Nonlinear State Estimation Using Optimal Gaussian Sampling with Applications to Tracking

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Jannik Steinbring

aus Duisburg

Tag der mündlichen Prüfung:	21. Juli 2017
Erster Gutachter:	Prof. Dr.-Ing. Uwe D. Hanebeck
Zweiter Gutachter:	doc. Ing. Ondřej Straka, Ph.D.

Acknowledgment

This thesis is the result of my research at the Intelligent Sensor-Actuator-Systems Lab (ISAS) at the Karlsruhe Institute of Technology. First of all, I would like to thank my advisor Uwe D. Hanebeck for giving me the opportunity to study at the ISAS, and his guidance and support over the last several years, starting with the supervision of my diploma thesis. I also thank Ondřej Straka to be my co-advisor and for our scientific collaborations.

Moreover, all my work would be unthinkable without the help of my ISAS colleagues Florian Faion, Antonio Zea, Christof Chlebek, Florian Pfaff, Igor Gilitschenski, Benjamin Noack, Maxim Dolgov, Jörg Fischer, Gerhard Kurz, Martin Pander, Florian Rosenthal, Achim Langendörfer, and Sascha Faber. Especially our trips to the various conferences were fantastic and a great experience I will never forget. Not to mention the coffee breaks (recently enhanced with many cakes and cookies) that often led to fruitful (scientific) discussions. Special thanks go to Florian Faion and Antonio Zea, who accompanied me since my first days as a student at the lab. Our close friendship is the foundation for the success of my scientific work and the thesis at hand.

I am also very grateful for the support of my family and friends outside the ISAS: the Steinbrings, the Mühlenbergs, the Schmi-Wis, the Denkerts, Christian Mandery, Michael Heck, Jasmin Faion, Fabian Blenski, Wolfgang Woeste, Daniel Neuendorf, Anne Wittkopp, Sandra Rudolph, and the weekly meetings at the Stövchen for relaxing from work and drinking beer with Ruben Baumann, Sebastian Bodenstedt, Matthias Weidemann, and Janine Altschuh.

Karlsruhe, Fall 2017 Jannik Steinbring

Contents

Notation

General Notation

$\mathbb{R}$	Set of real numbers
$\mathbb{R}_+$	Set of positive real numbers
$\mathbb{N}$	Set of natural numbers
$\mathbb{N}_+$	Set of positive natural numbers
N	Natural number, e.g., dimension or number of samples
a	Scalar
$\boldsymbol{a}$	Column vector
$\mathbf{A}$	Matrix
$\|\boldsymbol{a}\|_2$	Euclidean norm of vector $\boldsymbol{a}$
$\|\mathbf{A}\|_\mathrm{F}$	Frobenius norm of matrix $\mathbf{A}$
$\lvert\mathbf{A}\rvert$	Determinant of matrix $\mathbf{A}$
$\mathbf{A}^{-1}$	Inverse of matrix $\mathbf{A}$
$\mathbf{I}_N$	Identity matrix of dimension N
$(\cdot)^\top$	Transpose of a vector/matrix
$\delta(\boldsymbol{a})$	Dirac-δ distribution of vector $\boldsymbol{a}$

Probability Theory

$\hat{s}$	Mean of a random variable s
$\hat{\boldsymbol{s}}$	Mean of a random vector $\boldsymbol{s}$
$\Sigma^{(s)}$	Variance of random variable s
$\mathbf{\Sigma}^{(s)}$	Covariance matrix of random vector $\boldsymbol{s}$
$\mathcal{N}(\boldsymbol{s}\,; \hat{\boldsymbol{s}}, \mathbf{\Sigma}^{(s)})$	Gaussian PDF of random vector $\boldsymbol{s}$ with mean $\hat{\boldsymbol{s}}$ and covariance $\mathbf{\Sigma}^{(s)}$
$\mathcal{U}(s\,; a, b)$	Uniform PDF of random variable s with support $[a, b]$
$\boldsymbol{s} \sim f$	Random vector $\boldsymbol{s}$ is distributed according to distribution f

State Estimation

$\boldsymbol{x}_k$	System state at time step k
$\hat{\boldsymbol{x}}_{k\mid k-1}$	Predicted state mean at time step k
$\hat{\boldsymbol{x}}_{k\mid k}$	Updated state mean at time step k
$\mathbf{P}_{k\mid k-1}$	Predicted state covariance matrix at time step k
$\mathbf{P}_{k\mid k}$	Updated state covariance matrix at time step k
$\boldsymbol{y}_k$	Measurement at time step k
$\hat{\boldsymbol{y}}_k$	Measurement mean at time step k
$\mathbf{Y}_k$	Measurement covariance matrix at time step k
$\mathbf{C}_k$	State–measurement cross-covariance matrix at time step k
$\mathbf{Q}_k$	System noise covariance matrix at time step k
$\mathbf{R}_k$	Measurement noise covariance matrix at time step k
$\tilde{\boldsymbol{y}}_k$	Received measurement at time step k
$\mathcal{Y}_k$	Set of received measurements at time step k
$f_{k\mid k-1}(\boldsymbol{x}_k)$	Predicted state PDF at time step k
$f_{k\mid k}(\boldsymbol{x}_k)$	Updated state PDF at time step k
$f_k(\tilde{\boldsymbol{y}}_k \mid \boldsymbol{x}_k)$	Likelihood function at time step k
$f_k^{(x,y)}(\boldsymbol{x}_k, \boldsymbol{y}_k)$	State–measurement joint PDF at time step k
$f_k^{(w)}(\boldsymbol{w}_k)$	System noise PDF at time step k
$f_k^{(v)}(\boldsymbol{v}_k)$	Measurement noise PDF at time step k

Abbreviations

ASIRPF	Auxiliary sampling importance resampling particle filter
CDF	Central difference filter
CKF	Cubature Kalman filter
CPU	Central processing unit
CvM	Cramér–von Mises
DKF	Distributed Kalman filter
EKF	Extended Kalman filter
EnKF	Ensemble Kalman filter
GHKF	Gauss–Hermite Kalman filter
GPU	Graphics processing unit
GPF	Gaussian particle filter
IoU	Intersection over union
LCD	Localized cumulative distribution
L-BFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno
MSE	Mean square error
NEES	Normalized estimation error squared
OpenCL	Open Computing Language
PDF	Probability density function
PGF	Progressive Gaussian filter
RHM	Random hypersurface model
RMSE	Root mean square error
RPF	Regularized particle filter
RUKF	Randomized unscented Kalman filter
S^2KF	Smart sampling Kalman filter
SIRPF	Sampling importance resampling particle filter
UKF	Unscented Kalman filter
WLS	Weighted least squares

Kurzfassung

Die Normalverteilung spielt eine entscheidende Rolle im Bereich der Zustandsschätzung von zeitdiskreten, stochastischen, nichtlinearen, dynamischen Systemen. Insbesondere das Sampling von Normalverteilungen, also das Approximieren deren kontinuierlichen Verteilungsdichten durch eine Menge von gewichteten Punkten, den Samples, ist zu einer essenziellen Aufgabe geworden. Ein vielversprechendes Sampling-Verfahren basiert auf dem Konzept der sogenannten Localized Cumulative Distribution (LCD) und ermöglicht es, multivariate Standardnormalverteilungen durch eine beliebige Anzahl an optimal platzierten und gleichgewichteten Samples zu approximieren. Derzeit berücksichtigt das LCD-Sampling für die Standardnormalverteilung jedoch nicht deren Punktsymmetrie. Um die Qualität des LCD-Samplings zu verbessern, wird daher im ersten Schritt der Arbeit ein punktsymmetrisches LCD-Sampling für Standardnormalverteilungen entwickelt. Dieses bildet dann die Grundlage für zwei stochastische Filterverfahren und deren Anwendung und Analyse im Bereich der Objektverfolgung. Zudem kommt das neue LCD-Sampling auch im Bereich der verteilten Zustandsschätzung zum Einsatz.

Die erste wichtige Anwendung des neuen Samplingverfahrens sind sample-basierte Kalman-Filter für nichtlineare Systeme. Bei diesen erfolgt die zeitliche Prädiktion der Zustandsschätzung als auch deren Korrektur durch verrauschte Messungen auf Basis von Samples, die zuerst durch das System- bzw. Messmodell transformiert werden, um anschließend mit den transformierten Samples Erwartungswerte und Kovarianzmatrizen zu berechnen. Diese benötigten Samples müssen jedoch die Normalverteilung, welche die aktuelle Schätzung des Systemzustands darstellt, bestmöglich approximieren. Daher wird zunächst, aufbauend auf dem punktsymmetrischen LCD-Sampling und unter Zuhilfenahme der Mahalanobis-Transformation, ein neues sample-basiertes Kalman-Filter, das Smart Sampling Kalman-Filter (S^{2}KF), eingeführt. Die Vorteile des S^{2}KF gegenüber anderen sample-basierten Kalman-Filtern sind zum einen die Kombination aus optimal platzierten Samples und der Möglichkeit, die Genauigkeit der Berechnungen von Erwartungswerten und Kovarianzmatrizen durch eine beliebig einstellbare Anzahl an Samples genauestens vorzugeben. Des Weiteren erlauben die gleichgewichteten Samples, diese Berechnungen zu vereinfachen und so zu beschleunigen. All dies ist insbesondere von Vorteil bei hohen Anforderungen sowohl an die Schätzqualität als auch an die Laufzeit. Zum anderen vermeiden die durchweg positiven Gewichte der Samples des S^{2}KF das Auftreten von indefiniten Kovarianzmatrizen, welche das Durchführen von Prädiktion oder Filterschritt unmöglich machen würden.

Diese Arbeit beschäftigt zudem damit, das S^{2}KF zum verteilten Schätzen eines Systems zu verwenden. Hierbei werden auf mehreren Sensorknoten verrauschte Messungen gewonnen. Um diese, möglicherweise umfangreichen, Messungen nicht über das Netzwerk zu einem Fusionsknoten schicken zu müssen, wird auf jedem Sensorknoten ein eigenes Kalman-Filter ausgeführt. Auf diese Weise müssen nur die lokal gewonnen Erwartungswerte und Kovarianzmatrizen zum Fusionsknoten geschickt werden, welcher diese zu einem globalen Erwartungswert und einer globalen Kovarianzmatrix fusioniert. Eine korrekte Fusion muss allerdings Korrelationen zwischen den lokalen Schätzungen berücksichtigen, die dadurch entstehen, dass alle Kalman-Filter den selben Systemzustand schätzen und somit einem gemeinsamen Prozessrauschen unterliegen. Methoden wie Covariance Intersection bzw. Covariance Inflation erlauben nur das konservative Abschätzen der Korrelationen, und das Distributed Kalman-Filter kann nur korrekte Ergebnisse liefern, wenn die verwendeten Messmodelle sowie die Zeitpunkte und die Anzahl von verarbeiteten Messungen auf allen Sensorknoten bekannt sind. Um diese Probleme zu umgehen, wird ein neues Verfahren vorgestellt, welches auf Samples beruht: Neben der eigentlichen Zustandsschätzung verwaltet und aktualisiert jeder Sensorknoten eine Menge von Samples, aus welchen die Korrelationen im Fusionsknoten exakt rekonstruiert werden können. Aus der Tatsache, dass mit diesem Ansatz die verarbeiteten Messungen und verwendeten (nichtlinearen) Messmodelle nur am jeweiligen Sensorknoten bekannt sein müssen, ist es möglich, lokal sample-basierte Kalman-Filter, wie beispielsweise das S^2KF, einzusetzen, obwohl die linearisierten Modelle im Voraus nicht bekannt sind und somit auch nicht an anderen Sensorknoten zur Verfügung stehen. Darüber hinaus ist der neue Ansatz gut geeignet für große Sensornetzwerke, da das Übertragen der Korrelations-Samples über das Netzwerk sehr viel effizienter sein kann als das Übertragen der unverarbeiteten Messungen, und die Anzahl der Korrelations-Samples unabhängig von der Anzahl der verwendeten Sensorknoten ist.

Die inhärente Linearisierung bei Kalman-Filtern kann jedoch die Qualität der Schätzung stark limitieren. Im Gegensatz dazu vermeidet die Klasse der Partikelfilter dieses Problem, indem sie direkt mit der Likelihood-Funktion arbeitet. Dieser Vorteil geht jedoch einher mit dem Problem der Sampledegeneration. Um dieser entgegenzuwirken müssen Partikelfilter, gerade bei hochdimensionalen Systemzuständen, eine sehr große Anzahl an Samples verwenden, und entsprechend die Likelihood-Funktion sehr häufig auswerten. Daher ist die zweite wichtige Anwendung des punktsymmetrischen LCD-Samplings die Weiterentwicklung eines neuartigen nichtlinearen Filters, dem Progressive Gaussian Filter (PGF). Die dem PGF zugrunde liegende Idee ist es, die Informationen von Messungen nach und nach, sprich progressiv, in die aktuelle Zustandsschätzung einfließen zu lassen. Dazu wird die priore Normalverteilung mittels LCD-Sampling approximiert und die erzeugten Samples nur minimal auf Basis der Likelihood-Funktion umgewichtet, sodass keine Sampledegeneration auftritt. Die umgewichteten Samples werden anschließend als Normalverteilung approximiert indem Erwartungswert und Kovarianzmatrix berechnet werden. Die so neu entstandene Normalverteilung wird abermals mit Samples approximiert, diese umgewichtet und wieder als Normalverteilung approximiert. Das wird solange wiederholt, bis eine posteriore Normalverteilung erreicht wurde. Weil in jedem Schritt eine geringe Menge von Samples ausreicht, wird die Anzahl der Auswertungen der Likelihood-Funktion, im Vergleich zu Partikelfiltern, drastisch reduziert und dennoch Sampledegeneration vermieden. Um die Leistungsfähigkeit des PGF weiter zu erhöhen, werden verschiedene Ansätze verfolgt. Zunächst wird das verwendete LCD-Sampling durch die neue punktsymmetrische Variante ersetzt, um die Qualität der benötigten Momentenberechnungen zu verbessern. Des Weiteren wird eine einfache, jedoch effektive, Heuristik hergeleitet, um eine geeignete Parametrisierung des PGF zu bestimmen. Dies verbessert nicht nur die Laufzeit des PGF, sondern nimmt dem Benutzer auch die Wahl der passenden Werte ab, wodurch das PGF einfacher und intuitiver zu benutzen ist. Darüber hinaus wird eine semianalytische Variante des PGF hergeleitet, welche bei Likelihood-Funktionen anwendbar ist, die nur von einem Teil des Systemzustands abhängig sind. Die Idee ist hierbei, dass der progressive Filterschritt des PGF lediglich dafür verwendet wird, die Schätzung dieses Teilzustands zu aktualisieren, während die Schätzung des anderen Teils in geschlossener Form angepasst werden kann. So wird die Laufzeit verringert und zugleich die Qualität der Schätzung verbessert. Außerdem wird eine hochparallele Implementierung des PGF zur Ausführung auf einer Grafikkarte entwickelt.

Eng verbunden mit der nichtlinearen Zustandsschätzung ist das Gebiet der Objektverfolgung, dem Tracking. Im Rahmen dieser Arbeit wird insbesondere das Tracking von ausgedehnten Objekten behandelt, dessen Formen im Vorhinein unbekannt sind und somit ebenfalls geschätzt werden müssen. Hier besteht ein vielversprechender Ansatz darin, Objekte und deren Form mittels sogenannten Random Hypersurface Models (RHMs) zu modellieren. Daher wird zum einen ein RHM-basiertes Messmodell für Zylinder hergeleitet, um diese mit dem S^2KF schätzen zu können. Dies schließt auch eine verteilte Schätzung mittels der entwickelten sample-basierten Rekonstruktion von Korrelationen mit ein. Für Objekte mit sternkonvexen Formen gibt es bereits den Ansatz der Star-Convex RHMs. Bislang wurde die Modellierung mit Star-Convex RHMs jedoch nur mit Kalman-Filtern umgesetzt. Um das Verfahren nun auch likelihood-basierten Filtern wie dem PGF oder Partikelfiltern zugänglich zu machen, wird zum anderen eine Likelihood-Funktion in geschlossener Form für Star-Convex RHMs hergeleitet und evaluiert. Auch für die Schätzung von Position und Größe einer Kugel wird eine neuartige Likelihood-Funktion erarbeitet, welche die geometrischen Zusammenhänge zwischen Sensor, Kugel und Messung bestmöglich ausnutzt. Anhand dieser Likelihood-Funktion werden außerdem die Vorteile einer hochparallelen Ausführung des PGF auf einer Grafikkarte demonstriert, wenn es darum geht zehntausende Messungen gleichzeitig zu verarbeiten.

Abstract

The normal distribution plays an important role when recursively estimating the hidden state of a discrete-time stochastic nonlinear dynamic system. In particular, an essential task is the sampling of normal distributions, that is, the approximation of their continuous probability density functions with a set of weighted discrete points called samples. A promising sampling technique is based on the concept of the localized cumulative distribution (LCD), which allows the approximation of a multivariate standard normal distribution with an arbitrary number of optimally placed and equally weighted samples. However, the current LCD-based sampling approach does not take into account the point symmetry of the standard normal distribution. Thus, in order to improve sampling quality, in the first part of this thesis a point-symmetric version of the LCD-based sampling for standard normal distributions is developed. This improved sampling technique will build the groundwork for two state estimators and their application and analysis in the field of (extended) object tracking. Further, the point-symmetric LCD-based sampling will also be used in distributed state estimation.

The first important application of this new sampling technique are sample-based Kalman filters for nonlinear systems. Here, both prediction of the state estimate and its correction using noisy measurements are based on samples that are propagated through system model and measurement model, respectively, followed by a computation of mean and covariance matrices of the transformed samples. For that, the required samples have to optimally approximate the normal distribution that represents the current state estimate. Hence, in this thesis we introduce a new sample-based Kalman filter for nonlinear systems: the smart sampling Kalman filter (S^{2}KF). It is based on the point-symmetric version of the LCD-based sampling for standard normal distributions and the Mahalanobis transformation, commonly used in sample-based Kalman filtering to approximate any normal distribution with a set of originally standard normal distributed samples. The advantages of the S^{2}KF, compared to other state-of-the-art sample-based Kalman filters, are on the one hand, the combination of optimal sample placement and the ability to precisely control the accuracy of the computation of means and covariance matrices by using any number of optimally placed samples. Moreover, the equally weighted samples simplify and accelerate these moment computations. All this is particularly useful when high estimation quality under tight runtime prerequisites is required. On the other hand, the throughout positive sample weights of the S^{2}KF avoid indefinite covariance matrices, which would prevent the Kalman filter from conducting a state prediction or a measurement update.

The thesis also focuses on employing the S^{2}KF in distributed state estimation. Here, measurements are obtained from several sensor nodes. In order to avoid sending these measurements over the network to a data fusion center, especially when dealing with a large amount of measurement data, each sensor node runs a local Kalman filter. In doing so, only the locally obtained state means and state covariances have to be sent to the fusion center, which combines these to an overall global state mean and state covariance. However, a correct fusion needs to consider correlations between the local estimates, which must exist as all Kalman filters estimate the same system state, and hence are subject to a common process noise. State-of-the-art methods like covariance intersection or covariance inflation can only give conservative approximations of the global state estimate, and the distributed

Kalman filter can only work correctly if utilized measurement models as well as time and amount of processed measurements are known at each sensor node. To circumvent these problems, we propose a new approach, which is once more sample-based: besides the actual local state estimation, each sensor node processes a set of samples that allows the exact reconstruction of the correlations at the fusion center. The fact that for this approach the processed measurements and utilized (nonlinear) measurement models only have to be known on the respective sensor nodes makes it possible to locally employ sample-based Kalman filters, such as the S^2KF, although the linearized models are unknown in advance, and thus are unavailable at other nodes. Furthermore, the proposed approach is well-suited for large sensor networks, as transferring the correlation samples over the network can be much cheaper than transferring the raw measurements, and the number of correlation samples is independent of the number of employed sensor nodes.

Unfortunately, when applying (sample-based) Kalman filters to nonlinear systems, their inherently performed linearization can negatively effect the quality of the state estimate. As opposed to this, the class of particle filters avoids this problem by directly working with the likelihood function. Unfortunately, this advantage is accompanied by the severe problem of sample degeneracy. In order to mitigate this, particle filters have to employ a large amount of particles, especially when dealing with high-dimensional state spaces. Those many particles, however, increase the computational burden as, e.g., the likelihood function has to be evaluated many times. This leads to the second important application of our new sampling approach: the point-symmetric LCD-based Gaussian sampling is also used to enhance a novel nonlinear estimator, the progressive Gaussian filter (PGF). The key idea of the PGF is to gradually, i.e., progressively, incorporate the information of a measurement into the current state estimate. For that, the prior Gaussian distribution is first approximated with the LCD-based sampling. Second, based on the likelihood function, these samples are only slightly reweighted. Third, the reweighted samples are subsequently approximated as a Gaussian distribution by means of moment matching. The resulting distribution is again approximated with LCD-based samples followed by a reweighting and approximation as Gaussian distribution. This procedure is repeated until a final posterior Gaussian distribution is obtained. As each progression step only requires a small amount of samples, the number of likelihood evaluations is drastically reduced compared to particle filters, while sample degeneracy is avoided due to the slight changes in the sample weights. To further improve the performance of the PGF, different approaches are pursued in this thesis. First, we replace the LCD-based sampling with its point-symmetric version in order to improve the quality of the required moment computations. Second, we provide a simple but yet effective heuristic to determine a proper parametrization of the PGF. This not only improves the filter runtime, but also relieves the user of finding the appropriate parameter value, which makes filter use much easier. Third, a semi-analytic version of the PGF is derived, which is applicable to likelihood functions that do not depend on all system state variables. Here, the idea is to use the progressive filter step of the PGF solely to update the estimate of the dependent system state variables, while the estimate of the independent variables is updated in closed form. This reduces the runtime of the PGF and improves its estimation quality at the same time. Finally, a massively parallel implementation of the PGF for graphics processing units is developed and evaluated.

Closely related to nonlinear state estimation is the field of object tracking. In this thesis, we will dedicate special attention to the tracking of extended objects, whose shapes are unknown in advance and therefore have to be estimated as well. A promising new approach in extended object tracking is to describe the object's shape with the aid of so-called random hypersurface models (RHMs). This description allows for simultaneously estimating the shape and pose, i.e., position and orientation, of an arbitrary and a priori unknown object. Based on the RHM approach, we derive a new measurement model for cylinders to be able to estimate those with a S^2KF. This also includes a distributed estimation using the developed sample-based correlation reconstruction. For star-convex shaped objects there already exists the approach of star-convex RHMs. However, so far target tracking with star-convex

RHMs has only been performed with Kalman filters. In order to make this approach accessible to likelihood-based estimators, such as the PGF or particle filters, we also derive and analyze a closed-form likelihood function for star-convex RHMs. We also propose a novel likelihood function for tracking pose and extent of a sphere that exploits the geometrical relationships between sensor, object, and measurement. Using this likelihood function, we additionally demonstrate the advantages of a massively parallel execution of the PGF on a graphics processing unit when it comes to process tens of thousands of measurements at a time.

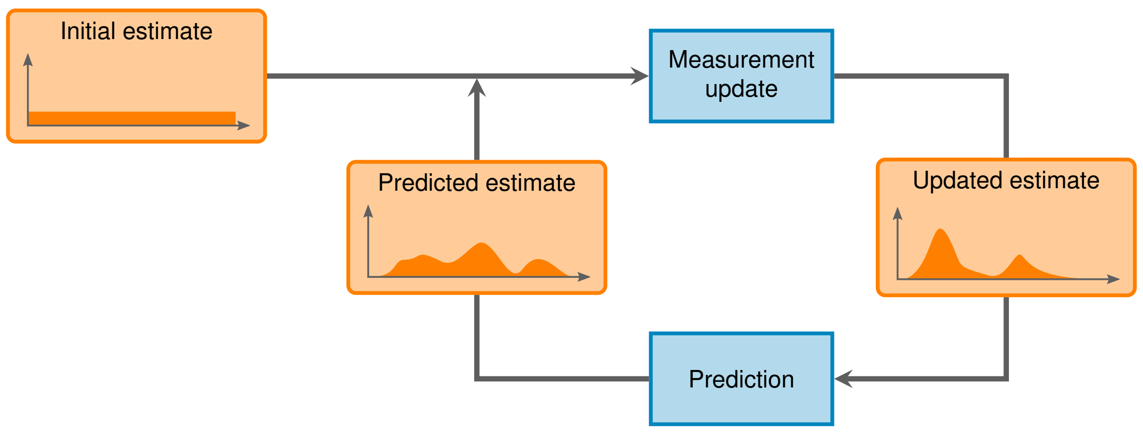
▶ Chapter 1

Introduction

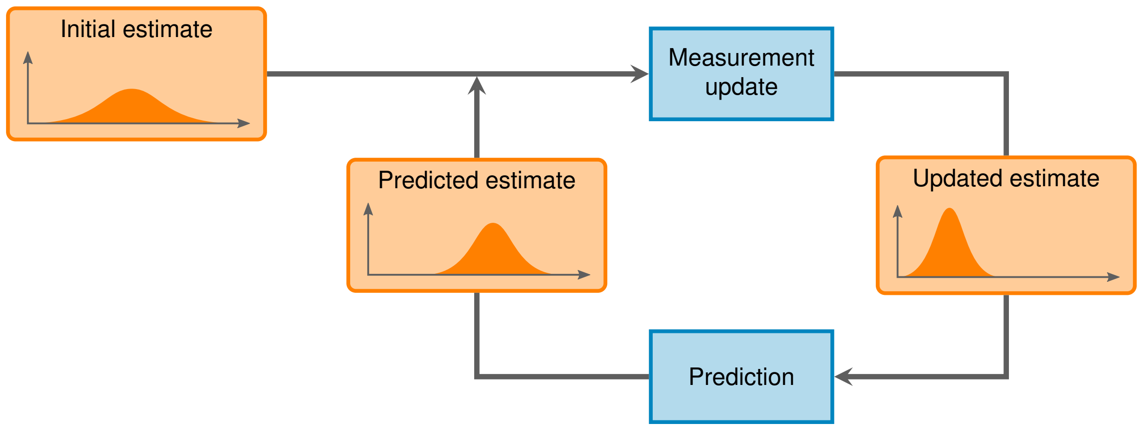
In many engineering tasks, knowledge about the current state of a considered dynamic system is essential. However, the determination of a system state is challenging due to several reasons. Typically, the system state of interest is hidden, that is, it is only indirectly observable through noisy measurements obtained from deployed sensors. Furthermore, the indispensable measurement models, which describe how the system state is related to those measurements, do not perfectly reflect the real world. The same holds for the system models that describe the temporal behavior of the dynamic system. Nevertheless, using those can improve the state estimation or even can be strictly necessary for properly inferring the system state, e.g., in case of sparse measurements. The sensor noise and imperfect models alone raise the need for a stochastic treatment and description of the involved processes, leading to a state estimate in the form of a probability density function (PDF) rather than exact knowledge. Furthermore, the required models can be strongly nonlinear, which calls for sophisticated state estimators. Last but not least, usually continuous-time systems are considered, which have to be discretized in time due to the common digital processing of data. Consequently, system states and their estimates are only considered at discrete time steps. This may also include the sampling of continuous-time measurements such as voltages or temperatures. All these aspects lead to the important class of state estimators for discrete-time stochastic nonlinear dynamic systems, which is the central topic of the present thesis.

Usually, state estimation is performed recursively over time in order to get procedures of constant complexity regarding computational resources and memory. Moreover, the recursion is split into two alternating parts. First, the time update or prediction step propagates a given state estimate from the last time step, i.e., the past, to the current time step by using the system model. Second, the measurement update or filter step combines this predicted estimate with newly available measurement data to an updated state estimate by exploiting the measurement model (or the corresponding likelihood function). This estimate then acts as starting point for the next recursion step, i.e., the next state prediction. While the prediction is based on the Chapman–Kolmogorov equation, the measurement update is conducted by applying Bayes' rule. Hence, the estimation procedures considered in this thesis are called recursive Bayesian estimators/filters. It should be noted that this recursive workflow implies that the system is modeled as a (first-order) Markov process, as future states are predicted solely based on the current state and not the entire state history.

The aimed recursive estimation additionally means that an initial state estimate has to be provided for starting the estimation process. This initial estimate can be crucial for a nonlinear estimation task, as it has a wide influence on the estimators convergence behavior. A proper initial estimate can be obtained, for example, from a priori known information about the considered system, some sort of "initial guess",



(a) General recursive Bayesian estimator.



(b) Special case of a Gaussian estimator.

Figure 1.1: Workflow in recursive Bayesian estimation.

or based on the first available measurements if those are sufficient to derive one. The initial estimate is then refined over time according to the concrete algorithm of the deployed recursive estimator, the used system model and measurement model, and the received measurements. In summary, the workflow of a general recursive Bayesian estimator is illustrated in Figure 1.1(a).

Historically, a prominent recursive Bayesian estimation task has been the trajectory estimation during the Apollo project in the 1960s [1]. Popular present applications are, for example, the optimal control of partially observable system states [2, Chap. 5], the field of robotics [3, 4], driver assistance systems for vehicles [5–7], autonomous driving [8], navigation systems based on a combination of the global positioning system and inertial navigation systems (GPS/INS) [9, 10], or target tracking and its special case of extended object tracking [11–14]. In particular, tracking plays also an important role in this thesis as it serves as evaluation for the derived state estimation techniques.

Although the recursive Bayesian approach builds a theoretically solid framework for state estimation, it unfortunately suffers from the fact that an implementation is only feasible with approximations, as the state estimate, i.e., the PDF, can become arbitrarily complex over time. Processing and describing such complex PDFs in reasonable time and with limited resources is intractable. An exception consists of the special case of an initial Gaussian state estimate and a linear system that suffers from additive Gaussian noise. The exact solution to this estimation problem is the well-known Kalman filter, a linear estimator that is named after its inventor Rudolf E. Kálmán [15]. Consequently, all the above listed applications are forced to rely on practical and reliable approximations, either in the used models or

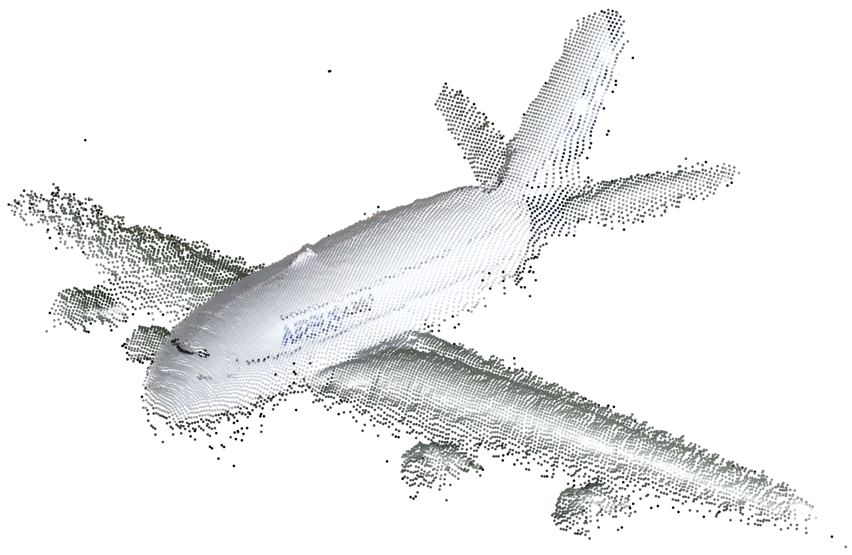


Figure 1.2: Example of a tremendous amount of approximately 16 000 3D noisy point measurements originating from the surface a complex extended object captured by Microsoft's second-generation "Kinect for Windows".

the used estimation algorithms. Hence, developing and improving Bayesian estimators that operate in reasonable time and with sufficient accuracy is still a challenging problem and an important field of research. Specifically, steady innovations in sensor technologies raise the demand for continuous improvements in state estimation. For example, increased sensor resolutions are able to provide more and more measurements per scan. In order to benefit from the newly available data, these need to be efficiently processed by the deployed estimation procedures. This is especially relevant in target tracking, where targets can now be modeled as extended objects rather than as a single point as in Figure 1.2. In particular, comprehensive measurement data offer the possibility to estimate the shape of an a priori unknown object in addition to its pose.

Over the last decades, various approaches for approximating the recursive Bayesian estimator have been proposed. A widely used approximation is the class of Gaussian state estimators. Those are filters that merely maintain a Gaussian state estimate, that is, both state prediction and measurement update result always in a Gaussian PDF, as in Figure 1.1(b). Nevertheless, a Gaussian estimator has several advantages [16]. Its estimate is described with a small and constant amount of parameters, i.e., mean vector and covariance matrix. Gaussian PDFs have useful properties such as closedness under linear transformation and multiplication. Moreover, even though a Gaussian filter means a unimodal estimate, multimodal Gaussian mixture estimators can be obtained in a suboptimal manner by using multiple weighted Gaussian filters. Thus, in this thesis we decided to pursue the approach of Gaussian state estimators.

In the following, we briefly summarize state-of-the-art nonlinear state estimation techniques and their applications to tracking. More thorough discussions of the various topics are given throughout this thesis in the beginning of the relevant chapters.

1.1 State-of-the-Art Nonlinear State Estimation

State-of-the-art recursive Bayesian estimators mainly differ in the way they approximate the measurement update and can be roughly divided into two classes. On the one hand, we have Kalman filters applied to nonlinear systems. This is achieved by approximating the actual nonlinear relationship between state and measurement as a linear one, which in turn allows for solving the Bayesian update analytically by simply computing first-order and second-order moments of nonlinear transformed Gaussian random vectors. Due to the Kalman filter's Gaussian estimate, those filters belong to the class of Gaussian estimators. On the other hand, we have explicitly formulated nonlinear estimators. Those avoid the model linearization of the Kalman filter by directly approximating the product of likelihood function and prior estimate, where the prior is mostly a Gaussian, a Gaussian mixture, or a Dirac mixture, i.e., a set of weighted samples/particles.

The class of Kalman filters applied to nonlinear systems can be further divided into two subclasses, depending on how the required first-order and second-order moments, i.e., the involved multidimensional integrals, are computed. First, we have those filters that approximate the nonlinear models in such a way that the integrals can be solved in closed form. This includes the popular extended Kalman filter (EKF) [17, Sec. 13.2], [18] and its second-order variant [19, 20] that rely on Taylor series expansions. Other filters use polynomial approximations such as the first-order and second-order divided difference filter (DDF) [21], the first-order and second-order central difference filter (CDF) [22], or the Chebyshev polynomial Kalman filter (CPKF) [23]. Second, we have Kalman filters that directly approximate the computation of the moment integrals. Those filters replace the involved multivariate Gaussian PDFs, i.e., prior state estimate and noise, with a properly chosen set of weighted samples. As a consequence, solving the integrals boils down to a computation of sample means and sample covariance matrices. Hence, those filters are also called sample-based Kalman filters. The most prominent one is the unscented Kalman filter (UKF) [16, 24, 25] and various approaches concerning its parametrization [26–30]. Others are the Gauss–Hermite Kalman filter (GHKF) [22], the cubature Kalman filter (CKF) [31] and its high-degree variant [32], or the randomized unscented Kalman filter (RUKF) [33–37].

Due to the popularity of Kalman filters, many different extensions have been proposed, which can be used in combination with any Kalman filter. For example, this includes different iterative measurement updates in order to find better linearizations of the measurement models [17, Sec. 13.3], [38–41] or square root Kalman filters to improve the precision of the state covariance matrix [11, Sec. 7.4], [17, Sec. 6.3], [42]. Also Gaussian mixture Kalman filters are widely used to obtain multimodal estimates. These can be configured in various aspects such as updating the component weights or splitting and merging of components [22, 43–45]. Moreover, several approaches exist to incorporate different type of constraints into Kalman filtering, e.g., [17, Sec. 7.5], [46–49]. In order to improve the Kalman filter's estimation quality, special structures in the measurement models can be exploited. For example, if a measurement model uses only a subset of the state variables, it is advised to perform a so-called state decomposition [4, App. E], [50]. Alternatively, semi-analytic moment calculations might also be possible in certain cases [51–53].

The class of nonlinear state estimators is dominated by the family of particle filters. In contrast to Kalman filters, their state estimates are Dirac mixtures rather than Gaussian distributions, which inherently allows for multimodal estimates. Particle filtering is based on the two concepts of importance sampling and (adaptive) random resampling. Further, a measurement update basically means a reweighting of the prior Dirac mixture. Proposed particle filters like the sampling importance resampling particle filter (SIRPF), the auxiliary sampling importance resampling particle filter (ASIRPF), the regularized particle filter (RPF), or the local linearization particle filter [12, Sec. 3.5], [54, 55] primarily differ in the choice of the importance density, and how and when resampling is done. Like for Kalman filters, special structures in the measurement model/likelihood function can be exploited to enhance estimation quality, which is known as Rao–Blackwellization [56] or marginalized particle filters [57]. A nonlinear Gaussian estimator is the Gaussian particle filter (GPF) [58, 59]. For each measurement update, its Gaussian estimate is randomly sampled and reweighted according to the likelihood function. Another well-known particle filter is the ensemble Kalman filter (EnKF) [60–64].

In contrast to the above filters, its state estimate is updated by moving the samples in state space instead of reweighting them. This comes at the cost of a filter step that closely resembles the Kalman filter update. In order to circumvent the major problem of sample degeneracy in particle filtering, a progressive measurement update for particle filters is proposed in [65]. The key idea is to split the given likelihood function into sublikelihoods and then do several consecutive updates, where each update consists of sample reweighting and resampling. The progression approach is reformulated in [66] to use deterministic Dirac mixtures for the state estimate instead of random samples. This idea finally leads to the so-called progressive Gaussian filter 42 (PGF 42), which relies on a Gaussian estimate rather than a Dirac mixture [67].

Another important topic in (nonlinear) state estimation is the field of distributed state estimation. where measurements are not directly accessible for processing. Instead, they are obtained by several (physically) distributed sensor nodes, while the actual state estimation has to be performed at a distinct fusion center. If full-rate communication is feasible, the extended information filter [68, Sec. 3.4.2] can be used to exactly obtain the same estimate of a centrally working EKF with the advantage that the fusion center does not need to know concrete information about the required measurement models. Moreover, if models are linear and comprehensive information about measurement processing is known to all sensor nodes, the distributed Kalman filter (DKF) can avoid a full-rate communication and still compute the optimal central estimate [69–72]. If all those requirements cannot be met, a common suboptimal approach is the weighted least squares fusion of locally obtained state estimates. The main issue is here that the common system noise introduces correlations between the local estimates that have to be taken into account during the fusion process [73]. Popular approximations to this are covariance inflation, i.e., the federated Kalman filter [74], or covariance intersection [75]. Recently, also a weighted least squares fusion based on the local processing of random samples is proposed [76].

1.2 State Estimation and its Applications to Tracking

Closely related to the state estimation of dynamic systems is the field of target tracking [11, 12]. Here, the considered dynamic system is the target to be tracked and its state usually encompasses pose, i.e., position and orientation, and certain motion parameters such as velocities or accelerations for a proper prediction. In fact, target tracking is frequently used to evaluate new developments in state estimation, e.g., [21, 34, 41, 57, 77] just to name a few. Besides rather simple linear position measurements, often nonlinear measurements are considered such as range measurements, bearing measurements, or measurements in polar coordinates. Note that the combination of target tracking and distributed state estimation is also referred to as track-to-track fusion [78].

Recently, the more demanding case of extended object tracking, where the target is no longer modeled as a single point, has become of special interest. Consequently, measurement models describing the relationship between state and measurement have become more complex. Furthermore, if the target's shape is unknown it has to be estimated in conjunction with the target's pose. As a result, the system state is augmented with various parameters describing the current shape of the object. Popular approaches are spatial distribution models (SDMs) [79, 80], [174] including the idea of random matrices [81, 82], random hypersurface models (RHMs) [83–89], or partial information models (PIMs) [90, 91].

Other relevant topics in target tracking are, for example, interacting multiple models (IMMs) [92, 93], e.g., to deal with unknown and time-varying system noise statistics [94], or multi target tracking in the form of multi hypotheses tracking (MHT) [95, Sec. 6.7], the joint probabilistic data association filter (JPDAF) [96], probability hypothesis density (PHD) filters [97–99], or multi-Bernoulli filters [100–102]. Nevertheless, these estimation techniques are out of the scope of this thesis.

1.3 Contributions and Outline

The two main contributions of this thesis are new developments in nonlinear state estimation and their application to the field of (extended) object tracking. While the first contribution includes the derivation of two state estimators and an optimal fusion approach for distributed state estimation, the second contribution comprises the derivation of new measurement models and closed-form likelihood functions. In the following, we shortly present each contribution of this thesis.

Optimal Point-Symmetric Gaussian Sampling In Chapter 2, we deal with the sampling of multivariate Gaussian distributions, i.e., the approximation of Gaussian densities with a carefully chosen set of weighted point masses. A promising sampling technique is based on the concept of the localized cumulative distribution (LCD), which allows the approximation of a multivariate standard normal distribution with an arbitrary number of optimally placed and equally weighted samples. However, the LCD-based sampling does not take into account the point symmetry of the standard normal distribution. Thus, in order to improve sampling quality, we propose a point-symmetric version of the LCD-based sampling for standard normal distributions. Moreover, we enhance the numerical stability of the LCD approach to be able to approximate large random vectors, which frequently arise in extended object tracking. We show that the proposed Gaussian sampling scheme can outperform state-of-the-art sampling approaches when computing higher-order moments of standard normal distributions and moments of nonlinear transformed Gaussian distributed random vectors.

The Smart Sampling Kalman Filter (S^{2}KF) As a first application of the proposed point-symmetric Gaussian sampling scheme, we introduce a new sample-based Kalman filter, the smart sampling Kalman filter (S^{2}KF), in Chapter 3. The advantage of the S^{2}KF over state-of-the-art Kalman filters is its arbitrary number of optimally placed and equally weighted samples, which allows for a fine-grained control over estimation quality and execution time. Additionally, the equal sample weights reduce the number of arithmetic operations required for a moment computation and also avoid numerical issues when computing covariance matrices. As a demanding evaluation, we estimate the pose and unknown shape of a cylinder in 3D based on hundreds of noisy point measurements. For that purpose, we first derive a new measurement model that combines the approaches of the random hypersurface model, the signed Euclidean distance, and the sampling of uniform distributions by transforming samples with its own cumulative distribution. Simulations demonstrate that the S^{2}KF can beat state-of-the-art Kalman filters regarding both estimation quality and numerical stability.

Optimal Sample-Based Fusion for Distributed State Estimation In Chapter 4, our goal is to deploy the S^{2}KF for challenging distributed nonlinear state estimation scenarios. Due to the shortcomings of established distributed state estimation techniques, we develop a novel weighted least squares fusion where correlations between local estimates can be exactly reconstructed at the fusion center. Basically, the approach relies on a local processing of samples that encode the correlations and are sent to the fusion center in addition to the actual locally obtained state estimate. Advantages are that sensor nodes do not need any information about the measurement processing from other nodes and that it is well-suited for large sensor networks due to its scalability. Target tracking evaluations show that the proposed fusion technique can outperform the popular covariance intersection approach for both linear and nonlinear models. In particular, we reconsider the previously performed pose and shape estimation of a cylinder, but now in a distributed setup. Simulations show that covariance intersection is not even capable of estimating the cylinder's pose and shape, while our sample-based fusion approach performs quite well.

The Progressive Gaussian Filter (PGF) In Chapter 5, we take up the promising progressive approach of the PGF 42 in order to circumvent the severe problem of sample degeneracy in nonlinear filtering and get a more efficient progressive Gaussian filter called simply PGF. Among other improvements, the PGF works directly with given likelihood functions, it uses the proposed point-symmetric LCD-based Gaussian sampling instead of its original asymmetric version, and it relies on an effective heuristic for an automatic parametrization. We also derive a semi-analytic filter step to further improve estimation quality and reduce runtime, and propose a GPU-accelerated implementation to efficiently deal with tens of thousands of measurements in a single filter step. Several target tracking evaluations reveal that the PGF can beat state-of-the-art nonlinear and linear estimators including the RPF, GPF, PGF 42, and S^2KF. In addition, its semi-analytic filter step yielded noticeable improvements. For extended object tracking, we develop a closed-form likelihood function for star-convex random hypersurface models that is used to track an airplane in 2D, and we develop a novel likelihood function for tracking the pose and extent of a sphere that demonstrates the superiority of a GPU-accelerated PGF over a multithreaded CPU implementation.

Conclusions The thesis is concluded in Chapter 6. We summarize the various proposed approaches for nonlinear state estimation and extended object tracking and additionally discuss interesting future work and open challenges.

▶ Chapter 2

Optimal Point-Symmetric Gaussian Sampling

In recursive Bayesian state estimation, the inferred estimate is a probability distribution over the state space rather than only a point estimate due to the applied Bayes' rule. The advantage in this is that modelling errors and imperfect measurements prevent an exact knowledge of the system state anyway, i.e., the gained knowledge is only certain to some degree. A probability distribution inherently reflects this uncertainty and allows to quantify it, e.g., by computing variances and covariances. Moreover, there might be hidden state variables that can solely be estimated by means of other state variables due to known correlations. For example, in target tracking the target's position is often directly inferred from measurements, while its velocity is only updated due to correlations between position and velocity. In addition, the probability distribution can be used to effectively discard measurement outliers, or to initiate procedures if the estimate becomes too uncertain, and thus cannot be trusted anymore, e.g., perform an emergency stop of a system to prevent mechanical or personal damage.

The Gaussian distribution plays an important role in the field of Bayesian state estimation. Due to the central limit theorem, noise distributions are typically assumed to be Gaussian [103, Chap. 2]. Also the state estimate itself is frequently a Gaussian distribution [58, 104]. Although this implies that the resulting filter maintains only a unimodal state estimate, a multimodal filter can be obtained in a suboptimal manner by using multiple weighted filters, i.e., a Gaussian mixture estimator [22, 43].

When dealing with Gaussian distributions in the context of recursive state estimation, a key aspect is the nonlinear transformation of a Gaussian random vector. That is, let $\boldsymbol{p} \in \mathbb{R}^N$ be a Gaussian distributed random vector. Its probability density function (PDF) with mean vector $\hat{\boldsymbol{p}} \in \mathbb{R}^N$ and covariance matrix $\boldsymbol{\Sigma}^{(p)} \in \mathbb{R}^{N \times N}$ is given by [105, Chap. 5]

$$\mathcal{N}(\boldsymbol{p}\,; \hat{\boldsymbol{p}}, \boldsymbol{\Sigma}^{(p)}) = \frac{1}{(2\pi)^{\frac{N}{2}} \sqrt{|\boldsymbol{\Sigma}^{(p)}|}} \exp\left(-\frac{1}{2} (\boldsymbol{p} - \hat{\boldsymbol{p}})^\top \left(\boldsymbol{\Sigma}^{(p)}\right)^{-1} (\boldsymbol{p} - \hat{\boldsymbol{p}}) \right) \ ,$$

where $|\cdot|$ denotes the determinant. Then, for a nonlinear function $\boldsymbol{g} : \mathbb{R}^N \to \mathbb{R}^T$, a transformation of the random vector $\boldsymbol{p}$ is conducted according to

$$\boldsymbol{t} = \boldsymbol{g}(\boldsymbol{p}) \ ,$$

which results in the, generally not Gaussian, random vector $\boldsymbol{t} \in \mathbb{R}^T$. Now, our goal is to compute the mean $\hat{\boldsymbol{t}}$ of $\boldsymbol{t}$ according to [106, Sec. 6.4]

$$\hat{\boldsymbol{t}} = \int_{\mathbb{R}^N} \boldsymbol{g}(\boldsymbol{p}) \mathcal{N}(\boldsymbol{p}\,; \hat{\boldsymbol{p}}, \boldsymbol{\Sigma}^{(p)}) \,\mathrm{d}\boldsymbol{p} \ . \tag{2.1}$$

Note that we do *not* need to know the actual distribution of the random vector $\boldsymbol{t}$ in order to compute (2.1), only the known Gaussian distribution of $\boldsymbol{p}$ is required. In the context of state estimation, $\boldsymbol{p}$ can be, for example, the state space or the joint space of state and noise, and $\boldsymbol{g}$ a measurement equation mapping state and noise to the measurement space. Unfortunately, closed-form solutions for the multi-dimensional integral (2.1) are only possible for rare special cases such as linear, polynomial, or trigonometric functions [52]. Hence, in most cases only approximative solutions can be obtained. However, in recursive state estimation, solving such integrals typically has to be performed *online*. Consequently, fast but still accurate integration methods are needed. The major challenge in this is that the space of $\boldsymbol{p}$ increases exponentially in its dimension N.

Fortunately, we do not have to consider each region in the space with the same priority. More precisely, the Gaussian distribution of $\boldsymbol{p}$ inherently determines the important regions of the space, i.e., regions with larger probability mass are more important than regions with less probability mass. In order to exploit this and get an adequate approximation of (2.1), we want to replace the Gaussian density of $\boldsymbol{p}$ with an appropriate sample-based representation comprising $M \in \mathbb{N}_+$ weighted samples defined as

$$\mathcal{N}(\boldsymbol{p}\,; \hat{\boldsymbol{p}}, \boldsymbol{\Sigma}^{(p)}) \approx \sum_{i=1}^{M} \omega^{(i)} \delta(\boldsymbol{p} - \boldsymbol{p}^{(i)}) \ , \tag{2.2}$$

where δ denotes the Dirac-δ distribution, $\boldsymbol{p}^{(i)}$ the position of the i^{th} sample, and $\omega^{(i)}$ its corresponding weight. The sample weights have to satisfy

$$\sum_{i=1}^{M} \omega^{(i)} = 1 \ ,$$

and thus (2.2) integrates to one. The sample-based representation (2.2) is occasionally called probability mass function [11, Sec. 1.4]. Analogously to Gaussian mixture distributions, we will denote (2.2) as *Dirac mixture* throughout this thesis. By replacing $\mathcal{N}(\boldsymbol{p}\,; \hat{\boldsymbol{p}}, \boldsymbol{\Sigma}^{(p)})$ in (2.1) with a proper Dirac mixture approximation (2.2) and using the sifting property of the Dirac-δ distribution, we get the desired approximation

$$\hat{\boldsymbol{t}} \approx \int_{\mathbb{R}^N} \boldsymbol{g}(\boldsymbol{p}) \sum_{i=1}^{M} \omega^{(i)} \delta(\boldsymbol{p} - \boldsymbol{p}^{(i)})\, \mathrm{d}\boldsymbol{p} = \sum_{i=1}^{M} \omega^{(i)} \boldsymbol{g}(\boldsymbol{p}^{(i)}) \ , \tag{2.3}$$

which in fact is a weighted sum of evaluations of the nonlinear function $\boldsymbol{g}$. In doing so, accurately solving the integral (2.1) in decent time boils down to find a suitable Dirac mixture approximation of a multivariate Gaussian distribution. On the one hand, suitable means efficient in terms of the number of samples M in order to reduce the number of (maybe costly) evaluations of $\boldsymbol{g}$. To achieve this, the samples $\boldsymbol{p}^{(i)}$ should be placed only in regions where the probability mass of $\mathcal{N}(\boldsymbol{p}\,; \hat{\boldsymbol{p}}, \boldsymbol{\Sigma}^{(p)})$ is sufficiently large, and thus contribute to the sum in (2.3). Furthermore, all important regions should be covered by the samples. On the other hand, the generated Dirac mixture should preserve certain properties of the Gaussian distribution, such as mean, covariance, or non-skewness.

In this chapter, we present an optimal (in terms of a selected optimization criterion) point-symmetric sampling technique for multivariate Gaussian distributions that matches all these requirements. It will be extensively used later in this thesis as it builds the groundwork for two state estimators, namely the smart sampling Kalman filter and the progressive Gaussian filter.

This chapter is based on the publication [175].

2.1 Related Work

In literature, there exist many approaches to compute Dirac mixtures (2.2) that approximate a Gaussian distribution. However, most of them do *not* directly approximate the Gaussian PDF $\mathcal{N}(\boldsymbol{p}\,;\hat{\boldsymbol{p}},\boldsymbol{\Sigma}^{(p)})$. Instead, they merely approximate the PDF $\mathcal{N}(\boldsymbol{s}\,;\mathbf{0},\mathbf{I}_N)$ of a standard normal distributed random vector $\boldsymbol{s}\in\mathbb{R}^N$ with a Dirac mixture

$$\mathcal{N}(\boldsymbol{s}\,;\mathbf{0},\mathbf{I}_N)\approx\sum_{i=1}^{M}\omega^{(i)}\delta(\boldsymbol{s}-\boldsymbol{s}^{(i)}) \ . \tag{2.4}$$

The reason for this is that the Gaussian distribution is closed under affine transformations. In particular, this leads to the Mahalanobis transformation discussed next.

2.1.1 Mahalanobis Transformation

Given a mean vector $\hat{\boldsymbol{p}}$ and a covariance matrix $\boldsymbol{\Sigma}^{(p)}$, and let a random vector $\boldsymbol{s}$ be standard normal distributed, i.e., $\boldsymbol{s}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_N)$. With the aid of the Mahalanobis transformation, i.e., rewriting [107, Theorem 4.5], we can transform $\boldsymbol{s}$ to a random vector $\boldsymbol{p}\sim\mathcal{N}(\hat{\boldsymbol{p}},\boldsymbol{\Sigma}^{(p)})$ according to

$$\boldsymbol{p}=\mathbf{D}\boldsymbol{s}+\hat{\boldsymbol{p}} \ , \tag{2.5}$$

where the matrix $\mathbf{D}\in\mathbb{R}^{N\times N}$ has to satisfy $\mathbf{D}\mathbf{D}^\top=\boldsymbol{\Sigma}^{(p)}$. Using (2.5) and again (2.1), we get

$$\hat{\boldsymbol{t}}=\int_{\mathbb{R}^N}\boldsymbol{g}(\mathbf{D}\boldsymbol{s}+\hat{\boldsymbol{p}})\mathcal{N}(\boldsymbol{s}\,;\mathbf{0},\mathbf{I}_N)\,\mathrm{d}\boldsymbol{s} \ . \tag{2.6}$$

By replacing $\mathcal{N}(\boldsymbol{s}\,;\mathbf{0},\mathbf{I}_N)$ with the Dirac mixture (2.4), we can rewrite (2.3) according to

$$\begin{aligned}\hat{\boldsymbol{t}}&\approx\int_{\mathbb{R}^N}\boldsymbol{g}(\mathbf{D}\boldsymbol{s}+\hat{\boldsymbol{p}})\sum_{i=1}^{M}\omega^{(i)}\delta(\boldsymbol{s}-\boldsymbol{s}^{(i)})\,\mathrm{d}\boldsymbol{s}\\&=\sum_{i=1}^{M}\omega^{(i)}\boldsymbol{g}(\mathbf{D}\boldsymbol{s}^{(i)}+\hat{\boldsymbol{p}})\\&=\int_{\mathbb{R}^N}\boldsymbol{g}(\boldsymbol{p})\sum_{i=1}^{M}\omega^{(i)}\delta(\boldsymbol{p}-\boldsymbol{p}^{(i)})\,\mathrm{d}\boldsymbol{p} \ .\end{aligned}$$

In other words, the Dirac mixture with sample positions

$$\boldsymbol{p}^{(i)}=\mathbf{D}\boldsymbol{s}^{(i)}+\hat{\boldsymbol{p}} \ , \quad \forall i\in\{1,\dots,M\} \ , \tag{2.7}$$

and corresponding weights $\omega^{(i)}$ approximates the PDF $\mathcal{N}(\boldsymbol{p}\,;\hat{\boldsymbol{p}},\boldsymbol{\Sigma}^{(p)})$. Hence, we only have to provide a Dirac mixture that approximates the PDF of a standard normal distribution in order to approximate an arbitrary Gaussian PDF.

The matrix $\mathbf{D}$ for the sample transformation (2.7) is not unique and can be obtained in different ways, depending on the chosen matrix decomposition method. As $\boldsymbol{\Sigma}^{(p)}$ is a covariance matrix, it is a real symmetric positive definite matrix that only has positive real eigenvalues. Hence, various decomposition methods can be applied. For example, the Cholesky decomposition results in $\boldsymbol{\Sigma}^{(p)}=\mathbf{L}\mathbf{L}^\top$, where $\mathbf{L}$ is a lower triangular matrix, which directly gives $\mathbf{D}=\mathbf{L}$. The eigendecomposition computes $\boldsymbol{\Sigma}^{(p)}=\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$, where $\mathbf{U}$ is an orthogonal matrix that contains the eigenvectors of $\boldsymbol{\Sigma}^{(p)}$ and $\boldsymbol{\Lambda}$ a diagonal matrix that is composed of the corresponding eigenvalues. From this, it simply follows that $\mathbf{D}=\mathbf{U}\sqrt{\boldsymbol{\Lambda}}$. The advantage of the Cholesky decomposition over the eigendecomposition is that it is cheaper to compute and unique. Additionally, for dimensions $N > 4$, the eigendecomposition generally cannot be computed in closed form and an approximate numerical method has to be used.

Which decomposition should be used depends on various factors such as the concrete nonlinear function $\boldsymbol{g}$, the context, e.g., sample-based Kalman filtering, and possible constraints on quality and/or execution time, e.g., see [10, 108]. Throughout this thesis, we will use the Cholesky decomposition to compute the matrix $\mathbf{D}$.

2.1.2 State-of-the-Art Sampling Techniques

Many efficient Gaussian sampling techniques are closely related to sample-based Kalman filters for nonlinear systems. Although Chapter 3 is dedicated to these filters, their respective sampling techniques can be used for any purpose, and not only in the context of state estimation or specifically in Kalman filtering. Nevertheless, most of the following sampling techniques are named after the Kalman filter for which they were developed.

Monte Carlo Integration

In order to obtain the mean $\hat{\boldsymbol{t}}$, the integration in (2.1) can be computed based on random sampling, called Monte Carlo integration, e.g., [12, Sec. 3.1]. Here, the Dirac mixture (2.4) is obtained by (i) randomly drawing standard normal distributed samples $\boldsymbol{s}^{(i)}$ using a (pseudo) random number generator, and (ii) assigning each sample the same weight

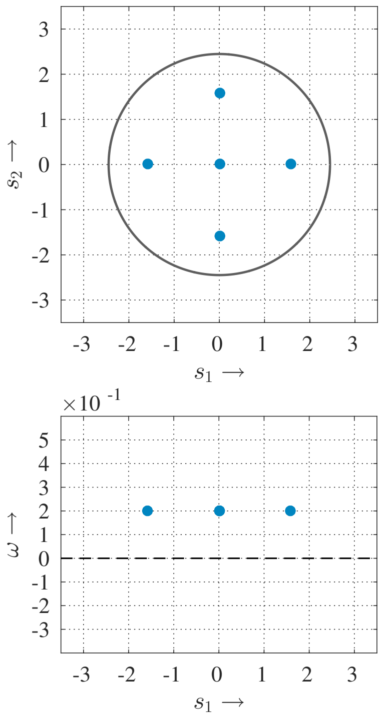
$$\omega^{(i)} = \frac{1}{M} \ , \quad \forall i \in \{1, \dots, M\} \ ,$$

that is, all samples are equally weighted. Due to the law of large numbers, Monte Carlo integration will asymptotically converge to the true mean $\hat{\boldsymbol{t}}$ for $M \to \infty$. Unfortunately, a meaningful approximation of $\hat{\boldsymbol{t}}$ will consequently require many samples, especially for large dimensions N. Furthermore, the randomly drawn samples will not guarantee that the resulting Dirac mixture has a sample covariance matrix equal to $\mathbf{I}_N$. However, especially in Kalman filtering this is a very important property as these filters only rely on the first-order and second-order moments of the state and measurement distributions.

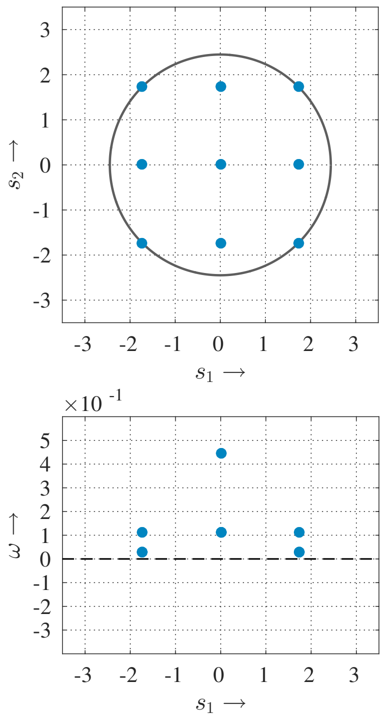
Unscented Kalman Filter

The most prominent sampling technique for the standard normal distribution is the one used by the unscented Kalman filter (UKF) [16, 24]. It employs systematically chosen $M = 2N + 1$ axis-aligned samples, see Figure 2.1(a). The sample spread (and with that the sample weights) can be controlled by a scaling factor. For certain scaling factors, the sample located at the state space origin can have a negative weight. Advantages of the UKF sampling are its simple creation and the relatively small computational effort. However, drawbacks are that it is not possible to increase the number of samples in order to increase the quality of the approximation of $\hat{\boldsymbol{t}}$ and that the state space coverage suffers from the fact that the samples are placed solely on the principal axes. A consequence is that the even moments of a Gaussian greater than the second order cannot precisely be matched [16].

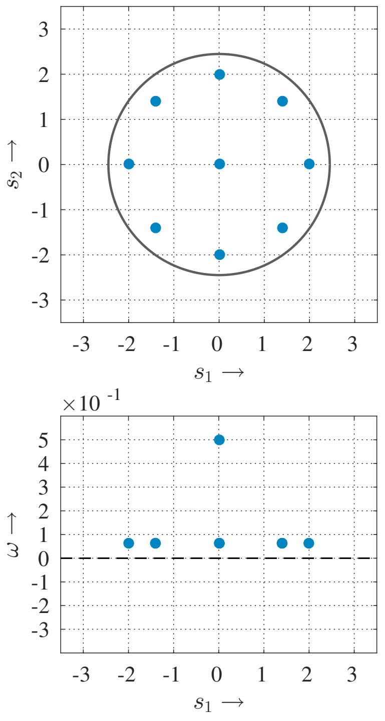
In addition, a simplex version of the UKF sampling exists [25]. Here, only $M = N + 2$ samples are employed. In doing so, the number of samples is nearly reduced by half. On the one hand, this decreases the computational effort. On the other hand, if the sampling is used to compute covariance matrices, e.g., during a Kalman filter measurement update, these are more prone to become indefinite due to the reduced amount of used samples. The simplex sampling is not further considered here, but will be used in Chapter 4 in the context of distributed state estimation.



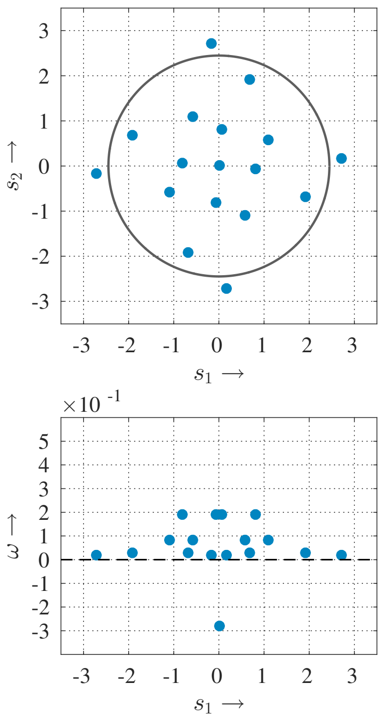
(a) UKF sampling with equally weighted samples.



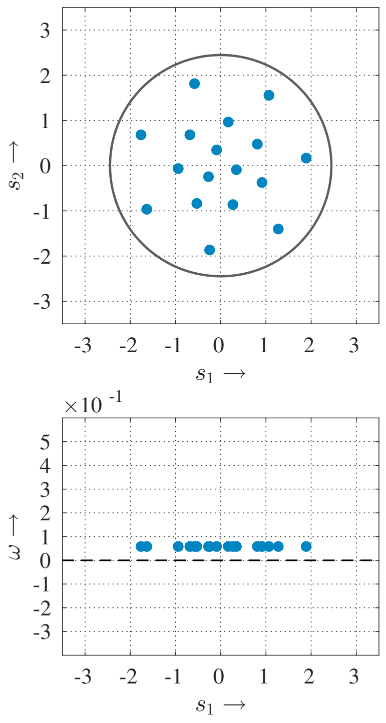
(b) GHKF sampling with 3 quadrature points.



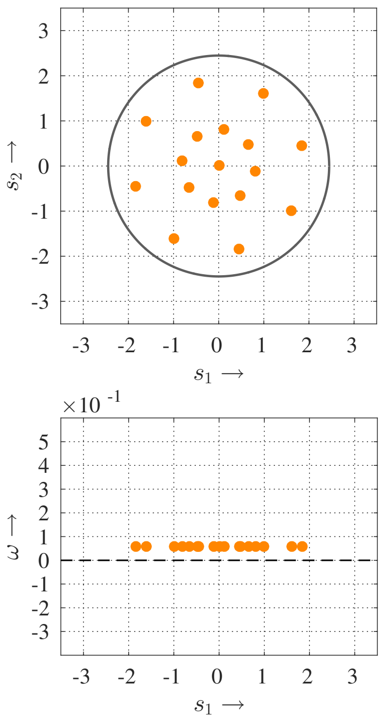
(c) 5th-degree CKF sampling.



(d) RUKF sampling with 4 iterations.



(e) Asymmetric LCD-based sampling with 17 samples.



(f) Point-symmetric LCD-based sampling with 17 samples.

Figure 2.1: State-of-the-art Gaussian sampling techniques and the proposed point-symmetric LCD-based Gaussian sampling technique. Top figures: sample positions of a 2D standard normal distribution and confidence interval of 95 % (gray circle). Bottom figures: respective weights of the samples.

Gauss–Hermite Quadrature/Kalman Filter

In [22], the Gauss–Hermite Kalman filter (GHKF) based on the Gauss–Hermite quadrature is proposed. The Gauss–Hermite quadrature is a scalar integration scheme that uses the roots of Hermite polynomials to determine the scalar sample positions and non-equal sample weights. In order to extend the quadrature to the multi-dimensional case, the product formula is applied [109], i.e., the Cartesian product, see Figure 2.1(b). However, like any product formula, the number of samples growths exponentially in the dimension N, as $M = P^N$ samples are required, where P denotes the number scalar samples used by the Gauss–Hermite quadrature. As a consequence, this sampling scheme is intractable for large dimensions N.

Cubature Kalman Filter

A special case of the UKF is proposed in [31] and called the cubature Kalman filter (CKF). The idea of the CKF is to split the integration (2.6) in a spherical cubature rule and a radial rule. Depending on the selected rules, a spherical-radial rule of a certain degree is obtained. In [31], a 3rd-degree spherical-radial rule comprising $M = 2N$ samples is suggested. However, this sampling is equivalent to the sampling of the UKF, where the sample at the state space origin has zero weight and all other samples are equally weighted.

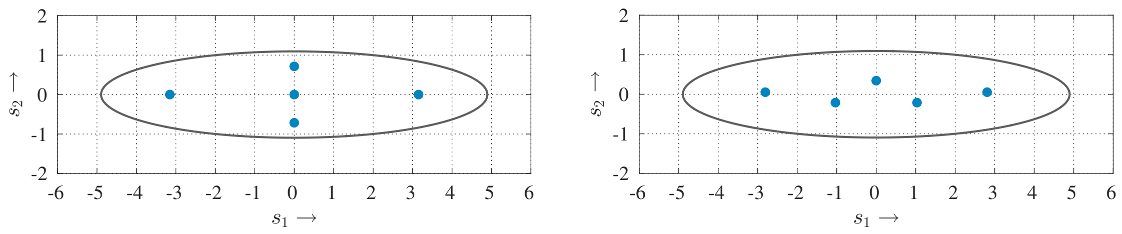
The spherical-radial rule approach is extended to an arbitrary degree in [32], which results in the high-degree CKF. Concrete formulas are given for a 5th-degree CKF with $M = 2N^2 + 1$ non-equally weighted samples, see Figure 2.1(c). Due to its higher degree, the 5th-degree CKF can exactly match the fourth-order moments of a standard normal distribution. Compared to the UKF, the high-degree CKF has the advantage that the number of samples M can be changed for a given dimension N (by changing the degree of the spherical-radial rule), but not in a “smooth” fashion, i.e., at most in a linear increase. For example, for $N = 6$, the 3rd-degree, 5th-degree, 7th-degree, 9th-degree, and 11th-degree rules comprise 12, 73, 584, 865, and 7 092 samples, respectively. Moreover, for $N \geq 5$, several sample weights become negative. If the sampling is used to compute covariance matrices, this might lead to indefinite covariance matrices due to numerical issues [110].

Randomized Unscented Kalman Filter

An extended version of the UKF, the randomized unscented Kalman filter (RUKF), is proposed in [33, 36]. Here, an arbitrary user-defined number of UKF sample sets are combined to an overall set of samples. More precisely, each UKF sample set is randomly rotated (using a random orthogonal matrix) and also randomly scaled according to the χ distribution. This gives a set of $S \cdot 2N + 1$ samples, where S denotes the number of UKF sample sets to be used. Due to the different scalings, the samples are not equally weighted and the sample located at the state space origin can be negative, see Figure 2.1(d). The advantage of the RUKF sampling is that the number samples can be changed in a linear fashion for a given dimension N. Furthermore, in contrast to simple random sampling, i.e., Monte Carlo integration, the sampling approach guarantees that the unit covariance matrix is always captured correctly. However, due to the random approach, the state space is not filled systematically. Also the creation of the random orthogonal matrices, which has to be performed online, increases the computational effort compared to the other Gaussian sampling techniques.

Localized Cumulative Distribution-Based Gaussian Sampling

A completely different approach is taken by the Gaussian sampling technique introduced in [111]. In contrast to the other approaches, it can directly compute an equally weighted Dirac mixture comprising an *arbitrary* number of samples M that approximates a given Gaussian distribution, not only a standard normal distribution. The key idea of this approach is to turn the density approximation problem into



(a) UKF sampling with equally weighted samples. **(b)** Online LCD-based sampling with $M = 5$ samples.

Figure 2.2: Different approximations of a 2D Gaussian distribution. Note the adverse sample placement of the UKF compared to the online LCD-based sampling.

an *optimization* problem. That is, the parameters of the equally weighted Dirac mixture, i.e., its sample positions, are optimized such that the Gaussian distribution is well approximated. In order to achieve this, a distance measure between the continuous Gaussian PDF and the discrete Dirac mixture is proposed that is based on the so-called localized cumulative distribution (LCD). More precisely, by computing the LCD for the Gaussian PDF and the LCD for the discrete Dirac mixture, both are transformed in the same *continuous* N-dimensional space. This in turn allows to define a distance measure between these LCDs, which gives different values for different sample positions of the Dirac mixture. Thus, the distance measure acts as optimization criterion, and minimizing this criterion means optimizing the sample positions. As a consequence, the samples are not placed randomly, which results in a much better state space coverage compared to random approaches such as Monte Carlo integration or the RUKF.

Another advantage of the LCD-based distance measure is that it is not invariant to the scaling of a Gaussian distribution. To illustrate this, we assume a 2D Gaussian with covariance matrix $\mathbf{\Sigma}^{(p)} = \mathrm{diag}(4, 0.2)$, see Figure 2.2. When now considering, for example, the sampling of the UKF that relies on the Mahalanobis transformation, we can see that the samples are still axis-aligned (Figure 2.2(a)). Thus, the state space is not well covered by these samples. That is, it would be much better to place the samples more along the larger axis of the Gaussian. In fact, this is done by the LCD-based sampling technique (Figure 2.2(b)).

However, the drawback of the LCD approach is that for each approximation of a Gaussian distribution an optimization procedure has to be conducted. When thinking, for example, of a sample-based Kalman filter, this means that for each prediction and each measurement update a time-consuming optimization is needed in order to obtain the required approximation of the current Gaussian state estimate. Consequently, for such applications the advantage of an online approximation can be dropped in favor of an offline approximation of a standard normal distribution, see Figure 2.1(e), combined with the Mahalanobis transformation. Compared to other state-of-the-art Gaussian sampling techniques, this has still the advantage of using an arbitrary number of equally weighted and deterministically placed samples. Moreover, for the special case of a standard normal distribution, closed-form solutions for the distance measure are given in [112]. Unfortunately, these formulas can only be applied to an even dimension N.

Up to now, the LCD-based sampling technique for standard normal distributions does not consider their point symmetry. In addition, the LCD approach is only a *shape* approximation. This means that the samples of the Dirac mixture are placed by the optimization procedure such that the shapes of the respective LCDs match as best as possible. However, this does not guarantee that the covariance matrix of the resulting Dirac mixture matches the covariance matrix of the Gaussian distribution to be optimized (no matter if only a standard normal distribution or any other Gaussian distribution is considered). Hence, in the next section, we will take up the promising LCD approach and address these problems in order to improve its sampling quality, see Figure 2.1(f).

2.2 Point-Symmetric LCD-Based Gaussian Sampling

Combining the LCD-based sampling scheme with the Mahalanobis transformation yields an efficient online Gaussian sampling technique. In order to further improve its quality, in this thesis we pursue three different approaches. First, we explicitly take into account the point symmetry of the standard normal distribution by using equally weighted *point-symmetric* Dirac mixtures. Compared to the original (asymmetric) LCD-based sampling, now all odd moments of the standard normal distribution, including mean and skewness, are exactly matched by the optimized Dirac mixture. In doing so, the LCD-based sampling catches up to state-of-the-art Gaussian sampling techniques, as all of them utilize point-symmetric Dirac mixtures. A side-effect of the used point-symmetric Dirac mixtures is that the number of sample positions to be optimized is reduced by half. This in turn reduces the time needed for an approximation as the sample positions to be optimized enter the computational complexity in a quadratic way. Nonetheless, because the approximation is performed offline, this has no influence on the application of the sampling. It should also be noted that besides point symmetry other symmetries such as axial symmetry could also be exploited. However, this would prevent us from using an arbitrary number of samples and would limit the optimizer's control over the sample placement. Second, after the optimization, we apply a simple sample-wise correction of the optimized Dirac mixture such that the covariance matrix of the Dirac mixture matches the identity covariance matrix of the standard normal distribution. This makes it possible to use the LCD-based Gaussian sampling for a new sample-based Kalman filter, the smart sampling Kalman filter, which will be introduced in Chapter 3. Third, we also improve the numerical stability of the LCD approach by tweaking its distance measure without affecting the sampling quality. In doing so, this allows for approximating standard normal distributions of very high dimensions, e.g., $N > 200$, compared to the originally proposed LCD-based distance measure.

In the following, we first define the parameter set describing an equally weighted point-symmetric Dirac mixture. These parameters have then to be optimized in order to approximate a standard normal distribution in an optimal way. This requires to adapt the LCD-based distance measure from [111] to the introduced point-symmetric Dirac mixtures. Subsequently, the gradients of the new distance measures are derived, which are needed for the chosen gradient-based iterative optimization procedure. Finally, we give a procedure to compute point-symmetric Dirac mixture approximations of standard normal distributions based on the introduced distance measures, their gradients, and the sample-wise Dirac mixture correction to match the unit covariance matrix.

2.2.1 Point-Symmetric Dirac Mixtures

In order to introduce equally weighted point-symmetric Dirac mixtures, we have to modify the generic Dirac mixture (2.4) and to distinguish between an even and an odd number of samples M. In case of an even number of samples $M = 2L$ with $L \in \mathbb{N}_+$, we place the samples point-symmetrically around the state space origin resulting in the Dirac mixture

$$\frac{1}{2L} \sum_{i=1}^{L} \delta(\boldsymbol{s} - \boldsymbol{s}^{(i)}) + \delta(\boldsymbol{s} + \boldsymbol{s}^{(i)}) \ , \tag{2.8}$$

with sample positions $\{\boldsymbol{s}^{(1)}, -\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(L)}, -\boldsymbol{s}^{(L)}\}$. In case of an odd number of samples $M = 2L + 1$ with $L \in \mathbb{N}_+$, we additionally place a sample fixed at the state space origin to preserve the desired point symmetry, which gives the Dirac mixture

$$\frac{1}{2L+1} \left(\delta(\boldsymbol{s}) + \sum_{i=1}^{L} \delta(\boldsymbol{s} - \boldsymbol{s}^{(i)}) + \delta(\boldsymbol{s} + \boldsymbol{s}^{(i)}) \right) \ , \tag{2.9}$$

with sample positions $\{\mathbf{0}, \boldsymbol{s}^{(1)}, -\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(L)}, -\boldsymbol{s}^{(L)}\}$. For both cases, the sample positions $\boldsymbol{s}^{(i)}$ completely describe the Dirac mixtures (2.8) and (2.9). Thus, we define the set

$$\mathcal{S} := \{\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(L)}\} \tag{2.10}$$

encompassing $L \cdot N$ values as the parameter set for both equally weighted point-symmetric Dirac mixtures.

