Bundesamt
für Strahlenschutz

Ressortforschungsberichte zum Strahlenschutz

Auswertung von Schilddrüsen-Messungen und Bewegungsprofilen zur Ermittlung der Iod-Expositionen mit KI-Verfahren

Vorhaben 3622S62583

singularIT GmbH

Das Vorhaben wurde mit Mitteln des Bundesministeriums für Umwelt, Klimaschutz, Naturschutz und nukleare Sicherheit (BMUKN) und im Auftrag des Bundesamtes für Strahlenschutz (BfS) durchgeführt.

Impressum

Bundesamt für Strahlenschutz
Postfach 10 01 49
38201 Salzgitter

Tel.: +49 30 18333-0
Fax: +49 30 18333-1885
E-Mail: ePost@bfs.de
De-Mail: epost@bfs.de-mail.de

www.bfs.de

BfS-RESFOR-246/25

Bitte beziehen Sie sich beim Zitieren dieses Dokumentes immer auf folgende URN:
urn:nbn:de:0221-2025052352265

Salzgitter, Mai 2025

Abschlussbericht

Auswertung von Schilddrüsen-Messungen und Bewegungsprofilen zur Ermittlung der Iod-Expositionen mit KI-Verfahren

Ausschreibende Stelle:

Bundesamt für Strahlenschutz

Erstellt von:

singularIT GmbH
Inselstraße 27
04103 Leipzig

Zusammenfassung

Bei Unfällen in kerntechnischen Anlagen können radioaktive Iod-Isotope freigesetzt werden. Diese verbreiten sich über die Luft und können von Personen in den betroffenen Gebieten eingeatmet werden, was unter anderem zu Krebserkrankungen der Schilddrüse führen kann. Die Bestimmung der räumlichen und zeitlichen Verbreitung dieser Iod-Isotope ist daher von hoher Bedeutung für einen guten Umgang mit einem aus einem solchen Unfall resultierenden Notfall, um entsprechende Maßnahmen zum Schutz der Bevölkerung einzuleiten. Wir haben uns konkret mit der Rekonstruktion der Iod-Konzentration aus Bewegungsprofilen von betroffenen Personen befasst. Eine Schwierigkeit ist, dass keine Daten von echten Notfällen existieren. Deshalb haben wir zunächst einen hochkonfigurierbaren Simulator entwickelt, der sowohl korrekte als auch fehlerbehaftete realistische Bewegungsprofile generieren kann. Basierend auf den simulierten Bewegungsprofilen haben wir erstmals Methoden entwickelt und untersucht, die aus Bewegungs- und Expositionsdaten betroffener Personen die räumliche und zeitliche Verteilung der Iod-Isotope rekonstruieren können. Unsere Ergebnisse zeigen, dass eine Rekonstruktion der Iod-Konzentrationen möglich ist, aber durch die Anzahl der zu einem Notfall verfügbaren Personendaten limitiert ist. Die Schwierigkeit des Problems liegt in dem ungünstigen Verhältnis zwischen der hohen gewünschten Auflösung der Rekonstruktion und der knappen Verfügbarkeit der dafür notwendigen Daten von betroffenen Personen. Vor allem mittels neuartiger KI-Methoden aus den Bereichen Neural Radiance Fields und Gaussian Splatting konnten wir die besten Rekonstruktionen, gemessen anhand multipler Bewertungsmetriken, erreichen.

Abstract

In the event of accidents at nuclear facilities, radioactive iodine isotopes can be released. These spread through the air and can be inhaled by people in the affected areas, potentially leading to thyroid cancer, among other health issues. Therefore, determining the spatial and temporal distribution of these iodine isotopes is of great importance for effectively managing emergencies resulting from such accidents and implementing protective measures for the population. Our research specifically focused on reconstructing iodine concentration from the movement profiles of affected individuals. One challenge is that no data from real emergencies exist. Therefore, we first developed a highly configurable simulator capable of generating both accurate and error-prone realistic movement profiles. Based on these simulated movement profiles, we have for the first time developed and investigated methods that can reconstruct the spatial and temporal distribution of iodine isotopes using the movement and exposure data from affected individuals. Our results show that reconstruction of iodine concentrations is possible but limited by the amount of personal data available during an emergency. The complexity of the problem lies in the unfavorable ratio between the high desired resolution of the reconstruction and the limited availability of necessary data from affected individuals. Notably, using innovative AI methods from the fields of Neural Radiance Fields and Gaussian Splatting, we achieved the best reconstructions, as measured by multiple evaluation metrics.

Inhalt

Hinweis: Der Bericht gibt die Auffassung und Meinung des Auftragnehmers wieder und muss nicht mit der Meinung der Auftraggeberin übereinstimmen.

1. Einführung in die Problemstellung

1.1 Beschreibung

Bei Unfällen in kerntechnischen Anlagen können radioaktive Stoffe, vor allem verschiedene Isotope des Iods, freigesetzt werden. Es ist wichtig, möglichst genaue Informationen über die Ausbreitung dieser radioaktiven Stoffe zu bekommen. Aktuell werden diese Informationen mittels Simulationen, Wetterdaten und Kontrollmessungen abgeschätzt. Außerdem werden personenbezogene Datenerhebungen in Notfallstationen durchgeführt. Personen aus betroffenen Gebieten werden dazu aufgefordert, sich zu diesen Notfallstationen zu begeben. Dort wird dann ein Bewegungsprofil erstellt und eine Messung der Aktivität des Iods in der Schilddrüse vorgenommen. Die Kernfrage dieser Arbeit ist:

> Kann aus den Bewegungsprofilen und Aktivitätsmessungen der Schilddrüse der Personen die Verteilung der Iod-Konzentration in der Luft sowohl räumlich als auch zeitlich rekonstruiert werden?

Die Bewegungsprofile setzen sich aus verschiedenen Informationen zusammen. Zunächst wird abschnittsweise der Weg einer Person mit Koordinaten und Zeitangaben erfasst. Dazu werden mehrere Informationen erhoben, mit deren Hilfe abgeschätzt werden soll, wie sehr eine Person dem radioaktiven Iod ausgesetzt war. So ist zu vermuten, dass sich ein Aufenthalt in einem Keller anders auf die Aufnahme von Iod auswirkt als ein Spaziergang im Freien. Als besondere Schutzmaßnahmen werden das Tragen von Atemmasken oder die Einnahme von Iodtabletten berücksichtigt. Alle diese Umstände werden in einem Schutzfaktor erfasst. Dazu kommt, dass auch das Alter einer Person einen Einfluss auf die Aufnahme von Iod hat. Diesen Einfluss betrachten wir vor allem in Form einer Atemrate. Der Schutzfaktor und die Atemrate beschreiben, zu welchem Anteil das in der Luft vorhandene Iod auf eine Person zu einem Zeitpunkt eingewirkt hat. Zu jedem Abschnitt jedes Weges gibt es einen separaten Schutzfaktor. Die Atemrate betrachten wir für eine Person als konstant.

Als nächstes muss modelliert werden, dass das Iod einem radioaktiven Zerfall unterliegt. Den größten Anteil des Iods macht das Iod-131-Isotop aus, das eine Halbwertszeit von ungefähr acht Tagen hat. Es gibt also einen Unterschied zwischen der Menge an Iod, die eine Person zu einem Zeitpunkt aufgenommen hat, und der Menge an Iod, die bei einer Messung am Ende des Weges in der Notfallstation noch gemessen werden kann. Dieser Unterschied lässt sich durch einen Retentionsfaktor beschreiben. Jeder Punkt eines Weges hat seinen eigenen Retentionsfaktor.

Wir benutzen im Rahmen dieses Forschungsvorhabens explizit keine Annahmen über die physische Verbreitung des Iods, indem wir beispielsweise dessen Ausbreitung mittels Wetterdaten simulieren. Weiterhin führen wir keine räumliche oder zeitliche Interpolation durch, die über eine natürliche Interpolationswirkung einzelner von uns genutzter Methoden hinausgeht. Das Vorhaben konzentriert sich stattdessen auf die reine Auswertung der Bewegungsdaten.

1.2 Mathematische Formulierung

Das oben beschriebene Problem kann mathematisch formuliert werden. Wir nutzen dazu die in Tabelle 1.1 dargestellten Größen. Die Ausgabe ist eine Rekonstruktion der Verteilung des Iods. Es gibt zwei räumliche und eine zeitliche Dimension. Entlang dieser drei Dimensionen ist die Verteilung des Iods gesucht. Für einen Raumzeitpunkt $p \in \mathbb{R}^3$ notieren wir die Konzentration des Iods bei p mit x_p. Als Eingabe dient ein Datensatz aus Bewegungsdaten von $n \in \mathbb{N}$ Personen. Zu jeder Person gehört ein Weg $w \in \{1, \ldots, n\}$. Zu jedem Weg w gehört ein Messwert des Iods in der

Schilddrüse m_w am Ende des Weges. Entlang jedes Weges w gibt es für jeden Raumzeitpunkt $p \in \mathbb{R}^3$ einen Schutzfaktor $S_{w,p}$ und einen Retentionsfaktor $R_{w,p}$. Dazu kommen eine Atemrate B, die abhängig vom Alter der jeweiligen Person ist, und eine zeitliche Schrittgröße T, die als konstant angenommen werden.

Für jeden Weg w gilt die Gleichung:

$$m_w = \sum_{p \in R^3} B\, T S_{w,p} R_{w,p} x_p.$$

Die Faktoren B, T, $S_{w,p}$ und $R_{w,p}$ sind als Eingaben bekannt. Deren Produkt kann zu einem einzelnen Faktor $A_{w,p}$ zusammengefasst werden. Daraus ergibt sich die einfachere Formulierung:

$$m_w = \sum_{p \in \mathbb{R}^3} A_{w,p}\, x_p$$

oder über allen Wegen in Matrixschreibweise, wobei die Matrix A eine Zeile pro Person und eine Spalte pro Raumzeitpunkt beinhaltet:

$$m = Ax.$$

Daraus ergibt sich folgende Formulierung unseres Problems:

> Welches x erklärt für die Faktoren A die Messungen m?

Wie oben bereits beschrieben, sind die Eingabe (Wegfaktoren) und Ausgabe (Messwerte) bekannt. Unbekannt sind die Systemparameter in Form der Verteilung des Iods. Gesucht ist also nicht die Ausgabe eines Systems in der Form von Messungen, sondern die Beschaffenheit des Systems selbst. Deshalb handelt es sich bei unserem Problem um ein *inverses Problem*.

Inverse Probleme sind oft nicht leicht zu lösen. Die Bestimmung der Systemparameter kann beispielsweise schwierig sein, weil das System zu komplex ist, oder die Menge an Messungen für ein einfaches „Zurückrechnen" nicht ausreicht. In solchen Fällen handelt es sich um ein *schlecht gestelltes Problem*. Ein schlecht gestelltes Problem kann keine Lösung, eine Lösung oder viele Lösungen haben. Falls ein schlecht gestelltes Problem keine Lösung hat, kann versucht werden es umzuformulieren, sodass alle Ausprägungen eine Lösung sind. Dann entscheidet man sich für die Lösung mit der geringsten Abweichung zur unmöglichen perfekten Lösung. Hat ein Problem von vornherein schon viele Lösungen, können zusätzliche Anforderungen (Heuristiken) formuliert werden, mit denen die Lösungen weiter unterschieden werden können. Gewählt wird dann die Lösung, die unter Betrachtung der Heuristiken am besten ist.

Tabelle 1.1: Übersicht über die Größen des Problems

Name	Beschreibung	Einheit	Details
m	Messwert	Bq	Aktivität in der Schilddrüse
x	Konzentration des Iods in der Luft	Bq/m^3	orts- und zeitabhängige Unbekannte
B	Atemrate	m^3/h	kann als konstant angenommen werden
T	Zeitschritt	h	

R	Retention (verbleibender Teil der Aktivität)		von der Zeit zwischen der Exposition und der Messung abhängig
S	Schutzfaktor		ist für jedes *t* bekannt

1.3 Beispiel

Wir betrachten die in Abbildung 1.1 dargestellte Landschaft aus 3 x 3 Feldern. Die Landschaft ist jeweils links und rechts einmal als ein Raster dargestellt. In dem linken Raster ist zu erkennen, dass zu jedem Feld eine Variable $x_{i,j}$ gehört. Jede dieser Variablen beschreibt die Konzentration des Iods in dem dazugehörigen Feld. Rechts sind drei Wege eingezeichnet, die durch die Landschaft führen. Für jeden Weg ist auf jedem Feld ein Faktor gegeben. Am Ende jedes Weges gibt es eine Messung. Abbildung 1.2 zeigt zum Vergleich eine simulierte Iod-Verteilung über einer Landschaft.

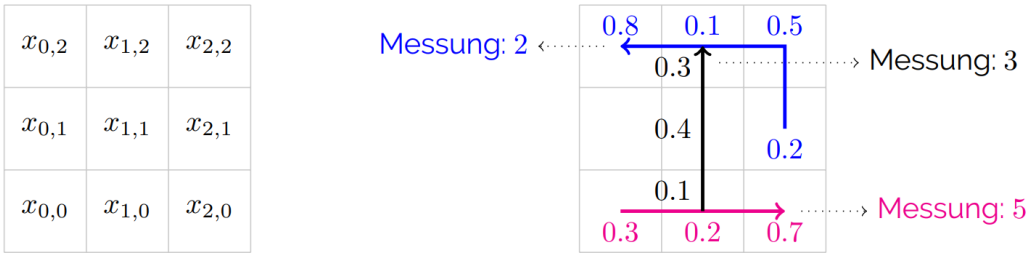

Abbildung 1.1: Schematische Darstellung einer Landschaft als Raster

Aus jedem Weg können wir nun eine Gleichung bilden. Wir summieren alle Variablen entlang eines Weges gewichtet mit den dazugehörigen Faktoren. Die Summe soll der Messung gleichen. Entsprechend der Messungen auf der rechten Seite der Gleichungen, ergeben sich für den magentafarbenen Weg die erste Gleichung, für den blauen Weg die zweite Gleichung und für den schwarzen Weg die dritte Gleichung:

$$\begin{aligned} 0.3\,x_{0,0} + 0.2\,x_{1,0} + 0.7\,x_{2,0} &= 5 \\ 0.2\,x_{2,1} + 0.8\,x_{0,2} + 0.1\,x_{1,2} + 0.5\,x_{2,2} &= 2 \\ 0.1\,x_{1,0} + 0.4\,x_{1,1} + 0.3\,x_{1,2} &= 3 \end{aligned}$$

Eine Lösung des Problems ist eine Belegung der Variablen $x_{0,0}, x_{0,1}, \ldots, x_{2,2}$, sodass die Gleichungen erfüllt sind. Das Beispiel hat unendlich viele Lösungen. Man kann sehen, dass die Variable $x_{0,1}$ in keinem Weg enthalten ist. Entsprechend gibt es auch keine Beschränkungen hinsichtlich ihrer Belegung. Es kann daher sinnvoll sein, sich nur auf die Variablen zu konzentrieren, die in mindestens einem Weg liegen. Darüber hinaus müssen für die eindeutige Lösung eines Gleichungssystems so viele unabhängige Gleichungen wie Variablen vorliegen. Das ist in diesem Beispiel, selbst nach Ausschluss von $x_{0,1}$, nicht gegeben. Man kann daran sehr gut die Komplexität unseres Problems erkennen. Für eine Auflösung wie sie in Abbildung 1.2 gezeigt ist, sind Daten von sehr vielen Wegen notwendig, um eine eindeutige Lösung erhalten zu können. Sollten sich unsere Gleichungen widersprechen, kann es auch sein, dass keine Lösung existiert.

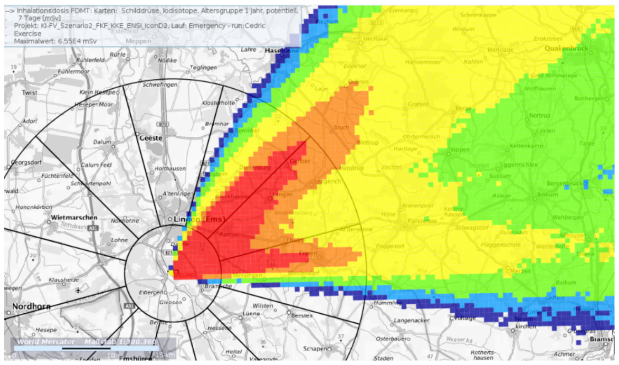

Abbildung 1.2: Eine Simulation einer Verteilung der Inhalationsdosis für Iodisotope über sieben Tage für die Altersgruppe 1 Jahr. Hier beispielhaft als Visualisierung für eine Ausbreitung von Iod genutzt.

1.4 Rahmenbedingungen

Ein Anwendungsfall von maschinellem Lernen läuft immer unter bestimmten Rahmenbedingungen. In unserem Fall sind besonders die Anzahl an Personen, von denen Bewegungsdaten erhoben werden, die Qualität dieser Bewegungsdaten und die Hardwareanforderungen wichtig.

Anzahl der Wege: Die Menge an Wegen ist nicht konstant, sondern steigt während eines Notfalles mit der Zeit an. Dies ist darin begründet, dass Personen erst in den Notfallstationen aufgenommen werden müssen. In den ersten Tagen gehen wir davon aus, dass Daten von wenigen tausend Wegen zur Verfügung stehen. In dieser Phase sollen möglichst trotzdem schon innerhalb von wenigen Stunden erste, möglicherweise gröbere Ergebnisse gewonnen werden. Im Verlauf der folgenden Wochen kann die Zahl der Wege, von denen Daten vorliegen, bis auf 200000 ansteigen. Eine mögliche Methode zur Lösung soll sowohl für eine geringe als auch eine hohe Anzahl an Wegen untersucht werden. In der aktuellen Version der betrachteten Software zur Erfassung der Bewegungsprofile von Personen enthält ein Bewegungsprofil schätzungsweise bis zu 1000 Wegpunkte. Diese Anzahl ist jedoch ein Erfahrungswert und keine technische Grenze.

Qualität der Bewegungsprofile: Eine hohe Anzahl an Wegen alleine reicht nicht aus, um verwertbare Rückschlüsse auf eine Iod-Verteilung zu ziehen. Es ist wichtig, dass die Wege der Menschen sind überschneiden, dabei aber nicht identisch sind. Ein Raumzeitpunkt ist dann gut rekonstruierbar, wenn viele unterschiedliche Wege durch ihn führen. Darüber hinaus werden die Bewegungsprofile von den betroffenen Personen per Befragung ermittelt. Dabei kann es zu Diskrepanzen zwischen den erhobenen Bewegungsprofilen und den tatsächlichen Bewegungsprofilen, die die Messung der Schilddrüsenaktivität vor Ort bedingen, kommen. Die Schilddrüsenmessungen in den Notfallstationen können ebenfalls Messungenauigkeiten unterliegen.

Verfügbare Computer: Die Methoden sollen möglichst auf einem Hochleistungs-PC gemäß der bereitgestellten Beschreibung (Bitkom, 2019) laufen können. Ein solcher PC ist mit 16 GB Arbeitsspeicher ausgestattet. Eine Grafikkarte ist bei PCs dieser Kategorie möglich, aber nicht vorgeschrieben.

2. Literaturrecherche zum Stand der Wissenschaft (AP 1)

In diesem Kapitel beschreiben wir unsere Literaturrecherche, die wir zu Beginn des Projektes durchgeführt haben. Der Bericht zu dieser Recherche, der insgesamt die Kapitel 1 und 2 umfasst, wurde separat angefertigt und für diesen Abschlussbericht in leicht modifizierter Variante übernommen. Wir haben ihn vor allem um die Methode Gaussian Splatting ergänzt, die erst während des weiteren Verlaufs des Projektes publiziert wurde.

2.1 Methodik

Ziel unseres Vorhabens ist, herauszufinden, ob und wie das in Kapitel 1 vorgestellte Problem gelöst werden kann. Um diese Frage zu beantworten, haben wir nach geeigneten Methoden gesucht, deren Übertragbarkeit geprüft und sie anhand ausgewählter Kriterien bewertet. Die folgenden Abschnitte beschreiben unser Vorgehen in den einzelnen Schritten.

2.1.1 Suche

Im ersten Schritt haben wir in der Literatur nach Ansätzen gesucht, die auf unser Problem übertragbar sein könnten. Dazu haben wir die Suchmaschinen Google Scholar und Papers With Code genutzt.

In den beiden Suchmaschinen haben wir eine Schlagwortsuche durchgeführt. Die Schlagworte sind sowohl deutsch als auch englisch. Wir haben die folgenden Schlagworte verwendet:

- **Optimierung:** Optimization, Approximation
- **Inverses Problem:** inverse problem, reconstruction, sparse approximation, ill-posed problems
- **Modellierung:** paths, spatial-temporal modeling, linear constraints
- **Bildgebende Ansätze:** CT/PET/SPECT-reconstruction, Trajektorienanalyse, computer tomography, reverse rendering, Neural Radiance Fields, image reconstruction
- **Optimierungsprobleme:** linear programming, stochastic programming, interior-point method
- **Neuronale Netzwerke:** recurrent neural networks, Long Short-Term Memory, Generative Adversarial Networks

Die Ergebnisse unserer Schlagwortsuche haben wir dann in einer zweiten Iteration mit Verweisen aus Zitierungen gefundener Literatur, Metastudien und Enzyklopädien ergänzt. Beispielsweise bietet das Portal Papers With Code Möglichkeiten, um Publikationen anhand von Aufgabenstellungen, Fachgebieten und verwendeten Datensätzen von bereits gefundener Literatur zu suchen.

2.1.2 Übertragung

Nachdem wir verschiedene Methoden in der Literatur identifizieren konnten, haben wir versucht, diese auf unser Problem zu übertragen. Für die Prüfung der Übertragung teilen wir das Problem in drei Teile:

- die Verteilung des Iods in der Umwelt,
- die Aufnahme des Iods durch eine Person auf ihrem Weg und
- der Messwert der Aktivität des Iods in einer Person am Ende ihres Weges.

Wir geben bei jeder Methode an, wie sich die einzelnen Teile des Problems in ihr abbilden lassen. Kann ein Teil nicht übertragen werden, ist die betrachtete Methode nicht anwendbar. Wir schließen Ansätze von einer tieferen Bewertung aus, wenn aus unserer Sicht klar ist, dass sie keine Anwendung finden werden.

2.1.3 Bewertung

Die gefundenen und nicht ausgeschlossenen Ansätze haben wir nach festen Kriterien bewertet. Für jedes Kriterium definieren wir eine oder mehrere Fragen, die wir zur Beurteilung der Methode für wichtig betrachten. Die Kriterien listen wir in Abschnitt 2.1.3.1 auf. Es ist wichtig zu beachten, dass wir diese Bewertungen ohne ausführliche Testungen der Methoden und nur auf Basis der Literatur vorgenommen haben und deshalb teilweise Abschätzungen vornehmen mussten.

Jede Bewertung findet auf einer Skala von 1 bis 5 Punkten statt und wird im Format „*Punkte*/5" notiert.

Aus den einzelnen Bewertungen der Kriterien haben wir dann für jeden Ansatz eine gewichtete Summe gebildet. Diese Summe bildet die Gesamtbewertung der Methode. Die Gewichte wurden in Absprache mit der Auftraggeberin gewählt und werden in Abschnitt 2.1.3.2 diskutiert.

2.1.3.1 Kriterien

Ähnlichkeit

- Wie ähnlich sind sich unser Problem und die mit der Methode gelösten Probleme?
- Können wir die Methode mit angemessenem Aufwand auf unser Problem übertragen?

Bedarf an Zeit

- Wie viel Rechenleistung und damit auch Zeit benötigt die Methode?
- Kann die Methode mit der gegebenen voraussichtlichen Problemgröße in der gewünschten Zeit auf dem gewünschten Endgerät durchgeführt werden?

Bedarf an Speicher

- Wie viel Speicher benötigt die Methode?
- Kann die Methode mit der gegebenen voraussichtlichen Problemgröße mit dem gewünschten Speicherbedarf auf dem gewünschten Endgerät durchgeführt werden?

Bedarf an Daten

- Wie viele Trainingsdaten sind notwendig?
- Kann die Methode aus der gegebenen Menge an Daten eine sinnvolle Lösung extrahieren?
- Kann die Methode auch mit wenigen Daten verwertbare Ergebnisse produzieren?

***A priori*-Fähigkeit**

- Ist das Einbringen von Vorwissen möglich?
- Ist das Einbringen von Vorwissen nötig?
- Die folgenden Formen von Vorwissen über das Problem werden von uns betrachtet:
    - Vorwissen über den Zustand einzelner Datenpunkte (z.B. durch Messungen vor Ort)

  - Vorwissen über eine ungefähre Ausprägung der Lösung (z.B. durch Simulationen)
  - Annahmen über die Form der Lösung, insbesondere bezüglich Sparsamkeit und Glätte

Implementierungsaufwand

- Wie viel Implementierung ist notwendig? Wir bewerten insbesondere den Aufwand für zwei Teile: Kern der Methode und Aufbereitung der Daten zur Anwendung der Methode auf diese.
- Gibt es Bibliotheken oder Frameworks, die eine Implementierung vereinfachen?

Unsicherheit

- Kann die Methode die Rekonstruktion mit einer Unsicherheit, zum Beispiel einer Varianz, bewerten?

Robustheit

- Wie gut kann die Methode mit fehlerhaften Daten umgehen?
- Wie stark reagiert die Methode auf Schwankungen in Eingaben?

Reife

- Wie gut ist die Methode erforscht?

Bekanntheit

- In welchen Bereichen findet die Methode Anwendung?

Anwendungsbreite

- Wie vielfältig sind die Anwendungsbereiche der Methode?

2.1.3.2 Gewichte

Wir unterscheiden die vorgestellten Bewertungskriterien bezüglich ihrer Wichtigkeit. So ist es für eine Methode selbstverständlich entscheidender, ob sie gut übertragbar ist und schnell berechnet werden kann, als dass sie weit verbreitet ist. Deshalb nutzen wir für die Ermittlung der Gesamtbewertung die in Tabelle 2.1 aufgeführten Gewichte bei der Summierung der einzelnen Bewertungen. Wir bewerten die zentralsten Kriterien *Ähnlichkeit* und jeweils den *Bedarf an Zeit, Speicher* und *Daten* für die Ausführung einer Methode mit vier Punkten. Die Fähigkeit *Unsicherheit* abzubilden, und die *Robustheit* der Methoden werden mit drei Punkten gewichtet. Mit der *a priori-Fähigkeit* und der Einfachheit der *Implementierung* bewerten wir wünschenswerte Eigenschaften bei der Modellierung mit zwei Punkten. Die Kriterien *Reife*, *Bekanntheit* und *Anwendungsbreite* bewerten die Methoden eher indirekt und werden von uns mit einem Punkt gewichtet.

Tabelle 2.1: Die Gewichtungen der einzelnen Kriterien für die Gesamtbewertung.

Kriterium	**Gewicht**
Ähnlichkeit	4
Bedarf an Zeit	4
Bedarf an Speicher	4
Bedarf an Daten	4
Unsicherheit	3
Robustheit	3
A priori-Fähigkeit	2
Implementierung	2
Reife	1
Bekanntheit	1
Anwendungsbreite	1

2.2 Ergebnisse

2.2.1 Verlauf unserer Suche

Wir haben die von uns gefundene Literatur in 10 Methoden gruppiert. Alle identifizierten Methoden befinden sich zusammen mit den wichtigsten Literaturquellen in Tabelle 2.2.

Wir haben unsere Recherche mit zwei Ansätzen begonnen: Neural Radiance Fields und gefilterte Rückprojektion. Eine weitere Suche entlang der gefundenen Literatur hat dann zu einer Abgrenzung des speicherbasierten inversen Renderings von Neural Radiance Fields geführt. Obwohl speicherbasiertes inverses Rendering seinen Ursprung in Neural Radiance Fields hat, haben wir es als eigene Methode extrahiert, weil wir es für unser Problem besonders interessant finden.

Mithilfe von Breitensuche in Enzyklopädien konnten wir lineare Optimierung, evolutionäre Algorithmen und die Definition unseres Problems als inverses Problem ausfindig machen. Mittels Schlagwortsuche konnten wir dann die stochastische robuste Optimierung als eine Erweiterung der linearen Optimierung finden. Wir sehen die Modellierungsmöglichkeiten der stochastischen robusten Optimierung als unterschiedlich genug an, damit sich eine detaillierte Betrachtung lohnt. Insbesondere ist es mittels stochastischer Optimierung möglich, mit Unsicherheiten in der Problembeschreibung umzugehen.

Ebenfalls mittels Schlagwortsuche konnten wir die bayesschen inversen Probleme finden. Sie betrachten inverse Probleme unter Einbringung von Unsicherheit, vor allem mithilfe von Gauß-Verteilungen.

Schlussendlich haben wir mit Graph Neural Networks, rekurrenten neuronalen Netzen (RNN) und Generative Adversarial Networks (GAN) drei mögliche Methodiken aus dem Bereich der neuronalen Netzwerke betrachtet.

Im August 2023, nach der durchgeführten Literaturrecherche, wurde eine weitere Methode erstveröffentlicht: Gaussian Splatting (Kerbl et al., 2023). Bei dieser Methode handelt es sich ebenfalls um eine Weiterentwicklung im gleichen Bereich wie Neural Radiance Fields. Diese

bringt einen aus unserer Sicht so vielversprechenden Ansatz mit sich, dass wir sie nachträglich noch als weitere Methode aufgenommen haben.

Tabelle 2.2: Übersicht über alle Methoden und Quellen

Methodik	**Literatur**
Neural Radiance Fields	NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis (Mildenhall, et al., 2020) „Space-Time Neural Irradiance Fields for Free-Viewpoint Video“ (Xian, et al., 2021) außerdem: (Barron, et al., 2022), (Barron, et al., 2021), (Barron et al., 2023), (Dellaert & Yen-Chen, 2021), (Dey, et al., 2022), (Li, et al., 2021), (Oechsle, et al., 2021), (Park, et al., 2021), (Park, et al., 2021), (Ramirez, et al., 2022), (Tancik et al., 2021)
Speicherbasiertes inverses Rendering	„Neural Sparse Voxel Fields“ (Liu, et al., 2020) „Plenoxels: Radiance Fields Without Neural Networks“ (Fridovich-Keil, et al., 2022) *NeAT: Neural Adaptive Tomography* (Rückert, et al., 2022) „Instant neural graphics primitives with a multiresolution hash encoding“ (Müller, et al., 2022) außerdem: (Peng, et al., 2020), (Sun, et al., 2022)
Neu: Gaussian Splatting	“3D Gaussian Splatting for Real-Time Radiance Field Rendering” (Kerbl, et al., 2023)
Gefilterte Rückprojektion	„Deep learning computed tomography“ (Würfl, et al., 2016)
Lineare Optimierung	*Linear programming* (Vanderbei et al., 2020) *Convex optimization* (Boyd, et al., 2004) außerdem: (Cohen, et al., 2018), (Dantzig, 1982), (Luby & Nisan, 1993), (Tauman Kalai, et al., 2016), (Vaidya, 1989)
Stochastische robuste Optimierung	*Convex optimization* (Boyd, et al., 2004) „Stochastic Programming“ (Shapiro, 2019) außerdem: (Biel & Johansson, 2022), (Prekopa, 1973), (Shapiro, 1993)
Bayessche inverse Probleme	*The Bayesian Approach To Inverse Problems* (Dashti & Stuart, 2013) außerdem: (Bui-Thanh, 2012), (Görtler, et al., 2019), (Jidling, et al., 2017), (Kekkonen, 2019), (Raissi & Karniadakis, 2016)
Evolutionäre Algorithmen	„Evolutionary algorithms“ (Bartz-Beielstein et al., 2014)
Graph Neural Networks	*Inductive Graph Neural Networks for Spatiotemporal Kriging* (Wu, et al., 2020) außerdem: (Agrawal & de Alfaro, 2019), (Eliasof, et al., 2022), (Nicolicioiu, et al., 2019), (Su, et al., 2016)

Rekurrente Neuronale Netzwerke	„Long short-term memory“ (Hochreiter & Schmidhuber, 1997) außerdem: (Cer, et al., 2018), (Conneau, et al., 2018), (Fernández-González & Gómez-Rodríguez, 2020), (Iyyer, et al., 2015), (Socher, et al., 2013)
Generative Adversarial Networks	Generative Adversarial Networks (Goodfellow, et al., 2014) außerdem: (Mirza & Osindero, 2014), (Zhu, et al., 2017)

2.2.2 Vorauswahl

Wir haben alle von uns aufgelisteten Ansätze nach den identifizierten Kriterien bewertet. In Tabelle 2.3 listen wir die von uns gefundenen und nicht ausgeschlossenen Methoden mit ihren Bewertungen auf. Die ausgeschlossenen Methoden sind am Ende der Tabelle ebenfalls ausgegraut enthalten. Unsere Bewertungen sehen die stochastische robuste Optimierung und die bayesschen inversen Probleme gemeinsam auf dem ersten Platz. Die lineare Optimierung belegt den dritten Platz. Auf Platz vier befindet sich das speicherbasierte inverse Rendering. Den letzten Platz belegen die klassischen Neural Radiance Fields. Wie in Kapitel 2.2.1 erläutert, wurde nun zusätzlich die Methode Gaussian Splatting aufgenommen. Diese liegt in ihrer theoretischen Bewertung unterhalb von stochastischer robuster Optimierung, bayesschen inversen Problemen und linearer Optimierung, aber überhalb von speicherbasiertem inversen Rendering und Neural Radiance Fields.

In diesem Kapitel folgen Steckbriefe zu jeder einzelnen Methode, inklusive Begründungen zu den Bewertungskriterien. Die beschriebenen Ergebnisse sind im Stand der Literaturrecherche belassen worden, sodass eine Nachvollziehbarkeit der Auswahlentscheidung gegeben ist. Bestimmte Bewertungen oder getroffene Implementierungsentscheidungen können sich im weiteren Projektablauf unterscheiden. Die detaillierten Informationen, wie welche Methode tatsächlich implementiert wurden, befinden sich in Kapitel 4.3.

Tabelle 2.3: Übersicht über die Bewertungen aller Methoden

	Summe	Ähnlichkeit	Bedarf an Zeit	Bedarf an Speicher	Bedarf an Daten	Unsicherheit	Robustheit	*A priori*-Fähigkeit	Implementierungsaufwand	Reife	Bekanntheit	Anwendungsgebiete
Gewichte		*4*	*4*	*4*	*4*	*3*	*3*	*2*	*2*	*1*	*1*	*1*
Stochastische robuste Optimierung	**111**	5	4	2	3	4	5	3	4	5	4	5
Bayesian Inverse Problems	**111**	5	2	2	5	5	5	5	1	5	3	5
Lineare Optimierung	**95**	5	5	3	2	1	1	2	5	5	5	5
Neue Methode: Gaussian Splatting	**89**	3	4	2	4	2	4	2	3	2	4	3
Speicherbasiertes inverses Rendering	**86**	3	3	3	4	2	3	2	3	2	4	3
Neural Radiance Fields	**72**	3	1	5	1	2	3	2	2	2	4	3
Gefilterte Rückprojektion	**0**											
Evolutionäre Algorithmen	**0**											
Graph Neural Networks	**0**											
Generative Adversarial Networks	**0**											
Rekurrente Neuronale Netzwerke	**0**											

2.2.2.1 Neural Radiance Fields

Überblick

Neural Radiance Fields (NeRFs) sind neuronale Netze. Sie werden benutzt, um 3D-Objekte aus Bildern zu lernen. Für jedes Bild sind die Kameraposition und die Blickrichtung bekannt.

Ein Neural Radiance Field berechnet für jede Position und jede Blickrichtung im Raum eine Dichte und eine Farbe. Die Abhängigkeit von der Blickrichtung ist bei Farben sinnvoll, weil so Lichteffekte wie Reflexionen gelernt werden können. Die Dichte ist unabhängig von der Blickrichtung.

Um aus der Verteilung von Farbe und Dichte im Raum Bilder zu berechnen, werden Strahlen wie von einer Kamera durch den Raum gezogen. Die Abbildung 2.1 veranschaulicht diesen Prozess in einer Draufsicht. Auf der linken Seite befindet sich ein Bild, das aus vier Pixeln besteht. Ausgehend von jedem Pixel des Bildes wird ein Strahl durch den Raum gezogen. Im Raum befindet sich das 3D-Objekt, das in der Abbildung rechts durch eine Wolke mit Farbverlauf dargestellt ist. Entlang der Strahlen werden an ausgewählten Punkten mithilfe des Neural Radiance Fields die Farben und Dichten bestimmt. Die Farben werden dann nach den Dichten gewichtet aufsummiert. Das Ergebnis ist die Farbe des Pixels, zu dem der jeweilige Strahl gehört. In der Abbildung 2.1 ist dieser Berechnungsprozess für das unterste Pixel eingezeichnet.

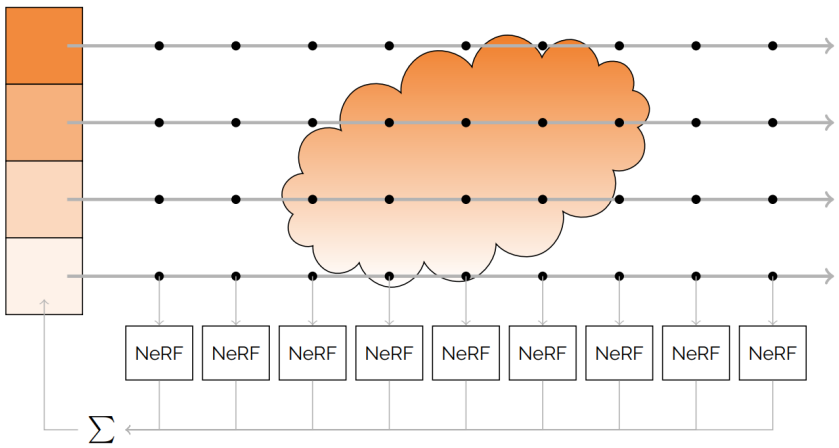

Abbildung 2.1: Berechnung der Pixelfarben mithilfe eines Neural Radiance Fields

Beispiel

Ein Neural Radiance Field soll genutzt werden, um ein Schlagzeug im dreidimensionalen Raum zu rekonstruieren. Für das Training stehen 100 Bilder von einem Schlagzeug bereit. Jedes Bild wurde aus einer bekannten Kameraposition aufgenommen. Auf diesen Bildern wird nun das Neural Radiance Field so trainiert, dass es die Bilder reproduzieren kann. In diesem Prozess lernt es, das Schlagzeug als 3D-Objekt zu repräsentieren. Es abstrahiert das Wissen aus den Bildern. Nach dem Training kann das Neural Radiance Field deshalb nicht nur zur Rekonstruktion der bereits bekannten Bilder benutzt werden, sondern auch, um Bilder aus neuen Blickwinkeln zu erzeugen. Das kann natürlich nur dann besonders gut gelingen, wenn die Trainingsdaten möglichst gute Informationen über das ganze Objekt liefern konnten. Eine Visualisierung dieses Beispiels findet sich in der Publikation von Mildenhall et al. (2020).

Übertragung

Problemstellung	**Methode**
Verteilung des Iods	Gelernt durch das neuronale Netz
Kontamination einer Person entlang ihres Weges	Gewichtete Summe über den Farben pro Strahl
Messwert der Kontamination einer Person	Farbwert eines Pixels

Die Iod-Verteilung wird durch das neuronale Netzwerk modelliert, das darauf trainiert wird, einen Punkt in der Raumzeit auf eine Iod-Konzentration abzubilden. Die Wege der Personen entsprechen den Kamerastrahlen. Entlang der Wege werden Punkte ausgewählt, an denen die Iod-Konzentration mithilfe des neuronalen Netzes ermittelt wird. Es ist wichtig zu beachten, dass es bei Neural Radiance Fields keine mathematische Notwendigkeit dafür gibt, dass die Kamerastrahlen geradlinig sind. Sie werden lediglich über ausgewählte Punkte repräsentiert. Deswegen ist eine Übertragung auf die Wege der Personen in unserem Problem möglich. Die einzelnen Konzentrationen an den ausgewählten Punkten werden dann nach den gegebenen Wegfaktoren gewichtet aufsummiert. Die gewichtete Summe entspricht der gemessenen Kontamination einer Person am Ende eines Weges.

Bewertung

Ähnlichkeit	3/5	Gewichtete Summe, Gewichte werden mit gelernt
Bedarf an Zeit	1/5	Training dauert 1-2 Tage auf einer GPU
Bedarf an Speicher	5/5	Hoher Bedarf in unserem Test, Optimierungen möglich
Bedarf an Daten	1/5	Hoher Datenbedarf beschrieben
Unsicherheit	2/5	nicht enthalten, aber denkbar
Robustheit	3/5	Datenartefakte können, aber müssen nicht gelernt werden
A priori-Fähigkeit	2/5	Vorwissen nicht vorgesehen, Fehlerfunktion modifizierbar
Implementierungsaufwand	2/5	modifizierte Neuimplementierung mit PyTorch oder TensorFlow
Reife	2/5	sehr aktuelle Forschung
Bekanntheit	4/5	hohe Aufmerksamkeit in den letzten Jahren
Anwendungsbreite	3/5	Anwendungen für Bilder und bildgebende Verfahren

Begründungen

Ähnlichkeit (3/5) Neural Radiance Fields lassen sich gut auf unser Problem übertragen. Die Konzentration des Iods in der Umwelt entspricht der Verteilung der Dichte des 3D-Objektes. Eine Analogie zur Farbe gibt es in unserem Problem nicht. Deshalb besteht auch keine Abhängigkeit von Blickwinkeln. Die Wege der Menschen sind übertragbar auf die Kamerastrahlen. Im Vergleich zur gefilterten Rückprojektion erfolgt bei Neural Radiance Fields keine Projektion des Objektes. Stattdessen werden die Strahlen durch einzelne Punkte repräsentiert. Dass die Kamerastrahlen gerade sind, ist inhaltlich sinnvoll, aber mathematisch nicht notwendig. Entsprechend lässt sich die gewichtete Summe der Iod-Kontamination über den Faktoren auf die gewichtete Summe der

Farbe über der Dichte übertragen. Die Gewichte der Summe, die der Dichte entsprechen, werden in unserem Problem jedoch nicht gelernt, sondern sind durch die Faktoren entlang der Wege gegeben.

Bedarf an Zeit (1/5) Klassische Neural Radiance Fields trainieren auf sehr leistungsfähigen Grafikkarten ein bis zwei Tage (Mildenhall, et al., 2020). Entsprechend unserer Übertragung des Problems und einer Anwendung von Neural Radiance Fields für Computertomographie (Tancik et al., 2021) halten wir eine Lösung unseres Problems jedoch auch mit einer kleineren Architektur für möglich. Zusätzlich lassen sich mit Meta-Learning kontextspezifische Initialisierungen finden, die ein wesentlich schnelleres Training ermöglichen (Tancik et al., 2021). So müsste ein Neural Radiance Field nicht komplett von Neuem trainiert, sondern nur mittels eines kürzeren Trainings auf die aktuelle Datenlage angepasst werden. Ein solches Meta-Learning müsste mit möglichst vielen und verschiedenen Szenarien trainiert werden. Zur Gewinnung dieser Szenarien könnten Simulationen genutzt werden.

Bedarf an Speicher (5/5) Der Speicherbedarf hängt im Wesentlichen nur von der Anzahl der Gewichte und damit von der Architektur des neuronalen Netzes ab. Die Größe eines Neural Radiance Fields wird für das in der originalen Publikation verwendete neuronale Netzwerk mit ungefähr 5 MB angegeben (Mildenhall, et al., 2020). Im Vergleich zu den verfügbaren 16 GB RAM oder einer expliziten Speicherung der Daten des abgebildeten Raumes, die ebenfalls mehrere GB betragen kann, ist das ein sehr kleiner Speicherbedarf. Es ist denkbar, dass ein noch kleineres Netz ebenfalls ausreicht (Tancik et al., 2021).

Bedarf an Daten (1/5) Neural Radiance Fields werden klassischerweise mit Bildern trainiert, auf denen das komplette Objekt zu sehen ist. Das bedeutet, dass es an jedem Punkt des zu lernenden Objektes einen Lichtstrahl aus jedem Blickwinkel gibt. Bei genügend vielen und gut verteilten Blickwinkeln kann damit sichergestellt werden, dass sich an jedem relevanten Punkt mehrere Strahlen kreuzen. Neural Radiance Fields wurden zum Beispiel mit 479 Bildern mit einer Auflösung von 512 x 512 Pixeln trainiert (Mildenhall, et al., 2020). Das entspricht in etwa 125 000 000 sich gleichmäßig überschneidenden Kamerastrahlen. Diese Eigenschaft ist bei unserem Problem nicht gegeben. Wir können nicht davon ausgehen, dass sich an jedem Punkt genügend viele Wege kreuzen. Daher vermuten wir, dass die Menge an Daten in unserem Problem nicht ausreicht, um ein Neural Radiance Field gut zu trainieren.

Unsicherheit (2/5) Neural Radiance Fields bilden keine Unsicherheiten ab. Eine Modellierung eines neuronalen Netzwerkes, das Unsicherheiten zu seinen Ergebnissen ausgibt, ist jedoch möglich. Dazu wäre bei der ohnehin nötigen Anpassung der Architektur von Neural Radiance Fields an unser Problem eine Ergänzung um eine Ausgabe von Unsicherheit nötig. Während des Trainings müsste es dem neuronalen Netz dann besser angerechnet werden, wenn es sich bei möglichst richtigen Vorhersagen sicher und bei eher falschen Vorhersagen unsicher ist.

Robustheit (3/5) Neural Radiance Fields sind so konstruiert, dass sie auch hochfrequente Informationen lernen können (Mildenhall, et al., 2020; Tancik, et al., 2020). Das ist nötig, um einen hohen Detailgrad bei den Rekonstruktionen zu erreichen. Es ist deshalb aber auch möglich, dass Ausreißer in den Daten mitgelernt werden. Andererseits müssen die Ausreißer nicht mitgelernt werden, weil Neural Radiance Fields ihre Trainingsdaten nicht perfekt rekonstruieren können müssen. Es ist denkbar, dass außergewöhnliche Trainingsdaten zugunsten eines generell besseren Modells während des Trainings ignoriert werden.

***A priori*-Fähigkeit (2/5)** Vorwissen ist nicht nötig, um das Modell zu trainieren. Es ist jedoch möglich, zusätzliche Kriterien wie Sparsity, Smoothing oder die Ähnlichkeit zu einer Simulation

hinzuzufügen. Bekannte Iod-Konzentrationen an einzelnen Punkten können im Modell nicht erzwungen werden. Sie können beim Training jedoch berücksichtigt werden.

Implementierungsaufwand (2/5) Eine für unser Problem formulierte Variante von Neural Radiance Fields muss neu implementiert werden. Es muss so konstruiert sein, dass es die Strahlungsverteilung pro Raumzeitpunkt abbilden kann und die für unser Problem gegebene Akkumulation verwendet. Für die Implementierung von neuronalen Netzen besteht eine gute Auswahl an Bibliotheken, wie zum Beispiel TensorFlow (Abadi et al., 2015) und PyTorch (Paszke et al., 2019).

Reife (2/5) Neural Radiance Fields sind ein junges Fachgebiet, deren Erforschung sehr aktiv voranschreitet. Obwohl sie erst seit wenigen Jahren existieren, liefern sie in ihrem Feld schon jetzt beeindruckende Erfolge. Rund um Neural Radiance Fields ist ein Fachgebiet entstanden, das aktuell viele neue Ideen und Verbesserungen hervorbringt.

Bekanntheit (4/5) Im Bereich des Machine Learnings sind Neural Radiance Fields sehr gut bekannt. Das Originalpaper (Mildenhall, et al., 2020) wurde auf der European Conference on Computer Vision 2020 als eines von zwei Papern mit *Best Paper Honorable Mention* ausgezeichnet. Die Seite *Papers With Code* registriert für 2020 37, für 2021 209, für 2022 537 und für 2023 (Stand vom 27. April 2023) bereits 336 Veröffentlichungen im Zusammenhang mit Neural Radiance Fields (siehe https://paperswithcode.com/dataset/nerf).

Anwendungsbreite (3/5) Neural Radiance Fields finden im Allgemeinen Anwendung in der Rekonstruktion von Objekten und in der Bildsynthese. Anwendungen für Computertomographie (Tancik et al., 2021) und Rekonstruktion von 3D-Szenen aus Kameraaufnahmen werden beschrieben (Park, et al., 2021). Als Anwendungsziele wurden von Gao, et al. (2022) urbane Szenen und Menschen identifiziert.

2.2.2.2 Speicherbasiertes Inverses Rendering

Überblick

Wir trennen von Neural Radiance Fields eine Gruppe an Ansätzen ab, die nicht auf einem neuronalen Netz, sondern auf einer Datenstruktur basieren. In der Literatur findet diese Abgrenzung nicht so klar statt. Das Unterscheidungskriterium besteht in der Modellierung der Verteilung von Informationen. Neural Radiance Fields haben Parameter, aus denen eine Information *berechnet* wird. Ansätze des speicherbasierten inversen Renderings *speichern* die Information dagegen direkt als Parameter in einer Datenstruktur ab. Zur Bestimmung einer Information sind deshalb nur wenige Parameter notwendig. Mögliche Datenstrukturen sind Gitter (Fridovich-Keil, et al., 2022), Bäume (Rückert, et al., 2022) oder Hashtabellen (Müller, et al., 2022). Einige Ansätze benutzen eine Datenstruktur, um Informationen zu speichern, aber verarbeiten daraus ausgelesene Daten mit einem neuronalen Netz weiter. Abbildung 2.2 zeigt den Berechnungsprozess. Auf der linken Seite befindet sich ein Bild aus vier Pixeln. Aus jedem Pixel wird ein Strahl durch den Raum und die zu rekonstruierende Wolke gesandt. In dem grünen Gitter sind lokale Informationen über die Wolke gespeichert. Entlang der Strahlen können diese Informationen ausgelesen und direkt zu einem Pixel summiert werden.

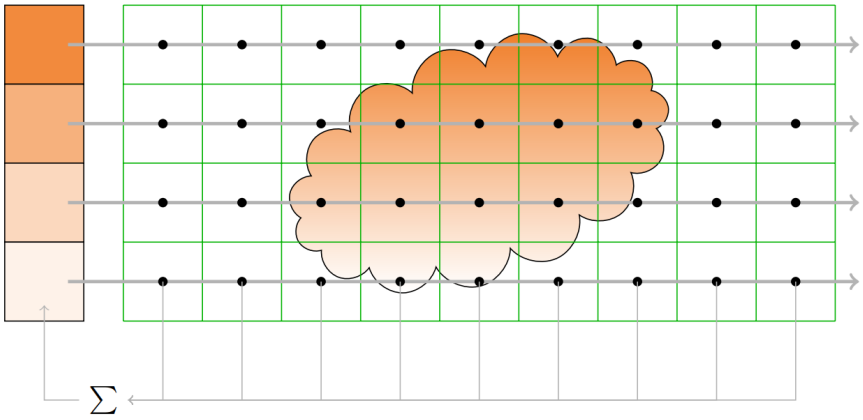

Abbildung 2.2: Berechnung der Pixelfarben mithilfe einer Datenstruktur

Beispiel

Gelernt werden soll die 3D-Repräsentation einer Zimmerpflanze. Von einer Kameraposition aus, wird ein Strahl durch ein Trainingsbild dieser Zimmerpflanze in eine gitterförmige Datenstruktur gezogen. An den Eckpunkten des Gitters sind Informationen zu Farbe und Dichte des auf dem Bild dargestellten Objektes gespeichert. Diese Informationen werden während eines Trainings aus Bildern gewonnen, die aus verschiedenen Winkeln von dem Objekt gemacht wurden. An dieser Stelle findet der gleiche Vorgang statt, der in Kapitel 2.2.2.1 zu Neural Radiance Fields beschrieben wurde. Besonders für speicherbasiertes inverses Rendering ist die räumliche Datenstruktur, in der die Informationen gespeichert sind. Zudem können Teile der Datenstruktur, die sich während des Trainings als irrelevant herausstellen, gelöscht werden können. Eine Visualisierung dieses Beispiels findet sich in der Publikation von Liu et al. (2020).

Übertragung

Problemstellung	**Methode**
Verteilung des Iods	Gespeichert in der Datenstruktur
Kontamination einer Person entlang ihres Weges	Gewichtete Summe über den Farben pro Strahl
Messwert der Kontamination einer Person	Farbwert eines Pixels

Es wird eine Datenstruktur konstruiert, die die Raumzeit abdeckt, in dem sie abschnittsweise nicht ein Paar aus Farbe und Dichte, sondern eine Iod-Konzentration speichert. Diese Datenstruktur modelliert dann die Iod-Verteilung. Nach dem gleichen Prinzip wie bei Neural Radiance Fields können nun die Kamerastrahlen des inversen Renderings auf die Wege der Personen übertragen werden. Sowohl die Kamerastrahlen als auch die Wege werden jeweils durch eine Menge von einzelnen Punkten repräsentiert. An jedem Punkt eines Weges kann dann durch Nachschlagen in der Datenstruktur eine Iod-Konzentration ermittelt werden. Die Kontamination einer Person entlang eines Weges wird durch die gewichtete Summe aller Iod-Konzentrationen nach den Wegfaktoren berechnet. Das Ergebnis davon entspricht dem Messergebnis der Person in der Notfallstation. Im Vergleich dazu wird beim speicherbasierten Rendering eine gewichtete Summe über Dichten und Farben gebildet, die den Wert eines Pixels bestimmt.

Bewertung

Ähnlichkeit	3/5	Gewichtete Summe, Gewichte werden mit gelernt
Bedarf an Zeit	3/5	deutlich schneller als Neural Radiance Fields
Bedarf an Speicher	3/5	Hoher Bedarf in unserem Test, Optimierungen möglich
Bedarf an Daten	4/5	Auflösung an Datenmenge anpassbar
Unsicherheit	2/5	wird nicht modelliert, könnte aber hinzugefügt werden
Robustheit	3/5	Artefakte durch Datenstrukturen, robust im Training
A priori-Fähigkeit	2/5	Vorwissen nicht vorgesehen, Fehlerfunktion modifizierbar
Implementierungsaufwand	3/5	modifizierte Neuimplementierung nötig
Reife	2/5	sehr aktuelle Forschung
Bekanntheit	4/5	hohe Aufmerksamkeit in den letzten Jahren
Anwendungsbreite	3/5	Anwendungen für Bilder und bildgebende Verfahren

Begründungen

Ähnlichkeit (3/5) Datenstrukturbasierte Ansätze sind unserem Problem genauso ähnlich wie klassische Neural Radiance Fields.

Bedarf an Zeit (3/5) Speicherbasiertes inverses Rendering ist schneller als Neural Radiance Fields. Bei einem Neural Radiance Field werden bei jeder Ausführung alle Gewichte verrechnet. Beim Speicherbasierten inverses Rendering wird dagegen nur ein Zugriff in eine Datenstruktur durchgeführt. Der Ansatz von Plenoxeln ist beispielsweise um ein hundertfaches schneller als der

Ansatz mit Neural Radiance Fields und braucht auf einer sehr guten Grafikkarte lediglich 11 Minuten zum Trainieren (Fridovich-Keil, et al., 2022).

Bedarf an Speicher (3/5) Datenstrukturbasierte Ansätze erkaufen ihre schnellere Geschwindigkeit mit der Notwendigkeit alle Daten explizit speichern zu müssen. Dadurch ergibt sich in Abhängigkeit von der Größe des zu rekonstruierenden Problems eine potenziell sehr große Menge an Daten. Verschiedene Methoden existieren, um den hohen Speicherbedarf einzugrenzen (Fridovich-Keil, et al., 2022; Liu, et al., 2020; Müller, et al., 2022; Rückert, et al., 2022). So kann zum Beispiel zunächst auf einer grob aufgelösten und daher kleineren Datenstruktur trainiert werden. Entsprechend der Ergebnisse können dann uninteressante Teile der Struktur verworfen werden und die übrigen Teile mit einer höheren Auflösung erneut trainiert werden.

