Herausgeber

F. HOFFMANN
E. HÜLLERMEIER
R. MIKUT

Dortmund | 29. – 30. November 2018

# PROCEEDINGS **28. WORKSHOP** COMPUTATIONAL INTELLIGENCE

KIT Scientific Publishing

F. Hoffmann, E. Hüllermeier, R. Mikut (Hrsg.)

# Proceedings. 28. Workshop Computational Intelligence

Dortmund, 29. – 30. November 2018

# PROCEEDINGS **28. WORKSHOP** COMPUTATIONAL INTELLIGENCE

**Dortmund, 29. – 30. November 2018**

Herausgegeben von
F. Hoffmann
E. Hüllermeier
R. Mikut

KIT Scientific Publishing

**Impressum**

Scientific Publishing

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe

KIT Scientific Publishing is a registered trademark
of Karlsruhe Institute of Technology.
Reprint using the book cover is not allowed.

www.ksp.kit.edu





Print on Demand 2018 – Gedruckt auf FSC-zertifiziertem Papier

ISBN 978-3-7315-0845-8
DOI 10.5445/KSP/1000085935

# Inhaltsverzeichnis

# Positionswechsel eines Inversen Pendels auf Rädern mittels eines Neuronalen Netzes

Christian Dengler, Boris Lohmann

Technische Universität München
Lehrsthl für Regelungstechnik
E-Mail: {c.dengler, lohmann}@tum.de

## 1 Einführung

Das inverse Pendel auf Rädern ist seit der Einführung als Transportmittel durch Segway Inc. den meisten Menschen bekannt. Während für das Balancieren die lineare Regelungstechnik ausreicht, werden Positionswechsel für gewöhnlich von Menschen gesteuert. Es handelt sich um ein nicht-holonomes System, was automatisierte Positionswechsel in zwei Dimensionen als deutlich komplexeres Problem gestaltet. Im Falle eines analytischen Vorgehens werden hierzu neben einer stabilisierenden Regelung Maßnahmen zum Kurvenfahren sowie eine Trajektoriengenerierung benötigt.

In diesem Beitrag wird ein generischer Ansatz angewendet, welcher die vorher genannten Schritte durch einen einzigen nichtlinearen Regler in Form eines Neuronalen-Netzes ersetzt. Der Regler wird anhand eines Modells trainiert, welches aus einem analytisch hergeleiteten, und einem auf Daten basierten Teil besteht. Während bereits Neuronale Netze für dieses Problem eingesetzt wurden, wurden diese lediglich zur Bestimmung von Parametern von PID-Reglern benutzt [1] und mit Modellansätzen für die Kurvenfahrt kombiniert [2], jedoch nach bestem Wissen der Autoren noch nie als end-to-end Lösung.

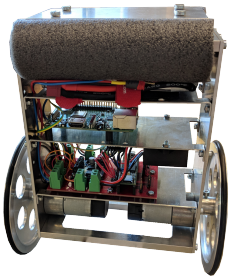


Bild 1: Das für den Versuch verwendete inverse Pendel auf Rädern

## 2 Beschreibung des Inversen Pendel auf Rädern

Das Inverse Pendel auf Rädern ist ein instabiles System, welches idealisiert mittels sieben Zustandsvariablen beschrieben werden kann. Die Zustandsvariablen, zusammengefasst im Zustandsvektor $\mathbf{x}$, sind in Tabelle 1 beschrieben und basieren auf der Modellierung in [3]. Es handelt sich um ein nicht-holonomes System, da ein Verfahren in Richtung der Radachsen nicht möglich ist. Die Eingänge sind hier die jeweiligen Spannungen, welche auf die Motoren aufgebracht werden. Die Eingänge wurden hierbei auf $u_{1,2} \in [-1,1]$ normiert, und entsprechen beim aktuellen Aufbau bis zu 10.5V.

Als wichtige Komponenten besitzt der Roboter neben den entsprechenden Sensoren zum Messen der Zustände (wobei die Position $x, y$ aus den Motor-Encodern aufintegriert wird) einen Mikrocontroller sowie einen Raspberry Pi 3. Letzterer ist nötig, da das Abspeichern der Reglerparameter und Auswerten des Regelgesetzes, welches in Form eines neuronalen Netzes vorliegt, auf dem Mikrocontroller aufgrund des zu geringen Arbeitsspeichers und der Rechenleistung nicht möglich ist.

Tabelle 1: Betrachtete Zustände des inversen Pendel

| Symbol | Beschreibung |
|---|---|
| $x$ | Position in x-Richtung |
| $y$ | Position in y-Richtung |
| $\phi$ | Drehwinkel des Roboters |
| $\alpha$ | Kippwinkel |
| $\dot{\alpha}$ | Kippgeschwindigkeit |
| $v$ | Geschwindigkeit des Roboters |
| $\dot{\phi}$ | Drehgeschwindigkeit |

# 3 Identifikation des dynamischen Modells als Blackbox

In diesem Abschnitt werden die einzelnen Schritte zur Generierung des Modells beschrieben. Zuerst werden die Modellgleichungen mit ihren verbleibenden freien Parametern vorgestellt, daraufhin werden am realen System Daten generiert und im letzten Schritt werden die freien Parameter anhand der Daten trainiert.

## 3.1 Modellstruktur

In der Literatur finden sich bereits einige Modellierungsansätze für das betrachtete System, z.B. in [3, 4]. Wir verwenden das Modell in [3]; da dieses jedoch als Eingang die Momente an den Rädern verwendet, fügen wir eine Blackbox-Komponente mit freien Parametern hinzu, welche die Umwandlung der Stellgröße $u_{1,2}$ zusammen mit dem Zuständen $\dot{\alpha}$, $v$ und $\dot{\phi}$ auf die Radmomente $\tau_{1,2}$ vornimmt. Die Zustände werden hier mit betrachtet aus der Überlegung, dass das Moment des Motors nicht nur von der Spannung, sondern auch der Drehgeschwindigkeit abhängt. Es wird ein Neuronales Netz mit einer versteckten Schicht mit 6 Neuronen verwendet, die freien Parameter des Neuronales Netzes seien in $\boldsymbol{\theta}$ zusammengefasst, daraus ergibt sich für das erste Teilmodell

$$\boldsymbol{\tau}_t = \mathbf{g}_1(\dot{\alpha}, v, \dot{\phi}, \mathbf{u}_t; \boldsymbol{\theta}). \tag{1}$$

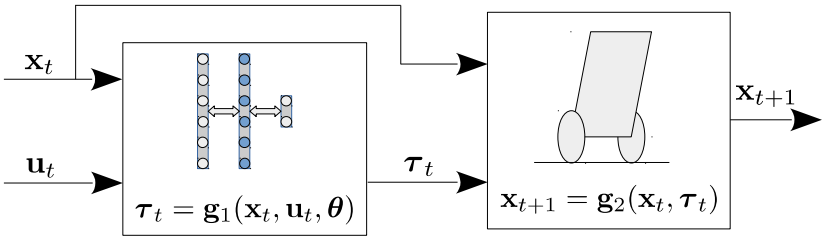



Bild 2: Modellstruktur

Der Index $t$ beschreibt hierbei den diskreten Zeitschritt. Die benötigten Parameter für das analytische Teilmodell sind Massen, Dimensionen und Trägheitsmomente des Roboters. Diese lassen sich entweder messen oder berechnen. Die Modellgleichungen sollen hier nicht wiederholt werden und sind in [3] nachzulesen, die Zeitdiskretisierung wurde mittels des Runge-Kutta-Verfahrens vierter Ordnung implementiert. Dieses Teilmodell hat als Eingänge den Zustand sowie die aus dem ersten Teilmodell berechneten Momente, welche hier in einem Vektor zusammengefasst werden. Das zeitdiskrete Teilmodell schreibt sich

$$\mathbf{x}_{t+1} = \mathbf{g}_2(\mathbf{x}_t, \boldsymbol{\tau}_t). \tag{2}$$

Die Komponenten des Gesamtmodells sind im Bild 2 veranschaulicht. Beide Komponenten zusammen ergeben das benötigte Zustandsraummodell, welches im Folgenden als

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t; \boldsymbol{\theta}) \tag{3}$$

beschrieben wird.

## 3.2 Datengenerierung

Da es sich um ein instabiles System handelt, wurde zuerst ein stabilisierender PID-Regler per Hand getuned, sowie die Stellgröße anschließend mit unterschiedlichen Rauschsignalen beaufschlagt, welche den Roboter immer wieder aus seiner Ruhelage auslenkten und dieser somit zufällig herumfuhr. Die Rauschsignale wurden für unterschiedliche Experimente aus Gleichverteilungen mit unterschiedlichen Varianzen erzeugt und in einigen Experimenten auch

mittels PT1-Filter geglättet. Insgesamt wurden für das Training und die Validierung Daten über 12min16s bei einer Abtastzeit von 0.01s verwendet.

## 3.3 Training der Parameter

Das in Abschnitt 3.1 vorgestellte Modell wurde mittels Datensequenzen zwischen 0.5s und 1s trainiert. Hierfür wurden die Experimente ,welche über längere Zeitabschnitte gingen, in Teiltrajektorien der genannten Länge geschnitten. Von den Teiltrajektorien wurden $4/5$ für das Training und $1/5$ als Testdatensatz verwendet. Die Simulation des Modells wurde mit der gleichen Stellgröße und dem gleichen Anfangswert wie die Trainingsdaten beaufschlagt, und eine gewichtete Summe über die Fehlerquadrate als Gütefunktion $J(\boldsymbol{\theta})$ verwendet. Im Folgenden seien reale Messdaten mit einer Tilde, z.B. $\tilde{x}_t$, gekennzeichnet um diese von dem Zustandswerten aus der Simulation, z.B. $x_t$, zu unterscheiden.

$$\begin{aligned} J(\boldsymbol{\theta}) = & \frac{1}{N_{traj}} \sum_{i=1}^{N_{traj}} \frac{1}{T_i} \sum_{t=0}^{T_i} (x_t - \tilde{x}_t)^2 + (y_t - \tilde{y}_t)^2 + 12(\alpha_t - \tilde{\alpha}_t)^2 \\ & + \left(\sin(\phi_t) - \sin(\tilde{\phi}_t)\right)^2 + \left(\cos(\phi_t) - \cos(\tilde{\phi}_t)\right)^2 \\ & + 2(\dot{\alpha}_t - \tilde{\dot{\alpha}}_t)^2 + 2(v_t - \tilde{v}_t)^2 + 0.4(\dot{\phi}_t - \tilde{\dot{\phi}}_t)^2 \end{aligned} \tag{4}$$

Ein erster Ansatz über die genaue Berechnung des Fehler-Gradienten mittels BackPropagation führte zu erheblichen Rechenzeiten, weshalb als Alternative hier der CMA-ES Algorithmus [8] verwendet wurde. Es handelt sich um ein Verfahren zweiter Ordnung, welches eine Gauß-Verteilung über die Parameter bildet, und deren Mittelwert sowie Kovarianzmatrix mittels Stichproben korrigiert. Das Verfahren benötigt in der Regel mehr Funktionsauswertungen als Gradientenverfahren um zu konvergieren, allerdings werden keine Auswertungen des Gradienten benötigt.

Ein Vergleich von Messdaten mit Simulationsdaten nach dem Training sind in Bild 3 zu sehen. Obwohl der Optimierungsalgorithmus konvergierte zeigt eine Simulation über längere Zeiträume deutliche Abweichungen.

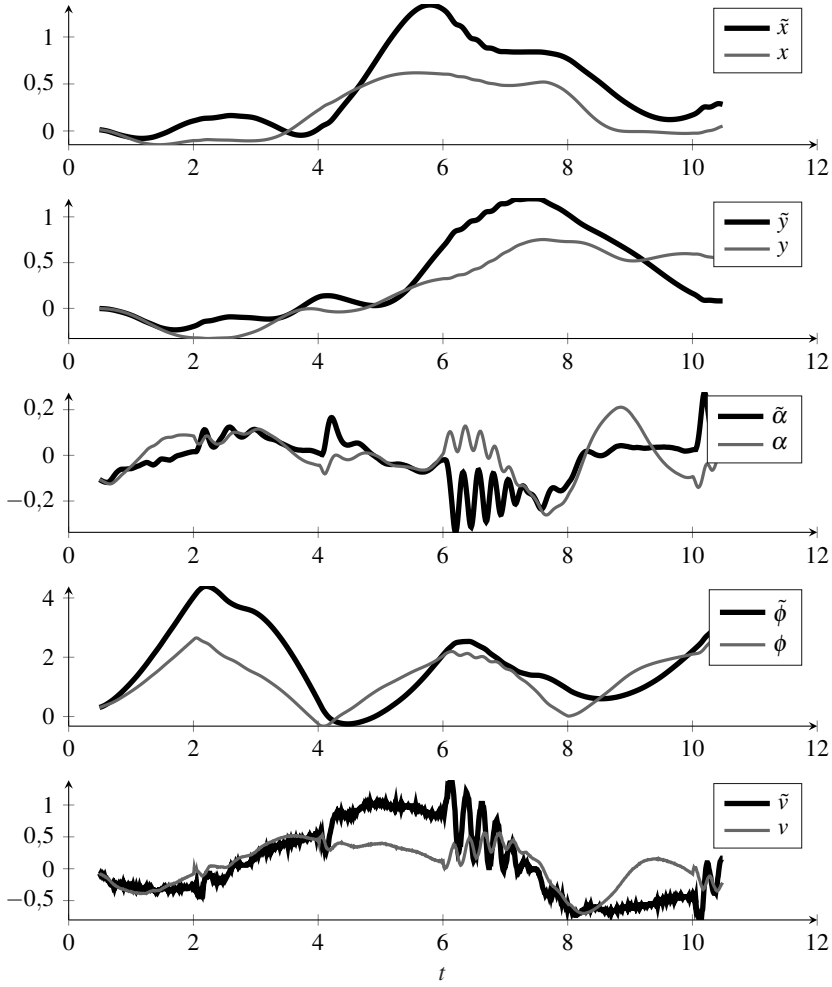



Bild 3: Vergleich zwischen der Simulation des trainierten Modells und Messdaten, bei gleicher Stellgröße. Signale mit einer Tilde kennzeichnen Messdaten, während Signal ohne Tilde aus der Simulation stammen. Ausschnitt über 10s.

# 4 Erstellen eines Blackbox Reglers auf Basis des Modells

Zur Auslegung des Blackbox-Reglers wird das Optimierungsproblem

$$\begin{aligned} \boldsymbol{\psi}^* &= \underset{\psi}{\operatorname{argmin}}\, \mathbb{E}\left[\sum_{t=0}^{T} c(\mathbf{x}_t, \mathbf{u}_t) + c_f(\mathbf{x}_T) | \mathbf{x}_0 \sim d_0(\mathbf{x})\right] \\ \text{s.t. } \mathbf{x}_{t+1} &= \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t; \boldsymbol{\theta}) \\ \mathbf{u}_t &= \mathbf{r}(\mathbf{x}_t; \boldsymbol{\psi}) \end{aligned} \tag{5}$$

gelöst. Die Anfangszustände bei $t = 0$ stammen dabei aus einer vom Benutzer vorzugebenden Verteilung $d_0$ und $T$ beschreibt den Optimierungshorizont. Zu beachten gilt, dass für dieses Optimierungsproblem die Modell-Parameter $\boldsymbol{\theta}$ festgehalten werden, und nur die Regelparameter $\boldsymbol{\psi}$ optimiert werden.

Als mögliche Ansätze zum Lösen des Problems bieten sich hier unter anderem gradientenbasierte Methoden des Reinforcement Learning [5], Evolutionäre Strategien [6], welche z.B. in [7] als Alternative zum Reinforcement Learning verwendet wurden, sowie andere Blackbox-Optimierungsverfahren, z.B. das bereits verwendete CMA-ES [8], an .

Eine angenäherte Lösung an (5) lässt sich ermitteln, indem zuerst optimale Trajektorien mit der gleichen Kostenfunktion erstellt werden, und danach der Regler mittels überwachtem Lernen auf den generierten Trajektorien trainiert wird.

Ein ähnlicher Ansatz wird in [9] verfolgt, wobei dort ein größeres Optimierungsproblem aufgestellt wird welches alle Stellgrößen $\mathbf{u}_{0:T}$ und Zustände $\mathbf{x}_{0:T}$ über alle Trajektorien gemeinsam mit $\boldsymbol{\psi}$ optimiert um die Konvergenz zu verbessern. Dies führt jedoch zu einem deutlich höheren Speicherbedarf und höherer Komplexität.

In diesem Beitrag daher wird die angenäherte Lösung vorgestellt und verwendet. Diese hat den Vorteil, dass beide Optimierungen, die Trajektorienoptimierung und das anschließende überwachte Lernen, in der Regel robust in der Anwendung sind. Im ersten Schritt werden 500 Startpunkte zufällig in einem Radius von 1m um den Ursprung in aufrechter Position erstellt, und für jeden

Startpunkt die optimale Stellgrößen $[\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_T]$ und die auf der Trajektorie besuchten Zustände $[\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T]$ abgespeichert. Für die Trajektorienoptimierung verwenden wir den open-source Optimierer Ipopt [10] über die Schnittstelle in der Programmiersprache Julia [11]. Als Kostenfunktion $c(\mathbf{x}_t, \mathbf{u}_t)$ verwenden wir

$$\begin{aligned} c(\mathbf{x}_t, \mathbf{u}_t) =& 10x^2 + 10y^2 + \sin(\phi)^2 + (\cos(\phi) - 1)^2 + 2\alpha^2 \\ &+ 0.1\dot{\alpha}^2 + 0.1v^2 + 0.1\dot{\phi}^2 + 0.5u_1^2 + 0.5u_2^2 \end{aligned} \tag{6}$$

ohne Endkosten $c_f(\mathbf{x}_T)$ mit den Nebenbedingungen $-1 \leq u_{1,2} \leq 1$ sowie als Endbedingung $x_T = y_T = \cos(\phi_T) - 1 = \alpha_T = 0$. Der Optimierungshorizont beträgt $T = 4$s. Die Endbedingung führt zu einer Abweichung von ursprünglichen Optimierungsproblem (5), verbessert hier jedoch die Konvergenz.

Anschließend wird ein Neuronales Netz mit zwei versteckten Schichten mit jeweils 64 und 32 Neuronen darauf trainiert, die vorher optimierten Stellgrößen nachzubilden. Als Eingang hat das Neuronale Netz demnach den Zustand $\mathbf{x}_t$, wobei $\phi_t$ hier wieder auf $\sin(\phi_t)$ und $\cos(\phi_t)$ aufgeteilt wird, und als Ausgang die Stellgröße $\mathbf{u}_t$. Für das Training wurde ein stochastischer Gradientenabstieg mit Schrittweitenanpassung mittel Adam [12] implementiert und als zu minimierende Güte wird der mittlere quadratische Fehler über den Trainingsdatensatz verwendet. Der Trainingsdatensatz umfasst dabei $4/5$ aller Tupel $(\mathbf{x}_t, \mathbf{u}_t)$, und $1/5$ wird als Testdatensatz verwendet.

In der Simulation stabilisiert der so erhaltene Blackbox-Regler das inverse Pendel auf Rädern für alle getesteten Startpunkte innerhalb des gewählten Radius und fährt es sogar für Startpositionen außerhalb des gewählten Radius von 1m zurück in den Ursprung. In Bild 4 ist eine Simulation ausgehend vom Startpunkt $x_0 = 0.2$, $y_0 = 1.0$ zu sehen. Auffällig ist die verrauschte Stellgröße. Ursachen sind vermutlich ein zu groß gewähltes Neuronalen Netz für den Regler oder auch Probleme, welche vom Modell herführen.

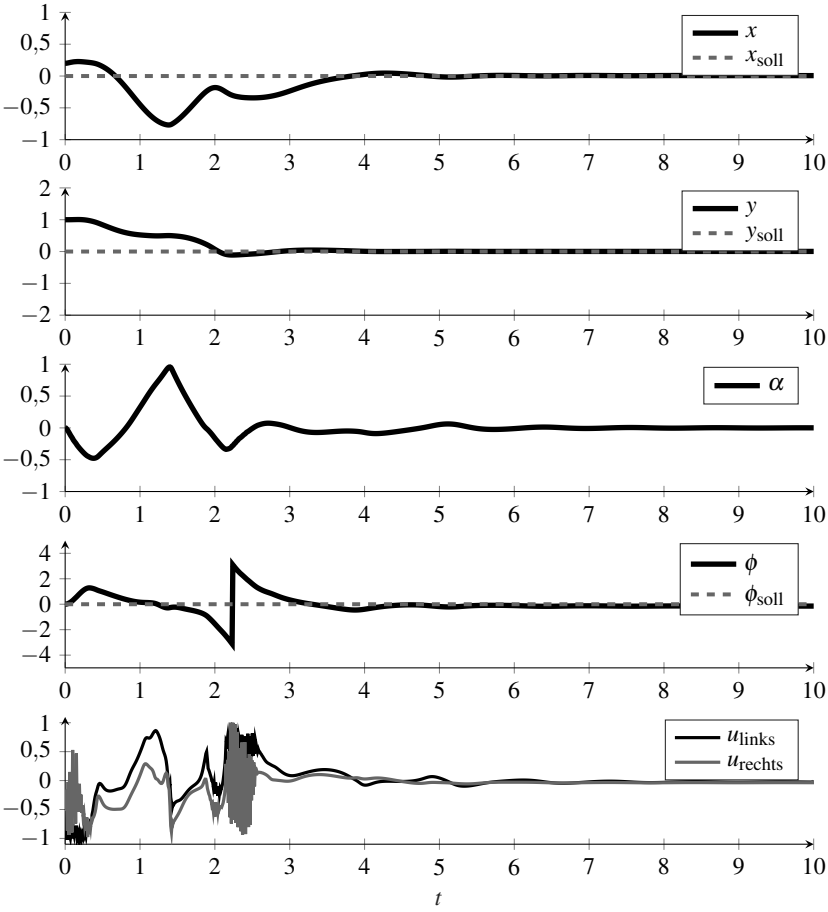



Bild 4: Simulationsergebnisse mit dem Blackbox-Regler an dem trainierten Modell. Der Gierwinkel wurde auf einen Bereich zwischen $-\pi$ und $\pi$ abgeschnitten.

# 5 Übertragung auf das reale System

Die Übertragung des Blackbox-Reglers auf das reale System führt zum Zeitpunkt dieses Beitrags zu einem Umkippen des Segway nach wenigen Sekunden. Eine Messung mit dem gelernten Regler ist in Bild 5 zu sehen. Als Grund für die schlechte Regelperformance wird zum einen der bestehenden Modellfehler vermutet. Die Datenbasis für das Modell wurde mittels eines stabilisierenden Reglers aufgenommen, was der Grund sein könnte, weshalb das gelernte Modell weniger leicht umkippt als das reale System. Wie in Bild 4 zu sehen, lässt sich das Modell noch ab einem Kippwinkel von fast 1rad noch aufrichten, was am realen System nicht möglich wäre. Zum anderen könnten in den Trainingsdaten für den Regler zu wenige Daten für den Fall eines Umkippens vorhanden sein, sodass der Regler hier nicht weiß, wie er agieren soll. In den Trainingsdaten aus den optimierten Trajektorien kippt der Roboter für gewöhnlich nur während des Beschleunigens und des Abbremsens stark, jedoch nie ungewollt.

# 6 Zusammenfassung

In dem Beitrag wurde ein Blackbox-Regler für das inverse Pendel auf Rädern mittels überwachtem Lernen auf optimalen Trajektorien vorgestellt. Als Modell wurde dabei eine Mischung aus einem Neuronalen Netz und einem analytischen Modell gewählt und die freien Parameter anhand von Daten trainiert. Während der gelernte Regler in der Simulation beliebige Positionen anfahren kann, ist die Übertragbarkeit auf das reale System zum jetzigen Zeitpunkt nicht gegeben. Als Grund wird das daten-basierte Modell vermutet, welches ein stabileres Verhalten aufweist, als das echte System.

Als Ausblick sind deshalb Verbesserungen geplant, welche den Ansatz für die Praxis ausreifen sollen. Hierzu ist zum einen eine Verbesserung des Modells geplant, als auch das Implementieren und Evaluieren weiterer Trainingsmethoden für den Regler mit mehr Fokus auf Robustheit gegenüber Modellungenauigkeiten. Hierfür sind die Methoden des Reinforcement-Learning und der Blackbox Optimierung geeignet, welche auch mit stochastischen Modellen umgehen können.

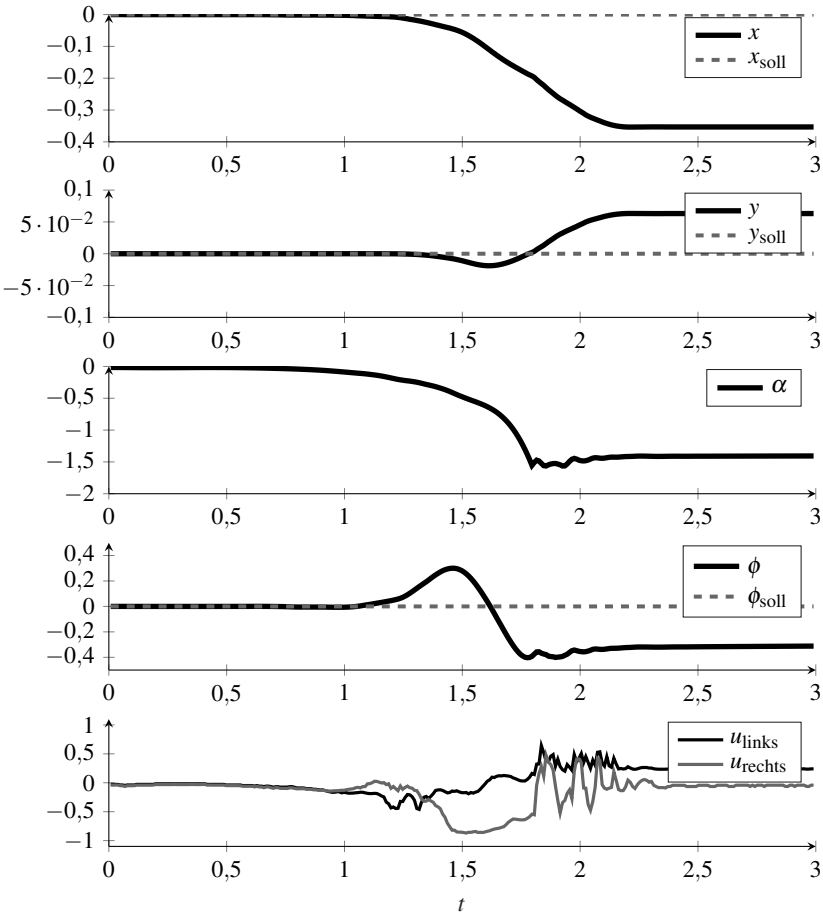



Bild 5: Der gelernte Regler kann das reale System nicht im Ursprung stabilisieren und kippt nach weniger als 2s um.

## Danksagung

Der Autor dankt der Deutschen Forschungsgemeinschaft (DFG) für die Förderung dieser Forschung im Rahmen des Sonderforschungsbereichs 768 (Zyklenmanagement von Innovationsprozessen – verzahnte Entwicklung von Leistungsbündeln auf Basis technischer Produkte).

## Literatur

[1] S. Jung, S. S. Kim: „Control experiment of a wheel-driven mobile inverted pendulum using neural network“. IEEE Transactions on Control Systems Technology, IEEE. 2008.

[2] J. S. Noh, G. H. Lee , S. Jung: „Position control of a mobile inverted pendulum system using radial basis function network“ International Journal of Control, Automation and Systems, Springer. 2010.

[3] K. Pathak, J. Franch, S. K. Agrawal: „Velocity and position control of a wheeled inverted pendulum by partial feedback linearization“ IEEE Transactions on robotics 2005.

[4] S. Kim, S. Kwon: „Dynamic modeling of a two-wheeled inverted pendulum balancing mobile robot“. International Journal of Control, Automation and Systems: Springer. 2015.

[5] R. S. Sutton, D. McAllester, S. Singh, Y. Mansour: „Policy Gradient Methods for Reinforcement Learning with Function Approximation“ Advances in Neural Information Processing Systems 2000.

[6] H.-G. Beyer, H.-P. Schwefel: „Evolution strategies–A comprehensive introduction“ Natural computing, Springer 2002.

[7] T. Salimans,J. Ho, X. Chen, I. Sutskever: „Evolution strategies as a scalable alternative to reinforcement learning“ arXiv preprint 2017.

[8] N. Hansen:„The CMA Evolution Strategy: A Tutorial“. CoRR. 2016.

[9] I. Mordatch, E. Todorov: „Combining the benefits of function approximation and trajectory optimization“. Robotics: Science and Systems 2014.

[10] A. Wächter, L. T. Biegler: „On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming“ Mathematical programming, Springer 2006.

[11] J. Bezanson, A. Edelman, S. Karpinski, V.B. Shah: „Julia: A fresh approach to numerical computing“ SIAM review, SIAM 2017.

[12] D. P. Kingma, J. Ba: „Adam: A Method for Stochastic Optimization“ CoRR 2014.

# Integrierter Entwurf zur Fehlerrekonstruktion und fehlertoleranten Regelung für eine Klasse von Takagi-Sugeno Fuzzy Systemen

Horst Schulte, Nico Goldschmidt

Hochschule für Technik und Wirtschaft Berlin,
Fachbereich Ingenieurwissenschaften - Energie und Information,
Fachgebiet Regelungstechnik, 12459 Berlin
E-Mail: {schulte,nico.goldschmidt}@htw-berlin.de

## Kurzfassung

In diesem Beitrag wird ein integrierter LMI-basierter Entwurf zur Fehlerrekonstruktion und fehlertoleranten Regelung für eine Klasse von Takago-Sugeno Fuzzy Systemen vorgestellt. Die Formulierung der Anforderungen an den Regelkreis erfolgt über lineare Matrixungleichungen. Die Anwendbarkeit und Leistungsfähigkeit des Entwurfsverfahrens wird am Beispiel einer fehlertoleranten Regelung von AC-DC-AC Umrichtern gezeigt.

## 1 Einführung

In den letzten zwei Dekaden ist das Interesse an dem systematischen Entwurf von Fehlerdiagnosesystemen (FD) und der fehlertoleranten Regelung (FTC) auch aufgrund des steigenden Automatisierungsgrad von sicherheitskritischen Anwendungen enorm gestiegen [1]. Bisher werden die Algorithmen zur Fehlerdiagnose und der fehlertoleranten Regelung separat entworfen und implementiert [2]. Zur Fehlerdiagnose werden sowohl modell- als auch datengestützte Verfahren eingesetzt. In diesem Beitrag beschränken wir uns auf das erstgenannte Verfahren, das Beobachter einsetzt, die mathematische Modelle des zu regelnden Prozesses enthalten. Klassisch werden dynamische Beobach-

ter verwendet, um aus der Differenz zwischen dem geschätzten Prozessausgang und den Messungen Residuen zu generieren, die anschließend zur Fehlerdetektion und Fehlerisolation ausgewertet werden. Problematisch ist hierbei die Festlegung von geeigneten Schwellwerten bzw. Schwellwertfunktionen (statisch/dynamisch) und die Annahme, dass der aufgetretene Fehler im Prozess dazu führt, dass sich die Residuen signifikant ändern (Fehlerdetektion). Weiterhin müssen die Fehler nicht nur detektiert sondern auch isoliert werden, d.h. die Residuen müssen so ausgewertet werden, dass Prozessfehler eindeutig identifiziert werden. Um mehrere Fehler unterscheiden zu können, müssen hierfür Beobachter parallel betrieben werden. Die Struktur der parallelen Beobachter wird als Beobachterbank bezeichnet.

