

Bachelor Abschlussarbeit  
**Entwicklung und Evaluation eines  
echtzeitfähigen SLAM-Algorithmus´ für den  
humanoiden Roboter Myon**

---

von Milan Ruiz Holtgreffe

18. November 2025

Matr.	936865
Fachbereich	VII
Studiengang	Humanoide Robotik (HROB)
Zeitraum	18. August 2025 - 18. November 2025
Neurorobotik Labor	Berliner Hochschule für Technik Berlin Luxemburger Str. 10, 13353 Berlin
Betreuer	Prof. Dr. Manfred Hild

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Problemstellung . . . . .	1
1.2	Zielsetzung und Ansatz . . . . .	2
1.3	Aufbau der Arbeit . . . . .	2
<b>2</b>	<b>Verwandte Arbeit</b>	<b>3</b>
2.1	Visuelle Lokalisierung in der Robotik . . . . .	3
2.2	Biologisch inspirierte Navigation . . . . .	4
2.3	SeqSLAM: Sequence SLAM . . . . .	4
2.4	Vorarbeiten und Modifikationen . . . . .	5
2.4.1	Echtzeitfähigkeit durch Räumliche Abtastung . . . . .	5
2.4.2	Status Quo und Erweiterungen . . . . .	6
<b>3</b>	<b>Aufbau und Methodik</b>	<b>7</b>
3.1	Die Konfiguration Myons . . . . .	7
3.2	Versuchsumgebung . . . . .	9
3.3	Datensätze . . . . .	10
<b>4</b>	<b>Lokalisierung</b>	<b>11</b>
4.1	Vorverarbeitung . . . . .	11
4.1.1	Patch Normalization . . . . .	12
4.1.2	Histogrammausgleich . . . . .	13
4.1.3	Inverse Gammakorrektur . . . . .	13
4.2	Differenzenmatrix . . . . .	14
4.3	Odometrie . . . . .	16
4.4	Lokalisierung durch Sequenzvergleich . . . . .	18
4.4.1	Verzicht auf Dynamic Time Warping . . . . .	18

4.4.2	Berechnung der Sequenzkosten . . . . .	19
4.4.3	Confidence-Score . . . . .	20
<b>5</b>	<b>Mapping</b>	<b>21</b>
5.1	Pose-Graph . . . . .	21
5.2	Gridmap . . . . .	22
<b>6</b>	<b>Algorithmus Auswertung</b>	<b>24</b>
6.1	Bildauffösungen . . . . .	27
6.2	Bildvorverarbeitung . . . . .	29
6.3	Analyse der Sequenzlänge . . . . .	31
6.4	Fenstergröße . . . . .	35
6.5	Odometriefehler . . . . .	36
<b>7</b>	<b>Confidence-Threshold</b>	<b>40</b>
<b>8</b>	<b>Anwendung</b>	<b>46</b>
8.1	Evaluation auf der Hardware . . . . .	46
8.2	Analyse von Trajektorien-Sprüngen . . . . .	48
<b>9</b>	<b>Fazit</b>	<b>50</b>
9.1	Zusammenfassung . . . . .	50
9.2	Kritische Einordnung: Von der Lokalisierung zu SLAM . . . . .	51
9.3	Ausblick . . . . .	52
	<b>Abbildungsverzeichnis</b>	<b>53</b>
	<b>Tabellenverzeichnis</b>	<b>55</b>
	<b>Literaturverzeichnis</b>	<b>56</b>

# 1 Einleitung

Die Fähigkeit zur autonomen Navigation ist eine der fundamentalen Voraussetzungen für mobile Robotersysteme. Damit ein Roboter sinnvoll mit seiner Umwelt interagieren kann, muss er nicht nur Hindernisse erkennen, sondern auch seine eigene Position im Raum bestimmen und eine Karte seiner Umgebung aufbauen. Dieses Problem ist in der Robotik als *Simultaneous Localization and Mapping* (SLAM) bekannt: Für eine präzise Lokalisierung wird eine Karte benötigt, während für den Aufbau einer konsistenten Karte eine genaue Lokalisierung unerlässlich ist.

Besondere Herausforderungen stellen dabei Umgebungen dar, die sich visuell stark ähneln oder extremen Beleuchtungsänderungen unterworfen sind. Lange, gleichförmige Büroflure führen bei herkömmlichen Algorithmen oft zu Fehlschlüssen. Zudem verfügen viele mobile Plattformen, wie der in dieser Arbeit verwendete humanoide Roboter Myon, über begrenzte Rechenressourcen, die eine effiziente Verarbeitung der Sensordaten in Echtzeit erfordern.

## 1.1 Problemstellung

Etablierte Verfahren der visuellen Lokalisierung sind primär für strukturreiche Umgebungen mit statischer Beleuchtung entwickelt worden. In der Einsatzumgebung des Roboters Myon – einem Forschungslabor mit langen, texturarmen Fluren und stark wechselndem Tageslichteinfall – stoßen diese Methoden jedoch an ihre Grenzen. Die visuelle Ambiguität der Umgebung führt zu fehlerhaften Positionsbestimmungen.

Erschwerend kommt hinzu, dass die Hardware des Roboters die verfügbare Bildauflösung und Rechenleistung stark limitiert. Daher muss ein geeigneter SLAM-Algorithmus nicht nur robust gegenüber visuellen Veränderungen

sein, sondern auch extrem ressourceneffizient arbeiten, um parallel zu anderen Steuerungsprozessen in Echtzeit funktionsfähig zu bleiben.

## 1.2 Zielsetzung und Ansatz

Das Ziel dieser Bachelorarbeit ist die Entwicklung, Implementierung und Evaluation eines echtzeitfähigen visuellen Lokalisierungsalgorithmus für den Roboter Myon. Der gewählte Ansatz basiert auf dem *SeqSLAM*-Verfahren (Sequence SLAM), welches nicht Einzelbilder, sondern ganze Bildsequenzen vergleicht, um Eindeutigkeit zu gewährleisten.

Im Gegensatz zur ursprünglichen Implementierung von SeqSLAM, die auf zeitlicher Abtastung basiert, verfolgt diese Arbeit einen neuen Ansatz der räumlichen Abtastung. Durch die direkte Kopplung der Bildaufnahme an die Odometrie des Roboters wird die algorithmische Komplexität signifikant reduziert. Dies ermöglicht den Verzicht auf rechenintensive Synchronisationsverfahren und gewährleistet die geforderte Echtzeitfähigkeit auf der eingebetteten Hardware.

## 1.3 Aufbau der Arbeit

Die Arbeit gliedert sich wie folgt: Kapitel 2 ordnet den Ansatz in den Stand der Forschung ein und grenzt ihn technisch von feature-basierten Methoden ab. Kapitel 3 beschreibt die Hardware-Plattform Myon und die Methodik der Datenerfassung. Kapitel 4 detailliert den entwickelten Lokalisierungsalgorithmus und die mathematische Umsetzung der räumlichen Abtastung. Kapitel 5 stellt den konzeptionellen Rahmen für das Mapping (Pose-Graph und Gridmap) vor. Kapitel 6 evaluiert den Algorithmus anhand aufgezeichneter Datensätze und analysiert den Einfluss verschiedener Parameter. Kapitel 7 entwickelt eine Strategie zur Wahl eines *Confidence-Thresholds* zur Vermeidung von *False Positives*. Kapitel 8 validiert das System abschließend im Realeinsatz auf dem Roboter und untersucht das Verhalten bei Trajektorien-Sprüngen. Den Abschluss bilden eine Zusammenfassung der Ergebnisse und ein Ausblick auf zukünftige Erweiterungen.

## 2 Verwandte Arbeit

Die visuelle Lokalisierung in veränderlichen Umgebungen ist ein zentrales Forschungsfeld der Robotik. Dieses Kapitel ordnet den in dieser Arbeit verwendeten Ansatz in den aktuellen Stand der Forschung ein. Zunächst werden klassische Feature-basierte Verfahren und ihre Grenzen in repetitiven Umgebungen diskutiert. Anschließend wird der biologisch inspirierte Ansatz von RatSLAM und dessen Weiterentwicklung zu SeqSLAM vorgestellt. Abschließend werden die spezifischen Vorarbeiten aus der Praxisphase und die daraus resultierenden Modifikationen für den Roboter Myon dargelegt.

### 2.1 Visuelle Lokalisierung in der Robotik

Klassische visuelle SLAM-Systeme (V-SLAM), wie *ORB-SLAM* von [Mur-Artal et al., 2015], basieren auf der Extraktion und dem Wiedererkennen lokaler Bildmerkmale (*Features*). Diese Algorithmen identifizieren markante Punkte im Bild und beschreiben diese durch Feature-Deskriptoren.

Dieser Ansatz erweist sich in strukturreichen Umgebungen als sehr robust und präzise. In Umgebungen mit stark repetitiven Strukturen, wie langen Bürofluren oder identischen Arbeitsplätzen, stoßen Feature-basierte Methoden jedoch an ihre Grenzen. Ebenso ist Myons Pixelpipeline, eine integrierte Bildverarbeitungshardware, auf Bilder mit niedriger Auflösung limitiert, was für Feature-basierte Methoden ungeeignet ist.

Im Gegensatz dazu verzichten *Appearance-based* Ansätze auf die Extraktion lokaler Merkmale. Stattdessen wird das Bild als Ganzes (oder globaler Deskriptor) betrachtet. Diese Methode ist robuster gegenüber lokaler Unschärfe und fehlender Textur, erfordert jedoch spezielle Strategien zur Bewältigung

von starken Beleuchtungsänderungen und ist schwach gegenüber Perspektivenwechseln.

## 2.2 Biologisch inspirierte Navigation

Ein bedeutender Zweig der Appearance-based Navigation ist von biologischen Vorbildern inspiriert, insbesondere von der Orientierung bei Nagetieren. Forschungen haben gezeigt, dass im Hippocampus von Ratten spezialisierte Neuronen existieren:

- *Place Cells* feuern, wenn sich das Tier an einem visuell bekannten Ort befindet.
- *Grid Cells* feuern in einem regelmäßigen gitterartigen Muster und unterstützen die Pfadintegration.

Aufbauend auf diesen Erkenntnissen entwickelten [Milford et al., 2004] das *RatSLAM*-System. RatSLAM verwendet ein *Continuous Attractor Network*, um die Pose und Ort in einer topologischen Karte zu repräsentieren. Anstatt eine metrisch exakte geometrische Karte zu erstellen, verknüpft RatSLAM visuelle Eindrücke (*Templates*) mit Posen im neuronalen Netzwerk. Dies ermöglicht eine robuste Navigation auch mit niedrig aufgelösten Kameras und ohne teure Laserscanner.

## 2.3 SeqSLAM: Sequence SLAM

Um den minimalen visuellen Informationsbedarf für zuverlässige Navigation zu untersuchen, stellte [Milford, 2012] den sequenzbasierten *SeqSLAM*-Algorithmus vor. SeqSLAM adressiert gezielt das Problem extremer visueller Veränderungen (z. B. Tag vs. Nacht, Sommer vs. Winter) und eignet sich zugleich als robustes Lokalisierungsmodul in RatSLAM.

Die Kernidee von SeqSLAM ist der Verzicht auf den Vergleich von Einzelbildern (*Single Image Matching*). Stattdessen wird eine Sequenz von aktuellen Bildern mit Sequenzen aus einer Datenbank verglichen. Durch die Betrachtung

von Sequenzen wird die Wahrscheinlichkeit einer eindeutigen Zuordnung signifikant erhöht, da die zeitliche Abfolge der visuellen Eindrücke als zusätzliches Unterscheidungsmerkmal dient.

Das originale SeqSLAM verwendet eine Differenzmatrix (siehe Abb. 1), in der die lokalen Kontraste der Bilder vorverarbeitet werden (*Patch Normalization*), um Beleuchtungsinvarianz zu erreichen. Um Geschwindigkeitsunterschiede zwischen der Lern- und der Fahrt-Sequenz auszugleichen, nutzt das Originalverfahren *Dynamic Time Warping (DTW)* – eine Suche nach Pfaden mit variabler Geschwindigkeit innerhalb der Differenzmatrix. Dieser Schritt ist rechenintensiv, da er quadratisch mit der Länge der Sequenzen skaliert.

## 2.4 Vorarbeiten und Modifikationen

Diese Arbeit baut auf den Ergebnissen einer vorangegangenen Praxisphase des Autors auf [Ruiz, 2025]. Ziel war die Adaption des SeqSLAM-Prinzips, spezialisiert auf die Hardware des humanoiden Roboters Myon für den Einsatz in Innenräumen.

### 2.4.1 Echtzeitfähigkeit durch Räumliche Abtastung

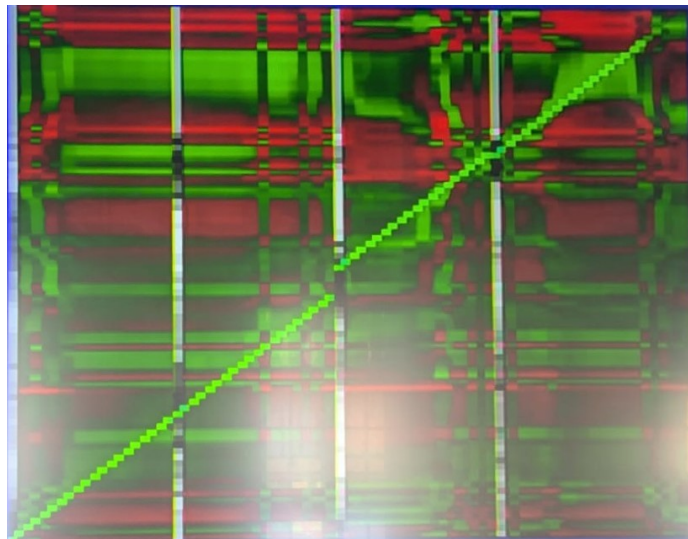
Während das originale SeqSLAM und verwandte Arbeiten wie [Sünderhauf et al., 2013] und [Aschenbrenner, 2015] auf zeitlicher Abtastung (z.B. 1 Bild/-Sekunde) basieren, wurde in den Vorarbeiten eine fundamentale Änderung der Architektur vorgenommen: der Wechsel zur räumlichen Abtastung (*Spatial Sampling*).

Anstatt Bilder zeitbasiert aufzunehmen, triggert die Odometrie des Roboters eine Aufnahme in festen Distanzintervallen (z.B. alle 20 cm). Diese Modifikation hat weitreichende Konsequenzen für die algorithmische Effizienz: Das Problem wird linearisiert, es entfällt deshalb die Notwendigkeit, Geschwindigkeitsunterschiede herauszurechnen. Dadurch kann auf rechenintensive Verfahren wie *Dynamic Time Warping* verzichtet werden. Der Vergleich reduziert sich auf eine lineare Suche bzw. das Scannen der Diagonale in der Differenzmatrix, was die Echtzeitfähigkeit auf der eingebetteten Hardware von Myon

gewährleistet. Das ist notwendig um parallel weitere Prozesse laufen lassen zu können.

### 2.4.2 Status Quo und Erweiterungen

In der Praxisphase wurde der Algorithmus sequenzweise implementiert, d.h. eine Positionsschätzung erfolgte erst nach Abschluss einer vollen Sequenz, danach begann eine neue Sequenz. Abbildung 1 zeigt die Ergebnisse dieser frühen Implementierung: Die hellgrüne Linie markiert den korrekt erkannten Pfad (die Winkelhalbierende) bei zwei Fahrten entlang derselben Route.



**Abbildung 1:** Ergebnisse der Vorarbeiten: Die Differenzmatrix zeigt niedrige und hohe Kosten in Grün- und Rottönen. Die vertikalen Linien stellen die Kosten für jeden Startpunkt der Sequenz dar, hell und dunkel stehen für teuer und günstig. Die hellgrüne Linie entlang der Diagonalen stellt den Weg mit den niedrigsten Kosten dar, was einer korrekten Lokalisierung entspricht [Ruiz, 2025].

Obwohl dieser Ansatz erste, vielversprechende Ergebnisse lieferte, zeigten sich Einschränkungen in der Flexibilität (keine bildweise Lokalisierung) und der Robustheit (kein Schwellenwert für unsichere Matches). Die vorliegende Bachelorarbeit erweitert diesen Ansatz daher um eine bildweise Auswertung, ein Konzept für eine Karte (Pose-Graph) und einen *Confidence-Threshold* zur Detektion von *Loop-Closures*. wie in den folgenden Kapiteln detailliert beschrieben wird.

