Diese Arbeit wurde vorgelegt am Institut für Flugsystemdynamik

von

Benedikt Wiethof

Matrikel-Nr.: 335806

Master-Thesis

Integration von Inverse Depth Parametrisiertem UKF-SLAM im Kontext von Multi-Sensor Data Fusion in unbemannten Luftfahrzeugen

Aachen, 25. November 2020

Wissenschaftliche Leitung:	Univ.-Prof. Dr.-Ing. Professor Dieter Moormann
Betreuender Mitarbeiter:	Tobias Islam, M.Sc. (RWTH)

1. Prüfer:	Univ.-Prof. Dr.-Ing. Professor Dieter Moormann
2. Prüfer:	Dr.-Ing. Norbert Siepenkötter

RWTH Aachen University
Institut für Flugsystemdynamik

Zusammenfassung

In dieser Arbeit wird eine Möglichkeit der optischen Navigation von unbemannten Luftfahrzeugen unter der Verwendung von Simulataneous Localization And Mapping mit einem Square-Root UKF. Dazu wurde zur Repräsentation einzelner Objekte die Inverse Depth Parametrisierung verwendet.
Die Realisierung erfolgt durch eine geeignete Initialisierung neuer Objekte sowie einer effizienten Verwaltung der Karte. Dafür wird die Karte in eine kurzfristige und eine langfristige Karte unterteilt. Die kurzfristige Karte verwaltet die lokale Umgebung, während die langfristige Karte eine Karte der zurückgelegten Umgebung repräsentiert. Die Initialisierung unterliegt zudem einer Vorauswahl der Objekte, um nur eine repräsentative Menge der lokalen Umgebung aufzunehmen
Abschließend wird gezeigt, dass die entwickelte Methodik geeignet ist, eine korrekte Navigation nur über optische Sensordaten zu ermöglichen. Zudem wird gezeigt, dass die Verwendung eines weiteren Sensors die Navigation verbessert und stabilisiert.

Abstract

This thesis will show a possibility of optical navigation of unmanned aerial vehicles using Simulataneous Localization And Mapping with a Square-Root UKF. For this purpose the inverse depth parameterisation was used to represent individual objects.
The realisation is achieved by a suitable initialisation of new objects and an efficient map management. Therefore the map is divided into a short-term and a long-term map. The short-term map manages the local environment, while the long-term map represents a map of the covered environment. The initialisation is also subject to a pre-selection of the objects in order to include only a representative quantity of the local environment
Finally, it is shown that the developed methodology is suitable to enable a correct navigation only via optical sensor data. It is also shown that the use of an additional sensor improves and stabilises the navigation.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

Abkürzungen

DP Distributed Particle 7

ECEF Earth Centered Earth Fixed Koordinatensystem 12

EKF Extended Kalmanfilter 14

MSER Maximally stable extremal regions 5

NED North East Down Koordinatensystem 12

PTU Pan-Tilt-Unit 2

SIFT Scale-invariant feature transform 5

SLAM Simultaneous Localization And Mapping 6

SURF Speeded up robust features 5

UAV Unmanned Aerial Vehicle 1

UKF Unscented Kalmanfilter 4

UT Unscented Transform 15

VO Visuelle Odometrie 1

Indizes

$\mathcal{B}$ Körperfestes Koordinatensystem 12

$\mathcal{C}$ Kamerafestes Koordinatensystem 12

$\mathcal{PTU}_B$ Befestigung der Pan-Tilt-Unit 13

$\mathcal{PTU}_C$ Befestigungspunkt der Kamera an der Pan-Tilt-Unit 13

$\mathcal{W}$ Weltkoordinatensystem 12

C Objekte in karteischer Repräsentation 26

I Objekte in Inverse Depth Parametrisierung 26

a Erweiterter Zustand 17

c Bezüglich der Kovarianzmatrix 16

m Bezüglich des Mittelwerts 16

Symbole

$\delta\mathbf{r}_{\mathfrak{W}}$ Positionsfehler 24

$\mathcal{X}$ Sigmapunkt 16

$\mathcal{Y}$ Transformierter Sigmapunkt 16

$\delta\theta_{\mathfrak{W}}^{\mathfrak{W}}$ Lagefehler 24

$\delta\mathbf{q}_{\mathfrak{W}}^{\mathfrak{W}}$ Lagefehler als Quaternion 25

$\delta\mathbf{v}_{\mathfrak{W}}$ Geschwindigkeitsfehler 24

$\delta\mathbf{x}$ Fehlerzustandsvektor 24

$\overline{\mathbf{y}}$ Prädizierter Messwert 16

1 Einleitung

Die Verwendung von unbemannten Luftfahrzeugen (eng. Unmanned Areal Vehicle UAV) steigt stetig in verschiedenen Bereichen. So können UAVs zu Dokumentationszwecken im Bereich der Forschung, der Begutachtung von Gelände oder der Medien verwendet werden. Weiterhin steigt der Einsatz zu logistischen Zwecken oder auch bei Rettungseinsätzen. Bei diesen können UAVs ein Bild der Lage übermitteln, bevor Fachkräfte am Einsatzort eintreffen.

Da UAVs in Regionen vordringen können, die von Menschen nicht ohne Probleme erreichbar oder zu gefährlich sind, werden Systeme zur Selbstlokalisierung benötigt. Dabei werden vor allem globale Navigationsssatellitensysteme, wie z.B. GPS, GLONASS oder Galileo verwendet. Um robuster gegen Störungen dieser Systeme zu sein, werden alternative Positionierungssysteme benötigt. Diese Störungen können einerseits durch Sensorausfälle auftreten, aber auch durch eine zu geringe Zahl an Satelliten. Letzteres kann durch Störsignale oder physische Barrieren wie z.B. Wände oder Dächer entstehen.

Als alternative Positionierungssysteme können kann auf Trägheitsnavigation basierende Systeme zurückgreifen werden, die mit Hilfe von Beschleunigungs- und Gierratensensoren sowie Magnetometer auf Basis der Bewegung die aktuelle Position schätzen. Zudem kann mit Radarsensorik die Entfernung zu definierten Landmarken oder Objekten gemessen werden und durch die Relationen die eigene Position bestimmen.

Neben diesen Verfahren existieren optische Navigationsverfahren, die mit Hilfe von Kamerabildern die eigene Postion bestimmen und auf Fokus dieser Arbeit gesetzt ist. Basierend auf den Kamerabildern kann die Eigenbewegung mittels Visueller Odometrie (VO) geschätzt werden. Dabei wird mithilfe der aufgenommenen Kamerabilder die Eigenbewegung geschätzt.

Die Auswertung von Kamerabildern zur Selbstlokalisierung wird im Bereich der Robotik bereits in vielen Bereichen verwendet. Allerdings ist die Übertragung auf ein UAV nicht ohne weiteres möglich, da das aufgenommene Bild, aufgrund der Bewegung oberhalb der Szene, die Umgebung als planare Ebene wahrnimmt. Dadurch enthält eine Aufnahme weniger Informationen als am Boden.

Obwohl die Kamerabilder keine Tiefeninformationen abbilden können, besteht die Möglichkeit unter Verwendung von detektierten Objekten die eigene Orientierung zu schätzen. Einerseits können ortsfeste Landmarken verwendet werden, dessen exakte Position in einem Weltkoordinatensystem bereits bekannt ist. Dieses Koordinatensystem ist häufig das geographische oder geozentrische Koordinatensystem.

Andererseits besteht die Möglichkeit Objekte der lokalen Umgebung zu detektieren und im Kontext von Simultaneous Localization and Mapping (SLAM) zu speichern sowie die eigene Positionierung zu verbessern. Die Verbesserung entsteht, indem das UAV eine bessere Kenntnis der Umgebung enthält. Aus der Verwendung optischer Sensordaten für SLAM entstand die Inverse Depth Parametrisierung. Diese erlaubt detektierte Objekte direkt zu verwenden. Insbesondere lassen sich Objekte in großer Distanz früh zur Schätzung Orientierung nutzen. Das bietet im Bereich der Luftfahrt den Vorteil, Objekte, die sich am Horizont befinden, früh für eine korrekte Orientierung zu verwenden.
Ein Luftfahrzeug besitzt sechs Freiheitsgrade, sodass diese Objekte aus dem Kamerabild verschwinden können. Um diese Objekte weiterhin im Blickfeld zu behalten, befindet sich am UAV eine Pan-Tilt-Unit (PTU). Diese kann die Orientierung der Kamera entsprechend anpassen, damit Objekte länger im Bild zu detektieren sind. Diese Anpassungen des Blickfeldes können dann mit der Inverse Depth Parametrisierung zur Orientierung und Positionsschätzung eingesetzt werden.

Die Inverse Depth Parametrisierung wird in einen Unscented Kalmanfilter (UKF) implementiert. Dieser bildet eine Erweiterung des Kalmanfilters für nichtlineare Zustandsübergänge und bietet die Möglichkeit der einfachen Erweiterung um neu detektierter Objekte. Zunächst erfolgt eine Einführung in die verwendeten Koordinatensysteme, so wie die entsprechenden Transformationen. Diese Transformationen sind zudem durch die Verwendung der PTU relevant.
Anschließend erfolgt eine Einführung in die Kalmanfilterung sowie der entsprechenden Erweiterungen. Die Erweiterung des UKF zur Verwendung in SLAM wird darauf folgend im Kontext der Optischen Navigation eingeführt. Vor diesem Hintergrund wird zudem die Inverse Depth Parametrisierung dargestellt sowie die Verwendung der entsprechend notwendigen Kameramodelle. Anschließend werden die bestehenden Rahmenbedingungen dargelegt.
Im folgenden Abschnitt wird zunächst eine Anpassungen der Parametrisierung vorgenommen, um Winkelungenauigkeiten und -unstetigkeiten zu umgehen. Anschließend wird eine Initialisierung mehrere Objekte dargelegt und die Notwendigkeit der Verwendung des Square Root UKF begründet. Nachfolgend wird zunächst das verwendete Sensormodell beschrieben unter Betrachtung der Transformationen durch PTU und Lage des UAV.
Abschließend wird eine Kartenverwaltung beschrieben, die eine Echtzeitfähigkeit ermöglichen soll. Das geschieht einerseits durch die Transformationen der Objekte in eine kartesische Repräsentation und andererseits durch eine Verwaltung einer kurzfristigen und einer langfristigen Karte. Diese erzeugen effektiv ein Bild der lokalen und der globalen Umgebung.
Die Modellierung wird anschließend in einer Simulationsumgebung mittels Simulink® validiert. Dabei erfolgt zunächst jeweils die isolierte Betrachtung der Parametrisierung im Kontext von UKF SLAM und die der Kartenverwaltung. Zum Schluss wird die Kombination erprobt inklusive Ausfall von Sensorik.

2 Stand der Technik

Simultaneous Localization And Mapping (SLAM) kann mit verschiedenen Methodiken erfolgen. Grundlegend sind dabei häufig die in Abschnitt 2.1 beschriebenen Zustandsschätzer. Darauf aufbauend sind die verschiedenen Algorithmen in 2.2.2 entstanden. Hinzu kommen ausschließlich auf SLAM ausgerichtete Methodiken.
Die Inverse Depth Parametrisierung ist Teil der optischen Navigation und ermöglicht die direkte Verwendung neuer Objekte im Kontext von SLAM. Neben dieser existieren weitere Möglichkeiten, beschrieben in 2.2

2.1 Zustandsschätzer

Ein grundlegender Teil der optischen Navigation ist die Schätzung des eigenen Zustands. Die bekanntesten Schätzer sind Kalmanfilter und Partikelfilter. Beide bilden einen rekursiven Filter, indem sie zur Schätzung des Zustands im nächsten Zeitschritt den a priori geschätzten verwenden.
Ein Zustand setzt sich im Bereich von UAVs in der Regel aus Position, Geschwindigkeit und räumlicher Lage zusammen. Mit Kenntnis über diese Zustandsvariablen ist eine Navigation möglich.

2.1.1 Kalmanfilter

Der Kalmanfilter nach Kalman [25] ist eine weit verbreitete Möglichkeit, den Zustand zu schätzen und bildet im Fall eines linearen Zustandsübergangs die optimale Schätzung. Diese Zustandsschätzung wird durch einen Mittelwert $\mathbf{x}$ und einer zugehörigen Kovarianzmatrix $\mathbf{P}$ repräsentiert. Der Mittelwert gibt die aktuelle Schätzung an, während die Kovarianzmatrix die Genauigkeit dieser Schätzung durch Angabe der Abweichung wiederspiegelt.
Die Filterung ist in zwei Schritte gegliedert. Zum einen die Prädiktion, die den Zustand anhand eines Bewegungsmodelles initial schätzt. Zum anderen die Korrektur, die den geschätzten Zustand mit Sensordaten verbessert.

Da der Kalmanfilter ausschließlich für lineare Zustandsübergänge geeignet ist, existieren diverse Erweiterungen, um eine Schätzung für nicht lineare Zustandsübergänge zu ermöglichen. Weit verbreitet sind Extended Kalmanfilter (EKF) und Unscented Kalmanfilter (UKF). Der EKF nutzt zur Abbildung des nichtlinearen Zustandsübergang eine linearisierte Form des Modells, indem eine Taylorreihenentwicklung nach dem linearen Glied abgebrochen wird. [16, 20, 31, 45]
Der UKF nutzt einen reduzierten Samplingansatz der Monte-Carlo-Methoden, um die Wahrscheinlichkeitsverteilung des Zustands mit diskreten Punkten darzustellen [22]. Diese werden anschließend durch die nicht lineare Funktion propagiert und wieder zu einer Wahrscheinlichkeitsverteilung aus $\mathbf{x}$ und $\mathbf{P}$ zusammen gesetzt. Das ermöglicht die Erfassung Momente höherer Ordnung [51].

Die Rechenkomplexität von EKF und UKF ist in der Ordnung von $O(n^2)$ [15]. Der UKF besitzt dabei Vorteile gegenüber dem EKF [16, 23, 52]. Zum einen werden durch die Linearisierung Terme höherer Ordnung ignoriert, die größere Fehler induzieren und zur Divergenz führen können [51, 52]. Weiterhin ist die Jacobimatrix zu bestimmen, was viele partielle Ableitungen benötigt. Das wiederum ist ein fehleranfälliger Prozess, da diese im Vorhinein händisch bestimmt werden müssen. Weiterhin zeigt H. Khazraj, F. Faria da Silva und C. L. Bak [16], dass der UKF auch hinsichtlich der Rechenzeit Vorteile mit sich bringt.

2.1.2 Partikel Filter

Partikel Filter sind nicht-parametrische rekursive Bayes-Filter. Nicht-parametrische Filter benötigen keine Funktion für die posterior, sondern approximieren diese mit einer endlichen Anzahl an Parametern [45, S. 187ff].
Der Bayes Filter ist in Algorithmus 1 dargestellt. Dabei wird auf Basis des vorherigen Zustands unter Einbezug der Dynamik der aktuelle Zustand postuliert.

Algorithm 1 Bayes Filter Algorithmus[45, S. 23]

procedure ParticleFilter($bel(x_{t-1})$, u_t, z_t)
  Initialize $\overline{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
  for $m = 1$ **to** M **do**
    $\overline{bel}(x_t) = \int p(x_t|u_t,x_{t-1})bel(x_{t-1})\mathrm{d}x$
    $bel(x) \ = \eta \ p(z_t|x_t)\overline{bel}(x)$
  end for
  return $bel(x_t)$
end procedure

Der Partikel Filter ist in Algorithmus 2 dargestellt. Dieser bildet eine diskrete Lösung des Bayes-Filters, indem die Wahrscheinlichkeitsverteilung zum Zeitpunkt t durch M sogenannte Partikel repräsentiert wird. Jedes Partikel enthält eine Zustandshypothese sowie ein

Gewicht entsprechend der aktuellen Wahrscheinlichkeitsverteilung (Zeile 4). Im nächsten Schritt werden die Sensordaten verwendet, um die Gewichte jedes Partikels anzupassen, sodass sie der realen Verteilung entsprechen (Zeile 5). Anschließend findet das sogenannte "Importance sampling" statt. In diesem Schritt werden aus dem aktualisierten Partikelset Partikel mit einer Wahrscheinlichkeit proportional zu ihrem Gewicht gezogen und dem Ergebnisset hinzugefügt (Zeile 8ff). Infolgedessen sind unwahrscheinlichere Zustände entfernt und damit die Konvergenzrate erhöht worden [45, S. 77ff].

Algorithm 2 Partikel Filter Algorithmus [45, S. 78]

1: **procedure** ParticleFilter($bel(x_{t-1})$, u_t, z_t)
2: Initialize $\overline{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: **for** $m = 1$ **to** M **do**
4: sample $x_t^{[m]} \sim p(x_t|u_t,x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t|x_t^{[m]})$
6: $\overline{\mathcal{X}}_t = \overline{\mathcal{X}}_t + \left\langle x_t^{[m]},w_t^{[m]} \right\rangle$
7: **end for**
8: **for** $m = 1$ **to** M **do**
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: **end for**
12: **return** $bel(x_t)$
13: **end procedure**

2.2 Optische Navigation

Optische Navigation beschreibt die Schätzung des Zustands under Verwendung von optischen Sensoren. Das können beispielweise LIDar-Sensoren oder Kameras sein.
Kameras nehmen eine Projektion der Umgebung auf, sodass Kameramodelle benötigt werden, die eine Übertragung in den dreidimensionalem Raum ermöglichen. Dementsprechend beschränkt sich das Anwendungsgebiet der optischen Navigation häufig auf die Lageschätzung [36, 47].
Der Aufbau einer Karte wird mit Hilfe von SLAM Algorithmen realisiert, wie beschrieben in 2.2.2. Diese ergänzen und verbessern die in 2.2.1 beschriebene visuelle Odometrie. Das heißt die Schätzung der Eigenbewegung mit Hilfe der Kamerabilder.

Weiterhin ist die Detektion und Wiedererkennung von Objekten im Kontext von SLAM relevant, um neue Objekte aufzunehmen oder erneut für die Lokalisation zu verwenden. In der Regel wird eins der drei Verfahren Scale-invariant feature transform (SIFT)[28], Speeded up robust features (SURF)[3] oder Maximally stable extremal regions (MSER)[33] verwendet.

2.2.1 Visuelle Odometrie

Visuelle Odometrie (VO) bezeichnet die Schätzung der Eigenbewegung auf Basis von Kamerabildern. Dabei wird eine Sequenz von Kamerabildern ausgewertet, um die relative Bewegung von einem Bild zum nächsten zu schätzen. Diese können dann zu einer Gesamtbewegung aufsummiert werden. Dadurch unterliegt die Schätzung einer Akkumulation der Fehler über die Zeit [12].
Es kann zwischen Stereo und Monocular unterschieden werden. Bei der Stereo VO werden zwei Kameras verwendet, um daraus 3D Informationen zu rekonstruieren. Dabei werden Bildobjekte direkt aus den beiden Kamerabildern rekonstruiert. Monocular VO verwendet nur eine Kamera und rekonstruiert die 3D Informationen aus zwei oder mehreren zeitlich aufeinanderfolgenden Kamerabildern. Dabei unterliegt die Rekonstruktion zusätzlich der Ungenauigkeit hinsichtlich der Kamerabewegung.
Die VO kann unter Verwendung von inertialen Messdaten verbessert werden [26].
Die Verbesserungen (VIO) nutzen verschiedene Ansätze zur Schätzung. Häufig werden Nichtlineare Versionen des Kalmanfilters verwendet, die sich in Messmodell und Tiefenkopplung der optischen Messungen unterscheiden. Bekannte Vertreter sind Algorithmen MSCKF [43], ROVIO [4] und SVO+MSF [5, 29].
Weiterhin werden auch Smoother und Sliding Window Esitmator verwendet [5, 27, 30, 39].

2.2.2 Simultaneous Lokalisation and Mapping

Simultaneous Localization And Mapping (SLAM) ist ein Verfahren aus der Robotik, bei der ein Roboter seine räumliche Position schätzt und gleichzeitig eine Karte seiner Umgebung anfertigt. Die Karte wird dabei durch Objekte repräsentiert. Die Schätzung von Position und Umgebung findet online ohne vorherige Kenntnis des Ortes statt [10].
SLAM wird häufig über probabilistic SLAM formuliert [44]. Dabei wird zu jedem Zeitpunkt k eine Wahrscheinlichkeitsverteilung in Gl. 2.1 über die eigene Position $\mathbf{x}$ und die der Objekte $\mathbf{m}$ geschätzt [14, 24, 46, 49].

$$P(\mathbf{x}_k,\mathbf{m}|\mathbf{Z}_{0:k},\mathbf{U}_{0:k},\mathbf{x}_0) \tag{2.1}$$

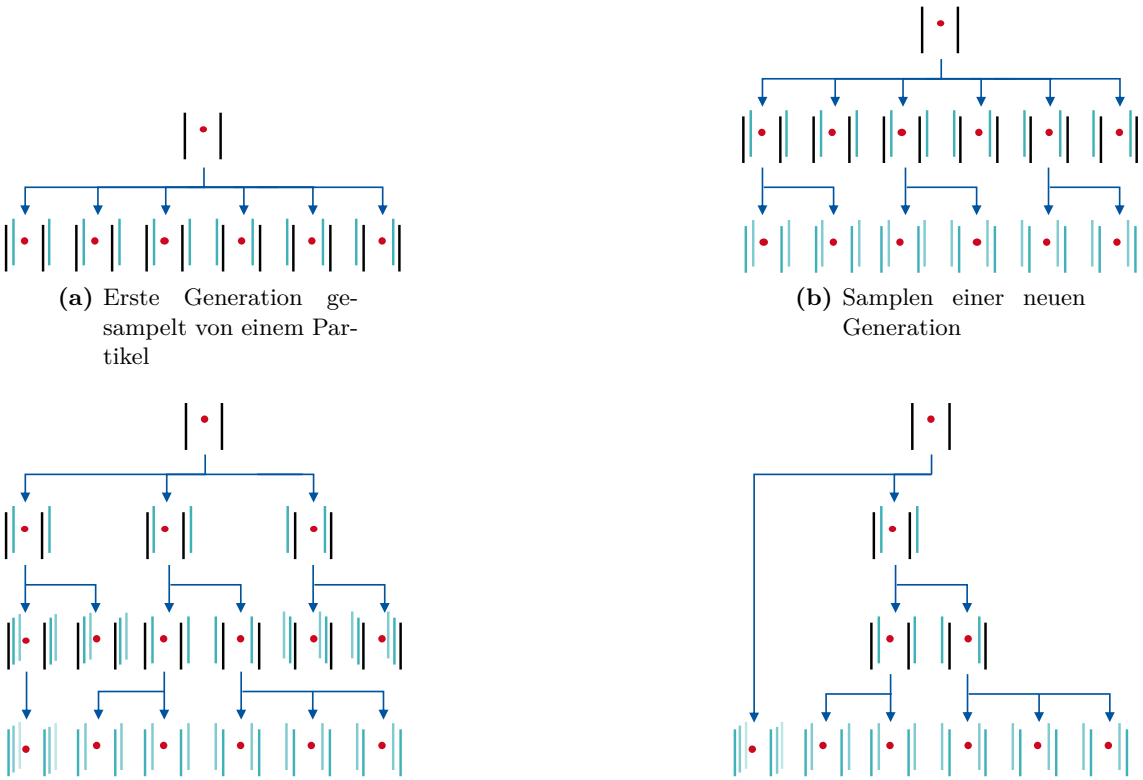
Wobei $\mathbf{Z}_{0:k}$ und $\mathbf{U}_{0:k}$ die Beobachtungen bzw. Bewegungen vom initialen 0 Zustand bis zum aktuellen Zustand beschreiben.
Zur Lösung des Problems wurden zum einen die Zustandsschätzer aus 2.1 um das Mapping erweitert sowie Graph-basierte Verfahren entwickelt.

SLAM mit Partikelfiltern

Partikelfilter aus 2.1.2 sind zur Zustandsschätzung konzipiert und zur Verwendung im Kontext von SLAM um den Mapping Aspekt zu erweitern. Dementsprechend existieren verschiedene Erweiterungen wie FastSLAM nach Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit [35] und Distributed Particle(DP)-SLAM nach Austin I. Eliazar und Ronald Parr [2], die den Aspekt unterschiedlich behandeln.
FastSLAM erweitert jeden Partikel um die beobachteten Objekte. Diese werden jeweils durch einen 2×2 Extended KalmanFilter dargestellt sowie einem Gewicht.
In jedem Schritt werden zusätzlich Beobachtungen für jedes Objekt im Partikel prädiziert und der EKF ausgeführt. Dabei korrigiert der EKF das Gewicht.
Im FastSLAM 2.0 von Michael Montemerlo, Sebastian Thrun, Daphne Koller und Ben Wegbreit [34] werden die Beobachtungen bereits beim Sampling der Partikel miteinbezogen. Dadurch kann die Partikelanzahl reduziert werden und der Algorithmus ist robuster mit dem Nachteil einer höheren Komplexität.

Im Gegensatz zum FastSLAM erweitert DP-SLAM den Partikelfilter in jedem Partikel um eine sogenannte GridMap. Diese stellt die Karte als Raster dar. Die effiziente Implementierung wird über die Verwendung eines Stammbaums für jeden Partikel sowie die Verwaltung einer gemeinsamen statt mehrerer Karten gewährleistet. Dadurch wird verhindert, dass jedes Partikel eine eigene mögliche Trajektorie mit der entsprechenden Umgebung verwaltet. Dafür wird ein Partikel des Zeitschritts i, das zum samplen ein oder mehrerer Partikel im Zeitschritt $i + 1$ genutzt wird, zu deren Elternpartikel. Die Partikel des Zeitschritts $i + 1$ bilden damit die Kindpartikel. Partikel mit dem gleichen Elternpartikel sind Geschwister und stellen jeweils eine mögliche Pose des Roboters basierend auf der Pose des Elternpartikels dar. Die Kinder erhalten eine Referenz zu ihrem Elternpartikel und verwalten eine Liste ihrer aktualisierten Rasterquadrate. Dadurch wird in jedem Zeitschritt der Stammbaum erweitert.
Die Größe des Stammbaums ist aus speichereffizienten Gründen begrenzt. Das wird einerseits dadurch gewährleistet, dass Partikel, die kein Kinder des aktuellen Zeitschritts haben, und damit keinen Beitrag mehr leisten, aus dem Stammbaum entfernt werden. Elternpartikel dieses Partikels, die dadurch keine Kinder mehr besitzen, werden dadurch ebenfalls entfernt. Andererseits werden Elternpartikel mit ihren Kinderpartikeln zusammengefasst, wenn die Elternpartikel nur ein Kind besitzen. Das Zusammenfassen sorgt dafür, dass alle Beobachtungen des Elternpartikels in das Kindpartikel übertragen werden. Zudem wird die Id des Elternpartikels übertragen. Diese beiden Schritte werden in jedem Zeitschritt für jeden Baum durchgeführt.
Die GridMap speichert die Geschichte der Änderungen durch die Partikel, indem ein balancierter Baum mit den entsprechenden Partikel Ids generiert wird. Dementsprechend wird die GridMap als Matrix aus leeren Bäumen initialisiert. Dadurch wird nicht mehr

jedem Partikel eine Karte, sondern den entsprechenden Kartenteilen Partikel zugeordnet. Die Abb. 2.1 illustriert den DPSLAM Algorithmus, beginnend bei der ersten Generation in 2.1a ist in Abb. 2.1b zu erkennen, dass nicht alle Partikel Kinderpartikel erzeugen. In Abb. 2.1c sind die Partikel ohne Kinder entfernt und eine neue Generation erzeugt worden. Das anschließende Entfernen ist in Abb. 2.1d illustriert. Dort ist der linke Partikel mit seinen Elternknoten zusammengefasst worden, da diese keine weiteren Kinder mehr besitzen.



(a) Erste Generation gesampelt von einem Partikel

(b) Samplen einer neuen Generation

(c) Erzeugung weiterer Generationen nach entfernen von Partikeln

(d) Reduzieren des Baumes

Abbildung 2.1: Illustration von DP-SLAM [13]

Graph-basierte Methoden

Beim Graph-basiertem SLAM wird das SLAM Problem unter Verwendung eines Graphen gelöst, der kontinuierlich aufgebaut wird. Bekannte Vertreter sind GraphSLAM [46] und TreeMap [14].

Im GraphSLAM entspricht jeder Knoten einer Pose während des Mappingprozesses und jede Kante zwischen zwei Knoten repräsentiert eine räumliche Beschränkung hinsichtlich der Bewegung. Hinzu kommen Beschränkungen durch Beobachtungen von Objekten, wobei diese als zeitinvariante Knoten zu interpretieren sind. Dadurch wird kontinuierlich ein

Graph aufgebaut, wie er in Abb. 2.2 illustriert ist.

Den Kanten wird jeweils ein Gewicht für die Bewegung und den Beobachtungen zu-

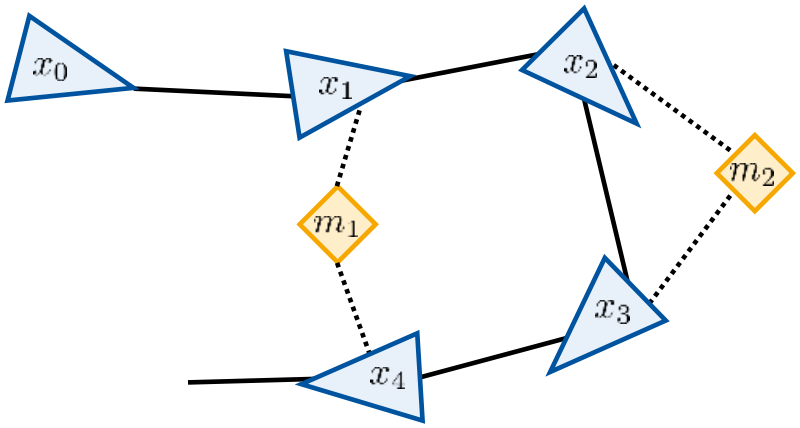


Abbildung 2.2: Graph beim GraphSLAM [46]

gewiesen unter Verwendung eines Bewegungsmodells g und Beobachtungsmodells h. Das Beobachtungsmodell inkludiert dabei die Objekte. Ziel beim GraphSLAM ist, einen Graph zu erzeugen mit einer entsprechenden Knotenkonfiguration, die den Einschränkungen am besten gerecht wird, sodass sich ein Minimierungsproblem ergibt. Das Minimierungsproblem wird dann für jede Pose mit Hilfe des Gauss-Newton oder Levenberg-Marquardt Verfahren gelöst.

