Untersuchung von Samplingverfahren zur Erstellung aufgabenorientierter Bewegungskarten

eingereichte
DIPLOMARBEIT
von

cand. ing. Andreas Dömel

geb. am 25.04.1984
wohnhaft in:
Hirtenweg 10
82229 Hechendorf
Tel.: 08152 3961387

Lehrstuhl für
Datenverarbeitung
Technische Universität München

Univ.-Prof. Dr.-Ing. Klaus Diepold

Betreuer: Dr.-Ing. Michael Suppa, Univ.-Prof. Dr.-Ing. Klaus Diepold
Abgabe: 9. Februar 2010

Inhaltsverzeichnis

Abkürzungsverzeichnis

$\mathbf{q}$	Konfiguration des Roboters
$\mathcal{A}(\mathbf{q})$	physikalischer Raum, der vom Roboter in Konfiguration $\mathbf{q}$ eingenommen wird
$\mathcal{A}_i(\mathbf{q})$	physikalischer Raum, der vom Roboterteil i in Konfiguration $\mathbf{q}$ eingenommen wird
$\mathcal{B}$	Teilbereich von $\mathcal{P}_{free}$
$\mathcal{C}$	Konfigurationsraum
$\mathcal{C}_{free}$	Bereich des Konfigurationsraums ohne Kollisionen
$\mathcal{C}_{known}$	bekannter Bereich des Konfigurationsraums
$\mathcal{C}_{occ}$	Bereich des Konfigurationsraums in dem Kollisionen vorliegen
$\mathcal{C}_{unk}$	unbekannter Bereich des Konfigurationsraums
$\mathcal{O}$	durch Hindernisse eingenommener physikalischer Raum
$\mathcal{P}$	physikalischer Raum
$\mathcal{P}_{free}$	von Hindernissen freier Bereich von $\mathcal{P}$
$\mathcal{V}(x)$	Bereich in dem Daten durch eine Messung an x aufgenommen werden
$\mathcal{X}(x)$	$\mathbf{q} \in \mathcal{C}_{unk} : x \in \mathcal{A}(\mathbf{q})$
$ERD_c(x)$	Entropie-Reduktions-Dichte an der Stelle x
H	*C-space Entropy*
M	Anzahl der Roboterteile
N	Anzahl der Freiheitsgrade des Roboters

$N_{Samples}$	Anzahl der durch Sampling erzeugten Konfigurationen
P	Menge aller Roboterteilpaare, für die gelten muss: $\mathcal{A}_i(\mathbf{q}) \cap \mathcal{A}_j(\mathbf{q}) = \emptyset$
$p(\mathcal{B})$	Wahrscheinlichkeit für $\mathcal{B} \subset \mathcal{P}_{free}$
Q	binäre Zufallsvariable des Besetzungszustandes von $\mathbf{q}$
$vol(...)$	Volumen von ...
X	gleichverteilte Zufallsvariable auf Basis der Ergebnisse der Planungsdurchläufe
x_s	Lage des Sensors im physikalischen Raum
M0...M3	View-Point Sampling Methoden
DH	Denavit-Hartenberg
DLR	Deutsches Zentrum für Luft- und Raumfahrt
DoF	Freiheitsgrade
IG	Informationsgewinn
MER	Maximal Entropy Reduction
MPV	Maximal Physical Volume
NBV	nächster bester Sensorpunkt
PRM	Probabilistic Roadmap
RDT	Rapidly-Exploring Dense Tree
RDTPRM	kombiniertes Verfahren aus RDT und PRM
SamPP	Sampling-based Path Planning
SBP-Simulator	Sensor-based Motion Planning Simulator

Kapitel 1

Einleitung

Samplingbasierte Pfadplanung ist ein etabliertes Werkzeug für komplexe Pfadplanungsprobleme. Insbesondere für Roboter mit vielen Freiheitsgraden, wie z.B. humanoide Roboter, sind diese Verfahren eine gängige und oft auch die einzig praktikable Lösung. Das Einsatzgebiet reicht dabei von der Berechnung des optimalen Pfades zur Montage eines Getriebes über Bewegungsplanungen von virtuellen Schauspielern bis hin zur Entwicklung von neuen Medikamenten [15].
Ein besonderer Schwerpunkt wird auf das Lösen von Planungsaufgaben mit Engstellen oder Käferfallen gesetzt, da dies die potenziell schwierigsten Fälle sind. So wird z.B. ein L-förmiger Gegenstand durch ein Loch oder Labyrinth bewegt oder ein Roboter durch einen langen, engen Gang manövriert. Anhand dieser Szenarien wird die Leistungsfähigkeit der entwickelten Algorithmen demonstriert. Ein bekannter Fall für ein solches Szenario ist das α-Puzzle, Abbildung 1.1, das von James Kuffner mit einem *RDT*-Verfahren gelöst wurde.

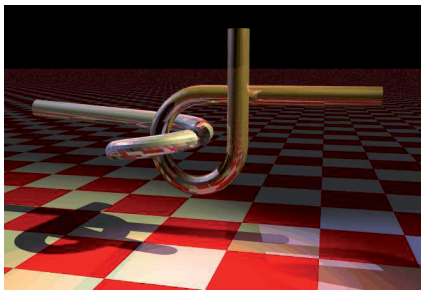


Abbildung 1.1: α-Puzzle [15]

Mit dem sich ausweitenden Einsatz von Robotern in der Industrie, aber auch bei kleinen Handwerksbetrieben und im Haushalt stellt sich das Pfadplanungsproblem in einem sehr praktischen Umfeld. Von den dafür zu entwickelnden Pfadplanern wird aber nicht in erster Linie gefordert, hoch komplizierte Planungsaufgaben wie die des Geduldspiels zu lösen, sondern die für ihre Aufgabe nötigen Planungen effizient und flexibel durchzuführen.
Die bisher entwickelten Planer sind meist dafür konzipiert, ein möglichst breites Spektrum an Planungsaufgaben abdecken zu können. Wird der Pfadplaner aber für einen Roboter mit einem bestimmten Aufgabenbereich ent-

wickelt, so ist es ausreichend, wenn nur die dabei auftretenden Planungsprobleme gelöst werden können.
Durch den universellen Ansatz der gängigen samplingbasierten Pfadplaner ist es nicht möglich, den Planungsvorgang durch schon vorhandenes Wissen über die Planungsaufgabe zu stützen.
Diese Punkte und die Tatsache, dass Robotersysteme durch eine zunehmende Steigerung der Autonomie auf eine effiziente Pfadplanung immer mehr angewiesen sind, führen dazu, sich mit einer auf Anwendungen zugeschnitten Pfadplanung zu beschäftigen.
Ziel dieser Arbeit ist, mit samplingbasierter Pfadplanung Methoden, die für die jeweiligen Anwendungen angepasst sind zu entwickeln und auf ihre Eignung hin zu überprüfen. Dafür werden zwei zentrale Aufgabenstellungen im Bereich der autonomen Robotik behandelt. Zum einen die Exploration einer teilweise unbekannten Umgebung und zum anderen diverse Manipulationsaufgaben mit einem humanoiden Roboter.
Die dafür benötigten Grundlagen werden in Kapitel 2 behandelt. Der Einsatz eines Hand-Auge-Systems zur Exploration mit samplingbasierten Pfadplanungsmethoden wird in Kapitel 3 entwickelt und evaluiert. Kapitel 4 enthält die Konzeption und Umsetzung einer Pfadplanung für Manipulationsaufgaben mit einem humanoiden Robotersystem. In Kapitel 5 wird die Arbeit mit einer Diskussion der ermittelten Ergebnisse sowie einem Ausblick auf künftige Entwicklungsaufgaben abgeschlossen.

Kapitel 2

Grundlagen

In diesem Kapitel werden die für die Entwicklung von anwendungsorientierten Pfadplanungsverfahren benötigten Grundlagen hinsichtlich der beiden betrachteten Szenarien dargestellt. In Abschnitt 2.1 wird kurz auf die Planung von Griffen eingegangen. Eine Einführung in samplingbasierte Pfadplanung wird in Abschnitt 2.2 gegeben. Die Grundlagen für einen Explorationsvorgang werden in Abschnitt 2.3 dargestellt.

2.1 Greifplanung

Eine grundlegende Aufgabe von autonomen Robotern ist die Manipulation von Gegenständen. Um einen möglichst flexiblen Einsatz dieser Systeme zu gewähren, wurden komplexe, der menschlichen Hand nachempfundene Greifsysteme, wie z.B. die in Abschnitt 4.1 vorgestellte DLR-Hand II entwickelt. Die Planung von möglichen Griffen mit einem solchen Manipulator ist eine komplexe Problemstellung.
Jeder Griff ist durch die Position der Fingerspitzen auf dem Gegenstand definiert. Da reale Gegenstände vielfältige Möglichkeiten bieten, die Fingerspitzen zu platzieren, ergeben sich viele potentielle Griffe. Die verschiedenen Greifkonfigurationen haben unterschiedliche Qualitäten. So ist je nach Position der Finger die Stabilität des Griffs hinsichtlich der Kräfte und Momente unterschiedlich. Eine hohe Qualität hat ein Griff dann, wenn auch kleine Fingerkräfte den Gegenstand trotz der im jeweiligen Szenario wirkenden Störgrößen in einem stabilen Zustand halten können.
Neben der Qualität des Griffs muss auch die Planungsdauer in Betracht gezogen werden. Der Greifplaner ist nur dann sinnvoll einsetzbar, wenn der Griff innerhalb der in der jeweiligen Anwendung zur Verfügung stehenden Zeit berechnet werden kann. Oft ist es besser, in kürzerer Zeit einen nur ausreichend guten Griff zu finden.
Für die Bewertung der Griffe oder der Anforderungen, die durch ein Ob-

jekt entstehen können wurde der sogenannte *Wrench Space* entwickelt. Mit diesem Werkzeug kann berechnet werden, welche Kräfte und Momente mit bestimmten Fingerpositionen ausgeglichen werden können, also wie stabil der entsprechende Griff ist. Umfasst der *Wrench Space* alle Kraft-Moment-Kombinationen, die in der Aufgabenstellung entstehen können, ist der Griff ausreichend.
Es lässt sich mit diesem Ansatz auch berechnen, welche Momente und Kräfte ausgeglichen werden müssen, wenn auf ein Objekt Störkräfte wirken. Damit kann die Anforderung an den Griff auch ohne explizite Beschreibung der Kräfte in einem Szenario anhand der zu manipulierenden Objekte abgeschätzt werden.
Ist mit der Greifplanung ein Griff gefunden worden, wird dieser durch eine inverse Kinematik in eine Greifkonfiguration umgewandelt und kann für die Pfadplanung eingesetzt werden. Da in dieser Arbeit kein Greifplaner eingesetzt wird, sondern die Greifkonfigurationenen vorausgesetzt werden, wird nicht tiefer auf die Greifplanung eingegangen. Für ausführliche Informationen über die Greifplanung kann die Veröffentlichung von Borst et al. [1] herangezogen werden.

2.2 Samplingbasierte Pfadplanung

Soll ein Roboter von einer aktuellen Konfiguration in eine andere bewegt werden, müssen die Gelenke des Roboters dementsprechend verändert werden. Dabei müssen in der Regel Nebenbedingungen wie z.B. Kollisionsfreiheit beachtet werden. Da das Ausführen von Bewegungen in vielen Robotersystemen eine grundlegende Rolle spielt, ist Pfadplanung ein in der Robotik häufig auftretendes Problem.
Eine übliche Herangehensweise, dieses Problem zu lösen, ist bei Industrieroboter, den Pfad zu *teachen*. In der Regel werden dem Roboter dafür einige Konfigurationen zwischen Start- und Zielkonfiguration vorgegeben und dazwischen wird eine Interpolationsfunktion genutzt. Dieses Verfahren hat allerdings den Nachteil, dass der Einsatz der Roboters sehr unflexibel ist. Ändern sich die Rahmenbedingungen, wie zum Beispiel die Umgebung, muss der Vorgang wieder neu *geteacht* werden. Ein weiteres Problem bei diesem Vorgehen ist, dass sich für komplizierte Roboterkinematiken das Finden geeigneter Zwischenkonfigurationen wenig intuitiv gestaltet. Ist die Umgebung, in welcher der Roboter agieren soll zudem sehr eng, z.B. bei Montageaufgaben, ist auch der Bediener des Roboters schnell überfordert. Daher, und auf Grund der allgemeinen Bestrebung, autonome Roboter zu gestalten, wird schon lange versucht, den Pfadplanungsvorgang zu automatisieren.
Bei den meisten Methoden wird dabei der Pfad im sogenannten Konfigu-

rationsraum, siehe Abschnitt 2.2.1, des Roboters gesucht. Einer der frühen Ansätze war, die Hindernisse in den Konfigurationsraum zu transformieren und einen kollisionsfreien Pfad analytisch zu finden [15]. Der Rechenaufwand für diese Methode ist allerdings sehr hoch. Ein darauf aufbauender Ansatz ist die *Potential Field Methode* [15]. Hier wird ein virtuelles Kraftfeld um die Hindernisse simuliert, so dass der Roboter einen kollisionsfreien Pfad findet, indem er sich durch die resultierende Gesamtkraft von allen Hindernissen möglichst weit entfernt aufhält. Enge Passagen im Konfigurationsraum sind mit dieser Methode allerdings schwer zu finden, außerdem treten Probleme mit lokalen Minima im Potentialfeld auf. Ein weiterer Nachteil ist, dass für diese Verfahren die Hindernisse explizit in den Konfigurationsraum des Roboters transformiert werden müssen. Für Roboter mit vielen Freiheitsgraden ist dies rechentechnisch sehr aufwändig bzw. nicht möglich.
Samplingbasierte Pfadplanungsverfahren umgehen dieses Problem, indem der Konfigurationsraum nicht explizit berechnet wird, sondern durch einzelne Konfigurationen, welche miteinander verbunden werden können, angenähert wird.
Die Methode, diese Konfigurationen zu gewinnen wird Samplingstrategie genannt und in Abschnitt 2.2.2 näher behandelt. Um die Verbindung zwischen diesen Zuständen auf Kollisionsfreiheit prüfen zu können, wird ein sogenannter lokaler Planer eingesetzt, siehe dazu Abschnitt 2.2.3.
Auf Grundlage dieser Methodik sind vielfältige Algorithmen entwickelt worden, welche einen globalen Planer realisieren. Dieser gibt vor, wie die durch Sampling gewonnenen Konfigurationen zur Pfadplanung verwendet werden. Die Algorithmen lassen sich dabei in zwei Untergruppen unterteilen. Zum einen die *Rapid-Exploring Dense Trees (RDT)*, siehe Abschnitt 2.2.4, und zum anderen die *Probabilistic Roadmaps (PRM)*, siehe Abschnitt 2.2.5.
Der Nachteil von samplingbasierten Pfadplanern ist, dass durch die Annäherung des Konfigurationsraums durch diskrete Konfigurationen solange keine Aussage über die Existenz eines Pfades getroffen werden kann, bis ein Pfad gefunden worden ist. Mit einer geeigneten Wahl der Samplingstrategie kann aber erreicht werden, dass jeder existierende Pfad bei ausreichend langer Planungsdauer gefunden wird. Ein weiteres Problem ist, dass die Planungsdauer bei Samplingstrategien mit zufälligen Samples nicht determinierbar ist.
Es zeigt sich allerdings, dass diese Beschränkungen bei praktischen Pfadplanungsproblemen eine untergeordnete Rolle spielen.

2.2.1 Konfigurationsraum

Im physikalischen Raum $\mathcal{P} \in \mathbb{R}^3$ wird der Bereich, der vom Roboter in der Konfiguration $\mathbf{q}$ eingenommen wird, mit $\mathcal{A}(\mathbf{q})$ bezeichnet. Ist N die

Anzahl der Freiheitsgrade des Roboters, definiert die Konfiguration $\mathbf{q} \in \mathbb{R}^N$ den Zustand aller Freiheitsgrade. Hat dieser nur unabhängige, rotatorische Gelenke so entspricht $\mathbf{q}$ dem Gelenkwinkelvektor des Roboters. $\mathcal{O} \subset \mathcal{P}$ bezeichnet den Bereich, der von Hindernissen eingenommen wird.
Der Konfigurationsraum $\mathcal{C}$ ist definiert durch die Menge aller $\mathbf{q}$. Daher gilt $\mathcal{C} \subset \mathbb{R}^N$. Für die Pfadplanung wird $\mathcal{C}$ in zwei disjunkte Bereiche zerlegt. Für einen Roboter, bei dem keine Eigenkollisionen möglich sind, gilt:

$$\mathcal{C}_{occ} = \{\mathbf{q} \in \mathcal{C} | \mathcal{A}(\mathbf{q}) \cap \mathcal{O} \neq \emptyset\} \tag{2.1}$$

$$\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{occ} \tag{2.2}$$

Sind bei dem Roboter Eigenkollisionen möglich, so muss Gleichung 2.1 erweitert werden. Dafür sei $\mathcal{A}_i(\mathbf{q})$ der Bereich in $\mathcal{P}$, der vom Roboterteil i eingenommen wird und damit gilt:

$$\mathcal{A}(\mathbf{q}) = \bigcup_{i=1}^{M} \mathcal{A}_i(\mathbf{q}) \tag{2.3}$$

für einen Roboter mit M Teilen. Des Weiteren ist P die Menge aller Roboterteilpaare, für die gelten muss:

$$\mathcal{A}_i(\mathbf{q}) \cap \mathcal{A}_j(\mathbf{q}) = \emptyset \tag{2.4}$$

Dies gilt gewöhnlich für alle Roboterteilpaarungen außer denen, die durch ein Gelenk miteinander verbunden sind. Damit ergibt sich:

$$\mathcal{C}_{occ} = \{\mathbf{q} \in \mathcal{C} | \mathcal{A}(\mathbf{q}) \cap \mathcal{O} \neq \emptyset\} \bigcup \left(\bigcup_{[i,j] \in P} \{\mathbf{q} \in \mathcal{C} | \mathcal{A}_i\mathbf{q} \cap \mathcal{A}_j\mathbf{q} \neq \emptyset\} \right) \tag{2.5}$$

Ein Pfadplanungsproblem wird durch diese Definition darauf reduziert, einen Pfad für einen Punkt zu finden, welcher ausschließlich in $\mathcal{C}_{free}$ liegt. Durch die vielfältigen Kollisionsmöglichkeiten eines komplexen Roboters ist allerdings die Berechnung von $\mathcal{C}_{free}$ äußerst aufwändig [15].
Samplingbasierte Pfadplaner umgehen dieses Problem, indem sie $\mathcal{C}_{free}$ durch diskrete Konfigurationen annähern. Für viele praktische Pfadplanungsprobleme genügt eine relativ grobe Annäherung an $\mathcal{C}_{free}$, um einen Pfad zu finden. Dies macht samplingbasierte Pfadplanung für hochdimensionale $\mathcal{C}$ sehr effizient.
Um den Abstand zweier $\mathbf{q} \in \mathcal{C}$ bestimmen zu können, muss eine entsprechende Metrik definiert werden. Diese Abstandsberechnung sollte an die Bewegungskosten des Roboters angepasst sein. Als einfache Realisierung wird im Folgenden die L_1-Metrik, auch *Manhattan*-Metrik genannt, verwendet. Die unterschiedlichen Auswirkungen der verschiedenen Gelenke auf

den Roboter werden durch eine Gewichtung der Beträge der Gelenkwertdifferenzen berücksichtigt. Der Abstand zwischen zwei Konfigurationen ist über die Summe dieser Werte definiert. Da sich keine sinnvolle reale Einheit dieser Abstände oder Längen ergibt, wird im weiteren Verlauf der Arbeit der Ausdruck *Metrikeinheit* verwendet.

2.2.2 Samplingstrategien

Ein Sample ist eine Konfiguration, die auf Basis der Samplingstrategie ausgewählt wurde, um $\mathcal{C}$ anzunähern. Da die Anzahl der Samples $N_{Samples}$ abzählbar ist, $\mathcal{C}$ aber kontinuierlich ist, kann $\mathcal{C}$ nie exakt durch Samples dargestellt werden. Bei einer entsprechenden Wahl der Samplingstrategie kann allerdings erreicht werden, dass der Abstand zu jeder Konfiguration zu der Menge des Samples beliebig klein wird, wenn $N_{Samples} \rightarrow \infty$. Eine solche Menge wird als *dicht* bezeichnet und ermöglicht für samplingbasierte Pfadplaner mit *dichter* Samplingstrategie die Aussage zu treffen, dass für $N_{Samples} \rightarrow \infty$ die Wahrscheinlichkeit, einen existierenden Pfad zu finden gegen eins konvergiert. Für die praktische Anwendung von Pfadplanern ist dies aber nur von geringer Bedeutung, da $N_{Samples}$ immer endlich bleibt. Für die Praxis ist wünschenswert, dass $\mathcal{C}$ durch möglichst wenige Samples mit einer gewissen Auflösung, bzw. Verteilung abgedeckt wird. Im Idealfall lässt sich diese Auflösung dann durch weitere Samples verfeinern. Im Folgenden werden einige Ansätze für Samplingstrategien vorgestellt.

Gitterraster Ein Ansatz, eine gleichmäßige Abdeckung von $\mathcal{C}_{free}$ zu erreichen ist, Konfigurationen gitternetzförmig auszuwählen. Diese Methode hat allerdings zwei entscheidende Nachteile: Die Anzahl der Konfigurationen $N_{Samples}$ um ein Gitternetz zu erzeugen ist für einen Roboter mit vielen Freiheitsgraden auch bei einer sehr groben Rasterung extrem hoch, da gilt:

$$N_{Samples} = \text{Samples pro Achse}^{Dimension(\mathcal{C})} \tag{2.6}$$

Ist die Dimension von $\mathcal{C}$ beispielsweise 15, müssen bei einem Gitter von zwei Konfigurationen pro Achse bereits 32768 Konfigurationen erzeugt werden. Ein weiterer Nachteil ist die schlechte Skalierbarkeit des Gitterrastersamplings. Sind die Konfigurationen im Gitterraster zu weit voneinander entfernt, muss die Anzahl der Konfigurationen pro Achse um mindestens eins erhöht werden. Für obiges Beispiel würde sich damit eine Anzahl von 14348907 Konfigurationen ergeben.

Probabilistisch Eine *dichte* Menge von Samples lässt sich über eine Zufallsvariable erzeugen. Der einfachste und am meisten verbreitete Ansatz ist eine gleichverteilte Zufallsvariable für jeden Freiheitsgrad. Die Vorteile bei

diesem Ansatz liegen bei dem geringen Rechenaufwand und der guten Skalierbarkeit, da jederzeit neue zufällige Konfigurationen hinzugefügt werden können. Diese Samplingstrategie ist allerdings nicht deterministisch, d.h. auch unter identischen Planungsvoraussetzungen ist der ermittelte Pfad jedes Mal ein anderer. Auch die Planungsdauer kann sehr stark schwanken. Des Weiteren kann keine exakte Aussage über die Auflösung bei einer bestimmten Anzahl von Samples getroffen werden.

Deterministische Sequenzen Ein Beispiel für deterministische Sequenzen sind die *Halton-Sequenzen*. Die Eigenschaften dieses Ansatzes sind ähnlich wie die des probabilistischen, gleichverteilten Samplings. Allerdings sind die *Halton-Sequenzen* jedes Mal gleich. Daher werden bei identischen Planungsvoraussetzungen immer die gleichen Ergebnisse erzielt. Dies hat den Vorteil, dass die Planungsdauern nicht mehr schwanken. Allerdings kann die Planung bei einer ungünstigen Umgebungskonstellation länger dauern als der Erwartungswert beim zufälligen Sampling. Ein Vorteil ist, dass die Konfigurationen sicher gleichmäßig über den gesamten Konfigurationsraum verteilt sind. Bei den probabilistischen Verfahren ist dies nur wahrscheinlich.

Minimale Ausbreitung Bei dieser Samplingstrategie werden die Konfigurationen ausgewählt, die den größten Abstand zu allen anderen Samples haben. Dadurch wird der Raum sehr gleichmäßig abgedeckt und das größte freie Volumen minimal gehalten. Der Nachteil dieses Verfahrens ist der große Rechenaufwand, der für jedes neue Sample betrieben werden muss.

