Karlsruher Institut für Technologie (KIT)
Fakultät für Informatik
Institut für Anthropomatik
Lehrstuhl für Intelligente Sensor-Aktor-Systeme (ISAS)
Prof. Dr.-Ing. Uwe D. Hanebeck

Praktikum

Forschungsprojekt Anthropomatik praktisch erfahren

14. Februar 2018

He Jingyu, Martin Miller

Roboterlokalisierung in 2D mittels SE(2) Filterung

Roboterlokalisierung in 2D mittels SE(2) Filterung

– Praktikum: Forschungsprojekt Anthropomatik praktisch erfahren–

He Jingyu, Martin Miller

14. Februar 2018

Zusammenfassung

In der Robotik spielt die Lokalisierung des Roboters eine große Rolle. Dabei sollen die Position sowie die Orientierung des Roboters erfasst werden. In einem zweidimensionalen Szenario kann man die Pose als Element der speziellen Euklidischen Gruppe SE(2) auffassen, welche alle Kombinationen von Rotationen und Translationen in 2D enthält. Um die Pose des Roboters basierend auf verrauschten Sensordaten zu schätzen, kommen häufig Kalman-Filter und ähnliche Verfahren zum Einsatz. Am ISAS wurde eine Methode entwickelt, welche das besondere Verhalten der SE(2), insbesondere die Periodizität der Orientierung, besser behandelt als traditionelle Verfahren. In diesem Praktikumsprojekt soll das am ISAS entwickelte Verfahren zur SE(2) Schätzung in einem praktischen Szenario implementiert und evaluiert werden. Dazu kommt der am ISAS entwickelte Crawler zum Einsatz.

Inhaltsverzeichnis

1 Einführung

Die Fähigkeit eines Roboters, sich selbst zu lokalisieren, ist das Grundkonzept für autonome Rbotersysteme. Die Aufgabe des Roboters ist es, durch Auswertung der Sensordaten herauszufinden, wo er ist.
Um das zu schaffen, wird eine Filterung im Robotersystem eingesetzt. Schätzung einer Starrkörperbewegung, d.h.Gleichzeitige Schätzung von Transtorlation und Rotation, ist in vielen Situationen ein großes Problem. Dazu muss die Anwendungen robotische Wahrnehmung und Verarbeitung von Sensordaten gleichzeitig einsetzen. Dieses Problem ist aus zwei Gründen besonders schwierig.
Erstens hat es eine nichtlineare Struktur, weil die Werte auf einer instationären nichtlinearen Funktionsbereich definiert sind. Das Problem kann mit Hilfe der Mannigfaltigkeiten der Starrkörperbewegungen in der Ebene SE(2) gelöst werden. Zweitens, es gibt keine kanonische Möglichkeit, Abhängigkeiten zwischen Position und Orientierung zu beschreiben. Der SE(2)-Filter ist eine von ISAS selbst entwickeltes Verfahren, das zur Roboterlokalisierug verwendet werden kann. In dieser Arbeit wird eine neuartige Wahrscheinlichkeitsverteilung, die Bingham-Verteilung, eingesetzt, welche in der Lage ist, die zugrunde liegende Struktur von der planarer starrer Transformationen darzustellen. Das ist durch die Verwendung von duale Quaternionen (beschränkt auf planare Transformationen) realisiert. Daher ist es nicht notwendig, eine gesamte Matrix zur Beschreibung der tatsächlichen Transformation zu erstellen. Außerdem kann der Bingham-Verteilung zur Darstellung unsicherer Orientierungen durch Quaternionen repräsentiert werden, was klassischen Messupdate- und Fusionsszenarien wie Beyas-Filterung möglich macht.
Zur Evaluierung des SE(2)-Filters wird Partikel-Filter in diese Arbeit implementiert. Die Partikel sind Proben, die die aktuelle Positionen und Orientierungen des Roboters schätzen. Auf diesen Teil werden wir noch später eingehen.
Es gibt zwei wesentliche Teile, die implementiert werden müssen, nämlich das System und das Messmodell. Abschließend wird diese Arbeit mit einem mobilen Roboter und einer Kamera ausgewertet. Die Kamera wird verwendet, um die tatsächliche Position des Roboters zu verfolgen, die dann mit der geschätzten Ortskurve verglichen wird.
Zur Durchführung haben wir im Rahmen des Praktikum Forschungsprojekts 'Anthropomatik praktisch erfahren' einen etwa zehn Zentimeter hohen Laufroboter zur Verfügung gestellt bekommen. Dieser besitzt drei Beine, mithilfe derer er sich fortbewegen kann und vier Ultraschallsensoren, die für die Erkundung der Umwelt verwendet werden können. Weitere Details bzgl. des Aufbaus des Crawlers können Abschnitt 6 entnommen werden.

2 Aufgabenstellung

- Einarbeitung in die nötigen Grundlagen (Bayes-Filterung, duale Quaternionen, Bingham Verteilung).
- Einarbeitung in das am ISAS entwickelte Verfahren.
- Entwurf geeigneter System- und Messmodelle unter Verwendung von Vorarbeiten der letzten Semester.
- Implementierung des Verfahrens und eines Vergleichsverfahrens (z. B. UKF oder Partikelfilter)
- Evaluation des Verfahrens in Simulationen und mit realen Daten

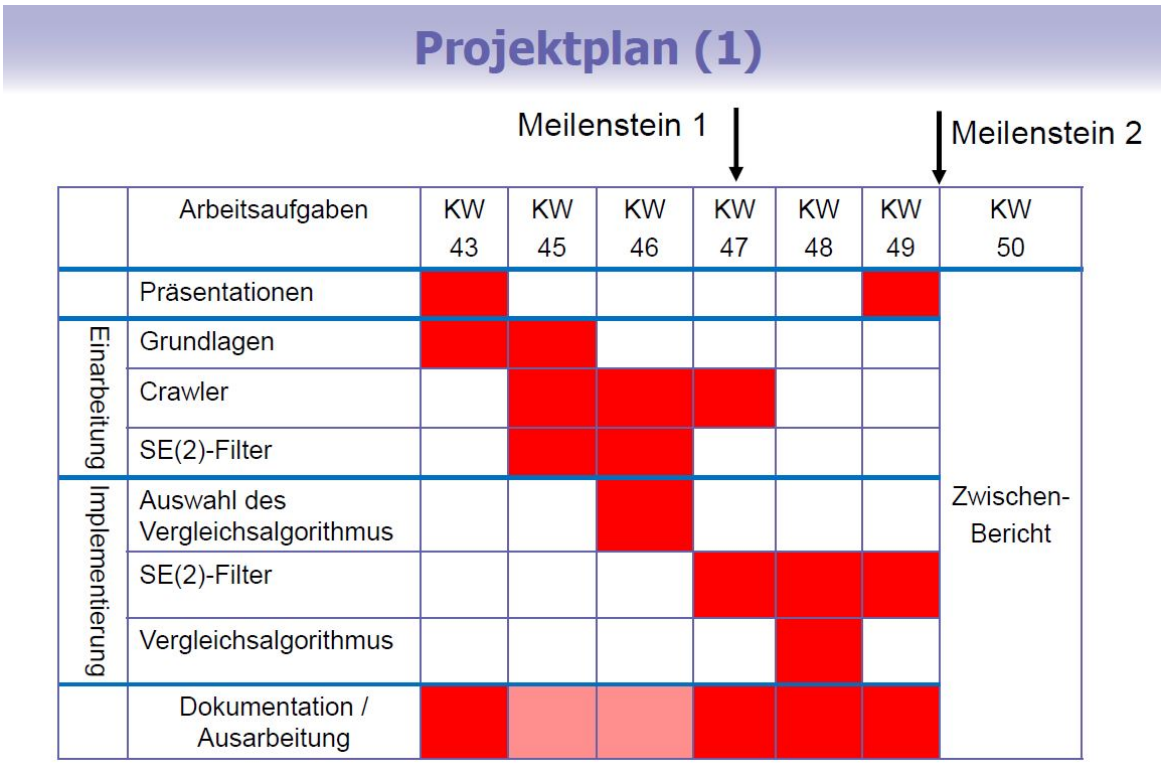


Abbildung 1: Zeitplaung 2017

3 Projektplan

Zu Beginn des Projektes wurde ein Zeitplan (Abbildung 1 und 2) zwecks besserer Koordinierung und Zeitmanagement erstellt. Bei der zeitlichen und organisiatorischen Einteilung sind somit zwei logische Blöcke entstanden:

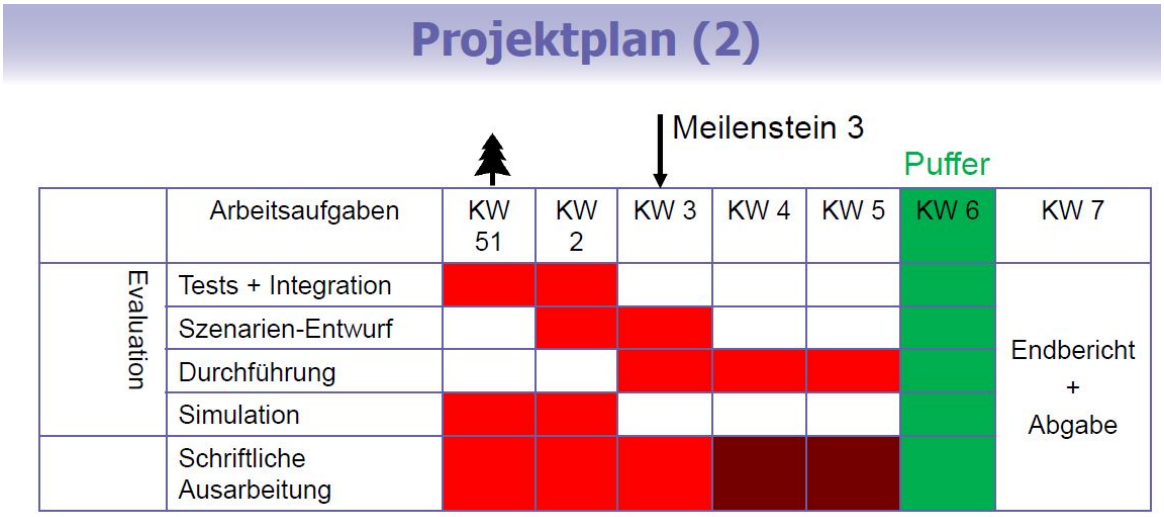


Abbildung 2: Zeitplannung 2018

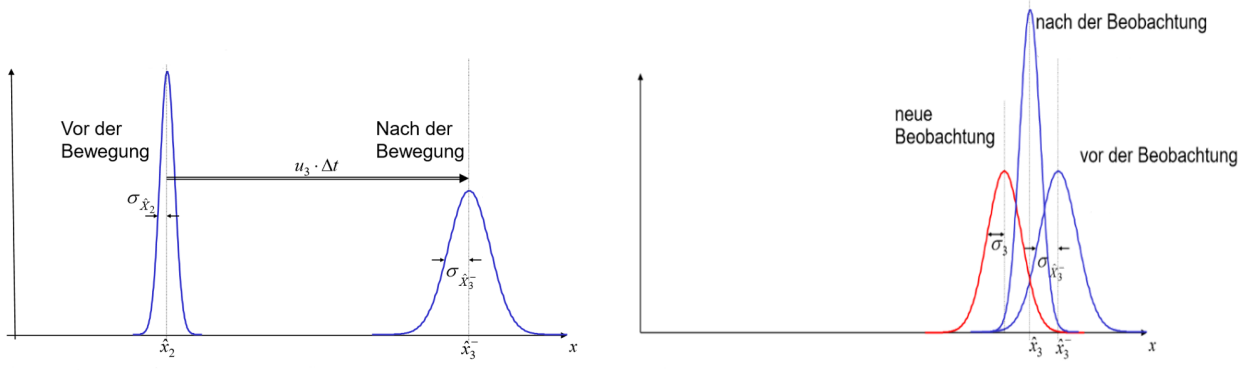
4 Schätztheorie

4.1 Bayes-Filterung

Der allgemeinste Algorithmus zur Berechnung von Posterioren Wahrscheinlichkeiten, in unserem Projekt für die Position und Ausrichtung des Laufroboters, ist durch den Bayes-Filter gegeben. Dieser Algorithmus berechnet die Wahrscheinlichkeitsverteilung aus der Messung und Befehlen. Wir werden zuerst den grundlegenden Algorithmus mit den bisherigen Annahmen angeben. Eine Ablauf des Bayesfilterung-Algorithmus ist in der folgenden Abbildung dargestellt.

1: **Algorithm Bayes_filter(**$bel(x_{t-1}), u_t, z_t$**):**
2: *for all* x_t *do*
3: $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\ dx_{t-1}$
4: $bel(x_t) = \eta\ p(z_t \mid x_t)\ \overline{bel}(x_t)$
5: *endfor*
6: *return* $bel(x_t)$

Die gewünschte Ausgabe ist eine Schätzung des Zustands $bel(x_t)$ zur Zeit t (In unsere Projekt ist dies die Wahrscheinlichkeitsverteilung des aktuellen Zustands). Die Eingabe besteht aus der Schätzung des Vorgängerzustands $bel(x_{t-1})$, zusammen mit der letzten Befehl u_t und der letzten Messung z_t. Die gesuchten Werte sind die Posterioren Wahrscheinlichkeiten $bel(x_t)$ zum Zeitpunkt t. Dieser Updateschritt wird rekursiv angewendet. Erst wird das Schätzung $bel(x_t)$ aus der Schätzung $bel(x_{t-1})$ zuvor berechnet. Deshalb besitzt die Bayes-Filter-Algorithmus aus zwei wesentlichen Schritten: Prediction und Update. Veranschaulicht in folgendem Diagramm.



4.2 Kalman-filter

Der Kalman-Filter dient dazu, Fehler in realen Messwerten zu reduzieren und Schätzungen für nicht messbare Systemgrößen zu liefern. Voraussetzung dabei ist, dass die interessierenden Werte durch ein mathematisches Modell beispielsweise in Form von Bewegungsgleichungen beschrieben werden können. Der Filter ermöglicht den Einsatz in Echtzeitsystemen verschiedener technischer Bereiche. Dazu zählen u.a. die Auswertung von Radarsignalen oder GPS-Daten zur Positionsbestimmung sich bewegender Objekte (Tracking). Deshalb ist er für unsere Projekt zur Positionschätzung geeignet.

Das Kalman-Filter bestimmt wie das Bayes Filter seine Schätzungen rekursiv. Während das Bayes-Filter jedoch mit der Schätzfunktion jeweils die exakte Verteilung des aktuellen Zustands bestimmt, werden beim Kalman-Filter in jedem Schritt nur der Erwartungswert des Zustands und die Kovarianz dieser Schätzung bestimmt. Zur Initialisierung der Rekursion wird für $k = 0$ die Anfangsschätzung $\widehat{X}_0 = X_0$ mit der Kovarianz P_0 gewählt.Da der Kalman-Filter den Zustand eines Systems aus dem Eingangs- und Ausgangssignal schätzt, wird es als Zustandsschätzer bezeichnet. Das Kalman-Filter bestimmt zu jedem Zeitpunkt den Erwartungswert und die Kovarianz der zu schätzenden Größe. Die Berechnungsvorschrift wird nachfolgend hergeleitet.

4.2.1 Prädiktionsschritt

Im Prädiktionsschritt soll die beste Schätzung von X_k und die zugehörigen Kovarianz aus der Befehlsfolge mit den zugehörigen Messungen $u_1, ..., u_k, y_1, ..., y_{k-1}$ und X_0, P_0 bestimmt werden. Außerdem ist die Schätzung des vorherigen Zeitschritts $\widehat{x}_{k-1}$ und deren Kovarianz P_{k-1} verfügbar, welche die beste erwartungstreue Schätzung von X_{k-1} unter Nutzung aller oben genannter Information außer u_k darstellt.

Der Erwartungswert des Systemzustands wird mit Hilfe der Systemgleichung bestimmt wobei der unbekannte Zustand X_{k-1} durch die aktuelle Schätzung $\widehat{x}_{k-1}$ ersetzt werden kann.

$$\widehat{x}_k^- = EX_k = EAX_{k-1} + Bu_k + R_k \tag{3.1}$$

Die Kovarianz der Prädiktion lautet damit

$$P_k^- = Cov(X_k^-) = AP_{k-1}A^T + \Sigma_R \tag{3.2}$$

In der Umformung wurde genutzt, dass das Systemrauschen zum Zeitpunkt k unkorreliert mit Größen vorheriger Zeitpunkte ist. Der erste Summand der Kovarianz gibt die in Schritt $k-1$ bestehende Unsicherheit gewichtet mit der Systemdynamik A wieder. Durch den zweiten Summanden kommt noch das neue Systemrauschen R_k hinzu. Gleichungen (3.1) und (3.2) bestimmen somit, wie aus der Schätzung des Systemzustands und deren Kovarianz zum vorherigen Zeitpunkt$k-1$ eine Prädiktion derselben Größen für den Zeitpunkt k.