For example, the UKF sampling schemes comprising $M = 2N$ or $M = 2N + 1$ equally weighted samples [24] are special cases of these Dirac mixtures. The first case is an even Dirac mixture with the parametrization

$$\mathcal{S} = \{\sqrt{N} \cdot \boldsymbol{e}^{(1)}, \ldots, \sqrt{N} \cdot \boldsymbol{e}^{(N)}\} \ ,$$

where $\boldsymbol{e}^{(i)}$ denotes the unit vector along the i^{th} dimension, and the second case is an odd Dirac mixture with the parametrization

$$\mathcal{S} = \{\sqrt{N + 0.5} \cdot \boldsymbol{e}^{(1)}, \ldots, \sqrt{N + 0.5} \cdot \boldsymbol{e}^{(N)}\} \ .$$

Note the larger sample spread in the latter case due to the additional point mass located at the state space origin.

It is important to note that the Dirac mixtures (2.8) and (2.9) are always point-symmetric, no matter what values (2.10) will take. As a consequence, an important property of these Dirac mixtures is that, like for a standard normal distribution, all their odd moments are zero (a proof is given in Appendix A.1). In other words, the point-symmetric Dirac mixtures capture all odd moments of a standard normal distribution.

2.2.2 Distance Measures for Point-Symmetric Dirac Mixtures

After introducing the point-symmetric Dirac mixtures, we have to find suitable parameters $\mathcal{S}$ to obtain an optimal approximation of the standard normal PDF $\mathcal{N}(\boldsymbol{s}\,; \mathbf{0}, \mathbf{I}_N)$ for a given dimension N. This requires to compare the continuous standard normal PDF with the discrete point-symmetric Dirac mixtures. For that, we transform both into the same continuous N-dimensional space with the aid of the LCD that is defined as follows.

Definition 2.1: Localized cumulative distribution [111]

Let $\boldsymbol{s} \in \mathbb{R}^N$ be a random vector with density function $f(\boldsymbol{s})$. The corresponding localized cumulative distribution is defined as

$$F(\boldsymbol{m}, b) := \int_{\mathbb{R}^N} f(\boldsymbol{s}) \cdot K(\boldsymbol{s} - \boldsymbol{m}, b) \, \mathrm{d}\boldsymbol{s} \ , \tag{2.11}$$

with $\boldsymbol{m} \in \mathbb{R}^N$, $b \in \mathbb{R}_+$, and symmetric and integrable kernel

$$K(\boldsymbol{s} - \boldsymbol{m}, b) = \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s} - \boldsymbol{m}\|_2^2}{b^2}\right) \ .$$

Here, $\boldsymbol{m}$ characterizes the location of the kernel and b its size.

First, we consider the LCD of the standard normal distribution. According to (2.11), this is an integral over the product of two (unnormalized) Gaussian PDFs and is given by [111]

$$
\begin{aligned}
F_{\mathcal{N}}(\boldsymbol{m}, b) &= \int_{\mathbb{R}^N} \mathcal{N}(\boldsymbol{s}\,; \mathbf{0}, \mathbf{I}_N) \cdot \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s}-\boldsymbol{m}\|_2^2}{b^2}\right) \mathrm{d}\boldsymbol{s} \\
&= \int_{\mathbb{R}^N} \mathcal{N}(\boldsymbol{s}\,; \mathbf{0}, \mathbf{I}_N) \cdot (2\pi)^{\frac{N}{2}} b^N \mathcal{N}(\boldsymbol{s}\,; \boldsymbol{m}, b^2\,\mathbf{I}_N)\,\mathrm{d}\boldsymbol{s} \\
&= \frac{(2\pi)^{\frac{N}{2}} b^N}{(2\pi)^{\frac{N}{2}}\sqrt{|(1+b^2)\mathbf{I}_N|}} \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{m}\|_2^2}{(1+b^2)}\right) \\
&= \left(\frac{b^2}{1+b^2}\right)^{\frac{N}{2}} \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{m}\|_2^2}{(1+b^2)}\right) \quad .
\end{aligned}
$$

Second, the respective LCDs of the point-symmetric Dirac mixtures can be simply obtained by exploiting the sifting property of the Dirac-δ distribution (as done in [111]), and are given by

$$
\begin{aligned}
F_\delta^{\mathrm{e}}(\boldsymbol{m}, b, \mathcal{S}) &= \int_{\mathbb{R}^N} \frac{1}{2L}\left(\sum_{i=1}^{L} \delta(\boldsymbol{s}-\boldsymbol{s}^{(i)}) + \delta(\boldsymbol{s}+\boldsymbol{s}^{(i)})\right) \cdot \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s}-\boldsymbol{m}\|_2^2}{b^2}\right) \mathrm{d}\boldsymbol{s} \\
&= \frac{1}{2L}\left(\sum_{i=1}^{L} \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)}-\boldsymbol{m}\|_2^2}{b^2}\right) + \exp\left(-\frac{1}{2}\frac{\|-\boldsymbol{s}^{(i)}-\boldsymbol{m}\|_2^2}{b^2}\right)\right)
\end{aligned}
$$

for the even case and by

$$
\begin{aligned}
F_\delta^{\mathrm{o}}(\boldsymbol{m}, b, \mathcal{S}) &= \int_{\mathbb{R}^N} \frac{1}{2L+1}\left(\delta(\boldsymbol{s}) + \sum_{i=1}^{L} \delta(\boldsymbol{s}-\boldsymbol{s}^{(i)}) + \delta(\boldsymbol{s}+\boldsymbol{s}^{(i)})\right) \cdot \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s}-\boldsymbol{m}\|_2^2}{b^2}\right) \mathrm{d}\boldsymbol{s} \\
&= \frac{1}{2L+1}\left(\exp\left(-\frac{1}{2}\frac{\|\boldsymbol{m}\|_2^2}{b^2}\right) + \sum_{i=1}^{L} \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)}-\boldsymbol{m}\|_2^2}{b^2}\right) + \right. \\
&\qquad\qquad \left. \exp\left(-\frac{1}{2}\frac{\|-\boldsymbol{s}^{(i)}-\boldsymbol{m}\|_2^2}{b^2}\right)\right)
\end{aligned}
$$

for the odd case, respectively. Note that the LCDs of the Dirac mixtures (2.8) and (2.9) still depend on the sampling parameters $\mathcal{S}$.

Now, in order to construct our pursued distance measures, we compare the above LCDs by utilizing a modified Cramér–von Mises (CvM) distance as in the asymmetric LCD approach. However, the modified CvM distance used here slightly differs from the original one due to numerical reasons.

Definition 2.2: Modified Cramér–von Mises distance

For two LCDs $F(\boldsymbol{m}, b)$ and $\tilde{F}(\boldsymbol{m}, b)$, the modified Cramér–von Mises distance D is defined as

$$
D(F, \tilde{F}) := \int_0^\infty w(b) \int_{\mathbb{R}^N} \left(F(\boldsymbol{m}, b) - \tilde{F}(\boldsymbol{m}, b)\right)^2 \mathrm{d}\boldsymbol{m}\,\mathrm{d}b \ , \tag{2.12}
$$

with weighting function

$$
w(b) = \begin{cases} \pi^{-\frac{N}{2}} b^{1-N} & , b \in (0, b_{max}] \\ 0 & , \textit{elsewhere} \end{cases}
$$

and $b_{max} \in \mathbb{R}_+$.

Compared to the original definition from [111], we have an additional term $\pi^{-\frac{N}{2}}$ in the weighting function w. Without this extra term, the modified CvM distance D would be unbounded for an increasing dimension N, and thus would make the distance measures numerically unstable.

First, we build the distance measure between the standard normal distribution and the even point-symmetric Dirac mixture by computing the modified CvM distance between the LCDs $F_{\mathcal{N}}$ and F_{δ}^{e}.

Theorem 2.1:

The modified CvM distance D^{e} between the LCDs $F_{\mathcal{N}}$ and F_{δ}^{e} is given by

$$D^{\mathrm{e}}(\mathcal{S}) = D(F_{\mathcal{N}}, F_{\delta}^{e}) = D_1^{\mathrm{e}} - 2D_2^{\mathrm{e}}(\mathcal{S}) + D_3^{\mathrm{e}}(\mathcal{S}) \ , \tag{2.13}$$

where

$$D_1^{\mathrm{e}} = \int_0^{b_{max}} b \left(\frac{b^2}{1+b^2} \right)^{\frac{N}{2}} \mathrm{d}b \ ,$$

$$D_2^{\mathrm{e}}(\mathcal{S}) = \int_0^{b_{max}} \frac{2b}{2L} \left(\frac{2b^2}{1+2b^2} \right)^{\frac{N}{2}} \cdot \sum_{i=1}^{L} \exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)}\|_2^2}{(1+2b^2)} \right) \mathrm{d}b \ ,$$

$$D_3^{\mathrm{e}}(\mathcal{S}) = \int_0^{b_{max}} \frac{2b}{(2L)^2} \sum_{i=1}^{L} \sum_{j=1}^{L} \left(\exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)} - \boldsymbol{s}^{(j)}\|_2^2}{2b^2} \right) + \exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)} + \boldsymbol{s}^{(j)}\|_2^2}{2b^2} \right) \right) \mathrm{d}b \ .$$

Proof. The proof is given in Appendix A.2. ■

Note that the integration over b is bounded by b_{max} due to the bounded support of the weighting function w. Similar to the asymmetric LCDs [111], we can compute the integral in D_3^{e} analytically, which is given by the next theorem.

Theorem 2.2:

For a given b_{max}, the following expression for D_3^{e} can be obtained

$$\begin{aligned} D_3^{\mathrm{e}}(\mathcal{S}) = \frac{2}{(2L)^2} \sum_{i=1}^{L} \sum_{j=1}^{L} \Bigg(\frac{b_{max}^2}{2} \Bigg(& \exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)} - \boldsymbol{s}^{(j)}\|_2^2}{2b_{max}^2} \right) + \\ & \exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)} + \boldsymbol{s}^{(j)}\|_2^2}{2b_{max}^2} \right) \Bigg) + \\ & \frac{1}{8} \Bigg(\|\boldsymbol{s}^{(i)} - \boldsymbol{s}^{(j)}\|_2^2 \operatorname{Ei}_0\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)} - \boldsymbol{s}^{(j)}\|_2^2}{2b_{max}^2} \right) + \\ & \|\boldsymbol{s}^{(i)} + \boldsymbol{s}^{(j)}\|_2^2 \operatorname{Ei}_0\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)} + \boldsymbol{s}^{(j)}\|_2^2}{2b_{max}^2} \right) \Bigg) \Bigg) \ , \end{aligned}$$

where $\operatorname{Ei}_0(x)$ is defined as

$$\operatorname{Ei}_0(x) := \begin{cases} 0 & , \textit{if } x = 0 \\ \operatorname{Ei}(x) & , \textit{elsewhere} \end{cases}$$

and $\operatorname{Ei}(x)$ denotes the exponential integral

$$\operatorname{Ei}(x) := \int_{-\infty}^{x} \frac{e^t}{t} \,\mathrm{d}t \ .$$

Proof. The proof is given in Appendix A.3. ■

Although the exponential integral Ei in Theorem 2.2 cannot be computed analytically, there exist efficient numerical approximations for it we make use of in our implementation, e.g., see [113, Chap. 5].

Second, we build the distance measure between the standard normal distribution and the odd point-symmetric Dirac mixture by computing the modified CvM distance between the LCDs $F_{\mathcal{N}}$ and F_{δ}^{o}.

Theorem 2.3:

The modified CvM distance D^{o} between the LCDs $F_{\mathcal{N}}$ and F_{δ}^{o} is given by

$$D^{\mathrm{o}}(\mathcal{S}) = D(F_{\mathcal{N}}, F_{\delta}^{o}) = D_1^{\mathrm{o}} - 2D_2^{\mathrm{o}}(\mathcal{S}) + D_3^{\mathrm{o}}(\mathcal{S}) \ , \tag{2.14}$$

where

$$\begin{aligned} D_1^{\mathrm{o}} &= D_1^{\mathrm{e}} \ , \\ D_2^{\mathrm{o}}(\mathcal{S}) &= \frac{2L}{2L+1} D_2^{\mathrm{e}}(\mathcal{S}) + \int_0^{b_{max}} \frac{b}{2L+1} \left(\frac{2b^2}{1+2b^2} \right)^{\frac{N}{2}} \mathrm{d}b \ , \\ D_3^{\mathrm{o}}(\mathcal{S}) &= \frac{(2L)^2}{(2L+1)^2} D_3^{\mathrm{e}}(\mathcal{S}) + \frac{b_{max}^2}{2(2L+1)^2} + \int_0^{b_{max}} \frac{4b}{(2L+1)^2} \sum_{i=1}^{L} \exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)}\|_2^2}{2b^2} \right) \mathrm{d}b \ . \end{aligned}$$

Proof. The proof is given in Appendix A.4. ■

The extra terms in D_2^{o} and D_3^{o}, compared to their even counterparts, reflect the additional point mass of the sample placed at the state space origin. Furthermore, like for the part D_3^{e} of D^{e}, we can compute D_3^{o} in closed form.

Theorem 2.4:

For a given b_{max}, the following expression for D_3^{o} can be obtained

$$\begin{aligned} D_3^{\mathrm{o}}(\mathcal{S}) = {} & \frac{(2L)^2}{(2L+1)^2} D_3^{\mathrm{e}}(\mathcal{S}) + \frac{b_{max}^2}{2(2L+1)^2} + \\ & \frac{4}{(2L+1)^2} \sum_{i=1}^{L} \left(\frac{b_{max}^2}{2} \exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)}\|_2^2}{2b_{max}^2} \right) + \frac{1}{8} \|\boldsymbol{s}^{(i)}\|_2^2 \, \mathrm{Ei}_0\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)}\|_2^2}{2b_{max}^2} \right) \right) \ , \end{aligned}$$

where Ei_0 is defined as in Theorem 2.2.

Proof. The proof is given in Appendix A.5. ■

Finally, after introducing the distance measures between the standard normal distribution and the point-symmetric Dirac mixtures, will give some concluding remarks.

- Regarding the above mentioned numerical stability, a proof of the boundedness of the distance measures (2.13) and (2.14) is given in Appendix A.6.
- Both distance measures are invariant under rotation/reflection (a proof is given in Appendix A.7).
- Due to the constrained sample placement introduced by the point symmetry, the proposed distance measures can be seen as constrained versions of the original asymmetric LCD-based distance measure from [111]. That is, Dirac mixtures obtained by minimizing D^{e} or D^{o} can be suboptimal with respect to the asymmetric distance measure. However, our goal is to approximate the standard normal distribution in an optimal way, and distance measures are only tools to achieve this. Here, optimality additionally means preserving the point symmetry of the standard normal distribution. Hence, slightly modified distance measures are required to reflect this.

- The numerically stable modified CvM distance (2.12) can immediately replace the original distance used by the asymmetric sample computation. In doing so, the asymmetric LCD-based sampling can also profit from its numerical stability. In fact, this will be done in the evaluation to get a fair comparison between both approaches.

2.2.3 Gradients of the Distance Measures

Minimizing the distance measures D^{e} and D^{o} requires an optimization procedure. We choose to use a gradient-based iterative optimization procedure, namely a quasi-Newton method. This requires the partial derivatives of the distance measures with respect to all $L \cdot N$ parameters in $\mathcal{S}$. In order to avoid numerical approximations of the derivatives and simultaneously improve quality and runtime of the optimization, we will next derive the partial derivatives of D^{e} and D^{o}.

The partial derivatives for the even distance measure D^{e} are given by the partial derivatives of D_2^{e} and D_3^{e} according to

$$\frac{\partial D^{\mathrm{e}}(\mathcal{S})}{\partial s_d^{(i)}} = -2\frac{\partial D_2^{\mathrm{e}}(\mathcal{S})}{\partial s_d^{(i)}} + \frac{\partial D_3^{\mathrm{e}}(\mathcal{S})}{\partial s_d^{(i)}} \quad , \quad \forall i \in \{1, \ldots, L\} \quad , \quad \forall d \in \{1, \ldots, N\} \quad ,$$

where the scalar $s_d^{(i)}$ denotes the entry for the d^{th} dimension of the i^{th} sample $\boldsymbol{s}^{(i)}$. The partial derivatives $\frac{\partial D_2^{\mathrm{e}}(\mathcal{S})}{\partial s_d^{(i)}}$ and $\frac{\partial D_3^{\mathrm{e}}(\mathcal{S})}{\partial s_d^{(i)}}$ can be simply obtained by applying the chain rule according to

$$\frac{\partial D_2^{\mathrm{e}}(\mathcal{S})}{\partial s_d^{(i)}} = -\frac{s_d^{(i)}}{2L} \int_0^{b_{\max}} \frac{2b}{(1+2b^2)} \left(\frac{2b^2}{1+2b^2}\right)^{\frac{N}{2}} \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)}\|_2^2}{(1+2b^2)}\right) \mathrm{d}b \quad ,$$

$$\begin{aligned}\frac{\partial D_3^{\mathrm{e}}(\mathcal{S})}{\partial s_d^{(i)}} = -\frac{2}{(2L)^2} \int_0^{b_{\max}} \frac{1}{b} \sum_{j=1}^{L} \Bigg(&(s_d^{(i)} - s_d^{(j)}) \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)} - \boldsymbol{s}^{(j)}\|_2^2}{2b^2}\right) + \\ &(s_d^{(i)} + s_d^{(j)}) \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)} + \boldsymbol{s}^{(j)}\|_2^2}{2b^2}\right)\Bigg) \mathrm{d}b \quad .\end{aligned}$$

To speed up the computation of the partial derivatives, we can use the following theorem.

Theorem 2.5:
For a given b_{max}, the following expression for $\frac{\partial D_3^{\mathrm{e}}(\mathcal{S})}{\partial s_d^{(i)}}$ can be obtained

$$\begin{aligned}\frac{\partial D_3^{\mathrm{e}}(\mathcal{S})}{\partial s_d^{(i)}} = \frac{1}{(2L)^2} \sum_{j=1}^{L} \Bigg(&(s_d^{(i)} - s_d^{(j)}) \operatorname{Ei}_0\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)} - \boldsymbol{s}^{(j)}\|_2^2}{2b_{max}^2}\right) + \\ &(s_d^{(i)} + s_d^{(j)}) \operatorname{Ei}_0\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)} + \boldsymbol{s}^{(j)}\|_2^2}{2b_{max}^2}\right)\Bigg) \quad ,\end{aligned}$$

where Ei_0 *is defined as in Theorem 2.2.*

Proof. The proof is given in Appendix A.8. ■

Considering the partial derivatives for the odd distance measure D^{o}, we obtain analogously

$$\frac{\partial D^{\mathrm{o}}(\mathcal{S})}{\partial s_d^{(i)}} = -2\frac{\partial D_2^{\mathrm{o}}(\mathcal{S})}{\partial s_d^{(i)}} + \frac{\partial D_3^{\mathrm{o}}(\mathcal{S})}{\partial s_d^{(i)}} \quad , \quad \forall i \in \{1, \ldots, L\} \quad , \quad \forall d \in \{1, \ldots, N\} \quad ,$$

with

$$\frac{\partial D_2^{\mathrm{o}}(\mathcal{S})}{\partial s_d^{(i)}} = \frac{2L}{2L+1} \frac{\partial D_2^{\mathrm{e}}(\mathcal{S})}{\partial s_d^{(i)}} \ ,$$

$$\frac{\partial D_3^{\mathrm{o}}(\mathcal{S})}{\partial s_d^{(i)}} = \frac{(2L)^2}{(2L+1)^2} \frac{\partial D_3^{\mathrm{e}}(\mathcal{S})}{\partial s_d^{(i)}} - \frac{2 s_d^{(i)}}{(2L+1)^2} \int_0^{b_{\max}} \frac{1}{b} \exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)}\|_2^2}{2b^2} \right) \mathrm{d}b \ .$$

To again speed up the computation, we can make use of the next theorem.

Theorem 2.6:

For a given b_{max}, the following expression for $\frac{\partial D_3^{\mathrm{o}}(\mathcal{S})}{\partial s_d^{(i)}}$ can be obtained

$$\frac{\partial D_3^{\mathrm{o}}(\mathcal{S})}{\partial s_d^{(i)}} = \frac{(2L)^2}{(2L+1)^2} \frac{\partial D_3^{\mathrm{e}}(\mathcal{S})}{\partial s_d^{(i)}} + \frac{s_d^{(i)}}{(2L+1)^2} \operatorname{Ei}_0\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)}\|_2^2}{2b_{max}^2} \right) \ ,$$

where Ei_0 *is defined as in Theorem 2.2.*

Proof. The proof is given in Appendix A.9. ■

2.2.4 Compute the Optimal Point-Symmetric Sampling

Before we can minimize the introduced distance measures D^{e} and D^{o}, we have to specify a maximum kernel size $b_{\max}$. Generally speaking, the larger the dimension N the larger $b_{\max}$ has to be in order to consider all sample positions appropriately. Empirically, we have found that a value of $b_{\max} = 200$ is large enough for up to $N \leq 10\,000$. Further, the remaining integrals over the kernel size b in the distance measures and partial derivatives have to solved for a given set of parameters $\mathcal{S}$. Here, we approximate these integrals with an adaptive 31-point Gauss–Kronrod quadrature [114].

As in [111], the actual minimization of the distance measures is performed with the limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) quasi-Newton optimization [115, Sec. 7.2]. The limited-memory variant is essential due to the large number of parameters to be optimized. More precisely, the point-symmetric Dirac mixture parameter set $\mathcal{S}$ encompasses $L \cdot N$ individual parameters. Thus, the Hessian of D^{e} or D^{o} (or its approximation used by quasi-Newton methods) would consist of $(L \cdot N)^2$ values. As an example, consider a 100-dimensional Dirac mixture with 1 000 samples. The Hessian of the distance measures would require ≈ 20 gigabyte of data (in case of 64-bit floating-point numbers). For only 500 samples, the Hessian would still be larger than 4 gigabytes. Storing and working with such Hessians is intractable, of course. Fortunately, the data consumption of the limited-memory variant of the BFGS procedure increases only linearly with the number of parameters to be optimized, making it the ideal optimization procedure for the considered Dirac mixture approximation problem.

The entire procedure for computing equally weighted point-symmetric Dirac mixtures is given in Algorithm 2.1. In line 1, the number of point-symmetric sample pairs L is computed. Then, initial Dirac mixture parameters $\mathcal{S}_0$ are obtained by randomly drawing L samples from an N-dimensional standard normal distribution (line 2). Depending on the number of samples M, the L-BFGS procedure uses D^{e} or D^{o} (and their respective partial derivatives) as distance measure to optimize the parameters $\mathcal{S}_0$ resulting the Dirac mixture parameters $\mathcal{S}$ (lines 3–7). In order to get a better understanding of how the L-BFGS procedure optimizes a Dirac mixture, we have a closer look on the optimization of a 2D point-symmetric Dirac mixture comprising 13 samples, see Figure 2.3. In each iteration of the quasi-Newton method, the parameters $\mathcal{S}$ from the last iteration (the initial random parameters in the first iteration)

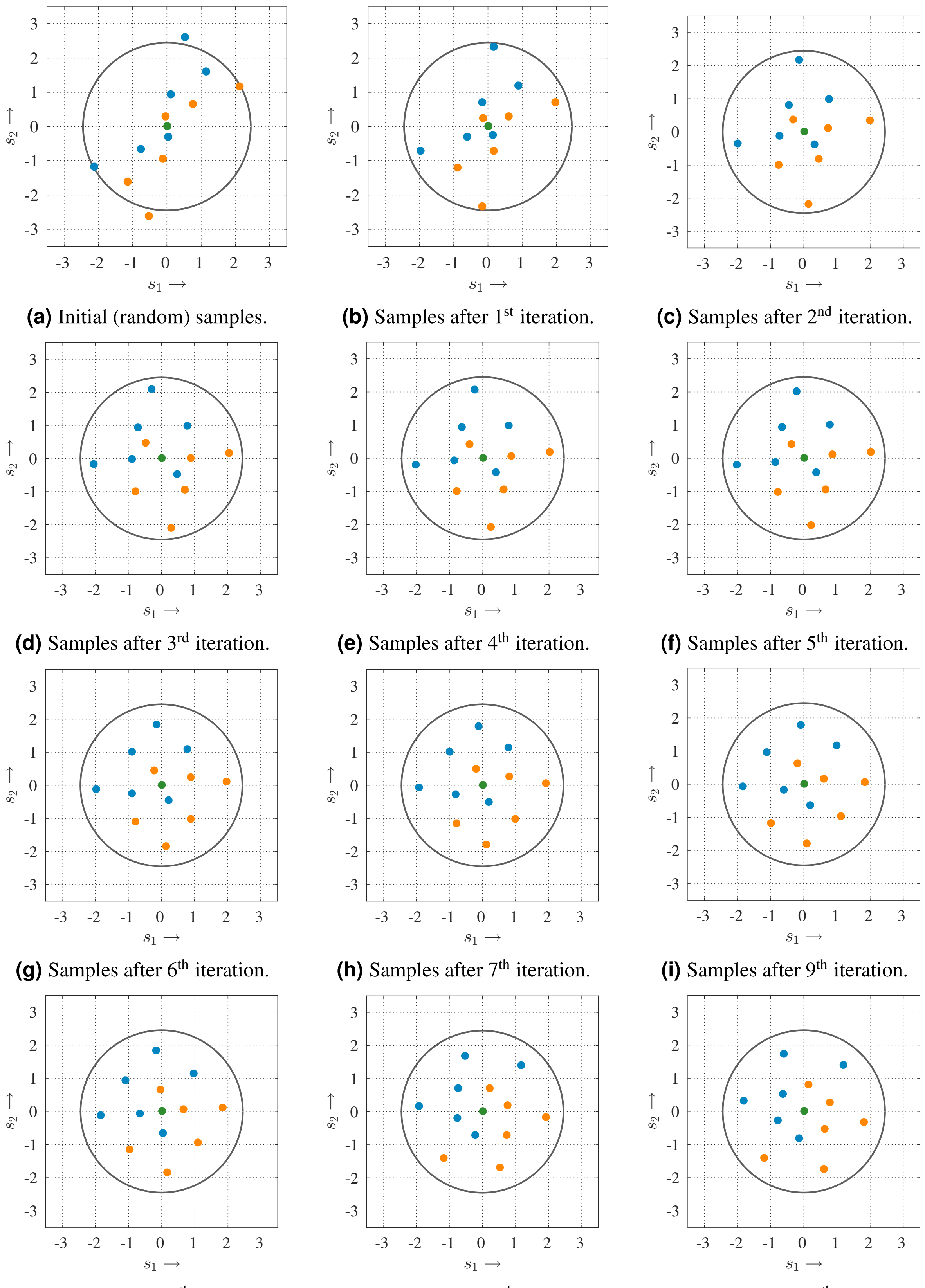
```
Input:   dimension $N$, number of samples $M$, maximum kernel size $b_{\max}$
Output:  set of equally weighted samples $\mathcal{D}$ approximating $\mathcal{N}(\boldsymbol{s}\,;\mathbf{0},\mathbf{I}_N)$

 1: $L \leftarrow \lfloor \frac{M}{2} \rfloor$
    // Draw initial point-symmetric sample positions
 2: $\mathcal{S}_0 \leftarrow \{\boldsymbol{s}_0^{(i)}\}_{i=1}^{L}$, with $\boldsymbol{s}_0^{(i)} \sim \mathcal{N}(\mathbf{0},\mathbf{I}_N)$
    // Perform optimization
 3: if $M$ is even then
 4:     $\mathcal{S} \leftarrow$ L-BFGS$(D^{\mathrm{e}}, b_{\max}, \mathcal{S}_0)$
 5: else
 6:     $\mathcal{S} \leftarrow$ L-BFGS$(D^{\mathrm{o}}, b_{\max}, \mathcal{S}_0)$
 7: end if
    // Covariance matrix correction
 8: $\mathbf{\Sigma}^{(s)} \leftarrow \frac{2}{M}\sum_{i=1}^{L} \boldsymbol{s}^{(i)}(\boldsymbol{s}^{(i)})^\top$
 9: $\mathbf{L} \leftarrow \mathrm{chol}(\mathbf{\Sigma}^{(s)})$
10: $\boldsymbol{s}^{(i)} \leftarrow \mathbf{L}^{-1}\cdot\boldsymbol{s}^{(i)} \quad , \quad \forall i \in \{1,\ldots,L\}$
    // Construct Dirac mixture
11: if $M$ is even then
12:     $\mathcal{D} \leftarrow \{\boldsymbol{s}^{(1)}, -\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(L)}, -\boldsymbol{s}^{(L)}\}$
13: else
14:     $\mathcal{D} \leftarrow \{\mathbf{0}, \boldsymbol{s}^{(1)}, -\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(L)}, -\boldsymbol{s}^{(L)}\}$
15: end if
```

Algorithm 2.1: Proposed point-symmetric LCD-based Gaussian sampling.

are changed in some way, that is, the L samples $\boldsymbol{s}^{(i)}$ (and implicitly their point-symmetric counterparts $-\boldsymbol{s}^{(i)}$) are moved in the state space such that the considered distance measure is minimized. Typically, the most extensive movements happen in the first iterations, whereas only smaller corrections happen in the last iterations (like in Figure 2.3).

However, like for the asymmetric LCD-based sampling, minimizing the distance measures results only in a shape approximation of the involved LCDs. This means in particular that no identity covariance matrix of the optimized Dirac mixture is guaranteed. Consequently, we propose to additionally constrain the possible Dirac mixtures by transforming the optimized sample positions $\mathcal{S}$ after they have been optimized. This is done in lines 8–10. First, is covariance matrix of the optimized Dirac mixture is computed. Then, its Cholesky decomposition is computed. Finally, the inverse of the decomposition is used to transform each sample $\boldsymbol{s}^{(i)}$ of $\mathcal{S}$. A proof for this covariance matrix correction is given in Appendix A.10. Besides this approach, we also considered to add the unit covariance matrix as an explicit constraint to the optimization procedure. However, it turned out that the sample covariance matrix of the resulting Dirac mixtures were less accurate compared to the proposed transformation approach. Moreover, the explicit constraints made the optimization procedure much more time-consuming as they increase quadratically in the dimension N (the covariance matrix has $\frac{N(N+1)}{2}$ unique parameters). Thus, we choose to rely on the transformation approach to guarantee a unit covariance matrix. Finally, the equally weighted Dirac mixture approximating $\mathcal{N}(\boldsymbol{s}\,;\mathbf{0},\mathbf{I}_N)$ is constructed in lines 11–15.

Various Dirac mixture approximations, computed with Algorithm 2.1, of a 2D standard normal distribution comprising different number of samples are depicted in Figure 2.4. The point-symmetric



(a) Initial (random) samples.

(b) Samples after 1st iteration.

(c) Samples after 2nd iteration.

(d) Samples after 3rd iteration.

(e) Samples after 4th iteration.

(f) Samples after 5th iteration.

(g) Samples after 6th iteration.

(h) Samples after 7th iteration.

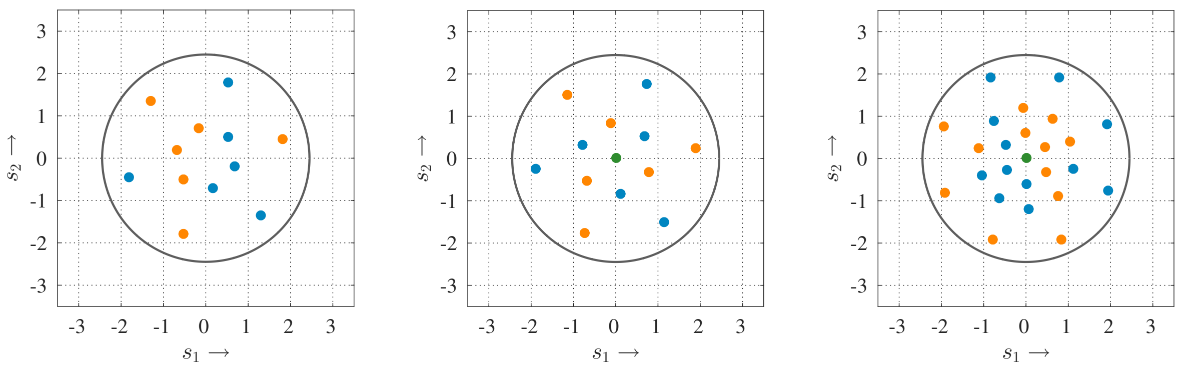
(i) Samples after 9th iteration.

(j) Samples after 19th iteration.

(k) Samples after 24th iteration.

(l) Samples after 40th iteration.

Figure 2.3: Computation of a 2D point-symmetric Dirac mixture comprising 13 samples: sample positions $\boldsymbol{s}^{(i)}$ (blue), point-symmetric counterparts $-\boldsymbol{s}^{(i)}$ (orange), and fixed sample placed at the state space origin (green) during the quasi-Newton optimization.



(a) Approx. with 12 samples. **(b)** Approx. with 13 samples. **(c)** Approx. with 25 samples.

Figure 2.4: Various Dirac mixture approximations of a 2D standard normal distribution computed with Algorithm 2.1.

arrangement around the state space origin can be clearly seen. Note the subtle difference in the sample spread of the samples near the state space origin between Figure 2.4(a) and Figure 2.4(b). This is due to point mass introduced by the additional sample located at the state space origin. It is also important to note that a Dirac mixture for a given dimension N and number of samples M is *not* unique. The reason for this is that the initial Dirac mixture parameters are chosen randomly and that the utilized distance measures D^{e} and D^{o} are invariant under rotation. For example, consider the Dirac mixture approximations obtain from Algorithm 2.1 depicted in Figure 2.5. All of them approximate a 2D standard normal distribution with 14 samples. Their sample placements, however, are (much) different.

Although it is intended to use Algorithm 2.1 offline, e.g., not during state estimation, the runtime of optimization is still acceptable. On an Intel Core i7-3770 CPU (3.4 GHz) and implemented in C++ [176], it takes only about 4 minutes to compute a Dirac mixture approximating a 500D standard normal distribution with 10 000 samples and 35 minutes to compute a Dirac mixture approximating a 1000D standard normal distribution with 20 000 samples.

2.3 Evaluation

In this section, we compare the performance of the proposed point-symmetric LCD-based sampling with the original asymmetric version and other state-of-the-art Gaussian sampling techniques. In order to make a fair comparison, we use the numerically stable modified CvM distance (2.12) to compute asymmetric LCD-based samples and subsequently apply the proposed sample correction such that the asymmetric samples possess a unit covariance matrix as well. First, we study how higher-order moments of multivariate standard normal distributions are approximated by the sampling techniques. Second, we compute moments of a Fourier series that is given by random coefficients and random input angles.

2.3.1 Higher-Order Moments of a Standard Normal Distribution

We are interested in how well Gaussian sampling techniques approximate higher-order moments of multivariate standard normal distributions. That is, we want to compute the expectations

$$\mathbb{E}[s_1^{n_1} s_2^{n_2} \cdots s_N^{n_N}] = \int_{\mathbb{R}^N} s_1^{n_1} s_2^{n_2} \cdots s_N^{n_N} \mathcal{N}(\boldsymbol{s}\,; \mathbf{0}, \mathbf{I}_N) \,\mathrm{d}\boldsymbol{s} \enspace ,$$

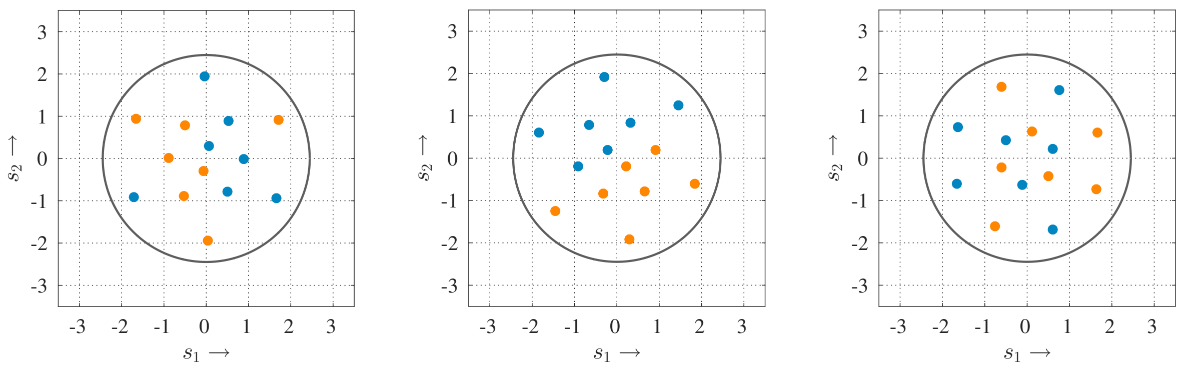


Figure 2.5: Different approximation results for a 2D standard normal distribution comprising 14 samples computed by the point-symmetric LCD-based sampling.

with $n_i \in \mathbb{N}$ and

$$\sum_{i=1}^{N} n_i = m \ , \quad 0 \leq n_i \leq m \ ,$$

for different dimensions N and moment orders $m \in \mathbb{N}$. For given N and m, the number of possible combinations $J_{N,m}$ to select the values for n_i is equal to the number of terms in a multinomial sum with N summands raised to the power m, i.e.,

$$J_{N,m} = \binom{m+N-1}{N-1} = \frac{(m+N-1)!}{(N-1)!m!} \ .$$

Hence, the m^{th} moment is characterized by $J_{N,m}$ distinct values.

We assess the approximation quality of the sampling techniques with the aid of a normalized moment error defined as

$$\sqrt{\frac{1}{J_{N,m}} \sum_{j=1}^{J_{N,m}} (\mathbb{E}_j^{\text{t}} - \mathbb{E}_j^{\text{s}})^2} \ , \tag{2.15}$$

where the $\mathbb{E}_j$ denote one of the $J_{N,m}$ possible combinations for the m^{th} moment, the superscript “t” the true moment value and “s” the value obtained the respective sampling technique. The true moment values are computed analytically with the formula proposed in [116].

Comparison of the LCD-Based Gaussian Sampling Techniques

First, we compare the asymmetric and point-symmetric Gaussian LCD-based sampling schemes for a two-dimensional standard normal distribution, i.e., $N = 2$. We perform 100 Monte Carlo runs. In each run, we compute new sets of samples with both variants and build the normalized moment errors for the moment orders $m \in \{3, 4, 5, 6, 7, 8\}$.

Figure 2.6 shows the minimum and maximum moment errors for different number of samples over all Monte Carlo runs. First of all, we notice that the new point-symmetric variant has no errors in the odd moments as expected. The asymmetric variant, however, has errors, which are especially larger for a small amount of samples but decrease for an increasing number of samples. When looking at the even moments, we see that the minimum and maximum errors for both variants are very similar, and that the errors also decrease when the number of samples rises. Nevertheless, for a small number of samples, the minimum errors of the new point-symmetric sampling are larger than the minimum errors of the asymmetric version. This can be explained by the fact that the exactly captured odd moments lead to an increase in the higher-order even moment errors, or in other words: errors in the odd moments allow for smaller errors in the even moments. Fortunately, with an increase in the number of samples, the even moment errors become as small as the errors of the asymmetric version. Hence, we can conclude that the point-symmetric LCD-based sampling exactly captures the odd moments, while it nearly keeps the approximation quality of the original asymmetric variant for the even moments. Moreover, the error spread of both variants is rather small, which indicates that all the computed sample sets nearly exhibit the same moment quality.

Comparison with Other Point-Symmetric Gaussian Sampling Techniques

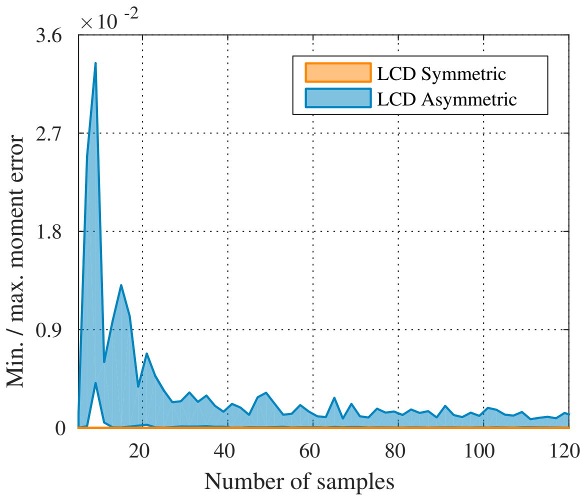
Second, we compare the new point-symmetric LCD-based sampling against other state-of-the-art sampling approaches. In many practical applications, 3D and 6D Gaussian distributions are of special interest. For example, the location and orientation in 2D or the position in 3D can be estimated using a three-dimensional system state. When additionally considering velocities in the 2D case or the orientation in the 3D case, a six-dimensional state is required. Thus, we now choose to study the approximations of standard normal distributions with these two dimensions, i.e., $N \in \{3, 6\}$.

We compare the performance of

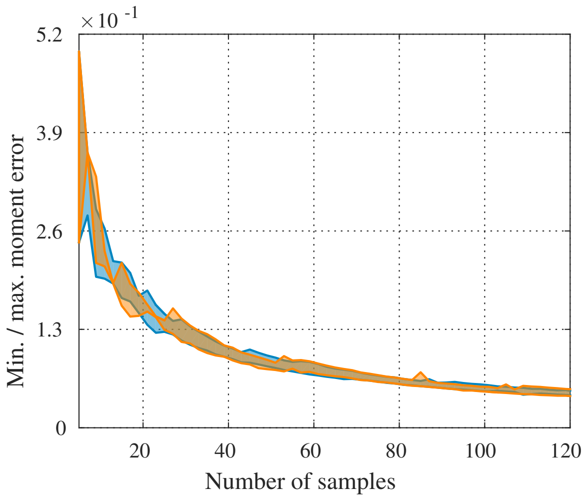
- the point-symmetric LCD-based sampling with different numbers of samples,
- the RUKF sampling with different numbers of samples,
- the 5th-degree CKF sampling,
- the GHKF sampling with two quadrature points, and
- the UKF sampling with equally weighted samples.

As all these Gaussian sampling techniques use point-symmetric samples, they exactly capture odd moments and we solely consider the even moments $m \in \{4, 6, 8\}$. Again, we perform 100 Monte Carlo runs due to the non-unique sample sets computed by both the point-symmetric LCD sampling and the RUKF sampling. Moreover, the sample-based moment computation is not invariant under sample rotations. Thus, the same holds for the normalized moment error (2.15). In order to mitigate this, we also randomly rotate the sample sets of the 5th-degree CKF, GHKF, and UKF, i.e., like for the point-symmetric LCD-based sampling and the RUKF, we have 100 different sample sets for each sampling technique.

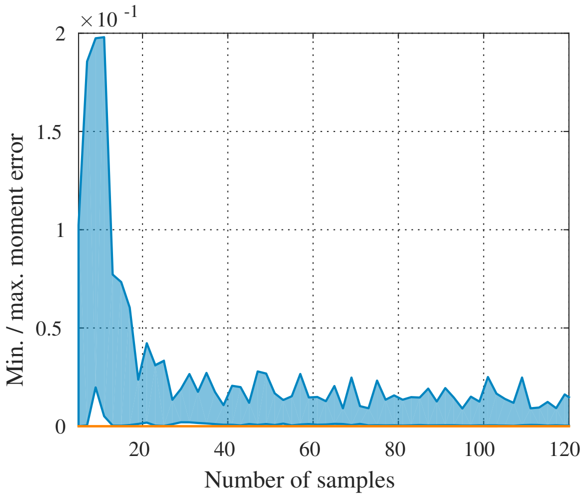
The average, minimum, and maximum normalized moment errors are depicted in Figures 2.7 and 2.8 for the 3D and 6D case, respectively. The 5th-degree CKF, GHKF, and UKF only have a fixed number of samples, so they are depicted as a bar at their respective employed number of samples. It can be seen that for all moments, the point-symmetric LCD-based approach and UKF have nearly identical error properties when using the same number of samples. This is due to the facts that both employ equally weighted samples and that the point-symmetric arrangement of the LCD-based samples forces the optimizer to place the samples in the way it is done by the UKF (except for the rotation). Opposed to this, the error properties of the RUKF, which scales the utilized UKF sample sets randomly, are not the same as for the UKF and point-symmetric LCD-based sampling (when using $2N + 1$ samples). Furthermore, all average moment errors of the point-symmetric LCD-based sampling are smaller than the errors of the RUKF and GHKF (for same number of samples). Regarding the error spread, the point-symmetric LCD-based sampling has a much smaller variability than the RUKF. Often, its errors are as small as the smallest error of the RUKF or even smaller, especially for the 6D case. Also the error characteristics in general are much smoother for the point-symmetric LCD-based sampling compared to the RUKF. Finally, the 5th-degree CKF is the only sampling scheme that matches the 4th moment exactly. The reason for this is that its employed spherical-radial rule has a 5th-degree accuracy [32].



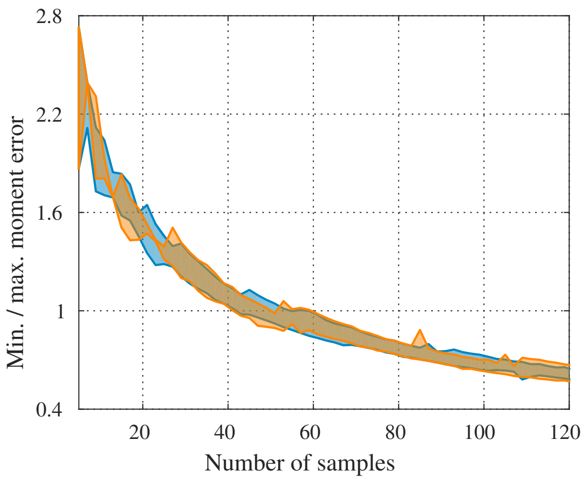
(a) Errors for 3rd moment.



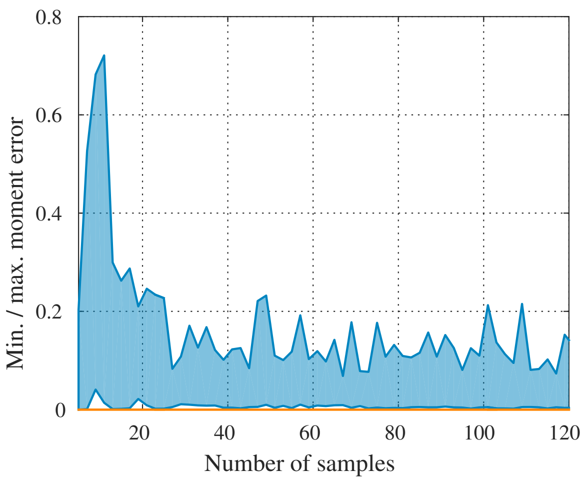
(b) Errors for 4th moment.



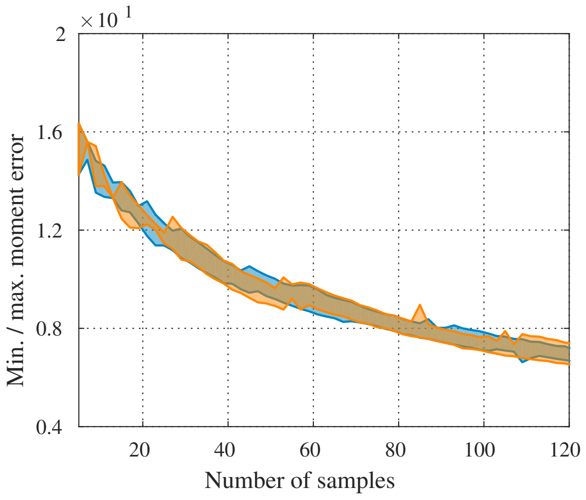
(c) Errors for 5th moment.



(d) Errors for 6th moment.



(e) Errors for 7th moment.



(f) Errors for 8th moment.

Figure 2.6: Minimum and maximum normalized moment errors for a two-dimensional standard normal distribution.

2.3.2 Moments of a Fourier Series

In the second evaluation, we consider a Fourier series

$$r(\boldsymbol{c}, \phi) = \frac{a^{(0)}}{2} + \sum_{j=1}^{F} a^{(j)} \cos(j\phi) + b^{(j)} \sin(j\phi)$$

that is defined by a fixed number of $2F + 1$ real coefficients stored in the parameter vector

$$\boldsymbol{c} = \left[a^{(0)}, a^{(1)}, b^{(1)}, \ldots, a^{(F)}, b^{(F)}\right]^\top \quad ,$$

and is evaluated for a given angle $\phi \in \mathbb{R}$ [84]. A Fourier series is a 2π-periodic function and it will be later used in Chapter 5 to describe the shape of an a priori unknown object in the context of extended object tracking.

For the evaluation of the new point-symmetric LCD-based Gaussian sampling, we consider a Fourier series with 7 coefficients, i.e., $F = 3$ and $\boldsymbol{c} \in \mathbb{R}^7$. For two different sets of coefficients, the series are plotted for $\phi \in [-2\pi, 4\pi]$ in Figure 2.9. Further, we have a joint random vector consisting of coefficients and angle

$$\boldsymbol{p} = \begin{bmatrix} \boldsymbol{c} \\ \phi \end{bmatrix} \in \mathbb{R}^8$$

that is assumed to be Gaussian distributed according to $\boldsymbol{p} \sim \mathcal{N}(\hat{\boldsymbol{p}}, \boldsymbol{\Sigma}^{(p)})$. Now, we want to compute mean

$$\hat{r} = \int_{\mathbb{R}} \int_{\mathbb{R}^7} r(\boldsymbol{c}, \phi) \cdot \mathcal{N}(\boldsymbol{p}\,; \hat{\boldsymbol{p}}, \boldsymbol{\Sigma}^{(p)})\, \mathrm{d}\boldsymbol{c}\, \mathrm{d}\phi \tag{2.16}$$

and variance

$$\Sigma^{(r)} = \int_{\mathbb{R}} \int_{\mathbb{R}^7} (r(\boldsymbol{c}, \phi) - \hat{r})^2 \cdot \mathcal{N}(\boldsymbol{p}\,; \hat{\boldsymbol{p}}, \boldsymbol{\Sigma}^{(p)})\, \mathrm{d}\boldsymbol{c}\, \mathrm{d}\phi \tag{2.17}$$

of the Fourier series r.

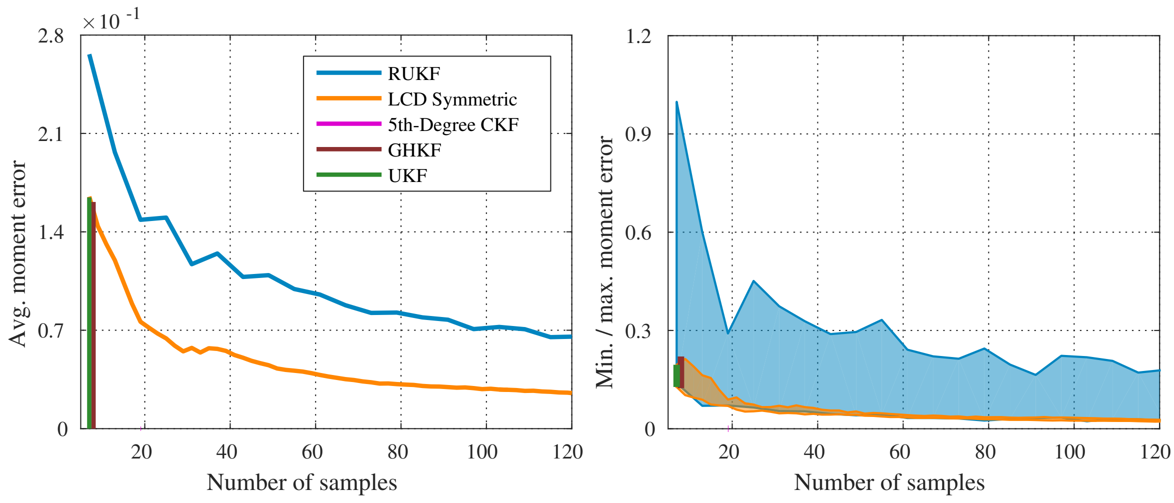
We perform $R = 100$ Monte Carlo runs. In each run, we randomly generate a mean $\hat{\boldsymbol{p}}$ and covariance $\boldsymbol{\Sigma}^{(p)}$ to compute the transformations (2.16) and (2.17). For the mean $\hat{\boldsymbol{p}}$, we randomly draw a vector from the distribution $\mathcal{N}(\boldsymbol{0}, 3\,\mathbf{I}_8)$. The covariance matrix $\boldsymbol{\Sigma}^{(p)}$ is constructed with an "inverse" eigendecomposition according to

$$\boldsymbol{\Sigma}^{(p)} = \mathbf{M}\, \mathrm{diag}(d_1, \ldots, d_8) \mathbf{M}^\top \quad ,$$

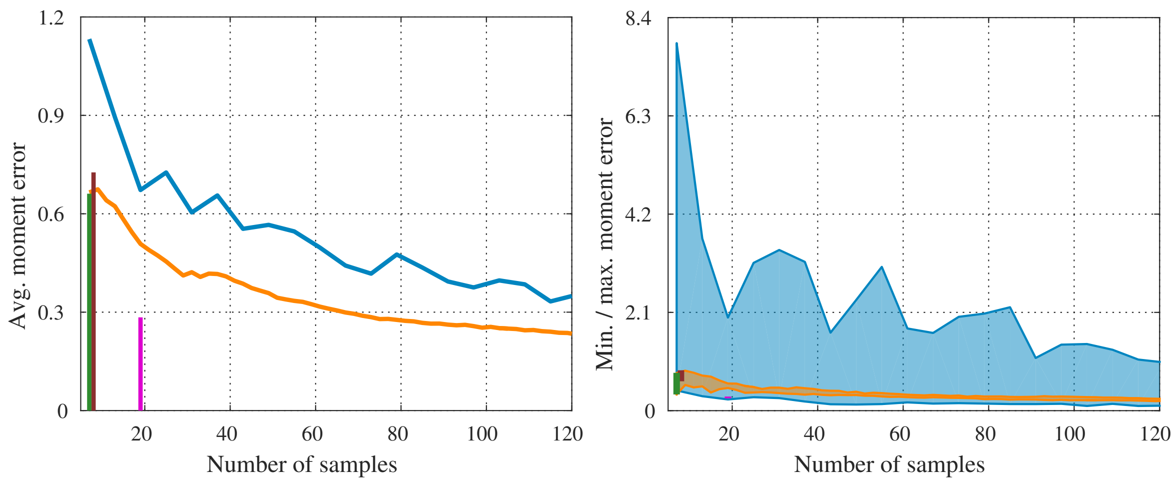
where $\mathbf{M} \in \mathbb{R}^{8\times 8}$ is a random orthogonal matrix and d_i is drawn from $\mathcal{U}(0, 10)$. For each run, the ground truth for $\hat{r}$ and $\Sigma^{(r)}$ is computed by means of 10^7 random samples.

We evaluate various Gaussian sampling techniques:

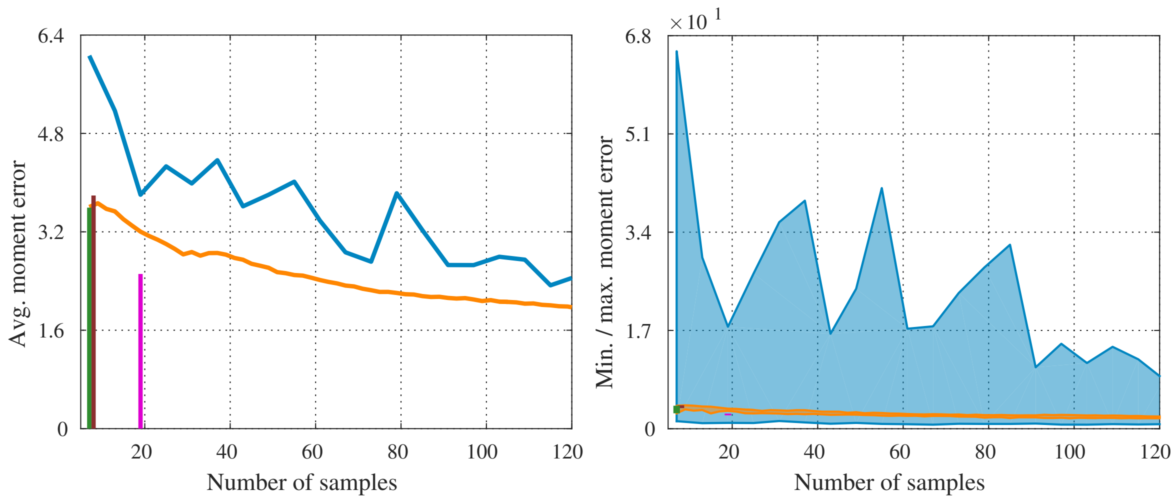
- the point-symmetric LCD-based sampling with 129 samples,
- the asymmetric LCD-based sampling with 129 samples,
- the 5th-degree CKF that uses $2 \cdot 8^2 + 1 = 129$ samples,
- the RUKF with 8 iterations resulting in $8 \cdot 2 \cdot 8 + 1 = 129$ samples,
- the GHKF with 2 quadrature points resulting in $2^8 = 256$ samples, and
- the UKF that uses $2 \cdot 8 + 1 = 17$ samples.



(a) Errors for 4th moment.

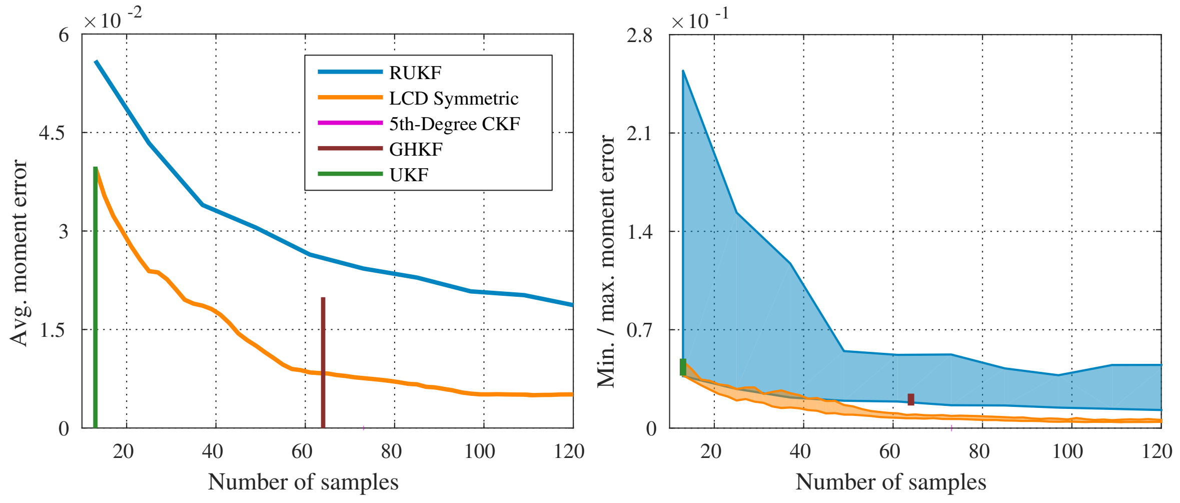


(b) Errors for 6th moment.

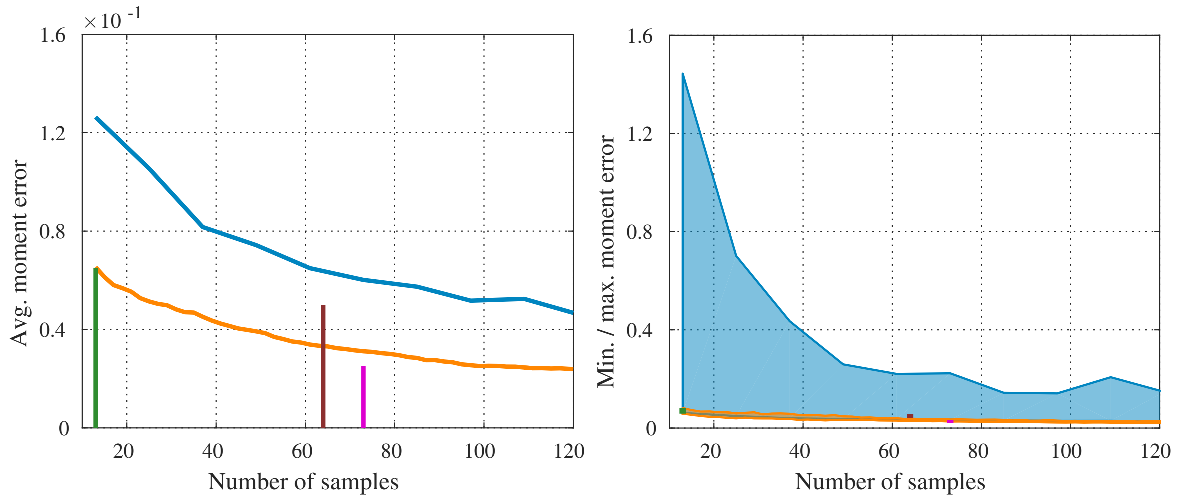


(c) Errors for 8th moment.

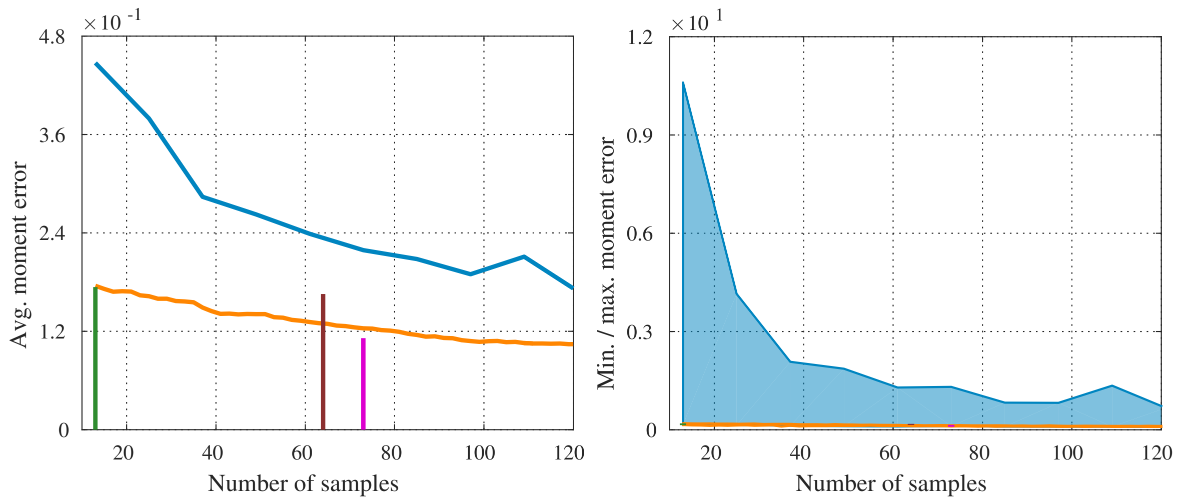
Figure 2.7: Average, minimum, and maximum normalized moment errors for a three-dimensional standard normal distribution (cf. [175]).



(a) Errors for 4th moment.



(b) Errors for 6th moment.



(c) Errors for 8th moment.

Figure 2.8: Average, minimum, and maximum normalized moment errors for a six-dimensional standard normal distribution (cf. [175]).

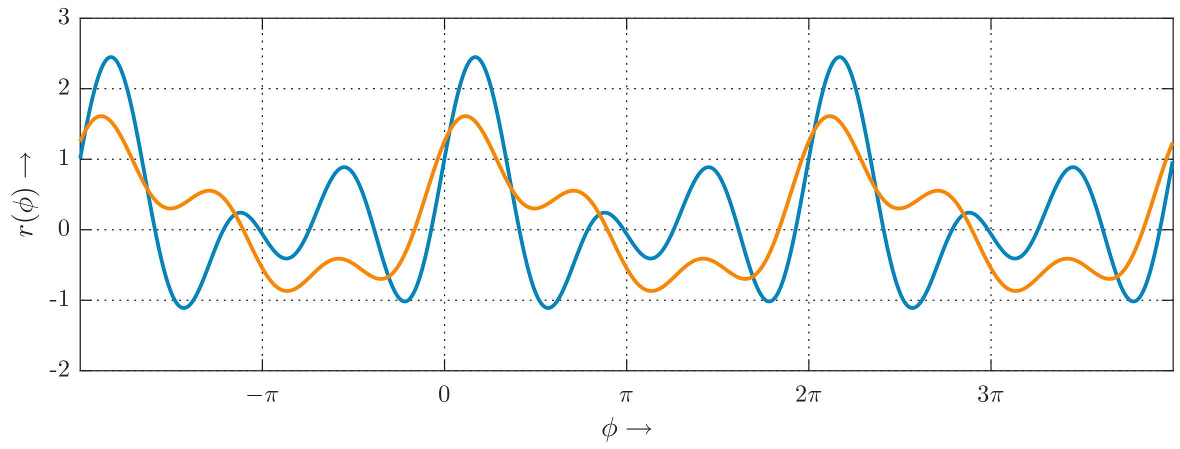


Figure 2.9: Two exemplary Fourier series with different coefficients $\boldsymbol{c} \in \mathbb{R}^7$. The 2π-periodicity can be clearly seen.

We compute the root mean square error (RMSE) of the mean $\hat{r}$ according to

$$\sqrt{\frac{1}{R}\sum_{i=1}^{R}\left(\hat{r}_i^{\mathrm{t}} - \hat{r}_i^{\mathrm{s}}\right)^2}$$

and the RMSE of the variance $\Sigma^{(r)}$ according to

$$\sqrt{\frac{1}{R}\sum_{i=1}^{R}\left(\left(\Sigma^{(r)}\right)_i^{\mathrm{t}} - \left(\Sigma^{(r)}\right)_i^{\mathrm{s}}\right)^2} \ ,$$

where the superscripts “t” and “s” denote the true value and the value obtained by the investigated sampling technique, respectively.

The results are shown in Figure 2.10. On the one hand, we can see that the errors of the UKF sampling are the worst for both mean and variance, and that the variance error is much larger than the errors of all other sampling techniques. Further, although the GHKF sampling uses the most number of samples in this evaluation, it is only slightly better than the RUKF sampling that only uses half the number of samples. On the other hand, from all sampling methods that use 129 samples (the first four methods), the LCD-based approaches offer the best performance. Regarding the mean, the point-symmetric LCD-based sampling is slightly better than the asymmetric one. For the variance, however, the point-symmetric LCD-based sampling has a much smaller error compared the asymmetric LCD-based sampling and all other sampling techniques. Finally, the 5th-degree CKF sampling yields the worst results out of these four methods.

2.4 Conclusions

In this chapter, we were concerned with the sampling of multivariate Gaussian distributions, i.e., the approximation of Gaussian PDFs with a carefully chosen set of weighted samples. Those samples can then be used to approximate moments of nonlinear transformed Gaussian distributed random vectors. Besides the rather naive and computationally expensive Monte Carlo integration, the overwhelming amount of state-of-the-art sampling techniques have their origin in Kalman filtering, i.e., the sampling approaches used by the UKF, GHKF, 5th-degree CKF, and RUKF. Another promising approach is to turn the Gaussian PDF approximation problem into an optimization problem. This approach relies on a

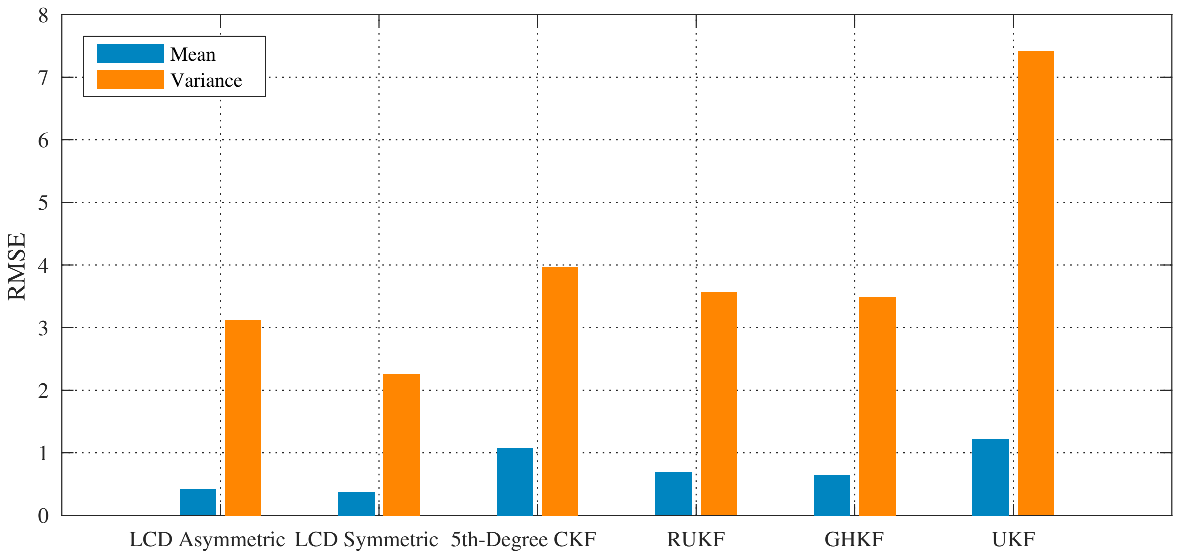


Figure 2.10: Results of the Fourier series evaluation.

distance measure between a Dirac mixture and a Gaussian PDF and is based on the localized cumulative distribution. The parameters of the Dirac mixture to be determined, i.e., the sample positions, are then obtained by minimizing this distance measure. A downside of this approach, however, is that the point symmetry of the standard normal distribution is not considered.

Thus, we proposed an improved LCD-based sampling technique that allows to approximate multivariate standard normal distributions with an arbitrary number of equally weighted and optimally placed point-symmetric samples. For its derivation, we introduced point-symmetric Dirac mixtures and adapted the LCD-based distance measure to work with these. In doing so, we caught up to state-of-the-art Gaussian sampling techniques as all of them rely on point-symmetric samples. Compared to the original asymmetric LCD-based sampling, the proposed point-symmetric variant captures all odd moments of a standard normal distribution exactly. Additionally, we changed the involved weighting function such that the new distance measure becomes numerically stable. This enables the approximation of large random vectors, e.g., $N > 200$, which can arise, for example, in the context of extended object tracking in order to process many measurements per time step as we will see in Chapter 3.

We compared the performance of the proposed sampling technique against state-of-the-art approaches by first computing higher-order moments of multivariate standard normal distributions. The results showed that the point-symmetric LCD-based sampling exactly captures the odd moments, while it nearly keeps the approximation quality of the original asymmetric variant for the even moments. Moreover, the moment approximation quality only slightly varies between different computed sample sets. Compared to other state-of-the-art approaches, our proposed sampling scheme delivered much better results than the UKF, RUKF, and GHKF. The 5$^{\text{th}}$-degree CKF is the only sampling scheme that captures the 4$^{\text{th}}$ moments exactly. However, its quadratically increase of the number of samples can make it intractable for large random vectors and its negative sample weights can lead to numerical issues. Second, we computed moments of a Fourier series that was given by random coefficients and random input angles. Here, the point-symmetric LCD-based sampling outperforms all state-of-the-art Gaussian sampling schemes.

In Chapter 3, we will use the proposed point-symmetric LCD-based sampling scheme to introduce a new sample-based Kalman filter, the smart sampling Kalman filter. We also use the sampling technique in Chapter 5 to improve the quality of a sophisticated nonlinear estimator, the progressive Gaussian filter. Moreover, due to the possible arbitrary number of optimally placed samples, the point-symmetric

LCD-based sampling can also be used, for example, to easily split a Gaussian to a Gaussian mixture. However, note that the new sampling scheme cannot be simply extended to the Gaussian mixture case like it was done for the asymmetric variant in [177], as a Gaussian mixture is not point-symmetric in general.

Chapter 3

The Smart Sampling Kalman Filter (S^{2}KF)

In the previous chapter, we dealt with the optimal sampling of Gaussian distributions. With this, we laid the groundwork for the central topic of this thesis: the state estimation of stochastic nonlinear dynamic systems. In this and the following chapters, we will extensively use the proposed point-symmetric LCD-based sampling in the context of nonlinear state estimation, formally described as follows.

We consider estimating the hidden state $\boldsymbol{x}_k \in \mathbb{R}^N$ of a discrete-time stochastic nonlinear dynamic system at a discrete time step k. How the system will evolve over time is described by a (potentially time-varying) system model

$$\boldsymbol{x}_k = \boldsymbol{a}_k(\boldsymbol{x}_{k-1}, \boldsymbol{w}_k) \ , \tag{3.1}$$

where $\boldsymbol{w}_k \in \mathbb{R}^{W_k}$ denotes white system noise with Gaussian PDF $f_k^{(w)}(\boldsymbol{w}_k) = \mathcal{N}(\boldsymbol{w}_k\,; \hat{\boldsymbol{w}}_k, \mathbf{Q}_k)$ that is assumed to be independent of the system state and accounts for unavoidable modeling errors and simplifications of the real world. Solely estimating the hidden system state by means of a system model (3.1) would not work out in many cases as the uncertainty of the state estimate would steadily grow over time due to the characteristics of the employed system model. This in turn would prevent a meaningful estimate of the system state at an arbitrary point in time. Thus, it is inevitable to gain information about the system state at time step k in the form of a received noisy measurement $\tilde{\boldsymbol{y}}_k$. In order to incorporate such a measurement into a given estimate, it has to be related to the hidden system state by means of a measurement model

$$\boldsymbol{y}_k = \boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{v}_k) \ , \tag{3.2}$$

where $\boldsymbol{v}_k \in \mathbb{R}^{V_k}$ denotes white measurement noise with Gaussian PDF $f_k^{(v)}(\boldsymbol{v}_k) = \mathcal{N}(\boldsymbol{v}_k\,; \hat{\boldsymbol{v}}_k, \mathbf{R}_k)$ that is assumed to be independent of the system state and accounts for sensor noise and modeling errors. It is important to note that a received measurement $\tilde{\boldsymbol{y}}_k$ is a realization of the random vector $\boldsymbol{y}_k$.

The state estimate at time step k that is based on all measurements received so far is given by the conditional PDF

$$f_{k|k}(\boldsymbol{x}_k) := f(\boldsymbol{x}_k \,|\, \tilde{\boldsymbol{y}}_k, \ldots, \tilde{\boldsymbol{y}}_1) \ ,$$

whereas the conditional PDF

$$f_{k|k-1}(\boldsymbol{x}_k) := f(\boldsymbol{x}_k \,|\, \tilde{\boldsymbol{y}}_{k-1}, \ldots, \tilde{\boldsymbol{y}}_1)$$

denotes the state estimate at time step k that does not have incorporated the newest measurement $\tilde{\boldsymbol{y}}_k$ yet. Usually, state estimation is performed in a recursive manner, consisting of two alternating steps. First, the prediction step or time update uses the system model (3.1) to predict the estimate $f_{k-1|k-1}(\boldsymbol{x}_{k-1})$ from the last time step $k-1$ to the current time step k with the aid of the Chapman–Kolmogorov equation resulting in the PDF $f_{k|k-1}(\boldsymbol{x}_k)$. Second, the filter step or measurement update uses the measurement model (3.2) in combination with the measurement $\tilde{\boldsymbol{y}}_k$ to correct the predicted estimate $f_{k|k-1}(\boldsymbol{x}_k)$ by means of Bayesian inference resulting in the filtered PDF $f_{k|k}(\boldsymbol{x}_k)$.

