hierbei ist, dass das natürliche Rauschen realer Kameradaten im Training abgebildet werden sollte, um die Diskrepanz zwischen Trainings- und späteren Anwendungsdaten zu minimieren. Die Bilder liegen mit einer Auflösung von 1920×1080 Pixeln vor, wobei die Tiefenbilder nachträglich auf die Farbbilder ausgerichtet wurden. Ein Beispiel für die im Trainingsdatensatz enthaltenen Bilder ist in Abbildung 7.15 zu sehen.

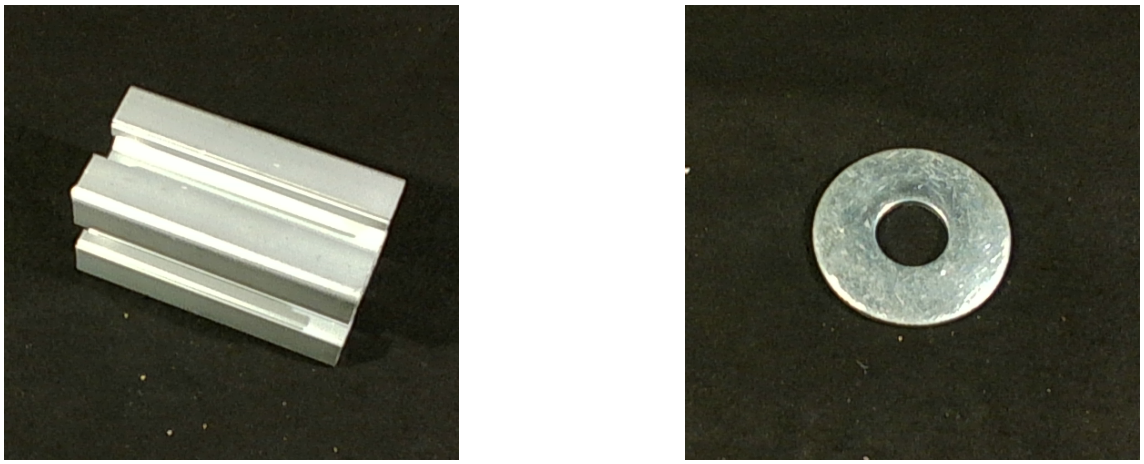


Abbildung 7.15.: Beispiele für zwei im Datensatz vorkommende Bauteile: ein Aluminiumprofil und eine Unterlegscheibe.

Da ein Datensatz mit 175 Bildern für das Training eines tiefen neuronalen Netzes unzureichend ist, werden Techniken zur Datenerweiterung (Data Augmentation) eingesetzt. Dies umfasst Verfahren wie Translation, Rotation und die Manipulation von Farbwerten (Helligkeit, Sättigung). Die Anwendung dieser Techniken ermöglicht es, den Datensatz künstlich auf 2000 Bilder zu vergrößern. Für die Evaluation wird der Datensatz vor der Datenerweiterung in Trainings-, Validierungs- und Testdaten aufgeteilt. Dabei werden acht Bilder als dedizierter Testdatensatz separiert; die verbleibenden 167 Aufnahmen werden im Verhältnis 80:20 für Training und Validierung verwendet. Erst im Anschluss wird die Datenerweiterung auf die einzelnen Teildatensätze angewendet.

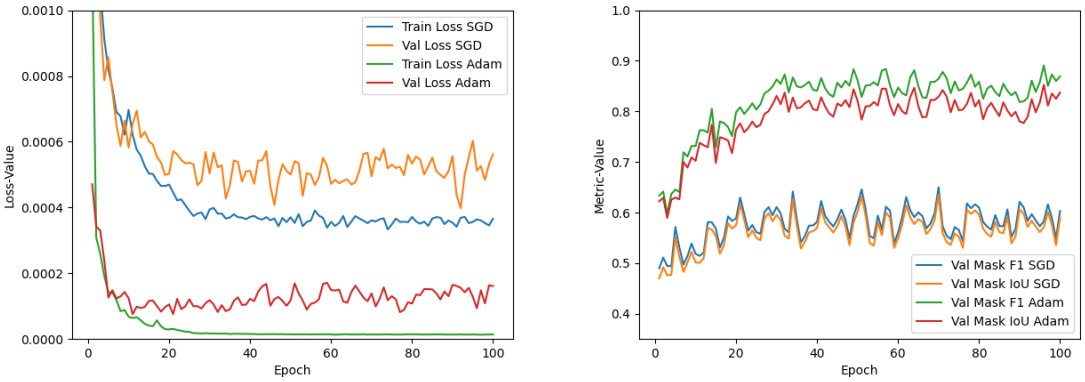
Die Durchführung der Experimente erfolgt auf einer Workstation mit einer Nvidia RTX 2080 Grafikkarte, die mit 11 GB Grafikspeicher ausgestattet ist. Aufgrund der Speicherlimitierung dieser Hardware müssen die Eingabebilder während des Trainings auf 512×512 Pixel herunterskaliert werden, was die Verwendung einer sehr kleinen Batch-Größe von zwei zur Folge hat.

7.4.2. Durchgeführte Experimente und Ergebnisse

Die Experimente konzentrieren sich zunächst auf die einzelnen Bestandteile des Netzes: die Objekterkennung und die Affordanz-Erkennung. Bei der Untersuchung der Objekterkennung steht die Optimierung des Region-Proposal-Netzwerks (RPN) und der nachgelagerten Klassifizierung im Fokus. Als Optimierer wird hier, analog zu vergleichbaren State-of-the-Art-Ansätzen wie Mask R-CNN, der stochastische Gradientenabstieg (Stochastic Gradient Descent, SGD) verwendet.

Zunächst wird untersucht, ob ein isoliertes Training des Region-Proposal-Netzwerks Vorteile gegenüber einem gemeinsamen Training mit dem Objekterkennungs-Zweig bietet. Die Ergebnisse zeigen, dass das isolierte Training im Schnitt um 2 % genauere Intersection-over-Union-Werte (IoU) für die Regionsvorschläge (Region Proposals) liefert; jedoch wird die Klassifizierungsgüte massiv durch Overfitting beeinträchtigt. Da eine präzise Klassifizierung der Regionsvorschläge essenziell für die nachfolgende Non-Maximum-Suppression ist, fiel die Entscheidung zugunsten eines gemeinsamen Trainings von RPN und Objekterkennung.

Ein kritischer Faktor bei der Objekterkennung ist die Größe der Ankerboxen. Es zeigt sich, dass große Objekte im Bild oft nicht präzise lokalisiert werden, da kleinere Bounding-Boxen (Begrenzungsrahmen), die anteilig mehr Objektfläche abdecken, vom Netzwerk bevorzugt werden. Experimente mit Ankergrößen von 200, 256 und 300 Pixeln machen deutlich, dass eine Seitenlänge von 256 Pixeln, bezogen auf die skalierte Eingabe, die besten Ergebnisse für große Objekte liefert, da sie der tatsächlichen Verteilung der Bounding-Box-Größen im Datensatz am nächsten kommt.



(a) **(b)**

Abbildung 7.16.: Trainings- und Validierungsverlust (a) sowie Metriken (b) für SGD und Adam-Optimierer. Basisnetz: `ResNet50`, Lernrate SGD: $0,01$, Adam: $0,001$, Weight Decay: $0,0001$.

Weiterhin wird der Schwellenwert für die Non-Maximum-Suppression (NMS) evaluiert. Ein zu niedriger Wert von $0,6$ führt dazu, dass zu wenige Regionsvorschläge generiert werden und der Trainingsverlust (Loss) auf Null abfällt, da keine positiven Interessenregionen (Regions of Interest, RoIs) mehr ausgewählt werden. Ein Schwellenwert von $0,7$ erweist sich als stabil und liefert vergleichbare Ergebnisse wie $0,85$. Deshalb wurde $0,7$ für den weiteren Verlauf gewählt.

Bei der Optimierung der Hyperparameter zeigt sich, dass die Aktivierung von Batch-Normalisierung (Batch Normalization, BN) essenziell für die Güte der Vorhersagen ist. Bezüglich der Regularisierung über Gewichtsabnahme (Weight Decay / L2-Regularisierung) liefert ein Wert von $0,0005$ insbesondere bei größeren Objekten präzisere Bounding-Boxen als ein Wert von $0,0001$, da er verhindert, dass größere Boxen fälschlicherweise durch kleinere verdrängt werden.

Nach der Evaluation der Objekterkennung erfolgt die Untersuchung der Affordanz-Erkennung. Für die Hauptkomponente, die pixelweise Segmentierung in die Affordanzklassen, erfolgt ebenfalls eine Optimierung. Im Gegensatz zur Objekterkennung erweist sich hier der Adam-Optimierer als deutlich überlegen gegenüber SGD. Adam führt zu einer stabileren Konvergenz, schnellerem Lernen und signifikant besseren Metriken (F1-Score und IoU). Dies ist deutlich in Abbildung 7.16 erkennbar.

Zur Vermeidung von Overfitting werden verschiedene Regularisierungsstrategien getestet. Die Kombination aus aktivierter Batch-Normalisierung und Dropout-Schichten liefert die besten Ergebnisse. Dabei scheint eine Dropout-Rate von 30 % ein Optimum darzustellen; eine höhere Rate von 50 % fügt zu viel Varianz hinzu und verschlechtert das Ergebnis. Beim Weight Decay zeigte sich im Kontext der Affordanz-Erkennung ein gegenläufiges Bild zur Objekterkennung: Hier führt ein geringerer Wert von $0,0001$ (im Vergleich zu $0,0005$) zu höheren F1-Scores, insbesondere bei der Verwendung des tieferen ResNet-101-Basisnetzes.

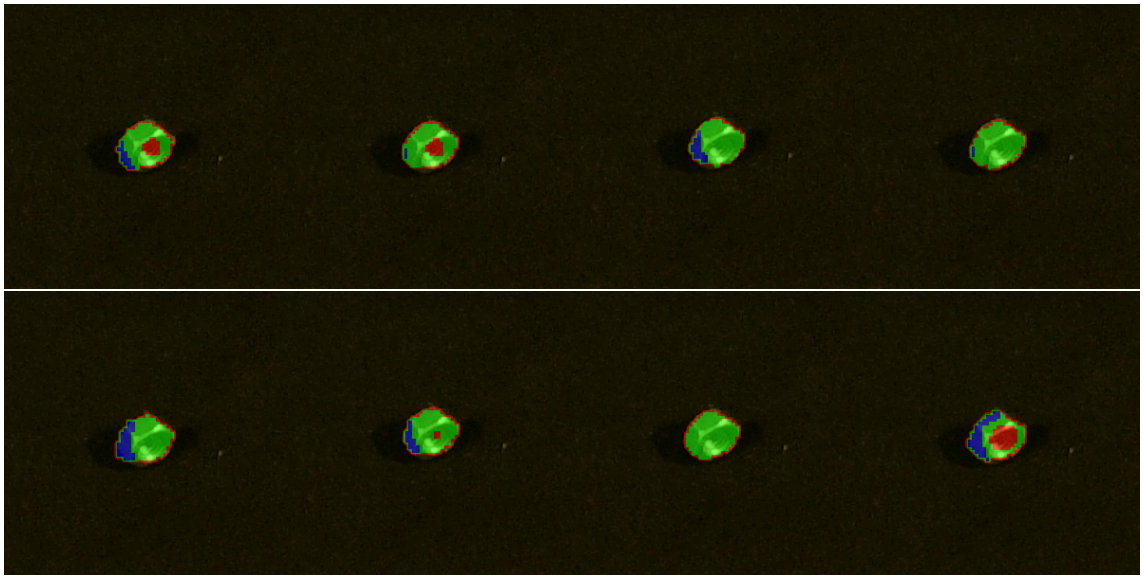


Abbildung 7.17.: Vergleich der Affordanz-Erkennung am Beispiel einer Mutter für die Upsampling-Varianten und `ResNet50`- (oben) bzw. `ResNet101`-Basisnetz (unten). Links: 3×3, Links Mitte: 5×5, Rechts Mitte: 7×7, Rechts: Bilineare Hochskalierung.

Ein zentraler Bestandteil der Architektur ist die Hochskalierung der Merkmalskarten im Decoder-Zweig. Untersucht werden transponierte Faltungen mit Filtergrößen von 3×3, 5×5 und 7×7 sowie bilineare Hochskalierung. Die Ergebnisse zeigen, dass die 3×3 transponierte Faltung (und auch 5×5) die besten Resultate liefert. Dies lässt sich qualitativ am Beispiel einer Mutter demonstrieren: Nur die Varianten mit 3×3- und 5×5-Filterkernen sind in der Lage, das feine Innengewinde korrekt zu erkennen und von der Fläche abzugrenzen (vgl. Abbildung 7.17). Aus Effizienzgründen wird die 3×3-Variante als finaler Standard gewählt.

In der letzten Experimentreihe wird das Gesamtsystem evaluiert und mit einer in der Literatur verfügbaren Referenzarchitektur, dem *AffordanceNet* [147], verglichen. Ein wesentliches Ergebnis ist, dass ein getrenntes Training der beiden Zweige (erst Objekterkennung, dann Affordanz-Erkennung bei eingefrorenen Gewichten) deutlich bessere Ergebnisse liefert als ein Ende-zu-Ende-Training. Bei letzterem dominiert der Verlust der Objekterkennung jenen der Affordanz-Erkennung, was die Segmentierungsgüte negativ beeinflusst, selbst wenn die Affordanz-Fehlerfunktion zehnfach höher gewichtet wird.

Der Vergleich mit *AffordanceNet* zeigt die Vorteile des entwickelten Ansatzes. Während das *AffordanceNet* die Affordanz-Erkennung auf den ausgeschnittenen relevanten Bildbereich (RoIs) durchführt, arbeitet der hier vorgestellte Ansatz entkoppelt auf den Merkmalskarten. Dies ermöglicht eine kontinuierliche Segmentierung, die nicht an den Grenzen der Bounding-Boxen endet. Bei *AffordanceNet* führt die Beschränkung auf RoIs zu Fragmentierungen, insbesondere wenn sich Bounding-Boxen überlappen oder Objekte nicht vollständig erfasst werden. Zudem neigt AffordanceNet dazu, Affordanzbereiche durch das Padding in den

Randbereichen der RoIs künstlich zu verkleinern. Die Unterschiede sind auch auf den segmentierten Bildern klar erkennbar (siehe Abbildung 7.18).

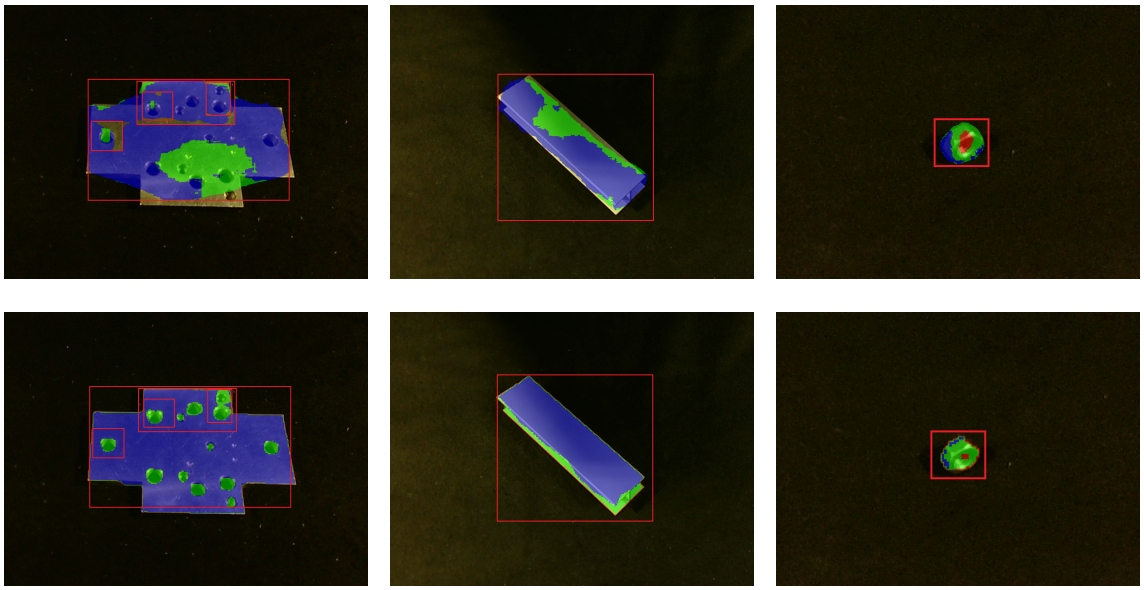


Abbildung 7.18.: Vergleich der Vorhersagen von AffordanceNet (oben) mit dem Ansatz dieser Arbeit (unten).