Der TreeMap-Algorithmus nach Frese [14] beruht auf der Überlegung, dass ein Roboter in einem Gebäude befindet, dass virtuell in zwei Abschnitte A und B geteilt wurde. Der Roboter befindet sich in Abschnitt A und benötigt dann nur die Informationen aus Abschnitt B, die er von der aktuellen Position in B beobachten kann. Alle weiteren Objekte, die sich in B befinden, sind zu dem Zeitpunkt nicht von Interesse. Dementsprechend ist nur die Marginalverteilung der Objekte, die aus beiden Abschnitten zu beobachten sind, im aktuellen Zustand von Interesse. Die Marginalverteilung ist zusätzlich durch die Beobachtungen im anderen Abschnitt bedingt.

Mit diesem Ansatz wird beim TreeMap-Algorithmus die Karte rekursiv in einen binären Baum aus Unterabschnitten geteilt. Infolgedessen ergibt sich der Graph nach Abb. 2.3. Hier stehen die schwarzen Linien für den Teil des Algorithmus, der das Mapping beschreibt und die blauen Linien für den Teil zur Zustandsschätzung.

Das Mapping startet an den Blättern des Baumes, die die Messwerte als gausschen Eingang speichern. An jedem Blatt werden alle Objekte, die außerhalb des Blattes nicht mehr verwendet werden, durch Marginalisierung entfernt und in einem bedingenden Blatt gespeichert (blauer Kasten). Die restlichen Informationen werden an den Elternknoten weitergegeben. An diesem werden die Informationen mit denen des anderen Kindes kombiniert und die Marginalisierung wird erneut durchgeführt. Dieser Algorithmus setzt sich fort bis zum Wurzelknoten.

Zur Zustandsschätzung wird der Mittelwert der marginalisierten Objekte an der Wurzel

verwendet und an die Kinder weitergegeben, an denen es mit den gespeicherten Bedingungen kombiniert wird. Das Ergebnis wird anschließend weitergegeben und erneut kombiniert bis zum Ende des Baumes.

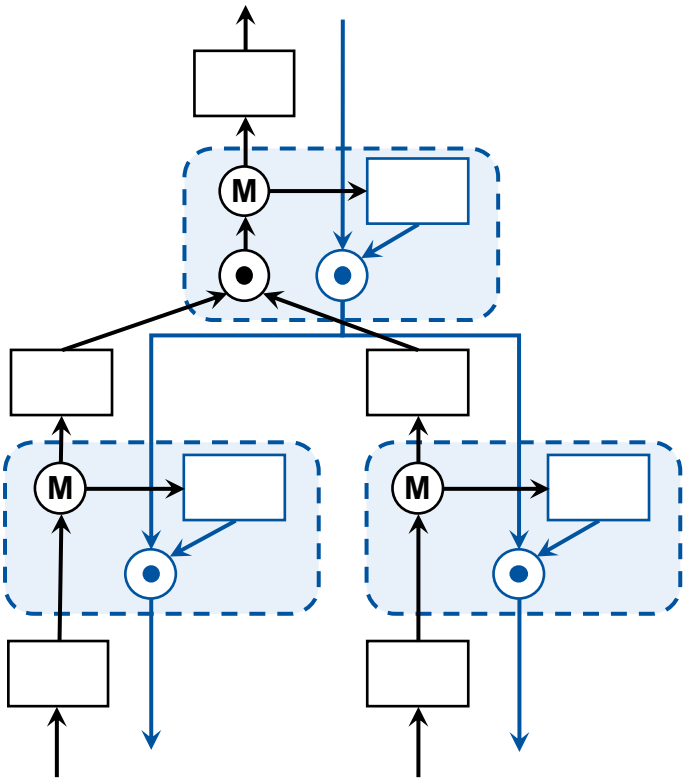


Abbildung 2.3: Aufbau des Baumes vom TreeMap Algorithmus[49]

SLAM mit Kalmanfiltern

Die Erweiterung des Kalmanfilters im SLAM-Kontext unterscheidet sich nicht für KF, EKF und UKF. In allen Fällen wird der Zustandsvektor $\mathbf{x}$ und die Kovarianzmatrix $\mathbf{P}$ um die entsprechenden Objekte erweitert[15, 42]. Dementsprechend beschreiben die Untermatrizen auf der Hauptdiagonalen die Kovarianzmatrizen des ursprünglichen Zustands und die der Objekte. Auf den Nebendiagonalen befinden sich sowohl die Kreuzvarianzen der Objekte untereinander als auch die mit dem ursprünglichen Zustand. Die Kreuzvarianzen beschreiben entsprechenden die Korrelationen.
Die Objekte können, sofern bekannt, im Vorhinein festgelegt sein oder im Laufe der Bewegung erfasst und integriert werden. Um ein fehlerhaftes Lokalisieren zu vermeiden, ist eine im Vorhinein bekanntes Objekt mit einer hohen Ungenauigkeit zu initialisieren [45, S.253].
Wenn Objekte während der Bewegung erfasst und in die Zustandsbeschreibung aufgenommen werden, wachsen Zustandsvektor und Kovarianzmatrix kontinuierlich.
Neben der Erweiterung der Zustandsbeschreibung sind auch das System- und das Mess-

rauschen sowie die Modelle anzupassen. Im Bewegungsmodell ist zu garantieren, dass eine ortsfestes Objekt keine Translation erfährt und lediglich durch das Systemrauschen beeinflusst wird. Im Beobachtungsmodell ist eine Prädiktion der Messung von Objekten in Abhängigkeiten der aktuellen Position vorzusehen, um die Karte im Korrekturschritt verbessern zu können.

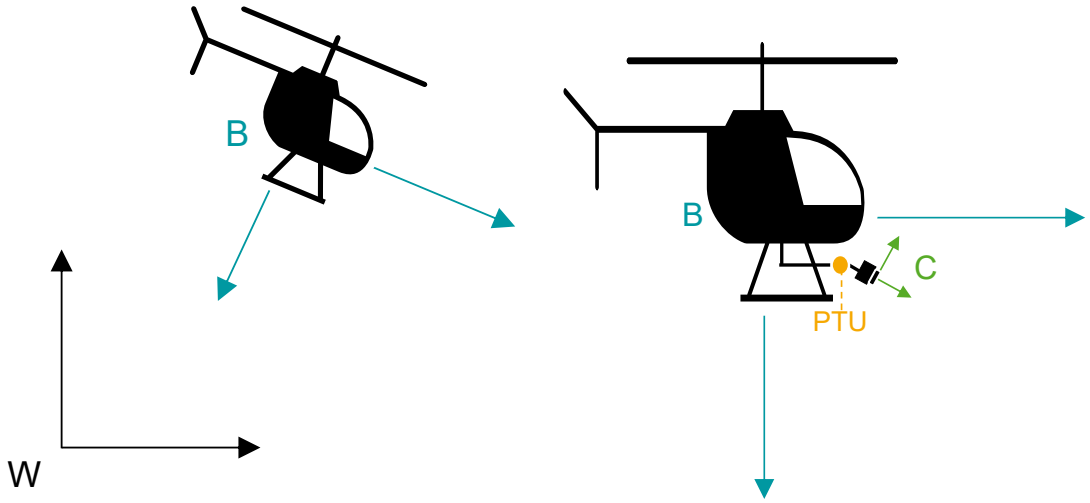
3 Grundlagen

Im Zuge dieser Arbeit wird der Unscented Kalmanfilter als Zustandsschätzer im Kontext von SLAM verwendet. In 3.2 wird zunächst der grundlegende Kalmanfilter erklärt und darauf aufbauend der UKF eingeführt Im Abschnitt 3.3 wird die Verwendung des Kalmanfilters im Kontext von SLAM sowie die verwendeten Kameramodelle beschrieben. Insbesondere der Gebrauch von Objekten im Filterprozessschritt wird dargelegt. Objekte sind durch die Sensorik detektierte Teile der Umgebung. Diese befinden sich zum Zeitpunkt der Detektion in unmittelbarer Umgebung. Mit Hilfe dieser Objekte kann dann eine Karte der Umgebung erzeugt werden. Zur Beschreibung der Objekte innerhalb der Karte existiert die Inverse Depth Parametrisierung.
Abschließend wird in Abschnitt 3.4 das bestehende Framework sowie dessen Besonderheiten im Hinblick auf die Parametrisierung dargelegt.
Die Ergebnisse dieser Arbeit sind zur Verwendung eines UAV gedacht. Dementsprechend werden verschiedene Koordinatensysteme und Transformationen benötigt, um Bewegungs- und Sensormodell richtig darzustellen. Diese werden nachfolgend vorgestellt.

3.1 Koordinatensysteme und -transformationen

Durch die Verwendung einer Kamera ergeben sich drei Koordinatensysteme mit entsprechender Translation und Rotation. Zum einen ein Weltkoordinatensystem. In diesem wird die globale Position des Flugobjekts angegeben. In der Regel wird hierfür North East Down (NED) oder Earth Centered Earth Fixed (ECEF) verwendet. Dieses Koordinatensystem wird folgend mit $\mathcal{W}$ indiziert. Hinzukommt ein körperfestes Koordinatensystem, indiziert mit $\mathcal{B}$, welches den Ursprung im Schwerpunkt hat. Ausgehend von diesem sind alle Translationen und Rotationen zur entsprechenden Sensorik angegeben. Das körperfeste Koordinatensystem ist ein rechtsdrehendes, wobei die x-Achse nach vorne und die z-Achse nach unten orientiert ist. Das dritte Koordinatensystem ist ein kamerafestes Koordinatensystem. Die Fläche, aufgespannt durch die x- und y-Achse, liegt im Kamerabild und die z-Achse zeigt aus der Kameraebene heraus. Beschreibungen in diesem Koordinatensystem werden mit $\mathcal{C}$ indiziert. Für die Inverse Depth Parametrisierung ist die Kenntnis über die Transformationen zwischen den Koordinatensystemen erforderlich. Dabei ist zusätzlich die Besonderheit einzugehen, dass die Kamera an einer Pan-Tilt-Unit befestigt ist. Dadurch muss bei der Transformation in das kamerafeste Koordinatensystem die zusätzliche Trans-

lation und Rotation miteinbezogen werden.
Zwischen den Koordinatensystemen ist entsprechend zu transformieren. Dabei ist zu beachten, dass die PTU eine zusätzliche Rotation und Translation einbringt. Dadurch ergibt sich eine Transformation von der Befestigung der PTU $\mathcal{PTU}_B$ am UAV zur Befestigung der Kamera an der PTU $\mathcal{PTU}_C$.
Abb. 3.1 zeigt die entsprechenden Koordinatensysteme. Dabei ist in Abb. 3.1a die Relation des Weltkoordinatensystem zum körperfesten Koordinatensystem dargestellt. Hierbei ist entsprechend die Möglichkeit der unterschiedlichen Lage zu beachten. In Abb. 3.1b ist nur das verwendete UAV inklusive der PTU und der Kamera illustriert. Durch die Verwendung der PTU kann das aufgenommene Kamerabild entsprechend beeinflusst werden.



(a) Welt- und körperfestes Koordinatensystem in Relation

(b) Körperfestes und sensorfestes Koordinatensystem in Relation

Abbildung 3.1: Verwendete Koordinatensysteme

3.2 Kalmanfilter

Der Kalmanfilter ist wie in 2.1.1 beschrieben eine verbreitete Möglichkeit zur Zustandsschätzung, die den Zustand durch einen Mittelwert $\mathbf{x}$ und eine Kovarianzmatrix $\mathbf{P}$ beschreibt. Die Schätzung ist dabei in einen Prädiktions- und einen Korrekturschritt gegliedert. Letztere verbessert den durch ein Bewegungsmodell geschätzten Zustand unter Verwendung von Sensordaten.
Die Berechnung der prädizierten Schätzung erfolgt unter Verwendung des Mittelwerts und der Kovarianzmatrix des letzten bekannten Zustands sowie des Signals der Steuergröße. (3.1, 3.2).

$$\widehat{\mathbf{x}}_k = \mathbf{A}_d \cdot \mathbf{x}_{k-1} + \mathbf{B}_d \cdot \mathbf{u}_k \tag{3.1}$$

$$\widehat{\mathbf{P}}_k = \mathbf{A}_d \mathbf{P}_{k-1} \mathbf{A}_d^T + \mathbf{Q}_k \tag{3.2}$$

Das Bewegungsmodell in 3.1 und 3.2 wird beschrieben durch die Matrizen $\mathbf{A}_d$ und $\mathbf{B}_d$. Letztere bildet den Einfluss der Steuergröße $\mathbf{u}_k$ auf den neuen Zustand ab, während $\mathbf{A}_d$ den Einfluss des letzten Zustands beschreibt.

Da das Modell die Realität nicht vollständig abbildet, wird zusätzlich ein Systemrauschen $\mathbf{Q}$, das den Modellfehler beschreibt, in die Prädiktion eingebracht. Wenn die einzelnen Einträge des Zustandsvektors linear unabhängig sind, kann $\mathbf{Q}$ als Varianz dargestellt werden. Damit reduziert sich das Systemrauschen auf eine Diagonalmatrix [32, S. 142-143].

Der anschließende Korrekturschritt benötigt zur Verarbeitung der Sensordaten ein Sensormodell. Dieses bildet den Zustandsvektor auf den Bereich der Sensordaten ab und ermöglicht damit die Postulierung der Messdaten.
Da auch die Sensordaten keine vollständig perfekte Abbildung der Umgebung liefern können, ist analog zum Bewegungsmodell ein sensorbedingtes Rauschen $\mathbf{R}$ zu beachteten. In der Regel bildet das Rauschen den minimalen hardwarebedingten Fehler ab [32, S. 13].
Die Fehler durch System- und Messrauschen werden durch die sogenannte Kalmanverstärkung gegeneinander abgewägt. Dieser gibt an, wie stark die Korrektur durch die Sensordaten in Abhängigkeit des Messrauschens stattfindet. Je geringer das Messrauschen, desto ausgeprägter ist der Einfluss der Sensordaten.

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

$$\mathbf{x}_k = \widehat{\mathbf{x}}_k + \mathbf{K}_k \cdot (\mathbf{z}_k - \mathbf{C}_k \widehat{\mathbf{x}}_k) \tag{3.4}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \widehat{\mathbf{P}}_k \tag{3.5}$$

Mit Hilfe der Kalmanverstärkung erfolgt die Korrektur der in der Prädiktion geschätzten Werte unter Verwendung von 3.4 und 3.5. Der korrigierte Mittelwert $\mathbf{x}$ und die Kovarianzmatrix $\mathbf{P}$ werden anschließend im nächsten Filterprozessschritt als Eingangsparameter verwendet.

Der Kalmanfilter ist weit verbreitet, da die Zustandsschätzung immer konvergiert, solange das Bewegungs- und Sensormodell nicht zu stark verrauscht sind. Die Problematik des Kalmanfilters ist die Begrenzung auf lineare Zustandsübergänge. In der Realität sind diese in den meisten Anwendungen allerdings nichtlinear. Dementsprechend existieren Erweiterungen, die die Approximation mit nichtlinearen Übergängen ermöglichen. Weit verbreitet sind der sogenannte Extended Kalmanfilter (EKF) oder Unscented Kalmanfilter (UKF). Der UKF weißt dabei Vorteile gegenüber dem EKF auf. Zum einen bildet er bei gleicher

Komplexität eine genauere Schätzung. Zum anderen benötigt der EKF Jacobi-Matrizen, deren händische Bestimmung stark fehleranfällig ist. [6, 50–53]

3.2.1 Unscented Kalmanfilter

Der Unscented Kalmanfilter (UKF) bildet eine Alternative zur Verwendung des Kalmanfilters mit nichtlinearen Funktionen nach Julier, Uhlmann und Durrant-Whyte [22]. Die Motivation ist, eine bessere Schätzung zu ermöglichen, ohne den Rechenaufwand zu erhöhen, indem der Sampling-Ansatz der Monte-Carlo-Methoden verwendet wird. Diese ziehen eine große Anzahl zufälliger Zustände aus der aktuellen Schätzung, um die Verteilung möglichst genau abzubilden [38].

Grundlage des UKF bildet die Unscented Transform (UT) nach Julier, Uhlmann und Durrant-Whyte [23], die als Eingangsparameter den Mittelwert sowie die zugehörige Kovarianzmatrix benötigen.
Die Parameter werden zur Erzeugung sogenannter Sigmapunkte verwendet, die die statistische Verteilung des aktuellen Zustands beschreiben. Im Gegensatz zu den Monte-Carlo-Methoden ist die Menge der zu erzeugenden Punkte deutlich geringer, um die Rechenkomplexität zu minimieren. Gleichzeitig ist die Anzahl so zu wählen, dass der Zustand inklusiver aller Ungenauigkeiten ausreichend repräsentiert wird. Dementsprechend wurde die Anzahl auf $2L + 1$ gesetzt, wobei L die Größe des Zustandsvektors beschreibt. Die Sigmapunkte werden symmetrisch um den Mittelwert verteilt 3.6[53]. Zusätzlich bildet der Mittelwert selbst einen Sigmapunkt. Mit Hilfe von λ, α und β kann die Verteilung um $\mathbf{x}$ beeinflusst werden [51].
Um die Sigmapunkte wieder zu einem Mittelwert und einer Kovarianz zusammenzufügen, werden Gewichte nach 3.7 definiert [51]. Dabei sind die Gewichte gekennzeichnet mit (m) zur Berechnung des Mittelwerts und mit (c) zur Berechnung der Kovarianz.

$$
\begin{aligned}
\mathcal{X}_0(k|k) &= \mathbf{x}_i(k|k) \\
\mathcal{X}_i(k|k) &= \mathbf{x}_i(k|k) + (\sqrt{(L+\lambda)\mathbf{P}(k|k)})_i \quad i = 0,\ldots,L \\
\mathcal{X}_{i+L}(k|k) &= \mathbf{x}_i(k|k) - (\sqrt{(L+\lambda)\mathbf{P}(k|k)})_{i-L_i} \quad i = L,\ldots,2L
\end{aligned} \tag{3.6}
$$

$$
\begin{aligned}
W_0^{(m)} &= \frac{\lambda}{L+\lambda} \\
W_0^{(c)} &= \frac{\lambda}{L+\lambda} + (1 - \alpha^2 + \beta) \\
W_i^{(c)} &= W_i^{(m)} = \frac{1}{2(L+\lambda)} \qquad i = 1,\ldots,2L \\
&mit\ \lambda = \alpha^2(L+\kappa) - L
\end{aligned} \tag{3.7}
$$

UT Nach der Erzeugung der Sigmapunkte erfolgte die Propagierung durch eine beliebige Funktion, wobei diese auf jeden Sigmapunkt einzeln angewendet wird, sodass die Anzahl der Sigmapunkte gleich bleibt. Die Funktion kann sowohl linear als auch nicht linear sein. (3.8).

$$\mathcal{Y}_i = g(\mathcal{X}_i), \quad i = 0, \ldots, 2L \tag{3.8}$$

Anschließend werden der gewichtete Mittelwert und die zugehörige Kovarianz nach 3.9 und 3.10 berechnet.

$$\overline{\mathbf{y}} = \sum_{n=0}^{2L} W_i^{(m)} \mathcal{Y}_i \tag{3.9}$$

$$\mathbf{P}_y = \sum_{n=0}^{2L} W_i^{(c)} \{\mathcal{Y}_i - \overline{\mathbf{y}}\}\{\mathcal{Y}_i - \overline{\mathbf{y}}\}^T \tag{3.10}$$

Die UT wird im UKF sowohl für das Bewegungs- als auch das Beobachtungsmodell verwendet. Algorithmus 3 zeigt einen Zeitschritts des UKF, bestehend aus Prädiktion (Zeile 5 - 12) und Korrektur (Zeile 12 - 17).

Algorithm 3 Unscented Kalman Filter

1: **procedure** UKF($\mathbf{x}$,$\mathbf{P}$)
2: Calculate sigma points
3: $\mathcal{X}_{k-1} = [\widehat{x}_{k-1} \; \widehat{x}_{k-1} \pm \sqrt{(L+\lambda)\mathbf{P}_{k-1}}]$
4:
5: Prediction
6: $\mathcal{X}_{k|k-1} = \mathbf{F}(\mathcal{X}_{k-1}) + \mathbf{Q}_k$
7: $\widehat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}_{i,k|k-1}$
8: $\mathbf{P}_k^- = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1} - \widehat{\mathbf{x}}_k^-][\mathcal{X}_{i,k|k-1} - \widehat{\mathbf{x}}_k^-]^T$
9: $\mathcal{Y}_{k|k-1} = \mathbf{H}(\mathcal{X}_{k|k-1}) + \mathbf{R}_k$
10: $\widehat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,k|k-1}$
11:
12: Measurement Update
13: $\mathbf{P}_{\mathbf{y}_k\mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{Y}_{i,k|k-1} - \widehat{\mathbf{y}}_k^-][\mathcal{Y}_{i,k|k-1} - \widehat{\mathbf{y}}_k^-]^T$
14: $\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1} - \widehat{\mathbf{x}}_k^-][\mathcal{Y}_{i,k|k-1} - \widehat{\mathbf{y}}_k^-]^T$
15: $\mathcal{K}_k = \mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} \mathbf{P}_{\mathbf{y}_k\mathbf{y}_k}^{-1}$
16: $\widehat{\mathbf{x}}_k = \widehat{\mathbf{x}}_k^- + \mathcal{K}_k(\mathbf{y}_k - \widehat{\mathbf{y}}_k^-)$
17: $\mathbf{P}_k = \mathbf{P}_k^- - \mathcal{K}_k \mathbf{P}_{\mathbf{y}_k\mathbf{y}_k} \mathcal{K}^T$
18: **end procedure**

In der Regel sind Modell- und Sensorrauschen nicht additiv, sodass die Schätzung mittels der Augmented UT erfolgen muss [52, 53]. Diese verwendet angepasste Übergangsfunktionen, indem Zustandsvektor und Kovarianzmatrix um Mess- und Systemrauschen erweitert werden. Das hat einer Erhöhung der Dimension des Zustandsvektors zur Folge und führt zu einer größeren Menge Sigmapunkte. Dementsprechend sind die Funktionen

für Bewegungs- und Beobachtungsmodell nach 3.11 und 3.12 anzupassen, sodass System- und Messrauschen innerhalb der Funktion bereits addiert werden.

$$f^a(\mathbf{x}^a) = f^a(\begin{bmatrix}\mathbf{x}^T & \mathbf{w}^T\end{bmatrix}^T) \tag{3.11}$$
$$= f(\mathbf{x}) + \mathbf{w}$$
$$h^a(\mathbf{x}^a) = h^a(\begin{bmatrix}\mathbf{x}^T & \mathbf{v}^T\end{bmatrix}^T) \tag{3.12}$$
$$= h(\mathbf{x}) + \mathbf{v}$$

Durch Verwendung der Augmented UT im UKF erfolgt die Erweiterung nach 3.13, wobei das Rauschen von Bewegungs- und Beobachtungsmodell mit einem Erwartungswert von 0 angenommen wird. Dementsprechend wird der Zustandsvektor um Nullen der Größe des System- und Messrauschens erweitert. Die erweiterte Kovarianzmatrix ist eine Blockdiagonalmatrix, bestehend aus der ursprünglichen Kovarianzmatrix sowie des System- und Messrauschens mit $\mathbf{Q} = E[\mathbf{w}\mathbf{w}^T]$ und $\mathbf{R} = E[\mathbf{v}\mathbf{v}^T]$.

$$x^a = [x^T \ 0 \ 0]^T \tag{3.13}$$
$$P^a = \begin{bmatrix} \mathbf{P} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{bmatrix}$$

In Folge der Erweiterung von System- und Messrauschen ist das Bewegungsmodell nach 3.14 anzupassen, sodass das Messrauschen unbeeinflusst bleibt. Das Systemrauschen wird dabei entsprechend für jeden Sigmapunkt addiert.

$$f^a(\mathbf{x}^a) = f^a(\begin{bmatrix}\mathbf{x}^T & \mathbf{w}^T & \mathbf{v}^T\end{bmatrix}^T) \tag{3.14}$$
$$= \begin{bmatrix} f(\mathbf{x}) + \mathbf{w} \\ \mathbf{v} \end{bmatrix}$$

3.2.2 Square Root Unscented Kalmanfilter

Eine Erweiterung des UKF bildet der Square Root UKF nach VAN DER MERWE und WAN [50], dargestellt in Algorithmus 4. Dieser reduziert die Komplexität des UKF und erhöht die

numerische Stabilität, indem die Wurzel der Kovarianzmatrix $\sqrt{\mathbf{P}} = \mathbf{S}$ direkt aktualisiert wird. Dafür werden die drei folgenden mathematischen Operationen verwendet:

1. die QR Zerlegung,
2. die Cholesky Faktor Aktualisierung
3. die Methode der kleinsten Quadrate

Durch das direkte Aktualisieren von $\mathbf{S}$ kann diese beim Ziehen der Sigmapunkte verwendet werden. Anschließend können die Sigmapunkte wie bisher durch die Modellfunktion propagiert und zum neuen Mittelwert zusammengesetzt werden. Die Aktualisierung von $\mathbf{S}$ erfolgt in Zeile 8 und 9. Dabei gibt *qr* die obere rechte Dreiecksmatrix $\mathbf{R}$ der QR-Zerlegung $\mathbf{A}^T = \mathbf{QR}$ zurück. Der obere Dreieckteil der Matrix ist dabei genau die Transponierte des Cholesky Faktors von $\mathbf{P} = \mathbf{SS}^T$. Das *cholupdate* beschreibt eine Änderung von Rang 1, wobei für das Ergebnis von $\mathbf{S} = chol\{\mathbf{S},\mathbf{u},\pm\nu\}$ gelten muss $\mathbf{SS}^T = \mathbf{P} \pm \sqrt{\nu}\mathbf{uu}^T$. Falls $\mathbf{u}$ eine Matrix ist, wird die Rang 1 Aktualisierung für jede Spalte von $\mathbf{u}$ durchgeführt. Alle weiteren Berechnungen der Prädiktion werden wie bisher durchgeführt.

Für den Korrekturschritt wird bei der Berechnung von $\mathbf{S}_{\widehat{\mathbf{y}}_k}$ analog vorgegangen (Zeile 15 und 16). Die Kreuzvarianzmatrix $\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k}$ wird wie bisher bestimmt.
Die Kalmanverstärkung $\mathcal{K}$ wird in Zeile 18 mit Hilfe der Methode der kleinsten Quadrate bestimmt, gekennzeichnet durch den Operator /. In dieser Arbeit wird als Lösungsansatz eine QR-Zerlegung mit voller Pivotisierung verwendet .
Anschließend sind der korrigierte Mittelwert $\widehat{\mathbf{x}}$ und die korrigierte Matrixwurzel $\mathbf{S}$ zu bestimmen. Letztere wird in Zeile 20 und 21 entsprechend mit einem *choleskyupdate* berechnet.

Algorithm 4 Square-Root Algorithmus nach VAN DER MERWE und WAN [50]

procedure $Square - RootUKF(\mathbf{x}, \mathbf{S})$

Calculate Sigma points

$\mathcal{X}^*_{k-1} = [\widehat{x}_{k-1}\ \widehat{x}_{k-1} \pm \sqrt{\gamma}\mathbf{S}_{k-1}]$

Prediction

$\mathcal{X}^*_{k|k-1} = \mathbf{F}(\mathcal{X}_{k-1})$

$\widehat{\mathbf{x}}^-_k = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}^*_{i,k|k-1}$

$\mathbf{S}^-_k = qr\left\{\left[\sqrt{W_i^{(c)}}(\mathcal{X}^*_{1:2L,k|k-1} - \widehat{\mathbf{x}}^-_k)\ \sqrt{Q}\right]\right\}$

$\mathbf{S}^-_k = cholupdate(\mathbf{S}^-_k,\ \mathcal{X}^*_{0,k|k-1} - \widehat{\mathbf{x}}^-_k,\ W_0^{(c)})$

$\mathcal{X}_{k|k-1} = [\widehat{x}_{k-1}\ \widehat{x}_{k-1} \pm \sqrt{\gamma}\mathbf{S}^-_k]$

$\mathcal{Y}_{k|k-1} = \mathbf{H}(\mathcal{X}^x_{k|k-1})$

$\widehat{\mathbf{y}}^-_k = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,k|k-1}$

Measurement Update

$\mathbf{S}_{\widehat{\mathbf{y}}_k} = qr\left\{\left[\sqrt{W_i^{(c)}}(\mathcal{Y}_{1:2L,k|k-1} - \widehat{\mathbf{y}}^-_k)\ \sqrt{R}\right]\right\}$

$\mathbf{S}_{\widehat{\mathbf{y}}_k} = cholupdate(\mathbf{S}_{\widehat{\mathbf{y}}_k},\ \mathcal{Y}_{0,k|k-1} - \widehat{\mathbf{y}}^-_k,\ W_0^{(c)})$

$\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1} - \widehat{\mathbf{x}}^-_k][\mathcal{Y}_{i,k|k-1} - \widehat{\mathbf{y}}^-_k]^T$

$\mathcal{K}_k = (\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} / \mathbf{S}^T_{\widehat{\mathbf{y}}_k}) / \mathbf{S}_{\widehat{\mathbf{y}}_k}$

$\widehat{\mathbf{x}}_k = \widehat{\mathbf{x}}^-_k + \mathcal{K}_k(\mathbf{y}_k - \widehat{\mathbf{y}}^-_k)$

$\mathbf{U} = \mathcal{K}\mathbf{S}_{\widehat{\mathbf{y}}_k}$

$\mathbf{S}_k = cholupdate(\mathbf{S}^-_k,\ \mathbf{U},\ -1)$

end procedure

3.3 Optische Navigation

Optische Navigation verwendet Kamerabilder zur Schätzung des Zustands sowie der Umgebung. Diese nehmen eine Projektion der Umgebung auf, sodass Kameramodelle benötigt werden, die eine Übertragung in den dreidimensionalen Raum ermöglichen. Diese werden in 3.3.2 beschrieben. Allerdings entstehen durch die Kameramodelle eine große Ungenauigkeit hinsichtlich der Tiefe. Dementsprechend.