In literature, special attention is paid to the measurement update as this is the more demanding part of recursive state estimation. According to Bayes' rule, the filtered estimate is given by

$$\begin{aligned} f_{k|k}(\boldsymbol{x}_k) &= \frac{f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) f_{k|k-1}(\boldsymbol{x}_k)}{\int_{\mathbb{R}^N} f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) f_{k|k-1}(\boldsymbol{x}_k)\, \mathrm{d}\boldsymbol{x}_k} \\ &= c_k \; f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) f_{k|k-1}(\boldsymbol{x}_k) \ , \end{aligned} \tag{3.3}$$

where $f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k)$ is the so-called likelihood function or likelihood for short. As the denominator in (3.3) is independent of the system state, it is merely a normalization constant c_k that forces $f_{k|k}(\boldsymbol{x}_k)$ to be a proper PDF. The likelihood can be derived from the measurement model (3.2) according to

$$f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) = \int_{\mathbb{R}^{V_k}} \delta(\tilde{\boldsymbol{y}}_k - \boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{v}_k)) f_k^{(v)}(\boldsymbol{v}_k)\, \mathrm{d}\boldsymbol{v}_k \ .$$

Note that the likelihood is in general not a proper PDF as it does not necessarily integrates to one. In the update context, $f_{k|k-1}(\boldsymbol{x}_k)$ is referred to as prior (PDF) and $f_{k|k}(\boldsymbol{x}_k)$ as posterior (PDF). Thus, in other words the posterior is proportional to the prior multiplied by the likelihood. Unfortunately, computing the Bayesian update (3.3) in closed form is not possible for arbitrary likelihood functions $f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k)$ and prior PDFs $f_{k|k-1}(\boldsymbol{x}_k)$. Only for rare special cases, such as a linear measurement model corrupted by additive Gaussian noise and a Gaussian prior, the posterior can be obtained analytically. Hence, most times we have to be content with approximative solutions.

State-of-the-art state estimation techniques mainly differ in the way they approximate the Bayesian update and can be roughly divided into two classes. The first class directly approximates the product of likelihood and prior PDF, where the prior is mostly a Gaussian, a Gaussian mixture, or a Dirac mixture. The second class also assumes a Gaussian prior but approximates the (nonlinear) measurement model (3.2) as a linear one. In doing so, the Bayesian update can be solved analytically, leading to the well-known Kalman filter. Both approaches have their pros and cons, which will be discussed and examined in the course of this thesis. While the first approach is pursued in Chapter 5, the latter approach is taken in this chapter and will be further extended to the case of distributed state estimation in Chapter 4. Specifically, we utilize the point-symmetric LCD-based sampling for Gaussian distributions from Chapter 2 to formulate a new sample-based Kalman filter applicable to nonlinear systems. Furthermore, we use the new estimator for extended object tracking and derive a comprehensive measurement model for estimating pose and a priori unknown extents of a cylinder in 3D.

This chapter is based on the publications [175, 178, 179].

3.1 The Kalman Filter Applied to Nonlinear State Estimation

In 1960, Rudolf E. Kálmán proposed a filter to optimally estimate the state of a linear system corrupted by additive noise [15], known as the Kalman filter. Nevertheless, the Kalman filter was quickly applied to nonlinear systems, e.g., the trajectory estimation in the Apollo project [1], and is still a widely used estimator for nonlinear systems. In the following, we briefly recapitulate how the Kalman filter is used to estimate the state of a nonlinear system. First, prior PDF and posterior PDF are always assumed to be Gaussian, that is,

$$f_{k|k}(\boldsymbol{x}_k) = \mathcal{N}(\boldsymbol{x}_k\,; \hat{\boldsymbol{x}}_{k|k}, \mathbf{P}_{k|k})$$

and

$$f_{k|k-1}(\boldsymbol{x}_k) = \mathcal{N}(\boldsymbol{x}_k\,;\hat{\boldsymbol{x}}_{k|k-1}, \mathbf{P}_{k|k-1}) \ .$$

Second, starting with an initial state estimate

$$f_{0|0}(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0\,;\hat{\boldsymbol{x}}_{0|0}, \mathbf{P}_{0|0}) \ ,$$

sometimes referred to as "initial guess", we can perform alternating time updates and measurement updates.

3.1.1 Time Update

Given a state estimate $f_{k-1|k-1}(\boldsymbol{x}_{k-1})$ at time step $k-1$, the predicted state estimate $f_{k|k-1}(\boldsymbol{x}_k)$ can be obtained by using the Chapman–Kolmogorov equation [11, Sec. 10.2] according to

$$\begin{aligned} f_{k|k-1}(\boldsymbol{x}_k) &= \int_{\mathbb{R}^N} f_k(\boldsymbol{x}_k \,|\, \boldsymbol{x}_{k-1}) f_{k-1|k-1}(\boldsymbol{x}_{k-1}) \,\mathrm{d}\boldsymbol{x}_{k-1} \\ &= \int_{\mathbb{R}^{W_k}} \int_{\mathbb{R}^N} \delta(\boldsymbol{x}_k - \boldsymbol{a}_k(\boldsymbol{x}_{k-1}, \boldsymbol{w}_k)) f_{k-1|k-1}(\boldsymbol{x}_{k-1}) f_k^{(w)}(\boldsymbol{w}_k) \,\mathrm{d}\boldsymbol{x}_{k-1} \,\mathrm{d}\boldsymbol{w}_k \ . \end{aligned} \tag{3.4}$$

Here, $f_k(\boldsymbol{x}_k \,|\, \boldsymbol{x}_{k-1})$ is called the transition density. The predicted PDF, however, generally is not Gaussian. Hence, we approximate the predicted PDF as a Gaussian by means of moment matching, i.e., we compute mean and covariance matrix of this PDF. With the sifting property of the Dirac-δ distribution, we immediately get the predicted state mean according to

$$\hat{\boldsymbol{x}}_{k|k-1} = \int_{\mathbb{R}^{W_k}} \int_{\mathbb{R}^N} \boldsymbol{a}_k(\boldsymbol{x}_{k-1}, \boldsymbol{w}_k) f_{k-1|k-1}(\boldsymbol{x}_{k-1}) f_k^{(w)}(\boldsymbol{w}_k) \,\mathrm{d}\boldsymbol{x}_{k-1} \,\mathrm{d}\boldsymbol{w}_k \tag{3.5}$$

and the predicted state covariance matrix according to

$$\begin{aligned} \mathbf{P}_{k|k-1} = \int_{\mathbb{R}^{W_k}} \int_{\mathbb{R}^N} &(\boldsymbol{a}_k(\boldsymbol{x}_{k-1}, \boldsymbol{w}_k) - \hat{\boldsymbol{x}}_{k|k-1})(\boldsymbol{a}_k(\boldsymbol{x}_{k-1}, \boldsymbol{w}_k) - \hat{\boldsymbol{x}}_{k|k-1})^\top \cdot \\ &f_{k-1|k-1}(\boldsymbol{x}_{k-1}) f_k^{(w)}(\boldsymbol{w}_k) \,\mathrm{d}\boldsymbol{x}_{k-1} \,\mathrm{d}\boldsymbol{w}_k \ . \end{aligned} \tag{3.6}$$

That is, we never have to deal with the exact (and unknown) predicted PDF of (3.4).

3.1.2 Measurement Update

As previously discussed, the general Bayesian update is given by (3.3). We can rewrite this using the joint density

$$f_k^{(x,y)}(\boldsymbol{x}_k, \boldsymbol{y}_k \,|\, \tilde{\boldsymbol{y}}_{k-1}, \ldots, \tilde{\boldsymbol{y}}_1)$$

of state and measurement at time step k given the past measurements $\tilde{\boldsymbol{y}}_{k-1}, \ldots, \tilde{\boldsymbol{y}}_1$ according to

$$f_{k|k}(\boldsymbol{x}_k) = c_k \; f_k^{(x,y)}(\boldsymbol{x}_k, \tilde{\boldsymbol{y}}_k \,|\, \tilde{\boldsymbol{y}}_{k-1}, \ldots, \tilde{\boldsymbol{y}}_1) \ .$$

In other words, the measurement $\tilde{\boldsymbol{y}}_k$ determines where to slice the joint density in order to obtain the posterior density. Now, by approximating the joint density as a Gaussian distribution

$$f_k^{(x,y)}(\boldsymbol{x}_k, \boldsymbol{y}_k \,|\, \tilde{\boldsymbol{y}}_{k-1}, \ldots, \tilde{\boldsymbol{y}}_1) \approx \mathcal{N}\left(\begin{bmatrix}\boldsymbol{x}_k \\ \boldsymbol{y}_k\end{bmatrix}; \begin{bmatrix}\hat{\boldsymbol{x}}_{k|k-1} \\ \hat{\boldsymbol{y}}_k\end{bmatrix}, \begin{bmatrix}\mathbf{P}_{k|k-1} & \mathbf{C}_k \\ \mathbf{C}_k^\top & \mathbf{Y}_k\end{bmatrix}\right) \ , \tag{3.7}$$

the posterior PDF also becomes a Gaussian

$$\begin{aligned} f_{k|k}(\boldsymbol{x}_k) &\approx c_k \, \mathcal{N}\left(\begin{bmatrix}\boldsymbol{x}_k \\ \tilde{\boldsymbol{y}}_k\end{bmatrix}; \begin{bmatrix}\hat{\boldsymbol{x}}_{k|k-1} \\ \hat{\boldsymbol{y}}_k\end{bmatrix}, \begin{bmatrix}\mathbf{P}_{k|k-1} & \mathbf{C}_k \\ \mathbf{C}_k^\top & \mathbf{Y}_k\end{bmatrix}\right) \\ &= \mathcal{N}(\boldsymbol{x}_k\,;\hat{\boldsymbol{x}}_{k|k}, \mathbf{P}_{k|k}) \ , \end{aligned}$$

with posterior state mean

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \mathbf{C}_k \mathbf{Y}_k^{-1} (\tilde{\boldsymbol{y}}_k - \hat{\boldsymbol{y}}_k) \tag{3.8}$$

and posterior state covariance matrix

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{C}_k \mathbf{Y}_k^{-1} \mathbf{C}_k^\top \ , \tag{3.9}$$

whose calculation merely requires the measurement mean

$$\hat{\boldsymbol{y}}_k = \int_{\mathbb{R}^{V_k}} \int_{\mathbb{R}^N} \boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{v}_k) f_{k|k-1}(\boldsymbol{x}_k) f_k^{(v)}(\boldsymbol{v}_k) \,\mathrm{d}\boldsymbol{x}_k \,\mathrm{d}\boldsymbol{v}_k \ , \tag{3.10}$$

the measurement covariance matrix

$$\begin{aligned} \mathbf{Y}_k = \int_{\mathbb{R}^{V_k}} \int_{\mathbb{R}^N} &(\boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{v}_k) - \hat{\boldsymbol{y}}_k)(\boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{v}_k) - \hat{\boldsymbol{y}}_k)^\top \cdot \\ &f_{k|k-1}(\boldsymbol{x}_k) f_k^{(v)}(\boldsymbol{v}_k) \,\mathrm{d}\boldsymbol{x}_k \,\mathrm{d}\boldsymbol{v}_k \ , \end{aligned} \tag{3.11}$$

and the state–measurement cross-covariance matrix

$$\begin{aligned} \mathbf{C}_k = \int_{\mathbb{R}^{V_k}} \int_{\mathbb{R}^N} &(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1})(\boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{v}_k) - \hat{\boldsymbol{y}}_k)^\top \cdot \\ &f_{k|k-1}(\boldsymbol{x}_k) f_k^{(v)}(\boldsymbol{v}_k) \,\mathrm{d}\boldsymbol{x}_k \,\mathrm{d}\boldsymbol{v}_k \ . \end{aligned} \tag{3.12}$$

Equations (3.8) and (3.9) are the well-known Kalman filter formulas, and the product $\mathbf{K}_k = \mathbf{C}_k \mathbf{Y}_k^{-1}$ is also referred to as the Kalman gain. It is important to note that the Gaussian approximation (3.7) implicitly linearizes the actual nonlinear relationship between state and measurement, which might be a very rough approximation. If the models are already linear and are corrupted by additive Gaussian noise, then (3.7) is no approximation and (3.8) and (3.9) yield the minimum mean square error (MMSE) estimator [17, Sec. 5.2]. Otherwise, it is the linear minimum mean square error (LMMSE) estimator. For a nonlinear measurement model corrupted by additive noise, an analysis of how the Gaussian approximation (3.7) can affect the performance of the estimator can be found in [117].

In summary, the Kalman filter boils down to the computation of the integrals (3.5) and (3.6) for the time update, and the integrals (3.10), (3.11), and (3.12) for the measurement update.

3.2 Related Work

Unfortunately, the all these five moments cannot be computed analytically for arbitrary functions $\boldsymbol{a}_k$ and $\boldsymbol{h}_k$. Only for rare special cases, e.g., linear functions or polynomials, closed-form solutions are possible. A survey on such analytic moment computation can be found in [118]. In all other cases, the moments have to be approximated in order to apply the Kalman filter to nonlinear systems.

State-of-the-art Kalman filters can be divided into two classes, depending on how these moments are approximated. One class of Kalman filters approximates the system model (3.1) and the measurement model (3.2) such that it is possible to solve the integrals in closed form. The other class of Kalman filters does not touch the involved models but instead directly approximates the integrals. In the following, we briefly discuss both classes of Kalman filters and subsequently give an overview of common Kalman filter extensions, which improve different aspects in Kalman filtering. A survey on different state-of-the-art Kalman filters can also be found, for example, in [104].

3.2.1 Approximations of the Nonlinear Models

The most prominent Kalman filter that approximates the nonlinear models is the extended Kalman filter (EKF), e.g., [11, Sec. 10.3], [17, Sec. 13.2], [18]. The EKF approximates the models with a Taylor series expansion and evaluates the series around the current state mean, i.e., $\hat{\boldsymbol{x}}_{k-1|k-1}$ for the prediction and $\hat{\boldsymbol{x}}_{k|k-1}$ for the measurement update. Usually, the EKF refers to the first-order Taylor series expansion of the models. That is, it only uses the Jacobians of the models for approximation. If second-order terms of the Taylor series expansion are included as well, i.e., the Hessians of the models, we obtain the so-called second-order EKF, e.g., see [19,20]. A major drawback of the EKF is the fact that it does not take the current uncertainty of the state estimate into account for the linearization.

Instead of a Taylor series expansion, polynomial interpolations can be used for approximating the nonlinear models. In doing so, derivative-free and closed-form moment calculations are possible that require only multiple evaluations of the nonlinear models. Moreover, compared to the EKF, filters taking this approach consider the uncertainty of the state estimate for the linearization, which is also referred to as statistical linearization. Such an approach is taken by the first-order and second-order divided difference filter (DDF) [21] as well as by the first-order and second-order central difference filter (CDF) [22]. Although these filters evaluate the nonlinear models at the same points as the UKF does, the filter results are different due to their different ways of computing the desired moments [119]. In [120], the partitioned update Kalman filter (PUKF) is proposed that is based on the approximation of the second-order CDF. The idea of that filter is to reduce the linearization error in case of a vector-valued measurement by processing its parts sequentially, starting with the most linear part, instead of processing the entire measurement vector as a whole. Another option is to approximate the nonlinear models by means of Chebyshev polynomial series expansion which results in the Chebyshev polynomial Kalman filter (CPKF) [23]. The actual polynomial approximation is obtained by using discrete cosine transformations. However, the proposed approach only works for a one-dimensional state space.

3.2.2 Direct Approximations of the Integrals

In contrast to an approximation of the nonlinear models, we can directly approximate the integrals, which also results in a statistical linearization. For that purpose, we simply sample the involved Gaussian state and noise densities and use these samples to evaluate the unmodified models to obtain the desired moments. The naive way would be to separately approximate the state and noise densities with Dirac mixtures. However, this would result in a Cartesian product of their respective sample sets. Fortunately, we can do better. When looking at the integrals (3.5) and (3.6) for the prediction, it can be seen that we have to sample the product of two independent Gaussian distributions. Due to their independence, the product yields the joint density of state and noise, which is also Gaussian, i.e.,

$$f_{k-1|k-1}(\boldsymbol{x}_{k-1}) \cdot f_k^{(w)}(\boldsymbol{w}_k) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{x}_{k-1}\\ \boldsymbol{w}_k\end{bmatrix}; \begin{bmatrix}\hat{\boldsymbol{x}}_{k-1|k-1}\\ \hat{\boldsymbol{w}}_k\end{bmatrix}, \begin{bmatrix}\mathbf{P}_{k-1|k-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{Q}_k\end{bmatrix}\right) \ .$$

Consequently, by approximating this joint density with a Dirac mixture comprising P_k samples according to

$$\sum_{i=1}^{P_k} \omega_{k-1}^{(i)} \delta\left(\begin{bmatrix}\boldsymbol{x}_{k-1}\\ \boldsymbol{w}_k\end{bmatrix} - \begin{bmatrix}\boldsymbol{x}_{k-1}^{(i)}\\ \boldsymbol{w}_k^{(i)}\end{bmatrix}\right) \ , \tag{3.13}$$

with weights $\omega_{k-1}^{(i)}$ and sample positions $[(\boldsymbol{x}_{k-1}^{(i)})^\top, (\boldsymbol{w}_k^{(i)})^\top]^\top$, we immediately obtain approximations of the predicted state mean

$$\hat{\boldsymbol{x}}_{k|k-1} \approx \sum_{i=1}^{P_k} \omega_{k-1}^{(i)} \boldsymbol{a}_k(\boldsymbol{x}_{k-1}^{(i)}, \boldsymbol{w}_k^{(i)}) \tag{3.14}$$

and the predicted state covariance matrix

$$\mathbf{P}_{k|k-1} \approx \sum_{i=1}^{P_k} \omega_{k-1}^{(i)} \big(\boldsymbol{a}_k(\boldsymbol{x}_{k-1}^{(i)}, \boldsymbol{w}_k^{(i)}) - \hat{\boldsymbol{x}}_{k-1|k-1}\big)\big(\boldsymbol{a}_k(\boldsymbol{x}_{k-1}^{(i)}, \boldsymbol{w}_k^{(i)}) - \hat{\boldsymbol{x}}_{k-1|k-1}\big)^\top \ . \tag{3.15}$$

The same procedure can be done for the measurement update, where we have to consider the Gaussian joint density of predicted state and measurement noise

$$f_{k|k-1}(\boldsymbol{x}_k) \cdot f_k^{(v)}(\boldsymbol{v}_k) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{x}_k \\ \boldsymbol{v}_k\end{bmatrix}; \begin{bmatrix}\hat{\boldsymbol{x}}_{k|k-1} \\ \hat{\boldsymbol{v}}_k\end{bmatrix}, \begin{bmatrix}\mathbf{P}_{k|k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_k\end{bmatrix}\right) \ .$$

Approximating the joint density with U_k weighted samples according to

$$\sum_{i=1}^{U_k} \omega_k^{(i)} \delta\left(\begin{bmatrix}\boldsymbol{x}_k \\ \boldsymbol{v}_k\end{bmatrix} - \begin{bmatrix}\boldsymbol{x}_k^{(i)} \\ \boldsymbol{v}_k^{(i)}\end{bmatrix}\right) \tag{3.16}$$

directly gives approximative solutions for the measurement mean

$$\hat{\boldsymbol{y}}_k \approx \sum_{i=1}^{U_k} \omega_k^{(i)} \boldsymbol{h}_k(\boldsymbol{x}_k^{(i)}, \boldsymbol{v}_k^{(i)}) \ , \tag{3.17}$$

the measurement covariance matrix

$$\mathbf{Y}_k \approx \sum_{i=1}^{U_k} \omega_k^{(i)} \big(\boldsymbol{h}_k(\boldsymbol{x}_k^{(i)}, \boldsymbol{v}_k^{(i)}) - \hat{\boldsymbol{y}}_k\big)\big(\boldsymbol{h}_k(\boldsymbol{x}_k^{(i)}, \boldsymbol{v}_k^{(i)}) - \hat{\boldsymbol{y}}_k\big)^\top \ , \tag{3.18}$$

and the state–measurement cross-covariance matrix

$$\mathbf{C}_k \approx \sum_{i=1}^{U_k} \omega_k^{(i)} \big(\boldsymbol{x}_k^{(i)} - \hat{\boldsymbol{x}}_{k|k-1}\big)\big(\boldsymbol{h}_k(\boldsymbol{x}_k^{(i)}, \boldsymbol{v}_k^{(i)}) - \hat{\boldsymbol{y}}_k\big)^\top \ . \tag{3.19}$$

With these sample-based moment computations, we obtain the important class of sample-based Kalman filters, also called linear regression Kalman filters (LRKFs) [121], [4, App. A]. At this point, the Gaussian sampling techniques from Chapter 2 come into play, that is, we can simply use these sampling schemes to obtain the required Dirac mixture approximations (3.13) and (3.16). In fact, most of the proposed sampling techniques were originally developed in the context of sample-based Kalman filtering.

The most popular sample-based Kalman filter is the UKF [16,24,25]. Despite a higher-order variant that is only applicable to univariate system states [122], the number of samples cannot be further increased in order to improve the approximation quality of the moments. The only adjustable parameters are the sample set rotation and the scaling of the samples. However, finding an appropriate scaling parameter is not an intuitive task. Hence, different approaches are proposed to determine this parameter. For example, in [26], different predetermined scaling parameters are tested, but then only that scaling is used for the actual measurement update that best matches the given measurement. A scaling parameter optimization using Gaussian processes is proposed in [27]. This approach, however, introduces new parameters to be determined, i.e., the possible scaling values and parameters controlling the actual optimization. Another way is to adapt the scaling parameter online by means of a maximum a posteriori estimation [28]. That is, the scaling is used that maximizes the a posteriori probability. The aspect of sample set rotation is thoroughly discussed and analyzed in [29]. There, different ways of determining appropriate sample set rotations (offline and online) are proposed. Nevertheless, all these techniques still suffer from the problem that solely a fixed number of samples is propagated through the nonlinear models. Instead of finding appropriate parameters, time could also be spent on propagating more (carefully chosen) samples to improve the moment computations as it is done by the sample-based Kalman filter proposed later in this chapter. Finally, an exhaustive survey on various UKF variants is given in [30].

The originally proposed (3$^{\text{rd}}$-degree) CKF [31] is only a reformulating of the UKF without a sample placed in the state space origin. Nonetheless, the high-degree CKF [32] is capable of using (much) more samples. The problem is that with a higher degree the number of samples increases polynomial in the dimension N. To mitigate this issue, the same authors introduce an adaptive high-degree CKF [123]. It uses merely a 3$^{\text{rd}}$-degree spherical rule but radial rules of adaptive degree, which depend on the uncertainty of the prior state estimate. In doing so, (i) more than one sample can be placed on each of the principal axes, and (ii) the number of samples can vary between the axes. On the one hand, this results only in an linear increase of the number of samples in the dimension N. On the other hand, all samples are solely placed on the principal axes, that is, they do not fill the space between the principal axes. Note that the adaptive high-degree CKF is similar to the approach taken in [124], where an identical number of optimally placed samples is located on the principal axes.

As already discussed in Chapter 2, the GHKF [22] is based on the Gauss–Hermite quadrature in combination with the product formula to handle multivariate system states. However, the number of samples grows exponentially in the considered dimension N, making it intractable for large dimensions. In contrast, the user can control the number of samples in a linear fashion for the RUKF [33–37], which makes the filter well-suited for large dimensions. However, its samples are randomly rotated and randomly scaled. Thus, the RUKF achieves only an adequate state space coverage on average.

In order to cope with heavy-tailed noise, recently a sample-based Kalman filter for Student's t-distribution is proposed in [125]. This filter samples the Student's t-distribution instead of the closely related Gaussian distribution. Finally, there can be nonlinear models $\boldsymbol{a}_k$ or $\boldsymbol{h}_k$ for which closed-form moment computations are possible if certain state variables are assumed to be constant. The semi-analytic moment calculation approach [51–53] exploits this fact in order to reduce the state space to be sampled, which can reduce the computational effort and improve the quality of the moment approximations.

3.2.3 Kalman Filter Extensions

Kalman filters applied to nonlinear systems can be extended in many different ways in order to improve estimation quality or filter execution time, regardless of the taken linearization approach. The most popular Kalman filter extensions are briefly discussed in the following.

Iterative Kalman Filters

In order to find a more appropriate nominal state for the measurement model linearization in the EKF, an iterative version, called the iterated extended Kalman filter, e.g., [17, Sec. 13.3], can be used. However, like the standard EKF, the iterated EKF does not take into account the uncertainty of the current state estimate during the linearization. Recently, iterated versions for statistical linearization, e.g., done by sample-based Kalman filters, have been proposed as well [39–41]. In contrast to the iterated EKF, both state mean and state covariance matrix are adapted over the iterations to find a proper linearization of the nonlinear measurement model.

A Kalman filter that also uses the first-order approximation of the measurement model in an iterative manner is the recursive update filter [38]. It differs from the iterated EKF in the way that it performs the entire update gradually. In doing so, an "overshot" of the state mean, which might happen in the iterated EKF, can be avoided. In addition, the idea of the recursive update filter is extended to sample-based Kalman filters in [126]. That is, a recursive measurement update is performed where the measurement model linearization is conducted with the aid of samples instead of the first-order Taylor series expansion.

Square Root Kalman Filters

The idea of square root Kalman filters is to work solely with the square root of the state covariance matrix, see e.g., [11, Sec. 7.4], [17, Sec. 6.3]. In doing so, the precision of the state covariance matrix is doubled. In particular, this is important if variances of the state entries differ by several orders of magnitude. Originally, a square root filter was developed for the (extended) Kalman filter [1]. Subsequently, a square root federated Kalman filter for distributed state estimation was proposed [74], followed by square root variants of sample-based Kalman filters using the QR decomposition, such as the square root UKF [42], and square root information filters based on the UKF, the CDF, or the CKF [127–130].

Gaussian Mixture Kalman Filters

Due to the Gaussian state estimate, the Kalman filter is only a unimodal estimator. In the linear case this poses no problem. For the nonlinear case, however, the intractable true state density can become arbitrary complex and, in particular, multimodel. In order to allow for multimodel estimates, extensions to Gaussian mixture filters have been suggested, e.g., [22, 43]. There, the state density is given by a Gaussian mixture, i.e., a weighted sum of Gaussian distributions, instead of a single Gaussian. In fact, each of these Gaussian distributions is predicted and updated by means of individual Kalman filters. Thus, a Gaussian mixture KF is simply a bank of weighted Kalman filters.

Gaussian mixture estimators can be implemented with any type of Kalman filter, e.g., with square root UKFs [44], UKFs with adaptive scaling parameters [131], or with semi-analytic statistical linearization [52]. Moreover, besides using a fixed number of components, the number of components can be determined in an adaptive way online, e.g., see [45, 132]. Also the component weights can be adapted over time in different ways. For example, the weights can be adapted during a measurement update by evaluating the measurement in the measurement distribution $\mathcal{N}(\boldsymbol{y}_k\,; \hat{\boldsymbol{y}}_k, \mathbf{Y}_k)$ [22] or during the time update by solving a convex optimization problem [133].

Constrained Kalman Filters

A Kalman filter can only estimate unconstrained quantities. Nevertheless, several approaches exist to incorporate different type of constraints into Kalman filtering, e.g., linear equality constraints (perfect measurements) and linear inequality constraints (projection techniques, PDF truncation) [17, Sec. 7.5], nonlinear equality constraints [46], and nonlinear inequality constraints [47, 48]. A survey on different constraint types is also given in [49].

Other Extensions

Another extension for Kalman filters is the concept of “linear in transform” estimation [134, 135]. The idea is to improve the estimation performance by reducing linearization errors. This reduction is performed by properly transforming measurement model and measurement such that the relationship between the system state and the transformed measurement becomes less nonlinear, and consequently better fulfills the linearity assumption of the Kalman filter. However, finding a suitable transformation is highly problem-dependent and in [135] only design principles for such a transformation are provided.

A further extension for Kalman filters is called state decomposition [4, App. E], [50]. It is applicable if a measurement model solely depends on a subspace of the system state. In such a case, the estimate of the independent parts of the system can be updated analytically given an updated estimate for the measurement-dependent state variables. Updating the entire state estimate in this way improves the overall estimation quality of the filter and can also reduce its execution time. We take up the idea of state decomposition in Chapter 5 and apply it to a novel nonlinear state estimator.

3.3 The Smart Sampling Kalman Filter

After discussing state-of-the-art Kalman filters, we will now introduce a new sample-based Kalman filter. Developing a sample-based Kalman filter essentially means to find suitable Dirac mixtures (3.13) and (3.16) for prediction and measurement update, respectively. Here, we propose to use the new point-symmetric LCD-based Gaussian sampling scheme from Chapter 2 to conduct both the time update and the measurement update.

For the time update, we compute a set of equally weighted point-symmetric samples $\{\boldsymbol{p}^{(i)}\}_{i=1}^{P_k}$ with Algorithm 2.1 that approximates a standard normal distribution of dimension $N + W_k$, i.e., $\boldsymbol{p}^{(i)} \in \mathbb{R}^{N+W_K}$. In combination with the Mahalanobis transformation, we obtain the sample positions

$$\begin{bmatrix} \boldsymbol{x}_{k-1}^{(i)} \\ \boldsymbol{w}_k^{(i)} \end{bmatrix} = \begin{bmatrix} \operatorname{chol}(\mathbf{P}_{k-1|k-1}) & \mathbf{0} \\ \mathbf{0} & \operatorname{chol}(\mathbf{Q}_k) \end{bmatrix} \cdot \boldsymbol{p}^{(i)} + \begin{bmatrix} \hat{\boldsymbol{x}}_{k-1|k-1} \\ \hat{\boldsymbol{w}}_k \end{bmatrix} \quad , \quad \forall i \in \{1, \ldots, P_k\} \ , \tag{3.20}$$

with corresponding sample weights $\omega_{k-1}^{(i)} = \frac{1}{P_k}$. Based on these transformed samples, we can perform the sample-based prediction by computing the moments (3.14) and (3.15). The same procedure is done to obtain the samples for the measurement update. That is, with Algorithm 2.1 we compute the samples $\{\boldsymbol{u}^{(i)}\}_{i=1}^{U_k}$ where $\boldsymbol{u}^{(i)} \in \mathbb{R}^{N+V_K}$ and transform them according to

$$\begin{bmatrix} \boldsymbol{x}_{k}^{(i)} \\ \boldsymbol{v}_k^{(i)} \end{bmatrix} = \begin{bmatrix} \operatorname{chol}(\mathbf{P}_{k|k-1}) & \mathbf{0} \\ \mathbf{0} & \operatorname{chol}(\mathbf{R}_k) \end{bmatrix} \cdot \boldsymbol{u}^{(i)} + \begin{bmatrix} \hat{\boldsymbol{x}}_{k|k-1} \\ \hat{\boldsymbol{v}}_k \end{bmatrix} \quad , \quad \forall i \in \{1, \ldots, U_k\} \ , \tag{3.21}$$

with equal sample weights $\omega_k^{(i)} = \frac{1}{U_k}$. Based on these transformed samples, we compute the moments (3.17), (3.18), and (3.19) in order to conduct the measurement update of the Kalman filter. By performing time update and measurement update in this way, we introduce the smart sampling Kalman filter (S^{2}KF).

In order to avoid the, potentially time-consuming, re-computation of the samples $\{\boldsymbol{p}^{(i)}\}_{i=1}^{P_k}$ and $\{\boldsymbol{u}^{(i)}\}_{i=1}^{U_k}$ on every program start, we additionally introduce a sample cache: a persistent storage of computed samples in the filesystem. The idea behind this cache is to check online if a required combination of sample dimension and number of samples was already computed in the past. If so, the samples can be simply retrieved from the cache. If not, the samples are computed with Algorithm 2.1 and subsequently stored in the sample cache for later retrieval. This mechanism is transparent for the user and eases the usage of the S^{2}KF. Of course, if the S^{2}KF is used for a dedicated estimation problem or in a special, e.g., embedded, system, all necessary sample sets should be computed in advanced and stored in a read-only memory.

The entire procedure of the S^{2}KF is given in Algorithm 3.1. The S^{2}KF has several advantages over existing (sample-based) Kalman filters. First, it is capable of using an *arbitrary* number of samples for time update and measurement update. That is, the number of samples does not depend on the dimension of state and noise. This is different from state-of-the-art sample-based Kalman filters such as the GHKF, the 5th-degree CKF, or the RUKF. Second, due to same sample weight assigned to all samples, the computation of sample means and sample covariance matrices can be simplified and sped up. By factoring the sample weights out of the sum in the moment computation, we only have to multiply the unweighted sum once with the sample weight. In doing so, much less multiplications are required. Third, the strictly positive sample weights avoid indefinite or negative covariance matrices,

```
Input: system models $\boldsymbol{a}_k(\boldsymbol{x}_{k-1}, \boldsymbol{w}_k)$, system noise moments $\hat{\boldsymbol{w}}_k$ and $\mathbf{Q}_k$,
       measurement models $\boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{v}_k)$, measurement noise moments $\hat{\boldsymbol{v}}_k$ and $\mathbf{R}_k$,
       number of prediction samples $P_k$, number of update samples $U_k$, and
       initial state estimate $\hat{\boldsymbol{x}}_{0|0}$ and $\mathbf{P}_{0|0}$

 1: for $k = 1, 2, \dots$
        // Get $P_k$ equally weighted samples approximating $\mathcal{N}(\hat{\boldsymbol{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1})$
 2:     if $P_k$ samples for dimension $N + W_k$ are stored in sample cache then
 3:         Retrieve samples $\{\boldsymbol{p}^{(i)}\}_{i=1}^{P_k}$ from cache
 4:     else
 5:         Compute samples $\{\boldsymbol{p}^{(i)}\}_{i=1}^{P_k}$ using Algorithm 2.1
 6:         Store samples $\{\boldsymbol{p}^{(i)}\}_{i=1}^{P_k}$ in cache
 7:     end if
 8:     Compute sample positions (3.20)
        // Perform time update
 9:     Compute $\hat{\boldsymbol{x}}_{k|k-1}$ and $\mathbf{P}_{k|k-1}$ using (3.14) and (3.15) with $\omega_{k-1}^{(i)} = \frac{1}{P_k}$
        // Get $U_k$ equally weighted samples approximating $\mathcal{N}(\hat{\boldsymbol{x}}_{k|k-1}, \mathbf{P}_{k|k-1})$
10:     if $U_k$ samples for dimension $N + V_k$ are stored in sample cache then
11:         Retrieve samples $\{\boldsymbol{u}^{(i)}\}_{i=1}^{U_k}$ from cache
12:     else
13:         Compute samples $\{\boldsymbol{u}^{(i)}\}_{i=1}^{U_k}$ using Algorithm 2.1
14:         Store samples $\{\boldsymbol{u}^{(i)}\}_{i=1}^{U_k}$ in cache
15:     end if
16:     Compute sample positions (3.21)
        // Perform measurement update
17:     Compute $\hat{\boldsymbol{y}}_k$, $\mathbf{Y}_k$, and $\mathbf{C}_k$ using (3.17), (3.18), and (3.19) with $\omega_k^{(i)} = \frac{1}{U_k}$
18:     Compute $\hat{\boldsymbol{x}}_{k|k}$ and $\mathbf{P}_{k|k}$ using (3.8) and (3.9)
19: end for
```

Algorithm 3.1: The smart sampling Kalman filter (S^{2}KF).

which might occur due to numerical issues in case of negative sample weights. Such situations even might happen in square root formulations of the Kalman filters due to the involved rank-one downdates needed for samples with negative weights[1]. We will discuss this problem in more detail in the context of extended object tracking in the next section.

3.4 Evaluation

After introducing the S^{2}KF, we compare its performance with state-of-the-art Kalman filters. First, we demonstrate the advantage of using the point-symmetric version of the LCD-based Gaussian sampling over its original asymmetric approach in the context of sample-based Kalman filtering. Second, we consider estimating pose and extents of a cylinder in 3D based on many noisy point measurements. This includes the derivation of a sophisticated measurement model.

[1]Given the square root $\mathbf{L} = \operatorname{chol}(\mathbf{A})$ of a semi-positive definite matrix $\mathbf{A}$, a rank-one downdate efficiently computes $\mathbf{L}^* = \operatorname{chol}(\mathbf{A} - \boldsymbol{s}\boldsymbol{s}^\top)$ without explicitly computing $\mathbf{A} = \mathbf{L}\mathbf{L}^\top$, where $\boldsymbol{s}$ is a vector of appropriate dimension, i.e., a negatively weighted sample used for the covariance computation. The result $\mathbf{L}^*$, however, is not guaranteed to be a valid square root as $\mathbf{A} - \boldsymbol{s}\boldsymbol{s}^\top$ might be indefinite or negative definite (in contrast to a rank-one update that computes $\operatorname{chol}(\mathbf{A} + \boldsymbol{s}\boldsymbol{s}^\top)$).

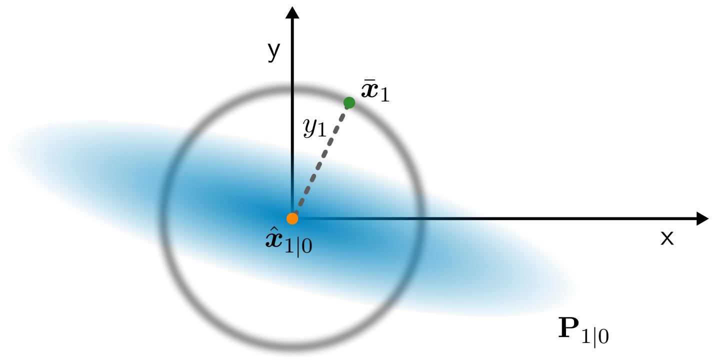


Figure 3.1: Considered estimation problem with symmetric measurement model: initial state mean (orange dot), initial state covariance (blue ellipse), true system state (green dot), and possible noisy distance measurements (gray shaded circle) (cf. [175]).

3.4.1 Asymmetric vs. Point-Symmetric LCD-Based Sampling

We consider estimating the hidden system state

$$\boldsymbol{x}_k = \begin{bmatrix} \mathsf{x}_k \\ \mathsf{y}_k \end{bmatrix}$$

based on the scalar and symmetric measurement equation

$$y_k = h(\boldsymbol{x}_k, v) = \sqrt{\mathsf{x}_k^2 + \mathsf{y}_k^2} + v \quad , \tag{3.22}$$

with state-independent time-invariant Gaussian noise $v \sim \mathcal{N}(0, 10^{-2})$. That is, we measure the Euclidean distance from the state space origin. Such a symmetric measurement equation arises for example in [136, 137]. Further, we assume that the true system state at time step $k = 1$ is

$$\bar{\boldsymbol{x}}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad .$$

Beginning with the state estimate

$$\hat{\boldsymbol{x}}_{1|0} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad , \quad \mathbf{P}_{1|0} = \begin{bmatrix} 4 & -1 \\ -1 & 0.5 \end{bmatrix} \quad ,$$

our goal is to estimate the system state $\boldsymbol{x}_1$ with the aid of a Kalman filter. This situation is illustrated in Figure 3.1. From the the estimator's perspective, a received measurement $\tilde{y}_1$ could originate from any system state located on the gray circle around the prior mean, not only $\bar{\boldsymbol{x}}_1$. Hence, a Kalman filter cannot gain any information about the hidden system state from a measurement $\tilde{y}_1$. This is reflected by a zero cross-covariance matrix $\mathbf{C}_1$ of state and measurement, which in turn prevents any change in the current state estimate, i.e., see the Kalman filter update formulas (3.8) and (3.9). As a consequence, the posterior estimate always equals the prior estimate, i.e., $\hat{\boldsymbol{x}}_{1|1} = \hat{\boldsymbol{x}}_{1|0}$ and $\mathbf{P}_{1|1} = \mathbf{P}_{1|0}$. Moreover, this holds for any prior covariance matrix $\mathbf{P}_{1|0}$.

Now, we try to reproduce this by means of sample-based Kalman filters. More precisely, we compare

- the S^{2}KF using 11 samples,
- the S^{2}KF with the original asymmetric LCD-based sampling for the approximation of the standard normal distribution (instead of the point-symmetric version) also using 11 samples, and

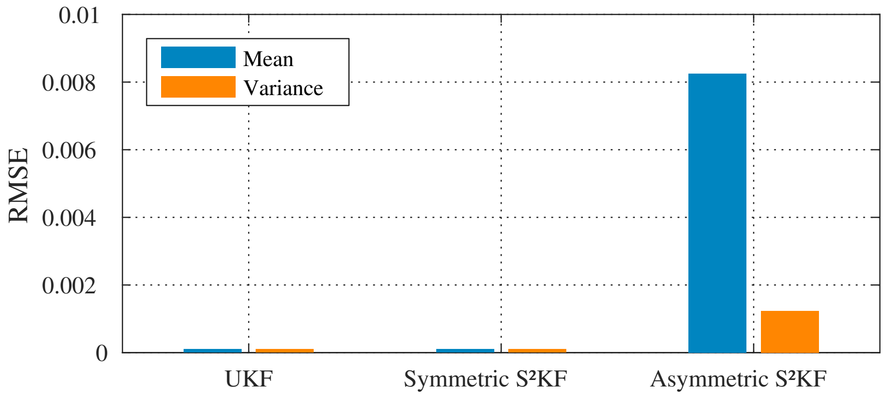


Figure 3.2: Results of the Monte Carlo simulation.

- the UKF with equally weighted samples.

We perform $R = 100$ Monte Carlo runs. In each run, we reset the initial state estimate, simulate a noisy measurement $\tilde{y}_1$ according to (3.22), and perform a single measurement update. Moreover, in order to get more convincing results, both S^{2}KF variants compute a new set of samples $\{\boldsymbol{u}^{(i)}\}_{i=1}^{U_k}$ approximating a standard normal distribution in each Monte Carlo run. We compute the RMSE of the posterior mean according to

$$\sqrt{\frac{1}{R}\sum_{r=1}^{R} \left\| \hat{\boldsymbol{x}}_{1|1}^{(r)} - \hat{\boldsymbol{x}}_{1|0} \right\|_2^2}$$

and the RMSE of the posterior covariance according to

$$\sqrt{\frac{1}{R}\sum_{r=1}^{R} \left\| \mathbf{P}_{1|1}^{(r)} - \mathbf{P}_{1|0} \right\|_{\mathrm{F}}^2} \ ,$$

where $\hat{\boldsymbol{x}}_{1|1}^{(r)}$ and $\mathbf{P}_{1|1}^{(r)}$ denote the posterior moments of simulation run r.

The results of the evaluation are shown in Figure 3.2. It can be seen that both the UKF and the proposed S^{2}KF (with point-symmetric samples) have no errors. This behavior is due the fact that these filters compute the cross-covariance matrix correctly according to

$$\begin{aligned}\mathbf{C}_1 &= \frac{1}{2L+1}\left(\begin{bmatrix}0\\0\end{bmatrix}\left(\hat{y}_1 - \sqrt{0^2+0^2}\right) + \sum_{i=1}^{L}\left(\begin{bmatrix}\mathrm{x}_1^{(i)}\\ \mathrm{y}_1^{(i)}\end{bmatrix} - \begin{bmatrix}\mathrm{x}_1^{(i)}\\ \mathrm{y}_1^{(i)}\end{bmatrix}\right)\left(\hat{y}_1 - \sqrt{(\mathrm{x}_1^{(i)})^2 + (\mathrm{y}_1^{(i)})^2}\right)\right)\\ &= \begin{bmatrix}0\\0\end{bmatrix} \ ,\end{aligned}$$

where $\hat{y}_1$ denotes the measurement mean, $L = 2$ for the UKF, and $L = 5$ for the S^{2}KF. In contrast, for the S^{2}KF using the asymmetric LCD-based sampling scheme, the cross-covariance matrix does not necessarily evaluate to zero. In such a case, it introduces (theoretically non-existent) correlations between the measurement and the system state. Consequently, the asymmetric S^{2}KF slightly changes its state estimate mistakenly. Over time, those small errors can accumulate to non-negligible estimation errors or even yield filter divergence. The other estimators do not have this problem due to their point-symmetric sampling schemes. So even such a simple scenario demonstrates the advantage of the new point-symmetric version of the LCD-based Gaussian sampling when used in Kalman filtering.

3.4.2 Tracking a Cylinder in 3D

In the second evaluation, we want to track the pose of a cylinder in 3D whose shape is unknown in advance and changes over time. Hence, we have to estimate the cylinder's current shape in addition to its pose and motion parameters. In particular, we are interested in

- the position of the cylinder's centroid $\boldsymbol{p}_k = [p_k^{(\mathsf{x})}, p_k^{(\mathsf{y})}, p_k^{(\mathsf{z})}]^\top$ in m,
- the centroid's velocities $\dot{\boldsymbol{p}}_k = [\dot{p}_k^{(\mathsf{x})}, \dot{p}_k^{(\mathsf{y})}, \dot{p}_k^{(\mathsf{z})}]^\top$ in $\mathrm{m/s}$,
- the cylinder's orientation described by the angles $\boldsymbol{\phi}_k = [\phi_k^{(\mathsf{x})}, \phi_k^{(\mathsf{y})}]^\top$ in rad,
- the angular velocities $\dot{\boldsymbol{\phi}}_k = [\dot{\phi}_k^{(\mathsf{x})}, \dot{\phi}_k^{(\mathsf{y})}]^\top$ in $\mathrm{rad/s}$,
- the cylinder radius r_k in m, and
- the cylinder height l_k in m,

see Figure 3.3. The rotation matrix of the cylinder is defined as the matrix product $\mathbf{T}(\phi^{(\mathsf{y})})\mathbf{T}(\phi^{(\mathsf{x})})$, where $\mathbf{T}$ denotes the 3D rotation matrix around the respective axis. Note that, due to rotation invariance, we do not estimate the cylinder's rotation around its z-axis. Thus, the entire system state to be estimated is

$$\boldsymbol{x}_k = \left[\boldsymbol{p}_k^\top, \dot{\boldsymbol{p}}_k^\top, \boldsymbol{\phi}_k^\top, \dot{\boldsymbol{\phi}}_k^\top, r_k, l_k\right]^\top \in \mathbb{R}^{12} \ . \tag{3.23}$$

The temporal evolution of the cylinder is described with a constant velocity/turn rate model for the pose and a random walk model for the shape parameters according to

$$\boldsymbol{x}_k = \mathbf{A}\boldsymbol{x}_{k-1} + \mathbf{B}\boldsymbol{w} \tag{3.24}$$

with matrices

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_3 & \Delta t\,\mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_2 & \Delta t\,\mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_2 \end{bmatrix} \ , \quad \mathbf{B} = \begin{bmatrix} \Delta t\,\mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Delta t\,\mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_2 \end{bmatrix} \ ,$$

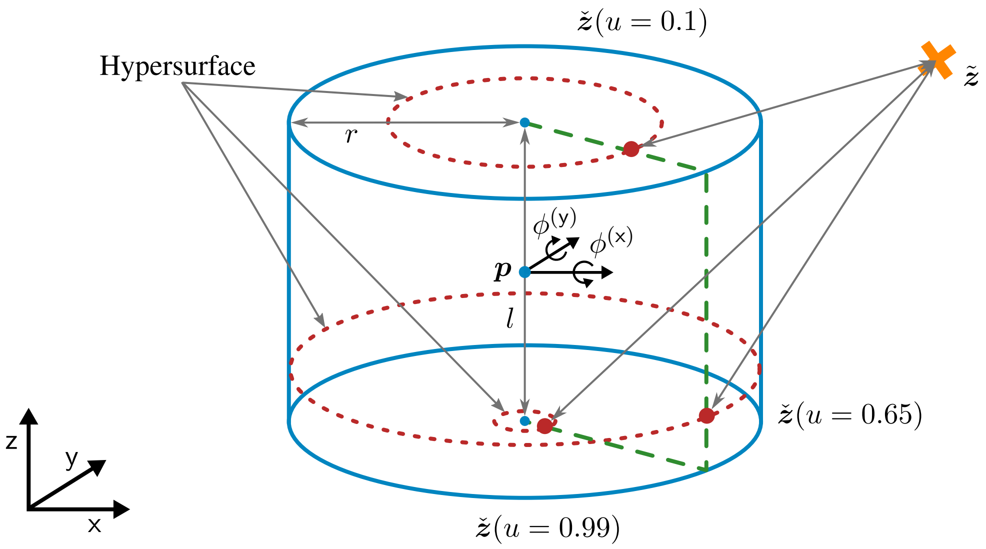
time period $\Delta t = 0.03\,\mathrm{s}$, and time-invariant zero-mean white Gaussian noise with covariance matrix $\mathbf{Q} = \mathrm{diag}(10\,\mathbf{I}_3, 10^{-1}\,\mathbf{I}_2, \mathbf{I}_2)$.

Measurement Model

Over time, we receive noisy Cartesian measurement $\tilde{\boldsymbol{y}}_k$ originating from the entire cylinder's surface, including top and bottom. Further, it is assumed that $\tilde{\boldsymbol{y}}_k$ is generated according to the model

$$\boldsymbol{y}_k = \boldsymbol{z}_k + \boldsymbol{v} \ ,$$

where $\boldsymbol{z}_k$ is called the measurement source (on the cylinder's surface) and $\boldsymbol{v}$ is time-invariant zero-mean Gaussian noise with covariance $\mathbf{R} = 10^{-3}\,\mathbf{I}_3$. The problem is, however, that we do *not* know the measurement source $\boldsymbol{z}_k$. In other words, we do not have an explicitly formulated relationship between system state $\boldsymbol{x}_k$ and source $\boldsymbol{z}_k$ for a received measurement $\tilde{\boldsymbol{y}}_k$. This problem is analogous to the problem of multi target tracking with unknown associations. In literature, there exist different techniques to model the unknown relationship between state and measurement source. For example, the spatial distribution model (SDM) [79–82], the partial information model (PIM) [90, 91], [174], or the random hypersurface model (RHM) [83–89].



$\check{\boldsymbol{z}}(u=0.1)$
Hypersurface
$\tilde{\boldsymbol{z}}$
r
$\phi^{(y)}$
$\phi^{(x)}$
$\boldsymbol{p}$
l
z
y
x
$\check{\boldsymbol{z}}(u=0.65)$
$\check{\boldsymbol{z}}(u=0.99)$

Figure 3.3: Cylinder parametrization and pursued RHM approach to deal with the unknown measurement sources. For different realizations of u, the respective measurement sources $\check{\boldsymbol{z}}_k$ are shown. The time index k is omitted for better readability.

We choose to use an RHM that consists of two aspects: (i) reducing the actual object's surface to a surface of lower dimension, i.e., a *hypersurface* (in case of the cylinder from 2D to 1D) and (ii) determining the point $\check{\boldsymbol{z}}_k$ on the hypersurface that is closest to the point $\tilde{\boldsymbol{z}}_k = \tilde{\boldsymbol{y}}_k - \boldsymbol{v}$, i.e., a *greedy association*. The point $\check{\boldsymbol{z}}_k$ is then assumed to be the measurement source for $\tilde{\boldsymbol{y}}_k$. Which hypersurface is used for the greedy association is chosen in a probabilistic way. More precisely, the hypersurface selection depends on a random variable with a predefined probability distribution. For the considered cylinder, this is illustrated in Figure 3.3. Here, a hypersurface is a circle of certain radius located on the surface of the cylinder. Consequently, if the circle is on the side of the cylinder, it has a radius of r_k. If it is on the top or bottom, it can have a radius in $[0, r_k]$. Moreover, we assume that we receive measurements that are uniformly distributed on the entire surface of the cylinder. Hence, location and radius of the circle is determined with a scalar random variable $u \sim \mathcal{U}(0, 1)$. How u is used to determine a concrete hypersurface will be explained later.

Based on the obtained measurement source $\check{\boldsymbol{z}}_k$, we can now formulate a measurement equation that can be used in a Kalman filter to perform the measurement update. Inspired by [87, 138], we choose to rely on the signed Euclidean distance between source $\check{\boldsymbol{z}}_k$ and point $\tilde{\boldsymbol{z}}_k$. The distance is positive if $\tilde{\boldsymbol{z}}_k$ is inside the cylinder and negative if it is outside. Thus, the scalar measurement equation is given by

$$y_k = h(\boldsymbol{x}_k, \boldsymbol{v}, u, \tilde{\boldsymbol{y}}_k) = d(\check{\boldsymbol{z}}_k(\boldsymbol{x}_k, u, \tilde{\boldsymbol{z}}_k), \tilde{\boldsymbol{z}}_k) \enspace , \tag{3.25}$$

where d denotes the signed Euclidean distance[2]. Note that u can be seen as an additional noise term entering the measurement equation. How d is computed is listed in Algorithm 3.2. First, we have to compute the absolute values of radius r_k and length l_k as they are assumed to be positive (line 1). Then, in order to ease the distance computation, we transform $\tilde{\boldsymbol{z}}_k$ into the local coordinate system of the cylinder (line 2). Next, we determine the sign of the distance (lines 3–8). As we assume the measurements to be uniformly distributed on the cylinder surface, we compute the relative areas of the cylinder top/bottom a and cylinder side b (lines 9–10). With a and b, we compute the signed distance

[2]Using the scalar signed distance also has the advantage that we only have a scalar measurement, which is beneficial for the runtime as the measurement dimension enters the runtime complexity of the Kalman filter cubic fashion (we have to invert the measurement covariance matrix).

```
Input:   state $\boldsymbol{x}$, noise $\boldsymbol{v}$, noise $u$, measurement $\tilde{\boldsymbol{y}}$
Output:  signed Euclidean distance $d$
     // Radius and length are always positive
 1:  $r \leftarrow |r|$ and $l \leftarrow |l|$
     // Transform $\tilde{z}$ into local (cylinder) coordinate system
 2:  $\boldsymbol{m} = \left(\mathbf{T}(\phi^{(\mathsf{y})})\mathbf{T}(\phi^{(\mathsf{x})})\right)^{\top}((\tilde{\boldsymbol{y}} - \boldsymbol{v}) - \boldsymbol{p})$
     // Determine distance sign
 3:  $t = \sqrt{(m^{(\mathsf{x})})^2 + (m^{(\mathsf{y})})^2}$
 4:  if $t \leq r$ and $-\frac{1}{2}l \leq m^{(\mathsf{z})} \leq \frac{1}{2}l$ then
         // Within cylinder
 5:      $q \leftarrow 1$
 6:  else
 7:      $q \leftarrow -1$
 8:  end if
     // Get relative areas of top/bottom and side of the cylinder
 9:  $a = \frac{\pi r^2}{2\pi r^2 + 2\pi r l}$
10:  $b = \frac{2\pi r l}{2\pi r^2 + 2\pi r l}$
     // Select cylinder part according to uniform noise $u$
11:  if $u < a$ then
         // Source on top
12:      $d \leftarrow q \cdot \sqrt{\left(\sqrt{\frac{u}{a}} \cdot r - t\right)^2 + \left(\frac{1}{2}l - m^{(\mathsf{z})}\right)^2}$
13:  else if $u < a + b$ then
         // Source on side
14:      $d \leftarrow q \cdot \sqrt{\left(r - t\right)^2 + \left(\left(\frac{1}{2} - \frac{u-a}{b}\right) \cdot l - m^{(\mathsf{z})}\right)^2}$
15:  else
         // Source on bottom
16:      $d \leftarrow q \cdot \sqrt{\left(\sqrt{1 - \frac{u-(a+b)}{a}} \cdot r - t\right)^2 + \left(-\frac{1}{2}l - m^{(\mathsf{z})}\right)^2}$
17:  end if
```

Algorithm 3.2: Cylinder measurement model. For better readability, the time index k is omitted.

depending on the value of u (lines 11–17). Note that we do not explicitly compute the source $\check{\boldsymbol{z}}_k$ to get the Euclidean distance between $\check{\boldsymbol{z}}_k$ and $\tilde{\boldsymbol{z}}_k$. For the determination of the radius of the hypersurface (if $\check{\boldsymbol{z}}$ is located on the top or bottom), we exploit that points on a unit disk are uniformly distributed if their distance from the center is distributed according to $\sqrt{u}$, where $u \sim \mathcal{U}(0, 1)$ [83].

Next, in order to work with the measurement equation (3.25), we need a measurement as input for the Kalman filter, as the received measurement $\tilde{\boldsymbol{y}}_k$ directly enters the measurement equation h. For a cylinder that perfectly fits to a received measurement, the measurement equation h becomes 0. Hence, we use the *pseudo measurement* 0 as actual measurement for the Kalman filter. In other words, the Kalman filter is forced to locate, i.e., estimate, the cylinder such that the measurement mean $\hat{y}_k$ becomes 0. This in turn means that the estimated cylinder has an average signed Euclidean distance of 0 to a measurement $\tilde{\boldsymbol{y}}_k$.

At this point, we still have the problem that the random variable u is uniformly distributed. Sample-based Kalman filters, however, can only sample Gaussian distributions. This issue could be addressed by either sampling u separately and using the Cartesian product with the Gaussian samples or by approximating u as Gaussian distribution by means of moment matching. Both approaches have their drawbacks, though. On the one hand, the Cartesian product becomes computational expensive, especially if many measurements are processed at once, which will be the case in our evaluation (each measurement requires its own random variable u as those are assumed to be mutually independent for different measurements). On the other hand, approximating u as a Gaussian distribution would not only change the actual distribution of u, it also could result in samples that are outside of the interval $[0, 1]$, and thus violates the assumptions of the measurement model: how to deal with values $u < 0$ or $u > 1$? A much better solution is to exploit the fact that transforming an arbitrary random variable s with its own cumulative distribution function $F(s)$ yields a random variable $u = F(s)$ for which $u \sim \mathcal{U}(0, 1)$ holds [106, Sec. 5.2.]. In our case, we set s to be standard normal distributed. Hence, we can correctly sample u with any sample-based Kalman filter. We merely need the cumulative distribution function of the univariate standard normal distribution, which is given by

$$\Phi(s) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{s}{\sqrt{2}}\right)\right) \quad ,$$

where erf denotes the error function that can be found, for example, in the C standard library or MATLAB. Thus, by using s as actual noise term instead of u, the final measurement equation becomes

$$y_k = h(\boldsymbol{x}_k, \boldsymbol{v}, s, \tilde{\boldsymbol{y}}_k) = d(\check{\boldsymbol{z}}_k(\boldsymbol{x}_k, \Phi(s), \tilde{\boldsymbol{z}}_k), \tilde{\boldsymbol{z}}_k) \enspace . \tag{3.26}$$

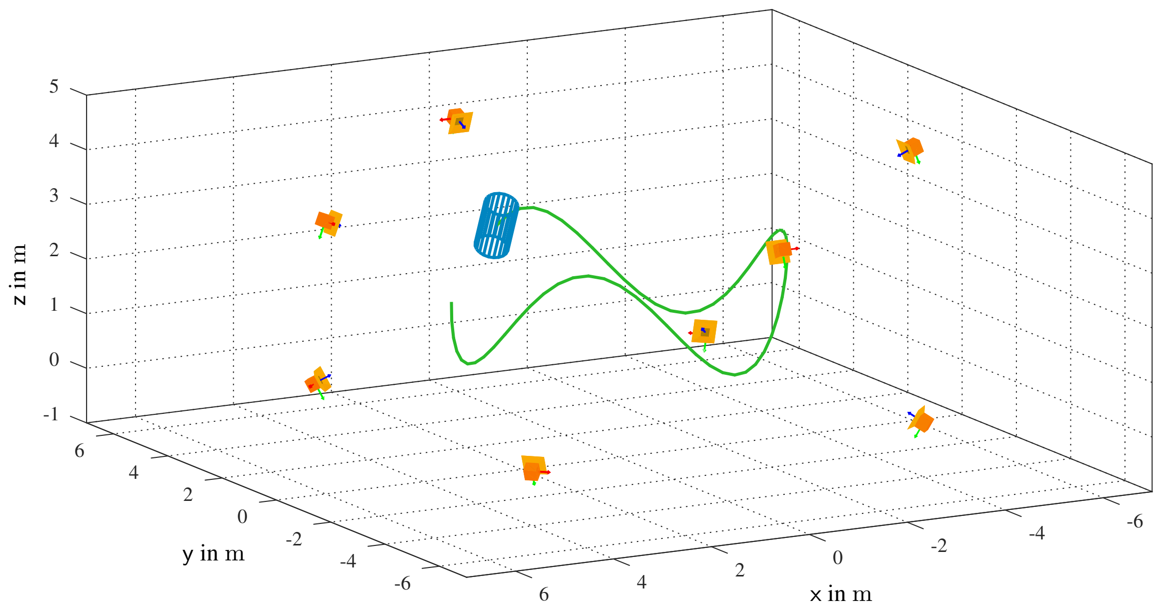
Note that the cumulative distribution approach can be simply extended to sample an arbitrary non-Gaussian distribution as long as its cumulative distribution function is invertible, e.g., see [139]. Please also note that the proposed cylinder measurement model differs from those used in [86], [175]. First, in this thesis we consider the more demanding case where measurements can also originate from top and bottom of the cylinder. Second, the uniform distribution u is not approximated as a Gaussian distribution by means of moment matching. Finally, we rely on a scalar measurement equation, whereas in [86] the equation is two-dimensional and in [175] three-dimensional.

Simulation

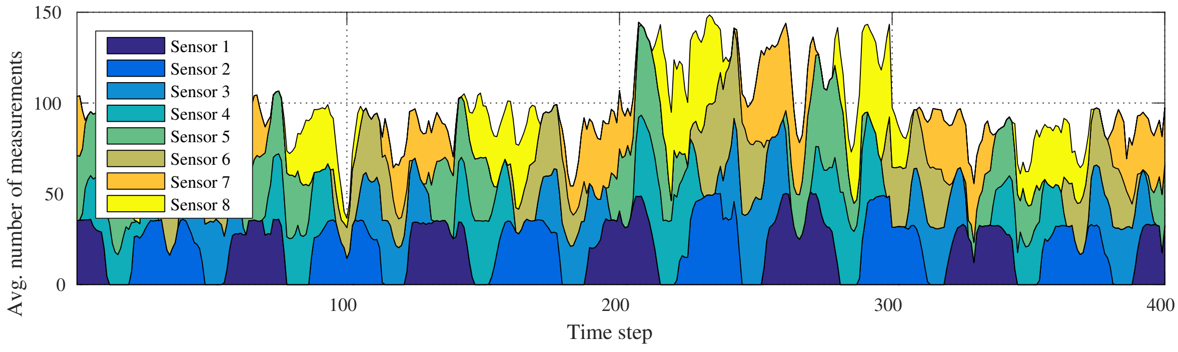
We simulate a nonlinear trajectory of a cylinder over 400 time steps including rotations in all its three degrees of freedom, see Figure 3.4(a). Additionally, the initial cylinder radius of $0.3\,\mathrm{m}$ increases to $0.4\,\mathrm{m}$ after 200 time steps, and the initial length of $1\,\mathrm{m}$ shrinks to $0.5\,\mathrm{m}$ after further 100 time steps. During the cylinder's movement, it is observed by eight sensors with a limited field of view. Consequently, on average the cylinder is only seen by three sensors at the same time. If the cylinder is inside the i^{th} sensor's field of view, it generates $Y_k^{(i)}$ noisy measurements $\mathcal{Y}_k^{(i)} = \{\tilde{\boldsymbol{y}}_k^{(i,j)}\}_{j=1}^{Y_k^{(i)}}$. In each time step, we collect the available $Y_k = Y_K^{(1)} + \cdots + Y_k^{(8)}$ measurements from all sensors in the set $\mathcal{Y}_k = \{\mathcal{Y}_k^{(1)}, \ldots, \mathcal{Y}_k^{(8)}\}$. Note that both the changing cylinder shape and the sensor properties lead to a time-varying number of measurements.

We perform 100 Monte Carlo runs on a system with Intel Core i7-3770 CPU ($3.4\,\mathrm{GHz}$, 4 cores, 8 threads). In Figure 3.4(b), the average number of measurements over all simulation runs is shown. It can be seen that most times we receive a total number of approximately 100 measurements per time step. The overall minimum and maximum number of measurements was 29 and 169, respectively. All measurements $\mathcal{Y}_k$ are processed *at once* by the Kalman filters using the stacked measurement equation

$$\begin{bmatrix} y_k^{(1)} \\ \vdots \\ y_k^{(Y_k)} \end{bmatrix} = \begin{bmatrix} h(\boldsymbol{x}_k, \boldsymbol{v}^{(1)}, s^{(1)}, \tilde{\boldsymbol{y}}_k^{(1)}) \\ \vdots \\ h(\boldsymbol{x}_k, \boldsymbol{v}^{(Y_k)}, s^{(Y_k)}, \tilde{\boldsymbol{y}}_k^{(Y_k)}) \end{bmatrix}$$



(a) Eight sensors (orange) with a limited field of view observe a rotating cylinder (blue) that moves along a nonlinear path (parts of it shown as green curve).



(b) Average number of measurements over all simulation runs.

Figure 3.4: Considered cylinder tracking scenario.

and pseudo measurement vector $[0, \dots, 0]^\top$. The noise variables $\boldsymbol{v}^{(j)}$ and $s^{(j)}$ are assumed to be white and mutually independent. Due to the stacked measurements, a sample-based Kalman filter has to sample a joint space of state and noise variables that has $J_k = 12 + Y_k \cdot 4$ dimensions in order to perform a measurement update. This results in very large joint spaces of up to $12 + 169 \cdot 4 = 688$ dimensions (for $Y_k = 169$). Regarding the time update, the linear system model allows for a closed-form prediction for all Kalman filters.

We investigate the following estimators:

- the EKF using finite differences to approx. the Jacobian of the stacked measurement equation,
- the UKF with $2J_k + 1$ equally weighted samples,
- the RUKF with 5 iterations (“RUKF (5x)”), resulting in $5 \cdot 2J_k + 1 = 10J_k + 1$ samples,
- the RUKF with 8 iterations (“RUKF (8x)”), resulting in $8 \cdot 2J_k + 1 = 16J_k + 1$ samples, and
- the S^{2}KF also using $10J_k + 1$ samples.

Note that the GHKF is obviously intractable for the considered tracking scenario as it would require at least 2^{128} samples (for $Y_k = 29$) to conduct the measurement update. Moreover, the 5th-degree CKF would require the enormous number of up to $2 \cdot 688^2 + 1 = 946\,689$ samples. Unfortunately, even for a smaller number of $Y_k = 50$ measurements (where only $89\,889$ samples are used) the 5th-degree CKF fails to compute valid, i.e., positive definite, measurement covariance matrices or posterior state covariance matrices. The reason for those failures are numerical issues originating from its sampling technique. In particular, the combination of an imbalanced distribution of sample weights (the largest weight is 400 times larger than the smallest weight) and several hundred negatively weighted samples lead very likely to indefinite measurement covariance matrices. And even if a valid measurement covariance matrix is obtained, it is badly conditioned, making a Kalman update impossible. Also a square root variant fails to compute valid covariance matrices due to hundreds of rank-one downdates needed for the negatively weighted samples. Unlike a rank-one update, a rank-one downdate can result in an indefinite covariance matrix, which was the case in our experiments. Thus, we had to excluded the 5th-degree CKF from this evaluation as well.

In each run, we initialize the estimators based on the first set of available measurements $\mathcal{Y}_0$, i.e., we compute mean and covariance matrix

$$\hat{\boldsymbol{p}} = \frac{1}{Y_0} \sum_{j=1}^{Y_0} \tilde{\boldsymbol{y}}_0^{(j)} \enspace ,$$

$$\boldsymbol{\Sigma}^{(p)} = \frac{1}{Y_0} \sum_{j=1}^{Y_0} \left(\tilde{\boldsymbol{y}}_0^{(j)} - \hat{\boldsymbol{p}}\right)\left(\tilde{\boldsymbol{y}}_0^{(j)} - \hat{\boldsymbol{p}}\right)^\top \enspace ,$$

and set the initial state estimate according to

$$\hat{\boldsymbol{x}}_{0|0} = \left[\hat{\boldsymbol{p}}^\top, 0, \ldots, 0, 1, 2\right]^\top \enspace ,$$

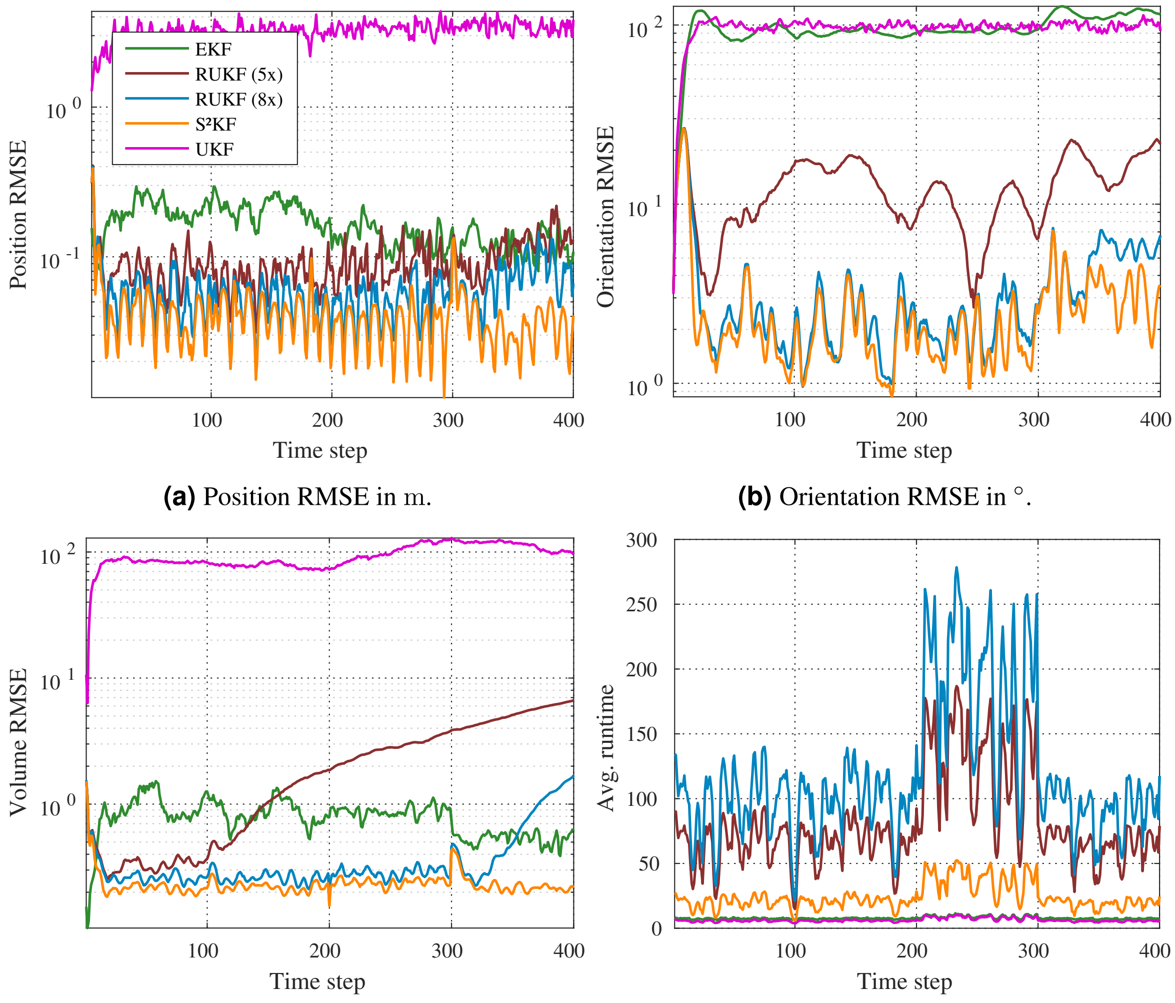
$$\mathbf{P}_{0|0} = \mathrm{diag}(\boldsymbol{\Sigma}^{(p)}, 10^{-3}\,\mathbf{I}_3, 10^{-6}\,\mathbf{I}_4, 10^{-1}\,\mathbf{I}_2) \enspace ,$$

i.e., an upright cylinder of radius $1\,\mathrm{m}$ and length $2\,\mathrm{m}$.

Results

First of all, we have to point out that the RUKF also fails to conduct several measurement updates due to indefinite covariance matrices. Like for the 5th-degree CKF, a square root implementation does not solve this issue. Our solution is to abort the measurement update if a indefinite covariance matrix occurs and leave the state estimate unchanged. On average, $7\,\%$ of the measurement updates of the "RUKF (5x)" and $4\,\%$ of the measurement updates of the "RUKF (8x)" could not be conducted in our evaluation. In contrast, the UKF and S^{2}KF with their equally weighted samples have no problem conducting a measurement update. We will analyze the update issues of the RUKF in more detail later.

We compare the estimation quality of the investigated estimators by means of three indicators: position RMSE, orientation RMSE (given by the angle between the true cylinder's z-axis and the z-axis of the estimated cylinder), and volume RMSE. While the first two errors reflect the performance of the pose estimation, the latter one accounts for the shape estimation quality. The results are depicted in Figures 3.5(a)–3.5(c). From those, we can see that the UKF clearly fails to estimate the cylinder correctly. In fact, it diverges directly from the beginning. While the position estimate of the EKF is acceptable, it fails to estimate the orientation. In addition, the quality of its shape estimate is not very good as well. Regarding position and shape, the "RUKF (5x)" performs much better than the EKF. Even though the cylinder shape estimate is very accurate at the beginning, after 100 time steps it starts to constantly get worse over time. The "RUKF (8x)" has even better results than the "RUKF (5x)". It not only has a smaller position error, it can also estimate cylinder orientation and cylinder shape



(a) Position RMSE in m. **(b)** Orientation RMSE in $^\circ$. **(c)** Volume RMSE in m^3. **(d)** Average runtimes in ms.

Figure 3.5: Results of the cylinder tracking evaluation.

very well. However, like the "RUKF (5x)", it starts to diverge when the cylinder shape rapidly changes at time step 300. Nevertheless, the "RUKF (8x)" is beaten by the S^{2}KF (with fewer samples) as it delivers the best estimation results of all estimators. Moreover, it never starts to diverge when the cylinder shape abruptly changes from time to time. The evaluation results can be visually inspected by comparing the exemplary cylinder estimates shown in Figure 3.6. Note the very smoothly estimated cylinder trajectories of the S^{2}KF and "RUKF (8x)". Also the estimated cylinder shapes are close to the true cylinder. In contrast, the trajectories of the EKF and the "RUKF (5x)" are more jagged. Furthermore, orientation and shape of the EKF are obviously far from being correct.

This evaluation reveals that using more and better placed samples than the widely used UKF can greatly improve the estimation performance. The problem with the UKF sampling is that it only results in three different values for u, and thus only three different measurement sources are probed for all measurements. More precisely, for most state samples, the mean of u ($\hat{u} = 0.5$) is used to determine the unknown measurement source, which means that most samples are mistakenly assigned to the side of the cylinder.

