

Hochschule Karlsruhe

– Fakultät für Maschinenbau und Mechatronik–

Projektaufgabe Roboterprogrammierung

Mobiler Manipulator

Projektdokumentation zum
Roboterprogrammierung – Wintersemester 25/26

vorgelegt am 26. Januar 2026 von

Paul Glaser, B. Eng (glpa1013 / Matrikelnr. 100215)

Tim Schäfer, B. Eng (scti1057 / Matrikelnr. 71606)

Felix Wietschel, B. Eng (wife1013 / Matrikelnr. 74940)

Erstprüfer : Prof. Dr.-Ing. Björn Hein

Inhaltsverzeichnis

1 Einleitung	1
2 Implementierung und Evaluation	2
2.1 Implementierung des CollisionCheckers	2
2.2 Benchmarking der Planungsverfahren	5
2.3 Diskussion der Ergebnisse	11
3 Pick-And-Place Szenario	12
3.1 Problemzerlegung und Sequenzierung	12
3.2 Algorithmische Umsetzung	13
3.3 Simulation und Ergebnisse	14
4 Theoretische Konzepte und Erweiterungen	20
4.1 Erweiterung um translatorische Gelenke	20
4.2 Optimierung und Glättung von Bewegungsbahnen	21
4.3 Weitere Erweiterungsmöglichkeiten	22
A Anhang	24
A.1 Planungsumgebungen	24
A.2 Ergebnisse Multi-Run MIT Überprüfung der Eigenkollision	26
A.3 Ergebnisse Multi-Run OHNE Überprüfung der Eigenkollision	28
A.4 Ergebnisse Multi-Run Pick-And-Place MIT Überprüfung der Eigenkollision	30
A.5 Ergebnisse Multi-Run Pick-And-Place OHNE Überprüfung der Eigenkollision	32
Literatur	34

Abbildungsverzeichnis

2.1	Schematische Darstellung des mobilen Manipulators mit 5 Freiheitsgraden. Die mobile Basis (blau) verfügt über 3 Freiheitsgrade (x, y, θ_{base}), der Roboterarm mit zwei Segmenten (gelb) über 2 rotatorische Gelenke (θ_1, θ_2). Der Greifer ist am Ende des zweiten Segments montiert.	3
2.2	Vergleich der generierten Roadmaps im Szenario <i>Forest</i> . (a) VisibilityPRM generiert einen Graphen, wobei Knoten primär an Hinderniskanten und in freien Sichtbereichen platziert werden. (b) LazyPRM erzeugt eine gleichverteilte Wolke an Knoten.	6
2.3	Statistische Auswertung für das Szenario <i>Forest</i> (mit Eigenkollisionsprüfung). Oben links: Erfolgsrate (Visibility 100 %, Lazy 50 %). Unten links: Die Planungszeit von VisibilityPRM ist deutlich höher, korreliert aber mit der höheren Robustheit.	8
2.4	Statistische Auswertung für das Szenario <i>Forest</i> (ohne Eigenkollisionsprüfung). Oben links: Erfolgsrate (Visibility 100 %, Lazy 100 %). Unten links: Die Planungszeit von VisibilityPRM bleibt hoch, da die Sichtbarkeitsprüfungen gegen externe Hindernisse dominieren.	10
3.1	Vergleich der Pick-and-Place Sequenz in der Umgebung <i>Empty World</i> . (a) Der VisibilityPRM erzeugt einen spärlichen, geometrisch optimierten Graphen. (b) Der LazyPRM flutet den Freiraum mit Knoten. In beiden Fällen ist die Sequenz: Start → Pick (oben) → Place (unten rechts) → Move (unten links) erfolgreich.	15
3.2	Statistische Auswertung der Pick-and-Place Aufgabe in <i>Empty World</i> (mit Self-Check). Die Erfolgsrate misst das erfolgreiche Absolvieren der gesamten Sequenz.	16
3.3	Statistische Auswertung der Pick-and-Place Aufgabe in <i>Empty World</i> (ohne Self-Check).	18
A.1	Benchmarks / Planungsumgebungen	24
A.2	Ergebnisse Multi-Run mit Überprüfung der Eigenkollision	26
A.3	Ergebnisse Multi-Run ohne Überprüfung der Eigenkollision	28
A.4	Ergebnisse Multi-Run Pick-And-Place mit Überprüfung der Eigenkollision	30
A.5	Ergebnisse Multi-Run Pick-And-Place ohne Überprüfung der Eigenkollision	32

Tabellenverzeichnis

2.1 Kinematische Parameter des Roboterarms	2
2.2 Konfiguration der Planungsverfahren	5

Abkürzungsverzeichnis

LazyPRM Lazy Probabilistic Roadmap Method

VisibilityPRM Visibility Probabilistic Roadmap Method

PRM Probabilistic Roadmap Method

LE Längeneinheit(en)

1 Einleitung

In diesem Projekt wird ein mobiler Manipulator simuliert, der aus einer beweglichen Basis (3 Freiheitsgrade) und einem Arm mit zwei rotatorischen Gelenken besteht (insgesamt 5 Freiheitsgrade). Ziel der Arbeit ist die Implementierung eines robusten Kollisionsprüfers (Collision Checker) sowie die Planung und Evaluierung von kollisionsfreien Bewegungsbahnen unter Verwendung von Sampling-basierten Algorithmen. Im Gegensatz zu exakten Verfahren, die den gesamten Konfigurationsraum explizit berechnen müssten, generieren diese Methoden zufällige Stichproben (Samples) der Roboterpositionen, um probabilistisch eine kollisionsfreie Roadmap (einen Graphen) zu erstellen. In dieser Arbeit werden speziell die Varianten

- **Lazy Probabilistic Roadmap Method (LazyPRM)** und
- **Visibility Probabilistic Roadmap Method (VisibilityPRM)**

betrachtet. Der erste Teil dieses Berichts, **Kapitel 2, Implementierung und Evaluation**, beschreibt die Implementierung des Kollisionsprüfers und die Durchführung von Benchmark-Tests zur Evaluierung der Planungsverfahren in verschiedenen Szenarien. Zusätzlich wird im zweiten Teil, **Kapitel 3, Pick-And-Place Szenario**, die Erweiterung zu einer Pick-And-Place Anwendung vorgestellt sowie die Ergebnisse diskutiert. Im dritten Teil, **Kapitel 4, Theoretische Konzepte und Erweiterungen**, folgen theoretische Konzepte zur Erweiterung des Systems um translatorische Gelenke sowie Möglichkeiten zur Bahnoptimierung.

Die Aufgabe gliedert sich in folgende Teilbereiche:

1. Implementierung eines *CollisionCheckers* für einen planaren Roboter (Basis + 2 Links).
2. Benchmarking verschiedener Planungsverfahren (LazyPRM vs. VisibilityPRM) in unterschiedlichen Szenarien.
3. Durchführung einer Pick-and-Place Aufgabe, bei der ein Hindernis gegriffen und transportiert wird.
4. Theoretische Betrachtung von Systemerweiterungen (translatorische Gelenke und Pfadglättung).

Der vollständige Quellcode dieses Projekts ist im entsprechenden Git-Repository hinterlegt: https://github.com/Paulchenn/Mobiler_Manipulator. Für detaillierte Anweisungen zur Installation und Verwendung wird auf die dort befindliche README-Datei verwiesen.

2 Implementierung und Evaluation

2.1 Implementierung des CollisionCheckers

Die Basis für die Pfadplanung bildet die korrekte kinematische und geometrische Modellierung des Robotersystems sowie eine effiziente Kollisionsprüfung. Die Validierung einer Roboterkonfiguration erfolgt durch die Klasse CollisionChecker. Diese nutzt die Bibliothek shapely für geometrische Operationen.

2.1.1 Modellierung des Roboters

Der betrachtete mobile Manipulator besteht aus einer beweglichen Basis und einem seriellen Roboterarm. Die geometrische und kinematische Definition erfolgt parametrisch, um eine flexible Anpassung an verschiedene Szenarien zu gewährleisten. Abbildung 2.1 zeigt eine schematische Darstellung des hier betrachteten Roboters.

Geometrische Definition

Die Geometrie wird in *IPTestSuite.py* durch folgende Parameter definiert:

- **Mobile Basis:** Die Basis ist als Rechteck modelliert (Shape: $[(0, 0), (2, 0), (2, 1), (0, 1)]$). Dies entspricht einer Länge von 2.0 Längeneinheit(en) (LE) und einer Breite von 1.0 LE.
- **Referenzpunkt:** Der lokale Ursprung der Basis („Center“) wird auf die lokalen Koordinaten (1.0, 0.5) festgelegt. Dies entspricht dem geometrischen Mittelpunkt der Basisfläche. Rotationen der Basis erfolgen um diesen Punkt.
- **Arm-Montage:** Der Roboterarm ist an der vorderen Kante der Basis montiert (Offset relativ zum Basis-Nullpunkt: (2.0, 1.0)).

Der Roboterarm setzt sich aus zwei Segmenten und einem Greifer zusammen. Die kinematischen Parameter sind in Tabelle 2.1 aufgeführt.

Tabelle 2.1: Kinematische Parameter des Roboterarms

Komponente	Länge [LE]	Dicke [LE]	Winkelbereich [rad]
Segment 1	0.5	0.1	$[0, \pi]$
Segment 2	1.0	0.1	$[-\pi, \pi]$
Greifer	0.1	-	-

Konfigurationsraum

Da die Basis über drei Freiheitsgrade (x, y, θ_{base}) verfügt und der Arm zwei rotatorische Gelenke aufweist, beschreibt ein **5-dimensionaler Konfigurationsvektor** q den Zustand des Roboters vollständig:

$$q = [x \ y \ \theta_{base} \ \theta_1 \ \theta_2] \quad (2.1)$$

Hierbei beschreiben (x, y, θ_{base}) die Pose der Basis im Weltkoordinatensystem, während θ_1 und θ_2 die relativen Gelenkwinkel des Arms beschreiben.

Schematische Darstellung

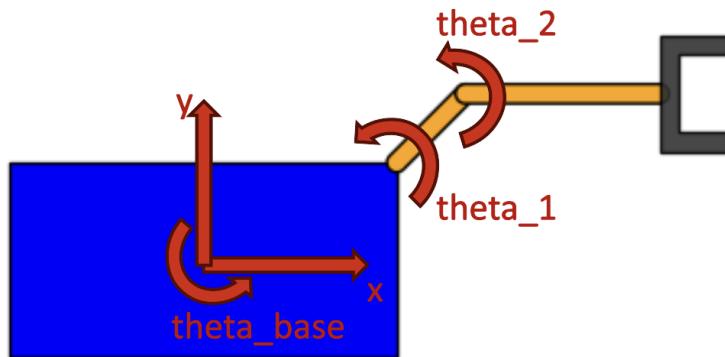


Abbildung 2.1: Schematische Darstellung des mobilen Manipulators mit 5 Freiheitsgraden. Die mobile Basis (blau) verfügt über 3 Freiheitsgrade (x, y, θ_{base}), der Roboterarm mit zwei Segmenten (gelb) über 2 rotatorische Gelenke (θ_1, θ_2). Der Greifer ist am Ende des zweiten Segments montiert.

2.1.2 Kollisionsprüfung

Die Validierung einer Roboterkonfiguration erfolgt durch die Klasse `CollisionChecker`. Diese nutzt die Bibliothek `shapely` für geometrische Operationen und Schnitttests. Die Methode `pointInCollision` prüft die Gültigkeit einer Konfiguration q in einem hierarchischen Ablauf, um rechenintensive geometrische Operationen zu minimieren.