Bedarf an Daten (4/5) Die Auflösung der verwendeten Datenstrukturen kann an die Informationsdichte angepasst werden. Liegen in einem Bereich wenig Bewegungsdaten vor, kann eine gröbere Auflösung gewählt werden. Liegen dagegen in einem Bereich viele Daten vor, kann eine detaillierte Auflösung gewählt werden. Teile einer Datenstruktur, die einen irrelevanten Bereich abdecken würden, können ausgelassen werden. Diesen flexiblen Umgang mit der Menge an Daten sehen wir als deutlichen Vorteil. Gerade in der Anfangsphase der Anwendung, wenn erst wenige Bewegungsdaten erhoben werden konnten, können trotzdem bereits erste grobe Rekonstruktionen ausgeführt werden.

Unsicherheit (2/5) Eine Modellierung von Unsicherheit findet beim speicherbasierten inversen Rendering nicht statt. Wir halten eine zusätzliche Modellierung von Unsicherheit jedoch für möglich, indem ein Wert für Unsicherheit zu jedem Datum in der Datenstruktur hinzugefügt wird. Dann müsste beim Training ein Aufwiegen der Qualität der Daten in der Datenstruktur mit der angegebenen Sicherheit stattfinden, sodass entweder ein möglichst richtiges Ergebnis, oder wenigstens eine hohe Unsicherheit extrahiert werden können.

Robustheit (3/5) Speicherbasiertes inverses Rendering ist wegen des direkten Trainings einzelner Datenpunkte vergleichsweise anfällig gegenüber Datenartefakten. Darüber hinaus können komplexere Datenstrukturen, wie zum Beispiel Bäume, zusätzliche Artefakte hinzufügen, die ausgeglichen werden müssen. In der Literatur ist zu sehen, dass zusätzliche Annahmen notwendig sind, um die Ansätze gut trainieren zu können (Rückert, et al., 2022). Grundsätzlich ist das Training jedoch in der Lage, Ausreißer in den Daten auszugleichen, wenn genügend gegensätzliche Evidenz in den Trainingsdaten vorliegt.

***A priori*-Fähigkeit (2/5)** Es ist kein Vorwissen nötig, um die Methode zu nutzen. Äquivalent zu Neural Radiance Fields ist es beim Training möglich, Sparsity, Smoothing oder die Ähnlichkeit zu einer Simulation hinzuzufügen. Vorwissen über einzelne Datenpunkte kann erzwungen werden, weil die einzelnen Datenpunkte explizit in der Datenstruktur vorliegen und deshalb einfach eingetragen und beim Training konstant gehalten werden können. Ein Training würde dann nur noch die übrigen Datenpunkte optimieren.

Implementierungsaufwand (3/5) Ebenso wie ein Ansatz mit Neural Radiance Fields muss auch ein auf einer Datenstruktur basierender Ansatz neu implementiert werden. Da die Methode mittels eines Gradientenverfahrens optimiert wird, bieten sich die Frameworks Tensorflow (Abadi et al., 2015) und PyTorch (Paszke et al., 2019) an.

Reife (2/5) Speicherbasiertes inverses Rendering wurde von uns aus dem Gebiet der Neural Radiance Fields aufgrund der unterschiedlichen Funktionsweise ausgegliedert. In Bezug auf die Reife können wir zwischen beiden Methoden jedoch keinen Unterschied ausmachen. Speicherbasiertes inverses Rendering ist ein neues Verfahren, dessen Entwicklung aktuell schnell

voranschreitet. Dennoch sind die in der Literatur beschriebenen Ergebnisse bereits jetzt beeindruckend.

Bekanntheit (4/5) Entsprechend unserer Einschätzung zur Reife, schätzen wir speicherbasiertes inverses Rendering als genauso bekannt wie Neural Radiance Fields ein. Herausragend ist beispielsweise die Publikation von Müller, et al. (2022) . Sie gewann den *SIGGRAPH Best Paper Award* und wurde in *TIME's Best Inventions of 2022* aufgelistet.

Anwendungsbreite (3/5) Die Methode bewegt sich im gleichen Anwendungsbereich wie Neural Radiance Fields. So gibt es beispielsweise Anwendungen in der Rekonstruktion von 3D-Objekten (Müller, et al., 2022) oder in der Computertomographie (Rückert, et al., 2022).

2.2.2.3 Lineare Optimierung

Überblick

Für einen Variablenvektor *x*, eine Matrix *A* und einen Vektor *b* soll eine Belegung der Variablen *x* gefunden werden, sodass $Ax=b$ gilt und für einen Vektor *c* die gewichtete Summe c^Tx maximiert (oder minimiert) wird.

$$\begin{aligned} \max_{x} \quad & c^T x \\ \text{bzgl.} \quad & Ax = b \end{aligned}$$

Beispiel

Wir betrachten eine Landwirtin, die aus ihren Ressourcen den größtmöglichen Gewinn erzielen möchte. Sie hat eine Anbaufläche von 100 Hektar Land zur Verfügung. Auf jedem Hektar kann sie entweder Weizen oder Mais anbauen. Zur Bewässerung stehen ihr insgesamt 600 Einheiten Wasser zur Verfügung. Weizen benötigt 9 Einheiten Wasser pro Hektar, Mais nur eine Einheit. Der Ertrag soll nach der Ernte verkauft werden. Weizen hat einen Wert von 500 Euro pro Hektar. Mais hat einen Wert von 300 Euro pro Hektar.

Gesucht sind die Variablen x_{Weizen} und x_{Mais}, die die Hektar angeben, auf denen die jeweilige Pflanze angebaut wird. Die Landwirtin möchte ihren Gewinn maximieren, deshalb maximiert sie den gesamten Verkaufspreis. Es ergibt sich folgende Problemformulierung:

$$\begin{aligned} \max \quad & (500\, x_{\text{Weizen}} + 300\, x_{\text{Mais}}) \\ \text{bzgl.} \quad & x_{\text{Weizen}} + x_{\text{Mais}} + x_{\text{freies Land}} = 100, \\ & 9\, x_{\text{Weizen}} + 1\, x_{\text{Mais}} + x_{\text{übriges Wasser}} = 600. \\ & x_{\text{Weizen}}, x_{\text{Mais}}, x_{\text{freies Land}}, x_{\text{übriges Wasser}} \geq 0. \end{aligned}$$

Die folgende Abbildung 2.3 zeigt ein 2D-Koordinatensystem über den beiden Variablen x_{Weizen} und x_{Mais}. Die blaue Fläche sind die zulässigen Werte. Der Vektor, der vom Koordinatenursprung ausgeht, ist die Richtung, in die optimiert wird. Für jede Gleichung gibt es eine Gerade, die die Fläche eingrenzt. Das Optimum ist der Punkt, der am weitesten in Richtung der Optimierung liegt. Nach der Lösung des Problems beträgt der größtmögliche Gewinn 42500 Euro, wenn genau 62.5 ha Weizen und 37.5 ha Mais angebaut werden.

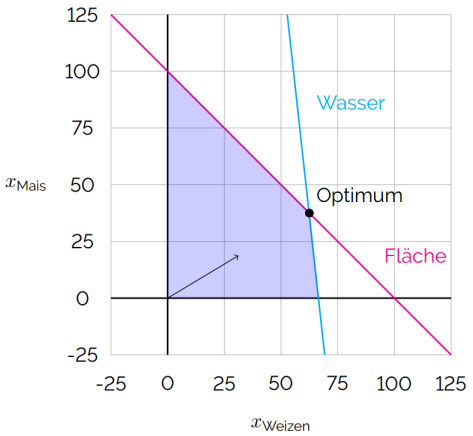

Abbildung 2.3: Visualisierung des Problems. Jedes Kriterium ist eine Gerade, die den Lösungsraum (blau) begrenzt. Die Kostenfunktion bildet einen Vektor. Der Lösungspunkt, der in Richtung der Kostenfunktion am weitesten entfernt liegt, ist das Optimum.

Übertragung

Problemstellung	**Methode**
Verteilung des Iods	eine Variable pro Punkt in der Raumzeit
Kontamination einer Person entlang ihres Weges	gewichtete Summe über den Variablen pro Weg
Messwert der Kontamination einer Person	Werte der gewichteten Summen

In den Kapiteln 1.2 und 1.3 haben wir bereits die Formulierung unseres Problems als ein lineares Gleichungssystem der Form $Ax = m$ beschrieben, in dem jeder Weg eine Gleichung der mit den Schutzfaktoren A gewichteten Summe der Iodkonzentrationen an den Raumzeitpunkten x und der Schilddrüsenmessung m ist. Analog zum Beispiel im vorigen Abschnitt können wir dieses Gleichungssystem als Nebenbedingung einer linearen Optimierung benutzen. Wie auch schon im vorigen Beispiel würde zusätzlich die Lösungsmenge auf positive Werte von x begrenzt werden. Das Optimierungsziel kann die Minimierung der Summe der Iodkonzentrationen x sein. Wenn das Gleichungssystem lösbar ist, würde damit die Lösung gefunden werden, die die wenigste Gesamtmenge an Iod benötigt, um die Schilddrüsenmessungen zu erklären. Wenn keine Lösung gefunden werden kann, müssen dem Gleichungssystem eventuell noch Fehlervariablen hinzugefügt werden, die die personenabhängigen Abweichungen von einer perfekten Lösung modellieren. Diese würden dann priorisiert als Optimierungsziel minimiert werden, um eine möglichst wenig von den Bewegungsdaten abweichende Rekonstruktion zu erhalten.

Bewertung

Ähnlichkeit	5/5	gewichtete Summe passt perfekt zur Kontamination
Bedarf an Zeit	5/5	in der Praxis schnelle Laufzeit, in Theorie kubisch
Bedarf an Speicher	3/5	hoher aber passender Bedarf in unserem Test
Bedarf an Daten	2/5	perfekte Lösung möglich, aber zu wenig Daten vermutet
Unsicherheit	1/5	Unsicherheit ist nicht vorgesehen
Robustheit	1/5	Optimierung anfällig für Artefakte
A priori-Fähigkeit	2/5	zusätzliche Randbedingungen möglich
Implementierungsaufwand	5/5	SciPy bietet fertige Implementierung
Reife	5/5	seit vielen Jahrzehnten erforscht
Bekanntheit	5/5	gut etabliert, vor allem für Planungsprobleme
Anwendungsbreite	5/5	sehr allgemein nutzbares Verfahren

Begründungen

Ähnlichkeit (5/5) Das gegebene Problem besteht aus vielen Linearkombinationen von Strahlungswerten mit bekanntem Akkumulationsergebnis und ist daher perfekt über ein lineares Gleichungssystem modellierbar.

Bedarf an Zeit (5/5) Es existieren effiziente Algorithmen zur Lösung von linearen Optimierungsproblemen. In der Theorie steigt der Zeitbedarf in etwa kubisch mit der Problemgröße. Gerade bei größeren Problemen wie dem unseren kann das zu zu langen

Laufzeiten führen. Dieser theoretische Worst Case scheint in der Praxis jedoch selten zu sein. Auf einem von uns genutzten Laptop (16 GB RAM, Intel i5-12450H) benötigt eine kleine Testimplementierung mit rund zehn Millionen Raumzeitpunkten und 100000 Wegen, die aus jeweils 500 Punkten bestehen, nur wenige Minuten. Es ist damit die Methode, die in unseren Tests am schnellsten ausführbar war.

Bedarf an Speicher (3/5) Es müssen mindestens alle Variablen des Problems und die Wegfaktoren einmal gespeichert werden. Daraus ergibt sich ein vergleichsweise hoher Bedarf an Arbeitsspeicher. Das unter *Bedarf an Zeit* bereits genannte Beispiel benötigte bei uns in etwa 8.3 GB Arbeitsspeicher.

Bedarf an Daten (2/5) Eine exakte Lösung ist mit genügend Daten möglich. In der Praxis wird dies jedoch vermutlich nicht gegeben sein.

Unsicherheit (1/5) Eine Modellierung von Unsicherheit ist bei linearer Optimierung nicht vorgesehen. Eine mögliche Erweiterung mit Unsicherheit ist die stochastische robuste Optimierung, die wir jedoch einzeln betrachten.

Robustheit (1/5) Die lineare Optimierung ist anfällig gegenüber Artefakten in Trainingsdaten, weil sie ein globales Optimum ermittelt, dass exakt aus der Modellierung berechnet wird. Änderungen an einzelnen Parametern haben deshalb einen starken Einfluss auf das Ergebnis.

***A priori*-Fähigkeit (2/5)** Es ist notwendig eine Kostenfunktion anzugeben, die minimiert werden soll. Es ist technisch gesehen jedoch nicht notwendig, dass diese Kostenfunktion Vorwissen abbildet. Weiterhin besteht die Möglichkeit Terme zu der Kostenfunktion hinzuzufügen, die typische Heuristiken, wie Regularisierung, und Smoothing modellieren.

Fixierungen einzelner Werte oder Minimierung des Abstandes von Vorwissen können in zusätzlichen Gleichungen ergänzt werden. Limitiert werden die Möglichkeiten dadurch, dass nur lineare Gleichungen und Terme benutzt werden können. So kann zum Beispiel keine Tikhonov-Regularisierung genutzt werden, da diese einen quadratischen Term in der Kostenfunktion hinzufügen würde.

Implementierungsaufwand (5/5) Die Methode der linearen Optimierung ist ohne Anpassung für unser Problem verwendbar. Dementsprechend können auch fertige Implementierungen ohne Anpassung genutzt werden. Die Python-Bibliothek SciPy (Virtanen et al., 2020) bietet eine einfach zu benutzende Implementierung für lineare Optimierung.

Reife (5/5) Lineare Optimierung ist eine gut erforschte und etablierte Methode. Bei den möglichen Lösungsalgorithmen zeigt sich eine lange Entwicklung hin zu immer schnelleren Verfahren. Online ist eine Vielzahl an Implementierungen zu finden. Das Vorliegen einer Implementierung in SciPy zeigt, dass die Methode zum Standard gehört.

Bekanntheit (5/5) Lineare Optimierung ist eine vergleichsweise alte und etablierte Methode. So existiert lineare Optimierung nicht nur in Publikationen, sondern auch in Lehrmaterial für Studierende, Schülerinnen und Schüler (siehe https://www.schulentwicklung.nrw.de/materialdatenbank/material/view/153).

Anwendungsbreite (5/5) Lineare Optimierung findet Anwendung in einer breiten Vielzahl von Problemen (Vanderbei et al., 2020). Oft handelt es sich um Planungsprobleme, bei denen innerhalb von gegebenen Grenzen optimiert werden soll. Beispiele sind Zuweisungsprobleme, Verwendung von Ressourcen oder Zuordnung von Aufgaben.

2.2.2.4 Stochastische Robuste Optimierung

Überblick

Stochastische robuste Optimierung ist eine Erweiterung der linearen Optimierung, bei der Unsicherheiten bezüglich der Problemparameter berücksichtigt werden. Die Methode ermittelt eine *konkrete Lösung* für ein *unsicheres Problem*. Eine Lösung mit unsicheren Parametern ist nicht leicht. Näherungsweise kann man deshalb eine endliche Menge an Ausprägungen der unsicheren Parameter betrachten und für diese eine Lösung ermitteln.

Typische Anwendungsfälle sind Probleme, bei denen mit in der Zukunft liegenden Ereignissen geplant werden muss.

Beispiel

Wir betrachten das bereits unter linearer Optimierung vorgestellte Beispiel einer Landwirtin. In der originalen Formulierung hat jede Pflanze einen festen Bedarf an Bewässerung. Dieser Bedarf kann in der Praxis jedoch abhängig vom Wetter sein. Fällt viel Regen, wird weniger Bewässerung nötig. Dann wäre es von Vorteil, mehr von den teuren Pflanzen anzubauen, die mehr Wasser benötigen. Fällt wenig Regen, wird mehr Bewässerung nötig. Dann könnte ein Teil der Ernte verloren gehen. In diesem Fall wäre es besser, mehr von den billigeren Pflanzen anzubauen, die weniger Wasser benötigen. Dieser Zusammenhang wird modelliert, indem der Verbrauch an Wasser als eine Zufallsvariable modelliert wird.

Eine mögliche Lösung dieses Problems ist, im Mittel gut für alle Ausprägungen der Zufallsvariablen zu sein. Eine andere mögliche Lösung ist, das schlechteste Szenario zu optimieren, um eine gute untere Schranke für das Risiko zu schaffen. Die Wahl der Eigenschaften der gewünschten Lösung ist Teil der Modellierung.

Die Menge der betrachteten konkreten Ausprägungen ist in der Regel unendlich. Um dennoch verschiedene Szenarien berücksichtigen zu können, werden Monte-Carlo-Methoden benutzt. Bei diesen wird eine endliche Anzahl an konkreten Szenarien gewählt, für die die Optimierung durchgeführt wird. Vereinfacht kann man sagen, dass das Problem für jede der gewählten Ausprägungen gelöst wird.

Übertragung

Problemstellung	**Methode**
Verteilung des Iods	Eine Variable pro Abschnitt in der Raumzeit
Kontamination einer Person entlang ihres Weges	Mit Zufallsvariablen gewichtete Summe über den Variablen pro Weg
Messwert der Kontamination einer Person	Werte der gewichteten Summen

Wir orientieren uns am Beispiel in Abschnitt 1.3. Die Raumzeit wird in Abschnitte unterteilt, die jeweils eine Variable zugeordnet bekommen. Für jeden Weg wird dann über den mit den Schutzfaktoren gewichteten Raumzeitpunkten und dem Messwert eine Gleichung gebildet. Unterschiedlich ist nun, dass die Parameter in dieser Gleichung Zufallsvariablen sind, die einer Verteilung unterliegen. Typischerweise ist diese Verteilung eine auf den gegebenen oder geschätzten Wert zentrierte Gaußverteilung. Damit wird modelliert, dass die Faktoren, die den Messwert beeinflusst haben, in der Realität von den durch die Befragungen ermittelten Messwerten abweichen können. Um das Problem zu lösen, wird deshalb eine Reihe von

konkreten zufälligen Ausprägungen dieser Faktoren gewählt. Die Lösung wird durch eine Optimierung bezüglich aller dieser gewählten Ausprägungen erlangt.

Bewertung

Ähnlichkeit	5/5	Modellierung passt perfekt, Unsicherheit bildet Realität ab
Bedarf an Zeit	4/5	vergleichbar mit mehreren linearen Optimierungen
Bedarf an Speicher	2/5	viele konkrete Szenarien zu betrachten
Bedarf an Daten	3/5	weniger als lineare Optimierung
Unsicherheit	4/5	Unsicherheit der Wege, keine Unsicherheit der Lösung
Robustheit	5/5	Robust gegenüber Daten mit größter Unsicherheit
A priori-Fähigkeit	3/5	Unsicherheit der Wege notwendig
Implementierungsaufwand	4/5	mit mehr Aufwand als lineare Optimierung verbunden
Reife	5/5	gut erforscht
Bekanntheit	4/5	weniger bekannt als lineare Optimierung
Anwendungsbreite	5/5	für verschiedene unsichere Probleme anwendbar

Begründungen

Ähnlichkeit (5/5) Das gegebene Problem lässt sich, wie schon bei der linearen Optimierung beschrieben, gut als gewichtete Summe modellieren. Die stochastische robuste Optimierung bietet darüber hinaus noch eine Modellierung der Unsicherheit der Parameter entlang der Wege. Das ist besonders deshalb interessant, weil diese Parameter aus den persönlichen Aussagen der Menschen geschätzt werden müssen und vermutlich starken Schwankungen unterliegen.

Bedarf an Zeit (4/5) Eine stochastische robuste Optimierung besteht etwas vereinfacht betrachtet aus vielen linearen Optimierungen. Entsprechend ist der Zeitbedarf ein Vielfaches des Zeitbedarfs einer linearen Optimierung. Der Zeitbedarf hängt von der gewünschten Anzahl an Ausprägungen und dem möglichen Parallelisierungsgrad ab. Letzterer ist vor allem durch die Menge an verfügbarem Arbeitsspeicher begrenzt.

Bedarf an Speicher (2/5) Eine stochastische robuste Optimierung betrachtet im Gegensatz zu einer linearen Optimierung mehrere unterschiedliche Ausprägungen der Modellparameter. Diese Ausprägungen müssen gespeichert werden. Je nach Modellierung und Lösung des Problems müssen diese Ausprägungen gleichzeitig im Speicher vorliegen oder können nacheinander betrachtet werden. Damit ist der Speicherbedarf entweder ähnlich zu dem der linearen Optimierung oder um ein Vielfaches größer.

Bedarf an Daten (3/5) Aufgrund der Modellierung von Unsicherheit fallen einzelne Datenartefakte weniger ins Gewicht als bei der linearen Optimierung. Deshalb erwarten wir, dass im Gegensatz zur linearen Optimierung aus wenigen Daten robuste Ergebnisse gewonnen werden können. Die Methode benutzt standardmäßig jedoch keine *a priori*-Verteilung. Wir erwarten deshalb einen größeren Bedarf an Daten als für bayessche Methoden (siehe Abschnitt 2.2.2.5).

Unsicherheit (4/5) Eine Formulierung von Unsicherheiten der Problemparameter ist möglich und deckt eventuelle Ungenauigkeiten bei der Erhebung der Bewegungsdaten ab. Eine Angabe von Unsicherheiten bei der rekonstruierten Iod-Verteilung ist nicht möglich.

Robustheit (5/5) Die Bewegungsdaten der Menschen sind vermutlich die Daten in unserem Problem mit der höchsten Unsicherheit. Bei einer stochastischen robusten Optimierung werden genau die von diesen Daten abgeleiteten Parameter als unsicher modelliert. Damit kann während der Optimierung diese Unsicherheit berücksichtigt werden. Die Methode wird dadurch robuster gegenüber Artefakten bei der Datenerhebung.

***A priori*-Fähigkeit (3/5)** Alle Möglichkeiten der linearen Optimierung gelten für die stochastische Optimierung ebenfalls. Es besteht darüber hinaus die Notwendigkeit, Annahmen über die Verteilung der unsicheren Problemparameter zu modellieren. An dieser Stelle kann damit ein mögliches Vorwissen in Form einer Abschätzung der Unsicherheit in die Modellierung eingebracht werden. Genau diese Modellierung ist der Vorteil der Methode gegenüber linearer Optimierung, die diese Unsicherheit nicht abbilden kann.

Implementierungsaufwand (4/5) Die Python-Bibliothek RSOME (Chen & Xiong, 2021; Chen et al., 2020) implementiert stochastische robuste Optimierung. Mithilfe der linearen Optimierung von SciPy (Virtanen et al., 2020) kann eine eigene Implementierung von robuster stochastischer Optimierung angefertigt werden.

Reife (5/5) Die stochastische robuste Optimierung baut auf der bereits sehr gut erforschten linearen Optimierung auf. Die von uns untersuchte Literatur vermittelt einen sehr fundierten Eindruck der Theorie, die in die Praxis übertragen werden kann.

Bekanntheit (4/5) Die stochastische robuste Optimierung schätzen wir als etwas weniger bekannt als die klassische lineare Optimierung ein. Als eine Erweiterung der linearen Optimierung ist sie dennoch etabliert und hat ihren klaren Platz in der Literatur.

Anwendungsbreite (5/5) So wie lineare Optimierung wird stochastische robuste Optimierung für Planungsprobleme benutzt. Ein besonderes Augenmerk liegt dabei auf Problemen mit unsicheren Komponenten. Das sind beispielsweise Probleme, deren Planung von zukünftigen Ereignissen, wie Wetter oder Aktienkursen, abhängen.

2.2.2.5 Bayessche Inverse Probleme

Überblick

Im Bereich der bayesschen inversen Probleme werden inverse Probleme mit Methoden der bayesschen Statistik gelöst. Bayessche inverse Probleme sind inverse Probleme. Sie berücksichtigen die Unsicherheiten in den zu rekonstruierenden Variablen. Die Lösung eines bayesschen inversen Problems ermittelt eine *unsichere Lösung* für ein *unsicheres Problem*. Verschiedene probabilistische Modelle können verwendet werden, um die Unsicherheiten zu modellieren. Häufig werden für die Modellierung der Unsicherheit Gaußverteilungen verwendet. So wird für jede Variable in einer Rekonstruktion zusätzlich eine Varianz ermittelt.

Die Optimierung des Problems wird durch Inferenz mittels des Satzes von Bayes durchgeführt. Seien x die gesuchte Rekonstruktion und m eine Evidenz, zum Beispiel in Form einer Messung. Dann ist der Satz von Bayes die folgende Gleichung aus Wahrscheinlichkeiten über x und m:

$$\mathrm{P}(x \mid m) = \frac{\mathrm{P}(m \mid x)\mathrm{P}(x)}{\mathrm{P}(m)}.$$

Die Wahrscheinlichkeit P(x | m) gibt die *a posteriori*-Wahrscheinlichkeit (kurz: Posterior) an, die misst, wie wahrscheinlich eine Rekonstruktion x für eine gegebene Evidenz m ist. Das x, das diese Wahrscheinlichkeit maximiert, ist die beste Lösung für unser Rekonstruktionsproblem.

Die Wahrscheinlichkeit P(x) gibt die *a priori*-Wahrscheinlichkeit (kurz: Prior) einer Rekonstruktion x an. Sie modelliert, für wie wahrscheinlich eine Lösung gehalten wird. Durch sie kann Einfluss auf die Beschaffenheit einer Lösung genommen werden.

Die Wahrscheinlichkeit P(x | m) gibt die *Likelihood* an, die misst, wie wahrscheinlich die Messung m für eine mögliche Rekonstruktion x ist. Damit wird die Plausibilität einer Rekonstruktion bezüglich der gesammelten Evidenz modelliert.

Die Wahrscheinlichkeit P(m) gibt die Wahrscheinlichkeit der Evidenz m unter Betrachtung aller möglichen Rekonstruktionen x an. Damit ist sie jedoch unabhängig von einem speziellen x und konstant für ein konstantes m. Für unsere Formulierung von bayesschen inversen Problemen ist ein festes m gegeben, für das ein möglichst gutes x gefunden werden soll. Damit bildet P(m) lediglich einen konstanten Faktor in der Gleichung. Dieser hat jedoch keinen Einfluss bei der Suche nach dem x mit dem besten Posterior. Deswegen kann P(m) ignoriert werden. Das ist insbesondere deshalb nützlich, weil eine Berechnung von P(m) eine Betrachtung aller möglichen x benötigen würde. Das ist jedoch praktisch oft nicht umsetzbar.

Für die Methode ist zwingend eine Formulierung eines Priors, also einer Annahme über die Beschaffenheit der gesuchten Rekonstruktion, notwendig. In unserem Problem könnte ein solcher Prior durch eine Ausbreitungssimulation des Iods gegeben sein. Dieser Prior der Iod-Konzentration wird dann entsprechend der Bewegungsdaten mittels bayesscher Inferenz angepasst. Der daraus resultierende Posterior der Konzentration ist dann das Ergebnis der Rekonstruktion. Die Wahl eines guten Priors ist also gerade bei wenigen Bewegungsdaten ein wichtiger Bestandteil der Modellierung.

Sollte kein guter Prior verfügbar sein, kann das Fehlen von Vorwissen auch mittels eines gleichverteilten Priors ausgedrückt werden. Somit ist eine Betrachtung des Problems als bayessches inverses Problem auch ohne explizites Vorwissen möglich. Als Vergleich sei an dieser Stelle erwähnt, dass andere Verfahren, die keine Priors verwenden, implizit ebenfalls eine Annahme von Unwissen beinhalten. Bayessche inverse Probleme sind in diesem Teil der

Modellierung lediglich expliziter und geben damit eine Möglichkeit, Vorwissen in die Lösung einzubringen, falls es vorhanden ist. Die Qualität des Vorwissens bei der Rekonstruktion der Iodkonzentrationen, skalierend von völligem Unwissen über grobe Windrichtungen bis hin zu sehr detailliertem Wissen über die Lösung, wie es beispielsweise durch eine Ausbreitungssimulation erhalten werden kann, ist bei der Modellierung frei wählbar.

Beispiel

Wir betrachten als Beispiel die Rekonstruktion eines Signales, wenn es nur unscharf gemessen werden kann. Wir visualisieren dieses Beispiel im Folgenden und verweisen für eine detaillierte mathematische Details auf die originale Ausführung (Kekkonen, 2019). Mit einem unscharfen Signal meinen wir ein Signal, bei dem die Werte “verwischt” sind. Um eine Lösung zu ermitteln, werden zwei Wahrscheinlichkeiten modelliert und gemeinsam optimiert: die Wahrscheinlichkeit, dass eine Rekonstruktion das unscharfe Signal erklärt (Likelihood) und die Wahrscheinlichkeit, dass wir diese Rekonstruktion für plausibel halten (Prior).

Wir nehmen an, dass wir aus Erfahrung wissen, dass das gesuchte Signal glatt ist, benachbarte Werte also ähnlich zueinander sind. Entsprechend dieser Annahme wird der Prior modelliert, indem davon ausgegangen wird, dass optimalerweise ein Punkt genau der Mittelwert seiner Nachbarpunkte ist. Weil das nicht immer der Fall sein kann, wird die Wahrscheinlichkeit eines Punktes mittels einer Gaußverteilung um diesen Mittelpunkt modelliert. Abbildung 2.4 veranschaulicht diese Modellierung.

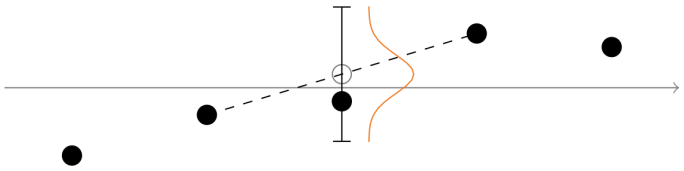

Abbildung 2.4: Modellierung der Smoothness. Der dritte Punkt wird beispielhaft betrachtet. Er ist gaußverteilt (orange) um den Mittelpunkt (grau) des zweiten und vierten Punktes.

Abbildung 2.5 zeigt die Modellierung des verwischten gemessenen Signals aus dem originalen Signal. Die Modellierung des Problems orientiert sich an der Entstehung des gemessenen Signals und verläuft im Bild von unten nach oben. Aus dem Blickwinkel der Rekonstruktion ist das gemessene Signal gegeben und das originale Signal soll rekonstruiert werden. Das entspricht der entgegengesetzten Richtung von oben nach unten.

Wir folgen zunächst der Modellierung des gemessenen Signals. Von unten nach oben wird das originale Signal zunächst mittels einer Gaußverteilung verwischt. Die Berechnung ist für den dritten Punkt beispielhaft mit eingezeichnet. Zunächst wird eine Gaußverteilung auf dem betrachteten Punkt zentriert. Dann werden alle Punkte nach dieser Verteilung gewichtet aufsummiert. Das Ergebnis ist der verwischte Wert des dritten Punktes. Zur Berechnung aller Punkte des verwischten Signales muss diese Berechnung für jeden Punkt wiederholt werden.

Zusätzlich zur Verwischung modellieren wir außerdem im letzten Schritt, dass die gemessenen unscharfen Datenpunkte jeweils einer gaußverteilten Schwankung unterliegen dürfen. Die Likelihood berechnet sich dann mittels dieser gaußverteilten Schwankung aus dem Unterschied zwischen den aus der Rekonstruktion mittels Verwischung ermittelten Werten (grau) und den gemessenen Werten des unscharfen Signals (schwarz).

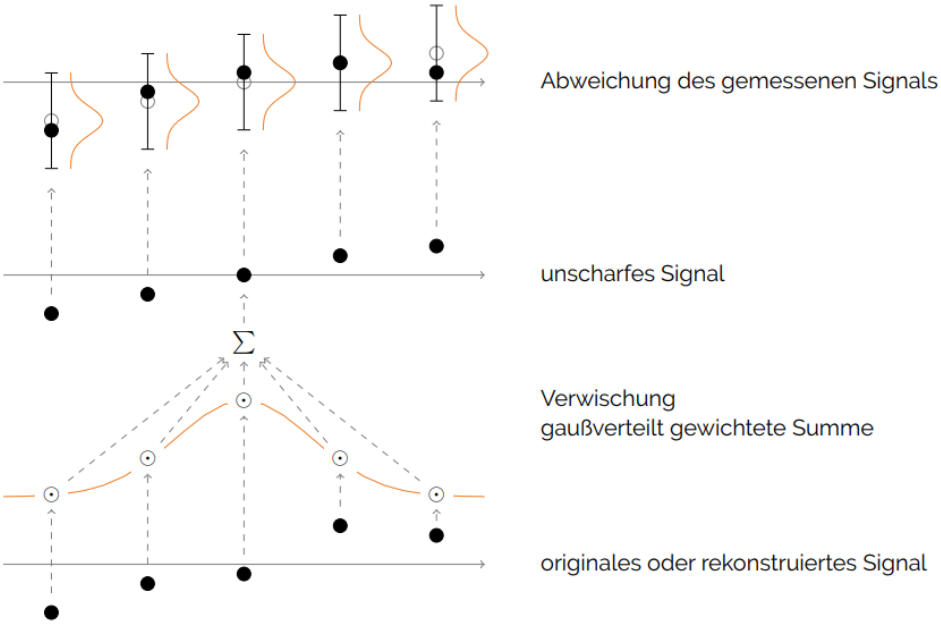

Abbildung 2.5: Modellierung des unscharfen verrauschten Signals aus dem originalen Signal. Das Problem ist in umgekehrter Reihenfolge definiert: aus einem verrauschten unscharfen Signal (Messung) soll das originale Signal rekonstruiert werden.

Der Posterior setzt sich aus dem Prior und der Likelihood zusammen. Deshalb ist auch der Posterior gaußverteilt und hat ein eindeutiges globales Optimum, das sich exakt berechnen lässt. Ebenfalls lässt sich die Varianz exakt berechnen. Eine exakte Angabe dieser Berechnungen würde eine umfassende mathematische Formulierung des Problems benötigen. Wir verzichten deshalb an dieser Stelle darauf und verweisen erneut auf die originale Quelle, in der diese Formulierung samt der dazugehörigen Lösung zu finden ist (Kekkonen, 2019).

Übertragung

Problemstellung	**Methode**
Verteilung des Iods	Zufallsvariablen (mit Varianz)
Kontamination einer Person entlang ihres Weges	gewichtete Summen der Variablen und zufälliges Rauschen pro Weg
Messwert der Kontamination einer Person	Werte der gewichteten Summen

In Kapitel 1 haben wir unser Problem als ein inverses Problem definiert. Die Modellierung als bayessches inverses Problem fügt dieser Definition einen Summanden für zufälliges Rauschen, μ, hinzu:

$$Ax + \mu = m.$$

Das Rauschen modelliert einen Fehler zwischen der theoretischen Akkumulation und der Messung. Dieser Fehler wird als gaußverteilt mit der Wahrscheinlichkeit $P_\mu(\mu)$ angenommen. Die Faktoren der Wege A sind konstant und gegeben. Die Variablen x und m werden als Zufallsvariablen modelliert.

Gesucht ist der Posterior über allen Rekonstruktionen *x*, gegeben der Messwerte *m*, die mittels des Satz von Bayes berechnet wird:

$$\mathrm{P}(x \mid m) = \frac{\mathrm{P}(m \mid x)\mathrm{P}(x)}{\mathrm{P}(m)}.$$

Die Iod-Verteilung *x* mit dem höchsten Posterior ist dann die Lösung des Problems. Aus dem Posterior lässt sich außerdem die Unsicherheit der Rekonstruktion *x* in Form der Varianz ablesen. Gäbe es an einem Raumzeitpunkt sehr wenig Bewegungsdaten, wären vermutlich verschiedene Rekonstruktionen für diesen Punkt plausibel und die Varianz an diesem Punkt wäre hoch. Umgekehrt wäre an einem Punkt mit viel Evidenz nur eine einzelne Lösung plausibel und die Varianz entsprechend gering.

Um diese Berechnung durchführen zu können, werden die drei Wahrscheinlichkeiten auf der rechten Seite der Gleichung benötigt. Der Prior P(*x*) gibt basierend auf Vorwissen die Wahrscheinlichkeit einer Iod-Verteilung *x* an und ist beispielsweise durch eine Simulation gegeben. Er ist eine Wahrscheinlichkeitsverteilung über Ausprägungen von *x*. Die Likelihood P(*m* | *x*) gibt an, ob die Messwerte *m* mithilfe der Iod-Verteilung *x* erklärt werden können. Sie ist definiert über das Rauschen μ, das den Fehler zwischen *Ax* und *m* misst, als:

$$\mathrm{P}(m \mid x) = \mathrm{P}^{\star}(m - Ax).$$

Wir annotieren P⋆(*m* - *Ax*) mit hochgestelltem ⋆, um abzugrenzen, dass es sich um eine Wahrscheinlichkeitsverteilung über Ausprägungen von μ und nicht von *x* handelt. Die dritte Wahrscheinlichkeit P(*m*) gibt die Wahrscheinlichkeit der Messung unter allen Ausprägungen von *x* an. Sie ist nicht leicht zu berechnen. Weil P(*m*) jedoch nicht von *x* abhängt, ist das beste *x* für P(*x* | *m*) unabhängig von P(*m*). Für die Bestimmung einer Lösung reicht es, die ersten zwei Wahrscheinlichkeiten zu berechnen. Daraus ergibt sich, dass eine Verteilung der Iod-Konzentration dann am besten ist, wenn sie sowohl plausibel bezüglich unseres Vorwissens ist, als auch die Messdaten gut erklären kann.

Bewertung

Ähnlichkeit	5/5	inverses Problem
Bedarf an Zeit	2/5	*nicht abschätzbar*
Bedarf an Speicher	2/5	eventuell sehr große Matrizen
Bedarf an Daten	5/5	auch mit wenig Daten verwendbar
Unsicherheit	5/5	Messung und Rekonstruktion werden unsicher modelliert
Robustheit	5/5	durch Modellierung von Unsicherheit sehr robust
A priori-Fähigkeit	5/5	Angabe einer *a priori*-Verteilung nötig
Implementierungsaufwand	1/5	vergleichsweise groß mit theoretischem Anteil
Reife	5/5	bayessche Methoden sind gut erforscht
Bekanntheit	3/5	etabliert in einzelnen Gebieten
Anwendungsbreite	5/5	Anwendung auf Vielzahl von Problemen

Begründungen

Ähnlichkeit (5/5) Da es sich bei unserem Problem um ein inverses Problem handelt, ist die Übertragbarkeit gegeben. Die Erweiterung des Modells um die bayessche Betrachtung von Unsicherheiten fügt sich gut in den gegebenen realistischen Kontext. Die Unsicherheiten, die beim Erheben der Bewegungsdaten auftreten können, werden im Modell berücksichtigt. Zusätzlich wird jeder Punkt der Rekonstruktion mit einer Unsicherheit bewertet. Das ist besonders deshalb interessant, weil die Abdeckung der Raumzeitpunkte durch die Bewegungen sehr unterschiedlich ausfallen kann. In Bereichen mit viel Evidenz kann das Modell dann eine hohe Sicherheit ausdrücken. In Bereichen mit wenig Evidenz kann entsprechend vor einer unsicheren Rekonstruktion gewarnt werden.

Bedarf an Zeit (2/5) Der Bedarf an Zeit hängt stark von der konkreten Modellierung der Wahrscheinlichkeitsverteilungen ab. Wir vermuten, dass die benötigte Zeit kubisch mit der Problemgröße steigt. Eine genaue Abschätzung ist zum jetzigen Zeitpunkt schwierig. Wir konnten der Literatur keine Informationen dazu entnehmen.

Bedarf an Speicher (2/5) Wie der Bedarf der Zeit ist auch der Bedarf an Speicher stark von der konkreten Modellierung der Wahrscheinlichkeitsverteilungen abhängig. Im Worst Case müssen beispielsweise für multivariate Gaußverteilungen Kovarianz-Matrizen gespeichert werden, die den verfügbaren Speicher weit überschreiten. Ist es möglich diese Matrizen über Funktionen zu modellieren, so ist deren explizite Speicherung jedoch nicht nötig.

Bedarf an Daten (5/5) Aufgrund des notwendigen Priors kann die Methode auch mit wenigen Daten bereits gute Ergebnisse liefern, wenn der Prior passend gewählt wurde. Im Verlauf der Datenaufnahme können ausgehend von einem Prior wiederholt bayessche Updates durchgeführt werden, wenn neue Messdaten vorliegen. Die Ergebnisse können so während des Einsatzes im Verlauf der Zeit fortführend verfeinert werden.

Unsicherheit (5/5) Eine Modellierung von Unsicherheit ist der Kernaspekt der Modellierung von bayesschen inversen Problemen. Sie können Unsicherheit hinsichtlich des Fehlers zwischen Akkumulation und Messergebnis und vor allem hinsichtlich der rekonstruierten Lösung abbilden. Eine Ausgabe der Unsicherheit zu einer Lösung ist möglich.

Robustheit (5/5) Bayessche Modelle können aufgrund der Modellierung von Unsicherheit und verrauschten Eingabedaten mit fehlerbehafteten Daten verwendet werden. Außerdem können sie durch die Angabe von Unsicherheit im Ergebnis auch mit Punkten mit wenig Evidenz transparent umgehen. Sie sind die robusteste Methode in unserer Recherche.

***A priori*-Fähigkeit (5/5)** Eine *a priori*-Verteilung ist nötig und wirkt sich vor allem bei wenig Daten auf das erhaltene Ergebnis aus. Die Wahl einer guten *a priori*-Verteilung ist ein wesentlicher Bestandteil der Lösung des Problems. Es können alle Formen von Vorwissen kodiert werden. Das bedeutet insbesondere jedoch auch, dass das Fehlen von Vorwissen mittels einer Gleichverteilung modelliert werden kann.

Implementierungsaufwand (1/5) Für die Implementierung von Gauß-Prozessen stehen mit GPyTorch (Gardener et al., 2018), PyMC (Wiecki et al., 2023) und PyLops (Ravasi & Vasconcelos, 2020) verschiedene Python-Bibliotheken bereit. Eine Implementierung der Methode würde einen vergleichsweisen hohen theoretischen Anteil haben, um eine gute Modellierung zu erreichen.

Reife (5/5) Bayessche Methoden zur Anwendung auf inverse Probleme wurden bereits 1960 in der Form von Kalman-Filtern beschrieben. Moderne Publikationen (Dashti & Stuart, 2013)

zeigen, dass das Themengebiet auch weiterhin Beachtung findet. Die Theorie rund um die oft in diesem Kontext benutzten Gauß-Verteilungen ist gut erforscht.

Bekanntheit (3/5) Wir haben den Eindruck, dass bayessche Methoden in manchen Themengebieten wie zum Beispiel den Geowissenschaften sehr etabliert sind. Dennoch sehen wir bei ihnen nicht den Bekanntheitsgrad einer linearen Optimierung oder den aktuell stark im Aufschwung begriffenen Neural Radiance Fields gegeben.

Anwendungsbreite (5/5) Die Formulierung und Lösung als bayessche inverse Probleme wird auf ein breites Spektrum an Problemen angewandt. Wir konnten beispielsweise Anwendungen in den Geowissenschaften (Bui-Thanh et al., 2013; Michalak & Kitanidis, 2003), der Physik (Del Debbio et al. , 2022), der Computertomographie (Tarvainen et al., 2013) und beim Entrauschen von Bildern (Wang R. & Tao D., 2014) finden.

2.2.2.6 Ergänzte Methode: Gaussian Splatting

Überblick

Gaussian Splatting ist eine im August 2023 neu veröffentlichte Methode, die sich seitdem im Bereich Computer Vision als Weiterentwicklung des Ansatzes von Neural Radiance Fields (NeRFs) etabliert hat. (Originalpublikation: Kerbl, et al. (2023): 3D Gaussian Splatting for Real-Time Radiance Field Rendering)

Wie bei NeRFs und dem inversen Rendering geht es dabei in der Originalmethode um die Rekonstruktion von 3D-Objekten aus Fotos, zu denen die Kameraposition und der Blickwinkel bekannt sind. Anders als bei diesen beiden Ansätzen erfolgt die Rekonstruktion allerdings nicht durch ein neuronales Netz oder eine Datenstruktur, sondern durch die Verteilung von dreidimensionalen Gauß-Verteilungen (Gaussians oder gaußförmige Volumen) im 3D-Raum. Diese werden während des Trainings des Modells so gelernt, also im Raum positioniert, geformt und bezüglich ihrer Farbe und Dichte angepasst, dass sie das rekonstruierte 3D-Objekt bestmöglich repräsentieren.

Hierzu werden, wie bei der Methode inverses Rendering, die Kamerastrahlen genutzt. Diese durchkreuzen auf ihrem Weg durch den Raum die darin befindlichen Gaussians und akkumulieren dabei Farb- und Dichte-Werte. Das Training erfolgt ebenso über ein Gradientenverfahren, bei dem durch einen Abgleich zwischen den originalen Trainingsbildern und den rekonstruierten Bildern im Prozess das Modell verbessert wird. Des Weiteren können im Laufe des Trainings, durch ein Verfahren namens „Adaptive Density Control“, wo nötig weitere Gaussians ergänzt werden oder überflüssige Gaussians gelöscht werden.

Abbildung 2.6 illustriert die Rekonstruktion der bereits zuvor als Beispiel genutzten Wolke durch Gaussians. Diese sind nach dem Training so im Raum angeordnet und geformt, dass sie die Wolke bestmöglich nachbilden.

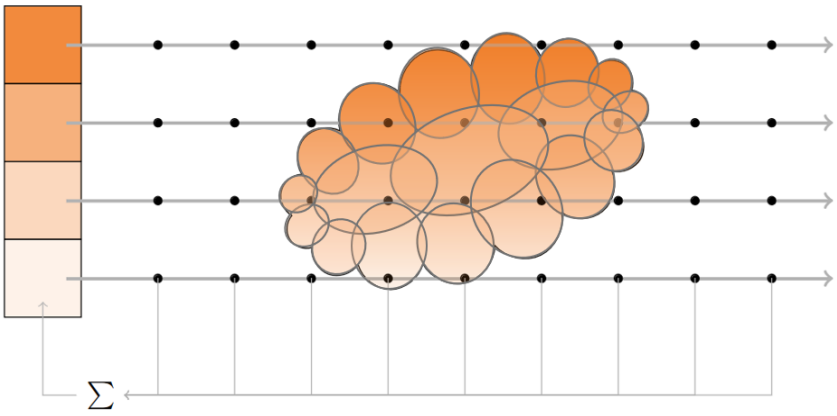

Abbildung 2.6: Berechnung der Pixelfarben mittels der Repräsentation über Gaussians

Beispiel

Die 3D-Szenerie eines Waldes soll rekonstruiert werden. Hierzu werden zunächst die Input-Daten für das Modell erzeugt: Fotos aus verschiedenen Blickwinkeln werden aufgenommen und die Kamerapositionen rekonstruiert. Als nächstes werden die Startpositionen der Gaussians im 3D-Raum initialisiert. Dies kann entweder durch eine zufällige Verteilung der Startpunkte im Raum erfolgen oder gezielt durch ein sogenanntes „Structure from Motion“ Verfahren. Auf all diesen Startpositionen werden nun Gaussians mit festgelegten Start-Werten für ihre Größe, Form,

Rotation, Farbwerte und Transparenz erzeugt. Anschließend werden die Parameter dieser Gaussians (Position, Größe, Form, Rotation, Farbwerte, Transparenz) mit einem Gradientenverfahren optimiert („trainiert“). Gleichzeitig wird die Anzahl der Gaussians optimiert. Hierzu werden zum einen dort zusätzliche Gaussians ergänzt, wo die Szene schlecht rekonstruiert ist. Zum anderen werden Gaussians gelöscht, die nicht mehr benötigt werden, weil sie nahezu durchsichtig sind. Im Laufe der Iterationen durch diesen Prozess entsteht das finale 3D-Modell der rekonstruierten Szene im Wald. Eine Visualisierung dieses Beispiels findet sich auf der Webseite der Original-Publikation von Kerbl et al. (2023).

Übertragung

Problemstellung	**Methode**
Verteilung des Iods	Repräsentiert durch die gelernte Verteilung und Eigenschaften der Gaussians im 3D-Raum
Kontamination einer Person entlang ihres Weges	Gewichtete Summe über den Farben pro Strahl
Messwert der Kontamination einer Person	Farbwert eines Pixels

Anstelle des direkten Lernens und Speicherns der Iodkonzentrationen der Raumzeit (im Folgenden Belastungswerte genannt) in der Datenstruktur beim inversen Rendering werden die Belastungen hier durch die gelernte Verteilung und Eigenschaften der Gaussians im 3D-Raum über die Parameter: Position, Größe, Form und Belastungswert (statt Farb- und Transparenzwerten) repräsentiert, aus denen die Belastungswerte der Punkte in der Raumzeit berechnet werden können. Intuitiv betrachtet handelt es sich also um eine Repräsentation der radioaktiven Wolke über (sich gegebenenfalls überlappende) Volumen in der Raumzeit anstelle einer Zerschneidung der Raumzeit in einzelne Zellen mit gelernten Belastungswerten wie beim inversen Rendering. Wie beim inversen Rendering lassen sich die Kamerastrahlen in der Original-Methode auf die Wege der Personen in der Raumzeit in unserem Anwendungsfall übertragen. Die Akkumulation der radioaktiven Belastung einer Person erfolgt durch das Durchschreiten von Gaussians mit bestimmten Belastungswerten auf den Punkten ihres Weges durch die Raumzeit, woraus eine gewichtete Summe gebildet wird.

Wie schon bei den NeRFs und beim inversen Rendering sind die Input-Daten bei uns nicht Fotos und Kamerapositionen, sondern die Daten des Weges, den eine Person in der Raumzeit zurückgelegt hat, und die an der Schilddrüse gemessene Belastung dieser Person am Ende ihres Weges. Die Initialisierung der Positionen der Gaussians erfolgt zufällig im 3D-Raum mit bestimmten Start-Werten für ihre Länge, Breite und Höhe und einem initialen Belastungswert. Diese Parameter werden dann durch den wiederholten Abgleich der echten Schilddrüsen-Messwerte am Ende des Weges einer Person mit den berechneten Messwerten auf Basis der aktuellen Rekonstruktion der Belastungswerte in der Raumzeit (aktuelle Gauss-Volumen Verteilung des Modells) optimiert. Gleichzeitig werden im Rahmen des „Adaptive Density Control“-Verfahrens Gaussians gelöscht, die einen sehr geringen (fast null) Belastungswert aufweisen, da diese (nahezu) unbelastete Luft darstellen. Außerdem werden zusätzliche Gaussians entlang von bislang unterrekonstruierten Wegen (bezüglich des berechneten Belastungswertes in der Notfallstation) eingestreut. Nach einigen Iterationen durch diesen

Prozess entsteht so eine möglichst gute Rekonstruktion der radioaktiven Belastungswerte in der Raumzeit, über die im Raum verteilten, fertig trainierten Gaussians.

Besonderheit unserer Implementierung

Die von uns entwickelte Übertragung der Methode auf unseren Anwendungsfall beinhaltet eine Veränderung bezüglich der genutzten geometrischen Objekte. Anstelle von Gaussians (dreidimensionale Gaußverteilungen, Ellipsoide) verwenden wir Cuboids (Quader). In unserem Anwendungsfall spielt die Berechnung von Überlappungsvolumen zwischen den Rasterzellen der 3D-Raumzeit und den verwendeten geometrischen Objekten eine große Rolle. Hierbei wären unter Verwendung von Gaussians sowohl die Implementierung als auch die Berechnungen im Trainingsprozess des Modells sehr viel aufwendiger. Dies würde den Bedarf an Zeit und Speicher des Modells im Training erheblich erhöhen, ohne dass die geometrische Form der Gaussians in unserem Fall größere Vorteile gegenüber den Cuboids mit sich bringen würde. Die Verwendung von Gaussians im originalen Anwendungsfall ist sinnvoll, da damit feinste geometrische Details eingefangen werden können, was in unserem Anwendungsfall aber weniger entscheidend ist, da wir vor allem die übergeordneten raumzeitlichen Strukturen der Strahlungsverteilung modellieren wollen. Weitere Details zur Implementierung finden sich in Kapitel 4.3.5. Die Implementierung orientiert sich ansonsten an der von Kerbl, et al. (2023) beschriebenen Idee des Gaussian Splatting, verwendet aber nicht deren Software, sondern unsere eigene, davon unabhängige Implementierung, die auf unseren Anwendungsfall zugeschnitten ist.

Bewertung

Ähnlichkeit	3/5	Gewichtete Summe, Gewichte werden mit gelernt
Bedarf an Zeit	4/5	Schneller als NeRFs und inverses Rendering
Bedarf an Speicher	2/5	Höherer Bedarf als bei NeRFs und inversem Rendering
Bedarf an Daten	4/5	Auflösung an Datenmenge anpassbar, weniger besuchte Punkte besser rekonstruierbar als mit inversem Rendering
Unsicherheit	2/5	Wird nicht modelliert, könnte aber anhand von Pfad-Überlappung ergänzt werden
Robustheit	4/5	Robustheit durch Erlernen übergeordneter räumlicher Strukturen der radioaktiven Belastung
A priori-Fähigkeit	2/5	Vorwissen nicht nötig und bislang nicht vorgesehen, Ansätze zum Einbezug von Vorwissen aber denkbar
Implementierungsaufwand	3/5	Neuimplementierung nötig, mit PyTorch gut umsetzbar
Reife	2/5	Sehr aktuelle Forschung
Bekanntheit	4/5	Hohe Aufmerksamkeit im letzten Jahr
Anwendungsbreite	3/5	Anwendungen im Bereich 3D-(Bild-)Rekonstruktion

Begründungen

Ähnlichkeit (3/5) Der Gaussian Splatting-Ansatz ist in der Wissenschaft eine Weiterentwicklung von Neural Radiance Fields und inversem Rendering mit der gleichen Grundidee bezüglich der Nutzung der Kamerastrahlen, die auf ihrem Weg durch den Raum Farbwerte als gewichtete Summe akkumulieren. Dementsprechend ist die Ähnlichkeit genauso zu bewerten

wie bei den beiden anderen Methoden. Gegebenenfalls liegt die Repräsentation der radioaktiv belasteten Wolke über Volumen in der Raumzeit sogar noch näher am Anwendungsfall, als die Repräsentation über ein neuronales Netz bei NeRFs oder die Datenstruktur der Raumzeitpunkte beim inversen Rendering.

Bedarf an Zeit (4/5) Die höhere Geschwindigkeit des Trainingsprozesses bei gleicher oder besserer Qualität ist eine der Haupteigenschaften, die in der Originalveröffentlichung der Methode im Vergleich zu Ansätzen der Neural Radiance Fields und inversem Rendering immer wieder hervorgehoben werden (Kerbl, et al., 2023). Dieses Kriterium ist daher positiv zu bewerten.

Bedarf an Speicher (2/5) Während Qualität und Geschwindigkeit Vorteile des Modells gegenüber Neural Radiance Fields und inversem Rendering sind, ist der Bedarf an Speicher ein Nachteil. In der Originalveröffentlichung zu Gaussian Splatting von Kerbl, et al. (2023) wird von einem signifikant höheren Speicherbedarf im Vergleich zu den anderen Ansätzen berichtet.

Bedarf an Daten (4/5) Der Bedarf an Daten kann ähnlich zum inversen Rendering bewertet werden. Je nach dem, wie viele Trainingsdaten bereits vorliegen, kann zunächst in einer gröberen Auflösung gearbeitet werden und bei einer höheren Menge an vorhandenen Pfaddaten auch in feineren Auflösungen trainiert werden. Ein Vorteil gegenüber dem inversen Rendering besteht darin, dass die Belastungen von Pfadabschnitten bereits in einem Zusammenhang stehen, wenn sie durch den gleichen Gaussian (bei uns: Cuboid) verlaufen, ohne dafür explizite Schnittpunkte haben zu müssen. Dadurch können teilweise auch Punkte, durch die weniger Pfade verlaufen, gut rekonstruiert werden, wenn sie durch umliegende Gaussians (bei uns: Cuboids) miterfasst werden.

Unsicherheit (2/5) Eine Modellierung von Unsicherheit findet beim Gaussian Splatting-Ansatz zunächst nicht statt. Es wäre allerdings denkbar, die Unsicherheit der Rekonstruktion eines Punktes daran zu messen, wie viele Pfade durch diesen hindurchgeführt haben. Je mehr Pfade sich in einem bestimmten Punkt überlappt haben, desto wahrscheinlicher ist die Rekonstruktion an dieser Stelle richtig. Des Weiteren wäre auch eine Konstruktion des Modells denkbar, in der zusätzlich zu den Parametern der Gaussians (bei uns: Cuboids) Unsicherheitswerte mitgelernt werden.