2.2.3 Lokaler Planer

Für die samplingbasierte Pfadplanung muss überprüft werden, ob zwei Konfigurationen miteinander verbunden werden können. Diese Planung wird von dem lokalen Planer durchgeführt, der damit eine wichtige Grundlage für die globalen Planer ist. Handelt es sich bei dem Robotersystem um einen nicht-holonomen Roboter, kann die Planung, wie ein Zustand mit dem anderen verbunden wird sehr komplex werden, da alle individuellen Beschränkungen des Roboters beachtet werden müssen. In der Regel führt das dazu, dass Trajektorien mit bestimmten Beschränkungen in $\mathcal{C}$ gelegt werden müssen.
Bei holonomen Robotern beschränkt sich die lokale Planung meist darauf, die direkte Verbindung zwischen den beiden Konfigurationen auf Kollisionen zu überprüfen. Auch dafür gibt es verschiedene Herangehensweisen, die zu unterschiedlich effizienten Lösungen führen [5].

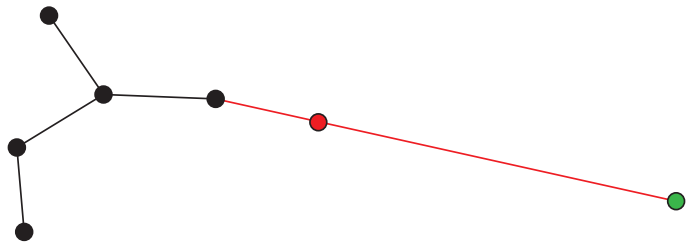


Abbildung 2.1: Erweiterungsstrategie *RDT-classic*

2.2.4 Rapidly-Exploring Dense Tree Verfahren

Die *Rapidly-Exploring Dense Tree* Verfahren, kurz *RDT* Verfahren, stellen eine Gruppe von globalen samplingbasierten Pfadplanern dar. Ursprünglich unter dem Namen *Rapidly-Exploring Radom Tree* veröffentlicht [14], sind diese Verfahren heute die am meisten verwendeten Algorithmen für *Single-Query* Probleme, also Pfadplanungsprobleme, bei denen ein Pfad für ein bestimmtes Start-/Zielkonfigurationspaar gesucht wird. Im Folgenden werden der ursprüngliche Algorithmus und einige wichtige Variationen vorgestellt. Gemeinsam haben all diese Varianten, dass sie, die richtige Samplingstrategie vorausgesetzt, siehe Abschnitt 2.2.2, $\mathcal{C}$ schnell mit relativ wenigen Konfigurationen abdecken und danach mit weiteren Konfigurationen den Raum immer dichter besetzen.

RDT-classic Der ursprüngliche *RDT* [14] lässt von einer Startkonfiguration einen Baum auf eine Zielkonfiguration *hinwachsen*. Dazu wird bei jeder Iteration eine Konfiguration durch Sampling erzeugt, in Abbildung 2.1 grün dargestellt. Diese wird aber nicht dem Baum hinzugefügt sondern es wird die dieser Konfiguration am nächsten gelegene Konfiguration aus dem Baum gesucht. Von diesem nächsten Nachbarn wird eine definierte Strecke in Richtung des gesampelten Zustandes auf Kollisionen überprüft. Ist diese Strecke kollisionsfrei, wird dem *RDT* ein neuer Zustand, in der Abbildung rot dargestellt, im überprüften Abstand auf der Verbindungslinie zum gesampleten Zustand hinzugefügt. Um den *RDT* bevorzugt in die Richtung der Zielkonfiguration wachsen zu lassen, kann mit einer definierbaren Quote anstelle eines gesampleten Zustandes die Zielkonfiguration verwendet werden, um den Baum zu erweitern.

RDT-visibility Für den *RDT-visibility*, dargestellt in Abbildung 2.2, wird ebenfalls in jeder Iteration eine Konfiguration gesampled. Wieder wird die

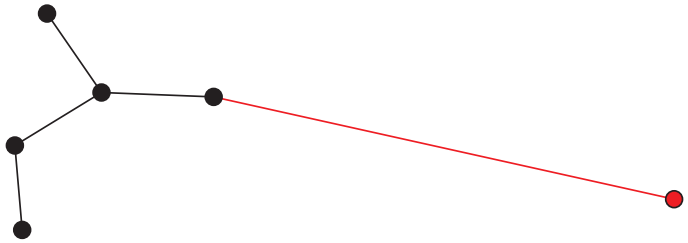


Abbildung 2.2: Erweiterungsstrategie *RDT-visibility*

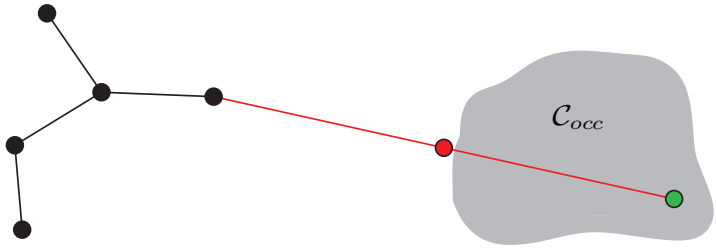


Abbildung 2.3: Erweiterungsstrategie *RDT-connect*

Konfiguration aus dem *RDT* gesucht, die dieser neuen Konfiguration am nächsten liegt. Bei dieser Variante wird aber kein Zustand auf der Verbindungslinie erstellt, sondern die Verbindungslinie wird vollständig auf Kollisionen überprüft. Ist die Verbindung möglich, wird die rot dargestellte Konfiguration dem Baum hinzugefügt, ansonsten wird sie verworfen und die nächste Iteration wird gestartet.

RDT-connect Der *RDT-connect* ist im Grunde eine Mischung aus den beiden obigen Varianten. Wie auch bei den anderen Verfahren wird mit der Samplingstrategie eine neue Konfiguration erzeugt und deren nächster Nachbar aus dem *RDT* gesucht. Bei dieser Variante wird allerdings, wie in Abbildung 2.3 dargestellt, die längste mögliche kollisionsfreie Strecke auf der Verbindungslinie von der nächsten Nachbarkonfiguration aus gesucht. Am Ende dieser Strecke wird eine neue Konfiguration erzeugt und diese dem *RDT* hinzugefügt. Ist die Strecke vollständig kollisionsfrei, verhält sich der *RDT-connect* wie der *RDT-visibility*. Für eine genauere Beschreibung kann die Publikation von Kuffner und LaValle [12] herangezogen werden.

Basierend auf diesen Verfahren gibt es weitere Möglichkeiten, die Pfadplanung effizienter zu gestalten. Pfadplanungsprobleme lassen sich in komplexer Umgebung meist schneller lösen, wenn sowohl von der Start- als auch von der Zielkonfiguration ein *RDT* gestartet wird [16]. Auch die angewand-

te Samplingstrategie, siehe Abschnitt 2.2.2, hat einen großen Einfluss auf die Planung. Um die Effizienz des Planers zu steigern, können die Samples, welche für die Planung in Frage kommen vorselektiert werden. Je nach Umgebung und Robotersystem können z.B. nur Konfigurationen zugelassen werden, die weit von Hindernissen entfernt sind oder aber Konfigurationen, die an $\mathcal{C}_{occ}$ grenzen. Prominentes Beispiel, enge Passagen in $\mathcal{C}$ zu finden ist das sogenannte *Bridge-Sampling*. Ausführliche Informationen zu diesem Thema können bei LaValle [15] gefunden werden.
Anhand von empirischen Untersuchungen, wie sie z.B. in einer Studienarbeit [5] durchgeführt wurden, lässt sich zeigen, dass die unterschiedlichen Variationen der *RDT*s, sowie die weiteren Einflussmöglichkeiten auf den Planer bei verschiedenen Robotersystemen und Umgebungen zu jeweils unterschiedlichen Ergebnissen führen. Keine der vorgestellten Varianten kann sich gegen die anderen immer durchsetzen. Tendenziell lässt sich sagen, dass die *RDT-classic* Variante sehr gut für enge, verschachtelte Konfigurationsräume geeignet ist, während die *RDT-visibility*s bei offenen Konfigurationsräumen effizienter sind. Das Anwendungsgebiet der *RDT-connect* Variante liegt zwischen diesen beiden Extremen.
Basis der *RDT*s ist die Konfiguration, von der aus der *RDT* erweitert wird. Da hier meist die Start- bzw. Zielkonfiguration verwendet wird, handelt es sich bei dieser Verfahrensgruppe um so genannte *Single-Query* Verfahren, also Planer, welche für ein bestimmtes Start-/Zielkonfigurationspaar einen Pfad finden. So können einzelne Planungsvorgänge effizient gelöst werden, sind aber mehrere Pfadanfragen in der gleichen Umgebung zu bearbeiten, wird bei den *Single-Query* Verfahren jeweils eine neue Planung gestartet. Die Information aus den vorherigen Planungen wird dabei nicht berücksichtigt. Daher sind für diese Problemklasse meist die *Multi-Query* Verfahren, wie in Abschnitt 2.2.5 beschrieben, besser geeignet.

2.2.5 Probabilistic-Roadmap Verfahren

Die Gruppe der *Roadmap* Verfahren ist ein häufig eingesetzter Vertreter globaler, samplingbasierter Planer für *Multi-Query* Anwendungen. Erstmals wurde das Prinzip der *Roadmap* [10] mit der *Probabilistic Roadmap PRM* vorgestellt.
Der grundsätzliche Ansatz dieser Verfahren ist, $\mathcal{C}_{free}$ durch ein Netz von Konfigurationen und Verbindungen darzustellen. Diese *Roadmap* soll die Eigenschaft haben, dass jede Konfiguration aus $\mathcal{C}_{free}$ mit ihr durch den lokalen Planer verbunden werden kann. Des Weiteren sollen alle Konfigurationen der *Roadmap* über die registrierten Verbindungen erreichbar sein. Ist dies aufgrund der Struktur von $\mathcal{C}_{free}$ nicht möglich, so muss die *Roadmap* in verschiedene Komponenten zerlegt werden, die diese Bedingung für alle Teilbereiche von $\mathcal{C}_{free}$ erfüllen.

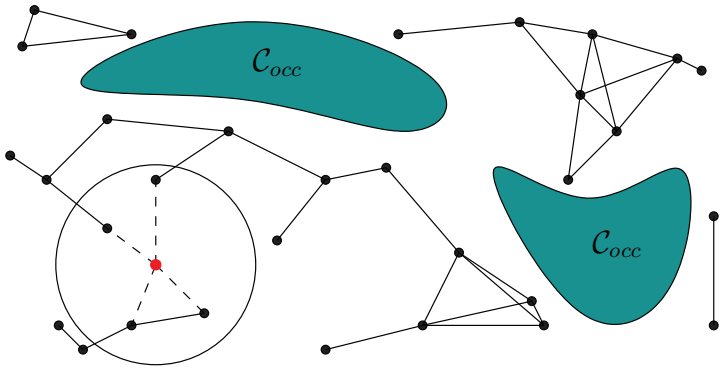


Abbildung 2.4: Erweiterungsstrategie PRM [15]

Wird die *Roadmap* diesen beiden Forderungen gerecht, reduziert sich das Pfadplanungsproblem nach dem Erstellen der *Roadmap* auf zwei einfache Schritte. Zunächst wird versucht, Start- und Zielkonfiguration zu einer Komponente der *Roadmap* hinzuzufügen. Ist dies nicht möglich, existiert kein Pfad in $\mathcal{C}_{free}$. Wenn die Ausgangskonfigurationen hinzugefügt werden konnten, wird der Pfad mit Hilfe eines Graphen-Such-Algorithmus wie z.B. dem *A** ermittelt. Ändern sich die Planungsbedingungen nicht, kann die *Roadmap* für jede beliebige Planungsanfrage benutzt werden. Es handelt sich bei den *Roadmap*-Verfahren daher um *Multi-Query* Verfahren.
Im Folgenden werden zwei oft angewandte *PRM*-Varianten vorgestellt.

PRM-classic Bei der ursprünglichen Variante der *Probabilistic Roadmaps* [10] werden mit der Samplingstrategie neue Konfigurationen erzeugt, in Abbildung 2.4 rot dargestellt. Für jedes Sample wird überprüft, ob Konfigurationen aus der bestehenden Roadmap innerhalb eines bestimmten Abstandes liegen. Ist dies der Fall, werden mit dem lokalen Planer die Verbindungen zu diesen überprüft. Wenn eine Verbindung kollisionsfrei möglich ist, wird die neue Konfiguration zu der verbundenen Komponente hinzugefügt und die Verbindung gespeichert. Wurde die Konfiguration bereits mit einer anderen Komponente verbunden, so werden die beiden Komponenten zu einer zusammengefasst. Ist es nicht möglich, die Konfiguration mit mindestens einer Komponente zu verbinden, wird diese zu einer neuen Komponente der Roadmap.

PRM-visibility Die zweite Variante der *PRM*s ist die *Visibility-based Probabilistc Roadmap* [17]. Auch hier werden zunächst Samples generiert, die zu einer bestehenden *Roadmap* hinzugefügt werden sollen. Bei dieser Variante werden jedoch alle Konfigurationen, die sich ausschließlich mit einer Komponente verbinden lassen, verworfen. Der Abstand, in dem nach Konfigurationen gesucht wird ist hierbei meist unbeschränkt.

Neben den grundsätzlichen Variationen der *PRM* Verfahren existieren noch eine Reihe an Parametern, die sich je nach Anwendungsfall sehr stark auf die Effizienz des jeweiligen Planers auswirken können.
Ein wichtiges Kriterium ist die Abbruchbedingung der *Roadmap* Erzeugung. Da bei *PRM* Verfahren kein direkt überprüfbares Merkmal für eine ausreichend detaillierte *Roadmap* existiert, besteht ein großer Spielraum, wann der Algorithmus zur *Roadmap* Erzeugung beendet wird. Ist die *Roadmap* zu groß, dauert die Planung unnötig lange. Wird zu früh abgebrochen, können die Pfadplanungsprobleme nicht gelöst werden. Der einfachste Ansatz ist, im Voraus die Anzahl der zu erzeugenden Konfigurationen zu definieren. Weitere Möglichkeiten sind, zu kontrollieren, wann die Anzahl der Komponenten sich durch neue Konfigurationen nicht mehr verändert oder eine feste Dauer für die Kartenerstellung zu definieren.
Mit weiteren Parametern lassen sich z.B. die Komponentenanzahl der *Roadmap* bei Beginn der Erstellung festlegen oder die Dichte der Vernetzung der *Roadmap* bestimmen. Eine genaue Beschreibung dieser Parameter und ihrer Auswirkungen sowie ein Vergleich der verschiedenen *PRM* Variationen kann in verschiedenen Arbeiten gefunden werden [5] [6] [8].

2.3 Exploration

Ein wichtiges Ziel der aktuellen Forschung im Bereich der Robotik ist es, Robotersystemen zu ermöglichen, in alltäglichen Umgebungen, wie z.B. in Haushalt oder Industrie, autonom zu agieren. Die Tatsache, dass die Umgebung oft gar nicht oder nur fehlerhaft und unvollständig bekannt ist stellt eine große Herausforderung dar. Es kann z.B. einem Roboter, der bei der Montage eines Werkstücks assistieren soll, zwar die exakte Position der Werkbank vorgegeben werden, wo aber das Werkzeug von dem Arbeiter abgelegt wird variiert von Fall zu Fall. Ebenso ist es denkbar, dass neue Hindernisse wie z.B. Kisten im Arbeitsbereich des Roboters abgestellt wurden.
Daher müssen Roboter, sollen sie in der Kleinindustrie oder im Haushalt eingesetzt werden, in teilweise unbekannten Umgebungen arbeiten können. Das benötigte Wissen über die Umgebung müssen diese Robotersysteme dafür autonom durch ihre Sensoren ermitteln.
Bei dieser sogenannten Exploration können je nach Szenario verschiedene Ziele angestrebt werden. Soll der Roboter an eine bestimmte Konfiguration fahren, ist es wichtig, den Konfigurationsraum $\mathcal{C}$, siehe Abschnitt 2.2.1, des Roboters zu ermitteln, wird hingegen ein Werkstück auf der Werkbank gesucht, ist eine Erkundung des physikalischen Raums $\mathcal{P}$ anzustreben. Damit der Roboter seine Aufgabe effizient erfüllen kann, sollten möglichst wenige

Bewegungen und Messungen ausgeführt werden. Um eine Explorationsstrategie zu entwickeln, die diesen Anforderungen gerecht wird, müssen verschiedene Probleme gelöst werden.
Die Bewegungen des Roboters werden in $\mathcal{C}$ geplant, die Sensoren arbeiten aber in $\mathcal{P}$. Soll ein bestimmter unbekannter Bereich in $\mathcal{C}$ exploriert werden, muss an den entsprechenden Stellen in $\mathcal{P}$ gescannt werden. Während die Bedeutung eines Punktes aus $\mathcal{C}$ für $\mathcal{P}$ über die Kinematik und die Geometrie des Roboters eindeutig definiert ist, stellt sich die umgekehrte Beziehung schwieriger dar. Ein Versuch, dennoch eine Verknüpfung herzustellen, ist die Einführung einer *C-space Entropy*, auf die in Abschnitt 2.3.1 eingegangen wird.
Im Folgenden werden die Schritte, die für eine Exploration durchgeführt werden müssen, vorgestellt. Zunächst wird eine für die Messung mit den Sensoren geeignete Konfiguration, im weiteren Verlauf der Arbeit *View-Point VP* genannt, durch die *View-Point* Planung ermittelt. Dieser Teil der Exploration wird in Abschnitt 2.3.2 behandelt. Danach wird der *VP* durch Einsatz des Pfadplaners angefahren und die Messung durchgeführt. Da die Sensoren jedoch mit Fehlern behaftet sind, können die Ergebnisse nicht direkt in das Umgebungsmodell der Exploration übertragen werden, sie werden zunächst durch Algorithmen interpretiert. Die darauf folgende Aktualisierung der Umgebung wird in Abschnitt 2.3.3 behandelt.

2.3.1 C-space Entropy

Um Aussagen über die Auswirkungen der Erkundungen von $\mathcal{P}$ auf $\mathcal{C}$ machen zu können, wurde in der Veröffentlichung von Yu et al. [27] der Begriff der *C-space-Entropy* entwickelt. Dabei wird sowohl die nicht explorierte Teilmenge von $\mathcal{P}$, im weiteren Verlauf mit $\mathcal{P}_{unk}$ bezeichnet, als auch die nicht erkundeten Bereiche von $\mathcal{C}$, im Folgenden durch $\mathcal{C}$ abgekürzt, als stochastischer Prozess betrachtet. Die *C-Space-Entropy* ist eine Entropie nach der Definition von Shannon [20].
Es wird angenommen, dass der stochastische Prozess von $\mathcal{P}_{unk}$ bekannt ist. Durch die Funktion $\mathcal{A}(\mathbf{q})$, definiert in Abschnitt 2.2.1, kann dieser auf $\mathcal{C}$ abgebildet werden. Die binäre Zufallsvariable Q der Konfiguration $\mathbf{q} \in \mathcal{C}_{unk}$ kann entweder 0 für $\mathbf{q} \in \mathcal{C}_{free}$ oder 1 für $\mathbf{q} \in \mathcal{C}_{occ}$ sein. Daraus folgt für n Konfigurationen die *C-space Entropy*:

$$H(\mathcal{C}) = -\sum_{Q_1=0,1} \cdots \sum_{Q_n=0,1} P(Q_1,\ldots,Q_n) \log P(Q_1,\ldots,Q_n) \tag{2.7}$$

Die Wahrscheinlichkeit $P(\ldots)$, dass eine Konfiguration mit Hindernissen kollidiert oder nicht, kann mit $\mathcal{A}(\mathbf{q})$ und der Wahrscheinlichkeitsverteilung von $\mathcal{P}_{unk}$ berechnet werden.
Um die Eigenschaften von $\mathcal{P}_{unk}$ durch eine Wahrscheinlichkeitsverteilung

auszudrücken, wird in der Publikation von Yu et al. [27] aus zwei Gründen ein *Poisson-Prozess* [21] verwendet. Zum einen ist er rechentechnisch effizient umzusetzen, zum anderen sinkt die Wahrscheinlichkeit, dass ein Bereich hindernisfrei ist mit seinem Volumen und repräsentiert damit eine elementare Eigenschaft der meisten realen $\mathcal{P}$.
Ein *Poisson-Prozess* entspricht einem zufälligen, gleichverteilten Setzen von Hindernispunkten im Raum. Damit ergibt sich für die Wahrscheinlichkeit $p(\mathcal{B})$, dass ein beliebiger Bereich $\mathcal{B} \subset \mathcal{P}$ Teilbereich von $\mathcal{P}_{free}$ ist, folgender Ausdruck:

$$p(\mathcal{B}) = e^{-\lambda \cdot vol(\mathcal{B})} \tag{2.8}$$

Mit dem Parameter $\lambda \in]0, \infty[$, der Intensität des *Poisson-Prozesses*, ist es möglich, die Dichte der Hindernisse in $\mathcal{P}_{unk}$ zu simulieren.
Basierend auf diesen Annahmen kann der Zusammenhang zwischen dem Wissen über $\mathcal{B}$ und der daraus resultierenden, zu erwartenden Reduktion der *C-space Entropy* $E(-\Delta H(\mathcal{C}))$, also dem Informationsgewinn in $\mathcal{C}$, hergestellt werden. Durch Grenzwertbildung lässt sich damit eine *Entropie-Reduktions-Dichte* $ERD_c(x)$ für einen Punkt $x \in \mathcal{P}$ ableiten. Mit der vereinfachenden Annahme, dass die Zufallsvariablen Q_i voneinander unabhängig sind und der Schlussfolgerung, dass nur die Konfigurationen $\mathcal{X}$ zur Entropiereduktion durch Messung des Punktes x beitragen können, für die gilt:

$$\mathcal{X}(x) = \{\mathbf{q} \in \mathcal{C}_{unk} : x \in \mathcal{A}(\mathbf{q})\} \tag{2.9}$$

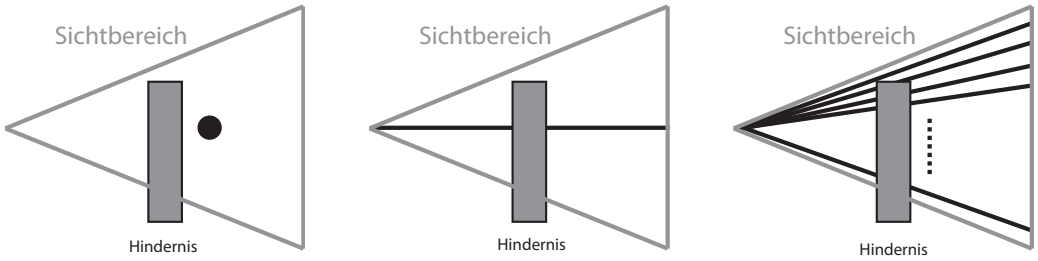
kann die Näherung für $ERD_c(x)$ wie folgt angegeben werden:

$$\widetilde{ERD}_c(x) = \sum_{q \in \mathcal{X}(x)} -\lambda \log(1 - e^{-\lambda \cdot vol(\mathcal{A}_{unk}(\mathbf{q}))}) \tag{2.10}$$

Eine ausführliche Herleitung der *C-space Entropy* findet sich in der Veröffentlichung von Yu und Gupta [28].