Ablauf der Gültigkeitsprüfung

Der Algorithmus prüft die Kriterien in folgender Reihenfolge. Sobald ein Kriterium verletzt ist, wird die Konfiguration sofort als ungültig (True) markiert:

1. **Gelenkwinkelgrenzen:** Es wird geprüft, ob die Winkel θ_1 und θ_2 innerhalb der definierten Min/Max-Werte liegen.
2. **Arbeitsraumgrenzen:** Es wird überprüft, ob irgendein Roboterteil die definierten Weltgrenzen (Limits) überschreitet.

3. **Statische Hindernisse:** Jedes Segment des Roboters (Basis, Armsegmente, Greifer und ggf. gegriffenes Objekt) wird gegen die Liste der statischen Hindernisse (obstacles) auf Schnittmengen geprüft.
4. **Eigenkollision:** Abschließend erfolgt die Prüfung auf Kollisionen des Roboters mit sich selbst (siehe folgender Abschnitt).

Behandlung von Eigenkollisionen

Die Eigenkollisionsprüfung (*Self-Collision Check*) stellt sicher, dass der Manipulator sich nicht selbst beschädigt. Diese Prüfung kann optional über das Flag SELF_CHECK (in *IPTestSuite.py*) deaktiviert werden, um bspw. die Performance in Szenarien zu erhöhen, in denen Eigenkollisionen ausgeschlossen sind.

Folgende Konflikte werden detektiert:

- **Arm gegen Basis:** Es wird geprüft, ob eines der Armsegmente in die Fläche der Basis eindringt. Um numerische Ungenauigkeiten an der Kontaktstelle (Montagepunkt) zu tolerieren, wird eine Kollision erst gemeldet, wenn die Schnittfläche einen Grenzwert (`intersect_limit`, standardmäßig 0.002) überschreitet.
- **Greifer gegen Basis:** Der Greifer wird separat gegen die Basis geprüft.
- **Arm gegen Greifer:** Die Armsegmente werden gegen den Greifer geprüft. Hierbei wird das letzte Segment des Arms explizit ausgeschlossen, da der Greifer an diesem befestigt ist.
- **Gegriffenes Objekt:** Befindet sich ein Objekt im Greifer („Pick“-Zustand), wird dieses als temporärer Teil der Roboterkonfiguration betrachtet. Es wird sowohl gegen die Basis als auch gegen alle Armsegmente (außer dem befestigenden letzten Segment) geprüft.

Eine Überprüfung von Interaktionen zwischen benachbarten Armsegmenten entfällt. Konstruktionsbedingt wird davon ausgegangen, dass sich die Segmente überschneidungsfrei aneinander vorbei bewegen können.

2.2 Benchmarking der Planungsverfahren

Zur Evaluierung der implementierten Algorithmen LazyPRM und VisibilityPRM wird eine quantitative Analyse in fünf verschiedenen Simulationsumgebungen durchgeführt. Ziel ist der Vergleich hinsichtlich Erfolgsrate, Pfadqualität und Berechnungseffizienz.

2.2.1 Versuchsaufbau

Die Benchmarks wurden auf einem MacBook Air M2 2022 (mit 16 GB Speicher) durchgeführt. Jedes Szenario wurde in zwei Konfigurationen getestet:

1. **Mit Eigenkollisionsprüfung (Self-Check):** Realistische Simulation, bei der Arm-Basis- und Arm-Greifer-Kollisionen geprüft werden.
2. **Ohne Eigenkollisionsprüfung:** Zur Isolierung der algorithmischen Performance des Planers unabhängig von der Geometrie-Prüfung (siehe Diskussion in Abschnitt 2.2.4).

Die für die Experimente verwendeten Konfigurationsparameter der Planer sind in Tabelle 2.2 aufgeführt.

Tabelle 2.2: Konfiguration der Planungsverfahren

Planer	Parameter	Wert	Beschreibung
VisibilityPRM	ntry	500	Sampling-Versuche für Guards
LazyPRM	Initial Roadmap Size	500	Initiale Abdeckung
	Update Roadmap Size	200	Knoten pro Iteration bei Misserfolg
	k-Nearest	12	Konnektivität
	Max. Iterations	15	Obergrenze für Graphen-Wachstum

Pro Szenario und Planer wurden $N = 10$ Durchläufe (Runs) durchgeführt. Als Abbruchkriterien dienen das Finden eines Pfades oder das Erreichen der maximalen Knotenzahl.

Die fünf Testumgebungen sind in Anhang A.1 dargestellt:

- **Empty World:** Eine hindernisfreie Umgebung zur Messung der Basis-Performance (siehe Abb. A.1a).
- **The Wall:** Eine Wand trennt Start und Ziel, verbunden nur durch eine schmale Türöffnung (siehe Abb. A.1b).
- **Narrow Passage:** Ein langer, enger Korridor, der präzise Bewegungen der Basis erfordert (siehe Abb. A.1c).
- **Forest:** Eine „Clutter“-Umgebung mit vielen verteilten Hindernissen (Säulen), die ein häufiges Umgreifen erfordern (siehe Abb. A.1d).
- **Shelf Reach:** Ein Szenario, bei dem die Basis das Ziel nicht erreichen kann und der Arm das Objekt hinter einem Hindernis greifen muss (siehe Abb. A.1e).

2.2.2 Qualitative Analyse der Roadmaps

Ein visueller Vergleich der erzeugten Graphen (Roadmaps) verdeutlicht die unterschiedlichen Strategien der Algorithmen (siehe Abb. 2.2 für das Szenario *Forest*).

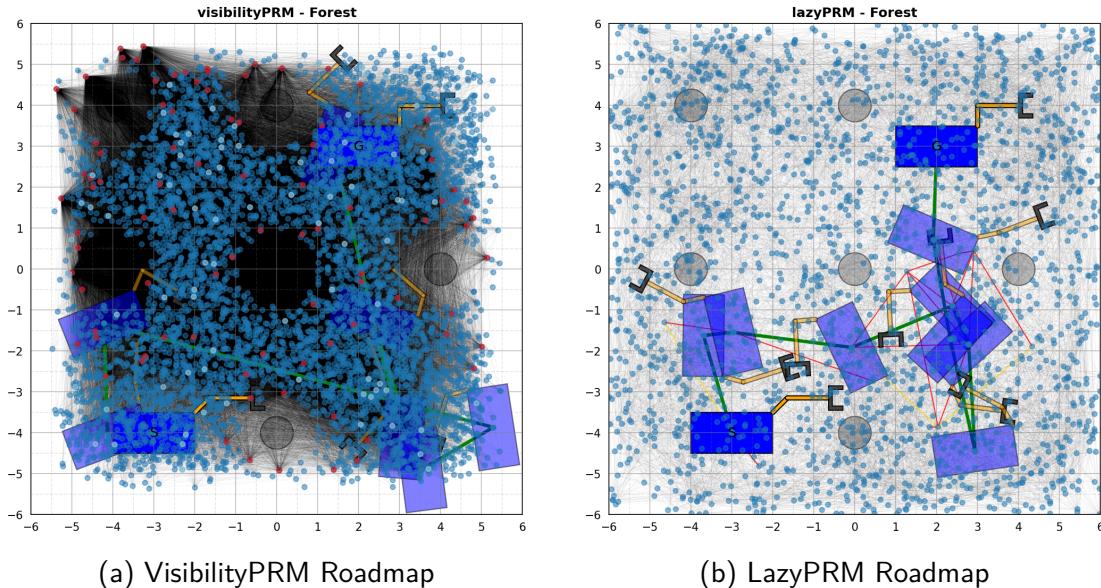


Abbildung 2.2: Vergleich der generierten Roadmaps im Szenario *Forest*. (a) VisibilityPRM generiert einen Graphen, wobei Knoten primär an Hinderniskanten und in freien Sichtbereichen platziert werden. (b) LazyPRM erzeugt eine gleichverteilte Wolke an Knoten.

Es ist zu erkennen, dass LazyPRM den Freiraum mit einer hohen Anzahl an zufälligen Knoten „flutet“. Im Gegensatz dazu konzentriert VisibilityPRM die Knoten in den Bereichen, in denen sich keine Hindernisse befinden (siehe Abb. 2.2a), indem nur solche Konfigurationen zum Graphen hinzugefügt werden, die die Sichtbarkeit erweitern (Guard-Nodes, in rot dargestellt) oder unabhängige Komponenten verbinden (Connector-Nodes, in hellblau dargestellt). Die Knoten, die nicht zur Sichtbarkeit beitragen, werden verworfen (in dunkelblau dargestellt).

2.2.3 Quantitative Ergebnisse (Baseline)

Im ersten Schritt werden die Ergebnisse unter realitätsnahen Bedingungen, demnach **mit aktiver Eigenkollisionsprüfung**, betrachtet.

Erfolgsrate und Robustheit

Die Ergebnisse zeigen eine klare Dichotomie zwischen den beiden Planungsansätzen:

- **Freie Umgebungen:** Im Szenario *Empty World* erreichen beide Planer erwartungsgemäß eine Erfolgsrate von 100 % (siehe Anhang A.2, Abb. A.2a).
- **Engstellen (The Wall, Narrow Passages):** In den Szenarien *The Wall* und *Narrow Passage* offenbart sich die Schwäche des LazyPRM. Während

VisibilityPRM hier konstant 100 % Erfolgsrate erzielt, scheitert LazyPRM vollständig (0%). Dies bestätigt, dass reine Zufalls-Sampling-Strategien ohne heuristische Steuerung statistisch unwahrscheinlich kritische Engstellen im Konfigurationsraum finden (siehe Anhang A.2, Abb. A.2b und A.2c).

- **Komplexe Umgebungen:** Im Szenario *Forest* (viele Hindernisse) zeigt VisibilityPRM seine Stärke mit 100 % Erfolg. LazyPRM bricht hier ein und findet nur in 50 % der Fälle eine Lösung, da der Arm oft in „Sackgassen“ zwischen den Hindernissen gerät (siehe Abb. 2.3 und Anhang A.2, Abb. A.2e).

Das Szenario *Shelf Reach* erweist sich als das anspruchsvollste. Hier kann lediglich VisibilityPRM in 20 % der Fälle eine Lösung finden, während LazyPRM konstant scheitert.

Planungszeit und Effizienz

Bezüglich der Rechenzeit bestätigt sich der theoretische Unterschied der Algorithmen:

- LazyPRM ist in einfachen Szenarien extrem effizient (ca. 0.5 s in *Empty World*). Die Strategie, Kollisionsprüfungen zu verzögern (*lazy*), zahlt sich hier aus.
- VisibilityPRM benötigt signifikant mehr Rechenzeit (Mittelwert ca. 140 s im Szenario *Forest*), da für jeden potenziellen neuen Knoten aufwendige Sichtbarkeitsprüfungen gegen alle Hindernisse durchgeführt werden müssen.