Im Kontext von SLAM existiert die Inverse Depth Parametrisierung nach MONTIEL, CIVERA und DAVISON [37]. Diese ermöglicht es, Objekte zu verwenden, die in größerer Distanz sind, indem die vorhandenen Informationen zur Orientierung genutzt werden. Dabei liefern die vorhandenen Information vor allem die Richtung, in der das Objekt sich befindet. Dementsprechend kann ohne Information zur exakten Position bereits eine Aussage über die Orientierung getroffen werden.

Die Verwendung von Kalmanfiltern für SLAM wird in 3.3.1 beschrieben. Dabei wird nochmal auf die Inverse Depth Parametrisierung in 3.3.1 eingegangen.

3.3.1 SLAM mit Kalmanfiltern

Die Erweiterung des Kalmanfilters im SLAM-Kontext unterscheidet sich nicht für KF, EKF und UKF. In allen Fällen wird der Zustandsvektor $\mathbf{X}$ und die Kovarianzmatrix $\mathbf{P}$ um die entsprechenden Objekte erweitert[15, 42]. Damit ergibt sich ein erweiterter Zustandsvektor nach 3.15 sowie eine erweiterte Kovarianzmatrix nach 3.16. Die Untermatrizen auf der Hauptdiagonalen in 3.16 beschreiben die Kovarianzmatrizen des ursprünglichen Zustands und die der Objekte. Auf den Nebendiagonalen befinden sich die Kreuzvarianzen der Objekte untereinander sowie die mit dem ursprünglichen Zustand.

$$\mathbf{x}_{SLAM} = [\mathbf{x}\ l_0 \dots l_m]^T \tag{3.15}$$

$$\mathbf{P}_{SLAM} = \begin{bmatrix} \mathbf{P_{x,x}} & \mathbf{P_{x,l_0}} & \cdots & \mathbf{P_{x,l_m}} \\ \mathbf{P_{l_0,x}} & \mathbf{P_{l_0,l_0}} & \cdots & \mathbf{P_{l_0,l_m}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P_{l_m,x}} & \mathbf{P_{l_m,l_0}} & \cdots & \mathbf{P_{l_m,l_m}} \end{bmatrix} \tag{3.16}$$

Die Objekte werden hier kontinuierlich dem Zustandsvektor hinzugefügt und sorgen dementsprechend für eine stetige Vergrößerung.
Neben der Erweiterung der Zustandsbeschreibung sind ebenfalls das System- und das Messrauschen sowie die Modelle anzupassen. Im Bewegungsmodell ist zu garantieren, dass eine ortsfestes Objekt keine Translation erfährt und lediglich durch das Systemrauschen beeinflusst wird. Im Beobachtungsmodell ist eine Prädiktion der Messung von Objekten nach 3.17 in Abhängigkeit der aktuellen Position vorzusehen, um die Karte im Korrekturschritt verbessern zu können.

$$z = [z_x\ z_{l_0}\ \dots\ z_{l_m}]^T \tag{3.17}$$

SLAM als Erweiterung des Kalmanfilters hat ferner die Anforderung einer Lokalisierung in Echtzeit. Das bedeutet, dass die Schätzung des nächsten Zustands abgeschlossen sein muss, bis neue Sensordaten zur Verfügung stehen. Andernfalls ist ohne entsprechendes Modell keine akkurate Schätzung mehr möglich. Dementsprechend ist die Größe des Zustandsvektors zu begrenzen. Im Falle der stetigen Integration neuer Objekte ist eine kontinuierliche Verwaltung der Karte notwendig. Hierbei ist neben einer maximalen Objektanzahl im Zustandsvektor eine Abwägung zu treffen, wie diese Grenze einzuhalten ist.

Inverse Depth Parametrisierung

Die Inverse Depth Parametrisierung, eingeführt von Montiel, Civera und Davison [37], beschreibt ein Objekt, sodass trotz größerer Tiefenungenauigkeit eine Aussage über die aktuelle Position und Orientierung getroffen werden kann. Insbesondere Objekte in großer

Entfernung können so schon früh verwendet werden. Das begründet sich der Entkopplung von Richtung und Abstand. Im Gegensatz dazu beschreiben andere Parametrisierungen die Ort eines Objekts direkt ohne verkoppeln damit jegliche alle Dimensionen.
Ein Objekt in Inverse Depth Parametrisierung besteht nach Gl. 3.18 aus der Position, an der dieses zum ersten Mal detektiert wurde $[x\ y\ z]^T$, dem Azimut θ und Höhenwinkel ϕ sowie der inversen Tiefe ρ.

$$\mathbf{y}_i = [x_i\ y_i,z_i,\theta_i\ \phi_i\rho_i] \tag{3.18}$$

Mithilfe von 3.19 kann ein Objekt wieder in eine kartesische Repräsentation überführt werden, wobei $\mathbf{m}$ einen Richtungsvektor von der ersten Detektion hin zum Objekt beschreibt.

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \frac{1}{\rho_i}\mathbf{m}(\theta_i,\phi_i) \tag{3.19}$$

Die Darstellung erlaubt Objekte in großer Distanz in die Landkarte aufzunehmen und für das SLAM-Problem zu nutzen.
Die Nutzung der Parametrisierung im Kontext von SLAM setzt voraus, dass jedes Objekt mittels eines Kameramodells in den Sensorbereich transformiert wird. Hierfür wird zunächst der Vektor $\mathbf{h}_C$ von der Kamera zum Objekt mit Gl. 3.20 berechnet, wobei $\mathbf{r}^{WC}$ den Ortsvektor der Kamera im Weltkoordinatensystem beschreibt.

$$\mathbf{h}_C = \left(\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \frac{1}{\rho_i}\mathbf{m}(\theta_i,\phi_i) \right) - \mathbf{r}^{WC} \tag{3.20}$$

Der Vektor $\mathbf{h}_C$ kann anschließend mithilfe des gewählten Kameramodells aus 3.3.2 transformiert und mit Sensordaten verwendet werden.
Da die Inverse Depth Parametrisierung je Objekt den Zustandsvektor um sechs Einträge vergrößert, ist im Hinblick Rechenkomplexität eine Transformation in kartesische Koordinaten sinnvoll. Zur Abschätzung eines geeigneten Zeitpunkts kann ein Linearitätsfaktor mit 3.21 berechnet werden[21]. Dabei beschreibt P_{yy} die Kovarianzmatrix des betrachteten Objekts und damit $P_{yy}(6,6)$ die entsprechende Varianz der inversen Tiefe. Die Paralaxe α bildet den Winkel zwischen Richtungsvektor m und Abstandsvektor h_C, zu sehen in Abb. 3.2.

$$L_d = \left| \frac{\frac{\partial^2 f}{\partial x^2}\ \sigma_d}{\frac{\partial f}{\partial x}} \right| = \frac{\sigma_d}{d_i} |\cos(\alpha)| \tag{3.21}$$

$$Mit\ \sigma_d = \frac{\sqrt{P_{yy}(6,6)}}{\rho^2} \quad d_i = \|\mathbf{h}_C\| \cos(\alpha) = \frac{\mathbf{m} \cdot \mathbf{h}_C}{\|\mathbf{h}_C\|}$$

Wenn der Linearitätsfaktor $L_d \approx 0$ kann die Funktion annähernd als linear bezeichnet werden, was wiederum eine Erhaltung der gaußschen Eigenschaften ermöglicht. Nach Civera, Davison und Montiel [7] und J. Civera, A. J. Davison und J. M. M. Mon-

TIEL [21] ist bei $L_d \approx 0.1$ eine Transformation in kartesische Koordinaten ohne signifikaten Verlust der gaußschen Eigenschaften möglich.

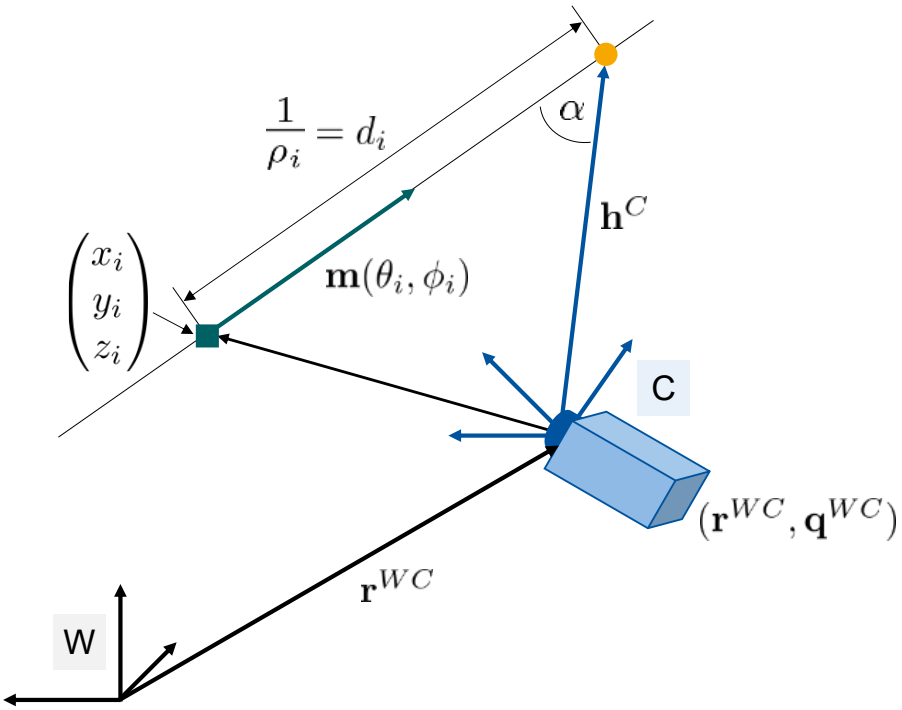


Abbildung 3.2: Grafische Darstellung der Inverse Depth Parametrisierung[37]

3.3.2 Kameramodelle

Kameramodelle dienen im Allgemeinen dazu, die aufgenommenen Bilder in den dreidimensionalen Raum zurück zu transformieren, da Kameras nur eine Projektion der Umgebung warnehmen. Generell kann zwischen drei Modellen unterschieden werden: Lochkamera, Fischauge und Omnidirektional.

Im Lochkameramodell wird ein dreidimensionaler Punkt auf die Bildebene projiziert unter Verwendung einer perspektivischen Transformation nach Gl. 3.23. Dabei beschreibt (X,Y,Z) den dreidimensionalen Punkt im Weltkoordinatensystem, (u,v) die Pixelkoordinaten der Projektion, (c_x,c_y) den Bezugspunkt der Pixelkoordinaten und f_x,f_y die Brennweite der Kamera in die jeweilige Koordinatenrichtung ausgedrückt in Pixel nach Gl. 3.22. Dabei stellen d_x und d_y die Größe eines Pixels in x- bzw. y-Richtung dar.

$$\begin{bmatrix} f_x \\ f_y \end{bmatrix} = \begin{bmatrix} \frac{f}{d_x} \\ \frac{f}{d_y} \end{bmatrix} \tag{3.22}$$

In Matrixschreibweise ergibt sich Gl. 3.24, wobei $[\mathbf{R}|\mathbf{t}]$ eine Rotationstranslationsmatrix vom Weltkoordinatensystem in das entsprechende Kamerakoordinatensystem. Mittels die-

ser Matrix wird die Kamerabewegung oder die Bewegung des beobachteten Objekts beschrieben. Die Matrix **A** fasst die kameraspezifischen Parameter zusammen.

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.23}$$

$$s\tilde{\mathbf{m}} = \mathbf{A}\,[\mathbf{R}|\mathbf{t}]\,\tilde{\mathbf{M}} \tag{3.24}$$

Unter Verwendung von Gl. 3.25 und $z \neq 0$ lässt sich Gl. 3.23 in Gl. 3.26 umschreiben.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{R} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t} \tag{3.25}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{z} \begin{bmatrix} f_x x + c_x \\ f_y y + c_y \end{bmatrix} \tag{3.26}$$

Abb. 3.3 illustriert das Lochkameramodell, wobei die Bezugsebene mit dem Bezugspunkt (c_x,c_y) im Abstand der Brennweiten f liegt. Dabei wurde das Kamerakoordinatensystem so gewählt, dass die Z-Achse durch den Bezugspunkt normal zur Bezugsebene verläuft.

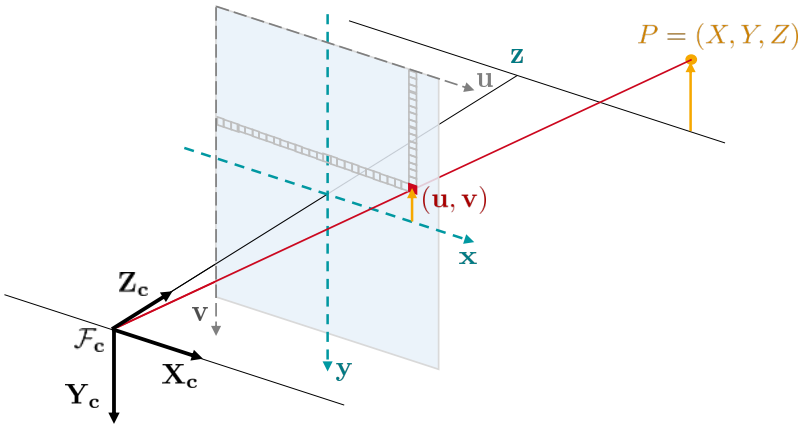


Abbildung 3.3: Grafische Darstellung des Lochkamera Modells

Das Lochkameramodell findet Anwendung in einfachen Kameras, deren Blickfeld mit geeigneten Objektiven auf bis zu 130° erweitert werden kann. Mithilfe einer Fischaugenkamera besteht die Möglichkeit, den Blickwinkel auf bis zu 180° zu erweitern. Allerdings wird weiterhin eine Projektion aufgenommen, wodurch es zu einer größeren Verzerrung kommt. Dadurch kann das Lochkameramodell nicht mehr angewendet werden. Dementsprechend werden für Fischaugenkameras in der Regel eines der vier Modelle stereographische Projektion, äquidistante Projektion, flächentreuer Projektion oder orthographische Projektion

verwendet.

Omnidirektionale Kameras können ein Sichtfeld von 360 Grad in horizontaler Ebene besitzen oder eine Hemisphäre oder den ganzen Raum auf einmal betrachten. Das Modell muss für die Kamera entsprechend der Konfiguration bestimmt werden, da zur Realisierung des Sichtfeldes unterschiedliche Spiegelsysteme verwendet werden. Infolgedessen ist das Kameramodell von der verwendeten Kamera abhängig.

3.4 Rahmenbedingungen

Die Inverse Depth Parametrisierung wird in das von Ammann und Andert [1] beschriebene Framework zur Lokalisierung implementiert. Der Zustand ergibt sich nach Gl. 3.27, wobei $\mathbf{r}_{\mathfrak{W}}$ die Position, $\mathbf{v}_{\mathfrak{W}}$ die Geschwindigkeit und $\mathbf{q}_{\mathfrak{B}}^{\mathfrak{W}}$ die Lage in Form eines Quaternions beschrieben. Position und Geschwindigkeit sind im gewählten Weltkoordinatensystem $\mathfrak{W}$ angegeben und die Lage beschreibt die Rotationen eines körperfesten Koordinatensystems $\mathfrak{B}$ zum Weltkoordinatensystem $\mathfrak{W}$.

$$\mathbf{x} = \begin{bmatrix} \mathbf{r}_{\mathfrak{W}} \\ \mathbf{v}_{\mathfrak{W}} \\ \mathbf{q}_{\mathfrak{B}}^{\mathfrak{W}} \end{bmatrix} \in \mathbb{R}^{10} \tag{3.27}$$

Dieser Zustand wird durch die IMU hochfrequent zwischen 0,1 kHz bis 100 kHz aktualisiert. Da die resultierende Navigation durch inkorrekte Messungen oder Modellfehler anfällig ist, wird weiterhin ein ErrorState nach Gl. 3.28 definiert. Dieser enthält die Fehler in Position $\delta\mathbf{r}_{\mathfrak{W}}$, Geschwindigkeit $\delta\mathbf{v}_{\mathfrak{W}}$ und Lage $\delta\theta_{\mathfrak{W}}^{\mathfrak{W}}$. Zusätzlich dazu sind noch die Fehler der IMU in Form einer Verschiebung für Beschleunigung $\mathbf{a}_{\mathfrak{B}}$ und Gyroskop $\omega_{\mathfrak{B}}$ inkludiert. Weiterhin ist der Modellabweichung der Gravitation $\mathbf{g}_{\mathfrak{W}}$ Teil des ErrorStates.

$$\delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{r}_{\mathfrak{W}} \\ \delta\mathbf{v}_{\mathfrak{W}} \\ \delta\theta_{\mathfrak{W}}^{\mathfrak{W}} \\ \mathbf{b}_{\mathbf{a}_{\mathfrak{B}}} \\ \mathbf{b}_{\omega_{\mathfrak{B}}} \\ \mathbf{b}_{\mathbf{g}_{\mathfrak{W}}} \end{bmatrix} \in \mathbb{R}^{18} \tag{3.28}$$

Die Fehlerschätzung erfolgt mit dem UKF bei einer niedrigeren Frequenz. Der geschätzte Fehler wird anschließend zur Korrektur des durch die IMU geschätzten Zustands

verwendet. Die Korrektur ergibt sich nach Gl. 3.29, wobei $\delta\mathbf{q}_{\mathfrak{W}}^{\mathfrak{W}} = f(\delta\theta_{\mathfrak{W}}^{\mathfrak{W}})$.

$$\mathbf{x} = \begin{bmatrix} \tilde{\mathbf{r}}_{\mathfrak{W}} + \delta\mathbf{r}_{\mathfrak{W}} \\ \tilde{\mathbf{v}}_{\mathfrak{W}} + \delta\mathbf{v}_{\mathfrak{W}} \\ \delta\mathbf{q}_{\mathfrak{W}}^{\mathfrak{W}} \otimes \tilde{\mathbf{q}}_{\mathfrak{B}}^{\mathfrak{W}} \end{bmatrix} \tag{3.29}$$

Die Kovarianzmatrix $\mathbf{P}$ repräsentiert die Ungenauigkeit des ErrorStates und damit auch die der Navigation[1].

4 Modellierung und Implementierung

Die Verwendung der Inverse Depth Parametrisierung im UKF-SLAM kann in drei Abschnitte gegliedert werden. Zum einen sind die Objekte mit den gegebenen Messwerten entsprechend im Zustandsvektor zu initialisieren. Bei der Initialisierung ist zunächst eine Weiterverarbeitung der Kovarianzmatrix zu beachten. Hinzu kommt die Anforderung, die Initialisierung in kürzester Zeit zu vollziehen.
Weiterhin wird wie in Abschnitt 3 beschrieben, eine Verwaltung der entstehenden Karte benötigt, um den Rechenaufwand zu begrenzen und die Echtzeitfähigkeit zu erhalten. Dabei wird einerseits auf eine Begrenzung des Zustandsvektors, andererseits auf die Transformation in kartesische Koordinaten zurückgegriffen.
Hinzu kommt eine Anpassung der Parametrisierung infolge von Schätzungsungenauigkeiten und -unstetigkeiten von Winkeln, die in der Umgebung von 0° oder 360° liegen. In dieser Region kann eine Schwankung zu einer fehlerhaften sowie ungenauen Schätzung führen.

Gl. 4.1 zeigt den Aufbau des Zustandsvektors. Am Anfang steht der zu schätzende Zustand. Darauf folgen die Objekte in kartesischer und Inverse Depth Repräsentation. Die Reihenfolge ist so gewählt worden, damit Objekte einfach am Ende dem Zustandsvektor angefügt werden können. Objekte in kartesischer Repräsentation können nur durch eine interne Transformation aus der Inverse Depth Repräsentation dem Zustandsvektor hinzugefügt werden.

$$X^{SLAM} = [x^T \; l_C^T \; l_I^T]^T \tag{4.1}$$

4.1 Anpassung der Parametrisierung

Die Parametrisierung von Montiel, Civera und Davison [37] erzeugt den Richtungsvektor aus den zugehörigen Winkeln Azimut θ und Höhenwinkel ϕ. Durch die Verwendung von Winkeln können Probleme in der Umgebung von 0° bzw. 360° auftreten, da die Funktionen Sinus und Kosinus vom Wertebereich $0 - 2\pi$ bzw. 0° bis 360° abbilden. Infolgedessen kann ein Winkel nahe 0° durch die Ungenauigkeiten einen negativen Wert zugewiesen bekommen. Dieser wird allerdings durch die Beschränkung des Wertebereichs auf eine Umgebung von 360° abgebildet.
Aufgrund dessen kann trotz einer guten Schätzung ein größerer Fehler entstehen. Das kann

einerseits vermieden werden, indem die Winkel zunächst in den zugehörigen Quadranten durch Berechnung des Tangens mit Sinus und Kosinus abgebildet werden. Anschließend wird der arctan 2 des Ergebnisses gezogen. Dieser bildet die Winkel im zweiten Quadranten ab. Das behebt den Fehler allerdings nicht vollständig, sondern verschiebt die Problematik nur.
Andererseits besteht die Möglichkeit der direkten Verwendung des Richtungsvektors. Dieser wird direkt aus den Pixelkoordinaten mit Hilfe eines der Kameramodelle aus 3.3.2 erzeugt. Dabei wird aus den Pixelkoordinaten (u,v) nach Gl. 4.2 mit der inversen Projektion ein Vektor gebildet und anschließend normiert, um den Richtungsvektor zu erhalten.

$$\mathbf{v} = \begin{bmatrix} \frac{(u - c_x)}{f_x} & \frac{(u - c_y)}{f_x} & 1 \end{bmatrix}^T \tag{4.2}$$

Ein Vorteil des Richtungsvektors bildet die Eindeutigkeit durch die zusätzliche Bedingung der Normierung. Das verringert die Fehleranfälligkeit im Vergleich zur Anwendung mit Winkeln. Allerdings erhöht sich dadurch die Dimension und erzeugt eine Korrelation der Dimensionen.
Wie in Abschnitt 3.3.1 beschrieben, bietet die Inverse Depth Parametrisierung den Vorteil, Objekte ohne Tiefeninformationen zur Orientierung zu nutzten. Unter Verwendung der Winkel besteht dadurch die Gefahr, dass Fehler induziert werden die zur Divergenz des UKF führen. Dementsprechend wurde die Parametrisierung angepasst, sodass der Richtungsvektor anstatt der Winkel für ein Objekt in den Zustandsvektor **dX** aufgenommen wird. Damit ergibt sich die Parametrisierung ein Objekt nach Gl. 4.3.

$$\mathbf{p}_i = \begin{bmatrix} \mathbf{X}_i \\ \mathbf{v}_i^{dir} \\ \rho_i \end{bmatrix} \in \mathbb{R}^7 \tag{4.3}$$

Die Erhöhung der Dimension wird durch die entsprechende Kartenverwaltung aus Abschnitt 4.4 kompensiert, um die Echtzeitfähigkeit zu erhalten.

4.2 Initialisierung neuer Objekte

Die Initialisierung neuer Objekte erfolgt unter Verwendung der Unscented Transform (UT), da die Transformation von den gegebenen Messwerten in die Inverse Depth Parametrisierung eine nicht-lineare Funktion darstellt. Weiterhin werden zunächst alle Objekte in der Inverse Depth Parametrisierung initialisiert, um die Initialisierung zu vereinfachen. Das erlaubt jedes Objekt an den bestehenden Zustandsvektor anzufügen. Ein Objekt mit Tiefeninformationen kann anschließend wie in Abschnitt 4.4.3 beschrieben in kartesische Koordinaten transformierten werden.
Für die UT existieren verschiedenste Möglichkeiten der Berechnung der Sigmapunkte. Aus

Gründen der Konsistenz fiel die Wahl für die Initialisierung auf die Augmented Scaled UT unter Verwendung der Parameter α, β und κ. Die Erweiterung des Zustandsvektors und der Kovarianzmatrix erfolgt grundsätzlich nach Gl. 4.4. Dabei wird der bisherige Zustandsvektor um den aktuellen Messvektor erweitert.

$$x^a = [x^T \ z^T]^T \tag{4.4}$$

Die Größe der Messdaten hängt von der Detektion ab, wobei zwischen einer zwei- und dreidimensionalen unterschieden werden kann. In beiden Fällen wird aus den Pixeln unter Verwendung des Lochkameramodells und einer UT ein Richtungsvektor erzeugt. Im Falle einer dreidimensionalen Messung stehen zusätzlich Tiefeninformationen z_{dist} zur Verfügung.
Die zweidimensionale Detektion erfordert eine Erweiterung um eine feste Tiefe, damit zu jeder Initialisierung eine inverse Tiefe gesetzt werden kann. Analog ist auch die Kovarianzmatrix R der Messung zu erweitern um eine initiale Varianz. Beschrieben von Montiel, Civera und Davison [37] werden diese nach Gl. 4.5 gesetzt, wobei d_{min} die minimale Sichtweite der Kamera beschreibt

$$\rho_0 = \frac{\rho_{min}}{2},\ \sigma_0 = \frac{\rho_{min}}{4}\ mit\ \rho_{min} = \frac{1}{d_{min}} \tag{4.5}$$

Im Anschluss werden die Sigmapunkte gezogen und die entsprechenden Gewichte berechnet.

Die Sigmapunkte werden durch eine Initialisierungsfunktion 4.6 propagiert, wobei $\mathcal{Y} \in \mathbb{R}^{n+7\times m}$ und n und m die Größe des Zustandsvektors vor der Initialisierung bzw. die Anzahl der Sigmapunkte beschreiben.

$$\mathcal{Y} = f_{init}(\mathcal{X}) \tag{4.6}$$

Die Funktion f_{init} erweitert den Zustand für jedes Objekt. Dabei ist zur Erzeugung einer Karte die Darstellung in einem Weltkoordinatensystem zu wählen. Dadurch kann im Bewegungsmodell jedes Objekt als konstant angesehen werden und ist unabhängig von Position und Lage des Flugobjekts. Die Sensordaten z_{dir} und z_{dist} sind allerdings im sensorfesten Koordinatensystem angegeben. Dementsprechend ist die Position der ersten Detektion die des Sensors. Dieser hat einerseits eine Translation durch PTU und Befestigung, andererseits eine Rotation durch die Lage des Flugobjekts. Dadurch ergibt sich unter Beachtung der aktuellen Position $\mathbf{r}_{\mathfrak{W}}$ und Rotation $\mathbf{q}^{\mathfrak{W}}_{\mathfrak{B}}$ die aktuelle Sensorposition nach Gl. 4.7.

$$\mathbf{r}^{\mathfrak{W}}_{\mathfrak{C}} = \mathbf{r}_{\mathfrak{W}} + \mathbf{q}^{\mathfrak{W}}_{\mathfrak{B}} \otimes \mathbf{t}^{\mathfrak{S}}_{\mathfrak{B}} \tag{4.7}$$

Um zu einer korrekten Darstellung der Translation $\mathbf{t}^{\mathfrak{S}}_{\mathfrak{B}}$ vom körperfesten zum sensorfesten Koordinatensystem unter Mithilfe der aktuellen Lage des UAV in das Weltkoordinaten-

system zu erhalten, ist zu Rotieren.
Da auch der Richtungsvektor im sensorfesten Koordinatensystem angegeben wird, ist dieser ins Weltkoordinatensystem nach Gl. 4.8 zu transformieren.

$$\mathbf{z}_{dir}^{\mathfrak{W}} = \mathbf{q}_{\mathfrak{B}}^{\mathfrak{W}} \otimes \mathbf{q}_{\mathfrak{PTU}_{\mathfrak{B}}}^{\mathfrak{B}} \otimes \mathbf{q}_{\mathfrak{PTU}_{\mathfrak{C}}}^{\mathfrak{PTU}_{\mathfrak{B}}} \otimes \mathbf{q}_{\mathfrak{C}}^{\mathfrak{PTU}_{\mathfrak{C}}} \otimes \mathbf{z}_{dir}^{\mathfrak{C}} \tag{4.8}$$

Die Rotation vom sensorfesten Koordinatensystem setzt sich zusammen aus den vier einzelnen Rotationen

- vom Sensor zur PTU $\mathbf{q}_{\mathfrak{S}}^{\mathfrak{PTU}}$,
- der PTU zur Befestigung $\mathbf{q}_{\mathfrak{PTU}_C}^{\mathfrak{PTU}_B}$,
- der Befestigung zum körperfesten Koordinatensystem $\mathbf{q}_{\mathfrak{PTU}_S}^{\mathfrak{B}}$ und
- vom körperfesten in das Weltkoordinatensystem $\mathbf{q}_{\mathfrak{B}}^{\mathfrak{W}}$.