4.2.2 Innovationsschritt

Im Innovationsschritt soll zusätzlich die aktuelle Beobachtung y_k berücksichtigt werden. In dieser Schritt wird erst die Verstärkungsmatrix (Kalman Gain) berechnet:

$$K = \widehat{P}_k^- C^T (C P_k^- C^T + \Sigma_S)^{-1} \tag{3.3}$$

Entsprechend erfolgt die Innovation linear, als gewichtete Summe der aktuellen Schätzung und einer Innovation, die proportional zum aktuellen Beobachtungsresiduum ist

$$\widehat{x}_k = \widehat{x}_k^- + K(y_k - C\widehat{x}_k^-) \tag{3.4}$$

Unter Nutzung der Voraussetzung, dass das Beobachtungsrauschen S_k unkorreliert ist mit der Prädiktion, lässt sich die Kovarianz der Innovation berechnen gemäß

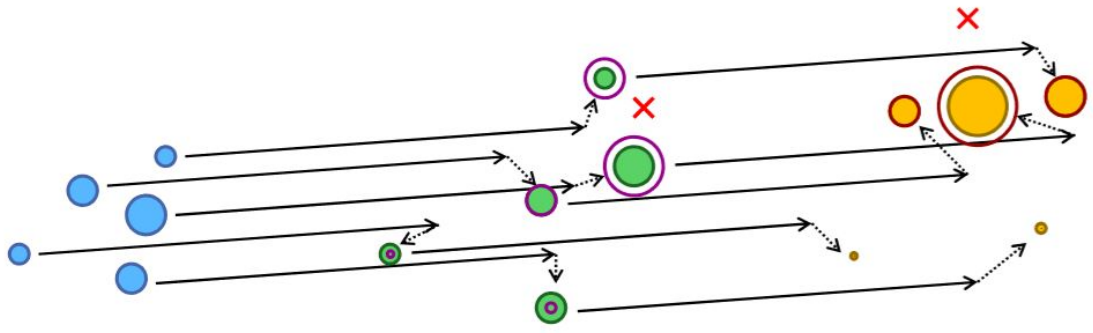
$$P_k = (I - KC)P_k^-(I - KC)^T + K\Sigma_S K^T \tag{3.5}$$

Wie noch gezeigt wird, lässt sich dieser Ausdruck für die optimale Verstärkungsmatrix des KalmanFilters K weiter vereinfachen zu

$$P_k = (I - KC)P_k^- \tag{3.6}$$

4.3 Partikel-Filter

Als Vergleichsverfahren für den SE(2)-Filter wird der Partikel-Filter verwendet. Partikel-Filter ist auch als SMC-Methode bekannt. Die Funktionsweise der Filterung ebenfalls auf der Bayes-Filterung. Der Partikelfilter ist eine alternative nichtparametrische Implementierung des Bayes-Filter. Partikelfilter nähern sich der posterioren Wahrscheinlichkeit an. Es wird eine große Anzahl an Partikeln mit verschiedenen Parametern zu generiert und in dem den Zustandsraum zu füllen ist die Schlüsselidee. Die Posteriori Schätzung $bel(x_t)$ soll durch eine Zufallsmenge dargestellt werden. Von dieser posterioren Verteilung werden Zustandsproben gezogen. Anstatt die Verteilung durch eine parametrische Form darzustellen, stellt der Partikelfilter die Verteilung durch eine Menge von Stichproben aus der aktuelle Verteilung dar. Die ganze Prozess kann durch folgendes Diagramm veranschaulicht werden.



Das oben gezeigtem Bild beschreibt den Prediction- und Update-Schritt des Partikelfilters. Die blauen Partikel ist die Stichproben des aktuellem Zustands, durch die Zustandsüberführung werden die Partikel für den nächsten Schritt erzeugt (in Bild sind dies die grünen Partikel), im nächsten Schritt werden die Partikel durch die aktuelle Messungen korrigiert. Daraus bekommen wir posteriore Stichproben. In unsere Projekt wird die Partikel-Filter durch die PF-Klasse von robotics system toolbox eingesetzt. Damit können wir direkt das Systemmodell für die Zustandüberführung und das Messmodell des Partikel-Filter anpassen.

5 SE(2) Filter

5.1 Lie Gruppe

Eine Lie-Gruppe G ist eine Mannigfaltigkeit G, welche zusammen mit einer glatten Operation

$$G \times G \ni (g_1, g_2) \to g_1 \cdot g_2 \in G$$

eine Gruppe ist.

Ein Vorteil der abstrakten Definition von Lie-Gruppe besteht darin, dass sie wesentlich flexibler ist. Zum Beispiel gilt folgendes: Ist G eine Lie-Gruppe und N ein abgeschlossener Normalteiler, dann ist G/N wieder eine Lie-Gruppe.

5.2 SE(2)-Bingham-Verteilung

In diesem Projekt verwenden wir einen Filter, der auf der SE(2)-Bingham-Verteilung für zweidimensionale Translationensvektoren und einem Skalar für die Rotation basiert. Die SE(2)-Bingham-Verteilung kann zur Darstellung der Orientierungen auf der Ebene verwendet werden. [5] Anstatt sich auf die auf Gauß'sche-Verteilung basierende Annäherungen zu verlassen, haben wir uns entschieden, alle auftretenden Wahrscheinlichkeiten durch SE(2)-Bingham-Verteilung darzustellen. Diese verwendet für die Translation weiterhin eine zweidimensionale Normalverteilung, kombiniert mit einer Bingham-Verteilung, die auf der Hypersphere der Rotationen definiert wird. Die antipodisch Symmetrie ist in Abbildung 3 gezeigt.

In unsere Fall ist die Bingham-Verteilung einfacher für die Darstellung der Orientierung des Laufroboters. Als eine Folge der obigen Motivation kann man sehen, dass die Bingham- Wahrscheinlichkeitsdichtefunktion (pdf) zwei symmetrische Moden aufweist. Die Parametermatrix der Bingham-Verteilung aus der Exponentialfunktion (welche die inverse Kovarianzmatrix im Gauß'schen Fall ist) wird üblicherweise zerlegt in eine orthogonale und eine diagonale Matrix. Dies führt zu der folgenden Definition:

$$f\left(\underline{x}\right) = \frac{1}{F} \cdot exp\left(\underline{x}^T M Z M^T \underline{x}\right) \tag{1}$$

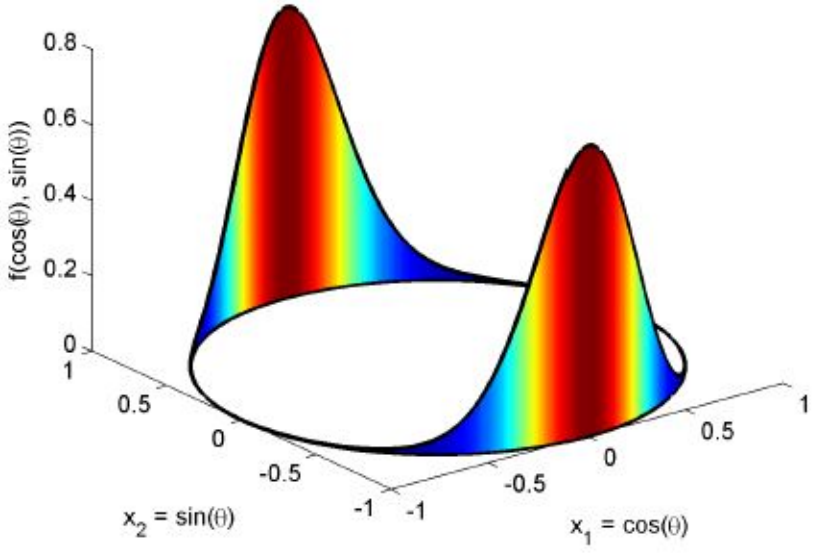


Abbildung 3: Darstellung von hypersphärischen Zufallsvektoren [2]

5.3 Duale Quaternionen

In unsere Projekt werden duale Quaternionen zur Darstellung der Starrkörperbewegung verwendet. Die dualen Quaternionen werden durch folgende Formel darstellt [1]

$$\sigma = p + \epsilon q \tag{2}$$

Unter Ihnen die p steht für jeweilige Quaternionen

$$p = p_0 + \underline{p} = p_0 + p_1\underline{i} + p_2\underline{j} + p_3\underline{k} \tag{3}$$

$$q = q_0 + \underline{q} = q_0 + q_1\underline{i} + q_2\underline{j} + q_3\underline{k} \tag{4}$$

Für die duale Einheit ϵ gitl $\epsilon^2 = 0$.