2.3.2 View-Point Planung

Für die Exploration muss der unbekannte Raum mit Sensoren des Roboters untersucht werden. Dabei ist zu beachten, dass die Auswertung der Sensoren oft einen erheblichen Rechenaufwand hervorrufen. Ebenso sollen die Bewegungen des Roboters meist so gering wie möglich gehalten werden. Ziel der Auswahl der Sensorlage $x_s \in \mathcal{P}$ ist daher, mit möglichst wenigen Messungen viel Information über die unbekannte Umgebung zu sammeln. Die Relevanz der Daten hängt dabei auch von dem Ziel der Exploration ab. Entweder soll eine bestimmte Konfiguration erreicht oder ein Bereich aus $\mathcal{P}$ erkundet werden.
Die *View-Point* Planung ermittelt nach diesen Kriterien den besten *VP* mit



(a) Punktmodell (b) Strahlmodell (c) Strahlenfächermodell

Abbildung 2.5: Sensormodelle zur *View-Point* Planung [24]

der Sensorlage x_s für den nächsten Scanvorgang, den *Next Best View-Point*, kurz *NBV*. Um Aussagen über die Qualität eines *VP*s machen zu können, muss berechnet werden, über welchen Bereich $\mathcal{V}(x_s) \subset \mathcal{P}$, durch eine Messung an der Stelle x_s Informationen ermittelt werden können. Dafür muss ein Modell der verwendeten Sensoren benutzt werden. Um eine effiziente Berechnung zu ermöglichen, werden hier sehr starke Vereinfachungen verwendet, dargestellt in Abbildung 2.5.
Der Messbereich der Sensoren kann durch einen Punkt in der Mitte des Sichtfeldes angenähert werden. Hindernisse, die vor oder hinter diesem Punkt liegen, werden dabei nicht erfasst. Durch eine Modellierung des Sensorbereichs durch einen Strahl mit der Länge der maximalen Messdistanz können Hindernisse auf einer Linie detektiert und somit Verdeckungen im Sensorbereich registriert werden. Das realistischste Modell wird durch eine Menge von Strahlen erreicht, die den Messbereich vollständig, falls nicht durch Hindernisse verdeckt, ausfüllen kann.
Eine Möglichkeit, den *NBV* auszuwählen ist, die Sensorlage zu finden, in welcher das größte Volumen von $\mathcal{P}_{unk}$ erfasst wird [11]:

$$x_{NBV} = \underset{x_s}{\operatorname{argmax}}\, \mathcal{V}(x_s) \cap \mathcal{P}_{unk} \tag{2.11}$$

Dieses Verfahren wird mit *Maximal Physical Volume*, kurz *MPV* bezeichnet. Es wird das x_s gesucht, dessen $\mathcal{V}(x_s)$ den größten Bereich von $\mathcal{P}_{unk}$ abdeckt. Damit wird aber nicht berücksichtigt, wie sich die Messung auf $\mathcal{C}$ auswirkt. Im zweiten Ansatz wird, im Gegensatz dazu, versucht, durch die Wahl des Punktes x_{NBV} die maximale Informationsmenge über $\mathcal{C}$ zu erhalten. Dies entspricht nach der Definition einer maximalen Reduktion der *C-space Entropy*, siehe Abschnitt 2.3.1. Es wird der Punkt gesucht, für den die höchste Abnahme der Entropie durch eine Messung erwartet wird. Dieses Verfahren wird *Maximal Entropy Reduction*, kurz *MER* genannt [28] :

$$x_{NBV} = \underset{x_s}{\operatorname{argmax}}\, \widetilde{ER}_{x_s} \tag{2.12}$$

Mit diesen Kriterien und den Sensormodellen ist es möglich, eine qualitativ hochwertige Sensorlage zu bestimmen. Für das *MPV* Kriterium ist das realistischste Sensormodell geeignet. Beim *MER* wird aufgrund des höheren Rechenaufwands in der Regel das Punkt- oder Strahlmodell verwendet. Eine explizite Beschreibung dieser Ansätze kann in der Publikation von Suppa [24] gefunden werden.
Eine Möglichkeit, die Auswahl des *NBV* zu verbessern ist, bereits bei der *View-Point* Planung die Messungenauigkeiten und Messfehler zu berücksichtigen [23]. Nach der Publikation von Suppa [24] kann die Performanz der Exploration dadurch verbessert werden, da die Sicherheitsbereiche um die Hindernisse reduziert werden können.
Die Bewertung der *VP*s ist relativ aufwändig, deshalb können nicht alle $x_s \in \mathcal{P}$ überprüft werden. Die Exploration kann maßgeblich über die Wahl der *VP* Kandidaten beeinflusst werden. Gegenüber der zufälligen Auswahl der Punkte kann die *View-Point* Planung durch die Wahl von gut erreichbaren Punkten oder solchen in der Nähe des unbekannten Bereichs verbessert werden.

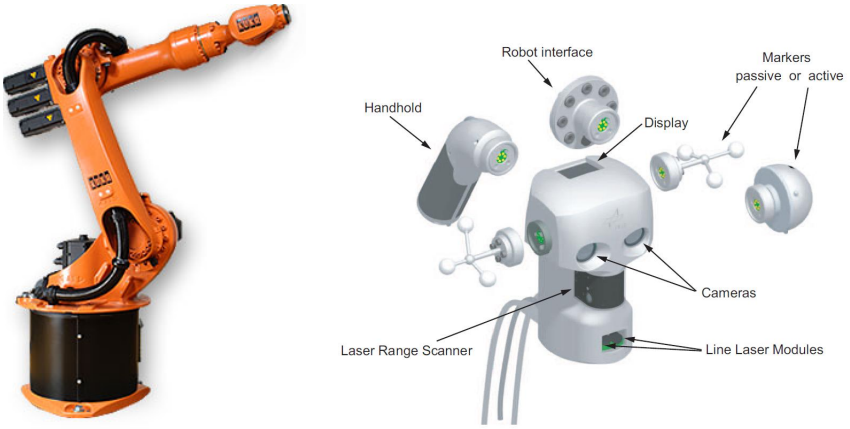
2.3.3 Aktualisierung von $\mathcal{P}$

Aus den durch die Sensoren aufgenommenen Informationen muss eine Repräsentation der Umgebung für den Roboter erzeugt werden. Im idealisierten Fall, der in dieser Arbeit angenommen wird, können dabei die registrierten Hindernisse an der gemessenen Position modelliert werden. Wird kein Hindernis gemessen, so gilt dieser Bereich als frei.
Da die Sensoren mit Ungenauigkeiten und Fehlern behaftet sind, werden die Daten in der realen Exploration zunächst mit Hilfe von stochastischen Verfahren auf ihre Aussagekraft hin überprüft. Dabei werden die Eigenschaften der Sensoren, wie z.B. Probleme bei der Detektion von spiegelnden oder schwarzen Oberflächen oder die mit der Messdistanz steigende Messungenauigkeit berücksichtigt.
Für die Integration von unsicheren Sensordaten in ein Umgebungsmodell, welches sichere Bewegungen des Roboters ermöglichen muss, stehen drei verschiedene Ansätze zur Verfügung. Die *Bayes' Rule*, das *Belief Update* und die *Fuzzy Fusion*. Eine Beschreibung und Evaluierung dieser Verfahren findet sich in der Veröffentlichung von Suppa [24].

Kapitel 3

Hand-Auge-System

Um den Einsatzbereich von Robotern erweitern zu können, müssen diese flexibel mit ihrer Umgebung interagieren können. Anders als in einer industriellen Fertigungstraße kann in einer kleinen Werkstatt oder im Haushalt nicht davon ausgegangen werden, dass die Umgebung immer bekannt ist. Es müssen daher Methoden entwickelt werden, die dem Robotersystem ermöglichen, eine teilweise unbekannte Umgebung zu erfassen. Diese Explorationsverfahren nutzen dafür über Sensoren erfasste Daten, um eine Repräsentation der Umgebung zu erhalten. Dabei können verschiedene Schwerpunkte gesetzt werden, wie z.B. möglichst viel physikalischen Raum zu explorieren, den Bereich, in dem sich der Roboter bewegen kann, zu vergrößern oder eine bestimmte Zielregion zu erforschen.
Maßgeblich zu der Auswahl der Explorationsmethoden trägt dabei der verwendete Roboter bei. In diesem Kapitel wird ein sogenanntes Hand-Auge-System eingesetzt. Dabei handelt es sich um einen stationären Industrieroboter, an dessen Endeffektor eine Sensoreinheit montiert ist. Die möglichen Scanlagen sind durch ihre Erreichbarkeit im jeweils aktuell bekannten Raum limitiert. Die Pfadplanung muss daher bei der Planung des *NBV* miteinbezogen werden. Außerdem müssen die gewählten Sensorlagen effizient angefahren werden können.
In diesem Kapitel soll eine Pfadplanung für ein Hand-Auge-System entwickelt werden, welche für den Einsatz in einem Explorationsszenario optimiert ist. Dafür werden in Abschnitt 3.1 zunächst die der Pfadplanung zugrunde liegenden Voraussetzung dargestellt. In Abschnitt 3.2 werden die in dieser Arbeit entwickelten Methoden für eine Pfadplanung in einem Explorationsszenario beschrieben und anhand von Simulationen evaluiert.



(a) Kuka KR16 [13] (b) DLR 3D-Modellierer [22]

Abbildung 3.1: Hardware des Hand-Auge-Systems

3.1 Voraussetzungen für die explorationsorientierte Pfadplanung

In diesem Abschnitt werden die Komponenten, welche die Rahmenbedingungen für die Pfadplanung in dieser Anwendung bestimmen, beschrieben. Dazu ist die in Abschnitt 3.1.1 vorgestellte Hardware mit Roboterarm und Sensoreinheit zu zählen, ebenso das für die Pfadplanung entwickelte Planungsmodell des Robotersystems, auf welches in Abschnitt 3.1.2 eingegangen wird. Abschnitt 3.1.3 beschreibt das verwendete Szenario und in Abschnitt 3.1.4 wird die Explorationsoftware, in welcher die Pfadplanung eingesetzt wird vorgestellt.

3.1.1 Hardware

Die Hardware des eingesetzten Hand-Auge-Systems besteht aus zwei Komponenten. Bei dem Arm des Robotersystems handelt es sich um einen Industrieroboter von Kuka. Als Sensoreinheit kommt der am DLR entwickelte 3D-Modellierer zum Einsatz. Dieser ist am Endeffektor des Kuka-Roboters montiert.

Kuka KR16 Der in Abbildung 3.1a dargestellte Kuka KR16 ist ein Industrieroboter mit sechs rotatorischen Freiheitsgraden. Mit einer maximalen Reichweite von 1,6m in radialer Richtung, sowie 2,0m in der Höhe, kann der Roboter eine Arbeitszelle mit den Maßen 2m$\times$2m$\times$2m abdecken. Die Kinematik des Roboters ermöglicht, viele Endeffektorpositionen in unterschiedlichen Orientierungen anzufahren.

DLR 3D-Modellierer Die Sensoreinheit des Systems ist der am DLR entwickelte 3D-Modellierer [22]. Diese Einheit enthält drei verschiedene Sensortypen zur Entfernungsmessung. Als passive Methode wird ein Stereokamerasystem verwendet. Dieses ist für größere Distanzen geeignet und kann Bereiche des Raums bis 2m Entfernung erfassen. Für den mittleren Entfernungsbereich, von 15 bis 50cm, wurde ein Lasersensor, der mit Hilfe des Lichtschnittverfahrens arbeitet, integriert. Für präzise Messungen im Nahbereich kommt ein auf Triangulation basierender Laserentferungsmesser zum Einsatz. In Abbildung 3.1b ist die Sensoreinheit abgebildet.

3.1.2 Pfadplanungsmodell des Hand-Auge-Systems

Für die Pfadplanung mit *SamPP* muss ein Modell des verwendeten Roboters erstellt werden. Da die Sensoreinheit aus einem starren Körper besteht, geht diese nur mit dem vorhandenen 3D-Modell in die Geometrie des Roboters ein, hat aber keinen weiteren Einfluss auf die Kinematik des Systems. Die Geometrie des KR16 ist aus den vorhandenen VRML-Dateien entnommen. Die Kinematik des Roboters ist mit den aus dem Datenblatt des Roboters ermittelten DH-Parametern für die Planung beschrieben worden. Als weiteres Roboterteil wurde der 3D-Modellierer an den Endeffektor des KR16 angebracht.
Für eine effiziente Planung wurden die Gelenkgewichte im Pfadplanungsmodell nach der maximal möglichen Armlänge senkrecht zur Drehachse festgelegt. Dadurch wird sichergestellt, dass die Bewegung von einer Konfiguration in eine die Auflösungslänge der Pfadplanung entfernte Nachbarkonfiguration an allen Punkten des Roboters unter einem bestimmten Wert bleibt. Danach kann der einzuhaltende Sicherheitsabstand von Hindernissen ermittelt werden.

3.1.3 Szenario

Das Testszenario für das Hand-Auge-System ist eine 2m$\times$2m$\times$2m große Zelle, in welcher der Roboter montiert ist. Die Gegenstände in dieser Zelle sind variabel. Ebenso ändert sich der zu Beginn der Planung bekannte Raum.
Da die Bewegung des Roboters ausschließlich in sicher freien Bereichen stattfinden darf, sind die unbekannten Bereiche für die Pfadplanung als Hindernisse zu betrachten. Nach jeder Messung vergrößert sich die bekannte Umgebung und damit die Hinderniskonstellation. Um $\mathcal{P}_{unk}$ darzustellen, wird der physkalische Raum daher in Würfel zerlegt, die, wenn unbekannt oder besetzt als Hindernis in die Pfadplanung integriert werden.
Ziel dieses Szenarios kann die vollständige Erkundung der Arbeitszelle sein.

Denkbar sind aber auch Aufgaben, in denen nur eine bestimmte Region exploriert werden soll oder auch nur eine Konfiguration im unbekannten Bereich angefahren werden soll.

3.1.4 Software SBP-Simulator

Für die Explorationsszenarien wird eine bestehende Software, der *Sensor-based Motion Planning Simulator*, kurz *SBP*-Simulator [24], mit der Pfadplanungssoftware *SamPP* kombiniert.
Der SBP-Simulator stellt dabei die Funktionen, welche die Exploration betreffen. Zentraler Punkt hierbei ist die Simulation der Sensoren, sowie die Aktualisierung der Umgebung auf Basis dieser Messungen. Hier kommen die in Kapitel 2.3 beschriebenen Ansätze zum Einsatz. In dieser Arbeit wurde eine Schnittstelle zu Übertragung der internen Darstellung der Umgebung an *SamPP* implementiert. Dabei werden alle Regionen, die nicht sicher frei sind als Hindernis deklariert.
Auch die Planung des *NBV* wird durch den *SBP*-Simulator unter Berücksichtigung des Explorationsziels durchgeführt. Dabei wird die *C-space Entropy* als Bewertungskriterium für die möglichen *VP*s verwendet.
Des Weiteren ermöglicht der *SBP*-Simulator, einen realen Roboter zu steuern und reale Sensordaten zu integrieren. Da das Thema dieser Arbeit die Pfadplanung ist, die in jedem Fall auf den durch die Explorationssoftware berechneten Daten arbeitet, werden die entwickelten Methoden ausschließlich in der Simulation überprüft.

3.2 Methoden und Ergebnisse der explorationsorientierten Pfadplanung

In diesem Kapitel werden die in dieser Arbeit entwickelten Methoden vorgestellt, in denen samplingbasierte Pfadplanung für Explorationsaufgaben eingesetzt wird. Neben der reinen Methodik werden die Verfahren in verschiedenen Szenarien eingesetzt und anhand ihrer Ergebnisse evaluiert.
In Abschnitt 3.2.1 wird ein auf Trajektorien basierender Ansatz zur Exploration entwickelt. Methoden zur Erzeugung von *VP*s werden in Abschnitt 3.2.2 untersucht.
Die Simulationen, die in diesen Abschnitt untersucht werden sind auf den in Tabelle 3.1 aufgeführten Rechnern durchgeführt worden. In der Regel werden für jede Planung 20 Simulationen durchgeführt, deren Ergebnisse als Werte einer gleichverteilten Zufallsvariable X betrachtet werden.

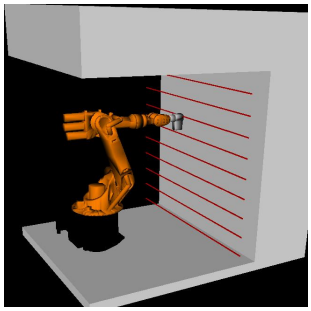
Tabelle 3.1: Daten der eingesetzten Simulationsrechner für das Hand-Auge-System

reine Pfadplanungssimulationen	
Prozessor	Intel Core 2 Duo P9600 2,66 GHz
Arbeitsspeicher	4 GB RAM
Betriebssystem	Microsoft Windows Vista Business
Simulationen mit dem SBP-Simulator	
Prozessor	Intel Pentium D 3,0 GHz
Arbeitsspeicher	3 GB RAM
Betriebssystem	openSUSE 10.2

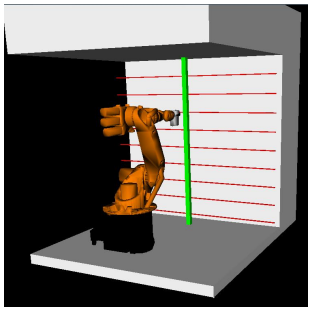
3.2.1 $\mathcal{P}$-Exploration durch Scantrajektorien

Für eine vollständige Erkundung von $\mathcal{P}$ muss jeder Punkt sicher erfasst werden. Damit hängt die Exploration nicht in erster Linie von einer möglichst guten Auswahl des VP in Abhängigkeit des aktuell bekannten $\mathcal{C}$ ab sondern wird hauptsächlich von dem zu erkundenden physikalischen Bereich definiert. Da das betrachtete Robotersystem nicht mobil ist, verändert sich $\mathcal{P}$ nur durch Hindernisse, die im Raum platziert werden, nicht aber in der grundsätzlichen Struktur, also der Ausdehnung des Raums oder der generellen Erreichbarkeit bestimmter Regionen.
Der Ansatz, der im Folgenden umgesetzt wird, ist, von der konkreten Umgebung unabhängige Trajektorien zu verwenden, durch die $\mathcal{P}$ systematisch exploriert werden kann. Die Erzeugung von geeigneten Trajektorien muss auf Basis der Explorationsstrategie erfolgen und ist damit nicht Thema dieser Arbeit. Es wird im weiteren Verlauf davon ausgegangen, dass eine Trajektorie vorliegt, welche die Sensoreinheit abfahren soll.
Dabei stellt sich zunächst das Problem, dass die Pfadplanung in $\mathcal{C}$ durchgeführt wird, die Trajektorie aber in $\mathcal{P}$ definiert wird. Des Weiteren ist für die Exploration nicht davon auszugehen, dass die vordefinierten Trajektorien vollständig im freien Raum von $\mathcal{P}$ liegen. Es muss daher eine Strategie entwickelt werden, mit Trajektorien umzugehen, die durch Hindernisse teilweise blockiert sind.

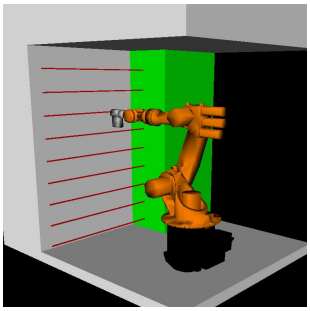
Methodik Für die Umsetzung einer Trajektorienplanung mit der Pfadplanung *SamPP* wurde die in $\mathcal{P}$ gegebene Trajektorie diskretisiert, d.h. in endlich viele Zwischenpositionen zerlegt. Um eine Planung zu ermöglichen, werden diese Lagen aus $\mathcal{P}$ mit Hilfe einer inversen Kinematik in die ent-



(a) Arbeitszelle des KR16 mit eingezeichneten Trajektorien



(b) Arbeitszelle mit Säule als Hindernis



(c) Arbeitszelle rechter Bereich blockiert

Abbildung 3.2: Umgebungen zur Evaluierung der Scantrajektorien-Methode

sprechenden Konfigurationen aus $\mathcal{C}$ umgerechnet. Um einen größeren Bereich von $\mathcal{P}$ erreichen zu können, wird durch die Trajektorie nur die Position des Endeffektors, sowie eine parallele Ausrichtung des letzten Roboterarmteils bezüglich der Bodenplatte definiert. Werden die verbleibenden Freiheitsgrade nicht durch die Kinematik beschränkt, geschieht dies durch Auswahl einer beliebigen Konfiguration.
Um die Trajektorie zu planen wird mit *SamPP* eine Pfadplanung von jeder Konfiguration zu der jeweils Nachfolgenden durchgeführt. Da die Pfadplanung in $\mathcal{C}$ stattfindet, wird auch die kürzeste Strecke in $\mathcal{C}$ gesucht. Die Diskretisierung der Trajektorie muss daher so fein durchgeführt werden, dass der Unterschied zwischen der kürzesten Strecke in $\mathcal{C}$ und der in $\mathcal{P}$ innerhalb der Toleranz zum Abfahren der Trajektorie liegt.
Liegen berechnete Konfigurationen innerhalb von $\mathcal{C}_{occ}$ werden diese in der Planung nicht berücksichtigt. Um die Zielkonfiguration für den jeweiligen Planungschritt zu erhalten, wird jede nachfolgende Konfiguration überprüft bis eine Kollisionsfreie gefunden ist. Hindernisse, die in der Trajektorie liegen, werden so auf einem möglichst kurzen Pfad in $\mathcal{C}$ umfahren.
Da die Trajektorien kontinuierlich angelegt werden, also nicht bestimmte Konfigurationen als *NBV* ermittelt werden, ist es sinnvoll, auch die Sensoren kontinuierlich zu verwenden.

Szenario In Abbildung 3.2a ist die Arbeitzelle des KR16-Roboters dargestellt. Die roten Balken im Bild markieren die Trajektorie die abgefahren werden soll um den dahinter liegenden, unbekannten Bereich, dargestellt durch eine graue Wand, zu erfassen. Die Linien werden dabei von oben nach unten in alternierender Richtung abgefahren, sodass eine mäanderformige Trajektorie entsteht. Die erste Linie liegt auf einer Höhe von 1,8m,

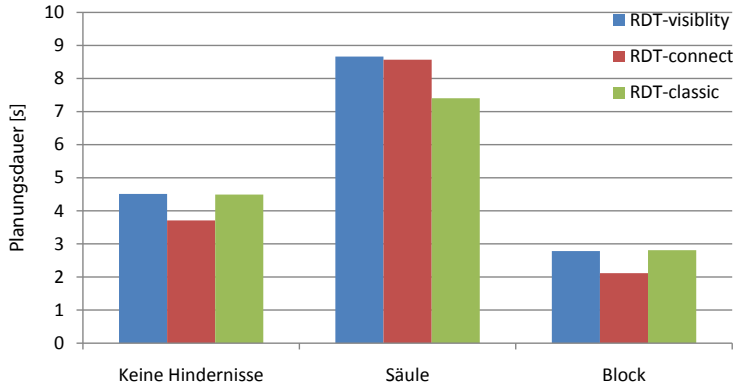


Abbildung 3.3: Erwartungswert der Planungszeiten für die Trajektorienplanung

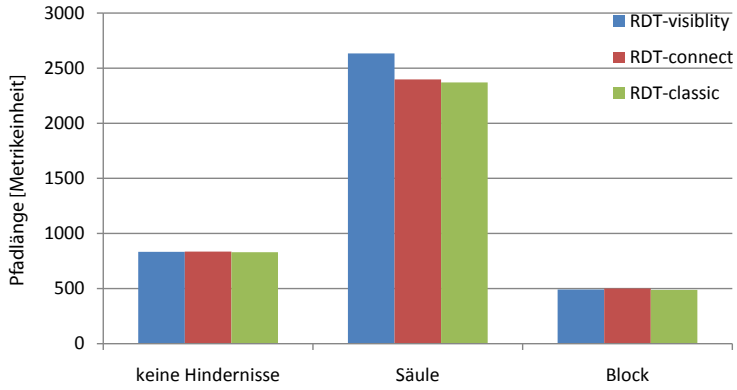


Abbildung 3.4: Erwartungswert der Pfadlängen für die Trajektorienplanung

die letzte auf 0,2m. Mit einer Teillänge von 2m deckt die Trajektorie eine vollständige Seite der Arbeitszelle ab. Zueinander haben die Balken einen Abstand von 0,2m. Für die Berechnung der entsprechenden Konfigurationen wurde jeder Balken durch 17 Lagen diskretisiert. Insgesamt wird die Trajetorie damit einschließlich der Start- und der Zielkonfiguration durch 155 Konfigurationen beschrieben.
Die Pfadplanung wird für den Fall, dass Hindernisse die Trajektorie teilweise blockieren, wie in Abbildungen 3.2b und 3.2c dargestellt untersucht. Die Trajektorie wird in dabei mit einer Säule, bzw. einem großen Block unterbrochen. Die im Bild grün markierte Säule blockiert dabei einige wenige, schmale Bereiche der Trajektorie, während der ebenfalls grüne Block die gesamte rechte Hälfte der Wand abdeckt.

