Technische Universität Ilmenau
Fakultät für Informatik und Automatisierung
Fachgebiet Neuroinformatik und Kognitive Robotik

Prädiktion von Personentrajektorien auf einer mobilen Roboterplattform

Masterarbeit zur Erlangung des akademischen Grades Master of Science

Nikolas Dahn

Betreuer: M.Sc. Thanh Quang Trinh
M.Sc. Tim Wengefeld
Verantwortlicher Hochschullehrer:
Univ.-Prof. Dr.-Ing. H.-M. Groß, FG Neuroinformatik und Kognitive Robotik

Die Masterarbeit wurde am 18.11.2016 bei der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau eingereicht.

Erklärung: „Hiermit versichere ich, dass ich diese Masterarbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe. Alle von mir aus anderen Veröffentlichungen übernommenen Passagen sind als solche gekennzeichnet.“

Ilmenau, 18.11.2016 .....

Nikolas Dahn

Inhaltsverzeichnis

Kapitel 1

Einleitung

Autonom navigierende Systeme haben in den letzten Jahren viel Aufmerksamkeit erhalten. Die Zeiten, in denen ein KUKA-Montageroboter nur mit Sicherheitsabstand betrieben werden konnte, sind vorbei. Ob im Straßenverkehr als selbstfahrendes Auto, im Baumarkt als Ratgeber und Wegweiser oder im häuslichen und medizinischen Bereich als Begleiter und Unterstützer, Roboter rücken näher an den Menschen heran. Um die Personensicherheit zu gewährleisten und die gesellschaftliche Akzeptanz zu steigern, dürfen diese Systeme Personen in ihrer Umgebung nicht ignorieren. Es gilt, Sicherheitsabstände und Verhaltensregeln einzuhalten und dennoch die erwarteten Aufgaben zu erfüllen. Einen wichtigen Beitrag hierzu leisten Algorithmen, welche vorhersagen, wann und wo sich Personen aufhalten werden.

1.1 Motivation

Die an der Technischen Universität Ilmenau vom Fachgebiet Neuroinformatik entwickelten Assistenzroboter sollen mit Menschen in sehr unterschiedlichen Umgebungen interagieren; geplant sind u.A. der Einsatz in Krankenhäusern, Baumärkten und den eigenen vier Wänden. Durch den engen Kontakt zum Anwender und weiteren, nicht direkt beteiligten Personen muss das Verhalten des Roboters für Menschen interpretierbar sein. Das Ziel ist eine höfliches, nutzerorientiertes und dennoch zielführendes Navigationsverhalten. Dafür ist es notwendig, die Bewegung der Personen

zu antizipieren und die eigene Navigation frühzeitig an der Prädiktion auszurichten, um Personen nicht über die Intention des Roboters zu irritieren.

Zur Vorhersage von Personentrajektorien existiert bereits ein Algorithmus am Fachgebiet von K. Schenk 2017, welcher in der Lage ist, typische Bewegungsmuster aus Beobachtungen zu erlernen. Die Unsicherheit und Variabilität werden dabei probabilistisch modelliert. Das Verfahren wurde allerdings für ortsgebundene Trackingsysteme konzipiert, der Einsatz auf einer mobilen Roboterplattform erfordert dementsprechend eine genauere Untersuchung. Ziel der Arbeit ist es, ausgehend von diesem Verfahren eine Implementierung auf einem der Fachgebietsroboter zu realisieren und zu evaluieren.

1.2 Themenbeschreibung

Folgende Ziele wurden verfolgt:

- Aufarbeitung und Analyse des Quellcodes,
- Übertragung des Algorithmus in das MIRA-Framework,
- Anpassung des Verfahrens an den Einsatz auf mobilen Plattformen,
- Reduktion der zur Initialisierung benötigten Datenmenge,
- Testen der Implementierung auf einem der Fachgebiets-Roboter und
- Ausführliche Dokumentation der Implementierung.

Das Endziel dieser Arbeit ist somit nicht die soziale Navigation an sich, sondern die Implementierung eines wichtigen Moduls, auf welchem diese aufbaut. Herausforderungen entstehen unter Anderem durch die begrenzten Ressourcen des Roboters sowie die größere Unsicherheit der Beobachtungen. Im besten Fall steht am Ende eine Kostenkarte, welche ein Roboter nutzen kann, um Kollisionen mit Personen in Bewegung zu vermeiden.

1.3 Übersicht

Kapitel 2 gibt einen Überblick zum Stand der Technik und geht auf aktuelle Herausforderungen bei der Prädiktion von Trajektorien ein. In Kapitel 3 wird anschließend der Algorithmus von K. Schenk 2017 ausführlich erläutert und mit Beispielen ergänzt. Die eigene Implementierung sowie Erweiterungen und Besonderheiten werden in Kapitel 4 vorgestellt. Die Implementierung wird in Kapitel 5 auf mehreren Datensätzen evaluiert. In diesem Kapitel wird außerdem auf besondere Probleme beim Einsatz von Prädiktionsalgorithmen auf Robotern eingegangen. Kapitel 6 fasst noch einmal die wichtigsten Ergebnisse zusammen, gibt Anregungen für weitere Entwicklungen und schließt mit einem Fazit ab.

Kapitel 2

State of the Art

In diesem Kapitel soll zunächst eine mögliche Einteilung der Algorithmen zur Prädiktion von Trajektorien gegeben werden. Dabei werden die am häufigsten verwendeten Algorithmen kurz beschrieben und die üblichen mit diesen Kategorien verbundenen Vor- und Nachteile benannt. Im Anschluss wird der in dieser Arbeit verwendete Algorithmus beschrieben und eingeordnet.

2.1 Einteilung

Lefèvre, Vasquez und Laugier 2014 schlägt eine Einteilung von Prädiktionsalgorithmen in vier Kategorien vor:

- Physikalisch,
- Umgebungsbasiert,
- Verhaltensbasiert, und
- Interaktionsbasiert.

Diese werden im folgenden kurz näher beschrieben. Zudem wird für jede Kategorie ein typisches Verfahren exemplarisch beschrieben. Eine Übersicht und weitere Unterteilung findet sich in Abbildung 2.1.

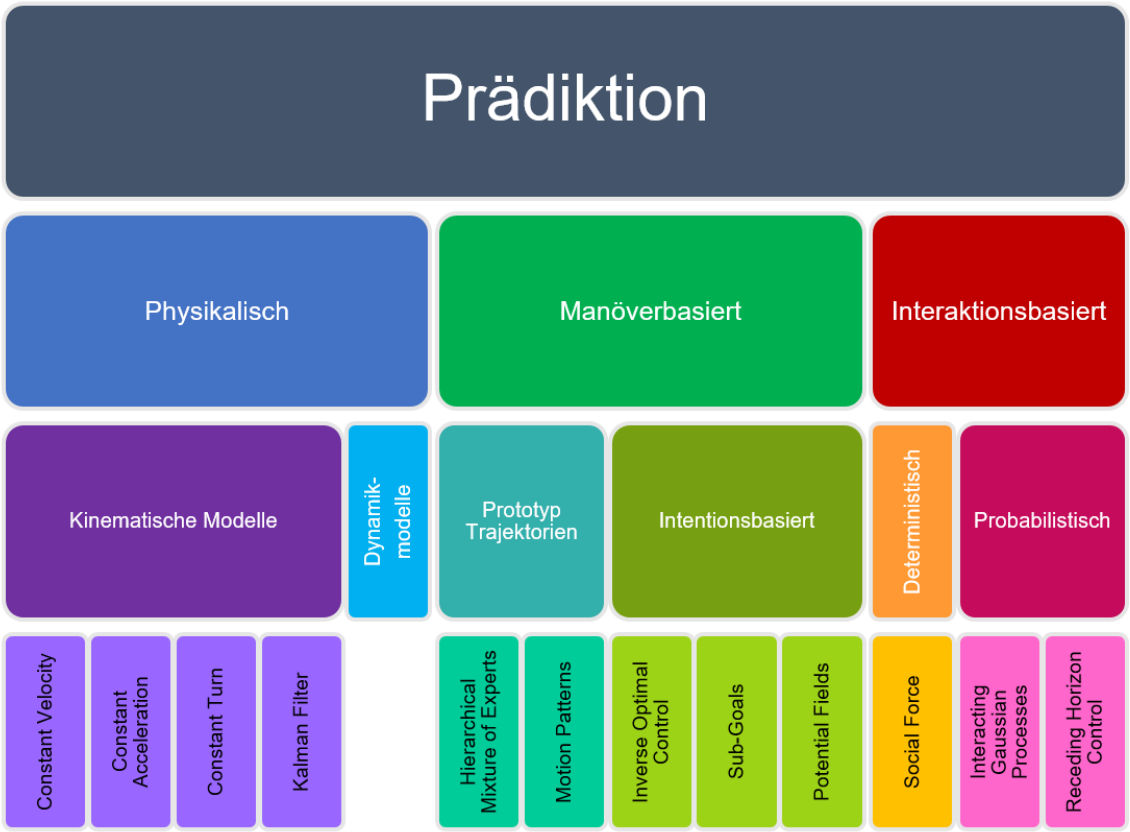


Abbildung 2.1: Einteilung von Algorithmen zur Prädiktion von Trajektorien
In der untersten Reihe finden sich einige ausgewählte Algorithmen, die in der Praxis Anwendung finden. Physikalische Dynamikmodelle sind zwar theoretisch möglich, scheitern jedoch praktisch an zu vielen unbekannten Variablen.

2.1.1 Physikalische Modelle

Dies sind in der Regel die einfachsten Verfahren und finden vor allem im Straßenverkehr bei der Prädiktion anderer Fahrzeuge Verwendung. Als Grundlage dienen meistens newtonsche Bewegungsgleichungen, die Prädiktion erfolgt z.B. mit Hilfe von Kalman-Filtern. Verfahren, bei denen die Bewegung durch Fluidsimulationen angenähert werden, sind ebenfalls denkbar, scheinen jedoch in der Praxis nur wenig erforscht zu sein (siehe z.B. Burgess und Darken 2004). Physikalische Modelle gehen meist mit einem sehr kurzen Prädiktionshorizont (Größenordnung 2s) und geringer Verallgemeinerungsfähigkeit einher. Für Ansätze dieser Kategorie spricht, dass sie meist auf weniger oder gar keine Trainingsdaten angewiesen sind.

***Physikalisch*: Kalman-Filter mit Constant Velocity**

Kalman-Filter stellen eine einfache Möglichkeit dar, um fehlerbehaftete Messwerte zu korrigieren. Voraussetzung ist, dass der Messfehler und das zugrunde liegende mathematische Modell bekannt sind. Für die Prädiktion einer Messgröße können zeitabhängige Modelle verwendet werden, die Differenz von prädizierter und tatsächlicher Messgröße wird zur Korrektur der Schätzung verwendet. Zur Prädiktion von Trajektorien kann im einfachsten Fall angenommen werden, dass Geschwindigkeit und Richtung konstant bleiben. Dieses Modell wird in der Literatur auch als Constant Velocity Modell bezeichnet (Meuter u. a. 2008). Die typische Struktur eines Filters, welcher dieses Modell verwendet, ist in Abbildung 2.2 gezeigt.

2.1.2 Umgebungsbasierte Modelle

Umgebungsbasierte Modelle prädizieren Trajektorien anhand von Umgebungsmerkmalen. Typische Vertreter lernen die ortsgebundenen Übergangswahrscheinlichkeiten und verwenden häufig (Hidden-)Markov-Modelle, um Vorhersagen zu treffen. Sie zeichnen sich i.d.R. durch einen weiten Prädiktionshorizont (Größenordnung 20-40s) aus, benötigen jedoch viele Trainingsdaten und sind nicht ohne weiteres auf andere Umgebungen übertragbar.

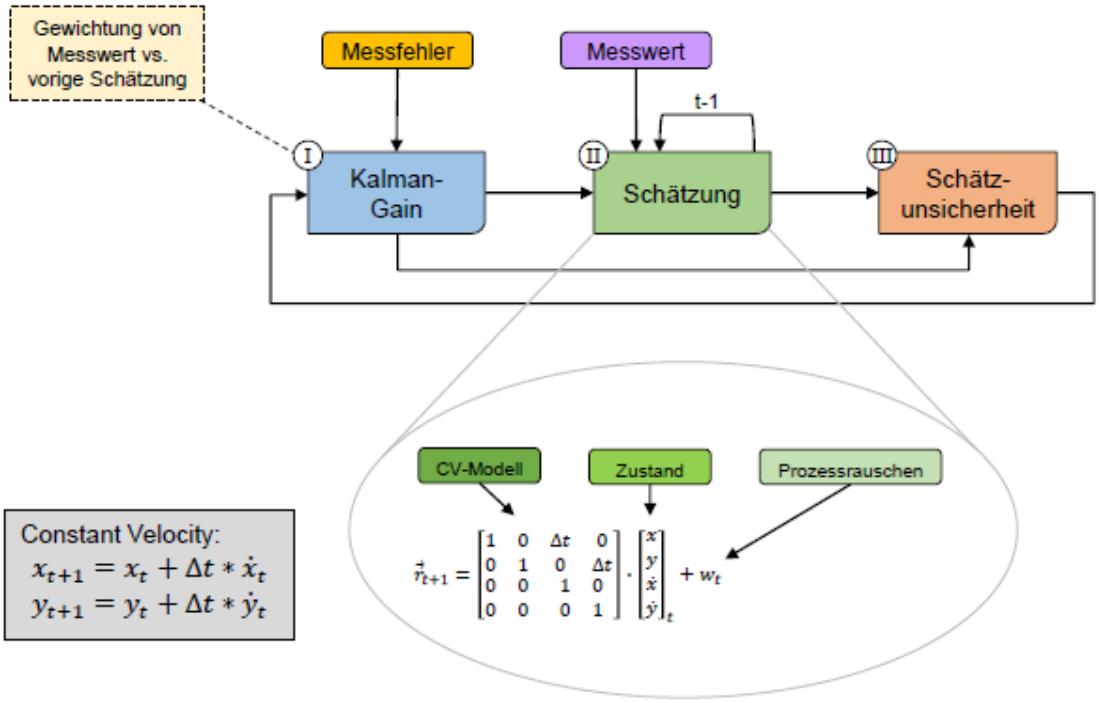


Abbildung 2.2: Struktur eines Kalman-Filters mit Constant Velocity

Das Constant Velocity Modell nimmt eine konstante Bewegungsgeschwindigkeit und Richtung an. Die Schätzung des Filters wird durch neue Beobachtungen und den Vergleich mit der vorigen Schätzung korrigiert.

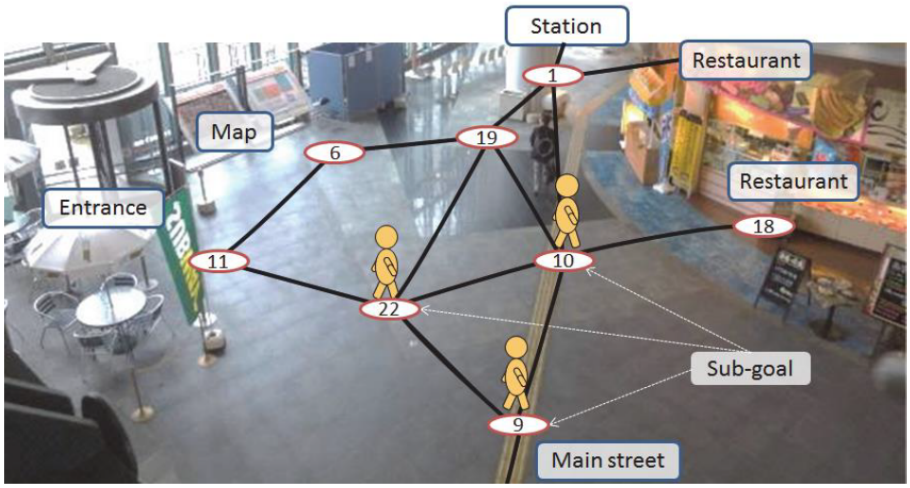


Abbildung 2.3: Veranschaulichung des Verfahrens von Ikeda u. a. 2012
Zu sehen sind mehrere mögliche „Points of Interest“ und weitere „Sub-Goals“, durch welche sich Personen bewegen, um zu den Zielpunkten zu gelangen.

***Umgebungsbasiert*: Sub-Goals**

Das Verfahren von Ikeda u. a. 2012 teilt die Karte in Zellen ein und lernt für jede Zelle typische Stopppunkte und Richtungsänderungen. Des weiteren werden mögliche Interessepunkte identifiziert. Durch die Berechnung der bedingten Wahrscheinlichkeiten kann die zukünftige Trajektorie abgeschätzt werden. Eine veranschaulichende Grafik ist in Abbildung 2.3 zu finden.

2.1.3 Verhaltensbasierte Modelle

Diese Modelle bilden das Verhalten der vorherzusagenden Aktoren ab (i.d.R. Personen). Das Verhalten kann von Experten als Regelsystem modelliert oder aus Beobachtungen gelernt werden. Die Komplexität menschlichen Verhaltens hat es bislang verhindert, ein allgemeingültiges, übertragbares und interpretierbares Verhaltensmodell abzuleiten, sodass in dieser Kategorie viele unterschiedliche Ansätze aktiv verfolgt werden. Diese Verfahren hängen stark von der Beschreibung des beobachteten Verhaltens ab.

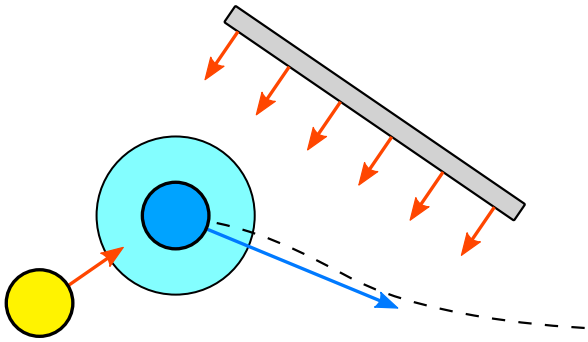


Abbildung 2.4: Veranschaulichung des Social Force Modells
Die Einflüsse der Umgebung auf die Bewegung einer Person wird durch treibende oder anziehende (blau) und abstoßende Kräfte (rot) modelliert. Ein möglicher zukünftiger Pfad ist als gestrichelte Linie eingezeichnet.

***Verhaltensbasiert*: Social Force**

Social Force wurde erstmals von Helbing und Molnár 1995 beschrieben und bezeichnet den phänomenologischen Ansatz, anziehende und abstoßende Kräfte in der Umgebung zu modellieren. Auf der Grundlage dieser Kräfte wird versucht, das zukünftige Verhalten einer einzelnen Person, z.B. ein Abweichen vom Weg oder eine Beschleunigung, vorherzusagen. Der Ansatz wird auch in Abbildung 2.4 veranschaulicht.

2.1.4 Interaktionsbasierte Modelle

Aufbauend auf den verhaltensbasierten Ansätzen versuchen diese Modelle, auch die Interaktion zwischen Aktoren mit zu berücksichtigen. So kann z.B. die Bewegung von Personen in und durch Menschenmengen besser prädiziert werden. Im Unterschied zu verhaltensbasierten Modellen müssen bei diesen Verfahren für die Prädiktionen mehrere Aktoren gleichzeitig berücksichtigt werden. Die Schwierigkeit bei der Beschreibung entsprechender Modelle sorgt dafür, dass bislang nur wenige Ansätze verfolgt werden, bei denen Interaktionen explizit mitberücksichtigt werden.

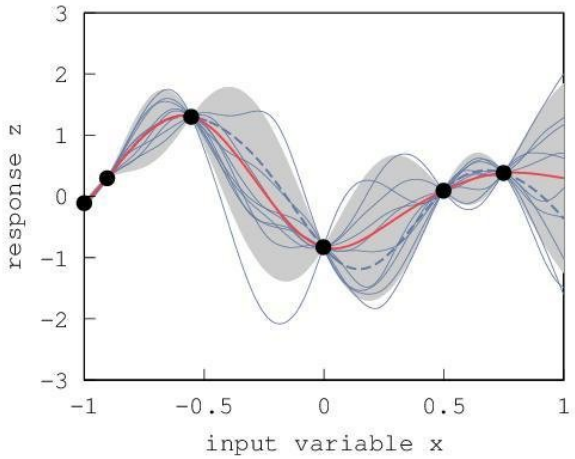


Abbildung 2.5: Veranschaulichung eines Gaussian Process
Messwerte (schwarze Punkte) dienen als Stützstellen, um die möglichen Funktionen weiter einzuschränken. Einige mögliche Realisierungen sind in blau dargestellt, die mittlere Funktion ist rot eingezeichnet. Das 2σ-Konfidenzintervall ist grau hinterlegt.

Quelle: STK Toolbox `https://sourceforge.net/projects/kriging/`

***Interaktionsbasiert*: Gaussian Processes**

Trautman und Krause 2010 beschreibt ein Verfahren, bei dem zur interaktionsbasierten Prädiktion von Personentrajektorien Gaußprozesse eingesetzt werden. Gaußprozesse betrachten Messwerte als Stützstellen einer Funktionsschar und können auch in hochdimensionalen Räumen eingesetzt werden. Je mehr Beobachtungen gemacht werden, desto stärker werden die möglichen Funktionen eingeschränkt. Abbildung 2.5 veranschaulicht das Prinzip.

2.2 Anwendungsgebiete

In diesem Abschnitt wird eine mit Beispielen ergänzte Übersicht über einige Domänen gegeben, in welchen Algorithmen zur Trajektorienprädiktion Verwendung finden.

2.2.1 Automobilindustrie

Prädiktionssysteme werden vor allem zur Erhöhung der Sicherheit in Autos verbaut. Dabei gibt es unterschiedliche Ansätze: einerseits zur Erkennung von Personen, andererseits auch zur Vorhersage der Trajektorien anderer Verkehrsteilnehmer. Systeme zur Personenerkennung sind bereits seit 2008 serienmäßig in diversen Modellen verfügbar (etwa im Toyota Crown Hybrid). Durch die Vorhersage der künftigen Bewegung lassen sich weitere Gefahrensituationen abdecken, z.B. wenn Personen von der Seite auf die Straße laufen. Entsprechende Algorithmen können den Fahrer warnen und Notbremsungen durchführen und sind für die aktuelle Forschung an autonomen Fahrzeugen besonders relevant.

2.2.2 Überwachungssysteme

Zur Erhöhung der Sicherheit an z.B. Flughäfen werden bereits verschiedene Trackingsysteme eingesetzt, welche nicht nur Personen verfolgen, sondern auch beim Wechsel zwischen Kameras oder nach vorübergehender Verdeckung wiedererkennen. Das frühzeitige Erkennen von Zielen auffälliger Personen kann in solchen Szenarien entscheidend sein. Ein weiteres Beispiel stellt die Überwachung von Straßenkreuzungen dar. So werden in Salas u. a. 2007 Falschfahrer z.B. daran erkannt, dass die anhand der extrahierten Merkmale erwarteten Trajektorie nicht mit der Beobachtung übereinstimmt (etwa *Blinker links* $\rightarrow$ *links abbiegen*).

2.2.3 Robotik

Mit der Zunahme der Autonomie von Robotern erhält auch die Trajektorienprädiktion mehr Bedeutung. Viele der vorgesehenen Einsatzgebiete weisen dynamische Hindernisse auf, z.B. Personen, Haustiere, andere Roboter oder auch Fahrzeuge. Ein Roboter, welcher nur spontan um solche Hindernisse herumfährt, erfüllt zwar sein Ziel, bleibt jedoch für außenstehende Beobachter nur schlecht interpretierbar und unterstreicht zudem seine Andersartigkeit, was wiederum die Akzeptanz senkt. Um den Roboter zur langfristigen Planung und somit zu besser interpretierbarem Verhalten zu befähi-

gen, sind daher Algorithmen erforderlich, welche Vorhersagen darüber treffen, wann und wo mit Hindernissen zu rechnen ist.

Kapitel 3

Beschreibung des Algorithmus

Ziel des in K. Schenk 2017 beschriebenen und hier verwendeten Verfahrens ist es, für eine Trajektorie $T = (p_0, p_1, \ldots p_r)$ die Wahrscheinlichkeiten zukünftiger Aufenthaltsorte $p(x, y, t)$ vorherzusagen. Der Algorithmus wurde ursprünglich für fest installierte Beobachtungssysteme entwickelt und kann aufgrund des ortsgebundenen Charakters den umgebungsbasierten Modellen zugeordnet werden. Der Algorithmus basiert auf der Betrachtung der Okkupanz (Aufenthaltswahrscheinlichkeit) und wie sich diese über die Zeit im Raum verteilt. Dazu wird zunächst eine räumliche Abtastung möglicher Aufenthaltsorte erstellt und in einen topologischen Graphen überführt. Beobachtete Trajektorien können als Folge von Graphknoten abgebildet werden, indem für jeden Punkt der Trajektorie der am nächsten liegende Graphknoten zugeordnet wird. Diese nicht zwangsläufig äquidistante Abtastung wird hier auch als Projektion bezeichnet. Für jeden Knoten des Graphen wird ein weiterer nicht-topologischer Graph erzeugt, in welchem Teile der Projektion effizient abgebildet werden können. Mit Hilfe dieser sogenannten Markov-Trees können nach ausreichendem Training die Übergangswahrscheinlichkeiten zwischen den Knoten des Graphen abgeschätzt und die Okkupanz entsprechend verteilt werden. Indem die Transitionszeiten zwischen Knoten mitgelernt werden, kann die Okkupanz auch über die Zeit verteilt werden.
Im Folgenden werden die einzelnen Elemente des Algorithmus und ihr Zusammenspiel ausführlich beschrieben.

3.1 Topologischer Graph

Der Algorithmus legt einen topologischen Graphen zu Grunde, welcher eine räumliche Abtastung möglicher Aufenthaltsorte von Personen in der Umgebung darstellt. Gleichzeitig wird, wie im Folgenden Abschnitt 3.2 beschrieben, durch den Graphen eine Diskretisierung (und somit Bündelung) von Trajektorien erzielt. Die Abtastung muss nicht äquidistant sein und kann z.B. von einem Clusterer erzeugt werden. In K. Schenk 2017 wird eine Variante des Mean-Shift-Algorithmus verwendet, welcher gegenüber z.B. k-means den Vorteil hat, dass die Anzahl der Cluster nicht vorgegeben werden muss. Nachteilig ist jedoch der hohe Rechenaufwand von $O(In^2)$, da pro Iteration I jeder Datenpunkt mit jedem anderen verglichen werden muss Comaniciu, Ramesh und Meer o.D. Große Datensätze von Trajektorien können mehrere Millionen Trajektorien und somit leicht einige hundert Millionen Punkte enthalten. Obwohl schnellere Varianten mit vereinfachenden Annahmen existieren, bleibt der Rechenaufwand im Allgemeinen hoch. Da der Graph jedoch nur einmal in der Initialisierungsphase erstellt werden muss, ist dies unter Umständen vertretbar.