Die Rotationsmatrix R, Translationsvektor t kann kompakt durch eine duales Quaternion dargestellt werden. Der Translationsvektor t ist ein reines Quaternion. Wir kombinieren es mit dem Rotationsquaternion $r = \cos\frac{\theta}{2} + \underline{\widehat{u}}sin\frac{\theta}{2}$, das die Rotation um den Vektor $\widehat{u}$ beschreibt, zu dem folgendem dualem Quaternion:

$$\sigma = r + \frac{\epsilon}{2}\underline{t}r = cos\frac{\theta}{2} + \underline{\widehat{u}}sin\frac{\theta}{2} + \frac{\epsilon}{2}\left(-sin\frac{\theta}{2}\left(\underline{t}\cdot\widehat{u}\right) + cos\frac{\theta}{2}\underline{t} + sin\frac{\theta}{2}\underline{t}\times\widehat{u}\right) \tag{5}$$

In unseren Fall wird die Bewegung der Roboter hauptsächlich auf eine Eben durchgeführt werden. Deshalb stehen $\underline{t}_x$ und $\underline{t}_y$ für die Translation des Roboters und θ für die Rotation. Um Rotationen und Translationen in 2D zu kombinieren, wird hier duale Quaternion-Multiplikation

verwendet. Eine Rotation mit anschliesender Translation wird durch die Multiplikation beschrieben.

$$\left[1 + \epsilon\frac{1}{2}\left(t_x\underline{i} + t_y\underline{j}\right)\right] \cdot \left[cos\left(\frac{\alpha}{2}\right) + sin\left(\frac{\alpha}{2}\right)\underline{k}\right] \tag{6}$$

6 Crawler

In unsere Projekt wird ein Laufroboter verwendetder vom Labor für Intelligente Sensor-Aktuator-Systeme (ISAS) selbst aufgebaut wurde. Es bestehen viele verschiedene Versionen des Crawlers. Diese Roboter werden für mehrere Zwecke verwendet. Darüber hinaus ist das Chassis von diesen Roboter komplett in 3D gedruckt. Daher können diese sehr schnell gebaut werden.

6.1 Aufbau

Der Crawler, der im Projekt benutzt wird, ist die neueste Version. Die Hauptkomponente der Version ist ein Arduino Yun.
Der Arduino Yun verfügt über ein WLAN-Modul, mit dem ein WLAN generiert Oder in ein bestehendes WLAN eingefügt wird. Der Craeler verfügt es über drei verschiedene Beine, wobei jedes einzeln mittels Servomotor gesteuert werden kann. Dies ermöglicht dem Roboter, sich gerade, seitlich und omnidirektional zu bewegen.
Der Crawler enthält vier Ultraschallsensoren und eine IMU. Es sind insgesamt vier Ultraschallsensoren installiert, die jeweils aus einem bestehen Empfänger und ein Sender existieren. Die Sensoren sind in 90-Winkeln zueinander angeordnet
Zusätzlich befinden sich oben zwei LEDs (gelb und rot), mit denen der Roboter einfacher, zum

Abbildung 4: Der Crawler

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \Phi_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \Phi_k \end{bmatrix} + \begin{bmatrix} cos(\Phi_k) & -sin(\Phi_k) & 0 \\ sin(\Phi_k) & cos(\phi_k) & 0 \\ 0 & 0 & \Phi_k \end{bmatrix} \begin{bmatrix} \widehat{s}_k^F \\ \widehat{s}_k^S \\ \widehat{\omega}_k \end{bmatrix} + \begin{bmatrix} \omega_k^x \\ \omega_k^y \\ \omega_k^\Phi \end{bmatrix}$$

Abbildung 5: Umwandlung in Crawlerkoordinaten

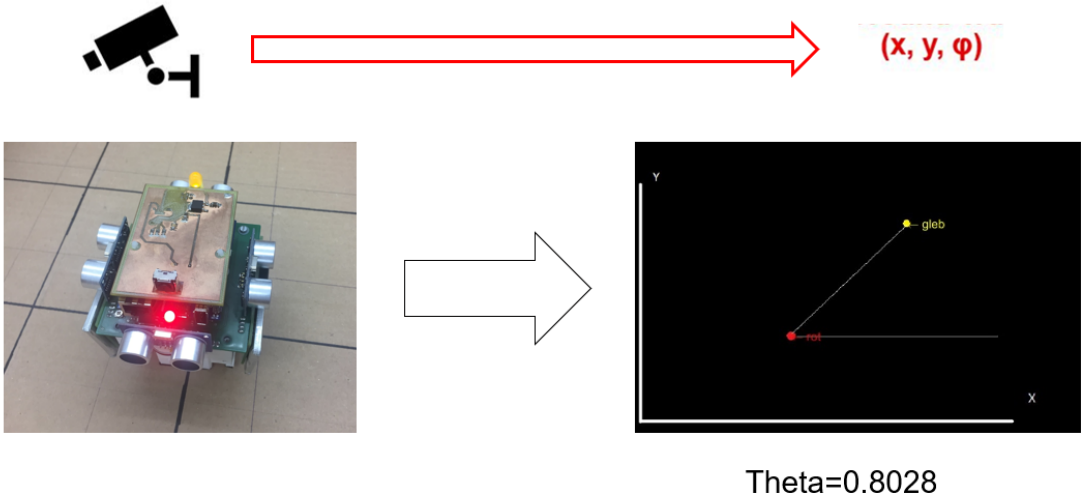


Abbildung 6: Erkennung der LEDs aus dem Kamerabild, zum Tracking des Laufroboters.

Beispiel von einer Deckenkamera, verfolgt werden kann. Das Design der Roboter ermöglicht eine Vielzahl unterschiedlicher Grundfunktionen Bewegungsmuster.

6.2 Position- und Rotationschätzung mittels Kameratracking

Zur bessere Simulation und Evaluierung muss die genau Position und Rotation ermittelt werden. Dazu wird ein Deckenkamerasystem verwendet. Zur Ermittelung der Position unseres Laufroboters wird ein rote LED auf dem Crawler aktiviert, die durch das an der Decke befestigte Kamerasystem aufgenommen wird. Durch eine Farberaum-Umwandlung kann die rote Markierung im Bild zuverlässig gefunden werden. Mittles anschließender Koordinatentransform, die und Gleichung 5 gezeigt ist, wird die Position des echten Crawlers ermittelt und umgewandelt. Für die Rotation wird entsprechend eine zweite gelbe LED in Crawler eingesetzt, durch den bekannten Aufbau kann die Bewegungsrichtung durch das Kamerasystem erkannt werden. Die gesammelten Bewegungsdaten können zusammen mit den Messdaten des Crawlers als Ground Truth exportiert werden. Das ganze Prozess wird durch unten stehend Bild dargestellt.