Ergebnisse In Diagramm 3.3 sind die aus jeweils 20 Planungsdurchläufen gewonnenen Erwartungswerte der Planungsdauer für die Trajektorie mit den verschiedenen Hinderniskonstellationen aufgeführt. Als Verfahren ka-

men die drei Varianten der *RDT* zum Einsatz.
Für die Planung in der Umgebung ohne Hindernis ist die *RDT-connect* Variante mit einem Erwartungswert von unter vier Sekunden die schnellste. Die beiden anderen Verfahren benötigen im Durchschnitt etwa eine Sekunde länger.
In der für die Pfadplanung anspruchvollsten Umgebung, mit vor dem Roboter platzierter Säule, ist das *RDT-classic* Verfahren die beste Wahl.
Die Planungsdauer in der dritten Umgebung ist bei allen Varianten kürzer als in der Umgebung ohne Hindernisse. Dies ist dadurch zu erklären, dass in dieser Umgebung ein großer Teil der anzufahrenden Konfigurationen bei Beginn der Planung ausgeschlossen werden kann und die verbleibenden Konfigurationen gut miteinander verbunden werden können. Den niedrigsten Erwartungswert erzielt hier, wie schon in der ersten Umgebung, das *RDT-connect* Verfahren.
Neben der benötigten Planungzeit ist die Qualität des ermittelten Pfades für eine Scantrajektorie entscheidend. In Tabelle 3.4 sind die mit den verschiedenen Verfahren ermittelten Erwartungswerte der Trajektorienlängen dargestellt. Die Pfadlänge wird in den über die Metrik ermittelten insgesamten Bewegungskosten angegeben. Je kürzer der gefundene Pfad, desto effizienter werden die vorgegebenen Konfigurationen angefahren.
Bei der ersten und der letzten Umgebung zeigt sich, dass alle Verfahren identische Ergebnisse liefern. Das liegt an der Tatsache, dass in diesen beiden Hinderniskonstellationen die anzufahrenden Konfigurationen durch direkte Verbindungen in $\mathcal{C}$ verknüpft werden können. Ebenfalls lässt sich anhand dieser Daten ablesen, dass der Block in der letzten Umgebung die zu fahrende Trajektorie erheblich verkürzt.
In der Umgebung mit Säule sind die Ergebnisse unterschiedlich, da ein Pfad um das Hindernis gefunden werden muss. Den kleinsten Erwartungswert für die Pfadlänge hat dabei das *RDT-classic* Verfahren.

Evaluierung im SBP-Simulator Um die Exploration mit einer Scantrajektorie bewerten zu können, wurde eine Simulation mit dem *SBP*-Simulator erstellt. Als Umgebung kam dabei ein unbekannter Bereich, der mehrere Hindernisse enthält zum Einsatz. Da der *SBP*-Simulator keine Scanvorgänge während der Bewegung des Roboters unterstützt, wurde an allen Quantisierungskonfigurationen eine Messung aufgenommen und die dazwischen liegende Bewegung mit abgeschalteter Sensoreinheit zurückgelegt. In der untersuchten Umgebung konnte durch die Messungen entlang der Trajektorie dennoch der unbekannte Bereich auf der untersuchten Seite vollständig erkundet werden.
In der Bilderserie 3.5 sind die Explorationsfortschritte der untersuchten Trajektorienplanung abgebildet. Graue Bereiche markieren $\mathcal{P}_{unk}$, die schwarzen Bereiche stehen für ermittelte Hindernisse.

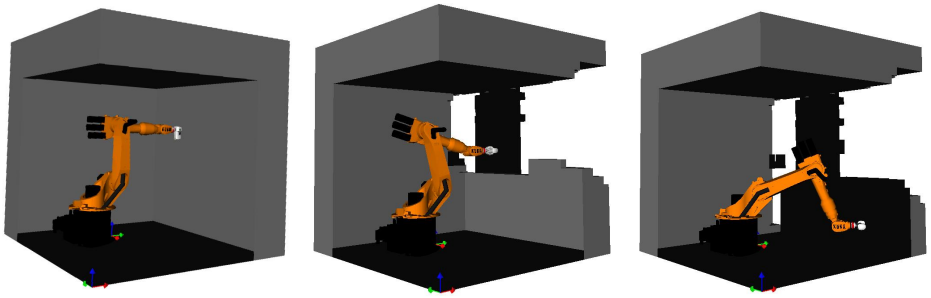


Abbildung 3.5: Explorationsfortschritte bei Einsatz einer Trajektorienplanung

3.2.2 Erzeugen von View-Point Kandidaten

Die Erkundung der Umgebung ist mit dem Hand-Auge-System nur dann möglich, wenn der Roboter sich zu dem entsprechenden *VP* bewegen kann. Da der Roboter nur in bekannten Regionen bewegt werden darf, ist in einer zu Beginn der Planung sehr wenig bekannten Umgebung das erste Ziel, die Bewegungsmöglichkeiten des Roboters zu erhöhen.
Dies kann durch eine Erkundung des Konfigurationsraums erreicht werden, denn hier findet die Pfadplanung statt. Die Sensoren des Roboters arbeiten allerdings im physikalischen Raum. Um eine Beziehung zwischen den Bereichen in $\mathcal{C}$ und $\mathcal{P}$ herzustellen wurde die in Abschnitt 2.3.1 vorgestellte *C-space Entropy* eingeführt. Basierend auf diesem Kriterium kann die Auswirkung der Exploration eines physikalischen Bereiches auf den Konfigurationsraum abgeschätzt werden.
Im Idealfall wird immer die Region untersucht, für welche der größte Informationsgewinn *IG* für den Konfigurationsraum erwartet wird. Um diesen Bereich und den entsprechenden *NBV* zu finden, muss eine große Menge an Konfigurationen hinsichtlich des zu erwartenden *IG* untersucht werden. Da dies rechentechnisch sehr aufwändig ist, kann nicht jede Konfiguration überprüft werden sondern es muss eine Vorauswahl für den *VP*, im Weiteren als *VP* Kandidat bezeichnet, getroffen werden.
In diesem Abschnitt werden einige Methoden entwickelt, um Konfigurationen mit potenziell hohem *IG* durch die Pfadplanung zu ermitteln.

Methodik und Umsetzung Für die *VP*s gibt es zwei Anforderungen, die erfüllt werden müssen. Die erste ist, dass die Konfiguration im bekannten Bereich von $\mathcal{C}$ liegt und auch vom Roboter in der aktuell bekannten Umgebung erreicht werden kann. Dies kann durch den Einsatz der Pfadplanung gewährleistet werden.
Die zweite Anforderung an den *VP* ist, dass in der daraus resultierenden

Sensorlage eine möglichst große Reduktion der *C-space Entropy* erreicht wird. Da der Pfadplanung jedoch keine Informationen über diesen Wert vorliegen und auch die Sensoren nicht durch die Pfadplanung simuliert werden, müssen verschiedene Methoden entwickelt werden, welche die Wahrscheinlichkeit erhöhen, dass eine gute Sensorlage vorliegt, ohne diese auswerten zu können.
Der einfachste Ansatz *M0*, mögliche *VP*s zu finden ist, zufällige Konfigurationen aus $\mathcal{C}$ zu ziehen und sie auf Kollisionen und Erreichbarkeit zu untersuchen. Auch wenn dieses Vorgehen den *IG* nicht beachtet, reduziert sich der Aufwand der anschließenden Berechnung des *NBV*, da nur für die in der aktuellen Umgebung erreichbaren Konfigurationen gerechnet werden muss. Die Erreichbarkeit wird in diesem Fall durch einen einfachen Verbindungstest zu den bekannten *VP*s durchgeführt.
Ein weitergehender Ansatz *M1* ist, Konfigurationen auf der Grenze zwischen $\mathcal{C}_{free}$ und $\mathcal{C}_{unk}$ zu suchen. Diese Methode basiert auf der Annahme, dass, wenn der Roboter sich im Konfigurationsraum in der Nähe des unbekannten Bereichs befindet, die Wahrscheinlichkeit steigt, mit den Sensoren unbekannte Regionen zu erfassen und somit den *IG* zu vergrößern. Auch hier müssen die entsprechenden Konfigurationen auf Erreichbarkeit überprüft werden. Zur Umsetzung dieser Schritte wird daher eine Konfiguration aus $\mathcal{C}_{unk}$ gesampled, zu welcher der nächste bekannte *VP* ermittelt wird. Von diesem aus wird auf der Verbindungslinie zwischen diesen beiden Konfigurationen die letzte in $\mathcal{C}_{free}$ liegende gesucht. Diese befindet sich auf dem Rand von $\mathcal{C}_{free}$ und ist erreichbar.
Bei dem beschriebenen Ansatz wird eine Konfiguration in der Nähe von $\mathcal{C}_{unk}$ gesucht, dabei wird aber nicht berücksichtigt, mit welcher Roboterstelle in den unbekannten physikalischen Raum eingedrungen wird. Handelt es sich dabei nicht um den Sensorkopf des Roboters, ist es unwahrscheinlich, dass eine Messung mit hohem *IG* durchgeführt werden kann. Für eine bessere Vorauswahl des *VP*s in Ansatz *M2* sollte daher die Konfiguration in der Nähe einer unbekannten Region in $\mathcal{C}$ liegen, die durch die Kollision des Sensorkopfes mit dem unbekannten physikalischen Raum entsteht, während der Roboterarm vollständig im bekannten freien Bereich liegt. In der Pfadplanung wurden daher für den Sensorkopf und den Roboterarm unterschiedliche Kollisionsklassen erzeugt. Von der Konfiguration wird wie auch schon in den vorherigen Ansätzen der nächste bekannte *VP* gesucht, von dem aus auf der Verbindungslinie die letzte kollisionsfreie Konfiguration als *VP*-Kandidat gewählt wird.
Speziell auf die Anwendung des Hand-Auge-Systems im *SBP*-Simulator lässt sich ein weiterer Ansatz *M3* entwickeln. Der Scanvorgang wird bei der Exploration nicht bei statischem Roboter durchgeführt, sondern die Sensoreinheit wird um das Gelenk *A5*, siehe Abbildung 3.6, des Arms geschwenkt. Ebenso wird bei der Bewertung des *VP* eine Optimierung der Konfigurati-

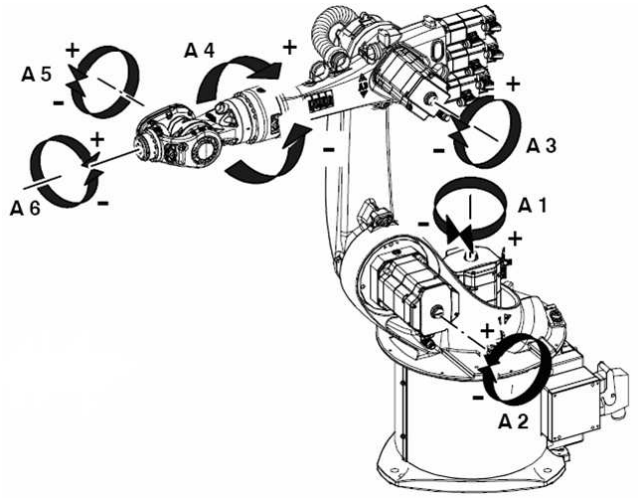
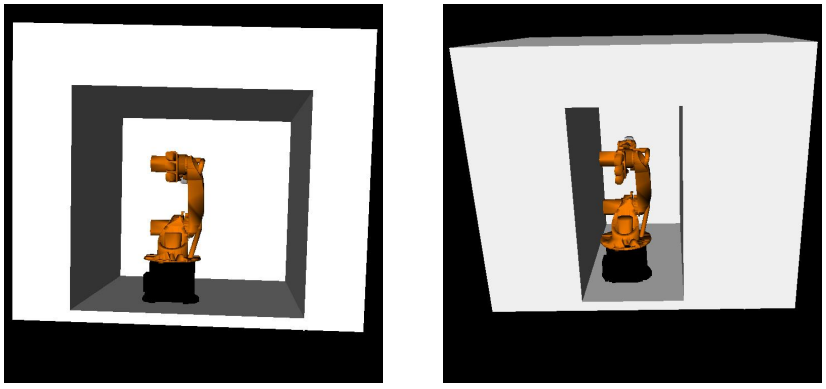


Abbildung 3.6: Gelenknummerierung des KR16 [24]

on hinsichtlich des Gelenks *A4* durchgeführt. Um diese *VP*-Menge abzudecken, wird anstelle des Robotermodells am Gelenk *A5* eine Kugel mit Radius des letzten Robotergliedes mit angeschlossenem 3D-Modellierer eingefügt. Mit dieser veränderten Geometrie wird genauso wie im vorherigen Verfahren geplant.
Bei den hier beschriebenen Verfahren ist nicht definiert, wie weit die kollisionsfreie von der gesampleten Konfiguration entfernt sein darf. Daher muss die Konfiguration, die an die *NBV* Planung weitergegeben wird nicht in der Nähe der durch die verschiedenen Verfahren ausgewählten Konfigurationen liegen. Damit verliert diese die erwünschten Eigenschaften. Es wird daher untersucht, wie sich eine Beschränkung dieser Entfernung auf die Planungsdauer auswirkt.

Szenario Für das *VP* Sampling werden zwei verschiedene Szenarien untersucht. Eine Umgebung ist relativ offen, sodass sich der Roboter gut bewegen kann. In der anderen Umgebung ist der bekannte Bereich sehr klein und es verbleibt dementsprechend kaum Bewegungsfreiheit.
In Abbildung 3.7a ist die offene Umgebung des Roboters dargestellt. Die Arbeitszelle von 2m$\times$2m$\times$2m ist hier vollständig frei. Außerhalb sind bis auf die Ansicht von vorne jeweils 0,5m starke unbekannte Zonen, die erkundet werden sollen.
Die enge Umgebung ist in Abbildung 3.7b dargestellt. Der freie Bereich ist hier auf eine Zelle von 1m$\times$2m$\times$2m reduziert und um 0,1m aus der vorderen Sicht nach links versetzt. Die äußeren Abmessungen des unbekannten Bereichs sind mit denen aus der anderen Umgebung identisch.



(a) Umgebung mit offenen Raum (b) Umgebung mit engem Raum

Abbildung 3.7: Explorationsszenarien Hand-Auge-System

Ergebnisse In den Diagrammen 3.8 und 3.9 sind die Erwartungswerte der Planungsdauer für jeweils 100 *VP*-Samples dargestellt. Für jede Planung wurde eine Reihe von 20 Messungen aufgenommen. Da für die erste Methode *M0* keine Beschränkungslänge definiert ist, wird diese nur in den unbeschränkten Messungen mit angegeben.
Bei beiden Umgebungen und allen betroffenen Methoden zeigt sich, dass die Planung durch die Beschränkung langsamer wird. Dies ist die direkte Folge des engeren Auswahlkriteriums, da mehr Konfigurationen gesampled werden müssen, um die gewünschte Anzahl an *VP*s zu erhalten. Während die Planungsdauer bei der Beschränkung auf 50 Metrikeinheiten zu einer leichten Zunahme der führt, ist die Planung mit einer Beschränkung auf 10 Metrikeinheiten entscheidend langsamer. Bei der Einordnung der Beschränkung ist zu beachten, dass die Auflösung des KR16 fünf Metrikeinheiten beträgt, eine Beschränkung auf zehn Metrikeinheiten also nahe an der theoretisch möglichen Forderung liegt.
Die offene Umgebung führt im Vergleich zu der engeren Umgebung zu schnelleren Ergebnissen bei den meisten Verfahren. Auf *M3* hat die Umgebung allerdings relativ wenig Einfluss.

Evaluierung im SBP-Simulator Für die Evaluierung der praktischen Bedeutung der Vorselektion der *VP*s wurde unter Einsatz des *SBP*-Simulators eine sensorgestützte Exploration einer Arbeitzelle durchgeführt. Die Exploration ist für die Untersuchung in fünf identische Schritte unterteilt worden. Am Anfang jeder Iteration wird die Umgebung auf Basis des aktuell bekannten Raums im *SBP*-Simulator für die Pfadplanung generiert. Dann werden mit der jeweiligen Methode 200 *VP* Kandidaten ermittelt. Als Beschränkungslänge wurde auf Basis der Auswertung der Samplingdauer eine Län-

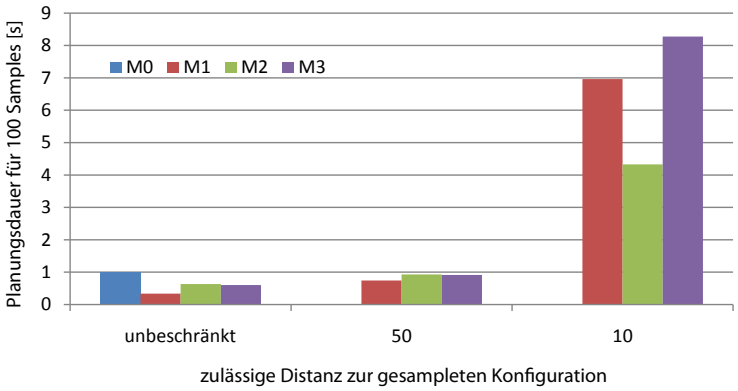


Abbildung 3.8: Erwartungswert der Planungsdauer für 100 *View-Point* Samples in offener Umgebung

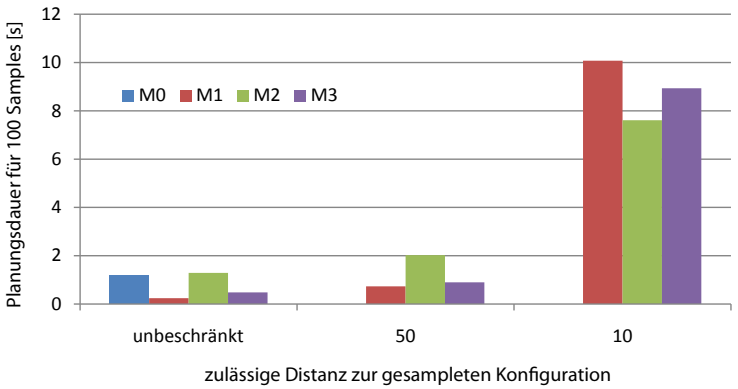


Abbildung 3.9: Erwartungswert der Planungsdauer für 100 *View-Point* Samples in enger Umgebung

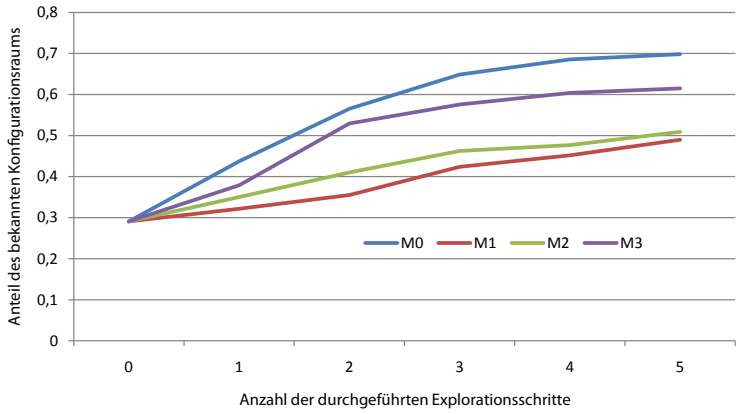
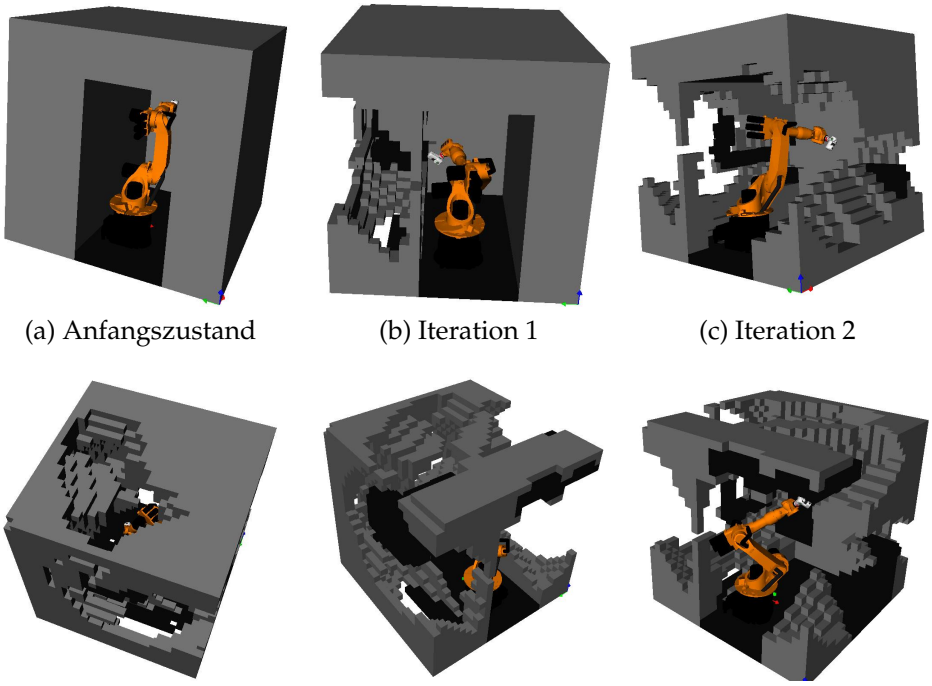


Abbildung 3.10: Explorationsverlauf unter Verwendung verschiedener *View-Point* Samplingstrategien

ge von 50 Metrikeinheiten gewählt. Diese werden mit Hilfe eines sogenannten *Beam-Sweep-Verfahrens* hinsichtlich des zu erwartenden *IG*s bewertet, nähere Informationen dazu siehe Suppa [23]. Die zehn besten *VP*s werden dann der Reihe nach auf den von der Pfadplanung berechneten Wegen angefahren um dort eine simulierte Messung aufzunehmen. Da bereits bei der Generierung auf eine gute Erreichbarkeit der Kandidaten geachtet wurde, sind die Wege sehr schnell durch ein *RDT-connect* Verfahren zu berechnen.
In Diagramm 3.10 sind die Explorationsfortschritte über die Iterationszahl für die verschiedenen *VP* Samplingverfahren aufgetragen. Da die Planung durch den Einsatz von samplingbasierten Methoden nicht deterministisch ist, die Simulationsdauer aber mit ca. 1 Stunde sehr lang ist, wird der Erwartungswert auf Basis von fünf Simulationen geschätzt. Die Anteile des freien Konfigurationsraums werden mit Hilfe von 5000 zufällig gesetzten Konfigurationen, die auf Kollisionen überprüft werden, ermittelt.
Am Anfang der Exploration, also im Iterationschritt 0, sind etwa 30 % des Konfigurationsraums bekannt. Mit Methode M0, also dem zufälligen Auswählen von gut erreichbaren Konfigurationen sind nach fünf Iterationen etwa 70 % von $\mathcal{C}$ bekannt. Dieses ist das beste Verfahren. Methode M3, die eine Kugel um das letzte Roboterglied und den Sensorkopf verwendet, kann nach den Iterationen 60 % von $\mathcal{C}$ dem bekannten Raum zuordnen. Die Verfahren M1 und M2, die auf dem Rand von $\mathcal{C}_{unk}$ die Kandidaten suchen, kommen nach dem fünften Iterationschritt auf eine Quote von circa 50 %.
In der Bilderserie 3.11 sind die Zustände der Exploration nach dem jeweiligen Iterationsschritt abgebildet. Als Methode zur Bestimmung der *VP*-Kandidaten wurde M0 verwendet. Bei der eingesetzten Explorationsstrategie handelt es sich um ein *Beam-Sweep-Verfahren*. Die grauen Bereiche markieren den unbekannten Raum, die schwarzen den bekannten aber von Hindernissen besetzten Raum.