Einige Cluster-Verfahren liefern keine Kanten bzw. Nachbarschaften, diese müssen daher in einem Folgeschritt erstellt werden. Bei der Auswahl eines Verfahrens zur Erzeugung der Kanten ist es wichtig, dass die Kanten mögliche Übergänge zwischen Aufenthaltsorten widerspiegeln und beispielsweise keine Knoten überspringen. K. Schenk 2017 verwendet hierfür die Delauney-Triangulation, allerdings sind auch andere Verfahren wie z.B. Nearest-Neighbour-Algorithmen denkbar. Ist eine Hinderniskarte vorhanden, kann der Graph zudem vereinfacht werden, indem die Sichtlinien zwischen Punkten berücksichtigt werden. Diese können effizient mit dem Bresenham- oder einem Raycasting-Algorithmus überprüft werden.
Im Rahmen dieser Arbeit wurden mehrere Varianten zur Erzeugung von Knoten und Kanten implementiert. Diese werden im Kapitel 4 vorgestellt.

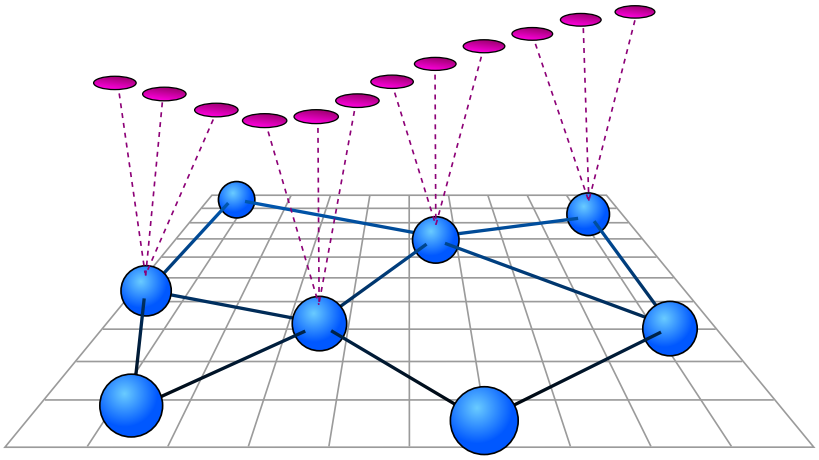


Abbildung 3.1: Projektion einer Trajektorie
lila: projizierte Trajektorie. blau: Knoten des Graphen

3.2 Projektion von Trajektorien

Trajektorien können einer Folge von Knoten des topologischen Graphen zugeordnet werden, indem für jeden Punkt der Trajektorie der nächste Knoten bestimmt wird. Die entstehende Folge von Knoten wird hier auch als Projektion bezeichnet. Durch den topologischen Graphen werden kleine Unterschiede zwischen Trajektorien unerheblich, sodass für ähnliche Trajektorien die gleiche Projektion entsteht. Ein Beispiel einer Projektion ist in Abbildung 3.1 visualisiert.

In dieser Arbeit wird für (Teile von) Projektionen die Notation $(Z|X_{T-\tau}X_{T-\tau+1}\dots X_T)$ verwendet, wobei X_T den letzten Knoten des betrachteten Abschnitts und Z den unmittelbar auf diesen folgenden Knoten repräsentiert.
Diese Abtastung der Trajektorien hat zur Folge, dass sich der topologische Graph nach der Initialisierung nicht mehr ändern darf. Eine Verschiebung einzelner Knoten würde ab einem gewissen Grad die Projektion verzerren und somit die zuvor gelernten räumlichen Zusammenhänge ungültig machen.

3.3 Markov-Trees

Markov-Trees stellen eine Möglichkeit dar, einfache Markov-Ketten zu bündeln. Voraussetzung ist, dass die zu bündelnden Ketten kreisfrei sind (d.h. keine Transition überspringt einen benachbarten Knoten) und mindestens der letzte zusammenzufassende Zustand bei allen Ketten gleich ist. Die Ketten werden nun zu einem Baum geformt, indem vom Ende beginnend gleiche Zustände zusammengefasst werden. Unterschiedliche Zustände werden zu getrennten Ästen, wobei jeder Ast nun getrennt weiterbehandelt wird. Eine Wiedervereinigung ist nicht vorgesehen. Durch dieses Vorgehen entsteht ein zusammenhängender, kreisfreier Graph, welcher durch die Definition eines Endknotens als Baum aufgefasst werden kann. Die Bezeichnung als Baum trifft insbesondere dann zu, wenn es sich bei den Markov-Ketten um Kausalketten handelt, also ausschließlich Übergänge in Richtung des Endknotens existieren. Abbildung 3.2 veranschaulicht diesen Prozess.

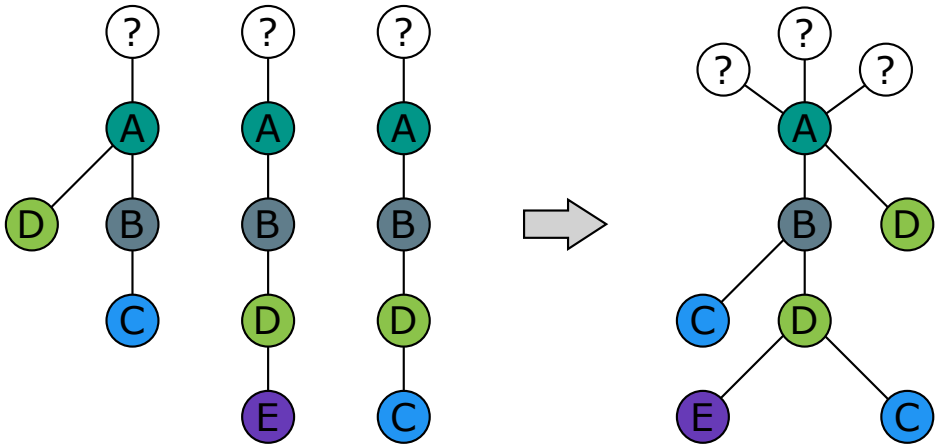


Abbildung 3.2: Bündelung mehrerer Markov-Ketten zu einem Markov-Baum
Beginnend bei A werden die identischen Zustände jeder Kette zusammengefasst. Unterschiede zwischen den Ketten resultieren in Verzweigungen, welche fortan getrennt vom Rest behandelt werden.

Da durch die Projektion von Trajektorien auf den topologischen Graphen diskrete Folgen von Knoten entstehen, können diese ebenfalls in Markov-Trees abgebildet werden. Für den hier verwendeten Algorithmus wird ein Markov-Tree je Knoten des topologischen Graphen erzeugt. Die Knoten des Baums repräsentieren Knoten des Graphen,

wobei tiefere Ebenen in der Projektion weiter zurückliegende Knoten darstellen. Der Wurzelknoten des Baums repräsentiert den Knoten des Graphen, zu welchem der Baum gehört, und wird hier auch als Referenzknoten bezeichnet. Diese Markov-Trees bilden lediglich den Teil $(N_{R,t}, \ldots, N_{t-D})$ jeder Projektion ab, wobei $N_{R,t}$ der Referenzknoten an der Stelle t der Projektion ist und D die maximale Tiefe des Baums.

3.3.1 Abbildung der Übergangswahrscheinlichkeiten

Soll eine neue Trajektorie gelernt werden, wird zunächst, wie zuvor beschrieben, eine Projektion erzeugt. Für jeden Knoten dieser Projektion wird nun der zugehörige Baum trainiert. Der Knoten, zu welchem der aktuelle Baum gehört, wird hier als Referenzknoten R bezeichnet. Zum Training werden außerdem die h Knoten der Vorgeschichte H, so wie der unmittelbar folgende Zielknoten Z benötigt (insgesamt also eine Folge von $h + 2$ Knoten). Da auf den letzten Knoten der Projektion kein weiterer Knoten folgt, also kein Zielknoten existiert, kann der Baum dieses Knotens mit dieser Projektion nicht trainiert werden.

Im Baum wird nun, ausgehend vom Wurzelknoten (welcher R repräsentiert), H abgebildet. Sollten Knoten der Vorgeschichte im Baum noch keine Repräsentation haben, werden diese angelegt. An jedem durch diese Abbildung berührten Knoten des Baums wird vermerkt, wie oft Z beobachtet wurde. Der Algorithmus ist formal in 3.1 dargestellt.

Ein Update eines Markov-Trees für eine beobachtete Trajektorie ist in Abbildung 3.3 dargestellt. Es ist wichtig zu bemerken, dass ein Knoten des topologischen Graphen mehrmals in einem Baum repräsentiert sein kann. In dem in Abbildung 3.3 dargestellten Baum würde z.B. eine Trajektorie $(A|\beta\alpha BR)$ zu einem zweiten Knoten β als Kind von α führen.

Sei nun die Projektion einer Trajektorie $P_T = (n_0, n_1, ..., n_k)$ gegeben. Es soll die Übergangswahrscheinlichkeit von n_k zum (benachbarten) Knoten C berechnet werden. Dazu wird zunächst P_T im Baum von n_k abgebildet. Hierbei werden *keine* neuen Knoten im Baum erzeugt; ist ein Knoten in P_T nicht mehr im Baum abbildbar, stoppt die Rekursion. Der tiefste noch abbildbare Knoten im Baum sei im weiteren als $L_{k,P}$

Eingaben

1 $G = \{n_0, n_1, \ldots n_m, e_0, e_1, \ldots e_m\}$ // *Graph mit n Knoten und m Kanten*

2 $T = (p_0, p_1, \ldots p_r)$ // *Trajektorie mit r Punkten*

Initialisierung

3 $P_T \leftarrow G : T$; // *Projektion von T auf G*

Algorithmus

Next Markov-Tree: **// Iteration über Projektion**

4 `for` R_i `in` $P_T(0 \ldots k-1)$; // *Für jeden Knoten R_i der Projektion*

5 $p = i$; // *Position in Projektion*

6 $Z = P_T[i+1]$; // *Zielknoten*

7 $L = tree\_of(R_i).root$; // *Beginne beim Wurzelknoten des Baums von R_i*

Training: **// Training des Markov-Trees**

8 `incr` $L(Z)$; // *Erhöhe Zähler des Zielknotens an L*

9 `decr` p; // *Gehe in Projektion weiter zurück*

10 `if` $p < 0 \ || \ (i - p) > max\_depth$; // *Abbruchbedingung*

11 **goto Next Markov-Tree**

12 $L = find\_child(L, P_T(p))$; // *Finde Kind von L, welches $P_T(p)$ abbildet*

13 `if` L `not found`; // *Falls Abbildung nicht existiert, erzeuge sie*

14 $L = create\_child(L, P_T(p))$;

15 **goto Training**

Algorithmus 3.1: Training der Markov-Trees

Algorithmische Beschreibung des Trainingsalgorithmus für Markov-Trees

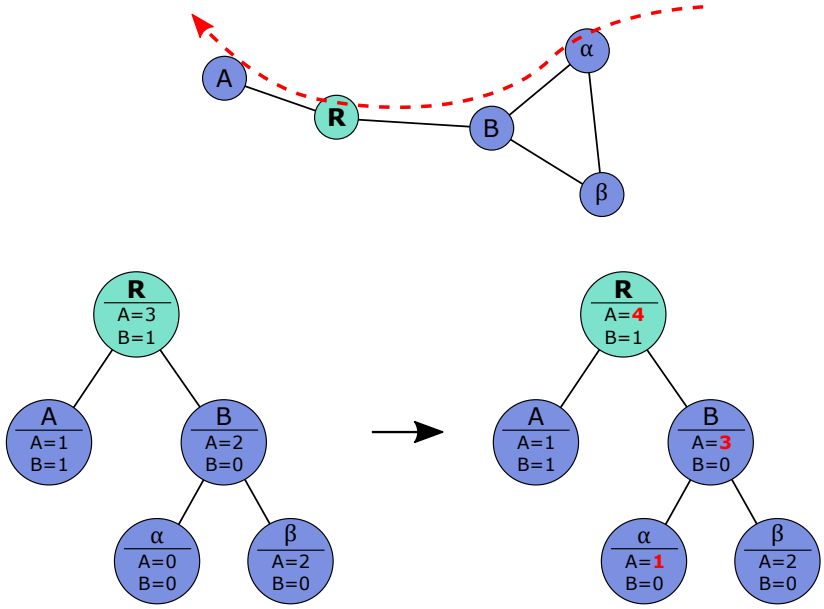


Abbildung 3.3: Lernen in Markov-Trees

Oben: Ein topologischer Beispielgraph. Die eingezeichnete Trajektorie würde als (αBRA) *abgebildet werden.*

Links: Markov-Tree des Knotens R*. An jedem Knoten des Baums ist vermerkt, wie oft ein Übergang zum jeweiligen Nachbarn von* R *beobachtet wurde.*

Rechts: Update des Markov-Trees, nachdem die rot eingezeichnete Trajektorie $(A|\alpha BR)$ *beobachtet wurde.*

Anmerkung: $A : A = 1$ *impliziert, dass eine Beobachtung* $(A|AR)$ *gemacht wurde, die Person also von* A *nach* R *und zurück nach* A *gegangen ist.*

(L für "Leaf") bezeichnet. Wie oben erwähnt, wird an jedem Knoten des Baums gezählt, wie oft welcher Nachbar des Referenzknotens beobachtet wurde. Die Übergangswahrscheinlichkeit berechnet sich nun aus der relativen Häufigkeit der Beobachtungen $L_P(C)$, wie in Formel 3.1 dargestellt.

$$p_{k \to A} = \frac{L_{k,P}(C)}{\sum_N L_P(N)}$$

Formel 3.1: Berechnung der Übergangswahrscheinlichkeit
$p_{k \to C}$*: Übergangswahrscheinlichkeit von* N_k *nach* C
$L_{k,P}$*: Tiefste Repräsentation der Projektion* P_T *im Markov-Tree von* N_k
$\sum_N$*: Summe über alle Nachbarn von* N_k

Beispiel 1 Im in Abbildung 3.3 gezeigten Graphen wird die Trajektorie (αBR) beobachtet. Nun werden die Transitionswahrscheinlichkeiten für die Nachbarn von R, also A und B, bestimmt. Dafür wird versucht, die beobachtete Vorgeschichte im Baum von R abzubilden, beginnend bei R. Dies gelingt vollständig, der letzte abbildbare Knoten ist in diesem Fall α. Der entsprechende Knoten des Baums für α hat bisher eine Transition zu A und keine zu B beobachtet. Daher ist die Transitionswahrscheinlichkeit für $A = \frac{1}{1} = 1$ und für $B = \frac{0}{1} = 0$.

Beispiel 2 Es wird lediglich (R) beobachtet, sodass R der letzte abbildbare Knoten ist. In diesem Fall ist die Transitionswahrscheinlichkeit für $A = \frac{4}{5} = 0.8$ und für $B = \frac{1}{5} = 0.2$.

3.3.2 Abbildung der Transitionszeiten

Um die benötigte Zeit beim Übergang zwischen zwei Knoten in der Prädiktion berücksichtigen zu können, müssen neben den Übergangswahrscheinlichkeiten auch die Übergangszeiten bzw. -verzögerungen gelernt werden. Hierfür werden in K. Schenk 2017 Histogramme mit fester voreingestellter Intervall-Breite geführt, in welchen in

der Lernphase die Übergangszeiten verzeichnet werden. Die Breite der Bins legt gleichzeitig auch die Schrittweite der Prädiktion fest und kann im Nachhinein nicht mehr verändert werden.
Um Lücken im Histogramm gering zu halten und kleinere Abweichungen berücksichtigen zu können, werden Beobachtungen über mehrere Bins verteilt. Da Histogramme eine diskrete Aufteilung des Wertebereichs vornehmen, wird dazu in K. Schenk 2017 statt der Normalverteilung eine Binomialverteilung verwendet, welche effektiv iterativ berechnet werden kann. Um die Breite der Verteilungsfunktion abzuschätzen, wird „Silverman's rule of thumb" verwendet, welche im Kontext der Kernel Density Estimation 1986 von Bernard Silverman vorgeschlagen wurde Silverman 1986. Diese schätzt die Breite eines (Gauss-)Kernels anhand der Anzahl bisheriger Beobachtungen ab. Die Formel ist in Gleichung 3.2 aufgeführt.

$$k_g = (\frac{4}{3})^{\frac{1}{5}} \cdot \sigma_s s^{-\frac{1}{5}} \approx 1.06 \cdot \sigma_s s^{-\frac{1}{5}}$$

Formel 3.2: Silvermans Faustregel
k_g*: Breite des Kernels*
σ_s*: Standardabweichung der Datenpunkte*
s*: Anzahl der Datenpunkte*

Die Verwendung eines Histogramms ermöglicht die Gewichtung verschiedener Übergangszeiten im Prädiktionsschritt. Das Histogramm ist ortsabhängig und im Graphen eine Eigenschaft einer Kante. K. Schenk 2017 schlägt jedoch vor, die Histogramme stattdessen in den Markov-Trees zu speichern. Dies hat den Vorteil, dass die Verzögerungen in Abhängigkeit der Vorgeschichte abgeschätzt werden können, erhöht jedoch den Speicher- und Trainingsbedarf.

3.4 Prädiktion

Ziel der Vorhersage ist es, festzustellen, wo sich eine Person wann aufhalten wird. Mit steigender Entfernung von der letzten Beobachtung (sowohl im Raum als auch in

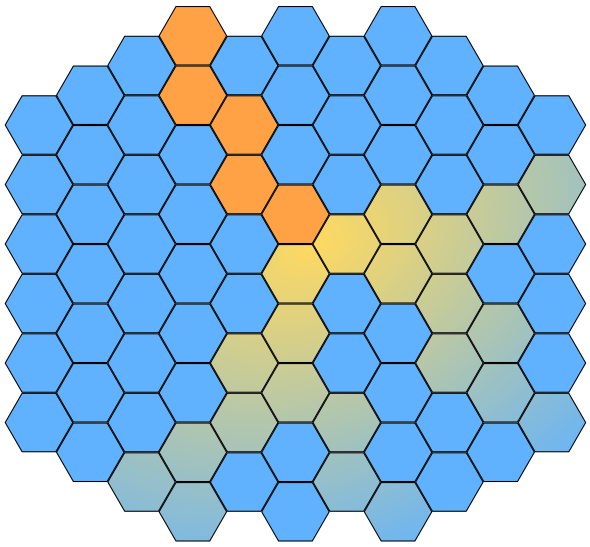


Abbildung 3.4: Prädiktionsergebnis
Vereinfachte Darstellung. Orange: Beobachtete Trajektorie; Gelb: Vorhersage für einzelne Knoten; Blau: nicht aktivierte Knoten

der Zeit) steigt jedoch auch die Unsicherheit, sodass sich die anfänglich noch relativ fokussierte Aufenthaltswahrscheinlichkeit (Okkupanz) ähnlich einer Wellenfront immer weiter verteilt und dabei in ihrer Amplitude abnimmt (das Integral ist jedoch zu jedem Zeitschritt 1). In Abbildung 3.4 ist eine vereinfachte Darstellung zu sehen, wie sich die Okkupanz im Raum verteilt. Die Zuständigkeitsbereiche der Knoten sind hier der Einfachheit halber als reguläre Hexfelder dargestellt.

3.4.1 Probabilistischer Fluss

Der probabilistische Fluss ist das von K. Schenk 2017 vorgestellte Werkzeug, um die Okkupanz im Graphen und über die Zeit zu verteilen. Dieser lässt sich am ehesten mit Wasser in einem verzweigten Rohrsystem vergleichen, wobei die Knoten des Graphen durch Verzweigungen und die Kanten durch Rohre repräsentiert werden. Als Wasser dient die Okkupanz. An Knoten/Abzweigungen wird der Fluss verteilt, durch Kanten/Rohre wird der Fluss verzögert. Fließt nun Okkupanz durch den topologischen Graphen, wird für jeden an einer Verzweigung entstehenden Teilfluss seine Herkunft

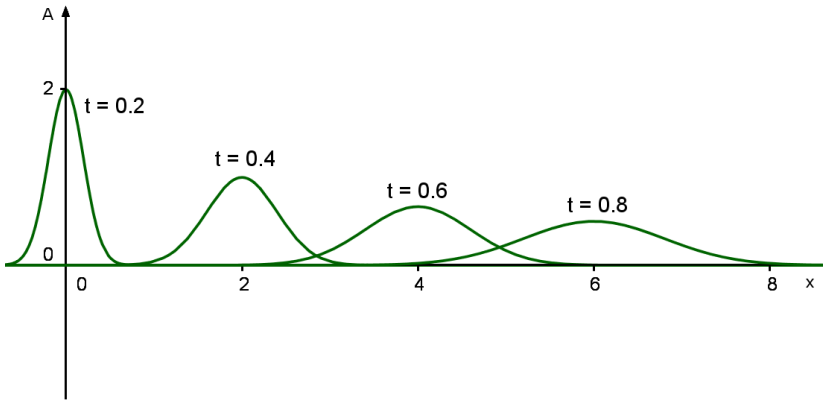


Abbildung 3.5: Verteilung über Raum und Zeit
Beispiel einer Verteilung eines anfänglich scharf abgegrenzten Wertes über die Zeit und eine räumliche Dimension

aufgezeichnet. Aufgrund der Vorgeschichte lassen sich, wie weiter oben erklärt, die Übergangswahrscheinlichkeiten berechnen, d.h. welcher Anteil über welche Kante fließen wird. Mit Hilfe des gelernten Histogramms dieser Transition wird dieser Teilfluss zudem über mehrere Zeitschritte verteilt. Abbildung 3.5 zeigt ein einfaches Beispiel dafür, wie sich ein Signal in einer Dimension über die Zeit verteilt. Die Formel zur Berechnung des Flusses findet sich in 3.3.

$$f_{ij,t+\tau} = \sum_k f_{ki}(t) \ * \ p_{ij} \ * \ \frac{H_{ij}(\tau)}{\sum_T H_{ij}(T)}$$

Formel 3.3: Formel zur Berechnung des Flusses zwischen zwei Knoten
f_{ij}*: Fluss vom Knoten i zum Knoten j*
f_{ki}*: Fluss nach i vom benachbarten Knoten k*
p_{ij}*: Wahrscheinlichkeit eines Übergangs von i nach j*
H_{ij}*: Histogramm der an der Kante ij beobachteten Verzögerungen*
t*: aktueller Zeitschritt*
τ*: Positiver Offset zum Zeitschritt t*

Aus der Berechnung des probabilistischen Flusses ergeben sich Ein- und Ausflüsse

an den Knoten des Graphen. Die Okkupanz zum Zeitpunkt t an einem Knoten K errechnet sich nun aus der Okkupanz im vorigen Zeitschritt und der Summe über alle Einflüsse im aktuellen Zeitschritt, reduziert durch die Summe über alle Ausflüsse im aktuellen Zeitschritt. Dieser Sachverhalt ist auch in Formel 3.4 festgehalten.

$$O_K(t) = O_K(t-1) + \sum_i f_{K,in,i}(t) - \sum_i f_{K,out,i}(t)$$

Formel 3.4: Formel zur Berechnung der Okkupanz aus den Flüssen
O_K: *Okkupanz des Knotens K*
$f_{k,in}$: *Einfluss am Knoten K*
$f_{k,out}$: *Ausfluss am Knoten K*
t: *aktueller Zeitschritt*

Ein wichtiges, nicht sofort offensichtliches, Detail ist, dass jeder Teilfluss auch seine Vorgeschichte mitführt (haben die Markov-Trees eine maximale Tiefe, so kann die Vorgeschichte auf diese Anzahl Knoten beschränkt werden). Daher wird jeder Teilfluss auch individuell weiterverteilt, lediglich die Okkupanz summiert sich an den Knoten auf, da sie von der Vorgeschichte unabhängig ist. Teilflüsse mit gleicher Vorgeschichte können natürlich zusammengefasst werden.

Rechenbeispiel

Anhand der folgenden Abbildungen wird nun ein Beispiel zum probabilistischen Fluss durchgerechnet. Das vollständige Geschehen lässt sich aus Abbildung 3.6 ablesen, wird jedoch in den weiteren Abbildungen noch einmal schrittweise entwickelt. In den Abbildungen werden die Verzögerungshistogramme mit **D** gekennzeichnet und zeigen, *nach* wie vielen Zeitschritten welcher Anteil über die jeweilige Kante fließt. Die Zeitreihen der Aufenthaltswahrscheinlichkeiten an den Knoten sind mit einem $\mathcal{O}$ beschriftet und tragen auf der X-Achse den Zeitschritt. Die Zu- und Abflüsse an den Knoten werden nicht dargestellt.