Anstatt Residuen auszuwerten, können Beobachter auch dazu verwendet werden, Fehlersignale zu rekonstruieren. Dabei sind die Fehlersignale ein Konstrukt, um die Wirkung im Prozessmodell mittels additiver Signale in der Zustandsgleichung (sogenannte Aktuatorfehler) und additiver Signale am Systemausgang (Sensorfehler) zu beschreiben. Erstmalig wurde in [3] für ein lineares Zustandsraummodell ein Unknown Input Observer (UIO) vorgestellt, mit dem Aktuatorfehler rekonstruierbar sind. Dadurch entfällt nicht nur die Residuenauswertung, es ist unter bestimmten Voraussetzungen auch möglich, die Wirkung des Fehlers über eine Störgrößenaufschaltung zu kompensieren und im Idealfall zu eliminieren.

Fehlertolerante Regelungskonzepte werden in passive und aktive Verfahren unterteilt. Bei der passiven fehlertoleranten Regelung wird beim Auftreten des Fehlers die Struktur nicht verändert. Der Fehler wird dabei als unbekannte äußere Störung oder Modellunsicherheit interpretiert, für die der Regler robust ausgelegt werden muss. Diese Ansätze sind nur einsetzbar, wenn die Fehler keine gravierenden Änderungen in der Prozessdynamik verursachen, der nominale Fall sich nicht signifikant vom Fehlerfall unterscheidet und die Regelung daher nicht zu konservativ ausgelegt werden muss. Treten Fehler im Prozess auf, die gravierende Änderungen der Prozessdynamik verursachen, führt der passive Ansatz meist nicht zum Ziel, da die Dynamik des geschlossenen Kreises soweit eingeschränkt ist, dass die prozessbedingten Anforderungen nicht mehr erfüllt werden können. In diesem Fall werden aktive Verfahren eingesetzt, die den Fehler zunächst detektieren, isolieren und gegebenenfalls die Fehlersignale rekonstruieren, um anschließend mit dieser Information die Struktur

der Regelung oder die Reglerparameter anzupassen. Die Anpassung hat zum Ziel, den Einfluss des aufgetretenen Fehlers auf die Prozessdynamik so klein wie möglich zu halten. Das erstgenannte Verfahren hat den Vorteil, dass man im Prinzip die Regelung für den nominalen Fall (nominale Regelung) und den Fehlerfall getrennt entwerfen kann. Damit wird die Struktur überschaubarer und die Performance der Regelung wird beim Auftreten eines Fehlers durch das aktive inverse Aufschalten der rekonstruierten Sensor- und Aktuatorfehler im Idealfall nicht beeinträchtigt [4]. Dem sind aufgrund der nicht idealen Rekonstruktion der Fehler jedoch Grenzen gesetzt, so dass auch hierbei die Regelung robust ausgelegt werden muss.

Für Takagi-Sugeno (TS) Fuzzy Systeme [5] mit Sensor- und Aktuatorfehlern, die den Ansatz der Rekonstruktion und Kompensation durch Aufschaltung aufgreifen, sind in den letzten Jahren verschiedene Entwurfsverfahren vorgestellt worden. In der Arbeit [6] wird eine passive fehlertolerante Regelung mittels Ausgangsrückführung vorgestellt. Zur simultanen Schätzung der Systemzustände und Aktuatorfehler unter Berücksichtigung der fehlertoleranten Regelungseigenschaften wird in [7] ein LMI[1]-basiertes Verfahren entwickelt. Ein fehlertoleranter Reglerentwurf für zeitdiskrete Takagi-Sugeno Systeme mittels Anwendung des Delta-Operators [8] wird in [9] diskutiert. Im ersten Schritt wird ein TS Beobachter in Delta-Operator-Schreibweise zur Fehlerrekonstruktion vorgestellt. Danach wird eine aktive fehlertolerante Regelung mit garantierter Stabilität des geschlossenen Regelkreises (auch im Fehlerfall) entworfen. Eine weitere Variante der beobachterbasierten Fehlersignalschätzung für FTC Anwendungen wurde in [10] und [11] untersucht. Hierzu wurden die Eigenschaften von Unknown Input Observer zur Schätzung unbekannter Eingänge analysiert, wobei die Aktuatorfehler als unbekannte Eingänge interpretiert worden sind.

Eine ganze Reihe von Untersuchungen zu strukturvariablen TS Fuzzy Beobachtern im Kontext der FTC ist in den Arbeiten [12], [13], [14] durchgeführt worden. Ein weiterer Beobachtertyp, der Proportionale-Multi-Integral (PMI) Beobachter, der Aufgrund der Skalierbarkeit gut für den integrierten FTC Entwurf geeignet ist, wurde in [15] erstmals für LTI Systeme vorgestellt und in [16] auf TS Fuzzy Systeme erweitert .

[1] Lineare Matrix Inequality, Lineare Matrixungleichung (dt.)

In dieser Arbeit wird, im Gegensatz zu den vorhergehenden Studien, welche die Beobachter und FTC Regler unabhängig entwerfen, eine integrierte Synthese für eine Klasse von Takagi-Sugeno Fuzzy Systemen vorgestellt. Der Ansatz basiert auf einem erweiterten PMI Beobachter für die gemeinsame Rekonstruktion der Systemzustände und Aktuatorfehler in Form von additiven Fehlersignalen, wodurch mittels Störgrößenaufschaltung am Ausgang des nominalen Zustandsreglers ein fehlertolerantes Verhalten erreicht wird.

*Anmerkung*: In diesem Aufsatz werden die folgenden Notationen verwendet: Die Einheitsmatrix mit der Dimension $n$ wird mit $\mathbf{I}_n$ bezeichnet. Die Nullmatrix mit $n$-Zeilen und $m$-Spalten notieren wir mit $\mathbf{O}_{n\times m}$. Und durch $\mathbf{P} \succ 0$, ($\mathbf{P} \prec 0$) wird ausgedrückt, dass die quadratische Matrix $\mathbf{P}$ positiv (negativ) definit ist.

# 2 Problemformulierung mit Takagi-Sugeno Fuzzy Systemen

In diesem Abschnitt wird der formale Rahmen der Prozessmodellierung für eine spezielle Klasse von Takagi-Sugneo Fuzzy Systeme mit Berücksichtigung von Fehlertermen beschrieben. Danach wird ein PMI Beobachterschema vorgestellt und die für den Entwurf notwendigen Rangbedingungen werden angegeben. Abschließend wird die Integration der Beobachterstruktur in das fehlertolerante Regelungsschema erläutert.

## 2.1 Takagi-Sugeno Fuzzy Systeme mit Aktuatorfehlern

Die in dieser Arbeit zu untersuchende Formulierung der Reglersynthese für nichtlineare Systeme in Takagi-Sugeno Form basiert auf der folgenden Systemklasse

$$\begin{aligned} \dot{\mathbf{x}}(t) =& \sum_{i=1}^{N_r} h_i(\mathbf{z}(t))\,\mathbf{A}_i\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) + \mathbf{B}_f\,\mathbf{f}(t)\,, \\ \mathbf{y}(t) =& \mathbf{C}\,\mathbf{x}(t)\,, \end{aligned} \tag{1}$$

mit $\mathbf{A}_i$ als Systemmatrix der $i = 1,\ldots,N_r$ Teilsysteme, $\mathbf{B} \in \mathbb{R}^{n\times m}$ als gemeinsame Eingangsmatrix (für alle Teilmodelle identisch), der gemeinsamen Ausgangsmatrix $\mathbf{C} \in \mathbb{R}^{p\times n}$ und den Aktuatorfehlern $\mathbf{f} \in \mathbb{R}^k$. Die Prämissenvariablen werden in $\mathbf{z} \in \mathbb{R}^l$ zusammengefasst und als messbar angenommen, $\mathbf{x} \in \mathbb{R}^n$ ist der Systemzustandsvektor, $\mathbf{u} \in \mathbb{R}^m$ der Eingangsvektor und $\mathbf{y} \in \mathbb{R}^p$ der Ausgangsvektor. Die Funktionen $h_i : \mathbb{R}^l \mapsto \mathbb{R}$ gewichten die $\mathbf{z}(t)$-abhängige Zugehörigkeit der Teilmodelle zum TS System. Diese erfüllen die konvexe Summenbedingung:

$$\begin{aligned} \sum_{i=1}^{N_r} h_i(\mathbf{z}) = 1\,, \quad i = 1,\ldots,N_r\,, \\ h_i(\mathbf{z}) \geq 0\,, \quad i = 1,\ldots,N_r\,. \end{aligned} \tag{2}$$

## 2.2 Proportionale-Multi-Integral Beobachter zur Zustandsschätzung und Fehlerrekonstruktion

Zur gemeinsamen Zustandsschätzung und Fehlerrekonstruktion in einem Beobachter wird das zugrunde liegende Prozessmodell in Takagi-Sugeno Form wie folgt erweitert:

$$\begin{aligned} \dot{\tilde{\mathbf{x}}}(t) &= \sum_{i=1}^{N_r} h_i(\mathbf{z}(t))\,\tilde{\mathbf{A}}_i\,\tilde{\mathbf{x}}(t) + \tilde{\mathbf{B}}\,\mathbf{u}(t) + \tilde{\mathbf{G}}\,\mathbf{f}^{(q)}(t)\,, \\ \mathbf{y}(t) &= \tilde{\mathbf{C}}\,\tilde{\mathbf{x}}(t) \end{aligned} \tag{3}$$

mit dem erweiterten Zustandsvektor

$$\tilde{\mathbf{x}} = \begin{pmatrix} \mathbf{x}^T & \boldsymbol{\xi}_1^T & \boldsymbol{\xi}_2^T & \cdots & \boldsymbol{\xi}_q^T \end{pmatrix}^T \in \mathbb{R}^{\tilde{n}}, \tag{4}$$

wobei

$$\begin{aligned}
\dot{\boldsymbol{\xi}}_1 &= \mathbf{f}^{(q)} \quad , \\
\dot{\boldsymbol{\xi}}_2 = \boldsymbol{\xi}_1 &= \mathbf{f}^{(q-1)} , \\
\dot{\boldsymbol{\xi}}_3 = \boldsymbol{\xi}_2 &= \mathbf{f}^{(q-2)} , \\
\vdots &\quad \vdots \\
\dot{\boldsymbol{\xi}}_q = \boldsymbol{\xi}_{q-1} &= \mathbf{f}^{(1)} , \\
\boldsymbol{\xi}_q &= \mathbf{f} \quad .
\end{aligned} \tag{5}$$

Die Erweiterung enthält somit den Fehlervektor $\mathbf{f}$ und die $(q-1)$'te Ableitungen. Die verbleibende $q$-te Ableitung des Fehlersignals geht als unbekannte Störung in die Zustandsgleichung ein. Die Matrizen des erweiterten Systems (3) werden wie folgt gebildet

$$\begin{aligned}
\tilde{\mathbf{A}}_i &= \begin{pmatrix} \mathbf{A}_i & \mathbf{0}_{n\times k(q-1)} & \mathbf{B}_{f\,n\times k} \\ & \mathbf{0}_{k\times\tilde{n}} & \\ \mathbf{0}_{k(q-1)\times n} & \mathbf{I}_{k(q-1)} & \mathbf{0}_{k(q-1)\times k} \end{pmatrix} \in \mathbb{R}^{\tilde{n}\times\tilde{n}} , \\
\tilde{\mathbf{B}} &= \begin{pmatrix} \mathbf{B} \\ \mathbf{0}_{kq\times m} \end{pmatrix} \in \mathbb{R}^{\tilde{n}\times\tilde{n}} , \quad \tilde{\mathbf{C}} = \begin{pmatrix} \mathbf{C} & \mathbf{0}_{p\times kq} \end{pmatrix} \in \mathbb{R}^{p\times\tilde{n}} , \\
\tilde{\mathbf{G}} &= \begin{pmatrix} \mathbf{0}_{n\times k} \\ \mathbf{I}_k \\ \mathbf{0}_{k(q-1)\times k} \end{pmatrix} \in \mathbb{R}^{\tilde{n}\times\tilde{k}}
\end{aligned}$$

mit $\tilde{n} = n + k\,q$.

Basierend auf dem erweiterten TS Fuzzy System hat der PMI Beobachter folgende Form:

$$\begin{aligned}
\dot{\hat{\tilde{\mathbf{x}}}} &= \sum_{i=1}^{N_r} h_i(\mathbf{z}) \left[ \tilde{\mathbf{A}}_i \hat{\tilde{\mathbf{x}}} + \tilde{\mathbf{L}}_i (\mathbf{y} - \hat{\mathbf{y}}) \right] + \tilde{\mathbf{B}}\,\mathbf{u} , \\
\hat{\mathbf{y}} &= \tilde{\mathbf{C}}\,\hat{\tilde{\mathbf{x}}} .
\end{aligned} \tag{6}$$

Die Differentialgleichung des PMI Beobachterfehlers

$$\dot{\mathbf{e}} = \sum_{i=1}^{N_r} h_i(\mathbf{z}) \left[\tilde{\mathbf{A}}_i - \tilde{\mathbf{L}}_i \tilde{\mathbf{C}}\right] \mathbf{e} + \tilde{\mathbf{G}} \mathbf{f}^{(q)} \tag{7}$$

mit $\mathbf{e} = \tilde{\mathbf{x}} - \hat{\tilde{\mathbf{x}}}$ folgt aus $\dot{\mathbf{e}} = \dot{\tilde{\mathbf{x}}} - \dot{\hat{\tilde{\mathbf{x}}}}$ wobei die Ableitungen der rechten Seite durch (3) und (6) ersetzt werden.

## 2.3 Beobachterentwurf: Notwendige Rangbedingungen

Der in diesem Beitrag vorgestellte Beobachterentwurf setzt voraus, dass die folgenden Bedingungen für die einzelnen Teilmodelle von (1) und dem erweiterten TS Fuzzy System (3) erfüllt sind:

- $p \geq k$ mit $p$ als Anzahl der Ausgänge und $k$ als die Anzahl der zu rekonstruierenden Fehler
- Alle Teilmodelle mit $(\mathbf{A}_i, \mathbf{C})$ aus (1) sind beobachtbar.
- Die Teilmodelle $(\tilde{\mathbf{A}}_i, \tilde{\mathbf{C}})$ von (3) sind beobachtbar.
- Die Rangbedingung

$$\operatorname{rank} \begin{pmatrix} \mathbf{A}_i & \mathbf{B}_f \\ \mathbf{C} & \mathbf{0}_{p \times k} \end{pmatrix} = n + k \tag{8}$$

  muss erfüllt sein.

## 2.4 Beobachterbasierte fehlertolerante Reglerstruktur in Takagi-Sugeno Form

Die fehlertolerante Takagi-Sugeno Regelung

$$\mathbf{u}(t) = -\sum_{i=1}^{N_r} h_i(\mathbf{z}(t)) \mathbf{K}_i \hat{\mathbf{x}}(t) - \hat{\mathbf{f}}(t) \tag{9}$$

basiert auf einer klassischen Zustandsrückführung mit $h_i$-gewichteten Rückführungsmatrizen $\mathbf{K}_i$ und einer Störgrößenaufschaltung mit dem durch den

PMI Beobachter (6) geschätzten Fehlervektor $\hat{\mathbf{f}}(t)$ , der in der Schätzung von dem erweiterten Zustandsvektor (5) enthalten ist. Bei dem Regelungsgesetz wird vorausgesetzt, dass die Fehlerverteilungsmatrix identisch zur Eingangsmatrix ist ($\mathbf{B}_f = \mathbf{B}$). Das heißt, das Fehlersignal wirkt als additive Störung auf den Eingang.

# 3 Integrierte Synthese beobachterbasierter Zustandsregler mit fehlertolerantem Verhalten

In diesem Abschnitt wird die Analyse und Synthese der integrierten Fehlerrekonstruktion und fehlertoleranten Regelung mittels linearer Matrixungleichungen vorgestellt. Nach einem kurzen Abriss zur Bedeutung von LMIs in der Kontrolltheorie werden die LMI Nebenbedingungen zur Systemanalyse und robusten Synthese motiviert und hergeleitet.

## 3.1 Lineare Matrix Ungleichungen in der Kontrolltheorie

Lineare Matrixungleichungen sind mit der Stabilitätstheorie von Lyapunov eng verknüpft. Dieser hat in seiner Arbeit gezeigt [18], dass das Differentialgleichungssystem

$$\dot{\mathbf{x}}(t) = \mathbf{A}\,\mathbf{x}(t) \tag{10}$$

asymptotisch stabil ist, wenn es eine positiv definite Matrix $\mathbf{P}$ gibt, die die lineare Matrixungleichungen

$$\mathbf{P} \succ 0, \quad \mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} \prec 0 \tag{11}$$

erfüllt. Die Ermittlung einer Lösung $\mathbf{P} \succ 0$ lässt sich mit einem gewählten $\mathbf{Q} = \mathbf{Q}^T \succ 0$ auf das explizite Lösen der linearen Matrixgleichungen

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{Q} \tag{12}$$

zurückführen. Bei einer größeren Anzahl von Matrixungleichungen, die sich insbesondere bei der Stabilitätsanalyse von TS Systemen ergeben, ist eine ana-

lytische Lösung nicht mehr gegeben. Ein effizientes Lösungsverfahren für LMIs ist im Jahre 1988 von Nesterov und Nemirovskii [19] basierend auf der Inneren-Punkt-Methode entwickelt worden.

## 3.2 LMI Nebenbedingungen zur Systemanalyse und Synthese

Zunächst wird die Dynamik des fehlertoleranten Regelkreises für Prozesse in der Takagi-Sugeno Form (1) zusammen mit dem Regelungsgesetz (9) untersucht

$$\dot{\mathbf{x}} = \sum_{i=1}^{N_r} h_i(\mathbf{z})\,\mathbf{A}_i\,\mathbf{x} + \mathbf{B}\left[-\sum_{i=1}^{N_r} h_i(\mathbf{z})\mathbf{K}_i\hat{\mathbf{x}} - \hat{\mathbf{f}}\right] + \mathbf{B}_f\,\mathbf{f}\,. \tag{13}$$

Mit der Erweiterung

$$+\sum_{i=1}^{N_r} h_i(\mathbf{z})\mathbf{B}\mathbf{K}_i\mathbf{x} - \sum_{i=1}^{N_r} h_i(\mathbf{z})\mathbf{B}\mathbf{K}_i\mathbf{x}\,,$$

und falls $\mathbf{B}_f = \mathbf{B}$ gilt, genügt die Regelkreisdynamik der Differentialgleichung

$$\dot{\mathbf{x}} = \sum_{i=1}^{N_r} h_i(\mathbf{z})\,[\mathbf{A}_i - \mathbf{B}\mathbf{K}_i]\,\mathbf{x} + \left(\sum_{i=1}^{N_r} h_i(\mathbf{z})\mathbf{B}\mathbf{K}_i \quad \mathbf{0}_{n\times(q-1)k} \quad \mathbf{B}\right)\mathbf{e} \tag{14}$$

mit

$$\mathbf{e} = \begin{pmatrix}\mathbf{e}_x\\ \mathbf{e}_\xi\end{pmatrix}, \quad \mathbf{e}_x = \mathbf{x} - \hat{\mathbf{x}}, \quad \mathbf{e}_\xi = \begin{pmatrix}\boldsymbol{\xi}_1 - \hat{\boldsymbol{\xi}}_1\\ \vdots\\ \boldsymbol{\xi}_q - \hat{\boldsymbol{\xi}}_q\end{pmatrix},$$

basierend auf den Definitionen der erweiterten Zustandsvektoren (4) und (5).

In der weiteren Betrachtung wird der Term in (3) mit der Ableitung $\mathbf{f}^{(q)}$ vernachlässigt ($\mathbf{f}^{(q)} \approx 0$). Die Rekonstruktionsfehler, die sich durch eine nicht ideale Schätzung der Fehler ergeben, sind jedoch in der Fehlerdifferentialgleichung des Beobachters (7) enthalten. Um auch noch die Störung durch $\mathbf{f}^{(q)} \neq 0$

zu berücksichtigen, können die hier hergeleiteten LMIs durch Nebenbedingungen mit $H_\infty$-Kriterien erweitert werden. Dies wird in diesem Aufsatz jedoch nicht betrachtet, aber wird in zukünftigen Arbeiten weiter untersucht.

Durch die Kombination der Fehlerdynamik des Beobachters (7) mit $\mathbf{f}^{(q)} \approx 0$ und der geschlossenen Dynamik des FTC (14) erhält man

$$\dot{\bar{\mathbf{x}}} = \sum_{i=1}^{N_r} h_i(\mathbf{z}) \underbrace{\begin{pmatrix} \mathbf{A}_i - \mathbf{B}\mathbf{K}_i & (\mathbf{B}\mathbf{K}_i & \mathbf{0}_{n\times(q-1)k} & \mathbf{B}) \\ \mathbf{0}_{\tilde{n}\times n} & & \tilde{\mathbf{A}}_i - \tilde{\mathbf{L}}_i\tilde{\mathbf{C}} & \end{pmatrix}}_{:=\bar{\mathbf{A}}_i} \bar{\mathbf{x}} \tag{15}$$

mit

$$\bar{\mathbf{x}} = \begin{pmatrix} \mathbf{x}^T & \mathbf{e}^T \end{pmatrix}^T .$$

Für die Stabilitätsanalyse von (15) wird das folgende Theorem angegeben:

**Theorem 3.1.** *Der geschlossene Regelkreis* (15) *ist asymptotisch stabil, falls eine* $\mathbf{P}_1 \succ 0$, $\mathbf{P}_1 = \mathbf{P}_1^T$ *und* $\mathbf{P}_2 \succ 0$, $\mathbf{P}_2 = \mathbf{P}_2^T$ *für die gegebenen Rückführungsmatrizen* $\mathbf{K}_i$ *und Beobachtermatrizen* $\tilde{\mathbf{L}}_i$, $i = 1,\ldots,N_r$ *existiert, so dass*

$$\begin{pmatrix} \mathbf{H}_{11}(\mathbf{P}_1) & (\mathbf{P}_1\mathbf{B}\mathbf{K}_i & \mathbf{0}_{n\times(q-1)k} & \mathbf{P}_1\mathbf{B}) \\ \begin{pmatrix} \mathbf{K}_i^T\mathbf{B}^T\mathbf{P}_1 \\ \mathbf{0}_{(q-1)k\times n} \\ \mathbf{B}\mathbf{P}_1 \end{pmatrix} & & \mathbf{H}_{22}(\mathbf{P}_2) & \end{pmatrix} \prec 0 \tag{16}$$

*mit*

$$\begin{aligned} \mathbf{H}_{11}(\mathbf{P}_1) &= (\mathbf{A}_i^T - \mathbf{K}_i^T\mathbf{B}^T)\mathbf{P}_1 + \mathbf{P}_1(\mathbf{A}_i - \mathbf{B}\mathbf{K}_i), \\ \mathbf{H}_{22}(\mathbf{P}_2) &= (\tilde{\mathbf{A}}_i^T - \tilde{\mathbf{C}}^T\tilde{\mathbf{L}}_i^T)\mathbf{P}_2 + \mathbf{P}_2(\tilde{\mathbf{A}}_i - \tilde{\mathbf{L}}_i\tilde{\mathbf{C}}) \end{aligned}$$

*für* $i = 1,\ldots,N_r$ *gilt.*

*Beweis:* Mit der quadratischen Funktion als Lyapunov-Funktionskandidat

$$V = \bar{\mathbf{x}}^T \begin{pmatrix} \mathbf{P}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_2 \end{pmatrix} \bar{\mathbf{x}}$$

mit $\mathbf{P}_1 \succ 0$, $\mathbf{P}_1 = \mathbf{P}_1^T$ and $\mathbf{P}_2 \succ 0$, $\mathbf{P}_2 = \mathbf{P}_2^T$ und dessen Ableitung

$$\dot{V} = \dot{\bar{\mathbf{x}}}^T \begin{pmatrix} \mathbf{P}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_2 \end{pmatrix} \bar{\mathbf{x}} + \bar{\mathbf{x}}^T \begin{pmatrix} \mathbf{P}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_2 \end{pmatrix} \dot{\bar{\mathbf{x}}} \,.$$

werden in der rechten Seite von (15) die Ausdrücke $\dot{\bar{\mathbf{x}}}^T$ und $\dot{\bar{\mathbf{x}}}$ ersetzt. Damit erhalten wir die aus der Literatur [20] bekannten Stabilitätsbedingungen für Rückkopplungssysteme in TS Form. Abschließend, nach einigen Umformungen von Matrizenausdrücken, erhält man (16). □

## 3.3 Herleitung von LMI Bedingungen für den integrierten Entwurf

Nach der Analyse der Dynamik der beobachterbasierten FTC im vorhergehenden Abschnitt ($\mathbf{K}_i$ und $\tilde{\mathbf{L}}_i$ werden als gegeben angenommen) werden nun Sätze mit LMI Bedingungen zur FTC Synthese vorgestellt.

**Theorem 3.2.** *Der beobachterbasierte fehlertolerante Regelkreis ist mit einer gemeinsamen Zustands- und Fehlerschätzung ist asymptotisch stabil, falls die Matrizen* $\mathbf{X}_1 \succ 0$, $\mathbf{X}_1 = \mathbf{X}_1^T$, $\mathbf{P}_2 \succ 0$, $\mathbf{P}_2 = \mathbf{P}_2^T$, *und die Matrizen* $\mathbf{M}_i$, $\mathbf{N}_i$ *der* $i = 1, \ldots, N_r$ *Teilsysteme existieren, so dass*

$$\underbrace{\begin{pmatrix} \mathbf{G}_{11}(\mathbf{X}_1) & (\mathbf{B}\mathbf{K}_i \quad \mathbf{0}_{n\times(q-1)k} \quad \mathbf{B}) \\ \begin{pmatrix} \mathbf{K}_i^T\mathbf{B}^T\mathbf{P}_1 \\ \mathbf{0}_{(q-1)k\times n} \\ \mathbf{B} \end{pmatrix} & \mathbf{G}_{22}(\mathbf{P}_2) \end{pmatrix}}_{=:\mathbf{G}} \prec 0 \tag{17}$$

*mit*

$$\mathbf{G}_{11}(\mathbf{X}_1) = \mathbf{X}_1\mathbf{A}_i^T + \mathbf{A}_i\mathbf{X}_1 - \mathbf{M}_i^T\mathbf{B}^T - \mathbf{B}\mathbf{M}_i\,,$$
$$\mathbf{G}_{22}(\mathbf{P}_2) = \tilde{\mathbf{A}}_i^T\mathbf{P}_2 + \mathbf{P}_2\tilde{\mathbf{A}}_i - \tilde{\mathbf{C}}^T\mathbf{N}_i^T - \mathbf{N}_i\tilde{\mathbf{C}}$$

*für $i = 1,\ldots,N_r$ gilt.*

*Beweis:* Unter Ausnutzung der Kongruenzeigenschaft [20]

$$\mathbf{G} \prec 0 \quad \Longleftrightarrow \quad \mathbf{Q}\mathbf{G}\mathbf{Q}^T \prec 0$$

mit Matrizen

$$\mathbf{Q} = \begin{pmatrix} \mathbf{P}_1^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix},$$

die einen vollen Rang besitzen, erhalten wir (17) wobei $\mathbf{X}_1 := \mathbf{P}_1^{-1}$, $\mathbf{M}_i = \mathbf{K}_i\mathbf{X}_1$ und $\mathbf{N}_i = \mathbf{P}_2\tilde{\mathbf{L}}_i$. □

Damit erfolgt der Beobachterentwurf zur Fehlerrekonstruktion mit integrierter fehlertoleranter Regelung für TS Systeme der Form (1) durch das Aufstellen der linearen Matrixungleichungen aus (17). Falls es eine Lösung gibt, also die Matrizen $\mathbf{X}_1 \succ 0$, $\mathbf{X}_1 = \mathbf{X}_1^T$, $\mathbf{P}_2 \succ 0$, $\mathbf{P}_2 = \mathbf{P}_2^T$, $\mathbf{M}_i$ und $\mathbf{N}_i$ für $i = 1,\ldots,N_r$ die Ungleichung (17) erfüllen, kann damit das Regelgesetz (9) mit

$$\mathbf{K}_i = \mathbf{M}_i\mathbf{X}_1^{-1}\,, \tag{18}$$

und der PMI Beobachter (6) mit

$$\mathbf{L}_i = \mathbf{P}_2^{-1}\mathbf{N}_i \tag{19}$$

aufgestellt werden.

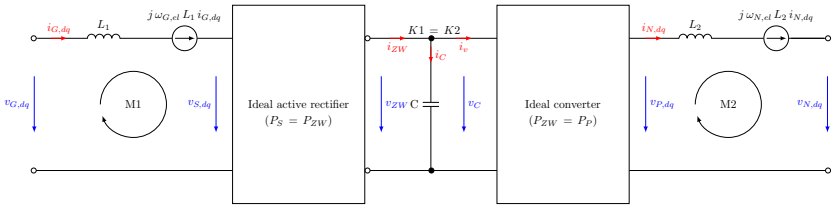



Bild 1: Elektrisches Ersatzschaltbild von AC-DC-AC Umrichtern

# 4 Fallbeispiel: Fehlerrekonstruktion und fehlertolerante Regelung von AC-DC-AC Frequenzumrichtern

In diesem Abschnitt werden die notwendigen Schritte für den vorgestellten Reglerentwurf am Beispiel der modellbasierten d-q Stromregelung für leistungselektronische Umrichter diskutiert. Zunächst wird ein geeignetes Prozessmodell für AC-DC-AC Umrichter in Takagi-Sugeno Form aus [23] eingeführt. Danach wird in den Grundzügen die Anwendung der in dem vorhergehenden Abschnitt vorgestellte Entwurfsmethode beschrieben.

## 4.1 Prozessmodell des Frequenzumrichters

Das Ersatzschaltbild des zu regelnden AC-DC-AC Umrichters ist in dem Bild 1 dargestellt. Es besteht aus drei zu modellierenden Teilsystemen: (1) dem generatorseitigen AC-DC Umrichter (ideal active rectifier), (2) dem DC-Zwischenkreis und (3) dem netzseitigen DC-AC Umrichter (ideal converter). Dabei werden die Verluste in den elektronischen Bauteilen und die der Grundwelle überlagerten Harmonischen, welche durch das Schalten der Leistungshalbleiter entstehen und durch netzseitige passive Filter nicht ideal bezogen auf den Netzeinspeisepunkt gefiltert werden können, vernachlässigt. Die Modellierung erfolgt daher nur für die AC Grundwellen des Drehstromsystems, das in rotierende d-q Koordinaten mittels Park-Clarke Transformation beschrieben wird.