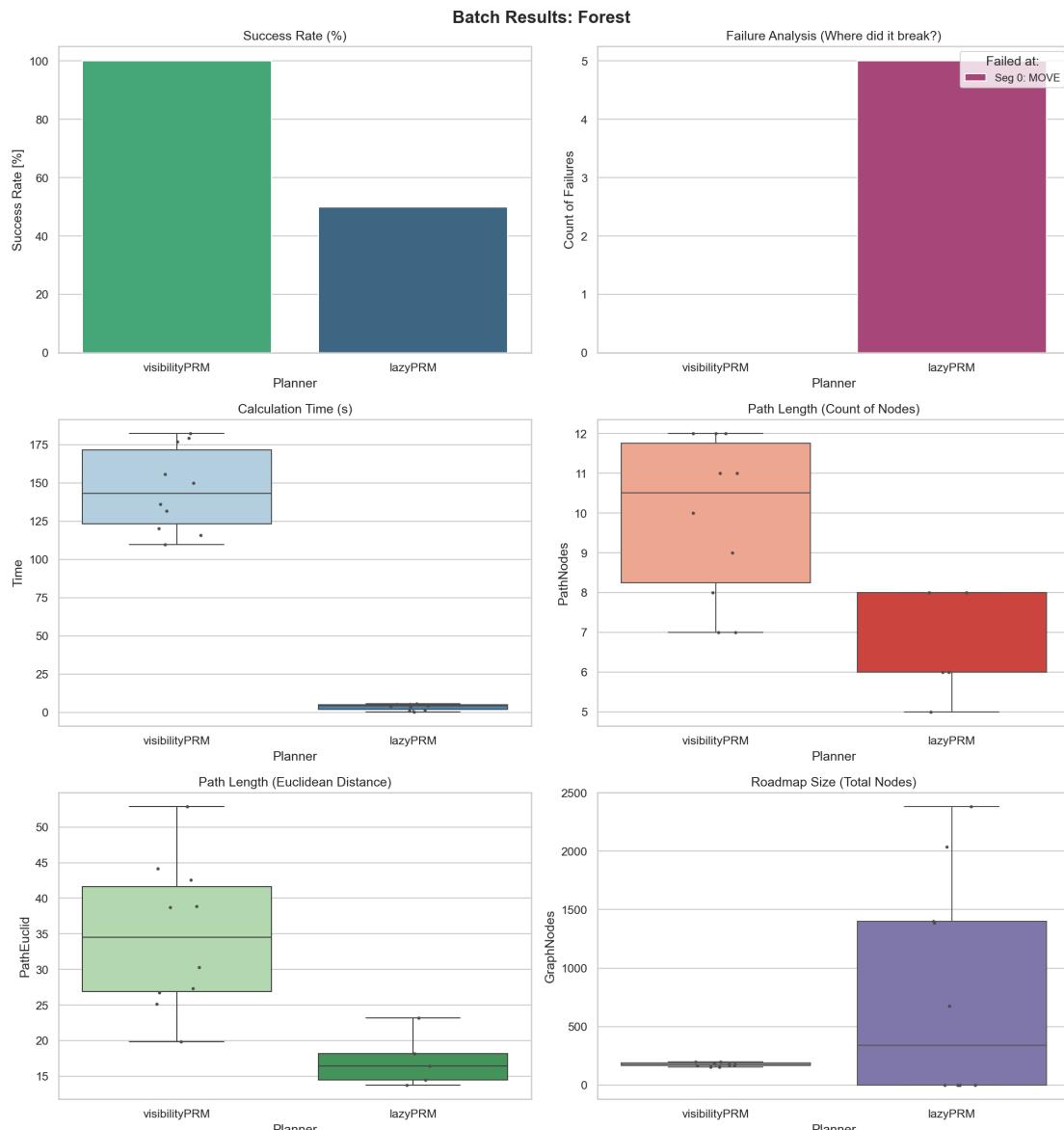


Abbildung 2.3: Statistische Auswertung für das Szenario *Forest* (mit Eigenkollisionsprüfung). Oben links: Erfolgsrate (Visibility 100 %, Lazy 50 %). Unten links: Die Planungszeit von VisibilityPRM ist deutlich höher, korreliert aber mit der höheren Robustheit.

2.2.4 Einfluss der Eigenkollision

Um den Einfluss der geometrischen Komplexität des Roboters auf die Planung zu untersuchen, wurden alle Szenarien erneut **ohne Eigenkollisionsprüfung** durchlaufen. Hierbei wird der Roboter effektiv als „Geist“ behandelt, der sich selbst durchdringen darf. Ein Vergleich der Resultate im Anhang (vgl. Abschnitt A.2 mit A.3) liefert Rückschlüsse auf die Topologie des Konfigurationsraums.

Auswirkung auf die Erfolgsrate

Besonders signifikant ist der Unterschied im Szenario *Forest*:

- **Steigerung bei LazyPRM:** Die Erfolgsrate des LazyPRM stieg von 50 % (mit Self-Check, siehe Abb. 2.3) auf 100 % (ohne Self-Check, siehe Abb. 2.4).
- **Interpretation:** Ohne die Beschränkung der Eigenkollision kann der Planer den Roboterarm durch sich selbst hindurch „falten“, um Hindernissen auszuweichen. Dies vereinfacht die Topologie des Konfigurationsraums erheblich: Bereiche, die zuvor durch die eigene Geometrie blockiert waren (Self-Collision-Regionen), werden passierbar. Der einfache Sampling-Ansatz des LazyPRM profitiert hiervon massiv.

Hingegen zeigt sich in den Engstellen-Szenarien (*The Wall*, *Narrow Passage*), dass die Deaktivierung der Eigenkollision keinen positiven Einfluss auf die Erfolgsrate des LazyPRM hat (verbleibt bei ca. 0 %, vgl. Anhang A.3, Abb. A.2c mit Abb. A.3c).

- **Interpretation:** Das Scheitern in diesen Szenarien wird primär durch die externen Hindernisse (Wände) verursacht. Die „Narrow Passage“ im Konfigurationsraum wird durch die Umwelt definiert, nicht durch die Roboterkonfiguration. Das Entfernen der Eigenkollision ändert nichts daran, dass der Roboter präzise durch die Türöffnung manövrieren muss.

Auswirkung auf die Rechenzeit

Entgegen der Erwartung führt das Abschalten der Eigenkollisionsprüfung bei VisibilityPRM im Szenario *Forest* nicht zu einer signifikanten Reduktion der Rechenzeit (ca. 140 s mit vs. ca. 150 s ohne Prüfung).

- **Analyse:** Dies deutet darauf hin, dass der dominierende Faktor im VisibilityPRM die Anzahl der Sichtbarkeitsprüfungen (Raycasts) gegen die externen Hindernisse ist. Die Prüfung der Roboterkonfiguration selbst (Arm gegen Basis) ist im Vergleich zur Prüfung gegen komplexe Umgebungsgeometrien (viele Säulen im Wald) rechentechnisch weniger ins Gewicht fallend.
- **Schlussfolgerung:** Da die Sichtbarkeitsprüfungen den Großteil der Rechenzeit beanspruchen, könnte beim hierarchischen Aufbau des *CollisionCheckers* die Reihenfolge der Prüfungen optimiert werden. Beispielsweise könnte die Eigenkollisionsprüfung vor den Sichtbarkeitsprüfungen erfolgen, um unnötige Raycasts zu vermeiden, wenn die Konfiguration bereits als ungültig erkannt wurde.

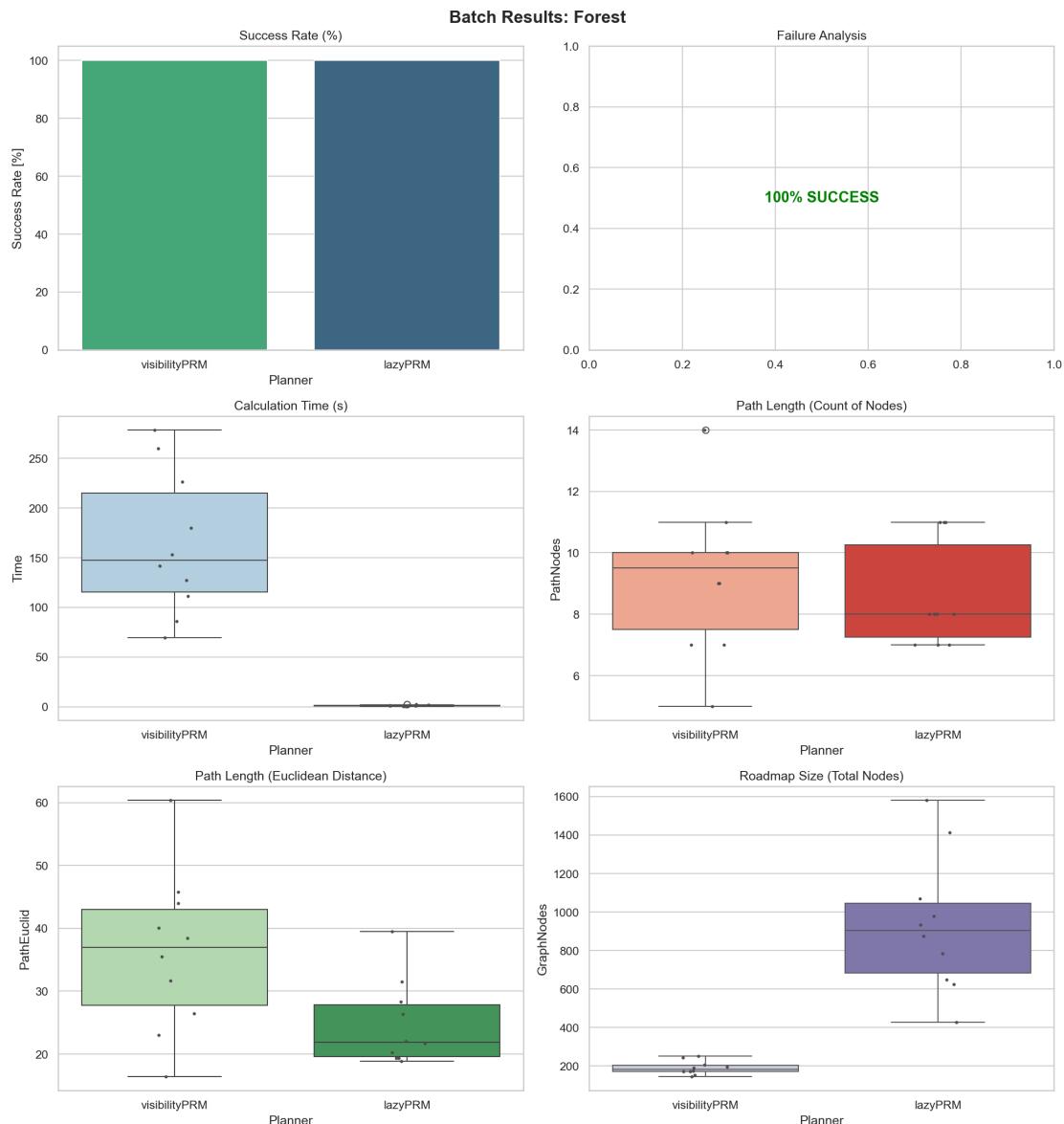


Abbildung 2.4: Statistische Auswertung für das Szenario *Forest* (ohne Eigenkollisionsprüfung). Oben links: Erfolgsrate (Visibility 100 %, Lazy 100 %). Unten links: Die Planungszeit von VisibilityPRM bleibt hoch, da die Sichtbarkeitsprüfungen gegen externe Hindernisse dominieren.

2.3 Diskussion der Ergebnisse

Zusammenfassend kann festgehalten werden, dass VisibilityPRM für den betrachteten mobilen Manipulator die robustere Wahl darstellt, insbesondere in unübersichtlichen oder engen Umgebungen. Die höheren Rechenzeiten werden durch die Garantie gerechtfertigt, auch schwierige Passagen (wie Türen oder enge Korridore) zuverlässig zu finden. LazyPRM eignet sich hingegen hervorragend für offene Areale oder Szenarien mit geringer Hindernisdichte, verliert jedoch an Zuverlässigkeit, sobald komplexe Ausweichmanöver oder die Berücksichtigung von Eigenkollisionen erforderlich sind.

Zur dynamischen Visualisierung der geplanten Pfade steht im Jupyter Notebook (Datei `Mobile_Manipulator_Main.ipynb`) eine Animations-Routine zur Verfügung. Durch Ausführen der entsprechenden Code-Zelle können die Bewegungsabläufe lokal reproduziert werden.

3 Pick-And-Place Szenario

In diesem Kapitel wird die Anwendung der implementierten Planungsverfahren auf eine komplexe Manipulationsaufgabe demonstriert. Ziel ist es, ein Objekt an einer definierten Position aufzunehmen („Pick“), zu transportieren und an einer Zielposition abzulegen („Place“).

