Masterarbeit

Entwurf und Entwicklung
eines generischen
Multisensor-Trackers

Manfred Constapel

2017

Fachbereich Elektrotechnik und Informatik
Dekanat - Prüfungsausschuss

FACH HOCHSCHULE LÜBECK
University of Applied Sciences

Thema der Masterarbeit

für

Herrn Manfred C o n s t a p e l

Entwurf und Entwicklung eines generischen Multisensor Trackers

Professor Dr. Kratzke

gefördert durch:

Ausgabedatum: 02. Januar 2017
Abgabedatum: 02. Juli 2017

(Professor Dr. Hanemann)
Vorsitzender des Prüfungsausschusses

EMAS
GEPRÜFTES UMWELTMANAGEMENT

Aufgabenstellung:

Gegenstand dieser Masterarbeit ist die Entwicklung einer generischen Tracker-Architektur zur Fusion und Anzeige einer mittels mehrerer Radarsystemen erfassten Luftlage sowie der Nachweis der Implementierbarkeit anhand eines Software-Prototypen inkl. dessen Evaluation.

- Es sind mehrere mobile (Landfahrzeug-, Schiffs- oder Flugzeug-gestützte) Radarsysteme zu berücksichtigen, die in einem Einsatzgebiet positioniert werden oder patrouillieren.

- Es gibt einen zentralen Knoten, der die Daten zu Azimut, Elevation und Entfernung und die Fehler von den einzelnen Stationen erhält und daraus den Track generiert ("der Tracker").

- Zu berücksichtigende Probleme sind z. B. die unterschiedlichen Abtastfrequenzen, die (zeitliche) Synchronisation und die Fusion der Daten. Sensoren sind geeignet zu parametrisieren und deren generische Einbindung in das System vorzusehen.

- Der Tracker soll mehrere Trackingalgorithmen und Bewegungsmodelle für die Vorhersage und Korrektur des Tracks sinnvoll integrieren. Es soll berücksichtigt werden, das grundsätzlich mehrere Trackingalgorithmen bzw. Bewegungsmodelle je nach Fall bzw. Bewegung angewendet werden können.

- Die Trackingalgorithmen und Bewegungsmodelle sollen systematisch evaluiert werden. Hierzu ist ein Experimentplan und daraus abgeleitete Szenarien (Testdaten, z. B. ein Flugobjekt auf einer Kreisbahn im Raum oder anhand von Echtdaten) zu entwickeln, zu dokumentieren und durchzuführen.

- Als Programmiersprache für die Implementierung ist Python vorgesehen.

Professor Dr. Kratzke

Fachbereich Elektrotechnik und Informatik
Dekanat - Prüfungsausschuss

FACH HOCHSCHULE LÜBECK
University of Applied Sciences

Erklärung zur Abschlussarbeit

Ich versichere, dass ich die Arbeit selbständig, ohne fremde Hilfe verfasst habe.

Bei der Abfassung der Arbeit sind nur die angegebenen Quellen benutzt worden. Wörtlich oder dem Sinne nach entnommene Stellen sind als solche gekennzeichnet.

Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, insbesondere dass die Arbeit Dritten zur Einsichtnahme vorgelegt oder Kopien der Arbeit zur Weitergabe an Dritte angefertigt werden.

_____ _____

(Datum) Unterschrift

Abschlussarbeit

zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

an der

Fachhochschule Lübeck (University of Applied Sciences)
Fachbereich Elektrotechnik und Informatik

im Studiengang

Medieninformatik

zum Thema

Entwurf und Entwicklung eines generischen Multisensor-Trackers

ausgeschrieben von

MBDA Deutschland GmbH
Bereich Applikationssoftware und Führungssystem-Algorithmen

Autor:	Manfred Constapel
Geburtsdatum:	09.12.1979
Matrikelnummer:	259013
Betreuer:	Dipl.-Math. Michael Soellner
Erstprüfer:	Prof. Dr. rer. nat. Nane Kratzke
Zweitprüfer:	Dr. phil. Andreas Harder

Danksagung

Diese Masterarbeit entstand bei der MBDA Deutschland GmbH im Bereich Applikationssoftware und Führungssystem-Algorithmen. Ich möchte mich bei meinen dortigen Betreuern Torsten Ludwig, Michael Soellner und Florian Maurer bedanken, die mir die Bearbeitung dieses sehr interessanten Themas ermöglicht haben. Für das entgegengebrachte Vertrauen, die stets freundliche, engagierte und kollegiale Betreuung sowie die hilfreichen Gespräche und Diskussionen gilt Ihnen mein besonderer Dank.

Ich bedanke mich bei Prof. Dr. Nane Kratzke aus dem Fachbereich Elektrotechnik und Informatik der Fachhochschule Lübeck für die Möglichkeit diese Arbeit unter seiner Leitung durchzuführen und das konstruktive Feedback. Mein Dank geht auch an Dr. Andreas Harder, welcher sich für die Zweitkorrektur bereiterklärt und außerhalb dieser Arbeit meinen Blick auf fachliche Grundlagen geschärft hat.

All jenen, die mich im Rahmen meines Studiums und dieser Masterarbeit begleitet und unterstützt haben, und denen, die ihre Zeit und Energie zur Verfügung gestellt haben, um ihr Wissen und ihre Erfahrung weiterzugeben, gilt ebenfalls mein Dank.

Abstract

In a multi-sensor environment noisy measurements of sensors tracking targets are processed by filters. Filters and motion models are used for state estimation of tracked targets.

Since tracking with radar by azimuth, elevation and range is inherently a non-linear process, the application of non-linear estimation techniques becomes inevitable. For that, implementations of two estimation techniques - extended Kalman filter (EKF) and unscented Kalman filter (UKF) - are provided.

In order to increase the accuracy and robustness of the estimation motion models are applied to filters. Therefore, two standard motion models - constant velocity (CV) and constant acceleration (CA) - and two advanced models - constant yaw rate pitch rate velocity (CYRPRV) and constant yaw rate pitch rate acceleration (CYRPRA) - are derived.

Filters act as local trackers performing data association and senor-to-track fusion of measurements containing positions only. Their outputs are sent to a central node carrying out track-to-track fusion. To raise ease of management and interchangeability for various sensors, filters and motions models employable in 2D and 3D space, an architectural framework based on an object-oriented paradigm is developed.

To demonstrate and evaluate implemented filters and motions models nested in the object-oriented framework, an interactive Python application is presented serving as a virtual testbed in this thesis. Evaluation can be done within the testbed by means of Monte Carlo simulations either in 2D or in a virtually real world environment defined by geograhical areas.

For further evaluation UKF is chosen to be nested in the object-oriented architecture due to evidence of its superiority in tracking highly manoeuvring targets. By purpose designed scenarios are utilised to evaluate all motion models presented in this thesis for identifying the most versatile pairing. In comparsion UKF and CYRPRV achieve best results, which makes them beside tracking air targets applicable to a wide range of tracking problems.

Inhaltsverzeichnis

1. Einleitung

1.1 Einführung

Die ersten praktisch eingesetzten Radare (radio detection and ranging) erlaubten im Zweiten Weltkrieg die Detektion und Entfernungsmessung von weit entfernten Objekten. Das Radar fungierte dabei vor allem als Frühwarnsystem für sich annähernde und feindliche Flugzeuge in Küstenabschnitten und wurde auch auf Schiffen installiert [KM07]. Die Detektion und Entfernungsermittlung ist in diesem Zusammenhang ein überwiegend technisches und nach wie vor anspruchsvolles Problem. Die an der Sendeantenne des Radars abgestrahlten elektromagnetischen Wellen erzeugen durch Reflexion an einem Objekt ein Signal, welches durch die Radar-Empfangsantenne aufgefasst werden kann [YJX16]. Die Anzeige einer Detektion erfolgte in der Regel durch einen aufleuchtenden Punkt oder "Blip" auf dem Radarbildschirm.

Mit Fortschreiten der Radartechnik und durch die Entwicklung von Rechnertechnik konnten die ersten Tracker entwickelt werden, welche neben der einfachen Detektion eine Zielverfolgung durch die Zuordnung eines "Tracks" zu einem "Blip" ermöglichten. Durch diese trivial anmutende Verknüpfung besteht neben der Möglichkeit der Zuordnung einer eindeutigen Identifikation zu einem Ziel auch die Möglichkeit dessen Bewegung über die Zeit zu speichern und vorherzusagen [SSCB14].

Die Anzeige der vergangenen Bewegung als "Vergangenheitsspur" sowie der voraussichtliche Bewegung eines jeden Ziels bzw. "Blips" kann bei Vorhandensein sehr vieler Ziele oder in komplexen Lagen, z. B. bei der Flugsicherung oder bei der Radarberatung für Schiffe in sehr engen und vielbefahrenen Gewässern bei schlechtem Wetter, die Beobachtung der wirklich relevanten und interessanten Ziele extrem vereinfachen [BDW+05]. Tracker sind insofern essentiell und stellen einen Sicherheitsaspekt dar, wenn es um die Beurteilung und Interpretation einer Lage mit Hilfe eines Radarbildschirms geht.

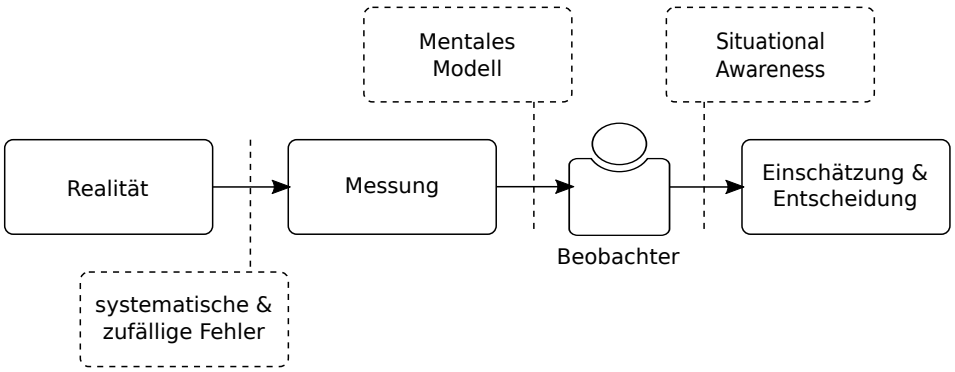


Abbildung 1.1: Spannungsfeld zwischen Realität und Beobachtung

1.2 Motivation

Sensordatenfusion

Die Messung der Entfernung zu einem Ziel erfolgt durch Vergleich der Sendezeit mit der Empfangszeit des reflektierten Signals bei Kenntnis der Ausbreitungsgeschwindigkeit der gesendeten Welle. Die Peilung des Objekts (Azimut) ergibt sich aus der Ausrichtung der Empfangsantenne. Die Bestimmung der Höhe eines Objekts (Elevation) erfolgt in der Regel durch ein weiteres Radar (Höhenmesser).

Die aufgrund einer Messung ermittelten Werte zu Entfernung, Azimut und Elevation sind wie bei allen physikalischen Messungen, welche mit technischen Hilfsmitteln durchgeführt wurden, grundsätzlich fehlerbehaftet. Etwaige Umwelteinflüsse, Toleranzen bei der Konstruktion des Radars, elektrische und mechanische Störungen, konstruktionsbedingte Einschränkungen und sonstige Fehler erlauben lediglich die Durchführung "verrauschter" Messungen [Mit07]. Dieses Rauschen steigt in der Regel mit zunehmender Entfernung zum Ziel. Dennoch können verrauschte Messungen unter bestimmten Bedingungen verwertet werden und insgesamt eine verbesserte Messung liefern, was im Rahmen dieser Arbeit gezeigt werden soll [FR99].

Realistische Simulation

Die nachfolgende Abbildung zeigt einen dreiminütigen Ausschnitt einer simulierten Lage auf der ostfriesischen Halbinsel in einer zweidimensionalen Kartenansicht. Die Lage enthält vier seegehende und drei fliegende Ziele (quadratische Symbole). Ein stationäres Radar mit großer Reichweite im westlichen Teil des Gebiets und ein mobiles Radar östlich davon (dreieckige Symbole) liefern die Messungen (gekreuzte Symbole) zu den Zielen.

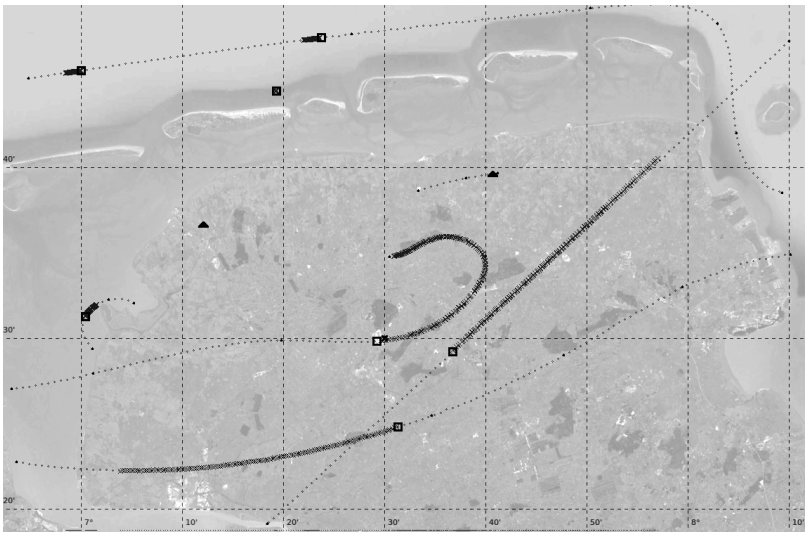


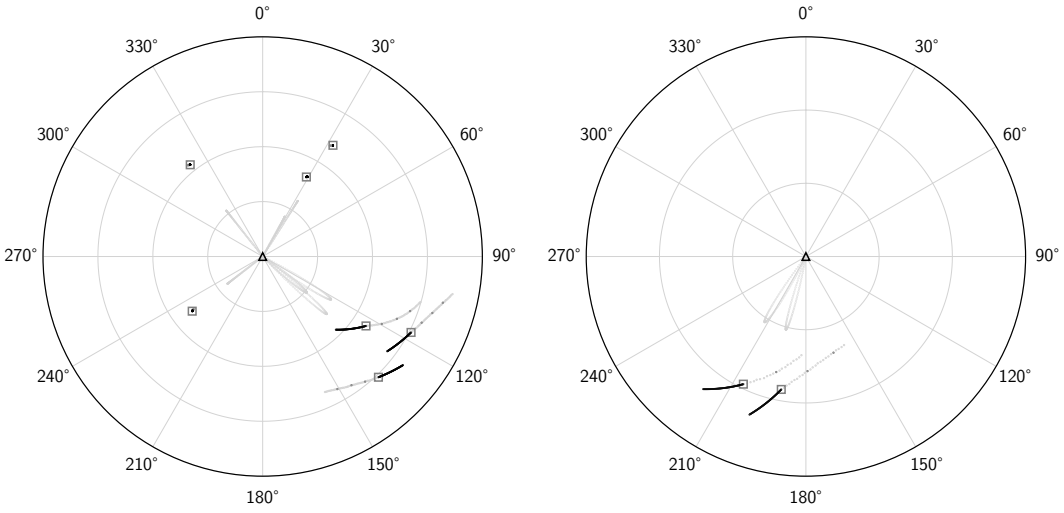
Abbildung 1.2: Simulierte Lage mit zwei Radaren und sieben Zielen; 2D-Darstellung

Die Simulation von möglichst realistischen Lagen mit "virtueller Hardware" und der Option des Imports von Echtdaten stellt eine kostengünstige, sehr flexible und jederzeit verfügbare Alternative zu Untersuchungen mit echten Radarsystemen und Flugobjekten dar. Eine softwaremäßige Lösung dieses Ansatzes - mittels einer Applikation derartige Simulationen durchzuführen - konnte im Rahmen der Recherche nicht identifiziert werden. Zu diesem Zweck wird in dieser Arbeit das Virtual Testbed (siehe Anhang A) mit der Programmiersprache Python entwickelt, welches ebenfalls im Rahmen der Evaluation zur Anwendung kommen soll [MS15, LL15, Lab15].

Erstellung eines Lagebilds

Die folgenden Abbildungen zeigen in Anlehnung der Lage aus Abbildung 1.2 die Radarbilder der beiden eingesetzten Radare mit Distanzringen im Abstand von 10 km. Das stationäre Radar kann aufgrund der höheren Reichweite alle Ziele in der Lage erkennen. Das mobile Radar stellt nur die beiden dichtesten Ziele wegen der geringeren Reichweite dar.

Der Radarbildschirm des mobilen Radars zeigt nur einen Teil der realen Lage, dafür aber mit einer potentiell höheren Genauigkeit aufgrund der geringeren Entfernung der Ziele.



(a) Stationäres Radar mit großer Reichweite (b) Mobiles Radar mit geringer Reichweite

Abbildung 1.3: Simulierte Lage mit zwei Radaren und sieben Zielen; Radarbildschirme

Durch die Kombination (Fusion) der Radarbilder entsteht ein ganzheitliches Lagebild. Ein solches Lagebild kann zentral angezeigt werden oder die einzelnen Radarbilder ergänzen, so dass auf jedem Radarbildschirm das gesamte Lagebild ersichtlich ist. Dieser Vorgang kann schrittweise beschrieben werden [Fou15]:

1. Die Generierung von Sensor-Tracks erfolgt mittels Sensor-Track-Fusion und lediglich mit Messungen, welche der Sensor selbst durchgeführt hat. Bei Radar-Systemen findet diese Verarbeitung häufig in einer der Signalverarbeitung nachgelagerten Logik im Radar-System selbst statt.

2. Die Sensoren schicken für die weitere Verarbeitung ihre Sensor-Tracks über ein Sensornetz an eine übergeordnete Instanz, den sogenannten Tracker.

3. Der Tracker erzeugt System-Tracks aus den Sensor-Tracks für die Anzeige eines Lagebilds. Diejenigen Sensor-Tracks, welche aus Sicht des Trackers demselben Ziel zuzuordnen sind, werden mittels Track-Track-Fusion zu einem einzelnen System-Track fusioniert.

In dieser Arbeit wird ein zentralistischer Ansatz der Sensor-Track-Fusion zur Generierung von System-Tracks verfolgt: Die Messungen der Radare werden anstelle von Sensor-Tracks in das Sensornetz geschickt. Die Erzeugung der System-Tracks erfolgt direkt aus den Messungen. Dadurch ist auf Sensorebene keine Verarbeitung zur Generierung von Sensor-Tracks erforderlich.

Dreidimensionale Visualisierung

Die Darstellung einer frei schwenkbaren Anzeige in der nachfolgenden Abbildung zeigt das Lagebild der aus den beiden Radaren aus Abbildung 1.3 integrierten Messungen in drei Dimensionen. Es ist ersichtlich, dass eine solche räumliche und plastische Anzeige einen Mehrwert bezüglich des Informationsgehalts zur Beurteilung der Lage darstellt [DP15].

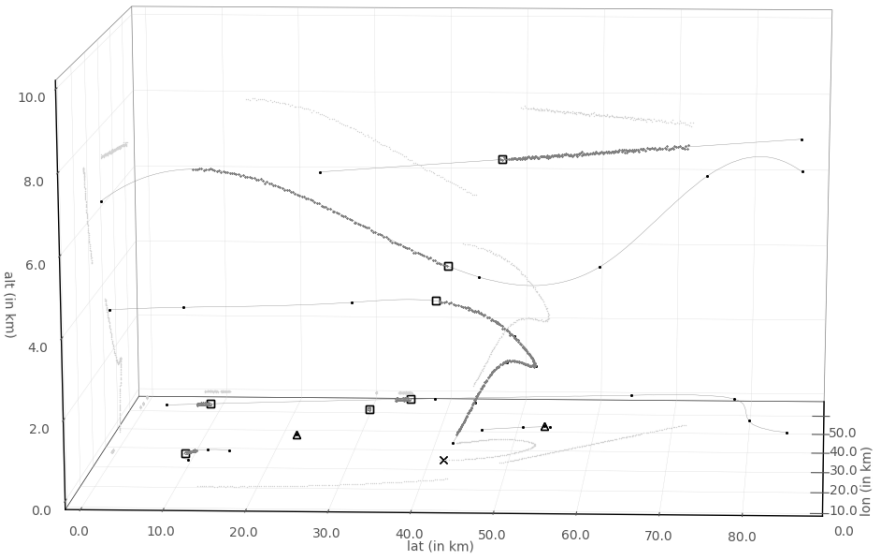


Abbildung 1.4: Simulierte Lage mit zwei Radaren und sieben Zielen; 3D-Darstellung

Die Umsetzung einer Lage-Darstellung in 3D erfolgt in dieser Arbeit zur Anzeige einer Luftlage prototypisch im Virtual Testbed (siehe Anhang A).

Vorhersage der Zielbewegung

In den Radarbild-Darstellungen der Abbildung 1.3 sind neben den gräulich eingefärbten Vergangenheitsspuren, welche die vergangene Bewegung des Ziels darstellen, auch schwarz eingefärbte Fahrtvektoren zu sehen. Die Fahrtvektoren geben mit ihrer Länge und Richtung die voraussichtliche Position des Ziels nach einer bestimmten Zeitspanne an (hier 30 Sekunden).

Die Vorhersage der voraussichtlichen Zielbewegung und damit auch die Erzeugung des Fahrtvektors ist in keinster Weise trivial. Eine einfache Extrapolation der letzten Messungen erlaubt keine Vorhersage der Zielbewegung, da die verrauschten Messungen weder stets noch grundsätzlich im korrekten Abstand und direkt auf der Trajektorie des Ziels liegen. Ferner ist es möglich, dass zeitweilig gar keine Messungen verfügbar sind, beispielsweise weil das Ziel durch ein Hindernis abgedeckt ist.

In diesem Zusammenhang sind Filter und Bewegungsmodelle unerlässlich. In Kombination gestatten sie die Schätzung des Zielzustands (Ort, Geschwindigkeit usw.), aus welchem die voraussichtliche Bewegung abgeleitet werden kann. Typische Vertreter für Filter sind in diesem Bereich die Varianten der Kalman Filter. Diese ermöglichen unter bestimmten Voraussetzungen eine optimale Schätzung des Zielzustands mittels eines Bewegungsmodells; auch bei verrauschten Messungen [Zar05].

Die Implementierung und der Einsatz von Filtern und Bewegungsmodellen zur Schätzung eines Zustands bringt in der hier behandelten Problem-Domäne eine Reihe von Herausforderungen mit sich, welche in dieser Arbeit diskutiert mit den hier implementierten Filtern dargestellt werden.

Generischer Ansatz

In dieser Arbeit soll mit Hilfe einer objektorientierten Tracker-Architektur ein über ein Sensornetz geordnetes Zusammenspiel von Sensoren, Filtern und Bewegungsmodellen orchestriert werden. Ferner soll die Architektur als Grundlage für ein solides Sensor- und Filtermanagement mit dem Ziel der Anzeige eines korrekten Lagebilds dienen. Für die Tracker-Architektur erfolgt ein Proof-of-Concept im Rahmen einer durchzuführenden Evaluation mit einer Auswahl von Kombinationen aus implementierten Filtern und Bewegungsmodellen sowie parametrisierten Radaren in dafür entwickelten Szenarien.

Ein weiteres Ziel der Evaluation ist die Identifikation eines Verfahrens - einer Paarung aus Filter und Bewegungsmodell - welches die Vorhersage eines möglichst breiten Spektrums an Bewegungen erlaubt (generischer Tracker). Dazu wird ein systematischer und quantitativer Vergleich von Ergebnissen aus Simulationen mit Fokus auf einen möglichst breiten und vielseitigen Einsatz durchgeführt.

In diesem Zusammenhang spielt das Virtual Testbed (siehe Anhang A) für die Simulation von Szenarien sowie der Erzeugung von Datensätzen für Auswertungen eine maßgebliche Rolle.

1.3 Eingrenzung

In der Literatur sind häufig die dort diskutierten Aspekte, Verfahren und Probleme zum Tracking aus meist didaktischen Gründen auf zwei Dimensionen reduziert. In dieser Arbeit erfolgt die Diskussion in großen Teilen in drei Dimensionen, was eine schnell wachsende Komplexität mit sich bringt, aber in Anbetracht des Fokus dieser Arbeit unerlässlich ist.

Bezogen auf die betrachteten Datenfusionsverfahren nimmt die Sensor-Track-Fusion in dieser Arbeit einen wesentlich größeren Stellenwert ein als die Track-Track-Fusion [BSCK01, BSLS00], welcher im Verlauf keine besondere Rolle zuteil kommt. Dies ist vor allem begründet durch das Design der Architektur und der direkten Verfügbarkeit von Messungen aus der Sensorebene.

Hinsichtlich der verwendeten Filter liegt der Fokus auf statistischen Filtern (Bayesschen Filtern) [C$^+$03], hier im Speziellen den Kalman Filtern [K$^+$60], welche im Bereich Radar-Tracking sehr populär sind.

Es werden nur 1:1-Beziehungen zwischen Filter und Bewegungsmodell betrachtet. Unabhängig davon werden 1:n-Beziehungen, also Interacting Multiple Model (IMM) [Gen01, JK13], grundsätzlich berücksichtigt, erfahren in dieser Arbeit aber keine besondere Beachtung.

Eine Diskussion des Ursprungsproblems (die Zuordnung von Messungen zu Zielen) erfolgt im Allgemeinen und im Rahmen der Erörterung der Probleme bei der Datenassoziierung bzw. Präsenz von Mehrfachzielen.

1.4 Ablauf

Im zweiten Abschnitt werden die Grundlagen für die folgenden Abschnitte erörtert. Dies beinhaltet inhaltlich vor allem die Funktion der verwendeten Filter und die bei einem Einsatz zu behandelnden Probleme, aber auch Aspekte der Datenfusion.

Der dritte Abschnitt beinhaltet im Rahmen der Modellierung eine Auswahl von relevanten Bewegungsmodellen für eine Analyse. Ein Ansatz für zwei Bewegungsmodelle, welche speziell an die Erfordernisse der hier betrachteten Zielverfolgung von Luftfahrzeugen angepasst sind, schließt den Abschnitt ab.

Im vierten Abschnitt sind die Anforderungen und der Entwurf der Komponenten des Virtual Testbed verankert. Dazu gehören die Tracker-Architektur, die Simulationsumgebung und die Visualisierung.

Der fünfte Abschnitt dokumentiert die Realisierung des Virtual Testbed.

Der sechste Abschnitt dient der Evaluation der Tracker-Architektur und dem Vergleich der beiden vorgeschlagenen Bewegungsmodelle mit zwei Standard-Bewegungsmodellen zur Identifikation eines möglichst vielseitigen Verfahrens zur Zielverfolgung.

2. Grundlagen

Einen großen Anteil nehmen in dieser Arbeit die Filter ein. Filter dienen als Vehikel zur Schätzung des Zielzustands und der Integration von Messungen zur Verbesserung einer Schätzung. Entsprechend erfolgt eine Einordnung solcher Filter in diesem Abschnitt.

Verschiedene Sensoren liefern Messungen zu unterschiedlichen Zeitpunkten und in verschiedenen Qualitäten. Die Datenfusion ist ein zentraler Aspekt, welcher essentiell ist für eine möglichst optimale "Vermengung" solcher Messungen. Eine Erläuterung der Grundlage für die Datenfusion befindet sich ebenfalls in diesem Abschnitt.

Ein Filter erhält neben den für ihn interessanten Messungen auch Messungen, welche explizit nicht in die Korrektur einer Schätzung eingehen sollen. Dies betrifft beispielsweise Messungen, die nicht zu dem verfolgten Ziel gehören. Die Ansätze, die versuchen dieses Problem zu lösen, fallen in die Kategorie der Datenassoziierung, welche in diesem Abschnitt als Abschluss beschrieben ist.

2.1 Filter

Filter sind aus verschiedenen Gründen notwendig für die Bestimmung eines Zielzustands. Zum einen können Messungen nur sehr unregelmäßig zur Verfügung stehen, dennoch soll ein Zielzustand kontinuierlich (beispielsweise für die Anzeige auf einem Lagebildschirm) definiert sein. Zum anderen sind Messungen grundsätzlich verrauscht und mit Unsicherheiten verbunden, welche die Filter versuchen zu minimieren.

Die Grundlage für eine Schätzung bildet ein Prozess (oder eine Dynamik), welcher dem Filter bekannt sein muss. Ein Prozess ist in diesem Kontext stets ein Bewegungsmodell. Bewegungsmodelle müssen korrekt formuliert und in den Filter eingebettet sein, damit der Filter auf dieser Grundlage Schätzungen durchführen kann (vgl. Abschnitt 3). Die hier betrachteten Bayesschen Filter nutzen einen rekursiven Ansatz zur Schätzung, indem sie den letzten, älteren Zustand für die Schätzung des nächsten, neueren Zustands nutzen.

Ein Zielzustand beinhaltet neben den Zustandsvariablen mit den Werten beispielsweise zu Ort oder Geschwindigkeit stets eine ihm zugeordnete Unsicherheit. Die Pflege und die Repräsentation eines Zielzustands erfolgt in der Regel mittels Vektoren und Matrizen. Dies erlaubt dem Filter sehr effizient die häufig und schnell wiederkehrenden Berechnungen mit den Methoden der linearen Algebra durchzuführen.