### 4.1.1 Generatorseitiges Umrichtermodell (AC-DC)

Der generatorseitige Umrichter wird als idealer, aktiver Gleichrichter ohne Verluste in den Leistungshalbleitern und ohne Harmonische unter Ausnutzung des model-averaging Verfahrens [21], [22] modelliert und durch folgende Gleichungen beschrieben:

$$\frac{d}{dt} i_{G,d} = \frac{1}{L_1} v_{G,d} - \frac{1}{L_1} v_{S,d} + \omega_G i_{G,q} \tag{20a}$$

$$\frac{d}{dt} i_{G,q} = \frac{1}{L_1} v_{G,q} - \frac{1}{L_1} v_{S,q} - \omega_G i_{G,d} \tag{20b}$$

$$\dot{v}_c = \frac{3}{2C} \frac{v_{S,d} i_{G,d}}{v_C} + \frac{3}{2C} \frac{v_{S,q} i_{G,q}}{v_C} - \frac{1}{C} i_v \tag{20c}$$

mit $L_1$ als Statorinduktivität des Generators, $\omega_G$ als elektrische Kreisfrequenz des Generators, $C$ als die Kapazität des Spannungszwischenkreises (vgl. Bild 1). Die Größen $i_{G,d}$, $i_{G,q}$ repräsentieren die Statorströme in d-q Koordinaten, $v_{G,d}$, $v_{G,q}$ die Statorspannungen des Generators, $v_{S,d}$, $v_{S,q}$ die Eingangsspannungen des Umrichters auf der Generatorseite sowie $i_V$, $v_C$ als Strom und Spannung im Zwischenkreis.

### 4.1.2 Netzseitiges Umrichter Modell (DC-AC)

Der netzseitige Teil des Frequenzumrichters wird ebenfalls als ein idealer Umrichter für das elektrische Netz mit der Grundfrequenz $\omega_N$ ohne die Erzeugung zusätzlicher harmonischer Signalanteile betrachtet und wie folgt modelliert:

$$\frac{d}{dt} i_{N,d} = \frac{1}{L_2} v_{P,d} - \frac{1}{L_2} v_{N,d} + \omega_N i_{N,q} \tag{21a}$$

$$\frac{d}{dt} i_{N,q} = \frac{1}{L_2} v_{P,q} - \frac{1}{L_2} v_{N,q} - \omega_N i_{N,d} \tag{21b}$$

$$\dot{v}_c = \frac{1}{C} i_{zw} - \frac{3}{2C} \frac{v_{P,d} i_{N,d}}{v_C} - \frac{3}{2C} \frac{v_{P,q} i_{N,q}}{v_C} \tag{21c}$$

wobei $L_2$ die Ersatzinduktion des netzseitigen LCL Filters beschreibt. Der Einfluss der weiteren C und L Komponenten werden dabei nicht berücksichtigt. Diese können jedoch bei Bedarf mittels Systemerweiterung noch in die Modellierung einfließen. Die Größen $i_{N,d}$, $i_{N,q}$ sowie $v_{N,d}$, $v_{N,q}$ in (21) repräsentieren

die netzseitigen elektrischen Ströme und Spannungen der Umrichters in d-q Koordinaten. Die Ausgangsspannungen der Netzseite werden mit $v_{P,d}$ und $v_{P,q}$ bezeichnet.

### 4.1.3 Zusammenfassung der Teilmodelle in ein Zustandsraummodell

Zur Vorbereitung des zentralen Zustandsraummodells werden nun die beiden Teilmodelle (20), (21) zu einem AC-DC-AC Frequenzumrichtermodell zusammengefasst

$$\frac{d}{dt} i_{G,d} = \frac{1}{L_1} v_{G,d} - \frac{1}{L_1} v_{S,d} + \omega_G i_{G,q} \tag{22a}$$

$$\frac{d}{dt} i_{G,q} = \frac{1}{L_1} v_{G,q} - \frac{1}{L_1} v_{S,q} - \omega_G i_{G,d} \tag{22b}$$

$$\begin{aligned} \dot{v}_c = & \frac{3}{2C} \frac{v_{S,d} i_{G,d}}{v_C} + \frac{3}{2C} \frac{v_{S,q} i_{G,q}}{v_C} \\ & - \frac{3}{2C} \frac{v_{P,d} i_{N,d}}{v_C} - \frac{3}{2C} \frac{v_{P,q} i_{N,q}}{v_C} \end{aligned} \tag{22c}$$

$$\frac{d}{dt} i_{N,d} = \frac{1}{L_2} v_{P,d} - \frac{1}{L_2} v_{N,d} + \omega_N i_{N,q} \tag{22d}$$

$$\frac{d}{dt} i_{N,q} = \frac{1}{L_2} v_{P,q} - \frac{1}{L_2} v_{N,q} - \omega_N i_{N,d} \tag{22e}$$

Die Modellgleichungen der Form (22) werden nun in einem Zustandsraummodell zusammengefasst

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}(\mathbf{x}, \mathbf{u})\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) + \mathbf{B_d}\,\mathbf{d}(t) \\ \mathbf{y}(t) &= \mathbf{x}(t) \end{aligned} \tag{23}$$

mit dem Zustandsvektor $\mathbf{x} \in \mathbb{R}^n$, den Eingängen (steuerbar) $\mathbf{u} \in \mathbb{R}^m$ und den Störungen $\mathbf{d} \in \mathbb{R}^d$, welche messbar aber nicht steuerbar sind:

$$\mathbf{x} = \begin{pmatrix} i_{G,d} \\ i_{G,q} \\ v_C \\ i_{N,d} \\ i_{N,q} \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} v_{S,d} \\ v_{S,q} \\ v_{P,d} \\ v_{P,q} \end{pmatrix}, \quad \mathbf{d} = \begin{pmatrix} v_{G,d} \\ v_{G,q} \\ v_{N,d} \\ v_{N,q} \end{pmatrix}$$

mit

$$\mathbf{A}(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} 0 & \omega_G & 0 & 0 & 0 \\ -\omega_G & 0 & 0 & 0 & 0 \\ \frac{3}{2C}\frac{u_1}{x_3} & \frac{3}{2C}\frac{u_2}{x_3} & 0 & -\frac{3}{2C}\frac{u_3}{x_3} & -\frac{3}{2C}\frac{u_4}{x_3} \\ 0 & 0 & 0 & 0 & \omega_N \\ 0 & 0 & 0 & -\omega_N & 0 \end{pmatrix}, \tag{24}$$

$$\mathbf{B} = \begin{pmatrix} -\frac{1}{L_1} & 0 & 0 & 0 \\ 0 & -\frac{1}{L_1} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{L_2} & 0 \\ 0 & 0 & 0 & \frac{1}{L_2} \end{pmatrix}, \quad \mathbf{C} = \mathbf{I},$$

$$\mathbf{B_d} = \begin{pmatrix} \frac{1}{L_1} & 0 & 0 & 0 \\ 0 & \frac{1}{L_1} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{L_2} & 0 \\ 0 & 0 & 0 & -\frac{1}{L_2} \end{pmatrix},$$

wobei der Wertebereich von $x_3$ wie folgt festgelegt wird

$$x_3 \quad = \quad \begin{cases} x_3 > 0 \ , & x_3 \\ x_3 \leq 0 \ , & x_{3,max} \cdot 10^{-3} \end{cases} \tag{25}$$

mit $x_{3,max} = \max(x_3(t))\forall t$ um die Unstetigkeitsstelle bei **A** im Fall eines spannungslosen Zwischenkreises (keine Energie im Kondensator gespeichert) auszuschließen.

## 4.2 Zustandsraummodell in TS-Form

Zur Vorbereitung des LMI-basierten Entwurfs wird nun (23) in ein äquivalentes Takagi-Sugeno Modell überführt. Für eine exakte Beschreibung des nichtlinearen Umrichtermodells als TS-System wird für den erwarteten Arbeitsbereich (Großsignalverhalten) die Methode der Sektornichtlinearitäten [20] angewandt. Hierzu werden im ersten Schritt die statischen Funktionen

$$\begin{array}{rclrcl} \gamma_1(\mathbf{z}) & = & \frac{u_1}{x_3} & \gamma_2(\mathbf{z}) & = & \frac{u_2}{x_3} \\ \gamma_3(\mathbf{z}) & = & -\frac{u_3}{x_3} & \gamma_4(\mathbf{z}) & = & -\frac{u_4}{x_3} \end{array} \tag{26}$$

mit $\mathbf{z} = (x_3, \mathbf{u})$. eingeführt, um die entsprechenden Einträge in (24) zu ersetzen. Dies führt auf eine **z**-variable Systemmatrix

$$\mathbf{A}(\mathbf{z}) = \begin{pmatrix} 0 & \omega_G & 0 & 0 & 0 \\ -\omega_G & 0 & 0 & 0 & 0 \\ \frac{3}{2C}\gamma_1(\mathbf{z}) & \frac{3}{2C}\gamma_2(\mathbf{z}) & 0 & \frac{3}{2C}\gamma_3(\mathbf{z}) & \frac{3}{2C}\gamma_4(\mathbf{z}) \\ 0 & 0 & 0 & 0 & \omega_N \\ 0 & 0 & 0 & -\omega_N & 0 \end{pmatrix} . \tag{27}$$

Im zweiten Schritt werden die Funktionen (26) durch die äquivalenten Sektorfunktionen

$$\gamma_j(\mathbf{z}) = w_{j,1}\,\overline{\gamma}_j + w_{j,2}\,\underline{\gamma}_i \tag{28}$$

für alle $j = 1,...,N_l$ mit $N_l = 4$ ersetzt. Diese enthalten die Gewichtsfunktionen

$$w_{j,1}(\mathbf{z}) = \frac{\gamma_j(\mathbf{z}) - \underline{\gamma}_j}{\overline{\gamma}_j - \underline{\gamma}_j}, \qquad w_{j,2}(\mathbf{z}) = \frac{\overline{\gamma}_j - \gamma_i(\mathbf{z})}{\overline{\gamma}_j - \underline{\gamma}_j}, \tag{29}$$

die den Anteil der konstanten oberen $\overline{\gamma}_i$ und unteren Schranke $\underline{\gamma}_i$ gewichten. Die Festlegung der Schranken ergeben sich aus den physikalischen Randbedingungen und den Leistungsparametern des Frequenzumrichters. In diesem Fall sind es die Beschränkungen der generatorseitigen und netzseitigen Eingangsspannungen $u_i$ für $i = 1,\ldots,4$ und der maximalen unteren und oberen Zwischenkreisspannung $x_3 := v_C$ mit

$$\begin{aligned} u_i &= [\,-1000V\,,\,1000V\,]\,, \qquad i = 1,\ldots,4\,, \\ x_3 &= [\,1V\,,\,1000V\,]\,. \end{aligned}$$

Daraus ergeben sich die Schranken der Sektorfunktionen (28) zu

$$\begin{aligned} \overline{\gamma}_i(\mathbf{z}) &= \frac{\overline{u}_i}{\underline{x}_3} = -\frac{\underline{u}_i}{\underline{x}_3} = 1000\,, \\ \underline{\gamma}_i(\mathbf{z}) &= \frac{\underline{u}_i}{\underline{x}_3} = -\frac{\overline{u}_i}{\underline{x}_3} = -1000\,. \end{aligned} \tag{30}$$

Aus (4.2) folgt direkt, dass die konvexe Summenbedingung auch für die Gewichtsfunktionen

$$w_{j1} + w_{j2} = 1\,, \qquad j = 1,\ldots,4 \tag{31}$$

erfüllt ist. Mit (31) und der Konstruktionsvorschrift der Zugehörigkeitsfunktionen (2) als Produkt der Gewichtsfunktionen $w_{jk}$ [5] folgt die Beziehung

$$\sum_{i=1}^{N_r} h_i(\mathbf{z}) = \prod_{j=1}^{N_l=4} (w_{j1}(\mathbf{z}) + w_{j2}(\mathbf{z})) \tag{32}$$

mit $N_r = 2^{N_l}$, womit die konvexe Summenbedingung (2) für die Zugehörgkeitsfunktionen $h_i(\mathbf{z})$ erfüllt ist. Damit erhält man für $N_l = 4$ eine konvexe gewichtete Kombination aus $N_r = 2^{N_l} = 16$ Teilmodellen mit

$$\begin{aligned} h_1 &= w_{11}\, w_{21}\, w_{31}\, w_{41} \\ &\vdots \\ h_{16} &= w_{12}\, w_{22}\, w_{32}\, w_{42} \end{aligned} \tag{33}$$

Zugehörigkeitsfunktionen. Basierend auf der Kombinatorik in (33) werden die konstanten Systemmatrizen $\mathbf{A}_i$ aus den oberen und unteren Schranken $\overline{\gamma}_i$ und $\underline{\gamma}_i$ mit (30) gebildet. Mittels der Vorarbeit erhalten wir das nominale Modell des Frequenzumrichters basierend auf (23) in TS Form

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \sum_{i=1}^{N_r=16} h_i(\mathbf{z}(\mathbf{t}))\,\mathbf{A}_i\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) + \mathbf{B_d}\,\mathbf{d}(t)\,, \\ \mathbf{y}(t) &= \mathbf{x}(t) \qquad . \end{aligned} \tag{34}$$

## 4.3 Zustandsraummodell in TS-Form mit Aktuatorfehlern

Zur Überprüfung der Leistungsfähigkeit des PMI Beobachters zur Fehlerrekonstruktion wird (34) um einen Aktuatorfehler erweitert

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \sum_{i=1}^{N_r=16} h_i(\mathbf{z}(\mathbf{t}))\,\mathbf{A}_i\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) + \mathbf{B_d}\,\mathbf{d}(t) + \mathbf{B_f}\mathbf{f}(t)\,, \\ \mathbf{y}(t) &= \mathbf{x}(t) \end{aligned} \tag{35}$$

mit $\mathbf{f} = [\ f_1\ f_2\ f_3\ f_4\ ]^T \in \mathbb{R}^4$ wobei $\mathbf{B_f} = \mathbf{B}$, d.h. die Spannungsfehler in d-q Koordinaten wirken direkt auf die Eingangsspannungen. Bei den prinzipiellen Untersuchungen betrachten wir zunächst synthetische sprung- und rampenförmige Fehlerverläufe auf der Netzseite, die jeweils ab $t_1 = 0.1\,\mathrm{s}$ auftreten:

1. Fehlertyp: $f_3(t \geq t_1) = -20\,[\mathrm{V}]$ und $f_4(t \geq t_1) = -53\,[\mathrm{V}]$
2. Fehlertyp: $\dot{f}_3(t \geq t_1) = -40\,[\mathrm{V/s}]$ und $\dot{f}_4(t \geq t_1) = 60\,[\mathrm{V/s}]$

Die Generatorseite des Umrichters wird als fehlerfrei angenommen, d.h. $f_1(t) = f_2(t) = 0\ \forall t$.

Anmerkung: In der Praxis werden diese Art der Fehlerverläufe in den d-q-Koordinaten nicht auftreten, weil Spannungsfehler (Spannungseinbrüche) üblicherweise durch Kurzschlüsse im Drehstromsystem entstehen und aufgrund der nichtlinearen Park-Clark Transformation andere Verläufe annehmen wird. Für eine erste Untersuchung der stationären und dynamischen Rekonstruktionsgüte sind diese Testsignale jedoch gut geeignet.

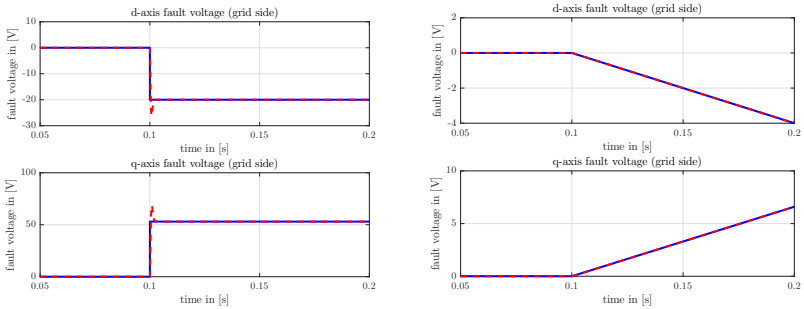



Bild 2: Vergleich der wahren sprung- und rampenförmigen Fehler $f_{3,4}$ (blau) mit den rekonstruierten Fehlern $\hat{f}_{3,4}$ (rot)

### 4.4 Simulationsergebnisse zur Fehlerrekonstruktion

Die Simulationsergebnisse zur Rekonstruktion vom Fehlertyp 1 und 2 sind in Bild 2 dargestellt. Die hierbei eingesetzte PMI Beobachter (6) enthält eine Integratorkette von $q = 2$, d.h. der Fehlervektor und die erste Ableitung werden geschätzt. Der erweiterte Zustandsvektor lautet hierfür:

$$\tilde{\mathbf{x}} = \left(\mathbf{x}^T \, , \, \boldsymbol{\xi}_1^T \, , \, \boldsymbol{\xi}_2^T\right)^T$$

mit $\boldsymbol{\xi}_1 = \dot{\mathbf{f}}$ und $\boldsymbol{\xi}_2 = \mathbf{f}$.

Die ausreichende Güte der Rekonstruktion der Spannungsfehler ist in allen vier Abbildungen (vgl. Bild 2) gut zu erkennen. Bei dem sprungförmigen Fehlersignal (Fehlertyp 1) sieht man ein leichtes Überschwingen der Signale $\hat{f}_{3,4}$ (rot) gegenüber den Aktuatorfehlern $f_{3,4}$ (blau). Auch bei den rampenförmigen Fehlern (Fehlertyp 2) werden die Verläufe gut reproduziert.

## 5 Zusammenfassung und Ausblick

In diesem Beitrag wurde ein Entwurf zur fehlertoleranten Regelung für eine Klasse von Takago-Sugeno Fuzzy Systemen vorgestellt. Die Formulierung der Anforderungen an den Regelkreis erfolgte mit Hilfe linearer Matrixungleichungen. Die Sätze zur Systemanalyse und Synthese wurden erläutert und

bewiesen. Die Anwendbarkeit und Leistungsfähigkeit des Entwurfsverfahrens wurde am Beispiel eines AC-DC-AC Frequenzumrichters gezeigt. Simulationen zur Rekonstruktion verschiedener Fehlerverläufe wurden vorgestellt. Experimentelle Untersuchungen zur Güte der fehlertoleranten Regelung werden in zukünftigen Arbeiten an dem an der HTW Berlin aufgebauten Prüfstand zur Untersuchung von Kraftwerkseigenschaften von Windenergieanlagen [24] durchgeführt.

## Literatur


[1] R. J. Patton, "Fault-tolerant control systems: The 1997 situation," In *Proc. IFAC Symp. Fault Detection Supervision Safety Tech. Processes*, vol. 3, 1997, pp. 1033–1054.

[2] Y. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," *Annu. Rev. Control*, vol. 32, no. 2, pp. 229–252, 2008.

[3] M. Saif and Y. Guan, "A New Approach to Robust Fault Detection and Identification," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 29, no. 3, pp. 685–695, 1993.

[4] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, *Diagnosis and Fault-Tolerant Control*, 2nd ed. Springer-Verlag Berlin Heidelberg, 2006.

[5] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Application to Modeling and Control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, no. 1, pp. 116–132, 1985.

[6] K. Zhang, B. Jiang, and M. Staroswieck, "Dynamic output feedback-fault tolerant controller design for Takagi-Sugeno fuzzy systems with actuator faults," *IEEE Transaction on Fuzzy Systems*, vol. 18, no. 1, pp. 194–201, Feb. 2010.

[7] D. Ichalal, B. Marx, J. Ragot, and D. Maquin, "New fault tolerant control strategies for nonlinear Takagi-Sugeno systems," *Int. J. Appl. Math. Comput. Sci.*, vol. 22, no. 1, pp. 197–210, 2012.

[8] R. H. Middleton and G. C. Goodwin , “Digital Control and Estimation: A unified approach,” Prentice Hall, New Jersey, 1990

[9] H. Yang, P. Shi, X. Li, and Z. L. ., “Fault-tolerant control for a class of T-S fuzzy systems via delta operator approach,” *Signal Processing*, vol. 98, pp. 166–173, 2014.

[10] M. Witczak, L. Dziekan, V. Puig, and J. Korbicz, “Design of a fault-tolerant control scheme for Takagi-Sugeno fuzzy systems,” In *Proc. 16th Mediterranean Conf. Control Autom.*, 2008, pp. 280–285.

[11] J. Lan and R. J. Patton, “Integrated design of fault-tolerant control for nonlinear systems based on fault estimation and T–S fuzzy modeling,” *IEEE Transaction on Fuzzy Systems*, vol. 25, no. 5, pp. 1141–1154, Oct. 2017.

[12] M. Liu, X. Cao, and P. Shi, “Fault estimation and tolerant control for fuzzy stochastic systems,” *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 2, pp. 221–229, Apr. 2013.

[13] ——, “Fuzzy-model-based fault-tolerant design for nonlinear stochastic systems against simultaneous sensor and actuator faults,” *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 5, pp. 789–799, Oct. 2013.

[14] P. Gerland, D. Groß, H. Schulte, and A. Kroll, “Design of Sliding Mode Observers for TS Fuzzy Systems with Application to Disturbance and Actuator Fault Estimation,” In *IEEE Conference on Decision and Control*, Hilton Atlanta Hotel, Atlanta, GA, USA, 2010, pp. 4373–4378.

[15] Z. Gao, S.X. Ding , Y. Ma, “Robust fault estimation approach and its application in vehicle lateral dynamic systems“ *Optimal Control Applications and Methods*, **28**:143–156, 2007.

[16] P. Kühne, F. Pöschke, and H. Schulte, “Fault estimation and fault-tolerant control of the FAST NREL 5-MW reference wind turbine using a proportional multi-integral observer,” *International Journal of Adaptive Control and Signal Processing,*, vol. 32, no. 4, pp. 568-585, April 2018.

[17] Z. Lendek, T. M. Guerra, R. Babuška, and B. De Schutter, *Stability Analysis and Nonlinear Observer Design Using Takagi-Sugeno Fuzzy Models* Springer-Verlag Berlin Heidelberg, 2010.

[18] A. M. Lyapunov, *The general problem of the stability of motion*. International Journal of Control, 55(3), 531–773.

[19] Y. Nesterov and A. Nemirovsky. A general approach to polynomial-time algo- rithms design for convex programming. *Technical report, Centr. Econ. & Math. Inst., USSR Acad. Sci.*, Moscow, USSR, 1988.

[20] Z. Lendek, T. M. Guerra, R. Babuška, and B. De Schutter, *Stability Analysis and Nonlinear Observer Design Using Takagi-Sugeno Fuzzy Models*. Springer-Verlag Berlin Heidelberg, 2010.

[21] R. Teodorescu, M. Liserre, and P. Rodriguez, *Grid converters for photovoltaic and wind power systems* John Wiley & Sons, 2011, vol. 29.

[22] F. Vasca and L. Iannelli, *Dynamics and control of switched electronic systems: Advanced perspectives for modeling, simulation and control of power converters* Springer, 2012.

[23] N. Goldschmidt, S. Betker, and H. Schulte: “Generalized Reduced Order Modeling of AC-DC-AC Converters with Application to Fault Diagnosis“ In *Proceedings of 16th Wind Integration Workshop*, Berlin, 2017, ISBN 978-3-9816549-6-7

[24] A. Kisser, M. Engel, L. Rezai, M. Andrejewski, J. Fortmann, and H. Schulte, “A Test-bed System for Validation of Ancillary Services of Wind Farms under Realistic Conditions“ In *Proceedings of 16th Wind Integration Workshop*, Berlin, 2017, ISBN 978-3-9816549-6-7

# Zum optimalen Offline-Testsignalentwurf für die Identifikation dynamischer TS-Modelle: Steuerfunktionen zur optimalen Schätzung der Partitionsparameter

Matthias Gringard, Andreas Kroll

Fachgebiet Mess- und Regelungstechnik,
Institute for System Analytics and Control,
Fachbereich Maschinenbau,
Universität Kassel
E-Mail: {matthias.gringard,andreas.kroll}@mrt.uni-kassel.de

## 1 Einführung

Die Systemidentifikation hat sich zu den Methoden der theoretischen Modellbildung aufgrund zunehmender Komplexität sowie erforderlichen Expertenwissens als effektive und effiziente Alternative etabliert. Um mit den Methoden der Systemidentifikation qualitativ hochwertige Modelle generieren zu können, müssen im Wesentlichen drei Kernaspekte berücksichtigt werden. Diese Aspekte sind die Auswahl der Modellklasse, die zur Identifikation verwendeten Daten sowie die Algorithmen, welche die Modelle aus den Daten schätzen. Das Thema dieses Beitrags ist die Generierung geeigneter Daten. Der Testsignalentwurf ist eine Möglichkeit die Eigenschaften der Identifikationsdaten zu beeinflussen.

Eine wichtige Kategorisierung der Testsignalentwürfe ist die Unterscheidung zwischen prozessmodellfreien und prozessmodellbasierten Entwürfen [1]. Bei prozessmodellfreien Testsignalentwürfen werden Signalkenngrößen zum Entwurf der Eingangsgröße sowie zur Bewertung der Ausgangsgröße eingesetzt. Im Gegensatz dazu werden bei prozessmodellbasierten Testsignalentwürfen neben dem System auch die Modellgleichungen berücksichtigt. Ein Beispiel

dafür ist die für statische Probleme genutzte Fisher-Informationsmatrix (FIM) [2], welche auch auf dynamische Systeme angewendet wird [3]. Es konnte bereits in [4] gezeigt werden, dass sich die Varianzen der geschätzten Parameter lokal-affiner Fuzzy-Takagi-Sugeno-(TS-)Modelle durch einen FIM-basierten Entwurf reduzieren lassen. Neben der Reduktion der Varianz ist auch die Schätzung der Parameter mit einer geringen systematischen Abweichung („Bias“) von großer Bedeutung.

Bei lokal-affinen TS-Modellen ist das nichtlineare Verhalten vollständig in den Fuzzy-Basisfunktionen kodiert. Systematische Abweichungen der lokalen Teilmodellparameter sind also an die der Partitionierungsparameter gekoppelt. Um das nichtlineare Verhalten identifizieren zu können, ist es notwendig, im Bereich der Teilmodellgrenzen Daten zu generieren, da die Sensitivität des Modellausgangs bezüglich Änderungen in den Partitionsparametern nur dort signifikant ist [5]. Es ist daher erstrebenswert, den Zustand dynamischer Systeme entlang dieser Teilmodellgrenzen zu führen. Um zu untersuchen, ob sich ein Pfad gemäß [1] auch zur Identifikation dynamischer Systeme eignet, muss zunächst ein Weg gefunden werden, dem System diesen Pfad aufzuzwingen, was keine triviale Aufgabe darstellt. Der gewünschte Pfad muss auch eine differentielle Kopplung einhalten und das zur Steuerung verwendete Modell basiert auf prozessmodellfreien Identifikationen, was insgesamt in einem iterativen Prozess mündet.

In Abschnitt 2 wird zunächst die verwendete Modellklasse und der Identifikationsprozess vorgestellt. Im Anschluss daran wird in Abschnitt 3 die grundlegende Idee präsentiert, bevor in Abschnitt 4 der Entwurfsprozess aufgegriffen wird. Bevor der Beitrag mit einer Zusammenfassung und einem Ausblick abgeschlossen wird, werden qualitative Erkenntnisse anhand einer Simulationsfallstudie präsentiert.

## 2 Modellklasse und Identifikationsprozess

Es werden zeitdiskrete SISO-TS-Modelle verwendet. Die $i$-te Teilmodellgleichung ist gegeben als:

$$\hat{y}_i(k+n) = c_i - \sum_{l=0}^{n-1} a_{i,l} \cdot y(k+l) + \sum_{l=0}^{m} b_{i,l} \cdot u(k+l-\tau) \tag{1}$$

Hierbei ist $k$ die diskrete Zeit, $a_{i,l}$ stellen die Koeffizienten des Ausgangs $y$, $b_{i,l}$ die Koeffizienten des Eingangs $u$ sowie $n$ und $m$ die dynamischen Ordnungen des Aus- und Eingangs dar. Auf die Totzeit $\tau$ wird im Folgenden verzichtet. Mit Hilfe der Matrixschreibweise kann (1) geschrieben werden als:

$$\begin{aligned}
\hat{y}_i(k+n) &= \begin{bmatrix} 1 & y(k) & \cdots & y(k+n-1) & u(k) & \cdots & u(k+m) \end{bmatrix} \\
&\quad \cdot \begin{bmatrix} c_i & -a_{i,0} & \cdots & -a_{i,n-1} & b_{i,0} & \cdots & b_{i,m} \end{bmatrix}^\top \\
&= \begin{bmatrix} 1 & \boldsymbol{\varphi}_y^\top(k) & \boldsymbol{\varphi}_u^\top(k) \end{bmatrix} \cdot \begin{bmatrix} c_i & \boldsymbol{\Theta}_{i,y}^\top & \boldsymbol{\Theta}_{i,u}^\top \end{bmatrix}^\top \\
&= \boldsymbol{\varphi}^\top \boldsymbol{\Theta}_{\mathrm{LM},i}
\end{aligned} \tag{2}$$

$\boldsymbol{\varphi}$ und $\boldsymbol{\Theta}_{\mathrm{LM},i}$ sind Regressions- und lokaler Parametervektor. Die lokalen Modelle (2) werden mit den zugehörigen von der Schedulingvariable $\mathbf{z}(k)$ abhängigen Fuzzy-Basisfunktionen $\phi_i(\mathbf{z}(k))$ gewichtet und zum Gesamtmodell überlagert. Die Fuzzy-Basisfunktionen sind eine gewichtete Überlagerung der Zugehörigkeitsfunktionen $\mu_i(\mathbf{z}(k))$:

$$\phi_i(\mathbf{z}(k)) = \frac{\mu_i(\mathbf{z}(k))}{\sum_{j=1}^{c} \mu_j(\mathbf{z}(k))} \tag{3}$$

mit

$$\mu_i(\mathbf{z}(k)) = \left[ \sum_{j=1}^{c} \left( \frac{|\mathbf{z}(k) - \mathbf{v}_i|}{|\mathbf{z}(k) - \mathbf{v}_j|} \right)^{\frac{2}{\nu-1}} \right]^{-1} \tag{4}$$

Bei Verwendung von FCM-Zugehörigkeitsfunktionen wie in (4) gilt wegen $\sum_{j=1}^{c} \mu_j(\mathbf{z}(k)) = 1$ auch $\mu_i(\mathbf{z}(k)) = \phi_i(\mathbf{z}(k))$.

Der Identifikationsprozess besteht aus vier Schritten. Mittels Vorwissen oder Selektionsverfahren [6] werden die dynamischen Ordnungen $n, m$, die Anzahl der Teilmodelle $c$, sowie der Unschärfeparamter $\nu$ bestimmt. Durch bspw. eine Clusterung im Schedulingraum $\mathbf{z}(k)$ ergeben sich die $c$ initialen Prototypen $\mathbf{v}_i$. Sie sind in im Fall der verwendeten Zugehörigkeitsfunktionen (4) gleich den Partitionsparametern $\boldsymbol{\Theta}_{\mathrm{MF},i}$. Wenn sie bekannt sind, reduziert sich die Minimierung des quadratischen Prädiktionsfehlers auf ein lineares Least-Squares-Problem, dessen Lösung explizit angegeben werden kann. Im letzten Schritt werden die lokalen Modellparameter $\boldsymbol{\Theta}_{\mathrm{LM},i}$ sowie die Partitionsparameter $\boldsymbol{\Theta}_{\mathrm{MF},i}$ parallel bezüglich des zu minimierenden mittleren quadratischen Simulationsfehlers optimiert.