Quantitativ spiegelt sich dies in den Metriken wider: Der entwickelte Ansatz (mit separatem Training) erzielt auf dem ResNet-50-Basisnetz einen mittleren F1-Score von ca. $0,519$ für die Affordanz-Erkennung. Dies ist vergleichbar mit *AffordanceNet*, jedoch bei qualitativ besserer Struktur der Segmentierung. Auch die Inferenzzeit spricht für den entwickelten Ansatz: Mit ca. $427\,\mathrm{ms}$ pro Bild (ResNet-50, separat trainiert) ist das System schnell genug für den Einsatz in der robotischen Montageplanung. Eine Aufstellung der verschiedenen Inferenzzeiten im Vergleich zum AffordanceNet ist in Tabelle 7.8 dargestellt.

Ansatz	**Training**	**Basisnetz**	**Inferenzzeit pro Bild** (in ms)
AffordanceNet	Ende-zu-Ende	ResNet50	427
		ResNet101	461
	Separat	ResNet50	**427**
		ResNet101	**445**
Eigener Ansatz	Ende-zu-Ende	ResNet50	495
		ResNet101	541
	Separat	ResNet50	**491**
		ResNet101	**529**

Tabelle 7.8.: Inferenzzeit pro Bild für den Ansatz dieser Arbeit und AffordanceNet.

7.4.3. Diskussion und Einordnung der Ergebnisse

Zusammenfassend bestätigen die Experimente, dass die Entkopplung von Objekterkennung und Affordanz-Erkennung in Kombination mit einem ResNet-50-Basisnetz, Adam-Optimierung und einer 3×3-Hochskalierungsstrategie die robustesten Ergebnisse für den Anwendungsfall der Montage liefert. Das System ist in der Lage, Montageaffordanzen zuverlässig zu erkennen, wobei die Unterscheidung zwischen den einzelnen Affordanzklassen stark von der Annotationsqualität des Datensatzes und der Wahl des Basisnetzes abhängt.

7.5. Montageplanverfeinerung

Die Evaluation der Montageplanverfeinerung teilt sich in zwei Abschnitte. Zunächst wird in mehreren kleineren Experimenten die Fähigkeit des Algorithmus validiert, auf einfachen, eindeutigen Baugruppen die korrekte Montagetätigkeit aus den Montageaffordanzen abzuleiten. Anschließend findet ein Experiment mit einer großen komplexen Baugruppe statt. Diese ähnelt in der Komplexität realen Baugruppen und enthält mehrere Montageaffordanzen.

7.5.1. Datensatz und Evaluationsumgebung

Um etwaige Schwachstellen des Algorithmus eindeutig und schnell zu identifizieren, wird dieser zunächst auf einer Menge kleinerer Baugruppen geprüft. Hierfür wird ein Datensatz bestehend aus 30 Baugruppen verwendet. Diese zeichnen sich dadurch aus, dass sie jeweils nur eine Untermenge der möglichen Fügeoperationen enthalten. Die Anzahl der Bauteile pro Baugruppe variiert je nach Modell zwischen zwei und fünf. Pro Baugruppe sind zwischen einer und drei unterschiedliche Montageaktionen notwendig. In Abbildung 7.19 sind zwei dieser Baugruppen dargestellt.

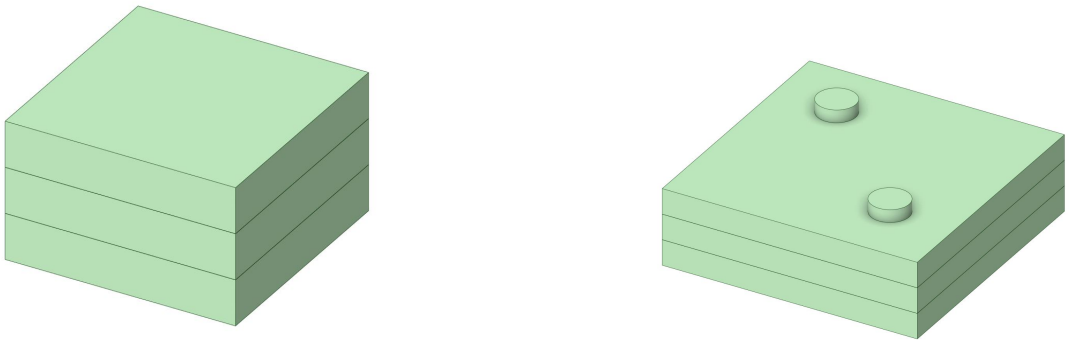


Abbildung 7.19.: Zwei Exemplare aus dem Testdatensatz mit 30 Baugruppen.

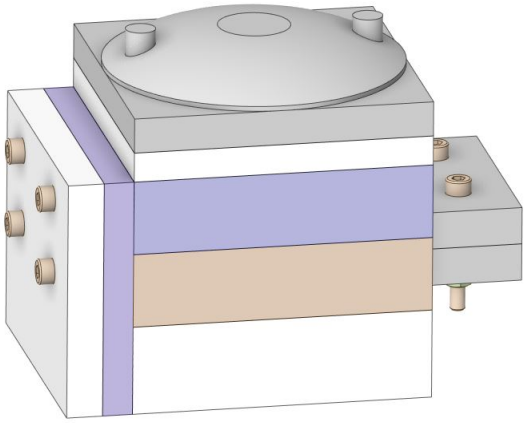


Abbildung 7.20.: Rendering der komplexen Baugruppe, welche für die Evaluation des semantischen Planers verwendet wird.

Um im weiteren Verlauf der Evaluation Aussagen über den Einsatz in realen Szenarien treffen zu können, wird anschließend eine komplexe Baugruppe betrachtet. Hierzu wurde zunächst geprüft, ob im Rahmen dieser Arbeit eine reale Baugruppe genutzt werden kann, welche alle für das System möglichen Fügeoperationen beinhaltet. Da jedoch keine entsprechende Baugruppe verfügbar war, wurde eine eigene Baugruppe speziell für die Evaluation des Algorithmus entwickelt. Sie zeichnet sich durch folgende Merkmale aus:

- Die Baugruppe besteht aus 21 Bauteilen und erfordert daher 20 Fügeoperationen.
- Die Baugruppe setzt sich aus mehreren Unterbaugruppen zusammen.
- Alle Fügeoperationen kommen mindestens einmal vor.
- Die Fügeoperation *LAYUP* kommt in verschiedenen Kombinationen vor.

Die Baugruppe ist in Abbildung 7.20 dargestellt; detaillierte Ansichten können dem Anhang entnommen werden. Um ausschließlich die Performanz des Algorithmus zu bewerten, werden die vom Algorithmus verwendeten Montageaffordanzen zuvor über die Methode *Texture Painting* in der Software Blender auf jedem Bauteil korrekt markiert. Auch der initiale Montagebaum wird für die Experimente manuell erstellt. Die Experimente werden auf einer mobilen Workstation durchgeführt, die mit einer Intel Core i7-11800H CPU, 32 GB Arbeitsspeicher und einer NVIDIA RTX A2000 (4 GB VRAM) ausgestattet ist.

7.5.2. Durchgeführte Experimente und Ergebnisse

Zur Evaluation des Ansatzes wird zunächst ein Korrektheitsmaß definiert. Dieses ermöglicht es, im Fall von Diskrepanzen zwischen dem korrekten Montageplan (Ground Truth) und dem vom Algorithmus generierten Plan eine Aussage darüber zu treffen, wie kritisch die Abweichungen sind.

Es wird bewusst kein binäres Fehlermaß verwendet, da eine Vielzahl potenzieller Fehlerquellen nicht nur im Algorithmus selbst, sondern auch in den zugrundeliegenden Daten begründet liegt. Ein Beispiel für einen solchen Fehler stellt die inkorrekte Bestimmung eines Übermaßes dar, welches für die Unterscheidung zwischen den Fügeoperationen *TELESCOPE* und *PRESS* verwendet wird. Dieser Fehler kann u. a. entstehen, wenn das Baugruppenmodell nicht als parametrisiertes CAD-Modell vorliegt, sondern bereits vorverarbeitet wurde (z. B. durch Umwandlung in ein Oberflächenmodell).

Das Korrektheitsmaß wird als *Assembly Operation Correctness Score (AOCS)* bezeichnet und ist wie folgt definiert:

$$\text{AOCS} = \frac{\sum_{v \in V} p_v}{4|V|} \tag{7.1}$$

Hierbei bezeichnet V die Menge aller Knoten des Montagebaums, $|V|$ deren Mächtigkeit (Anzahl) und p_v die Punktzahl, welche einem spezifischen Knoten v abhängig vom Fehler zugewiesen wird.

Der AOCS entspricht dem auf das Intervall $[0, 1]$ normierten Mittelwert der Punktzahlen p, die jedem Knoten des Montagebaums aufgrund der Schwere des konkreten Fehlers zugewiesen werden. Die konkrete Punktvergabe für die einzelnen Fehlerfälle ist in Tabelle 7.9 dargestellt. Die Unterscheidung der einzelnen Fälle erfolgt wie folgt:

- Im Falle keiner Abweichung zwischen prädizierter und korrekter Fügeoperation nimmt p den Wert 4 an.

Punktzahl	**Fügeoperation Algorithmus**	**Fügeoperation Ground Truth**
4	X	X
3	*TELESCOPE*	*PRESS*
1	*LAYUP*	*MATERIAL*
0	X	Y

Tabelle 7.9.: Punktzahl, welche den unterschiedlichen Ergebnissen des Algorithmus zugewiesen wird. Mit der Ground Truth übereinstimmende Ergebnisse erhalten die höchste Punktzahl.

- Sollte das System das Übermaß falsch bestimmen, wird p der Wert 3 zugewiesen. In diesem Fall wurde ein relativ hoher Wert gewählt, da der Fehler nur geringe Folgen für die spätere Montage hat; die beiden Fügeoperationen *PRESS* und *TELESCOPE* unterscheiden sich primär in ihrem Kraftprofil.
- Im Fall einer Ableitung von *LAYUP* anstelle von *MATERIAL* nimmt p den Wert 1 an. In diesem Fehlerfall ist die Montage der Baugruppe zwar weiterhin möglich, jedoch befindet sich am Ende ein nicht fest verbundenes Bauteil in der Baugruppe, was einer der Grundannahmen des Ansatzes widerspricht.
- In allen anderen Fällen, in denen sich die prädizierte und die korrekte Fügeoperation unterscheiden, wird p der Wert 0 zugewiesen.

Sollte der AOCS einen Wert von 1 annehmen, sind alle Fügeoperationen korrekt. Befindet sich der Wert nahe an 1, sind mit hoher Wahrscheinlichkeit Probleme bei der Vorverarbeitung der Baugruppe entstanden. Bei Werten nahe 0 oder gleich 0 ist der abgeleitete Montagebaum grundlegend falsch.

Im Rahmen der ersten Untersuchungsreihe werden nur Baugruppen betrachtet, welche einen einzelnen Typ von Fügeoperation aufweisen. Dies stellt den einfachsten Fall für das System dar. Auf allen diesen Baugruppen konnte das System einen AOCS-Wert von 1 erreichen und somit alle Fügeoperationen korrekt herleiten. In einem nächsten Schritt werden die restlichen Baugruppen hinzugenommen, welche unterschiedliche Fügeoperationen miteinander kombinieren. Auch auf diesen Baugruppen wird ein AOCS-Wert von 1 erreicht. Wird neben dem AOCS-Wert auch die Laufzeit betrachtet, so ist zu erkennen, dass die Ausführung des Algorithmus über alle Baugruppen hinweg eine ähnliche Zeitspanne benötigt. Genauere Untersuchungen haben ergeben, dass bei kleineren Baugruppen die Vorverarbeitung anteilig am stärksten ins Gewicht fällt. Eine Aufstellung der entsprechenden Laufzeiten ist in Tabelle 7.10 zu finden.

Bei den bisher betrachteten Baugruppen handelt es sich um Gruppen mit wenigen Bauteilen, bei denen auch die für das System notwendigen Überprüfungen, beispielsweise auf Übermaße, stark vereinfacht sind. Auf diesen Baugruppen konnte erfolgreich nachgewiesen werden, dass der Ansatz korrekte Fügeope-

	Vorverarbeitung	**Ableitung Fügeoperation**	**Nachbearbeitung**	**Gesamt**
Minimum	365 ms	23 ms	52 ms	440 ms
Maximum	477 ms	48 ms	69 ms	594 ms
Durchschnitt	413 ms	39 ms	55 ms	507 ms

Tabelle 7.10.: Laufzeit der Montageaffordanz-basierten Planverfeinerung für den Datensatz der weniger komplexen Baugruppen. Die Vorverarbeitung umfasst die gesamte Datenextraktion, während die Nachbearbeitung die Überführung des Montageplans in eine Baumdarstellung und eine textuelle Darstellung beinhaltet.

