Diplomarbeit
im Studiengang Informatik

Multimodales Tracking und Trajektorien Vorhersage

am Arbeitsbereich für
Technische Aspekte Multimodaler Systeme,
Universität Hamburg

vorgelegt von
Martin Weser
April 2006

betreut von
Prof. Dr. Jianwei Zhang
Prof. Dr. Bernd Neumann

Kurzfassung

Um Interaktion mit einem Serviceroboter zu ermöglichen, muss dem Roboter ein sensorbasiertes Modell der Umwelt vorliegen, das die Positionen der Interaktionspartner beinhaltet. Neben der momentanen Position können auch Vorhersagen über Aufenthaltsorte zu späteren Zeitpunkten wichtig sein, um beispielsweise eine effiziente Pfadplanung bei Botendiensten zu ermöglichen. In dieser Diplomarbeit wird ein konzeptioneller Rahmen für Systeme entwickelt, die unterschiedliche Sensormodalitäten in ein robustes Gesamttracking integrieren und eine Vorhersage der Trajektorien von Personen aufgrund gelernter Bewegungsmuster treffen. Darauf aufbauend wird ein System implementiert, das Trackingalgorithmen in Laserentfernungsmessungen und Kamerabildern durch einen Partikelfilter fusioniert. Die Auswirkungen von Fehlern in den einzelnen Sensormessungen auf das Gesamttracking können durch Multimodalität verringert werden. Die beobachteten Trajektorien werden durch eine Self Organizing Map zu Bewegungsmustern generalisiert. Diese Bewegungsmuster sind Grundlage einer Vorhersage der Trajektorien von beobachteten Personen. In praktischen Versuchen wird gezeigt, dass die Robustheit und Genauigkeit von Trackingalgorithmen durch Multimodalität verbessert werden kann. Desweiteren können typische Bewegungsmuster durch eine Self Organizin Map gelernt und generalisiert werden, und diese Bewegungsmuster für eine realistische Vorhersage von Bewegungen genutzt werden.

Abstract

A sensor-based model of a service robot's environment is a prerequisite for interaction. Such a model should contain the positions of the robot's interaction partners. Additionally to the actual positions of the partners it is important for the service robot to predict their possible future positions. This knowledge could for example be used to realize efficient path planning for delivery tasks. This diploma thesis propose an extensible framework for systems, that combine different sensor modalities in a general tracking system. Based on this conceptual framework a tracking system that fuses tracking algorithms in laser range scans as well as in camera images by a particle filter is implemented. The observed trajectories are generalized to trajectory patterns by a novel method which uses Self Organizing Maps. Those patterns are used to predict trajectories of the currently observed persons. Practical experiments show that multimodality increases the system's robustness to incorrect measurements of single sensors. It is also demonstrated that a self organizing map is suitable for learning and generalizing trajectories. Convenient predictions of future trajectories are presented which are deduced from these generalizations.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Serviceroboter werden in Zukunft die Arbeit zu Hause oder im Büro mehr und mehr unterstützen. Mögliche Aufgabenfelder reichen von Botendiensten über Reinigungsarbeiten bis zu Arbeiten im Bereich der häuslichen Pflege. Die jüngste technische Entwicklung macht den Einsatz von Robotern im Alltag möglich [SNM98, TBB+99, RBF+00]. Die Akzeptanz von Robotern in der Bevölkerung bestimmt deren Verbreitung und Weiterentwicklung maßgeblich. Ein wichtiges Akzeptanzkriterium werden vor allem die Interaktionsfähigkeiten mit der Umwelt und den Menschen sein.

Um Interaktion zu ermöglichen, muss dem Serviceroboter ein Modell des Umfeldes vorliegen. Interaktion ist nur möglich, wenn dem Roboter die Positionen von Interaktionspartnern bekannt ist. Ein solches Modell kann in dynamischen Umgebungen oder Umgebungen mit dynamischen Komponenten nur durch Sensoren generiert werden. Eine explizite Vorgabe des Umfeldes ist im Allgemeinen nicht möglich.

Die Bildung eines auf Sensordaten basierenden Umgebungsmodells ist keine triviale Aufgabe. Durch rauschende Sensordaten, mangelnde Sensorkalibrierung und Verdeckung von Objekten ergeben sich Unsicherheiten, die durch den Einsatz unterschiedlicher Sensoren mit unterschiedlichen Eigenschaften verringert werden können. Die Vorteile verschiedener Sensormodalitäten können sich ergänzen, indem sie durch geeignete Verfahren fusioniert werden. Diese Arbeit beschreibt einen Lösungsansatz für einen Teilbereich der sensorbasierten Modellierung des Umfeldes. Es wird ein robustes Verfahren vorgestellt, das Personen erkennt und verfolgt. Dieses verfahren basiert auf der Fusionierung multimodaler Sensordaten.

Eine weitere wichtige Vorraussetzung für sinnvolle Interaktion ist neben dem momentanen Umgebungsmodell eine realistische Einschätzung der Umgebung zu späteren Zeitpunkten. Soll eine zeitaufwendige Aktion ausgeführt werden, die zu einer Interaktion mit einem dynamischen Objekt führt, muss die Aktionsplanung das Objekt in einer späteren Konfiguration berücksichtigen.

Vorhersagen über Bewegungen dynamischer Objekte werden auf Grund eines dynamischen Modells getroffen. Dieses Modell kann explizit gegeben oder erlernt sein. Die Vorhersage der Umgebungskonfiguration durch ein erlerntes Modell ist Gegenstand dieser Arbeit.

1.1 Motivation

Das Erkennen und Vorhersagen von Positionen und Bewegungen von Personen ist eine wünschenswerte Fähigkeit für Serviceroboter in realen Umgebungen. Die möglichen Anwendung der Vorhersage von Positionen und Trajektorien von Personen gehen weit über das bloße „nicht im Weg stehen“ hinaus. Im Folgenden sind exemplarisch zwei Beispiele für Aufgaben genannt, die erst durch Kenntnis der Position bestimmter Personen durchführbar werden.

Interaktion zur Effektivitätssteigerung Das folgende Szenario zeigt, wie notwendig die Information über Aufenthaltsorte und Anwesenheit von Personen für Aufgaben sein können, die vom Roboter selbst initiiert sind.

- Der Roboter führt eine Routineaufgabe aus.
- Es kommt zu einem Ausnahmezustand, den der Roboter nicht ohne menschliche Hilfe lösen kann[1].
- Der Roboter fährt zu der Person, die ihm am nächsten ist, und fragt um Hilfe.

Personengebundene Aufgabenformulierung Häufig sind Aufgaben nicht an geometrische Orte, sondern an Personen gebunden. Im Folgenden ist eine solche Aufgabe in Stichworten beschrieben.

- Person A beauftragt den Roboter, ihr ein Buch zu bringen, das Person B vor kurzem gelesen hat.
- Der Roboter begibt sich zu Person B, um nach dem Buch zu fragen.
- Person B übergibt das Buch.
- Person A verlässt während dessen den Ort des Auftrags.
- Der Roboter bringt das Buch zu Person A an den neuen Aufenthaltsort.

Die Frage, wie dieser neue Aufenthaltsort ermittelt oder vorhergesagt werden kann, ist Teil der in dieser Arbeit behandelten Fragen.

[1] Beispielsweise ist ein für die Aufgabe notwendiger Gegenstand ausserhalb der aktorischen Grenzen des Roboters oder eine Information kann nicht sensorisch erfasst werden.

1.2 Zielsetzung

Im Rahmen dieser Arbeit wird die Architektur eines verteilten Systems entwickelt, das Sensordaten verschiedener Modalitäten verarbeitet, um Personen zu verfolgen und deren Trajektorien vorherzusagen. Diese Architektur ist Grundlage eines Systems, das multimodale Sensordaten durch einen Partikelfilter fusioniert und zu einem robusten Personen-Tracking führt. Die ermittelten Trajektorien werden durch Self Organizing Maps gelernt und für eine anschließende Vorhersage generalisiert. Der Schwerpunkt liegt auf dem Umgang mit Unsicherheiten der Sensordaten und der Realisierung eines verteilten Gesamtsystems. Gründe für eine verteilte Architektur sind in Abschnitt 1.4 genannt. Für Tests der einzelnen Algorithmen werden Simulationen für Teile des Systems implementiert.

Die folgenden Abschnitte dieses Kapitels stellen verwandte Arbeiten und die Architektur des hier entwickelten Systems vor. Kapitel 2 beschreibt die Realisierung des multimodalen Trackings. Es wird auf die Fusionierung der multimodalen Sensordaten durch einen Partikelfilter, sowie die Vorverarbeitungsalgorithmen der einzelnen Sensoren eingegangen. Kapitel 3 beschreibt verschiedene Ansätze, die beobachteten Trajektorien zu Bewegungsmustern zu generalisieren, um sie für Vorhersagealgorithmen nutzbar zu machen. Ein Verfahren, die gewonnenen Bewegungsmuster für eine Vorhersage zu nutzen, ist exemplarisch in Kapitel 3.4 beschrieben. Kapitel 4 fasst die Ergebnisse zusammen. Es werden Vorschläge für Verbesserungen und darauf aufbauende Verfahren gemacht.

1.3 Verwandte Arbeiten

In der Literatur sind viele Veröffentlichungen zu finden, deren Gegenstand einzelne Teile des hier vorgestellten Systems sind. Nur wenige Arbeiten beschreiben Tracking, Lernverfahren und die darauf aufbauende Vorhersage von Trajektorien in einem Gesamtsystem. Literaturverweise zu einzelnen Komponenten des Systems werden an geeigneter Stelle in den einzelnen Kapiteln gegeben. Im Folgenden werden zwei Arbeiten vorgestellt, die den momentanen Stand der Technik darstellen.

In [Ben04] ist ein System beschrieben, dass Personen mit Hilfe eines Kalman Filters [WB95] in den Messungen von Laserscannern verfolgt. Die beobachteten Trajektorien zwischen je zwei Rastpositionen[2] werden als Ganzes betrachtet und durch einen Expectation Maximation Algorithmus zu Trajektorienmustern generalisiert. Diese Trajektorienmuster werden durch eine konstante Anzahl aufeinander folgender Positionen repräsentiert, die durch Gaußverteilungen über der Grundebene beschrieben werden. Bei der Vorhersage wird eine konkret beobachtete Teiltrajektorie mit

[2] Als Rastpositionen werden hier Positionen bezeichnet, an denen sich Personen häufig längere Zeit aufhalten.

allen Trajektorienmustern verglichen. Das bezüglich eines definierten Vergleichsmaßes ähnlichste Muster beschreibt die vorhergesagte Trajektorie.

Amalia Foka beschreibt in [Fok05] einen Ansatz, der alternativ zu ganzen Trajektorien sogenannte *Hot Points* (HP) lernt. Diese HPs beschreiben Positionen, die häufig Ziele von Personen sind[3]. Zur Vorhersage wird hier nicht die gesamte momentan beobachtete Teiltrajektorie einer Person betrachtet, sondern nur die aktuelle Position mit Orientierung. Die Achse, die durch die Position und die Bewegungsrichtung beschrieben wird, wird hier als *global direction of obstacle* (GDO) bezeichnet. Anders als in [Ben04] wird hier für jeden HP die Wahrscheinlichkeit berechnet, dass er das Ziel der Person ist. Trajektorien werden nicht direkt vorhergesagt, sondern durch einen Pfadplaner zum wahrscheinlichsten HP berechnet. Die Wahrscheinlichkeit, dass ein HP das Ziel der beobachteten Person ist, ist um so größer, je dichter der HP an der GDO liegt und je kleiner der Abstand zwischen HP und Person ist.

Die geringe Anzahl an Veröffentlichungen zeigt, dass es sich bei der Vorhersage von Trajektorien um ein aktuelles und noch wenig bearbeitetes Forschungsgebiet handelt. Diese Arbeit zeigt eine weitere Möglichkeit auf, Trajektorienvorhersagen zu erstellen.

1.4 Architektur des Systems

Das in dieser Arbeit vorgestellte Gesamtsystem beinhaltet eine Vielzahl verschiedener Komponenten, die in drei Module gegliedert sind. Inhalt der einzelnen Module ist Tracking, Generalisierung und Vorhersage. Die Module setzen sich wiederum aus mehreren funktionalen Einheiten zusammen. Abbildung 1.1 zeigt schematisch den Informationsfluss zwischen den drei Modulen.

Die funktionalen Einheiten, aus denen sich die Module zusammensetzen, können einzeln auf verschiedenen Rechnern, die durch ein Netzwerk verbunden sind, gestartet werden. Die Komunikation der einzelnen Komponenten erfolgt durch die Roblet®-Technologie , die in Abschnitt A.1 beschrieben ist. Gründe für die Kapselung der Softwarekomponenten sind:

- Die Komplexität des Gesamtsystems lässt sich durch Kapselung in abgeschlossene funktionale Einheiten reduzieren,
- Bildverarbeitungsalgorithmen sind rechenintensiv und benötigen daher dedizierte Rechner und
- Vorhandene externe Komponenten wie Kartenserver und Pfadplanung laufen auf unterschiedlichen Rechnern.

[3] HPs können beispielsweise Türen, Arbeitsplätze, Sofas o. ä. sein.

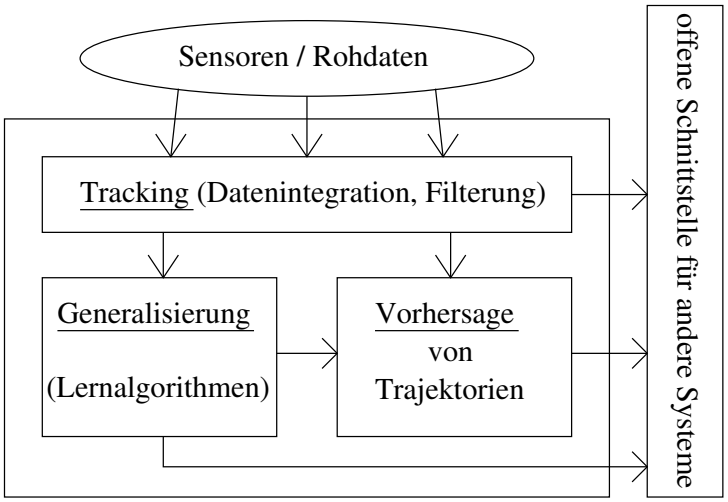


Abbildung 1.1: Das Gesamtsystem (großer Kasten) wird in drei Module (Tracking, Generalisierung und Vorhersage) unterteilt. Die Ergebnisse aller Module werden über eine Schnittstelle für andere Systemen bereitgestellt.

Das Problem der Ansteuerung der verschiedenen Sensoren ist ein weiterer Grund für den Aufbau einer verteilten Architektur. Die verwendeten Laserscanner sind fest auf dem Serviceroboter des AB TAMS montiert, weshalb eine Steuerung des Roboters notwendig ist. Die Steuersoftware der Laserscanner und des Roboters, sowie die Lokalisierung läuft lokal auf dem Roboter. Die verwendete Kamera setzt zur Nutzung bestimmte Hardware voraus, die nicht an jedem Rechner vorhanden ist. In dieser Arbeit wurde eine Framegrabber-Karte eines Rechners im Robotiklabor des AB TAMS genutzt. Die verteilte Architektur ermöglicht so die Integration weiterer Kameras, Laserscanner oder anderer Sensoren in der Umgebung des Roboters. Für die Beschreibung der verwendeten Hardware siehe Anhang A.2.

2 Erkennen und Verfolgen von Personen

Grundlage einer Vorhersage von Trajektorien ist ein robustes Tracking von Personen. Sowohl für die Lernalgorithmen, als auch zur Evaluation der Vorhersagen sind reale Trajektorien erforderlich. Im Rahmen dieser Arbeit wurde ein Modul entwickelt, das robustes Tracking durch mehrere Sensoren ermöglicht. Dieses Trackingmodul ist Gegenstand dieses Kapitels. Besondere Bedeutung kommt der Fusionierung und Filterung der verschiedenen Sensormessungen zu. Dies wurde mit einen Partikelfilter realisiert, die Architektur lässt aber den Austausch des Filterverfahrens zu.

Im Folgenden wird ein Überblick über aktuelle Arbeiten zum Verfolgen von Personen mit unterschiedlichen Sensoren gegeben. Anschließend wird die Architektur des entwickelten Moduls beschrieben, die eine Ergänzung um weitere Sensormodalitäten zulässt. Der darauf folgende Abschnitt beschreibt die mathematischen Grundlagen des Partikelfilters, der für die Fusionierung und Filterung der verschiedenen Eingabedaten verwendet wird. Außerdem wird die Implementierung des Partikelfilters diskutiert. Die Trackingalgorithmen für die zwei zur Verfügung stehenden Sensoren werden in den weiteren Abschnitten beschrieben. Der letzte Abschnitt beschreibt die Ergebnisse des Trackingmoduls. Es wird dabei besonders auf den Nutzen von Multimodalität und Filterung eingegangen.

2.1 Verwandte Arbeiten

Dieser Abschnitt gibt einen Überblick über den momentanen Stand der Technik des Personentrackings. Es werden aktuelle Arbeiten zum Tracking mit Kameras und mit Lasermesssystemen, sowie multimodalem Tracking vorgestellt. Da in diesen drei Bereichen viele verschiedene Ansätze entwickelt wurden, erhebt diese Zusammenfassung keinen Anspruch auf Vollständigkeit. Es werden exemplarisch einige aktuelle Arbeiten vorgestellt, die in der Literatur häufig verwendete Verfahren beschreiben. Für eine Vertiefung in diesem Gebiet sei auf die Literaturangaben der einzelnen Veröffentlichungen verwiesen.

Das Aussehen eines zu verfolgenden Objektes im Kamerabild verändert sich durch unterschiedliche Beleuchtung, Verdeckung oder die relative Lage zur Kamera praktisch permanent. Es gibt verschiedene Ansätze, Tracking in Kamerabildern trotz dieser Schwierigkeiten robust zu gestalten. In [GSRL98, UMS00] wird Verdeckung

durch einen hohen Standort der Kamera weitgehend vermieden. Ein anderer Ansatz den Effekt von Verdeckung zu verringern, ist der Einsatz mehrerer Kameras. In [KHM+00] werden deshalb mehrere Stereokamerasysteme verwendet.

Bei fest montierten Kameras wird zur Erkennung von Bewegungen häufig Backgroundsubstraction verwendet. Grimpson *et al.* [GSRL98] modellieren jedes Pixel als statistischen Prozess durch eine *mixture of Gaussians*, um die Zugehörigkeitwahrscheinlichkeit der Pixel zum Hintergrund möglichst optimal bestimmen zu können. Um bestimmte Vordergrundobjekte zu verfolgen und zu unterscheiden, wird in [SHT+01, Sen02] ein System vorgestellt, das Erscheinungsmodelle der Zielobjekte berücksichtigt. Die Erscheinungsmodelle beinhalten neben einem Farbmodell auch eine Wahrscheinlichkeitsmaske. Dadurch wird jedem Pixel des Farbmodells eine *a priori* Wahrscheinlichkeit zugeordnet, die die Objektzugehörigkeit der Pixel beschreibt. Das in [CRM00] vorgestellte Kameratracking nutzt Farbverteilungen als Objektmodell. Hier wird der Bhattacharyya Koeffizient (siehe Abschnitt 2.6.1) zwischen Zielobjekt und Zielkandidat im Bild durch ein *Mean Shift* Verfahren maximiert. Dieses Verfahren ist detailliert in Kapitel 2.6 beschrieben. Erweiterungen zur Skalierung des getrackten Objektes sind in [Col03] beschrieben.

Tracking von Personen in Entfernungsmessdaten von Laserscannern wird ebenfalls häufig durch Backgroundsubstraction realisiert [ZSI03, FHM02, ZS03]. Die vorgestellten Systeme unterscheiden sich in der Repräsentation und Generierung der Hintergrundmodelle. Diese Herangehensweise ist für mobile Lasermesssysteme nicht sinnvoll, da sich der Abstand zwischen Messsystem und Hintergrund durch die Bewegung des Sensors ändert. In [FZ00] werden mögliche Beine von Personen deshalb nur über die Form und Größe registriert.

Die oben genannten Verfahren zum Tracking in Kamerabildern und Entfernungsmessungen beinhalten meist eine Filterung der Positionsschätzungen. Häufig verwendet werden der in Kapitel 2.3 vorgestellte Partikelfilter oder der Kalmanfilter. Werden mehrere Sensormodalitäten verwendet, so gewinnt die Filterung zusätzlich an Bedeutung. Die Fusionierung und Gewichtung der Sensordaten erfolgt meist ebenfalls über die Filterung. In [FZ00] werden Messdaten eines Laserscanners und einer Kamera durch einen Kalmanfilter fusioniert. Ein anderer Ansatz, der multimodale Trackingdaten durch *anchoring* verknüpft, ist in [KLF+02] beschrieben.

2.2 Architektur des Trackingmoduls

Das Trackingmodul steht am Anfang einer Kette von Verarbeitungsschritten, deren Ziel die Vorhersage von Trajektorien beobachteter Personen ist (siehe Abschnitt 1.4). Als Eingabe dieses Moduls dienen Rohdaten verschiedener Sensoren. Als Ausgabe des Trackingmoduls (und dadurch Eingabe des Generalisierungsmoduls, Kapitel 3) sollen Schätzungen von Anzahl, Positionen und Trajektorien von Personen berechnet werden. Ziele bei der Entwicklung der Architektur des Moduls sind:

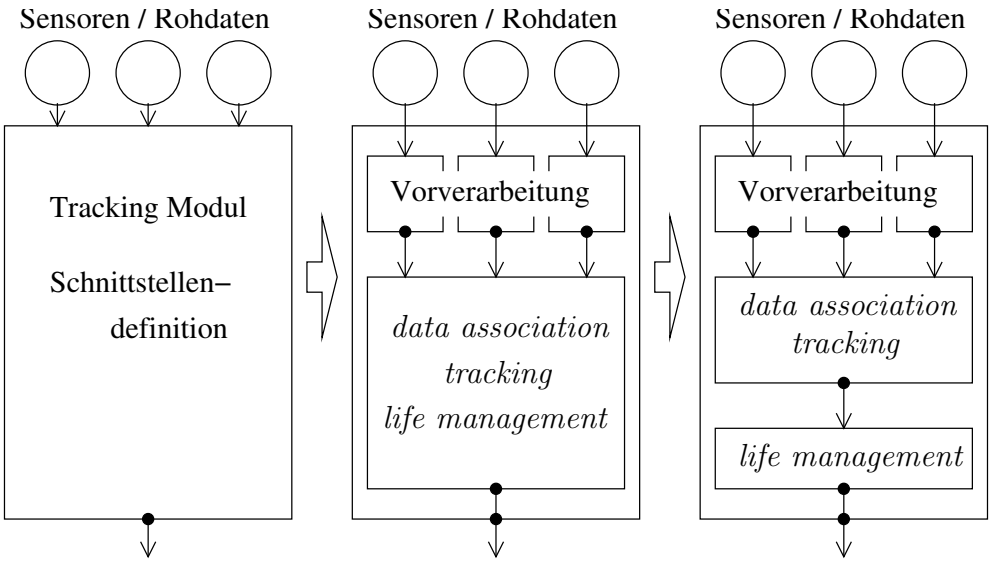


Abbildung 2.1: Entwicklungsprozess der Architektur des Trackingmoduls im Verlauf dieser Arbeit. Bei den schwarzen Punkten ist die Schnittstelle des gesamten Trackingmoduls implementiert.

- klar definierte Schnittstellen, um Algorithmen austauschbar zu machen,
- Offenheit und Erweiterbarkeit bezüglich weiterer Sensoren und
- eine klare und flexible Struktur, um effiziente Entwicklung, Wartung und Tests zu ermöglichen.

Bei der Entwicklung der Architektur wurde ein *top-down* Ansatz verfolgt. Es wurden erst Schnittstellen des Trackingmoduls nach aussen spezifiziert und anschliessend das Modul in funktionale Einheiten unterteilt. Abbildung 2.1 verdeutlicht den Entwicklungsprozss der Architektur. Für die funktionalen Einheiten wurde eine einheitliche Ausgabeschnittstelle definiert, die sowohl von komplexen multimodalen Filtern, sowie von einfachen Algorithmen zur Sensordatenverarbeitung implementiert wird. Dadurch können Ergebnisse einfacher Trackingalgorithmen als Ausgabe des Trackingmoduls oder als Eingabe komplexerer Filter verwendet werden. Eine detaillierte Beschreibung der Schnittstelle ist in A.3 gegeben.

