Schnelles Greifen und Ablegen unbekannter Objekte mit einem Industrieroboter

Masterarbeit
von
Johannes Baumgartl
E-Mail: johannes.baumgartl@gmail.com

Institut für Informatik
Fakultät für Mathematik, Physik und Informatik
Universität Bayreuth

1. Gutachter:	Prof. Dr. Dominik Henrich
2. Gutachter:	Prof. Dr. Marc Erich Latoschik

Tag der Einreichung: 28.3.2011

Inhaltsverzeichnis

Kapitel 1

Einleitung

Für die Roboter in Science-Fiction-Filmen ist es ein leichtes komplexe Aufgaben zu lösen. Der Roboter WALL · E aus dem gleichnamigen Film [Pixar 11] räumt die Erde auf, nachdem die Menschen unseren Planten wegen der massiven Umweltverschmutzung verlassen mussten. Für ihn ist es eine Selbstverständlichkeit unbekannte Objekte zu greifen und sie wieder abzulegen. Die Aufgabe, irgendwelche Objekte aufzugreifen, auf eine Ablagefläche zu legen und zu einem späteren Zeitpunkt wieder aufzugreifen, um mit ihnen erneut eine Aufgabe zu erledigen, ist für heutige Roboter noch eine schwierige Aufgabe.

In klassischen industriellen Anwendungen ist die Stelle an der ein Objekt gegriffen werden soll, auf den Millimeter genau definiert. Und falls das Objekt nur ein kleines Stück von dieser Position abweicht, schlägt der Griff fehl. Die Roboterprogramme, die von den Robotern ausgeführt werden, sind mit ganz festen Befehlsabfolgen programmiert. Dabei sind nicht nur die Bewegungen fest vorgegeben, sondern auch die Position und Orientierung des Greifers, in der ein Objekt gegriffen werden soll. Diese festen Vorgaben müssen aufgehoben werden, um den nächsten Schritt hin zum oft gewünschten Roboter als Helfer machen zu können. In unserem Alltag gibt es häufig Situationen, in denen gerade benutzte Gegenstände wieder aufgeräumt werden müssen.

In einem Handwerksbetrieb ist es üblich, dass, bevor der Arbeitstag zu Ende geht, alle Werkzeuge von der Werkbank eingesammelt, gereinigt und zurück an ihren angestammten Platz gelegt werden.

Im Haushalt muss jeden Morgen der Frühstückstisch abgeräumt werden. Nicht selten legt man die Marmelade, Brot, Butter und Besteck auf ein Tablett, um nicht mehrmals zwischen Küche und Esszimmer hin und her zu

laufen. An der Kasse im Supermarkt räumt man die Einkäufe zurück in den Wagen.

In der Altenpflege sind die Fachkräfte oft mit mit dem Aufräumen, beispielsweise von Bettlagen, die von der Wäscherei zurück gekommen sind, beschäftigt. Die dafür benötigte Zeit könnte sinnvoller genutzt werden, indem man sich mit den alten Menschen beschäftigt oder mehr Zeit für die eigentliche Pflege hat, wenn ein Roboter diese Aufgabe übernehmen würde. Auch in den anderen Beispielsituationen ist es denkbar, einen Roboter einzusetzen, der diese Handlungen ausführt, um den Menschen zu entlasten.

Es ist nicht unbedingt notwendig, dass dem Roboter von vornherein bekannt ist, um welches Objekt es sich handelt. Er muss die Objekte erst nur greifen und wieder ablegen können. Der Roboter ist nur eine Hilfe, wenn er die Aufgaben in einer Zeit erledigt, die mindestens so schnell ist, wie wenn ein Mensch diese ausgeführt hätte. Deshalb ist es das Ziel dieser Arbeit, unbekannte Objekte schnell zu greifen und wieder abzulegen.

1.1 Aufgabenstellung

In dieser Arbeit soll ein Prototyp entwickelt, getestet und bewertet werden, der das schnelle Greifen und Ablegen von unbekannten Objekten ermöglicht.

Hardwareseitig ist ein sieben-achsiger Industrieroboter, ein Backengreifer und eine auf dem Handgelenk des Roboters montierte Farbkamera (Abbildung 1) vorhanden.

Der Roboter soll von einer Fläche unbekannte Objekte greifen und diese auf einer Ablagefläche platzieren. Es dürfen mehrere Objekte auf der Aufgreiffläche liegen und auf der Ablagefläche sollen möglichst viele Objekte abgelegt werden können. Diese Objekte sollen zu einem späteren Zeitpunkt wieder von der Ablagefläche aufgegriffen und einem Menschen übergeben werden. Der Mensch soll die Objekte auf die Aufgreiffläche legen.

Es müssen vornehmlich zwei Komponenten, die schon im Titel der Arbeit genannt sind, das Greifen und Ablegen, betrachtet werden. In jeder Komponente muss eine Planungsphase und danach eine Ausführungsphase durchlaufen werden. Am Ende der Ausführung soll validiert werden, ob diese erfolgreich absolviert worden ist. Diese Phase besteht vornehmlich aus der Bewegung des Roboters und der Steuerung des Greifers. Natürlich müssen diese Bewegungen auch geplant werden. Der Fokus der Arbeit liegt jedoch auf den Greif- und Ablageplanungsphasen.

Abbildung 1: Die verfügbare Hardware: Roboter KUKA Leichtbauroboter 4, Backengreifer PG 70 von Schunk und Logitech Webcam Pro 9000.

Gegeben für die Planungsphasen ist ein Kamerabild, die aktuellen Kameraparameter für die kalibrierte Kamera und die Sensorwerte des Roboters und Greifers. Des Weiteren ist die Ebene bekannt, auf der die Objekte liegen und die genauen Abmessungen der Ablagefläche.

Gesucht sind die *Greifkonfiguration* und die *Ablagekonfiguration*, die beide aus der Position und Orientierung des Greifers bestehen, wenn das Objekt gegriffen oder abgelegt werden soll. Aus diesen Konfigurationen müssen die Bewegungen des Roboters berechnet werden, um diese einzunehmen. Nachdem der Griff erfolgt ist, soll evaluiert werden, ob dieser erfolgreich war.

Die Greifplanung soll einen kraftschlüssigen Griff planen, der fehlertolerant gegenüber sensorischer Ungenauigkeiten ist. Für alle Komponenten gilt, dass sie im zweidimensionalen Raum planen und die Bewegungen im dreidimensionalen Raum ausgeführt werden.

Die Frage „Was ist schnell?“ lässt sich auf den ersten Blick nicht so leicht beantworten. Für Roboterbewegungen ist in der Norm ISO-10218 [ISO-10218 06] definiert, welche Geschwindigkeiten unter welchen Bedingungen erlaubt sind. Wenn der Mensch mit dem Roboter interagieren soll, ist eine maximale Geschwindigkeit von 0,25 m/s erlaubt. Diese Geschwindigkeit ist eine sehr konservative Grenze. In den Arbeiten von Haddadin et al. [Haddadin 07, Haddadin 08, Haddadin 11] sind umfangreiche Analysen gemacht worden, die sich mit der Geschwindigkeit von Roboterbewegungen und den resultie-

rendem Verletzungsgrad beschäftigen. Es sind unter anderem Tests durchgeführt worden, bei denen der Roboter verschiedene Objekte gegriffen hat. Darunter waren auch scharfe Gegenstände. Eine Erkenntnis dieser Arbeit ist, dass die zu erwartenden Verletzungen auch bei Geschwindigkeiten über 0,25 m/s, abhängig vom Objekt, noch gering bleiben. In dieser Arbeit sollen zumindest die Geschwindigkeiten aus der Norm erreicht werden.

Für die Planungsvorgänge soll „schnell“ bedeuten, dass deren Laufzeit so kurz ist, dass sie den Fluss der Anwendung nicht merklich unterbrechen. Das heißt, dass der Benutzer, der mit dem Roboter interagiert, nicht bemerkt, dass eine Planung durchgeführt worden ist.

1.2 Stand der Forschung

In diesem Abschnitt werden ausschließlich Gesamtsysteme vorgestellt. Für die beiden Planer wird der Überblick über den Stand der Forschung jeweils in deren Kapiteln gegeben.

Für jedes System wird die Funktionsweise und der angestrebte Einsatzbereich beschrieben. Darunter fallen, soweit bekannt, die verwendete Hardware sowie Planungsalgorithmen und die vorgestellte Beispielanwendung.

Grundsätzlich lassen sich die Systeme in zwei Domänen einteilen: Systeme, die für eine Anwendung im Haushalt entwickelt werden und die, die eine Aufgabe in der Industrie erfüllen sollen.

Zuerst werden die Robotersysteme für den Haushalt vorgestellt und im zweiten Teil dieses Abschnitts werden System für den industriellen Einsatz beleuchtet.

Das Deutsche Forschungszentrum für künstliche Intelligenz (DFKI) hat den Roboter AILA [DFKI Bremen 11] entwickelt. Der Roboter kann seine Umwelt dreidimensional wahrnehmen und Objekte sowie ihre Eigenschaften erkennen und somit gezielt mit ihnen umgehen. Durch Stereokameras und Laserscanner wird die Umgebung aufgenommen und mit Hilfe einer RFID-Antenne in der Roboterhand werden die Informationen über das Objekt aus der Produktdatenbank abgerufen. Es können ihm komplexe Aufgaben zugetragen werden wie: „Hole die rote Dose aus dem Regal“. Das Hauptaugenmerk bei diesem Roboter liegt auf der Beschreibung der Gegenstände und deren Manipulation. Es existieren weitere Systeme von diversen namhaften Forschungseinrichtungen, darunter dem Institut für Robotik und Mechatronik des DLR, das den Roboter Rollin Justin [DLR 11] entwickelt hat, und

der Technischen Universität München. Diese Systeme zielen stark auf die kognitiven Fähigkeiten von Robotern ab und sind deshalb für diese Arbeit nicht relevant. Es sollen keine Objekte erkannt oder gelernt werden.

Das Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA hat eine Vielzahl von Beispielanwendungen definiert. Sie umfassen beide Domänen: den Haushalt sowie das industrielle Umfeld. Der Care-O-bot^{TM}3 [IPA 11] ist ein Serviceroboter, der Aufgaben im Haushalt bewältigt. Er kann Hol- und Bringdienste erledigen, den Tisch decken und Türen und Schubladen öffnen. Es ist eine mobile Plattform mit einem sieben-achsigen Arm und Dreifingergreifer. Er kann sich selbständig in unbekannten Umgebungen bewegen, Objekte erkennen und lokalisieren, um diese zu handhaben, aber auch neue Objekte eigenständig erlernen. Er soll als interaktiver Butler eingesetzt werden, der die Personen selbständig anspricht und diese auffordert über ein Eingabeterminal ihre Bestellung aufzugeben. Er greift dann selbständig nach dem Getränk, stellt es auf sein Serviertablett und offeriert das Getränk der Person, die es bestellt hat.

Im industriellen Umfeld werden vom IPA die unterschiedlichsten Anwendungen angeboten. Es werden hier die zwei relevantesten vorgestellt. Einmal haben sie eine Anwendung, in der mit einem Industrieroboter Teile von Kurbelwellen aus einer Kiste vereinzelt werden. Dabei sind die Geometrie und die möglichen Greifkonfigurationen bekannt. Es wird ein Objekt mit einem 3D-Sensor erkannt und geprüft, ob es durch eine der gespeicherten Greifkonfigurationen greifbar ist. Falls ja, wird der Griff ausgeführt. Die zweite Anwendung ist aus der Intralogistik. Speziell werden das Palettieren, Depalettieren, Kommisionieren und Sortieren als Anwendungen genannt. Es werden allerdings keine genauen Informationen über die einzelnen Komponenten benannt. Es wird nur auf die heute immer wichtiger werdenden Methoden zur dreidimensionalen Objekterkennung hingewiesen.

Der „pi4_workerbot“ von pi4_robotics GmbH [pi4_robotics 11] ist im Zuge des EU-Projekts PiSA (Assembly System Integrated Project) entwickelt worden. Sein Einsatzgebiet sind Prüf- und Montageaufgaben, die zuvor nur von Menschen erledigt werden konnten. Er ist ausgestattet mit zwei Armen und einem Kopf, an dem bis zu drei verschiedene Kameras montiert sind. Eine S/W-Kamera, eine Farbkamera und eine optionale TOF-Kamera. In einer Beispielanwendung wird gezeigt, wie aus einem Setzkasten metallische Objekte gegriffen, mit Hilfe der Kameras überprüft und dann in zwei verschiedene Setzkästen einsortiert werden. Der eine enthält die Bauteile, deren Qualität ausreichend gut ist, der andere die Objekte, die die Qualitätsprüfung nicht bestanden haben. Laut Produktbeschreibung sind aber weitere Anwendun-

gen in dieser Art einfach und intuitiv programmierbar. Es ist nicht davon auszugehen, dass das System unbekannte Objekte greifen kann. Eine genaue Bewertung des Systems ist nicht möglich, da die einzelnen Verfahrensschritte nicht weiter spezifiziert werden.

In [Marturi 10] wird ein Prototyp beschrieben, der entwickelt wurde, um eine Fertigungsaufgabe zu erledigen. In der Beispielanwendung werden auf einen Stab zwei Holzscheiben gesteckt und diese mit einem Stift gesichert. Es wird in einem Kamerabild nach den vier Objekten gesucht. Diese werden nach einem vorgegebenen Fertigungsplan gegriffen und montiert. Die Greifplanung ist speziell auf die vier verwendeten Bauteile beschränkt.

1.2.1 Schlussfolgerung

Es existieren viele Prototypen und auch Systeme, die aus dem Prototypenstadium heraus sind, die sich mit dem Greifen und Ablegen beschäftigen. Dabei sind die übergeordneten Ziele höchst unterschiedlich. Einmal sollen Objekte erkannt und damit komplexe Befehle verarbeitet werden und dann durch den Roboter ausgeführt werden. Bei industriell motivierten Systemen existieren Anwendungen, die entweder auf großindustrielle Anwendungsentwicklung abzielen oder ein Prüfroboter, dessen genaue Fähigkeiten nicht weiter beschrieben werden. Oder es sind Prototypen für eine ganz spezielle Fertigungsaufgabe entwickelt worden, bei der die einzelnen Planungsphasen stark spezialisiert sind.

1.3 Kapitelübersicht

Die Arbeit gliedert sich in drei Hauptkapitel. Das Kapitel 2 behandelt die Greifplanung und Kapitel 3 die Ablageplanung. Darin sind in beiden eine Übersicht über den Stand der Forschung, Problemanalyse, Umsetzung des Planers und Experimente-Teil enthalten. Im Kapitel 4 wird der Prototyp beschrieben, der die Planungsalgorithmen aus den beiden vorherigen Kapiteln verwendet. Mit dem dort beschriebenen Prototypen werden einige Experimente durchgeführt, um dessen Fähigkeiten und Grenzen zu bewerten. Am Ende der drei Hauptkapitel wird jeweils eine Schlussfolgerung gezogen, das neben einer Zusammenfassung und Interpretation auch einen Ausblick enthält. Im letzten Kapitel 5 wird die gesamte Arbeit noch einmal zusammengefasst.

Kapitel 2

Greifen

Die Greifplanung hat zum Ziel, in einer vorgegebenen Umweltsituation ein bestimmtes Objekt mit dem Roboter zu greifen. Der Planer soll im zweidimensionalen Raum für unbekannte Objekte die Position und Orientierung eines Backengreifers berechnen. Dafür ist ein Bild einer kalibrierten Kamera und die Fläche auf der sich die Objekte befinden gegeben. Die sensorischen Ungenauigkeiten, wie Bildrauschen und Positionsungenauigkeiten des Greifers und Roboters, sollen von der Planung berücksichtigt werden. Die Planungszeit soll dabei möglichst klein gehalten werden. Das übergeordnete Ziel ist es, schnell Objekte zu greifen und abzulegen.

Um dieses Themenspektrum in adäquater Weise zu beleuchten, gliedert sich das Kapitel 2 in folgende Unterpunkte: Stand der Forschung (Abschnitt 2.1), Problemanalyse (Abschnitt 2.2), Umsetzung (Abschnitt 2.3), Experimente (Abschnitt 2.4) und Schlussfolgerung (Abschnitt 2.5).

2.1 Stand der Forschung

Greifplaner unterscheiden sich durch die Art des durchzuführenden Griffes. Die meisten veröffentlichten Verfahren planen ein Objekt kraftschlüssig zu greifen. Es existieren aber auch Verfahren, die einen formschlüssigen Griff zum Ziel haben, beispielsweise [Cornellà 09]. Im weiteren Verlauf dieses Abschnitts werden ausschließlich Ansätze vorgestellt, die einen kraftschlüssigen Griff zum Ziel haben, da mit einem Backengreifer gearbeitet wird. Mit diesem besteht nahezu keine Möglichkeit einen Formschluss zwischen Greiferbacken und Objekt herzustellen.

Die Kriterien nach denen die Planungsverfahren beurteilt werden sind: Wird die Planung für ein bekanntes oder unbekanntes Objekt ausgeführt? Wie viel Zeit benötigt die gesamte Planungsphase? Berücksichtigt der Planer die Positionierungenauigkeit des Greifers und fehlerbehaftete Sensorwerte, wie beispielsweise verrauschte Bilder?

Für ein Objekt kann es mehrere mögliche *Greifkonfigurationen* geben, mit denen ein Griff erfolgreich durchgeführt werden kann. Eine Greifkonfiguration besteht aus Position und Orientierung des Greifers relativ zum Objekt und der Stellung der Greiferfinger/-backen. Diese Greifkonfigurationen werden durch Qualitätsmetriken bewertet. Diese Metriken arbeiten meistens auf ganz bestimmten Objektmodellen und mit ganz bestimmten Greifern. Eine sehr häufige Paarung ist ein polyedrisches Objektmodell auf den eine Greifkonfiguration für einen n-Fingergreifer geplant wird. In [Bone 01] werden vier solcher Metriken für einen 3-Fingergreifer verglichen. In [Cornellà 03] werden zwei der in [Bone 01] vorgestellten Metriken von Ferrari und Canny erweitert, so dass Greifkonfigurationen für einen 4-Finger Greifer geplant werden können. Suarez gibt in [Suárez 06] eine noch ausführlichere Auflistung von verschiedenen Qualitätsmetriken an. Viele Metriken sind nicht zum Bewerten von Greifkonfigurationen mit unbekannten Objekten geeignet, da sie zu viele Informationen über das Objekt selbst benötigen. Ganz abgesehen von einem möglichst exakten Objektmodell werden oft Informationen über die Oberflächenstruktur und das Material des Objekts benötigt, um die auftretende Reibung für die Metrik zu verwenden. Diese Metriken berücksichtigen auch nicht, dass bei der Durchführung des Griffs Positionierungsungenauigkeiten durch den Roboter und die Greiferfinger selbst auftreten können.

In [Cornellà 05] wird die Positionierungenauigkeit der Greiferfinger bei der Planung berücksichtigt. Es werden unabhängige *Greifregionen* für ein gegebenes polygonales Objekt berechnet. Diese sind so definiert, dass, sobald ein Finger des Greifers irgendwo an jeder Region angreift, ein kraftschlüssiger Griff zustande kommt. Hierbei wird davon ausgegangen, dass ein Finger genau an einem Punkt der unabhängigen Greifregion ansetzt. In [Roa 09] wird ein Verfahren beschrieben, dass diese unabhängigen Greifregionen im 3D berechnet. Bei einem Backengreifer muss die Breite der Backen beachtet werden. Damit besteht ein flächiger Kontakt. Einen solchen Kontakt berücksichtigen beide Veröffentlichungen nicht.

Es gibt einige weitere Veröffentlichungen, die für bekannte Objekte und für verschiedenste Greifer, sowohl im dreidimensionalen als auch im zweidimensionalen Raum, eine Greifkonfiguration planen [Berenson 07, Bone 08, Smith 99, Zacharias 09]. Um ein Verfahren zu verwenden, das ein gegebenes

Objektmodell voraussetzt, muss dieses erst online erstellt werden. Beispielsweise verwendet das Verfahren aus [Zacharias 09] ein dreidimensionales polygonales, nicht konvexes Objektmodell. Solche Modelle sind nur mit einem erheblichen zeitlichen Aufwand zu erstellen. Des Weiteren sind die Verfahren nicht darauf ausgelegt, mit schlechten oder gar unvollständigen Objektmodellen umzugehen.

Den ersten Schritt weg von bekannten Objekten hin zu unbekannten macht Christopoulos in [Christopoulos 07]. Dort wird ein Template-basierter Ansatz im zweidimensionalen Raum vorgestellt. Das Verfahren arbeitet mit einer Template-Kontur von einem Objekt. In einem Kamerabild wird mit Hilfe dieses Templates das zu greifende Objekt gesucht. Dabei wird angenommen, dass die Kontur des Objekts im Bild geschlossen und vollständig ist. Die Template-Kontur und die Kontur aus dem Kamerabild werden vereinigt und darauf die Greifplanung durchgeführt. Dieses Verfahren setzt eine exakte und fehlerfreie Bildverarbeitung voraus. Sobald die Kontur nicht komplett geschlossen ist, kann das Verfahren nicht arbeiten.

Hebt man die Einschränkung auf, dass das Objekt bekannt oder zumindest teilweise bekannt ist, gibt es Ansätze die auf Lernverfahren zurückgreifen [Bodenbagen 09, Gorges 09, Saxena 08]. Lernen bedeutet immer, dass entweder mit einer Beispielmenge der Planer trainiert werden muss, oder dass durch ausprobieren, eine Greifkonfiguration gelernt wird, was den Nachteil eines hohen Zeitaufwandes mit sich bringt. Die Verwendung einer Beispielmenge zielt darauf ab, dass die Objekte wiedererkannt werden und dadurch die Greifkonfiguration schon bekannt ist.

Wenn man keine Verknüpfung zwischen Objekt und Greifkonfiguration lernen will, steht der Ansatz zur Verfügung, unbekannte Objekte durch Primitive wie Quader, Zylinder oder Kugeln zu approximieren. Für diese Primitive wird dann eine Greifkonfiguration berechnet. Diese Variante verfolgen beispielsweise [Huebner 09] und [Miller 03]. Da hierbei die Greifkonfiguration mit Hilfe eines unter Umständen stark approximierten Objektmodells berechnet wurde, ist es schwer, einen kraftschlüssigen Griff zu garantieren.

In [Bone 08] wird ein Prototyp zum greifen von unbekannten Objekten vorgestellt. Es wird durch den Einsatz von Laserscanner und einer Kamera, die beide an dem Handgelenk des Roboters montiert sind, ein dreidimensionales Modell des zu greifenden Objekts erstellt. Auf diesem Modell wird dann für einen Backengreifer eine Greifplanung durchgeführt. Das System benötigt eine Vielzahl von Sensoren, die alle an dem Roboter montiert sind. Diese schränken die Bewegungsfreiheit des Roboters stark ein. Die Ausführungszeit, ausgehend von der ersten Aufnahme eines Bildes bis zur abgeschlossenen

Greifplanung, beträgt für die beiden verwendeten Beispielobjekte 3,7 sec bzw. 2,18 sec. Bei dieser Laufzeit kann man nicht mehr von schnell sprechen.

In [Yamazaki 06] wird ein Greifplanungsverfahren vorgestellt das Greifkonfigurationen für einen Backengreifer und unbekannte Objekte berechnet. Es werden Stereobilder von dem Objekt aufgenommen und daraus ein Voxelmodell berechnet. Die verwendete Qualitätsmetrik für eine Greifkonfiguration ist der Inhalt der Fläche, die von dem Greiferbacken verdeckt wird; je größer diese Fläche, desto besser die Greifkonfiguration. Dabei wird berücksichtigt, dass die Greiffläche groß genug ist, damit das Objekt überhaupt zwischen die Greiferbacken passt. Die Positionierungenauigkeit des Greifers und Roboters bei der Ausführung des Griffs wird während der Planung nicht berücksichtigt. Es werden Ergebnisse für vier Objekte gezeigt. Dabei wurde die Planung immer schneller als eine Sekunde durchgeführt, wobei die Zeit für die Objektmodellierung nicht berücksichtigt wurde.

Zusammenfassend kann man feststellen, dass diverse Verfahren zur Greifplanung entwickelt worden sind. Es werden kraftschlüssige oder formschlüssige Griffe realisiert. Eine Qualitätsmetrik wählt eine Greifkonfiguration aus, um bekannte oder unbekannte Objekte zu greifen. Dabei hängen alle Komponenten des Planers sehr stark von dem verwendeten Greifer ab. Es existiert jedoch kein Greifplaner für einen Parallelbackengreifer, der im zweidimensionalen Raum einen kraftschlüssigen Griff für unbekannte Objekte plant, bei dem die Planung auf einem einzigen Kamerabild erfolgt.

2.2 Problemanalyse

Wenn man sich mit der Greifplanung auseinandersetzt, muss ein Modell des zu greifenden Objekts zur Verfügung stehen oder es muss aus Sensordaten modelliert werden. An diesem *Objektmodell O* werden *Greifregionen* GR identifiziert. Eine solche Region muss sich dazu eignen, dass man einen Finger des Greifers an ihr anlegen kann. Dazu spielt der Typ des Griffs eine Rolle. Bei einem formschlüssigen Griff muss die Geometrie des Greifers mit berücksichtigt werden. Bei einem kraftschlüssigen Griff spielt die Oberflächenbeschaffenheit und das Material, sowohl des Objekts als auch des Greifers, in Bezug auf die Reibung, eine Rolle.

Falls die Menge M_{GR} der extrahierten *Greifregionen* GR nicht leer ist, wird diese Menge in nicht disjunkte Teilmengen $U_{\mathrm{GR}} \subset M_{\mathrm{GR}}$ aufgeteilt. Diese Teilmengen müssen so gewählt sein, dass man mit den enthaltenen Greifregionen aus jeder Teilmenge eine *Greifkonfiguration* GK berechnen könnte.

Die Kandidatenmenge $M_{U_{\mathrm{GR}}}$ besteht aus allen dieser U_{GR}.

$$M_{U_{\mathrm{GR}}} := \{U_{\mathrm{GR}} | U_{\mathrm{GR}} \subset M_{\mathrm{GR}}, P(U_{\mathrm{GR}}) \in M_{\mathrm{GK,O}}\}$$

Dabei ist $M_{\mathrm{GK,O}}$ die Menge aller möglichen Greifkonfigurationen GK für das vorliegende Objektmodell O. Die Greifkonfiguration besteht aus Position und Orientierung des Greifers und der Stellungen der einzelnen Greiferfinger/-backen. Die Funktion P berechnet aus einer Teilmenge von Greifregionen U_{GR} eine Greifkonfiguration $gk \in M_{\mathrm{GK,O}}$.

$$\begin{aligned} P : M_{U_{\mathrm{GR}}} &\rightarrow M_{\mathrm{GK}} \\ P(U_{\mathrm{GR}}) &\mapsto gk \end{aligned}$$

In Abbildung 2 wird ein Beispiel gegeben, wie die Menge der Greifregionen M_{GR} deren Teilmengen U_{GR} und die Menge der Greifkonfigurationen $M_{\mathrm{GK,O}}$ für ein Objekt aussehen kann, das mit einem parallelen Backengreifer gegriffen wird. In rot sind die Greifregionen gekennzeichnet und in blau werden die Greifkonfigurationen angedeutet, wie der Backengreifer positioniert werden kann.

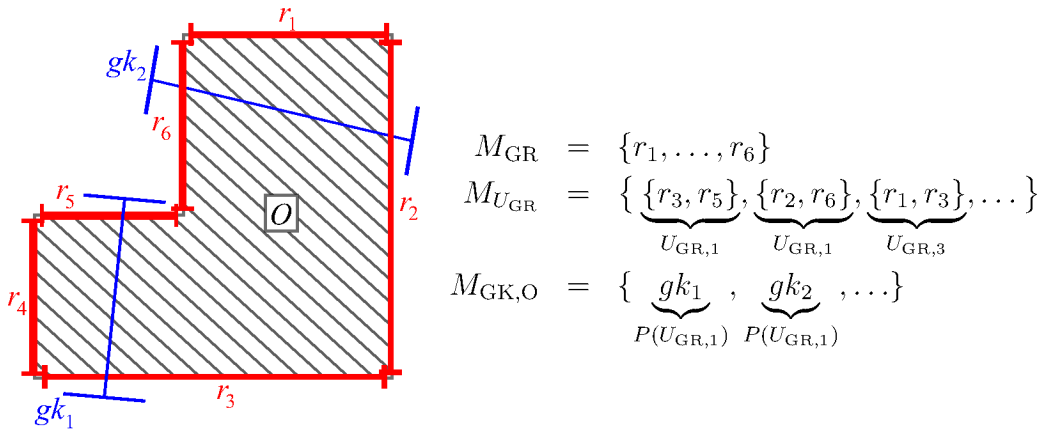


Abbildung 2: Beispiel wie die einzelnen Mengen für einen parallelen Backengreifer aussehen können. In rot sind die Greifregionen gekennzeichnet und in blau werden die möglichen Greifkonfigurationen für einen Backengreifer angedeutet.

Mit Hilfe einer *Qualitätsmetrik qm* wird aus den vorhanden Teilmengen der beste Kandidat $U_{\mathrm{GR,best}}$ gewählt. Mit $P(U_{\mathrm{GR,best}}) = gk_{\mathrm{best}}$ wird die beste Greifkonfiguration berechnet. Die Qualitätsmetrik kann die unterschiedlichsten Bedingungen berücksichtigen. Beispielsweise, dass die Greiferfinger/-

backen möglichst im gleichen Abstand um den Objektschwerpunkt angreifen, um auftretende Momente bei einem kraftschlüssigen Griff zu minimieren [Bone 01].

Abbildung 3 stellt den logischen Ablauf der Greifplanung schematisch in einem Flussdiagramm dar.

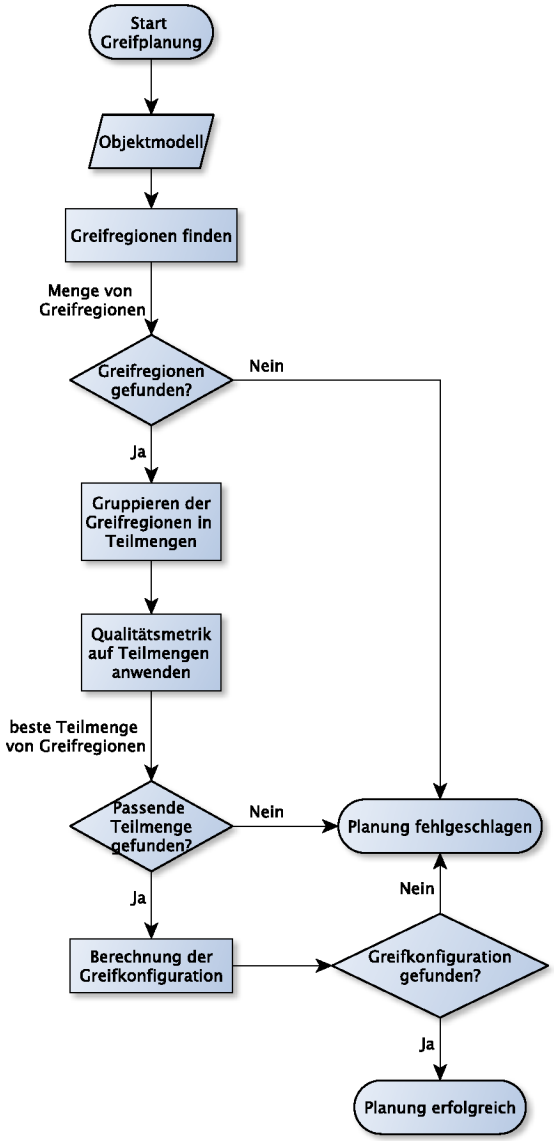


Abbildung 3: Flussdiagramm zeigt den Prozess der Greifplanung

2.3 Umsetzung

In diesem Unterkapitel wird auf die konkrete Umsetzung des entwickelten Greifplaners beschrieben. Dieser soll, wie schon am Ende von Unterkapitel 2.1 dargelegt, einen kraftschlüssigen Griff mit einem Backengreifer für unbekannte Objekte im zweidimensionalen Raum planen. Die Objekte werden ausgehend von einem einzigen Kamerabild modelliert (Abschnitt 2.3.1). Auf Basis dieses bildbasierten Objektmodells plant der Algorithmus die Greifkonfiguration für den Griff. Dabei sollen sensorische Ungenauigkeiten bei der Objektmodellierung, die Positionierungenauigkeit des Greifers und die Geometrie der Greiferbacken berücksichtigt werden. Der Greifplaner wird im Abschnitt 2.3.2 vorgestellt.