Next, we have a look at the runtimes of the Kalman filters, see Figure 3.5(d). Clearly, EKF and UKF are the fastest estimators with a very constant execution time. However, they are not able to estimate the cylinder. For the other estimators, it can be seen that their runtimes depend much more

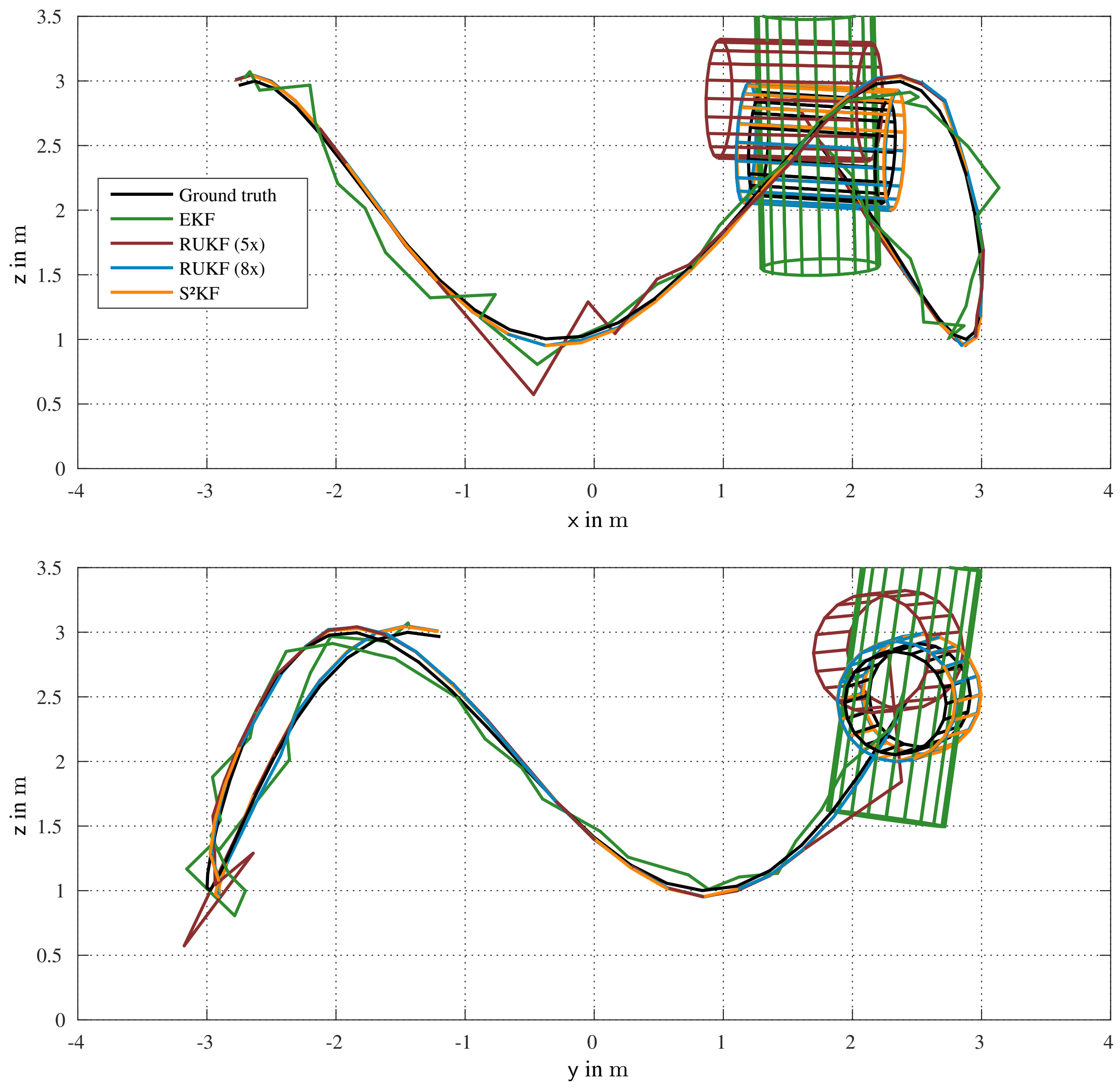


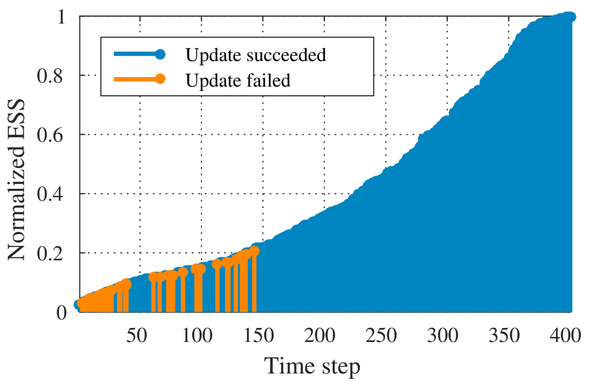
Figure 3.6: Estimated cylinders of a simulation run at time step $k = 210$. The UKF is not shown due to its divergence.

on the number of measurements to be processed. Even though the S^{2}KF and the “RUKF (5x)” use the same amount of samples, the “RUKF (5x)” is much slower. This can be explained by the five (large) random orthogonal matrices which have to be computed in each measurement update. The situation gets even worse for the “RUKF (8x)” as it uses 60 % more samples and three additional random orthogonal matrices per measurement update. All in all, the S^{2}KF delivers the best estimation performance regarding both quality and execution time.

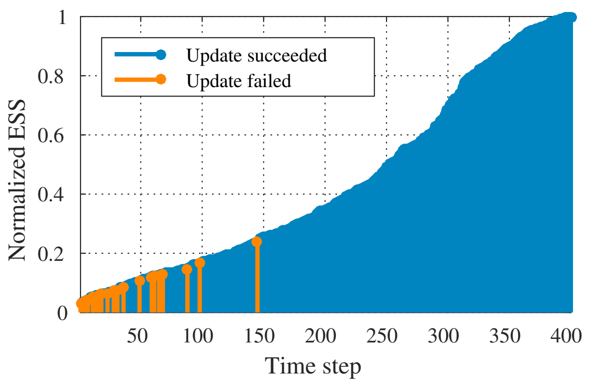
Finally, we analyze the measurement update issues of the RUKF by looking at the normalized effective sample size (ESS), see [12, Sec. 3.2], of the RUKF’s employed samples. The normalized ESS is computed according to

$$\frac{1}{U_k} \leq \frac{1}{U_k \cdot \sum_{i=1}^{U_k} \left(\omega_k^{(i)}\right)^2} \leq 1 \ ,$$

where a value of 1 means all samples are equally weighted and a value of $\frac{1}{U_k}$ means a single sample has a weight of $\omega_k^{(i)} = 1$. Due to the random sample scaling of the RUKF, the sample weights are different for each measurement update. Hence, we have computed the normalized ESS for each time



(a) Normalized ESS for “RUKF (5x)”.



(b) Normalized ESS for “RUKF (8x)”.

Figure 3.7: Sorted normalized ESS of both RUKFs for a representative simulation run.

step of a representative evaluation run. These are plotted in Figure 3.7 in ascending order. Time steps with a failed measurement update are indicated as orange. From these, we can conclude that the RUKF especially has problems in case of a small effective sample size, i.e., if the sample weights are very imbalanced (like the samples of the 5th-degree CKF for larger dimensions).

3.5 Conclusions

In this chapter, we dealt with Kalman filters applied to nonlinear systems. First, we discussed the approach of Kalman filters to approximate the, in general, intractable recursive Bayesian estimation. It turned out that Kalman filters have to compute several first-order and second-order moments of nonlinear transformed Gaussian random vectors. Hence, we studied state-of-the-art approaches for computing these moments, i.e., approximating the considered nonlinear system and measurement models or approximating the multivariate Gaussian distributions with an appropriate set of weighted samples. The latter approach led to the important class of sample-based Kalman filters, whose estimators solely differ in the way they select the samples for the moment computations. In addition, we shortly discussed various existing Kalman filter extensions.

Next, we introduced a new sample-based Kalman filter, the S^2KF, that is based on the point-symmetric LCD-based Gaussian sampling proposed in Chapter 2. The S^2KF has several advantages over state-of-the-art sample-based Kalman filters. Most important, it is the first sample-based Kalman filter for nonlinear systems that uses an arbitrary number of optimally placed and equally weighted samples. This allows for a fine-grained control over estimation quality and execution time. Second, the equal sample weights reduce the number of arithmetic operations required for the moment computations. Third, the strictly equally weighted samples avoid numerical issues when computing covariance matrices, which can be a severe problem for other state-of-the-art Kalman filters, e.g., the 5th-degree CKF or the RUKF. Finally, like other Kalman filters, the S^2KF can directly be used with any Kalman filter extension like iterative measurement updates or Gaussian mixture models.

As a first evaluation of the S^2KF, we compared its point-symmetric sampling scheme against the original asymmetric LCD-based Gaussian sampling by means of a symmetric measurement equation. From this evaluation we can conclude that it is strictly advisable to use the new point-symmetric sampling instead of its asymmetric version as this can introduce non-existent correlations between state and measurement. Second, we investigated the performance of the S^2KF in an extended object tracking application, i.e., we estimated pose and unknown extents of a cylinder based on noisy measurements originating from its surface. For this purpose, we derived a new measurement model using the

RHM approach together with the signed Euclidean distance. This also required to sample a uniform distribution, which is challenging when using sample-based Kalman filters. Thus, we made use of the fact that a random variable transformed with its own cumulative distribution is uniformly distributed. Simulations showed that the S^{2}KF can outperform state-of-the-art Kalman filters in the considered scenario. Moreover, it is much more stable regarding positive definiteness of covariance matrices due to its strictly positive sample weights.

The S^{2}KF is already used for several nonlinear state estimation task. For example, it is extensively utilized for extended object tracking [87, 89, 138, 140], [174, 180], improves simultaneous localization and mapping (SLAM) [141], or allows for real-time whole-body human motion tracking [181]. In Chapter 4, we will also formulate a new algorithm to incorporate the S^{2}KF in a distributed nonlinear state estimation scenario.

Chapter 4

Optimal Sample-Based Fusion for Distributed State Estimation

In Chapter 3, we presented the S^{2}KF, a sample-based Kalman filter for the estimation of discrete-time stochastic nonlinear dynamic systems. One (implicit) assumption during the derivation of the S^{2}KF was that the filter has access to all measurements that are available to infer the state of the considered system. However, this might not always be the case. For example, several, potentially different, sensors observing the same system can be physically distributed over multiple computers, called *sensor nodes*, and the actual state estimation has to be performed at a *fusion center* that is connected with all these sensor nodes[1]. As an example, consider multiple embedded systems that are placed at different locations in a car, each equipped with a sensor such as a radar or a laser scanner. The goal is to estimate the entire environment of the car at a fusion center to enhance safety or to carry out autonomous actions such as turning or overtaking other cars.

In order to estimate the system state, measurements obtained on the sensor nodes could be sent to the fusion center, where the measurements are centrally processed by a recursive estimator. For linear/Gaussian models, the information form of the Kalman filter should be used to communicate the measurement data, e.g., see [68, Sec. 3.4.2]. However, for nonlinear models the advantageous structure of the information form no longer holds when relying on statistical linearization due to the linearization error that introduces correlated measurement noise [77]. Hence, its not simply possible to use the S^{2}KF in its information form. When alternatively sending the raw measurement data to the fusion center, it needs to know all measurement models including the noise properties. If those cannot be known in advance, they have to be communicated as well. Further, there might be situations, where sending all the measurements is simply not possible. These include, for example, situations where (i) the measurement data are enormous and a (frequent) transmission to the fusion center is intractable, (ii) it is impossible for the fusion center to process the measurement data from all sensor nodes in reasonable time, or (iii) communication is not possible at any point in time. In addition, sensor nodes might also need locally obtained state estimates to operate independently from the fusion center, e.g., during a network outage. Hence, some sort of measurement pre-processing on the sensor nodes is required to reduce the amount of data and/or to compensate network outage.

A natural approach to solve these issues is to execute (sample-based) Kalman filters on each sensor node. Those filters separately estimate the common system state solely based on the locally obtained measurements. The fusion center then fuses the local state estimates in the weighted least

[1] Of course, one of the sensor nodes can also act as fusion center.

squares (WLS) sense to infer an overall global state estimate. In doing so, measurements are preprocessed on the sensor nodes, as their information is now encoded in the local estimates. As a consequence, the amount of data that has to be sent to the fusion center is constant over time and does not depend on the number of locally processed measurements. Formally, this distributed state estimation is done as follows. Suppose, the system state $\boldsymbol{x}_k \in \mathbb{R}^N$ is of interest. Moreover, assume we have $L \in \mathbb{N}_+$ local state estimates to be fused at time step k, and the ith local estimate is given by mean $\hat{\boldsymbol{x}}_{k|k}^{(i)}$ and covariance matrix $\mathbf{P}_{k|k}^{(i)}$. Based on these, we first construct a joint mean vector according to

$$\hat{\boldsymbol{m}}_{k|k} = \begin{bmatrix} \hat{\boldsymbol{x}}_{k|k}^{(1)} \\ \vdots \\ \hat{\boldsymbol{x}}_{k|k}^{(L)} \end{bmatrix} , \tag{4.1}$$

and a joint covariance matrix according to

$$\mathbf{J}_{k|k} = \begin{bmatrix} \mathbf{P}_{k|k}^{(1)} & \mathbf{P}_{k|k}^{(1,2)} & \cdots & \mathbf{P}_{k|k}^{(1,L)} \\ \mathbf{P}_{k|k}^{(2,1)} & \mathbf{P}_{k|k}^{(2)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{P}_{k|k}^{(L-1,L)} \\ \mathbf{P}_{k|k}^{(L,1)} & \cdots & \mathbf{P}_{k|k}^{(L,L-1)} & \mathbf{P}_{k|k}^{(L)} \end{bmatrix} , \tag{4.2}$$

where $\mathbf{P}_{k|k}^{(i,j)}$ denotes the correlation matrix of the ith and jth local estimate. These correlations are, in general, non-zero due to the common system noise [73]. Hence, to avoid a too optimistic global estimate, the correlations have to be taken into account during the fusion process. Second, based on $\hat{\boldsymbol{m}}_{k|k}$ and $\mathbf{J}_{k|k}$, the sought-after weighted least squares fusion, i.e., the Bar-Shalom–Campo formulas [73] for the multisensor case, is given by

$$\begin{aligned} \mathbf{P}_{k|k} &= \left(\mathbf{H}^\top \left(\mathbf{J}_{k|k}\right)^{-1} \mathbf{H}\right)^{-1} , \\ \hat{\boldsymbol{x}}_{k|k} &= \mathbf{P}_{k|k} \mathbf{H}^\top \left(\mathbf{J}_{k|k}\right)^{-1} \hat{\boldsymbol{m}}_{k|k} , \end{aligned} \tag{4.3}$$

with matrix $\mathbf{H} = [\mathbf{I}_N, \ldots, \mathbf{I}_N]^\top$.

The distributed state estimation has the advantages that the processing load is distributed among the sensor nodes and that the robustness against system failures is improved. Further, the distributed approach scales much better than a centralized fusion. Nevertheless, it is important to note that the fusion of local estimates is not equivalent to a centralized fusion, where all measurements are available at the fusion center [73]. Note also that in the context of target tracking the distributed state estimation is also called track-to-track fusion [78].

The main problem with this approach, however, is that the fusion center cannot reconstruct the required correlation matrices $\mathbf{P}_{k|k}^{(i,j)}$ solely from the local estimates $\{\hat{\boldsymbol{x}}_{k|k}^{(i)}, \mathbf{P}_{k|k}^{(i)}\}_{i=1}^{L}$. More precisely, a computation of the correlation matrices would require the points in time when measurements were processed as well as the respectively used measurement matrices and Kalman gains. Communicating all this information would need much more bandwidth than the actual measurements, though.

In this chapter, we present a novel approach to exactly reconstruct correlation matrices with the aid of *samples*: besides the actual state estimation, each sensor node processes a set of deterministic samples that encode the correlations with the other local state estimates. These are then sent to the fusion center in addition to state mean and state covariance matrix. The correlation matrix $\mathbf{P}_{k|k}^{(i,j)}$ can then be simply reconstructed by computing a sample cross-covariance matrix using the correlation samples from the ith and jth sensor node. In doing so, we are able to optimally use the S^{2}KF in a distributed setup.

This chapter is based on the publication [182].

4.1 Related Work

There are many different approaches to deal with the problem of unknown correlation matrices in the context of distributed state estimation. The simplest approach would be to assume uncorrelated local state estimates, i.e., use the block diagonal joint covariance matrix approximation

$$\mathbf{J}_{k|k} \approx \operatorname{diag}\left(\mathbf{P}_{k|k}^{(1)}, \ldots, \mathbf{P}_{k|k}^{(L)}\right)$$

for the weighted least squares fusion. Although straight forward to implement and easy to use, assuming uncorrelated estimates would lead to inconsistent state estimates, i.e., the trace of the fused covariance matrix would be too small. Hence, other approaches have been proposed, which are also based on block diagonal joint covariance matrix approximations, but still lead to consistent estimates. One the one hand, the federated Kalman filter [74] artificially inflates the system noise covariance matrix. Consequently, the local estimates become more uncertain as they theoretically should be. Fusing these inflated estimates then under the assumption of uncorrelatedness still yield a consistent estimate. The inflation, however, depends on the number of utilized sensor nodes. Thus, if sensor nodes dynamically are added or removed from the network, each node has to be updated accordingly. On the other hand, covariance intersection [75] directly inflates the covariance matrices $\mathbf{P}_{k|k}^{(i)}$ according to

$$\mathbf{J}_{k|k} \approx \operatorname{diag}\left((\omega_k^{(1)})^{-1}\mathbf{P}_{k|k}^{(1)}, \ldots, (\omega_k^{(L)})^{-1}\mathbf{P}_{k|k}^{(L)}\right) ,$$

with $\sum_{i=1}^{L} \omega_k^{(i)} = 1$, that is, the rate of inflation can be different for the local estimates. The values $\omega_k^{(i)}$ can be either optimized for each individual fusion process such that a certain criterion is minimized, e.g., the trace of $\mathbf{P}_{k|k}$, or can be predefined, e.g., all local estimates are equally weighted with $\omega_k^{(i)} = \frac{1}{L}$. Although both the federated Kalman filter and covariance intersection yield consistent estimates, they can be very pessimistic, as they do not have any information about the true correlations. Another similar approach is ellipsoidal intersection [142], for which consistency, however, has not been proofed yet.

A completely different approach is taken by the optimal distributed Kalman filter (DKF) [69–72]. The key idea of the DKF is to locally combine measurements over an arbitrary number of time steps to a compact representation of constant size. In doing so, the amount of data that has to be sent to the fusion center does not change over time. Moreover, a fusion of the locally obtained measurement representations at the fusion center can be performed after an arbitrary number of time steps. Compared to the weighted least squares fusion (4.3), the DKF is even equivalent to the centralized processing of all measurements. A limitation of the DKF is that each sensor node needs to know the measurement models of *all* other sensor nodes (measurement matrices and noise covariances) as well as the points in time when measurements are processed. Thus, to avoid any communication between the nodes, all of these factors have to be known in advance, which is especially hard to achieve in a dynamic network. In order to circumvent this, the knowledge assumption can be relaxed by replacing it with a hypothesis, which results in the hypothesizing DKF [143, 144]. Of course, if the hypothesis is incorrect, the filter is no more equivalent to the optimal central processing of measurements. A further important limitation of the DKF is that it cannot simply work with nonlinear models. The problem is that the required linearized measurement models depend on the current local state estimates. Consequently, these cannot be known in advance. Hence, it is not simply possible to use the S^{2}KF in combination with the DKF.

A sample-based distributed state estimation approach is proposed in [76]. Here, the joint covariance matrix $\mathbf{J}_{k|k}$, and in particular the correlation matrices $\mathbf{P}_{k|k}^{(i,j)}$, is approximated from samples that are processed on the sensor nodes. In order to get $\mathbf{J}_{k|k}$, only a sample covariance matrix has to be computed out of these samples, which have to be sent to the fusion center. The key idea of this approach is to model the common system noise with randomly drawn noise samples that are identical on all sensor nodes. This is achieved by synchronizing the node's pseudo random number generators. The problem with this approach is, however, that it only asymptotically converges to the true joint covariance matrix, which can lead to unsatisfactory fusion results. The approximation quality can only be improved by increasing the number of samples, which in turn increases the communication overhead. Another problem is that the samples have to span the joint space of all local estimates. As a consequence, the required number of samples increase at least linearly in the number of sensor nodes, which in turn results in a quadratic increase in the overall amount of sample data that has to be transferred to the fusion center by each node. Moreover, this also implies that each sensor node has to be aware of the number of all nodes in the network.

4.2 Optimal Fusion for Distributed Linear State Estimation

In this section, we propose an approach to *exactly* reconstruct the correlation matrices $\mathbf{P}_{k|k}^{(i,j)}$ at the fusion center without communicating measurements or measurement model information. That is, we can optimally fuse locally obtained state estimates in the weighted least squares sense. Like in [76], the approach also relies on samples, but, instead of using random samples, we use deterministic samples. Further, the samples do not represent the entire joint covariance matrix $\mathbf{J}_{k|k}$ of all estimates, i.e., they are only used to reconstruct the correlations. Next, we consider optimal distributed state estimation for the linear case. The proposed method is then extended to the nonlinear case in Section 4.3.

Our goal to estimate the hidden state $\boldsymbol{x}_k \in \mathbb{R}^N$ of a discrete-time stochastic linear dynamic system described by the system model

$$\boldsymbol{x}_k = \mathbf{A}_k \boldsymbol{x}_{k-1} + \mathbf{B}_k \boldsymbol{w}_k \ , \tag{4.4}$$

with zero-mean white state-independent Gaussian noise $\boldsymbol{w}_k \in \mathbb{R}^{W_k}$ with covariance matrix $\mathbf{Q}_k$. In addition, we assume $L \in \mathbb{N}_+$ sensor nodes. The i^{th} node obtains noisy measurements according to the measurement model

$$\boldsymbol{y}_k^{(i)} = \mathbf{H}_k^{(i)} \boldsymbol{x}_k + \boldsymbol{v}_k^{(i)} \ , \tag{4.5}$$

where $\boldsymbol{v}_k^{(i)} \in \mathbb{R}^{V_k}$ denotes zero-mean white state-independent Gaussian noise with covariance matrix $\mathbf{R}_k^{(i)}$. Moreover, it is assumed that $\boldsymbol{v}_k^{(i)}$ is independent of $\boldsymbol{v}_k^{(j)}$ for $i \neq j$.

The optimal sample-based fusion technique for distributed state estimation is an iterative procedure, where each iteration consists of three phases:

(i) re-initialization of the sensor nodes,

(ii) recursive prediction and filtering on the sensor nodes, and

(iii) optimal weighted least squares fusion at the fusion center.

In the following, we first go step by step through all these phases and then summarize the proposed fusion technique and discuss its properties and advantages.

4.2.1 (Re-)Initialization of the Sensor Nodes

At time step k, the fusion center sends the latest global state estimate $\hat{\boldsymbol{x}}_{k|k}$ and $\mathbf{P}_{k|k}$ to all sensor nodes. In particular, for $k = 0$ the initial global state estimate $\hat{\boldsymbol{x}}_{0|0}$, $\mathbf{P}_{0|0}$ is sent. With these, the i^{th} sensor node first resets its local estimate to

$$\begin{aligned} \hat{\boldsymbol{x}}_{k|k}^{(i)} &= \hat{\boldsymbol{x}}_{k|k} \ , \\ \mathbf{P}_{k|k}^{(i)} &= \mathbf{P}_{k|k} \ . \end{aligned} \tag{4.6}$$

```
Input:   sample dimension $D$
Output:  simplex sample set $\{\boldsymbol{s}^{(m)}\}_{m=1}^{D+1}$
1: $\omega = \frac{1}{D+1}$
2: for $1 \leq d \leq D$
3:     $c = -\frac{1}{\sqrt{d(d+1)\omega}}$
       // Add additional dimension to existing samples (for $d = 1$ this adds the first sample)
4:     for $1 \leq m \leq d$
5:         $s_d^{(m)} = c$
6:     end for
       // Add additional sample
7:     $\boldsymbol{s}^{(d+1)} = \begin{bmatrix} \mathbf{0}_{d-1} \\ -dc \end{bmatrix}$
8: end for
```

Algorithm 4.1: Simplex sample set computation (simplified version from [25]).

Second, the i^{th} sensor node re-initializes the samples required for the reconstruction of the correlations. For this, we consider the joint space

$$\boldsymbol{d}_k = \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{w}_{k+1} \\ \vdots \\ \boldsymbol{w}_{k+P} \end{bmatrix} \tag{4.7}$$

consisting of the current state and the system noise for the next $P \in \mathbb{N}_+$ time steps, where P is a user-defined parameter. It controls for how many time steps a sensor node can do recursive prediction and filtering before it has to be re-initialized. We discuss the role of P and its determination in more detail later. For now, we assume that P is known by all sensor nodes. Further, $\boldsymbol{d}_k$ is of dimension

$$D = N + W_{k+1} + \ldots + W_{k+P} \ .$$

Next, all sensor nodes create an *identical* set of $M = D+1$ equally weighted samples $\{\boldsymbol{s}^{(m)}\}_{m=1}^{M}$ with $\boldsymbol{s}^{(m)} \in \mathbb{R}^D$. These are generated according to the simple deterministic spherical simplex sampling scheme[2] proposed in [25], which is listed in Algorithm 4.1. The samples $\{\boldsymbol{s}^{(m)}\}_{m=1}^{M}$ have zero mean and unit covariance matrix, i.e.,

$$\frac{1}{M}\sum_{m=1}^{M} \boldsymbol{s}^{(m)} = 0$$

and

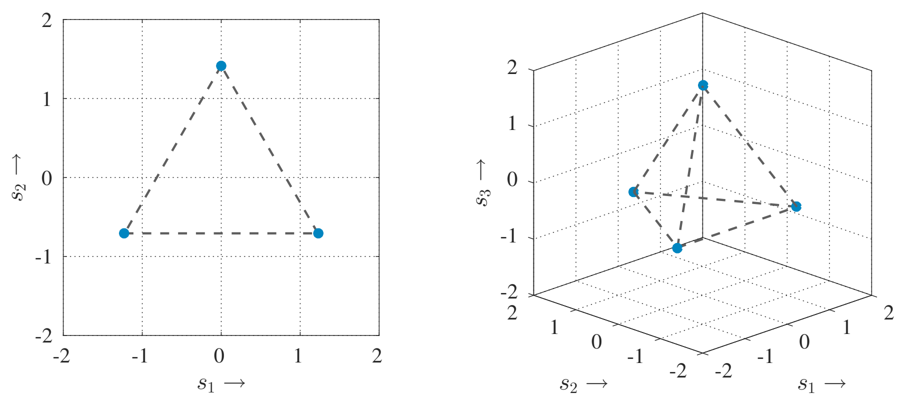
$$\frac{1}{M}\sum_{m=1}^{M} \boldsymbol{s}^{(m)}\big(\boldsymbol{s}^{(m)}\big)^\top = \mathbf{I}_D \ .$$

Note that the number of samples M cannot be further reduced as at least $D+1$ samples are required to represent a valid covariance matrix in D dimensions. Simplex sample sets for different dimensions D are depicted in Figure 4.1.

Next, we build a joint covariance matrix for the joint space $\boldsymbol{d}_k$ according to

$$\mathbf{D}_k = \operatorname{diag}\big(\mathbf{P}_{k|k}, \mathbf{Q}_{k+1}, \ldots, \mathbf{Q}_{k+P}\big) \ .$$

[2]That is, the samples $\{\boldsymbol{s}^{(m)}\}_{m=1}^{M}$ *never* have to be transferred over the network!



(a) Samples for $D = 2$. **(b)** Samples for $D = 3$.

Figure 4.1: Simplex sample sets (blue dots) for different dimensions D (gray dashed lines for better visualization).

Its block diagonal structure reflects the white system noise and its assumed independence of the system state. We compute the Cholesky decomposition $\mathbf{D}_k = \mathbf{L}_k \mathbf{L}_k^\top$ in order to perform a Mahalanobis transformation, see Section 2.1.1. Note that the lower triangular matrix $\mathbf{L}_k$ is also block diagonal and is composed of the Cholesky decompositions of the block matrices of $\mathbf{D}_k$. Based on $\mathbf{L}_k$, we transform the samples $\{\boldsymbol{s}^{(m)}\}_{m=1}^M$ according to

$$\boldsymbol{d}_k^{(m)} = \begin{bmatrix} \boldsymbol{c}_{k|k}^{(i,m)} \\ \boldsymbol{w}_{k+1}^{(m)} \\ \vdots \\ \boldsymbol{w}_{k+P}^{(m)} \end{bmatrix} = \mathbf{L}_k \boldsymbol{s}^{(m)} \quad , \quad \forall m \in \{1, \ldots, M\} \quad , \tag{4.8}$$

i.e., we obtain the equally weighted samples $\{\boldsymbol{d}_k^{(m)}\}_{m=1}^M$ that have zero mean and covariance matrix $\mathbf{D}_k$. These samples are then partitioned into the respective subspaces for system state and system noise for the next P time steps. As a result, we get

- the set of correlation samples $\{\boldsymbol{c}_{k|k}^{(i,m)}\}_{m=1}^M$ for the i^{th} sensor node and
- P sets of system noise samples $\{\boldsymbol{w}_{k+1}^{(m)}\}_{m=1}^M, \ldots, \{\boldsymbol{w}_{k+P}^{(m)}\}_{m=1}^M$.

These system noise sample sets will be used to predict the correlation samples, see Section 4.2.2.

It is important to note that although the correlation samples $\boldsymbol{c}_{k|k}^{(i,m)}$ lie in the system state space, they do *not* represent the state estimate of the i^{th} sensor node such as in particle filtering. They only encode the correlations with the other locally obtained state estimates. Moreover, note that the generated sample sets $\{\boldsymbol{c}_{k|k}^{(i,m)}\}_{m=1}^M$ and $\{\boldsymbol{w}_{k+1}^{(m)}\}_{m=1}^M, \ldots, \{\boldsymbol{w}_{k+P}^{(m)}\}_{m=1}^M$ are *identical* on all sensor nodes as $\{\boldsymbol{s}^{(m)}\}_{m=1}^M$ and $\mathbf{D}_k$ are identical for all nodes. However, depending on the to be performed measurement updates, the correlation samples $\{\boldsymbol{c}_{k|k}^{(i,m)}\}_{m=1}^M$ individually change over time for each sensor node.

As already mentioned, the correlations between locally obtained state estimates will be reconstructed at the fusion center by simply computing sample cross-covariance matrices (more on that in Section 4.2.4). Thus, due to the identical correlation samples $\{\boldsymbol{c}_{k|k}^{(i,m)}\}_{m=1}^M$, after a re-initialization the correlation between the i^{th} and j^{th} sensor node is

$$\mathbf{P}_{k|k}^{(i,j)} = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{c}_{k|k}^{(i,m)} \big(\boldsymbol{c}_{k|k}^{(j,m)}\big)^\top = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{c}_{k|k}^{(i,m)} \big(\boldsymbol{c}_{k|k}^{(i,m)}\big)^\top = \mathbf{P}_{k|k} \quad , \quad \forall i \neq j \quad .$$

Consequently, all the just re-initialized local state estimates are *fully* correlated. This makes sense as all sensor nodes have identical information about the current state estimate at time step k, i.e., the latest global estimate $\hat{\boldsymbol{x}}_{k|k}$ and $\mathbf{P}_{k|k}$.

After the re-initialization is done, each sensor node can perform alternating time updates and measurement updates as usual in Kalman filtering. However, our sample-based fusion approach additionally requires to predict and update the correlation samples as described next.

4.2.2 Time Update

First, the time update of the i^{th} sensor node predicts its local estimate $\hat{\boldsymbol{x}}^{(i)}_{k-1|k-1}$ and $\mathbf{P}^{(i)}_{k-1|k-1}$ from time step $k-1$ to time step k. Due to the linear system model (4.4), this boils down to the standard Kalman filter prediction formulas given by

$$\begin{aligned}\hat{\boldsymbol{x}}^{(i)}_{k|k-1} &= \mathbf{A}_k \hat{\boldsymbol{x}}^{(i)}_{k-1|k-1} \ , \\ \mathbf{P}^{(i)}_{k|k-1} &= \mathbf{A}_k \mathbf{P}^{(i)}_{k-1|k-1} \mathbf{A}_k^\top + \mathbf{B}_k \mathbf{Q}_k \mathbf{B}_k^\top \ .\end{aligned} \tag{4.9}$$

Second, we have to predict the correlation samples $\{\boldsymbol{c}^{(i,m)}_{k-1|k-1}\}_{m=1}^M$ to the next time step. The prediction utilizes the system noise sample set $\{\boldsymbol{w}^{(m)}_k\}_{m=1}^M$ that was already generated during the sensor node re-initialization. With this, we individually propagate each correlation sample in time according to

$$\boldsymbol{c}^{(i,m)}_{k|k-1} = \mathbf{A}_k \boldsymbol{c}^{(i,m)}_{k-1|k-1} + \mathbf{B}_k \boldsymbol{w}^{(m)}_k \ , \quad \forall m \in \{1, \dots, M\} \ . \tag{4.10}$$

Keep in mind that the noise sample set $\{\boldsymbol{w}^{(m)}_k\}_{m=1}^M$ is identical on all nodes. Thus, as the most important aspect of our sample-based fusion approach, this takes the common system noise into account during the prediction and allows for the exact correlation reconstruction later at the fusion center.

4.2.3 Measurement Update

Analogous to the time update, the i^{th} senor node first updates its local state estimate based on the received measurement $\tilde{\boldsymbol{y}}^{(i)}_k$ and the linear measurement model (4.5) in form of the standard Kalman filter measurement update according to

$$\begin{aligned}\mathbf{K}^{(i)}_k &= \mathbf{P}^{(i)}_{k|k-1} (\mathbf{H}^{(i)}_k)^\top \Big(\mathbf{H}^{(i)}_k \mathbf{P}^{(i)}_{k|k-1} (\mathbf{H}^{(i)}_k)^\top + \mathbf{R}^{(i)}_k \Big)^{-1} \ , \\ \hat{\boldsymbol{x}}^{(i)}_{k|k} &= \hat{\boldsymbol{x}}^{(i)}_{k|k-1} + \mathbf{K}^{(i)}_k \big(\tilde{\boldsymbol{y}}^{(i)}_k - \mathbf{H}^{(i)}_k \hat{\boldsymbol{x}}^{(i)}_{k|k-1} \big) \ , \\ \mathbf{P}^{(i)}_{k|k} &= \big(\mathbf{I}_N - \mathbf{K}^{(i)}_k \mathbf{H}^{(i)}_k \big) \mathbf{P}^{(i)}_{k|k-1} \ .\end{aligned} \tag{4.11}$$

Second, we also use the Kalman gain $\mathbf{K}^{(i)}_k$ to update the correlation samples $\{\boldsymbol{c}^{(i,m)}_{k|k-1}\}_{m=1}^M$ according to

$$\boldsymbol{c}^{(i,m)}_{k|k} = \big(\mathbf{I}_N - \mathbf{K}^{(i)}_k \mathbf{H}^{(i)}_k \big) \boldsymbol{c}^{(i,m)}_{k|k-1} \ , \quad \forall m \in \{1, \dots, M\} \ . \tag{4.12}$$

Note that, in contrast to the time update, we do not have any measurement noise samples involved. This is due to the fact that the measurement noises for different sensor nodes are assumed to be mutually independent. Also the actual measurement $\tilde{\boldsymbol{y}}^{(i)}_k$ is not required as it has no influence on the correlations between local estimates.

If no measurement is available at time step k, we simply set

$$\begin{aligned}\hat{\boldsymbol{x}}^{(i)}_{k|k} &= \hat{\boldsymbol{x}}^{(i)}_{k|k-1} \ , \\ \mathbf{P}^{(i)}_{k|k} &= \mathbf{P}^{(i)}_{k|k-1} \ ,\end{aligned}$$

and

$$\boldsymbol{c}_{k|k}^{(i,m)} = \boldsymbol{c}_{k|k-1}^{(i,m)} \ , \quad \forall m \in \{1, \ldots, M\} \ ,$$

i.e., the update estimate equals the predicted one.

4.2.4 Optimal Fusion

Now, we assume that all sensor nodes have conducted P predictions and maybe measurement updates in between since the last re-initialization at time step k, i.e., the current time step is $k+P$. It is important to note that at this point a sensor node cannot perform further predictions as no set of system noise samples is available for prediction of the correlation samples[3]. Thus, we are forced to fuse the locally obtained state estimates at the fusion center. Hence, all sensor nodes send their current local estimate $\hat{\boldsymbol{x}}_{k+P|k+P}^{(i)}$ and $\mathbf{P}_{k+P|k+P}^{(i)}$ and correlation samples $\{\boldsymbol{c}_{k+P|k+P}^{(i,m)}\}_{m=1}^{M}$ to the fusion center.

Here, we first build the joint mean $\hat{\boldsymbol{m}}_{k+P|k+P}$ out of the local state means $\hat{\boldsymbol{x}}_{k+P|k+P}^{(i)}$. Second, we build the joint covariance matrix $\mathbf{J}_{k+P|k+P}$ based on the local state covariance matrices $\mathbf{P}_{k+P|k+P}^{(i)}$ and the correlation matrices of the local estimates that are simply computed according to

$$\mathbf{P}_{k+P|k+P}^{(i,j)} = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{c}_{k+P|k+P}^{(i,m)} \big(\boldsymbol{c}_{k+P|k+P}^{(j,m)}\big)^{\top} \ , \quad \forall i \neq j \ . \tag{4.13}$$

Finally, we can fuse the local state estimates using (4.3) and get the global estimate $\hat{\boldsymbol{x}}_{k+P|k+P}$ and $\mathbf{P}_{k+P|k+P}$. In Appendix B, a proof is given that confirms that this sample-based fusion procedure indeed is optimal in the sought after weighted least squares sense. However, it is important to note that for a correct reconstruction the ordering of the correlation samples and system noise samples is essential, i.e., the ordering has to be the same on all nodes to be able to exactly reconstruct the correlations!

At this point, an iteration of the sample-based distributed state estimation is completed. The next iteration is initiated by sending the new global estimate $\hat{\boldsymbol{x}}_{k+P|k+P}$ and $\mathbf{P}_{k+P|k+P}$ to each sensor node in order to re-initialize local estimates, correlation samples, and system noise samples.

4.2.5 Summary

Algorithm 4.2 summarizes the proposed optimal sample-based distributed state estimation for linear systems in case of L sensor nodes. First, a global initial state estimate is set at the fusion center (line 1). Then, an iteration of the distributed state estimation starts with data transfer from fusion center to sensor nodes (line 3). The nodes are (re-)initialized (lines 4–7) and then perform P alternating state predictions and measurement updates (lines 8–13). After these P time steps, each node sends its local estimate to the fusion center (lines 14–16). Based on these, the current iteration ends at the fusion center by computing a new global state estimate (lines 17–18).

In addition to Algorithm 4.2, the data flow is sketched in Figure 4.2. From both it can be seen that communication only happens every P time steps. Moreover, as data is solely transferred between sensor nodes and fusion center, this implies that a sensor node does not get any information about the measurement processing from other others, i.e., when measurements are processed and which measurement models are used.

What still has to be discussed is the role of the user-defined parameter P. Its choice mainly depends on two factors: (i) how often a communication is required between fusion center and nodes, i.e., how often a global state estimate has to be provided and (ii) how much sample data can be transferred. The latter

[3]Of course, this in turn also prevents a sensor node to process measurements from future time steps.

```
1:  Set initial state estimate $\hat{\boldsymbol{x}}_{0|0}$ and $\mathbf{P}_{0|0}$ at the fusion center
2:  for $k = 0, P, 2P, \ldots$
3:      Fusion center sends $\hat{\boldsymbol{x}}_{k|k}$ and $\mathbf{P}_{k|k}$ to all nodes
4:      for $1 \leq i \leq L$
5:          $i^{\text{th}}$ node sets local state estimate according to (4.6)
6:          $i^{\text{th}}$ node computes samples according to (4.8)
7:      end for
8:      for $1, \ldots, P$
9:          for $1 \leq i \leq L$
10:             $i^{\text{th}}$ node performs state prediction (4.9) and (4.10)
11:             $i^{\text{th}}$ node performs measurement update (4.11) and (4.12)
12:         end for
13:     end for
14:     for $1 \leq i \leq L$
15:         $i^{\text{th}}$ node sends $\hat{\boldsymbol{x}}^{(i)}_{k+P|k+P}$, $\mathbf{P}^{(i)}_{k+P|k+P}$, and $\{\boldsymbol{c}^{(i,m)}_{k+P|k+P}\}_{m=1}^{M}$ to fusion center
16:     end for
17:     Fusion center computes $\mathbf{P}^{(i,j)}_{k+P|k+P}$, $i \neq j$, with (4.13)
18:     Fusion center computes $\hat{\boldsymbol{x}}_{k+P|k+P}$ and $\mathbf{P}_{k+P|k+P}$ with (4.1)–(4.3)
19: end for
```

Algorithm 4.2: Optimal sample-based distributed state estimation.

comes from the fact that the larger P the larger will be the joint space of state and system noise $\boldsymbol{d}_k$ in (4.7). This in turn means that more correlation samples are required, as the number of samples M depends on the joint space dimension D. Fortunately, the data consumption of the correlation samples $\{\boldsymbol{c}^{(i,m)}_{k|k}\}_{m=1}^{M}$ does *not* depend on the number of utilized sensor nodes, which makes this approach very well-suited for large sensor networks. In particular, for a fixed state dimension N and a fixed system noise dimension W, i.e., the common case of a constant number of state variables and time-invariant system noise, the data consumption of the correlation samples $\{\boldsymbol{c}^{(i,m)}_{k|k}\}_{m=1}^{M}$ is

$$\dim\left(\boldsymbol{c}^{(i,m)}_{k|k}\right) \cdot M = N \cdot (D+1) = N^2 + N \cdot (1 + W \cdot P) \ .$$

That is, it increases only linearly in P.

Note that a fusion of local estimates and subsequent re-initialization of the sensor nodes can also be performed if less than P predictions/measurement updates were performed by the sensor nodes. In fact, P is only an upper bound for the time span between consecutive communications. It is also straight forward to change P over time. The fusion center can simply send the currently selected P together with the latest fused state estimate at the next re-initialization. In doing so, each sensor node can compute the correlation samples and system noise samples using the just received P and also knows when the next communication is required.

In summary, the proposed sample-based distributed state estimation offers several advantages:

- we can exactly reconstruct the correlations needed for the weighted least squares fusion,
- a sensor node neither needs any information about the number of processed measurements at other nodes nor their used measurement models (this is especially important for nonlinear state estimation, see Section 4.3),
- the additional communication overhead are the correlation samples, which growth only linearly in the number of predictions P,

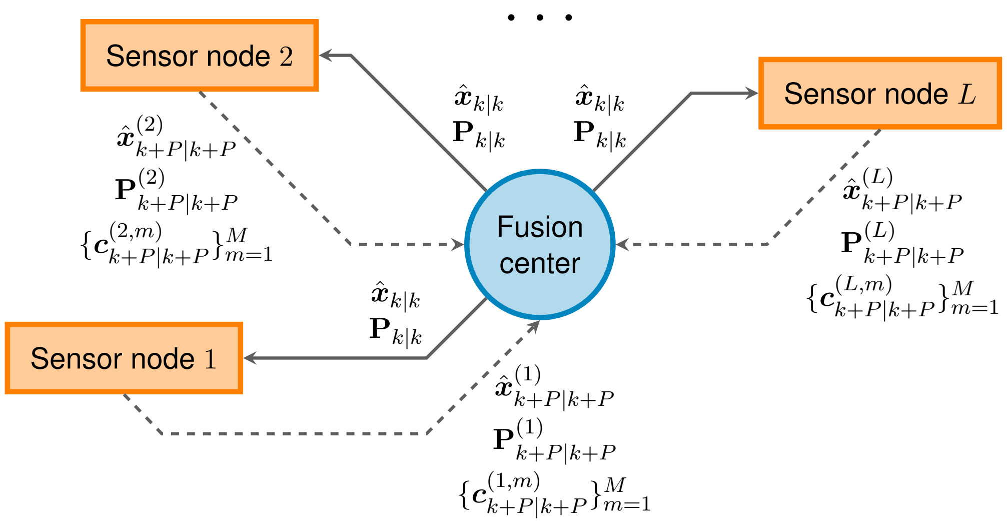

Sensor node 2
Sensor node L
Sensor node 1
Fusion center
$\hat{\boldsymbol{x}}_{k|k}$ $\mathbf{P}_{k|k}$
$\hat{\boldsymbol{x}}_{k|k}$ $\mathbf{P}_{k|k}$
$\hat{\boldsymbol{x}}_{k|k}$ $\mathbf{P}_{k|k}$
$\hat{\boldsymbol{x}}^{(2)}_{k+P|k+P}$ $\mathbf{P}^{(2)}_{k+P|k+P}$ $\{\boldsymbol{c}^{(2,m)}_{k+P|k+P}\}^{M}_{m=1}$
$\hat{\boldsymbol{x}}^{(L)}_{k+P|k+P}$ $\mathbf{P}^{(L)}_{k+P|k+P}$ $\{\boldsymbol{c}^{(L,m)}_{k+P|k+P}\}^{M}_{m=1}$
$\hat{\boldsymbol{x}}^{(1)}_{k+P|k+P}$ $\mathbf{P}^{(1)}_{k+P|k+P}$ $\{\boldsymbol{c}^{(1,m)}_{k+P|k+P}\}^{M}_{m=1}$

Figure 4.2: Required communication between fusion center and senor nodes over time: the fusion center sends the latest global estimate to all nodes at time step k, while the sensor nodes send their local estimates and correlation samples to the fusion center at time step $k + P$ (cf. [182]).

- it is well-suited for large sensor networks as the number of correlation samples is independent of the number of utilized sensor nodes, and finally
- additional nodes can be added over time by simply sending the latest estimate to the new node.

Compared to the DKF, our proposed approach provides meaningful local state estimates. Those can be required, for example, by other procedures operating on a sensor node. Furthermore, the DKF will not produce correct estimates if data from the sensor node get lost, e.g., due to network issues. The problem in this case is that the a priori made assumptions, e.g., the number of processed measurements, are no more valid. The federated Kalman filter suffers from a similar problem: if the number of local estimates be fused is less than expected, the system noise covariance was inflated too much on each sensor node, which results in a too pessimistic global estimate. As opposed to this, our approach will even work if not all local state estimates arrive at the fusion center as expected. In such a case, we simply fuse those local estimates that are available. Of course, measurement information from the missing sensor nodes is lost. However, the conducted fusion will still be optimal in the weighted least squares sense.

Finally, we want to point out that the processing of the correlation samples resembles the workflow of the ensemble Kalman filter (EnKF), e.g., see [63]. The EnKF is a particle filter for nonlinear state estimation, see also Section 5.1. This means in particular that its state estimate is a set of equally weighted samples (rather than a mean vector and a covariance matrix), which is modified over time by prediction and measurement update. On the one hand, the state prediction of the EnKF is very similar to the prediction of our correlation samples except for the used system noise samples. The only difference is that the EnKF uses noise samples that are randomly drawn according to the system noise characteristics. However, those random samples do neither necessarily reflect the correct mean and covariance matrix of the system noise nor guarantee a white sequence of system noise samples, which are also independent of the system state. In contrast, due to the taken sample computation in (4.8), we satisfy all these requirements (which is the reason for the exact correlation reconstruction). On the other hand, the measurement update of EnKF requires the measurement $\tilde{\boldsymbol{y}}^{(i)}_k$ and measurement noise samples. Both are needed as the samples of the EnKF are the actual state estimate, which of course depends upon the measurement and its noise.

4.3 Optimal Fusion for Distributed Nonlinear State Estimation

Up to now, we only considered linear systems. However, our goal is still to apply the S^{2}KF to a nonlinear estimation problem in a distributed setup. Thus, we now assume a general nonlinear measurement model

$$\boldsymbol{y}_k^{(i)} = \boldsymbol{h}_k^{(i)}(\boldsymbol{x}_k, \boldsymbol{v}_k^{(i)}) \ , \tag{4.14}$$

where the measurement noise $\boldsymbol{v}_k^{(i)}$ is defined as in Section 4.2. The key idea to make the nonlinear model (4.14) applicable for the proposed sample-based distributed state estimation is to find an appropriate affine approximation

$$\boldsymbol{y}_k^{(i)} \approx \mathbf{H}_k^{(i)} \boldsymbol{x}_k + \boldsymbol{b}_k^{(i)} + \boldsymbol{\varepsilon}_k^{(i)} \ , \tag{4.15}$$

with zero-mean additive Gaussian noise $\boldsymbol{\varepsilon}_k^{(i)}$ with covariance matrix $\boldsymbol{\Xi}_k^{(i)}$. In other words, we have to find suitable $\mathbf{H}_k^{(i)}$, $\boldsymbol{b}_k^{(i)}$, and $\boldsymbol{\Xi}_k^{(i)}$. Once an approximation is found, we can directly apply the proposed sample-based fusion technique for linear systems.

An optimal affine approximation in the mean square error (MSE) sense, e.g., see [40], is obtained by computing first-order and second-order moments of (4.14). That is, we need to compute the measurement mean

$$\hat{\boldsymbol{y}}_k^{(i)} = \int_{\mathbb{R}^N} \int_{\mathbb{R}^{V_k}} \boldsymbol{h}_k^{(i)}(\boldsymbol{x}_k, \boldsymbol{v}_k^{(i)}) \cdot \mathcal{N}\left(\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{v}_k^{(i)} \end{bmatrix}; \begin{bmatrix} \hat{\boldsymbol{x}}_{k|k-1}^{(i)} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{k|k-1}^{(i)} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_k^{(i)} \end{bmatrix} \right) \mathrm{d}\boldsymbol{v}_k^{(i)} \, \mathrm{d}\boldsymbol{x}_k \ , \tag{4.16}$$

the measurement covariance matrix

$$\begin{aligned} \mathbf{Y}_k^{(i)} = \int_{\mathbb{R}^N} \int_{\mathbb{R}^{V_k}} & \big(\boldsymbol{h}_k^{(i)}(\boldsymbol{x}_k, \boldsymbol{v}_k^{(i)}) - \hat{\boldsymbol{y}}_k^{(i)}\big)\big(\boldsymbol{h}_k^{(i)}(\boldsymbol{x}_k, \boldsymbol{v}_k^{(i)}) - \hat{\boldsymbol{y}}_k^{(i)}\big)^\top \cdot \\ & \mathcal{N}\left(\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{v}_k^{(i)} \end{bmatrix}; \begin{bmatrix} \hat{\boldsymbol{x}}_{k|k-1}^{(i)} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{k|k-1}^{(i)} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_k^{(i)} \end{bmatrix} \right) \mathrm{d}\boldsymbol{v}_k^{(i)} \, \mathrm{d}\boldsymbol{x}_k \ , \end{aligned} \tag{4.17}$$

and the cross-covariance matrix of state and measurement

$$\begin{aligned} \mathbf{C}_k^{(i)} = \int_{\mathbb{R}^N} \int_{\mathbb{R}^{V_k}} & \big(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}^{(i)}\big)\big(\boldsymbol{h}_k^{(i)}(\boldsymbol{x}_k, \boldsymbol{v}_k^{(i)}) - \hat{\boldsymbol{y}}_k^{(i)}\big)^\top \cdot \\ & \mathcal{N}\left(\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{v}_k^{(i)} \end{bmatrix}; \begin{bmatrix} \hat{\boldsymbol{x}}_{k|k-1}^{(i)} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{k|k-1}^{(i)} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_k^{(i)} \end{bmatrix} \right) \mathrm{d}\boldsymbol{v}_k^{(i)} \, \mathrm{d}\boldsymbol{x}_k \ . \end{aligned} \tag{4.18}$$

With these, we get the affine approximation

$$\begin{aligned} \mathbf{H}_k^{(i)} &= \big(\mathbf{C}_k^{(i)}\big)^\top \big(\mathbf{P}_{k|k-1}^{(i)}\big)^{-1} \ , \\ \boldsymbol{b}_k^{(i)} &= \hat{\boldsymbol{y}}_k^{(i)} - \mathbf{H}_k^{(i)} \hat{\boldsymbol{x}}_{k|k-1}^{(i)} \ , \\ \boldsymbol{\Xi}_k^{(i)} &= \mathbf{Y}_k^{(i)} - \mathbf{H}_k^{(i)} \mathbf{P}_{k|k-1}^{(i)} \big(\mathbf{H}_k^{(i)}\big)^\top \ . \end{aligned} \tag{4.19}$$

The above integrals can be computed with the aid of the LCD-based Gaussian sampling scheme from Chapter 2, i.e., the S^{2}KF, or with any other linearization technique used by Kalman filters, e.g., Taylor series expansion. In fact, the described linearization procedure, that is, finding the affine approximation (4.15), is implicitly taken by all Kalman filters applied to nonlinear measurement models (the moment formulas (4.16)–(4.18) are identical to those from Section 3.1.2)!

By plugging the linear approximation (4.19) into the measurement update formulas from Section 4.2.3, the update of the local state estimate and the correlation samples becomes

$$\begin{aligned}
\mathbf{K}_k^{(i)} &= \mathbf{C}_k^{(i)} \big(\mathbf{Y}_k^{(i)}\big)^{-1} \ , \\
\hat{\boldsymbol{x}}_{k|k}^{(i)} &= \hat{\boldsymbol{x}}_{k|k-1}^{(i)} + \mathbf{K}_k^{(i)} \big(\tilde{\boldsymbol{y}}_k^{(i)} - \hat{\boldsymbol{y}}_k^{(i)}\big) \ , \\
\mathbf{P}_{k|k}^{(i)} &= \big(\mathbf{I}_N - \mathbf{K}_k^{(i)} \mathbf{H}_k^{(i)}\big) \mathbf{P}_{k|k-1}^{(i)} \ , \\
\boldsymbol{c}_{k|k}^{(i,m)} &= \big(\mathbf{I}_N - \mathbf{K}_k^{(i)} \mathbf{H}_k^{(i)}\big) \boldsymbol{c}_{k|k-1}^{(i,m)} \ , \quad \forall m \in \{1, \ldots, M\} \ .
\end{aligned}$$

It is important to note that each sensor node linearizes its measurement model around its own local estimate. In particular, this means that even if the same nonlinear measurement model is used for each node, the obtained linearizations are, in general, not identical. This, however, imposes no problem for our proposed sample-based fusion approach as neither the fusion center nor other sensor nodes need to know the linearized measurement models. Hence, there is no need to communicate any information about the linearized measurement models over the network.

Regarding the DKF, such a local linearization approach would not be possible: the required local estimate is simply not at hand for a linearization. Of course, a sensor node could run a local Kalman filter in parallel solely for the linearization. Nonetheless, then all nodes would still be forced to send their linearized models to all other nodes in order allow for a proper processing of the measurement data. Moreover, this also would no more be equivalent to a centralized fusion as the linearizations are not based on the global estimate.

4.4 Evaluation

As a first evaluation of the proposed sample-based distributed state estimation, we consider a target tracking scenario using several sensors and linear models in order to experimentally verify that our approach indeed can (i) reconstruct the true correlations between local estimates and (ii) outperform existing distributed estimation techniques. In the second evaluation, we extend the first one by replacing the linear position measurements with nonlinear distance measurements. Finally, we take up the cylinder tracking scenario from Chapter 3 and show that it is even possible to estimate pose and shape of the cylinder in a distributed way when applying the proposed sample-based fusion technique.

4.4.1 Distributed Target Tracking Based on Position Measurements

We consider tracking a target in the xy-plane with the aid of several spatially distributed sensor nodes. The target's system state consists of

- position $\boldsymbol{p}_k = [p_k^{(\mathrm{x})}, p_k^{(\mathrm{y})}]^\top$ in m,
- velocity $\dot{\boldsymbol{p}}_k = [\dot{p}_k^{(\mathrm{x})}, \dot{p}_k^{(\mathrm{y})}]^\top$ in $\mathrm{m/s}$, and
- acceleration $\ddot{\boldsymbol{p}}_k = [\ddot{p}_k^{(\mathrm{x})}, \ddot{p}_k^{(\mathrm{y})}]^\top$ in $\mathrm{m/s^2}$,

that is, we have $\boldsymbol{x}_k = [\boldsymbol{p}_k^\top, \dot{\boldsymbol{p}}_k^\top, \ddot{\boldsymbol{p}}_k^\top]^\top$. In addition, we describe the target's motion with a constant acceleration model [11, Sec. 6.3.3] according to

$$\boldsymbol{x}_k = \mathbf{A}\boldsymbol{x}_{k-1} + \mathbf{B}\boldsymbol{w} \ , \tag{4.20}$$

with matrices

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_2 & \Delta t\, \mathbf{I}_2 & \frac{1}{2}\Delta t^2\, \mathbf{I}_2 \\ \mathbf{0} & \mathbf{I}_2 & \Delta t\, \mathbf{I}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_2 \end{bmatrix} \ , \quad \mathbf{B} = \begin{bmatrix} \frac{1}{2}\Delta t^2\, \mathbf{I}_2 \\ \Delta t\, \mathbf{I}_2 \\ \mathbf{I}_2 \end{bmatrix} \ ,$$

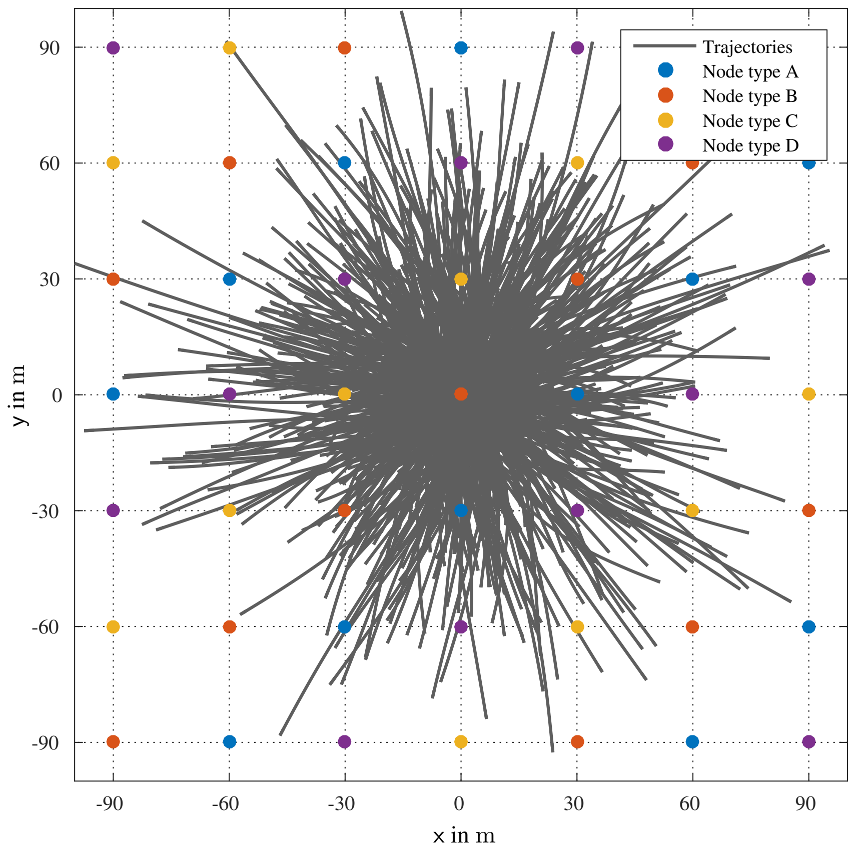


Figure 4.3: Target tracking evaluation with using 49 sensor nodes.

time period $\Delta t = 0.05\,\mathrm{s}$, and time-invariant zero-mean white Gaussian noise $\boldsymbol{w}$ with covariance matrix $\mathbf{Q} = 0.3\,\mathbf{I}_2$.

While the target moves, it is observed by $L = 49$ sensors nodes located on a regular grid, see Figure 4.3. Moreover, the sensor node located at $(0, 0)$ also acts as fusion center. At time step k, each node measures the current position of the target according to the linear measurement model

$$\boldsymbol{y}_k^{(i)} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \boldsymbol{x}_k + \boldsymbol{v}^{(i)} \ , \tag{4.21}$$

where $\boldsymbol{v}^{(i)}$ denotes zero-mean white Gaussian noise. The measurement noise, however, differs between the nodes. In fact, we have four different node types “A”, “B”, “C”, and “D” with respective measurement noise covariance matrices $\mathbf{R}^{(\mathrm{A})} = \mathbf{I}_2$, $\mathbf{R}^{(\mathrm{B})} = 4\,\mathbf{I}_2$, $\mathbf{R}^{(\mathrm{C})} = 0.5\,\mathbf{I}_2$, and $\mathbf{R}^{(\mathrm{D})} = 2\,\mathbf{I}_2$.

We evaluate the following fusion techniques:

- naive fusion assuming uncorrelated estimates,
- covariance intersection that minimizes the trace of $\mathbf{P}_{k|k}$,
- the optimal weighted least squares fusion based on the true correlations,
- the proposed sample-based fusion approach, and
- a central Kalman filter as baseline that processes the measurements from all nodes.

Every fifth time step, the fusion center fuses all 49 local estimates to a global estimate, i.e., the fusion interval is $P = 5$. After a fusion, all local estimates are reset to just obtained global estimate (individually for each investigated fusion technique, of course).

We perform $R = 1\,000$ Monte Carlo runs. In each run, the initial global estimate is set to

$$\begin{aligned}\hat{\boldsymbol{x}}_{0|0} &= [0, 0, 0, 0, 0, 0]^\top \ , \\ \mathbf{P}_{0|0} &= \mathrm{diag}(10\,\mathbf{I}_2, \mathbf{I}_2, 10^{-1}\,\mathbf{I}_2) \ .\end{aligned}$$

Furthermore, we simulate the true system state $\bar{\boldsymbol{x}}_k$ over 100 time steps. The initial state $\bar{\boldsymbol{x}}_0$ is drawn from $\mathcal{N}(\hat{\boldsymbol{x}}_{0|0}, \mathbf{P}_{0|0})$ and propagated in time using the system model (4.20) together with random realizations of $\boldsymbol{w}$. In addition, for each time step we simulate a noisy position measurement per sensor node according to (4.21) and the respective measurement noise covariances. All the simulated target trajectories are depicted in Figure 4.3 as well.

Over all simulation runs, we compute for all fusion techniques the respective position RMSE, velocity RMSE, and acceleration RMSE of the estimate obtained at the fusion center, see Figures 4.4(a)–4.4(c). While the central Kalman filter yields the smallest errors and serves as lower bound, the naive approach diverges directly after the first fusion operation at time step $k = 5$. All other fusion techniques exhibit sawtooth-shaped error curves. This is due to the fact that the estimate from the fusion node is merely based on its local measurements for four time steps until the next fusion operation is conducted. As expected, the proposed sample-based fusion is identical the optimal WLS fusion, i.e., the sample-based approach is able to reconstruct the true correlation matrices at the fusion center. Furthermore, covariance intersection yields always worse results than the sample-based fusion.

Next, we have a look at the average normalized estimation error squared (NEES) [11, Sec. 5.4.2], see Figure 4.4(d). It is given by

$$\frac{1}{R}\sum_{r=1}^{R}(\bar{\boldsymbol{x}}_k^{(r)} - \hat{\boldsymbol{x}}_{k|k}^{(r)})^\top \big(\mathbf{P}_{k|k}^{(r)}\big)^{-1}(\bar{\boldsymbol{x}}_k^{(r)} - \hat{\boldsymbol{x}}_{k|k}^{(r)}) \ , \tag{4.22}$$

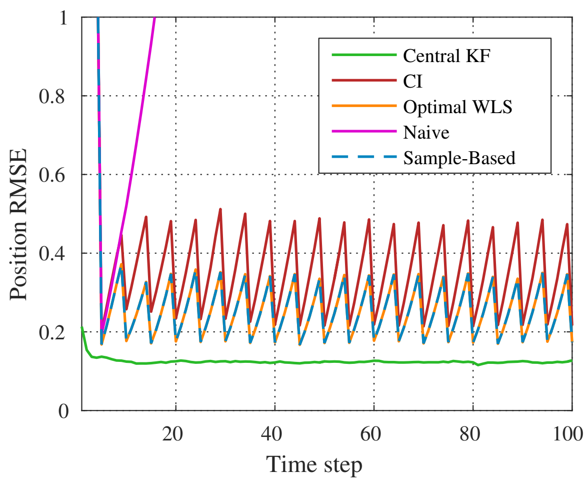
where $\bar{\boldsymbol{x}}_k^{(r)}$ denotes the true system state of the r^{th} simulation run, and $\hat{\boldsymbol{x}}_{k|k}^{(r)}$ and $\mathbf{P}_{k|k}^{(r)}$ the state estimate of the r^{th} simulation run and respective fusion technique. If an estimator/fusion technique is consistent[4], that is, it (i) fulfills unbiasedness and (ii) the covariance matrix computed by the estimator matches the actual MSE matrix of its state mean, the average NEES (4.22) converges to the state dimension for $R \to \infty$ (six in our case). It can be seen that this is only true for the central Kalman filter, the optimal WLS fusion, and the proposed sample-based fusion, but not for covariance intersection and the naive fusion approach, which even diverges completely. Thus, although covariance intersection is consistent in the sense of a pessimistic state covariance matrix, it is not consistent in the sense that the covariance matrix matches the MSE matrix. In contrast, the proposed sample-based fusion with its exact correlations can fulfill this, and again gives proof that it indeed yields the optimal WLS fusion results.

We further compare the average trace of the MSE matrix

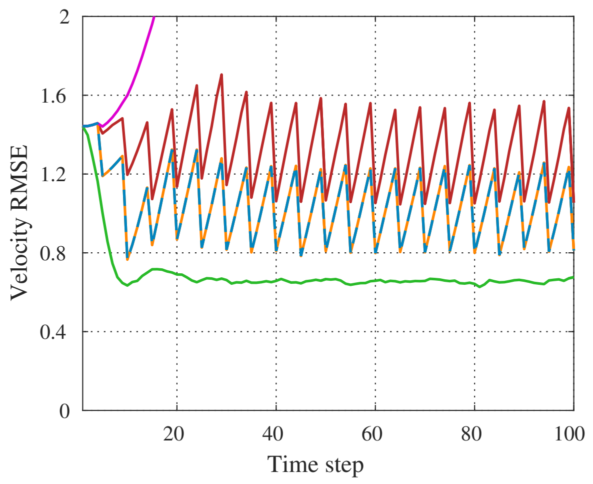
$$\frac{1}{R}\sum_{r=1}^{R}\left\|\bar{\boldsymbol{x}}_k^{(r)} - \hat{\boldsymbol{x}}_{k|k}^{(r)}\right\|_2^2 \ ,$$

i.e., the average MSE, shown in Figure 4.4(e) with the trace of $\mathbf{P}_{k|k}$, see Figure 4.4(f). This confirms the values of the average NEES. On the one hand, the MSE matrix of the covariance intersection estimate is smaller than its covariance matrix. On the other hand, MSE matrix and $\mathbf{P}_{k|k}$ are nearly the same for the central Kalman filter, the optimal WLS fusion, and the sample-based fusion. Moreover, covariance intersection is much more pessimistic than the sample-based fusion approach. Finally, the covariance of the naive fusion approach is much too small. This explains its divergence as such small covariance matrices let the estimator ignore the vital information from the received measurements.

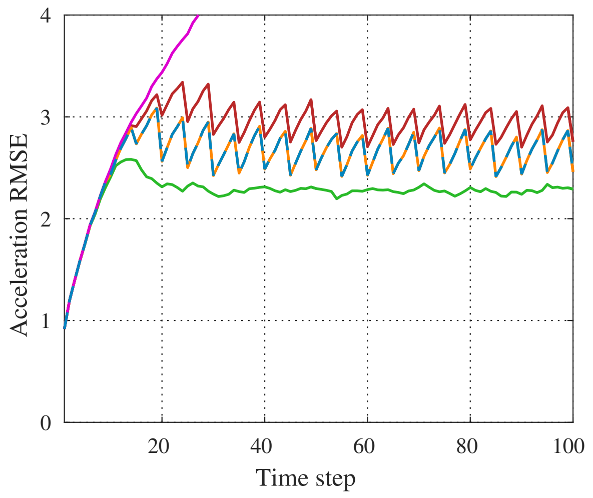
[4]Not to be confused with the consistency definition of a pessimistic covariance matrix approximation, e.g., done by covariance intersection.



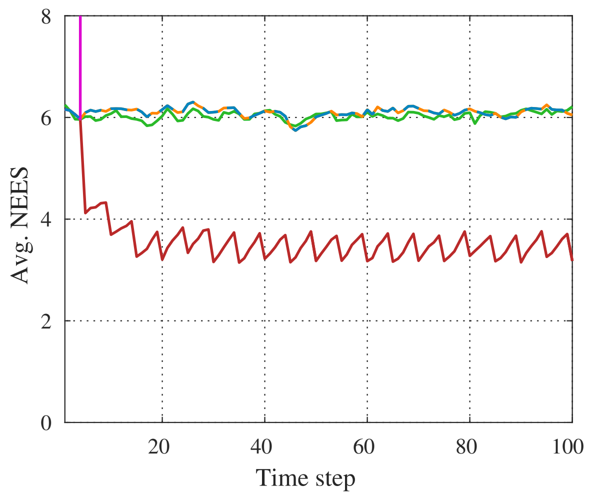
(a) Position RMSE in m.



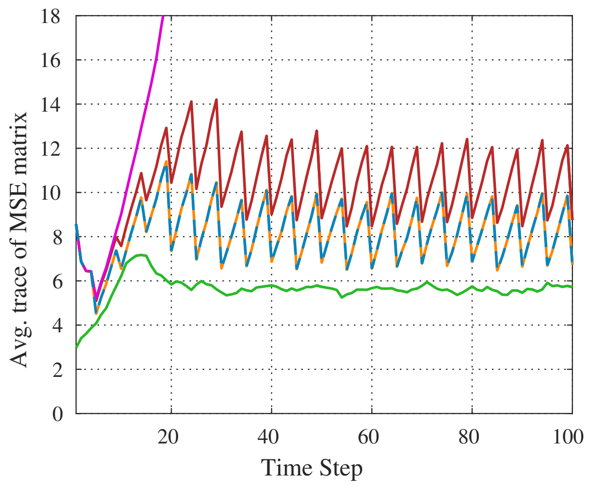
(b) Velocity RMSE in $\mathrm{m/s}$.



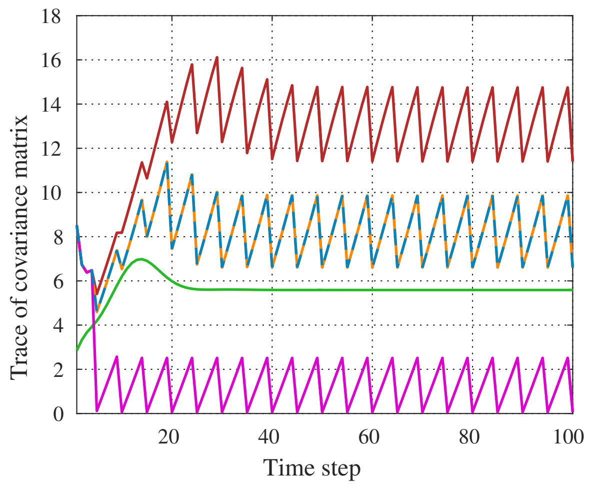
(c) Acceleration RMSE in $\mathrm{m/s^2}$.



(d) Average NEES.



(e) Average trace of MSE matrix.



(f) Trace of $\mathbf{P}_{k|k}$.

Figure 4.4: Results of the target tracking evaluation based on linear position measurements.

4.4.2 Distributed Target Tracking Based on Distance Measurements

In the previous evaluation, we demonstrated the effectiveness of the proposed sample-based fusion technique when dealing with linear measurement models. Now, our goal is to assess the quality of the sample-based correlation reconstruction when coping with nonlinear measurement models instead. Thus, we reconsider the same evaluation setup as used in the previous section except for the position measurements (4.21), which are replaced by distance measurements according to

$$y_k^{(i)} = h^{(i)}(\boldsymbol{x}_k) + v^{(i)} = \|\boldsymbol{p}_k - \boldsymbol{n}^{(i)}\|_2 + v^{(i)} \ ,$$

where $\boldsymbol{n}^{(i)}$ denotes the location of the i^{th} sensor node and $v^{(i)}$ zero-mean white Gaussian noise. Again, each of the four sensor node types has its individual noise level given by $\mathbf{R}^{(\text{A})} = 0.1$, $\mathbf{R}^{(\text{B})} = 1$, $\mathbf{R}^{(\text{C})} = 0.5$, and $\mathbf{R}^{(\text{D})} = 1.5$, respectively.

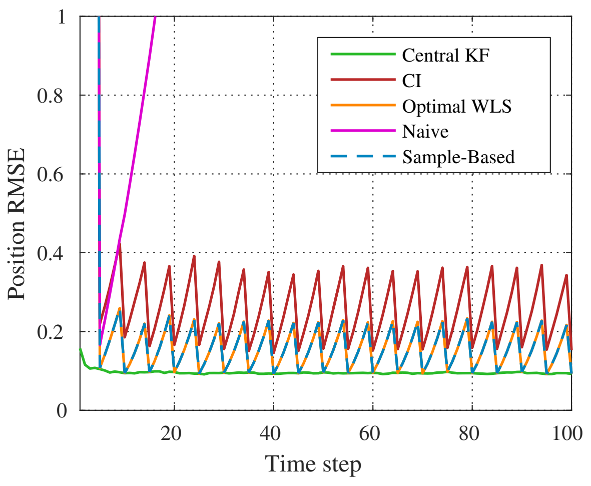
In order to locally process the measured distances, all nodes rely on the measurement update of the S^{2}KF that is configured to use 31 samples. Note that these samples refer to the point-symmetric LCD-based Gaussian sampling technique used for the moment computations (4.16)–(4.18), not the correlation samples that are automatically configured based on the selected number of prediction steps P. Also the central Kalman filter at the fusion center is replaced by a S^{2}KF that uses 31 samples per measurement update. In each time step, it processes the distance measurements from all 49 sensor nodes in a single filter step.

We perform again $R = 1\,000$ Monte Carlo runs. First, we analyze the tracking performance by comparing position RMSE, velocity RMSE, and acceleration RMSE of the distributed state estimation techniques, see Figures 4.5(a)–4.5(c). The results are very similar to those from the linear evaluation: the central Kalman filter gives a lower bound for the estimation quality, the naive fusion approach diverges directly after the first fusion process, optimal WLS fusion and sample-based fusion yield identical results, and covariance intersection is significantly worse than the latter approaches. However, the absolute error values are smaller compared to the previous evaluation. In addition, a just fused estimate of the sample-based fusion reaches the same low error levels as the central measurement processing.

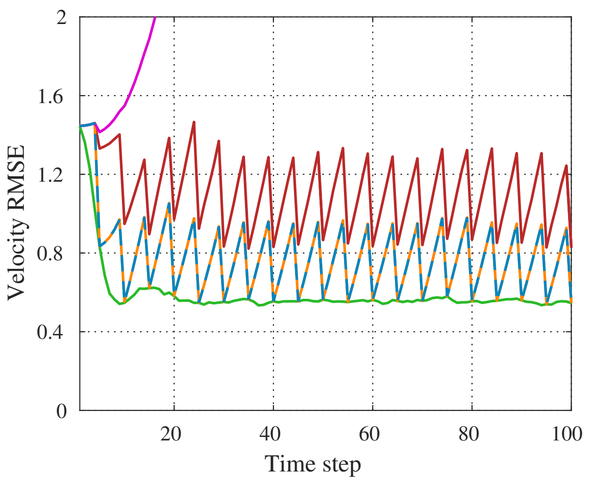
Second, the average NEES of the estimators are shown in Figure 4.5(d). As can be seen, they are nearly identical to those from the previous evaluation. Similar holds for the MSE (Figure 4.5(e)) and the state covariance matrix (Figure 4.5(f)). Note that due to the nonlinear measurements, the state covariance matrices now depend on the concrete measurements, and thus vary over the runs. Hence, we computed the average trace of $\mathbf{P}_{k|k}$ over all simulation runs. Like for the other errors, the absolute values are smaller compared to the first evaluation. Nevertheless, MSE and covariance matrix are still not the same for covariance intersection, which reflects the suboptimal values of its average NEES. Moreover, the sample-based fusion comes very close to the covariance matrix obtained by the central Kalman filter, at least just after a fusion was performed.