Es wurde ein flexibler Rahmen für beliebige Algorithmen und Sensormodalitäten geschaffen. Für diese Arbeit wurden exemplarisch einige Sensoren und Algorithmen implementiert und in diesen Rahmen integriert. Abbildung 2.2 zeigt die implementierten Komponenten und die Struktur des Moduls. Die einzelnen Komponenten sind in den folgenden Abschnitten detailliert beschrieben.

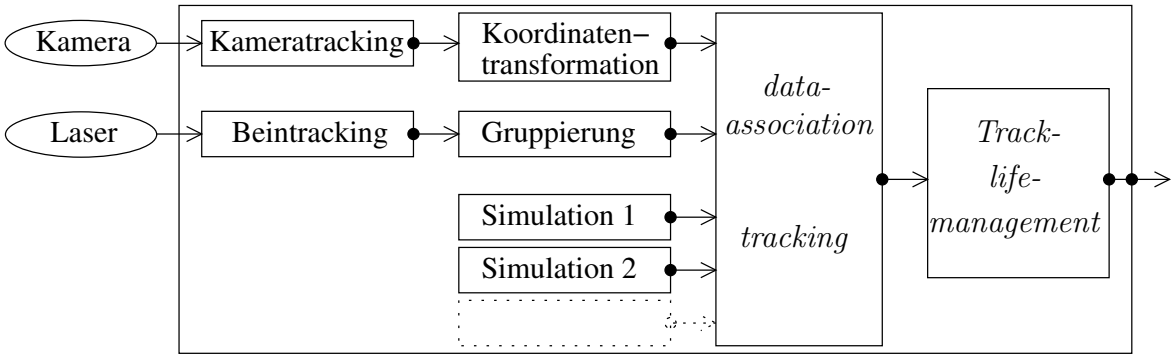


Abbildung 2.2: Implementierte Komponenten des Trackingmoduls. Die schwarzen Punkte symbolisieren eine einheitliche Schnittstelle. Es können beliebig viele weitere simulierte und reale Sensoren hinzugefügt werden.

2.3 Tracking durch Partikelfilter

Durch die oben beschriebene erweiterbare Architektur wird ein Mechanismus notwendig, der Daten verschiedener Sensormodalitäten fusioniert. Dieser Mechanismus berücksichtigt die Unsicherheiten der verschiedenen Sensoren und berechnet eine möglichst optimale Schätzung der Positionen von getrackten Objekten. Die Anzahl der Sensoren ist variabel.

Der nächste Abschnitt beschreibt die konzeptionelle Lösung dieses Problems durch nichtlineare Filterung. Es gibt verschiedene Möglichkeiten, diese generelle Lösung unter bestimmten Annahmen zu realisieren. Durch welche Methode die Filterung realisiert wird, ist für die Systemarchitektur unerheblich. Ebenso wie die einzelnen Sensoren ist dieser Teil austauschbar. Es werden zwei Lösungen vorgestellt, die von bestimmten Annahmen ausgehen, die die allgemeine Anwendbarkeit einschränken. Anschliessend wird die hier verwendete Approximation der generellen Lösung durch Partikelfilter beschrieben. Die mathematischen Beschreibungen dieses Kapitels entsprechen weitgehend denen in [RSG04] und [BSLK01].

2.3.1 Konzeptuelle Lösung durch Nichtlineare Filterung

Nichtlineare Filterung wird seit mehr als 35 Jahren diskutiert [Jaz70]. Sie bezieht sich auf das Problem, den Zustand eines dynamischen Systems zu aufeinander folgenden Zeitpunkten zu schätzen. Grundlage der Schätzung ist eine Folge von ungenauen Messungen. Es wird ein zeit-diskreter Ansatz verfolgt, bei dem Systemgleichungen genutzt werden, um den Systemzustand von einem Zeitpunkt in den des Nachfolgezeitpunktes zu überführen. Der Zustandsvektor beinhaltet alle wichtigen Informationen um das System zu beschreiben. In dem hier behandelten Trackingproblem

beinhaltet der Zustandsvektor $x_k \in \mathbb{R}^4$ Position und Bewegung[1] des beobachteten Objektes in der Grundebene. Messungen sind nur zu diskreten Zeitpunkten vorhanden. Der Messungsvektor beinhaltet verrauschte Beobachtungen über das System.

Um Schlüsse über das System ziehen zu können, sind zwei Modelle nötig: Das Systemmodel beschreibt, wie sich der Zustandsvektor über die Zeit ändert, und das Messmodell beschreibt die Beziehung zwischen Zustand und Beobachtung. Diese Modelle werden als in probabilistischer Form gegeben angenommen.

Da alle Messungen mit Unsicherheit behaftet sind, können Zustände nur durch Wahrscheinlichkeitsverteilungen über dem Zustandsraum repräsentiert werden. Im Bayes Ansatz zur dynamischen Zustandsschätzung wird eine Wahrscheinlichkeitsverteilung (*probability density function*, *pdf*) des Zustandsvektors gebildet, die alle vorhandenen Informationen berücksichtigt. Wenn diese *pdf* sämtliche vorhergehenden Messungen beinhaltet und als vollständig angenommen wird, kann eine *pdf* zum Zeitpunkt k aus der *pdf* zum Zeitpunkt $k-1$ rekursiv gebildet werden. Es ist nicht notwendig, die vollständige Sequenz der vorhergehenden Messungen zu speichern, da sie implizit in der vorherigen *pdf* vorhanden ist.

Um das Problem zu beschreiben, wird der Zustandsvektor $x_k \in \mathbb{R}^{n_x}$ und der Messvektor $z_k \in \mathbb{R}^{n_z}$ definiert, wobei n_x und n_z die Dimension des Zustandsvektors und des Messvektors sind. Das oben genannte Systemmodell ist eine Funktion $x_k = f_{k-1}(x_{k-1}, v_{k-1})$, die den Zustand x_k abhängig von dem vorherigen Zustand x_{k-1} und einem Unsicherheitsfaktor v_{k-1} beschreibt. Die Beziehung zwischen Systemzustand und Messvektor wird im Messmodell durch $z_k = h_k(x_k, w_k)$ beschrieben, wobei w_k die Unsicherheit des Messmodells und der Messung selbst beschreibt. Die Unsicherheiten v_k und w_k werden als stochastisch unabhängig und gegeben angenommen. Die Funktionen f_k und h_k sind möglicherweise nichtlinear und werden als bekannt vorrausgesetzt. Die Folge von allen Messungen wird mit $Z_k = \{z_i | i = 1, \ldots, i = k\}$ bezeichnet.

Die *pdf* zum Zeitpunkt k unter Berücksichtigung aller Messungen Z_k wird nach der Bayes Regel (in Gleichung 2.2 angewendet) wie folgt gebildet.

$$
\begin{aligned}
p(x_k|Z_k) &= p(x_k|z_k, Z_{k-1}) && (2.1)\\
&= \frac{p(z_k|x_k, Z_{k-1})p(x_k|Z_{k-1})}{p(z_k|Z_{k-1})} && (2.2)\\
&= \frac{p(z_k|x_k)p(x_k|Z_{k-1})}{p(z_k|Z_{k-1})} && (2.3)
\end{aligned}
$$

Gleichung 2.3 gilt, da angenommen wird, dass die Messung z_k von den vorhergehenden Messungen Z_{k-1} unabhängig ist. $p(z_k|x_k)$ folgt direkt aus dem Messmodell. Das Systemmodell mit Unsicherheit v_{k-1} wird hier durch die Chapman-Kolmogorov

[1] Als Repräsentation der Bewegung wird dx und dy und nicht Winkel und Betrag gewählt.

Gleichung [Pap84, Seite 531] (2.4) berücksichtigt, und die Normalisierungskonstante (2.5) berücksichtigt das Messmodell mit Unsicherheit w_k.

$$p(x_k|Z_{k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|Z_{k-1})\,dx_{k-1} \tag{2.4}$$

$$p(z_k|Z_{k-1}) = \int p(z_k|x_k)p(x_k|Z_{k-1})\,dx_k \tag{2.5}$$

Die Gleichung 2.4 beschreibt die *pdf* zum Zeitpunkt k, ohne eine Messung zu diesem Zeitpunkt zu berücksichtigen. Sie beschreibt also eine Vorhersage des Systemzustandes und wird deshalb als *prior pdf* bezeichnet. Wird sie wie in (2.3) mit $p(z_k|x_k)$ multipliziert und normiert, folgt daraus die *posterior pdf* , die sämtliche Information beinhaltet.

Wie oben beschrieben muss die Folge Z_{k-1} der Messungen nicht explizit gespeichert werden, wenn $p(x_{k-1}|Z_{k-1})$ als gegeben angenommen wird. In Gleichung 2.4 wird die vorherige *pdf* rekursiv verwendet. Wird eine initiale *pdf* $p(x_0|Z_0)$ mit der leeren Messfolge Z_0 angenommen, stellt dieser Ansatz theoretisch eine optimale Lösung dar. Praktisch lässt sich dieser Ansatz so nicht realisieren, da eine nicht lineare, kontinuierliche *pdf* $p(x_k|Z_k)$ im Allgemeinen nicht vollständig repräsentiert werden kann. Die Wahrscheinlichkeitsverteilung des Zustandsvektors muss eingeschränkt oder approximiert werden.

2.3.2 Realisierung der Nichtlinearen Filterung

Die in Abschnitt 2.3.1 beschriebene konzeptionelle Lösung kann nicht ohne Einschränkungen realisiert werden. Es existieren verschiedene Verfahren, um den Zustandsraum oder die Wahrscheinlichkeitsverteilung des Zustandes so einzuschränken oder zu approximieren, dass dieser Ansatz praktisch durchführbar wird. Welches Verfahren hier zur Anwendung kommt, ist für das Gesamtsystem dieser Arbeit prinzipiell unerheblich. Wie zu Beginn dieses Kapitels beschrieben, ist das Trackingmodul in sich modular aufgebaut. Die einzelnen Algorithmen sind dadurch erweiterbar und austauschbar. Im Folgenden werden zwei Verfahren vorgestellt, die den Zustandsraum bzw. die *pdf* einschränken. Danach wird das für diese Arbeit implementierte Verfahren zur Approximierung des kontinuierlichen Zustandsraumes beschrieben.

Wird der Zustandsraum diskretisiert und auf eine endliche Menge von Zuständen eingeschränkt, lässt sich der oben beschriebene Bayes Ansatz anwenden. In [MN90] wurde gezeigt, wie der Zustandsraum eines Trackingproblems auf eine endliche Anzahl diskreter Zustände eingeschränkt werden kann. Ist der Zustandsraum zum Zeitpunkt k durch eine Menge mit N diskreten Zuständen $\{x_k^i|i = 0, \ldots, i = N\}$ definiert, so kann die *pdf* durch $p(x_k) = \sum_{i=1}^{N} P(x_k = x_k^i)\delta(x_k - x_k^i)$ beschrieben werden,

wobei δ die Dirac Funktion ist. Die bedingte Wahrscheinlichkeit der Zustände wird durch

$$\omega_{k|m}^{i} \stackrel{\triangle}{=} P(x_k = x_k^i | Z_m) \tag{2.6}$$

definiert. Wird (2.6) in (2.4) und (2.5) eingesetzt, folgt

$$p(x_k|Z_{k-1}) = \sum_{i=1}^{N} \omega_{k|k-1}^{i} \delta(x_k - x_k^i) \tag{2.7}$$

$$p(z_k|Z_{k-1}) = \sum_{i=1}^{N} p(z_k|x_k^i)\omega_{k|k-1}^{i} \tag{2.8}$$

wobei

$$\omega_{k|k-1}^{i} = \sum_{j=1}^{N} p(x_k^i|x_{k-1}^j)\omega_{k-1|k-1}^{j} \tag{2.9}$$

gilt. Werden Übergangswahrscheinlichkeit $p(x_k|x_{k-1})$ und Messmodell $p(z_k|x_k^i)$ wie in Kapitel 2.3.1 als bekannt vorrausgesetzt, ist durch Einsetzen von (2.7) und (2.8) in (2.3) eine optimale und berechenbare Lösung gegeben.

Bei Anwendung des Kalman Filters wird der Zustandsraum nicht diskretisiert. Um eine Filterung zu ermöglichen, wird davon ausgegangen, dass die Funktionen f_k und h_k des System- und Messmodells linear sind. Der Kalman Filter ist desshalb keine Anwendung der *nichtlinearen* Filterung und wird hier nur wegen der Vollständigkeit vorgestellt. Die Unsicherheitsfaktoren v_{k-1} und w_k entsprechen Gaußverteilungen mit Mittelwert null und bekannten Kovarianzmatrizen R_{k-1} und Q_k. Unter diesen Annahmen kann gezeigt werden, dass aus einer gaußverteilten *pdf* $p(x_{k-1}|Z_{k-1})$ mit Mittelwert μ_{t-1} und Kovarianz Σ_{t-1} wieder eine gaußverteilte *pdf* $p(x_k|Z_k)$ folgen muss [WB95]. Systemmodell und Messmodell werden durch Matrixmultiplikationen wie folgt beschrieben.

$$\begin{aligned} x_k &= A_{k-1}x_{k-1} + v_{k-1} \\ z_k &= C_k x_k + w_k \end{aligned}$$

Der Kalman Filter berechnet in einem ersten Schritt eine Schätzung des Zustandes $\mathcal{N}(\overline{\mu}_k, \overline{\Sigma}_k)$ ohne Berücksichtigung der Messung z_k. In einem zweiten Schritt wird daraus die Annahme über den tatsächlichen aktuellen Zustand unter Einbeziehung der Messung z_k getroffen. In Algorithmus 1 sind die Gleichungen des Kalman Filters beschrieben. Die Variable K_k in Zeile 5 wird *Kalman Gain* genannt und gewichtet den Einfluss der Messung auf die neue Positionsschätzung. Für eine mathematische Herleitung des Kalman Filters siehe [TBF05]. Es existieren zusätzlich Erweiterungen und Varianten des Kalman Filters wie Extended Kalman Filter, Unscented Kalman Filter und Information Filter. Der Leser sei ebenfalls auf [TBF05] verwiesen.

Algorithm 1 Kalman Filter

Require: Normalverteilung der Zustandsschätzung $\mathcal{N}(\mu_{k-1}, \Sigma_{k-1})$ zum Zeitpunkt $k-1$

Ensure: Aktualisierte Normalverteilung der Zustandsschätzung $\mathcal{N}(\mu_k, \Sigma_k)$

1: Schätzung des Zustandes ohne Berücksichtigung der Messung z_k
2: $\overline{\mu}_k = A_{k-1}\mu_{k-1}$
3: $\overline{\Sigma}_k = A_{k-1}\Sigma_{k-1}A_{k-1}^T + R_k$
4: Annahme des Zustandes unter Einbeziehung der Messung z_k
5: $K_k = \overline{\Sigma}_k C_k^T \left(C_k \overline{\Sigma_k} C_k^T + Q_k\right)^{-1}$
6: $\mu_k = \overline{\mu}_k + K_k \left(z_k - C_k \overline{\mu}_k\right)$
7: $\Sigma_k = \left(I - K_k C_k\right) \overline{\Sigma}_k$

In [Ben81] wurde gezeigt, dass es neben dem Kalmanfilter und der Diskretisierung des Zustandsraumes weitere Einschränkungen gibt, die konkrete Lösungen von nichtlinearen Wahrscheinlichkeitsfunktionen zulassen.

Partikelfilter approximieren die kontinuierliche *pdf* durch eine Menge gewichteter Sampel. Es ist dabei nicht nötig, den Zustandsraum oder die Art der Verteilungsfunktion einzuschränken. Situationen, in denen die Verteilungsfunktion des Zustandsvektors mehrere lokale Maxima hat, können durch Kalman Filter nicht behandelt werden. Abb. 2.3 zeigt, dass solche Situationen durchaus auftreten können. Die in A.3 beschriebene Schnittstelle lässt jedoch nur *eine* Schätzung der Position eines getrackten Objektes zu. Um mehrere wahrscheinliche Positionen an höhere Ebenen weiterzugeben[2], könnte die Schnittstelle erweitert werden. Da hier der Zustandsraum $\mathbb{R}^4$ nicht diskretisiert werden soll, kommt ein Partikelfilter zur Anwendung. Im Folgenden werden die zum Verständnis des Partikelfilters wichtigen mathematischen Grundlagen erläutert.

Die Basis des Partikelfilters ist die *Monte Carlo Integration*. Ziel ist es, eine multidimensionale Funktion $g(x)$ mit $x \in \mathbb{R}^{n_x}$ durch $g(x) = f(x)\pi(x)$ zu faktorisieren, so dass $\pi(x)$ als Wahrscheinlichkeitsverteilung interpretiert werden kann[3]. Wird nun eine $\pi(x)$ verteilte Menge Sampels $\{x^i | i = 1, \ldots, i = N\}$ mit $N \gg 1$ erzeugt, so kann das Integral von $g(x)$ wie in 2.10 numerisch approximiert werden.

$$\int g(x)\, dx \approx \frac{1}{N} \sum_{i=1}^{N} f(x^i) \tag{2.10}$$

Ist nur eine $\pi(x)$ ähnliche[4] Funktion $\tau(x)$ bekannt, so kann die numerische Integration durch *Importance Sampling* erfolgen. Die Sampelmenge $\{x^i | i = 1, \ldots N\}$ ist nun

[2] Denkbar ist auch die Übertragung der ganzen *pdf* des Objektes an höhere Ebenen.
[3] Es muss also $\pi(x) \geq 0$ und $\int \pi(x)\, dx = 1$ gelten.
[4] Ähnlichkeit bezieht sich hier auf die Gleichheit des Trägers, d.h. $\forall_x : \pi(x) > 0 \Rightarrow \tau(x) > 0$.

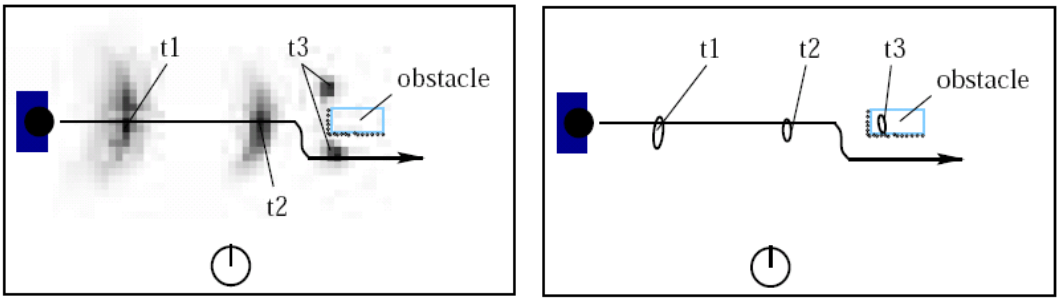


Abbildung 2.3: Die Grafik zeigt eine Person, die einem Hindernis ausweicht. Sie wird von einem Partikelfilter (links) und einem Kalmanfilter (rechts) getrackt. Die nichtlineare *pdf* der verfolgten Person wird durch einen Kalman Filter nicht korrekt behandelt. (Quelle: [SBFC03, Seite 22])

$\tau(x)$ verteilt und es wird (2.10) neu formuliert:

$$\int g(x)\,dx = \int f(x)\frac{\pi(x)}{\tau(x)}\tau(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x^i)\omega(x^i) \tag{2.11}$$

mit

$$\omega(x^i) = \frac{\pi(x^i)}{\tau(x^i)}\ . \tag{2.12}$$

Beim *Sequencial Importance Sampling* wird nun die *pdf* aus 2.1 durch eine Menge diskreter Sample mit zugehörigen Gewichten approximiert. Es wird $\{x_k^i, \omega_k^i\}$ definiert, wobei $\{x_k^i | i = 1, \ldots, N\}$ eine Menge diskreter Sampel aus $\mathbb{R}^{x_n}$ und $\{\omega_k^i | i = 1, \ldots, N\}$ die dazugehörigen Gewichte mit $\sum_{i=1}^{N} \omega_k^i = 1$ bezeichnet. Mit $f(x) = 1$ und Gleichung 2.11 kann nun die gesuchte Verteilungsfunktion durch

$$p(x_k|Z_k) \approx \frac{1}{N}\sum_{i=1}^{N} \omega_k^i \delta(x_k - x_k^i) \tag{2.13}$$

approximiert werden. Dabei ergibt sich ω_k^i wie folgt aus Gleichung 2.12

$$\omega_k^i \triangleq \omega(x_k^i) = \frac{p(x_k^i|Z_k)}{\tau(x_k^i|Z_k)}, \tag{2.14}$$

wobei $p(x_k^i|Z_k)$ der tatsächlichen Dichteverteilung $\pi(x^i)$ aus Gleichung 2.12 entspricht und $\tau(x_k^i|Z_k)$ der ähnlichen Dichteverteilung.

Bei jeder neuen Messung soll die *pdf* durch eine Menge gewichteter Sampels möglichst gut approximiert werden. Wird jedes Sampel bezüglich einer Verteilungsdichte $\tau(x_k^i|x_{k-1}^i, Z_k)$ verschoben, so ergibt sich $\tau(x_k^i|Z_k) = \tau(x_{k-1}^i|Z_{k-1})\tau(x_k^i|x_{k-1}^i, Z_k)$. Die

tatsächliche Dichteverteilung $p(x_k^i|Z_k)$ ergibt sich aus der Anwendung von Gleichung 2.3 auf ein einzelnes Sampel:

$$\begin{aligned} p(x_k^i|Z_k) &= \frac{p(z_k|x_k^i, Z_{k-1})p(x_k^i|Z_{k-1})}{p(z_k|Z_{k-1})} \\ &= \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)p(x_{k-1}^i|Z_{k-1})}{p(z_k|Z_{k-1})} \ . \end{aligned} \tag{2.15}$$

Wird nun (2.15) in (2.14) eingesetzt, folgt

$$\begin{aligned} \omega_k^i &= \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)p(x_{k-1}^i|Z_{k-1})}{p(z_k|Z_{k-1})\tau(x_{k-1}^i|Z_{k-1})\tau(x_k^i|x_{k-1}^i, Z_k)} \\ &= \omega_{k-1}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{p(z_k|Z_{k-1})\tau(x_k^i|x_{k-1}^i, Z_k)} \ . \end{aligned} \tag{2.16}$$

Der Normierungsfaktor $p(z_k|Z_{k-1})$ bleibt über alle ω_k^i konstant. Werden die Gewichte der Sampel anschließend normiert und vereinfachend $p(x_k^i|x_{k-1}^i) \approx \tau(x_k^i|x_{k-1}^i, Z_k)$ angenommen, so gilt

$$\omega_k^i \propto \omega_{k-1}^i p(z_k|x_k^i) \tag{2.17}$$

Wird die *pdf* des Zustandsvektors eines getrackten Objektes wie hier beschrieben durch Gleichung 2.13 approximiert, so kann die rekursive Lösung der nichtlinearen Filterung in Form von (2.16) bzw. (2.17) erreicht werden. Der Ablauf des Partikelfilter Algorithmus wird im folgenden Abschnitt beschrieben.

2.3.3 Partikelfilter Algorithmus

Die mathematische Herleitung der Approximation der nichtlinearen Filterung durch Partikelfilter wurde in Abschnitt 2.3.2 gegeben. Bei der Umsetzung in einen ausführbaren Algorithmus wird zwischen *predict* und *update* Phase unterschieden. Die beiden Phasen werden bei jeder neuen Messung nacheinander ausgeführt. In der *predict* Phase werden die Sampel nach der Verteilung $\tau(x_k^i|x_{k-1}^i, Z_k)$, die sich aus dem Systemmodel ergibt, verschoben. In der *update* Phase werden die Gewichtungen aller Sampel wie in Gleichung 2.17 neu berechnet. Der Ablauf ist in Algorithmus 2 beschrieben.

Durch die Schätzung $\tau(x_k^i|x_{k-1}^i, Z_k)$ kommt es bei jedem Durchlauf des Algorithmus zu einer Verbreiterung der *pdf* . Das führt dazu, dass die Gewichte der meisten Sampel sehr klein werden. Die *pdf* wird effektiv nur noch durch wenige Sampel repräsentiert. Um nicht die meiste Rechenzeit für nichtrelevante Sampel zu verwenden, wird ein *resampling* Schritt eingeführt.