## 3 Aufbau und Methodik

Dieses Kapitel beschreibt die verwendete Roboterplattform, die charakteristischen Eigenschaften der Versuchsumgebung sowie die Methodik der Datenerfassung. Um die Leistungsfähigkeit des entwickelten Algorithmus zu evaluieren, wurden Datensätze direkt auf der Zielhardware aufgenommen, welche Bilddaten und Odometrieeinformationen umfassen.

### 3.1 Die Konfiguration Myons

Als Versuchsplattform dient der humanoide Roboter Myon, in Abbildung 2 zu sehen. Für die Fortbewegung und Odometrie-Berechnung wird dessen so genannter „Unterbau“ genutzt, der über einen Differentialantrieb verfügt. Ebenso ist er mit Infrarot-Abstandssensoren ausgestattet, deren Daten für die spätere Kartenerstellung (siehe Kapitel 5.2) relevant sind, für den reinen Lokalisierungsalgorithmus jedoch nicht herangezogen werden.

Bei der hier vorgestellten Methode zur visuellen Lokalisierung werden die Odometriewerte des Systems genutzt, um die Bildaufnahme zu steuern. Anstatt Aufnahmen in festen zeitlichen Intervallen zu erstellen (z.B. 1 Bild pro Sekunde wie von

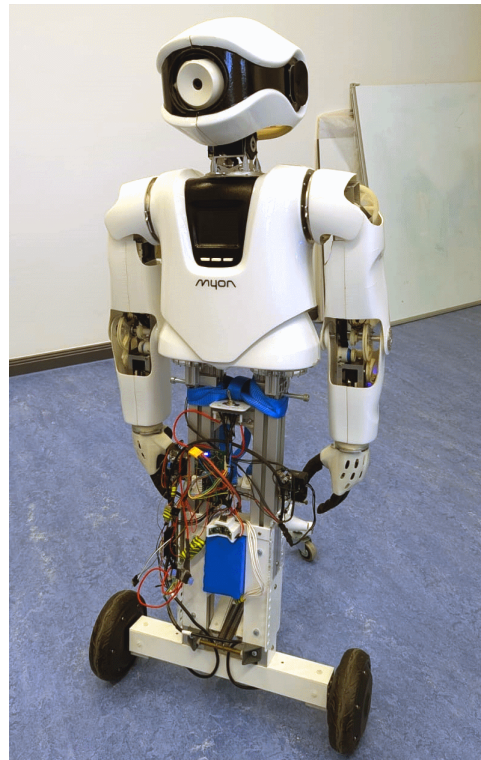


Abbildung 2: Myon mit Unterbau.

[Milford, 2013]), wird eine räumliche Abtastung (*Spatial Sampling*) implementiert. Konkret wird alle 20 cm (Euklidische Distanz) eine Aufnahme ausgelöst.

Das Kamerasystem basiert auf einer 1/4-Zoll CCD-Kameramodul der Videology 20K-Serie. Dieses liefert Bilddaten im YUV-Farbmodell mit einer Farbtiefe von 8 Bit. Eine Besonderheit der Plattform ist die nachgelagerte *Pixel Pipeline*, eine FPGA-basierte Vorverarbeitungseinheit. Diese übernimmt das Downsampling der Rohdaten zu sogenannten „Superpixeln“ direkt in Hardware, wodurch die CPU-Last für die Bildaufnahme minimiert wird. Das Kameramodul verwendet zudem eine interne Gammakorrektur von  $\gamma = 0,45$ , welche die Helligkeitswerte bereits an die menschliche Wahrnehmung anpasst [VIDEOLOGY, 2012]. Die Kamera ist in Fahrtrichtung ausgerichtet und um direkte Blendeffekte durch die Deckenbeleuchtung zu vermeiden ist die Blickrichtung um ca.  $10^\circ$  nach unten geneigt. Da der SeqSLAM-Ansatz primär auf Kontrast- und Strukturinformationen basiert, werden alle Bilder in Graustufen verarbeitet.

Die Aufnahmen werden simultan in verschiedenen Auflösungen gespeichert, um den Einfluss der Bildgröße zu untersuchen – falls die Robustheit ohne DTW stärker bei niedriger Auflösung leidet. Die Tabelle 1 gibt eine Übersicht der verwendeten Auflösungs-Modi.

**Tabelle 1:** Übersicht der verwendeten Bildauflösungen und ihrer Bezeichnungen.

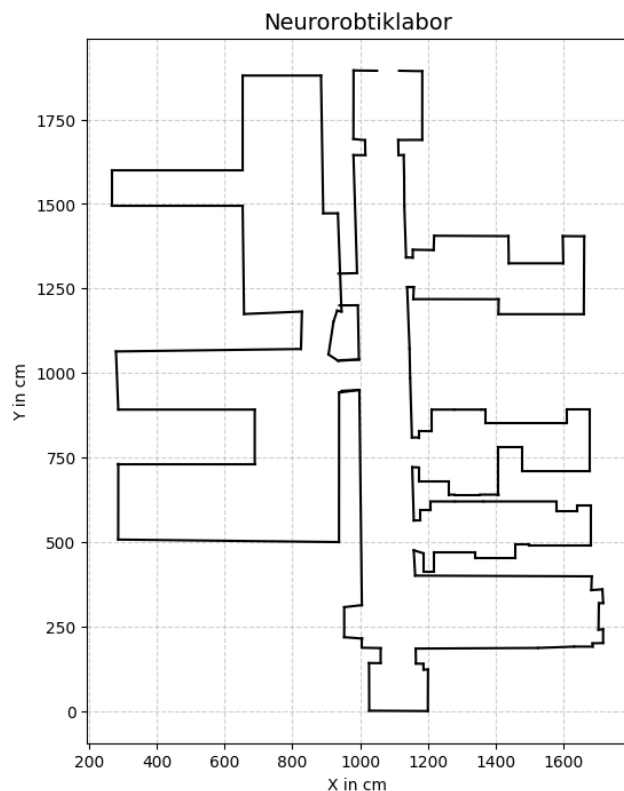
Modus	Auflösung [px]	Beschreibung
Standard	$32 \times 16$	Basis-Auflösung für Referenztests
Ausfüllend	$25 \times 20$	Optimiertes Seitenverhältnis
Hoch	$36 \times 28$	Maximale getestete Detaildichte, zusammengesetzt aus zwei Bildern
Niedrig	$12 \times 10$	Reduzierte Auflösung ("Handful of Bits")

## 3.2 Versuchsumgebung

Die Versuche wurden in einem Ingenieurlabor durchgeführt, welches sich durch eine stark repetitive visuelle Struktur auszeichnet. Viele weiße Wände, lange Flure und Arbeitsplätze mit identischem Aufbau (Stuhl, Monitor, weiße Tische) definieren die optische Grundlage. Einzigartige Vorkommen, wie ein roter Feuerlöscher oder blaue Schränke, stechen in dieser ansonsten reizarmen Umgebung nur vereinzelt hervor.

Abbildung 3 zeigt die verwendete *Ground Truth*, der vollständigen Grundriss der befahrbaren Umgebung, welcher aus der Arbeit von [Meißner, 2025] hervorgegangen ist. Die Struktur ist global nicht symmetrisch, was theoretisch eine eindeutige Zuordnung ermöglicht. Lokal besitzt sie jedoch eine hohe Ambiguität.

Die Datenerhebung erfolgte entlang einer semi-festen Route. Um die Robustheit des Algorithmus gegenüber realen Fahrfehlern zu testen, wurden Abweichungen von der Ideal-Route von bis zu  $\pm 50$  cm in der Position und bis zu  $\pm 25^\circ$  in der Orientierung zugelassen. Dies simuliert das Verhalten des Roboters in realen Navigationsszenarien, in denen er nicht exakt derselben Spur folgt.



**Abbildung 3:** Schematischer Grundriss der Versuchsumgebung (Forschungslabor Neurorobotik).

### 3.3 Datensätze

Um die Invarianz des Algorithmus gegenüber Beleuchtungsänderungen zu evaluieren, wurden Datensätze zu unterschiedlichen Tageszeiten und unter variierenden Lichtbedingungen aufgenommen.

Ein kritischer Faktor in der Versuchsumgebung sind die Fensterfronten zur Nord- und Südseite. Je nach Sonnenstand (morgens vs. nachmittags) entstehen starke Schlagschatten, Spiegelungen auf dem Boden oder Überbelichtungen durch direktes Sonnenlicht. Zusätzlich wurden Vergleichsfahrten mit und ohne künstliche Deckenbeleuchtung (DL) durchgeführt. Es ist zu erwarten, dass Datensätze mit konstanter Deckenbeleuchtung aufgrund der stabileren Lichtverhältnisse bessere Lokalisierungsergebnisse liefern als die mit rein natürlichem Lichteinfall.

Tabelle 2 fasst die Eigenschaften der verwendeten Datensätze zusammen. Die Helligkeit wurde dabei qualitativ auf einer subjektiven Skala von 1 (sehr dunkel) bis 10 (sehr hell) eingeordnet, um die relative Beleuchtungsstärke vergleichbar zu machen.

**Tabelle 2:** Übersicht der ausgewerteten Datensätze. Die Helligkeit ist eine subjektive Einordnung (1-10). „DL“ signalisiert aktive Deckenbeleuchtung.

Datensatz	Helligkeit	Bedingungen
1	3	Morgens, sonnig (tiefstehende Sonne)
2	DL	Mittags, sonnig mit wenigen Wolken, Kunstlicht an
3	DL	Morgens, sonnig, Kunstlicht an
4	9	Morgens, sehr helles Sonnenlicht
5	5	Vormittags, ca. eine Stunde nach Sonnenaufgang
6	DL	Vormittags, Kunstlicht an, Referenzfahrt zu Datensatz 5

## 4 Lokalisierung

Dieses Kapitel beschreibt die algorithmischen Komponenten des Lokalisierungssystems: Die Vorverarbeitung der Bilder zur Erhöhung der Invarianz gegenüber Beleuchtungsänderungen, die Berechnung einer Differenzmatrix als Maß für die visuelle Ähnlichkeit und schließlich die eigentliche Lokalisierung durch Sequenzvergleich.

### 4.1 Vorverarbeitung

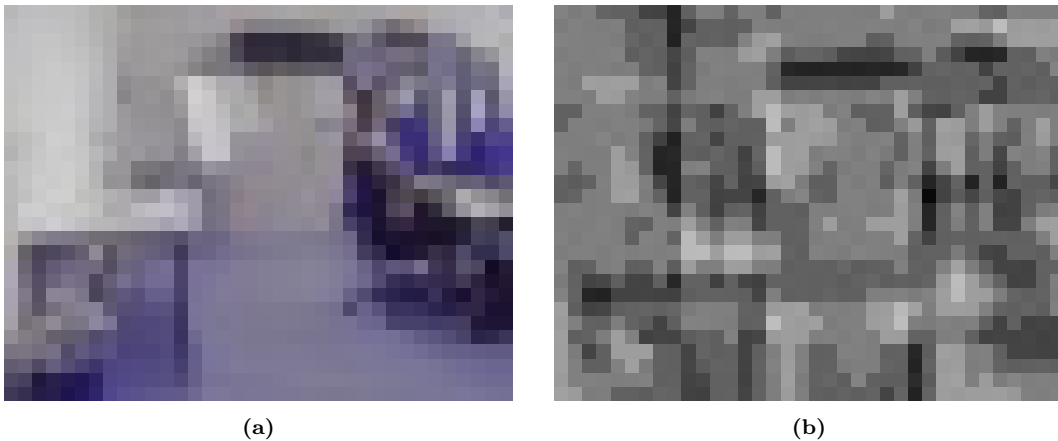
Die visuelle Erscheinung eines Ortes variiert drastisch bei Veränderungen der Lichtverhältnissen. Um eine robuste Wiedererkennung gewährleisten zu können, müssen die Rohbilder so transformiert werden, dass strukturelle Informationen erhalten bleiben, während beleuchtungsbedingte Varianz minimiert wird. Im Rahmen dieser Arbeit werden drei Verfahren und deren Kombinationen untersucht. Das von [Milford, 2012] vorgeschlagene SeqSLAM bleibt selbst bei Tag und Nacht Unterschieden robust durch die dort verwendete *Patch Normalization (PN)*. PN hat sich gut für die Anwendungen an Schienenstrecken und im Straßenverkehr erwiesen. In dieser Arbeit wird die Performance zwei weiteren simplen Vorverarbeitungsverfahren, dem Histogrammausgleich (HA) und inversen Gammakorrektur (iGK) und Kombinationen aus den drei Vorverarbeitungsmethoden verglichen, da in den bisherigen SeqSLAM Arbeiten andere Bildvorverarbeitungsmethoden nicht betrachtet wurden. In der Arbeit von [Shan et al., 2003] wurde die Erkennung einzelner Bilder (Gesichter) durch eine Kombination aus HA und iGK verbessert. Daher wird diesem Ansatz hier nachempfunden, in der Annahme dass so eine höhere Beleuchtungsinvarianz erreicht werden kann.

### 4.1.1 Patch Normalization

Die Anwendung der Patch Normalization erhöht die Robustheit des SeqSLAM-Algorithmus gegenüber extremen Erscheinungsänderungen. Die Kernherausforderung sind starke, räumlich inkonsistente Beleuchtungsschwankungen, wie sie durch harten Schattenwurf, unterschiedliche Wetterlagen oder den Tag-Nacht-Zyklus entstehen. Das Bild wird in ein Raster aus Regionen (*Patches*) mit möglichst gleichen Seitenlängen unterteilt, dabei sollten die einzelnen Patches mindesten 16 Pixel umfassen. Jeder Patch  $c$  wird lokal normalisiert:

$$\hat{p}_{c,(x,y)} = \frac{p_{c,(x,y)} - \mu_c}{\sigma_c + \epsilon} \quad (4.1)$$

wobei  $p_{c,(x,y)}$  der Pixelwert an der Stelle  $(x, y)$  im Patch  $c$  ist, und  $\mu_c$  bzw.  $\sigma_c$  den Mittelwert und die Standardabweichung des Patches bezeichnen. Der Term  $\epsilon$  verhindert eine Division durch Null in kontrastarmen Bereichen. Abschließend werden die Werte wieder in den 8-Bit-Bereich  $[0, 255]$  skaliert.



**Abbildung 4:** Darstellungen des Bild 143 aus dem Datensatz 3, links die originalen Bilddaten für die Auflösung „Hoch“ und rechts nach der Patch Normalization

Dieser Prozess unterdrückt die variablen Beleuchtungsbedingungen und hebt stattdessen die zugrunde liegende Struktur und Textur innerhalb jedes Patches hervor. In Abbildung 4 ist genau dieser Prozess zu erkennen, das Bild 4a wurde auf das Wesentliche reduziert in Abbildung 4b. Ein Nachteil ist die Verstärkung

von Rauschen in homogenen Flächen (z.B. weiße Wände).

### 4.1.2 Histogrammausgleich

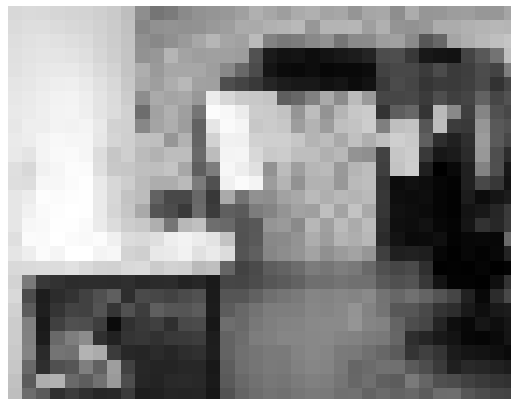
Der Histogrammausgleich ist ein globales Verfahren zur Kontrastmaximierung. Die Wahrscheinlichkeitsmassendichte (*PMF*) der Intensitätsverteilung wird berechnet durch

$$pmf(r) = \frac{h(r)}{N}, \quad (4.2)$$

wobei  $h(r)$  die Häufigkeit des Grauwerts  $r$  und  $N$  die Gesamtpixelzahl ist. Die kumulative Verteilungsfunktion (*CDF*) dient dann als Transformationsfunktion:

$$\hat{r} = \text{round} \left( 255 \cdot \sum_{j=0}^r pmf(j) \right). \quad (4.3)$$

Dies spreizt das Histogramm über den gesamten Wertebereich und macht Details in über- oder unterbelichteten Bildern sichtbar. Abbildung 5 zeigt diese Vorverarbeitungsmethode anhand des gleichen Bildes wie in 4b.