2.3.1 Bildbasiertes Objektmodell

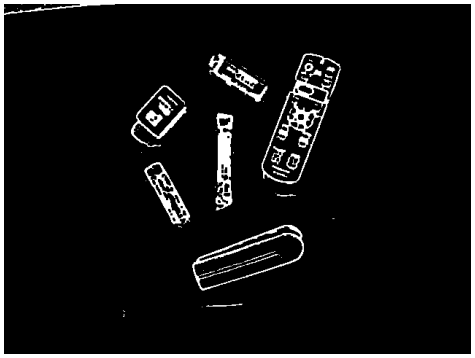
Das Objektmodell wird aus einem einzelnen Kamerabild erstellt. Es hat die Auflösung 640 × 480 Pixel. Dazu werden einige Bildverarbeitungsschritte durchgeführt. Das Ausgangsbild I_{src} ist ein Grauwertbild. Dieses wird mit einem adaptiven Schwellwertverfahren in ein Binärbild I_{dest} umgewandelt.

$$I_{\text{dest}}(x,y) = \begin{cases} 1 & \text{wenn } I_{\text{src}}(x,y) > T(x,y), \\ 0 & \text{sonst} \end{cases}$$

Dabei ist $T(x,y)$ der Mittelwert der Umgebung von (x,y). Auf das Binärbild wird anschließend der morphologische Filter „CLOSE“ [Gonzalez 02] angewandt, um eventuell entstandene Lücken zu schließen . Beide Filter haben eine Maskengröße von 5×5 Pixel. Danach wird der Algorithmus aus [Suzuki 85] auf dem Bild ausgeführt. Dieser liefert die Konturpixel aller Objekte (Abbildung 4(a)). Pro Objektmodell werden die Konturpixel O_{P} und die minimale Boundingbox $O_{\text{B}} := \{(p_1, \dots p_4) | p_1, p_2, p_3, p_4 \in \mathbb{N}^2\}$ gespeichert. Damit ist das hier verwendete Objektmodell definiert als $O := (O_{\text{P}}, O_{\text{B}})$. Die Abbildung 4(b) zeigt das Ergebnis der Modellierung. Für alle sechs Objekte auf dem Bild ist in hellblau die minimale Boundingbox eingezeichnet.

2.3.2 Planung

Dieser Abschnitt beleuchtet, wie die einzelnen Schritte des Planungsablaufs konkret umgesetzt worden sind. Dabei werden die Fragen beantwortet: Wie



(a) Konturbild nach der aktiven Schwellwertfilterung und der Anwendung des „CLOSE“-Filters.

(b) Orginalbild mit den hellblau eingezeichneten minimalen Boundingboxen für alle Objekte.

Abbildung 4: Zwischenschritt und Ergebnis der bildbasierten Objektmodellierung

sind Greifregionen definiert und wie werden sie gruppiert? Mit welcher Qualitätsmetrik wird die beste Gruppe ausgewählt und wie berechnet man die Greifkonfiguration?

Finden und Gruppieren von Greifregionen

Für diesen Greifplaner wird eine Greifregion als gerade Kante definiert, die als Tupel $(\theta, p_{\text{Start}}, p_{\text{Ende}})$ repräsentiert wird. Dabei sind p_{Start} und p_{Ende} Start- und Endpunkt einer Strecke auf einer Geraden und θ der Winkel zwischen der Normalen von dieser Geraden und der x-Achse (Abbildung 5).

Im folgenden wird erklärt, wie die drei Komponenten θ, p_{Start} und p_{Ende} mit Hilfe der Hough-Transformation berechnet werden.

Die Hough-Transformation ist 1962 von Paul V. C. Hough [Hough 62] entwickelt worden, um in einem Gradienten- bzw. Binärbild Geraden oder Kreise zu finden. In unserem Fall sollen Geraden im Bild identifiziert werden. Dazu wird das vorliegende Bild in einen Dualraum, den so genannten Hough-Raum, transformiert. Ein Punkt in diesem Dualraum entspricht einer Geraden im Orginalbild.

Im konkreten Fall besteht das zu transformierende Bild nur aus der Menge O_{P} der Konturpixel des Objektmodells. Die Aufgabe ist diese Menge in den Hough-Raum zu transformieren. In einem weiteren Schritt werden die Pixel, die auf einer Geraden liegen zu Strecken zusammengefasst.

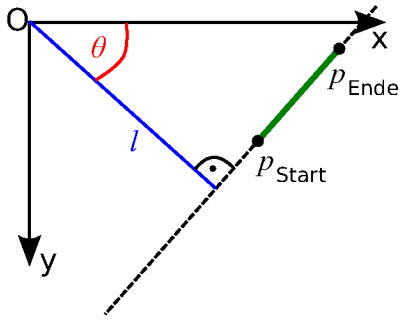


Abbildung 5: Darstellung der Parameter einer Greifregion und des Hough-Raums: l ist der Abstand zum Ursprung, θ ist der Winkel zwischen dem Lot auf die Gerade durch den Ursprung und der x-Achse. Die Punkte p_{Start} und p_{Ende} sind die Start- und Endpunkte der grünen Strecke.

Für jeden Konturpixel $(x_0, y_0) \in O_{\text{P}}$ muss jede Gerade gefunden werden, auf der dieser liegt. Dazu wird folgende Parametergleichung $\chi(\theta)$ für eine Gerade $g(l, \theta)$ verwendet:

$$\chi(\theta) := x_0 \cdot \cos(\theta) + y_0 \cdot \sin(\theta)$$

Dabei ist (x_0, y_0) ein Punkt auf der Geraden, θ ist der Winkel zwischen der x-Achse und dem Lot auf die Gerade durch den Ursprung und $\chi(\theta) = l$ ist der kürzeste Abstand zwischen dieser Geraden und dem Ursprung, (Abbildung 5). Damit wird der Hough-Raum für diesen Fall als $HR := \{(l, \theta) | l \in \mathbb{R}, \theta \in [0, \pi[\,\}$ definiert.

Sei nun G_{xy} die Menge aller Geraden $g(l, \theta)$, die den Punkt (x, y) schneiden.

$$G_{xy} = \left\{ (l, \theta) \middle| (l, \theta) = \big(\chi(\theta), \theta\big), \text{ mit } \chi(\theta) = x \cdot \cos(\theta) + y \cdot \sin(\theta) \right\}$$

Damit ist φ die Funktion, die für alle (l, θ) die Mengen G_{xy} zählt, in denen die Gerade $g(l, \theta)$ enthalten ist.

$$\varphi_{O_{\text{P}}} : HR \to \mathbb{N}$$

$$\varphi_{O_{\text{P}}}\big((l, \theta)\big) \mapsto n = \left\| \left\{ G_{xy} \middle| \, (l, \theta) \in G_{xy},\ (x, y) \in O_{\text{P}} \right\} \right\|$$

Der Wert $\varphi_{O_{\text{P}}}\big((l, \theta)\big)$ ist direkt proportional zu der Anzahl der Pixel $(x, y) \in O_{\text{P}}$, die auf der Geraden $g(l, \theta)$ liegen.

Die Hough-Transformation kann konstruktiv folgendermaßen durchgeführt werden:

$$\forall (x, y) \in O_{\text{P}}\, \forall \theta' \in [0, \pi[\text{: inkrementiere } \varphi_{O_{\text{P}}}\big((\chi(\theta'), \theta')\big) \text{mit } 1$$

Für jeden Konturpixel $(x, y) \in O_\mathrm{P}$ wird für alle $\widehat{\theta} \in [0, \pi[$ die Parametergleichung $\chi(\widehat{\theta})$ ausgewertet und der Funktionswert an dieser Stelle $\varphi_{O_\mathrm{P}}\big((\chi(\widehat{\theta}), \widehat{\theta})\big)$ inkrementiert. Die Abbildung 6 (a) zeigt das Binärbild der Konturpixel O_P und die Abbildung 6 (b) zeigt die dazugehörige Funktion φ. Die vier roten Punkte markieren die Geraden, auf denen die meisten Konturpixel liegen.

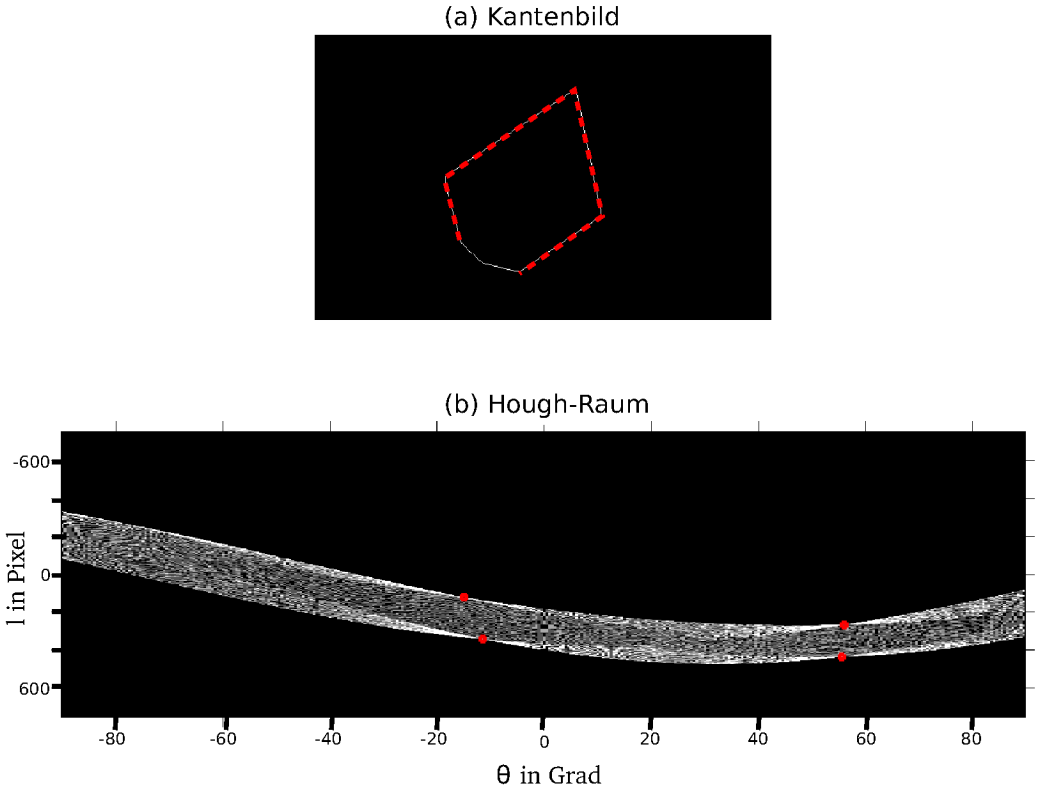


Abbildung 6: Beispiel für die Hough-Transformation: (a) Zeigt das Binärbild mit den Konturpixeln (Weiß) und den vier längsten Kanten (rot gestrichelt). (b) zeigt die dazugehörige Funktion φ_{O_P}, mit den vier größten Werten in rot eingezeichnet, diese entsprechen den rot gestrichelten Linien in (a).

Die Hough-Transformation ist besonders tolerant gegenüber Bildrauschen und Ungenauigkeiten [Hough 62], die bei der Objektmodellierung entstehen, da einzelne Fehlerpixel in O_P die große Anzahl an Einträgen in der Wertemenge der Funktion φ_{O_P} kaum beeinflussen.

Nachdem die Konturpixel O_P in den Hough-Raum transformiert worden sind. Kann man jetzt die Greifregionen bestimmen. Sie besteht aus drei Parametern θ, p_Start und p_Ende. Der Winkel θ soll der sein, der zwischen der Normalen der Geraden, auf der die Strecke mit den Start- und Endpunkten

(p_{Start} und p_{Ende}) liegt, mit der x-Achse sein. Damit entspricht das θ der Greifregion gerade dem θ aus dem Hough-Raum. Damit ist der Winkel *theta* für jede Greifregion bekannt. Jetzt müssen nur noch die Start- und Endpixel (θ, p_{Start} und p_{Ende}) der Strecken auf den durch (l, θ) definierten Geraden gefunden werden. Für jede Strecke auf der Geraden wird eine Greifregion erstellt die, die jeweiligen Start- und Endpunkte der Strecke enthält und den Winkel *theta*, durch den die Grade definiert ist auf der die Strecke liegt.

Das Verfahren benötigt die oben berechnete Funktion $\varphi_{O_{\text{P}}}$ und die Menge der Konturpixel O_{P}. Das Resultat ist die Menge der Greifregionen $M_{\text{GR}} = \left\{(\theta, p_{\text{Start}}, p_{\text{Ende}}) \middle| \theta \in [0, \pi[\, , p_{\text{Start}}, p_{\text{Ende}} \in O_{\text{P}} \right\}$.

Es wird wird die Menge $O_{\text{P,besucht}} = \left\{(x, y) \middle| (x, y) \in \mathbb{N}\right\}$ der schon betrachteten Konturpixel eingeführt. Diese ist anfange leer und wird im Laufe des Verfahrens gefüllt werden. Nun werden für jeden Pixel $(x, y)_{curr} \in O_{\text{P}} \backslash O_{\text{P,besucht}}$ die Hough-Parameter (l, θ) gesucht, für die $\varphi_{O_{\text{P}}}(l, \theta)$ maximal ist und gilt: $(l, \theta) \in G_{xy}$ von $(x, y)_{curr}$. Das ist die Gerade, die die meisten Pixel aus O_{P} enthält. Diese Gerade $g(l, \theta)$ wird, ausgehend von $(x, y)_{curr}$, in beide Richtungen verfolgt. Solange $(x, y)_i \in O_{\text{P}} \backslash O_{\text{P,besucht}}$ auf $g(l, \theta)$ liegt, werden die Start- und Endpixel p_{Start}, p_{Ende} der aktuellen Strecke angepasst. Der gerade betrachtete Konturpixel $(x, y)_i$ wird zu $O_{\text{P,besucht}}$ hinzugefügt. Es sind kleine Lücken auf den Strecken erlaubt, falls weniger als $l_{\text{Lücke,max}} \in \mathbb{N}$ Pixel in Folge nicht in den Konturpixeln enthalten sind oder schon zu der Menge der besuchten Pixel hinzugefügt wurden, wird trotzdem die Gerade weiter entlanggelaufen. Damit können sich Strecken schneiden und kleine Lücken innerhalb der Konturpixel ausgeglichen werden. Ist nun die gefundene Strecke länger als l_{min}, wird eine neue Greifregion $r = (\theta, p_{\text{Start}}, p_{\text{Ende}})$ zu M_{GR} hinzugefügt. Für alle Pixel auf der Strecke von r werden die G_{xy} aus dem Hough-Raum gelöscht und $\varphi_{O_{\text{P}}}$ angepasst.

Es wird die Anzahl der Strecken auf ein Maximum limitiert, um die Laufzeit zu begrenzen. Der Algorithmus wählt zufällig Punkte aus den Konturpixeln aus. Wenn ein Punkt zu einer Kante gehört, die über dem Schwellwert für $\varphi_{O_{\text{P}}}(l, \theta)$ liegt wird die dazugehörige Strecke bestimmt und hinzugefügt. Das geschieht so lange, bis die maximale Streckenanzahl erreicht ist oder alle Strecken gefunden sind.

Gruppieren der Greifregionen und Anwendung der Qualitätsmetrik

Da mit einem Backengreifer gegriffen werden soll, werden die Greifregionen entsprechend ihres Winkels θ zu den Teilmengen $U_\theta \subseteq M_{\text{GR}}$ zusammenge-

fasst.

$$U_\theta := \left\{ r \middle| r \in M_{\mathrm{GR}},\, \theta = const \right\}$$

In U_θ befinden sich nur Greifregionen die parallel zueinander liegen. Sei Ξ die Funktion, die die Strecke zwischen Start- und Endpunkt einer Region berechnet.

$$\begin{aligned} \Xi : U_\theta \to S = \{p + \lambda \cdot v | \lambda \in [0;1], p, v \in \mathbb{R}^2\} \\ \Xi(r) \mapsto s_r = \{p + \lambda \cdot v | \lambda \in [0;1], p &= p_{\mathrm{Start}}, \\ v &= p_{\mathrm{Ende}} - p_{\mathrm{Start}}, p_{\mathrm{Ende}}, p_{\mathrm{Start}} \in r\} \end{aligned}$$

Weiter sei die Funktion Ψ, die die zwei parallelen Strecken s_{r1} und s_{r2} auf die Gerade projiziert, die genau den gleichen Abstand zu beiden Strecken hat und zu beiden parallel ist. Die neue Strecke $s_{r1,r2}$ ist diejenige, auf der sich die beiden projizierten Strecken überlappen (Abbildung 7).

$$\begin{aligned} \Psi : S \times S &\to S \\ \Psi(s_{r1}, s_{r2}) &\mapsto s_{r1,r2} \end{aligned}$$

Sei die Funktion Ω, die mit Hilfe der Kameraparameter [Heikkila 97] die Strecke s vom Kamerabild auf die Aufgreiffläche projiziert.

$$\begin{aligned} \Omega : S &\to S^p := \{p + \lambda \cdot v | \lambda \in [0;1], p, v \in \mathbb{R}^3\} \\ \Omega(s) &\mapsto s^p \end{aligned}$$

Sei weiter die Funktion $D : S^p \times S^p \to \mathbb{R}$ die, die den Abstand zwischen zwei Strecken berechnet und $L : S^p \to \mathbb{R}$ die, die die Länge einer Strecke berechnet.

Damit lässt sich die Kandidatenmenge der Greifregionenpaare $M_{U_{\mathrm{GR}}}$ formal als:

$$\begin{aligned} M_{U_{\mathrm{GR}}} := \{\{r_1, r_2\} \in U_\theta | \Psi(\Xi(r_1), \Xi(r_2)) &\neq \emptyset, \\ D \circ \Omega \circ \Psi(\Xi(r_1), \Xi(r_2)) &< D_{\mathrm{Greifer,max}}, \\ L \circ \Omega \circ \Psi(\Xi(r_1), \Xi(r_2)) &> L_{\mathrm{Greifer,min}}, \} \end{aligned}$$

beschreiben. Dabei ist $D_{\mathrm{Greifer,max}}$ die maximal mögliche Distanz zwischen den Greiferbacken und $L_{\mathrm{Greifer,min}}$ die Breite der Greiferbacken.

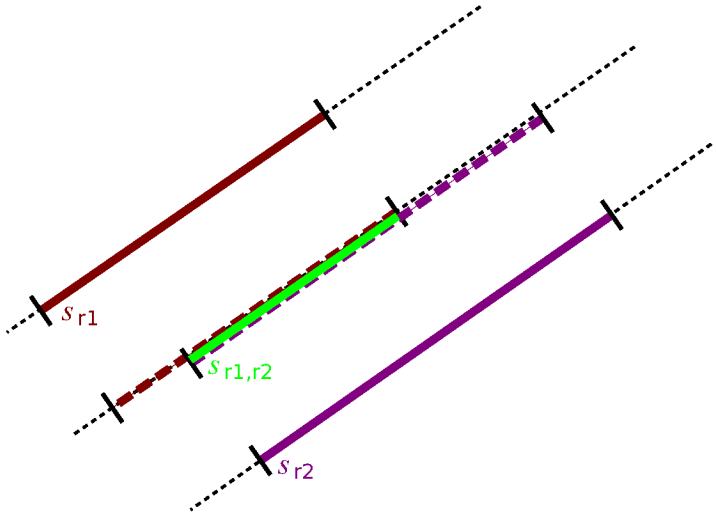


Abbildung 7: Visualisierung der Funktionsweise der Funktion Ψ. Es werden die beiden Strecken der Greifregionen (rot und lila durchgezogene Linie), die beiden projizierten Stecken (rot und lila gestrichelte Linie) und die gemeinsame Stecke (grüne Linie) dargestellt. Die drei parallelen Geraden sind als schwarz gestrichelten Linien eingezeichnet.

Der beste Kandidat $U_{\mathrm{GR,best}}$ aus $M_{U_{\mathrm{GR}}}$ ist

$$\begin{aligned} U_{\mathrm{GR,best}} &:= \{\overline{r_1}, \overline{r_2}\} \\ &:= \mathrm{argmax}_{\{r_1,r_2\} \in M_{U_{\mathrm{GR}}}} \{L \circ \Omega \circ \Psi(\Xi(r_1), \Xi(r_2))\} \end{aligned}$$

Es ist genau der Kandidat, dessen *gemeinsame Strecke* $s_{\mathrm{gemeinsam}} = \{p + \lambda \cdot v | \lambda \in [0;1]\} \in \Omega \circ \Psi(\Xi(\overline{r_1}), \Xi(\overline{r_2}))$ am längsten ist.

Aus diesem Kandidaten wird die Greifkonfiguration, das heißt Position und Orientierung des (nsa)-Koordinatensystems (Abbildung 8) des Greifers, berechnet. Der Ursprung des Koordinatensystems wird derart translatorisch verschoben, dass die a-Achse die Aufgreiffläche im Mittelpunkt der projizierten gemeinsamen Strecke schneidet. Dieser Punkt wird zu dem Objektmodell hinzugefügt. Damit ist $O = (O_{\mathrm{P}}, O_{\mathrm{B}}, s_{\mathrm{gemeinsam}})$ vollständig.

Es wird der Winkel γ zwischen der n-Achse und $s_{\mathrm{gemeinsam}}$ in der s-n-Ebene berechnet. Mit diesem Winkel γ wird dann um die a-Achse rotiert. Damit ist der Greifer so ausgerichtet, dass er direkt über dem zu greifenden Objekt steht. Da nur zweidimensional geplant wurde, ist die Höhe des Objekts und damit die Länge der translatorischen Bewegung in a-Achsen-Richtung unbekannt. Es sind zwei Strategien möglich, um mit dieser Problematik umzugehen. Die erste vollzieht die Translation bis zur Aufgreiffläche. Dabei wird

angenommen, dass die Höhe des Objekts kleiner ist, als die Länge der Greiferbacken, dass also keine ungewollte Kollision zwischen Greifer und Objekt auftritt. Die zweite Möglichkeit ist, dass man diese translatorische Bewegung solange in a-Achsen-Richtung durchführt, bis eine Kraft entgegen der Bewegungsrichtung auftritt. Dies tritt entweder ein, wenn der Greifer die Aufgreiffläche berührt oder wenn das zu greifende Objekt berührt wird. Damit können auch höhere Objekte gegriffen werden.

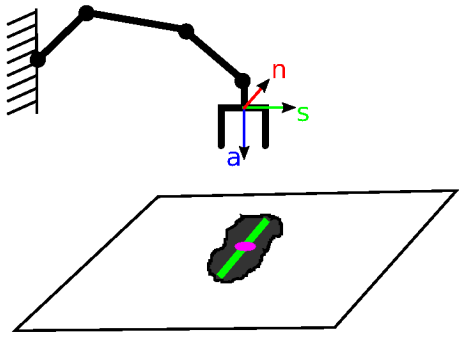


Abbildung 8: Visualisierung des nsa-Koordinatensystems zusammen mit dem Objektmodell auf der Aufgreiffläche, dem Mittelpunkt (rosa) der, auf die Aufgreiffläche, projizierten Strecke (grün).

Betrachtung der Kraftschlüssigkeit

Als kraftschlüssig bezeichnet man einen Griff, bei dem zwischen Greiferbacken und gegriffenen Objekt ein Kraftschluss $F_{\text{Last}} \leq F_{\text{Haft}} = \mu_{\text{H}} \cdot F_{\text{N}}$ besteht. Dabei ist F_{N} die Greifkraft (normal zur Reibfläche), $F_{\text{Last}} = m \cdot g$ die Gewichtskraft des Objekts und μ_{H} die Haftreibungszahl, abhängig von den Materialien und Oberflächen der Greiferbacken und dem Objekt.

Nachdem die zu greifenden Objekte unbekannt sind, damit auch deren Gewicht, Material und Oberflächenbeschaffenheit, kann man erstmal keine Aussage über die Kraftschlüssigkeit treffen. Man kann aber das maximale Gewicht m_{max} des Objekts bei gegebener minimaler Haftreibungszahl $\mu_{\text{H,min}}$ und Greifkraft angeben als:

$$m_{\text{max}} = \frac{|\mu_{\text{H,min}} \cdot F_{\text{N}}|}{g}$$

mit Erdbeschleunigung g. Für diese Abschätzung werden Punktkontakte angenommen, da nach den Amontonsschen Gesetzen [Wikimedia Foundation Inc. 11] die Größe der Reibflächen keine Rolle spielt.

2.4 Experimente

Alle Experimente wurden auf einem handelsüblichen Computer durchgeführt, der mit einem Intel®Core™2 Quad Q9450 Prozessor mit 2,66 GHz und 4 GB RAM ausgestattet ist. Die Bilder wurden mit einer Logitech Webcam Pro 9000 aufgenommen. Die Auflösung der Bilder ist 640×480 Pixel und deren Kameraparameter sind durch das Verfahren aus [Heikkila 97] ermittelt worden. Implementiert wurde der Greifplaner in C++ mit Hilfe der OpenCV-Bibliothek [Bradski 00].

Das erste Experiment in Abschnitt 2.4.1 hat zum Ziel, möglichst gute Werte für die diversen Parameter des Planers zu finden und dessen Laufzeit zu ermitteln. Mit dem zweiten Experiment aus Abschnitt 2.4.2 sollen die Grenzen des Planers zusammen mit der vorgeschalteten bildbasierten Objektmodellierung aufgezeigt werden.

2.4.1 Parameterbestimmung und Laufzeitmessung

In diesem Experiment wird für verschiedene Objekte eine Greifplanung durchgeführt. Alle Parameter des Planers werden dabei in gewissen Grenzen variiert. Diese Parameter sind:

- Die Schrittweiten $l_{\text{Step}}, \theta_{\text{Step}}$ auf den Achsen des Hough-Raums mit $l_{\text{Step}} > 0$ und $\theta_{\text{Step}} \in [0, \pi[$.
- Der minimale Funktionswert $\varphi_{\min}$ von $\varphi_{O_{\text{P}}}(l, \theta)$ mit $\varphi_{\min} > 0$
- Die minimale Länge $l_{\min}$ der Strecke mit $l_{\min} := ||\Xi(r)|| > 0$ einer Greifregion r in Pixel
- Die maximale Größe einer Lücke $l_{\text{Lücke,max}}$ auf einer Strecke mit $l_{\text{Lücke,max}} \geq 0$ Pixel

Eine Parameterkombination wird mit Hilfe der Größe der Kandidatenmenge, der Länge des besten Kandidaten und der Lage der beiden Greifregionen des besten Kandidaten bewertet. Die ersten beiden Kriterien werden direkt während der Laufzeit gemessen. Die Lage der ausgewählten Greifregionen wird durch manuelle Sichtung der Kamerabilder beurteilt. In diesen sind die beiden Greifregionen des gewählten Kandidaten in rot eingezeichnet und die gemeinsame Strecke in grün.

Für die Versuchsreihen wurden die Aufnahmen der Objekte aus Abbildung 9 verwendet, dabei wurde die Position und Orientierung der Kamera nicht verändert. Die Objektmodellierung wurde bei diesen Versuchen etwas angepasst. Die Größe der minimalen Boundingbox des Objekts wurde gleich der Bildgröße gesetzt, um die Ergebnisse der Laufzeitmessung nicht von der Objektgröße abhängig zu machen, sondern um eine Obere Schranke für die Laufzeit zu bestimmen.

Abbildung 9: Die Abbildung zeigt die Objekte mit denen die besten Parametersätze ermittelt wurden

Die Grenzen und Schrittweiten in denen die einzelnen Parameter variieren, sind aus Tabelle 2.1 zu entnehmen.

Wenn man den Parameter für den minimal zulässigen Funktionswert von $\varphi_{\min}$ höher als 60 ansetzt, dann wird für kleine Objekte wie den Kugelschreiber keine Greifkonfiguration gefunden. Bei deutlich niedrigeren Werten werden die Ergebnisse unbrauchbar. In der Abbildung 10(a) werden ganz kurze Kanten, anstatt der Kanten über die gesamte Länge des Objekts, bevorzugt.

Wenn man größere Lücken als sieben Pixel ($l_{\text{Lücke,max}} = 7$) innerhalb einer Strecke zulässt, werden Greifregionen erzeugt, deren Strecke keiner realen Objektkante entspricht (vgl. Abbildung 10(c)). Wenn man $l_{\text{Lücke,max}}$ nahe

(a) Schwellwert für $\varphi_{\min} = 1$ zu niedrig gewählt

(b) Schwellwert für $\varphi_{\min} = 31$ passend gewählt

(c) Wert für die Größe der maximalen Lücke zu groß ($l_{\text{Lücke,max}} = 9$)

(d) Wert für die Größe der maximalen Lücke passend gewählt ($l_{\text{Lücke,max}} = 5$)

Abbildung 10: Gegenüberstellung von fehlerhaften Planung, auf Grund schlechter Parameter, und korrekten Planungen mit passenden Parametern. Grün: gemeinsame Strecke $s_{\text{gemeinsam}}$ von den beiden Strecken der Greifregionen (rot).

Parameter	min	max	Schrittweite
l_{Step} [Pixel]	0,5	2,5	0,5
θ_{Step} [rad]	$\pi/180$	$10 \cdot \pi/180$	$\pi/180$
$\varphi_{\min}$	1	120	30
$l_{\min}$ [Pixel]	10	140	10
$l_{\text{Lücke,max}}$ [Pixel]	0	10	1

Tabelle 2.1: Tabelle der Wertebereiche, in denen die einzelnen Parameter getestet wurden. Für jeden Parameter ist der minimale und maximale Wert und die Schrittweite aufgeführt.

null wählt wird das Verfahren gegenüber Fehlern in den Objektmodellen intoleranter. Die Strecken bleiben sehr kurz und werden dann häufig wegen ihrer Länge verworfen.

Die minimale Länge einer Strecke spielt insofern eine Rolle, dass, wenn dieser Parameter zu hoch gewählt wurde, kaum Greifregionen gefunden werden. Somit ist die Planung nur bei Objekten erfolgreich, die in ausreichender Größe abgebildet werden.

Die getesteten Schrittweiten auf den Achsen des Hough-Raums haben keinen erkennbaren Einfluss auf die Ergebnisse.

Die Auswertung der Daten hat ergeben, dass der Parametersatz

$$(l_{\text{Step}}, \theta_{\text{Step}}, \varphi_{\min}, l_{\min}, l_{\text{Lücke,max}}) = ((1, 3 \cdot \frac{\pi}{180}, 36, 15, 6))$$

für die meisten klein abgebildeten Objekte und der Parametersatz

$$(l_{\text{Step}}, \theta_{\text{Step}}, \varphi_{\min}, l_{\min}, l_{\text{Lücke,max}}) = (1, \frac{\pi}{180}, 50, 30, 7)$$

für die meisten groß abgebildeten Objekte gute Ergebnisse liefert. Das heißt, dass nur mit sehr wenigen gewählten Kandidaten der reale Griff ($P(U_{\text{GR,best}}) \neq \{\}$) nicht erfolgreich wäre.

Für die Auswertung der Laufzeit wurden 6759 Testläufe mit unterschiedlichen Parametersätzen und Objekten durchgeführt. Daraus hat sich ergeben, dass die Laufzeit des Planers im Mittel 26,51 ms ist, mit einer minimalen Laufzeit von 23,84 ms und einer maximalen von 34,45 ms.

2.4.2 Grenzen der Greifplanung anhand ausgewählter Objekte

Bei diesem Versuch ist mit den verschiedensten Objekten (Abbildung 11) eine Greifplanung durchgeführt worden. Dabei wird immer auf Basis der in Kamerabild eingezeichneten ausgewählten Greifregionen entschieden, ob die Planung erfolgreich war, das heißt, ob damit ein realer Griff möglich ist.

Abbildung 11: Die Abbildung zeigt die Objekte, mit denen die Grenzen des Planers ausgetestet worden sind.

Auf Basis der Erkenntnisse aus Abschnitt 2.4.1 ist die Hough-Transformation zweimal mit den beiden verschiedenen Parametersätzen ausgeführt und danach erst die Einteilung in die Greifregionenpaare vorgenommen worden. Damit kann sowohl für klein abgebildete Objekte, als auch für groß abgebildete Objekte gut geplant werden. Für jedes Objekt wurde die Planung inklusive Objektmodellierung zehnmal durchgeführt, mit immer neuen Objektlagen. Dabei wurde gezählt, wie viele Planungen erfolgreich waren und wie viele fehlgeschlagen sind.

