Analyse der Fehlertoleranz eines Unscented Kalman Filters anhand einer Simulation zur Lokalisierung eines Roboters

Raoul Bickmann 2217470
raoul.bickmann@haw-hamburg.de
HAW Hamburg

1 ABSTRACT

In diesem Paper wird die Fehlertoleranz einer Implementation des Unscented Kalman Filter Algorithmus (UKF) analysiert. Hierzu werden mehrere Durchläufe einer Beispielsimulation ausgeführt, in welcher der UKF zur Lokalisierung eines Roboters verwendet wird.

Anschließend werden diese mit fehlerhaften Konfigurationen wiederholt und die Performance zwischen den Konfigurationen verglichen. Nach Auswertung der Ergebnisse lässt sich sagen, dass der UKF, trotz Fehlern in der Konfiguration, vergleichbar präzise Ergebnisse liefern kann, wie ein UKF ohne Fehler.

2 EINLEITUNG

Der Kalman-Filter Algorithmus, zuerst publiziert von [1], extrahiert aus Messdaten den Zustand eines dynamischen Systems. Voraussetzung dafür ist die System-Modellierbarkeit mit linearen Differentialgleichungen sowie Kenntnisse über die Genauigkeit des Modells und der Messungen.

Aufbauend auf dem Kalman Filter wurde von Simon J. Julier und Jeffrey K. Uhlmann der Unscented Kalman Filter für nichtlineare Problemstellungen vorgestellt [2]. Vor dieser Entwicklung existierte bereits der Extended Kalman Filter (EKF), der jedoch für viele Anwendungsmöglichkeiten als komplex zu implementieren oder ungeeignet gilt [2].

Für den UKF führen die Autoren die Unscented Transformation ein. Mit Hilfe dieser, kann der UKF eine bessere Annäherung an das Verhalten einer nicht-linearen Funktion erreichen. Dafür müssen sogenannte Sigma-Punkte festgelegt werden, wobei es für deren Auswahl eine Reihe verschiedener Verfahren gibt [3]. Hierzu gehören beispielsweise die initial vorgestellte Vorgehensweise von Julier und Uhlmann [2] oder ein von Rudolph van der Merwe entwickeltes Verfahren [4].

In dem Buch „Kalman and Bayesian Filters in Python“ [5] beschreibt Roger R. Labbe den Ablauf des UKFs wie folgt:

Zuerst werden im Prädiktionsschritt des UKFs die Sigma-Punkte χ und ihre zugehörigen Gewichtungen w generiert. Auf diese wird die Übergangsfunktion f angewendet:

$$\Upsilon = f(\chi, \Delta t)$$

Darauf folgend können von der resultierenden Punktmenge Υ, durch Nutzung der Unscented Transformation die neuen Vorhersagen des Mittelwertes $\hat{\mu}$ und der Kovarianz $\hat{P}$ bestimmt werden:

$$\hat{\mu} = \sum_{i=0}^{2n} w_i \Upsilon_i$$

$$\hat{P} = \sum_{i=0}^{2n} w_i (\Upsilon_i - \hat{\mu})(\Upsilon_i - \hat{\mu})^T + Q$$

Hier nimmt auch das Systemrauschen Q Einfluss. Anschließend findet der Korrekturschritt statt. Für diesen wird auf die Punkte Υ_i die Messfunktion angewendet, um die zugehörigen Messwerte zu erhalten:

$$Z = h(\Upsilon)$$

Anschließend werden von diesen ebenfalls Mittelwerte μ_z und, unter Einbeziehung des Messrauschens R, auch die Kovarianz P_z ausgerechnet.