Aufgabe des Resamplings ist es, die Sampel und deren Gewichte so umzuverteilen, dass die approximierte *pdf* möglichst nicht verändert wird. Zwei häufig genutzte Verfahren sind *Select with Replacement* [Rek03] und *Resampling* nach [LCL01]. Das hier

Algorithm 2 Sequential Importance Sampling (SIS)

Require: Menge $\{x_{k-1}^i, \omega_{k-1}^i\}_{i=1}^N$ gewichteter Sampel
Ensure: Aktualisierte Menge $\{x_k^i, \omega_k^i\}_{i=1}^N$

1: **for** $i := 1$ to N **do**
2: setze x_k^i nach $\tau(x_k^i | x_{k-1}^i, Z_k)$
3: berechne ω_k^i nach Gleichung 2.17
4: **end for**
5: berechne $t = \sum_{i=1}^N \omega_k^i$
6: **for** $i := 1$ to N **do**
7: normiere die Gewichte mit $\omega_k^i = \omega_k^i t^{-1}$
8: **end for**

verwendete Verfahren mit Komplexität $O(N)$ ist *Systematic Resampling* [RSG04]. Bei diesem Verfahren werden die Sampel so umverteilt, dass die Gewichtung aller Sampel gleich $1/N$ ist. Sampel mit hoher Wahrscheinlichkeit werden dabei vervielfacht, während Sampel mit niedriger Wahrscheinlichkeit möglicherweise gelöscht werden. Der Ablauf ist in Algorithmus 3 formuliert.

Algorithm 3 Systematic Resampling

Require: Menge $\{x_k^i, \omega_k^i\}_{i=1}^N$ mit gewichteten Sampel
Ensure: neue Menge $\{x_k^{j*}, \omega_k^{j*}\}_{j=1}^N$

1: $c_1 = \omega_k^1$
2: **for** $i := 2$ to N **do**
3: $c_i = c_{i-1} + \omega_k^i$
4: **end for**
5: $i = 1$
6: **for** $j := 1$ to N **do**
7: $u = \frac{1}{N}(j-1)$
8: **while** $u > c_i$ **do**
9: $i = i + 1$
10: **end while**
11: $k_k^{j*} = x_k^i$
12: $\omega_k^{j*} = \frac{1}{N}$
13: **end for**

Auch wenn es ein Ziel des Resamplings ist, die approximierte *pdf* möglichst nicht zu verändern, geht bei jedem Resamplingschritt Information verloren. Resampling ist nur dann nötig, wenn die Sampelmenge wie oben beschrieben degeneriert. In [LCL01] werden zwei Möglichkeiten beschrieben, wie die Menge der Sampel, deren Gewichtung $\omega_k^i \to 0$ geht, berechnet werden kann: Der *coefficient of variation* cv_k^2

und die in dieser Arbeit verwendete *effective sample size* N_{eff}.

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}(\omega_k^i)^2} \tag{2.18}$$

Wenn alle Sampel gleich gewichtet sind ($\omega_k^i = \frac{1}{N}$, für $i = 1 \dots N$), so ist $N_{eff} = N$. Existiert genau ein Sampel mit $\omega_k^i = 1$, so gilt $N_{eff} = 1$. Die *effective sample size* ist also ein Maß, wie viele Sampel effektiv etwas zur *pdf* beitragen. Das Resampling wird nur ausgeführt, wenn N_{eff} einen bestimmten Schwellwert[5] N_{thr} unterschreitet.

Es wurden nun alle Teile des Partikel Filter Algorithmus beschrieben. Die Zusammensetzung der einzelnen Algorithmen ist in Algorithmus 4 beschrieben.

Algorithm 4 Particle Filter

Require: Menge $\{x_{k-1}^i, \omega_{k-1}^i\}_{i=1}^N$ mit und Messung z_k
Ensure: neue Menge $\{x_k^i, \omega_k^i\}_{i=1}^N$

1: Filterung durch SIS wie in Algorithmus 2
2: Berechnung von N_{eff} wie in Formel 2.18
3: **if** $N_{eff} < N_{thr}$ **then**
4: Resampling wie in Algorithmus 3
5: **end if**

2.4 Erkennen und Tracken von mehreren Objekten

In Kapitel 2.3 wurde gezeigt, wie der Zustand eines Objekts trotz verrauschter Messungen optimal geschätzt werden kann. Es wurde ein nichtlinearer Filter angewendet, um eine Sequenz von Messungen zu verarbeiten. Sollen mehrere Objekte gleichzeitig verfolgt werden, so wird für jedes Objekt eine *pdf* instanziiert und ein Filter angewendet. Im Folgenden wird eine *pdf* mit Filter als *Track* bezeichnet.

Es besteht beim Tracking mehrerer Objekte das Problem, dass M Messungen z_k^i mit $i = 1 \dots M$ den N verschiedenen Tracks t^j mit $j = 1 \dots N$ zugeordnet werden müssen. Diese Zuordnung wird als *data association* bezeichnet. Im Folgenden wird die Zuordnung der Messung z_k^i zu einem Track t^j mit a^{ij} bezeichnet. Die Wahrscheinlichkeit $P(a^{ij}) = 1$ bedeutet, dass die Messung z_k^i durch das von t^j getrackte Objekt verursacht wurde. Das Problem, die Wahrscheinlichkeiten aller Zuordnungen zu bestimmen, ist generell np-vollständig [HCP00]. Es existieren Verfahren, die nicht alle Zuordnungsmöglichkeiten untersuchen. Solchen Verfahren schränken den Suchraum durch unterschiedliche Heuristiken ein. Zwei dieser Verfahren sind der *Gibbs-Sampler*[HCP00] und der *JPDA-Filter*[SBFC03].

[5] Üblicherweise wird der Schwellwert in Prozent von N angegeben.

In dieser Arbeit kommt zur Lösung des Zuordnungsproblems ein neu entwickelter einfacher *nearest-neighbour* Ansatz zur Anwendung. Es wird dabei angenommen, dass jedes getrackte Objekt maximal eine Messung verursacht. Der Algorithmus berechnet keine echten Wahrscheinlichkeiten der Zuordnungen sondern nur Annahmen darüber, ob eine Zuordnung zutrifft. Die Zuordnungswahrscheinlichkeiten ergeben sich aus Gleichung 2.19.

$$P(a^{ij}) = \begin{cases} 1 & \text{wenn der Ursprung von } z_k^i \text{ in } t^j \text{ vermutet wird} \\ 0 & \text{sonst} \end{cases} \tag{2.19}$$

Es wird iterativ die Zuordnung mit dem geringsten euklidischen Abstand zwischen Messung und Track gesucht. Ist dieser Abstand geringer als ein Schwellwert d_{thr}, so wird diese Zuordnung angenommen und Messung und Track nicht weiter betrachtet. Wenn der Abstand größer als der Schwellwert ist, wird diese Zuordnung nicht angenommen. Die restlichen Messungen werden nicht zugeordnet und der Algorithmus terminiert. Der Algorithmus hat die Komplexität $O(MN)$. In Algorithmus 5 ist der Ablauf in Pseudocode formuliert.

Algorithm 5 Nearest Neighbour Ansatz zum Zuornungsproblem

Require: Menge $z_k = \{z_k^i | i = 1 \dots M\}$ und Menge $t = \{t^j | j = 1 \dots N\}$
Ensure: Menge der angenommenen Zuordnungen $a \subset \{a^{ij} | i = 1 \dots M, j = 1 \dots N\}$
wobei jedes $i \in \underline{M}$ und $j \in \underline{N}$ in maximal einer Zuordnung enthalten ist

initialisiere Zuordnungen: $a = \emptyset$
while $z_k \neq \emptyset$ and $t \neq \emptyset$ **do**
  finde Messung $z_k^i \in z_k$ und Track $t^j \in t$ mit geringstem Abstand zueinander
  if $|z_k^i - t^j| < d_{thr}$ **then**
    füge Zuordnung a^{ij} der Lösungsmenge zu: $a = a \cup \{a^{ij}\}$
    lösche z_k^i aus den nicht zugeordneten Messungen: $z_k = z_k \setminus z_k^i$
    lösche t^j aus den nicht zugeordneten Tracks: $t = t \setminus t^j$
  else
    terminiere mit a
  end if
end while

Aus den Messungen z_k^i ist nicht direkt ersichtlich, wieviele Objekte tatsächlich vorhanden sind. Es muss neben dem Zuordnungsproblem auch berücksichtigt werden, wann neue Tracks initialisiert und wann andere gelöscht werden. Dieser Vorgang wird als *track life management* bezeichnet. Es werden dabei drei Zustände der Tracks definiert:

- *Vorläufige* Tracks werden mit Messungen initialisiert, die keinem Track zugeordnet werden konnten.

- Diese vorläufigen Tracks können den Zustand *bestätigt* erlangen, wenn ihnen über einen bestimmten Zeitraum Messungen zugeordnet wurden.
- Wenn einem Track über einen bestimmten Zeitraum keine Messung zugeordnet werden konnte, wird er als *gelöscht* markiert. Gelöschte Tracks werden bei weiteren Zuordnungen nicht mehr betrachtet.

Es wurden bisher die drei wichtigsten Teile des Trackingsystems vorgestellt. Mit *track life management*, einer Lösung zum *data association* Problem und der *nichtlinearen Filterung* können multimodale Eingabedaten unterschiedlicher Qualität verarbeitet werden. Die beiden in dieser Arbeit verwendeten Sensoren werden in den folgenden Abschnitten vorgestellt.

2.5 Tracking mit festen Lasermesssystemen

In Arbeiten, in denen Lasermesssysteme als einzige Sensoren verwendet werden [NZS+04, ZSI03, FHM02], wird das Tracking häufig in drei Teile unterteilt. Durch *Backgroundsubstraction* werden Messungen des Hintergrundes herausgefiltert. Anschliessend werden die Messungen, die vom selben Objekt stammen, *gruppiert*. Zum Schluss werden die Zustandsvektoren der beobachteten Objekte gefiltert und *getrackt*. Dieser Vorgang beinhaltet die in Kapitel 2.3 und 2.4 beschriebenen Schritte.

Durch die hier realisierte offene Architekur (siehe Kapitel 2.2) wird das Tracking für mehrere Sensoren gemeinsam genutzt. Es soll hier also nicht Bestandteil der Algorithmen sein, die die einzelnen Sensordaten verarbeiten. Die Algorithmen der einzelnen Sensoren dienen eher der Vorverarbeitung und sollen aus den Sensordaten nutzbare Messvektoren z_k^i berechnen.

Der nächste Abschnitt beschreibt die Generierung eines Hintergrundmodells und dessen Anwendung auf die gemessenen Daten. Anschliessend wird beschrieben, wie die Vordergrundmesspunkte zu Objekten gruppiert werden.

2.5.1 Backgroundsubstraction

Übliche Verfahren zum Tracking mit Laserscannern, generieren ein Hintergundmodell durch Bildung eines Histogrammes über die Entfernungsmessungen für jeden Winkel. Die gemessenen Entfernungen eines Winkels werden über einen bestimmten Zeitraum in ein Histogramm eingetragen. Es wird angenommen, dass das Maximum des Histogrammes durch den Hintergrund bedingt ist. Das Hintergrundmodell ordnet jedem Messwinkel $i\delta$, wobei δ die Winkelauflösung des Lasermesssystems ist, eine Entfernung h_i zu, die als Entfernung des Hintergrundes angenommen wird. Die Modellierung kann auch über Verteilungsfunktionen der Entfernungen erfolgen [FHM02].

Entfernungsmessungen eines Winkels, die kleiner als der Hintergrundwert für diesen Winkel sind, werden als Vordergund klassifiziert und in weiteren Berechnungen berücksichtigt. Der Hintergrund wird so für folgende Berechnungen ausgeblendet. Solche Systeme funktionieren im einfachen Fall zweischrittig: Der erste Schritt generiert ein Hintergrundmodell; der zweite Schritt löscht Messpunkte, die dem Hintergrund zugeordnet werden. Dieses zweischrittige Verfahren hat folgende Nachteile:

- Vor Nutzung des Systems muss eine zeitaufwendige Initialisierungsphase durchlaufen werden.
- Bleibt während der Initialisierung ein Vordergrundobjekt unbewegt, wird es als Hintergrund klassifiziert.
- Verändert sich der Hintergrund zur Laufzeit des Systems, wird er fortwährend als Vordergrund klassifiziert.

Die hier vorgestellte Arbeit verwendet ein neues Verfahren mit gleitendem Hintergrund. Die Entfernung des Hintergrundes $h_i(t)$ zum Zeitpunkt t wird mit jeder Messung $m_i(t)$ rekursiv durch Formel 2.20 gebildet.

$$h_i(t) = h_i(t-1) + \begin{cases} \epsilon_1 & \text{für } h_i(t-1) < m_i(t) \\ (-\epsilon_2) & \text{sonst} \end{cases} \tag{2.20}$$

Durch Größe der unterschiedlichen Inkremente ϵ_1 und ϵ_2 wird gesteuert, wie schnell sich das Hintergrundmodell an das momentane Umfeld angleicht. Diese Faktoren bestimmen einerseits, wie lange ein unbewegtes Vordergrundobjekt als solches erkannt wird und andererseits, wie lange es dauert, bis ein neues Hintergrundobjekt[6] korrekt klassifiziert wird. Für die in Abschnitt 2.7 durchgeführten Testreihen wurden $\epsilon_1 = 10\text{cm}$ und $\epsilon_2 = 3\text{cm}$ verwendet.

2.5.2 Gruppierung der Vordergrundmesspunkte

Durch das in Kapitel 2.5.1 beschriebene Verfahren werden Messpunkte herausgefiltert, die nicht durch Vordergrundobjekte verursacht wurden. Die übrigen Vordergrundmesspunkte sollen so zu Gruppen zusammengefasst werden, dass genau die Punkte der selben Gruppe zugehörig sind, deren Ursprung am selben Objekt liegt.

Für die Gruppierung werden Vordergrundmesspunkte in Reihenfolge der Messwinkel betrachtet. Der erste Messpunkt wird einer offenen Gruppe zugeordnet. Nun werden iterativ alle folgenden Vordergrundmessungen bearbeitet. Ist der Abstand zwischen dem Schwerpunkt der aktuellen Gruppe und dem betrachteten Messpunkt kleiner

[6] Langfristig abgestellte Kisten oder Schränke können den Hintergrund verändern.

als ein Schwellwert, so wird der Punkt der Gruppe zugefügt; sonst wird die Gruppe geschlossen und eine neue Gruppe geöffnet, die diesen Punkt enthällt. Der folgende Algorithmus beschreibt dieses Verfahren in Pseudocode.

Algorithm 6 Gruppierung der Vordergrundmesspunkte

Require: Menge $v = \{v_i | i = 1 \dots N\}$ mit sortierten Vordergrundpunkten
Ensure: Menge von Gruppen g

```
1: initialisiere Gruppen: g = ∅
2: initialisiere offene Gruppe: open = {v_1}
3: for i := 2 to N do
4:    berechne Mitte der Gruppe: c = center(open)
5:    if distance(c, v_i) < d_thr then
6:       füge Messpunkt der Gruppe zu: open = open ∪ {v_i}
7:    else
8:       speichere offene Gruppe: g = g ∪ {open}
9:       initialisiere neue offene Gruppe: open = {v_i}
10:   end if
11: end for
```

Der maximale Abstand d_{thr} zwischen Gruppe und Scannpunkt gibt den Radius der gesuchten Gruppen an. Der zu verwendende Wert folgt aus der Anwendung: Die hier interessierenden Objekte sind die Beine von Menschen, die etwa 20 cm horizontal über dem Boden gescannt werden. Ein Radius von 15 cm hat sich in der Praxis als bewährt. Die Funktion $center(open)$ berechnet den Schwerpunkt der in der Gruppe enthaltenen Messpunkte.

Die so berechneten Gruppen können aus unterschiedlich vielen Messpunkten bestehen. Gruppen mit sehr wenigen Messpunkten sollen ignoriert werden, da sie sich möglicherweise aus Fehlern der Backgroundsubstraction ergeben. Es können aber nicht alle Gruppen mit wenigen Messpunkten ignoriert werden, da Beine, deren Abstand zum Laserscanner groß ist, nur von wenigen Strahlen erfasst werden. Die erwartete Menge der Messpunkte einer Gruppe ist durch den Abstand d der Gruppe zum Laserscanner, die Winkelauflösung δ und die Größe des gescannten Objektes[7] bestimmt. Aus Abbildung 2.4 wird ersichtlich, dass für die Anzahl s der Messpunkte einer Gruppe $s = \alpha/\delta$ gelten muss, wobei α der Öffnungswinkel der randnahen Laserstrahlen ist, die das Objekt treffen.

Für den Öffnungswinkel gilt $\alpha = 2 \arctan(r/d)$. Da die Winkelauflösung der Lasermesssysteme konstant ist, lässt sich die Anzahl der Messpunkte wie in Gleichung 2.21 direkt aus d berechnen.

$$s = \frac{2 \arctan(r/d)}{\delta} \tag{2.21}$$

[7]Die Größe wird weiterhin mit $r = 15$ cm konstant angenommen.

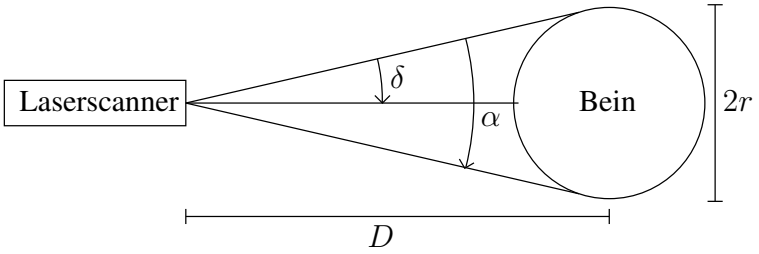


Abbildung 2.4: Die Grafik verdeutlicht den Zusammenhang zwischen Objektabstand und Größe sowie den Winkeln und der Anzahl der Messpunkte pro Objekt.

Weicht die tatsächliche Menge der Messpunkte zu stark von der erwarteten ab, so wird die Gruppe nicht als Bein interpretiert.

Die hier verwendeten Lasermesssysteme (siehe A.2) werden mit einer Winkelauflösung von $\delta = 0.5$ Grad betrieben. Durch die hohe Messgenauigkeit werden Beine von Personen einzeln erfasst. Bei der Zuordnung von Beinen zueinander kann großer Aufwand betrieben werden. In [ZS03] wird beispielsweise ein Modell verwendet, in dem bei einer gehenden Person ein Bein fest steht, wärend sich das andere in Laufrichtung bewegt. Da die Trackingalgorithmen nicht Hauptgegenstand dieser Arbeit sein sollen, kommt hier ein einfacheres Verfahren zur Anwendung.

Um das Problem der Zuordnung von bis zu zwei[8] Beinen zu je einer Person zu lösen, wird der Algorithmus 5 erweitert. Die Zuordnungen werden zu a^{hij} erweitert und ordnen je einem Track t^j zwei Messungen z_k^h und z_k^i zu. Wenn einem Track nur eine Messung zugeordnet wird, so ist der Index der zweiten Messung 0. Der Algorithmus ist in 7 formuliert. Er stellt keine optimale Lösung dar, reicht aber für die in Abschnitt 2.7 durchgeführten Untersuchungen aus.

2.6 Kameratracking

Es gibt viele Ansätze, Objekte in Kamerabildern zu verfolgen. Ein Überblick über aktuelle Arbeiten ist in [Rus04] gegeben. In der hier vorgestellten Arbeit wird der *Mean Shift Algorithmus* verwendet, der in [CRM00] vorgestellt wurde. Zu verfolgende Objekte werden dabei durch ihre statistische Farbverteilung charakterisiert. Die Zielobjekte (bzw. deren Farbmodell) werden manuell vorgegeben.

Die manuelle Initialisierung stellt eine Einschränkung der Flexibilität des Gesamtsystems dar. Es können mit diesem Verfahren nur Personen verfolgt werden, deren Farbmodel bekannt ist. Das Verfahren wurde dennoch ausgewählt, da es echtzeitfähig ist und eine Übertragung auf bewegte Kameras mit geringem Aufwand

[8] Möglichrweise können wegen Verdeckung nicht jeder Person zwei Beine zugeordnet werden.

Algorithm 7 Nearest Neighbor Ansatz zum Zuornungsproblem mit zwei Messungen pro Track

Require: Menge $z_k = \{z_k^i | i = 1 \dots M\}$ und Menge $t = \{t^j | j = 1 \dots N\}$

Ensure: Menge der angenommenen Zuordnungen $a \subset \{a^{hij} | h = 1 \dots M, i = 0 \dots N, j = 1 \dots N\}$ wobei jedes $h \in \underline{M}$, $i \in \underline{N}$ und $j \in \underline{N}$ in maximal einer Zuordnung enthalten ist

initialisiere Zuordnungen: $a = \emptyset$
while $z_k \neq \emptyset$ and $t \neq \emptyset$ **do**
  finde Messung $z_k^h \in z_k$ und Track $t^j \in t$ mit geringstem Abstand zueinander
  if $|z_k^h - t^j| < d_{thr}$ **then**
    lösche z_k^h aus den nicht zugeordneten Messungen: $z_k = z_k \setminus z_k^h$
    lösche t^j aus den nicht zugeordneten Tracks: $t = t \setminus t^j$
    finde Messung $z_k^i \in z_k$ mit geringstem Abstand zu Track t^j
    if $|z_k^i - t^j| < d_{thr}$ **then**
      lösche z_k^i aus den nicht zugeordneten Messungen: $z_k = z_k \setminus z_k^i$
      füge Zuordnung a^{hij} der Rückgabemenge zu: $z = z \cup \{a^{hij}\}$
    else
      füge Zuordnung a^{h0j} der Rückgabemenge zu: $a = a \cup \{a^{h0j}\}$
    end if
  else
    terminiere mit a
  end if
end while

möglich ist. Eine Anpassung des Trackingalgorithmus ist nicht notwendig, nur die externen Kameraparameter für die Transformation von Bild- zu Weltkoordinaten müssten für jede Bewegung der Kamera angepasst werden.

Der folgende Abschnitt beschreibt das Mean Shift Tracking in Kamerakoordinaten. Die mathematischen Beschreibungen sind weitgehend aus [CRM00] entnommen. Die Kamerakalibrierung und die Transformation in Weltkoordinaten wird im darauffolgenden Abschnitt beschrieben.

2.6.1 Mean Shift Tracking Algorithmus

Der hier beschriebene Algorithmus findet die Position eines zu verfolgenden Objektes in Bildkoordinaten. Es wird eine initiale Position vorgegeben, die mittels *Mean Shift* Algorithmus iterativ verschoben wird, bis das Objekt gefunden ist. Im Folgenden wird die angenommene Position des Objektes im Bild als *Zielkandidat* bezeichnet.

Zu Beginn wird die Repräsentation des Zielobjektes und des Zielkandidaten beschrieben. Um zu bestimmen, ob das Objekt gefunden wurde, wird anschließend ein Vergleichsmaß zwischen Zielobjekt und Zielkandidat im Bild vorgestellt. Im folgenden Abschnitt wird das Mean Shift Verfahren erst allgemein beschrieben und dann auf das Tracking Problem dieser Arbeit angewendet.

Die Repräsentation des Zielobjektes erfolgt über eine Verteilung der Farben über einem diskretisierten Farbraum. Diese Verteilung entspricht einem normierten Farbhistogramm der Pixel des Zielobjektes. Die Koordinaten der Pixel des Zielobjektes werden mit $X^* = \{x_i^*\}_{i=1\ldots n^*}$ bezeichnet, wobei der Koordinatenursprung im Schwerpunkt des Zielobjektes liegt. Der Farbraum wird für das Histogramm in m Farben diskretisiert. Die Funktion $b : \mathbb{R}^2 \rightarrow \{1 \ldots m\}$ ordnet jedem Pixel den Index der zugehörigen diskreten Farbe zu. Um die Wahrscheinlichkeit der Farbe u zu berechnen, sollen Pixel, die am Rand des Objektes liegen, weniger gewichtet werden. Die Gewichtung wird über eine monoton abfallende Kernelfunktion $K : \mathbb{R}^2 \rightarrow \mathbb{R}$ realisiert. Die Wahrscheinlichkeit der Farbe u wird anhand der Gleichung 2.22 berechnet.

$$\hat{q}_u = C \sum_{x \in X^*} K\left(\frac{x}{h^*}\right) \delta\big(b(x) - u\big), \tag{2.22}$$

wobei die Skalierung h^* und der Normierungsfaktor C durch

$$\begin{aligned} h^* &= max\{|x|^2 \ | x \in X^*\} \\ C &= \frac{1}{\sum_{x \in X^*} K\left(\frac{x}{h^*}\right)} \end{aligned}$$

gegeben sind.