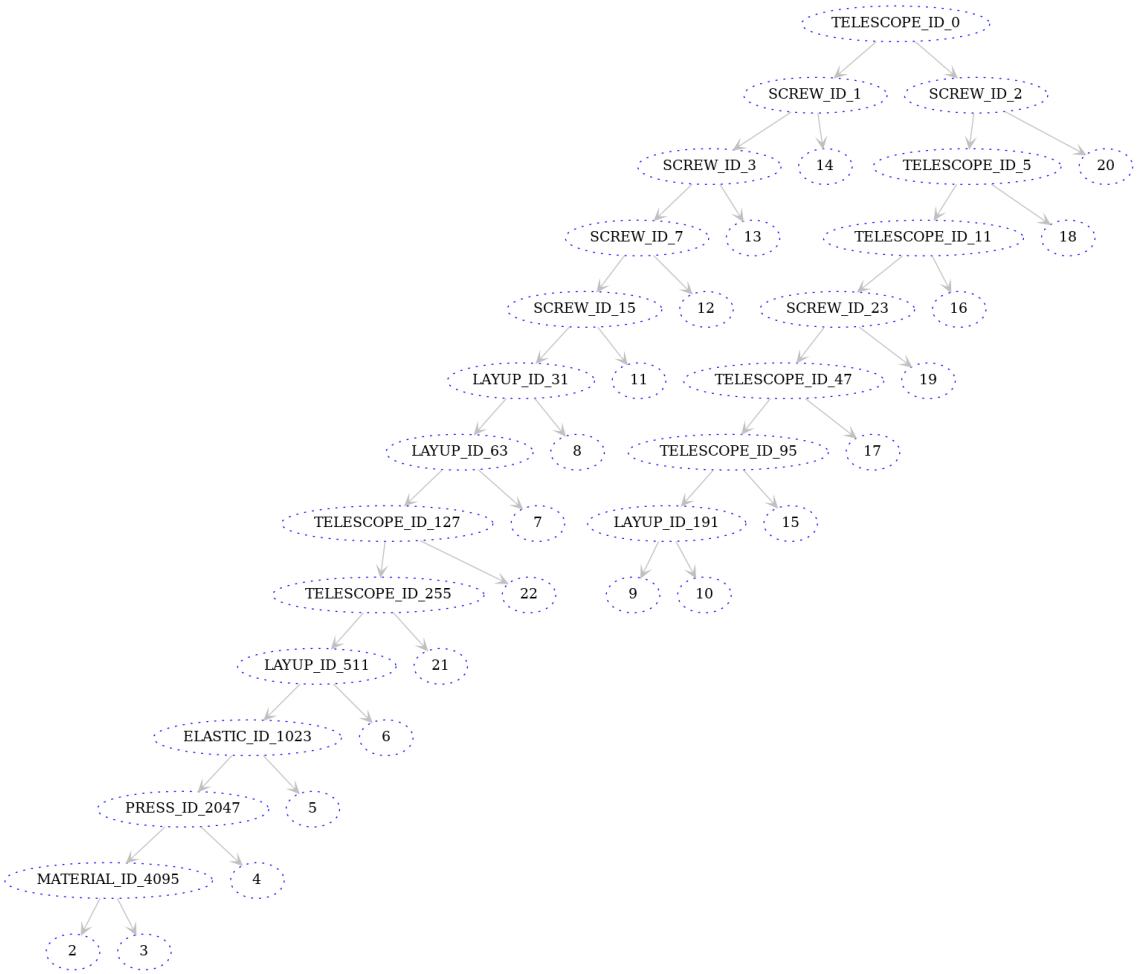


Abbildung 7.21.: Visuelle Darstellung des Montagebaums.

rationen ableiten kann. Nachdem die grundlegende Korrektheit des Systems verifiziert wurde, wird in einem zweiten Schritt geprüft, wie sich der Algorithmus auf der zuvor beschriebenen komplexeren Baugruppe verhält. Auch bei dieser komplexeren Baugruppe war das System in der Lage, sowohl die Vorverarbeitung durchzuführen als auch alle Fügeoperationen korrekt abzuleiten, sodass ein AOCS-Wert von 1 erreicht werden konnte. Abbildung 7.21 stellt den durch den Algorithmus abgeleiteten finalen Montagebaum dar.

Ebenfalls untersucht wird die Laufzeit des Systems bei der Ausführung auf der komplexen Baugruppe. Die Ergebnisse hierzu sind in Tabelle 7.11 dargestellt. Es ist zu erkennen, dass die Vorverarbeitung, wie zuvor auch bei den kleineren Bau-

Vorverarbeitung	Ableitung Fügeoperation	Nachbearbeitung	Gesamt
1872 ms	532 ms	121 ms	2525 ms

Tabelle 7.11.: Laufzeit der Montageaffordanz-basierten Planverfeinerung für die komplexe Baugruppe.

gruppen, am zeitintensivsten ist. Die Ausführung der eigentlichen Ableitung aller korrekten Fügeoperationen ist mit unter 550 ms für den Einsatz in einem realen Montageszenario hinreichend schnell. Auch die Aufbereitung der Ergebnisse für die spätere Weiterverarbeitung in Form von Montagebaum und textueller Darstellung stellt zeitlich kein Problem dar.

7.5.3. Diskussion und Einordnung der Ergebnisse

Die Evaluation der Montageplanverfeinerung durch den Einsatz von Montageaffordanzen zeigt das Potenzial dieses Ansatzes auf. Das Ableiten der Fügeoperationen war in allen getesteten Fällen erfolgreich, und die notwendige Gesamtlaufzeit liegt auch bei der größeren Baugruppe unter 3 Sekunden, was den Einsatz zur Laufzeit der Montageanwendung ermöglichen sollte. Kritisch zu betrachten sind die für die korrekte Ausführung relevanten Vorberechnungen. Bei diesen ist besonders die Detektion der Montageaffordanzen entscheidend. Während in den durchgeführten Experimenten die Affordanzen zuvor händisch markiert wurden, wäre für einen späteren Einsatz in einer autonomen Anwendung wünschenswert, dass diese entweder bereits bekannt sind oder durch das zuvor untersuchte Detektionssystem automatisch erkannt werden können.

Insgesamt stellt der Ansatz zur Planverfeinerung eine Alternative zur händischen Definition notwendiger Manipulationsschritten dar. Gleichzeitig kann er auch als zusätzliche Grounding-Komponente für den in Kapitel 5.3 vorgestellten Ansatz zur Montageplanung mittels LLM dienen.

7.6. Zusammenfassung und Fazit der Experimente und Evaluation

Die durchgeführten Experimente und die Evaluation der umgesetzten Konzepte verdeutlichen, dass KI-basierte Ansätze in verschiedensten Montageszenarien erfolgreich einsetzbar sind.

Obwohl Ansätze mit kleineren Modellen in spezifischen Szenarien eine sehr gute Performanz erzielen können, wie der Fall des Sequenz-zu-Sequenz-Ansatzes verdeutlicht, sind sie aufgrund diverser Einschränkungen nicht universell einsetzbar. Besonders die Restriktion auf voxelisierte Baugruppen wiegt schwer. Auch wenn der eigentliche Voxelisierungsprozess technisch einfach umsetzbar ist, bleibt der Detailverlust in vielen Montageszenarien ein Ausschlusskriterium. Der Ansatz in dieser Arbeit lernt primär auf Basis der Objektgeometrie und der Position der Bauteile, eine mögliche Montagesequenz zu generieren. Dementsprechend entwickelt er kein direktes Verständnis für die technologischen Implikationen einer Baugruppe, wie beispielsweise die funktionale Bedeutung einer Dichtung.

Dies erweist sich primär bei komplexen Abhängigkeiten der Bauteile untereinander als hinderlich.

Ein signifikant größeres Potenzial für die allgemeine Anwendung zeigte hingegen der LLM-basierte Ansatz. Die Einbindung von Detailwissen über die Baugruppe mittels des Wissensgraphen hat sich als zielführend erwiesen; zudem konnte durch die Validierungskomponente die Güte der generierten Montagesequenzen gesteigert werden. Eine aktuelle Herausforderung stellt die Grounding-Komponente dar: Deren Ausgabe liefert dem LLM in vielen Fällen noch kein hilfreiches Feedback. Zukünftige VLM-Modelle haben jedoch das Potenzial, die Relevanz dieser Komponente deutlich zu steigern. Spezifische Sonderfälle belegen bereits jetzt, dass die Grounding-Komponente einen Mehrwert bieten kann. Die Verwendung großer Foundation Models ermöglicht in diesem Ansatz zusätzlich die Generierung der erforderlichen Manipulationssequenz. Während dies für die untersuchten Baugruppen zuverlässig funktionierte, erfordert es zusätzliche Inferenzdurchläufe und verlängert somit die Gesamtlaufzeit des Ansatzes. Im Vergleich zum affordanzbasierten Ansatz sind die Laufzeiten deutlich höher. Das LLM bietet jedoch den Vorteil, dass lediglich ein einziges Framework implementiert werden muss.

Ein weiteres generelles Fazit des LLM-basierten Ansatzes ist, dass die Performanz des Systems maßgeblich von der Leistungsfähigkeit der verwendeten Modelle abhängt. Kleinere Modelle erreichen trotz des Einsatzes der Validierungs- und Grounding-Komponenten keine ausreichende Qualität. Ähnlich wie bei den VLMs ist zu berücksichtigen, dass LLMs ein hochdynamisches Forschungsfeld darstellen, sodass zukünftige Modellgenerationen weitere Verbesserungen versprechen. Abschließend lässt sich feststellen, dass das umfangreiche Hintergrundwissen des Foundation Models in Verbindung mit dem szenariospezifischen Wissen aus dem Wissensgraphen einen Transfer auf neue Montageszenarien prinzipiell ermöglicht. Im Hinblick auf die Validierung durch das Framework sowie die Einhaltung der gewünschten Ausgabeformate zeigt sich jedoch, dass hierfür – wenn auch geringfügige – Anpassungen an der Prompt-Vorlage und der Validierungskomponente erforderlich sind.

Affordanzen haben sich ebenfalls als geeignete zusätzliche Informationsquelle für die Bestimmung der korrekten Fügeoperation erwiesen. Eine robuste Erkennung durch einen bildbasierten KI-Ansatz konnte realisiert werden. Zudem erwies sich die Entwicklung eines spezifischen Ansatzes für Montageaffordanzen als effektiv, wie der Vergleich mit Implementierungen aus dem Stand der Technik belegt.

Die Verarbeitung der erkannten Affordanzen durch einen semantischen Planer erfolgt reibungslos. Dabei bietet das System die Möglichkeit, unabhängig von der KI-basierten Erkennung eingesetzt zu werden. Dies ermöglicht eine einfache Einbindung in Applikationen, in denen Affordanzinformationen bereits hinterlegt sind. Durch die Berücksichtigung der Norm DIN 8580 ist sichergestellt, dass alle relevanten Fügeoperationen beachtet werden. Ein wesentlicher Vorteil dieses

Ansatzes ist seine geringe Laufzeit im Vergleich zur Manipulationssequenzgenerierung durch das LLM. Allerdings beschränkt sich dieser Teil auf die Herleitung der Fügeoperationen und deckt nicht die damit zusammenhängenden restlichen Manipulationen ab.

Das entwickelte und evaluierte Konzept ist konsequent auf einen vollständig autonomen Betrieb ausgelegt. Eine Interaktion mit einem Anwender ist jenseits der Ansteuerung der einzelnen Komponenten nicht vorgesehen. Zusätzliche Interaktionsmöglichkeiten während des Lernvorgangs oder im späteren Produktionsbetrieb wären primär dann relevant, wenn die vorliegende Informationsbasis unzureichend ist, beispielsweise bei unvollständigen CAD-Modellen oder nicht erkannten Montageaffordanzen. Eine solche Komponente könnte zur weiteren Steigerung der Robustheit in das Konzept integriert werden.

8. Zusammenfassung und Ausblick

Die automatisierte dynamische Montage- und Demontageplanung ist ein zentrales Element im Bestreben, Fertigungs- und Recyclinganwendungen zukunftssicher zu gestalten. Ein Planungssystem muss in der Lage sein, selbstständig auf neue und teilweise unbekannte Produkte zu reagieren, um deren Verarbeitung ohne Stillstandszeiten sicherzustellen. Große Fortschritte wurden in diesem Forschungsfeld bereits im Bereich der Optimierung existierender Pläne erzielt. Jedoch ermöglichen erst moderne KI-Verfahren, die notwendigen Adaptionsprozesse bei Produktänderungen zeitnah umzusetzen. Um dabei über verschiedene Produktklassen hinweg eingesetzt werden zu können, müssen diese Systeme sowohl ausreichend generalisiert sein als auch ein tiefgehendes Verständnis für die konkreten Baugruppen aufweisen. Diese Arbeit konzentriert sich auf drei wesentliche Problemfelder in diesem Kontext:

- **Informationsbereitstellung:** Durch das Verfügbarmachen von konkretem, komplexem Wissen über die Baugruppe können sich KI-Verfahren während der Ausführung adaptieren.
- **Sequenzplanung:** Basierend auf dem generalisierten Wissen und dem konkreten Wissen über eine Baugruppe wird eine kollisionsfreie Montage- oder Demontagesequenz generiert.
- **Sequenzüberführung:** Für eine Ausführung mit einem robotischen System muss eine Sequenz mit konkreten Manipulationsoperationen in jedem Sequenzschritt angereichert werden, wodurch die Sequenz in einen symbolischen Plan überführt wird.

In dieser Arbeit werden die genannten drei Aspekte in einem dynamischen Planungsframework verbunden. Es werden Wissensgraphen verwendet, um Detailwissen über eine vorliegende Baugruppe einzubinden. Das vorgestellte Planungsframework ist unabhängig von konkreten Szenarien und kann so auf verschiedene Baugruppen übertragen werden. Unter der Voraussetzung einer zerstörungsfreien Demontage können die vorgestellten Konzepte sowohl auf die Montage als auch auf die Demontage angewandt werden.

Kapitel 1 vermittelt einen allgemeinen Überblick zu aktuellen Herausforderungen der Montage- und Demontageplanung. Des Weiteren werden die relevanten Forschungsfragen und Ziele der Arbeit vorgestellt. Kapitel 2 bietet einen umfassenden Überblick zu relevanten theoretischen Grundlagen der Thematik. In Kapitel 3 steht der aktuelle Forschungsstand im Bereich KI-basierter Montageplanung im Mittelpunkt.

Kapitel 4 widmet sich dem Gesamtkonzept der Montage-/Demontageplanung und ordnet es in den Zusammenhang von Herausforderungen und Entwicklungsmöglichkeiten ein. Die Kapitel 5 und 6 stellen den Kernbeitrag zur Montagesequenzgenerierung und zur affordanzbasierten Sequenzüberführung dar. Dazu werden zunächst die Konzepte der Montagesequenzgenerierung vorgestellt und angepasste Verfahren, z. B. der genetische Algorithmus, erläutert. Abschnitt 5.2 stellt als erstes Konzept einen Sequenz-zu-Sequenz-basierten Generierungsansatz vor, der auf Basis von voxelisierten Eingangsdaten eine mögliche Montagesequenz erlernt und anschließend für ähnliche Baugruppen erzeugen kann. Abschnitt 5.3 beschreibt einen alternativen Ansatz, der mittels eines LLMs die Sequenzgenerierung durchführt. Das breite Basiswissen des Foundation Models wird hierfür um Detailwissen aus einer CAD-Datei ergänzt. Hierzu wird das Konstrukt des Wissensgraphen verwendet, welches es erlaubt, die Baugruppeninformationen dem LLM in strukturierter und verständlicher Form zugänglich zu machen. Für eine höhere Zuverlässigkeit des Ansatzes werden eine Validierungskomponente und eine VLM-basierte Grounding-Komponente eingebunden. In Kapitel 6 liegt der Fokus auf der Überführung einer Montagesequenz in einen konkreten Montageplan durch die Verwendung von Montageaffordanzen. Dazu wird das Konzept der Montageaffordanzen in Abschnitt 6.1 skizziert und in Abschnitt 6.2 mit Blick auf das Vorgehen der automatisierten Generierung von Trainingsdaten für eine spätere KI-basierte Affordanzerkennung erläutert. Abschnitt 5.3 stellt einen konkreten Ansatz für die Detektion von Montageaffordanzen vor. Diese detektierten Affordanzen werden in Abschnitt 5.4 von einem semantischen Planer verwendet, um die Überführung einer Montagesequenz in einen Montageplan durchzuführen.

Kapitel 7 umfasst die Evaluation der entwickelten Konzepte. In verschiedenen Experimenten wurde die Machbarkeit und Qualität des dynamischen Montageplanungskonzeptes nachgewiesen. Der erste Teil der Evaluation befasst sich mit der Montagesequenzgenerierung auf der Grundlage der verschiedenen Ansätze, im zweiten Teil liegt der Fokus auf den Montageaffordanzen und deren Verarbeitung durch den semantischen Planer.

Dieses letzte Kapitel gibt einen Überblick und diskutiert die in Abschnitt 1.2 vorgestellten ursprünglichen Forschungsfragen. Anschließend werden die aktuellen Grenzen des vorgestellten Ansatzes aufgezeigt und weitere Forschungsmöglichkeiten abgeleitet.

8.1. Beitrag der Arbeit

Der wissenschaftliche Beitrag dieser Arbeit konzentriert sich in erster Linie auf drei offene Forschungsfragen. In diesem Abschnitt wird jede dieser Fragen im Kontext der in dieser Arbeit vorgeschlagenen Konzepte erneut aufgegriffen und diskutiert.