Robustheit (4/5) Der Gaussian Splatting-Ansatz ist im Vergleich zum inversen Rendering etwas weniger anfällig gegenüber Datenartefakten, da nicht direkt die Belastungswerte einzelner Punkte, sondern in Form der Gaussians (bei uns: Cuboids) übergeordnete räumliche Strukturen der Belastungsverteilung gelernt werden, in denen einzelne Abweichungen nicht so stark ins Gewicht fallen sollten. Insbesondere wenn genügend gegensätzliche Evidenz in den Trainingsdaten vorliegt, sollten durch einzelne Abweichungen keine größeren Probleme entstehen.

***A priori*-Fähigkeit (2/5)** Es ist kein Vorwissen nötig, um die Methode zu nutzen und bislang ist keine Nutzung eines Priors vorgesehen. Die Nutzung eines Priors ist allerdings denkbar, indem die bisher im inversen Rendering genutzten Prioren in eine Gaussian- bzw. Cuboid-Struktur mit Durchschnittswerten des jeweiligen Priors umgewandelt würden. Außerdem wäre es möglich, an einzelnen Stellen mit bekannten Belastungswerten, beispielsweise an einer bestehenden Messstation nahe eines Atomkraftwerks, festgelegte Gaussians (bei uns: Cuboids) in das Modell mit aufzunehmen, die in die Berechnungen einfließen aber nicht mittrainiert werden.

Implementierungsaufwand (3/5) Wie auch bei den Neural Radiance Fields und inversem Rendering, muss die Methode Gaussian Splatting zur Nutzung für unseren Anwendungsfall neu implementiert werden. Es kann nicht einfach eine bestehende Implementierung genutzt werden,

da sich die Nutzung zur Bildrekonstruktion im Bereich Computer Vision von der Übertragung auf die Rekonstruktion einer radioaktiv belasteten Wolke in der Raumzeit stark unterscheidet. Allerdings bietet das frei verfügbare Framework PyTorch (Paszke et al., 2019) viele sinnvolle Werkzeuge wie beispielsweise verschiedene Gradientenverfahren und automatische Ableitung, um diese Implementierung vorzunehmen.

Reife (2/5) Wie eingangs beschrieben, fand die Erstveröffentlichung der Methode Gaussian Splatting erst im August 2023 statt (Kerbl, et al., 2023), weshalb es sich um eine sehr neue Methode handelt. Diese hat sich allerdings in kürzester Zeit aufgrund der hohen Qualität ihrer Ergebnisse und dem vergleichsweise niedrigeren Bedarf an Zeit im Bereich der 3D-Rekonstruktion etabliert. Aktuell wird viel weiterführende Forschung an der Methode betrieben.

Bekanntheit (4/5) Die Methode Gaussian Splatting hat innerhalb des letzten Jahres viel Aufmerksamkeit im Bereich Computer Vision erhalten. Während Neural Radiance Fields und Abwandlungen davon zuvor als „State of the Art“ bezüglich der erreichbaren Qualität von 3D-Rekonstruktionen gesehen wurden, wurden diese schnell von Gaussian Splatting überholt. In den zehn Monaten ab der Erstveröffentlichung der Methode im August 2023 sind insgesamt 194 wissenschaftliche Veröffentlichungen zu dem Thema erschienen und der Trend der Nutzung der Methode Gaussian Splatting steigt stark (siehe: https://paperswithcode.com/method/3d-gaussian-splatting, Stand: 02.07.2024).

Anwendungsbreite (3/5) Der Gaussian Splatting-Ansatz wurde bisher im Bereich Computer Vision und dort primär zu Zwecken der 3D-Rekonstruktion von Szenen aus Bilddaten und sogenannter „Novel View Synthesis“ eingesetzt (siehe: https://paperswithcode.com/method/3d-gaussian-splatting, Stand: 02.07.2024). Übertragungen in andere Bereiche sind bisher noch nicht bekannt geworden. Die Übertragung des Ansatzes in andere Bereiche, in denen ebenfalls 3D-Rekonstruktionen benötigt werden, wie in unserem Fall eine Rekonstruktion in der 3D-Raumzeit, ist aus unserer Sicht aber sehr sinnvoll.

2.2.3 Ausgeschlossene Ansätze

In unserer Recherche haben wir uns auch mit Methoden beschäftigt, die wir aufgrund verschiedener Kriterien schon vor der detaillierten Bewertung ausschließen konnten. Im Folgenden beschreiben wir diese Methoden kurz und erläutern unsere Gründe für einen Ausschluss.

2.2.3.1 Gefilterte Rückprojektion

Die gefilterte Rückprojektion ist ein Verfahren aus dem Bereich der Computertomographie. Sie wird für die Rekonstruktion von durchleuchteten Objekten aus deren Projektionen benutzt. Computertomographie ist generell ein Problem, dass unserem Problem sehr ähnlich ist. Entsprechend liegt eine Anwendung von Methoden, die für Computertomographie benutzt werden, sehr nahe.

Abbildung 2.7 zeigt den Kontext der Methode. Ein Objekt wird zunächst von verschiedenen Richtungen aus mit Strahlen durchleuchtet und auf eine Fläche projiziert. Dieser Prozess wird *Radon-Transformation* genannt. Für die Definition der Radon-Transformation ist es wichtig, dass die Strahlen der Projektion geradlinig durch den Raum verlaufen. Alle Projektionen bilden zusammen ein Sinogramm. In der Praxis wird dieses Sinogramm durch ein Messgerät gewonnen. Die gefilterte Rückprojektion ist die inverse Abbildung zur Radon-Transformation und ermittelt aus einem Sinogramm eine Rekonstruktion des durchleuchteten Objektes.

In unserem Problem projizieren die Wege der Menschen die gesuchte Iod-Verteilung jedoch nicht, weil sie keine geraden Linien durch die Raumzeit sind. Damit findet keine Radon-Transformation statt. Entsprechend ist kein Sinogramm gegeben und Verfahren, die ein Sinogramm erfordern, können für unser Problem nicht angewendet werden.

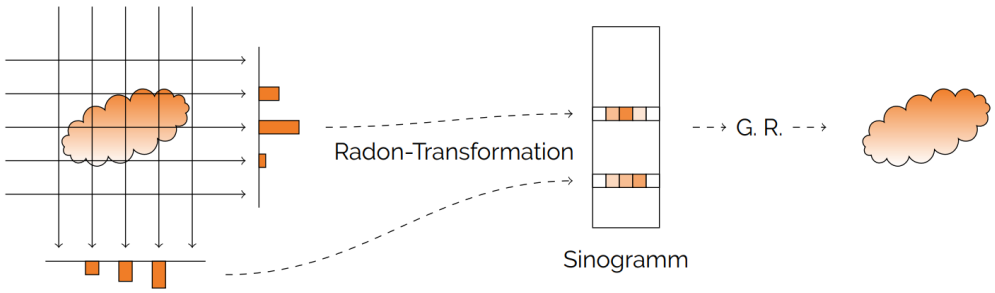

Abbildung 2.7: Durchleuchtung einer Wolke. Entlang der Strahlen wird die Information in der Wolke integriert.

2.2.3.2 Evolutionäre Algorithmen

Evolutionäre Algorithmen sind eine Klasse von Algorithmen zur Optimierung. Ihr Grundgedanke beruht darauf, viele Kandidaten für eine Lösung zu erschaffen und diese nach einem Optimierungskriterium zu vergleichen. Der Speicherbedarf einer konkreten Lösung ist für die gegebenen Ressourcen bereits sehr hoch. Eine Vielzahl von Lösungen vorzuhalten und zu vergleichen, erscheint uns unter den gegebenen Bedingungen nicht umsetzbar. Die Speicherung mehrerer potenzieller Lösungen ist deshalb das Kriterium anhand dessen wir evolutionäre Algorithmen ausschließen.

2.2.3.3 Rekurrente Neuronale Netzwerke

Eine Idee in unserer Recherche war der Einsatz von rekurrenten neuronalen Netzwerken. Diese Netzwerke eignen sich für die Verarbeitung von Sequenzen. Die Wege in unserem Problem sind solche Sequenzen. Eine mögliche Modellierung anhand dieses Ansatzes hat jedoch ergeben, dass rekurrente neuronale Netzwerke in diesem Einsatzgebiet äquivalent zu Neural Radiance Fields sind, zusätzlich aber außerdem noch die Akkumulationsfunktion modellieren. Die Akkumulation muss jedoch nicht gelernt werden, sondern ist bereits bekannt. Rekurrente neuronale Netzwerke sind also Neural Radiance Fields mit zusätzlicher, nicht notwendiger Komplexität. Wir schließen sie deshalb aus.

2.2.3.4 Graph Neural Networks

Graph Neural Networks sind neuronale Netzwerke, die als Daten Graphen verarbeiten. Wir haben Graph Neural Networks betrachtet, weil die Wege in unserem Problem als Graph modelliert werden können. Diese Idee wurde jedoch ausgeschlossen, weil es keine sinnvollen Nachbarschaften in unserem Problem gibt, die über einfache räumlich-temporäre Nachbarschaften hinausgehen. Diese Nachbarschaften können wir jedoch mit den vorgestellten rasterbasierten Ansätzen ausreichend beschreiben. Eine Modellierung mittels Graph Neural Networks würde das Problem also in einer komplizierteren Form beschreiben, bei der wir keinen Mehrwert erkennen können.

2.2.3.5 Generative Adversarial Neural Networks

Generierende Methoden, wie zum Beispiel Generative Adversarial Neural Networks (GANs) (Goodfellow, et al., 2014; Zhu, et al., 2017), sind unter anderem mit der Erzeugung sehr realistischer Bilder bekannt geworden. GANs bestehen aus zwei Teilen: einem Generator und einem Diskriminator. Der Generator erzeugt Daten, die realistisch erscheinen sollen. Der Diskriminator versucht, aus einer Auswahl an echten und erzeugten Daten die erzeugten zu bestimmen. Beide Teile werden gleichzeitig so trainiert, dass sie sich jeweils gegenseitig verbessern, indem sie jeweils versuchen, besser als der andere Teil zu sein. Überträgt man die Idee von GANs auf unser Problem, könnte der Generator dafür verantwortlich sein, aus Wegen eine realistische Iod-Verteilung zu generieren. Der Diskriminator würde dann versuchen, diese synthetisierte Iod-Verteilung von einer gegebenen echten Iod-Verteilung zu unterscheiden. Das Training solcher Ansätze würde jedoch große Mengen an realistischen Szenarien mit Bewegungsdaten benötigen, um denkbar zu sein. Insgesamt sehen wir das Anwendungsgebiet von GANs als zu unpassend für unser Problem an. Wir schließen eine Anwendung zur Erzeugung von Iod-Verteilungen daher aus.

2.3 Zwischenergebnis der Literaturrecherche

Entsprechend der mathematischen Formulierung wurde von uns eine Auswahl an Methoden erstellt, die das vorliegende Rekonstruktions-Problem lösen können. Die Modellierung des Problems als inverses Problem impliziert, dass es nötig sein kann, mit Heuristiken und Optimierungen zu arbeiten. Heuristiken werden zumeist in Kostenfunktionen oder *a priori*-Verteilungen umgesetzt. Zur Optimierung verwenden die Methoden eine Vielzahl verschiedener Optimierungsverfahren, wie zum Beispiel das Gradientenverfahren, das Simplex-Verfahren oder Schätzfunktionen. Wir gehen deshalb davon aus, eine methodisch breite Abdeckung der theoretischen Lösungsmöglichkeiten ermittelt zu haben.

Stochastische robuste Optimierung und bayessche inverse Probleme erreichen in unserer Bewertung beide die gleiche höchste Bewertung. Stochastische robuste Optimierung ist eine Erweiterung der linearen Optimierung, die den dritten Platz einnimmt. Wir schlagen vor, die lineare Optimierung nur als Spezialfall der stochastischen robusten Optimierung mit aufzunehmen, uns aber auf letztere zu konzentrieren.

Zusätzlich zu stochastischer robuster Optimierung und bayesschen inversen Problemen wählen wir für die weitere Testung entsprechend unserer Bewertungen das speicherbasierte inverse Rendering aus. Speicherbasiertes inverses Rendering ist eine Modifikation von Neural Radiance Fields. Wir schätzen es als passender für unser Problem ein. Diese Einschätzung spiegelt sich in den von uns vergebenen Bewertungen wider. Wir schließen Neural Radiance Fields nicht komplett aus, denken jedoch, dass deren Ansatz durch speicherbasiertes inverses Rendering ausreichend abgedeckt ist.

Die Mischung der drei gewählten Methoden stellt nach unserer Einschätzung eine gute Abdeckung aller von uns bewerteten Ansätze dar.

Ergänzend zu Obigem wurde nachträglich auch noch die im August 2023 neu veröffentlichte Methode Gaussian Splatting mit aufgenommen und implementiert, da wir sie bezüglich unseres Anwendungsfalls für einen sehr vielversprechenden Modell-Ansatz halten. Genaueres hierzu findet sich in den jeweiligen Unterkapiteln.

Entsprechend gehen wir mit den vier benannten Methoden in die weitere Evaluierung.

3. Generierung der erforderlichen Daten (AP 2)

In den folgenden Kapiteln beschreiben wir die Implementierung und Benutzung unserer Implementierung der ausgewählten Methoden. In diesem Kapitel fokussieren wir dabei zunächst auf den Teil, der die notwendigen Daten generiert, weil keine Daten eines realen Notfalls vorliegen, die von uns genutzt werden könnten. Die Generierung dieser Daten ist lediglich für die Evaluation der Methoden relevant und findet bei einem realen Notfall keine Anwendung, da in einem solchen Fall mit den während des Notfalls anfallenden Daten gearbeitet werden wird.

Im Rahmen dieses Forschungsvorhabens ist ein Softwaresystem entstanden, das darauf ausgelegt ist, möglichst einfach eine große Anzahl an Experimenten an verschiedenen Methoden und unter verschiedenen Konfigurationen durchzuführen.

Zu diesem Zweck wurde eine Ordnerstruktur entworfen. Diese Ordnerstruktur muss zum Teil vor der ersten Benutzung angelegt werden. Das betrifft die nötigen Eingabedateien, wie zum Beispiel diverse Konfigurationsdateien und auch die verwendeten Verteilungen radioaktiver Stoffe, die den Experimenten zugrunde liegen. Das System kann dann alle weiteren Daten für die festgelegten Experimente automatisch erzeugen. Zu diesen automatisch erzeugten Daten gehören generierte Bewegungsdaten, daraus rekonstruierte Ergebnisse und diverse Visualisierungen. In unseren Experimenten wurden ungefähr 100 GB Daten erzeugt. Im Folgenden beschreiben wir die einzelnen Teile der Ordnerstruktur und die nötigen Schritte, um diese anzulegen.

Zuerst muss ein Ordner angelegt werden, in dem sich alle weiteren Daten befinden werden. Im Folgenden werden wir die Details dieses System beschreiben. Dabei werden wir immer wieder auf diesen Ordner als den *Hauptordner* Bezug nehmen.

3.1 Übersicht

Um eine Methode zu entwickeln, die die Verteilung von Iod in der Umwelt aus Bewegungsprofilen von Personen rekonstruiert, werden ebenjene Bewegungsprofile benötigt. Ein Datensatz dieser Profile liegt jedoch nicht vor. Eine Lösung dieses Problems ist die Verwendung von synthetischen Daten. Deshalb haben wir im Rahmen des zweiten Arbeitspaketes einen Generator für Bewegungsprofile erstellt und implementiert.

Wir fassen den Sachverhalt des Generators einmal in einem Satz zusammen: Personen mit verschiedenen Eigenschaften bewegen sich auf einem Straßennetz aus einem Gebiet heraus und kommen dabei mit radioaktivem Iod in Kontakt.

Konkret sind also verschiedene Aspekte zu betrachten: das Straßennetz, die Personen mit ihren individuellen Eigenschaften, das Verhalten der Personen in dem Straßennetz, sowie der Kontakt der Personen mit Iod. Im Folgenden beschreiben wir, wie der Generator diese Sachverhalte abbildet und wie er zu benutzen ist.

3.2 Phasen

Der Generator arbeitet grob in drei Phasen:

1. Generierung eines Straßennetzes.
2. Generierung von Personengruppen und deren Bewegungen auf dem Straßennetz.
3. Berechnung der Iod-Expositionen für die Bewegungsprofile.

Die Ergebnisse der letzten Phase werden die Eingabedaten für die Rekonstruktionsmethode sein.

3.3 Straßennetze

Straßennetze sind die Grundlage für die Bewegung von Personen durch den Raum. Die Personen bewegen sich entlang von Straßen in einem Straßennetz. Je nachdem, welche Region in welcher Granularität betrachtet wird, kann ein Straßennetz eine sehr unterschiedliche Struktur aufweisen. So besteht ein Autobahnnetz aus wenigen langen Straßen, die sich in einzelnen Punkten kreuzen, wogegen ein Straßennetz einer Stadt aus vielen kleinen Straßen mit sehr vielen Kreuzungen besteht. Die Form des Straßennetzes ist damit relevant für die Bewegung der Personen in diesem Straßennetz. In diesem Kapitel beschreiben wir, wie wir Straßennetze in unseren Experimenten benutzen.

3.3.1 Modellierung

Wir stellen Straßennetze als mathematische Graphen dar, also Knoten, die teilweise mit Kanten verbunden sind. Knoten sind Kreuzungen und Kanten sind die Straßen dazwischen.

Ein Straßennetz wird immer für ein bestimmtes rechteckiges Gebiet generiert.

Alle Knoten liegen innerhalb dieses Gebietes. Ein Teil der Knoten wird gleich verteilt zufällig in dem Gebiet generiert. Die restlichen Knoten werden in Gaußverteilungen generiert. Diese Gaußverteilungen sind in ihrer Anzahl und Größe konfigurierbar. Dadurch können Städte und Ortschaften modelliert werden. Eine zusätzliche Menge von besonderen Knoten repräsentieren Ausfahren zu den Notfallstationen. Diese Knoten liegen am Rand des Gebietes und tragen zusätzlich noch die Information, wie lange eine Person nach der Ausfahrt noch weiterfahren muss, um tatsächlich an einer Notfallstation anzukommen.

Jeder Knoten ist mit anderen benachbarten Knoten durch Kanten verbunden. Die Anzahl der verbindenden Kanten pro Knoten ist konfigurierbar. Dadurch können sowohl baumartige Strukturen mit geringer Vernetzung als auch stark vernetzte Strukturen erschaffen werden.

3.3.2 Warnzonen

Zu einem Straßennetz gehören zusätzlich noch Warnzonen. Eine Warnzone ist ein Gebiet, in dem die Bewegung der Personen eingeschränkt werden kann. Sie sind außerdem zeitlich begrenzt.

Wir unterscheiden drei verschiedene Warnzonen:

1. die Warnzone, in der die Einnahme von Iodtabletten empfohlen wird,
2. die Warnzone, in der die Evakuierung der Personen empfohlen wird,
3. die Warnzone, in der der Verbleib in Gebäuden empfohlen wird.

In unserem Generator wirken sich diese Warnzonen auf das Verhalten der Personen aus. Außerdem versuchen Personen auf ihrem Weg zu einer Notfallstation die Warnzonen zu vermeiden. Dafür gehen sie immer vom aktuellen Zustand der Warnzone zum jeweiligen Zeitpunkt aus, können also nicht wissen, ob eine Warnzone in der Zukunft gültig oder ungültig wird.

Eine Warnzone kann entweder als ein Rechteck oder über ein Shapefile angegeben werden. In dem Shapefile können mehrere Flächen angegeben sein. Für jede Fläche muss ein Attribut mit dem Namen “cm” existieren, dass einen der Werte 1 (Iodtabletten), 2 (Aufenthalt in Gebäuden) oder 3 (Evakuierung) hat. Die Flächen werden dann entsprechend dieser Werte geladen und zu Warnzonen zusammengefügt.

3.3.3 Benutzung

Graphen müssen zu Beginn manuell erzeugt werden. Dafür wird folgender Befehl verwendet:

```
python –m reconstruction.generate.graph config.json
```

Der Pfad der Datei config.json kann nach Bedarf angepasst werden. Das Programm erzeugt als Ausgabe eine Graph-Datei mit dem Namen output.graph.json. Diese kann nun wiederum mit dem folgenden Befehl visualisiert werden:

```
python –m reconstruction.visualization.graph output.graph.json
```

Dieser Befehl erzeugt eine Bilddatei, sowie mehrere GeoJSON-Dateien, die mit einem GIS-Programm benutzt werden können.

3.4 Szenarien

Ein Szenario beschreibt die räumlich-zeitliche Verteilung eines Wertes, meistens die Iod-Konzentration, in der Umwelt. In diesem Kapitel beschreiben wir, wie wir Szenarien modellieren und verarbeiten.

3.4.1 Modellierung

Szenarien sind beschrieben durch ein quadratisches Gebiet, das gleichmäßig in gleich große quadratische Zellen gerastert ist, sowie aus einer Reihe von stündlichen Zeitschritten. Daraus ergibt sich als Datenstruktur ein dreidimensionales Array, in dem jeder Eintrag die Iod-Konzentration in einer bestimmten Ortszelle zu einer bestimmten Stunde anzeigt. Wir nennen diese Einträge der Einfachheit halber auch Raumzeitpunkte.

3.4.2 Dateien

Wir gehen davon aus, dass ein Szenario ursprünglich als eine Menge von CSV-Dateien gegeben ist, die von dem System RODOS des BfS erstellt werden. Jede dieser Dateien beschreibt die Ausbreitung eines Iod-Isotops.

Die gegebenen Szenarien sind im Koordinatenreferenzsystem (CRS) EPSG:4326 gegeben. Um besser mit den Daten arbeiten zu können, müssen diese zunächst nach EPSG:25832 konvertiert werden. Dafür kann der folgende Befehl genutzt werden:

```
python –m reconstruction.scenario.convert <Pfad zu Szenario-CSV>
```

Als Szenario-CSV-Datei wird der Pfad der zu konvertierenden Szenario-Datei angegeben. Als Ausgabe wird im gleichen Ordner eine Datei erzeugt, die zusätzlich den Suffix "_25832" trägt, also beispielsweise I131_25832.csv.

Sobald die Dateien im korrekten Format vorliegen, muss eine JSON-Datei mit dem Namen scenario.json angelegt werden, die für jedes Isotop den Dateipfad der entsprechenden CSV-Datei im korrekten CRS enthält. Ein Beispiel wäre:

```
{
    "I131": "I131.csv",
    "I132": "I132.csv",
    "I133": "I133.csv"
}
```

Um die interne Handhabung des Szenarios zu verbessern, muss das Szenario danach einmal in ein neues Dateiformat übertragen werden. Der Befehl zum Starten des Skriptes lautet:

```
python -m reconstruction.scenario <Pfad zu scenario.json>
```

Dieses Skript erzeugt eine Datei save.json, die die Metadaten des Szenarios enthält, sowie eine Datei save_array.npy, die die Werte der Raumzeitpunkte als numpy-Array enthält. Nur diese beiden Dateien werden vom Experimentsystem für alle weiteren Vorgänge benutzt.

Um diese Dateien für das Experimentiersystem nutzbar zu machen, müssen diese im Hauptordner unter dem `Pfad scenarios/<Name des Szenarios>` abgelegt werden. Der Name des Szenarios kann frei gewählt werden und muss danach in der Interaktion mit der Software verwendet werden, um das Szenario zu referenzieren.

3.4.3 Weitere Benutzung

Die Rekonstruktionen der von uns ermittelten Methoden werden ebenfalls als Szenarien gespeichert.

Da Szenarien die Verteilung von Information in der Raumzeit darstellen können, benutzen wir in unseren Experimenten auch eine Reihe weiterer Szenarien, die keine Verteilungen von radioaktiven Stoffen darstellen. Zum einen sind das Heatmaps, die die Anzahl von Wegen an einem bestimmten Raumzeitpunkt messen, und zum anderen berechnete Szenarien, wie das Differenzszenario aus einer Rekonstruktion und einer originalen Verteilung.

3.4.4 Skalierung der Szenarien auf andere Auflösungen

Wir betrachten in unseren Experimenten Szenarien in verschiedenen räumlichen Auflösungen. Das Originalszenario ist in einer hohen Auflösung gegeben, in der sich eine Rekonstruktion schnell als nicht sinnvoll herausgestellt hat. Das Experimentsystem ist deshalb in der Lage, die gegebenen Szenarien in die jeweils benötigte niedrigere räumliche Auflösung umzurechnen. Das geschieht während des normalen Durchlaufs automatisch.

3.4.5 Anpassung der Prioren

Wir benutzen in manchen Experimenten Szenarien, die dem Originalszenario ähnlich sind, als Prioren. In unseren Experimenten können das beispielsweise Simulationen des gleichen Notfalls mit leicht unterschiedlichen Wetterdaten sein. Um ein Szenario als Prior für ein Experiment benutzen zu können, müssen der Prior und das Originalszenario jedoch in der gleichen Auflösung den gleichen Raum abdecken. Deshalb kann das Experimentsystem das Priorszenario automatisch auf das Format des Originalszenarios umrechnen. Dies geschieht in zwei Schritten: zunächst wird mit der originalen Auflösung des Originalszenarios das Priorszenario abgetastet und der jeweilige Wert übertragen. Als Ergebnis entsteht dadurch ein Szenario, das die originale Auflösung der Originalszenarios und die Werte des Priorszenarios hat. Im zweiten Schritt wird dieses Szenario dann auf die aktuell verwendete Auflösung des Originalszenarios skaliert. Es hat sich gezeigt, dass das so umgerechnete Priorszenario von höherer Qualität ist, wenn die Schritte in dieser Reihenfolge durchgeführt werden.

3.4.6 Anpassung der Straßennetze an Szenarien

Um ein Straßennetz mit verschiedenen Szenarien testen zu können, müssen wir das Straßennetz auf das Gebiet des Szenarios übertragen können. In unseren Daten decken sowohl die Szenarien

als auch die Straßennetze immer ein quadratisches Gebiet ab, sodass die Übertragung mit einer Verschiebung und Skalierung gut umzusetzen ist.

3.5 Bewegungsprofile

Ein Bewegungsprofil besteht aus den Bewegungsdaten einer Person. Dazu gehören neben zeitlich aufgelösten Ortsangaben auch die Altersklasse der Person sowie weitere zu den einzelnen Zeitpunkten gehörige Informationen, wie z.B. das Tragen einer Atemmaske, oder Angaben zum Aufenthalt in Gebäuden. Im Folgenden beschreiben wir zwei verschiedene Arten von Bewegungsprofilen und wie wir diese in unserer Software generieren und verwenden. Das Wort Pfad wird in dieser Arbeit synonym für das Wort Bewegungsprofil verwendet.

3.5.1 Modellierung

Wir betrachten zwei verschiedene Modellierungen von Bewegungsprofilen, wobei die Bewegungsprofile der zweiten Art aus denen der ersten Art berechnet werden:

1. Kontinuierliche Bewegungsprofile, die kontinuierliche Koordinaten haben und nur Bewegungen beschreiben, und
2. Diskrete Bewegungsprofile, die sich in Schritten zwischen gerasterten Raumzeitpunkten bewegen, deren Koordinaten als Array-Indizes innerhalb eines Szenarios gegeben sind und die bereits Informationen zu Iod-Expositionen beinhalten.

Der Generator erzeugt zunächst kontinuierliche Bewegungsprofile auf dem Straßennetz. Aus diesen kontinuierlichen Bewegungsprofilen werden dann für das jeweilige Szenario diskrete Bewegungsprofile berechnet. Diese Teilung in unterschiedliche Schritte ermöglicht es beispielsweise auf demselben Szenario dieselben Pfade für unterschiedliche Auflösungen zu benutzen.

3.5.2 Konfiguration

Die Konfiguration des Pfadgenerator geschieht ebenfalls über die allgemeine Konfigurationsdatei des Generators. Details dazu sind in Kapitel 3.6 beschrieben.

3.5.3 Generierung der kontinuierlichen Pfade

Zunächst werden auf dem Straßennetz kontinuierliche Pfade generiert, die eine möglichst realistische Betrachtung der Bewegungen von Personen abbilden.

Die Generierung der Wege geschieht in Gruppen von Personen. Eine solche Gruppe kann beispielsweise eine Familie darstellen. Die Verteilung der Gruppengrößen kann konfiguriert werden. Die erste Person in jeder Gruppe ist ein Erwachsener. Für jede weitere Person wird eine Altersklasse zufällig nach ebenfalls konfigurierbaren Gewichtungen gewählt. Alle Personen einer Gruppe bewegen sich gemeinsam auf dem gleichen Weg. Sie unterscheiden sich jedoch darin, ob sie Atemmasken tragen oder Iodtabletten einnehmen.

Jeder Weg beginnt zur festgelegten Startzeit an einer zufälligen Position auf einer zufälligen Kante des Straßennetzes. Von dort aus wird sich die Person auf den Kanten des Straßennetzes in Richtung einer Notfallstation begeben. Die Bewegung findet in Segmenten statt. Die Länge eines Segmentes kann konfiguriert werden. Zu Beginn jedes Segmentes werden verschiedene Entscheidungen getroffen, die wir im Folgenden behandeln. Innerhalb des Segmentes werden diese Entscheidungen beibehalten und umgesetzt. Die Entscheidungen sind:

1. Ob (einmalig) eine Iodtablette eingenommen wird.
2. Ob das Tragen einer Atemmaske begonnen (und dann beibehalten) wird.
3. Ob der Aufenthalt des vorherigen Segmentes beibehalten wird.
4. Falls nicht, ob ein Aufenthalt im Gebäude oder im Freien stattfindet.
5. Falls ein Aufenthalt im Gebäude stattfindet, ob insbesondere ein Aufenthalt im Keller stattfindet.
6. Falls ein Aufenthalt im Freien stattfindet, ob eine Bewegung stattfindet.
7. Falls eine Bewegung stattfindet, ob sie zufällig oder auf eine Notfallstation gerichtet erfolgt.
8. Mit welcher Geschwindigkeit die Bewegung stattfindet.
9. Ob bei der Bewegung Warnzonen vermieden werden.
10. Die Wahrscheinlichkeit für einen Aufenthalt im Freien ist um 12 Uhr mittags am höchsten und um Mitternacht am geringsten.

Wenn eine Person am Ende ihres Pfades eine Ausfahrt zu einer Notfallstation nimmt, wird die zeitliche Distanz zwischen der Ausfahrt und der Notfallstation ebenfalls in dem Pfad der Person abgespeichert.

3.5.4 Generierung der diskreten Pfade

Die kontinuierlichen Pfade werden jeweils einzeln in diskrete Pfade umgerechnet. Im Folgenden beschreiben wir die dafür notwendigen Schritte.

3.5.4.1 Rasterung

Der erste Schritt ist die Rasterung, bei der die Koordinaten eines kontinuierlichen Pfades entsprechend der Auflösung eines Szenarios in den räumlichen Dimensionen und in Stundenschritten in der zeitlichen Dimension gerastert werden. Da wir alle Pfade zur Berechnung oder Evaluation einer Rekonstruktion benutzen, entspricht die Auflösung für die Rasterung der Pfade der gewünschten Auflösung der Rekonstruktion.

Das Ergebnis dieser Rasterung ist eine Liste von zweidimensionalen Rasterindizes, die die jeweilige räumliche Zelle beschreiben, in der sich die Person aufgehalten hat. Die zeitliche Rasterung ist implizit gegeben, indem jeder Eintrag in der Liste einen Abstand von einer Stunde zu dem vorherigen Eintrag hat und die Startzeit bekannt ist.

3.5.4.2 Bestimmung der Schutzfaktoren

Der Schutzfaktor einer Person zu einem Zeitpunkt ist abhängig von verschiedenen Einzelfaktoren, wie in Kapitel 1 bereits dargelegt wurde. Die verschiedenen Werte sind über Dateien konfigurierbar. Die Konfigurationsdateien liegen alle in dem Hauptordner unter dem Pfad protection. Vor allem befindet sich darin eine Datei protection/protection.json, die alle Pfade für die Konfigurationsdateien beinhaltet. Sie hat standardmäßig diesen Inhalt:

```
{
  "protective_mask": "atemmaske.json",
  "breathing_rate": "atemrate.json",
  "residence": "residenz.json",
  "iodine": {
    "A1": "iodtablette_A1.csv",
    "A2": "iodtablette_A2.csv",
```

```
    "A3": "iodtablette_A3.csv",
    "A4": "iodtablette_A4.csv",
    "A5": "iodtablette_A5.csv",
    "A6": "iodtablette_A6.csv"
  },
  "retention": {
    "A1": "Retention Auswahl_A1.csv",
    "A2": "Retention Auswahl_A2.csv",
    "A3": "Retention Auswahl_A3.csv",
    "A4": "Retention Auswahl_A4.csv",
    "A5": "Retention Auswahl_A5.csv",
    "A6": "Retention Auswahl_A6.csv"
  }
}
```

Die Datei für "protective_mask" beinhaltet eine Zahl, die den Schutzfaktor darstellt, wenn eine Atemmaske getragen wird.

Die Datei für "breathing_rate" beinhaltet Atemraten für alle Altersgruppen.

Die Datei für "residence" beinhaltet die Schutzfaktoren für den Aufenthalt an verschiedenen Orten.

Die Wertverläufe für Iodtabletten und Retentionskurven sind pro Altersgruppe durch eine einzelne CSV-Datei konfigurierbar. Sie enthalten jeweils eine Tabelle mit zwei Spalten. Die erste Spalte beinhaltet den zeitlichen Abstand, die zweite den dazugehörigen Wert. Für Iodtabletten sind die Werte in Prozent angegeben, für die Retentionen in Werten zwischen 0 und 1.

Bei der Berechnung der zeitabhängigen Schutzfaktoren wird jeweils der zeitliche Abstand zwischen der Ausfahrt und der Notfallstation am Ende eines Pfades mitberechnet. Die Werte der Schutzfaktoren werden zuletzt pro Pfad parallel zu den Raster-Indizes ebenfalls in einer Liste abgespeichert.

3.5.5 Berechnung der Exposition

Die Exposition einer Person am Ende des Pfades kann nun mithilfe der Indizes, der Schutzfaktoren und des bereits in Abschnitt 3.4.2 konvertierten Szenarios entsprechend der in Kapitel 1 beschriebenen Gleichung berechnet werden. Dafür werden die Belastungswerte des Szenarios an den Positionen der Indizes mit den Schutzfaktoren multipliziert und anschließend summiert. Die daraus resultierende Exposition wird dann ebenfalls in dem Bewegungsprofil gespeichert.

3.5.6 Verfügbarkeit von Bewegungsprofilen

In unseren Experimenten verwenden wir generierte Bewegungsprofile, weil keine Daten aus realen Notfällen vorliegen. Bei einem Notfall werden die Bewegungsprofile in den Notfallstationen erhoben. Für diese Bewegungsprofile werden auch die Expositionen der Personen gemessen. Diese erhobenen Bewegungsprofile können dann als Trainingsdaten für Modelle verwendet werden. Ein Teil dieser Bewegungsprofile kann aber auch als Evaluations- oder Testdaten verwendet werden, um die Qualität einer trainierten Methode zu evaluieren. Manche Methoden verhindern so beispielsweise ein Overfitting (Details dazu folgen in Kapitel 4). Darüber hinaus kann es auch Bewegungsprofile von Personen geben, deren Exposition nicht in einer

Notfallstation gemessen wurde. Die Daten dieser Personen können lediglich dazu verwendet werden, das Modell auf sie anzuwenden, um eine Exposition abzuschätzen.

3.6 Konfigurationsdatei

Es gibt eine allgemeine Konfigurationsdatei für den Generator. In diesem Abschnitt erklären wir alle möglichen Optionen, die diese Datei bietet. Wir betrachten folgendes Beispiel (Tabelle 3.1):

Tabelle 3.1: Beispielhafte Konfigurationsdatei mit Erklärungen der Elemente

Konfigurationsdatei	**Beschreibung**
{	
„general“: {	Allgemeine Einstellungen
„crs”: “EPSG:4326”,	Das Koordinatenreferenzsystem, das in dieser Datei benutzt wird.

„area“: {
„x“: {
„min“: 8.9,
„max“: 9.4
},
„y“: {
„min“: 50.0,
„max“: 50.5
}
},
	Begrenzungen des betrachteten rechteckigen Gebietes.

„time“: {
„start“: „2020-12-19 05:00“,
„end“: „2020-12-20 22:00“
}
},
	Betrachteter Zeitraum.
	Die Zeiten sind im ISO 8601-Format als String gegeben.

„graph“: {	Einstellungen zur Graph-Generierung
„node“: {	Einstellungen zur Generierung von Knoten.
„load“: false,	Gibt an, ob ein Straßennetz von OpenStreetMap geladen werden soll.
„count“: 100,	Anzahl der Knoten als positive ganze Zahl
„connectivity“: { “1”: 2, “2”: 1 },	Gewichtungen für die zufällig gewählte Anzahl der nächstgelegenen Knoten im Graph, mit denen ein neu hinzugefügter Knoten verbunden wird.
„uniformly_distributed“: 0.1,	Anteil der Knoten, die gleichverteilt in der Fläche generiert werden.
„cluster“: [0.04, 0.04, 0.04] },	Liste von Standardabweichungen. Für jede Standardabweichung wird eine zufällige Region erzeugt. Wird ein neuer Knoten erzeugt und wird dieser nicht gleichverteilt generiert (siehe „uniformly_distributed“), wird zunächst eine Region nach den Standardabweichungen gewichtet gewählt. Danach wird innerhalb dieser Region mithilfe der Standardabweichung der neue Knoten platziert.

„emergency_centers“: {	Einstellungen zu Notfallstationen
„count“: 10,	Anzahl der Notfallstationen
„max_distance“: 60 },	Maximale zusätzliche Entfernung der Notfallstation von der Ausfahrt in Minuten.

„warning_zones“: {
„iodine“: {
„area“: {
„x“: {
„min“: 9.0,
„max“: 9.3
},
„y“: {
„min“: 50.1,
„max“: 50.4
}
},
„time“: {
„start“: „2020-12-19 07:30“,
„end“: „2020-12-19 22:00“
}
},
„stay_inside“: {
„area“: „path/file.shp“,
„time“: „generate“
},
„evacuation“: „none“
}
},
	Es sollen verschiedene Angaben zu den drei Warngebieten gemacht werden können. Verschiedene Angaben sind möglich:
	
	1. Ein Warngebiet kann mit einer räumlichen und zeitlichen Ausdehnung angegeben werden.
	2. Anstelle der räumlichen Ausdehnung kann ein Shapefile angegeben werden.
	3. Anstelle der räumlichen oder zeitlichen Ausdehnung kann „generate“ angegeben werden. Entsprechend der Hierarchie der Warngebiete werden Warngebiete ineinander gleichverteilt generiert.
	4. Ein Warngebiet kann nicht existieren und wird dann mit „none“ angegeben.
	
	Die Optionen der Punkte 1 bis 4 sind für „area“ und „time“ frei wählbar und können beliebig kombiniert werden.

„path“: {	Einstellungen für Pfade
„count“: 1000,	Anzahl der Pfade
„batch_size“: 50,	Die Batchsize ist ein technischer Parameter, der angibt, wie viele Pfade innerhalb eines Schrittes generiert werden. Eine höhere Anzahl führt zu einer schnelleren Generierung von Pfaden. Je nach benutztem System gibt es jedoch eine Obergrenze, bis zu der der Generator stabil funktioniert. Für uns hat sich 50 als ein guter Wert herausgestellt.
„greed“: 0.0,	Gewichtung zwischen zwei Bewertungen für die Berechnung der kürzesten Pfade: kürzester bisher zurückgelegter Weg (wenig greed) und kürzeste Luftlinie zum Ziel (viel greed). 0.0 ist mathematisch robuster, aber sehr viel langsamer zu berechnen. 1.0 ist schneller zu berechnen, kann aber einen längeren Weg wählen.

„group_sizes“: { „1“: 200, „2“: 10, „3“: 5, „4“: 1, „5“: 1 },	Auflistung von möglichen Gruppengrößen mit ihrer jeweiligen Gewichtung. Sowohl die Größen als auch die Gewichtungen können frei gewählt werden. Die Größen sind (aufgrund des JSON-Formats) als positive ganze Zahlen in Form eines Strings anzugeben.
„age_group_weights“: { „A1“: 1.2, „A2“: 1.2, „A3“: 6.1, „A4“: 6.1, „A5“: 6.1, „A6“: 79.3 },	Auflistung der Altersklassen mit Gewichtungen für die Generierung der Altersklassen. Die Altersklassen sind fest vorgeschrieben. Die Gewichte können beliebig angepasst werden.
„protective_mask_probability“: 0.1,	Wahrscheinlichkeit dafür, dass begonnen wird an einem Segment eine Atemschutzmaske zu tragen.

„iodine_probability“: {
„in_warning_zone“: {
„iodine“: {
„A1“: 0.2,
„A2“: 0.2,
„A3“: 0.2,
„A4“: 0.2,
„A5“: 0.2,
„A6“: 0.02
},
„stay_inside“: {
„A1“: 0.2,
„A2“: 0.2,
„A3“: 0.2,
„A4“: 0.2,
„A5“: 0.2,
„A6“: 0.02
},
„evacuation“: {
„A1“: 0.4,
„A2“: 0.4,
„A3“: 0.4,
„A4“: 0.4,
„A5“: 0.4,
„A6“: 0.05
}
},
„outside_of_warning_zone“: {
„A1“: 0.02,
„A2“: 0.02.
„A3“: 0.02,
„A4“: 0.02,
„A5“: 0.02,
„A6“: 0.01
}
},
	Wahrscheinlichkeiten dafür, dass eine Person an einem Segment anfängt Iodtabletten einzunehmen. Die Wahrscheinlichkeiten sind unterteilt nach dem Aufenthalt in einem Warngebiet und der Altersklasse, der die Person angehört. Beginnt eine Person an einem Punkt ihres Weges mit der Einnahme von Iodtabletten bleibt diese Einnahme für den Rest des Weges bestehen.

„exploration_rate“: 0.1,	Wahrscheinlichkeit, dass eine Entscheidung der Bewegungsrichtung zufällig passiert. Solche Entscheidungen werden zu Beginn eines Segments und an jedem Knoten getroffen. Ist die Bewegung nicht zufällig, wird der kürzeste Weg zur nächsten Notfallstation gewählt.
„segment_length_in_minutes“: 30,	Maximale Länge eines Segmentes in Minuten. Die tatsächliche Länge wird gleichverteilt in diesem Zeitraum gewählt. Bei einer Länge von 30 min sind die Segmente im Durchschnitt somit 15 min lang.
„speed“: {	Einstellungen zur Bewegungsgeschwindigkeit
„max“: 130,	Maximale Geschwindigkeit in km/h.
„momentum“: 0.5,	Trägheit bei Geschwindigkeitsänderungen. Bei einer Änderung der Geschwindigkeit wird ein gewichteter Mittelwert aus der alten Geschwindigkeit und der neuen Geschwindigkeit berechnet. Dieser Parameter gibt den Anteil der alten Geschwindigkeit an.
„outside_but_not_moving“: 0.1, },	Wahrscheinlichkeit dafür, dass sich eine Person nicht bewegt, obwohl sie draußen ist.
„residence“: {	Einstellungen zur Residenz
„continue“: 0.1,	Wahrscheinlichkeit, dass eine Person ihre Residenz aus dem letzten Segment beibehält.

„outside_at_midday“: {
„in_warning_zone“: {
„iodine“: 0.2,
„stay_inside“: 0.1,
„evacuation“: 0.8
},
„outside_warning_zone“: 0.5
},
	Die Wahrscheinlichkeit, dass eine Person sich draußen aufhält ist mit einer Glockenkurve in Abhängigkeit von der Uhrzeit modelliert. Die maximale Wahrscheinlichkeit ist 12 Uhr mittags erreicht.
	Die Wahrscheinlichkeit wird nach dem eventuellen Aufenthalt in Warnzonen aufgeschlüsselt.

„basement_if_inside“: 0.1 },	Wahrscheinlichkeit, dass sich eine Person in einem Keller befindet, wenn bereits ermittelt wurde, dass sie sich nicht im Freien aufhält.
„avoidance“: { „iodine“: 0.7, „stay_inside“: 0.9, „evacuation“: 0.99 },	Wahrscheinlichkeiten dafür, dass sich einer Person nicht in ein Warngebiet bewegt, unterteilt nach den drei Warngebieten.

4. Entwicklung eines Verfahrens des maschinellen Lernens (AP 3)

In diesem Kapitel wird die Entwicklung eines Verfahrens des maschinellen Lernens für die in Kapitel 1 beschriebene Problemstellung dargestellt. Dafür beschreiben zur zunächst das von uns zu diesem Zweck entwickelte Experimentiersystem in seinem Aufbau und seiner Konfiguration (Abschnitt 4.1). Im darauffolgenden Kapitel beschreiben wir die verwendeten Metriken, mit denen wir die Ergebnisse der Experimente bewertet haben (Abschnitt 4.2). Danach folgen die Implementierungen der einzelnen Methoden (Abschnitt 4.3). Im Anschluss werden die Ergebnisse der Experimente mit den beiden vollständig evaluierten Methoden *Inverse Rendering* und *Cuboid Splatting* dargestellt (Abschnitt 4.4) und danach verschiedene Dimensionen der Ergebnisse genauer beleuchtet (Abschnitt 4.5).

4.1 Entwicklung eines Experimentiersystems

Das entwickelte Experimentiersystem ermöglicht es, eine große Zahl an unterschiedlichen Methoden mit unterschiedlichen Eingabedaten und Parametrisierungen vollautomatisch auszuführen und zu evaluieren. In diesem Kapitel beschreiben wir, welche Einstellungen man vornehmen kann und welche wir benutzt haben, auf welcher Hardware wir die Experimente ausgeführt haben und welche Metriken in dem System verwendet werden, um die Ergebnisse zu bewerten.

4.1.1 Aufbau der Experimente

4.1.1.1 Straßennetze

Wir verwenden für unsere Experimente drei verschiedene Straßennetze. Bei einem Straßennetz betrachten wir zusätzlich noch eine weitere Version, in der Warnzonen eingefügt sind. Die Straßennetze wurden so gewählt, dass verschiedene Bewegungsmuster abgebildet werden können.

Das erste Straßennetz nennen wir *Hauptstraßen*. Es besteht aus einigen wenigen Straßen mit wenigen Kreuzungen. Man kann es sich wie ein Netz von Autobahnen vorstellen.

Das zweite Straßennetz nennen wir *Ortschaften*. Es besteht aus einigen Ballungsgebieten und wenigen Straßen dazwischen. Man kann es sich wie eine ländliche Gegend mit kleinen Ortschaften und Landstraßen dazwischen vorstellen.

Das dritte Straßennetz nennen wir *Platz*. Es besteht aus vielen sehr eng vernetzten Straßen und ermöglicht eine nahezu freie Bewegung in der Fläche. Man kann es sich wie einen frei begehbaren Platz oder eine Stadt vorstellen.

Für das Straßennetz *Ortschaften* haben wir noch eine zweite Version mit dem Namen *Ortschaften (Warnzonen)* benutzt, die die gleichen Straßen beinhaltet, aber zusätzlich auch noch vom BfS für das Szenario Emsland gegebene Warnzonen im mittleren Bereich des Gebietes, in der Nähe des Kernkraftwerkes, beinhaltet.

4.1.1.2 Szenarien

Wir betrachten zwei realistische Szenarien, die den Daten des BfS entstammen und mit dem Entscheidungshilfe- und Prognosemodell RODOS berechnet wurden, und zwei synthetische

Szenarien, die wir zusätzlich erzeugt haben, um die Methoden unter Extrembedingungen zu untersuchen.

Das erste realistische Szenario betrachtet das Emsland. Wir haben es in zwei Varianten gegeben. Sie heißen *Emsland* und *EuRim-Emsland*. Wir benutzen *Emsland* als ein Originalszenario, das wir rekonstruieren wollen, und *EuRim-Emsland* als den dazugehörigen Prior.

Das zweite realistische Szenario betrachten das Gebiet rund um das Kernkraftwerk in Gundremmingen. Wir haben drei Varianten dieses Szenarios gegeben, von denen wir die Variante *Gundremmingen* als Originalszenario und die zwei Varianten *Gundremmingen_P1* und *Gundremmingen_P2* als zwei verschiedene Prioren verwenden. Gundremmingen ist vor allem für die weitere Evaluation in AP 4 vorgesehen und deshalb in AP 3 nur teilweise enthalten.

Das erste von uns generierte Szenario nennen wir *Box*. Dabei handelt es sich um eine quaderförmige Ausbreitung, die über den kompletten Zeitraum in einem Viertel des Gebietes einen konstanten Wert hat. Das Szenario ist damit in einem großen Bereich konstant, hat aber an den Rändern des Viertels einen sehr starken Gradienten von dem konstanten Wert auf null.

Das zweite Szenario nennen wir *Gaussian*. Es ist eine dreidimensionale gaußförmige Verteilung, die ihren Mittelpunkt in der Mitte des Gebietes zum mittleren Zeitpunkt hat und von dort ausgehend sowohl in zeitliche als auch räumliche Richtungen normalverteilt abnimmt. Dadurch ist *Gaussian* ein sehr weich verlaufendes Szenario und unser Gegenentwurf zum *Box*-Szenario.

Auf der Zeitachse benutzen wir für die Szenarien *Box* und *Gaussian* die gleichen 116 Zeitschritte wie das Szenario *Emsland*. Außerdem benutzen die beiden synthetischen Szenarien das gleiche Gebiet wie das Emsland. Deshalb ist es uns möglich, die gleichen Bewegungsprofile für alle Szenarien zu verwenden.

In Tabelle 4.1 sind die Größen der Zellen in räumlicher und zeitlicher Dimension für einige Kombinationen von Szenarien und Auflösungen beispielhaft dargestellt. Wir haben die zeitliche Auflösung von einer Stunde stets beibehalten.

Tabelle 4.1: Auflistung der Größen der Zellen der Szenarios Emsland und Gundremmingen in Abhängigkeit verschiedener Auflösungen

Szenario	**Auflösung**	**Räumliche Größe der Zellen**	**Zeitliche Größe der Zellen**
Emsland	10 x10x116	38,4 km	1 Stunde
Emsland	30x30x116	12,8 km	1 Stunde
Emsland	100x100x116	3,84 km	1 Stunde
Emsland	768x768x116 (original)	0,5 km	1 Stunde
Gundremmingen	10x10x30	40 km	1 Stunde
Gundremmingen	30x30x30	13,3 km	1 Stunde
Gundremmingen	100x100x30	4 km	1 Stunde
Gundremmingen	800x800x30 (original)	0,5 km	1 Stunde

4.1.1.3 Kreuzprodukt der Experimentparameter

Zunächst wollen wir die Szenarien systematisch über einem Großteil des kartesischen Produkts aus verschiedenen Hyperparametern evaluieren. Wir betrachten:

- die Raumauflösungen 10x10, 30x30 und 100x100,
- 5.000, 10.000 und 25.000 Bewegungsprofilen und
- die Graphen Hauptstraßen, Ortschaften und Platz.

Die Anzahl der Bewegungsprofile bezieht sich auf die Gesamtzahl der für die Rekonstruktion zur Verfügung stehenden Bewegungsprofile. Diese werden für die Methode des inversen Renderings aufgeteilt in Trainingsdaten und Validierungsdaten. Etwa 90% der Bewegungsprofile sind Trainingsdaten, die zur Berechnung der Rekonstruktion benutzt werden. Die restlichen 10% der Bewegungsprofile sind die Evaluationsdaten, die während des Trainings benutzt werden, um aus den Zwischenergebnissen der einzelnen Epochen die beste auszuwählen. Unabhängig von diesen Daten evaluieren wir das Endergebnis der Rekonstruktion jeweils noch auf einem Testdatensatz von 1000 Bewegungsprofilen für *Inverse Rendering* und 5000 Bewegungsprofilen für *Cuboid Splatting*. Diese Testdaten zählen nicht zur Anzahl der Bewegungsprofile dazu, weil sie keine Eingabedaten für die Rekonstruktion sind und in der realen Anwendung der Methoden nicht benutzt werden würden.

Wir gehen davon aus, alle Bewegungsprofile von Anfang an für das Training zur Verfügung zu haben. Es ist theoretisch möglich, das Training mit einer kleineren Menge an Bewegungsprofilen zu starten und im Verlauf mehr Bewegungsprofile hinzuzufügen. Damit könnte das Training bereits nach Vorliegen einer Teilmenge von Bewegungsprofilen gestartet und dann um weitere erhobene Bewegungsprofile erweitert werden. Diese Strategie ist jedoch nicht implementiert.

Neben der systematischen Analyse des Kreuzproduktes widmen wir uns in Kapitel 4.5 der detaillierten Auswertung entlang einzelner Parameter mit Blick auf deren Auswirkungen auf das Rekonstruktionsergebnis. Um diese Dimensionen jeweils besser ausleuchten zu können, haben wir nach Bedarf weitere Durchläufe mit den notwendigen Parametern durchgeführt. So kommen beispielsweise Durchläufe mit 1000, 50.000 und 100.000 Pfaden hinzu.

4.1.2 Konfiguration der Durchläufe

Sobald ein Szenario und ein Straßennetz vorliegen, können Experimentdurchläufe durchgeführt werden. Wir nennen einen solchen Durchlauf in unserer Software *run*. In der Datei runs/runs.csv werden alle Durchläufe aufgelistet. Dabei ist es zunächst nicht relevant, ob diese Durchläufe bereits gelaufen sind, noch laufen sollen, oder nicht laufen sollen, weil sie als inaktiv markiert sind. Das System führt jeweils genau die Durchläufe aus, die noch nicht gelaufen sind, aber noch laufen sollen. Doppelte Ausführungen werden vermieden. Nur die Evaluationen der aktiven Durchläufe werden immer ausgeführt, unabhängig davon, ob der Durchlauf selbst bereits vorhanden war oder zum ersten Mal ausgeführt wurde. Dadurch ist es möglich, eine große Anzahl von Durchläufen zu verwalten und auch gemeinsam zu starten.

Ein einzelner Durchlauf besteht aus den folgenden Eigenschaften, die jeweils eine Spalte der Tabelle bilden:

1. *id*: eine numerische Identifikationsnummer des Durchlaufs.
2. *active*: die Angabe, ob ein Durchlauf beim Ausführen des Experimentsystems durchgeführt werden soll, oder nicht (true oder false). Diese Einstellung macht es

möglich, dass auch Durchläufe eingetragen werden können, die aktuell nicht betrachtet werden sollen.
3. *graph*: der Name des Graphen, der das Straßennetz abbildet.
4. *scenario*: der Name des Szenarios, das verwendet werden soll. Der Name entspricht dem Namen des Ordners, in dem das konvertierte Szenario gespeichert ist.
5. *prior*: der Name des Priorszenarios. Der Name entspricht dem Namen des Ordners, in dem das konvertierte Szenario gespeichert ist. Falls es keinen Prior gibt, wird das Feld freigelassen.
6. *resolution*: die Auflösung, die für das Szenario gewählt werden soll als Zahl der Länge entlang der beiden Raum-Achsen. Die Auflösung ist immer quadratisch. Beispiel: ein Wert von 10 sorgt für ein Szenario, das im Raum eine Auflösung von 10x10 hat.
7. *train/eval/test-start/stop*: Der Index des ersten bzw. letzten (inklusiv) Pfades des Sets, das für das Training, zur Evaluation während des Trainings oder zum Berechnen der Bewertungen (test) verwendet werden soll.
8. *noise*: der Name der Konfigurationsdatei, die die Verrauschung verschiedener Daten steuert. Für mehr Details siehe Kapitel 5. Dieses Feld kann frei gelassen werden, wenn keine Verrauschung gewünscht ist.
9. *linear_optimization*: der Name der Konfigurationsdatei mit Hyperparametern für die *Lineare Optimierung*. Wenn die Methode nicht angewendet werden soll, kann dieses Feld frei gelassen werden.
10. *stochastic_optimization*: der Name der Konfigurationsdatei mit Hyperparametern für die *Stochastische Optimierung*. Wenn die Methode nicht angewendet werden soll, kann dieses Feld frei gelassen werden.
11. *inverse_rendering*: der Name der Konfigurationsdatei mit Hyperparametern für das *Inverse Rendering*. Wenn die Methode nicht angewendet werden soll, kann dieses Feld frei gelassen werden.
12. *inverse_bayes*: der Name der Konfigurationsdatei mit Hyperparametern für die Methode *Inverse Bayes*. Wenn die Methode nicht angewendet werden soll, kann dieses Feld frei gelassen werden.
13. *cuboid_splatting*: der Name der Konfigurationsdatei mit Hyperparametern für die Methode *Cuboid Splatting*. Wenn die Methode nicht angewendet werden soll, kann dieses Feld frei gelassen werden.