Die Farbverteilung des Zielkandidaten im Bild wird analog berechnet. Die Menge $X = \{x_i\}_{i=1\ldots n}$ bezeichnet die Pixelkoordinaten des Bildes. Der Zielkandidat hat Größe h und Mittelpunkt y. Hat die Kernelfunktion K einen endlichen Träger, so müssen nur die Pixel mit $K(\frac{y-x}{h}) \neq 0$ für die Berechnung der Farbverteilung betrachtet werden. Es wird $X_h^y = \{x | x \in X, K(\frac{y-x}{h}) \neq 0\}$ definiert. Die Farbverteilung ergibt sich damit zu Gleichung 2.23.

$$\hat{p}_u(y) = C_h \sum_{x \in X_h^y} K\left(\frac{y-x}{h}\right) \delta\big(b(x) - u\big). \tag{2.23}$$

Unter oben gemachten Angaben ist C_h nur von der Größe h des Zielkandidaten abhängig und kann im Voraus mit Gleichung 2.24 berechnet werden.

$$C_h = \frac{1}{\sum_{x \in X_h^y} K\left(\frac{y-x}{h}\right)} \tag{2.24}$$

Als Vergleichsmaß zwischen dem zu verfolgenden Objekt und dem Zielkandidaen wird der *Bhattacharyya* Koeffizient verwendet. Die allgemeine Form des Koeffizienten ist in [Kai67] definiert. Eine diskrete Formulierung ist in Gleichung 2.25 gegeben.

$$\rho\big(\hat{p}(y), \hat{q}\big) = \sum_{u=1}^{m} \sqrt{\hat{p}_u(y)\hat{q}_u} \tag{2.25}$$

Stimmen die Modelle von Ziel und Kandidat exakt überein, so gilt $\rho\big(\hat{p}(y), \hat{q}\big) = 1$. Es gilt also den Wert des Bhattacharyya Koeffizient zu maximieren.

Die Sample Mean Shift Methode ist ein effizientes Verfahren, das Maximum einer Dichteverteilung zu finden. Es wird dabei *ad hoc* die mittlere Verschiebung zwischen Testkandidat und Maximum geschätzt. Dadurch kann mit wenigen Iterationen das Maximum der Verteilung gefunden werden.

Um das Verfahren allgemein zu beschreiben, wird die Schätzung einer d-dimensionalen Kerneldichte (*multivariate kernel density estimate*, *mkde*) einer Sampelmenge $\{x_i | x_i \in \mathbb{R}^d\}_{i=1\ldots n}$ definiert. Die *mkde* mit Kernel $K(x)$ und Größe h ist in (2.26) definiert.

$$\hat{f}_K(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{2.26}$$

Als Kernel wird dabei (2.27) verwendet, es sind aber auch andere Kernel möglich.

$$K_n(x) = (2\pi)^{-d/2} \exp\left(-\frac{1}{2}|x|^2\right) \tag{2.27}$$

Die Funktion $k[x, \infty[\rightarrow \mathbb{R}$ mit $K(x) = k(|x|^2)$ wird als *Profil* des Kernels K bezeichnet. Mit dieser Notation wird Gleichung 2.26 in Gleichung 2.28 umgeformt. Um die

Richtung des Maximums der Dichte zu bestimmen, wird der Gradient von $\hat{f}_K(x)$ gebildet.

$$\hat{f}_K(x) = \frac{1}{nh^d}\sum_{i=1}^{n} k\left(\left|\frac{x-x_i}{h}\right|^2\right) \tag{2.28}$$

$$\begin{aligned}\nabla\hat{f}_K(x) &= \frac{2}{nh^{d+2}}\sum_{i=1}^{n}(x-x_i)k'\left(\left|\frac{x-x_i}{h}\right|^2\right)\\ &= \frac{2}{nh^{d+2}}\sum_{i=1}^{n}-k'\left(\left|\frac{x-x_i}{h}\right|^2\right)(x_i-x)\\ &= \frac{2}{nh^{d+2}}\left[\sum_{i=1}^{n}-k'\left(\left|\frac{x-x_i}{h}\right|^2\right)\right]\left[\frac{\sum_{i=1}^{n}-k'\left(\left|\frac{x-x_i}{h}\right|^2\right)x_i}{\sum_{i=1}^{n}-k'\left(\left|\frac{x-x_i}{h}\right|^2\right)}-x\right]\end{aligned} \tag{2.29}$$

Wird ein Kernel $G_k(x)$ wie folgt definiert:

$$\begin{aligned}G_k(x) &= Cg(|x|^2)\\ g(x) &= -k'(x)\end{aligned} \tag{2.30}$$

so beinhaltet (2.29) die Dichteschätzung mit Kernel G und den *Mean Shift Vektor* $M_{h,G}(x)$, der die geschätzte Differenz zwischen der Position x des Kernels und dem Maximum der *mkde* bezeichnet:

$$\hat{f}_G(x) = \frac{C}{nh^d}\sum_{i=1}^{n} g\left(\left|\frac{x-x_i}{h}\right|^2\right) \tag{2.31}$$

$$M_{h,G}(x) = \frac{\sum_{i=1}^{n} x_i g\left(\left|\frac{x-x_i}{h}\right|^2\right)}{\sum_{i=1}^{n} g\left(\left|\frac{x-x_i}{h}\right|^2\right)} - x \tag{2.32}$$

Um das Maximum der Dichteverteilung zu finden, wird rekursiv die Position x des Kernel G um $M_{h,G}(x)$ verschoben. Es werden die Folgen $[y_j]_{j=1,2,\ldots}$ und $[\hat{f}_K(y_j)]_{j=1,2\ldots}$ definiert, wobei y_1 die initiale Position des Kernels ist.

$$y_{j+1} = y_j + M_{h,G}(y_j) \tag{2.33}$$

In [CRM00] wurde gezeigt, dass die Folgen $[y_j]_{j=1,2,\ldots}$ und $[\hat{f}_K(y_j)]_{j=1,2\ldots}$ konvergieren, wenn der Kernel K ein konvexes, monoton fallendes Profil hat und Kernel G wie in (2.30) definiert wird. Dies gilt auch, wenn jedem Sampel x_i der betrachteten Menge eine positive Gewichtung ω_i zugewiesen wird.

Für die Anwendung der Sample Mean Shift Methode auf das hier beschriebene Kameratracking ist die oben genannte Konvergenzeigenschaft wesentlich. Die den

Berechnungen zu Grunde liegenden Sampel sind hier Pixelkoordinaten. Da die Pixelkoordinaten im Bild gleichverteilt sind, kann eine sinnvolle Dichteverteilung nur über die Gewichtung der einzelnen Sampel gebildet werden. Die Gewichtung erfolgt gemäß dem verwendeten Vergleichskriterium (hier der Bhattacharyya Koeffizient, Gleichung 2.25). Das Vergleichskriterium beschreibt, auf alle Pixelkoordinaten angewendet, die zu betrachtende Dichteverteilung über $\mathbb{R}^2$. In [CRM00] wird der Bhattacharyya Koeffizient wie folgt approximiert dargestellt

$$\rho\big(\hat{p}(y_{j+1}), \hat{q}\big) \approx \frac{1}{2}\sum_{u=1}^{m}\sqrt{\hat{p}_u(y_j)\hat{q}_u} + \frac{1}{2}\sum_{u=1}^{m}\hat{p}_u(y_{j+1})\sqrt{\frac{\hat{q}_u}{\hat{p}_u(y_j)}}\,, \tag{2.34}$$

wobei y_{j+1} die nachfolgende Position bezüglich der Folge 2.33 ist. Für diese Approximation wird angenommen, dass sich die Bhattacharyya Koeffizienten zwischen aufeinander folgenden y_j nicht drastisch unterscheiden. Wird nun (2.22) und (2.23) mit dem Kernelprofil formuliert und in (2.34) eingesetzt, so folgt

$$\rho\big(\hat{p}(y_{j+1}), \hat{q}\big) \approx \frac{1}{2}\sum_{u=1}^{m}\sqrt{\hat{p}_u(y_j)\hat{q}_u} + \frac{C_h}{2}\sum_{i=1}^{n_h}\omega_i k\left(\left|\frac{y_{j+1}-x_i}{h}\right|^2\right), \tag{2.35}$$

mit

$$\omega_i = \sum_{u=1}^{m}\delta\big(b(x_i)-u\big)\sqrt{\frac{\hat{q}_u}{\hat{p}_u(y_j)}} \text{ ist.} \tag{2.36}$$

Der erste Term der Gleichung 2.35 ist unabhängig von y_{j+1}. Die durch diese Gleichung beschriebene Ähnlichkeit der Farbmodelle des Zielobjektes und des Kandidaten hängt nur von dem zweiten Teil der Summe ab. Dieser Teil beschreibt die durch ω_i gewichtete Dichte an der Stelle y_{j+1} mit Kernel K. Da die Ähnlichkeit maximiert werden soll, ist diese gewichtete Dichte Grundlage der Anwendung der Mean Shift Prozedur. Aus (2.32), (2.33) und (2.35) ergibt sich die Folgeposition des Zielkandidaten y_j mit Kernel K.

$$y_{j+1} = \frac{\sum_{i=1}^{n} x_i\omega_i g\left(\left|\frac{y_j-x_i}{h}\right|^2\right)}{\sum_{i=1}^{n}\omega_i g\left(\left|\frac{x_j-x_i}{h}\right|^2\right)} \tag{2.37}$$

Nachdem alle für ein Kameratracking auf Basis des Mean Shift Verfahrens nötigen theoretischen Grundlagen beschrieben sind, wird der Ablauf der Anwendung, die dieses Verfahren realisiert, in Algorithmus 8 beschrieben. Durch diesen Algorithmus sind die Kamerakoordinaten des gesuchten Objekts bekannt. Wenn mehrere Objekte verfolgt werden, wird dieser Algorithmus für das Farbmodell jedes einzelnen Objektes instanziiert. Für das hier vorgestellte System wurde die Implementierung des Kameratracking einer Bibliothek des AB TAMS genutzt.

Für die weitere Verarbeitung der Positionsschätzungen müssen diese noch in Weltkoordinaten transformiert werden. Der nächste Abschnitt beschreibt die Kamerakalibrierung nach [Tsa85], sowie die Transformation der Koordinaten.

Algorithm 8 Mean Shift Maximierung des Bhattacharyya Koeffizienten

Require: Ein Bild und die geschätzte Position des Zielobjektes y_0 im letzten Bild
Ensure: Geschätzte Position des Zielobjektes y_1 im aktuellen Bild

1: initialisiere Position des Zielkandidaten: y_0
2: **repeat**
3: berechne Bhattacharyya Koeffizient $\rho\big(\hat{p}(y_0), \hat{q}\big)$ mit Gleichung 2.25
4: berechne neue Position des Zielkandidaten y_1 mit Gleichung 2.37
5: berechne Farbverteilung des neuen Kandidaten $\hat{p}_u(y_1)$ mit Gleichung 2.23
6: **while** $\rho\big(\hat{p}(y_1), \hat{q}\big) < \rho\big(\hat{p}(y_0), \hat{q}\big)$ **do**
7: setze y_1 ein um die Hälfte zurück: $y_1 \leftarrow \frac{1}{2}(y_0 + y_1)$
8: **end while**
9: **until** $|y_1 - y_0| < \epsilon$

2.6.2 Kamerakalibrierung nach Tsai

Um Informationen über Position und Trajektorie verschiedener Sensoren fusionieren zu können, müssen diese in einem einheitlichen Koordinatensystem vorliegen. Die Positionen in Bildkoordinaten, die durch das in Kapitel 2.6.1 beschriebene Verfahren berechnet werden, müssen somit in ein globales, gemeinames Koordinatensystem transformiert werden. Um diese Transformation zu realisieren, ist es erforderlich, die Rückprojektion von Bildkoordinaten in Weltkoordinaten mathematisch zu beschrieben.

Der folgende Abschnitt beschreibt das Lochkameramodell, dass es ermöglicht, solche Projektionen zu berechnen. Es wird die Notwendigkeit einer Kalibrierung erläutert und einige wichtige Begriffe eingeführt. Anschließend wird mit der zweistufigen Kalibrierung nach [Tsa85] die für diese Arbeit verwendete Kalibrierung vorgestellt. Am Ende dieses Abschnitts wird beschrieben, wie das Verfahren bei den durchgeführten Experimenten angewendet wurde.

Das Lochkameramodell entspricht nicht dem realen Aufbau einer handelsüblichen Kamera. Mit entsprechenden Erweiterungen ist es damit jedoch möglich, die Projektion einer realen Kamera mit Linsenverzeichnung ausreichend zu beschreiben. Das verzeichnungsfreie Lochkameramodell ist in Abb. 2.5 dargestellt.

Das *Kamerakoordinatensystem* ist dabei wie folgt definiert: Der Ursprung liegt im optischen Zentrum O_z der Kamera, die z-Achse liegt auf der optischen Achse, und die x- und y-Achsen liegen parallel zu den entsprechenden Achsen der Bildebene. Die verzeichnungsfreie Projektion eines dreidimensionalen Punktes in Kamerakoordina-

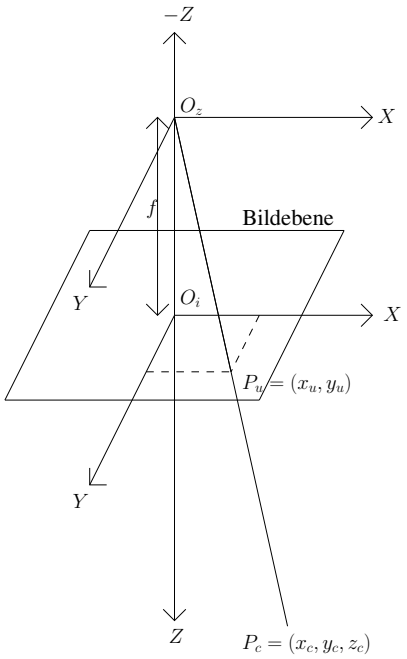


Abbildung 2.5: Das Modell einer Lochkamera.

ten auf die Bildebene erfolgt mit Gleichung 2.38.

$$\begin{aligned} x_u &= f\frac{x_c}{z_c} \\ y_u &= f\frac{y_c}{z_c} \end{aligned} \tag{2.38}$$

$P_c = (x_c, y_c, z_c)^T$ bezeichnet den Punkt in Kamerakoordinaten und $P_u = (x_u, y_u)^T$ den projezierten Punkt in der Bildebene.

Ist die Linse der realen Kamera nicht verzeichnungsfrei, so ist der Punkt auf der Bildebene gegenüber dem durch (2.38) berechneten verschoben. Die verzeichneten Koordinaten werden wie in (2.39) berechnet.

$$\begin{aligned} x_d + D_x(P_u) &= x_u \\ y_d + D_y(P_u) &= y_u, \end{aligned} \tag{2.39}$$

wobei $P_d = (x_d, y_d)^T$ der verzeichnete Bildpunkt und $D_x(P_u)$ und $D_y(P_u)$ die Differenzen zwischen den Projektionen mit und ohne Verzeichnung in x- und y- Richtung bezeichnen. Die Verzeichnung lässt sich in radiale und tangentiale Anteile aufspalten. Abbildung 2.6 verdeutlicht diese Unterscheidung.

Um die Koordinaten der Bildebene in Pixel eines Bildes umzurechnen, muss die Lage des CCD-Chips in der xy-Ebene und die Größe der einzelnen Pixel bekannt

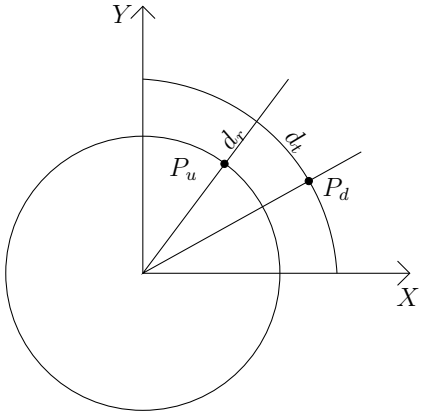


Abbildung 2.6: Der Punkt P_u wird in radialer (d_r) und tangentialer (d_t) Richtung zu P_d verzeichnet.

sein. Die Umrechnung entspricht damit einer linearen Funktion der diskretisierten Koordinaten. Zusammen mit der Brennweite f und den Koeffizienten der Verzeichnung[9] $D_x(p)$ und $D_y(p)$ ist der Kameraaufbau und damit die Transformation von Kamerakoordinaten in Bildpixel bestimmt. Die bisher genannten Parameter werden als *intrinsische* Kameraparameter bezeichnet.

Für die Projektion von Weltkoordinaten P_w in Kamerakoordinaten muss die Lage der Kamera (genauer: des optischen Zentrums) bekannt sein. Die Lage ist durch eine Rotation R und eine Translation $T = (t_x, t_y, t_z)^T$ bestimmt. Die Transformation erfolgt durch $P_c = RP_w + T$, wobei die Rotationsmatrix durch die drei Eulerwinkel (*yaw* Θ, *pitch* Φ, *roll* Ψ) mit Gleichung 2.40 eindeutig bestimmt ist. Die Rotationsmatrix R und der Translationsvektor T werden zusammen als *extrinsische* Kameraparameter bezeichnet.

$$R = \begin{pmatrix} \cos\Psi\cos\Theta & \sin\Psi\cos\Theta & -\sin\Theta \\ -\sin\Psi\cos\Phi + \cos\Psi\sin\Theta\cos\Phi & \cos\Psi\cos\Phi + \sin\Theta\sin\Phi\sin\Psi & \cos\Theta\sin\Phi \\ \sin\Psi\sin\Phi + \cos\Psi\sin\Theta\cos\Phi & -\cos\Psi\sin\Phi + \sin\Theta\sin\Phi\cos\Psi & \cos\Theta\cos\Phi \end{pmatrix} \tag{2.40}$$

Die Kalibrierung beschreibt den Vorgang, aus einer Menge von bekannten Punkten in Weltkoordinaten mit korrespondierenden Pixelkoordinaten die intrinsischen und extrinsischen Parameter zu bestimmen.

Die Kalibrierung nach [Tsa85] geht von einer vernachlässigbar kleinen tangentialen Verzeichnung aus. Die Verzeichnung aus (2.39) wird dadurch nur durch den

[9] Die Art und Anzahl der Koeffizienten ist durch die Art der Modellierung der Verzeichnung bestimmt.

radialen Anteil und mit Gleichung 2.41 beschrieben.

$$\begin{aligned} D_x &= x_d(k_1 r^2 + k_2 r^2 + \cdots) \\ D_y &= y_d(k_1 r^2 + k_2 r^2 + \cdots), \end{aligned} \tag{2.41}$$

wobei nur der erste Koeffizient k_1 bestimmt wird. Durch Annahme einer rein radialen Verzeichnung kann das *Radial Alignment Constraint* (RAC) formuliert werden. Es besagt, dass die Vektoren $\vec{O_i P_d}$ und $\vec{P_z P_c}$ koplanar sind, wobei P_z jedem beliebigen Punkt der optischen Achse entspricht.

Das Problem der Kamerakalibrierung ist, dass die zu kalibrierenden Parameter in nichtlinearem Zusammenhang miteinander stehen. Die Optimierung einer Lösung erfordert eine aufwendige, nichtlineare Suche über dem Parameterraum. In [Tsa85] wurde gezeigt, dass unter Annahme des RAC die zu kalibrierenden Parameter teilweise entkoppelt werden können. Die Kalibrierung erfolgt dadurch in zwei Stufen [TL86]:

1. Stufe:
   - Unter Annahme des RAC werden 5 Parameter[10] in linearer Zeit berechnet: R, t_x, t_y
2. Stufe:
   - Durch die Projektionsgleichung wird aus den bekannten Parametern $\tilde{f}$ und $\tilde{t_z}$ geschätzt
   - Durch nichtlineare Optimierung wird aus T, t_x, t_y, $\tilde{f}$, $\tilde{t_z}$ iterativ f, t_z und k_1 berechnet

Der Suchraum der im letzten Schritt durchgeführten nichtlinearen Optimierung beschränkt sich auf nur drei Parameter, und es liegt eine Schätzung zu Grunde, die die Komplexität weiter reduziert.

Für die Anwendung der Kalibrierung wurde das Verfahren nicht neu implementiert. Die Carnegie Mellon School of Computer Science[11] stellt bereits eine Implementierung zur Verfügung, die nur geringe Anpassungen in der Formatierung der Ausgabe erfordert.

Zur Generierung der für die Kalibrierung nötigen Menge von korrespondierenden Punkten wurden auf dem Boden des Robotiklabors des AB TAMS 30 Marken angebracht. Mit Hilfe der Selbstlokalisierung des zur Verfügung stehenden Serviceroboters wurden diese Marken vermessen. Die Pixelkoordinaten der Marken im aufgenommenen Bild wurden manuell bestimmt. Das Kamerabild mit den Marken ist in Abb. 2.7 zu sehen.

[10] Die $3x3$ Rotationsmatrix ist durch drei Winkel Θ, Φ, Ψ eindeutig bestimmt
[11] http://www.cs.cmu.edu

Abbildung 2.7: Die weißen Punkte auf dem Boden des Robotiklabors des AB TAMS dienen als Marken für die Kamerakalibrierung.

2.7 Experimentelle Ergebnisse

Der folgende Abschnitt beschreibt die vorgenommenen Tests der Trackingkomponenten und deren Ergebnisse. Ein Problem bei der Auswertung der Trackingalgorithmen stellt der Mangel an *ground truth* dar: Im Rahmen dieser Arbeit kann die tatsächlich gegangene Trajektorie einer Person nicht genauer bestimmt werden, als durch den hier vorgestellten Trackingalgorithmus. Die Berechnung des mittleren Fehlers der beobachteten Trajektorie ist somit unmöglich.

Der nächste Abschnitt beschreibt eine Simulation der einzelnen Trackingalgorithmen. Auf dieser Grundlage wird der in Kapitel 2.3 beschriebene Partikelfilter evaluiert. Es wird gezeigt, dass durch die Fusionierung multimodaler Sensordaten eine Verbesserung der Ergebnise erzielt werden kann. Anschließend werden Ergebnisse der Algorithmen vorgestellt, die reale Sensordaten verarbeiten.

2.7.1 Simulation

Die Sensorhardware war aus verschiedenen Gründen während der Entwicklung dieser Arbeit nicht permanent verfügbar. Um ein reibungsloses Entwickeln und Testen der auf das Tracking aufbauenden Algorithmen zu ermöglichen, wurde eine Simulation implementiert. Ein weiterer Grund für eine Simulation sind bessere Testmöglichkeiten, da Trajektorien automatisiert generiert werden können.

Während der Simulation läuft ein Thread, der die simulierte Trajektorie berechnet und zur Verfügung stellt. Die simulierten Sensoralgorithmen lesen die exakte Position in variablen Zeitintervallen aus. Die Positionen werden mit Varianz σ^2 normalverteilt verrauscht und über die Schnittstelle der realen Tracker als Messung weitergegeben.

Durch den Partikelfilter wird die optimale Lösung der nichtlinearen Filterung nach dem Bayes Ansatz (siehe Kapitel 2.3.1) approximiert. In die Positionsschätzung gehen alle vergangenen Messungen ein. In dem Systemmodell des Filters ist zusätzlich die Entwicklung des Zustandsvektors (hier die Position) modelliert. Entspricht die Trajektorie dem Systemmodell (es wird eine geradlinige Bewegung angenommen), so verbessert sich die Standardabweichung der gefilterten Trajektorie um bis zu 40%. Abbildung 2.8 zeigt die Filterung simulierter Messungen mit verschiedenen Standardabweichungen. Ändert sich die Richtung der Trajektorie zufällig, kann durch eine Filterung keine Verbesserung der Schätzung erzielt werden.