4.4.3 Distributed Tracking of a Cylinder in 3D

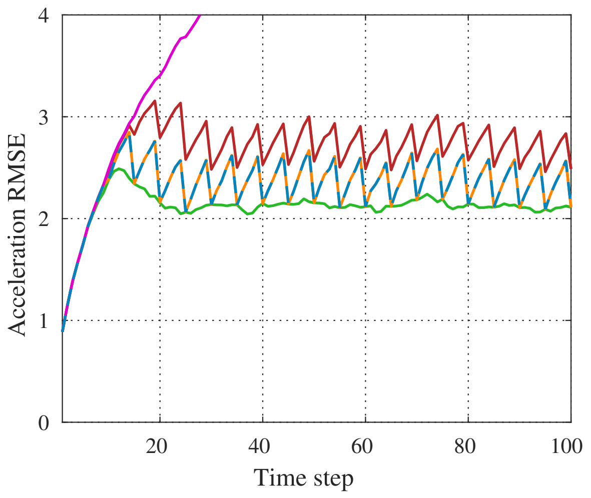
Finally, we reconsider the cylinder tracking from Section 3.4.2. More precisely, we keep the complete setup where the moving and rotating cylinder with a time-varying shape is observed by eight sensors with a limited field of view, see Figure 3.4. This also includes the 12D system state (3.23) consisting of pose, shape, and motion parameters, the constant velocity/turn rate system model (3.24) as well as the derived nonlinear random hypersurface measurement model (3.26). But, instead of processing the measurements from all sensors by a central estimator, we turn each sensor into a sensor node with local measurement processing.



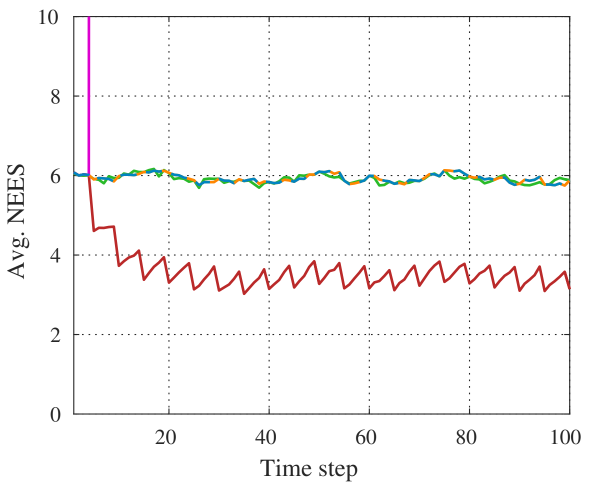
(a) Position RMSE in m.



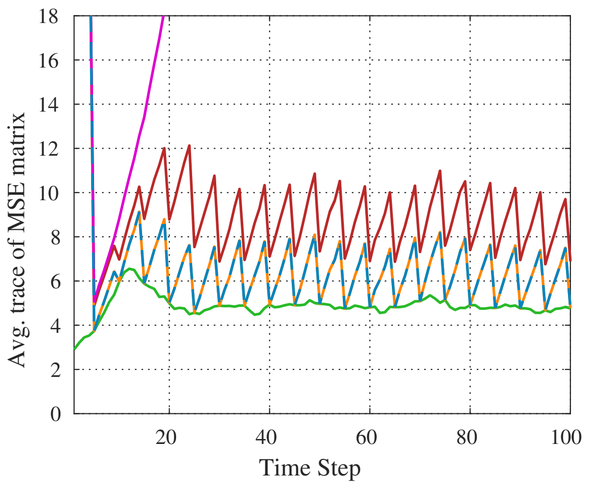
(b) Velocity RMSE in $\mathrm{m/s}$.



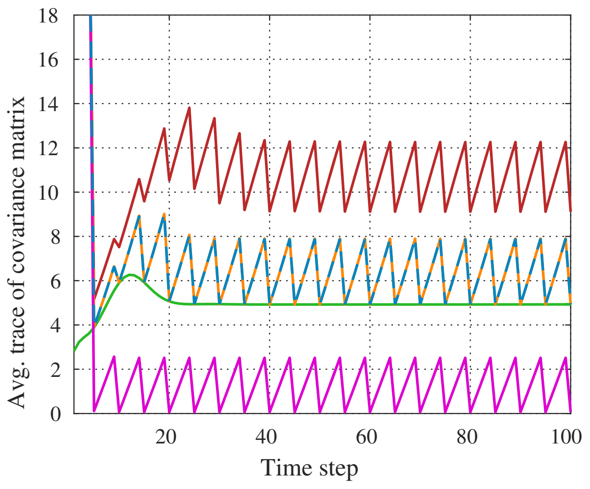
(c) Acceleration RMSE in $\mathrm{m/s^2}$.



(d) Average NEES.



(e) Average trace of MSE matrix.



(f) Average trace of $\mathbf{P}_{k|k}$.

Figure 4.5: Results of the target tracking evaluation based on nonlinear distance measurements.

In particular, the i^{th} node executes a S^{2}KF for locally estimating the cylinder. If the cylinder is in the sensor's field of view, the S^{2}KF processes the gained set of measurements $\mathcal{Y}_k^{(i)}$, consisting of $Y_k^{(i)}$ measured 3D points from the visible cylinder's surface. Like for the central processing of the measurements, a local S^{2}KF processes all the measurements $\mathcal{Y}_k^{(i)}$ in a single filter setup. Consequently, it has to deal with a joint space encompassing state and measurement noise variables of $J_k^{(i)} = 12 + Y_k^{(i)} \cdot 4$ dimensions. Thus, analogous to the central S^{2}KF, we configure the local S^{2}KF to use $10J_k^{(i)} + 1$ samples per measurement update. Again, this refers to the LCD-based Gaussian sampling scheme used for the moment computation, not the correlation samples.

We evaluate the following estimators:

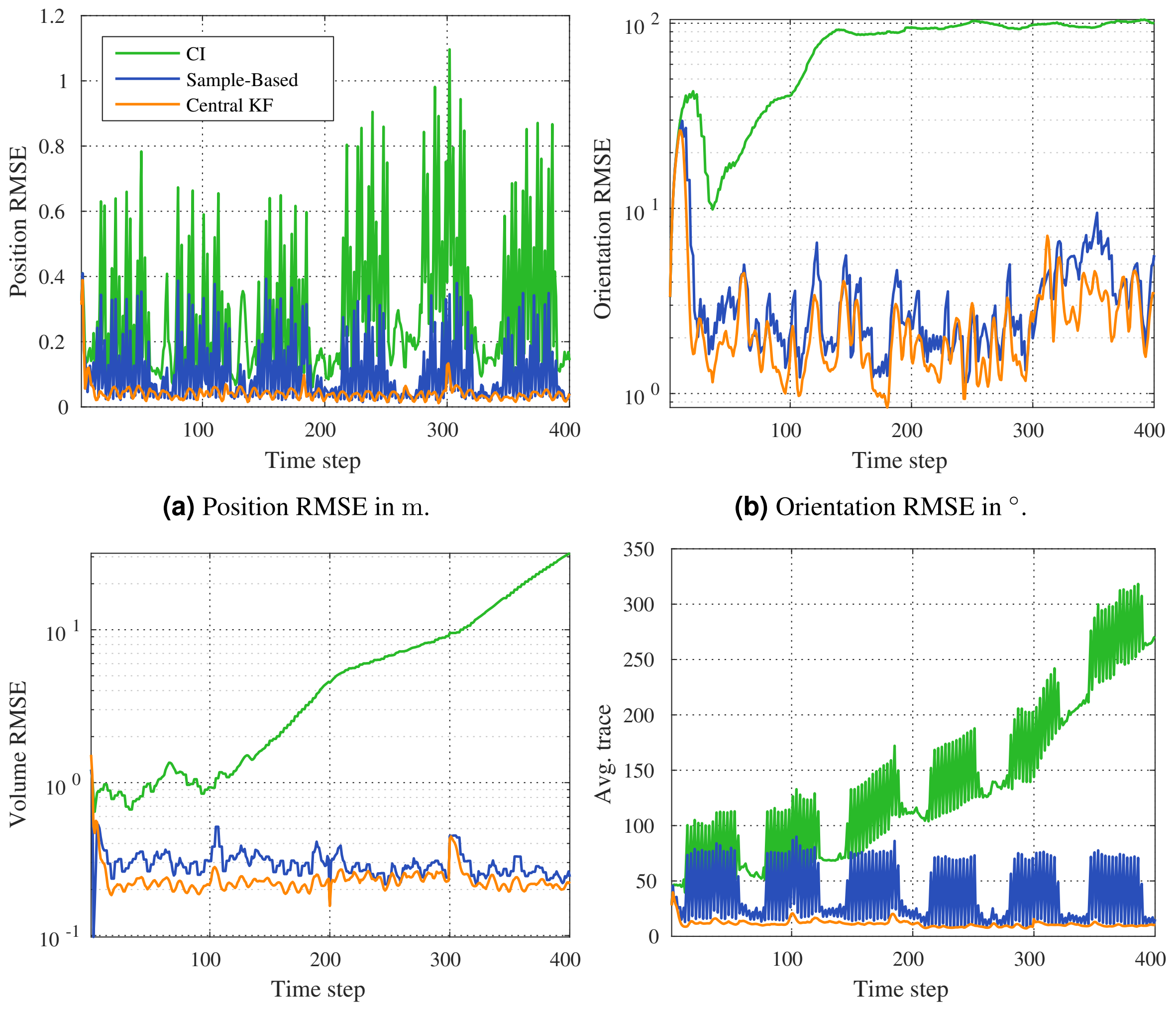
- covariance intersection that minimizes the trace of $\mathbf{P}_{k|k}$,
- the sample-based fusion approach, and
- the central S^{2}KF from the evaluation in Section 3.4.2 as lower bound for the estimation quality.

The first sensor node also acts a fusion center. Every third time step, both distributed approaches perform a fusion operation, i.e., the fusion interval is $P = 3$. As for the sample-based approach, covariance intersection resets the local estimates to a just obtained global estimate. We perform 100 Monte Carlo runs. In each run, the initial estimate $\hat{\underline{x}}_{0|0}$ and $\mathbf{P}_{0|0}$ for the central S^{2}KF is obtained in the same way as in the original evaluation based on the first sets of measurements available from all sensors. For the distributed approaches, however, the initial global estimate is obtained solely based on the measurements $\mathcal{Y}_0^{(1)}$ available at the fusion node.

First, we have a look at the position RMSE of the respective estimation techniques, see Figure 4.6(a). The distributed estimators have spiky error curves interrupted by longer phases of nearly constant errors. Those phases are present if the cylinder is in the fusion node's field of view. If it is not, the estimate at the fusion center is merely based on predictions, which is only corrected every third time step by a fusion operation. Furthermore, for the covariance intersection approach the overall position error is much larger than the sample-based approach and even growths over time. In contrast to this, the sample-based fusion keeps the position error very constant over time and is often close the results of the central S^{2}KF. Second, we consider the orientation RMSE (given by the angle between the true cylinder's z-axis and the z-axis of the estimated cylinder) shown in Figure 4.6(b). Here, we can see that covariance intersection is not able to correctly estimate the cylinder's orientation. In fact, it starts to diverge after the first time steps. As opposed to this, the sample-based fusion technique can deliver quite good orientation estimates, which are also very similar to the central measurement processing. The same can be observed for the volume RMSE depicted in Figure 4.6(c). From the beginning, covariance intersection possesses worse shape estimates than the other estimators and starts to diverge after approximately 100 time steps. The sample-based approach is again only slightly worse than the centrally operating S^{2}KF.

Next, we take a look at the average traces of the state covariance matrices shown in Figure 4.6(d). The distributed approaches again have spiky curves combined with flat levels of uncertainty in between that are aligned with position error curves, i.e., the uncertainty at the fusion center increases if only predictions are possible. However, while the sample-based fusion keeps the uncertainty constant, we notice an overall growth in the covariance intersection's uncertainty over time. In addition, the sample-based fusion can nearly reach the low uncertainty of the central S^{2}KF.

Finally, the estimation results can be confirmed when looking at the estimates of a representative simulation run shown in Figure 4.7. On the one hand, the CI-based movement is rather jagged and its shape estimate is clearly wrong. On the other hand, the estimates of the central S^{2}KF and the sample-based fusion are very close to the ground truth.



(a) Position RMSE in m.

(b) Orientation RMSE in °.

(c) Volume RMSE in m^3.

(d) Average trace of $\mathbf{P}_{k|k}$.

Figure 4.6: Results of the distributed cylinder tracking evaluation.

4.5 Conclusions

In this chapter, we enabled the S^{2}KF to optimally work in a distributed setup. In order to achieve this, we were forced to develop a novel sample-based fusion approach for distributed state estimation, where correlations can be exactly reconstructed at the fusion center. The problem with state-of-the-art distributed state estimation is that either correlations are unknown or that sensor nodes require comprehensive information about the measurement processing from all other sensor nodes. While the first approach only results in conservative approximations of the joint covariance matrices, the latter is not applicable to nonlinear models as the required linearizations cannot be known in advance, and thus would lead to a massive communication overhead between the nodes.

The new sample-based fusion approach was first derived for linear models and was then extended to the case of nonlinear measurement models, for which any Kalman filter can be used that is applicable to nonlinear systems. This, of course, includes the S^{2}KF, but also the EKF, UKF, and others. It relies on a local processing of correlation samples that are sent to the fusion center in addition to the actual locally obtained state estimate, i.e., mean and covariance matrix. For the processing, each sensor node generates an initial set of correlation samples and sets of system noise samples that guarantee the made assumptions about whiteness and independence of the system state. Furthermore, these

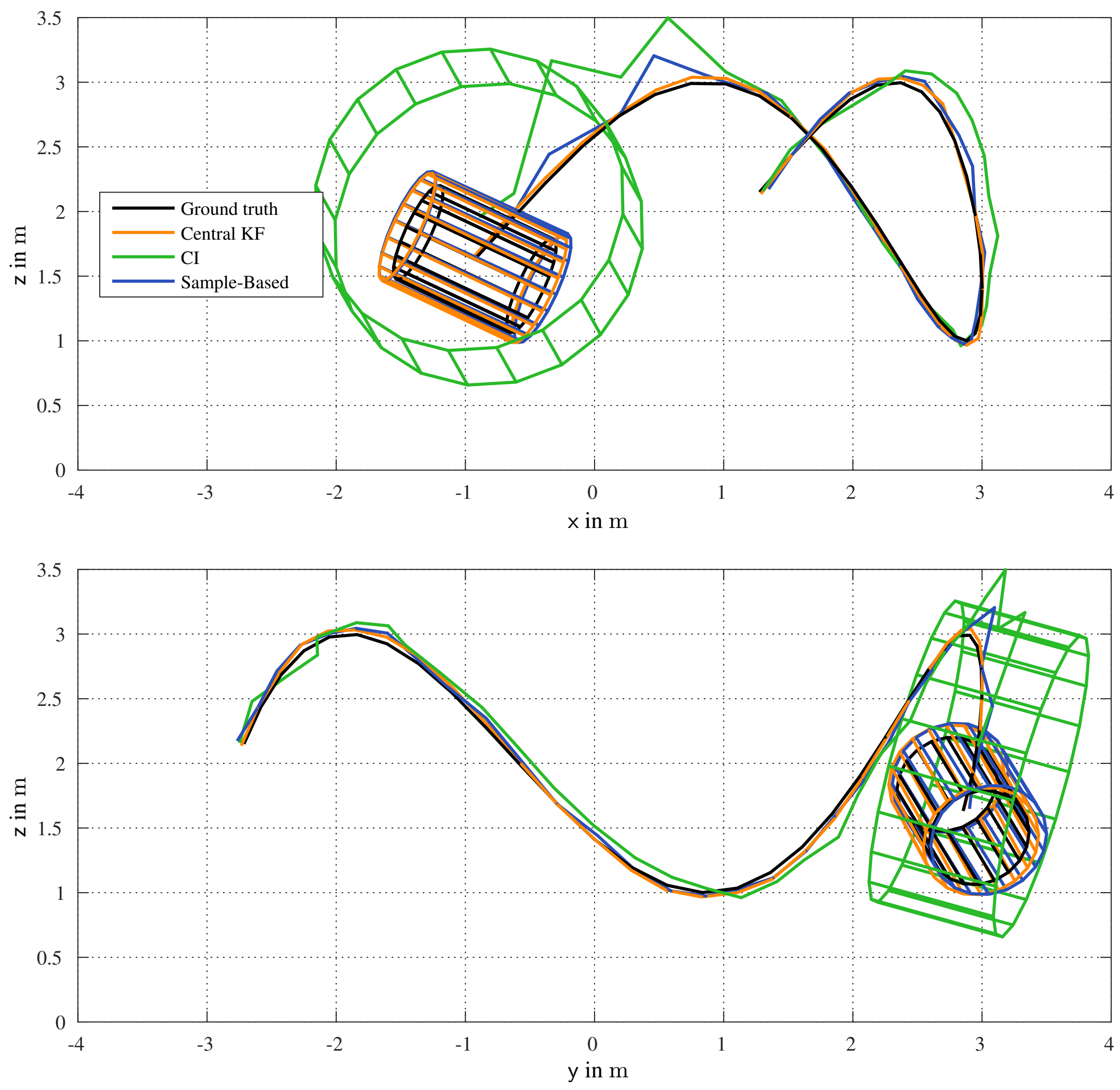


Figure 4.7: Estimated cylinders of a simulation run at time step $k = 220$.

samples are identical on all nodes in order to reflect the common system noise and its influence on the correlations between local estimates. The fusion center obtains the sought after correlation matrices by simply computing sample cross-covariance matrices out of the received correlation samples. The new sample-based distributed state estimation approach has the advantages that (i) it can exactly reconstruct the correlation information, (ii) a sensor node does not need any information about the measurement processing from other nodes, (iii) it is well-suited for large sensor networks as the number of correlation samples is independent of the number of utilized sensor node, and (iv) additional nodes can easily be added or removed over time without affecting existing nodes.

The target tracking evaluations showed that the proposed fusion technique indeed can exactly reconstruct the correlations between locally obtained state estimates, no matter if linear or nonlinear measurement models are used. Moreover, it outperformed the popular covariance intersection approach. In particular, the distributed cylinder tracking revealed that it was not even possible for covariance intersection to estimate the cylinder's pose and shape, while our sample-based fusion approach performed very well.

Chapter 5

The Progressive Gaussian Filter (PGF)

In Chapter 3, we dealt with the ubiquitous problem of estimating the hidden state of a discrete-time stochastic nonlinear dynamic system. More precisely, we considered the approach of applying the Kalman filter to the nonlinear system and introduced a new sample-based Kalman filter, the S^{2}KF. However, a direct consequence of the Kalman filter approach is the inherent Gaussian approximation of the joint density of state and measurement, which in turn means that the posterior estimate is only a linear combination of prior and measurement. Although Kalman filters can perform quite well, they have limitations in estimation quality and even in runtime under certain conditions. For example, in Kalman filtering the posterior covariance matrix is always smaller than or equal to the prior covariance matrix. Even though this holds for linear systems, in case of a nonlinear system this is in general not true and, what is more, the posterior estimate is not guaranteed to be Gaussian or at least unimodal.

Thus, our goal is to mitigate some of these restrictions to get a more advanced estimator. In particular, we want to avoid the model linearization of the Kalman filter by directly working with the likelihood function, while still forcing the state estimate to be Gaussian in order to keep the estimation tractable[1]. Recall from (3.3) that the Bayesian measurement update formula for a system state $\boldsymbol{x}_k \in \mathbb{R}^N$ and prior density $f_{k|k-1}(\boldsymbol{x}_k)$ is given by

$$f_{k|k}(\boldsymbol{x}_k) = c_k f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) f_{k|k-1}(\boldsymbol{x}_k) \quad , \tag{5.1}$$

where $f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k)$ denotes the likelihood function and c_k a normalization constant. As already mentioned, $f_{k|k}(\boldsymbol{x}_k)$ will, in general, not be Gaussian even if $f_{k|k-1}(\boldsymbol{x}_k)$ is Gaussian. Hence, due to our Gaussian assumption, we seek the approximation

$$f_{k|k}(\boldsymbol{x}_k) \approx \mathcal{N}(\boldsymbol{x}_k \,; \hat{\boldsymbol{x}}_{k|k}, \mathbf{P}_{k|k}) \quad ,$$

with

$$\hat{\boldsymbol{x}}_{k|k} = \int_{\mathbb{R}^N} \boldsymbol{x}_k c_k f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) f_{k|k-1}(\boldsymbol{x}_k) \, \mathrm{d}\boldsymbol{x}_k \quad ,$$

$$\mathbf{P}_{k|k} = \int_{\mathbb{R}^N} \left(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k}\right)\left(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k}\right)^\top c_k f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) f_{k|k-1}(\boldsymbol{x}_k) \, \mathrm{d}\boldsymbol{x}_k \quad ,$$

i.e., the filter step boils down to compute $\hat{\boldsymbol{x}}_{k|k}$ and $\mathbf{P}_{k|k}$. But even solving these integrals in closed form is not simply possible. Consequently, we have to be content with approximative solutions, like we did for the moment computations in Chapter 3.

[1]Such a Gaussian estimator can still be brought to the multimodal case by extending it to a Gaussian mixture estimator, e.g., see [22, 45, 59].

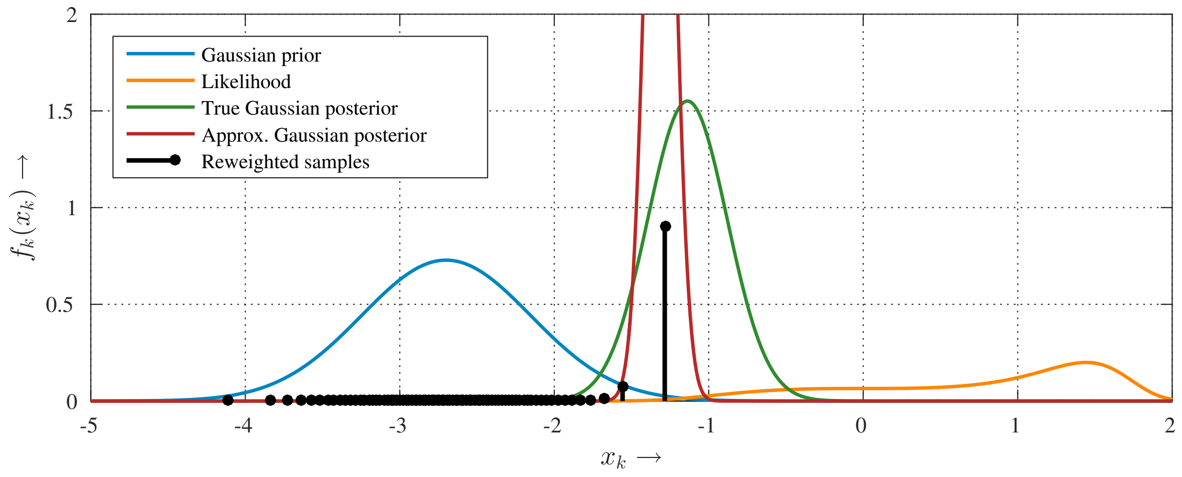


Figure 5.1: Measurement update suffering from severe sample degeneracy (cf. [183]).

A naive solution to this would be the following. Assume an adequate sample representation for the Gaussian prior $f_{k|k-1}(\boldsymbol{x}_k)$ is available, i.e., a Dirac mixture

$$f_{k|k-1}(\boldsymbol{x}_k) \approx \sum_{i=1}^{M} \omega_k^{(i)} \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^{(i)}) \tag{5.2}$$

comprising M samples with corresponding positions $\boldsymbol{x}_k^{(i)}$ and weights $\omega_k^{(i)}$. Plugging this into (5.1) immediately gives the Dirac mixture

$$f_{k|k}(\boldsymbol{x}_k) \approx \sum_{i=1}^{M} c_k f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(i)}) \omega_k^{(i)} \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^{(i)}) = \sum_{i=1}^{M} \tilde{\omega}_k^{(i)} \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^{(i)}) \ , \tag{5.3}$$

with

$$\tilde{\omega}_k^{(i)} = c_k f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(i)}) \omega_k^{(i)} = \frac{f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(i)}) \omega_k^{(i)}}{\sum_{j=1}^{M} f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(j)}) \omega_k^{(j)}} \ , \quad \forall i \in \{1, \ldots, M\} \ , \tag{5.4}$$

i.e., the posterior $f_{k|k}(\boldsymbol{x}_k)$ gets approximated as Dirac mixture as well. Notice that the samples are just reweighted, and their positions remain the same. We then get the desired posterior mean and posterior covariance matrix by simply computing mean and covariance matrix of the posterior Dirac mixture according to

$$\begin{aligned} \hat{\boldsymbol{x}}_{k|k} &\approx \sum_{i=1}^{M} \tilde{\omega}_k^{(i)} \boldsymbol{x}_k^{(i)} \ , \\ \mathbf{P}_{k|k} &\approx \sum_{i=1}^{M} \tilde{\omega}_k^{(i)} \big(\boldsymbol{x}_k^{(i)} - \hat{\boldsymbol{x}}_{k|k}\big)\big(\boldsymbol{x}_k^{(i)} - \hat{\boldsymbol{x}}_{k|k}\big)^\top \ . \end{aligned} \tag{5.5}$$

Although approximating the posterior moments in this way seems straightforward, it suffers from the severe problem of *sample degeneracy*: if the supports of prior and likelihood do not share great portions, it is very likely that only a single or even none of the posterior sample weights $\tilde{\omega}_k^{(i)}$ is (numerically) greater than zero. The problem is illustrated in Figure 5.1. Here, the prior Gaussian is approximated with $M = 80$ equally weighted samples obtained from the point-symmetric LCD-based sampling from Chapter 2. It can be seen that we have a massive problem with sample degeneracy, i.e., only a few posterior samples have weights significantly greater than zero. Moreover, the approximated posterior differs significantly from the true Gaussian posterior. For large state dimensions N, sample degeneracy will become even worse due to the curse of dimensionality. So even the advanced LCD-based Gaussian sampling technique cannot avoid the problem of sample degeneracy.

In addition, sample degeneracy can especially be a problem for the posterior covariance matrix. While a single weight $\tilde{\omega}_k^{(i)} > 0$ is sufficient to get a valid posterior mean[2], a valid posterior covariance matrix requires at least $N + 1$ weights that are significantly larger than zero, for which the corresponding samples also must not lie in the same hyperplane. However, it is also important to note that the posterior mean is a convex combination of the prior sample positions, which leads to the fact that the posterior mean always lies in their convex hull. In contrast to Kalman filters where the posterior mean can take any value during a filter step, this heavily constrains the posterior mean on the sample approximation of the prior.

Due to all these issues, conducting the measurement update in this naive way is not an option. Fortunately, a novel estimator, called the progressive Gaussian filter 42 (PGF 42), was recently proposed in [67]. The PGF 42 avoids sample degeneracy by approximating the posterior Gaussian distribution by means of several intermediate Gaussian distributions. Hence, this chapter is dedicated to progressive Gaussian filtering: we take up the promising PGF 42 approach and improve its quality and performance in various ways. This also includes a massively parallel implementation for graphics processing units. Additionally, we derive closed-form likelihood functions for efficiently estimating pose and shape of star-convex-shaped objects in 2D and spheres in 3D.

This chapter is based on the publications [183–186].

5.1 Related Work

The most popular nonlinear estimators are particle filters. In contrast to our assumption of a Gaussian distributed state estimate, a particle filter's estimate is a Dirac mixture, i.e., a set of weighted samples also called particles. On the one hand, this inherently allows for multimodal estimates. On the other hand, particle filters also suffer from sample degeneracy and usually a large amount of samples is required to get satisfactory estimation results. Particle filtering is mainly based on two concepts: (i) importance sampling using an importance density and (ii) (adaptive) resampling. While importance sampling is responsible for moving the samples $\boldsymbol{x}_k^{(i)}$ in regions where the posterior state density will likely has most of its probability mass, resampling randomly duplicates samples with larger weights $\tilde{\omega}_k^{(i)}$ in order to drop samples with smaller weights. Often, the decision of resampling is based on the so-called effective sample size: if it drops below a predefined threshold, a resampling is conducted [12, Sec. 3.2]. Both importance sampling and resampling are needed to counteract sample degeneracy as best as possible. Prediction in particle filtering is usual done sample-wise by drawing for each state sample a random realization of the system noise distribution and propagating both through the system equation in order to get a predicted Dirac mixture (contrary to the measurement update, the sample weights do not change).

Proposed particle filters primarily differ in the choice of the importance density, and how and when resampling is done. The simplest particle filter is the sampling importance resampling particle filter (SIRPF) [12, Sec. 3.5.1], [54], [55]. For the SIRPF, the importance density equals the prior, which results in a simple reweighting of the prior samples based on the likelihood function, i.e., the update is the same as in (5.3), but the prior samples do not necessarily reflect a Gaussian distribution. A variant of the SIRPF is the auxiliary sampling importance resampling particle filter (ASIRPF) [12, Sec. 3.5.2], [54], [55]. The idea of the modification is to improve the resampling at time step $k - 1$ based on the measurement from time step k. Also the regularized particle filter (RPF) [12, Sec. 3.5.3], [54]

[2]Nevertheless, the resulting mean can still be far away from the true posterior mean.

closely resembles the SIRPF except for the resampling step: it tries to improve the diversity of the posterior by individually perturbing the already resampled particles based on the sample covariance of the reweighted Dirac mixture. In order to get an improved importance density, the local linearization particle filter [12, Sec. 3.5.4] runs a separate Kalman filter, e.g., EKF or UKF, for each particle. However, the additional overhead coming from all these Kalman filters makes this estimator even more computationally demanding.

A completely different particle filter approach is taken by the box particle filter [145, 146]. For this filter, the estimate is composed of weighted “boxes”, i.e., an N-D uniform mixture, rather than a Dirac mixture. Due to the uniform mixture, the filter heavily relies on interval arithmetic instead of point-wise evaluations of the likelihood function. Also the measurement update of the convolution particle filter [147, 148] is quite different from the above described procedures. It is based on a combination of user-defined kernel densities, e.g., Gaussian kernels, and randomly drawn realizations of the measurement distribution. The result is again a kernel mixture over the state space, from which resampling is done to finally get a Dirac mixture.

What comes closest to our estimator demands (Gaussian state estimate combined with likelihood-based filter step) is the Gaussian particle filter (GPF) [58] and its Gaussian mixture extension [59]. The GPF differs from the above particle filters due to its state estimate, which is always a Gaussian distribution rather than a set of weighted samples. In fact, its measurement update is as in (5.2)–(5.5), where the samples are randomly drawn from the prior Gaussian estimate. For the state prediction, the Gaussian estimate from the last time step is randomly sampled and for each sample a random realization of the system noise distribution is drawn. Propagating these through the system model and subsequently doing moment matching gives the predicted state estimate. In [58], also a variant is suggested where the predicted samples are not reduced to a Gaussian. Instead, they are directly reweighted with the likelihood function. Similar to the ASIRPF, we get some sort of combined time update and measurement update that avoids an intermediate Gaussian approximation.

A further well-known particle filter is the ensemble Kalman filter (EnKF) [60–64]. Originally, the EnKF was developed for nonlinear system equations and linear measurements. Due to the latter assumption, the filter step closely resembles the Kalman filter update, hence its name. More precisely, its state estimate, which is always an equally weighted Dirac mixture, is updated by moving the samples in state space instead of reweighting them. Consequently, sample degeneracy is no issue here. The movement is based on a linear combination of prior sample position, measurement, and measurement noise realization (to account for the measurement noise). Another advantage of the EnKF is that the measurement update never requires a covariance matrix of the state estimate. In particular, this means that the EnKF even works with very few samples, which is especially important when dealing with very large state spaces, where processing at least $N + 1$ samples would not be possible in a reasonable time. The main drawback of the EnKF is, however, that in case of nonlinear measurement models the same linearization error is introduced as in Kalman filtering, i.e., the EnKF has to linearize the relationship between state and measurement as it does not work with a likelihood function.

In [65], the approach of a progressive measurement update for RPFs is proposed to tackle the problem of sample degeneracy. The key idea is to split the given likelihood function into a product of likelihood functions and then, starting with the prior Dirac mixture, do several consecutive updates, where each update consists of sample reweighting and resampling in order to get a final posterior Dirac mixture. Splitting the likelihood function is done adaptively based on the current likelihood evaluations. This progression approach is reformulated in [66] to use deterministic Dirac mixtures for the state estimate instead of random samples, i.e., the resampling step of the RPF is replaced by an optimization procedure. That means, a reweighted Dirac mixture is replaced with an equally weighted one by optimizing its sample positions such that they optimally represent the reweighted Dirac mixture. In particular, the optimization is a minimization of a LCD-based distance measure between both Dirac mixtures paired with regularization. The “resampling by optimization” from [66] finally leads to the PGF 42 introduced in [67], where its state estimate is now a Gaussian distribution rather than a Dirac mixture. A progression step of the measurement update is conducted by sampling the prior Gaussian estimate with the aid of the asymmetric LCD-based Gaussian sampling technique and reweighting the samples based on the likelihood. The reweighted Dirac mixture is then replaced with a Gaussian by means of moment matching, which is again re-approximated with equally weighted samples in the next progression step. Thus, the optimization-based resampling is replaced with a combination of moment matching and subsequent Gaussian sampling. On the one hand, the PGF 42 is restricted to a unimodal estimate. On the other hand, doing resampling in this way is much cheaper than solving a nonlinear optimization problem in each progression step. To improve the adaptive splitting of the likelihood function into sublikelihoods, the PGF 42 uses additional so-called forward/backward updates during the progression in order to avoid too large progression steps, which can negatively affect the estimation quality. The progressive approach from the PGF 42 is also ported to the circular domain in [149].

5.2 Progressive Gaussian Filtering

The progressive measurement update of the PGF 42 perfectly meets our filter requirements of a Gaussian estimate combined with a likelihood-based update. However, even though the PGF 42 is a promising filtering approach, it has still room for improvements. Thus, in this section we derive an enhanced progressive Gaussian filter that we simply abbreviate as PGF. These enhancements touch many different aspects of the PGF 42:

1. The PGF 42 is formulated to directly work with an arbitrary generative measurement model $\boldsymbol{y}_k = \boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{v}_k)$ rather than with a likelihood function $f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k)$. That is, the filter is responsible to translate a given measurement model into a likelihood function in order to conduct the filter step. For pure additive noise, i.e., $\boldsymbol{y}_k = \boldsymbol{h}_k(\boldsymbol{x}_k) + \boldsymbol{v}_k$, the corresponding closed-form likelihood function can be immediately obtained, and thus poses no problem. In case of non-additive noise, however, the PGF 42 estimates the joint space of system state $\boldsymbol{x}_k$ and non-additive noise $\boldsymbol{v}_k$, i.e., the PGF 42 estimates more variables than the actual system state consists of. Although this approach makes filter usage much easier, it turns out that it does not work quite well as we will see in the evaluation. Moreover, if we have to process many measurements per time step, a lot of noise variables have to be estimated, which does not scale very well. Hence, we revise the progressive update in order to work with a given likelihood function, not an arbitrary generative measurement model.

2. We replace the originally used asymmetric LCD-based Gaussian sampling scheme with the new point-symmetric sampling scheme from Chapter 2 to improve both the state prediction and the progressive measurement update.

3. The PGF 42’s forward/backward updates are dropped. Analyses revealed that these can significantly increase the filter runtime, while only marginally improving the estimation quality.

4. We propose a heuristic to automatically parameterize the PGF.

The new PGF will be further improved with a semi-analytic progressive measurement update in Section 5.3 and a massively parallel implementation for a graphics processing unit in Section 5.4.

5.2.1 Measurement Update

In order to perform the sought after progressive measurement update with a given likelihood function, we first introduce the concept of progressive likelihoods.

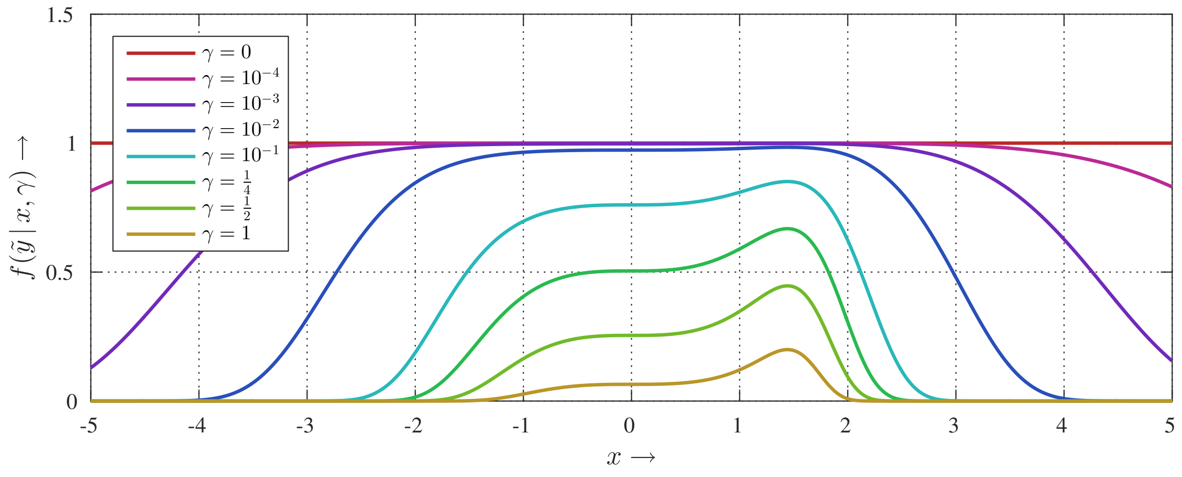


Figure 5.2: Example for a progressive likelihood.

Definition 5.1: Progressive Likelihood [65]
Let $f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k)$ be a likelihood. Then, the function

$$f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k, \gamma) := f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k)^\gamma \tag{5.6}$$

with progression parameter $\gamma \in [0, 1]$ is its progressive likelihood. Especially, for the extreme cases $\gamma = 0$ and $\gamma = 1$ it holds

$$\begin{aligned} f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k, 0) &= 1 \ , \\ f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k, 1) &= f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) \ , \end{aligned} \tag{5.7}$$

respectively.

The progressive likelihood for the likelihood function $f_k(\tilde{y}_k \,|\, x_k) = \mathcal{N}(\tilde{y}_k - x^3\,; 0, 4)$ with $\tilde{y}_k = 3$ is illustrated in Figure 5.2. Note the unit function for $\gamma = 0$ and the original likelihood function $f_k(\tilde{y}_k \,|\, x_k)$ for $\gamma = 1$. Based on the progressive likelihood (5.6), we can formulate a progressive Bayesian update according to

$$\begin{aligned} f_{k|k}(\boldsymbol{x}_k, \gamma) &= \frac{f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k, \gamma) f_{k|k-1}(\boldsymbol{x}_k)}{\int_{\mathbb{R}^N} f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k, \gamma) f_{k|k-1}(\boldsymbol{x}_k) \,\mathrm{d}\boldsymbol{x}_k} \\ &= c_k(\gamma) f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k, \gamma) f_{k|k-1}(\boldsymbol{x}_k) \ , \end{aligned} \tag{5.8}$$

with progressive posterior $f_{k|k}(\boldsymbol{x}_k, \gamma)$ and progression parameter $\gamma \in [0, 1]$. Note that the normalization constant c_k now also depends on γ. For a Gaussian prior $f_{k|k-1}(x_k) = \mathcal{N}(x\,; -3, 1)$ and the same likelihood as in Figure 5.2, the resulting progressive posterior is shown for different progression levels in Figure 5.3. As we can see, even though the prior is Gaussian, a progressive posterior, in general, is not (like for the standard Bayesian update).

Due to the extreme cases (5.7) for the progressive likelihood, we analogously get the progressive posterior extreme cases

$$\begin{aligned} f_{k|k}(\boldsymbol{x}_k, 0) &= f_{k|k-1}(\boldsymbol{x}_k) \ , \\ f_{k|k}(\boldsymbol{x}_k, 1) &= c_k f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) f_{k|k-1}(\boldsymbol{x}_k) = f_{k|k}(\boldsymbol{x}_k) \ . \end{aligned}$$

These can be interpreted as follows. On the one hand, for $\gamma = 0$, no measurement information is processed and consequently the posterior equals the prior. On the other hand, for $\gamma = 1$, the entire measurement information is fused with the prior estimate, which yields the unmodified Bayesian update.

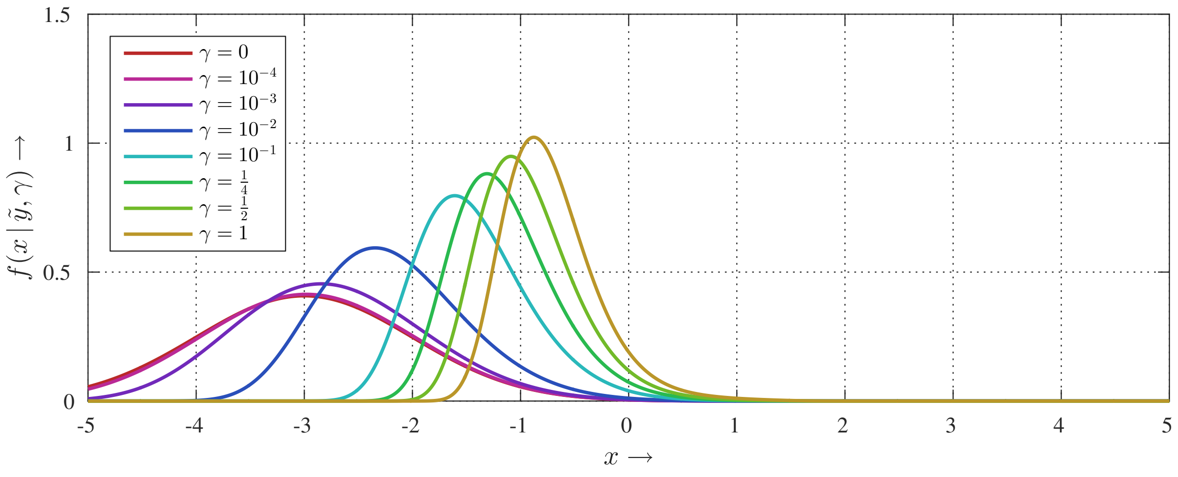


Figure 5.3: Example for a progressive posterior.

Recursive Bayesian Progression

Next, like in [67], we derive a recursive version of the progressive Bayesian update (5.8). For this, we assume to have two progressive updates: the first with progression level $0 \leq \gamma < 1$

$$f_{k|k}(\boldsymbol{x}_k, \gamma) = c_k(\gamma) f_k(\tilde{\boldsymbol{y}}_k \mid \boldsymbol{x}_k, \gamma) f_{k|k-1}(\boldsymbol{x}_k)$$

and the second with progression level $0 < \gamma + \Delta \leq 1$

$$f_{k|k}(\boldsymbol{x}_k, \gamma + \Delta) = c_k(\gamma + \Delta) f_k(\tilde{\boldsymbol{y}}_k \mid \boldsymbol{x}_k, \gamma + \Delta) f_{k|k-1}(\boldsymbol{x}_k) \ ,$$

where $0 < \Delta \leq 1$ is called the step size. Combining these and using the definition (5.6), we get a recursive formulation

$$\begin{aligned} f_{k|k}(\boldsymbol{x}_k, \gamma + \Delta) &= \frac{c_k(\gamma + \Delta) f_k(\tilde{\boldsymbol{y}}_k \mid \boldsymbol{x}_k, \gamma + \Delta)}{c_k(\gamma) f_k(\tilde{\boldsymbol{y}}_k \mid \boldsymbol{x}_k, \gamma)} f_{k|k}(\boldsymbol{x}_k, \gamma) \\ &= \frac{c_k(\gamma + \Delta)}{c_k(\gamma)} f_k(\tilde{\boldsymbol{y}}_k \mid \boldsymbol{x}_k, \Delta) f_{k|k}(\boldsymbol{x}_k, \gamma) \ , \end{aligned} \tag{5.9}$$

i.e., we get $f_{k|k}(\boldsymbol{x}_k, \gamma + \Delta)$ based on $f_{k|k}(\boldsymbol{x}_k, \gamma)$ and the step size Δ.

Starting with $\gamma = 0$, that is, the prior $f_{k|k}(\boldsymbol{x}_k, 0) = f_{k|k-1}(\boldsymbol{x}_k)$, and D positive (not necessarily equal) step sizes $\Delta^{(1)}, \Delta^{(2)}, \ldots, \Delta^{(D)}$ for which

$$\sum_{i=1}^{D} \Delta^{(i)} = 1$$

holds, we can perform the Bayesian update in a recursive manner and get the exact posterior $f_{k|k}(\boldsymbol{x}_k)$. However, this is only true if we can solve each recursion step in closed form. Unfortunately, like for the unmodified Bayesian update, this is in general not possible.

Recursive Sample-Based Gaussian Progression

Hence, to make the recursive update tractable, we now additionally assume $f_{k|k}(\boldsymbol{x}_k, \gamma)$ to be Gaussian, i.e.,

$$f_{k|k}(\boldsymbol{x}_k, \gamma) \approx \mathcal{N}(\boldsymbol{x}_k \, ; \hat{\boldsymbol{x}}_{k|k}(\gamma), \mathbf{P}_{k|k}(\gamma)) \ , \tag{5.10}$$

with mean $\hat{\boldsymbol{x}}_{k|k}(\gamma)$ and covariance matrix $\mathbf{P}_{k|k}(\gamma)$. In particular, for $\gamma = 0$, we have

$$\mathcal{N}(\boldsymbol{x}_k\,;\hat{\boldsymbol{x}}_{k|k}(0),\mathbf{P}_{k|k}(0)) = \mathcal{N}(\boldsymbol{x}_k\,;\hat{\boldsymbol{x}}_{k|k-1},\mathbf{P}_{k|k-1})$$

due to our Gaussian estimate. Further, we approximate the Gaussian (5.10) with an equally weighted Dirac mixture

$$\mathcal{N}(\boldsymbol{x}_k\,;\hat{\boldsymbol{x}}_{k|k}(\gamma),\mathbf{P}_{k|k}(\gamma)) \approx \frac{1}{M}\sum_{i=1}^{M}\delta(\boldsymbol{x}_k - \boldsymbol{x}_k^{(i)}(\gamma)) \tag{5.11}$$

using a the point-symmetric LCD-based sampling technique. Substituting (5.10) and (5.11) into in the recursion (5.9), we get the reweighted Dirac mixture

$$\begin{aligned} f_{k|k}(\boldsymbol{x}_k,\gamma+\Delta) &\approx \frac{c_k(\gamma+\Delta)}{c_k(\gamma)} f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k,\Delta)\mathcal{N}(\boldsymbol{x}_k\,;\hat{\boldsymbol{x}}_{k|k}(\gamma),\mathbf{P}_{k|k}(\gamma)) \\ &\approx \sum_{i=1}^{M}\frac{c_k(\gamma+\Delta)}{c_k(\gamma)M} f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(i)}(\gamma),\Delta)\delta(\boldsymbol{x}_k - \boldsymbol{x}_k^{(i)}(\gamma)) \ . \end{aligned}$$

After normalizing the sample weights according to

$$\omega_k^{(i)}(\gamma+\Delta) = \frac{f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(i)}(\gamma),\Delta)}{\sum_{j=1}^{M} f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(j)}(\gamma),\Delta)} \ , \quad \forall i \in \{1,\ldots,M\} \ , \tag{5.12}$$

we get the approximation

$$f_{k|k}(\boldsymbol{x}_k,\gamma+\Delta) \approx \sum_{i=1}^{M}\omega_k^{(i)}(\gamma+\Delta)\delta(\boldsymbol{x}_k - \boldsymbol{x}_k^{(i)}(\gamma)) \ . \tag{5.13}$$

Finally, we reduce the Dirac mixture (5.13) again to a Gaussian distribution by means of moment matching, that is, we compute mean

$$\hat{\boldsymbol{x}}_{k|k}(\gamma+\Delta) \approx \sum_{i=1}^{M}\omega_k^{(i)}(\gamma+\Delta)\boldsymbol{x}_k^{(i)}(\gamma)$$

and covariance matrix

$$\mathbf{P}_{k|k}(\gamma+\Delta) \approx \sum_{i=1}^{M}\omega_k^{(i)}(\gamma+\Delta)(\hat{\boldsymbol{x}}_{k|k}(\gamma+\Delta) - \boldsymbol{x}_k^{(i)}(\gamma))(\hat{\boldsymbol{x}}_{k|k}(\gamma+\Delta) - \boldsymbol{x}_k^{(i)}(\gamma))^\top \ ,$$

and set

$$f_{k|k}(\boldsymbol{x}_k,\gamma+\Delta) \approx \mathcal{N}(\boldsymbol{x}_k\,;\hat{\boldsymbol{x}}_{k|k}(\gamma+\Delta),\mathbf{P}_{k|k}(\gamma+\Delta)) \ .$$

With this, we can proceed the recursion until we get a final Gaussian

$$f_{k|k}(\boldsymbol{x}_k) \approx \mathcal{N}(\boldsymbol{x}_k\,;\hat{\boldsymbol{x}}_{k|k},\mathbf{P}_{k|k}) = \mathcal{N}(\boldsymbol{x}_k\,;\hat{\boldsymbol{x}}_{k|k}(1),\mathbf{P}_{k|k}(1))$$

as posterior state estimate. In other words, the progressive update transforms the prior Gaussian distribution into a posterior Gaussian distribution by means of several intermediate Gaussian distributions.

Automatic Step Size Control

Nevertheless, what still has to be determined is an adequate number of recursion steps D and appropriate step sizes $\Delta^{(i)}$. Recall that a progressive Gaussian measurement update is a sequence of Gaussian approximations. Consequently, the larger the step sizes, the smaller the total number of intermediate Gaussian approximations, and in turn less approximation errors accumulate until the final posterior Gaussian is reached. However, for too large step sizes, we can again run into the problem of sample degeneracy, which we actually want to avoid. Thus, we have to make a trade-off such that each recursion step is as large as possible but as small as necessary to avoid sample degeneracy. Moreover, in order to put more emphasis on measurements containing much information and process less informative measurements more quickly, a variable amount of recursion steps per measurement update is desirable.

Like the PGF 42, we determine the step size Δ of the current recursion step based on the minimum and maximum *posterior* sample weights (5.12). Let $1 \leq s_k, l_k \leq M$ denote the indices of the samples that have the smallest and largest posterior sample weight, respectively, i.e.,

$$\begin{aligned} \omega_k^{(s_k)}(\gamma + \Delta) \leq \omega_k^{(i)}(\gamma + \Delta) \quad , \quad \forall i \in \{1, \ldots, M\} \ , \\ \omega_k^{(i)}(\gamma + \Delta) \leq \omega_k^{(l_k)}(\gamma + \Delta) \quad , \quad \forall i \in \{1, \ldots, M\} \ . \end{aligned}$$

Now, we *force* the ratio of these sample weights to be equal to a given ration $R \in (0, 1)$, i.e.,

$$R \overset{!}{=} \frac{\omega_k^{(s_k)}(\gamma + \Delta)}{\omega_k^{(l_k)}(\gamma + \Delta)} = \frac{f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(s_k)}(\gamma), \Delta)}{f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(l_k)}(\gamma), \Delta)} \ .$$

Due to

$$f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(i)}(\gamma), \Delta) = \exp\Big(\log\big(f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(i)}(\gamma))\big)\Delta \Big) \ ,$$

we can get the desired step size according to

$$\Delta = \frac{\log(R)}{\log\big(f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(s_k)}(\gamma))\big) - \log\big(f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(l_k)}(\gamma))\big)} \ . \tag{5.14}$$

Thus, the smaller the forced ratio R, the large will be the step size Δ. Note that a value of $R = 1$ would imply that all posterior samples are equally weighted, which would result in

$$\mathcal{N}(\boldsymbol{x}_k \,; \hat{\boldsymbol{x}}_{k|k}(\gamma + \Delta), \mathbf{P}_{k|k}(\gamma + \Delta)) = \mathcal{N}(\boldsymbol{x}_k \,; \hat{\boldsymbol{x}}_{k|k}(\gamma), \mathbf{P}_{k|k}(\gamma)) \ .$$

In particular, this means no further recursion step will change our current Gaussian $f_{k|k}(\boldsymbol{x}_k, \gamma)$ and we can stop (or abort) the measurement update[3]. A value of $R = 0$, which implies $\omega_k^{(s_k)}(\gamma + \Delta) = 0$, would prohibit any useful value for Δ. Hence, we have to ensure that $0 < \omega_k^{(s_k)}(\gamma + \Delta)$. However, unlike the PGF 42, we deal with a given likelihood functions that might reweight a sample to zero, i.e., if the likelihood value is zero. We solve this issue by selecting $\omega_k^{(s_k)}(\gamma + \Delta)$ and $\omega_k^{(l_k)}(\gamma + \Delta)$ solely from the set of positive posterior sample weights.

At this point, we still have to figure out the indices s_k and l_k, although we do not know the step size Δ that is required for the computation of the $\omega_k^{(i)}(\gamma + \Delta)$. Fortunately, from (5.12) we see that the smallest (largest) posterior sample weight $\omega_k^{(i)}(\gamma + \Delta)$ also has the smallest (largest) progressive likelihood value $f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(i)}(\gamma), \Delta)$. Thus, we simply have to determine if

$$\exp\Big(\log\big(f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(i)}(\gamma))\big)\Delta \Big) \leq \exp\Big(\log\big(f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(j)}(\gamma))\big)\Delta \Big)$$

[3]Moreover, in (5.14) we would have a division by zero and could not even compute Δ.

to identify the smallest (largest) posterior sample weight (and with these the corresponding sample indices s_k and l_k). This is equivalent to

$$\log\left(f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(i)}(\gamma))\right) \leq \log\left(f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(j)}(\gamma))\right) \ .$$

That is, to get the indices of the smallest and largest posterior sample weights, we only have to evaluate the logarithm of the likelihood function, i.e., the log-likelihood, for all samples $\boldsymbol{x}_k^{(i)}(\gamma)$, which we have to do anyway to get the posterior sample weights[4]. Consequently, we first evaluate the log-likelihood and afterwards compute the progression step size Δ in order to get the desired posterior sample weights. It is important to note that this is only possible due to the equally weighted samples (5.11), i.e., the point-symmetric LCD-based Gaussian sampling scheme! Hence, the sampling techniques from GHKF, 5th-degree CKF, or RUKF cannot be used for the PGF.

Parametrization

Besides the number of samples M, the ratio parameter R is a user-defined parameter of the PGF 42. However, investigations revealed that increasing the number of samples while leaving the ratio R constant leads to worse estimation results. This is an unintuitive behavior, as one should expect that using more samples would result in better estimation results. We conclude that the ratio R has to be selected in such a way that it works well with the employed number of samples M. Evaluations showed that a heuristic where the forced sample weight ratio is set to

$$R := \frac{1}{M}$$

works quite well. Hence, if M becomes larger R decreases, which, in general, yield larger step sizes Δ (the step size still depends upon the concrete log-likelihood evaluations). Roughly speaking, the more samples we have the less recursion steps are performed for a measurement update. A positive side effect of setting R in this way is that the number of samples M is now the only remaining user-defined parameter of the PGF, which is intuitively to select.

Summary

The entire measurement update of the PGF is listed in Algorithm 5.1. First, standard normal distributed samples (required for the sampling of the intermediate Gaussians) are computed using the point-symmetric LCD sampling scheme (line 1). Of course, this should only be done once before the PGF is actually used for estimation. In fact, we use a sample cache to avoid unnecessary re-computations like for the S^{2}KF. Next, the recursion is initialized with the prior estimate (lines 2–4). A recursion step starts with the sampling of the current Gaussian given by $\hat{\boldsymbol{x}}_{k|k}$ and $\mathbf{P}_{k|k}$ based on the Mahalanobis transformation (lines 6–7). Then, the log-likelihood is evaluated for each sample (line 8), zero likelihood values are excluded (line 9), and minimum and maximum log-likelihood values are determined (lines 10–11). If all log-likelihood values are zero or if the minimum equals the maximum, we stop the measurement update (lines 12–14). The step size for the current recursion step is computed (line 15) and truncated if it would cause $\gamma > 1$ (lines 16–18). Based on Δ, the samples are reweighted and subsequently normalized (lines 19–21). For numerical reasons, we subtract the largest log-likelihood value $z_k^{(l_k)}$ from all values (the $\tilde{\omega}_k^{(i)}$ only have to be correct up to proportionality)[5]. Finally, the current recursion step ends by computing mean and covariance matrix to get the next Gaussian approximation (lines 22–23) and increasing γ according to Δ (line 24). The update is finished if γ reaches one.

An additional crucial difference to the PGF 42 is the missing of the already mentioned forward/-backward updates. These are intended to find more suitable step sizes Δ. The idea is to improve the

[4]Like in particle filtering, we would have used the log-likelihood anyway for numerical reasons.

[5]This is a common technique in particle filtering to ensure that at least a single particle has a weight greater than zero.

Input: prior state estimate $\hat{\boldsymbol{x}}_{k|k-1}$ and $\mathbf{P}_{k|k-1}$,
log-likelihood $\log\left(f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k)\right)$, and
number of samples M

Output: posterior state estimate $\hat{\boldsymbol{x}}_{k|k}$ and $\mathbf{P}_{k|k}$

```
🟧  1: Compute equally weighted samples $\{\boldsymbol{s}^{(i)}\}_{i=1}^{M}$ approximating
       $\mathcal{N}(\boldsymbol{s}\,;\mathbf{0},\mathbf{I}_N)$ with Algorithm 2.1
🟧  2: $\gamma \leftarrow 0$
🟧  3: $\hat{\boldsymbol{x}}_{k|k} \leftarrow \hat{\boldsymbol{x}}_{k|k-1}$
🟧  4: $\mathbf{P}_{k|k} \leftarrow \mathbf{P}_{k|k-1}$
🟧  5: while $\gamma < 1$
🟦  6:     $\mathbf{L}_k = \mathrm{chol}(\mathbf{P}_{k|k})$
🟦  7:     $\boldsymbol{x}_k^{(i)} = \mathbf{L}_k \cdot \boldsymbol{s}^{(i)} + \hat{\boldsymbol{x}}_{k|k} \quad , \quad \forall i \in \{1,\ldots,M\}$
🟦  8:     $z_k^{(i)} = \log\left(f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(i)})\right) \quad , \quad \forall i \in \{1,\ldots,M\}$
🟦  9:     $\mathcal{Z}_k = \{z_k^{(i)} \;|\; i \in \{1,\ldots,M\} \;\wedge\; -\infty < z_k^{(i)}\}$
🟦 10:     $z_k^{(s_k)} = \min(\mathcal{Z}_k)$
🟦 11:     $z_k^{(l_k)} = \max(\mathcal{Z}_k)$
🟧 12:     if $\mathcal{Z}_k = \emptyset$ or $z_k^{(s_k)} = z_k^{(l_k)}$ then
🟧 13:         No progression possible $\Rightarrow$ abort update.
🟧 14:     end if
🟧 15:     $\Delta = -\log(M) \,/\, (z_k^{(s_k)} - z_k^{(l_k)})$
🟧 16:     if $\gamma + \Delta > 1$ then
🟧 17:         $\Delta = 1 - \gamma$
🟧 18:     end if
🟦 19:     $\tilde{\omega}_k^{(i)} = \exp\left((z_k^{(i)} - z_k^{(l_k)})\Delta\right) \quad , \quad \forall i \in \{1,\ldots,M\}$
🟦 20:     $\tilde{\omega}_k = \sum_{j=1}^{M} \tilde{\omega}_k^{(j)}$
🟦 21:     $\omega_k^{(i)} = \tilde{\omega}_k^{(i)} \,/\, \tilde{\omega}_k \quad , \quad \forall i \in \{1,\ldots,M\}$
🟦 22:     $\hat{\boldsymbol{x}}_{k|k} \leftarrow \sum_{i=1}^{M} \omega_k^{(i)} \boldsymbol{x}_k^{(i)}$
🟦 23:     $\mathbf{P}_{k|k} \leftarrow \sum_{i=1}^{M} \omega_k^{(i)} (\boldsymbol{x}_k^{(i)} - \hat{\boldsymbol{x}}_{k|k})(\boldsymbol{x}_k^{(i)} - \hat{\boldsymbol{x}}_{k|k})^{\top}$
🟧 24:     $\gamma \leftarrow \gamma + \Delta$
🟧 25: end while
```

Algorithm 5.1: The progressive Gaussian filter. Blue indicates GPU-accelerated parts, while orange means the computation remains on the CPU (see Section 5.4).

estimation quality by keeping the deviation between successive intermediate Gaussians

$$\mathcal{N}(\boldsymbol{x}_k\,; \hat{\boldsymbol{x}}_{k|k}(\gamma), \mathbf{P}_{k|k}(\gamma))$$

and

$$\mathcal{N}(\boldsymbol{x}_k\,; \hat{\boldsymbol{x}}_{k|k}(\gamma + \Delta), \mathbf{P}_{k|k}(\gamma + \Delta))$$

small. However, this approach has some drawbacks. First of all, the deviation has to be quantified, e.g., by comparing mean vectors and covariance matrices element-wise or based on norms, or by computing the Kullback–Leibler divergence such as done in iterative Kalman filtering [41]. In all cases, more or less thresholds have to be specified by the user, which is not an easy and intuitive task. Another problem is what to do if a threshold cannot be reached regardless of how small Δ is chosen? Second, the forward/backward updates require additional likelihood evaluations. Third, keeping the deviations small likely result in many recursion steps, which further negatively effect filter runtime. Finally, just because deviations are kept small the resulting Gaussian posterior is not necessarily a good approximation of the (unknown) true posterior. Due to all these issues, we omitted the forward/backward updates. As a result, the PGF now comes closer to the filter proposed in [66].

An example measurement update of the PGF with the same prior and likelihood as in Figure 5.1 is illustrated in Figure 5.4. We choose to use $M = 5$ samples per recursion step. In total, the update performs ten steps. Note the movement of the samples in state space over the recursion. Compared to the naive result in Figure 5.1, sample degeneracy is completely avoided, while only requiring a total of $10 \times 5 = 50$ likelihood evaluations compared to 80 evaluations for the naive approach. For higher state dimensions, the reduction in the likelihood evaluations is even better.

Finally, limitations of the PGF's measurement update are illustrated in Figure 5.5. On the one hand, in Figure 5.5(a), the problem of equally weighted posterior samples is exemplified. Here, we could not conduct at least a single recursion step as the uniformity of the likelihood function covers most of the prior support. On the other hand, a case of severe sample degeneracy caused by a bounded support of the likelihood function is shown in Figure 5.5(b). Due to the invalid posterior variance, the recursion step cannot be finished. Such bound constraints can be handled by modelling them with an exponential decrease in the likelihood values instead of simply setting the likelihood value to zero. The progressive approach will then move the samples "into" the support of the likelihood function and sample degeneracy can be avoided. Nevertheless, we can deal with both issues in two ways. Either we abort the measurement update (like in Algorithm 5.1) and leave the state estimate unchanged, i.e., completely ignore the measurement $\tilde{\boldsymbol{y}}_k$, or use the Gaussian from the last (valid) recursion step as posterior estimate.

5.2.2 Time Update

For linear system models, we can compute the prediction in closed form due to the Gaussian state estimate. For nonlinear models, however, the time update of the PGF 42 works as follows. First, the current state estimate (and also non-additive system noise) is approximated using the combination of the asymmetric LCD-based sampling scheme and the Mahalanobis transformation. Second, the samples are propagated through the nonlinear system model. Finally, the predicted state mean and the predicted state covariance matrix are obtained by means of moment matching [67]. Concerning the PGF, we propose a marginally different state prediction that replaces the asymmetric LCD-based sampling scheme with the point-symmetric LCD-based sampling scheme from Chapter 2. Thus, the PGF prediction is identical to the S^{2}KF prediction (see Section 3.3).

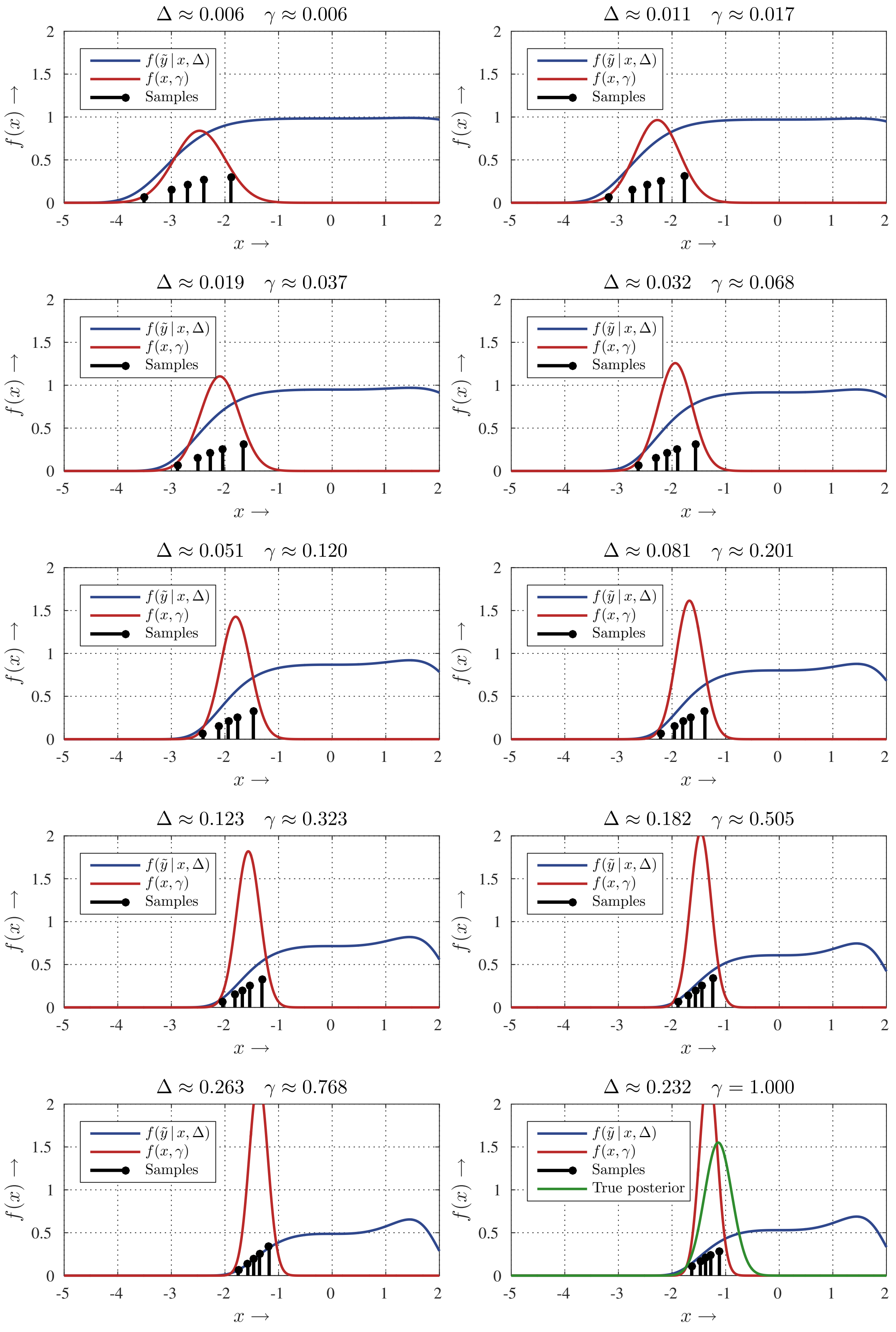
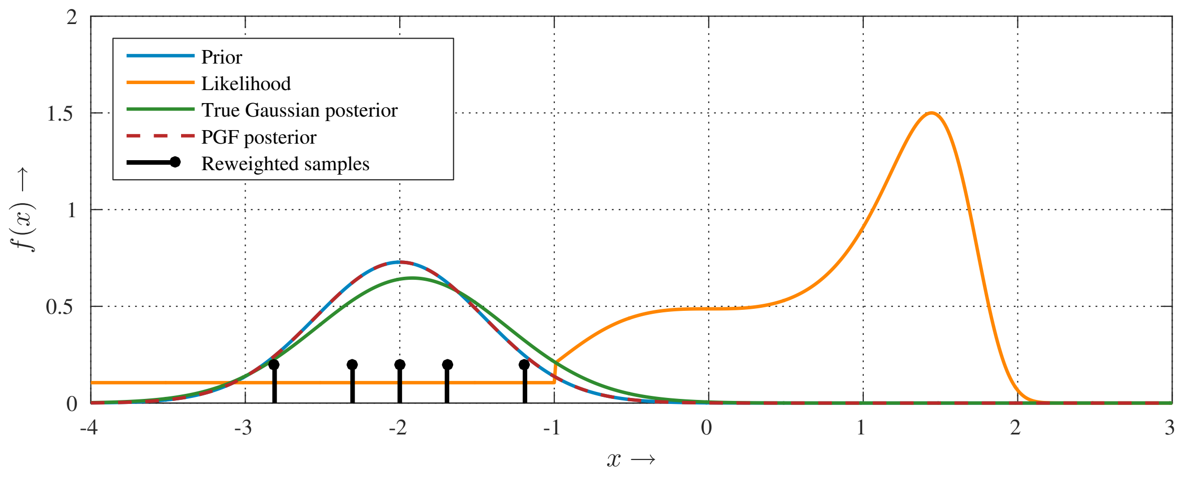
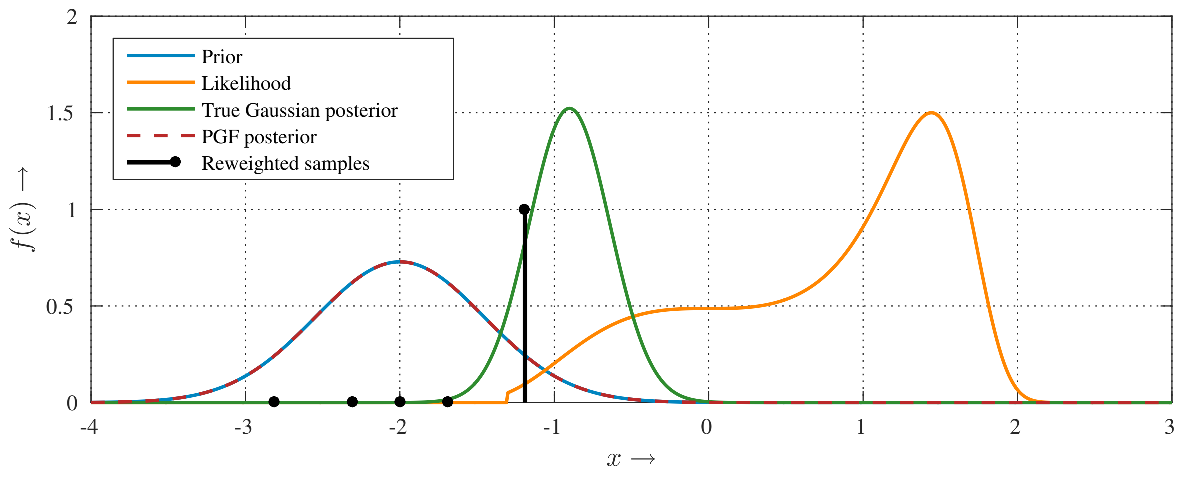


Figure 5.4: Example of the PGF’s measurement update.



(a) Likelihood with uniform part.



(b) Likelihood with bounded support.

Figure 5.5: PGF measurement update limitations.

5.3 Semi-Analytic Progressive Gaussian Filtering

In the previous section, we proposed a new version of progressive Gaussian filtering. It works with given likelihood functions, which also includes cases where a likelihood does not depend on the entire system state, that is, it only depends on a subspace of it, called the *observable* state variables[6]. This is a very common situation in (extended) target tracking. For example, usually motion parameters such as velocities or accelerations are not present in the measurement equation, and are only estimated based on existing correlations with other state variables, e.g., position or orientation.

Thanks to our Gaussian state estimate, we can exploit the information about observable state variables and derive a semi-analytic measurement for the PGF. On the one hand, we first update the state estimate of the observable state variables by applying the proposed PGF measurement update solely on the observable part of the system state. On the other hand, using the just updated state estimate of the observable part and exploiting the Gaussian state estimate, we can analytically update the estimate of the unobservable part of the system state. This semi-analytic treatment of the measurement update is not only beneficial for the estimation quality but also saves computation time.

[6]Not to be confused with the concept of observability, e.g., see [17, Sec. 1.7].

In the following, we assume the system state $\boldsymbol{x}_k$ is partitioned into two subspaces $\boldsymbol{x}_k^{(o)} \in \mathbb{R}^A$ and $\boldsymbol{x}_k^{(u)} \in \mathbb{R}^B$, that is,

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{x}_k^{(o)} \\ \boldsymbol{x}_k^{(u)} \end{bmatrix} .$$

Further, the prior Gaussian state density is given by

$$f_{k|k-1}(\boldsymbol{x}_k) = f_{k|k-1}(\boldsymbol{x}_k^{(o)}, \boldsymbol{x}_k^{(u)}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{x}_k^{(o)} \\ \boldsymbol{x}_k^{(u)} \end{bmatrix} ; \begin{bmatrix} \hat{\boldsymbol{x}}_{k|k-1}^{(o)} \\ \hat{\boldsymbol{x}}_{k|k-1}^{(u)} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{k|k-1}^{(o)} & (\mathbf{P}_{k|k-1}^{(u,o)})^\top \\ \mathbf{P}_{k|k-1}^{(u,o)} & \mathbf{P}_{k|k-1}^{(u)} \end{bmatrix} \right) . \tag{5.15}$$

Next, we consider a likelihood function

$$f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) = f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(o)}) \tag{5.16}$$

that solely depends upon the subspace $\boldsymbol{x}_k^{(o)}$. Hence, $\boldsymbol{x}_k^{(o)}$ contains the observable state variables, whereas $\boldsymbol{x}_k^{(u)}$ encompasses the unobservable state variables.

By combining (5.15) and (5.16), the posterior state density is proportional to

$$\begin{aligned} f_{k|k}(\boldsymbol{x}_k) &\propto f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) f_{k|k-1}(\boldsymbol{x}_k) \\ &= f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(o)}) f_{k|k-1}(\boldsymbol{x}_k^{(o)}, \boldsymbol{x}_k^{(u)}) \\ &= f_{k|k-1}(\boldsymbol{x}_k^{(u)} \,|\, \boldsymbol{x}_k^{(o)}) f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(o)}) f_{k|k-1}(\boldsymbol{x}_k^{(o)}) \\ &\propto f_{k|k-1}(\boldsymbol{x}_k^{(u)} \,|\, \boldsymbol{x}_k^{(o)}) f_{k|k}(\boldsymbol{x}_k^{(o)}) . \end{aligned}$$

Note that $f_{k|k-1}(\boldsymbol{x}_k^{(u)} \,|\, \boldsymbol{x}_k^{(o)})$ is a conditionally Gaussian distribution with mean

$$\hat{\boldsymbol{x}}_{k|k-1}^{(u\,|\,o)} = \hat{\boldsymbol{x}}_{k|k-1}^{(u)} + \mathbf{K}_k \big(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k-1}^{(o)}\big) \tag{5.17}$$

and covariance matrix

$$\mathbf{P}_{k|k-1}^{(u\,|\,o)} = \mathbf{P}_{k|k-1}^{(u)} - \mathbf{K}_k \big(\mathbf{P}_{k|k-1}^{(u,o)}\big)^\top , \tag{5.18}$$

where

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}^{(u,o)} \big(\mathbf{P}_{k|k-1}^{(o)}\big)^{-1} .$$

In fact, these are the Kalman filter equations. The posterior state density of subspace $\boldsymbol{x}_k^{(o)}$ is again given by Bayes' rule accoring to

$$f_{k|k}(\boldsymbol{x}_k^{(o)}) = \frac{f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(o)}) f_{k|k-1}(\boldsymbol{x}_k^{(o)})}{\int_{\mathbb{R}^A} f_k(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(o)}) f_{k|k-1}(\boldsymbol{x}_k^{(o)}) \,\mathrm{d}\boldsymbol{x}_k^{(o)}} .$$

However, like for the Bayesian update of the entire system state $\boldsymbol{x}_k$, the posterior $f_{k|k}(\boldsymbol{x}_k^{(o)})$ is, in general, not Gaussian even though $f_{k|k-1}(\boldsymbol{x}_k^{(o)})$ is.