(a) Anfangszustand (b) Iteration 1 (c) Iteration 2

(d) Iteration 3 (e) Iteration 4 (f) Iteration 5

Abbildung 3.11: Explorationsfortschritte bei Einsatz von *VP* Kandidaten der Methode M0 und der *Beam-Sweep-Explorationsstrategie*

Kapitel 4

Humanoider Roboter Justin

In der aktuellen Forschung wird die Entwicklung eines humanoiden Roboters angestrebt. Der Ansatz ist, die menschliche Physiologie durch einen Roboter nachzubilden. Von diesem Vorgehen versprechen sich die Forscher eine Reihe von Vorteilen gegenüber anders geformten Robotern.
Der Mensch ist durch seine Physiologie in der Lage, ein sehr breites Spektrum an Aufgaben zu erfüllen. Angefangen beim Transport von schweren Lasten bis hin zu feinen Manipulationen wie z.B. dem Anfertigen einer Uhr ist der Mensch durch seinen Körper in der Lage beeindruckende Leistungen zu vollbringen. Die sich im Laufe der Evolution herausgebildeten Proportionen und Freiheitsgrade stellen daher eine gute Lösung für die Durchführung verschiedenster Aufgaben dar. So ist es nahe liegend, diese als Basis für die Entwicklung eines Roboters zu nehmen.
Ein weiterer Punkt ist, dass Roboter in Zukunft als Haushaltshilfe oder Assistent eines Handwerkers im alltäglichen Umfeld eingesetzt werden sollen. Diese Umgebung, die Räume, Arbeitsplätze und Werkzeuge sind für die menschliche Physiologie ausgelegt. Hat der Roboter eine identische Kinematik, so ist in für Menschen geschaffenen Umgebungen eine Kompatibilität sichergestellt.
Trotz der Vorteile eines humanoiden Roboters sind bis zum aktuellen Zeitpunkt keine derartigen Systeme im kommerziellen Einsatz. Dies liegt vor allem daran, dass viele Probleme, welche durch die hochkomplexe Kinematik eines humanoiden Roboters entstehen, noch nicht gelöst sind. Hinzu kommt, dass diese Roboter ihren maximalen Nutzen in der Interaktion mit Menschen oder der autonomen Durchführung von Aufgaben in alltäglichen Umgebungen haben. Erst durch ausgereifte Methoden der Exploration, Aufgaben- und Bewegungsplanung sowie durch Konzepte, die eine sichere Interaktion mit dem Menschen ermöglichen, kommen die Vorteile eines humanoiden Roboters voll zum tragen.
Für die Erforschung von Methoden, die zur Manipulation von Objekten mit einem humanoiden Roboter nötig sind, wurde am DLR das Robotersystem

Justin entwickelt. Es handelt sich dabei um einen Roboter mit einem Torso, montiert auf einer mobilen Plattform. In Abschnitt 4.1 wird kurz auf die Komponenten von Justin eingegangen.
Wesentlicher Bestandteil von Manipulationsaufgaben ist die Planung des Pfades, der abgefahren werden soll, um die Greifkonfiguration zu erreichen. Da humanoide Roboter eine sehr hohe Anzahl an Freiheitsgraden aufweisen, sind samplingbasierte Pfadplaner wie in Kapitel 2.2 beschrieben eine gute Wahl. In dieser Arbeit wird ein im Rahmen einer Studienarbeit entwickelter Pfadplaner *SamPP* [5] eingesetzt und erweitert.
Zentrales Element einer Pfadplanung ist die Kinematik und Geometrie des verwendeten Roboters. Abschnitt 4.2 zeigt wesentliche Punkte, die zur Erstellung des benötigten Modells von Justin für die Pfadplanung durchgeführt wurden.
Die in dieser Arbeit untersuchten Methoden werden anhand von zwei Szenarientypen erprobt, die in Abschnitt 4.3 beschrieben werden. Der Konfigurationsraum von Justin ist durch die hohe Anzahl an Freiheitsgraden sehr komplex, in Abschnitt 4.4 werden daher Methoden zur Analyse entwickelt und angewendet. Auf dieser Basis werden in Abschnitt 4.5 Methoden zur anwendungsorientierten Pfadplanung entwickelt und hinsichtlich ihrer Leistungsfähigkeit überprüft.

4.1 Hardware

In diesem Kapitel wird die Hardware des Robotersystems Justin, dargestellt in Abbildung 4.1, beschrieben. Da für diese Arbeit nur die Kinematik des Roboters relevant ist, wird auf eine Ausführung über die Sensorik und Regelungskonzepte verzichtet. Für eine genauere Betrachtung kann in den Veröffentlichungen von Ott et al. [18] und Fuchs et al. [7] nachgelesen werden.
Justin ist modular aufgebaut und besteht aus einer mobilen Plattform, einem Torso und einem Kopf. Des Weiteren hat das Robotersystem zwei symmetrische Arme und Hände. Im Folgenden werden die Komponenten, aus denen Justin aufgebaut ist, beschrieben.

DLR LWR III Bei den Armen von Justin handelt es sich um Leichtbauroboterarme LWR III, die am DLR [9] entwickelt wurden. Wie der menschliche Arm besitzt der LWR III sieben Freiheitsgrade, wodurch eine Redundanz in der Endeffektorlage entsteht. Die Kinematik des Armes lehnt sich dabei, ebenso wie die Gelenkwinkelbeschränkungen, stark an der menschlichen Physiologie an. Die Arme zeichnen sich durch ein geringes Eigengewicht, ca. 14 kg, bei gleichzeitig hoher Nutzlast von bis zu 15 kg aus. Die mechanische Leistungsfähigkeit der Arme liegt daher in einem ähnlichen Bereich

Abbildung 4.1: Humanoides Robotersystem Justin

wie die eines Menschen.

DLR-Hand II Als Hände werden bei dem Robotersystem die DLR-Hand II eingesetzt [2]. Die diese hat vier Finger: Daumen, Zeige-, Mittel- und Ringfinger. Jeder dieser Finger hat drei Freiheitsgrade, die Hand insgesamt zwölf. Die Gelenkpositionen und die Gelenkwinkelbeschränkungen sind dabei denen in der menschlichen Hand nachempfunden. Das letzte Gelenk in jedem Finger ist von dem vorherigen Gelenk direkt abhängig. Die DLR-Hand II ist von den Abmessungen etwas größer als die durchschnittliche menschliche Hand, aber immer noch klein genug, um die meisten Werkzeuge bedienen zu können. Durch die integrierten Sensoren und Aktoren ist es möglich, sowohl kräftige als auch präzise Griffe durchzuführen.

Torso, Kopf und mobile Plattform Der Torso des Robotersystems hat vier rotatorische Gelenke, von denen die drei Horizontalen immer eine senkrechte Lage des obersten Torsoelements, an dem die Arme und der Kopf montiert sind, sicherstellen. Mit der Rotation um die senkrechte Achse von Justin hat der Torso somit drei Freiheitsgrade.
Der Kopf des Systems enthält die visuellen Sensoren des Roboters und verfügt über zwei mechanische Freiheitsgrade ähnlich der menschlichen Physiologie.
Die mobile Plattform des Roboters ermöglicht eine holonome Bewegung des Robotersystems auf einer Ebene. Außerdem kann die Standfläche des Roboters durch Ein- und Ausfahren der vier Räder verändert werden. Dadurch ist die Stabilität des Systems bei Manipulationen sichergestellt und es können dennoch enge Passagen, z.B. eine Zimmertür, passiert werden.

4.2 Pfadplanungsmodell des Robotersystems Justin

In diesem Abschnitt wird die Entwicklung eines Modells von Justin für *SamPP* beschrieben. Dieses muss die für die Pfadplanung relevanten Daten über Justin enthalten. Dazu zählt zunächst die Geometrie des Roboters, siehe Abschnitt 4.2.1. Neben der Form des Roboters sind die möglichen Bewegungen die zweite wichtige Komponente, die *SamPP* mit Hilfe des Modells zur Verfügung gestellt werden muss. Des Weiteren muss die Auswirkung von Kollisionen definiert werden, wie in Abschnitt 4.2.3 beschrieben.

4.2.1 Geometrie

Die Geometrie von Justin, also die Form der Roboterteile, wird mit Hilfe von VRML-Dateien an *SamPP* übergeben. Dazu wird jedes Roboterteil als einzelne Datei angegeben.
Für die 3D-Modelle dieser Teile wurde auf bestehende, konvexe Vereinfachungen der CAD-Daten zurückgegriffen. Konvexe 3D-Modelle haben den Vorteil, dass die verwendete Kollisionsdetektion *Solid*, welche den Gilbert-Johnson-Keerthi Algorithmus nutzt, effizienter arbeitet. Dies hat den Grund, dass der Algorithmus nur auf konvexe Objekte anwendbar ist. *Solid* zerlegt nichtkonvexe Objekte in konvexe Teile um eine Kollisionsabfrage bearbeiten zu können. Sind die Objekte alle konvex, kann dieser Schritt übersprungen werden. Weitere Informationen dazu können in der Veröffentlichung von Bergen [25] nachgelesen werden.
Für die mobile Plattform wurde auf eine einzelne Modellierung der Teile verzichtet. Stattdessen wurde eine konvexe Einhüllung der Gesamtstruktur als 3D-Modell verwendet. Auf Basis einer Betrachtung der Kinematik, siehe dazu Abschnitt 4.2.2, wurde die Plattform im Zustand minimaler Ausmaße, also mit vollständig eingefahrenen Rädern, verwendet.
Bei der Bearbeitung der 3D-Modelle der Roboterteile muss darauf geachtet werden, dass die Koordinatensysteme mit den durch die Kinematik bestimmten Koordinatensystemen übereinstimmen.

4.2.2 Kinematik

Für die Planung der Bewegungen von Justin muss *SamPP* die Kinematik des Robotersystems bekannt sein. Die Beschreibung muss dafür im Roboter-Modell in einem auf den Denavit-Hartenberg Parametern [4] basierenden Format abgelegt sein. *SamPP* vewendet dafür die Parameter nach Paul [19] während die Parameter der Bauteile von Justin nach Craig [3] vorlagen.
Mit den DH-Parametern lassen sich kinematische Ketten mit wenigen Parametern darstellen. Der Roboter Justin kann allerdings nicht ausschließlich durch kinematische Ketten beschrieben werden.
In *SamPP* werden daher die DH-Parameter so erweitert, dass ein kinematischer Baum beschrieben werden kann. Als Wurzelknoten wurde dabei das unterste Element des Torsos gewählt. Daran schließt die mobile Plattform sowie die anderen Torsoelemente an. Am letzten Torsobauteil verzweigt sich der kinematische Baum in die beiden Arme und die Kopfelemente. Am letzten Teil der Arme ist jeweils eine Hand verknüpft, die sich dann wieder in die jeweils vier Finger verzweigt.
Mit den vier DH-Parametern, welche für jedes Roboterteil zur Verfügung stehen, kann nicht jede beliebige Transformation dargestellt werden. Um dennoch die Kinematik von Justin beschreiben zu können, wurden virtuel-

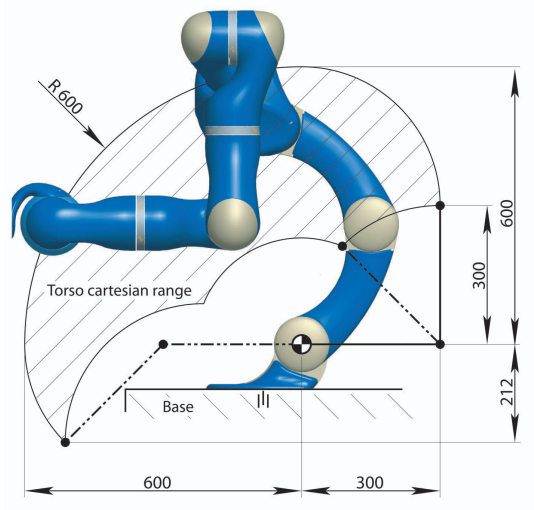


Abbildung 4.2: Arbeitsraum des Torsos von Justin [18]

le Roboterteile, also Teile ohne zugehöriges 3D-Modell, in die kinematische Beschreibung mit aufgenommen. So hat das Modell von Justin 86 Komponenten während das Robotersystem nur aus 55 realen Elementen besteht.
Nicht alle Gelenke stellen dabei Freiheitsgrade dar. Die Gelenke für die Fingerspitzen haben immer den halben Winkel des vorherigen Gelenks und sind daher keine Freiheitsgrade.
Die Gelenke im Torso werden über Seile gestellt. Die Konstruktion stellt dabei sicher, dass das oberste Torsoelement immer senkrecht steht. Damit haben die drei waagrechten Gelenke im Torso zusammen nur zwei translatorische Freiheitsgrade. Der sich daraus ergebende Arbeitsraum ist in Abbildung 4.2 dargestellt. Diese Kinematik ist mit *SamPP* aber nicht umsetzbar, da abhängige Gelenke nicht mehrere Referenzgelenke haben können. Daher wurde im Modell der translatorische Freiheitsgrad in der horizontalen Ebene auf Null festgelegt. Der Oberkörper steht so immer senkrecht über der mobilen Plattform. Da sich damit das maximale Moment, welches zum Kippen des Roboters führen könnte verringert ist mit eingefahrenen Rädern eine ausreichende Stabilität gewährleistet ist.
Für das Gesamtsystem ergeben sich, wie aus Tabelle 4.1 zu entnehmen ist, 45 Freiheitsgrade für Justin. Da die beiden Freiheitsgrade des Kopfes für eine Manipulation nicht relevant sind, werden in der weiteren Arbeit diese beiden Freiheitsgrade als starr betrachtet. Damit ergeben sich für das Robotersystem 43 Freiheitsgrade.
Um leichter auf die Freiheitsgrade verweisen zu können wurde die in der letzten Spalte der Tabelle aufgeführte Nummerierung eingeführt.

Tabelle 4.1: Freiheitsgrade des Robotersystem Justin

Bauteil	DoF	Nummerierung
Torso	2	2, 3
Kopf	2	-
Mobile Plattform	3	0, 1, 42
Arme	2×7	rechts 4-10, links 11-17
Hände	2×12	rechts 18-29, links 30-41
Insgesamt	45	0-42

4.2.3 Kollisionen

Um die Konfiguration des Roboters bewerten zu können müssen Kollisionen richtig erkannt werden. Dabei ist zu beachten, dass zwei Arten von Kollisionen auftreten können.
Im ersten Fall berühren ein oder mehrere Roboterteile Hindernisse. Eine solche Situation bedeutet immer, dass die zugehörige Konfiguration ein Element aus $\mathcal{C}_{occ}$ ist.
Anders liegt der Fall, wenn sich zwei Roberterteile berühren, also eine Eigenkollision vorliegt. Hier gibt es die Möglichkeit, dass eine Kollision erlaubt ist und daher nicht automatisch dazu führt, dass die Konfiguration Element aus $\mathcal{C}_{occ}$ ist. Dies ist beispielsweise meist bei über ein Gelenk aneinander grenzenden Teilen der Fall. Hier wird davon ausgegangen, dass die Gelenkbeschränkungen nicht erlaubte Kollisionen verhindern. Eine Überprüfung auf Kollision solcher Teile ist daher unnötig und sollte aus Performanzgründen vermieden werden.
Dürfen sich zwei Roboterteile nicht berühren, so führt eine Kollision dazu, dass die Konfiguration in $\mathcal{C}_{occ}$ liegt. Solche Roboterteilpaare müssen daher auf Kollisionen überprüft werden und als solche im Robotermodell angegeben werden. Es sei denn, eine Kollision kann durch die Kinematik ausgeschlossen werden.

4.3 Szenarien

In diesem Abschnitt werden die in dieser Arbeit verwendeten Szenarien zur Evaluierung der Pfadplanung beschrieben. Damit ist zunächst die Umgebung, in welcher der Roboter eine Aufgabe erfüllen soll, zu verstehen. Die Hindernisse schränken den freien Konfigurationsraum des Roboters ein und determinieren damit die effizienten Methoden, in dieser Umgebung einen Pfad zu finden.

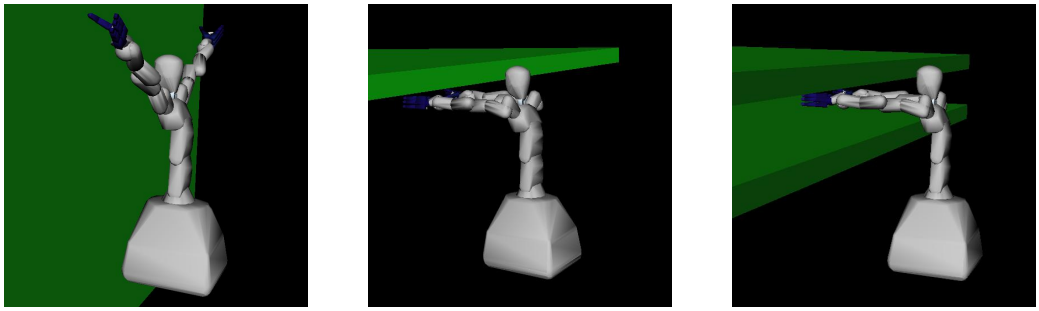


Abbildung 4.3: Zur Planungsevaluierung verwendete abstrakte Szenarien

Neben der Anordnung der Hindernisse ist aber auch wichtig, welche Aufgaben in der jeweiligen Umgebung sinnvoll bzw. realistisch sind. Diese Fragestellung ist allerdings nur für wirklichkeitsnahe Szenarien zu beantworten.
Im Folgenden werden die zwei verwendeten Szenarientypen vorgestellt. Zum einen ein abstraktes Szenario 4.3.1 und andererseits ein potentiell in der Zukunft realistisches Szenario, beschrieben in Abschnitt 4.3.2, in welchem Justin an einem Montageplatz arbeiten soll.

4.3.1 Abstraktes Szenario

Das abstrakte Szenario, also eine Planungssituation in einer auf wenige rudimentäre Objekte reduzierten Umgebung, ist der Normalfall für die meisten Arbeiten , die sich mit Pfadplanung beschäftigen [8] [26] [29]. Dies hat neben dem trivalen Grund, dass abstrakte Szenarien schneller zu erstellen sind, verschiedene Begründungen.
Oft will der Autor mit diesen Szenarien zeigen, wie der jeweilige Algorithmus sich bei engen Stellen oder Gängen im Konfigurationsraum verhält. Da dies die schwierigsten Situationen in der Pfadplanung sind, wird oft der Roboter durch ein Loch oder zwischen Wänden hindurch manövriert. Bei weniger eindeutigen Hinderniskonstellationen ist es durch die Vielfalt der Kollisionsmöglichkeiten bei komplexen Robotern oft schwierig, Aussagen über die Beschaffenheit des Konfigurationsraums zu machen.
Ein weiter Grund ist die Tatsache, dass die samplingbasierte Pfadplanung sich größtenteils noch in der Grundlagenforschung befindet. Die wenigsten Arbeiten behandeln die Entwicklung einer Pfadplanung für eine bestimmte Aufgabe oder ein bestimmtes Robotersystem.
Auch in dieser Arbeit werden zur Evaluierung verschiedener Methoden die in Abbildung 4.3 dargestellten abstrakten Szenarien eingesetzt um eine Vergleichbarkeit verschiedener Ansätze zu ermöglichen und Einblicke in die Pfadplanung mit dem Robotersystem Justin zu gewinnen.

4.3.2 Werkstatt/Arbeitsplatz

Wie in der Einleitung dieses Kapitels schon erwähnt, können humanoide Robotersysteme vor allem in für Menschen geschaffenen Umgebungen von ihrer Kinematik profitieren. Ein denkbares Einsatzgebiet eines humanoiden, autonomen Roboters ist ein Montagearbeitsplatz in einem Handwerksbetrieb. In dieser Arbeit wird daher der Einsatz von Justin in einer solchen Umgebung untersucht. Da Justin für Manipulationsaufgaben entwickelt wurde, werden auch in dieser Arbeit Pfadplanungen für Manipulationen untersucht. Allerdings liegt der Schwerpunkt darauf, bekannte Greifpositionen autonom anzufahren. Die Gegenstände werden dabei nicht bewegt.
Der in diesem Szenario verwendete Arbeitsplatz, siehe Abbildung 4.4, hat eine Fläche von 3,4m $\times$ 3,6m. Darauf befindet sich ein L-förmiger Arbeitstisch. An der Wand hinter einer Schreibtischseite sind auf 1,45m und auf 1,9 m Höhe mehrere Regale. Am Arbeitsplatz befinden sich auch verschiedene Gegenstände wie z.B. Boxen oder Werkzeug.
Für einen effizienten Einsatz des Roboters müssen verschiedene Aufgaben autonom erfüllt werden können. Da es sich nicht um eine abgesperrte Fertigungsstraße handelt kann nicht davon ausgegangen werden, dass sich die Gegenstände und Werkzeuge immer an exakt der gleichen Position befinden. Der Roboter muss also in der Lage sein, Gegenstände, die an nicht fest definierten Stellen in seinem Arbeitsbereich sind, greifen zu können. Die Detektion der Gegenstände und die Greifplanung ist dabei nicht Thema dieser Arbeit, sondern ausschließlich, wie die ermittelte Greifkonfiguration angefahren werden kann.

4.4 Konfigurationsraum von Justin

Die Pfadplanung findet im Konfigurationsraum von Justin statt. Um effiziente Pfadplanungsverfahren zu finden ist es wichtig, die Topologie des Konfigurationsraums zu kennen. Bei hochdimensionalen Robotern wie Justin ist es nicht mehr möglich, ein Bild von $\mathcal{C}$ zu erstellen. In Abschnitt 4.4.1 werden daher Methoden entwickelt, mit denen Eigenschaften von $\mathcal{C}$ auch ohne explizite Darstellung abgeleitet werden können.
Bei einem komplexen Robotersystem wie Justin entsteht auch ohne Hindernisse durch Eigenkollisionen ein komplizierter Konfigurationsraum, siehe Abschnitt 4.4.2. Dieser bildet die Grundlage für jede Pfadplanung, da die besetzten Gebiete unabhängig von der Umgebung sind. Werden der Umgebung Hindernisse hinzugefügt, verringert sich der freie Raum im Konfigurationsraum. Je nach Art und Position des Hindernisses hat dies unterschiedlich starke Auswirkungen auf $\mathcal{C}$, siehe dazu Abschnitt 4.4.3.

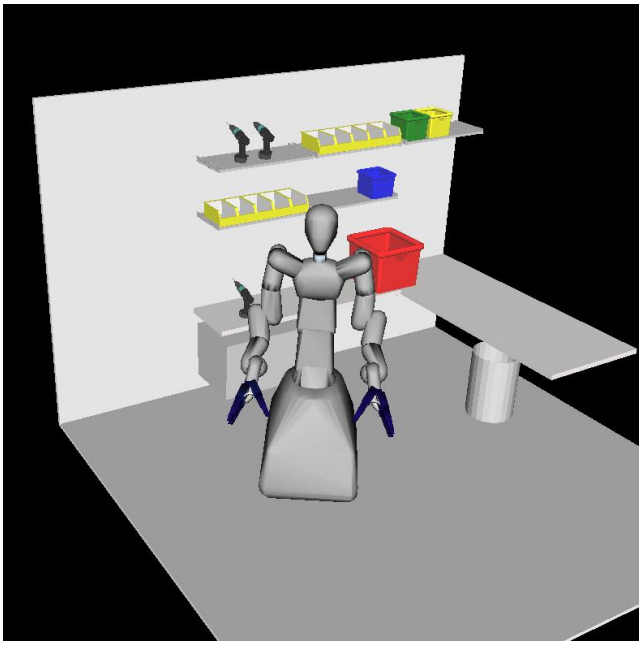


Abbildung 4.4: Werkstattszenario für die Durchführung von Manipulationsaufgaben

4.4.1 Methoden zur Untersuchung des Konfigurationsraums

Aufgrund der hohen Dimensionalität des Konfigurationsraums von Justin, ist er nicht mehr bildlich darstellbar. Auch durch Projektionen oder Schnitte kann dieses Problem nicht gelöst werden, da die Anzahl der benötigten Bilder viel zu hoch wäre. Selbst falls es durch eine geschickte Wahl der Schnitte gelänge, sämtliche Informationen über einen Bereich aus $\mathcal{C}$ darzustellen, so ist ein 43 dimensionaler Raum nicht vorstellbar.
In diesem Abschnitt werden daher einige Methoden entwickelt, mit denen zwar keine explizite Darstellung von $\mathcal{C}$ möglich ist, aber durch die erhaltenen Kenngrößen zumindest verschiedene Konstellationen von $\mathcal{C}$ miteinander verglichen werden können.

Volumen Das Volumen von $\mathcal{C}$ kann durch eine Multiplikation der einzelnen gewichteten Gelenkbereiche berechnet werden. Durch die Gewichtungsfaktoren ist die Beziehung zwischen translatorischen und rotatorischen Freiheitsgraden bereits hergestellt. Allerdings hat der Wert des Volumens keine hohe Aussagekraft über $\mathcal{C}$. Dies hat zum einen den Grund, dass die Einheit des Volumens von der Dimension von $\mathcal{C}$ abhängt und damit nur $\mathcal{C}$ der gleichen Dimension miteinander verglichen werden können. Zum anderen ist eine zentrale Eigenschaft der samplingbasierten Pfadplanungsmethoden nicht vom Volumen abzuhängen, sondern ausschließlich diskrete Punkte oder Verbindungen zu betrachten. Da sich das Volumen auch in-

tuitiv nicht erschließt wird es in dieser Arbeit nicht zur Charakterisierung von $\mathcal{C}$ eingesetzt.