Die Initialisierung der Objekte mit der von Julier, Uhlmann und Durrant-Whyte [23] beschriebenen UT ist einzeln durchzuführen aufgrund sinkender Stabilität, wenn mehrere Objekte gleichzeitig initialisiert werden sollen. Die Sensitivität der UT hinsichtlich der Wahl der Sigmapunkte ist hinreichend bekannt, sodass verschiedene Ansätze zur Bewältigung der Problematik bestehen. Bei der Verwendung der Augmented Scaled UT stehen drei Parameter zur Steuerung dieser zu Verfügung. Je nach Parameterset können Kovarianzmatrizen auftreten, die nicht mehr positiv definit sind. Infolgedessen ist keine weitere UT mehr möglich und die Initialisierung unbrauchbar. Die Auswahl der korrekten Parameter ist in 4.2.1 dargestellt.
Die einzelne Initialisierung bringt eine Problematik hinsichtlich der Laufzeit mit, da der benötigte Rechenaufwand einer UT in der Ordnung $O(n^2)$ ist, wobei n der Größe des Zustandsvektors entspricht. Gleichungen 4.9 und 4.10 zeigen den benötigten Rechenaufwand für die Initialisierung aller m Objekte. Dabei stehen O_g und O_e für eine gemeinsame bzw. einzelne Initialisierung. Im betrachteten Fall hat die zu einem Objekt korrespondierende Messung immer die Größe 4.
Dementsprechend ist entweder ein Set von Sigmapunkten zu finden, die eine gemeinsame Initialisierung erlauben oder eine Auswahl der zu initialisierenden Objekte im Vorhinein zu treffen. Zudem besteht die Möglichkeit der Verwendung des Square-Root UKF. Dieser ist für additives System- und Messrauschen numerisch stabiler und besitzt eine geringere Komplexität.

$$O_g = (n + m * 4)^2 \tag{4.9}$$

$$O_e = \sum_{i=0}^{m-1} ((n + 7 * i) + 4)^2 \tag{4.10}$$

4.2.1 Berechnung der Sigmapunkte

Aufgrund der Sensitivität der UT auf die Wahl der Sigmapunkte muss ein Parameterset gefunden werden, dass eine gute Initialisierung ermöglicht. Die Bewertung der Initialisierung kann an zwei Eigenschaften des resultierenden Zustandsvektors und der Kovarianzmatrix festgemacht werden. Zum einen muss für eine Weiterverarbeitung die Kovarianzmatrix immer symmetrisch positiv definit sein, da anderenfalls keine Choleskyzerlegung möglich ist. Andererseits können die Ko- und Kreuzvarianzmatrizen der initialisierten Objekte einen Aufschluss über die Initialisierung geben.
In [48] wird der Ansatz gewählt, dass ein modellspezifisches optimales Parameterset $\theta = [\alpha,\beta,\kappa]$ für einen UKF mittels Machine Learning Algorithmen bestimmt werden kann.
Da im Zuge dieser Arbeit der Fokus der Inverse Depth Parametrisierung im Kontext von SLAM steht, wurde auf eine Implementierung komplexerer Algorithmen verzichtet. Stattdessen wurde das Set mithilfe einer einfache Grid-Search bestimmt. Bei dieser wird auf einem Intervall ein äquidistantes Gitter aus den freien Parametern erzeugt, sodass alle möglichen Kombinationen an den Knotenpunkten dargestellt werden. Die Knotenpunkte dienen als Eingangparameter zur weiteren Berechnung.

Zur Reduktion des Rechenaufwands wurde ein vereinfachtes Modell des im Navigation-Framework verwendeten Modells benutzt. Der Zustandsvektor enthält initial ausschließlich Informationen über die Abweichung der aktuellen Position und Rotation. Beide sind initial auf 0 gesetzt. Die verwendete Kovarianzmatrix enthält demnach auch nur die Abweichungen hinsichtlich Position und Rotation. Demzufolge wird angenommen, dass das körperfeste und kamerafeste Koordinatensystem identisch ist und keine Transformation zwischen diesen notwendig ist.

Das Hauptkriterium zur Bewertung der Initialisierung, ob die entstehende Kovarianzmatrix symmetrisch positiv definit ist. Die Symmetrie ergibt sich implizit durch die Berechnung, da die Summe symmetrischer Matrizen erneut eine symmetrische Matrix bildet. Infolgedessen ist nur zu prüfen, ob die resultierende Matrix positiv definit ist. Andernfalls kann das Parameterset nicht verwendet werden. Die möglichen Parametersets können anschließend hinsichtlich der Initialisierung bewertet werden.

Die Bewertung der Initialisierung erfolgt durch einen Vergleich von Zustandsvektor und Kovarianzmatrix nach Gl. 4.11 und Gl. 4.12. Der Referenzzustandsvektor setzt sich aus dem initial gesetzten Zustandsvektor sowie den Objekten, bestehend aus den Mittelwerten der korrigierten Position $X + dX$ und den jeweiligen Messungen z_{dir} und z_{dist}, zusammen. Dabei wurde z_{dir} in das Weltkoordinatensystem transformiert. Die Referenzkovarianzmatrix wird zunächst als Blockdiagonalmatrix initialisiert, bestehend aus der ursprünglichen Kovarianzmatrix, der Kovarianzmatrix bzgl. der Position$P_{l,x}$, dem Messrauschen des Richtungsvektors $\widehat{R}_{dir}$ sowie die Inverse des Messrauschen der Tiefe R_{dist}. Die Matrix $\widehat{R}_{dir}$ beinhaltet dabei schon die zusätzliche Ungenauigkeit durch die Rotation des Richtungs-

vektors. Auf den Nebendiagonalen befinden sich die Kreuzvarianzmatrizen der Position und des Richtungsvektors mit der initialen Kovarianzmatrix.

$$dX_{ref} = \left[dX_{init}^T \; (X + dX)^T \; z_{dir}^T \; \frac{1}{z_{dist}} \right]^T \tag{4.11}$$

$$P_{ref} = \begin{bmatrix} P_{init} & P_{x,init} & P_{v,init} & 0 \\ P_{x,init} & P_{l,x} & 0 & 0 \\ P_{v,init} & 0 & \widehat{R}_{dir} & 0 \\ 0 & 0 & 0 & \frac{1}{R_{dist}} \end{bmatrix} \tag{4.12}$$
$$mit \; P_{init} = diag(P_x, P_{rot}), \; \widehat{R}_{dir} = P_{rot} * R_{dir}$$

Die Wahl der Referenzgrößen ergeben sich einerseits aus der Vorgabe, dass eine Initialisierung keinen Einfluss auf die Schätzung haben soll. Dementsprechend soll der bisherige Mittelwert übertragen werden. Andererseits sind zum Zeitpunkt der Initialisierung sowohl Richtungsvektor als auch Tiefe und inverse Tiefe unabhängig von der Position. Der Zusammenhang zwischen der aktuellen Position und des Objekts wird vollständig durch die korrigierte Position $X + dX$ und der zugehörigen Kovarianzmatrix P_x dargestellt. Dementsprechend enthält eine gute Initialisierung keine Kreuzkorrelationen von Richtungsvektor, inverser Tiefe und der Kovarianzmatrix der initialen Position.
Infolge der Zusammenfassung der Translation in $X + dX$, sind Kreuzkorrelationen zwischen der Kovarianzmatrix der aktuellen Position P_x und der Kovarianzmatrix $P_{l,x}$ zu erwarten. Aufgrund des reduzierten Modells bilden die Kreuzkorrelationen der aktuellen Position mit dem Zustandsvektor die initial gewählte Kovarianzmatrix der Position.
Die Kovarianzmatrix des Richtungsvektors ergibt sich als Produkt der Kovarianzmatrix der Rotation mit der des Messrauschens infolge der Rotation des Richtungsvektors. Die inverse Tiefe ist unabhängig vom aktuellen Zustand, da diese eine erste Approximation des Abstands darstellt und dementsprechend einer hohen Ungenauigkeit unterliegt.
Die Kreuzkorrelationen von Position und Rotation mit der Detektionsposition bzw. dem Richtungsvektor ergeben aufgrund des vereinfachten Models die initiale Positionsmatrix bzw. Rotationsmatrix.

Nachdem die Referenzgrößen erzeugt wurden, wird eine Initialisierung der Objekte für jedes Parameterset durchgeführt. Das Ergebnis wird von nicht verwendbaren Sets bereinigt, um es anschließend mit den Referenzgrößen zu vergleichen. Aus der Konstruktion der Referenzgrößen ergeben sich einerseits die Betrachtung der Abweichung vom Mittelwert und andererseits die Abweichung der Ko- und Kreuzvarianzmatrizen. Damit ergeben sich zusätzlich zur Abweichung der ursprünglichen Matrix je initialisiertem Objekt sechs zu betrachtende Matrizen.
Für diese Matrizen und die Abweichung des Mittelwertes wird jeweils die mittlere Abweichung bestimmt. Ist die mittlere Abweichung kleiner als die numerisch mögliche Genauig-

keit, wird diese mit 0 angenommen.
Die Parametersets mit minimaler Abweichung in den einzelnen Bewertungsabschnitten werden in die engere Auswahl genommen. Demnach existieren abschließend maximal $2+6m$ mögliche Parametersets. Aus diesen ist anschließend eine Wahl zu treffen.

Die Erhaltung einer positiv definiten Kovarianzmatrix kann neben der Optimierung des Parametersets θ durch eine Anpassung der Initialisierung erfolgen. Zum einen kann die UT, wie von Julier, Uhlmann und Durrant-Whyte [22, S. 481] beschrieben, modifiziert werden, um die positive Definitheit zu erhalten. Das ist insbesondere bei der Wahl von $\lambda < 0$ möglich, da dann das zum Mittelwert gehörende Gewicht kleiner 0 ist.
Die Modifikation besteht in der Verwendung des zum Mittelwert gehörenden transformierten Sigmapunkt $\mathcal{Y}_0$ als Referenzgröße anstelle des errechneten Mittelwerts. Infolgedessen verschwindet der Einfluss des Gewichts w_0^c und dementsprechend auch von β. Somit ist das Parameterset auf $\theta^{MOD} = [\alpha,\kappa]$ reduziert. Diese Modifikation bestimmt nicht für alle Parametersets positiv definite Matrizen, konnte allerdings die Initialisierung stabilisieren. Zu erkennen ist dies an einer höheren Anzahl akzeptierter Parametersets bei gleichen Eingangparametern.

4.2.2 Vorauswahl der Objekte

Wenn neue Objekte wie beschrieben nicht gemeinsam initialisiert und dementsprechend nacheinander dem Zustandsvektor hinzugefügt werden, führt dies zu einer Problematik hinsichtlich der Rechenzeit, da die UT quadratisch mit der Größe des Zustandsvektors steigt. Das kann je nach Anzahl neuer Objekte zu einer signifikanten Erhöhung der Rechenzeit führen. Infolgedessen ist eine Vorauswahl über die zu initialisierenden Objekte zu treffen.
Zudem ist der UKF selbst in der Ordnung $O(n^2)$, weshalb die maximale Anzahl der Objekte begrenzt ist. Dementsprechend ist eine gezielte Auswahl der zu initialisierenden Objekte sinnvoll. So werden nur die Objekte in den Zustandsvektor aufgenommen, die den größten Beitrag zur Schätzung leisten. Die Vorauswahl erfolgt einem Clustering Ansatz, um das aktuelle Blickfeld mit einer geringen Anzahl Objekte bestmöglich abbilden zu können. Eine Auswahl über den Informationsgehalt eines Objekt, wie in Abschnitte 4.4 beschrieben, ist nicht sinnvoll, da die Transformation in einen Richtungsvektor für jedes Objekt mit derselben Varianz geschieht. Folglich liefert der Informationsgehalt in diesem Fall keinen validen Wert zur Auswahl.
Jeder Cluster stellt eine Region des Blickfeldes dar. Dementsprechend wird das Objekt mit dem geringsten Abstand zum Zentrum des zugehörigen Clusters repräsentativ für die Region ausgewählt.
Als Clustering Algorithmus wird kmeans++ verwendet, da dieser einerseits eine einfache Implementierung und andererseits eine Vorgabe der Cluster-Anzahl ermöglicht. Dadurch

kann die Anzahl neuer Objekte und damit die Größe des Zustandsvektors direkt gesteuert werden.

KMeans ordnet der gewählten Anzahl Clusterzentren die eingegebenen Datenpunkte nach einer Metrik zu. Ziel ist dabei die Minimierung des quadratischen Abstands zwischen jedem Datenpunkt und seines nächsten Zentrums nach Gl. 4.13.

$$\Phi = \sum_{x \in \mathcal{X}} \min_{c \in \mathcal{C}} \|x - c\|^2 \tag{4.13}$$

Algorithmus 5 stellt die Bestimmung der Clusterzentren mit den Datenpunkten $\mathbf{X}$ und der Anzahl Cluster k als Eingangparameter dar. Kmeans verschiebt damit die Clusterzentren, bis diese keine weitere Veränderung durch Entfernen oder Hinzufügen neuer Datenpunkte erfahren.

Algorithm 5 Kmeans

1: **procedure** $k - means$ ($\mathbf{X}$, k)
2: Setze zufällig k initiale Zentren $\mathcal{C} = \{c_1, \ldots, c_k\}$
3: **repeat**
4: **for all** $i \in \{1, \ldots, k\}$ **do**
5: Für Cluster C_i ordne $\mathcal{X} \in \mathbf{X}$ zu mit $\|\mathcal{X} - c_i\| < \|\mathcal{X} - c_j\|, \forall j \neq i$
6: **end for**
7: **for all** $i \in \{1, \ldots, k\}$ **do**
8: $c_i \leftarrow \frac{1}{|C_i|} \sum_{x \in C_i} x$
9: **end for**
10: **until** $\mathcal{C}_n = \mathcal{C}_{n-1}$
11: **end procedure**

Zur Erhöhung der Konvergenzgeschwindigkeit setzt kmeans++ die initialen Zentren nach Algorithmus 6 mit $D(x)$ als kürzesten Abstand eines Datenpunkts x zum nächstgelegenen Zentrum. Die Wahrscheinlichkeit, dass ein Datenpunkt zu einem Zentrum wird, steigt nach Zeile 6, je weiter es von den bereits gewählten Clusterzentren entfernt ist.

Algorithm 6 kmeans++ Zentren

1: **procedure** $kmeans++ - Seeding(\mathbf{X}, k)$
2: Wähle initial zufällig $c_1 \in \mathbf{X}X$
3: **repeat**
4: **for all** $x \in \mathbf{X}$ **do**
5: Berechne $D(x) = \min_{c \in \mathcal{C}} \|x - c\|$
6: **end for**
7: Wähle $c_i = x' \in \mathbf{X}$ mit $p(x') \propto \frac{D(x')^2}{\sum_{x \in \mathbf{X}} D(x)^2}$
8: **until** $\dim(\mathcal{C}) = k$
9: **end procedure**

Zur Vorauswahl der Objekte wird die euklidische Norm verwendet, da ein Clustering hinsichtlich des Ortes bzw. der Abstände der Objekte zueinander erfolgen soll.
Kmeans ordnet als partitionierender Clustering Algorithmus alle Datenpunkte einem Cluster zu. Dementsprechend können einzelne Datenpunkte Cluster bilden. Dadurch werden Objekte mit Tiefeninformationen einem anderen Cluster zugeordnet, als die ohne Tiefeninformationen. Da die Objekte meist keine Tiefeninformationen haben, bilden diese einen eigenen Cluster und werden entsprechend direkt aufgenommen. Abbildung 4.1 zeigt das Ergebnis nach dem Clustering mit 40 Objekten und 10 Clustern. Jedes Objekt wird durch einen Punkt gekennzeichnet, wobei Objekte der gleichen Farbe demselben Cluster zugeordnet werden. Die Asterisk kennzeichnen die Objekte, die sich am nächsten zum exakten Clusterzentrum befinden.
Zu erkennen ist, dass Objekte in der Inverse Depth Parametrisierung in einer Ebene liegen und in mehrere Cluster aufgeteilt werden. Objekte mit Tiefeninformationen bilden einzelne Cluster und werden dementsprechend als Zentrum direkt übernommen.

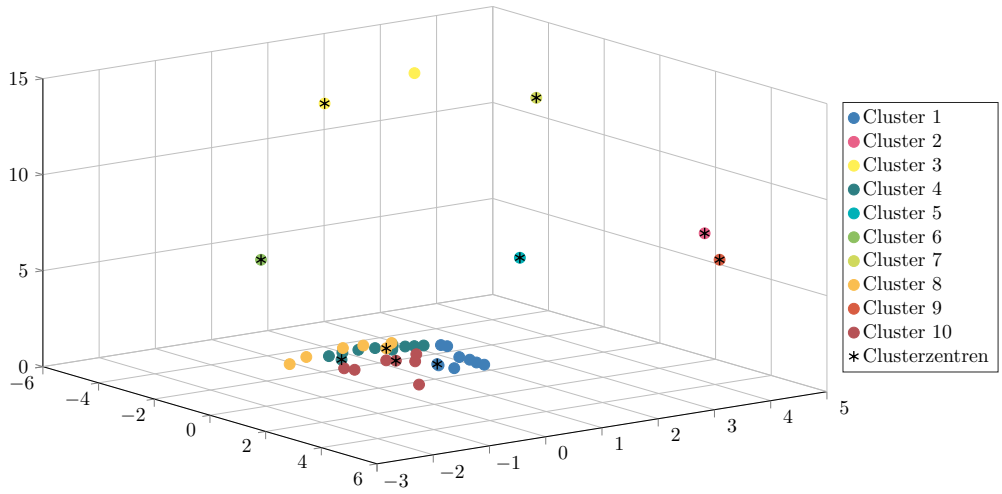


Abbildung 4.1: Clustering neuer Objekte mit kmeans++

Die einzelne Initialisierung birgt aufgrund der stark erhöhten Komplexität erhebliche Nachteile gegenüber der gemeinsamen. Das macht eine Vorauswahl der Objekte zwingend notwendig. Im Gegensatz dazu bietet der Sqaure Root UKF nach van der Merwe und Wan [50] die Möglichkeit einer gemeinsamen Initialisierung mit dem Nachteil, dass dieser in der Form nur mit additivem Rauschen verwendbar ist.
Holmes, Klein und Murray [19] beschreiben eine Erweiterung des SRUKF, sodass die Wurzel der Kovarianzmatrix S um System und Messrauschen erweitert werden kann. Damit kann auch nicht additives Rauschen verwendet werden. Dementsprechend wird der UKF durch den SRUKF ersetzt, da die gemeinsame Initialisierung unabhängig der Parametrisierung funktioniert. Zudem ist der SRUKF numerisch stabiler, sowie von geringerer Komplexität.

4.3 Prädiktion der Messdaten

Die Verwendung der Inverse Depth Parametrisierung für SLAM benötigt ein entsprechendes Sensormodell. Die Kamera detektiert Bilder, die mithilfe der gängigen Verfahren ausgewertet werden, sodass einzelne Objekte mit Pixelkoordinaten beschrieben werden. Diese Pixelkoordinaten werden mittels einer UT in einen Richtungsvektor unter Verwendung von Gl. 4.2 transformiert, wobei die Varianz hinsichtlich der Pixelgenauigkeit gewählt wird. Zur Verwendung der Objekte in Inverse Depth Parametrisierung ist eine Transformation in kartesische Koordinaten nach Gl. 4.14 notwendig. Dabei ist durch die Normalisierung gewährleistet, dass die Länge des Richtungsvektors $\mathbf{v}^{dir}$ immer eins ist.

$$\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \frac{1}{\rho_i} \frac{\mathbf{v}_i^{dir}}{\|\mathbf{v}_i^{dir}\|} \tag{4.14}$$

Die Position $\mathbf{x}$ bildet die geschätzte Position des Objekts im Weltkoordinatensystem. Um diese Position mit den Sensordaten verwenden zu können, wird der Abstandstandsvektor $\mathbf{h} = \mathbf{x} - \mathbf{X}$ zwischen der aktuellen Position des UAV und des Objekts gebildet. Dieser ist anschließend in das sensorfeste Koordinatensystem zu transformieren.

$$\mathbf{h}^{\mathfrak{S}} = \mathbf{A}_{\mathfrak{C}}^{\mathfrak{PTU}_{\mathfrak{C}}} \otimes \mathbf{A}_{\mathfrak{PTU}_{\mathfrak{C}}}^{\mathfrak{PTU}_{\mathfrak{B}}} \otimes \mathbf{A}_{\mathfrak{PTU}_{\mathfrak{B}}}^{\mathfrak{B}} \otimes \mathbf{q}_{\mathfrak{B}}^{\mathfrak{W}} \otimes \mathbf{h}^{\mathfrak{W}} \tag{4.15}$$

In Gl. 4.15 beschreiben $\mathbf{A}_{\mathfrak{S}}^{\mathfrak{PTU}}$, $\mathbf{A}_{\mathfrak{PTU}}^{\mathfrak{PTU}}$ und $\mathbf{A}_{\mathfrak{PTU}}^{\mathfrak{B}}$ die affinen Abbildungen von PTU zu Sensor, Befestigung zur PTU und körperfesten Koordinatensystem zur PTU. Diese enthalten nach Gl. 4.16 entsprechend sowohl Translation $\mathbf{t}$ als auch Rotation $\mathbf{R}$.

$$\mathbf{A}\mathbf{x} = \mathbf{R}\mathbf{x} + \mathbf{t} \tag{4.16}$$

Der Abstandstandsvektor vom sensorfesten Koordinatensystem wird anschließend als prädizierter Messwert normalisiert, um ausschließlich den Richtungsvektor zum entsprechenden Objekt zu verwenden. Falls neben den optischen Sensordaten Informationen zur Entfernung zur Verfügung stehen, wird zusätzlich die Länge des Abstandstandsvektors als Messwert prädiziert. Dabei werden als erstes Objekte ohne und anschließend mit Tiefeninformationen wie in Gl. 4.17 prädiziert.

$$\mathcal{Z}_{k|k-1_i} = \left[z_{dir,0}^T, \ldots, z_{dir,l}^T, [z_{dir}^T, z_{dist}]_0, \ldots, [z_{dir}^T, z_{dist}]_s \right]^T \tag{4.17}$$

Die Prädiktion wird für jeden Sigmapunkt durchgeführt und anschließend nach Gl.4.18 zu einer prädizierten Messung zusammengesetzt.

$$\widehat{\mathbf{z}}_{k|k-1_i}^{-} = \sum_{i=0}^{2L} w_i^{(m)} \mathcal{Z}_{k|k-1_i} \tag{4.18}$$

Diese wird dann anschließend, wie in Algorithmus 4 und Gl. 4.19 beschrieben, zur Korrektur des Zustands und zur Korrektur der Wurzel der Kovarianzmatrix verwendet.

$$\begin{aligned}
\mathbf{S}_{\widehat{\mathbf{z}}_k} &= qr\left\{\left[\sqrt{W_i^{(c)}}(\mathcal{Z}_{1:2L,k|k-1} - \widehat{\mathbf{z}}_k^-)\ \sqrt{R}\right]\right\} \\
\mathbf{S}_{\widehat{\mathbf{z}}_k} &= cholupdate(\mathbf{S}_{\widehat{\mathbf{z}}_k},\ \mathcal{Z}_{0,k|k-1} - \widehat{\mathbf{z}}_k^-,\ W_0^{(c)}) \\
\mathbf{P}_{\mathbf{x}_k\mathbf{z}_k} &= \sum_{i=0}^{2L} W_i^{(c)}[\mathcal{X}_{i,k|k-1} - \widehat{\mathbf{x}}_k^-][\mathcal{Z}_{i,k|k-1} - \widehat{\mathbf{z}}_k^-]^T \\
\mathcal{K}_k &= (\mathbf{P}_{\mathbf{x}_k\mathbf{z}_k}/\mathbf{S}_{\widehat{\mathbf{z}}_k}^T)/\mathbf{S}_{\widehat{\mathbf{z}}_k} \\
\widehat{\mathbf{x}}_k &= \widehat{\mathbf{x}}_k^- + \mathcal{K}_k(z_k - \widehat{\mathbf{z}}_{k|k-1_i}^-)
\end{aligned} \tag{4.19}$$

4.4 Kartenverwaltung

Im Hinblick auf Rechenzeit ist, wie in Abschnitt 2.2.2 beschrieben, eine Verwaltung der erzeugten Karte notwendig. Bei der optischen Navigation können in Abhängigkeit des Sensors in jedem Zeitschritt eine größere Menge neuer Objekte detektiert werden. Dementsprechend ist neben einer Vorauswahl, beschrieben in Abschnitt 4.2.2, eine Abwägung zu treffen, welche und wieviele Objekte entfernt werden müssen.
Einerseits besteht das Bestreben nach einer maximalen Genauigkeit der Karte. Andererseits entstehen neue Informationen zur Umgebung mit neuen Objekte, was dazuführt, dass sich zwischen den alten und neuen Objekten entschieden werden muss. Diese können ggf. einen besseren Beitrag zur Lokalisierung leisten als Objekte, die seit einigen Zeitschritten nicht mehr detektiert wurden, aber eine höhere Genauigkeit aufweisen.

Diese Abwägung spiegelt sich in der Verwendung einer kurzfristigen und langfristigen Karte wieder. Erstere verwaltet die Objekte der aktuellen Umgebung. Im Gegensatz dazu verwaltet die langfristige Karte Objekte zur bisher detektierten Umgebung, die möglichst genau dargestellt sein soll. Beide Karten besitzen eine Obergrenze, sodass die Summe der beiden Obergrenzen die maximale Objektanzahl im Zustandsvektor definieren. Diese ist so gewählt, dass der UKF weiterhin echtzeitfähig bleibt.

Neben einer allgemeinen Verwaltung der Objekte kann die Rechenzeit durch Transformation der Objekte von der Inverse Depth Repräsentation in kartesische Koordinaten verbessert werden. Das ermöglicht entweder eine geringere Rechenzeit des UKF oder mehr Objekte zu initialisieren bei konstanter Rechenzeit.

4.4.1 Kurzfristige Karte

Die kurzfristige Karte enthält die Objekte der aktuellen Umgebung, deren Initialisierung nur wenige Zeitschritte her ist. Grundsätzlich werden alle neuen Objekte zunächst der

kurzfristigen Karte zugeordnet. Die Karte besitzt zudem eine Obergrenze an Objekten, die ihr zugeordnet sein dürfen. Sobald diese überschritten wird, findet eine Aktualisierung statt, sodass Objekte entfernt werden.
Zudem wird der Karte ein Zeitintervall zugeordnet. Damit wird eine regelmäßige Aktualisierung der Karte gewährleistet, wobei diese nur stattfinden kann, wenn neue Sensordaten existieren. Andernfalls entstünde ein Informationsverlust ohne Vorteil hinsichtlich der Laufzeit, da die maximale Objektanzahl entsprechend gewählt wurde.

Die Objektentfernung aus der kurzfristigen Karte erfolgt analog zu dem Algorithmus von Dissanayake, Williams, Durrant-Whyte und Bailey [9]. Hierbei werden alle Objekte gesammelt, die seit der letzten Aktualisierung der Karte nicht mehr detektiert wurden. Aus diesen Objekten wird eine bestimme Menge ausgewählt, die erhalten bleibt. Damit können Objekte erhalten bleiben, die sich zum Zeitpunkt gerade eben außerhalb des sichtbaren Bereichs befinden und im nächsten Schritt wieder sichtbar sind. Alle anderen werden von der Karte entfernt. Sichtbare Objekte werden dabei nur verwendet, wenn nicht genug Objekte entfernt werden würden.
Die Auswahl erfolgt über den Informationsgehalt der Objekte, der über die Kovarianzmatrix bestimmt wird. Zur Bestimmung des Informationsgehalts gibt es verschiedene Möglichkeiten[8]. Generell werden die Fisher-Information [11], die Entropie [11, 54] oder die Inverse der Spur [18] verwendet. Dissanayake, Durrant-Whyte und Bailey [8, S. 1013] erklärt, dass der Algorithmus im Allgemeinen unabhängig von der Wahl zur Berechnung des Informationsgehalts ist. Infolgedessen wird die Wahl durch die Rechnekomplexität der einzelnen Verfahren bestimmt. Tab. 4.1 illustriert drei Möglichkeiten. Die Bestimmung des Informationsgehalts über die Entropie ist aufgrund der Rechenkomplexität zur Berechnung der Determinanten erheblich höher. Fisher Information und inverse Spur unterscheiden sich nur geringfügig voneinander, wobei letztere eine geringere Komplexität aufweist. Dementsprechend wird die inverse Spur zur Bestimmung des Informationsgehalts verwendet.

Tabelle 4.1: Vergleich der Informations berechnung

	Fisher	Entropie (Shannon)	Inverse Spur
Formel	$\sum_i^n \sigma_{ii}^{-1}$	$ln(\det(P)^{-1})$	$(\sum_i^n \sigma_{ii})^{-1}$
Komplexität	$O(2n)$	$O(n^{2,373})$	$O(n+1)$

Eine Aktualisierung erfolgt immer nach der Initialisierung, damit neue Objekte mit in die Betrachtung des Informationsgehalts einbezogen werden. Von diesen werden die mit dem geringsten Informationsgehalt von der Karte entfernt, bis die maximale Grenze erreicht wurde. Dabei können sowohl neue als auch ältere Objekte entfernt werden. Ältere Objekte werden dann entfernt, wenn ein neues Objekt durch Tiefeninformationen eine bessere Schätzung darstellt, als ein Objekt, das nur einmal detektiert wurde. Dadurch erhält die Karte die zu diesem Zeitpunkt genauste Schätzung.

4.4.2 Langfristige Karte

Die langfristige Karte enthält Objekte, die durch den UKF bereits verbessert wurden. Dementsprechend ist die repräsentierte Karte akkurater als die kurzfristige Karte. Diese Karte wird in einem längeren Intervall aktualisiert. Dabei wird bei jeder Aktualisierung ein Objekt von der kurzfristigen in die langfristige Karte verschoben. Ziel dieser Karte ist, bei einer Rückkehr die Objekte wiederzuerkennen und einen Loop Closure zu ermöglichen, um die Schätzung zu verbessern.