Now, the key idea is to obtain a Gaussian approximation of $f_{k|k-1}(\boldsymbol{x}_k^{(o)})$ by applying PGF's measurement update solely on the subspace $\boldsymbol{x}_k^{(o)}$. That is, we compute

$$f_{k|k}(\boldsymbol{x}_k^{(o)}) \approx \mathcal{N}(\boldsymbol{x}_k^{(o)} \,; \hat{\boldsymbol{x}}_{k|k}^{(o)}, \mathbf{P}_{k|k}^{(o)}) \tag{5.19}$$

with Algorithm 5.1, where $\hat{\boldsymbol{x}}_{k|k-1}^{(o)}$ and $\mathbf{P}_{k|k-1}^{(o)}$ are used as the prior. Then, based on the Gaussian approximation (5.19), the posterior density of the entire system state can be approximated as a Gaussian in closed form according to

$$\begin{aligned} f_{k|k}(\boldsymbol{x}_k) &\approx f_{k|k-1}(\boldsymbol{x}_k^{(u)} \,|\, \boldsymbol{x}_k^{(o)}) \mathcal{N}(\boldsymbol{x}_k^{(o)} \,; \hat{\boldsymbol{x}}_{k|k}^{(o)}, \mathbf{P}_{k|k}^{(o)}) \\ &= \mathcal{N}\left(\begin{bmatrix} \boldsymbol{x}_k^{(o)} \\ \boldsymbol{x}_k^{(u)} \end{bmatrix} ; \begin{bmatrix} \hat{\boldsymbol{x}}_{k|k}^{(o)} \\ \hat{\boldsymbol{x}}_{k|k}^{(u)} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{k|k}^{(o)} & (\mathbf{P}_{k|k}^{(u,o)})^\top \\ \mathbf{P}_{k|k}^{(u,o)} & \mathbf{P}_{k|k}^{(u)} \end{bmatrix} \right) , \end{aligned}$$

with

$$\begin{aligned}
\hat{\boldsymbol{x}}^{(u)}_{k|k} &= \hat{\boldsymbol{x}}^{(u)}_{k|k-1} + \mathbf{K}_k\big(\hat{\boldsymbol{x}}^{(o)}_{k|k} - \hat{\boldsymbol{x}}^{(o)}_{k|k-1}\big) \ , \\
\mathbf{P}^{(u)}_{k|k} &= \mathbf{P}^{(u)}_{k|k-1} + \mathbf{K}_k\big(\mathbf{P}^{(o)}_{k|k} - \mathbf{P}^{(o)}_{k|k-1}\big)\mathbf{K}_k^\top \ , \\
\mathbf{P}^{(u,o)}_{k|k} &= \mathbf{K}_k\mathbf{P}^{(o)}_{k|k} \ , \\
\mathbf{K}_k &= \mathbf{P}^{(u,o)}_{k|k-1}\big(\mathbf{P}^{(o)}_{k|k-1}\big)^{-1} \ .
\end{aligned} \tag{5.20}$$

The proof is given in Appendix C. From (5.20), it follows that the estimate for subspace $\boldsymbol{x}^{(u)}_k$ (and the correlations with subspace $\boldsymbol{x}^{(o)}_k$) can be updated analytically given the updated state estimate for subspace $\boldsymbol{x}^{(o)}_k$. Moreover, the updated estimate will only differ from the prior one if correlations already exist, i.e., if $\mathbf{P}^{(u,o)}_{k|k-1} \neq \mathbf{0}$. Further, the mean only changes if $\hat{\boldsymbol{x}}^{(o)}_{k|k} \neq \hat{\boldsymbol{x}}^{(o)}_{k|k-1}$, while the covariance matrix is only adapted if $\mathbf{P}^{(o)}_{k|k} \neq \mathbf{P}^{(o)}_{k|k-1}$.

An analogous semi-analytic measurement update explicitly formulated for Kalman filters exists [4, App. E] and is also called state decomposition [50]. More precisely, the formulas (5.20) for updating the unobservable state variables given an updated estimate for the observable variables are identical. The only difference is how the updated estimate of the observable state variables is obtained, i.e., based on the Kalman filter update (3.8)–(3.9) or the PGF measurement update from Algorithm 5.1. For particle filters, the approach of Rao–Blackwellization [56] exists, which is also called marginalized particle filters [57]. Rao–Blackwellization exploits linear/Gaussian substructures in order to reduce the runtime of the particle filters and at the same time improve their estimation quality. In particular, these substructures include the special case of a likelihood function (5.16) that does not depend upon the entire system state. Nevertheless, it is important to note that the PGF's semi-analytic update is much different from the approach taken by marginalized particle filters. For instance, a particle filter has to perform the closed-form part of the update for *each* particle, whereas for our proposed semi-analytic update the closed-form part has only to be performed *once* per update. Moreover, we additionally have to update the correlation matrix $\mathbf{P}^{(u,o)}_{k|k}$, which does not exist for a marginalized particle filter.

The proposed semi-analytic measurement update for the PGF offers several advantages. First, the analytic treatment of the unobservable state variables improves the overall estimation quality. Second, the PGF's measurement update has to consider fewer state variables. On the one hand, a smaller state space allows to use less samples M for an update. On the other hand, several Cholesky decompositions and covariance matrix computations have to be conducted in each update. These are in $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$, respectively. Hence, reducing the considered state space also has a positive effect on the PGF's runtime. Third, in some situations the PGF can diverge when dealing with likelihood functions with unobservable state variables. The new semi-analytic measurement update avoids this.

Finally, we want to point out that the proposed semi-analytic measurement update can be used by *any* nonlinear estimator as long as its state estimate is Gaussian, e.g., the GPF. In such a case, the Gaussian approximation (5.19) will be computed with the respective measurement update, and not with the progressive update of the PGF. For instance, for the evaluation in Section 5.5.2, we will apply the semi-analytic update to the GPF as well.

5.4 GPU-Accelerated Progressive Gaussian Filtering

For quite some time, sensors have been able to provide more and more measurements *at a given time*, e.g., due to increased sensor resolutions. This is especially relevant to target tracking, where targets can now be modeled as extended objects rather than only as a single point, e.g., see [79,82,86,88,150,151]. For example, Microsoft's second-generation "Kinect for Windows" provides per frame a point cloud encompassing up to 200 000 points at a rate of 30 frames per second. However, processing such vast amount of measurements must not take longer than $\approx 33\,\mathrm{ms}$ to operate in real time. Even though the PGF is a great improvement over particle filters regarding computational demands, dealing with those requirements call for an efficient and powerful implementation.

Formally, we have the problem to process at time step k a set of $Y_k \in \mathbb{N}_+$ noisy measurements

$$\mathcal{Y}_k = \{\tilde{\boldsymbol{y}}_k^{(1)}, \ldots, \tilde{\boldsymbol{y}}_k^{(Y_k)}\} \ ,$$

which are related to the hidden system state by means of the likelihood function $f_k(\mathcal{Y}_k \,|\, \boldsymbol{x}_k)$. Under the common assumption of mutually independent measurement noise, the likelihood simplifies to

$$f_k(\mathcal{Y}_k \,|\, \boldsymbol{x}_k) = \prod_{j=1}^{Y_k} f_k^{(j)}(\tilde{\boldsymbol{y}}_k^{(j)} \,|\, \boldsymbol{x}_k) \ ,$$

where $f_k^{(j)}$ denotes the likelihood for the j^{th} measurement. For the log-likelihood, we then have

$$\log(f_k(\mathcal{Y}_k \,|\, \boldsymbol{x}_k)) = \sum_{j=1}^{Y_k} \log\left(f_k^{(j)}(\tilde{\boldsymbol{y}}_k^{(j)} \,|\, \boldsymbol{x}_k)\right) \ . \tag{5.21}$$

So a first insight is that likelihood functions have the advantage that the computational effort for their evaluation only increases *linearly* in the number of measurements, and so hopefully the overall computational effort of the PGF increases only linearly as well. A problem might be that many measurements will likely result in a very narrow likelihood function $f_k(\mathcal{Y}_k \,|\, \boldsymbol{x}_k)$, which can lead to more recursion steps of the PGF if the prior uncertainty is rather large compared to the spread of the likelihood. This can especially be the case when starting the estimation process due to the usually large uncertainty of an initial estimate. Nevertheless, if the PGF has conducted some measurement updates, the posterior will become narrow as well, leading to fewer recursion steps. Second, we can independently evaluate $\log\left(f_k^{(j)}(\tilde{\boldsymbol{y}}_k^{(j)} \,|\, \boldsymbol{x}_k^{(i)})\right)$ for each combination of measurement $\tilde{\boldsymbol{y}}_k^{(j)}$ and state sample $\boldsymbol{x}_k^{(i)}$, i.e., the likelihood evaluation can be easily parallized.

The linearity is a great advantage over Kalman filters, where the computational burden increases cubically in the number of measurements. In case of nonlinear systems, those need to stack all $\mathcal{Y}_k$ measurements to process them at once, e.g., as in Section 3.4.2. However, an increased measurement dimension expands the measurement covariance $\mathbf{Y}_k$ (3.11) we have to invert for the measurement update (or analogously solve a linear system of equations). For instance, increasing the amount of measurements by a factor of ten, it roughly takes 1 000 times longer to invert the measurement covariance matrix. Of course, another advantage of using nonlinear estimators like the PGF is that we avoid the linearization errors introduced by Kalman filters.

Although the properties of a likelihood function come in handy, a limiting factor when dealing with such large number of measurements are still currently available central processing units (CPUs). Despite a fairly easy parallelization for CPUs, e.g., by using OpenMP in C/C++ implementations [152], it is limited due to the usual CPU setup of four to eight cores. Even widely used hardware improvements such as hardware-based multithreading or single instruction multiple data (SIMD) instruction sets, e.g., SSE or AVX, can only mitigate this problem.

In contrast, today's graphics processing units (GPUs) offer massive parallel computation capabilities that can be fully utilized when processing thousands of measurements. Hence, in this section our goal is to port the PGF's measurement update to a GPU. Besides circumventing CPU limitations, another advantage of using a GPU is the reduction of the CPU load and the possibility to then use it for other tasks, e.g., measurement pre-processing. Practically, there are two options for implementing code on a GPU: either the Open Computing Language (OpenCL), an open standard developed by the Khronos

Group[7] [153], or CUDA, a proprietary parallel computing platform and programming model developed by the NVIDIA corporation [154].

5.4.1 Related Work

Utilizing the GPU for non-graphics-related computation is nowadays frequently used in scientific computing such as parallel computations of large-scale linear algebra [155], machine learning [156], or computer vision and image processing [157, 158].

Also nonlinear state estimators, that is, particle filters, are already ported to GPUs, where the most challenging parts are resampling and random number generation, which are also the runtime bottlenecks [159]. Additionally, due to the lack of a proper random number generator, in [159] the required random numbers are computed on the CPU. Unfortunately, this caused a large amount of data that had to be transferred to the GPU's memory on every measurement update.

Due to this fact, the authors of [160] completely port a particle filter on a GPU by additionally implementing a random number generator on their own. They use their GPU-accelerated particle filter for a single target video tracking application, where their GPU version is about ten times faster than their OpenMP version on a multi-core CPU. Fortunately, nowadays libraries for random number generation on a GPU are available, e.g., the cuRAND library [161].

In [162], a real-time human motion tracking based on GPU-accelerated particle filters is presented. Their original state estimation problem comprises 22 parameters, which is a demanding problem for a particle filter. Thus, they split the state vector into five subspaces and applied a separate particle filter to each subspace. Compared to the CPU implementation, the GPU implementation has a speedup of about ten. However, although a GPU was used, the original estimation problem of 22 variables was not tractable due to the many particles required for such a large state space. In contrast, for the PGF a 22D system state poses no problem as will be demonstrated in Section 5.5.3.

5.4.2 GPU Computing

When implementing non-graphics-related functions for a GPU, certain aspects have to be considered. First of all, the workflow on a GPU is much different from a usual CPU. Typically, on modern CPUs only a few instruction streams, i.e., threads, work in parallel. Despite of data/resources dependencies between threads, e.g., read/write on the same memory location, those are executed independently. Especially, this means that threads can follow code paths regardless of paths taken by other threads, e.g., taken branches and called subroutines. In summary, CPUs are very efficient in cases where different code paths have to be taken. They also are very fast when only a single thread is executed. By contrast, today's GPUs work much differently. They are optimized to execute the *same* code path many times in parallel, each on different data. That is, the computational power comes from massive parallel executions, not from faster *individual* code executions. This is because graphics-related operations usually have to be executed multiple times on different data.

That means on the one hand, many (thousands) of threads work in parallel on a GPU. However, they are tightly coupled. More precisely, threads are organized and executed in groups (typically 32 or 64 threads). All threads in a group have to follow the same code path. If threads follow different paths, the instructions from all paths are executed one after another. Threads are only active during the execution of their code paths and stalled during the execution of the other paths. For example, consider the case where only a single thread of a group follows code path "A" and all other threads follow code path "B". First, path "A" is executed and only the single thread is active whereas all other threads are stalled

[7]The Khronos Group develops various open standards such as OpenGL.

until the instructions from path “A” are executed. After that path “B” is executed. Now, only the single thread is stalled and all other threads are active. Thus, in the worst case only a single thread is active at a time, and thus the computational resources of the GPU are wasted. On the other hand, if only a few threads are executed, e.g., because a problem cannot be massively parallized as expected by a GPU, not only most of the GPU’s arithmetic logical units are not used, but also the few threads are usually executed much slower compared to an execution on a CPU.

Hence, we give some general tips regarding GPU programming:

1. Commonly, GPUs and their memory are placed on extension cards that are connected with the rest of the system via the PCI express bus. A GPU can only work on data that is available in its own memory. Hence, before the GPU can perform arithmetic operations, the required data has to be transferred, i.e., copied, from the CPU memory to the GPU memory. Unfortunately, compared to the usual memory access latencies, the PCI express bus is very slow and copying data between CPU and GPU memory can take milliseconds. Hence, the amount of data that has to be copied between CPU memory and GPU memory should be reduced to a minimum.
2. The workflow in GPU computing is as follows. First, any required data is copied by a CPU thread from the CPU memory to the GPU memory. Second, functions on the GPU are launched by the CPU thread, which process and write data in the GPU memory. Finally, the resulting data is copied from the GPU memory to the CPU memory.
3. Data transfer between CPU and GPU memory can be done in parallel to the actual computation on today’s GPUs. This saves time as data required by future GPU functions can be transferred while the GPU is still busy with the current function execution.
4. In order to maximally profit from the massive parallel execution capabilities of a GPU, it should be kept busy with arithmetic operations by running hundreds or thousands of threads at the same time in order to hide memory access latencies. In doing so, threads that have to wait for data to be transferred between the GPU memory and the GPU caches/registers do not prevent the GPU from conducting arithmetic operations as other threads can be processed, for which the required data is already available.
5. To circumvent memory access latencies, it can also be favorable to conduct identical arithmetic operations multiple times by different threads instead of sharing results between threads.
6. Due to the group execution model of threads, it is important to ensure that threads most times follow the same code path, e.g., all threads of a group should take the same branches.

5.4.3 PGF Implementation on a GPU

After briefly describing workflow and design principles of GPUs, we now focus on porting the PGF’s measurement update on a GPU. Of course, also the state prediction can be offloaded onto a GPU. Nevertheless, the measurement update is the computational expensive part, especially when dealing with thousands of measurements. Often it is necessary to transfer the filtered state estimate back to the CPU anyway, e.g., to save and/or process it. In such a case, one can perform the state prediction on the CPU and transfer the predicted estimate back to the GPU. This makes especially sense when the state prediction can be performed in closed form, e.g., in case of linear system models like in our evaluation.

Although the measurement update of the PGF works recursively, we can still effectively utilize the GPU, especially in case of many measurements per time step. In the following, we go step by step through the update given by Algorithm 5.1. Blue marked lines denote operations that are moved to the GPU. The other operations, marked as orange, will remain on the CPU. That is, not the entire procedure will be moved to the GPU as will be explained below.

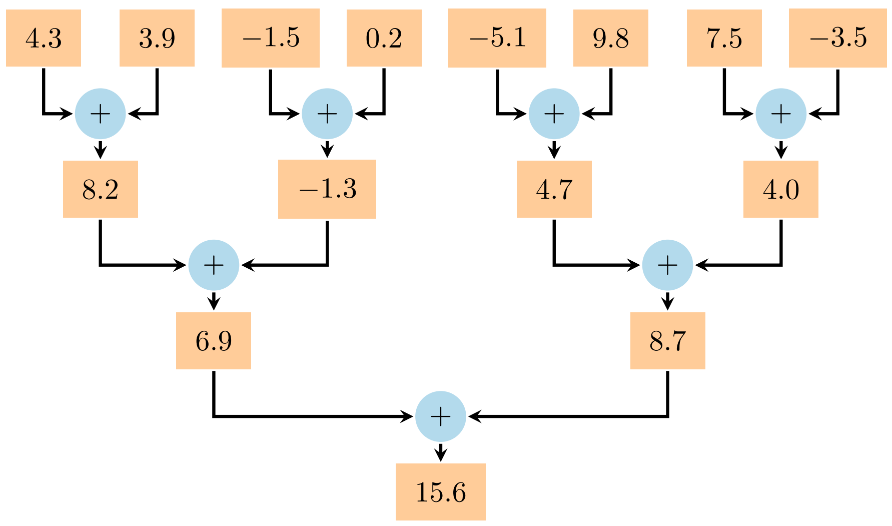


Figure 5.6: Parallel reduction scheme for the addition (here with three stages). All operations per reduction stage are performed in parallel. Other reductions such as "<" or ">" can be parallelized in the same way.

1. We copy all relevant data from CPU memory to GPU memory: predicted state mean $\hat{\boldsymbol{x}}_{k|k-1}$ and predicted state covariance matrix $\mathbf{P}_{k|k-1}$, the pre-computed samples $\{\boldsymbol{s}^{(i)}\}_{i=1}^{M}$ stored in a matrix according to
$$\mathbf{S} = \left[\boldsymbol{s}^{(1)} \cdots \boldsymbol{s}^{(M)}\right] \in \mathbb{R}^{N \times M} \quad ,$$
the measurements $\mathcal{Y}_k$, and maybe other likelihood related data such as noise covariance matrices. Note that if the samples $\mathbf{S}$ do not change over time, e.g., dimension or number of samples, they only have to transferred once to the GPU memory for the entire program execution. Thus, their transfer yield no additional runtime overhead during the actual state estimation, as will be the case in our evaluation.

2. The while loop (line 5) itself is executed on the CPU, i.e., it enqueues all the necessary function calls on the GPU and manages the data transfer between CPU and GPU.

3. We have to compute the Cholesky decomposition $\mathbf{L}_k$ of the current covariance matrix $\mathbf{P}_{k|k}$ (line 6). Unfortunately, the Cholesky decomposition is an iterative procedure, and hence only the involved data can be read and written in parallel. The decomposition itself has to be computed by a single thread, which, however, is still faster than performing the decomposition on the CPU (including the required data transfer).

4. The samples $\boldsymbol{x}_k^{(i)}$ are obtained by performing a parallelized matrix-matrix multiplication of $\mathbf{L}_k$ and $\mathbf{S}$, and subsequently by performing a parallelized column-wise addition of the current mean $\hat{\boldsymbol{x}}_{k|k}$ (line 7). The result is a matrix $\mathbf{M}_k \in \mathbb{R}^{N \times M}$, which stores the samples $\boldsymbol{x}_k^{(i)}$ column-wise.

5. Evaluating the log-likelihood can basically be performed in parallel (line 8). However, the evaluation depends on the concrete likelihood function. A general approach for log-likelihoods (5.21) is to launch threads over all combinations of sublikelihoods $f_k^{(j)}$ and state samples $\boldsymbol{x}_k^{(i)}$, which store their values in a large matrix $\mathbf{V}_k \in \mathbb{R}^{Y_k \times M}$. Then, we column-wise sum up to get the final log-likelihood values. These sums can be completely obtained in parallel per column, i.e., per sample, but also the summation itself can be parallelized. This parallelization is done by the common tree-like parallel reduction scheme that requires only a logarithmic runtime in the number of values (Y_k in our case), e.g., see [159]. The reduction scheme is also illustrated in

Figure 5.6. We will address the log-likelihood computation in more detail during the evaluation in Section 5.5.4, when a concrete likelihood has to be implemented.

6. We combine excluding zero likelihood values and finding the values $z_k^{(s_k)}$ and $z_k^{(l_k)}$ (lines 9–11) by again making use of parallel reductions (now with "<" and ">"). This is done as follows. A zero likelihood value will cause a log-likelihood value of $-\infty$. Regarding finding the maximum, we are fine as any finite value will be favored over $-\infty$. In case of finding the minimum, we first have to check if one of the operands is $-\infty$. If so, the other operand (the finite one) will be defined as the smaller one. In case of both operands are $-\infty$, it will be used in the next comparison. In doing so, $z_k^{(s_k)}$ and $z_k^{(l_k)}$ will only be equal to $-\infty$ if all log-likelihood values are $-\infty$. Note that we do both finding the minimum and finding the maximum in parallel.
7. We copy the two *scalars* $z_k^{(s_k)}$ and $z_k^{(l_k)}$ to CPU memory. On the CPU, we check for equality and abort the update if necessary (lines 12–14)[8]. Then, we compute the step size Δ (lines 15–18), and copy it to the GPU memory.
8. On the GPU, we compute the weights $\tilde{\omega}_k^{(i)}$ in parallel for each sample (line 19). The normalization constant $\tilde{\omega}_k$ is again computed by using the tree-like parallel reduction scheme, now with the sum operator instead of the comparison operators (line 20). The subsequent sample weight normalization is performed in parallel and the resulting weights $\omega_k^{(i)}$ are stored in the vector $\boldsymbol{\omega}_k \in \mathbb{R}^M$ (line 21).
9. We compute the new state mean by simply performing a parallelized matrix-vector multiplication $\hat{\boldsymbol{x}}_{k|k} = \mathbf{M}_k \boldsymbol{\omega}_k$ (line 22). To get the new covariance matrix (line 23), we first do a parallel column-wise subtraction of $\mathbf{M}_k$ with $\hat{\boldsymbol{x}}_{k|k}$, which is stored in the matrix $\mathbf{A}_k \in \mathbb{R}^{N \times M}$. Next, a parallel row-wise multiplication of $\boldsymbol{\omega}_k$ and $\mathbf{A}_k$ yields the matrix $\mathbf{B}_k$. Finally, we do the parallelized matrix-matrix multiplication $\mathbf{P}_{k|k} = \mathbf{B}_k \mathbf{A}_k^\top$.
10. Before the CPU thread launches the next recursion step, we increase γ (line 24).
11. Finally, after all recursion steps are conducted, we transfer the posterior estimate $\hat{\boldsymbol{x}}_{k|k}$ and $\mathbf{P}_{k|k}$ from the GPU memory to the CPU memory.

In summary, during the measurement update we only have interact with the CPU thread to compute the step size Δ and to launch all the GPU tasks. Moreover, only a few scalars are exchanged between CPU memory and GPU memory during the update, namely $z_k^{(s_k)}$, $z_k^{(l_k)}$, and Δ. It should also be noted that newer OpenCL and CUDA versions can launch GPU tasks directly from other GPU tasks. This means that the entire loop (lines 5–25) is executed on the GPU. In doing so, there is no interaction with the CPU thread during a measurement update at all.

5.5 Evaluation

In a first evaluation, we compare the performance of the proposed PGF with various configurations of the PGF 42 and state-of-the-art particle filters by means of estimating pose and shape of a stick target. Second, we track a target in 2D to investigate the effect of the semi-analytic measurement update on the PGF and other Gaussian estimators. After that, we use the PGF to track an airplane in a third evaluation. This includes the derivation of a closed-form likelihood in order to deal with star-convex-shaped extended objects. The last evaluation is dedicated to the GPU-accelerated PGF. We develop a novel likelihood function for tracking pose and extent of a sphere that exploits geometrical relationships between sensor, object, and measurement. Based on this likelihood, we track a sphere with tens of thousands of measurements generated by several simulated Kinect cameras.

[8]Note that the equality check includes the special case $z_k^{(s_k)} = z_k^{(l_k)} = -\infty$, which means $\mathcal{Z}_k$ is empty.

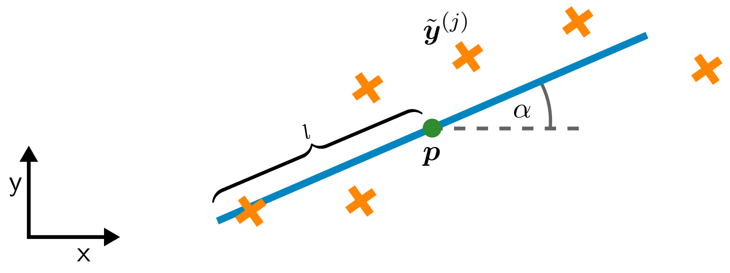


Figure 5.7: Considered stick target model. For better readability, we omitted the time index k.

5.5.1 Tracking a Stick Target in 2D

We consider estimating pose, shape, and motion parameters of a stick target in the xy-plane, e.g., a moving vessel. The target is modeled as shown in Figure 5.7, while the complete system state

$$\boldsymbol{x}_k = [\boldsymbol{p}_k^\top, \alpha_k, l_k, \nu_k, \dot{\alpha}_k]^\top$$

consists of

- position $\boldsymbol{p}_k = [p_k^{(\mathrm{x})}, p_k^{(\mathrm{y})}]^\top$ in m,
- orientation α_k in rad,
- length l_k in m,
- speed ν_k in $\mathrm{m/s}$ along the target's heading, and
- turn rate $\dot{\alpha}_k$ in $\mathrm{rad/s}$.

The target's motion is described by a nonlinear constant velocity/turn rate model and changes in its length by a random walk according to

$$\boldsymbol{x}_k = \boldsymbol{a}(\boldsymbol{x}_{k-1}, \boldsymbol{w}) = \begin{bmatrix} p_{k-1}^{(\mathrm{x})} + \cos(\alpha_{k|k-1}) \cdot \Delta t \cdot \nu_{k|k-1} \\ p_{k-1}^{(\mathrm{y})} + \sin(\alpha_{k|k-1}) \cdot \Delta t \cdot \nu_{k|k-1} \\ \alpha_{k|k-1} \\ l_{k-1} + w^{(l)} \\ \nu_{k|k-1} \\ \dot{\alpha}_{k|k-1} \end{bmatrix} ,$$

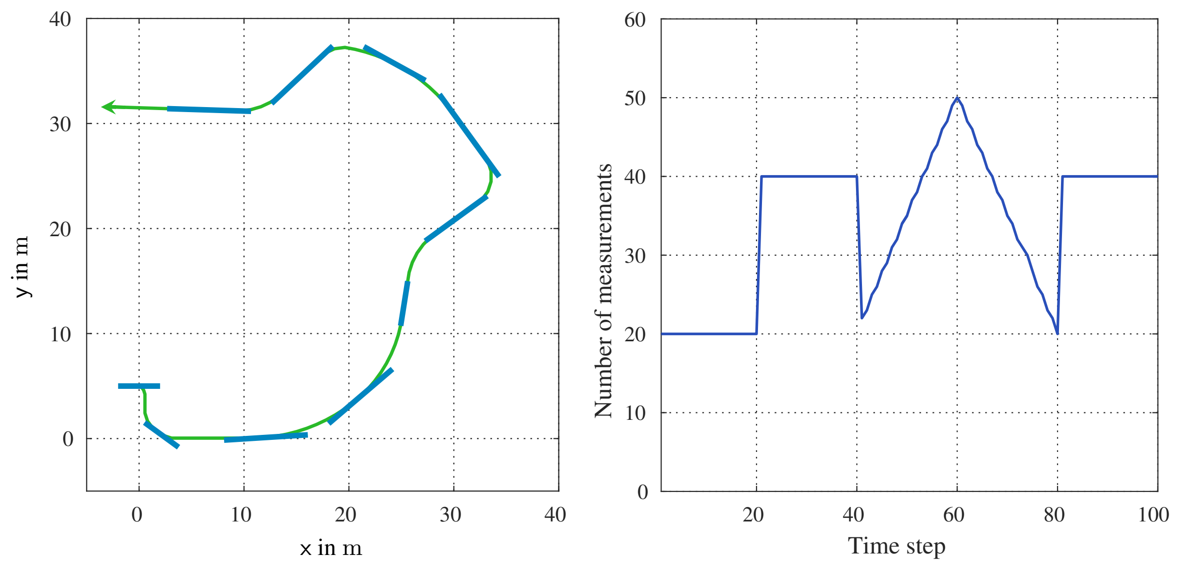
with

$$\begin{aligned} \alpha_{k|k-1} &= \alpha_{k-1} + \Delta t \cdot \dot{\alpha}_{k|k-1} \ , \\ \nu_{k|k-1} &= \nu_{k-1} + w^{(\nu)} \ , \\ \dot{\alpha}_{k|k-1} &= \dot{\alpha}_{k-1} + w^{(\dot{\alpha})} \ , \end{aligned}$$

time period $\Delta t = 0.1\,\mathrm{s}$, and time-invariant zero-mean Gaussian noise $\boldsymbol{w} = [w^{(l)}, w^{(\nu)}, w^{(\dot{\alpha})}]^\top$ with covariance matrix $\mathbf{Q} = \mathrm{diag}(10^{-1}, 5, 10^{-1})$.

We simulate the target's trajectory including changes in its length over 100 time steps as depicted in Figure 5.8(a). In each time step k, we generate Y_k noisy Cartesian measurements $\mathcal{Y}_k = \{\tilde{\boldsymbol{y}}_k^{(j)}\}_{j=1}^{Y_k}$, which are uniformly distributed over the target. Moreover, the amount of measurements is proportional to the current length of the target, that is, a smaller target will result in a smaller number of measurements, see Figure 5.8(b). A measurement $\tilde{\boldsymbol{y}}_k^{(j)}$ is assumed to be generated according to the spatial distribution model [80]

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, u^{(j)}) + \boldsymbol{v}_k^{(j)} = \boldsymbol{p}_k + u^{(j)} \cdot l_k \begin{bmatrix} \cos(\alpha_k) \\ \sin(\alpha_k) \end{bmatrix} + \boldsymbol{v}_k^{(j)} \ ,$$



(a) Nonlinear trajectory (green curve) of a stick target with time-varying length (blue). **(b)** Number of available measurements Y_k.

Figure 5.8: Considered stick target tracking scenario.

with white scaling noise $u^{(j)} \sim \mathcal{U}(-1, 1)$ and additive time-invariant zero-mean white Gaussian noise $\boldsymbol{v}_k^{(j)}$ with covariance matrix $\mathbf{R} = \mathbf{I}_2$. Moreover, $u^{(j)}$ and $\boldsymbol{v}_k^{(j)}$ are mutually independent. The corresponding likelihood function is given by

$$f(\mathcal{Y}_k \,|\, \boldsymbol{x}_k) = \prod_{j=1}^{Y_k} f^{(j)}(\tilde{\boldsymbol{y}}_k^{(j)} \,|\, \boldsymbol{x}_k) \ , \tag{5.22}$$

with individual likelihoods

$$\begin{aligned} f^{(j)}(\tilde{\boldsymbol{y}}_k^{(j)} \,|\, \boldsymbol{x}_k) &= \int_{\mathbb{R}^2} \int_{\mathbb{R}} \delta(\tilde{\boldsymbol{y}}_k^{(j)} - \boldsymbol{h}(\boldsymbol{x}_k, u^{(j)}) - \boldsymbol{v}_k^{(j)}) \cdot \\ &\qquad \mathcal{U}(u^{(j)} \,; -1, 1) \cdot \mathcal{N}(\boldsymbol{v}^{(j)} \,; \mathbf{0}, \mathbf{R}) \, \mathrm{d}u^{(j)} \, \mathrm{d}\boldsymbol{v}^{(j)} \\ &= \int_{\mathbb{R}} \mathcal{N}(\tilde{\boldsymbol{y}}_k^{(j)} - \boldsymbol{h}(\boldsymbol{x}_k, u^{(j)}) \,; \mathbf{0}, \mathbf{R}) \cdot \mathcal{U}(u^{(j)} \,; -1, 1) \, \mathrm{d}u^{(j)} \ . \end{aligned} \tag{5.23}$$

In order to evaluate the likelihood (5.22), we approximate the remaining integral in (5.23) by equidistantly sampling the uniform noise $u^{(j)}$.

We evaluate the following estimators:

- the PGF,
- the PGF 42 with forced sample weight ratio $R = 0.1$,
- the PGF 42 using the likelihood (5.22) and forced sample weight ratio $R = \frac{1}{101}$,
- the PGF 42 using the likelihood (5.22) and forced sample weight ratio $R = 0.1$,
- the PGF 42 using the likelihood (5.22) and forced sample weight ratio $R = 0.5$,
- the GPF with 10^4 particles,
- the SIRPF with 10^4 particles and resampling threshold of 0.9 (normalized ESS), and

- the RPF with 10^4 particles and resampling threshold of 0.9 (normalized ESS).

Although the original PGF 42 does not work with a given likelihood function, we configure some PGF 42 instances to directly use the likelihood (5.22) instead of estimating the non-additive noise variables $u^{(j)}$. Thus, those filters work more like the proposed PGF, but still rely on the forward/backward updates and user-defined forced sample weight ratios R. The idea behind these PGF 42 variants is to allow for a more comprehensive evaluation of the proposed PGF.

Due to the time-varying number of measurements, the PGF 42 also estimates a time-varying joint space $[\boldsymbol{x}_k^\top, u^{(1)}, \ldots, u^{(Y_K)}]^\top$ of $D_k = 6 + Y_k$ dimensions. We configure the PGF 42 to use an adaptive number of $10 \cdot D_k$ samples for a measurement update, i.e., depending on the dimension D_k. The PGF and the PGF 42 variants using the likelihood function use 101 samples for a measurement update. Moreover, the PGF and all PGF 42 variants employ 91 samples for the nonlinear time update. In addition, for all PGF 42 variants we have to quantify the maximum allowed deviation between successive intermediate Gaussians in order to accept or reject a computed step size Δ. We choose to rely on the maximum norm and accept a step size if

$$\left\|\hat{\boldsymbol{x}}_{k|k}(\gamma) - \hat{\boldsymbol{x}}_{k|k}(\gamma + \Delta)\right\|_\infty \leq 0.5$$

and

$$\left\|\mathbf{P}_{k|k}(\gamma) - \mathbf{P}_{k|k}(\gamma + \Delta)\right\|_\infty \leq 0.5$$

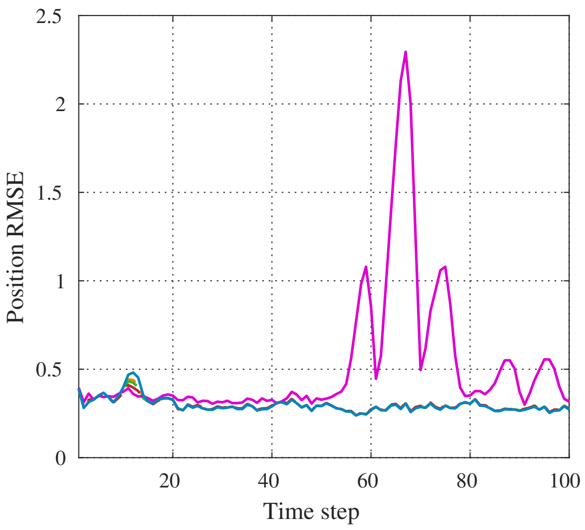
holds.

We perform 200 Monte Carlo runs on a system with Intel Core i7-3770 CPU (3.4 GHz, 4 cores, 8 threads). Additionally, the evaluation of the likelihood function is effectively parallelized with OpenMP for all estimators. In each run, we initialize the estimators with the density

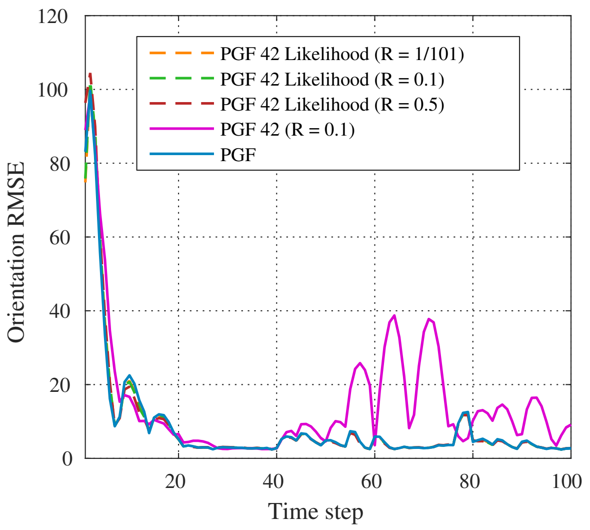
$$f_{0|0}(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0\,; [0, 0, 0, 1, 0, 0]^\top, \mathrm{diag}(10\,\mathbf{I}_2, 0.5, 1, 0.1, 0.1)) \enspace .$$

First, we compare the PGF with the different PGF 42 variants regarding estimation quality, number of performed recursion steps, and runtime. In Figures 5.9(a)–5.9(c) the position RMSE, the orientation RMSE, and the length RMSE are shown. On the one hand, it can be seen that the original PGF 42 has certain problems estimating the stick target's pose and shape. On the other hand, all other estimators perform very well on a quite similar level. Only at time steps where the target abruptly changes its shape, i.e., $k = 20$, $k = 40$, and $k = 80$, their length errors increase for a short time. From this, we can conclude that the additional forward/backward updates of the PGF 42 do not yield any significant benefit regarding the estimation quality. Also the forced sample weight ratio R has only a marginal effect on the estimation performance, e.g., see the position errors and orientation errors at time step $k = 10$.

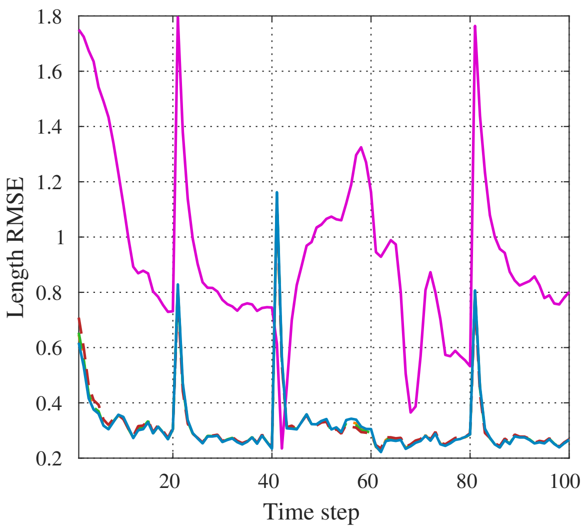
However, when looking at the average number of recursion steps required by each filter (see Figure 5.9(d)) we notice exceptional differences. Even though the PGF 42 with ratio $R = \frac{1}{101}$ uses the same ratio as the PGF (due to the PGF's heuristic selecting R) it needs approximately twice the number of recursion steps for a measurement update due to the forward/backward updates. Moreover, the PGF 42 with ratio $R = 0.5$ requires even more recursion steps, while it has only a limited effect on the estimation quality. These results also underline the fact that a larger ratio R leads to smaller step sizes Δ and consequently to more recursion steps. Furthermore, the original PGF 42 need significantly more recursion steps than the other estimators. Its number of recursion steps heavily depends on the number of measurements Y_k as the amount of variables D_k to be estimated also depends on Y_k (compare Figures 5.8(b) and 5.9(d)).



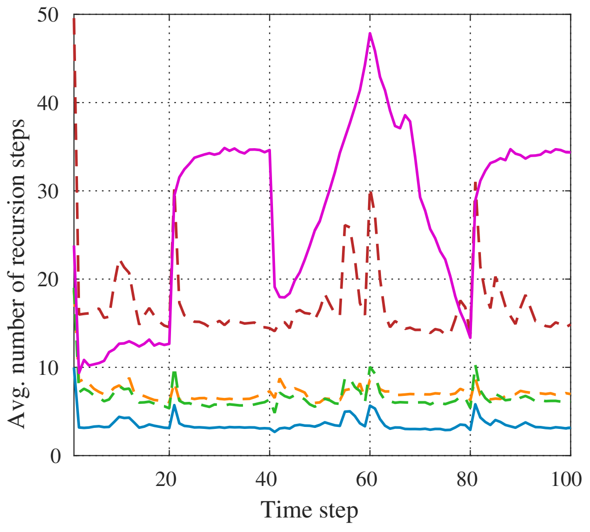
(a) Position RMSE in m.



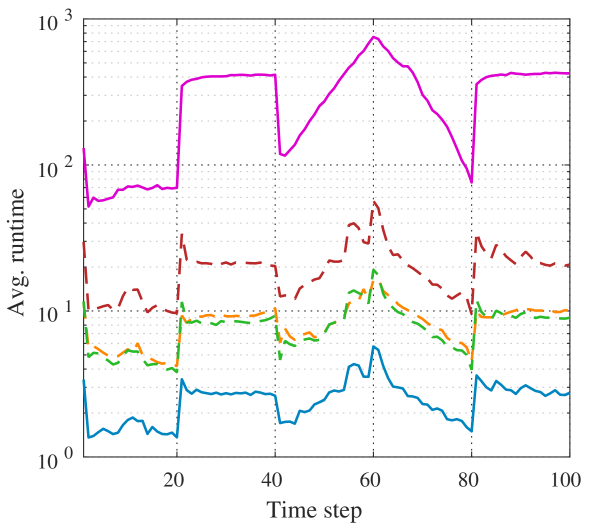
(b) Orientation RMSE in °.



(c) Length RMSE in m.

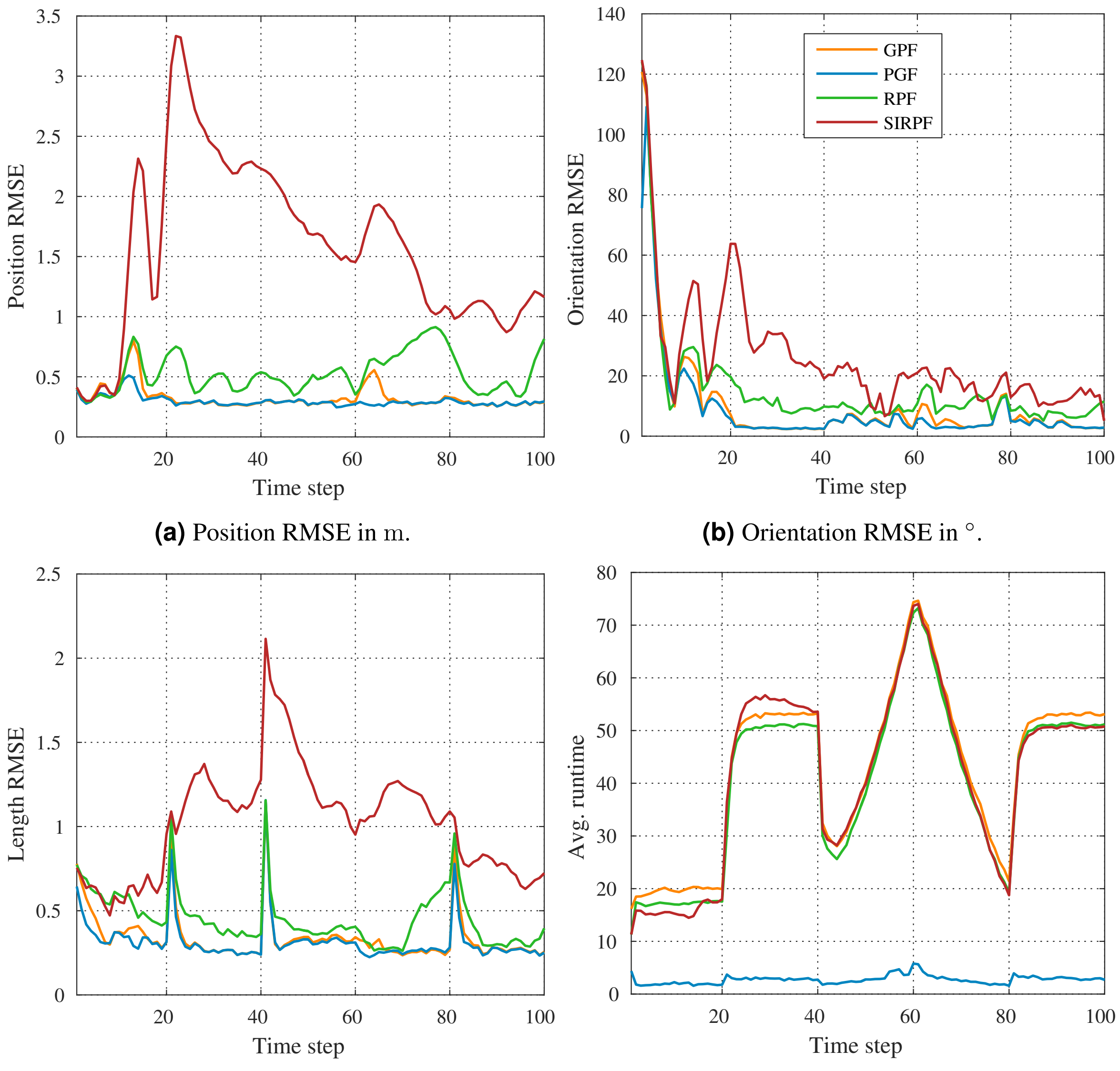


(d) Average number of recursion steps.



(e) Average runtimes in ms.

Figure 5.9: Results of the stick target tracking evaluation (PGF vs. PGF 42).



(a) Position RMSE in m. **(b)** Orientation RMSE in $°$. **(c)** Length RMSE in m. **(d)** Average runtimes in ms.

Figure 5.10: Results of the stick target tracking evaluation (PGF vs. particle filters).

Finally, we analyze the average filter runtimes depicted in Figure 5.9(e). First of all, these reflect the individual number of required recursion steps. Additionally, the runtimes exhibit a strong correlation with the number of measurements as the likelihood evaluations themselves require more time for a larger amount of measurements. Nonetheless, a measurement update of the PGF 42 takes significantly longer than the updates of all other estimators. This can be explained by its larger amount of used samples and an overall larger number of recursion steps.

Now, we analyze how the PGF compares to the particle filters. Again, we first assess the respective estimation quality by means of position RMSE, orientation RMSE, and length RMSE, see Figures 5.10(a)–5.10(c). Regarding all three quality measures, the SIRPF is the worst of all estimators. The RPF delivers much better results, but cannot reach the quality of the well performing GPF and PGF. Nevertheless, the PGF performs always a little bit better than the GPF. When additionally taking the filter runtimes (see Figure 5.10(d)) into account, the PGF with its remarkable fast execution time is by far the best of all estimators. Moreover, the PGF's runtime is nearly unaffected by the changing amount of measurements, whereas the runtimes of the particle filters are highly correlated with the number of processed measurements. Although the studied particle filters have different resampling strategies, all of them have almost the same execution times due to the same number of used particles.

5.5.2 Tracking a Target in 2D

Next, we analyze the impact of the proposed semi-analytic measurement update on both estimation quality and runtime of the PGF. This is done by a target tracking scenario in the xy-plane. The target is described by the hidden system state

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{x}_k^{(o)} \\ \boldsymbol{x}_k^{(u)} \end{bmatrix}$$

consisting of

- target position $\boldsymbol{x}_k^{(o)} = [p_k^{(\mathrm{x})}, p_k^{(\mathrm{y})}]^\top$ in m and
- their velocities and accelerations $\boldsymbol{x}_k^{(u)} = [\dot{p}_k^{(\mathrm{x})}, \dot{p}_k^{(\mathrm{y})}, \ddot{p}_k^{(\mathrm{x})}, \ddot{p}_k^{(\mathrm{y})}]^\top$ in $\mathrm{m/s}$ and $\mathrm{m/s^2}$, respectively.

Analogous to the distributed target tracking in Section 4.4.1, we model the target's motion with a constant acceleration model according to

$$\boldsymbol{x}_k = \mathbf{A}\boldsymbol{x}_{k-1} + \mathbf{B}\boldsymbol{w} \ , \tag{5.24}$$

with matrices

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_2 & \Delta t\,\mathbf{I}_2 & \frac{1}{2}\Delta t^2\,\mathbf{I}_2 \\ \mathbf{0} & \mathbf{I}_2 & \Delta t\,\mathbf{I}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_2 \end{bmatrix} \ , \quad \mathbf{B} = \begin{bmatrix} \frac{1}{2}\Delta t^2\,\mathbf{I}_2 \\ \Delta t\,\mathbf{I}_2 \\ \mathbf{I}_2 \end{bmatrix} \ ,$$

time period $\Delta t = 0.01\,\mathrm{s}$, and time-invariant zero-mean white Gaussian noise with covariance matrix $\mathbf{Q} = \mathrm{diag}(10^{-2}, 10^{-2})$.

Over time, we get a noisy position measurement in polar coordinates according to

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k^{(o)}) + \boldsymbol{v} = \begin{bmatrix} \sqrt{\left(p_k^{(\mathrm{x})}\right)^2 + \left(p_k^{(\mathrm{y})}\right)^2} \\ \mathrm{atan2}(p_k^{(\mathrm{y})}, p_k^{(\mathrm{x})}) \end{bmatrix} + \boldsymbol{v} \ , \tag{5.25}$$

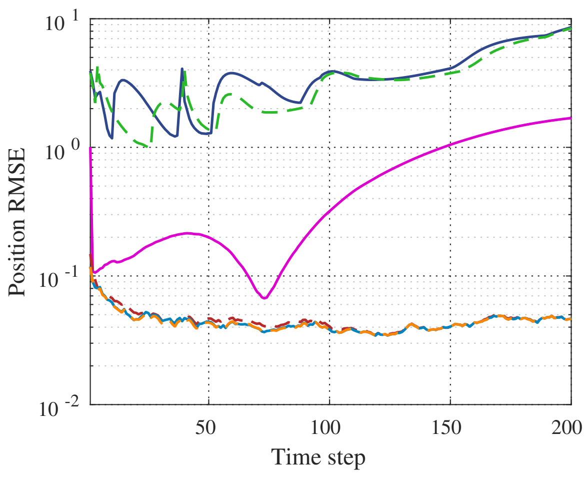
where $\mathrm{atan2}$ denotes the four-quadrant inverse tangent and $\boldsymbol{v}$ time-invariant zero-mean white Gaussian noise with covariance matrix $\mathbf{R} = \mathrm{diag}(10^{-2}, 10^{-4})$. The corresponding likelihood function is given in closed form by

$$f(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k^{(o)}) = \mathcal{N}(\tilde{\boldsymbol{y}}_k - \boldsymbol{h}(\boldsymbol{x}_k^{(o)})\,; \mathbf{0}, \mathbf{R}) \ . \tag{5.26}$$

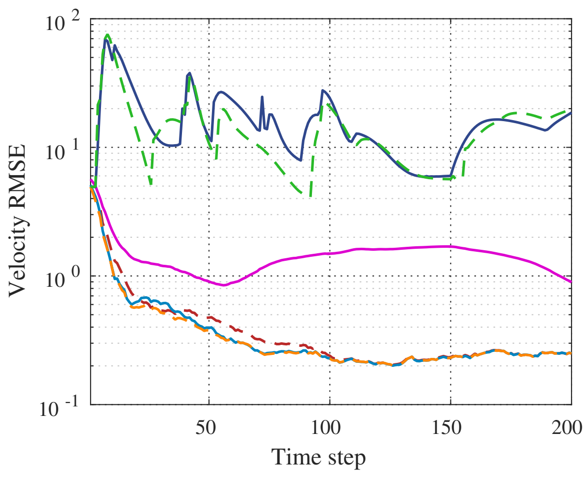
From (5.26), we see that $\boldsymbol{x}_k^{(o)}$ contains the observable variables of the system state, while $\boldsymbol{x}_k^{(u)}$ comprises the unobservable ones.

We evaluate the following estimators:

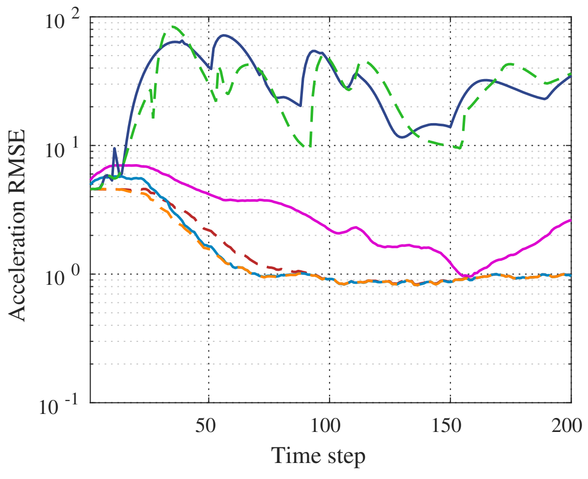
- the PGF with $M = 101$ samples,
- the semi-analytic PGF with $M = 51$ samples,
- the S^{2}KF with 101 samples,
- the semi-analytic S^{2}KF with 101 samples,
- the GPF with 10^4 particles, and finally
- the semi-analytic GPF also with 10^4 particles.



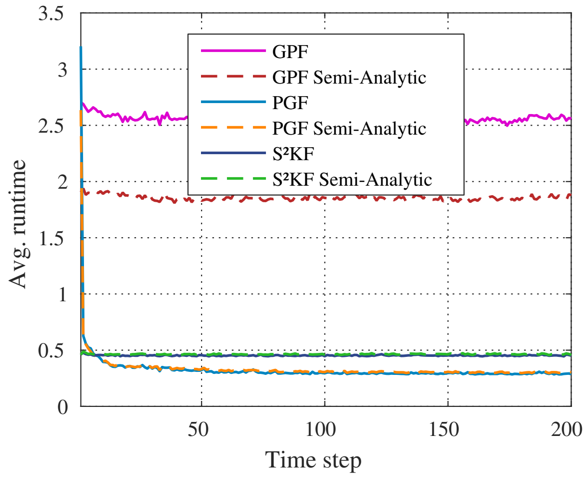
(a) Position RMSE in m.



(b) Velocity RMSE in $\mathrm{m/s}$.



(c) Acceleration RMSE in $\mathrm{m/s^2}$.



(d) Average runtimes in ms.

Figure 5.11: Results of the target tracking evaluation.

That is, we also execute a S^{2}KF and a GPF with semi-analytic measurement updates. Due to the linear system model (5.24), the state prediction is computed in closed form for all filters. We perform 100 Monte Carlo runs on a system with Intel Core i7-3770 CPU (3.4 GHz, 4 cores, 8 threads). In each run, we initialize all filters with

$$f_{0|0}(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0\,; [1, 1, 0, 0, 0, 0]^\top, 10\,\mathbf{I}_6) \enspace .$$

Additionally, the true system state is initialized by randomly drawing a realization of $f_{0|0}$. Then, we simulate the temporal behavior of the system for 200 time steps by propagating the current system state, together with a random realization of $\boldsymbol{w}$, through the system model (5.24). Furthermore, in each time step we simulate a noisy measurement $\tilde{\boldsymbol{y}}_k$ by evaluating (5.25) together with a random realization of $\boldsymbol{v}$.

We compare the performance of the estimators by computing the RMSE of target position $[p_k^{(\mathrm{x})}, p_k^{(\mathrm{y})}]^\top$, target velocity $[\dot{p}_k^{(\mathrm{x})}, \dot{p}_k^{(\mathrm{y})}]^\top$, and target acceleration $[\ddot{p}_k^{(\mathrm{x})}, \ddot{p}_k^{(\mathrm{y})}]^\top$, respectively. The results are shown in Figures 5.11(a)–5.11(c). First of all, we notice that the S^{2}KF completely fails to estimate the system state. Its semi-analytic variant can only slightly mitigate this, if at all. Please note that the S^{2}KF acts here as representative estimator for the class of Kalman filters. Other Kalman filters should perform similarly. Further, the GPF also has serious estimation problems, especially for the position.

However, the semi-analytic GPF can estimate all state variables quite well. Regarding the PGF, both variants perform also very well. In the first 100 time steps, they are better than the semi-analytic GPF. Especially, the acceleration error of the semi-analytic PGF is even smaller than the error of the standard PGF, although it uses only half the number of samples.

In Figure 5.11(d), the average filter update runtimes are plotted. Here, we see that the GPF with its many particles has the worst runtime. Interestingly, the semi-analytic GPF with the same number of particles performs better. This can be explained with the reduced overhead for computing sample mean and sample covariance to obtain the posterior state estimate, i.e., now with $N = 2$ instead of $N = 6$ as the other parts are handled analytically. For the PGF, such effect cannot be seen, but we notice the adaptively chosen number of recursion steps. In the beginning, where the uncertainty is large, we are even slower than the GPF. However, after a few time steps the PGF converges to a very fast execution. Thus, for this scenario its semi-analytic version yields no runtime improvement due to the rather small system state and the small amount of samples. Nevertheless, the PGFs are faster than both S^{2}KFs (after convergence). All in all, the semi-analytic PGF delivers the best estimation quality, while being the fastest estimator.

5.5.3 Tracking an Airplane in 2D

In this evaluation, we consider estimating pose and shape of a 2D extended object based on noisy Cartesian measurements originating from its surface. Compared to the evaluation from Section 5.5.1, where we knew that the object was shaped like a stick, here we have no prior information about the object's shape except that it maybe exhibit some sort of symmetry.

Common approaches to deal with an a priori unknown object shape is to approximate it

- as an ellipse using random matrices [81, 82] or elliptic RHMs [83],
- as a star-convex shape using star-convex RHMs [84, 88, 150, 163], or
- as a non-convex polygon using level-set RHMs [87, 89].

The advantage of a star-convex RHM is its quite effective shape approximation, while being rather simple to implement. Nevertheless, state-of-the-art approaches use star-convex RHMs only in combination with linear filters. Thus, in order to allow for an improved estimation quality, our goal is to apply the PGF to a star-convex RHM. However, a computationally efficient implementation requires a closed-form likelihood function. Consequently, in the following we first introduce star-convex RHMs, second derive a closed-form likelihood for star-convex RHMs, and finally use it for tracking pose and shape of a moving airplane.

Star-Convex Random Hypersurface Models

When tracking pose and shape of a 2D extended object with the aid of a star-convex RHM, the object's pose is described by position $\boldsymbol{p}_k = [p_k^{(\mathrm{x})}, p_k^{(\mathrm{y})}]^\top$ and orientation α_k, while the object's shape is described by a radial function that specifies the distance from the object position to any point on the object boundary. This radial function is given by the Fourier series

$$r(\boldsymbol{c}_k, \phi) = \frac{a_k^{(0)}}{2} + \sum_{j=1}^{F} a_k^{(j)} \cos(j\phi) + b_k^{(j)} \sin(j\phi)$$

that is parameterized with a fixed number of $2F + 1$, $F \in \mathbb{N}$, Fourier coefficients

$$\boldsymbol{c}_k = \left[a_k^{(0)}, a_k^{(1)}, b_k^{(1)}, \ldots, a_k^{(F)}, b_k^{(F)}\right]^\top \in \mathbb{R}^{2F+1}$$

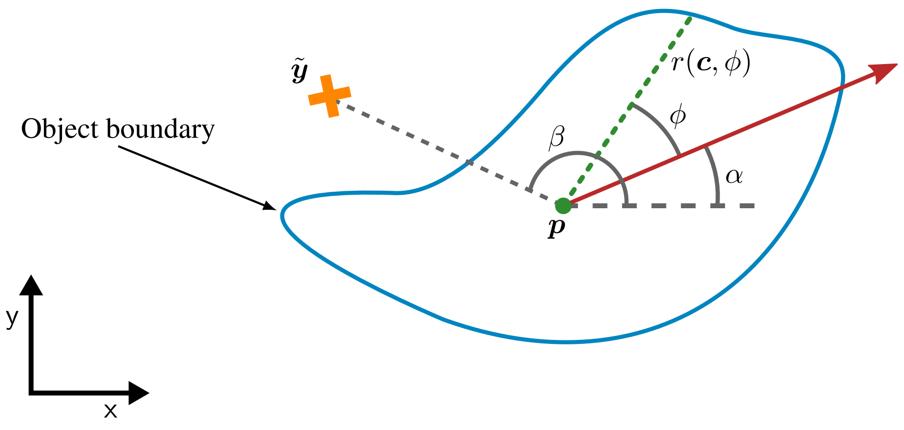


Figure 5.12: Star-convex random hypersurface model. For better readability, the time index k is omitted.

and is evaluated for an angle $\phi \in \mathbb{R}$. The object surface is then given by the set of all points within the object boundary, see Figure 5.12. Consequently, the system state of the extended object encompasses the shape and pose parameters according to

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{c}_k \\ \boldsymbol{p}_k \\ \alpha_k \end{bmatrix} \quad ,$$

where the number of Fourier coefficients has to be determined a priori by the user.

In order to estimate the system state $\boldsymbol{x}_k$ using a Cartesian measurement $\tilde{\boldsymbol{y}}_k = [\tilde{y}_k^{(\mathrm{x})}, \tilde{y}_k^{(\mathrm{y})}]^\top$ that originates from the object's surface, we rely on the nonlinear measurement model

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \tilde{\boldsymbol{y}}_k, s) + \boldsymbol{v}_k = \boldsymbol{p}_k + s \cdot r(\boldsymbol{c}_k, \psi_k) \begin{bmatrix} \cos(\beta_k) \\ \sin(\beta_k) \end{bmatrix} + \boldsymbol{v}_k \quad , \tag{5.27}$$

with white multiplicative noise $s \in [0,1]$, additive zero-mean white Gaussian noise $\boldsymbol{v}_k$ with covariance matrix $\mathbf{R}_k$, and

$$\begin{aligned} \psi_k &= \beta_k - \alpha_k \quad , \\ \beta_k &= \operatorname{atan2}(d_k^{(\mathrm{y})}, d_k^{(\mathrm{x})}) \quad , \\ \boldsymbol{d}_k &= [d_k^{(\mathrm{x})}, d_k^{(\mathrm{y})}]^\top = \tilde{\boldsymbol{y}}_k - \boldsymbol{p}_k \quad , \end{aligned}$$

see Figure 5.12. As the radial function solely represents the object boundary, the multiplicative noise s scales r to be able to cover all points on the object surface, not only the points on its boundary. In addition, it is assumed that s is independent of the system state and the additive noise $\boldsymbol{v}_k$. Also note that the measurement model (5.27) depends on the received measurement $\tilde{\boldsymbol{y}}_k$, i.e., we use a greedy measurement association like we already did for the cylinder tracking in Section 3.4.2.

What is still missing is a proper distribution of the multiplicative noise s. Here, we assume that measurements are uniformly distributed over the entire object shape, which leads to the multiplicative noise $s = g(u) = \sqrt{u}$, with $u \sim \mathcal{U}(0,1)$, see [83, 163]. According to the fundamental theorem of transforming a random variable, e.g., see [106, Sec. 5.2], the PDF of s is given by

$$f^s(s) = \frac{1}{|g'(s^2)|} \mathcal{U}(s^2\,; 0,1) = 2s \cdot \mathcal{U}(s^2\,; 0,1) = \begin{cases} 2s & , s \in [0,1] \\ 0 & , \text{elsewhere} \end{cases} \quad .$$

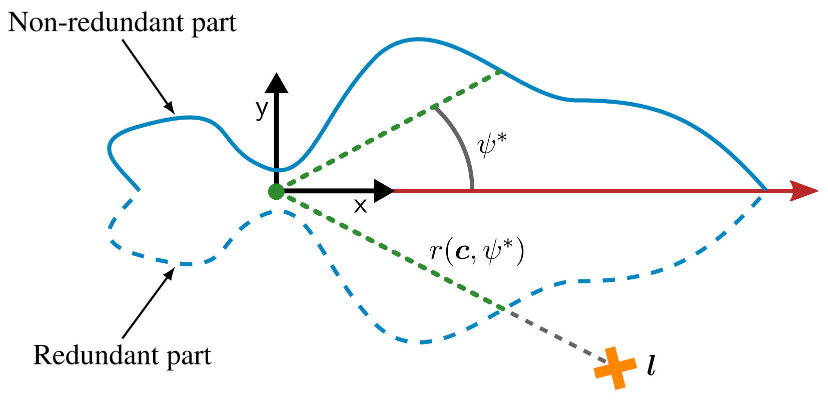


Figure 5.13: Star-convex random hypersurface model with x-axial symmetry. For better readability, the time index k is omitted.

Based on this star-convex RHM, we derived a closed-form likelihood function in [185], which is easily applicable to nonlinear filters. However, in this thesis we additionally want to exploit possibly known object shape reflection symmetries in order to improve the overall estimation quality as it is done for linear filters in [163]. For simplicity, we merely consider a 1-axial reflection symmetry. Specifically, we assume that the object shape is symmetric with respect to the object's local x-axis. Nevertheless, this approach can be extended more sophisticated reflection symmetries as shown in [163].

We incorporate the desired x-axial symmetry by first rewriting the measurement model (5.27). More precisely, we transform the received measurement $\tilde{\boldsymbol{y}}_k$ to the object's local coordinate system according to

$$\boldsymbol{l}_k = \begin{bmatrix} l_k^{(\mathsf{x})} \\ l_k^{(\mathsf{y})} \end{bmatrix} = \begin{bmatrix} \cos(\alpha_k) & \sin(\alpha_k) \\ -\sin(\alpha_k) & \cos(\alpha_k) \end{bmatrix} \boldsymbol{d}_k \ .$$

With $\boldsymbol{l}_k$, we can then compute the angle

$$\psi_k = \operatorname{atan2}(l_k^{(\mathsf{y})}, l_k^{(\mathsf{x})})$$

and also avoid the computation of β_k due to

$$\begin{bmatrix} \cos(\beta_k) \\ \sin(\beta_k) \end{bmatrix} = \frac{\boldsymbol{d}_k}{\|\boldsymbol{d}_k\|_2} \ .$$

Based on these, we get an equivalent formulation of (5.27) according to

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \tilde{\boldsymbol{y}}_k, s) + \boldsymbol{v}_k = \boldsymbol{p}_k + s \cdot r(\boldsymbol{c}_k, \psi_k) \frac{\boldsymbol{d}_k}{\|\boldsymbol{d}_k\|_2} + \boldsymbol{v}_k \ . \tag{5.28}$$

Second, we split the object boundary, and thus the entire object shape, in a non-redundant part and a redundant part. The non-redundant part consists of all points with a non-negative y component, while the redundant part encompasses all points with a negative y component. Third, we take the absolute value of $l_k^{(\mathsf{y})}$ in order to flip measurements $\boldsymbol{l}_k$ from the redundant part into the non-redundant part and determine the angle for evaluating the radial function according to

$$\psi_k^* = \operatorname{atan2}(|l_k^{(\mathsf{y})}|, l_k^{(\mathsf{x})}) \ .$$

By replacing ψ_k with ψ_k^* in (5.28), we have completely implemented the x-axial symmetry, see Figure 5.13. In doing so, we effectively evaluate r solely for angles $\psi_k^* \in [0, \pi]$ and mirror the radial function at the local x-axis.

Closed-Form Likelihood for Nonlinear Filters

After discussing the generative model for (symmetric) star-convex RHMs, we have to derive a closed-form likelihood function from it. Using the sifting property of the Dirac-δ distribution, the likelihood for (5.28), including its symmetric variant, is given by

$$\begin{aligned} f(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) &= \int_{\mathbb{R}} \int_{\mathbb{R}^2} \delta(\tilde{\boldsymbol{y}}_k - (\boldsymbol{h}(\boldsymbol{x}_k, \tilde{\boldsymbol{y}}_k, s) + \boldsymbol{v}_k)) \cdot \mathcal{N}(\boldsymbol{v}_k \,; \boldsymbol{0}, \mathbf{R}_k) \cdot f^s(s) \, \mathrm{d}\boldsymbol{v}_k \, \mathrm{d}s \\ &= \int_{\mathbb{R}} \mathcal{N}(\tilde{\boldsymbol{y}}_k - \boldsymbol{h}(\boldsymbol{x}_k, \tilde{\boldsymbol{y}}_k, s) \,; \boldsymbol{0}, \mathbf{R}_k) \cdot f^s(s) \, \mathrm{d}s \ . \end{aligned} \tag{5.29}$$

Also the remaining integral in (5.29) *can* be solved analytically leading to a closed-form likelihood, where analytically still implies evaluations of the error function erf. However, the resulting expression (and especially its $\log$-likelihood) is not numerically stable for larger differences $\tilde{\boldsymbol{y}}_k - \boldsymbol{h}$. Hence, to circumvent this, we approximate the triangle distribution of s as Gaussian distribution by means of moment matching like it is usually done when using RHMs in combination Kalman filters. That is, we use the approximation

$$f^s(s) \approx \mathcal{N}(s \,; \hat{s}, \Sigma^{(s)}) \ , \tag{5.30}$$

with mean $\hat{s} = \frac{2}{3}$ and variance $\Sigma^{(s)} = \frac{1}{18}$. By replacing f^s with (5.30) in (5.29), we get the likelihood function

$$f^{(\mathrm{sc})}(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) = \int_{\mathbb{R}} \mathcal{N}(\tilde{\boldsymbol{y}}_k - \boldsymbol{h}(\boldsymbol{x}_k, \tilde{\boldsymbol{y}}_k, s) \,; \boldsymbol{0}, \mathbf{R}_k) \cdot \mathcal{N}(s \,; \hat{s}, \Sigma^{(s)}) \, \mathrm{d}s \ . \tag{5.31}$$

for which a numerically stable closed-form solution can be obtained.

Theorem 5.1: Star-Convex RHM Likelihood
A closed-form solution for the star-convex RHM likelihood function (5.31) *is given by*

$$f^{(sc)}(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) = \frac{z_k}{\sqrt{1 + u_k \Sigma^{(s)}}} \exp\left(-\frac{1}{2} \frac{\left(\hat{s} - \frac{t_k}{u_k}\right)^2}{\left(\frac{1}{u_k} + \Sigma^{(s)}\right)} \right) \ , \tag{5.32}$$

where

$$\begin{aligned} z_k &= \frac{1}{2\pi \sqrt{|\mathbf{R}_k|}} \exp\left(-\frac{1}{2} \left(q_k - \frac{t_k^2}{u_k} \right) \right) \ , \\ u_k &= \boldsymbol{b}_k^\top \mathbf{R}_k^{-1} \boldsymbol{b}_k \ , \\ t_k &= \boldsymbol{d}_k^\top \mathbf{R}_k^{-1} \boldsymbol{b}_k \ , \\ q_k &= \boldsymbol{d}_k^\top \mathbf{R}_k^{-1} \boldsymbol{d}_k \ , \\ \boldsymbol{b}_k &= r(\boldsymbol{c}_k, \psi_k) \frac{\boldsymbol{d}_k}{\|\boldsymbol{d}_k\|_2} \ , \\ r(\boldsymbol{c}_k, \psi_k) &= \frac{a_k^{(0)}}{2} + \sum_{j=1}^{F} a_k^{(j)} \cos(j\psi_k) + b_k^{(j)} \sin(j\psi_k) \ , \\ \psi_k &= \begin{cases} \operatorname{atan2}(|l_k^{(\mathrm{y})}|, l_k^{(\mathrm{x})}) & \textit{, if use } \mathrm{x}\textit{-axial symmetry} \\ \operatorname{atan2}(l_k^{(\mathrm{y})}, l_k^{(\mathrm{x})}) & \textit{, otherwise} \end{cases} \ , \\ \boldsymbol{l}_k &= \begin{bmatrix} \cos(\alpha_k) & \sin(\alpha_k) \\ -\sin(\alpha_k) & \cos(\alpha_k) \end{bmatrix} \boldsymbol{d}_k, \ , \\ \boldsymbol{d}_k &= \tilde{\boldsymbol{y}}_k - \boldsymbol{p}_k \ . \end{aligned}$$

Proof. The proof is given in Appendix D. ■

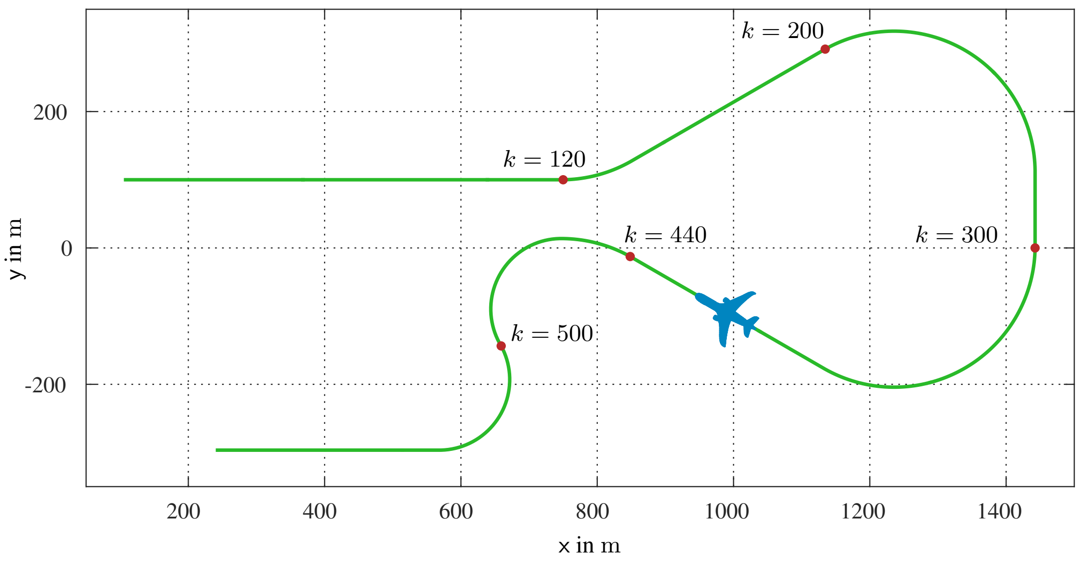


Figure 5.14: Nonlinear trajectory (green curve) of the airplane to be estimated (blue). Important changes in the trajectory are marked with red dots.

Note that the logarithm of the closed-form star-convex RHM likelihood (5.32) can be effectively computed according to

$$\log\left(f^{(\mathrm{sc})}(\tilde{\boldsymbol{y}}_k \mid \boldsymbol{x}_k)\right) = -\log(2\pi) - \frac{1}{2}\log(|\mathbf{R}_k|) - \frac{1}{2}\left(q_k - \frac{t_k^2}{u_k}\right)$$
$$- \frac{1}{2}\operatorname{log1p}(u_k\Sigma^{(s)}) - \frac{1}{2}\frac{\left(\hat{s} - \frac{t_k}{u_k}\right)^2}{\left(\frac{1}{u_k} + \Sigma^{(s)}\right)} ,$$

where $\operatorname{log1p}(x)$ denotes the more accurate implementation of $\log(1+x)$. An implementation of $\operatorname{log1p}(x)$ can be found, for example, in the C standard library or in MATLAB.

Simulation

We evaluate the proposed analytic star-convex RHM likelihood by means of tracking pose and shape of a moving airplane. The airplane has a wingspread of $100\,\mathrm{m}$ and flies with a constant velocity along a nonlinear path for 600 time steps as depicted in Figure 5.14. Note that the airplane does not exhibit a star-convex shape. Hence, the estimates based on the star-convex RHM can merely be an approximation of the true airplane shape.

For an adequate state prediction, we extend the system state with additional motion parameters leading to the 20D system state

$$\boldsymbol{x}_k = \left[\boldsymbol{c}_k^\top, \boldsymbol{p}_k^\top, \alpha_k, \nu_k, \dot{\alpha}_k\right]^\top ,$$

consisting of

- 15 Fourier coefficients, i.e., $F = 7$, $\boldsymbol{c}_k$ in m,
- the airplane position $\boldsymbol{p}_k$ in m,
- the airplane orientation α_k in rad,
- the airplane speed ν_k in $\mathrm{m/s}$ along the its heading, and
- the airplane turn rate $\dot{\alpha}_k$ in $\mathrm{rad/s}$.

Analogous to the stick target tracking in Section 5.5.1, we describe the airplane's motion with a nonlinear constant velocity/turn rate model and changes in the Fourier coefficients with a random walk according to

$$\boldsymbol{x}_k = \boldsymbol{a}(\boldsymbol{x}_{k-1}, \boldsymbol{w}) = \begin{bmatrix} \boldsymbol{c}_{k-1} + \boldsymbol{w}^{(c)} \\ p^{(\mathrm{x})}_{k-1} + \cos(\alpha_{k|k-1}) \cdot \Delta t \cdot \nu_{k|k-1} \\ p^{(\mathrm{y})}_{k-1} + \sin(\alpha_{k|k-1}) \cdot \Delta t \cdot \nu_{k|k-1} \\ \alpha_{k|k-1} \\ \nu_{k|k-1} \\ \dot{\alpha}_{k|k-1} \end{bmatrix} ,$$

with

$$\begin{aligned} \alpha_{k|k-1} &= \alpha_{k-1} + \Delta t \cdot \dot{\alpha}_{k|k-1} \ , \\ \nu_{k|k-1} &= \nu_{k-1} + w^{(\nu)} \ , \\ \dot{\alpha}_{k|k-1} &= \dot{\alpha}_{k-1} + w^{(\dot{\alpha})} \ , \end{aligned}$$

time period $\Delta t = 1\,\mathrm{s}$, and time-invariant zero-mean white Gaussian noise $\boldsymbol{w} = [(\boldsymbol{w}^{(c)})^\top, w^{(\nu)}, w^{(\dot{\alpha})}]^\top$ with covariance matrix $\mathbf{Q} = \mathrm{diag}(10^{-2}\,\mathbf{I}_{15}, 10^{-7}, 10^{-5})$.