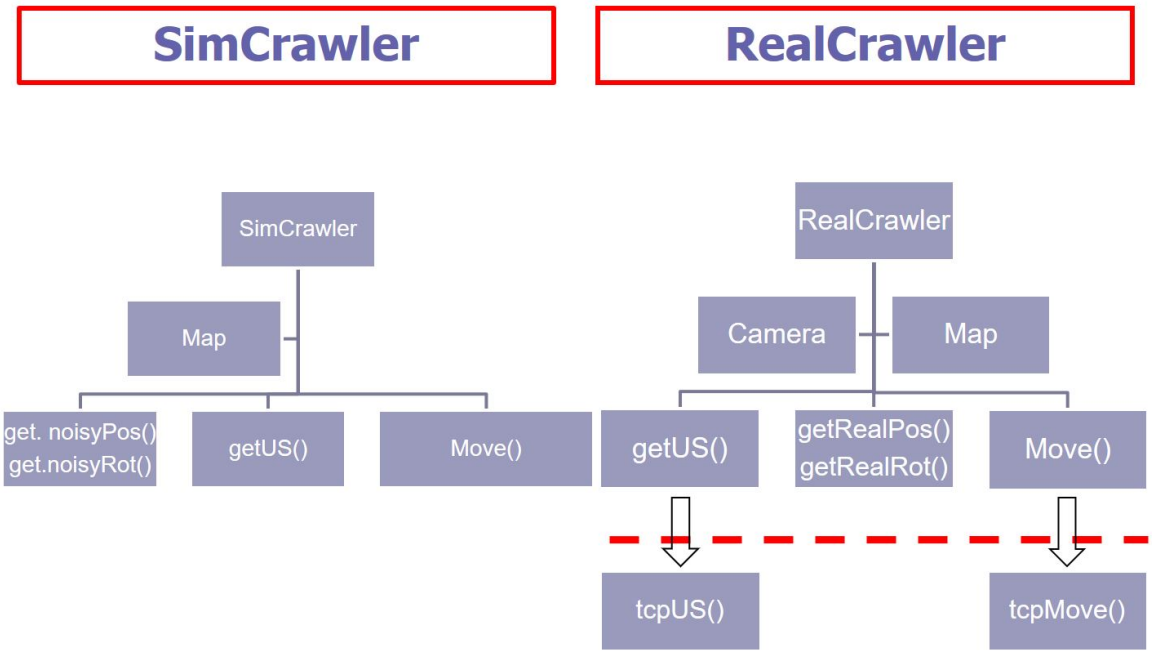


Abbildung 7: Gemeinsam genutzte Schnittstelle der Crawler-Klassen

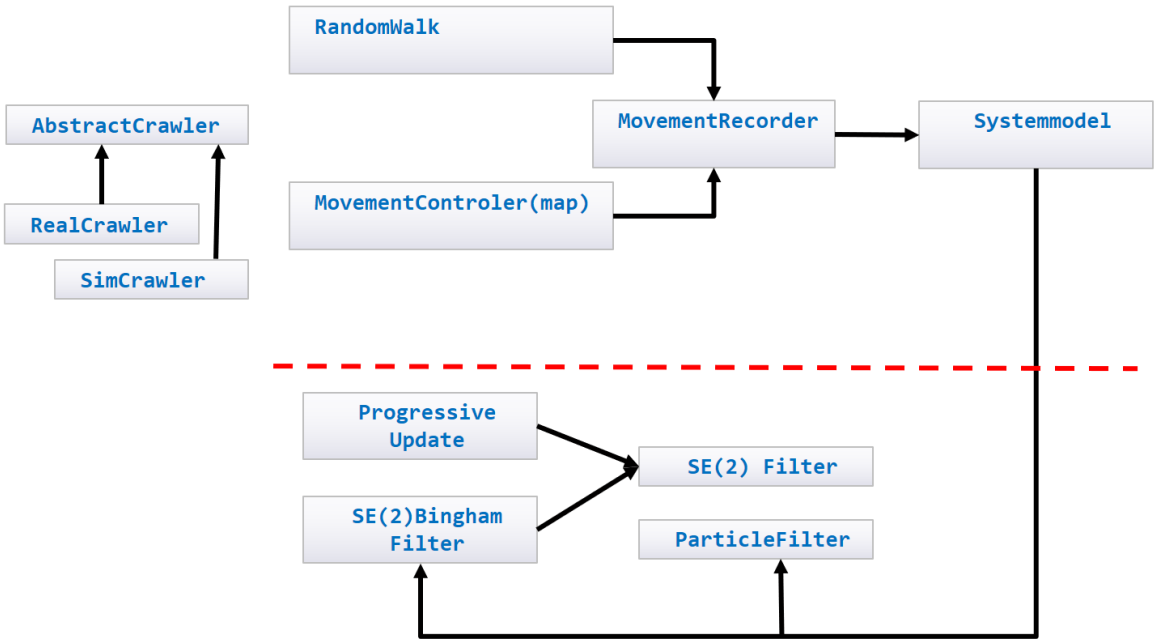


Abbildung 8: Aufbau des Simulationsframeworks

7 Simulation

Zur Überprüfung der Implementierung wurde die Bewegung des Crawlers zunächst simuliert. Die Simulation bietet ein Ground Truth ohne aufwändiges Kameratracking vorteilhaft, sowie verschiedene Karten. Diese können als beliebige konvexe Polygone definiert werden, in denen sich der Crawler frei bewegen kann. Um Komponenten der Simulation zur Auswertung des Experiments wiederverwenden zu können, wurde auf einen modularen Aufbau geachtet. Die grobe Struktur des Frameworks wird in 8 verdeutlicht. Es gibt eine einheitliche Schnittstelle, die zur Kommunikation mit der realen Crawler-Hardware, sowie dem simuliertem Crawler genutzt wird. Die Steuerung ist über drei verschiedene Modi möglich:

- Auswahl der Bewegungsanweisungen durch den Benutzer
- Zufällige Bewegungsanweisungen
- Wiedergabe eines aufgezeichneten Bewegungspfads des echten Crawlers

Crawler Modell Das vereinfachte Modell des Crawlers, das zur Simulation verwendet wurde ist in 9 zu sehen. Der Ursprung des Koordinatensystems liegt auf der vorderen roten LED, die zum Tracking der Position verwendet wurde. Der eigentlich kegelförmige Messbereich der vier Ultraschallsensoren wurde als Halbgerade modelliert.

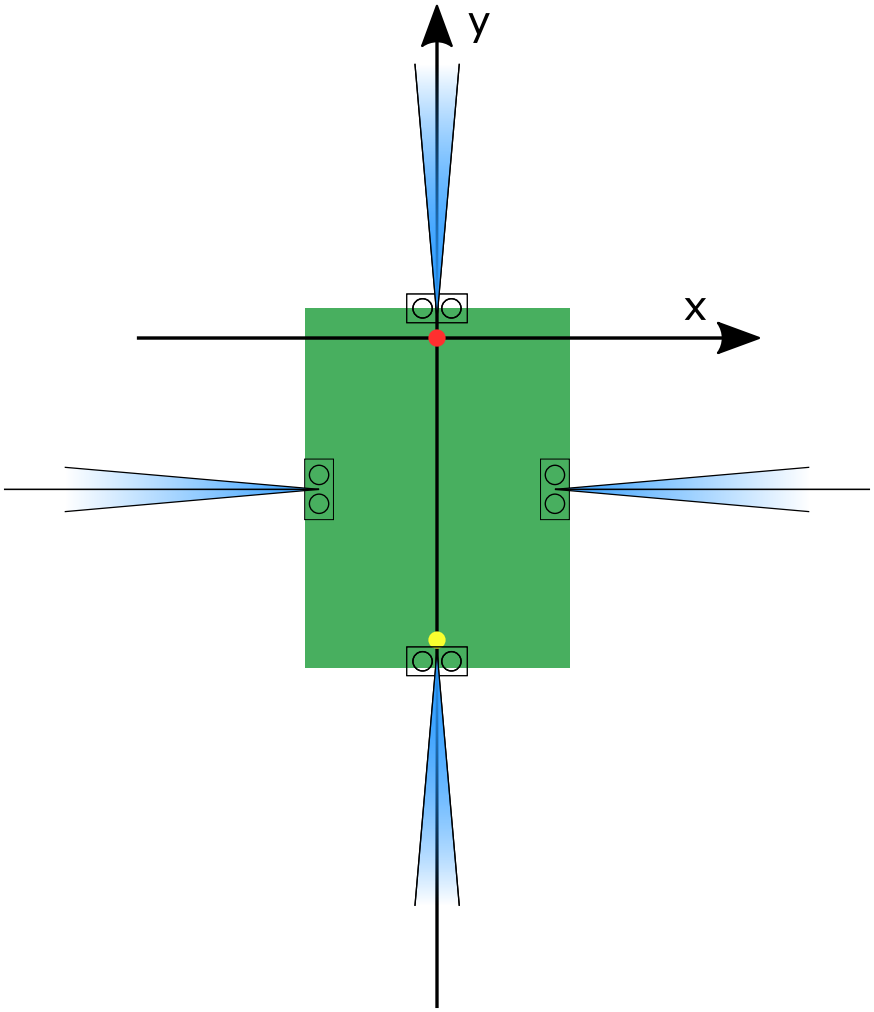


Abbildung 9: In der Simulation verwendetes Schema des Crawlers mit lokalem Koordinatensystem. Markeirt sind die vier Ultraschallsensoren mit Messbereich, sowie die zwei verschiedenfarbenen LEDs.

7.1 Systemmodell

Vorwissen über das System findet im Systemmodell Anwendung. Mit der Überführungsfunktion a wird der Folgezustand $\underline{x}_{t+1}$ aus dem aktuellem Zustand $\underline{x}_t$ abgeleitet. Der verwendete Zustandsraum $\mathbb{S} \times \mathbb{R}^2$ beschreibt die position und Orientierung des Crawlers.

$$\underline{x}_{t+1} = a(\underline{x}_t) \oplus w_t$$

Die Folge von Bewegungsbefehlen an den Crawler ist bekannt. Je nach Bewegungsbefehl ist die entsprechende Überführungsfunktion zu wählen:

$$a(\underline{x}_t, command) = \begin{cases} a_{fwd} \oplus w_{fwd}, & \text{falls } command = fwd \\ a_{left} \oplus w_{left}, & \text{falls } command = left \\ a_{right} \oplus w_{right}, & \text{falls } comman = right \end{cases} \tag{7}$$

Die tatsächliche ausgeführten Schritte variieren jedoch, auch für gleichbleibende Bewegungsrichtungen. Dies wir in Abbildung 10 deutlich. Deswegen wird bei jedem Schritt zusätzlich ein Rauschterm w_t addiert. Dies lässt sich in SE(2) durch die Multiplikation der dualen Quaternionen darstellen, was durch den $\oplus$-Operator ausgedrückt wird. Die Schrittlängen und Drehungen wurden in einem Vorexperiment durch Tracking der LEDs mit der Deckenkamera aufgenommen. Es wurden drei SE(2)-Bingham-Verteilungen auf die gewonnen Daten angepasst, aus denen die jeweiligen w_t zufällig gezogen werden.