Die Auswahl der Objekte erfolgt über den Informationswert für die gesamte Karte. Das beinhaltet einerseits den in Abschnitt 4.4.1 beschrieben Informationsgehalt und andererseits die Position des Objekt im Verhältnis zu den anderen Objekten. Der Bewertungsansatz erfolgt über einen Clustering-Ansatz, beschrieben von Hochdorfer und Schlegel [18].
Als Clustering Algorithmus wird, wie von Hochdorfer und Schlegel [18] beschrieben, DBSCAN, verwendet [41]. Hier werden in einer Umgebung *Eps* um einen Datenpunkt weitere Datenpunkte gesucht, die zu einem Cluster zusammen gehören. Datenpunkte außerhalb dieser Umgebung werden als Ausreißer deklariert und werden damit keinem Cluster zugeordnet. Mit dem Parameter $MinPts$ wird die minimale Anzahl an Datenpunkten, die zu einem Cluster gehören, festgelegt. DBSCAN bildet Cluster in Regionen mit einer hohen Datenpunktdichte, die getrennt sind durch Regionen geringer Dichte und benötigt dementsprechend keine Cluster Anzahl im Vorhinein. Um die maximale Information zu erhalten, sind Datenpunkte in weniger dichten Regionen bzw. Ausreißer von Interesse, da diese die Region alleine repräsentieren.
Im Anschluss des Clusterns ist zunächst für jedes Objekt der Informationsgehalt zu bestimmen. Darauffolgend wird für jedes Objekt die Position berechnet, um das Objekt zu clustern. Jedem Cluster C_i wird ein Informationswert nach Gl. 4.20 zugewiesen. Dabei wird die Differenz zwischen dem maximalen und minimalen Informationsgehalt aller dem Cluster zugeordneten Objekte m gebildet [17, 40].

$$D(C_i) = \max_{j=0...m}(I_j) - \min_{j=0...m}(I_j) \tag{4.20}$$

Die Cluster mit der größten Differenz enthalten immer mindestens zwei Objekte [40]. Damit ist die Region des Clusters auf der Karte besser repräsentiert als mit einer Differenz von null. Diese impliziert, dass nur ein Objekt vorhanden ist und die Region alleine repräsentiert wird. Dementsprechend ist der Informationswert dieser Objekte höher und trägt einen größeren Beitrag zur Erzeugung der Karte bei. Falls kein Cluster ausschließlich aus einem Objekt besteht, wird der Cluster mit der minimalen Differenz verwendet, da diese eine gleichmäßigere Verbesserung aller Objekte induziert. Demzufolge repräsentiert das Objekt eine Region, die weitestgehend bekannt ist. Aus diesem Cluster wird anschließend

das Objekt mit dem höchsten Informationswert in die langfristige Karte verschoben.
Die langfristige Karte hat, wie die kurzfristige Karte, eine Obergrenze hinsichtlich der Anzahl der Objekte. Bei Überschreitung dieses Limits wird der Informationsgehalt aller in der langfristigen Karte befindlichen Objekte bestimmt und die mit dem geringesten Wert der Karte und aus dem Zustandsvektor entfernt.

4.4.3 Transformation in kartesische Koordinaten

Neben der Möglichkeit einer Kartenverwaltung kann die Größe durch Transformation der Objekte in Inverse Depth Parametrisierung in kartesische Repräsentation der Zustandsvektor verkleinert werden. Diese Möglichkeit bietet den Vorteil der Erhaltung der bisher gesammelten Informationen zum Objekt.
Die Transformation ist nichtlinear und ist dementsprechend mit einer UT durchzuführen. J. Civera, A. J. Davison und J. M. M. Montiel [21] verwenden einen EKF und transformieren einzelne Objekte nacheinander unter Verwendung einer Jacobimatrix nach Gl. 4.21 für das entsprechende Objekt.

$$\mathbf{J} = \begin{bmatrix} \mathbf{I} & 0 & 0 \\ 0 & \frac{d\mathbf{y}}{d\mathbf{x}} & 0 \\ 0 & 0 & \mathbf{I} \end{bmatrix} \tag{4.21}$$

Diese wird anschließend mit der Kovarianzmatrix nach Gl. 4.22 multipliziert, um die aktualisierte Kovarianzmatrix zu erhalten. Der Zustandsvektor wird entsprechend direkt angepasst.

$$\mathbf{P}^{new} = \mathbf{JPJ}^T \tag{4.22}$$

Damit ist zum einen für jedes Objekt einzeln eine Jacobimatrix zu bestimmen. Die Bestimmung wiederum ist, wie beschrieben, fehleranfällig. Weiterhin findet jede Transformation einzeln statt. Dadurch finden für jedes transformierbare Objekt zwei Matrixmultiplikationen statt. Das erhöht die Komplexität erheblich. Die Verwendung der UT ermöglicht eine Reduktion der Komplexität durch die Verwendung einer gesamten Transformation für alle Objekte oder einer einzelnen Transformation unter Verwendung des zum Objekt gehörenden Teils des Zustands und der Kovarianzmatrix.
Der Vorteil einer Transformation einzelner Objekte bildet die Reduktion der Rechenkomplexität auf ein Minimum. Dabei wird nur der zum Objekt zugehörige Teil des Zustandsvektors und der Kovarianzmatrix verwendet. Allerdings werden damit alle Kreuzkorrelationen in der Kovarianzmatrix ignoriert. Dementsprechend sind alle Kreuzvarianzen zwischen dem transformierten Objekt und den anderen Objekten sowie dem Zustand auf null zu setzten, mit der Folge eines Informationsverlusts.
Die Transformation mit dem gesamten Zustand und damit aller Objekte gleichzeitig erhält alle Informationen auf Kosten einer höheren Rechenkomplexität. Allerdings kann der Informationsverlust durch eine einzelne Transformation zur Instabilität des UKF führen, da

nie eine vollständige Unabhängigkeit zwischen dem transformierten Objekt, dem Zustand und den restlichen Objekten besteht. Das ist insbesondere durch die Inverse Depth Parametrisierung nicht gegebenen, da diese maßgeblich auf der Schätzung der eigenen Position beruht. Demnach wird die Transformation mit dem gesamten Zustand gewählt.
Aufgrund der Verwendung des gesamten Zustandes ist es möglich, die entsprechenden Objekte gleichzeigt zu transformieren. Hierfür ist zunächst für alle Objekte der Linearitätsfaktor nach Gl. 3.21 zu berechnen. Eine Transformation ohne die gaußschen Eigenschaften zu verlieren, ist möglich, wenn $L_d < 0.1$. Dementsprechend wird für den UT-Parameter β ein Wert von zwei angenommen [51].

5 Validierung

Die Validierung der in Abschnitt 4 beschriebene Modellierung gliedert sich in der Verwendung der Parametrisierung im Kontext von UKF-SLAM für UAVs, die korrekte Verwaltung der entstandenen Karte sowie die Verwendung der Parametrisierung in Kombination mit der Kartenverwaltung. Die Verwendung ist nur dann möglich, wenn detektierte Objekte konvergieren und zur Lokalisierung beitragen. Die Verwaltung der Karte hingegen muss eine Karte erzeugen, die möglichst viele konvergierte Objekte und gleichzeitig die Aufnahme neuer Objekte zulässt, um die neue Umgebung gut dartsellen zu können.

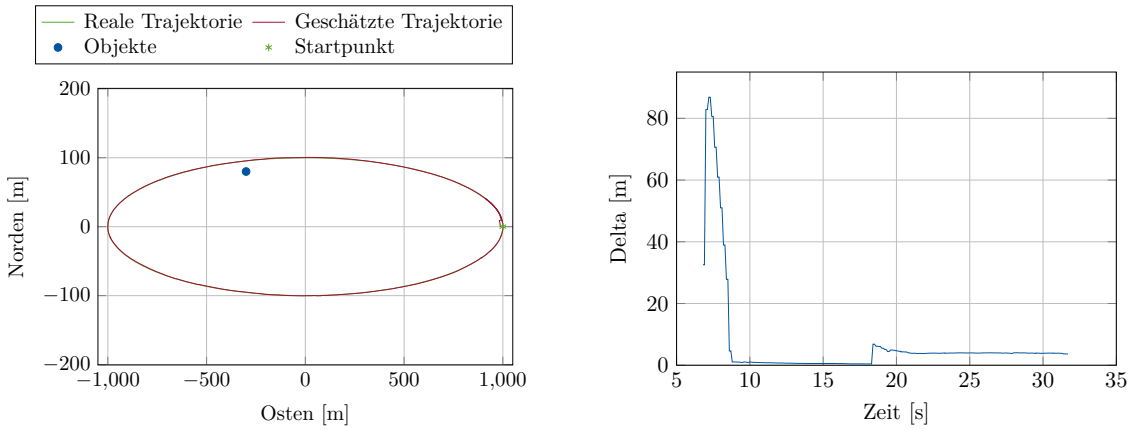
Die Implementierung erfolgte in C++ in das in Abschnitt 3.4 beschriebe Framework. Dieses Framework wird in eine Simulink® Simulationsumgebung eingebunden. Dort werden Routen und die entsprechenden Sensordaten generiert, die für den SLAM benötigt werden. Zur Beschreibung der Route werden Lissajous-Kurven in jede Koordinatenrichtung verwendet. Damit ergibt sich als einfachste Route ein Oval auf einer Ebene. In der Simulationsumgebung sind bereits Sensoren zur Bestimmung der Position in ECEF und Longitudinal, Lattitude und Altittude (LLA), der Geschwindigkeit in ECEF und NED sowie der aktuellen räumlichen Lage und ein Magnetometer implementiert. Neben diesen Sensoren ist ein Feature Emulator eingebunden. Dieser detektiert Objekte im Bild unter Verwendung einer Kamera, bietet aber auch die Möglichkeit, in der Simulationsumgebung Objekte unter Eingabe vorher generierter Objekte zu emulieren. Dementsprechend sind die Sensordaten für die optische Navigation in der Simulationsumgebung im gleichen Format wie die realen Sensordaten.

5.1 Verwendung der Inverse Depth Parametrisierung in SLAM

Die Verwendung der Inverse Depth Parametrisierung im SLAM für UAVs ist nur dann möglich, wenn aufgenommene Objekte konvergieren und im Falle von Sensorausfällen die Lokalisierung weiterhin ermöglichen. Zudem sollte ein Loop Closure die Genauigkeit der Schätzung erheblich verbessern. Die Validierung der Parametrisierung erfolgt inkrementell mit wachsender Komplexität. Zudem werden keine Tiefeninformationen zu den einzelnen Objekten verwendet, da diese durch zusätzliche Sensorik gebildet werden. Damit wird gewährleistet, dass sowohl die Bestimmung der Position als auch die Navigation nur mit

optischen Sensorwerten in Form von Kamerabildern erfolgt.
Dafür wird im ersten Schritt ein Objekt fest definiert, das sich möglichst lange im Blickfeld der Kamera befindet. Damit kann die Konvergenz des Objekts mit dem gewählten Messmodell eingeschätzt und die Konvergenzgeschwindigkeit abgeschätzt werden. Letztere muss entsprechenden hoch sein, um auf einen Sensorausfall zeitnah vorbereitet zu sein. Zudem bildet die Konvergenzgeschwindigkeit das Maß an welchen Zeitpunkt eine Lokalisierung ohne weitere Sensoren möglich ist.
Anschließend wurden mehrere Objekte entlang der Strecke definiert, um die Konvergenz mehrerer wohldefinierter Objekte zu prüfen. Zunächst wurden vier Objekte zentral gesetzt, sodass sich diese ebenso möglichst lange im Blickfeld befinden. Das ermöglicht eine Einflussabschätzung der Objekte untereinander im Hinblick auf die Konvergenz. Anschließend sind je Seite der Runde elf Objekte äquidistant platziert worden.
Mit dieser Konfiguration ist es bereits möglich, erfolgreich Objekte aufzunehmen und nach entsprechender Konvergenz zur Lokalisierung ohne weitere Sensordaten zu verwenden. Das ist möglich aufgrund der repräsentativen Objektwahl für die entsprechenden Flugabschnitte.
Als nächstes wird eine größere Menge Objekte zufällig in der Umgebung verteilt. Die Anzahl ist dabei so gewählt, dass die Zustandsvektorgröße eine laufzeitabhängige Maximalgröße nicht überschreitet. Dadurch ist keine Kartenverwaltung notwendig und die isolierte Betrachtung der Inverse Depth Parametrisierung möglich.

Die Abb. 5.1b zeigt die Abweichung der Positionsschätzung eines Objekts zur realen Position. Die Position des Objekts in Relation zur Trajektorie ist in Abb. 5.1a dargestellt. Das Objekt konvergiert nach der erstmaligen Detektion und Aufnahme in den Zustandsvektor innerhalb weniger Sekunden. Der Anstieg nach der ersten Detektion kann auf die Initialisierung zurückgeführt werden, da der gesetzte Tiefenwert zunächst keine physikalisch sinnvolle Größe besitzt.

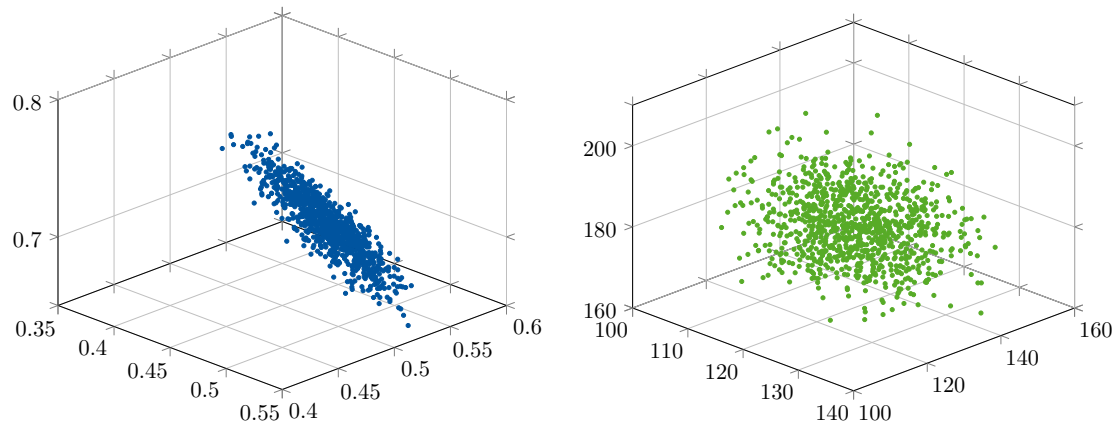


(a) Trajektorie mit einzelnem Objekt

(b) Zeitliche Verändung des Unterschieds zwischen Schätzung und realer Position

Abbildung 5.1: Verwendung eines einzelnen Objekts

Anschließend konvergiert das Objekt in weniger als 2 s auf einen Abstand von unter 1 m. Der Anstieg bei 18 s ist auf die erneute Detektion des Objekts und dem Messmodell zurückzuführen. Dieses Verhalten begründet sich in der Messung des Abstandsvektors zum Objekt als Projektion in das Kamerabild und der Transformationen in einen Richtungsvektor. Dieser wird durch Normalisieren des Abstandsvektors zum Objekt gebildet. Dadurch wird auch die Kovarianzmatrix der Messung normalisiert und die Sicherheit fälschlicherweise erhöht. Infolgedessen wird der Einfluss kleinerer Abweichungen größer. Allerdings zeigt Abb. 5.1b auch, dass die Erhöhung begrenzt ist und die Objekte anschließend erneut konvergieren. Abb. 5.2 zeigt die Verteilung eines normal verrauschten Abstandsvektors



(a) Normalisierter normal verrauschter Abstandsvektor

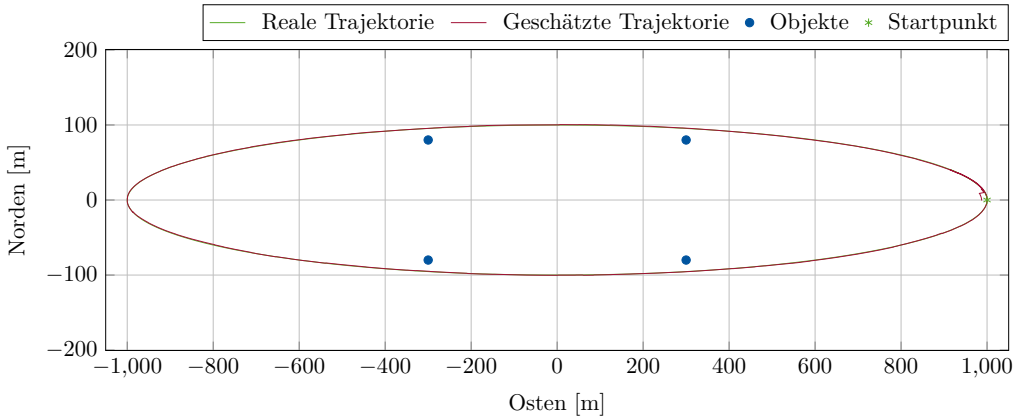
(b) Normal verrauschter Abstandsvektor

Abbildung 5.2: Veränderung der Verteilung durch Normalisierung des Abstandsvektors

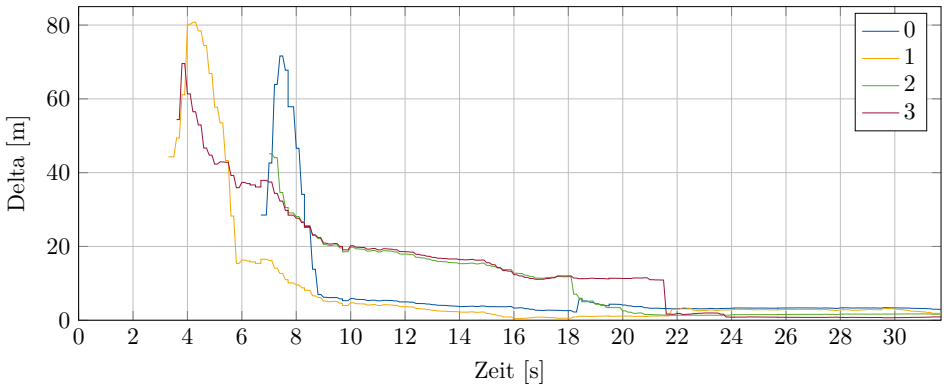
und der entsprechenden Normalisierung. Dabei ist deutlich zu erkennen, dass insbesonde-

re die Ungenauigkeit hinsichtlich der Tiefe stark gesunken ist. Zudem ist die Verteilung der einzelnen Punkte erkennbar geringer.

Abb. 5.3b zeigt die Konvergenz von vier fest definierten Objekten. Die vier Objekte relativ zur Trajektorie sind in Abb. 5.3a dargestellt. Die Objekte konvergieren ähnlich zum Fall mit einem Objekt. Allerdings reduziert sich die Konvergenzgeschwindigkeit bei der Initialisierung neuer Objekte, erkennbar bei der Aufnahme der Objekte 0 und 2. Die Konvergenzgeschwindigkeit der bereits initialisierten Objekte 1 und 3 stagniert, bis die neueren Objekte konvergieren.



(a) Trajektorie mit vier fest definierten Objekten Objekt



(b) Zeitliche Verändung des Unterschieds zwischen Schätzung und realer Position

Abbildung 5.3: Verwendung von vier festen Objekten

Die starken Verbesserungen der Objekte 2 und 3 bei 20 s wird durch die erneute Detektion hervorgerufen und beschreibt damit einen erfolgreichen Loop-Closure.

Abb. 5.4 zeigt die Positionierung der 22 Objekte entlang der Flugtrecke. Die Positionierung ist so gewählt worden, dass zu jedem Zeitpunkt ein Objekt detektiert werden kann. Damit sind die einzelnen Abschnitte der Route durch ein Objekt repräsentiert.

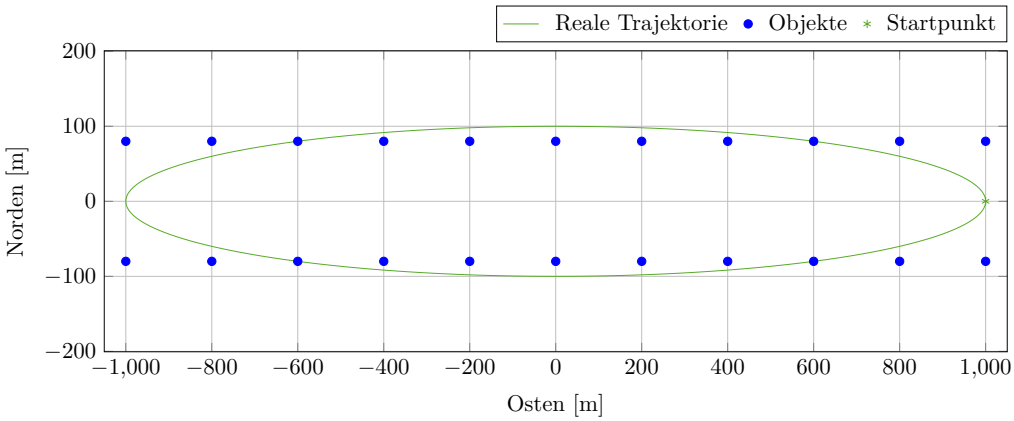


Abbildung 5.4: Trajektorie mit 22 Objekten entlang der Strecke

Abb. 5.5 zeigt die Konvergenz aller 22 Objekte, wenn kein Sensor ausfällt. Die bereits beschriebenen Phänomene tauchen auch in diesem Fall auf. Insbesondere die Erhöhung der Differenz zur realen Position, hervorgerufen durch Abweichungen des Richtungsvektors bei der erneuten Detektion, ist zu erkennen.

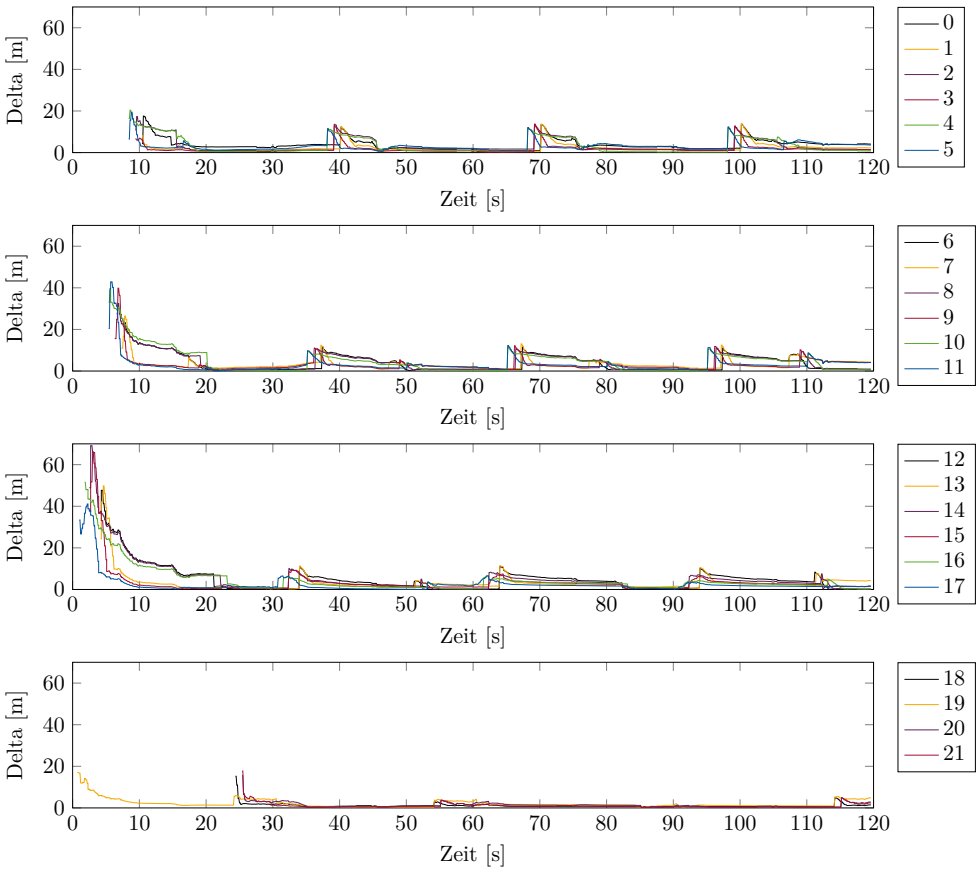


Abbildung 5.5: Zeitliche Verändung des Unterschieds zwischen Schätzung und realer Position der 22 Objekte

Abb. 5.6 zeigt die Abweichung der Objekte zur realen Position mit Ausfall der Sensorik nach 50 s bei einer Gesamtflugzeit von 120 s.
Dabei bleibt die Abweichung zur realen Lage des Objekts relativ gering. Allerdings ist auch an den Objekten 0 bis 5 zu erkennen, dass ohne weitere Sensorik eine Divergenz von weniger konvergierten Objekten auftritt. Dieses Phänomen ist unabhängig vom Ausfallzeitpunkt zu erkennen. Im Gegensatz dazu bleiben gut konvergierte Objekte von einem Sensorausfall unberührt.

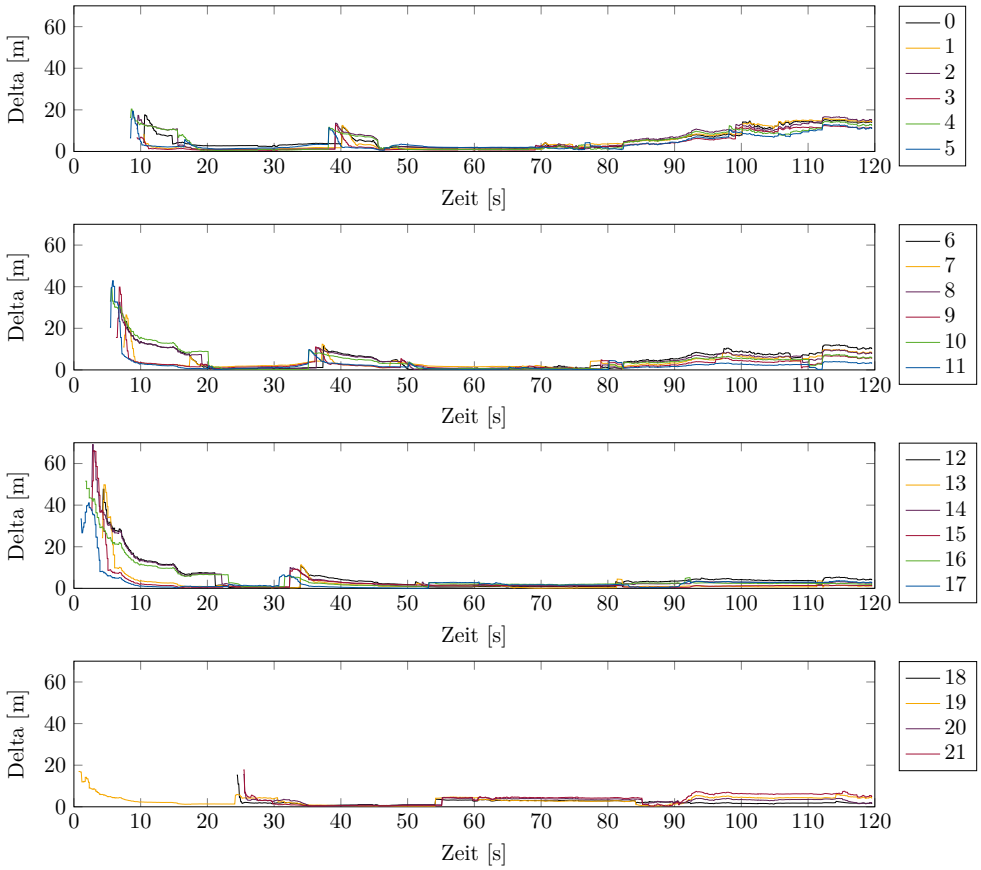


Abbildung 5.6: Konvergenz der geschätzten Objekte bei einem Sensorausfall nach 50 s

Abbildung 5.7 zeigt den geschätzten Fehler bei dem Sensorausfall nach 50 s. An der Abbildung ist einerseits der geringe Einfluss neuer Objekte auf die Position erkennbar. Andererseits ist die Lokalisierung bei einem Sensorausfall weiterhin erfolgreich unter der Bedingung genügend konvergierter Objekte. Insbesondere der Zeitpunkt des Sensorausfalls hat einen geringen Einfluss auf die Lokalisierung.