Die Abbildungen 12 und 13 zeigen die jeweils besten Planungserfolge. Für jedes Objekt ist die Anzahl der Erfolge und Misserfolge angegeben.

Bei der Taschenlampe (Abbildung 12(b)) wurden sehr wenige Planungen erfolgreich abgeschlossen. Die metallisch-spiegelnde Oberfläche macht eine

gute Kantenerkennung schwierig. Bei der Bestimmung der minimalen Boundingbox wird auch der bewegliche Anhänger hinten berücksichtigt. Damit wird das Objektmodell deutlich größer, als das eigentliche Objekt. Bei dem Kochlöffel (Abbildung 13(c)) hat die Objektmodellierung gänzlich versagt.

Die Kinderrassel (Abbildung 12(j)) hat eine sehr schwierige Kontur, da es kaum gerade Kanten gibt. Deshalb ist die Zahl der Misserfolge hoch. Bei dem Salzstreuer (Abbildung 13(b)) ist aus dem gleichen Grund sogar überhaupt kein einziger Erfolg erzielt worden. Der Brieföffner (Abbildung 13(f)) hat ebenso wenige gute Kanten beziehungsweise spiegelt dessen Schwert zu sehr, als dass eine gute Kantenerkennung möglich wäre.

Bei der Nikolaus-Puppe (Abbildung 13(e)) wurden erstaunlich viele Planungserfolge erzielt, obwohl mit dem bloßen Auge keine deutlichen parallelen Kanten zu erkennen sind.

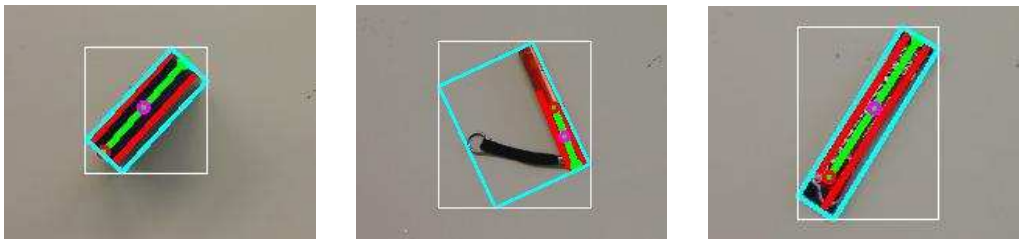
Bei dem Salz- und Pfefferstreuer-Kästchen (Abbildung 13(a)) kommen die vielen Misserfolge von der fehlerhaften Objektmodellierung. Das gleiche Problem besteht bei der Taschentuchpackung (Abbildung 13(i)).

Für alle anderen Objekte wurde die Planung mit einer sehr hohen Erfolgsquote durchgeführt. Selbst für den Zollstock (Abbildung 12(h)), der sehr viele äußerst lange parallele Kanten besitzt, sind immer die Außenkanten als Greifregionen für den besten Kandidat gewählt worden.

2.5 Schlussfolgerung

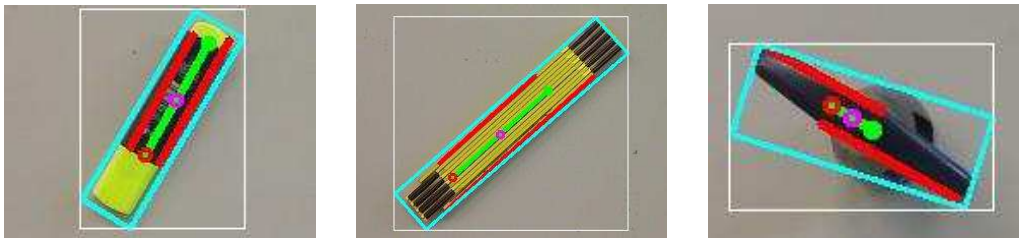
Es ist ein Greifplaner entwickelt worden, der auf Basis eines einzigen Kamerabildes, den Kameraparametern und der bekannten Aufgreiffläche eine zweidimensionale Greifplanung durchführt. Dazu wird ein bildbasiertes Objektmodell erstellt, das die Kantenpixel des Objekts und die minimal Boundingbox beinhaltet. Mit Hilfe der Hough-Transformation werden aus den Kantenpixeln Greifregionpaare extrahiert, die parallele Strecken repräsentieren. Es wird das Greifregionenpaar r_1, r_2 gewählt, das die längste gemeinsame Strecke $s_{\text{gemeinsam}} = \Omega \circ \Psi(\Xi(r_1), \Xi(r_2))$ hat. Durch die Experimente wurde gezeigt, dass für Objekte mit gut detektierbaren Kanten die Planung erfolgreich ist. Dabei dürfen sogar Kanten innerhalb der Objektgrenzen vorhanden sein.

Bei einfachen Objekten, wie dem schwarzen Klotz (Abbildung 12(a)), arbeitet der Greifplaner hervorragend. Es wird immer erfolgreich mit den augenscheinlich optimalen Kanten geplant, da überhaupt nur vier Konturkanten vorhanden sind. Von diesen vier Kanten sind jeweils zwei gleich lang und



(a) Schwarzer Klotz (10 / 0) (b) Taschenlampe (2 / 8) (c) Kaffepulver (8 / 2)

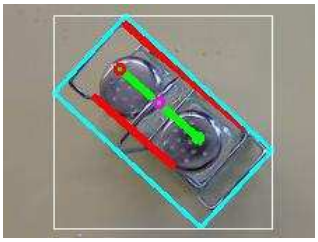
(d) Kugelschreiber (10 / 0) (e) Gelbes Feuerzeug (5 / 1) (f) mp3-Player (10 / 0)



(g) Textmarker (10 / 0) (h) Gelber Zollstock (9 / 1) (i) Korkenzieher (8 / 2)

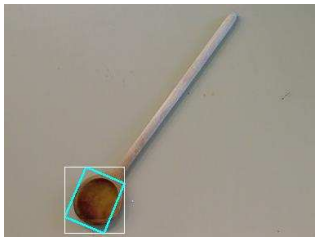
(j) Kinderrassel (7 / 13) (k) Kosmetikflasche (9 / 1) (l) Tacker (10 / 0)

Abbildung 12: Jedes Bild zeigt in Rot die beiden Strecken der Greifregionen, in Grün die gemeinsame Strecke, in Rosa deren Mittelpunkt, in Hellblau die minimale Boundingbox und in Weiß den Bildausschnitt, der für die Hough-Transformation relevant ist. Für jedes Bild ist in Klammern die Zahl der Erfolge zu der Zahl der Misserfolge angegeben.

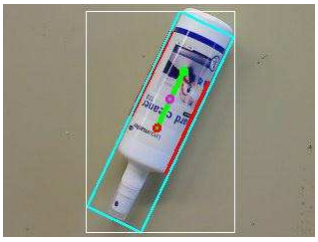


(a) Salz- und Pfefferstreuer Kästchen (6 / 4)

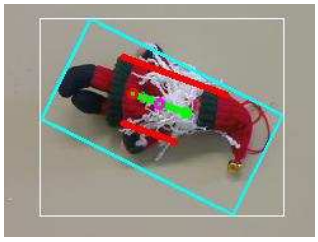
(b) Salzstreuer (0 / 10)



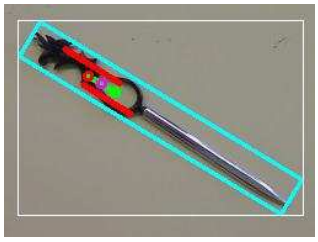
(c) Kochlöffel (0 / 10)



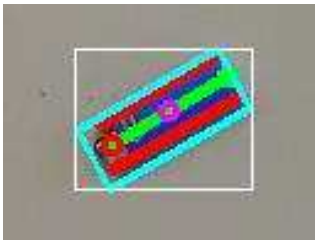
(d) Sprühflache (8 / 2)



(e) Nikolaus-Puppe (3 / 7)

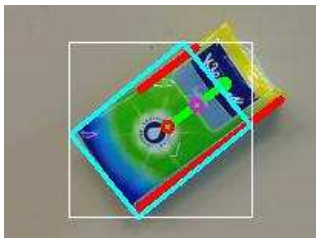


(f) Brieföffner (3 / 7)



(g) Blaues Feuerzeug (8 / 2)

(h) Rote Sortierbox (10 / 0)



(i) Taschentuchpackung (2 / 1)

Abbildung 13: Jedes Bild zeigt in Rot die beiden Strecken der Greifregionen, in Grün die gemeinsame Strecke, in Rosa deren Mittelpunkt, in Hellblau die minimale Boundingbox und in Weiß den Bildausschnitt, der für die Hough-Transformation relevant ist. Für jedes Bild ist in Klammern die Zahl der Erfolge zu der Zahl der Misserfolge angegeben.

parallel. Sobald das Objekt auch innere Kanten hat, wie die Fernbedienung oder die Sprühflasche, werden die Ergebnisse des Planers schlechter. Dennoch kann meistens die Planung erfolgreich abgeschlossen werden. Bei Objekten mit sehr welligen und unscharfen, aber klaren Kanten auf der Textur des Objekts, wie der Taschentuchpackung, ist nur in wenigen Ausnahmen die Planung erfolgreich.

Der Planer berücksichtigt implizit auftretende Positionierungenauigkeit durch die Qualitätsmetrik, da die längsten Kanten bevorzugt werden. Es wird auch nicht die Einschränkung vorgenommen, dass nur ein punktueller Kontakt zwischen Greiferbacken und Objekt besteht. Vielmehr werden Kandidatenpaare verworfen, bei denen die Flächen an denen der Greifer anliegt, zu kurz sind. Durch die Hough-Transformation werden Ungenauigkeiten, wie fehlende Kantenpixel oder Rauschpixel, des Objektmodells kompensiert.

Die Zeit, die für einen Planungsdurchlauf benötigt wird, liegt bei maximal 34,45 ms. Das entspricht etwa der Zeit, die zwischen zwei Bildern liegt, die mit einer Kamera aufgenommen wurden, die 30 fps leistet. Dieses Zeitintervall ist so kurz, dass es vom Menschen fast nicht wahrgenommen wird. Damit der Planer laut Aufgabenstellung schnell.

Durch die Einschränkung auf ein einziges Kamerabild und damit auf den zweidimensionalen Raum kann nicht garantiert werden, dass eine Kante im Bild auch einer realen Kante am Objekt entspricht. Bei Objekten, die an den Außenkanten im Bild in der Realität keine Fläche haben, an der der Greifer anliegen kann, wie beispielsweise die Sortierbox (Abbildung 13(h)), werden nur gegriffen, da der Greifer das gesamte Objekt umschließt. Das Objekt ist allerdings nicht so gegriffen, wie es geplant war. Dieser Fehler wirkt sich unmittelbar auf alle weiteren Aufgaben aus, die von dem Griff des Objektes abhängen, wie beispielsweise das wieder ablegen.

Das aktuelle Planungsverfahren sucht ausschließlich nach geraden Kanten. Dabei wäre es durchaus möglich einen kraftschlüssigen Griff auch bei leicht gebogenen Kanten durchzuführen. Eine leicht gebogene Linie ist im Hough-Raum nicht mehr ein Punkt, wie eine Gerade dort repräsentiert wird, sondern eine Region. Solche Regionen könnten von dem Greifplaner berücksichtigt werden. Damit eröffnen sich ganz neue Objekttypen, die nun auch gegriffen werden können. Neue Qualitätsmetriken wie die Krümmung der einzelnen Kanten sind vorstellbar. Damit ist der Schritt nicht mehr weit, auch für runde Objekte eine Greifplanung durchzuführen. Diese kann auch mit Hilfe der Hough-Transformation erfolgen. Es muss dafür nur die Parametrisierung des Hough-Raums geändert werden. Anstatt Winkel zwischen der Normalen einer Geraden zur x-Achse und des kleinsten Abstandes zum Ursprung von

der Geraden, wäre der Radius und die Koordinaten des Kreismittelpunktes eine mögliche Parametrisierung.

Durch ein dreidimensionales Objektmodell, würden die Fehlplanungen auf Grund von irrtümlicherweise verwendeter Greifregionen, die in der Realität nicht existieren, vermieden werden. Das Objekt könnte durch eine Punktwolke repräsentiert sein. Mit der Hough-Transformation kann dann in dieser Punktwolke nach parallelen Ebenen gesucht werden. Dabei wären die Hough-Parameter beispielsweise die Kugelkoordinaten des Lotfußpunktes vom Ursprung des Koordinatensystems auf die Ebene. Die gekrümmten Strecken aus dem zweidimensionalen lassen sich auch einfach auf gekrümmte Flächen im 3D erweitern. Die Stärke der Hough-Transformation auch mit ungenauen oder unvollständigen Daten zurecht zu kommen, würde erhalten bleiben.

Kapitel 3

Ablegen

Die Ablageplanung hat zum Ziel, ein bereits gegriffenes Objekt in einer vorgegebenen Umweltsituation wieder abzulegen. Der Planer soll auf einer bekannten Ablagefläche die Position und Orientierung des Objekts berechnen, wie es abgelegt werden soll. Dazu muss die Ablagekonfiguration berechnet werden, die die Position und Orientierung des Greifers festlegt. Für diese Aufgabe stehen die Informationen über das Objekt aus Kapitel 2 zur Verfügung. Diese umfassen das Objektmodell, das im Zuge der Greifplanung erstellt worden ist. Des Weiteren ist bekannt, wie und wo das Objekt aufgegriffen worden ist, damit aus dem bildbasierten Objektmodell die reale Größe des Objekts berechnet werden kann. Das Objekt soll so abgelegt werden, dass möglichst viele auf der Ablagefläche Platz haben, aber gleichzeitig ein problemloses Aufgreifen durch den Roboter zu ermöglichen.

In diesem Kapitel wird zuerst in 3.1 ein Überblick über den Stand der Forschung gegeben. Danach wird die Problemstellung formuliert und im Abschnitt 3.3 ein Algorithmus vorgestellt, der das Problem löst. Dessen Fähigkeiten und Grenzen werden in 3.4 getestet. Zum Schluss wird die Schlussfolgerung bezogen (Abschnitt 3.5).

3.1 Stand der Froschung

Heutzutage werden Roboter hauptsächlich im industriellen Umfeld und nur sehr selten im privaten Umfeld eingesetzt. In der Industrie werden Roboter zur Bestückung von Platinen, Montage von Bauteilen aller Art verwendet. Dafür muss festgelegt werden, wie und wo das zu platzierende Objekt abgelegt, eingebaut oder abgeworfen werden soll. In [Whelan 96] wird das gesam-

te Gebiet in Handhabungsaufgaben, Ver- und Auspack-, Speicherungs- und Transport- und Zu- und Ausschneideaufgaben unterteilt. Jedes dieser vier Gebiete wird nochmals unterteilt und Veröffentlichungen angegeben, die sich mit dem jeweiligen Problem auseinandersetzen.

Im Folgenden wird sich auf das dichte Packen von Objekten beschränkt. Generell gibt es drei Varianten das Packproblems. In Abbildung 14 sind schematisch die drei Problemtypen dargestellt: das Strip Packing Problem (SPP) (Abbildung 14 (a)), das Container Loading Problem (CLP) (Abbildung 14 (b)) und das Bin Backing Problem (BPP) (Abbildung 14 (c)). In blau ist jeweils der Behälter dargestellt, in den die Boxen (schwarz) eingepackt werden müssen.

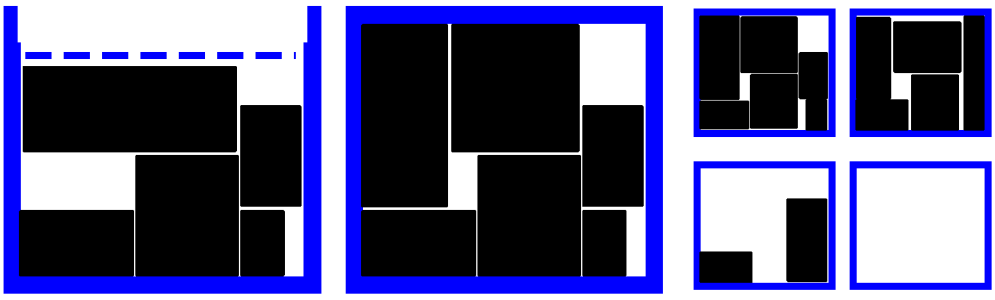


Abbildung 14: Schematische Darstellung der drei Varianten des Packproblems: (a) Strip Packing Problem: minimieren der Höhe die benötigt wird (gestrichelte Linie). (b) Container Loading Problem: minimieren des Freiraums. (c) Bin Packing Problem: minimieren der Anzahl der Container. In Blau sind die Behälter/Container dargestellt und in schwarz die Boxen.

Wie unterscheiden sich diese Probleme? Der Behälter in den die Boxen gepackt werden müssen, ist beim SPP oben offen. Die gestrichelte Linie in der schematischen Abbildung zeigt die erreichte Höhe der Packung an. Diese Höhe soll möglichst minimal sein. Beim CLP ist der Behälter oder Container zu allen Seiten geschlossen. Die Aufgabe ist, den Freiraum innerhalb des Containers zu minimieren. Das BPP ist eine Erweiterung des CLP. Die Anzahl der Container ist nicht mehr auf eins beschränkt. Das Ziel ist es, bei diesem Problem möglichst wenige Container zu verwenden. Bei allen drei Varianten sind bei der klassischen Problemstellung alle Boxen, die zu packen sind, bekannt.

In diesem Abschnitt werden Arbeiten vorgestellt, die sich mit allen drei Problemvarianten beschäftigen. Dabei werden die Arbeiten nach folgenden Fra-

gen beurteilt: Welche Objekte können gepackt werden? Ist die Packreihenfolge *roboterpackbar*, d.h. können die Objekte durch einen Roboter abgelegt werden? Wie lange ist die Laufzeit der Algorithmen? Eignet sich der Algorithmus die Platzierung der Boxen online zu berechnen?

Nachdem das Packproblem NP-hard [Bischoff 95b] ist, werden Heuristiken oder Metaheuristiken verwendet, um gute Lösungen für das Problem zu finden. In [Lodi 02b, Ortmann 10] werden einige Heuristiken und Metaheuristiken für das BPP vorgestellt und verglichen. In [Lodi 02a] werden neben einem Überblick über Lösungen für das BPP auch einige für das SPP beleuchtet. Hyde [Hyde 10] stellt in seiner Doktorarbeit einen hyper-heuristischen Ansatz vor, um alle Varianten der Packprobleme zu lösen. Hyper-heuristisch bedeutet, dass aus bestehenden Heuristiken die gewählt wird, die am besten zu dem gegebenen Problem passt, oder, dass, falls nötig, die bestehenden zu neuen kombiniert werden. In dessen Arbeit wird eine Möglichkeit vorgestellt, automatisch Heuristiken zu erzeugen und bestehende zu kombinieren. Dafür werden genetische Algorithmen verwendet. Diese Arbeiten beschränken sich auf das Design von Heuristiken, die ausschließlich die klassische Problemstellung bestmöglich lösen sollen. Dabei werden Forderungen wie die Roboterpackbarkeit nicht berücksichtigt. Es gibt diverse Verfahren [Bischoff 95b, Bischoff 95a, Bischoff 06, Eley 02], die das klassische CLP mit rechteckigen Boxen behandeln. Sie sind jedoch alle nicht darauf ausgelegt, dass die Boxen im Vorfeld unbekannt sind.

In dem Verfahren aus [Parreño 08] werden nicht direkt Positionen, an denen Boxen angelegt werden können, betrachtet, sondern *Freiräume*, auf die eine Box gestellt werden kann. Diese Freiräume sind rechteckige Flächen (2D) oder Quader (3D), die immer maximal groß gewählt werden. Dabei dürfen sich diese Freiräume überlappen. Der dort vorgestellte Algorithmus sucht für einen Freiraum die am besten passenden Boxen heraus. Dafür sind alle einzusortierenden Boxen nach Abmessungen in Typen gruppiert. Es werden so viele Boxen eines Typs in den gewählten Freiraum gepackt, bis keine mehr in diesen passt. Basierend auf einer *Zielfunktion* wird der am besten geeignete Boxtyp, die Anzahl der Boxen und deren Platzierung gewählt. Falls nicht eingepackte Boxen vorhanden sind, wird eine gewisse Anzahl an Boxen (25% - 75% aller Boxen) aus dem Container ausgepackt. Die zuvor nicht gepackten Boxen werden dann in bei wiederholten Packen zuerst eingepackt und dann werden die restlichen Boxen berücksichtigt.

In [Bouganis 06] wird ein Algorithmus vorgestellt, der das BPP für nicht rechteckige zweidimensionale Objekte löst. Torra et al [Torra 09] haben eine Lösung für das CLP für nicht rechteckige Objekte im dreidimensionalen

Raum. Es wurden nur Testfälle mit konvexen Polyedern gezeigt. In [Egeblad 10] wird das CLP noch allgemeiner gelöst. Es wird eine Heuristik vorgestellt, die das Problem im dreidimensionalen Raum für Möbel löst. Diese Objekte sind nicht mehr konvex wie in [Torra 09]. Sie verwenden eine Reihe von Sub-Heuristiken, die jeweils auf ganz bestimmte Objektklassen beschränkt sind. Diese werden dann zu einer Gesamtheuristik zusammengefasst. Der Algorithmus erreicht nach eigenen Angaben eine durchschnittliche Platzausnutzung von 91 %. Die Laufzeit des Algorithmus ist mit 100 Sekunden angegeben. Sie ist vor allem nicht dazu geeignet, online Objekte zu packen, da einzelne Objekte anfangs zu Templates zusammengefasst werden und somit die zu packenden Objekte vorsortiert werden.

Die bisher vorgestellten Veröffentlichungen berücksichtigen allerdings nicht, dass die Objekte durch einen Roboter packbar sein sollen. Diese Anforderung ist zwar nicht Teil der klassischen Packprobleme, aber für diese Arbeit essentiell. Martello et al haben sich in [Martello 00] mit dieser zusätzlichen Anforderung auseinandergesetzt. In [Martello 07] wird dieser Ansatz noch verallgemeinert. Es werden zwei Algorithmen vorgestellt, die eine Lösung für das BPP erarbeiten. Der erste verwendet Ebenen. Für jeden Container ($H \times W \times T$ - Höhe x Breite x Tiefe) wird für die Ebenen $H \times W$ das zweidimensionale BPP gelöst. Für alle diese gefüllten Ebenen wird dann das eindimensionale BPP gelöst, um die Ebenen in die eigentlichen Container zu packen. Der zweite Algorithmus ist ein „greedy“-Algorithmus für das BPP, der für jeden Container das CLP löst. Um die Roboterpackbarkeit zu gewährleisten, treffen sie die Annahme, dass, ausgehend von einer Ecke, die Boxen immer nur an drei Seiten angelegt werden dürfen. Die Seiten werden eindeutig, sobald die erste Box in die Ecke gelegt wurde; die drei Seiten sind nun genau die, die nicht an einer Containerwand anliegen. Es wird allerdings nicht beschrieben, wie dann ein Roboter die Boxen wirklich einladen kann, ohne mit anderen Boxen oder dem Container zu kollidieren.

In [Martello 00] wird festgestellt, dass alle Packungen, die guillotine-schneidbare Packungen sind auch roboterpackbar sind. Diese Arbeit gibt auch den ersten Einstieg in das Feld dieser theoretischen Problemstellung.

In den bisherigen Arbeiten wurden die Packprobleme hauptsächlich auf eine sehr theoretische Art und Weise behandelt oder in einer Weise, die nur für den industriellen Einsatz geschaffen ist. Es gibt etliche Ansätze, mit den unterschiedlichsten Verfahren gute Lösungen für die klassischen Varianten der Packprobleme zu finden. In einigen Arbeiten können nicht nur reguläre Boxen gepackt werden, sondern irreguläre, nicht konvexe Objekte, wie Möbel. Die Anzahl der Ansätze, die eine Roboterpackbarkeit ermöglichen, sind bisher

beschränkt. Im industriellen Umfeld wird mit speziellen Greifern oder sogar Spezialrobotern gearbeitet.

Bei allen Arbeiten sind die zu packenden Boxen oder Objekte vorher bekannt. Die vorgestellten Algorithmen eignen sich nicht zur Lösung des online Packproblems, da sie normalerweise ein Vorsortierung auf der Menge der zu packenden Boxen vornehmen oder gar das Objekt für eine gestimmte Position auswählen, oder nachträglich einen Optimierungsschritt durchführen, indem sie schon gepackte Objekte wieder entfernen und andere hinzufügen. Eine solche Praxis ist nur möglich, wenn das Problem offline gelöst wird. Ein ständiges Aus- und Einpacken ist in der Realität nicht praktikabel.

Da in dieser Arbeit nur mit unbekannten Objekten gearbeitet wird, kann keiner der vorhanden Ansätze genutzt werden. Der Algorithmus muss ohne Vorwissen über die Objekte diese online packen können. Als Objektmodelle werden, wie auch bei den klassischen Packproblemen, Boxen verwendet. Des Weiteren muss die Objektanordnung roboterpackbar sein und auch wieder durch einen gewöhnlichen Industrieroboter mit einem Backengreifer möglich sein.

3.2 Problemanalyse

Ziel ist es, ein Objekt auf eine vorgegebene Fläche abzulegen. Dabei soll beachtet werden, dass die Objekte möglichst dicht gepackt sind. Diese Anforderungen entsprechen dem CLP im zweidimensionalen Raum. Die weitere Anforderung, die das Problem erheblich schwieriger macht, ist, dass die Anzahl der Objekte und deren Größe im Vorfeld nicht bekannt sind. Wie sieht nun die Ausgangssituation beim zweidimensionalen online CLP aus? Es ist eine Ablagefläche gegeben, in den die Objekte gepackt werden sollen. Als Objekte werden hier nur Boxen erlaubt. Die Box soll durch den Roboter auf die Fläche gelegt werden und auch wieder aufgegriffen werden können. Die Frage, die für jede Box beantwortet werden muss, ist: Wo und wie wird die Box innerhalb der Ablagefläche platziert, um möglichst viele Boxen auf die Fläche zu bekommen?

Innerhalb der Ablagefläche kann man im Grunde zwei Strategien verfolgen, um Plätze zu verwalten, an denen Boxen platziert werden können. Einmal kann man explizit Positionen angeben, an denen eine Box als nächstes platziert werden kann. Aus dieser Menge von Plätzen muss nun ein passender gewählt werden. Oder man geht einen anderen Weg und verwaltet die Freiräume innerhalb der Ablagefläche. Man wählt nicht aus einer Men-

ge von Plätzen, an denen eine Box gelegt, sondern man stellt Freiraum zur Verfügung, in dem die Box platziert werden kann.

Die Auswahl des Platzes, an dem die Box abgelegt werden soll, erfolgt durch heuristische Verfahren. Dabei müssen verschiedene Kriterien beachtet werden. Das offensichtlichste ist wohl, dass die Box überhaupt auf diesen Platz passt und dabei nicht mit der Umwelt oder anderen Boxen kollidiert. Des Weiteren soll die Ablageposition so gewählt werden, dass möglichst wenige nicht mehr erreichbare Zwischenräume entstehen und, dass die Box auf der Ablageposition stabil liegen bleibt. Es muss weiter gewährleistet sein, dass der Roboter die Box an ihrem Bestimmungsort platzieren kann. In diesem Fall muss entschieden werden, wo die Box innerhalb des gegebenen Freiraums platziert werden soll. Hier wird die zweite Variante mit den *Freiräumen* bevorzugt.

Bei der *Roboterpackbarkeit* selbst müssen auch einige Randbedingungen beachtet werden. Die wohl wichtigste Rolle spielt der Typ des Greifers und dessen Geometrie. Je mehr Freiheitsgrade der Greifer selbst hat, desto mehr Möglichkeiten gibt es, wie das Objekt gegriffen ist. Wenn man einen Greifer betrachtet, der der menschlichen Hand nachempfunden ist, hat man viele Möglichkeiten, wie ein Objekt gegriffen sein kann. Hat man nur einen Backengreifer sind die Möglichkeiten sehr begrenzt. Des Weiteren muss beachtet werden, dass der Greifer selbst nicht mit anderen Objekten und auch nicht mit der restlichen Umwelt kollidiert.

Der Ablageplaner verwaltet eine *Liste von Freiräumen* L_{FR} aus der mittels einer *Zielfunktion* der Freiraum $f_{\mathrm{best}} \in L_{\mathrm{FR}}$ gewählt wird, in den das Objekt platziert werden soll. Mit einer Heuristik wird dann die genaue Position des Objektes innerhalb des Freiraums festgelegt. Die *Zielfunktion* stellt sicher, dass die harten Anforderung an eine Ablageposition eingehalten werden. Die Heuristik muss so arbeiten, dass sie keine der gegebenen Bedingungen verletzt. Aus der geplanten Platzierung des Objekts wird die Ablagekonfigurations des Greifers berechnet.

In Abbildung 15 wird der gesamte Ablauf der Ablageplanung bis hin zur Berechnung der Ablagekonfiguration in einem Flussdiagramm dargestellt. Für jedes Objekt wird ein Freiraum bestimmt in den es gepackt werden kann. Wird keiner gefunden, so passt das Objekt nicht mehr auf die Ablagefläche. Wenn ein Freiraum gefunden wird ist klar, dass das Objekt auch dort abgelegt werden kann.

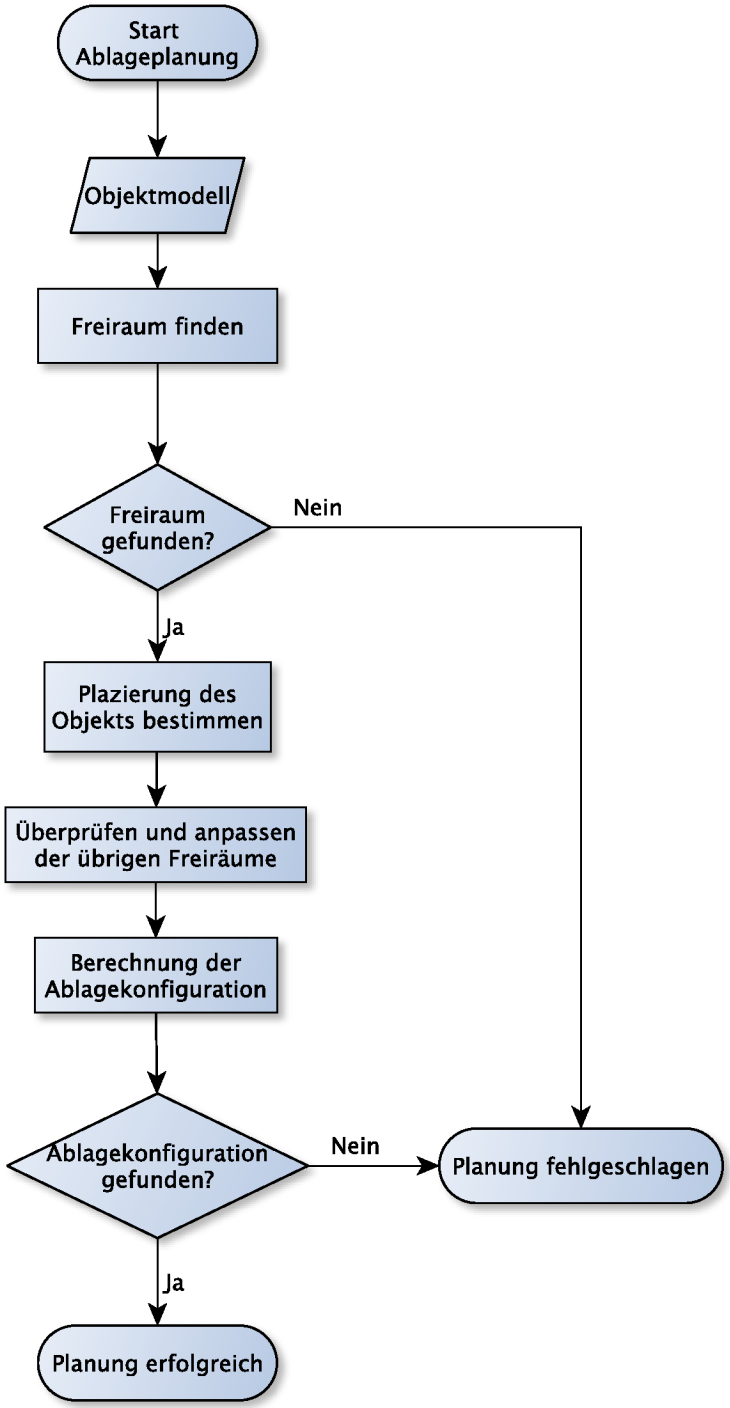


Abbildung 15: Ablauf der Ablageplanung dargestellt in einem Flussdiagramm

3.3 Umsetzung

Für den hier entwickelten Algorithmus wird ohne Einschränkung angenommen, dass die Ablagefläche und die zu packenden Boxen achsen-parallel ausgerichtet sind. Sie werden durch achsen-parallele Boundingboxen („axis-alligned boundingbox“, AABB) $(p_{\min}, p_{\max})$ repräsentiert. Dabei sind $p_{\min}, p_{\max} \in \mathbb{R}^2$ und die Koordinaten von $p_{\min}$ sind komponentenweise kleiner der Koordinaten von $p_{\max}$. Damit ist M_{AABB} die Menge der achsen-parallelen Boundingboxen.