**Abbildung 5:** Das Bild 143 aus dem Datensatz 3 nach der inversen Gammakorrektur.

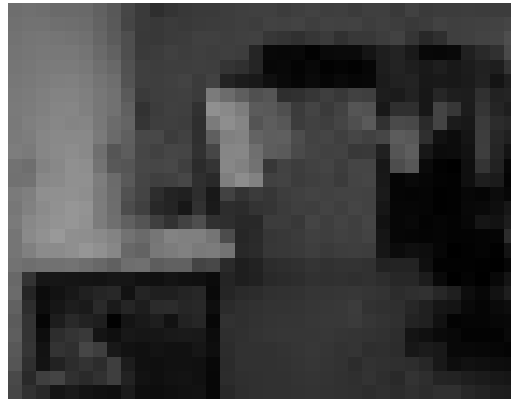
### 4.1.3 Inverse Gammakorrektur

Da das Kameramodul hardwareseitig eine Gammakorrektur von  $\gamma = 0,45$  anwendet (vgl. Kapitel 3), liegen die Bilddaten nicht linear zur physikalischen

Lichtintensität vor. Die inverse Gammakorrektur linearisiert den Farbraum durch

$$L_{linear} = L_{sRGB}^{\frac{1}{0.45}}. \quad (4.4)$$

Theoretisch ermöglicht dies eine physikalisch korrektere Weiterverarbeitung, führt jedoch bei 8-Bit-Bildern zu einem Informationsverlust in dunklen Bereichen (Quantisierungsrauschen).



**Abbildung 6:** Das Bild 143 aus dem Datensatz 3 nach der inversen Gammakorrektur.

Es wird erwartet, dass die iGK den beiden vorher beschriebenen Vorverarbeitungsmethoden eine höhere Beleuchtungsinvarianz verleiht.

## 4.2 Differenzenmatrix

Der Vergleich zwischen einem aktuellen Query-Bild  $i$  und einem Bibliotheks-Bild  $j$  (der Karte) erfolgt über die Summe der absoluten Differenzen (SAD), normiert auf die Bildfläche  $A = w \cdot h$  (MAD):

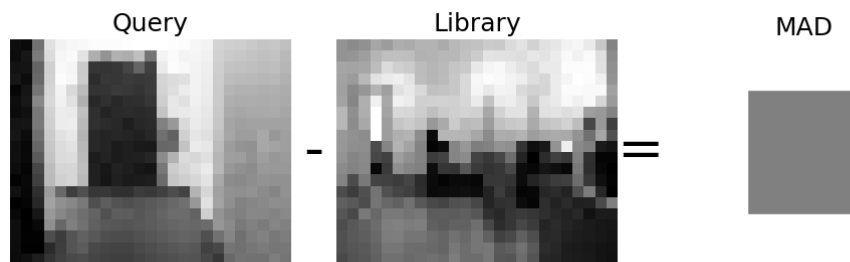
$$D_i(j) = \frac{1}{A} \sum_{x=0}^w \sum_{y=0}^h |p_{i,(x,y)} - p_{j,(x,y)}|, \quad (4.5)$$

wobei  $p$  die Pixelintensität an der Bildkoordinate  $(x, y)$  des Bildes entspricht. Der resultierende Matrix  $D_i$  enthält die Ähnlichkeitswerte aller möglichen Bildpaare. Ein essenzieller Schritt für die Robustheit des Algorithmus ist das lokale

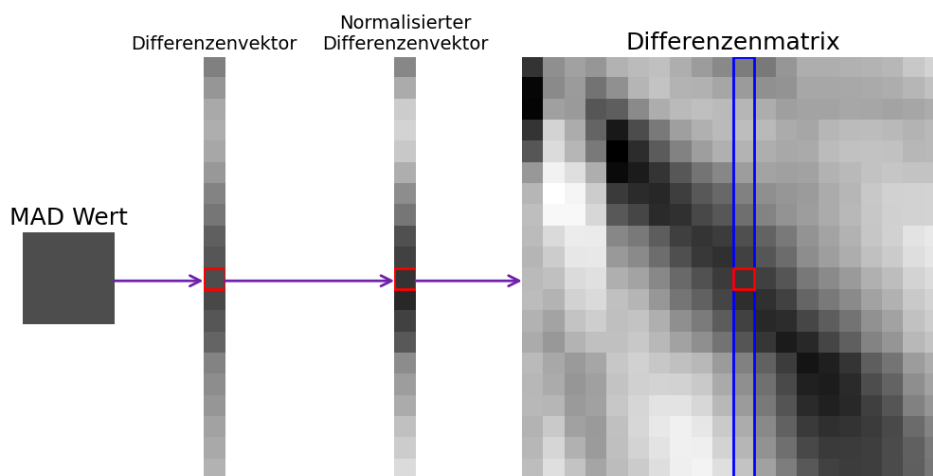
normalisieren laut [Sünderhauf et al., 2013] durch:

$$\hat{D}_i = \frac{D_i - \overline{D}_W}{\max(\sigma_W, \sigma_{\min})}, \quad (4.6)$$

wobei ein *sliding window* mit der Breite  $W$  den lokalen Durchschnitt  $\overline{D}_W$  und die lokale Standardabweichung  $\sigma_W$  erzeugt. Dies stellt sicher, dass ein „guter“ Match relativ zu seiner unmittelbaren Nachbarschaft bewertet wird. Die Konstante  $\sigma_{\min}$  sichert vor undefinierten Ergebnissen ab.



(a) Berechnung des mittleren absoluten Differenzwerts (MAD) zwischen einem Query-Bild und einem Bibliotheks-Bild. Der resultierende „Pixel“ repräsentiert den visuellen Abstand.



(b) Konstruktion der Differenzmatrix  $\hat{D}$ . Der einzelne MAD-Wert wird Teil eines Differenzvektors (Vergleich mit der gesamten Bibliothek), welcher lokal normalisiert und als Spalte in die Matrix eingefügt wird.

**Abbildung 7:** Schematische Darstellung des Prozesses von Einzelbildvergleichen zur Differenzmatrix. (a) zeigt die Berechnung auf Bildebene, (b) die Integration in die Matrixstruktur.

Abbildung 7 visualisiert den gesamten Prozess: Zunächst wird die pixelweise Differenz zu einem einzelnen Skalar zusammengefasst (a), bevor dieser Wert im Kontext seiner Nachbarschaft normalisiert und in der Matrixstruktur hinzugefügt wird (b).

### 4.3 Odometrie

Odometrie bezeichnet die Schätzung der Position und Orientierung relativ zum Startpunkt, welche gemeinsam als Pose bezeichnet werden. Dieser Prozess basiert auf der Integration von Bewegungsdaten über die Zeit. Die Radmotoren Myons sind mit hochauflösenden Drehgebern ausgestattet, die eine volle Radumdrehung in 4096 Schritte abbilden, welche auch *Ticks* genannt werden. Die Odometrie dient in diesem System nicht nur der Aufnahmesteuerung, sondern liefert auch eine initiale Posen-Schätzung  $(x, y, \varphi)$  für den Pose-Graphen (siehe Kapitel 5).

Die Berechnung erfolgt mittels expliziter Euler-Integration der Encoder-Daten des Differentialantriebs. Diese Messung ist kurzfristig sehr präzise, da die verwendeten Gummireifen den Schlupf auf ein Kleinstes reduzieren. Für einen Zeitschritt  $\Delta t$ , in Myons Fall 10 ms, mit den zurückgelegten Rad-Distanzen  $\Delta d_L$  und  $\Delta d_R$  gilt:

$$\Delta d_{center} = \frac{\Delta d_R + \Delta d_L}{2} \quad (4.7)$$

$$x_{t+1} = x_t + \Delta d_{center} \cdot \cos(\varphi_t) \quad (4.8)$$

$$y_{t+1} = y_t + \Delta d_{center} \cdot \sin(\varphi_t) \quad (4.9)$$

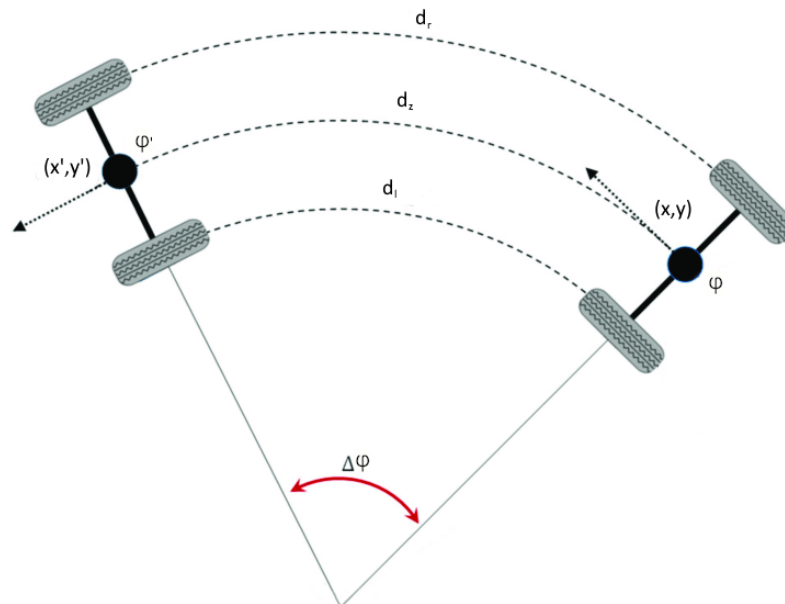
$$\varphi_{t+1} = \varphi_t + \frac{\Delta d_R - \Delta d_L}{L}, \quad (4.10)$$

wobei  $L$  die Spurbreite des Roboters ist, Myons beträgt 47,4 cm. Wie in Kapitel 6.5 analysiert wird, ist dieses Verfahren anfällig für Drift, insbesondere durch systematische Fehler im Parameter  $L$  und numerische Ungenauigkeiten bei der Integration von Rotationen. Die Distanzen  $d$  sind eine Umrechnung von

Encoderwerten zu Zentimetern und wird berechnet durch:

$$D = U \cdot \frac{\Delta E_{tick}}{N}, \quad (4.11)$$

wobei  $N$  die Anzahl der Ticks für eine volle Umdrehung mit dem Radumfang  $U$  ist. Die Variablen der Formeln sind in Abb. 8 visualisiert.



**Abbildung 8:** Visualisierung der Odometrie für einen sehr großen Zeitschritt  $\Delta t$ .

Diese Akkumulation der kleinen Bewegungs-Updates durch Integration löst das fundamentale Problem der Odometrie aus. Kleine Fehler werden bei der Integration ebenfalls akkumuliert, dieses Phänomen wird Drift genannt. Die Fehlerquellen dafür sind vielfältig:

1. Systematische Fehler: Ungenauigkeiten im Modell, z.B. wenn der gemessene Radabstand  $L$  oder die Radumfänge leicht von der Realität abweichen. Ein Roboter, der „gradeaus“ fahren soll, also gleiche Ticks auf beiden Rädern vorgegeben bekommt, würde in der Realität eine leichte Kurve fahren, während die Odometrie eine perfekte Gerade berechnet.
2. Nicht-systematische Fehler: Physikalische Interaktionen, die nicht vom

Modell erfasst werden, wie Unebenheiten im Boden, das Überfahren von Kanten und Schlupf.

3. Numerische Fehler: Minimale Rundungsfehler bei Fließkommaberechnungen sowie Fehler, die durch die diskrete Natur der Euler-Integration selbst entstehen.

Obwohl jeder einzelne Fehler vernachlässigbar klein ist, führt die Integration über Hunderte von Metern dazu, dass die berechnete Odometrie-Position signifikant von der wahren Position abweicht. Die Odometrie ist daher kurzfristig sehr genau, aber langfristig unzuverlässig. Die Fehler werden in Kapitel 6.5 betrachtet.

## 4.4 Lokalisierung durch Sequenzvergleich

Der Kernbeitrag dieser Implementierung ist die Anpassung des SeqSLAM-Algorithmus an die räumliche Abtastung.

### 4.4.1 Verzicht auf Dynamic Time Warping

Die Verwendung der Odometriewerte spielt eine essenzielle Rolle bei der Reduktion des Rechenaufwands von SeqSLAM und entspricht dem Kern der *“how low can you go?”*-Philosophie. Wie in Kapitel 3 eingeführt werden die Bilder anhand räumlicher, anstatt zeitlicher Abtastung verglichen. Diese Herangehensweise normalisiert die Datensequenzen von vornherein im räumlichen Bereich. Geschwindigkeitsunterschiede zwischen verschiedenen Durchläufen werden dadurch effektiv eliminiert. Da die Aufnahmen an identische Abstände gebunden sind, spielt die pro Distanz benötigte Zeit keine Rolle mehr.

Bei einer zeitlichen Abtastung unterschiedlicher Geschwindigkeiten kommt es zwangsläufig zu nicht-linearen Verzerrungen zwischen den Sequenzen. Um eine korrekte Übereinstimmung zu finden, muss ein Alignment-Algorithmus wie *Dynamic Time Warping (DTW)* eingesetzt werden, der diese Verzerrungen kompensiert. DTW ermittelt dabei einen optimalen, nicht-linearen Pfad durch

eine Kostenmatrix, um den visuellen Abstand zwischen den beiden Sequenzen zu minimieren.

Der Hauptnachteil von DTW ist jedoch die hohe algorithmische Komplexität, die quadratisch mit  $N$  und  $M$  skaliert -  $O(N \times M)$ . Im Fall von SeqSLAM wächst die Komplexität linear, da eine feste Sequenzlänge für  $N$  verwendet wird und nur die Bibliothek  $M$  bei Erweiterung der Karte wächst. Trotzdem wird in [Milford, 2012] angemerkt, dass dieser Schritt die rechenintensivste Komponente des gesamten Lokalisierungsalgorithmus ist, insbesondere weil längere Sequenzen bessere Resultate liefern (größeres  $N$ ).

Durch die räumliche Abtastung wird die Notwendigkeit für Dynamic Time Warping umgangen. Stattdessen wird die am besten übereinstimmende Sequenz der Länge  $S$  durch ein *Sliding-Window* direkt über die Differenzmatrix  $\hat{D}$  ermittelt. Für jedes Query-Bild  $i$  wird ein Kostenvektor  $P_i$  berechnet, wobei  $i \geq S$  gilt. Dieser Vektor  $P$  enthält die Kosten für die Übereinstimmung mit allen gültigen Startpositionen  $j_x$  in der Bibliothek mit der Länge  $L$ , mit  $x \in [L, L+1, \dots, S-1, S]$ , da keine vollständige Sequenz für die ersten  $S$  Indizes gebildet werden kann.

Dadurch reduziert sich das Problem der Sequenzsuche auf eine einfache Summation entlang der Diagonalen der Differenzmatrix.

#### 4.4.2 Berechnung der Sequenzkosten

Für jedes aktuelle Bild  $i \geq S$  werden die Kosten  $P_i$  für die Annahme berechnet, dass es dem Bibliotheksbild  $j$  entspricht. Dazu werden die Differenzwerte der vergangenen  $S$  Bilder aufsummiert durch

$$P_i(j_x) = \sum_{k=0}^{S-1} \hat{D}(i-k, j_x-k). \quad (4.12)$$

Der Index  $j_{i,min}$  mit den geringsten Sequenzkosten gilt als beste Schätzung der aktuellen Position („Best Match“):

$$j_{i,min} = \arg \min_{j_x} (P_i(j_x)). \quad (4.13)$$

### 4.4.3 Confidence-Score

Um Perzeptuelles Aliasing zu erkennen, wird ein Confidence-Score  $C_i$  berechnet. Dieser setzt den besten Match  $P_i(j_{i,min})$  ins Verhältnis zum zweitbesten Match, der außerhalb eines Toleranzfensters  $R$  um das Optimum liegt. Dieses Fenster verhindert, dass direkte Nachbarn des besten Treffers (die zwangsläufig auch ähnlich aussehen) als "Konkurrenz" gewertet werden.