# 3 Motivation, Lösungsansatz und Problemstellung

In [5] wurde im Rahmen eines optimalen Eingangsgrößenentwurfs eines statischen Systems mittels der Fisher-Informationsmatrix (FIM) gezeigt, dass zur Minimierung der Schätzvarianz der Partitionsparameter eines statischen TS-Modells die Punkte im Eingangsgrößenraum, welche in diesem Beispiel gleich dem Schedulingraum waren, in Richtung der Teilmodellgrenzen zu verlegen sind. Dies ist dadurch begründet, dass die FIM unter bestimmten Rauschannahmen aus den nach den Parametern abgeleiteten Modellgleichungen aufgebaut ist und die Sensitivität bezüglich der Teilmodellparameter in der Nähe der Teilmodellzentren sehr gering ist. Diese geringe Sensitivität bedeutet wiederum eine hohe Unsicherheit, da Schwankungen in den Partitionsparametern eine verschwindende Auswirkung auf den Ausgang haben.

## 3.1 Motivation

Die Generierung von Daten in den Teilmodellgrenzbereichen hat eine weitere Auswirkung. Da das nichtlineare Verhalten eines Systems bei der Approximation durch TS-Modelle ausschließlich in den Partitionsparametern kodiert ist, wird angenommen, dass unter Anderem die Generierung von Daten im Grenz-

bereich zu einer Reduktion der systematischen Abweichung beiträgt. Es wird angenommen, dass durch prozessmodellfrei entworfene Experimente Modelle das nichtlineare Verhalten so erfasst haben, dass ein modellbasierter Entwurf möglich ist.

## 3.2 Problemstellung

Um nun die systematische Abweichung in den Partitionsparametern bei dynamischen Systemen zu reduzieren, wird vorgeschlagen, die Grenzbereiche im Schedulingraum abzufahren. Beim Übergang auf dynamische Systeme ergeben sich unmittelbar Probleme:

1. Die Punkte im Schedulingraum sind nicht mehr voneinander unabhängig, sondern sind Teil einer Trajektorie.
2. Der Schedulingraum besteht nicht mehr nur aus den Eingangsgrößen sondern zusätzlich aus den Ausgangsgrößen.
3. Die Schedulingvariablen sind differentiell gekoppelt.

Das erste Problem ist praktisch das Standardproblem bei jeder Übertragung einer Methode von statischen auf dynamische Systeme. Punkte im Schedulingraum, welche aus statischer Sicht eine gute Abtastung der nichtlinearen Mannigfaltigkeit bedeuten, können nur als Teil einer Trajektorie erreicht und dann auch nicht gehalten werden, denn die aus statischer Sicht geeigneten Punkte müssen keine stationäre Lösung (Arbeitspunkt) des dynamischen Systems sein. Das „Abtasten“ des Bereichs der nichtlinearen Systemfunktion an Orten, wo aus Sicht der TS-Modellierung ein Partitionsübergang stattfinden sollte, muss also in Form einer vollständigen Trajektorie stattfinden. Hieraus ergibt sich zum Einen die Anforderung, die als relevant angenommenen Bereiche durch eine geeignete Wahl des Gesamtpfades abzudecken, wie in [1] vorgeschlagen, und zum Anderen dabei die Eigenschaften des dynamischen Systems auszunutzen, so dass nicht gegen die Eigendynamik gearbeitet werden muss.

Das zweite Problem fußt darin, dass anzunehmen ist, dass nichtlineares Systemverhalten nicht nur durch ein Scheduling von Außen zustandekommt, sondern auch durch die internen Größen (Zustandsgrößen) beschrieben werden muss. Weiterhin wird zur Vereinfachung der Lösung die Annahme getroffen,

dass die Eingangsgröße linear in das System eingeht. Dies bedeutet, dass die Eingangsgröße keinen Einfluss auf die FBF hat, was es erlaubt, die Entwürfe der Steuerfunktion für die Teilmodelle getrennt durchzuführen und anschließend zu überlagern.

## 3.3 Lösungsansatz

Hieraus ergeben sich folgende Aufgaben:

1. Es ist zu überprüfen, ob der Ansatz aus [1] für dynamische Systeme den gewünschten Effekt hat.
2. Dazu muss ein Pfad im Schedulingraum entworfen werden, welcher näherungsweise den Vorgaben aus [1] und streng der differentiellen Kopplung genügt.
3. Durch einen geeigneten Steuerungsentwurf soll das dynamische System gezwungen werden, den zur Abtastung optimalen Pfad abzufahren.

Der Fokus liegt in diesem Beitrag auf dem Entwurf der Steuerfunktion bei bekanntem optimalen Pfad.

## 3.4 Gesamtkonzept der Identifikation

Die Idee ist, durch eine geeignete Steuerfunktion eine zunächst als bekannt angenommene Trajektorie im Schedulingraum abzufahren, da diese Trajektorie eine optimale Abtastung der nichtlinearen Systemfunktion verspricht. Als optimaler Weg wird der Pfad minimaler Gesamtlänge welcher sich aus den Schnittpunkten der Delauney-Triangulation und Voronoi-Zerlegung ergibt, angenommen (siehe Bild 1). Zur Berechnung der Steuerfunktion wird ein initiales TS-Modell eingesetzt. Deshalb ist zu erwarten, dass diese Steuerfunktion nicht ausreichend ist, um das zugrundeliegende System die optimale Referenztrajektorie ausführen zu lassen. So sollen durch den Einsatz eines Fuzzy-PI-Reglers diese Abweichungen ausgeglichen werden. Die im Vorfeld berechnete Steuerfunktion in Kombination mit dem Regelungseingriff sowie das Ausgangssignal des geregelten Systems wird als folgender Identifikationsdatensatz verwendet. Durch diese iterierten Experimente und Identifikationen soll sichergestellt wer-

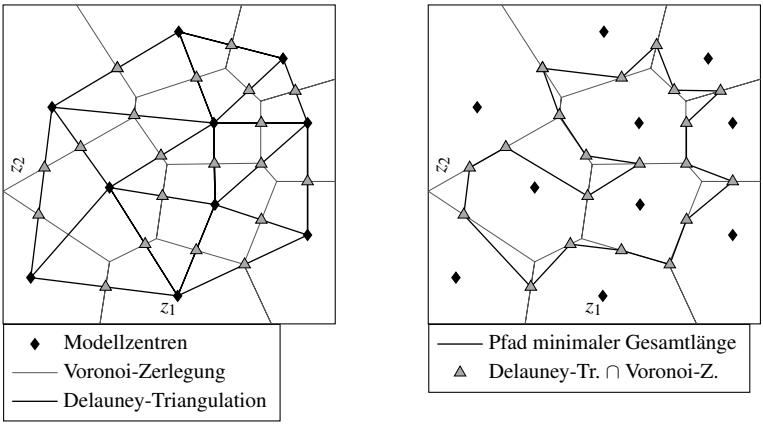



Bild 1: Darstellung der Methode zur Bestimmung des optimalen Pfads

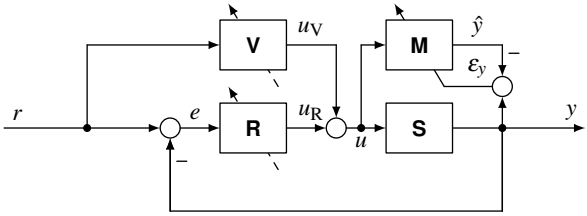



Bild 2: Vorsteuerungsgestützte Closed-Loop-(CL-)Identifikation

den, dass der Eingriff der Regelung praktisch verschwindet. Bild 2 zeigt das Vorgehen. Durch eine mit einem Initialmodell ermittelte Vorsteuerung **V** wird das System **S** angeregt, um dem vorgegebenen Referenzverlauf $r$ zu folgen. Weicht die Ausgangsgröße $y$ von diesem Referenzverlauf ab, soll diese Abweichung durch eine Regelung **R** ausgeglichen werden. Zur Identifikation des Modells **M** wird die Systemeingangsgröße $u = u_V + u_R$ genutzt. Unter der Annahme, dass die Referenzgröße das dynamische Verhalten hinreichend abdeckt, wird darauf geschlossen, dass das identifizierte TS-Modell genau dann gut ist, wenn es mit einer Vorsteuerung basierend auf diesem TS-Modell möglich ist, die Abweichung von Systemausgangs- und Referenzgröße zu eliminieren, ohne dass ein Eingriff der Regelung stattfindet.


Bild 3: Zur Überführung in den Zustandsraum

# 4 Vorsteuerungsbasierter Entwurfsprozess

In diesem Abschnitt wird beschrieben, wie die Steuerfunktion basierend auf einem TS-Modell ermittelt werden kann. Da die FBF, wie erwähnt, nicht von der Eingangsgröße abhängen, werden die lokal gültigen Steuerfunktionen zur globalen Steuerfunktion überlagert. Die globale Steuerfunktion wird dabei anhand des TS-Modells bestimmt. Jede lokale Steuerfunktion hat den selben Aktivierungsgrad wie das entsprechende lokale Teilmodell. Ob die globale Steuerfunktion zum zugrundeliegenden nichtlinearen System passt, hängt von der Qualität des Modells ab.

## 4.1 Übergang zur Zustandsformulierung

Da (1) potentiell verschiedene Zeitschritte der Eingangsgröße $u$ enthält, muss ein weiterer Schritt berücksichtigt werden. Da es sich lokal um eine lineare Differenzengleichung handelt, ist es zulässig die Dynamik des Eingangs („Zähler“) und die des Ausgangs („Nenner“) zu trennen. In der Zustandsdifferenzengleichung sind keine Zeitschritte der Eingangsgröße erlaubt. Daher wird zuerst die Ausgangsdynamik in den Zustandsdifferenzengleichungen berücksichtigt und die Eingangsdynamik in einem zweiten Schritt. Dies ist entgegen einer intuitiven Vorgehensweise, wo zunächst die Dynamik des Eingangs ausgewertet wird, um erst dann die resultierende Eingangsgröße auf die Systemdynamik wirken zu lassen (siehe Bild 3).

Zunächst wird (1) entsprechend aufgeteilt, wobei in diesem Teilabschnitt nicht beachtet wird, dass es sich um Teilmodelle handelt:

$$x(k+n) = -\sum_{l=0}^{n-1} a_l \cdot x(k+l) + \underbrace{u(k)+c}_{\tilde{u}(k)} \tag{5}$$

$$y(k) = \sum_{l=0}^{m} b_l \cdot x(k+l) \tag{6}$$

Es werden Zustandsgrößen definiert als $x_{l+1}(k) = x(k+l)$, woraus sich die Koppelgleichungen ergeben: $x_l(k+1) = x_{l+1}(k)$. In Matrixschreibweise folgt dann die Darstellung in Regelungsnormalform:

$$\begin{bmatrix} x_1(k+1) \\ \vdots \\ x_n(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \\ -a_0 & \cdots & \cdots & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1(k) \\ \vdots \\ x_n(k) \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tilde{u}(k)$$

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{b}\tilde{u}(k) \tag{7}$$

Für die Ausgabegleichung folgt:

$$\begin{aligned} y(k) = &\begin{bmatrix} b_0 & \cdots & b_m & 0 & \cdots & 0 \end{bmatrix} \\ &\cdot \begin{bmatrix} x_1(k) & \cdots & x_m(k) & x_{m+1}(k) & \cdots & x_n(k) \end{bmatrix}^\top \\ y(k) = &\mathbf{c}^\top \mathbf{x}(k) \end{aligned} \tag{8}$$

## 4.2 Entwurf der Steuerfunktion

Für einen einfachen Steuerungsentwurf ist Ausgangssteuerbarkeit nötig. Diese ist selten gegeben, weshalb der Umweg über einen virtuellen Ausgang $\xi(k) = \boldsymbol{\lambda}^\top \mathbf{x}(k)$ gegangen wird, wie in [7] vorgeschlagen. Der virtuelle Ausgabevektor $\boldsymbol{\lambda}^\top$ soll so bestimmt werden, dass für den virtuellen Ausgang die Ausgangssteuerbarkeit vorliegt.

Dazu werden die zukünftigen Schritte des virtuellen Ausgangs $\xi(k)$ betrachtet.

$$\begin{aligned}
\xi(k) &= \boldsymbol{\lambda}^\top \mathbf{x}(k) \\
\xi(k+1) &= \boldsymbol{\lambda}^\top \mathbf{A}\mathbf{x}(k) + \underbrace{\boldsymbol{\lambda}^\top \mathbf{b}}_{=0} \tilde{u}(k) \\
&\vdots \\
\xi(k+n-1) &= \boldsymbol{\lambda}^\top \mathbf{A}^{n-1}\mathbf{x}(k) + \underbrace{\boldsymbol{\lambda}^\top \mathbf{A}^{n-2}\mathbf{b}}_{=0} \tilde{u}(k) \qquad (9) \\
\xi(k+n) &= \boldsymbol{\lambda}^\top \mathbf{A}^{n}\mathbf{x}(k) + \underbrace{\boldsymbol{\lambda}^\top \mathbf{A}^{n-1}\mathbf{b}}_{=1} \tilde{u}(k) \qquad (10)
\end{aligned}$$

In $\xi(k)$ soll die Eingangsgröße $\tilde{u}(k)$ das erste Mal $n$ Schritte in der Zukunft auftreten. So ergibt sich die Bestimmungsgleichung für $\boldsymbol{\lambda}^\top$:

$$\begin{aligned}
\boldsymbol{\lambda}^\top \underbrace{\begin{bmatrix} \mathbf{b} & \cdots & \mathbf{A}^{n-1}\mathbf{b} \end{bmatrix}}_{\mathbf{Q}_\mathrm{S}} &= \begin{bmatrix} 0 & \cdots & 1 \end{bmatrix} \\
\boldsymbol{\lambda}^\top &= \begin{bmatrix} 0 & \cdots & 1 \end{bmatrix} \mathbf{Q}_\mathrm{S}^{-1} \qquad (11)
\end{aligned}$$

Dabei ist $\mathbf{Q}_\mathrm{S}$ die Steuerbarkeitsmatrix. Ein virtueller Ausgang $\xi(k)$ kann genau dann ermittelt werden, wenn das System steuerbar ist. So folgt für ein steuerbares System aus (9):

$$\boldsymbol{\xi}(k) = \underbrace{\begin{bmatrix} \boldsymbol{\lambda}^\top \\ \vdots \\ \boldsymbol{\lambda}^\top \mathbf{A}^{n-1} \end{bmatrix}}_{\mathbf{Q}_\mathrm{B}} \mathbf{x}(k) \qquad (12)$$

Dieser Zusammenhang ist eineindeutig, wenn die Beobachtbarkeitsmatrix $\mathbf{Q}_\mathrm{B}$ bezüglich des virtuellen Ausgangs regulär ist. Dann kann eine Steuerfunktion in zwei Schritten bestimmt werden. Mit der Forderung $r(k) \stackrel{!}{=} y(k) = \mathbf{c}^\top \mathbf{x}(k)$ folgt:

$$r(k) = \mathbf{c}^\top \mathbf{Q}_\mathrm{B}^{-1} \boldsymbol{\xi}(k) \qquad (13)$$

Das nun bekannte $\boldsymbol{\xi}(k)$ wird verwendet, um die gewünschte Eingangsgröße entsprechend (10) zu bestimmen:

$$u(k) = \xi(k+n) - \boldsymbol{\lambda}^\top \mathbf{A}^n \mathbf{Q}_\mathrm{B}^{-1} \boldsymbol{\xi}(k) - c \tag{14}$$

Nun wird berücksichtigt, dass diese Entwürfe lokal sind

$$u_i(k) = \xi_i(k+n) - \boldsymbol{\lambda}_i^\top \mathbf{A}_i^n \mathbf{Q}_{\mathrm{B},i}^{-1} \boldsymbol{\xi}_i(k) - c_i \tag{15}$$

und die Steuerfunktion sich aus ihrer Überlagerung ergibt:

$$u_\mathrm{V}(k) = \sum_{j=1}^{c} u_i(k) \phi_i(\mathbf{z}(k)) \tag{16}$$

Die lokal gültigen Steuerfunktionen $u_i$ werden so ausgelegt, dass jedes lokale Teilmodell dieselbe Trajektorie im Zustandsraum durchläuft. So würde, ohne reale Schaltungseffekte zu berücksichtigen, in ein schaltendes System in jedem Teilmodell die ideale Steuerfunktion aktiv sein, welche für jedes Teilmodell zur selben globalen Trajektorie führt. Durch die unscharfe Überlagerung wird durch die Verschachtelung der FBF ein systematischer Fehler induziert, welcher jedoch für kleine Unschärfeparameter $\nu$ vernachlässigbar ist. In Zukunft muss dieser Fehler abgeschätzt und ggf. kompensiert werden.

Da das System nicht in der Modellklasse liegt existiert eine nichteliminierbare systematische Abweichung zwischen der vom System ausgeführten und vom Modell wiedergegebenen Dynamik. Um dieser Abweichung vorzubeugen wird eine Regelung eingesetzt. Die Regelung wird ebenfalls aus lokalen Fuzzy-Reglern aufgebaut. Für jedes Teilmodell wird ein lokaler PI-Regler verwendet, wobei die lokalen Regeleingriffe ebenfalls durch die Basisfunktion gewichtet werden. Die Regler werden zu diesem Zeitpunkt durch Minimieren eines quadratischen Gütemaßes, welches Zustände und Eingangsgrößen berücksichtigt, ausgelegt. Das Verfahren kann damit entsprechend Bild 4 dargestellt werden.

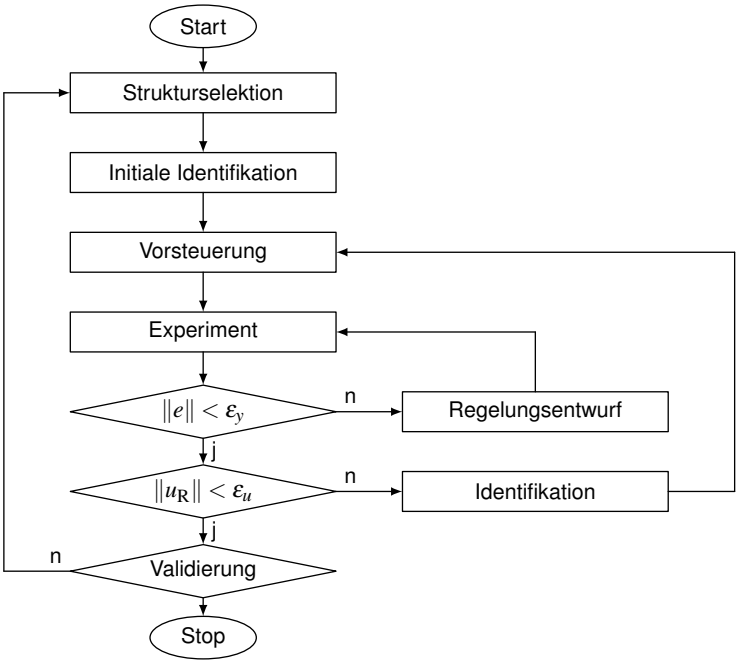



Bild 4: Ablauf des Verfahrens

## 4.3 Ablauf des Verfahrens

Entsprechend Bild 4 wird nach der Festlegung der Modellstruktur (Anzahl der Teilmodelle $c$, dynamische Ordnungen $n$ und $m$, ...) ein prozessmodellfreier Testisignalentwurf inklusive einer nach der vorgestellten Methode durchgeführten Identifikation durchgeführt. Mit dem initialen Modell wird eine Steuerfunktion berechnet. Diese wird auf das geregelte System geschaltet.

Ist die Norm von $\varepsilon_y$ unter einer zu wählenden Schranke, wird überprüft, ob ein zu großer Regelungseingriff vorliegt. Dazu wird überprüft, ob die Norm von $\varepsilon_u$ unter einer zu wählenden Schranke liegt. Ist dies nicht der Fall, wird ein weiteres Modell identifiziert, welches zur Bestimmung der Steuerfunktion eingesetzt wird. Sonst wird mit Hilfe eines externen Datensatzes die Simulationqualität

überprüft. Kann das Modell nicht zufriedenstellend validiert werden, müssen Modellstruktur sowie Referenz überarbeitet werden.

# 5 Simulationsfallstudie

Als eine erste Simulationsfallstudie wird ein System 1. Ordnung verwendet, um die generelle Wirkungsweise zu überprüfen. Das zu untersuchende System ist:

$$\dot{y}(t) = -y^3(t) + u(t) \tag{17}$$

Zunächst wird das System (17) mit der Abtastzeit $\Delta t$ diskretisiert. Es wird festgelegt: $y(t = k\Delta t) = y_k = z_k$, $u(t = k\Delta t) = u_k$, $\dot{y}(t = k\Delta t) = (y_{k+1} - y_k)/\Delta t$. Weiterhin ist $c = 2$, $n = 1$, $m = 0$ und $\nu = 1{,}1$. Damit ist der Modellansatz:

$$\hat{y}_{k+1} = \sum_{i=1}^{c} \phi_i(z_k, \boldsymbol{\Theta}_{\mathrm{MF}})\, \boldsymbol{\varphi}_k^{\top} \boldsymbol{\Theta}_{\mathrm{LM},i} \tag{18}$$

Entsprechend der Festlegungen ist der Regressionsvektor:

$$\boldsymbol{\varphi}_k^{\top} = \begin{bmatrix} 1 & y_k & u_k \end{bmatrix} \tag{19}$$

Die Abtastzeit wird zu $\Delta t = 0{,}01\,\mathrm{s}$ gewählt.

## 5.1 Vereinfachungen

Die Vereinfachungen haben Auswirkungen auf den Entwurf der Referenzgröße, der Steuerung sowie der Regelung. Wie bereits erwähnt, bedeutet Eingangslinearität, dass die Fuzzy-Basisfunktionen (FBF) nicht von der Eingangsgröße abhängen, da die FBF von den Variablen abhängen, die das nichtlineare Verhalten des Systems beschreiben. Dies macht den vorgestellten Steuerungsentwurf möglich. Eine weitere Vereinfachung ist die Verwendung eines Systems erster Ordnung. Jedes lineare, instabile System 1. Ordnung mit $m \leq n$ und keiner Nullstelle mit positivem Realteil kann durch einen PI-Regler mit negativer Reglernullstelle stets durch die Wahl einer geeigneten Proportionalver-

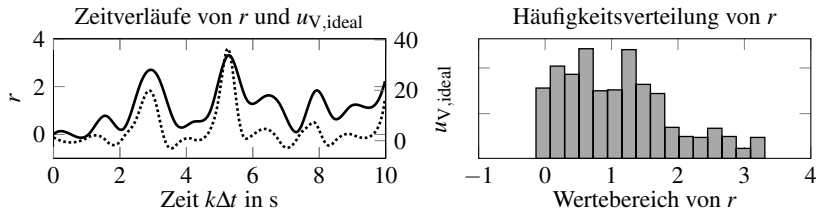



Bild 5: Zeitverlauf der Referenzgröße (links, durchgezogen), ideale Steuerfunktion (links, gepunktet), Häufigkeitsverteilung der Referenzgröße (rechts)

stärkung stabilisiert werden, was die potentielle Schwierigkeit einer komplexen Reglerauslegung für diese Fallstudie eliminiert. Die Systemordnung sowie die Eingangslinearität begrenzen die Dimension der Schedulingvariablen. Es muss sich lediglich in einem Wertebereich bewegt werden, um das nichtlineare Verhalten abzutasten. In dieser Simulationsfallstudie ist keine Stellgrößenbeschränkung vorgesehen.

## 5.2 Entwurf der Referenzgrößen

Da es sich um ein Simulationsbeispiel handelt, können vor der Validierung im Rahmen der Bewertung des Prädiktions- und Simulationsfehlers die Ergebnisse auch mit den wahren „Zwischengrößen“ verglichen werden. Nicht nur ist es ein Zwischenziel, die Ausgangsgröße der Referenz anzugleichen, auch kann die entworfene Steuerfunktion betrachtet werden. Da das System als Gleichung vorliegt, kann auch die Fuzzy-Vorsteuerung während der iterativen Identifikation mit der idealen Vorsteuerung verglichen werden. Ein Betriebsbereich für dieses System 1. Ordnung ist einfach der Wertebereich der Ausgangsgröße. Der Wertebereich der Referenzgröße in Bild 5 ist willkürlich festgelegt worden. Dies hindert jedoch nicht daran, Aussagen über die folgenden Ergebnisse zu treffen. An der Verteilung der Referenzgröße ist zu sehen, dass der Wertebreich ca. zwischen $-0{,}5$ und 3,5 liegt. Die erste Erwartung ist also, dass eine TS-Modellierung, basierend auf der vorgestellten Methode, das nichtlineare Systemverhalten in diesem Bereich zufriedenstellend bechreiben kann. Die untere Hälfte des Wertebreichs hat außerdem eine deutlich höhere Repräsen-

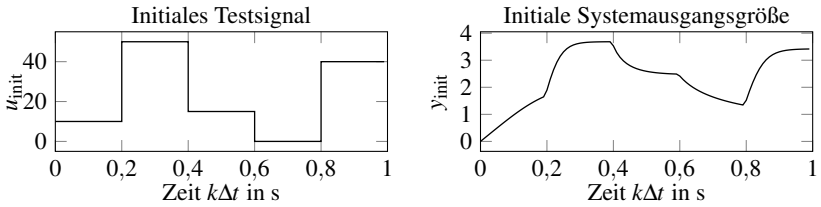



Bild 6: Zeitverläufe des initialen Identifikationsdatensatzes

tation in der Referenzgröße, was zur zweiten Erwartung, dass die Gewichtung der Schätzung sich auf diesen Bereich konzentriert, führt. Diese Erwartungen, welche für das System 1. Ordnung simpel ausfallen, lassen sich jedoch mit wenigen Änderungen auf Systeme höherer Ordnung übertragen.

## 5.3 Durchführung der Experimente

Um eine Fuzzy-Vorsteuerung entwerfen zu können, muss ein initiales Fuzzy-TS-Modell identifiziert werden. Für ein technisches System würden hier prozessmodellfreie Entwurfsverfahren angewendet werden. Für das in dieser Fallstudie verwendete System ein APBRS-Testsignal genutzt, um verschiedene stationäre Ausgangssignale sowie verschiedenes Übergangsverhalten deutlich zu machen. Bild 6 zeigt das initiale Testsignal sowie die zugehörige Systemantwort.

Mit dieser Identifikation wird eine erste Fuzzy-Vorsteuerung und ein neues Experiment entsprechend der vorgestellten Vorgehensweise entworfen. In Bild 7 ist zu erkennen, dass einerseits die reine Fuzzy-Vorsteuerung ohne Regeleingriff (dunkelgrau, durchgezogen, links) deutlich von der idealen Steuerfunktion (schwarz, gepunktet, links) abweicht, wohingegen die Fuzzy-Vorsteuerung inklusive des Regelungseingriffs vergleichsweise ähnlich der idealen Steuerfunktion ist. Dasselbe Schema ist auf der rechten Seite dargestellt. Die Ausgangsgröße, welche zum ungeregelten, nur durch Fuzzy-Vorsteuerung angeregten System (dunkelgrau, durchgezogen, rechts) gehört, weicht deutlich sowohl von der Referenzgröße (schwarz, gepunktet, rechts) als auch von der Systemantwort des Fuzzy-vorgesteuerten und geregelten Systems (hellgrau, strichpunkt,

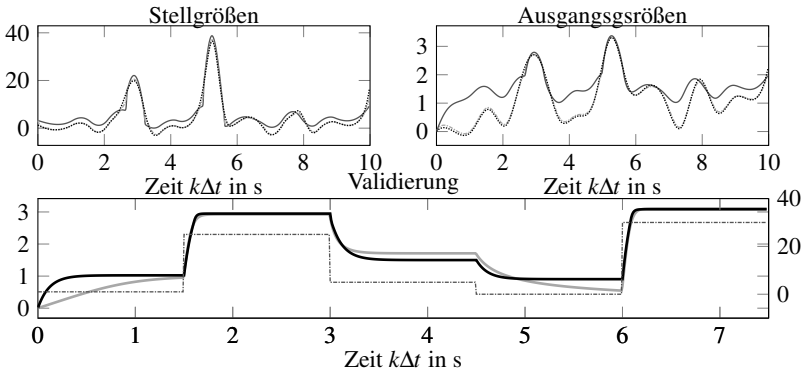



Bild 7: Initiales Experiment mit $c = 2$, Fuzzy-Vorsteuerung ohne Regeleingriff (dunkelgrau, durchgezogen), Fuzzy-Vorsteuerung mit Regeleingriff (hellgrau, strichpunkt), Ideale Steuerfunktion (schwarz, gepunktet); Validierung (unten): Validierungstestsignal (dunkelgrau, strichpunkt, Skala rechts), Systemausgang (hellgrau, durchgezogen), Modellausgang (schwarz, durchgezogen)

rechts) ab. Im unteren Plot sind die Validierungsdaten zu erkennen. Ein Eingangssignal, welches nicht im Rahmen einer Identifikation verwendet wurde, wird sowohl auf das zu identifizierende System geschaltet als auch für die Modellauswertung genutzt. Die Modellauswertung geschieht hierbei in rekursiver Modellauswertung bei gegebenen und identischen Anfangswerten. Die Abweichung zwischen Modellausgang und Systemausgangssignal sind stellenweise deutlich zu erkennen, wobei sich die Abweichungen sowohl durch unzureichende Identifikation des dynamischen als auch des stationären Verhaltens zeigen.

Im folgenden Schritt wird das Paar an Ein- und Ausgangsgrößen, welche zum geregelten und vorgesteuerten System gehören, für eine weitere Identifikation genutzt. Basierend auf diesem Modell wird eine neue Vorsteuerung entworfen. Bild 8 zeigt die Ergebnisse. Auch wenn die Abweichungen bezüglich der Trainingsdaten (Bild 8 oben) deutlich verringert werden konnten, genügen sie den gewählten Schranken von $\|\varepsilon_u\|$ und $\left\|\varepsilon_y\right\|$ nicht. Dasselbe Verhalten zeigen die Validierungsdaten im Bild 8 unten. Die Validierungsdaten des sukzessiven Experiments verhalten sich bezüglich des zurückliegenden Experiments genau so wie die Trainingsdaten. Insgesamt fällt auf, dass die Abweichungen zwischen

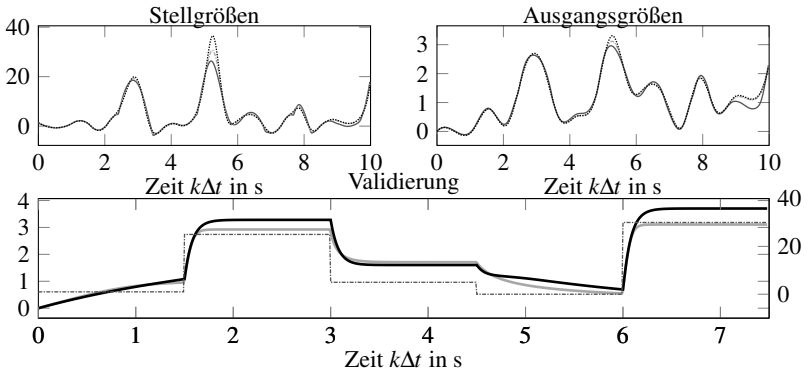



Bild 8: Folgeexperiment mit $c = 2$ basierend auf dem Initialexperiment

Simulation und Systemausgangssignal deutlich kleiner geworden sind, aber immer noch wird sowohl das dynamische als auch das stationäre Systemverhalten nicht vollständig im Wertebereich der Referenzgröße erfasst. Aus diesem Grund wird die Anzahl der Teilmodelle in folgenden Experimenten (Bilder 9 und 10) sukzessive erhöht. Durch die Erhöung der Anzahl der Teilmodelle lässt sich weiterhin eine bessere Beschreibung des Systemverhaltens erreichen. Bei den Trainingsdaten (Bild 9 oben ist der Unterschied zwischen den Zeitreihen kaum mehr erkennbar. Auch beim Validierungsdatensatz zeigt sich dieses Verhalten. Da im Rahmen der Simulationsfallstudie die zulässigen Werte für die Abweichungen bewusst klein gewählt wurden, wurde in einem letzten Folgeexperiment die Anzahl der Teilmodelle auf $c = 6$ erhöht, um die Ergebnisse weiter zu verbessern.