$$\mu_z = \sum_{i=0}^{2n} w_i Z_i$$

$$P_z = \sum_{i=0}^{2n} w_i (Z_i - \hat{\mu})(Z_i - \hat{\mu})^T + R$$

Anschließend kann hieraus wie folgt das Kalman Gain berechnet werden:

$$P_{xz} = \sum_{i=0}^{2n} w_i (\Upsilon_i - \hat{\mu})(Z_i - \mu_z)^T$$

$$K = P_{xz} P_z^{-1}$$

Mit Hilfe dessen und der Differenz aus tatsächlichen und vorhergesagten Messwerten $y = z - \mu_z$ kann dann der neue Zustand

$$x = \hat{x} + Ky$$

und die neue Kovarianz

$$P = \hat{P} - K P_z K^T$$

bestimmt werden. Anschließend kann das Verfahren für den nächsten Schritt erneut durchgeführt werden.

In diesem Paper wird die Fehlertoleranz des UKFs anhand einer Beispielsimulation getestet.

Hierzu werden verschiedene Arten von Fehlern in die Konfiguration des UKFs hinzugefügt und deren Auswirkungen anhand der Ergebnisunterschiede überprüft.

3 SYSTEMMODELLIERUNG

Für die Analyse des UKFs wurde eine bereits bestehende Simulation aus Roger R. Labbes Buch „Kalman and Bayesian Filters in Python" [5] verwendet, in welcher der Kalman Filter für die Lokalisierung eines Roboters verwendet wird.

Diese Simulation verwendet für den UKF die Bibliothek Filterpy [6], ebenfalls von Labbe, welche den Großteil der Implementation übernimmt. Darin beinhaltet sind eine Funktion, die die Sigma Punkte nach dem Verfahren von Rudolph van der Merwe [4] generiert, sowie Funktionen für Prädiktions- und Korrekturschritt.

Lediglich die anwendungsspezifischen Funktionen und Variablen für diese Schritte müssen erstellt werden.

3.1 Robotermodell der Simulation

In der Simulation verwendet Labbe einen Roboter mit zwei Rädern, angeordnet wie die Räder eines Fahrrads. Der Systemzustand x des UKFs besteht aus der Position und der Orientierung des Roboters:

$$x = [x\ y\ \theta]^T$$

Dieser Roboter bewegt sich mit einer bestimmten Geschwindigkeit v und einem Steuerwinkel α, die auch dem UKF als Control-Input:

$$u = [v\ a]^T$$

zur Verfügung stehen.

Er ist außerdem mit einem Sensor ausgestattet, der zu i bekannten Orientierungspunkten die Entfernung sowie den Winkel misst. Dies sind die Messwerte des Kalman Filters:

$$z_i = [d\ \gamma]^T$$

3.2 Übergangsfunktion

Damit Filterpy den Prädiktionsschritt durchführen kann, erstellt Labbe die Übergangsfunktion f, die aus einem zu einem vorigen Systemzustand x, den Nächsten $\hat{x}$ berechnet.

Diese bekommt zusätzlich den Control Input des Roboters sowie die Zeit eines Simulationsschrittes Δt und die Länge der Radachse l. Berechnet werden kann dann die zurückgelegte Distanz $d = v \cdot \Delta t$.

Daraus ergibt sich, wenn Δt möglichst klein ist und $a \approx 0$ gilt, der nächste Zustand:

$$\hat{x} = x + [\mathrm{d} \cdot \cos\theta\ \ \mathrm{d} \cdot \sin\theta\ \ \theta]^T$$

Ist $a > 0$ lässt dieser sich wie folgt berechnen:

$$\beta = \frac{d}{l} \cdot \tan a$$

$$r = \frac{l}{\tan \alpha}$$

$$\hat{x} = x + \begin{bmatrix} -r \sin\theta + r \sin(\theta + \beta) \\ r \cos\theta - r \cos(\theta + \beta) \\ \beta \end{bmatrix}$$

3.3 Messfunktion

Für den Korrekturschritt wird die Messfunktion h benötigt, welche zu einem Zustand die Messwerte z des Sensors berechnet.