Diese Einschränkung bei der Planung ist für das reale Platzieren durch den Roboter keine, da durch eine einfache Transformation die an den Achsen ausgerichtete Ablagefläche in die reale Position und Orientierung überführt werden kann. Diese Transformation wendet man auch auf die Ablagekonfigurationen an und hat somit auch bei diesen die Verknüpfung zwischen der achsen-parallelen Ausrichtung und der realen Ausrichtung.

Nachdem der Algorithmus nicht nur ein theoretisches Problem lösen, sondern auch in der Realität mit Hilfe eines Industrieroboters die Objekte platzieren soll, wird in 3.3.1 auf das Objektmodell und auf die Schritte eingegangen wie eine AABB aus dem Objektmodell aus Kapitel 2 berechnet wird. Im Abschnitt 3.3.2 wird der entwickelte Algorithmus vorgestellt.

3.3.1 Anpassung des bildbasierten Objektmodells

Das *bildbasierte Objektmodell* $O = (O_{\text{P}}, O_{\text{B}}, s_{\text{gemeinsam}})$ besteht am Ende des Kapitels 2 aus der minimalen Boundingbox O_{B}, den Konturpixeln O_{P} und der gemeinsamen Strecke, mit der die Greifkonfiguration berechnet wurde.

Aus dem bildbasierten Objektmodell muss die Box berechnet werden, mit der der Algorithmus arbeitet. Dazu wird die minimale Boundingbox O_{B} auf die Aufgreiffläche projiziert. Die Abbildung 16 zeigt die noch gedrehte Boundingbox. Die Box wird nun abhängig von den Winkeln α und β um den Mittelpunkt von $s_{\text{gemeinsam}}$ gedreht.

Falls $\alpha < \beta$ wird im mathematisch positiven Sinn um den Mittelpunkt von $s_{\text{gemeinsam}}$ mit dem Winkel α gedreht, andernfalls wird um den Winkel $-\beta$ gedreht. Damit ist die Boundingbox achsenparallel ausgerichtet. Diese wird im Folgenden mit $\widehat{O}_{\text{B}} \in M_{\text{AABB}}$ angegeben. Damit ist das *angepasste Objektmodell* $\widehat{O} = \{\widehat{O}_{\text{B}}, s_{\text{gemeinsam}}\}$. Die Strecke $s_{\text{gemeinsam}}$ wird relative zum minimalen Punkt $p_{\min}$ der AABB $\widehat{O}_{\text{B}}$ angegeben. Jetzt ist es ausreichend die

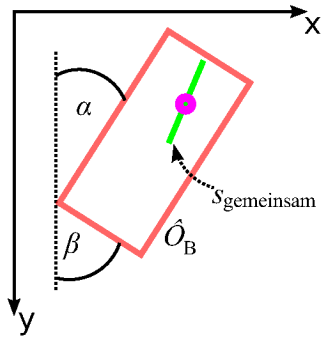


Abbildung 16: Objektmodell mit Winkeln zur y-Achse (α, β), der gemeinsamen Strecke $s_{\text{gemeinsam}}$ und der minimalen Boundingbox $\widehat{O}_{\text{B}}$

Position und Orientierung von $\widehat{O}_{\text{B}}$ anzugeben. Die Konturpixel O_{P} aus dem bildbasierten Objektmodell werden für die Ablageplanung nicht benötigt.

3.3.2 Planung

Nachdem das Objektmodell so angepasst worden ist, dass es nun die Informationen bereit stellt, die für die Ablageplanung benötigt werden, kann der eigentliche Algorithmus vorgestellt werden. In diesem Abschnitt wird zuerst definiert, wie in dem vorliegenden Fall ein Freiraum repräsentiert wird, mit dem der Algorithmus arbeitet. Dann wird der Ablauf des Algorithmus beschrieben und auf dessen Einzelkomponenten eingegangen.

Ein *Freiraum* f besteht aus drei Komponenten, aus einer maximal großen AABB B_f, die den Bereich auf der Ablagefläche definiert, der frei ist, einem Eckpunkt p_{nah}, in diese Ecke das Objekt innerhalb des Freiraums platziert wird und d_f, einem Abstand, der zur Bewertung des Freiraums hergenommen wird. Die Repräsentation des freien Raumes durch die Freiräume ist nicht eindeutig. Die AABBs der verschiedenen Freiräume überschneiden sich.

Um einen Freiraum zu definieren müssen erst einige Funktionen eingeführt werden. Die Funktion Γ berechnet den Abstand zwischen zwei Punkten einzeln für jede Koordinatenachse.

$$\begin{aligned} \Gamma : \mathbb{R}^2 \quad &\times \quad \mathbb{R}^2 \to \mathbb{R}^2 \\ \Gamma(a, b) \quad &\mapsto \quad d = (|x_a - x_b|,\ |y_a - y_b|) \,\text{mit}\, a = (x_a,\ y_a),\ b = (x_b,\ y_b) \end{aligned}$$

Sei S die Funktion, die die Koordinaten eines Punktes aus dem $\mathbb{R}^2$ lexiko-

grafisch sortiert.

$$\begin{aligned} S : \mathbb{R}^2 &\rightarrow \mathbb{R}^2 \\ S(v) &\mapsto \begin{cases} (x,y) & \text{wenn } x \neq y \\ (y,x) & \text{sonst} \end{cases}, \text{mit } v = (x,y) \end{aligned}$$

Weiter sei Λ die Funktion, die die Koordinaten der Ecke p einer achsenparallelen Boundingbox $B \in M_{\text{AABB}}$ berechnet, deren euklidischer Abstand zu einer Ecke der anderen Boundingbox B' am kleinsten ist.

$$\begin{aligned} \Lambda : M_{\text{AABB}} \times M_{\text{AABB}} &\rightarrow \mathbb{R}^2 \\ \Lambda(B, B') &\mapsto p \end{aligned}$$

Ein *Freiraum* f ist nun definiert als

$$\begin{aligned} f := \{&(B_{\text{f}}, p_{\text{nah}}, d_{\text{f}}) | B_{\text{f}} \in M_{\text{AABB}}, \\ &p_{\text{nah}} = \Lambda(B_{\text{f}}, C), \\ &d_{\text{f}} = S \circ \Gamma(\Lambda(B_{\text{f}}, C), \Lambda(C, B_{\text{f}}))\}. \end{aligned}$$

Dabei ist B_f die achsen-parallele Boundingbox, d_f ist der lexikografisch kleinste Abstand einer Freiraumecke zu einer Ecke der Ablagefläche und p_{nah} ist die Ecke, die den kleinsten Abstand zu einer Ecke der Ablagefläche $C \in M_{\text{AABB}}$ hat.

In Abbildung 17 ist ein solcher Freiraum zu sehen. In blau ist die Ablagefläche C eingezeichnet und in grau die Boundingbox B_f. Die Ecke p_{nah}, die am nächsten an C liegt, ist rot markiert. Der lexikografische kleinste Abstand d_f ist nicht eingezeichnet, aber dafür die Abstände in den Koordinatenrichtungen d_{x} und d_{y}. Auf der Abbildung wäre $d_f = (d_{\text{x}}, d_{\text{y}})$ da $d_{\text{x}} < d_{\text{y}}$.

Nachdem nun ein Freiraum genau definiert ist, wird im Folgenden beschrieben, wie die Planung abläuft. Der Algorithmus verwaltet alle Freiräume in einer Liste, der *Liste der Freiräume* L_{FR}. Zu Beginn wird angenommen, dass die Ablagefläche leer ist. Die Liste L_{FR} wird mit einem einzigen Freiraum initialisiert. Dieser Freiraum ist gerade so groß wie die Ablagefläche selbst. Damit ist die Ecke p_{nah} frei wählbar und der Abstand $d_f = (0; 0)$.

Jetzt ist der Planer bereit für ein Objekt die Ablageplanung durchzuführen. In der Abbildung 18 wird von oben nach unten der Ablauf des Algorithmus für die ersten zwei Objekte gezeigt. In blau ist immer die Ablagefläche C

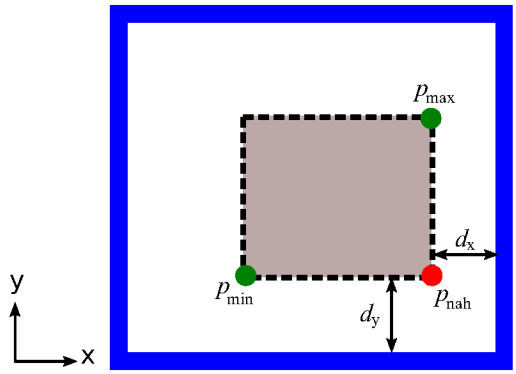


Abbildung 17: Illustration eines Freiraums innerhalb der Ablagefläche. Diese ist blau eingezeichnet, die achsen-parallele Boundingbox (grau) mit deren minimaler Ecke p_{min} und der maximalen Ecke p_{max}, in rot ist die Ecke p_{nah} eingezeichnet deren Abstand zu einer Ecke der Ablagefläche minimal ist.

eingezeichnet, in schwarz die Objekte und die anders farbigen Flächen mit der gestrichelten Umrandung sind die Freiräume. Diese sind einmal innerhalb der Ablagefläche eingezeichnet und zur Verdeutlichung nochmals rechts daneben.

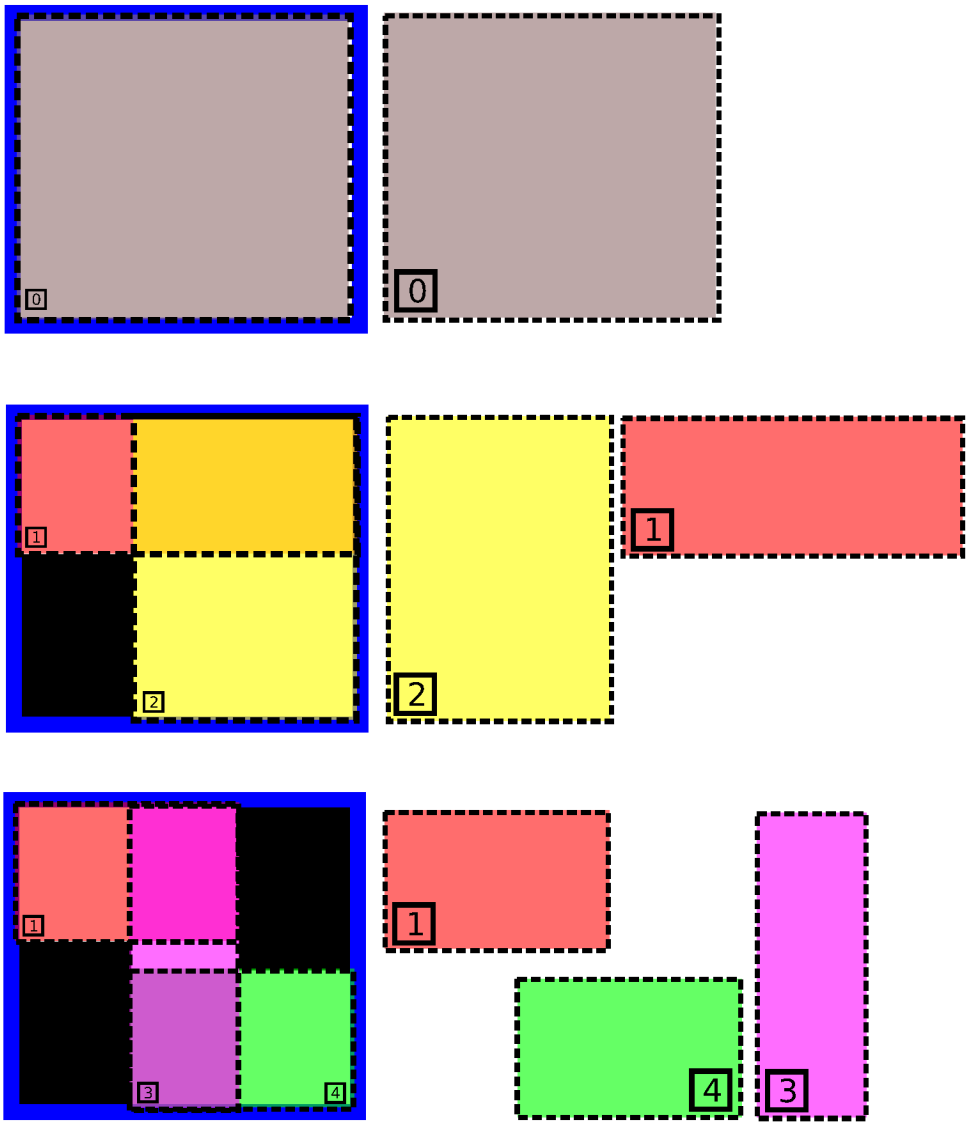


Abbildung 18: Zeigt wie zwei Objekte (schwarz) auf die Ablagefläche (blau umrandet) gelegt werden und wie dabei die Freiräume (verschiedenfarbig) verändert werden. Dabei wird in der ersten Spalte die Ablagefläche mit den Objekten und Freiräumen dargestellt und immer links daneben werden nochmals die Freiräume überlappungsfrei aufgelistet.

Für jedes neue Objekt werden nun folgende Schritte durchgeführt: Zuerst wird aus der Liste der Freiräume L_{FR} mit Hilfe einer *Zielfunktion* der Freiraum f_{best} gewählt, in den das Objekt $\widehat{O}$ platziert werden soll. In Abbildung 18 wird für das erste Objekt das abgelegt werden soll, der Freiraum mit der

Nummer 0 ausgewählt. Um einen passenden Freiraum zu finden darf das Objekt um 90 Grad gedreht werden. Als nächstes wird das Objekt in diesem Freiraum platziert und zwar genau in der Ecke p_{nah} (Abbildung 19).

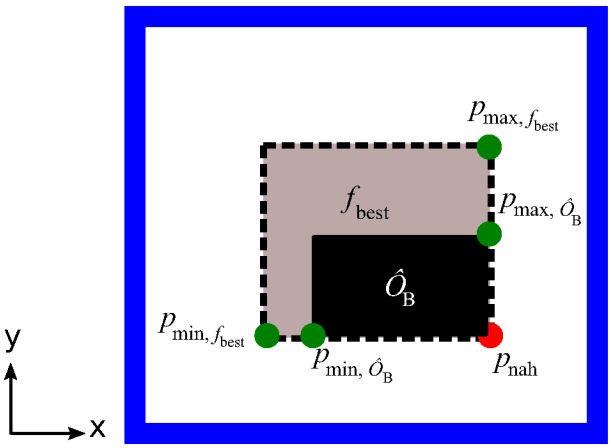


Abbildung 19: Die Boundingbox $\widehat{O}_{\text{B}}$ des Objekts wird an die Ecke p_{nah} des Freiraums (gestrichelt umrandet) gelegt, die am nächsten an einer Ecke der Ablagefläche (blau) liegt.

Der gewählte Freiraum f_{best} wird gelöscht und der Raum von f_{best}, der nicht vom Objekt bedeckt wird, wird durch zwei neue Freiräume repräsentiert. Die AABBs der beiden Freiräume können direkt aus den vorhandenen Koordinaten der AABBs des alten Freiraums f_{best} und des gerade platzierten Objekts $\widehat{O}_{\text{B}}$ abgelesen werden. Die anderen beiden Parameter werden mit Hilfe der beiden AABBs berechnet. In Abbildung 19 sehen die zwei neuen Freiräume f_1 und f_2 sehen wie folgt aus: Die AABB des ersten Freiraums würde durch $p_{\min,f_1} = p_{\min,f_{\text{best}}}$ und $p_{\max,f_1} = (x_{p_{\min},\widehat{O}_{\text{B}}}, y_{p_{\max},f_{\text{best}}})$ begrenzt werden und die AABB des zweiten durch $p_{\min,f_2} = (x_{p_{\min},f_{\text{best}}}, y_{p_{\max},\widehat{O}_{\text{B}}})$ und $p_{\max,f_2} = p_{\max,f_{\text{best}}}$. Der Wert von d_f und der Punkt p_{nah} wird für jeden der beiden neuen Freiräume, wie oben beschrieben, berechnet.

Weil sich die Freiräume überlappen, muss, nachdem ein Objekt platziert worden ist, die Liste der Freiräume L_{FR} überprüft werden. Ein bestehender Freiraum kann auf fünf verschiedene Arten Überschneidungen mit dem neu platzierten Objekt haben. In der Abbildung 20 sind alle Möglichkeiten dargestellt, wie ein Objekt einen bestehenden Freiraum schneiden kann und wie dann die neu konstruierten Freiräume aussehen. Sie sind farblich abgehoben und an den Stellen, an denen sich die neuen Freiräume unterscheiden, überlagern sich die Farben. Für jeden neu erzeugten Freiraum muss noch geprüft werden, ob dieser nicht degeneriert ist. Das heißt, dass sein Flächeninhalt leer ist. Ein Objekt kann an jeder Ecke oder jeder Kante den

Freiraum schneiden, aber auch komplett im Freiraum liegen. In der Abbildung 18 wird, nachdem das zweite Objekt im Container platziert worden ist, der Freiraum mit der Nummer 1 verkleinert. Das entspricht dem Fall aus Abbildung 20(c).

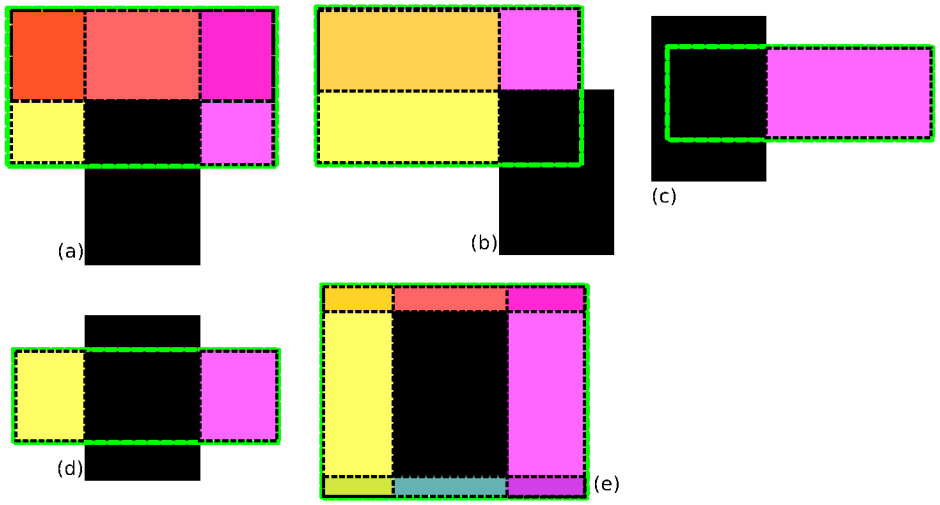


Abbildung 20: Auflistung der Möglichkeiten, wie ein existierender Freiraum (grün umrandet) durch neue Freiräume (halbtransparent und verschiedenfarbig) ersetzt werden muss, wenn dieser von einem platzierten Objekt (schwarz) geschnitten wird. In (a) entstehen drei neue Freiraume: gelb, rot und lila. In (b) und (d) entstehen zwei: gelb und lila. In (c) einer: lila. Und in (e) vier: rot, gelb, lila und hellblau.

Während der Manipulation der Freiräume kann es vorkommen, dass ein Freiraum komplett innerhalb eines anderen liegt. In diesem Fall wird der kleinere aus L_{FR} entfernt.

Nachdem die Freiräume alle aktualisiert worden sind, wird die Listw L_{FR} anhand der d_{f} aus jedem Freiraum aufsteigend sortiert. Ein Freiraum f_1 wird vor einem anderen Freiraum f_2 einsortiert, wenn für $d_{\mathrm{f},1} = (x_1, y_1)$ aus f_1 und $d_{\mathrm{f},2} = (x_2, y_2)$ aus f_2 gilt, entweder ist $x_1 < x_2$, oder $x_1 = x_2$ und $y_1 < y_2$. Man schreibt dann $d_{\mathrm{f},2} \prec d_{\mathrm{f},2}$ Damit steht der Freiraum an erster Stelle, dessen Komponenten von d_{f} am kleinsten sind. In der Abbildung 18 ist, wie schon erwähnt, der Ablauf des Algorithmus für die ersten zwei Objekte dargestellt. Dabei sind die Freiräume rechts von dem Container so angeordnet, dass sie ihrer Position in L_{FR} entsprechen. Ganz links steht der Freiraum, dessen Komponenten von d_{f} am kleinsten sind.

Diese Sortierung bewirkt, dass tendenziell die Objekte in Freiräume platziert

werden, die näher an den Ecken der Ablagefläche liegen. Innerhalb dieser Freiräume wird das Objekt in die Ecke p_{nah} gelegt, die bekanntlich die Ecke des Freiraums ist, die am nächsten an einer Containerecke liegt. Da bei gleicher Bewertung zweier Freiräume immer der zuerst in der Liste kommt, der näher an einer Ecke der Ablagefläche ist, wird in diesen das Objekt gelegt.

Nun ist die Planung für das Objekt erfolgt. Für weitere Objekte wird nun die Liste der Freiräume nicht neu initialisiert, sondern einfach weiter geführt. In Abbildung 18 wird dargestellt wie die ersten beiden Objekte platziert werden, wie die Freiräume auf der Ablagefläche liegen und wie sie in der Liste der Freiräume sortiert sind. Rechts von der Ablagefläche sind die Freiräume so angeordnet, wie sie auch in der Liste der Freiräume angeordnet sind.

Auswahl des Freiraums durch eine Zielfunktion

Die *Zielfunktion* wählt aus der *Liste der Freiräume* L_{FR} den Freiraum f_{best}, in den das aktuelle Objekt $\widehat{O}$ platziert werden soll. Es sind insgesamt drei verschiedene entwickelt worden, die im Folgenden vorgestellt werden.

Es müssen noch ein paar Definitionen vorgenommen werden, um die Auswahl von f_{best} formal zu erläutern. Die Funktion $V : M_{\text{AABB}} \to \mathbb{R}$ berechnet das Volumen einer achsen-parallelen Boundingbox, weiter sei $F : \widehat{O} \to \widehat{O}$ die Funktion, die ein Objekt um 90 Grad dreht und $K : \widehat{O} \times L_{\text{FR}} \to \{0, 1\}$ die Funktion, die entscheidet, ob das Objekt in den Freiraum passt.

Firstfit ist die denkbar einfachste Zielfunktion. Sie iteriert über die L_{FR} und wählt den ersten Freiraum in den $\widehat{O}$ hineinpasst. Falls beim ersten Durchlauf kein passender Freiraum gefunden wurde, wird das Objekt um 90 Grad gedreht und nochmals über die Liste iteriert.

Bestfit-Volumen iteriert über alle Freiräume L_{FR} einmal mit der gegebenen Orientierung des Objekts und einmal mit dem um 90 Grad gedrehten Objekt. Es wird der Freiraum unter allen anderen gewählt, dessen Volumendifferenz zwischen Freiraum und Objekt am geringsten ist. Der Freiraum $f_{\text{best,no Flip}}$ ist der Freiraum, der für für die ursprüngliche Orientierung des Objekts, aus L_{FR} gewählt wird. Ebenso ist $f_{\text{best, Flip}}$ der Freiraum, der für das gedrehte Objektmodell der beste Freiraum ist. Aus diesen beiden wird

anschließend f_{best} gewählt.

$$
\begin{aligned}
f_{\text{best,no Flip}} &:= \Big\{ f \in L_{\text{FR}} \Big| \forall f' \in L_{\text{FR}},\, f \neq f' : \\
&\qquad V(B_f) - V(\widehat{O}_{\text{B}}) < V(B_{f'}) - V(\widehat{O}_{\text{B}}) \\
&\qquad\quad K(\widehat{O}, f) = K(\widehat{O}, f') = 1 \Big\} \\
f_{\text{best,Flip}} &:= \Big\{ f \in L_{\text{FR}} \Big| \forall f' \in L_{\text{FR}},\, f \neq f' : \\
&\qquad V(B_f) - V \circ F(\widehat{O}_{\text{B}}) < V(B_{f'}) - V \circ F(\widehat{O}_{\text{B}}) \\
&\qquad\quad K(F(\widehat{O}), f) = K(F(\widehat{O}), f') = 1 \Big\} \\
f_{\text{best}} &:= \begin{cases} f_{\text{best,Flip}} & \text{wenn} \\ & \quad V(B_{f_{\text{best,Flip}}}) - V(\widehat{O}_{\text{B}}) < V(B_{f_{\text{best,no Flip}}}) - V(\widehat{O}_{\text{B}}), \\ f_{\text{best,no Flip}} & \text{sonst} \end{cases}
\end{aligned}
$$

Die Variablen B_f, $B_{f'}$, $B_{f_{\text{best,Flip}}}$ und $B_{f_{\text{best,no Flip}}}$ repräsentieren die AABBs der Freiräume f, f', $f_{\text{best,Flip}}$ und $f_{\text{best,no Flip}}$ und $\widehat{O}_{\text{B}}$ die Boundingbox des Objekts $\widehat{O}$. Diese Zielfunktion iteriert immer über die gesamte Liste und prüft auch immer beide Objektorientierungen. Für einen festen Freiraum, in den das Objekt in beiden Orientierungen hineinpasst, ist die Volumendifferenz identisch für beide Objektorientierungen. Es gibt aber Freiräume, in die das Objekt nur in einer der beiden Orientierungen passt, deshalb muss die Liste immer mit beiden Objektorientierungen durchlaufen werden.

Bestfit-Abstand iteriert ebenso wie Bestfit-Volumen über alle Freiräume einmal mit der gegebenen Orientierung des Objekts und einmal mit dem um 90 Grad gedrehten Objekt. Der Freiraum f_{best} wird folgendermaßen gewählt: Wie bei Bestfit-Volumen wird der am besten passende Freiraum für gedrehte und nicht gedrehte Objekt berechnet und dann aus diesen beiden Freiräumen

der passendere gewählt.

$$
\begin{aligned}
d_{\text{xy}} &:= \Gamma(p_{\max,\widehat{O}}, p_{\min,\widehat{O}}) \\
f_{\text{best,no Flip}} &:= \Big\{ f \in L_{\text{FR}} \Big| \forall f' \in L_{\text{FR}},\ f \neq f' : \\
&\qquad S\left(d_{\text{xy}} - \Gamma(p_{\max,f}, p_{\min,f}\right) \prec S\left(d_{\text{xy}} - \Gamma(p_{\max,f'}, p_{\min,f'}\right) \\
&\qquad K\left(\widehat{O}, f\right) = K\left(\widehat{O}, f'\right) = 1 \Big\} \\
d_{\text{xy,Flip}} &:= \Gamma(p_{\max,F(\widehat{O}}, p_{\min,F(\widehat{O}}) \\
f_{\text{best,Flip}} &:= \Big\{ f \in L_{\text{FR}} \Big| \forall f' \in L_{\text{FR}},\ f \neq f' : \\
&\qquad S\left(d_{\text{xy,Flip}} - \Gamma(p_{\max,f}, p_{\min,f}\right) \prec S\left(d_{\text{xy,Flip}} - \Gamma(p_{\max,f'}, p_{\min,f'}\right) \\
&\qquad K(F(\widehat{O}), f) = K(F(\widehat{O}), f') = 1 \Big\} \\
f_{\text{best}} &:= \begin{cases} f_{\text{best,Flip}} & \text{wenn} \\ & \quad S\left(d_{\text{xy,Flip}} - \Gamma(p_{\max,f_{\text{best,Flip}}}, p_{\min,f_{\text{best,Flip}}}\right) \\ & \quad \prec S\left(d_{\text{xy}} - \Gamma(p_{\max,f_{\text{best,no Flip}}}, p_{\min,f_{\text{best,no Flip}}}\right), \\ f_{\text{best,no Flip}} & \text{sonst} \end{cases}
\end{aligned}
$$

Die Punkte $p_{\min,\widehat{O}}$, $p_{\max,\widehat{O}}$ sind die beiden Punkte, die AABB $\widehat{O}_{\text{B}}$ des aktuell zu packenden Objekts $\widehat{O}$ definieren. Und damit ist d_{xy} der Punkt, bei dem in der ersten Komponente die Breite und in der zweiten Komponente die Länge von $\widehat{O}_{\text{B}}$ steht. Die Punkte $p_{\max,f'}, p_{\min,f'}$ und $p_{\max,f}, p_{\min,f}$ definieren ebenso die AABB der Freiräume.

Es wird der lexikografisch sortierte Γ-Abstand zwischen der Höhe und Breite des Objekts $\widehat{O}$ und jeden Freiraum $f' \in L_{\text{FR}}$ berechnet. Es wird der Freiraum gewählt, dessen sortierter Abstand kleiner, im Sinne von dem oben definierten komponentenweise Kleiner ($\prec$), ist als die der anderen Freiräume. Dieser Freiraum wird für das rotierte und nicht rotierte Objekt berechnet ($f_{\text{best,Flip}}$ und $f_{\text{best,no Flip}}$). Aus diesen beiden wird dann der gewählt, der wiederum nach dieser Zielfunktion besser passt.

Alle drei Zielfunktionen entscheiden sich im Zweifel so, dass das Objekt $\widehat{O}$ nicht gedreht werden muss. Damit werden häufige Umorientierungen vermieden, die wiederum eine gewisse Zeit in Anspruch nehmen.

Der Unterschied zwischen den beiden Bestfit-Zielfunktionen, welche Freiräume besser und welche schlechter bewertet werden, lässt sich anhand der Abbildung 21 gut beschreiben. In dieser werden in 21(a) und 21(b) ist jeweils ein

Freiraum abgebildet, in dem das Objektmodell $\widehat{O}$ eingezeichnet ist, um die Entscheidungen der beiden Funktionen besser erklären zu können. Die Frage, die zu beantworten bleibt, ist welchen der beiden Freiräume welche der beiden Bestfit-Zielfunktionen wählt, wenn sie sich zwischen diesen beiden Freiräumen entscheiden muss.

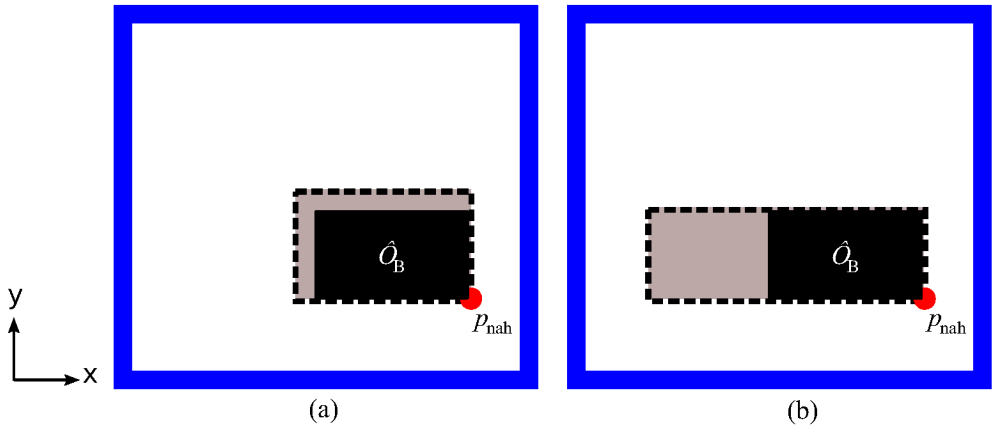


Abbildung 21: Vergleich der Wahl des Freiraums (grau) durch die Zielfunktionen Bestfit-Abstand und Bestfit-Volumen. Wenn beide Zielfunktionen zwischen den beiden in (a) und (b) entscheiden müssen, wählt Bestfit-Volumen (a) und Bestfit-Abstand (b). Die AABB des Objekts $\widehat{O}_B$ ist in beiden Freiräumen zum Vergleich eingezeichnet.