Für das hier betrachtete Problem der Verfolgung von Zielen mit Radarmessungen bieten sich die Kalman Filter an. Diese finden in der Praxis häufig in diesem Bereich Verwendung, sind aber aufgrund ihrer Eigenschaften auch in ganz anderen Bereichen vorzufinden. Es gibt daneben eine Vielzahl anderer Filter mit unterschiedlichen Merkmalen und Eigenschaften [C$^+$03].

Eine sehr einfache Variante eines Filters sind die α/β-Filter. Sie nutzen im Gegensatz zu den anderen hier genannten Filtern einen fest eingebauten und sehr einfachen Prozess [KM97]. Die sehr bekannten Kalman Filter (KF) sind optimale Schätzer für einen linearen Prozess, was sie für viele Probleme mit entsprechenden Einschränkungen sehr gut anwendbar macht [K$^+$60]. Ein linearer Prozess aus Sicht eines KF wäre in diesem Kontext beispielsweise eine gradlinige Bewegung. Der Extended Kalman Filter (EKF)

versucht die Einschränkungen der Linearität durch lokale Linearisierung zumindest abzuschwächen. In der Praxis zeigt der EKF für viele nicht-lineare Prozesse gute Schätzungen [SBMH16]. Eine weitere Evolutionsstufe in der Entwicklung der Kalman Filter stellt der Unscented Kalman Filter (UKF) dar [WVDM00]. Dieser nutzt ähnlich wie der Particle Filter (PF) ein Sampling-Verfahren, um den neuen Zustand zu schätzen.

Die nachfolgende Tabelle gibt einen Überblick von in der Literatur häufig anzutreffenden Filtern und ihren wichtigsten Eigenschaften.

Filter	Modell	Verfahren	Einschränkung
α/β	linear	Linearkombination mit fester Gewichtung	Normalverteilung
KF	linear	Linearkombination mit variabler Gewichtung	Normalverteilung
EKF	nicht-linear	wie KF, zusätzlich Linearisierung mittels Taylorreihen-Entwicklung	Normalverteilung
UKF	nicht-linear	Unscented Transform von Sigma Points in geringer und definierter Anzahl mit jeweils zwei Gewichten	Normalverteilung
PF	nicht-linear	Monte-Carlo-Sampling mit Particles in hoher bis sehr hoher Anzahl mit jeweils einem Gewicht	keine

Tabelle 2.1: Eigenschaften von verschiedenen Filtern im Vergleich

Der PF ist eine Quasi-Verallgemeinerung des UKF ohne die Voraussetzung bzw. die Annahme von normalverteilten Größen. Der Verzicht dieser Voraussetzung ist beim PF durch eine im Vergleich zum UKF sehr hohe Anzahl von Samples (Particles) erkauft, was je nach Dimensionalität des betrachteten Problems sehr rechenintensiv sein kann. Die Voraussetzung der Normalverteilung ist für das Problem der Zielverfolgung von physikalischen Objekten basierend auf Radar-Messungen in der Regel erfüllt. Entsprechend liegt im Verlauf der Arbeit, vor allem in der Evaluation, der Fokus auf dem UKF.

Kalman Filter

Die mathematische Grundlage für alle Filter der Kalman Filter-Familie bildet das Zustandsraummodell. Es beinhaltet eine Zustandsgleichung zur Beschreibung der Transition eines vorherigen Zustands in den nächsten Zustand und eine Beobachtungsgleichung, welche beschreibt, wie die Beobachtung in den Zustandsraum zu überführen ist.

$$\begin{aligned} x_k &= Fx_{k-1} + Gu_k + w_k \\ z_k &= Hx_k + v_k \end{aligned} \tag{2.1}$$

Die Matrizen F und G spiegeln die für die Transition von x_{k-1} in x_k anteiligen Gewichte dar. Die Transition von einem Zustand x_{k-1} in den Zustand x_k kann ein externes Signal u_k beinhalten, welches aber in dieser Arbeit keine Verwendung findet.

Das Prozessrauschen w_k dient zur Beschreibung der Unsicherheit des Prozesses. Ohne eine Unsicherheit in Form eines Prozessrauschens gäbe es keinen Grund für einen Filter, Messungen für die Verbesserung oder Korrektur einer Schätzung zu nutzen, da der Filter davon ausgehen würde, dass der Prozess perfekt ist. Aus diesem Grund muss es ein neben dem Bewegungsmodell zu modellierendes Rauschen im Prozess

geben. Für das Prozessrauschen gilt die starke Annahme der Normalverteilung. Im dem Fall, dass ein für ein Zustandsraummodell zugrunde gelegter Prozess keiner Normalverteilung folgt, sind die Schätzungen des Kalman Filter nicht verlässlich.

Die Beobachtungsmatrix H beschreibt zusammen mit dem Zustand x_k, in Verbindung mit dem Messrauschen v_k, die Beobachtungsgleichung für eine Beobachtung z_k. Das Messrauschen v_k ist integraler Bestandteil dieser Gleichung, da Beobachtungen (Messungen) grundsätzlich einen Rauschanteil mit sich führen. Für das Messrauschen gilt, wie für den Prozess, die Annahme der Normalverteilung.

Schätzung

Die folgenden Gleichungen enthalten in diskreter Form die Zustandsschätzung $\hat{x}_k$ und P_k aus der letzten Schätzung $\hat{x}_{k-1}$ sowie P_{k-1}. Die Kovarianzmatrix P_k drückt die Unsicherheit der Schätzung aus. Die Prozessmatrix F resultiert hier direkt aus dem zugrunde gelegten Bewegungsmodell. Die Addition der Prozessrauschverteilungsmatrix Q zu P_k modelliert die wachsende Unsicherheit, welche sich schrittweise und diskret erhöht, wenn Beobachtungen ausbleiben.

$$\begin{aligned} \hat{x}_k &= F\hat{x}_{k-1} + Gu_k \\ P_k &= FP_{k-1}F^T + Q \end{aligned} \tag{2.2}$$

Korrektur

Die Realisierung der Beobachtungsgleichung beinhaltet in diskreter und hier erweiterter Form sechs elementare Gleichungen. Im Rahmen einer Datenassoziierung haben die Schätzung einer Messung s_k und deren zugeordnete Kovarianz S_k zusammen mit der Innovation y_k einen unmittelbaren Nutzen für die Bildung eines Gate. Das Gate erlaubt in Folge eine statistische Aussage darüber, ob die aktuelle Messung zur Schätzung $\hat{x}_k$ passt oder nicht - also ob die Messung zu akzeptieren oder zu verwerfen ist.

$$\begin{aligned} s_k &= H\hat{x}_k \\ S_k &= (HP_kH^T + R)^{-1} \\ y_k &= z_k - s_k \\ K_k &= P_kH^TS_k \\ \hat{x}_k &= \hat{x}_k + K_k\, y_k \\ P_k &= (I - K_kH)P_k \end{aligned} \tag{2.3}$$

Das Kalman Gain K_k bestimmt als Gewichtung den Einfluss der Messung für die darauffolgende Korrektur der Schätzung $\hat{x}_k$ und P_k. Ein geringer Wert in K_k hat eine kleinere, ein großer Wert eine größere Korrektur der aktuellen Schätzung zur Folge. In diesem Zusammenhang spricht man häufig von "a priori" für die unkorrigierte und "a posteriori" für die korrigierte Schätzung [MDW95].

Lebenszyklus

Die folgende Abbildung zeigt die Verarbeitungsschritte des Kalman Filter für die Schätzung und Korrektur. In der Regel erfolgt die Schätzung wiederkehrend nach Ablauf eines definierten Zeitintervalls Δt, welches sich typischerweise auch in der Prozessmatrix F für die Transition der Zustände manifestiert.

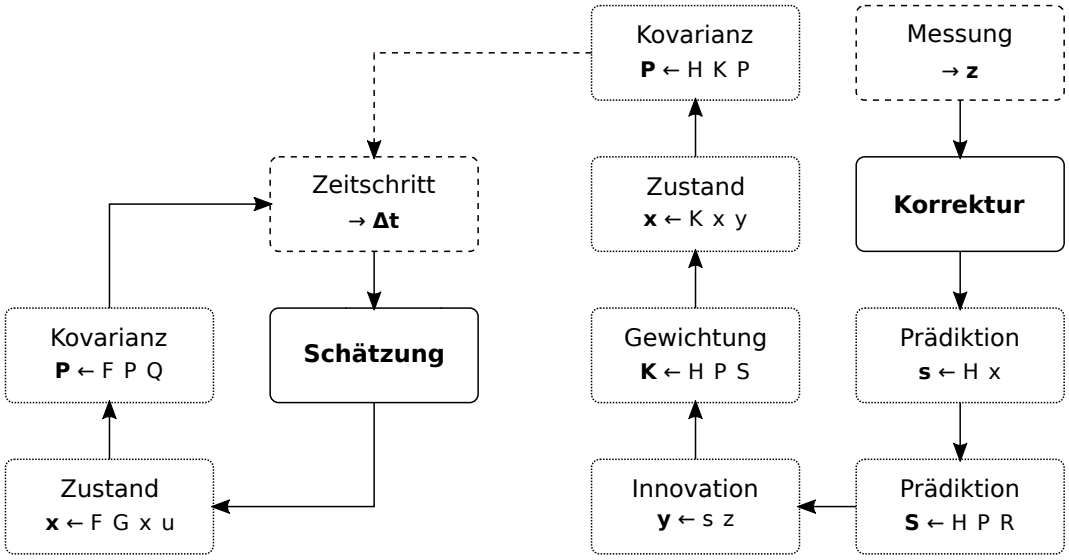


Abbildung 2.1: Kalman Filter - Darstellung der Verarbeitungsschritte

Der Treiber für die Schätzungen ist meist ein Zeitgeber. Die Korrektur ist hingegen in der Regel sehr viel seltener und auch unregelmäßiger aktiv, nämlich nur, wenn Messungen für eine mögliche Korrektur vorliegen.

Extended Kalman Filter

Der Extended Kalman Filter (EKF) ist eine Erweiterung des KF. Die Motivation des EKF liegt darin, auch nicht-lineare Prozesse, in diesem Kontext also beispielsweise Kurvenbewegungen, schätzen zu wollen. Die nicht-lineare Funktion $f(x_{k-1}, u_k)$ beschreibt hierbei den Prozess. Durch die Einführung der nicht-linearen Funktion $h(x_k)$ sind auch nicht-lineare Inkorporationen von Beobachtungen möglich.

$$\begin{aligned} x_k &= f(x_{k-1}, u_k) + w_k \\ z_k &= h(x_k) + v_k \end{aligned} \tag{2.4}$$

Häufig erzeugen die Funktionen $f(x_{k-1}$ und $h(x_k)$ einfach entsprechende Matrizen F und H, welche vor jeder Schätzung neu errechnet werden. Prinzipiell erzeugt diese Veränderbarkeit, vor allem der Prozessmatrix F, die Möglichkeit zur Schätzung von nicht-linearen Prozessen.

Die Grundlage für die Neuberechnung der Prozessmatrix F bilden partielle Ableitungen der Prozessgleichungen über den Zustandsvektor $\vec{x}$. Die daraus resultierenden Jacobi-Matrizen ermöglichen mit den Werten der jeweils aktuellen Schätzung $\vec{x}$ eine Approximation der Prozessmatrix F für die Transition des Zustands in der nächsten Schätzung.

Unscented Kalman Filter

Der Unscented Kalman Filter nutzt ein Sampling-Verfahren. Eine definierte Anzahl von Sigma Points, strategisch gut im Zustandsraum platziert, beschreiben dabei einen Zustand mit assoziierter Kovarianz. Die Unscented Transform (UT) dient in diesem Zusammenhang zur Wiederherstellung solcher Zustände und Kovarianzen anhand der Sigma Points.

Das Sampling-Verfahren des UKF macht in Folge ein Erfordernis des EKF wett: Der UKF transitiert die Sigma Points unmittelbar mit den Prozessgleichungen des Prozesses. Für die Berechnung der Prozessmatrix F sind keine partiellen Ableitungen notwendig.

Eine Mischung aus EKF und UKF ist auch denkbar: Die Schätzung erfolgen wie beim EKF, die Korrektur erfolgt mit der Unscented Transform. Derartige Mischformen sind während der Recherche aber nur vereinzelt aufgetreten.

Lebenszyklus

Der Lebenszyklus des UKF ist dem des KF bzw. EKF sehr ähnlich. Hinzu kommen die Schritte zur Erzeugung der Sigma Points X_i in einer Schätzung oder Korrektur sowie die Transition der Sigma Points.

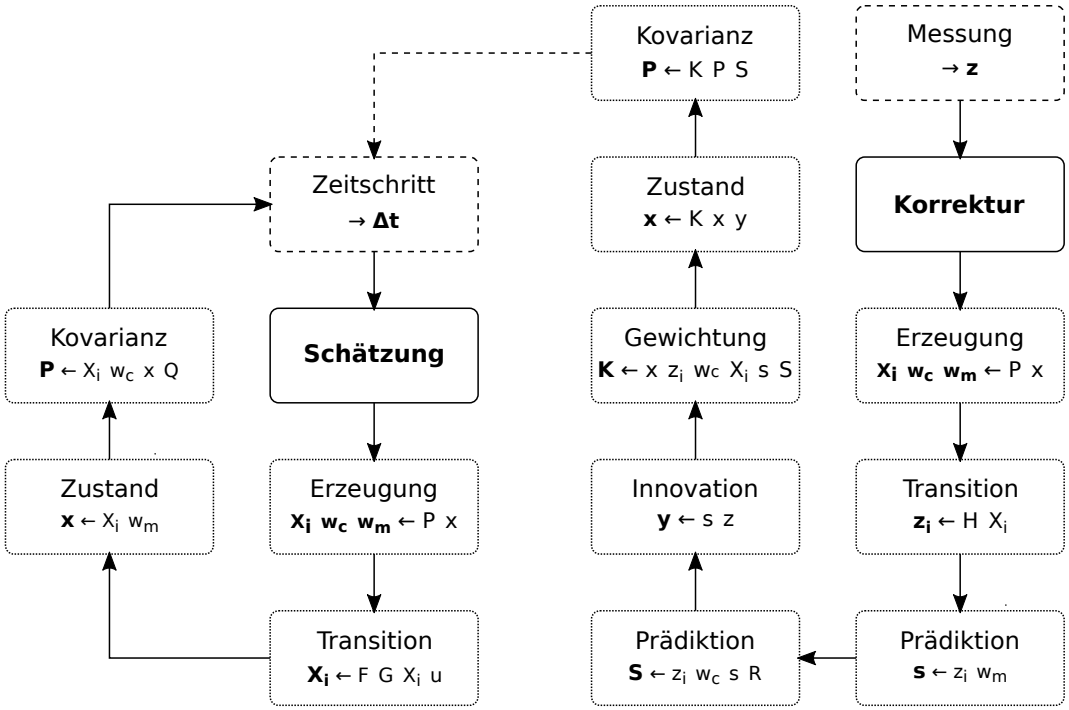


Abbildung 2.2: Unscented Kalman Filter - Darstellung des Verarbeitungsschritte

Besonderheit

Nachfolgend sind die Programmteile, welche den UKF charakterisieren und von anderen Filtern der Kalman Filter-Familie abgrenzen, umfassend beschrieben.

Die Erzeugung der Sigma Points ist aufgrund der Cholesky-Zerlegung der rechenintensivste Teil des UKF. Die Anwendung der Cholesky-Zerlegung ist möglich aufgrund des Wissens, dass die zerlegte Kovarianzmatrix Σ (hier gemäß Zustands- bzw. Beobachtungsgleichung P und S) eine symmetrisch positiv definite Matrix sein muss.

Der UKF ist bezogen auf den Rechenaufwand im Vergleich mit dem EKF aufgrund dessen höheren Komplexität für die Berechnung der Prozessmatrix F aus den partiell abgeleiteten Prozessgleichungen ähnlich.

```
in:  μ  Vektor mit n Komponenten
     Σ  Kovarianzmatrix zu μ
     α  Ausbreitungsskalierung
     κ  = 0
     β  = 2
out: X_i Sigma Points
     w_m Gewichte der Sigma Points für μ
     w_c Gewichte der Sigma Points für Σ
begin
  λ = α²(n+κ) − n
  erzeuge Vektoren w_m und w_c mit jeweils 2n+1 Komponenten
  initialisiere w_m und w_c mit 1/(2(n+λ)) für alle Komponenten
  w_m[0] ← λ/(n+λ)
  w_c[0] ← w_m[0]+(1−α²+β)
  erzeuge Matrix X_i bestehend aus 2n+1 Spalten und n Zeilen
  X_i[0] ← μ
  C ← erzeuge Quadratwurzel von (n+λ)Σ mittels Cholesky−Zerlegung
  for k in 1 to n:
    X_i[k] ← μ + C^T[k−1]
    X_i[n+k] ← μ − C^T[k−1]
  return X_i, w_m, w_c
end
```

Abbildung 2.3: Erzeugung von Sigma Points und Gewichtungen

```
in:  X_i Sigma Points
     w_m Gewichte der Sigma Points für μ
out: μ  Vektor mit n Komponenten
begin
  m ← Anzahl der Spalten in X_i
  n ← Anzahl der Zeilen in X_i
  erzeuge Nullvektor μ bestehend aus n Komponenten
  for k in 0 to m−1:
    μ ← μ + (X_i[k] w_m[k])
  return μ
end
```

Abbildung 2.4: Unscented Transform - Rekonstruktion eines Zustands

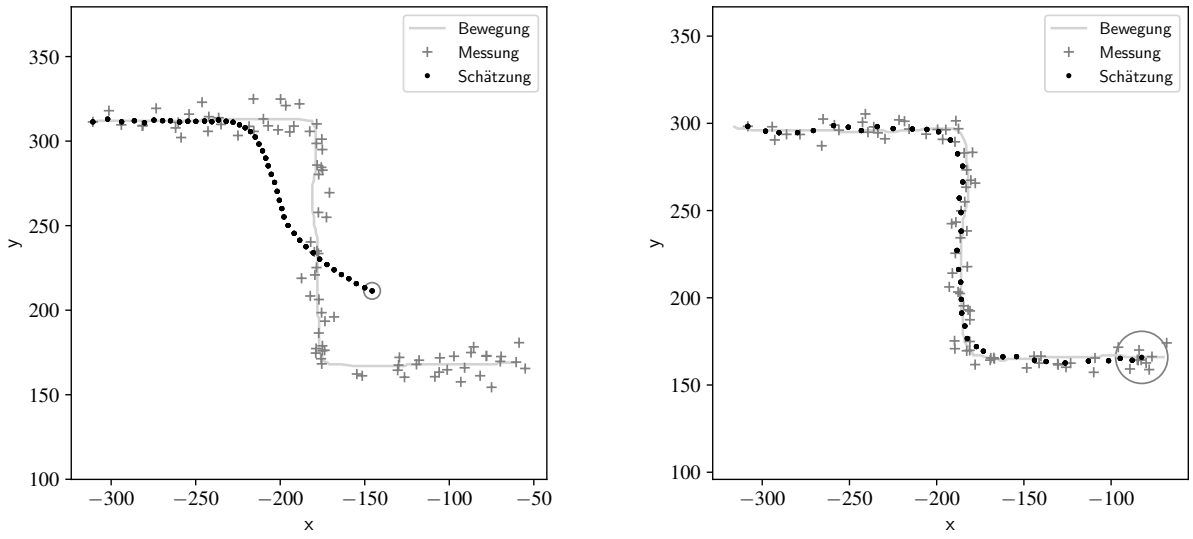
```
in:  X_i Sigma Points
     w_c Gewichte der Sigma Points für Σ
     μ  Vektor mit n Komponenten (x bzw. s)
     M  Rauschverteilungsmatrix (Q bzw. R)
out: Σ  Kovarianz (P bzw. S)
begin
  m ← Anzahl der Spalten in X_i
  erzeuge Nullmatrix Σ bestehend aus n Spalten und n Zeilen
  for k in 0 to m−1:
    σ ← X_i[k]−μ
    Σ ← Σ + (σ w_c[k] σ^T)
  return Σ
end
```

Abbildung 2.5: Unscented Transform - Rekonstruktion einer Kovarianz

Prozessrauschen

Das Prozessrauschen ist eine wichtige Stellschraube beim Einsatz von Filtern. Im Filter ist das Prozessrauschen in einer Prozessrauschverteilungsmatrix Q hinterlegt. Der Effekt unterschiedlicher Intensitäten im Prozessrauschen ist in der nachfolgenden Abbildung illustriert.

Zwei Sensoren mit einem Sensorrauschen von $\sigma_x = \sigma_y = 5$ und einer Abtastrate von 200 ms verfolgen die Bewegung. Die Schätzung des Filters erfolgt alle 40 ms. Insgesamt ist eine Bewegung über einen Zeitraum von 8 s zu sehen.



(a) geringe Intensität ($q = 100$)

(b) hohe Intensität ($q = 1000$)

Abbildung 2.6: Vergleich von niedriger und hoher Intensität beim Prozessrauschen

Bei geringem Prozessrauschen ist der Filter "prozessgläubiger" und entwickelt Vorhersagen ohne großen Einfluss der Messungen. Dies erzeugt eine sehr starke Glättung der tatsächlichen Bewegung und in Folge einen großen Unterschied zu der zu verfolgenden Bewegung.

Bei hohem Prozessrauschen hat der Filter weniger Vertrauen in den Prozess, entsprechend erfolgt die Messung fast ausschließlich auf Grundlage der einfallenden Messungen. Dieses im ersten Augenblick wünschenswerte Verhalten stößt an Grenzen, wenn Messungen sehr verrauscht oder sehr unregelmäßig einfallen. Zudem ist der Zweck eines Modells bei zu hohem Prozessrauschen in Frage zu stellen, da dieser dann kaum noch Einfluss auf die Schätzungen nimmt.

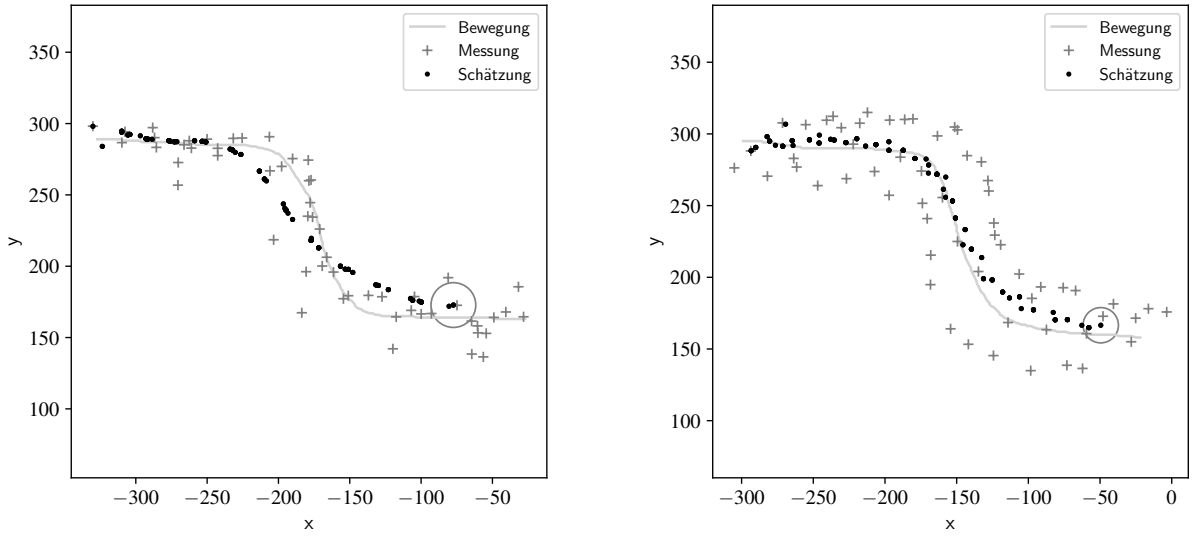
Eine falsche Modellierung des Prozessrauschens oder eine fehlerhafte Parametrisierung hat in der Regel einen großen und negativen Einfluss auf die Qualität der Schätzungen.

2.2 Datenfusion

Im Rahmen der Datenfusion ist zwischen Sensor-Track-Fusion und Track-Track-Fusion zu unterscheiden. Die Sensor-Track-Fusion kommt zum Tragen, wenn Messungen von mehreren Sensoren zu einem Ziel zur Verfügung stehen und in die Zustandsschätzung des Ziels (in dessen Track) einfließen sollen. Die Track-Track-Fusion erzeugt aus den Schätzungen $\vec{x}$ und P verschiedener Quellen einen "gemeinsamen" Track. Sie kommt in der Regel dann zum Einsatz, wenn mehrere Sensor-Tracks in einen gemeinsamen System-Track zu überführen sind.

Sensor-Track-Fusion

Drei Sensoren mit einer Abtastrate von 300 ms, 500 ms und 700 ms verfolgen in der nachfolgenden Abbildung eine Bewegung bei mittlerem Prozessrauschen. Im ersten Beispiel haben die Sensoren ein deutlich unterschiedliches Sensorrauschen, aber keine permanenten Messfehler (Bias). Im zweiten Beispiel haben alle Sensoren ein geringes Messrauschen, dafür zwei davon einen deutlichen Bias.



(a) kein Bias, mäßiges Messrauschen

(b) deutlicher Bias, geringes Messrauschen

Abbildung 2.7: Sensordatenfusion am Beispiel mit drei Sensoren

Trotz der zeitlich sehr unterschiedlich einfallenden Messungen und den großen qualitativen Unterschieden konnte der Filter in beiden Beispielen Schätzungen durchführen, welche in der Summe die tatsächliche Bewegung in akzeptablem Maß beschreiben. Die Grundlage dafür bildet die Möglichkeit der Multiplikation zweier Gaußscher Funktionen, welche über die Parameter μ und σ^2 vollständig beschrieben sind [Far12].

$$\mu = \frac{\mu_1 \sigma_2^2 + \mu_2 \sigma_1^2}{\sigma_1^2 + \sigma_2^2} \qquad \sigma = \sqrt{\frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2}} \tag{2.5}$$

Dies erlaubt die Verschmelzung einer Messung μ_1 von Sensor a mit dem Sensorrauschen σ_1^2 mit einer anderen Messung μ_2 des Sensors b mit dem Sensorrauschen σ_2^2. Dieser Ansatz ist den hier ausgewählten Filtern inhärent; er wird bei jedem Durchlauf einer Korrektur durchgeführt.

Track-Track-Fusion

Die Track-Track-Fusion ist dann notwendig, wenn bereits erstellte Tracks und keine Messungen zur Verfügung stehen. Für den Fall unkorrelierter Kovarianzen P_j und P_j ist eine Fusion zur Bestimmung einer gemeinsamen Schätzung $\hat{x}$ und P durch einfache Linearkombination möglich [CMBC00].

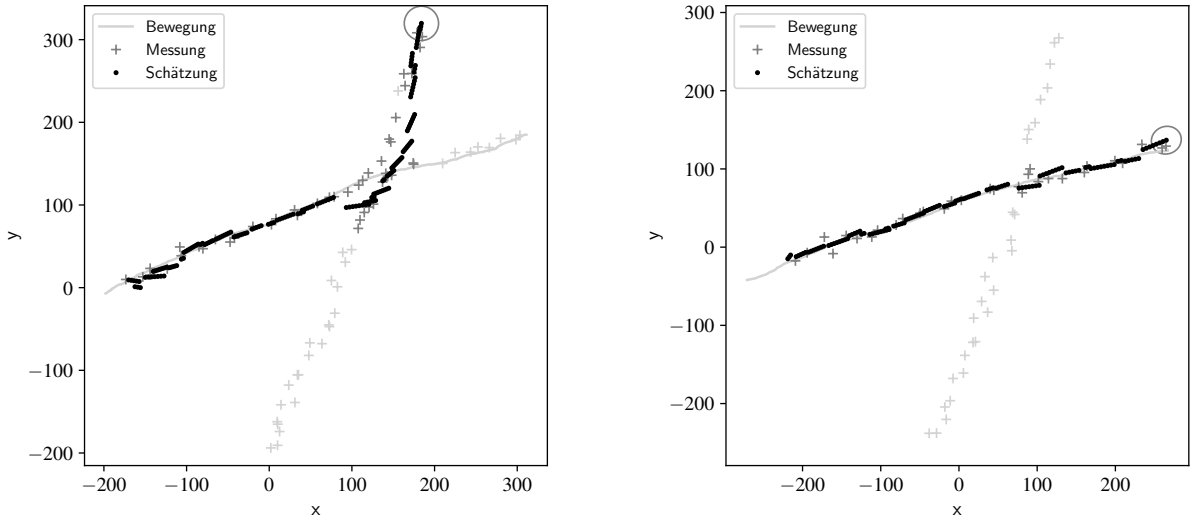
$$\begin{aligned} P &= (P_i^{-1} + P_j^{-1})^{-1} \\ \hat{x} &= P(P_i^{-1}\hat{x}_i + P_j^{-1}\hat{x}_j) \end{aligned} \tag{2.6}$$

2.3 Datenassoziierung

Ein wichtiger Teil bei der Verarbeitung von Messungen ist die Datenassoziierung. Im Fall eines Radars enthält eine Messung einen Zeitstempel, die eigene Position, das vermutete Messrauschen und die eigentliche Messung zu Azimut, Elevation und Entfernung - aber keine Zuordnung zu einem Ziel. Deshalb führt ein Filter bei Erhalt einer Messung eine Prüfung durch, ob die Messung für ihn interessant ist.