In each time step k, we get a set of $Y = 200$ random measurements $\mathcal{Y}_k = \{\tilde{\boldsymbol{y}}_k^{(j)}\}_{j=1}^{Y}$ that originate from the airplane's surface and are disturbed by additive noise with covariance matrix $\mathbf{R} = \mathbf{I}_2$. We process all measurements in a single filter step. Nonlinear filters use the log-likelihood

$$\log\big(f(\mathcal{Y}_k \,|\, \boldsymbol{x}_k)\big) = \sum_{j=1}^{Y} \log\big(f^{(\mathrm{sc})}(\tilde{\boldsymbol{y}}_k^{(j)} \,|\, \boldsymbol{x}_k)\big) \ ,$$

while the S^{2}KF use a stacked measurement vector in combination with the signed-distance-based star-convex RHM proposed in [163]. Further, as we know that the airplane to be tracked has a reflection symmetry with respect to its longitudinal axis, we use the 1-axial symmetric star-convex RHMs for both the nonlinear filters and the S^{2}KF.

We evaluate the following nonlinear and linear estimators:

- the PGF with 201 and 401 samples for prediction and filter step, respectively,
- the GPF with 10^5 particles,
- the RPF with 10^5 particles and resampling threshold 0.9 (normalized ESS), and
- the S^{2}KF with 201 and 2 001 samples for prediction and filter step, respectively.

We perform 100 Monte Carlo runs on a system with Intel Core i7-3770 CPU (3.4 GHz, 4 cores, 8 threads). Additionally, the evaluations of the likelihood function and the stacked measurement equation are effectively parallelized with OpenMP. In each run, we initialize all estimators based on the first set of available measurements $\mathcal{Y}_0$. Specifically, we compute mean and covariance matrix

$$\begin{aligned} \hat{\boldsymbol{p}} &= \frac{1}{Y} \sum_{j=1}^{Y} \tilde{\boldsymbol{y}}_0^{(j)} \ , \\ \boldsymbol{\Sigma}^{(p)} &= \frac{1}{Y} \sum_{j=1}^{Y} \big(\tilde{\boldsymbol{y}}_0^{(j)} - \hat{\boldsymbol{p}}\big)\big(\tilde{\boldsymbol{y}}_0^{(j)} - \hat{\boldsymbol{p}}\big)^\top \ , \end{aligned}$$

for the initial airplane position, and the largest distance

$$d_{\max} = \max_{j=1,\ldots,Y} \|\tilde{\boldsymbol{y}}_0^{(j)} - \hat{\boldsymbol{p}}\|_2$$

for the initial shape estimate. With these, we construct an initial Gaussian estimate

$$f_{0|0}(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0\,;\hat{\boldsymbol{x}}_{0|0}, \mathbf{P}_{0|0}) \ ,$$

where

$$\begin{aligned}\hat{\boldsymbol{x}}_{0|0} &= [2d_{\max}, 0, \ldots, 0, \hat{\boldsymbol{p}}^\top, 0, 0, 0]^\top \ , \\ \mathbf{P}_{0|0} &= \mathrm{diag}(10^2, 10, \ldots, 10, \mathbf{\Sigma}^{(p)}, 10^{-8}, 10^3, 10^{-8}) \ .\end{aligned}$$

Thus, the initial airplane shape is initialized as circle of radius $d_{\max}$.

Results

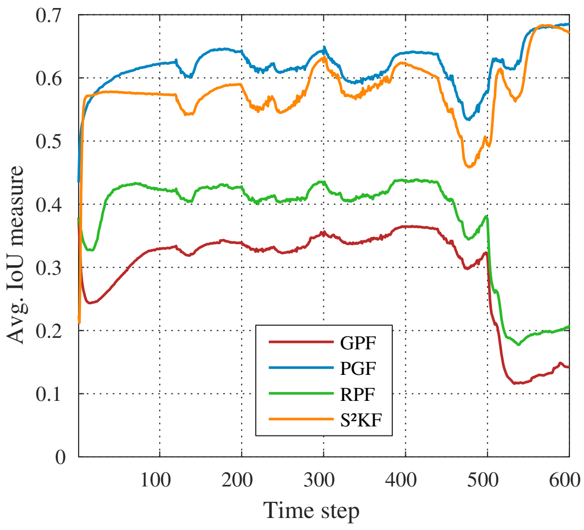
We assess the estimation performance of each filter by means of the “intersection over union” (IoU) measure [164]. The advantage of IoU measure is that it combines pose and shape estimate of a filter to a single key figure. Let $\mathcal{G}_k \subset \mathbb{R}^2$ denote the area of the airplane at time step k, i.e., the ground truth, and $\mathcal{E}_k^{(r)} \subset \mathbb{R}^2$ the area of a star-convex object estimated by one of the investigated filters at time step k from the r^{th} simulation run. The IoU measure $m_k^{(r)}$ for time step k and r^{th} simulation run is then defined as

$$m_k^{(r)} := \frac{\displaystyle\int_{\mathcal{G}_k \cap \mathcal{E}_k^{(r)}} \mathrm{d}x\,\mathrm{d}y}{\displaystyle\int_{\mathcal{G}_k \cup \mathcal{E}_k^{(r)}} \mathrm{d}x\,\mathrm{d}y} \ , \quad 0 \le m_k^{(r)} \le 1 \ .$$

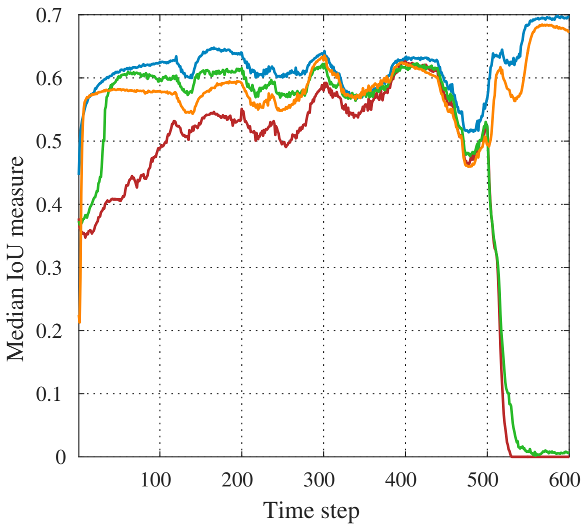
A value of $m_k^{(r)} = 0$ means no overlap of airplane and estimated star-convex object, whereas a value of $m_k^{(r)} = 1$ means both overlap perfectly. Thus, the larger the IoU measure the better an estimate approximates pose and shape of the airplane. However, keep in mind that due to the airplane’s non-star-convex shape a value of 1 cannot be achieved by any of the investigated estimators.

The average IoU measures over all simulation runs are shown in Figure 5.15(a). It can be seen that the particle filters perform much worse than the PGF and the S^{2}KF although they use hundreds of thousands of particles. In fact, the GPF is the worst estimator, whereas and the PGF is the best one. Nevertheless, the S^{2}KF performs very similar to the PGF in some situations. Furthermore, various noticeable drops in the IoU measures can be observed for all estimators. Those occur after significant changes in the airplane’s trajectory such as (fast) turns, e.g., for $k = 300$ or $k = 440$. Due to the divergence of particle filters in several simulation runs, we additional consider the median of the IoU measures over all simulation runs in order to exclude those runs from the results, see Figure 5.15(b). On the one hand, we notice that the values of the PGF and the S^{2}KF are nearly identical to their average IoU measures. Hence, both do not seem to have severe outliers. On the other hand, the particle filters now have much better IoU measures at the beginning, which come even very close to those of the PGF and the S^{2}KF. However, after the airplane’s fast turn at time step 500, the RPF and the GPF completely lose the track indicated by an instantly drop in their IoU measures to nearly 0. All these results are also illustrated by visualizing the respective filter estimates from a simulation run in Figure 5.16.

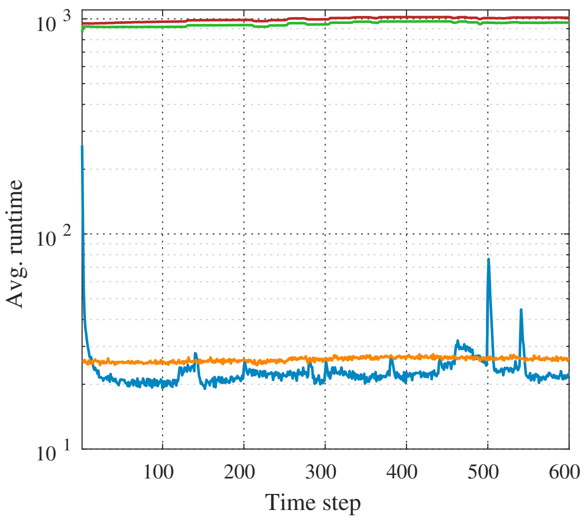
Next, we consider the runtimes of all estimators, see Figure 5.15(c). By no surprise, the particle filters with their huge amount of particles are 40 times slower than the PGF and the S^{2}KF, while at the same time delivering much worse estimation results. The PGF is most times a little bit faster than the S^{2}KF. However, the PGF’s execution time also has some peeks, e.g., for $k = 500$. This is due to the adaptive number of recursion steps performed by the PGF: in case of abruptly changes in the airplane’s trajectory, the measurement update is more complex, which is inherently reflected by the number of recursion steps. In fact, the PGF’s adaptive measurement updates are the reason for its superior estimation quality. In summary, the PGF yields the best compromise of estimation quality and execution time. Nevertheless, the S^{2}KF comes very close. The investigated particle filters are by far the worst estimators regarding both estimation performance and runtime.



(a) Average IoU measure.



(b) Median IoU measure.



(c) Average runtimes in ms.

Figure 5.15: Results of the airplane tracking evaluation.

5.5.4 Tracking a Sphere in 3D

In the last evaluation, we analyze how a proper GPU implementation can accelerate the PGF. More precisely, our goal is to perform extended object tracking in real time when processing a vast amount of measurements. For this purpose, we estimate pose, shape, and motion parameters of a sphere in 3D, namely

- center $\boldsymbol{p}_k = [p_k^{(\mathrm{x})}, p_k^{(\mathrm{y})}, p_k^{(\mathrm{z})}]^\top$ in m,
- radius r_k in m, and
- velocity $\dot{\boldsymbol{p}}_k = [\dot{p}_k^{(\mathrm{x})}, \dot{p}_k^{(\mathrm{y})}, \dot{p}_k^{(\mathrm{z})}]^\top$ in $\mathrm{m/s}$,

leading to the system state $\boldsymbol{x}_k = [\boldsymbol{p}_k^\top, r_k, \dot{\boldsymbol{p}}_k^\top]^\top$.

The considered tracking scenario is as follows. We simulate the sphere's movement along a nonlinear path and changes in its radius while it is observed by a network of five simulated Microsoft Kinect cameras, see Figure 5.17. Each camera produces per frame a point cloud of noisy 3D point measure-

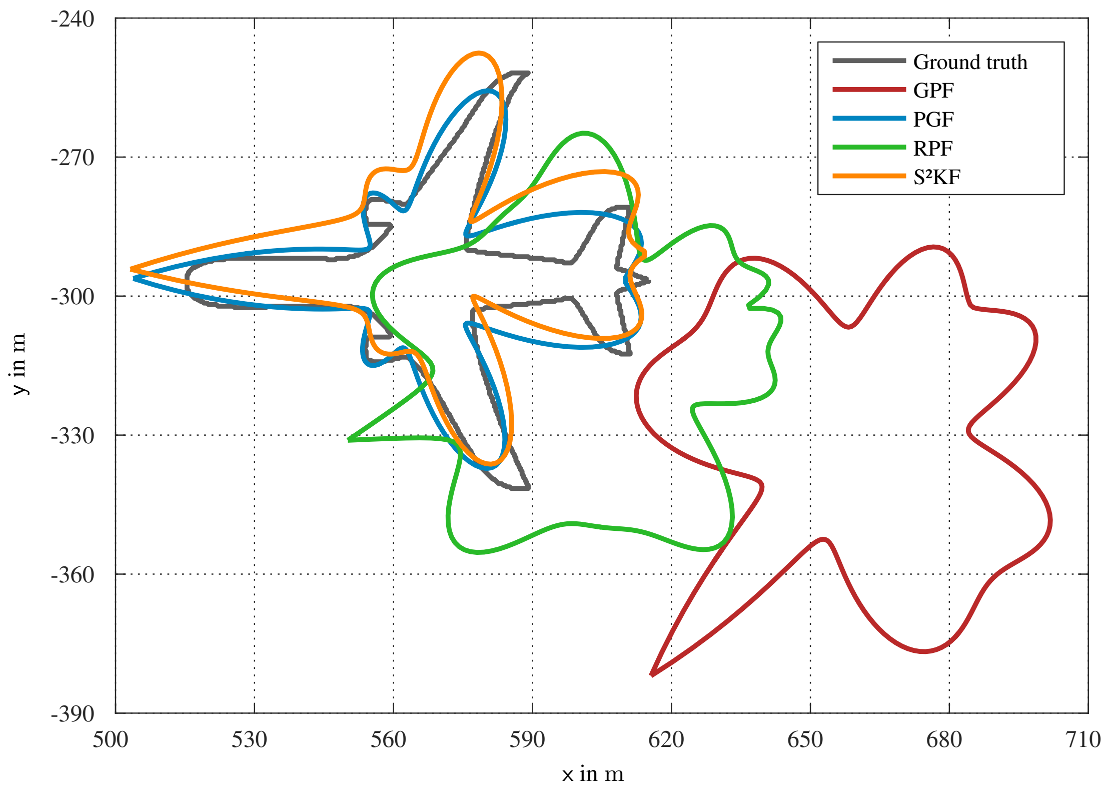


Figure 5.16: Estimated airplanes of a simulation run at time step $k = 480$. The 1-axial symmetric shape estimates can be clearly seen.

ments originating from the sphere's surface. The sphere, with initial radius of $15\,\mathrm{cm}$, moves in total for $5\,\mathrm{s}$. After $2\,\mathrm{s}$, the radius of the sphere increases from $15\,\mathrm{cm}$ to $40\,\mathrm{cm}$ over the next $0.75\,\mathrm{s}$. The sphere radius shrinks abruptly back to $15\,\mathrm{cm}$ after another $2\,\mathrm{s}$. Due to the sphere's movement and its shape changes, the number of available measurements varies (greatly) over time, and allows to best assess how well the CPU and GPU implementations of the PGF handle such different workloads.

This setup is a challenging estimation task, in particular computationally, as (i) each Kinect camera has a high resolution (640×480 pixels) resulting in large point clouds and (ii) the sphere can be seen from several cameras at the same time. Both lead to an enormous number of measurements that have to be processed in a short time (the Kinect cameras work at a rate of ≈ 33 frames per second). Next, we derive an appropriate likelihood function to effectively estimate the system state based on those noisy point measurements.

Likelihood Function

At each time step k, we receive a set of Y_k Cartesian measurements $\mathcal{Y}_k = \{\tilde{\boldsymbol{y}}_k^{(j)}\}_{j=1}^{Y_k}$ originating from the sphere's surface, where each measurement is obtained by one of the Kinect cameras. Like in Section 3.4.2, it is assumed that $\tilde{\boldsymbol{y}}_k^{(j)}$ is generated according to the model

$$\boldsymbol{y}_k^{(j)} = \boldsymbol{z}_k^{(j)} + \boldsymbol{v}_k^{(j)} \ ,$$

where $\boldsymbol{z}_k^{(j)}$ is an unknown measurement source on the sphere's surface and $\boldsymbol{v}_k^{(j)}$ is zero-mean white Gaussian noise with covariance matrix $\mathbf{R}_k^{(j)}$. The latter are obtained according to the Kinect sensor noise model proposed in [165], which gives realistic non-isotropic measurement covariance matrices. Moreover, we assume that the noise from different measurements are mutually independent.

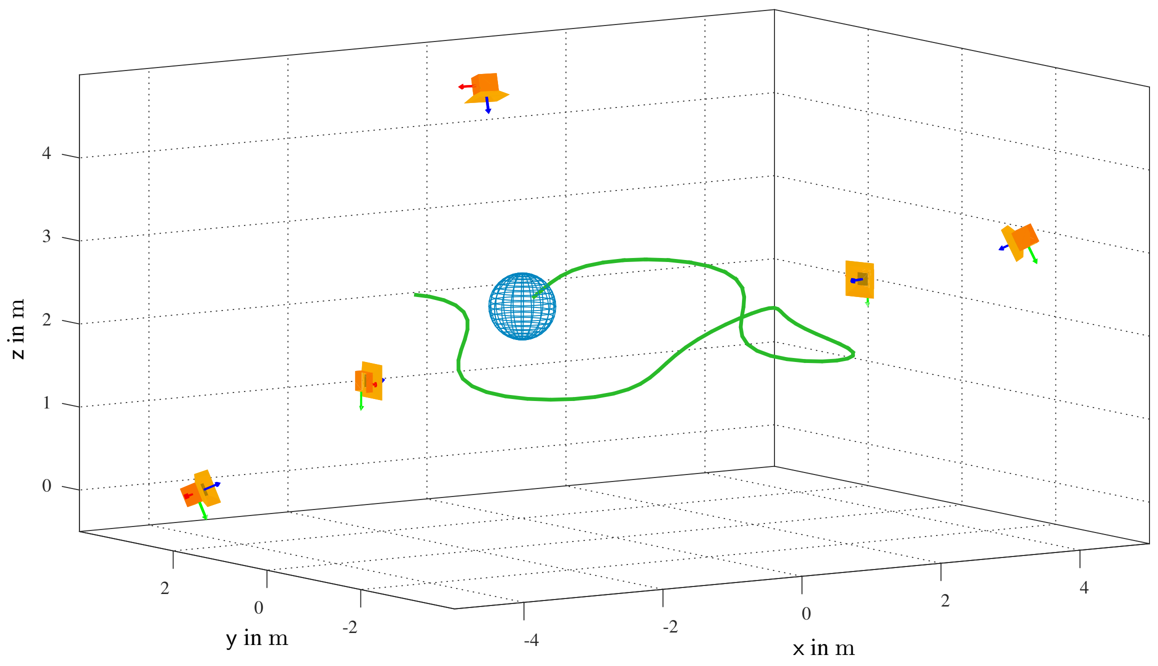


Figure 5.17: Considered tracking scenario: five simulated Kinect cameras (orange) observe a sphere (blue) that moves along a nonlinear path (parts of it shown as green curve) (cf. [184]).

Now, deriving an appropriate likelihood function boils down to determining a good measurement source approximation for each measurement. We again rely on a greedy association to approximate the source $\boldsymbol{z}_k^{(j)}$ according to

$$\boldsymbol{z}_k^{(j)} \approx \check{\boldsymbol{z}}_k^{(j)}(\boldsymbol{x}_k, \boldsymbol{c}, \tilde{\boldsymbol{y}}_k^{(j)}) \ ,$$

where $\boldsymbol{c}$ denotes the known position of the Kinect camera from which the measurement originates. We use the camera position to best exploit the geometrical interaction between camera, sphere, and measurement. That is, we roughly assume that a Kinect camera can only observe the half of the sphere that is between the camera and the so-called "visibility plane". This plane is orthogonal to the direction from camera position $\boldsymbol{c}$ to sphere center $\boldsymbol{p}_k$ (see Figure 5.18). Thus, only points from this side of the sphere are treated as possible measurement sources for a received measurement. Out of these points, we select the most reasonable source $\check{\boldsymbol{z}}_k^{(j)}$ for the measurement $\tilde{\boldsymbol{y}}_k^{(j)}$ based on the projection line from camera position to measurement and its possible intersection with the sphere given by the state $\boldsymbol{x}_k$. We have to distinguish between four possible cases:

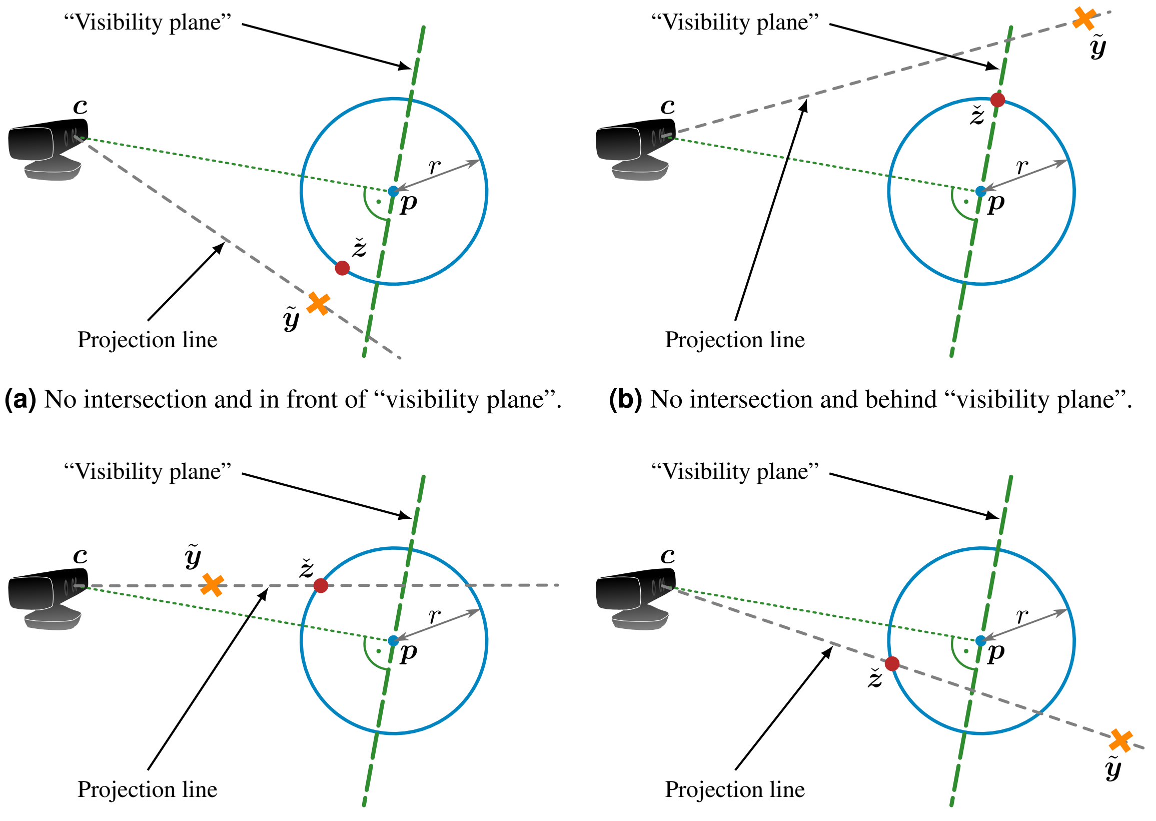
(a) the projection line has no intersection with the sphere and the measurement is in front of the "visibility plane" (see Figure 5.18(a)),

(b) the projection line has no intersection with the sphere and the measurement is behind the "visibility plane" (see Figure 5.18(b)),

(c) the projection line intersects the sphere and the measurement is in front of it (see Figure 5.18(c)),

(d) the projection line intersects the sphere and the measurement is behind it (see Figure 5.18(d)).

Hence, we do not always simply choose the closest point on the sphere's surface as the measurement source.



(a) No intersection and in front of “visibility plane”. **(b)** No intersection and behind “visibility plane”.

(c) Intersection and in front of sphere. **(d)** Intersection and behind sphere.

Figure 5.18: Visualization of the four possible measurement source cases (cf. [184]). For better readability, the time index k is omitted.

The above considerations finally yield the likelihood function

$$f(\mathcal{Y}_k \,|\, \boldsymbol{x}_k) = \prod_{j=1}^{Y_k} \mathcal{N}(\tilde{\boldsymbol{y}}_k^{(j)} - \check{\boldsymbol{z}}_k^{(j)}\,; \boldsymbol{0}, \mathbf{R}_k^{(j)}) \ ,$$

where $\check{\boldsymbol{z}}_k^{(j)}$ is computed with the aid of Algorithm 5.2. Its logarithmic version is

$$\begin{aligned} \log(f(\mathcal{Y}_k \,|\, \boldsymbol{x}_k)) = \sum_{j=1}^{Y_k} -\frac{3}{2}\log(2\pi) - \frac{1}{2}\log\left(|\mathbf{R}_k^{(j)}|\right) \\ - \frac{1}{2}\left(\tilde{\boldsymbol{y}}_k^{(j)} - \check{\boldsymbol{z}}_k^{(j)}\right)^\top \left(\mathbf{R}_k^{(j)}\right)^{-1}\left(\tilde{\boldsymbol{y}}_k^{(j)} - \check{\boldsymbol{z}}_k^{(j)}\right) \ . \end{aligned} \tag{5.33}$$

From this, it can be seen that the proposed measurement source computation penalizes estimates $\boldsymbol{x}_k$ where the sphere is placed between camera and measurements, as the resulting Mahalanobis distances between measurements and associated sources become larger.

Due to the detailed modeling of the physical background, the likelihood should yield good estimation results. However, the likelihood is also rather complicated and, when additionally coping with thousands of measurements $\mathcal{Y}_k$, its evaluation requires a substantial amount of computational resources, making it perfectly suitable for an implementation on a GPU.

```
Input:    state $\boldsymbol{x}$, camera position $\boldsymbol{c}$, and measurement $\tilde{\boldsymbol{y}}$
Output:   measurement source $\check{\boldsymbol{z}}$

        // Radius is always positive
 1:     $r \leftarrow |r|$
        // Unit vector pointing from camera position to measurement
 2:     $\boldsymbol{d} = \mathrm{normalize}(\tilde{\boldsymbol{y}} - \boldsymbol{c})$
        // Vector pointing from sphere center to measurement
 3:     $\boldsymbol{e} = \tilde{\boldsymbol{y}} - \boldsymbol{p}$
        // Check for line-sphere-intersection
 4:     $a = \boldsymbol{d}^\top \boldsymbol{e}$
 5:     $b = a^2 - (\boldsymbol{e}^\top \boldsymbol{e} - r^2)$
 6:     if $b < 0$ then   // No line-sphere-intersection
            // "Visibility plane" normal pointing from camera to sphere
 7:         $\boldsymbol{n} = \boldsymbol{p} - \boldsymbol{c}$
 8:         if $\boldsymbol{e}^\top \boldsymbol{n} < 0$ then   // In front of "visibility plane"
                // Project measurement on sphere (case in Figure 5.18(a))
 9:             $\check{\boldsymbol{z}} \leftarrow \boldsymbol{p} + r \cdot \mathrm{normalize}(\boldsymbol{e})$
10:         else   // Behind "visibility plane"
                // Intersection of projection line and "visibility plane"
11:             $\boldsymbol{l} = \tilde{\boldsymbol{y}} - (\boldsymbol{e}^\top \boldsymbol{n} / \boldsymbol{d}^\top \boldsymbol{n}) \boldsymbol{d}$
                // Vector pointing from sphere center to intersection
12:             $\boldsymbol{q} = \boldsymbol{l} - \boldsymbol{p}$
                // Project intersection on sphere (case in Figure 5.18(b))
13:             $\check{\boldsymbol{z}} \leftarrow \boldsymbol{p} + r \cdot \mathrm{normalize}(\boldsymbol{q})$
14:         end if
15:     else   // Line-sphere-intersection
            // Distances between measurement and sphere intersections
16:         $d_1 = -a + \sqrt{b}$
17:         $d_2 = -a - \sqrt{b}$
            // Choose intersection closest to the camera position
18:         if $d_1 < d_2$ then
19:             $d_{\mathrm{min}} \leftarrow d_1$
20:         else
21:             $d_{\mathrm{min}} \leftarrow d_2$
22:         end if
            // If $d_{\mathrm{min}} \geq 0$ case in Figure 5.18(c), otherwise case in Figure 5.18(d)
23:         $\check{\boldsymbol{z}} \leftarrow \tilde{\boldsymbol{y}} + d_{\mathrm{min}} \boldsymbol{d}$
24:     end if
```

Algorithm 5.2: Proposed computation of sphere measurement sources. For better readability, the time index k is omitted.

Likelihood Implementation on a GPU

In order to execute the PGF on a GPU as described in Section 5.4, we have to provide a proper parallelized implementation of the log-likelihood (5.33) evaluations, i.e., we consider step 5 in Section 5.4.3. But first of all, before the actual PGF recursion starts, we have to copy the camera positions $\boldsymbol{c}$ associated to each measurement and all noise covariance matrices $\mathbf{R}_k^{(j)}$ from CPU memory to GPU memory among the other relevant data during step 1 in Section 5.4.3. Then, we compute the inverse covariance matrices required for the log-likelihood completely in parallel by starting Y_k threads on the GPU. Each thread first loads the data of its associated covariance matrix from GPU memory and then computes the inverse by using the analytic formula for inverting 3×3 matrices. Finally, each thread writes its computed inverse matrix back to GPU memory to make it accessible for the log-likelihood evaluation coming later.

Like for particle filters, we could evaluate the log-likelihood independently for each state sample $\boldsymbol{x}_k^{(i)}$. However, compared to particle filters, we only have a small number of M samples per recursion step. Launching solely M GPU threads, where each thread computes the large sum of Y_k terms in (5.33), would by no means harness all the available GPU's resources. Thus, as already mentioned in step 5 in Section 5.4.3, we schedule $Y_k \times M$ GPU threads to compute each combination of state sample and measurement in parallel. That is, each thread loads its relevant state sample $\boldsymbol{x}_k^{(i)}$, measurement $\tilde{\boldsymbol{y}}_k^{(j)}$, camera position $\boldsymbol{c}$, and inverse measurement noise covariance matrix $(\mathbf{R}_k^{(j)})^{-1}$ from GPU memory. Then, each thread executes Algorithm 5.2 on its own data and computes

$$v_k^{(j,i)} = -\frac{1}{2}\big(\tilde{\boldsymbol{y}}_k^{(j)} - \check{\boldsymbol{z}}_k^{(j)}(\boldsymbol{x}_k^{(i)}, \boldsymbol{c}, \tilde{\boldsymbol{y}}_k^{(j)})\big)^\top \big(\mathbf{R}_k^{(j)}\big)^{-1}\big(\tilde{\boldsymbol{y}}_k^{(j)} - \check{\boldsymbol{z}}_k^{(j)}(\boldsymbol{x}_k^{(i)}, \boldsymbol{c}, \tilde{\boldsymbol{y}}_k^{(j)})\big) \ .$$

Note that the state-independent terms in (5.33) can be omitted as we require the log-likelihood only up to proportionality. Finally, each thread writes its value $v_k^{(j,i)}$ back to GPU memory resulting in the large matrix $\mathbf{V}_k \in \mathbb{R}^{Y_k \times M}$. After that, we schedule M GPU threads to get the final log-likelihood values by performing the tree-like sum reduction (described in Section 5.4.3) in parallel for each column of $\mathbf{V}_k$.

Simulation

We configure the PGF to use $M = 51$ samples. The CPU implementation is written in C++ using the Eigen linear algebra library [166] to benefit from SSE accelerated SIMD computations. In addition, the log-likelihood evaluations are parallelized with OpenMP. Further, the GPU implementation is based on OpenCL 1.2. The evaluation was done on a system with Intel Core i7-3770 CPU (3.4 GHz, 4 cores, 8 threads) and AMD Radeon R9 280X GPU that offers great double-precision performance.

We perform 50 Monte Carlo runs. In each run, an initial state estimate $\hat{\boldsymbol{x}}_{0|0}$ and $\mathbf{P}_{0|0}$ is obtained using the first set of available measurements $\mathcal{Y}_0$. That is, we compute mean and covariance of the measurements to get an initial position estimate

$$\hat{\boldsymbol{p}} = \frac{1}{Y_0}\sum_{j=1}^{Y_0} \tilde{\boldsymbol{y}}_0^{(j)} \ ,$$
$$\boldsymbol{\Sigma}^{(p)} = \frac{1}{Y_0}\sum_{j=1}^{Y_0} \big(\tilde{\boldsymbol{y}}_0^{(j)} - \hat{\boldsymbol{p}}\big)\big(\tilde{\boldsymbol{y}}_0^{(j)} - \hat{\boldsymbol{p}}\big)^\top \ ,$$

and mean and variance of the respective radii to get an initial radius estimate

$$\hat{r} = \frac{1}{Y_0}\sum_{j=1}^{Y_0} \|\tilde{\boldsymbol{y}}_0^{(j)} - \hat{\boldsymbol{p}}\|_2 \ ,$$
$$\Sigma^{(r)} = \frac{1}{Y_0}\sum_{j=1}^{Y_0} \big(\|\tilde{\boldsymbol{y}}_0^{(j)} - \hat{\boldsymbol{p}}\|_2 - \hat{r}\big)^2 \ .$$

Together with an initial velocity of zero with unit covariance matrix, this leads to the initial state estimate

$$\hat{\boldsymbol{x}}_{0|0} = [\hat{\boldsymbol{p}}^\top, \hat{r}, 0, 0, 0]^\top \ ,$$
$$\mathbf{P}_{0|0} = \operatorname{diag}\big(\boldsymbol{\Sigma}^{(p)}, \Sigma^{(r)}, \mathbf{I}_3\big) \ .$$

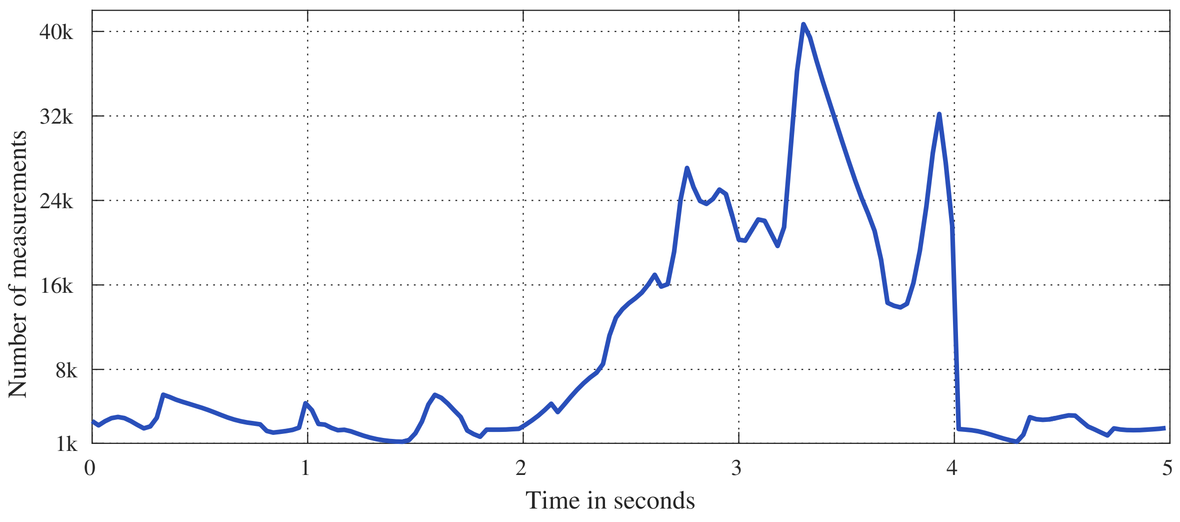
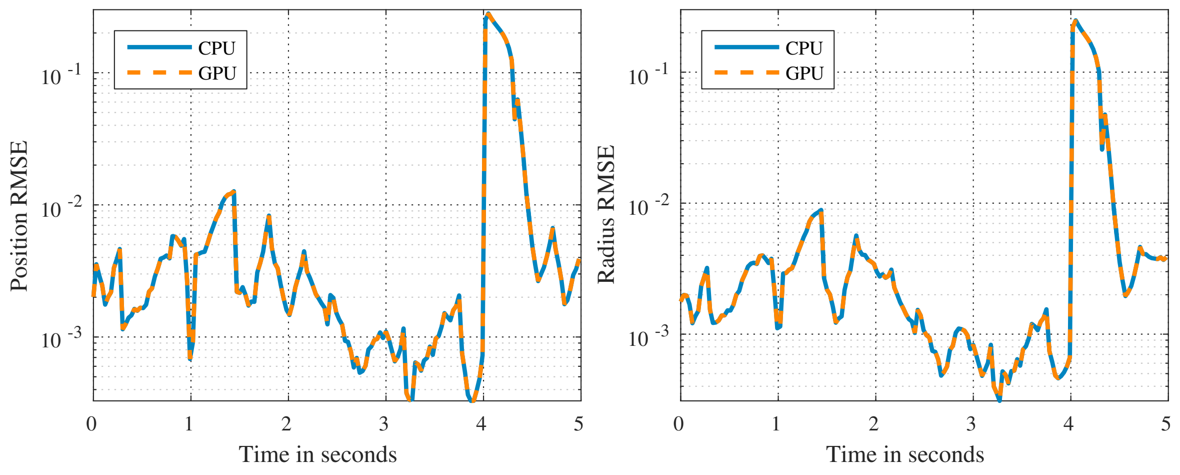


Figure 5.19: Number of measurements to be processed by the PGF implementations (cf. [184]).



(a) Position RMSE in m. **(b)** Radius RMSE in m.

Figure 5.20: Sphere estimation errors (cf. [184]).

Moreover, we describe the sphere's temporal behavior with a constant velocity model combined with a random walk for the radius given by

$$\boldsymbol{x}_k = \mathbf{A}\boldsymbol{x}_{k-1} + \mathbf{B}\boldsymbol{w} \ ,$$

with matrices

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} & \Delta t\,\mathbf{I}_3 \\ \mathbf{0} & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_3 \end{bmatrix} \ , \quad \mathbf{B} = \begin{bmatrix} \Delta t\,\mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & 1 \\ \mathbf{I}_3 & \mathbf{0} \end{bmatrix} \ ,$$

time period $\Delta t = 0.03\,\mathrm{s}$, and time-invariant zero-mean white Gaussian noise $\boldsymbol{w}$ with covariance matrix $\mathbf{Q} = \mathrm{diag}(\mathbf{I}_3, 10^{-5})$. Due to the linear system model, we can perform the prediction in closed form as usual. Also for the GPU implementation, the prediction is computed on the CPU.

Results

First of all, we take a look on the number of measurements that have to be processed over time (see Figure 5.19). It can be seen that the amount of measurements greatly varies over time and ranges from

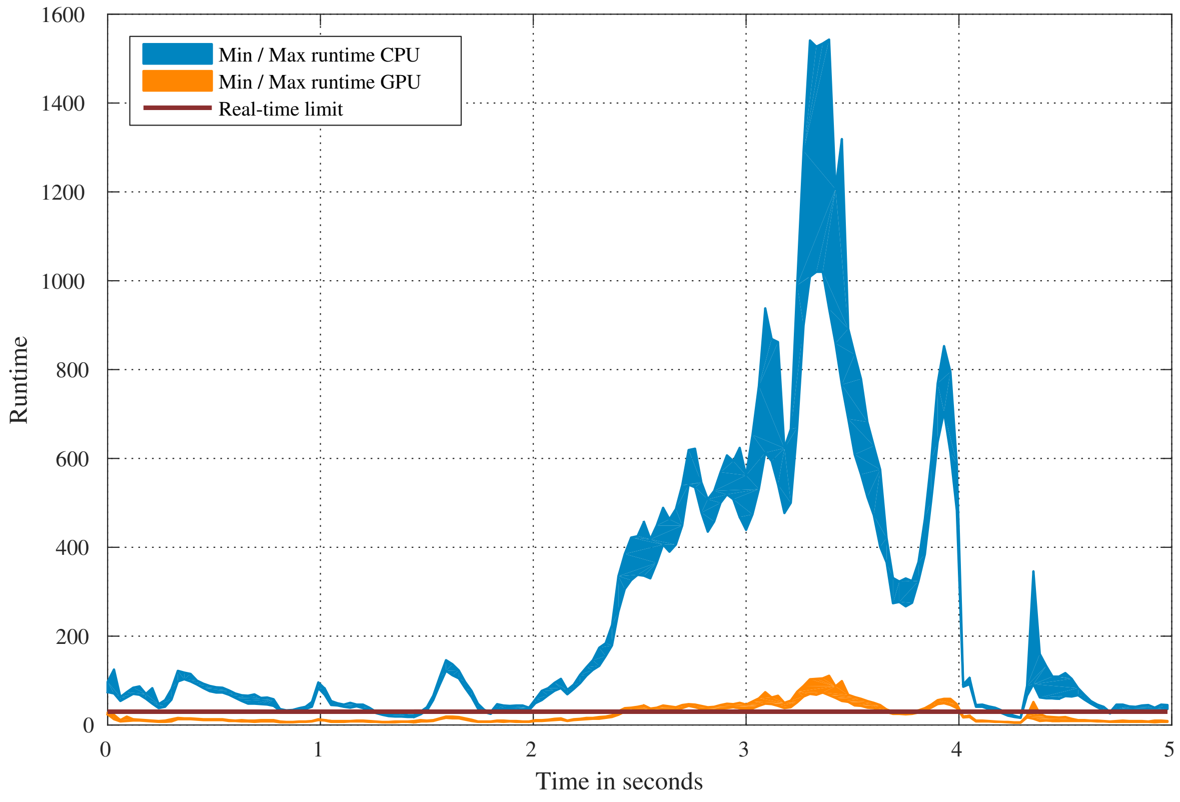


Figure 5.21: Runtimes in ms of the two PGF implementations (cf. [184]).

1 000 to over 40 000 measurements per time step. A very noticeable increase starts at 2 s, when the sphere's radius growths. After further 2 s, the amount abruptly drops when the sphere shrinks back to its initial radius. Nonetheless, the number of measurements changes even when the size of the sphere is constant, e.g., between 0 s and 2 s. At this point, we want to point out that processing 40 000 measurements at once with a Kalman filter is intractable as the resulting measurement covariance matrix would require several gigabytes of data, not to mention the time to process it.

Next, we consider the estimation errors of both implementations. In Figure 5.20, the RMSE of the sphere's center and radius are plotted. It can be seen that the errors in general are very small. This is due to the large number of measurements, which allow for accurate estimates. The only significant error jump happens when the sphere abruptly changes its radius at 4 s. But more importantly, we see that both implementations possess nearly identical estimation errors (despite roundoff errors coming from a different order of arithmetic operations). This was expected and confirms that both implementations work properly. Concluding, we can say that it should be no problem to switch from a CPU implementation of the PGF to a GPU-accelerated variant in other applications as well.

After the correctness of the GPU-accelerated PGF is ensured, we turn to the actual interesting comparison: the filter execution time. For both variants, we compare the time required to conduct a full time step. Especially for the GPU implementation, this encompasses

1. the state prediction on the CPU,
2. the time needed to copy the predicted state estimate, all measurements and their associated camera positions, and all noise covariance matrices from CPU memory to GPU memory,
3. the measurement update itself, and
4. the time needed to copy the filtered state estimate back to the CPU.

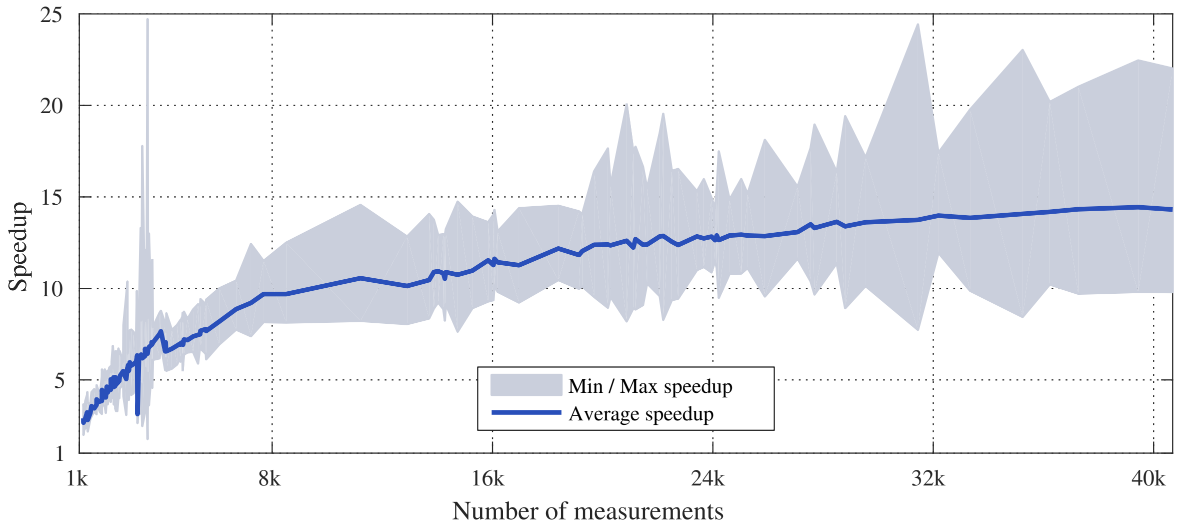


Figure 5.22: Attained speedup of the GPU-accelerated PGF (cf. [184]).

In Figure 5.21, the respective minimum execution times (lower bound of a curve) and maximum execution times upper bound of a curve) are shown. We notice that the GPU implementation has an overall much faster execution than the CPU implementation. In fact, the GPU variant is always faster than the CPU variant. On the one hand, the GPU-accelerated PGF requires 110 ms at most for a time step, whereas the CPU-based PGF requires over 1.5 s in the worst case. On the other hand, although we have to copy data between CPU memory and GPU memory, the minimum runtime of the GPU is 5 ms, while the CPU does not fall below 16 ms. In addition, the runtime has much less fluctuations on the GPU, in particular when it comes to a large amount of measurements. Note that in order to work in real time in this scenario, we must not exceed the limit of 30 ms per time step. Unfortunately, the CPU is often far away from this, i.e., only in 8 % of the time it is below the real-time limit. In contrast, the GPU satisfies this in 68 % of the time.

All these can be explained by comparing Figure 5.19 and Figure 5.21. We notice that the CPU runtime is highly correlated with the number of processed measurements, which means that already 1 000 measurements keep all CPU cores busy, leaving no room for the processing additional measurements. In contrast, the GPU with its many computational resources is rather unaffected by the changes in the number of measurements.

Finally, we build the speedup of the GPU implementation over the number of processed measurements, see Figure 5.22. As mentioned above, the GPU is always faster than the CPU. Consequently, we always have speedup greater than one. Furthermore, the speedup exhibits a logarithmic characteristic that reaches already an average of 10 when processing 10 000 measurements, and even speedups over 20 are possible. Hence, compared to the speedup of particle filters reported in literature, the PGF can achieve similar speedups, if not better.

5.6 Conclusions

In the last part of this thesis, we dealt with nonlinear state estimators. Other than Kalman filters applied to nonlinear systems, these estimators do not linearize the relationship between state and measurement in order to perform a measurement update. Instead, they rely on the evaluation of likelihood functions. The main issue of most state-of-the-art likelihood-based estimators is, however, the problem of sample degeneracy. Particle filters such as the SIRPF or RPF tackle sample degeneracy with a combination of importance sampling and adaptive resampling. Nevertheless, particle filters are computationally expensive due to their huge amount of particles required especially for larger state spaces.

Hence, we took up the progressive approach of the PGF 42 in order to circumvent sample degeneracy and proposed several improvements leading to a new progressive Gaussian filter called PGF for short. First, we reformulated the PGF 42 to be able to directly work with given likelihood functions. Second, we dropped the PGF 42's forward/backward updates as evaluations revealed that those are not necessary. Third, we replaced the originally used asymmetric LCD-based Gaussian sampling with its point-symmetric version from Chapter 2. Fourth, we proposed an effective heuristic for an automatic parametrization of the PGF. Fifth, we derived a semi-analytic measurement update to further improve estimation quality and runtime. Finally, we also proposed a GPU-accelerated implementation of the PGF to deal with tens of thousands of measurements at the same time.

Several target tracking evaluations showed that the PGF can outperform state-of-the-art nonlinear and linear estimators including the PGF 42, GPF, RPF, and S^{2}KF. Also the PGF's semi-analytic measurement update yielded noticeable improvements. Furthermore, the evaluations included derivations of a closed-form likelihood function for star-convex RHMs and a likelihood function for efficiently tracking pose and extent of a sphere. Although the PGF solely maintains a unimodal state estimate, the evaluations revealed that the PGF works quite well. Nevertheless, if multimodal estimates are indispensable, the PGF can be extended to a Gaussian mixture estimator in a straightforward manner as it is done for the GPF [59].

The PGF was already successfully applied to different topics in target tracking such as extended object tracking based on extrusion random hypersurface models [140], pose and shape estimation of 3D objects using transformed plane curves [174], unbiased and bias reduced extended object tracking [90, 167], tracking elongated extended objects using splines [168], or when exploiting negative information to enhance extended object tracking [169], [180]. Moreover, the authors of [170] already take up the PGF approach and develop a progressive Kalman filter.

Chapter 6

Conclusions

Estimating the hidden state of a discrete-time stochastic nonlinear dynamic system is an ubiquitous problem in many engineering tasks such as robotics, optimal control, or target tracking. Additionally, for the last few years off-the-shelf sensors, such as Microsoft's Kinect or laser scanners, have been able to provide a huge amount of noisy measurements per scan, increasing the demand for new estimation techniques able to efficiently handle the available data. Motivated by this, the present thesis dealt with a variety of different topics in nonlinear state estimation and their applications to tracking. In particular, emphasis was laid on both the development and improvement of recursive Bayesian estimators that maintain a Gaussian state estimate and extended object tracking for the simultaneous estimation of an object's pose, shape, and motion parameters. While an enhanced optimal Gaussian sampling technique built the foundation for the Bayesian estimators, the considered object tracking included the derivation of elaborate measurement models and corresponding likelihood functions, which are required for adequate tracking performance.

6.1 Summary

We started with optimal sampling of multivariate Gaussian distributions, i.e., the approximation of Gaussian densities with a carefully chosen set of weighted point masses called Dirac mixtures. We discussed state-of-the-art sampling techniques that were originally developed for Kalman filtering, i.e., sampling schemes used by the UKF, the GHKF, the 5$^{\text{th}}$-degree CKF, or the RUKF, but also an approach that is based on the localized cumulative distribution. This LCD-based sampling technique turns the approximation problem into an optimization problem by defining a proper distance measure between the Gaussian PDF and the Dirac mixture to be optimized. However, all these techniques have their individual drawbacks, ranging from limitations in the number of employed samples over negative sample weights that can result in numerical issues to non-deterministic or asymmetric samples. In order to overcome all these issues and get a more advanced Gaussian sampling technique, we proposed an improved LCD-based sampling scheme that allows the approximation of multivariate standard normal distributions with an arbitrary number of equally weighted and optimally placed point-symmetric samples. Compared to the original asymmetric LCD-based sampling, the proposed one captures all odd moments of a standard normal distribution exactly. Moreover, we improved the numerical stability of the LCD approach to be able to approximate large random vectors, which frequently arise in extended object tracking when processing many measurements per time step. Evaluations showed that the proposed point-symmetric LCD-based Gaussian sampling scheme can outperform the state-of-the-art approaches when computing moments of nonlinear transformed Gaussian distributed random vectors.

A first application of the proposed optimal Gaussian sampling scheme were Kalman filters, i.e., linear estimators applied to nonlinear systems. Those estimators have to compute several moments of transformed Gaussian random vectors. While approaches like the EKF directly approximate the nonlinear models to solve the moment integrals, sample-based Kalman filters simply approximate the involved multivariate Gaussian distributions with appropriate sampling techniques. Hence, we introduced a new sample-based Kalman filter, the smart sampling Kalman filter, that is based on the proposed point-symmetric LCD-based Gaussian sampling. The advantage of the S^2KF over state-of-the-art sample-based Kalman filters are its arbitrary number of optimally placed and equally weighted samples. This allows for a fine-grained control over estimation quality and execution time of the filter. Additionally, the equal sample weights reduce the number of arithmetic operations required for a moment computation and also avoid numerical issues when computing covariance matrices. Furthermore, the S^2KF can directly be used with any typical Kalman filter extension like iterative measurement updates or Gaussian mixture models. As an elaborate evaluation, we performed extended object tracking, i.e., estimating pose and unknown shape of a cylinder in 3D based on hundreds of noisy measurements originating from its surface. For that purpose, we first had to derive a new measurement model that combines the random hypersurface model approach to tackle the major problem of unknown measurement sources and the signed Euclidean distance to keep the measurement space small. However, the derived model requires to sample a uniform distribution, which is not simply possible for sample-based Kalman filters. We solved this issue by using the fact that a random variable transformed with its own cumulative distribution is uniformly distributed. Simulations showed that the S^2KF can beat state-of-the-art Kalman filters. Moreover, it is much more stable regarding positive definiteness of covariance matrices due to its equal sample weights.

Our next goal was to deploy the S^2KF for challenging distributed nonlinear state estimation scenarios. However, established distributed state estimation techniques have certain problems. For example, information filtering requires full-rate communication and additional approximations when using sample-based Kalman filters, while federated Kalman filtering and covariance intersection simply do not have concrete correlation information. The DKF requires that sensor nodes have comprehensive information about the measurement processing from all other sensor nodes. In particular, the latter is not applicable to nonlinear models as the required linearizations cannot be known in advance, which would lead to a massive communication overhead. Thus, we developed a novel sample-based fusion approach, where correlations between local estimates can be exactly reconstructed at the fusion center. Basically, the approach relies on a local processing of samples that encode the correlations and are sent to the fusion center in addition to the actual locally obtained state estimate. Additional advantages of the proposed approach are that sensor nodes do not need any information about the measurement processing from other nodes, it is well-suited for large sensor networks as the number of correlation samples is independent of the number of utilized nodes, and nodes can be added or removed over time without affecting the other nodes. Target tracking evaluations showed that the proposed fusion technique can outperform the popular covariance intersection approach for both linear and nonlinear models. In particular, we reconsidered the previously performed pose and shape estimation of a cylinder, but now in a distributed setup. Simulations revealed that covariance intersection was not even capable of estimating the cylinder's pose and shape, while our sample-based fusion approach performed very well.

A second application of the proposed optimal Gaussian sampling technique were nonlinear state estimators. In contrast to Kalman filters applied to nonlinear systems, these estimators do not linearize the relationship between state and measurement in order to conduct a filter step. However, most state-of-the-art nonlinear estimators like particle filters suffer from the severe problem of sample degeneracy. Furthermore, particle filters are computationally burdensome due to their enormous amount of particles required especially for larger state spaces. Thus, we took up the promising progressive approach of the PGF 42 in order to circumvent sample degeneracy. We proposed several improvements that resulted in a new progressive Gaussian filter called simply PGF. Among other improvements, the PGF works directly with given likelihood functions, it uses the proposed point-symmetric LCD-based Gaussian sampling instead of its original asymmetric version, and it relies on an effective heuristic for an automatic parametrization. We also derived a semi-analytic filter step to further improve estimation quality and reduce runtime, and proposed a GPU-accelerated implementation to efficiently deal with tens of thousands of measurements in a single filter step. Although the PGF only has a unimodal Gaussian estimate, several target tracking evaluations revealed that it can beat state-of-the-art nonlinear and linear estimators including the RPF, the GPF, the PGF 42, and the S^2KF. In addition, its semi-analytic filter step yielded noticeable improvements. For extended object tracking, we developed a closed-form likelihood function for star-convex random hypersurface models that was used to track an airplane in 2D, and we developed a novel likelihood function for tracking the pose and extent of a sphere. The sphere tracking also illustrated the superiority of a GPU-accelerated PGF over a multithreaded CPU implementation.

Finally, open-source implementations of both the asymmetric and the point-symmetric LCD-based Gaussian sampling technique, the S^2KF, and the PGF are available in the Nonlinear Estimation Toolbox [176]. The toolbox also contains many state-of-the-art linear and nonlinear state estimators like the first-order and second-order EKF, the UKF, the 5^{th}-degree CKF, the RUKF, and several particle filters. Other features include iterative measurement updates and measurement gating for all Kalman filters, semi-analytic measurement updates for Gaussian estimators, and automatic approximation of first-order and second-order model derivatives.

6.2 Outlook

In this thesis, we proposed various novel and improved nonlinear state estimation algorithms. The gained insights are fundamental, meaning that the proposed methods can be further extended to more complex estimation techniques. Besides other possible applications and extended object tracking scenarios for the S^2KF, the PGF, and the sample-based distributed state estimation, further interesting research topics are listed below.

- The S^2KF can be combined with proposed iterative measurements updates [41, 126],
- A square root version of the S^2KF can be implemented and studied as for other sample-based Kalman filters [42, 44]. An advantage of the S^2KF would be that negative sample weights do not have to be considered unlike other state-of-the-art filters.
- If multimodal estimates are indispensable, both the S^2KF and the PGF can be extended to Gaussian mixture estimators analogous to [22, 59].
- Possible performance benefits of using the PGF in applications where Kalman filters are currently used should be analyzed. This is straightforward as both estimators have a Gaussian state estimate. The only requirement is that a likelihood function for the considered measurement model can be provided.
- The proposed sample-based distributed state estimation can be extended to Gaussian mixture state estimates.
- Another enhancement to the sample-based fusion technique would be the ability to deal with nonlinear system models.
- Regarding extended object tracking, it would be interesting to incorporate constraints to the derived closed-form star-convex RHM likelihood as it is already done for Kalman filters [171].
- Finally, the S^2KF or the PGF could be used for multi target tracking.

Appendix A

Proofs of the Point-Symmetric LCD-Based Gaussian Sampling

A.1 Odd Moments of a Point-Symmetric Dirac Mixture

Let $\boldsymbol{s} \in \mathbb{R}^N$ be a random vector and $f(\boldsymbol{s})$ its PDF. The m^{th} moment, with $m = 2k + 1$ and $k \in \mathbb{N}$, is defined as [106, Sec. 6.4]

$$\mathbb{E}_f[\prod_{d=1}^{N} s_d^{n_d}] = \int_{\mathbb{R}^N} \prod_{d=1}^{N} s_d^{n_d} \cdot f(\boldsymbol{s}) \,\mathrm{d}\boldsymbol{s} \ , \tag{A.1}$$

where s_d denotes the entry for the d^{th} dimension of $\boldsymbol{s}$, $n_d \in \mathbb{N}$ with $0 \leq n_d \leq m$, and

$$\sum_{d=1}^{N} n_d = m \ .$$

For the point-symmetric Dirac mixture with an even number of samples (2.8), the m^{th} moment (A.1) is always zero due to

$$\begin{aligned}
\mathbb{E}_\delta[\prod_{d=1}^{N} s_d^{n_d}] &= \int_{\mathbb{R}^N} \prod_{d=1}^{N} s_d^{n_d} \frac{1}{2L} \sum_{i=1}^{L} \delta(\boldsymbol{s} - \boldsymbol{s}^{(i)}) + \delta(\boldsymbol{s} + \boldsymbol{s}^{(i)}) \,\mathrm{d}\boldsymbol{s} \\
&= \frac{1}{2L} \sum_{i=1}^{L} \left(\prod_{d=1}^{N} \big(s_d^{(i)}\big)^{n_d} + \prod_{d=1}^{N} \big(-s_d^{(i)}\big)^{n_d} \right) \\
&= \frac{1}{2L} \sum_{i=1}^{L} \left(\prod_{d=1}^{N} \big(s_d^{(i)}\big)^{n_d} + (-1)^m \prod_{d=1}^{N} \big(s_d^{(i)}\big)^{n_d} \right) \\
&= \frac{1}{2L} \sum_{i=1}^{L} \left(\prod_{d=1}^{N} \big(s_d^{(i)}\big)^{n_d} - \prod_{d=1}^{N} \big(s_d^{(i)}\big)^{n_d} \right) \\
&= 0 \ ,
\end{aligned}$$

where $s_d^{(i)}$ denotes the entry for the d^{th} dimension of the i^{th} sample $\boldsymbol{s}^{(i)}$. Analogously, the same can be shown for the point-symmetric Dirac mixture with an odd number of samples (2.9). ■

A.2 Proof of Distance D^{e}

Like in [111], the modified CvM distance D^{e} can be split into three parts according to

$$
\begin{aligned}
D^{\mathrm{e}}(\mathcal{S}) = &\underbrace{\int_0^\infty w(b) \int_{\mathbb{R}^N} F_{\mathcal{N}}(\boldsymbol{m}, b)^2 \,\mathrm{d}\boldsymbol{m} \,\mathrm{d}b}_{=:D_1^{\mathrm{e}}} - \\
&\underbrace{2 \int_0^\infty w(b) \int_{\mathbb{R}^N} F_{\mathcal{N}}(\boldsymbol{m}, b) F_\delta^{\mathrm{e}}(\mathcal{S}, \boldsymbol{m}, b) \,\mathrm{d}\boldsymbol{m} \,\mathrm{d}b}_{=:D_2^{\mathrm{e}}(\mathcal{S})} + \\
&\underbrace{\int_0^\infty w(b) \int_{\mathbb{R}^N} F_\delta^{\mathrm{e}}(\mathcal{S}, \boldsymbol{m}, b)^2 \,\mathrm{d}\boldsymbol{m} \,\mathrm{d}b}_{=:D_3^{\mathrm{e}}(\mathcal{S})} \ .
\end{aligned}
$$

By exploiting the fact that the product of two (unnormalized) Gaussian PDFs again yields an unnormalized Gaussian PDF and that a PDF always integrates to one we get

$$
\begin{aligned}
D_1^{\mathrm{e}} &= \int_0^\infty w(b) \left(\frac{b^2}{1+b^2} \right)^N (2\pi)^N (1+b^2)^N \int_{\mathbb{R}^N} \mathcal{N}(\boldsymbol{m}\,; \boldsymbol{0}, (1+b^2)\mathbf{I}_N)^2 \,\mathrm{d}\boldsymbol{m} \ \mathrm{d}b \\
&= \int_0^\infty w(b) \left(\frac{b^2}{1+b^2} \right)^N \pi^{\frac{N}{2}} (1+b^2)^{\frac{N}{2}} \,\mathrm{d}b \\
&= \int_0^{b_{\max}} b \left(\frac{b^2}{1+b^2} \right)^{\frac{N}{2}} \mathrm{d}b \ ,
\end{aligned}
$$

$$
\begin{aligned}
D_2^{\mathrm{e}}(\mathcal{S}) &= \int_0^\infty w(b) \left(\frac{b^2}{1+b^2} \right)^{\frac{N}{2}} (2\pi)^{\frac{N}{2}} (1+b^2)^{\frac{N}{2}} \frac{(2\pi)^{\frac{N}{2}} b^N}{2L} \cdot \\
&\qquad \int_{\mathbb{R}^N} \mathcal{N}(\boldsymbol{m}\,; \boldsymbol{0}, (1+b^2)\mathbf{I}_N) \sum_{i=1}^L \Big(\mathcal{N}(\boldsymbol{m}\,; \boldsymbol{s}^{(i)}, b^2\mathbf{I}_N) + \mathcal{N}(\boldsymbol{m}\,; -\boldsymbol{s}^{(i)}, b^2\mathbf{I}_N) \Big) \,\mathrm{d}\boldsymbol{m} \ \mathrm{d}b \\
&= \int_0^\infty w(b) \frac{(2\pi)^N b^{2N}}{2L} \frac{1}{(2\pi)^{\frac{N}{2}} (1+2b^2)^{\frac{N}{2}}} \cdot \\
&\qquad \sum_{i=1}^L \left(\exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)}\|_2^2}{(1+2b^2)} \right) + \exp\left(-\frac{1}{2} \frac{\|-\boldsymbol{s}^{(i)}\|_2^2}{(1+2b^2)} \right) \right) \mathrm{d}b \\
&= \int_0^{b_{\max}} \frac{2b}{2L} \left(\frac{2b^2}{1+2b^2} \right)^{\frac{N}{2}} \cdot \sum_{i=1}^L \exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)}\|_2^2}{(1+2b^2)} \right) \mathrm{d}b \ ,
\end{aligned}
$$

$$
\begin{aligned}
D_3^{\mathrm{e}}(\mathcal{S}) = \int_0^\infty w(b) \left(\frac{(2\pi)^{\frac{N}{2}} b^N}{2L} \right)^2 \int_{\mathbb{R}^N} &\sum_{i=1}^L \Big(\mathcal{N}(\boldsymbol{m}\,; \boldsymbol{s}^{(i)}, b^2\mathbf{I}_N) + \mathcal{N}(\boldsymbol{m}\,; -\boldsymbol{s}^{(i)}, b^2\mathbf{I}_N) \Big) \cdot \\
&\sum_{j=1}^L \Big(\mathcal{N}(\boldsymbol{m}\,; \boldsymbol{s}^{(j)}, b^2\mathbf{I}_N) + \mathcal{N}(\boldsymbol{m}\,; -\boldsymbol{s}^{(j)}, b^2\mathbf{I}_N) \Big) \,\mathrm{d}\boldsymbol{m} \ \mathrm{d}b
\end{aligned}
$$

$$= \int_0^\infty w(b) \frac{(2\pi)^N b^{2N}}{(2L)^2} \frac{1}{(2\pi)^{\frac{N}{2}} (2b^2)^{\frac{N}{2}}}$$

$$\sum_{i=1}^{L} \sum_{j=1}^{L} \left(\exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)} - \boldsymbol{s}^{(j)}\|_2^2}{2b^2} \right) + \exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)} + \boldsymbol{s}^{(j)}\|_2^2}{2b^2} \right) + \right.$$

$$\left. \exp\left(-\frac{1}{2} \frac{\| -\boldsymbol{s}^{(i)} - \boldsymbol{s}^{(j)}\|_2^2}{2b^2} \right) + \exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(j)} - \boldsymbol{s}^{(i)}\|_2^2}{2b^2} \right) \right) \mathrm{d}b$$

$$= \int_0^{b_{\max}} \frac{2b}{(2L)^2} \sum_{i=1}^{L} \sum_{j=1}^{L} \left(\exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)} - \boldsymbol{s}^{(j)}\|_2^2}{2b^2} \right) + \exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)} + \boldsymbol{s}^{(j)}\|_2^2}{2b^2} \right) \right) \mathrm{d}b \ .$$

■

A.3 Proof of Theorem 2.2

Like in [111], to efficiently compute the term D_3^{e} we use that for $z > 0$

$$\int_0^{b_{\max}} \frac{2}{b} \exp\left(-\frac{1}{2} \frac{z}{2b^2} \right) \mathrm{d}b = -\operatorname{Ei}\left(-\frac{1}{2} \frac{z}{2b_{\max}^2} \right) \ , \tag{A.2}$$

where $\operatorname{Ei}(x)$ denotes the exponential integral defined as [113, Chap. 5]

$$\operatorname{Ei}(x) := \int_{-\infty}^{x} \frac{e^t}{t} \, \mathrm{d}t \ , \quad x > 0 \ .$$

Moreover, the product rule gives

$$\frac{b_{\max}^2}{2} \exp\left(-\frac{1}{2} \frac{z}{2b_{\max}^2} \right) = \int_0^{b_{\max}} b \exp\left(-\frac{1}{2} \frac{z}{2b^2} \right) \mathrm{d}b + \frac{z}{4} \int_0^{b_{\max}} \frac{1}{b} \exp\left(-\frac{1}{2} \frac{z}{2b^2} \right) \mathrm{d}b \ ,$$

and together with (A.2) we obtain

$$\int_0^{b_{\max}} b \exp\left(-\frac{1}{2} \frac{z}{2b^2} \right) \mathrm{d}b = \frac{b_{\max}^2}{2} \exp\left(-\frac{1}{2} \frac{z}{2b_{\max}^2} \right) + \frac{z}{8} \operatorname{Ei}\left(-\frac{1}{2} \frac{z}{2b_{\max}^2} \right) \ . \tag{A.3}$$

Note that $\operatorname{Ei}(x)$ is not defined for $x = 0$. Nonetheless, for $z = 0$, we have

$$\int_0^{b_{\max}} b \exp\left(-\frac{1}{2} \frac{0}{2b^2} \right) \mathrm{d}b = \int_0^{b_{\max}} b \, \mathrm{d}b = \frac{b_{\max}^2}{2} \ .$$

In order to cover both cases $z > 0$ and $z = 0$ in the same expression, we introduce the function

$$\operatorname{Ei}_0(x) := \begin{cases} 0 & \text{, if } x = 0 \\ \operatorname{Ei}(x) & \text{, elsewhere} \end{cases} \ , \tag{A.4}$$

and replace $\operatorname{Ei}(x)$ in (A.3) with $\operatorname{Ei}_0(x)$, which results in

$$\int_0^{b_{\max}} b \exp\left(-\frac{1}{2} \frac{z}{2b^2} \right) \mathrm{d}b = \frac{b_{\max}^2}{2} \exp\left(-\frac{1}{2} \frac{z}{2b_{\max}^2} \right) + \frac{z}{8} \operatorname{Ei}_0\left(-\frac{1}{2} \frac{z}{2b_{\max}^2} \right) \ . \tag{A.5}$$

Based on this, we get the closed-form expression for D_3^{e} according to

$$D_3^{\mathrm{e}}(\mathcal{S}) = \frac{2}{(2L)^2} \sum_{i=1}^{L} \sum_{j=1}^{L} \left(\frac{b_{\max}^2}{2} \left(\exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)} - \boldsymbol{s}^{(j)}\|_2^2}{2b_{\max}^2} \right) + \exp\left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)} + \boldsymbol{s}^{(j)}\|_2^2}{2b_{\max}^2} \right) \right) + \frac{1}{8} \left(\|\boldsymbol{s}^{(i)} - \boldsymbol{s}^{(j)}\|_2^2 \operatorname{Ei}_0 \left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)} - \boldsymbol{s}^{(j)}\|_2^2}{2b_{\max}^2} \right) + \|\boldsymbol{s}^{(i)} + \boldsymbol{s}^{(j)}\|_2^2 \operatorname{Ei}_0 \left(-\frac{1}{2} \frac{\|\boldsymbol{s}^{(i)} + \boldsymbol{s}^{(j)}\|_2^2}{2b_{\max}^2} \right) \right) \right) .$$

■

A.4 Proof of Distance D^{o}

Like for the modified CvM distance D^{e}, we can split D^{o} into three parts according to

$$D^{\mathrm{o}}(\mathcal{S}) = \underbrace{\int_0^\infty w(b) \int_{\mathbb{R}^N} F_{\mathcal{N}}(\boldsymbol{m}, b)^2 \,\mathrm{d}\boldsymbol{m} \,\mathrm{d}b}_{=:D_1^{\mathrm{o}}} - \underbrace{2\int_0^\infty w(b) \int_{\mathbb{R}^N} F_{\mathcal{N}}(\boldsymbol{m}, b) F_\delta^{\mathrm{o}}(\mathcal{S}, \boldsymbol{m}, b) \,\mathrm{d}\boldsymbol{m} \,\mathrm{d}b}_{=:D_2^{\mathrm{o}}(\mathcal{S})} + \underbrace{\int_0^\infty w(b) \int_{\mathbb{R}^N} F_\delta^{\mathrm{o}}(\mathcal{S}, \boldsymbol{m}, b)^2 \,\mathrm{d}\boldsymbol{m} \,\mathrm{d}b}_{=:D_3^{\mathrm{o}}(\mathcal{S})} .$$

We directly see that $D_1^{\mathrm{o}} = D_1^{\mathrm{e}}$, and for the second and third part we get

$$\begin{aligned} D_2^{\mathrm{o}}(\mathcal{S}) &= \frac{2L}{2L+1} D_2^{\mathrm{e}}(\mathcal{S}) + \int_0^\infty w(b) \left(\frac{b^2}{1+b^2} \right)^{\frac{N}{2}} (2\pi)^{\frac{N}{2}} (1+b^2)^{\frac{N}{2}} \frac{(2\pi)^{\frac{N}{2}} b^N}{2L+1} \cdot \\ &\qquad \int_{\mathbb{R}^N} \mathcal{N}(\boldsymbol{m}; \mathbf{0}, (1+b^2)\mathbf{I}_N) \mathcal{N}(\boldsymbol{m}; \mathbf{0}, b^2\mathbf{I}_N) \,\mathrm{d}\boldsymbol{m} \;\mathrm{d}b \\ &= \frac{2L}{2L+1} D_2^{\mathrm{e}}(\mathcal{S}) + \int_0^\infty w(b) \frac{(2\pi)^N b^{2N}}{2L+1} \frac{1}{(2\pi)^{\frac{N}{2}} (1+2b^2)^{\frac{N}{2}}} \,\mathrm{d}b \\ &= \frac{2L}{2L+1} D_2^{\mathrm{e}}(\mathcal{S}) + \int_0^{b_{\max}} \frac{b}{2L+1} \left(\frac{2b^2}{1+2b^2} \right)^{\frac{N}{2}} \mathrm{d}b \end{aligned}$$

and

$$\begin{aligned} D_3^{\mathrm{o}}(\mathcal{S}) = \frac{(2L)^2}{(2L+1)^2} D_3^{\mathrm{e}}(\mathcal{S}) + \int_0^\infty w(b) \left(\frac{(2\pi)^{\frac{N}{2}} b^N}{2L+1} \right)^2 \cdot \\ \int_{\mathbb{R}^N} \mathcal{N}(\boldsymbol{m}; \mathbf{0}, b^2\mathbf{I}_N)^2 + 2\mathcal{N}(\boldsymbol{m}; \mathbf{0}, b^2\mathbf{I}_N) \sum_{i=1}^{L} \Big(\mathcal{N}(\boldsymbol{m}; \boldsymbol{s}^{(i)}, b^2\mathbf{I}_N) + \\ \mathcal{N}(\boldsymbol{m}; -\boldsymbol{s}^{(i)}, b^2\mathbf{I}_N) \Big) \,\mathrm{d}\boldsymbol{m} \;\mathrm{d}b \end{aligned}$$

$$= \frac{(2L)^2}{(2L+1)^2} D_3^{\mathrm{e}}(\mathcal{S}) + \int_0^{\infty} w(b) \frac{(2\pi)^N b^{2N}}{(2L+1)^2} \frac{1}{(2\pi)^{\frac{N}{2}} (2b^2)^{\frac{N}{2}}} \cdot$$
$$\left(1 + 2\sum_{i=1}^{L} \left(\exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)}\|_2^2}{2b^2}\right) + \exp\left(-\frac{1}{2}\frac{\|-\boldsymbol{s}^{(i)}\|_2^2}{2b^2}\right)\right)\right) \mathrm{d}b$$
$$= \frac{(2L)^2}{(2L+1)^2} D_3^{\mathrm{e}}(\mathcal{S}) + \frac{b_{\max}^2}{2(2L+1)^2} + \int_0^{b_{\max}} \frac{4b}{(2L+1)^2} \sum_{i=1}^{L} \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)}\|_2^2}{2b^2}\right) \mathrm{d}b \ ,$$

respectively. ■

A.5 Proof of Theorem 2.4

A closed-form expression for D_3^{o} can directly be obtained by using again (A.5) and the closed-form expression for D_3^{e} resulting in

$$D_3^{\mathrm{o}}(\mathcal{S}) = \frac{(2L)^2}{(2L+1)^2} D_3^{\mathrm{e}}(\mathcal{S}) + \frac{b_{\max}^2}{2(2L+1)^2} +$$
$$\frac{4}{(2L+1)^2} \sum_{i=1}^{L} \left(\frac{b_{\max}^2}{2} \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)}\|_2^2}{2b_{\max}^2}\right) + \frac{1}{8}\|\boldsymbol{s}^{(i)}\|_2^2 \operatorname{Ei}_0\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)}\|_2^2}{2b_{\max}^2}\right)\right) \ .$$

■

A.6 Boundedness of D^{e} and D^{o}

We show the boundedness of the distances D^{e} and D^{o} for an increasing dimension N.