Forschungsfrage 1: In welchem Format und mit welcher Methodik können komplexe Produktdaten einem KI-System zugänglich gemacht werden?

Das Verfügbarmachen der komplexen Produktdaten für das KI-System stellt eine der entscheidendsten Fähigkeiten für ein Montageplanungssystem dar. Da das generalisierte Wissen der KI auf das konkret vorliegende Problem angewandt werden muss, ist es unerlässlich, dass die Informationen über die aktuelle Baugruppe klar strukturiert vorliegen, um Änderungen, etwa bei Bauteilen, deutlich erkennbar zu machen.

Im Verlauf dieser Arbeit wurde hierzu ein Framework entwickelt, welches die vorliegenden CAD-Daten einer Baugruppe automatisiert verarbeitet und in eine strukturierte Darstellung in einem Wissensgraphen überführen kann. Ein zentraler Aspekt ist die klare Darstellung der Abhängigkeiten der Komponenten untereinander durch die Abgrenzung zwischen Montage- und Kollisionskanten. Während der betrachtete Sequenz-zu-Sequenz-Ansatz nur Teilwissen aus dem Wissensgraphen benötigt (konkret das Oberflächenmodell der Baugruppe), zeigt im LLM-basierten Ansatz der Wissensgraph sein volles Potenzial. Hier stellt er gebündeltes Szenarienwissen bereit, das in verständlicher Form dem LLM als zusätzliche Informationsquelle zugeführt werden kann.

Das Konzept des Wissensgraphen und insbesondere dessen Einbindung in ein KI-System sind ein konkretes Beispiel dafür, wie komplexe Daten für KI zugänglich gemacht werden können. Auch wenn ein System nicht die vollständige Informationsmenge aus einem Wissensgraphen benötigt, so kann dieser als Grundlage verwendet werden, um effizient verschiedene Informationen aus ihm auszulesen. Zudem ist das Konzept des Wissensgraphen nicht auf die Domäne der Montageplanung beschränkt und bietet somit eine einfache Transferierbarkeit in andere robotische Domänen.

Forschungsfrage 2: Wie sieht eine KI-Architektur aus, welche eine Montage- bzw. Demontagesequenz generieren kann, und wie gut generalisiert ein solcher Ansatz?

Die Montagesequenzgenerierung ist als NP-schweres Problem nicht robust auf triviale Weise lösbar. KI-basierte Verfahren können hierbei eine potenzielle Lösung darstellen.

In der Arbeit wurden hierzu verschiedene Ansätze betrachtet. Während das Sequenz-zu-Sequenz-basierte Konzept in dem untersuchten Szenario gute Ergebnisse liefern konnte, ist die generelle Anwendbarkeit dieses Verfahrens eingeschränkt, da es die Informationen über die Baugruppe stark abstrahiert. Zudem kann dieser Ansatz das Potenzial des Wissensgraphen nur bedingt verwerten.

Der alternative Ansatz, der ein LLM als Planungskomponente verwendet, ist in dieser Hinsicht besser geeignet. Aktuelle Foundation Models verfügen über ein ausreichend großes Grundlagenwissen, welches durch den Wissensgraphen

sinnvoll ergänzt werden kann. Eine Montagesequenzgenerierung wird somit möglich. Über verschiedene Baugruppenklassen hinweg genügen das Grundlagenwissen und die spezifischen Baugruppeninformationen, um ohne zusätzliches Fine-Tuning Montagesequenzen von ausreichender Güte zu erzeugen. Herausforderungen wie Halluzinationen oder das Abweichen von gewünschten Ausgabeformatierungen können durch Validierungsalgorithmen oder ein Grounding durch ein zusätzliches VLM adressiert werden. Um der noch nicht immer optimalen Performance aktueller VLM-Modelle Rechnung zu tragen, ist es möglich, das VLM aus dem Verifikationszyklus herauszulösen. Über die ursprünglichen Erwartungen hinaus stellt das Konzept des LLMs bereits eine verwendbare Basis dar, um die Montagesequenz anschließend in eine Manipulationssequenz zu überführen.

Forschungsfrage 3: Mit welcher Methodik ist es möglich, eine Sequenz ohne Manipulationsinformationen in einen symbolischen High-Level-Plan zu überführen?

Um die für die spätere Verarbeitung durch ein Robotersystem notwendige Verfeinerung der Montagesequenz durch Anreicherung mit Manipulationsinformationen zu erreichen, hat die Arbeit zwei mögliche Wege aufgezeigt:

1. Das im Konzept integrierte LLM hat gezeigt, dass es über ein ausreichend tiefes Wissen verfügt (welches lediglich durch Informationen über verfügbare Manipulationsfähigkeiten erweitert werden muss), um bereits für jeden Montageschritt eine sinnvolle Manipulationssequenz zu generieren. Herausforderungen, wie z. B. Probleme bei der korrekten Formatierung der Ausgabe, konnten durch Verfeinerung des verwendeten Prompts und Anpassungen des Validierungsansatzes vermindert werden, sodass auch eine direkte Weiterverarbeitung durch eine intelligente Montagezelle im Bereich des Möglichen liegt.

2. Als Alternative stellen sich die Montageaffordanzen dar. Dieses regelbasierte Konzept bietet in Kombination mit einem semantischen Planer eine stets nachvollziehbare Ableitung der korrekten Fügeoperation pro Montageschritt. Hierbei kann der Ansatz durch eine nachgelagerte Betrachtung der gesamten Baugruppe auch stoffschlüssige Fügeverfahren herleiten, was rein auf Basis der Objektgeometrie nicht erkennbar ist. Im Vergleich zu einem KI-basierten Verfahren ist dieses deutlich performanter und somit für den Online-Einsatz geeignet.

8.2. Ausblick

In dieser Arbeit wurden verschiedene Konzepte vorgestellt und diskutiert, die den aktuellen Stand der Technik in der Montageplanung erweitern. Mittels Evaluation wurde die Machbarkeit der vorgeschlagenen Ansätze belegt und gezeigt,

dass die Entwicklungen die Lücken im aktuellen Stand der Forschung schließen. Dennoch bestehen weitere mit dem Thema der Arbeit eng verknüpfte Forschungsfragen.

Ausführungszeit: Ein entscheidender Faktor bei der Nutzung von Methoden für die dynamische Montage- und Demontageplanung ist die Laufzeit. Während klassische Verfahren ggf. mehrere Stunden oder Tage rechnen, bis sie ein Ergebnis produzieren, können KI-Verfahren das Problem während der Inferenz mitunter in Sekunden lösen. Auch wenn es in den betrachteten Montageplanungsfällen nicht zwingend erforderlich ist, eine Lösung innerhalb von Millisekunden zu generieren, so ist eine möglichst geringe Ausführungszeit erstrebenswert.

Eine bestehende Herausforderung, die allerdings nicht vollständig in dieser Arbeit adressiert werden konnte, ist die Laufzeit des LLM-basierten Ansatzes. Aspekte wie die langwierige Argumentationskette (was bei Chain-of-Thought-Modellen gewünscht ist) und die sequenzielle Tokenausgabe sorgen dafür, dass die Laufzeit des LLMs im Bereich von ein bis zwei Stunden für eine Baugruppe liegen kann. Auch wenn dies teilweise in der verwendeten Hardware begründet ist, so trägt das Modell selbst am stärksten zu dieser Problematik bei. Aspekte wie das Deaktivieren der VLM-basierten Grounding-Komponente oder die Verwendung des semantischen Planers in Verbindung mit Montageaffordanzen für die Ableitung der Fügeoperationen haben gezeigt, dass sie die Möglichkeit bieten, die Laufzeit zu reduzieren, ohne die Ergebnisqualität zu verschlechtern. Dennoch wäre eine weitere Reduzierung der Laufzeit wünschenswert. Die in dieser Arbeit getesteten alternativen LLMs hatten das Potenzial, dies zu erreichen, konnten jedoch keine ausreichende Ergebnisgüte erzielen, um für einen Einsatz berücksichtigt zu werden.

Das Forschungsfeld der LLMs und VLMs ist eines der aktivsten in der Informatik. In regelmäßigen Abständen ermöglichen neue Modelle deutliche Performancesprünge oder erreichen bei kleinerer Größe die Performance der vorherigen Generation. Daher erscheint es möglich, dass in naher Zukunft ein Modell verfügbar ist, welches das in dieser Arbeit präferierte QwQ-Modell ersetzen kann und eine kürzere Ausführungszeit bietet. Bei Fortschritten im Zuge der Quantisierung von Foundation Models erscheint es auch realistisch, solche Netze in Zukunft lokal in der intelligenten Automatisierungszelle auszuführen und somit eine weitere Laufzeitverkürzung zu erreichen.

Vision-Language-Action-Modelle: Ein weiterer Nachteil des im Rahmen der Arbeit entwickelten Konzeptes ist die Fragmentierung der Komponenten. Daten müssen jeweils von der einen in die nächste Komponente übertragen und dafür ggf. in ein korrektes Format überführt werden. Dies kann auch eine kumulative Fehlerpropagation zur Folge haben. Ansätze wie das automatisierte Generieren von Robotercode aus symbolischen High-Level-Plänen, wie im Projekt GANResilRob [180] realisiert, minimieren diese Problematik, können sie jedoch nicht vollständig vermeiden.

Vision-Language-Action-Modelle (VLAs) adressieren dieses Defizit umfänglich, indem sie einen Paradigmenwechsel hin zu einer echten Ende-zu-Ende-Architektur ermöglichen, die multimodale Eingaben direkt in physische Aktionen übersetzt. Das große Potenzial solcher Ansätze ergibt sich dadurch, dass diese Modelle semantisches Weltwissen aus großen Sprachmodellen mit visueller Wahrnehmung in einem einzigen neuronalen Netz fusionieren. Hierdurch entfällt die Notwendigkeit für komplexe, manuell definierte Schnittstellen zwischen der semantischen Planung und der späteren Trajektorienplanung.

VLAs stellen daher ein wichtiges zukünftiges Forschungsfeld dar. Durch die Integration aller Komponenten in ein System kann neben einer Robustheitssteigerung auch eine mögliche Reduktion der Laufzeit erreicht werden, da kein Wechsel zwischen verschiedenen Komponenten oder KIs notwendig ist.

Anhang

A. Architektur des Affordanz-Erkennungs-Netzwerks

Im Folgenden werden die für das Single-Stream-Netzwerk der Affordanz-Erkennung verwendeten Architekturen dargestellt.

A.1. Feature-Pyramid-Network-Decoder-Backbone

Die Architektur des im Objekterkennungszweig eingesetzten Feature-Pyramid-Network-Decoder, welcher für die Erkennung unterschiedlicher Objektgrößen dient.

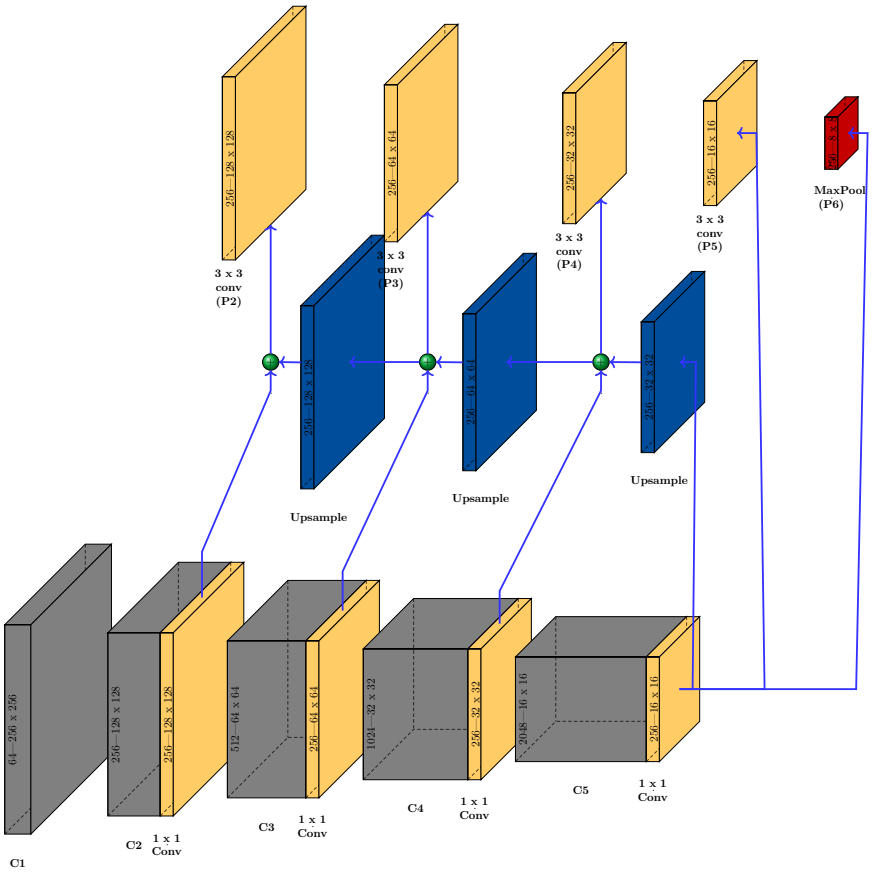


Abbildung A.1.: Architektur des Feature-Pyramid-Network-Backbones

A.2. Region-Proposal-Netzwerk

Das verwendete Region-Proposal-Netzwerk, welches für die Erkennung von relevanten Bildbereichen genutzt wird.

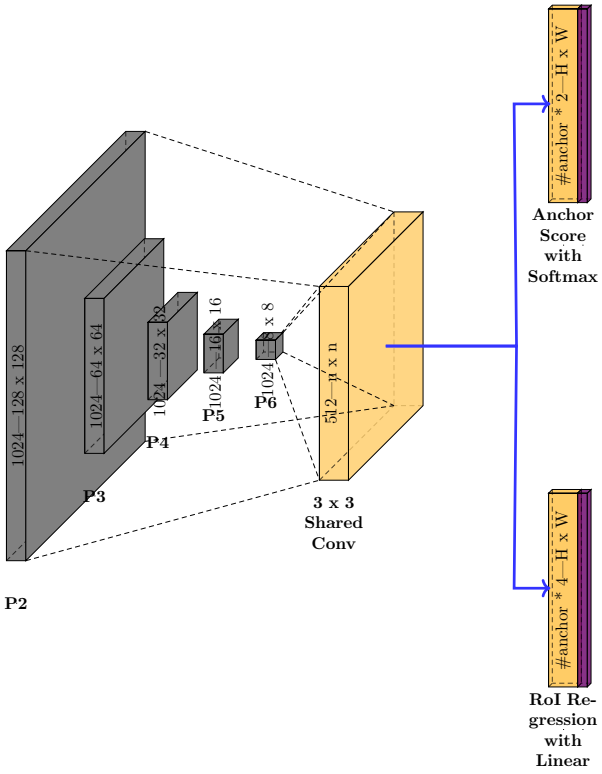


Abbildung A.2.: Architektur des Region-Proposal-Netzwerk

A.3. Objekterkennung

Durch die gewählte Architektur der Objekterkennung können gefunde Objekte über Bounding-Boxes markiert werden.