Messungen, welche nicht zu dem verfolgten Ziel gehören, aber in die Korrektur einfließen, können eine Schätzung deutlich verschlechtern. Dies hat einen kaskadierenden Effekt: Die Datenassoziierung kann aufgrund des "Abdriftens" der Schätzung vom tatsächlichen Ziel andere Messungen, welche eigentlich zu anderen Zielen gehören, als interessant ansehen und in die Schätzung einfließen lassen. Datenassoziierung ist insofern ein sehr wichtiges Element im Filter. In diesem Zusammenhang spielt auch das Prozessrauschen eine maßgebliche Rolle: Bei sehr hohem Rauschen kann eine Datenassoziierung in der Regel keine guten Ergebnisse erzielen, da der Prozess aus Sicht des Filter unklar ist. Eine verlässlich Vorhersage, ob eine Messung zu einer Schätzung zu assoziieren ist, fällt mit steigendem Prozessrauschen aus Sicht der Datenassoziierung immer schwerer.

Das Aufsetzen auf ein Falschziel in Folge falscher Datenassoziierung ist in der nachfolgenden Abbildung im ersten Beispiel zu sehen. Ein Gegenbeispiel zeigt eine funktionierende Datenassoziierung, wodurch die Schätzungen des Filters weiterhin auf dem korrekten Ziel beruhen.



(a) Falschziel übernommen

(b) Echtziel erhalten

Abbildung 2.8: Datenassoziierung am Beispiel mit zwei Zielen

Die Mahalanobis-Distanz ist ein statistisches Distanzmaß für mehrdimensionale Vektorräume. Sie dient beim Einfall einer Messung zur Überprüfung, ob die Innovation y_k skaliert durch die Kovarianz S_k aus der Schätzung der Messung unterhalb einer Schranke liegt [ABE+]. Diese Schranke ergibt sich aus der chi^2-Verteilung, hier mit einem Signifikanzniveau von $\alpha = 5\%$, bei zwei bzw. drei Freiheitsgraden df.

$$d = \sqrt{\|v-w\|^T \Sigma^{-1} \|v-w\|} = \sqrt{y_k^T S_k^{-1} y_k}$$
$$\chi^2_{\alpha=0,05;df=2} = 5,99 \qquad \chi^2_{\alpha=0,05;df=3} = 7,81 \tag{2.7}$$

Die Freiheitsgrade ergeben sich aus der Dimensionalität der Messung: Für in ein kartesisches System umgerechnete Radar-Messungen gilt $df = 3$. Ein Spezialfall der Mahalanobis-Distanz entsteht bei $\Sigma = I$, wo die Mahalanobis-Distanz keine Skalierung erfährt und sich wie der Euklidische Abstand verhält.

3. Modellierung

Bewegungsmodelle dienen den Filtern als Prozessbeschreibung zur Schätzung eines Zielzustands. Die Bewegung eines Ziels ist in der Regel nicht vorhersehbar, entsprechend erfolgen Annahmen bei der Modellierung einer Bewegung.

Eine Möglichkeit besteht darin, nur die sehr wahrscheinlichen oder häufig vorkommenden Bewegungsanteile zu modellieren, beispielsweise wenn bekannt ist, dass ein Ziel sich im größten Teil seiner Bewegung mit konstanter Geschwindigkeit und ohne Änderung der Richtung bewegt. In diesem Ansatz überprüft der Filter anhand einer ihm zur Verfügung gestellten Menge von Modellen fortlaufend, ob das eine oder andere Modell besser passt und wechselt es für die nächsten Schätzungen aus [Gen01, JK13].

Auf der anderen Seite besteht die Möglichkeit, eine komplexe Bewegung mit einem stark vereinfachten Modell zu beschreiben und passable Schätzungen durch eine Erhöhung des Prozessrauschens zu erzielen. Dies birgt den Nachteil, dass zum einen die Schätzungen ebenfalls verrauscht (ungenau) sind, da sie zum größten Teil auf den verrauschten Messungen beruhen, und zum anderen, dass eine verlässliche Datenassoziierung in einem Umfeld mit mehreren Zielen erschwert wird.

Eine Analyse in Frage kommender Bewegungsmodelle für die betrachteten Filter erfolgt in diesem Abschnitt mit dem Ziel, ein möglichst vielseitig einsetzbares Bewegungsmodell für einen Tracker abzuleiten.

3.1 Analyse

Brownian Motion

Das Brownian Motion Model (BM) ist streng genommen kein Bewegungsmodell, ist aber in der Literatur durch seinen Namen als solches zu klassifizieren. Es bietet durch seine Einfachheit keine Möglichkeit der Vorhersage einer Bewegung, da lediglich der Ort mit den Zustandsvariablen x und y im Zustand $\vec{x}_{BM}$ enthalten ist.

Die Transition von einem in den nächsten Zustand beinhaltet nur die Übernahme des alten Zustands für den neuen Zustand. Eine Änderung des Zustands kann folglich nur durch Messungen im Rahmen der Korrektur erfolgen.

$$\vec{x}_{BM} = \begin{pmatrix} x \\ y \end{pmatrix} \qquad \vec{x}_{BM}(t + \Delta t) = \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.1}$$

$$F_{BM} = I_2 \tag{3.2}$$

Dieses Modell kommt dann zum Einsatz, wenn lediglich Positionen mit einer gewissen, durch das Prozessrauschen einstellbaren, Glättung zu schätzen sind.

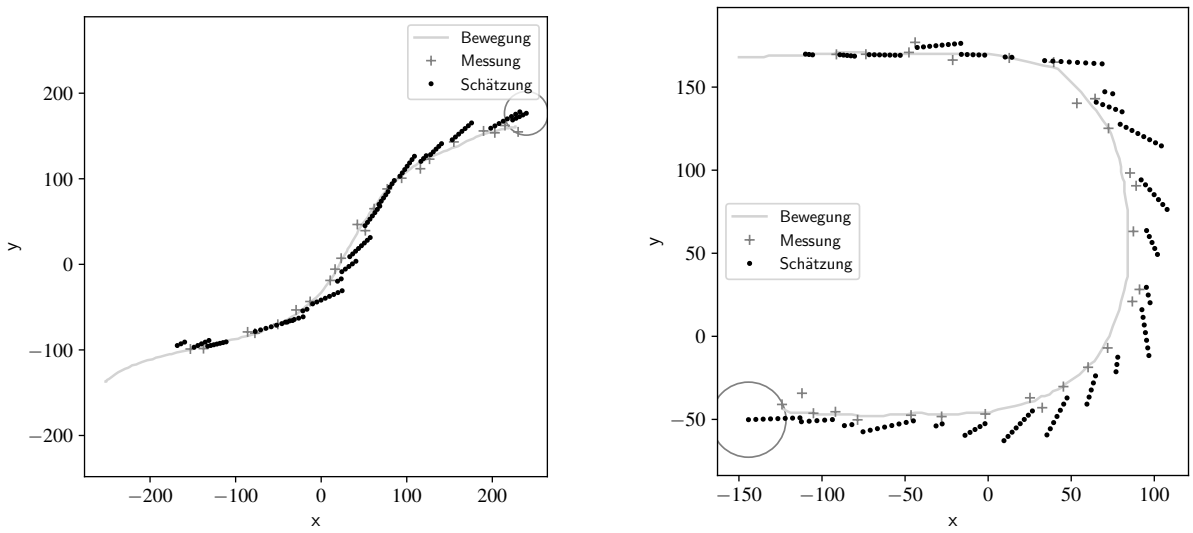
Constant Velocity

Dieses Modell beinhaltet neben den Zustandsvariablen x und y für den Ort die Zustandsvariablen v_x und v_y für eine Geschwindigkeit [SRW08]. Das Modell bietet somit die Möglichkeit der Schätzung einer Geschwindigkeit und daraus resultierend die Möglichkeit der Vorhersage eines Ortes anhand der Geschwindigkeit nach Ablauf einer Zeit Δt.

$$\vec{x}_{CV} = \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix} \qquad \vec{x}_{CV}(t + \Delta t) = \begin{pmatrix} x + v_x \Delta t \\ y + v_y \Delta t \\ v_x \\ v_y \end{pmatrix} \tag{3.3}$$

$$F_v = \begin{pmatrix} \frac{(\Delta t)^1}{1!} & 0 \\ 0 & \frac{(\Delta t)^1}{1!} \end{pmatrix} \qquad F_{CV} = \left(\begin{array}{c|c} I_2 & F_v \\ \hline 0_2 & I_2 \end{array}\right) \tag{3.4}$$

Die Schätzung des nächsten Zustands $\vec{x}_{CV}(t + \Delta t)$ erfolgt im Constant Velocity Modell (CV) unter Bezugnahme der zuletzt geschätzten Geschwindigkeit. In der folgenden Abbildung ist das Verhalten des CV für zwei Bewegungen illustriert.



(a) gradlinige Bewegung

(b) kurvige Bewegung

Abbildung 3.1: Constant Velocity; Verfolgung zweier Cursorbewegungen

Im ersten Beispiel ermöglicht das CV gute Schätzungen, da die Bewegung fast linear (gradlinig) ist. Die große Schwäche des CV liegt in der Beschreibung von Kurvenbewegungen bzw. im Allgemeinen von nicht-linearen Bewegungen, wie das zweite Beispiel zeigt. Die Schätzungen verlaufen stets in Richtung der zuletzt geschätzten Geschwindigkeit. Bei Einfall einer Messung korrigiert der Filter die Geschwindigkeit und den Ort im Zustand, worauf die Schätzungen bis zur nächsten Messung beruhen. Entsprechend sind bei nicht-linearen Bewegungen mit dem CV grundsätzlich keine belastbaren Vorhersagen zu einer Zielbewegung möglich.

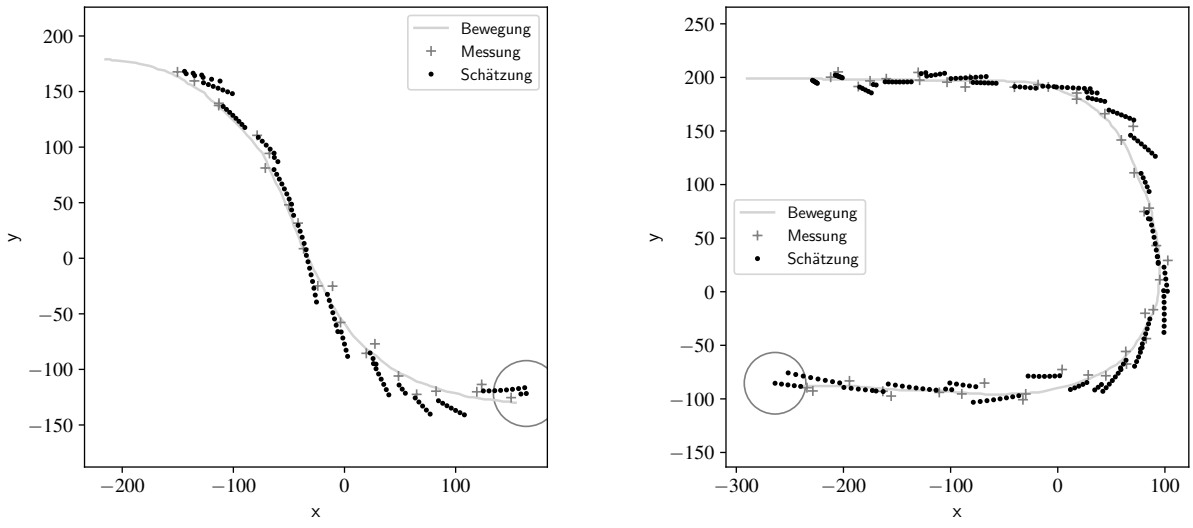
Constant Acceleration

Das Constant Acceleration Modell (CA) erweitert das CV um die zweite Ableitung des Ortes nach der Zeit: Beschleunigung [SRW08]. Für den hier betrachteten, zweidimensionalen Fall hat das Model bereits einen Zustandsvektor $\vec{x}_{CA}$ sowie eine Prozessmatrix F_{CA} mit sechs Zustandsvariablen. Bei der Transition, beschrieben durch $\vec{x}_{CA}(t + \Delta t)$, erfolgt die Integration der zuletzt geschätzten Beschleunigung in den neuen Ort sowie in die neue Geschwindigkeit.

$$\vec{x}_{CA} = \begin{pmatrix} x \\ y \\ v_x \\ v_y \\ a_x \\ a_y \end{pmatrix} \qquad \vec{x}_{CA}(t+\Delta t) = \begin{pmatrix} x + v_x\Delta t + a_x\frac{(\Delta t)^2}{2} \\ y + v_y\Delta t + a_y\frac{(\Delta t)^2}{2} \\ v_x + a_x\Delta t \\ v_y + a_y\Delta t \\ a_x \\ a_y \end{pmatrix} \tag{3.5}$$

$$F_a = \begin{pmatrix} \frac{(\Delta t)^2}{2!} & 0 \\ 0 & \frac{(\Delta t)^2}{2!} \end{pmatrix} \qquad F_{CA} = \left(\begin{array}{c|c|c} I_2 & F_v & F_a \\ \hline 0_2 & I_2 & F_v \\ \hline 0_2 & 0_2 & I_2 \end{array}\right) \tag{3.6}$$

Durch die Integration der Beschleunigung in das CA sind im Vergleich zum CV bessere Ergebnisse bei der Schätzung von gradlinigen und kurvigen Bewegungen möglich, wie das erste und zweite Beispiel der nachfolgenden Abbildung zeigt. Dies gilt im Allgemeinen auch für Geschwindigkeitsänderungen (hier nicht gezeigt) während der Bewegung.



(a) gradlinige Bewegung

(b) kurvige Bewegung

Abbildung 3.2: Constant Acceleration; Verfolgung zweier Cursorbewegungen

Coordinate Turn

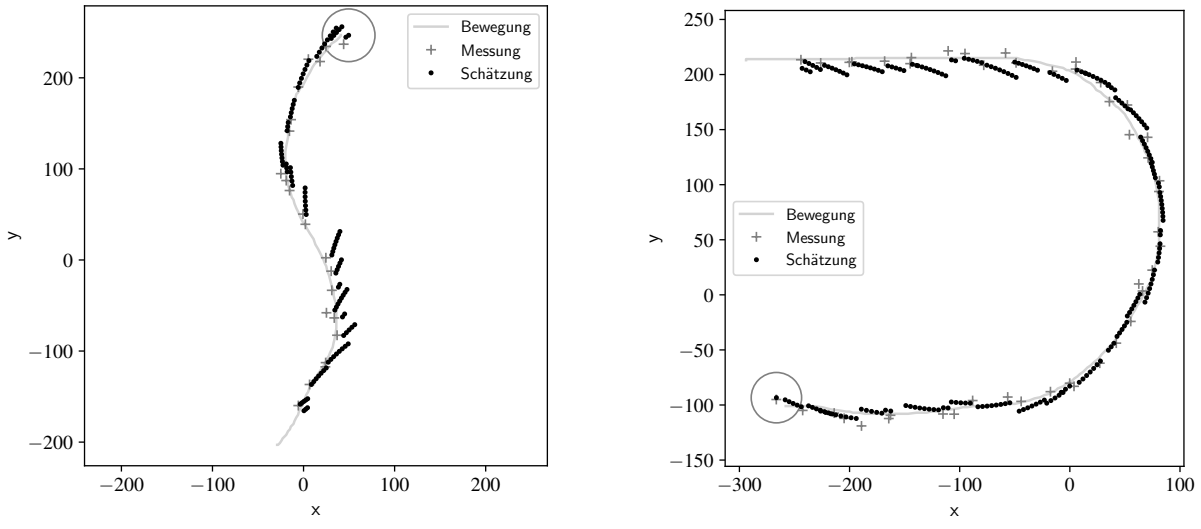
Die Modellierung von kurvigen Abschnitten erfordert für die Verbesserung der Schätzungen eine Kenntnis über die Richtungsänderung. Im einfachsten Fall ist diese bekannt, was einen festen Einbau einer Drehrate ω in das Modell erlaubt [RHG14]. Das Coordinate Turn Modell (CT) verhält sich für den Fall $\omega = 0$ wie ein CV, da dann $F_{CT} = F_{CV}$ gilt.

$$\vec{x}_{CT} = \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix} \qquad \vec{x}_{CT}(t+\Delta t) = \begin{pmatrix} x + \frac{v_x}{\omega} sin(\omega\Delta t) + \frac{v_y}{\omega}(1 - cos(\omega\Delta t)) \\ y - \frac{v_x}{\omega}(1 - cos(\omega\Delta t)) + \frac{v_y}{\omega} sin(\omega\Delta t) \\ v_x cos(\omega\Delta t) + v_y sin(\omega\Delta t) \\ -v_x sin(\omega\Delta t) + v_y cos(\omega\Delta t) \end{pmatrix} \tag{3.7}$$

$$F_{CT} = \begin{pmatrix} 1 & 0 & \frac{sin(\omega\Delta t)}{\omega} & \frac{-cos(\omega\Delta t)+1}{\omega} \\ 0 & 1 & \frac{-(-cos(\omega\Delta t)+1)}{\omega} & \frac{sin(\omega\Delta t)}{\omega} \\ 0 & 0 & cos(\omega\Delta t) & sin(\omega\Delta t) \\ 0 & 0 & -sin(\omega\Delta t) & cos(\omega\Delta t) \end{pmatrix} \tag{3.8}$$

Die nachfolgende Abbildung zeigt das Verhalten eines CT mit $\omega = 30°/s$. Das erste Beispiel zeigt deutlich im unteren Teil der Bewegung die Tendenz des Modells in Richtung der vorgegebenen Drehrate ω. Im zweiten Teil der Bewegung beschreibt das Modell die Bewegung sehr gut.

Der größte Vorteil und auch gleichzeitig Nachteil des CT ist in dem zweiten Beispiel zu sehen: Die vorgegebene Drehrate ω erlaubt hervorragende Schätzungen in dem Kurvenabschnitt, aber sehr schlechte bei den gradlinigen Bewegungen.



(a) gradlinige Bewegung

(b) kurvige Bewegung

Abbildung 3.3: Coordinate Turn mit $\omega = +30°/s$; Verfolgung zweier Cursorbewegungen

Das CT eignet sich im Rahmen eines IMM zur Modellierung von unterschiedlichen Bewegungsabschnitten durch Vorgabe unterschiedlicher Drehraten. Dabei ergänzen im Regelfall mehrere CT ein CV oder CA. Der Filter entscheidet dabei, welches Modell er für die (bestmögliche) Schätzung nutzt.

Constant Turn Rate Velocity

Der große Nachteil einer festen Drehrate wie beim CT umgeht das Constant Turn Rate Velocity (CTRV), indem dieses die Drehrate ω in den Zustandsvektor integriert [SBMH16]. Das Modell verliert dadurch seinen linearen Charakter, wodurch es nicht mit der einfachen Variante des Kalman Filter zu verwenden ist. Das CTRV setzt die Nutzung eines Filters voraus, der auch nicht-lineare Prozesse schätzen kann. Dafür kommen in dieser Arbeit der EKF oder UKF in Frage.

$$\vec{x}_{CTRV} = \begin{pmatrix} x \\ y \\ v \\ \phi \\ \omega \end{pmatrix} \qquad \vec{x}_{CTRV}(t+\Delta t) = \begin{pmatrix} x + \frac{2v}{\omega} sin(\frac{\omega\Delta t}{2}) sin(\phi + \frac{\omega\Delta t}{2}) \\ y + \frac{2v}{\omega} sin(\frac{\omega\Delta t}{2}) cos(\phi + \frac{\omega\Delta t}{2}) \\ v \\ \phi + \omega\Delta t \\ \omega \end{pmatrix} \tag{3.9}$$

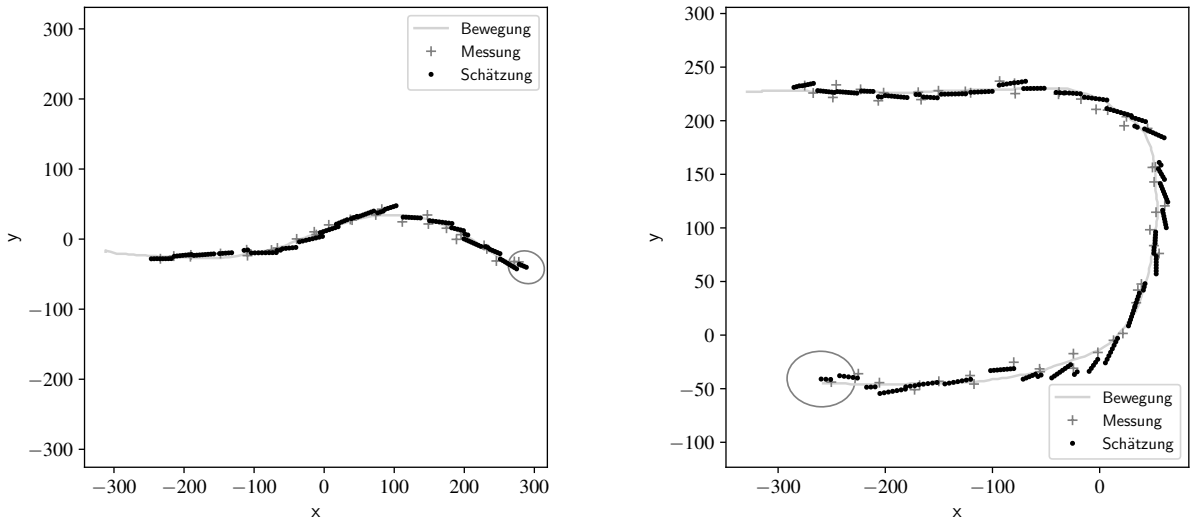
Die Konstruktion der nicht-linearen Prozessmatrix F_{CTRV} erfolgt in Abhängigkeit von ω für zwei mögliche Fälle, um die Transition des Zustands $\vec{x}_{CTRV}$ für gradlinige und kurvige Bewegungen zu ermöglichen.

$$\begin{aligned} c_\omega^a &= \frac{sin(\frac{\omega\Delta t}{2}) sin(\phi + \frac{\omega\Delta t}{2})}{\omega} & c_\omega^b &= \frac{sin(\frac{\omega\Delta t}{2}) cos(\phi + \frac{\omega\Delta t}{2})}{\omega} \\ c_\omega^c &= \frac{cos(\frac{\omega\Delta t}{2}) sin(\phi + \frac{\omega\Delta t}{2})}{\omega} & c_\omega^d &= \frac{cos(\frac{\omega\Delta t}{2}) cos(\phi + \frac{\omega\Delta t}{2})}{\omega} \end{aligned} \tag{3.10}$$

$$F_{CTRV} = \begin{cases} \begin{pmatrix} 1 & 0 & 2c_\omega^a & 2vc_\omega^b & v\Delta t c_\omega^b + v\Delta c_\omega^c - 2v\frac{c_\omega^a}{\omega} \\ 0 & 1 & 2c_\omega^b & -2vc_\omega^a & -v\Delta t c_\omega^a + v\Delta c_\omega^d - 2v\frac{c_\omega^b}{\omega} \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, & \text{wenn } \omega \neq 0. \\ \begin{pmatrix} 1 & 0 & sin(\phi)\Delta t & cos(\phi)v\Delta t & 0 \\ 0 & 1 & cos(\phi)\Delta t & -sin(\phi)v\Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, & \text{sonst.} \end{cases} \tag{3.11}$$

Das CTRV ist häufig für die Modellierung von zweidimensionalen Bewegungen und im Fahrzeugbereich anzutreffen. Es bildet eine wichtige Grundlage für den im Abschluss dieses Abschnitts vorgestellten, dreidimensionalen Ansatz.

Die folgende Abbildung zeigt die CTRV-Schätzungen für eine gradlinige und kurvige Bewegung. Es ist zu erkennen, dass das CTRV durch die Integration der Drehrate ω die Vorteile des CV und CT teilweise vereint. Dafür verliert es seinen linearen Charakter und macht es nur noch mit nicht-linearen Filtern einsetzbar.



(a) gradlinige Bewegung

(b) kurvige Bewegung

Abbildung 3.4: Constant Turn Rate Velocity (EKF); Verfolgung zweier Cursorbewegungen

Constant Turn Rate Acceleration

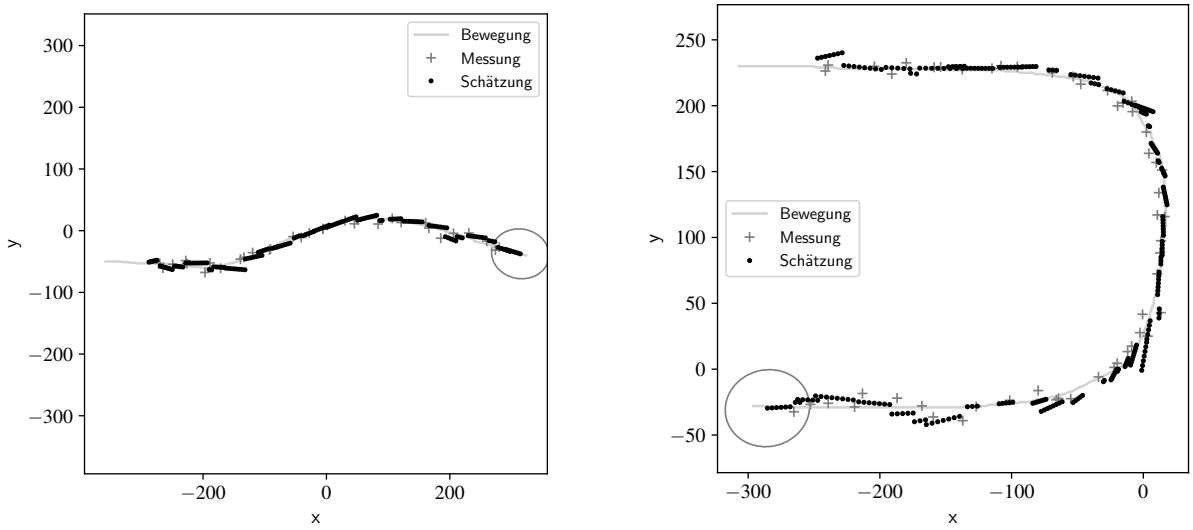
Das Constant Turn Rate Acceleration (CTRA) ist eine Erweiterung des CTRV. Wie bei der Erweiterung des CV zum CA erfolgt die Erweiterung durch die Integration einer weiteren Zustandsvariablen für die Beschleunigung [TPA05].

$$\vec{x}_{CTRA} = \begin{pmatrix} x \\ y \\ v \\ a \\ \phi \\ \omega \end{pmatrix} \qquad \vec{x}_{CTRA}(t+\Delta t) = \begin{pmatrix} x + \frac{1}{\omega^2}((-v\omega - a\omega\Delta t)cos(\phi\omega\Delta t) \\ ... + sin(\phi + \omega\Delta t)a + cos(\phi)\omega v \\ ... - sin(\phi))a \\ y + \frac{1}{\omega^2}((v\omega + a\omega\Delta t)sin(\phi\omega\Delta t) \\ ... + cos(\phi + \omega\Delta t)a - sin(\phi)\omega v \\ ... - cos(\phi))a \\ v + a\Delta t \\ a \\ \phi + \omega\Delta t \\ \omega \end{pmatrix} \tag{3.12}$$

Die partielle Ableitung der Prozessgleichung $\vec{x}_{CTRA}(t+\Delta t)$ über den Zustandsvektor $\vec{x}_{CTRA}$ erzeugt eine sehr umfangreiche Jacobi-Matrix F_{CTRA}. Auf die explizite Darstellung wird an dieser Stelle verzichtet.

$$F_{CTRA} = \frac{\delta}{\delta x_{k-1}} f(x_{k-1}, u_k) = \frac{\delta}{\delta \vec{x}_{CTRA}} \vec{x}_{CTRA}(t+\Delta t) \tag{3.13}$$

Das CTRA zeigt in der folgenden Abbildung ein ähnliches Verhalten wie das CTRV. Die bessere Adaption an eine sich verändernde Geschwindigkeit ist hier nicht dargestellt, stellt aber einen Vorteil des CTRA gegenüber dem CTRV dar.



(a) gradlinige Bewegung

(b) kurvige Bewegung

Abbildung 3.5: Constant Turn Rate Acceleration (UKF); Verfolgung zweier Cursorbewegungen

3.2 Ansatz

Die bis hier vorgestellten Modelle gelten für den zweidimensional Fall, also zur Beschreibung einer Bewegung auf einer Ebene. Die linearen Modelle CV und CA sind prinzipiell ohne Probleme und mit wenig Aufwand auf den dreidimensionalen Fall erweiterbar, indem an entsprechender Stelle im Zustandsvektor eine weitere Zustandsvariable für die dritte Dimension hinzugefügt wird.

Für das in dieser Arbeit betrachtete Problem der Zielverfolgung im Raum erfolgt eine Vorstellung zweier Bewegungsmodelle für den dreidimensionalen Fall. Diese Modell sind für ein rechtshändiges Koordinatensystem (x-Achse: Daumen, y-Achse: Zeigefinger, z-Achse: Mittelfinger) mit mathematisch negativem Drehsinn (im Uhrzeigersinn) konzipiert.