Diagonale Die maximale Entfernung, die zwei Konfigurationen in $\mathcal{C}$ haben können, also die Diagonale des Raums, kann mit Hilfe der Metrik der Pfadplanung berechnet werden. Die Pfadplanung muss für eine Kollisionsüberprüfung dieser Verbindung ebensoviele Zwischenkonfigurationen überprüfen wie der Quotient aus Länge durch Auflösung ergibt. Da die Pfadplanungsalgorithmen auf Verbindungsüberprüfungen basieren, ist die Länge der Diagonalen geeignet Rückschlüsse über die Größe des Konfigurationsraums zu ziehen. Eine weitere positive Eigenschaft dieser Länge ist, dass sie dimensionsunabhängig ist. So können verschieden dimensionale Räume miteinander verglichen werden.

Verhältnis $\mathcal{C}_{free}$ zu $\mathcal{C}$ Die Größe von $\mathcal{C}$ ist nur ein Faktor für die Pfadplanung, entscheidend ist auch, wie viele freie Bereiche $\mathcal{C}_{free}$ existieren und viele besetzt sind $\mathcal{C}_{occ}$. Da eine explizite Berechnung dieser Bereiche nicht möglich ist, müssen hier stochastische Methoden eingesetzt werden. Da für Justin nur uniforme Samplingstrategien eingesetzt werden, kann von Folgendem ausgegangen werden:

$$\lim_{N_{Samples}\to\infty} \frac{N_{Samples}(\mathbf{q} \in \mathcal{C}_{\mathbf{free}})}{N_{Samples}} = \frac{vol(\mathcal{C}_{free})}{vol(\mathcal{C})} \tag{4.1}$$

Rückschlüsse aus der Erstellung einer Roadmap Neben den quantitativ ermittelbaren Kenngrößen können aus der Erstellung einer *Roadmap* eine Reihe von qualitativen Rückschlüssen über $\mathcal{C}$ gezogen werden. Besonders die *PRM-visibility* eignet sich zur Analyse von $\mathcal{C}$:

- Verbinden sich die Komponenenten am Anfang sehr schnell zu einer gemeinsamen Komponente, so ist $\mathcal{C}$ sehr zusammenhängend und wenig zerklüftet.
  Kristallisieren sich zunächst mehrere größere Komponenten heraus, die lange nicht miteinander verbunden werden können, so enthält $\mathcal{C}$ mehrere gut zusammenhängende Bereiche, die über Engstellen miteinander verbunden sind.
  Verbinden sich die Komponenten trotz einer sehr hohen Anzahl von Samples nicht, so ist der Konfigurationsraum mit hoher Wahrscheinlichkeit in mehrere Bereiche unterteilt, die nicht miteinander verbunden werden können.

- Ist die *Roadmap* klein und dennoch werden die meisten Samples verworfen, so ist $\mathcal{C}$ mit wenigen Konfigurationen abdeckbar und damit

wenig zerklüftet. Ebenso existieren keine Engstellen, die große Bereiche von $\mathcal{C}$ trennen. Ist die *Roadmap* sehr groß bevor die meisten Samples verworfen werden oder wird dieser Zustand gar nicht erreicht, so ist $\mathcal{C}$ sehr verwinkelt.

- Ist die Lebensdauer kleiner Komponenten bei einer *Roadmap*, welche die meisten Samples verwirft, hoch, so hat $\mathcal{C}_{free}$ einen zerfurchten Rand. Können kleine Komponenten schnell in die großen Komponenten integriert werden, ist der Rand glatt.

4.4.2 Freier Konfigurationsraum

In diesem Abschnitt wird der Konfigurationsraum von Justin ohne Hindernisse analysiert. Trotz der leeren Umgebung sind viele Bereiche Teilmengen von $\mathcal{C}_{occ}$. Mit diesen durch Eigenkollisionen verursachten Bereichen und dem damit eingeschränkten Konfigurationsraum müssen sämtliche Planungen durchgeführt werden. Es ist daher sinnvoll, die im Abschnitt 4.4.1 vorgestellten Methoden zunächst auf den Konfigurationsraum ohne Hindernisse anzuwenden.
Da die Kinematik von Justin definiert ist, kann der Konfigurationsraum in diesen Fall nur durch die Einschränkung der Gelenkbereiche oder Blockierung von Freiheitsgraden beeinflusst werden. Die Gelenkbereiche sind bereits so weit wie möglich auf das Szenario angepasst, daher werden im Folgenden die Auswirkungen einer Reduzierung der Freiheitsgrade untersucht.
Die Freiheitsgrade, die aus der Kinematik herausgenommen werden, müssen dabei so gewählt werden, dass das resultierende Robotersystem weiterhin die Manipulationsaufgaben erfüllen kann. So sind die Freiheitsgrade der mobilen Plattform oder des Torsos entbehrlich, wenn die Objekte auch ohne Bewegung des Gesamtsystems erreichbar sind. Wird ein Gegenstand nur mit einer Hand gegriffen, kann auf die Freiheitsgrade des zweiten Arms verzichtet werden.
In Tabelle 4.2 sind einige denkbare Varianten zur Reduzierung der Freiheitsgrade aufgelistet. In der zweiten Spalte sind die jeweils herausgenommenen Freiheitsgrade angegeben. Die dritte Spalte zeigt die daraus folgenden verbliebenen Freiheitsgrade. In der vierten Spalte sind die Längen der Diagonalen von $\mathcal{C}$ in Metrikeinheiten angegeben. Der prozentuale Anteil von $\mathcal{C}_{free}$ an $\mathcal{C}$, berechnet mit 100000 Samples, ist in der letzten Spalte dargestellt.
Anhand dieser Daten lassen sich einige Feststellungen über $\mathcal{C}$ machen. Zum einen ist $\mathcal{C}$ in Hinblick darauf, dass die Auflösung von *SamPP* in allen diesen Fällen eine Metrikeinheit beträgt, sehr groß. In der Variante mit allen Freiheitsgraden müssen nahezu 900 Kollisionsdetektionen durchgeführt werden um eine Verbindungslinie quer durch $\mathcal{C}$ zu überprüfen. Weiterhin ist

Tabelle 4.2: Konfigurationsraum Robotersystem Justin ohne Hindernisse

Nr	Reduzierte Freiheitsgrade	DoF	Diagonale	$\mathcal{C}_{free}$ [%]
1	keine	43	894	0.744
2	Hände	19	699	76.5
3	Hände, Torso, Plattform	14	547	78.5
4	Hände, linker Arm	12	425	87.0
5	Hände, Torso, Plattform, linker Arm	7	273	88.1

bemerkenswert, dass der größte Anteil der Diagonallänge durch die Arme verursacht wird.
Bei der Analyse der Eigenkollisionen zeigt sich, dass durch die Freiheitsgrade der Hände große Teile von $\mathcal{C}$ unabhängig von Hindernissen zu $\mathcal{C}_{occ}$ gehören. So liegt beim uniformen Sampling nicht einmal jede 1000. Konfiguration in $\mathcal{C}_{free}$. Weiteren, wenn auch weit geringeren Einfluss hat die Kollision zwischen den beiden Armen, während die Freiheitsgrade von Torso und mobiler Plattform nahezu keine Eigenkollision auslösen.
Trotz der teils erheblichen Menge an auftretenden Eigenkollisionen lässt sich durch das *PRM-visibility* Verfahren zeigen, dass alle Varianten einen sehr zusammenhängenden Konfigurationsraum haben. Die Größe der nötigen *Roadmap* lag bei allen Kinematikvariationen unter zwanzig Konfigurationen. Weitere Konfigurationen konnten immer sofort dieser *Roadmap* hinzugefügt werden.

4.4.3 Konfigurationsraum mit Hindernissen

Das Robotersystem Justin soll in einer Umgebung mit verschiedenen Objekten eingesetzt werden. Diese Objekte sind aus Sicht der Pfadplanung Hindernisse, die den Konfigurationsraum von Justin weiter einschränken. Dabei können je nach Position oder Form der Hindernisse auch kleine Objekte weite Bereiche von $\mathcal{C}$ unpassierbar machen. Bei einer komplexen Kinematik wie der von Justin ist durch die vielfältigen Kollisionsmöglichkeiten schwer vorherzusagen, wie sich welches Hindernis auf den Konfigurationsraum auswirkt. Eine glatte Wand in $\mathcal{P}$ kann zu einem zerklüfteten $\mathcal{C}$ führen. Auch können Passagen zwischen Hindernissen, die in $\mathcal{P}$ weiträumig wirken, in $\mathcal{C}$ sehr eng, oder sogar unpassierbar sein.
Im Folgenden wird der Konfigurationsraum von Justin unter Einbeziehung verschiedener Hinderniskonstellationen untersucht. Dabei werden zwei der in Abschnitt 4.4.2 untersuchten Kinematiken eingesetzt. Zum einen die mit allen Freiheitsgraden außer den Händen und andererseits die Kinematik, in

Tabelle 4.3: Konfigurationsraum Robotersystem Justin mit Hindernissen

Nr	Hindernis	Kinematik	$\mathcal{C}_{free}$ [%]
1	Wand	2	35,4
	Wand 0,5m entfernt	3	34,4
	Wand 1m entfernt	3	73,2
2	waagrechte Platte	2	23,7
		3	62,2
3	2 waagrechte Platten, Zwischenraum 0,5m	2	18,3
		3	55,0

der nur die Freiheitsgrade der Arme aktiv sind (siehe Nr. 2 und 3 der Tabelle 4.2). Außer des Anteils von $\mathcal{C}_{free}$ sind alle Kennwerte von der Umgebung unabhängig. Auf die Freiheitsgrade der Hände wird verzichtet, da durch diese die Gesamtform von Justin nur gering beeinflusst wird. Es kann davon ausgegangen werden, dass sich der Einfluss der Hindernisse auf den Konfigurationsraum in Beziehung auf die Hände vernachlässigen lässt.

4.4.3.1 Abstrakte Hindernisse

Mit Hilfe von geometrisch einfachen Hindernissen sollen die Auswirkungen verschiedener Hinderniskonstellationen auf $\mathcal{C}$ untersucht werden. Es kommen Platten zum Einsatz, die so positioniert werden, dass sie abstrahierte reale Situationen darstellen. So kann eine waagrechte Platte als Tischplatte oder Regal interpretiert werden, während senkrechte Platten Wände simulieren können.
In Tabelle 4.3 sind die quantitativen Ergebnisse der verschiedenen Umgebungen mit der jeweils verwendeten Kinematik angegeben. Es ist zu entnehmen, dass sich die Freiheitsgrade der Plattform und des Torsos im Gegensatz zur leeren Umgebung massiv auf das Verhältnis von $\mathcal{C}_{free}$ auswirken können.
Mit dem *PRM-visibility* Verfahren wird der Einfluss einer Wand auf $\mathcal{C}$ untersucht. Dabei ergibt sich, dass trotz des weiter eingeschränkten Konfigurationsraums in allen Fällen sehr schnell eine kleine *Roadmap* erzeugt werden kann, welcher in den meisten Fällen sämtliche folgende Konfigurationen hinzugefügt werden können. Lediglich im Fall des nicht mobilen Justins, dessen Mittelpunkt nur einen halben Meter von der Wand entfernt ist, entstehen einige Komponenten, die jeweils aus einer einzelnen Konfiguration bestehen. Bei diesen Konfigurationen ist ein Arm zwischen der Wand und dem Roboter eingeklemmt und kann nicht mit dem übrigen Konfigurationsraum verbunden werden.

In Umgebung 2 besteht das Hindernis aus einer waagrechten Wand, welche sich auf Schulterhöhe des aufgerichteten Justin befindet. Auch in dieser Umgebung kann jeweils sehr schnell eine kleine *Roadmap* erzeugt werden, bei der alle folgenden Konfigurationen hinzufügbar sind.
Es lässt sich daher für das Robotersystem Justin sagen, dass sowohl Wände als auch freistehende waagrechte Flächen, wie z.B. Tische den Konfigurationsraum nicht stark zerklüften, sondern, dass trotz teilweise erheblicher Einschränkungen des freien Volumens der Konfigurationsraum gut zusammenhängend bleibt. Auch der Rand von $\mathcal{C}_{free}$ bleibt glatt.
In der dritten Hinderniskonstellation wird eine Engstelle im physikalischen Raum durch zwei relativ eng übereinanderliegenden Ebenen erzeugt. Damit kann z.B. ein tiefes Regal simuliert werden. Bei der Analyse der *Roadmap* Erzeugung ergeben sich dadurch auch die zu erwartenden Schwierigkeiten. Es benötigt eine erheblich größere Roadmap um alle Konfigurationen in einer Komponente zu vereinen. Dabei stellt sich heraus, dass durch die Platten im Fall der Kinematik 3 nicht eine Engstelle in $\mathcal{C}$ entsteht, sondern eine Reihe untereinander nicht verbindbarer Bereiche. Es zeigt sich auch, dass die Kinematik 2 trotz der prozentual geringeren freien Bereiche eine bessere Konnektivität des Konfigurationsraums aufweist.

4.4.3.2 Werkstatt Szenario

Die Untersuchungen der vorherigen Kapitel dienen dazu, für das Werkstatt Szenario geeignete Pfadplanungsalgorithmen zu finden. In diesen Abschnitt wird nun der Konfigurationsraum von Justin in der Werkstatt Umgebung untersucht. Da sich für die nicht beweglichen Kinematiken stark variierende Konfigurationsräume je nach Position des Roboters ergeben, werden hier nur die Konfigurationsräume der drei mobilen Kinematiken untersucht. Dies führt zu keiner detaillierten Lösung für die Pfadplanung, ermöglicht aber eine grobe Einordnung der Werkstattumgebung.
Die Ergebnisse der Untersuchung sind in Tabelle 4.4 zusammengefasst. In Spalte 2 ist der absolute, prozentuale Anteil von $\mathcal{C}_{free}$ an $\mathcal{C}$ aufgeführt. Spalte 3 gibt die Differenz zu dem Anteil der Besetzung in freier Umgebung an. Dabei zeigt sich, dass im Werkstatt Szenario viel freier Raum zur Planung verbleibt.
Die Analyse des Werkstatt Szenarios mit Hilfe einer *Roadmap* zeigt, wie schon nach den Ergebnissen in der abstrakten Umgebung zu erwarten war, dass der Konfigurationsraum gut zusammenhängend ist.

Tabelle 4.4: Konfigurationsraum Robotersystem Justin bei Werkstattumgebung

Kinematik	$\mathcal{C}_{free}$ [%]	$\Delta\mathcal{C}_{free}$ [%]
1	0,584	-0,16
2	57,8	-18,7
4	63,0	-24,0

4.5 Anwendungsorientierte Pfadplanung

In diesem Kapitel wird mit verschiedenen Ansätzen eine Pfadplanung für den Einsatz von Justin in einer Werkstatt entwickelt. Damit wird versucht, die These zu belegen, dass durch eine Anpassung der Pfadplanung an den jeweiligen Anwendungsfall eine Performanzsteigerung zu erreichen ist. Dabei wird auf die Ergebnisse der vorherigen Kapitel zurückgegriffen. Ebenso werden die in diesem Szenario zu erwartenden Anforderungen bei dem Entwurf der Verfahren berücksichtigt.
Wie aus Kapitel 4.4 zu entnehmen ist, haben die Freiheitsgrade großen Einfluss auf den Konfigurationsraum. Daher liegt die Vermutung nahe, dass sich dies auch auf die Pfadplanung auswirkt. In Abschnitt 4.5.1 wird daher die Pfadplanung in Hinblick auf die Freiheitsgrade von Justin untersucht.
Der Einsatz von Justin in einer Werkstatt dient dazu, bestimmte Aufgaben zu erfüllen. Diese sind, ebenso wie einige feste Elemente in der Umgebung bekannt. In Abschnitt 4.5.2 werden Möglichkeiten untersucht, das Wissen über den Einsatz des Roboters und der Umgebung auszunutzen, um die Pfadplanung effizienter gestalten zu können.
Die in Abschnitt 4.5.3 vorgestellten Verfahren untersuchen eine aus der Greifplanung entstehende Aufgabenstellung. Für Gegenstände, die manipuliert werden sollen, können oft sehr viele, unterschiedlich gute Griffe ermittelt werden. Es ist aber nicht ohne weiteres möglich, die Erreichbarkeit dieser Konfigurationen anzugeben. Für Manipulationsszenarien ist es daher interessant, mit Zielkonfigurationsmengen arbeiten zu können. Da die in *SamPP* implementierten Verfahren nicht deterministisch sind, sondern stochastischen Schwankungen unterliegen wird im Folgenden das Ergebnis jeder Planung durch den Erwartungswert der gleichverteilten Zufallsvariablen X, welche durch die Ergebnisse von 20 Planungsdurchläufen definiert wird, angegeben. In Tabelle 4.5 sind die Daten des für die Simulationen in diesem Abschnitt eingesetzten Rechners aufgeführt.

Tabelle 4.5: Pfadplanungsrechner für das humanoide Robotersystem

Daten des Pfadplanungsrechners	
Prozessor	Intel Core 2 Duo P9600 2,66 GHz
Arbeitsspeicher	4 GB RAM
Betriebssystem	Microsoft Windows Vista Business

4.5.1 Variable Freiheitsgrade

In diesem Abschnitt werden die schon in Kapitel 4.4 vorgestellten Kinematikvarianten hinsichtlich der Pfadplanung untersucht. Damit soll experimentell ermittelt werden, welche Freiheitsgrade sich wie auf die Pfadplanung in verschiedenen Situationen auswirken.
Zunächst werden dazu in einigen, ebenfalls aus vorherigen Kapiteln bekannten, abstrakten Umgebungen verschiedene Pfadplanungen mit jeweils unterschiedlichen Kinematiken untersucht, siehe dazu Abschnitt 4.5.1.1.
Anschließend werden in der Praxis möglicherweise auftretende Planungen in der Werkstatt Umgebung durchgeführt, siehe Abschnitt 4.5.1.2. Damit soll eine Relevanz auch für die Anwendung belegt werden.
Abschnitt 4.5.1.3 befasst sich mit den Problemen, die durch den Einsatz verschiedener Kinematiken zur Pfadplanung entstehen und zeigt Ansätze, wie diese bewältigt werden können.
In diesem Kapitel kommen *RDT* Verfahren des Typs *RDT-connect* und *RDT-visibility* zum Einsatz. Die *RDT-classic* Variante ist Aufgrund der in Kapitel 4.4 ermittelten Größe von $\mathcal{C}$ nicht geeignet. Auch auf einen Einsatz der *Roadmap* Verfahren wird verzichtet, da diese stark von Parametern abhängen, was in Hinblick auf die Untersuchungen in diesem Kapitel die Ergebnisse verfälschen würde.

4.5.1.1 Abstrakte Umgebung

Anhand von klar strukturierten, abstrakten Umgebungen, siehe Abschnitt 4.3.1, werden in diesem Abschnitt die Auswirkungen der Freiheitsgrade auf die Pfadplanung untersucht. Dafür wurde eine Pfadplanungsaufgabe erstellt, die in zwei der in Abschnitt 4.3.1 eingeführten Umgebungen durchgeführt wird. Zum einen müssen Start- und Zielkonfiguration in allen Umgebungen kollisionsfrei sein. Außerdem müssen diese mit allen Kinematiken erreichbar sein, d.h. nur die Komponenten der Konfiguration, die auch in der beschränktesten Kinematik aus Tabelle 4.2 variabel sind, können verändert werden. Folglich unterscheiden sich Start- und Zielkonfiguration in

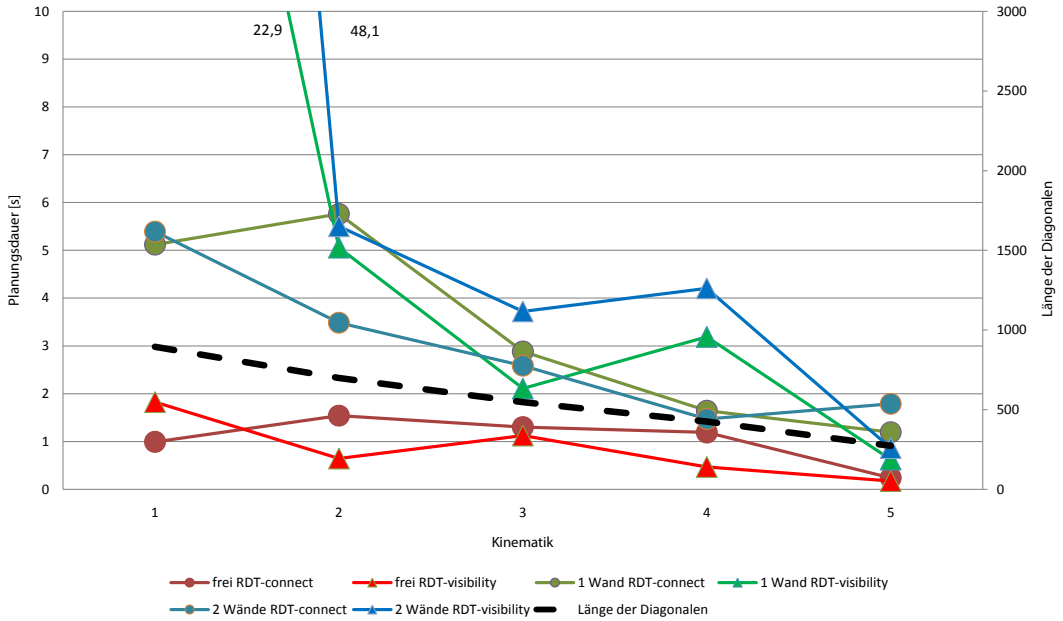


Abbildung 4.5: Ergebnisse der Pfadplanung mit Justin in abstrakten Umgebungen

den folgenden Planungen nur in den Freiheitsgraden des rechten Arms. Als Startkonfiguration wurde die Nulllage des Roboters gewählt. Die Zielkonfiguration ist bei waagrechtem, vollständig nach vorne gestreckten Arm erreicht. Die feste Position des Roboters liegt bei 1,2m in x-Richtung und 1,0m in y-Richtung.

Ergebnisse Im Diagramm 4.5 sind die Ergebnisse der Planungen dargestellt. Auf der X-Achse sind die verwendeten Kinematiken aufgetragen, die Y-Achse trägt den Erwartungswert der jeweiligen Planungsdauer in Sekunden auf. Die mit *RDT-visiblity* erzeugten Datenreihen sind mit Dreiecken markiert, die *RDT-connect* Werte mit Kreisen. Die drei untersuchten Umgebungen sind farblich gekennzeichnet. Die roten Kurven sind in einer Umgebung ohne Hindernisse erzeugt worden. Für die grünen Kurven wurde eine waagrechte Platte eingefügt und für die blauen Kurven zusätzlich eine Zweite. Mit der schwarzen, unterbrochenen Linie wurde auf der sekundär Y-Achse die zur jeweiligen Kinematik berechnete Diagonallänge von $\mathcal{C}$ in Metrikeinheiten aufgetragen.
Aufgrund der vielfältigen Faktoren, die sich auf die Pfadplanungsdauer auswirken, ist eine genaue Analyse jeder Kurve sehr aufwändig und liefert dabei letztendlich nur Informationen zu dieser speziellen Planungskonstellation. Es wird daher versucht, Gemeinsamkeiten in den verschiedenen Da-

tenreihen zu finden um damit allgemeinere Aussagen über das Verhalten des Pfadplaners machen zu können und davon in den anwendungsorientierten Planungen zu profitieren.
Zunächst lässt sich feststellen, dass die Werte der Diagonallängen für die meisten Planungen eine ähnliche Tendenz wie die Erwartungswerte der Planungsdauer aufweist. Damit ist die Vermutung aus Kapitel 4.4.1, mit Hilfe der Diagonallänge eine zumindest grobe Bewertung des Konfigurationsraums hinsichtlich der Pfadplanung machen zu können, bestätigt.
Für das *RDT-visibility* Verfahren gilt, dass die Planungsdauer mit steigender Komplexität der Umgebung zunimmt. Mit Kinematik 5 bleibt der Erwartungswert in jeder Umgebung unter einer Sekunde und ist auch immer geringer als der des *RDT-connect* Verfahrens. Bei den Planungen mit Kinematik 1, also inklusive der Freiheitsgrade der Hände, steigt die Planungsdauer in den beiden Umgebungen mit Hindernissen stark an. Dies ist auf die geringe Quote von $\mathcal{C}_{free}$ bei dieser Kinematik zurückzuführen.
Auch bei den Werten, die mit dem *RDT-connect* Verfahren ermittelt wurden, liegt die minimale Planungszeit bei Kinematik 5. Die Komplexität der Hindernisse in der Umgebung hat aber keine direkt nachvollziehbare Auswirkung auf die Planungsdauer. Sehr interessant ist, dass die Eigenkollisionen der Hände in Kinematik 1 keine stark negative Auswirkung auf die Planung haben. In den beiden freieren Umgebungen nimmt die Planungszeit durch den Einbezug der Freiheitsgrade der Hände sogar ab.
Zusammenfassend lassen sich aus diesen Untersuchungen für die weiteren Planungen folgende Erkenntnisse gewinnen: Für Planungen mit ausschließlich einem Arm sind die *RDT-visibility* Verfahren effizienter. Werden die Freiheitsgrade der Hände miteinbezogen kommen vor allem *RDT-connect* Verfahren in Frage.