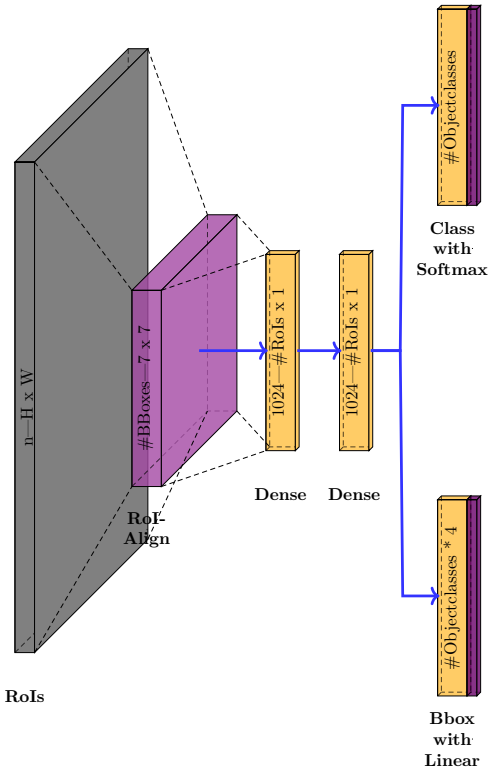


Abbildung A.3.: Architektur des Zweiges zur Objekterkennung

A.4. Affordance-Erkennung

Für die Segmentierung der Montageaffordanzen wird ein separates Encoder-Decoder-Teilnetzwerk eingesetzt mit folgender Architektur eingesetzt.

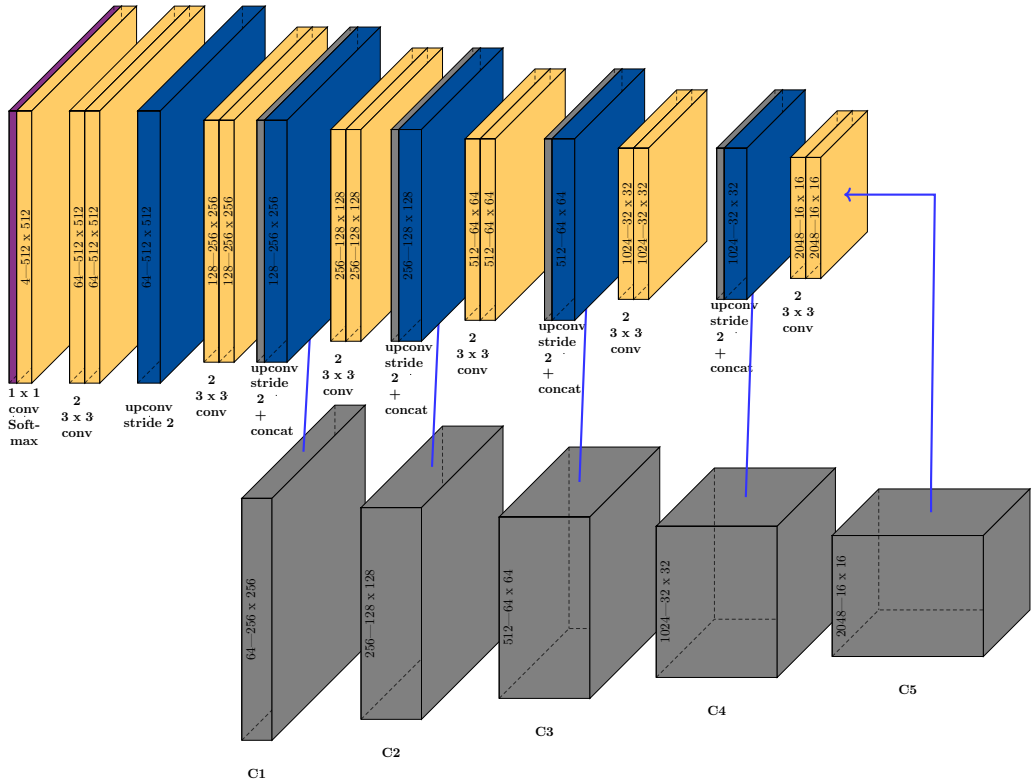


Abbildung A.4.: Architektur des Zweiges zur Affordance-Erkennung

B. Testbaugruppen

Im folgenden werden die verschiedenen Baugruppen, welche für die Evaluation der Affordanzen-basierten Montageplanung verwendet worden erläutert.

B.1. Drei Platten, zwei Stifte

Dieses Modell wirde in zwei Ausführungen für die Evaluation genutzt: passgenau und mit Übermaß.

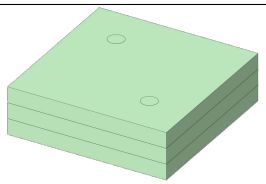
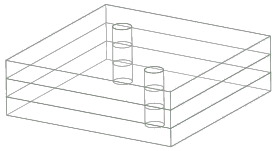
3D-Modell	Drahtmodell
	

Tabelle B.1.: Eine Baugruppe bestehend aus drei Platten und zwei Stiften.

B.2. Drei Platten, zwei Stifte mit Köpfen

Die Baugruppe ist mit passgenauen Stiften modelliert.

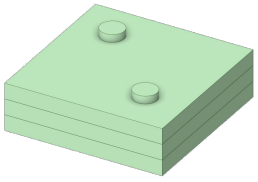
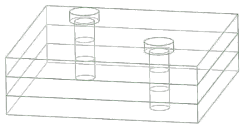
3D-Modell	Drahtmodell
	

Tabelle B.2.: Eine Baugruppe bestehend aus drei Platten und zwei Stiften mit Knöpfen.

B.3. Zwei Platten mit Noppen

Die Baugruppe wird in zwei Ausführungen evaluiert: passgenau und mit Übermaß

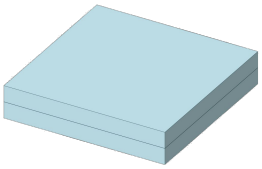
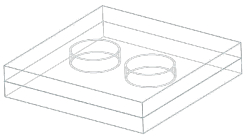
3D-Modell	Drahtmodell
	

Tabelle B.3.: Eine Baugruppe bestehend aus zwei Platten mit Noppen.

B.4. Vier Platten

Bei dieser Baugruppe handelt es sich um 4 übereinander gestapelten Platten.

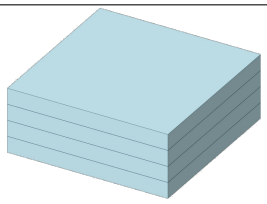
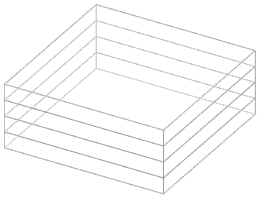
3D-Modell	Drahtmodell
	

Tabelle B.4.: Eine Baugruppe bestehend aus vier Platten.

B.5. Zwei Platten, Schraube

Die Baugruppe besteht aus zwei Platten, welche mit einer Schraube verbunden sind.

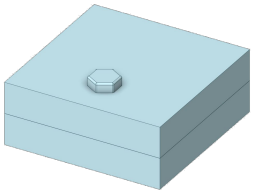
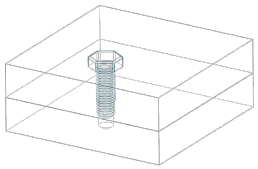
3D-Modell	Drahtmodell
	

Tabelle B.5.: Eine Baugruppe bestehend aus zwei Platten und einer Schraube.

B.6. Gewölbte Bauteile mit Schnappverschluss und Stiften

Das Modell wird in drei Ausführungen evaluiert: Stifte mit Untermaß, passgenaue Stifte und Stifte mit Übermaß.

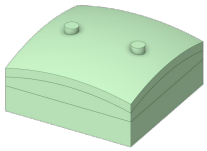
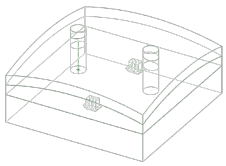
3D-Modell	Drahtmodell
	

Tabelle B.6.: Eine Baugruppe bestehend aus mehreren Platten über Schnappverschluss und Stifte verbunden.

B.7. Schräge

Die Baugruppe wird in drei Varianten verwendet: Noppen mit Untermaß, passgenaue Noppen und Noppen mit Übermaß.

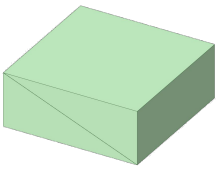
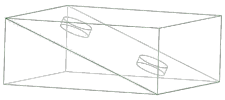
3D-Modell	Drahtmodell
	

Tabelle B.7.: Eine Baugruppe bestehend aus zwei Platten.

B.8. Schräge mit Anbau

Das Modell ist mit passgenauen Noppen konstruiert.

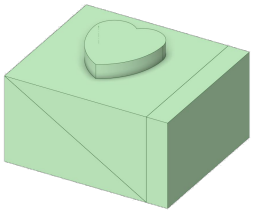
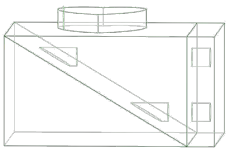
3D-Modell	Drahtmodell
	

Tabelle B.8.: Eine Baugruppe bestehend aus drei miteinander verzahnten Platten.

B.9. Stack mit Anbau

Fünf Varianten dieser Baugruppe werden in der Evaluation verwendet:
1. Noppen Mitte passgenau, Noppen Seite passgenau;
2. Noppen Mitte passgenau, Noppen Seite Übermaß;
3. Noppen Mitte Übermaß, Noppen Seite passgenau;
4. Noppen Mitte Übermaß, Noppen Seite Untermaß;
5. Noppen Mitte Untermaß, Noppen Seite Übermaß.

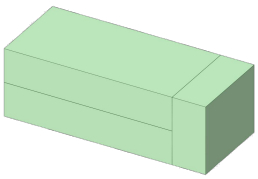
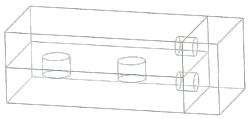
3D-Modell	Drahtmodell
	

Tabelle B.9.: Eine Baugruppe bestehend aus drei miteinander verzahnten Platten.

B.10. Stack mit Noppen

Auch diese Baugruppe wird in fünf Varianten evaluiert:
1. Noppen unten passgenau, Noppen oben passgenau;
2. Noppen unten Untermaß, Noppen oben passgenau;
3. Noppen unten passgenau, Noppen oben Übermaß;
4. Noppen unten Übermaß, Noppen oben Übermaß;
5. Noppen unten Untermaß, Noppen oben Übermaß;

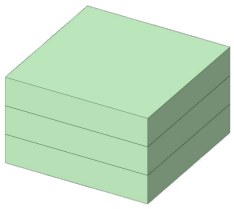
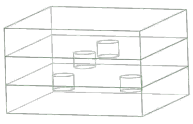
3D-Modell	Drahtmodell
	

Tabelle B.10.: Eine Baugruppe bestehend aus drei vertikal übereinander gestapelten Platten.

B.11. Stack mit Noppen und einem Stift

Die folgende Baugruppe wird in sechs Varianten in der Evaluation berücksichtigt:
1. Stift passgenau, Noppen passgenau;
2. Stift Übermaß, Noppen passgenau;
3. Stift Übermaß, Noppen Untermaß;
4. Stift Untermaß, Noppen passgenau;
5. Stift Untermaß, Noppen Übermaß;
6. Stift Untermaß, Noppen Untermaß;

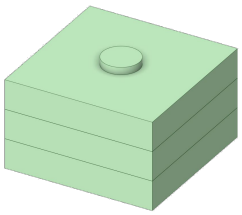
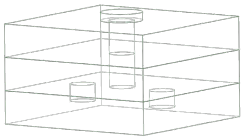
3D-Modell	Drahtmodell
	

Tabelle B.11.: Eine Baugruppe bestehend aus drei vertikal übereinander gestapelten Platten mit einem Stift verbunden.

B.12. Komplexe Baugruppe

Die für die Evaluation verwendete komplexe Baugruppe bestehend aus 21 Bauteilen.

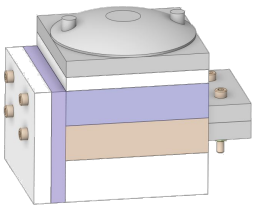
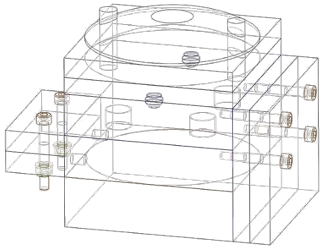
3D-Modell	Drahtmodell
	

Tabelle B.12.: Die komplexe Baugruppe mit 21 Bauteilen.

C. iBOSS

Satelliten und die von ihnen bereitgestellten Dienste sind heutzutage allgegenwärtig. Eine zentrale Herausforderung im Satellitenbetrieb stellt jedoch ihre begrenzte Lebensdauer dar. Diverse Ereignisse, wie etwa Beschädigungen durch Weltraumschrott oder das Versagen von Bauteilen, beeinflussen die Nutzungsdauer eines Systems meist unvorhersehbar. Während terrestrische Systeme durch den Austausch defekter Komponenten repariert werden können, stellt dies in der Raumfahrt eine erhebliche Schwierigkeit dar. Ein wesentlicher Grund hierfür ist die komplexe, monolithische Architektur aktueller Satelliten, bei der ein Austausch einzelner Komponenten, beispielsweise durch In-Orbit-Wartung, nicht vorgesehen ist [181].

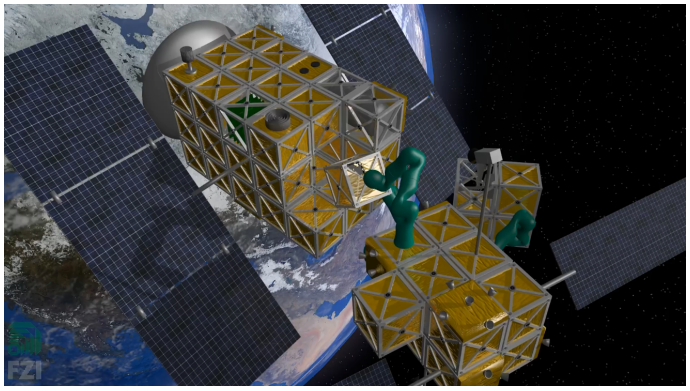


Abbildung C.1.: Rendering einer In-Orbit-Wartungsmission.

Das Konzept von iBOSS (Intelligent Building Blocks for On-Orbit Servicing and Assembly) [182] adressiert genau diese Problematik. Satelliten werden in diesem Ansatz nicht mehr monolithisch gefertigt, sondern aus modularen Funktionsbausteinen zusammengesetzt. Abbildung C.1 zeigt das Rendering einer möglichen In-Orbit-Wartungsmission. Die verwendeten Funktionsbausteine weisen eine kubische Form auf und verfügen über eine oder mehrere standardisierte Schnittstellen, die sogenannten iSSIs (Intelligent Space System Interface). Diese dienen sowohl der mechanischen Kopplung mit anderen Bausteinen als auch dem Austausch von Wärme, Energie und Daten. Ein entsprechender Funktionsbaustein ist in Abbildung C.2 dargestellt. Ein Satellitensystem, das auf dieser modularen Architektur basiert, ermöglicht den gezielten Austausch defekter Module und verlängert somit die operative Lebensdauer des Gesamtsystems signifikant.

Abbildung C.2.: Ein einzelner Funktionsbaustein[1]

[1]Bildquelle: www.iboss.space (Zugegriffen am 16.02.2026)

D. Blender

Blender [182] ist eine freies und quelloffenes 3D-Grafiprogramm, das unter der GNU General Public License steht und den gesamten Prozess der 3D-Modell-Erstellung abdeckt. Der Funktionsumfang erstreckt sich von der geometrischen Modellierung, dem Sculpting und der Texturierung bis hin zur Animation und dem Rendering mittels der leistungsstarken Engines *Cycles* und *Eevee*. Ein wesentlicher Bestandteil der Softwarearchitektur ist die integrierte Physik- und Simulationsumgebung. Für die Simulation von festen Objekten greift Blender auf Starrkörper-Dynamiken (engl. Rigid Body Dynamics) zurück, die auf der weitverbreiteten Bullet Physics Library basieren und grundlegende physikalische Interaktionen wie Kollisionen und Schwerkraft berechnen. Ergänzend stehen spezialisierte Solver für deformierbare Körper (engl. Soft Bodies) sowie komplexe Fluid- und Rauchsysteme zur Verfügung.