Die Idee für diesen Ansatz lieferte die Analyse des CTRV und CTRA. Die Grundlage der Überlegung, das CTRV und CTRA für den dreidimensionalen Fall zu adaptieren, fußt auf Rotationsmatritzen [Die06].

$$R_y(\alpha) = \begin{pmatrix} cos(\alpha) & 0 & sin(\alpha) \\ 0 & 1 & 0 \\ -sin(\alpha) & 0 & cos(\alpha) \end{pmatrix} \qquad R_z(\alpha) = \begin{pmatrix} cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.14}$$

$$R = R_z\, R_y$$

Die Lage und Orientierung (die Haltung) eines Ziels ist im dreidimensionalen Raum durch drei Ortskoordinaten und drei Verdrehwinkel beschreibbar. Radar-Messungen können grundsätzlich die drei Ortskoordinaten zu einem Ziel liefern, die Verdrehwinkel des Objekts selber sind mit Radaren kaum messbar. Die Bewegung des Objekts ist aber über die Zeit mit zwei Winkeln beschreibbar: Neigung (pitch) und Gieren (yaw).

Ein Vergleich der beiden hier entwickelten und nachfolgend vorgestellten Modelle erfolgt mit dem CV und CA im Rahmen der Evaluation.

Constant Yaw Rate Pitch Rate Velocity

Das Constant Yaw Rate Pitch Rate Velocity Modell (CYRPRV) beinhaltet im Zustandsvektor drei Ortskoordinaten x, y und z, und den Betrag der Geschwindigkeit v, welcher zusammen mit den Winkeln Φ für das Gieren und Θ für die Neigung einen Geschwindigkeitsvektor beschreibt. Die Gierrate ϕ und eine Neigungsrate θ vervollständigen den Zustandsvektor.

Nachfolgend sind der Zustandsvektor $\vec{x}_{CYRPRV}$, die Prozessgleichung $\vec{x}_{CYRPRV}(t+\Delta t)$ für die Transition eines Zustands in den nächsten und die Jacobi-Matrix F_{CYRPRV} dargestellt.

$$\vec{x}_{CYRPRV} = \begin{pmatrix} x \\ y \\ z \\ v \\ \Phi \\ \Theta \\ \phi \\ \theta \end{pmatrix} \qquad \vec{x}_{CYRPRV}(t+\Delta t) = \begin{pmatrix} x + sin(\Phi)cos(\Theta)v\Delta t \\ y + cos(\Phi)cos(\Theta)v\Delta t \\ z + sin(\Theta)v\Delta t \\ v \\ \Phi + \phi\Delta t \\ \Theta + \theta\Delta t \\ \phi \\ \theta \end{pmatrix} \tag{3.15}$$

$$\begin{aligned} c^a_{\phi\theta} &= sin(\Phi)cos(\Theta) & c^b_{\phi\theta} &= cos(\Phi)cos(\Theta) \\ c^c_{\phi\theta} &= sin(\Theta)sin(\Phi) & c^d_{\phi\theta} &= sin(\Theta)cos(\Phi) \end{aligned} \tag{3.16}$$

$$F_{CYRPRV} = \begin{pmatrix} 1 & 0 & 0 & c^a_{\phi\theta}\Delta t & c^b_{\phi\theta}v\Delta t & c^c_{\phi\theta}(-v)\Delta t & 0 & 0 \\ 0 & 1 & 0 & c^b_{\phi\theta}\Delta t & c^a_{\phi\theta}(-v)\Delta t & c^d_{\phi\theta}(-v)\Delta t & 0 & 0 \\ 0 & 0 & 1 & sin(\Theta)\Delta t & 0 & cos(\Theta)v\Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.17}$$

Constant Yaw Rate Pitch Rate Acceleration

Die Erweiterung des CYRPRV mit einer Zustandsvariablen für die Beschleunigung ergibt das Constant Yaw Rate Pitch Rate Acceleration Modell (CYRPRA). Die Richtung der Beschleunigung ergibt sich wie bei der Geschwindigkeit über Φ und Θ.

Nachfolgend sind der Zustandsvektor $\vec{x}_{CYRPRA}$ und die Prozessgleichung $\vec{x}_{CYRPRA}(t+\Delta t)$ dargestellt.

$$\vec{x}_{CYRPRA} = \begin{pmatrix} x \\ y \\ z \\ v \\ a \\ \Phi \\ \Theta \\ \phi \\ \theta \end{pmatrix} \qquad \vec{x}_{CYRPRA}(t+\Delta t) = \begin{pmatrix} x + (v + \frac{a\Delta t}{2}) sin(\Phi) cos(\Theta) \Delta t \\ y + (v + \frac{a\Delta t}{2}) cos(\Phi) cos(\Theta) \Delta t \\ z + (v + \frac{a\Delta t}{2}) sin(\Theta) \Delta t \\ v + a\Delta t \\ a \\ \Phi + \phi \Delta t \\ \Theta + \theta \Delta t \\ \phi \\ \theta \end{pmatrix} \tag{3.18}$$

Auf eine Darstellung der sehr umfangreichen Jacobi-Matrix F_{CYRPRA} wird an dieser Stelle verzichtet.

$$F_{CYRPRA} = \frac{\delta}{\delta x_{k-1}} f(x_{k-1}, u_k) = \frac{\delta}{\delta \vec{x}_{CYRPRA}} \vec{x}_{CYRPRA}(t+\Delta t) \tag{3.19}$$

4. Planung

Für die Bearbeitung dieser Arbeit und für spätere Anwendungen soll eine integrierte Test- und Simulationsumgebung - das Virtual Testbed - entwickelt werden, welche Analysen und Tests von Filtern und Bewegungsmodellen im zwei- und dreidimensionalen Raum erlaubt. Die Daten sollen mit virtuellen und parametrisierbaren Sensoren erzeugt werden.

Den Rahmen für die Filter, Bewegungsmodelle und Sensoren soll eine objektorientierte Tracker-Architektur bilden. Sie soll entsprechende Schnittstellen und Abstraktionen zur Verfügung stellen, um neue Filter, Bewegungsmodelle und Sensoren nahtlos einbinden zu können.

Für die Rekonstruktion von Szenarien sollen externe Daten eingebunden werden können. Dazu ist in der Simulationsumgebung die Möglichkeit der Definition von geografischen Gebieten vorzusehen. Eine entsprechende Visualisierung für den zwei- und dreidimensionalen Fall soll die Ansicht einer Simulation in Echtzeit ermöglichen.

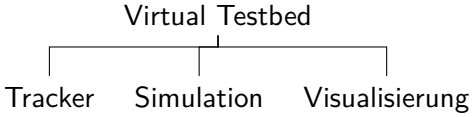


Abbildung 4.1: Komponenten des Virtual Testbed

Die Komponenten des Virtual Testbed sollen keine oder nur eine sehr geringe Kopplung zu einer der anderen Komponenten haben, um funktionale Abhängigkeiten zu vermeiden. In Folge soll eine Wartung vereinfacht und allgemein eine verbesserte Portabilität des Quellcodes gewährleistet werden.

Dieser Abschnitt legt im Folgenden die Anforderungen an die zu entwickelnden Komponenten fest. Die Konzeption legt den Fokus auf die Tracker-Architektur, welche von dem ganzheitlichen Entwurf des Virtual Testbed vorangestellt ist.

4.1 Anforderung

Die Tracker-Architektur soll die folgenden, funktionalen Anforderungen erfüllen:

- Die einfache Austauschbarkeit von Filtern und Bewegungsmodellen ist auf Implementierungsebene zu berücksichtigen.
- Filter sollen adaptiv arbeiten; die Unsicherheit der Schätzung P ist in regelmäßigen Abständen oder bei Eintreten gewisser Kriterien zu überprüfen und ggf. zu adaptieren.
- Die Orchestrierung von beliebig vielen Zielen (Filtern) und Sensoren soll möglich sein.

Für die Simulationsumgebung gelten die nachfolgenden Anforderungen:

- Radare mit realistisch modellierten Messfehlern sind für eine Einbindung zu berücksichtigen.
- Ein Ereignis- oder Nachrichtensystem soll eingebunden sein zur zeitlich synchronen Transition der Objekte bei einen Simulationsschritt (ereignisorientierte Simulation).
- Der Import von externen Daten soll grundsätzlich möglich sein.

Die Visualisierung soll die folgenden Anforderungen umsetzen:

- Ein interaktiver Anteil soll die Erstellung von Trajektorien und Fixpunkten in geografischen Gebieten und eine Zuordnung von Zielen und Radaren ermöglichen.
- Die Anzeige der aktuell simulierten Lage soll in Echtzeit erfolgen.
- Es besteht die Möglichkeit des Wechselns zwischen 2D- und 3D-Ansicht.

4.2 Konzeption

Eine beliebige Anzahl von Sensoren können jeweils eine beliebige Anzahl von Objekten messen. Die Messungen der Sensoren erreichen über ein Netzwerk alle durch einen Tracker instanziieren Filter. Ein Filter dient der Verfolgung genau eines Ziels, entsprechend überprüft ein jeder Filter, ob eine neue Messung zu dem eigenen Ziel passt. In Folge akzeptiert oder verwirft der Filter eine Messung.

Ein neuer Filter ist genau dann durch den Tracker zu instanziieren, wenn durch mindestens einen Sensor ein neues Objekt detektiert wurde. Ein neues Objekt macht sich genau dann bemerkbar, wenn keiner der Filter die korrespondierende Messung akzeptiert. Eine Instanziierung eines Filters beinhaltet dabei auch die Initialisierung des Filters mit der Messung zu dem neuen Objekt. Diese nutzt der Filter, um in Abhängigkeit des Sensorrauchens initial ein zirkulares Gate aufzubauen.

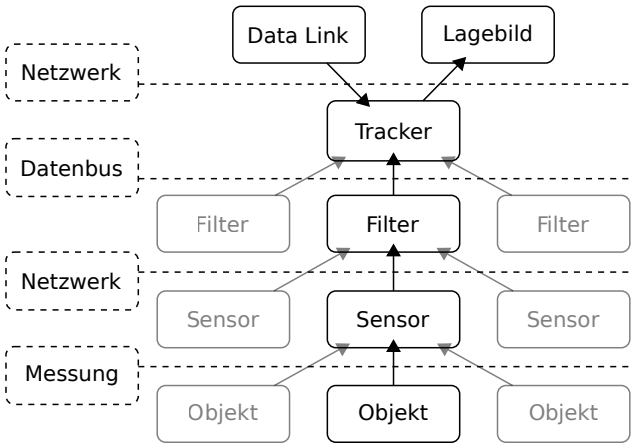


Abbildung 4.2: Konzeption der Tracker-Architektur

Der Tracker zerstört einen Filter, wenn dieser über einen definierten Zeitraum keine Messungen mehr akzeptiert oder ausschließlich verworfen hat. In dem Fall, dass zwei Filter dasselbe Ziel verfolgen, zerstört der Tracker einen der Filter.

4.3 Entwurf

Die nachfolgende Abbildung zeigt den Entwurf des Virtual Testbed bestehend aus den drei Komponenten für die Visualisierung, der Tracker-Architektur und den Anteilen der Simulationsumgebung.

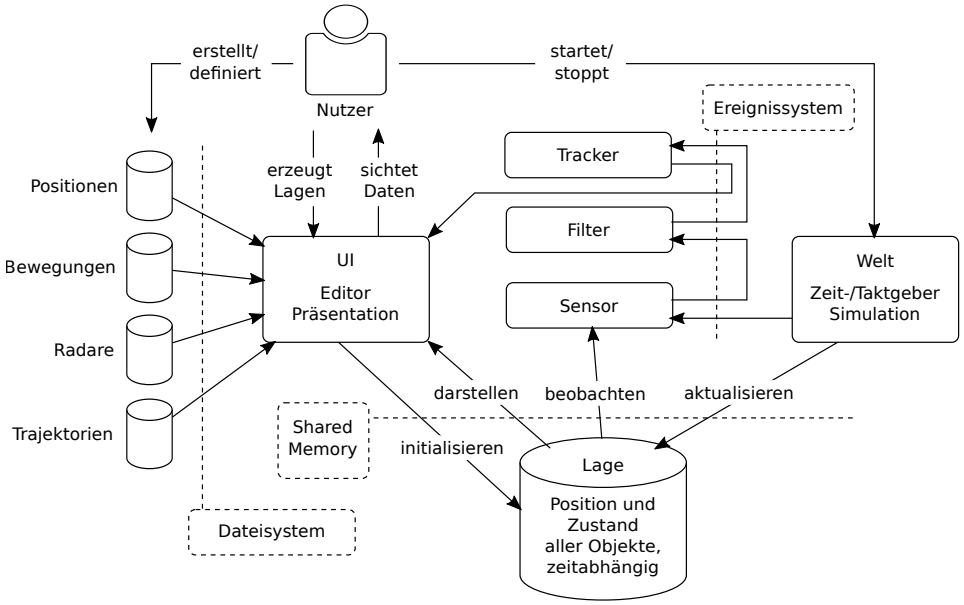


Abbildung 4.3: Abgeleitete Architektur

Die UI beinhaltet einen interaktiven Teil für eine Lageerstellung sowie die Präsentation in Echtzeit. Über externe Dateien können vordefinierte Parametrisierungen in die Lage übernommen werden. Die eingebettete Tracker-Architektur bestehend aus Tracker, Filtern und Sensoren kommuniziert über ein Ereignissystem, welches aus Sicht der Tracker-Architektur das Sensornetz darstellt. Ein globaler Zeitgeber erzeugt ein Signal, welches alle Objekte anweist, sich in den nächsten Zustand zu überführen.

5. Realisierung

Dieser Abschnitt enthält ausgewählte Implementierungsdetails und UML-Diagramme zu den wesentlichen Teilen des Virtual Testbed. Aus Gründen der Übersichtlichkeit wurde weitestgehend auf die Angabe von Typen und Argumenten und der vollständigen Angabe von Methoden in den UML-Klassendiagrammen verzichtet.

Die Anhänge A und C enthalten weiterführende Informationen zum Virtual Testbed.

5.1 Tracker-Architektur

Bewegungsmodelle

Die abstrakte Klasse Process dient als Vorschrift für die Implementierung eines Bewegungsmodells. Alle Instanzen von Process müssen lediglich zwei Methoden implementieren, was das Hinzufügen eines neuen Bewegungsmodells sehr einfach gestaltet.

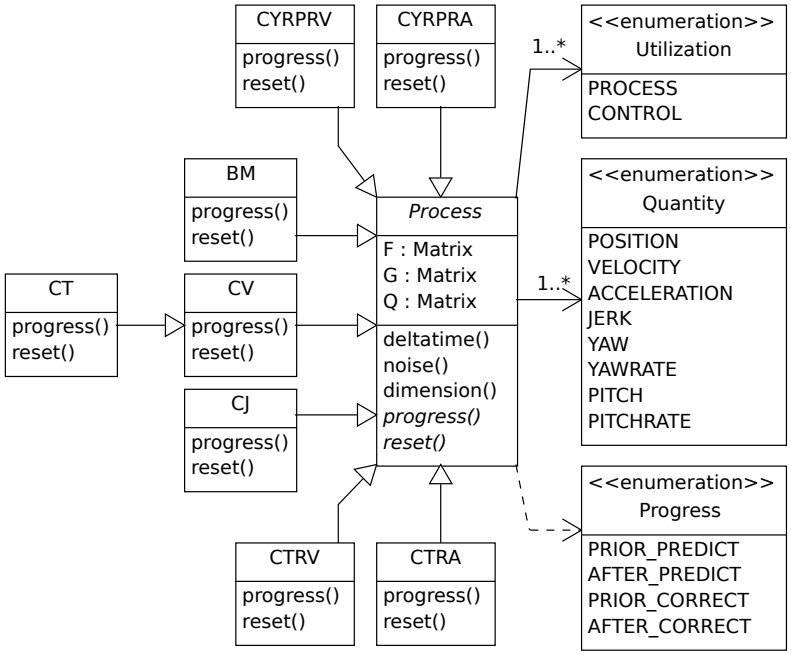


Abbildung 5.1: UML-Klassendiagramm: Bewegungsmodelle

Datencontainer

Im Bereich der Datencontainer beginnt alles mit der Klasse State, welche letztlich einen Vektor repräsentiert. Sämtliche Zustände, darunter auch Messungen (Measurement) und Schätzungen (Estimate) sind Ableitungen von State.

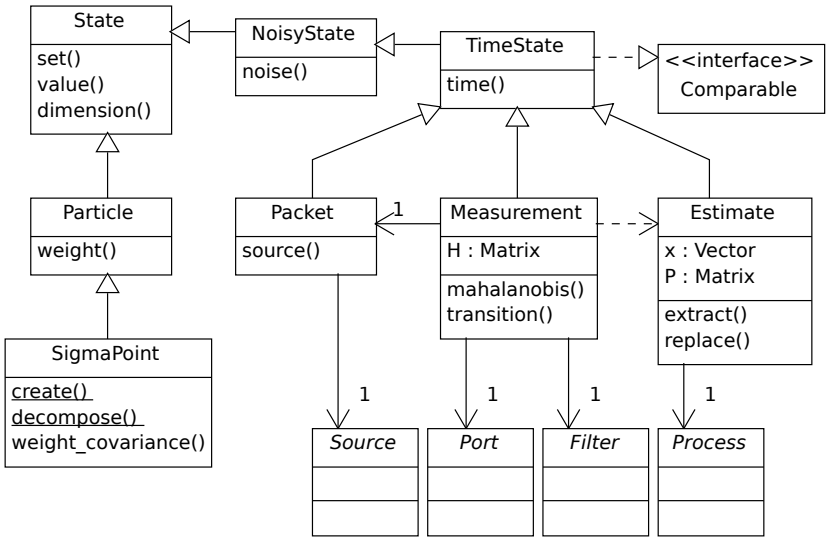


Abbildung 5.2: UML-Klassendiagramm: Datencontainer

Filter

Im Bereich der Filter gibt es ähnlich wie bei Process nur zwei Methoden, die von abgeleiteten Filtern zu implementieren sind. Im Fall der von Filter abgeleiteten Klasse KF kommen noch zwei weitere hinzu.

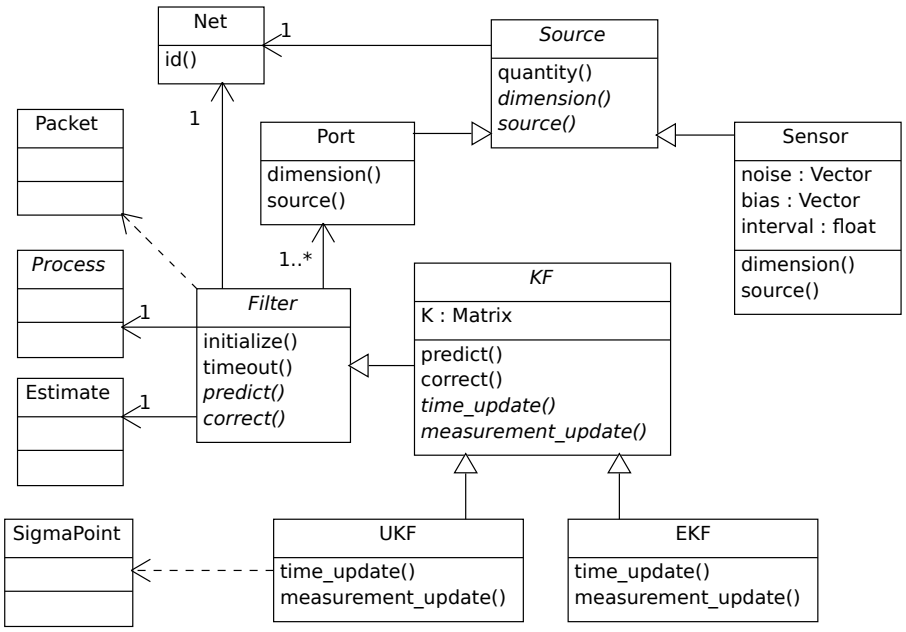


Abbildung 5.3: UML-Klassendiagramm: Filter

5.2 Simulationsumgebung

Ereignissystem

Ein zentrales Element in der Simulationsumgebung ist das Ereignissystem (Events). Es erlaubt einer jeden Instanz mittels eines publish-subscribe-Patterns Ereignisse (Nachrichten) von anderen Instanzen zu empfangen oder selbst zu senden. Eine wichtige Eigenschaft des Ereignissystems ist, dass es stets eine festgelegte Reihenfolge bei der Abarbeitung von Ereignissen einhält und ausschließlich synchron läuft, wodurch es ein reproduzierbares Verhalten erhält.

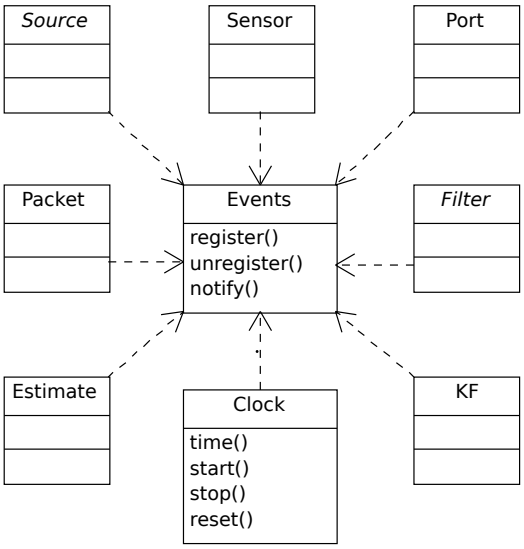


Abbildung 5.4: UML-Klassendiagramm: Ereignissystem

Die World-Instanz folgt einem Singleton-Pattern und übernimmt die Rolle eines zentralen Datencontainers, welcher Clock-Ereignisse auslöst und damit einen Simulationsschritt auslöst. Jede Instanz erhält die aktuelle Zeit aus dem Clock-Ereignis über das Ereignissystem.

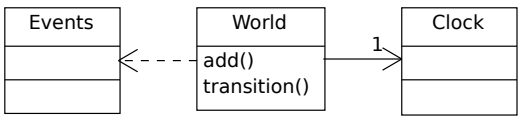


Abbildung 5.5: UML-Klassendiagramm: Zentraler Treiber und Taktgeber der Simulation

Trajektorie

Für die Bewegung eines Ziels im Raum bietet sich eine im Vorfeld konstruierte Trajektorie an. Für die Konstruktion einer solchen bieten sich wiederum Splines an, wobei die Wegpunkte der Trajektorie als Stützpunkte fungieren. Eine mit einem Spline erzeugte Trajektorie ist in äquidistante Abschnitte zu unterteilen, damit Zielbewegungen auch in engen Kurvenabschnitten gleichmäßig verlaufen und eine einfache lineare Interpolation von Positionen auf der Trajektorie möglich ist.

Der nachfolgende Algorithmus approximiert iterativ aus einer Sequenz von Wegpunkten eine äquidistante Trajektorie unter Verwendung von Splines.

```
in:  R  Sequenz von n Wegpunkten
     a  Abschnittslänge
out: T  Menge von Wegpunkten mit Abstand a
begin
  T ← R
  do
    T ← erzeuge Spline mit n gleichverteilten Punkten aus T
    D ← for i in 1 to n−1: berechne Euklidischen Abstand von T(i) zu T(i+1)
    d ← sum: for i in 1 to n−1: D(i)
    E ← for i in 1 to n−1: abs: D(i) − (d / n)
    e ← sum: for i in 1 to n−1: E(i)
    n ← floor: d / a + 3
  while e > a
  return T
end
```

Abbildung 5.6: Erzeugung einer Trajektorie mit äquidistanten Abschnitten

Zielbewegung

Bewegung spielt für die Simulation eine zentrale Rolle. Sie beschreibt die Veränderung eines Ortes über die Zeit und bildet die Grundlage für alle Positionsänderungen. Die Berechnung eines neuen Ortes x zum Zeitpunkt t kann stets erfolgen, wenn die den Ort x über die Zeit verändernden Größen wie die Geschwindigkeit v oder die Beschleunigung a bekannt sind.

$$\begin{aligned}
&1. \quad c_0 = x(0) \quad c_1 = v(t) \quad x(t) = \frac{c_0\, t^0}{1} + \frac{c_1\, t^1}{1} \\
&2. \quad c_1 = v(0) \quad c_2 = a(t) \quad x(t) = \frac{c_0\, t^0}{1} + \frac{c_1\, t^1}{1} + \frac{c_2\, t^2}{2} \\
&3. \quad c_2 = a(0) \quad c_3 = j(t) \quad x(t) = \frac{c_0\, t^0}{1} + \frac{c_1\, t^1}{1} + \frac{c_2\, t^2}{2} + \frac{c_3\, t^3}{6}
\end{aligned} \tag{5.1}$$

Die folgende und sehr allgemeine Formulierung zur Ortsbestimmung bildet die Grundlage für die Erzeugung von dynamischen Bewegungsmustern und für die Positionsbestimmung eines Ziels auf einer Trajektorie.

$$x(t) = \sum_{n=0}^{k} \frac{c_n\, t^n}{n!} \tag{5.2}$$

Messung

Die Messung eines Radars findet in dessen, lokalen und polaren, Koordinatensystem statt. Der modellierte Messfehler σ enthält einen horizontalen, vertikalen und longitudinalen Anteil und ist in der Einheit mrad bzw. ‰ angegeben. Dies erlaubt die einfache Berechnung eines Fehlers basierend auf der gemessenen Entfernung und resultiert in der Erzeugung des Messfehlers $\vec{v}_{err}$ in einem kartesischen Koordinatensystem.

$$\vec{v}_{err} = (x_{err} \sim \mathcal{N}(0,\, \sigma_{hor}), y_{err} \sim \mathcal{N}(0,\, \sigma_{lon}), z_{err} \sim \mathcal{N}(0,\, \sigma_{ver}))^T \tag{5.3}$$

Die Erzeugung einer verrauschten Messung $\vec{v}_{meas}$ erfolgt mittels der tatsächlichen Position des gemessenen Ziels $\vec{v}_{true}$. Der kartesische Rauschanteil $\vec{v}_{err}$ ist entsprechend zu rotieren, um eine Korrelation zu der

zugrundeliegenden, polaren Radarmessung herzustellen. Die Matrix-Funktion $R(\beta, \epsilon)$ erzeugt aus den Euler-Winkeln zu Azimut β und Elevation ϵ eine Rotationsmatrix für die Rotation von $\vec{v}_{err}$.

$$R(\beta,\epsilon) = \begin{pmatrix} cos(\epsilon) & -sin(\epsilon) & 0 \\ sin(\epsilon) & cos(\epsilon) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos(0) & 0 & sin(0) \\ 0 & 1 & 0 \\ -sin(0) & 0 & cos(0) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(-\beta) & -sin(-\beta) \\ 0 & sin(-\beta) & cos(-\beta) \end{pmatrix} \tag{5.4}$$

$$\vec{v}_{meas} = \vec{v}_{true} + R(\beta,\epsilon)\,\vec{v}_{err} \tag{5.5}$$

5.3 Visualisierung

Die Visualisierung nutzt für die Anzeige hauptsächlich Instanzen von zwei Klassen: Artifact und Actor. Eine Instanz von Artifact kann sich selbst auf ein 2D-Canvas (tk.Canvas) zeichnen oder in der 3D-Darstellung (mpl.Figure) plotten.

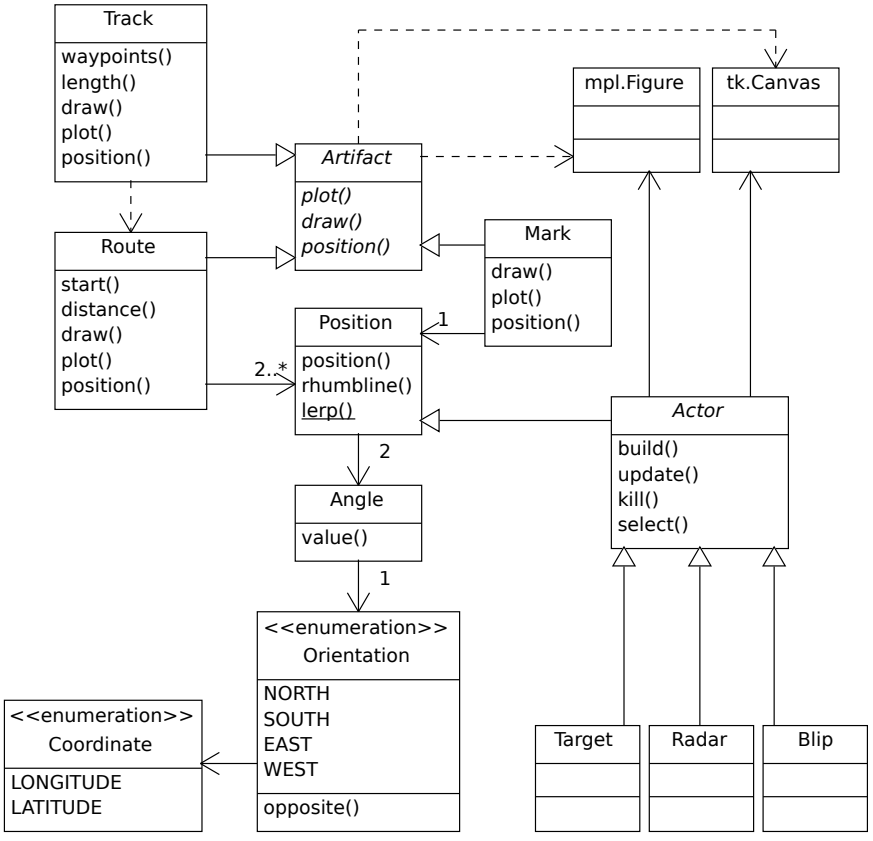


Abbildung 5.7: UML-Klassendiagramm: Artefakte und Akteure

Artifacts und Actors verfügen über das Wissen ihrer eigenen, geografischen Position. Die 2D-Karten und die 3D-Plots bieten für das Mapping einer geografischen Position zu einer xy- bzw. xyz-Koordinate in ihrem lokalen kartesischen Koordinatensystem eine Methode an. Dies ermöglicht die Bewegung eines Actors in einem global geografischen Koordinatensystem bei gleichzeitiger Kenntnis der Position im jeweiligen lokalen kartesischen Koordinatensystem einer 2D-Karte oder eines 3D-Plots.