Zunächst wird ein maskierter Kostenvektor  $P_i^*$  definiert, in dem das Fenster um den besten Treffer ausgeblendet wird:

$$P_i^*(j_x) = \begin{cases} \inf, & \text{wenn } |j_x - j_{i,min}| \leq R, \\ P_i(j_x), & \text{sonst.} \end{cases} \quad (4.14)$$

Anschließend wird der minimale Wert dieses maskierten Vektors gesucht, was dem besten Match außerhalb des Toleranzbereichs entspricht:

$$j_{i,min}^* = \arg \min_{j_x} (P_i^*(j_x)). \quad (4.15)$$

Der Confidence-Score  $C_i$  ergibt sich schließlich aus dem Verhältnis der beiden Kostenwerte:

$$C_i = \frac{P_i(j_{i,min})}{P_i(j_{i,min}^*)} \quad (4.16)$$

Dieser Wert liegt im Intervall  $[0, 1]$ . Ein Wert nahe 1 indiziert eine hohe Unsicherheit (der zweitbeste Ort ist fast genauso wahrscheinlich wie der beste), während Werte gegen 0 eine hohe Eindeutigkeit signalisieren. In Kapitel 7 wird basierend hierauf ein Threshold definiert.

## 5 Mapping

Bisher wurde der Algorithmus als reines Lokalisierungssystem beschrieben, dieses Kapitel liefert das theoretische Konzept zur Kartenerstellung, welches der Mapping-Komponente (das „M“ in SLAM) entspricht. Die Ergebnisse in den folgenden Kapiteln beziehen sich auf den Lokalisierungspart des Algorithmus, benötigen jedoch diesen Kontext. Bisher wird eine neue Sequenz mit einer vordefinierten Bibliothek verglichen. Ein Treffer bedeutet sich innerhalb dieses statischen Rahmens wiederzufinden. In einem vollständigen Algorithmus ist die Umgebung anfangs unbekannt, und eine Karte muss dynamisch während der Fahrt erstellt werden. Die Karte erlaubt dann Odometriedrift durch *Loop Closure* auszugleichen und Routenplanung vorzunehmen, also gezielt bekannte oder unbekannte Orte anzusteuern. Loop Closure geschieht, wenn Myon einen bereits erkundeten Ort durch die Lokalisierung in Kapitel 4 wiedererkennt. Loop Closure ist ein essenzieller Bestandteil von SLAM Algorithmen. [Cadena et al., 2017] beschreibt einen Roboter der Odometrie auswertet ohne Loop Closures zu beachten als „Roboter der seine Umwelt als einen „unendlichen Korridor“ interpretiert“ .

### 5.1 Pose-Graph

Eine für SeqSLAM geeignete Kartenform ist ein *Pose-Graph*. Das ist eine topologische Kartendarstellung, im Gegensatz zur folgend beschriebenen, metrischen Gridmap. Hier repräsentieren Knoten (*Nodes*) einen Ort, sowohl visuell als auch geometrisch - ein Bild zusammen mit der Posenschätzung der Odometrie ( $P_i = (x_i, y_i, \varphi_i)$ ).

Die Kante (*Constraint*) zwischen zwei Knoten stellt deren räumliche Beziehung zueinander dar. Dabei wird unterschieden in zwei Arten von Kanten:

1. Odometrie-Constraints: Das sind Kanten, die zwei aufeinanderfolgende Knoten durch Integration der Odometrie als Differenz der Posen darstellt (z.B.  $P_4$  wird mit  $P_5$  durch die Kante  $e_{4,5}$  als Transformation  $(\Delta x_{4,5}, \Delta y_{4,5}, \Delta \varphi_{4,5})$  verbunden).
2. Loop Closure-Constraints: Diese Kanten verbinden zwei Knoten die nicht aufeinander folgen, wenn ein Ort wiedererkannt wird (z.B.  $P_{150}$  und  $P_8$ ).

Dieser zweite Typ von Kanten hat ideeller Weise die Form  $e_{i,j} = (0, 0, 0)$ , da zwei identische Orte zu einem „Ring“ verknüpft werden, unter der Annahme, dass der visuelle Match den identischen physischen Ort repräsentiert. In der Realität wird die berechnete Strecke einen beliebig großen Fehler angesammelt haben, um diesen zu minimieren, wird *Pose-Graph-Optimization* (PGO) angewendet. Der akkumulierte Fehler wird nun rückwirkend auf alle Posen entlang des Rings verteilt, sodass eine konsistente Karte gebildet wird und die gefahrene Trajektorie „einrastet“. Der Prozess wird ausführlich in der Arbeit von [Grissetti et al., 2011] beschrieben.

Wenn dieser Schritt durchgeführt wird für ein False Positive, wird eine fehlerhafte Loop Closure-Constraint eingetragen und PGO wird gezwungen den Graph anzupassen, was oft zu einer irreparablen Korruption der gesamten Kartenstruktur führt. Daher ist ein akkurater Lokalisierungsalgorithmus eine kritische Voraussetzung für einen erfolgreichen SLAM-Algorithmus. In Kapitel 7 wird näher darauf eingegangen. Der Pose-Graph allein ist für die Navigation und Pfadplanung ungeeignet, da er primär die Trajektorie des Roboters optimiert, nicht aber die detaillierte Geometrie der Umgebung speichert.

## 5.2 Gridmap

Für diese Aufgabe wird eine *Gridmap*, oder Belegungsrasterkarte verwendet (formalisiert von [Elfes, 2002]). Eine Gridmap teilt die Umgebung in ein Gitter aus diskreten Zellen (z.B. 10x10 cm) auf. Jede Zelle speichert einen Wahrscheinlichkeitswert, mit der diese Zelle als „belegt“ (z.B. durch eine Wand oder einen Tisch) oder „frei“ angesehen wird.

Um die Karte zu füllen, werden die Abstandssensoren Myons verwendet. Bei jeder Pose wird die Messung der Sensoren (z.B. "Hindernis in 90 cm") in das Gitter projiziert und die Wahrscheinlichkeiten der betroffenen Zellen aktualisiert.

Die größte Herausforderung hierbei ist der Odometriefehler. Würde die Gridmap direkt mit den Roh-Odometriedaten aufgebaut werden, würde der Drift zu „verschmierten“ Wänden und inkonsistenten Raumstrukturen führen. Hier schließt sich der Kreis zum Pose-Graph: Die Gridmap wird nicht mit den fehlerbehafteten, sondern mit den optimierten Posen aus dem PGO-Prozess erstellt. Wenn eine Loop Closure die Trajektorie korrigiert, werden die Sensormessungen relativ zu den neuen Posen in das Gitter eingetragen. Das Ergebnis ist eine konsistente 2D-Karte der Umgebung, die als Grundlage für die Routenplanung dienen kann.

## 6 Algorithmus Auswertung

Der Algorithmus wurde anhand von 6 Datensätzen, die in Kapitel 3.3 beschrieben sind, getestet. Die Evaluation des Algorithmus basiert auf dem Vergleich der generierten Ergebnisse mit einer manuell erstellten Referenz (Ground Truth). Die Auswertung wurde mit Python Skripts durchgeführt, welche auf GitHub zusammen mit den Datensätzen verfügbar ist (siehe [Ruiz Holtgreffe, 2025]).

Die manuelle Annotation der korrekten Matches unterliegt einer systembedingten Ungenauigkeit von bis zu  $\pm 1$  Bild (entsprechend  $\pm 20$  cm). Um diese Unsicherheit in den Referenzdaten zu berücksichtigen, wurde für die Bewertung ein Toleranzfenster definiert. Eine vom Algorithmus als übereinstimmend befundene Sequenz wird Treffer genannt. Ein detektierter Treffer wird als korrekt gewertet, wenn er innerhalb einer Abweichung von  $\pm 1$  Bild vom manuell annotierten Referenzpunkt liegt. Diese Vorgehensweise stellt sicher, dass die Ungenauigkeit der Referenzerstellung die Bewertung der tatsächlichen Algorithmusleistung nicht unverhältnismäßig negativ beeinflusst. Im ungünstigsten Fall ergibt sich aus der Kombination der Annotationsungenauigkeit ( $\pm 1$  Bild) und dem Toleranzfenster ( $\pm 1$  Bild) eine maximal akzeptierte Abweichung von 2 Bildern (40 cm) zwischen dem Ergebnis des Algorithmus und der tatsächlichen Position.

Die Effektivität des Algorithmus wird anhand von *Precision-Recall*-Kurven bewertet. Precision bezieht sich dabei auf die Genauigkeit der geschätzten Position und wird bestimmt mit

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}, \quad (6.1)$$

wobei *True Positives* die Anzahl der als korrekt und *False Positives* die Anzahl

der als falsch gewerteten Treffer ist. Recall beschreibt den gefundenen Anteil der möglichen korrekten Treffer mit

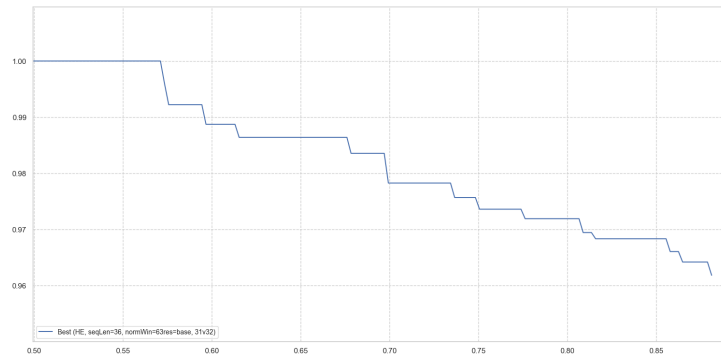
$$\text{Recall} = \frac{\text{TruePositives}}{\text{TruePositives} + \text{FalseNegatives}}, \quad (6.2)$$

wobei *False Negatives* die Anzahl der richtigen Treffer ist, die der Algorithmus nicht gefunden hat. Im Fall dieses Algorithmus hat jedes Bild einen entsprechenden Treffer, wodurch die Summe aus True Positives und False Negatives konstant ist:  $\text{TruePositives} + \text{FalseNegatives} = j_n - S$ , da die ersten Bilder mit einem Index kleiner als die Sequenzlänge noch keine Sequenz an Bildern zur Berechnung der Position haben. Dadurch kann ein Recall-Wert von 1,0 nicht erreicht werden.

Ein Precision-Recall-Diagramm visualisiert diesen Kompromiss und zeigt die Präzision über alle möglichen Recall-Stufen. Zur quantitativen Bewertung wird häufig die Fläche unter der Precision-Recall-Kurve (AUC-PR) herangezogen, welche sowohl hohe Präzision als auch hohen Recall belohnt. Abbildung 9 zeigt die Precision-Recall-Kurve des insgesamt besten Einzeldurchlaufs über alle getesteten Parameterkombinationen und Datensatzpaare hinweg. Die spezifischen Parameter für dieses Ergebnis waren:

- Bildvorverarbeitung: Histogrammausgleich (HA)
- Sequenzlänge: 36 Bilder
- Fenstergröße: 63 Differenzen
- Auflösung: standard (32×16 Pixel)
- Vergleichene Datensätze: Bibliothek 2 vs. Query 3

Wie in der Abbildung zu sehen ist, hält dieser Lauf eine Präzision von nahezu 100 % bis zu einem Recall von etwa 57 % aufrecht. Selbst bei einem Recall von 85 % liegt die Präzision noch bei über 96 %. Der maximal erreichte Recall für diesen spezifischen Durchlauf beträgt ca. 88 %. Die stufenartige Form der Kurve resultiert direkt aus der Art, wie die Metriken berechnet werden.



**Abbildung 9:** Die beste Performance aus allen getesteten Parametern und Bibliotheken.

Der Algorithmus sortiert zunächst alle gefundenen Treffer nach ihrer Vertrauenswürdigkeit, von der besten zur schlechtesten. Anschließend wird die Kurve Punkt für Punkt aufgebaut, indem immer mehr dieser Treffer berücksichtigt werden:

- **Horizontale Segmente:** Wenn der Algorithmus mehrere aufeinanderfolgende Treffer auswertet, die *nicht korrekt* sind (False Positives), erhöht sich zwar die Anzahl der betrachteten Treffer, aber nicht die Anzahl der *korrekt gefundenen* (True Positives). Dadurch bleibt der Recall-Wert (der Anteil der insgesamt gefundenen korrekten Treffer) konstant, während die Precision (der Anteil korrekter Treffer unter den bisher betrachteten) tendenziell sinkt. Dies führt zu einem horizontalen Verlauf in der Kurve.
- **Vertikale Segmente (Sprünge):** Sobald der Algorithmus in der sortierten Liste auf einen *korrekten Treffer* (True Positive) stößt, erhöht sich die Anzahl der korrekt gefundenen Treffer sprunghaft. Dies führt zu einem Anstieg des Recall-Wertes (ein Sprung nach rechts im Diagramm). Gleichzeitig kann sich auch die Precision ändern (oft ein kleiner Sprung nach oben, gefolgt von einem weiteren Abfall), was zu den vertikalen Kanten in der Kurve führt.

Die Kurve verbindet diese Punkte miteinander und visualisiert so, wie sich die Präzision verändert. Der Algorithmus bewegt den Recall durch das finden von weiteren True Positives. Die erzielte Leistung ist außergewöhnlich hoch,

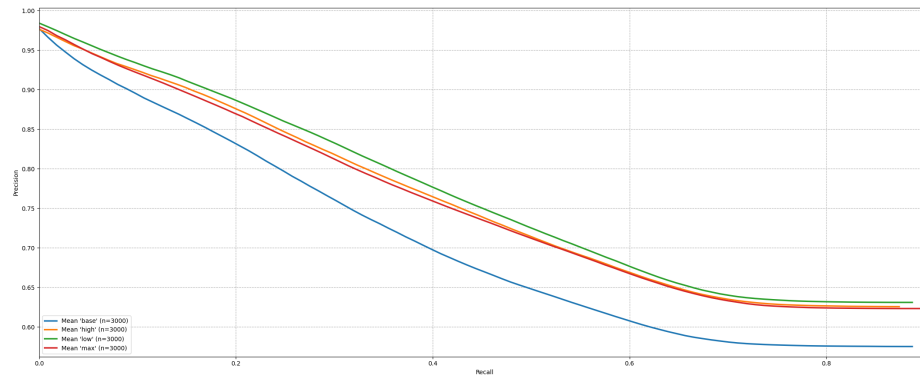
**Tabelle 3:** Eine Übersicht der getesteten Parameter und ihrer Werte.

Parameter	Erklärung	Wert
Bildauffösung	In Kapitel 3.1 beschrieben.	Standard, Ausfüllend, Hoch, Niedrig
Bild-Vorverarbeitung	In Kapitel 4.1 beschrieben.	PN, HA, iGK, HA-PN, iGK-PN, iGK-HA
Sequenzlänge	Anzahl der Bilder, die zur Bewertung aller möglichen Pfade herangezogen werden.	10, 20, 30, 40, 50
Fenstergröße	Anzahl der Werte im Schiebefenster über die Differenzenmatrix, die zur Normalisierung verwendet werden.	0, 7, 10, 15, 30, 50

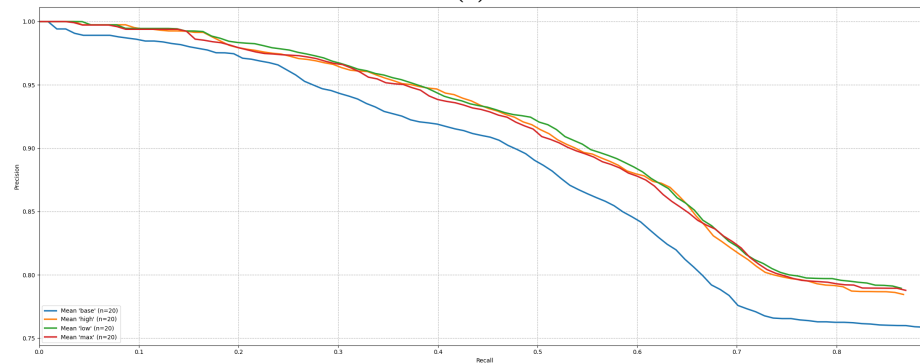
da selbst bei einem hohen Recall-Wert von 88 % (was fast 92 % der möglichen Treffer entspricht) noch eine Präzision von über 96 % gehalten werden kann. Die konkret verwendeten Parameter werden folgend verglichen, wobei Tabelle 3 eine Übersicht zu allen getesteten Parametern bietet.