Das Experimentsystem kann mit dem folgenden Befehl per Kommandozeile gestartet werden:

```
python -m reconstruction.benchmark all <Pfad zu Hauptordner>
```

Für jeden Durchlauf wird dann im Ordner “runs” ein Ordner angelegt, der die ID des jeweiligen Durchlaufs als Namen trägt. In diesem Ordner befinden sich alle Daten, die zu diesem Durchlauf erzeugt werden. Anstatt des Keywords `all` können auch `prepare` oder `reconstruct` verwendet werden, um jeweils nur die Hälfte des Systems auszuführen, die entweder die Szenarien und Bewegungsprofile vorbereitet, generiert und visualisiert (prepare), oder die Methoden ausführt und deren Evaluationen und Visualisierungen berechnet (reconstruct).

4.1.3 Verwendete Hardware

Die Experimente wurden auf zwei ähnlichen Laptops getestet. Bei beiden Modellen handelt es sich um Geräte der Kategorie „Gaming-Laptop“, die von uns verwendet wurden, weil sie eine Grafikkarte beinhalten, mit deren Hilfe wir das Training der Methoden *Inverse Rendering* und

Cuboid Splatting schneller als nur auf einer CPU durchführen konnten. Wir listen die Eigenschaften beider Laptops in Tabelle 4.2 auf. Für die Ausführung der Methoden *Lineare Optimierung*, *Stochastische Optimierung*, *Inverse Bayes* und *Inverse Rendering* wurde Laptop 1 verwendet. Für die Ausführung der Methode *Cuboid Splatting* Laptop 2.

Tabelle 4.2: Eigenschaften der verwendeten Laptops

Eigenschaft	**Laptop 1**	**Laptop 2**
CPU	12th Gen Intel® Core™ i5-12450H	13th Gen Intel® Core™ i7-13700H
GPU	NVIDIA GeForce RTX 3050 Ti Mobile	NVIDIA GeForce RTX 4050 Laptop GPU
Arbeitsspeicher (CPU)	16 GB	16 GB
Grafikspeicher (GPU)	4 GB	6 GB

4.2 Bewertungsmetriken

Um die in den Durchläufen berechneten Rekonstruktionen evaluieren zu können, verwenden wir eine Reihe von Metriken, die wir im Folgenden vorstellen.

4.2.1 Überlappung und Relevanz

Die *Überlappung* eines Raumzeitpunktes ist definiert als die Anzahl an Pfaden, die durch diesen Raumzeitpunkt verläuft. Wir messen die Überlappung, weil wir glauben, dass eine höhere Überlappung eine bessere Rekonstruktion hervorbringen müsste, da in einem solchen Punkt die Information von mehr Pfaden miteinander kombiniert wird.

Wir bezeichnen einen Raumzeitpunkt als *relevant*, wenn seine Überlappung einen festgelegten Mindestwert hat. Wir benutzen diese Eigenschaft, um uns in unseren Bewertungen nur auf solche relevanten Punkte konzentrieren zu können. Wir beabsichtigen damit, nur solche Raumzeitpunkte als Rekonstruktion zu benutzen, die aufgrund ihrer höheren Überlappung ein Mindestmaß an Vertrauen in ihre rekonstruierten Werte haben. Den jeweiligen Mindestwert geben wir zu jeder Relevanz mit an. Eine Relevanz von 1 bedeutet beispielsweise lediglich, dass keine Raumzeitpunkte beachtet werden, durch die keine Pfade geführt haben.

Die *Überlappung* eines Szenarios definieren wir als die durchschnittliche Überlappung über allen Raumzeitpunkten mit Relevanz 1, durch die also mindestens ein Pfad verläuft. Bei einer hohen durchschnittlichen Überlappung rechnen wir ebenfalls mit einer besseren Qualität der Rekonstruktion.

4.2.2 Unterscheidung nach Datenquelle: Szenariofehler und Pfadfehler

Zuerst unterscheiden wir die Metriken anhand der Daten, die sie vergleichen. Es gibt zwei Kategorien: Szenariofehler und Pfadfehler.

Szenariofehler vergleichen die Raumzeitpunkte von zwei Szenarien miteinander. Wir vergleichen jeweils das rekonstruierte Szenario mit dem Originalszenario. Wir vergleichen jedoch nicht immer alle Raumzeitpunkte, weil nicht immer alle Raumzeitpunkte für den Vergleich sinnvoll sind. Dafür benutzen wir den Grad der Relevanz eines Raumzeitpunktes, der sich aus der Anzahl an Pfaden ergibt, die sich in diesem Punkt überschneiden. So können wir beispielsweise Raumzeitpunkte ausschließen, durch die kein oder nur ein Pfad verläuft. Unsere Hoffnung ist, dass ein Raumzeitpunkt, der auf der Basis von mehr Pfaden rekonstruiert wurde, auch eine höhere Qualität hat. Ist das der Fall, könnten wir die Rekonstruktion auf weniger, aber qualitativ hochwertigere Raumzeitpunkte beschränken. Während der Anwendung der Methode in der Praxis, können keine Szenariofehler berechnet werden, weil kein originales Szenario vorliegt. In unseren Experimenten sind die Szenariofehler jedoch die primäre Metrik, weil sie direkt messen, wie gut das originale Szenario rekonstruiert werden konnte.

Pfadfehler vergleichen die Expositionen von Pfaden miteinander. Dabei werden die gleichen Pfade verwendet, aber einmal über dem rekonstruierten und einmal über dem originalen Szenario verrechnet. Zu beachten ist, dass wir immer nur Pfadfehler über Szenarien mit der gleichen Auflösung berechnen. Das bedeutet, dass die Expositionen über dem originalen Szenario mit der Auflösung der Rekonstruktion berechnet werden. Wir unterscheiden dabei außerdem zwischen Pfadfehlern auf den Trainingsdaten, die angeben, wie sehr die Methode die gegebenen Trainingspfade erklären kann, und Pfadfehlern auf weiteren, nicht zum Trainieren benutzen Testpfaden, die messen, wie allgemeingültig die Rekonstruktion Pfade erklären kann. Ist der

Trainingspfadfehler niedrig, aber der Testpfadfehler hoch, hat ein Overfitting der Rekonstruktion stattgefunden. Pfadfehler stehen auch während der Anwendung der Methode in der Praxis zur Verfügung, weil für die Personen, die in einer Notfallstation gemessen wurden, die originalen Expositionen zu den Bewegungsprofilen bekannt sind. Diese Pfade können wir als Trainingspfade verwenden. In unseren Experimenten sind Pfadfehler die sekundäre Metrik, weil sie nur indirekt messen, wie gut das originale Szenario rekonstruiert werden konnte.

4.2.3 Unterscheidung nach Berechnung

Unabhängig davon, ob wir Raumzeitpunkte oder Expositionen vergleichen, benutzen wir eine Reihe von verschiedenen Fehlermaßen. Diese lassen sich in absolute und relative Fehlermaße unterteilen. Die absoluten Fehlermaße sind der mittlere absolute Fehler, der mittlere quadratische Fehler und der maximale Fehler. Die relativen Fehlermaße sind der anteilige Prozentfehler und der Differenzprozentfehler, die es jeweils mit der Aggregation als Durchschnitt, Median und Maximum gibt, sowie der summierte Prozentfehler.

Wir beschreiben im Folgenden die verschiedenen Fehlermaße. Dabei verwenden wir das Symbol o für die Originalwerte und r für die rekonstruierten Werte. Wir benutzen einen Index i von 0 bis n, wobei n die Anzahl aller Werte einer Sorte ist. Wir gehen davon aus, dass die originalen und rekonstruierten Werte jeweils punktweise zusammengehören, dass also o_i und r_i jeweils miteinander vergleichbar sind. Das ist gegeben, indem es sich bei i um die gleiche Position in der Raumzeit oder um den Index desselben Pfades handelt.

Der mittlere absolute Fehler wird berechnet mittels:

$$MAF = \frac{1}{n}\sum_{i=0}^{n} |o_i - r_i|.$$

Der mittlere quadratische Fehler wird berechnet mittels:

$$MQF = \frac{1}{n}\sum_{i=0}^{n} (o_i - r_i)^2.$$

Der maximale Fehler wird berechnet mittels:

$$MF = \max_{0 \leq i < n} o_i - r_i.$$

Der anteilige Prozentfehler misst, wie viel Prozent der Originalwerte rekonstruiert werden konnte. Dem liegt die Beobachtung zugrunde, dass die Rekonstruktion vor allem bei den gradientenbasierten Verfahren natürlicherweise dazu neigt, das Original zu unterschätzen, weil sie sich dem Original von null aus schrittweise annähert. Der Fehler berechnet punktweise den prozentualen Anteil der rekonstruierten Werte an den originalen Werten und aggregiert diesen jeweils als Durchschnitt, Median und Maximum. Wir notieren diese Aggregation im Folgenden mit der Funktion f:

$$PF_f^A = f\left(100\frac{r_i}{o_i}\middle| 0 \leq i < n\right).$$

Der Differenzprozentfehler misst dagegen nicht den Anteil der Rekonstruktion am Original, sondern den Anteil des Abstandes zwischen Rekonstruktion und Original. Auch dieser Fehler wird punktweise berechnet und dann als Durchschnitt, Median und Maximum aggregiert. Wir verwenden ebenfalls die Funktion f:

$$PF_f^D = f\left(100\frac{|o_i - r_i|}{o_i}\middle|0 \leq i < n\right).$$

Zuletzt definieren wir noch den summierten Prozentfehler, der ähnlich zum Differenzprozentfehler den Anteil des Fehlers am Original misst. Dabei wird dieser Prozentsatz jedoch nicht punktweise berechnet und dann aggregiert, sondern zuerst summiert und dann ein Prozentsatz berechnet:

$$PF^S = \frac{\sum_{i=0}^{n}|o_i - r_i|}{\sum_i^n o_i}.$$

4.2.4 Auswahl der verwendeten Metriken

Aus der Zusammensetzung von einer Datenquelle und einer Berechnungsvorschrift entsteht dann eine Metrik, die wir für eine Rekonstruktion nach einem Trainingsdurchlauf messen können. Ein oft benutztes Beispiel ist der summierte Szenarioprozentfehler mit einer Relevanz von 1, also nur auf Raumzeitpunkten, durch die mindestens ein Pfad verlaufen ist.

Als Datenquellen verwenden wir zwei Pfadfehler. Einen für die Trainingspfade, sowie einen für vom Training unabhängige Testpfade. Auf den Szenarien untersuchen wir Fehler vor allem für die Relevanzen von 1 und 5. Die Relevanz von 1 filtert ein Szenario auf alle Punkte, die überhaupt rekonstruierbar waren. Die Relevanz von 5, die alle Punkte selektiert, durch die mindestens 5 Pfade verlaufen, sehen wir als einen guten Kompromiss zwischen der Erreichbarkeit dieser Relevanz für einzelne Punkte und einer Verbesserung der Ergebnisse gegenüber einer Relevanz von 1.

Unsere Implementierung berechnet alle beschriebenen Metriken für jeden Durchlauf. Für unsere Auswertung müssen wir uns jedoch auf einige wenige Metriken beschränken.

Unter den Berechnungen konnten wir alle durchschnittlichen quadratischen Fehler schnell ausschließen, weil sie die sowieso schon um Größenordnungen auseinanderliegenden Werte noch weiter überbetont hätten. Den mittleren absoluten Fehler fanden wir unter den absoluten Fehlern am nützlichsten. Wie alle absoluten Fehler hat aber auch diesen den Nachteil, dass er immer im Kontext eines speziellen Szenarios interpretiert werden muss, um eine Güte aus ihm abzuleiten. Dazu kommt, dass verschiedene Szenarien mit verschiedenen absoluten Fehlern nicht gut untereinander vergleichbar sind. Deswegen haben wir uns bei unseren Auswertungen für den Einsatz der relativen Metriken entschieden, die jeweils nach den Originalwerten normiert sind und deshalb eine relative Güte beschreiben, die auch zwischen Szenarien vergleichbar ist. Mit den punktweisen Metriken absoluter Prozentfehler und Differenzprozentfehler hatten wir Probleme mit verschiedenen Artefakten. Zum einen muss darauf geachtet werden, dass keine Punkte verglichen werden, die eine Division durch null verursachen. Zum anderen hatten wir Probleme mit sehr kleinen Werten, die nah bei null lagen. Bei diesen Werten waren die relativen Fehler oft sehr hoch, obwohl faktisch nur eine sehr kleine Menge an Iod nicht rekonstruiert werden konnte. Eine Rekonstruktion, die einen Wert nah bei null mit null rekonstruiert hat, liegt in diesen Metriken 100% falsch. Letztendlich haben wir uns als Hauptmetrik für den summierten Prozentfehler entschieden, weil dieser numerisch stabil ist, solange nicht das gesamte Original null ist, und durch die Summenbildung vor der Normierung eher die absolut größeren Werte eines Szenarios fokussiert, ohne jedoch die kleinen Werte völlig außenvor zu lassen.

4.3 Die Implementierungen der Methoden (AP 3)

In diesem Kapitel stellen wir alle von uns implementierten Methoden und die Ergebnisse, die wir mit ihnen erzielen konnten, vor. Dabei fokussieren wir uns auf das methodische Verständnis und die Konfiguration der Methoden für die Durchführung von Experimenten. Die Methoden *Lineare Optimierung*, *Stochastische Optimierung* und *Inverse Bayes* haben wir nach einigen Evaluationsdurchläufen für die vertiefte Evaluation ausgeschlossen, weil sich herausgestellt hat, dass sie mit der geforderten Problemgröße unter den gegebenen Bedingungen nicht nutzbar sind. Die Methoden *Inverse Rendering* und *Cuboid Splatting* haben sich als geeigneter erwiesen.

In den nachfolgenden Kapiteln gehen wir detaillierter auf die Ergebnisse der Methoden *Inverse Rendering* und *Cuboid Splatting* ein, die wir zum einen allgemein über einem Raum verschiedener Parameterkombinationen und zum anderen spezifisch entlang einzelner Parameter evaluiert haben.

4.3.1 Die Implementierung der Methode *Lineare Optimierung*

Zur Implementierung der linearen Optimierung wurde die Bibliothek *SciPy* benutzt, die eine Funktion zur Lösung von linearen Optimierungsproblemen mit verschiedenen Solvern bereitstellt. Unsere Implementierung stellt damit in erster Linie eine Verbindung zwischen dieser Funktion und den gegebenen Daten dar, indem diese Daten in einer für SciPy nutzbaren Darstellung modelliert werden.

Wir modellieren das Problem mit Gleichungen, die ähnlich zu der Modellierung in Kapitel 2.2.2.3 ist, jedoch die Unsicherheit der Daten mit beachtet: $Ax = m$. Wir leiten im Folgenden die Bestandteile dieser Gleichung schrittweise her.

Zuerst sei **x** der Vektor, in dem alle Iod-Konzentrationen aller Raumzeitpunkte enthalten sind, und **A** die Matrix, in der alle Vorfaktoren von allen Raumzeitpunkten und allen Bewegungsprofilen enthalten sind. Jede Zeile dieser Matrix entspricht dem Bewegungsprofil einer Person. Die meisten Einträge von A sind Null, weil eine Person sich durch die meisten Raumzeitpunkte nicht bewegt hat. Deswegen nutzen wir eine Sparse-Matrix, die nur die Einträge speichert, die ungleich null sind. Der Vektor **m** beinhaltet für jedes Bewegungsprofil die dazugehörige Expositionsmessung.

Die Modellierung als eine solche Gleichung lässt keine Unsicherheiten bei den Eingabedaten zu. Wenn es keine perfekte Lösung gibt, kann keine Lösung gefunden werden. Wir fügen deshalb für jedes Bewegungsprofil zu **x** zwei Fehlervariablen hinzu: einen positiven und einen negativen Fehler. Der positive Fehler wird mit einem konstanten Vorfaktor von 1 addiert, der negative Fehler mit einem Vorfaktor von –1. Die Vorfaktoren werden jeweils der Matrix A hinzugefügt. Beide Fehlervariablen werden darauf beschränkt, positive Werte darzustellen.

$$\begin{array}{c} \begin{array}{ccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & e_1^+ & e_1^- & e_2^+ & e_2^- \end{array} \\ \left(\begin{array}{ccccccccc} 0.1 & 0 & 0 & 0 & 0.3 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0.2 & 0.4 & 0 & 0 & 0 & 1 & -1 \end{array} \right) \end{array}$$

Abbildung 4.1: Zu sehen ist die Matrix A aus der Gleichung Ax=m mit ihren Elementen. Jede Zeile modelliert ein Bewegungsprofil. Die Spalten bestehen aus Variablen für die Iod-Konzentrationen der einzelnen Raumzeitpunkte (blau) und pro Bewegungsprofil aus zwei Fehlervariablen, die für das jeweilige Bewegungsprofil mit -1 und 1 gewichtet sind (rot; Über- und Unterschätzung, der Wert der Variablen ist immer positiv).

Das Problem wird dann darauf optimiert, die Summe aller Fehler, die immer positiv ist, zu minimieren. Diese Modellierung erlaubt es, dass immer eine Lösung für x gefunden werden kann, auch wenn diese nur mit hohen Fehlerwerten zu einer Erfüllung der Gleichung führt. Die Werte in x werden dann wieder als Raumzeitpunkte interpretiert und als Ergebnis ausgegeben.

Zuletzt wurde noch eine weitere Optimierung implementiert, die die Größe der Modellierung deutlich reduzieren kann. Es gibt in unseren Experimenten viele Raumzeitpunkte, die von keinem Bewegungsprofil durchlaufen werden. Diese Punkte müssen nicht modelliert und auch nicht gelöst werden. Deshalb haben wir ein Mapping von relevanten Raumzeitpunkten auf Elemente in dem Vektor x implementiert, das darin resultiert, dass in A und x nur noch relevante Raumzeitpunkte enthalten sind. Da ein einzelnes Bewegungsprofil dennoch nur durch wenige relevante Raumzeitpunkte führt, bleibt die Implementierung von A als Sparse Matrix dadurch unberührt.

Die Konfiguration der Methode erfolgt mittels einer JSON-Datei mit folgendem Format:

```
{
  “regularization_factor”: 1e-6
}
```

Der Hyperparameter gibt den Vorfaktor für die Raumzeitvariablen im Optimierungsziel der linearen Optimierung an und gibt damit an, wie stark eine Lösung mit möglichst wenig Gesamtmenge angestrebt wird.

Die Fehlervariablen fließen in der Optimierung immer mit einem Vorfaktor von 1 ein. Sie sind das primäre Optimierungsziel, bei dem eine Rekonstruktion mit möglichst wenig Gesamtabweichung erreicht werden soll.

Wir haben uns gegen eine tiefere Evaluation der linearen Optimierung entschieden, weil die Methode harte Grenzen bei der Verarbeitung von Daten aufweist. Ihre Schwäche liegt darin, das ganze Problem in einem Durchlauf kodieren und dann komplett lösen zu müssen. Im Vergleich dazu können die von uns präferierten Methoden das Problem in Teilschritten optimieren und damit auch mit größeren Mengen an Daten umgehen. Eine Anwendung der linearen Optimierung innerhalb von bestimmten Problemgrenzen ist damit jedoch weiterhin denkbar. In Tabelle 4.3 sind die von uns durchgeführten Experimente eingetragen. Die Durchläufe ohne Werte mussten wir nach unverhältnismäßig langer Laufzeit abbrechen. Wir kommen damit zu dem Ergebnis, dass wir mit der linearen Optimierung unter den gegebenen Hardwarebedingungen nicht im gesamten geforderten Größenbereich arbeiten können. Zusätzlich haben unsere Experimente gezeigt, dass

die Qualität der Rekonstruktion einer linearen Optimierung schlechter ist als die eines inversen Rendering.

Tabelle 4.3: Ergebnisse der Experimente für die Methode Lineare Optimierung

ID	Auflösung	Pfade	% scen 1 rel sum	% scen 5 rel sum	% scen 10 rel sum	% train rel sum	% test rel sum
6000	10	10k	104.5273	93.6234	89.4909	0.0	15.7385
6005	10	25k	56.3371	53.9339	49.4991	0.0	1.9624
6006	10	50k	20.9523	16.6616	14.6114	0.0	0.0004
6007	10	100k					
6004	30	25k	118.2751	109.8493	109.0903	0.0	99.4243
6009	30	50k					
6008	100	10k	105.0309	99.0578	85.5875	0.0	14.2075
6011	100	25k					

4.3.2 Die Implementierung der Methode *Stochastische Optimierung*

Die Implementierung der stochastischen Optimierung baut direkt auf der linearen Optimierung auf. Einfach zusammengefasst, werden mehrere Durchläufe einer linearen Optimierung mit leicht verrauschten Parametern durchgeführt und deren Lösungen jeweils in einem Mittelwert aggregiert, der die Lösung der stochastischen Optimierung bildet. Die Implementierung der Stochastischen Optimierung baut direkt auf der Implementierung der Linearen Optimierung auf und erweitert diese nur um Methoden zur Verrauschung der Schutzfaktoren und Expositionswerte. Beide Werte werden mit einem zufälligen multiplikativen Rauschen verrechnet.

Die Konfiguration der Hyperparameter erfolgt mittels einer JSON-Datei mit folgendem Format:

```
{
  "regularization_factor": 1e-6,
  "steps": 100,
  "factor_noise": 0.1,
  "exposure_noise": 0.1
}
```

Der Regularisierungsfaktor funktioniert analog zu dem der linearen Optimierung. Die Anzahl der Schritt gibt an, wie viele verrauschte linearen Optimierungen in ein Ergebnis akkumuliert werden sollen. Die beiden Rauschfaktoren geben die maximale Rauschweite an. Ein Wert von 0,1 lässt damit ein multiplikatives Rauschen zwischen 0,9 und 1,1 zu.

Aufbauend auf unserer Entscheidung zur linearen Optimierung schließen wir auch die stochastische Optimierung aus der weiteren Evaluierung aus, da sie natürlicherweise den gleichen Beschränkungen unterliegt, weil sie aus vielen linearen Optimierungen besteht.

4.3.3 Die Implementierung der Methode *Inverse Bayes*

Inverse Bayes formuliert, wie bereits in Kapitel 2.2.2.5 beschrieben, das Problem als Inferenz eines Posteriors aus einem Prior und der gegebenen Evidenz. Wird kein Prior bei der Verwendung angegeben, verwenden wir einen Prior, der an allen Stellen den Wert Null hat, also annimmt, dass kein radioaktives Iod existiert.

Die Methode wurde mithilfe der Bibliothek PyMC implementiert. PyMC bietet die Möglichkeit probabilistische Modelle zu formulieren und mittels Markov-Chain-Monte-Carlo-Methoden (MCMC) Lösungen für diese anzunähern. Dafür werden Werte aus einer Verteilung gesampelt, die proportional zu dem gesuchten Posterior ist. Aus diesen Samples kann dann der Posterior abgeschätzt werden.

Insgesamt wurden von uns zwei Implementierungen angefertigt. Beide Implementierungen nutzen die gleiche Formulierung des Problems, die bereits auch schon für *Lineare Optimierung* beschrieben wurde. Die erste Implementierung stellt eine Modellierung des Grundproblems dar und nutzt die durch PyMC bereitgestellte Funktion zur automatischen Ermittlung einer Lösung des Problems. Weil sich diese Implementierung schnell als zu ungünstig herausgestellt hat, haben wir eine zweite Implementierung erstellt, in der wir die Lösungsformel direkt implementiert haben, und so weniger auf interne Funktionalitäten von PyMC angewiesen waren. Diese Lösung ist bereits vereinfacht und modelliert die Raumzeitpunkte nur noch als unabhängige multivariate Normalverteilung, bei der keine Kovarianz zwischen den einzelnen Raumzeitpunkten existiert, und nicht, wie das bei in der Standardlösung der Fall ist, als multivariate Normalverteilung mit Kovarianz.

Die Konfiguration der Methode erfolgt über eine JSON-Datei mit folgendem Format:

```
{
"exposure_sigma": 0.1,
      "factor_sigma": 0.01,
      "scenario_sigma": 5,
      "draws": 2000,
      "tune": 1000,
      "chains": 4,
      "cores": 4
}
```

Es sind jeweils eine Standardabweichung für die Messwerte, die Schutzfaktoren und die Werte des Priors gegeben. In der Implementierung mit Lösungsformel wird die Standardabweichung für die Schutzfaktoren jedoch nicht benutzt. Die restlichen Parameter sind technische Parameter für die MCMC. Es wird angegeben, wie viele Samples generiert werden (“draws”), wie viele Schritte das Modell benutzt, um sich zu stabilisieren (“tune”), wie viele Sampling-Chains benutzt werden (“chains”) und auf wie vielen CPU-Kernen diese Chains parallel ausgeführt werden (“cores”).

Die Implementierung von *Inverse Bayes* unterliegt dem gleichen Problem wie auch lineare und stochastische Optimierung: das Problem muss als Ganzes kodiert und gelöst werden. Ein Durchlauf mit einer Auflösung von 30x30 und 10.000 Pfaden kann mit 16 GB Arbeitsspeicher nicht durchgeführt werden. In zwei Durchläufen mit unterschiedlichen Hyperparametern konnten wir mittels der Methode darüber hinaus auch für eine Auflösung von 10x10 keine guten Ergebnisse erzielen.

Tabelle 4.4: Ergebnisse der Experimente für die Methode Inverse Bayes

ID	Auflösung	Pfade	% scen 1 rel sum	% scen 5 rel sum	% scen 10 rel sum	% train rel sum	% test rel sum
6002	10x10	10k	384.2372	431.7838	436.1916	3726.7971	2583.6285
6014	10x10	10k	11066.3017	11739.7461	11476.8112	17538.9618	14321.6871
6015	30x30	10k					

4.3.4 Die Implementierung der Methode *Inverse Rendering*

4.3.4.1 Überblick über die Methode *Inverse Rendering*

Die Methode *Inverse Rendering* optimiert mittels eines schrittweisen Gradientenverfahrens die relevanten Raumzeitpunkte eines Szenarios entlang einer Teilmenge, auch Batch genannt, von Pfaden. Für jeden Pfad wird dafür die tatsächliche Exposition am Ende des Pfades mit der auf den aktuellen Raumzeitpunkten berechneten Exposition verglichen. Das Gradientenverfahren passt dann die Raumzeitpunkte so an, dass ein Fehlermaß zwischen den aktuellen und originalen Raumzeitpunkten minimiert wird. Wir benutzen als Fehlermaß die mittlere quadratische Abweichung und als Gradientenverfahren AdamW (Loshchilov & Hutter, 2019).

Im maschinellen Lernen wird ein Training oft in Epochen durchgeführt. Eine Epoche bedeutet das einmalige Betrachten aller Trainingsdaten. Das Training der Methode inverses Rendering erfolgt über viele Epochen. Während jeder Epoche werden zunächst alle Trainingspfade gemischt und dann in gleich große Batches aufgeteilt. Pro Batch wird einmal das Gradientenverfahren angewendet. Wurden einmal alle Trainingspfade verrechnet, endet die Epoche und das Modell wird einmal auf weiteren Validierungspfaden evaluiert, die nicht zum Trainieren verwendet wurden. Dafür wird für diese Validierungspfade ebenfalls das Fehlermaß berechnet. Nach Durchlauf aller Epochen, ist das Ergebnis des Trainings das Modell aus der Epoche, in der der Evaluationsfehler am geringsten war. Dadurch verhindern wir ein Overfitting auf den Trainingsdaten.

Ein Problem ist, dass das Gradientenverfahren die Werte des Szenarios negativ werden lassen kann. Wir haben verschiedene Lösungen ausprobiert, darunter die Berechnung der absoluten und quadratisches Werte des Szenarios, sowie den Einsatz der Aktivierungsfunktion ReLU, die alle negativen Zahlen auf null und alle positiven auf sich selbst abbildet. Die besten Ergebnisse konnten wir jedoch damit erlangen, dass wir nach jedem Schritt des Gradientenverfahrens einfach alle negativen Werte auf null zurückgesetzt haben.

Einen großen positiven Einfluss auf unsere Ergebnisse hatte außerdem ein zusätzliches Scheduling der Lernrate des Gradientenverfahrens. Wir benutzen einen Scheduler, der die Lernrate immer dann um einen gewissen Prozentsatz verringert, wenn das Training über eine bestimmte Anzahl an Epochen keine bessere Lösung mehr produzieren konnte. Wir konnten beobachten, dass dadurch eine gute Mischung aus einem zügigen Antrainieren des Modells zu Beginn und einem detaillierten Verfeinern im späteren Trainingsverlauf möglich wird.

Weiterhin haben wir eine Reihe von zusätzlichen Heuristiken implementiert, deren Nutzung sich aber schnell als nicht zielführend herausgestellt hat. So haben wir Regularisierungen für positive Raumzeitpunkte eingebaut, sodass diese möglichst gering, oder bei Angabe eines Priors, möglichst nah am Prior orientiert sind. Details zum Einsatz von Prioren folgen in Abschnitt 4.3.4.3. Parallel dazu haben wir eine Regularisierung von negativen Werten getestet, die jedoch seit der Kürzung der negativen Werte keinen Nutzen mehr hat. Eine Regularisierung zur Glättung reduziert die Unterschiede zwischen benachbarten Punkten, um ein gleichmäßigeres Volumen zu erreichen. Dazu haben wir versucht, zwischen den Epochen eine geringe Menge an Rauschen zu addieren, um das Modell aus lokalen Optima zu befreien. Das war vor allem relevant, als wir noch mit ReLU experimentiert haben, weil ein Gradientenverfahren Werte von null nicht verändern konnte, da die ReLU-Funktion bei null selbst einen Gradienten von null hat. Außerdem haben wir mit einem Dropout-Mechanismus versucht, ob das Modell ähnlich zur Bildverarbeitung robuster trainiert, wenn wir manche Raumzeitpunkte für einen Schritt auf null setzen. AdamW hat zudem die Möglichkeit ein Weight-Decay durchzuführen, das bei jedem Schritt alle Werte leicht

verringert. Auch damit soll eine gemäßigtere Lösung gefunden werden können. Alle diese Heuristiken haben sich nicht als zielführend oder ausschlaggebend herausgestellt, sodass sie in unseren späteren Experimenten mit null konfiguriert wurden.

Wir haben die Methode mit PyTorch implementiert. Dadurch ist es möglich und empfehlenswert, das Training auf einer GPU auszuführen. Der Zeitaufwand kann durch eine GPU deutlich reduziert werden. Wenn keine GPU gegeben ist, kann allerdings auch eine CPU genutzt werden.

4.3.4.2 Konfiguration der Methode *Inverse Rendering*

Die Konfiguration erfolgt durch eine JSON-Datei mit folgendem Format:

```
{
  "min_epochs": 10000,
  "max_epochs": 10000,
  "early_stopping_epochs": 10000,
  "threshold": 0.0,
  "learning_rate": {
    "max": 100,
    "min": 0.01,
    "patience": 30,
    "threshold": 0.0,
    "factor": 0.4,
    "cooldown": 200
  },
  "device": "cuda",
  "batch_size": 10000,
  "buffer": true,
  "regularization": 0.0,
  "negative_regularization": 0.0,
  "smoothing": 0.0,
  "noise": 0.0,
  "dropout": 0.0,
  "weight_decay": 0.0
}
```

Konfiguriert werden können die minimale und maximale Anzahl der Epochen, die Anzahl der Epochen, ab der ein vorzeitiges Stoppen des Trainings möglich ist, wenn keine signifikante Verbesserung mehr passiert, die Mindestgröße dieser Verbesserung (threshold), Angaben zur Lernrate, wie die maximale Lernrate zu Beginn und die minimale, ab der keine Reduzierung mehr stattfindet, die Anzahl der Epochen, nach denen die Lernrate reduziert wird (patience), wenn keine Mindestverbesserung (threshold) erreicht wurde, den Faktor zur Reduzierung der Lernrate und eine Anzahl an Epochen, in denen nach einer Reduzierung der Lernrate zunächst keine weitere Reduzierung stattfinden kann (cooldown), weil das Training in diesen Phasen meist etwas ausschlägt und sich erst wieder einpegeln muss. Außerdem ein Bezeichner für das Gerät, auf dem das Training stattfindet, den PyTorch versteht (beispielsweise “cuda” für CUDA-fähige GPUSs, oder alternativ “cpu”, wenn nur die CPU verwendet werden soll), die Anzahl der Pfade pro Batch, eine Angabe, ob alle Trainingspfade permanent im RAM geladen werden sollen (buffer), was eine Entscheidung zwischen der notwendigen Größe des Speichers und der notwendigen Laufzeit ist, einem Regularisierungsfaktor für positive Werte, einem Regularisierungsfaktor für negative Werte,

einem Regularisierungsfaktor für Glättung, einem Faktor für additives Rauschen zwischen den Epochen, einer Dropout-Wahrscheinlichkeit und einem Faktor für Weight Decay für AdamW.

Den meisten Einfluss konnten wir bei der Konfiguration der Epochen und des Scheduling der Lernrate feststellen. Die oben gezeigte Konfiguration, die sich daraus ergeben hat, haben wir dann für fast alle Experimente verwenden können. Anpassungen waren danach vor allem deswegen sinnvoll, weil Durchläufe mit mehr Trainingspfaden pro Epoche mehr Schritte des Gradientenverfahrens machen und deswegen auf weniger Epochen angewiesen sind, wogegen Trainings auf weniger Pfaden mehr Wiederholungen benötigen.

4.3.4.3 Der Einsatz von Prioren mit der Methode *Inverse Rendering*

Die Methode *Inverse Rendering* kann mit einem Priorszenario angewendet werden. Ein Prioszenario ist ein Szenario, das im Idealfall bereits eine Annäherung der angestrebten Rekonstruktion darstellt. Es kann zum Beispiel durch eine Wettersimulation erhalten werden.

Ist ein Priorszenario gegeben, wird die Rekonstruktion zu Beginn des Gradientenverfahrens mit dessen Werten initialisiert. Im Vergleich dazu wird die Rekonstruktion mit Nullen initialisiert, wenn kein Prior gegeben ist, sodass dann an jedem Raumzeitpunkt zunächst davon ausgegangen wird, dass es eine Iod-Konzentration von Null gibt.

Wir betrachten das Gradientenverfahren als schrittweise Korrektur der Initialisierung. Allerdings erreicht ein Gradientenverfahren immer nur ein lokales Optimum. Es wird also nicht unbedingt die eigentliche Iod-Konzentration in der Umwelt rekonstruiert, sondern gegebenenfalls eine andere, die unter den beobachteten Daten ebenfalls plausibel ist. Bei einem Problem, das in der Regel so unterbestimmt ist, wie das unsere, könnte daher die Wahl einer besser geeigneteren Initialisierung das Gradientenverfahren zu einem besseren lokalen Optimum konvergieren lassen, weil die Initialisierung sich bereits näher an der tatsächlichen Rekonstruktion befindet.

Während der Implementierung der Methode haben wir auch probiert, die Regularisierung der Methode nicht in Richtung Null, sondern in Richtung des Priors zeigen zu lassen. Dieses Vorgehen hat aber nicht gut funktioniert, sodass es in unserer finalen Implementierung nicht enthalten ist.

4.3.5 Die Implementierung der Methode *Cuboid Splatting*

4.3.5.1 Zusammenhang mit der Original-Methode *Gaussian Splatting*

Wie bereits im Kapitel 2.2.2.6 skizziert, handelt es sich bei unserer Implementierung um eine Abwandlung der Methode *Gaussian Splatting*. Hierbei haben wir uns an der in Kerbl, et al. (2023) beschriebenen Grundidee des *Gaussian Splatting* orientiert, aber nicht die zugehörige Software übernommen, sondern eine eigene, auf unseren Anwendungsfall zugeschnittene, Implementierung angefertigt. Dabei verwendet unsere Methode Cuboids anstelle von Gaussians, da diese geometrischen Objekte für die bei uns nötigen Berechnungen der Volumenüberlappungen vorteilhaftere Eigenschaften mit sich bringen. Aus unserer Sicht wäre eine Implementierung mit Gaussians grundsätzlich auch möglich, würde aber voraussichtlich deutlich höhere Rechenzeiten und benötigte Speicherkapazitäten mit sich bringen, ohne dass dies anderweitig einen großen Mehrwert gegenüber Cuboids hätte.

4.3.5.2 Einführung in die Implementierung der Methode *Cuboid Splatting*

Die Abbildung 4.2 gibt einen ersten Einblick in die Funktionsweise der Methode *Cuboid Splatting*. Zu sehen sind zwei verschiedene Verteilungen von Cuboids in der Raumzeit. Das Koordinatensystem bildet auf der x- und y-Achse die räumlichen Dimensionen und auf der z-Achse die zeitliche Dimension der Raumzeit ab. Jeder Cuboid hat eine bestimmte Position in der Raumzeit, Ausdehnungen entlang der Achsen und einen ihm zugehörigen Belastungswert (dargestellt über die Intensität der Einfärbung). Über diese Repräsentation erfolgt die Rekonstruktion eines Szenarios, in dem die Positionen, Ausdehnungen und Belastungswerte der Cuboids in der Raumzeit im Trainingsprozess des Modells optimiert werden. Zudem kann auch die Anzahl der Cuboids im Training optimiert werden, indem sowohl Cuboids hinzugefügt als auch gelöscht werden können. Auf der linken Seite ist die Verteilung der Cuboids in der Raumzeit vor dem Trainingsprozess (nach der zufälligen Initialisierung der Cuboids im 3D-Raum) zu sehen. Auf der rechten Seite ist die Verteilung der Cuboids in der Raumzeit, nachdem das Training abgeschlossen wurde, zu erkennen. Man sieht, dass sich die Cuboids in der Form des zugrundeliegenden Gaussian-Szenarios angeordnet haben.

Die Vorteile der Methode *Cuboid Splatting* im Vergleich zur zuvor vorgestellten Methode *Inverse Rendering* liegen darin, dass das Modell stärker die übergeordneten raumzeitlichen Strukturen der Belastungsverteilung lernt. Dadurch wird Overfitting vermieden und die Methode erlangt eine höhere Robustheit. Während bei der Methode *Inverse Rendering* nur explizite Überschneidungen von Pfaden in Raumzeitpunkten zur Korrelation ihrer Belastungswerte führen, reicht dafür bei der Methode *Cuboid Splatting* bereits eine Benachbarung (räumliche Nähe) von Pfaden zueinander aus, wenn diese dadurch durch den gleichen Cuboid verlaufen.

Eine genauere Erläuterung der zugrundeliegenden Implementierung der Methode folgt in den nächsten Unterkapiteln. Wie auch die Methode *Inverse Rendering*, ist die Methode *Cuboid Splatting* mit dem Python Framework PyTorch implementiert worden.

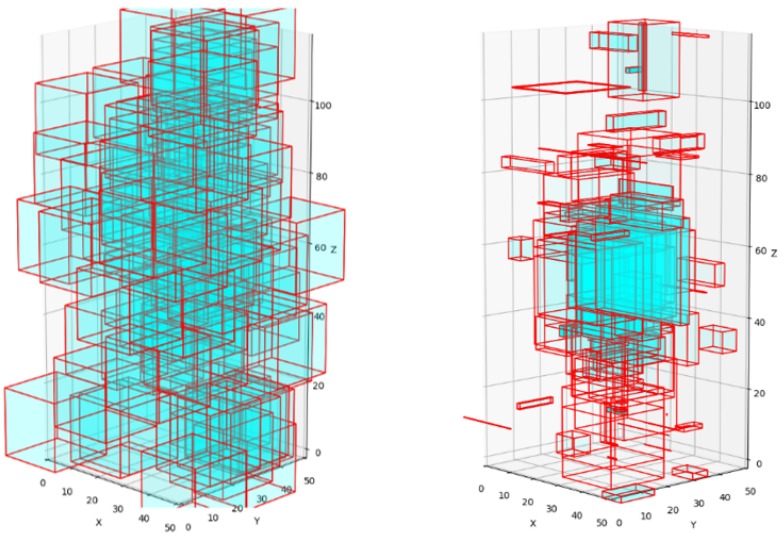

Abbildung 4.2: Die Verteilung der Cuboids in der Raumzeit, links: vor dem Training, rechts: nach abgeschlossenem Training (Training mit 50.000 Pfaden, Gaussian Szenario, Ortschaften Graph)

4.3.5.3 Das grundlegende Cuboid-Modell

Für die Initialisierung des Cuboid-Modells müssen über die Config-Datei des Modells einige Parameter festgelegt und der Software übergeben werden:

- die Anzahl der Cuboids zu Beginn des Trainings (*nr_of_cuboids*),
- den initialen Belastungswert der Cuboids zu Beginn des Trainings (*initial_radiation*),
- die Maße der Cuboids zu Beginn des Trainings (also: Länge, Breite und Höhe in jede Richtung ab dem Mittelpunkt des Cuboids) (*initial_length_width_height*),
- die Rasterung der dreidimensionalen Raumzeit (also: Auflösung in x- und y-Richtung, Anzahl der Zeitschritte in z-Richtung) (*grid_size*),
- die Wertebereiche der x-, y- und z-Achse (i.d.R. 0 bis Auflösung bzw. 0 bis letzter Zeitschritt) (*space_range_xy, space_range_z*)
- das Gerät, auf dem trainiert werden soll (also: CPU oder GPU) (*device*)

Das Modell initialisiert die Startpositionen der Cuboids in der Raumzeit und speichert diese. Sie werden zufällig in den Raum hineingestreut, allerdings wird die Reproduzierbarkeit dieses Zufallsprozesses durch einen in der Config-Datei konfigurierbaren Random Seed (*seed*) gesichert.

Es finden in diesem Schritt zudem weitere vorbereitende Berechnungen für das Training des Cuboid-Modells statt, die automatisch ablaufen. Dabei werden unter anderem Größe, Volumen, Minimal- und Maximalkoordinaten der Rasterzellen der Raumzeit berechnet (benötigt für die späteren Volumenberechnungen) und eine dreidimensionale Matrix zur Speicherung der jeweils aktuellen Rekonstruktion der Belastungswerte in der Raumzeit erzeugt. Diese ist schon aus der Implementierung der Methode *Inverse Rendering* bekannt. Anstatt dass die darin gespeicherten Werte direkt trainierbar sind, wie es beim *Inverse Rendering* der Fall ist, sind die trainierbaren Parameter im Fall des *Cuboid Splatting* die Positionen der Cuboids, deren Längen, Breiten und Höhen und ihre Belastungswerte. Aus diesen Parametern lassen sich dann die aktuellen Belastungswerte in der Raumzeit errechnen. Hierdurch wird die übergeordnete räumliche Struktur der Belastungsverteilung gelernt und im Vergleich zu *Inverse Rendering* sogenanntes Overfitting des Modells stärker vermieden.

Das Modell verfügt außerdem über eine Visualisierungsfunktion, mit der auf Basis des aktuellen Stands des Modells die Verteilung der Cuboids im Raum wie in Abbildung 4.2 visualisiert werden kann. Diese Funktion kann vor, während oder nach dem Training dazu genutzt werden, einen visuellen Einblick in die aktuelle Cuboid-Verteilung in der Raumzeit zu erhalten.

4.3.5.4 Die Forward-Methode des Cuboid-Modells

Ein Kernstück der Implementierung des Cuboid-Modells sind die Forward-Methode und die zugehörige Berechnung der aktuellen Rekonstruktion. Diese Methoden werden für den sogenannten Forward-Pass im Training des Modells eingesetzt. Dies ist ein typisches Vorgehen beim Einsatz des PyTorch Frameworks und diversen Deep Learning Trainingsprozessen.

Im Rahmen der Forward-Methode werden zunächst die aktuellen Belastungswerte in der Raumzeit auf Basis der aktuellen Verteilung und Form der Cuboids im Raum berechnet. Im Anschluss daran wird mithilfe eines Skalarprodukts aus Schutzfaktoren und Belastungswerten entlang eines jeden Pfades die rekonstruierte Belastung dieses Pfades nach aktuellem Trainings-Stand des Modells berechnet. Im Trainingsprozess können diese berechneten Pfadbelastungen dann mit den echten Pfadbelastungen verglichen werden und anhand dessen das Cuboid-Modell verbessert werden.

Die Berechnung der aktuellen Belastungswerte in der Raumzeit geschieht nach folgender Berechnungslogik: Es wird die Überlappung aller Cuboids mit den Rasterzellen der Raumzeit berechnet. Hierbei wird das Überlappungsvolumen mit dem Cuboid anteilig am Gesamtvolumen einer Raster-Zelle betrachtet. Dieser Anteil wird dann mit dem Belastungswert des jeweiligen Cuboids multipliziert und auf den Belastungswert der Rasterzelle in der Raumzeit addiert. Folglich bedeutet dies auch, dass sich an Stellen, wo sich mehrere Cuboids überlappen, deren Belastungswerte in diesen Bereichen addieren. Beispiel: Cuboid 1 hat Belastungswert 0.5, Cuboid 2 hat Belastungswert 0.2 → im Überlappungsbereich der beiden Cuboids liegt eine Belastung von 0.7 vor.

4.3.5.5 Der Trainingsprozess des Cuboid-Modells - Setup

Nun erfolgt das Setup des Trainingsprozesses der Methode *Cuboid Splatting*. Für das Training müssen ebenfalls einige Parameter über die Config-Datei des Modells konfiguriert werden:

- Die zu Trainingsbeginn angewandte Learning Rate (*initial_lr*). Diese bestimmt darüber, wie „groß“ die Optimierungsschritte in Richtung des negativen Gradienten der Loss-Funktion sind, die während des Trainings gemacht werden, um den Fehler des Modells zu minimieren.
- Wie viele Epochen lang das Modell trainieren soll (*num_epochs*).
- Wie groß die einzelnen Batches an Daten sein sollen, mit denen das Modell trainiert (*batch_size*, z.B. 100 Pfade pro Batch). Die Größe der Batches muss insofern zur Anzahl der verwendeten Trainingspfade passen, dass kein Rest übrigbleibt. Beispiel: Bei 10.000 Trainingspfaden könnten zum Beispiel Batch-Größen von 100, 500 oder 1.000 Pfaden verwendet werden. Eine Batch-Größe von 700 bietet sich nicht an, da 10.000 nicht ohne Rest durch 700 teilbar ist.

Automatisch festgelegt werden an dieser Stelle zudem:

- Die im Training angewandte Loss-Funktion. Im Falle der *Cuboid Splatting* Implementierung wird der Mean Squared Error (MSE) Loss verwendet. Anhand der Loss-Funktion wird die Abweichung zwischen den echten und den berechneten Belastungsmesswerten am Ende eines Pfades bestimmt und damit der aktuelle „Fehler“ des Modells festgestellt, der im Trainingsverlauf minimiert werden soll.
- Das verwendete Gradientenverfahren, über das der Fehler des Modells minimiert werden soll. Im Falle unserer Methode wird der Adam (Adaptive Moment Estimation) Optimizer verwendet.
- Der Learning Rate Scheduler, der dafür sorgt, dass die Learning Rate im Laufe des Trainings angepasst wird. Hier wird ReduceLROnPlateau verwendet, der die Learning Rate verringert, wenn der Loss innerhalb einer bestimmten Anzahl von Epochen nicht mehr gesunken ist. Die genauen Parameter des Learning Rate Schedulers (*factor, patience, cooldown, threshold*), werden im Config-File der Methode *Cuboid Splatting* festgelegt.

4.3.5.6 Der Trainingsprozess des Cuboid-Modells - Durchführung

Nach der Festlegung der Rahmenbedingungen des Trainings wird der eigentliche Trainingsprozess angestoßen.

Dabei wird über die Anzahl der durchzuführenden Trainingsepochen iteriert. In jeder Epoche wird der beschriebene Forward-Pass auf die einzelnen Batches an Trainingsdaten angewandt. Damit wird für jeden der Forward-Methode übergebenen Pfad der Pfadbelastungswert berechnet, den dieser Pfad nach dem aktuellen Stand des Modells (der Rekonstruktion) haben müsste. Diese berechneten Pfadbelastungswerte werden im nächsten Schritt mit den realen

Pfadbelastungswerten (Schilddrüsen-Messung am eine eines Pfades) verglichen, in dem die zuvor definierte Loss-Funktion (Mean Squared Error), darauf angewandt wird. So wird der Loss-Wert, also der „Fehler“ bezüglich dieses Batches der Pfaddaten bestimmt.

Im Anschluss folgt der sogenannte Backward-Pass. Hierbei wird der Gradient der Loss-Funktion am aktuellen Loss-Wert bestimmt. Der Gradient ist die „mehrdimensionale Ableitung“ der Funktion an dieser Stelle. Die Ableitung erfolgt dabei auf Basis der trainierbaren Parameter des Cuboid-Modells (die 3D-Koordinaten der Mittelpunkte, Längen, Breiten, Höhen und Belastungswerte der Cuboids), da dies die Parameter sind, die im Trainingsprozess optimiert werden. Da der Loss-Wert (der Fehler des Modells) minimiert werden soll, wird nun ein Schritt, in der Größe der aktuellen Learning Rate, in Richtung des negativen Gradienten der Loss-Funktion ausgehend vom aktuellen Loss-Wert vollzogen und die trainierbaren Parameter des Cuboid-Modells daraufhin angepasst. Der eben skizzierte Ablauf wird für alle Batches von Pfaddaten einer Epoche wiederholt.

Anschließend an das Durchlaufen aller Pfaddaten innerhalb der Epoche wird der aktuelle Stand des Modells evaluiert. Hierzu wird der aktuelle Loss (Fehler) des Modells bezogen auf alle Pfaddaten berechnet. Zum Vergleich: Vorher wurde immer nur der Loss pro Batch der Pfaddaten berechnet. Es wird nun also für alle Pfade die Abweichung zwischen dem realen Pfadbelastungswert (Schilddrüsen-Messung) und dem berechneten Pfadbelastungswert nach dem aktuellen Stand des Modells bestimmt. Ist dieser Fehler-Wert besser, als alle zuvor im Training des Modells erreichten Loss-Werte, so wird der aktuelle Stand des Modells als bislang bestes erreichtes Modell gespeichert. Sollte der Loss bereits seit mehreren Epochen stagnieren oder sogar schlechter geworden sein, wird mithilfe des Learning Rate Schedulers die aktuelle Learning Rate verringert, um in zukünftigen Epochen wieder bessere Loss-Werte zu erreichen.

Alle acht Epochen erfolgt außerdem das sogenannte Adaptive Density Control Verfahren, das in einem separaten Unterkapitel (4.3.5.7) beschrieben wird.

Zur Vermeidung von Schwierigkeiten im Trainingsprozess werden zwei zusätzliche Maßnahmen ergriffen:

- Um Probleme mit einem sich füllenden GPU-Speicher zu vermeiden, wird alle zehn Epochen der Cache der CUDA geleert.
- Damit keine negativen Werte bei den Belastungswerten und den Längen, Breiten und Höhen der Cuboids entstehen, werden hiervon jeweils nur quadrierte Werte in den Berechnungsprozessen verwendet. Während die zugrunde liegenden Werte also negativ werden können, wird so effektiv sichergestellt, dass in den Modellberechnungen und somit auch in der entstehenden Rekonstruktion immer nur positive Werte vorkommen können.

Nach Abschluss aller Epochen wird dann das Modell aus der Epoche mit dem geringsten Rekonstruktionsfehler (Loss-Wert) als finales Modell bestimmt und dessen gespeicherte Rekonstruktion der Belastungswerte in der Raumzeit entnommen und als Output zurückgegeben. In der Regel handelt es sich beim besten Modell, um das Modell aus einer der letzten Epochen. Manchmal verläuft ein Training jedoch auch so, dass es für das Endergebnis besser ist, ein Modell aus einer früheren Epoche zu wählen.

4.3.5.7 Das Adaptive Density Control Verfahren

Wie die Original-Methode Gaussian Splatting (Kerbl, et al., 2023) umfasst auch unsere *Cuboid Splatting* Implementierung ein Adaptive Density Control Verfahren. Das bedeutet, dass während

des Trainingsprozesses auch die Anzahl der Cuboids optimiert wird. Zum einen werden dabei Cuboids gelöscht, die einen sehr geringen Belastungswert aufweisen, da diese quasi „leere“ Luft darstellen. Zum anderen werden zusätzliche Cuboids entlang von bislang schlecht rekonstruierten Pfaden eingestreut.

Die Löschung leerer Cuboids: Alle acht Epochen werden die aktuellen Belastungswerte der Cuboids des Modells mit einem Grenzwert verglichen. (beispielsweise 0.001, der Grenzwert kann angepasst werden). Die Cuboids, deren Belastungswerte unterhalb dieses Grenzwerts liegen, werden aus dem Modell gelöscht, da diese nahezu unbelastete Luft darstellen, welche nicht durch Cuboids abgedeckt sein muss. Die übrigen Cuboids werden im Modell beibehalten.

Das Einstreuen neuer Cuboids: Für die Ergänzung weiterer Cuboids im Modell wurden mehrere Varianten implementiert, die von komplett zufälligem Einstreuen bis hin zu exakt gezieltem Einstreuen neuer Cuboids in das Modell reichen. Für gezieltes Einstreuen werden die Pfade mit der größten Unterrekonstruktion (größte positive Differenz zwischen dem realen und dem berechneten Belastungswert eines Pfades) herangezogen. Hierzu wurden mehrere Funktionen definiert, die neue Startpositionen für Cuboids im Raum erzeugen können. Die im Trainingsprozess angewandte Variante (*add_option*) kann in der Config-Datei der Methode konfiguriert werden.

- Option *random*: hier wird auf dieselbe Funktion zurückgegriffen wie bei der Initialisierung des Cuboid-Modells und es werden fünf neue Cuboids zufällig in die Raumzeit eingestreut.
- Option *specific_a*: Diese Variante ermittelt die 20 Pfade mit der größten Unterrekonstruktion. Aus diesen Pfaden werden dann die fünf häufigsten Rasterzellen bestimmt, die von diesen Pfaden durchlaufen werden. In diese fünf häufigsten Rasterzellen werden mittig neue Cuboids platziert.
- Option *specific_b*: Diese Variante ermittelt die 10 Pfade mit der größten Unterrekonstruktion. Aus diesen Pfaden werden dann zufällig fünf Rasterzellen ausgewählt und in diese Rasterzellen werden mittig neue Cuboids platziert.
- Option *specific_c*: Diese Variante ermittelt die 20 Pfade mit der größten Unterrekonstruktion. Aus diesen Pfaden werden die fünfzig häufigsten Rasterzellen bestimmt, die von diesen Pfaden durchlaufen werden. Aus diesen häufigsten Rasterzellen werden dann zufällig fünf Rasterzellen ausgewählt und in diese werden mittig neue Cuboids platziert.

Bei der Evaluation der Methode *Cuboid Splatting* wurde von uns am häufigsten die Option *specific_b* eingesetzt. Das Adaptive Density Control Verfahren führt unserer Beobachtung nach in der Regel zunächst zu einem Anstieg der Anzahl der Cuboids im Modell im Vergleich zur Anzahl bei der Initialisierung. Nach einigen Trainingsepochen pendelt sich diese Zahl dann um einen bestimmten Wert ein.