6. Evaluation

In diesem Abschnitt erfolgt eine Evaluation der implementierten Tracker-Architektur mit einem ausgewählten Filter und vier Bewegungsmodellen. Aufgrund der Komplexität des Trackings-Problems kommen rein analytische Lösungen nicht in Frage, entsprechend stützt sich die Evaluation auf diskrete und ereignisorientierte Monte-Carlo-Simulationen mit Hilfe der dafür implementierten Simulationsumgebung.

Die Simulationen erfolgen in Lagen, welche verschiedene Zielbewegungen mit unterschiedlichen Radaranordnung kombinieren. Der Filter versucht innerhalb einer Lage und während einer Simulation anhand der ihm bereitgestellten Messungen von einem oder mehreren Radaren den Zustand eines zu verfolgenden Luftziels in einem geografisch definierten Gebiet zu schätzen.

Einen essentiellen Baustein für die Evaluation bilden die durch den Filter während der Simulation erzeugten Datensätze, welche zu jedem Zeitpunkt eine quasi-kontinuierliche Einschätzung des Filters erlauben. Die Ausgabe der zeitlich markierten Datensätze als Grundlage für die in diesem Abschnitt folgenden, quantitativen Diskussionen erfolgt während einer Simulation über eine Schnittstelle.

6.1 Vorbereitung

Auswahl und des Verfahrens

Der UKF zeigt entsprechend der Erwartung in der Testumgebung (siehe Anhang A) in Verbindung mit dem CTRV- und CTRA-Modell eine vergleichbare und teilweise bessere Schätzgenauigkeit als der EKF, bei ähnlicher Laufzeitkomplexität und gleichzeitig geringeren Anforderungen an die Implementierung des Bewegungsmodells.

Aus diesem Grund ist der UKF für die Evaluation in die Tracker-Architektur eingebettet. Der Fokus liegt auf der Ermittlung der Performance, der Datenfusionsfähigkeit und der Trackgenauigkeit des UKF mit dem CV-, CA-, CYRPRV- und CYRPRA-Bewegungsmodell innerhalb der Tracker-Architektur.

Parametrisierung

Für die Erstellung der Sigma Points im UKF gilt die folgende und unter Annahme von normalverteiltem Rauschen optimale Parametrisierung: $\alpha = 0.001$, $\kappa = 0$ und $\beta = 2$ [WVDM00].

Die Konstruktion der Prozessrauschverteilungsmatrix Q erfolgt nach einer Heuristik, welche sich im Rahmen von Tests in der Testumgebung als praktikabel erwiesen hat. Die Heuristik hat einen normierenden Effekt auf die Prozessrauschverteilungsmatrizen der unterschiedlichen Zustandsräume in den Bewegungsmodellen. Sie dient somit vor allem zur Wahrung der Vergleichbarkeit der unterschiedlichen Bewegungsmodelle bei einheitlicher Parametrisierung der Intensität für ein zugrunde gelegtes Prozessrauschen.

```
in:  q  Sequenz mit n Werten zwischen 2 und 1
     Δt Zeitschritt der Schätzung
out: Q  Matrix des Prozessrauschens
begin
  for i in 1 to n:
    for j in 1 to n:
      if abs(i−j) mod n = 0:
        Q[i,j] ← (Δt(q[i]+q[j]))/(q[i]q[j])
  return Q
end
```

Abbildung 6.1: Erzeugung einer einheitlichen Prozessrauschverteilungsmatrix

Die Sequenz q enthält n Werte zur Beschreibung der Rauschanteile der n Zustandsvariablen für die Prozessrauschverteilungsmatrix Q. Die Zustandsvariablen erzeugen in Abhängigkeit der Ordnung der von ihnen repräsentierten Größe einen Wert zwischen 2 und 1 in der Sequenz q.

Diejenigen Variablen, die Größen erster Ordnung darstellen, erzeugen den Wert 2. Positionen, Gierwinkel (yaw) und Neigungswinkel (pitch) sind Größen erster Ordnung, entsprechend erzeugen alle Variablen zur Beschreibung dieser Größen den Wert 2 für den Rauschanteil in der Sequenz q. Die Variablen höchster Ordnung erzeugen den Wert 1. Die verbliebenen Variablen erzeugen in Abhängigkeit ihrer Ordnung einen gleichverteilten Wert zwischen 2 und 1 in der Sequenz q zur Beschreibung ihres jeweiligen Rauschanteils.

Lage

Eine Lage befindet sich stets in einem geografischen Gebiet, welches mittig auf dem Schnittpunkt von Äquator und Nullmeridian liegt. Es hat eine Ausdehnung von 8 km in Breite, Länge und Höhe. In einer Lage befindet sich stets ein zu verfolgendes Ziel und ein bis drei Radare, welche anhand ihrer Parametrisierung Messungen zu dem Ziel durchführen und in ein Sensornetz geben. Die Beschreibung der Zielbewegung erfolgt mit einer Trajektorie und einem Bewegungsmuster. Radare sind durch ihren Typ und den verwendeten Träger, welcher stationär oder mobil sein kann, charakterisiert.

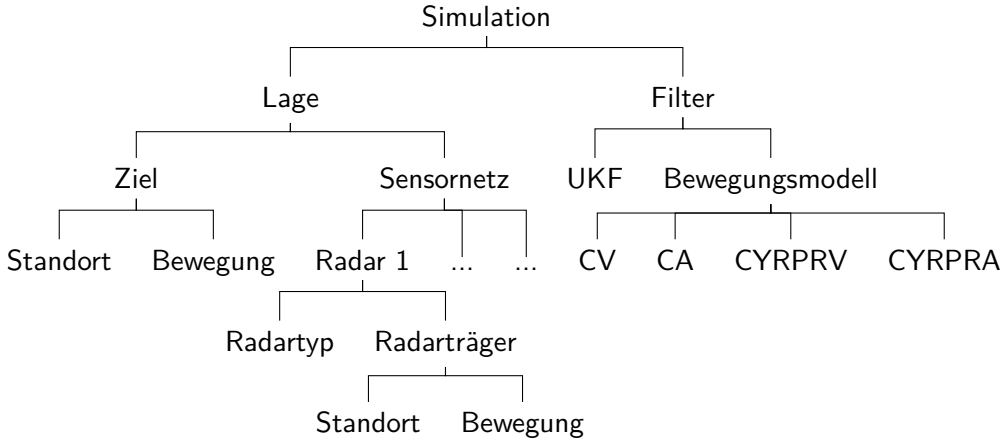


Abbildung 6.2: Elemente einer Lage im Kontext der Simulation

Eine Lage enthält den Standort oder die Bewegung (eine Trajektorie und ein Bewegungsmuster) für ein zu verfolgendes Ziel sowie für mindestens ein Radar, welches Messungen zu dem Ziel liefern kann. Die Verknüpfung von Bewegungsmustern und Trajektorien zu einer Zielbewegung erzeugt für die Evaluation stufenweise abgrenzbare Schwierigkeitsgrade für die Zielverfolgung.

Trajektorie

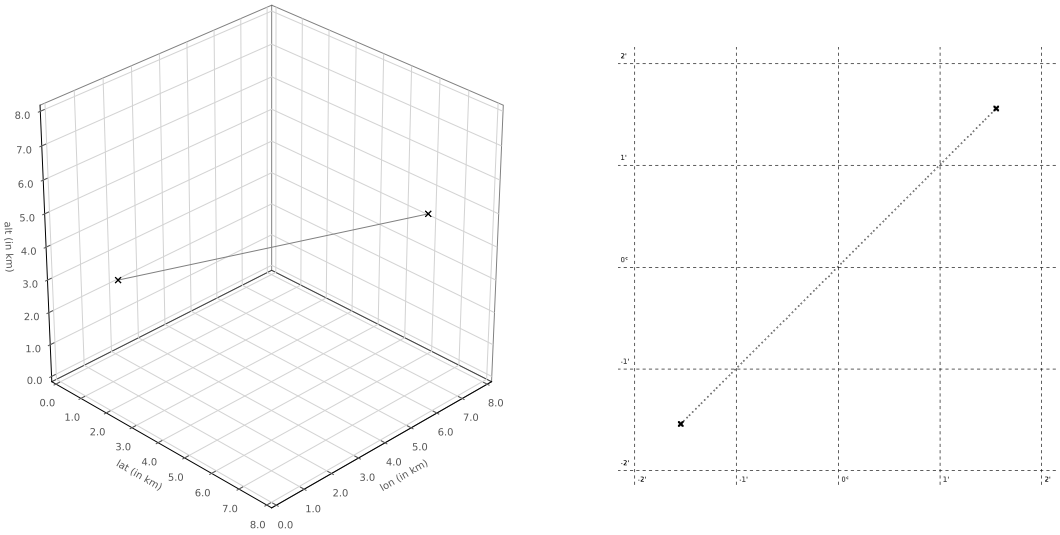
Für die Zielbewegung in der Lage stehen drei Trajektorien mit steigender Komplexität zur Verfügung. Aus praktischen Gründen sind die Trajektorien in der Länge ungefähr gleich, um mit einem festgelegten Bewegungsmuster ähnliche Simulationszeiten und Datensatzgrößen erzielen zu können sowie direkte und qualitative Vergleiche der Trajektorien untereinander zu erlauben.

ID	Bezeichnung	Aspekte	Wegpunkte	Länge
T1	geradlinig	linear, keine Manöver	01.540S 01.546W 3000 m 01.557N 01.550E 5000 m	8352 m
T2	kurvig	nicht-linear, ein langgezogenes Manöver	00.610S 01.546W 5000 m 00.629N 00.003E 6000 m 00.610S 01.550E 4000 m	8352 m
T3	komplex	nicht-linear, viele unterschiedliche Manöver	00.920S 00.230E 1000 m 00.300S 00.313E 1500 m 00.319N 00.307W 2500 m 00.629N 00.616W 2250 m 01.558N 00.152W 2000 m 00.939N 00.313W 2750 m	8243 m

Tabelle 6.1: Trajektorien für Ziele

Gerader Flug

Die Trajektorie T1 beschreibt eine gerade Flugbahn ohne Richtungsänderung. Im Verlauf des Fluges erfolgt eine Steigung von 3000 m auf 5000 m über die Gesamtdistanz von etwas mehr als 8300 m. Diese sehr einfache Trajektorie bildet im ersten Schritt der Evaluation die Möglichkeit der Diskussion einer Filter-Performance unter sehr einfachen und klaren Bedingungen.

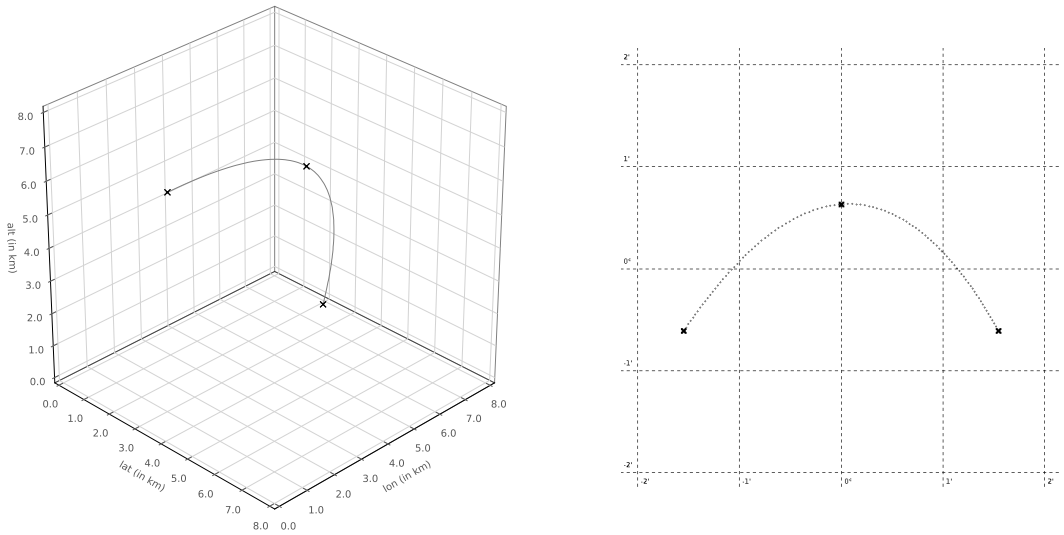


(a) 3D, kartesisch (b) 2D, geografisch

Abbildung 6.3: Trajektorie T1

Kurviger Flug

Der Kurvenflug der Trajektorie T2 besteht aus zwei Phasen. Innerhalb der ersten Phase erfolgt ein gekrümmter Steigflug von 5000 m auf 6000 m. Etwa im Scheitelpunkt beginnt in der zweiten Phase ein deutlicher und ebenfalls gekrümmter Sinkflug auf 4000 m. Die Krümmung und die damit einhergehenden, nicht-linearen Anteile für die Bewegung eines Ziels auf der Trajektorie stellen eine erste Schwierigkeit für die Zielverfolgung dar.

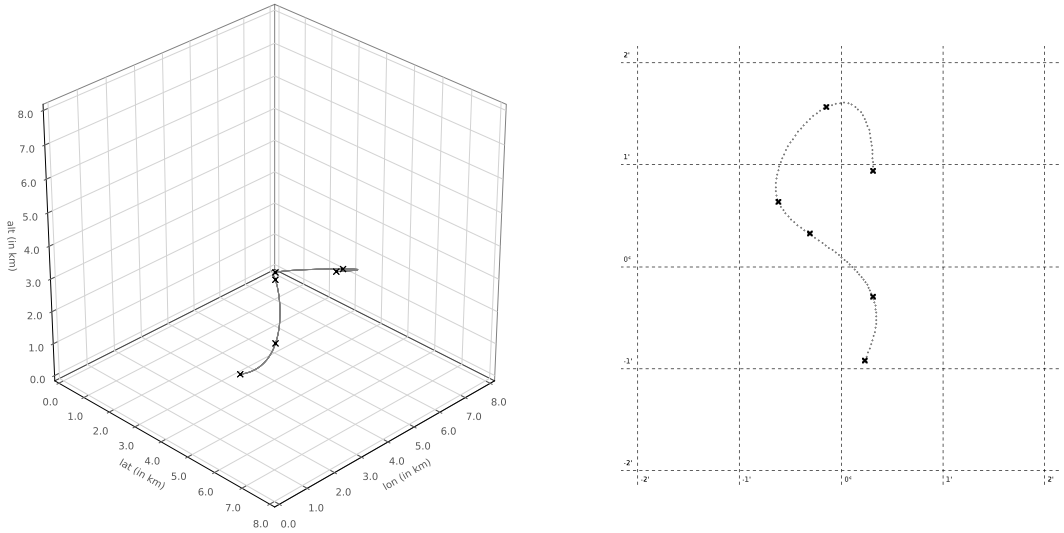


(a) 3D, kartesisch (b) 2D, geografisch

Abbildung 6.4: Trajektorie T2

Komplexer Flug

Die Trajektorie T3 beinhaltet eine komplexe Flugbahn mit verschiedenen Manöveranteilen für ein ganzheitliches Filter-Benchmarking. Nach einem leicht gekrümmten Startabschnitt folgt ein deutlicher Steigflug mit anschließendem, leichten Sinkflug. Im zweiten Teil erzeugen zwei in der Krümmung zunehmende Kurvenabschnitte mit gleichzeitiger Höhenänderung eine für eine Zielverfolgung anspruchsvolle Bewegung.



(a) 3D, kartesisch (b) 2D, geografisch

Abbildung 6.5: Trajektorie T3

Bewegungsmuster

Ein Bewegungsmuster beschreibt die zurückgelegte Distanz eines Ziels auf einer Trajektorie über die Zeit. Bewegungsmuster sind konstant oder dynamisch. Dynamische Bewegungsmuster verändern sich über die Zeit, um beliebige (und möglichst realistische) Bewegungen zu modellieren. Die Änderung der Dynamik einer Bewegung erfolgt über eine Veränderung der Geschwindigkeit v, der Beschleunigung a oder des Rucks j zu einem Zeitpunkt t. Insgesamt sind drei Bewegungsmuster für eine Zielbewegung vorgesehen.

ID	Typ	Zeit	Änderung
B1	konstant, langsam	$t = 0$	$v(t) = 50$ m/s $a(t) = 0$ m/s^2 $j(t) = 0$ m/s^3
B2	dynamisch	$t = 0$	$v(t) = 70$ m/s $a(t) = 1$ m/s^2 $j(t) = 0{,}01$ m/s^3
		$t = 30$	$j(t) = 0{,}001$ m/s^3
		$t = 75$	$j(t) = -0{,}05$ m/s^3
		$t = 100$	$j(t) = 0$ m/s^3
B3	konstant, schnell	$t = 0$	$v(t) = 175$ m/s $a(t) = 0$ m/s^2 $j(t) = 0$ m/s^3

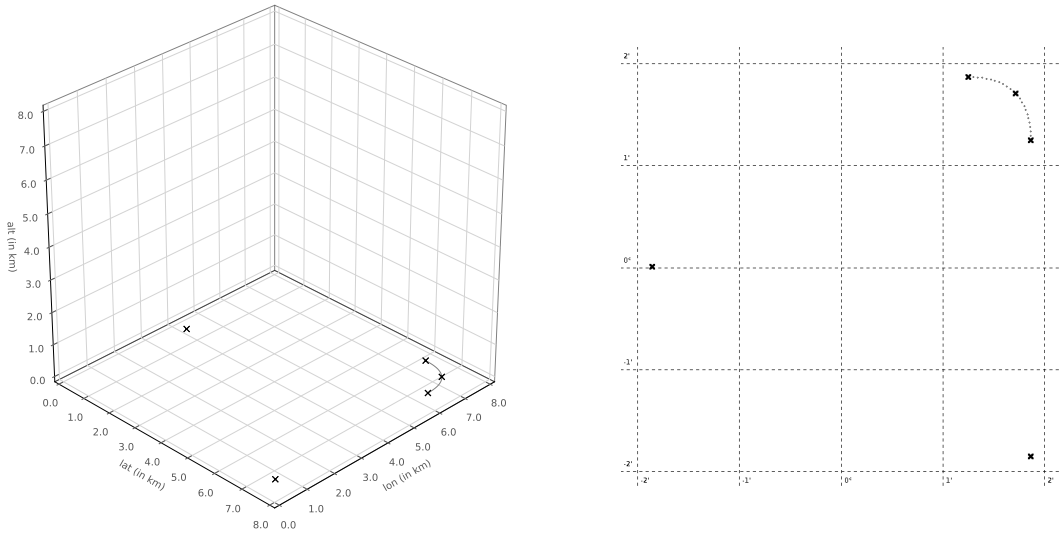
Tabelle 6.2: Bewegungsmuster für Ziele

Die Bewegungsmuster B1 und B3 modellieren eine konstante Bewegung für kleine/langsame und große/schnelle Flugzeuge. Eine dynamische Bewegung erzeugt das Bewegungsmuster B2 durch eine dreistufige Veränderung des Rucks j zu definierten Zeitpunkten t: Beginnend mit einer $v(0)$ von 70 m/s und einer Beschleunigung $a(0)$ von 1 m/s^2 hat ein Ziel mit einem konstanten Ruck $j(0)$ von 0,001 m/s^3 nach 10 s bereits eine Geschwindigkeit von etwas über 80 m/s, nach 30 s über 100 m/s und nach 60

s sind es bereits über 140 m/s. Dieses nicht-lineare Bewegungsmuster für Zielbewegungen stellt eine Hürde für die Zielverfolgung dar.

Radarträger

Radare sind an einen Radarträger gebunden, welche zu jedem Zeitpunkt über eine Position verfügen. Die Radarträger teilen ihre Position dem Radar für eine Messung mit, damit eine Transformation der Messung aus dem lokalen Radar-Koordinatensystem in das globale Welt-Koordinatensystem erfolgen kann.



(a) 3D, kartesisch (b) 2D, geografisch

Abbildung 6.6: Positionen für Radarstationen und Trajektorie für Radarträger

Für stationäre Radarträger (Radarstationen) sind die zwei festen Positionen P1 und P2 reserviert. Mobile Radarträger, beispielsweise Schiffe oder bodengestützte Fahrzeuge, sind der Trajektorie Q1 zuzuordnen.

ID	Typ	Position
P1	stationär	01.849S 01.862E
P2	stationär	00.009N 00.003E
Q1	mobil, $v(0) = 10$ m/s	1) 01.248N 01.862E 0 m 2) 01.713N 01.707E 0 m 3) 01.868N 01.248E 0 m

Tabelle 6.3: Positionen und Bewegungen für Radarträger

Radartyp

Für die Simulation stehen drei Radare zur Verfügung. Jedes Radar deckt von jeder möglichen Position innerhalb des geografisch definierten Gebiets aufgrund seiner Reichweite das gesamte Gebiet ab. Die Messfehler in Azimut, Elevation und Distanz sind abhängig von der Entfernung zum gemessenen Ziel; je weiter ein Ziel entfernt ist, umso schlechter sind die Messungen zu diesem Ziel.

Bei einer Entfernung von 10 km zum Ziel hat beispielsweise das Radar R1 einen normalverteilten Standardfehler von jeweils 40 m horizontal, vertikal und longitudinal. Eine Positionsmessung dieses Radars liegt zu

ca. 68 % innerhalb einer entsprechend zum Azimut und zur Elevation rotierten Box mit einem Schwerpunkt an der wahren Position des Ziels und einer Länge von 40 m für die orthogonal zueinander liegenden Kanten.

ID	Drehfrequenz	Messfehler		
		Azimut	Elevation	Distanz
R1	1,0/s	4,0 mrad	4,0 mrad	4,0 ‰
R2	1,0/s	2,5 mrad	2,5 mrad	2,5 ‰
R3	0,5/s	1,0 mrad	1,0 mrad	1,0 ‰

Tabelle 6.4: Parametrisierung der Radare mit zufälligen und normalverteilten Messfehlern

Im Rahmen der Evaluation besteht nicht der Anspruch komplexe Radarsysteme und deren Fehler zu modellieren. In der Praxis sind die typischen Messfehler geringer als hier angegeben. Das hier genutzte, vereinfachte und künstlich verschlechterte Fehlermodell ist insofern vorteilhaft, als das es eine Bewertung der Filter-Performance unter erschwerten Bedingungen erlaubt und gut vorhersagbar ist.

6.2 Durchführung

Die Durchführung der Evaluation beinhaltet die Simulation von Lagen in definierten Szenarien und die Erhebung von Datensätzen als Datengrundlage für eine Auswertung mit Hilfe von eingeführten Fehlermaßen.

Szenarien

Die Evaluation erfolgt mittels drei definierter Szenarien. Die Basis eines jeden Szenarios bildet eine Trajektorie und ein Bewegungsmuster zur Beschreibung der Bewegung eines zu verfolgenden Ziels. Die jeweils vier Varianten eines Szenarios sollen unterschiedliche Fragestellungen beleuchten und qualitative Vergleiche ermöglichen. Die quantitative Untersuchung der insgesamt zwölf Varianten erfolgt anhand von zehn Durchläufen pro Variante, dessen Ergebnisse in den folgenden Abschnitten dargestellt sind.

Die nachfolgende Tabelle gibt einen umfassenden Überblick über die Szenarien und ihrer Varianten inklusive der an einer Lage beteiligten Akteure.

Szenario	Basis	Variante			
		Detaillierung der Lage		Filter	Modell
1	T1+B1	1	R1:P1	UKF	CV
		2	R2:P1		
		3	R3:P1		
		4	R1:P2 + R2:P1 + R3:Q1		
2	T2+B2	1	R2:P2 + R1:P1	UKF	CV
		2	R2:P2 + R1:P1		CA
		3	R2:P2 + R1:P1		CYRPRV
		4	R2:P2 + R1:P1		CYRPRA
3	T3+B3	1	R3:P2 + R2:Q1	UKF	CV
		2	R3:P2 + R2:Q1		CA
		3	R3:P2 + R2:Q1		CYRPRV
		4	R3:P2 + R2:Q1		CYRPRA

Tabelle 6.5: Überblick der Szenarien mit Lagebeschreibung

Die Lagebeschreibung jeder Variante enthält die Positionen bzw. Bewegungen der Radarträger. Die Zielverfolgung erfolgt in jedem Szenario mit dem UKF als Filter in Verknüpfung mit einem der vier angegebenen Bewegungsmodelle.

Das Szenario 1.1 bis 1.3 dient zur Verifikation der korrekten Datenintegration bei unterschiedlicher Messqualität im Radar-Einzelbetrieb. Das Szenario 1.4 beinhaltet den Betrieb aller Radare und dient zur Verifikation der korrekten Datenfusion mit heterogenen Messqualitäten sowie zum Vergleich des Radar-Einzelbetriebs mit einem Radar-Mehrbetrieb.

Im Szenario 2 erfolgt in den Varianten 1 bis 4 ein Vergleich der Bewegungsmodelle mit einer dynamischen Zielbewegung in einem verrauschten Sensorumfeld. Die Anordnung der Radare erlaubt die vollständige Abdeckung der Zieltrajektorie. Das Ziel entfernt sich von dem besseren Radar und nähert sich dem schlechteren Radar an, wodurch im Verlauf sehr unterschiedliche Messfehler möglich sind. Die Herausforderung in diesem Szenario ist vor allem die Datenfusion der verrauschten Messungen und die korrekte Schätzung des Zielzustands mit Hilfe des jeweils verwendeten Bewegungsmodells.

Das Szenario 3 dient zur Untersuchung der Performance der Bewegungsmodelle unter möglichst realistischen Bedingungen. Das relativ schnelle Luftziel dieses Szenarios bewegt sich im Verlauf sehr agil. Der Filter muss trotz der hohen Geschwindigkeit und der verhältnismäßig geringen Anzahl von Messungen die dynamische Bewegung des Ziels schätzen. Bei Verlust des Ziels (Loss) erfolgt nach einem definierten Timeout ein Reset, gefolgt von einer Reinitialisierung des Filters. Während der Initialisierung erfolgen keine Schätzungen, wodurch der Filter keine Daten zum Ziel liefern kann.

Im Rahmen der Auswertung sind lediglich Phasen berücksichtigt, in welchen ein Filter Schätzungen durchführt und entsprechende Daten liefert. Eine höhere Anzahl von Resets geht einher mit einer verringerten Zeit, in welchen ein Filter aktiv das Ziel verfolgt. Entsprechend wirken sich häufige Zielverluste negativ auf die Bewertung der Filter-Performance aus.

Datensätze

Die Datenerzeugung und -erhebung findet in bzw. mit der Simulationsumgebung während eines Durchlaufs statt. Ein Datensatz beinhaltet für die gesamte Simulationszeit eines Durchlaufs im Wesentlichen

- die vom Filter akzeptierten (in die Schätzung integrierten) Positionsmessungen,
- die vom Filter verworfenen Positionsmessungen,
- die geschätzten Positionen des verfolgten Ziels und
- die wahren Positionen des verfolgten Ziels.

Die polaren Positionsmessungen aus den lokalen Radar-Koordinatensystemen sind aus Darstellungsgründen und für eine einfachere Auswertung in das kartesische Welt-Koordinatensystem transformiert. Das kartesische Welt-Koordinatensystem hat seinen Urspung stets im südwestlichsten Punkt und auf der Ebene des geografischen Gebiets.

Fehlermaße

Die Diskussion und Auswertung der Datensätze erfolgt hauptsächlich anhand zweier Fehlermaße: dem prozentualen Volumenfehler P_{bias} für relative Betrachtungen und dem Effektivwert der logarithmierten Abweichung RMS_D für absolute Betrachtungen.

Prozentualer Volumenfehler

Der prozentuale Volumenfehler gibt Aufschluss über den relativen Fehler zweier Zeitreihen; eine simulierte bzw. geschätzte Zeitreihe gegenüber der wahren bzw. beobachteten Zeitreihe. Er dient als Kennzahl für systematische Über- oder Unterschätzung. Hier erfolgt die Anwendung des P_{bias} mit den Residuen aus wahrer und beobachteter sowie wahrer und geschätzter Zeitreihe.

$$P_{bias} = \frac{\sum_{i=1}^{n}(o_i - y_i)}{\sum_{i=1}^{n}(o_i)} \qquad o_i = \|x_i - z_i\| \qquad y_i = \|x_i - \hat{x}_i\| \tag{6.1}$$

In die Berechnung des P_{bias} gehen aus einem Datensatz n zeitlich konsekutive Werte zu dem verfolgten Ziel ein: Die wahren und geschätzten Positionen $x_i \in \mathbb{R}^3$ und $\hat{x}_i \in \mathbb{R}^3$ sowie die gemessenen Positionen $z_i \in \mathbb{R}^3$. Die Residuen o_i und y_i ergeben sich aus den Euklidischen Abständen von wahrer Position zu gemessener Position $\|x_i - z_i\|$ und wahrer Position zu geschätzter Position $\|x_i - \hat{x}_i\|$.