3.1 Problemzerlegung und Sequenzierung

Da die verwendeten Sampling-basierten Planer (LazyPRM und VisibilityPRM) primär für Punkt-zu-Punkt-Verbindungen ausgelegt sind, wird die Gesamtaufgabe in eine Sequenz von Teilproblemen zerlegt. Die Ziele der Teilprobleme werden weiterhin durch einen 5-dimensionalen Konfigurationsvektor q (Gl. 2.1) definiert. Dieser wird erweitert mittels der am Zielpunkt zu absolvierenden Aufgabe A_n sowie ggf. einer *Standoff-Konfiguration* q_{standoff} . Die Einzelziele der Teilprobleme werden in der Matrix Q zusammengefasst:

$$Q = \begin{bmatrix} q_1 & A_1 & q_{\text{standoff},1} \\ q_2 & A_2 & q_{\text{standoff},2} \\ \vdots & \vdots & \vdots \\ q_n & A_n & q_{\text{standoff},n} \end{bmatrix} \quad (3.1)$$

Hierbei bezeichnet $A_n \in \{\text{„Pick“}, \text{„Place“}, \text{„Move“}\}$ den Aktionsbefehl am Zielpunkt q_n , wobei „Move“ für eine reine Anfahrtsbewegung ohne Manipulation steht. Der Term $q_{\text{standoff},n}$ definiert die entsprechende Standoff-Position vor dem Zielpunkt.

Letzterer Term gliedert sich in einen Abstand in x - und einen in y -Richtung, vor der Zielposition (siehe Abschnitt 3.2.1):

$$q_{\text{standoff}} = [dx_{\text{standoff}} \quad dy_{\text{standoff}}] \quad (3.2)$$

Die Koordinaten des Abstands der Standoff-Position bezieht sich nur auf die x und y Koordinaten des Roboters. Die Ausrichtung θ_{base} sowie die Stellung der Armgelenke θ_1 und θ_2 ist an der Standoff-Position identisch zur eigentlich in q_n definierten Zielposition. Realisiert wird dies durch die Klasse MultiGoalPlannerRunner (Datei `IPMultiGoalPlannerRunner.py`).

Der Ablauf gliedert sich in folgende Phasen:

1. **Global Move (Start → Pre-Grasp):** Der Planer berechnet einen kollisionsfreien Pfad von der Startkonfiguration zu einer Vor-Greif-Position (Standoff) in der Nähe des Objekts.
2. **Local Approach (Lineare Annäherung):** Der Roboter fährt linear interpoliert von der Standoff-Position zur exakten Greifposition.
3. **Pick (Greifen):** Das Objekt wird logisch an den Endeffektor angehängt und aus der Hindernisliste entfernt.
4. **Retreat (Rückzug):** Der Roboter fährt linear zurück zur Standoff-Position, nun mit dem Objekt am Greifer.
5. **Transport (Pre-Grasp → Pre-Place):** Ein neuer globaler Pfad wird geplant, um das Objekt zur Ablageposition zu bringen.
6. **Place (Ablegen):** Äquivalent zum Greifvorgang erfolgt eine Annäherung, das Loslassen des Objekts (wird wieder zum Hindernis) und der Rückzug.

3.2 Algorithmische Umsetzung

3.2.1 Berechnung der Standoff-Position

Ein direktes Anfahren der Greifposition mittels Probabilistic Roadmap Method (PRM) ist oft nicht zielführend, da der Zufalls-Sampler in unmittelbarer Nähe von Hindernissen (dem zu greifenden Objekt) ineffizient arbeitet. Stattdessen wird eine *Standoff-Konfiguration* q_{standoff} berechnet, die einen definierten Abstand zum Ziel hat.

Gegeben sei die Zielkonfiguration $q_{\text{target}} = [x_t \ y_t \ \theta_t \ \theta_{1t} \ \theta_{2t}]$ und ein lokaler Offset-Vektor $q_{\text{standoff,local}} = [dx_{\text{standoff}} \ dy_{\text{standoff}}]$, der den gewünschten Abstand in Relation zur Greifer-Orientierung beschreibt. Die Berechnung der Weltkoordinaten für den Standoff-Punkt erfolgt durch Rücktransformation:

$$\begin{bmatrix} x_{\text{standoff}} \\ y_{\text{standoff}} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - R(\theta_t) \cdot \begin{bmatrix} dx_{\text{standoff}} \\ dy_{\text{standoff}} \end{bmatrix} \quad (3.3)$$

wobei $R(\theta_t)$ die Rotationsmatrix um den Winkel θ_t darstellt:

$$R(\theta_t) = \begin{bmatrix} \cos \theta_t & -\sin \theta_t \\ \sin \theta_t & \cos \theta_t \end{bmatrix} \quad (3.4)$$

In der Implementierung wird standardmäßig ein Offset von 0.6 LE in x -Richtung verwendet ($q_{\text{standoff,default}} = [0, 6 \ 0, 0]$). Falls in der Zieldefinition jedoch ein spezifischer Offset hinterlegt ist (z. B. für seitliches Greifen), wird dieser priorisiert verwendet.

3.2.2 Handhabung des Objekts (Attach/Detach)

Ein kritischer Aspekt der Simulation ist der Zustandswechsel des manipulierten Objekts.

- **Vor dem Greifen:** Das Objekt ist ein statisches Hindernis in der Liste obstacles des CollisionCheckers.
- **Nach dem Greifen (attach_object):** Sobald der Greifzustand erreicht ist, wird das Objekt aus der Hindernisliste entfernt und stattdessen als Polygon an das letzte Segment des Roboterarms angehängt. Für die Kollisionsprüfung bedeutet dies, dass das Objekt nun Teil der Roboterkonfiguration ist und sich mitbewegt (siehe Abschnitt 2.1.2, Prüfung „Gegriffenes Objekt“).
- **Beim Ablegen (detach_object):** Das Objekt wird vom Roboter entfernt und an der aktuellen Position als neues statisches Hindernis in die Welt eingefügt.

Da sich durch die Interaktion mit dem Objekt sowohl die Geometrie des Roboters (beim Greifen oder Ablegen) als auch die Liste der statischen Hindernisse verändert, wird der Planungsgraph für jedes Teilproblem vollständig neu aufgebaut. Für die Visualisierung werden die resultierenden Graphen und Pfade anschließend aneinandergehängt.

3.3 Simulation und Ergebnisse

Die Validierung des implementierten MultiGoalPlannerRunner erfolgt exemplarisch in der Umgebung *Empty World*. Diese Umgebung wird gewählt, da sie eine klare visuelle Unterscheidung zwischen den globalen Pfadsegmenten (PRM-basiert) und den lokalen Manipulationspfaden (linear interpoliert) ermöglicht, ohne dass Sichtlinien durch Hindernisse verdeckt werden.

3.3.1 Qualitativer Vergleich der Roadmaps

Abbildung 3.1 stellt die generierten Roadmaps und Trajektorien beider Planungsverfahren gegenüber.

Es zeigen sich charakteristische Unterschiede:

- **Struktur:** Während der VisibilityPRM (Abb. 3.1a) klare, direkte Verbindungen zwischen den Aktionspunkten schafft, ist die Lösung des LazyPRM (Abb. 3.1b) durch eine hohe Dichte an redundanten Knoten und Kanten geprägt.
- **Local Approach:** Bei beiden Verfahren ist der Übergang vom globalen Planer zur lokalen Interpolation deutlich erkennbar. Die „Stacheln“ an den Greif- und Ablagepositionen repräsentieren die linearen Anfahrtswege (Approach/Retreat), die ohne den probabilistischen Planer berechnet werden.

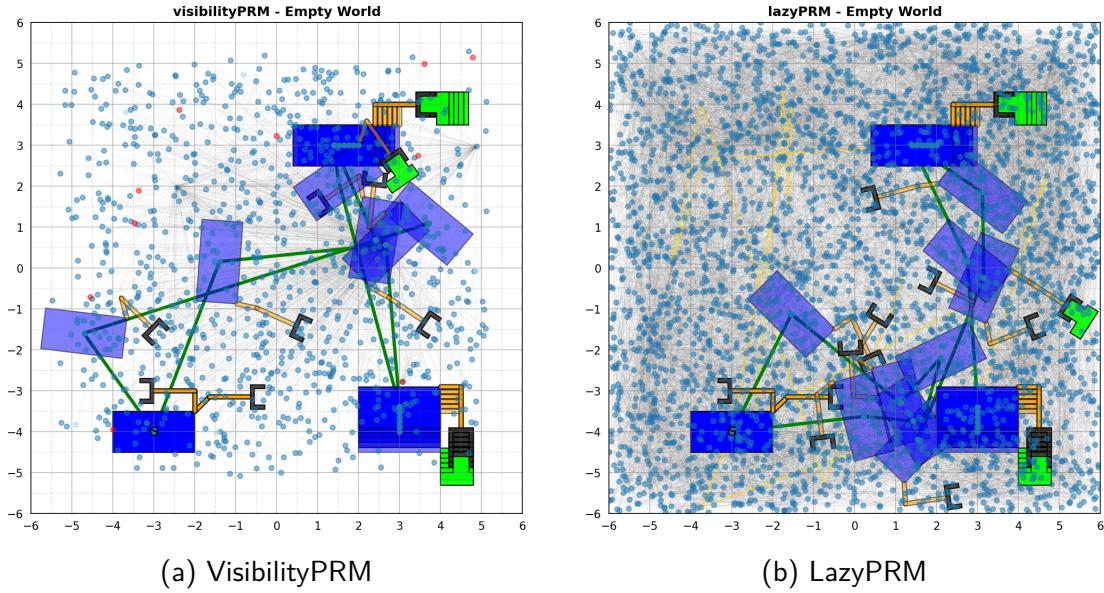


Abbildung 3.1: Vergleich der Pick-and-Place Sequenz in der Umgebung *Empty World*.

(a) Der VisibilityPRM erzeugt einen spärlichen, geometrisch optimierten Graphen. (b) Der LazyPRM flutet den Freiraum mit Knoten. In beiden Fällen ist die Sequenz: Start → Pick (oben) → Place (unten rechts) → Move (unten links) erfolgreich.

3.3.2 Quantitative Auswertung (Multi-Run)

Um die Robustheit der sequenziellen Planung zu bewerten, wird die Pick-and-Place Aufgabe in einem Batch-Prozess ($N = 10$ Durchläufe) über alle fünf Benchmark-Szenarien hinweg evaluiert. Dabei wird analog zu Kapitel 2.2 unterschieden zwischen aktivierter und deaktivierter Eigenkollisionsprüfung.

Performance mit Eigenkollisionsprüfung

Unter realistischen Bedingungen (aktiver Self-Check) zeigt sich, dass die Segmentierung der Aufgabe in Teilprobleme („Pick“, „Place“, „Move“) zwar die Komplexität strukturiert, die Anforderungen an den Planer jedoch im Vergleich zur reinen Navigation signifikant steigen.

Abbildung 3.2 zeigt exemplarisch die Ergebnisse in der *Empty World*, während die detaillierten Ergebnisse der komplexen Szenarien im Anhang A.4 (Abb. A.4a bis A.4e) aufgeführt sind.

Eine szenarienübergreifende Analyse liefert folgende Erkenntnisse:

- **Basis-Performance:** In der hindernisfreien Umgebung (*Empty World*) erreichen beide Planer eine hohe Zuverlässigkeit (100 % Erfolg).
- **Einbruch bei LazyPRM in Clutter-Umgebungen:** Im Szenario *Forest* offenbart sich eine signifikante Schwäche des LazyPRM. Während der VisibilityPRM hier weiterhin 100 % Erfolgsrate erzielt, fällt der LazyPRM auf 10 % ab (siehe Anhang A.4, Abb. A.4d). Die Fehleranalyse zeigt, dass das Scheitern fast ausschließlich im Segment „PICK“ auftritt.