4.3.5.8 Einstellung der Hyperparameter der Methode *Cuboid Splatting*

Wie in den vorherigen Kapiteln beschrieben, müssen einige Parameter der *Cuboid Splatting* Methode über die Config-Datei des Modells je nach Art des Runs (bspw. je nach Auflösung, Anzahl der Pfaddaten, Szenario) angepasst werden. Bei einigen dieser Parameter handelt es sich um sogenannte Hyperparameter, deren Optimierung zu besseren Ergebnissen in der Rekonstruktion führen kann. Aufgrund des zeitlichen Umfangs einzelner Runs (insbesondere in höheren Auflösungen mehrere Stunden pro Run), ist es nicht möglich, ein umfangreichen Grid Search und Cross Validation Prozess dafür aufzusetzen, wie es sonst im Bereich Machine Learning gemacht wird. Stattdessen können nur einzelne Hyperparameter-Anpassungen getestet und ausgewertet werden. Es ist zudem sinnvoll, bei der Durchführung neuer Runs auf die Erfahrungen aus bisher

durchgeführten, ähnlichen Runs zurückzugreifen (Ähnlichkeit bezüglich Auflösung, Pfadanzahl und realistisches oder künstliches Szenario). Die Hyperparameter, die wir im Training der Modelle hauptsächlich optimiert haben (nach Priorität absteigend), sind die initiale Learning Rate (*initial_lr*), die Batch Größe (*batch_size*), Anzahl der Epochen (*num_epochs*), die Parameter des Learning Rate Schedulers (*factor, patience, threshold*), die initiale Größe der Cuboids (*initial_length_width_height*), die Anzahl der Cuboids zu Beginn des Trainings (*nr_of_cuboids*) und die Variante des Adaptive Density Control (*add_option*). Diese Hyperparameter sollten auch für zukünftige Runs an die Rahmenbedingungen des jeweiligen Runs über die Config-Datei angepasst werden.

Die Konfiguration der Parameter der Methode *Cuboid Splatting* wurde in den vorangegangenen Abschnitten erläutert. Zur Orientierung noch einmal das Format der benötigten JSON-Datei mit allen konfigurierbaren Parametern:

```
{
    "cuboid_model": {
        "nr_of_cuboids": 75,
        "grid_size": [10, 10, 116],
        "space_range_xy": [0, 10],
        "space_range_z": [0, 116],
        "initial radiation": 0.1,
        "initial_length_width_height": 2.5
    },
    "training": {
        "initial_lr": 0.05,
        "batch_size": 100,
        "num_epochs": 300,
        "scheduler": {
            "factor": 0.5,
            "patience": 2,
            "cooldown": 3,
            "threshold": 0.00001
        }
    },
    "adaptive_density_control": {
        "add_option": "specific_b"
    },
    "plots": {
        "filter": 5,
        "timesteps": 120
    },
    "seed": 3,
    "torch_seed": 1,
    "device": "cuda"
}
```

4.4 Ergebnisse der Experimente mit *Cuboid Splatting* und *Inverse Rendering*

In diesem Kapitel werden die Ergebnisse der Experimente mit den Methoden *Cuboid Splatting* und *Inverse Rendering* präsentiert. Eine genauere Aufschlüsselung der erzielten Ergebnisse mit den beiden Methoden findet sich in den Tabellen 4.6 bis 4.11. Darin werden die wichtigsten Qualitätsmetriken der erlangten Rekonstruktionen mit den beiden Methoden präsentiert. Dabei handelt es sich zum einen um den prozentualen Szenariofehler bezüglich Raumzeitpunkten, durch die mindestens einer, fünf oder zehn Pfade verlaufen (% scen 1 rel sum, % scen 5 rel sum, % scen 10 rel sum) und den prozentualen Pfadfehler auf den Trainingspfaden (% train rel sum) und auf einem dem Modell unbekannten Set aus Testpfaden (% test rel sum). Genauere Erklärungen bezüglich der Qualitätsmetriken finden sich in Abschnitt 4.2.

Es liegt je eine Tabelle pro Methode und räumlicher Auflösung der Rekonstruktionen vor (Tabelle 4.6 und 4.7: Auflösung 10x10, 4.8 und 4.9: 30x30, 4.10 und 4.11: 100x100) in der die Ergebnisse für verschiedene Szenarien und Anzahlen von Bewegungsprofilen (Pfaden) gezeigt werden. Diese Ergebnisse werden in den folgenden Unterkapiteln ausführlich ausgewertet (4.4.1 bis 4.4.5 und 4.5.1 bis 4.5.9).

Die Methode *Cuboid Splatting* hat in unseren Experimenten zu sehr vielversprechenden Ergebnissen geführt. Diese Methode wurde vorrangig auf dem Straßennetz Ortschaften getestet. Einige Untersuchungen auf den drei verschiedenen Straßennetzen (Hauptstraßen, Ortschaften, Platz) zeigten, dass in allen Fällen ähnliche Muster bezüglich der Auswirkungen der verschiedenen Dimensionen der Experimentparameter (Anzahl der Pfade, Szenario, Auflösung) auf die Qualität der Ergebnisse vorliegen. Die Ergebnisse zusätzlicher Experimente bezüglich des Einflusses von Warnzonen finden sich in Unterkapitel 4.5.5. Für die Methode *Cuboid Splatting* ist bislang keine Nutzung eines Priors vorgesehen, weshalb auch keine Experiment-Ergebnisse diesbezüglich vorliegen.

Auch die Methode *Inverse Rendering* hat in unseren Experimenten zu interessanten Ergebnissen geführt. In manchen Fällen ist die erreichte Rekonstruktions-Qualität darin sogar höher als mit der Methode *Cuboid Splatting*, in den meisten Fällen ist die Methode *Inverse Rendering* qualitativ aber unterlegen. Die in diesem Kapitel dargestellten Ergebnis-Tabellen enthalten der zunächst nur die Ergebnisse auf dem Straßennetz Ortschaften, ohne Warnzonen und ohne Prior-Nutzung. Zusätzlich zu den hier gezeigten Ergebnissen, wurden für die Methode *Inverse Rendering* aber auch der Einfluss der verschiedenen Straßennetze, der Einfluss von Warnzonen und der Einfluss der Nutzung von Prioren ausführlich getestet. Die Ergebnisse dazu werden der Übersichtlichkeit halber erst in den nachfolgenden Unterkapiteln (4.5.4, 4.5.5, 4.5.6) gezeigt und analysiert.

4.4.1 Zeit- und Speicherbedarf der Experimente

In diesem Kapitel werden wir den Speicher- und Zeitbedarf der Methoden *Inverses Rendering* und *Cuboid Splatting* aus. Da wir die Methoden jeweils auf Rechnern evaluiert haben, die den in Kapitel 1.4 genannten Anforderungen an einen Office-PC sehr ähnlich sind, können wir zusammenfassend berichten, dass eine Rekonstruktion mit diesen Methoden unter der gegebenen Hardware lauffähig ist. Es gilt jedoch zu beachten, dass in unseren Laptops, wie in Kapitel 4.1.3 beschrieben, jeweils über eine moderne GPU verfügen, die wir in allen Trainingsdurchläufen genutzt haben. Die Verwendung einer GPU bringt eine Beschleunigung der Berechnungen bei gleichzeitig weniger verfügbarem Speicher mit sich.

Inverse Rendering: Die Laufzeiten der Durchläufe der Methode *Inverse Rendering* bewegen sich in einem Rahmen von wenigen Stunden bis zu einem Tag. Sie sind sowohl von der Anzahl der

verwendeten Bewegungsprofile als auch von der Auflösung abhängig. Tabelle 4.5 zeigt einige Laufzeiten für verschiedene Parameterkombinationen. Für eine schnelle erste Rekonstruktion, die mit wenigen Daten in kurzer Zeit erfolgen soll, empfehlen wir eine geringere räumliche Auflösung, wie zum Beispiel 10x10.

Tabelle 4.5: Darstellung der Laufzeit für verschiedene Parameterkombinationen bei Durchläufen

Auflösung	**Pfadanzahl**	**Epochen**	**Laufzeit (gerundet)**
10x10	1.000	10.000	20 Minuten
10x10	10.000	10.000	3,5 Stunden
10x10	25.000	10.000	9 Stunden
30x30	25.000	10.000	9,5 Stunden
100x100	50.000	5.000	15 Stunden
100x100	100.000	5.000	23 Stunden

Der Speicherbedarf hat sich in unseren Durchläufen nicht als Problem herausgestellt. Der Hauptteil der Rekonstruktion lief auf der GPU, die nur über einen Speicher von 4 GB verfügt. Damit ist die Anforderung von 16 GB erfüllt.

Cuboid Splatting: Bezüglich des Zeitaufwands bewegen sich die Modell-Trainings mit der Methode *Cuboid Splatting* in einer räumlichen Auflösung von 10x10 im Rahmen von bis zu 120 Minuten, während die Rekonstruktionen in höheren Auflösungen (30x30 und 100x100) über mehrere Stunden laufen. Hierbei steigt die Laufzeit dann vor allem mit der genutzten Anzahl an Trainingspfaden. Bei einer hohen Menge an Trainingspfaden (100.000) läuft ein hochaufgelöster Run dann sogar bis zu 36 Stunden. Diese Laufzeiten können mit der genutzten Hardware variieren und sind daher nur als Orientierungswerte zu sehen. Bezüglich des Speicherbedarfs der Modell-Trainings bei einer vorhandenen Laptop-GPU mit 6 GB RAM ließen sich alle Rekonstruktionen der Auflösungen 10x10 und 30x30 problemlos berechnen, bei den Rekonstruktionen in der Auflösung 100x100 stießen die Berechnungen teilweise an Speichergrenzen. Diese Einschränkungen sollten sich bei Bedarf durch Hardware-Anpassungen lösen lassen.

4.4.2 Erreichbare Rekonstruktionsqualität in der Auflösung 10x10

Die zu erreichende Qualität der Rekonstruktionen mit der Methode *Cuboid Splatting* liegt in der räumlichen Auflösung 10x10 bei den realistischen Szenarien bei einem prozentualen Szenariofehler von 16,0% beim Szenario Emsland und 10,4% beim Szenario Gundremmingen. Bezüglich der künstlichen Szenarien liegt die bislang erreichbare Qualität der Rekonstruktion bezogen auf den Szenariofehler beim Gaussian-Szenario bei 13,2% und beim Box-Szenario sogar bei 0,04%. Die besten Ergebnisse wurden hierbei immer bei Rekonstruktionen mit der größten Anzahl an Trainingspfaden (in der 10x10 Auflösung: 50.000 Pfade) erreicht. Der prozentuale Rekonstruktionsfehler bezüglich der rekonstruierten Belastungen als Schilddrüsenmesswert am Ende eines Pfades (Pfadfehler) liegt sogar noch wesentlich niedriger. Beim Emsland-Szenario ist in der Auflösung 10x10 eine Rekonstruktion der Testpfade (Pfade, die nicht im Modelltraining verwendet wurden, sondern dem Modell bis dahin unbekannt sind) mit einem prozentualen Fehler von nur 3,2% möglich. Der prozentuale Rekonstruktionsfehler der Testpfade liegt für die anderen Szenarien sogar noch geringer, bei: 0,3% im Gundremmingen-Szenario, 1,7% im Gaussian-Szenario und 0,003% im Box-Szenario. Darüber hinaus ist anzumerken, dass der

erreichte Rekonstruktionsfehler auf den Testpfaden in allen vier Szenarien in der Auflösung 10x10 nur unwesentlich vom erreichten Rekonstruktionsfehler auf den Trainingspfaden abweicht. Dies spricht für eine gute Generalisierbarkeit des erreichten Modells und gegen das Vorliegen eines sogenannten Overfitting-Problems, wie es mit der Methode *Inverse Rendering* häufig auftritt. Dieses Resultat bewerten wir als schlüssig, da mit der Methode *Cuboid Splatting* übergeordnete Strukturen der Belastungsverteilung in der dreidimensionalen Raumzeit gelernt werden können, während mit der Methode *Inverse Rendering* direkt die Belastungen einzelner Raumzeitpunkte trainiert werden, was ein größeres Risiko zu Overfitting (quasi das „Auswendiglernen“ der Trainingspfade durch das Modell, anstelle der Erschaffung eines über die Trainingspfade hinaus generalisierbaren Modells) mit sich bringt.

*Tabelle 4.6: Ergebnisse der Methode **Cuboid Splatting** in der Auflösung 10x10 auf dem Ortschaften-Graphen*

ID	Szenario	Pfade	% scen 1 rel sum	% scen 5 rel sum	% scen 10 rel sum	% train rel sum	% test rel sum
1014	Emsland	5k	37.9401	35.22	42.0136	4.23	17.8677
1005	Emsland	10k	21.1235	18.8305	19.1834	4.798	6.0893
1018	Emsland	25k	17.7439	15.804	15.0946	2.7523	3.6999
1001	Emsland	50k	16.0123	14.5396	14.0389	2.78	3.2471
1015	Box	5k	0.801	0.6886	0.5429	0.0293	0.0327
1006	Box	10k	0.5179	0.4766	0.4258	0.0196	0.0223
1019	Box	25k	0.0611	0.0531	0.0507	0.0042	0.0048
1002	Box	50k	0.0412	0.0381	0.0369	0.003	0.003
1020	Gaussian	5k	23.2771	22.6724	23.1755	2.1571	4.5622
1021	Gaussian	10k	18.7118	17.5243	17.7617	2.2786	3.0326
1022	Gaussian	25k	22.9401	21.6386	20.6265	3.4442	3.9545
1023	Gaussian	50k	13.2124	12.7824	12.6523	1.8142	1.7387
1016	Gundremmingen	5k	62.3331	58.5216	61.2783	5.0679	5.1094
1004	Gundremmingen	10k	62.1304	56.1842	54.4308	4.1495	3.5736
1017	Gundremmingen	25k	33.2565	32.6601	31.9465	3.7104	3.3618
1000	Gundremmingen	50k	10.4444	10.4409	10.4348	0.2368	0.3041

Mit der Methode *Inverse Rendering* können in der räumlichen Auflösung 10x10 bei den realistischen Szenarien unter Verwendung einer hohen Menge von Pfaden ebenfalls sehr gute Rekonstruktionsergebnisse erreicht werden. Die in den Experimenten beste erreichte Qualität liegt hier bei einem prozentualen Szenariofehler von 13,2% beim Emsland-Szenario und 19,0% beim Gundremmingen-Szenario. Das künstliche Szenario Box kann ebenfalls gut rekonstruiert werden (Szenariofehler: 10,0%), während das künstliche Gaussian-Szenario nicht gut rekonstruiert wird (Szenariofehler: 79,5%). Auffällig in den Ergebnissen der Methode *Inverse Rendering* in dieser Auflösung sind auch die sehr guten Pfadrekonstruktions-Ergebnisse. Bei einer Verwendung ausreichend vieler Trainingspfade, liegt der prozentuale Fehler auf den Testpfaden bei dieser Methode für die Szenarien Emsland, Box und Gundremmingen bei weniger als 1%. Während dies bei der Methode *Cuboid Splatting* weniger stark zu erkennen war, sieht man in den Ergebnissen zu *Inverse Rendering* deutlich, dass Punkte mit steigender Relevanz (Punkte, durch die mehr Pfade geführt haben) im Durchschnitt wesentlich besser rekonstruiert werden können (in der Regel: *% scen 1 rel sum* < *% scen 5 rel sum* < *% scen 10 rel sum*). Die Ergebnisse zeigen

des Weiteren sehr deutlich die Tendenz der Methode *Inverse Rendering* zu Overfitting, wenn die Menge der verwendeten Trainingspfade zu gering ist. Der prozentuale Rekonstruktionsfehler auf den Trainingspfaden liegt bei den Ergebnissen dieser Methode nahezu immer geringer als 0,1%. Es gelingt der Methode *Inverse Rendering* also immer, ein zugrundeliegendes Modell der Belastungsverteilung in der Raumzeit zu finden, das die Belastungswerte aller Trainingspfade gut erklären kann. Gleichzeitig ist erkennbar, dass der prozentuale Rekonstruktionsfehler auf Testpfaden (Pfade, die nicht Teil des Modelltrainings waren und dem Modell bis dahin unbekannt sind) häufig wesentlich höher als der Trainingspfad-Fehler ist. Dies bedeutet, dass sich das gefundene Modell nicht gut generalisieren lässt und ein Overfitting auf die Trainingsdaten vorliegt. Anstelle des Lernens einer generalisierbaren Szenario-Rekonstruktion wurden dann also nur die Trainingspfade „auswendig gelernt". Die vorliegenden Ergebnisse zeigen, dass dieses Overfitting-Problem dann überwunden wird, wenn eine ausreichend hohe Menge an Trainingspfaden genutzt wird.

*Tabelle 4.7: Ergebnisse der Methode **Inverse Rendering** in der Auflösung 10x10 auf dem Ortschaften-Graphen*

ID	Szenario	Pfade	% scen 1 rel sum	% scen 5 rel sum	% scen 10 rel sum	% train rel sum	% test rel sum
3	Emsland	1k	87.4467	63.8475	89.8442	0.0287	70.3606
4	Emsland	5k	91.1422	66.5488	63.1631	0.0297	29.2149
5	Emsland	10k	72.8932	49.2621	44.0634	0.0185	6.8187
2000	Emsland	25k	32.9829	24.0683	21.212	0.0528	2.155
2001	Emsland	50k	17.3135	11.5506	9.7652	0.0235	3.4996
2002	Emsland	100k	13.2083	9.6572	8.051	0.0061	0.0473
51	Box	1k	33.6871	19.8418	12.4044	0.0021	3.1309
52	Box	5k	21.4381	17.8651	13.0198	0.002	0.2067
53	Box	10k	16.4794	15.0146	13.6631	0.0006	0.4662
5015	Box	25k	10.0041	9.485	9.0178	0.0008	0.061
5009	Gaussian	5k	103.3026	74.5743	69.0654	0.0084	10.412
5011	Gaussian	10k	79.458	60.615	58.5262	0.0123	7.4502
5018	Gaussian	25k	81.2373	73.2387	70.863	0.1127	4.9646
5006	Gundremmingen	5k	21.3321	17.3036	6.4694	0.0036	0.0906
87	Gundremmingen	10k	32.9417	26.4365	19.3111	0.0012	0.1214
5007	Gundremmingen	25k	19.0329	19.0333	16.8176	0.0022	0.0196

4.4.3 Erreichbare Rekonstruktionsqualität in der Auflösung 30x30

Je höher die räumliche Auflösung der gewünschten Rekonstruktion mit der Methode *Cuboid Splatting* gewählt wird, desto schwieriger ist es, eine gute Qualität bezüglich der genannten Metriken zu erreichen. In der räumlichen Auflösung 30x30 (also einer neunmal so hohen Auflösung bezüglich der zu rekonstruierenden Raumzeitzellen verglichen mit 10x10), liegen die bislang erreichten prozentualen Szenariofehler bei 25,1% für das Emsland-Szenario und 50,7% für das Gundremmingen-Szenario. Für die beiden künstlichen Szenarien liegt der erreichte Szenariofehler bei 22,9% für das Gaussian-Szenario und 0,05% für das Box-Szenario. Die erreichte Rekonstruktionsqualität der Testpfade liegt bei beiden realistischen Szenarien auch hier deutlich niedriger als der Szenariofehler: bei etwa 3% (Emsland: 3,4% und Gundremmingen 3,0%). Bei den künstlichen Szenarien liegt der Testpfad-Fehler bei 5,0% für das Gaussian-Szenario und 0,01% beim Box-Szenario. Auch hier sind, vorausgesetzt der Verwendung einer hohen Menge an Trainingspfaden (100.000 bei den realistischen Szenarien, 25.000 bei den künstlichen Szenarien), keine wesentlichen Unterschiede zwischen der prozentualen Abweichung auf Trainingspfaden und auf Testpfaden zu erkennen, was weiterhin für eine gute Generalisierbarkeit des entstandenen Modells spricht.

*Tabelle 4.8: Ergebnisse der Methode **Cuboid Splatting** in der Auflösung 30x30 auf dem Ortschaften-Graphen*

ID	Szenario	Pfade	% scen 1 rel sum	% scen 5 rel sum	% scen 10 rel sum	% train rel sum	% test rel sum
1024	Emsland	5k	125.0413	154.3511	174.2978	4.6652	104.0152
1028	Emsland	10k	82.5425	82.6863	73.4593	2.7137	29.4117
1032	Emsland	25k	40.1371	50.5235	53.3516	6.3673	20.8333
1036	Emsland	100k	25.1294	18.2885	18.4983	3.6984	3.3716
1026	Box	5k	0.2465	0.1446	0.1002	0.0361	0.0315
1030	Box	10k	0.0449	0.0377	0.0312	0.0072	0.0075
1034	Box	25k	0.0496	0.0297	0.0172	0.0086	0.01
1025	Gaussian	5k	35.7978	32.7954	29.0605	1.7664	8.197
1063	Gaussian	10k	30.0209	28.2264	29.3555	4.022	6.3699
1062	Gaussian	25k	22.9586	20.4519	19.3893	3.3057	5.0044
1077	Gundremmingen	5k	328.7224	236.821	211.5975	56.8621	54.4282
1076	Gundremmingen	10k	109.8267	81.0751	79.0415	11.7515	17.5124
1075	Gundremmingen	25k	67.1407	56.1473	51.5129	6.9679	9.3589
1374	Gundremmingen	100k	50.7157	48.8899	47.8563	3.2611	3.0117

Die Ergebnisse mit der Methode *Inverse Rendering* in der räumlichen Auflösung 30x30 fallen schwächer aus als mit der Methode *Cuboid Splatting*. Der beste erreichte prozentuale Szenariofehler für das Emsland-Szenario liegt bei 53,5% unter der Verwendung der höchsten Pfadanzahl von 100.000 Pfaden. Die Szenarien Box und Gundremmingen mit je 23,3% und 32,4% Szenariofehler rekonstruiert werden. Der prozentuale Szenariofehler beim Gaussian-Szenario liegt über 100%. Hier wird also das Gundremmingen-Szenario mit *Inverse Rendering* besser rekonstruiert als mit *Cuboid Splatting*, aber das Emsland-Szenario, sowie Box- und Gaussian-Szenario deutlich schlechter. Insgesamt ist auch wieder erkennbar, dass die

Rekonstruktionsergebnisse mit der Methode *Inverse Rendering* mit steigender Trainingspfadanzahl besser werden und dass auch hier, anders als bei der Methode *Cuboid Splatting*, Raumzeitpunkte mit höherer Relevanz (Punkte, durch die mind. 5 oder 10 Pfade führen) wesentlich besser rekonstruiert werden als Raumzeitpunkte mit niedrigerer Relevanz (Punkte, durch die mind. 1 Pfad führt). Des Weiteren wird anhand der Ergebnisse wieder die Tendenz der Methode *Inverse Rendering* zu Overfitting deutlich. Während der prozentuale Rekonstruktionsfehler auf den Trainingspfaden wieder überall unter 1% (meist sogar unter 0,1%) liegt, liegt der Testpfadfehler in den Experimenten deutlich höher. Das spricht für eine mangelnde Generalisierbarkeit der entstandenen Modelle (ausführlichere Erklärung im vorherigen Abschnitt) und dafür, dass immer möglichst viele Trainingspfade verwendet werden sollten, um dem Overfitting-Problem so weit wie möglich entgegenzuwirken.

*Tabelle 4.9: Ergebnisse der Methode **Inverse Rendering** in der Auflösung 30x30 auf dem Ortschaften-Graphen*

ID	Szenario	Pfade	% scen 1 rel sum	% scen 5 rel sum	% scen 10 rel sum	% train rel sum	% test rel sum
111	Emsland	1k	127.3025	313.0893	100.0002	0.0002	179.443
112	Emsland	5k	121.244	154.5268	152.3368	0.0306	282.0479
113	Emsland	10k	118.2228	154.3735	167.6143	0.0244	117.2218
2006	Emsland	25k	102.8315	105.2734	101.792	0.2116	122.0175
2007	Emsland	50k	90.6708	71.0087	66.6543	0.4355	56.5589
2008	Emsland	100k	53.4832	41.2779	35.1363	0.3251	23.8841
117	Box	1k	33.9436	20.5129	14.408	0.0008	15.6845
118	Box	5k	29.8115	21.1828	14.22	0.0025	1.831
119	Box	10k	23.3191	18.1072	14.7576	0.0054	1.9106
120	Gaussian	5k	118.5921	87.1839	74.8429	0.0017	31.6273
121	Gaussian	10k	116.4561	79.1446	71.7459	0.0028	29.5389
89	Gundremmingen	10k	32.4279	11.4585	8.9807	0.0759	1.3003

4.4.4 Erreichbare Rekonstruktionsqualität in der Auflösung 100x100

Im Rahmen der räumlichen Auflösung 100x100 lassen sich mit der Methode *Cuboid Splatting* bislang nur die beiden künstlichen Szenarien gut rekonstruieren. Hier werden beim Gaussian-Szenario 26,4% Szenario-Fehler und beim Box-Szenario 0,03% Szenario-Fehler erreicht. Bei den beiden realistischen Szenarien liegt der erreichte Szenariofehler der Rekonstruktion selbst bei der höchsten verwendeten Menge an Trainingspfaden (100.000 Pfade) bislang bei über 100%. Ein vorhandener Trainingseffekt ist bei den realistischen Szenarien hier nur anhand des Pfad-Fehlers zu erkennen. Die Rekonstruktion der Testpfade gelingt in der Auflösung 100x100 für das Emsland-Szenario mit einem prozentualen Fehler von 32,7% und für das Gundremmingen-Szenario mit 19,1%. Es ist also erkennbar, dass das Modell etwas „Sinnvolles“ im Trainingsprozess gelernt hat, mit dem es auch die Belastungswerte unbekannter Pfade in einem gewissen Maße rekonstruieren kann. Die Rekonstruktion der tatsächlichen Belastungswerte in der Raumzeit liegt hier aber deutlich entfernt von den realen Werten. In einzelnen Fällen (beispielsweise Run 1072 mit dem Emsland-Szenario) gab es in der 100x100 Auflösung auch mit der Methode *Cuboid Splatting* Probleme mit einem Overfitting des Modells auf die Trainingspfade (hier Trainingspfad-Fehler: 15,2%, Testpfad-Fehler: 47,3%), in den meisten Runs trat dies bei einer ausreichenden Menge an Trainingspfaden mit *Cuboid Splatting* aber auch in der 100x100 Auflösung nicht auf. Ein größeres Problem der Methode *Cuboid Splatting* in der 100x100 Auflösung ist die Speicherbegrenzung der GPU, wodurch einige Experimente bereits vor Erreichen ihrer geplanten Anzahl an Trainings-Epochen beendet werden mussten. Manche Trainingsparameter-Kombinationen waren nicht nutzbar, weil sie aufgrund der Speicherbegrenzung zu sehr schnellen Abbrüchen des Trainings führten.

*Tabelle 4.10: Ergebnisse der Methode **Cuboid Splatting** in der Auflösung 100x100 auf dem Ortschaften-Graphen*

ID	Szenario	Pfade	% scen 1 rel sum	% scen 5 rel sum	% scen 10 rel sum	% train rel sum	% test rel sum
1040	Emsland	5k	159.1895	198.8582	112.8033	43.8579	182.427
1044	Emsland	10k	158.2743	166.7886	142.3315	45.4439	45.5685
1048	Emsland	25k	145.3063	166.0051	157.4991	31.2892	89.7805
1052	Emsland	100k	121.0867	107.2308	108.342	29.8104	32.7498
1072	Emsland	100k	107.5218	79.475	80.4796	15.1576	47.341
1073	Emsland	100k	116.4937	110.3146	149.0826	37.3238	32.8884
1042	Box	5k	0.4956	0.3826	0.2937	0.1332	0.1291
1046	Box	10k	0.1808	0.1328	0.1132	0.0478	0.0438
1050	Box	25k	0.0261	0.0175	0.0134	0.0065	0.0072
1041	Gaussian	5k	33.8434	35.1908	40.5981	3.8089	10.7518
1045	Gaussian	10k	26.319	27.2789	36.3472	5.7803	6.1958
1049	Gaussian	25k	26.3823	26.7511	28.5811	6.4726	7.7874
1178	Gundremmingen	100k	111.788	100.5128	93.6168	22.3447	19.0849

Mit der Methode *Inverse Rendering* lässt sich in der räumlichen Auflösung 100x100 nur das Box-Szenario einigermaßen gut rekonstruieren (Szenariofehler: 32,9%), an den beiden realistischen Szenarien Emsland und Gundremmingen (Szenariofehler >100%) scheitert die Methode. Anders

als bei der Methode *Cuboid Splatting*, bei der zumindest anhand des Testpfad-Fehlers ein Lerneffekt des Modells zu erkennen war, ist dies bei der Methode *Inverse Rendering* in der Auflösung 100x100 nicht mehr der Fall. Hier liegt lediglich ein sehr starkes Overfitting auf die Trainingsdaten vor (Trainings-Pfadfehler <1%), welches gleichzeitig mit einem sehr hohen Testpfadfehler (beim Emsland-Szenario sogar immer >100%) einhergeht. Dies bedeutet, dass kein generalisierbares Modell entsteht. Auch ein höheres Relevanzlevel der Raumzeitpunkte bringt in dieser Auflösung bei den realistischen Szenarien keine Verbesserungen der Rekonstruktionsqualität mehr mit sich. Der einzige Vorteil gegenüber der Methode *Cuboid Splatting* ist hier, dass mit der Methode *Inverse Rendering* keine Probleme durch die Begrenztheit des GPU-Arbeitsspeichers aufgetreten sind.

*Tabelle 4.11: Ergebnisse der Methode **Inverse Rendering** in der Auflösung 100x100 auf dem Ortschaften-Graphen*

ID	Szenario	Pfade	% scen 1 rel sum	% scen 5 rel sum	% scen 10 rel sum	% train rel sum	% test rel sum
15	Emsland	1k	129.5128	100.0086	100.024	0.0	133.8899
16	Emsland	5k	132.2702	117.5666	116.6401	0.0762	279.0004
17	Emsland	10k	127.7402	110.0038	120.8819	0.0901	370.6709
2003	Emsland	25k	131.5654	151.3134	127.0186	0.6321	982.4975
2004	Emsland	50k	127.7882	112.2489	138.3196	0.5775	133.3623
2005	Emsland	100k	127.1958	110.071	154.0102	0.2569	4957.6907
63	Box	1k	38.2917	24.7904	12.9925	0.0004	42.6216
64	Box	5k	34.529	21.9118	15.0953	0.0022	10.2014
65	Box	10k	32.8768	22.7079	16.6426	0.0024	7.1682
88	Gundremmingen	10k	183.369	134.5591	132.1556	1.8679	78.7458

4.4.5 Zusammenfassung bezüglich der erreichbaren Rekonstruktionsqualität

Insgesamt lässt sich also schlussfolgern, dass die Methode *Cuboid Splatting* insbesondere in einer niedrigeren räumlichen Auflösung (10x10) sehr gute und zeitlich effiziente Rekonstruktions-Ergebnisse liefern kann. Auch in einer mittleren räumlichen Auflösung (30x30) können, bei einer ausreichenden Menge an Trainingspfaden, noch gute Ergebnisse erzielt werden. Bei hoher räumlicher Auflösung (100x100) stößt allerdings bislang auch diese Methode an ihre Grenzen, sowohl in Bezug auf die erreichbare Qualität der Rekonstruktion von Szenarien und Pfad-Belastungen, als auch in Bezug auf einen sehr hohen benötigten Speicher. Des Weiteren fallen insgesamt die prozentualen Abweichungen der rekonstruierten Pfad-Belastungswerte besser aus, als die prozentualen Abweichungen der rekonstruierten Raumzeit-Belastungswerte.

Bezüglich der Methode *Inverse Rendering* ergibt sich, dass zumindest in der räumlichen Auflösung 10x10 ebenfalls sehr gute Ergebnisse möglich sind. Insgesamt ist die Methode aber, sowohl bezüglich ihrer zeitlichen Performance als auch bezüglich der mit ihr erreichten Rekonstruktions-Qualitäten insbesondere bei steigender Auflösung, der Methode *Cuboid Splatting* unterlegen. Dies liegt vor allem an der Neigung der Methode zu Overfitting und dem dadurch entstehenden Mangel an Generalisierbarkeit des gelernten Modells.

4.5 Genauere Betrachtung einzelner Aspekte der Ergebnisse

In diesem Abschnitt werden einzelne Aspekte der Ergebnisse der Experimente mit den Methoden *Cuboid Splatting* und *Inverse Rendering* genauer betrachtet. Dabei werden zum einen die Ergebnisse nach den Dimensionen der Experimentparameter (Anzahl der Bewegungsprofile, Räumliche Auflösung der Rekonstruktion, verschiedene realistische und künstliche Szenarien, verschiedene Straßennetze, Einfluss von Warnzonen, Einfluss von Prioren) aufgeschlüsselt. Zum anderen werden weitere übergeordnete Aspekte (Einfluss der Relevanz und Überlappung, Detailbetrachtungen zu den Rekonstruktionen, Verlauf des Trainings) genauer analysiert.

4.5.1 Anzahl der Bewegungsprofile (Pfade)

Cuboid Splatting: Die Ergebnisse der Experimente mit der Methode *Cuboid Splatting* zeigen deutlich, dass eine höhere Anzahl verwendeter Pfade zu besseren Rekonstruktionen führt. Eine Visualisierung dieses Zusammenhangs findet sich in Abbildung 4.3. Darin wird die Anzahl der zur Rekonstruktion verwendeten Pfade (auf der x-Achse) dem prozentualen Szenario- und Testpfadfehler (auf der y-Achse links und rechts) gegenübergestellt. Wie viele Trainingspfade mindestens nötig sind, hängt vom Szenario, der gewählten Auflösung und den Anforderungen an die Qualität der Szenario-Rekonstruktion und der Pfad-Rekonstruktionen ab. Während bei einer räumlichen Auflösung von 10x10 bereits Runs mit 5.000 oder 10.000 Pfaden zu nutzbaren Ergebnissen führen können, werden bei einer räumlichen Auflösung von 30x30 die Szenario-Rekonstruktionen der beiden realistischen Szenarien erst ab einer Pfadmenge von 25.000 Pfaden brauchbar. Durchweg wurden die besten Rekonstruktions-Ergebnisse mit der höchsten verwendeten Pfadanzahl erreicht. Diese lag bei 50.000 Pfaden in der Auflösung 10x10 und 100.000 Pfaden in den Auflösungen 30x30 und 100x100.

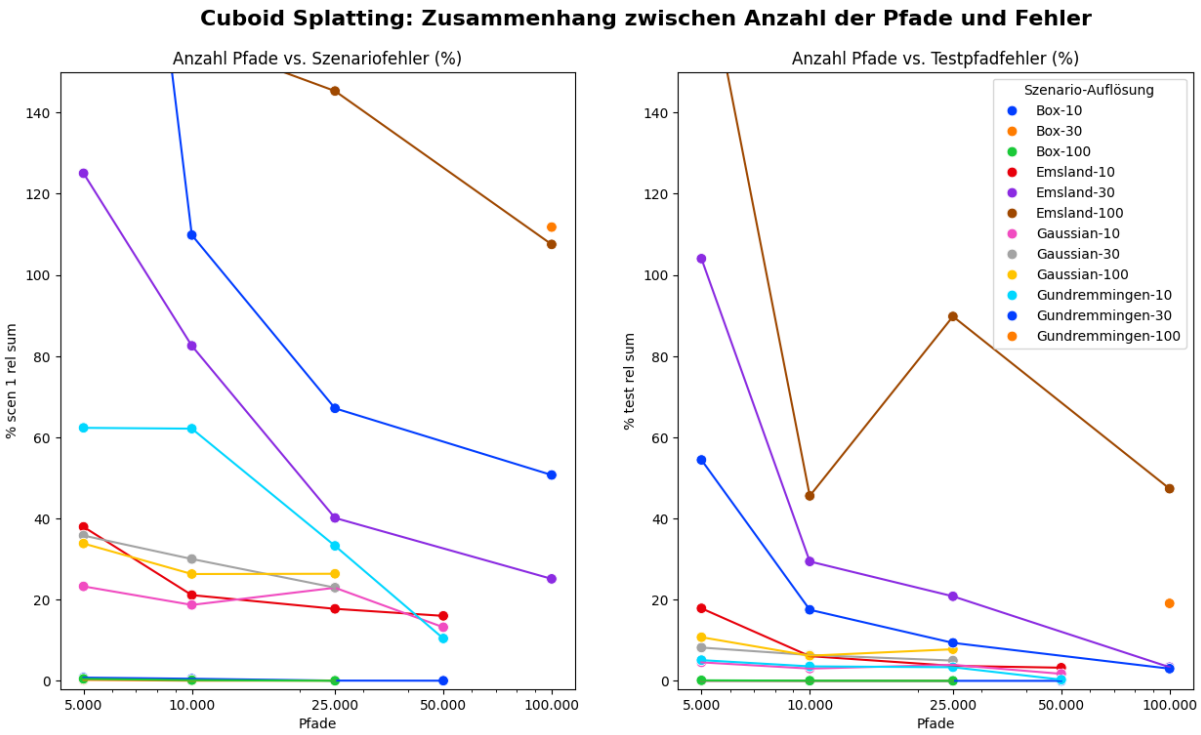

*Abbildung 4.3: Zusammenhang zwischen der Anzahl der Pfade (x-Achse) und dem Fehler (y-Achse, links: Szenariofehler, rechts: Testpfadfehler) bei der Methode **Cuboid Splatting**. Mit steigender Pfadanzahl wird das Modell deutlich besser.*

Bezüglich des Unterschieds zwischen dem prozentualen Rekonstruktionsfehler auf den Trainingspfaden und den Testpfaden (siehe Tabellen 4.6, 4.8, 4.10) zeigt sich bei der Methode *Cuboid Splatting* folgendes Muster: Bei einer zu geringen Anzahl verwendeter Trainingspfade (z.B. 5.000 oder 10.000) kommt es im Vergleich zu einer hohen Anzahl verwendeter Trainingspfade (z.B. 100.000) häufig zu einem Overfitting des Modells auf die Trainingspfade anstelle der Erzeugung eines generalisierbaren Modells. Dann fällt der prozentuale Rekonstruktionsfehler auf den Trainingspfaden niedrig aus („das Modell hat etwas gelernt und kann diese Pfade jetzt gut rekonstruieren“), der Rekonstruktionsfehler auf den Testpfaden ist aber sehr hoch („das Modell hat leider das Falsche gelernt und lässt sich nicht gut generalisieren“). Dieses Overfitting-Problem wird gelöst oder zumindest deutlich verringert, wenn mehr Trainingspfade verwendet werden.

Die Verbesserung der Ergebnisse der Methode *Cuboid Splatting* mit steigender Anzahl der im Training verwendeten Pfade hat verschiedene Gründe. Eine Erklärung liegt in der steigenden Überlappung bei steigender Pfadanzahl. Je mehr Pfade einen bestimmten Raumzeitpunkt beziehungsweise einen bestimmten Cuboid durchlaufen, desto eher muss das Modell einen zu all diesen Pfaden an der Stelle passenden Belastungswert finden und desto wahrscheinlicher ist dieser Wert dann auch generalisierbar (also über die Trainingsdaten hinaus, auch für bisher unbekannte Pfade richtig). Wir definieren und messen die Überlappung als durchschnittliche Überlappung von Pfaden pro Raumzeitpunkt über alle relevanten Raumzeitpunkte hinweg (über alle Raumzeitpunkte, durch die mindestens ein Pfad verläuft). Eine genauere Analyse bezüglich der Auswirkungen der Überlappung in Raumzeitpunkten findet sich in Unterkapitel 4.5.7. Eine Besonderheit der Methode *Cuboid Splatting* im Vergleich zur Methode *Inverse Rendering* liegt darin, dass für eine Korrelation der Belastungswerte bestimmter Pfade nicht zwingend eine explizite Überlappung in einem Raumzeitpunkt vorliegen muss. Bereits wenn zwei Pfade nur benachbart sind, und dabei durch den gleichen Cuboid verlaufen, liegt ein Zusammenhang zwischen ihrer Belastungsakkumulation in diesem Bereich vor. Diese Beschaffenheit der Methode *Cuboid Splatting* führt zu einer besseren Performance gegenüber der Methode *Inverse Rendering*.

Über die steigende Überlappung hinaus, erhöht eine höhere Pfadanzahl auch die Wahrscheinlichkeit, mehr verschiedenartige Pfade im Training inkludiert zu haben. Die Qualität der erzeugten Rekonstruktionen verbessert sich zum einen durch die Überlappung und zum anderen durch die Verschiedenhaftigkeit der vorliegenden Informationen. Kurz gesagt dadurch, dass sich unterschiedliche Pfade an unterschiedlichen Stellen in der Raumzeit überlappen und das Modell dadurch insgesamt passende Belastungswerte für die Raumzeitpunkte finden muss. Zuletzt bringt eine höhere Pfadanzahl auch schlicht mehr Informationen über die in der Raumzeit vorliegenden Belastungen mit und führt zu einer besseren Abdeckung des Raums und mehr rekonstruierbaren Raumzeitpunkten im Vergleich zu weniger Pfaden, was im Sinne des Anwendungsfalls positiv zu bewerten ist.

Auffällig in den 3D-Visualisierungen der Rekonstruktionen ist zudem, dass häufig die ersten Zeitschritte eines Szenarios besser rekonstruiert werden können als die letzten. Dies lässt sich dadurch erklären, dass sich im Laufe der Zeit immer mehr Personen aus dem betroffenen Gebiet heraus bewegen. Während die Rekonstruktion eines Szenarios in den ersten Zeitschritten also noch mit etwa 100% der Trainingspfade stattfindet, haben nach etwa der Hälfte der Zeitschritte bereits etwa 60% der Personen den Gefahrenbereich verlassen, sodass die Rekonstruktion der späteren Zeitschritte mit zunehmend weniger Pfaden und damit auch weniger Überlappung unter den Pfaden stattfindet.

Inverse Rendering: Auch in den Ergebnissen der Methode *Inverse Rendering* ist der allgemeine Trend zu erkennen, dass die Verwendung von mehr Pfaden zu besseren Rekonstruktions-Ergebnissen führt. Dieser Zusammenhang wurde in der Abbildung 4.4 visualisiert. Der Aufbau der Abbildung ist im Wesentlichen der gleiche, wie der Aufbau der Abbildung 4.3 zur Methode *Cuboid Splatting* (x-Achse: Anzahl der Pfade, y-Achse: Fehlermaße). Es gibt jedoch zwei wichtige Unterschiede: Zum einen wurde hier der prozentuale Szenariofehler sowohl für Raumzeitpunkte mit der Relevanz 1 (Punkte, durch die mindestens ein Pfad verlaufen ist - volle Linie), als auch für Raumzeitpunkte mit der Relevanz 5 (Punkte, durch die mindestens fünf Pfade verlaufen sind - gestrichelte Linie) eingezeichnet. Zum anderen geht die Skala der y-Achsen deutlich weiter: Während die y-Achsen bei der Methode *Cuboid Splatting* auf einen Wertebereich von 0 bis 150% skaliert waren, liegt für die Methode *Inverse Rendering* eine Skalierung auf 0 bis 300% vor.

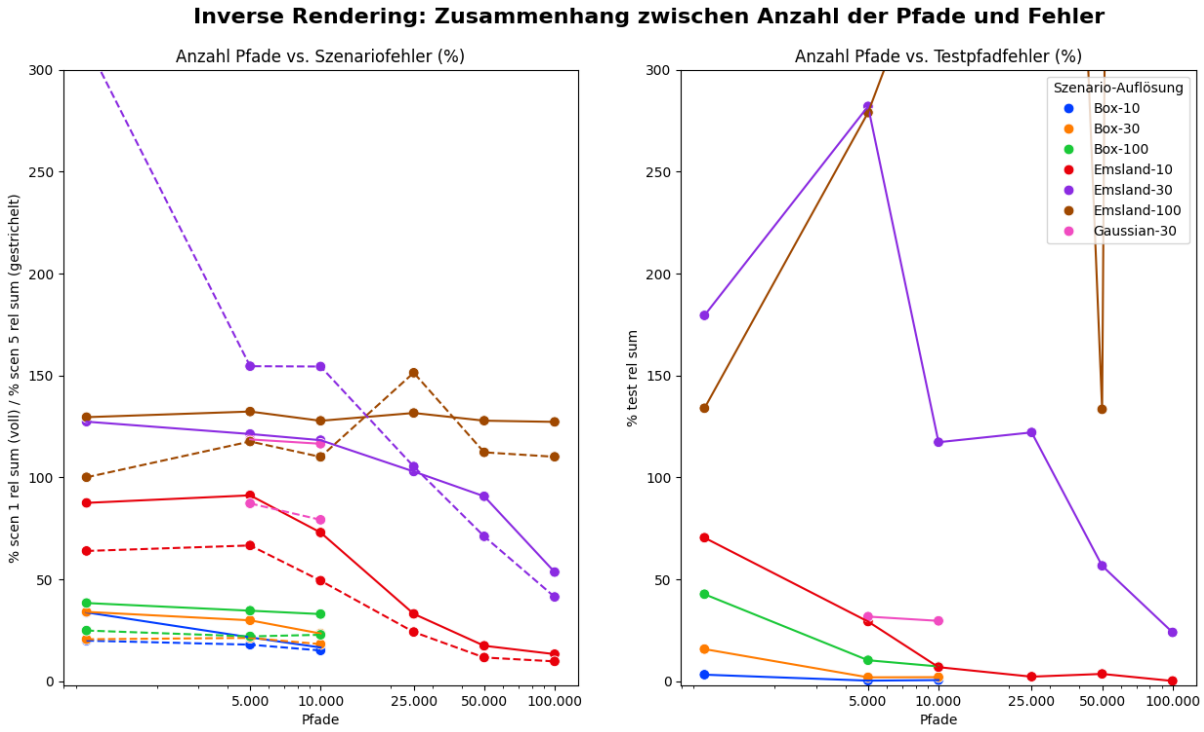

*Abbildung 4.4: Zusammenhang zwischen der Anzahl der Pfade (x-Achse) und dem Fehler (y-Achse, links: Szenariofehler (volle Linie: Relevanz 1, gestrichelte Linie: Relevanz 5), rechts: Testpfadfehler) bei der Methode **Inverse Rendering**. Im Allgemeinen wird das Modell mit steigender Pfadanzahl besser.*

Die obige Abbildung zeigt für die Methode *Inverse Rendering* im Allgemeinen einen fallenden prozentualen Szenario- und Testpfadfehler mit steigender Anzahl verwendeter Trainingspfade. Dies trifft insbesondere auf Rekonstruktionen in den räumlichen Auflösungen 10x10 und 30x30 zu. Dennoch gibt es auch Linien, die vom Trend abweichen: Bei der Rekonstruktion des Emsland-Szenarios in der Auflösung 100x100 bringt auch eine Erhöhung der Anzahl der Trainingspfade keine Verbesserung mit sich. Und bei Rekonstruktionen mit vorliegendem Szenariofehler von über 100% (also unbrauchbaren Rekonstruktionen), liegt auch kein sinnvoll interpretierbarer Verlauf des Testpfadfehlers vor. Ansonsten ist der Zusammenhang „steigende Pfadanzahl = sinkende Fehlermaße“ aber deutlich erkennbar.

Die Begründungen hierfür sind sehr ähnlich zu denen zur Methode *Cuboid Splatting*. Bei der Methode *Inverse Rendering* führt eine höhere Überlappung von Pfaden in Raumzeitpunkten zu deutlich besseren Rekonstruktionen (hierzu eine genauere Analyse in Abschnitt 4.5.7). Anders als bei der Methode *Cuboid Splatting* reicht hier eine Benachbarung von Pfaden nicht aus, damit ihre

Belastungswerte korreliert sind, sondern es muss dafür eine explizite Überschneidung in einem Raumzeitpunkt geben. Wie zuvor erklärt, führt dann die Überlappung dazu, dass das Modell einen zu allen Pfaden passenden und damit eher generalisierbaren Belastungswert für diesen Raumzeitpunkt finden muss. In Abbildung 4.4 ist auch erkennbar, dass die Linien des Szenariofehlers der Raumzeitpunkte mit Relevanz 5 unterhalb der Linien des Szenariofehlers der Raumzeitpunkte mit Relevanz 1 verlaufen. Raumzeitpunkte, durch die mindestens fünf Trainingspfade verlaufen, werden also mit der Methode *Inverse Rendering* im Durchschnitt deutlich besser rekonstruiert als Raumzeitpunkte, durch die nur mindestens ein Trainingspfad verläuft. Auch die Begründungen bezüglich der positiven Auswirkung höherer Pfadanzahlen durch die Verschiedenheit der vorliegenden Informationen und die höhere Raumausleuchtung treffen auf die Methode *Inverse Rendering* genauso zu wie auf die Methode *Cuboid Splatting*.

Im Abschnitt zur Methode *Cuboid Splatting* wurde erwähnt, dass diese Methode bei einer zu geringen Pfadanzahl teilweise zu Overfitting neigt, was durch eine höhere verwendete Pfadanzahl ausgeglichen werden kann. Wie bereits in vorherigen Abschnitten erwähnt, ist dieses Problem bei der Methode *Inverse Rendering* wesentlich ausgeprägter als bei der Methode *Cuboid Splatting*. Bei der Betrachtung der Ergebnisse in den Tabellen 4.7, 4.9, 4.11 werden die extremen Unterschiede zwischen dem sehr niedrigen Trainingspfadfehler und dem meist wesentlich höheren Testpfadfehler deutlich. Eine Angleichung der beiden Pfadfehlermaße auf das gleiche Niveau erfolgt beim Szenario Emsland in der Auflösung 10x10 erst bei einer Verwendung von 100.000 Pfaden im Training. In der Auflösung 30x30 liegt beim Szenario Emsland das Niveau des Trainings- und Testpfadfehlers selbst bei der Verwendung der höchsten Pfadanzahl (100.000 Pfade) weit auseinander (Trainingspfadfehler: 0,33%, Testpfadfehler: 23,88%), was starkes Overfitting indiziert.

4.5.2 Räumliche Auflösung der Rekonstruktion

Cuboid Splatting: Wie bereits im Eingangstext zu den Ergebnissen der Methode *Cuboid Splatting* erwähnt, ist die Auflösung der Raumzeit maßgeblich für die erreichbare Qualität der Rekonstruktionen. Je geringer die räumliche Auflösung (Auflösung in x- und y-Richtung) gewählt wird, desto besser werden die Ergebnisse des Modells. Dieser Zusammenhang zwischen der räumlichen Auflösung und der erreichbaren Qualität der Rekonstruktions-Ergebnisse mit der Methode *Cuboid Splatting* (jeweils mit der höchsten verwendeten Pfadanzahl) wird in der Abbildung 4.5 visualisiert. Der Vergleich mit verschiedenen Pfadanzahlen ist aus unserer Sicht möglich, weil die jeweiligen Lösungen mit Pfadanzahlen ermittelt wurden, bei denen eine weitere Erhöhung der Pfadanzahl die Lösung nicht mehr wesentlich verbessert hätte.

Den Ergebnis-Tabellen (4.6, 4.8, 4.10) im vorherigen Abschnitt (4.4.2 bis 4.4.4) ist zu entnehmen, dass der mit der Methode *Cuboid Splatting* erreichbare prozentuale Szenariofehler in der Auflösung 10x10, bei einer ausreichenden Menge von Pfaden, bislang zwischen ca. 10% und 16% liegt. In der Auflösung 30x30 liegt der erreichbare prozentuale Szenariofehler zwischen ca. 22% und 50%. Im Falle der 100x100 Auflösung liegt er bei den realistischen Szenarien um die 100%, wo das „Lernen“ des Modells dann nur noch am prozentualen Pfadfehler zu erkennen ist.

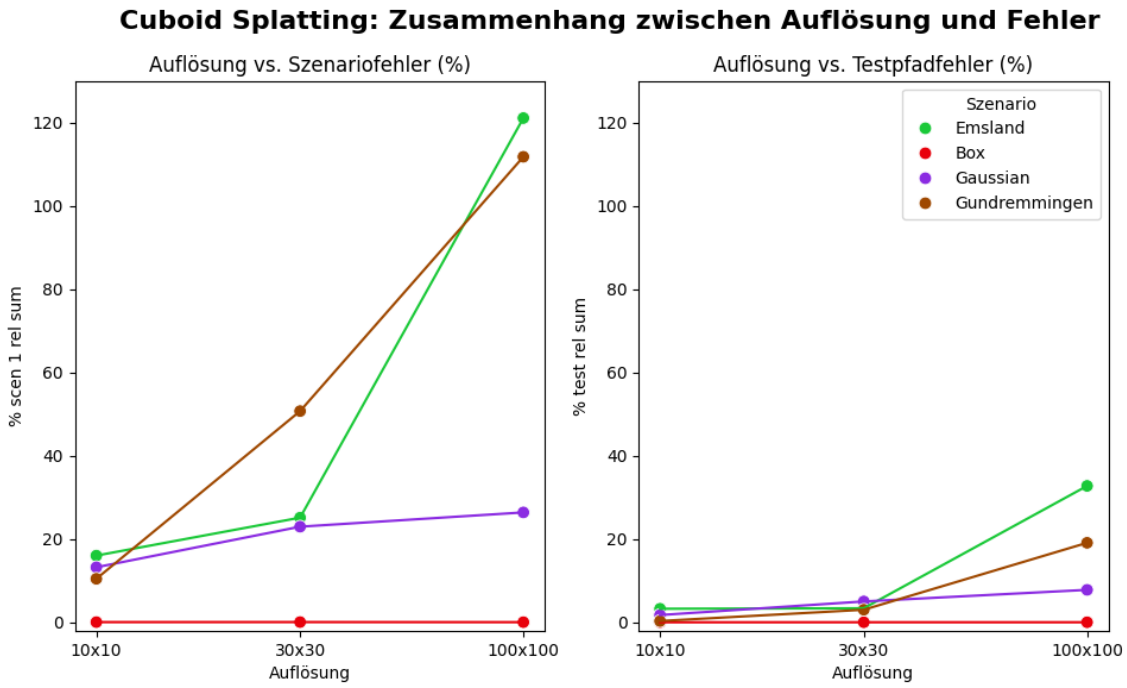

*Abbildung 4.5: Zusammenhang zwischen der räumlichen Auflösung (x-Achse) und dem Fehler (y-Achse, links: Szenariofehler, rechts: Testpfadfehler) bei der Methode **Cuboid Splatting**. Mit höherer Auflösung steigt der Fehler des Modells.*

Bezüglich des prozentualen Fehlers auf den Trainings- und Testpfaden liegen die mit *Cuboid Splatting* erreichbaren Werte in der Auflösung 10x10 zwischen 0% und 3,5% Abweichung. In der Auflösung 30x30 liegen sie zwischen 3% und 5%. Und in der Auflösung 100x100 werden prozentuale Pfadabweichungen zwischen 6% und 33% erreicht. Im Allgemeinen weicht der Fehler zwischen Trainings- und Testpfaden, bis auf in wenigen Ausnahmefällen wie bspw. Run 1072, kaum voneinander ab, was insgesamt für eine gute Generalisierbarkeit und gegen Overfitting der erreichten Modelle, auch in höheren Auflösungen spricht.

Einen Sonderfall bildet das Box-Szenario, welches in allen drei getesteten Auflösungen sehr gut rekonstruiert werden kann (prozentuale Szenariofehler und Pfadfehler <1%). Für die Rekonstruktion dieses künstlichen Szenarios ist die Methode *Cuboid Splatting* aufgrund der darin verwendeten geometrischen Form der Cuboids schlicht besonders gut geeignet. Die gute Performance der Methode auf diesem Szenario kann als Validierung des Funktionierens der Methode gesehen werden. Für die Einschätzung der Gesamt-Performance der Methode sind die erreichten Ergebnisse auf den realistischen Szenarien (Emsland, Gundremmingen) aber wesentlich relevanter.

Inverse Rendering: Bei der Methode *Inverse Rendering* liegt grundsätzlich der gleiche Zusammenhang vor wie bei der Methode *Cuboid Splatting*: Mit steigender räumlicher Auflösung wird es schwieriger, gute Rekonstruktionen zu erreichen. Die Rekonstruktions-Qualität ist also besser, wenn die Auflösung geringer ist. Bei steigender räumlicher Auflösung (von 10x10 auf 30x30), ist die Methode *Inverse Rendering* der Methode *Cuboid Splatting* in Bezug auf die Qualität der Rekonstruktionen in der Regel unterlegen. Der Zusammenhang zwischen der räumlichen Auflösung und der erreichten Qualität der Rekonstruktions-Ergebnisse mit der Methode *Inverse Rendering* (wie oben jeweils mit der höchsten verwendeten Pfadanzahl) wird in der Abbildung 4.6 visualisiert.