Eine signifikante Weiterentwicklung der letzten Jahre stellt das node-basierte System *Geometry Nodes* dar, welches eine nicht-destruktive und prozedurale Generierung komplexer Geometrien ermöglicht. Parallel dazu erlaubt der Shader-Editor die Erstellung physikalisch korrekter Materialien, was essenziell für das Erreichen eines hohen Grades an Fotorealismus ist. Durch die umfangreiche Python-API lassen sich sämtliche dieser Parameter skriptbasiert steuern, was auch den Betrieb im *Headless-Modus* auf Servern ohne grafische Benutzeroberfläche erlaubt. Dies macht Blender zu einem besonders wertvollen Werkzeug in der Robotik und KI-Forschung, da es die massenhafte Erzeugung von annotierten, synthetischen Trainingsdaten (engl. Synthetic Data Generation) automatisiert. Zudem profitiert die Software von einem modularen Aufbau, der es Forschern ermöglicht, spezifische Funktionalitäten durch eigene Add-ons nahtlos in die bestehende Pipeline zu integrieren.

Akronyme

ALB Montagelinienaufteilung / Assembly Line Balancing

AP Montageplanung / Assembly Planning

APP Montagepfadplanung / Assembly Path Planning

ASP Montagereihenfolgenplanung / Assembly Sequence Planning

CAD Computer-Aided Design

CAM Computer-Aided Manufacturing

CNN Faltungsnetze / Convolutional Neuronal Networks

CoD Curse of Dimensionality

CoT Chain-of-Though

DP Demontageplanung / Disassembly Planning

FPN Feature-Pyramid-Network-Decoder

GA Genetischer Algorithmus

GRU Gated Recurrent Units

IoU Intersection over Union

JSON JavaScript Object Notation

KG Wissensgraph / Knowledge Graph

KI Künstliche Intelligenz

LLM Large Language Models

LM Language Modeling

MAE Mittlerer absoluter Fehler / Mean Absoulte Error

ML Maschinelles Lernen

MLP Mehrschichtiges Perzeptron / Multi Layer Percepetron

MRK Mensch-Roboter-Kollaboration

NLM Neuronales Sprachmodell / Neural Language Model

NLP Natural Language Processing

NMS Non-Maximum Suppression

OCC OpenCascade

PLM Vortrainiertes Sprachmodell / Pre-trained Language Model

R-CNN Region-based Convolutional Neural Networks

ReLU Rectified Linear Unit

ROS2 Robot Operating System 2

Seq2Seq Sequenz-zu-Sequenz

SLM Statistisches Sprachmodell / Statistical Language Model

STEP Standard for the Exchange of Product model data

VLA Vision-Language-Action-Modelle

VLM Vision-Language Model

Glossar

Affordanz / Affordance Die visuell erkennbare Eigenschaft eines Bauteils, die anzeigt, wie und wo es montiert werden kann (z. B. Schraubfläche).

Ankerbox / Anchor Box Vordefinierte Referenzrahmen in der Objekterkennung, die dem Netz helfen, Objekte verschiedener Skalierung zu finden.

Basisnetz / Backbone Der Teil eines neuronalen Netzes (z. B. ResNet), der für die Merkmalsextraktion zuständig ist.

Batch Begriff aus dem Training von neuronalen Netzen, der definiert, wie viele Datenpunkte gleichzeitig verarbeitet werden.

Begrenzungsrahmen / Bounding Box Ein Rechteck (2D) oder Quader (3D), das ein Objekt vollständig umschließt.

Computer-Aided Design Umfasst die Verwendung von Software zur Erstellung, Modifikation und Analyse von zwei- oder dreidimensionalen digitalen Modellen technischer Objekte.

Computer-Aided Manufacturing Bezeichnet den Einsatz von Software zur Steuerung von Werkzeugmaschinen und zur computergestützten Fertigung von Bauteilen auf Basis von digitalen Modellen.

Curse of Dimensionality Der Fluch der Dimensionalität oder Curse of Dimensionality beschreibt das Phänomen, dass Algorithmen und mathematische Konzepte, die in niedrigen Dimensionen (2D oder 3D) gut funktionieren, in hochdimensionalen Räumen plötzlich ineffizient werden oder gar nicht mehr funktionieren.

Encoder-Decoder-Architektur Ein grundlegendes Architekturprinzip, bei dem Daten erst in eine komprimierte Repräsentation überführt (Encoder) und dann wieder rekonstruiert oder in eine andere Form gebracht werden (Decoder).

Epoche Begriff aus dem Training von neuronalen Netzen, der definiert, wie oft der Datensatz durchlaufen wird.

Foundation Model Ein großes, auf riesigen Datenmengen trainiertes KI-Modell (wie GPT-4 oder Qwen), das für verschiedene nachgelagerte Aufgaben (Downstream Tasks) adaptiert werden kann.

Genetischer Algorithmus Ein Optimierungsverfahren, das Prinzipien der Evolution (Selektion, Crossover, Mutation) nutzt.

Grounding Die Verankerung von generierten Informationen in der Realität oder in verifizierbaren Daten (in deiner Arbeit: durch den Wissensgraphen und die visuelle Prüfung mittels VLM), um Halluzinationen zu vermeiden.

Halluzination Das Phänomen, dass generative KI-Modelle falsche oder nicht existente Fakten plausibel darstellen (z. B. Bauteile erfinden, die nicht existieren).

Intersection over Union Ein Maß für die Überlappung zweier Flächen/Volumen, genutzt zur Bewertung der Genauigkeit von Bounding Boxes.

JavaScript Object Notation Eein kompaktes, textbasiertes Datenformat zum einfachen Austausch strukturiert gespeicherter Daten zwischen Anwendungen.

Large Language Models Auf Deep Learning basierende KI-Modelle, die darauf trainiert wurden, menschliche Sprache zu verstehen, zu generieren und komplexe Aufgaben in natürlicher Sprache zu bearbeiten.

Latenter Raum / Latent Space Ein abstrakter, meist niedrigdimensionaler Vektorraum, in dem Merkmale repräsentiert werden.

MoveIt2 Eine fortgeschrittene Software-Plattform für die Manipulationsplanung in ROS2, die Funktionen für die Bewegungsplanung, Kinematik und Kollisionsvermeidung von Roboterarmen bietet.

Non-Maximum Suppression Ein Verfahren, um mehrfache Detektionen desselben Objekts zu filtern und nur die beste zu behalten.

OpenCascade Ein leistungsfähiges Open-Source-Framework (SDK) für die geometrische 3D-Modellierung, das insbesondere für die Entwicklung von CAD-, CAM- und CAE-Anwendungen genutzt wird.

Prompt / Prompt Engineering Die Eingabeaufforderung an ein KI-Modell und die Kunst, diese so zu gestalten, dass das Modell das gewünschte Ergebnis liefert (z. B. durch Prefix-Prompting, Few-Shot).

Robot Operating System 2 Ein quelloffenes Framework, das Werkzeuge und Bibliotheken zur modularen Entwicklung komplexer Roboteranwendungen sowie zur Kommunikation zwischen verschiedenen Softwarekomponenten bereitstellt.

Sequenz-zu-Sequenz Ein Modellansatz, der eine Eingabesequenz (z. B. Bauteile) in eine Ausgabesequenz (z. B. Montagereihenfolge) übersetzt.

Standard for the Exchange of Product model data Das genutzte Dateiformat für CAD-Daten (AP 214).

Token Die grundlegende Einheit (Wortteil), in der LLMs Text verarbeiten. Wichtig für deine Diskussion über Kontextlänge und Rechenaufwand.

Verlustfunktion / Loss Function Die mathematische Funktion (z. B. Kreuzentropie, Focal-Loss), die misst, wie stark die Vorhersage vom Soll-Wert abweicht.

Vision-Language Model Ein multimodales KI-Modell, das sowohl Bilder als auch Text verarbeiten kann.

Vision-Language-Action-Modelle Eine Weiterentwicklung von multimodalen KI-Systemen, die visuelle Informationen und sprachliche Anweisungen direkt in konkrete Handlungsbefehle für Roboter übersetzen.

Voxel Das 3D-Äquivalent zum Pixel.

Wissensgraph / Knowledge Graph Eine strukturierte Darstellung von Wissen in Form von Knoten (Bauteile) und Kanten (Beziehungen/Kollisionen).

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] P. Dash, B. K. Sahu, und M. Dash, "Generation of robotized assembly order using liaison and matrix methods: A comparative study," 2024.

[2] L. A. Bewoor, V. C. Prakash, und S. U. Sapkal, "Production scheduling optimization in foundry using hybrid particle swarm optimization algorithm," *Procedia Manufacturing*, vol. 22, pp. 57–64, 2018. 11th International Conference Interdisciplinarity in Engineering, INTER-ENG 2017, 5-6 October 2017, Tirgu Mures, Romania.

[3] G. Lanza, F. Klenk, M. Martin, O. Brützel, und R. Hörsting, "Sonderforschungsbereich 1574: Kreislauffabrik für das ewige innovative produkt: Integrierte lineare und zirkuläre produktion mittels hochvernetztem produkt-produktions-codesign," *Zeitschrift für wirtschaftlichen Fabrikbetrieb*, vol. 118, no. 12, pp. 820–825, 2023.

[4] A. H. Fritz und G. Schulze, *Fertigungstechnik*. Springer, 2001.

[5] B. Lotter und H.-P. Wiendahl, *Montage in der industriellen Produktion: Ein Handbuch für die Praxis*. Springer-Verlag, 2013.

[6] H. Dubbel, *DUBBEL: Taschenbuch für den Maschinenbau*. Springer-Verlag, 2013.

[7] S. Ghandi und E. Masehian, "Review and taxonomies of assembly and disassembly path planning problems and approaches," *Computer-Aided Design*, vol. 67, pp. 58–86, 2015.

[8] L. Kavraki, J.-C. Latombe, und R. H. Wilson, "On the complexity of assembly partitioning," *Information Processing Letters*, vol. 48, no. 5, pp. 229–235, 1993.

[9] H. Lv und C. Lu, "An assembly sequence planning approach with a discrete particle swarm optimization algorithm," *The international journal of advanced manufacturing technology*, vol. 50, no. 5, pp. 761–770, 2010.

[10] M. R. Bahubalendruni und B. B. Biswal, "A review on assembly sequence generation and its automation," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 230, no. 5, pp. 824–838, 2016.

[11] T. Cao und A. C. Sanderson, "And/or net representation for robotic task sequence planning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 28, no. 2, pp. 204–218, 2002.

[12] C. Champatiray, M. R. Bahubalendruni, R. N. Mahapatra, und D. Mishra, "Optimal robotic assembly sequence planning with tool integrated assembly interference matrix," *AI EDAM*, vol. 37, p. e4, 2023.

[13] H. You, Y. Ye, T. Zhou, Q. Zhu, und J. Du, "Robot-enabled construction assembly with automated sequence planning based on chatgpt: Robogpt," *Buildings*, vol. 13, no. 7, p. 1772, 2023.

[14] T. Vossen, M. Ball, A. Lotem, und D. Nau, "Applying integer programming to ai planning," *The Knowledge Engineering Review*, vol. 15, no. 1, pp. 85–100, 2000.

[15] J. P. Garcia-Sabater, J. Maheut, und J. J. Garcia-Sabater, "A two-stage sequential planning scheme for integrated operations planning and scheduling system using milp: the case of an engine assembler," *Flexible services and manufacturing journal*, vol. 24, no. 2, pp. 171–209, 2012.

[16] N. Sundström, O. Wigström, P. Falkman, und B. Lennartson, "Optimization of operation sequences using constraint programming," *IFAC Proceedings Volumes*, vol. 45, no. 6, pp. 1580–1585, 2012.

[17] Z.-F. LIU, D. Hu, X. GAO, und J.-D. ZHANG, "Product disassembly sequence planning based on greedy algorithm," *China Mechanical Engineering*, vol. 22, no. 18, p. 2162, 2011.

[18] H. Shan, S. Li, D. Gong, und P. Lou, "Genetic simulated annealing algorithm-based assembly sequence planning," *International Technology and Innovation Conference*, Nov. 2006.

[19] H.-E. Tseng, C.-C. Chang, S.-C. Lee, und Y.-M. Huang, "A block-based genetic algorithm for disassembly sequence planning," *Expert Systems with Applications*, vol. 96, pp. 492–505, 2018.

[20] Z. Han, Y. Wang, und D. Tian, "Ant colony optimization for assembly sequence planning based on parameters optimization," *Frontiers of Mechanical Engineering*, vol. 16, no. 2, pp. 393–409, 2021.

[21] S. Ghandi und E. Masehian, "Review and taxonomies of assembly and disassembly path planning problems and approaches," *Computer-Aided Design*, vol. 67-68, pp. 58–86, 2015.

[22] M. F. F. Rashid, W. Hutabarat, und A. Tiwari, "A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches," *The International Journal of Advanced Manufacturing Technology*, vol. 59, pp. 335–349, Mar. 2012. Publisher: Springer Science and Business Media LLC.

[23] S. Rakshit und S. Akella, "The influence of motion paths and assembly sequences on the stability of assemblies," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 615–627, 2014.

[24] L. H. De Mello und A. C. Sanderson, "And/or graph representation of assembly plans," *IEEE Transactions on robotics and automation*, vol. 6, no. 2, pp. 188–199, 1990.

[25] L. Haicheng, L. Yuan, Y. Jianfeng, und Z. Yuan, "Path planning algorithm for assembly of complex product based on v-map and ant colony optimization algorithm," in *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, vol. 5, pp. V5–398, IEEE, 2010.

[26] Q. Luo und J. Xiao, "Haptic rendering involving an elastic tube for assembly simulations," in *(ISATP 2005). The 6th IEEE International Symposium on Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing, 2005.*, pp. 53–59, IEEE, 2005.

[27] O. O. Martins, A. A. Adekunle, O. M. Olaniyan, und B. O. Bolaji, "An improved multi-objective a-star algorithm for path planning in a large workspace: Design, implementation, and evaluation," *Scientific African*, vol. 15, p. e01068, 2022.

[28] H. Chang und T.-Y. Li, "Assembly maintainability study with motion planning," in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 1, pp. 1012–1019, IEEE, 1995.

[29] T. Hermansson, R. Bohlin, J. S. Carlson, und R. Söderberg, "Automatic assembly path planning for wiring harness installations," *Journal of manufacturing systems*, vol. 32, no. 3, pp. 417–422, 2013.

[30] J. S. Carlson, D. Spensieri, R. Söderberg, R. Bohlin, und L. Lindkvist, "Non-nominal path planning for robust robotic assembly," *Journal of manufacturing systems*, vol. 32, no. 3, pp. 429–435, 2013.

[31] L. Zhang, X. Huang, Y. J. Kim, und D. Manocha, "D-plan: Efficient collision-free path computation for part removal and disassembly," *Computer-Aided Design and Applications*, vol. 5, no. 6, pp. 774–786, 2008.

[32] C. Hui, L. Yuan, und Z. Kai-fu, "Efficient method of assembly sequence planning based on gaaa and optimizing by assembly path feedback for complex product," *The International Journal of Advanced Manufacturing Technology*, vol. 42, pp. 1187–1204, 2009.

[33] N. Ladeveze, J.-Y. Fourquet, und B. Puel, "Interactive path planning for haptic assistance in assembly tasks," *Computers & Graphics*, vol. 34, no. 1, pp. 17–25, 2010.