Effektivwert der Abweichung

Der Effektivwert der Abweichungen RMS_D erzeugt einen durchschnittlichen Wert für zwei zeitlich angeglichene Positionsmessreihen. In diesem Kontext ermöglicht er den absoluten Vergleich der Abweichungen zwischen wahren und geschätzten Positionen sowie den Abweichungen zwischen wahren und gemessenen Positionen.

$$\log_e RMS_D = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\log_e\|x_i - w_i\|^2} \tag{6.2}$$

Die Berechnung des RMS_D der Schätzung erfolgt mittels der wahren Positionen x_i und der geschätzten Positionen $\hat{x}_i = w_i$. Analog erfolgt die Berechnung des RMS_D der Messungen mit den gemessenen Positionen $z_i = w_i$. Der Euklidische Abstand $\|x_i - w_i\|$ dient zur Bildung der Abweichungen; die Grundlage für die Berechnung des RMS_D.

Der logarithmische Anteil bei der Berechnung der Abweichungen dient als inhärente Ausreißerdiskussion zur Minimierung des Einflusses von hohen Abweichungen. Der Logarithmus im RMS_D stellt im Prinzip einen Tiefpass dar, welcher große Werte abflacht und kleine Werte weitestgehend unberührt lässt. Neben der Reduzierung der überproportionalen Gewichtung von Ausreißern entsteht auch ein glättender Effekt bei der Berechnung der Abweichungen.

6.3 Auswertung

Im ersten Schritt erfolgt eine Einzelbetrachtung der Szenarien anhand aller durchgeführten Durchläufe mit Hilfe der eingeführten Fehlermaße. Eine darauf folgende Schlussbetrachtung vergleicht die Ergebnisse der Szenarien ganzheitlich.

Das Szenario 1 dient initial neben der Untersuchung des Filters auch der Überprüfung der korrekten Funktion des Filters innerhalb der Tracker-Architektur durch Abgleich der Ergebnisse mit eigenen Erfahrungen und Erwartungen. Ferner dient das sehr einfach gehaltene Szenario 1 der Überprüfung der Fehlermaße hinsichtlich ihrer Aussagekraft zur Beschreibung wesentlicher Aspekte bei der Bestimmung von Trackgenauigkeiten und Datenfusionsfähigkeiten.

Die nachfolgenden Tabellen beinhalten für das jeweilige Szenario pro Variante die Mittelwerte der zehn durchgeführten Durchläufe für die RMS_D der Schätzungen und Messungen sowie die des P_{bias}. Die

Mittelwerte der RMS_D sind zusätzlich mit Standardabweichungen ergänzt. Die Anzahl (der Umfang) der in den Durchläufen einer Variante durchgeführten Schätzungen und vom Filter verwerteten (akzeptierten) Messungen befinden sich in den letzten beiden Spalten der Tabellen.

Die Ergebnisse der einzelnen Durchläufe, welche die Grundlage für diese Auswertung bilden, sind im Anhang B dokumentiert.

Szenario 1

Das Szenario 1 beinhaltet einen gradlinigen Flug mit einem bzw. drei Radaren (vgl. Tabelle 6.5).

	RMS_D (in m)						
	Schätzung		Messung			Umfang	
Variante	μ	σ	μ	σ	P_{bias}	Schätzung	Messung
1	32,98	2,03	40,03	2,08	0,05	33866	1322
2	23,19	1,26	24,62	0,92	-0,12	39956	1557
3	14,56	0,89	10,36	0,54	-0,98	39503	784
4	16,19	1,19	24,17	0,73	0,14	40529	3921

Tabelle 6.6: Zusammenfassung der Ergebnisse der Durchläufe aus dem Szenario 1

In der Variante 1 zeigt der leicht positive P_{bias} und der kleinere RMS_D bei den Schätzungen eine gute Filter-Performance trotz der stark verrauschten Messungen des Radar R1.

Der Filter zeigt in der Variante 2 bei mittelmäßig verrauschten Messungen nur einen sehr kleinen Unterschied bei den RMS_D. Der leicht negative P_{bias} lässt den Schluss zu, dass der Filter nicht immer die guten Messungen des Radar R2 in entsprechend gute Schätzungen überführen kann.

Die Variante 3 zeigt die beste Schätzgenauigkeit aufgrund des geringen RMS_D bei den Schätzungen. Der hohe und negative P_{bias} zeigt eine hohe Diskrepanz zwischen den sehr guten Messungen des Radar R3 und den im Verhältnis schlechteren Schätzungen des Filters. In der relativen Betrachtung zeigt der Filter in dieser Variante die schlechteste Performance, da die Schätzungen durchweg schlechter sind als die Messungen bedingt durch die geringe Anzahl von verfügbaren Messungen.

In der Variante 4 sind alle verfügbaren Radare involviert, wodurch sehr viele Messungen zur Verfügung stehen. Wie erwartet zeigt der geringe RMS_D bei den Schätzungen in Kombination mit dem relativ hohen P_{bias} und trotz des hohen Anteils verrauschter Messungen eine sehr gute Filter-Performance aufgrund der Datenfusion der Messungen.

Szenario 2

Das Szenario 2 beinhaltet einen Kurvenflug mit zwei Radaren (vgl. Tabelle 6.5).

	RMS_D (in m)						
	Schätzung		Messung			Umfang	
Variante	μ	σ	μ	σ	P_{bias}	Schätzung	Messung
1	29,97	3,76	33,87	1,05	-0,18	17514	1252
2	40,95	4,11	34,67	1,58	-0,82	17256	1246
3	37,09	6,08	34,72	1,38	-0,53	17299	1283
4	38,52	3,32	34,60	1,15	-0,72	17249	1294

Tabelle 6.7: Zusammenfassung der Ergebnisse der Durchläufe aus dem Szenario 2

Auffällig im Szenario 2 ist das schlechte Abschneiden des CA- und CYRPRA-Modells im Vergleich zu den Bewegungsmodellen 1. Ordnung (ohne einen Beschleunigungsanteil im Zustand). Das CYRPRV-Modell bietet die zweitbeste, aber eine verhältnismäßig schlechte Performance mit Blick auf den P_{bias} und im direkten Vergleich zum CV-Modell.

Die besten Schätzungen erfolgen mit dem CV-Modell, was sich auch in der höchsten Anzahl akzeptierter Messungen widerspiegelt. Es ist das einzige Modell, dessen RMS_D der Schätzungen geringer ist als jener der Messungen. Insgesamt zeigt das CV-Modell für die gradlinige und lineare Bewegung das beste Ergebnis, trotz der in allen Varianten durch die Radare R1 und R2 erzeugten und relativ stark verrauschten Messungen mit einem $RMS_D > 33$ m.

Szenario 3

Das Szenario 3 beinhaltet eine komplexe Trajektorie und eine schnelle Zielbewegung mit einem stationären und einem mobilen Radar (vgl. Tabelle 6.5).

In diesem Szenario sind Zielverluste durch die agile Zielbewegung sehr wahrscheinlich, entsprechend ist zusätzlich in die Übersichtstabelle eine Spalte mit der Bezeichnung Loss eingefügt, welche die durchschnittliche Anzahl der Zielverluste für einen Durchlauf in der jeweiligen Variante enthält.

Zielverluste sind insofern unerwünscht, als das sie einen Neustart des Filters forcieren bzw. eine neue Instanz eines Filters erzeugen. Während der Initialisierung eines Filters sind Schätzungen zu dem Ziel nicht möglich, was mit Blick auf die Robustheit einer möglichen Zielverfolgung als sehr nachteilig zu bewerten ist.

		RMS_D (in m)						
		Schätzung		Messung			Umfang	
Variante	Loss	μ	σ	μ	σ	P_{bias}	Schätzung	Messung
1	2,2	38,29	4,61	14,84	1,32	-2,28	5415	273
2	0,2	21,19	1,22	14,19	0,77	-1,36	10129	593
3	0	25,17	1,07	14,18	0,76	-2,28	10588	620
4	0,7	25,96	2,93	13,69	0,96	-2,45	8927	515

Tabelle 6.8: Zusammenfassung der Ergebnisse der Durchläufe aus dem Szenario 3

Konträr zum Szenario 2 zeigt das CV-Modell im Szenario 3 ein sehr schlechtestes Ergebnis. Der Filter verliert bei Nutzung des CV-Modells in diesem Szenario durchschnittlich 2,2-mal das Ziel. Ein Zielverlust erfolgt regelmäßig bei den Kurvenabschnitten. Dadurch kann der Filter nur ca. 50 % der Zeit in einem Durchlauf das Ziel verfolgen (vgl. Tabelle B.3).

Die Modelle mit Beschleunigungsanteil schneiden erneut relativ schlecht ab, wenn auch aus verschiedenen Gründen. Das CA-Modell hat zu 20 % einen Zielverlust während eines Durchlaufs bei sonst im Verhältnis sehr guter Track-Genauigkeit aus Sicht des P_{bias} und RMS_D. Dagegen deutlich abgeschlagen befindet sich das CYRPRA-Modell mit den schlechtesten Werten im P_{bias} und RMS_D und einer 70 % Wahrscheinlichkeit des Zielverlusts während eines Durchlaufs.

In diesem eher realistisch angelegten Szenario zeigt das CYRPRV-Modell ganzheitlich die besten Ergebnisse durch seine hohe Ausfallsicherheit beim Zielverlust und einer verhältnismäßig guten Trackgenauigkeit.

6.4 Ergebnisse

Das CV-Modell zeigt bei der linearen Bewegung des Szenario 1 die besten Ergebnisse, wobei die Linearität eine starke Voraussetzung für das CV-Modell darstellt. Bei deutlich nicht-linearen und eher realistischen Bewegungen wie im Szenario 3 versagt das CV-Modell.

Das CA-Model hat insgesamt durchschnittliche Werte hinsichtlich Trackgenauigkeit und Robustheit. Dies ist vermutlich bedingt durch die schnell wachsenden Fehler bei der zeitlichen Integration der Beschleunigung aus den verrauschten Ortsmessungen.

Das CYRPRA-Modell schneidet in der dafür eigentlich vorgesehenen Domäne der nicht-linearen Bewegungen am schlechtesten ab. Dies hat vermutlich ähnliche Gründen wie beim CA-Modell, nur das dies aufgrund der Komplexität des CYRPRA-Modells deutlicher sichtbar ist.

Ganzheitlich betrachtet ist das CYRPRV-Modell mit dem UKF der beste Kompromiss aus Trackgenauigkeit und Robustheit innerhalb der durchgeführten Szenarien. Diese Kombination erlaubt weitestgehend unabhängig von den zu erwartenden Messqualitäten der Positonen eine zuverlässige Zielverfolgung in einem breiten Spektrum möglicher Zielbewegungen.

7. Fazit

7.1 Bewertung

Mit der Simulationsumgebung konnten in insgesamt 120 Durchläufen und einer Gesamtsimulationszeit von etwas mehr als drei Stunden in drei unterschiedlichen Szenarien ausreichend umfangreiche Datensätze für die systematische Untersuchung der Tracker-Architektur und eines möglichst vielseitigen Verfahrens zur Zielverfolgung erhoben werden.

Der mit der Tracker-Architektur umgesetzte Ansatz der Sensor-Track-Fusion in dezentral organisierten Filtern hat im Rahmen der Evaluation für die vier untersuchten Verfahren die Erwartungen erfüllt. Dies betrifft die Fähigkeiten der Sensorfusion und der Trackgenauigkeit. Die Architektur kann insofern als Fundament für das Design eines Systems mit den gewählten Anforderungen und Eigenschaften dienen.

Die Evaluation zeigt im Vergleich die Eignung und vielseitige Anwendbarkeit des Verfahrens UKF+CYRPRV als Tracker in qualitativ heterogenen Sensorumfeldern und für ein breites Spektrum in Frage kommender Bewegungen bei Luftfahrzeugen. Das CYPPRA-Modell zeigt entgegen der Erwartung deutlich schlechtere Ergebnisse, welche durch Overfitting begründbar sind; das Modell versucht mehr zu beschreiben, als die Messungen erlauben.

7.2 Offene Fragen

Die Evaluation beinhaltete maximal drei relativ stark verrauschte Sensoren mit Messintervallen im Sekundenbereich; typisch für eine breite Palette von in Frage kommender Radare. In diesem Zusammenhang ist eine Untersuchung der Tracker-Architektur mit

- sehr vielen Sensoren,
- sehr kleinen Messintervallen und
- sehr genauen Messungen,

beispielsweise mit einer größeren Anzahl von Feuerleitradaren, interessant. Da jede Messung entsprechende Datenpakete im Sensornetz erzeugt, spielt mit Sicherheit auch der mögliche Effekt von Latenzen und Paketverlusten im Sensornetz eine Rolle, welche hier nicht betrachtet wurden.

Für die Zustandsschätzung von Zielen wurden in dieser Arbeit lediglich Positionsmessungen von Radaren berücksichtigt. Die Tracker-Architektur bietet die Möglichkeit der Anbindung von Sensoren, welche auch andere Größen als eine Position messen können. Dies beinhaltet beispielsweise Doppler-Radare zur Messung von Geschwindigkeiten oder in - aus Sicht des Fokus dieser Arbeit - exotischen Anwendungen gar Accelerometer, welche Beschleunigungsmessungen in das Sensornetz geben könnten. Durch solche zusätzlichen Daten ist eine Integration aus reinen Positionsmessungen in vielen Fällen nicht oder nur in begrenztem Umfang erforderlich, was sich insgesamt positiv auf eine Track-Genauigkeit auswirken sollte.

Potentiell ist ein Vergleich des hier favorisierten Verfahrens mit einem IMM-Ansatz bestehend aus einer Menge von linearen Bewegungsmodellen interessant, um die Überlegenheit des UKF+CYRPRV zu validieren. Die hier erstellten Szenarien bieten dafür eine gute Grundlage. Dies gilt ebenfalls für den in den letzten Jahren für das Tracking-Problem sehr populär gewordenen Particle Filter, welcher im direkten Vergleich mit dem UKF aufgrund der hier betrachteten Problemdomäne keine signifikant besseren Ergebnisse erzielen sollte.

Da letztlich die Erkenntnisse dieser Arbeit auf künstlichen Zielbewegungen, welche mit modellierten Radaren verfolgt wurden, beruhen, könnten Echtdaten mit angepassten Parametrisierungen für konkret zu untersuchende Radartypen weitere Erkenntnisse liefern. Die Simulationsumgebung im Virtual Testbed ist aus diesem Grund vorbereitet für den Import von externen Positionsdaten und bietet die Möglichkeit der Anpassung entsprechender Parametrisierungen.

7.3 Zusammenfassung

Diese Arbeit beleuchtet wesentliche Aspekte für das Design eines Trackers in einer Multisensor-Umgebung.

- Die Grundlage eines Trackers bilden verrauschte Messungen von Radaren, welche diese in ein Sensornetz ausstrahlen. Filter erhalten und verwerten diese zeitlich unregelmäßig ausgestrahlten Messungen und bilden daraus Tracks für die Zielverfolgung des von ihnen akquirierten oder von einer übergeordneten Instanz vorgegebenen Ziels. Dies erfolgt mittels Datenfusion.
- Ein Filter entscheidet im Rahmen der Datenassoziierung bei jeder empfangenen Messung, ob diese für das von ihm verfolgte Ziel relevant ist. Filter nutzen die akzeptierten Messungen, um den von ihnen geschätzten Zustand des Ziel zu korrigieren. Eine möglichst genaue und stabile Zustandsschätzung ist hierbei das Ziel, damit Vorhersagen über die Bewegung des Ziels möglich sind.
- Ein Bewegungsmodell unterstützt den Filter bei der Schätzung. Es bildet die Grundlage für den Prozess, nach dem der Filter seine Schätzungen durchführt.
- Der Einsatz von mehreren Radaren und das Vorhandensein von mehreren Zielen in einem überwachten Gebiet erfordert einen Rahmen bzw. eine dafür geeignete Architektur, welche eine Lagebild-Darstellung ermöglicht.

Die Entwicklung, Implementierung und Evaluation der genannten Aspekte schlagen sich im Virtual Testbed, einer interaktiven und grafischen Python-Anwendung, nieder. Sie ist ein unmittelbares Resultat aus und gleichzeitig wichtiges Werkzeug für diese Arbeit (siehe Anhang A).

Abbildungsverzeichnis

Tabellenverzeichnis

Formelverzeichnis

Literaturverzeichnis

[ABE+] Anderson, Luke, Ankur Bakshi, Kareem Elnahal, Joe Kelly, David Kim, Vikram Modi, Adam Pantel, Joe Park, Alex Schnayder, Alex Sood et al.: *PRINCIPLES OF RADAR TRACKING*.

[BDW+05] Bole, Alan G, William O Dineley, Alan Wall et al.: *Radar and ARPA manual*. Elsevier Butterworth Heinemann, 2005.

[BSCK01] Bar-Shalom, Y, H Chen und T Kirubarajan: *Performance limits of track-to-track fusion vs. centralized estimation: Theory and application*. IEEE Transactions on Aerospace and Electronic Systems, 39(2):386–400, 2001.

[BSLS00] Beugnon, Céline, Tarunraj Singh, James Llinas und Rajat K Saha: *Adaptive track fusion in a multisensor environment*. In: *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*, Band 1, Seiten TUC2–24. IEEE, 2000.

[C+03] Chen, Zhe et al.: *Bayesian filtering: From Kalman filters to particle filters, and beyond*. Statistics, 182(1):1–69, 2003.

[CMBC00] Chong, Chee-Yee, Shozo Mori, William H Barker und Kuo-Chu Chang: *Architectures and algorithms for track association and fusion*. IEEE Aerospace and Electronic Systems Magazine, 15(1):5–13, 2000.

[Die06] Diebel, James: *Representing attitude: Euler angles, unit quaternions, and rotation vectors*. Matrix, 58(15-16):1–35, 2006.

[DP15] Dunn, Fletcher und Ian Parberry: *3D math primer for graphics and game development*. CRC Press, 2015.

[Far12] Faragher, Ramsey: *Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes]*. IEEE Signal processing magazine, 29(5):128–132, 2012.

[Fou15] Fourati, Hassen: *Multisensor Data Fusion: From Algorithms and Architectural Design to Applications*. CRC Press, 2015.

[FR99] Fitzgerald, WJ und PJW Rayner: *Bayesian signal processing*. In: *IEE Colloquium (Digest)*, Seiten 35–40, 1999.

[Gen01] Genovese, Anthony F: *The interacting multiple model algorithm for accurate state estimation of maneuvering targets*. Johns Hopkins APL technical digest, 22(4):614–623, 2001.

[JK13] Jan, Shau-Shiun und Yu-Chun Kao: *Radar tracking with an interacting multiple model and probabilistic data association filter for civil aviation applications*. Sensors, 13(5):6636–6650, 2013.

[K+60] Kalman, Rudolph Emil et al.: *A new approach to linear filtering and prediction problems*. Journal of basic Engineering, 82(1):35–45, 1960.

[KM97] Kalata, P. R. und K. M. Murphy: *Alpha-Beta Target Tracking and Track Rate Variations*. In: *Proceedings of the 29th Southeastern Symposium on System Theory (SSST '97)*, SSST '97, Seiten 70–, Washington, DC, USA, 1997. IEEE Computer Society.

[KM07] Klussmann, Niels und Arnim Malik: *Lexikon der Luftfahrt*. Springer-Verlag, 2007.

[Lab15] Labbe, RR: *Kalman and bayesian filters in python*. 2015.

[LL15] Linge, Svein und Hans Petter Langtangen: *Programming for Computations-A Gentle Introduction to Numerical Simulations with Python*, 2015.

[MDW95] Manyika, James und Hugh Durrant-Whyte: *Data Fusion and Sensor Management: a decentralized information-theoretic approach*. Prentice Hall PTR, 1995.

[Mit07] Mitchell, Harvey B: *Multi-sensor data fusion: an introduction*. Springer Science & Business Media, 2007.

[MS15] Malthe-Sørenssen, Anders: *Elementary mechanics using Python*. Cham: Springer, 2015.

[RHG14] Roth, Michael, Gustaf Hendeby und Fredrik Gustafsson: *EKF/UKF maneuvering target tracking using coordinated turn models with polar/Cartesian velocity*. In: *Information Fusion (FUSION), 2014 17th International Conference on*, Seiten 1–8. IEEE, 2014.

[SBMH16] Siegert, Gregor, Paweł Banyś, Cristina Sáez Martínez und Frank Heymann: *EKF based trajectory tracking and integrity monitoring of AIS data*. In: *Position, Location and Navigation Symposium (PLANS), 2016 IEEE/ION*, Seiten 887–897. IEEE, 2016.

[SRW08] Schubert, Robin, Eric Richter und Gerd Wanielik: *Comparison and evaluation of advanced motion models for vehicle tracking*. In: *Information Fusion, 2008 11th International Conference on*, Seiten 1–6. IEEE, 2008.

[SSCB14] Stone, Lawrence D, Roy L Streit, Thomas L Corwin und Kristine L Bell: *Bayesian multiple target tracking*. Artech House, 2014.

[TPA05] Tsogas, Manolis, Aris Polychronopoulos und Angelos Amditis: *Unscented Kalman filter design for curvilinear motion models suitable for automotive safety applications*. In: *Information Fusion, 2005 8th International Conference on*, Band 2, Seiten 8–pp. IEEE, 2005.

[WVDM00] Wan, Eric A und Rudolph Van Der Merwe: *The unscented Kalman filter for nonlinear estimation*. In: *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, Seiten 153–158. Ieee, 2000.

[YJX16] You, He, Xiu Jianjuan und Guan Xin: *Radar data processing with applications*. John Wiley & Sons, 2016.

[Zar05] Zarchan, Paul: *Progress In Astronautics and Aeronautics: Fundamentals of Kalman Filtering: A Practical Approach*, Band 208. Aiaa, 2005.

A. Virtual Testbed

Das Virtual Testbed ist ein in Python 3.5 implementiertes Programm, welches im Rahmen dieser Arbeit entstand und für die Arbeit eine zentrale Rolle einnimmt. Dieser Abschnitt gibt einen Überblick über das Programm, dessen Zweck und Bedienung.

Beschreibung

Das Virtual Testbed ist eine **Simulationsumgebung** für realistische Lagen mit Land-, See- und Flugzielen und Radarträgern in geografisch definierten Gebieten. Es dient ferner als interaktive **Testumgebung** für implementierte Filter und Bewegungsmodelle im zwei- und dreidimensionalen Raum.

Das Programm beinhaltet im Kern die drei im theoretischen Teil der Arbeit entwickelten Anteile als Komponenten in schwacher Kopplung:

- Tracker-Architektur
- Framework für die ereignisorientierte Simulation
- 2D- und 3D-Visualisierung

Eine vierte Komponente in Form einer textuellen **Schnittstelle** steht für quantitative Betrachtungen einer Filter-Performance zur Verfügung. Diese liefert zeitlich markierte Datensätze, welche vor allem für die Datenerhebung im Rahmen der Evaluation zum Einsatz kam.

Installation und Betrieb

Eine **Python 3.5**-Laufzeitumgebung sowie die Bibliotheken **NumPy** 1.12, **SciPy** 0.18, **SymPy** 1.0 und **matplotlib** 2.0 sind für den Betrieb des Virtual Testbed erforderlich.

Aus Gründen der Portabilität befinden sich für Windows 64-bit und GNU/Linux 64-bit lokale Laufzeitumgebungen inklusive aller benötigten Bibliotheken im Programmverzeichnis.

Der Aufruf des Programms kann aus einem beliebigen und nicht-schreibgeschützten Verzeichnis erfolgen. Es gibt sonst keine besonderen Voraussetzungen für den Betrieb des Programms.

- **Windows**: Der Programmaufruf erfolgt über die Datei `start.exe`; die Laufzeitumgebung befindet sich im Verzeichnis `run/win64/`.
- **GNU/Linux**: Der Aufruf des Programms erfolgt mittels einer Bash aus dem Programmverzeichnis mit `./start.sh`; die entsprechende Laufzeitumgebung befindet sich im Verzeichnis `run/lin64/`.

Programmverzeichnis

Das Programmverzeichnis beinhaltet alle Dateien zum Betrieb des Programms. Die Python-Skripte im Verzeichnis /src umfassen etwas mehr als $\sim$ 5.000 NCLOC.

Pfad		**Beschreibung**
cfg/		Definitionen und Parametrisierungen (im JSON-Format)
marker/		Geografische Fixpunkte
motion/		Bewegungsmuster (Geschwindigkeit, Beschleunigung, Ruck über Zeit)
sensory/		Radarkonfigurationen (Drehate, Reichweite, Messfehler)
track/		Trajektorien
res/		Ressourcen
map/		Hintergrundbilder für die geografischen Gebiete
run/		Python 3.5.2-Laufzeitumgebung mit benötigten Bibliotheken
lin64/		Laufzeitumgebung für GNU/Linux 64-bit
win64/		Laufzeitumgebung für Windows 64-bit
src/		Quellcode
base/		Module mit grundlegenden Funktionen und Datenstrukturen
	dat.py	Abstraktionen für alle Objekte mit einem Positionsbezug
	env.py	Implementierung der Filter, Sensoren, Modelle und Zustände
	ext.py	Schnittstelle zu Defintionen und Parametrisierungen (cfg/)
	gui.py	Basisklassen für alle grafischen Steuerelemente
	ply.py	Testing für 2D und 3D
	utl.py	(Hilfs-)Funktionen
ui/		Module für die grafische Benutzeroberfläche und -interaktion
	app.py	GUI-Hauptprogramm
	art.py	Akteure und deren Symbolik für die 2D- und 3D-Darstellung
	map.py	Karten-Ansicht (2D)
	plt.py	Plot-Ansicht (3D)

Benutzeroberfläche

Die Bedienung erfolgt in der grafischen Benutzeroberfläche (GUI) mit einer Maus mit Mausrad als mittlere Maustaste oder einem Eingabegerät mit äquivalenten Eingabemöglichkeiten.

Das Programm ist in vier Bereiche unterteilt, welche über die Navigation im oberen Teil der GUI auswählbar sind: **Simulation**, **Testing (2D)**, **Testing (3D)** und **Monitoring**.

Diese Bereiche sind kontextabhängig wiederum in Bereiche aufgeteilt, welche ebenfalls über eine Navigation im oberen Teil bzw. linken Teil auswählbar sind.

Simulationsumgebung

Im initial ausgewählten Bereich **Simulation** besteht die Möglichkeit beliebige Lagen in einem oder mehreren geografisch definierten Gebieten zu erstellen und zu simulieren.

Die verfügbaren Gebiete unterscheiden sich räumlich neben ihrer Position auf der Erdoberfläche auch in ihrer Ausdehnung, um Lagen in verschiedenen Größenordnungen erstellen und simulieren zu können:

- Playground (ca. 8 x 8 x 8 km)
- Ostfriesland (ca. 80 x 60 x 10 km)
- Schrobenhausen (ca. 30 x 20 x 3 km)
- Pazifik (ca. 10 x 5 x 1 km)

Die zeitliche Steuerung der Simulation erfolgt über den [Clock]-Knopf im rechten Teil der GUI. Der [Apply]-Button im unteren Teil übernimmt je nach zeitlichem Zustand die Start- und Stopp-Funktion.

Map

In der zweidimensionalen Ansicht dient die Karte in der Mitte neben der Darstellung der Lage aus der Vogelperspektive auch als Editor für die Erstellung von Lagen. Im rechten Teil der GUI sind die aktuelle Simulationszeit sowie die aktuellen Bewegungsparameter aller Ziele in der Lage ersichtlich.

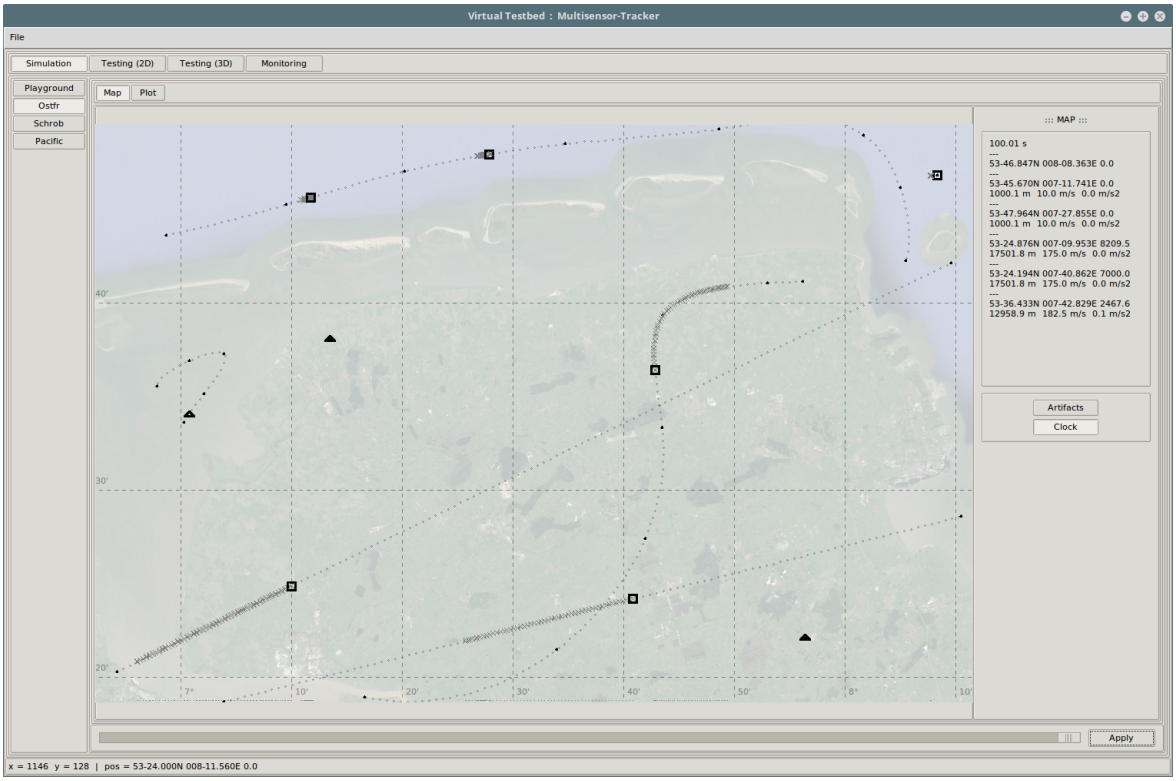


Abbildung A.1: 2D-Darstellung einer Lage auf der Ostfriesland-Karte; drei Radare, sechs Ziele

Die Erstellung einer Lage beinhaltet stets die Erzeugung von geografischen **Referenzpunkten** in der Karte, die Realisierung von **Artefakten** anhand der erstellten Referenzpunkte und die Zuordnung von **Akteuren** zu Artefakten. Diese Vorgehensweise erzeugt eine große Flexibilität für die Anordnung der Objekte in einer Lage bei gleichzeitig minimalem Aufwand für die interaktive Erstellung einer Lage.