4.5.1.2 Werkstatt Szenario

In diesem Kapitel sollen die Auswirkungen der Freiheitsgrade auf Planungen unter realen Bedingungen untersucht werden. Kapitel 4.5.1.1 liefert dafür bereits grobe Anhaltspunkte. Durch die Vielfalt der Einflüsse kann aber eine genauere Aussage über die Pfadplanung nur durch eine möglichst realistische Simulation der Situation gewonnen werden. Dazu gehört auch, verschiedene Pfadplanungsaufgaben mit unterschiedlichen Start- und Zielkonfigurationen durchzuspielen.
In diesem Abschnitt werden daher auch Planungen, für welche die Freiheitsgrade der Plattform oder beider Arme benötigt werden, untersucht. Diese können nicht mit allen Kinematiken durchgeführt werden. Planungen mit Kinematik 1 werden nur noch mit *RDT-connect* Verfahren untersucht.
In Tabelle 4.6 sind Pfadplanungsaufgaben, die sich in einer Werkstatt stellen könnten aufgeführt. Neben der Planungsaufgabe findet sich jeweils ein

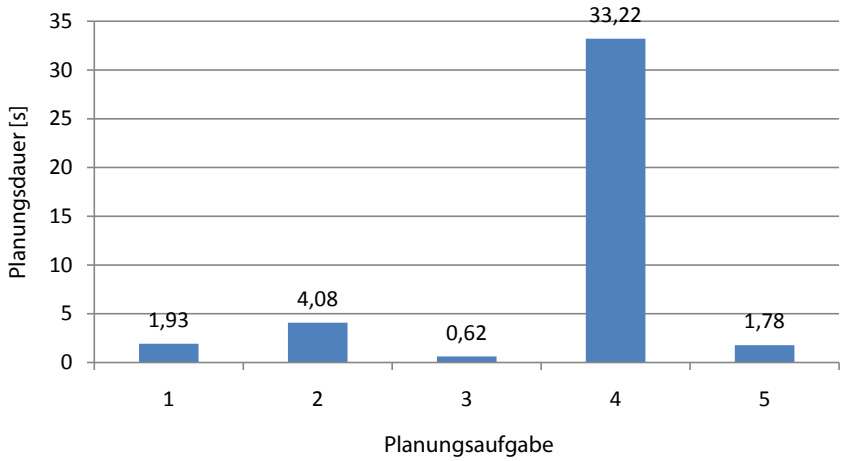


Abbildung 4.6: Planungsdauer der verschiedenen Aufgaben mit vollen Freiheitsgraden und *RDT-connect* Planungsverfahren

Bild von der Start- und Zielkonfiguration. Bei diesen Planungen kann nur die Aufgabe 1 und 3 mit anderen Kinematiken als der ersten untersucht werden, da sich bei allen anderen die Freiheitsgrade der Hände verändern.

Ergebnisse In Abbildung 4.6 sind die Erwartungswerte der Planungsdauer mit Kinematik 1 unter Verwendung des *RDT-connect* Verfahrens dargestellt. Damit kann der Schwierigkeitsgrad der jeweiligen Aufgabe eingeschätzt und eine generelle Eignung samplingbasierter Pfadplaner für einen Einsatz von Justin in einer Werkstatt abgeleitet werden. Deutlich zeigt sich dabei, dass die Aufgabe 4, das Ziehen einer schlecht erreichbaren Box, hier die höchste Anforderung an die Pfadplanung stellt. Alle anderen Aufgaben können teilweise in deutlich unter fünf Sekunden gelöst werden. Es zeigt sich, dass die Planungsdauer zumindest im zumutbaren Bereich für ein solches Szenario liegt.
Um die Auswirkungen der Freiheitsgrade untersuchen zu können werden im Folgenden die Planungsaufgaben 1 und 3 mit verschiedenen Kinematiken und sowohl *RDT-connect* als auch *RDT-visibility* Verfahren gelöst.

Ergebnisse Aufgabe 1 Bei Aufgabe 1, also dem Greifen in eine tiefe Box, variieren die Planungszeiten je nach Verfahren sehr stark. Daher wird auf eine Darstellung durch ein Diagramm verzichtet und stattdessen das Ergebnis tabellarisch angegeben. Neben dem Erwartungswert der Planungsdauer werden in Tabelle 4.7 Minimal- und Maximalwert sowie die Standardabweichung angegeben. Generell lässt sich für diese Pfadplanungsaufgabe

Tabelle 4.6: Pfadplanungsaufgaben in einer Werkstatt

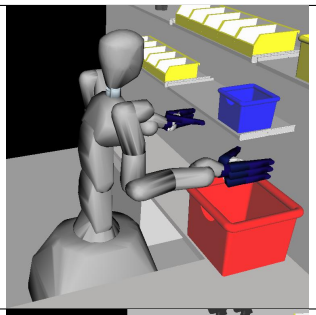
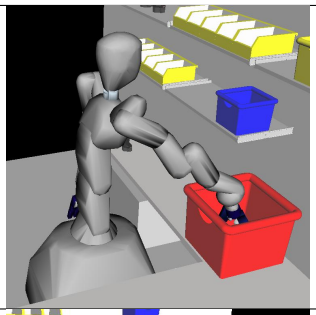
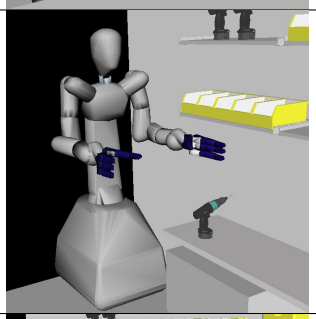
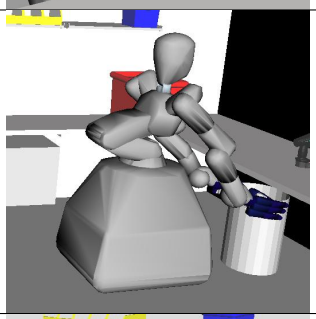
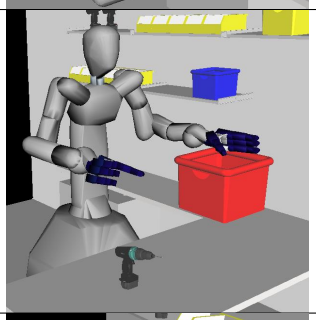
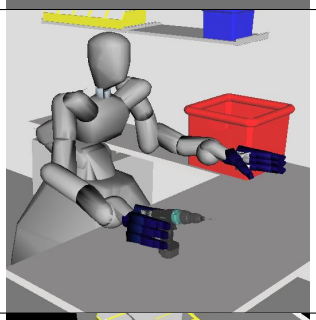
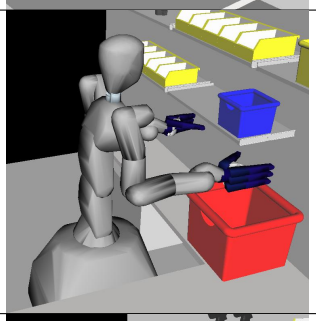
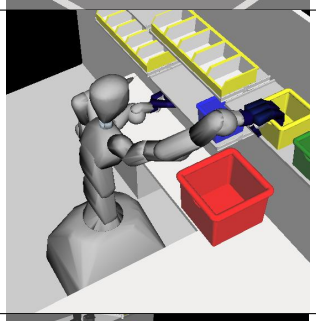
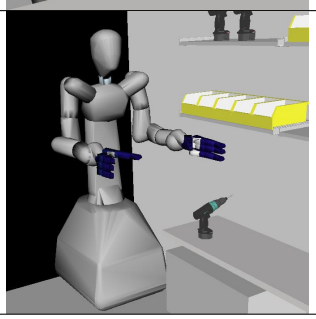
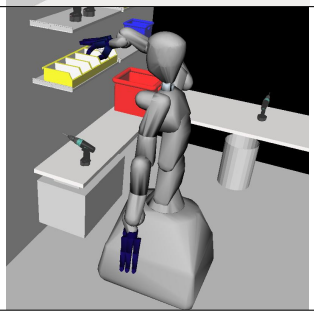
Nr	Planungsaufgabe	Startkonfiguration	Zielkonfiguration
1	Greifen in eine tiefe Box		
2	Aus einer Ruheposition den Mülleimer anfahren und greifen		
3	Greifen eines auf dem Arbeitstisch platzierten Werkzeugs		
4	Herausziehen einer schwer erreichbaren Box		
5	Greifen eines kleinen Gegenstandes aus einer schmalen Sortierbox		

Tabelle 4.7: Ergebnisse der Planungsaufgabe 1 mit verschiedenen Kinematiken und Planungsverfahren

Kinematik	Verfahren	$min(\mathbf{X})[s]$	$max(\mathbf{X})[s]$	$\sigma(\mathbf{X})[s]$	$E(\mathbf{X})[s]$
1	Connect	0,47	5,6	1,3	1,9
	Visibility	—	—	—	—
2	Connect	0,48	8,0	1,6	2,2
	Visibility	1,7	41,2	10,9	11,4
3	Connect	0,85	106,2	26,6	24,8
	Visibility	9,4	1252,3	353,8	359,3
4	Connect	0,25	4,0	1,0	1,2
	Visibility	0,40	12,5	3,5	4,3
5	Connect	0,23	46,5	13,0	12,9
	Visibility	3,9	180,4	48,7	60,8

die Aussage treffen, dass das *RDT-connect* Verfahren dem *RDT-visibility* Verfahren überlegen ist. Dies gilt nicht nur hinsichtlich der durchgängig niedrigeren Erwartungswerte sondern auch bezüglich der teils erheblich geringeren Standardabweichung. Die Daten des *RDT-visibility* Verfahrens werden in dieser Aufgabe daher nicht näher untersucht.
Die Auflistung der Extremwerte vermittelt einen guten Eindruck, wie stark die Planungsdauer auch unter exakt identischen Planungsvoraussetzungen durch die stochastische Komponente der samplingbasierten Planungsverfahren schwanken kann.
Anhand der Erwartungswerte lässt sich der schon im vorherigen Abschnitt 4.5.1.1 aufgetretene Effekt, dass die Planungsdauer sich durch die Reduzierung um die Freiheitsgrade der Hände verringert, beobachten.
Auch das Weglassen der Freiheitsgrade des Torsos und der Plattform wirken sich negativ auf die Planungsdauer aus (vgl. Ergebnisse Kinematiken 2 zu 3 und 4 zu 5). Dies lässt sich mit Hilfe der Analyse der Konfigurationsräume mit abstrakten Hindernissen, siehe Abschnitt 4.4.3.1, erklären. Wie auch dort in der Umgebung mit den beiden waagrechten Platten angedeutet, werden die Nachteile, die durch den größeren Konfigurationsraum entstehen durch die Vorteile einer besseren Konnektivität überwogen.
Abschließend lässt sich zu dieser Aufgabe bemerken, dass sich die beste Kinematik durch das Festsetzten der Freiheitsgrade des linken Arms erreichen lässt. Dies ist schlüssig, da der linke Arm zum Lösen der Aufgabe nicht relevant ist, weil er sich in der gemeinsamen Start- und Zielkonfiguration in einer an den Roboter anliegenden Position befindet.

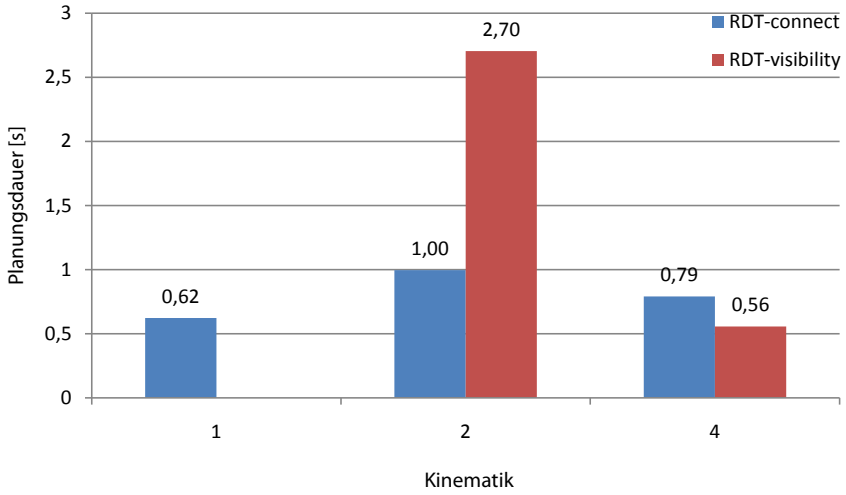


Abbildung 4.7: Planungsdauer bei Aufgabe 3 mit verschiedenen Kinematiken

Ergebnisse Aufgabe 3 Die in Planungsaufgabe 3 ermittelten Ergebnisse sind in Diagramm 4.7 dargestellt. Auch hier zeigt sich der Performanzgewinn durch die Freiheitsgrade der Hände im *RDT-connect* Verfahren. Kinematik 1 ist bei dieser Aufgabe Kinematik 4 überlegen. Wie sich schon in Abschnitt 4.3.1 abzeichnet, wirkt sich dieser Effekt umso stärker aus je einfacher die Umgebung ist.
Die besten Planungsergebnisse werden mit dem *RDT-visibility* Verfahren unter der Verwendung von Kinematik 4 erzielt. Dies lässt den Schluss zu, dass die *RDT-visibility* Verfahren vor allem in einfachen Umgebungen eingesetzt werden sollten.

4.5.1.3 Planung mit variabler Kinematik

In beiden vorherigen Kapiteln wurde der Einfluss der Freiheitsgrade von Justin auf die Pfadplanung untersucht. Diese lassen sich in zwei zentrale Punkte zusammenfassen:
Zunächst gibt es keine ideale Kinematik, die angestrebt werden soll, sondern je nach Planungsproblem wirken sich bestimmte Freiheitsgrade positiv oder negativ auf die Planungsdauer aus.
Das zweite Ergebnis, welches aus den Untersuchungen in der Werkstatt Umgebung gewonnen werden kann ist, dass für die meisten praktischen Aufgaben die vollständige Kinematik von Justin benötigt wird.
Um dennoch die Effizienz der Pfadplanung durch eine Anpassung der Freiheitsgrade steigern zu können, wird in dieser Arbeit ein Ansatz entwickelt, wie mit variablen Kinematiken geplant werden kann. Das bedeutet, dass während einer Planung verschiedene Freiheitsgrade nur teilweise aktiv sind.

Da die Freiheitsgrade, wie in Kapitel 2.2.1 ausführlich behandelt, für den Konfigurationsraum und damit für die gesamte Planung die Basis sind, ergeben sich durch eine flexible Handhabung eine Reihe von Problemen, die zunächst gelöst werden müssen.
Konfigurationen verschiedener Kinematiken sind zueinander nicht kompatibel, da in ihnen nur die Werte der aktiven Freiheitsgrade notiert sind. Wird ein Freiheitsgrad statisch gesetzt, ist er in der Konfiguration nicht mehr enthalten und wird als Teil des Robotermodells betrachtet. Für eine Umsetzung von flexiblen Freiheitsgraden durch jeweiliges Laden eines passenden Robotermodells müsste für jede mögliche Wertekombination der statisch gesetzten Freiheitsgrade ein Modell erstellt werden.
Dieses Problem kann durch die Software gelöst werden, indem man ermöglicht, die starren Teile des Roboters zu manipulieren. Dadurch entsteht das Problem, dass nicht mehr klar unterschieden werden kann, was wirklich starre Teile sind und welche statisch gesetzte Freiheitsgrade sind. Auch bleibt das Problem bestehen, dass mit Konfigurationen unterschiedlicher Dimension und Bedeutung umgegangen werden muss.
Aufgrund dieser Probleme wird in dieser Arbeit ein anderer Ansatz für den Umgang mit flexiblen Freiheitsgraden entwickelt. Eine Reduzierung der Freiheitsgrade ist im mathematischen Sinn als eine Beschränkung des Konfigurationsraums auf eine Hyperebene zu sehen. Um bestimmte Freiheitsgrade statisch zu setzten, darf nur in der entsprechenden Hyperebene geplant werden. Dies kann durch zwei Maßnahmen erreicht werden, da alle samplingbasierten Pfadplaner neue Konfigurationen nur durch Sampling oder Selektieren von Konfigurationen auf Verbindungslinien zwischen bestehenden oder gesampleten Konfigurationen gewinnen. Zum einen müssen alle bestehenden Konfigurationen, die in die Planung mit einbezogen werden, in dieser Hyperebene liegen. Außerdem darf nur in dieser Hyperebene gesampled werden.
In der Praxis geht es allerdings weniger darum, wirklich Freiheitsgrade statisch zu setzten als viel mehr, einen Performanzgewinn durch die Fokussierung auf bestimmte Freiheitsgrade zu erreichen. Dafür ist die Beschränkung des Samplings ausreichend. Es kann daher auch mit beliebigen Konfigurationen geplant werden. Ein weiterer Vorteil ist, dass ein Robotermodell mit den vollen Freiheitsgraden für jegliche kinematische Beschränkung eingesetzt werden kann.
Für den Einsatz der variablen Freiheitsgrade muss die Planung in verschiedene Teile zerlegt werden, die dann mit der jeweils optimalen Beschränkung des Konfigurationsraums bearbeitet werden können. Wie diese Unterteilung vorgenommen wird, hängt von dem Wissen über die Aufgabe und die Umgebung ab und wird im nächsten Kapitel untersucht.

4.5.2 Pfadplanung mit Vorwissen

Bei dem Einsatz des Robotersystems in einer Werkstatt liegt Wissen über das Szenario vor, welches für die Planung nützlich sein könnte. Dabei spielen sowohl die Kenntnisse über Justin und die Umgebung als auch das Wissen über das Aufgabenschema eine Rolle. In diesem Abschnitt soll untersucht werden, wie vorhandenes Wissen in die Pfadplanung integriert werden kann.
Ein Ansatz ist, wie schon in Abschnitt 4.5.1.3 erwähnt, die gesamte Planungsaufgabe in kleinere Teile zu zerlegen und diese separat zu lösen. Diese Technik wird intuitiv auch vom Menschen eingesetzt. Zum Beispiel wird niemand mit dem Auftrag, etwas abzuholen sich sofort überlegen, was er mit seinen Armen macht. Zunächst wird man zu dem Gegenstand laufen und dann erst den Griff planen. In Abschnitt 4.5.2.1 wird gezeigt, wie dieser Ansatz im Werkstatt Szenario umgesetzt werden kann.
Eine andere Methode, die Planung effizienter zu gestalten, ist, die sich nicht verändernden aber regelmäßig vorkommenden Planungsaufgaben, wie z.B. das Erreichen bestimmter Positionen im Raum, abzuspeichern und ohne Planungsaufwand wieder aufrufen zu können. Wieder ist das menschliche Verhalten ein Vorbild, bei dem häufig wiederholte Aufgaben auch blind und ohne Überlegen durchgeführt werden können. Die hierzu entwickelten Verfahren werden in Abschnitt 4.5.2.2 vorgestellt.

4.5.2.1 Unterteilung von Planungsaufgaben

Für eine Unterteilung der Planung müssen zunächst die Aufgabenstruktur und die daraus resultierenden Schwierigkeiten für die Pfadplanung analysiert werden. Dies soll hier exemplarisch für die Planungsaufgabe 4, siehe Abschnitt 4.5.1.2, durchgeführt werden.
Die Aufgabe ist, wie in allen untersuchten Fällen, einen Gegenstand zu greifen. Daraus resultiert, dass in der Nähe der Zielkonfiguration $\mathcal{C}$ Hindernisse aufweist, die Planung dort also schwieriger wird. Insbesondere die Freiheitsgrade der Hände spielen in der Nähe der Zielkonfiguration eine wichtige Rolle. Durch eine relativ geringe Bewegung des Arms kann die direkte Kollision der Finger mit dem zu greifenden Gegenstand verhindert werden. Bei der Startkonfiguration sind in der Regel keine nahen Hindernisse zu erwarten, sodass die Planung in zwei Teile zerlegt werden kann. Der erste Teil dient dazu, den Roboter in die Nähe des zu greifenden Gegenstandes zu bringen. Im zweiten Teil muss die Verbindung zwischen dieser Zwischenkonfiguration und der Zielposition hergestellt werden. Dabei muss der Roboter physikalisch nicht mehr weit bewegt werden, um einen kollisionsfreien Zugang zu der nahen Zielposition zu finden.
Für die Wahl der Zwischenkonfiguration gelten zwei Kriterien. Zum einen

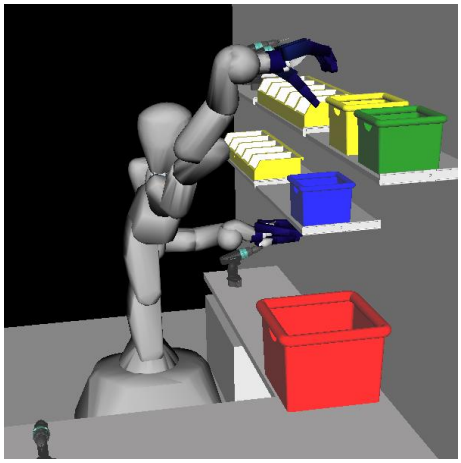


Abbildung 4.8: Zwischenkonfiguration für Planungsaufgabe 4

muss diese Konfiguration physikalisch so weit vom Gegenstand entfernt sein, dass durch die Freiheitsgrade der Hand keine Kollisionen mit diesem entstehen können. Andererseits soll die Zwischenkonfiguration dem Ziel im Konfigurationsraum besonders bezüglich der Freiheitsgrade, die nicht die Hand definieren, sehr nahe sein.
Im vorliegenden Beispiel wird eine solche Konfiguration erzeugt, indem auf den Wert des Schultergelenks des rechten Arms in der Zielkonfiguration ein Betrag von etwa 17 Grad addiert wird. Damit ist diese Konfiguration immer noch sehr nahe der Zielkonfiguration, eine Kollision mit dem Gegenstand kann aber durch die fast senkrechte Lage des Arms vermieden werden, siehe Abbildung 4.8.

Ergebnisse Aus den Simulationsergebnissen in Tabelle 4.8 lässt sich die Effizienz dieser Methode ablesen. Wird für jeden Teil das jeweils schnellste Verfahren eingesetzt, so ergibt sich für den Erwartungswert der Gesamtplanungsdauer ein Wert von unter einer Sekunde. Bei der in Kapitel 4.5.1.2 ermittelten Planungsdauer ohne Unterteilung lag der Erwartungswert bei 33 Sekunden. Selbst bei einem Aufeinandertreffen der beiden größten mit diesen Verfahren ermittelten Planungsdauern ergibt sich ein Gesamtwert von deutlich unter fünf Sekunden.