Zusätzlich kennt die Messfunktion dafür die Position der i Orientierungspunkte $p_i = [x_p\ y_p]^T$.

Daraus kann für jeden Orientierungspunkt der Messwert berechnet werden:

$$z_i = \begin{bmatrix} \sqrt{(x_p - x)^2 + (y_p - y)^2} \\ \tan^{-1}(\frac{y_p - y}{x_p - x}) - \theta \end{bmatrix}$$

3.4 Zusatzfunktionen

Für die Berechnungen der Winkel werden zusätzlich noch weitere Funktionen implementiert. Zum einen werden die Winkel nach jeder Berechnung normalisiert, sodass nur Winkel im Intervall $[-\pi,\ \pi)$ vorkommen.

Des anderen muss eine Funktion erstellt werden, welche die Berechnung des gewichteten Mittels für den Zustand und die Messwerte ausführt. Dies kann Filterpy für lineare Zusammenhänge automatisch, für Winkel muss jedoch eine eigene Funktion erstellt werden. Für die Berechnung des Mittels der Winkel wird folgende Formel verwendet:

$$\theta = \mathrm{atan2}(\sum \sin\theta_i w_i, \sum \cos\theta_i w_i)$$

4 SIMULATION

4.1 Vorbereitungen

Die Simulation von Labbe wurde in Python implementiert. Die Fortbewegung des Roboters findet mit der Übergangsfunktion f zehnmal pro Sekunde statt. Hierzu wurden für jeden Durchlauf der Simulation jeweils die gleichen Control Inputs benutzt, sodass jedes Mal die gleiche Strecke zurückgelegt wurde.

Bei jedem zehnten Simulationsschritt wird ein Schritt des UKFs ausgeführt. Die Länge der Radachse des Roboters wurde auf 0.5m festgelegt.

Der Roboter startet auf Position (0,0) des Koordinatensystems und die Orientierungspunkte jeweils auf (10, 20) und (80, 20). Zu diesen werden bei jedem UKF-Schritt Entfernung d und Winkel γ zum Roboter berechnet und als Messwert übergeben.

Zu beiden Werten werden normalverteilte Rauschterme addiert, um die Ungenauigkeit der Sensoren zu simulieren. Diese liegen bei $\sigma_d = 0{,}1$ für die Entfernung und $\sigma_\gamma = 0{,}3$ für den Winkel.

Daraus kann für die Matrix des Messrauschens R zweier voneinander unabhängigen Messwerte abgeleitet werden:

$$R = \begin{bmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_\gamma^2 \end{bmatrix} = \begin{bmatrix} 0{,}1 & 0 \\ 0 & 0{,}09 \end{bmatrix}$$

Das Systemrauschen Q wurde festgelegt auf

$$Q = \begin{bmatrix} 0{,}1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0{,}05 \end{bmatrix}$$

Die vollständige Implementation der Simulation kann im Github-Repository zu diesem Paper eingesehen werden [8].

4.2 Methodik

Um die Fehlertoleranz des UKFs zu ermitteln, wurden 1000 Simulationen mit der Standardkonfiguration (Konfiguration 1) des UKFs durchgeführt. Anschließend gab es weitere vier Durchläufe, mit jeweils 1000 Simulationen, bei denen die Konfiguration des UKFs durch Fehler von den richtigen Werten der Simulation abweicht.

Die fehlerhaften Parameter wurden mit einer Funktion der Bibliothek NumPy [9] normalverteilt generiert. Dafür wurde der richtige Wert als Mittelwert gesetzt und die Standardabweichung je nach Konfiguration geändert.

Beim ersten Durchlauf beträgt die Standardabweichung der Länge der Radachse 0,05m (Konfiguration 2), beim zweiten Durchlauf 0,125m (Konfiguration 3). Zwei weitere Durchläufe wurden durchgeführt, bei denen stattdessen die Position der Orientierungspunkte fehlerhaft ist. Diese haben eine Standardabweichung in Richtung x und y von 0,1m (Konfiguration 4) und 0,25m (Konfiguration 5).