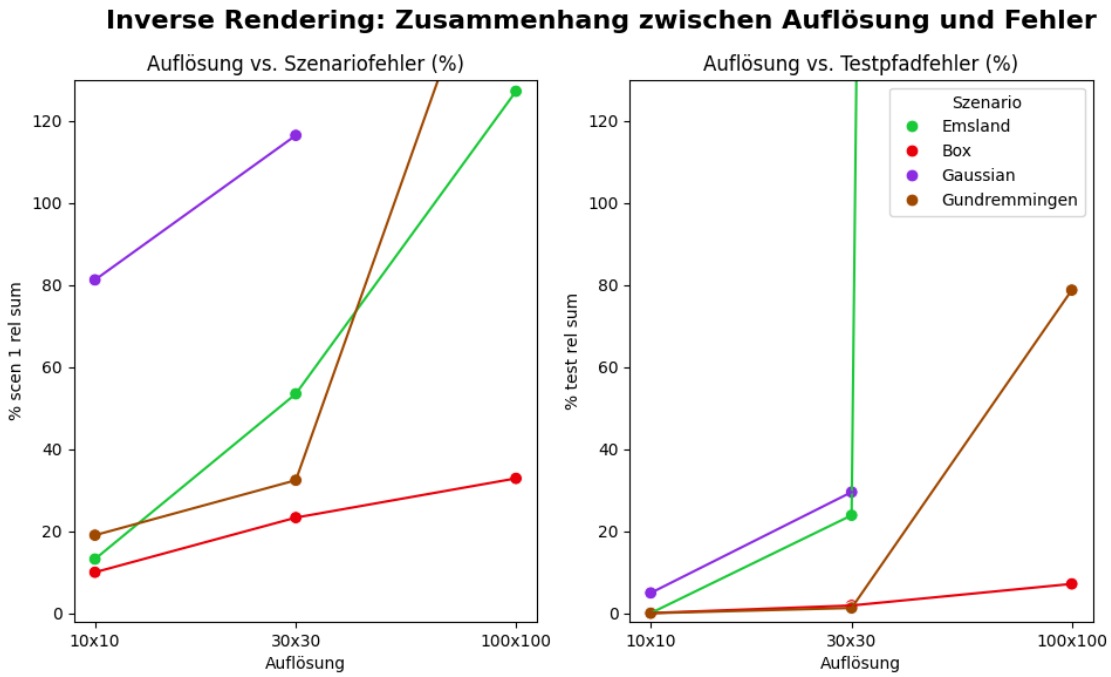

*Abbildung 4.6: Zusammenhang zwischen der räumlichen Auflösung (x-Achse) und dem Fehler (y-Achse, links: Szenariofehler, rechts: Testpfadfehler) bei der Methode **Inverse Rendering**. Mit höherer Auflösung steigt der Fehler des Modells, hier insbesondere auch der Testpfadfehler.*

Den Ergebnis-Tabellen (4.7, 4.9, 4.11) im vorherigen Abschnitt (Kapitel 4.4.2 bis 4.4.4) ist zu entnehmen, dass der mit der Methode *Inverse Rendering* erreichbare prozentuale Szenariofehler in der Auflösung 10x10, bei einer ausreichenden Menge von Pfaden, zwischen ca. 10% und 19% liegt (außer Gaussian Szenario, dort: 79%). In der Auflösung 30x30 liegt der erreichbare prozentuale Szenariofehler zwischen ca. 23% und 53% (außer Gaussian Szenario, dort: 116%). Im Falle der 100x100 Auflösung liegt er bei den realistischen Szenarien über 100% und anders als bei der Methode *Cuboid Splatting* ist dort auch anhand des prozentualen Testpfadfehlers kein „Lernen“ des Modells mehr zu erkennen. In der 100x100 Auflösung wird mit der Methode *Inverse Rendering* nur das Box-Szenario passabel rekonstruiert (Szenariofehler: 32,9%).

Bezüglich des prozentualen Fehlers auf den Testpfaden liegen die mit *Inverse Rendering* erreichbaren Werte in der Auflösung 10x10 zwischen 0% und 5%. In der Auflösung 30x30 liegen sie zwischen 1% und 29%. Und in der Auflösung 100x100 werden prozentuale Pfadabweichungen zwischen 7% (Box-Szenario) und über 100% (Emsland-Szenario) erreicht. Anders als bei der Methode *Cuboid Splatting* wird der Trainingspfadfehler hier nicht interpretiert, da dieser bei der Methode *Inverse Rendering* immer unter 1% liegt, was aber aufgrund des ausgeprägten Overfittings der Methode nicht aussagekräftig ist.

4.5.3 Verschiedene Szenarien von Belastungswerten in der Raumzeit

Cuboid Splatting: Bei den Ergebnissen der Methode *Cuboid Splatting* bezüglich der verschiedenen Szenarien, die in den Experimenten rekonstruiert wurden, ist vor allem zwischen den beiden realistischen Szenarien: Emsland und Gundremmingen und den beiden künstlichen Szenarien: Gaussian und Box zu unterscheiden. Der Zusammenhang zwischen den verschiedenen Szenarien und den erreichbaren Qualitätsmetriken wird in der Abbildung 4.7 visualisiert. Hierbei wird immer der pro Szenario und Auflösung beste in den Experimenten erreichte Fehler gezeigt. Diese wurde generell mit der höchsten verwendeten Pfadanzahl erreicht, welche je nach Kombination zwischen 25.000 und 100.000 Pfaden lag. Genaueres hierzu lässt

sich den Tabellen 4.6, 4.8, 4.10 entnehmen. Wie im vorherigen Abschnitt halten wir den Vergleich mit verschiedenen Pfadanzahlen für möglich, weil die jeweiligen Lösungen mit Pfadanzahlen ermittelt wurden, bei denen eine weitere Erhöhung der Pfadanzahl die Lösung nicht mehr wesentlich verbessert hätte. Die vorliegenden Zusammenhänge werden im Folgenden näher analysiert.

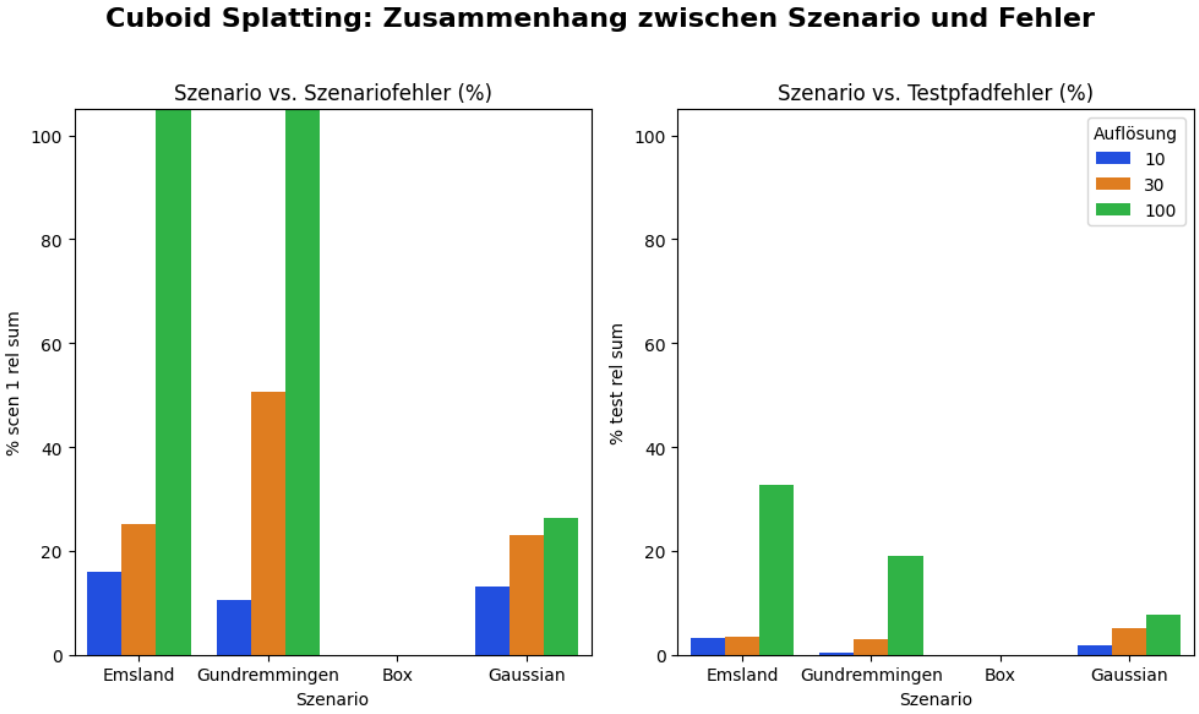

*Abbildung 4.7: Zusammenhang zwischen dem Szenario (x-Achse) und dem besten erreichten Fehler (y-Achse, links: Szenariofehler, rechts: Testpfadfehler) bei der Methode **Cuboid Splatting**. Die Auflösung ist farblich kodiert. Es lassen sich alle Szenarien in den Auflösungen 10x10 und 30x30 rekonstruieren, in der Auflösung 100x100 nur noch Box und Gaussian. Für das Szenario Box liegen alle Fehler unter 0,1%. Generell sind die künstlichen Szenarien leichter zu rekonstruieren als die realistischen.*

Insgesamt fällt es mit der Methode *Cuboid Splatting* leichter, die beiden künstlichen Szenarien zu rekonstruieren, als die beiden realistischen Szenarien. Insbesondere in höheren Auflösungen kommt es hier zu niedrigeren prozentualen Szenariofehlern. Dies lässt sich damit erklären, dass es sich bei den beiden künstlichen Szenarien um einheitlichere übergeordnete räumliche Strukturen handelt, die nachgebildet werden müssen. Währenddessen folgen die realistischen Szenarien weniger einheitlichen Formen. Dies ist eine wichtige Erkenntnis für Bewertungen der Performance von Rekonstruktionsmethoden, da die Fähigkeit zur Rekonstruktion realistischer Szenarien als wichtiger zu beurteilen ist, als die Fähigkeit zur Rekonstruktion künstlicher Szenarien. Insbesondere in höheren Auflösungen ist es für die Methode *Cuboid Splatting* bisher leichter, einheitliche Strukturen wie die Box oder das Gaussian Szenario nachzubilden, während die Rekonstruktion komplexerer Formen, die in den realistischen Szenarien auftreten, schwieriger ist.

Nichtsdestoweniger verläuft auch die Rekonstruktion der realistischen Szenarien mit der Methode *Cuboid Splatting* erfolgreich. Insbesondere im Vergleich zu den anderen implementierten Rekonstruktions-Methoden schneidet sie am erfolgreichsten bei der Rekonstruktion der realistischen Szenarien ab. Unter Voraussetzung der Verwendung einer ausreichenden Menge Pfaddaten, werden in geringerer räumlicher Auflösung (10x10) dabei

prozentuale Szenariofehler von etwa 10% bis 16% erreicht. Bei mittlerer räumlicher Auflösung (30x30) werden bei den realistischen Szenarien prozentuale Szenariofehler von etwa 25% bis 50% erreicht. Erst an der Rekonstruktion der realistischen Szenarien in der höheren räumlichen Auflösung von 100x100 scheitert die Methode bislang, selbst bei der Verwendung von 100.000 Trainingspfaden und liefert hier Ergebnisse mit einem etwa 100-prozentigen Szenariofehler. Der Lerneffekt des Modells ist in der höchsten Auflösung bei den realistischen Szenarien dann nur anhand des prozentualen Pfadfehlers auf den Trainings- und Testdaten erkennbar.

Um die Rekonstruierbarkeit realistischer Szenarien noch etwas genauer zu beleuchten, findet sich unten eine weitere Visualisierung (Abbildung 4.8). In den Auflösungen 10x10 und 30x30 sind gute Rekonstruktionen erreichbar (Szenariofehler je ca. 16 % und ca. 25 %). In der Zwischenabstufung 60x60 liegt eine deutliche Verschlechterung der Rekonstruktionsqualität auf etwa 73 % Szenariofehler vor. Bei einer Auflösung von 100x100 ist keine sinnvolle Rekonstruktion mit der Methode *Cuboid Splatting* mehr möglich (Szenariofehler >100%). Der Testpfadfehler steigt in etwa proportional zum Szenariofehler, liegt aber insgesamt niedriger (10x10: 3,2%, 30x30: 3,4%, 60x60: 32,1%, 100x100: 47,3%). In höheren Auflösungen sind also trotz schlechterer Szenario-Rekonstruktionen, auch in einem gewissen Maße Pfad-Rekonstruktionen möglich.

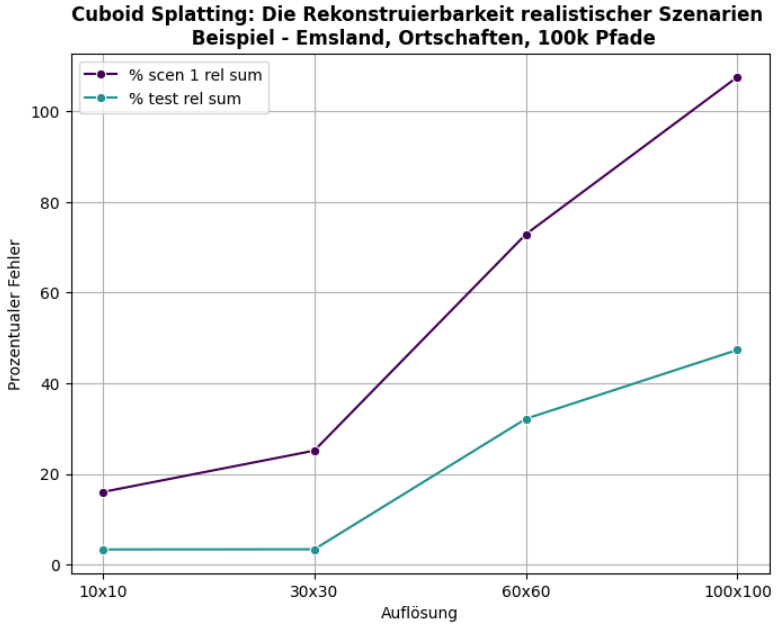

*Abbildung 4.8: Die Rekonstruierbarkeit realistischer Szenarien mit der Methode **Cuboid Splatting** bei steigender Auflösung illustriert an der Kombination „Emsland, Ortschaften, 100.000 Pfade“. Der Szenariofehler ist in lila dargestellt und der Testpfadfehler in türkis.*

Inverse Rendering: Anders als bei der Methode *Cuboid Splatting*, mit der die beiden künstlichen Szenarien deutlich leichter zu rekonstruieren waren als die beiden realistischen Szenarien, sieht das Bild bei der Methode *Inverse Rendering* etwas anders aus: Das künstliche Box-Szenario lässt sich in allen Auflösungen gut rekonstruieren, das künstliche Gaussian-Szenario jedoch in keiner. Die beiden realistischen Szenarien sind bei geringerer Auflösung gut rekonstruierbar, in hoher Auflösung jedoch nicht mehr. Wie oben, wird der Zusammenhang zwischen den verschiedenen Szenarien und den erreichbaren Qualitätsmetriken in der Abbildung 4.9 visualisiert. Hierbei wird immer der pro Szenario und Auflösung beste in den Experimenten erreichte Fehler gezeigt. Dieser wurde auch hier generell mit der höchsten verwendeten Pfadanzahl erreicht, welche je nach Kombination zwischen 25.000 und 100.000 Pfaden lag. Genaueres hierzu lässt sich den Tabellen 4.7, 4.9, 4.11 entnehmen. Bei der Methode *Inverse Rendering* wird zusätzlich noch der

Szenariofehler auf Raumzeitpunkten mit der Relevanz 5 gezeigt. Die vorliegenden Zusammenhänge werden im Folgenden näher analysiert.

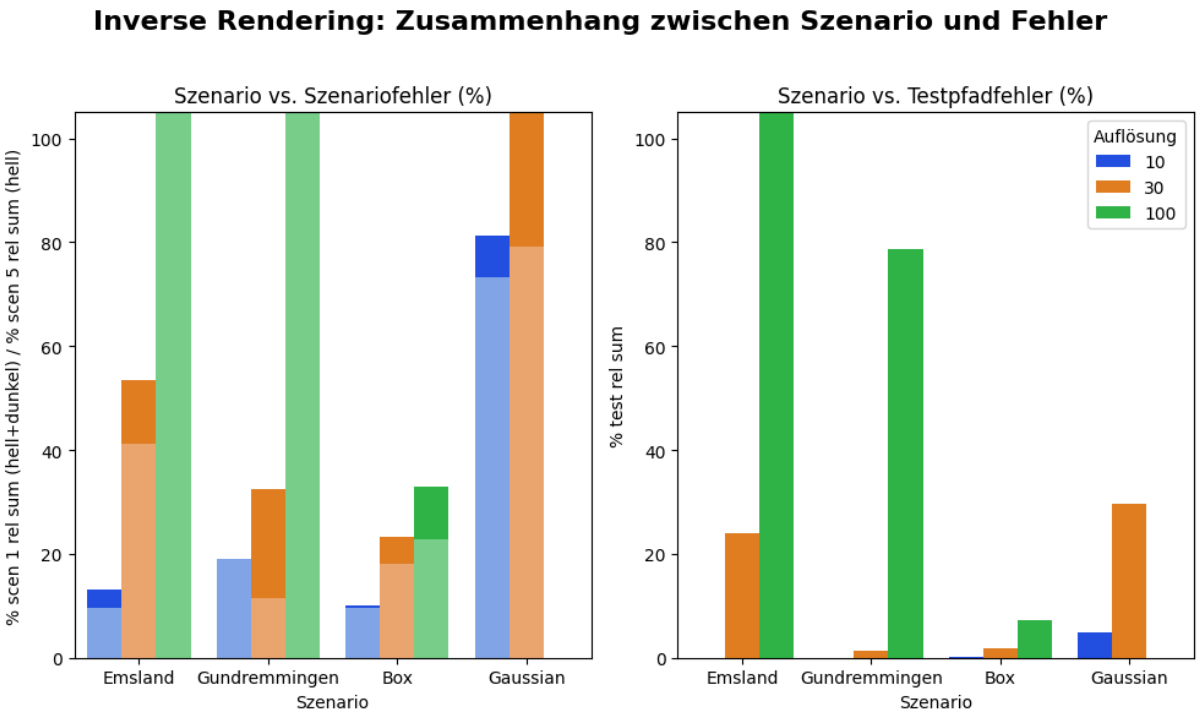

*Abbildung 4.9: Zusammenhang zwischen dem Szenario (x-Achse) und dem besten erreichten Fehler (y-Achse, links: Szenariofehler, rechts: Testpfadfehler) bei der Methode **Inverse Rendering**. Die Auflösung ist farblich kodiert. Die beiden realistischen Szenarien lassen sich in den Auflösungen 10x10 und 30x30 rekonstruieren, das Box-Szenario in allen Auflösungen und das Gaussian-Szenario in keiner Auflösung. Der prozentuale Szenariofehler ist bei Raumzeitpunkten der Relevanz 5 in der Regel geringer als bei denen mit Relevanz 1.*

Das Box-Szenario lässt sich über alle drei getesteten Auflösungen hinweg gut rekonstruieren (prozentualer Szenariofehler zwischen 10% und 33%). Wir erklären dies damit, dass das Box-Szenario eine klare Form und hohe Belastungswerte im Szenario hat. Dadurch sind Pfade, die durch belastete Raumzeitpunkte verlaufen sind, in ihrer Exposition eindeutig von Pfaden zu unterscheiden, die nicht durch belastete Raumzeitpunkte verlaufen sind, was sich positiv auf die Rekonstruierbarkeit mit der Methode *Inverse Rendering* auswirkt.

Das Gaussian-Szenario hingegen lässt sich mit der Methode *Inverse Rendering* in keiner der getesteten Auflösungen gut rekonstruieren. Selbst in der geringsten Auflösung liegt hier ein Szenariofehler von etwa 80% vor. Eine mögliche Erklärung bilden hier die geringen Belastungswerte des Szenarios und dass es, anders als beim Box-Szenario, keine klare Abgrenzung zwischen belasteten und unbelasteten Raumzeitpunkten gibt. Dadurch scheint es der Methode schwer zu fallen, die entlang der Pfade akkumulierte Belastung für die Rekonstruktion richtig in der Raumzeit zu verteilen.

Die beiden realistischen Szenarien Emsland und Gundremmingen lassen sich mit der Methode *Inverse Rendering* in der geringsten Auflösung (10x10) gut rekonstruieren: Szenariofehler je 13,2% bzw. 19,0%. In der mittleren Auflösung (30x30) gelingt das Gundremmingen-Szenario mit einem Szenariofehler von 32,4% besser als das Emsland-Szenario mit 53,5%. Interessanterweise ist es hier also andersherum als bei der Methode *Cuboid Splatting*, bei der das Emsland-Szenario besser rekonstruiert werden konnte als das Gundremmingen-Szenario. Je nach Art des realistischen Szenarios kann also mal die eine Methode und mal die andere bessere

Rekonstruktionsergebnisse liefern. In der hohen Auflösung (100x100) scheitert die Rekonstruktion der beiden realistischen Szenarien sowohl in Bezug auf den Szenariofehler als auch auf den Testpfadfehler (beide >100%).

Da bei der Methode *Inverse Rendering* die Relevanz der Raumzeitpunkte bei der Qualität der Rekonstruktionen wesentlich stärker ins Gewicht fällt als bei der Methode *Cuboid Splatting*, wurde in Abbildung 4.9 auch noch der prozentuale Szenariofehler an Raumzeitpunkten, durch die mindestens fünf Pfade verlaufen sind, eingetragen (% scen 5 rel sum). Dabei ist zu erkennen, dass eine Betrachtung lediglich von Punkten ab einer Relevanz von 5 in der Regel zu einer deutlichen Verbesserung der gemessenen Rekonstruktionsqualität führt. Es liegen Verbesserungen des prozentualen Szenariofehlers in der Regel zwischen 5% und 20% vor.

Bezüglich der Rekonstruierbarkeit realistischer Szenarien, exemplarisch anhand des Emsland-Szenarios und des Ortschaften-Straßennetzes betrachtet, lässt sich für die Methode *Inverse Rendering* aussagen, dass in der geringsten Auflösung (10x10) eine sehr gute Rekonstruktion erreicht werden kann. In der mittleren Auflösung (30x30) ist eine Rekonstruktion weiterhin möglich, dennoch liegt bereits eine deutliche Verschlechterung der Rekonstruktionsqualität, hier insbesondere auch in Bezug auf den Testpfadfehler, vor. Eine Rekonstruktion in der hohen Auflösung (100x100) ist nicht mehr möglich. Weitere Zwischenauflösungen wurden hier aufgrund des zeitlichen Umfangs einzelner Experimente mit der Methode *Inverse Rendering* nicht mehr analysiert. Zu erwarten ist eine in etwa proportionale Verschlechterung der Rekonstruktionsqualität mit steigender Auflösung.

4.5.4 Vergleich der Straßennetze (Graphen)

Cuboid Splatting: Wie eingangs erwähnt, wurde die Methode *Cuboid Splatting* ausschließlich auf dem Ortschaften-Straßennetz ausführlich getestet. Zunächst wurde jedoch ein einfacher Vergleich der Ergebnisse von Experimenten auf allen drei Straßennetzen (Hauptstraßen, Ortschaften, Stadt bzw. Platz) in der räumlichen Auflösung 10x10 durchgeführt. Dabei waren zwei Dinge zu erkennen: Zum einen liegen bei den Experimenten auf allen drei Straßennetzen sehr ähnliche Muster bezüglich der Auswirkungen der verschiedenen Dimensionen der Experimentparameter auf die Qualität der Ergebnisse vor. Also beispielsweise, dass mit steigender Trainingspfadanzahl die Ergebnisse deutlich besser werden und dass das Box-Szenario am besten rekonstruiert werden kann, aber auch die Szenarien Emsland, Gaussian und Gundremmingen gut rekonstruiert werden können. Zum anderen zeigte sich, dass die Rekonstruktionen auf dem Ortschaften-Straßennetz in den meisten Fällen am besten funktionierten, weshalb dieses für die ausführlichere Evaluation der Methode bezüglich der verschiedenen Einfluss-Dimensionen herangezogen wurde.

Inverse Rendering: Für die Methode *Inverse Rendering* wurde ein ausführlicher Vergleich der Performance der Methode bei Experimenten mit den verschiedenen Straßennetzen Ortschaften, Hauptstraßen und Stadt (bzw. Platz) durchgeführt. Ein Vergleich des erreichten prozentualen Szenariofehlers bei den drei verschiedenen zugrundeliegenden Straßennetzen wird in Abbildung 4.10 gezeigt. Hierbei wurde in allen dargestellten Experimenten ein Trainingsdatensatz von 10.000 Pfaden zugrunde gelegt. In der ersten Reihe sind die Ergebnisse des Emsland-Szenarios (Auflösungen: 10x10, 30x30, 100x100) abgebildet, in der zweiten Reihe die des Box-Szenarios (Auflösungen: 10x10, 30x30, 100x100) und in der dritten Reihe das Gaussian-Szenario (Auflösungen: 10x10, 30x30) und das Gundremmingen-Szenario (Auflösung: 10x10). Nachfolgend werden die dargestellten Ergebnisse analysiert.

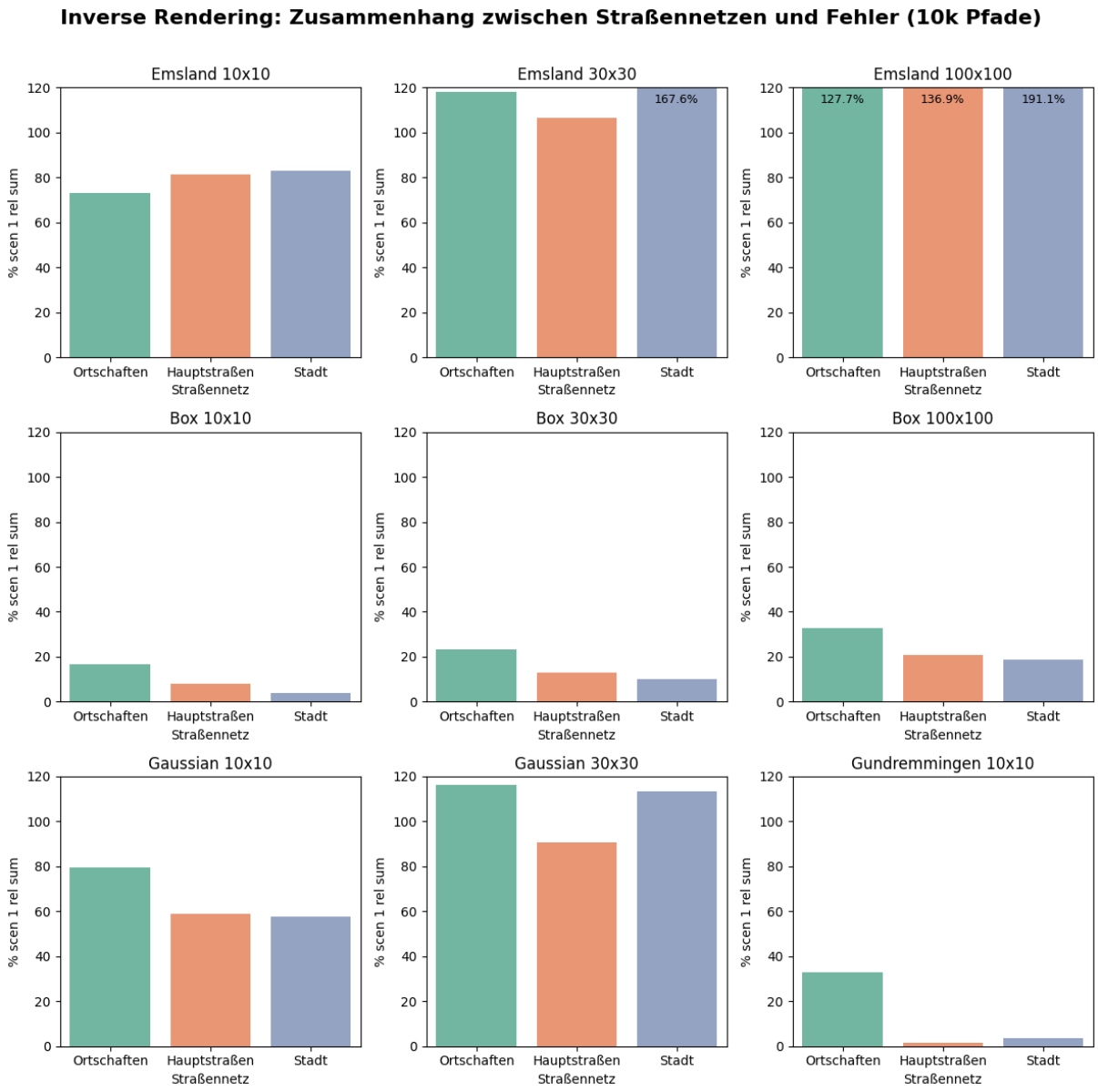

*Abbildung 4.10: Zusammenhang zwischen dem Straßennetz (Graphen) und dem Szenariofehler bei der Methode **Inverse Rendering**. Alle dargestellten Experimente wurden mit 10.000 Trainingspfaden durchgeführt. Die Ergebnisse variieren mit dem Szenario und der Auflösung, sodass es keinen allgemeinen Trend dahingehend gibt, dass eine bestimmte Art von Straßennetz zu besonders guten oder schlechten Ergebnissen führt.*

Beim Emsland-Szenario wird in der Auflösung 10x10 mit 10.000 Trainingspfaden die beste Rekonstruktionsqualität mit dem Ortschaften-Straßennetz erreicht. Am zweitbesten ist die Qualität mit dem Hauptstraßen-Straßennetz und am schlechtesten mit dem Stadt-Straßennetz. In der Auflösung 30x30 gelingt die Rekonstruktion auf dem Hauptstraßen-Straßennetz am besten, wobei hier sowie in der Auflösung 100x100 alle Rekonstruktionsergebnisse mit 10.000 Pfaden zu einem Szenariofehler größer als 100% führen. Insgesamt ist hier also kein eindeutiger Trend der Auswirkungen der verschiedenen Straßennetze zu erkennen.

Beim Box-Szenario werden in allen drei Auflösungen (10x10, 30x30, 100x100) die besten Rekonstruktionen auf dem Stadt-Straßennetz erreicht. Am zweitbesten gelingen sie auf dem Hauptstraßen-Straßennetz und am wenigsten gut auf dem Ortschaften-Straßennetz. Bei diesem

Szenario gibt es also ein Muster darin, wie sich die unterschiedlichen Arten von Straßennetzen auf die Qualität der Rekonstruktionen auswirken.

Beim Gaussian-Szenario wird in der Auflösung 10x10 die beste Rekonstruktion auf dem Stadt-Straßennetz erreicht. In der Auflösung 30x30 ist die einzige Rekonstruktion mit einem Szenariofehler von unter 100% die auf dem Hauptstraßen-Straßennetz. Dementsprechend ist auch hier kein klarer Trend der Auswirkungen der verschiedenen Straßennetze zu erkennen.

Beim Gundremmingen-Szenario wurde nur die Auflösung 10x10 vergleichend ausgewertet. Hierbei zeigte sich ein großer Unterschied zwischen den erreichten Rekonstruktionsqualitäten. Die mit Abstand beste Rekonstruktion wurde auf dem Hauptstraßen-Straßennetz erreicht und die zweitbeste Rekonstruktion auf dem Stadt-Straßennetz. Die auf dem Ortschaften-Straßennetz erreichte Rekonstruktion fällt hier um fast 30% schlechter aus.

Insgesamt zeigen die Ergebnisse der soeben analysierten Experimente, dass sich das zugrundeliegende Straßennetz und damit die Art des Gebietes, für das eine Rekonstruktion in Folge eines Unfalls in einer kerntechnischen Anlage erstellt werden soll, auf die Qualität der Rekonstruktionen mit der Methode *Inverse Rendering* auswirken. Hierbei gibt es aber keinen klaren Trend dahingehend, bei welcher Gebietsart die Rekonstruktionen besonders gut oder schlecht gelingen. Vielmehr variiert dies mit dem zu rekonstruierenden Szenario und der räumlichen Auflösung dieser Rekonstruktion.

4.5.5 Einfluss von Warnzonen

Cuboid Splatting: Der Einfluss von Warnzonen wurde für die Methode *Cuboid Splatting* anhand des Emsland-Szenarios und des Ortschaften-Straßennetzes mit und ohne Warnzonen getestet. Die Ergebnisse zeigen ein interessantes Muster, bei dem in der Auflösung 30x30 mit weniger Pfaden mit Warnzonen deutlich bessere Rekonstruktionsergebnisse erzielt werden können als ohne Warnzonen. Dieser Zusammenhang wird in der Abbildung 4.11 visualisiert und nachfolgend analysiert.

In der räumlichen Auflösung 10x10 führen bei der Methode *Cuboid Splatting* Warnzonen im Vergleich zur Rekonstruktion ohne Warnzonen zu kaum unterschiedlichen Ergebnissen. Der prozentuale Szenariofehler bei Experimenten fällt hier sehr ähnlich wie ohne Warnzonen aus. Die Beeinflussung des Verhaltens der Personen durch die Warnzonen führt hier jedoch dazu, dass der erreichte prozentuale Fehler bezüglich der Trainingspfade und der Testpfade mit Warnzonen etwas niedriger ausfällt als ohne Warnzonen. Es scheint dem Modell hier also leichter zu fallen, die Pfade richtig zu rekonstruieren. Gleichzeitig führt dies nicht automatisch zu einer besseren Rekonstruktion der Belastungswerte in der Raumzeit, teilweise sogar zu einer geringfügig schlechteren.

In der räumlichen Auflösung 30x30 liegt bei der Methode *Cuboid Splatting* dagegen ein sehr viel deutlicheres Muster vor. Dabei führen Warnzonen und das dadurch beeinflusste Verhalten der Personen zu erheblich geringeren prozentualen Szenariofehlern bei Rekonstruktionen mit 5.000, 10.000 und 25.000 Pfaden im Vergleich zur Rekonstruktion ohne Warnzonen. Ohne Warnzonen ist eine Rekonstruktion des Emsland-Szenarios mit der Methode *Cuboid Splatting* mit 5.000 Pfaden unmöglich (Szenariofehler >100%) und mit 10.000 Pfaden immer noch schlecht (Szenariofehler ca. 82,5%). Die Einführung von Warnzonen jedoch verändert dieses Bild komplett und führt zu Szenariofehlern unter 40% selbst bei nur 5.000 und 10.000 Trainingspfaden. Mit Warnzonen wird der beste Szenariofehler in Höhe von 25,9% mit 25.000 Pfaden erreicht. Bei 100.000 Pfaden verschlechtert sich das Ergebnis dann allerdings wieder deutlich (Szenariofehler ca. 64%). Ohne

Warnzonen wird das beste Ergebnis mit der größtmöglichen Pfadanzahl (100.000 Pfade, Szenariofehler: 25,1%) erreicht. Auch der Testpfadfehler ist in dieser Auflösung bei 5.000, 10.000 und 25.000 mit Warnzonen erheblich viel besser als ohne Warnzonen, bei 100.000 Pfaden ist er schlechter als ohne Warnzonen.

Insgesamt sind also mit Warnzonen die Rekonstruktionen mit der Methode *Cuboid Splatting* in der Auflösung 30x30 mit 5.000, 10.000 oder 25.000 Pfaden viel besser als ohne Warnzonen.

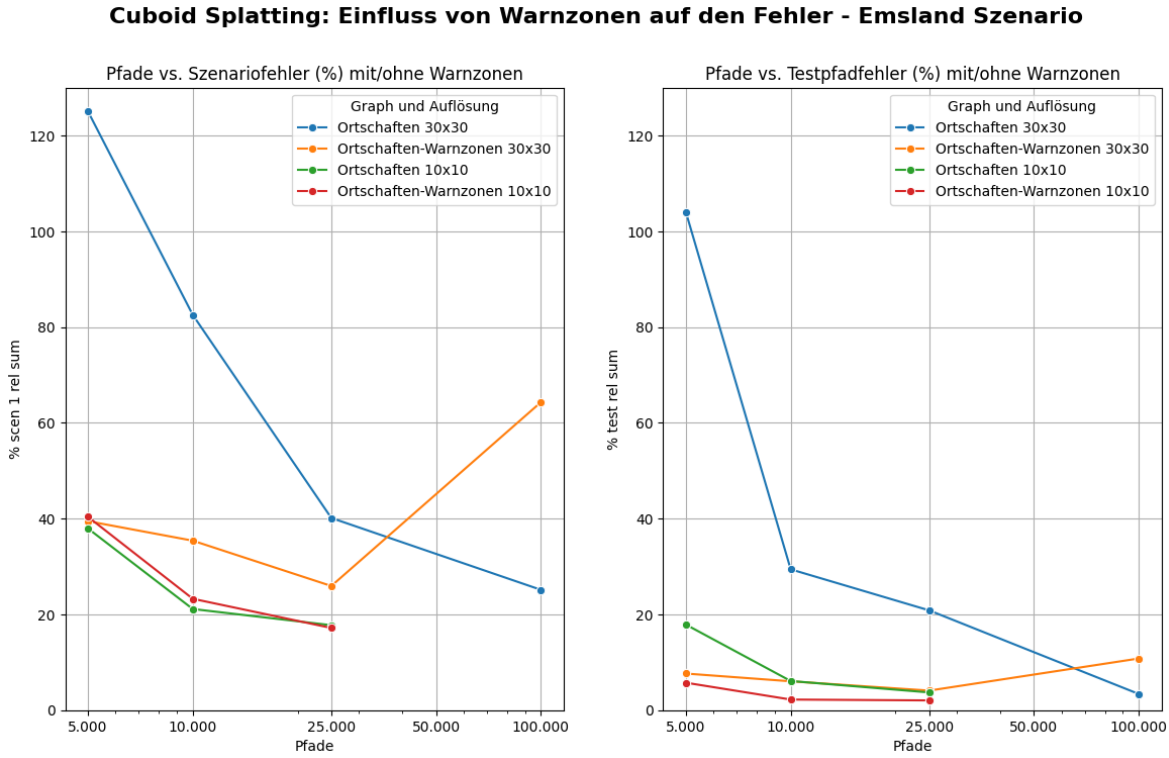

*Abbildung 4.11: Der Einfluss von Warnzonen auf den Fehler (y-Achse, links: Szenariofehler, rechts: Testpfadfehler) bei verschieden Anzahlen von Pfaden (x-Achse) in den Auflösungen 10x10 und 30x30. Diese Analyse für die Methode **Cuboid Splatting** wurde für die Szenario-Graph Kombination Emsland-Ortschaften durchgeführt. In der Auflösung 10x10 führen Warnzonen zu keinen wesentlichen Unterschieden. In der Auflösung 30x30 ermöglichen sie bei weniger Pfaden (5k, 10k, 25k) deutlich bessere Rekonstruktionen.*

Inverse Rendering: Auch für die Methode *Inverse Rendering* wurde der Einfluss von Warnzonen anhand des Emsland-Szenarios und des Ortschaften-Straßennetzes mit und ohne Warnzonen getestet. Hierbei wurden die räumlichen Auflösungen 10x10 und 100x100 über verschiedene Pfadanzahlen hinweg betrachtet. Die Ergebnisse zeigen, dass auch bei dieser Methode die Warnzonen zu deutlich besseren Rekonstruktionsergebnissen führen. Der Zusammenhang wird in der Abbildung 4.12 visualisiert und nachfolgend analysiert.

In den Ergebnissen der Experimente der Methode *Inverse Rendering* mit und ohne Warnzonen wird deutlich, dass die Warnzonen über alle getesteten Trainings-Pfadanzahlen hinweg (1.000, 5.000, 10.000) sowohl in der räumlichen Auflösung 10x10 als auch in der räumlichen Auflösung 100x100 zu deutlich besseren Ergebnissen führen. Anders als bei der Methode *Cuboid Splatting* ist die Verbesserung hier also auch in der geringsten Auflösung (10x10) zu erkennen.

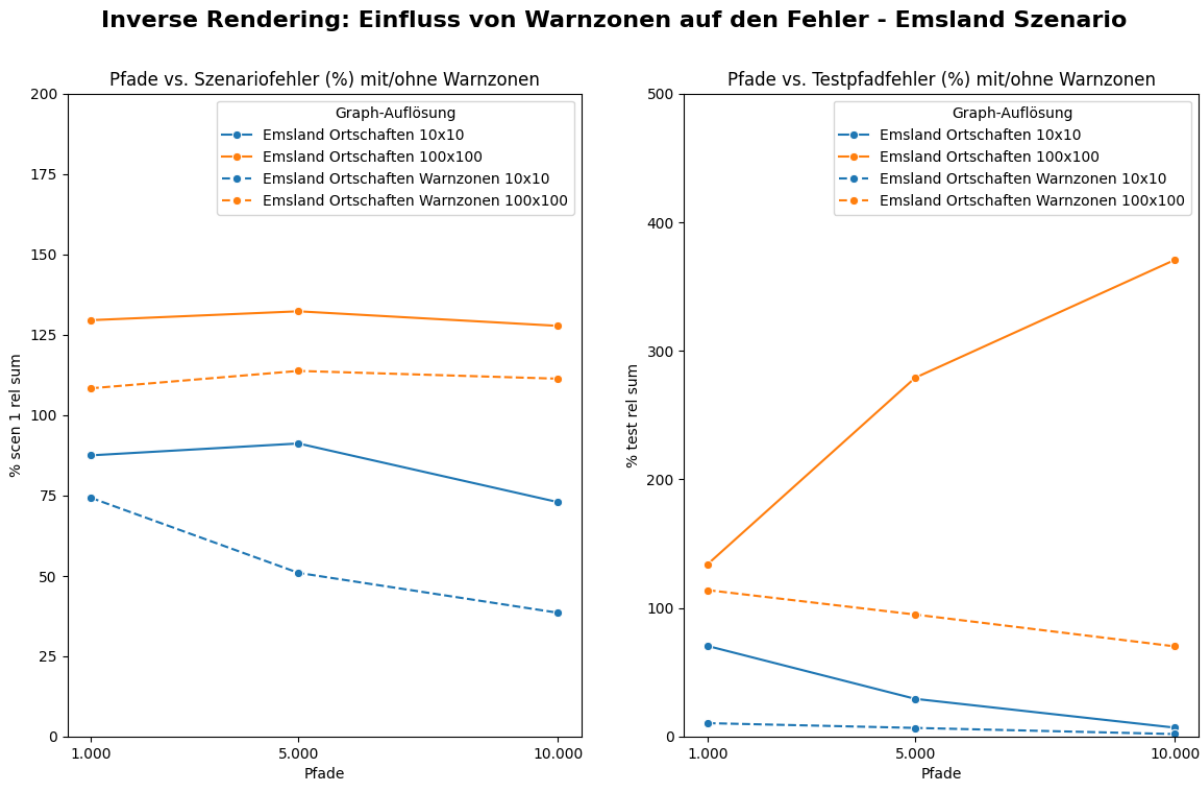

*Abbildung 4.12: Der Einfluss von Warnzonen auf den Fehler (y-Achse, links: Szenariofehler, rechts: Testpfadfehler) bei verschieden Anzahlen von Pfaden (x-Achse) in den Auflösungen 10x10 und 100x100. Diese Analyse für die Methode **Inverse Rendering** wurde für die Szenario-Graph Kombination Emsland-Ortschaften durchgeführt. Sowohl in der Auflösung 10x10 als auch in der Auflösung 100x100 führen Warnzonen zu deutlich besseren Rekonstruktionen.*

In der obigen Visualisierung verlaufen die gestrichelten Linien (Experimente mit Warnzonen) immer unterhalb der durchgezogenen Linien (Experimente ohne Warnzonen), was bedeutet, dass sowohl der prozentuale Szenariofehler als auch der prozentuale Testpfadfehler mit Warnzonen besser ausfallen als ohne. Beim Szenariofehler liegen die Verbesserungen zwischen etwa 10% und 40%, beim Testpfadfehler teilweise sogar noch deutlich höher mit über 100% Verbesserung in der Auflösung 100x100 mit 5.000 und 10.000 Pfaden. Zu erklären ist dies mit dem durch die Warnzonen beeinflussten Bewegungsverhalten der Personen in dem Gebiet.

Das durch die Warnzonen beeinflusste Bewegungsverhalten äußert sich wie folgt: In den Warnzonen außerhalb der Evakuierungszone bleiben die Personen entsprechend den Empfehlungen eher im Haus, anstatt sich draußen aufzuhalten und verlassen dadurch nicht so schnell das Gebiet, wie ohne die Warnzonen. Durch den längeren Verbleib der Personen im Gebiet sind einige Raumzeitpunkte mit unseren Modellen besser rekonstruierbar als ohne Warnzonen. Dieser Zusammenhang wird mit den nachfolgenden Visualisierungen noch einmal verdeutlicht.

In den folgenden Visualisierungen (Abbildungen 4.13, 4.14, 4.15, 4.16) stellen wir über die Zeitachse summierte Werte von Szenarien dar. Die x- und y-Achse ergeben die Ortsangabe, die Einfärbung den Wert. Je dunkler die Farbe ist, desto höher ist der Wert. Bei dem Wert handelt sich je nach Abbildung um einen Fehler, oder eine Überlappung.

In den Abbildungen 4.13 und 4.14 sehen wir den über die Zeit summierten Fehler, für alle Orte in der Auflösung 10x10. Der Fehler ist als Einfärbung am jeweiligen Ort gegeben. Die Skalierung des Fehlers ist logarithmisch. In den Abbildungen 4.15 und 4.16 sehen wir die Summierung der Heatmap, also der Angabe, wie hoch die Überlappung zu jedem Zeitpunkt in jedem

Raumzeitpunkt ist. In der summierten Darstellung wird daraus die Darstellung der Gesamtüberlappung des Ortes, unabhängig von der Zeit. Wir können beispielsweise erkennen, dass es einige Orte gibt, an denen keinen Personen waren. Diese Orte sind mit dem verwendeten Straßennetz nicht erreichbar gewesen.

Vergleichen wir nun diese Darstellungen, ist zu erkennen, dass bei einer Anwendung der Warnzonen, zum Beispiel im Punkt (5, 5), eine höhere Überlappung entsteht. Da diese Gebiete im Emsland-Szenario außerdem auch die Orte mit der höchsten Strahlenbelastung sind, werden genau diese Punkte besser rekonstruiert.

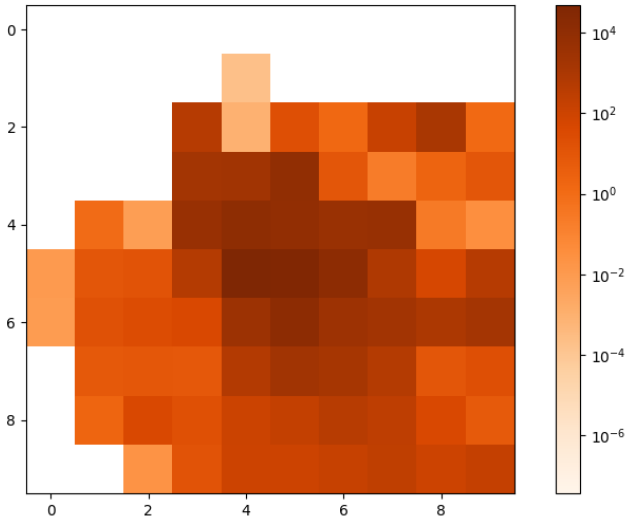

Abbildung 4.13: Über die Zeit summierter Fehler für Ortschaften ohne Warnzonen (Emsland, 10.000 Pfade).

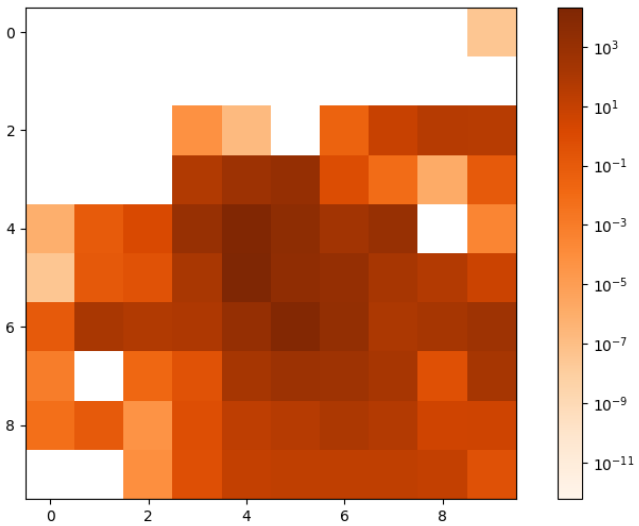

Abbildung 4.14: Über die Zeit summierter Fehler für Ortschaften mit Warnzonen. (Emsland, 10.000 Pfade). Wir sehen, dass der Fehler bei (5,5) deutlich geringer ausfällt als im Beispiel ohne Warnzonen.

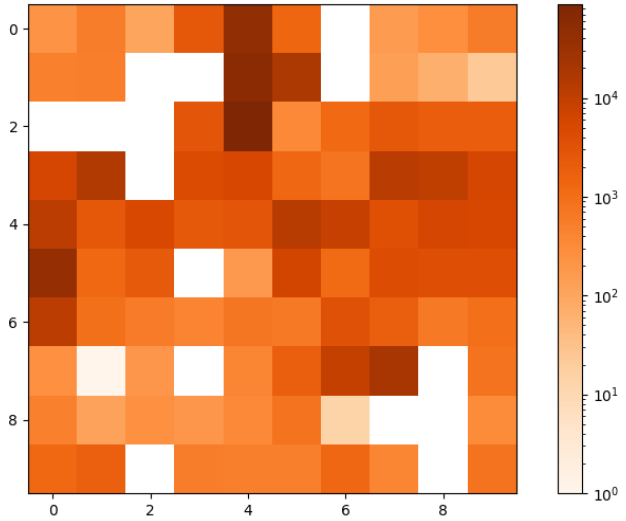

Abbildung 4.15: Über die Zeit summierte Heatmap (Anzahl der Pfade pro Raumzeitpunkt) für Ortschaften ohne Warnzonen (Emsland, 10.000 Pfade). Durch die Summe wird die Gesamtüberlappung jedes Ortes angezeigt. Weiße Zellen beschreiben Punkte, an denen keine Personen waren. Je dunkler die Farbe ist, desto größer ist die Anzahl an Personen. Die Skalierung ist logarithmisch.

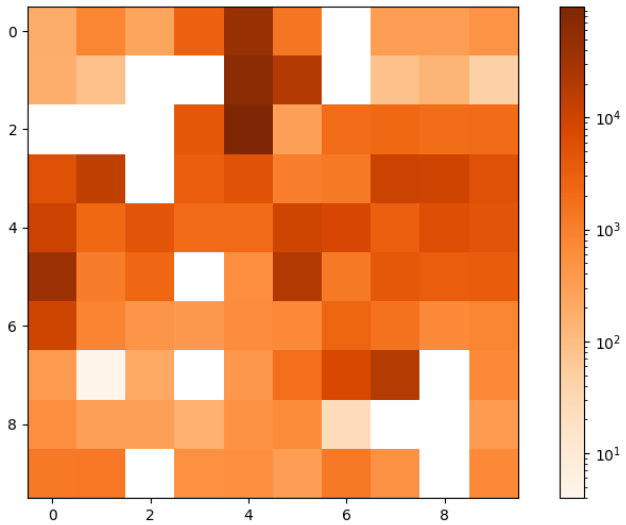

Abbildung 4.16: Über die Zeit summierte Heatmap für Ortschaften mit Warnzonen (Emsland, 10.000 Pfade). Durch die Summe wird die Gesamtüberlappung jedes Ortes angezeigt. Weiße Zellen beschreiben Punkte, an denen keine Personen waren. Je dunkler die Farbe ist, desto größer ist die Anzahl an Personen. Die Skalierung ist logarithmisch. Wir sehen, dass die Anzahl der Pfade bei (5,5) deutlich größer ist als in Abbildung 4.15.

4.5.6 Einfluss von Prioren

Cuboid Splatting: Die Nutzung eines Priors wurde für die Methode *Cuboid Splatting* bisher nicht entwickelt und daher auch keine Rekonstruktion damit getestet. Der Einbezug eines Priors gestaltet sich in dieser Methode weniger simpel als bei der Methode *Inverse Rendering*, wo der Prior einfach als Initialisierung der Belastungswerte der Raumzeit genutzt werden kann. Für eine

Weiterentwicklung der Methode *Cuboid Splatting* in zukünftigen Forschungsvorhaben wäre denkbar, den bisher beim *Inverse Rendering* verwendeten Prior aus Belastungswerten von Raumzeitpunkten mittels eines Clustering-Verfahrens in eine initiale Cuboid-Verteilung umzurechnen. Diese könnte dann anstelle der bisher zufälligen Initialisierung der Cuboids verwendet werden.

Inverse Rendering: Wie in Abschnitt 4.3.4.3 dargestellt, wurde für die Methode *Inverse Rendering* die Nutzung eines Priors in Form der Initialisierung mit bestimmten Belastungswerten in der Raumzeit implementiert. Prioren wurden in Experimenten mit den Szenarien Emsland (Prior: EuRim-Emsland) und Gundremmingen (Prioren: Gundremmingen_P1, Gundremmingen_P2) getestet. Die Abbildung 4.17 visualisiert die Ergebnisse der Experimente, in denen Prioren verwendet wurden, im Vergleich zu Experimenten, in denen keine Prioren verwendet wurden. In den dargestellten Experimenten waren bis auf die Verwendung eines oder keines Priors alle anderen Faktoren (Szenario, Straßennetz, Auflösung, Anzahl Trainingspfade) gleich.

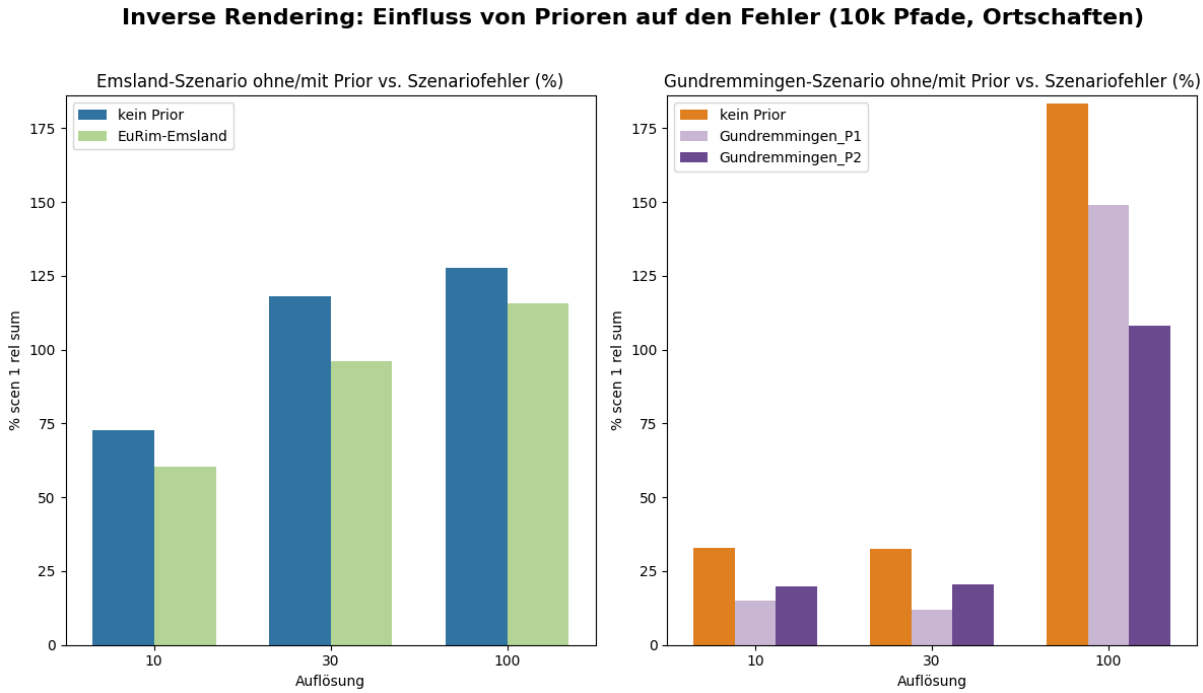

Abbildung 4.17: Einfluss der Verwendung von Prioren auf den Szenariofehler (%) bei der Rekonstruktion der Szenarien Emsland (links) und Gundremmingen (rechts) mit der Methode ***Inverse Rendering****. Die Verwendung von Prioren führt zu besseren Ergebnissen bei der Qualität der Rekonstruktionen.*

In den Ergebnissen der vergleichenden Experimente mit und ohne Prioren mit der Methode *Inverse Rendering* zeigt sich deutlich, dass die Verwendung von Prioren zu besseren Ergebnissen in der Rekonstruktion von Szenarien führt.