## 6.1 Bildauflösungen

In der Arbeit von [Milford, 2013] wurden verschiedene Auflösungen bereits ausgiebig getestet, mit dem Ziel eine möglichst niedrige Auflösung zu finden, die weiterhin robuste Resultate liefert. Jedoch wurde mit einer 360 deg-Kamera gearbeitet. Da das Sichtfeld (FOV) auf ca. 60 deg beschränkt und die Datensätze fundamental unterschiedlich sind, werden hier verschiedene Auflösungen erneut betrachtet. Um die reduzierte Auflösung „Niedrig“ zu erzeugen, wurde lineares *Downsampling* genutzt. Die Graphen in den folgenden Abbildungen bilden den Durchschnitt aus allen, für die gewählten Parameter verfügbaren, Durchläufen der Anzahl  $n$ . Die abgebildete maximale Recallrate ist die, die maximal in den verglichenen Datensätzen erreicht wurde. In der Abb. 10a werden die verschie-



(a)



(b)

**Abbildung 10:** Precision-Recall Rates für verschiedene Bildauflösungen (base: Standard, max: Ausfüllend, high: Hoch, low: Niedrig). Wobei ein höheres Sichtfeld der Kamera höhere Auswirkung auf die Performance hat als die Auflösung (base vs. low). In der unteren Grafik werden die festen Parameter HA, Sequenzlänge= 30 und Fenstergröße= 50 verwendet.

denen Bildauflösungen für alle Parameteroptionen verglichen. In der Grafik fällt schnell auf, dass die Leistung deutlich schlechter ist als in Abb. 9: Bei einem Recall von 50 % liegt die höchste Precision bereits nur noch bei 73 % und erreicht bei 88 % Recall nur noch 63 % Precision. Grund dafür ist die Mischung aller getesteten Bildvorverarbeitungsverfahren (insbesondere iGK) und die schlechtere Leistung des Algorithmus bei weniger gut beleuchteten Datensätzen. Beim Vergleich der Bildauflösungen fällt auf, dass die Auflösungen *low*, *max* und *high* keine signifikante Verbesserung der Performance aufweisen, obwohl die Auflösungen *high* und *low* sich fast um ein vierfaches unterscheiden.

Daraus lässt sich schließen, dass die relevante Informationsdichte in der

Länge der Sequenzen liegt. Ebenso lässt sich daraus schließen, dass niedrigere Auflösungen bei niedrigen Sequenzlängen schlechter abschneiden, weil der Mangel an Informationsdichte nicht kompensiert werden kann. Beachtlich ist jedoch der Unterschied von *base* zu *max*, wobei beide eine ungefähr gleiche Auflösung von etwa 500 Pixeln haben und sich nur relevant in ihrem FOV unterscheiden. Die *base* Auflösung hat oben und unten im Bild ein jeweils etwa  $11^\circ$  geringeres Sichtfeld, was insgesamt einer Reduktion um etwa 25 % entspricht. Diese zusätzliche Information, wenn auch nur gemittelt in bereits existierenden Pixeln, verbessert die Precision von 57 % auf 63 %. In Abbildung 10a wird noch deutlicher, dass mit gut gewählten Parametern (in den folgenden Kapiteln erläutert) die Auflösung der Bilder eine noch geringere Rolle spielt.

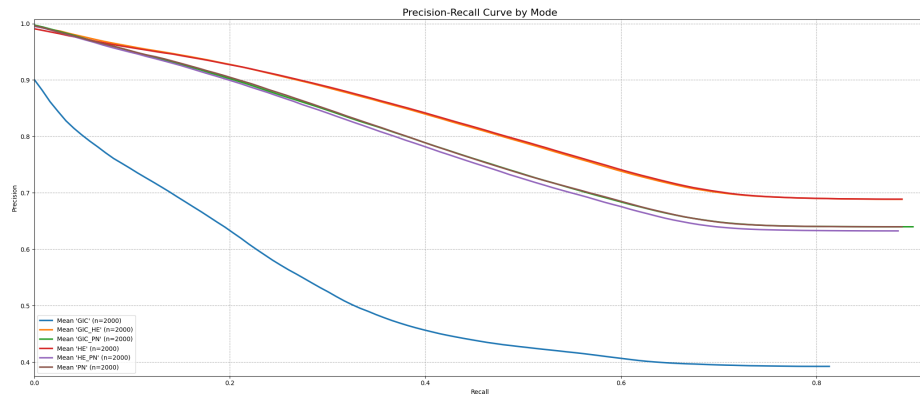
Für die Implementierung ist die niedrigste verfügbare Auflösung mit dem größtmöglichen FOV die effizienteste Wahl.

## 6.2 Bildvorverarbeitung

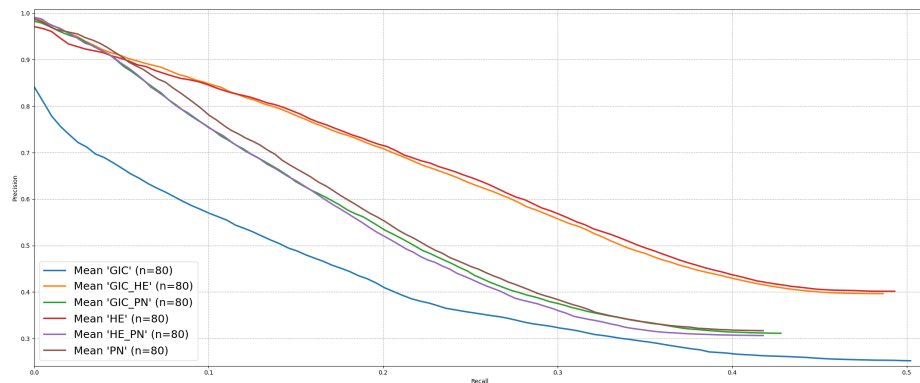
Da in der Arbeit von [Milford, 2013] nur Patch Normalization betrachtet wurde als Ausgleich von starken Änderungen der Lichtbedingungen, werden hier weitere Methoden betrachtet. In Abb. 11a fällt als erstes auf, dass inverse Gammakorrektur deutlich schlechter abschneidet als alle anderen Methoden: Maximal 82 % Recall mit 39,5 % Precision und selbst bei 1 % Recall nur 90 % Precision. Zu erklären ist das durch die fehlende Anpassung veränderter Lichtverhältnisse, die Pixelwerte werden nur nicht-linear verschoben und durch Informationsverlust der mit iGK verursacht wird, das Quantisierungsrauschen. Bei genauer Betrachtung fällt ebenso auf, dass der Histogrammausgleich die besten Ergebnisse liefern mit maximal 69 % Precision bei 87 % Recall und 79 % Precision bei 50 % Recall.

Jedoch unterscheiden sich die Ergebnisse nur marginal von HE mit iGK, sodass iGK sich als irrelevanter Faktor erweist. Bestätigt wird das durch den Vergleich von PN und iGK mit PN, da diese auch (fast) identische Ergebnisse liefern. Die Annahme, dass der globale HA und die lokale PN sich ergänzen, bestätigt sich also nicht. Erklären lässt sich das durch eine doppelte Verstärkung

von Rauschen, besonders in kontrastarmen Bildregionen. Ebenso kann HA in Bildern mit hohen Konzentrationen heller und dunkler Pixel, also einem „Ü-Förmigen“ Histogramm, zuerst lokale Kontraste reduzieren und dann mit PN weniger aussagekräftige Patches berechnen. In der Laborumgebung mit vielen weißen Wänden, Möbeln und schwarzen Stühlen und Monitoren kommt dieses Szenario häufig vor. Die Abb. 11b zeigt, dass bei kürzeren Sequenzen HA



(a)



(b)

**Abbildung 11:** Precision-Recall Raten für verschiedene Bildvorverarbeitungen (Patch Normalization, inverse Gammakorrektur, Histogrammausgleich und drei der möglichen Kombinationsmöglichkeiten). Die Ergebnisse mit PN und HA sind deutlich besser als mit iGK, die Kombinationen scheinen keine besondere Auswirkung auf das Ergebnis zu haben. In der Grafik (b) beträgt der Parameter Sequenzlänge= 10. HA sticht als beste Methode hervor.

auch bessere Ergebnisse liefert als PN, jedoch insgesamt alle deutlich schlech-

ter abschneiden - die maximalen Recallwerte sind um ca. 45% reduziert und die Precision ist an jedem Punkt niedriger. Bezogen auf den Vergleich von PN und HA bedeutet dies, dass HA öfter True Positives produziert. PN verliert globale Kontraste durch das lokale Normalisieren im Bild, wodurch häufiger ganze Sequenzen *False Positives* werden. Auf Basis dieser Ergebnisse ist die Verwendung des Histogrammausgleichs über Patch Normalization zu empfehlen.

### 6.3 Analyse der Sequenzlänge

Der wichtigste Parameter, die Sequenzlänge, hat nicht nur Auswirkung darauf wie gut der Algorithmus die Position bestimmen kann, sondern auch auf zeitliche und räumliche Verzögerungen. Bei einer Sequenz von 20 Bildern, die im Abstand von 20 cm aufgenommen werden, ist eine Sequenz vollständig nach vier Metern. Bis Myon die vier Meter hinter sich gebracht hat, gibt der Algorithmus keine Schätzung über seine Position ab.

Von diesem Punkt aus betrachtet, ist es naheliegend, dass längere Sequenzen (bei quasi-identischen Fahrten) zu besseren Resultaten führen. Der Grund warum von [Milford, 2013] eine Sequenzlänge von 20 Bildern empfohlen wurde, liegt an der Abwägung zwischen Rechenleistung und Precision-Recall. Mit der Modifikation des Algorithmus, jetzt räumlich abzutasten, ist eine Vergrößerung der Sequenzlänge nicht mehr das *Bottleneck* der Rechenleistung, da die quadratisch wachsende Komplexität des Algorithmus wegfällt. Der Anwendungsfall ist jedoch ein anderer, weshalb trotzdem eine Abwägung bei der Länge der Sequenzen gemacht werden muss.

Im realen Betrieb Myons soll er beliebig in seiner Umgebung navigieren können. In einem Szenario, in dem eine bereits gefahrene und gespeicherte Strecke erneut abgefahren wird, die Orte aber in einer unterschiedlichen Reihenfolge angesteuert werden, ist eine größere Sequenzlänge hinderlich. Dieses Szenario wird im Kapitel 8 analysiert. Wenn Myon von der bekannten Fahrbahn abkommt und einen anderen (bereits erkundeten) Weg fährt, betrachtet der Algorithmus für die nächsten vier Meter (laut Beispiel) noch Bilder aus dem vorherigen Fahrbahnabschnitt. Das führt dazu, dass eine neue Sequenz, die noch gar nicht bekannt ist, gesucht wird. Zu Beginn dieses Übergangs wird

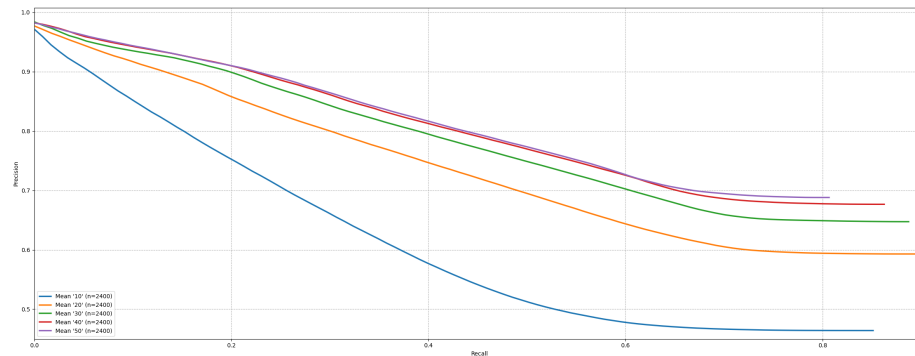


Abbildung 12: Precision-Recall Raten für verschiedene Sequenzlängen.

noch die Weiterverfolgung der alten Trajektorie geschätzt. Bei der Hälfte des Übergangs werden falsche Resultate mit einer geringen Confidence produziert und dementsprechend in den meisten Fällen nicht als Treffer gewertet. Nur im Fall, dass eine visuell ähnliche Fahrt bereits absolviert wurde, wird eine falsche Schätzung mit hoher Confidence abgegeben. Gegen Ende des Übergangs „fängt“ der Algorithmus sich und gibt eine gute Schätzung für die aktuelle Position ab.

Dieser Zustand der Unsicherheit soll nach Möglichkeit klein gehalten werden, gleichzeitig soll die Precision-Recall-Werte möglichst hoch sein. In Abb. 12 werden die Precision-Recall-Werte für die Sequenzlängen von 10 bis 50 in Zehnerschritten verglichen. Dabei wurden alle Parameterkombinationen in den Vergleich mit einbezogen. Ein klarer Trend ist zu erkennen - eine höhere Sequenzlänge führt zu einer höheren Precision. Diese Erkenntnis deckt sich mit den Versuchsergebnissen von [Milford, 2013] und [Sünderhauf et al., 2013], die mit DTW gearbeitet haben.

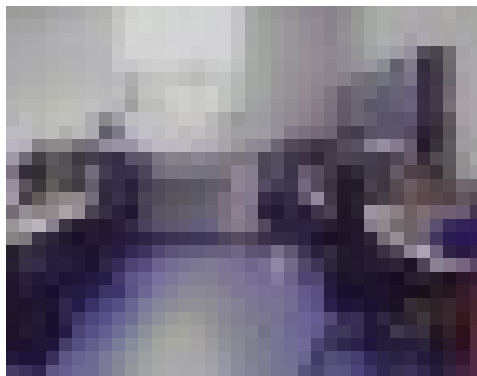
Die Sequenzlänge  $S$ , welche die Anzahl der zu einem „Pfad“ zusammengefassten Bilder definiert, ist der zentrale Parameter des Algorithmus. In der Abbildung 12 werden über alle Datensatz-Parameter-Kombinationen die Sequenzlängen von 10 bis 50 in Zehnerschritten verglichen. Alle erreichen einen Recall-Wert von über 80 %, was die Effizienz des Algorithmus nochmal hervorhebt. Allen Kombinationsmöglichkeiten erreichen immer noch so hohe Recall-Werte mit durchschnittlichen Precision-Werten die ebenso noch vergleichsweise hoch sind. Bei einer Sequenzlänge von 20 wird sogar einen Recall-Wert von 90 % erreicht, während die durchschnittliche Precision bei knapp 59,5 % liegt.

Auffällig ist die große Verschlechterung, sowohl bei Precision- als auch bei Recall-Werten, wenn die Sequenzlänge bei zehn liegt. Daraus lässt sich direkt schlussfolgern, dass kürzere Sequenzen schlechter sind. Oft besitzen sie eine geringere Eindeutigkeit und können leicht mit visuell ähnlichen Orten verwechselt werden, sogenanntes „perzeptuelles Aliasing“. Dies führt vermehrt zu Falsch-Positiv-Erkennungen, da der Sequenz der nötige Kontext zur robusten Unterscheidung fehlt. Die resultierenden Precision-Recall-Werte sind dementsprechend geringer.

Bei den Sequenzlängen 30 und 40 erkennt man eine leichte Reduktionen der maximalen Recall-Werte. Das geschieht auf Grund der höheren Verzögerung, die längere Sequenzen mit sich bringen. Erst nach erfolgreichem Vergleich aller Bilder in einer Sequenz wird eine Schätzung abgegeben, daher verursacht eine höhere Sequenzlänge eine größere Verzögerung. In Abbildung 13e ist diese Verzögerung gut ersichtlich. Wenn die verglichenen Datensätze mehr Bilder beinhalten, fällt dieses Phänomen weniger auf in der Statistik. Bei den aktuellen Datensatzlängen von ca. 440 Bildern bedeuten zehn mögliche Bilder weniger bereits eine ca. 2,3% Reduktion im maximal möglichen Recall.

Die Abbildung 13e bildet in rot den tatsächlich übereinstimmenden Pfad als Indizes der verglichenen Bibliotheken dar. Die gefundenen Treffer sind in blau dargestellt. Es ist ersichtlich, dass der blaue dem roten Pfad mit hoher Genauigkeit folgt. Dabei fällt sowohl das eben genannte Phänomen der Verzögerung als auch perzeptuelles Aliasing auf.