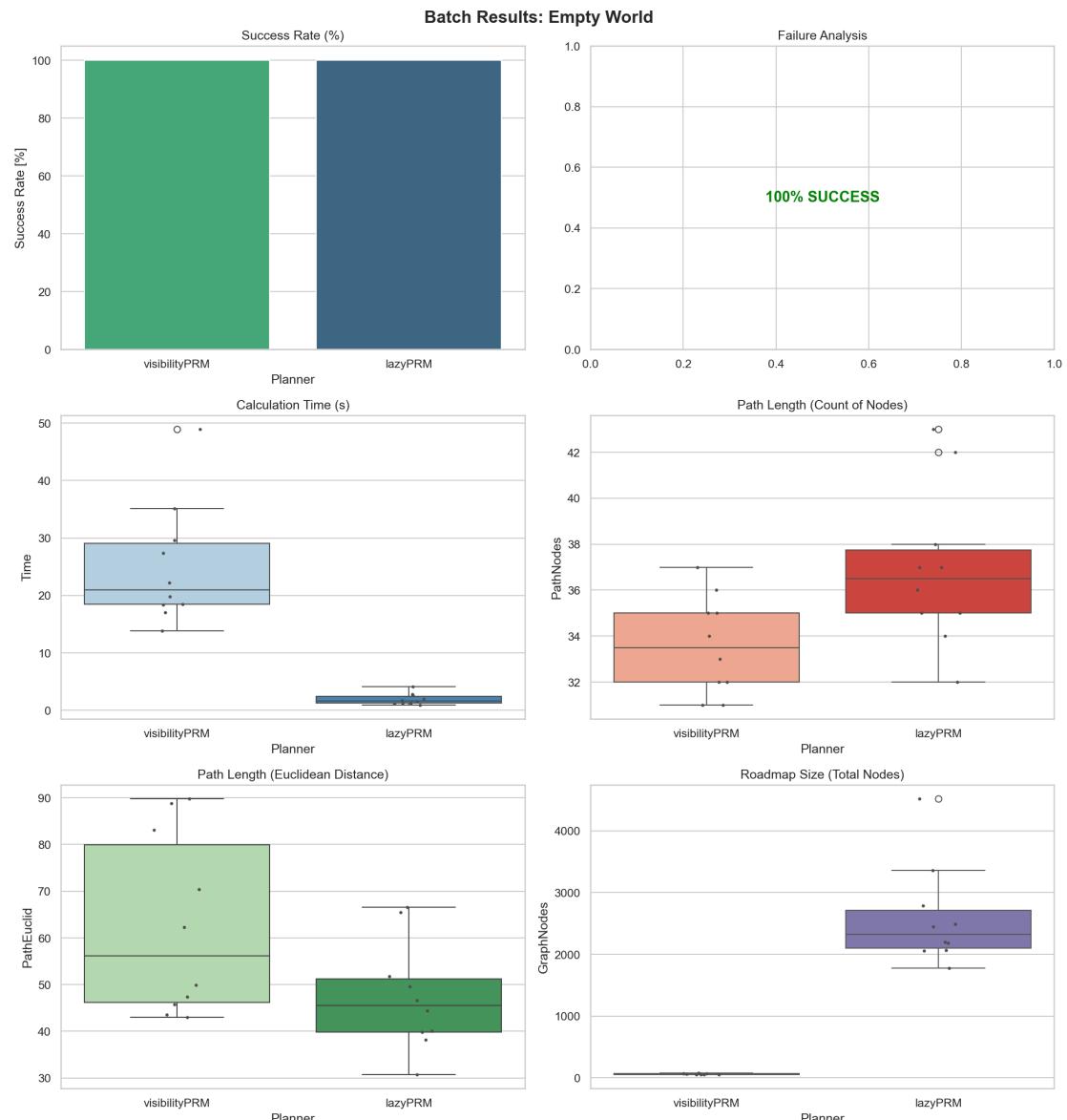


Abbildung 3.2: Statistische Auswertung der Pick-and-Place Aufgabe in *Empty World* (mit Self-Check). Die Erfolgsrate misst das erfolgreiche Absolvieren der *gesamten Sequenz*.

- **Herausforderung Engstellen:** In den Szenarien *The Wall* und *Narrow Passage* sinkt selbst die Erfolgsrate des VisibilityPRM auf ca. 70 % (vgl. Anhang A.4, Abb. A.4b und A.4c). Die Kombination aus präziser Basis-Navigation durch die Engstelle und anschließender Arm-Manipulation erhöht die Wahrscheinlichkeit, in lokale Minima zu geraten oder das Zeitlimit zu überschreiten.
- **Kinematische Grenzen:** Das Szenario *Shelf Reach* konnte von keinem der Planer gelöst werden (0 % Erfolg, vgl. Anhang A.5, Abb. A.4e). Dies verdeutlicht die physikalischen Grenzen des Robotersystems: Das Greifen hinter einem Hindernis unter Berücksichtigung der Eigenkollision erfordert eine komplexe Arm-Konfiguration, die durch reines Sampling in diesem 5D-Raum schwer zu finden ist. Eine Erhöhung der Rechenzeit oder eine Anpassung der ntry-Parameter könnte hier die probabilistische Vollständigkeit des Algorithmus besser ausnutzen.
- **Ursachenanalyse:** Der Unterschied liegt in der Instanziierung des zu greifenden Objekts als zusätzliches Hindernis. Im reinen Navigations-Test ist der Raum um die Zielkoordinate frei. Im Pick-Szenario blockiert das Objekt-Polygon nun diesen Bereich. Auch wenn der Standoff-Punkt (0.6 m Abstand) selbst kollisionsfrei ist, reduziert das Objekt das Volumen der validen Konfigurationen in der Zielumgebung. Pfade, die zuvor durch den Objekt-Raum führten, sind nun invalide. Es entsteht eine lokale Engstelle (vgl. *Narrow Passage*) um das Ziel, die für den probabilistischen Ansatz des LazyPRM deutlich schwerer zu finden ist als der zuvor freie Raum. Da LazyPRM Kanten lazy validiert und keine explizite Strategie besitzt, um Knoten exakt an Hindernisgrenzen zu platzieren, werden Verbindungspfade zum Ziel, die das Objekt-Hindernis minimal tangieren, häufig als Kollision verworfen. Der Planer findet somit den „Zugang“ zum Objekt nicht.

Einfluss der Eigenkollision auf die Manipulationsaufgabe

Abschließend wurden alle Szenarien erneut ohne aktive Eigenkollisionsprüfung evaluiert. Abbildung 3.3 zeigt die Ergebnisse der Referenzumgebung. Der Vergleich mit den komplexen Szenarien im Anhang A.5 (Abb. A.5a bis A.5e) liefert wichtige differenzierte Erkenntnisse:

1. **Bestätigung der Beobachtungen aus Abschnitt 2.2.4:** Im Szenario *Forest* steigt die Erfolgsrate des LazyPRM im Pick-and-Place Test ohne Eigenkollisionsprüfung auf 100 % an (vgl. Anhang A.5, Abb. A.5d). Dies bestätigt die These, dass die Eigenkollision in diesem Szenario der limitierende Faktor ist. Ohne diese Einschränkung erweitert sich der gültige Konfigurationsraum erheblich, was es dem LazyPRM ermöglicht, valide Pfade zu finden.
2. **Bestätigung kinematischer Limits:** Im Szenario *Shelf Reach* führt auch die Deaktivierung der Eigenkollision zu keinem Erfolg (0 %, siehe Anhang A.5 Abb. A.5e). Dies untermauert die These, dass die Komplexität der Regal-Umgebung den Lösungsraum so stark einschränkt, dass dieser mit Standard-Sampling-Dichten praktisch unauffindbar bleibt.
3. **Dominanz externer Constraints:** Im Gegensatz zum *Forest*-Szenario zeigt die Deaktivierung der Eigenkollisionsprüfung in den Szenarien *The Wall* und *Narrow Passage* keine signifikante Verbesserung der Erfolgsrate für den LazyPRM (verbleibt bei 0 %, vgl. Anhang A.5 Abb. A.5b). Auch beim VisibilityPRM bleibt

die Erfolgsrate stabil bei ca. 70 % (vgl. Anhang A.5 Abb. A.5c).

Ursachenanalyse: Dies bestätigt, dass in diesen Fällen nicht die interne geometrische Beschränkung des Roboters, sondern die externe Topologie der Flaschenhals ist. Die „Narrow Passage“ im 5-dimensionalen Konfigurationsraum wird hier primär durch den schmalen Korridor in der Welt definiert. Da der LazyPRM keine Strategie besitzt, um gezielt Proben in diesen kritischen, engen Regionen zu generieren, scheitert er unabhängig davon, ob Eigenkollisionen erlaubt sind oder nicht.

4. **Transport-Phase:** Lediglich in den wenigen erfolgreichen Läufen zeigt sich eine leichte Reduktion der Pfadkosten, da der Roboter das Objekt beim Transport „durch sich selbst“ ziehen kann, um die Distanz zu verkürzen.

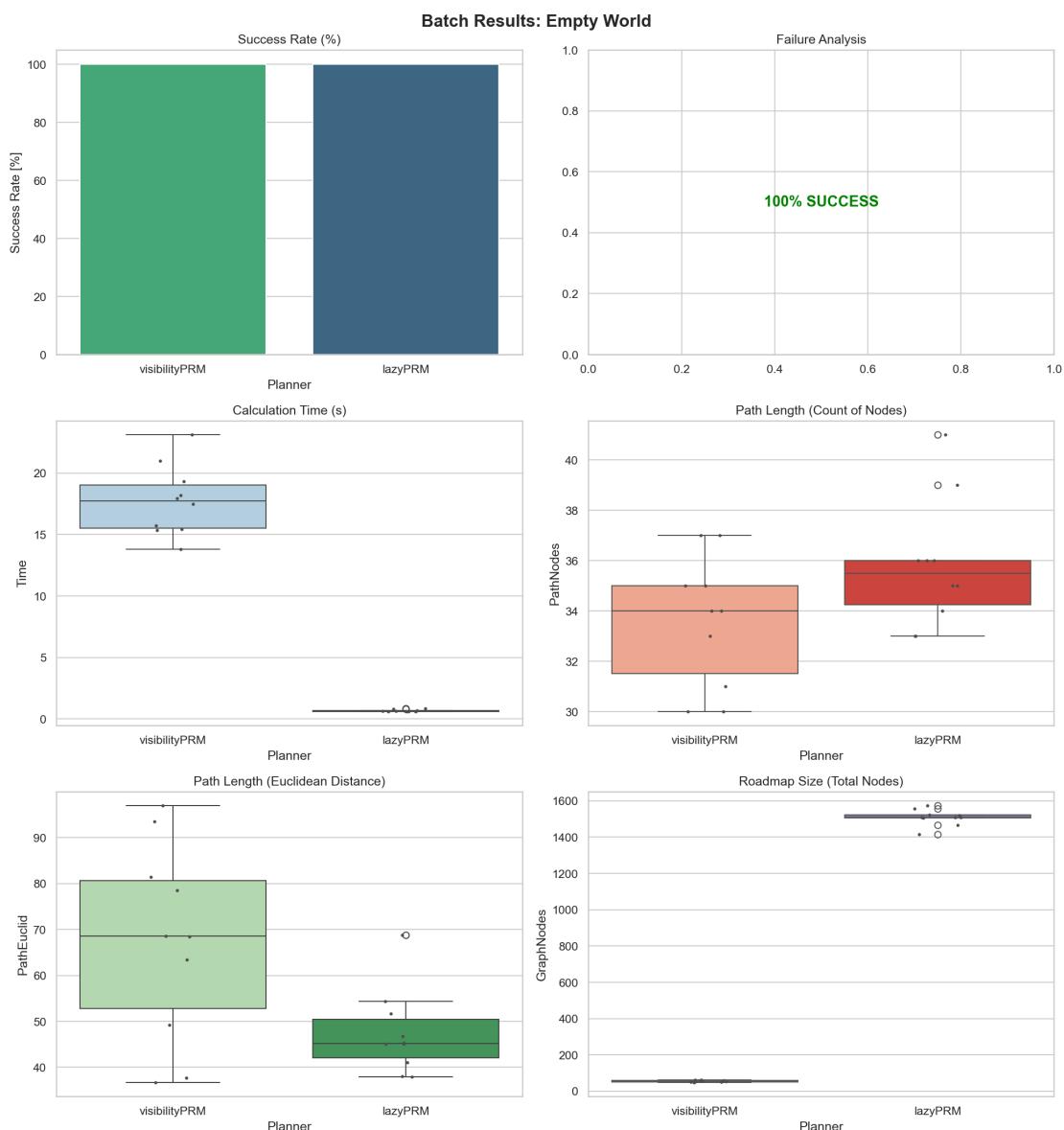


Abbildung 3.3: Statistische Auswertung der Pick-and-Place Aufgabe in *Empty World* (ohne Self-Check).