Die Funktion Bestfit-Abstand wird den Freiraum aus Abbildung 21(b) wählen, da die Differenz der Höhe zwischen dem Freiraum und dem Objekt in y-Richtung null ist und in 21(a) die Differenz in beiden Koordinatenrichtungen größer null ist. Die Funktion Bestfit-Volumen wählt hingegen Freiraum aus Abbildung 21(a), da die Volumendifferenz zwischen Freiraum und Objekt deutlich geringer als in Abbildung 21(b) ist.

Berechnung der Ablagekonfiguration

Aus der neuen Lage und Orientierung des Objekts muss nun die Ablagekonfiguration berechnet werden. Hierbei wird davon ausgegangen, dass das Objekt so wie in Kapitel 2 gegriffen wurde. Neben der Angabe der Ablagekonfiguration wird an dieser Stelle auch die chronologische Abfolge der Bewegungen des nsa-Koordinatensystems beschrieben, wie diese Konfiguration erreicht werden kann. Zuerst einmal muss der Roboter eine Korrekturbewegung durchführen, so dass die reale Orientierung mit der Orientierung

des Objekts zum Start der Planungsphase übereinstimmt. Wenn, wie in Abschnitt 3.3.1 beschrieben, eine Rotation um den Winkel α durchgeführt wurde, wird der Greifer zusammen mit dem gegriffenen Objekt um die a-Achse des nsa-Koordinatensystems gedreht. Für den Fall, dass um $-\beta$ gedreht wurde, wird entsprechend um die a-Achse mit dem Winkel $-\beta$ gedreht. Wenn während der Planungsphase das Objekt um 90 Grad gedreht wurde, muss der Roboter diese Drehung auch durchführen. Diese beiden Drehungen werden in Kombination ausgeführt. Falls die a-Achse nicht parallel zur Normalen der Ablagefläche ist, muss das nsa-Koordinatensystem so gedreht werden, dass dies der Fall ist. Das Objekt ist nun richtig orientiert, jetzt muss nur noch die Translation auf die geplante Position erfolgen. Diese wird in drei Teile unterteilt: Zuerst wird der Greifer in Richtung der Achse bewegt, die senkrecht zur Ablagefläche ist, bis sich der Greifer über der Ablagefläche befindet. Die Höhe über dieser Fläche ist frei wählbar und kann nicht berechnet werden, da nur zweidimensionale Informationen über die Objekte vorliegen. Sie sollte so gewählt sein, dass der Roboter den Greifer mit dem Objekt kollisionsfrei über die Ablagefläche bewegen kann. Dann wird parallel zu der Ablagefläche verfahren, bis der Greifer über der geplanten Ablageposition zum stehen kommt. Die letzte Bewegung erfolgt nun normal zu der Ablagefläche direkt bis zur Ablageposition. Damit ist die Ablagekonfiguration erreicht. Wenn die Ablagefläche nicht achsen-parallel ausgerichtet ist, müssen alle Bewegungen des nsa-Koordinatensystems mit der Transformation, die die achsen-parallele Ablagefläche an die reale transformiert, modifiziert werden.

3.3.3 Roboterpackbarkeit

Es ist nicht ohne Weiteres möglich, das Objekt direkt neben die anderen Objekte zu legen, da die Greiferbacken mit anderen Objekten kollidieren würden.

Diese Problematik wurde hier folgendermaßen gelöst: Die achsen-parallele Boundingbox $\widehat{O}_{\mathrm{B}}$ des Objektmodells $\widehat{O}$ wurde auf jeder Seite um die halbe Stärke der Greiferbacken vergrößert. Damit ist zwischen zwei Objekten immer ein Zwischenraum der Breite, die der Stärke der Greiferbacken entspricht. Mit dieser Anpassung von $\widehat{O}_{\mathrm{B}}$ ist ein kollisionsfreies Ablegen und wieder Aufgreifen problemlos möglich. In Abbildung 22 ist die Situation vor der Ablage dargestellt. Das Objekt ist gegriffen und auf der Ablagefläche liegen schon andere Objekte.

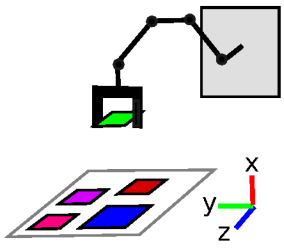


Abbildung 22: Schematische Darstellung der Situation kurz vor der Ablage des Objekts. Auf der Ablagefläche sind schon einige Objekte platziert und das aktuell zu platzierende ist vom Roboter gegriffen worden.

3.4 Experimente

Der Algorithmus wurde in C++ implementiert und die Testläufe wurden auf einem Intel®Core™2 Quad Q9450 Prozessor mit 2,66 GHz und 4 GB RAM ausgeführt. Experimente, die die Roboterpackbarkeit betrachten, können nur in Verbindung mit dem Greifplaner aus Kapitel 2 durchgeführt werden, da von diesem die zusätzlichen Informationen über die Position des Objekts zwischen den Greiferbacken geliefert werden. Diese Tests werden im Zuge der Experimente mit dem Prototypen in Kapitel 4 gemacht.

3.4.1 Testprobleme

Die Testläufe wurden mit den Benchmarkproblemen von Bischoff und Ratcliff [Bischoff 95b, Bischoff 95a] durchgeführt. Es wurden die Problemklassen 1-15 verwendet, die jeweils aus 100 verschiedenen Einzeltests bestehen. Über die Problemklassen wird die Anzahl der unterschiedlichen Boxgrößen variiert. In der ersten Problemklasse werden drei verschiedene Boxgrößen bis hin zu 100 in der 15. Problemklasse verwendet. Die Testdatensätze sind mit dem Programm in Anhang A erzeugt worden. Nachdem der Ablageplaner nur im zweidimensionalen Raum arbeitet, wurde die Höhe der Objekte ignoriert. Innerhalb einer Problemklasse werden die Größe und die Anzahl der Objekte pro Objektgröße variiert.

Jedes der hundert Probleme von jeder der 15 Problemklassen ist hundertmal mit dem Algorithmus ausgeführt worden. Damit sind pro Problemklasse 100000 Tests absolviert worden. Bei jedem dieser Tests sind die Objekte neu

gemischt worden, damit die Reihenfolge der Objekte möglichst nicht die Testergebnisse verfälscht. Diese 100000 Tests werden im folgenden als Testlauf bezeichnet.

Als Vergleichswert wird die durchschnittliche Platzausnutzung pro Problemklasse in % verwendet und zusätzlich die durchschnittliche Laufzeit betrachtet.

3.4.2 Vergleich der Zielfunktionen

Für den Vergleich der Zielfunktionen sind die Testläufe viermal wiederholt worden. In den Grafiken aus Abbildung 23 ist die durchschnittliche Platzausnutzung über die Problemklassen für alle drei Zielfunktionen (Firstfit, Bestfit-Abstand und Bestfit-Volumen) gemittelt über alle vier Testläufe aufgetragen.

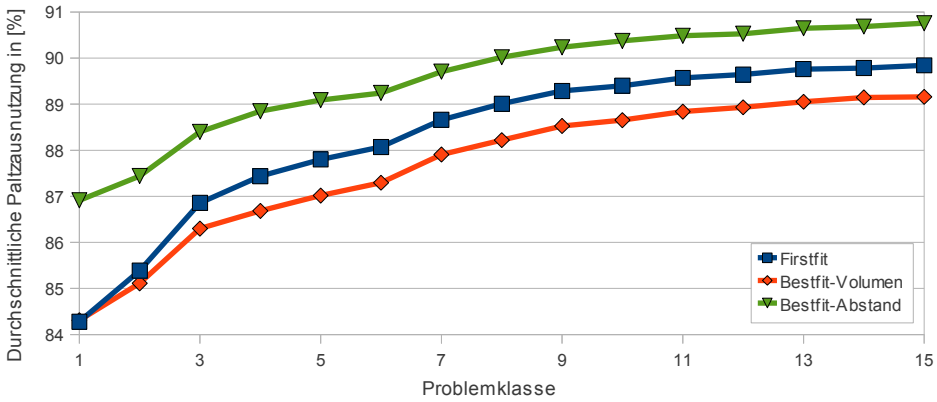


Abbildung 23: Durchschnittliche Platzausnutzung in % für die drei Zielfunktionen pro Problemklasse.

Alle drei Zielfunktionen zeigen das gleiche Verhalten. Für die Problemklassen mit wenigen unterschiedlichen Objekten sind die Ergebnisse der durchschnittlichen Platzausnutzung schlechter, als für die Problemklassen mit vielen unterschiedlichen Objekten. Die Zielfunktion Bestfit-Abstand erzielt in allen Problemklassen die besten Ergebnisse, gerade auch in den unteren Problemklassen. Bestfit-Volumen erzielt die schlechtesten Ergebnisse.

Firstfit und Bestfit-Volumen sind in den ersten zwei Problemklassen etwa gleich gut. Ab der 4. Problemlassen ist die Platzausnutzung von Firstfit um durchschnittlich 0,73 %-Punkte besser. Der Abstand zwischen den Ergebnissen von Bestfit-Abstand zu Firstfit sind in der 1. Problemklasse 2,6 %-

Punkte. Dieser wird aber kleiner und ab der 9. Problemklasse bleibt er im Mittel auf 0,92 %-Punkten.

Eine Aussage über einen Zusammenhang zwischen der Größe der Objekte und dem erreichten Belegungsgrad lässt sich nicht tätigen.

In Abbilung 24 sind die Ergebnisse des 9. Tests aus der 3. Problemklasse der Benchmarks zu sehen. Bei allen drei Ausführungen sind die Objekte in der gleichen Reihenfolge dem Algorithmus zugeführt worden. Sie sind in der Reihenfolge nummeriert, in der sie auf die Ablagefläche gelegt wurden. Ihre farbliche Markierung ist zufällig und dient nur der besseren Unterscheidbarkeit der Objekte. Anhand dieses Tests wird erklärt, wie sich die Zielfunktionen auf die Packung auswirken.

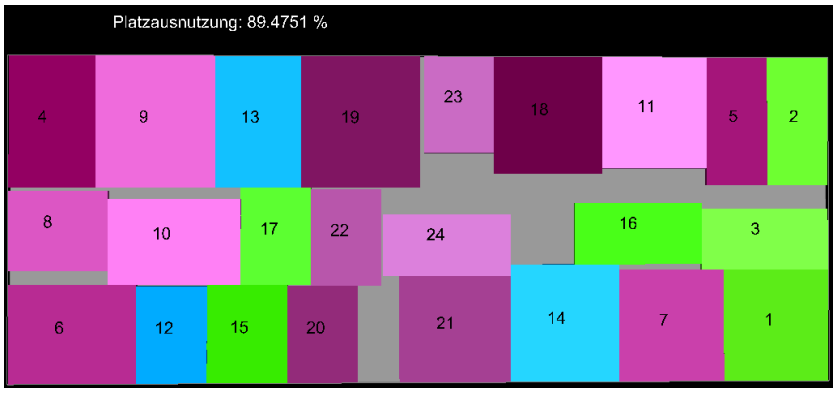
Es fällt auf, dass die Anordnung der Objekte bei Bestfit-Abstand subjektiv viel aufgeräumter wirkt, als bei den anderen beiden Zielfunktionen. Die Ausrichtung der Objekte durch die Bestfit-Abstand-Zielfunktion (Abbildung 24(a)), vor allen Dingen bei den ersten Objekten, stellt sich so dar, dass die Objekte mit den Seiten aneinander gelegt werden, die etwa gleichlang sind. Die ersten Objekte werden fast immer auf dieselbe Art und Weise platziert: Das 1. unten rechts in die Ecke; das 2. oben rechts in die Ecke, wenn es nicht zu groß ist; das 3. zwischen das 1. und 2. Objekt, wenn es hineinpasst; Für das 4. Objekt wird der Freiraum gewählt, der durch das 2. Objekt von rechts und das 3. Objekt von unten begrenzt wird. Nachdem p_{nah} dieses Freiraums mit der oberen linken Containerecke zusammen fällt, wird das Objekt an diese Ecke gelegt. Durch die Wahl der Freiräume entstehen regelrechte Reihen von Objekten, die so aneinander gelegt sind, dass deren ähnlich lange Seiten aneinander liegen (Objekte 4, 9, 13, 19 und 6, 12, 15, 20 und 1, 7, 14, 21 und 2, 5, 11, 18, 23). Sie werden immer in Richtung der nächsten Containerecke in den Freiraum gelegt, deshalb entstehen die Lücken nahe der Symmetrieachse des Containers, zwischen den Objekten 19 und 23 ,und 20 und 21. Die Lücke auf der anderen Symmetrieachse ist so groß, dass noch weitere Objekte in diese gelegt werden. Bei der Platzierung von Objekt 8 fällt auf, dass dieses weder direkt an Objekt 4 noch an 6 liegt. Diese Platzierung lässt sich einfach durch die starke Bevorzugung von Freiräumen erklären, bei denen die Differenz der Länge des Objekts zur Länge des Freiraums in einer Dimension sehr klein ist. In diesem Fall ist es der Freiraum gewesen, der von oben durch das Objekt 4 begrenzt wird, vom Objekt 7 von unten, vom Objekt 3 von rechts und links durch den Container selbst. Die linke untere Ecke des Freiraums ist am nächsten an einer Containerecke dran, deshalb liegt das Objekt 8 frei im Raum. Die Lücken, die zwischen Objekt 8 und 4 sowie 8 und 6 entstehen, sind äußerst ungünstig, da sie zu dünn sind, als dass darin noch ein weiteres

Objekt Platz finden könnte. Damit sind nach der Platzierung von Objekt 8 die ersten nicht mehr nutzbaren Freiräume entstanden.

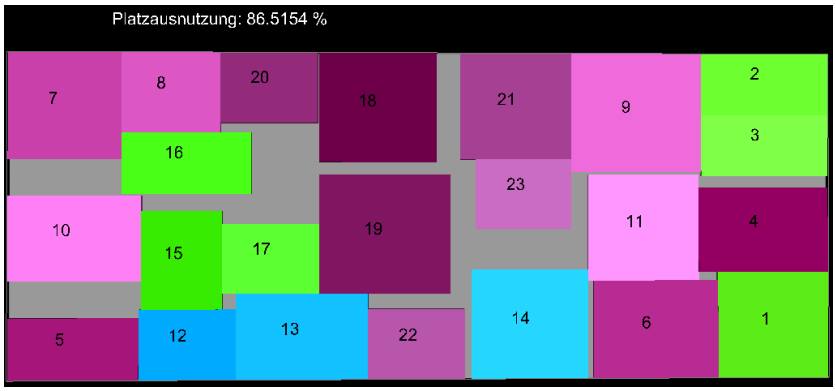
In Abbildung 24(b) ist ebenfalls das Ergebnis des 9. Tests aus der 3. Problemklasse zu sehen, allerdings mit der anderen der beiden Bestfit-Zielfunktionen. Der subjektive Eindruck der Ordnung ist nicht so ausgeprägt wie der, bei Belegung durch die Bestfit-Abstand-Zielfunktion. Diese Zielfunktion Bestfit-Volumen wählt den kleinsten Freiraum in den das Objekt passt. Die ersten Objekte werden ähnlich wie bei Bestfit-Abstand angeordnet. Da noch keine kleinen und dünnen Freiräume entstanden sind, unterscheiden sich die beiden Strategien nicht stark. Die volumenbasierte Funktion dreht die Objekte nicht so häufig im Gegensatz zur abstandbasierten. Deshalb können die ersten vier Objekte an die rechte Seite des Containers gelegt werden. Bis zum 9. Objekt werden die Objekte fast reihum in Richtung der Containerecken gelegt. Das 10. Objekte wird aus den gleichen Gründen wie bei Bestfit-Abstand das 8. Objekt zwischen dem 5. und 7. Objekt platziert. Je mehr Objekte gepackt werden, desto mehr kleinere Freiräume entstehen auf den Symmetrieachsen der Ablagefläche. Diese Freiräume werden dann natürlich auf Grund ihrer Größe von der Zielfunktion Bestfit-Volumen bevorzugt. Es entstehen zunehmend kleine Lücken zwischen schon platzierten Objekten, die nicht mehr groß genug sind, um ein Objekt dort hinein zu legen. Beispiele sind dafür: In der Mitte von 12, 13, 15 und 17, zwischen 16 und 15 und rund um das 19. Objekt, das den kompletten inneren freien Bereich in zwei Teile spaltet, nachdem es auf der Ablagefläche platziert wurde.

Die Firstfit-Zielfunktion (Abbildung 24(c)) platziert die Objekte reihum in den Freiraum, der am nächsten an einer Containerecke liegt. Die Objekte 1-17 werden nach diesem Schema einsortiert. Diese Reihenfolge ist durch die Sortierung der Liste der Freiräume bedingt. Die Objekte 18 und 19 zerteilen den zusammenhängenden Freiraum so, dass keine weiteren Objekte dieser Größe platziert werden können. Die Art und Weise, wie die Objekte reihum gelegt werden, ist ab jetzt nicht mehr leicht zu erkennen, wird aber weiterhin so betrieben. Dadurch entstehen Lücken wie beispielsweise zwischen den Objekten 8 und 20.

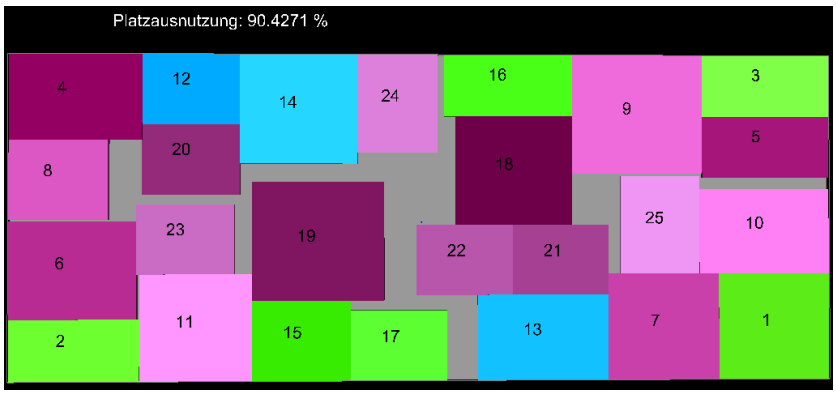
In diesem beispielhaft herausgegriffenen Testlauf ist sogar die einfachste Zielfunktion Firstfit die erfolgreichste, dicht gefolgt von Bestfit-Abstand. Selbst bei den Durchschnittswerten ist Firstfit besser als Bestfit-Volumen, da der Fall, dass ein Objekt in die Mitte der Ablagefläche gelegt wird, nur weil da viele kleinere Freiräume entstanden sind, bei denen besonders kleine Volumendifferenzen auftreten, nicht auftritt. Die Sortierung der Liste der Freiräume sorgt dafür, dass der erste passende Freiraum auch immer der ist, der am



(a) Bestfit-Abstand



(b) Bestfit-Volumen



(c) Firstfit

Abbildung 24: Ergebnisse des Algorithmus mit den verschiedenen Zielfunktionen des Benschmarktests: 3. Problemklasse 9. Einzeltest. Die Objekte sind in der Reihenfolge nummeriert, in der sie platziert worden sind.

nächsten an einer Ecke der Ablagefläche liegt. Die Platzierung und Reihenfolge der Objekte erinnert an eine Spirale, die sich entlang der Außenkanten bis ins Innere der Ablagefläche läuft.

In der Grafik aus Abbildung 25 ist die durchschnittliche Laufzeit aller drei Zielfunktionen über alle Problemklassen aufgetragen. Die Laufzeit des Algorithmus mit den unterschiedlichen Zielfunktionen, um die Platzierung der Objekte auf der Ablagefläche zu planen bis diese voll ist, ist erwartungsgemäß. Die beiden Bestfit-Varianten haben eine deutlich längere Laufzeit als Firstfit mit durchschnittlich 1,02 ms.

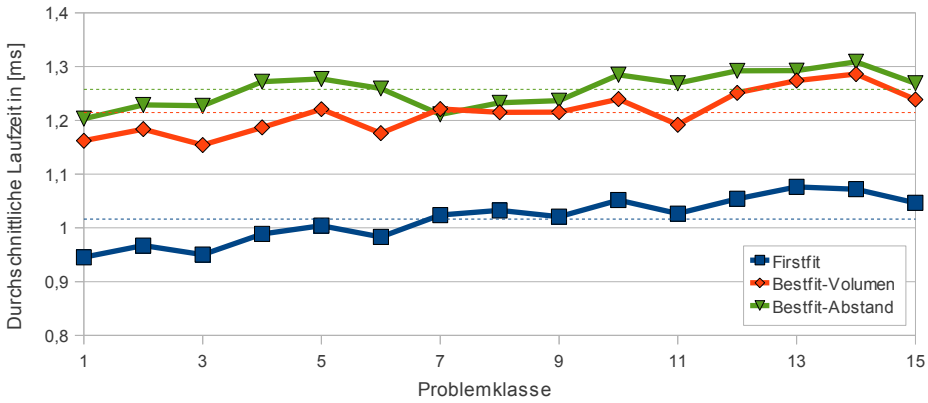


Abbildung 25: Durchschnittliche Laufzeit in ms für die drei Zielfunktionen pro Problemklasse. Der Mittelwert über alle Problemklassen ist jeweils gestrichelt eingezeichnet.

Im Durchschnitt benötigt Bestfit-Volumen 1,21 ms und Bestfit-Abstand 1,26 ms. Der Zeitunterschied von 0,05 ms ist durch den höheren Berechnungsaufwand für die Bewertung eines Freiraums bei Bestfit-Abstand zu erklären.

Alles in allem gesehen, ist die Laufzeit so kurz, dass diese für die Wahl der besten Zielfunktion nicht für relevant erachtet wird. In anderen Domänen wie der Speicherverwaltung [Tanenbaum 01] spielt diese sehr wohl eine Rolle und somit wäre Firstfit wohl die beste Wahl. In unserem Fall wählen wir Bestfit-Abstand als Zielfunktion, da sie die beste durchschnittliche Platzausnutzung erzielt.

In Abbildung 26 ist eines der besten Resultate der Ablageplanung aus der 3. Problemklasse zu sehen (Test 34).

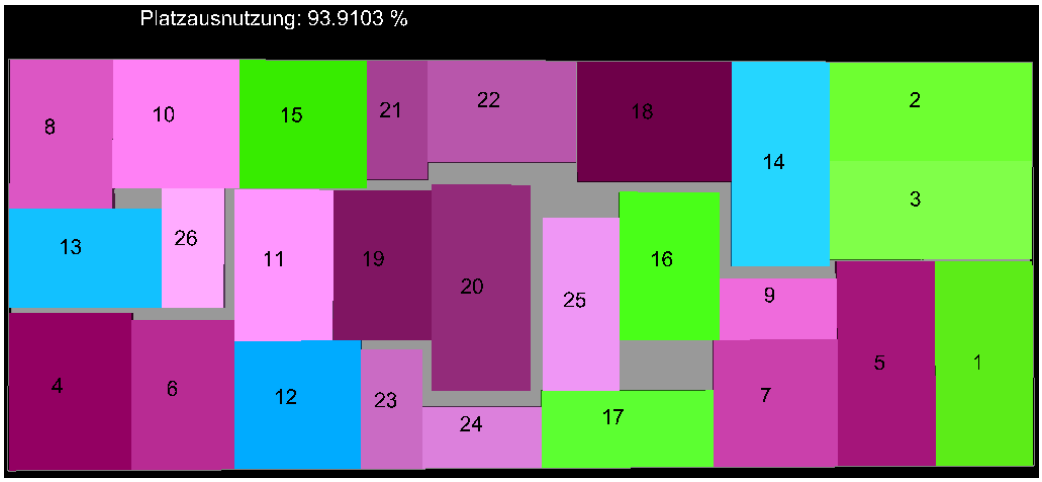


Abbildung 26: Ein sehr gutes Ergebnis erzielt mit der Zielfunktion Bestfit-Abstand. (Problemklasse 3 Test 34)

3.4.3 Vergleich der Heuristik zur Platzierung der Objekte

Aus dem letzten Abschnitt ist hervorgegangen, dass die Zielfunktion Bestfit-Abstand die beste durchschnittliche Platzausnutzung unter den drei Zielfunktionen erzielt. Diese legt allerdings nur den Freiraum und die Orientierung des Objekts fest. An welcher Stelle innerhalb des Freiraums die Objekte gelegt werden, ist bisher so geregelt gewesen, dass das Objekt an die Ecke des Freiraums gelegt wurde, die am nächsten an einer Ecke liegt. Diese Heuristik wird in diesem Abschnitt näher betrachtet, indem sie mit einer anderen sehr nahe liegenden verglichen wird: Die Heuristik, die sagt, dass ein Objekt immer in Richtung einer ganz bestimmten Ecke der Ablagefläche gelegt werden muss. Diese Strategie ähnelt sehr der Einschränkung aus [Martello 07]. Dort dürfen die Objekte immer nur ausgehend von einer Ecke an ein bestehendes Objekt angelegt werden. Wenn man in der unteren rechten Ecke das erste Objekt platziert, dann darf das nächste nur links davon oder direkt darüber angelegt werden.

Die Heuristik, die den Abstand zu allen vier Containerecken berechnet wird im Folgenden mit 4CH abgekürzt und die andere mit 1CH (für eine Ecke).

In Abbildung 27 wird die durchschnittliche Platzausnutzung des Algorithmus mit beiden Varianten der Heuristiken mit jeweils der gleichen Zielfunktion (Bestfit-Abstand) abgebildet. Bestfit-Abstand hat mit beiden Heuristiken jeweils das beste Ergebnis geliefert. In Abbildung 28 ist das Ergebnis des 9. Test

aus der 3. Problemklasse der Benchmarks mit beiden Heuristiken zu sehen. 1CH ist in allen Problemklassen konstant um etwa 2 %-Punkte schlechter als

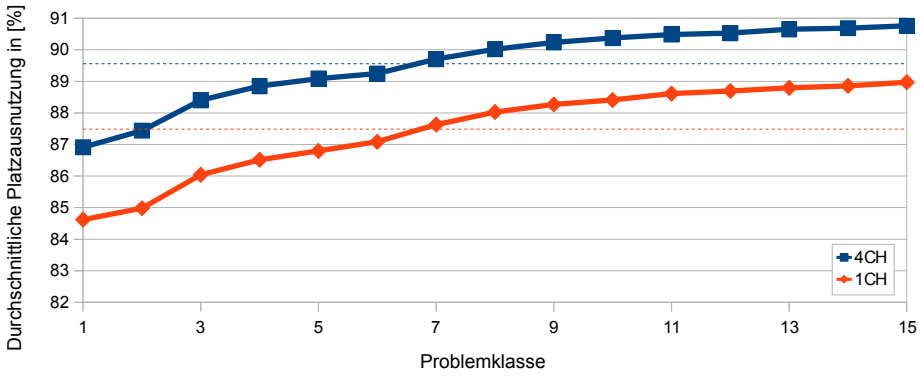


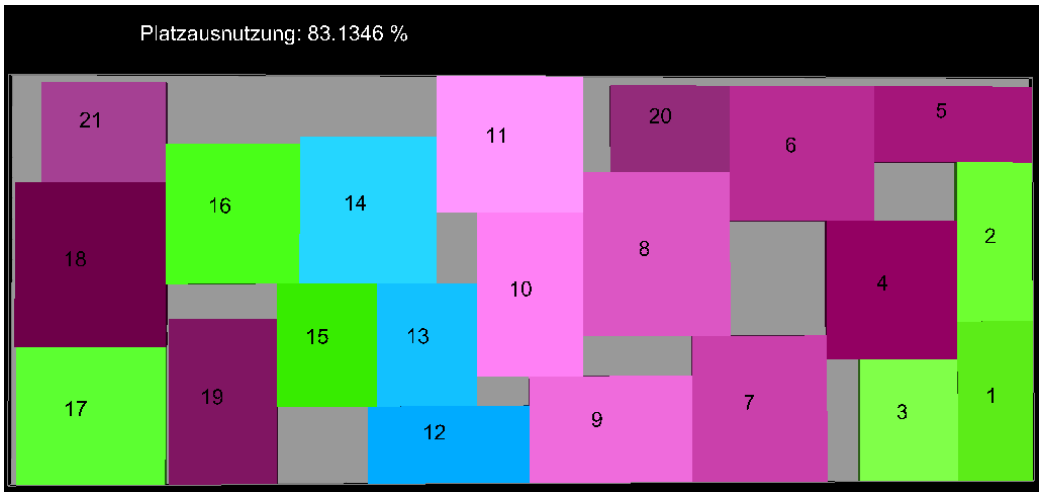
Abbildung 27: Vergleich der Heuristiken 1CH und 4CH. Mittelwert über alle Problemklassen ist jeweils gestrichelt eingezeichnet.

4CH. Die Lücken, die bei 1CH entstehen, liegen vornehmlich an Kanten der Ablagefläche, die nicht an der Ecke liegen, zu der der Abstand berechnet wurde. Der subjektive Eindruck, dass die Belegung mit 1CH (Abbildung 28(a)) unordentlicher aussieht, ist der erste Indikator dafür, dass der Platz der Ablagefläche gut ausgenutzt worden ist, die kleinen Zwischenräume, die von Anfang an entstehen können, später nicht mit einem anderen Objekt gefüllt werden. Schon nach dem 6. Objekt entsteht in diesem Beispiel eine Lücke, die nicht mehr gefüllt wird (zwischen Objekt 2, 4, 5 und 6). Mit 4CH entstehen diese ungewollten Lücken erst bei der Platzierung des 8. Objekts und danach bleiben diese Lücken die Ausnahme. Mit 1CH entstehen die Lücken bei fast jedem Objekt, das nach dem 6. Objekt auf die Ablagefläche gepackt wird.

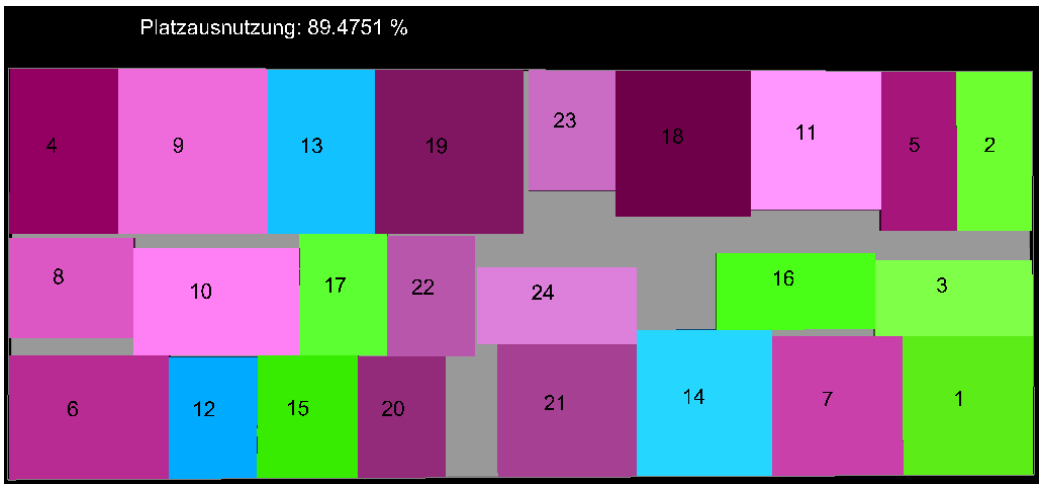
Mit diesem Vergleich wird klar, dass 4CH die bessere Heuristik ist.

3.5 Schlussfolgerung

Es ist ein Ablageplaner entwickelt worden, der für rechteckige Objekte eine Ablagekonfiguration auf einer gegebenen Ablagefläche bestimmt. Dabei wird berücksichtigt, dass das Objekt durch den Roboter abgelegt werden kann und von diesem jeder Zeit wieder aufgegriffen werden kann, indem zwischen den Objekten ein so großer Abstand gelassen wird, dass der Greifer gerade noch dazwischen passt. Der Planer verwaltet ein Liste von Freiräumen auf der Ablagefläche. Diese Freiräume überlappen sich. Aus der Liste der Freiräume



(a) Objekte werden immer in Richtung der unteren rechten Ecke in dem Freiraum platziert



(b) Objekte werden immer in Richtung der nächsten Ecke der Ablagefläche in dem Freiraum gelegt.