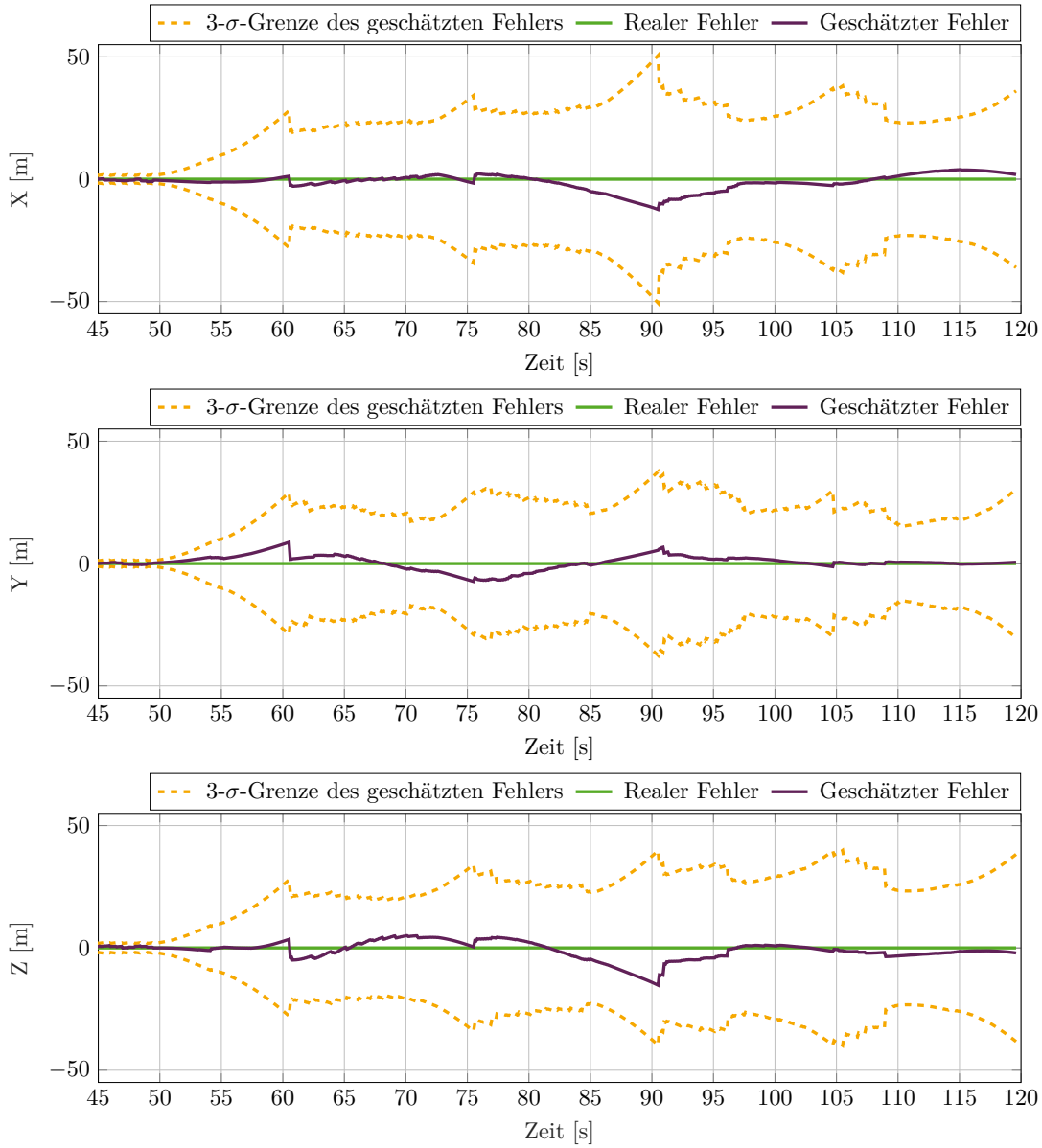


Abbildung 5.7: Positionsfehler bei Sensorausfall nach 50 s

Es treten lediglich geringe Abweichungen hinsichtlich der Z-Koordinate auf, wobei sich die Schätzung der Position jedoch wieder verbessert. Das ist auf die konvergierten Objekte zurückzuführen, die in dieser Region erneut zur Schätzung beitragen.

Abb. 5.8 zeigt eine zufällige Verteilung von 106 Objekten in der Umgebung der simulierten Route. Abb. 5.9 beschreibt die in den Zustandsvektor aufgenommen Objekte. Die Anzahl der aufgenommenen Objekte ist geringer als die Gesamtanzahl, da nicht alle Objekte während der Simulation detektiert wurden.

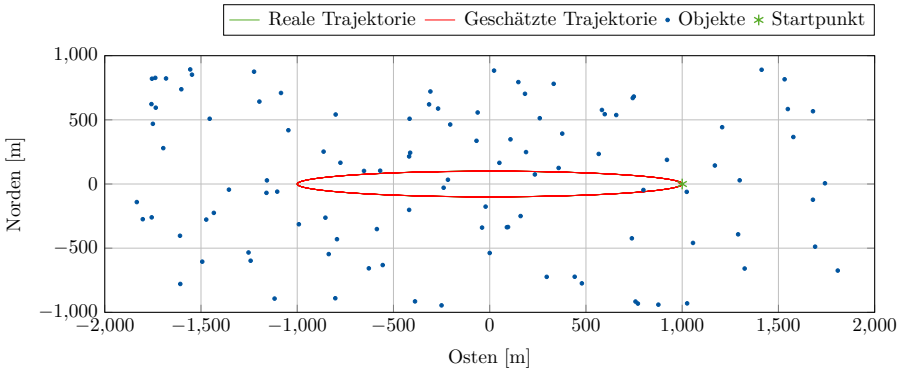


Abbildung 5.8: Position zufällig erzeugter Objekte mit Trajektorie

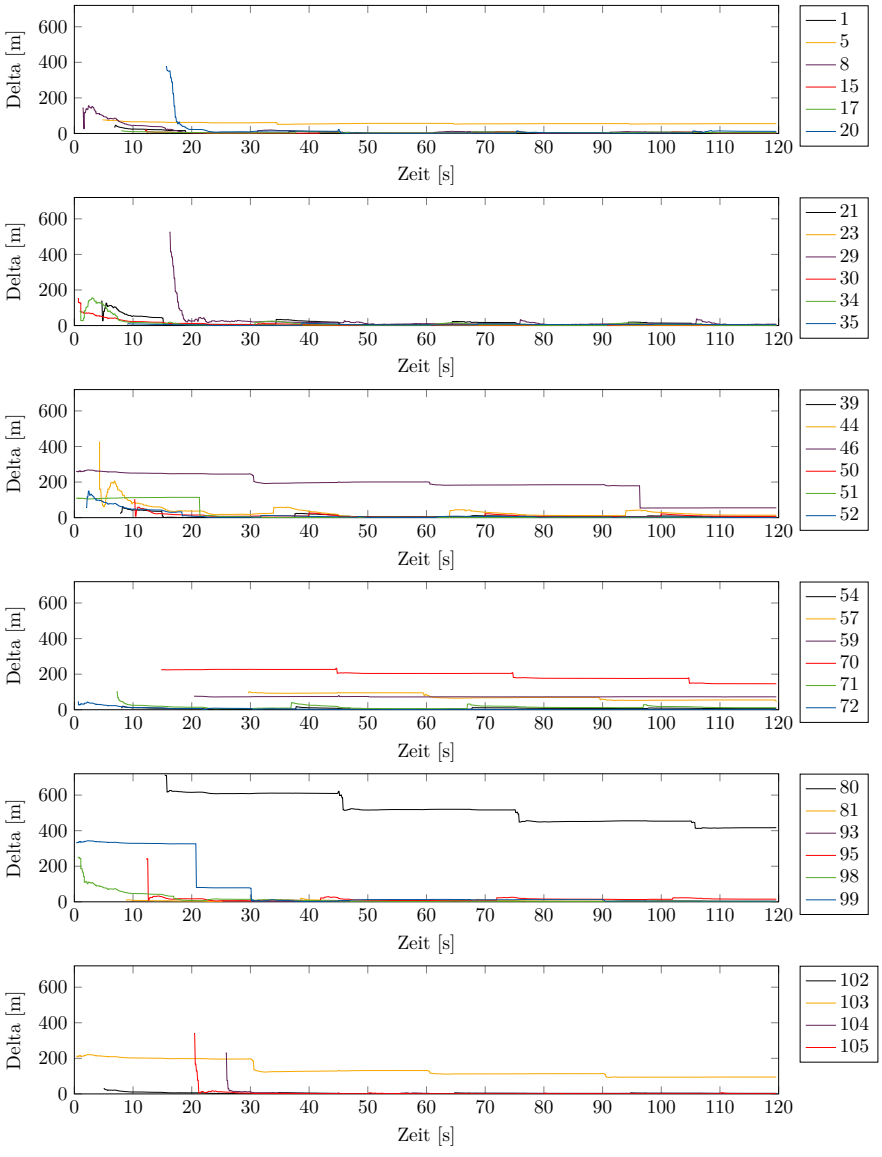


Abbildung 5.9: Konvergenz der aufgenommenen Objekte bei zufällig generierten Punkten

Dementsprechend wurden in diesem Fall lediglich 34 Objekte aufgenommen. Auch hier ist zu erkennen, dass die meisten Objekte zeitnah konvergieren. Allerdings existieren einige Objekte, die nach 120 s noch nicht konvergiert sind. Diese Objekte befinden sich immer nur kurzzeitig im Blickfeld der Kamera, sodass sich die Konvergenzgeschwindigkeit dieser Objekte reduziert. Das Objekt 80 zeigt deutlich, dass diese wenige Male pro Runde detektiert und dabei jedes Mal die Schätzung verbessert wird.

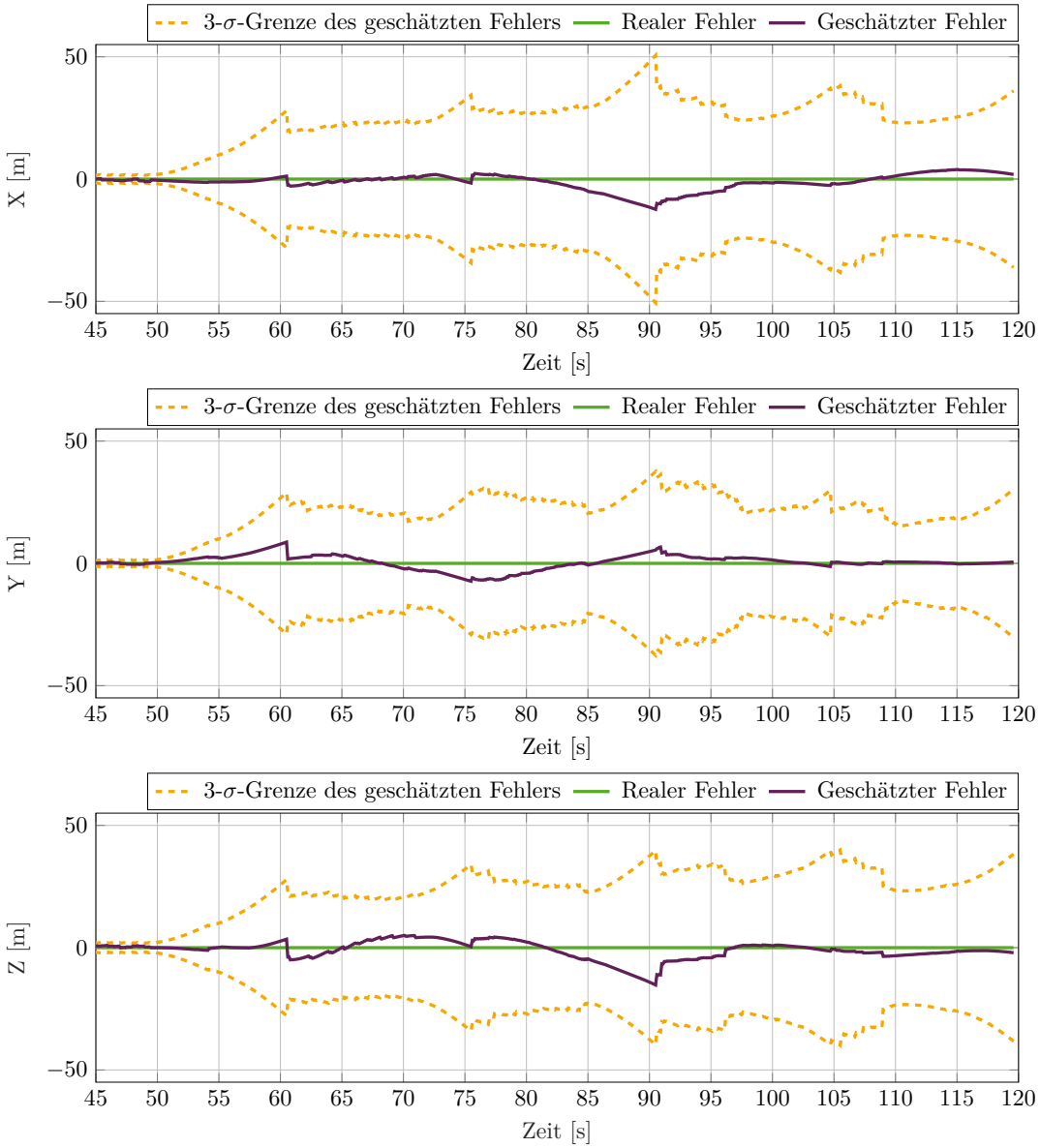


Abbildung 5.10: Positionsfehler bei zufällig generierten Objekten

Analog zu den 22 fest definierten Objekte wird ein Sensorausfall nach 50 s simuliert, dargestellt in Abb. 5.10 . Dieser zeigt die zuvor festgestellte Charakteristik. Insbesondere die Abhängigkeit der Lokalisierung ohne zusätzliche Sensorik von der Konvergenz der Objekte ist auffällig. Der Sensorausfall nach 90 s hat bei 106 s eine größere Abweichung der Z-Koordinate zur Folge, wird aber anschließend wieder durch besser konvergierte Objek-

te korrigiert. Im Gegensatz dazu führt der Sensorausfall nach 50 s zu einer kleinen, aber regelmäßigen Verschiebung der Z-Koordinate. Auch das kann auf nicht ausreichend konvergierte Objekte zurückgeführt werden.
Bei einer ausschließlich optischen Navigation ist einerseits auffällig, dass diese trotz einiger nicht genügend konvergierten Objekte mit geringer Abweichung weiterhin erfolgreich ist. Andererseits zeigt Abb. 5.11, dass weniger gut konvergierte Objekte weiterhin konvergieren können unter Verwendung bereits konvergierter Objekte. Dementsprechend ist SLAM mit Inverse Depth Parametrisierung robust gegenüber Sensorausfällen und der Verwendung von Objekten, die nicht im optimalen Blickfeld liegen.

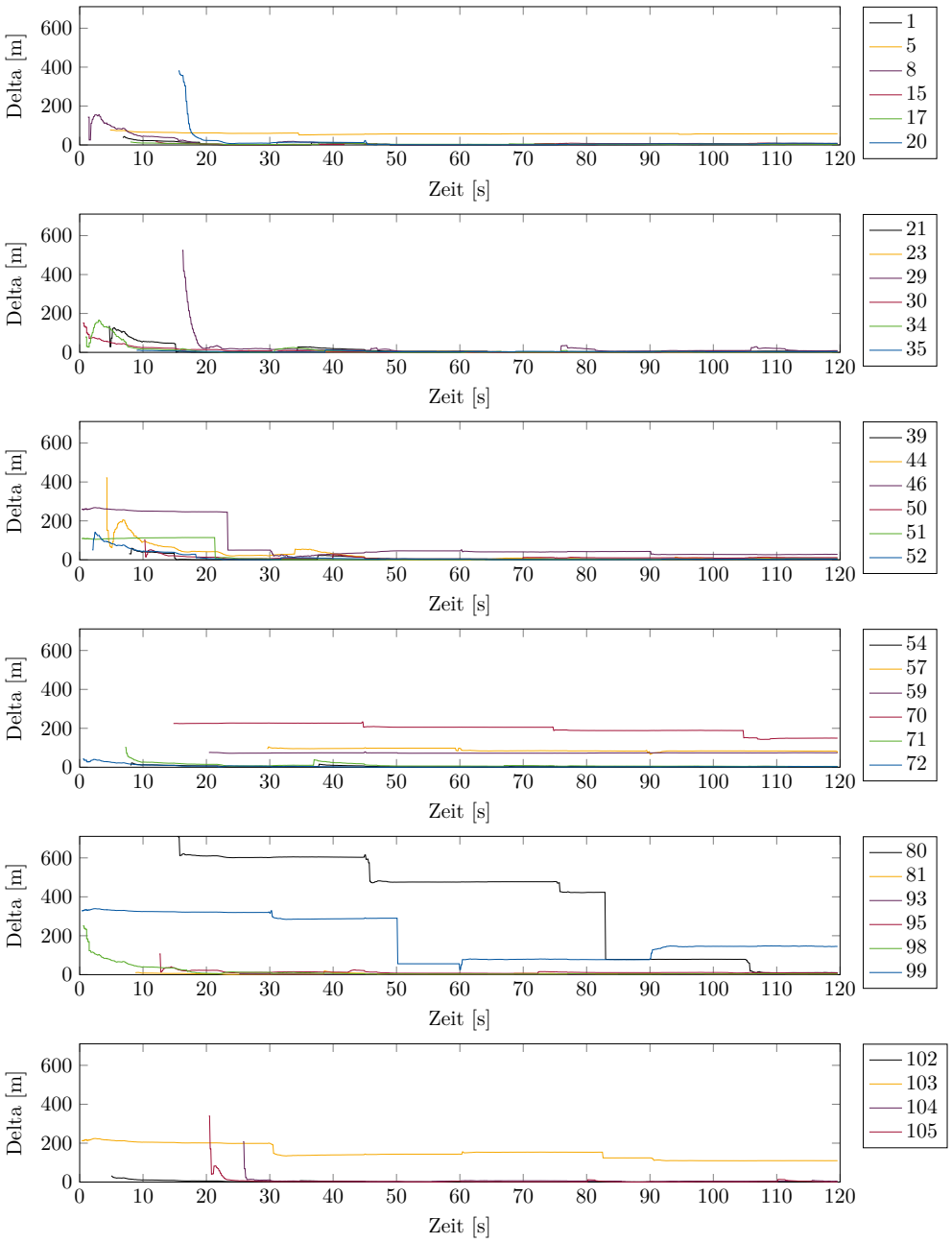


Abbildung 5.11: Konvergenz von Objekten mit Sensorausfall

5.2 Verwaltung der Karte

Die in Absatz 4.4 beschriebe Kartenverwaltung wird in die zwei Abschnitte der kurzfristigen und langfristigen Kartenverwaltung unterteilt. Dabei repräsentiert die kurzfristige Karte die lokale und die langfristige Karte die gesamte bisher detektierte Umgebung.
Die Funktionalität des in der kurzfristigen Karte verwendeten Algorithmus ist bereits von Dissanayake, Williams, Durrant-Whyte und Bailey [9] gezeigt worden. Weiterhin ist die Verwendung eines Clustering-Algorithmus zur Ermittlung des Wertes in der Karte unabhängig von der tatsächlichen Genauigkeit von Hochdorfer und Schlegel [18] gezeigt worden. Im Gegensatz zu der beschriebenen Methodik wird hier die Information genutzt, um ein Objekt zu finden, das die Region am besten repräsentiert. Dementsprechend besteht der Fokus hinsichtlich des Übertrags in die langfristige Karte sowie dem Entfernen von Objekten.
Aus Gründen der Übersicht ist die Menge der zu detektierenden Objekte auf 30 begrenzt worden. Von diesen sind sechs im Vorhinein festgelegt. Die übrigen Objekte sind zufällig gesetzt worden.
Die sechs Objekte mit den Id 0 bis 5 befinden sich im Rechteck um den Mittelpunkt bei $(\pm 80, \pm 300, -145)$, an den Scheitelpunkten der Ellipse bei $(0, \pm 950, -145)$ sowie im Mittelpunkt bei (0,0,0). Diese Objekte stehen repräsentativ für die lokale Umgebung. Dementsprechend werden diese früh in die langfristige Karte geschoben. Die gesamte Verteilung ist in Abb. 5.12 dargestellt.

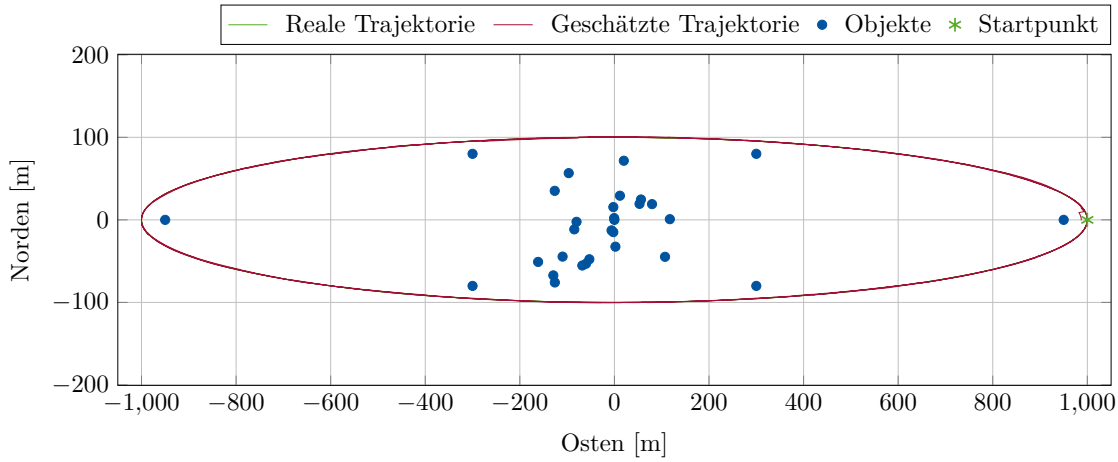
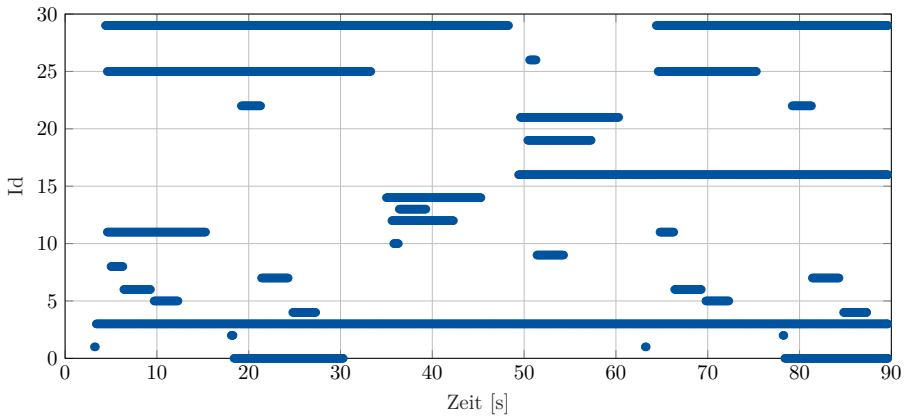
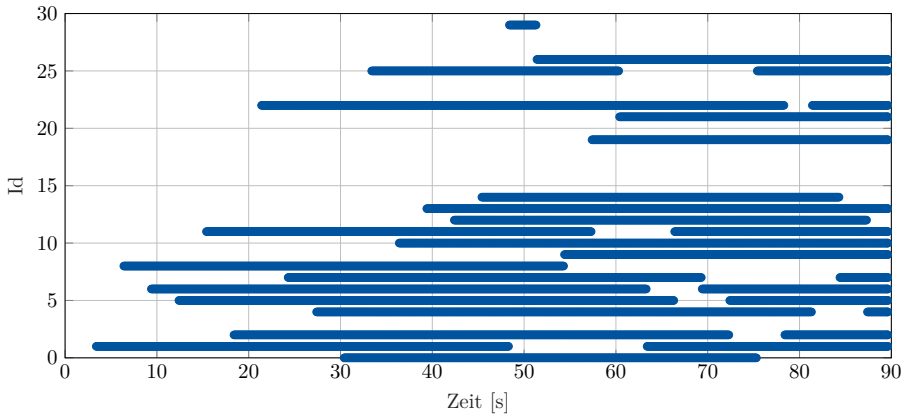


Abbildung 5.12: Verteilung der Objekte mit sechs fest definierten Objekten

Abb. 5.13 zeigt den zeitlichen Verlauf der Objekte in der kurzfristigen Karte (oben) und der langfristigen Karte (unten). Die Größe der einzelnen Karten ist dabei auf 5 für die kurzfristige und 15 für die langfristige Karte gesetzt worden. Damit wird gewährleistet, dass zu keinem Zeitpunkt alle Objekte gleichzeitig im Zustandsvektor sein können und so ein regelmäßiger Austausch notwendig ist. Zudem ist der Zyklus für die kurzfristige Karte auf 2 s und für die langfristige Karte auf 3 s gesetzt worden.



(a) Objekte in der kurzfristigen Karte



(b) Objekte in der langfristigen Karte

Abbildung 5.13: Verteilung der Objekte af kurzfristige und langfristige Karte über die Zeit

Das Objekt mit der Id 5, also am Scheitelpunkt der Ellipse, wird initial in die kurzfristige Karte aufgenommen und bei der nächsten Möglichkeit direkt in die langfristige Karte überführt, da diese alleine repräsentativ für die gesamte Umgebung steht. Analog wird das gegenüberliegende Objekt mit der Id 4 zu einem späteren Zeitpunkt direkt überführt. Neben dem Übertrag in die langfristige Karte ist zu erkennen, dass die kurzfristige Karte regelmäßig aktualisiert wird. Allerdings werden nur die am wenigsten konvergierten Objekte entfernt. Deutlich wird das durch das Objekt mit der Id 3. Dieses Objekt befindet sich nach der Detektion dauerhaft in der kurzfristigen Karte. Demgegenüber existieren Objekte, die nur kurz in der kurzfristigen Karte auftauchen. Diese konvergieren entweder nicht schnell genug oder bilden einen signifikanten Bereich der gesamten Umgebung ab.
Die Abb. 5.14 stellt eine Momentaufnahme während des Fluges dar. Die zurückgelegte Strecke zu diesem Zeitpunkt ist durch die grüne Linie und die aktuelle Position durch den roten Punkt gekennzeichnet. Der Großteil der Objekte in der kurzfristigen Karte befindet sich vor dem UAV. Die Objekte hinter dem UAV sind entsprechend gut konvergiert, sodass diese einen signifikanten Beitrag zur Schätzung leisten können. Weiterhin ist zu erkennen,

dass die Objekte der langfristigen Karte jeweils den zurückgelegten Bereich abbilden und dabei jeweils für einen Abschnitt der Strecke stehen.

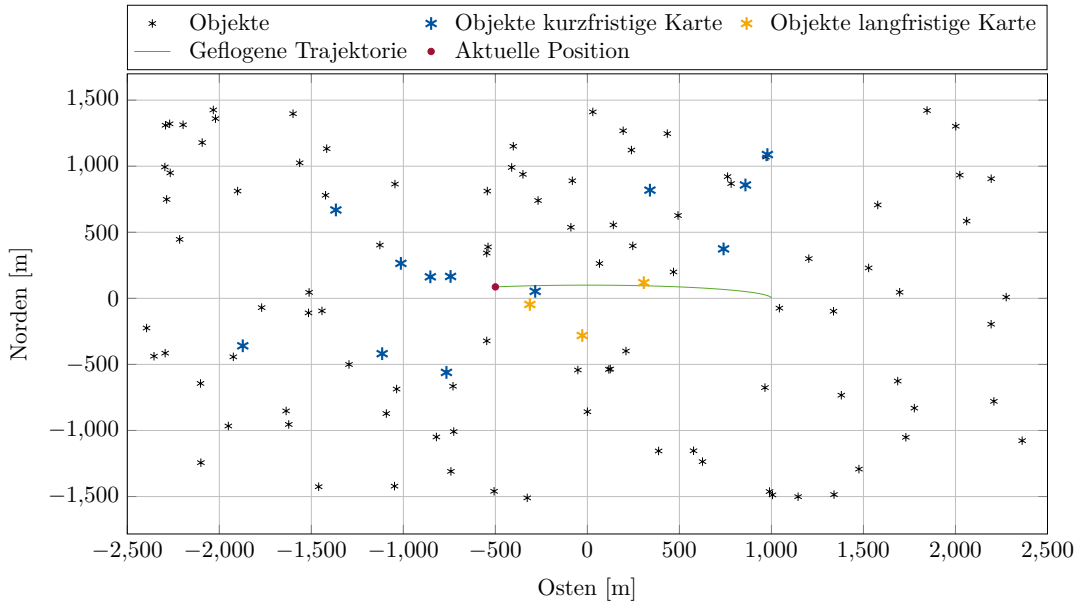
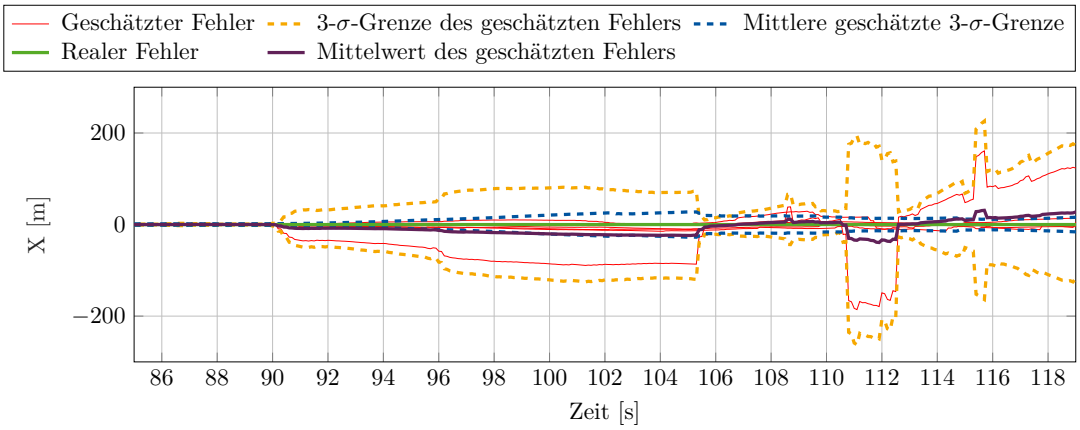


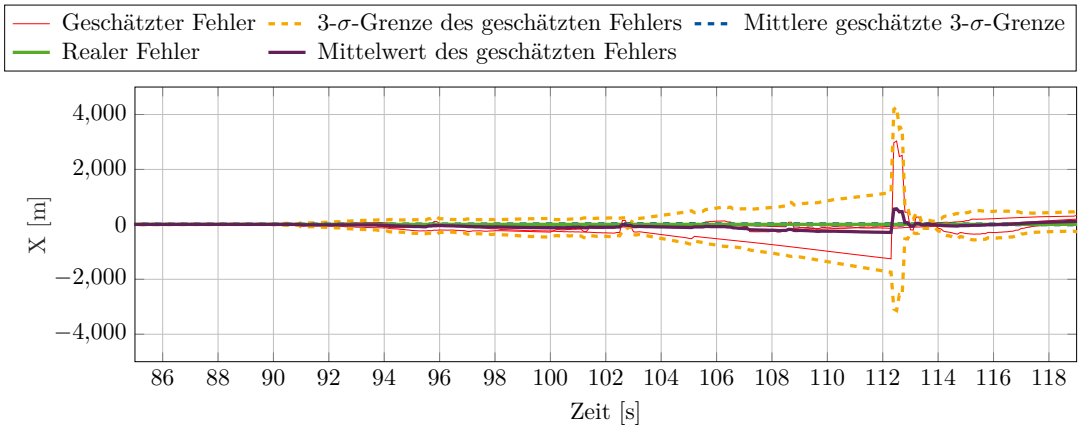
Abbildung 5.14: Aufgenommene Objekte getrennt in unterschiedliche Karten

5.3 Inverse Depth SLAM mit Kartenverwaltung

Nachdem in Abschnitt 5.1 und 5.2 eine isolierte Betrachtung erfolgte, wird im folgenden die Verwendung der Inverse Depth Parametrisierung im SLAM mit der Kartenverwaltung demonstriert. Dementsprechend wird eine entsprechende Anzahl Objekte erzeugt, die zur Orientierung genutzt werden können.
Zudem werden mehrere Simulationen mit einer Konfiguration verwendet, wobei sich in jeder Simulation die Position und Sichtbarkeit der Objekte verändert. Analog zu Abschnitt 5.1 wird ein Sensorausfall nach 50 s und 90 s simuliert. Weiterhin wird neben einer rein optischen Navigation auch der Ausfall von Satellitennavigationssystemen simuliert, indem weiterhin Sensordaten über die Lage mithilfe eines Magnetometers zur Verfügung stehen. Zur Überprüfung der Robustheit, werden Monte-Carlo-Simulation durchgeführt.
Als Referenzgrößen werden der geschätzte Fehler sowie die reale Position im ECEF-Koordinatensystem der X-Koordinate verwendet. Die zugehörigen Abbildung für die Y- und Z-Koordinate sind in in Anhang A dargestellt.
Abb. 5.15 zeigt den Fehler der Position nach 90 s mit und ohne Magnetometer. Dabei beschreiben die dickeren Linien den Mittelwert aller Simulationen und die dünneren Linien die einzelnen Simulationen. In beiden Fällen steigt erwartungsgemäß die Ungenauigkeit der Position, zu erkennen an der 3-σ-Grenze. Dabei steigt das Intervall ohne Magnetometers deutlich stärker an. Das spiegelt sich insbesondere an der Größenordnung der Skala wieder.