Zusammenfassend bestätigt die Auswertung, dass für Manipulationsaufgaben in engen Umgebungen die Wahl des Planers (VisibilityPRM vs. LazyPRM) und die Sampling-Strategie entscheidender sind als die reine Relaxierung von Kollisionsbedingungen.

3.3.3 Zusammenfassendes Fazit der Pick-and-Place-Evaluation

Die quantitative Auswertung der Manipulationsaufgabe verdeutlicht, dass die Komplexität gegenüber der reinen Navigation signifikant ansteigt, da das zu greifende Objekt zusätzliche geometrische Restriktionen im Konfigurationsraum erzeugt.

- **Robustheit vs. Effizienz:** Während der VisibilityPRM durch seine gezielte Sampling-Strategie (Guard-Nodes) eine hohe Robustheit in engen und hindernisreichen Umgebungen beweist, bleibt der LazyPRM aufgrund seines rein zufälligen Samplings in diesen Szenarien oft erfolglos.
- **Rechenzeit und Skalierbarkeit:** Es zeigt sich eine deutliche Diskrepanz in der Recheneffizienz. Der LazyPRM profitiert in offenen Umgebungen wie der *Empty World* massiv von der verzögerten Kollisionsprüfung und ist mit ca. 1–2 s deutlich schneller als der VisibilityPRM (ca. 20–30 s). In komplexen Szenarien wie dem *Forest* steigt der Aufwand für Visibility-Prüfungen jedoch stark an (ca. 600 s), da die hohe Anzahl an Raycasts gegen die Umweltgeometrie den rechenintensivsten Faktor darstellt. Dieser Zeitaufwand wird jedoch durch die notwendige Robustheit in schwierigen Passagen gerechtfertigt.
- **Interne vs. externe Constraints:** Die Analyse zeigt eine klare Differenzierung der Fehlerursachen: In Szenarien wie *Forest* ist die Eigenkollision der limitierende Faktor, da ihre Deaktivierung die Erfolgsrate steigert. Im Gegensatz dazu wird das Scheitern in den Szenarien *The Wall* und *Narrow Passage* primär durch die externe Umweltgeometrie (Engstellen) bestimmt, weshalb ein Verzicht auf die Eigenkollisionsprüfung hier keinen Performance-Vorteil bringt.
- **Physikalische Grenzen:** Das vollständige Scheitern im *Shelf Reach*-Szenario markiert die Grenze des aktuellen Systems. Es lässt sich konstatieren, dass für solche hochgradig eingeschränkten Aufgaben entweder die Sampling-Dichte massiv erhöht werden müsste oder spezialisierte Heuristiken für den Greifraum erforderlich sind, um die theoretische probabilistische Vollständigkeit der Planer in praktischer Rechenzeit zu erreichen.

Insgesamt bestätigt die Evaluation, dass für komplexe Manipulationsaufgaben die Wahl einer intelligenten Sampling-Strategie (Visibility-Ansatz) entscheidender für den Erfolg ist als die bloße Rechengeschwindigkeit des Planers.

4 Theoretische Konzepte und Erweiterungen

Dieser Abschnitt behandelt weiterführende Fragestellungen der Roboterbahnplanung, die über die Implementierung des planaren Manipulators hinausgehen. Gemäß der Aufgabenstellung werden im Folgenden die notwendigen Systemerweiterungen für translatorische Gelenke sowie Methoden zur nachträglichen Bahnoptimierung diskutiert.

4.1 Erweiterung um translatorische Gelenke

Das in dieser Arbeit betrachtete System besteht aus einer mobilen Basis (3 Freiheitsgrade) und einem Arm mit rotatorischen Gelenken (Revolute Joints). Ein translatorisches Gelenk (Prisma-Gelenk) ermöglicht hingegen eine lineare Relativbewegung zwischen zwei Segmenten, was sowohl den Arbeitsraum als auch die kinematische Modellierung beeinflusst [1].

4.1.1 Auswirkung auf die Kinematik

In der Vorwärtsskinematik wird die Lage eines Segments relativ zum vorherigen Segment üblicherweise durch homogene Transformationsmatrizen T beschrieben.

Für ein **rotatorisches Gelenk** rotiert das lokale Koordinatensystem um die Gelenkkachse (z. B. z-Achse) um den variablen Winkel θ . Die Matrix hat die Form:

$$T_{rot}(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.1)$$

Für ein **translatorisches Gelenk** ist die Rotation statisch, während die Verschiebung d entlang der Gelenkkachse variabel ist. Die Transformationsmatrix ändert sich folglich zu:

$$T_{trans}(d) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.2)$$

Implementierungstechnische Konsequenzen: Für die Integration in den bestehenden CollisionChecker müsste die Methode zur Berechnung der Robotergeometrie (`get_robot_geometry`) angepasst werden. Anstatt für jedes Gelenk den Winkel zu akkumulieren, müsste für Prisma-Gelenke die effektive Länge des Segments dynamisch berechnet werden. Das Gelenk q_i würde direkt als Längenänderung in die Berechnung der Endpunktkoordinaten eingehen.

4.1.2 Anpassung des Konfigurationsraums (C-Space)

Die Einführung translatorischer Freiheitsgrade verändert die Topologie und die Metrik des Konfigurationsraums [2]:

- **Wertebereich und Topologie:** Während rotatorische Gelenke meist periodisch definiert sind ($-\pi$ bis π), sind translatorische Gelenke durch feste mechanische Anschläge begrenzt (d_{min} bis d_{max}). Mathematisch betrachtet ändert sich die Topologie des Gelenks von einem Kreis (S^1)¹ zu einer Linie (\mathbb{R}).
- **Sampling und Metrik:** Wie bereits bei der mobilen Basis (x, y in LE, θ in Radian), liegen auch bei einem Arm mit translatorischen Gelenken gemischte Einheiten vor. Dies stellt besondere Anforderungen an die Distanzfunktion (Metrik) für die *Nearest-Neighbor*-Suche. Da eine Differenz von 1 LE nicht äquivalent zu 1 rad ist, muss zwingend eine **gewichtete Euklidische Metrik** verwendet werden. Hierbei werden die translatorischen und rotatorischen Dimensionen mit Gewichtungsfaktoren w_i skaliert, um einen homogenen Abstandsraum zu schaffen.

4.2 Optimierung und Glättung von Bewegungsbahnen

Die von Sampling-basierten Algorithmen erzeugten Pfade bestehen naturgemäß aus linearen Segmenten zwischen zufällig platzierten Wegpunkten. Dies resultiert oft in „zackigen“, suboptimalen Bahnen mit abrupten Richtungsänderungen. Eine Nachbearbeitung (Post-Processing) ist daher essenziell [3].

4.2.1 Shortcut-Methode (Pruning)

Ein effizienter Ansatz zur Reduktion der Pfadlänge ist die heuristische Shortcut-Methode. Dieser Algorithmus arbeitet iterativ auf dem initialen diskreten Pfad:

1. Es werden zwei zufällige Konfigurationen q_a und q_b auf dem Pfad ausgewählt, die nicht direkt benachbart sind.
2. Es wird geprüft, ob die direkte Verbindungsgeraden (Line-of-Sight) im Konfigurationsraum zwischen q_a und q_b kollisionsfrei ist.

¹Die Topologie S^1 (1-Sphäre) beschreibt einen kreisförmigen Raum, bei dem die Werte periodisch umlaufen, d. h. $-\pi$ und π sind identisch.

3. Ist die Verbindung valide, werden alle dazwischenliegenden Wegpunkte entfernt und durch die direkte Kante ersetzt.

Das Ergebnis ist ein stückweise linearer Pfad. Dieser ist zwar kürzer, bleibt jedoch lediglich C^0 -stetig², was zu mechanisch ungünstigen Stopp-and-Go-Bewegungen an den Wegpunkten führt.

4.2.2 Glättung mittels Splines

Um eine kinematisch hochwertige Bahn zu generieren, können die Wegpunkte des optimierten Pfades durch Spline-Kurven approximiert werden. Ziel ist hierbei oft C^2 -Stetigkeit³.

- **B-Splines:** Die Wegpunkte des PRM-Pfades dienen hierbei als Kontrollpunkte. Der resultierende Spline verläuft nicht zwingend durch die Punkte, sondern wird von ihnen approximiert. Dies garantiert einen glatten Verlauf, birgt jedoch das Risiko, dass die Kurve Hindernisse schneidet („Corner Cutting“).
- **Validierung:** Nach der Spline-Generierung muss die Kurve diskretisiert und erneut auf Kollisionsfreiheit geprüft werden. Sollten Kollisionen auftreten, können iterative Verfahren (z. B. Einfügen zusätzlicher Stützpunkte) angewandt werden, um den Pfad lokal zu deformieren und in den freien Raum zurückzuführen.

Eine Kombination aus der Shortcut-Methode für die globale Optimierung und anschließender Spline-Interpolation für die lokale Glättung stellt den Stand der Technik für mobile Manipulatoren dar.

4.3 Weitere Erweiterungsmöglichkeiten

Neben den theoretischen Konzepten zur Kinematik und Bahnoptimierung ergeben sich aus den Evaluationen (Kapitel 2 und 3) konkrete Ansätze zur Verbesserung der implementierten Softwarearchitektur und der Planungsstrategien.

4.3.1 Optimierung der Kollisionsprüfung

Wie in Abschnitt 2.2.3 (Quantitative Ergebnisse (Baseline)) beschrieben, kann die Reihenfolge der Prüfungen im *CollisionChecker* optimiert werden. Da die Eigenkollisionsprüfung (*Self-Collision*) bedeutend weniger Rechenzeit benötigt als die Sichtbarkeitsprüfung gegen externe Hindernisse, sollte diese vorgezogen werden. Es ist ineffizient, aufwendige Raycasts durchzuführen, wenn die Konfiguration bereits aufgrund einer internen Kollision ungültig ist.

² C^0 -Stetigkeit bedeutet, dass der Pfad zusammenhängend ist (keine Sprünge in der Position), aber Knicke aufweist. An diesen Knicken ist die Geschwindigkeit unstetig (der Roboter müsste theoretisch anhalten, um die Richtung zu ändern).

³ C^2 -Stetigkeit bedeutet, dass nicht nur die Position und die Geschwindigkeit (C^1), sondern auch die Beschleunigung stetig verläuft. Dies ermöglicht ruckfreie Bewegungen.

4.3.2 Zeit-normalisiertes Benchmarking

Um die Leistungsfähigkeit der Planer fairer zu vergleichen, könnte ein *Time-Budget*-Ansatz gewählt werden. Dabei erhalten beide Planer (LazyPRM und VisibilityPRM) exakt die gleiche Rechenzeit. Die Parameter der Planer werden so angepasst, dass sie dieses Zeitfenster optimal nutzen. Anschließend wird die Erfolgsrate (*Success Rate*) als primäres Vergleichskriterium ausgewertet.