## 5.4 Bewertung der Identifikationen

Die Präsentation der Validierungsdaten innerhalb des Gesamtprozesses dienen der Transparenz. Die eigentliche Validierung geschieht am Ende. Auch wurden die Zwischenvalidierungen so gewählt, dass sie den Wertebereich der Referenzgröße nicht verlassen. Die beschriebene Erwartung war, dass Abschnitte von Ausgangssignalen, welche außerhalb des Wertebereichs der Referenzgröße liegen, nicht durch die Modelle wiedergegeben werden können. Bild 10

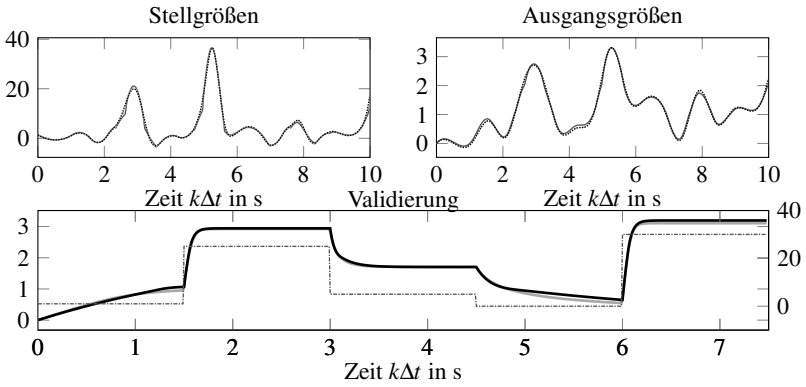



Bild 9: Folgeexperiment mit $c = 3$ basierend auf dem letzten Experiment mit $c = 2$

zeigt deutlich, dass das Systemverhalten im Wertebereich der Referenzgröße gut durch das Fuzzy-System beschrieben werden kann. Bild 11 zeigt die Validierung auf Testdaten, wobei der Systemausgang deutlich außerhalb des Wertebereichs der Referenzgröße liegt. Außerhalb des Bereichs ist das TS-Modell nicht in der Lage, das Systemverhalten wiederzugeben. Da die nichtlineare Funktion exakt bekannt ist, ist es möglich sie direkt mit den Ergebnissen der Modellschätzung zu vergleichen. Bild 12 zeigt die nichtlineare Systemfunktion (schwarz), die durch das Fuzzy-System modellierte Systemfunktion (dunkelgrau) sowie die Lage und Gestalt der FBF. Der Unterschied zwischen dem Bild oben links und dem Bild oben rechts besteht hauptsächlich in der Lage der Teilmodelle. Die nichtlinearen Abweichungen werden in dem Bereich verkleinert, welche durch die Referenzgröße stärker abgedeckt ist. Dies war entsprechend Bild 5 die untere Hälfte des Wertebereichs. Nicht nur wird die Abtastung der nichtlinearen Funktion insgesamt besser, auch ist die Anzahl der Teilmodelle in dem durch die Referenzfunktion stärker abgedeckten Bereich höher und die Abweichung der nichtlinearen Systemfunktion zur Fuzzy-Modellierung wird im Randbereich größer. Dies entspricht den Erwartungen und deckt sich mit der Validierung im Hinblick auf die Bewertung des Simulationsfehlers.

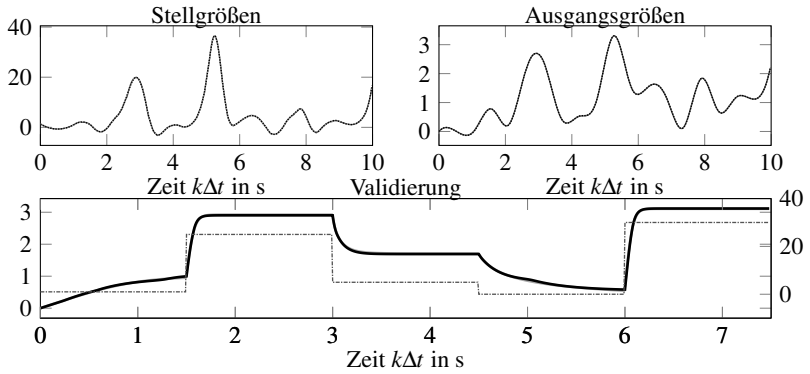



Bild 10: Folgeexperiment mit $c = 6$ basierend auf dem letzten Experiment mit $c = 3$

# 6 Zusammenfassung und Ausblick

In diesem Beitrag wurde eine Methode zum Testsignalentwurf vorgestellt, die zur Reduktion der systematischen Abweichung der geschätzten Parameter eingesetzt werden kann. Die Methode basiert auf zwei wesentlichen Aspekten. Zum einen muss eine Referenzgröße ermittelt werden, die das charakteristische Verhalten des zugrundeliegenden dynamischen Systems im zu identifizierenden Betriebsbereich abdeckt. Zum anderen muss eine Steuerfunktion ermittelt werden, die das zugrundeliegende System die Referenzgröße ausführen lässt. Dieser Beitrag addressierte die iterative Ermittlung der Steuerfunktion bei bekannter Referenzgröße. Die Abweichung der Ausgangsgröße von der Referenzgröße, die insbesondere auf Modellabweichungen in den Partitionsübergängen zurückzuführen sind, wurden durch den Einsatz einer Regelung minimiert, um die Fuzzy-Steuerfunktion der idealen Steuerfunktion anzupassen. Das Potential der Methode wurde anhand einer einfachen Simulationsfallstudie demonstriert.

Es konnte im Rahmen dieser Fallstudie demonstriert werden, dass es mit einer TS-Steuerfunktion möglich ist, ein nichtlineares Systeme eine Referenzgröße ausführen zu lassen und dass die Beschreibung des Systems durch das Modell im Bereich der Referenzgröße durch die Iterationen verbessert werden kann. Bei der Simulationsfallstudie wurden viele signifikante Vereinfachungen

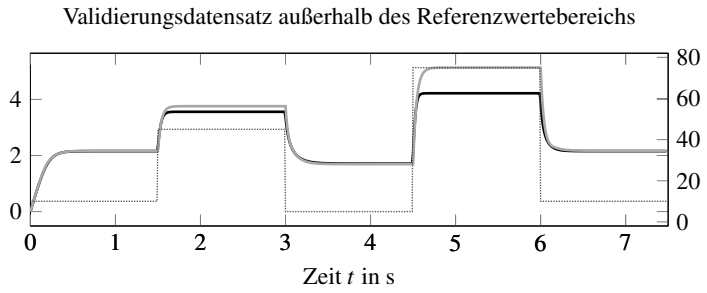



Bild 11: Validierung außerhalb des Referenzbereichs; Ausgabe des nichtlinearen Systems (schwarz, linke Skala), Ausgabe des Fuzzy-Systems (hellgrau, linke Skala), Validierdungstestsignal (dunkelgrau, gepunktet, rechte Skala)

vorgenommen. Dazu zählen die Vernachlässigung von Prozess- oder Messrauschen sowie die durch die Kenntnis über das wahre System gegebene Regressorstruktur und die Schedulingvariable. Neben der Berücksichtigung von Rauscheinflüssen und der Erweiterung auf Systeme höherer Ordnung muss für den Entwurf der Steuerfunktion, in welchem Rahmen die Überlagerung der lokalen Steuerfunktionen zur globalen Steuerfunktion zulässig ist. Für Systeme höherer Ordnung kann auch nicht mehr davon ausgegangen werden, dass eine nahezu beliebige Parametrierung von PI-Reglern ausreicht, um die tatsächliche Systemausgangsgröße entlang der Referenzgröße zu stabilisieren. Die Wahl der Systemordnung 1 hat neben der vereinfachten Durchführbarkeit der vorgestellten Methode zur Folge, dass die Wahl einer geeigneten Referenzfunktion trivial ist. Anstelle eines Wertebereichs, der gleichmäßig abgedeckt sein sollte, tritt nun eine Region im Zustandsraum, in der sich entsprechend des vorgeschlagenen optimalen Pfads entlang der Partitionsgrenzen bewegt werden soll. Die Pfadkoordinaten sind zusätzlich differentiell miteinander gekoppelt, was ein neues Entwurfsverfahren erfordert. Neben weiteren Simulationsfallstudien wird die Methode auch auf technische Testsysteme angewandt werden.

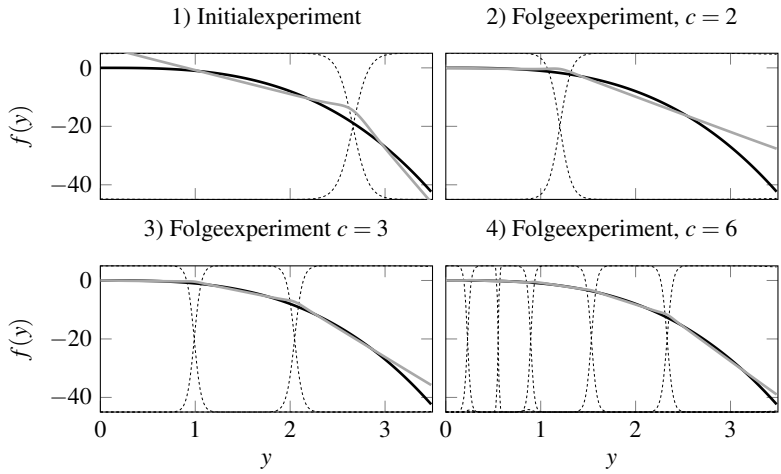



Bild 12: Vergleich der wahren Systemfunktion mit der Fuzzy-TS-Approximation; Fuzzy-Basisfunktionen (schwarz, gestrichelt), Originalfunktion (schwarz), Fuzzy-(TS-)Approximation (hellgrau)

## Danksagung

Die Forschungsarbeit wird durch die Deutsche Forschungsgemeinschaft (DFG) unterstützt, Projektnummer KR 3795/7-1.

## Literatur

[1] A. Kroll, „Zum optimalen Testsignalentwurf für die Partitionierung und Teilmodellparametrierung dynamischer Takagi-Sugeno-Modelle: Problemstellung und Lösungsansätze“, in *Proceedings of the 26th Workshop Computationsal Intelligence*, Dortmund, S. 97–118, 2016.

[2] H. Bandemer, *Theorie und Anwendung der optimalen Versuchsplanung*, Berlin: Akademie-Verlag, 1972.

[3] C. Hametner, M. Stadlbauer, M. Deregnaucourt, M. Jakubek und T. Winsel, „Optimal experiment design based on local model networks and

multilayer perceptron networks“, *Engineering Applications of Artificial Intelligence*, Band 26, Nr. 1, S. 251–261, 2013.

[4] M. Gringard und A. Kroll, „Optimal Experiment Design for Identifying Dynamical Takagi-Sugeno Models with Minimal Parameter Uncertainty“, in *Proceedings of the 18th IFAC Symposium on System Identification (SysID)*, Stockholm, Schweden, S. 353–358, 2018.

[5] A. Kroll und A. Dürrbaum, „On joint optimal experiment design for identifying partition and local model parameters of Takagi-Sugeno models“, in *Proceedings of the 17th IFAC Symposium on System Identification (SysID)*, Beijing, China, S. 1427–1432, 2015.

[6] M. Kahl, A. Kroll, R. Kästner und M. Sofsky, „Zur automatisierten Auswahl signifikanter Regressoren für die Identifikation eines dynamischen Ladedruckmodells“, in *Proceedings of the 24th Workshop Computationsal Intelligence*, Dortmund, S. 33–53, 2014.

[7] M. Zeitz, „Differenzielle Flachheit: Eine nützliche Methodik auch für lineare SISO-Systeme“, in *Automatisierungstechnik*. Band 58, Nr. 1, S. 5–13, 2010.

# Datengetriebene Modellierung zur Prädiktion des Eigenspannungstiefenverlaufs beim Hartdrehen

F. Wittich[1], M. Gringard[1], M. Kahl[1], A. Kroll[1],
W. Zinn[2], T. Niendorf[2]

[1]FG Mess- und Regelungstechnik, Institute for System Analytics and Control,
FB Maschinenbau, Universität Kassel
{felix.wittich, andreas.kroll, matthias.gringard,
matthias.kahl}@mrt.uni-kassel.de

[2]Institut für Werkstofftechnik – Metallische Werkstoffe,
FB Maschinenbau, Universität Kassel
{zinn, niendorf}@uni-kassel.de

## 1 Einleitung

In den letzten Jahren ist das Interesse an Modellen für die Prädiktion von Randschichteigenschaften bei Fertigungsverfahren stark gewachsen [1]. Treibende Faktoren dieser Entwicklung sind steigende Ansprüche an die Werkstückqualität und eine zeit- und kostenoptimierte Produktion. Ein spanendes Fertigungsverfahren für die Bearbeitung von hochbeanspruchten Bauteilen – wie beispielsweise Wellen und Wälzlagerteile – stellt das Hartdrehen dar. Während die Einstellung der gewünschten geometrischen Form des Werkstücks beim Drehen kein Problem mehr darstellt, ist das Einstellen der Randschichteigenschaften ein sehr aktives Forschungsfeld und steht daher auch im Fokus dieses Beitrags. Eine wichtige Randschichteigenschaft ist dabei der oberflächennahe Eigenspannungszustand, der maßgeblich die Bauteileigenschaften eines Bauteiles beeinflusst [2]. So kann beispielsweise die Dauerfestigkeit von Bauteilen durch das Einbringen von oberflächennahen Druckeigenspannungen deutlich erhöht werden [3]. Daher besteht ein großes Interesse daran, die Eigen-

spannungen vorteilhaft für das Bauteilverhalten beeinflussen zu können. Bisher kann die Zielerreichung beim Einstellen des Eigenspannungszustandes erst nach dem Drehprozess in Laboruntersuchungen geprüft werden.

Die datengetriebene Modellbildung bietet hier neue Möglichkeiten, den Zusammenhang zwischen Prozessparametern und Eigenspannungstiefenverlauf zu modellieren und das Ergebnis beispielsweise für den Entwurf eines modellbasierten Regelungssystems für den Randschichtzustand zu nutzen, das der eigentlichen Prozessregelung überlagert ist. In dieser Arbeit wird untersucht, wie gut globale lineare Regressionsmodelle und Takagi-Sugeno-Multi-Modelle für diese Modellbildungsaufgabe geeignet sind. Dazu werden Daten aus experimentellen Untersuchungen zur Eigenspannungsentwicklung beim Hartdrehen eines Stahls 51CrV4 als Basis verwendet.

In Abschnitt 2 werden der zu modellierende Prozess und die Zielgröße – der Eigenspannungstiefenverlauf – näher beschrieben, sowie ein kurzer Überblick über den Stand der Forschung bei der Modellierung von Randschichtzuständen gegeben. In Abschnitt 3 werden die verwendeten Modellierungsansätze beschrieben und die Methoden zur Bewertung der Modelle dargestellt. Darauf folgt in Abschnitt 4 die Darstellung der Ergebnisse der datengetriebenen Modellbildung. Hierauf folgen eine Zusammenfassung dieses Beitrags sowie ein Ausblick.

# 2 Randschichtkonditionierung mittels Hartdrehen

In diesem Abschnitt wird der Prozess des Hartdrehens als ein möglicher finaler Bearbeitungsschritt bei der Bearbeitung gehärteter Bauteile beschrieben und auf dessen Einfluss auf die Randschichteigenschaften, insbesondere des Eigenspannungszustandes, eingegangen. Außerdem wird der Stand der Forschung zur datengetriebenen Modellierung von Zerspanprozessen dargestellt.

## 2.1 Hartdrehprozess

Das Hartdrehen ist ein spanender Fertigungsprozess mit geometrisch bestimmter Schneide, bei dem gehärtete Werkstücke mit einer Härte oberhalb von 47

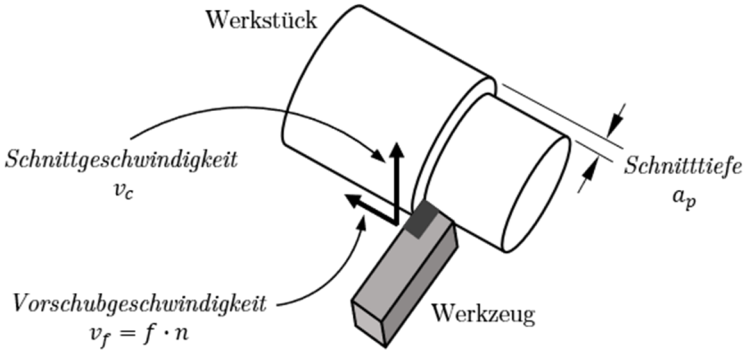



Bild 1: Skizze des Drehprozesses

HRC bearbeitet werden. Ermöglicht wurde das Hartdrehen durch die Entwicklung von Schneidstoffen mit hoher Härte, wie polykristallinem kubischem Bornitrid. Zu den Maschinenstellgrößen beim Hartdrehen zählen der Vorschub $f$ bzw. die Vorschubgeschwindigkeit $v_\mathrm{f}$, die Schnittgeschwindigkeit $v_\mathrm{c}$ und die Schnitttiefe $a_\mathrm{p}$, siehe Bild 1.

Durch den Härtungsprozess vor dem Hartdrehen stellt sich eine homogene initiale Oberflächenhärte $HV_0$ des Werkstücks ein. Diese wird mit dem Prüfverfahren nach Vickers ermittelt. Auf den Prozess wirken darüber hinaus verschiedene Störgrößen. Dazu zählen Maschinenschwingungen MS, Werkzeugfehlstellungen WF und der Werkzeugverschleiß WV. Neben dem Ziel, die gewünschte geometrische Form einzustellen, werden durch den Drehprozess die Randschichteigenschaften des Werkstücks beeinflusst – diese werden unter dem Begriff „surface integrity“ zusammengefasst. Hierzu zählen unter anderem die Rauheit, angegeben als gemittelte Rautiefe $R_\mathrm{z}$, die Post-Process-Vickershärte $HV$ und der im Fokus dieser Arbeit stehende Eigenspanungszustand $\sigma_\mathrm{res}$. Die Eigenspannungen sind vom Abstand unter der Oberfläche des Werkstücks abhängig und bilden sich bis in eine Tiefe von etwa 150 $\mu$m aus. In Bild 2 und Tabelle 1 sind die am Prozess beteiligten Größen dargestellt.

Der Randschichtzustand ist direkt für die Qualität und Lebensdauer des Werkstücks verantwortlich. Das Ziel des „surface engineering“ ist es, gezielt einen gewünschten Randschichtzustand einzustellen. Das Hartdrehen wird als finaler

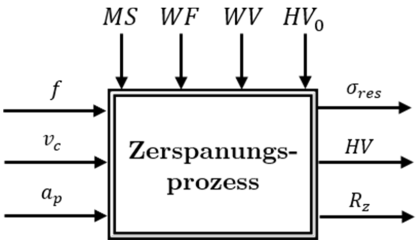



Bild 2: Stell- und Störgrößen sowie Ausgangsgrößen, die den Randschichtzustand beim Hartdrehen beeinflussen

Tabelle 1: Stell- und Störgrößen sowie Ausgangsgrößen, die den Randschichtzustand beim Hartdrehen beeinflussen

| | Symbol | Variable | Einheit |
|---|---|---|---|
| Stellgrößen | $f$ | Vorschub | mm |
| | $v_c$ | Schnittgeschwindigkeit | m/min |
| | $a_p$ | Schnitttiefe | mm |
| Störgrößen | $WV$ | Werkzeugverschleiß | — |
| | $MS$ | Maschinenschwingungen | — |
| | $WF$ | Werkzeugfehlstellungen | — |
| | $HV_0$ | initiale Vickershärte | — |
| Ausgangsgrößen | $\sigma_{res}$ | Eigenspanungen | MPa |
| | $HV$ | Post-Process-Vickershärte | — |
| | $R_z$ | gemittelte Rautiefe | $\mu$m |

Bearbeitungsschritt bei gehärteten Werkstücken eingesetzt und kann beispielsweise einen zusätzlichen Schleifvorgang ersetzen.

## 2.2 Messung der Eigenspannungen

Eigenspannungen sind Spannungen, die in einem Körper ohne das Einwirken von äußeren Kräften und Momenten oder von Temperaturgradienten im Bauteil herrschen. Oberflächennahe Eigenspannungen beeinflussen die Schwingfestigkeit, die Spannungsrisskorrosion und die Verschleißeigenschaften eines

Bauteils – prinzipiell wirken sich hier Druckeigenspannungen positiv und Zugeigenspannungen negativ aus [3]. Beim Hartdrehen wirken hohe mechanische und thermische Belastungen lokal im Eingriffsbereich des Werkzeugs, wodurch Eigenspannungen in der Randschicht induziert bzw. beeinflusst werden. Somit stellt sich ein tiefenabhängiger Eigenspannungsverlauf unter der Oberfläche des Werkstücks ein.

Die Messung des Eigenspannungszustands kann mit der Methode der Röntgendiffraktometrie erfolgen. Hierbei wird ein richtungsabhängiger Beugungswinkel gemessen, über den auf die Gitterdehnung und somit auf die Eigenspannung geschlossen werden kann. Zumeist wird hierzu die sogenannte $sin^2\psi$-Methode angewandt. Um einen Tiefenverlauf der Eigenspannungen messen zu können, wird an einer repräsentativen Stelle die Oberfläche des Werkstücks schichtweise elektrolytisch abgetragen. Bei der Eigenspannungsmessung mittels Röntgendiffraktometrie treten wie bei allen experimentellen Methoden Unsicherheiten bei der Messung aufgrund verschiedener Fehlerquellen, wie der Messgerätausrichtung oder dem Oberflächenzustand der Probe, auf [4].

In Bild 3 sind beispielhaft gemessene Eigenspannungstiefenverläufe für zwei verschiedene Parameterkombinationen beim Hartdrehen dargestellt. Die Messung des Tiefenverlaufs wurde jeweils an einer exemplarischen Stelle des Werkstücks vorgenommen. Dabei ist der charakteristische Verlauf, mit Druckeigenspannnungsmaximum unter der Oberfläche und einem Annähern der Eigenspannung an den Wert 0 in der Tiefe zu erkennen.

## 2.3 Stand der Technik bei der Modellbildung

Um den steigenden Anforderungen an Prädiktionsmodelle für Randschichtzustände bei spanenden Fertigungsverfahren zu begegnen, sind verschiedene Modellierungsansätze untersucht worden. Die Modelle können in drei Kategorien zusammengefasst werden; analytische, numerische und datengetriebene Modelle [6]. Da die analytische Modellierung von Eigenspannungen bei Zerspanprozessen auf Grundlage von physikalischen Zusammenhängen sehr aufwendig oder gar nicht möglich ist, werden überwiegend numerische Finite-Elemente-(FEM-)Modelle eingesetzt [1].

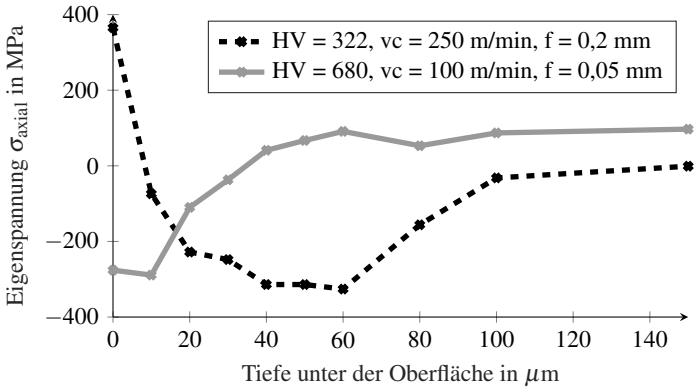



Bild 3: Gemessener Tiefenverlauf der Eingenspannung $\sigma_{axial}$ für zwei verschiedene Parameterkombinationen beim Hartdrehen

Der folgende Überblick konzentriert sich auf die datengetriebenen Modellierungsansätze, da diese eine Reihe von Vorteilen bieten. Dazu zählen die Folgenden:

1. Das System kann ohne Expertenwissen, bzw. ohne Einbringen von Wissen über die physikalischen Grundlagen modelliert werden. Dies ist ein Vorteil, wenn dieses Wissen nur schwer eingebracht werden kann oder nur unvollständig bzw. gar nicht vorhanden ist.
2. Die Komplexität und der Rechenaufwand können insbesondere gegenüber numerischen Modellen deutlich reduziert werden. Dies bietet Vorteile, unter anderem bei der Steuerung und Reglung und im Echtzeiteinsatz.
3. Die Modellstruktur kann anwendungsspezifisch gewählt werden, bspw. passend für den gewünschten Regleransatz.

Neben Modellierungsansätzen für den Tiefenverlauf unter Verwendung von Polynomen [5] und der Anpassung nicht-linearer Funktionen, wie einer exponentiell gedämpften Sinuskurve [7], finden auch Methoden der Computational Intelligence Einsatz – ein Überblick zur datengetriebenen Modellierung bei Drehprozessen ist in [8] zu finden. Ein möglicher Modellansatz sind dabei Künstliche Neuronale Netze (KNN), wie sie in [9] für die Eigenspannungsprä-

diktion beim Fräsen eingesetzt werden. Um allerdings die erforderliche Datenmenge für das Trainieren des KNNs zu erhalten, werden hierzu Trainingsdaten mit einem FEM-Modell erzeugt. Alternativ wird in [10] ein datengetriebenes Fuzzy-Modell auf Basis eines modifizierten Complete-Link-Clustering-Algorithmus für die Eigenspannungsprädiktion bei zerspanenden Verfahren eingesetzt.

# 3 Modellierungsansätze und Identifikationsmethoden

Bei der in diesem Beitrag betrachteten Modellierungsaufgabe soll ein Modell für ein statisches MISO-System identifiziert werden. Zuerst sollen hierzu Standardmethoden der linearen Regression eingesetzt werden, um ein Globalmodell des Prozesses zu erhalten. Im Vergleich dazu sollen Takagi-Sugeno-(TS-)Modelle als nicht-linearer Regressionsansatz in Form von Multi-Modellen zur lokalen Modellierung eingesetzt werden. Beide Ansätze arbeiten rein datengetrieben – ohne Prozess- oder Vorwissen einzubringen. Um die Fähigkeit der Modelle, auch unbekannte Datenpunkte prädizieren zu können, zu bewerten, werden entsprechende kreuzvalidierte Gütemaße verwendet.

## 3.1 Lineare Regressionmodelle

In einem ersten Schritt soll die Modellierung mit einem globalen multivariaten Regressionsmodell

$$\hat{y}(\boldsymbol{w}) = a_o + a_1 w_1 + \cdots + a_n w_n \tag{1}$$

mit den Parametern $a_j$ und den Regressoren $w_j$ aus der Kandidatenmenge $K$ erfolgen. $K$ enthält lineare Terme $x_i$, Monome $\left\{x_i^2, \ldots, x_i^k\right\}$ und Interaktionsterme $\left\{x_i x_j, x_i x_l, x_i x_j x_l, \ldots\right\}$, die aus den $p$ Eingangsgrößen $x_i$ mit $i \in \{1, \ldots, p\}$ konstruiert wurden. Das Regressionsmodell wurde gewählt, da es einen etablierten, einfach zu implementierenden Ansatz darstellt. Zur Parameterschätzung stellt sich das Optimierungsproblem mit einer quadratischen Kostenfunktion bei $N$

Datenpunkten folgendermaßen dar:

$$\hat{\boldsymbol{\Theta}} = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \,. \tag{2}$$

Da das Modell linear in den Parametern ist, führt das Least-Squares-Problem zu einem konvexen Optimierungsproblem, so dass die Parameter global optimal geschätzt werden können.

Um eine für die Modellierung geeignete Untermenge der Kandidatenterme $K_{\text{mod}} \subseteq K$ auszuwählen und so ein kompaktes Modell zu erzeugen, wurden Methoden der Variablenselektion angewendet, wie sie beispielsweise bei der Modellierung von dynamischen Systemen in [11] zum Einsatz kommen. In diesem Beitrag soll die schrittweise Regression (SWR) [12] als Wrapper-Methode und der Least Absolute Shrinkage and Selection Operator (LASSO) [13] als Embedded-Methode verwendet werden. Bei den Wrapper-Verfahren wird jedes geschätzte Kandidatenmodell mit einer Untermenge aus $K$ bewertet und verglichen, bei den Embedded-Verfahren hingegen findet die Variablenselektion während des Modelltrainings statt. Beide Verfahren, SWR und LASSO, werden in [11] genauer beschrieben. Die SWR ist ein einfache und universell einsetzbare Methode, wohingegen LASSO zu effizienteren Ergebnissen führen kann [14].

Bei der SWR werden sukzessive Regressoren auf Basis einer Bewertung mit einem Bewertungsmaß hinzugefügt bzw. entfernt. Der Algorithmus beginnt mit einem konstanten Modell $y = a_0$ und fügt schrittweise Regressoren aus $K$ hinzu, die zu einer Verbesserung des Bewertungsmaßes führen. Als Bewertungsmaß wird das adjustierte Bestimmtheitsmaß $R^2_{\text{adj}}$ verwendet, da dieses, wie in Abschnitt 3.3 beschrieben, die Anzahl der Regressionsparameter berücksichtigt. Als Schwellenwert für die Auswahl wurde $\tau = 0,05$ gewählt, d.h. das $R^2_{\text{adj}}$ muss mindestens um diesen Wert steigen, damit ein Kandidatenterm in das Modell miteinbezogen wurde.

LASSO führt gleichzeitig eine Regularisierung und Regressorenselektion durch, indem es die Fehlerquadratsumme, die beim Least-Squares-Verfahren minimiert wird, um einen Strafterm $\lambda \sum_{j=1}^{n} |\beta_j|$ erweitert. Dafür muss der Parameter $\lambda$ bestimmt werden. Dies geschah dadurch, dass das Modell mit dem $\lambda$

gewählt wurde, das die 10-fach kreuzvalidierte mittlere quadratische Abweichung minimiert.

## 3.2 Takagi-Sugeno-Modelle

Bei der Modellierung des Eigenspannungstiefenverlaufs mit einem TS-Modell werden lokal-affine Teilmodelle identifiziert und diese über im Schedulingraum definierte Zugehörigkeitsfunktionen gewichtet überlagert. Das TS-Modell besitzt die Eigenschaft, nichtlineare Funktionen approximieren zu können und dabei innerhalb der lokalen Modelle linear-affine Eigenschaften aufzuweisen [15].