(a) Positionsfehler mit Magnetometers

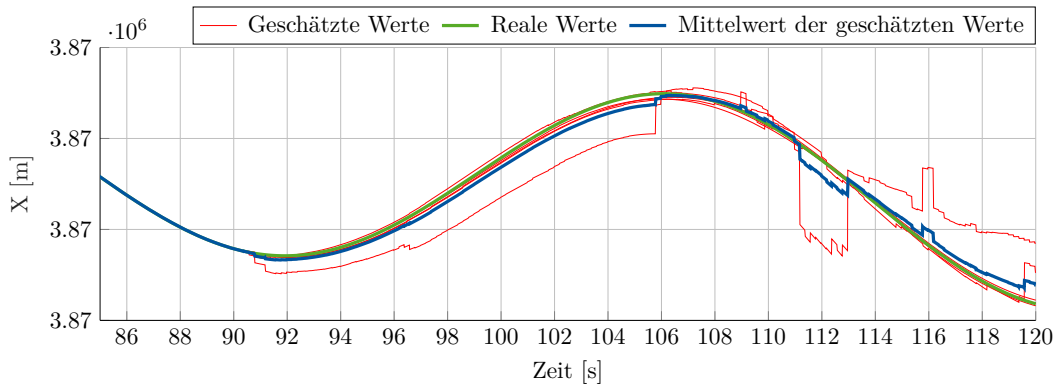


(b) Positionsfehler ohne Magnetometers

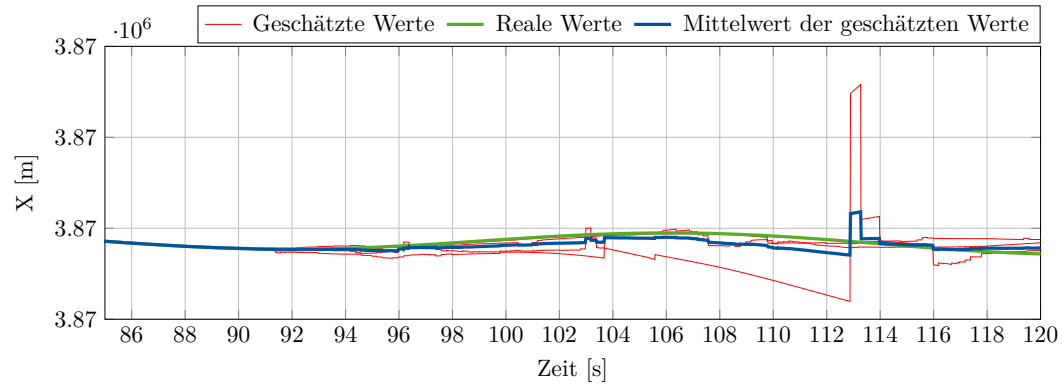
Abbildung 5.15: Positionsfehler bei Sensorausfall nach 90 s

Allerdings ist zu erkennen, dass unabhängig vom Sensorausfall der gemittelte geschätzte Fehler nah am echten Fehler liegt.

Dementsprechend verbessert die Verwendung eines zusätzlichen Sensors die Lokalisierung erheblich.
Abbildung 5.16 zeigt das beschriebene Verhalten gleichermaßen. Zudem ist zu erkennen, dass auch ohne Magnetometer eine gute Lokalisierung stattfinden kann. Allerdings ist hier die Gefahr größer, dass die Messung des Richtungsvektors einen größeren Einfluss erhält. Dieser Einfluss ist an den Ausreißern in Abb. 5.16b zu erkennen. Die geschätzte Position in den weiteren Simulationen waren erfolgreicher, weswegen die mittlere Schätzung deutlich näher an der realen Position verläuft.



(a) Geschätzte Position mit Magnetometers

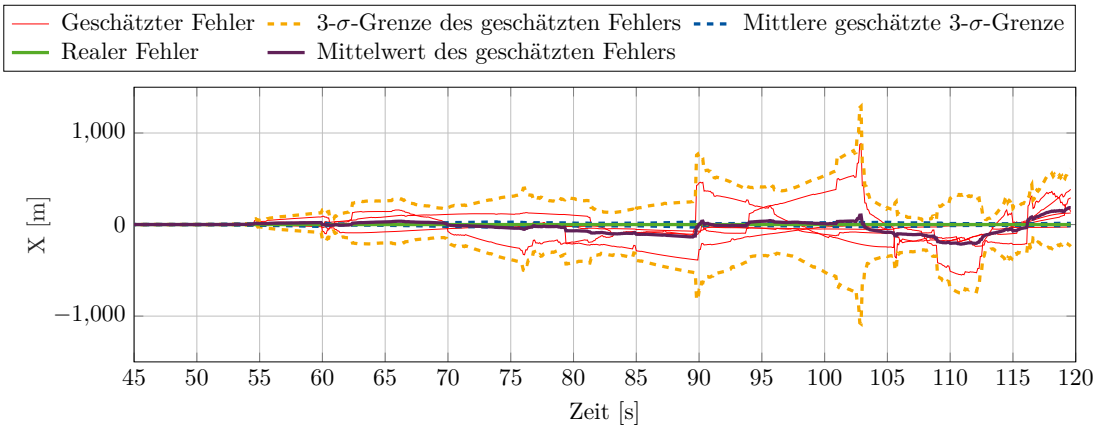


(b) Geschätzte Position ohne Magnetometers

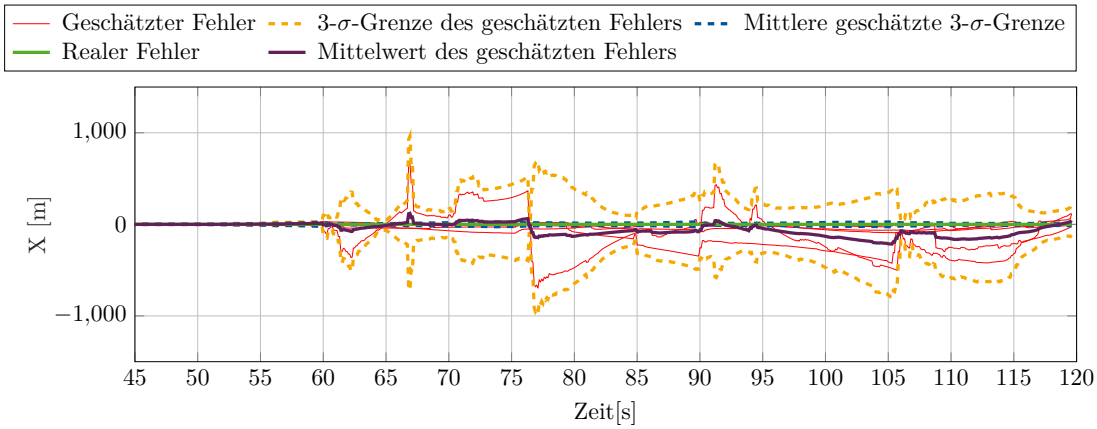
Abbildung 5.16: Geschätzte Position bei Sensorausfall nach 90 s

In Ab. 5.16a sind solche Ausreißer auch zu erkennen. Diese werden allerdings durch die Verwendung des Magnetometers gedämpft, sodass die Schätzung sowohl in den einzelnen Simulationen als auch im Mittel deutlich besser wird.

Die Abb. 5.17 zeigt den geschätzten Positionsfehler bei einem Sensorausfall nach 50 s. Analog zu Abb. 5.15 ist nach dem Sensorausfall ein schneller Abfall der Genauigkeit zu erkennen. Doch trotz des früheren Sensorausfalls divergiert die Schätzung nicht. Insbesondere die mittlere Abweichung bleibt in beiden Fällen in der Nähe des tatsächlichen Werts, wobei die Verwendung des Magnetometers die Schwankung ebenso reduziert.



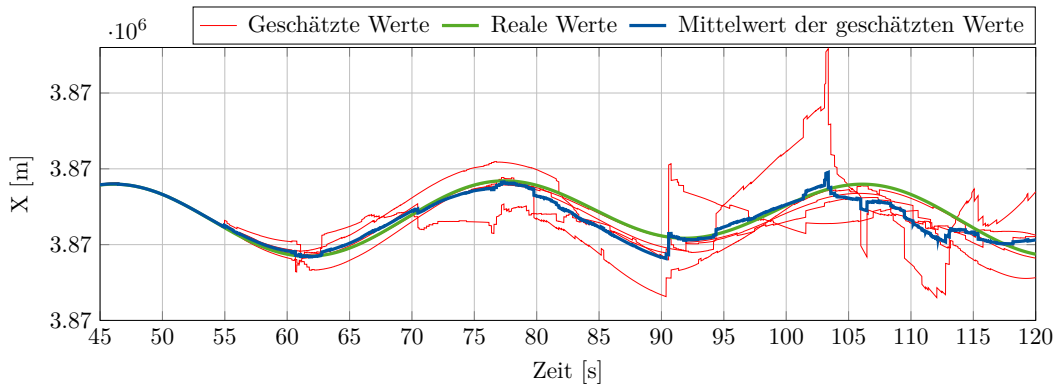
(a) Positionsfehler mit Magnetometers



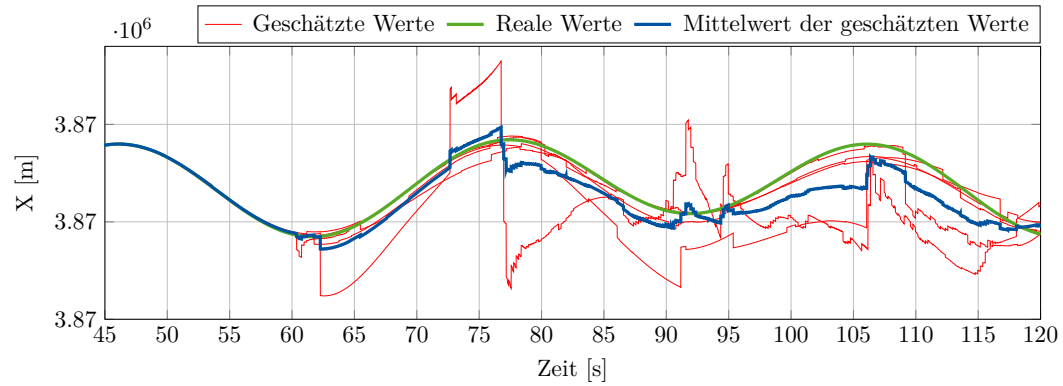
(b) Positionsfehler ohne Magnetometers

Abbildung 5.17: Positionsfehler bei Sensorausfall nach 50 s

Erneut ist eine Simulation zu erkennen, die deutlich größere Abweichungen hat, als die anderen und damit die mittlere Abweichung etwas verzerren. Der Mittelwert folgt sowohl mit als auch ohne Magnetometer der realen Position mit kleineren Abweichungen. Dabei hat das Magnetometer erneut einen dämpfenden Einfluss. Abb. 5.18a verdeutlicht diesen Zusammenhang. Inesbesondere die einzelnen Ausreißer sind deutlich zu erkennen. Allerdings werden diese kontinuierlich sowohl mit als auch ohne Magnetometer korrigiert. Dabei sind größere Ausschläge bei Einsatz des Magnetometers erst zu einem späten Zeitpunkt erkennbar. Zudem ist der Mittelwert in Abb. 5.18a insgesamt deutlich näher am realen Verlauf als in Abb. 5.18b.



(a) Geschätzte Position mit Magnetometers



(b) Geschätzte Position ohne Magnetometers

Abbildung 5.18: Geschätzte Position bei Sensorausfall nach 90 s

Die Validierung zeigt noch Instabilitäten hinsichtlich der Messung des Richtungsvektors auf. Durch die implizierte erhöhte Genauigkeit können einerseits bereits konvergierte Objekte divergieren. Andererseits können diese weniger gut konvergierten Objekte zu einer großen Schwankung oder auch zur Divergenz führen. Dieser Einfluss kann aber durch die Verwendung eines weiteren Sensors erheblich verbessert werden, sodass die Navigation weiterhin möglich bleibt.
Weiterhin geht aus der Validierung hervor, dass ein UKF-SLAM mit der Inverse Depth Parametrisierung möglich ist und ein UAV robuster gegenüber einem Sensorausfall von Satellitennavigationssystemen macht. Zudem ist die entwickelte Verwaltung in der Lage, eine Karte zu erzeugen, die sowohl eine langfristige Navigation als auch eine kurzfristige Orientierung ermöglicht.

6 Zusammenfassung und Ausblick

Die Motivation dieser Arbeit bestand in der Verbesserung der Robustheit von UAVs gegenüber Störungen von Satellitennavigationssystemen, um eine korrekte Navigation ohne externe Steuerung weiterhin zu gewährleisten. Insbesondere die Verwendung der optischen Navigation stand dabei im Fokus. Die Robustheit wird durch die Verwendung von SLAM gewährleistet, indem die Umgebung zur Positionsschätzung miteinbezogen wird.
In der Luftfahrt enthalten Sensordaten in Form von Kamerabildern weniger Informationen als am Boden aufgrund der Flughöhe. Insbesondere die im Kontext von SLAM detektierten Objekte können unter Umständen in größerer Entfernung sein. Als Lösungsansatz dieser Problematik wurde die Inverse Depth Parametrisierung verwendet. Diese bietet den Vorteil, Objekte in größerem Abstand schon früh zur Orientierung zu nutzen. Dabei wird ein Objekt durch die Position der ersten Detektion sowie einem Richtungsvektor und der inversen Tiefe dargestellt. Der Richtungsvektor wird dabei durch zwei Winkel parametrisiert.

In dieser Arbeit wurde zunächst die Parametrisierung des Richtungsvektors im Hinblick auf Winkelungenauigkeiten und -unstetigkeiten angepasst, indem die Winkel direkt durch einen Richtungsvektor ersetzt wurden. Damit wurde die Dimension eines Objekts zugunsten einer robusteren Repräsentation erhöht.
Anschließend wurde eine Initialisierung mehrerer Objekte in den Zustandsvektor erörtert. Im Zuge der Initialisierung musste aufgrund linearer Abhängigkeiten durch die Überparametrisierung eines Objekts durch Inverse Depth Parametrisierung auf den numerisch stabileren Square-Root UKF zurückgegriffen werden.
Neben einer Anpassung der Parametrisierung wurde das Sensormodell im Hinblick auf die Winkelungenauigkeiten und -unstetigkeitenangepasst, indem anstelle von Pixelkoordinaten ein Richtungsvektor prädiziert wird. Der Richtungsvektor aus der Sensorik ergibt sich aus der Transformation der detektierten Pixelkoordinaten mithilfe eines Kameramodells.
Abschließend ist eine Verwaltung der Karte entwickelt worden, die die Echtzeitfähigkeit des Systems gewährleisten soll. Diese Verwaltung teilt die entstehende Karte in eine kurzfristige und eine langfristige Karte auf. Damit besteht einerseits die Möglichkeit, kurzfristig in der lokalen Umgebung zu navigieren. Andererseits kann im Falle einer längeren Sensorstörung über die langfristige Karte eine Lokalisierung weiterhin stattfinden.

Diese Anpassungen der Modellierung sind mittels Simulink® validiert worden. Dabei ist die Funktionalität der Parametrisierung im Kontext von UAVs gezeigt worden. Insbesondere

die rein optische Navigation ist mit der gewählten Parametrisierung und Kartenverwaltung möglich. Zudem wurde gezeigt, dass die Verwendung eines weiteren Sensors die Navigation erheblich verbessert. Dementsprechend kann durch die Nutzung der Inverse Depth Parametrisierung im UKF SLAM die Navigation eines UAVs robuster gegenüber Störungen gestaltet werden.

Zur weiteren Verbesserung des entwickelten Lösungsansatzes ist eine Anpassung des Messmodells sinnvoll. Dieses weißt in der aktuellen Form Instabilitäten hinsichtlich der Genauigkeit auf. Dabei wäre es möglich, eine Dämpfung zu entwickeln, die den Einfluss einzelner fehlerhafter Messungen reduziert, indem die Ungenauigkeit erhöht wird. Damit wäre es zusätzlich möglich, mithilfe eines Hypothesentestverfahrens nicht verwendbare Messungen zu entfernen.
Weiterhin kann eine Abfangmöglichkeit für das Auftreten negativer inverser Tiefen in einzelnen Sigmapunkten entwickelt werden, da diese die prädizierte Messungen erheblich beeinflussen können. Einerseit besteht die Möglichkeit der Verwendung eines festen Grenzwertes als untere Grenze. Andererseits könnte eine inverse Tiefe verwendet werden, die sich aus den übrigen Sigmapunkten bildet oder der Sigmapunkt könnte von der Prädiktion der Messung ausgeschlossen werden.
Zur Abschließenden Validierung könnte die entwickelte Modellierung im Flugversuch eingesetzt und erprobt werden.

Literatur

[1] Ammann, N.; Andert, F.: Visual navigation for autonomous, precise and safe landing on celestial bodies using unscented Kalman filtering. In: 2017 IEEE Aerospace Conference, 04.03.2017 - 11.03.2017.

[2] Austin I. Eliazar; Ronald Parr: DP-SLAM: Fast, Robust Simultaneous Localization and Mapping Without Predetermined Landmarks. In: 2003.

[3] Bay, H.; Tuytelaars, T.; van Gool, L.: SURF: Speeded Up Robust Features. In: Computer Vision – ECCV 2006 (vol. # 3951), Leonardis, A.; Bischof, H.; Pinz, A., 2006.

[4] Bloesch, M.; Omari, S.; Hutter, M.; Siegwart, R.: Robust visual inertial odometry using a direct EKF-based approach. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 9/28/2015 - 10/2/2015.

[5] C. Forster; M. Pizzoli; D. Scaramuzza: SVO: Fast semi-direct monocular visual odometry. In: Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007.

[6] Cheng, Y.; Liu, Z.: Optimized selection of sigma points in the unscented Kalman filter. In: 2011 International Conference on Electrical and Control Engineering, 16.09.2011 - 18.09.2011.

[7] Civera, J.; Davison, A. J.; Montiel, J.: Inverse Depth Parametrization for Monocular SLAM. IEEE Transactions on Robotics, Bd. 24, Nr. 5, S. 932–945, 2008.

[8] Dissanayake, G.; Durrant-Whyte, H.; Bailey, T.: A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem. In: Robotics and Automation, 2000 IEEE International Conference, Aug. 2000.

[9] Dissanayake, G.; Williams, S. B.; Durrant-Whyte, H.; Bailey, T.: Map Management for Efficient Simultaneous Localization and Mapping (SLAM). Autonomous Robots, Bd. 12, Nr. 3, S. 267–286, 2002.

[10] Durrant-Whyte, H.; Bailey, T.: Simultaneous localization and mapping: part I. IEEE Robotics & Automation Magazine, Bd. 13, Nr. 2, S. 99–110, 2006.

[11] Durrant-Whyte, H. F.; Dissanayake, M. W. M. G.; Gibbens, P. W.: Toward Deployment of Large Scale Simultaneous Localisation and Map Building (SLAM) Systems. In: Robotics Research, Giralt, G.; Hirzinger, G., 1996.

[12] Dutton, B.; Maloney, E. S.: Dutton's Navigation & piloting, Annapolis, Md.: Naval Institute Press, ISBN 9780870211645, 1983, 1978.

[13] Eliazar, A.: DP-SLAM. Duke University, Dissertation, 2005.

[14] Frese, U.: Treemap: An O(log n) algorithm for indoor simultaneous localization and mapping. Autonomous Robots, Bd. 21, Nr. 2, S. 103–122, 2006.

[15] G. P. Huang; A. I. Mourikis; S. I. Roumeliotis: On the complexity and consistency of UKF-based SLAM. In: Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007.

[16] H. Khazraj; F. Faria da Silva; C. L. Bak: A performance comparison between extended Kalman Filter and unscented Kalman Filter in power system dynamic state estimation. In: 2016 51st International Universities Power Engineering Conference (UPEC), 2016.

[17] Hochdorfer, S.; Lutz, M.; Schlegel, C.: Lifelong localization of a mobile service-robot in everyday indoor environments using omnidirectional vision. In: IEEE International Conference on Technologies for Practical Robot Applications, 2009, 2009.

[18] Hochdorfer, S.; Schlegel, C.: Landmark rating and selection according to localization coverage: Addressing the challenge of lifelong operation of SLAM in service robots. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009.

[19] Holmes, S. A.; Klein, G.; Murray, D. W.: An O(N(2)) square root unscented Kalman Filter for visual simultaneous localization and mapping. IEEE transactions on pattern analysis and machine intelligence, Bd. 31, Nr. 7, S. 1251–1263, 2009.

[20] I. Jokić; Ž. Zečević and (B. Krstaji(ć: State-of-charge estimation of lithium-ion batteries using extended Kalman filter and unscented Kalman filter. In: 2018 23rd International Scientific-Professional Conference on Information Technology (IT), 2018.

[21] J. Civera; A. J. Davison; J. M. M. Montiel: Inverse Depth to Depth Conversion for Monocular SLAM. In: Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007.

[22] Julier, S.; Uhlmann, J.; Durrant-Whyte, H. F.: A new method for the nonlinear transformation of means and covariances in filters and estimators. IEEE Transactions on Automatic Control, Bd. 45, Nr. 3, S. 477–482, 2000.

[23] Julier, S. J.; Uhlmann, J. K.; Durrant-Whyte, H. F.: A new approach for filtering nonlinear systems. In: Proceedings of 1995 American Control Conference - ACC'95, 21-23 June 1995.

[24] Kaess, M.; Ranganathan, A.; Dellaert, F.: iSAM: Incremental Smoothing and Mapping. IEEE Transactions on Robotics, Bd. 24, Nr. 6, S. 1365–1378, 2008.

[25] Kalman, R. E.: A New Approach to Linear Filtering and Prediction Problems. Journal of Basic Engineering, Bd. 82, Nr. 1, S. 35–45, 1960.

[26] Kneip, L.; Chli, M.; Siegwart, R.: Robust Real-Time Visual Odometry with a Single Camera and an IMU. In: Proceedings of the British Machine Vision Conference 2011, Hoey, J., 2011.

[27] Leutenegger, S.; Lynen, S.; Bosse, M.; Siegwart, R.; Furgale, P.: Keyframe-based visual–inertial odometry using nonlinear optimization. The International Journal of Robotics Research, Bd. 34, Nr. 3, S. 314–334, 2015.

[28] Lowe, D. G.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, Bd. 60, Nr. 2, S. 91–110, 2004.

[29] Lynen, S.; Achtelik, M. W.; Weiss, S.; Chli, M.; Siegwart, R.: A robust and modular multi-sensor fusion approach applied to MAV navigation. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.

[30] M. Kaess; H. Johannsson; R. Roberts; V. Ila; J. Leonard; F. Dellaert: iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In: Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007.

[31] M. Rampelli; D. Jena: Advantage of Unscented Kalman Filter over Extended Kalman Filter in dynamic state estimation of power system network. In: Michael Faraday IET International Summit 2015, 2015.

[32] Marchthaler, R.; Dingler, S.: Kalman-Filter: Einführung in die Zustandsschätzung und ihre Anwendung für eingebettete Systeme. Lehrbuch, Wiesbaden: Springer Vieweg, ISBN 978-3-658-16727-1, 2017.

[33] Matas, J.; Chum, O.; Urban, M.; Pajdla, T.: Robust wide-baseline stereo from maximally stable extremal regions. Image and Vision Computing, Bd. 22, Nr. 10, S. 761–767, 2004.

[34] Michael Montemerlo; Sebastian Thrun; Daphne Koller; Ben Wegbreit: FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges, 2003.

[35] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit: FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem, 2002.

[36] Mitch Bryson; Salah Sukkarieh: Bearing-only SLAM for an airborne vehicle, 2005.

[37] Montiel, J.; Civera, J.; Davison, A.: Unified Inverse Depth Parametrization for Monocular SLAM. In: Robotics - science and systems II, Sukhatme, G. S.; Schaal, S., 2007.

[38] Müller-Gronbach, T.; Novak, E.; Ritter, K., Hrsg.: Monte-Carlo-Algorithmen. Springer-Lehrbuch, Berlin: Springer, ISBN 9783540891406, 2012.

[39] Qin, T.; Li, P.; Shen, S.: VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. IEEE Transactions on Robotics, Bd. 34, Nr. 4, S. 1004–1020, 2018.

[40] S. Hochdorfer; C. Schlegel: Towards a robust visual SLAM approach: Addressing the challenge of life-long operation. In: 2009 International Conference on Advanced Robotics, 2009.

[41] Sander, J.; Ester, M.; Kriegel, H.-P.; Xu, X.: Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. Data Mining and Knowledge Discovery, Bd. 2, Nr. 2, S. 169–194, 1998.

[42] Sola, J.; Monin, A.; Devy, M.; Lemaire, T.: Undelayed initialization in bearing only SLAM. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005.

[43] Sun, K.; Mohta, K.; Pfrommer, B.; Watterson, M.; Liu, S.; Mulgaonkar, Y.; Taylor, C. J.; Kumar, V.: Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight. IEEE Robotics and Automation Letters, Bd. 3, Nr. 2, S. 965–972, 2018.

[44] Thrun, S.: Robotic Mapping: A Survey. In: Exploring artificial intelligence in the new millennium. Amsterdam: Morgan Kaufmann, ISBN 1558608117, 2003.

[45] Thrun, S.; Burgard, W.; Fox, D.: Probabilistic robotics. Intelligent robotics and autonomous agents series, Cambridge, Massachusetts und London, England: MIT Press, ISBN 9780262201629, 2006.

[46] Thrun, S.; Montemerlo, M.: The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. The International Journal of Robotics Research, Bd. 25, Nr. 5-6, S. 403–429, 2006.

[47] Trawny, N.; Roumeliotis, S. I.: A unified framework for nearby and distant landmarks in bearing-only SLAM. In: Proceedings / 2006 IEEE International Conference on Robotics and Automation, 2006, ICRA 2006, 2006.

[48] Turner, R.; Rasmussen, C. E.: Model based learning of sigma points in unscented Kalman filtering. In: 2010 IEEE International Workshop on Machine Learning for Signal Processing, 29.08.2010 - 01.09.2010.

[49] U. Frese: Efficient 6-DOF SLAM with Treemap as a Generic Backend. In: Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007.

[50] van der Merwe, R.; Wan, E. A.: The square-root unscented Kalman filter for state and parameter-estimation. In: 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001.

[51] Wan, E. A.; van der Merwe, R.: The unscented Kalman filter for nonlinear estimation. In: Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373), 1-4 Oct. 2000.

[52] Wan, E. A.; van der Merwe, R.: The Unscented Kalman Filter. In: Kalman Filtering and Neural Networks. New York u. a.: Wiley, S. 221–280, ISBN 9780471221548, 2001.

[53] Wu, Y.; Hu, D.; Wu, M.; Hu, X.: Unscented Kalman filtering for additive noise case: augmented vs. non-augmented. In: Proceedings of the 2005, American Control Conference, 2005, June 8-10, 2005.

[54] Zhang, S.; Xie, L.; Adams, M. D.: Entropy based feature selection scheme for real time simultaneous localization and map building. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005.