Abbildung 28: Vergleich der zwei Heuristiken anhand des 9 Tests aus der 3. Problemklasse der Benchmarks

wird mittels einer Zielfunktion der Freiraum bestimmt, in den das Objekt gelegt wird und eine Heuristik bestimmt, an welche Ecke des Freiraums das Objekt platziert wird. Aus der geplanten Platzierung des Objekts wird die Ablagekonfiguration berechnet.

Insgesamt sind drei verschiedene Zielfunktionen für die Auswahl des Freiraums und zwei Heuristiken zur Platzierung des Objekts innerhalb des gewählten Freiraums untereinander verglichen worden. Als Vergleichsmaß wurde die durchschnittliche Platzausnutzung in % und die durchschnittliche Laufzeit benutzt. Bei den Zielfunktionen hat Bestfit-Abstand, mit einer Laufzeit unter 1,3 ms, die besten Ergebnisse bei der Platzausnutzung erzielt. Bei den beiden Heuristiken hat diejenige bessere Ergebnisse geliefert, die das Objekt immer in Richtung der nächsten Ecke der Ablagefläche innerhalb des Freiraums platziert.

Die Laufzeit, die in den Experimenten gemessen wurde, bezieht sich auf die Zeit, die benötigt wird, Objekte auf die Ablagefläche zu legen, bis diese voll ist. Dafür werden maximal 1,3 ms benötigt. Die Zeit, die für ein Objekt benötigt wird, liegt somit weit unter einer Millisekunde. Die Platzausnutzung auf der Ablagefläche liegt für stark heterogene Probleme bei über 90 %, wenn man den Freiraum für die Greiferbacken nicht mitrechnet. Durch die Berücksichtigung der Stärke der Greiferbacken liegt die Platzausnutzung deutlich darunter.

Die Anforderung, dass der Ablageplaner schnell arbeiten soll, ist mit einer Planungszeit von unter einer Millisekunde den Anforderungen entsprechend. Für ein Objekt wird nur ein Bruchteil einer Millisekunde benötigt, dieser Zeitraum ist so kurz, dass er während der Roboterbewegung nicht auffällt. Die Anforderung, dass die Packung möglichst dicht und roboterpackbar sein soll, ist erfüllt worden. Und die Möglichkeit, dass der Roboter das abgelegte Objekt wieder greifen kann, ist auch umgesetzt worden.

An der realen Platzausnutzung auf der Ablagefläche gibt es Verbesserungsbedarf. Die Roboterpackbarkeit ist im Moment durch die Annahme gewährleistet, dass nur zweidimensional auf einer Fläche gepackt wird und die Boundingboxen der Objekte so weit vergrößert worden sind, dass die Greiferbacken während der Ablage und dem Aufgreifen nicht mit benachbarten Objekten kollidieren.

Um eine bessere reale Platzausnutzung zu ermöglichen, müssen Strategien entwickelt werden, die entweder die Platzierung so berechnen, dass der Roboter diese ohne Kollision mit der Umwelt oder sich selber erreichen kann, oder es müssen Strategien entwickelt werden, bei denen der Roboter das Objekt, falls nötig, an seine Zielposition schiebt. Dafür ist es praktikabel, die

nötigen Bewegungsabfolgen zum Positionieren des Objekts in die Bewertung der Freiräume durch die Zielfunktion einzubeziehen.

Des Weiteren muss man sich Gedanken machen, wie ein erneutes Auspacken durch einen Industrieroboter bei einer sehr dichten Packung aussieht, ohne dass man auf spezielle Greifer, wie in der Industrie üblich, zurückgreift.

Vielleicht ist es möglich, Zielfunktionen aus Packstrategien vom Menschen abzuleiten, indem man beobachtet, wie Menschen Objekte in einen Korb packen. Nachdem der Industrieroboterarm einem menschlichen Arm nahe kommt, lassen sich sicherlich einige gute heuristische Ansätze daraus ableiten.

Eine weitere wichtige Erweiterung des Algorithmus wäre, dass mit nicht regulären Objekten geplant wird, damit die packbaren Objekte nicht nur auf Boxen beschränkt sind. Erlaubt man nun auch noch, dass der Ablagebereich eine beliebige Form hat, ist man auf dem Weg, die Ablageplanung ausgehend von dem industriellen CLP auf eine Vielzahl von anderen Problemfeldern zu erweitern.

Eine wirklich wichtige Erweiterung wäre, dass der Planer auch auf den dreidimensionalen Raum erweitert wird. Das ist für den reinen Planungsvorgang kein Problem. Es müssen nur Mechanismen entwickelt werden, die auf weitere Randbedingungen achten, wie zum Beispiel, dass ein Objekt an der Stelle, an der es abgelegt werden soll, auch liegen kann und nicht in der Luft läge.

Kapitel 4

Prototyp

Dieses Kapitel beschreibt den Aufbau des Prototypen. Dieser realisiert die Aufgabenstellung aus Abschnitt 1.1. Die Gliederung des Kapitels sieht folgendermaßen aus: Zuerst wird in Abschnitt 4.1 der Aufbau des Prototypen vorgestellt und in Abschnitt 4.2 sind die Experimente und deren Ergebnisse beschrieben, die mit der Beispielanwendung durchgeführt worden sind. Am Ende wird eine Schlussfolgerung gezogen (Abschnitt 4.3)

4.1 Aufbau

Das Herzstück des Hardwareaufbaus ist der Industrieroboter KUKA Leichtbauroboter 4 [KUKA Roboter GmbH 08]. An dem Roboter ist ein Backengreifer von der Firma Schunk montiert. Direkt an dem Greifer ist eine Logitech Pro 9000 Webcam fest montiert.

Die entwickelte Software wird auf deinem handelsüblichen Computer mit einem Intel®Core™2 Quad Q9450 Prozessor mit 2,66 GHz und 4 GB RAM ausgeführt auf dem Linux SuSe 11.3 als Betriebssystem läuft. Die Kommunikation zwischen Robotersteuerung und Computer erfolgt über eine Ethernet-Verbindung mit Hilfe des KUKA Fast Research Interfaces (FRI) [Schreiber 10], das speziell für diesen Roboter entwickelt wurde.

Es werden in Abschnitt 4.1.1 zuerst die technischen Daten des Roboters, die zur Verfügung stehenden Sensorwerte und das FRI vorgestellt und danach die technischen Daten der Kamera und deren Kalibrierung. Im Abschnitt 4.1.3 wird schließlich der logische Ablauf der Anwendung vorgestellt.

4.1.1 Hardware

KUKA Leichtbauroboter

Dieser Roboter ist entwickelt worden, um die Einschränkungen eines klassischen sechs-achsigen Industrieroboters bei Handhabungsaufgaben weiter zu reduzieren. Er besitzt sieben Achsen (Abbildung 29)und damit genauso viele wie der menschliche Arm. Durch seine spezielle Bauweise, ist er im Stande 14 kg zu heben bei einem Eigengewicht von 15 kg. Jedes Gelenk ist mit einem Momentensensor, Drehwinkelsensor und einem Motorstellungssensor ausgestattet. Das Handgelenk ist mit einem Kraft-Momenten-Sensor mit sechs Freiheitsgraden ausgestattet.

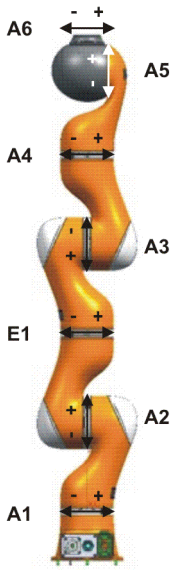


Abbildung 29: Nummerierung der Gelenke des KUKA Leichtbauroboter 4 [KUKA Roboter GmbH 08].

Der Roboter kann mit Hilfe der KUKA Robot Langugage (KRL) programmiert werden oder über eine eigens für diesen Roboter entwickelte Schnittstelle über eine Netzwerkverbindung gesteuert werden. Das Fast Research Interface (FRI) ist eine rudimentäre Schnittstelle, die über einen Socket die Netzwerkverbindung zwischen PC und Robotersteuerung herstellt. Auf PC-Seite steht eine in C++ implementierte Bibliothek zur Verfügung. Sie stellt die Datenstruktur bereit, die alle Sensorwerte des Roboters enthält. Mit dem Roboter wird über das UDP-Protokoll kommuniziert. Durch ein Zeitstempelverfahren wird überwacht, ob die Verbindungsqualität ausreichend gut

ist. Die Verbindungsqualität wird in vier Stufen („nicht akzeptabel“, „ausreichend“, „gut“ und „perfekt“) bewertet. Nur bei einer Verbindungsqualität von „gut“ oder „perfekt“ erlaubt das FRI auf Robotersteuerungsseite, dass der Roboter Befehle von der PC-Seite ausführt. Falls ein Paket verloren geht, wird die Verbindungsqualität eine Stufe nach unten korrigiert und erst wieder nach oben gesetzt, wenn genügend Pakete korrekt gesendet und empfangen wurden. Falls die Verbindungsqualität nur „ausreichend“ oder gar „nicht akzeptabel“ ist wird eine eventuell gerade ausgeführte Bewegung sofort unterbrochen, es werden aber immer die aktuellen Sensordaten des Roboters von der Robotersteuerung geschickt.

Es muss sichergestellt werden, dass ein Paketaustausch in fest vorgegebenen Zeitintervallen stattfindet. Falls dies nicht der Fall ist, wird die Verbindungsqualität auf „nicht akzeptabel“ gesetzt. Wenn der Roboter bewegt werden soll, muss pro Zeitschritt eine absolute Konfiguration über das FRI geschickt werden. Je nach Regelstrategie (Positions-, Kartesische-Impedanz- und Gelenk-Impedanz-Regelung) müssen in dieser Konfiguration unter anderem die absoluten Gelenkwinkel oder die absolute Position und Orientierung des nsa-Koordinatensystems angegeben sein. Das FRI bietet keine Möglichkeit, komplexere Befehle wie eine Punkt-zu-Punkt Bewegung an den Roboter zu übermitteln, sondern nur, eine absolute Konfiguration zu übermitteln, die der Rotor nach dem aktuellen Zeitintervall haben soll.

Logitech Webcam Pro 9000

Die Logitech Webcam Pro 9000 [Logitech 11] ist mit einem Carl Zeiss^{TM} Objektiv mit Autofokus ausgestattet. Die native Auflösung ihrer Bilder ist 1600×1200 Pixel, die mit einer maximalen Bildrate von 30 fps aufgenommen werden. Die Bilder werden im RGB-Farbraum aufgenommen und über USB 2.0 an den Computer übertragen.

Zur Ansteuerung der Kamera unter Linux stellt Logitech keine speziellen Treiber zur Verfügung. Die Kamera erfüllt jedoch die *USB Device Class Definition for Video Devices* der USB-IF, Inc.[USB-IF Inc. 11]. Diese Spezifikation regelt die minimalen Anforderungen, die an ein USB-Gerät gestellt werden, die Video-Daten über USB übertragen. Das heißt, dass die Kamera durch einen generischen UVC-Treiber angesprochen werden kann. Mit diesem Treiber liefert die Kamera eine Bildauflösung von 640×480 Bildern (VGA-Auflösung) bei einer Bildwiederholrate von etwa 24-30 fps. Einstellungen wie Fokus, Weißabgleich oder Belichtungszeit können bei dieser Kamera über den UVC-Treiber nicht vorgenommen werden, stattdessen werden diese Parame-

ter automatisch geregelt.

Die Kamera ist mit dem Verfahren aus [Heikkila 97] kalibriert worden. Dabei sind sowohl die intrinsischen als auch extrinsischen Parameter bestimmte worden. Diese ermöglichen es aus einer Pixelkoordinate einen Sichtstrahl ausgehend von dem Kameramittelpunkt in den Raum zu berechnen.

4.1.2 Software

Die gesamte Software ist in C++ implementiert. Die Entscheidung für diese Sprache ist durch die hardwarenahen Implementierungsaufgaben und durch die vorhandenen Softwarekomponenten begründet.

Die Ausgangssituation ist durch die gegebene Hardware festgelegt. Für die Ansteuerung vom Roboter und der Kamera sind eigens Softwarekomponenten entwickelt worden.

Die Softwarekomponente, die auf das FRI aufsetzt, um eine komfortable Interaktion mit dem Roboter umzusetzen, bietet die Möglichkeit, Punkt-zu-Punkt Fahrbefehle, sowohl im Gelenkwinkelraum, als auch im Kartesischenraum, abzusetzen. Im Gelenkwinkelraum werden zeitoptimale Bewegungen nach Schinn [Schwinn 99] und im Kartesischenraum lineare Bewegungen implementiert. Bei der Bewegung im Kartesischenraum wird die Position und Orientierung des nsa-Koordinatensystems angegeben. Um die Gelenkwinkel zu berechnen, die dann von der Robotersteuerung eingestellt werden, muss eine Rückwärtstransformation durchgeführt werden. Diese wird von der Robotersteuerung vorgenommen und ist von der aktuellen Robotergeschwindigkeit abhängig. Nachdem der Roboter sieben Achsen hat, ist es nicht möglich, die Rücktransformation analytisch zu berechnen.

Des weiteren bereitet sie die Sensordaten des Roboters so auf, dass sie direkt weiter verwendet werden können. Ihre wohl wichtigste Aufgabe ist jedoch, dass der Datenaustausch innerhalb des fest vorgegebenen Zeitintervalls stattfindet.

Für die Kamera existieren zwei voneinander unabhängige Komponenten: Zum einen ist eine logische Kamera implementiert worden, die für Pixelkoordinaten die Richtung des Sichtstrahls berechnet. Dazu müssen Position und Orientierung der extrinsischen Parameter der Kamera mit jeder Roboterbewegung neu gesetzt werden, da die Kamera physisch an dem Roboter befestigt ist. Diese feste Verbindung ist durch das Observer-Entwurfsmuster [Gamma 95] realisiert worden. Nach jedem Datenaustausch zwischen Roboter und PC werden die Kameraparameter neu berechnet.

Die Kamerabilder werden mit Hilfe openCV-Bibliothek [Bradski 00] über den UVC-Treiber kontinuierlich aufgenommen und für die weitere Benutzung zur Verfügung gestellt.

Die Softwarekomponente für die Ansteuerung des Greifers ist in einer früheren Arbeit [Schörner 10] implementiert worden. Damit sind alle Schnittstellen zur Hardware so implementiert, dass sie für andere Systemkomponenten über eine klar definierte und übersichtliche Schnittstelle benutzbar sind. In Anhang B ist eine CD enthalten, auf der sich der gesamte Quellcode befindet.

4.1.3 Anwendung

Die beiden entwickelten Planer sind zusammen mit der Hardware zu einem Gesamtsystem vereinigt worden. Es sind drei große Komponenten, die zusammen arbeiten müssen. Das Greifen, Ablegen und die Bewegung zwischen der Ablage- und der Aufgreiffläche. Die Anwendung ermöglicht es, unbekannte Objekte aufzugreifen und geordnet abzulegen. Die abgelegten Objekte können wieder aufgegriffen und zurückgegeben werden. Die Objekte werden von einem Menschen auf die Aufgreiffläche gelegt. Sobald die Ablagefläche voll ist greift der Roboter die abgelegten Objekte wieder auf und übergibt sie dem Benutzer.

In Abbildung 30 ist der logische Ablauf der Anwendung in einem Flussdiagramm dargestellt. Es sind folgende Bedingungen gegeben: Natürlich wird die oben beschriebene Hardware verwendet. Des Weiteren ist die Ablagefläche und die Höhe über der Ablagefläche, die von dem Greifer eingenommen werden muss, wenn er sich über diese Fläche bewegt, gegeben. Die Aufgreiffläche ist nicht exakt festgelegt, es wird nur die Ebene angegeben auf der sich Objekte befinden können. Von dem gesamten Bereich, der von der Kamera gesehen wird, können Objekte gegriffen werden. Der Roboter wird mit der Kartesischen-Impedanz-Regelung betrieben. Alle Bewegungen werden entweder durch Angabe der Zielorientierung und -position des nsa-Koordinatensystems oder durch Angabe von Entfernungen und relativen Winkeln, in die sich der Roboter bewegen soll, definiert.

Die *Startkonfiguration* des Roboters ist in Gelenkwinkeln (0,296726; -0,538593; -1,44948; -1,83679; 1,12469; 1,08737; -0.217824)[rad], die Ablagefläche liegt zwischen den Punkten (0,370 m; 0,032 m) und (0,670 m; 0,220 m) in der yz-Ebene und die Höhe über der Ablagefläche, die vom Roboter über der Ablagekonfiguration eingenommen wird, ist absolut bei $x = 0,064$ m. Der Greifer schließt die Backen mit einer konstanten Kraft von 14 N.

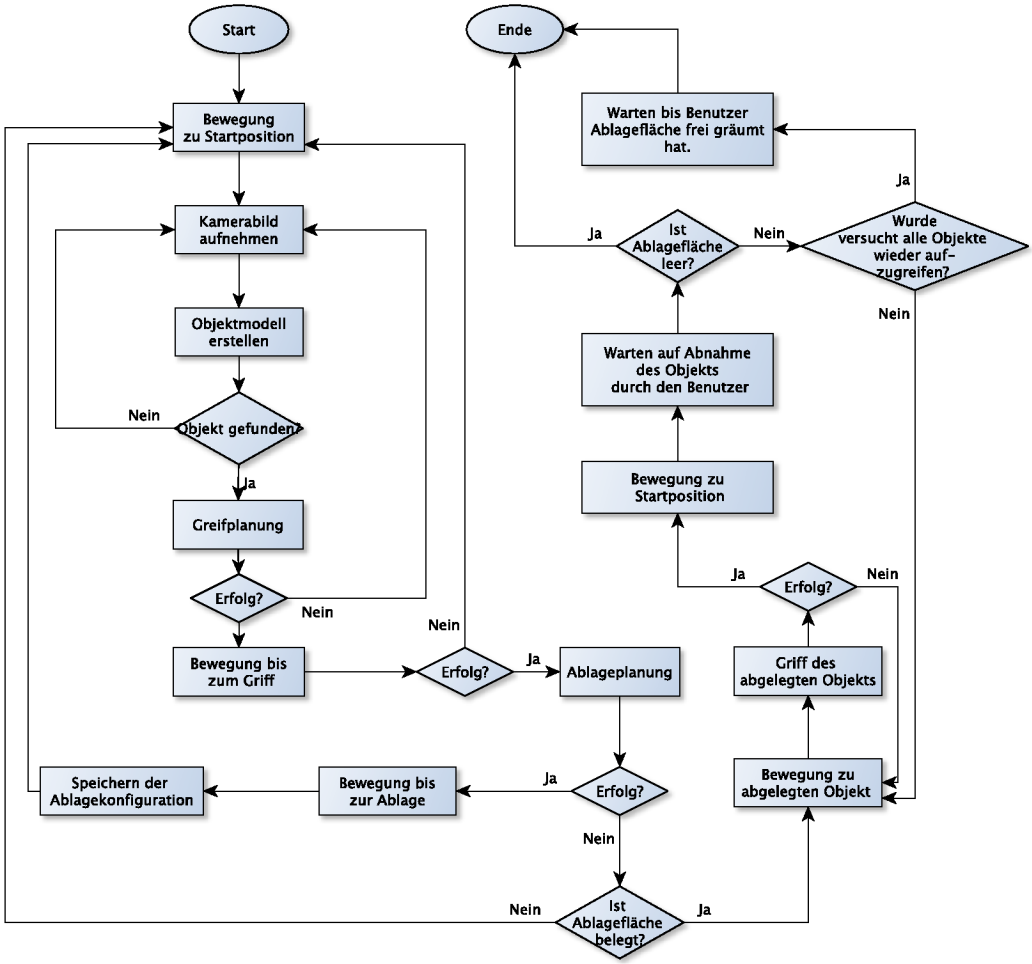


Abbildung 30: Flussdiagramm des Ablaufs eines Durchlaufs der Anwendung

Zum Start der Anwendung fährt der Roboter in die fest definierte Startkonfiguration (Abbildung 31). Von dieser Position aus wird ein Bild der Aufgreiffläche aufgenommen. In diesem Bild werden die bildbasieren Objektmodelle (Abschnitt 2.3.1) der Objekte auf der Ablagefläche bestimmt. Falls keine Objektmodelle berechnet werden konnten, wird erneut ein Kamerabild aufgenommen. Aus der Menge der Objektmodelle wird eines herausgenommen und mit diesem die Greifplanung (Abschnitt 2.3.2) durchgeführt. Falls für das aktuelle Objektmodell keine Greifkonfiguration gefunden wurde und weitere Objektmodelle aus dem Kamerabild erstellt werden konnten, wird mit dem nächsten Modell erneut geplant. Wenn für keines der zur Verfügung stehenden Objektmodelle die Greifplanung erfolgreich ist, wird erneut ein Kamerabild aufgenommen.

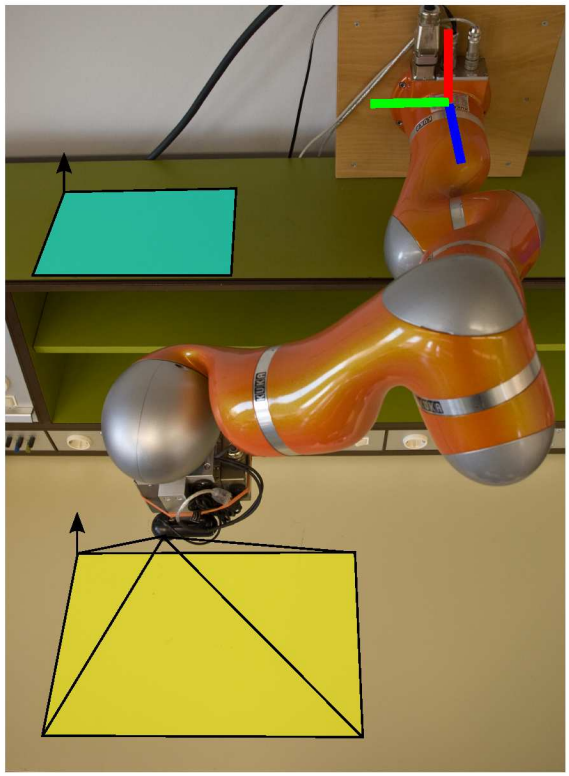


Abbildung 31: Bild des Aufbaus der Hardware mit dem Roboter in Startkonfiguration, der eingezeichneten Aufgreif- (gelb) und Ablagefläche (hellblau) und dem Weltkoordinatensystem. Für beide Flächen ist die Richtung der Normalen eingezeichnet (Pfeil)

Wenn eine Greifplanung erfolgreich abgeschlossen werden konnte, wird die Greifkonfiguration vom Roboter angefahren. Zuerst dreht sich der Greifer um die a-Achse, bis die Greiferbacken so stehen, dass sie parallel zu den beiden ausgewählten Greifregionen sind (Bewegung 1). In der 2. Bewegung wird der Greifer translatorisch über das Objekt und schon in Richtung der Aufgreiffläche bewegt. Die Bewegung 3 ist ausschließlich in Richtung der Aufgreiffläche. Diese Bewegung ist sehr langsam und endet sobald eine Kraft von 15 N entgegen der Bewegungsrichtung auftritt. Das bedeutet, dass entweder die Aufgreiffläche erreicht ist oder der Greifer mit dem Objekt kollidiert, wenn das Objekt höher ist als die Länge der Greiferbacken. Nun schließt der Greifer. Sind die Greiferbacken ganz geschlossen, ist der Griff nicht erfolgreich gewesen und der Roboter bewegt sich zurück in die Startkonfiguration.

Wenn der Griff erfolgreich gewesen ist, wird das bildbasierte Objektmodell für die Ablageplanung angepasst und diese auch durchgeführt(Abschnitt 3.3).

Wenn eine Ablagekonfiguration bestimmt werden konnte, hebt der Roboter das gegriffene Objekt von der Ablagefläche langsam an (Bewegung 4). Der Greifer wird so um die a-Achse gedreht, dass das Objekt für die Ablage richtig ausgerichtet ist (Bewegung 5). Dann bewegt der Roboter den Greifer auf die Höhe über der Ablagefläche, die zuvor festgelegt wurde (Bewegung 6). Dann erfolgt eine translatorische Bewegung genau über die Ablageposition (Bewegung 7). Von dort aus wird wieder in einer langsamen Bewegung das Objekt auf die Ablagefläche gelegt. Die Bewegung 8 stoppt auch, wenn eine Kraft von 15 N entgegen der Bewegungsrichtung auftritt. Damit ist das Objekt auf der Ablagefläche platziert. Der Greifer öffnet sich um 2 mm und entfernt sich in Bewegung 9 in Normalenrichtung der Ablagefläche von dem Objekt. Jetzt verfährt der Roboter in Bewegung 10 und 11 zurück in die Startkonfiguration.

Falls die Ablageplanung nicht erfolgreich gewesen ist, weil kein Platz mehr auf der Ablagefläche ist, löst der Greifer den Griff und bewegt sich zurück in die Startkonfiguration. Von dort aus werden alle Objekte, die schon auf die Ablagefläche gelegt worden sind, wieder aufgegriffen. Der Roboter bewegt den Greifer in Bewegung 12 in die Position über das erste abgelegte Objekt. Diese Position ist während der Ablage gespeichert worden. Der Greifer wird soweit geöffnet, wie er auch geöffnet war, als dieses Objekt abgelegt wurde. Die Bewegung 13 ist wieder in Richtung der Aufgreiffläche, bis die Kraft von 15 N entgegen der Bewegungsrichtung auftritt. Der Greifer wird geschlossen und es wird wieder anhand der Greiferstellung überprüft, ob der erneute Griff erfolgreich gewesen ist. Mit Bewegung 14 wird das Objekt normal zur Ablagefläche angehoben. Mit den Bewegungen 15 und 16 verfährt der Roboter zurück in Startkonfiguration. Dort wartet der Roboter so lange, bis das gegriffene Objekt von einem Menschen gegriffen wird und dem Roboter aus dem Greifer genommen wird. Dabei werden die Kräfte am Greifer überwacht. Sobald eine Kraft mit in Richtung der a-Achse des Greifers mit einem Betrag größer als 15 N auftritt, öffnet der Greifer. Es wird versucht, alle Objekte wieder von der Ablagefläche aufzugreifen. Falls mindestens ein Objekt nicht erfolgreich wieder aufgegriffen werden konnte, wartet der Roboter in der Startkonfiguration bis der Benutzer bestätigt, dass der Ablagebereich manuell leer geräumt wurde. Damit endet ein Durchlauf der Anwendung. Die Anwendung kann sofort wieder neu gestartet werden, indem der Ablageplaner neu initialisiert wird.

Der Griff des Objekts wird während der Bewegung zur Ablageposition nicht überwacht. Wenn das Objekt nicht kraftschlüssig gegriffen wurde, kann es seine Position im Greifer verändern oder sogar wieder aus dem Greifer herausfallen. In diesem Fall führt der Roboter die Bewegung weiter aus. Für

den Ablageplaner ist der Bereich, in den das Objekt gelegt werden sollte, trotzdem als belegt markiert.

4.2 Experimente

In diesem Abschnitt werden die Experimente für die entwickelte Beispielanwendung präsentiert. In Abschnitt 4.2.1 sind die einzelnen Versuche beschrieben und in Abschnitt 4.2.2 werden die Testkriterien vorgestellt, mit denen die Versuche bewertet werden. Die Auswertung der Experimente steht in Abschnitt 4.2.3.

4.2.1 Versuchsaufbau

Das System, dessen Ablauf im Abschnitt 4.1.3 beschrieben ist, wird mit verschiedenen Objekten getestet. Es werden im Wesentlichen drei verschiedene Tests durchgeführt. Einmal sollen die Objekte auf der Aufgreiffläche nur gegriffen werden, um das Greifen einzeln bewerten zu können. Mit einem einfachen Quader, der an den Rändern der Aufgreiffläche positioniert wurde, soll die Genauigkeit der Kamerakalibrierung im Zusammenspiel mit der Positioniergenauigkeit des Roboters bewertet werden. Der dritte Test wird mit dem gesamten Ablauf aus Abschnitt 4.1.3 durchgeführt. In diesem Test werden die Grenzen des Greifplaners, Ablageplaners und der Objektmodellierung im Zusammenhang mit der Ausführung durch die Hardware betrachtet. Bei diesem Test ist es erlaubt, dass mehrere Objekte auf der Aufgreiffläche liegen.

4.2.2 Testkriterien

Beim Greifen wird bewertet, wie und wo das Objekt gegriffen worden ist und ob der Griff der Planung entspricht. Beim Ablegen wird die reale Position bewertet, das Objekt kollisionsfrei abgelegt wurde und ob es wieder aufgegriffen werden konnte.

Bei der Ausführung der Anwendung wird zuerst die Laufzeit gemessen, um die Schnelligkeit der Anwendung zu beurteilen. Weiter werden die Momente an den Robotergelenken gemessen. Diese sollen Aufschluss darüber geben, in wieweit die Bewegungsgeschwindigkeit des Roboters ausgereizt ist.

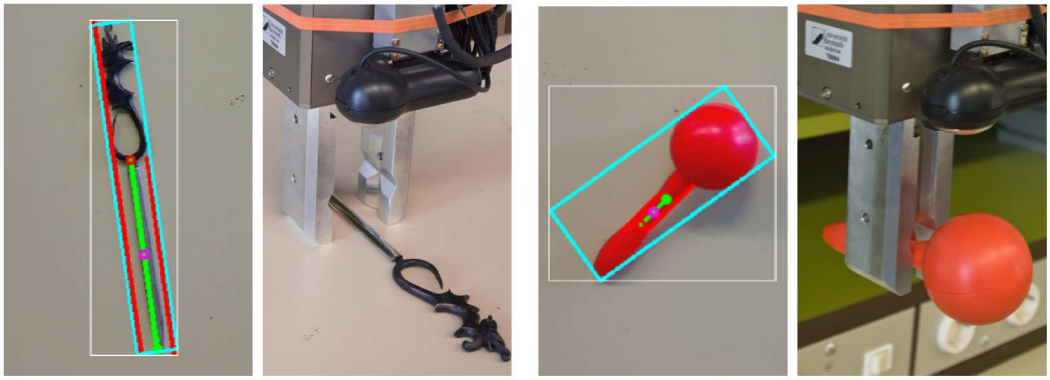
4.2.3 Auswertung

Auswertung der Griffe der Objekte

Es sind diverse Objekte getestet worden, die teilweise schon in Kapitel 2 für die Experimente genutzt wurden. In Abbildung 32 wird eine Auswahl dargestellt. Es werden immer die Paare Planungergebnis im Kamerabild und ein Bild vom realen Griff gezeigt. Für jedes Objekt wurde die Greifplanung und die anschließende Bewegung bis zum Griff so lange durchgeführt, bis ein Griff erfolgreich gewesen ist. Dazu ist gemäß des Ablaufs der Anwendung (Abschnitt 4.1.3) immer ein neues Kamerabild aufgenommen, die Objektmodellierung und, falls eine Greifkonfiguration gefunden ist, auch der Griff durchgeführt worden.

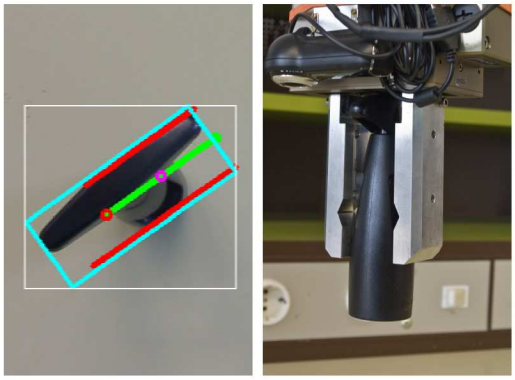
Bei der Betrachtung der Ergebnisse fällt auf, dass sobald erfolgreich geplant wurde, auch der Griff erfolgreich gewesen ist. Bei dem Brieföffner, der Kinderrassel und dem Korkenzieher wurden sehr viele Planungsdurchläufe benötigt, bis eine Greifkonfiguration gefunden wurde. Nachdem der Brieföffner sehr dünn und schmal ist, mussten drei Versuche unternommen werden, bis schlussendlich der Griff erfolgreich war. Beim ersten Mal hat der Greifer zu das Objekt überhaupt nicht zu fassen bekommen und ein anderes Mal ist das Objekt sofort weggerutscht als sich der Greifer geschlossen hat. Der Brieföffner ist durch seine geringe Höhe und seine glatte Oberfläche schwer zu greifen.