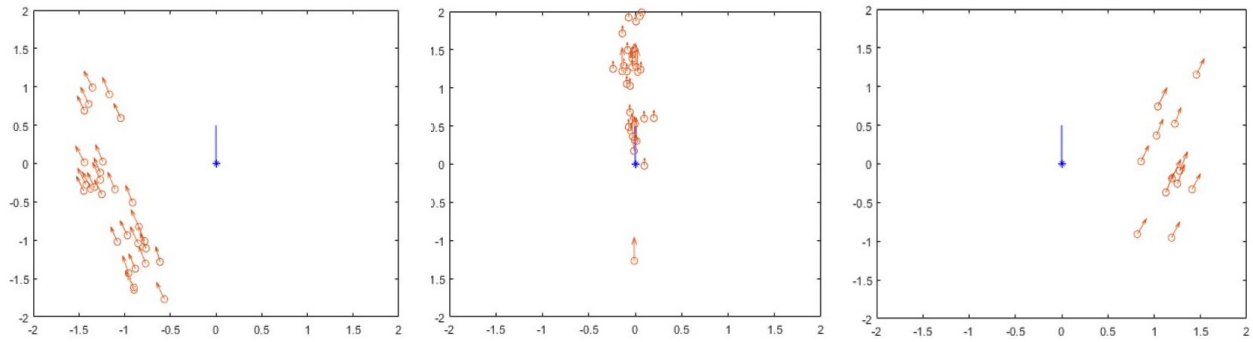


Abbildung 10: Gemessene Verteilungen der Position und Rotation des Laufrobotters nach der Ausführung einer Bewegung. Blau ist der Ausgangszustand dargestellt. Die Zustände nach je einem Schritt in rot. Jede Verteilung wurde für einen der drei Bewegungsbefehle 'links', 'vorwärts' und 'rechts' mit einer Deckenkamera gemessen.

Daraus werden die Erwartungswert und Varianz berechnet,und wir bezeichnen diese Datensätze als die echte Systemrauschen von Roboter.

7.2 Messmodell

Der Crawler kann seine momentane Position und Ausrichtung nicht direkt messen. Es stehen jediglich vier nach vorne, hinten, links und rechts gerichtet Ultraschallsensoren zur Verfügung, die Abstandsmessungen zu den nächstliegenden Wänden vornehmen könne. Die Messwerte hängen bei einer festen Karte von der Position und Ausrichtung des Crawlers ab. Mit dem Messmodel wird beschrieben, welche Messergebnisse bei einem Systemzustand zu erwarten sind. Eine Messung ist Abbildung vom dreidimensionalen Zustandsraum des Laufroboters $\mathbb{S} \times \mathbb{R}^2$ in einen vierdimensionalen Messraum $\mathbb{R}^{+4}$.

Da die Ultraschallsensoren des Crawler nur grobe Abstandsmessungen ermöglichen, wird ein normalverteiltes mittelwertfreies Rauschen $w_t \sim \mathcal{N}(0, \sigma)$ addiert:

$$\underline{z}_t = h(\underline{x}_t) \oplus w_t$$

$h(x_t)$ für einen Zustand x_t zu berechnen ist leicht. Es muss lediglich eine Messung auf einer virtuellen Karte an der von x_t beschriebenen Position durchgeführt werden. In unserem Fall lässt sich h jedoch nicht einfach invertieren. h^{-1} muss nicht eindeutig sein. Die selbe Messung könnte kann in mehreren verschiedenen Zuständen beobachtet werden. Aus h lässt sich ohne weiteres die eine Likelihood $f(z_t|x_t)$ ableiten [3]. Diese beschreibt die Plausibilität einer Messung für einen gegebenen Zustand. Die Likelihood kann folgendermaßen berechnet werden $f(z_t|x_t) = f_{\mathcal{N}}(h(x_t) - z_t)$, wobei $f_{\mathcal{N}}$ komponentenweise in eine Normalverteilung $\mathcal{N}$ einsetzt und die resultierenden Wahrscheinlichkeiten multipliziert. Die Abbildung 11 zeigt den Crawler in der Position im linken Bild, mit der Likelihood-Funktion für den gezeigten Zustand rechts. Der hell gelbe Punkt rechts, mit der größten Wahrscheinlichkeit befindet sich an der Stelle des Crawlers, von dem die Messung z_t stammt.

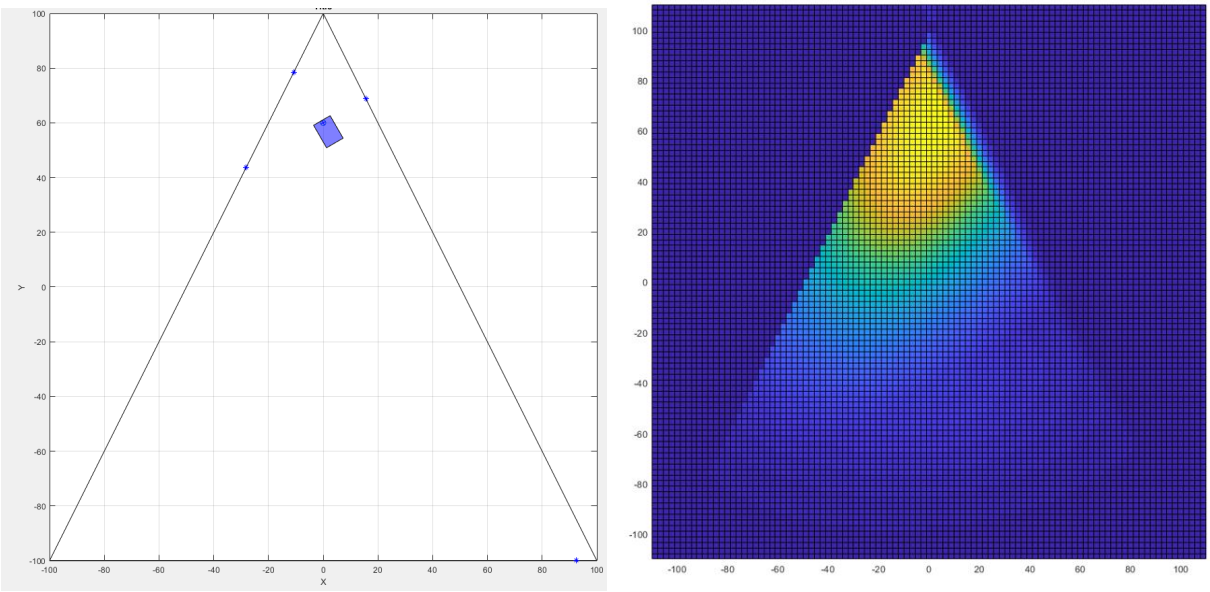


Abbildung 11: Linkes Bild: In blau ist der Crawler auf der dreieckigen Karte eingetragen. Rechts die Likelihood-Funktion $f(z_t|x_t)$, für die Messung, die der Crawler im linken Bild aufnimmt.

7.3 Progressive Updates

Nach Anwendung des Systemmodells im Prediction-Schritts folgt der Aktualisierungsschritt, der den geschätzten Zustand x_t^e durch Einbeziehung der Messung z_t korrigiert Beim Progressiven Update werden vom aktuell geschätze Systemzustand x_t^e dazu zunächst deterministisch Proben gezogen. Anstatt diese in einem Schritt mit den Werten der Likelihood-Funktion neu zu gewichten, wird iterativ vorgegangen [4]. Dies ist nach der Regel von Bayes möglich, wenn die $\lambda_1 + \lambda_s$ eine Zerlegung der 1 bilden. Dadurch kann eine Degenerierung der Proben mit Likelihoodwerten nahe null vermieden werden. Degenerierte Proben verringern die Genauigkeit des Filters. Außerdem ermöglicht das Progressive Update die Korrektur des geschätzten Orts ohne die Messfunktion h invertieren zu müssen. Gemessene Werte können direkt in die Likelihood eingesetzt werden.

$$\begin{aligned} f(\underline{x}|\underline{z}) &\propto f(\underline{z}|\underline{x}) \cdot f(\underline{x}) \\ &= (f(\underline{z}|\underline{x})^{\lambda_1} \cdot ... \cdot f(\underline{z}|\underline{x})^{\lambda_s}) \cdot f(\underline{x}) \end{aligned}$$

Durch den folgenden Ablauf wird garantiert, dass der Quotient zwischen dem kleinsten neuen Gewicht l_{min} und das größte neue Gewicht l_{max} nicht über einen vorbestimmten Schwellwert steigt. Dieser Schwellwert wird mit τ bezeichnet.