Um die Performance des UKFs in den jeweiligen Konfigurationen miteinander vergleichen zu können, wurde in jedem UKF-Schritt die Differenz $\Delta p = [\Delta x \;\; \Delta y]$ zwischen der vom UKF approximierten zur wirklichen Position errechnet.

Anschließend konnte für jeden Simulationsdurchlauf mit n Schritten die durchschnittliche Abweichung des UKFs über den Verlauf der Simulation berechnet werden.

$$A_d = \frac{1}{n}\sum_{i=1}^{n} \Delta p_i$$

Um auch zu erfassen wie genau die Positionsschätzung des UKF am Ende der zurückgelegten Strecke ist, wurde die finale Abweichung am Ende jeder Simulation notiert:

$$A_f = \Delta p_n$$

Von beiden Werten wurde erneut das arithmetische Mittel über alle Simulationen einer Konfiguration gebildet. Daraus ergeben sich die Mittelwerte der durchschnittlichen Abweichungen

$$M_d = \frac{1}{n}\sum_{i=1}^{1000} A_{di}$$

und die Mittelwerte der finalen Abweichungen

$$M_f = \frac{1}{n}\sum_{i=1}^{1000} A_{fi}$$

4.3 Ergebnisse

In Abbildung 1 sind die zurückgelegten Strecken der Simulationen dargestellt. Es lässt sich erkennen, dass die geschätzten Streckenverläufe größtenteils dicht beieinander liegen und lediglich in den Kurven stärker auffächern. Zudem sind bei Konfiguration 3 einige besonders starke Ausreißer zu sehen.