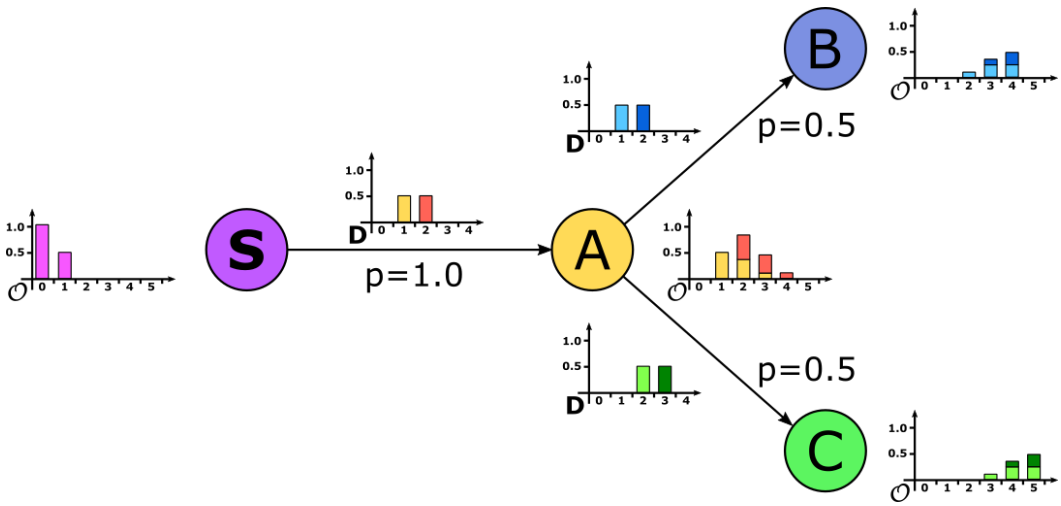
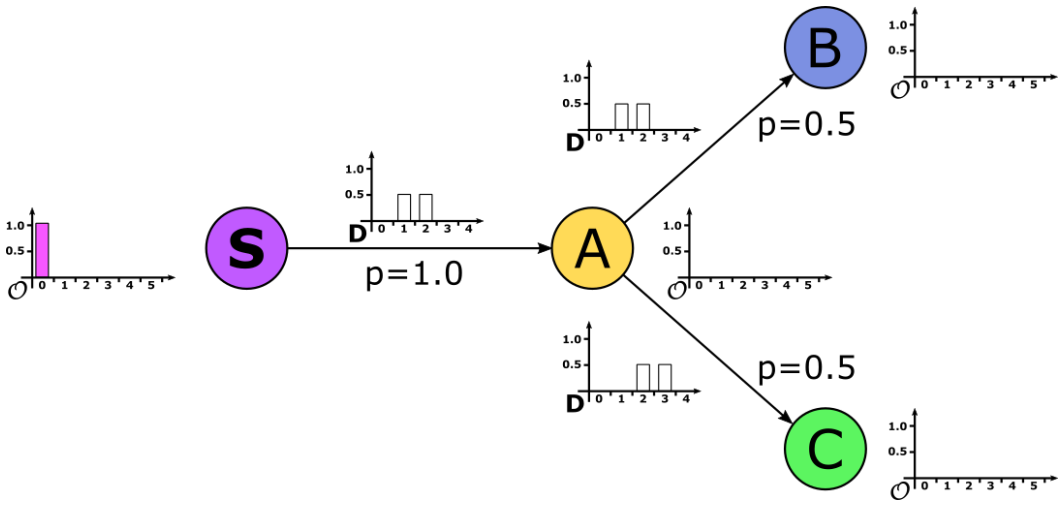


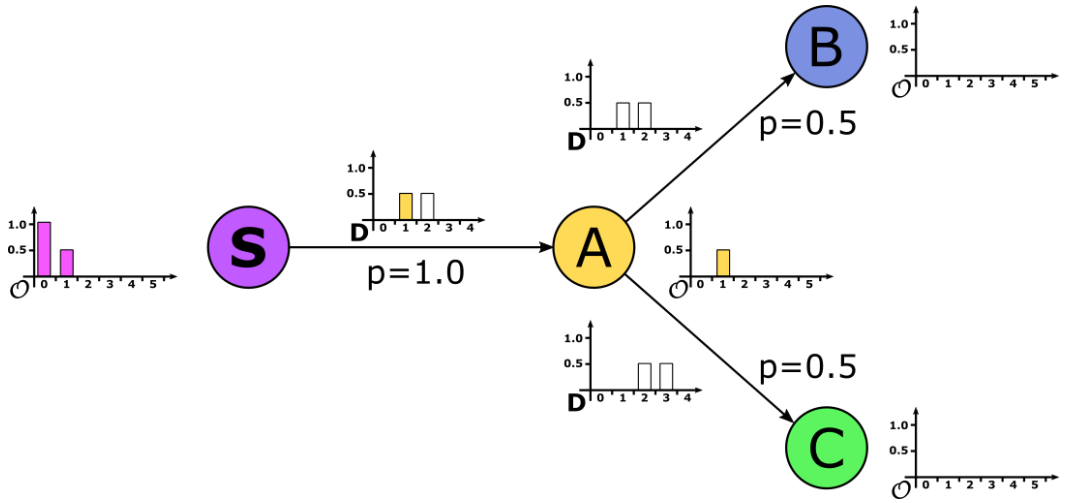
Abbildung 3.6: Rechenbeispiel zum probabilistischen Fluss

*Die Verzögerungshistogramme sind mit **D** markiert und geben an, wie viele Schritte, nachdem ein Fluss eingetroffen ist, dieser weiterverteilt wird. Die Zeitreihen der Okkupanzen sind mit einem $\mathcal{O}$ beschriftet. Das gezeigte Modell besitzt mit **B** und **C** zwei absorbierende Zustände, wodurch sich nach 5 Zeitschritten ein stationärer Zustand ergibt.*



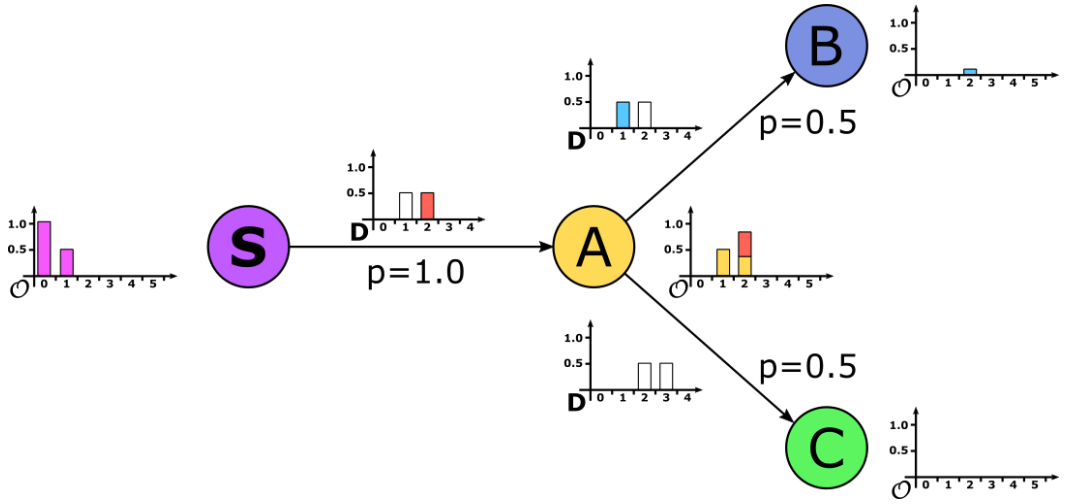
(a) t=0

*Die Person befindet sich am Knoten **S**, die Aufenthaltswahrscheinlichkeit dort ist 1.*



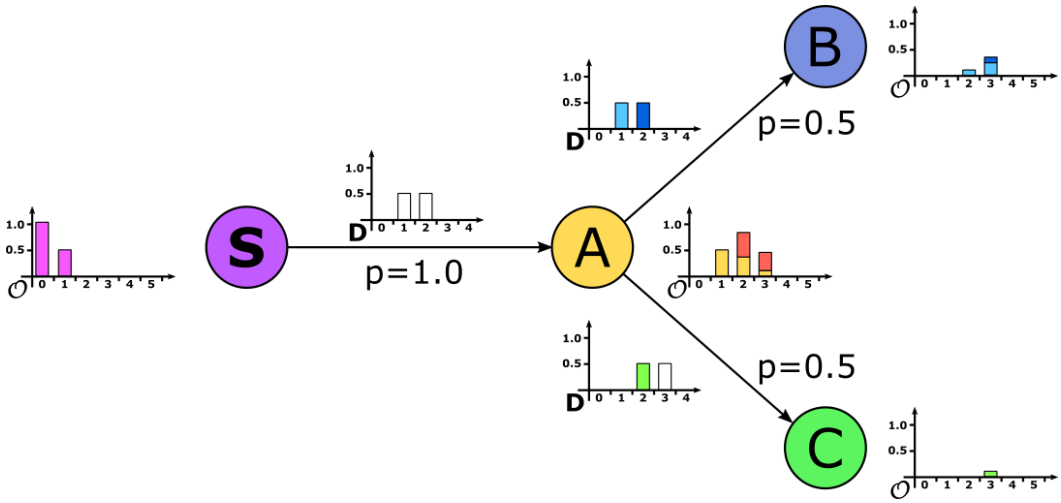
(b) t=1

*Nach einem Zeitschritt Verzögerung werden 50% der Aufenthaltswahrscheinlichkeit nach **A** übertragen. Da es keine weiteren Kanten von **S** gibt, entspricht dies einem Wert von 0,5.*



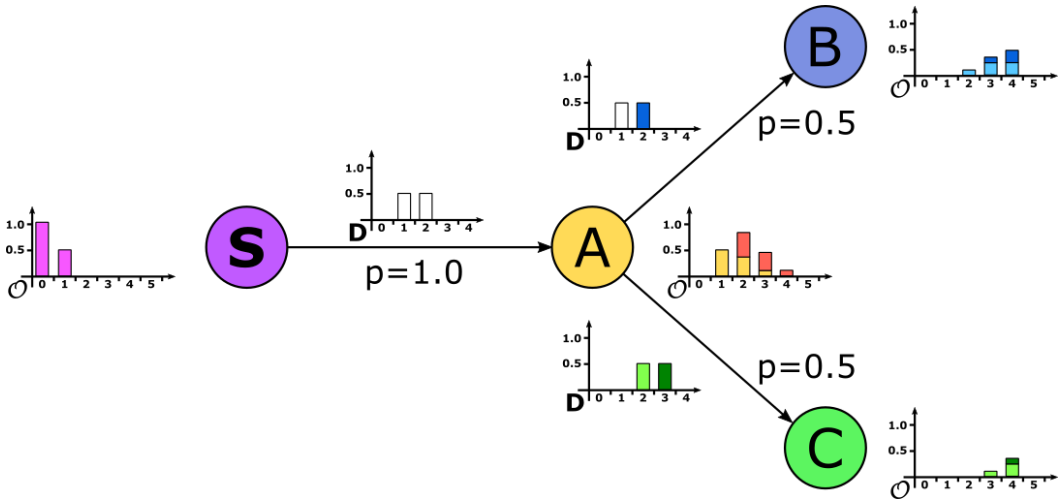
(c) t=2

*Mit zwei Zeitschritten Verzögerung wird die zweite Hälfte von **S** nach **A** übertragen. Gleichzeitig fließen mit einem Zeitschritt Verzögerung 50% des Einflusses nach **A** vom vorigen Zeitschritt nach **B**. Von den eingeflossenen 0,5 werden, da die Kante nur ein Gewicht von 0,5 hat, zunächst nur* $f_{AB}(t=2) = f_{in,A}(t=1) * p_{AB} * D = 0,125$ *nach **B** übertragen.*



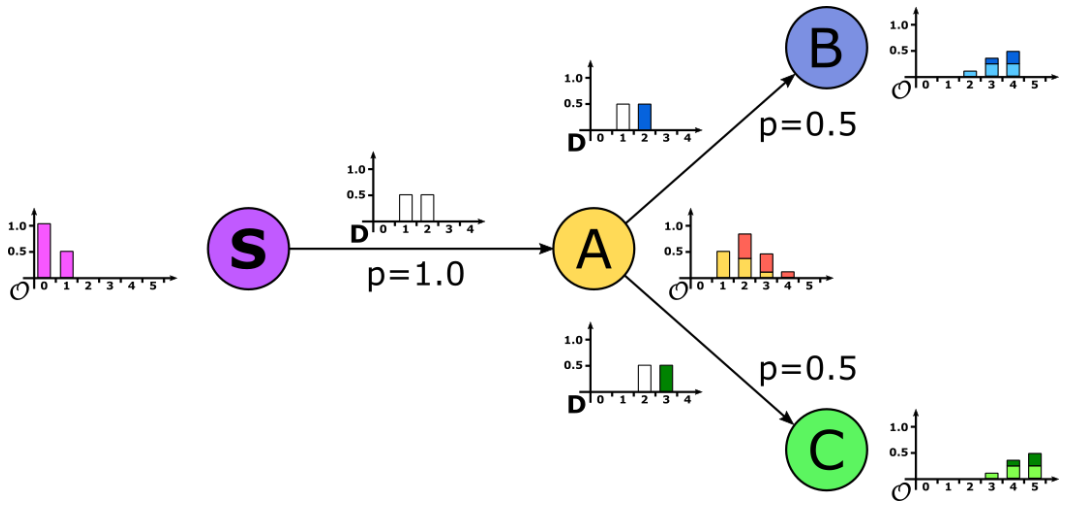
(d) t=3

*Der Strom von **S** nach **A** ist nun erschöpft. **B** erhält weitere 0,125 aus dem ersten Zeitschritt und darüber hinaus noch einmal 0,125, die aus dem zweiten Fluss von **S** nach **A** resultieren. Parallel dazu erhält **C** mit zwei Zeitschritten Verzögerung 0,125 des ersten Teilflusses von **S** nach **A**.*



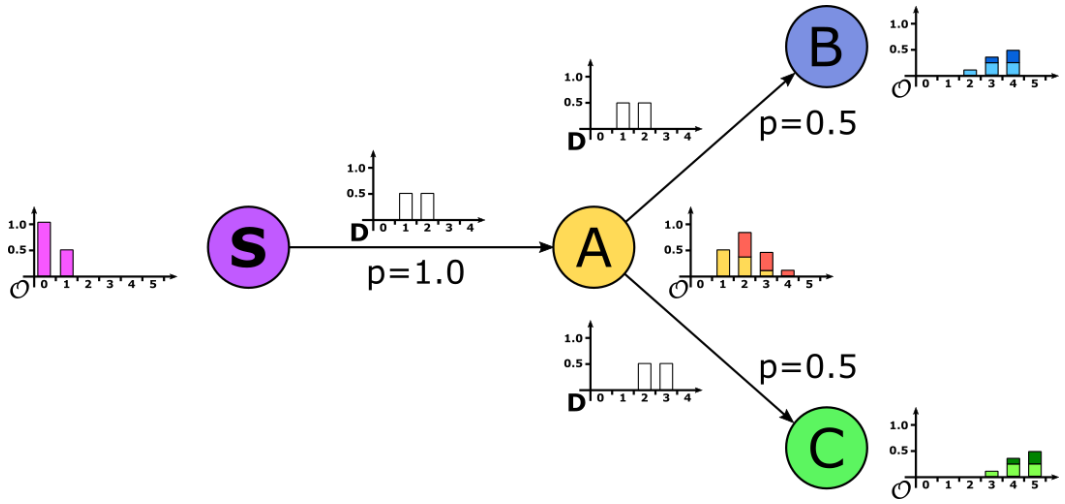
(e) t=4

***B** erhält weitere 0,125 aus Zeitschritt 2, während **C** zweimal 0,125 aus den Zeitschritten 1 und 2 erhält.*



(f) t=5

*Der Strom von **A** nach **B** ist nun ebenfalls erschöpft. **C** erhält 0,125 aus Zeitschritt 2.*



(g) t=6

*Auch der Strom von **A** nach **C** ist nun erschöpft. Da **B** und **C** absorbierende Zustände darstellen, befindet sich das System nun in einem stationären Zustand - die Prädiktion ist nach 5 Zeitschritten beendet.*

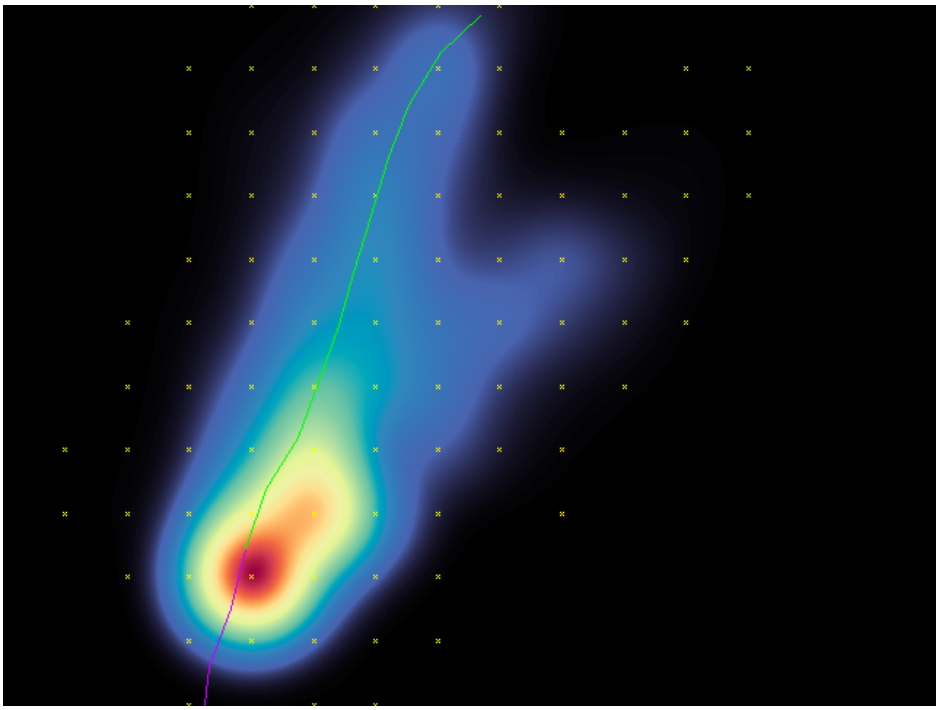


Abbildung 3.8: Exemplarisches Prädiktionsergebnis
Beispielergebnis einer Prädiktion. In Lila ist die dem Algorithmus präsentierte Trajektorie eingezeichnet, die grüne Linie zeigt den tatsächlichen weiteren Verlauf (Groundtruth). Gelbe Punkte stellen Knoten des topologischen Graphen dar, welche während der Prädiktion von Flüssen erreicht wurden. Die zeitliche Entwicklung ist nicht dargestellt.

3.4.2 Ergebnis

Das Ergebnis der Prädiktion ist ein 4-dimensionales Array über den Ort (x und y), den Zeitschritt sowie die Aufenthaltswahrscheinlichkeit. In Abbildung 3.8 ist ein Beispielergebnis einer Prädiktion zu sehen. Die Darstellung zeigt auch, dass das Prädiktionsergebnis leicht in eine Kostenkarte für die weitere Verarbeitung überführt werden kann. Zur Darstellung wurde die Dimension der Zeit ignoriert. Eine detaillierte Untersuchung zur Prädiktionsgüte des Algorithmus findet sich in Kapitel 5.

3.5 Eigenschaften, Stärken und Schwächen

Modellplastizität Das Verfahren sieht bislang keine Methoden vor, um Änderungen der Umgebung und ein damit einhergehendes verändertes Bewegungsverhalten zu berücksichtigen. Insbesondere der topologische Graph darf sich nach der Erstellung nicht mehr ändern, da sonst die in den Markov-Trees der Knoten gelernten Übergangswahrscheinlichkeiten ihre Gültigkeit verlieren.

Online-Training Wie die meisten Algorithmen zur Trajektorienprädiktion, hat das Verfahren den Vorteil, dass die Trainingsdaten nicht gelabelt werden müssen. Die Markov-Trees können daher im laufenden Betrieb durch neue Trajektorien ohne weiteres aktualisiert werden. Da der topologische Graph sich nach der Initialisierung nicht mehr ändern darf, kann prinzipiell jeder beliebige Clusterer verwendet werden.

Parallelisierung Der Algorithmus kann sowohl im Lern- als auch im Prädiktionsschritt leicht parallelisiert werden, allerdings mit unterschiedlichen Konzepten. Im Lernschritt können mehrere Markov-Trees gleichzeitig gelernt werden, jedoch immer nur eine Trajektorie (mehrere sind möglich, wenn die Projektionen keine gemeinsamen Knoten haben). Im Prädiktionsschritt können alle Teilflüsse innerhalb eines Zeitschritts auf mehrere Threads verteilt werden. Die parallele Prädiktion mehrerer Trajektorien ist prinzipiell ebenfalls möglich.

Verallgemeinerungsfähigkeit Im Gegensatz zu z.B. neuronalen Netzen weist das Verfahren keine gute Verallgemeinerungsfähigkeit auf. Da jeder Markov-Tree mit starkem Ortsbezug lernt, jedoch von allen anderen Bäumen isoliert ist, ist ein Wissenstransfer ausgeschlossen. Ähnliche Trajektorien an unterschiedlichen Orten müssen daher separat gelernt werden.

Trainingsaufwand Durch die Identität der Übergangswahrscheinlichkeiten mit der relativen Häufigkeit des Ereignisses sind relativ viele Trajektorien notwendig, um die Markov-Trees auch in tieferen Ebenen stark genug auszuprägen. Dies folgt aus der kombinatorischen Explosion, welche jeder zurückliegende Zeitschritt mit sich bringt:

bei einem Graph mit einheitlich 8er-Nachbarschaft gibt es für jeden Knoten der Vorgeschichte 8 Möglichkeiten, welcher Knoten zu diesem geführt haben könnte. Um einen Baum mit 5 Ebenen komplett auszubilden, müssten daher $8^5 = 32768$ Trajektorien beobachtet werden. Auch wenn unter realen Bedingungen die Bäume wesentlich unvollständiger bleiben, sind Bäume mit einigen tausend Knoten keine Seltenheit und stellen somit auch einen nicht unerheblichen Speicheraufwand dar (einige Hundert Megabyte bis mehrere Gigabyte sind durchaus realistisch).

Bewegungsgeschwindigkeit Der Algorithmus verlässt sich vollständig auf die bisher beobachteten Transitionszeiten und lässt die beobachtete Bewegungsgeschwindigkeit für eine Prädiktion unberücksichtigt.

Echtzeitfähigkeit In der vorliegenden Fassung der Arbeit von K. Schenk 2017 wird angegeben, dass der Algorithmus echtzeitfähig ist, eine genaue Definition fehlt bislang jedoch. In 4 wird daher eine genauere Definition vorgeschlagen, welche auch für den Praxiseinsatz geeignet ist.

Prädiktion in komplexen Situationen Das Verfahren lernt lediglich ein ortsgebundenes Modell und geht nicht auf Interaktionen von Aktoren mit ihrer Umgebung oder anderen Aktoren ein (man stelle sich den Gang zum Kaffeeautomaten oder eine Unterhaltung vor). Komplexe Situationen, in denen beispielsweise eine Person eine neue Agenda annimmt (wohin geht es nach dem Kaffeeautomaten?) oder zwei Personen kollidieren würden, können von diesem Verfahren nicht abgebildet werden.

3.6 Zusammenfassung

Das von K. Schenk 2017 vorgeschlagene Verfahren stellt ein ortsgebundenes Modell, welches ähnliche Trajektorien mit Hilfe einer räumlichen Abastung in Form eines Graphen zu sogenannten Projektionen zusammengefasst 3.1 3.2. Zur Prädiktion werden für jeden Knoten des Graphen Markov-Ketten gelernt, die zu Bäumen gebündelt sind 3.3. Mithilfe dieser Markov-Trees können auf dem Graphen die Übergangswahrscheinlich-

keiten zu benachbarten Knoten bestimmt werden 3.3.1, während die Übergangszeiten zwischen Knoten aus den Trajektorien gelernt und in Histogrammen abgebildet werden 3.3.2. Mithilfe der Übergangswahrscheinlichkeiten und -verzögerungen kann für die Prädiktion die Aufenthaltswahrscheinlichkeit über den Graphen und die Zeit verteilt werden 3.4.1. Aus den ein- und abfließenden Flüssen kann die Aufenthaltswahrscheinlichkeit an den Knoten des Graphen berechnet werden.

Kapitel 4

Eigene Umsetzung

Ziel der Arbeit war unter Anderem die Einbettung des Verfahrens von K. Schenk 2017 in das am Fachgebiet Neuroinformatik verwendete Roboter-Framework MIRA (Einhorn u. a. 2012 `http://mira-project.org`). Um den Algorithmus besser untersuchen und verschiedene Erweiterungen und Varianten testen zu können, wurde die Implementierung modular gestaltet. Prinzipiell lassen sich zwei Arten von Modulen unterscheiden: Graph-Module oder Prozessoren, welche Operationen auf dem Graphen und seinen Komponenten ausführen, und Fluss-Modelle, welche den Prädiktionsalgorithmus anpassen. Eine Übersicht über die zentralen Komponenten der Implementierung findet sich in Abbildung 4.1 und wird in den folgenden Abschnitten näher erläutert.