4.3.3 Entkoppelte Planung

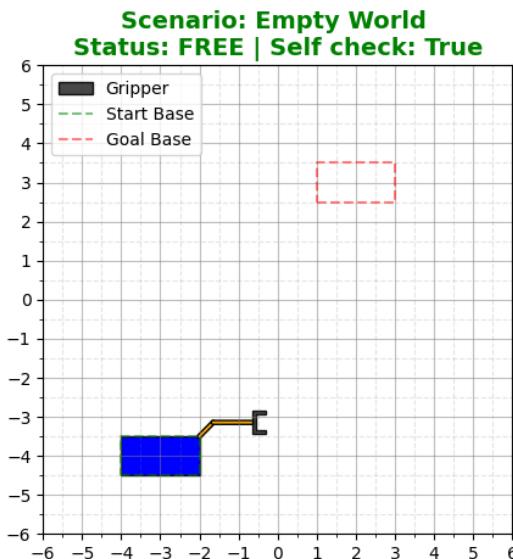
Zur Steigerung der Geschwindigkeit und Qualität kann die Planung hierarchisch entkoppelt werden. Im ersten Schritt wird lediglich ein Pfad für die Basis $(x, y, \theta_{\text{base}})$ geplant, während die Armgelenke (θ_1, θ_2) in einer starren Transportkonfiguration verbleiben. Erst in der unmittelbaren Umgebung einer Pick- oder Place-Aktion wird der volle 5-dimensionale Konfigurationsraum betrachtet, um die notwendigen Armbewegungen zu planen.

4.3.4 Heuristisches Sampling für LazyPRM

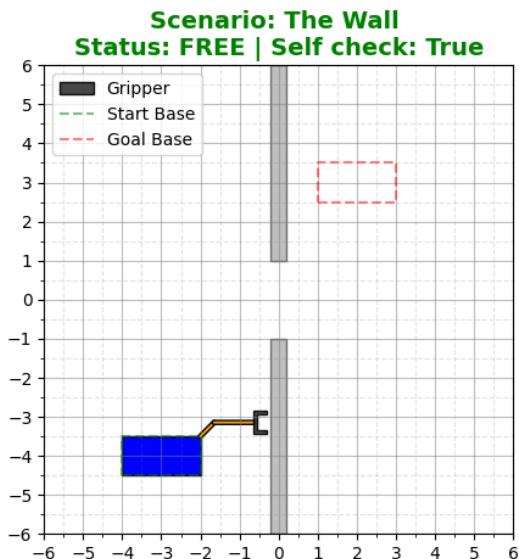
Der aktuelle LazyPRM verteilt neue Punkte in nachfolgenden Iterationen gleichverteilt über den gesamten Raum. Eine effizientere Strategie wäre es, diese Punkte gezielt an Engstellen zu platzieren. Dies kann beispielsweise dadurch erreicht werden, dass bevorzugt in der Nähe von Konfigurationen gesampelt wird, an denen zuvor unzulässige Kanten oder Knoten detektiert wurden, um so kritische Bereiche im Graphen gezielt aufzulösen.

A Anhang

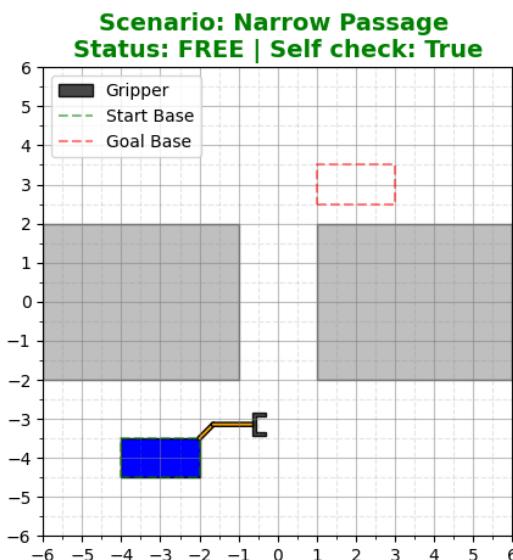
A.1 Planungsumgebungen



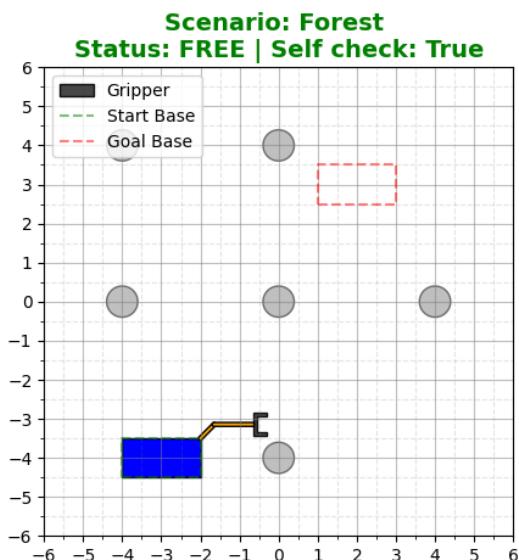
(a) Empty World



(b) The Wall

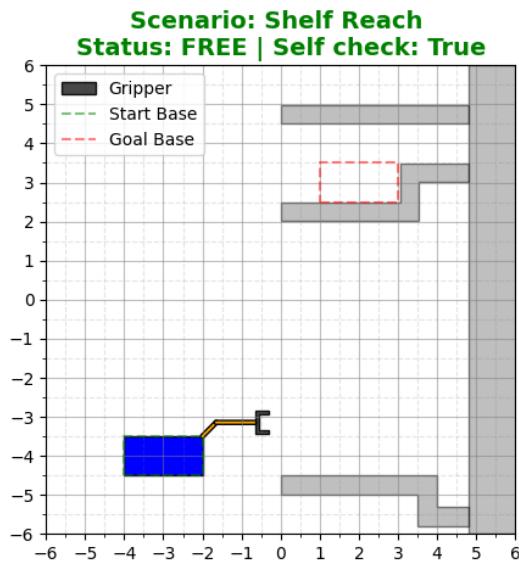


(c) Narrow Passage



(d) Forest

Abbildung A.1: Benchmarks / Planungsumgebungen



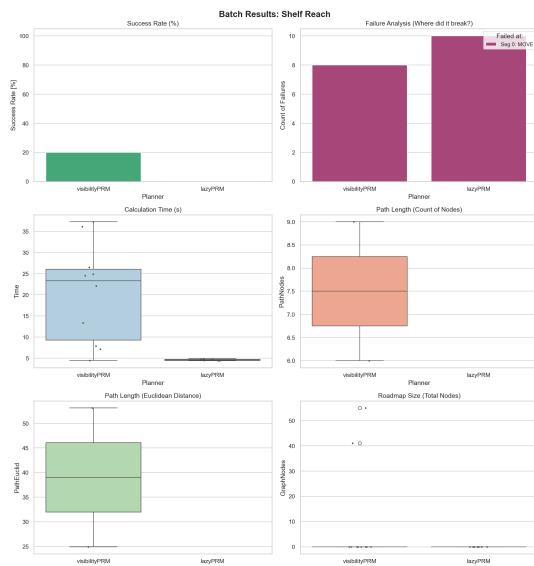
(e) Shelf Reach

Abbildung A.1: Benchmarks / Planungsumgebungen (Fortsetzung)

A.2 Ergebnisse Multi-Run MIT Überprüfung der Eigenkollision



Abbildung A.2: Ergebnisse Multi-Run mit Überprüfung der Eigenkollision



(e) Shelf Reach

Abbildung A.2: Ergebnisse Multi-Run mit Überprüfung der Eigenkollision (Fortsetzung)

A.3 Ergebnisse Multi-Run OHNE Überprüfung der Eigenkollision

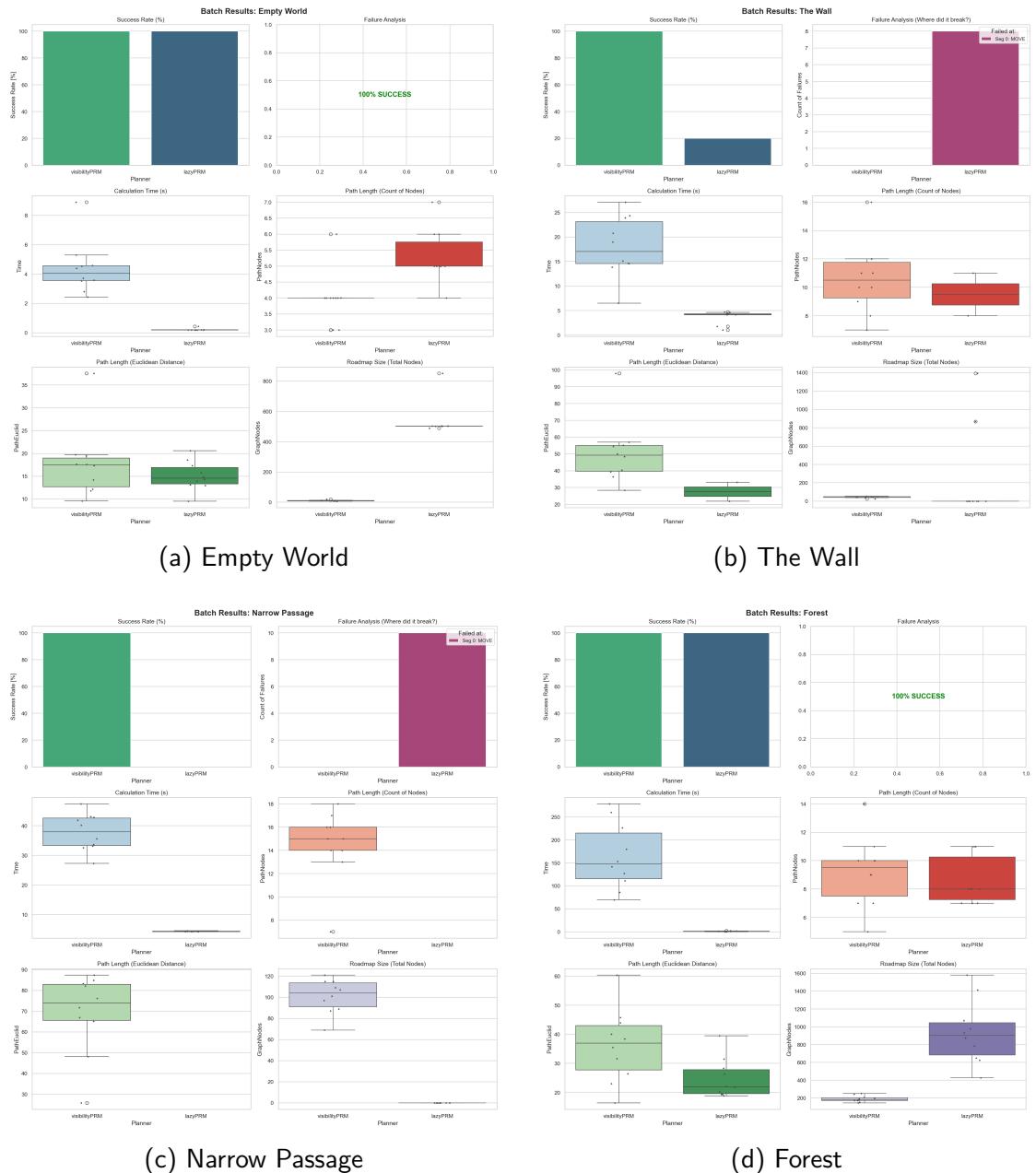
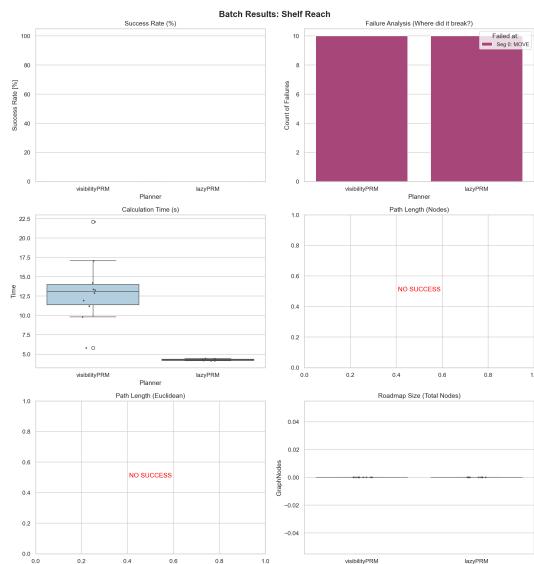