Referenzpunkte erstellen

Ein Klick der rechten Maustaste in die Karte erzeugt oder löscht an der Position des Cursors einen Referenzpunkt. Referenzpunkte sind mit einer roten Nummer dargestellt, welche Aufschluss über die Reihenfolge ihrer Erstellung geben.

Die Höhe eines Referenzpunkts ist mit dem Mausrad veränderbar, wobei der Mauscursor sich über oder in unmittelbarer Nähe zu dem jeweiligen Referenzpunkt befinden muss. Die Anzeige der Höhe erfolgt durch eine weitere rote Nummer, welche den Wert der Höhe in Meter über Normalhöhe (über der xy-Ebene) angibt; initial befinden sich Referenzpunkte in einer Höhe von 0 Metern (auf der xy-Ebene).

Artefakte realisieren

Ein Artefakt beschreibt innerhalb einer Lage eine Position (Marker) oder eine Trajektorie (Track). Die Erzeugung von Artefakten erfolgt mit Hilfe von Referenzpunkten. Ein Klick der mittleren Maustaste in die Karte realisiert Artefakte aus einem oder mehreren gesetzten Referenzpunkten.

Ein Referenzpunkt erzeugt einen Marker, mehrere Referenzpunkte bilden die Wegpunkte einer Route, welche daraufhin iterativ einen Track mit äquidistanten Abschnitten erzeugt.

- Marker sind als schwarze Punkte in der Karte erkennbar.
- Tracks bestehen aus mehreren schwarzen Punkten, welche mit einer grauen Linie verbunden sind.

Die Auswahl eines Artefakts erfolgt mit einem Klick der linken Maustaste auf einen schwarzen Punkt in der Karte. Die Abwahl eines Artefakts erfolgt durch einen Klick mit der linken Maustaste in einen freien Bereich auf der Karte.

Bei aktiver Auswahl eines Artefakts ist die Erstellung von Referenzpunkten nicht möglich.

Vordefinierte Artefakte sind in JSON-Dateien in den Verzeichnissen `cfg/marker/` und `cfg/track/` abgelegt. Ein Import dieser Artefakte ist über den [Artifacts]-Knopf im rechten Teil der GUI möglich. Nach Auswahl eines Artefakts erzeugt ein Klick auf den [Apply]-Knopf im unteren Teil der GUI das Artefakt in der Karte, sofern die Positionen des Artefakts in dem Gebiet der aktuell ausgewählten Karte liegen.

Akteure zuordnen

Akteure sind Ziele (Target) und Radare (Radar), welche in einer Lage existieren. Jeder Akteur benötigt für seine Existenz ein Artefakt, welches dem Akteur zu jeder Zeit eine Ortsbestimmung in der Lage ermöglicht.

Nach Auswahl eines Artefakts in der Karte besteht im rechten Teil der GUI die Möglichkeit über den [Target]- oder [Radar]-Knopf einen Akteur auszuwählen, um diesem das ausgewählte Artefakt zuzuordnen.

- Ziele und Radare benötigen bei der Zuordnung zu einem Track stets die Angabe eines Bewegungsmusters (Motion), welche sich in den JSON-Dateien im Verzeichnis `cfg/motion/` befinden.
- Radare benötigen zusätzlich eine Radarkonfiguration, welche ebenfalls in JSON-Dateien in dem Verzeichnis `cfg/sensory/` hinterlegt sind.

Der Schieberegler im unteren Teil der GUI erlaubt die Positionierung eines noch nicht realisierten Akteurs (rotes Symbol) auf einem beliebigen Punkt eines ihm zugeordneten Tracks. Ein Klick auf den [Apply]-Knopf realisiert den Akteur (schwarzes Symbol).

Plot

Die dreidimensionale Ansicht dient lediglich zur Visualisierung der Lage in dem ausgewählten Gebiet; Änderungen der Lage können dort nicht durchgeführt werden. Die Darstellung der Artefakte und Akteure ist ähnlich zu der Darstellung in der zweidimensionalen Ansicht.

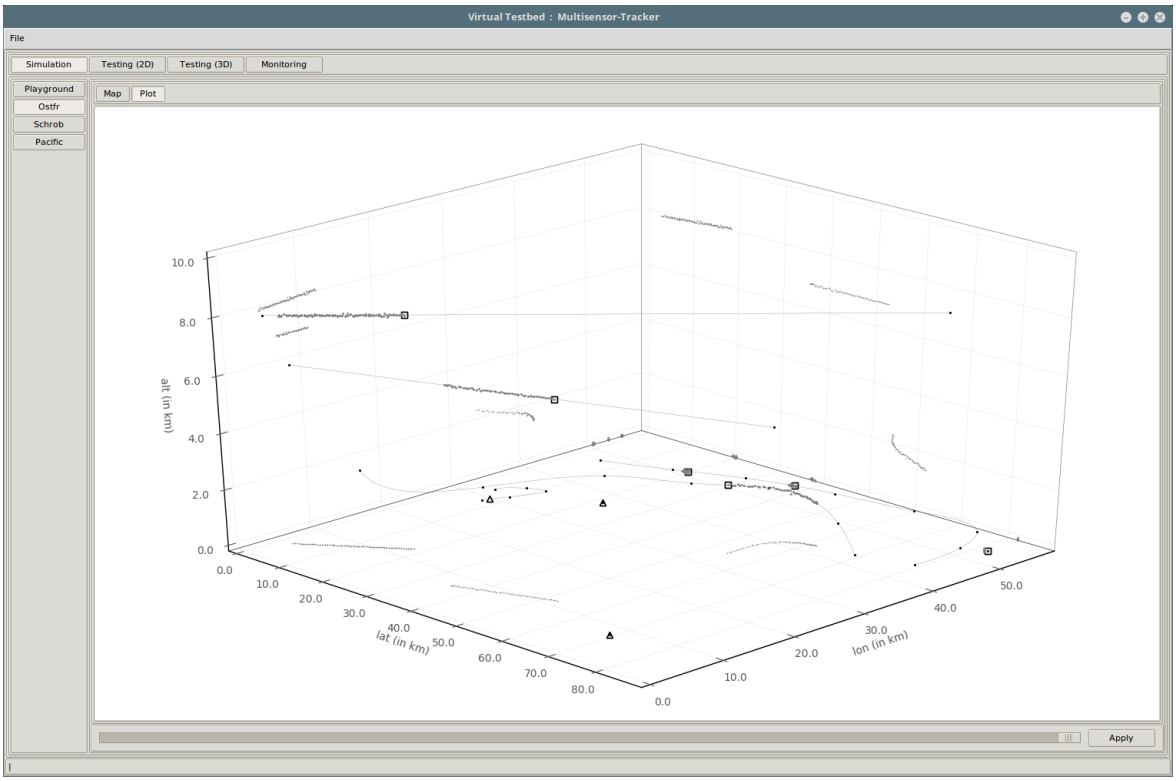


Abbildung A.2: 3D-Darstellung einer Lage auf der Ostfriesland-Karte; drei Radare, sechs Ziele

Ein Klick auf die mittlere Maustaste in das Plot wechselt die zu rotierende Achse (Pan bzw. Tilt). Die Rotation der Ansicht erfolgt gradweise mit dem Mausrad. Ein Doppelklick mit der linken Maustaste in das Plot erzeugt einen Screenshot des Plots und speichert diesen im Programmverzeichnis als PDF-Datei.

Testumgebung

Die Testumgebung erlaubt interaktive "on-the-fly"-Betrachtungen und -Bewertungen der implementierten Filter und Modelle unter Nutzung einzelner, mehrerer und unterschiedlich parametrisierter Sensoren. Der **Mauscursor** dient dabei als zu trackendes Objekt.

Die Testumgebung ist aufgeteilt in **Testing (2D)** zum Test von Filtern und Bewegungsmodellen auf der Ebene und **Testing (3D)** zum Test in einem von einer Ebene abgeleiteten Raum.

Im Bereich **Fusion** gehen alle Messungen der ausgewählten Sensoren in den Filter ein; es gibt kein Ursprungsproblem bzw. aktives Gate im Filter. Im Bereich **Association** besteht das Ursprungsproblem durch die Erzeugung eines weiteren Ziels. Der Filter entscheidet anhand seines Gate, ob Messungen für die Filterung zu akzeptieren oder zu verwerfen sind.

Sensoren und Filter

Die Aktivierung bzw. Deaktivierung von Sensoren erfolgt im rechten Teil der GUI durch Aus- und Abwahl mittels der entsprechenden Knöpfe.

Die Initialisierung und Beendigung eines Filters erfolgt mit einem Klick der mittleren Maustaste auf die weiße Fläche mit dem Koordinatenkreuz. Die rechte Maustaste pausiert den Filter für Momentanbetrachtungen. Die Internas des aktiven Filters sind unterhalb der Sensorauswahl dargestellt.

Darstellung

Die tatsächliche Bewegung des Mauscursors erzeugt eine graue, durchgezogene Spur. Die Anzeige akzeptierter Messungen erfolgt durch einen dunkelgrauen Punkt, verworfene Messungen sind hellgrau. Die Schätzungen des aktiven Filters zur tatsächlichen Bewegung des Mauscursors basierend auf den akzeptierten Messungen sind durch rote Punkte dargestellt.

Testing (2D)

Die folgenden Filter stehen mit entsprechenden Bewegungsmodellen zum Test dieser im zweidimensionalen Fall zur Verfügung:

- Kalman Filter (KF): CJ, CT, CA, CV, BM
- Extended Kalman Filter (EKF): CTRV, CTRA
- Unscented Kalman Filter (UKF): CTRV, CTRA

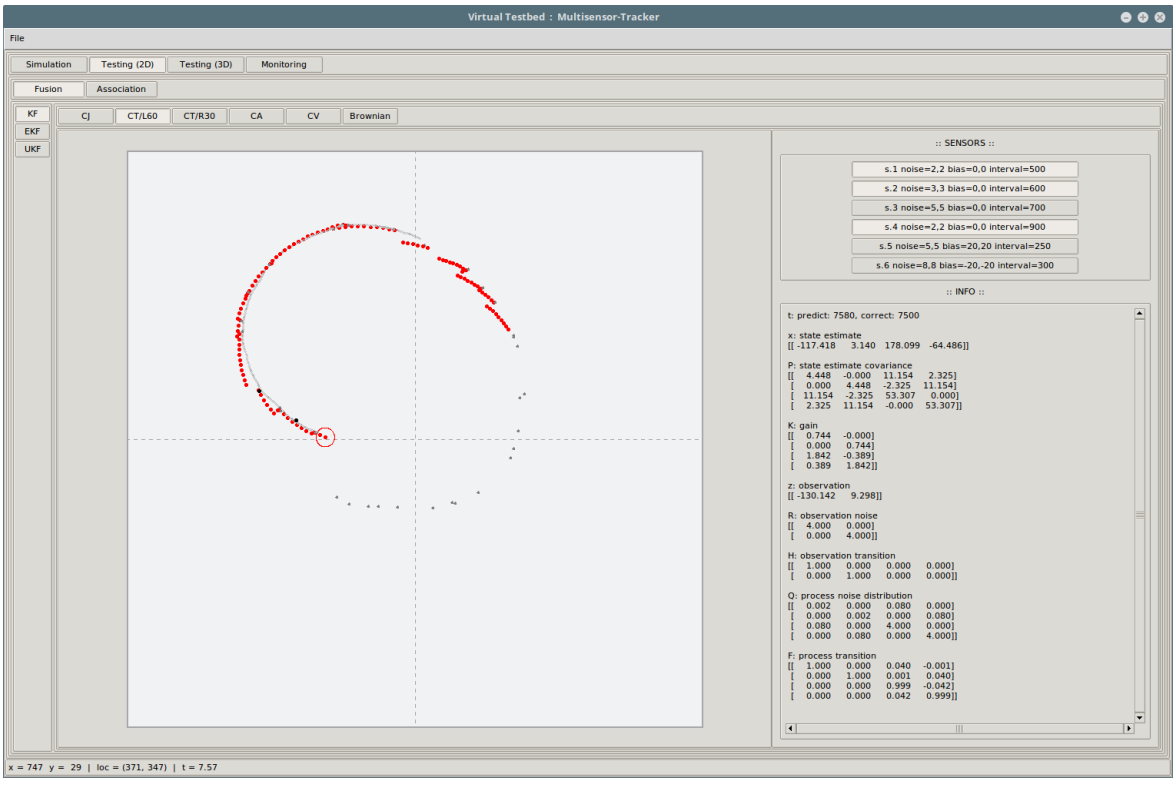


Abbildung A.3: CT-Modell mit einem KF auf der Ebene; drei Sensoren, ein Ziel

Testing (3D)

Der Unscented Kalman Filter (UKF) dient als Filter für die implementierten 3D-Bewegungsmodelle. Die Erzeugung des Raums erfolgt über Erweiterung der Koordinatenkreuz-Fläche um eine künstliche z-Koordinate, deren Wert aus den Werten der x- und y-Koordinate ermittelbar ist.

Insgesamt stehen fünf Bewegungsmodelle für den dreidimensionalen Fall zur Verfügung:

- Nicht-lineare Modelle: CYRPRV, CYRPRA
- Lineare Modelle: CV, CA, CJ

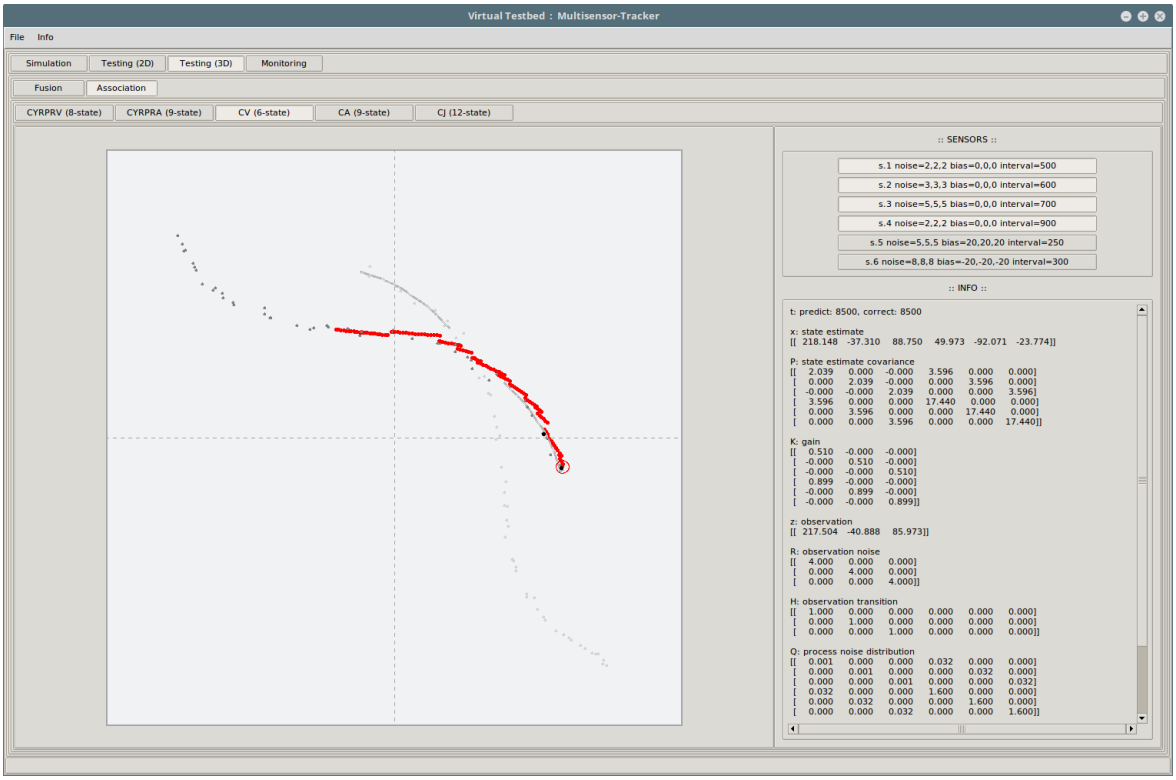


Abbildung A.4: CV-Modell mit einem UKF im Raum; vier Sensoren, zwei Ziele

Schnittstelle

Der Bereich **Monitoring** fungiert als einfache Schnittstelle mit zwei unterschiedlichen Sichten für die Erhebung von Datensätzen für weiterführende Auswertungen. Die textuellen Datensätze aller Textfelder sind nach einem Klick in das entsprechende Textfeld mit den Tastenkombinationen Ctrl-A und Ctrl-C in die Zwischenanlage kopierbar.

Im **Tracker** sind alle Filter in chronologischer Reihenfolge ihrer Erstellung auswählbar. Nach Auswahl eines Filters sind alle protokollierten Daten zu diesem Filter einsehbar. Unter **Network** sind die Messungen aller Sensoren in der Reihenfolge ihrer Erstellung zu sehen.

Im Monitoring erfolgen Zeitangaben stets in der Einheit Sekunde und Winkelangaben im Gradmaß. Positionsangaben können je nach Herkunft des Filters in Dimension und Einheit variieren.

- Filter, welche in der **Testumgebung** aktiv sind, erhalten zwei- oder dreidimensionale, absolute Pixel-Koordinaten des Mauscursors von den dortigen Sensoren. Entsprechend finden Positionsschätzungen solcher Filter in der Einheit Pixel statt.
- Radare erzeugen relative und polare Messungen innerhalb der **Simulationsumgebung**. Diese Messungen erfahren eine Transformation in dreidimensionale, kartesische Koordinaten, bevor sie zu einem Filter gelangen. Positionsangaben von Filtern aus der Simulationsumgebung liegen somit immer in drei Dimensionen und der Einheit Meter vor.

Tracker

Jeder Filter innerhalb des Virtual Testbed, egal ob in der Test- oder Simulationsumgebung instanziiert, erzeugt eine Auswahlmöglichkeit zu sich im linken Teil der GUI. Die ein- und ausgehenden Daten eines Filters sind in neun separaten Textfeldern dargestellt.

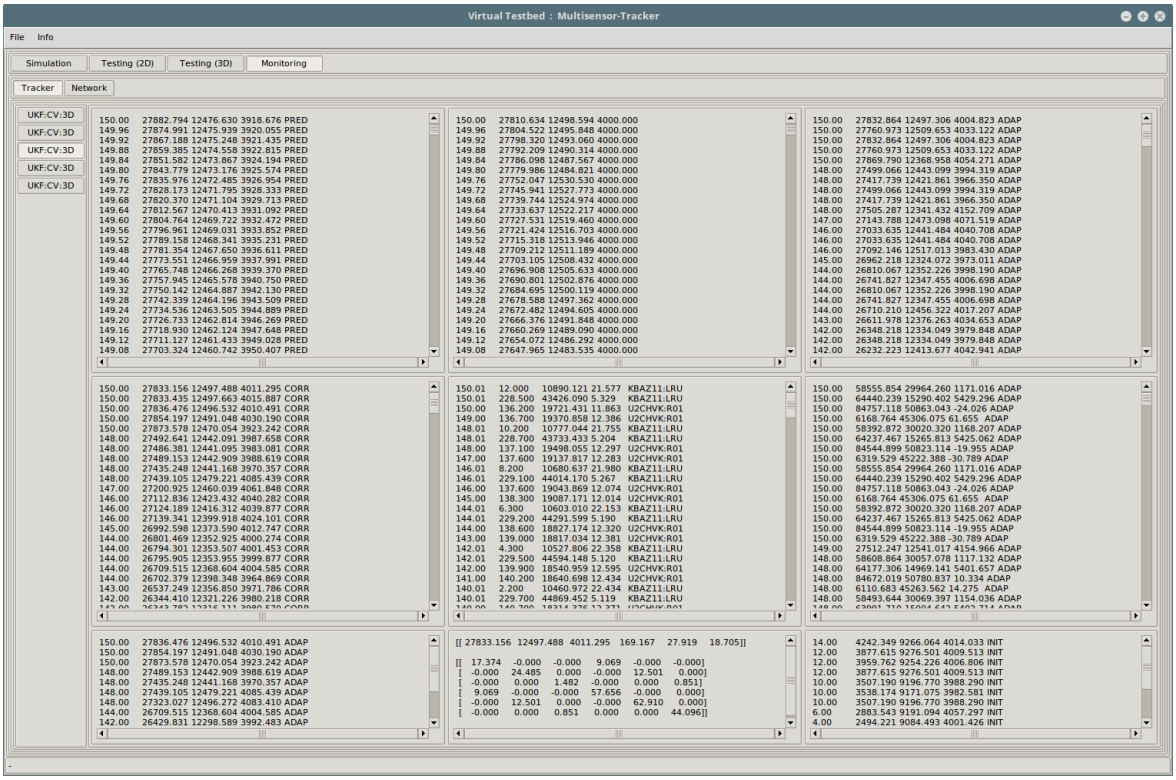


Abbildung A.5: Filter-Schnittstelle für die Erhebung von dedizierten Datensätzen

- Das vom Filter verfolgte Ziel gibt in bestimmten Intervallen seine wahre Position bekannt. Das **obere mittlere** Textfeld zeigt diese Positionen mit einem Zeitstempel an. Der Filter kennt diese Positionen nicht.
- Das **mittige** Textfeld erhält nur Messungen von Radaren, welche auch Messungen an den gewählten Filter liefern. Eine Zeile in dem Textfeld beinhaltet Zeitstempel, Azimut, Distanz und Elevation zum verfolgten Ziel sowie eine eindeutige Kennung des Radars.
- Im **unteren mittleren** Textfeld kommt die aktuelle Schätzung des Zustandsvektors und der Kovarianzmatrix zur Anzeige.

Jede Textzeile der nachfolgend aufgezählten und auf der linken Seite der GUI angeordneten Textfelder enthält einen Zeitstempel, die geschätzte Position in 2D oder 3D und ein Literal.

- Im **oberen linken** Textfeld befinden sich die Positionsschätzungen des Filters zu dem verfolgten Ziel nach jedem Prädiktionsschritt.
- Das **mittlere linke** Textfeld zeigt die Positionsschätzungen nach jedem Korrekturschritt. Ein Korrekturschritt nutzt zur Korrektur der Prädiktion die zuletzt aufgelaufenen und akzeptierten Messungen.
- Im Fall einer Adaption erzeugt der Filter einen Eintrag mit der letzten Positionsschätzung im **unteren linken** Textfeld.

Messungen dienen der Initialisierung und Korrektur eines Filters. In den rechts angeordneten Textfeldern befinden sich gruppiert alle Messungen, die ein Filter erhält und verarbeitet. Der Repräsentation der Messungen erfolgt zeilenweise mit Zeitstempel, der Positionsmessung in 2D oder 3D und einem Literal:

- Das **obere rechte** Textfeld erzeugt eine Anzeige aller vom Filter akzeptierten Messungen.
- Die verworfenen Messungen laufen im **mittleren rechten** Textfeld auf.
- Im **unteren rechten** Textfeld befinden sich diejenigen Messungen, welche für die Initialisierung des Filters verantwortlich sind.

Network

Das Network bietet in erster Linie zu Debug-Zwecken die Möglichkeit zur Einsicht des Datenverkehrs im Sensornetz. Alle Messungen der Sensoren im Virtual Testbed erzeugen einen Eintrag in diesem Protokoll.

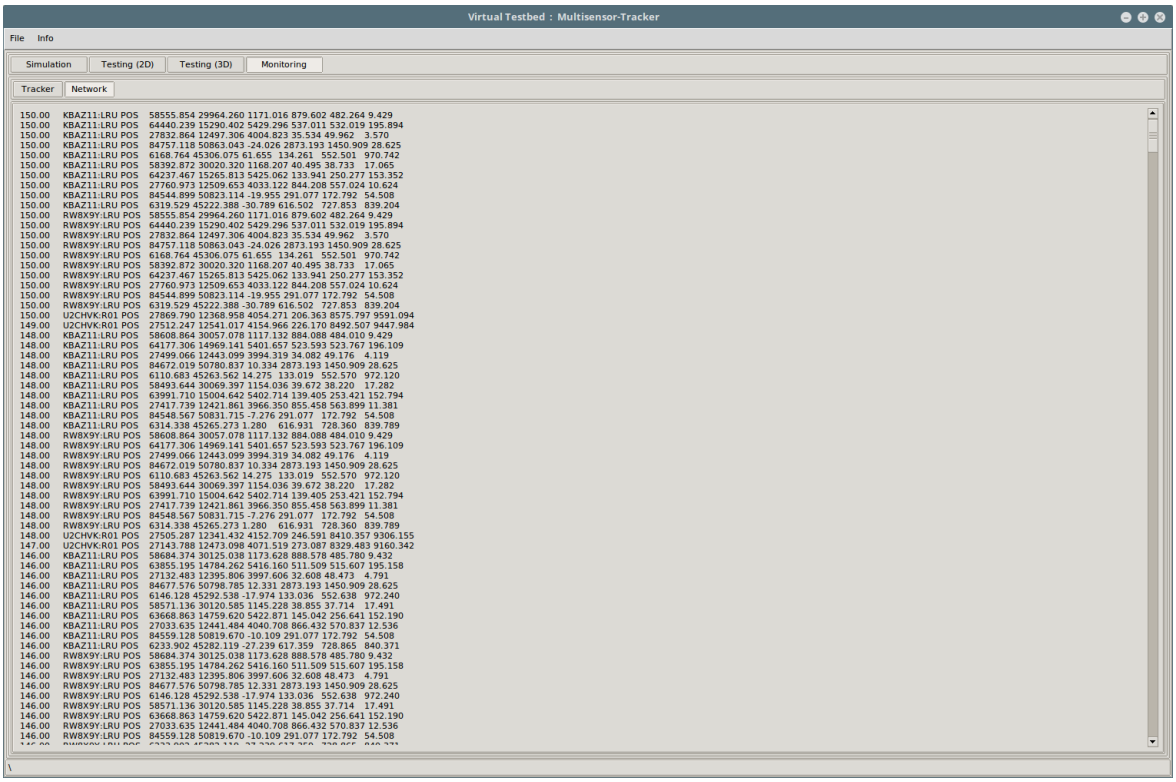


Abbildung A.6: Überwachung des Datenverkehrs im Sensornetz

Jede Zeile im Protokoll beinhaltet die Messung eines Sensors zu einem Ziel: Zeitstempel, Kennung des Sensors, gemessene physikalische Größe, n-dimensionaler Messwert und n-dimensionale Varianz des Messrauschens vom Sensor.

B. Evaluation

Dieser Abschnitt beinhaltet in tabellarischer Form die Einzel-Auswertungen der Durchläufe aus der Evaluation. Zur Einordnung der Ergebnisse steht zu jedem Szenario stellvertretend ein Durchlauf als visuelle Unterstützung zur Verfügung.

Die insgesamt 120 Durchläufe der drei Szenarien sind vierstellig mit einer Laufnummer kodiert: Die Tausenderstelle bezeichnet das Szenario, die Hunderterstelle die Variante und die beiden niederwertigen Stellen bilden eine fortlaufende Laufnummer.