We start with the distance D^{e}. For a given $b_{\max}$, the following limits exist

$$\lim_{N\to\infty} D_1^{\mathrm{e}} = \lim_{N\to\infty} \int_0^{b_{\max}} b \underbrace{\left(\frac{b^2}{1+b^2}\right)^{\frac{N}{2}}}_{\to 0 \text{ for } N\to\infty} \mathrm{d}b = 0 \ ,$$

$$\lim_{N\to\infty} D_2^{\mathrm{e}}(\mathcal{S}) = \lim_{N\to\infty} \int_0^{b_{\max}} \frac{2b}{2L} \left(\frac{2b^2}{1+2b^2}\right)^{\frac{N}{2}} \underbrace{\sum_{i=1}^{L} \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)}\|_2^2}{(1+2b^2)}\right)}_{\le L} \mathrm{d}b$$
$$\le \lim_{N\to\infty} \int_0^{b_{\max}} b \underbrace{\left(\frac{2b^2}{1+2b^2}\right)^{\frac{N}{2}}}_{\to 0 \text{ for } N\to\infty} \mathrm{d}b = 0 \ ,$$

$$\lim_{N\to\infty} D_3^{\mathrm{e}}(\mathcal{S}) = \lim_{N\to\infty} \int_0^{b_{\max}} \frac{2b}{(2L)^2} \underbrace{\sum_{i=1}^{L}\sum_{j=1}^{L} \left(\exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)} - \boldsymbol{s}^{(j)}\|_2^2}{2b^2}\right) + \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)} + \boldsymbol{s}^{(j)}\|_2^2}{2b^2}\right)\right)}_{\le 2L^2} \mathrm{d}b$$
$$\le \int_0^{b_{\max}} b \, \mathrm{d}b = \frac{b_{\max}^2}{2} \ .$$

Hence, the distance D^{e} is bounded by $b_{\max}$ according to

$$\lim_{N\to\infty} D^{\mathrm{e}}(\mathcal{S}) = \lim_{N\to\infty} D_1^{\mathrm{e}} - 2\lim_{N\to\infty} D_2^{\mathrm{e}}(\mathcal{S}) + \lim_{N\to\infty} D_3^{\mathrm{e}}(\mathcal{S}) \leq \frac{b_{\max}^2}{2} \ .$$

Next, we consider the distance D^{o}. For this, the following limits exist

$$\begin{aligned}
&\lim_{N\to\infty} D_1^{\mathrm{o}} = \lim_{N\to\infty} D_1^{\mathrm{e}} = 0 \ , \\
&\lim_{N\to\infty} D_2^{\mathrm{o}}(\mathcal{S}) = \frac{2L}{2L+1} \lim_{N\to\infty} D_2^{\mathrm{e}}(\mathcal{S}) + \lim_{N\to\infty} \int_0^{b_{\max}} \frac{b}{2L+1} \underbrace{\left(\frac{2b^2}{1+2b^2}\right)^{\frac{N}{2}}}_{\to 0 \text{ for } N\to\infty} \mathrm{d}b \leq 0 \ , \\
&\lim_{N\to\infty} D_3^{\mathrm{o}}(\mathcal{S}) = \frac{(2L)^2}{(2L+1)^2} \lim_{N\to\infty} D_3^{\mathrm{e}}(\mathcal{S}) + \frac{b_{\max}^2}{2(2L+1)^2} + \\
&\qquad \lim_{N\to\infty} \int_0^{b_{\max}} \frac{4b}{(2L+1)^2} \underbrace{\sum_{i=1}^{L} \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)}\|_2^2}{2b^2}\right)}_{\leq L} \mathrm{d}b \\
&\qquad \leq \frac{b_{\max}^2}{2}\frac{(2L)^2+1}{(2L+1)^2} + \frac{4L}{(2L+1)^2}\int_0^{b_{\max}} b \,\mathrm{d}b \\
&\qquad = \frac{b_{\max}^2}{2}\frac{(2L)^2+4L+1}{(2L+1)^2} = \frac{b_{\max}^2}{2} \ .
\end{aligned}$$

Hence, like the distance D^{e}, D^{o} is bounded by $b_{\max}$ according to

$$\lim_{N\to\infty} D^{\mathrm{o}}(\mathcal{S}) = \lim_{N\to\infty} D_1^{\mathrm{o}} - 2\lim_{N\to\infty} D_2^{\mathrm{o}}(\mathcal{S}) + \lim_{N\to\infty} D_3^{\mathrm{o}}(\mathcal{S}) \leq \frac{b_{\max}^2}{2} \ .$$

■

A.7 Invariance of D^{e} and D^{o} under Rotation/Reflection

We want to prove that the distances D^{e} and D^{o} are invariant under rotation and reflection. Let $\mathbf{R} \in \mathrm{I\!R}^{N\times N}$ be an orthogonal matrix and the sets

$$\mathcal{S} = \{\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(L)}\}$$

and

$$\mathcal{S}' = \{\mathbf{R}\boldsymbol{s}^{(1)}, \ldots, \mathbf{R}\boldsymbol{s}^{(L)}\} \ ,$$

with vectors $\boldsymbol{s}^{(i)} \in \mathrm{I\!R}^N$, parameterize either a point-symmetric Dirac mixture with an even number of samples (2.8) or a point-symmetric Dirac mixture with an odd number of samples (2.9). Furthermore, for $\boldsymbol{a}, \boldsymbol{b} \in \mathrm{I\!R}^N$ it holds

$$\begin{aligned}
\|\mathbf{R}\boldsymbol{a}\|_2^2 &= \|\boldsymbol{a}\|_2^2 \ , \\
\|\mathbf{R}\boldsymbol{a} \pm \mathbf{R}\boldsymbol{b}\|_2^2 &= \|\boldsymbol{a} \pm \boldsymbol{b}\|_2^2 \ .
\end{aligned}$$

From this, we can directly see that $D^{\mathrm{e}}(\mathcal{S}) = D^{\mathrm{e}}(\mathcal{S}')$ and $D^{\mathrm{o}}(\mathcal{S}) = D^{\mathrm{o}}(\mathcal{S}')$. ■

A.8 Proof of Theorem 2.5

For $\|\boldsymbol{s}^{(i)} \pm \boldsymbol{s}^{(j)}\|_2^2 > 0$, the terms

$$\int_0^{b_{\max}} \frac{1}{b}(s_d^{(i)} \pm s_d^{(j)}) \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)} \pm \boldsymbol{s}^{(j)}\|_2^2}{2b^2}\right) \mathrm{d}b \tag{A.6}$$

can be computed with the aid of (A.2) according to

$$-\frac{1}{2}(s_d^{(i)} \pm s_d^{(j)}) \operatorname{Ei}\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)} \pm \boldsymbol{s}^{(j)}\|_2^2}{2b_{\max}^2}\right) \quad . \tag{A.7}$$

For $\|\boldsymbol{s}^{(i)} \pm \boldsymbol{s}^{(j)}\|_2^2 = 0$, we especially have $s_d^{(i)} \pm s_d^{(j)} = 0$. Consequently, the integral (A.6) converges to zero as well. Like in the closed-form expression for D_3^{e}, we can replace $\operatorname{Ei}(x)$ in (A.7) with $\operatorname{Ei}_0(x)$ to get a valid expression for $\|\boldsymbol{s}^{(i)} \pm \boldsymbol{s}^{(j)}\|_2^2 \geq 0$. Based on this, we can get a closed-form expression for $\frac{\partial D_3^{\mathrm{e}}(\mathcal{S})}{\partial s_d^{(i)}}$ according to

$$\begin{aligned}\frac{\partial D_3^{\mathrm{e}}(\mathcal{S})}{\partial s_d^{(i)}} = \frac{1}{(2L)^2}\sum_{j=1}^{L}\Bigg(&(s_d^{(i)} - s_d^{(j)}) \operatorname{Ei}_0\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)} - \boldsymbol{s}^{(j)}\|_2^2}{2b_{\max}^2}\right) + \\ &(s_d^{(i)} + s_d^{(j)}) \operatorname{Ei}_0\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)} + \boldsymbol{s}^{(j)}\|_2^2}{2b_{\max}^2}\right)\Bigg) \quad .\end{aligned} \tag{A.8}$$

■

A.9 Proof of Theorem 2.6

Analogously to the closed-form expression for $\frac{\partial D_3^{\mathrm{e}}(\mathcal{S})}{\partial s_d^{(i)}}$, we can obtain a closed-form expression for $\frac{\partial D_3^{\mathrm{o}}(\mathcal{S})}{\partial s_d^{(i)}}$ by means of (A.8), (A.2), and (A.4) according to

$$\frac{\partial D_3^{\mathrm{o}}(\mathcal{S})}{\partial s_d^{(i)}} = \frac{(2L)^2}{(2L+1)^2}\frac{\partial D_3^{\mathrm{e}}(\mathcal{S})}{\partial s_d^{(i)}} + \frac{s_d^{(i)}}{(2L+1)^2} \operatorname{Ei}_0\left(-\frac{1}{2}\frac{\|\boldsymbol{s}^{(i)}\|_2^2}{2b_{\max}^2}\right) \quad .$$

■

A.10 Sample Covariance Matrix Correction

Given a point-symmetric Dirac mixture (2.8) or (2.9) parameterized by $\{\boldsymbol{z}^{(i)}\}_{i=1}^{L}$. Our goal is to find a linear transformation $\mathbf{T}$ such that the point-symmetric Dirac mixture parameterized by

$$\boldsymbol{s}^{(i)} = \mathbf{T}\boldsymbol{z}^{(i)} \quad , \quad \forall i \in \{1, \ldots, L\} \quad , \tag{A.9}$$

has an identity sample covariance matrix, i.e.,

$$\mathbf{\Sigma}^{(s)} = \frac{2}{M}\sum_{i=1}^{L}\boldsymbol{s}^{(i)}\big(\boldsymbol{s}^{(i)}\big)^{\top} = \mathbf{I}_N \quad , \tag{A.10}$$

where M is either equal to $2L$ or equal to $2L+1$. Plugging (A.9) into (A.10) gives

$$\begin{aligned}\mathbf{\Sigma}^{(s)} &= \frac{2}{M}\sum_{i=1}^{L}\big(\mathbf{T}\boldsymbol{z}^{(i)}\big)\big(\mathbf{T}\boldsymbol{z}^{(i)}\big)^\top \\ &= \mathbf{T}\frac{2}{M}\sum_{i=1}^{L}\boldsymbol{z}^{(i)}\big(\boldsymbol{z}^{(i)}\big)^\top\mathbf{T}^\top \\ &= \mathbf{T}\mathbf{\Sigma}^{(z)}\mathbf{T}^\top = \mathbf{I}_N \ .\end{aligned}$$

Furthermore, with the matrix decomposition of $\mathbf{\Sigma}^{(z)} = \mathbf{A}\mathbf{A}^\top$, we get

$$(\mathbf{T}\mathbf{A})(\mathbf{T}\mathbf{A})^\top = \mathbf{I}_N \ ,$$

which can be satisfied with $\mathbf{T} = \mathbf{A}^{-1}$. The matrix decomposition of $\mathbf{\Sigma}^{(s)}$ can be computed, for example, with the Cholesky decomposition or the eigendecomposition. ■

Appendix B

Proof of the Optimal Sample-Based Fusion for Distributed State Estimation

In this appendix, we prove that the sample-based correlation reconstruction from Chapter 4 is optimal in the mean square error sense. That is, the correlation between two sensor nodes i and j can be optimally reconstructed when their respective set of samples are initialized with (4.8), predicted with (4.10), and updated according to (4.12).

It is assumed that both nodes are re-initialized in time step k, and that the local state estimates are fully correlated with $\mathbf{P}_{k|k}^{(i,j)} = \mathbf{P}_{k|k}$. Moreover, w.l.o.g. it is assumed that

1. the ith node performs a prediction from time step k to $k+1$ followed by an update in time step $k+1$ and another prediction from time step $k+1$ to $k+2$, and
2. the jth node performs a prediction from time step k to $k+1$ followed by another prediction from time step $k+1$ to $k+2$ and finally performs an update in time step $k+2$.

Then, the correct correlation between their local estimates at time step $k+2$ is [172]

$$\mathbf{P}_{k+2|k+2}^{(i,j)} = (\mathbf{A}_{k+2}\mathbf{D}_3(\mathbf{D}_1 + \mathbf{D}_2)\mathbf{A}_{k+2}^\top + \mathbf{D}_4)\mathbf{D}_5^\top \ ,$$

with

$$\begin{aligned}
\mathbf{D}_1 &= \mathbf{A}_{k+1}\mathbf{P}_{k|k}^{(i,j)}\mathbf{A}_{k+1}^\top = \mathbf{A}_{k+1}\mathbf{P}_{k|k}\mathbf{A}_{k+1}^\top \ , \\
\mathbf{D}_2 &= \mathbf{B}_{k+1}\mathbf{Q}_{k+1}\mathbf{B}_{k+1}^\top \ , \\
\mathbf{D}_3 &= \mathbf{I} - \mathbf{K}_{k+1}^{(i)}\mathbf{H}_{k+1}^{(i)} \ , \\
\mathbf{D}_4 &= \mathbf{B}_{k+2}\mathbf{Q}_{k+2}\mathbf{B}_{k+2}^\top \ , \\
\mathbf{D}_5 &= \mathbf{I} - \mathbf{K}_{k+2}^{(j)}\mathbf{H}_{k+2}^{(j)} \ .
\end{aligned}$$

The correlation samples of the ith node are

$$\begin{aligned}
\boldsymbol{c}_{k+2|k+2}^{(i,m)} = \mathbf{B}_{k+2}\boldsymbol{w}_{k+2}^{(m)} + \mathbf{A}_{k+2}\mathbf{D}_3\mathbf{B}_{k+1}\boldsymbol{w}_{k+1}^{(m)} + \\
\mathbf{A}_{k+2}\mathbf{D}_3\mathbf{A}_{k+1}\boldsymbol{c}_{k|k}^{(i,m)} \ , \quad \forall m \in \{1, \ldots, M\} \ ,
\end{aligned}$$

and the correlation samples of the jth node are

$$\begin{aligned}
(\boldsymbol{c}_{k+2|k+2}^{(j,m)})^\top = (\boldsymbol{w}_{k+2}^{(m)})^\top\mathbf{B}_{k+2}^\top\mathbf{D}_5^\top + (\boldsymbol{w}_{k+1}^{(m)})^\top\mathbf{B}_{k+1}^\top\mathbf{A}_{k+2}^\top\mathbf{D}_5^\top + \\
(\boldsymbol{c}_{k|k}^{(j,m)})^\top\mathbf{A}_{k+1}^\top\mathbf{A}_{k+2}^\top\mathbf{D}_5^\top \ , \quad \forall m \in \{1, \ldots, M\} \ .
\end{aligned}$$

First, from the fact that the samples obtained from (4.8) have zero mean, we can see that both sample means are zero according to

$$\begin{aligned}\hat{\boldsymbol{c}}^{(i)}_{k+2|k+2} &= \mathbf{A}_{k+2}\mathbf{D}_3\mathbf{A}_{k+1}\hat{\boldsymbol{c}}^{(i)}_{k|k} = \mathbf{0} \ , \\ \hat{\boldsymbol{c}}^{(j)}_{k+2|k+2} &= \mathbf{D}_5\mathbf{A}_{k+2}\mathbf{A}_{k+1}\hat{\boldsymbol{c}}^{(j)}_{k|k} = \mathbf{0} \ .\end{aligned}$$

Second, for both nodes the samples obtained from (4.8) are identical, and hence it holds that

$$\begin{aligned}\frac{1}{M}\sum_{m=1}^{M}\boldsymbol{c}^{(i,m)}_{k|k}(\boldsymbol{c}^{(j,m)}_{k|k})^\top &= \mathbf{P}_{k|k} \ , \\ \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{w}^{(m)}_{k+1}(\boldsymbol{w}^{(m)}_{k+1})^\top &= \mathbf{Q}_{k+1} \ , \\ \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{w}^{(m)}_{k+2}(\boldsymbol{w}^{(m)}_{k+2})^\top &= \mathbf{Q}_{k+2} \ ,\end{aligned}$$

and

$$\begin{aligned}\frac{1}{M}\sum_{m=1}^{M}\boldsymbol{c}^{(i,m)}_{k|k}(\boldsymbol{w}^{(m)}_{k+1})^\top &= \mathbf{0} \ , \\ \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{c}^{(i,m)}_{k|k}(\boldsymbol{w}^{(m)}_{k+2})^\top &= \mathbf{0} \ , \\ \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{c}^{(j,m)}_{k|k}(\boldsymbol{w}^{(m)}_{k+1})^\top &= \mathbf{0} \ , \\ \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{c}^{(j,m)}_{k|k}(\boldsymbol{w}^{(m)}_{k+2})^\top &= \mathbf{0} \ , \\ \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{w}^{(m)}_{k+1}(\boldsymbol{w}^{(m)}_{k+2})^\top &= \mathbf{0} \ .\end{aligned}$$

Finally, by exploiting these, the correlation between both nodes obtained from their respective samples is

$$\begin{aligned}\mathbf{S}^{(i,j)}_{k+2|k+2} &= \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{c}^{(i,m)}_{k+2|k+2}(\boldsymbol{c}^{(j,m)}_{k+2|k+2})^\top \\ &= \mathbf{A}_{k+2}\mathbf{D}_3\mathbf{A}_{k+1}\mathbf{P}_{k|k}\mathbf{A}^\top_{k+1}\mathbf{A}^\top_{k+2}\mathbf{D}^\top_5 + \\ &\quad \mathbf{A}_{k+2}\mathbf{D}_3\mathbf{B}_{k+1}\mathbf{Q}_{k+1}\mathbf{B}^\top_{k+1}\mathbf{A}^\top_{k+2}\mathbf{D}^\top_5 + \\ &\quad \mathbf{B}_{k+2}\mathbf{Q}_{k+2}\mathbf{B}^\top_{k+2}\mathbf{D}^\top_5 \\ &= (\mathbf{A}_{k+2}\mathbf{D}_3(\mathbf{D}_1+\mathbf{D}_2)\mathbf{A}^\top_{k+2}+\mathbf{D}_4)\mathbf{D}^\top_5 \\ &= \mathbf{P}^{(i,j)}_{k+2|k+2} \ .\end{aligned}$$

■

Appendix C

Proof of the Semi-Analytic Progressive Gaussian Filter

We prove that the product of the Gaussian densities $f_{k|k-1}(\boldsymbol{x}_k^{(u)} \,|\, \boldsymbol{x}_k^{(o)})$ and $f_{k|k}(\boldsymbol{x}_k^{(o)})$ yields an unnormalized Gaussian. We have

$$\begin{aligned} f_{k|k-1}(\boldsymbol{x}_k^{(u)} \,|\, \boldsymbol{x}_k^{(o)}) \cdot f_{k|k}(\boldsymbol{x}_k^{(o)}) &= \mathcal{N}(\boldsymbol{x}_k^{(u)}\,; \hat{\boldsymbol{x}}_{k|k-1}^{(u\,|\,o)}, \mathbf{P}_{k|k-1}^{(u\,|\,o)}) \cdot \mathcal{N}(\boldsymbol{x}_k^{(o)}\,; \hat{\boldsymbol{x}}_{k|k}^{(o)}, \mathbf{P}_{k|k}^{(o)}) \\ &\propto \exp\left(-\frac{1}{2}z\right) \end{aligned}$$

where

$$\begin{aligned} z =& \big(\boldsymbol{x}_k^{(u)} - \hat{\boldsymbol{x}}_{k|k-1}^{(u\,|\,o)}\big)^\top \big(\mathbf{P}_{k|k-1}^{(u\,|\,o)}\big)^{-1} \big(\boldsymbol{x}_k^{(u)} - \hat{\boldsymbol{x}}_{k|k-1}^{(u\,|\,o)}\big) + \\ & \big(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)}\big)^\top \big(\mathbf{P}_{k|k}^{(o)}\big)^{-1} \big(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)}\big) \ . \end{aligned}$$

First, with the definition of $\hat{\boldsymbol{x}}_{k|k-1}^{(u\,|\,o)}$ (5.17) and by adding twice the term $\hat{\boldsymbol{x}}_{k|k}^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)}$, we get

$$\begin{aligned} z =& \big(\boldsymbol{x}_k^{(u)} - \hat{\boldsymbol{x}}_{k|k-1}^{(u)} - \mathbf{K}_k(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k-1}^{(o)})\big)^\top \big(\mathbf{P}_{k|k-1}^{(u\,|\,o)}\big)^{-1} \cdot \\ & \big(\boldsymbol{x}_k^{(u)} - \hat{\boldsymbol{x}}_{k|k-1}^{(u)} - \mathbf{K}_k(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k-1}^{(o)})\big) + \\ & \big(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)}\big)^\top \big(\mathbf{P}_{k|k}^{(o)}\big)^{-1} \big(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)}\big) \\ =& \big(\boldsymbol{x}_k^{(u)} - \hat{\boldsymbol{x}}_{k|k-1}^{(u)} - \mathbf{K}_k(\hat{\boldsymbol{x}}_{k|k}^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)} + \boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k-1}^{(o)})\big)^\top \big(\mathbf{P}_{k|k-1}^{(u\,|\,o)}\big)^{-1} \cdot \\ & \big(\boldsymbol{x}_k^{(u)} - \hat{\boldsymbol{x}}_{k|k-1}^{(u)} - \mathbf{K}_k(\hat{\boldsymbol{x}}_{k|k}^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)} + \boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k-1}^{(o)})\big) + \\ & \big(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)}\big)^\top \big(\mathbf{P}_{k|k}^{(o)}\big)^{-1} \big(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)}\big) \ . \end{aligned}$$

Second, by defining the mean

$$\hat{\boldsymbol{x}}_{k|k}^{(u)} := \hat{\boldsymbol{x}}_{k|k-1}^{(u)} + \mathbf{K}_k\big(\hat{\boldsymbol{x}}_{k|k}^{(o)} - \hat{\boldsymbol{x}}_{k|k-1}^{(o)}\big) \ , \tag{C.1}$$

i.e., the posterior mean of the unobservable state variables, we get

$$\begin{aligned}
z =& \big(\boldsymbol{x}_k^{(u)} - \hat{\boldsymbol{x}}_{k|k}^{(u)} - \mathbf{K}_k(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)})\big)^\top \big(\mathbf{P}_{k|k-1}^{(u\,|\,o)}\big)^{-1} \cdot \\
& \big(\boldsymbol{x}_k^{(u)} - \hat{\boldsymbol{x}}_{k|k}^{(u)} - \mathbf{K}_k(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)})\big) + \\
& \big(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)}\big)^\top \big(\mathbf{P}_{k|k}^{(o)}\big)^{-1} \big(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)}\big) \\
=& \big(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)}\big)^\top \Big(\big(\mathbf{P}_{k|k}^{(o)}\big)^{-1} + \mathbf{K}_k^\top \big(\mathbf{P}_{k|k-1}^{(u\,|\,o)}\big)^{-1} \mathbf{K}_k\Big) \big(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)}\big) - \\
& \big(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)}\big)^\top \mathbf{K}_k^\top \big(\mathbf{P}_{k|k-1}^{(u\,|\,o)}\big)^{-1} \big(\boldsymbol{x}_k^{(u)} - \hat{\boldsymbol{x}}_{k|k}^{(u)}\big) - \\
& \big(\boldsymbol{x}_k^{(u)} - \hat{\boldsymbol{x}}_{k|k}^{(u)}\big)^\top \big(\mathbf{P}_{k|k-1}^{(u\,|\,o)}\big)^{-1} \mathbf{K}_k \big(\boldsymbol{x}_k^{(o)} - \hat{\boldsymbol{x}}_{k|k}^{(o)}\big) + \\
& \big(\boldsymbol{x}_k^{(u)} - \hat{\boldsymbol{x}}_{k|k}^{(u)}\big)^\top \big(\mathbf{P}_{k|k-1}^{(u\,|\,o)}\big)^{-1} \big(\boldsymbol{x}_k^{(u)} - \hat{\boldsymbol{x}}_{k|k}^{(u)}\big) \\
=& \left(\begin{bmatrix} \boldsymbol{x}_k^{(o)} \\ \boldsymbol{x}_k^{(u)} \end{bmatrix} - \begin{bmatrix} \hat{\boldsymbol{x}}_{k|k}^{(o)} \\ \hat{\boldsymbol{x}}_{k|k}^{(u)} \end{bmatrix}\right)^\top \mathbf{\Sigma}^{-1} \left(\begin{bmatrix} \boldsymbol{x}_k^{(o)} \\ \boldsymbol{x}_k^{(u)} \end{bmatrix} - \begin{bmatrix} \hat{\boldsymbol{x}}_{k|k}^{(o)} \\ \hat{\boldsymbol{x}}_{k|k}^{(u)} \end{bmatrix}\right) \quad ,
\end{aligned} \tag{C.2}$$

with block matrix

$$\mathbf{\Sigma}^{-1} = \begin{bmatrix} \big(\mathbf{P}_{k|k}^{(o)}\big)^{-1} + \mathbf{K}_k^\top \big(\mathbf{P}_{k|k-1}^{(u\,|\,o)}\big)^{-1} \mathbf{K}_k & -\mathbf{K}_k^\top \big(\mathbf{P}_{k|k-1}^{(u\,|\,o)}\big)^{-1} \\ -\big(\mathbf{P}_{k|k-1}^{(u\,|\,o)}\big)^{-1} \mathbf{K}_k & \big(\mathbf{P}_{k|k-1}^{(u\,|\,o)}\big)^{-1} \end{bmatrix} \quad .$$

Third, using the definition of $\mathbf{P}_{k|k-1}^{(u\,|\,o)}$ (5.18) and the formulas for inverting a 2×2 block matrix, e.g., see [173, Sec. 0.7.3], together with equating the coefficients, we get the inverse of $\mathbf{\Sigma}^{-1}$ according to

$$\begin{aligned}
\mathbf{\Sigma} &= \begin{bmatrix} \mathbf{P}_{k|k}^{(o)} & \mathbf{P}_{k|k}^{(o)} \mathbf{K}_k^\top \\ \mathbf{K}_k \mathbf{P}_{k|k}^{(o)} & \mathbf{P}_{k|k-1}^{(u)} + \mathbf{K}_k \big(\mathbf{P}_{k|k}^{(o)} - \mathbf{P}_{k|k-1}^{(o)}\big) \mathbf{K}_k^\top \end{bmatrix} \\
&=: \begin{bmatrix} \mathbf{P}_{k|k}^{(o)} & \big(\mathbf{P}_{k|k}^{(u,o)}\big)^\top \\ \mathbf{P}_{k|k}^{(u,o)} & \mathbf{P}_{k|k}^{(u)} \end{bmatrix} \quad .
\end{aligned} \tag{C.3}$$

Finally, with (C.1), (C.2), and (C.3) we have

$$\begin{aligned}
f_{k|k-1}(\boldsymbol{x}_k^{(u)} \,|\, \boldsymbol{x}_k^{(o)}) \cdot f_{k|k}(\boldsymbol{x}_k^{(o)}) &\propto \exp\left(-\frac{1}{2} z\right) \\
&\propto \mathcal{N}\left(\begin{bmatrix} \boldsymbol{x}_k^{(o)} \\ \boldsymbol{x}_k^{(u)} \end{bmatrix} ; \begin{bmatrix} \hat{\boldsymbol{x}}_{k|k}^{(o)} \\ \hat{\boldsymbol{x}}_{k|k}^{(u)} \end{bmatrix} , \begin{bmatrix} \mathbf{P}_{k|k}^{(o)} & \big(\mathbf{P}_{k|k}^{(u,o)}\big)^\top \\ \mathbf{P}_{k|k}^{(u,o)} & \mathbf{P}_{k|k}^{(u)} \end{bmatrix}\right) \quad .
\end{aligned}$$

■

Appendix D

Proof of the Closed-Form Likelihood for Star-Convex RHMs

In this appendix, we derive a closed-form solution for the star-convex RHM likelihood (5.31), which is given by

$$f^{(\mathrm{sc})}(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) = \int_{\mathbb{R}} \mathcal{N}\left(\tilde{\boldsymbol{y}}_k - \boldsymbol{p}_k - s \cdot r(\boldsymbol{c}_k, \psi_k) \frac{\boldsymbol{d}_k}{\|\boldsymbol{d}_k\|_2}\,; \boldsymbol{0}, \mathbf{R}_k\right) \cdot \mathcal{N}(s\,; \hat{s}, \Sigma^{(s)})\,\mathrm{d}s \enspace .$$

First, by defining

$$\boldsymbol{b}_k := r(\boldsymbol{c}_k, \psi_k) \frac{\boldsymbol{d}_k}{\|\boldsymbol{d}_k\|_2}$$

and using the definition of $\boldsymbol{d}_k$, we get

$$f^{(\mathrm{sc})}(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) = \int_{\mathbb{R}} \frac{1}{2\pi\sqrt{|\mathbf{R}_k|}} \exp\left(-\frac{1}{2}(\boldsymbol{d}_k - s\cdot \boldsymbol{b}_k)^\top \mathbf{R}_k^{-1} (\boldsymbol{d}_k - s \cdot \boldsymbol{b}_k)\right) \cdot \mathcal{N}(s\,; \hat{s}, \Sigma^{(s)})\,\mathrm{d}s \enspace .$$

Second, with completing the square, we obtain

$$\begin{aligned} f^{(\mathrm{sc})}(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) = {} & \frac{1}{2\pi\sqrt{|\mathbf{R}_k|}} \exp\left(-\frac{1}{2}\left(\boldsymbol{d}_k^\top \mathbf{R}_k^{-1} \boldsymbol{d}_k - \frac{(\boldsymbol{d}_k^\top \mathbf{R}_k^{-1} \boldsymbol{b}_k)^2}{\boldsymbol{b}_k^\top \mathbf{R}_k^{-1} \boldsymbol{b}_k}\right)\right) \cdot \\ & \int_{\mathbb{R}} \exp\Big(-\frac{1}{2}\boldsymbol{b}_k^\top \mathbf{R}_k^{-1} \boldsymbol{b}_k \Big(s - \frac{\boldsymbol{d}_k^\top \mathbf{R}_k^{-1} \boldsymbol{b}_k}{\boldsymbol{b}_k^\top \mathbf{R}_k^{-1} \boldsymbol{b}_k}\Big)^2\Big) \cdot \mathcal{N}(s\,; \hat{s}, \Sigma^{(s)})\,\mathrm{d}s \enspace . \end{aligned}$$

Third, we define

$$\begin{aligned} q_k &:= \boldsymbol{d}_k^\top \mathbf{R}_k^{-1} \boldsymbol{d}_k \enspace , \\ t_k &:= \boldsymbol{d}_k^\top \mathbf{R}_k^{-1} \boldsymbol{b}_k \enspace , \\ u_k &:= \boldsymbol{b}_k^\top \mathbf{R}_k^{-1} \boldsymbol{b}_k \enspace , \\ z_k &:= \frac{1}{2\pi\sqrt{|\mathbf{R}_k|}} \exp\left(-\frac{1}{2}\left(q_k - \frac{t_k^2}{u_k}\right)\right) \enspace , \end{aligned}$$

and get a simplified expression according to

$$f^{(\mathrm{sc})}(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) = z_k \sqrt{\frac{2\pi}{u_k}} \cdot \int_{\mathbb{R}} \mathcal{N}\left(s\,; \frac{t_k}{u_k}, \frac{1}{u_k}\right) \cdot \mathcal{N}(s\,; \hat{s}, \Sigma^{(s)})\,\mathrm{d}s \enspace .$$

Finally, by exploiting that the product of two Gaussians also yields an unnormalized Gaussian and that a density function always integrates to one, we yield the closed-form likelihood function

$$f^{(\mathrm{sc})}(\tilde{\boldsymbol{y}}_k \,|\, \boldsymbol{x}_k) = \frac{z_k}{\sqrt{1 + u_k \Sigma^{(s)}}} \exp\left(-\frac{1}{2} \frac{\left(\hat{s} - \frac{t_k}{u_k} \right)^2}{\left(\frac{1}{u_k} + \Sigma^{(s)} \right)} \right) \quad .$$

■

Bibliography

[1] Mohinder S. Grewal and Angus P. Andrews, "Applications of Kalman Filtering in Aerospace 1960 to the Present," *IEEE Control Systems Magazine*, vol. 30, no. 3, pp. 69–78, Jun. 2010.

[2] Dimitri P. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed. Belmont, Mass.: Athena Scientific, 2000.

[3] Sebastian Thrun, Wolfram Burgard, and Dieter Fox, *Probabilistic Robotics*. Cambridge, London: MIT Press, 2005.

[4] Tine Lefebvre, Herman Bruyninckx, and Joris De Schutter, *Nonlinear Kalman Filtering for Force-Controlled Robot Tasks*, ser. Springer Tracts in Advanced Robotics. Berlin Heidelberg: Springer, 2005, vol. 19.

[5] Daniel Meissner, Stephan Reuter, Elias Strigel, and Klaus Dietmayer, "Intersection-Based Road User Tracking Using a Classifying Multiple-Model PHD Filter," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 21–33, Apr. 2014.

[6] Mark Wielitzka, Matthias Dagen, and Tobias Ortmaier, "State Estimation of Vehicle's Lateral Dynamics using Unscented Kalman Filter," in *Proceedings of the 2014 IEEE 53rd Annual Conference on Decision and Control (CDC)*, Los Angeles, USA, Dec. 2014, pp. 5015–5020.

[7] Mark Wielitzka, Matthias Dagen, and Tobias Ortmaier, "Comparison of Unscented Kalman Filter in General and Additive Formulation for State Estimation in Vehicle Dynamics," in *Proceedings of the 2016 American Control Conference (ACC 2016)*, Bosten, USA, Jul. 2016, pp. 6899–6904.

[8] Frank Moosmann and Christoph Stiller, "Joint Self-Localization and Tracking of Generic Objects in 3D Range Data," in *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013, pp. 1146–1152.

[9] Audrey Giremus, Arnaud Doucet, Vincent Calmettes, and Jean-Yves Tourneret, "A Rao-Blackwellized Particle Filter for INS/GPS Integration," in *Proceedings of the 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2004.

[10] Matthew Rhudy, Yu Gu, Jason Gross, and Marcello R. Napolitano, "Evaluation of Matrix Square Root Operations for UKF within a UAV GPS/INS Sensor Fusion Application," *International Journal of Navigation and Observation*, 2011.

[11] Yaakov Bar-Shalom, X. Rong Li, and Thiagalingam Kirubarajan, *Estimation with Applications to Tracking and Navigation*. New York Chichester Weinheim Brisbane Singapore Toronto: Wiley-Interscience, 2001.

[12] Branko Ristic, Sanjeev Arulampalam, and Neil Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House Publishers, 2004.

[13] Marcus Baum, "Simultaneous Tracking and Shape Estimation of Extended Objects," Ph.D. dissertation, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, 2013.

[14] Florian Faion, "Tracking Extended Objects in Noisy Point Clouds with Application in Telepresence Systems," Ph.D. dissertation, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, 2016.

[15] Rudolf E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," in *Transaction of the ASME - Journal of Basic Engineering*, Mar. 1960, pp. 35–45.

[16] Simon J. Julier and Jeffrey K. Uhlmann, "Unscented Filtering and Nonlinear Estimation," in *Proceedings of the IEEE*, vol. 92, Mar. 2004, pp. 401–422.

[17] Dan Simon, *Optimal State Estimation*, 1st ed. Wiley & Sons, 2006.

[18] Brendan M. Quine, "A Derivative-Free Implementation of the Extended Kalman Filter," *Automatica*, vol. 42, no. 11, pp. 1927–1934, Nov. 2006.

[19] Michael Athans, Richard P. Wishner, and Anthony Bertolini, "Suboptimal State Estimation for Continuous-Time Nonlinear Systems from Discrete Noisy Measurements," *IEEE Transactions on Automatic Control*, vol. 13, no. 5, pp. 504–514, Oct. 1968.

[20] Michael Roth and Fredrik Gustafsson, "An Efficient Implementation of the Second Order Extended Kalman Filter," in *Proceedings of the 14th International Conference on Information Fusion (Fusion 2011)*, Jul. 2011.

[21] Magnus Nørgaard, Niels K. Poulsen, and Ole Ravn, "New Developments in State Estimation for Nonlinear Systems," *Automatica*, vol. 36, no. 11, pp. 1627–1638, 2000.

[22] Kazufumi Ito and Kaiqi Xiong, "Gaussian Filters for Nonlinear Filtering Problems," *IEEE Transactions on Automatic Control*, vol. 45, no. 5, pp. 910–927, May 2000.

[23] Marco F. Huber, "Chebyshev Polynomial Kalman Filter," *Digital Signal Processing*, vol. 23, no. 5, pp. 1620–1629, Sep. 2013.

[24] Simon J. Julier and Jeffrey K. Uhlmann, "A New Extension of the Kalman Filter to Nonlinear Systems," in *11th Int. Symp. Aerospace/Defense Sensing, Simulation and Controls*, 1997, pp. 182–193.

[25] Simon J. Julier, "The Spherical Simplex Unscented Transformation," in *Proceedings of the 2003 American Control Conference (ACC 2003)*, Jun. 2003, pp. 2430–2434.

[26] Jindřich Duník, Miroslav Šimandl, and Ondřej Straka, "Unscented Kalman Filter: Aspects and Adaptive Setting of Scaling Parameter," *IEEE Transactions on Automatic Control*, vol. 57, no. 9, pp. 2411–2416, Sep. 2012.

[27] Ryan Turner and Carl Edward Rasmussen, "Model Based Learning of Sigma Points in Unscented Kalman Filtering," *Neurocomputing*, vol. 80, pp. 47–53, 2012.

[28] Ondřej Straka, Jindřich Duník, and Miroslav Šimandl, "Unscented Kalman Filter with Advanced Adaptation of Scaling Parameter," *Automatica*, vol. 50, no. 10, pp. 2657–2664, Oct. 2014.

[29] Jindřich Duník, Ondřej Straka, Miroslav Šimandl, and Erik Blasch, "Sigma-Point Set Rotation for Derivative-Free Filters in Target Tracking Applications," *Journal of Advances in Information Fusion*, vol. 11, no. 1, pp. 91–109, Jun. 2016.

[30] Henrique M. T. Menegaz, João Y. Ishihara, Geovany A. Borges, and Alessandro N. Vargas, "A Systematization of the Unscented Kalman Filter Theory," *IEEE Transactions on Automatic Control*, vol. 60, no. 10, pp. 2583–2598, Oct. 2015.

[31] Ienkaran Arasaratnam and Simon Haykin, "Cubature Kalman Filters," *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254–1269, Jun. 2009.

[32] Bin Jia, Ming Xin, and Yang Cheng, "High-Degree Cubature Kalman Filter," *Automatica*, vol. 49, no. 2, pp. 510–518, Feb. 2013.

[33] Jindřich Duník, Ondřej Straka, and Miroslav Šimandl, "The Development of a Randomised Unscented Kalman Filter," in *Proceedings of the 18th IFAC World Congress (IFAC 2011)*, Milano, Italy, Aug. 2011, pp. 8–13.

[34] Ondřej Straka, Jindřich Duník, and Miroslav Šimandl, "Randomized Unscented Kalman Filter in Target Tracking," in *Proceedings of the 15th International Conference on Information Fusion (Fusion 2012)*, Singapore, Jul. 2012, pp. 503–510.

[35] Ondřej Straka, Jindřich Duník, Miroslav Šimandl, and Erik Blasch, "Randomized Unscented Transform in State Estimation of non-Gaussian Systems: Algorithms and Performance," in *Proceedings of the 15th International Conference on Information Fusion (Fusion 2012)*, Singapore, Jul. 2012, pp. 2004–2011.

[36] Jindřich Duník, Ondřej Straka, and Miroslav Šimandl, "Stochastic Integration Filter," *IEEE Transactions on Automatic Control*, vol. 58, no. 6, pp. 1561–1566, Jun. 2013.

[37] Ondřej Straka, Jindřich Duník, Miroslav Šimandl, and Erik Blasch, "Comparison of Adaptive and Randomized Unscented Kalman Filter Algorithms," in *Proceedings of the 17th International Conference on Information Fusion (Fusion 2014)*, Salamanca, Spain, Jul. 2014.

[38] Renato Zanetti, "Recursive Update Filtering for Nonlinear Estimation," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1481–1490, Jun. 2012.

[39] Sridhar Ungarala, "On the Iterated Forms of Kalman Filters Using Statistical Linearization," *Journal of Process Control*, vol. 22, no. 5, pp. 935–943, Jun. 2012.

[40] Ángel F. García-Fernández, Lennart Svensson, and Mark Morelande, "Iterated Statistical Linear Regression for Bayesian Updates," in *Proceedings of the 17th International Conference on Information Fusion (Fusion 2014)*, Salamanca, Spain, Jul. 2014.

[41] Ángel F. García-Fernández, Lennart Svensson, Mark Morelande, and Simo Särkkä, "Posterior Linearisation Filter: Principles and Implementation Using Sigma Points," *IEEE Transactions on Signal Processing*, vol. 63, no. 20, pp. 5561–5573, Oct. 2015.

[42] Rudolph van der Merwe and Eric A. Wan, "The Square-Root Unscented Kalman Filter for State and Parameter-Estimation," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)*, vol. 6, Salt Lake City, USA, May 2001, pp. 3461–3464.

[43] Daniel L. Alspach and Harold W. Sorenson, "Nonlinear Bayesian Estimation Using Gaussian Sum Approximations," *IEEE Transactions on Automatic Control*, vol. 17, no. 4, pp. 439–448, Aug. 1972.

[44] Miroslav Šimandl and Jindřich Duník, "Sigma Point Gaussian Sum Filter Design Using Square Root Unscented Filters," in *Proceedings of the 16th IFAC World Congress (IFAC 2005)*, Czech Republic, 2005.

[45] Friedrich Faubel, John McDonough, and Dietrich Klakow, “The Split and Merge Unscented Gaussian Mixture Filter,” *IEEE Signal Processing Letters*, vol. 16, no. 9, pp. 786–789, Sep. 2009.

[46] Simon J. Julier and Joseph J. LaViola, “On Kalman Filtering with Nonlinear Equality Constraints,” *IEEE Transactions on Signal Processing*, vol. 55, no. 6, pp. 2774–2784, Jun. 2007.

[47] Ondřej Straka, Jindřich Duník, and Miroslav Šimandl, “Truncation Nonlinear Filters for State Estimation with Nonlinear Inequality Constraints,” *Automatica*, vol. 48, no. 2, pp. 273–286, Feb. 2012.

[48] Ondřej Straka, Jindřich Duník, Miroslav Šimandl, and Jindřich Havlík, “Truncated Randomized Unscented Kalman Filter for Interval Constrained State Estimation,” in *Proceedings of the 16th International Conference on Information Fusion (Fusion 2013)*, Istanbul, Turkey, Jul. 2013, pp. 2081–2088.

[49] Dan Simon, “Kalman Filtering with State Constraints: A Survey of Linear and Nonlinear Algorithms,” *IET Control Theory & Applications*, vol. 4, no. 8, pp. 1303–1318, Aug. 2010.

[50] Frederik Beutler, Marco F. Huber, and Uwe D. Hanebeck, “Gaussian Filtering using State Decomposition Methods,” in *Proceedings of the 12th International Conference on Information Fusion (Fusion 2009)*, Seattle, USA, Jul. 2009.

[51] Frederik Beutler, Marco F. Huber, and Uwe D. Hanebeck, “Semi-Analytic Stochastic Linearization for Range-Based Pose Tracking,” in *Proceedings of the 2010 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2010)*, Salt Lake City, USA, Sep. 2010, pp. 44–49.

[52] Marco F. Huber, Frederik Beutler, and Uwe D. Hanebeck, “Semi-Analytic Gaussian Assumed Density Filter,” in *Proceedings of the 2011 American Control Conference (ACC 2011)*, San Francisco, USA, Jun. 2011.

[53] Marco F. Huber, Frederik Beutler, and Uwe D. Hanebeck, “(Semi-)Analytic Gaussian Mixture Filter,” in *Proceedings of the 18th IFAC World Congress (IFAC 2011)*, Milano, Italy, Aug. 2011.

[54] Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp, “A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking,” *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[55] Arnaud Doucet and Adam M. Johansen, “A Tutorial on Particle Filtering and Smoothing: Fifteen Years Later,” in *Oxford Handbook of Nonlinear Filtering*, 2011, pp. 656–704.

[56] George Casella and Christian P. Robert, “Rao-Blackwellisation of Sampling Schemes,” *Biometrika*, vol. 81, no. 1, pp. 81–94, 1996.

[57] Thomas Schön, Fredrik Gustafsson, and Per-Johan Nordlund, “Marginalized Particle Filters for Mixed Linear/Nonlinear State-Space Models,” *IEEE Transactions on Signal Processing*, vol. 53, no. 7, pp. 2279–2289, Jul. 2005.

[58] Jayesh H. Kotecha and Petar M. Djuric, “Gaussian Particle Filtering,” *IEEE Transactions on Signal Processing*, vol. 51, no. 10, pp. 2592–2601, Oct. 2003.

[59] Jayesh H. Kotecha and Petar M. Djuric, “Gaussian Sum Particle Filtering,” *IEEE Transactions on Signal Processing*, vol. 51, no. 10, pp. 2602–2612, Oct. 2003.

[60] Geir Evensen, “Sequential Data Assimilation with a Nonlinear Quasi-Geostrophic Model Using Monte Carlo Methods to Forecast Error Statistics,” *Journal of Geophysical Research: Oceans*, vol. 99, no. C5, pp. 10 143–10 162, May 1994.

[61] Gerrit Burgers, Peter Jan van Leeuwen, and Geir Evensen, “Analysis Scheme in the Ensemble Kalman Filter,” *Monthly Weather Review*, vol. 126, no. 6, pp. 1719–1724, Jun. 1998.

[62] Geir Evensen, “The Ensemble Kalman Filter: Theoretical Formulation and Practical Implementation,” *Ocean Dynamics*, vol. 53, no. 4, pp. 343–367, Nov. 2003.

[63] S. Gillijns, O. Barrero Mendoza, J. Chandrasekar, B. L. R. De Moor, D. S. Bernstein, and A. Ridley, “What Is the Ensemble Kalman Filter and How Well Does It Work?” in *Proceedings of the 2006 American Control Conference (ACC 2006)*, Minneapolis, USA, Jun. 2006, pp. 4448–4453.

[64] J. Prakash, Sachin C. Patwardhan, and Sirish L. Shah, “Constrained State Estimation Using the Ensemble Kalman Filter,” in *Proceedings of the 2008 American Control Conference (ACC 2008)*, Seattle, USA, Jun. 2008, pp. 3542–3547.

[65] N. Oudjane and C. Musso, “Progressive Correction for Regularized Particle Filters,” in *Proceedings of the 3rd International Conference on Information Fusion (Fusion 2000)*, Paris, France, Jul. 2000.

[66] Patrick Ruoff, Peter Krauthausen, and Uwe D. Hanebeck, “Progressive Correction for Deterministic Dirac Mixture Approximations,” in *Proceedings of the 14th International Conference on Information Fusion (Fusion 2011)*, Chicago, USA, Jul. 2011.

[67] Uwe D. Hanebeck, “PGF 42: Progressive Gaussian Filtering with a Twist,” in *Proceedings of the 16th International Conference on Information Fusion (Fusion 2013)*, Istanbul, Turkey, Jul. 2013.

[68] Arthur G. O. Mutambara, *Decentralized Estimation and Control for Multisensor Systems*. Boca Raton, USA: CRC Press, Inc., 1998.

[69] Wolfgang Koch, “On Optimal Distributed Kalman Filtering and Retrodiction at Arbitrary Communication Rates for Maneuvering Targets,” in *Proceedings of the 2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2008)*, Seoul, South Korea, Aug. 2008, pp. 457–462.

[70] Wolfgang Koch, “Exact Update Formulae for Distributed Kalman Filtering and Retrodiction at Arbitrary Communication Rates,” in *Proceedings of the 12th International Conference on Information Fusion (Fusion 2009)*, Seattle, USA, Jul. 2009, pp. 2209–2216.

[71] Felix Govaers and Wolfgang Koch, “Distributed Kalman Filter Fusion at Arbitrary Instants of Time,” in *Proceedings of the 13th International Conference on Information Fusion (Fusion 2010)*, Edinburgh, United Kingdom, Jul. 2010.

[72] Felix Govaers and Wolfgang Koch, “On the Globalized Likelihood Function for Exact Track-To-Track Fusion at Arbitrary Instants of Time,” in *Proceedings of the 14th International Conference on Information Fusion (Fusion 2011)*, Chicago, USA, Jul. 2011.

[73] Yaakov Bar-Shalom and Leon Campo, “The Effect of the Common Process Noise on the Two-Sensor Fused-Track Covariance,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-22, no. 6, pp. 803–805, Nov. 1986.

[74] Neal A. Carlson, “Federated Square Root Filter for Decentralized Parallel Processors,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 26, no. 3, pp. 517–525, May 1990.

[75] Simon J. Julier and Jeffrey K. Uhlmann, “A Non-divergent Estimation Algorithm in the Presence of Unknown Correlations,” in *Proceedings of the 1997 American Control Conference (ACC 1997)*, Albuquerque, USA, Jun. 1997, pp. 2369–2373.

[76] Marc Reinhardt, Benjamin Noack, and Uwe D. Hanebeck, “Reconstruction of Joint Covariance Matrices in Networked Linear Systems,” in *Proceedings of the 48th Annual Conference on Information Sciences and Systems (CISS 2014)*, Princeton, USA, Mar. 2014.

[77] Tom Vercauteren and Xiaodong Wang, “Decentralized Sigma-Point Information Filters for Target Tracking in Collaborative Sensor Networks,” *IEEE Transactions on Signal Processing*, vol. 53, no. 8, pp. 2997–3009, Aug. 2005.

[78] Kuo-Chu Chang, Rajat K. Saha, and Yaakov Bar-Shalom, “On Optimal Track-to-Track Fusion,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 4, pp. 1271–1276, Oct. 1997.

[79] K. Gilholm and D. Salmond, “Spatial Distribution Model for Tracking Extended Objects,” *IEE Proceedings Radar, Sonar and Navigation*, vol. 152, no. 5, pp. 364–371, Oct. 2005.

[80] Florian Faion, Antonio Zea, Marcus Baum, and Uwe D. Hanebeck, “Bayesian Estimation of Line Segments,” in *Proceedings of the IEEE ISIF Workshop on Sensor Data Fusion: Trends, Solutions, Applications (SDF 2014)*, Bonn, Germany, Oct. 2014.

[81] Johann Wolfgang Koch, “Bayesian Approach to Extended Object and Cluster Tracking Using Random Matrices,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, no. 3, pp. 1042–1059, Jul. 2008.

[82] Michael Feldmann, Dietrich Fränken, and Wolfgang Koch, “Tracking of Extended Objects and Group Targets Using Random Matrices,” *IEEE Transactions on Signal Processing*, vol. 59, no. 4, pp. 1409–1420, Apr. 2011.

[83] Marcus Baum, Benjamin Noack, and Uwe D. Hanebeck, “Extended Object and Group Tracking with Elliptic Random Hypersurface Models,” in *Proceedings of the 13th International Conference on Information Fusion (Fusion 2010)*, Edinburgh, United Kingdom, Jul. 2010.

[84] Marcus Baum and Uwe D. Hanebeck, “Shape Tracking of Extended Objects and Group Targets with Star-Convex RHMs,” in *Proceedings of the 14th International Conference on Information Fusion (Fusion 2011)*, Chicago, USA, Jul. 2011.

[85] Marcus Baum, Florian Faion, and Uwe D. Hanebeck, “Modeling the Target Extent with Multiplicative Noise,” in *Proceedings of the 15th International Conference on Information Fusion (Fusion 2012)*, Singapore, Jul. 2012, pp. 2406–2412.

[86] Florian Faion, Marcus Baum, and Uwe D. Hanebeck, “Tracking 3D Shapes in Noisy Point Clouds with Random Hypersurface Models,” in *Proceedings of the 15th International Conference on Information Fusion (Fusion 2012)*, Singapore, Jul. 2012.

[87] Antonio Zea, Florian Faion, Marcus Baum, and Uwe D. Hanebeck, “Level-Set Random Hypersurface Models for Tracking Non-Convex Extended Objects,” in *Proceedings of the 16th International Conference on Information Fusion (Fusion 2013)*, Istanbul, Turkey, Jul. 2013.

[88] Marcus Baum and Uwe D. Hanebeck, “Extended Object Tracking with Random Hypersurface Models,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 1, pp. 149–159, Jan. 2014.

[89] Antonio Zea, Florian Faion, Marcus Baum, and Uwe D. Hanebeck, "Level-set Random Hypersurface Models for Tracking Nonconvex Extended Objects," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 6, pp. 2990–3007, Dec. 2016.

[90] Florian Faion, Antonio Zea, Marcus Baum, and Uwe D. Hanebeck, "Partial Likelihood for Unbiased Extended Object Tracking," in *Proceedings of the 18th International Conference on Information Fusion (Fusion 2015)*, Washington D. C., USA, Jul. 2015.

[91] Antonio Zea, Florian Faion, and Uwe D. Hanebeck, "Shape Tracking using Partial Information Models," in *Proceedings of the 2015 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2015)*, San Diego, USA, Sep. 2015.

[92] Henk A. P. Blom and Yaakov Bar-Shalom, "The Interacting Multiple Model Algorithm for Systems with Markovian Switching Coefficients," *IEEE Transactions on Automatic Control*, vol. 33, no. 8, pp. 780–783, Aug. 1988.

[93] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, "Interacting Multiple Model Methods in Target Tracking: A Survey," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 1, pp. 103–123, Jan. 1998.

[94] X. Rong Li and Yaakov Bar-Shalom, "A Recursive Multiple Model Approach to Noise Identification," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 30, no. 3, pp. 671–684, Jul. 1994.

[95] Samuel Blackman and Robert Popoli, *Design and Analysis of Modern Tracking Systems*. Artech House Publishers, Jul. 1999.

[96] Yaakov Bar-Shalom, Fred Daum, and Jim Huang, "The Probabilistic Data Association Filter," *IEEE Control Systems Magazine*, vol. 29, no. 6, pp. 82–100, Dec. 2009.

[97] Ronald P. S. Mahler, "Multitarget Bayes Filtering via First-order Multitarget Moments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1152–1178, Oct. 2003.

[98] Ba-Ngu Vo and Wing-Kin Ma, "The Gaussian Mixture Probability Hypothesis Density Filter," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4091–4104, Nov. 2006.

[99] Ronald P. S. Mahler, "PHD Filters of Higher Order in Target Number," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 4, pp. 1523–1543, Oct. 2007.

[100] Ba-Tuong Vo, Ba-Ngu Vo, and Antonio Cantoni, "The Cardinality Balanced Multi-Target Multi-Bernoulli Filter and Its Implementations," *IEEE Transactions on Signal Processing*, vol. 57, no. 2, pp. 409–423, Feb. 2009.

[101] Stephan Reuter, Ba-Tuong Vo, Ba-Ngu Vo, and Klaus Dietmayer, "The Labeled Multi-Bernoulli Filter," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3246–3260, Jun. 2014.

[102] Ba-Ngu Vo, Ba-Tuong Vo, and Hung Gia Hoang, "An Efficient Implementation of the Generalized Labeled Multi-Bernoulli Filter," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 1975–1986, Apr. 2017.

[103] Steven W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. San Diego, USA: California Technical Publishing, 1997.

[104] Pawe Stano, Zsófia Lendek, Jelmer Braaksma, Robert Babuska, Cees de Keizer, and Arnold J. den Dekker, "Parametric Bayesian Filters for Nonlinear Stochastic Dynamical Systems: A Survey," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1607–1624, Dec. 2013.

[105] Allan Gut, *An Intermediate Course in Probability*, 2nd ed. New York: Springer, 2009.

[106] Athanasios Papoulis and S. Unnikrishna Pillai, *Probability, Random Variables and Stochastic Processes*, 4th ed. McGraw-Hill, 2002.

[107] Wolfgang Härdle and Léopold Simar, *Applied Multivariate Statistical Analysis*, 2nd ed. Berlin Heidelberg: Springer, 2008.

[108] Ondřej Straka, Jindřich Duník, Miroslav Šimandl, and Jindřich Havlík, "Aspects and Comparison of Matrix Decompositions in Unscented Kalman Filter," in *Proceedings of the 2013 American Control Conference (ACC 2013)*, Washington D. C., USA, Jun. 2013, pp. 3081–3086.

[109] Arthur H. Stroud, *Approximate Calculation of Multiple Integrals*. Englewood Cliffs, NJ: Prentice-Hall, 1971.

[110] Zhong Wang and Yan Li, "Cross Approximation-based Quadrature Filter," in *Proceedings of the 2016 IEEE 55th Annual Conference on Decision and Control (CDC)*, Las Vegas, USA, Dec. 2016, pp. 2023–2028.

[111] Uwe D. Hanebeck, Marco F. Huber, and Vesa Klumpp, "Dirac Mixture Approximation of Multivariate Gaussian Densities," in *Proceedings of the 2009 IEEE Conference on Decision and Control (CDC 2009)*, Shanghai, China, Dec. 2009.

[112] Igor Gilitschenski and Uwe D. Hanebeck, "Efficient Deterministic Dirac Mixture Approximation of Gaussian Distributions," in *Proceedings of the 2013 American Control Conference (ACC 2013)*, Washington D. C., USA, Jun. 2013.

[113] Milton Abramowitz and Irene A. Stegun, *Handbock of Mathematical Functions*, 9th ed. New York: Dover Publ., 1970.

[114] R. Piessens, E. de Doncker-Kapenga, C.W. Überhuber, and D.K. Kahaner, *QUADPACK: A Subroutine Package for Automatic Integration*, 1st ed., ser. Springer Series in Computational Mathematics. Berlin Heidelberg: Springer, 1983.

[115] Jorge Nocedal and Stephen J. Wright, *Numerical Optimization*, 2nd ed., ser. Springer Series in Operations Research and Financial Engineering. Springer, 2006.

[116] Raymond Kan, "From Moments of Sum to Moments of Product," *Journal of Multivariate Analysis*, vol. 99, no. 3, pp. 542–554, Mar. 2008.

[117] Mark Morelande and Ángel F. García-Fernández, "Analysis of Kalman Filter Approximations for Nonlinear Measurements," *IEEE Transactions on Signal Processing*, vol. 61, no. 22, pp. 5477–5484, Nov. 2013.

[118] Michael Roth, Gustaf Hendeby, and Fredrik Gustafsson, "Nonlinear Kalman Filters Explained: A Tutorial on Moment Computations and Sigma Point Methods," *Journal of Advances in Information Fusion*, vol. 11, no. 1, pp. 47–70, Jun. 2016.

[119] Rudolph van der Merwe, "Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models," Ph.D. dissertation, OGI School of Science & Engineering, Oregon Health & Science University, Portland, USA, Apr. 2004.

[120] Matti Raitoharju, Robert Piché, Juha Ala-Luhtala, and Simo Ali-Löytty, "Partitioned Update Kalman Filter," *Journal of Advances in Information Fusion*, vol. 11, no. 1, pp. 3–14, Jun. 2016.

[121] Tine Lefebvre, Herman Bruyninckx, and Joris De Schutter, "Kalman Filters for Non-Linear Systems: A Comparison of Performance," *International Journal of Control*, vol. 77, no. 7, pp. 639–653, May 2004.

[122] Dirk Tenne and Tarunraj Singh, “The Higher Order Unscented Filter,” in *Proceedings of the 2003 American Control Conference (ACC 2003)*, vol. 3, Jun. 2003, pp. 2441–2446.

[123] Bin Jia and Ming Xin, “Adaptive Radial Rule Based Cubature Kalman Filter,” in *Proceedings of the 2015 American Control Conference (ACC 2015)*, Chicago, USA, Jul. 2015, pp. 3156–3161.

[124] Marco F. Huber and Uwe D. Hanebeck, “Gaussian Filter Based on Deterministic Sampling for High Quality Nonlinear Estimation,” in *Proceedings of the 17th IFAC World Congress (IFAC 2008)*, vol. 17, Seoul, South Korea, Jul. 2008.

[125] Filip Tronarp, Roland Hostettler, and Simo Särkkä, “Sigma-Point Filtering for Nonlinear Systems with Non-Additive Heavy-Tailed Noise,” in *Proceedings of the 19th International Conference on Information Fusion (Fusion 2016)*, Heidelberg, Germany, Jul. 2016.

[126] Yulong Huang, Yonggang Zhang, Ning Li, and Lin Zhao, “Design of Sigma-Point Kalman Filter with Recursive Updated Measurement,” *Circuits, Systems, and Signal Processing*, pp. 1–16, Aug. 2015.

[127] Guoliang Liu, Florentin Wörgötter, and Irene Markelić, “Square-Root Sigma-Point Information Filtering,” *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2945–2950, Nov. 2012.

[128] Ienkaran Arasaratnam, “Sensor Fusion with Square-Root Cubature Information Filtering,” *Intelligent Control and Automation*, vol. 4, no. 1, 2013.

[129] Kumar Pakki Bharani Chandra, Da-Wei Gu, and Ian Postlethwaite, “Square Root Cubature Information Filter,” *IEEE Sensors Journal*, vol. 13, no. 2, pp. 750–758, Feb. 2013.

[130] Yulong Huang, Yonggang Zhang, Ning Li, and Lin Zhao, “Improved Square-Root Cubature Information Filter,” *Transactions of the Institute of Measurement and Control*, Oct. 2015.

[131] Ondřej Straka, Jindřich Duník, and Miroslav Šimandl, “Gaussian Sum Unscented Kalman Filter with Adaptive Scaling Parameters,” in *Proceedings of the 14th International Conference on Information Fusion (Fusion 2011)*, Chicago, USA, Jul. 2011.

[132] Marco F. Huber, “Adaptive Gaussian Mixture Filter Based on Statistical Linearization,” in *Proceedings of the 14th International Conference on Information Fusion (Fusion 2011)*, Chicago, USA, Jul. 2011.

[133] Gabriel Terejanu, Puneet Singla, Tarunraj Singh, and Peter D. Scott, “Adaptive Gaussian Sum Filter for Nonlinear Bayesian Estimation,” *IEEE Transactions on Automatic Control*, vol. 56, no. 9, pp. 2151–2156, Sep. 2011.

[134] X. Rong Li and Yu Liu, “Generalized Linear Minimum Mean-Square Error Estimation,” in *Proceedings of the 16th International Conference on Information Fusion (Fusion 2013)*, Istanbul, Turkey, Jul. 2013, pp. 1819–1826.

[135] Yu Liu, X. Rong Li, and Huimin Chen, “Linear Estimation with Transformed Measurement for Nonlinear Estimation,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 1, pp. 221–236, Feb. 2016.

[136] Marcus Baum and Uwe D. Hanebeck, “The Kernel-SME Filter for Multiple Target Tracking,” in *Proceedings of the 16th International Conference on Information Fusion (Fusion 2013)*, Istanbul, Turkey, Jul. 2013.

[137] Christof Chlebek and Uwe D. Hanebeck, “Bayesian Approach to Direct Pole Estimation,” in *Proceedings of the 2014 European Control Conference (ECC 2014)*, Strasbourg, France, Jun. 2014.

[138] Florian Faion, Antonio Zea, and Uwe D. Hanebeck, “Reducing Bias in Bayesian Shape Estimation,” in *Proceedings of the 17th International Conference on Information Fusion (Fusion 2014)*, Salamanca, Spain, Jul. 2014.

[139] Matti Raitoharju, Robert Piché, and Henri Nurminen, “A Systematic Approach for Kalman-Type Filtering with Non-Gaussian Noises,” in *Proceedings of the 19th International Conference on Information Fusion (Fusion 2016)*, Heidelberg, Germany, Jul. 2016.

[140] Antonio Zea, Florian Faion, Marcus Baum, and Uwe D. Hanebeck, “Tracking Simplified Shapes Using a Stochastic Boundary,” in *Proceedings of the Eighth IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM 2014)*, A Coruña, Spain, Jun. 2014.

[141] Cihan Ulas and Hakan Temeltas, “Planar Features and 6d-SLAM Based on Linear Regression Kalman Filters with N-Dimensional Approximated Gaussians,” in *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, Aug. 2014, pp. 10 194–10 199.

[142] Joris Sijs and Mircea Lazar, “State-Fusion with Unknown Correlation: Ellipsoidal Intersection,” *Automatica*, vol. 48, no. 8, pp. 1874–1878, Aug. 2012.

[143] Marc Reinhardt, Benjamin Noack, and Uwe D. Hanebeck, “The Hypothesizing Distributed Kalman Filter,” in *Proceedings of the 2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2012)*, Hamburg, Germany, Sep. 2012.

[144] Marc Reinhardt, Benjamin Noack, and Uwe D. Hanebeck, “Advances in Hypothesizing Distributed Kalman Filtering,” in *Proceedings of the 16th International Conference on Information Fusion (Fusion 2013)*, Istanbul, Turkey, Jul. 2013.

[145] Fahed Abdallah, Amadou Gning, and Philippe Bonnifait, “Box Particle Filtering for Nonlinear State Estimation Using Interval Analysis,” *Automatica*, vol. 44, no. 3, pp. 807–815, Mar. 2008.

[146] Amadou Gning, Branko Ristic, Lyudmila Mihaylova, and Fahed Abdallah, “An Introduction to Box Particle Filtering,” *IEEE Signal Processing Magazine*, vol. 30, no. 4, pp. 166–171, Jul. 2013.

[147] Fabien Campillo and Vivien Rossi, “Convolution Particle Filter for Parameter Estimation in General State-Space Models,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 45, no. 3, Jul. 2009.

[148] Donka Angelova, Lyudmila Mihaylova, Nikolay Petrov, and Amadou Gning, “A Convolution Particle Filtering Approach for Tracking Elliptical Extended Objects,” in *Proceedings of the 16th International Conference on Information Fusion (Fusion 2013)*, Istanbul, Turkey, Jul. 2013.

[149] Gerhard Kurz, Igor Gilitschenski, and Uwe D. Hanebeck, “Nonlinear Measurement Update for Estimation of Angular Systems Based on Circular Distributions,” in *Proceedings of the 2014 American Control Conference (ACC 2014)*, Portland, USA, Jun. 2014.

[150] Niklas Wahlström and Emre Özkan, “Extended Target Tracking Using Gaussian Processes,” *IEEE Transactions on Signal Processing*, vol. 63, no. 16, pp. 4165–4178, Aug. 2015.

[151] Karl Granström and Marcus Baum, “Extended Object Tracking: Introduction, Overview and Applications,” *arXiv preprint: Computer Vision and Pattern Recognition (cs.CV)*, Mar. 2016.

[152] OpenMP Architecture Review Board, “OpenMP.” [Online]. Available: http://openmp.org/

[153] Khronos Group, “Open Computing Language (OpenCL).” [Online]. Available: https://www.khronos.org/opencl/

[154] NVIDIA Corporation, "CUDA Parallel Computing Platform." [Online]. Available: http://www.nvidia.com/object/cuda_home_new.html

[155] NVIDIA Corporation, "NVIDIA CUDA Basic Linear Algebra Subroutines (cuBLAS)." [Online]. Available: https://developer.nvidia.com/cublas

[156] NVIDIA Corporation, "NVIDIA CUDA Deep Neural Network Library (cuDNN)." [Online]. Available: https://developer.nvidia.com/cudnn

[157] "Open Source Computer Vision Library (OpenCV)." [Online]. Available: http://opencv.org/

[158] Sebastian Bodenstedt, Martin Wagner, Benjamin Mayer, Katherine Stemmer, Hannes Kenngott, Beat Müller-Stich, Rüdiger Dillmann, and Stefanie Speidel, "Image-Based Laparoscopic Bowel Measurement," *International Journal of Computer Assisted Radiology and Surgery*, vol. 11, no. 3, pp. 407–419, Mar. 2016.

[159] Gustaf Hendeby, Rickard Karlsson, and Fredrik Gustafsson, "Particle Filtering: The Need for Speed," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, Feb. 2010.

[160] Matthew A. Goodrum, Michael J. Trotter, Alla Aksel, Scott T. Acton, and Kevin Skadron, "Parallelization of Particle Filter Algorithms," in *Computer Architecture*, ser. Lecture Notes in Computer Science Volume 6161, 2012, pp. 139–149.

[161] NVIDIA Corporation, "NVIDIA CUDA Random Number Generation Library (cuRAND)." [Online]. Available: https://developer.nvidia.com/curand

[162] Licong Zhang, Jürgen Sturm, Daniel Cremers, and Dongheui Lee, "Real-Time Human Motion Tracking Using Multiple Depth Cameras," in *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, Vilamoura, Portugal, Oct. 2012, pp. 2389–2395.

[163] Florian Faion, Antonio Zea, Marcus Baum, and Uwe D. Hanebeck, "Symmetries in Bayesian Extended Object Tracking," *Journal of Advances in Information Fusion*, Jun. 2015.

[164] Benjamin Sapp, Chris Jordan, and Ben Taskar, "Adaptive Pose Priors for Pictorial Structures," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, USA, Jun. 2010, pp. 422–429.

[165] Florian Faion, Simon Friedberger, Antonio Zea, and Uwe D. Hanebeck, "Intelligent Sensor-Scheduling for Multi-Kinect-Tracking," in *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, Vilamoura, Algarve, Portugal, Oct. 2012, pp. 3993–3999.

[166] Benoît Jacob, Gaël Guennebaud, and others, "Eigen C++ Linear Algebra Library." [Online]. Available: http://eigen.tuxfamily.org/

[167] Florian Faion, Maxim Dolgov, Antonio Zea, and Uwe D. Hanebeck, "Closed-Form Bias Reduction for Shape Estimation with Polygon Models," in *Proceedings of the 19th International Conference on Information Fusion (Fusion 2016)*, Heidelberg, Germany, Jul. 2016.

[168] Antonio Zea, Florian Faion, and Uwe D. Hanebeck, "Tracking Elongated Extended Objects Using Splines," in *Proceedings of the 19th International Conference on Information Fusion (Fusion 2016)*, Heidelberg, Germany, Jul. 2016.

[169] Antonio Zea, Florian Faion, and Uwe D. Hanebeck, "Exploiting Clutter: Negative Information for Enhanced Extended Object Tracking," in *Proceedings of the 18th International Conference on Information Fusion (Fusion 2015)*, Washington D. C., USA, Jul. 2015.

[170] Yulong Huang, Yonggang Zhang, Ning Li, and Lin Zhao, “Gaussian Approximate Filter with Progressive Measurement Update,” in *Proceedings of the 2015 IEEE 54th Annual Conference on Decision and Control (CDC)*, Osaka, Japan, Dec. 2015, pp. 4344–4349.

[171] Lifan Sun, Jian Lan, and X. Rong Li, “Extended Target Tracking Using Star-Convex Model with Nonlinear Inequality Constraints,” in *Proceedings of the 31st Chinese Control Conference*, Hefei, China, Jul. 2012.

[172] Shozo Mori, Kuo-Chu Chang, and Chee-Yee Chong, “Essence of Distributed Target Tracking,” in *Distributed Data Fusion for Network-Centric Operations*, ser. The Electrical Engineering and Applied Signal Processing Series. CRC Press, 2013, pp. 125–160.

[173] Roger A. Horn and Charles R. Johnson, *Matrix Analysis*, 20th ed. New York, USA: Cambridge University Press, 2006.

Own Publications

[174] Florian Faion, Antonio Zea, Jannik Steinbring, Marcus Baum, and Uwe D. Hanebeck, "Recursive Bayesian Pose and Shape Estimation of 3D Objects Using Transformed Plane Curves," in *Proceedings of the IEEE ISIF Workshop on Sensor Data Fusion: Trends, Solutions, Applications (SDF 2015)*, Bonn, Germany, Oct. 2015.

[175] Jannik Steinbring, Martin Pander, and Uwe D. Hanebeck, "The Smart Sampling Kalman Filter with Symmetric Samples," *Journal of Advances in Information Fusion*, vol. 11, no. 1, pp. 71–90, Jun. 2016.

[176] Jannik Steinbring, "Nonlinear Estimation Toolbox." [Online]. Available: https://bitbucket.org/nonlinearestimation/toolbox

[177] Igor Gilitschenski, Jannik Steinbring, Uwe D. Hanebeck, and Miroslav Šimandl, "Deterministic Dirac Mixture Approximation of Gaussian Mixtures," in *Proceedings of the 17th International Conference on Information Fusion (Fusion 2014)*, Salamanca, Spain, Jul. 2014.

[178] Jannik Steinbring and Uwe D. Hanebeck, "S^{2}KF: The Smart Sampling Kalman Filter," in *Proceedings of the 16th International Conference on Information Fusion (Fusion 2013)*, Istanbul, Turkey, Jul. 2013, pp. 2089–2096.

[179] Jannik Steinbring and Uwe D. Hanebeck, "LRKF Revisited: The Smart Sampling Kalman Filter (S^{2}KF)," *Journal of Advances in Information Fusion*, vol. 9, no. 2, pp. 106–123, Dec. 2014.

[180] Antonio Zea, Florian Faion, Jannik Steinbring, and Uwe D. Hanebeck, "Exploiting Negative Measurements for Tracking Star-Convex Extended Objects," in *Proceedings of the 2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2016)*, Baden-Baden, Germany, Sep. 2016.

[181] Jannik Steinbring, Christian Mandery, Florian Pfaff, Florian Faion, Tamim Asfour, and Uwe D. Hanebeck, "Real-Time Whole-Body Human Motion Tracking Based on Unlabeled Markers," in *Proceedings of the 2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2016)*, Baden-Baden, Germany, Sep. 2016.

[182] Jannik Steinbring, Benjamin Noack, Marc Reinhardt, and Uwe D. Hanebeck, "Optimal Sample-Based Fusion for Distributed State Estimation," in *Proceedings of the 19th International Conference on Information Fusion (Fusion 2016)*, Heidelberg, Germany, Jul. 2016.

[183] Jannik Steinbring and Uwe D. Hanebeck, "Progressive Gaussian Filtering Using Explicit Likelihoods," in *Proceedings of the 17th International Conference on Information Fusion (Fusion 2014)*, Salamanca, Spain, Jul. 2014.

[184] Jannik Steinbring and Uwe D. Hanebeck, "GPU-Accelerated Progressive Gaussian Filtering with Applications to Extended Object Tracking," in *Proceedings of the 18th International*

Conference on Information Fusion (Fusion 2015), Washington D. C., USA, Jul. 2015, pp. 1038–1045.

[185] Jannik Steinbring, Marcus Baum, Antonio Zea, Florian Faion, and Uwe D. Hanebeck, "A Closed-Form Likelihood for Particle Filters to Track Extended Objects with Star-Convex RHMs," in *Proceedings of the 2015 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2015)*, San Diego, USA, Sep. 2015, pp. 25–30.

[186] Jannik Steinbring, Antonio Zea, and Uwe D. Hanebeck, "Semi-Analytic Progressive Gaussian Filtering," in *Proceedings of the 2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2016)*, Baden-Baden, Germany, Sep. 2016.

[187] Uwe D. Hanebeck and Jannik Steinbring, "Progressive Gaussian Filtering Based on Dirac Mixture Approximations," in *Proceedings of the 15th International Conference on Information Fusion (Fusion 2012)*, Singapore, Jul. 2012, pp. 1697–1704.

[188] Christof Chlebek, Jannik Steinbring, and Uwe D. Hanebeck, "Progressive Gaussian Filter Using Importance Sampling and Particle Flow," in *Proceedings of the 19th International Conference on Information Fusion (Fusion 2016)*, Heidelberg, Germany, Jul. 2016.

[189] Florian Faion, Antonio Zea, Benjamin Noack, Jannik Steinbring, and Uwe D. Hanebeck, "Camera- and IMU-based Pose Tracking for Augmented Reality," in *Proceedings of the 2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2016)*, Baden-Baden, Germany, Sep. 2016.

Supervised Student Theses

[190] Martin Pander, “Symmetric Dirac Mixture Approximation of Multivariate Standard Normal Distributions Using Nonlinear, Constrained Optimization,” Diploma thesis, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, Aug. 2014.

[191] Florian Rosenthal, “Nonlinear Kalman Filtering with Applications to Extended Object Tracking,” Master thesis, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, May 2016.