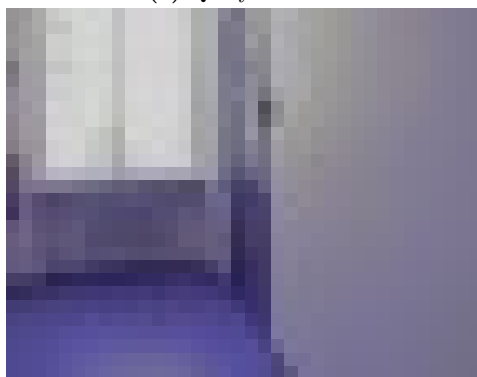
Die Query-Sequenzen im Bereich 360 bis 410 werden sowohl mit Bibliotheks-Sequenzen bei ca. 90 als auch bei 300 als übereinstimmend befunden, anstelle der um 390 liegenden richtigen Treffer. Alle drei Bilder haben ein dominantes, visuelles Merkmal gemeinsam, ein Fenster ist groß im Bild zu sehen. Bei den tatsächlich gesuchten Bibliotheks-Sequenzen steht jedoch eine Person vor dem Fenster, bei den anderen Fenstern nicht, wodurch die vom Histogrammausgleich und der Patch Normalization hervorgehobenen Kontraste einen höheren Fehler bei dem falschen Fenster produzieren. In den Abbildungen 13b, 13c und 13d werden die drei „ähnlichen“ Bilder abgebildet. Das Bild 13b zeigt das verdeckte Fenster und 13a ist des Query-Bild für das eine passende Sequenz gesucht wird. Da kontinuierlich auf die Fenster zugesteuert wird, sind nicht nur einzelne Bilder, sondern die gesamten Sequenzen vom Aliasing betroffen.



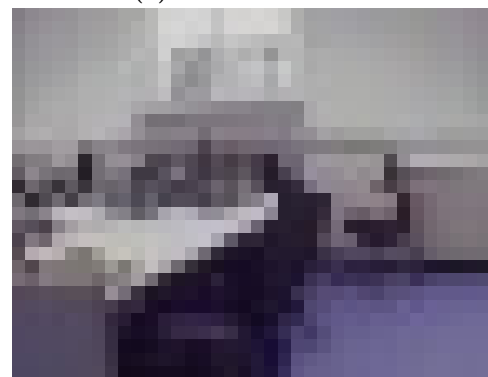
(a) Query-Bild 363



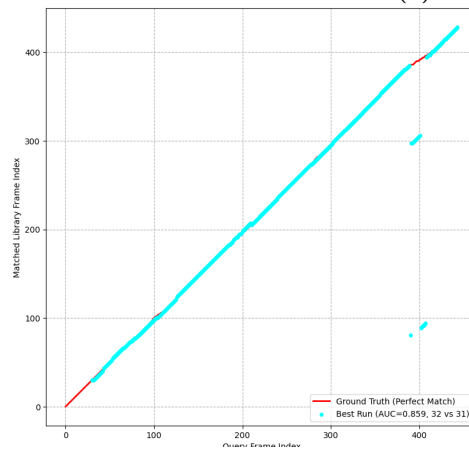
(b) Bibliothek-Bild 366



(c) Bibliothek-Bild 278



(d) Bibliothek-Bild 79

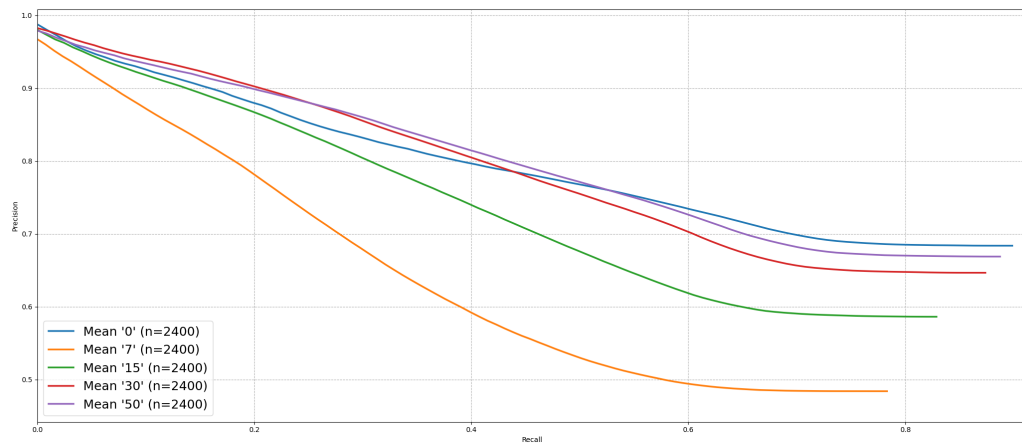


(e)

**Abbildung 13:** Ein Fall von perzeptuellem Aliasing, das gesuchte Bild (b) was zu (a) passt wird als (c) und (d) erkannt. In (e) sind die gefundenen Positives abgebildet, rot hinterlegt sind die True Positives.

## 6.4 Fenstergröße

Die Fenstergröße  $W$  ist der Parameter für die lokale Normalisierung des Differenzvektors (Gleichung 4.6). Sie definiert die Breite des Schiebefensters, über das der lokale Durchschnitt  $\mu_W$  und die Standardabweichung  $\sigma_W$  berechnet werden. Die Normalisierung dient dazu, die Signifikanz eines Treffers relativ zu seinen unmittelbaren Nachbarn im zeitlichen und räumlichen Kontext zu bewerten.



**Abbildung 14:** Precision-Recall Kurven für verschiedene Fenstergrößen.

Die Analyse, visualisiert in Abbildung 14, zeigt, dass eine geringere lokale Normalisierung (d.h. eine größere Fenstergröße  $W$ ) generell zu robusteren Ergebnissen führt. Dies liegt daran, dass eine stärkere lokale Normalisierung (kleineres Fenster) das Rauschen in der Differenzmatrix übermäßig verstärken und den für den Sequenzvergleich nötigen Kontext verzerren kann.

Das Ergebnis für das Auslassen des Normalisierungsschritts ( $W = 0$ ) sticht dabei als anomal negativ hervor. Ohne lokale Normalisierung verwendet der Algorithmus die unveränderten absoluten Differenzkosten ( $D$ ). Dies hat folgende Konsequenzen:

1. Die Kostenfunktion wird stark anfällig für globale Beleuchtungsartefakte, die nicht durch die Bildvorverarbeitung ausgeglichen wurden.
2. Bei der Berechnung der Sequenzkosten können False Positives mit sehr

geringen absoluten Kosten die Confidence verzerren.

Insgesamt steht dieser Fund im Kontrast zu den Ergebnissen von [Sünderhauf et al., 2013], wo die Fenstergröße als nicht zwingend notwendiger Schritt betrachtet wurde. Weil der Rechenaufwand für die lokale Normalisierung gering ist und somit das Kosten-Nutzen-Verhältnis bei einem Verzicht auf diesen Schritt ebenfalls gering wäre, ist die Beibehaltung der Normalisierung zur Erhöhung der Robustheit sinnvoll. Sie dient als wichtiger Filter zur Kompensation globaler Beleuchtungseffekte und sollte nicht ausgelassen werden.

## 6.5 Odometriefehler

Beim Betrachten der Odometriewerte fällt schnell auf, dass die Strecken einer Plausibilität folgen, wie Abbildung 15 zu sehen jedoch nicht fehlerfrei sind. Wie in Abschnitt 4.3 beschrieben, verwendet die Implementierung die explizite Euler-Integration (Gl. 4.8, 4.9). Eine gängige Alternative zur Reduzierung des numerischen Fehlers ist die *Midpoint-Methode*. Sie wird auch Runge-Kutta 2. Ordnung genannt. Hierbei wird der Winkel in der Mitte des Zeitschritts zur Positionsberechnung verwendet:

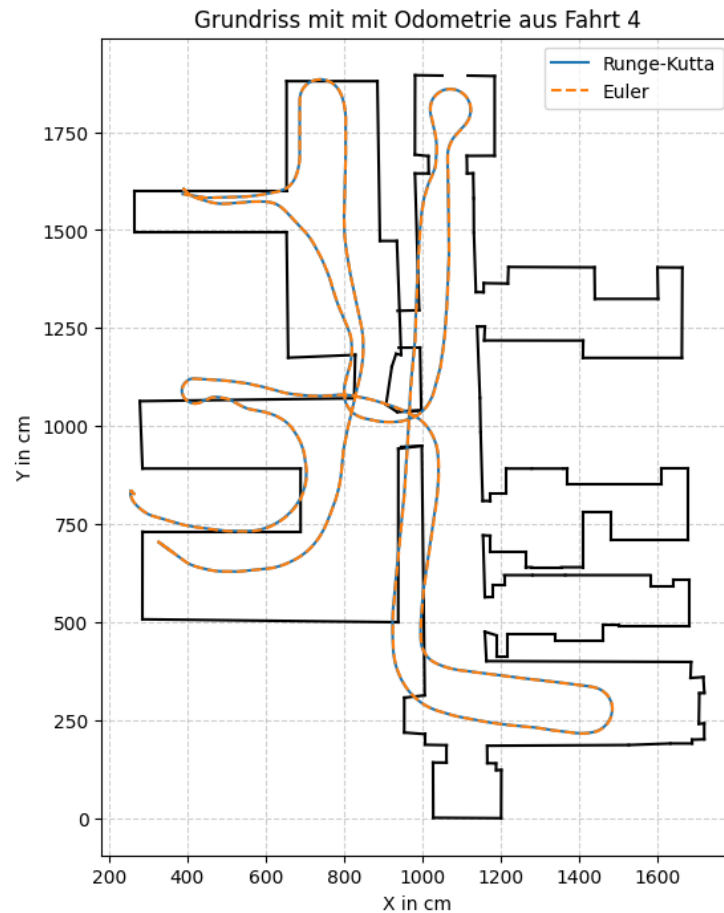
$$\Delta\varphi = \left( \frac{\Delta d_{rechts} - \Delta d_{links}}{L} \right) \quad (6.3)$$

$$x' = x + \Delta d_{Zentrum} \cdot \cos\left(\varphi + \frac{\Delta\varphi}{2}\right) \quad (6.4)$$

$$y' = y + \Delta d_{Zentrum} \cdot \sin\left(\varphi + \frac{\Delta\varphi}{2}\right) \quad (6.5)$$

$$\varphi' = \varphi + \Delta\varphi \quad (6.6)$$

Eine Analyse des Datensatzes 4 mit einer Gesamtstrecke von 92.1 m zeigt, dass der Unterschied zwischen der einfachen Euler-Methode und der verbesserten Midpoint-Methode auf dieser Distanz zu vernachlässigen ist (siehe Abbildung 15). Dargestellt wird die berechnete Trajektorie mit den ursprünglich im System hinterlegten Konstanten und dem von [Meißner, 2025] abgemessenen, befahrbaren Bereich des Labors. Der Unterschied in der finalen Endposition be-



**Abbildung 15:** Auswertung der Odometrie mit Euler-Verfahren im Vergleich zur Runge-Kutta-Methode.

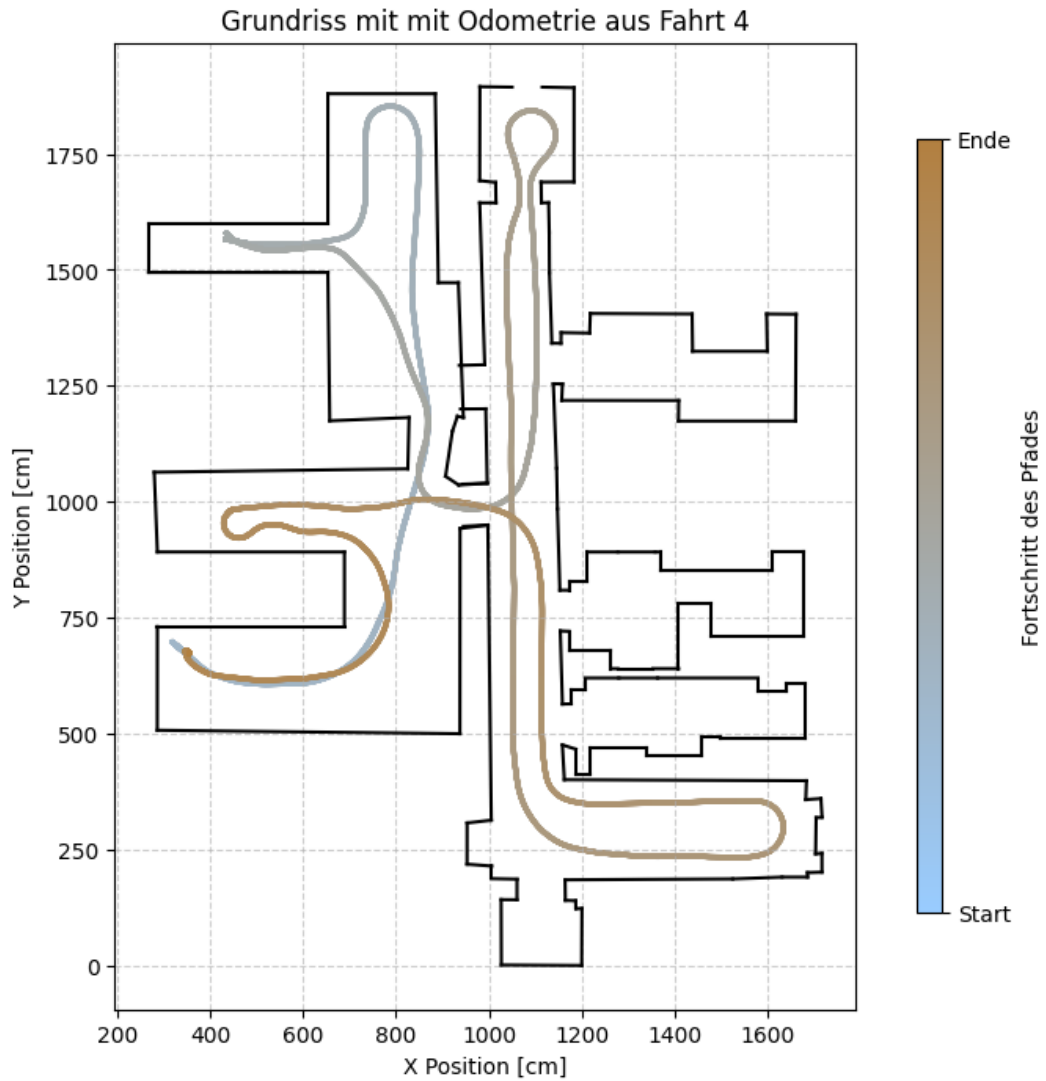
trägt 1.53 cm, und die durchschnittliche Abweichung pro Integrationsschritt liegt bei ca. 400 nm. Diese marginale Verbesserung der Trajektorie ist visuell nicht erkennbar.

Weitaus signifikanter ist der Drift, der sich auch visuell manifestiert: Die berechnete Trajektorie kollidiert mit bekannten Hindernissen (Wänden) und die Differenz zwischen Start- und Endposition beträgt bei 88,1 cm, obwohl der Roboter physisch ungefähr zum Ausgangspunkt zurückgekehrt ist. Analysen weiterer Datensätze bestätigen diesen Trend - die Trajektorien weisen konsistente Fehler auf, kollidieren an denselben Stellen mit Wänden und zeigen finale Positionsverschiebungen von bis zu zwei Metern in die oben-links Richtung.

Die Tatsache, dass die Midpoint-Methode keine signifikante Besserung bringt, während die Fehler über Datensätze hinweg konsistent auftreten, deutet stark auf eine systematische Problematik hin: Die im Modell verwendeten kinematischen Konstanten sind fehlerhaft oder nicht mehr aktuell.

Eine Neumessung der Radumfänge ergab 53,2 cm (links) und 53,227 cm (rechts). Diese Diskrepanz von 0,05% zwischen den Rädern ist zwar gering, führt jedoch bei der Integration zu einem persistenten Orientierungsfehler. Genauso relevant ist der Unterschied von etwa einem halben Zentimeter dem alten Radumfang gegenüber. Nach Korrektur dieser Konstanten im Odometrie-Modell verbessert sich die Positionsberechnung erheblich, siehe Abb. 16. Visualisiert ist die Trajektorie mit den korrigierten Konstanten und dem befahrbaren Bereich des Labors. Der Fehler zwischen Start- und Endposition bei Datensatz 4 reduziert sich von 88,1 cm auf 12,48 cm, dieser Restfehler liegt im Bereich des physisch erwartbaren Abstands zwischen Start- und Endpunkt der gefahrenen Route, dargestellt in der Abbildung 16. Die Konstanten sollten jedoch auch erneut mit möglichst präzisen Messgeräten nachgemessen werden, da, wie bereits dargestellt, kleinste Abweichungen eine große Auswirkung haben.