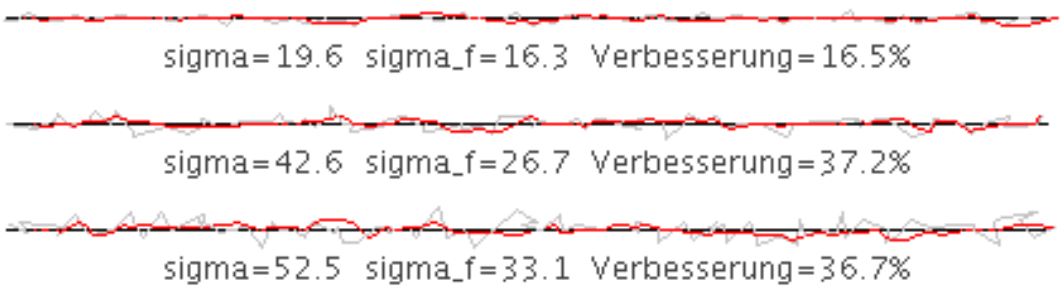


Abbildung 2.8: Test Trajektorien (schwarz), simulierte Messungen (grau, Standardabweichung: *sigma*) und gefilterte Trajektorien (rot, Standardabweichung: $sigma_f$) und Verbesserung in Prozent.

Werden n Messungen entsprechend ihrer Standardabweichung $\sigma_i^2, i = 1 \ldots n$ gewichtet und summiert, so berechnet sich die Standardabweichung der Summe σ_s^2 mit Gleichung 2.42.

$$\frac{1}{\sigma_s^2} = \sum_{i=1}^{n} \frac{1}{\sigma_i^2} \tag{2.42}$$

Auch hier kann eine Verbesserung der Positionsschätzung durch Anwendung des Partikelfilters erzielt werden. Abbildung 2.9 zeigt Ergebnisse der Filterung mehrerer simulierter Messungen mit verschiedenen Standardabweichungen. In diesem Beispiel bringt die Filterung eine Verbesserung von 30,5% gegenüber der gewichteten Summe (2.42), die bei diesem Beispiel eine Standardabweichung von $\sigma_s^2 = 29,5$ hat.

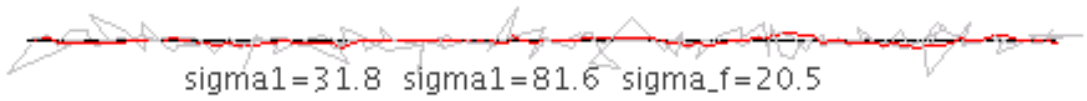


Abbildung 2.9: Test Trajektorie (schwarz), simulierte Messungen (grau, Standardabweichung: $sigma_1$ und $sigma_2$) und gefilterte Trajektorie (rot, Standardabweichung: $sigma_f$).

2.7.2 Praxis

Aus oben genannten Gründen ist eine Bewertung der Trackingalgorithmen schwierig. Unter der Annahme einer geradlinigen Bewegung der verfolgten Person in Abbildung 2.10 ist zu erkennen, dass das Kameratracking eine deutlich höhere Varianz als das Lasertracking aufweist. Die Fusionierung beider Sensormodalitäten durch einen Partikelfilter (hier rot dargestellt) führt zum besten Ergebnis[12].

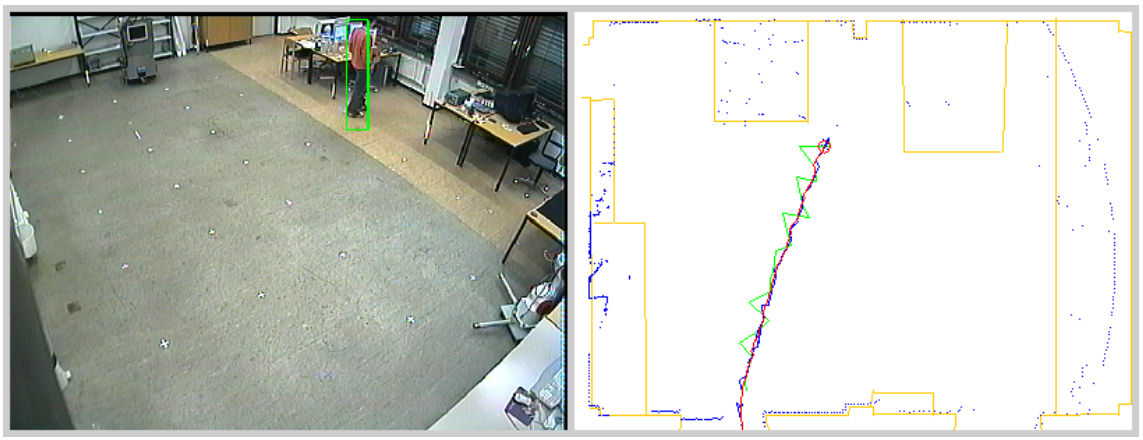


Abbildung 2.10: Vergleich der Trackingmodalitäten: Kameratracking (grün) und Lasertracking (blau) werden durch einen Partikelfilter (rot) fusioniert. Deutlich zu erkennen ist die höhere Varianz des Kameratrackings.

Es kann gezeigt werden, dass die Robustheit des Trackingsystems trotz der hohen Varianz des Kameratrackings durch Multimodalität gesteigert werden kann: In Abbildung 2.11 bewegt sich die Person durch einen von Laserscannern nicht abgedeckten Bereich. Die ungenaueren Messungen des Kameratrackings reichen aus, um die Person zu verfolgen, bis sie wieder von den Lasermessungen erfasst wird.

[12]Die Qualität der Ergebnise ist subjektiv bewertet, da die exakte Trajektorie nicht bekannt ist.

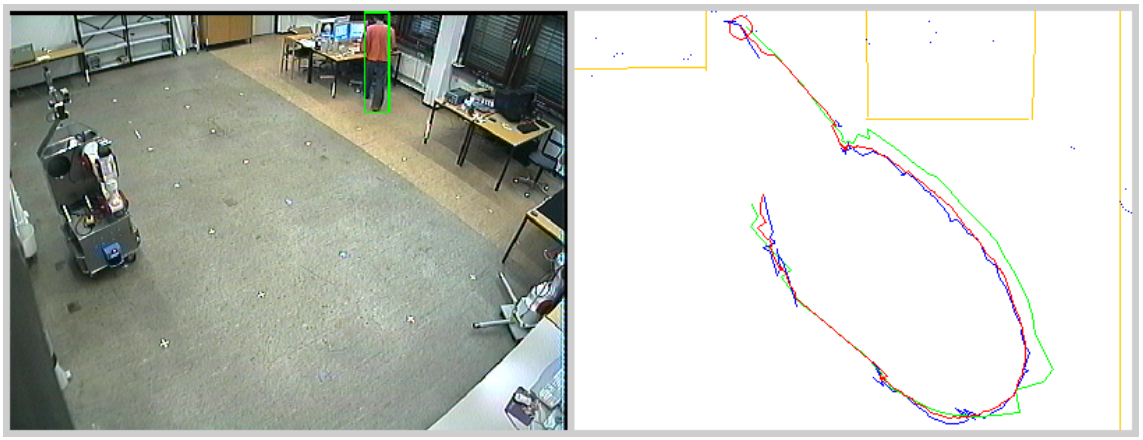


Abbildung 2.11: Die Robustheit des Trackings wird durch Fusionierung (rot) multimodaler Sensordaten erhöht. Das ungenauere Kameratracking (grün) reicht zur Verfolgung der Person aus, bis sie wieder vom genaueren Lasertracking (blau) erfasst wird.

Aus Mangel an Sensoren wurde Multimodalität im Rahmen dieses Projektes genutzt, um den Arbeitsraum des Trackings zu vergrößern. Abbildung 2.12 zeigt den Übergang einer Person vom Arbeitsbereich des einen Sensors in den eines anderen. Der Bereich innerhalb der Tür wird nicht durch Sensoren erfasst. Hier ist zu erkennen, dass der Partikelfilter eine geradlinige Bewegung annimmt.

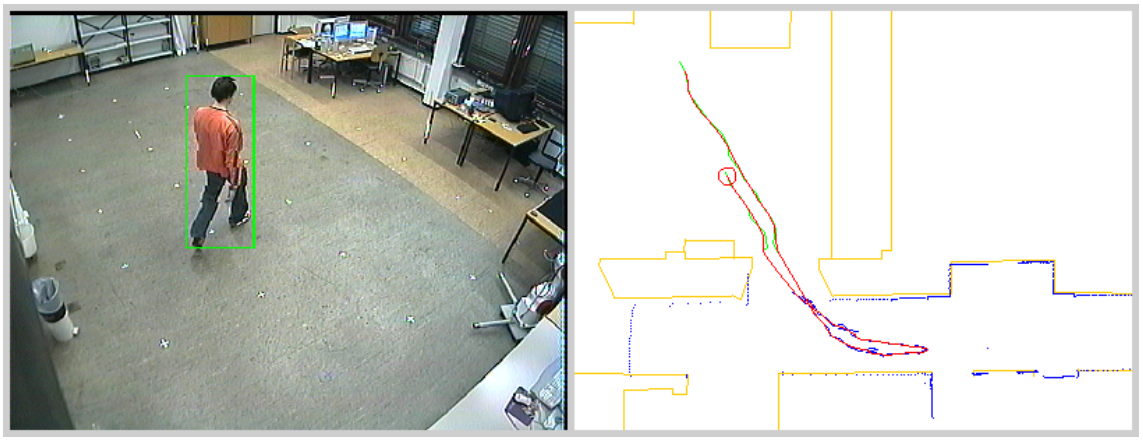


Abbildung 2.12: Die Trajektorie einer Person, wird vom Kameratracking (grün) und Lasertracking (blau) in unterschiedlichen Bereichen erfasst. Der Partikelfilter (rot) fusioniert beide Messungen. Ohne Messung wird eine geradlinigen Bewegung angenommen.

3 Generalisierung und Vorhersage von Bewegungen

Um Vorhersagen über Bewegungen treffen zu können, muss ein Modell vorhanden sein, das die typischen Bewegungen beschreibt. Dieses Bewegungsmodell kann explizit gegeben oder erlernt sein. Explizites Wissen, dass der Vorhersage von Trajektorien dient, kann die Belegung von Räumen mit Arbeitsplätzen oder die Intention einer beobachteten Person sein. Das hier vorgestellte System nutzt erlernte Bewegungsmuster.

Bewegungsmuster können nicht direkt beobachtet werden. Die beobachteten konkreten Trajektorien sind in der Praxis praktisch nie identisch, auch wenn sie dem selben Muster angehören. Selbst exakt gleiche Trajektorien von Personen würden durch das Rauschen der Sensoren zu unterschiedlichen Beobachtungen führen. Die beobachteten Trajektorien müssen in geeigneter Form repräsentiert und generalisiert werden. Häufig begangenen Wege sollen möglichst vollständig repräsentiert und Deteils der einzelnen Trajektorien abstrahiert werden.

Das vorgestellte Verfahren zur Vorhersage von Trajektorien wird in zwei Module unterteilt. Das eine Modul generalisiert Trajektorien und steht im Informationsfluss zwischen dem Trackingmodul (siehe Kapitel 2) und dem Modul, das Trajektorien vorhersagt. Siehe zur Architektur des Gesamtsystems auch Abschnitt 1.4. Als Eingabe der Generalisierung dienen Trajektorien, die durch das Trackingmodul generiert werden. Ausgabe der Generalisierung und Eingabe der Vorhersage sind die häufig begangenen Wege in Form von generalisierten Trajektorien, die im Folgenden als *Pfade* bezeichnet werden. Die Repräsentation der Pfade ist in Kapitel 3.3.4 beschrieben.

Der folgende Abschnitt gibt einen Überblick über aktuelle Arbeiten zum Thema Trajektorienvorhersage. Es werden Vor- und Nachteile der unterschiedlichen Verfahren genannt. Anschliessend wird der hier verwendete neue Ansatz zur Generalisierung mit Self Organizing Maps (SOM) beschrieben. Der darauf folgende Abschnitt beschreibt die Repräsentation der Pfade und das Verfahren, mit dem die implizit in der SOM gespeicherte Information extrahiert wird. Nach einer Beschreibung der Vorhersage, die sich auf die generalisierten Pfade stützt, beschreibt der letzte Abschnitt dieses Kapitels die Ergebnisse der Generalisierung und der Vorhersage.

3.1 Verwandte Arbeiten und Abgrenzung

Die Vorhersage der Bewegungen von Objekten wird häufig zur Kollisionsvermeidung bei der Pfadplanung verwendet. Hierfür ist eine Vorhersage der Gesamttrajektorie nicht notwendig, es werden Bewegungen kürzerer Dauer vorhergesagt. Bei diesen *short term* Vorhersagen wird die Bewegung des Objektes meist durch einen statistischen Prozess modelliert [NLK96, THM+95, YY98]. In [MS03] wird ein Kalmanfilter zur Vorhersage verwendet.

Auch wenn einige dieser Verfahren weitgehend akkurate Vorhersagen versprechen, ist der Nutzen dieser Verfahren zur Kolisionsvermeidung auf Grund des sich in jedem Zeitintervall akkumulierenden Fehlers fraglich [Fok05]. Für Aufgabenplanung auf einer höheren Abstraktionsebene (siehe Beispielszenarien in Abschnitt 1.1) sind *short term* Vorhersagen nicht ausreichend. Zur Vorhersage längerer Zeitintervalle (im folgenden als *long term* Vorhersage bezeichnet) wird angenommen, dass den Bewegungen dynamischer Objekte bestimmte Muster zu Grunde liegen.

In [YY98] wird angenommen, dass sich Personen vorwiegend auf dem Voronoi Graph des Umfelds bewegen. Aufgrund dieser Annahme werden Vorhersagen über Trajektorien von Personen getroffen. Um die Eignung des Voronoi Graphen zur Repräsentation häufig begangener Trajektorien zu testen, wurde im Rahmen dieser Arbeit eine Skelettierung der begehbaren Flächen als Näherung des Voronoi Graphen implementiert. Abbildung 3.1 zeigt, dass häufig begangene Pfade in großen Räumen nicht immer ausreichend repräsentert werden können.

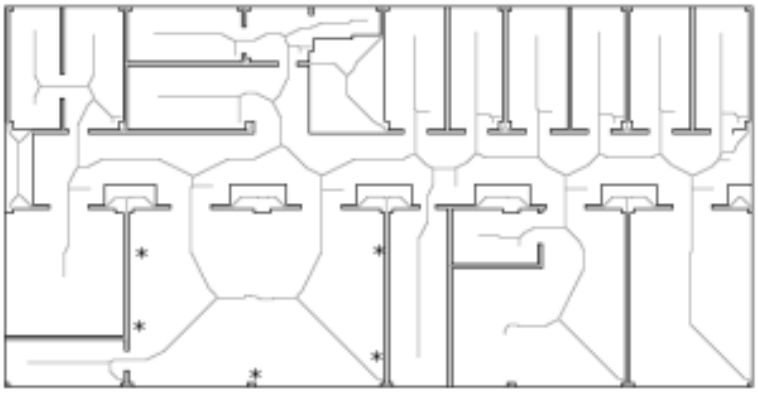


Abbildung 3.1: Der Voronoi Graph (hier durch eine Skelettierung der begehbaren Flächen approximiert) reicht in großen Räumen nicht aus, um häufig begangene Pfade zu repräsentieren. Bei einer Verteilung von Arbeitsplätzen wie hier mit (*) dargestellt, werden die Wege dorthin nicht ausreichend abgebildet.

Die im folgenden genannten Verfahren beobachten und erlernen Bewegungsmuster, um daraus Rückschlüsse über kommende Bewegungen zu ziehen.

In [MN90] wird ein Verfahren vorgestellt, dass Trajektorien diskretisiert und in einem raumzeitlichen Puffer (*trajectory accumulation frame*, TAF) speichert, der als erweitertes diskretes Potentialfeld gesehen werden kann. Die Erweiterung besteht darin, dass für jede Position mehrere Richtungen und Geschwindigkeiten repräsentiert werden können. Ein wesentlicher Nachteil von Potentialfeldmethoden, die mangelnde Flexibilität bei sich überschneidenden Trajektorien (vergleiche [Ben04]), wird dadurch umgangen. Die im TAF gespeicherten Trajektorien werden durch ein Erosion/Dilatation ähnliches Verfahren zu prototypischen Trajektorien generalisiert.

Andere Methoden zur Vorhersage längerer Trajektorien wurden erst in den letzten Jahren [Ben04, VF04, BG04] vorgestellt. In [Ben04] werden ähnliche Trajektorien von Personen durch einen *Expectation Maximation* Algorithmus zu Pfaden zusammengefasst und generalisiert. Bei der Vorhersage wird eine beobachtete Teiltrajektorie mit diesen Pfaden verglichen. Der bezüglich eines definierten Vergleichsmaß ähnlichste Pfad beschreibt die vorhergesagte Trajektorie. Die anderen Verfahren [VF04, BG04] sind ähnlich aufgebaut. Der wesentliche Unterschied besteht in der Art der Generierung von Pfaden. In [VF04] wird ein *pairwise clustering* Prozess angewendet. In [BG04] wird das Ziel der beobachteten Person geschätzt und die vorhergesagte Trajektorie durch einen Pfadplaner berechnet. Ein Nachteil solcher Verfahren ist, dass untypische Trajektorien nicht vorhergesagt werden können.

Das in dieser Arbeit entwickelte und im folgenden vorgestellte Verfahren stellt einen neuen Ansatz dar, mit dem Trajektorien unüberwacht gelernt und generalisiert werden können. Trajektorien dienen hier als Eingabevektoren einer Self Organizing Map. Die Eigenschaft der Topologiebildung einer SOM wird zur Generalisierung der Trajektorien direkt ausgenutzt. Eine Generalisierung der nach dem Lernprozess in der SOM gespeicherten Pfade ist daher nicht notwendig. Die Aufgabe besteht darin, in der Self Organizing Map gespeicherte prototypische Bewegungsmuster explizit nutzbar zu machen.

3.2 Lernen von Trajektorien durch Self Oranizing Maps

Erstmals wurde die Bezeichnung *self-organization* 1947 von W. R. Ashby verwendet [US02]. Eine allgemeine Definition von SOM's wurde bereits Anfang der 80er Jahre von Teuvo Kohonen gegeben. Bis heute konnte jedoch keine universelle Definition von *selbst-organisierend* gefunden werden [US02]. SOM's werden in der Literatur zu den künstlichen neuronalen Netzen gezählt.

Eine SOM ist ein Netz aus Referenzvektoren, das mit Vektoren eines Eingaberaums trainiert werden kann. Nach der Lernphase approximieren die Referenzvektoren die Verteilungsfunktion der Eingabevektoren innerhalb des Eingaberaums.

In dem hier vorgestellen System wird eine SOM mit Trajektorien trainiert, um deren Gruppierung und Speicherung zu bewirken. Nach dem Lernvorgang sind die Trajektorien implizit in generalisierter Form in der SOM enthalten. Ähnliche Trajektorien werden während des Lernprozesses zu Pfaden zusammengefasst. Die gelernten Pfade müssen in geeigneter Form aus der SOM extrahiert werden, um für weitere Algorithmen explizit vorzuliegen. Das hierfür entwickelte Verfahren ist in Abschnitt 3.3 beschrieben.

Der folgende Abschnitt beschreibt die zum Verständnis dieser Arbeit notwendigen Grundlagen der SOM. Es wird die Struktur der SOM sowie die einzelnen Schritte der Lernphase beschrieben und ein Algorithmus vorgestellt, der eine SOM realisiert. Im anschliessenden Abschnitt werden Anpassungen des SOM Algorithmus für das hier vorgestellte System vorgestellt und begründet.

3.2.1 SOM Grundlagen

Die Beschreibung des Algorithmus folgt weitgehend [Koh95] und [US02]. Der Algorithmus ist für diese Arbeit nicht uneingeschränkt nutzbar und wurde für die Anwendung in diesem Projekt erweitert (siehe Kapitel 3.2.2).

Im Folgenden wird erst die Struktur einer SOM definiert. Anschließend werden zwei verschiedene Distanzfunktionen vorgestellt, die zur folgenden Beschreibung der Lernphase einer SOM notwendig sind. Am Ende dieses Abschnittes wird ein Algorithmus vorgestellt, der eine SOM realisiert.

Eine SOM besteht aus einer Menge $M = \{m^i | m^i \in \mathbb{R}^{d_r}\}_{i=1...m}$ von Referenzvektoren (RV), die als *Knoten* bezeichnet werden, und einer Relation $R : (M \times M) \rightarrow \{0, 1\}$, die topologische Verknüpfungen zwischen den Knoten beschreibt. Die Dimension d_r der zu lernenden Eingabevektoren (EV) ist gleich der Dimension der Referenzvektoren (RV).

$$R(m^i, m^j) = \begin{cases} 1 & \text{wenn } m^i \text{ und } m^j \text{ topologisch verknüpft sind} \\ 0 & \text{sonst} \end{cases} \tag{3.1}$$

Die Topologie der SOM kann quadratisch, hexagonal oder irregulär sein. Ein zweidimensionales quadratisches Gitter ermöglicht eine einfache Visualisierung und kann effizient in einem zweidimensionalen Feld gespeichert werden. Die topologischen Verknüpfungen sind dann implizit in der Datenstruktur enthalten und müssen nicht explizit gespeichert werden. Im Folgenden wird von einer quadratischen Gitterstruktur ausgegangen. Die Referenzvektoren werden dementsprechend zu $M^* = \{m^{ij} | m^{ij} \in \mathbb{R}^{d_r}\}_{i=1...m_x, j=1...m_y}$ erweitert, wobei m_x und m_y die Anzahl der Knoten in x- und y-Richtung ist. Die topologische Verknüpfungen (3.1) sind durch

Gleichung 3.2 konkret definiert.

$$R(m^{ij}, m^{kl}) = \begin{cases} 1 & \text{wenn } |i-k| = 1 \text{ und } j = l \\ 1 & \text{wenn } |j-l| = 1 \text{ und } i = k \\ 0 & \text{sonst} \end{cases} \tag{3.2}$$

Die Distanzfunktion zwischen zwei Knoten innerhalb der Topologie der SOM ist frei wählbar. Informationen werden innerhalb des Netzes nur über die topologischen Verbindungen ausgetauscht. Ein Maß für die Anzahl der topologischen Verbindungen, die Informationen von einem Knoten zum anderen durchlaufen müssen, kommt dem biologischen Vorbild eines Neuronalen Netzes am nächsten. Dieses Maß ist durch die Manhatten Distanz (3.3) gegeben.

$$h(m^{ij}, m^{kl}) = |i-k| + |j-l| \tag{3.3}$$

Der im Folgenden verwendete Abstand zwischen zwei Vektoren des Eingaberaums $\mathbb{R}^{d_r}$ ist ebenso frei wählbar. Denkbar sind u.a. Correlation, Minkowski Metrik oder Direction Cosine. In dieser Arbeit wird die Euklidische Distanz (3.4) als Vergleichsmass verwendet werden.

$$|x-y| = \sqrt{\sum_{i=0}^{d_r} (x_i - y_i)^2} \tag{3.4}$$

Während der Lernphase wird die SOM schrittweise mit je einem EV x trainiert. Der Knoten $m^{cd} \in M^*$, mit der größten Übereinstimmung zu x bezüglich des in (3.4) definierten Vergleichsmaßes, wird als *Antwortknoten* bezeichnet. Es gilt

$$min\{|x - m^{ij}| : m^{ij} \in M^*\} = |x - m^{cd}| \tag{3.5}$$

Der Antwortknoten m^{cd} und die Knoten, die ihm in der Gitterstruktur nahe sind, werden dem EV x angeglichen. Welche Knoten m^{ij} dem EV wie stark angeglichen werden, hängt von einer Nachbarschaftsfunktion $n_t(d)$ und der Lernrate $l(t)$ ab, wobei t der Zeitpunkt[1] ist. Die Werte der Knoten $m^{ij} \in M^*$ sind rekursiv durch Gleichung 3.6 definiert.

$$m^{ij}(t+1) = m^{ij}(t) + n_t(d) l(t) \left(x - m^{ij}(t)\right), \tag{3.6}$$

wobei d die Manhatten Distanz zwischen dem Knoten m^{ij} und dem Antwortknoten m^{cd} ist. Die Nachbarschaftsfunktion $n_t(d)$ ist eine Kernelfunktion, die einen Glättungsfaktor über die Gitterstruktur des Netzes beschreibt. Um ein lokales Lernen des Netzes zu ermöglichen, muss $n_t(d)$ einen endlichen Träger haben. Es gilt

[1] Der Zeitpunkt t beschreibt häufig die Anzahl der durchgeführten Lernschritte.

also $n_t(d) \to 0$ für $d > k_t$, wobei k_t die Größe der Nachbarschaftsfunktion bezeichnet. k_t fällt monoton bei wachsendem t. Die Größe und Form von $n_t(d)$ definiert die *Steifheit* des Netzes. Bei geeigneter Wahl der Nachbarschaftsfunktion kann das Verhalten des Netzes während des Lernprozesses als *elastisch* bezeichnet werden. Soll das Netz konvergieren, muss für die Lernrate $l(t) \to 0$ für $t \to \infty$ gelten.