[34] X. Jin, T. Zhang, und H. Yang, "An analysis of the assembly path planning of decelerator based on virtual technology," *Physics Procedia*, vol. 25, pp. 170–175, 2012.

[35] D. Halperin, J.-C. Latombe, und R. H. Wilson, "A general framework for assembly planning: The motion space approach," in *Proceedings of the fourteenth annual symposium on Computational geometry*, pp. 9–18, 1998.

[36] G. Ehrenstein und G. Ahlers-Hestermann, *Handbuch Kunststoff-Verbindungstechnik*. Hanser, 2004.

[37] "DIN 8580 Fertigungsverfahren - Begriffe, Einteilung," Dez 2022.

[38] International Organization for Standardization, "Industrial automation systems and integration — product data representation and exchange — part 1: Overview and fundamental principles," 01 2024.

[39] International Organization for Standardization, "Industrial automation systems and integration — product data representation and exchange — part 214: Application protocol: Core data for automotive mechanical design processes," 2010. Withdrawn, superseded by ISO 10303-242:2014.

[40] International Organization for Standardization, "Industrial automation systems and integration — product data representation and exchange — part 242: Application protocol: Managed model-based 3d engineering," 2025.

[41] I. Goodfellow, Y. Bengio, A. Courville, und Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[42] C. Eichmann, S. Bellmann, N. Hügel, L.-E. Enslin, C. Plasberg, G. Heppner, A. Roennau, und R. Dillmann, "Lauron vi: A six-legged robot for dynamic walking," *arXiv preprint arXiv:2508.07689*, 2025.

[43] A. Thengade und R. Dondal, "Genetic algorithm–survey paper," in *MPGI national multi conference*, pp. 7–8, Citeseer, 2012.

[44] K. Janocha und W. M. Czarnecki, "On loss functions for deep neural networks in classification," *arXiv preprint arXiv:1702.05659*, 2017.

[45] Y. Ho und S. Wookey, "The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2019.

[46] T.-Y. Lin, P. Goyal, R. Girshick, K. He, und P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[47] M. D. Zeiler und R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014.

[48] V. Dumoulin und F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[49] K. He, X. Zhang, S. Ren, und J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[50] G. Huang, Z. Liu, L. Van Der Maaten, und K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[51] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, und L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[52] A. E. Orhan und X. Pitkow, "Skip connections eliminate singularities," *arXiv preprint arXiv:1701.09175*, 2017.

[53] V. Badrinarayanan, A. Kendall, und R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[54] S. Ren, K. He, R. Girshick, und J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[55] J. Redmon, S. Divvala, R. Girshick, und A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[56] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, und J.-R. Wen, "A Survey of Large Language Models," Mar. 2025. arXiv:2303.18223 [cs].

[57] R. Bommasani und D. A. Hudson, "On the opportunities and risks of foundation models," 2022.

[58] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, und I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[59] J. Devlin, M.-W. Chang, K. Lee, und K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

[60] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heintz, und D. Roth, "Recent Advances in Natural Language Processing via Large Pre-trained Language Models: A Survey," *ACM Computing Surveys*, vol. 56, pp. 1–40, Feb. 2024.

[61] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, und L. Zettlemoyer, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," Oct. 2019. arXiv:1910.13461 [cs].

[62] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, und D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[63] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, *et al.*, "Training compute-optimal large language models," *arXiv preprint arXiv:2203.15556*, 2022.

[64] S. Schulhoff, M. Ilie, N. Balepur, K. Kahadze, A. Liu, C. Si, Y. Li, A. Gupta, H. Han, S. Schulhoff, *et al.*, "The prompt report: A systematic survey of prompting techniques," *arXiv preprint arXiv:2406.06608*, vol. 5, 2024.

[65] V. B. Parthasarathy, A. Zafar, A. Khan, und A. Shahid, "The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities," Oct. 2024. arXiv:2408.13296 [cs].

[66] Z. Li, X. Wu, H. Du, F. Liu, H. Nghiem, und G. Shi, "A Survey of State of the Art Large Vision Language Models: Alignment, Benchmark, Evaluations and Challenges," Apr. 2025. arXiv:2501.02189 [cs].

[67] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[68] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, pp. 8748–8763, PmLR, 2021.

[69] J. Li, D. Li, C. Xiong, und S. Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *International conference on machine learning*, pp. 12888–12900, PMLR, 2022.

[70] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, und T. Duerig, "Scaling up visual and vision-language representation learning with noisy text supervision," in *International conference on machine learning*, pp. 4904–4916, PMLR, 2021.

[71] B. Li, Y. Zhang, D. Guo, R. Zhang, F. Li, H. Zhang, K. Zhang, P. Zhang, Y. Li, Z. Liu, *et al.*, "Llava-onevision: Easy visual task transfer," *arXiv preprint arXiv:2408.03326*, 2024.

[72] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, *et al.*, "The llama 3 herd of models," *arXiv e-prints*, pp. arXiv–2407, 2024.

[73] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[74] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, *et al.*, "Flamingo: a visual language model for few-shot learning," *Advances in neural information processing systems*, vol. 35, pp. 23716–23736, 2022.

[75] Z. Peng, W. Wang, L. Dong, Y. Hao, S. Huang, S. Ma, und F. Wei, "Kosmos-2: Grounding multimodal large language models to the world," *arXiv preprint arXiv:2306.14824*, 2023.

[76] H. Lin, X. Cheng, X. Wu, und D. Shen, "Cat: Cross attention in vision transformer," in *2022 IEEE international conference on multimedia and expo (ICME)*, pp. 1–6, IEEE, 2022.

[77] J. Lu, D. Batra, D. Parikh, und S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," *Advances in neural information processing systems*, vol. 32, 2019.

[78] A. v. d. Oord, Y. Li, und O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[79] A. Rohrbach, L. A. Hendricks, K. Burns, T. Darrell, und K. Saenko, "Object hallucination in image captioning," *arXiv preprint arXiv:1809.02156*, 2018.

[80] X. Wang, J. Pan, L. Ding, und C. Biemann, "Mitigating hallucinations in large vision-language models with instruction contrastive decoding," *arXiv preprint arXiv:2403.18715*, 2024.

[81] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, und A. Chadha, "A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications," Mar. 2025. arXiv:2402.07927 [cs].

[82] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, und G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM computing surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[83] B. Chen, Z. Zhang, N. Langrené, und S. Zhu, "Unleashing the potential of prompt engineering for large language models," *Patterns*, p. 101260, May 2025. arXiv:2310.14735 [cs].

[84] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, und X. Wu, "Unifying Large Language Models and Knowledge Graphs: A Roadmap," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, pp. 3580–3599, July 2024. arXiv:2306.08302 [cs].

[85] D. Vrandečić und M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[86] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, und J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1247–1250, 2008.

[87] R. Speer, J. Chin, und C. Havasi, "Conceptnet 5.5: An open multilingual graph of general knowledge," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, 2017.

[88] J. D. Hwang, C. Bhagavatula, R. Le Bras, J. Da, K. Sakaguchi, A. Bosselut, und Y. Choi, "(comet-) atomic 2020: On symbolic and neural commonsense knowledge graphs," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, pp. 6384–6392, 2021.

[89] O. Bodenreider, "The unified medical language system (umls): integrating biomedical terminology," *Nucleic acids research*, vol. 32, no. suppl_1, pp. D267–D270, 2004.

[90] S. Ferrada, B. Bustos, und A. Hogan, "Imgpedia: a linked dataset with content-based analysis of wikimedia images," in *International Semantic Web Conference*, pp. 84–93, Springer, 2017.

[91] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, und Q. Liu, "Ernie: Enhanced language representation with informative entities," *arXiv preprint arXiv:1905.07129*, 2019.

[92] X. Wang, T. Gao, Z. Zhu, Z. Zhang, Z. Liu, J. Li, und J. Tang, "Kepler: A unified model for knowledge embedding and pre-trained language representation," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 176–194, 2021.

[93] J. J. Gibson, *The ecological approach to visual perception: classic edition*. Psychology press, 2014.

[94] E. Broermann, "Koffka k.: Die grundlagen der psychischen entwicklung. eine einführung in die kinderpsychologie. 2. verbesserte auflage. osterwieck 1925.," *Vierteljahrsschrift für wissenschaftliche Pädagogik*, vol. 3, no. 1, pp. 139–140, 1927.

[95] E. Schwanenberg, "Lewin, kurt: Grundzüge der topologischen psychologie. übertragen und herausgegeben von r. falk und f. winnefeld, unter mitarbeit," *Psyche*, vol. 28, no. 11, pp. 1048–1048, 1974.

[96] P. A. Heslin, U.-C. Klehe, und L. A. Keating, "The sage encyclopedia of industrial and organizational psychology, 2nd," 2017.

[97] E. von Leon, "Spaß machen–: Wie allein durch das partizipative lesen von spielbilderbüchern komik konstruiert wird," *Jahrbuch der Gesellschaft für Kinder-und Jugendliteraturforschung*, pp. 81–93, 2024.

[98] H. Fleisch, C. Mecking, und E. Steinsdörfer, "Gamification4good," 2018.

[99] R. Detry, D. Kraft, O. Kroemer, L. Bodenhagen, J. Peters, N. Krüger, und J. Piater, "Learning grasp affordance densities," *Paladyn*, vol. 2, pp. 1–17, 2011.

[100] M. Wang, R. Luo, A. Ö. Önol, und T. Padir, "Affordance-based mobile robot navigation among movable obstacles," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2734–2740, IEEE, 2020.

[101] T. E. Horton, A. Chakraborty, und R. S. Amant, "Affordances for robots: a brief survey," *AVANT. Pismo Awangardy Filozoficzno-Naukowej*, vol. 2, pp. 70–84, 2012.

[102] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, und H. Liu, "Graph learning: A survey," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 2, pp. 109–127, 2021.

[103] L. Ma, J. Gong, H. Xu, H. Chen, H. Zhao, W. Huang, und G. Zhou, "Planning assembly sequence with graph transformer," *arXiv preprint arXiv:2210.05236*, 2022.

[104] J. Huang, G. Zhan, Q. Fan, K. Mo, L. Shao, B. Chen, L. Guibas, und H. Dong, "Generative 3D Part Assembly via Dynamic Graph Learning," Dec. 2020. arXiv:2006.07793 [cs].

[105] M. Atad, J. Feng, I. Rodríguez, M. Durner, und R. Triebel, "Efficient and Feasible Robotic Assembly Sequence Planning via Graph Representation Learning," July 2023. arXiv:2303.10135 [cs].

[106] M. Hansjosten und J. Fleischer, "Disassembly graph generation and sequence planning based on 3d models for the disassembly of electric motors," in *Congress of the German Academic Association for Production Technology*, pp. 448–457, Springer, 2023.

[107] D. Gerhard, J. Rolf, J. L. Siewert, und P. Trentsios, "Machine learning methods for (dis-) assembly sequence planning-a systematic literature review," *International Journal of Advances in Production Research*, vol. 1, no. 1, pp. 83–98, 2024.

[108] M. Neves, M. Vieira, und P. Neto, "A study on a q-learning algorithm application to a manufacturing assembly problem," *Journal of Manufacturing Systems*, vol. 59, pp. 426–440, 2021.

[109] J. De Winter, I. EI Makrini, G. Van de Perre, A. Nowé, T. Verstraten, und B. Vanderborght, "Autonomous assembly planning of demonstrated skills with reinforcement learning in simulation," *Autonomous Robots*, vol. 45, no. 8, pp. 1097–1110, 2021.

[110] M. Zhao, X. Guo, X. Zhang, Y. Fang, und Y. Ou, "Aspw-drl: assembly sequence planning for workpieces via a deep reinforcement learning approach," *Assembly Automation*, vol. 40, no. 1, pp. 65–75, 2020.

[111] D. Antonelli und K. Aliev, "Robust assembly task assignment in human robot collaboration as a markov decision process problem," *Procedia CIRP*, vol. 112, pp. 174–179, 2022.

[112] J. Yin, M. Chen, und T. Zhang, "Optimization algorithm for cooperative assembly sequence of truss structure based on reinforcement learning," in *International Conference on Intelligent Robotics and Applications*, pp. 474–484, Springer, 2021.

[113] M. Frisch, J. Baumgärtner, I. Heider, A. Puchta, und J. Fleischer, "Efficient deployment of machine learning models in manufacturing and industrial environments using ros," *Procedia CIRP*, vol. 130, pp. 188–193, 2024.

[114] Y. Tian, J. Xu, Y. Li, J. Luo, S. Sueda, H. Li, K. D. Willis, und W. Matusik, "Assemble them all: Physics-based planning for generalizable assembly by disassembly," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 6, pp. 1–11, 2022.

[115] S. Ben Said und N. Aifaoui, "Assembly path planning with collision avoidance," in *International Conference Design and Modeling of Mechanical Systems*, pp. 44–49, Springer, 2023.

[116] Y. Tian, K. D. Willis, B. Al Omari, J. Luo, P. Ma, Y. Li, F. Javid, E. Gu, J. Jacob, S. Sueda, *et al.*, "Asap: Automated sequence planning for complex robotic assembly with physical feasibility," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4380–4386, IEEE, 2024.

[117] J. Baumgärtner, M. Hansjosten, D. Schönhofen, und I. J. Fleischer, "Pybullet industrial: A process-aware robot simulation," *Journal of Open Source Software*, vol. 8, no. 85, p. 5174, 2023.

[118] M. Hansjosten und J. Fleischer, "Towards autonomous adaptive disassembly of permanent-magnet synchronous motors with industrial robots," *Manufacturing Letters*, vol. 35, pp. 1336–1346, 2023.

[119] K. Tariki, T. Kiyokawa, T. Nagatani, J. Takamatsu, und T. Ogasawara, "Generating complex assembly sequences from 3d cad models considering insertion relations," *Advanced Robotics*, vol. 35, no. 6, pp. 337–348, 2021.

[120] T. Kiyokawa, I. Rodriguez, K. Nottensteiner, P. Lehner, T. Eiband, M. A. Roa, und K. Harada, "Cad-informed uncertainty-aware sequence and motion planning for robotic assembly," in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, pp. 418–425, IEEE, 2024.

[121] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, und P. Abbeel, "Learning robotic assembly from cad," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3524–3531, IEEE, 2018.

[122] S. Zobov, F. Chervinskii, A. Rybnikov, D. Petrov, und K. Vendidandi, "Auto-Assembly: a framework for automated robotic assembly directly from CAD," Jan. 2023. arXiv:2301.02643 [cs].

[123] K. D. D. Willis, P. K. Jayaraman, H. Chu, Y. Tian, Y. Li, D. Grandi, A. Sanghi, L. Tran, J. G. Lambourne, A. Solar-Lezama, und W. Matusik, "JoinABLe: Learning Bottom-up Assembly of Parametric CAD Joints," Apr. 2022. arXiv:2111.12772 [cs].

[124] L. Zhang, B. Le, N. Akhtar, S.-K. Lam, und T. Ngo, "Large language models for computer-aided design: A survey," *arXiv preprint arXiv:2505.08137*, 2025.

[125] X. Li, Y. Sun, und Z. Sha, "Llm4cad: Multimodal large language models for three-dimensional computer-aided design generation," *Journal of Computing and Information Science in Engineering*, vol. 25, no. 2, p. 021005, 2025.

[126] D. Mallis, A. S. Karadeniz, S. Cavada, D. Rukhovich, N. Foteinopoulou, K. Cherenkova, A. Kacem, und D. Aouada, "Cad-assistant: Tool-augmented vllms as generic cad task solvers," *arXiv preprint arXiv:2412.13810*, 2024.

[127] Y. You, M. A. Uy, J. Han, R. Thomas, H. Zhang, S. You, und L. Guibas, "Img2cad: Reverse engineering 3d cad models from images through vlm-assisted conditional factorization," *arXiv preprint arXiv:2408.01437*, 2024.