Der Griff wird unter der Annahme geplant, dass der Kontakt zwischen den Greiferbacken und dem unbekannten Objekt kraftschlüssig ist. Nachdem man nicht garantieren kann, ob ein Griff dieses Kriterium erfüllt, muss eine Abschätzung bezüglich des Objektgewichts vorgenommen werden, damit gerade noch ein Kraftschluss vorliegt und es nicht durch sein Eigengewicht wieder aus dem Greifer rutscht.

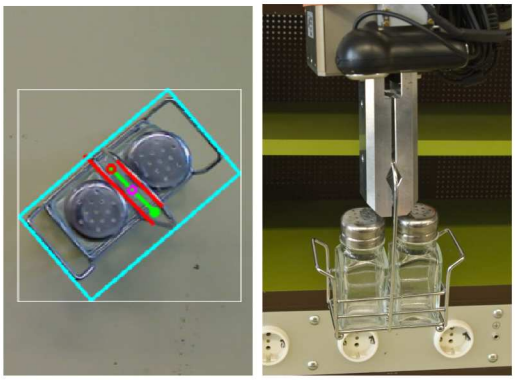


(a) Brieföffner (10/2/1)

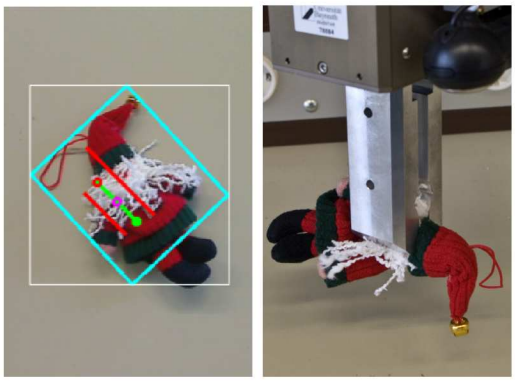
(b) Kinderrassel (7/1/1)



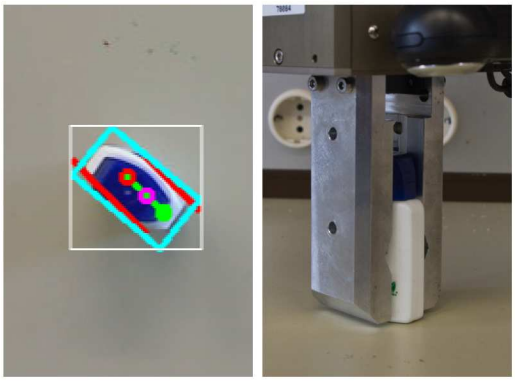
(c) Korkenzieher (7/1/1)



(d) Salz- und Pfefferstreuer (2/1/1)



(e) Weihnachtsmann(4/1/1)



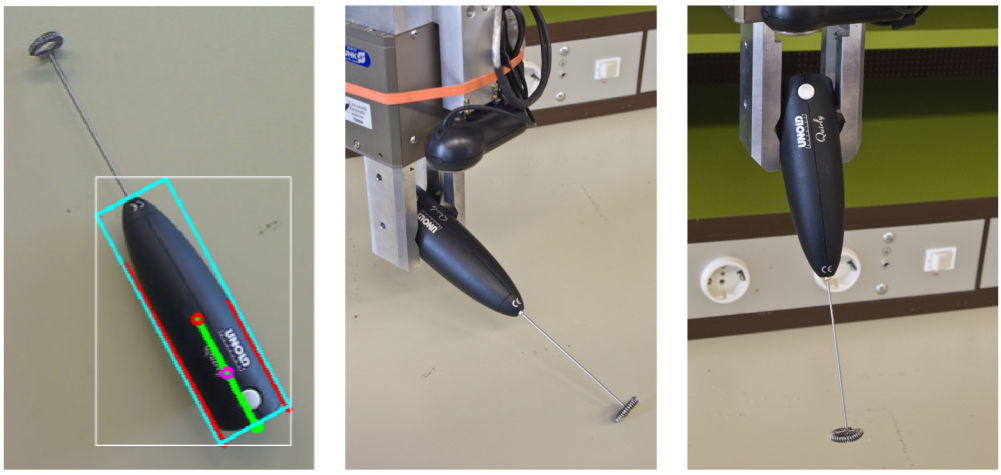
(f) Niveaplasche stehend(2/1/1)

Abbildung 32: Bildpaare Kamerabild mit den eingezeichneten Greifregionen und Bild vom Griff für jedes Objekt. In Klammern jeweils (Durchschnittliche Anzahl Planungen bis zum Griff / Anzahl Griffe / Erfolgreiche Griffe)

Abschätzung des Objektgewichts für dessen kraftschlüssigen Griff Die Greiferbacken des hier verwendeten Greifers sind aus Metall. Wenn man für das Material des Objekts annimmt, dass es Kunststoff ist, ergibt sich für $\mu_{\mathrm{H,min}} = 0,25$ [BBS Winsen 07]. Dieser Wert entspricht einer äußerst geringen Reibung und eignet sich daher zur konservativen Abschätzung. Es wird die Formel aus Abschnitt 2.3.2 verwendet, um das maximale Objektgewicht abzuschätzen. Die Greifkraft des Greifers ist 14 N. Damit wird für alle Objekte leichter als $m_{\mathrm{max}} = 1,01\,\mathrm{kg}$ ein Kraftschluss hergestellt.

Im Folgenden wird anhand zweier Objekte gezeigt, wie der Kraftschluss nicht hergestellt werden kann, obwohl die Objekte leichter sind, als das gerade ausgerechnete Gewicht. Das schwerste Objekt, das in allen Tests von dem Greifer gegriffen wurde, wiegt 350 g und damit weit unter dem maximalen Gewicht, bei dem noch ein Kraftschluss hergestellt werden kann.

In Abbildung 33 ist das Planungsergebnis und der Griff eines Milchschäumers zu sehen. Die Greifplanung veranlasst den Greifer das Objekt weit ab von dessen Schwerpunkt zu greifen, da nur dort parallele Kanten vorhanden sind. Durch die Entfernung zum Massenschwerpunkt entsteht ein zu großes Moment und das Objekt dreht sich (Abbildung 33(b) und 33(c)).



(a) Kamerabild mit den eingezeichneten Greifregionen.

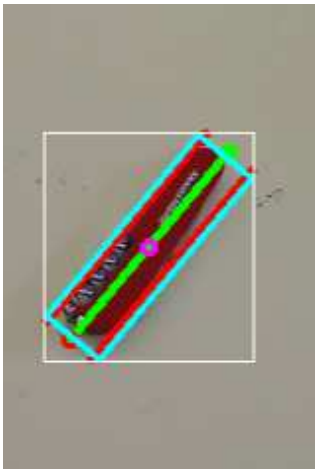
(b) Kurz nach dem Griff.

(c) Objekt dreht sich zwischen den Greiferbacken.

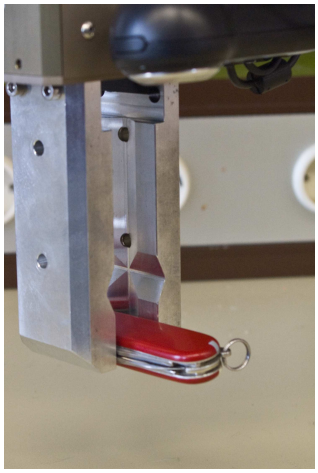
Abbildung 33: Milchaufschäumer: Planung erfolgreich. Griff nicht kraftschlüssig, Objekt dreht sich durch zu großes Moment.

In Abbildung 34(a) ist das sehr gute Planungsergebnis für ein Taschenmesser zu sehen. Dieses schnappt kurz nach dem Aufgreifen um seine Längsachse (Moment um die Längsachse des Objekts), da die Kontakte zwischen Grei-

ferbacke und Objekt auf unterschiedlichen Höhen für beide Backen sind. Dadurch entsteht das Moment um die Längsachse des Objekts und es fällt aus dem Greifer heraus.



(a) Kamerabild mit Planungsergebnis des Taschenmessers.



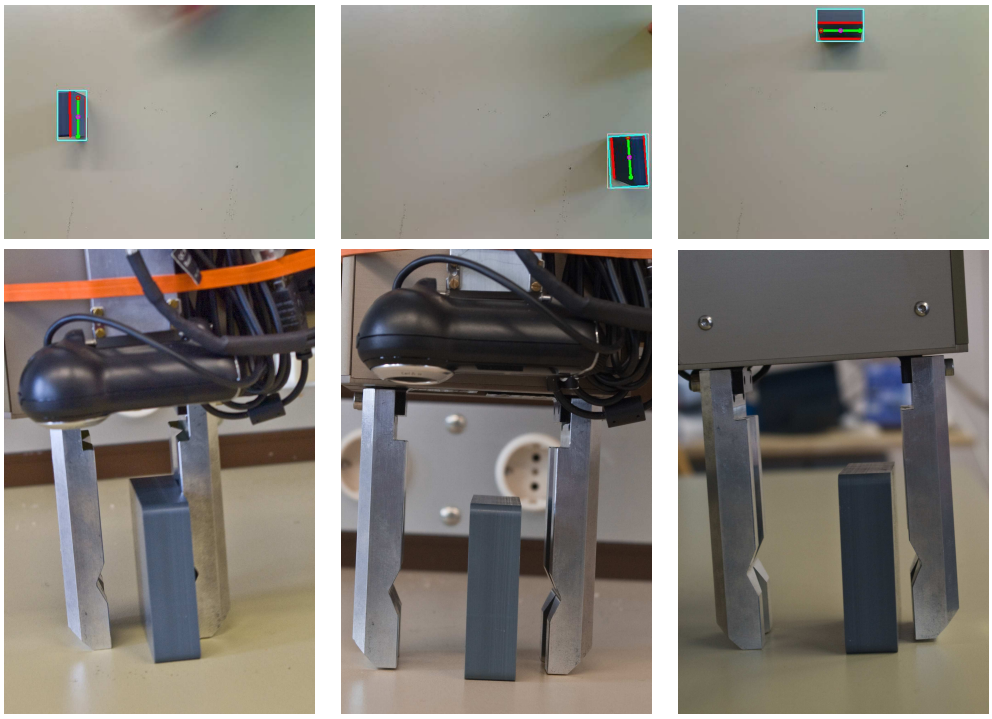
(b) Kurz nach dem Anheben kippt das Taschenmesser um die Längsachse und fällt aus dem Greifer.

Abbildung 34: Objekt kann nicht erfolgreich gegriffen werden. Der Kontakt zwischen Greifer und Objekt erzeugt ein Moment um die Längsachse des Objekts.

Bei diesen beiden Beispielen sieht man, dass obwohl der Griff erfolgreich war, während der Bewegung das Objekt zwischen den Greiferbacken verrutscht oder gar komplett aus dem Griff entkommt.

Positionierungsungenauigkeit Es ist ein rechteckiger Quader so auf die Aufgreiffläche gestellt worden, dass er gerade noch auf dem Kamerabild zu sehen ist. In Abbildung 35 sind die Kamerabilder mit den Ergebnissen der Planung zu sehen und immer passend dazu die Bilder, wie der Greifer das Objekt angefahren hat.

Die Greifplanung hat stets ein sehr gutes Ergebnis geliefert und damit die optimalen Voraussetzungen für einen präzisen Griff geschaffen. Der Greifer greift in allen Fällen das Objekt erfolgreich. Er ist aber gerade in den Randbereichen nicht mittig über dem Objekt platziert, wenn der Griff erfolgt. Optimal wäre, wenn der Abstand zwischen Greiferbacken und Klotz, kurz



(a) linker Bildrand (b) rechter Bildrand (c) oberer Bildrand

Abbildung 35: Jeweils Bild der Kamera mit den eingezeichneten Greifregionen und der gemeinsamen Strecke (grün); und ein Bild kurz vor dem Griff um die eingenomme Greifkonfiguration zu beurteilen.

bevor die Greiferbacken sich schließen gleich groß wäre. Der gegriffene Klotz hat eine Dicke von 2 cm. Damit sieht man auf den Bildern, dass am linken Bildrand (Abbildung 35(a)) und am rechten Bildrand (Abbildung 35(b)) die Abweichung von der Mitte zwischen den Greiferbacken etwa 1 cm ist. Am oberen Bildrand (Abbildung 35(c)) ist die Abweichung etwa 2 cm. In der Mitte des Bildes ist dieser Fehler nicht vorhanden.

Dieser Positionierfehler des Greifers entsteht durch die Kamerakalibierrung und durch die Positionierungsungenauigkeit des Roboters. Wenn sich einzelne Achsen des Roboter schnell bewegen mussten dann ist dieser Positionierungsfehler sehr hoch (genaue Werte wurden nicht gemessen). Beispielsweise vollzieht der Roboter einen Konfigurationswechsel, sobald er Objekte greifen will, die am linken Bildrand liegen. In diesem Fall schnappt das 2. Gelenk ruckartig über die 90 Grad Grenze. Ab diesem Zeitpunkt ist die Ausrichtung des nsa-Koordinatensystems nicht mehr wie eigentlich vorgegeben. Dieser Fehler wird vom Roboter auch nicht mehr korrigiert, obwohl die Korrektur-

werte gesendet werden.

Die Positionierungsungenauigkeit in Richtung der n-Achse des nsa- Koordinatensystems wird durch den Planer ausgeglichen, indem in der Mitte der gemeinsamen Strecke (grüne Strecke in den Kamerabildern) vom Greifer genutzt wird, um sich darüber zu positionieren. Die Ungenauigkeit in Richtung der s-Achse wird nur durch die maximal geöffneten Greiferbacken ausgeglichen. Wenn das Objekt selbst fast so breit ist, wie der Abstand zwischen den Greiferbacken, dann ist ein Griff unwahrscheinlicher je weiter das Objekt am Rand des Kamerabildes abgebildet wird.

Auswertung des Ein- und Auspackvorgangs

In Abbildung 36 ist anhand von vier Bildern der komplette Prozess, angefangen vom Griff des ersten Objekts bis hin zum Wiederaufgreifen des ursprünglich zuerst abgelegten Objekts, dargestellt. In Abbildung 36(b) ist gut zu erkennen, dass der Freiraum, der zwischen den Objekten gelassen wird, gerade ausreicht, dass die Greiferbacken dazwischen passen. Auf der Aufgreiffläche liegen mehrere Objekte, die einzeln nacheinander erfolgreich gegriffen und auch abgelegt werden. Dabei ist das Objektmodell in allen Fällen fehlerfrei gewesen.

(a) Das erste Objekt wird gegriffen

(b) Ablage des zweiten Objekts direkt neben das erste

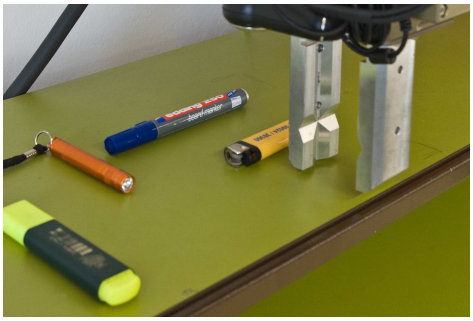
(c) Die Ablagefläche ist komplett gefüllt

(d) Das erste Objekt wird wieder aufgegriffen

Abbildung 36: Kronologischer Ablauf der Ablegens und Wiederaufgreifens anhand einiger ausgewähleter Bilder.

Wie schon aus Kapitel 3 ist das Ablegen und Aufgreifen von der Ablagefläche hier gleichbedeutend mit dem einpacken und auspacken aus einem zweidimensionalen Container. Ausschlaggebend für den fehlerfreien Ablauf ist, dass die Boundingbox des Objekts (immer blau dargestellt auf den Kamerabildern) korrekt bestimmt wurde. Wenn diese zu klein ist, können Kollisionen beim

Ablegen mit schon platzierten oder zukünftig abzulegenden Objekten auftreten. Nachdem die letzte Bewegung zur Ablageposition kraftgeregelt ist, stoppt der Roboter die Bewegung, sobald eine Gegenkraft zur Bewegungsrichtung auftritt und der Greifer lässt das Objekt los. Dadurch kann es passieren, dass Objekte auf andere gelegt werden oder schon platziere Objekte verschoben werden. Damit ist die vom Ablageplaner angenommene und gespeicherte Ordnung auf der Ablagefläche nicht mehr vorhanden und es kommt unweigerlich beim Wieder-Aufgreifen der Objekte zu Fehlern. Nachdem die Objekte nicht mehr an der Stelle liegen, an der sie abgelegt wurden, schlägt das Wieder-Aufgreifen für dieses Objekt fehl (Abbildung 37(a)). Alle Objekte, die immer noch an ihrem Platz liegen werden ohne Probleme aufgegriffen (Abbildung 36(d)). Der Milchschaumaufschäumer ist (Abbildung 37(b)) über der Fernbedienung abgelegt worden, da nur der schwarze Griff als Objekt identifiziert worden ist und der metallene Rühraufsatz bei der Objektmodellierung nicht berücksichtigt wurde. Neben der Kollision bei der Ablage des Milchschaufaufschäumers ist es auch nicht mehr möglich, die Fernbedienung wieder aufzugreifen.



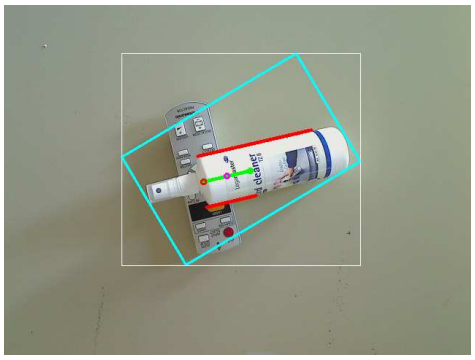
(a) Das Objekt ist beim Wiederaufgreifen nicht mehr an der angenommen Position.

(b) Objekte liegen übereinander.

Abbildung 37: Aufgertretene Fehler

In Abbildung 38(c) ist die Situation abgebildet, in der zwei Objekte übereinander auf der Ablagefläche liegen. Das Objektmodell der Sprühflasche wird nicht korrekt erstellt. Ihre minimale Boundingbox (Abbildung 38(a)) wird deutlich zu groß modelliert, aber die beiden in rot eingezeichneten Greifregionen führen zu einer erfolgreichen Greifplanung und dadurch auch zu einem erfolgreichen Griff (Abbildung 38(d)). Die Bewegung endet, nachdem die Greifer auf die Fernbedienung getroffen sind. Bei der Ablage (Abbildung 38(e)) wirkt sich jetzt der Fehler in der Objektmodellierung der Sprühflasche aus. Nachdem diese fast auf der Diagonalen ihrer Boundingbox liegt, wird sie auch schräg auf der Ablagefläche abgelegt. Für die auf der Aufgreiffläche ver-

bleibende Fernbedienung wird dann wie erwartet eine erfolgreiche Greifplanung (Abbildung 38(a)) durchgeführt, wobei auch hier ein kleiner Fehler im Objektmodell auftritt. Die Boundingbox ist etwas zu klein. Dadurch können wiederum Kollisionen beim Ablegen entstehen.



(a) Kamerabild mit der zu großen Boundingbox (hellblau) der Sprühflasche aber korrekter Greifplanung.

(b) Kamerabild mit dem korrekten Objektmodell der Fernbedienung, wobei die Boundingbox (hellblau) etwas zu klein ist, und korrekter Greifplanung.

(c) Initiale Objektanordnung auf der Aufgreiffläche.

(d) Erfolgreicher Griff der Sprühflasche.

(e) Schräge Ablage der Sprühflasche da die Boundingbox falsch modelliert ist.

Abbildung 38: Greif- und Ablagevorgang von zwei Objekten die auf der Aufgreiffläche übereinander liegen.

Wie schon angedeutet liegt die Hauptproblematik, warum die Ablage und dadurch auch das Wiederaufgreifen fehlgeschlagen ist, an fehlerhaften Ob-

jektmodellen. Oft ist die Boundingbox zu klein und dadurch sind die Objekte zu klein repräsentiert. Dadurch treten Kollisionen bei der Ablage und dem Wieder-Aufgreifen auf.

Auswertung bezüglich der Geschwindigkeit

In der Aufgabenstellung wurde bereits formuliert, das das Wort „schnell“, bezogen auf die Geschwindigkeit, differenziert betrachtet werden muss. In diesem Abschnitt wird vornehmlich die Bewegungsgeschwindigkeit des Roboters betrachtet. Die Laufzeit der einzelnen Planer ist schon in den Kapiteln 2 und 3 analysiert worden.

Der Roboter bewegt den TCP linear im Kartesischenraum mit einer konstanten Beschleunigung a und einer maximalen Geschwindigkeit v. Die Gesamtbewegung ist in einzelne Teilbewegungen aufgeteilt. Für jede Teilbewegung einer Gesamtbewegung wird in Tabelle 4.1 und 4.2 die relative Bewegung entlang der Koordinatenachsen, die Rotation um eine beliebige Achse und die Ausführungszeit angegeben. Eine Gesamtbewegung beginnt mit der Bewegung von der Startposition über der Aufgreiffläche in Richtung Objekt und endet mit dem erneuten Erreichen der Startposition, nachdem das gegriffene Objekt abgelegt wurde. Dazu wird die Bewegung vom Wieder-Aufgreifen addiert. Diese beginnt auch bei der Startposition und endet dort. Insgesamt werden 16 Teilbewegungen durchgeführt.

Die Bewegungen unterscheiden sich pro Objekt nur minimal, deshalb sind hier die Laufzeiten nur für ein Objekt abgedruckt. In der Tabelle 4.1 sind die Bewegungen mit einer Geschwindigkeit ($v = 0.25$ m/s, $a = 0,15$ m/s^2) vollzogen worden, mit der der Roboter in allen Versuchen fehlerfrei bewegt werden konnte. In Tabelle 4.2 hingegen ist der Roboter in einer Geschwindigkeit ($v = 0.28$ m/s, $a = 0,15$ m/s^2) bewegt worden, in der es zu Problemen gekommen ist. Nachdem die Rücktransformation von der Robotersteuerung selbst gerechnet wird, konnte kein Einfluss auf diese genommen werden. Sie ist von der Geschwindigkeit abhängig und achtet bei der Berechnung der Gelenkwinkel nicht auf fest eingestellte Softwareendschalter, die eine Selbstkollision verhindern. Die Rücktranformation neigt dazu, dass bei hohen Geschwindigkeiten vornehmlich die Gelenke bewegt werden, die nahe an dem TCP (Tool Center Point) sind. In Abbildung 39(a) ist der Roboter in einer Stellung, in der manuell der Notaus an der Robotersteuerung ausgelöst werden musste, da sonst eine Kollision zwischen Greifer und Roboter aufgetreten wäre. Bei niedriger Geschwindigkeit sind in der gleichen Situation die Gelenkstellungen wie in Abbildung 39(b). Dort wurden die Gelenkwinkel

	Bewegung				Laufzeit
	$\triangle\alpha$ [rad]	$\triangle x$ [m]	$\triangle y$ [m]	$\triangle z$ [m]	$\triangle t$ [sec]
Greifen	0,62	0	0	0	1,89
	0	-0,168	0,042	0,09	2,59
	0	$-\infty$	0	0	5,74
Ablegen	0	0,2	0	0	7,26
	-0,56	0	0	0	2,38
	0	0,414	0,079	-0,003	1,45
	0	0,005	-0,09	-0,523	3,76
	0	-0,148	0,003	-0,006	2,48
	0	0,12	0	0	2,2
	0	0	0	0,25	2,58
	0,11	-0,258	-0,02	0,185	2,63
Auspacken	0,09	0,288	0,0132	-0,435	5,356
	0	$-\infty$	0	0	7,48
	0	0,12	0	0	2,2
	0	0	0	0,35	4,5
	0,12	-0,26	-0,017	0,077	2,63
				Gesamtzeit:	57,13

Tabelle 4.1: Auflistung der Laufzeiten für die Teilbewegungen und die verfahrenen Distanzen. Die Geschwindigkeiten sind bei allen Bewegungen so gewählt worden, dass die Ausführung ohne Zwischenfälle möglich ist.

durch die Rücktransformation so berechnet, dass keine Selbstkollisionen am Roboter auftreten.

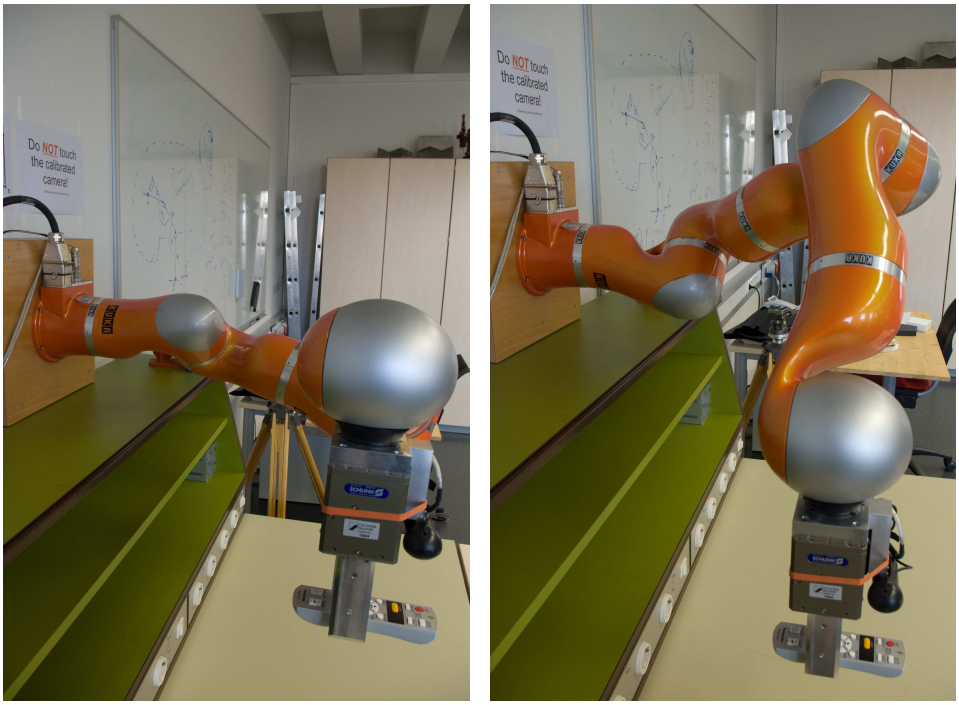
Die Gesamtausführungszeit beträgt mit den sicheren Geschwindigkeiten für jede Teilbewegung 57,13 sec und 52,85 sec für die schnelleren Bewegungen. Der zeitliche Unterschied ist mit 4,28 sec bezogen auf die Gesamtdauer relativ gering. Die Bewegungen 3, 4, 13 und 14 nehmen zusammen schon 22,66 sec bzw. 21,66 sec ein. Während der Bewegung 3 und 13 fährt der Roboter so lange in Richtung der negativen x-Achse, bis eine Gegenkraft eintritt. Nachdem nicht bekannt ist, wie hoch das zu greifende Objekt in x-Richtung ist, muss die Bewegung in einer Geschwindigkeit ($v = 0,03$ m/s) ausgeführt werden, in der das Objekt nicht beschädigt wird und auch jederzeit angehalten werden kann, ohne einen Nothalt zu erzwingen. Die Bewegung 4 ist genauso langsam, weil der Roboter die Gelenke 2 und 3 stark bewegen muss

	Bewegung				Laufzeit
	$\triangle\alpha$ [rad]	$\triangle x$ [m]	$\triangle y$ [m]	$\triangle z$ [m]	$\triangle t$ [sec]
Greifen	0,1	0	0	0	0,64
	0	-0,171	0,09	0,095	2,13
	0	$-\infty$	0	0	5,84
Ablegen	0	0,166	0	0	6,14
	-0,1	0	0	0	0,97
	0	0,446	0,037	-0,002	3,45
	0	0,005	-0,09	-0,567	3,4
	0	-0,148	0,002	-0,007	2,48
	0	0,12	0	0	2,2
	0	0	0	0,25	2,58
	0,09	-0,259	-0,019	0,222	2,63
Auspacken	0,06	0,289	0,03	-0,479	3,58
	0	$-\infty$	0	0	7,48
	0	0,12	0	0	2,2
	0	0	0	0,35	4,5
	0,12	-0,26	-0,017	0,077	2,63
				Gesamtzeit:	52,85

Tabelle 4.2: Auflistung der Laufzeiten für die Teilbewegungen und die verfahrenen Distanzen. Die Geschwindigkeiten sind so weit erhöht worden, dass die maximalen Momente und Geschwindigkeiten an einzelnen Gelenken des Roboters erreicht werden.

und die Rücktransformation Gelenkwinkel berechnet, die einen Konfigurationswechsel bedeuten. Bei höheren Geschwindigkeiten vom TCP treten Momente über 38 Nm im 2. Gelenk bei genau diesem Konfigurationswechsel auf. Bei deren Überschreitung löst die Robotersteuerung den Notaus aus. Bei der 4. Bewegung aus Tabelle 4.2 ist die Distanz verkleinert worden, die langsam in Richtung der x-Achse zurückgelegt wird, um Zeit zu sparen. Diese Verkürzung hat in der folgenden Bewegung dazu geführt, dass der Roboter nicht mehr den Konfigurationswechsel zurück auf die Startkonfiguration vollzogen hat und dann, bei den nächsten Bewegungen, eine Kollision zwischen Greifer und Roboter unvermeidlich ist.

Die Bewegung 14 entfernt den Greifer vom Objekt, bei einer höheren Geschwindigkeit ist die Bewegung des TCP durch den Roboter nicht mehr linear in Richtung der x-Achse, sondern beschreibt eine leichte Kurve. Das



(a) Die geschwindigkeitsabhängige Rückwärtstransformation hat eine Gelenkwinkelstellung berechnet, die zu einer Kollision zwischen Greifer und Roboter führt, wenn der Roboter sich weiter bewegen würde.

(b) Die Gelenkstellung ist von der Rückwärtstranformation so berechnet worden, so dass keine Selbstkollisionen auftreten.

Abbildung 39: Beispielergebnis für die geschwindigkeitsabhängige Rücktransformation. In (a) ist der Roboter mit einer höheren Geschwindigkeit als in (b) bewegt worden. Die Bewegung des TCP war in beiden Fällen identisch.

führt dazu, dass die gerade abgelegten Objekte durch den Kontakt mit dem Greifer verschoben werden.

Aus den Tabellen 4.1 und 4.2 kann man anhand der zurückgelegten Distanzen gut erkennen, dass die Bewegungen des Roboter sehr kantig sind. In Bewegung 6 fährt er in Richtung der x-Achse, bremst ab, da die Teilbewegung zu Ende ist, bewegt sich in der yz-Ebene (Bewegung 7), hält abermals an, und bewegt sich von dort aus in Richtung der negativen x-Achse bis zur Ablage des Objekts. Alleine in den Bewegungen 6 und 7 wird das nsa-Koordinatensystem etwa 1 m translatorisch bewegt. Dabei bremst der Roboter nach jeder Teilbewegung ab und muss dann wieder neu beschleunigen.

Bei der Bewegung 10 und 11 treten die gleichen Probleme auf. Wenn man die Zeit dieser vier Bewegungen addiert, kommt man in Tabelle 4.1 auf 10,42 sec und in Tabelle 4.2 auf 12,06 sec.