A Anhang

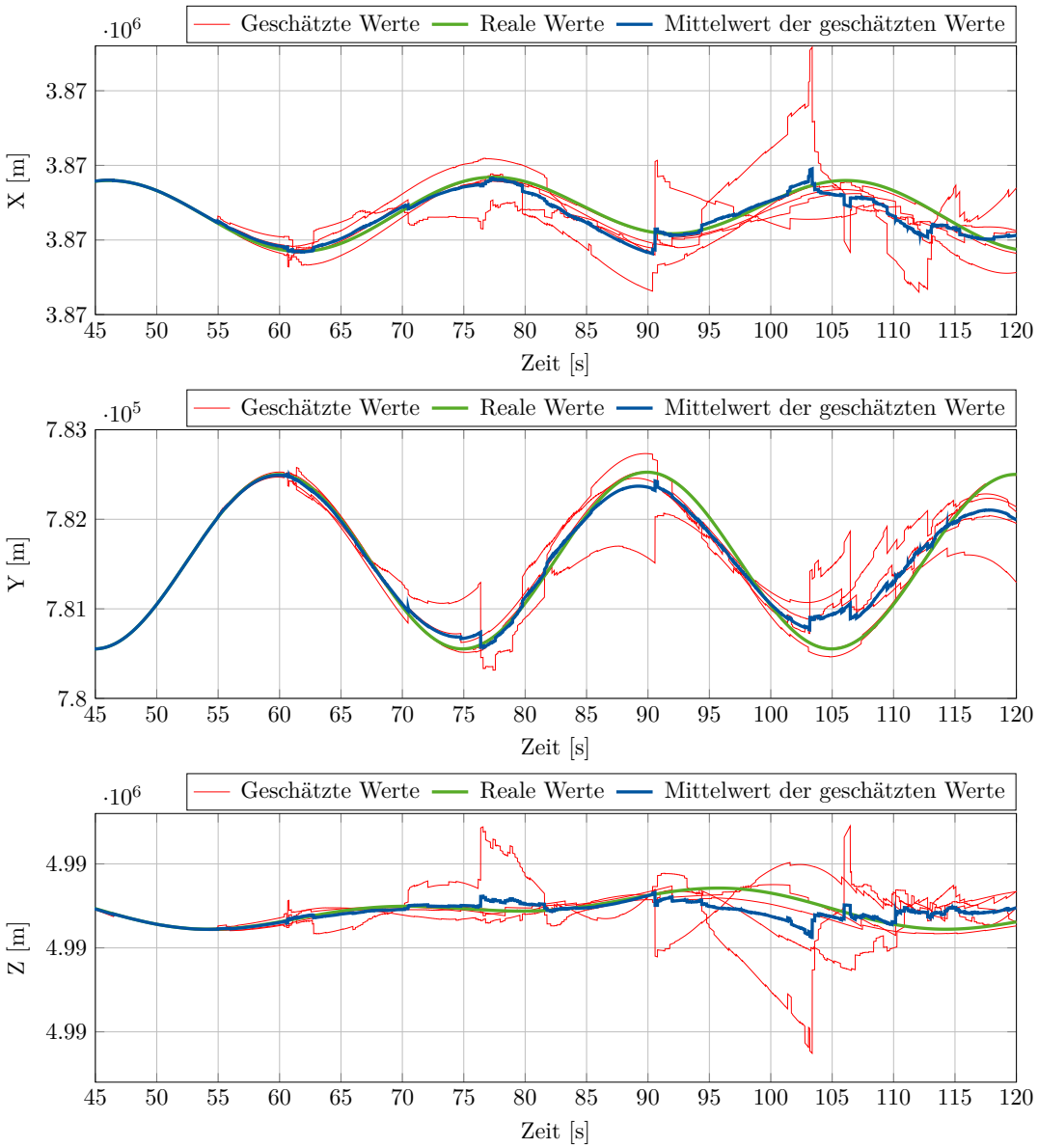


Abbildung A.1: Geschätzte Position in X, Y und Z-Koordinate mit Magnetometer mit Sensorausfall nach 50 s

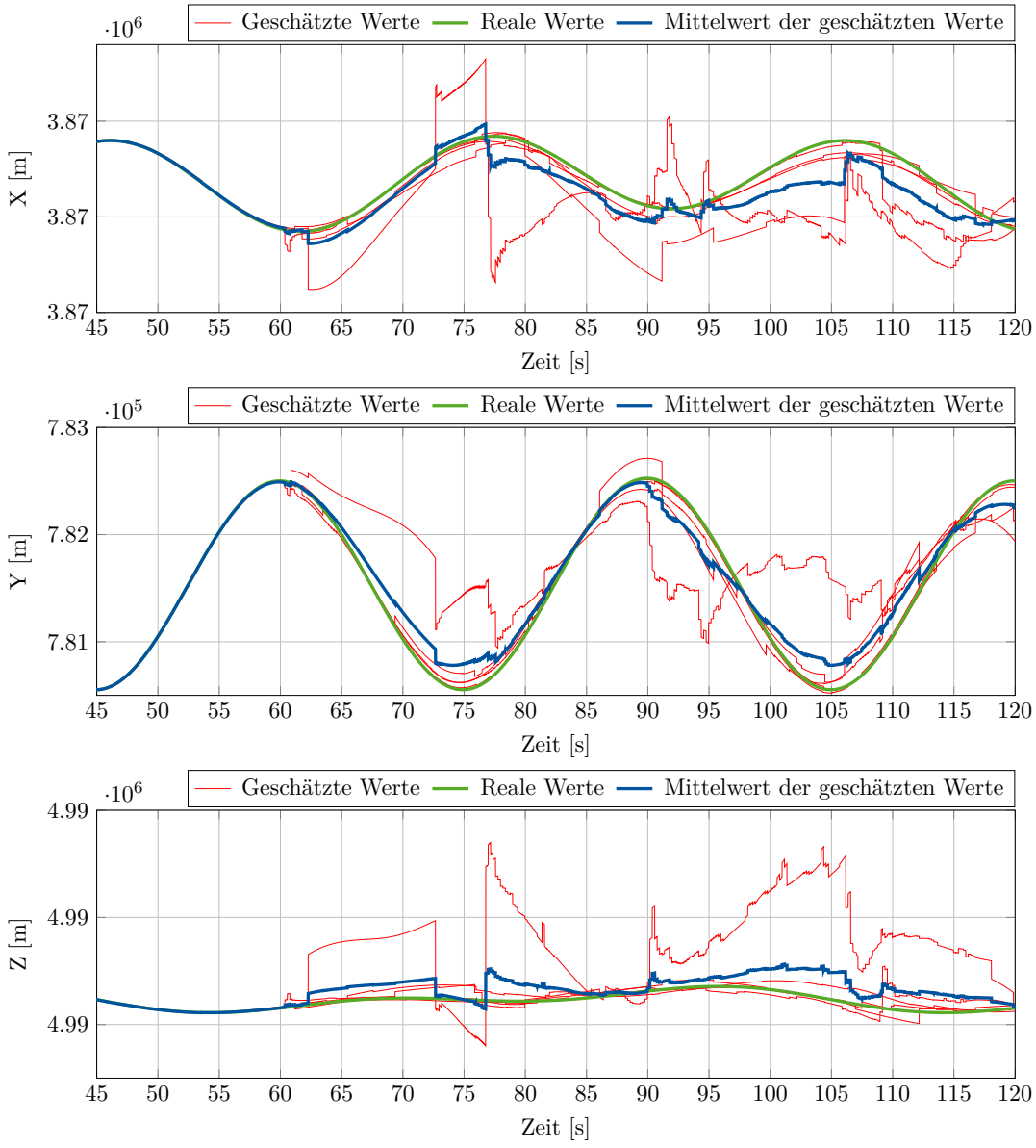


Abbildung A.2: Geschätzte Position in X, Y und Z-Koordinate ohne Magnetometer mit Sensorausfall nach 50 s

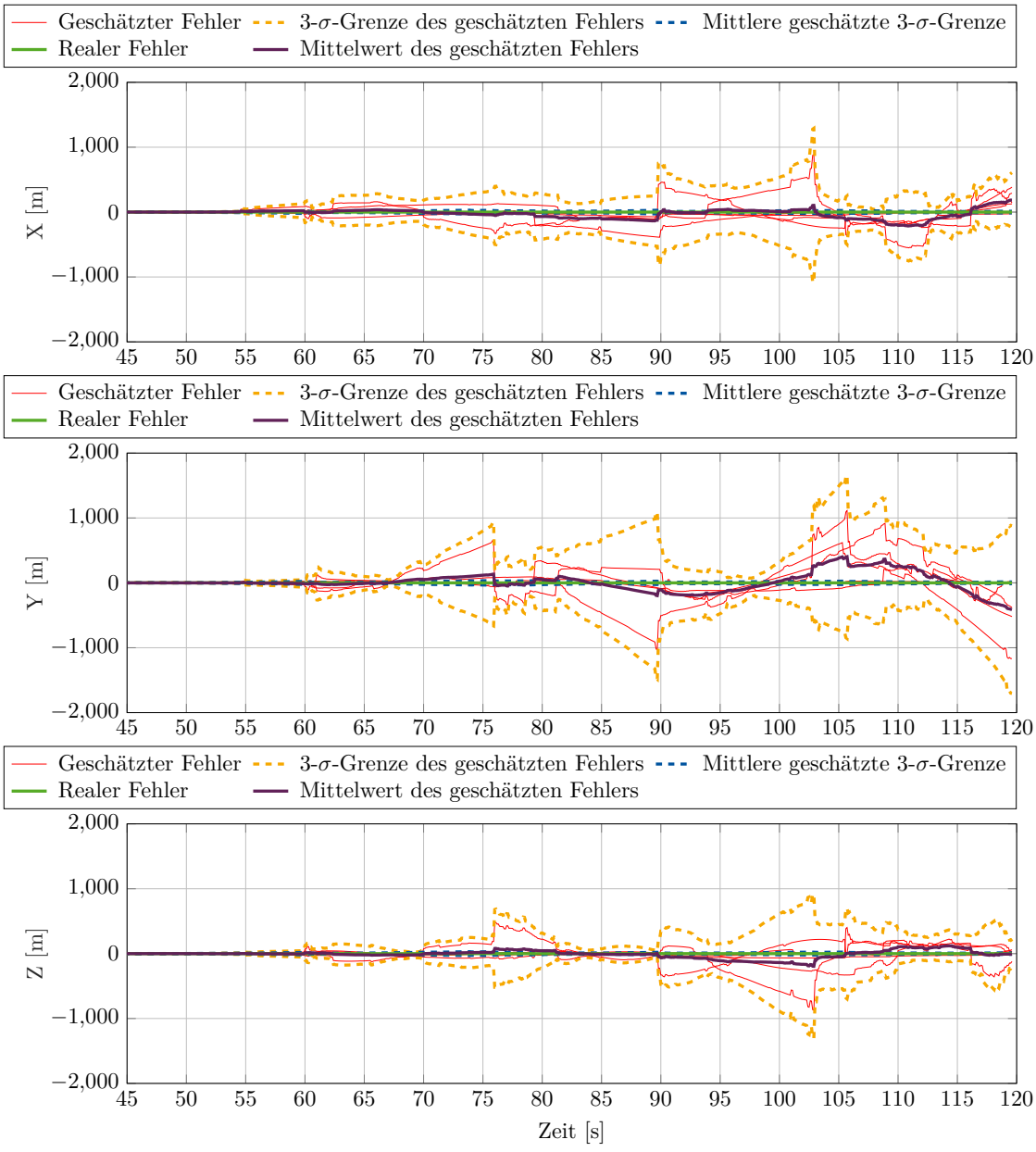


Abbildung A.3: Positionsfehler in X, Y und Z-Koordinate mit Magnetometer mit Sensorausfall nach 50 s

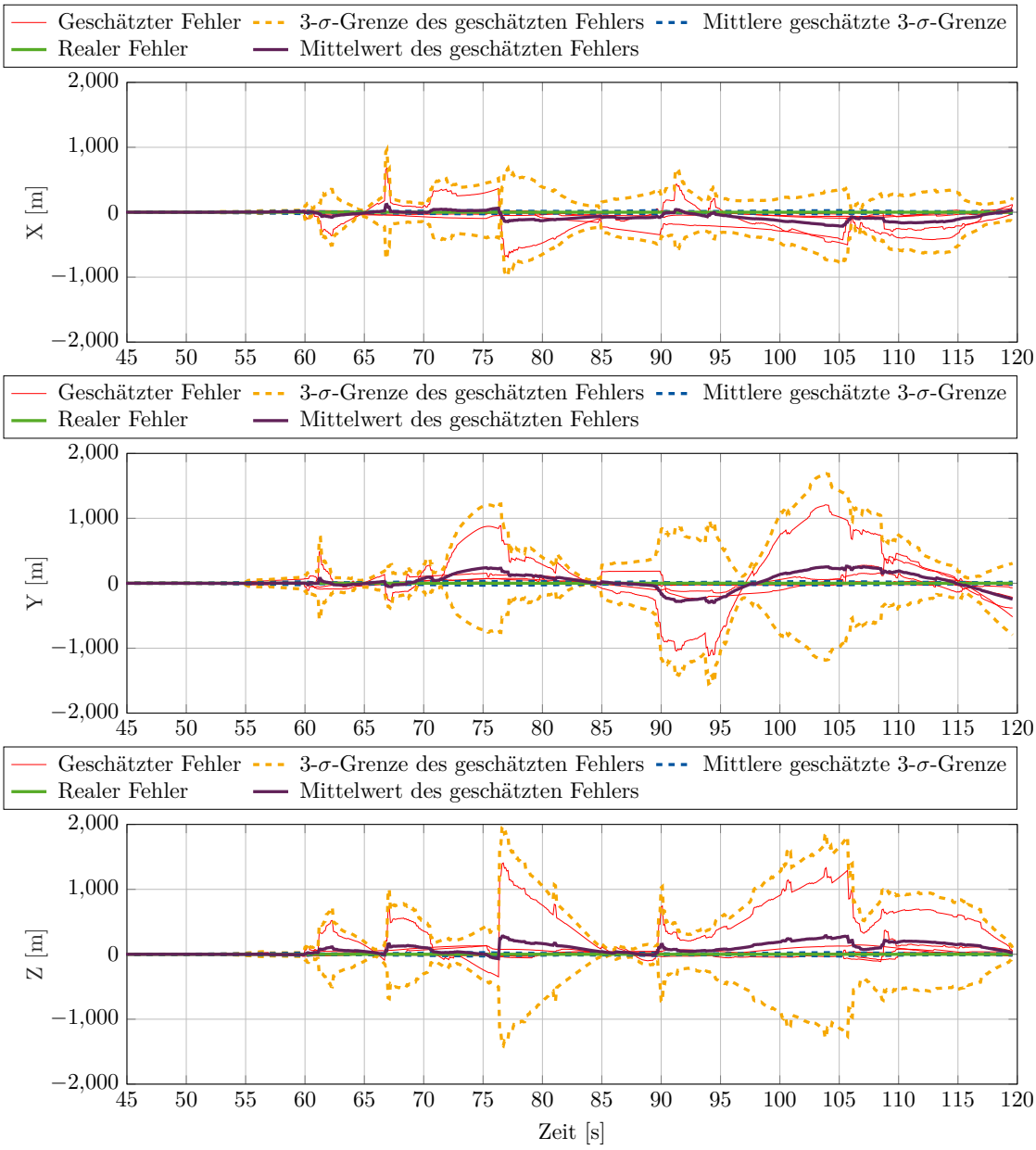


Abbildung A.4: Positionsfehler in X, Y und Z-Koordinate ohne Magnetometer mit Sensorausfall nach 50 s

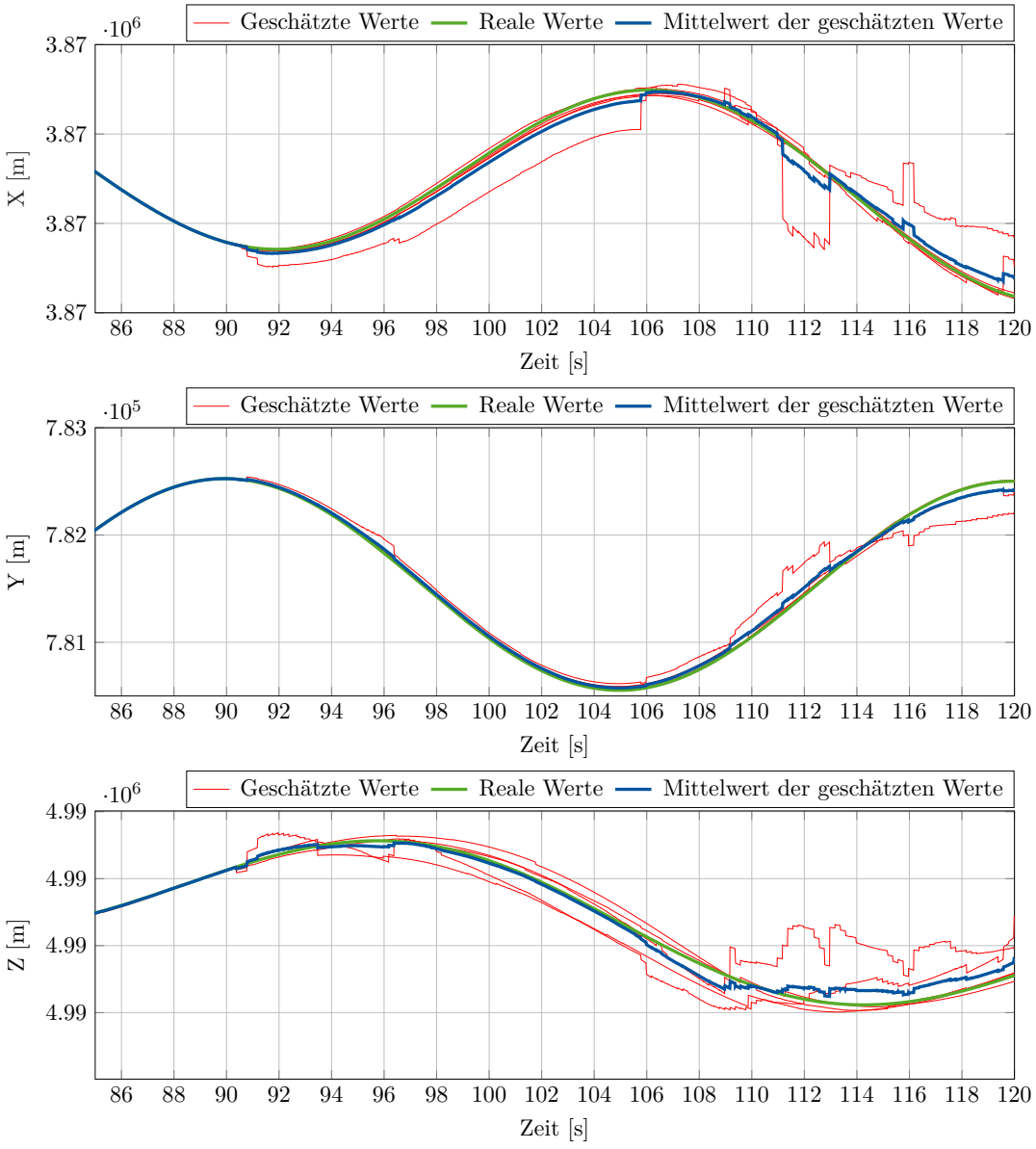


Abbildung A.5: Geschätzte Position in X, Y und Z-Koordinate mit Magnetometer mit Sensorausfall nach 90 s

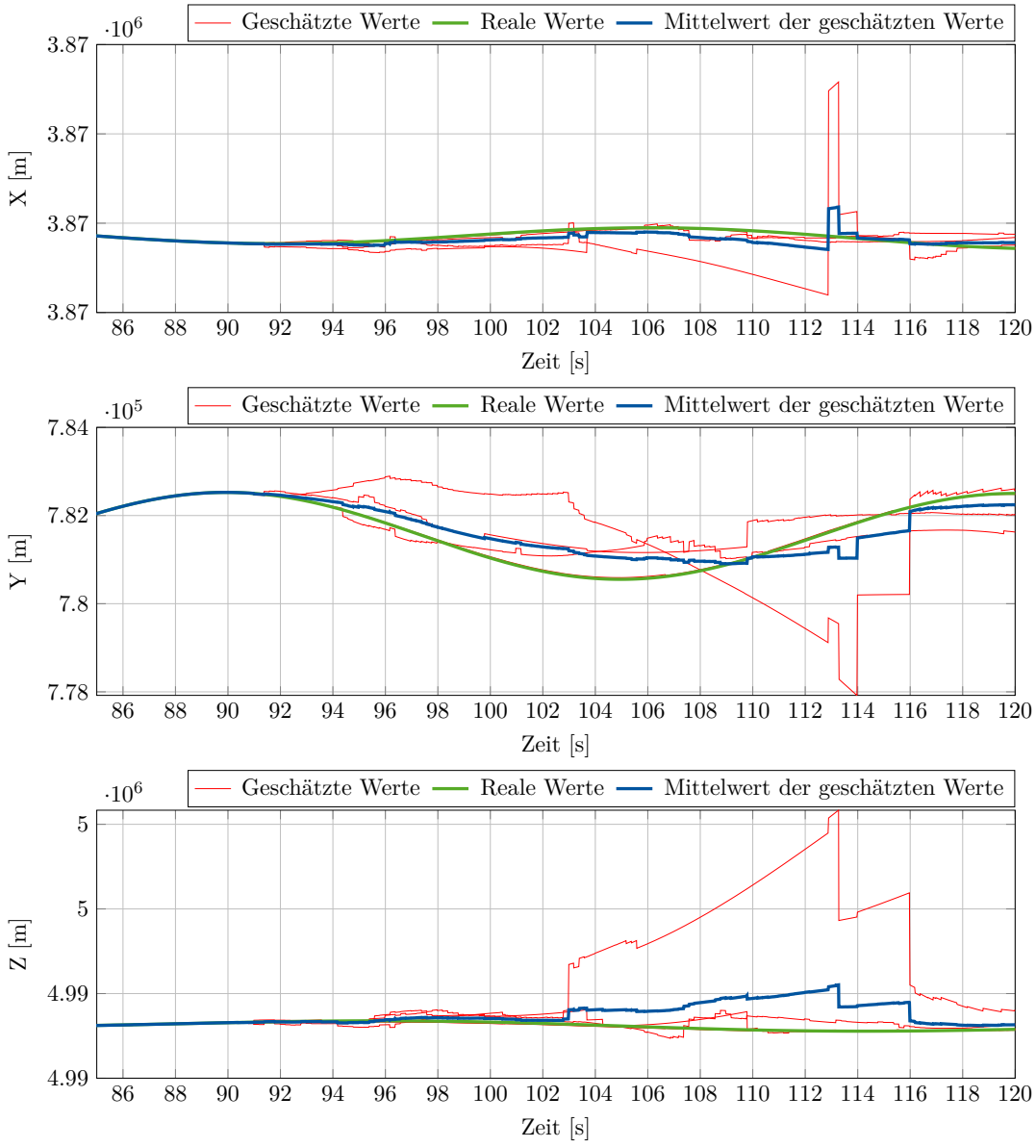


Abbildung A.6: Geschätzte Position in X, Y und Z-Koordinate ohne Magnetometer mit Sensorausfall nach 90 s

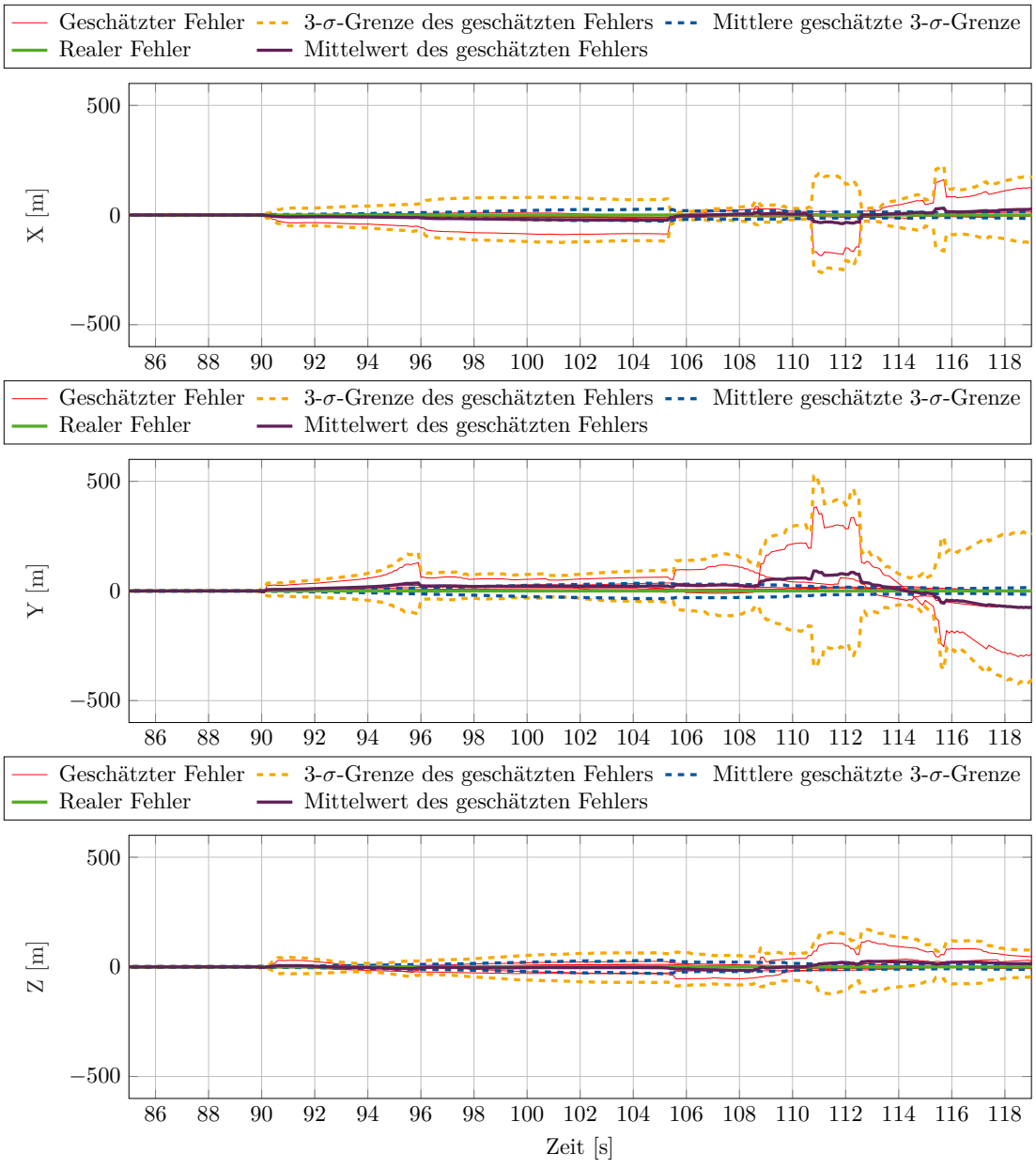


Abbildung A.7: Positionsfehler in X, Y und Z-Koordinate mit Magnetometer mit Sensorausfall nach 90 s

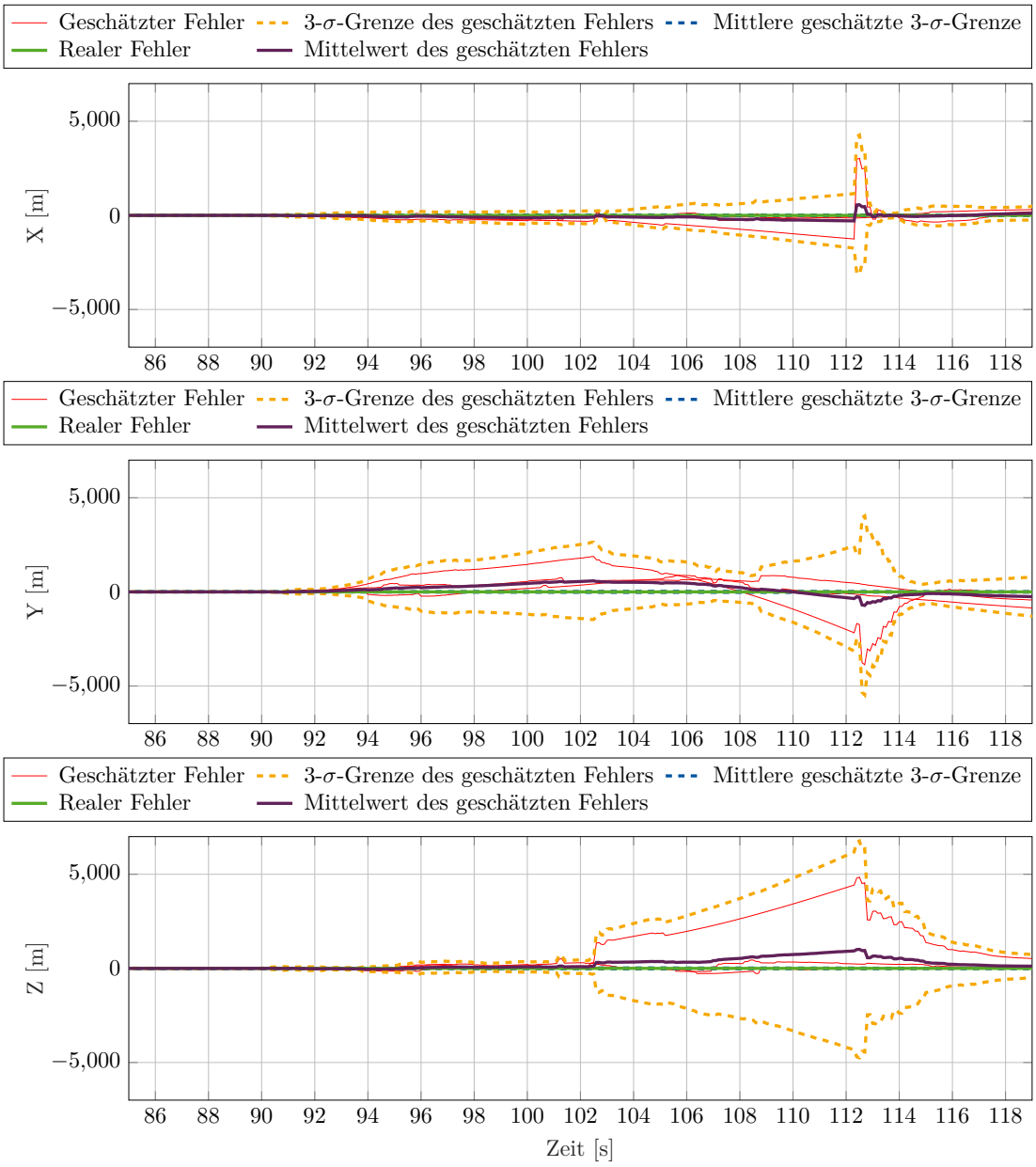


Abbildung A.8: Positionsfehler in X, Y und Z-Koordinate ohne Magnetometer mit Sensorausfall nach 90 s