Bei Experimenten mit dem Emsland-Szenario wurde die Verwendung des Priors Eurim-Emsland anstelle einer Null-Initialisierung zu Beginn des Modelltrainings getestet. Es wurde immer die Kombination aus dem Ortschaften-Straßennetz und 10.000 Trainingspfaden verwendet. Dabei konnten in allen drei getesteten räumlichen Auflösungen mit dem Prior bessere Ergebnisse erzielt werden als ohne den Prior. In der Auflösung 10x10 sank der prozentuale Szenariofehler um etwa 12% von 72,9% auf 60,3%. In der Auflösung 30x30 wurde eine Verringerung des Szenariofehlers um 22 % und in der Auflösung 100x100 um 12% erreicht. Dennoch liegen in diesen beiden

Auflösungen (30x30 und 100x100) die Ergebnisse bei der Rekonstruktion mit 10.000 Pfaden weiterhin nah an oder über 100%.

Bei Experimenten mit dem Gundremmingen-Szenario wurde die Verwendung der Prioren Gundremmingen_P1 und Gundremmingen_P2 getestet. Auch hier wurde immer die Kombination aus dem Ortschaften-Straßennetz und 10.000 Trainingspfaden verwendet. Wie beim Emsland-Szenario wurden auch beim Gundremmingen-Szenario in allen drei getesteten räumlichen Auflösungen mit den Prioren bessere Ergebnisse erzielt als ohne Prior. In den Auflösungen 10x10 und 30x30 wurden mit dem Gundremmingen_P1 Prior die besten Ergebnisse erzielt, während in der Auflösung 100x100 der Prior Gundremmingen_P2 zur größten Qualitätsverbesserung führte. Dabei konnte der Szenariofehler in der Auflösung 10x10 um etwa 18% von 32,9% auf 14,9% gesenkt werden. In der Auflösung 30x30 konnte sogar eine Verbesserung um etwa 20% von 32,4% auf 12,0% Szenariofehler erreicht werden. In der Auflösung 100x100 betrug die Senkung des Szenariofehlers sogar etwa 75%, wobei das verbesserte Ergebnis aber immer noch einen Szenariofehler von etwa 108% aufwies.

Darüber hinaus erwähnenswert ist, dass durch die Verwendung des Priors Gundremmingen_P2 hier der Szenariofehler von Raumzeitpunkten ab einer Relevanz von 5 (% scen 5 rel sum) von 134,6% (ohne Prior) auf 67,2% (mit Prior) sank, was eine erhebliche Verbesserung darstellt.

Die Auswertungen der Experimente mit und ohne Prioren haben zudem gezeigt, dass die Rekonstruktionen unter der Verwendung von Prioren selbst dann besser werden als ohne Prior, wenn der Prior an sich einen höheren prozentualen Szenariofehler aufweist als das Rekonstruktions-Ergebnis eines Experiments ohne Prior.

4.5.7 Einfluss Überlappung und der Relevanz

Sowohl das Konzept der Überlappung als auch der Relevanz beziehen sich auf Überschneidungen von Pfaden in Raumzeitpunkten. Sie werden daher in diesem Unterkapitel gemeinsam behandelt. Zur Erinnerung: Der Grad der Relevanz eines Raumzeitpunktes ergibt sich aus der Anzahl an Pfaden, die sich in diesem Punkt überschneiden. Beispiel: Durch alle Raumzeitpunkte mit der Relevanz 5 verlaufen mindestens fünf Pfade. Mit der Überlappung messen wir die durchschnittliche Überlappung von Pfaden pro Raumzeitpunkt über alle relevanten Raumzeitpunkte hinweg (über alle Raumzeitpunkte, durch die mindestens ein Pfad verläuft).

Cuboid Splatting: Das Konzept der Relevanz von Punkten erscheint im Rahmen der Methode *Cuboid Splatting* deutlich weniger wichtig als bei der Methode *Inverse Rendering*. Wir vergleichen den prozentualen Szenariofehler von Raumzeit-Punkten durch die mindestens 1 (% scen 1 rel sum), 5 (% scen 5 rel sum) oder 10 (% scen 10 rel sum) Pfade verlaufen (Ergebnisse siehe Tabellen 4.6, 4.8, 4.10). Während beim *Inverse Rendering* ein klarer Trend dahingehend zu erkennen ist, dass Raumzeitpunkte mit einer höheren Relevanz wesentlich besser rekonstruiert werden können (siehe Tabellen 4.7, 4.9, 4.11), ist das bei der Methode *Cuboid Splatting* nicht eindeutig der Fall. In vielen Fällen wirkt sich die erhöhte Relevanz von Raumzeitpunkten beim *Cuboid Splatting* leicht positiv auf den prozentualen Szenariofehler aus, in einigen Fällen ist dies aber auch nicht der Fall oder die Raumzeitpunkte mit höherer Relevanz werden sogar schlechter rekonstruiert. Zum Vergleich: Bei der Methode *Inverse Rendering* ist die Verbesserung in Punkten mit höherer Relevanz sehr viel eindeutiger zu erkennen und bringt deutlichere Verbesserungen des prozentualen Szenariofehlers mit sich.

Eine Erklärung für das unterschiedliche Verhalten der beiden Methoden an dieser Stelle ist, dass bei der Methode *Inverse Rendering* die Belastungswerte von Pfaden nur bei expliziten

Überschneidungen in Raumzeitpunkten korreliert sein können. Bei der Methode *Cuboid Splatting* reicht es hingegen schon aus, wenn zwei Pfade durch denselben Cuboid verlaufen, damit ihre Belastungswerte miteinander zusammenhängen. Dieser Unterschied wurde bereits im Kapitel 4.5.1 ausführlich erläutert. Daher ist die exakte Überschneidung von Pfaden in Raumzeitpunkten und damit das Relevanz-Level von Raumzeitpunkten bei der Methode *Cuboid Splatting* weniger entscheidend als bei der Methode *Inverse Rendering*. Bei der Methode *Cuboid Splatting* wird die Relevanz von Punkten erst so richtig wichtig, wenn teilweise verrauschte Trainingsdaten verwendet werden (siehe Kapitel 5). Hier werden dann in der Tat Raumzeitpunkte mit höherer Relevanz deutlich besser rekonstruiert.

Während sich das erhöhte Relevanz-Level (auf mind. 5 oder mind. 10 Pfade pro Raumzeitpunkt) auf die Ergebnisse der Methode *Cuboid Splatting* nur geringfügig auswirkt, zeigt sich dennoch sehr deutlich, dass sich eine höhere durchschnittliche Überlappung pro relevantem Raumzeitpunkt insgesamt sehr positiv auf die Qualität der Rekonstruktions-Ergebnisse auswirkt. Hierbei ist es stark vom jeweiligen Szenario abhängig, wie hoch die durchschnittliche Überlappung sein muss, damit dieses Szenario gut rekonstruiert werden kann. In der Abbildung 4.18 wird dieser Zusammenhang verdeutlicht.

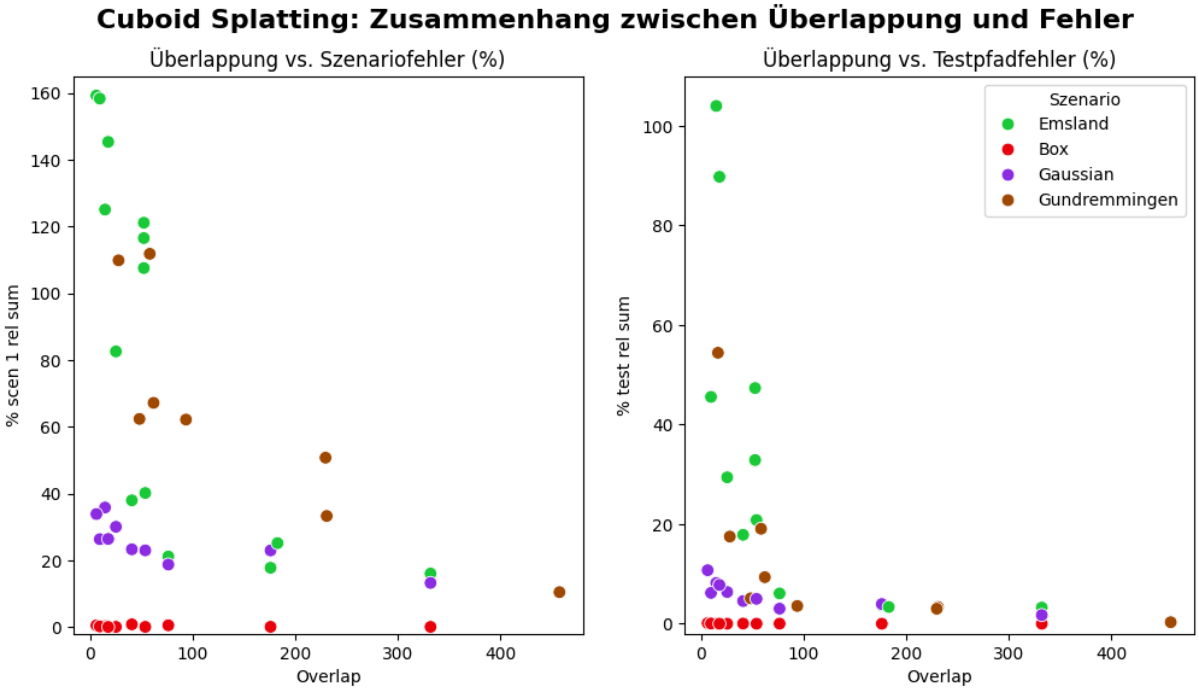

*Abbildung 4.18: Zusammenhang zwischen der Überlappung (x-Achse) und dem Fehler (y-Achse, links: Szenariofehler, rechts: Testpfadfehler) bei der Methode **Cuboid Splatting**. Mit steigender Überlappung wird das Modell deutlich besser. Die unterschiedlichen Szenarien benötigen unterschiedlich hohe Überlappungen, um gut rekonstruierbar zu sein.*

Man erkennt die einzelnen Szenarien als Schichten von Punkten übereinander. Das Box-Szenario (rot) wird unabhängig von der Überlappung immer sehr gut rekonstruiert. Beim Gaussian-Szenario (lila) ist hingegen deutlich zu erkennen, wie sich eine höhere durchschnittliche Überlappung positiv auf den Szenariofehler und den Testpfadfehler auswirken. Ab einer durchschnittlichen Überlappung von etwa 40 Pfaden pro relevantem Raumzeitpunkt werden hier die besten Rekonstruktionsergebnisse erreicht. Die beiden realistischen Szenarien (Emsland: grün und Gundremmingen: braun) hingegen, benötigen eine deutlich höhere durchschnittliche Überlappung, um gut rekonstruiert werden zu können. Ab einer durchschnittlichen Überlappung von etwa 50 Pfaden pro relevantem Raumzeitpunkt beginnt die Möglichkeit realistische Szenarien

mit einem Szenariofehler von unter 100% zu rekonstruieren. Beim Emsland-Szenario sind dann auch bereits Szenariofehler unter 50% möglich. Dies ist beim Gundremmingen-Szenario erst ab einer durchschnittlichen Überlappung von über 200 Pfaden pro relevantem Raumzeitpunkt möglich. Richtig gut werden die Rekonstruktionen der beiden realistischen Szenarien bei einer Überlappung über 300. Dann liegt der Szenariofehler unter 20 % und der Testpfadfehler unter 5%.

Es lässt sich also schlussfolgern, dass sich erstens eine höhere durchschnittliche Überlappung pro relevantem Raumzeitpunkt sehr positiv auf die Qualität der Rekonstruktionen auswirkt. Zweitens lassen sich Rückschlüsse darüber ziehen, je nach dem welche Art von Szenario zur Rekonstruktion vorliegt (künstlich oder realistisch) und wie viele Pfaddaten vorhanden sind, welche Auflösung zur Rekonstruktion gewählt werden sollte, um eine gewisse durchschnittliche Überlappung und damit verbunden eine gewisse Qualität der Rekonstruktion zu erreichen.

Inverse Rendering: Wie im Abschnitt zur Methode *Cuboid Splatting* bereits vergleichend angeführt, wirkt sich die Relevanz von Raumzeitpunkten bei der Methode *Inverse Rendering* erheblich auf die Qualitätsmetriken der Rekonstruktionen aus. Dies geht deutlich aus einem Vergleich der Werte in den Spalten „% scen 1 rel sum“ (Szenariofehler bei Raumzeitpunkten mit Relevanz 1), „% scen 5 rel sum“ (Szenariofehler bei Raumzeitpunkten mit Relevanz 5) und „% scen 10 rel sum“ (Szenariofehler bei Raumzeitpunkten mit Relevanz 10) in den Tabellen 4.7, 4.9, 4.11 am Anfang des Kapitels hervor.

Die Qualitätsverbesserungen bei einer Betrachtung der Raumzeitpunkte mit einer Relevanz von 5 im Vergleich zu 1 liegen in der räumlichen Auflösung 10x10 oft sogar bei über 20%. Sie fallen insbesondere dann hoch aus, wenn weniger Pfade zur Rekonstruktion verwendet wurden. Bei der Verwendung einer hohen Anzahl von Trainingspfaden gleichen sich die Fehlermaße der Raumzeitpunkte mit Relevanz 1, 5 und 10 stärker an. Das lässt sich damit erklären, dass es dann mehr Raumzeitpunkte mit der Relevanz 5 und 10 gibt, die ja ebenso in den Raumzeitpunkten mit der Relevanz 1 enthalten sind. In der räumlichen Auflösung 30x30 ist der Effekt der Qualitätsverbesserung mit einem höheren Relevanz-Level der Raumzeitpunkte ebenfalls deutlich zu beobachten. Beim Emsland-Szenario tritt dies hier allerdings erst ab einer Verwendung von 50.000 Trainingspfaden auf, bei den anderen Szenarien (Box, Gaussian, Gundremmingen) auch bei deutlich geringeren Pfadanzahlen. Auch in der räumlichen Auflösung 100x100 ist der Effekt der steigenden Rekonstruktions-Qualität bei Raumzeitpunkten mit höherer Relevanz ebenfalls, insbesondere am Box-Szenario, deutlich zu erkennen. Über die verschiedenen Auflösungen hinweg führt also eine höhere Relevanz von Raumzeitpunkten zu geringeren durchschnittlichen prozentualen Szenariofehlern an diesen Raumzeitpunkten.

Die Erklärung hierfür wurde oben bereits angesprochen. Anders als bei der Methode *Cuboid Splatting*, können bei der Methode *Inverse Rendering* die Belastungswerte von Pfaden nur bei expliziten Überschneidungen in Raumzeitpunkten korreliert sein. Das höhere Maß an Evidenz, dass dann für diesen Raumzeitpunkt vorliegt, wirkt sich positiv auf die Generalisierbarkeit des durch das Modell für diesen Raumzeitpunkt gefundenen Belastungswertes aus.

Neben dem Relevanz-Level spielt auch die durchschnittliche Überlappung pro Raumzeitpunkt bei der Methode *Inverse Rendering* eine entscheidende Rolle. Auch hier gilt: Je höher die durchschnittliche Überlappung, desto besser gelingen die Rekonstruktionen. Auffällig ist dabei, dass das Box-Szenario eine geringere durchschnittliche Überlappung benötigt, um mit der Methode *Inverse Rendering* gut rekonstruiert werden zu können, als die anderen Szenarien. Dieser Zusammenhang wird in der Abbildung 4.19 visualisiert.

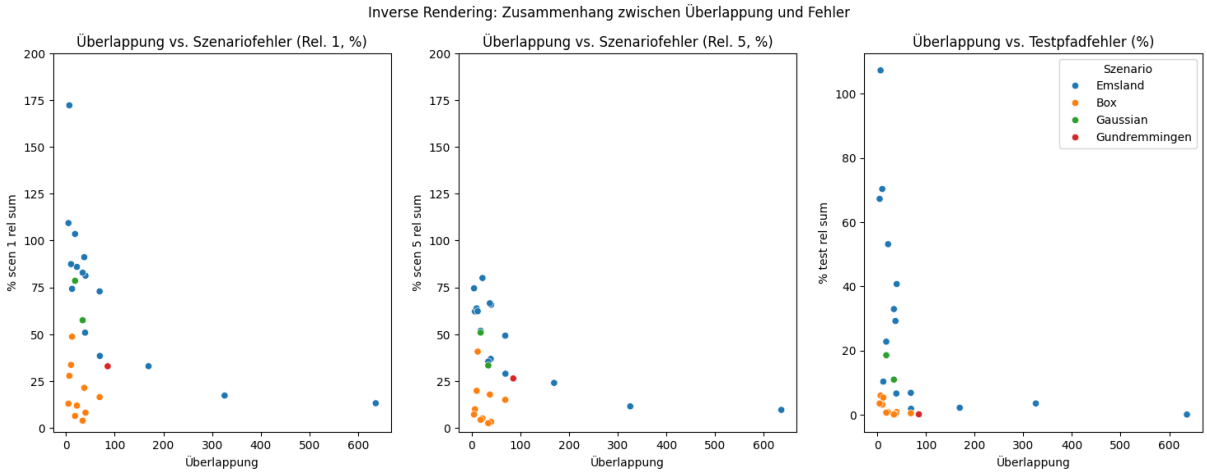

*Abbildung 4.19: Zusammenhang zwischen der Überlappung (x-Achse) und dem Fehler (y-Achse, links: Szenariofehler - Relevanz 1, mittig: Szenariofehler - Relevanz 5, rechts: Testpfadfehler) bei der Methode **Inverse Rendering**. Mit steigender Überlappung wird auch dieses Modell deutlich besser. Das Box-Szenario benötigt eine geringere Überlappung als die anderen Szenarien, um gut rekonstruierbar zu sein.*

Wie auch bei der Methode *Cuboid Splatting* ist bei den oben visualisierten Ergebnissen mit der Methode *Inverse Rendering* eine Schichtung zu erkennen. Das Box-Szenario wird bei einer geringeren durchschnittlichen Überlappung bereits gut rekonstruiert, während die anderen Szenarien, insbesondere das Emsland-Szenario, eine deutlich höhere durchschnittliche Überlappung benötigen, um gut rekonstruiert werden zu können. Dies geht aus den Ergebnissen mit allen drei dargestellten Fehlermaßen (prozentualer Szenariofehler bei Relevanz 1 und Relevanz 5, prozentualer Testpfadfehler) hervor.

Für die Methode *Inverse Rendering* lässt sich also genauso schlussfolgern, dass sich eine höhere durchschnittliche Überlappung pro relevantem Raumzeitpunkt sehr positiv auf die Qualität der Rekonstruktionen auswirkt. Zudem lassen sich auch hier auf Basis der entstehenden durchschnittlichen Überlappung Einschätzungen darüber ableiten, bei welcher Art von Szenario und Anzahl vorliegender Trainingspfade in welcher Auflösung sinnvoll rekonstruiert werden kann.

4.5.8 Detailbetrachtungen zu einer Rekonstruktion

Für jeden einzelnen Durchlauf eines Experimentes haben wir eine Reihe von Messungen durchgeführt, die uns geholfen haben, einen Einblick in die Verteilungen des Problems und der Lösung zu erhalten und zu verstehen, wie sich die Fehlermetriken zusammensetzen, die wir am Ende für einen Durchlauf angeben. Wir wollen in diesem Kapitel einige dieser Beobachtungen darstellen. Als Beispiel benutzen wir den Durchlauf mit der ID 2001 auf dem Szenario Emsland mit einer Auflösung von 10x10 und einer Pfadanzahl von 50.000 Pfaden, weil dessen Rekonstruktion vergleichsweise gut ist und sich daher für eine Analyse eignet.

Wir beginnen mit dem Szenario selbst in Abbildung 4.20. Dargestellt ist die Summe der Belastungen (d.h. der Iod-Konzentrationen) über alle Raumpunkte für jeden Zeitpunkt. Auf der x-Achse ist der zeitliche Verlauf in Stunden angegeben. Auf der y-Achse sehen wir die Summe des gesamten Iods zu diesem Zeitpunkt. Es wird deutlich, dass ein großer Teil des Iods innerhalb der Zeitspanne zwischen der 40. und der 90. Stunde vorliegt. Zu den anderen Zeitpunkten ist die Gesamtmenge an Iod fast null.

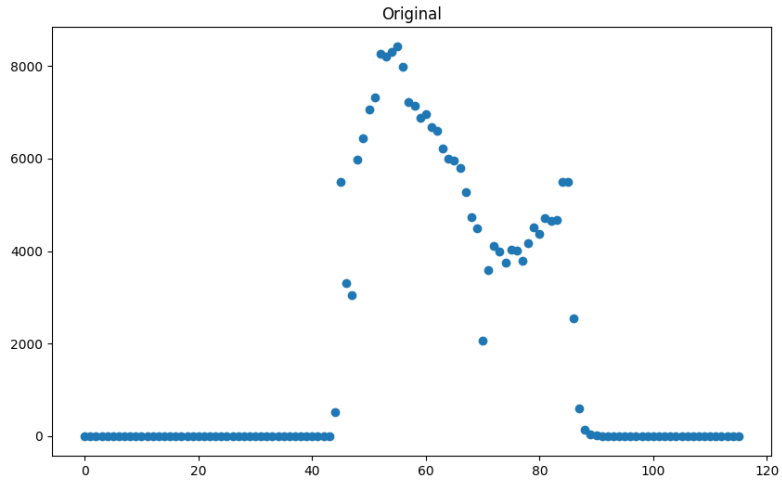

Abbildung 4.20: Die Summe der Iod-Konzentration des Emsland-Szenarios über allen Orten im zeitlichen Verlauf. Wir können sehen, dass der größte Teil der Exposition in Zeitraum zwischen der 40. und der 90

In Abbildung 4.21 betrachten wir den Verlauf, der durch die Summe der Heatmap der Pfade pro Zeitschritt berechnet wird. Er zeigt die Anzahl der Personen, die sich in dem Gebiet aufhalten, das wir rekonstruieren wollen. Wir können sehen, dass die Personen, wie simuliert, dieses Gebiet nach und nach verlassen. Das bedeutet für eine Rekonstruktion jedoch, dass wir während der Rekonstruktion zu den meisten Zeitschritten mit weniger als den immer genannten Pfadanzahlen auskommen müssen. Insbesondere zeigt sich hier, dass zwischen dem 40. und dem 90. Zeitschritt, in dem die meiste Aktivität zu rekonstruieren ist, nur noch zwischen einem Viertel und der Hälfte der Personen zur Rekonstruktion beitragen. Das ist zwar schade im Sinne der Rekonstruktion, aber zum einen können wir so feststellen, ob die Methoden auch erfolgreich Phasen mit wenig Iod rekonstruieren können, und zum anderen ist es auch denkbar, dass eine Eingrenzung des Zeitraumes in der Praxis nicht perfekt möglich ist.

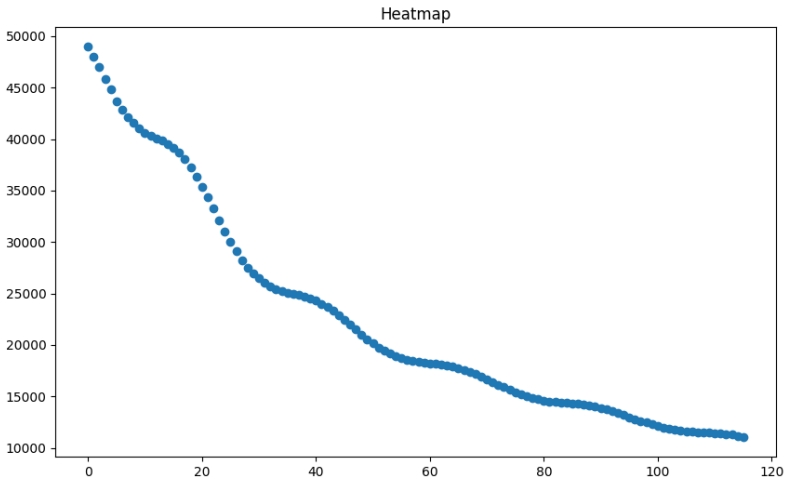

Abbildung 4.21: Die gesamte Anzahl der noch im Gebiet befindlichen Personen nimmt im zeitlichen Verlauf ab. Auf der x-Achse ist die Zeit in Stunden eingezeichnet. Die y-Achse zeigt die Anzahl der Personen im Gebiet.

Als nächstes betrachten wir die Verteilung der Fehler, die in der Rekonstruktion enthalten sind. Abbildung 4.22 zeigt, wie viele absolute Fehler in einem bestimmten Bereich liegen. Die Häufigkeit ist logarithmisch skaliert dargestellt. Man kann sehen, dass der größte Teil der Fehler sehr klein ist. Die Häufigkeit nimmt mit steigender Fehlergröße schnell ab. Einzelne große Fehler existieren, aber sie sind sehr selten.

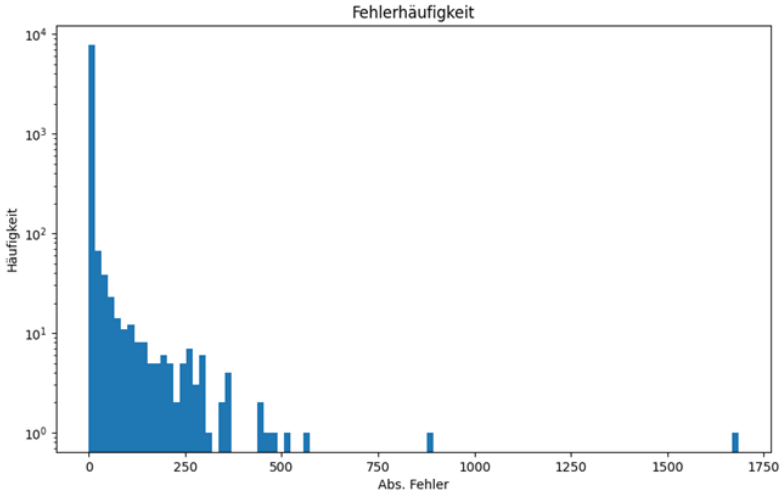

Abbildung 4.22: Häufigkeit der absoluten Fehler. Auf der x-Achse sind Fehler in kleinen Gruppen zusammengefasst. Jeder Gruppe wird entlang der y-Achse die Menge an Raumzeitpunkten zugeordnet, die in diesem Fehlerbereich liegen. Die y-Achse ist logarithmisch skaliert.

Als letztes betrachten wir noch den Verlauf des durchschnittlichen absoluten Fehlers, wenn wir die Menge der Raumzeitpunkte so reduzieren, dass nur Punkte mit einer Relevanz oberhalb eines Schwellwertes gewählt werden. Dieser Verlauf ist in Abbildung 4.23 dargestellt. Wir können sehen, dass der durchschnittliche Fehler für sehr hohe Überlappungen gegen null geht. Wir können auch sehen, dass eine Erhöhung des Schwellwertes im unteren Wertebereich eine schnelle Verbesserung erzeugt. So kann der durchschnittliche Fehler rasch halbiert oder sogar auf ein Achtel reduziert werden.

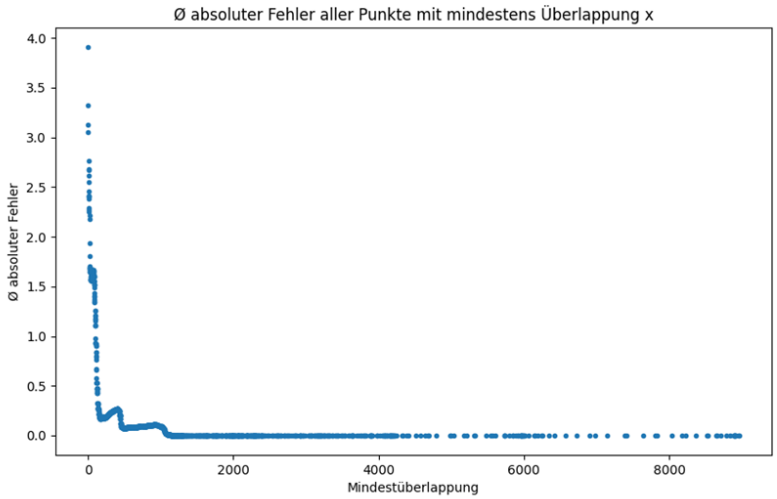

Abbildung 4.23: Der durchschnittliche absolute Fehler (y-Achse) nimmt für alle Punkte mit einer Mindestüberlappung schnell ab, wenn der Schwellwert der Überlappung (x-Achse) der Überlappung erhöht wird.

4.5.9 Verlauf des Trainings

Inverse Rendering: Das Training der Methode *Inverse Rendering* erfolgt, wie bereits beschrieben, in Epochen. Während jeder Epoche werden alle Trainingspfade in Batches iteriert und pro Batch ein Optimierungsschritt ausgeführt.

Wir evaluieren den Verlauf eines Trainings anhand des Rekonstruktionsfehlers pro Batch auf den Pfaden des Batches (Abbildung 4.24) und am Ende jeder Epoche auf allen Validierungspfaden (Abbildung 4.25), sowie anhand des Verlaufs der Lernrate (Abbildung 4.26). Alle Abbildungen zeigen auf der x-Achse die Zeit und auf der y-Achse den jeweiligen Wert in logarithmischer Skalierung.

Grundlegend ist zu beobachten, dass der Rekonstruktionsfehler langfristig sinkt, ab und zu jedoch spontan wieder stark ansteigt. Dann ist eine Reduktion der Lernrate notwendig. Wird diese Reduktion durchgeführt, pegelt sich der Rekonstruktionsfehler wieder ein und beginnt erneut zu sinken. Im Verlauf des Trainings konvergiert der Rekonstruktionsfehler. Der Konvergenzpunkt wird nach einer bestimmten Anzahl durchgeführter Optimierungsschritte erreicht und hängt damit von der Anzahl der Batches, bzw. Bewegungsprofile ab. Die Anzahl der Pfade pro Batch haben wir konstant gehalten, sodass die Anzahl der Batches von der Anzahl der Pfade anhängt. Wir haben meistens 10.000 Epochen trainiert. Ab 50.000 Pfaden haben wir nur noch 5000 Epochen trainiert, weil durch die hohe Pfadanzahl auch mit weniger Epochen genügend Optimierungsschritte durchgeführt werden.

Um ein Overfitting zu vermeiden, nutzen wir die Evaluation am Ende jeder Epoche, um die beste Rekonstruktion zu finden. Es handelt sich in diesem Falle um die Rekonstruktion aus der Epoche, in der die beste Evaluierung ermittelt wurde. Uns ist jedoch kein Fall bekannt, in dem nach einer genügenden Anzahl an Epochen die letzte Epoche nicht die beste ist.

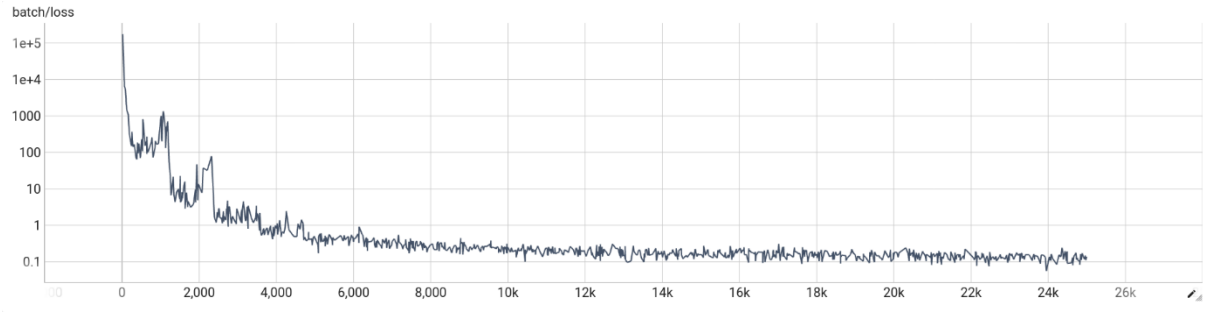

Abbildung 4.24: Entwicklung des Fehlers (y-Achse) während des Trainings pro Batch (x-Achse)

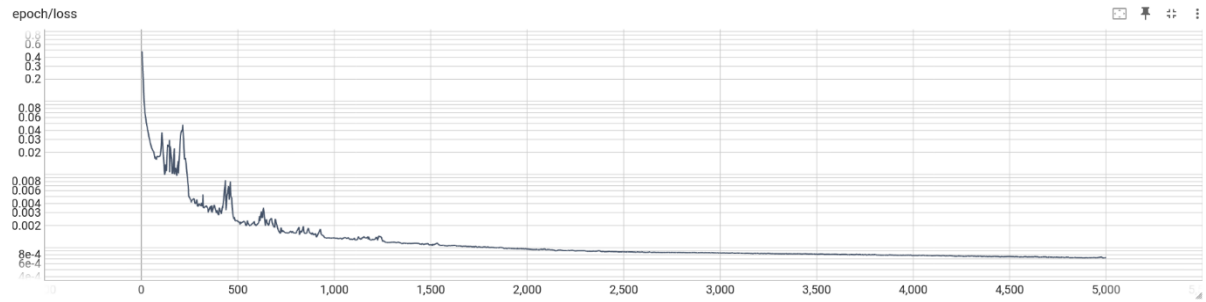

Abbildung 4.25: Entwicklung des Fehlers (y-Achse) während des Trainings pro Epoche (x-Achse)

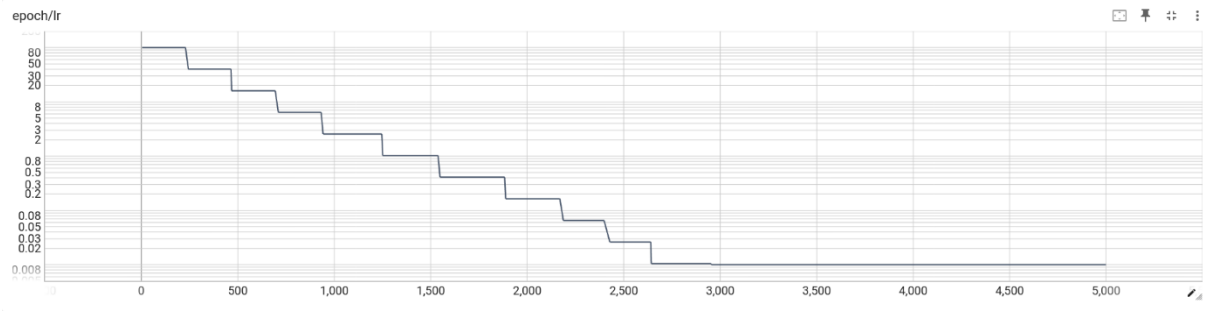

Abbildung 4.26: Entwicklung der Lernrate (y-Achse) während des Trainings pro Epoche (x-Achse)

Cuboid Splatting: Das grundsätzliche Vorgehen im Training der Methode *Cuboid Splatting* ähnelt stark der Methode *Inverse Rendering*. Die angewandte Fehlerfunktion (genannt Loss) bezieht sich auch hier auf den Trainingspfad-Fehler, der im Laufe des Trainings minimiert wird. In der Regel wurde zwischen 400 und 2000 Epochen lang trainiert, also 400 bis 2000 Male der Trainingsdatensatz durchlaufen. Hierbei wurden die Trainingspfade ebenfalls in Batches aufgeteilt, deren Größe zwischen 100 und 10.000 lag, je nach Auflösung und verwendeter Anzahl Trainingspfade. Anders als bei der Methode *Inverse Rendering*, wurde beim *Cuboid Splatting* kein Validierungsset zur Optimierung der Anzahl der Epochen im Training benutzt, da keine ausgeprägte Tendenz zum Overfitting beobachtet wurde (im allgemeinen ähnliche Fehler-Scores auf Trainings- und Testpfaden in den Ergebnissen) und dies daher nicht notwendig war. Wie beim *Inverse Rendering* wurde auch bei der Methode *Cuboid Splatting* ein Learning Rate Scheduler verwendet, der die angewandte Learning Rate im Training immer dann verringert (i. d. R. halbiert), wenn der Loss über eine bestimmte (konfigurierbare) Anzahl von Epochen nicht mehr gesunken ist. Wichtig ist auch an dieser Stelle noch einmal zu erwähnen, dass bei der Methode *Cuboid Splatting* die genutzten Hyperparameter eines Experiments individuell an den Run, insbesondere die Auflösung und die Pfadanzahl, angepasst werden sollten, um gute Rekonstruktionsergebnisse zu erzielen. Details zur Hyperparameteroptimierung sind in Abschnitt 4.3.5.8 beschrieben. Teilweise ist es dann auch sinnvoll, den gleichen Run mehrfach mit unterschiedlichen Hyperparametern durchzuführen, um daraus das beste Ergebnis nutzen zu können. Eine über einzelne Hyperparameteranpassungen hinausgehende, umfangreiche Hyperparameter-Optimierung (z. B. Grid Search) ist aufgrund des zeitlichen Umfangs einzelner Runs allerdings nicht möglich.

5. Testung der Methoden bei Fehlern in den Daten (AP 4)

Im letzten Arbeitspaket (AP4) wurde die Robustheit der beiden Methoden *Inverse Rendering* und *Cuboid Splatting* gegenüber Fehlern in den Trainingsdaten getestet. Hierbei wurden verschiedene Arten von Fehlern bewusst in die Trainingsdaten eingeführt und die Ergebnisse im Vergleich zu den fehlerfreien Trainingsdaten evaluiert. Es wurden verschiedene Arten von zufälligen und systematischen Fehlern in den Input-Daten der Methoden getestet. Alle Fehlerarten wurden auf der folgenden Konfiguration getestet: Graph Ortschaften, Szenario Emsland, räumliche Auflösung 10x10, 25.000 Trainingspfade. Ohne Verrauschung führt dieser Run bei der Methode *Cuboid Splatting* zu einem prozentualen Szenariofehler von 17,7% und zu einem prozentualen Pfadfehler von ca. 2,8% auf den Trainingspfaden und 3,7% auf den Testpfaden. Bei der Methode *Inverse Rendering* führt dieser Run zu einem prozentualen Szenariofehler von ca. 32,98% und einem prozentualen Pfadfehler von ca. 0,05% auf den Trainingspfaden und 2,16% auf den Testpfaden.

Während zur Methode *Cuboid Splatting* bislang keine Nutzung eines Priors existiert, wird bei der Methode *Inverse Rendering* auch noch die Verrauschung des genutzten Priors getestet. Daher gibt es hier zum Vergleich zusätzlich den obigen Run (Ortschaften, Emsland, 10x10, 25.000 Pfade) mit dem Prior „Eurim-Emsland“, der ohne Fehler zu folgenden Ergebnissen führt: prozentualer Szenariofehler 27,57%, Trainingspfadfehler 0,04% und Testpfadfehler 2,11%.

Die Arten von Datenfehlern, deren Auswirkung getestet wurden, werden in den nachfolgenden Tabellen ausführlich beschrieben. Hierbei werden zunächst die Arten der vorgenommenen Verrauschungen (Exposure noise, Exposure overestimation, Location variation, Prior noise, Prior rotation, Prior shift) definiert (Tabelle 5.1) und anschließend die Bedeutungen der einzelnen Ausprägungen der Verrauschungen erläutert, wobei jeweils eine mathematische und eine intuitive Erklärung vorliegen (Tabelle 5.2).

Tabelle 5.1: Definitionen der verschiedenen Arten von Verrauschung

Art	**Definition**
Exposure noise	Eine zufällige Verrauschung der durch das Messgerät aufgenommenen Schilddrüsenbelastungswerte in drei Abstufungen.
Exposure overestimation	Eine systematische Überschätzung der durch das Messgerät aufgenommenen Schilddrüsenbelastungswerte in drei Abstufungen. Hierbei wurde immer eine Überschätzung des betroffenen Expositionswertes um 20% angewandt.
Location variation	Eine zufällige Verrauschung der Ortsangaben in den Bewegungsprofilen in vier Abstufungen. Diese Art der Verrauschung ist in ihren Auswirkungen auf die Daten übertragbar auf Fehler in den Zeitangaben der Bewegungsprofile[1]. Zur Einführung dieser Art der Verrauschung in die Daten werden zunächst immer die anstelle der jeweiligen tatsächlichen Ortsangabe über den Graphen erreichbaren benachbarten Zellen in einer maximalen Entfernung von zwei Zellen ermittelt und dann die Verrauschung angewandt. Mit Zellen sind hierbei die Zellen der 10x10-Rasterung entlang der räumlichen Dimensionen (x- und y-Achse der Raumzeit) gemeint.

[1] Wenn Personen sich nicht darin verschätzen, an welchem Ort sie sich exakt aufgehalten haben, sondern darin, zu welcher Zeit dies exakt passiert ist, hat dies die gleiche Folge: eine fehlerhafte Ortsangabe zu einer bestimmten Zeit.

Prior noise	Eine zufällige Verrauschung der Werte des Priors der *Methode Inverse Rendering* in drei Abstufungen. Diese Art der Verrauschung wurde auf die gleiche Weise erzeugt, wie die zufällige Verrauschung der Expositionswerte.
Prior rotation	Eine systematische Verrauschung des Priors der *Methode Inverse Rendering* durch Drehung in zwei Abstufungen.
Prior shift	Eine systematische Verrauschung des Priors der Methode *Inverse Rendering* durch Verschiebung in zwei Abstufungen.

Tabelle 5.2: Übersicht über die Ausprägungen von Verrauschungen mit denen Experimente durchgeführt wurden (jeweils mathematische Bedeutung und intuitive Erklärung)

Ausprägung	**Mathematische Bedeutung**	**Intuitive Erklärung**
Exposure noise large	Diese Verrauschung wurde erzeugt durch die Multiplikation des Expositionswertes mit einem zufälligen Wert aus einer Gaussverteilung mit einer Standardabweichung von 0,2 um den Mittelwert 1.	Im realistischen Anwendungskontext bedeutet dies, dass der Großteil der Schilddrüsenmessungen (etwa zwei Drittel) um bis zu 20% verrauscht sind und etwa ein Drittel um mehr als 20%.
Exposure noise medium	Diese Verrauschung wurde erzeugt durch die Multiplikation des Expositionswertes mit einem zufälligen Wert aus einer Gaussverteilung mit einer Standardabweichung von 0,1 um den Mittelwert 1.	Im realistischen Anwendungskontext bedeutet dies, dass der Großteil der Schilddrüsenmessungen (etwa zwei Drittel) um bis zu 10% verrauscht sind und etwa ein Drittel um mehr als 10%.
Exposure noise small	Diese Verrauschung wurde erzeugt durch die Multiplikation des Expositionswertes mit einem zufälligen Wert aus einer Gaussverteilung mit einer Standardabweichung von 0,025 um den Mittelwert 1.	Im realistischen Anwendungskontext bedeutet dies, dass der Großteil der Schilddrüsenmessungen (etwa zwei Drittel) um bis zu 2,5% verrauscht sind und etwa ein Drittel um mehr als 2,5%.
Exposure overestimation large	Eine hohe systematische Überschätzung bedeutet, dass 100% der Pfadexpositionen überschätzt wurden.	Alle Messgeräte messen systematisch einen um 20% zu hohen Wert.
Exposure overestimation medium	Eine mittlere systematische Überschätzung bedeutet, dass 50% der Pfadexpositionen überschätzt wurden.	Eines von zwei Messgeräten misst systematisch einen um 20% zu hohen Wert.
Exposure overestimation small	Eine geringe systematische Überschätzung bedeutet, dass 10% der Pfadexpositionen überschätzt wurden.	Eines von zehn Messgeräten misst systematisch einen um 20% zu hohen Wert.

Location variation extra large	Es wird eine Gewichtung der erreichbaren benachbarten Zellen pro Ortsangabe über eine Standardabweichung von 2 Zellen ab der tatsächlichen Ortsangabe vorgenommen und die einzelnen Ortsangaben darüber zufällig verrauscht.	Die Ortsangaben der Bewegungsprofile sind sehr stark verrauscht und liegen in der Regel um ein bis zwei Zellen neben dem korrekten Ort.
Location variation large	Es wird eine Gewichtung der erreichbaren benachbarten Zellen pro Ortsangabe über eine Standardabweichung von 1 Zelle ab der tatsächlichen Ortsangabe vorgenommen und die einzelnen Ortsangaben darüber zufällig verrauscht.	Die Ortsangaben der Bewegungsprofile sind stark verrauscht und liegen in der Regel eine Zelle neben dem korrekten Ort.
Location variation medium	Es wird eine Gewichtung der erreichbaren benachbarten Zellen pro Ortsangabe über eine Standardabweichung von 0,5 Zellen ab der tatsächlichen Ortsangabe vorgenommen und die einzelnen Ortsangaben darüber zufällig verrauscht.	Die Ortsangaben der Bewegungsprofile sind verrauscht und liegen häufig um eine Zelle neben dem korrekten Ort.
Location variation small	Es wird eine Gewichtung der erreichbaren benachbarten Zellen pro Ortsangabe über eine Standardabweichung von 0,25 Zellen ab der tatsächlichen Ortsangabe vorgenommen und die einzelnen Ortsangaben darüber zufällig verrauscht.	Die Ortsangaben der Bewegungsprofile sind leicht verrauscht und liegen manchmal um eine Zelle neben dem korrekten Ort.
Prior noise large	Diese Verrauschung wurde durch die Multiplikation der Prior-Belastungswerte an den Raumzeitpunkten mit zufälligen Werten aus einer Gaussverteilung mit einer Standardabweichung von 0,2 um den Mittelwert 1 erzeugt.	Im Anwendungskontext bedeutet dies, dass der Großteil der Prior-Belastungswerte (etwa zwei Drittel) um bis zu 20% verrauscht sind und etwa ein Drittel um mehr als 20%.
Prior noise medium	Diese Verrauschung wurde durch die Multiplikation der Prior-Belastungswerte an den Raumzeitpunkten mit zufälligen Werten aus einer Gaussverteilung mit einer Standardabweichung von 0,1 um den Mittelwert 1 erzeugt.	Im Anwendungskontext bedeutet dies, dass der Großteil der Prior-Belastungswerte (etwa zwei Drittel) um bis zu 10% verrauscht sind und etwa ein Drittel um mehr als 10%.

Prior noise small	Diese Verrauschung wurde durch die Multiplikation der Prior-Belastungswerte an den Raumzeitpunkten mit zufälligen Werten aus einer Gaussverteilung mit einer Standardabweichung von 0,025 um den Mittelwert 1 erzeugt.	Im Anwendungskontext bedeutet dies, dass der Großteil der Prior-Belastungswerte (etwa zwei Drittel) um bis zu 2,5% verrauscht sind und etwa ein Drittel um mehr als 2,5%.
Prior rotation medium	Der zu Beginn des Modelltrainings geladene Prior als initiale Belastungswerte der Raumzeitpunkte wurde um 30 Grad bezüglich des Mittelpunkts des Szenarios gedreht.	Der geladene Prior enthält sinnvolle Werte, die sich durch die Drehung aber an den falschen Stellen im Raum befinden.
Prior rotation small	Der zu Beginn des Modelltrainings geladene Prior als initiale Belastungswerte der Raumzeitpunkte wurde um 10 Grad bezüglich des Mittelpunkts des Szenarios gedreht.	Der geladene Prior enthält sinnvolle Werte, die sich durch die Drehung aber an den falschen Stellen im Raum befinden. Sie befinden sich dennoch in der Nähe der richtigen Stellen im Raum.
Prior shift medium	Der zu Beginn des Modelltrainings geladene Prior als initiale Belastungswerte der Raumzeitpunkte wurde räumlich entlang der x-Achse um 38500 Einheiten (Meter) verschoben.	Der geladene Prior enthält sinnvolle Werte, die sich durch die Verschiebung aber etwa 38,5 km vom richtigen Ort entfernt befinden. Bei einer 10x10 Rasterung entspricht dies einer Verschiebung um eine Zelle.
Prior shift small	Der zu Beginn des Modelltrainings geladene Prior als initiale Belastungswerte der Raumzeitpunkte wurde räumlich entlang der x-Achse um 76900 Einheiten (Meter) verschoben.	Der geladene Prior enthält sinnvolle Werte, die sich durch die Verschiebung aber etwa 76,9 km vom richtigen Ort entfernt befinden. Bei einer 10x10 Rasterung entspricht dies einer Verschiebung um zwei Zellen.

Die Ergebnisse der Experimente mit den verrauschten Daten für die Methode *Cuboid Splatting* sind in der untenstehenden Tabelle (Tabelle 5.3) zu finden. Verglichen werden der prozentuale Szenariofehler von Punkten mit der Relevanz 1, 5 und 10 und der prozentuale Pfadfehler auf den Trainingspfaden und den Testpfaden. Bei den Trainingspfaden handelt es sich dabei um die verrauschten Pfaddaten und bei den Testpfaden um fehlerfreie Pfaddaten. Für die Methode *Cuboid Splatting* wurden die Fehlerarten Exposure noise, Exposure overestimation und Location variation getestet.

Die Ergebnisse der Experimente mit den verrauschten Daten für die Methode *Inverse Rendering* sind in der darauffolgenden Tabelle (Tabelle 5.4) zu finden. Verglichen werden auch hier der prozentuale Szenario Fehler von Punkten mit der Relevanz 1, 5 und 10 und der prozentuale Pfadfehler auf dem Set der Trainingspfade (Trainingsdaten mit Verrauschung) und der Testpfade (dem Modell unbekannte Testdaten ohne Verrauschung). Bei der Methode *Inverse Rendering* wurden zusätzlich die Fehlerarten Prior noise, Prior rotation und Prior shift getestet. Bei den Experimenten mit Exposure- und Location-Verrauschung wurde ohne Prior gearbeitet und bei den Experimenten mit Prior-Verrauschung mit Prior.

*Tabelle 5.3: Ergebnisse der Experimente mit der Methode **Cuboid Splatting** in der Konfiguration Ortschaften-Emsland-10x10-25.000 Pfade mit den verschiedenen Arten von verrauschten Daten*

ID	Verrauschung	% scen 1 rel sum	% scen 5 rel sum	% scen 10 rel sum	% train rel sum	% test rel sum
1018	keine	17.7439	15.804	15.0946	2.7523	3.6999
1064	exposure noise large	42.7284	33.2616	27.2014	21.4003	11.6169
1065	exposure noise medium	34.469	32.5524	31.3627	11.4696	11.5482
1066	exposure noise small	29.5327	24.6401	20.8843	4.8945	5.7049
1067	exposure overestimation large	31.8594	29.6116	27.8482	3.9421	21.9026
1068	exposure overestimation medium	50.9225	45.6613	40.7961	11.2372	17.0831
1069	exposure overestimation small	29.4439	24.5312	21.8459	6.8988	6.3986
1070	location variation extra large	129.997	130.0213	126.4079	90.237	39.6597
1071	location variation large	79.7119	72.6452	63.2696	49.6698	20.9168
1087	location variation medium	66.4917	58.9782	48.2845	24.9339	17.5265
1088	location variation small	36.7039	27.0069	21.4312	8.6715	8.6573

*Tabelle 5.4: Ergebnisse der Experimente mit der Methode **Inverse Rendering** in der Konfiguration Ortschaften-Emsland-10x10-25.000 Pfade mit den verschiedenen Arten von verrauschten Daten*

ID	Verrauschung	% scen 1 rel sum	% scen 5 rel sum	% scen 10 rel sum	% train rel sum	% test rel sum
2000	keine (ohne Prior)	32.9829	24.0683	21.212	0.0528	2.155
2009	keine (mit Prior)	27.5718	20.3348	17.9306	0.0366	2.1138
3098	exposure noise large	151.3881	115.8635	103.5601	9.8058	17.6418
3097	exposure noise medium	107.3527	84.672	72.0575	4.8266	12.836
3096	exposure noise small	45.4266	35.5068	30.8744	0.9498	4.0075
3101	exposure overestimation large	40.6669	32.0538	29.1105	0.0481	19.8759
3100	exposure overestimation medium	88.1823	67.7076	59.7157	2.2413	7.0648
3099	exposure overestimation small	47.4098	36.0173	30.2855	0.7947	4.9166
3103	location variation extra large	392.6951	334.7978	310.9298	51.7513	271.6188
3102	location variation large	278.4387	200.7515	157.9771	28.693	33.0724
3111	location variation medium	247.7962	164.8502	150.9697	8.1347	25.6271
3112	location variation small	134.2218	87.6596	69.9048	2.7517	19.1235
3106	prior noise large	27.3937	20.0445	17.8856	0.0397	1.7123
3105	prior noise medium	27.509	20.4659	17.8635	0.0452	1.7189
3104	prior noise small	27.9813	20.6526	18.1762	0.0416	1.8176
3108	prior rotation medium	27.4497	20.293	17.6366	0.0395	1.7206
3107	prior rotation small	27.3094	20.0286	17.2966	0.0416	1.6077
3110	prior shift medium	27.5687	20.1585	17.5571	0.0436	1.9687
3109	prior shift small	27.6092	20.2765	17.6547	0.0487	1.7715

5.1 Auswirkungen zufälliger Fehler in den Schilddrüsenmessungen

Cuboid Splatting: Eine zufällige Verrauschung der Pfadexpositionen (durch das Messgerät aufgenommenen Schilddrüsenbelastungswerte) führt bei der Methode *Cuboid Splatting* zu einer deutlichen Verschlechterung der erreichten Szenario- und Pfadrekonstruktionen. Der prozentuale Trainingspfadfehler wird hier auf verrauschten Trainingspfaden gemessen und der prozentuale Testpfadfehler auf fehlerfreien Testpfaden. Der prozentuale Szenariofehler wird genauso gemessen wie zuvor. Die nachfolgend im Detail analysierten Ergebnisse, basieren auf Tabelle 5.3 und werden in der Abbildung 5.1 visualisiert.

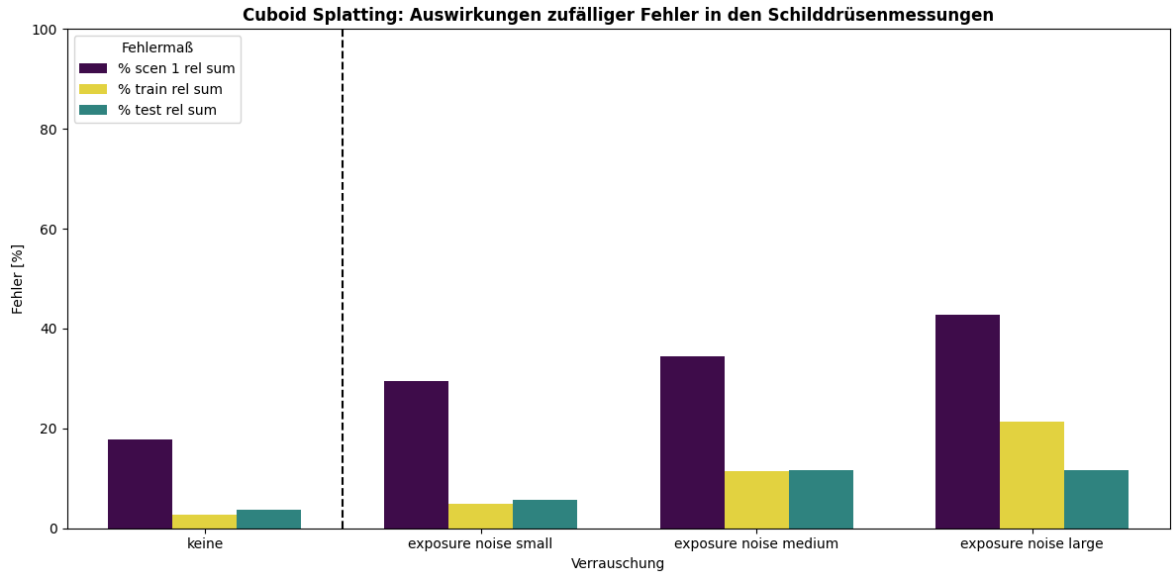

Abbildung 5.1: Auswirkungen zufälliger Fehler in den Schilddrüsenmessungen in drei Abstufungen (small, medium, large) auf die Fehlermaße der Rekonstruktionen im Vergleich zur Rekonstruktion mit fehlerfreien Daten mit der Methode ***Cuboid Splatting***

Bei einer mittleren Verrauschung (Verrauschung mit Standardabweichung 0,1, Bedeutung siehe Tabelle 5.2) kommt es zu einer ungefähren Verdopplung des prozentualen Szenariofehlers von 17,7% auf 34,5%. Sowohl der Trainings- als auch Testpfadfehler liegen nun bei ca. 11,5% anstelle von etwa 3%. Eine höhere Verrauschung mit einer Standardweichung von 0,2 führt wie erwartet zu einem noch höheren Fehler in der Szenariorekonstruktion (nun 42,7% anstelle von 17,7%). Interessanterweise führt dies zu einem ebenfalls deutlich höheren Fehler auf den Trainingspfaden (21,4%), jedoch im Vergleich zur mittleren Verrauschung nicht zu einem höheren Fehler auf den Testpfaden. Dem Modell ist es also nicht mehr richtig möglich, alle Trainingspfade sinnvoll zu rekonstruieren, da diese stark verrauscht sind, die Generalisierung des Modells auf unverrauschte Testpfade funktioniert aber weiterhin recht gut. Eine niedrigere Verrauschung mit einer Standardabweichung von 0,025 führt zu deutlich weniger verschlechterten Rekonstruktionsergebnissen, mit einem Szenariofehler von 29,5% (also ca. 12% schlechter als ohne Verrauschung) und Pfadfehlern von etwa 5% auf den Trainings- und Testdaten (im Vergleich zu etwa 3% ohne Verrauschung).