4.1 Graph

K. Schenk 2017 geht in seiner Arbeit davon aus, dass die Projektion einer Trajektorie sich entlang der Kanten des Graphen bewegt, der nächste Knoten also immer über eine Kante mit dem aktuellen verbunden ist. Daher ist in jener Implementierung der Suchraum stark eingeschränkt, der Graph kann seine Knoten ohne größeren Effizienzverlust in jedem beliebigen Standardcontainer speichern.
Da die Kanten jedoch erzeugt werden (z.B. mit Hilfe einer Delauney-Triangulation), ist es möglich, dass nicht alle real vorkommenden Übergänge abgebildet werden. Um die Abhängigkeit von den Kanten während der Projektion abzuschwächen, wird die

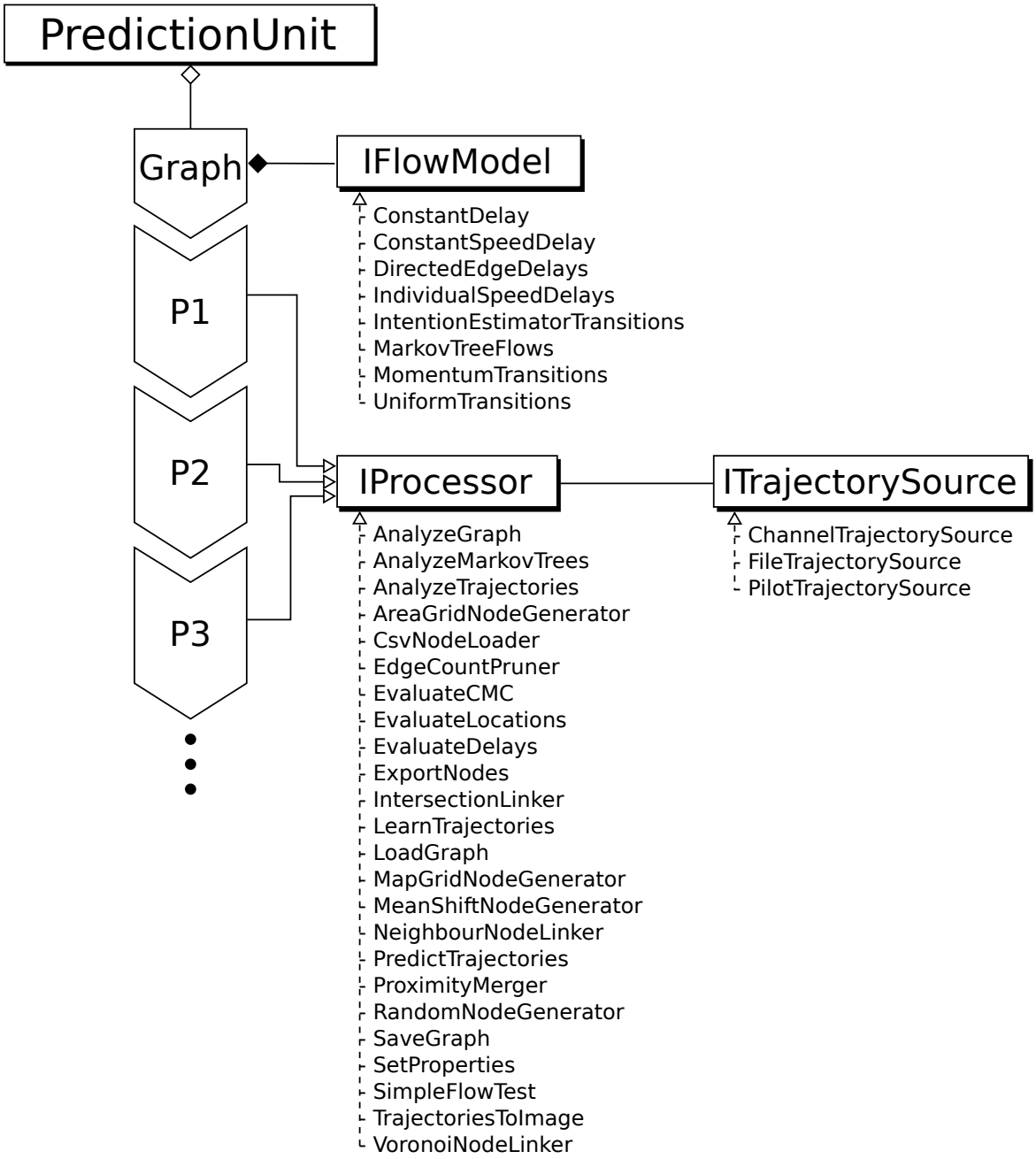
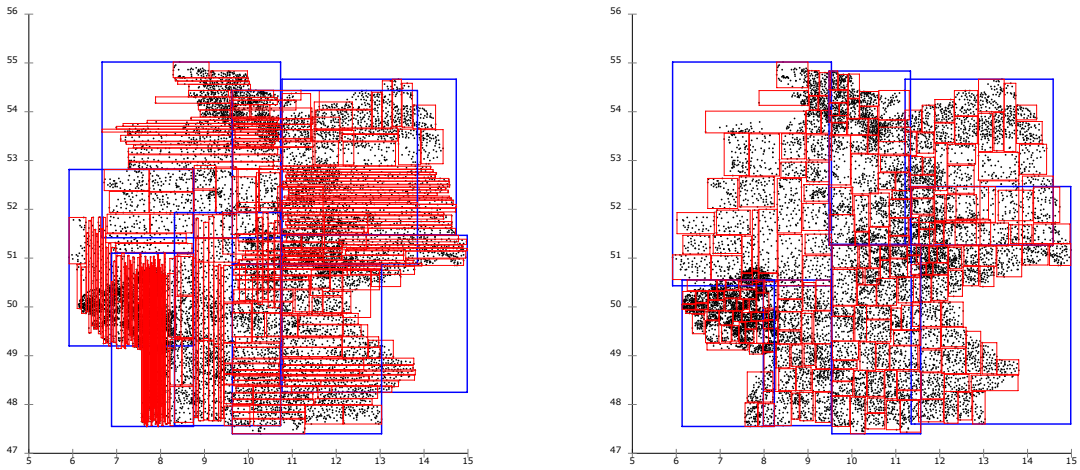


Abbildung 4.1: Übersicht über die zentralen Komponenten der Implementierung
Für die Verbindungslinien wurde die UML-Notation für Klassendiagramme verwendet.

Suche nach dem besten Knoten auf den gesamten Graphen ausgeweitet. Eine naive Implementierung würde allerdings in einer algorithmischen Komplexität von $\mathcal{O}(n)$ für jeden Punkt der Trajektorie resultieren, wobei n die Anzahl der Knoten im Graph ist. Stattdessen verwendet die für diese Arbeit erstellte Implementierung des Graphen einen R-Baum, welcher schnelle Suchen in mehrdimensionalen Räumen erlaubt. R-Bäume wurden erstmals 1984 von Antonin Guttman beschrieben Guttman 1984 und werden häufig in GIS-Datenbanken (Geo-Informations-Systeme) eingesetzt. Die Idee von R-Bäumen ist, die Punkte im Raum in Hyperrechtecken mit unterschiedlicher Überlappung zusammenzufassen. Enthält eines dieser Hyperrechtecke mehr als M Punkte, so wird der Raum weiter unterteilt. Zusätzlich wird ein Index gepflegt, mit welchem leicht Pfade zu Unterräumen gefunden werden können. Mit Hilfe des Index können Anfragen (meist analog zu Datenbanken als Queries bezeichnet) formuliert werden, um z.B. die k nächsten Nachbarn eines Punktes im Raum zu finden. Genau diese Anfrage wird auch genutzt, um für jeden Punkt einer Trajektorie den nächsten Knoten zu finden und so eine Projektion zu erzeugen. R-Bäume haben für Suchen eine algorithmische Komplexität von $\mathcal{O}(\log n)$ und bleiben daher auch für große Mengen von Punkten effizient. In Abbildung 4.2 sind beispielhaft ein R- und ein R*-Baum visualisiert. Die in der Implementierung verwendete R*-Variante verwendet weitere Algorithmen, um die Überlappung von Regionen zu minimieren, was einerseits den Verwaltungsaufwand leicht erhöht, andererseits jedoch die Laufzeit der Queries um 150% und mehr verringert Beckmann u. a. 1990.



(a) R-Baum **(b)** R*-Baum

Abbildung 4.2: Zwei Varianten von R-Bäumen

Die Punkte stellen die Koordinaten der Postleitzahlen in Deutschland dar.

Quelle: „R tree.“ Wikipedia. User:Chire. Version vom 25.10.2011*

Heruntergeladen am 06.11.2016

4.2 Graph-Module

Graph-Module werden in der Implementierung auch als Prozessoren bezeichnet und implementieren ein einfaches Interface, durch welches sie beim Aufruf einen Pointer zum Graphen erhalten, mit dem sie arbeiten können. Die Mächtigkeit dieser Module entsteht aus der Möglichkeit, mehrere als Sequenz zusammenzufassen und hintereinander auszuführen. Daraus ergibt sich alleine schon für die Erzeugung des Graphen eine Vielzahl von Möglichkeiten (ein hexagonales Gitter könnte z.B. aus zwei regulären Gittern und einem *NearestNeighbourLinker* erzeugt werden). Im Folgenden werden einige Prozessoren exemplarisch vorgestellt, eine Übersicht über die wichtigsten Module findet sich in Tabelle 4.1.

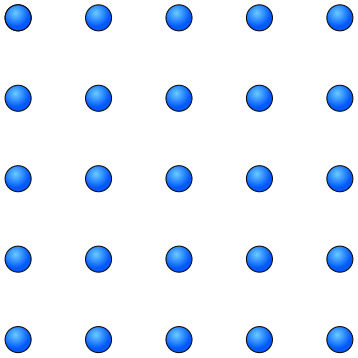
Bezeichnung	Beschreibung
AnalyzeGraph	Berechnet Statistiken des Graphen.
AnalyzeMarkovTrees	Berechnet Statistiken für das MarkovTreeFlows-Modell, soweit vorhanden.
AnalyzeTrajectories	Berechnet Statistiken für geladene Trajektorien.
AreaGridNodeGenerator	Erzeugt Knoten auf einem regulären Gitter.
CsvNodeLoader	Lädt Knoten aus einer .csv-Datei.
EdgeCountPruner	Löscht Knoten mit weniger als n Kanten.
EvaluateCMC	Erstellt eine CMC-Kurve (Cumulative Match Curve) aus Prädiktionsergebnissen.
IntersectionLinker	Erzeugt Knoten an Stellen, an denen sich Kanten kreuzen.
LearnTrajectories	Lädt Trajektorien und trainiert mit ihnen die Fluss-Modelle des Graphen.
MapGridNodeGenerator	Erzeugt Knoten auf einem regulären Gitter in den Freiräumen einer Karte.
MeanShiftNodeGenerator	Clustert Trajektorien mit dem Mean-Shift-Algorithmus und erzeugt Knoten aus den Clusterzentren.
NeighbourNodeLinker	Erzeugt Kanten zwischen nahen Knoten.
PredictTrajectories	Lädt Trajektorien und führt den Prädiktionsalgorithmus für sie aus.
ProximityMerger	Ersetzt nah beieinander liegende Knoten durch einen einzelnen Knoten.
TrajectoriesToImage	Rendert geladene Trajektorien als Bild.
VoronoiNodeLinker	Verknüpft Knoten mit Hilfe der Delauney-Triangulation.

Tabelle 4.1: Die wichtigsten implementierten Prozessoren

4.2.1 Erzeugen von Knoten

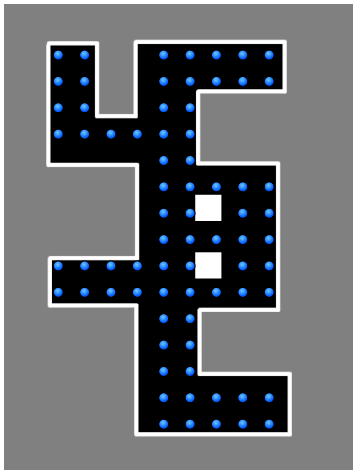
AreaGridNodeGenerator

Erzeugt Knoten auf einem Gitter mit festen Abständen.



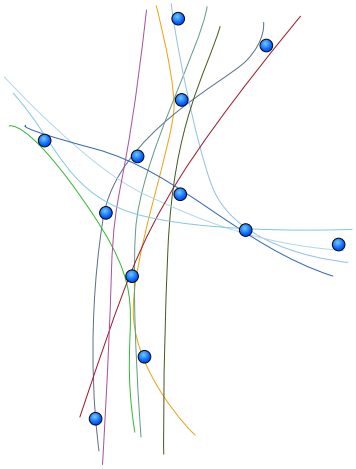
MapGridNodeGenerator

Erzeugt Knoten auf einem Gitter mit festen Abständen, welches über eine Belegtheitskarte gelegt wird. Knoten können nur dort erzeugt werden, wo die Karte Freiräume aufzeigt.



MeanShiftNodeGenerator

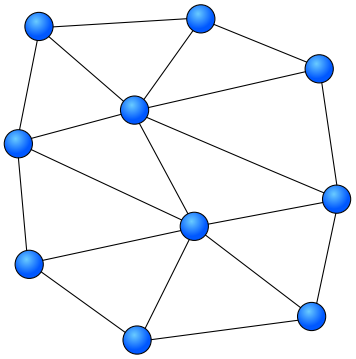
Verwendet den ebenfalls Mean-Shift Algorithmen (Comaniciu, Ramesh und Meer o.D.), um Clusterzentren zu berechnen, welche anschließend zu Knoten des Graphen werden.



4.2.2 Erzeugen von Kanten

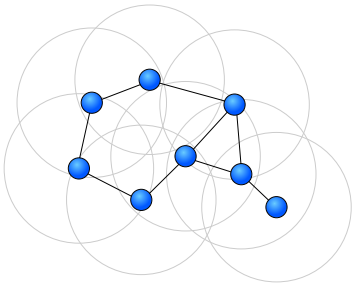
VoronoiNodeLinker

Berechnet die Voronoi-Regionen für die Knoten des Graphen und fügt Kanten entsprechend der Delauney-Triangulation ein.



NeighbourNodeLinker

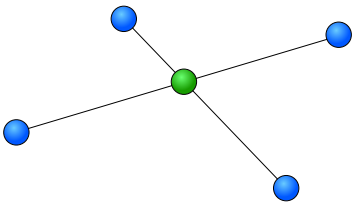
Verbindet alle Knoten, welche einen Abstand unterhalb eines Schwellwertes voneinander haben.



4.2.3 Bereinigung des Graphen

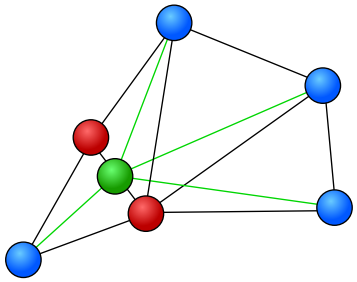
IntersectionLinker

Fügt an Stellen, wo sich Kanten kreuzen, neue Knoten ein und teilt die kreuzenden Kanten an diesen.



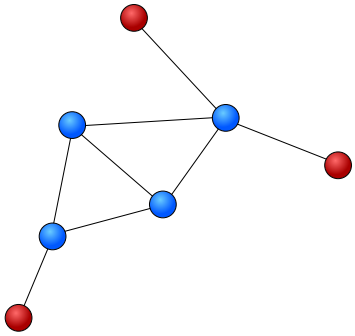
ProximityMerger

Ersetzt Knoten, welche einen Abstand unterhalb eines Schwellwertes voneinander haben, durch einen einzelnen neuen Knoten.



EdgeCountPruner

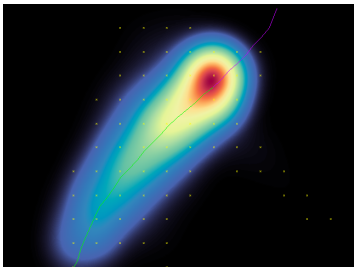
Löscht alle Knoten des Graphen, welche nicht mindestens über die eingestellte Anzahl von Kanten verfügen.



4.2.4 Prädiktion und Auswertung

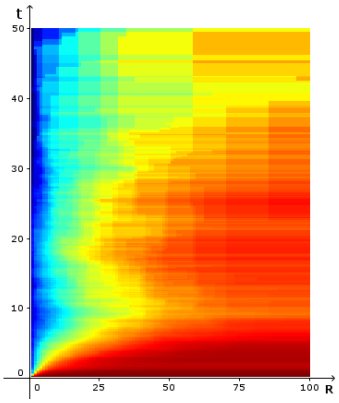
PredictTrajectories

Erstellt Prädiktionen für eine oder mehrere Trajektorien.



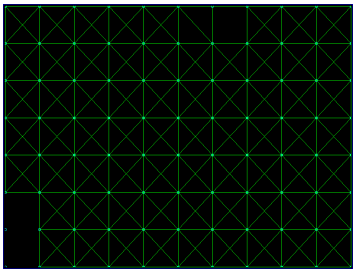
EvaluateCMC

Berechnet die Wahrscheinlichkeit, dass der korrekte Knoten zum korrekten Zeitpunkt vorhergesagt wird.



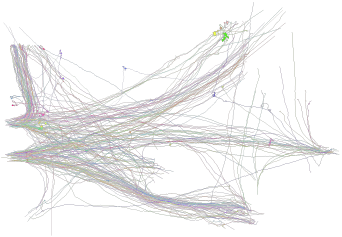
GraphToImage

Visualisiert die Knoten und Kanten des Graphen.



TrajectoriesToImage

Visualisiert eine oder mehrere Trajektorien.



4.3 Flussmodelle

Die Flussmodelle legen fest, wie die Übergangswahrscheinlichkeiten und -verzögerungen im Prädiktionsschritt berechnet werden, wobei Implementierungen nur eines von beiden bereitstellen können. Jedes Modell hat eine Priorität für Übergangswahrscheinlichkeiten und eine Priorität für Verzögerungen. Im Prädiktionsschritt werden die Modelle entsprechend dieser Prioritäten verwendet. Flussmodelle mit niedriger Priorität dienen somit als „Backup-Modelle“ für den Fall, dass Modelle mit höherer Priorität die geforderten Daten nicht bereitstellen können (z.B. wegen noch unzureichendem Training). Im Folgenden werden die wichtigsten implementierten Modelle näher vorgestellt. Eine vollständige Übersicht findet sich in Tabelle 4.2. Tests und Analyseergebnisse zu den verschiedenen Modellen finden sich in Kapitel 5.

4.3.1 MarkovTreeFlows

Dieses Modell verwendet die Markov-Trees, wie sie in Kapitel 3.3 beschrieben wurden, um die Übergangswahrscheinlichkeiten und -verzögerungen bereitzustellen. Die maximale Tiefe der Bäume sowie die maximale Tiefe, in welcher die Verzögerungshistogramme gespeichert werden, sind konfigurierbar. Das Modell verwendet außerdem wie von K. Schenk 2017 vorgeschlagen die Chebyshev-Ungleichung, um eine einstell-

bare Sicherheit der Übergangswahrscheinlichkeiten zu gewährleisten. Diese gibt in eine obere Schranke für die Wahrscheinlichkeit, dass eine Zufallsvariable mit endlicher Varianz einen Wert außerhalb eines (symmetrischen) Intervalls um den Erwartungswert annimmt. Da es sich bei den Beobachtungen um diskrete Ereignisse handelt, kann eine für binomialverteilte Zufallsvariablen angepasste Form der Ungleichung verwendet werden, mit welcher sich leichter arbeiten lässt. Stellt man die Gleichung entsprechend um und gibt Intervallbreite und Sicherheit vor, so erhält man eine Mindestanzahl von Beobachtungen, welche man zur Erfüllung der Vorgaben gemacht haben muss. Die Formel ist in 4.1 aufgeführt. Beim Abbilden einer Knotenfolge im Markov-Tree kann der Abstieg somit abgebrochen werden, wenn ein Knoten weniger als die von der Formel berechnete Anzahl Beobachtungen vorzuweisen hat.

$$p \geq 1 - \frac{1}{4s\epsilon^2} \qquad \rightarrow \qquad s \geq \frac{1}{4(1-p)\epsilon^2}$$

Formel 4.1: Chebyshev-Ungleichung für binomialverteilte Zufallsvariablen
p: Sicherheit der Vorhersage
s: Anzahl der Beobachtungen
ϵ: Maximale Abweichung vom Erwartungswert

4.3.2 IntentionEstimatorTransitions

Um die Flexibilität des Prädiktionsansatzes und der Abtrennung der Flussmodelle zu unterstreichen, wurde ein einfaches Bayes-basiertes Verfahren zur Abschätzung der Übergangswahrscheinlichkeiten implementiert. Der Ansatz basiert auf der „Bayesian Human Motion Intentionality Prediction“ von Ferrer und Sanfeliu 2014. Das Verfahren setzt voraus, dass die möglichen Ziele und ihre Gewichtung bereits bekannt oder identifiziert sind. In diesem Modell werden Ziele anhand der Endpunkte bzw. -knoten der Trajektorien identifiziert. Dies wird durch die durch den Graphen realisierte räumliche Abtastung ermöglicht. Die Wahrscheinlichkeiten der Ziele lassen sich analog zu Formel 4.2 aus der relativen Häufigkeit, mit welcher diese auftauchen, berechnen.

$$p(D) = \frac{o_D}{O}$$

Formel 4.2: Wahrscheinlichkeit eines Zielpunkts
$p(D)$: Wahrscheinlichkeit des Zielknotens D
o_D: Anzahl Trajektorien welche mit D endeten
O: Gesamtanzahl von beobachteten Trajektorien

Das von Ferrer und Sanfeliu 2014 beschriebene Verfahren formuliert Gleichung 4.3, um die bedingte Wahrscheinlichkeit $p(D|P_T)$ eines Ziels D zu berechnen, wenn eine (Projektion einer) Trajektorie P_T gegeben ist. Damit dieser Ansatz dennoch im vorgestellten Framework verwendet werden kann, muss allerdings die Transitionswahrscheinlichkeit zu den zu N benachbarten Knoten $p(N_k)$ berechnet werden. Um diese Brücke zu schlagen, wird hier außerdem für jeden Zielknoten gezählt, wie oft welcher Nicht-Zielknoten beobachtet wurde. Dies ermöglicht unter Anderem die Berechnung der Wahrscheinlichkeit, eine Projektion zu beobachten, wenn ein bestimmtes Ziel angesteuert wird ($p(P_T|D)$). Außerdem tauchen in dieser zielabhängigen Zählung die Gewichte der Nachbarn von N auf, aus welchen sich die bedingten Wahrscheinlichkeiten $p(N_k|D)$ berechnen lassen. Mit Hilfe von Gleichung 4.4 können so die Wahrscheinlichkeiten aller Nachbarn von N bestimmt werden.

$$p(D|P_T) = \frac{p(P_T|D) * p(D)}{p(P_T)}$$

Formel 4.3: Abschätzung der Intention nach Ferrer und Sanfeliu 2014
D: Möglicher Zielknoten
P_T: Projektion der Trajektorie T
Anmerkung: Der Nenner der Gleichung hängt nicht von D ab und kann, da das Ergebnis ohnehin auf $[0, 1]$ normiert werden muss, ignoriert werden.

Das Modell hat den Vorteil, dass es bereits nach wesentlich weniger Training Aussagen

$$p(N_k) = norm_k(p(D|P_T) * p(N_k|D))$$

Formel 4.4: Berechnung der Übergangswahrscheinlichkeiten der Nachbarn
N_k: k-ter Nachbar des aktuellen Knotens
D: Möglicher Zielknoten
P_T: Projektion der Trajektorie T

liefern kann als z.B. das *MarkovTreeFlows*-Modell. Allerdings ist die Identifikation von Zielpunkten stark vom zugrunde liegenden Graphen abhängig und kann eventuell, z.B. bei sehr eng beieinander liegenden Knoten, keine zuverlässigen Aussagen liefern. Die Identifikation der Zielknoten erfordert außerdem vollständige Trajektorien, was in der Regel einen globalen Beobachter voraussetzt. Das Modell unterstreicht die vielfältigen Einsatzmöglichkeiten des erstellten Frameworks und soll vor allem als Machbarkeitsstudie verstanden werden.

4.3.3 MomentumTransitions

Dieses Modell stellt eine einfache Möglichkeit dar, Abschätzungen der zukünftigen Bewegung zu treffen, ohne vorheriges Training zu benötigen. Dazu wird der mittlere Winkel über die letzten n Übergänge (projiziert oder prädiziert) zwischen Knoten errechnet. Benachbarte Knoten werden nun gewichtet, indem die Übereinstimmung zwischen dem ermittelten Winkel und dem Winkel, den sie zum aktuellen Knoten einnehmen, bestimmt wird. Für die Berechnung des Gewichts wird eine Gaußglocke verwendet. In 4.5 sind die entsprechenden Formeln aufgeführt.

Das Modell hat als Schwachstelle, dass es von linearen Transitionen zwischen Knoten ausgeht. Dies kann jedoch, mit großer Abhängigkeit vom verwendeten Graphen, nicht garantiert werden und ist insbesondere bei generischen Graphen, die z.B. auf Gittern basieren, nur durch hohe Knotendichten zu erreichen. Eine bessere Variante wäre daher, aus der beobachteten Trajektorie eine Kurve zu approximieren und diese fortzuführen.

$$c_k = \frac{\measuredangle(N_n, N_k)}{\frac{1}{n}\sum_{i=0}^{n-1} \measuredangle(N_i, N_{i+1})} \qquad w_k = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(1-c_k)^2}{2\sigma^2}\right)$$

Formel 4.5: Berechnung der Übergangswahrscheinlichkeiten der Nachbarn

N_n*: Aktueller Knoten*

N_k*: k-ter Nachbar von* N_n

c_k*: Übereinstimmung des Winkels zu* N_K *zum Eingangswinkel*

w_k*: Gewicht von* N_k

4.3.4 IndividualSpeedDelays

Wie in Abschnitt 3.5 beschrieben, ist eine Schwäche des Verfahrens, dass die individuelle Geschwindigkeit nicht mit in die Prädiktion einfließt. Dieses Modell erstellt daher aus der Geschwindigkeit über die letzten n Millisekunden der beobachteten Trajektorie und der Distanz zum jeweiligen Nachbarknoten ein Verzögerungshistogramm, wie es in Abschnitt 3.3.2 beschrieben wurde. Dieser Ansatz hat außerdem den Vorteil, dass kein Training erforderlich ist und somit keinen zusätzlichen Speicher benötigt. Allerdings bedeutet das Generieren eines Histogramms höheren Rechenaufwand, insbesondere, wenn eine Verteilungsfunktion, wie in 3.3.2 beschrieben, verwendet wird.

$$v = \frac{s}{t} \quad \rightarrow \quad t = \frac{s}{v}$$

Formel 4.6: Berechnung der Verzögerung anhand der Geschwindigkeit

v*: Beobachtete Geschwindigkeit*

s*: Während Beobachtung zurückgelegte Strecke*

t*: Dauer der Beobachtung*

4.3.5 ConstantSpeedDelay

Bei diesem Modell wird die Verzögerung aus einer angenommenen Durchschnittsgeschwindigkeit in Abhängigkeit von der Distanz zweier Knoten berechnet. Die Formeln sind äquivalent zu den in 4.6 angegebenen. Die Histogramme können im Vergleich zu *IndividualSpeedDelays* etwas schneller bestimmt werden, da die durchschnittliche Geschwindigkeit der Person nicht erst ermittelt werden muss. Untersuchungen legen Nahe, dass die bevorzugte Laufgeschwindigkeit von Menschen bei etwa 1,42m/s liegt und Situationsabhängig variiert Browning 2006.