Durch diese Form des unüberwachten Lernens wird die Verteilungsfunktion der Eingabevektoren auf eine Menge von diskreten Referenzvektoren abgebildet. Die Referenzvektoren sind in einem topolgischen Raum geordnet. Diese Abbildung stellt eine nicht lineare Projektion eines möglicherweise hochdimensionalen Eingaberaums auf einen im Allgemeinen niedrig dimensionalen (hier zweidimensionalen) topologischen Raum dar. EV, die im Eingaberaum dicht beieinander liegen, liegen auch innerhalb der Topologie SOM dicht beieinander, der Algorithmus ist somit strukturerhaltend.

Topologiebildung ist eine wesentliche Eigenschaft der SOM. Selbst bei zufälliger initialer Belegung der m^{ij} und hochdimensionalem Eingaberaum lässt sich nach einem (ausreichend langen) Lernprozess eine Ordnung der Referenzvektoren erkennen. Dieser Vorgang wird *Entfaltung* genannt. Ein Beispiel einer Entfaltung ist in Abb. 3.2 visualisiert. Die initiale Belegung der Referenzvektoren ist zufällig in der Mitte verteilt. Die SOM lernt die Verteilung der Eingabevektoren $x \in \mathbb{R}^2$, die hier innerhalb des Quadrates gleichverteilt ist. Die topologischen Verknüpfungen der Knoten sind als Linien dargestellt. Die Kreuzungspunkte der Linien beschreiben die Lage der Referenzvektoren in der Ebene. Abbildung 3.3 zeigt, wie SOM's in der Lage sind, auch andere Verteilungen der Eingabevektoren zu approximieren. Diese Eigenschaft kann zur Visualisierung unbekannter stochastischer Verteilungen im zwei- oder dreidimensionalen Raum genutzt werden. In [UM05, GTC01] sind Verfahren zur Visualisierung höherdimensionaler Daten vorgestellt, die auf SOM's beruhen.

Ein Algorithmus, der eine SOM realisiert, besteht im wesentlichen aus zwei Schritten: Im ersten Schritt werden alle Knoten $m^{ij} \in M^*$ mit dem Eingabevektor verglichen und der Antwortknoten wie in Gleichung 3.5 bestimmt. Anschliessend wird die Position jedes Knotens im Eingaberaum wie in Gleichung 3.6 verschoben. Dieses Vorgehen wird iterativ für jeden Eingabevektor wiederholt. Algorithmus 9 beschreibt dieses Verfahren in Pseudocode.

3.2.2 Modifikation und Realisierung der SOM

In dieser Arbeit wird eine SOM genutzt, um einzelne Trajektorien zu lernen und zu Pfaden zusammenzufassen. Um aus den beobachteten Trajektorien geeignete Eingabevektoren zu generieren, werden Zwischenpunkte dieser Trajektorien in konstanten Abständen interpoliert. Diese Zwischenpunkte stellen die Eingabevektoren der SOM dar.

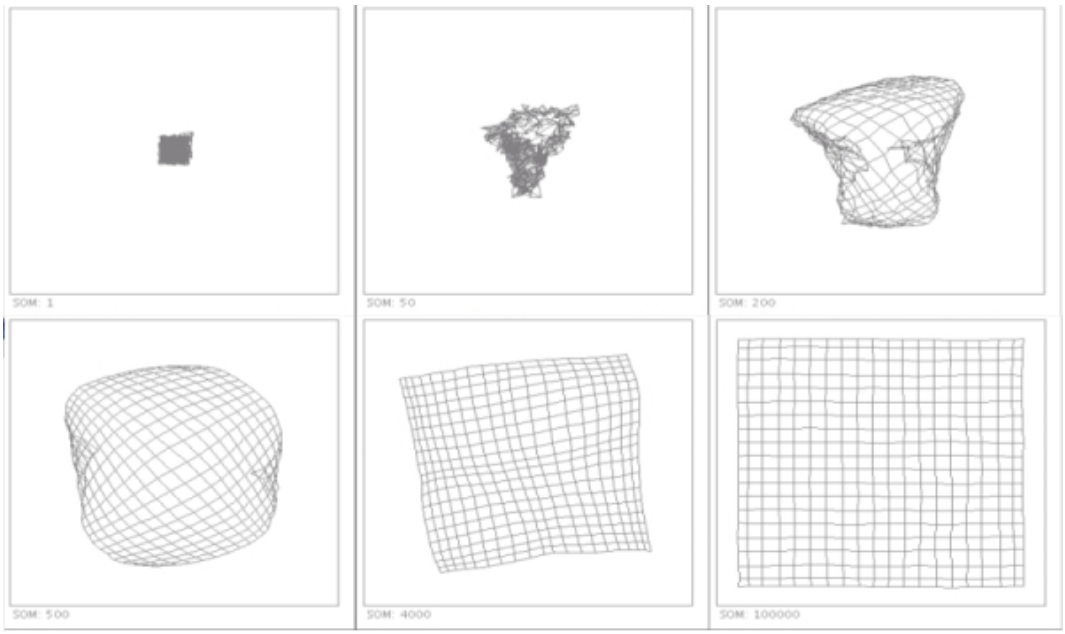
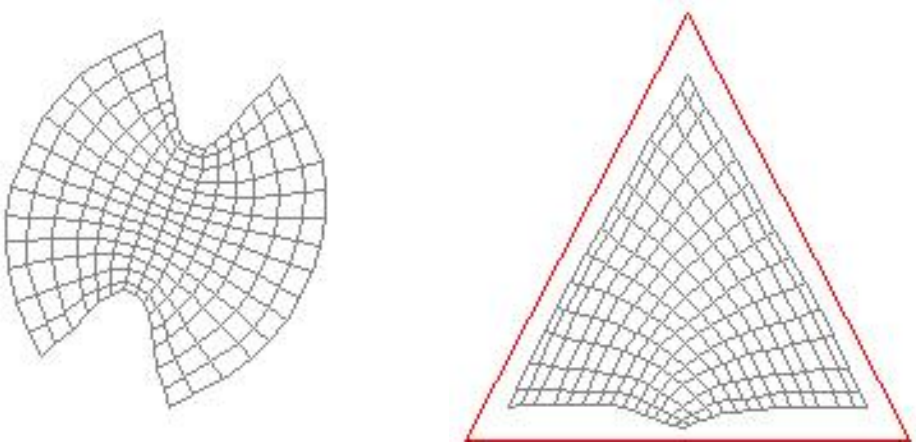


Abbildung 3.2: Eine SOM während des Lernprozesses. Die Eingabevektoren sind innerhalb des Rechteckes gleichverteilt. Die Zahl links unter den Darstellungen entspricht den Lernschritten.



(a) Gaußfunktion

(b) Gleichverteilung innerhalb des Dreieck

Abbildung 3.3: Approximation verschiedener Verteilungsfunktionen der Eingabevektoren durch eine SOM.

Algorithm 9 Lernvorgang einer SOM

Require: Menge M^* der Knoten einer SOM, Menge $\{x^a\}_{a=0...N}$ mit Eingabevektoren

Ensure: Die Menge M^* der Knoten approximiert nach dem Lernvorgang die Verteilung der Eingabevektoren x^a

initialisiere Zähler: $t = 0$
for alle x^a **do**
  initialisiere Gewinnerknoten: $m^{cd} = m^{11}$
  for alle $m^{ij} \in M^*$ **do**
    if $|m^{ij} - x^a| < |m^{cd} - x^a|$ **then**
      belege den Gewinnerknoten neu: $m^{cd} = m^{ij}$
    end if
  end for
  for alle $m^{ij} \in M^*$ **do**
    berechne topologischen Abstand mit (3.3): $d = |i - c| + |j - d|$
    verschiebe den Knoten mit (3.6): $m^{ij} = m^{ij} + n_t(d)l(t)\,(x - m^{ij})$
  end for
  Zähler erhöhen: $t = t + 1$
end for

Aus dem konkreten Szenario mit realen Mauern und dem nur zweidimensionalen Eingaberaum ergeben sich einige Modifikation des SOM Algorithmus, die im Folgenden beschrieben sind.

Geringe Lernrate Die aus den Trajektorien interpolierten Trainingsvektoren sind nicht stochastisch unabhängig. Da aufeinanderfolgende Trainingsvektoren einer Trajektorie sehr dicht beieinander liegen, kommt es bei einer hohen Lernrate häufig dazu, dass derselbe Knoten mehrmals hintereinander zum Gewinnerknoten gewählt wird. Die Lernrate wird in diesem Projekt durch Gleichung 3.7 realisiert.

$$l(t) = \begin{cases} \frac{1}{10} - \frac{t}{2t_{max}} & \text{für } t < t_{max} \\ 0 & \text{sonst} \end{cases}, \tag{3.7}$$

wobei t_{max} die maximale Anzahl der Lernschritte ist.

Einschränkung der Nachbarschaftsfunktion Die Nachbarschaftsbeziehungen der Knoten der SOM erstrecken sich über Raumgrenzen hinweg. Da nicht nur der Gewinnerknoten, sondern auch seine Nachbarn lernen, muss die Nachbarschaftsfunktion dort eingeschränkt werden, wo reale Mauern überschritten werden. Die Nachbarschaftsfunktion, die in Gleichung 3.2 beschrieben ist, wird daher zu Gleichung 3.8

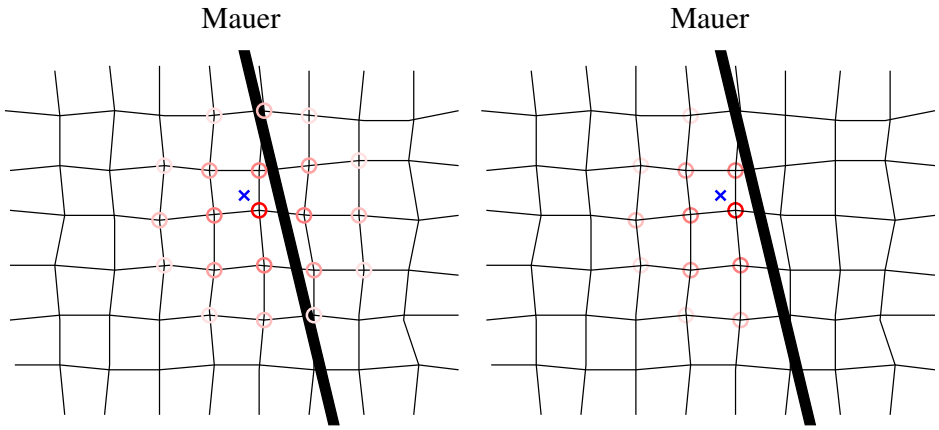


Abbildung 3.4: Eine SOM wird mit allgemeiner (links) und erweiterter (rechts) Nachbarschaftsfunktion trainiert: Der Gewinnerknoten (rot) und seine Nachbarn (pink) lernen von dem Eingabevektor (blau). Je dunkler der Pinkton ist, desto stärker gleichen sich die Knoten dem EV an.

erweitert. Abbildung 3.4 verdeutlicht den Unterschied zwischen allgemeiner und erweiterter Nachbarschaftsfunktion.

$$R(m^{ij}, m^{kl}) = \begin{cases} 0 & \text{wenn eine Mauer zwischen } m^{ij} \text{ und } m^{kl} \text{ liegt} \\ 1 & \text{wenn } |i-k| = 1 \text{ und } j = l \\ 1 & \text{wenn } |j-l| = 1 \text{ und } i = k \\ 0 & \text{sonst} \end{cases} \tag{3.8}$$

Größe der Nachbarschaft Zu Beginn der Lernphase wird die Größe der Nachbarschaft $n_t(d)$ häufig über zwei Drittel des gesammten Netzes gewählt [Koh95, Seite 80]. Dies ist notwendig, damit die SOM sich topologisch ordnen (*entfalten*) kann. In der hier vorgestellten Arbeit ist die Dimension der Eingabevektoren gleich der Dimension der Topologie des Netzes. Dadurch ist es möglich, die initiale Belegung der Referenzvektoren geordnet zu wählen. Eine anfängliche Entfaltung der SOM ist somit nicht notwendig. Die Nachbarschaftsfunktion kann daher schon zu Beginn relativ klein gewählt werden. Eine Nachbarschaftsfunktion der maximalen Größe $k_0 = 4$ hat sich als sinnvoll herausgestellt. Es lernen dadurch nur der Antwortknoten und die Knoten, die in der Topologie weniger als vier Knoten entfernt sind. Gleichung 3.9 beschreibt die hier verwendete Nachbarschaftsfunktion.

$$n_t(d) = \begin{cases} 1 - \frac{d}{k_t} & \text{für } d < k_t \\ 0 & \text{sonst} \end{cases} \tag{3.9}$$

wobei die Größe der Nachbarschaftsfunktion k_t für jeden Zeitpunkt t durch Gleichung 3.10 beschrieben ist.

$$k_t = \begin{cases} 4 - \frac{4t}{t_{max}} & \text{für } t < t_{max} \\ 0 & \text{sonst} \end{cases} \tag{3.10}$$

Erweiterung des Vergleichsmaßes zwischen Vektoren Es kann dazu kommen, dass Eingabevektor und Gewinnerknoten, wie Graphik 3.5 verdeutlicht, auf verschiedenen Seiten einer Mauer liegen. Die Trainingsvektoren stellen hier Orte von

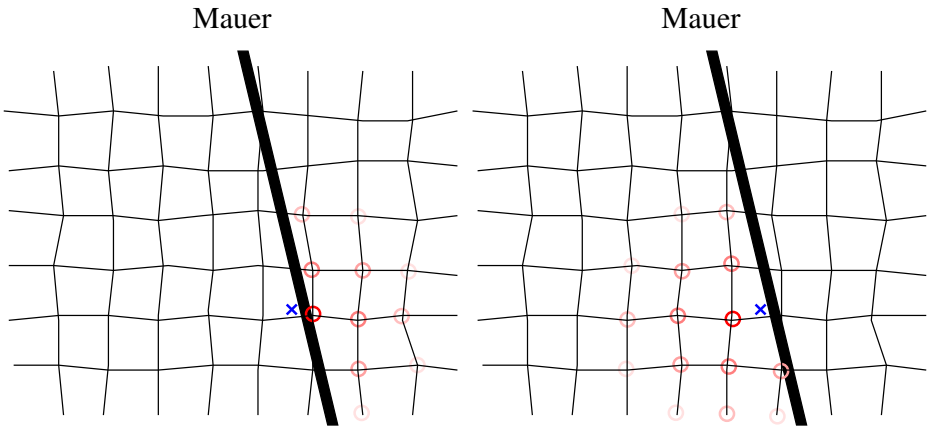


Abbildung 3.5: Eine SOM wird mit allgemeinem (links) und erweitertem (rechts) Vergleichsmaß der Eingabevektoren trainiert. Die Farbnotation entspricht der in Abb. 3.4 verwendeten.

getrackten Personen dar. Das Angleichen der Referenzvektoren an Eingabevektoren, die in anderen Räumen liegen, macht keinen Sinn. Es wird daher für die Bestimmung des Gewinnerknotens die Distanzfunktion der Gleichung 3.4 zu Gleichung 3.11 erweitert.

$$|x - y| = \begin{cases} \infty & \text{wenn Mauer zwischen } x \text{ und } y \text{ liegt} \\ \sqrt{\sum_{i=0}^{1}(x_i - y_i)^2} & \text{sonst} \end{cases} \tag{3.11}$$

3.2.3 Reduktion der Komplexität des SOM Algorithmus

Die hier verwendete SOM ist mit quadratischer Topologie organisiert. Die Knoten werden, wie in 3.2.1 vorgeschlagen, in einem zweidimensionalen Feld gespeichert. Bei der Initialisierung wird das Netz gleichmäßig über die gesamte zu betrachtende

Fläche gespannt, die sich hier über den Flur des AB TAMS erstreckt. Die Abstände zwischen den Knoten werden mit einem konstanten Wert $w = 333$mm initialisiert.

Durch die bekannte Initialisierung kann vor dem ersten Lernschritt der Index des Gewinnerknotens aus dem Eingabevektor durch eine lineare Transformation direkt berechnet werden. Durch die geringe Lernrate und die aus Mauern resultierende Einschränkung der Nachbarschaftsfunktion können sich die Referenzvektoren der Knoten nicht unbegrenzt bewegen. Es muss daher nicht die ganze Menge der Knoten zur Bestimmung des Gewinnerknotens betrachtet werden. Wird der Gewinnerknoten m^{cd}_{init} im untrainierten Netz berechnet, so kann angenommen werden, dass sich der Gewinnerknoten m^{cd} des trainierten Netzes in der topologischen Nähe befindet. Es müssen also nur die Knoten in der topologischen Nachbarschaft von m^{cd}_{init} betrachtet werden. Die Größe dieser zu betrachtenden Nachbarschaft hängt von der maximalen Verschiebung der Referenzvektoren und damit von der maximalen Raumgröße ab. Die Komplexität des Lernschrittes wächst dadurch mit der maximalen Raumgröße und nicht mehr mit der Größe der gesamten SOM.

3.3 Extraktion und Repräsentation von Pfaden

Die Knoten der SOM approximieren nach einem ausreichend langem Lernvorgang die Dichteverteilung der Eingabevektoren. Die Dichte der Knoten an einer Position entspricht hier also der Wahrscheinlichkeit, dass diese Position zu einer gelernten Trajektorie gehört. Um Trajektorien aus der SOM zu extrahieren, werden Regionen mit großer Knotendichte skelettiert.

Im folgenden Abschnitt wird das Verfahren zur Bestimmung der Dichteverteilung vorgestellt. Nach der Beschreibung eines alternativen Ansatzes, der eine Flächenverteilung zwischen den Knoten berechnet, wird die Repräsentation der Pfade beschrieben. Die Pfade werden als Ergebnis des Generalisierungsmoduls an das Vorhersagemodul weitergeleitet.

3.3.1 Dichteverteilung

In Kapitel 2.6.1 wurde bereits die Schätzung einer d-dimensionalen Kerneldichte (*mkde*) eingeführt (2.28). Als Kernel wird hier die Gaußfunktion verwendet, wodurch sich als Kernelprofil Gleichung 3.12 ergiebt.

$$g(x) = e^{-x} \tag{3.12}$$

Wird die *mkde* (2.28) mit Kernelprofil g auf die zweidimensionalen Knoten $\{m^{ij} | m^{ij} \in \mathbb{R}^2\}_{i=1\ldots m_x, j=1\ldots m_y}$ der SOM angewendet, so ergibt sich die Dichte der SOM mit Gleichung 3.13

$$\hat{d}_g(x) = \frac{1}{m_x m_y h^2} \sum_{i=1}^{m_x} \sum_{j=1}^{m_y} g\left(\left| \frac{x - m^{ij}}{h} \right|^2 \right), \tag{3.13}$$

wobei sich die Größe h des Kernels hier mit dem Dreifachen des initialen Knotenabstands w der SOM als sinnvoll gezeigt hat. Abbildung 3.6 visualisiert die Dichteverteilung einer trainierten SOM.

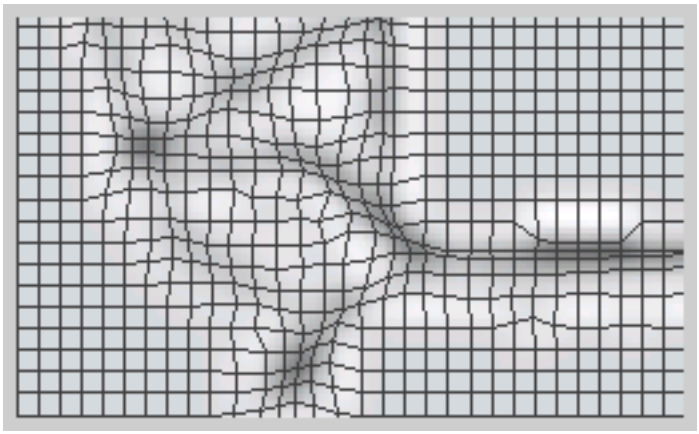


Abbildung 3.6: Eine SOM (schwarze Linien) und die *mkde* der Knoten. Je dunkler der Hintergrund, desto größer die Knotendichte.

3.3.2 Flächenverteilung

Die Menge der Knoten der SOM bleibt während des Lernvorgangs konstant. Die mittlere Dichte der Knoten über die ganze Ausbreitung der SOM bleibt dadurch unverändert. Die Dichteänderung während des Lernvorgangs beruht nur auf einer lokalen Umverteilung der Knoten. Wenn sich die Knotendichte auf einem Pfad erhöht, veringert sich dadurch die Dichte um den Pfad herum. In Abbildung 3.6 ist zu erkennen, dass am Rand jedes lokalen Dichtemaximums ein lokales Minimum liegt. Auf die Anwendung bezogen bedeutet dies, dass es besonders unwahrscheinlich ist, dass sich Personen dicht neben Pfaden bewegen. Dies entspricht nicht der Realität, die Dichteverteilung von Knoten einer SOM ist somit nur bedingt geeignet, um häufig begangene Pfade zu beschreiben.

Alternativ zur Dichtevertelung der Knoten einer SOM wurde ein Verfahren entwickelt und erprobt, das anstelle der Verteilung der Knoten die Verteilung der Fläche zwischen den Knoten berücksichtigt. In der quadratischen Topologie der SOM umschliessen je vier geschlossen zusammenhängende Knoten eine Fläche. Die gesamte Fläche der SOM bleibt ebenso wie die Anzahl der Knoten konstant. Um dennoch den Kontrast zwischen häufig und weniger häufig begangen Flächen zu erhöhen, wird jedem Knoten der Wert der kleinsten Fläche zugewiesen, die an den Knoten grenzt. Die Werte der Knoten werden bilinear in die Grundebene interpoliert und

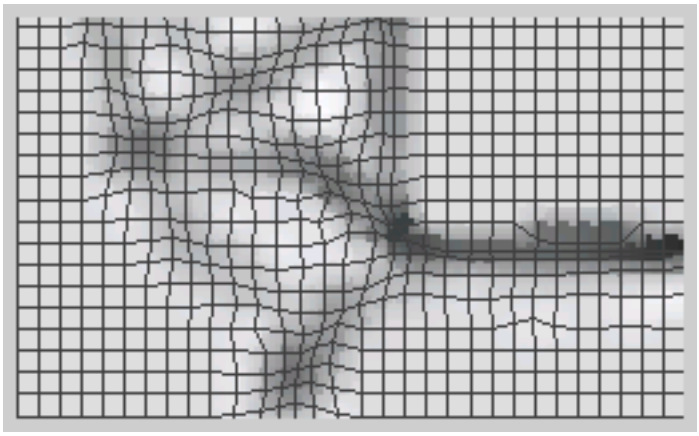


Abbildung 3.7: Eine SOM (schwarze Linien) und die in Abschnitt 3.3.2 beschriebene Flächenverteilung. Je dunkler der Hintergrund, desto kleiner ist die minimale Fläche, die an die umliegenden Knoten angrenzt.

anschließend normiert. Die so entstehende Flächenverteilung wird im folgenden Kapitel analog der Dichteverteilung der Knoten verwendet.

Abbildung 3.7 zeigt die Flächenverteilung im Vergleich zur in Abbildung 3.6 dargestellten Knotendichte, unter Verwendung der selben SOM. Der Kontrast zwischen mehr und weniger begangenen Flächen ist größer und die lokalen Minima in der Nachbarschaft von Maxima sind weniger ausgeprägt.

3.3.3 Skelettierung

Im Folgenden wird davon ausgegangen, dass die betrachtete Fläche einer der oben beschriebenen Verteilungsfunktionen in ein Raster $R = r^{xy}$ mit $x = 1 \dots r_x$ und $y = 1 \dots r_y$ der Breite r_x und Höhe r_y diskretisiert wird, wobei die Rasterpunkte r^{xy} (im folgenden als *Pixel* bezeichnet) den Werten der Verteilungsfunktion an der Stelle $(x, y)^T$ entsprechen.