1: **procedure** UPDATE(measurement z_t, likelihood $f(z_t|x_t)$, predicted density x_t^e)
2: $\Lambda \leftarrow 1$
3: $x_t \leftarrow x_t^e$
4: **while** $\Lambda > 0$ **do**
5: $(s_1, ..., s_L, \omega_1, ..., \omega_L) \leftarrow$ sampleDeterministicly(x_t)
6: $\omega_{min} \leftarrow min(f(z_k|s_k))$
7: $\omega_{max} \leftarrow max(f(z_k|s_k))$
8: $\lambda \leftarrow min(\Lambda, \frac{log(\tau)}{log(\omega_{min}/\omega_{max})})$
9: **for** $j \leftarrow 1$**to**L **do**
10: $\omega_k \leftarrow \omega_k \cdot f(z_k|x_k)^{\lambda}$
11: **end for**
12: $x_t \leftarrow matchBingham(s_1, ..., s_L, \omega_1, ..., \omega_L)$
13: $\Lambda \leftarrow \Lambda - \lambda$
14: **end while**
15: **end procedure**

8 Evaluation

Zur Evaluierung wurde der Crawler auf einer einfachen rechteckigen Karte zufällig bewegt. Dabei wurde aus drei Bewegungsmöglichkeiten nach Tabelle 1 probabilistisch eine ausgewählt, und ausgeführt, sofern dies ohne eine Kollision des Crawlers mit der Wand möglich ist. Die Position und Orientierung des Laufroboters werden parallel vom SE(2)-Filter und Partikelfilter geschätzt. Dann wird jeweils der Fehler zum Ground Truth berechnet. Für den Partikelfilter wurden 1000 eingesetzt, die Rotationskoordinate wurde in *mathtbbS* berechnet. In den Anfangsverteilungen wurden jeweils alle Koordinaten als stochastisch unabhängig voneinander angegeben.

Tabelle 1: Mögliche Bewegungen

Notation	Erklärung	Wahrscheinlichkeit
fwd	Schritt gerade nach vorne	50%
$left$	Drehung nach links	25%
$right$	Drehung nach rechts	25%

Als Ground Truth wird der Crawler mit einer Deckenkamera getrackt. Über zwei verschiedenfarbige LEDs auf dem Crawler lässt sich dessen Position und Rotation bestimmen.

Aufgrund des hohen Zeitaufwands für das Aufnehmen von Bewegungspfaden des echten Crawlers, wurden zusätzlich in der Simulation erstellte zufällige Bewegungspfade als Ground Truth verwendet. Hierbei gibt es mehrere Möglichkeiten, welche Verteilung für das Rauschen im Systemmodell verwendet wird. Um den SE(2)-Filter nicht zu bevorteiligen, wenn im Systemmodell genau die Bingham-Verteilung des Filters verwendet wird, wurde eine weitere Testreihe mit Normalverteiltem Rauschen durchgeführt. Die hierfür benötigten Mittelwerte und Varianzen wurden für jeden der drei Freiheitsgrade x, y, θ unabhängig aus den Messdaten des Vorexperiments bestimmt. Tabelle 2 zeigt die insgesamt sechs Kombinationen aus den zwei Verfahren und drei Ground Truths. Für die Simulationen wurden die drei Karten aus Abbildung 12 mit jeweils 100 zufälligen Schritten betrachtet. Die Bewegungsbefehle waren für beide Verfahren pro Karte gemeinsam generiert. Für den realen Versuch wurde nur die quadratische Karte untersucht, mit zwei Bewegungsmustern von 60 und 40 zufälligen Schritten, die nacheinander ausgeführt wurden.

Tabelle 2: Testreihen mit Anzahl Schritten

Rauschen	SE(2)-Filter	Partikelfilter
Normalverteilt	300	300
SE(2)-Bingham	300	300
Reale Daten	60+40	60+40

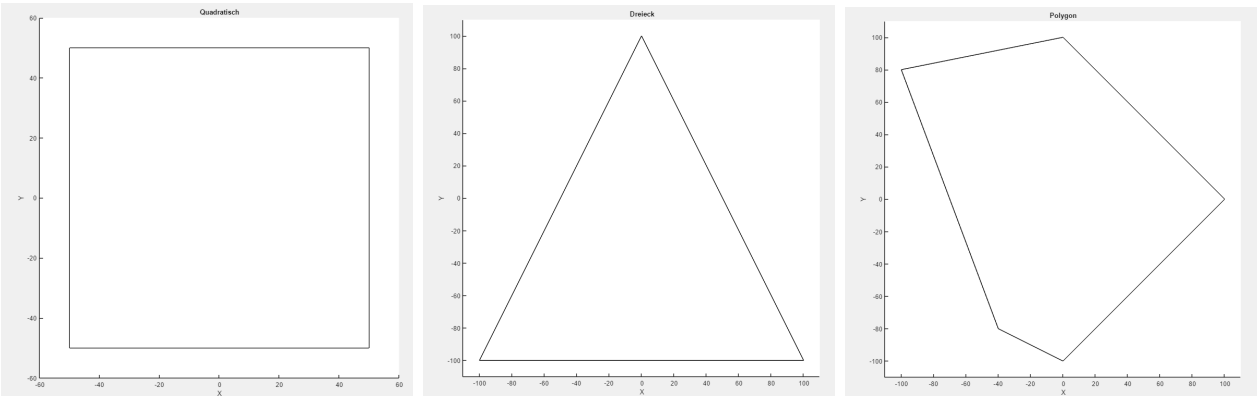


Abbildung 12: Die drei zur Auswertung verwendeten Kartentypen

8.1 Ergebnisse

Tabelle 3 stellt eine Übersicht der Evaluationsergebnisse. Die Fehlerdistanzen wurden über alle Schritte gemittelt. Abbildung 13 und 14 zeigen den Fehlerverlauf von jeweils zwei Testreihen auf der quadratischen Karte. Hier wurde die euklidische Distanz zwischen der Position des Laufroboters und Ground Truth verwendet. Wie zu erwarten ist jeweils das Fiter-Verfahren, dessen angenommene Rauschverteilung mit der des tatsächlichen Systemmodells übereinstimmt etwas besser. Auffällig ist außerdem, dass beim SE(2)-Bingham-Rauschen der Fehler beider Verfahren deutlich größer ist.

Die Diagramme in Abbildung 16 zeigen den Rotationsfehler. Die unterschiedliche Zahl an Nulldurchgängen könnte ein Hinweis darauf sein, dass der Partikelfilter sich im Vergleich träger verhält.

Abbildung 17 zeigt den Positionsfehler für ein Bewegungsmuster des echten Crawlers. Die Lokalisations funktioniert deutliche schlechter, als in Simulation. Teilweise sogar gar nicht, wenn der Anfangszustand zur Initialisierung der Filter zu weit vom der tatsächlichen Position abweicht. Besonders empfindlich reagiert die Lokalisation, wenn der anfängliche Rotoationsfehler groß ist. Um einen signifikanten Unterschied zwischen den beiden Verfahren feststellen zu können, würde eine wesentlich größere Anzahl an aufgezeichneten Schritten des echten Crawler nötig werden. Geringe Unterschiede in der Laufzeit lassen sich dennoch bemerkbar. Der Partikelfilter mit 100 Partikeln wertet die Messfunktion wesentlich häufiger aus, als der SE(2)-Filter mit dem Progressiven Update.

Tabelle 3: Mittlere Fehler

Rauschen	SE(2)-Filter	Partikelfilter
Normalverteilt	6.7	5.8
SE(2)-Bingham	7.5	8.8
Reale Daten	11.2	10.4