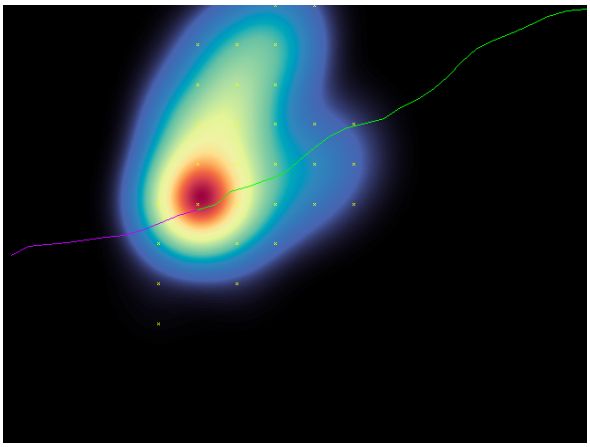
4.4 Performance-Optimierung

Die Übergangswahrscheinlichkeiten stellen letztendlich die relative Häufigkeit einer Beobachtung dar. Da einmal gelernte Beobachtungen nicht einfach verschwinden, erhalten mit der Zeit auch seltene Übergänge (z.B. durch atypisches Verhalten, unsichere oder fehlerhafte Beobachtungen) Wahrscheinlichkeiten verschieden von 0. Da die Teilflüsse individuell verteilt werden und sich daher immer weiter verzweigen, finden sich nach genügend Iterationsschritten unzählige beliebig kleine Teilflüsse. Dies ergibt sich aus den kombinatorischen Möglichkeiten: selbst wenn der wahrscheinlichste Knoten *immer* 70% und 3 weitere Knoten jeweils nur 10% des Flusses erhalten, existieren bereits nach 5 Prädiktionsschritten Flüsse mit einem Anteil von $10^5\% = 0.001\%$ am ursprünglichen Fluss. Dies kann den Rechen- und Speicheraufwand so weit in die Höhe treiben, dass auch moderne Systeme an ihre Grenzen stoßen. Da sie an allen Stellen über unterschiedliche Pfade entstehen, können sie zudem nur selten mit anderen Teilflüssen zusammengefasst werden.
Der Originalalgorithmus sieht für diese keine Sonderbehandlung vor. Als Teil dieser Arbeit muss der Algorithmus allerdings mit vielen anderen zusammen auf einem Roboter funktionieren, eine stundenlange Blockade ist nicht möglich. Eine einfache Lösung besteht darin, Flüsse, welche eine einstellbare Grenze unterschreiten, in den zukünftigen Berechnungsschritten zu ignorieren. Durch Anpassung der Unterschranke kann so die vom Algorithmus benötigte Rechenleistung angepasst werden, der Parameter

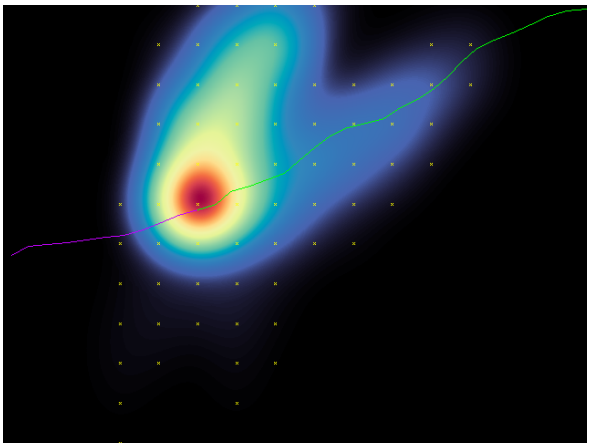
hängt daher auch vom verwendeten Datensatz, Graph und der Aufgabenstellung ab. Wie in Abbildung 4.3 gezeigt, gehen Details verloren, wenn die Grenze zu hoch liegt, während eine zu niedrige Grenze sich negativ auf die Rechenzeit auswirkt. Gute Ergebnisse wurden für den Edinburgh-Datensatz (siehe Abschnitt 5.3) mit einer empirisch ermittelten Grenze von $\min(f) \lessapprox 10^{-5}$ erzielt. Mit diesem Wert konnten im Vergleich zur Prädiktion ohne untere Schranke Laufzeitverbesserungen in der Größenordnung von 20-50 erzielt werden. Auf älteren Systemen wurde die Prädiktion erst durch die Einführung einer unteren Schranke überhaupt möglich.

4.5 Unterschiede zur Referenzimplementierung

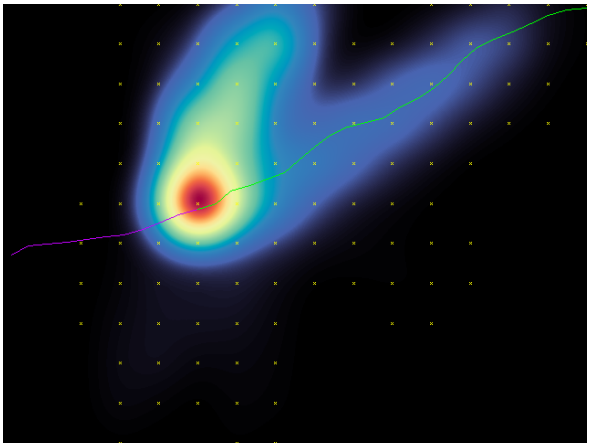
Als Referenz für die eigene Implementierung diente eine ältere Version des Codes von K. Schenk 2017, welche allerdings mehrere Eigenheiten aufwies. Das schwerwiegendste Implementierungsdetail war die Berechnung der Teilflüsse: diese wurden im Referenzcode vollständig im Graphen berechnet, d.h. (Teil-)Ergebnisse wurden in Feldern von Knoten und Markov-Trees gespeichert. Um eine neue Prädiktion zu starten, musste daher zunächst der Graph bereinigt werden, um die entsprechenden Felder zurückzusetzen. Dies verringert die Flexibilität des Codes und verhindert die parallele Prädiktion mehrerer Trajektorien - ein Szenario, welches beim Einsatz auf einem Roboter durchaus zu erwarten ist. Außerdem sind Ergebnisse dadurch fest an den Graphen gebunden und müssen erst aufwendig zusammengetragen werden, um exportiert oder weitergereicht werden zu können. Die eigene Implementierung führt dahingegen alle Berechnungen „in der Schwebe“aus und vereint diese in einem einzigen Objekt. Dies hat weitreichende strukturelle Auswirkungen in den zugrunde liegenden Klassen und verhinderte daher eine Übernahme von Code aus der Referenzimplementierung.
Ein weiterer wichtiger Aspekt ist die in dieser Arbeit angestrebte Modularisierung. Um die Markov-Trees als austauschbares Modul aufzufassen, musste eine Abstraktionsschicht eingefügt werden, die Bäume konnten nicht mehr als fester Bestandteil der Knoten aufgefasst werden.
Die eigene Implementierung wurde natürlich auch an die Gegebenheiten von MIRA angepasst. Beispielsweise bietet MIRA bereits ein einheitliches System, um Algorithmen



$\min(f) = 10^{-4}$



$\min(f) = 10^{-5}$



$\min(f) = 10^{-6}$

Abbildung 4.3: Prädiktion mit verschiedenen Mindestflussgrößen

Prädiziert wurde eine Trajektorie des Edinburgh-Datensatzes. Die präsentierte Trajektorie ist in lila dargestellt, der tatsächliche Verlauf in grün. Gelbe Punkte markieren Knoten, zu denen noch Fluss vorgedrungen ist. Erkennbar ist unter anderem, dass sich der alternative Seitenarm, welcher den tatsächlichen Verlauf besser annähern würde, erst ab einer Untergrenze von ca. 10^{-5} *sichtbar ausprägt.*

über XML-Dateien zu parametrieren und Daten mit anderen, nicht direkt assozierten Programmteilen auszutauschen Einhorn u. a. 2012. Diese wurden extensiv genutzt, um die generierten Daten auch in zukünftigen, auf der Implementierung aufbauenden, Arbeiten verwerten zu können. Des weiteren wurden verschiedene „Good Coding Practices“ mit mehr Elan verfolgt, insbesondere in den Bereichen „Information Hiding“, „Const Correctness“ und der Verwendung von Referenzen und Smart Pointern (siehe auch Sutter und Alexandrescu o.D.).

4.6 Zusammenfassung

Auf der Grundlage des Algorithmus von K. Schenk 2017 konnte ein modulares Framework aufgebaut werden, welches es erlaubt, unterschiedliche Prädiktionsmodelle einzubinden 4.3. Die im Original enthaltenen Markov-Trees werden zusammen mit mehreren anderen Modellen bereitgestellt. Mithilfe von Graph-Prozessoren lässt sich außerdem der zugrunde liegende Graph fast beliebig formen und auch für andere Aufgaben einspannen 4.2. Eine Performance-Analyse zeigte zudem, dass winzige Teilflüsse einen erheblichen Teil der Rechenzeit ausmachen 4.4. Um diesen beizukommen, wurde ein einstellbarer Grenzwert festgelegt, welcher zu kleine Teilflüsse von der Berechnung ausschließt. Bei entsprechend eingestelltem Grenzwert kann die Performance erheblich verbessert werden, während die Prädiktionsergebnisse nur wenig betroffen sind.

Modell	Beschreibung
ConstantDelay	Gibt für alle Übergänge das gleiche Verzögerungshistogramm zurück.
ConstantSpeedDelay	Berechnet die Verzögerung anhand des Knotenabstands und einer vorgegebenen konstanten Geschwindigkeit.
DirectedEdgeDelays	Führt zwei Verzögerungshistogramme je Kante, eines für jede Richtung.
IndividualSpeedDelays	Berechnet ein Verzögerungshistogramm anhand der beobachteten durchschnittlichen Geschwindigkeit.
IntentionEstimatorTransitions	Verwendet ein einfaches Bayes-Verfahren, um mögliche Ziele und somit die Übergangswahrscheinlichkeiten abzuschätzen.
MarkovTreeFlows	Berechnet die Übergangswahrscheinlichkeiten mit Hilfe von Markov-Trees, in welchen auch die Verzögerungshistogramme gespeichert werden.
MomentumTransitions	Gibt höhere Übergangswahrscheinlichkeiten für Knoten, welche mit der mittleren Bewegungsrichtung der letzten n Übergänge übereinstimmen.
UniformTransitions	Verteilt die Übergangswahrscheinlichkeiten gleichmäßig über alle Nachbarn (nur zum Testen gedacht).

Tabelle 4.2: Implementierte Flussmodelle

Kapitel 5

Evaluierung

Um die Implementierung testen und bewerten zu können, wurden zwei öffentlich verfügbare Datensätze von Trajektorien verwendet, welche durch „globale Beobachter“ aufgenommen wurden, d.h. Systeme, welche das zu beobachtende Gebiet vollständig abdecken. Außerdem wurde ein eigener Datensatz mithilfe eines Roboters erstellt, welcher nur Teile des Einsatzgebietes beobachten kann und daher als „lokaler Beobachter“ klassifiziert wird. Auf den Datensätzen wurden verschiedene Modelle trainiert, die Ergebnisse und weitere Untersuchungen werden gemeinsam mit den Datensätzen vorgestellt. Die wichtigsten Ergebnisse sind am Ende des Kapitels in Abschnitt 5.6 noch einmal zusammengefasst.

5.1 CMC-Kurven

Zur Bewertung des Algorithmus werden in dieser Arbeit sogenannte CMC-Kurven (Cumulative Match Curve) verwendet. Diese werden häufig für Klassifizierungsalgorithmen verwendet, wenn diese die Zugehörigkeit eines Samples zu verschiedenen Klassen berechnen. Als Beispiel seien Algorithmen zur Wiedererkennung von Gesichtern genannt, welche häufig Aussagen der Form „dieses Sample gehört mit einer Wahrscheinlichkeit von 60% zu Person 1, mit 12% zu Person 2, usw.“treffen. Für die CMC-Kurve kann für jedes Sample eines Validierungsdatensatzes ein Rang berechnet werden. Dieser entspricht der Position der Groundtruth in der nach Gewichtung sortierten Ausgabe

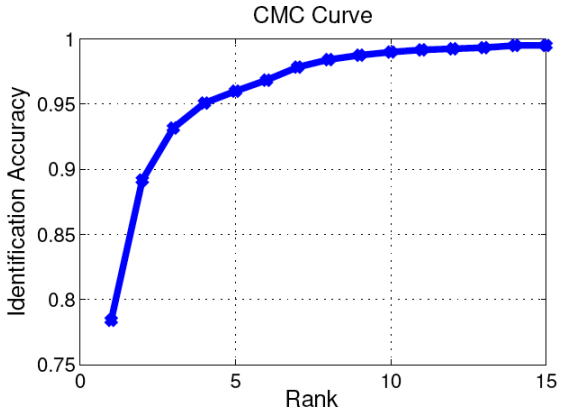


Abbildung 5.1: Beispiel einer CMC-Kurve

Quelle: DeCann und Ross 2013

des Algorithmus (hat die tatsächliche Klasse z.B. das dritthöchste Gewicht, erhält das Sample einen Rang von 3). Durch die Bewertung mehrerer Samples entsteht so eine Kurve über die Wahrscheinlichkeit, dass das richtige Ergebnis unter den r besten Ergebnissen ist. Diese Wahrscheinlichkeit wird häufig auch als (Re-)Identification Accuracy bezeichnet. Ein Beispiel einer CMC-Kurve ist in Abbildung 5.1 zu sehen.
Analog zu K. Schenk 2017 kann die Vorhersage einer Trajektorie als (Re-)Identifikationsproblem des korrekten Knotens aufgefasst werden, was durch die Diskretisierung des topologischen Graphen ermöglicht wird. Jeder Knoten des Graphen stellt somit eine eigene Klasse dar. Von jeder Trajektorie wird ein Teil (hier: 30%) dem Algorithmus als Basis für die Prädiktion gegeben, die Projektion des Rests dient als Groundtruth. Da der Algorithmus Aussagen zur zeitlichen und räumlichen Entwicklung trifft, werden in den Auswertungen drei Varianten verwendet, welche im Folgenden näher beschrieben werden.

5.1.1 Temporale Lokalisation

Um die Vorhersagen zu den Transitionszeiten zu bewerten, wird jede Trajektorie entlang ihrer Projektion ausgewertet. Dazu wird für jeden Knoten k die zeitliche Differenz zwischen dem diesem am nächsten liegenden Trajektorienpunkt P_k und dem nächsten Punkt des Folgeknotes P_{k+1} ermittelt. Der Rang ergibt sich nun aus dem Gewicht des

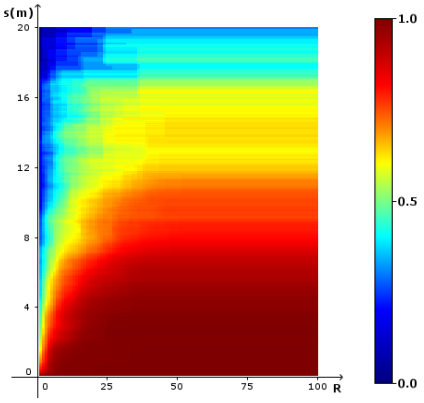


Abbildung 5.2: Beispiel einer CMC-Kurve zur Bestimmung der Lokalisationsgenauigkeit

Auf der X-Achse ist der Rang verzeichnet, die Y-Achse beschreibt die räumliche Entfernung zum Ende der zu prädizierenden Trajektorie in Längeneinheiten der Trajektorie. Die Z-Achse zeigt farblich das Gütemaß (0,0 = blau = 0% wurden richtig identifiziert; 1,0 = rot = 100% wurden richtig identifiziert).

korrespondierenden Bins des Verzögerungshistogramms, welches für diese Transition ($k \rightarrow k+1$) berechnet wurde. Die entstehende CMC-Kurve ist ähnlich der in 5.1 gezeigten.

5.1.2 Spatiale Lokalisation

Für die Bewertung der Verortungsgenauigkeit des Algorithmus werden die CMC-Kurven um eine dritte Dimension erweitert. Auf der X-Achse ist weiterhin der Rang verzeichnet, die Y-Achse beschreibt nun jedoch die räumliche Entfernung zum letzten dem Algorithmus bekannten Punkt der beobachteten Trajektorie (in Längeneinheiten der Trajektorie). Die Prädiktionsgüte wandert in die Z-Achse und wird farblich dargestellt, wobei in dieser Arbeit das verbreitete „Jet“-Farbschema verwendet wird. Ein Knoten gilt als korrekt prädiziert, wenn er mit dem aktuellen Knoten der Projektion übereinstimmt und das höchste Gewicht unter allen Knoten *im selben Zeitschritt* hat. Ein Beispiel ist in Abbildung 5.2 gezeigt.

5.1.3 Kombiniert

Als dritte Variante werden CMC-Kurven ähnlich zu denen für die Bewertung der spatialen Lokalisation (Abbildung 5.2) verwendet. Allerdings steht auf der Y-Achse nun der Prädiktionszeitschritt. Dazu wird zunächst bestimmt, in welchem Zeitschritt sich die Groundtruth an welchem Knoten befindet. Um einen guten (niedrigen) Rang zu erhalten, muss ein Knoten ein hohes Gewicht in dem Zeitschritt besitzen, in dem die Projektion der Groundtruth an diesem Knoten ist. Diese Kurven bewerten daher, wie gut der Algorithmus in Zeit *und* Raum prädiziert.

5.2 Beijing Taxi Trajectory Dataset

Dieser Datensatz[1] wurde auch in K. Schenk 2017 zur Evaluation verwendet und umfasst die GPS-Koordinaten von 28.000 Taxis in Peking (Beijing), welche über einen Monat hinweg aufgenommen wurden. Aufgrund des Formats des Datensatzes können einzelne Fahrten nicht unterschieden werden; stattdessen müssen die Daten durch eine Heuristik in einzelne Trajektorien zerlegt und bereinigt werden. Ein entsprechender Datensatz wurde dankenswerterweise von K. Schenk 2017 zur Verfügung gestellt, so dass hier ein direkter Vergleich angestellt werden kann. Für die Untersuchung standen somit etwa 2,5 Millionen Trajektorien zur Verfügung, welche insgesamt eine Strecke von ca. 13 Millionen Kilometer umfassen. Eine Visualisierung von 10.000 Trajektorien findet sich in Abbildung 5.3. Eine statistische Auswertung der Trajektorien findet sich in Tabelle 5.1.

5.2.1 Einordnung

Das abgedeckte Gebiet umfasst knapp 40 x 34km und deckt den größten Teil Pekings sowie einige Randbereiche ab. Auffällig ist, wie klar die Trajektorien strukturiert sind; für den Betrachter ist es bei diesem Maßstab leicht, einzelne Straßen zu identifizieren. Durch die Größe des Gebiets und der Auflösung von GPS (laut Team 2014 können

[1] `http://sensor.ee.tsinghua.edu.cn/datasets.html`

Abbildung 5.3: Visualisierung des Beijing-Datensatzes

Dargestellt sind 10.000 Trajektorien der von K. Schenk 2017 bereinigten Variante.

	Gesamt	Min	Max	Mean	Varianz
Punkte	5.85e+07	2	2266	22.531	42.960
Länge (LE)	1.31e+08	2	3674.96	50.765	92.705
Ausdehnung (LE)	5.97e+07	0	461.648	23.000	33.811
Dauer (ms)	3.37e+12	1000	8.62e+07	1298459	2618615
Geschwindigkeit (LE/s)	114350	2.37e-05	4.70297	0.044	0.028
P2P-Abstand (LE)	1.31e+08	0.00848	10	2.470	2.616
P2P-Verzögerung (ms)	3.37e+12	1000	900000	60305.293	65705.955

Tabelle 5.1: Die Trajektorien des Beijing-Datensatzes in Zahlen
Eine Längeneinheit (LE) entspricht 100m. Der verwendete Datensatz umfasst 2.598.213 Trajektorien. Die Ausdehnung beschreibt den größten Abstand zwischen zwei beliebigen Punkten einer Trajektorie und ist ein Maß für die Fläche, die von der Trajektorie eingenommen wird. P2P steht für Punkt-zu-Punkt.

ortsabhängig Genauigkeiten von ca. 3,4 x 4,7m mit einer Sicherheit von 95% erwartet werden) wird klar, dass für diesen Datensatz feine Bewegungen nicht abgebildet oder vorhergesagt werden können (eine Längeneinheit entspricht etwa 100m). Aufgrund des deutlich erkennbaren Straßennetzes und der zu erwartenden Abbildung von Hauptverkehrswegen in den Daten ist davon auszugehen, dass Prädiktionen auf diesem Datensatz in der Regel besser abschneiden, als auf anderen Datensätzen.

5.2.2 Auswertung

Für die Auswertung konnte ein Graph verwendet werden, welcher von K. Schenk 2017 mithilfe eines Mean-Shift-Clusterers erstellt wurde. Dieser ist in Abbildung 5.4 visualisiert und wurde mit 2.500.000 Trajektorien trainiert. In der Abbildung wird deutlich, wie viele Kanten ignoriert werden können, da sie keine Straßen abbilden und real niemals von einem Fahrzeug passiert werden können.
Aus der CMC-Kurve der Lokalisationsgenauigkeit in Abbildung 5.5b lässt sich ab-

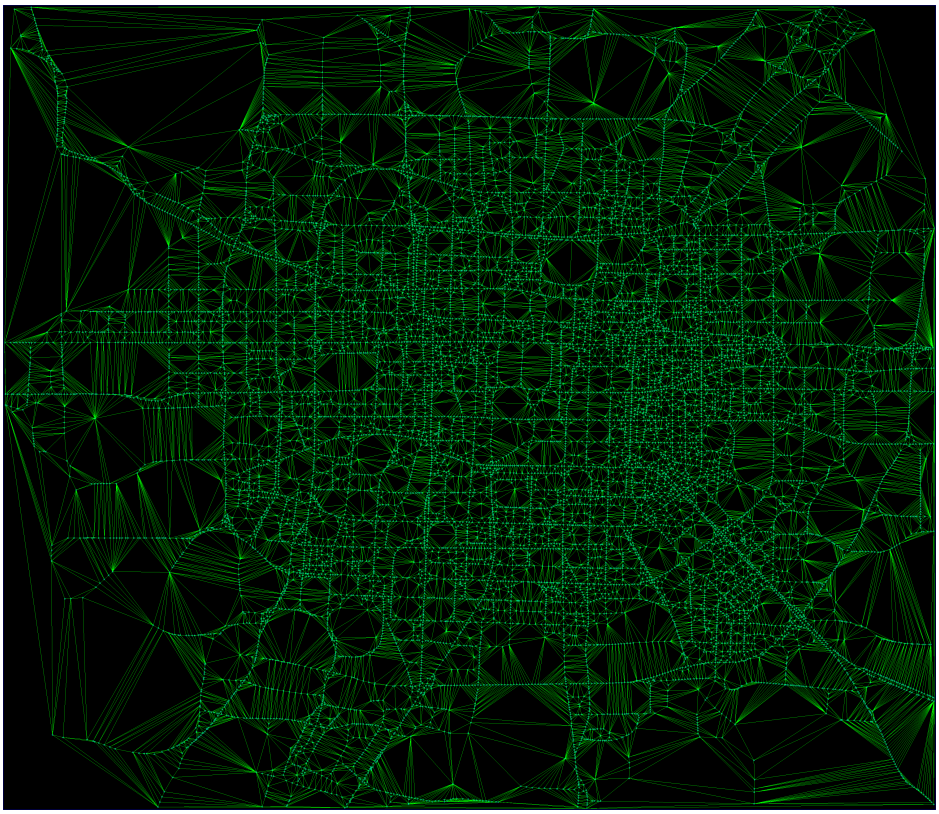


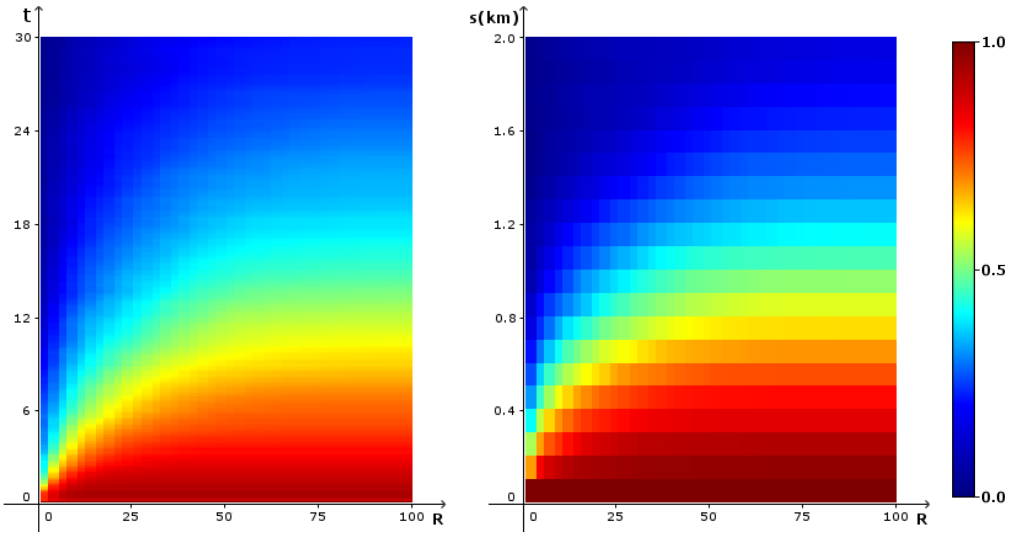
Abbildung 5.4: Topologischer Graph des Beijing-Datensatzes

Darstellung der Knoten und Kanten des von K. Schenk 2017 mit Hilfe eines Mean-Shift-Clusterers erstellten Graphen.

lesen, dass relativ zuverlässige Aussagen für die nächsten 400-500m gemacht werden können. Die Vorhersage der Transitionszeiten in Abbildung 5.5c erreicht hohe Trefferquoten von bis zu 97%, erlangt im Schnitt jedoch erst über die besten 50 Bins der Verzögerungshistogramme eine Trefferquote von über 80%. Dies schlägt sich auch in der kombinierten CMC-Kurve in Abbildung 5.5a nieder: bei einer perfekten zeitlichen Vorhersage wäre der Verlauf gleich zu dem der Lokalisationskurve, wird durch die inkorrekten Verzögerungen jedoch gedämpft.

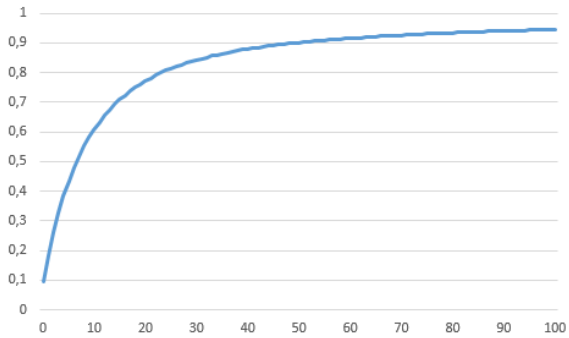
5.2.3 Prädiktionsbeispiele

Um sich ein besseres Bild von den zu prädizierenden Trajektorien zu machen, werden in Abbildung 5.6 einige Prädiktionen exemplarisch dargestellt. An der Anordnung der Knoten und Trajektorien lässt sich gut der Straßenverlauf erkennen. Die begrenzten Entscheidungsmöglichkeiten an den meisten Knoten spiegeln sich gut in der Präzision der Vorhersagen wider. Zur Darstellung der Prädiktionen wird die Dimension der Zeit ignoriert, jedem Knoten wird seine maximale Wahrscheinlichkeit zugeordnet. Die dargestellten Prädiktionsergebnisse werden daher eher durch die CMC-Kurven für die Lokalisation als durch die kombinierten CMC-Kurven beschrieben. Da die Ergebnisse nur punktuell vorliegen, wird eine Normalverteilung verwendet, um diese sichtbar zu machen.