4.5.2.2 Invariante Planungskomponenten

Bei vielen Objekten in der Werkstatt kann davon ausgegangen werden, dass sie sich nicht verändern. Der Tisch behält in der Regel seinen Platz, genauso wie die an die Wand geschraubten Regale. Mit der zusätzlichen Annahme, dass der Bereich vor dem Tisch freigehalten wird, ist der Großteil des physikalischen Raums bekannt. Veränderungen, und damit Unsicherheiten, sind

Tabelle 4.8: Ergebnisse der Planungsaufgabe 4 mit verschiedenen Kinematiken und Planungsverfahren

Teil	Verfahren	DoF	$min(\mathbf{X})[s]$	$max(\mathbf{X})[s]$	$E(\mathbf{X})[s]$
1	RDT-connect	0-42	0,34	3,2	0,89
	RDT-visibility	4-10	0,27	2,0	0,77
	RDT-connect	4-10	0,29	7,3	1,8
	RDT-connect	0-10	0,36	2,3	0,84
	RDT-connect	0-10;18-29	0,20	1,9	0,74
2	RDT-connect	0-10;18-29	0,10	3,3	0,47
	RDT-visibility	4-10;18-29	0,11	0,69	0,21

nur auf den Arbeitsflächen, unter dem Tisch oder in den Regalen zu erwarten.
Damit ist die Grundlage für eine Vorausberechnung der Pfadplanung gegeben. Alle Bewegungen, die ausschließlich im bekannten Raum stattfinden, müssen nur einmal berechnet werden. Werden sie wieder benötigt, kann auf die schon vorliegenden Ergebnisse zurückgegriffen werden.
Für eine Umsetzung dieser Überlegung eignet sich das Prinzip der *Roadmap*. Einmal erstellt, können damit verschiedene Pfadplanungsanfragen gelöst werden, ohne dass dafür eine neue Pfadplanung durchgeführt werden muss. Lediglich die Zeit für die Graphensuche in der bekannten *Roadmap* wird benötigt. Die Tatsache, dass die Erstellung einer *Roadmap* langwierig sein kann und von vielen Parametern abhängt, weshalb diese Verfahren für Justin in dieser Arbeit bisher nicht eingesetzt wurden, verliert an Bedeutung da diese *Roadmap*, einmal erstellt, für alle Pfadplanungen eingesetzt werden kann.
Ein denkbarer Ansatz, die *Roadmap* zu erstellen ist, die potentiell variablen Gebiete durch künstliche Hindernisse einzuschließen und in dem verbleibenden Raum eine so detaillierte *Roadmap* zu erzeugen, dass alle Konfigurationen, die in dieser Umgebung noch frei sind, angefahren werden können. Es bietet sich die Option, die Zwischenkonfigurationen, die im vorherigen Kapitel zur Unterteilung der Planungsaufgabe verwendet wurden, in die *Roadmap* mit einzubinden. Dies ist dann möglich, wenn diese Konfigurationen vollständig außerhalb des variablen Raums liegen. Sinnvoll ist das allerdings nur, wenn mit wenigen Konfigurationen viele potentielle Planungsaufgaben abgedeckt werden können.
Im Folgenden wird untersucht, ob die Planung auch von der Zwischenkonfiguration profitiert, wenn die Box nicht an derselben Position steht. Dazu wurde neben der gelben Box, die in Planungsaufgabe 4 gegriffen werden sollte eine grüne Box in die Umgebung eingefügt. Aus Abbildung 4.9 ist die genaue Position und die Zielkonfiguration zu entnehmen. Da die Position

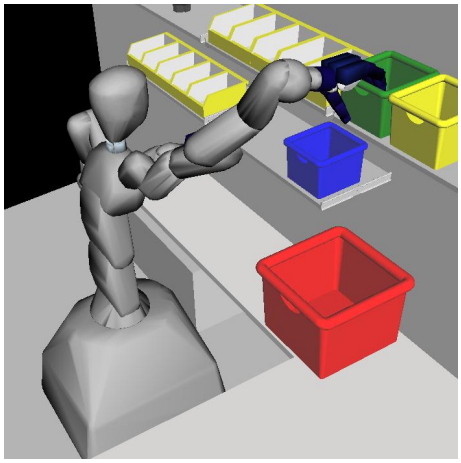


Abbildung 4.9: Greifen verschiedener Boxen von einer Zwischenkonfiguration

des Roboters in der Zielkonfiguration der grünen Box eine andere ist als bei der gelben Box, wird die Planung mit den Freiheitsgraden 0 bis 10 sowie 18 bis 29, siehe Abschnitt 4.2.2, durchgeführt.

Ergebnis Der Erwartungswert der Planungdauer mit dem *RDT-connect* Verfahren von der Zwischenkonfiguration zu der Greifkonfiguration der grünen Box beträgt 0,27 Sekunden. Die Planung ist also relativ flexibel gegenüber Positionsveränderungen der Box. Es ist daher anzunehmen, dass mit wenigen Konfigurationen der variable Raum für diesen Boxentyp abgedeckt werden kann. Daher ist anzunehmen, dass es sich mit anderen Objekten genauso verhält. Ähneln sich diese, können dieselben Zwischenkonfigurationen verwendet werden.

4.5.3 Planung für Greifkonfigurationsmengen

Bei der Ermittlung von Greifkonfigurationen entstehen oft große Mengen von möglichen Griffen. Diese können nach Güte des Griffs bewertet werden, allerdings lässt dies keine Rückschlüsse auf die Erreichbarkeit dieser Konfigurationen zu. In diesem Abschnitt wird eine Methode vorgestellt, wie die Erreichbarkeit von Griffmengen durch eine Pfadplanung ermittelt werden kann.

Methode Der einfachste Ansatz ist, die verschiedenen Greifkonfigurationen nacheinander durch ein *Single-Query* Verfahren zu testen. Dazu wird für jede Konfiguration ein Pfad von der Startkonfiguration aus geplant. Je

nach Zielsetzung kann dabei nach dem ersten gefundenen Pfad abgebrochen werden oder nach Alternativen gesucht werden. Der Nachteil ist, dass, wenn nicht zufällig am Anfang eine gut zu erreichende Konfiguration überprüft wird, viel Zeit verloren gehen kann. Sollen alle Griffe überprüft werden, müssen für den gleichen Raum immer wieder neue Planungen durchgeführt werden obwohl die Greifkonfigurationen aufgrund ihrer Nähe im physikalischen Raum zumindest mit hoher Wahrscheinlichkeit auch in gemeinsamen Bereichen des Konfigurationsraums zu finden sind.
Um Mehrfachplanungen zu vermeiden, ist ein anderer Ansatz die Erzeugung einer *Roadmap*, welcher die verschiedenen Greifkonfigurationen hinzugefügt werden können. Der Nachteil ist, dass die gängigen *Roadmap* Verfahren den *RDT* Verfahren in Umgebungen mit vielen Hindernissen unterlegen sind. Da es sich bei dieser Aufgabenstellung um Planungen direkt an Objekten, also Hindernissen handelt, werden die *Roadmap* Verfahren eine vergleichsweise schlechte Performanz aufweisen.
Für diese Aufgabenstellung wurde ein Planungsverfahren entwickelt, welches die Vorteile dieser beiden Planungsklassen in diesem Zusammenhang vereint. Das Verfahren, mit vielen *RDT*s zu planen, die sich wie Komponenten einer *Roadmap* verhalten, wird im weiteren Verlauf *RDTPRM* genannt. Wenn zwei *RDT*s miteinander verbunden werden können, vereinen sich diese zu einem großen *RDT*. Als Wurzel für die *RDT*s werden dabei die verschiedenen Greifkonfigurationen verwendet. Liegen diese im Konfigurationsraum eng zusammen, verbinden sie sich schnell und die Planung muss nur einmal durchgeführt werden. Ein weiter Vorteil ist, dass durch die parallele Planung die Gefahr, bei einer schwer erreichbaren Konfiguration Zeit zu verlieren nicht auftritt. So kann die Planung beendet werden sobald der erste *RDT* die Startkonfiguration erreicht. Außerdem können mit einem *RDT* oft mehrere Greifkonfigurationen erreicht werden, was die Möglichkeit eröffnet, weitere Kriterien wie z.B. die Griffqualität in die Wahl der anzufahrenden Konfiguration mit einzubeziehen.
Im Folgenden wird diese Methode an einem Szenario, in dem ein Zylinder gegriffen werden soll, evaluiert. Dazu wurde von einer Startkonfiguration zu einer Zielkonfigurationsmenge von 15 Greifkonfigurationen geplant. In den Abbildungen 4.10a und 4.10b sind die Startkonfiguration sowie eine Griffkonfiguration mit zwei Akkuschraubern als zusätzliche Hindernisse zu sehen.
Für die Analyse der vorgestellten Methode werden Planungen mit drei Verfahren betrachtet. Als Orientierung wird eine Planung von der Startkonfiguration zu einer bestimmten Greifkonfiguration mit dem *RDT-connect* Verfahren durchgeführt. Des Weiteren kommt das *RDTPRM* Verfahren zum Einsatz, zum einen in der Variante, in der alle Greifkonfigurationen mit der Ausgangskonfiguration verbunden werden, und andererseits, wenn nach dem ersten verbunden *RDT* die Planung abgebrochen wird, im Folgenden

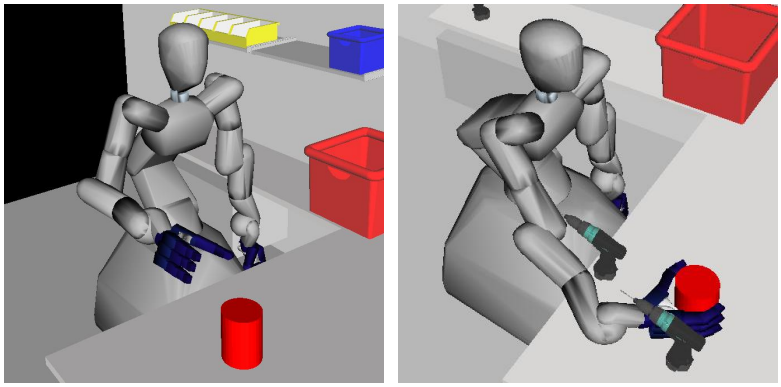


Abbildung 4.10: Planung mit Greifkonfigurationsmengen

als *RDTPRM-connect-break* bezeichnet.
Um den Einfluss von Hindernissen zu untersuchen, wird in den verschiedenen Planungen entweder ohne Hindernisse, mit einem Akkuschrauber neben dem Zylinder oder mit noch einem Weiteren direkt vor dem Gegenstand gearbeitet. Dabei werden von dem Hindernis neben dem Zylinder vier Konfigurationen blockiert. Wird auch noch der zweite Akkuschrauber hinzugefügt, fallen fünf weitere Greifkonfigurationen weg.

Ergebnisse In Tabelle 4.9 sind die Ergebnisse dieser Simulationen aufgeführt. In der ersten Zeile ist die Planung für nur eine Greifkonfiguration angegeben. Da die Planungszeit hier von der jeweiligen Greifkonfiguration abhängt, ist der ermittelte Erwartungswert nur als eine grobe Abschätzung einzustufen. Der gewählte Griff ist relativ gut zu erreichen, es kann daher davon ausgegangen werden, dass die Planungsdauer bei anderen Konfigurationen erheblich höher ist.
Die nächsten beiden Zeilen der Tabelle führen die Ergebnisse des *RDTPRM-connect* Verfahrens ohne Hindernis und mit einem Akkuschrauber an. Es wurde für jede erreichbare Greifkonfiguration ein Pfad ermittelt. Bei der Planung in der freien Umgebung zeigt sich, dass der Erwartungswert hier aufgrund der Vermeidung von Mehrfachplanungen deutlich unter dem 15-fachen der Einzelplanung liegt. Wird die Anzahl der Greifkonfigurationen erhöht, und liegen diese näher zusammen, so ist zu erwarten, dass sich dieser Effekt verstärkt.
In der Praxis ist selten der Pfad zu allen Greifkonfigurationen gefragt, sondern es soll in möglichst kurzer Zeit eine ausreichend gute Greifkonfiguration erreicht werden. Die letzten beiden Ergebnisse sind die Erwartungswerte der Planungsdauer, wenn nach dem ersten *RDT*, welcher die Startkonfiguration erreicht, abgebrochen wird. Die ermittelten Werte hierfür lie-

Tabelle 4.9: Ergebnisse für Planungen mit Zielkonfigurationsmengen

Verfahren	Hindernisse	$min(\mathbf{X})[s]$	$max(\mathbf{X})[s]$	$E(\mathbf{X})[s]$
1 Griff RDT-connect	keine	0,13	1,2	0,36
RDTPRM-connect	keine	2,1	7,8	4,2
RDTPRM-connect	1 Schrauber	2,8	170,2	29,6
RDTPRM-connect-break	1 Schrauber	0,56	2,5	1,1
RDTPRM-connect-break	2 Schrauber	0,27	1,8	0,82

gen in einer Größenordnung, welche auch für die Planung einer schwerer erreichbaren einzelnen Zielkonfiguration denkbar wäre. Die Planung mit mehr Hindernissen weist in diesem Fall einen geringeren Erwartungswert auf, da durch direkte Kollisionen für viele Greifkonfigurationen nicht mehr geplant werden muss.

Kapitel 5

Zusammenfassung und Ausblick

In dieser Arbeit wurden für zwei verschiedene Anwendungsgebiete samplingbasierte Methoden entwickelt. Die dafür durchgeführten Schritte werden in den beiden folgenden Abschnitten zusammengefasst. Ebenso werden die jeweils ermittelten Ergebnisse kurz diskutiert und mit einem Ausblick auf weitergehende Arbeiten abgeschlossen. In Abschnitt 5.3 wird eine allgemeine Zusammenfassung dieser Arbeit gegeben und durch eine Skizzierung der möglichen weiteren Schritte ergänzt.

5.1 Diskussion der Planung mit dem Hand-Auge-System

Für die Planung in einem Explorationszenario wurde eine umfangreiche Literaturrecherche zu Explorationsverfahren durchgeführt. Mit der Pfadplanungsbibliothek *SamPP* wurde eine Pfadplanung für das Robotersystem entwickelt und in der Simulation überprüft. Zur Integration des Planers in die Explorationssoftware wurde eine kompatible Version von *SamPP* erstellt. Auf dieser Basis wurden zwei Ansätze, die Trajektorienplanung und die *View-Point* Kandidatenerzeugung entwickelt und sowohl mit der Pfadplanung als auch im Einsatz in der Explorationssoftware evaluiert.
Die Ergebnisse der Trajektorienplanung lassen die Aussage zu, dass diese ein nützliches Werkzeug für die Exploration in Arbeitszellen ist. Eine Trajektorie, die hohen Informationsgewinn erwarten lässt, kann mit der vorgestellten Methode schnell und in der Umgebung flexibel eingesetzt werden.
Weitere Arbeiten könnten sich mit der Entwicklung geeigneter Trajektorien beschäftigen. Denkbar wäre es auch, ein festes Gitter von Konfigurationen in der Arbeitszelle zu definieren, welches mit Hilfe eines Pfadplanungsverfahrens der Umgebung entsprechend auf dem kürzesten Weg abgefahren wird. Somit würde die Trajektorienerstellung implizit von der Pfadplanung übernommen werden.

Bei den Methoden für das *View-Point* Kandidaten-Sampling ist die einfachste Variante die Beste. Dies legt den Schluss nahe, das die Thesen, die für die komplexeren Methoden gemacht wurden in dieser Umgebung nicht für die Performanz der *View-Point* Planung ausschlaggebend sind sondern, dass in diesem Fall eher von Hindernissen entfernte Konfigurationen einen höheren Informationsgewinn erwarten lassen.
Um die Ursache für diesen Effekt zu erforschen, muss die Explorationsstrategie hinsichtlich der Auswirkung verschiedener *View-Point* Kandidaten in Abhängigkeit des Roboters und der Umgebung genauer untersucht werden. Da dies nicht mehr im Bereich der Pfadplanung liegt, ist das Gegenstand weiterer Arbeiten.

5.2 Diskussion der Planung mit dem humanoiden Robotersystem Justin

Die mit dem humanoiden Roboter Justin durchgeführten Planungsaufgaben sind ebenfalls mit einem auf *SamPP* basierenden, für diese Arbeit entwickelten Planer durchgeführt worden. Da die Kinematik, und damit der Konfigurationsraum von Justin durch die hohe Anzahl an Freiheitsgraden sehr komplex ist, wurden Methoden entwickelt, diesen Planungsraum anhand von verschiedenen Kriterien zu analysieren. Auf dieser Basis und mit der Auswertung von Simulationen in abstrakter Umgebung wurden die für die Manipulationsaufgaben geeigneten Planungsverfahren ausgewählt. Es wurde im Folgenden eine Methode entwickelt, mit der für die Planung grundlegenden Größe der Freiheitsgrade variabel umzugehen. Außerdem wurden Ansätze herausgearbeitet, mit denen das Wissen über das Szenario in den Planungsprozess mit eingebracht werden kann. Abschließend wurde ein Verfahren speziell für die Schnittstelle zu der Greifplanung entwickelt und evaluiert.
Die Ergebnisse zeigen, dass mit dem für diese Arbeit implementierten *RDT-connect* Verfahren schnelle Lösungen für viele Manipulationsaufgaben gefunden werden können. Durch den jeweils auf die Aufgabe zugeschnittenen Einsatz der Freiheitsgrade kann die Effizienz der Planung weiter gesteigert werden. Komplexe Probleme lassen sich durch den Einbezug von Wissen über das Szenario in einfachere Teilplanungen zerlegen, die dennoch flexibel auf die Umgebung reagieren können. Einige Planungsabschnitte können durch eine Analyse des Szenarios als invariant bezeichnet werden. Für diese Teile ist es möglich, auf gespeicherte Planungsergebnisse zurückzugreifen um die Geschwindigkeit und Zuverlässigkeit der Gesamtplanung zu erhöhen.
Zusammenfassend lässt sich zu den Ergebnissen sagen, dass durch eine

Kombination der verschiedenen entwickelten Methoden Pfadplanungsprobleme im betrachteten Szenario in unter einer Sekunde zu lösen sind. Dies ist hinsichtlich der Tatsache, dass das Robotersystem 43 Freiheitsgrade aufweist ein sehr guter Wert. Auch für den praktischen Einsatz ist die Planungsdauer damit in einem akzeptablen Bereich.
Wegen des breiten Ansatzes konnten die entwickelten Methoden nur stichprobenartig untersucht werden. Weitere Arbeiten könnten sich auf bestimmte Teile des in dieser Arbeit erstellten Pfadplaners für Justin konzentrieren. Ein interessanter Anknüpfpunkt ist dabei der Effekt der Leistungssteigerung durch den Einbezug der Freiheitsgrade der Hände. Weitere Themen sind, die Unterteilung des Pfades mit der Aufgabenplanung zu verknüpfen oder eine Greifplanung mit der Pfadplanung zu kombinieren.

5.3 Allgemeine Zusammenfassung und Ausblick

Für die Arbeit als Ganzes lässt sich sagen, dass das Ziel, auf Anwendungen ausgerichtete Pfadplanungsmethoden zu entwickeln, erreicht wurde. Durch die Simulationen konnte die Effizienz der verschiedenen Ansätze belegt werden, sodass durch diese Arbeit eine nützliche Grundlage für den Einsatz von samplingbasierten Verfahren in diesen Szenarien gelegt werden konnte.
Der nächste Schritt, der Thema weiterer Arbeiten sein könnte, ist die Übertragung dieser Methoden auf ein reales Robotersystem und damit einer Verknüpfung der Pfadplanung mit den dafür relevanten Komponenten wie Bildverarbeitung, Explorationssoftware, Aufgabenplanung und der Robotersteuerung.

Abbildungsverzeichnis

Literaturverzeichnis

[1] C. Borst, M. Fischer, and G. Hirzinger. Efficient and precise grasp planning for real world objects. *Multi-point Interaction with Real and Virtual Objects, ser. Springer Tracts in Advanced Robotics, F. Barbagli, D. Prattichizzo, and K. Salisbury, Eds. Springer*, 18:91–111, 2005.

[2] J. Butterfass, M. Grebenstein, H. Liu, and G. Hirzinger. DLR-Hand II: next generation of a dextrous robot hand. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 109–114 vol.1, 2001.

[3] J.J. Craig. *Introduction to Robotics Mechanics and Control*. Addison-Wesley, 1989.

[4] J. Denavit and RS Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Trans. ASME J. Appl. Mech*, 22(1):215–221, 1955.

[5] Andreas Dömel. Entwicklung eines Pfadplaners für einen Virtual Reality Versuchsstand. Technical report, 2007.

[6] Andreas Dömel. Inbetriebnahme, Weiterentwicklung und Evaluation einer Pfadplanungs-Bibliothek an einem VR-Versuchsstand. Technical report, Technische Universität München, Lehrstuhl für Steuerungs und Regelungstechnik, 2008.

[7] M. Fuchs, C. Borst, P.R. Giordano, A. Baumann, E. Kraemer, J. Langwald, R. Gruber, N. Seitz, G. Plank, K. Kunze, R. Burger, F. Schmidt, T. Wimboeck, and G. Hirzinger. Rollin' justin - design considerations and realization of a mobile platform for a humanoid upper body. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 4131–4137, May 2009.

[8] R. Geraerts and M.H. Overmars. A comparative study of probabilistic roadmap planners. *Algorithmic Foundations of Robotics V*, Springer Tracts in Advanced Robotics:43–57, 2004.

[9] G. Hirzinger, N. Sporer, A. Albu-Schaffer, M. Hahnle, R. Krenn, A. Pascucci, and M. Schedl. Dlr's torque-controlled light weight robot iii-are we reaching the technological limits now? In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 2, pages 1710–1716 vol.2, 2002.

[10] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, Aug 1996.

[11] E. Kruse, R. Gutsche, and F.M. Wahl. Efficient, iterative, sensor based 3-d map building using rating functions in configuration space. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 2, pages 1067–1072 vol.2, Apr 1996.

[12] J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 995–1001, 2000.

[13] KUKA Roboter GmbH. www.kuka-robotics.com, Februar 2010.

[14] S. M. LaValle. Rapidly-Exploring Random Trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University, Oct. 1998.

[15] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[16] Steven M. Lavalle, James J. Kuffner, and Jr. Rapidly-Exploring Random Trees: Progress and Prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308, 2000.

[17] C. Nissoux, T. Simeon, and J.-P. Laumond. Visibility based probabilistic roadmaps. In *Intelligent Robots and Systems, 1999. IROS '99. Proceedings. 1999 IEEE/RSJ International Conference on*, volume 3, pages 1316–1321 vol.3, 1999.

[18] C. Ott, O. Eiberger, W. Friedl, B. Bauml, U. Hillenbrand, C. Borst, A. Albu-Schaffer, B. Brunner, H. Hirschmuller, S. Kielhofer, R. Konietschke, M. Suppa, T. Wimbock, F. Zacharias, and G. Hirzinger. A humanoid two-arm system for dexterous manipulation. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 276–283, Dec. 2006.

[19] R.P. Paul. *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. MIT, 1981.

[20] Claude Elwood Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[21] D. Stoyan and W. S. Kendall. *Stochastic Geometry and Its Applications*. J.Wiley, 1995.

[22] M. Suppa, S. Kielhofer, J. Langwald, F. Hacker, K.H. Strobl, and G. Hirzinger. The 3d-modeller: A multi-purpose vision platform. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 781–787, April 2007.

[23] M. Suppa, Pengpeng Wang, K. Gupta, and G. Hirzinger. C-space exploration using noisy sensor models. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 5, pages 4777–4782 Vol.5, April-1 May 2004.

[24] Michael Suppa. *Autonomous Robot Work Cell Exploration Using Multisensory Eye-in-Hand Systems*. PhD thesis, Universität Hannover, 2007.

[25] Gino van den Bergen. *Collision Detection in Interactive 3D Environments*. Morgan Kaufmann Publisher, 2004.

[26] A. Yershova and S. M. LaValle. Motion planning for highly constrained spaces. Technical report, Department of Computer Science, University of Illinois, 2008.

[27] Y. Yu and K. Gupta. An information theoretical approach to view planning with kinematic and geometric constraints. In *PROC IEEE INT CONF ROB AUTOM*, volume 2, pages 1948–1953, 2001.

[28] Yong Yu and Kamal Gupta. C-space entropy: A measure for view planning and exploration for general robot-sensor systems in unknown environments. *The International Journal of Robotics Research*, 23(12):1197–1223, December 2004.

[29] M. Zucker, J. Kuffner, and J.A. Bagnell. Adaptive workspace biasing for sampling-based planners. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3757–3762, May 2008.