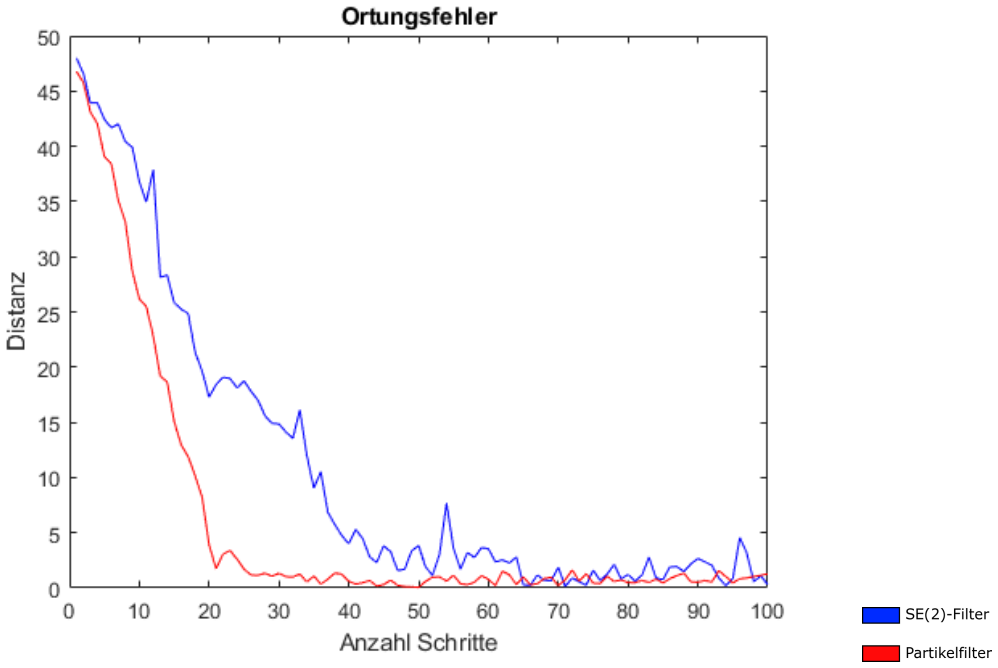


Abbildung 13: Normalverteiltes Rauschen

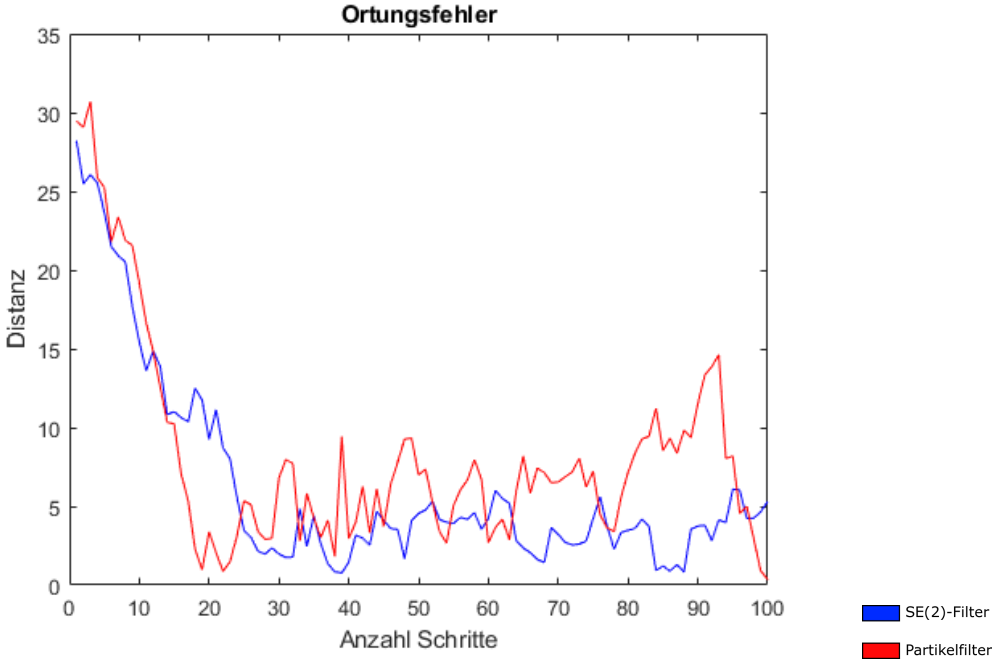


Abbildung 14: SE(2)-Bingham-verteiltes Rauschen

Abbildung 15: Ergebnisse der Simulation mit Quadratischer Karte

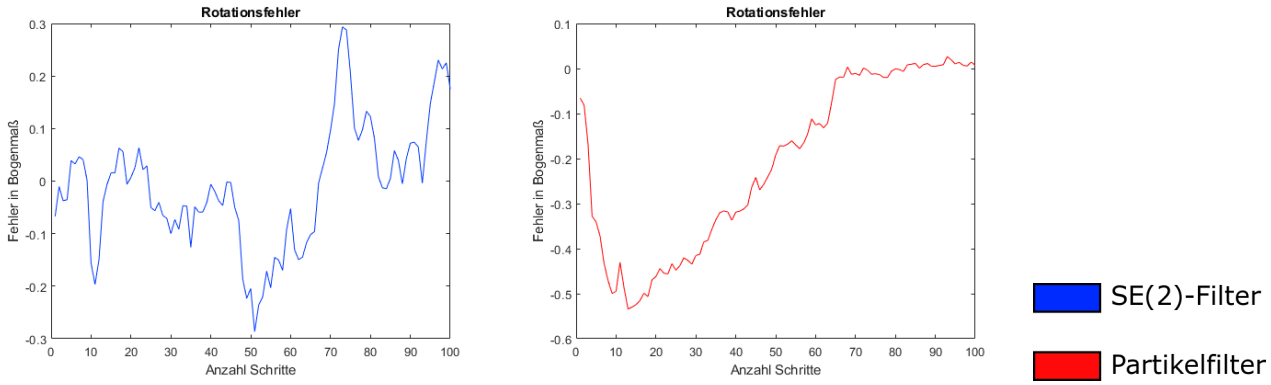


Abbildung 16: Rotationsfehler für quadratische Karte mit SE(2)-Bingham-Rauschen

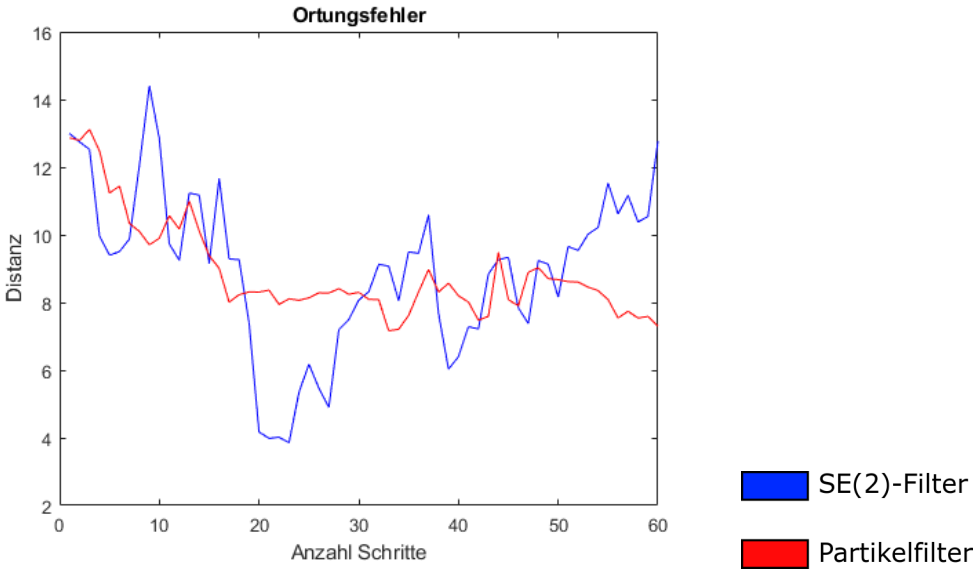


Abbildung 17: Fehler für die Bewegung des echten Crawlers

9 Zusammenfassung und Ausblick

Es wurde der SE(2)-Filter, sowie nötige Grundlagen der Schätztheorie, über Bayes-Filterung und duale Quaternionen vorgestellt. Gängige Verfahren wie der Partikel- und Kalman-Filter wurden beschrieben. Die Implementientierung des Filters, insbesondere System- und Messmodell, sowie das Progressive Update wurden erläutert. Zuletzt wurden die Ergebnisse der Evaluation des SE(2)-Filters im Vergleich zum Partikelfilter dargestellt. Es konnte kein signifikanter Unterschied zwischen den beiden Schätzmethoden festegestellt werden, dazu müssten mit größere Mengen an Bewegungsschritten des Crawlers getestet werden. Weiter mögliche Erweiterungen sind denkbar. Es können weitere Versuche mit neuen Karten vorgenommen werden. Die quadratische Karte war die einzige, mit der echte Daten gesammelt wurden. Eine komplexere Messfunktion, die die Kegelform des Ultraschallmessfelder berücksichten, würde den Crawler besser modellieren. Nur aus den vier stark verrauschten Distanzdaten auf die Position des Laufroboters zu schließen ist ein schwieriges Problem. Es wäre interessant zu überprüfen, wie sich die Verfahren verhalten, wenn enger an die Postion gekoppelte Messgrößen beobachtet werden, zum Beispiel durch die IMU des Crawlers. In späteren Versuchen könnte man sich von der Ebene loslösen und die Lokalisation in 3D fortsetzen.

References

[1] Jia, Yan-Bin (2013). Dual quaternion. Com S, 477, 577.

[2] Kurz, G., Gilitschenski, I., Julier, S., Hanebeck, U. D. (2014). Recursive Bingham filter for directional estimation involving 180 degree symmetry. Journal of Advances in Information Fusion, 9(2), 90-105.

[3] Gilitschenski, I., Kurz, G., Julier, S. J., Hanebeck, U. D. (2014, July). A new probability distribution for simultaneous representation of uncertain position and orientation. In Information Fusion (FUSION), 2014 17th International Conference on (pp. 1-7). IEEE.

[4] Kurz, G., Gilitschenski, I., Hanebeck, U. D. (2016). Recursive Bayesian filtering in circular state spaces. IEEE Aerospace and Electronic Systems Magazine, 31(3), 70-87.

[5] Gilitschenski, I., Kurz, G., Hanebeck, U. D. (2015, September). A stochastic filter for planar rigid-body motions. In Multisensor Fusion and Integration for Intelligent Systems (MFI), 2015 IEEE International Conference on (pp. 13-18). IEEE.

[6] Kurz, G., Gilitschenski, I., Hanebeck, U. D. (2014, June). Nonlinear measurement update for estimation of angular systems based on circular distributions. In American Control Conference (ACC), 2014 (pp. 5694-5699). IEEE.