Abbildung A.3: Ergebnisse Multi-Run ohne Überprüfung der Eigenkollision



(e) Shelf Reach

Abbildung A.3: Ergebnisse Multi-Run ohne Überprüfung der Eigenkollision (Fortsetzung)

A.4 Ergebnisse Multi-Run Pick-And-Place MIT Überprüfung der Eigenkollision

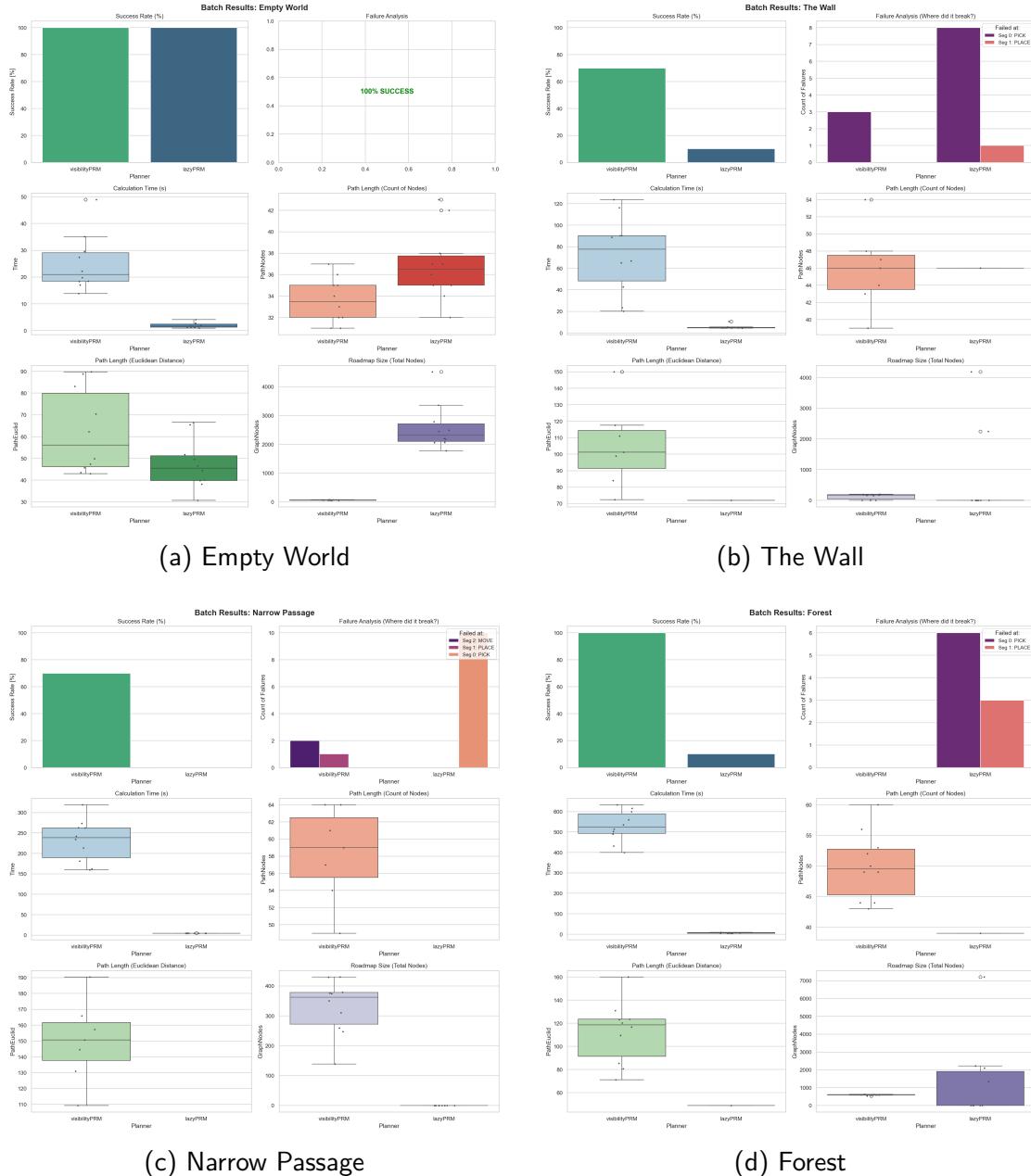
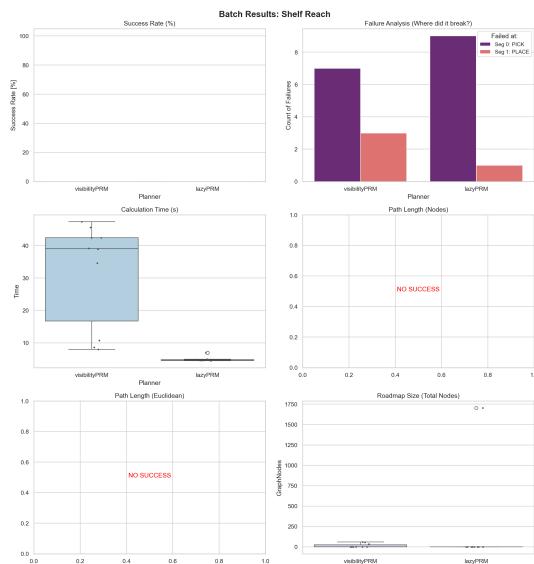


Abbildung A.4: Ergebnisse Multi-Run Pick-And-Place mit Überprüfung der Eigenkollision



(e) Shelf Reach

Abbildung A.4: Ergebnisse Multi-Run Pick-And-Place mit Überprüfung der Eigenkollision (Fortsetzung)

A.5 Ergebnisse Multi-Run Pick-And-Place OHNE Überprüfung der Eigenkollision

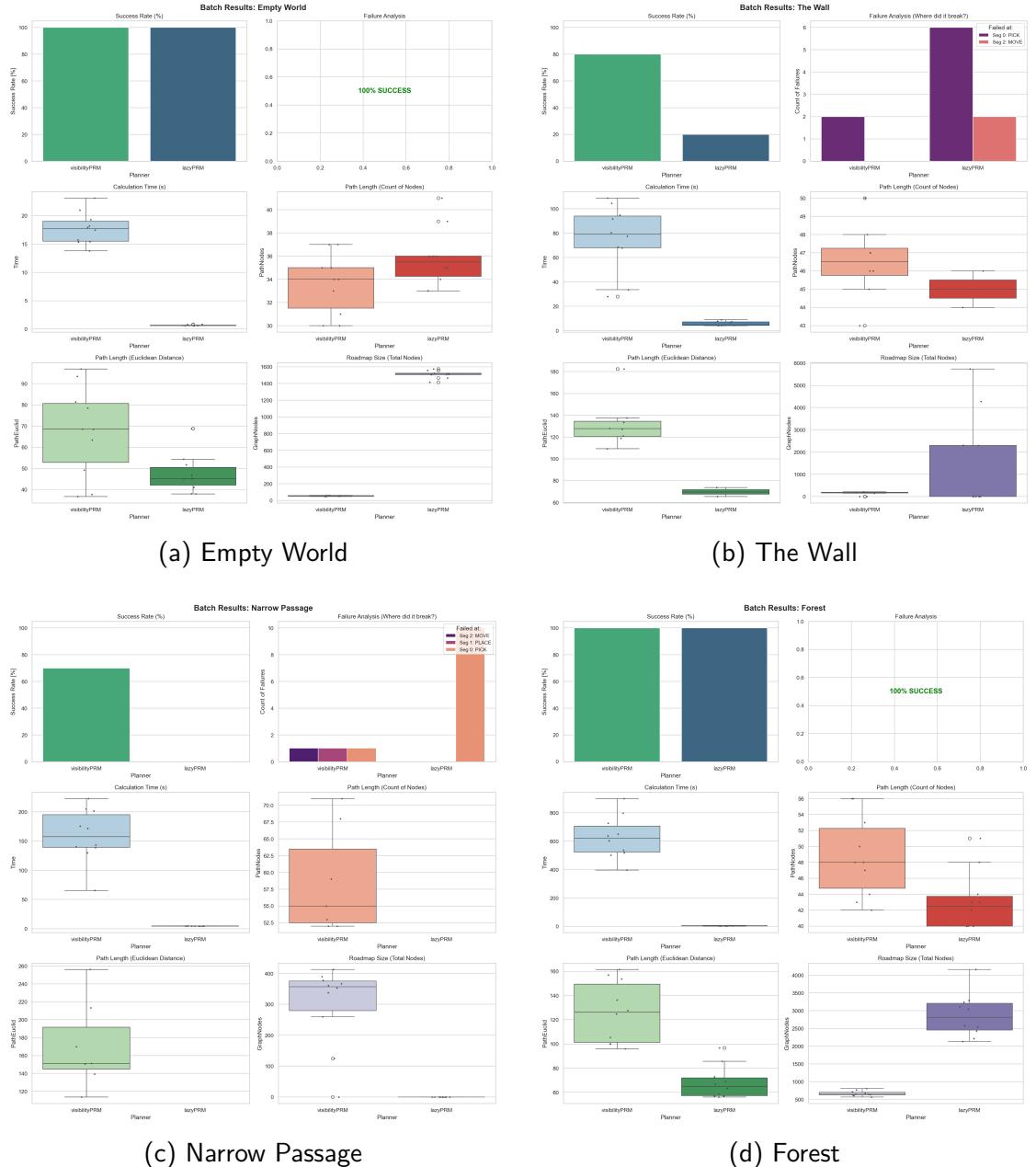
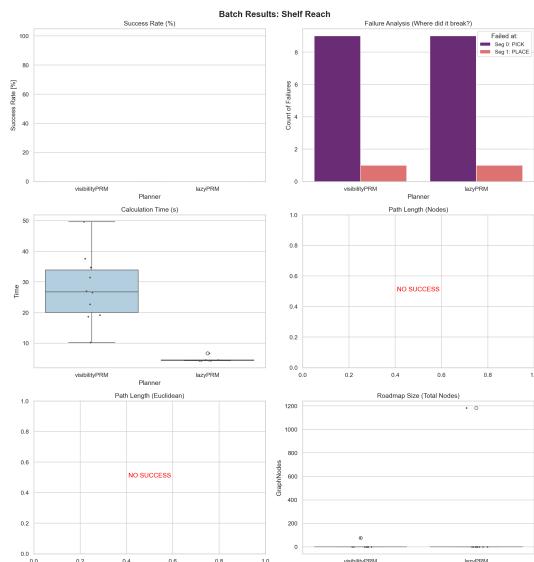


Abbildung A.5: Ergebnisse Multi-Run Pick-And-Place ohne Überprüfung der Eigenkollision



(e) Shelf Reach

Abbildung A.5: Ergebnisse Multi-Run Pick-And-Place ohne Überprüfung der Eigenkollision (Fortsetzung)

Literatur

- [1] B. Siciliano und O. Khatib, *Springer handbook of robotics*. Springer, 2016.
- [2] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.
- [3] H. M. Choset, *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Karlsruhe, 30. September 2025

Paul Glaser, B. Eng

Tim Schäfer, B. Eng

Felix Wietschel, B. Eng