(a) kombiniert

(b) Lokalisation



(c) Zeit

Abbildung 5.5: CMC-Kurven des Beijing-Datensatzes

Alle Grafiken wurden auf der Basis der selben 1.000 Trajektorien bis zum 100. Rang berechnet. Die Lokalisationskurve wirkt verpixelt, da die Trajektorien auf eine 100m (entsprechend einer Längeneinheit) interpoliert wurden.

links oben: Kombinierte CMC-Kurve für die nächsten 30s
rechts oben: Lokalisationsgenauigkeit für die nächsten 2.000m
unten: Genauigkeit der temporalen Vorhersagen

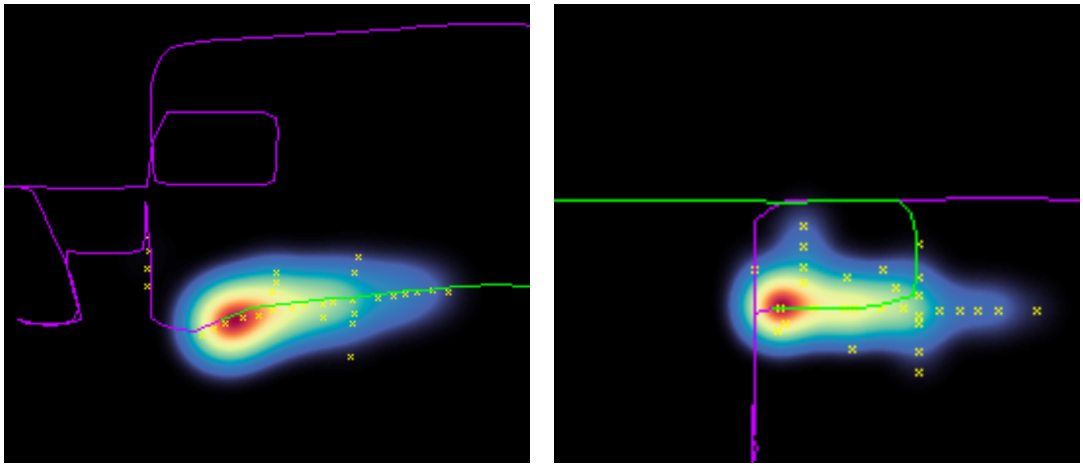


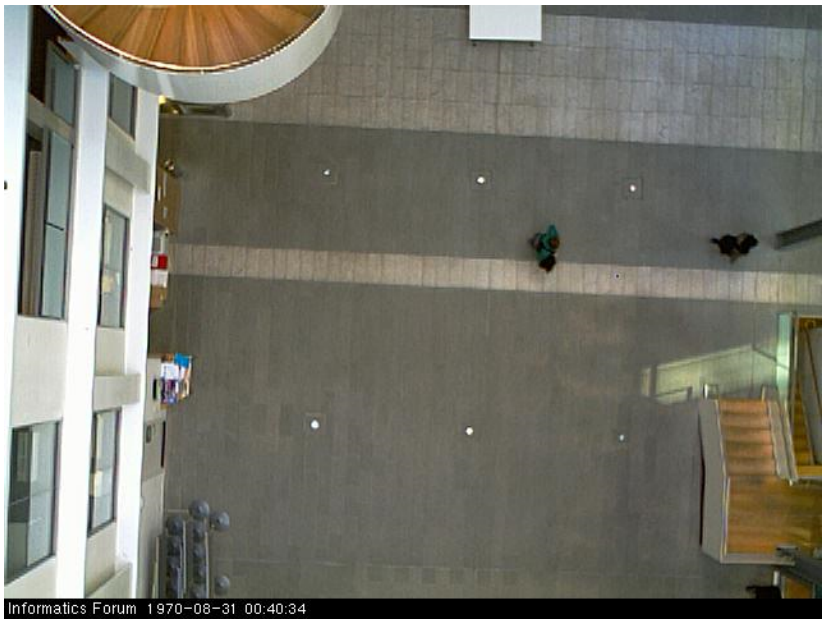
Abbildung 5.6: Ausgewählte Prädiktionsergebnisse

Die Knoten, welche noch Fluss erhalten haben, sind gelb markiert. Der gezeigte Teil der Trajektorie ist lila markiert, der weitere Verlauf grün. Die Prädiktionsergebnisse sind exakt und ohne Unschärfe an den Knoten lokalisiert, werden jedoch zur Darstellung mit einer Normalverteilung ausgebreitet und eingefärbt.

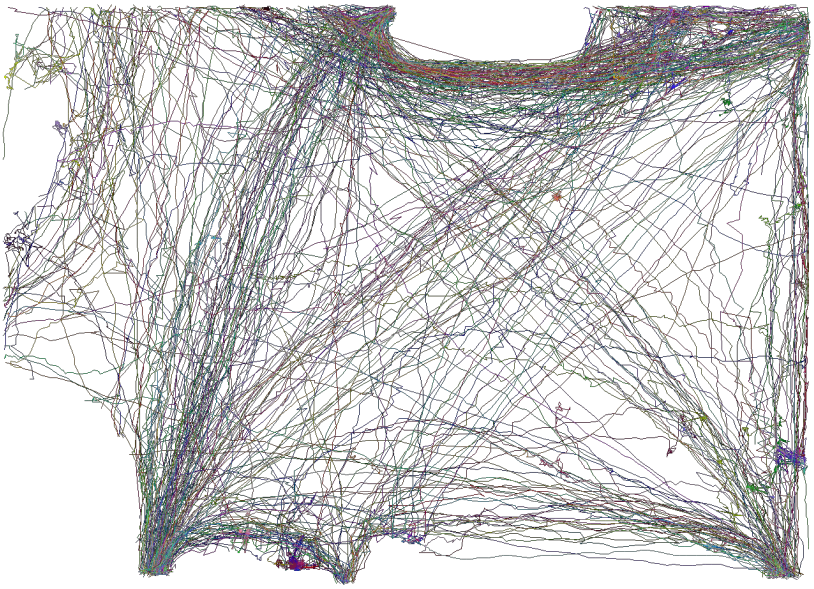
5.3 Edinburgh Informatics Forum Pedestrian Database

Der Edinburgh Datensatz[2] wurde im Forum der School of Informatics der Universität Edinburgh mithilfe einer deckenmontierten Kamera aufgenommen. Personen wurden mithilfe eines personenfreien Referenzbildes segmentiert und getrackt. Der Datensatz wurde von 2009 bis 2010 über 120 Tage auf 10 Monate verteilt aufgenommen. Der Datensatz beinhaltet etwa 110.000 Trajektorien, wobei aufgrund des Segmentierungsverfahrens auch Tische und andere Einrichtungsgegenstände enthalten sein können, welche für Veranstaltungen neu positioniert wurden. Diese sind jedoch aufgrund ihrer Unbeweglichkeit leicht herauszufiltern. Eine Beispielaufnahme der Kamera sowie eine Visualisierung von 10.000 Trajektorien findet sich in Abbildung 5.7. Eine statistische Auswertung der Trajektorien findet sich in Tabelle 5.2.

[2] `http://homepages.inf.ed.ac.uk/rbf/FORUMTRACKING/`



(a) Einzelne Aufnahme der Kamera



(b) 500 Trajektorien

Abbildung 5.7: Visualisierung des Edinburgh-Datensatzes

	Gesamt	Min	Max	Mean	Varianz
Punkte	9.99e+06	16	27814	90.570	168.169
Länge (LE)	5.61e+07	88.3196	14072.3	508.199	222.867
Ausdehnung (LE)	4.42e+07	26	746.776	400.413	147.478
Dauer (ms)	1.08e+09	1500	1.56e+07	9848.50	78613.572
Geschwindigkeit (LE/s)	7.62e+06	0.48375	505.607	69.105	32.797
P2P-Abstand (LE)	5.61e+07	1	627.204	6.302	5.954
P2P-Verzögerung (ms)	1.08e+09	100	1.56e+07	109.953	8111.703

Tabelle 5.2: Die Trajektorien des Edinburgh-Datensatzes in Zahlen
Eine Längeneinheit (LE) entspricht einem Pixel, bzw. 24.7mm. Der verwendete Datensatz umfasst 110.393 Trajektorien. Die Ausdehnung beschreibt den größten Abstand zwischen zwei beliebigen Punkten einer Trajektorie und ist ein Maß für die Fläche, die von der Trajektorie eingenommen wird. P2P steht für Punkt-zu-Punkt.

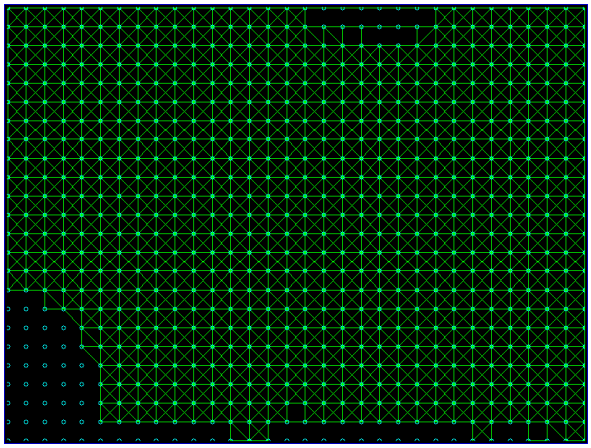
5.3.1 Einordnung

Der aufgenommene Bereich misst etwa 15,8 x 11,9m, die Basislängeneinheit ist ein Pixel und entspricht 24,7mm. Die Visualisierung zeigt, dass es zwar typische Pfade zu geben scheint, diese die Trajektorien jedoch weitaus weniger stark prägen, als es beim Beijing-Datensatz der Fall ist. Dieser Datensatz beschreibt die zu erwartende Einsatzumgebung daher wesentlich besser, allerdings stammen die Daten nach wie vor von einem stationären globalen Beobachter.

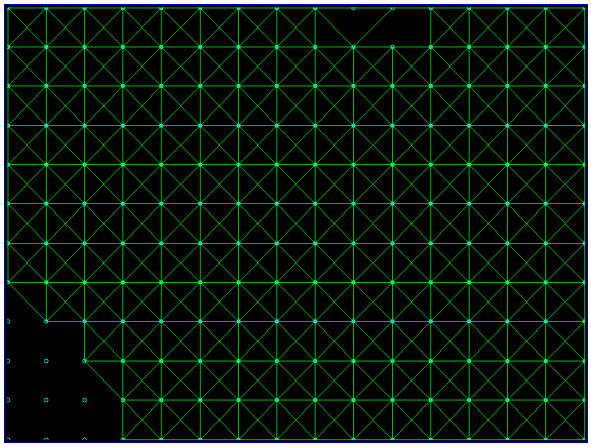
5.3.2 Auswertung

Bei der Auswertung dieses Datensatzes fand ein Graph basierend auf einem regulären Gitter Verwendung. Der Vergleich mehrerer Rohbilder (siehe 5.7) zeigte, dass Personen in der Regel eine Fläche von 40x40 Pixeln auf dem Bild einnehmen. Untersucht wurden daher die Gittergrößen 20x20, 40x40 und 60x60. Der Graph des 40x40 Gitters ist in Abbildung 5.8 abgebildet. Zu beachten ist, dass das Gitter keine Kanten vorgibt, alle

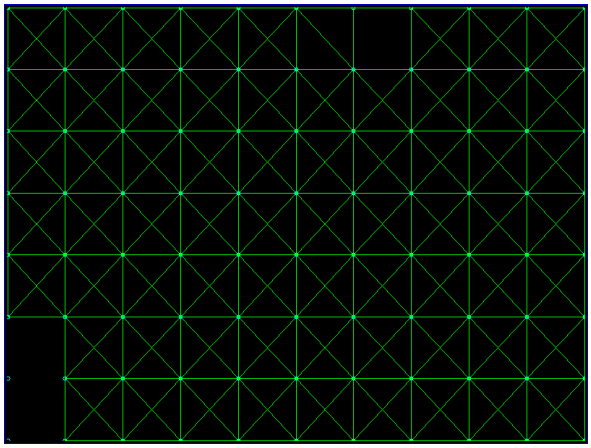
abgebildeten Kanten wurden beim Lernen aus der Projektion der Trajektorien erzeugt. Um zu verhindern, dass Knoten von der Projektion übersprungen werden, wurden die Trajektorien interpoliert. Jeder Graph wurde mit 100.000 Trajektorien trainiert.
In Abbildung 5.9 sind die CMC-Kurven der drei Graphen dargestellt. Ausgewertet wurden jeweils 1.000 Trajektorien. Die Auswertung zeigt, dass mit höherem Gitterabstand auch die Prädiktionsgüte steigt. Dies ist zu erwarten, da die Projektion kürzer wird, weniger Klassen zur Auswahl stehen und dadurch der durchschnittliche Rang sinkt (besser wird). Im Gegenzug wird die Aussagekraft jedes Knotens geringer, d.h. der vermutete Aufenthaltsort der Person wird ungenauer. Die Auswahl eines Gitters ist daher vor allem abhängig von der geplanten Anwendung. Das Prädiktionsergebnis einer Trajektorie ist in Abbildung 5.10 für die verschiedenen Graphen dargestellt. Für die weitere Betrachtung wurde das 40x40-Gitter gewählt, da dieses einen guten Kompromiss zwischen Genauigkeit und Güte darstellt. Um eine für Menschen besser interpretierbare Darstellung zu erreichen, wurden die Wahrscheinlichkeiten zwischen den Knoten interpoliert und als Heatmap dargestellt.
Die CMC-Kurven zur Lokalisation zeigen, dass die spatiale Prädiktion ein hohes Gütemaß aufweist. Allerdings liegen die temporalen Vorhersagen mit maximal 37% im Falle des 20x20-Graphen deutlich hinter den Erwartungen zurück, wodurch auch die kombinierten CMC-Kurven weniger überzeugend sind. Die sogar noch niedrigere Trefferquote beim 40x40-Graphen (maximal 28%) und dem 60x60-Graphen (maximal 21,5%) führen zu der Begründung, dass sich die Personen nicht geradlinig von Knoten zu Knoten bewegen, sondern sich im Raum frei bewegen können. Dabei bewegen sich die Personen auch häufig zwischen Knoten hindurch und streifen diese dabei nur, wodurch sich deutliche Abweichungen in der Transitionszeit ergeben. Das mittlere Verhältnis von erwarteter zu tatsächlicher Transitionszeit $\frac{d_{exp}}{d_{gt}}$ wurde zu $3,4102$ berechnet, wobei Faktoren im Bereich $[6,6 * 10^{-5}, 308]$ beobachtet wurden. Die erwartete Transitionszeit ist hier das Histogramm-Bin mit dem höchsten Gewicht. Auch die Verwendung eines individuell an die Bewegungsgeschwindigkeit der Person angepassten Histogramms (siehe 4.3.4) konnte die CMC-Kurve nur um wenige Prozent verbessern, da die gleichen Probleme bestehen. Für den Fall der Personenprädiktion müsste daher ein neuer Ansatz zur Berechnung der Transitionszeiten gewählt werden, um auch eine zeitlich korrekte



(a) 20x20 (768 Knoten)



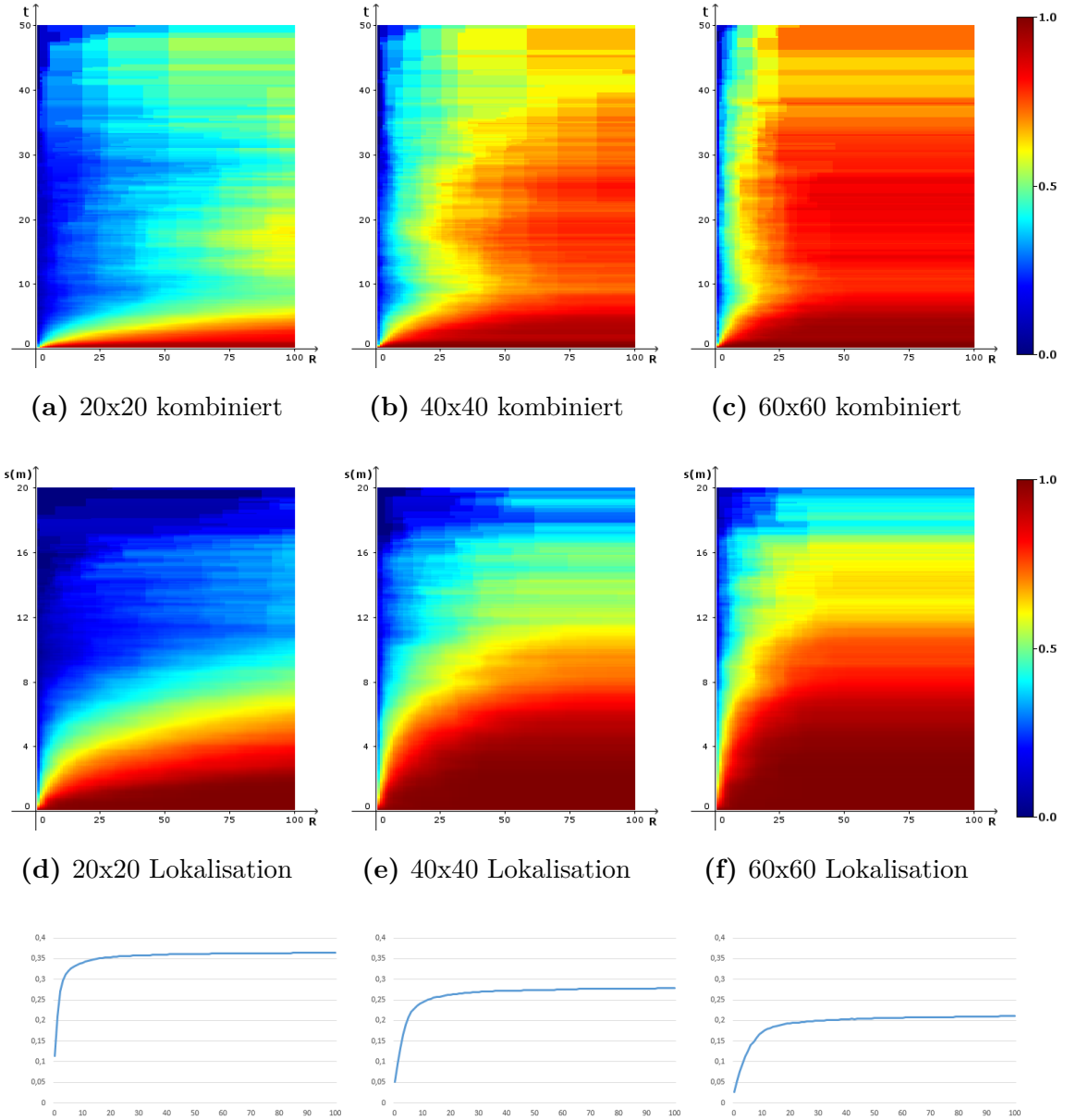
(b) 40x40 (192 Knoten)



(c) 60x60 (88 Knoten)

Abbildung 5.8: Topologische Graphen des Edinburgh-Datensatzes

Mehrere reguläre Gittergraphen mit unterschiedlichen Knotenabständen. Die Kanten wurden während des Trainings aus der Projektion der Trajektorien erzeugt.



(a) 20x20 kombiniert **(b)** 40x40 kombiniert **(c)** 60x60 kombiniert

(d) 20x20 Lokalisation **(e)** 40x40 Lokalisation **(f)** 60x60 Lokalisation

(g) 20x20 Zeit **(h)** 40x40 Zeit **(i)** 60x60 Zeit

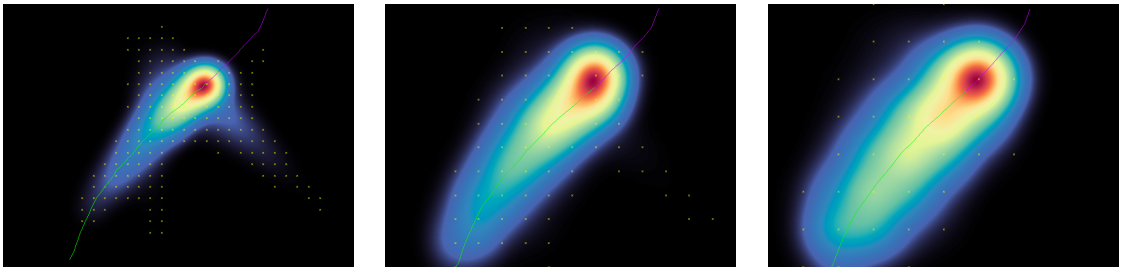
Abbildung 5.9: CMC-Kurven des Edinburgh-Datensatzes

Alle Grafiken wurden auf der Basis der selben 1.000 Trajektorien bis zum 100. Rang berechnet. Zu beachten ist, dass mit einem Rang bis 100 bereits ein signifikanter Anteil des Graphen berücksichtigt wird.

oben: Kombinierte CMC-Kurven für die nächsten 50s

mitte: Lokalisationsgenauigkeit für die nächsten 20m

unten: Genauigkeit der temporalen Vorhersagen (das Maximum der Y-Achse liegt bei 0,4)



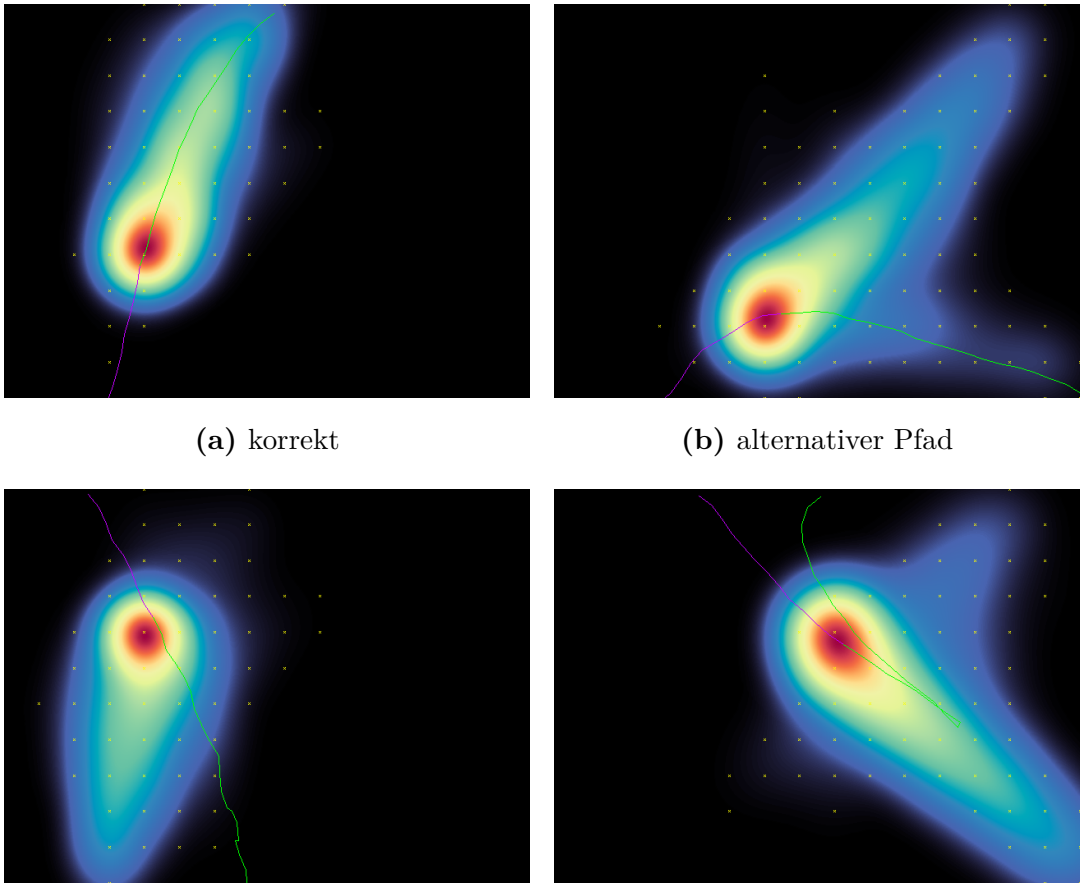
(a) 20x20 **(b)** 40x40 **(c)** 60x60

Abbildung 5.10: Vergleich einer Prädiktion auf unterschiedlichen Graphen
lila: Gezeigter Teil der Trajektorie.
grün: Tatsächlicher weiterer Verlauf.
gelb: Knoten, zu denen noch Fluss vordrang.

Prädiktion zu erhalten.

5.3.3 Prädiktionsbeispiele

Hier werden einige weitere Prädiktionsbeispiele exemplarisch vorgestellt. Die Beispiele enthalten typische (5.11a, 5.11b, 5.11c) und atypische (5.11d) Trajektorien, sowie korrekte (5.11a, 5.11b) und falsche (5.11c, 5.11d) Prädiktionen.



(a) korrekt **(b)** alternativer Pfad

(c) falsch **(d)** atypisch

Abbildung 5.11: Ausgewählte Prädiktionsergebnisse

5.4 Eigener Datensatz

Um die Eignung für den geplanten Einsatz auf einem Roboter zu zeigen, ist ein Datensatz vonnöten, welcher die Wege von Personen in der Umgebung eines fahrenden Roboters umfasst. Leider konnte jedoch ein solcher Datensatz nicht gefunden werden, sodass letztendlich ein eigener Datensatz erstellt wurde. Es zeigte sich allerdings, dass ein ähnlich umfangreicher Datensatz den zeitlichen Rahmen dieser Arbeit gesprengt hätte. Daher wurde nur ein sehr kleiner Datensatz mit 1.830 Trajektorien im Humboldt-Bau der TU Ilmenau aufgenommen, anhand dessen relevante Unterschiede zu durch globale Beobachter aufgenommenen Datensätzen aufgezeigt werden sollen.

Eine statistische Auswertung der Trajektorien findet sich in Tabelle 5.3.

	Gesamt	Min	Max	Mean	Varianz
Punkte	84920	3	635	46.404	49.236
Länge (LE)	8819.73	0.501574	52.109	4.820	5.138
Ausdehnung (LE)	6623.71	0.0596143	36.0775	3.620	4.090
Dauer (ms)	2.61e+07	392	4.66e+06	14307	154375
Geschwindigkeit (LE/s)	1479.47	0.05459	2.32374	0.808	0.399
P2P-Abstand (LE)	8819.73	0.00218	20.622	0.106	0.103
P2P-Verzögerung (ms)	2.61e+07	11	4.64e+06	314.551	22717.802

Tabelle 5.3: Die Trajektorien des aufgenommenen Humboldt-Datensatzes in Zahlen

Eine Längeneinheit (LE) entspricht 1m. Der erstellte Datensatz umfasst 1830 Trajektorien. Die Ausdehnung beschreibt den größten Abstand zwischen zwei beliebigen Punkten einer Trajektorie und ist ein Maß für die Fläche, die von der Trajektorie eingenommen wird. P2P steht für Punkt-zu-Punkt.