Für die Partitionierung des Eingangsgrößenraums wird der Fuzzy-c-Means-(FCM-)Algorithmus verwendet. Die Clusterung findet im Produktraum, also im von Eingangs- und Ausgangsgrößen aufgespannten Raum, statt. In Voruntersuchungen konnte festgestellt werden, dass eine Einschränkung des Schedulingraumes nicht zu besseren Ergebnissen geführt hat. Daher werden im Folgenden vereinfachend die Schedulingvariablen und die Regressoren identisch gewählt. Die durch den FCM ermittelten Klassifikatorfunktionen können direkt als Zugehörigkeitsfunktionen verwendet werden. Das statische MISO-TS-Gesamtmodell

$$\hat{y}(\boldsymbol{x}) = \sum_{j=1}^{c} \phi_j(\boldsymbol{x}) \hat{y}_j(\boldsymbol{x}) \tag{3}$$

mit den Eingangsgrößen $\boldsymbol{x}$ setzt sich aus $c$ überlagerten lokal-affinen Teilmodellen

$$\hat{y}_j(\boldsymbol{x}) = a_{0,j} + a_{1,j} x_1 + \cdots + a_{n,j} x_n \tag{4}$$

zusammen. Dabei sind

$$\phi_j(\boldsymbol{x}) = \frac{\mu_j(\boldsymbol{x})}{\sum_{j=1}^{c} \mu_j(\boldsymbol{x})} \tag{5}$$

die Fuzzy-Basisfunktionen mit den Zugehörigkeitsfunktionen

$$\mu_j(\boldsymbol{x}) = \left[ \sum_{l=1}^{c} \left( \frac{\|\boldsymbol{x} - \boldsymbol{v}_j\|_2}{\|\boldsymbol{x} - \boldsymbol{v}_l\|_2} \right)^{\frac{2}{\nu - 1}} \right]^{-1} . \tag{6}$$

Da für die Zugehörigkeitsfunktionen des FCM die Orthogonalitätsbedingung

$$\sum_{j=1}^{c} \mu_j(\boldsymbol{x}) = 1 \tag{7}$$

bereits erfüllt ist, gilt $\mu_j = \phi_j$. In (6) sind $\boldsymbol{v}_j$ die Clusterprotoypen und $\nu \in \mathbb{R}^{>1}$ der Fuzzyparameter, der die Unschärfe zwischen den Clustern festlegt. Danach werden die Parameter der lokalen Modelle $\boldsymbol{\Theta}_{\mathrm{LM}}$ und die Positionen der Prototypen $\boldsymbol{v}$ in einen Gesamtparametervektor $\boldsymbol{\Theta}_{\mathrm{ges}} = \left[\boldsymbol{\Theta}_{\mathrm{LM}}^T, \boldsymbol{v}^T\right]$ zusammengefasst und parallel optimiert. Für das nichtlineare Optimierungsproblem mit $N$ Datenpunkten wird eine quadratische Kostenfunktion verwendet:

$$\hat{\boldsymbol{\Theta}} = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}} \sum_{i=1}^{N} (y_i - \hat{y}_i(\boldsymbol{\Theta}_{\mathrm{ges}}))^2 \tag{8}$$

und mit der MATLAB-Funktion `lsqnonlin`, die einen Trust-Reflective-Least-Squares-Algorithmus verwendet, optimiert.

## 3.3 Modellvalidierung und Modellgütebewertung

Für die Bewertung der Modelle liegt der Fokus insbesondere darauf, wie gut die Prädiktion unbekannter Datenpunkte ist. Um zu bewerten, wie gut das Modell generalisiert, muss es auf Testdaten validiert werden. Für die relativ kleine Datenbasis eignet sich hierzu eine $k$-fache Kreuzvalidierung. Bei der $k$-fachen Kreuzvalidierung wird die verfügbare Datenmenge in $k$ gleichgroße Teilmengen $\{T_1, \ldots, T_k\}$ aufgeteilt und in $k$ Durchgängen die Parameterschätzung mit $k-1$ Teilmengen als Trainingsdaten und der verbleibenden unbekannten Teilmenge als Testdatensatz durchgeführt. Das Gesamtgütemaß ist dann der Durchschnitt aus den Teilgütemaßen. Wie in [16] vorgeschlagen, wird $k = 10$ als geeigneter Wert festgelegt. Als Bewertungsmaß werden dabei der Root Mean Square Error (RMSE):

$$\mathrm{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2} \tag{9}$$

mit $N$ der Anzahl der Datenpunkte sowie das Bestimmtheitsmaß $R^2$ verwendet:

$$R^2 = \frac{\sum_{i=1}^{N}(\hat{y}_i - \bar{y})}{\sum_{i=1}^{N}(y_i - \bar{y})}. \tag{10}$$

Das $R^2$ – als klassisches Gütemaß bei Regressionsmodellen – gibt an, welcher Teil der Streuung der Beobachtungen durch das Modell erklärt wird. Im Folgenden werden die kreuzvalidierten Varianten von (9) und (10) als $\mathrm{RMSE}_{\mathrm{cv}}$, bzw. $R^2_{\mathrm{cv}}$ bezeichnet.

Um die Modellgüte unter Berücksichtigung der Anzahl der zu schätzenden Modellparameter zu bewerten, werden das adjustierte Bestimmtheitsmaß $R^2_{\mathrm{adj}}$ sowie das Bayesian Information Criterium BIC verwendet. Diese Kriterien werden auf Basis des gesamten Datensatzes berechnet. Das adjustierte $R^2$

$$R^2_{\mathrm{adj}} = 1 - (1 - R^2)\frac{N-1}{N - N_\Theta} \tag{11}$$

mit der Anzahl der geschätzten Parameter $N_\Theta$ bestraft über den Faktor $(N-1)$ $/(N-N_\Theta)$ das Hinzufügen zusätzlicher Modellparameter. Als informationsbasiertes Gütekriterium bestraft das Bayesian Information Criterion (BIC)

$$\mathrm{BIC} = N\ln(I(\Theta)) + \ln(N)N_\Theta \tag{12}$$

mit

$$I(\Theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{13}$$

eine höhere Parameteranzahl $N_\Theta$ und belohnt eine bessere Prädiktionsgüte.

# 4 Fallstudie

Im Folgenden werden die Ergebnisse der Modellierung mit den in Abschnitt 3 beschriebenen Ansätzen dargestellt. Zuerst wird hierzu die verwendete Datenbasis beschrieben.

## 4.1 Datenbasis

Der zur Identifikation verwendete Datensatz stammt aus einer experimentellen Versuchsreihe [17]. Untersucht wurden zylindrische Proben des Stahls 51CrV4 in verschiedenen Härtestufen mit einem Durchmesser von 34 mm und einer Länge von 200 mm. Sie wurden zuvor auf einer CNC-Drehmaschine des Typs Monforts RNC602 am Institut für spanende Fertigung der TU Dortmund bearbeitet. Da die Erzeugung der verschiedenen Zustände und die Labormessungen sehr zeitaufwendig und kostenintensiv sind, ist die Datenlage spärlich, das heißt die Anzahl der Stützstellen im Produktraum ist klein. Am besten abgedeckt sind die Eingangsgrößen $HV_0$, bzw. ist die Tiefe $d_s$ der Messung der Eigenspannung unter der Oberfläche mit 6 bzw. 10 Stufen. Dagegen ist die Variation in den Größen $f$ und $v_c$ mit nur 3 unterschiedlichen Stufen gering. Insgesamt enthält der Datensatz 328 Datenpunkte. Der vierdimensionale Eingangsgrößenraum setzt sich aus den Größen Abstand zur Oberfläche $d_s$, Vorschub $f$, Schnittgeschwindigkeit $v_c$ und der initialen Härte $HV_0$ zusammen. Als Ausgangsgröße wird in dieser Arbeit exemplarisch die axiale Normaleigenspannung als zu prädizierende Größe verwendet, siehe Bild 4. Wie in Abschnitt 2.2 beschrieben, wird die Messung der Eigenspannung nur an einer Stelle des Werkstücks vorgenommen. Es ist aber zu erwarten, dass die Eigenschaften, z.B. durch unterschiedliche Gefügeausprägungen durch das Härten, nicht homogen über das gesamte Werkstück verteilt sind. Auch die initiale Härte weist eine heterogene Verteilung auf der Werkstückoberfläche auf und wird nur über einen gemittelten Wert wiedergegeben. Entsprechend weisen die Messwerte Unsicherheiten auf. Da die Ein- und Ausgangsdaten unterschiedliche Wertebereiche aufweisen und der FCM als verwendetes Clusterungsverfahren nicht skaleninvariant ist, wurden die Messwerte standardisiert.

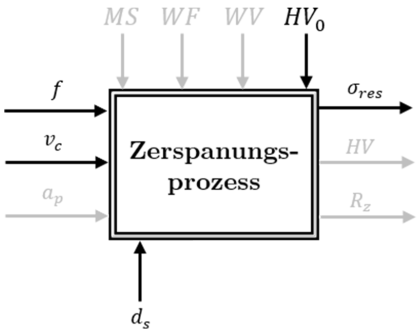


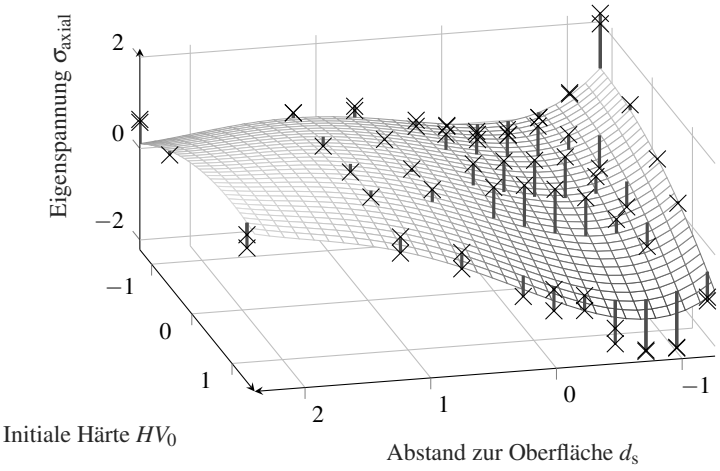

Bild 4: Bei der Modellierung im Fallbeispiel verwendete Größen

## 4.2 Ergebnisse mit linearer Regression

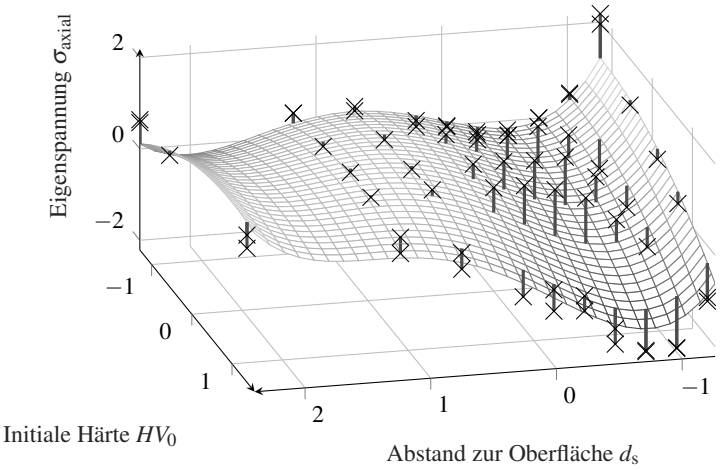
In die Kandidatenmenge $K$ der Regressoren wurden Monome bis zum vierten Grad aufgenommen, um die Nichtlinearitäten ausreichend genau modellieren zu können. Damit ergibt sich die Anzahl der Elemente der Kandidatenmenge $K$ zusammen mit den Kombinationen der Interaktionsterme zu $|K| = 31$. In Tabelle 3 sind die Ergebnisse der Identifikation dargestellt. Das mit LASSO identifizierte Modell weist im Vergleich zum mit der SWR identifizierten Modell eine bessere Modellgüte auf. Insgesamt weisen aber beide Modelle keine gute Prädiktionsgüte auf. In Bild 5 sind die Verläufe beider Modelle dargestellt. Mit LASSO wurden 20 Terme, mit der SWR nur 12 Terme selektiert, siehe Tabelle 2. Dabei hat LASSO alle Terme ausgewählt, die auch mit der SWR selektiert wurden, also $K_{\mathrm{SWR}} \subseteq K_{\mathrm{LASSO}}$. Außerdem wurde die Eingangsgröße $v_{\mathrm{c}}$ von der SWR gar nicht, und von LASSO nur in den Interaktionstermen selektiert.

## 4.3 Ergebnisse mit TS-Modellen

Da das TS-Modell die Hyperparameter $c$ und $\nu$ besitzt, wurden verschiedene Kombinationen dieser Parameter getestet. Nach [18] können für Modellierungsaufgaben geeignete Werte für $\nu$ im Intervall $(1;1,5]$ gefunden werden. Mit $\nu_i \in \{1,2;1,3;1,4\}$ wurden entsprechende Werte ausgewählt. Für die zu testende Anzahl der Clusterzentren wurde $c \in \{2;3;4;5\}$ gewählt.




(a) Regressionsmodell unter Verwendung von LASSO




(b) Regressionsmodell unter Verwendung von SWR

Bild 5: Kennflächen der Regressionmodelle (Gitter) und gemessene Eigenspannungen (×) mit Residuen in Abhängigkeit von initialer Härte $HV_0$ und Abstand von der Oberfläche $d_s$ bei konstanten Schnittparametern $f = 0,25$ mm und $v_c = 175$ m/min

Tabelle 2: Durch SWR und LASSO ausgewählte Modellterme

| | Monome | | | | Interaktion | | |
|---|---|---|---|---|---|---|---|
| Verfahren | 1. Ord. | 2. Ord. | 3. Ord. | 4. Ord. | 2. Ord. | 3. Ord. | 4. Ord. |
| SWR | $d_\mathrm{s}$<br>$HV_0$<br>$f$ | $HV_0^2$ | $d_\mathrm{s}^3$ | $HV_0^4$ | $d_\mathrm{s}f$<br>$d_\mathrm{s}HV_0$<br>$HV_0f$ | $d_\mathrm{s}HV_0f$ | |
| LASSO | $d_\mathrm{s}$<br>$HV_0$<br>$f$ | $HV_0^2$<br>$d_\mathrm{s}^2$<br>$f^2$ | $d_\mathrm{s}^3$<br>$f^3$ | $HV_0^4$<br>$d_\mathrm{s}^4$ | $d_\mathrm{s}f$<br>$d_\mathrm{s}HV_0$<br>$HV_0f$<br>$d_\mathrm{s}v_\mathrm{c}$<br>$HV_0v_\mathrm{c}$<br>$fv_c$ | $d_\mathrm{s}HV_0f$<br>$d_\mathrm{s}fv_\mathrm{c}$ | $d_\mathrm{s}HV_0fv_\mathrm{c}$ |

In Tabelle 4 lässt sich erkennen, dass das Prädiktionsmodell mit den Parametern $c = 3$ und $\nu = 1,2$ die besten out-of-sample Ergebnisse liefert, erkennbar am kreuzvalidierten $R^2_\mathrm{CV}$ von $0,8669$. Die besten Ergebnisse bei der in-sample Modellgüte liefert die Parameterkombination $c = 4$ und $\nu = 1,3$ mit einem BIC von $-751,95$. Dies ist ein Anzeichen für eine Überanpassung des Modells an das Rauschen; mit $c = 5$, also einer weiteren Vergrößerung der Anzahl der Modellparameter, nimmt die Überanpassung weiter zu und der kreuzvalidierte Modellfehler steigt an. Bild 6 zeigt eine gute Anpassungsgüte des Modells mit $c = 3$ und $\nu = 1,2$ an die gemessenen Werte. In Bild 7 sind die Lage der Prototypen und die Höhenlinien der Zugehörigkeitsfunktionen für $c = 3$ und $\nu = 1,2$ dargestellt. Die Prototypen $\boldsymbol{v}_i$ liegen erwartungsgemäß in den Bereichen, in denen der Eigenspannungsverlauf starkes nichtlineares Verhalten zeigt. Dabei fällt auf, dass ein Prototyp außerhalb des Wertebereichs der Trainingsdaten liegt, wobei dies prinzipiell als Folge der Nachoptimierung nicht ungewöhnlich ist.

Tabelle 3: Modellvergleich der Regressionsmodelle mit SWR und LASSO als Selektionsverfahren

| Verfahren | $\mathrm{RMSE}_{\mathrm{CV}}$ | $R^2_{\mathrm{CV}}$ | $R^2_{\mathrm{adj}}$ | BIC | $N_\Theta$ |
|---|---|---|---|---|---|
| SWR | 0,5903 | 0,6552 | 0,6870 | -275,95 | 11 |
| LASSO | **0,5669** | **0,6806** | **0,7012** | **-344,68** | 20 |

Tabelle 4: Modellvergleich der TS-Modelle für verschiedene Hyperparameter $c$ und $\nu$

| c | $\nu$ | $\mathrm{RMSE}_{\mathrm{CV}}$ | $R^2_{\mathrm{CV}}$ | $R^2_{\mathrm{adj}}$ | BIC | $N_\Theta$ |
|---|---|---|---|---|---|---|
| 2 | 1,2 | 0,4752 | 0,7676 | 0,8298 | -524,51 | 12 |
| 2 | 1,3 | 0,5477 | 0,6922 | 0,6017 | -245,72 | 12 |
| 2 | 1,4 | 0,5304 | 0,7141 | 0,7992 | -470,41 | 12 |
| **3** | **1,2** | **0,3552** | **0,8669** | 0,9085 | -699,65 | 18 |
| 3 | 1,3 | 0,3990 | 0,8380 | 0,8968 | -660,16 | 18 |
| 3 | 1,4 | 0,3955 | 0,8337 | 0,8917 | -644,25 | 18 |
| 4 | 1,2 | 0,4378 | 0,7744 | 0,9177 | -706,20 | 24 |
| **4** | **1,3** | 0,3681 | 0,8459 | **0,9284** | **-751,95** | 24 |
| 4 | 1,4 | 0,4415 | 0,7865 | 0,9121 | -684,50 | 24 |
| 5 | 1,2 | 0,5503 | 0,6784 | 0,6395 | -410,79 | 30 |
| 5 | 1,3 | 0,6743 | 0,7225 | 0,5989 | -387,91 | 30 |
| 5 | 1,4 | 0,4994 | 0,7816 | 0,6855 | -492,34 | 30 |

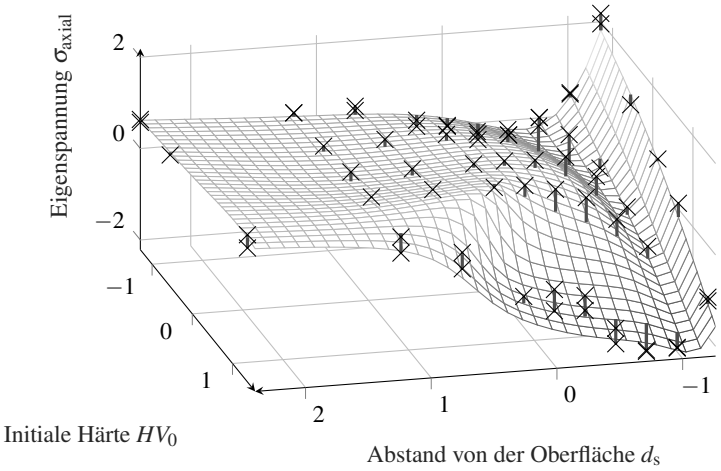



Bild 6: Kennfläche des TS-Modells (Gitter) mit $c = 3, \nu = 1,2$ und gemessene Eigenspannungen ($\times$) mit Residuen in Abhängigkeit von initialer Härte $HV_0$ und Abstand von der Oberfläche $d_s$ bei konstanten Schnittparametern $f = 0,25$ mm und $v_c = 175$ m/min

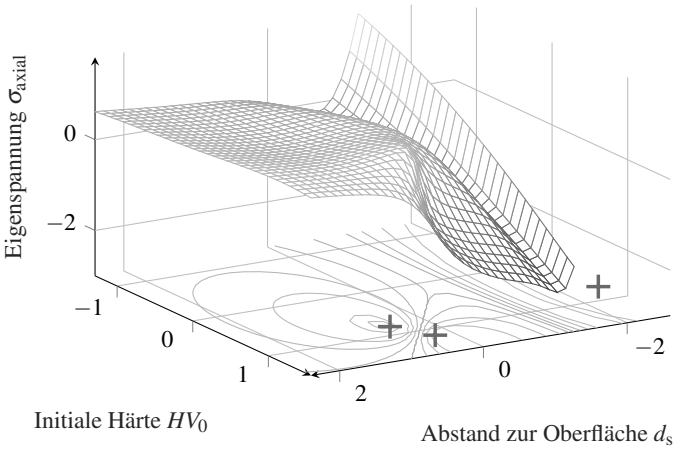



Bild 7: Kennfläche des TS-Modells (Gitter) und Höhenlinien der Zugehörigkeitsfunktionen mit Lage der Prototypen (+) für $c = 3, \nu = 1,2$ in Abhängigkeit von initialer Härte $HV_0$ und Abstand von der Oberfläche $d_s$ bei konstanten Schittparametern $f = 0,25$ mm und $v_c = 175$ m/min

## 4.4 Diskussion

Wie in den Ergebnissen zu erkennen, eignen sich TS-Modelle gut für den Einsatz als Prädiktionsmodelle für den Eigenspannungstiefenverlauf. Dagegen sind die Ergebnisse der globalen Regressionsmodelle trotz ähnlicher Parameteranzahl deutlich schlechter. Die TS-Modelle sind in der Lage, die Nichtlinearitäten des Prozesses besser abzubilden und eine bessere out-of-sample-Prädiktionsgüte zu erzielen.

Bei der Variablenselektion in Tabelle 2 lässt sich erkennen, dass die Eingangsgröße $v_c$ mit der SWR gar nicht, bzw. ausschließlich mit LASSO in den Interaktionstermen selektiert wurde. Diese Beobachtung deckt sich mit dem Ergebnissen anderer Veröffentlichungen, bspw. [3], dass die Schnittgeschwindigkeit einen geringen Einfluss auf die axialen Eigenspannungen hat. Auch in den Koeffizienten der lokalen Teilmodelle der TS-Modelle hat $v_c$ einen deutlich geringeren Einfluss als die restlichen Regressoren. Außerdem wurden bei der SWR Terme höherer Ordnung vorwiegend in den Größen $HV_0$ und $d_s$ selektiert. Wie in Abschnitt 4.1 beschrieben, sind bei diesen Größen die meisten Stufen in der Datenbasis vorhanden, wodurch die Modellierung von nichtlinearem Verhalten erst möglich wird.

Die Approximationsgüte der „globalen“ Regressionsmodelle lässt sich nur durch Hinzunahme von Termen höherer Ordnung verbessern. Dies geht bei Polynomen allerdings mit der Zunahme des oszillatorischen Verhaltens zwischen den Stützstellen einher. Die TS-Modelle hingegen approximieren lokal mit Modellen erster Ordnung und interpoliert zwischen diesen, was bei dem untersuchten Fallbeispiel zu deutlich besseren Ergebnissen geführt hat. Beispielsweise wird mit TS-Modellen der steile Verlauf in den Randbereichen gut approximiert, siehe Bild 7.

# 5 Zusammenfassung und Ausblick

In diesem Beitrag werden datengetriebene Modellierungsansätze untersucht, um einen Eigenspannungstiefenverlauf beim Hartdrehen in Abhängigkeit von den Stell- und Prozessgrößen prädizieren zu können. Dazu wurden „globale“ Regressionsmodelle mit „lokalen“ TS-Modellen unter Berücksichtigung der

Generalisierungsfähigkeit der Modelle verglichen. Das TS-Modell hat sich, trotz spärlicher Datenlage und Unsicherheiten in den Messgrößen als ein geeigneter Modellansatz für diesen Prozess erwiesen. Bei ähnlicher Parameteranzahl lässt sich die Modellgüte wesentlich erhöhen. Insbesondere im Hinblick auf die Verwendung des Prädiktionsmodells für Steuerungs- und Regelungsaufgaben zur Einstellung eines gewünschten Eigenspannungszustandes ist die Modellstruktur mit den lokal-affinen Teilmodellen vorteilhaft.

Es ist zu betonen, dass in diesem Beitrag nur ein Teil der Größen des Hartdrehprozesses untersucht wurde. Zukünftig sollen die Modelle um weitere Eingangs- und Ausgangsgrößen wie Schnittiefe, Rauheit und Härte erweitert werden, um den Randschichtzustand genauer und umfassender modellieren zu können. Für eine Erweiterung der Datenbasis müssen hierzu neue Proben hergestellt und Experimente durchgeführt werden.

## Danksagung

Dieser Beitrag wurde durch die Deutsche Forschungsgemeinschaft (DFG) gefördert – Projektnummer 401792249 (GZ: KR3795/8-1; NI1327/22-1; ZI1296/2-1). Dem Institut für spanende Fertigung der TU Dortmund wird für die Herstellung des Probenmaterials gedankt.

## Literatur

[1] P. J. Arrazola, T. Özel, D. Umbrello, M. Davies, und I. S. Jawahir „Recent advances in modelling of metal machining processes“, *CIRP Annals*, Bd. 62, Nr. 2, S. 695–718, 2013.

[2] J. Rech und C. Lescalier „Surface Integrity in Hard Turning“, In *Recent Advances in Integrated Design and Manufacturing in Mechanical Engineering* (G. Gogu, D. Coutellier, P. Chedmail, P. Ray, Hrsg.), Springer Netherlands, S. 251–260, 2003.

[3] Y. Matsumoto, F. Hashimoto, und G. Lahoti „Surface Integrity Generated by Precision Hard Turning“, *CIRP Annals - Manufacturing Technology*, Bd. 48, Nr. 1, S. 59–62, 1999.

[4] M. E. Fitzpatrick, A.T. Fry, P. Holdway, F. A. Kandil, J. Shackleton, L. Suominen „Determination of Residual Stresses by X-ray Diffraction - Issue 2“, *Measurement Good Practice Guide*, Nr. 50, 2005.

[5] S. Mittal, C. R. Liu „A method of modeling residual stresses in superfinish hard turning“, *Wear*, Bd. 218, Nr. 1, S. 21–33, 1998.

[6] C. A. van Luttervelt et al. „Present Situation and Future Trends in Modelling of Machining Operations Progress Report of the CIRP Working Group ‘Modelling of Machining Operations’“ *CIRP Annals* Bd. 47, Nr. 2, S. 587–626, 1998.

[7] J. Wang, D. Zhang, B. Wu, und M. Luo „Numerical and Empirical Modelling of Machining-induced Residual Stresses in Ball end Milling of Inconel 718“, *Procedia CIRP*, Bd. 58, S. 7–12, 2017.

[8] A. Garg, Y. Bhalerao, K. Tai „Review of empirical modelling techniques for modelling of turning process“, *International Journal of Modelling, Identification and Control*, Bd. 20, Nr. 2, S. 121–129, 2013.

[9] D. Umbrello, G. Ambrogio, L. Filice, und R. Shivpuri „An ANN approach for predicting subsurface residual stresses and the desired cutting conditions during hard turning“, *Journal of Materials Processing Technology*, Bd. 189, Nr. 1–3, S. 143–152, 2007.

[10] Q. Zhang, M. Mahfouf, L. Leon, S. Boumaiza, J. R. Yates, C. Pinna, R. J. Greene „Prediction of Machining Induced Residual Stresses in Aluminium Alloys Using a Hierarchical Data-Driven Fuzzy Modelling Approach“, IFAC Proceedings Volumes, Bd. 42, Nr. 23, S. 231–236, 2009.

[11] Kahl, M., Kroll, A., Kästner, R., Sofsky, M. „Zur automatisierten Auswahl signifikanter Regressoren für die Identifikation eines dynamischen Ladedruckmodells“ *Proceedings of the 24. Workshop Computational Intelligence*, S. 33-55, 2014.

[12] N. R. Draper, H. Smith *Applied Regression Analysis*, 3. Auflage. New York: Wiley, S. 327–368, 1998.

[13] R. Tibshirani „Regression Shrinkage and Selection Via the Lasso“, *Journal of the Royal Statistical Society, Series B*, Bd. 58, Nr.1, S. 267–288, 1994.

[14] Guyon, I., Elisseeff, A. „An introduction to variable and feature selection“, *Journal of machine learning research 3*, Bd. 3, 1157-1182, 2003.

[15] O. Nelles *Nonlinear System Identification*, New York: Springer, S. 309–340, 2001.

[16] R. Kohavi „A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection“, *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Bd. 2, S. 1137-1143, 1995.

[17] Lebsanft, M., Tiffe, M., Zabel, A., Zinn, W., Biermann, D., Scholtes, B. „Residual Stresses in Different Heat Treated Workpieces after Turning“, *Advanced Materials Research*, Bd. 996, S. 652–657, 2014.

[18] A. Kroll „On choosing the fuzziness parameter for identifying TS models with multidimensional membership functions“, *Journal of Artificial Intelligence and Soft Computing Research*, Bd. 1, Nr. 4, S. 283-300, 2011.

# Anwendung des Dynamic Parallel Distributed Compensation Verfahrens an einem thermoelektrischen Prozess

Alessio Cavaterra, Steven Lambeck

FB Elektrotechnik und Informationstechnik, Hochschule Fulda
Leipziger Str. 123, 36037 Fulda
E-Mail: {alessio.cavaterra,steven.lambeck}@et.hs-fulda.de

## 1 Einführung

Takagi-Sugeno Fuzzy Modelle (TS) eignen sich zur Modellierung einer Vielzahl nichtlinearer dynamischer Systeme und ermöglichen weiterhin einen anschließenden Reglerentwurf. Häufig wird dabei der Entwurf von Zustandsrückführungen mit Hilfe des Parallel Distributed Compensation (PDC) Verfahren angewendet. Hierbei wird die Prämisse des Takagi-Sugeno Fuzzy Modells im Regelgesetz übernommen und für jede Regel eine statische Zustandsrückführung entworfen [1].

In den Lehrbüchern finden sich zahlreiche Vorschläge zum Entwurf von Zustandsrückführungen nach dem PDC-Prinzip, welche auf bekannte Methoden wie die Polvorgabe, Riccati-Entwurf, uvm. zurückgreifen. Der Schwerpunkt liegt hierbei häufig auf der Stabilisierung bzw. auf der Sicherstellung der Sollwertfolge eines geregelten nichtlinearen Systems. In der Praxis sind überdies eine gute Störgrößenunterdrückung sowie Robustheit gegenüber Parameterunsicherheiten gefordert. Diese Anforderungen kann eine einfache PDC-Zustandsrückführung nicht gänzlich erfüllen, weshalb häufig verschiedene Erweiterungen im Regelgesetz implementiert werden.

Um die genannten Anforderungen zu erfüllen wird dieser Beitrag den Entwurfsweg für einen dynamischen Regler nach dem PDC Verfahren über lineare Matrixungleichungen vorstellen. Der entworfene Regler wird an einem

vorhandenem Versuchsstand erprobt. Bei dem Prozess handelt es sich um eine Temperaturregelstrecke mit thermoelektrischen Halbleiterelementen (sogenannte „Peltier-Elemente“). Die Arbeit liefert damit ein weiteres Anwendungsbeispiel für CI-Methoden in der Praxis.