Pixel mit $r^{xy} = 0$ werden als Hintergrund betrachtet und Pixel mit $r^{xy} = 1$ als Vordergrund. Um eine erste Näherung der Trajektorien zu bestimmen, werden Pixel, deren Wert unter einem bestimmten Schwellwert r_{thr} liegt, als Hintergrund klassifiziert. Gleichung 3.14 beschreibt diesen Vorgang.

$$r^{xy} = \begin{cases} 0 & \text{falls } r^{xy} < r_{thr} \\ r^{xy} & \text{sonst} \end{cases} \tag{3.14}$$

Der Schwellwert r_{thr} wird oberhalb der mittleren Dichte (bzw. der mittleren Fläche)

der untrainierten SOM gewählt. Dadurch wird garantiert, dass nie begangene Bereiche als Hintergrund betrachtet werden.

Der folgende Skelettierungsalgorithmus markiert iterativ Randpixel der unklassifizierten Fläche als Vorder- oder Hintergrund. Randpixel sind unklassifizierte Pixel, die ein Hintergrundpixel in ihrer Vierernachbarschaft haben. Randpixel werden als Hintergrund markiert, wenn sie

- mehr als ein Vordergrundpixel in der Achternachbarschaft besitzen
- und beim Umlauf um das fragliche Pixel[2] weniger als drei Wechsel zwischen Hintergrund und Vordergrund aufweisen.

Ist eine dieser Bedingungen nicht erfüllt, sind diese Randpixel Topologie erhaltend und werden als Vordergrund markiert. Algorithmus 10 beschreibt diesen Vorgang, es werden Randpixel temporär mit $r^{xy} = -1$ belegt. Dabei realisieren die Prädikate $hintergrundpixel(r^{xy})$ und $randpixel(r^{xy})$ die oben beschriebenen Bedingungen.

3.3.4 Repräsentation der generalisierten Trajektorien

Es liegen nun Pfade in skelettierter Form vor. Nicht repräsentiert sind bis jetzt die Wahrscheinlichkeiten, dass eine Person links oder rechts abbiegt, wenn sie an einer Verzweigung der Pfade angelangt ist. Um diese Übergänge effizient beobachten zu können, wird die skelettartige Repräsentation in einen topologischen Graph, den *Bewegungsgraph*, überführt. Ein weiterer Grund für die Repräsentation in Graphform ist eine Vielzahl bekannter und effizienter Graphalgorithmen [Sed02].

Dieser Bewegungsgraph beinhaltet als Knotenpunkte die Verzweigungs- und Endpunkte des Skeletts. Die Knotenpunkte sind durch ungerichtete Kanten miteinander verbunden. Ein Knotenpaar kann durch mehrere Kanten miteinander verbunden sein. Für die Kanten des Graphen wurde eine Schnittstelle definiert, die Referenzen auf die verbundenen Knotenpunkte und eine Distanzfunktion beinhaltet, die bei der Vorhersage für die Zuordnung von beobachteten Personen zu Kanten (siehe Abschnitt 3.4.1) genutzt wird. Die konkrete Repräsentation der Kanten ist austauschbar. Denkbar sind beispielsweise B-Splines oder Folgen von Zwischenpunkten. Die hier verwendete Repräsentation approximiert das Skelettsegment durch eine Folge von Liniensegmenten. Abbildung 3.8 zeigt die Flächenverteilung einer SOM und den approximierenden Bewegungsgraphen.

3.4 Vorhersage von Trajektorien

Vorhersagen können in *short term* und *long term* unterschieden werden. Als *short term* Vorhersage wird im Allgemeinen eine Schätzung der Position der beobachteten

[2] Bei dem Umlauf um das Pixel wird die Achternachbarschaft betrachtet.

Algorithm 10 Skelettierung der Vordergrundpixel

Require: Menge $R = r^{xy}$ mit $x = 1 \dots r_x, y = 1 \dots r_y$ der zu skelettierenden Verteilung

Ensure: Menge R mit $r^{xy} \in \{0, 1\}$ der skelettierten Verteilung

```
1: Erste Näherung des Hintergrundes bestimmen:
2: for alle r^xy ∈ R do
3:    Klassifiziere r^xy mit (3.14) grob in Vorder- oder Hintergrund
4: end for
5: Terminierung des Algorithmus: end = false
6: while ¬end do
7:    end = true
8:    for alle r^xy ∈ R do
9:       if randpixel(r^xy) then
10:         kennzeichne Randpixel: r^xy = −1
11:         end = false
12:      end if
13:   end for
14:   for alle r^xy ∈ R do
15:      if r^xy = −1 then
16:         if ¬hintergrundtpixel(r^xy) then
17:            Kennzeichne Pixel als Vordergrund: r^xy = 1
18:         else
19:            Kennzeichne Pixel als Hintergrund: r^xy = 0
20:         end if
21:      end if
22:   end for
23: end while
```

Person zum unmittelbar folgenden Zeitpunkt bezeichnet. Bei der hier realisierten Trajektorienvorhersage wird angenommen, dass die betrachtete Person einem Pfad folgt, der durch eine Kante des Bewegungsgraphen repräsentiert wird. In diesem Konetxt bezeichnet eine *short term* Vorhersage die Vorhersage der Trajektorie bis zu einem Knotenpunkt des Graphen. Diese Knotenpunkte stellen Verzweigungen oder Endpunkte der Pfade dar. Die *long term* Vorhersage beschreibt die darüber hinaus gehende Schätzung, in welche Richtung sich eine Person bei möglichen Verzweigungen begeben wird.

Der nächste Abschnitt beschreibt die Zuordnung einer Person zu dem Pfadsegment, auf dem sie sich am wahrscheinlichsten bewegt. Der darauf folgende Abschnitt beschreibt die eigentliche Vorhersage von Trajektorien in unterschiedlich langen Zeitintervallen.

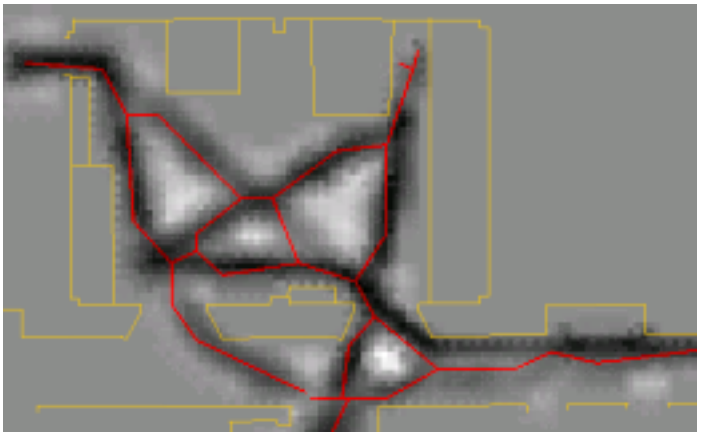


Abbildung 3.8: Die Flächenverteilung einer trainierten SOM und der approximierende Bewegungsgraph (rot).

3.4.1 Zuordnung von Personen zu Pfadsegmenten

Um eine Zuordnung einer Person zu einem Pfadsegment realisieren zu können, muss ein geeignetes Ähnlichkeitsmaß zwischen Person und Pfadsegment definiert werden. In [Ben04] wird ein Maß definiert, dass die Ähnlichkeit eines Trajektoriensegmentes mit einer Gesamttrajektorie beschreibt. Aufgrund der vom Trackingmodul realisierten Schnittstelle (siehe A.3) können hier nur die Position, die Richtung und deren Standardabweichung im Vergleichsmaß berücksicht werden. Die Möglichkeit, die übergebenen *Tracks* (siehe A.3) zu speichern und mit Hilfe ihrer eindeutigen ID zu Trajektorien zusammenzufügen, wird hier nicht betrachtet. Das Zuordnungsproblem ist hier in die Realisierung der Distanzfunktion der Kanten des Graphen verlagert (siehe dazu Abschnitt 3.3.4). Die hier verwendete Realisierung der Kanten durch Liniensegmente implementiert die Distanzfunktion durch den minimalen euklidischen Abstand zwischen Person und Liniensegment. Die Richtung der Person wird bei der Zuordnung nicht berücksichtigt, da die Kanten des Bewegungsgraphen ungerichtet sind.

3.4.2 Short term Vorhersage von Trajektorien

Der beobachteten Person wird zu jedem Zeitpunkt die Kante des Bewegungsgraphen zugeordnet, die die minimale Distanz zur Position der Person aufweist. Es wird angenommen, dass die Person dem durch die Kante beschriebenen Pfad folgt. Die Richtung der Person auf dem Pfad ist durch die beobachtete Richtung der Person bestimmt.

3.4.3 Long term Vorhersage von Trajektorien

Es wird vorrausgesetzt, dass sich das Verhalten der beobachteten Personen während des Lernvorgangs und der Vorhersage nicht drastisch ändert. Dadurch kann angenommen werden, dass die Verteilung der Abbiegerichtungen einer Verzweigung in der Vergangenheit und der Zukunft einander entsprechen.

Die Vorhersage der Abbiegerichtung einer Person wird daher direkt mit der Verteilung der beobachteten vergangenen Abbiegerichtungen gleich gesetzt. Dieses einfache Vorgehen hat zur Folge, dass eine Vorhersage dort unmöglich ist, wo Abbiegerichtungen gleich verteilt sind. Sind die Abbiegerichtungen jedoch ungleich verteilt, kann eine realistische Vorhersage getroffen werden. Ein Abbiegevorgang wird immer dann beobachtet, wenn die Kante, die einer Person zugeordnete wird, sich zwischen aufeinanderfolgenden Zeitpunkten ändert und wenn die Kanten durch einen gemeinsamen Knoten verbunden sind. Abbildung 3.12 zeigt die Häufigkeiten der beobachteten Abbiegerichtungen einer Verzweigung des Bewegungsgraphen.

Die *long term* Vorhersage erweitert die durch die *short term* Vorhersage gewählte Kante des Graphen iterativ um die Kante, deren Abbiegerichtung am wahrscheinlichsten ist. Die Wahrscheinlichkeit der so vorhergesagten Gesamttrajektorie entspricht dem Produkt der Abbiegewahrscheinlichkeiten. Die vorhergesagte Trajektorie endet dort, wo die Gesamtwahrscheinlichkeit einen bestimmten Schwellwert (hier 10%) unterschreitet. Abbildung 3.13 zeigt eine so berechnete *long term* Vorhersage.

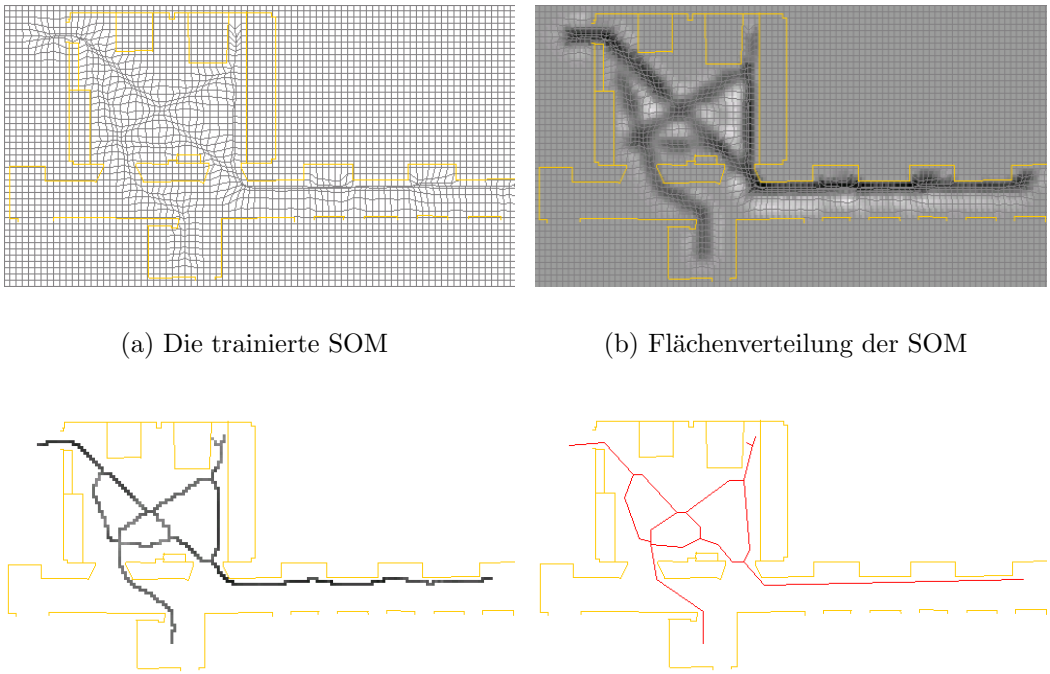
3.5 Experimentelle Ergebnisse

Sowol die *long term* als auch die *short term* Vorhersage hängen stark von der Güte des in Abschnitt 3.3.4 beschriebenen Bewegungsgraphen ab. Bewegt sich eine Person auf einer Trajektorie, die nicht durch eine Kante des Bewegungsgraphen repräsentiert wird, geht die *short term* Vorhersage fehl. Die Übergänge der fehlerhaft zugeordneten Kanten verfälschen die Abbiegewahrscheinlichkeiten der Verzweigungen des Graphen und wirken sich dadurch negativ auf die *long term* Vorhersage aus. Aufgrund der in Abschnitt 1.4 beschriebenen Architektur werden im Folgenden die Ergebnisse der oben beschriebenen Module getrennt beschrieben.

3.5.1 Ergebnisse der Generalisierung

Abbildung 3.9 visualisiert die Zwischenschritte bei der Bildung des Bewegungsgraphen. Die zugrunde liegende SOM wurde mit simulierten Trajektorien trainiert. Als Grundlage der Bildung eines Bewegungsgraphen wurden zwei verschiedene Verteilungsfunktionen erprobt: Die Dichteverteilung der Knoten der SOM hat sich als wenig sinnvoll erwiesen, da lokale Maxima immer von lokalen Minima umgeben sind.

Auf die Anwendung übertragen bedeutet dies, dass es für Personen unwahrscheinlicher ist, neben Pfaden zu gehen, als in nie begangenen Bereichen. Dies entspricht nicht der Realität. Alternativ wurde eine Flächenverteilung erprobt. Vorteile der Flächenverteilung sind ein wesentlich höherer Kontrast zwischen mehr und weniger begangenen Flächen und kaum vorhandene lokale Minima an den Rändern der Pfade. Die beiden Verteilungsfunktionen sind in Abbildung 3.10 gegenübergestellt.



(a) Die trainierte SOM

(b) Flächenverteilung der SOM

(c) Skelettierte Flächenverteilung

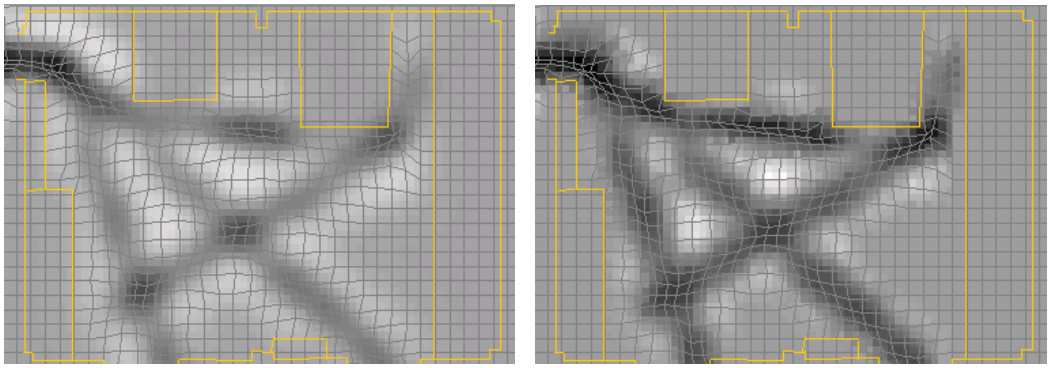
(d) Der approximierende Bewegungsgraph

Abbildung 3.9: Die implizit in der SOM enthaltenen Informationen werden in einen Bewegungsgraph überführt.

Die im Rahmen dieser Arbeit sensorisch erfassbare Fläche ist begrenzt. Da diese Fläche nur wenig unterschiedliche Trajektorien zulässt, sind Ergebnisse aus realen Eingabetrajektorien im Rahmen dieser Arbeit schwer zu bewerten. Abbildung 3.11 zeigt die während eines mehrstündigen Versuchs aufgenommenen Trajektorien und die daraus berechneten Pfade.

3.5.2 Ergebnisse der Vorhersage

Das in Abschnitt 3.4.3 beschriebene Verfahren lernt unüberwacht die Übergänge an Verzweigungen des Bewegungsgraphen. Die beobachteten Übergangswahrschein-



(a) Die Dichteverteilung der Knoten

(b) Die Verteilung der Flächen, die durch je vier Knoten umschlossen sind

Abbildung 3.10: Gegenüberstellung der verschiedenen Verteilungen bei gleicher SOM: Die Flächenverteilung eignet sich auf Grund ihres größeren Kontrastes besser zur anschliessenden Skelettierung.

Abbildung 3.11: Generalisierung meherer Trajektorien (hellblau) zu einem Bewegungsgraphen (rot).

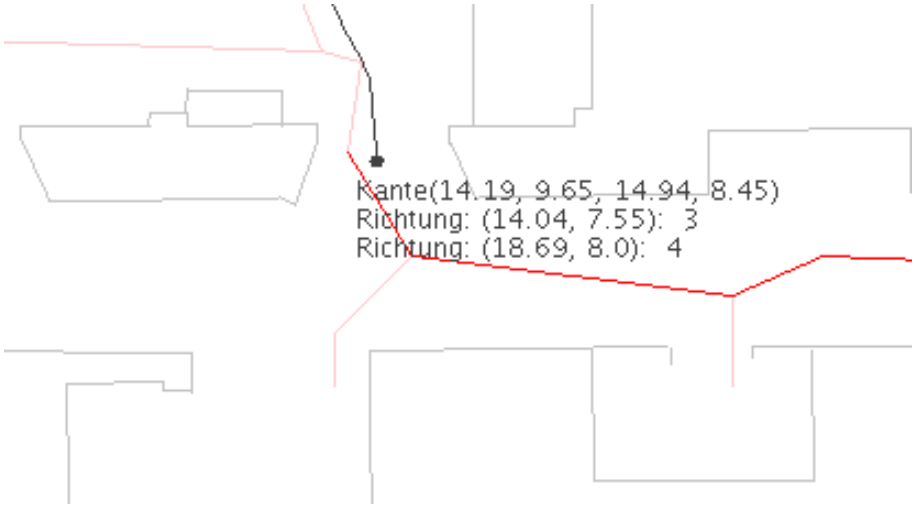
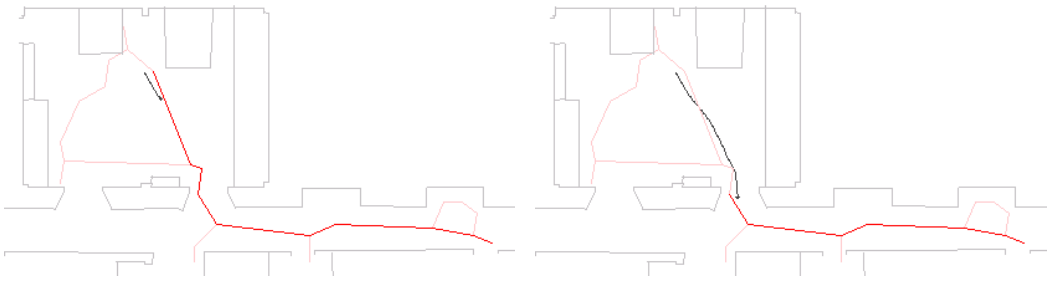


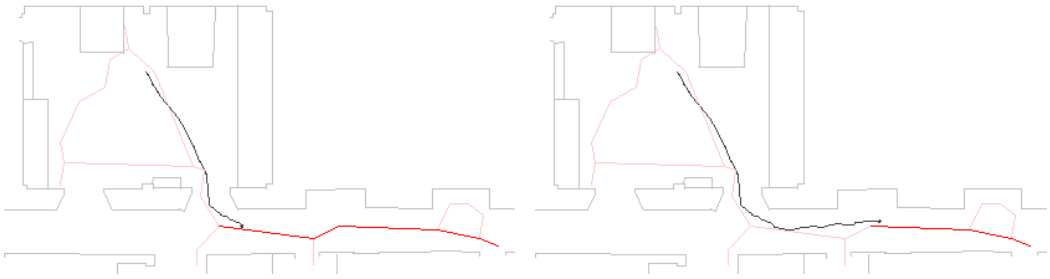
Abbildung 3.12: Die Häufigkeiten der Abbiegerichtungen geben die Wahrscheinlichkeiten der folgenden Abbiegerichtungen an. Es wurden vier Abbiegevorgänge nach rechts und drei nach links beobachtet. Die Wahrscheinlichkeit, dass die Person aus Abbildung 3.13(b) rechts abbiegt, beträgt daher $\frac{4}{7}$.

lichkeiten werden für eine Vorhersage von Trajektorien über größere Zeiträume genutzt. Die Wahrscheinlichkeit einer korrekten Vorhersage hängt direkt damit zusammen, wie die Abbiegerichtungen an Verzweigungen des Graphen verteilt sind. Bei großer Varianz der Richtungsverteilungen ist eine Vorhersage mit guten Ergebnissen möglich, da angenommen wird, dass sich die Verteilung über die Zeit nicht signifikant ändert.

Praktische Versuche im Flur des AB TAMS zeigen, dass die hier beschriebenen Verfahren zur Generalisierung und zur Vorhersage von Trajektorien in Büroumgebungen nutzbare Ergebnisse liefern. Abbbildung 3.13 zeigt, dass die *long term* Vorhergesage der Trajektorie einer Person dem Flur folgt. Diese Vorhersage ist ohne personifizierte Daten die einzig sinnvolle, da die meisten Trajektorien dem Flur folgen, bevor sie in einem seperaten Raum enden. Abbildung 3.12 zeigt die Häufigkeit der beobachteten Abbiegerichtungen an der Kante, die der Person aus Abbildung 3.13(b) zugeordnet ist.



(a) (b)



(c) (d)

Abbildung 3.13: *long term* Vorhersage (rot) einer realen Trajektorie (schwarz) durch Auswertung der Abbiegehäufigkeiten an Verzweigungen des Bewegungsgraphen (hellrot).

4 Zusammenfassung und Ausblick

In dieser Arbeit wurde ein konzeptioneller Rahmen entwickelt, der Algorithmen zum Verfolgen von Personen und die Vorhersage von Trajektorien integriert. Darauf aufbauend wurde ein System realisiert, dass multimodale Sensordaten verarbeitet und zu einem robusten Tracking von Personen führt. Es wurden Algorithmen zum Erkennen von Personen in Kamerabildern und in Laserentfernungsmessungen implementiert und durch das System für ein multimodales Tracking zusammengefasst. Es konnte gezeigt werden, dass die Robustheit und Genauigkeit des Trakingsystems durch Multimodalität verbessert werden kann.

Weiterhin wurde ein neues Verfahren entwickelt, dass die beobachteten Trajektorien durch eine SOM lernt und zu Bewegungsmustern generalisiert. Diese Muster werden in einem Bewegungsgraph repräsentiert, auf dem ein Verfahren implementiert wurde, dass unüberwacht die Übergangswahrscheinlichkeiten an Verzweigungen lernt. Darauf aufbauend wurde eine Vorhersage von Trajektorien realisiert und mit Erfolg getestet.

4.1 Ergebnisse

Im Folgenden werden die Ergebnisse des Trackings und der Trajektorienvorhersage zusammengefasst. Für eine detaillierte Beschreibung der Ergebnisse sei auf die Abschnitte 2.7 und 3.5 verwiesen.

4.1.1 Ergebnisse der Untersuchungen

Es konnte gezeigt werden, dass ein Partikelfilter den mittleren Fehler der Positionsschätzung durch Sensormessungen verringern kann, wenn die Bewegung dem angenommenen Systemmodell entspricht. Da die Position von Personen im Rahmen dieser Arbeit nicht genau ermittelt werden konnte, wurden diese Untersuchungen mit simulierten Messungen durchgeführt.