Die Ausführungszeit eines kompletten Durchlaufs der Anwendung benötigt für sieben erfolgreich gegriffene, abgelegte und wiederaufgegriffene Objekte und ein erfolgreich gegriffenes aber nicht abelegtes Objekt, da es nicht mehr auf die Ablagefläche passt, 8 min 46 sec. Die Zeit ist mit der Hand gestoppt. In dieser Ausführungszeit ist die Bewegung des Greifers, die Bewegung des Roboters, die Laufzeit der beiden Planer, die Objektmodellierung und die Benutzerinteraktion berücksichtigt. Die gleiche Aufgabe wurde durch einen Menschen durchgeführt, um eine Vergleichszeit zu bekommen, wie schnell der Mensch diese Aufgabe lösen kann. Der Testkandidat durfte nur einen Arm verwenden und wurde angewiesen, die Objekte in einer normalen, nicht hektischen Geschwindigkeit zu bewegen, möglichst so, als würde er gerade seinen Schreibtisch aufräumen. Die Objekte wurden in der gleichen Reihenfolge auf die Aufgreiffläche gelegt. Der Benutzer hat insgesamt 35 sec für die gestellte Aufgabe benötigt. Der Vergleich der Bewegungen zeigt, dass der Mensch viel glattere Bewegungen ohne Unterbrechungen durchführt, und die Bewegungen kurz vor und nach dem Griff genauso schnell wie die restlichen Bewegungen sind. Diese bessere Bewegungsplanung vom Menschen bewirkt, dass dieser eine viel kürzere Strecke mit der Hand zurücklegt, die Bewegungen viel glatter aneinander gereiht und dadurch deutlich schneller sind. Man braucht eigentlich nicht erwähnen, dass der Mensch nicht nach jeder Teilbewegung anhält, bevor er die nächste beginnt.

4.3 Schlussfolgerung

Es ist ein Prototyp mit dem Greifplaner (Kapitel 2) und Ablageplaner (Kapitel 3) entwickelt worden. Es wird ein siebenachsiger Industrierobter mit einem Backengreifer verwendet. Als optischer Sensor wird eine Farbkamera verwendet, die fest am Greifer montiert ist. Mit der Kamera wird eine Aufgreiffläche überwacht. Sobald unbekannte Objekte auf dieser Fläche liegen, wird ein Objektmodell erzeugt und mit diesem die Greifplanung durchgeführt. Ist diese erfolgreich, greift der Roboter nach dem Objekt. Der Griff wird validiert, indem die Distanz zwischen den Greiferbacken gemessen wird. Sind diese komplett geschlossen, war der Griff nicht erfolgreich und der Roboter bewegt sich wieder in Richtung Startposition, von der aus die Aufgreiffläche weiter beobachtet wird. War der Griff erfolgreich, ermittelt der Ablageplaner die Position und Orientierung in der das Objekt auf der Ablagefläche abgelegt

werden soll.

Sobald ein Objekt nicht gepackt werden konnte, weil es nicht mehr auf die Ablagefläche passt, werden die schon abgelegten Objekte wieder von der Ablagefläche aufgegriffen. Der Roboter bewegt sich mit jedem gegriffenen Objekt zurück zur Startposition und wartet dort, bis der Benutzer das Objekt entgegennimmt.

Die Greifplanung und Bewegungsplanung arbeiten so gut zusammen, dass unbekannte Objekte zielsicher gegriffen werden können. Ungenauigkeiten bei der Kamerakalibrierung und der Roboterbewegung verursachen jedoch so starke Fehler, dass der Greifer, gerade wenn das Objekt in den Randbereichen des Bildes abgebildet ist, bis zu 2 cm neben der im Bild geplanten Greifposition platziert wird. Dadurch werden Objekte, die ungefähr so stark sind wie die maximale Greiferöffnung, nur selten erfolgreich gegriffen, obwohl die Greifplanung eine korrekte Greifkonfiguration liefert. Es ist möglich, dass trotz erfolgreichem Griff das Objekt nicht im Greifer verbleibt. Es kann durch auftretende Momente am Objekt dazu kommen, dass das Objekt aus dem Greifer heraus kippt.

Objekte, die sich überlappen, werden als ein Objekt identifiziert. Damit ist die minimale Boundingbox des Objektmodells zu groß. Die Greifplanung selbst arbeitet auf den Konturpixeln und damit ist es durchaus möglich, dass eine Greifkonfiguration gefunden wird. Die Ablage ist auch möglich, wenn das zu große Objekt auf die Ablagefläche passt. Das zweite Objekt, dass dann auf der Aufgreiffläche zurückbleibt, kann in einem weiteren Durchlauf des Greif-Ablage-Prozesses korrekt modelliert werden. Der Gesamtablauf vom Greifen zum Ablegen und dem erneuten Aufgreifen steht bei einem zu großen Objektmodell nichts im Wege. Der Fehler, der gemacht wird, ist, dass auf der Aufgreiffläche zu viel Platz für das Objekt als belegt markiert wird.

Die Hauptprobleme bei der Ablageplanung sind die Positionierungsungenauigkeit vom Roboter während des Griffs des Objekts und fehlerhafte Boundingboxen im Objektmodell. Wenn das Objekt an einer anderen Stelle als der geplanten gegriffen wird, wird diese Ungenauigkeit an die Ablageposition weiter gegeben. Wenn jedoch die Boundingbox des Objekts zu klein ist, können Kollisionen mit anderen Objekten im Ablagebereich auftreten. Sobald die Position der Objekte innerhalb des Ablagebereichs verändert wird, ist ein erneutes Aufgreifen nicht mehr möglich.

Das erneute Aufgreifen konnte nicht durch eine erneute Greifplanung realisiert werden, da die Fehler durch die Kamerakalibierrung zu groß sind. Die Objekte sind so dicht gepackt, dass gerade die Greiferbacken zwischen den Objekten Platz haben. In den Experimenten wurde gezeigt, dass die rea-

le Position des Greifers und die geplante Position bis zu 2 cm voneinander abweichen.

Die Zeit, die für einen Durchlauf der Anwendung für ein Objekt benötigt wird liegt bei etwa 70 sec. Die Zeit, die für die Greifplanung, Ablageplanung und die Objektmodellierung benötigt wird, liegt zusammen im schlechtesten Fall bei 40 ms. Die Bewegungen des Roboters dauern mit sicheren Geschwindigkeiten etwa 57 sec und die restliche Zeit wird benötigt, um den Greifer zu betätigen. Es werden etwa 20 sec benötigt, um die unmittelbaren Bewegungen zum Aufgreifen und Ablegen des Objekts auszuführen. Diese Bewegungen lassen sich nicht beschleunigen solange die Rücktransformation von den kartesischen Koordinaten zu den Gelenkwinkeln nicht exakt kontrolliert werden kann.

Der Roboter kann die Aufgabe nicht so schnell ausführen, wie das der Mensch könnte. Mit den beiden Planern ist jedoch der Grundstock gelegt worden, die Ausführungszeit weiter zu verringern. Die Bewegungsgeschwindigkeit ($< 0,25$ m/s laut Norm ISO-10218) des Roboters ist in einem Bereich, in der keine Verletzungsgefahr für den Menschen besteht, der in der Beispielanwendung an der Aufgreiffläche Objekte ablegt und Objekte wieder entgegen nimmt, die vom Roboter von der Ablagefläche wieder aufgegriffen werden.

Bei der Bewegungsplanung und deren Ausführung mit dem Kuka LBR 4 sind viele Verbesserungen denkbar. Aktuell sind lineare Bewegungen im Kartesischenraum möglich und zeitoptimale Bewegungen im Achsraum nach Schwinn möglich. Bewegungen des TCP auf einer gekrümmten Bahn würden die Möglichkeiten, glatte Bewegungen zu realisieren, deutlich verbessern. Das verkürzt dann auch unmittelbar die Ausführungszeiten einzelner Bewegungen. Die Möglichkeit zwischen zwei Verfahrbefehlen die Bewegung zu überschleifen, wäre eine weiter Verbesserung.

Unmittelbare Probleme, wie die geschwindigkeitsabhängige Rücktransformation, müssen behoben werden, indem man entweder diese selbst berechnet und der Robotersteuerung eindeutige Gelenkwinkel übermittelt, oder eine Möglichkeit findet, den Konfigurationsraum des Roboters so einzuschränken, dass die Rückwärtstransformation eindeutig wird. Wäre man in der Lage, den Konfigurationsraum einzuschränken, ist nicht nur die Rücktransformation lösbar, sondern auch eine Umweltmodellierung einfach möglich. Mit diesem Schritt würde man Selbstkollisionen und Kollisionen mit der statischen Umwelt ausschließen. Im Moment sind solche Kollisionen immer denkbar, da, bei der Steuerung des Roboters über das Fast Research Interface (FRI) alle Softwareendanschläge ignoriert werden.

Nach den Experimenten hat sich herausgestellt, dass bei bestimmten Be-

wegungen, vor allem, wenn der Roboter die Konfiguration gewechselt hat, sehr hohe Momente an den Gelenken aufgetreten sind. Das hat dazu geführt, dass die Robotersteuerung, nach dem Überschreiben des maximalen Grenzwertes, abgeschaltet hat. Diese Problematik lässt sich beheben, indem man die gefahrene Geschwindigkeit online, abhängig von den aktuellen Momenten an den einzelnen Gelenken, anpasst. Eine solche Regelung wird vom Roboter vorgenommen, wenn man ihn durch die Kuka Robot Language bewegt. Diese Momente können nicht automatisch berücksichtigt werden, wenn der Roboter über das FRI gesteuert wird, da nur absolute Konfigurationen zu einem bestimmten Zeitpunkt übermittelt werden. Das FRI überwacht keine gefahrenen Geschwindigkeiten oder auftretende Kräfte. Es sorgt nur dafür, dass die empfangene Konfiguration, während des aktuellen Zeitintervalls eingenommen wird. Es wird nur überwacht, ob Hardware-Grenzwerte bei Geschwindigkeit, Beschleunigung und Momenten überschritten werden.

Nachdem die Schnittstelle zu dem Roboter so ausgelegt ist, dass fast keine Berechnungen von der Robotersteuerung übernommen werden, ist der Aufwand nicht mehr sonderlich hoch, die Bewegungsplanung und Ausführung komplett zu simulieren.

Kapitel 5

Schlussfolgerung

In dieser Arbeit ist ein Prototyp entwickelt worden, der das Greifen, Ablegen und Wieder-Aufgreifen von unbekannten Objekten realisiert. Dieser besteht aus einem Greifplaner, Ablageplaner und einer Softwarekomponente, die die Bewegungsplanung und -ausführung realisieren.

Das Hauptaugenmerk in der Arbeit wurde auf die Greifplanung und Ablageplanung gelegt. Beide Planer arbeiten im zweidimensionalen Raum mit einer vorgegebenen Umweltsituation. Sowohl die Ablagefläche, als auch die Aufgreiffläche sind bekannt. Auf Basis eines Kamerabildes wird ein Modell des realen Objekts erstellt. Der Greifplaner sucht mit Hilfe der Hough-Transformation die Greifkonfiguration für dieses Objektmodell.

Bei der Ablageplanung stand das Problem im Vordergrund, den vorhanden Freiraum auf der Ablagefläche bestmöglich zu nutzen. Der entwickelte Algorithmus verwaltet Freiräume, aus denen mit einer Zielfunktion ein passender gewählt wird. Die Platzierung des Objekts innerhalb des Freiraums wird durch eine Heuristik realisiert. Bei der Ablageplanung ist berücksichtigt worden, dass die Objekte vom Roboter abgelegt und wieder von der Ablagefläche aufgegriffen werden können.

Diese beiden Planer wurden in einem Prototypen integriert. Dieser ermittelt mit Hilfe des Greifplaners die Greifkonfiguration für eines der Objekte auf der Aufgreiffläche, führt den Griff aus und evaluiert diesen. Anschließend wird das Objekt auf die Position auf der Ablagefläche gelegt, die durch die Ablageplanung berechnet wurde. Sobald kein Platz für ein weiteres Objekt ist, wird die Ablagefläche vom Roboter wieder leer geräumt.

Bei den Experimenten ist untersucht worden, welche Objekte überhaupt erfolgreich gegriffen und abgelegt werden können. Weiter ist die Laufzeit und

Bewegungsgeschwindigkeit des Prototypen betrachtet worden.

Bei den Experimenten hat sich gezeigt, dass der Greifplaner erfolgreich für Objekte arbeitet, die wenig parallele Kanten auf ihrer Textur haben. Selbst Objekte, die übereinander auf der Aufgreiffläche liegen, wurden erfolgreich gegriffen. Für die Ablage haben die Experimente gezeigt, dass die Ablage erfolgreich ist, sobald die minimale Boundingbox des Objektmodells korrekt bestimmt wurde.

Die Zeit die für die Planung des Griffs und der Ablage benötigt wird, liegt bei maximal 33 ms bzw. bei unter 1 ms. Diese Laufzeiten sind so kurz, dass vom Benutzer keine Verzögerung wahr genommen wird. Die Geschwindigkeiten, die vom Roboter gefahren werden, liegen innerhalb der Norm ISO-10218 für den Fall, dass ein Mensch mit dem Roboter interagiert. Das heißt, die maximale Geschwindigkeit ist $0,25$ m/s. Bei einem Test, bei dem acht verschiedene Objekte gegriffen, abgelegt und wieder aufgegriffen wurden sind die Ausführungszeiten des Prototypen und eines Menschen verglichen worden. Der Mensch benötigt für diese Aufgabe maximal 35 sec und der Roboter 8 min 46 sec. Durch diesen Vergleich wird klar, dass der Prototyp noch nicht im Bezug auf die Geschwindigkeit mit dem Menschen konkurrieren kann. Es wurden jedoch die maximalen Geschwindigkeiten aus der Norm erreicht. Das ständige Anhalten, nach den Teilbewegungen des Roboters, und das erneute Anfahren verhindern kürzere Ausführungszeiten.

In weiteren Arbeiten muss die Implementierung der Bewegungsplanung verbessert werden, damit glatte Bewegungen ohne ständiges Anhalten und somit kürzere Laufzeiten möglich sind. Ein Umweltmodell in Kombination mit einer selbst berechneten Rückwärtstransformation würde höhere Geschwindigkeiten auch bei ungünstigen Gelenkwinkeln ermöglichen. Mit diesen beiden Verbesserungen lassen sich kürzere Bahnen von der Aufgreiffläche hin zur Ablagefläche berechnen.

Eine Erweiterung des Objektmodells auf den dreidimensionalen Raum würde die Probleme des Greifplaners, die durch die Textur bedingt sind, beheben. Für die Ablageplanung wäre eine solche Erweiterung auch ein großer Gewinn. Damit könnten die Objekte nicht nur auf einer Fläche abgelegt werden, sondern auch in eine Kiste oder in einen Schrank.

Literaturverzeichnis

[BBS Winsen 07] BBS Winsen. *http://www.bs-wiki.de*, 8.12.2007.

[Berenson 07] Dimitry Berenson, Rosen Diankov, Koichi Nishiwaki, Satoshi Kagami & James Kuffner. *Grasp Planning in Complex Scenes*. IEEE-RAS Int. Conf. on Humanoid Robots, 2007.

[Bischoff 95a] E. E. Bischoff, F. Janetz & M. S. W. Ratcliff. *Loading pallets with non-identical items*. European Journal of Operational Research, Band 84, Nr. 3, Seiten 681–692, August 1995.

[Bischoff 95b] E. E. Bischoff & M. S. W. Ratcliff. *Issues in the development of approaches to container loading*. Omega, Band 23, Nr. 4, Seiten 377–390, 1995.

[Bischoff 06] E.E. Bischoff. *Three-dimensional packing of items with limited load bearing strength*. European Journal of Operational Research, Band 168, Nr. 3, Seiten 952–966, February 2006.

[Bodenbagen 09] Leon Bodenbagen, Dirk Kraft, Mila Popović, Emre Başeski, Peter Eggenberger Hotz & Norbert Krüger. *Learning to grasp unknown objects based on 3D edge information*. In Proceedings of the 8th IEEE International Conference on Computational Intelligence in Robotics and Automation, CIRA'09, Seiten 421–428, 2009.

[Bone 01] Gary M. Bone & Yonghui Du. *Multi-Metric Comparison of Optimal 2D Grasp Planning Algorithms*. In Proceedings of the 2001 IEEE International Conference on Robotics and Automation, Seiten 3061–3066. IEEE, May 21-26 2001.

[Bone 08] Gary M. Bone, Andrew Lambert & Mark Edwards. *Automated Modeling and Robotic Grasping of Unknown Three-Dimensional Objects*. IEEE International Conference on Robotics and Automation, Seiten 292–298, 2008.

[Bouganis 06] Alexandros Bouganis & Murray Shanahan. *On Packing 2D Irregular Shapes*. In Proceeding of the 2006 conference on ECAI 2006: 17th European Conference on Artificial Intelligence, Seiten 853–854. IOS Press, 2006.

[Bradski 00] G. Bradski. *The OpenCV Library*. Dr. Dobb's Journal of Software Tools, 2000.

[Christopoulos 07] V.N. Christopoulos & P. Schrater. *Handling shape and contact location uncertainty in grasping two-dimensional planar objects*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, Seiten 1557–1563. IEEE, 2007.

[Cornellà 03] J. Cornellà & R. Suárez. *On 2D 4-finger frictionless optimal grasps*. In Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, volume 4, Seiten 3680–3685, 2003.

[Cornellà 05] J. Cornellà & R. Suárez. *Fast and flexible determination of force-closure independent regions to grasp polygonal objects*. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Seiten 766–771. IEEE, 2005.

[Cornellà 09] Jordi Cornellà & Raul Suárez. *Efficient Determination of Four-Point Form-Closure Optimal Constraints of Polygonal Objects*. IEEE Transactions on Automation Science and Engineering, Band 6, Seiten 121–130, 2009.

[DFKI Bremen 11] DFKI Bremen. *http://robotik.dfki-bremen.de/*, 24.3.2011.

[DLR 11] DLR. *www.dlr.de*, 24.3.2011.

[Egeblad 10] Jens Egeblad, Claudio Garavelli, Stefano Lisi & David Pisinger. *Heuristics for container loading of furniture*. European Journal of Operational Research, Band 200, Nr. 3, Seiten 881–892, 2010.

[Eley 02] Michael Eley. *Solving container loading problems by block arrangement*. European Journal of Operational Research, Band 141, Nr. 2, Seiten 393–409, September 2002.

[Gamma 95] Erich Gamma, Richard Helm, Ralph Johnson & John Vlissides. Design patterns: elements of reusable object-oriented software. Addison-Wesley professional computing series, 1995.

[Gonzalez 02] Rafael Gonzalez & Richard Woods. Digital image processing (2nd edition). Prentice Hall, 2002.

[Gorges 09] Nicolas Gorges & Heinz Wörn. *Learning an Object-Grasp Relation for Silhouette-Based Grasp Planning.* In Torsten Kröger & Friedrich M. Wahl, editeurs, Advances in Robotics Research, Seiten 227–237. Springer Berlin Heidelberg, 2009.

[Guennebaud 11] Gaël Guennebaud, Benoît Jacob*et al.* *Eigen v2.* http://eigen.tuxfamily.org, 25.3.2011.

[Haddadin 07] S. Haddadin, A. Albu-Schäffer & G. Hirzinger. *Safety evaluation of physical human-robot interaction via crash-testing.* In Robotics: Science and Systems Conference, Seiten 217–224. Citeseer, 2007.

[Haddadin 08] S. Haddadin, A. Albu-Schäffer & G. Hirzinger. *The role of the robot mass and velocity in physical human-robot interaction-part I: Non-constrained blunt impacts.* In International Conference on Robotics and Automation, Seiten 1331–1338. IEEE, 2008.

[Haddadin 11] S. Haddadin, A. Albu-Schäffer & G. Hirzinger. *Safe Physical Human-Robot Interaction: Measurements, Analysis and New Insights.* Robotics Research, Seiten 395–407, 2011.

[Heikkila 97] J. Heikkila & O. Silven. *A four-step camera calibration procedure with implicit image correction.* In CVPR '97 Proceedings of the 1997 Conference on Computer Vision and Pattern. IEEE, 1997.

[Hough 62] Paul Hough. *Method and Means for Recognizing Complex Patterns.* U.S. Patent 3.069.654, December 1962.

[Huebner 09] Kai Huebner, Kai Welke, Markus Przybylski, Nikolaus Vahrenkamp, Tamim Asfour, Danica Kragic & Rüdiger Dillmann. *Grasping Known Objects with Humanoid Robots: A Box-Based Approach.* In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Seiten 1–6, 2009.

[Hyde 10] M. Hyde. *A genetic programming hyper-heuristic approach to automated packing.* Doktorarbeit, The University of Nottingham, 2010.

[IPA 11] Fraunhofer IPA. *http://www.care-o-bot.de/*, 24.3.2011.

[ISO-10218 06] ISO-10218. *Robots for industrial environments - Safety requirements - Part 1: Robot*, 2006.

[KUKA Roboter GmbH 08] KUKA Roboter GmbH. *Betriebsanleitung Leichtbauroboter 4*, 6 2008.

[Lodi 02a] A. Lodi, S. Martello & M. Monaci. *Two-dimensional packing problems: A survey.* European Journal of Operational Research, Band 141, Nr. 2, Seiten 241–252, 2002.

[Lodi 02b] A. Lodi, S. Martello & D. Vigo. *Recent advances on two-dimensional bin packing problems.* Discrete Applied Mathematics, Band 123, Nr. 1-3, Seiten 379–396, 2002.

[Logitech 11] Logitech. *www.logitech.com*, 2.3.2011.

[Martello 00] S. Martello, D. Pisinger & D. Vigo. *The three-dimensional bin packing problem.* Operations Research, Seiten 256–267, 2000.

[Martello 07] Silvano Martello, David Pisinger, Daniele Vigo, Edgar Den Boef & Jan Korst. *Algorithm 864: General and robot-packable variants of the three-dimensional bin packing problem.* ACM Trans. Math. Softw., Band 33, 2007.

[Marturi 10] N. Marturi. Vision based grasp planning for robot assembly. Masterarbeit, Department of Technology at Örebro University, 2010.

[Miller 03] Andrew T. Miller, Steffen Knoop, Henrik I. Christensen & Peter K. Allen. *Automatic Grasp Planning Using Shape Primitives.* 2003.

[Ortmann 10] F. Ortmann. *Heuristics for Offline Rectangular Packing Problems.* Doktorarbeit, Department of Logistics, Stellenbosch University, 2010.

[Parreño 08] F. Parreño, R. Alvarez-Valdes, J. M. Tamarit & J. F. Oliveira. *A Maximal-Space Algorithm for the Container Loading Problem.* INFORMS J. on Computing, Band 20, Seiten 412–422, 2008.

[pi4_robotics 11] pi4_robotics. *http://www.pi4-robotics.com*, 24.3.2011.

[Pixar 11] Pixar. *http://www.pixar.com/*, 26.3.2011.

[Roa 09] Máximo A. Roa & Raúl Suárez. *Computation of independent contact regions for grasping 3-D objects.* IEEE Transactions on Robotics, Band 25, Seiten 839–850, 2009.

[Saxena 08] Ashutosh Saxena, Justin Driemeyer & Andrew Y. Ng. *Robotic Grasping of Novel Objects using Vision.* The International Journal of Robotics Research, Seiten 157–173, 2008.

[Schörner 10] Gernot Schörner. Konzeption und Implementierung einer intuitiven Schnittstelle zwischen Mensch und Robotergreifer. Bachelorarbeit, Univresität Bayreuth, Institut für Informatik, 2010.

[Schreiber 10] Dr. Günter Schreiber. *KUKA Fast Research Interface (FRI).* KUKA Roboter GmbH, 2010.

[Schwinn 99] W. Schwinn. Grundlagen der Roboterkinematik. Schwinn, 1999.

[Smith 99] Gordon Smith, Gordon Smith, Eric Lee, Ken Goldberg, Karl Böhringer & John"Craig. *Computing Parallel-Jaw Grip Points.* 1999.

[Suárez 06] Raúl Suárez, Máximo Roa & Jordi Cornella. *Grasp quality measures.* Rapport technique, Institut d'Organització i Control de Sistemes Industrials, Universitat Politècnica de Catalunya, 2006.

[Suzuki 85] S. Suzuki & K. Abe. *Topological Structural Analysis of Digital Binary Images by Border Following.* CVGIP, Band 30, Seiten 32–46, 1985.

[Tanenbaum 01] Andrew S. Tanenbaum. Modern operating systems. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 2001.

[Torra 09] Vicenç Torra, Isaac Cano, Sadaaki Miyamoto & Yasunori Endo. *Container loading for nonorthogonal objects: an approximation using local search and simulated annealing.* Soft Computing - A Fusion of Foundations, Methodologies and Applications, 2009.

[USB-IF Inc. 11] USB-IF Inc. *www.USB.org*, 2.3.2011.

[Whelan 96] P.F. Whelan & B.G. Batchelor. *Automated packing systems-a systems engineering approach.* IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans, Band 26, Nr. 5, Seiten 533–544, 1996.

[Wikimedia Foundation Inc. 11] Wikimedia Foundation Inc. *de.wikipedia.org/wiki/Amontonssche_Gesetze*, 4.3.2011.

[Yamazaki 06] Kimitoshi Yamazaki, Masahiro Tomono, Takashi Tsubouchi & Shin'ichi Yuta. *A Grasp Planning for Picking up an Unknown Object for a Mobile Manipulator.* In Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Florida, USA, Seiten 2143–2149, 2006.

[Zacharias 09] Franziska Zacharias, Christoph Borst & Gerd Hirzinger. *Object-Specific Grasp Maps for Use in Planning Manipulation Actions.* Proceedings of the German Workshop on Robotics, Seite 203ff, 2009.

Anhang A

Benchmarkgenerator

Mit diesem Programm sind die Benchmarktests erzeugt worden, mit denen die theoretischen Experimente mit dem Ablageplaner durchgeführt worden sind.

```
/*
  Bischoff/Ratcliff style dataset generator
  as described in:
  "Issues in the development
  of approaches to container loading"

  Sam Allen
  School of Computer Science
  University of Nottingham
  Jubilee Campus
  Wollaton Road
  NOTTINGHAM NG8 1BB
  email: sda@cs.nott.ac.uk
  web: www.cs.nott.ac.uk/~sda
*/

#include <iostream>
#include <algorithm>
#include <vector>
#include <fstream>

#define loop(a) for(int curPos = 0; curPos < a; ++curPos)

int seed;

const int aa = 16807; // (pseudo)random number constants
const int m = 2147483647;
const int q = 127773;
```

```
29 const int r = 2836;
30
31 const int a[] = {30, 25, 20}; // minimum dimensions
32 const int b[] = {120, 100, 80}; // maximum dimensions
33
34 const int L = 2; // stability constraint
35
36 const int cWidth = 233; // container dimensions
37 const int cLength = 587;
38 const int cHeight = 220;
39
40 const int Tc = cWidth * cLength * cHeight; // container volume
41
42 using namespace std;
43
44 // (pseudo)random number generator as described in "Random
45 // number generators: Good ones are hard to find"
46 double random1() {
47
48   int hi = seed / q;
49   int lo = seed % q;
50   int test = aa * lo - r * hi;
51
52   if(test > 0)
53     seed = test;
54   else
55     seed = test + m;
56   return 1.0 * seed / m;
57 }
58
59 void generateData(int maxI, int maxSets, char* fileName){
60
61   int p = 0; // dataset number
62   int C; // sum of box volumes
63
64   fstream outfile(fileName, ios::out);
65
66   if(!outfile.is_open()){
67     cerr << "Error:␣couldn't␣open␣"
68       << fileName << "␣for␣writing!" << endl;
69     return;
70   }
71
72   outfile << "␣" << maxSets << endl;
73
74   vector<vector <int> > boxVector;
75
76   loop(maxSets){
77
```

```
78      ++p;
79
80      C = 0;
81
82      boxVector.clear();
83
84      seed = 2502505 + 100 * (p - 1);
85
86      outfile << "␣" << p << "␣" << seed << endl;
87      outfile << "␣" << cLength << "␣" << cWidth
88        << "␣" << cHeight << endl;
89      outfile << "␣" << maxI << endl;
90
91      loop(10)
92        random1();
93
94      loop(maxI){
95        int d[3];
96        bool f[3];
97
98        loop(3)
99    d[curPos] = a[curPos] + int(random1()
100       * (b[curPos] - a[curPos] + 1));
101
102
103       sort(d, d+3);
104       reverse(d, d+3);
105
106       loop(3)
107   f[curPos] = (d[curPos] / d[2] < L);
108
109       // 3 dimensions, 3 possibly feasible orientations, volume, m
110       int tempBox[] = {d[0], d[1], d[2], f[0], f[1],
111            f[2], d[0] * d[1] * d[2], 1};
112
113       boxVector.push_back(
114   vector<int>(tempBox,
115         tempBox + sizeof(tempBox) / sizeof(tempBox[0])));
116
117     }
118
119     loop((int)boxVector.size())
120       C += boxVector[curPos][6];
121
122     int k;
123     while(true){
124       k = int(random1() * maxI);
125
126       if(Tc < C + boxVector[k][6])
```

```
127   break;
128       else {
129   ++boxVector[k][7];
130   C += boxVector[k][6];
131       }
132     }
133
134     loop((int)boxVector.size()){
135       outfile <<
136   "␣" << curPos+1 << "␣" <<
137   boxVector[curPos][0] << "␣" <<
138   boxVector[curPos][3] << "␣" <<
139   boxVector[curPos][1] << "␣" <<
140   boxVector[curPos][4] << "␣" <<
141   boxVector[curPos][2] << "␣" <<
142   boxVector[curPos][5] << "␣" <<
143   boxVector[curPos][7]
144   << endl;
145     }
146   }
147   outfile.close();
148 }
149
150 int main(int argc, char* argv[]) {
151
152   int maxI, maxSets;
153   char fileName[128];
154
155   cout << "Number␣of␣box␣types:␣";
156   cin >> maxI;
157
158   cout << "Number␣of␣sets␣to␣generate:␣";
159   cin >> maxSets;
160
161   cout << "Output␣filename:␣";
162   cin >> fileName;
163
164   generateData(maxI, maxSets, fileName);
165
166   return 0;
167 }
```

Anhang B

Inhalt der CD

Auf der CD sind auf der ersten Ebene zwei Ordner „src“ und „exp“.

Der Ordner ßrc“ auf der CD enthält den gesamten Quellcode, aufgeteilt in einzelne Projekte, die im Zuge dieser Arbeit entwickelt oder verwendet wurden:

- src/LBRUtil: In diesem Projekt sind die Mathematik-Bibliothek Eigen v2 [Guennebaud 11] und diverse Klassen enthalten, die in mehreren Projekten benötigt werden.
- src/LBR: Dieses Projekt ist die Schnittstelle zum Roboter.
- src/LBRCam: Hier sind die Softwarekomponenten implementiert, die die Kamerabilder bereitstellen und die Verwendung der Kameraparameter ermöglicht.
- src/LBRJB: In diesem Projekt ist der Greifplaner, der Ablageplaner und die Beispielanwendung implementiert.
- src/Gripper: In diesem Projekt befindet sich die Implementierung für den Greifer [Schörner 10].
- src/KuhnKernel: Dieses Projekt wurde Herrn Dipl.-Inf. Stefan Kuhn implementiert. Es wird nur die Basisimplementierung der logischen Kamera verwendet.
- src/visualization: Dieses Projekt wurde Herrn Dipl.-Inf. Stefan Kuhn implementiert. Es wird benötigt, um die Ergebnisse der Ablageplanung während der Testläufe zu visualisieren.

In den Klassen „src/LBRJB/src/apps/Application.h“ und „src/LBRJB/ src/apps/ApplicationsWith3DViewer.h“ sind alle Anwendungen implementiert.

Der Greifplaner ist in der Datei „src/LBRJB/src/_intern/ HoughGraspPlanner.cpp“ implementiert, der Ablageplaner in der Datei „src/LBRJB/src/_intern/ FRDeliveryPlanner.cpp“.

Der Ordner „exp“ enthält die Ergebnisse der Experimente, der Ordner „exp/greifen“ die Bilder, in denen jeweils die beiden Greifregionen eingezeichnet sind, aus denen die Greifkonfiguration berechnet wird. Weiter enthält der Ordner ein Video, in dem verschiedene Objekte gegriffen werden.

Der Ordner „exp/ablegen“ enthält die Ergebnisse der Benchmarktests mit dem Ablageplaner.

In dem Ordner „exp/prototyp“ sind mehrere Ordner enthalten, die jeweils die Ausgaben eines Testlaufs beinhalten. Für jeden Testlauf sind die Bilder mit den Ergebnissen des Greifplaners und die Logdatei mit der Kommandozeilenausgabe sowie ein Video enthalten. Dieses zeigt, wie der Roboter die Ablagefläche so voll packt, dass kein weiteres Objekt, das auf der Aufgreiffläche liegt, abgelegt werden kann. Weiter zeigt es wie der Roboter die volle Ablagefläche wieder entpackt und ein Mensch die Objekte entgegen nimmt. Für jedes Objekt ist auch immer das Ergebnis der Greifplanung zu sehen.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Bayreuth, den 28.3.2011

.....
Johannes Baumgartl