Obwohl die Kalibrierung der Konstanten der dominante Faktor ist, sollte dennoch die Midpoint-Methode beibehalten werden. Der numerische Fehler der Euler-Integration akkumuliert sich bei Rotationen quadratisch (im Gegensatz zum linearen Wachstum bei reiner Geradeausfahrt). Die Midpoint-Methode reduziert diesen Fehler bei nahezu identischem Rechenaufwand. Es wird daher empfohlen, stets beide Fehlerquellen zu minimieren - sowohl die systematische Kalibrierung durch präzise Messung als auch die numerische Integration durch die Midpoint-Methode.



**Abbildung 16:** Auswertung der Odometrie mit verbesserten Konstanten und dementsprechend auch verbessertem Ergebnis zusammen mit dem befahrbaren Bereich des Labors.

## 7 Confidence-Threshold

Für jedes Query-Bild muss eine Entscheidung getroffen werden:

- a) True Positive (Loop Closure): Liegt eine korrekte Übereinstimmung mit einem früheren Ort vor?
- b) False Positive (Perzeptuelles Aliasing): Handelt es sich um einen Trugschluss, bei dem ein Ort fälschlicherweise einem visuell ähnlichen Ort zugeordnet wird?
- c) True Negative (Novelty): Handelt es sich um eine neuartige Szene, die korrekt als „unbekannt“ eingestuft wird?

Kritisch ist der Fall, wenn ein False Positive (Fall b) fälschlicherweise als True Positive (Fall a) klassifiziert wird. Dies führt zum Eintrag eines fehlerhaften Constraints in den Pose-Graphen (siehe Kapitel 5.1), was die Karte korrumpieren kann. Umgekehrt ist das Verwerfen eines echten Treffers (False Negative) unkritisch, da lediglich eine Gelegenheit zur Drift-Korrektur verpasst wird. Der Confidence-Score liefert, besonders für die Unterscheidung zwischen dem ersten und zweiten Fall wichtige Informationen.

Der Confidence-Score liegt, wie in Gleichung 4.16 definiert, im Bereich  $[0, 1]$ , wobei Werte gegen null einer maximalen - und Werte gegen eins einer minimalen Confidence entsprechen. Gleichermaßen bedeutet das eine maximale Unsicherheit in die gefundenen Treffer deutet und auf perzeptuelles Aliasing oder eine unbekannt Pose hin.

In Abbildung 17 sind direkt mehrere relevante Informationen zu erkennen. In der Abbildung wird für einen graduell steigenden Confidence-Schwellwert, in Schritten von 0,1 (was sinkende Confidence bedeutet) die durchschnittliche

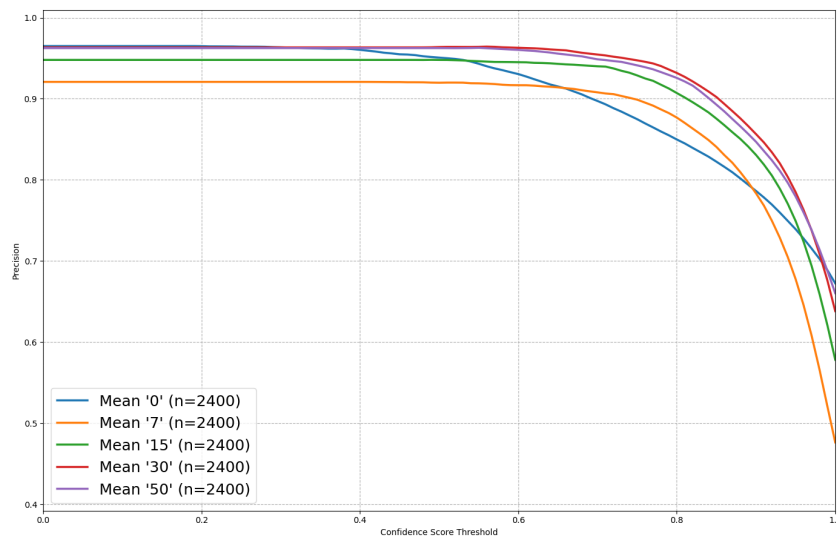


Abbildung 17: Confidence-Threshold.

Precision für die analysierten Fenstergrößen mit allen restlichen Parameterkombinationen dargestellt. Der konstante Start der Graphen bis ca. 0,5 (abgesehen von Fenstergröße „0“) bedeutet, dass für diese Confidence-Scores keine Treffer gefunden wurden. Der Wert, den die Graphen während dieser Phase einnehmen, ist die Precision, welche beim ersten Überschreiten des minimalen Thresholds, der überhaupt Treffer findet, vorliegt. Zu erkennen ist ein stetiger Trend: Sinkende Confidence resultiert in sinkender Precision, wobei kleinere Normalisierungsfenster schlechtere Precision-Werte erreichen.

Ausgenommen davon ist das Ergebnis für keinen Normalisierungsschritt, wie in Kapitel 6.4 bereits festgestellt wurde. Aufgrund der fehlenden Umrechnung der Ähnlichkeitsvektoren in ein Vielfaches der Standardabweichung verhalten sich Precision-Recall und Confidence hier einzigartig. Die Precision beginnt bereits ab einem Threshold von ca. 0,39 zu sinken, jedoch bedeutend langsamer als die restlichen Precision-Werte, sodass bei einem Threshold von 1 die Precision sogar am höchsten ist - was die Ergebnisse aus der Abbildung 14 widerspiegelt.

Jedoch bedeutet das nicht, dass diese Konfiguration die Beste ist. Um einen Threshold zu finden, der mit möglichst hoher Genauigkeit zwischen False Positives und True Positives unterscheiden kann, ist eine Kurve mit einem

möglichst charakteristischem „Knick“ hilfreich. Im direkten Vergleich schneiden auch alle Fenstergrößen besser ab, bis ein Threshold von ca. 0,9 für kleinere und ca. 0,97 für größere erreicht ist. Dass ein Threshold kleiner als 0,97 sein sollte, wird in der Abbildung 18 ersichtlich, dementsprechend ist der Normalisierungsschritt beizubehalten empfehlenswert.

Die Abbildung 18 ist eine Analyse der Sequenzkosten, welche durch die Gleichung 4.12 berechnet werden, und unterteilt in zwei Darstellungsformen. Die Sequenzkosten-Matrix hat für jeden Query-Fenster-Index eine Spalte mit den summierten Kosten der zugehörigen Sequenz in logarithmischer Darstellungsform, da besonders die niedrigen Kosten entscheidend sind. Die Spalte für  $i = 0$  ist die Summe der Kosten für die erste Sequenz, in diesem Fall 30. Die Bibliotheks-Indizes sind auch begrenzt auf den Bereich vollständiger Sequenzen, also gilt in diesem Fall auch dass 0 mit 30 korrespondiert. Die Matrix erhält ihr verschwommenes Aussehen durch die diagonale Überlappung innerhalb der  $\hat{D}$  Matrix, zwei benachbarte Spalten sind 29 Werte, die um eine Reihe verschoben sind.

In blau eingezeichnet ist der gefundene Pfad, also die für jeden Query-Index die geschätzte Position. Die dunkle Winkelhalbierende entspricht den richtigen Übereinstimmungen, auch wenn diese nicht immer gefunden wurde. Die verwendete Auflösung ist „ausfüllend“, die Bildvorverarbeitung Histogrammgleich, die Größe des Normalisierungsfensters 50 und die verglichenen Datensätze sind 2 und 3.

In der Unteren Grafik sind die absoluten und die relativen Kosten (die Confidence-Scores) für jeden Query-Fenster-Index abgebildet und teilen sich die gleiche X-Achse um einen leichten Abgleich mit der oberen Grafik zu ermöglichen. Eine klare Korrelation zwischen den beiden ist zu erkennen, hohe und niedrige absolute Kosten führen tendenziell auch zu hohen bzw. niedrigen relativen Kosten.

Hohe absolute Kosten sind ein Ergebnis von Bildern die innerhalb ihrer Normalisierungsfenster nicht als besonders guter Treffer herausstechen. In diesem Fall bedeuten hohe relative Kosten, dass schlechte Resultate im Vergleich zueinander nicht eindeutig unterschieden werden können und sollten in jedem Fall herausgefiltert werden, da die Unsicherheit in diese Ergebnisse zu hoch ist.

Das würde für die verglichenen Datensätze bedeuten, dass die geschätzten Posen für die Bereiche der Indizes 190-205, 265-300 und 355-365 nicht als Treffer akzeptiert werden. Für die letzteren beiden Bereiche werden so False Positives herausgefiltert. Der erste Bereich entspricht True Positives und würden ebenso herausgefiltert werden, was einer Strecke von etwa 3 Metern entspricht. Diese kann problemlos von der Odometrie übernommen werden.

Bei niedrigen absoluten Kosten ist es eine Voraussetzung, dass gute Treffer innerhalb ihres Fensters herausstechen jedoch garantiert das keine niedrigen relativen Kosten, da weitere Sequenzen mit hoher Übereinstimmung gefunden werden können. Dafür eignet sich ein Vergleich der Bereiche zwischen 50-60 und 125-135 - während die absoluten Kosten bei ca. 95 liegen, unterscheiden die Confidence-Scores sich um etwa 10 % (ca. 0,67 vs. 0,74). In diesem Fall eignet sich ein einfacher Threshold von z.B. 0,85. Hier kann ein strengerer, oder ein adaptiver Threshold als größere Absicherung gegenüber False Positives dienen.

## Sequenzkosten-Analyse

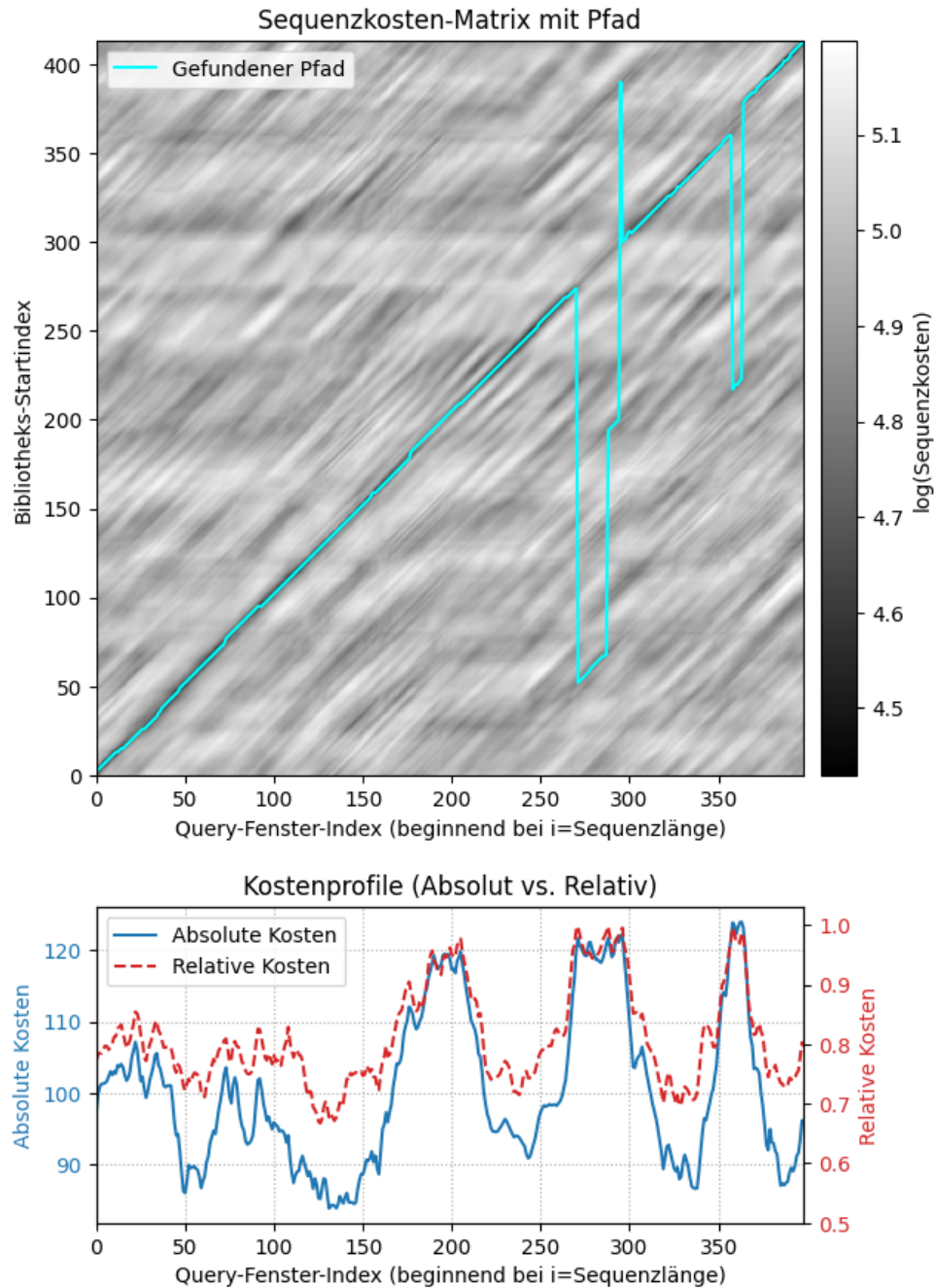


Abbildung 18: Kostenanalyse.

In Abbildung 17 geschieht der größte Verlust an Precision im Bereich von 0,8 bis 1,0. Dementsprechend sollte ein Wert in dieser Größenordnung als Schwellenwert gewählt werden.

Die Unterscheidung zwischen einem False Positive und einem tatsächlich unbekanntem Ort ist rein visuell oft mehrdeutig, insbesondere bei schlechten Confidence-Scores ( $C_i > Threshold$ ). Zur Auflösung dieser Ambiguität wird die Odometrie im Kontext des Pose-Graphen herangezogen. Korreliert eine hohe visuelle Unsicherheit mit einer Odometrie-Position, die weit entfernt von bekannten Knoten im Graphen liegt, so handelt es sich mit hoher Wahrscheinlichkeit um Novelty (Fall c). In diesem Szenario wird ein neuer Knoten zur Karte hinzugefügt. Signalisiert die Odometrie hingegen die Nähe zu einem bekannten Knoten, während der Confidence-Score dennoch schlecht bleibt, deutet dies auf perzeptuelles Aliasing oder temporäre Verdeckungen hin (Fall b). Der Match wird in diesem Fall verworfen, um die Konsistenz der Karte nicht zu gefährden.

## 8 Anwendung

Dieses Kapitel evaluiert den optimierten Algorithmus unter realen Bedingungen auf der Zielhardware (Myon). Zusätzlich wird ein erster Schritt in Richtung *Online-SLAM* unternommen, indem die Robustheit des Systems gegenüber Diskontinuitäten in der Trajektorie – also dem Anfahren bekannter Orte in veränderter Reihenfolge – untersucht wird.

Abbildung 19 visualisiert die Trajektorie der Bibliothek sowie deren Unterteilung in Segmente. Basierend auf der Analyse in Kapitel 6 wurden folgende Parameter gewählt: Auflösung „ausfüllend“ (für zukünftige Vergleiche, niedrigere Auflösungen sind ausreichend), Histogrammausgleich und ein räumlicher Bildabstand von 20 cm. Die Versuche wurden bei konstanter Deckenbeleuchtung durchgeführt.

### 8.1 Evaluation auf der Hardware

Es wurden drei Validierungsfahrten mit den ermittelten optimalen Parametern (Sequenzlänge  $S = 30$ , Fenstergröße  $W = 50$ ) durchgeführt. Abbildung 20 zeigt exemplarisch die Differenzmatrix eines Durchlaufs. Grüne Bereiche kennzeichnen niedrige Differenzwerte (hohe Übereinstimmung), während rote Bereiche hohe Differenzen darstellen. Die gepunktete Linie visualisiert den vom Algorithmus bestimmten Treffer.