## 2 Thermoelektrisches Modell

Im Rahmen des Forschungsprojektes „Dezent - Dezentrale Klimageräte“ am Fachbereich Elektrotechnik und Informationstechnik der Hochschule Fulda wird ein mobiles, kombiniertes Be- und Entfeuchtungsgerät entwickelt (HA-Projekt-Nr.: 514/16-26, „Hessen ModellProjekte“). Anstelle einer häufig verwendeten Kompressorkühlung zur Raumluftentfeuchtung wird eine Oberflächenkühlung mit Peltier-Elementen eingesetzt.
Die Funktionsweise dieser thermoelektrischen Halbleiterelemente beruht im Wesentlichen auf dem Seebeck- bzw. dem Peltier-Effekt: ein stromdurchflossener thermoelektrischer Halbleiter bewirkt einen Wärmetransport, sodass eine Wärmesenke (Kaltseite) und eine Wärmequelle (Warmseite) entsteht [2]. In der Praxis sind Peltier-Elemente beispielsweise in Kfz-Kühlboxen zu finden.
Weil eine detaillierte Modellbildung der thermoelektrischen Eigenschaften den Rahmen des Beitrages sprengen würde, sei im Folgenden nur das nichtlineare Differentialgleichungssystem (DGL-System) aufgeführt. Das Technologieprinzip des Gerätes ist in Bild 1 skizziert. Bei dem folgenden Modell handelt es sich um eine Erweiterung des thermoelektrischen Modells von Lineykin und Ben-Yaakov [3].

$$\begin{bmatrix} \dot{\vartheta}_a \\ \dot{\vartheta}_e \\ \dot{\vartheta}_{kkc} \\ \dot{\vartheta}_{kkh} \end{bmatrix} = \begin{bmatrix} \frac{1}{C_t}\left(\frac{\vartheta_{kkc}-\vartheta_a}{\theta_{cont}} - \alpha_m(\vartheta_a+T_0)I_1 + \frac{I_1^2 R_m}{2} + \frac{\vartheta_e-\vartheta_a}{\theta_m}\right) \\ \frac{1}{C_t}\left(\frac{\vartheta_{kkh}-\vartheta_e}{\theta_{cont}} + \alpha_m(\vartheta_e+T_0)I_1 + \frac{I_1^2 R_m}{2} + \frac{\vartheta_a-\vartheta_e}{\theta_m}\right) \\ \frac{1}{C_{hs,1}}\left(\frac{\vartheta_{amb}-\vartheta_{kkc}}{\theta_{iso,1}} - \frac{2(\vartheta_{kkc}-\vartheta_a)}{\theta_{cont}}\right) \\ \frac{1}{C_{hs,2}}\left(\frac{\vartheta_{amb}-\vartheta_{kkh}}{\theta_{iso,2}} - \frac{2(\vartheta_e-\vartheta_{kkh})}{\theta_{cont}}\right) \end{bmatrix} \quad (1)$$

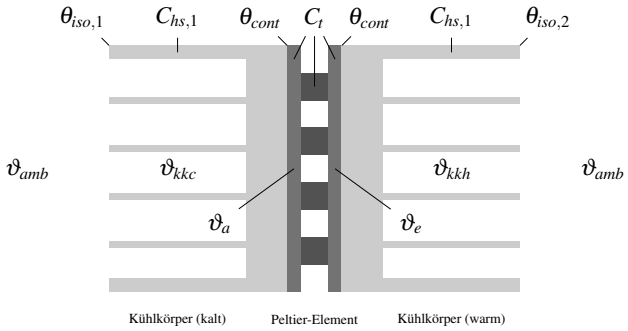



Bild 1: Technologieprinzip des thermoelektrischen Apparates.

Das DGL-System entspricht strukturell der Gleichung $\dot{x} = f(x,u,d)$ und wird später in die TS Zustandsraumdarstellung übertragen. Im Zustandsvektor $x \in \mathbb{R}^{n\times 1}$ werden $n = 4$ Temperaturen zusammengefasst, die in dieser Reihenfolge die Temperatur der Wärme-absorbierenden Seite $\vartheta_a$, die Temperatur der Wärme-emittierenden Seite $\vartheta_e$, die Temperatur des Oberflächenkühlers $\vartheta_{kkc}$ und die Temperatur des Wärmetauschers $\vartheta_{kkh}$ darstellen. Die beiden letzten Zustandsgrößen werden als „Kaltseitentemperatur“ und „Warmseitentemperatur“ bezeichnet. Der elektrische Strom $I_1$ stellt die Stellgröße $u$ des Systems dar. Die Umgebungstemperatur $\vartheta_{amb}$ wirkt ebenfalls als Eingangsgröße auf das System. Sie kann nicht beeinflusst werden und wird mit $d$ abgekürzt. Die genannten Temperaturen sind zusammen mit den Wärmespeicherkapazitäten und den Wärmewiderständen in Bild 1 zu finden, wobei die Parameter des Peltier-Elementes $\alpha_m$, $\theta_m$ und $R_m$ sowie der Strom $I_1$ nicht aufgeführt sind. Für den Reglerentwurf ist lediglich die Kaltseitentemperatur als Regelgröße wichtig. Die Ausgabegleichung lautet demnach $y = Cx = \vartheta_{kkc}$. Dies impliziert $C = [0\,0\,1\,0]$. Wird zusätzlich die Warmseitentemperatur $\vartheta_{kkh}$ gemessen, kann man zeigen, dass so eine schwache Beobachtbarkeit des Gesamtsystems gewährleistet wird [4]. Dies macht den Einsatz eines Extended-Kalman-Filters (EKF) möglich. Das EKF wird zur Schätzung des Zustandsvektors $x$ benötigt, der wiederum für eine Zustandsregelung notwendig ist.

Mit Hilfe einer Arbeitspunktlinearisierung über die Taylor-Reihe wird ein Takagi-Sugeno Fuzzy Modell abgeleitet. Vier Arbeitspunkte gemäß $u_{ap,i} = 2,4,6,8\,\mathrm{A}$ mit $i = 1,2,3,4$ bei einer konstanten Umgebungstemperatur $\vartheta_{amb,ap} = 22\,^{\circ}\mathrm{C}$ führen zu den stationären Temperaturen:

$$\vartheta_{a,ap,i} = 13.58, 7.29, 2.92, 0.29\,^{\circ}\mathrm{C} \tag{2}$$

$$\vartheta_{e,ap,i} = 25.48, 30.10, 35.87, 42.82\,^{\circ}\mathrm{C} \tag{3}$$

$$\vartheta_{kkc,ap,i} = 15.57, 10.76, 7.43, 5.43\,^{\circ}\mathrm{C} \tag{4}$$

$$\vartheta_{kkh,ap,i} = 22.78, 23.81, 25.10, 26.64\,^{\circ}\mathrm{C}. \tag{5}$$

Die angegebenen Temperaturen wurden numerisch aus dem stationären DGL-System (1) bestimmt. Das TS Modell besteht folglich aus $r = 4$ Regeln. Wie aus dem DGL-System ersichtlich ist, stellt der elektrische Strom die charakteristische Nichtlinearität dar, sodass für die Planungsvariable $z = I_1$ gilt. Die stets positiven Fuzzy-Basisfunktionen $h_i(z)$ erfüllen die konvexe Summeneigenschaft $\sum_i^r h_i(z) = 1$ aufgrund der Normierung über

$$h_i(z) = \frac{\mu_i(z)}{\sum_i^r \mu_i(z)}. \tag{6}$$

Die Fuzzy-Zugehörigkeitsfunktionen $\mu_i(z)$ werden als Gaußglocken ausgewählt. Deren Erwartungswerte entsprechen den Arbeitspunktströmen $u_{ap,i}$. Die Varianzen sind durchweg zu $\sigma_i = 0.5$ gewählt.

$$\mu_i(z) = e^{-\frac{(z-u_{ap,i})^2}{2\sigma_i^2}} \tag{7}$$

Somit kann eine Struktur für das TS Modell angegeben werden, wenn zuvor $\xi(t) = x(t) - x_{ap}$ und $\omega(t) = u(t) - u_{ap}$ definiert werden.

$$\dot{\xi}(t) = \sum_{i=1}^{4} h_i(z)\,(A_i\xi(t) + b_i\omega(t)) \tag{8}$$

Die Ausgabegleichung hat sich nicht verändert und lautet $y = Cx = \vartheta_{kkc}$. An dieser Stelle wird darauf hingewiesen, dass die Einflüsse durch die Umgebungstemperatur $\vartheta_{amb}$ in dieser Differentialgleichung ignoriert werden. Das stellt

kein Problem dar, wie im nächsten Abschnitt besprochen wird. Die Systemmatrizen $A_i$ sowie die Steuervektoren $b_i$ der Teilmodelle lauten dann

$$A_1 = \begin{bmatrix} -0.3744 & 0.0671 & 0.3022 & 0 \\ 0.0671 & -0.3641 & 0 & 0.3022 \\ 0.0050 & 0 & -0.0066 & 0 \\ 0 & 0.0025 & 0 & -0.0111 \end{bmatrix} \tag{9}$$

$$A_2 = \begin{bmatrix} -0.3796 & 0.0671 & 0.3022 & 0 \\ 0.0671 & -0.3590 & 0 & 0.3022 \\ 0.0050 & 0 & -0.0066 & 0 \\ 0 & 0.0025 & 0 & -0.0111 \end{bmatrix} \tag{10}$$

$$A_3 = \begin{bmatrix} -0.3847 & 0.0671 & 0.3022 & 0 \\ 0.0671 & -0.3538 & 0 & 0.3022 \\ 0.0050 & 0 & -0.0066 & 0 \\ 0 & 0.0025 & 0 & -0.0111 \end{bmatrix} \tag{11}$$

$$A_4 = \begin{bmatrix} -0.3899 & 0.0671 & 0.3022 & 0 \\ 0.0671 & -0.3487 & 0 & 0.3022 \\ 0.0050 & 0 & -0.0066 & 0 \\ 0 & 0.0025 & 0 & -0.0111 \end{bmatrix} \tag{12}$$

$$b_1^T = \begin{bmatrix} -0.6615 & 0.8460 & 0 & 0 \end{bmatrix} \tag{13}$$

$$b_2^T = \begin{bmatrix} -0.5684 & 0.9348 & 0 & 0 \end{bmatrix} \tag{14}$$

$$b_3^T = \begin{bmatrix} -0.4802 & 1.0266 & 0 & 0 \end{bmatrix} \tag{15}$$

$$b_4^T = \begin{bmatrix} -0.3965 & 1.1215 & 0 & 0 \end{bmatrix} \tag{16}$$

## 3 Reglerentwurf

Die Anforderungen des ersten Abschnittes erfordern den Einsatz eines Integrators in der Regelung. Dieser eliminiert eine bleibende Regelabweichung, welche durch Modellunsicherheiten, sprungförmige Sollwertänderungen in $r(t)$ und sprungförmige Störungen entstehen kann. Damit ist auch der Verzicht der

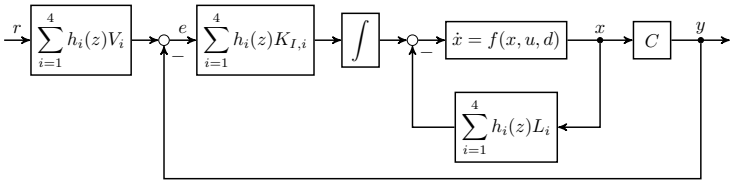



Bild 2: Blockschaltbild der Regelungsstruktur (ohne Zustandsbeobachter).

Störmatrix durch $\vartheta_{amb}$ in der Gleichung (8) erklärt. Durch die Einführung eines neuen Zustandsvektors $x_e(t)^T = \left[\xi(t)^T \; e(t)^T\right]$ ergeben sich so die erweiterten Darstellungen der Systemmatrizen und Steuervektoren.

$$A_{e,i} = \begin{bmatrix} A_i & 0^{(n\times p)} \\ -C & 0^{(p\times p)} \end{bmatrix} \quad B_{e,i} = \begin{bmatrix} b_i \\ 0^{(p\times m)} \end{bmatrix} \quad L_{e,i} = \begin{bmatrix} L_i & -K_{I,i} \end{bmatrix} \tag{17}$$

Die Anzahl der Regelgrößen $p$ entspricht der Anzahl der Steuergrößen $m$: $p = m = 1$. Es liegt ein quadratisches System vor. Die Struktur der Regelung ist im Bild 2 skizziert. Die Einführung von $L_{e,i}$ erlaubt zudem eine verkürzte Schreibweise des Regelgesetzes

$$u(t) = -\sum_{i=1}^{4} h_i(z)\left(L_{e,i}x_e(t) - V_i r(t)\right). \tag{18}$$

Das Regelgesetz nutzt die selben Fuzzy-Basisfunktionen $h_i(z)$ des TS Modells. Folglich stellt es ein dynamisches PDC-Regelgesetz (DPDC) mit zusätzlichem Vorfilter dar. Die Vorfilter $V_i$ erlauben schnelle Arbeitspunktwechsel bei Sollwertsprüngen in $r(t)$. Jeder $i$-te Vorfilter wird dabei nach der üblichen Beziehung $V_i = (C(b_iL_i - A_i)^{-1}b_i)^{-1}$ bestimmt.

Die Berechnung der integralen Zustandsrückführung $L_{e,i}$ kann nun auf unterschiedlichen Wegen erfolgen. Zum Beispiel können über einen LQ-Entwurf die einzelnen $L_{e,i}$ bestimmt und die entsprechenden Zustandsrückführungen $L_i$ sowie Integralverstärkungen $K_{I,i}$ extrahiert werden. Allerdings muss der Anwender hierbei beachten, dass stabile Teilsysteme keineswegs ein stabiles Gesamtsystem implizieren [5].

Dieses Problem lässt sich mit Hilfe linearer Matrixungleichungen (LMI) elegant umgehen. Ausgehend von der Lyapunov-Stabilitätstheorie lässt sich die Suche nach Reglerparametern für einen stabilen Regelkreis als „LMI-Problem“ formulieren. LMI-Probleme sind *convex feasability problems* [6], die mit Solvern für konvexe Optimierungsprobleme gelöst werden können. LMI-Probleme können erweitert werden, um beispielsweise Stellgrößenbeschränkungen zu berücksichtigen. Im vorliegendem Fall muss vermieden werden, dass der Strom $I_1$ des thermoelektrischen Systems einen festgelegten Wert $I_{1,max}$ überschreitet, da sonst die Verlustleistung des Gerätes ansteigt. Das folgende LMI-Problem ermöglicht eine stabile Regelung unter Berücksichtigung einer vom Anfangszustand $x_{e,0}$ abhängigen Stellgrößenbeschränkung.
*Finde* $X > 0$ *und* $M_i$ *für* $i = 1 \dots r$, *sodass*

$$-XA_{e,i}^T - A_{e,i}X + M_i^T B_{e,i}^T + B_{e,i}M_i \geq 0 \tag{19}$$

$$\begin{bmatrix} 1 & x_{e,0}^T \\ x_{e,0} & X \end{bmatrix} \geq 0 \tag{20}$$

$$\begin{bmatrix} X & M_i^T \\ M_i & \mu^2 I \end{bmatrix} \geq 0 \tag{21}$$

*und für* $i < j$*:*

$$\begin{aligned} -XA_{e,i}^T - A_{e,i}X - XA_{e,j}^T - A_{e,j}X \\ +M_j^T B_{e,i}^T + B_{e,i}M_j + M_i^T B_{e,j}^T + B_{e,i}M_j \geq 0. \end{aligned} \tag{22}$$

Über $L_{e,i} = M_i X^{-1}$ lassen sich die benötigten Reglerparameter extrahieren. Der Anfangszustand wird zu $x_{e,0} = [22\,^\circ\mathrm{C}\ 22\,^\circ\mathrm{C}\ 22\,^\circ\mathrm{C}\ 22\,^\circ\mathrm{C}\ 0\,^\circ\mathrm{C}]^T$ gewählt. Eine Regelabweichung sei zu Beginn nicht vorhanden $e(t=0) = 0\,^\circ\mathrm{C}$. Aus einer stationären Analyse des DGL-Systems (1) wurde $I_{1,max} = 10\,\mathrm{A}$ numerisch bestimmt. Damit wird $\mu = I_{1,max}^2 = 100$ gewählt. Die Ungleichungen (19) bis (22) findet man mitsamt Beweis in [5] wieder. Sie wurden mit dem numerischen Solver *SDPT3-4* für MATLAB gelöst [7].

Die gesuchte symmetrische Matrix $X$ ist positiv definit und deutet auf eine stabiles System hin. Sie lautet

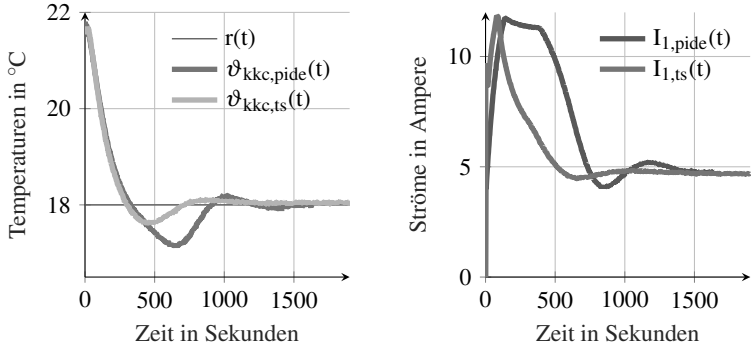
$$X = 1E6 \cdot \begin{bmatrix} 0.7061 & 0.0276 & -0.0094 & 0.0203 & 0.1026 \\ 0.0276 & 1.4052 & -0.0100 & 0.2822 & -0.5127 \\ -0.0094 & -0.0100 & 0.0015 & -0.0121 & 0.0325 \\ 0.0203 & 0.2822 & -0.0121 & 0.3252 & -0.1161 \\ 0.1026 & -0.5127 & 0.0325 & -0.1161 & 4.6088 \end{bmatrix} > 0. \tag{23}$$

## 4 Messergebnisse

Die entworfene integrale Zustandsrückführung mit Vorfilter wurde am thermoelektrischen Prozess erprobt und mit einem PI-Regler verglichen. Dessen Reglerparameter sind zuvor mit einem Differential-Evolutions-Algorithmus (s. [8]) am nichtlinearen DGL-System (1) optimiert worden, wobei nach 50 Generationen abgebrochen wurde. Als Kostenfunktion diente die Summe der Fehlerquadrate der Regelabweichung. Um den PI-Regler speziell für ein festgelegtes Szenario zu optimieren, wurde der Anfangszustand der Kaltseitentemperatur auf $\vartheta_{kkc}(t=0) = 22\,^{\circ}\mathrm{C}$ und der Sollwert auf $r = 18\,^{\circ}\mathrm{C}$ festgelegt. Dieses Szenario wurde auch im Versuch reproduziert, wie man dem Bild 3 entnehmen kann.

Beide Temperaturegelkreise regeln den Prozess mit circa 260 s an; der DPDC-geregelte Prozess weist jedoch eine weitaus geringere Einschwingzeit als der PI-geregelte Prozess auf und schwingt weniger unter, als die konventionell geregelte Variante. Der PI-Regler treibt bis circa 500 s einen energetisch-ineffizienten Strom durch die Peltier-Elemente. Es kommt zu einer starken Unterschwingung. Der DPDC-Regler arbeitet in dieser Hinsicht effizienter.

Die Software-seitigen Stellgrößenbegrenzungen für $0\,\mathrm{A} < I_1 \leq 10\,\mathrm{A}$ wurden vor den Versuchsdurchführungen ausgeschaltet, um die Regler unter Realbedingungen – welche nicht vollständig beim Reglerentwurf berücksichtigt werden konnten – zu überprüfen. Beide Regler verletzen die Beschränkung von $I_{1,max} = 10\,\mathrm{A}$, würden aber zu keiner Beschädigung der Peltier-Elemente führen. Eine kurzzeitige Überhöhung von $I_{1,max}$ kann einen positiven Kühleffekt hervorrufen [9].




(a) Kaltseitentemperaturen (b) Stellgrößen

Bild 3: Sprungantworten des Temperaturregelkreises mit DPDC-Regler und mit optimierten PI-Regler für einen Sprung auf $r(t = 0) = 18\,°C$.

## Literatur

[1] A. Kroll. „Computational Intelligence. Probleme, Methoden und technische Anwendungen“. *De Gruyter Studium* 2016.

[2] H. J. Goldsmid. „Introduction to Thermoelectricity“. *Springer-Verlag Berlin Heidelberg* 2016.

[3] S. Lineykin und S. Ben-Yaakov. „Modeling and Analysis of Thermoelectric Modules“. *IEEE Transactions on Industry Applications*, 43, S. 505-512. 2007.

[4] J. Adamy. „Nichtlineare Regelungen und Systeme“. *Springer-Verlag Berlin Heidelberg* 2014.

[5] K. Tanaka und H.O. Wang. „Fuzzy Control Systems Design and Analysis: A Linear Matrix Inequality Approach“. *John Wiley & Sons* 2001.

[6] S. Boyd und L. Vandenberghe. „Convex Optimization“. *Cambridge University Press* 2004.

[7] K. C. Toh et al. „On the implementation and usage of SDPT3 - a Matlab software package for semidefinite-quadratic-linear programming, version 4.0“. In: *Handbook on semidefinite, conic and polynomial optimization*, S.715-754. Springer, Boston, MA. 2012

[8] R. Storn und K. Price. „Differential Evolution - a simple and efficient heuristic for global optimization over continuous spaces“.*Journal of Global Optimization*, 11, S. 341–359. 1997.

[9] G.J. Snyder et al. „Supercooling of Peltier Cooler using a current pulse“. *Journal of Applied Physics*, Bd.92, Nr.3, S. 1564-1569. 2002.

# Schätzung von datenbasierten lokal-linearen Modellen auf der Grundlage von LOLIMOT für den systematischen Entwurf von lokal-linearen Zustandsreglern

Tim Voigt, Martin Kohlhase

Center for Applied Data Science Gütersloh, FH Bielefeld
Interaktion 1, 33619 Bielefeld
E-Mail: {tim.voigt, martin.kohlhase}@fh-bielefeld.de

## Kurzfassung

In dieser Arbeit werden basierend auf lokal-linearen Modellen lokal-lineare Zustandsregler entworfen. Zum einen werden lokal-lineare Zustandsregler für die Vorsteuerung (Regelung am Modell) eingesetzt. Zum anderen werden diese im Regler eingesetzt, wobei die Stellgrößenbegrenzung in beiden Ansätzen berücksichtigt wird. Die lokal-linearen Zustandsregler werden mit normierten Gaussfunktionen überlagert. Es ergeben sich sogenannte Fuzzy-Zustandsregler. Der systematische Entwurf und die erzielten Ergebnisse werden an einem Mehrgrößensystem gezeigt und erläutert.

## 1 Einführung

Der Automatisierungsgrad von Prozessen, Produkten oder Maschinen, z. B. im Bereich der Produktion nimmt stetig zu. Gleichzeitig werden die Prozesse in den Produkten oder Maschinen immer komplexer. Damit einhergehend steigen die Anforderungen an die Steuerungen und Regelungen dieser Prozesse, um auf dieser Grundlage intelligente bzw. smarte Produkte und Maschinen zu entwickeln. Aufgrund dessen wird der Steuerungs- und Regelungsentwurf

für den herkömmlichen Ingenieur zunehmend aufwendiger und schwieriger. In den meisten Fällen sind die zu steuernden oder zu regelnden Prozesse nichtlinear und sie besitzen mehrere Ein- und Ausgangsgrößen. Die Ausgangsgrößen stellen häufig die zu regelnden Größen dar, die untereinander durch Wechselwirkungen im Prozess verkoppelt sind. Der Steuerungs- und Regelungsentwurf für nichtlineare Mehrgrößensysteme stellt den Entwicklungsingenieur vor einige Herausforderungen.

In dieser Arbeit wird eine durchgängige Methodik für den (Vor-)Steue-rungs- und Reglerentwurf vorgestellt, der folgende Aspekte vereint:

- Entwurf einer Vorsteuerung und einer Regelung (Two-Degrees-of-Freedom), um das Führungsverhalten und das Störverhalten getrennt voneinander vorzugeben,
- Intuitive Einstellung der Vorsteuerung durch den Applikationsingenieur,
- Berücksichtigung der Stellgrößenbegrenzung in der Vorsteuerung,
- Intuitive Parametrierung des (Mehrgrößen-)Reglers,
- Sollwertfolge bei sprungförmiger Störgröße und
- Anti-Windup-Konzept.

Der Entwurf basiert auf einer Abbildung der Regelstrecke mit Hilfe von lokallinearen Modellen. Dazu wird das Programmpaket LOLIMOT (LOcal LInear MOdel Tree) verwendet. Die Grundlagen dieser Modellierungsmethode und die darauf aufbauenden Anpassungen zur Verbesserung der Modellgüte werden im folgenden Kapitel erläutert. Anschließend erfolgt eine Beschreibung der Struktur und Entwurfsmethodik für den entwickelten Fuzzy-Zustandsregler. Der vollständige Modellierungs- und Entwurfsprozess wird dann an einem Laborversuch demonstriert und die erzielten Ergebnisse vorgestellt. Abschließend folgt eine kurze Zusammenfassung und ein Ausblick.

# 2 Modellbildung

## 2.1 LOLIMOT

Die datenbasierte Modellbildung der Regelstrecke erfolgt auf Grundlage des Programmpakets LOLIMOT (LOcal LInear MOdel Tree) [8]. LOLIMOT bildet nichtlineare Prozesse durch Überlagerung von mehreren lokal-linearen Teilmodellen ab und bietet somit den Vorteil einer leichten Interpretierbarkeit. Da die einzelnen Teilmodelle im Allgemeinen einen Offset besitzen, handelt es sich strenggenommen um lokal-affine Teilmodelle. Im Weiteren werden die Modelle dennoch als lokal-linear bezeichnet, da sich diese Begrifflichkeit in vielen Veröffentlichungen wiederfindet.

Ein lokal-lineares Modell für das statische Verhalten eines Prozesses mit dem Modellausgang

$$\hat{y} = \sum_{i=1}^{M} (w_{i0} + w_{i1}x_1 + \ldots + w_{in}x_n)\Phi_i(\mathbf{z}). \tag{1}$$

besteht aus $M$ linearen Teilmodellen. Der Modellausgang ergibt sich aus der Summe der einzelnen Teilmodelle, die mit $\Phi_i(\mathbf{z})$ gewichtet werden. Dabei ist $\mathbf{z}$ ein arbeitspunktbeschreibender Vektor. Jedes dieser Teilmodelle besitzt $n$ Eingangsgrößen $\mathbf{x} = [x_1\ x_2\ \ldots\ x_n]^T$ und $n+1$ Parameter $\mathbf{w}_i = [w_{i0}\ w_{i1}\ \ldots\ w_{in}]^T$. Die Regressoren $\mathbf{z}$ und $\mathbf{x}$ setzen sich aus den insgesamt zur Verfügung stehenden Eingangsgrößen zusammen.

Diese Unterteilung in einen $z$-Regressor, der den Arbeitspunkt festlegt und einen $x$-Regressor, der als Eingangsvektor der Teilmodelle dient, hat insbesondere bei vielen Eingangsgrößen den Vorteil, das nicht alle Eingangsgrößen in die Teilmodelle eingehen müssen. So kann eine Eingangsgröße, die die Nichtlinearität stark beeinflusst nur in den $z$-Regressor einfließen und erhöht damit nicht die Dimensionalität der Teilmodelle. Im folgenden wird vereinfacht davon ausgegangen, dass alle Einganggrößen sowohl in den $z$-, als auch in den $x$-Regressor eingehen und damit beide Regressoren die Dimension $n$ besitzen.

Jedes der Teilmodelle besitzt eine Aktivierungsfunktion, die in Abhängigkeit des $z$-Regressors (Arbeitspunkt) den Ausgang eines jeden Teilmodells gewichtet. Jede Aktivierungsfunktion

$$\Phi_i(\mathbf{z}) = \frac{\mu_i(\mathbf{z})}{\sum_{m=1}^{M} \mu_m(\mathbf{z})} \tag{2}$$

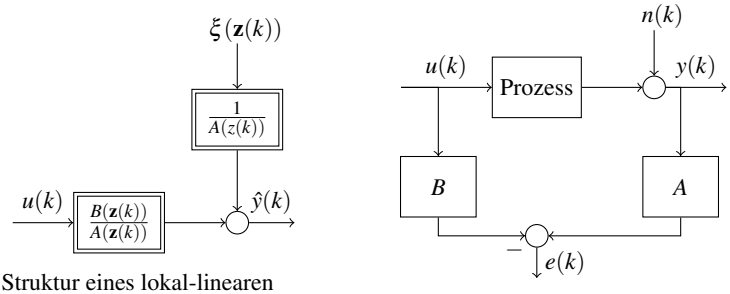
ist so normiert, so dass die Summe der Aktivierungsfunktionen in jedem Punkt im Eingangsraum gleich eins ist. Um glatte Übergänge zwischen den einzelnen Teilmodellen zu erzeugen, werden innerhalb der Aktivierungsfunktionen Gauss-Glocken

$$\mu_i(\mathbf{z}) = \exp\left(-\frac{1}{2}\frac{(z_1 - c_{i1})^2}{\sigma_{i1}^2}\right) \cdot \ldots \cdot \exp\left(-\frac{1}{2}\frac{(z_n - c_{in})^2}{\sigma_{in}^2}\right). \tag{3}$$

mit dem Zentrum $c_{ij}$ und der Standardabweichung $\sigma_{ij}$ des i-ten Teilmodells in der j-ten Dimension des $z$-Regressors, verwendet. Diese Parameter ergeben sich aus der von LOLIMOT vorgenommenen Partitionierung des Eingangsraums und aus der jeweiligen Lage und Größe der Bereiche. Zur Partitionierung des Eingangsraums verwendet LOLIMOT einen Konstruktionsalgorithmus, der fortlaufend das schlechteste Teilmodell weiter unterteilt. Dieses kann beispielsweise anhand einer Verlustfunktion, wie der Summe der gewichteten quadratischen Abweichung, beurteilt werden. LOLIMOT beschränkt sich darauf den jeweiligen Teilraum mittig und achsenorthogonal zu teilen. Dabei wird aus allen möglichen Teilungen diejenige ausgewählt, die zu dem geringsten globalen Fehler führt.

Ein großer Vorteil der in Gleichung (1) beschriebenen Struktur ist, das der Modellausgang linear in den Parametern ist. Somit lässt sich zur Parameterschätzung die lineare Methode der gewichteten kleinsten Fehlerquadrate einsetzen. Diese Methode zeichnet sich durch einen besonders geringen Rechenaufwand aus.

Das bisher beschriebene Verfahren dient zur Modellierung des statischen Verhaltens von Prozessen. Im Folgenden sollen allerdings dynamische Prozesse betrachtet werden. In dieser Arbeit wird LOLIMOT für die Erstellung eines initialen dynamischen Systemmodells eingesetzt. Dazu wird eine NARX-




(a) Struktur eines lokal-linearen Teilmodells

(b) Gleichungsfehler-Anordnung

Bild 1: LS-Schätzung eines lokal-linearen Teilmodells in Gleichungsfehler-Anordnung

Struktur (Nichtlinear autoregressiv mit exogenem Eingang (Nonlinear ARX)) des dynamischen Systems zugrunde gelegt, wie sie in Bild 1a zu sehen ist. Dabei ist das Nennerpolynom der Übertragungsfunktion des Modells mit $A$ und das dazugehörige Zählerpolynom mit $B$ bezeichnet. Der Modellausgang des dynamische lokal-lineare Gesamtmodell wird durch eine Einschrittvorhersage in der Form

$$\begin{aligned}\hat{y}(k) = \sum_{i=1}^{M} (&b_{i1}u(k-1) + \ldots + b_{in}u(k-n) \\ &- a_{i1}y(k-1) - \ldots - a_{in}y(k-n) + \xi_i)\Phi(\mathbf{z}(k))\end{aligned} \tag{4}$$

bestimmt. Diese Gleichung, die sich exemplarisch für ein SISO-System (Single-Input-Single-Output) $n$-ter Ordnung ergibt, ist genauso lokal-linear, wie die in Gleichung (1) beschriebene Struktur. Es werden verzögerte Eingangsgrößen mit den Parametern $a_i$ und verzögerte Ausgangsgrößen mit den Parametern $b_i$, sowie ein Offset $\xi$ als Regressoren verwendet.