[128] J. Xu, Z. Zhao, C. Wang, W. Liu, Y. Ma, und S. Gao, "Cad-mllm: Unifying multimodality-conditioned cad generation with mllm," 2025.

[129] V. Liu, J. Vermeulen, G. Fitzmaurice, und J. Matejka, "3dall-e: Integrating text-to-image ai in 3d design workflows," in *Proceedings of the 2023 ACM designing interactive systems conference*, pp. 1955–1977, 2023.

[130] C. Kienle, B. Alt, D. Katic, R. Jäkel, und J. Peters, "QueryCAD: Grounded Question Answering for CAD Models," Mar. 2025. arXiv:2409.08704 [cs].

[131] Z. Zhang, S. Sun, W. Wang, D. Cai, und J. Bian, "Flexcad: Unified and versatile controllable cad generation with fine-tuned large language models," *arXiv preprint arXiv:2411.05823*, 2024.

[132] M. U. Din, W. Akram, L. S. Saoud, J. Rosell, und I. Hussain, "Vision language action models in robotic manipulation: A systematic review," *arXiv preprint arXiv:2507.10672*, 2025.

[133] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[134] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, und P. Florence, "PaLM-E: An Embodied Multimodal Language Model," Mar. 2023. arXiv:2303.03378 [cs].

[135] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning*, pp. 2165–2183, PMLR, 2023.

[136] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, und L. Fan, "Vima: General robot manipulation with multimodal prompts," *arXiv preprint arXiv:2210.03094*, vol. 2, no. 3, p. 6, 2022.

[137] D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, *et al.*, "Octo: An open-source generalist robot policy," *arXiv preprint arXiv:2405.12213*, 2024.

[138] R. Patel und E. Pavlick, "Mapping language models to grounded conceptual spaces," in *International conference on learning representations*, 2022.

[139] W. Huang, F. Xia, D. Shah, D. Driess, A. Zeng, Y. Lu, P. Florence, I. Mordatch, S. Levine, K. Hausman, *et al.*, "Grounded decoding: Guiding text generation with grounded models for embodied agents," *Advances in Neural Information Processing Systems*, vol. 36, pp. 59636–59661, 2023.

[140] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, und N. Suenderhauf, "Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning," *arXiv preprint arXiv:2307.06135*, 2023.

[141] J. Gao, B. Sarkar, F. Xia, T. Xiao, J. Wu, B. Ichter, A. Majumdar, und D. Sadigh, "Physically grounded vision-language models for robotic manipulation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12462–12469, IEEE, 2024.

[142] B. Wang, J. Zhang, S. Dong, I. Fang, und C. Feng, "Vlm see, robot do: Human demo video to robot action plan via vision language model," *arXiv preprint arXiv:2410.08792*, 2024.

[143] S. Li, Z. Yan, Z. Wang, und Y. Gao, "Vlm-msgraph: Vision language model-enabled multi-hierarchical scene graph for robotic assembly," *Robotics and Computer-Integrated Manufacturing*, vol. 94, p. 102978, 2025.

[144] Y. Hu, F. Lin, T. Zhang, L. Yi, und Y. Gao, "Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning," *arXiv preprint arXiv:2311.17842*, 2023.

[145] M. Skreta, Z. Zhou, J. L. Yuan, K. Darvish, A. Aspuru-Guzik, und A. Garg, "Replan: Robotic replanning with perception and language models," *arXiv preprint arXiv:2401.04157*, 2024.

[146] A. Mei, G.-N. Zhu, H. Zhang, und Z. Gan, "Replanvlm: Replanning robotic tasks with visual language models," *IEEE Robotics and Automation Letters*, 2024.

[147] T.-T. Do, A. Nguyen, und I. Reid, "Affordancenet: An end-to-end deep learning approach for object affordance detection," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 1–5, IEEE, 2018.

[148] K. Simonyan und A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[149] A. Nguyen, D. Kanoulas, D. G. Caldwell, und N. G. Tsagarakis, "Object-based affordances detection with convolutional neural networks and dense conditional random fields," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5908–5915, IEEE, 2017.

[150] J. Dai, Y. Li, K. He, und J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, pp. 379–387, 2016.

[151] C. Sutton und A. McCallum, "An introduction to conditional random fields for relational learning," *Introduction to statistical relational learning*, vol. 2, pp. 93–128, 2006.

[152] A. Nguyen, D. Kanoulas, D. G. Caldwell, und N. G. Tsagarakis, "Detecting object affordances with convolutional neural networks," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2765–2770, IEEE, 2016.

[153] X. Zhao, Y. Cao, und Y. Kang, "Object affordance detection with relationship-aware network," *Neural Computing and Applications*, pp. 1–13, 2019.

[154] J. Sawatzky, A. Srikantha, und J. Gall, "Weakly supervised affordance detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2795–2804, 2017.

[155] D. Beßler, M. Pomarlan, und M. Beetz, "Owl-enabled assembly planning for robotic agents," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1684–1692, 2018.

[156] X. Shi, X. Tian, G. Wang, D. Zhao, und M. Zhang, "Semantic-based subassembly identification considering non-geometric structure attributes and assembly process factors," *The International Journal of Advanced Manufacturing Technology*, vol. 110, pp. 439–455, 2020.

[157] V. H. Isume, K. Harada, W. Wan, und Y. Domae, "Using affordances for assembly: Towards a complete craft assembly system," in *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, pp. 2010–2014, IEEE, 2021.

[158] S. Vongbunyong, S. Kara, und M. Pagnucco, "Learning and revision in cognitive robotics disassembly automation," *Robotics and computer-integrated manufacturing*, vol. 34, pp. 79–94, 2015.

[159] F.-J. Chu, R. Xu, und P. A. Vela, "Learning affordance segmentation for real-world robotic manipulation via synthetic images," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1140–1147, 2019.

[160] A. Myers, C. L. Teo, C. Fermüller, und Y. Aloimonos, "Affordance detection of tool parts from geometric features," in *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 1374–1381, IEEE, 2015.

[161] J. Li, Y. Zhu, Z. Tang, J. Wen, M. Zhu, X. Liu, C. Li, R. Cheng, Y. Peng, Y. Peng, *et al.*, "Coa-vla: Improving vision-language-action models via visual-text chain-of-affordance," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9759–9769, 2025.

[162] T. Wang, Z. Zhang, und et al., "Affordance reasoning-based sequence planning manner for human-robot collaborative disassembly," *Journal of Mechanical Engineering*, vol. 60, no. 17, pp. 297–310, 2024.

[163] "Manufacturing processes joining - part 0: General; classification, subdivision, terms and definitions," 09 2009.

[164] J. Riegel, W. Mayer, und Y. van Havre, "Freecad," *Freecadspec2002*, 2016.

[165] B. Falck, D. Falck, und B. Collette, *Freecad [How-To]*. Packt Publishing Ltd, 2012.

[166] R. Wu, Y. Zhuang, K. Xu, H. Zhang, und B. Chen, "Pq-net: A generative part seq2seq network for 3d shapes," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 829–838, 2020.

[167] R. Dey und F. M. Salem, "Gate-variants of gated recurrent unit (gru) neural networks," in *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pp. 1597–1600, IEEE, 2017.

[168] B. Chen, Z. Zhang, N. Langrené, und S. Zhu, "Unleashing the potential of prompt engineering for large language models," *Patterns*, 2025.

[169] A. Zeng, X. Liu, Z. Du, Z. Wang, H. Lai, M. Ding, Z. Yang, Y. Xu, W. Zheng, X. Xia, *et al.*, "Glm-130b: An open bilingual pre-trained model," *arXiv preprint arXiv:2210.02414*, 2022.

[170] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[171] International Organization for Standardization, "Quality management systems — requirements," 2015.

[172] W. Huang, F. Xia, D. Shah, D. Driess, A. Zeng, Y. Lu, P. Florence, I. Mordatch, S. Levine, K. Hausman, *et al.*, "Grounded decoding: Guiding text generation with grounded models for embodied agents," *Advances in Neural Information Processing Systems*, vol. 36, pp. 59636–59661, 2023.

[173] O. Ruiz-Celada, P. Verma, M. Diab, und J. Rosell, "Automating adaptive execution behaviors for robot manipulation," *IEEE access*, vol. 10, pp. 123489–123497, 2022.

[174] B. O. Community, *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[175] L. Flavell, *UV Mapping*, pp. 97–122. Berkeley, CA: Apress, 2010.

[176] K. He, G. Gkioxari, P. Dollár, und R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[177] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, und S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

[178] T. Takikawa, D. Acuna, V. Jampani, und S. Fidler, "Gated-scnn: Gated shape cnns for semantic segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5229–5238, 2019.

[179] K. Lupinetti, F. Giannini, M. Monti, und J.-P. Pernot, "A 3d cad assembly benchmark," in *Eurographics 2019 Workshop on 3D Object Retrieval*, pp. 79–83, Eurographics, 2019.

[180] D. Timmermann *et al.*, "Forschungsprojekt GANResilRob. Gefördert durch das Bundesministerium für Wirtschaft und Energie (BMWE) im Rahmen des Programmes KI-Innovationsprojekte (DE-FR)," 2022-2025.

[181] A. Ellery, J. Kreisel, und B. Sommer, "The case for robotic on-orbit servicing of spacecraft: Spacecraft reliability is a myth," *Acta Astronautica*, vol. 63, no. 5-6, pp. 632–648, 2008.

[182] M. Kortman, S. Ruhl, J. Weise, J. Kreisel, T. Schervan, H. Schmidt, und A. Dafnis, "Building block based iboss approach: fully modular systems with standard interface to enhance future satellites," in *66th International Astronautical Congress (Jerusalem)*, vol. 2, 2015.

Eigene Veröffentlichungen mit Bezug zur Dissertation

Dieses Verzeichnis listet alle Publikationen mit Bezug zur Dissertation, bei denen der Autor dieser Dissertation entweder der Erstautor ist, oder als Co-Autor maßgeblich zu der Veröffentlichung beigetragen hat (in Form von Problemstellung, -lösung, Diskussion oder experimenteller Evaluation).

[Gerlich et al., 2022] Gerlich, R., Flederer, F., Timmermann, D., und Gerlich, R. (2022). Verification of non-conformant software (oss and ai). In *Data Systems in Aerospace (DASIA)*.

[Goerke et al., 2021] Goerke, N., Timmermann, D., und Baumgart, I. (2021). Who controls your robot? an evaluation of ros security mechanisms. In *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*, pages 60–66.

[Heppner et al., 2020] Heppner, G., Mauch, F., Scherzinger, S., Timmermann, D., Becker, P., Ulbrich, S., Rönnau, A., Heiligensetzer, P., und Fürst, F. (2020). *FLA2IR—FLexible Automotive Assembly with Industrial Co-workers*, pages 97–126. Springer International Publishing, Cham.

[Plasberg et al., 2022] Plasberg, C., Nessau, H., Timmermann, D., Eichmann, C., Roennau, A., und Dillmann, R. (2022). Towards distributed real-time capable robotic control using ros2. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pages 2205–2210.

[Plasberg et al., 2023] Plasberg, C., Timmermann, D., Graaf, F., und Rönnau, A. (2023). Verifikation von ki methoden in dezentralen strukturen für new space anwendungen. In *2023 17th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*. ESA.

[Tanev et al., 2018] Tanev, A., Timmermann, D., Buettner, T., Mangler, J., Roennau, A., und Dillmann, R. (2018). Monitoring and controlling of modular meshed networks. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*. ESA.

[Timmermann et al., 2020] Timmermann, D., Friedel, T., Roennau, A., und Dillmann, R. (2020). Assembly status detection based on evaluation of not mounted parts. In *2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA)*, pages 224–229.

[Timmermann et al., 2024a] Timmermann, D., Graaf, F., Heppner, G., Schnell, T., und Dillmann, R. (2024a). Graph transformer based assembly sequence planning for robotic on-orbit assembly. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*. ESA.

[Timmermann et al., 2021a] Timmermann, D., Heid, D., Friedel, T., Roennau, A., und Dillmann, R. (2021a). A hybrid approach for object localization combining mask r-cnn and halcon in an assembly scenario. In *2021 IEEE 8th International Conference on Industrial Engineering and Applications (ICIEA)*, pages 270–276.

[Timmermann et al., 2024b] Timmermann, D., Kocher, C., Heppner, G., Schnell, T., und Dillmann, R. (2024b). Multi classification pareto optimal disassembly planning. In *2024 4th International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pages 01–06. IEEE.

[Timmermann et al., 2024c] Timmermann, D., Maklashevskikh, A., Heppner, G., Schnell, T., und Dillmann, R. (2024c). Using assembly affordances for flexible robotic task planning. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–6. IEEE.

[Timmermann et al., 2024d] Timmermann, D., Plasberg, C., Graaf, F., Rönnau, A., und Dillmann, R. (2024d). Ai-based assembly sequence planning in a robotic on-orbit assembly application. In *2024 10th International Conference on Automation, Robotics and Applications (ICARA)*, pages 69–74. IEEE.

[Timmermann et al., 2021b] Timmermann, D., Tanev, A., Roennau, A., und Dillmann, R. (2021b). Ai4assembly a human-robot collaboration assembly application with ai support. In *2021 IEEE 8th International Conference on Industrial Engineering and Applications (ICIEA)*, pages 186–191.

Studentische und andere Arbeiten mit Bezug zur Dissertation

Dieses Verzeichnis listet studentische Arbeiten mit Bezug zur Dissertation, die durch den Autor dieser Dissertation im Rahmen seiner Forschung ausgeschrieben und betreut wurden. Dies beinhaltet die maßgebliche Vorgabe der Problemstellung, Diskussion der Arbeit sowie Randvorgaben zur Lösung, Visualisierung und experimentelle Evaluation. Dieses Verzeichnis listet weiterhin andere Arbeiten mit Bezug zur Dissertation, die durch den Autor dieser Dissertation im Rahmen seiner Forschung erstellt und durchgeführt wurden. Dies beinhaltet die maßgebliche Konzeption und Erarbeitung der relevanten Ergebnisse.

[FT19] Thomas Friedel und David Timmermann. Kamera-basierte Bestimmung des Montagezustands für eine flexiblere robotische Montage. Bachelorarbeit, KIT Karlsruher Institut für Technologie, Karlsruhe, Germany, 2019.

[KT23] Claudius Kocher und David Timmermann. Rangbasierte Pareto-optimale Vorhersage zur Demontage des nächsten Bauteils mittels neuronaler Netze. Masterarbeit, KIT Karlsruher Institut für Technologie, Karlsruhe, Germany, 2023.

[MT23] Anastasiia Maklashevskikh und David Timmermann. Affordanzenbasierte semantische Planung für die automatisierte robotische Montage. Masterarbeit, KIT Karlsruher Institut für Technologie, Karlsruhe, Germany, 2023.

[PT20] Florian Pohl und David Timmermann. Vollautomatisierte Planung und Erstellung von Montageplänen mit Hilfe von genetischen Algorithmen. Masterarbeit, KIT Karlsruher Institut für Technologie, Karlsruhe, Germany, 2020.

[ST25] Felix Sigurd Schemenz und David Timmermann. From CAD to Robotic Assembly Planning: A Dual-Model Verification Framework Using Large Language and Vision Models. Masterarbeit, KIT Karlsruher Institut für Technologie, Karlsruhe, Germany, 2025.

[WT20] Jakob Weise und David Timmermann. Affordance-Bestimmung von Bauteilen in der Montage mit einem R-CNN basierten Ansatz. Bachelorarbeit, KIT Karlsruher Institut für Technologie, Karlsruhe, Germany, 2020.