Zu Beginn, während noch keine vollständige Sequenz vorhanden ist, sind die Treffer noch fehlerhaft verstreut. Sobald die volle Sequenzlänge fast erreicht ist, „fängt sich“ der Algorithmus und die folgenden Ergebnisse sind akkurat (bis auf einen kleinen Abschnitt kurz vor der Hälfte der Strecke). Über alle drei Datensätze hinweg wurde eine durchschnittliche Precision von 91 % erreicht, bei keiner Verwendung eines Thresholds und unter Einbeziehung der

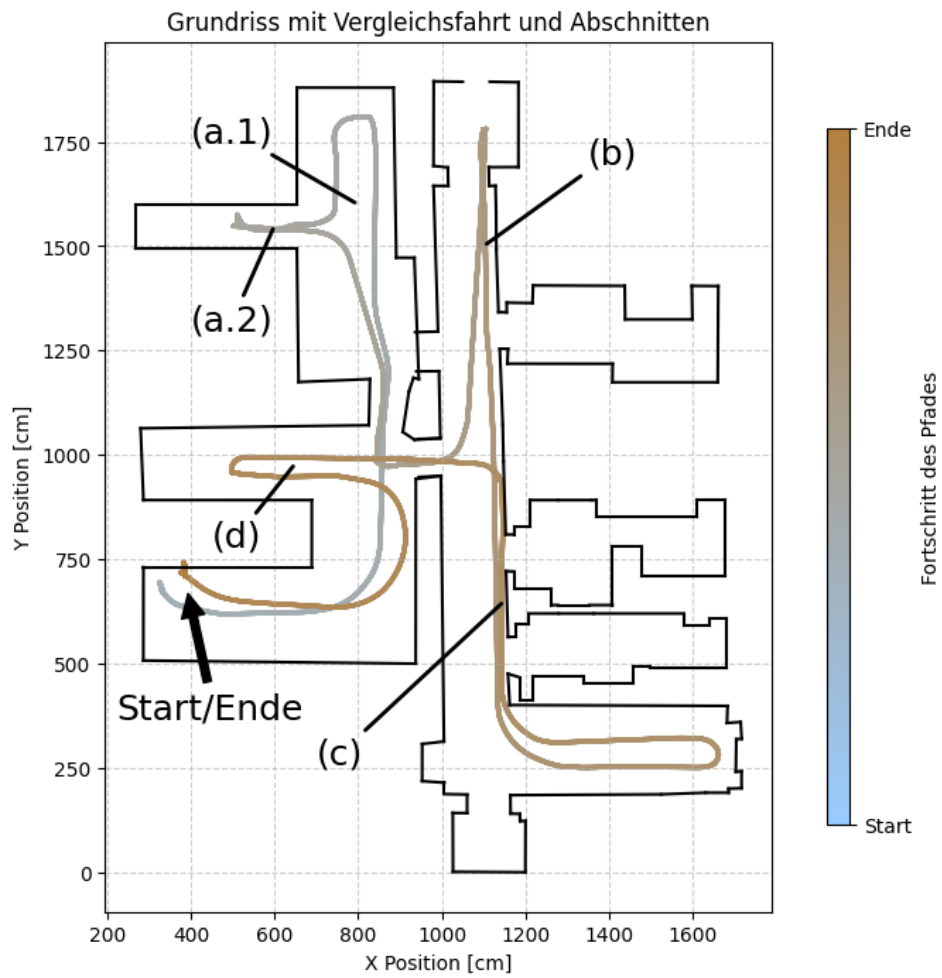
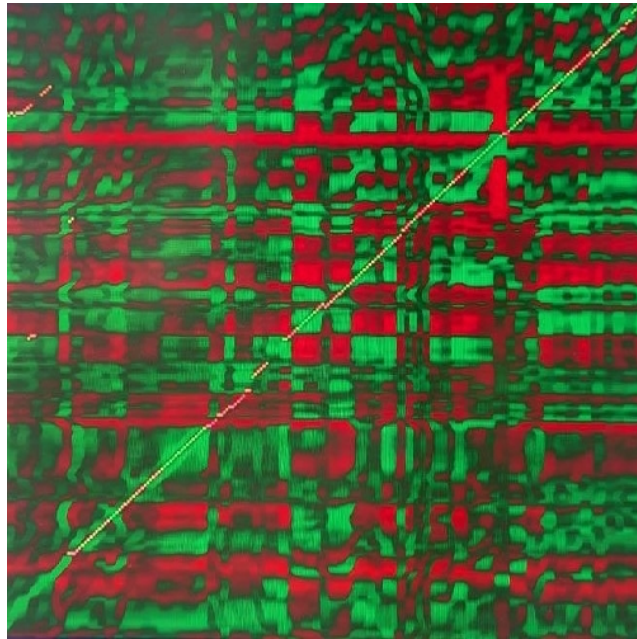


Abbildung 19: Sektionen der gefahrenen Sequenzen.



**Abbildung 20:** Durchlauf auf dem Roboter.

unvollständigen Sequenzen zu Beginn.

Dieses Resultat übertrifft sogar den Erwartungswert. Die hohe Präzision ist primär auf die kontrollierten Lichtverhältnisse (eingeschaltete Deckenbeleuchtung) zurückzuführen. Der Algorithmus funktioniert auf Myon also nicht besser als in der Simulation, sondern reagiert auf Änderungen der Lichtverhältnisse. Wie von [Milford, 2012] dargelegt, zielt SeqSLAM zwar auf Robustheit gegenüber visuellen Änderungen ab, bietet jedoch keine vollständige Invarianz gegenüber drastischen Beleuchtungswechseln.

## 8.2 Analyse von Trajektorien-Sprüngen

Anders als auf schienengebundenen Systemen, wird Myon im regulären Betrieb nicht eine festgelegte Route nachverfolgen. Von besonderem Interesse ist daher, wie lange der Algorithmus benötigt, um sich „zu fangen“ bei bewussten Sprüngen innerhalb der bekannten Karte. Also wenn Myon eine Strecke fährt die er vollständig kennt, welche aber in zwei unterschiedlichen Segmenten abgespeichert ist.

Um diesen Fall zu testen, wurde die bekannte Route der Bibliothek (zu finden in Abbildung 19) mit Querys in der Reihenfolge d, b, a.1 ohne die Segmente c und a.2 verglichen (Start- und Endposition sind identisch). Dabei sind die Übergänge zwischen den Segmenten unbekannte Posen und fallen daher aus der Bewertung heraus. Getestet wurde mit den gleichen Parametern wie zuvor, nur die Sequenzlänge 20 wurde ebenso getestet, um die Hypothese zu überprüfen, ob bei Übergängen kürzere Sequenzen bessere Resultate liefern. Die Resultate sind händisch annotiert und dienen als qualitative Analyse bei Sequenzbrüchen.

Ein Übergang ist in drei Phasen aufgeteilt:

1. Persistenz-Phase: Die niedrigen Kosten der vergangenen Sequenz werden in die nächsten Schritte weitergetragen – der Algorithmus findet Treffer, als würde die Bibliotheksstrecke weitergefahren werden.
2. Ambiguitäts-Phase: Die neuesten Query-Bilder sind Teil des neuen Segments, der mittlere Part (der Übergang) der Sequenz besteht aus unbekanntem Bildern und die ältesten Bilder innerhalb der Sequenz sind aus dem alten Segment. Hier werden mehr False Positives produziert, je länger die Strecke des Übergangs ist.
3. Relokalisierungs-Phase: Genug Bilder aus dem neuen Abschnitt sind in der Sequenz enthalten, um den niedrigsten Kostenwert zu erreichen und wieder True Positives zu produzieren.

Die Dauer der Phasen unterscheidet sich zwischen den Sequenzlängen 20 und 30 nur im Verhältnis der ersten beiden: Bei kürzeren Sequenzen fallen alte Bilder früher aus der Kostenrechnung und landen daher schneller in der Ambiguität. Wenn die Übergangsbilder (Bilder aus unbekanntem Posen) aus den Übergängen herausgerechnet werden, so benötigt der Algorithmus ca. 15 Bilder (3m) um sich zu relokalisieren.

Hier greift der Confidence-Threshold: Mit jedem Schritt ins Unbekannte wird der Confidence-Score schlechter. In Kombination mit einer Karte wird durch die Odometrie erkannt, dass ein unbekannter Pfad gefahren wird, und neue Knoten (Bilder mit Posen) werden angelegt.

# 9 Fazit

## 9.1 Zusammenfassung

Ziel dieser Arbeit war die Entwicklung eines visuellen Lokalisierungsverfahrens für den humanoiden Roboter Myon, das die spezifischen Herausforderungen repetitiver Innenräume bewältigt und gleichzeitig auf der limitierten eingebetteten Hardware in Echtzeit operiert. Die eingangs in Kapitel 1 definierte Problemstellung – die Überwindung von perzeptuellem Aliasing unter strengen Ressourcenbeschränkungen – konnte durch eine modifizierte Implementierung des SeqSLAM-Algorithmus erfolgreich gelöst werden.

Der zentrale methodische Beitrag dieser Arbeit ist der Wechsel von der zeitlichen Abtastung zu Spatial Sampling. Durch die direkte Kopplung der Bildaufnahme an die Odometrie wurde die Rechenlast der Sequenzsuche linearisiert. Dies eliminierte die Notwendigkeit für komplexe Synchronisationsverfahren wie Dynamic Time Warping und reduzierte die Laufzeit signifikant, was die geforderte Echtzeitfähigkeit auf der Zielhardware erfüllt.

Die quantitative Evaluation lieferte dabei folgende Kernbefunde:

- **Robustheit:** Das System erreichte in realen Testszenarien unter variierenden Lichtbedingungen (Tageslicht vs. Kunstlicht) eine durchschnittliche Präzision von 91 %. Dies bestätigt die Eignung des sequenzbasierten Ansatzes für die strukturarme Laborumgebung.
- **Parameter-Dominanz:** Es konnte nachgewiesen werden, dass das Sichtfeld (FOV) einen signifikant größeren Einfluss auf die Erkennungsrate hat als die reine Bildauflösung. Dies validiert die "How low can you go?-Hypothese: Für die Zuordnung ähnlicher Orte ist der räumliche Kontext wichtiger als die lokale Detailtiefe.

- Vorverarbeitung: Entgegen der Tendenz in der Literatur (Außenbereiche) erwies sich der Histogrammausgleich (HA) in dieser Innenraum-Anwendung als robuster gegenüber der Patch-Normalisierung (PN).
- Systemdynamik: Die Untersuchung von Trajektorien-Sprüngen zeigte, dass das System in der Lage ist, sich nach einem "Kidnapped Robot" Szenario innerhalb einer Latenz von ca. 3 Metern (entsprechend 15 Bildern) stabil zu relokalisieren.

## 9.2 Kritische Einordnung: Von der Lokalisierung zu SLAM

Der Titel dieser Arbeit suggeriert die Entwicklung eines vollständigen SLAM-Algorithmus. Bei der Betrachtung realisiert das vorgestellte System primär den Lokalisierungspart sowie die robuste Detektion von Loop-Closures (das sogenannte *Frontend*). Die kontinuierliche Korrektur der Kartenstruktur im Hintergrund (das *Backend*) wurde konzeptionell vorbereitet, aber noch nicht integriert.

Ein vollständiges SLAM-System zeichnet sich dadurch aus, dass erkanntes Wiederkehren an einen Ort dazu genutzt wird, den akkumulierten Fehler der Vergangenheit global zu korrigieren. Das hier entwickelte System identifiziert diese Momente durch den Konfidenzschwellenwert zuverlässig und verhindert durch die Unterdrückung von False Positives eine Korruption der Karte. Es fehlt jedoch der letzte Schritt der Pose-Graph-Optimierung, der die Odometrie-Trajektorie rückwirkend anpasst. Aktuell wächst die "Karte" linear mit der Odometrie, was bedeutet, dass der Drift zwar erkannt, aber in der metrischen Kartenstruktur noch nicht eliminiert wird.

Dennoch ist diese Fokussierung gerechtfertigt: In der spezifischen Problemstellung – repetitive Innenräume und limitierte Hardware – stellt das Frontend die kritische Komponente dar. Ein Optimierungs-Backend scheitert fatal, wenn das Frontend aufgrund von perzeptuellem Aliasing falsche Constraints liefert. Die Arbeit löst somit das eigentliche Problem für diese Umgebung: die Bereitstellung robuster, verifizierter Loop-Closure-Kandidaten.

## 9.3 Ausblick

Aufbauend auf den validierten Ergebnissen ergeben sich klare Schritte, um den Algorithmus zu einer vollständigen *online SLAM*-Lösung auszubauen:

1. Integration eines Graph-Optimierers: Die Implementierung des in Kapitel 5 konzipierten Pose-Graphs.
2. Generierung der Belegungskarte: Sobald die Posen durch den Graphen optimiert sind, können die bereits vorhandenen Infrarot-Abstandssensordaten genutzt werden, um eine metrische Gridmap für die Pfadplanung zu erstellen.
3. Aktive Exploration: Das System könnte von einer passiven Lokalisierung zu einer aktiven Steuerung erweitert werden. Erkennt der Algorithmus eine hohe visuelle Unsicherheit in Kombination mit einer unbekanntem Odometrie-Pose, könnte Myon gezielt Explorationsstrategien anstoßen, um leere Flecken auf der Karte autonom zu füllen.

# Abbildungsverzeichnis

1	Ergebnisse der Vorarbeiten . . . . .	6
2	Myon mit Unterbau . . . . .	7
3	Schematischer Grundriss der Versuchsumgebung . . . . .	9
4	Bild 143 aus Datensatz 3, links das Original, rechts nach der Patch Normalization . . . . .	12
5	Bild nach dem Histogrammausgleich . . . . .	13
6	Bild nach der inversen Gammakorrektur . . . . .	14
7	Schematische Darstellung des Prozesses von Einzelbildverglei- chen zur Differenzmatrix . . . . .	15
8	Visualisierung der Odometrie für einen sehr großen Zeitschritt $\Delta t$ . . . . .	17
9	Das beste Precision-Recall Ergebnis . . . . .	26
10	Precision-Recall für die Bildauflösungen . . . . .	28
11	Precision-Recall für die Bildvorverarbeitungsmethoden . . . . .	30
12	Precision-Recall für die Sequenzlängen . . . . .	32
13	Beispielbilder für Visuelles Aliasing . . . . .	34
14	Precision-Recall Kurven für Fenstergrößen . . . . .	35
15	Auswertung der Odometrie mit Euler-Verfahren im Vergleich zur Runge-Kutta-Methode. . . . .	37
16	Route mit verbesserter Odometrie . . . . .	39
17	Confidence-Threshold. . . . .	41
18	Kostenanalyse. . . . .	44
19	Sektionen der gefahrenen Sequenzen. . . . .	47

20 Durchlauf auf dem Roboter. . . . . 48

# Tabellenverzeichnis

1	Übersicht der verwendeten Bildauflösungen und ihrer Bezeichnungen . . . . .	8
2	Übersicht der ausgewerteten Datensätze . . . . .	10
3	Übersicht der getesteten Parameter . . . . .	27

# Literaturverzeichnis

- [Aschenbrenner, 2015] Aschenbrenner, B. (2015). Implementation and evaluation of image sequence based place recognition utilizing a humanoid robot. M.sc. thesis, Freie Universität Berlin.
- [Cadena et al., 2017] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J. (2017). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332.
- [Elfes, 2002] Elfes, A. (2002). Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57.
- [Grisetti et al., 2011] Grisetti, G., Kümmerle, R., Stachniss, C., and Burgard, W. (2011). A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43.
- [Meißner, 2025] Meißner, W. (2025). Praxisphasenbericht: Willi meißner. Technical report.
- [Milford, 2012] Milford, M. (2012). Visual route recognition with a handful of bits. In *Robotics: Science and Systems*. Sydney, Australia.
- [Milford, 2013] Milford, M. (2013). Vision-based place recognition: how low can you go? *The International Journal of Robotics Research*, 32(7):766–789.
- [Milford et al., 2004] Milford, M., Wyeth, G., and Prasser, D. (2004). Ratslam: a hippocampal model for simultaneous localization and mapping. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 1, pages 403–408 Vol.1.

- [Mur-Artal et al., 2015] Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). Orb-slam: A versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163.
- [Ruiz, 2025] Ruiz, M. (2025). Praxisbericht über sequence slam. Technical report.
- [Ruiz Holtgreffe, 2025] Ruiz Holtgreffe, M. (2025). Seqslam evaluation. Accessed: 2025-10-20.
- [Shan et al., 2003] Shan, S., Gao, W., Cao, B., and Zhao, D. (2003). Illumination normalization for robust face recognition against varying lighting conditions. In *2003 IEEE International SOI Conference. Proceedings (Cat. No.03CH37443)*, pages 157–164.
- [Sünderhauf et al., 2013] Sünderhauf, N., Neubert, P., and Protzel, P. (2013). Are we there yet? challenging seqslam on a 3000 km journey across all four seasons. In *Proc. of workshop on long-term autonomy, IEEE international conference on robotics and automation (ICRA)*, page 2013. Citeseer.
- [VIDEOLOGY, 2012] VIDEOLOGY (2012). Application note 21k15xdig. Technical Report 44068984, Videology Incorporated. Datenblatt.