Die Tabellen enthalten zeilenweise einen Durchlauf, welcher spaltenweise von links nach rechts wie folgt beschrieben ist:

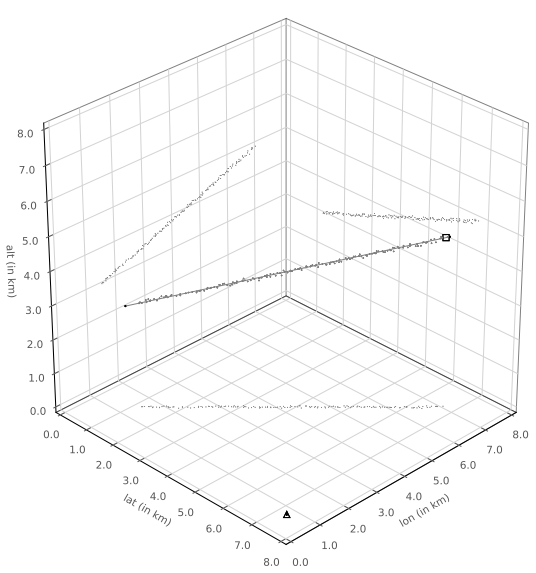
- Laufnummer
- Anzahl der Zielverluste
- Abdeckung
  - Zeitspanne, in welcher der Filter durchgängig Schätzungen durchgeführt hat
  - Verhältnis der Zeitspanne zur gesamten Laufzeit
- Messungen
  - Anzahl der Messungen, die der Filter akzeptiert hat
  - Anzahl der Messungen, die der Filter verworfen hat
- Effektivwerte der Abweichungen
  - von wahren Positionen zu Schätzungen
  - von wahren Positionen zu Messungen
- Prozentualer Volumenfehler

Die Anzahl der Schätzungen m ist aus der angegebenen Zeit t eines Durchlaufs durch den einfachen Zusammenhang $m = \frac{t}{\Delta t}$ mit $\Delta t = \frac{1}{25}$ ermittelbar, da alle Filter im Rahmen der Evaluation ein Intervall von 40 ms für die Durchführung einer einzelnen Schätzung nutzen.

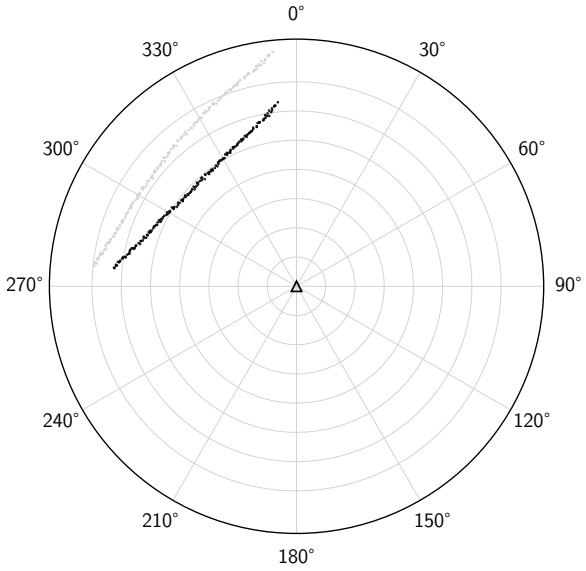
Die Datensätze inklusive der zu jedem Durchlauf erstellten Abbildungen befinden sich auf dem dieser Arbeit beigelegten Datenträger (siehe Anhang C).

Szenario 1a

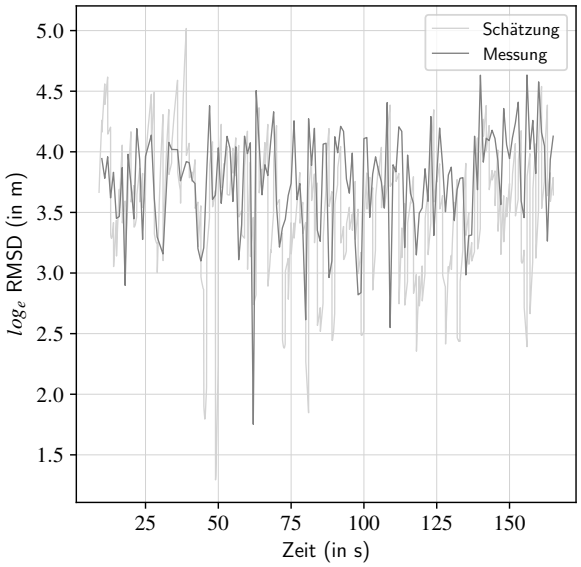
Exemplarische Darstellung



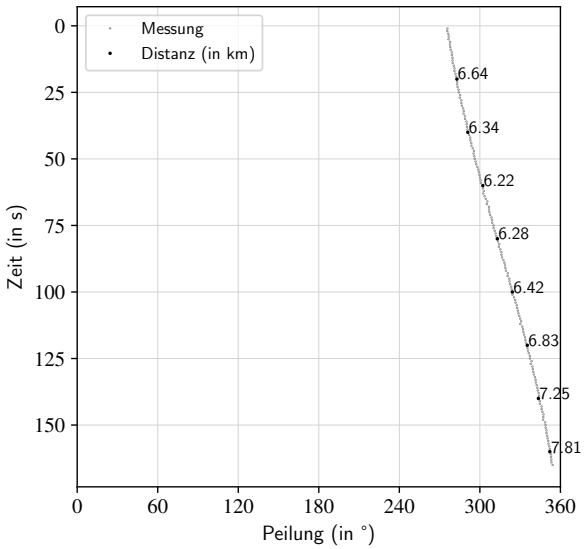
(a) 3D-Plot der Lage



(b) Radar-Plot, kombinierte Ansicht



(c) RMS_D über die Zeit

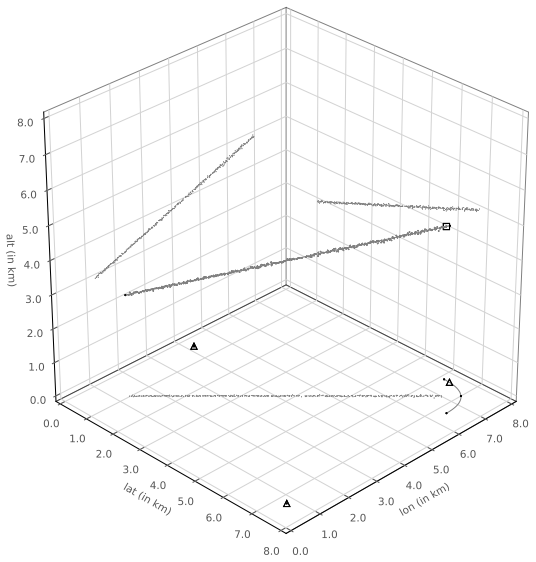


(d) Entfernungen mit Peilung und Zeit

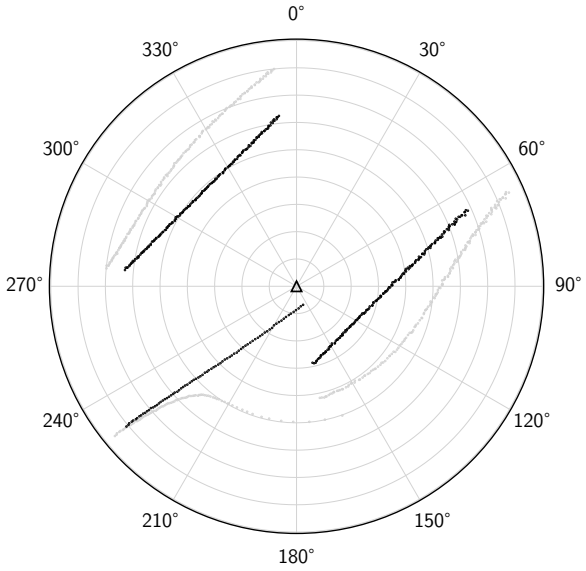
Abbildung B.1: Exemplarische Darstellung der Szenarien 1.1 bis 1.3; hier Durchlauf 1101

Szenario 1b

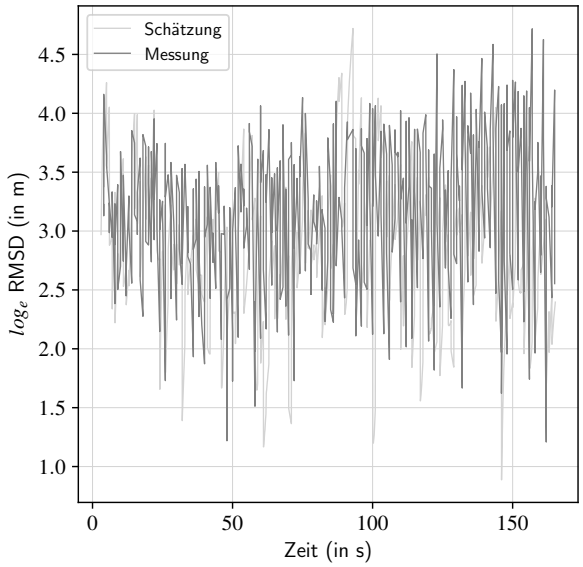
Exemplarische Darstellung



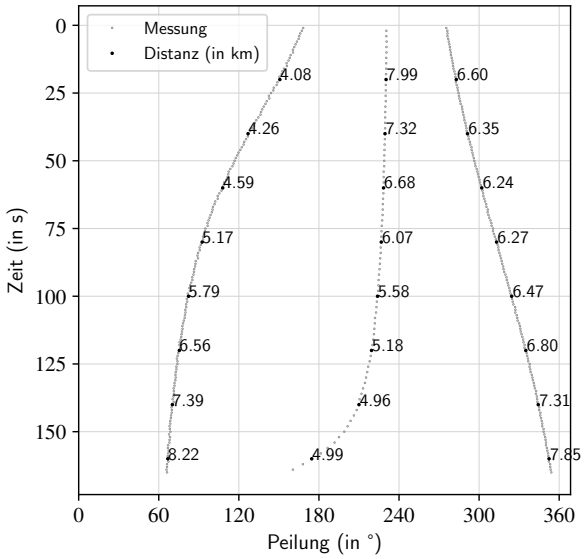
(a) 3D-Plot der Lage



(b) Radar-Plot, kombinierte Ansicht



(c) RMS_D über die Zeit



(d) Entfernungen mit Peilung und Zeit

Abbildung B.2: Exemplarische Darstellung des Szenarios 1.4; hier Durchlauf 1401

Szenario 1

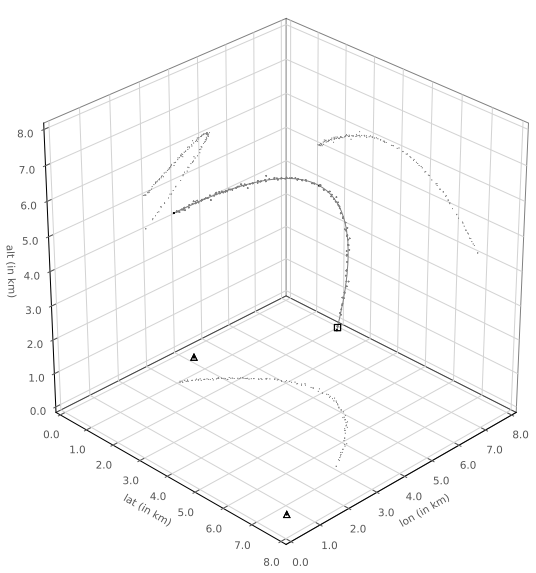
Ergebnisse der Durchläufe

		Abdeckung		Messung		RMS_D (m)		
Lauf	Loss	Zeit (s)	Zeit (%)	akzep.	verw.	Schätzung	Messung	P_{bias}
1101	0	156	94,55	151	5	36	44,2	0,07
1102	0	151,2	91,53	146	5	34,2	40,3	0,04
1103	0	110,2	66,71	110	0	35,2	42,4	0,01
1104	0	99,2	60,05	99	0	32	39	0,04
1105	0	111,2	67,31	110	1	33,7	39,8	0,05
1106	0	158,2	95,76	150	8	33,3	39,2	0,01
1107	0	131,16	79,41	127	4	33,4	40,2	0,04
1108	0	151,12	91,52	148	3	29,3	37,7	0,11
1109	0	159,12	96,37	156	3	30,7	37,1	0,08
1110	0	127,24	77	125	2	32	40,4	0,09
1201	0	160,12	96,97	155	5	23,7	24,4	-0,16
1202	0	159,2	96,37	154	5	23,2	24,4	-0,11
1203	0	159,2	96,37	154	5	23,9	25,9	-0,1
1204	0	157,2	95,16	151	6	24,7	25,1	-0,15
1205	0	160,28	96,97	159	1	23,7	25,5	-0,08
1206	0	161,24	97,58	160	1	20,8	23,1	-0,07
1207	0	161,28	97,58	154	7	24,1	25,3	-0,14
1208	0	159,24	96,37	156	3	21,1	23,3	-0,07
1209	0	161,24	97,58	157	4	23,4	24,2	-0,16
1210	0	159,24	96,37	157	2	23,3	25	-0,11
1301	0	157,2	95,16	78	0	14,1	10,2	-0,93
1302	0	157,24	95,16	78	0	13,7	9,7	-0,99
1303	0	157,2	95,16	78	0	14,7	10	-1,07
1304	0	159,2	96,37	79	0	14	10,1	-0,94
1305	0	157,24	95,16	78	0	16,2	11,2	-1,06
1306	0	159,2	96,37	79	0	15	10,7	-0,97
1307	0	159,24	96,37	79	0	15	10,7	-1,03
1308	0	157,2	95,16	78	0	13	9,6	-0,87
1309	0	159,2	96,37	79	0	14,9	10,4	-1,05
1310	0	157,2	95,16	78	0	15	11	-0,9
1401	0	162,2	98,18	390	15	18,7	25,5	0,07
1402	0	162,16	98,18	391	14	15,2	23,5	0,17
1403	0	162,16	98,18	391	14	16,3	25,1	0,18
1404	0	162,08	98,18	391	14	14,8	23,2	0,18
1405	0	161,96	98,18	394	11	15,5	24,3	0,2
1406	0	162,04	98,18	394	11	17,7	24,3	0,04
1407	0	162,04	98,18	394	11	15,7	24	0,16
1408	0	162,04	98,18	386	19	16,3	23,8	0,14
1409	0	162,24	98,18	394	11	16,2	23,5	0,13
1410	0	162,24	98,18	396	9	15,5	24,5	0,17

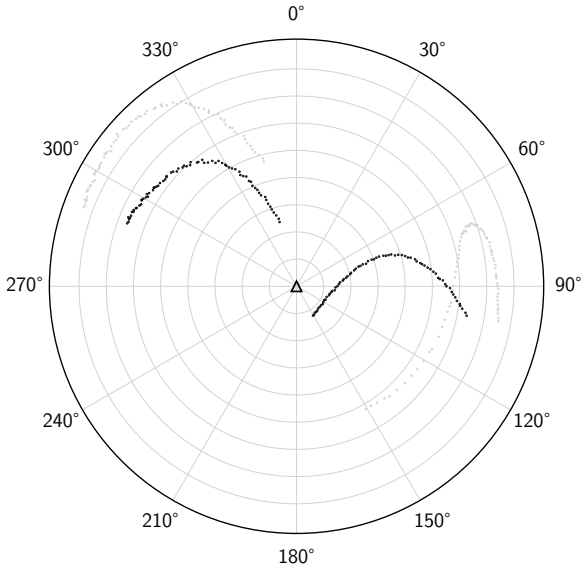
Tabelle B.1: Ergebnisse der Durchläufe des Szenario 1

Szenario 2

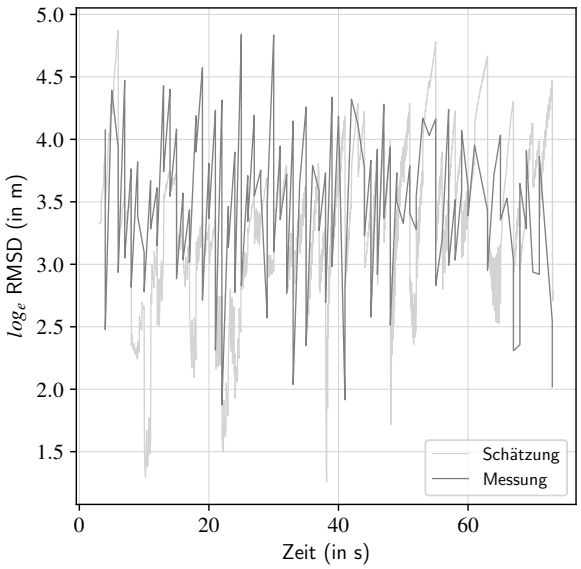
Exemplarische Darstellung



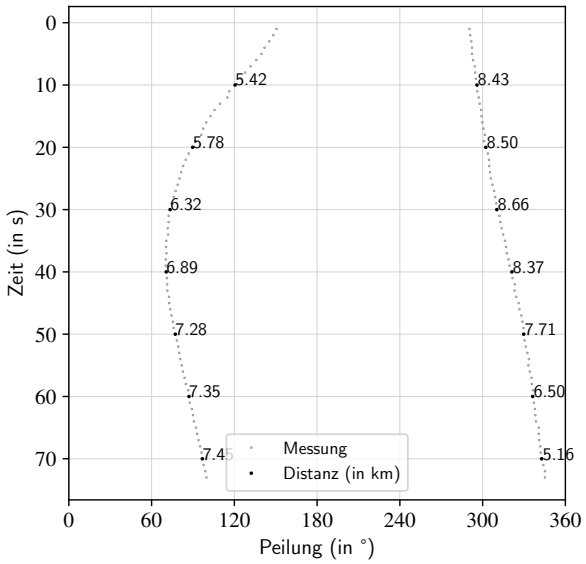
(a) 3D-Plot der Lage



(b) Radar-Plot, kombinierte Ansicht



(c) RMS_D über die Zeit



(d) Entfernungen mit Peilung und Zeit

Abbildung B.3: Exemplarische Darstellung der Szenarien 2.1 bis 2.4; hier Durchlauf 2101

Szenario 2

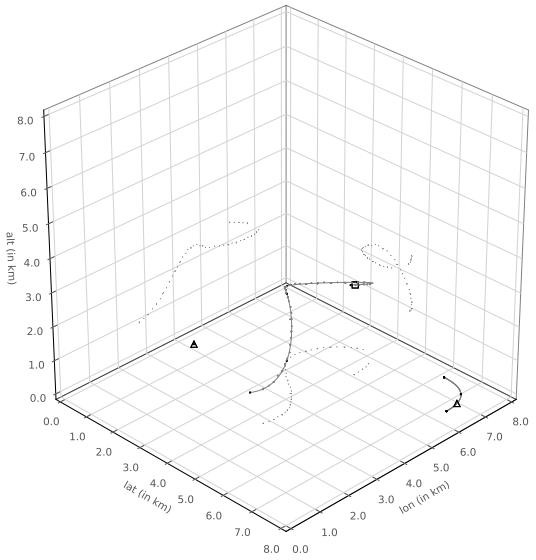
Ergebnisse der Durchläufe

		Abdeckung		Messung		RMS_D (m)		
Lauf	Loss	Zeit (s)	Zeit (%)	akzep.	verw.	Schätzung	Messung	P_{bias}
2101	0	70,12	95,9	124	16	29,9	33,8	-0,14
2102	0	70	95,89	126	14	34,1	33	-0,38
2103	0	69,96	95,89	121	17	24,7	32,9	-0,06
2104	0	70,04	95,89	123	17	30,5	34,5	-0,18
2105	0	69,96	95,89	125	15	37,1	34,8	-0,43
2106	0	69,96	95,89	127	11	24,5	32,4	0
2107	0	70	95,89	127	13	29,7	33,9	-0,18
2108	0	70,04	95,89	127	13	29,1	34	-0,12
2109	0	70,12	95,9	124	16	30,6	36	-0,07
2110	0	70,36	95,91	128	12	29,5	33,4	-0,2
2201	0	68,96	94,52	121	17	48,5	34,8	-1,07
2202	0	69,24	94,54	130	8	34	35,8	-0,44
2203	0	68,84	94,51	124	12	40,2	33,1	-0,93
2204	0	69	94,52	125	13	44,4	36,1	-0,9
2205	0	69,16	94,53	127	11	41,5	32,9	-0,86
2206	0	69,12	94,53	124	14	38,6	32,7	-0,91
2207	0	69,2	94,54	125	13	38	33,8	-0,87
2208	0	68,84	94,51	125	11	40	35,8	-0,69
2209	0	69	94,52	125	13	39,3	34,3	-0,66
2210	0	68,88	94,51	120	16	45	37,4	-0,9
2301	0	69,96	95,89	129	11	33,6	35,4	-0,29
2302	0	69,96	95,89	131	7	33,7	37,4	-0,31
2303	0	70,12	95,9	131	9	50,7	34,4	-1,25
2304	0	70,12	95,9	124	16	45,2	35,2	-0,85
2305	0	69,96	95,89	131	9	32,1	32,5	-0,52
2306	0	69,96	95,89	132	6	32,2	34,3	-0,27
2307	0	70	95,89	132	8	36,4	34,2	-0,55
2308	0	61,96	84,92	112	12	36,6	35,4	-0,41
2309	0	69,96	95,89	128	12	36,1	35,4	-0,37
2310	0	69,96	95,89	133	5	34,3	33	-0,52
2401	0	68,92	94,51	135	1	37,4	34,1	-0,72
2402	0	69	94,52	131	7	34,3	34,1	-0,53
2403	0	69,16	94,53	127	11	40,4	37,3	-0,63
2404	0	69,04	94,52	129	9	41,3	35,1	-0,71
2405	0	68,92	94,51	128	8	34,1	33,4	-0,61
2406	0	69,04	94,52	125	13	44,4	34,4	-1,04
2407	0	69,08	94,53	130	8	38,2	33,9	-0,79
2408	0	68,76	94,5	127	9	40,4	34,9	-0,81
2409	0	69	94,52	133	5	39,4	35,3	-0,7
2410	0	69,04	94,52	129	9	35,3	33,5	-0,61

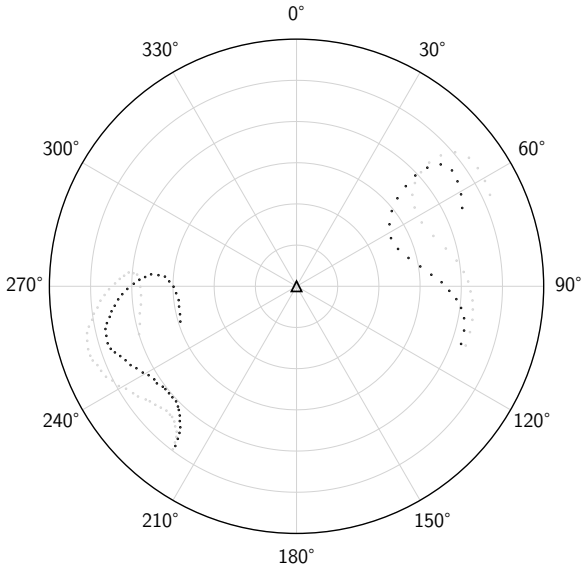
Tabelle B.2: Ergebnisse der Durchläufe des Szenario 2

Szenario 3

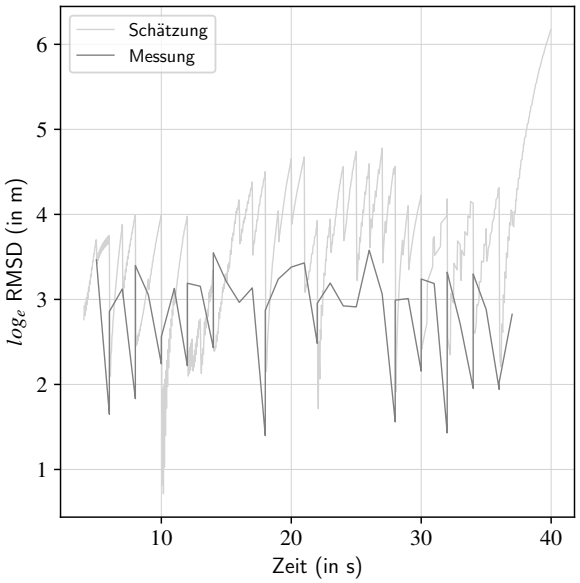
Exemplarische Darstellung



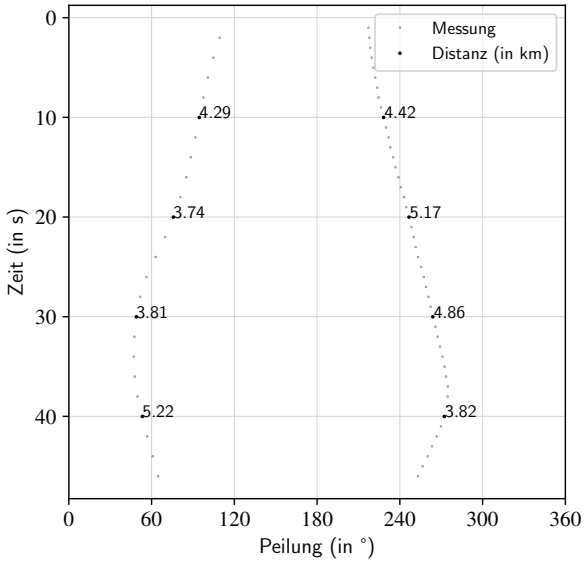
(a) 3D-Plot der Lage



(b) Radar-Plot, kombinierte Ansicht



(c) RMS_D über die Zeit



(d) Entfernungen mit Peilung und Zeit

Abbildung B.4: Exemplarische Darstellung der Szenarien 3.1 bis 3.4; hier Durchlauf 3101

Szenario 3

Ergebnisse der Durchläufe

		Abdeckung		Messung		RMS_D (m)		
Lauf	Loss	Zeit (s)	Zeit (%)	akzep.	verw.	Schätzung	Messung	P_{bias}
3101	1	35,96	78,17	45	9	42,1	17,2	-2,22
3102	2	21,96	47,91	28	5	35,7	13,9	-2,02
3103	2	23,96	52,04	31	5	35,9	14,5	-2,21
3104	2	25,96	56,48	34	5	33,1	16,3	-1,78
3105	2	15,96	34,61	19	5	39,5	14,9	-2,12
3106	1	17,96	38,98	22	5	44,3	13,8	-2,74
3107	2	11,96	26	13	5	46	13,1	-3,33
3108	2	22,96	49,57	30	4	34	15,9	-1,83
3109	2	16,96	36,93	21	5	38,7	15,2	-2,18
3110	2	22,96	50,04	30	5	33,6	13,6	-2,32
3201	0	41,88	91,28	61	0	20,5	14,9	-1,27
3202	0	40,96	89,12	61	1	22,7	14,1	-1,56
3203	0	41,12	89,16	60	2	21,1	15,3	-1,34
3204	0	42,12	91,33	61	2	23	14,8	-1,6
3205	0	41,96	91,3	63	0	19,5	13,3	-1,5
3206	1	35,96	79,91	50	4	21,3	13,8	-1
3207	0	40,96	89,12	62	0	19,3	13	-1,49
3208	0	36,96	80	51	5	21,3	13,6	-1,02
3209	0	41	89,13	61	1	22,1	14,2	-1,55
3210	0	42,24	91,35	63	0	21,1	14,9	-1,3
3301	0	41,56	91,22	61	0	24,5	13,3	-2,52
3302	0	42,92	93,47	63	0	24,1	13,2	-2,4
3303	0	42,6	93,42	63	0	26,6	15	-2,01
3304	0	41,96	91,3	62	1	26,3	13,6	-2,78
3305	0	41,76	91,26	61	0	24,8	14,4	-2,36
3306	0	41,52	91,21	61	0	23,2	14,5	-2,21
3307	0	42,84	93,46	62	1	25,1	13,5	-2,19
3308	0	42,68	93,43	62	1	25,3	15,4	-1,93
3309	0	42,8	93,45	62	1	25,5	14,8	-2,09
3310	0	42,88	93,46	63	0	26,3	14,1	-2,26
3401	0	41,88	91,28	60	1	25,4	14,1	-2,36
3402	0	41,84	91,27	60	1	25,8	14,9	-2,4
3403	0	40,84	89,09	60	0	22,3	13,2	-2,37
3404	0	41,88	91,28	60	1	26,4	13,2	-2,38
3405	0	32,8	71,62	48	0	24,7	13	-2,52
3406	0	40,84	89,09	60	0	27,6	15,5	-2,17
3407	1	23,96	52,36	31	5	32,7	13,8	-2,71
3408	0	41,08	89,15	62	0	24,5	13,9	-2,29
3409	0	10,96	23,83	12	5	60,8	13	-4
3410	0	41	89,13	62	0	24,2	12,3	-2,89

Tabelle B.3: Ergebnisse der Durchläufe des Szenario 3

C. Datenträger

Hauptverzeichnis

Pfad	Beschreibung
doc/	Thesis-Dokument (.tex und .pdf)
eval/	Messreihen sowie Abbildungen der Durchläufe für die Evaluation
misc/	Diagramme, Tabellen, Bilder und Python-Skripte
testbed/	Python-Implementierung des Virtual Testbed

Evaluation

Die Daten der Durchläufe sind in entsprechend benannten Unterverzeichnissen eingeordnet. Jeder Durchlauf einhaltet für den betrachteten Filter eine Menge von Dateien mit definiertem Inhalt:

Datei	Inhalt
fall.pdf	Scatterplot aus Radarsicht: relative Zielbewegung mit Distanz und wahrer Peilung
plot.pdf	3D-Plot für Lageüberblick: absolute Positionen aller Akteure und Messungen
polar.pdf	Polarplot aus Radarsicht: relative Zielbewegung mit Distanzringen
rmsd.pdf	Linienplot für RMS_D: Schätz- und Messfehler über die Zeit
e.csv	Datensatz: geschätzte Positionen für das verfolgte Ziel
m.csv	Datensatz: akzeptierte Messungen in kartesischen Welt-Koordinaten
p.csv	Datensatz: akzeptierte Messungen in polaren Radar-Koordinaten
r.csv	Datensatz: verworfene Messungen in kartesischen Welt-Koordinaten
t.csv	Datensatz: wahre Positionen des verfolgten Ziels in kartesischen Welt-Koordinaten