5.4.1 Beschreibung des Aufnahmesystems

Zur Aufnahme wurde der Roboter „Ringo“ des Fachgebiets Neuroinformatik verwendet Horst-Michael Gross u. a. 2016. Dieser ist u.A. mit zwei Laserscannern zur Abstandsmessung, vier Kameras in Richtung der vier Kardinalrichtungen zur Personenerkennung und -verfolgung, sowie drei Tiefenkameras zur Hinderniserkennung ausgestattet (siehe auch Abbildung 5.12). Zwei leistungsstarke Rechner (8 Kerne, 8GB Ram) kümmern sich um die Personen- und Hinderniserkennung, Selbstlokalisation und Navigation. Zur Personenerkennung werden mehrere Featuredetektoren für Körperteile und Gesicht, sowie ein HOG-basierter (Histogram of oriented Gaussians) Algorithmus zur Erkennung der Silhouette, verwendet Volkhardt, C. Weinrich und H.-M. Gross 2013 Ch. Weinrich u. a. 2014 Wengefeld u. a. 2016. Wie bei allen Robotern des Fachgebiets

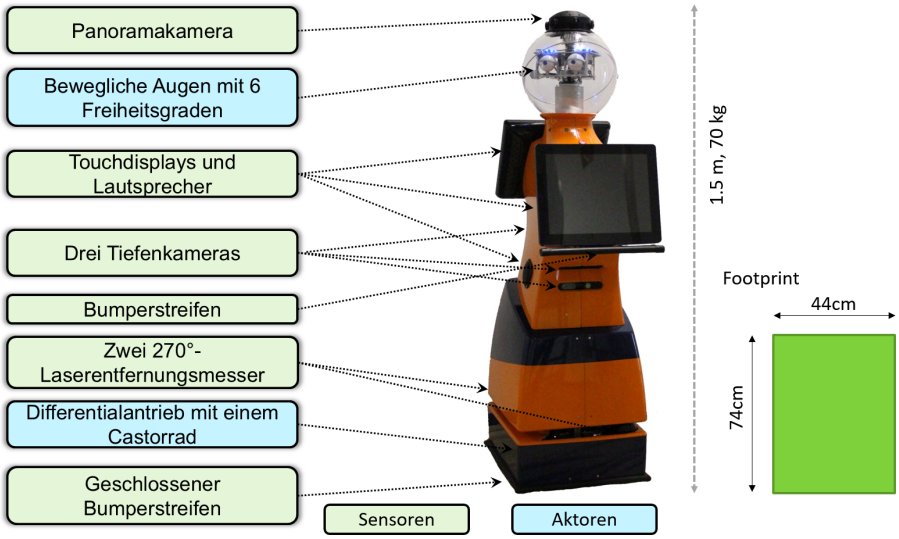


Abbildung 5.12: Sensoren ders Roboters Ringo

Ringo verfügt unter anderem über eine Panoramakamera, 3 Tiefenkameras und 2 270°-Lasersentfernungsmesser zur Navigation und Personen- und Hinderniserkennung Horst-Michael Gross u. a. 2016.

sind die verschiedenen Algorithmen und Module im „MIRA“-Framework eingebunden. Die Trajektorien wurden über einen Tag im Foyer des Humboldt-Baus der Technischen Universität Ilmenau während der Vorlesungszeit aufgenommen. Dabei steuerte der Roboter selbstständig zufällige Ziele auf der Karte an und zeichnete währenddessen die Positionen von erkannten Personen auf. Als Ziele wurden zufällige Punkte des Freiraumskeletts der Hinderniskarte verwendet. Ein Skelett ist in der Bildverarbeitung eine vereinfachte, jedoch topologisch korrekte Form des Eingangsbildes, die in der Regel mithilfe morphologischer Operatoren von einem Thinning-Algorithmus (dt. skelettieren) erzeugt wird. Diese werden häufig als Teil von Character Recognition Systemen verwendet, um die einem Buchstaben zu Grunde liegende Form zu extrahieren. Für diese Arbeit wurde der Algorithmus von Zhang und Suen Zhang und Suen 1984 verwendet. Abbildung 5.13 zeigt eine Beispielkarte und das aus ihr extrahierte Skelett.

Im Humboldt-Bau befinden sich zwei Hörsäle und ein Bistro, sodass sich in der Zeit zwischen den Vorlesungen hunderte von Personen im Foyer aufhielten, wohingegen es

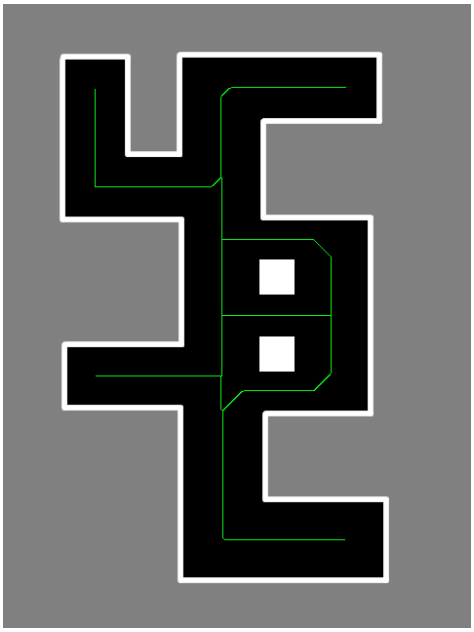


Abbildung 5.13: Skelett einer Beispielkarte
Skelett der Freiräume für eine Beispielkarte. Verwendet wurde der Algorithmus von Zhang und Suen 1984.

während der Vorlesungen nur vereinzelte Detektionen gab. Einige Fotos, die während der Aufnahmen entstanden sind, finden sich in 5.14, eine Visualisierung der Gebäudekarte und der aufgenommenen Trajektorien findet sich in 5.15.

5.4.2 Beobachtetes Verhalten

Um die Trajektorien möglichst unverfälscht aufzunehmen, navigierte der Roboter vollständig autonom und ohne sofort erkennbare Aufsichtspersonen. Dies brachte eine Reihe höchst interessanter Verhaltensmuster seitens der anwesenden Personen (vor allem Studenten) zu Tage, zu welchen sich mehrere weitere Abschlussarbeiten formulieren ließen. Da dies weit jenseits der Zielstellungen dieser Arbeit liegt, werden hier lediglich die auffälligsten beobachteten Verhaltensweisen kurz aufgeführt.

- Häufiges, absichtliches Versperren des Wegs
- Lockversuche, als wenn eine Katze angelockt wird

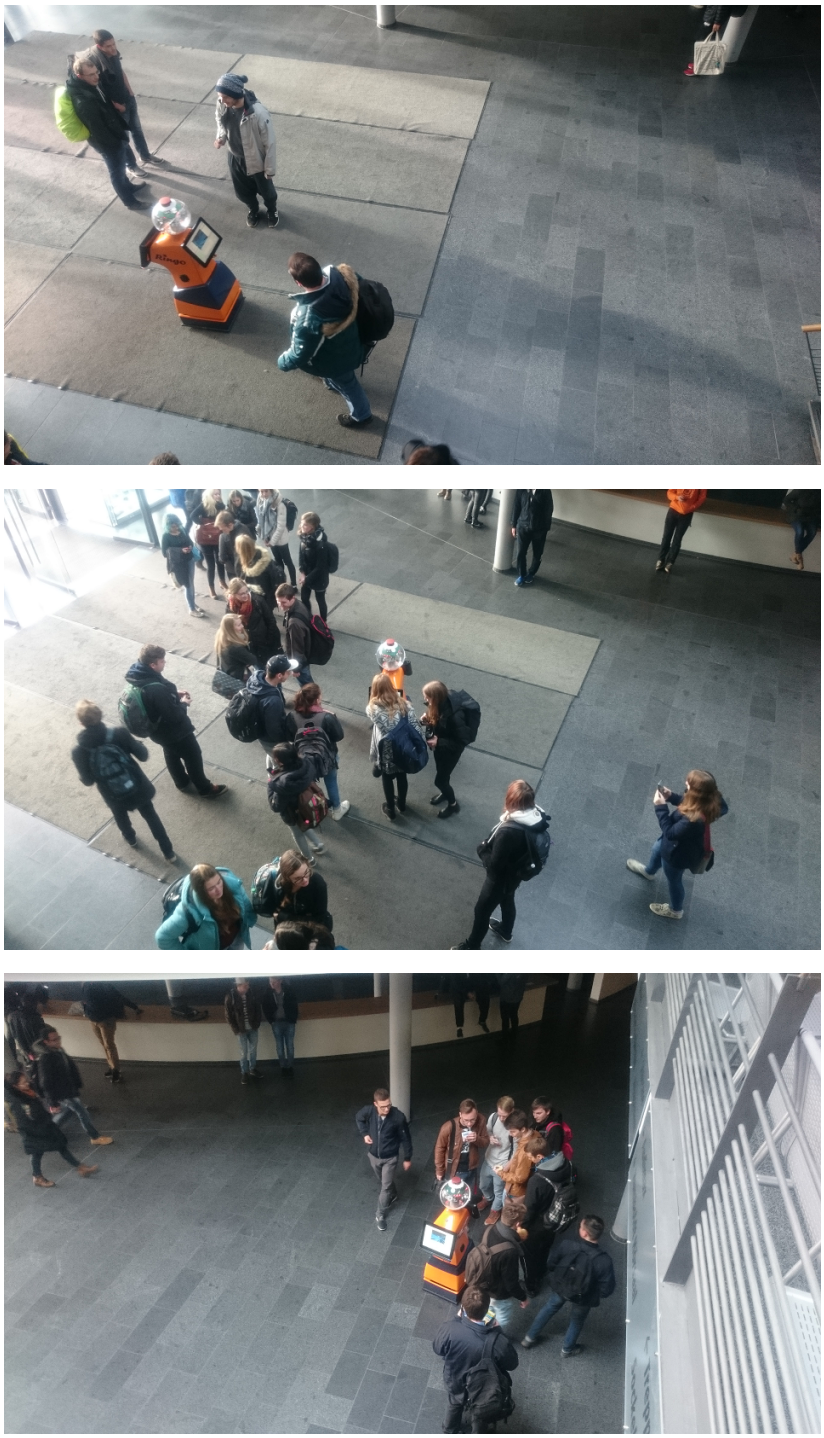
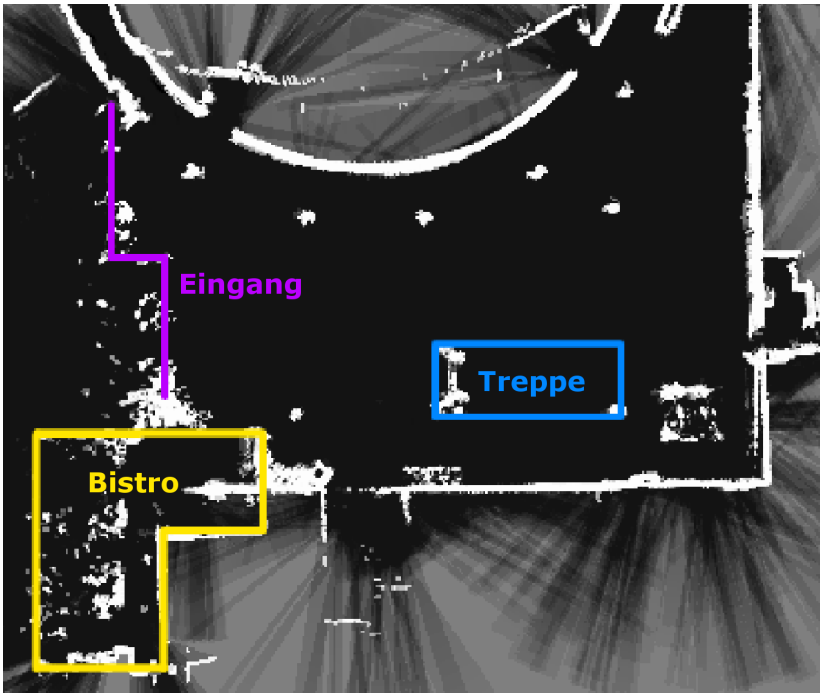
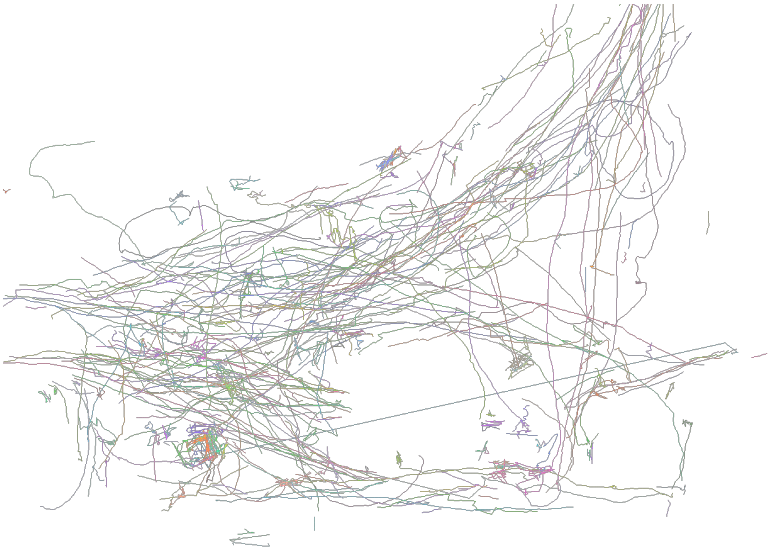


Abbildung 5.14: Fotos während der Datenaufnahme

Ringo wird mit verschiedenen Situationen konfrontiert. Die Aufnahmen erfolgten zu Zeiten mit mittlerem Personenaufkommen.



(a) Hinderniskarte des Humboldt-Baus



(b) 500 Trajektorien

Abbildung 5.15: Visualisierung des aufgenommenen Humboldt-Datensatzes

- Versuche, den Roboter zu umarmen
- Versuche, die Sicht zu versperren (Hand vor Kopf, Jacke vorhalten, etc.). Die Kameras befinden sich jedoch *auf* dem Kopf.
- Angedeutete Seitenhiebe
- Absichtliches, „provokantes“ Zugehen auf den Roboter, um Verhalten zu beobachten.
- Auf Roboter zu- und wieder weggehen („Kommt er jetzt mit?“)
- Umkreisen des Roboters
- Wenn das zufällige Ziel des Roboters innerhalb einer Personengruppe lag, so wurde dem Roboter belustigt Platz gemacht
- Mehrere Unterhaltungen im Sinne von „Ich will ihn mit nach Hause nehmen!“, „Ob man den schlagen kann?“
- Eine Handvoll Studenten haben wahllos auf dem Bildschirm herumgetippt (auf diesem war die Karte des Humboldt-Baus und die Position des Roboters zu sehen).
- Ein Student war dreist genug, den Notaus-Schalter zu betätigen

5.4.3 Einordnung

Der Datensatz besitzt mit etwa 2.000 Trajektorien einen wesentlich geringeren Umfang als die anderen vorgestellten Datensätze. Zudem stellt ein autonom navigierender Roboter eine Attraktion dar, welche viele Personen in ihrer Umgebung maßgeblich beeinflusst und von ihrer „natürlichen“ Trajektorie abbringt. Obwohl nach wie vor Hauptverkehrswege erkennbar sind, ist der Anteil atypischer Trajektorien insbesondere im Eingangsbereich, wo viele Personen nah an den Roboter herantraten, sehr hoch.

5.4.4 Auswertung

Aufgrund des geringen Volumens konnte für diesen Datensatz kein gutes Modell gelernt werden. Mit 1.800 der 1.830 Trajektorien und einem Graph auf einem Gitter von 4x4m konnten jedoch zumindest für längere Trajektorien korrekte Prädiktionen gemacht werden; für kurze Trajektorien, die in der Projektion nur einen Knoten lang sind, blieb die Prädiktion nahe an der Gleichverteilung. Zudem musste, um überhaupt sichtbare Ergebnisse zu erzielen, die Zuverlässigkeit der Bäume auf $50\% \pm 10\%$ eingestellt werden, was mindestens 50 Beobachtungen entspricht. Der Graph ist in Abbildung 5.16 dargestellt, Abbildung 5.17 zeigt drei Prädiktionsergebnisse unterschiedlicher Güte. An den ausgebildeten Kanten des Graphen lässt sich erkennen, dass die Trajektorien nur einen kleinen Bereich des Gebiets abgedeckt haben. Zu erkennen ist auch, wie gering bei einem Zuständigkeitsbereich von 4x4m die Aussagekraft einzelner Knoten bezüglich des Aufenthaltsortes einer Person ist. Da durch die Größe des Graphen auch lange Trajektorien meist nicht mehr als 1-2 Knoten zugeordnet werden, bringt eine CMC-Kurve in diesem Fall keinen Erkenntnisgewinn mehr und wurde daher ausgelassen.

5.4.5 Analyse

Im direkten Vergleich (siehe Abbildung 5.18 lassen sich jedoch Ähnlichkeiten zum Edinburgh-Datensatz erkennen. Ein 2012 im Humboldt-Bau von Konrad Schenk u. a. 2012 mithilfe mehrerer Laserrangescanner aufgenommener Datensatz ist ebenfalls gezeigt (siehe auch Tabelle 5.4).

Zunächst fällt auf, dass die vom Roboter aufgenommenen Trajektorien im Schnitt eine ähnliche Länge aufweisen wie die des Edinburgh-Datensatzes (Eigener Datensatz: 4,82m, Edinburgh: 5,07m, Konrad Schenk u. a. 2012: 18,38m). Dies folgt nicht zuletzt aus der begrenzten Wahrnehmungsdistanz des Roboters und unterliegt starken Schwankungen, welche auch davon abhängen, ob eine Person dem Roboter entgegenkommt oder in die selbe Richtung läuft. Die Aufnahmen der beiden Festinstallationen hatten zudem den Vorteil, dass es schwieriger für Personen war, aus dem Aufnahmebereich zu verschwinden (beim Edinburgh-Datensatz durch die Deckenmontage, beim Humboldt-Bau durch den Einsatz mehrerer Sensoren). Aus Sicht des Roboters ist es

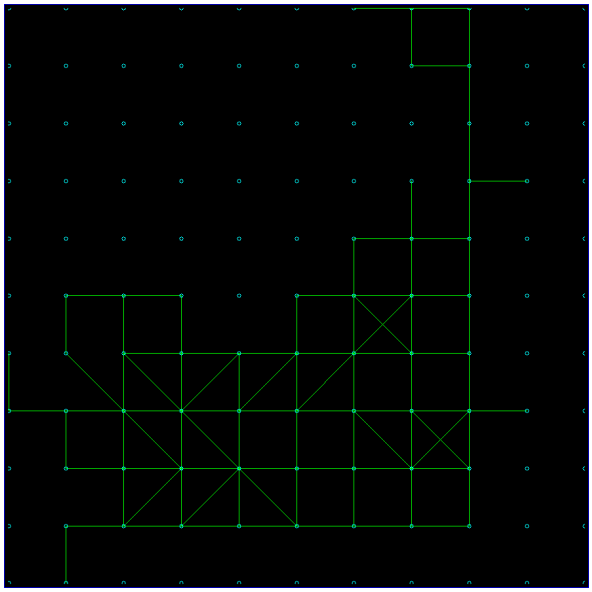


Abbildung 5.16: Topologischer Graph des eigenen Datensatzes
An den Kanten lässt sich erkennen, dass die Trajektorien nur einen kleinen Bereich des Gebiets abgedeckt haben. Mit einer Gittergröße von 4x4m sind einzelne Knoten nur wenig aussagekräftig bezüglich des Aufenthaltsortes einer Person.

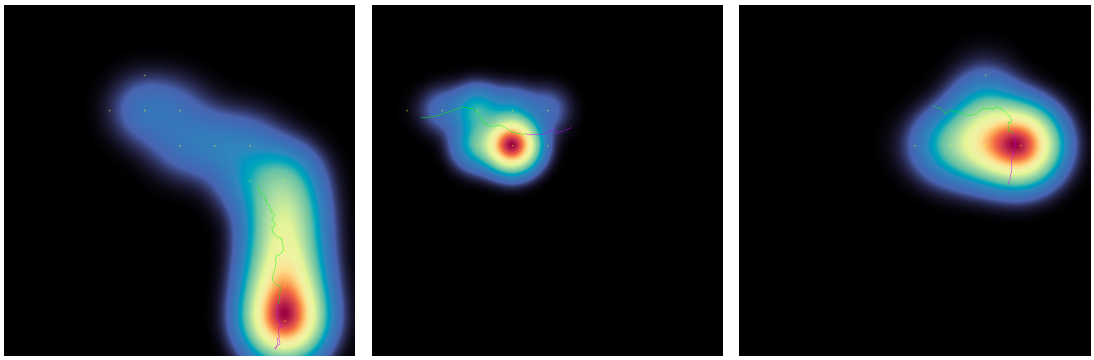
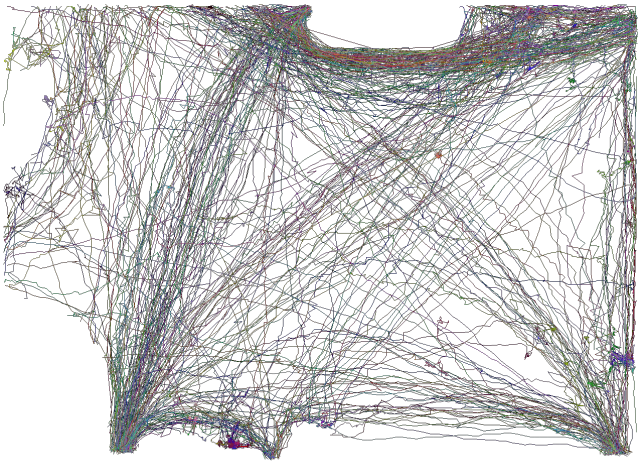
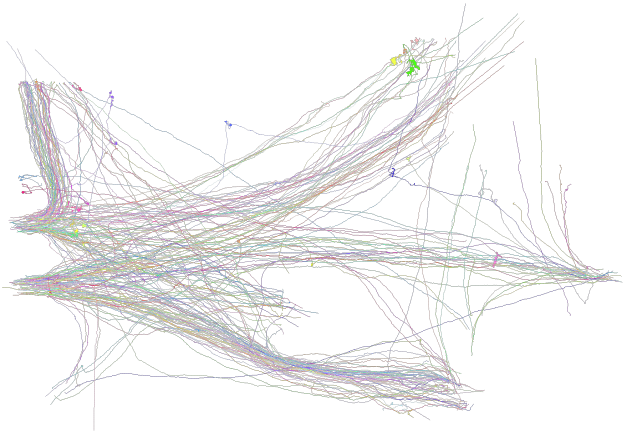


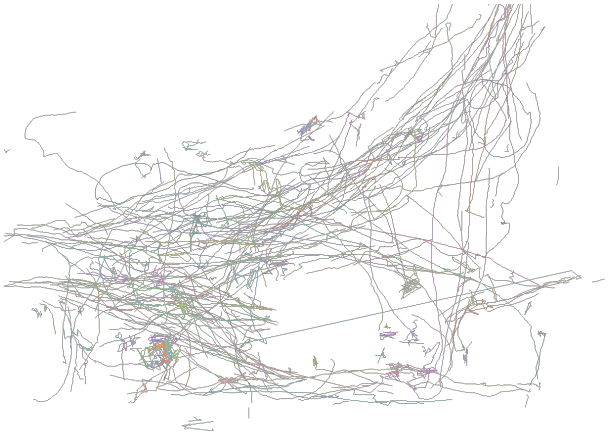
Abbildung 5.17: Prädiktionsbeispiele auf dem eigenen Datensatz
Prädiziert wurde auf einem Gittergraphen mit 4x4m. Bei dieser Gittergröße sind einzelne Knoten nur wenig aussagekräftig bezüglich des Aufenthaltsortes einer Person. Die Trajektorien sind in gezeigten (lila) und zu prädizierenden (grün) Teil geteilt.



(a) Trajektorien des Edinburgh-Datensatzes



(b) Datensatz des Humboldt-Baus von Konrad Schenk u. a. 2012 (Festinstallation)



(c) Eigener Datensatz des Humboldt-Baus (Roboter)

Abbildung 5.18: Vergleich der Edinburgh- und Humboldt-Trajektorien

	Gesamt	Min	Max	Mean	Varianz
Punkte	119790	87	6775	518.571	700.706
Länge (LE)	4245.17	2.83676	76.0269	18.377	9.500
Ausdehnung (LE)	3629.19	1.84531	27.905	15.711	7.170
Dauer (ms)	4.49e+06	3006	290304	19474.268	28584.127
Ausrichtung (rad)	46.5164	-2.18805	2.1064	0.201	1.335
Geschwindigkeit (LE/s)	2261.92	-2.61354	53.5031	9.792	7.791
P2P-Abstand (LE)	4245.17	1.04e-05	0.40616	0.036	0.029
P2P-Verzögerung (ms)	4.49e+06	5	104	37.625	18.902

Tabelle 5.4: Die Trajektorien des Humboldt-Datensatzes von Konrad Schenk u. a. 2012 in Zahlen

Eine Längeneinheit (LE) entspricht 1m. Der Datensatz umfasst 231 Trajektorien.

für Personen jedoch leicht, zu verschwinden (beispielsweise hinter einer Säule oder einer anderen Person). Eine Person, welche wieder auftauchte, führte zum Start einer neuen Trajektorie.

Auch wird deutlich, dass die vom Roboter aufgezeichneten Trajektorien wesentlich häufigere Richtungswechsel und Unstetigkeiten aufweisen. Die von der Festinstallation aufgenommenen Trajektorien sind dahingegen vergleichsweise glatt. Um dies mit Zahlen zu belegen, wurde die absolute Winkeländerung bezogen auf die Länge der Trajektorie berechnet und über alle Trajektorien gemittelt. Die Werte sind in Tabelle 5.5 aufgeführt. Zu sehen ist, dass die beobachtete mittlere Winkeländerung mit rund 7% oder 25° beim eigenen Datensatz signifikant höher liegt. Außerdem ist die maximale Winkeländerung pro Längeneinheit mit 3,5rad (201°) weit über den beobachteten maximal 2,08rad (119°) des älteren Datensatzes.

	Gesamt	Min	Max	Mean	Varianz
Eigener Datensatz	1333.67	0	3.5083	0.729	0.559
Konrad Schenk u. a. 2012	157.055	0	2.08749	0.680	0.549

Tabelle 5.5: Absolute Winkeländerung pro Längeneinheit
Die Einheiten sind in rad/LE (Längeneinheit), wobei eine Längeneinheit 1m entspricht. Erfasst wurden alle Trajektorien des jeweiligen Datensatzes.