In allen drei Ausprägungen der zufälligen Verrauschung wird deutlich, dass die Relevanz der Punkte (also ob mindestens 1, 5 oder 10 Pfade durch einen Raumzeitpunkt geführt haben), beim Auftreten von Fehlern in den Daten wesentlich mehr Einfluss auf das Szenario-Rekonstruktionsergebnis hat, als wenn die Daten fehlerfrei sind. Dies lässt sich bei einem Vergleich der Ergebnisse in Tabelle 5.3 mit den Ergebnissen in den Tabellen aus dem vorherigen Kapitel (Tabellen 4.6, 4.8, 4.10) feststellen. Raumzeitpunkte durch die mehr Pfade geführt haben,

werden im Falle von fehlerbehafteten Daten mit der Methode *Cuboid Splatting* deutlich besser rekonstruiert als Punkte, durch die weniger Pfade geführt haben. Dieses Ergebnis erscheint sehr logisch, da im Falle von Fehlern eine höhere Überlappung in einem Raumzeitpunkt zu mehr gegensätzlicher, richtiger Evidenz für den Belastungswert dieses Punktes führt und die Rekonstruktion dieses Punktes, wenn beispielweise vier relativ korrekt gemessene Pfade und ein deutlich fehlerhaft gemessener Pfad durch diesen Punkt verlaufen eher richtig sein wird, als wenn beispielsweise nur ein fehlerhaft gemessener Pfad durch diesen verläuft. Die Szenario-Rekonstruktion in der Raumzeit ist also im Fall von fehlerhaften Messungen an den Raumzeitpunkten mit mehr Überlappung deutlich robuster. Zudem hat sich die Methode *Cuboid Splatting* in diesem Experiment mit zufälliger Verrauschung der Pfadexpositionen als deutlich robuster erwiesen als die Methode *Inverse Rendering*, deren Ergebnisse als nächstes präsentiert werden.

Inverse Rendering: Auch bei der Methode *Inverse Rendering* führt eine zufällige Verrauschung der Pfadexpositionen (durch das Messgerät aufgenommenen Schilddrüsenbelastungswerte) zu einer deutlichen Verschlechterung der erreichten Szenario- und Pfadrekonstruktionen. Die Verschlechterung fällt hierbei gravierender aus als bei der Methode *Cuboid Splatting*. Die nachfolgend im Detail analysierten Ergebnisse basieren auf Tabelle 5.4 und werden in Abbildung 5.2 visualisiert. Bei der Abbildung 5.2 zu *Inverse Rendering* ist die Veränderung der Skalierung der y-Achse auf maximal 200% im Vergleich zur Skalierung der Abbildung 5.1 zu *Cuboid Splatting* auf maximal 100% zu beachten.

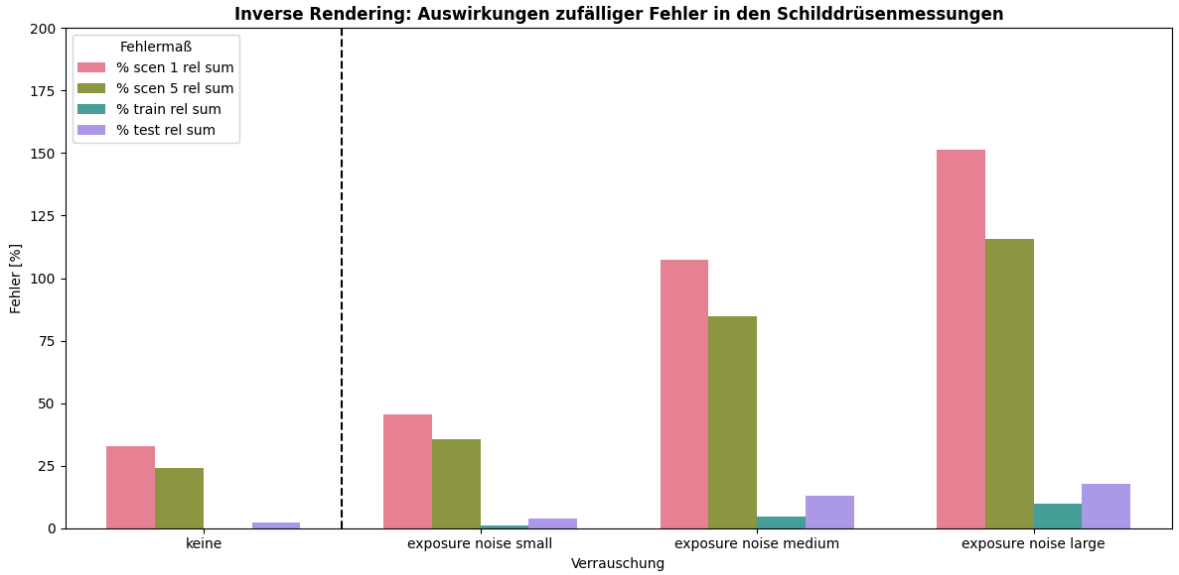

Abbildung 5.2: Auswirkungen zufälliger Fehler in den Schilddrüsenmessungen in drei Abstufungen (small, medium, large) auf die Fehlermaße der Rekonstruktionen im Vergleich zur Rekonstruktion mit fehlerfreien Daten mit der Methode ***Inverse Rendering***

Bei der Methode *Inverse Rendering* führt die geringe zufällige Verrauschung (mit einer Standardabweichung von 0,025) der Schilddrüsenmessungen zu einer Verschlechterung des prozentualen Szenariofehlers von 32,98% auf 45,43%. Auch der prozentuale Trainings- und Testpfadfehler verschlechtern sich deutlich (von 0,05% auf 0,95% und von 2,16% auf 4,01%). Die damit erreichten Rekonstruktionen sind also schlechter, aber dennoch weiterhin nutzbar.

Anders sieht es bei der mittleren und der hohen Verrauschung (mit einer Standardabweichung von 0,1 bzw. 0,2) aus. Beide führen in der Rekonstruktion zu prozentualen Szenariofehlern von über 100% (mittlere Verrauschung: 107,35%, hohe Verrauschung: 151,39%) und damit zu

unbrauchbaren Rekonstruktionsergebnissen bezüglich der Szenario-Rekonstruktion. Auch der Fehler bei der Rekonstruktion der Trainings- und Testpfade steigt hier deutlich. Der Rekonstruktionsfehler der Testpfade liegt aber in beiden Fällen unter 20% (mittlere Verrauschung: 12,84%, hohe Verrauschung: 17,64%), sodass zumindest diese Rekonstruktionen weiterhin auch bei mittlerer und höherer Verrauschung nutzbar sind.

Auffällig ist hier, dass anders als bei der Methode *Cuboid Splatting* bei höherer Verrauschung der prozentuale Pfadfehler auf den fehlerfreien Testdaten höher ist, als auf den fehlerhaften Trainingsdaten. Während es die Methode *Cuboid Splatting* also trotz Fehlern in den Trainingsdaten schafft, ein relativ gut generalisierendes Modell zu erreichen (Trainingspfadfehler: 21,4% vs. Testpfadfehler: 11.6%), ist das bei der Methode *Inverse Rendering* nicht der Fall (Trainingspfadfehler: 9,8% vs. Testpfadfehler: 17,6%). Stattdessen findet hier ein Overfitting auf die fehlerbehafteten Trainingspfade statt (Trainingspfadfehler deutlich niedriger als Testpfadfehler), sodass die Generalisierung auf die Testpfade deutlich schlechter ausfällt.

Wie auch bei der Methode *Cuboid Splatting*, ist in den Ergebnissen der Experimente mit zufällig verrauschten Expositionen mit der Methode *Inverse Rendering* erkennbar, dass Raumzeitpunkte mit steigender Relevanz (5 oder 10) besser rekonstruiert werden können, als Raumzeitpunkte mit Relevanz 1 (siehe Tabelle 5.4). Auch hier hilft es also, wenn mehr gegensätzliche, richtige Evidenz im Vergleich zu den fehlerhaften Pfaden in einem Raumzeitpunkt vorliegt, um die vorliegenden Fehler auszugleichen.

5.2 Auswirkungen systematischer Fehler in den Schilddrüsenmessungen

Cuboid Splatting: Auch eine systematische Überschätzung der Pfadexpositionen (durch Messgeräte aufgenommenen Schilddrüsenbelastungswerte) führt bei der Methode *Cuboid Splatting* zu einer deutlichen Verschlechterung der erreichten Szenario- und Pfadrekonstruktionen. Wie zuvor, wird der prozentuale Trainingspfadfehler auf verrauschten Trainingspfaden gemessen und der prozentuale Testpfadfehler auf fehlerfreien Testpfaden. Die nachfolgend im Detail analysierten Ergebnisse, basieren auf Tabelle 5.3 und werden in der Abbildung 5.3 visualisiert.

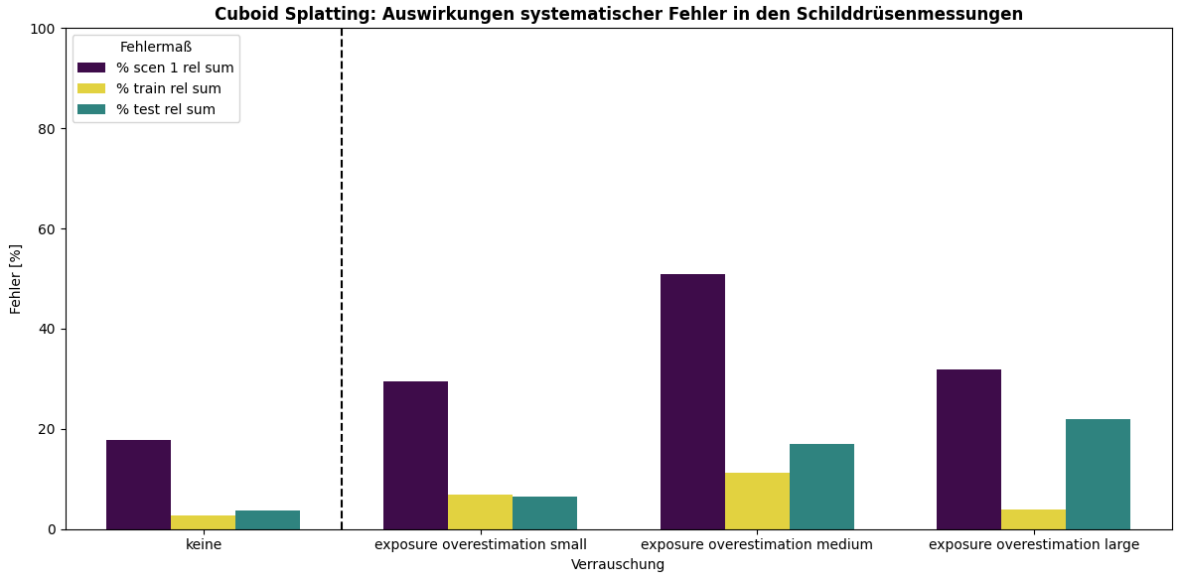

Abbildung 5.3: Auswirkungen systematischer Fehler in den Schilddrüsenmessungen in drei Abstufungen (small, medium, large) auf die Fehlermaße der Rekonstruktionen im Vergleich zur Rekonstruktion mit fehlerfreien Daten mit der Methode ***Cuboid Splatting***

Die hier in den Experimenten entstandenen Ergebnisse, basierend auf einer geringen, mittleren und hohe Verrauschung, zeichnen ein interessantes Bild. Wie in Tabelle 5.2 beschrieben, bedeutet in diesem Fall geringe systematische Überschätzung, dass 10% der Pfadexpositionen überschätzt wurden (quasi: eines von zehn Messgeräten misst systematisch einen um 20% zu hohen Wert), mittlere Verrauschung, dass die Hälfte der Pfadexpositionen überschätzt wurden (quasi: eines von zwei Messgeräten misst systematisch einen um 20% zu hohen Wert) und hohe Verrauschung, dass alle Pfadexpositionen überschätzt wurden (quasi: alle Messgeräte messen systematisch einen um 20% zu hohen Wert).

Im Falle der geringen Verrauschung (Überschätzung bei 10% der Messungen) kommt es zu einer Verschlechterung der Szenario-Rekonstruktion um etwa 12% (von 17,7% auf 29,4%) und der Pfadrekonstruktion (von etwa 3% auf etwa 6% bzgl. der Trainings- und Testpfade). Die Verschlechterung durch die geringe systematische Verrauschung ist ähnlich stark ausgeprägt, wie bei der geringen zufälligen Verrauschung (siehe Abschnitt 5.1). Zudem liegt der Testpfadfehler sogar etwas geringer als der Trainingspfadfehler (6,4% vs. 6,9%), was darauf hindeutet, dass die Trainingspfade aufgrund der in ihnen auftretenden Widersprüchliche bezüglich der Messungen durch 10% fehlerbehaftete Pfade insgesamt nicht so gut rekonstruiert werden können, aber dennoch die Rekonstruktion für die fehlerfreien Testpfade gut funktioniert. Dahingehend erscheint die Methode bezüglich der Pfadrekonstruktion, solange es nur einen kleinen Anteil überschätzter Pfade gibt, relativ robust.

Im Falle der hohen Verrauschung (Überschätzung bei 100% der Messungen) zeigt sich ein etwas anderes Bild. Während der entstandene Fehler in der Szenario-Rekonstruktion kaum höher ist als bei der geringen systematischen Verrauschung, zeigt der prozentuale Trainings- und Testpfadfehler ein anderes Muster. Hier liegt der Trainingspfad-Fehler ähnlich gering, wie ohne Verrauschung (3,9% mit Verrauschung, 2,7% ohne Verrauschung). Das Modell kann also die Pfade, mit denen es trainiert wurde, sehr gut rekonstruieren. Dies lässt sich damit erklären, dass sich diese Pfade untereinander nicht widersprechen, da die gesamten 100% der Pfade auf die gleiche systematische Art und Weise überschätzt wurden. Bei der Rekonstruktion der Testpfade, die nicht verrauscht wurden, liegt das Modell dann aber um 21,9% daneben. Aufgrund der systematischen Überschätzung der Trainingspfade, ist die Rekonstruktion der echten Exposition der Testpfade also wesentlich schlechter. Es ist davon auszugehen, dass das Modell diese nun auch systematisch um etwa 20% überschätzt.

Die mittlere Verrauschung (Überschätzung bei 50% der Messungen) hat den stärksten negativen Effekt auf den Fehler in der Szenario-Rekonstruktion. Hier liegt fast eine Verdreifachung des prozentualen Fehlers von 17,7% auf 50,9% vor. Dies lässt sich damit erklären, dass bei einer Überschätzung der Hälfte der Expositionen der Trainingspfade besonders hohe Widersprüche in den Trainingsdaten vorliegen. Dadurch ist es für das Modell insgesamt schwieriger, eine geeignete Rekonstruktion zu erzielen. Auch der prozentuale Fehler auf den Trainingspfaden ist im Vergleich zur geringen und hohen Verrauschung besonders hoch (11,2%), was zeigt, dass es dem Modell so nicht gelingen kann, eine zu allen Trainingspfaden gut passende Rekonstruktion zu erzeugen. Der Testpfadfehler liegt mit etwa 17,1% fast so hoch wie bei der hohen Verrauschung. Das Modell kann also auch ungesehene Pfade ähnlich schlecht rekonstruieren.

Wie bei der zufälligen Verrauschung der Pfadexpositionen, zeigt sich auch bei systematischer Verrauschung, dass mit steigender Relevanz eines Raumzeitpunktes (steigender Überlappung in diesem Punkt) die Abweichung in der Szenario-Rekonstruktion insgesamt kleiner, also besser, wird. Somit ist auch hier das Modell an den Stellen gegen die Verrauschung robuster, an denen mehr Evidenz vorliegt. Insgesamt erweist sich die Methode *Cuboid Splatting* auch bei dieser Art

der Verrauschung als robuster als die Methode *Inverse Rendering*, deren Ergebnisse nachfolgend präsentiert werden.

Inverse Rendering: Auch bei der Methode *Inverse Rendering* führt eine systematische Überschätzung der Pfadexpositionen (durch die Messgeräte aufgenommene Schilddrüsenbelastungswerte) zu einer deutlichen Verschlechterung der erreichten Szenario- und Pfadrekonstruktionen. Im Vergleich zu den Ergebnissen der Experimente mit zufälligen Fehlern in der Schilddrüsenmessungen und den Ortsangaben der Bewegungsprofile, wirkt sich diese Fehlerart bei der Methode *Inverse Rendering* am wenigsten stark aus. Die nachfolgend im Detail analysierten Ergebnisse, basieren auf Tabelle 5.4 und werden in Abbildung 5.4 visualisiert. Die Skalierung der y-Achse geht hier wie bei der Abbildung 5.3 zu *Cuboid Splatting* bis zu einem Wert von 100%.

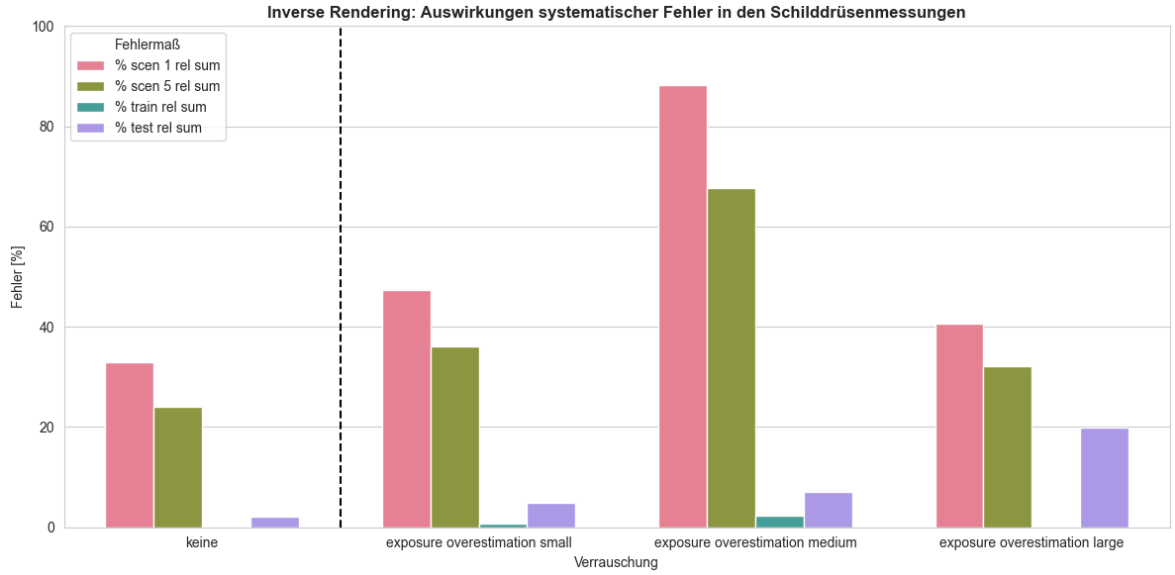

Abbildung 5.4: Auswirkungen systematischer Fehler in den Schilddrüsenmessungen in drei Abstufungen (small, medium, large) auf die Fehlermaße der Rekonstruktionen im Vergleich zur Rekonstruktion mit fehlerfreien Daten mit der Methode ***Inverse Rendering***

Die Ergebnisse der Experimente, basierend auf einer geringen, mittleren und hohen Verrauschung, mit der Methode *Inverse Rendering* zeichnen insgesamt ein sehr ähnliches Bild wie die Ergebnisse mit der Methode *Cuboid Splatting*. Zur Erinnerung: Hierbei wurden je 10% der Pfadexpositionen (geringe Verrauschung), 50% der Pfadexpositionen (mittlere Verrauschung) und 100% der Pfadexpositionen (hohe Verrauschung) um 20% überschätzt.

Wie auch bei der Methode *Cuboid Splatting*, wirkt sich die mittlere Verrauschung am gravierendsten verschlechternd auf die Qualität der Rekonstruktion aus. Im Falle von *Inverse Rendering* führt dies zu einer Verschlechterung der Qualität von 32,98% auf 88,18% beim prozentualen Szenariofehler. Wir erklären dies damit, dass beim Training mit 50% fehlerhaften und 50% fehlerfreien Pfaden, die meisten Widersprüche in den Daten vorliegen und damit die Rekonstruktion am schlechtesten gelingt.

Weniger stark wirkt sich die geringe Verrauschung aus (Anstieg des Szenariofehlers von 32,98% auf 47,4%). Der geringere Anteil an überschätzten Pfaden (1 von 10 Pfaden), kann hier also deutlich besser mit der Evidenz aus den fehlerfreien Pfaden (9 von 10 Pfaden) ausgeglichen werden. Am wenigsten stark bezüglich des prozentualen Szenariofehlers wirkt sich die hohe Verrauschung (Überschätzung aller Pfade) aus. Der Szenariofehler steigt hier von 32,98% auf

40,67%. Dies ist damit zu erklären, dass bei einer systematischen Überschätzung der Expositionen aller Pfade zwar fehlerhafte Daten vorliegen, diese sich aber in sich nicht widersprechen, was der Rekonstruktion mit der Methode *Inverse Rendering* sehr zuträglich ist. Gleichzeitig liegt hier, wie auch bei der Methode *Cuboid Splatting*, dann der höchste prozentuale Fehler auf den Testpfaden von 19,88% vor. Da alle Trainingspfade, mit denen das Modell trainiert wurde, um 20% überschätzt waren, ist davon auszugehen, dass nun auch die Rekonstruktionen der Expositionen der Testpfade um etwa 20% überschätzt werden.

Insgesamt bleiben die Ergebnisse mit der Methode *Inverse Rendering* bei dieser Art der Verrauschung brauchbar (im Sinne eines Szenariofehlers <100%). Dennoch wirkt sich die Verrauschung stärker aus, als bei der Methode *Cuboid Splatting*, welche auch hier als die robustere Methode einzustufen ist. Auch mit der Methode *Inverse Rendering* werden hier wieder Belastungswerte von Raumzeitpunkten mit steigender Relevanz besser rekonstruiert. Also wirkt sich auch hier wieder mehr (fehlerfreie) Evidenz positiv auf die Rekonstruktion der Belastungswerte der Raumzeitpunkte aus.

5.3 Auswirkungen zufälliger Fehler in den Ortsangaben der Bewegungsprofile

Cuboid Splatting: Ebenso wie die Verrauschung der Pfadexpositionswerte, führt auch die zufällige Verrauschung der Ortsangaben in den Bewegungsprofilen bei der Methode *Cuboid Splatting* zu einer deutlichen Verschlechterung der erreichten Szenario- und Pfadrekonstruktionen. Im Vergleich zur Verrauschung der Pfadbelastungswerte, wirkt sich diese Art von Verrauschung sogar besonders stark aus. Auch hier wird der prozentuale Trainingspfadfehler auf verrauschten Trainingspfaden gemessen und der prozentuale Testpfadfehler auf fehlerfreien Testpfaden. Die nachfolgend im Detail analysierten Ergebnisse, basieren auf Tabelle 5.3 und werden in der Abbildung 5.5 visualisiert.

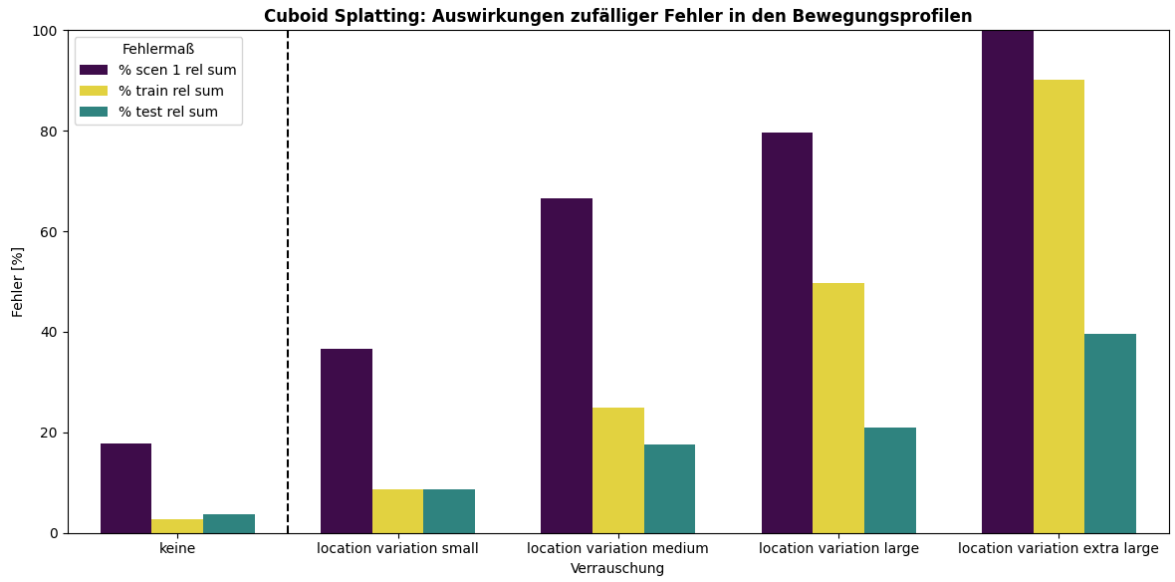

*Abbildung 5.5: Auswirkungen zufälliger Fehler in den Ortsangaben der Bewegungsprofile in vier Abstufungen (small, medium, large, extra large) auf die Fehlermaße der Rekonstruktionen im Vergleich zur Rekonstruktion mit fehlerfreien Daten mit **Cuboid Splatting***

Bereits die geringste getestete Verrauschung der Ortsangaben entlang der Pfade, also die Variation einzelner Ortsangaben in benachbarte Rasterzellen der Raumzeit, führt zu einer Verdopplung des prozentualen Fehlers der Szenario-Rekonstruktion (von etwa 17,7% auf 36,7%). Auch Trainings- und Testpfadfehler steigen hier bereits von etwa 3% ohne Verrauschung auf etwa

8%. Höhere Verrauschungen führen zu weitaus höheren Fehlern in den Szenario- und Pfadrekonstruktionen (mittlere und hohe Verrauschung: Szenariofehler 66,5% bzw. 79,7%) bis hin zur kompletten Unbrauchbarkeit der Szenario-Rekonstruktion (sehr hohe Verrauschung: Szenariofehler >100%). Diese tritt auf, wenn der Großteil der Ortsangaben in den Bewegungsprofilen fehlerhaft ist.

Beim Vergleich des prozentualen Fehlers auf den Trainingsdaten mit dem Fehler auf den Testdaten bei den verschiedenen Verrauschungs-Niveaus fällt auf, dass der Rekonstruktionsfehler auf den Trainingspfaden höher liegt als der Rekonstruktionsfehler auf den Testpfaden. Dies erscheint plausibel, da die Verrauschung der Trainingspfade zu viel widersprüchlicher Evidenz bezüglich der Belastungswerte in der Raumzeit führt, die das Modell nicht komplett nachzeichnen und rekonstruieren kann. Dennoch fällt die Rekonstruktion der Belastungen der fehlerfreien Testpfade besser aus, was zeigt, dass das Modell dennoch weiterhin recht gut generalisiert und bezüglich der Pfadrekonstruktionen eine gewisse Robustheit gegenüber Fehlern in den Trainingsdaten aufweist.

Wie schon bei den anderen beiden Arten der Verrauschung fällt gegenüber fehlerfreien Trainingsdaten hier die Relevanz der Raumzeitpunkte stärker ins Gewicht. Die Robustheit der Szenariorekonstruktionen ist generell dort höher, wo sich mehr Pfade überlappen und dadurch mehr potenziell richtige Evidenz einzelne Fehler in den Daten aufwiegen kann. Dieser Zusammenhang besteht nur bei der sehr hohen Verrauschung nicht mehr oder zumindest reicht eine Überlappung durch mindestens fünf Pfade hier nicht mehr aus, um zu besseren Rekonstruktionen zu führen.

Inverse Rendering: Auch bei der Methode *Inverse Rendering* führt die zufällige Verrauschung der Ortsangaben in den Bewegungsprofilen zu einer deutlichen Verschlechterung der erreichten Szenario- und Pfadrekonstruktionen. Wie auch bei der Methode *Cuboid Splatting* handelt es sich hierbei um die gravierendste Art der Verrauschung für die Qualität der Rekonstruktionen. Die nachfolgend im Detail analysierten Ergebnisse, basieren auf Tabelle 5.4 und werden in Abbildung 5.6 visualisiert. Bei der Visualisierung ist zu beachten, dass die y-Achse in diesem Fall bis zu einem Fehler-Wert von 450% skaliert ist, während die Skalierung bei der Methode *Cuboid Splatting* nur bis 100% ging.

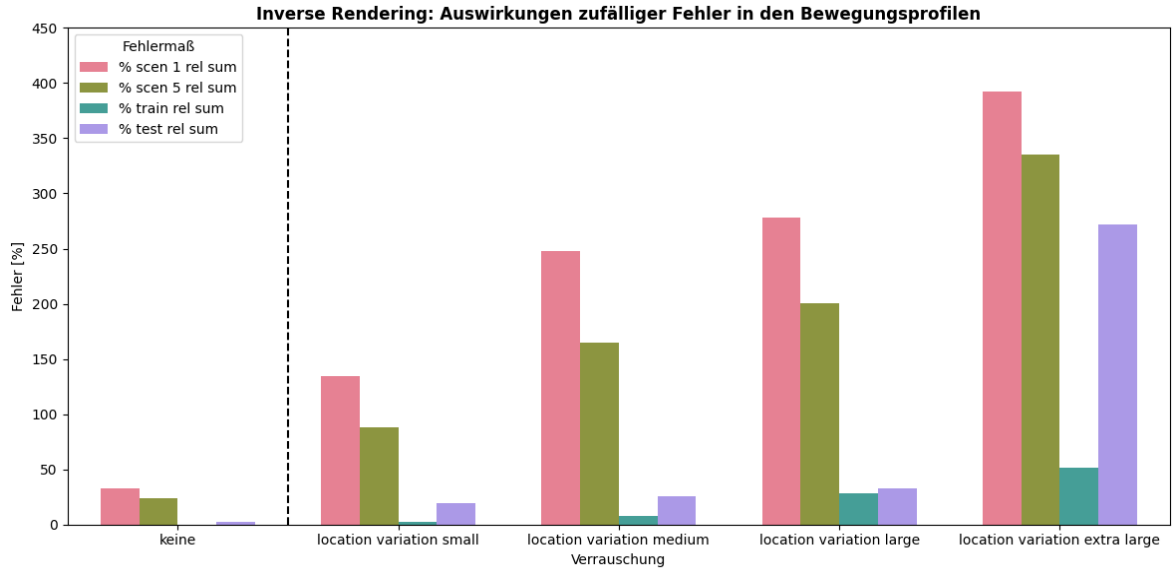

*Abbildung 5.6: Auswirkungen zufälliger Fehler in den Ortsangaben der Bewegungsprofile in vier Abstufungen (small, medium, large, extra large) auf die Fehlermaße der Rekonstruktionen im Vergleich zur Rekonstruktion mit fehlerfreien Daten mit **Inverse Rendering***

Bei der Methode *Inverse Rendering* führt bereits die geringste getestete Ausprägung der Verrauschung der Ortsangaben in den Bewegungsprofilen zu einem prozentualen Szenariofehler von über 100% und damit zu unbrauchbaren Szenario-Rekonstruktionen. Bei mittlerer, hoher und sehr hoher Verrauschung steigt der Szenariofehler noch erheblich weiter. Zur Übersicht: Szenariofehler ohne Verrauschung: 32,98%, geringe Verrauschung: 134,22%, mittlere Verrauschung: 247,8%, hohe Verrasuchung: 278,44% und sehr hohe Verrauschung: 392,7%. Dementsprechend ist die Methode *Inverse Rendering* auch hier deutlich weniger robust als die Methode *Cuboid Splatting*, wo erst bei einer sehr hohen Verrauschung der prozentuale Szenariofehler über 100% steigt.

Im Unterschied zur Methode *Cuboid Splatting* ist das auf den fehlerhaften Trainingspfaden trainierte Modell mit der Methode *Inverse Rendering* auch deutlich schlechter auf die fehlerfreien Testpfade generalisierbar. Während bei der Methode *Cuboid Splatting* der prozentuale Testpfadfehler durchweg niedriger ausfällt als der prozentuale Trainingspfadfehler, was bedeutet, dass die Trainingspfade aufgrund der darin enthaltenen Widersprüche nicht so gut rekonstruiert werden, die allgemeinen Lösungen, die das Modell findet, aber dennoch relativ gut die tatsächlichen Zusammenhänge treffen, ist es bei der Methode *Inverse Rendering* umgekehrt. Dort liegt ein recht niedriger Trainingspfadfehler vor, der Testpfadfehler ist aber deutlich höher, was bedeutet, dass ein Overfitting auf die fehlerhaften Trainingspfade stattfindet und das Modell es aber nicht schafft, die generellen Zusammenhänge treffend zu rekonstruieren.

Auch bei dieser Art der Verrauschung fällt die Bedeutung des Konzeptes der Relevanz der Raumzeitpunkte wieder auf. Die Punkte mit einer Relevanz von 5 oder 10 werden hier mit der Methode *Inverse Rendering* wie auch bei der Methode *Cuboid Splatting* messbar besser rekonstruiert. Insgesamt zeigen diese Experimente mit verrauschten Ortsangaben in den Bewegungsprofilen noch einmal deutlich die wesentlich höhere Robustheit der Methode *Cuboid Splatting* gegenüber der Methode *Inverse Rendering*.

5.4 Auswirkungen verschiedener Arten von Fehlern im Prior

Wie in den einführenden Erklärungen (siehe Tabellen 5.1 und 5.2) dargelegt, wurden die verschiedenen Arten von Fehlern im Prior (zufällige Verrauschung der Belastungswerte im Prior, Drehung des Priors, Verschiebung des Priors) nur für die Methode *Inverse Rendering* getestet, da nur für diese Methode bislang die Nutzung eines Priors vorgesehen und implementiert ist. Bei diesen Experimenten ist zudem zu beachten, dass bereits der unverrauschte Prior deutliche Abweichungen zu den realen Belastungswerten enthält und die Verrauschung damit zu einer zusätzlichen Verringerung der Qualität des Priors führt. Die ausführlichen Ergebnisse dieser Experimente finden sich in Tabelle 5.4 und werden nachfolgend analysiert. Die Abbildung 5.7 visualisiert die Zusammenhänge.

Die Ergebnisse der Experimente mit verschiedenen Arten von Prior-Verrauschung unterscheiden sich so gut wie gar nicht, weshalb sie hier gemeinsam ausgewertet werden. Insgesamt führt bei der Methode *Inverse Rendering* die Nutzung eines unverrauschten Priors im Vergleich zur Anwendung der Methode ohne Prior zu einem besseren Ergebnis in Bezug auf den prozentualen Szenariofehler: 27,57% mit Prior im Vergleich zu 32,98% ohne Prior. Zur Erinnerung: Beim Prior, in der Form wie er hier getestet wurde, handelt es sich um die Initialisierung der Belastungswerte in der Raumzeit, welche ohne Prior einfach mit 0 initialisiert werden.

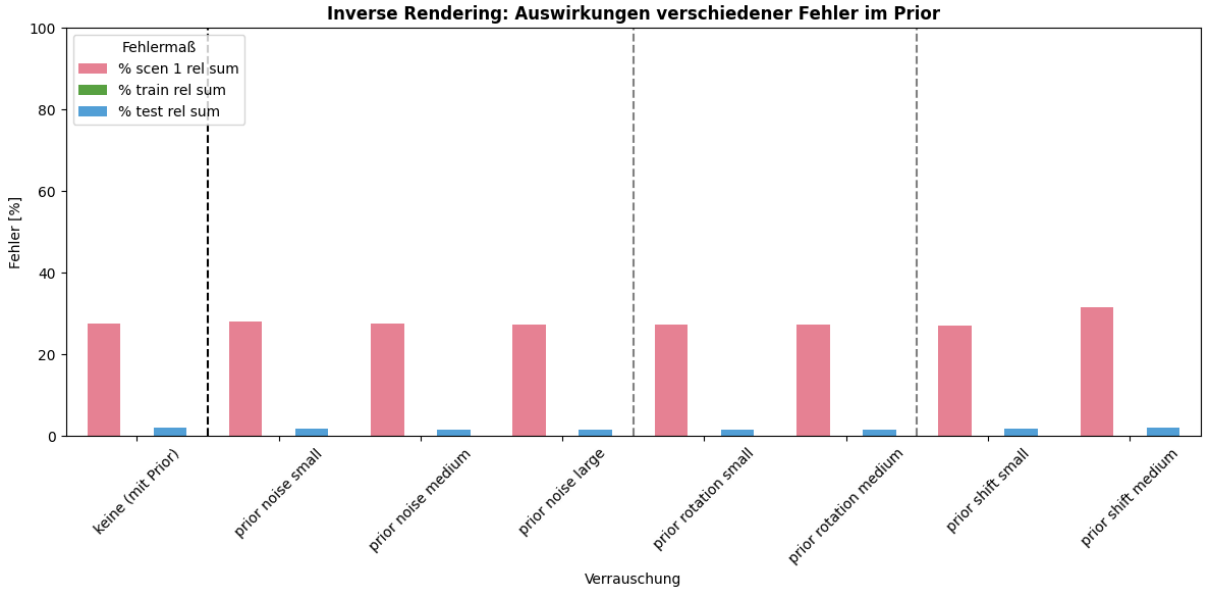

Abbildung 5.7: Auswirkungen verschiedener Arten und Abstufungen von Fehlern im Prior (Prior noise, Prior rotation, Prior shift) auf die Fehlermaße der Rekonstruktionen im Vergleich zur Rekonstruktion mit fehlerfreien Daten mit ***Inverse Rendering***

Bei der Analyse der Auswirkungen verschiedener Arten von Fehlern im Prior zeigt sich, dass diese die Rekonstruktionsqualität im Vergleich zum unverrauschten Prior nicht negativ beeinflussen. Es wird die gleiche Ergebnisqualität (mit minimalen Schwankungen) in Bezug auf den prozentualen Szenariofehler und den prozentualen Trainings- und Testpfadfehler erreicht. Lediglich eine deutliche Verschiebung des Priors (Prior shift medium) hat zu einer merklichen Verschlechterung des Rekonstruktions-Ergebnisses geführt, dessen Szenariofehler aber dennoch weiterhin niedriger ist, als im Experiment ohne Prior. Daraus lässt sich schlussfolgern, dass eine Initialisierung der Methode *Inverse Rendering* mit einem Prior im Vergleich zu einer Initialisierung mit Nullwerten immer sinnvoll ist, selbst wenn Zweifel an der Qualität des Priors bestehen und dieser gegebenenfalls Fehler enthält.

6. Diskussion

Das in Kapitel 1 beschriebene Problem einer verbesserten Rekonstruktion der Wolken freigesetzter radioaktiver Stoffe durch die Nutzung von individuellen Bewegungsprofilen und den dazugehörigen Expositionswerten lässt sich mit den beschriebenen Methoden grundsätzlich lösen. In Evaluierungen wurde deutlich, dass das Verhältnis zwischen gewünschter Raumzeit-Auflösung und Pfaden stimmen muss. Für eine feingranulare Auflösung (100x100 Zellen räumliche Auflösung, 100 Zeitschritte) im Katastrophenfall sind mehr Pfade notwendig, als realistisch verfügbar sein werden. Kleinere Auflösungen lassen sich hingegen gut lösen. Eine gute Orientierung für die Rekonstruktionsgüte bilden die durchschnittlichen Überlappungen von Pfaden pro Raumzeitpunkt. Je höher diese Zahl ist, desto besser funktioniert die Rekonstruktion.

Es wurde die Anwendbarkeit von verschiedenen Methoden evaluiert. Besonders tauglich haben sich Methoden aus dem Bereich Novel View Synthesis (Mildenhall, et al., 2020; Kerbl, et al., 2023) erwiesen. Diese haben im Vergleich zu klassischen statistischen Optimierungen und Regressionen den Vorteil, dass sie Daten in Teilmengen (sog. Batches) in Verbindung mit einem Gradientenverfahren verarbeiten können und so besser auf höhere Auflösungen und Pfadanzahlen skalieren. Im Detail wurden die Verfahren Cuboid Splatting und Inverse Rendering weiter evaluiert. Insgesamt erweist sich die Methode Cuboid Splatting, insbesondere in geringeren Auflösungen als die deutlich schnellere Rekonstruktionsmethode gegenüber Inverse Rendering, die dennoch mindestens zu vergleichbaren, oder häufig sogar zu deutlich besseren Ergebnissen bezüglich der Qualität der Rekonstruktionen führt. Insbesondere sieht man bessere Generalisierungen der Beobachtungen (auch unter Fehlereinfluss). Dies passt gut zu aktuellen Ergebnissen aus der Wissenschaft im Bereich Computer Vision, aus dem sowohl die Methode Inverse Rendering (Spezielle Form von Neural Radiance Fields), als auch die dem Cuboid Splatting zugrundeliegende Originalmethode Gaussian Splatting stammen. Dort, im Anwendungsfall der 3D-Bildrekonstruktionen und Novel View Synthesis, zeichnet sich ein ähnliches Bild ab, dass das Gaussian Splatting als Weiterentwicklung nach Neural Radiance Fields schnellere und qualitativ oft sogar hochwertigere Ergebnisse erzielen kann (Kerbl, et al., 2023).

In den detaillierten Evaluationen mit variierenden Parametern konnten weitere Zusammenhänge aufgezeigt werden. Realistische Szenarien wie das Emsland-Szenario zeichnen sich durch komplexe Formen mit stark variierenden Belastungswerten aus und sind schwerer zu rekonstruieren als einfachere Formen wie das Gaußverteilte oder das Box-Szenario. Die Verwendung von Warnzonen hat in den vorliegenden Experimenten eher zur Verbesserung der Rekonstruktion beigetragen, weil Personen vermutlich länger zuhause sind und sie somit länger im Gebiet bleiben und längere Pfade entstehen. Das Straßennetz beeinflusst zwar die Qualität der Vorhersagen in den verschiedenen Experimenten, allerdings nicht auf einheitliche Weise über die Experimente hinweg. Selbst ein relativ schlechter Prior hat für die Methode Inverse Rendering für bessere Ergebnisse gesorgt. Wie bessere Prioren helfen, konnte nicht umfassend evaluiert werden.

Fehler und Verrauschungen in den Evidenzdaten wirken sich wie erwartet negativ auf die Rekonstruktionsgüte aus. Kleine unsystemische Fehler können von den Modellen im Zuge der Generalisierung größtenteils mitigiert werden. Allerdings haben insbesondere Fehler in den Bewegungsprofilen eine starke Wirkung auf das Ergebnis. In den Szenarien mit kleiner Auflösung, in denen die Auswirkung der Verrrauschung evaluiert wurde, sind kleine Pfadabweichungen gleich sehr große Abweichungen im Raum (über mehrere Kilometer), was selbst die kleinste von uns getestete Verrauschungsstufe sehr pessimistisch macht. Insgesamt muss aber nochmal der

Vorteil von technischen Lösungen für die Vermeidung von Pfadfehlern betont werden, hier kommt vor allem die Nutzung von aufgezeichneten Bewegungsdaten in Mobiltelefonen mit freiwilliger Zustimmung der Personen in Betracht. Methodisch stellt sich Cuboid Splatting auch deutlich robuster in der Generalisierung von Beobachtungen mit Fehlern heraus als Inverse Rendering.

Das experimentelle Setup unterliegt einigen Limitationen. Zwei Limitationen, die mit den synthetischen Bewegungsdaten zu tun haben, sind, dass zum einen Personen teilweise lange in einem Explorationsmodus bleiben und das Gebiet erst spät verlassen. Eine weitere Limitation entstand durch Hardware- und Zeit-Begrenzungen. Die Leistungsbeschreibung des Projektes verlangte, dass die Modelle auf handelsüblichen lokalen Geräten und nicht bloß auf Hochleistungs-Rechenclustern laufen. Aufgrund der Vielzahl der Experimente mussten Begrenzungen bei besonders lang laufenden Durchläufen gesetzt werden, sodass wir nicht alle Parameterkombinationen mit großen Pfadmengen evaluieren konnten. Darüber hinaus empfehlen wir den Einsatz einer GPU, auch wenn diese für die Zielhardware als optional gegeben war.

Aus unserer Sicht gibt es viele spannende Folgefragestellungen zu der aktuellen Studie. Besonders hervorzuheben wäre aus unserer Sicht das Einbeziehen von Prioren für das Cuboid Splatting und die Ausbreitung der Fragestellung auf multimodale Datenquellen und räumliche Interpolation. Im Kern ginge es darum, dass man zum Beispiel reale Messwerte, Wetterausbreitungsmodelle und die Pfade mit Expositionsmessungen bei Personen alle gleichermaßen einem Modell zur Verfügung stellt.

Literaturverzeichnis

Abadi et al., 2015. *TensorFlow: Large-scale machine learning on heterogeneous systems.,* Software available from tensorflow.org.

Agrawal, R. & de Alfaro, L., 2019. *Learning Edge Properties in Graphs from Path Aggregations*. New York, NY, USA, Association for Computing Machinery, p. 15–25.

Barron et al., 2023. *Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields*. arXiv.

Barron, J. T. et al., 2021. *Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields.*, pp. 5855-5864.

Barron, J. T. et al., 2022. *Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields.*, pp. 5470-5479.

Bartz-Beielstein et al., 2014. *Evolutionary algorithms*. Wiley Interdisciplinary Reviews: Data Mining, Issue 4.3, p. 178–195.

Biel, M. & Johansson, M., 2022. *Efficient Stochastic Programming in Julia*. INFORMS Journal on Computing, July, Band 34, p. 1885–1902.

Bitkom, 2019. *Desktop PCs produktneutral ausschreiben. Leitfaden für den öffentlichen IT-Einkauf. Stand: September 2019*.

Boyd, S., Boyd, S. P. & Vandenberghe, L., 2004. *Convex optimization*. Cambridge university press.

Bui-Thanh et al., 2013. *A Computational Framework for Infinite-Dimensional Bayesian Inverse Problems Part I: The Linearized Case, with Application to Global Seismic Inversion*. SIAM Journal on Scientific Computing, Issue 35.6, eprint: doi.org/10.1137/12089586X, p. A2494–A2523.

Bui-Thanh, T., 2012. *A Gentle Tutorial on Statistical Inversion using the Bayesian Paradigm*.

Cer, D. et al., 2018. *Universal Sentence Encoder*. arXiv.

Chen et al., 2020. *Robust stochastic optimization made easy with RSOME*. Management Science, Issue 66.8, p. 3329–3339.

Chen, Z. & Xiong, P., 2021. *RSOME in Python: an open-source package for robust stochastic optimization made easy*. Optimization Online.

Cohen, M. B., Lee, Y. T. & Song, Z., 2018. *Solving Linear Programs in the Current Matrix Multiplication Time*. arXiv.

Conneau, A. et al., 2018. *What you can cram into a single vector: Probing sentence embeddings for linguistic properties*. arXiv.

Dantzig, G. B., 1982. *Reminiscences about the origins of linear programming*. Operations Research Letters, Band 1, pp. 43-48.

Dashti, M. & Stuart, A. M., 2013. *The Bayesian Approach To Inverse Problems*. arXiv.

Del Debbio et al. , 2022. *Bayesian approach to inverse problems: an application to NNPDF closure testing*. The European Physical Journal C, p. 330.

Dellaert, F. & Yen-Chen, L., 2021. *Neural Volume Rendering: NeRF And Beyond*. arXiv.

Dey, A., Ahmine, Y. & Comport, A. I., 2022. *Mip-NeRF RGB-D: Depth Assisted Fast Neural Radiance Fields*. Journal of WSCG, Band 30, p. 34–43.

Eliasof, M., Haber, E. & Treister, E., 2022. *pathGCN: Learning General Graph Spatial Operators from Paths*. arXiv e-prints, July.p. arXiv:2207.07408.

Fernández-González, D. & Gómez-Rodríguez, C., 2020. *Transition-based Semantic Dependency Parsing with Pointer Networks*. arXiv.

Fridovich-Keil, S. et al., 2022. *Plenoxels: Radiance Fields Without Neural Networks.*, pp. 5501-5510.

Gao et al., 2022. *NeRF: Neural Radiance Field in 3D Vision, A Comprehensive Review*. arXiv.

Gardener et al., 2018. *GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration*. Advances in neural information processing systems, Issue 31.

Goodfellow, I. J. et al., 2014. *Generative Adversarial Networks*. arXiv.

Görtler, J., Kehlbeck, R. & Deussen, O., 2019. *A Visual Exploration of Gaussian Processes*. Distill.

Hochreiter, S. & Schmidhuber, J., 1997. *Long short-term memory*. Neural computation, Band 9, p. 1735–1780.

Iyyer, M., Manjunatha, V., Boyd-Graber, J. & Daumé III, H., 2015. *Deep unordered composition rivals syntactic methods for text classification.*, p. 1681–1691.

Jidling, C., Wahlström, N., Wills, A. & Schön, T. B., 2017. *Linearly constrained Gaussian processes*. s.l., Curran Associates, Inc..

Kekkonen, H., 2019. *Bayesian Inverse Problems*.

Kerbl, B., Kopanas , G., Leimkühler, T. & Drettakis, G., 2023. *3D Gaussian Splatting for Real-Time Radiance Field Rendering*. ACM Trans. Graph. 42, 4 Article 139, doi.org/10.1145/3592433, p. 14.

Liu, L. et al., 2020. *Neural Sparse Voxel Fields.*, Curran Associates, Inc., p. 15651–15663.

Li, Z., Niklaus, S., Snavely, N. & Wang, O., 2021. *Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes.*, pp. 6498-6508.

Loshchilov, I. & Hutter, F., 2019. *Decoupled Weight Decay Regularization.*, ICLR 2019.

Luby, M. & Nisan, N., 1993. *A Parallel Approximation Algorithm for Positive Linear Programming*. New York, NY, USA, Association for Computing Machinery, p. 448–457.

Michalak, A. & Kitanidis, P., 2003. *A method for enforcing parameter nonnegativity in Bayesian inverse problems with an application to contaminant source identification*. Water Resources Research, Issue 39.2 eprint: https : / / agupubs . onlinelibrary . wiley . com / doi / pdf / 10 . 1029 /.

Mildenhall, B. et al., 2020. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. arXiv.

Mirza, M. & Osindero, S., 2014. *Conditional Generative Adversarial Nets*. arXiv.

Müller, T., Evans, A., Schied, C. & Keller, A., 2022. I*nstant neural graphics primitives with a multiresolution hash encoding*. ACM Transactions on Graphics, July, Band 41, p. 1–15.

Nicolicioiu, A., Duta, I. & Leordeanu, M., 2019. *Recurrent Space-time Graph Neural Networks.*, Curran Associates, Inc..

Oechsle, M., Peng, S. & Geiger, A., 2021. *UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction.*, pp. 5589-5599.

Park, K. et al., 2021. *Nerfies: Deformable Neural Radiance Fields.*, pp. 5865-5874.

Park, K. et al., 2021. *HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields.* arXiv.

Paszke et al., 2019. *PyTorch: An Imperative Style, High-Performance Deep Learning Library.* Advances in Neural Information Processing Systems *32,* p. 8024–8035.

Peng, S. et al., 2020. *Convolutional occupancy networks.*, p. 523–540.

Prekopa, A., 1973. *Contributions to the theory of stochastic programming.* Mathematical Programming, Band 4, p. 202–221.

Raissi, M. & Karniadakis, G., 2016. *Deep Multi-fidelity Gaussian Processes.* arXiv.

Ramirez, D., Jayasuriya, S. & Spanias, A., 2022. *Towards Live 3D Reconstruction from Wearable Video: An Evaluation of V-SLAM, NeRF, and Videogrammetry Techniques.* arXiv.

Ravasi, M. & Vasconcelos, I., 2020. *PyLops—A linear-operator Python library for scalable algebra and optimization*. SoftwareX *,* Issue 11, p. 100361.

Rückert, D. et al., 2022. *NeAT: Neural Adaptive Tomography.* arXiv.

Shapiro, A., 1993. *Asymptotic Behavior of Optimal Solutions in Stochastic Programming.* Mathematics of Operations Research, Band 18, pp. 829-845.

Shapiro, A., 2019. *Stochastic Programming.*

Socher, R. et al., 2013. *Recursive deep models for semantic compositionality over a sentiment treebank.*, p. 1631–1642.

Sun, C., Sun, M. & Chen, H.-T., 2022. *Improved direct voxel grid optimization for radiance fields reconstruction.* arXiv preprint arXiv:2206.05085.

Su, R.-Q., Wang, W.-X., Wang, X. & Lai, Y.-C., 2016. *Data-based reconstruction of complex geospatial networks, nodal positioning and detection of hidden nodes.* Royal Society Open Science, Band 3, p. 150577.

Tancik et al., 2021. *Learned Initializations for Optimizing Coordinate-Based Neural Representations.* arXiv.

Tancik, M. et al., 2020. *Fourier features let networks learn high frequency functions in low dimensional domains.* Advances in Neural Information Processing Systems, Band 33, p. 7537–7547.

Tarvainen et al., 2013. *Bayesian Image Reconstruction in Quantitative Photoacoustic Tomography".* IEEE Transactions on Medical Imaging, Issue 32.12, p. 2287–2298.

Tauman Kalai, Y., Raz, R. & Regev, O., 2016. *On the Space Complexity of Linear Programming with Preprocessing.* New York, NY, USA, Association for Computing Machinery, p. 293–300.

Vaidya, P. M., 1989. *Speeding-up linear programming using fast matrix multiplication.*, pp. 332-337.

Vanderbei et al., 2020. *Linear programming*. Springer.

Virtanen et al., 2020. *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*. Nature Methods, Issue 17, pp. 261-272.

Wang R. & Tao D., 2014. *Recent progress in image deblurring*. arXiv preprint arXiv:1409.6838.

Wiecki et al., 2023. *pymc-devs/pymc: v5.2.0. Version v5.2.0*.

Würfl, T., Ghesu, F. C., Christlein, V. & Maier, A., 2016. *Deep learning computed tomography.*, p. 432–440.

Wu, Y., Zhuang, D., Labbe, A. & Sun, L., 2020. *Inductive Graph Neural Networks for Spatiotemporal Kriging*. arXiv.

Xian, W., Huang, J.-B., Kopf, J. & Kim, C., 2021. *Space-Time Neural Irradiance Fields for Free-Viewpoint Video.*, pp. 9421-9431.

Zhu, J.-Y., Park, T., Isola, P. & Efros, A. A., 2017. *Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks*.