Bei der Filterung realer multimodaler Sensordaten konnte gezeigt werden, dass die Genauigkeit und Robustheit des Trackings auch von Modalitäten profitiert, die eine

hohe Standardabweichung der Messungen aufweisen. Im Rahmen dieser Arbeit wurde Multimodalität unter Anderem zur Vergrößerung des Arbeitsraumes des Systems genutzt.

Um den entwickelten konzeptionellen Rahmen auf Erweiterbarkeit und Flexibilität zu prüfen, wurden mehrere reale und simulierte Eingabequellen parallel getestet.

4.1.2 Ergebnisse der Generalisierung und Vorhersage von Trajektorien

Eine Analyse der Ergebnisse der Generalisierung von Trajektorien ist aufwendig, da die Relevanz von Pfaden subjektiv ist und dadurch nur manuell bestimmt werden kann. Abbildung 3.11 zeigt jedoch, dass die Ergebnisse dem subjektiven Empfinden nach alle wichtigen Pfade beinhalten.

Eine seperate Analyse von Generalisierung und Vorhersage ist nicht sinnvoll, da der generalisierte Bewegungsbaum Grundlage der Vorhersage ist. Es konnte gezeigt werden, dass die *long term* Vorhersage gute Ergebnisse liefert, wenn der Bewegungsbaum die vorherzusagende Trajektorie ausreichend abbildet. Da außer vorher beobachteten Trajektorien keine weitere Information genutzt wird, kann eine zufällige Trajektorie nicht vorhergesagt werden. Sind mehrere teilweise parallele Trajektorien gleich häufig beobachtet wurden, kann der Weg einer Person, die auf dem gemeinsamen Trajektoriensegment geht, nicht eindeutig entschieden werden. Es wurde gezeigt, dass die Vorhersage in der Büroumgebung des AB TAMS gute Ergebnisse liefert.

4.2 Ausblick

Ein wesentlicher Nachteil des hier vorgestellten Systems ist die Zweistufigkeit des Lernprozesses. Es werden unabhängig voneinander Trajektorien zu Pfaden generalisiert und Abbiegewahrscheinlichkeiten gelernt. Dieser Nachteil könnte dadurch vermieden werden, dass die zur Generalisierung verwendete SOM mit Eingabevektoren trainiert wird, die auch Bewegungsrichtungen beinhalten.

Eine Erweiterung der SOM auf einen dreidimensionalen Eingaberaum setzt folgende grundlegende Änderungen voraus, die im zeitlichen Rahmen dieser Arbeit nicht getestet werden konnten.

Das Entfernungsmaß aus Gleichung 3.4 muss die Richtung als dritte Dimension der Eingabevektoren mit den bereits verwendeten Dimensionen (Position auf der x- und y-Achse) in Beziehung setzen und gegeneinander gewichten. Die in Kapitel 3.3 beschriebene Dichteverteilung wurde in Abschnitt 2.6.1 bereits für höhere Dimensionen formuliert. Die Flächenverteilung, die zu besseren Ergebnissen als die Dichteverteilung führt, muss auf drei Dimensionen erweitert werden. Alorithmen für die Skelettierung von dreidimensionalen Datensätzen sind unter Anderem in [WHH+03]

beschrieben. Die in Kapitel 3.2.2 dargestellten Modifikationen der Nachbarschaftsfunktion, sowie des Vergleichsmaßes ist im dreidimensionalen Fall problematisch, da die Richtung der Person eine gesonderte Rolle gegenüber der Position in der xy-Ebene einnimmt. Das Lernen von Geschwindigkeiten erscheint wenig sinnvoll, da Trajektorien in unterschiedlichen Geschwindigkeiten begangen werden können, ohne sich darüber hinaus zu unterscheiden. Um die in Abschnitt 3.2.2 und 3.2.3 beschriebenen Vorteile einer initial geordneten SOM weiterhin nutzen zu können, muss die Topologie der SOM auf drei Dimensionen erweitert werden.

Der oben skizzierte Weg könnte zu prototypischen dreidimensionalen Pfaden führen, die für die Vorhersage von Trajektorien verwendet werden könnten. Eine anschließende Beobachtung der Abbiegerichtungen wäre nicht mehr notwendig.

Weiterhin können die Ergebnisse von Trajektorienvorhersagen durch personifizierte Daten verbessert werden. Da jede Person eigenen Bewegungsmustern folgt, könnte die Generalisierung und die Vorhersage für jede Person einzeln durchgeführt werden. Dadurch würden die Bewegungsmuster der einzelnen Personen seperat gelernt werden. Die bestehende verteilte Architektur könnte dies ermöglichen, da auf die einzelnen Sensordaten und Zwischenergebnisse getrennt zugegriffen werden kann. Dies begünstigt eigene personenspezifische Generalisierungs- und Vorhersagemodule.

Ein weiterer wichtiger Schritt zur sinnvollen Mensch-Roboter Interaktion ist es, dem Roboter nicht nur Positionen von Personen bekannt zu machen, sondern auch deren Identitäten. Durch ein robustes Tracking, wie in Kapitel 2 vorgestellt, könnte eine einmalige Identifikation von Personen gegenüber dem System genutzt werden, um personifizierte Dienste zu beliebigen Zeitpunkten zu ermöglichen.

Literaturverzeichnis

[Ben81] V. E. Benes. Exact finit-dimensional filters with certain diffusion non linear drift. *Stochastics*, 5:65–92, 1981.

[Ben04] Maren Bennewitz. *Mobile Robot Navigation in Dynamic Environments*. PhD thesis, University of Freiburg, Department of Computer Science, 2004.

[BG04] A. Bruce and G. Gordon. Better motion prediction for people-tracking. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[BSLK01] Yaakov Bar-Shalom, X.-Rong Li, and Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc, 2001.

[Col03] R. Collins. Mean-shift blob tracking through scale space. In *Proceedings of the IEEEConference on Computer Vision and Pattern Recognition*, 2003.

[CRM00] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proceedings of the IEEE Conference on Computer Vision andPattern Recognition*, Hilton Head Island, South Carolina, 2000.

[FHM02] Ajo Fod, Andrew Howard, and Maja J. Mataric. Laser-based people tracking. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[Fok05] Amalia Foka. *Predictive Autonomous Robot Navigation*. PhD thesis, University of Crete, May 2005.

[FZ00] S. Feyrer and A. Zell. Robust real-time pursuit of persons with a mobile robot using multisensor fusion. In *Proceedings of the 6th International Conference on Intelligent Autonomous Systems (IAS-6)*, 2000.

[GSRL98] W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1998.

[GTC01] G. Grinstein, M. Trutschl, and U. Cvek. High-dimensional visualizations. In *Proceedings of the Visual Data Mining workshop, KDD'2001*, 2001.

[HCP00] C. Hue, J. Le Cadre, and P. Perez. Tracking multiple objects with particle filtering, 2000.

[Jaz70] A. H. Jazwinski. *Stochastic Processes and Filtering Theory*. New York: Academic Press, 1970.

[Kai67] T. Kailath. The divergence and bhattacharyya distance measure in signal selection. In *Proceedings of the IEEE International Conference on Trans. Commun. Tech. (COM-15)*, 1967.

[KHM+00] John Krumm, Steve Harris, Brian Meyers, Barry Brumitt, Michael Hale, and Steve Shafer. Multi-camera multi-person tracking for easyliving. In *Proceedings of the Third IEEE International Workshop on Visual Surveillance (VS'2000)*, Washington, DC, USA, 2000.

[KLF+02] M. Kleinehagenbrock, S. Lang, J. Fritsch, F. Lmker, G. A. Fink, and G. Sagerer. Person tracking with a mobile robot based on multi-modal anchoring. In *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication (ROMAN)*, pages 423–429, 2002.

[Koh95] T. Kohonen. *Self-Organizing Maps*. Springer- Verlag Berlin Heidelberg, 1995.

[LCL01] Jun S. Liu, Rong Chen, and Logvinenko. *A theoretical framework for sequential importance sampling and resampling*. Springer Verlag, 2001.

[MN90] M. Mohnhaupt and B. Neumann. Understanding object motion: Recognition, learning, and spatiotemporal reasoning. *Journal of Robotics and Autonomous Systems, North Holland*, 1990.

[MS03] R. Madhavan and C. Schlennov. Moving object prediction for off-road autonomous navigation. In *Proceedings of the SPIE Aerosense Conference*, 2003.

[NLK96] Y. S. Nam, B. H. Lee, and M. S. Kim. View time based moving obstacle avoidance using stochastic prediction of obstacle motion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1081–1086, Minneapolis, 1996.

[NZS+04] K. Nakamura, H. Zhao, R. Shibasaki, K. Sakamoto, T. Oga, and N. Suzukava. Tracking pedestrian by using multiple laser range scanners.

Technical report, Center for Spatial Information Science, University of Tokyo, 2004.

[Pap84] A. Papoulis. *Probability, Random Variables, and Stochastic Processes, 2nd ed.* New York: McGraw-Hill, 1984.

[RBF+00] N. Roy, G. Baltus, D. Fox, F. Gemperle, J. Goetz, T. Hirsch, D. Magaritis, M. Montemerlo, J. Pineau, J. Schulte, and S. Thrun. Towards personal service robots for the elderly. In *Proceedings of the Workshop on Interactive Robotics and Entertainment (WIRE)*, Pittsburgh, 2000.

[Rek03] Ioannis M. Rekleitis. *Cooperative Localization and Multi-Robot Exploration.* PhD thesis, School of Computer Science, McGill University, 2003.

[RSG04] B. Ristic, S.Arulampalam, and N. Gordon. *Beyond the Kalman Filter.* Artech House, 2004.

[Rus04] James Russel. *Detecting Humans in Video Footage using Multiple Classifiers.* PhD thesis, University of Western Australia, 2004.

[SBFC03] Dirk Schulz, Wolfram Burgard, Dieter Fox, and Armin B. Cremers. People tracking with a mobile robot using sample-based joint probabilistic data association filters. *International Journal of Robotics Research (IJRR)*, 2003.

[Sed02] R. Sedgewick. *Algorithms in C++.Graph Algorithms, 3rd ed.* Boston: Addison-Wesley, 2002.

[Sen02] A. W. Senior. Tracking with probabilistic appearance models. In *International Workshop on Performance Evaluation of Tracking and Surveillance Systems (ECCV)*, pages 48–55, June 2002.

[SHT+01] A. Senior, A. Hampapur, Y-L Tian, L. Brown, S. Pankanti, and R. Bolle. Appearance models for occlusion handling. In *Second International Workshop on Performance Evaluation of Tracking and Surveillance Systems (ECCV)*, 2001.

[SNM98] R. D. Schraft, J. Neugebauer, and C. Schaefferand T. May. *Care-O-bot: Ein technisches Hilfssystem fr unterstützungs- und pflegebedürftige Personen im häuslichen Bereich*, pages 234–244. Springer, 1998.

[TBB+99] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. MINERVA: A second generation mobile tour-guide robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

[TBF05] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, Massachusetts, London, England, 2005.

[THM+95] S. Tadokoro, M. Hayashi, Y. Manabe, Y. Nakami, and T. Takamori. On motion planning of mobile robots which coexist and cooperatewith human. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 518–523 vol.2, August 1995.

[TL86] Roger Y. Tsai and Reimar K. Lenz. Review of the two-stage camera calibration technique plus some new implementation tips and new techniques for center and scale calibration. Technical report, IBM T. J. Watson Research Center, Yorktown Heights NY 10598, 1986.

[Tsa85] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *Journal of the IEEE International Conference on Computer Vision and Pattern Recognition*, 1985.

[UM05] A. Ultsch and F. Moerchen. Esom-maps: tools for clustering, visualization, and classification with emergent som. Technical report, Data Bionics Research Group, University of Marburg, 2005.

[UMS00] K. Uchida, J. Miura, and Y. Shira. Tracking many pedestrians in the crowd. In *Proceedings if the IAPR Workshop on Machine Vision and its Applications*, pages 533–536, 2000.

[US02] L. Jain U. Seiffert. *Self-Organizing Neural Networkes*. Physica Verlag Heidelberg New York, 2002.

[VF04] A. D. Vasquez and Th. Fraichard. Motion prediction for moving objects: a statistical approach. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3931–3936, New Orleans, LA (US), April 2004.

[WB95] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, 1995.

[WHH+03] Thomas Wittenberg, Peter Hastreiter, Ulrich Hoppe, Heinz Handels, Alexander Horsch, and Hans-Peter Meinzer. Bildverarbeitung für die Medizin, Algorithmen - Systeme - Anwendungen. In *Proceedings of the CEUR Workshop*, volume 80, March 2003.

[WSS+04] Daniel Westhoff, Hagen Stanek, Torsten Scherer, Jianwei Zhang, and Alois Knoll. A flexible framework for task-oriented programming of service robots. In *Robotik 2004, VDI/VDE-Gesellschaft fr Mess- und Automatisierungstechnik, VDI-Berichte (ISBN 3-18-091841-1)*, Munich, Germany, 2004.

[WSSK04] Daniel Westhoff, Hagen Stanek, Torten Scherer, and Alois Knoll. Mobile manipulatoren und ihre aufgabenorientierte programmierung. In *atp - Automatisierungstechnische Praxis 10/2004, Oldenbourg Industrieverlag GmbH, ISSN 0178-2320*, Munich, Germany, 2004.

[YY98] N. H. C. Yung and C. Ye. Avoidance of moving obstacles through behavior fusion and motion prediction. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 3424–3429, San Diego, 1998.

[ZS03] H. Zhao and R. Shibasaki. Pedestrian tracking using multiple laser range scanners. In *Proceedings of Computers on Urban Planning and Urban Management*, Sendai, Japan, May 2003.

[ZSI03] H. Zhao, R. Shibasaki, and N. Ishihara. Pedestrian tracking using single-row laser range scanners. In *Proceedings of Computers on Urban Planning and Urban Management*, 2003.

Appendix A

A.1 Roblet®-Technologie

Die Roblet®-Technologie ist ein in Java™ geschriebenes Rahmenwerk, das die aufgabenorientierte Entwicklung verteilter Systeme vereinfacht. Dieses Rahmenwerk erlaubt es dem Programmierer Programmteile, die Roblets®genannt werden, an einen Roblet®-Server zu schicken, der auf einem entfernten Rechner läuft. Ein Roblet kann dabei Daten und ausführbaren Code beinhalten. Der Roblet®-Server kann neben der grundlegenden Java™-Umgebung spezielle Funktionen, sogenannte *Einheiten*, bereitstellen, die im Kontext der Robotik zur Hardwaresteuerung genutzt werden können.

Um die Nutzbarkeit zu erleichtern, wird das Netzwerk durch Kapselung vor dem Programmierer versteckt. Treten in einem entfernt ablaufenden Roblet®Ausnahmen auf, werden diese automatisch dem lokalen System mitgeteilt. Das Arbeiten auf einem entfernten Roboter ist aus Sicht des Programmierers ähnlich der Nutzung lokaler Dienste. Für eine detaillierte Beschreibung siehe [WSS+04] oder auch [WSSK04].

A.2 Hardware

Im Rahmen dieser Arbeit wurden Laserscanner und eine Kamera zum Erkennen und Verfolgen von Personen verwendet. Die Laserscanner sind fester Bestandteil des zur Verfügung stehenden Serviceroboters und werden normalerweise für dessen Selbstlokalisierung verwendet. Um die Positionen der erkannten Personen von Scannerkoordinaten in Weltkoordinaten zu transformieren, muss die Position des Laserscanners bekannt sein. Die Schnittstelle zur Positionierung und Abfrage der Position des Roboters ist in einer Einheit eines Roblet®-Server (siehe A.1) definiert.

Die folgenden Abschnitte beschreiben die verwendeten Sensoren sowie die Roboterplattform.

A.2.1 Die Roboterplattform

Der für diese Arbeit verwendete Roboter ist eine modifizierte Version des MP-L655 der Firma NEOBOTIX[1]. Er besteht aus einer mobilen Plattform mit einem Differenzialantrieb und Rad-Encodern, die 4096 Schritte pro Motordrehung und Rad messen. Zur Manipulation der Umgebung wurde das System mit einem Mitsubishi Heavy Industries PA10-6C Manipulator[2] und einer Dreifingerhand von Barrett Technologies®Inc.[3] ausgestattet. Um visuell gesteuertes Greifen zu ermöglichen, ist an dem Manipulator zusätzlich eine analoge Handkamera[4] angebracht. Weiterhin ist die Plattform mit einen Stereo-Kamerakopf bestehend aus einer Pan-Tilt-Einheit[5] und zwei Sony DFW-VL500[6] Firewire-Digitalkameras mit 12x Zoom, sowie einem Omnivisionsystem mit Sony DFW-SX900[7] Firewire-Digitalkamera und hyperbolidem Spiegel[8] ausgestattet. Die in dieser Arbeit verwendeten zwei zwei SICK Lasermesssystemen[9] sind ebenfalls an dem Roboter angebracht. Zur Steuerung der Motoren und Erfassung der Daten ist ein Pentium IV PC in die Plattform integriert. Das Robotersystem in dem hier beschriebenen Aufbau ist in Abbildung A.1 zu sehen.

A.2.2 Laserscanner

In diesem Projekt wurden zum Verfolgen von Personen Lasermesssysteme LMS 200 der Firma SICK verwendet. Sie messen die Abstände von Objekten durch time-of-flight Messungen des ausgesandten Laserstrahls. Vorteile von Lasermesssystemen sind hohe Genauigkeit, geringes Datenvolumen und ein großer Messradius.

Die zwei hier verwendeten Lasermesssysteme (siehe Abb. A.2) messen Entfernungen in einem Radius von 180 Grad bei einer Auflösung von 10mm. Die maximale Reichweite beträgt 80m, verwendet wurde jedoch nur eine Reichweite von 8m. Der statistische Fehler ist mit $\sigma = 5mm$ und der systematische Fehler mit $+/-15mm$ angegeben. Der hier verwendete Modus misst in 0.5 Grad Schritten mit einer Frequenz von ca. 30Hz..

[1] http://www.neobotix.de/en/
[2] http://www.mhi.co.jp/kobe/mhikobe/products/mechatronic/index.html
[3] http://www.barretttechnology.com/robot/index.htm
[4] http://www.jai.com/camera/products_show.asp/id=8&sprog=uk#221
[5] http://www.dperception.com/products.html
[6] http://www.sony.net/Products/ISP/products/interface/DFWV500.html
[7] http://www.sony.net/Products/ISP/products/interface/DFWSX.html
[8] http://www.accowle.com/english/index.html
[9] http://ecatalog.sick.com/Products/ProductFinder/product.aspx?finder=Produktfinder&pid=9168&lang=en

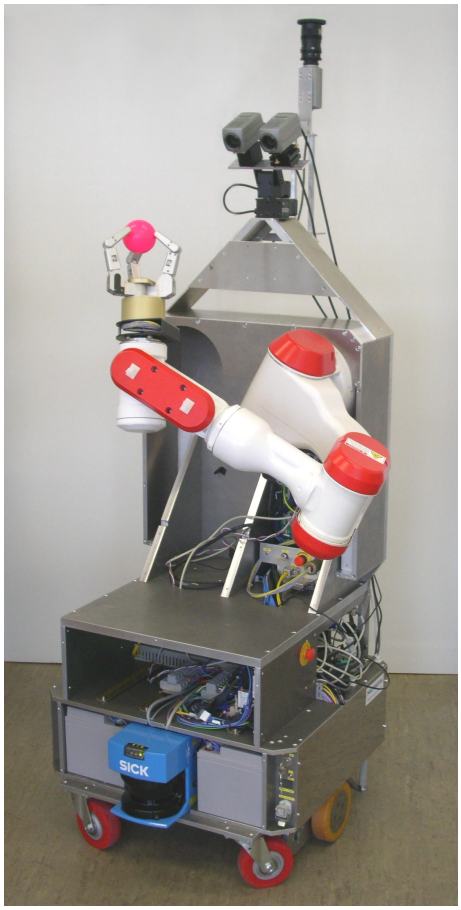


Abbildung A.1: Der in dieser Arbeit verwendete Serviceroboter des AB TAMS der Universität Hamburg.

Abbildung A.2: Sick Laserscanner LMS 200

A.2.3 Kamera

Für dieses Projekt wurde ein Trakingalgorithmus implementiert, der auf Kamerabildern basiert. Kameras sind für Trackingalgorithmen nur bedingt geeignet, da sie bei niedriger Auflösung nur eine geringe Genauigkeit bieten und bei höherer Auflösung sehr rechenintensiv sind. Vorteile hingegen sind geringe Kosten und große Flexibilität hinsichtlich der Algorithmen und Einsatzgebiete.

Die für diese Arbeit verwendete Kamera vom Typ WV-GPR464 der Firma Panasonic[10] ist in Abbildung A.3 zu sehen. Sie ist ca. 20cm unterhalb der Decke fest angebracht, um einen möglichst steilen Winkel auf die Personen im Raum zu erreichen. Durch die feste Montage ist eine Kalibrierung der Kameraparameter nur einmalig nötig (siehe dazu Kapitel 2.6.2). Diese Kamera wird bei einer Auflösung von 768 x 575 Pixeln betrieben. In Verbindung mit einer Grabberkarte vom Typ Meteor-II/Digital der Firma Matrox[11] wird eine Framerate von ca. 25Hz erreicht.

Abbildung A.3: Panasonic Camera WV-GPR464

A.3 Schnittstelle der Trackingalgorithmen

Das in Kapitel 2 vorgestellte Trackingmodul stellt den konzeptionellen Rahmen dar, der beliebige Trackingalgorithmen in ein Gesamtsystem integriert. Um eine flexibele Erweiterung der Algorithmen zu ermöglichen, wurde eine Schnittstelle entwickelt, die die Funktionsweise der Algorithmen möglichst wenig einschränkt, andererseits aber umfangreichen Datenaustausch zulässt.

Sowohl die Vorverarbeitung, die aus Sensordaten erste Positionsschätzungen berechnet, als auch abstraktere multimodale Filter, berechnen Annahmen über Position

[10] www.panasonic.com
[11] www.matrox.com

und Bewegung von Personen. Die einheitliche Schnittstelle *Tracker* stellt dementsprechend eine beliebig lange Liste von sogenannten *Tracks* bereit, die die eigentlichen Informationen über einzelne Personen beinhalten.

Ein *Track* beinhaltet:

- eine eindeutige Identifikationsnummer
- einen Zeitstempel
- die momentane Position x, y in einem zweidimensionalen Koordinatensystem[12]
- die Änderung der Position über die Zeit $\delta x, \delta y$
- die geschätzte Standardabweichung der Position σ_x, σ_y sowie die der Positionsänderung $\sigma_{\delta x}, \sigma_{\delta y}$

Die Verwendung von nur zwei Dimensionen folgt aus der Annahme, dass sich Personen nur auf der Grundebene bewegen. Für eine Nutzung in Umgebungen mit mehreren Stockwerken kann die Schnittstelle um eine dritte Dimension erweitert werden.

[12] Das Koordinatensystem ist frei wählbar. Denkbar sind unter Anderem Bildkoordinaten, Kamerakoordinaten oder Weltkoordinaten.

Danksagung

Ich möchte mich bei allen Mitarbeitern AB TAMS bedanken, die mich während der Erstellung dieser Diplomarbeit unterstützt haben und mir den Start in das wissenschafftliche Arbeiten so leicht wie möglich gemacht haben. Besonderer Dank gilt Daniel Westhoff, der mir gerade zu Anfang unzählige Tips gegeben hat und Markus Hüser für seine Arbeit an der verwendeten Bildverarbeitungsoftware.

Weiterhin möchte ich mich bei meinen Betreuern bedanken: Jianwei Zhang hat mir, dadurch dass er mir bei der Gestaltung des Themas weitgehend freie Hand lies, großes Vertrauen entgegengebracht und Bernd Neumann, der in konstruktiven Diskussionen neue Ideen und Lösungsansätze anregte, die für die Entwicklung dieser Arbeit unverzichtbar waren.

Ich bedanke mich auch bei all meinen Freunden und Kommilitonen, die mir, trotzdem ich sie in den letzten Wochen sehr vernachlässigt habe, Verständnis und Motivation entgegengebracht haben. Der gröste Dank gilt aber meinen Eltern, die mir den reibungslosen Ablauf meines Studiums ermöglichten. Nicht zuletzt ist es ihnen zu verdanken, den Abschluss meines Studiums erreicht zu haben.

Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Ich bin mit einer Einstellung in den Bestand der Bibliothek des Fachbereichs einverstanden.

(Ort, Datum) (Unterschrift)