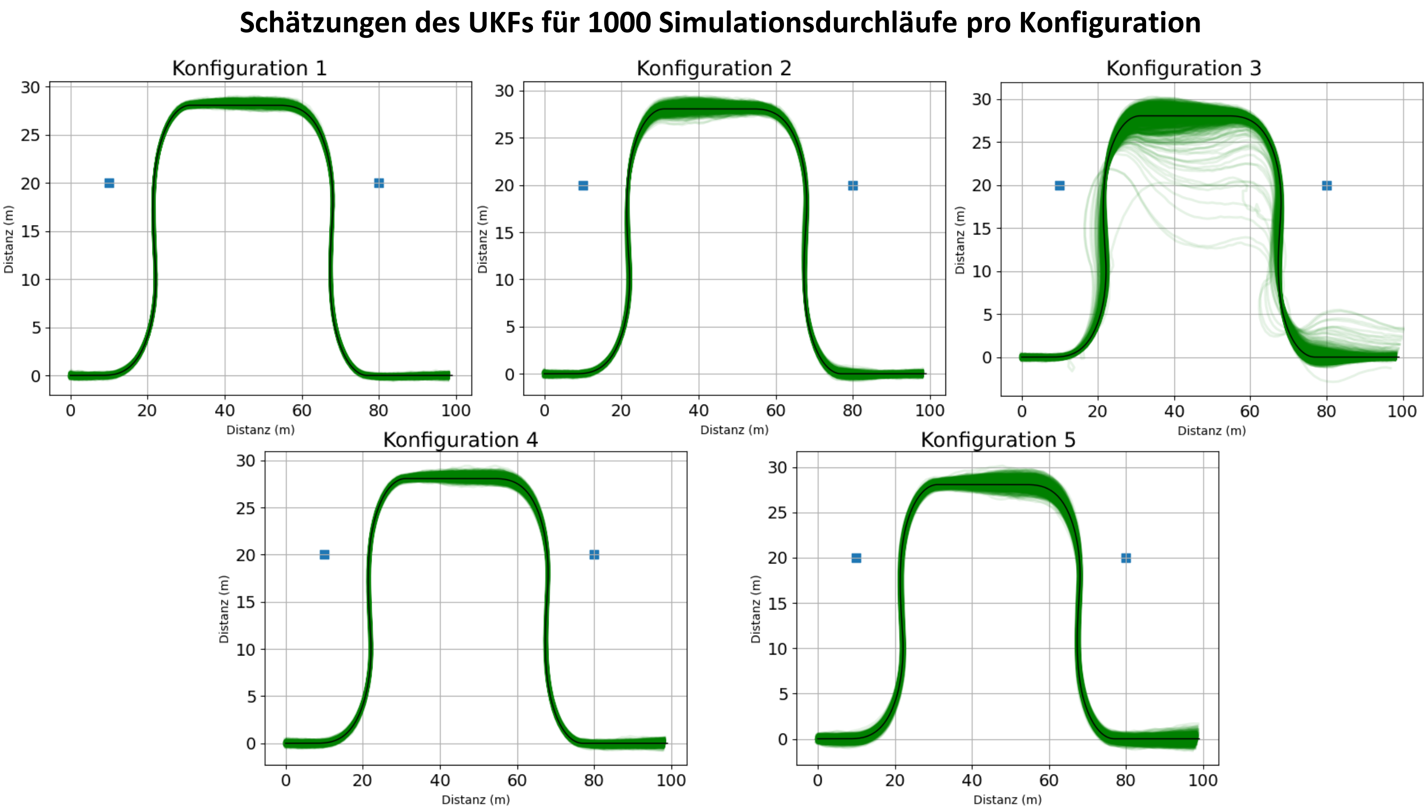

Abbildung 1: Von links oben nach rechts unten Konfigurationen 1 bis 5, Grün: Zurückgelegte Strecken des Roboters nach Schätzung des UKFs, Blau: Orientierungspunkte, Schwarz: Tatsächliche Strecke

Wie sich an Abbildung 2 erkennen lässt, sind die Differenzen der Konfigurationen 2 und 4 zu der fehlerfreien Konfiguration 1 sehr gering. So ist die Differenz der durchschnittlichen Abweichungen zwischen der Konfiguration 1 und 2 circa 0,012m in x- und 0,044m in y-Richtung. Zwischen Konfigurationen 1 und 4 liegen die Differenzen bei 0,024m und 0,039m.

Bei den Konfigurationen 3 und 5 mit stärkeren Abweichungen zu den simulierten Werten, lassen sich größere Differenzen feststellen. Diese liegen bei Konfiguration 3 bei 0,070m in x- und 0,222m in y-Richtung und bei Konfiguration 5 bei 0,099m und 0,160m.

Die Mittelwerte der finalen Abweichungen zeigen ähnliche Ergebnisse. Hier sind ebenfalls bei Konfigurationen 2 (x: < 0,001m, y: -0,005m) und 4 (x: 0,047m, y: 0,063m) nur geringe Unterschiede zur Standardkonfiguration festzustellen. Bei den y-Werten von Konfiguration 2 bestand, aufgrund der Schwankungen des UKFs, im Schnitt sogar eine leicht geringere Abweichung am Ende der Simulation. Zusätzlich sind hier auch bei Konfiguration 3 (x: 0,046m y: 0,049m) die Werte sehr eng an den Werten der Standardkonfiguration.

Nur Konfiguration 5 (x: 0,166m y: 0,234m) zeigt zu diesem Zeitpunkt erneut deutlich größere Abweichungen.