5.4.6 Schlussfolgerung

Die Lokalisierung von Hindernissen und Personen im Raum ist auf einer mobilen Plattform nicht nur unmittelbar von der Genauigkeit der Messsysteme abhängig, sondern auch mittelbar von der Selbstlokalisation des Roboters, und unterliegt somit naturgemäß größeren Schwankungen, als es bei vergleichbaren Festinstallationen der Fall wäre. Da jedoch die Selbstlokalisation des Roboters auch bei großem Personenaufkommen stabil blieb und die sichtbaren Abweichungen weit über das zu erwartende Maß hinausgehen, muss davon ausgegangen werden, dass die Anwesenheit des Roboters selbst für diese Abweichungen verantwortlich ist. Dies erlaubt den Schluss, dass der Roboter die Agenda von Personen in seiner Umgebung verändert und einen mobilen „Point of Interest“ darstellt. Eine Person verhält sich daher anders, wenn der Roboter sich in ihrer Nähe aufhält (siehe Abbildung 5.19). Dies ist in sofern problematisch, als dass der vorgestellte Algorithmus versucht, ein umgebungsabhängiges Modell zu lernen. Da die Änderungen spontan sind, sind sie zudem weitgehend unabhängig von der Vorgeschichte. Die hohe Dynamik, welche vom Roboter in die Umgebung eingebracht wird, lässt sich daher mit dem bisherigen Ansatz nicht abbilden. Die Schlussfolgerung ist, dass der Algorithmus nicht ohne weiteres auf einem Roboter eingesetzt werden kann und ein Modell, welches mögliche Ziel- und Interessepunkte berücksichtigt, prinzipiell besser geeignet wäre.

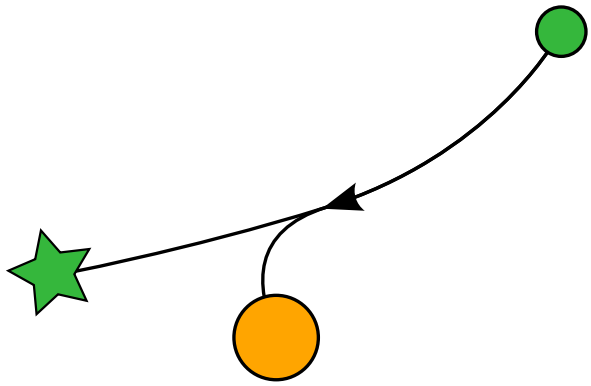
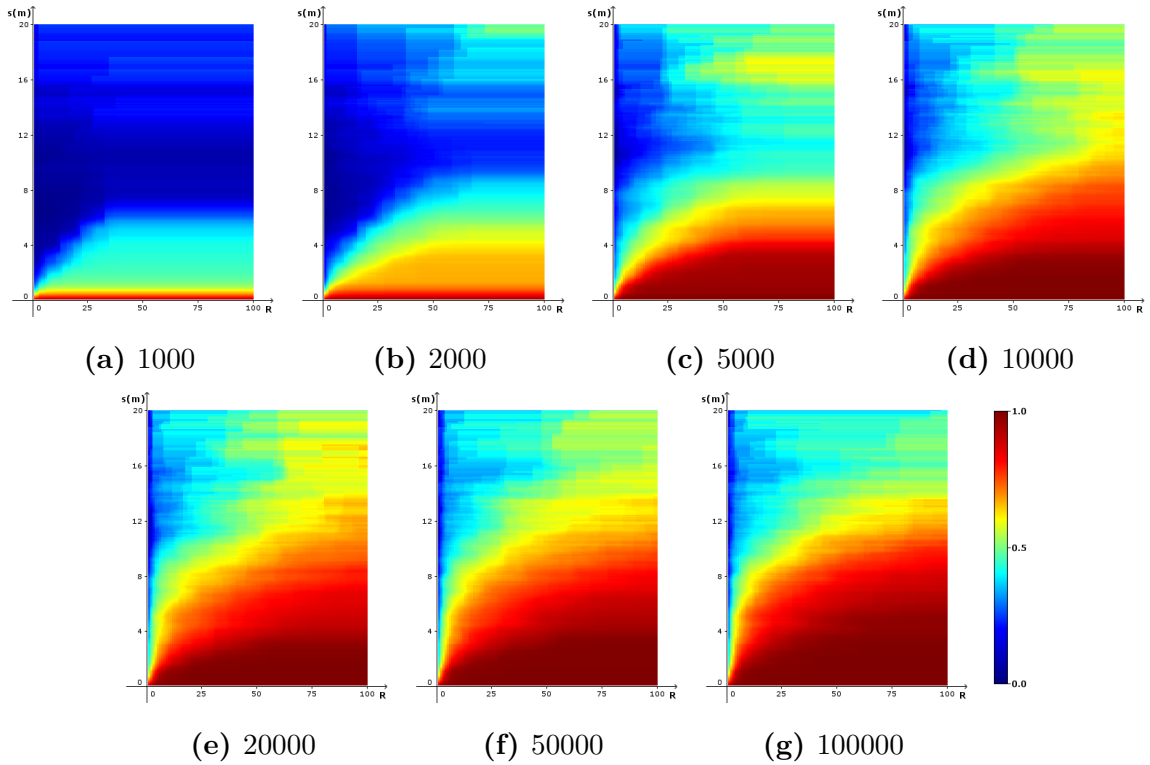


Abbildung 5.19: Ablenkung einer Person durch den Roboter
Die Anwesenheit eines Roboters führt zur Abweichung einer Person von ihrer ursprünglich geplanten, „natürlichen“ Trajektorie.
grüner Kreis: Person; grüner Stern: ursprüngliches Ziel; orangener Kreis: Roboter

5.5 Inkrementelles Lernen

Um das Lernverhalten des Algorithmus genauer zu untersuchen, wurde für den Edinburgh-Datensatz ein Graph mit einem 40x40 Gitter erstellt. Der Graph wurde anschließend in mehreren Intervallen trainiert. Nach 1.000, 2.000, 5.000, 10.000, 20.000, 50.000 und 100.000 Trajektorien wurde jeweils eine CMC-Kurve für die Lokalisationsgenauigkeit über 1000 vom Training ausgeschlossene Trajektorien, erzeugt. Die Kurven in Abbildung 5.20 zeigen, dass auf diesem Datensatz während der ersten 20.000 Trajektorien die größten Lernerfolge erzielt werden. Weiteres Training verbessert zwar noch die Prädiktion, verschlechtert allerdings auch die Prädiktion jenseits von etwa 12,5m. Dies lässt sich durch die Reduktion von Streuflüssen und die Seltenheit entsprechender Trajektorien als Lernbasis erklären.
Dies wird so interpretiert, dass die tieferen Ebenen der Markov-Bäume, aufgrund der starken Verzweigung, wesentlich mehr Trainingsdaten benötigen, um sichere Vorhersagen treffen zu können (siehe auch Gleichung 4.1). Dies wird auch durch das Verhältnis von Samples je Markov-Knoten und Ebene über den Verlauf des Trainings bestätigt (siehe Abbildung 5.21). Auch ist der Einsatz beliebig tiefer Markov-Trees nicht sinnvoll, da zum Training entsprechend lange Trajektorien benötigt werden: um einen Baum mit 5 Ebenen zu trainieren, muss die Projektion mindestens 5 Knoten lang sein, was beim



(a) 1000 **(b)** 2000 **(c)** 5000 **(d)** 10000

(e) 20000 **(f)** 50000 **(g)** 100000

Abbildung 5.20: Lokalisationsgenauigkeit nach verschiedenen Trainingseinheiten

Der Hauptteil des Trainings ist nach etwa 20.000 Trajektorien abgeschlossen.

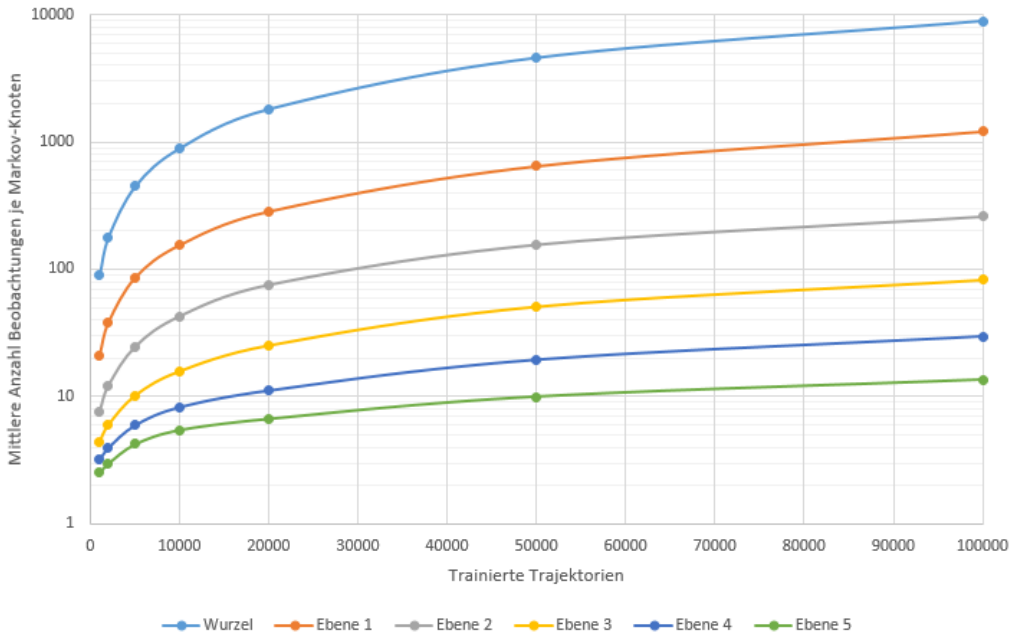


Abbildung 5.21: Beobachtungen in den Ebenen der Markov-Trees
Durch die kombinatorischen Möglichkeiten sind tiefere Ebenen deutlich schlechter trainiert als höhere Ebenen. Besonders wichtig ist in diesem Fall die Linie bei 100 Beobachtungen, da dies die sich aus der Chebyshev-Ungleichung (4.1) ergebende Grenze für die Beteiligung eines Markov-Knotens an der Prädiktion ist (für eine Sicherheit von $75\% \pm 10\%$*).*

Edinburgh-Datensatz auf dem 40x40-Graphen auf kürzestem Weg (direkt von Knoten zu Knoten) 4,94m entspricht. Die durchschnittliche Länge liegt mit 5,07m nur knapp darüber. Da sich, wie weiter oben festgestellt, die Knoten in freien Räumen nicht so positionieren lassen, dass sie die Wege der beobachteten Personen exakt beschreiben, müssen die Trajektorien entsprechend länger ausfallen, um 5 Knoten abzudecken.
Augenscheinlich bestehen Ähnlichkeiten zwischen diesem Problem und dem in der Neuroinformatik bekannten Vanishing Gradient Problem (siehe Hochreiter 1991). In diesem Fall versickert das Training jedoch aufgrund der kombinatorischen Vervielfältigung möglicher Pfade statt aus wiederholten Multiplikationen und ist daher nicht direkt mit diesem vergleichbar.

5.6 Zusammenfassung

Zur Auswertung wurden drei frei verfügbare Datensätze verwendet: die Edinburgh Informatics Forum Pedestrian Database 5.3, der Beijing Taxi Driver Datensatz 5.2 und ein selbst im Humboldt-Bau der TU Ilmenau mit Hilfe eines Roboters aufgenommener Datensatz 5.4. Während sich beim Beijing-Datensatz aufgrund des Maßstabes klar abgegrenzte Verkehrswege ergeben, können diese beim Edinburgh- und beim Humboldt-Datensatz nicht nachgewiesen werden. Trotz des höheren Bewegungsspielraums konnten auf dem Edinburgh-Datensatz gute Prädiktionen erzielt werden 5.3.2. Mit dem Humboldt-Datensatz konnte, aufgrund seines geringen Volumens, kein gutes Modell gelernt werden. Die Analyse zeigt jedoch, dass die Verwendung eines ortsgebundenen Modells auf einem Roboter höchst problematisch ist, da dieser als mobiler Attraktionspunkt eine hohe Dynamik in die Umgebung einbringt und die Bewegung von Personen in seiner Umgebung maßgeblich beeinflusst 5.4.5.

Kapitel 6

Abschluss

6.1 Zusammenfassung

Auf der Grundlage des von K. Schenk 2017 beschriebenen Algorithmus wurde ein flexibles Framework zur Prädiktion von (Personen-)Trajektorien entwickelt. Grundlage der Prädiktion ist die räumliche Abtastung der Einsatzumgebung in Form eines Graphen. Dieser erlaubt es, ähnliche Trajektorien zusammenzufassen und so kohärente Modelle zu lernen. Als bislang zuverlässigster Prädiktionsalgorithmus wurden die im Original verwendeten Markov-Trees implementiert, welche als gebündelte Markov-Ketten verstanden werden können und ein ortsabhängiges Transitionsmodell lernen 3.3. Das Framework erlaubt die Implementierung weiterer Prädiktionsalgorithmen, welche auch mit anderen kombiniert werden können. Zur Verarbeitung des Graphen stellt das Framework eine Vielzahl von Modulen bereit, um z.B. Knoten und Kanten zu erzeugen, Analysen durchzuführen oder Trajektorien zu prädizieren 4.2.

Die Analyse des Codes zeigte, dass die Leistungsfähigkeit des Algorithmus maßgeblich von der Anzahl zu berechnender Flüsse abhängt 4.4. Kleinste Teilflüsse in der Größenordnung 10^{-6} und kleiner bilden sich jedoch schnell im gesamten Graphen aus und führen zu erheblichen Performance-Einbußen. Durch die Verwendung eines einstellbaren Grenzwertes können diese Flüsse von der weiteren Berechnung ausgeschlossen werden. Die Prädiktionsergebnisse verlieren dadurch nur wenig an

Aussagekraft, während die Laufzeit deutlich verbessert wird.

Während die prädizierten Aufenthaltsorte eine hohe Güte aufweisen, liegen die zeitlichen Prädiktionen für Personen hinter den Erwartungen zurück 5.3.2. Dies liegt an der Beschaffenheit des verwendeten Datensatzes: während sich im Beijing-Datensatz auf dem verwendeten Maßstab (10-100m) deutliche, scharf abgegrenzte Pfade abbilden, ist dies für große Freiräume, in denen sich Menschen bewegen, in der Regel nicht der Fall. Dies führt dazu, dass sich Personen oft nicht direkt von Knoten zu Knoten, sondern stattdessen zwischen mehreren Knoten hindurchlaufen und somit stark abweichende Transitionszeiten erzeugen können.

Um die benötigte Datenmenge zu reduzieren, wurde zunächst nachgewiesen, dass mit dem Verfahren gute Prädiktionen unter Verwendung von generischen Graphen, die nicht durch Clusterer erzeugt wurden, möglich sind 5.3.3. Dies erleichtert die Initialisierung erheblich, da für diesen Schritt keine Trainingsdaten mehr benötigt werden. Der Aufbau des Frameworks erlaubt es außerdem, andere Prädiktionsmodelle zu verwenden, um die Markov-Trees zu unterstützen oder zu ersetzen, bis diese ausreichend Training erhalten haben 4.3. Da die Markov-Trees, wie in 5.5 gezeigt, inkrementelles Lernen unterstützen, kann dieser Ansatz somit auch online verwendet werden.

Durch die Sammlung von Trajektorien mithilfe eines Roboters für einen eigenen Datensatz stellte sich heraus, dass die gewonnenen Daten nicht ohne weiteres zum Training ortsgebundener Modelle verwendet werden können 5.4.5: der Roboter stellt einen mobilen Attraktionspunkt dar und verändert die Bewegung von Personen in seiner Umgebung maßgeblich. Diese hohe Umgebungsdynamik kann durch ortsgebundene Modelle wie z.B. die in dieser Arbeit verwendeten Markov-Trees nicht abgebildet werden. Ein Modell, welches mögliche Interessenpunkte lernt und berücksichtigt, wäre daher für den geplanten Einsatzzweck besser geeignet.

6.2 Ausblick

Das Framework bietet vielfältige Möglichkeiten für Erweiterungen. Für einige der in dieser Arbeit angesprochenen Probleme existieren bereits Ideen, wie diese zukünftig gelöst werden könnten. Im Folgenden werden diese kurz vorgestellt.

Prädiktion auf Robotern Wie in Abschnitt 5.4.5 beschrieben, ist der Einsatz eines ortsgebundenen Modells auf einem Roboter problematisch, da er die Trajektorien der Personen in seiner Umgebung sehr stark beeinflusst. Obwohl bei einem regelmäßgen Aufenthalt am Einsatzort voraussichtlich ein Gewöhnungseffekt eintreten wird, ist ein hoher Anteil atypischer Trajektorien zu erwarten. Dies ließe sich noch durch eine höhere Menge von Trainingsdaten statistisch ausgleichen. Die spontane Änderung der Agenda lässt sich jedoch durch den starken Verlass auf die Vorgeschichte nicht durch die Markov-Trees prädizieren. Die Idee wäre daher, zusammen mit weiteren Prädiktionsalgorithmen (welche auch im Framework implementiert werden können) ein Ensemble zu bilden. So könnte z.B. über ein trainigsfreies, physikalisches Modell eine Prädiktion mit kurzem Vorhersagehorizont für die nächsten 1-2s erstellt werden, während andere Modelle Prädiktionen mit weiterem Horizont liefern. Auf diese Weise könnten auch kurzfristige Änderungen der Bewegungsrichtung noch berücksichtigt werden.

Verständnis der Markov-Trees Die Arbeitsweise der ortsgebundenen Markov-Trees legt nahe, dass das von ihnen gelernte Modell die Abtastung einer hochdimensionalen Funktion der Form $f(x, y, N_{t-\tau}, N_{t-\tau+1}, ..., N_t)$ darstellt, welche also vom Raum und der Vorgeschichte abhängt. Derartige Funktionen können auch gut durch rekurrente neuronale Netze (RNN, z.B. Long Short-Term Memory (LSTM) Networks) approximiert werden. Interessant wäre daher eine Untersuchung, ob mit derartigen Netzen vergleichbare Ergebnisse erzielt werden können.

Weitere Modelle Das Framework bietet sich an, um weitere Modelle zu implementieren und zu untersuchen. Insbesondere die Möglichkeit, Ensemble-Modelle zu

erstellen, scheint dabei vielversprechend. Wie zuvor beschrieben, sind vorhergesagte Transitionszeiten insbesondere in freien Räumen nicht zufriedenstellend. Über eine Ausgleichsrechnung auf der beobachteten Trajektorie könnten eventuell genauere Transitionszeiten vorhergesagt werden, eine wirklich zündende Idee fehlt bislang jedoch.

Ortsbindung des Graphen Prinzipiell muss der Referenzpunkt des Graphen nicht ortsgebunden sein. Über die von MIRA bereitgestellten Transformationen könnte der Graph auch an den Roboter gebunden und somit beim Herumfahren mitgeführt werden. Dies würde es erlauben, Prädiktionen im Umfeld des Roboters durchzuführen. Durch diesen Ansatz würde allerdings der Einsatz ortsgebundener Prädiktionsmodelle unterbunden werden, da diese nur schwer auf die sich ändernde Umgebung eingehen können. Die Hauptaufgabe des Graphen, die Diskretisierung von Trajektorien, würde jedoch erhalten bleiben und könnte einen Ansatzpunkt für andere Modelle liefern.

6.3 Fazit

Die Implementierung eines komplexen Algorithmus auf der Grundlage lediglich einer (vorläufigen) Quelle stellte sich trotz der Unterstützung des Autors als unerwartet fordernd heraus. Da es für die Implementierung nur wenige Vorgaben gab, musste außerdem für diese eine geeignete Form gefunden werden, welche sowohl die Anforderungen der Abschlussarbeit als auch die Ziele des Fachgebiets Neuroinformatik abdeckt. In einem iterativen Entwicklungsprozess entstand so nach mehreren Monaten das nun vorliegende modulare Framework. Der damit verbundene Implementierungsaufwand musste auch zeitlich in den Rahmen der Arbeit eingepasst werden, sodass für die Erstellung eines eigenen Datensatzes weniger Zeit verfügbar war als geplant. Trotz all dieser Herausforderungen konnten die Ziele dieser Arbeit erfüllt werden. Insbesondere die Erfahrungen in den Bereichen Zeit- und Projektmanagement werden sich sicherlich zukünftig bewähren.

Abbildungsverzeichnis

Literatur

Beckmann, Norbert u. a. (1990). „The R∗-tree: an efficient and robust access method for points and rectangles“. In: *ACM SIGMOD Record* 19.2, S. 322–331. DOI: 10.1145/93605.98741. URL: http://dx.doi.org/10.1145/93605.98741.

Browning, R. C. (2006). „Effects of obesity and sex on the energetic cost and preferred speed of walking“. In: *Journal of Applied Physiology* 100.2, S. 390–398. DOI: 10.1152/japplphysiol.00767.2005. URL: http://dx.doi.org/10.1152/japplphysiol.00767.2005.

Burgess, René G. und Christian J. Darken (2004). „Realistic Human Path Planning Using Fluid Simulation“. In: *Proceedings of Behavior Representation in Modeling and Simulation BRIMS*.

Comaniciu, D., V. Ramesh und P. Meer. „The variable bandwidth mean shift and data-driven scale selection“. In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. Institute of Electrical und Electronics Engineers (IEEE). DOI: 10.1109/iccv.2001.937550. URL: http://dx.doi.org/10.1109/ICCV.2001.937550.

DeCann, Brian und Arun Ross (2013). „Relating ROC and CMC curves via the biometric menagerie“. In: *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*. Institute of Electrical und Electronics Engineers (IEEE). DOI: 10.1109/btas.2013.6712705. URL: http://dx.doi.org/10.1109/BTAS.2013.6712705.

Einhorn, Erik u. a. (2012). „MIRA - middleware for robotic applications“. In: http://mira-project.org. IEEE, S. 2591–2598. ISBN: 978-1-4673-1736-8, 978-1-4673-1737-5, 978-1-4673-1735-1. DOI: 10.1109/IROS.2012.6385959.

Ferrer, Gonzalo und Alberto Sanfeliu (2014). „Bayesian Human Motion Intentionality Prediction in urban environments“. In: *Pattern Recognition Letters* 44, S. 134–140. DOI: 10.1016/j.patrec.2013.08.013. URL: http://dx.doi.org/10.1016/j.patrec.2013.08.013.

Gross, Horst-Michael u. a. (2016). „ROREAS: robot coach for walking and orientation training in clinical post-stroke rehabilitation—prototype implementation and evaluation in field trials“. In: *Autonomous Robots*. DOI: 10.1007/s10514-016-9552-6. URL: http://dx.doi.org/10.1007/s10514-016-9552-6.

Guttman, Antonin (1984). „R-trees: A Dynamic Index Structure for Spatial Searching“. In: *INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA*. Bd. 14. Association for Computing Machinery (ACM), S. 47–57. DOI: 10.1145/971697.602266. URL: http://dx.doi.org/10.1145/971697.602266.

Helbing, Dirk und Péter Molnár (1995). „Social force model for pedestrian dynamics“. In: *Physical Review E* 51.5, S. 4282–4286. DOI: 10.1103/physreve.51.4282. URL: http://dx.doi.org/10.1103/PhysRevE.51.4282.

Hochreiter, Sepp (1991). „Untersuchungen zu dynamischen neuronalen Netzen“. Magisterarb. Technische Universität München.

Ikeda, Tetsushi u. a. (2012). „Modeling and Prediction of Pedestrian Behavior based on the Sub-goal Concept“. In: *Proceedings of Robotics: Science and Systems*. Sydney, Australia. DOI: 10.15607/RSS.2012.VIII.018.

Lefèvre, Stéphanie, Dizan Vasquez und Christian Laugier (2014). „A survey on motion prediction and risk assessment for intelligent vehicles“. In: *ROBOMECH Journal* 1.1. DOI: 10.1186/s40648-014-0001-z. URL: http://dx.doi.org/10.1186/s40648-014-0001-z.

Meuter, Mirko u. a. (2008). „The unscented Kalman filter for pedestrian tracking from a moving host“. In: *2008 IEEE Intelligent Vehicles Symposium*. Institute of Electrical und Electronics Engineers (IEEE). DOI: 10.1109/ivs.2008.4621191. URL: http://dx.doi.org/10.1109/IVS.2008.4621191.

Salas, Joaquin u. a. (2007). „Detecting Unusual Activities at Vehicular Intersections“. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*.

Institute of Electrical und Electronics Engineers (IEEE). DOI: 10.1109/robot.2007.363094. URL: http://dx.doi.org/10.1109/ROBOT.2007.363094.

Schenk, K. (2017). „Beitrag zur Langzeitprädiktion von Bewegungstrajektorien“. Vorläufige Version.

Schenk, Konrad u. a. (2012). „Automatic Calibration of Multiple Stationary Laser Range Finders Using Trajectories“. In: *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance*. Institute of Electrical und Electronics Engineers (IEEE). DOI: 10.1109/avss.2012.14. URL: http://dx.doi.org/10.1109/AVSS.2012.14.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Springer Science + Business Media. DOI: 10.1007/978-1-4899-3324-9. URL: http://dx.doi.org/10.1007/978-1-4899-3324-9.

Sutter, Herb und Andrei Alexandrescu. *C++ Coding Standards: 101 Rules, Guidelines, and Best Practices*, S. 195–207. ISBN: 978-0321113580.

Team, NSTB/WAAS T&E (2014). *Global Positioning Standard (GPS) Standard Positioning Service (SPS) Performance Analysis Report*. Techn. Ber. 86. William J. Hughes Technical Center.

Trautman, Peter und Andreas Krause (2010). „Unfreezing the robot: Navigation in dense, interacting crowds“. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Institute of Electrical und Electronics Engineers (IEEE). DOI: 10.1109/iros.2010.5654369. URL: http://dx.doi.org/10.1109/IROS.2010.5654369.

Volkhardt, M., C. Weinrich und H.-M. Gross (2013). „Multi-modal people tracking on a mobile companion robot“. In: *Mobile Robots (ECMR), 2013 European Conference on*, S. 288–293. DOI: 10.1109/ECMR.2013.6698856.

Weinrich, Christoph u. a. (2014). „People Detection and Distinction of their Walking Aids in 2D Laser Range Data based on Generic Distance-Invariant Features“. In: *IEEE Int. Symp. on Robot and Human Interactive Communication (RO-MAN)*. IEEE, S. 767–773.

Wengefeld, Tim u. a. (2016). „May I be your Personal Coach? Bringing Together Person Tracking and Visual Reidentification on a Mobile Robot“. In: *Int. Symposium on Robotics (ISR)*. VDE, S. 141–148.

Zhang, T. Y. und C. Y. Suen (1984). „A Fast Parallel Algorithm for Thinning Digital Patterns“. In: *Commun. ACM* 27.3, S. 236–239. ISSN: 0001-0782. DOI: 10.1145/357994.358023. URL: http://doi.acm.org/10.1145/357994.358023.