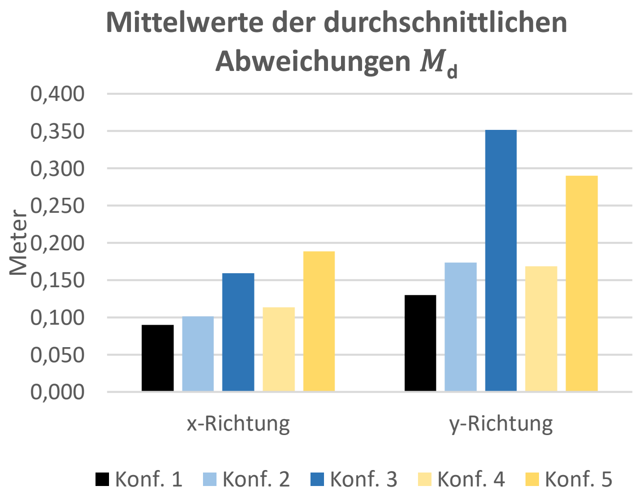

Abbildung 2: Konfiguration 1: Keine Fehler
Konfiguration 2: Radachse Fehler 0,05m
Konfiguration 3: Radachse Fehler 0,125m
Konfiguration 4: Orientierungspunkte Fehler 0,1m
Konfiguration 5: Orientierungspunkte Fehler 0,25m

4.4 Diskussion

Bei den Konfigurationen 2 und 4 ist zu erkennen, dass trotz der Fehler in der Konfiguration, die durchschnittliche Abweichung des Filters nur um bis zu vier Zentimeter ansteigt. Dies ist der Fall, obwohl die Konfigurationsfehler bei beiden in der gleichen Größenordnung von mehreren Zentimetern liegen. Auch an Abbildung 1 lässt sich erkennen, dass die Schätzungen des UKFs nur geringfügig stärker abweichen.

Bei Konfiguration 2 beträgt die Standardabweichung 5 Zentimeter. Dies entspricht 10 Prozent der eigentlichen Länge der Radachse von 50 Zentimetern. Bei Konfiguration 4 sind dies sogar 10 Zentimeter in x- sowie y-Richtung.

Messfehler dieser Höhe sollten bei der Konfiguration eines UKFs durch entsprechende Sorgfalt vermieden werden können, sodass auch die Fehler des UKFs wesentlich geringer bleiben.

Bei den Konfigurationen 3 und 5 sind die Abweichungen dagegen substanziell angestiegen. Diese sind mindestens viermal so hoch, obwohl sich die Konfigurationsfehler im Mittel nur um das 2,5-fache gegenüber den vorherigen Konfigurationen erhöht haben.

Bei so deutlichen Ungenauigkeiten in der Konfiguration kann der UKF dies scheinbar nicht mehr effektiv ausgleichen.

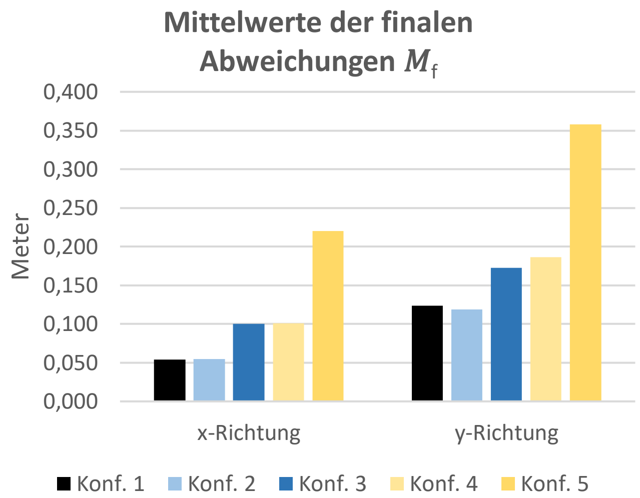

Abbildung 3: Konfiguration 1: Keine Fehler
Konfiguration 2: Radachse Fehler 0,05m
Konfiguration 3: Radachse Fehler 0,125m
Konfiguration 4: Orientierungspunkte Fehler 0,1m
Konfiguration 5: Orientierungspunkte Fehler 0,25m

In Abbildung 1 sind für Konfiguration 3 außerdem einige stärkere Ausreißer zu erkennen. Diese resultieren aus sehr kleinen Werten für die Radachse $l < 0{,}2$m. Bei der Konfiguration mit Werten in der Nähe von 0 sollte demnach bei diesem Anwendungsfall besondere Vorsicht geboten werden.

Auch zu sehen ist, dass selbst bei diesen verhältnismäßig großen Konfigurationsfehlern, die Werte des Kalman Filters nicht endgültig abdriften, sondern sich auch bei den Ausreißern wieder in die Nähe der richtigen Werte zurückkehren. Dies lässt sich auch an den finalen Abweichungen M_f in Abbildung 3 erkennen, die in der gleichen Größenordnung liegen, wie die durchschnittlichen Abweichungen M_d.

Je nach Anwendungsfall muss dann evaluiert werden, ob die Konfigurationsfehler geringer gehalten werden können, oder ob die Abweichungen des UKFs akzeptabel sind.

Denkbar wären beispielsweise medizinische Anwendungsfälle, bei denen selbst die relativ geringen Abweichungen des UKFs, wie bei den Simulationen der ersten Konfigurationen, bereits fatale Folgen haben könnten.

In diesem Fall müsste die Konfiguration des UKFs präziser durchgeführt werden.

Bei anderen Anwendungsfällen, wie beispielsweise der Navigation eines Roboters, ähnlich wie in dieser Simulation, ist es denkbar, dass auch Abweichungen von mehreren Zentimetern für die Orientierung in dem zu navigierenden Gebiet hinreichend sein können.

5 FAZIT

Abschließend lässt sich sagen, dass der Unscented Kalman Filter, trotz Messfehlern bei der Konfiguration, vergleichbar präzise Ergebnisse liefern kann, wie ein fehlerfrei konfigurierter Filter. Für diesen Anwendungsfall, der Lokalisation eines Roboters, lässt sich der UKF daher durchaus verwenden.

Bei weiteren Untersuchungen könnte die Analyse mit Simulationen aus anderen Anwendungsgebieten wiederholt werden, um die vorliegenden Ergebnisse zu validieren.

Auch denkbar wäre eine Wiederholung mit mehreren gleichzeitig abweichenden Faktoren, wie beispielsweise Fehler im Radabstand und in der Position der Orientierungspunkte, im gleichen Simulationsdurchlauf. Hierdurch könnte zusätzlich evaluiert werden, ob mehrere Abweichungen an verschiedenen Stellen, die Ergebnisse stärker beeinflussen, als Abweichungen an einzelnen Punkten.

6 QUELLEN

[1] Kalman, R. E.: A New Approach to Linear Filtering and Prediction Problems, Transaction of the ASME, Journal of Basic Engineering, 82 (Series D) S. 35–45, 1960

[2] Simon J. Julier, Jeffrey K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," Proc. SPIE 3068, Signal Processing, Sensor Fusion, and Target Recognition VI, (28.07.1997); https://doi.org/10.1117/12.280797

[3] H. M. T. Menegaz, J. Y. Ishihara, G. A. Borges and A. N. Vargas, "A Systematization of the Unscented Kalman Filter Theory," in IEEE Transactions on Automatic Control, vol. 60, no. 10, pp. 2583-2598, Oktober 2015. https://doi.org/10.1109/TAC.2015.2404511

[4] Rudolph van der Merwe: "Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models", Dissertation, 04.2004

[5] Roger R Labbe Jr, “Kalman and Bayesian Filters in Python”, https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python, 2015, zuletzt abgerufen: 12.12.2019

[6] Roger R Labbe Jr, Filterpy, https://github.com/rlabbe/filterpy, 2015, zuletzt abgerufen: 12.12.2019

[7] Roger R Labbe Jr, “Kalman and Bayesian Filters in Python”, https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python/blob/master/10-Unscented-Kalman-Filter.ipynb, 2015, zuletzt abgerufen: 12.12.2019

[8] https://github.com/RaoulBickmann/UKFAnalyse

[9]https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.normal.html