Diese Modellstruktur, kann in der in Bild 1b dargestellten Gleichungsfehler-Anordnung geschätzt werden. Damit ist der Vorteil der Linearität in den Parametern gegeben. Auf diese Weise ist auch bei dynamischen Systemen die Methode der kleinsten Fehlerquadrate anwendbar. Ein Nachteil der Gleichungsfehler-Anordnung ist, dass die Störung $n(k)$ mit $\mathbf{A}(\mathbf{z})$ gefiltert in den Gleichungsfehler eingeht. Damit wird die übliche Störsignal-Annahme für Prozess- und Messrauschen von weißem Rauschen, hin zu farbigem Rauschen verän-

dert, das stark von der Nenner-Dynamik des Prozesses beeinflusst wird. Ist ein solches Rauschen in den Messdaten vorhanden, entsteht ein systematischer Schätzfehler (Bias) [4].

Anstatt jedes Teilmodell mit der Aktivierungsfunktion zu gewichten, können auch die einzelnen Parameter als veränderlich angesehen werden. Sie könne damit als vom Arbeitspunkt $\mathbf{z}(k)$ abhängig betrachtet werden. Aus Gleichung (4) folgt somit die parameterveränderliche Form

$$\hat{y}(k) = \sum_{i=1}^{n} b_i(\mathbf{z}(k))u(k-i) - \sum_{i=1}^{n} a_i(\mathbf{z}(k))y(k-i) + \xi(\mathbf{z}(k)). \tag{5}$$

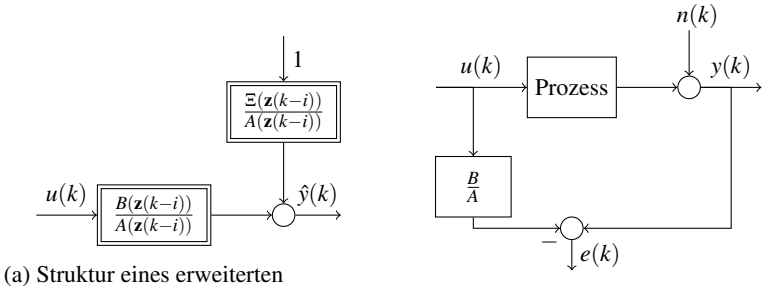
## 2.2 Erweiterung der Modellstruktur

Damit eine bessere Modellgüte erzielt werden kann, wird die Struktur um einen dynamischen Offset mit Zählerdynamik erweitert. Zusätzlich werden die $z$-Regressoren, die den Arbeitspunkt beschreiben, entsprechend der Eingangsgrößen verzögert. Diese Erweiterungen wurden von [10] beschrieben und in Form der Toolbox DYLAMOT (DYnamic Local Affine MOdeling Toolbox) in Matlab implementiert.

Durch die Verzögerung der $z$-Regressoren wird erreicht, dass bei einem Arbeitspunktwechsel die Zuordnung der Modellparameter zu den jeweiligen, verzögerten Ein- und Ausgangsgrößen erhalten bleibt. Unmittelbar nach einem Arbeitspunktwechsel, werden die verzögerten Werte unter den Bedingungen des vergangenen Arbeitspunktes erfasst, so dass für diese auch noch die vergangenen Parameter gültig sind. Durch die Einführung der verzögerten $z$-Regressoren wird dieser Sachverhalt berücksichtigt.

Ein wichtiger Vorteil der eingeführten Verzögerung ist, dass dadurch eine Überführung des lokal-linearen Modells in eine minimalrealisierte Zustandsraumdarstellung möglich ist und durch die minimale Anzahl an Zustandsgrößen der Reglerentwurf vereinfacht wird. Außerdem führt die Verzögerung zu einer verbesserten Modellgüte bei Prozessen mit schnellen Arbeitspunktwechseln und einer Systemordnung von $n \geq 2$ [10].

Eine weitere Anpassung der Struktur betrifft den Offset-Term. In Gleichung (5) wird für jedes lokal-lineare Modell ein einziger Offset-Term $\xi$ angenommen.




(a) Struktur eines erweiterten lokal-linearen Teilmodells

(b) Ausgangsfehleranordnung

Bild 2: Schätzung eines lokal-linearen Teilmodells in Ausgangsfehleranordnung

Durch die Einführung einer Offset-Zählerdynamik wird das Modell flexibler und in die Lage versetzt auch Prozesse mit einer signifikanten Zählerdynamik abzubilden [9].

Das entstandene dynamische lokal-lineare Modell mit verzögerter Parameteränderung und erweitertem Offset, kann somit gemäß folgender Differenzengleichung beschrieben werden:

$$\hat{y}(k) = \sum_{i=1}^{n} b_i(\mathbf{z}(k-i))u(k-i) - \sum_{i=1}^{n} a_i(\mathbf{z}(k-i))\hat{y}(k-i) + \sum_{i=1}^{n} \xi_i(\mathbf{z}(k-i)). \tag{6}$$

Es ergibt sich für jedes Teilmodell eine Struktur nach Bild 2a, die in der Ausgangsfehler-Anordnung (Bild 2b) geschätzt wird. Ein Vorteil dieser Methode ist im Gegensatz zur Gleichungsfehler-Anordnung eine realistische Störsignal-Annahme. Auf diese Weise wird eine biasfreie Parameterschätzung bei Rauschen am Prozessausgang ermöglicht. Allerdings gehen die Parameter nichtlinear in den Ausgangsfehler ein, sodass ein nichtlineares Optimierungsverfahren benötigt wird [4]. Für die Schätzung werden die zuvor mittels des LOLIMOT-Algorithmus ermittelten Aktivierungsfunktionen beibehalten und die zuvor ermittelten Parameter als Startwerte verwendet.

## 2.3 Transformation in die Zustandsraumdarstellung

Um für den Regler- und Vorsteuerungsentwurf auf eine Vielzahl von Entwurfsmethoden zurückgreifen zu können, wird das ermittelte lokal-lineare Modell in eine lokal-lineare Zustandsraumdarstellung transformiert. Diese Transformation kann auch auf Grundlage eines lokal-linearen dynamischen Modells, ohne die im vorherigen Abschnitt beschriebene Verzögerung der $z$-Regressoren erfolgen [3]. Allerdings ergibt sich somit eine Zustandsraumdarstellung mit einer höheren Anzahl an Zustandsgrößen, die strukturbedingt und nicht mehr physikalisch begründbar ist.

Durch die getroffenen Erweiterungen der Modellstruktur lässt sich eine minimale Zustandsraumrealisierung erzeugen, die in Beobachternormalform dargestellt werden kann [10]. Im Weiteren wird vereinfacht davon ausgegangen, dass Systeme ohne Durchgriff betrachtet werden. Das Zustandsraummodell wird somit für SISO-Systeme durch Gleichungen (7) und (8) beschrieben:

$$\mathbf{x}_{k+1} = \mathbf{A}(\mathbf{z}(k))\mathbf{x}(k) + \mathbf{b}(\mathbf{z}(k))u(k) + \xi(\mathbf{z}(k)) \tag{7}$$

$$y(k) = \mathbf{c}^{\mathrm{T}}(\mathbf{z}(k))\mathbf{x}(k). \tag{8}$$

Das Zustandsraummodell entsteht aus der Überlagerung von linearen Zustandsraummodellen. Somit ergibt sich beispielsweise die Systemmatrix

$$\mathbf{A}(\mathbf{z}(k)) = \sum_{i=1}^{M} \mathbf{A}_i \Phi_i(\mathbf{z}(k)) \tag{9}$$

aus der Überlagerung der einzelnen lokalen Systemmatrizen $\mathbf{A}_i$ mit der gleichen Gewichtung $\Phi_i$, wie beim initialen Modell. Auch die Vektoren $\mathbf{b}(\mathbf{z}(k))$, $\mathbf{c}^{\mathrm{T}}(\mathbf{z}(k))$ und $\xi(\mathbf{z}(k))$ werden analog dazu aus den Vektoren der Teilmodelle berechnet.

Die Transformation in die lokal-lineare Zustandsraumdarstellung ist auch auf MISO-Systeme (Multiple Input - Single Output) mit mehreren Eingangsgrößen anwendbar. Dabei können alle zuvor beschriebenen Ansätze zur Modellbildung direkt übertragen werden. Die MISO-Systeme werden separat für jede

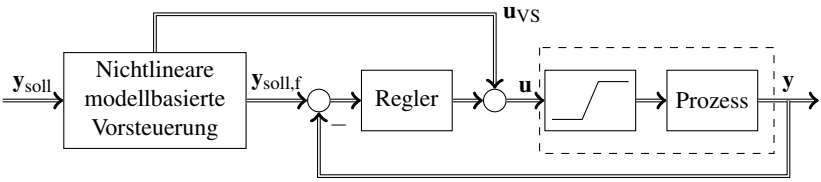



Bild 3: Regelungsstruktur mit dynamischer Vorsteuerung

Ausgangsgröße geschätzt und anschließend zu einem MIMO-System (Multiple Input - Multiple Output) zusammengefasst. In der Zustandsraumdarstellung werden dabei die Vektoren $\mathbf{b}(\mathbf{z}(k))$, $\mathbf{c}^{\mathrm{T}}(\mathbf{z}(k))$ und $\xi(\mathbf{z}(k))$ zu Matrizen erweitert. Durch die getroffenen Erweiterungen der Modellstruktur ergibt sich auch im MIMO-Fall eine minimale Zustandsraumrealisierung [10].

# 3 Reglerentwurf

In einem ersten Schritt wird ein Zustandsregler auf Basis des lokal-linearen Zustandsraummodells erstellt. Dieser Regler kann sowohl zur Regelung des realen Prozesses, als auch zur Regelung am Prozessmodell und somit als dynamische Vorsteuerung eingesetzt werden. Die gewählte Struktur für MIMO-Systeme ist in Bild 3 dargestellt. Die nichtlineare modellbasierte Vorsteuerung berechnet den Vorsteuerungsvektor $\mathbf{u}_{\mathrm{VS}}$ und den gefilterten Sollwertvektor $\mathbf{y}_{\mathrm{soll,f}}$. Diese Vorsteuerung kann hinsichtlich eines guten Führungsverhaltens ausgelegt werden und berücksichtigt zugleich die Stellgrößenbeschränkung. Der Prozess wird durch einen Mehrgrößenregler geregelt, der dazu dient Modellungenauigkeiten und Störungen zu kompensieren. Insgesamt ergibt sich eine Struktur mit zwei Freiheitsgraden (Two-Degrees-of-Freedom-Structure). Der Mehrgrößenregler kann somit in einfacher Weise umgesetzt werden und hinsichtlich des Störverhaltens optimiert werden. So ist zum Beispiel der Einsatz von dezentralen PID-Reglern oder die Erweiterung des entworfenen Zustandsreglers zu einem PI-Zustandsregler möglich und auch mit einem lokal-linearen Ansatz umsetzbar. Dabei kann durch die verwendete lokal-lineare Zustandsraumdarstellung auf eine große Anzahl an Anti-Windup Maßnahmen zurückgegriffen werden, so dass die Stellgrößenbeschränkung im Mehrgrößen-

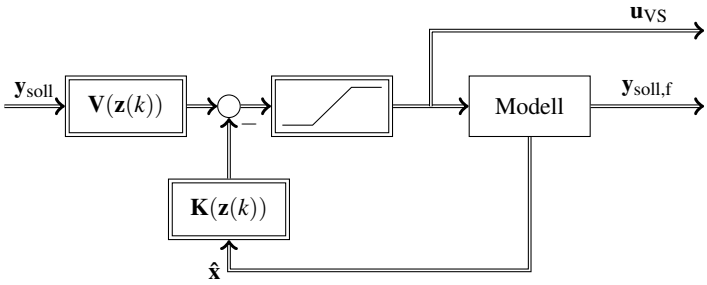



Bild 4: Struktur der Vorsteuerung

regler berücksichtigt werden kann [1]. Wir konzentrieren uns im Folgenden auf die Vorsteuerung, da diese in vielen Anwendungen bereits für ein gutes Führungsverhalten sorgt.

Anschließend wird als alternatives Regelungskonzept eine zustandsraumbasierte IMC-Regelung vorgestellt. Diese Regelung kann als eine Strukturerweiterung der Vorsteuerung gesehen werden und ist somit auch genauso einfach zu parametrieren. Sie wird auf das Führungsverhalten ausgelegt, wodurch sich auch das Störverhalten ergibt.

## 3.1 Vorsteuerung

Klassische Vorsteuerungen verwenden die inverse Übertragungsfunktion der Strecke um aus dem Sollwertverlauf den entsprechenden Stellgrößenverlauf zu berechnen. Diese inverse Übertragungsfunktion ist aber in vielen Fällen nicht direkt realisierbar, so dass eine Erweiterung um zusätzliche Polstellen notwendig ist [5]. Weist die erstellte Vorsteuerung eine zu hohe Dynamik auf, führt dies dazu, dass bei einem Sollwertwechsel die Stellgrößenbeschränkung verletzt wird. Dies führt zu einer deutlichen Verschlechterung des Führungsverhaltens und erhöht die Einschwingzeit.

Bild 4 zeigt die Struktur der im Rahmen dieser Arbeit betrachteten nichtlinearen modellbasierten Vorsteuerung, wie sie von [2, 9] beschrieben wurde.

Auf Grundlage des zuvor erstellten lokal-linearen Zustandsraummodells wurde ein Zustandsregler am Prozessmodell entworfen, dessen Zustandsrückführung

$\mathbf{K}(\mathbf{z}(k))$ und Vorfilter $\mathbf{V}(\mathbf{z}(k))$ ebenfalls aus lokal-linear überlagerten Matrizen bestehen. Analog zum lokal-linearen Zustandsraummodel ergibt sich damit die Zustandsrückführung

$$\mathbf{K}(\mathbf{z}(k)) = \sum_{i=1}^{M} \Phi_i(\mathbf{z}(k))\mathbf{K}_i. \tag{10}$$

Die einzelnen lokalen Zustandsrückführungen $\mathbf{K}_i$ können auf unterschiedliche Weisen bestimmt werden. So ist unter anderem ein Entwurf nach Polvorgabe möglich, oder es können optimale Entwurfsverfahren angewendet werden. Eines dieser optimalen Verfahren ist der Riccati-Entwurf, der im Rahmen dieser Arbeit verwendet wurde. Der Riccati-Regler kann auf einfache Weise über Gewichtungsmatrizen parametriert werden [6] und ermöglicht somit eine schnelle Auslegung der Vorsteuerung.

Neben der Zustandsrückführung wird ein Vorfilter $\mathbf{V}(\mathbf{z}(k))$ benötigt, der zur Sicherstellung der stationären Genauigkeit dient. Damit diese für die jeweiligen lokal-linearen Modelle gegeben ist, müssen neben den Matrizen des jeweiligen Zustandsraummodells ($\mathbf{A}_i$, $\mathbf{B}_i$, $\mathbf{C}_i$) und der Zustandsrückführung $\mathbf{K}_i$ auch der Offset $\xi$ und der aktuelle Sollwert $\mathbf{y}_{\text{soll}}$ berücksichtigt werden. Aufgrund dieser Einflussfaktoren ergibt sich ein besseres Verhalten, wenn der Vorfilter zu jedem Zeitschritt in Abhängigkeit des Arbeitspunkts $\mathbf{z}(k)$ und des Sollwerts neu berechnet wird [2]. Der Vorfilter

$$\begin{aligned}\mathbf{V}(\mathbf{z}(k)) = &\Big(\mathbf{C}(\mathbf{z}(k))\tilde{\mathbf{V}}(\mathbf{z}(k))\mathbf{B}(\mathbf{z}(k))\Big)^{-1} \\ &\cdot \Big(\mathbf{y}_{\text{soll}}(k) - \mathbf{C}(\mathbf{z}(k))\tilde{\mathbf{V}}(\mathbf{z}(k))\xi(\mathbf{z}(k))\Big)\end{aligned} \tag{11}$$

kann unter Verwendung der Hilfsmatrix

$$\tilde{\mathbf{V}}(\mathbf{z}(k)) = \Big(\mathbf{I} - \mathbf{A}(\mathbf{z}(k)) + \mathbf{B}(\mathbf{z}(k))\mathbf{K}(\mathbf{z}(k))\Big)^{-1} \tag{12}$$

online berechnet werden.

Die vorgestellte Struktur der Vorsteuerung weist den Vorteil auf, dass sie im Gegensatz zur klassischen inversionsbasierten Vorsteuerung, die Stellgrößenbeschränkung berücksichtigt. Auf diese Weise wird ein Stellgrößenverlauf generiert, der den Stellbereich möglichst gut ausnutzt.

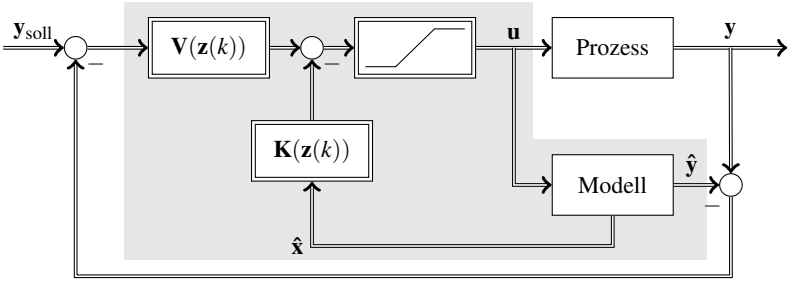



Bild 5: Struktur des zustandsbasierten IMC-Reglers

## 3.2 Zustandsbasierter IMC-Regler

Ein weiteres Regelungsverfahren, das auf Grundlage des erstellten Modells realisiert werden kann, ist die zustandsbasierte IMC-Regelung (Internal Model Control). Die Struktur dieses Reglungskonzepts ist in Bild 5 zu sehen. Es wurde von [7] für lineare Modelle beschrieben und von [9] für lokal-lineare Modelle erweitert. Wie bei der klassischen IMC-Reglung wird das Modell parallel zum Prozess betrieben und die Differenz zwischen Prozess- und Modellausgang zurückgeführt. Am Modell erfolgt hierbei jedoch eine Zustandsregelung unter Berücksichtigung der Stellgrößenbeschränkung, wie sie bereits in der Vorsteuerung zu finden ist (grau hinterlegt). Auf diese Weise ergibt sich ein Regler, der genauso wie die Vorsteuerung zu parametrieren ist und trotz Stellgrößenbeschränkung eine hohe Regelgüte aufweist. Ein weiterer Vorteil dieser Struktur ist, dass eine sprungförmige Störgröße am Ausgang keine bleibende Regelabweichung hervorruft. Die benötigte Rückführmatrix $\mathbf{K}(\mathbf{z}(k))$ und der Vorfilter $\mathbf{V}(\mathbf{z}(k))$ können mit denselben Methoden wie bei der Vorsteuerung entworfen werden.

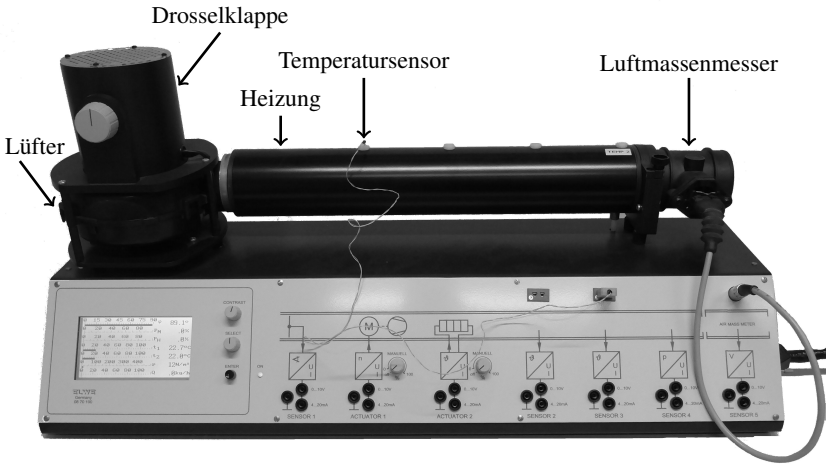



Bild 6: Labormodell der Heizstrecke

# 4 Anwendungsbeispiel – Heizstrecke

## 4.1 Systembeschreibung

Zur Entwicklung und Validierung des vorgestellten Verfahrens wurde eine Heizstrecke der Firma „ELWE Technik“ verwendet. Dabei handelt es sich um einen Labormodell, das über ein xPC Target-System unter Matlab-Simulink für Versuche verwendet werden kann. Verbaut sind neben einem Lüfter und einer Heizung, die jeweils mit variabler Leistung betrieben werden können, auch Sensoren, die zur Erfassung der Lufttemperatur und des Luftmassenstroms dienen. Außerdem ist eine manuell verstellbare Drosselklappe am Lufteinlass verbaut, um Störungen hervorzurufen. Das verwendete Labormodell ist in Bild 6 zu sehen.

Das System kann folglich als Mehrgrößensystem mit zwei Eingangs-, zwei Ausgangs- und einer Störgröße angesehen werden. Bei den Eingangsgrößen handelt es sich um die Lüfterleistung $P_\mathrm{L}$ und die Heizleistung $P_\mathrm{H}$. Als Ausgangsgrößen werden die Temperatur $T$ und der Luftmassenstrom $\dot{m}_\mathrm{L}$ erfasst. Schließlich kann noch der Öffnungswinkel der Drosselklappe $\alpha$ als Störgröße gemessen werden.

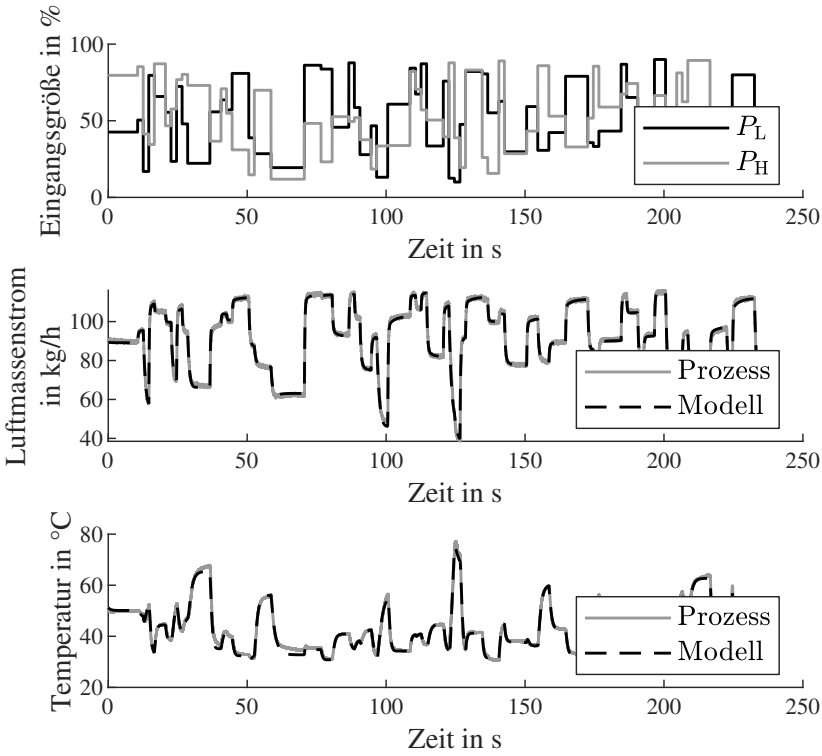
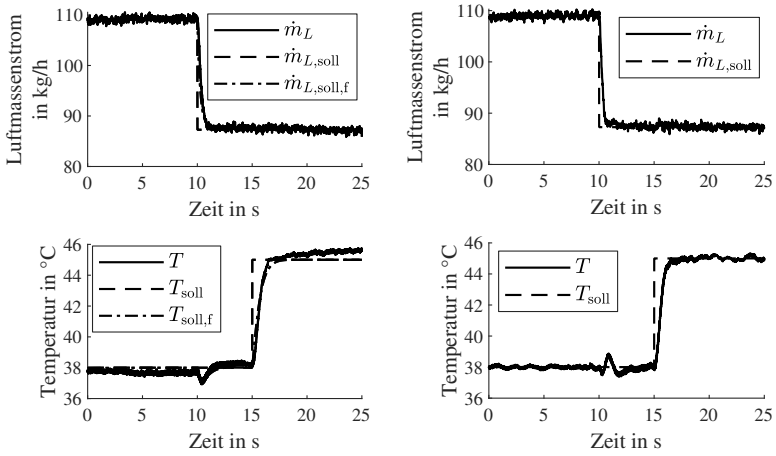



Bild 7: Ausschnitt aus einem Generalisierungsdatensatz mit den APRB-Signalen als Eingangsgrößen und Vergleich der Generalisierungsdaten mit den simulierten Modellausgängen

## 4.2 Modellbildung

Zur Anregung des Systems wird ein APRB (Amplituden-moduliertes Pseudo Rausch Binär)-Signale verwendet und auf die beiden Systemeingänge Lüfterleistung $P_L$ und Heizleistung $P_H$ aufgeschaltet. Um eine hohe Modellgüte zu erzielen, müssen dabei die entscheidenden Frequenz- und Amplitudenbereiche des Systems hinreichend angeregt werden. Hierzu wurden die APRB-Signale mit einer Länge von 1022 s und einer Taktzeit von 2 s gewählt. Die Amplituden sind gleichverteilt und stammen aus dem Intervall $[10, 90]\%$ der jeweils maximal möglichen Leistungen. Dieser Bereich wurde gewählt, das es ein üblicher




(a) Steuerung der Heizstrecke mit der nichtlinearen modellbasierten Vorsteuerung

(b) Regelung der Heizstrecke mit der zustandsraumbasierten IMC-Regelung

Bild 8: Vergleich der Vorsteuerung und IMC-Regelung bei Vorgabe von Sollwertsprüngen

Arbeitsbereich des Systems ist. Somit wird unter anderem vermieden, dass das System überhitzt, wie es beispielsweise bei einer hohe Heizleistung und einer niedrige Lüfterleistung der Fall wäre. Das erstellte lokal-lineare Modell besteht aus 7 Teilmodellen zweiter Ordnung.

Zur Validierung des Modells wurde ein Generalisierungsdatensatz verwendet. In Bild 7 ist der zeitliche Verlauf der zur Anregung verwendeten APRB-Signale für die beiden Eingangsgrößen $P_{\mathrm{L}}$ und $P_{\mathrm{H}}$ zu sehen. Außerdem sind für die Ausgangsgrößen Luftmassenstrom und Temperatur jeweils die gemessenen und geschätzten Werte aufgetragen. Die hohe Modellgüte wird anhand des Vergleichs der Generalisierungsdaten und der geschätzten Werten deutlich.

## 4.3 Vorsteuerung

Bild 8a zeigt den zeitlichen Verlauf des Luftmassenstroms und der Lufttemperatur an der Heizstrecke. Dabei wurde eine reine Steuerung der Strecke mit der vorgestellten nichtlinearen modellbasierten Vorsteuerung durchgeführt. Für beide Ausgangsgrößen wurde zeitversetzt ein Sprung der Sollwerte vorgegeben. Die daraus von der Vorsteuerung generierten Ausgangssignale, sowie die ermittelten Messwerte der am realen System gemessenen Größen sind ebenfalls dargestellt.

Bei dem betrachteten System ist der Luftmassenstrom weitgehend unabhängig von der Lufttemperatur und damit auch von der Heizleistung. Eine Beeinflussung durch die temperaturabhängige Dichteänderung der Luft ist im Rahmen der Messgenauigkeit nicht erfassbar. Die Luftmasse kann dementsprechend als ausschließlich von der Lüfterleistung abhängige Eingangsgröße angesehen werden. Die Temperatur wird hingegen sowohl von der Lüfterleistung als auch von der Heizleistung beeinflusst. Die beiden Eingangsgrößen sind bezüglich dieser Ausgangsgröße stark verkoppelt. In Bild 8a erfolgt zum Zeitpunkt $t_1 = 10$ s ein Sollwertsprung der Luftmasse. Die Vorsteuerung erzeugt einen entsprechenden Stellgrößenverlauf um den Luftmassenstrom auf den gewünschten Wert zu steuern. Gleichzeitig ist die Vorsteuerung in der Lage, trotz der Verkopplung der beiden Eingangsgrößen, auch die Temperatur so zu steuern, dass diese nur kurzzeitig um ca. 2 % des bei der Generalisierung (Bild 7) abgedeckten Temperaturbereichs von ihrem Sollwert abweicht.

Insgesamt zeigt sich, dass das betrachtete System mit der vorgestellten Vorsteuerung in verschiedenen Arbeitspunkten gesteuert werden kann. Damit werden die Anforderungen an das Führungsverhalten eines Reglers, der die verbleibenden Abweichungen ausregelt, deutlich herabgesetzt. Somit kann dieser Regler hinsichtlich seines Störverhaltens optimiert werden.

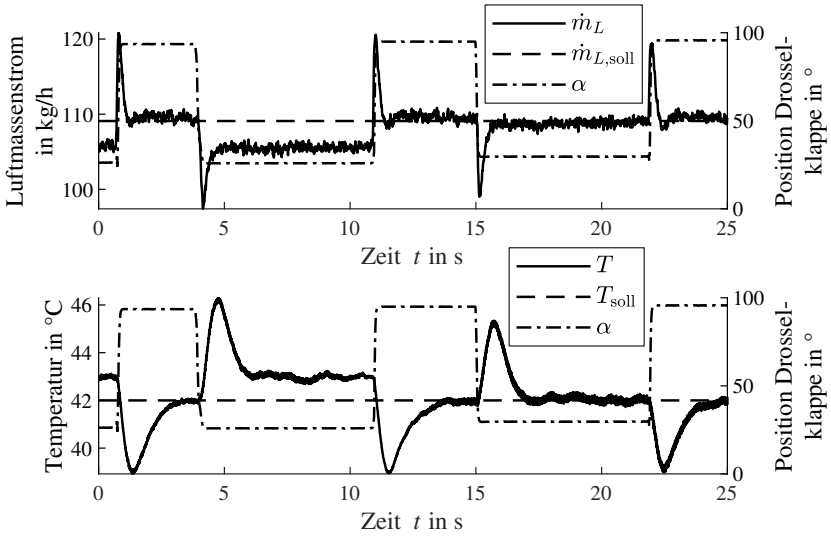



Bild 9: Anwendung der zustandsraumbasierten IMC-Regelung bei konstantem Sollwert und sprungförmiger Störgröße

## 4.4 Internal Model Control

Der entworfene zustandsbasierte IMC-Regler wurde an der Heizstrecke validiert. In Bild 8b ist das Verhalten dieser Regelung, der zuvor betrachteten Vorsteuerung gegenübergestellt. In beiden Fällen wurden dieselben Sollwertsprünge vorgegeben. Es zeigt sich ein ähnliches Führungsverhalten, mit dem Unterschied, dass durch die Regelung stationäre Genauigkeit gegeben ist.

Außerdem ist die Regelung in der Lage auf äußere Störungen zu reagieren. Eine solche Störung wurde durch Veränderung der Position der Drosselklappe $\alpha$ hervorgerufen. Dazu wurde die Drosselklappe mehrfach annähernd sprungförmig von der vollständig geöffneten Position $\alpha \approx 90^\circ$ in Positionen gebracht, in denen der Lufteinlass verengt ist. Bild 9 zeigt die Drosselklappenposition zusammen mit den gemessenen Regelgrößen bei konstanten Sollwerten. Während die zweite Störung im Zeitraum $(15\ \text{s} < t < 22\ \text{s})$ vollständig ausgeregelt wurde, bleibt bei der ersten Störung im Zeitraum $(4\ \text{s} < t < 12\ \text{s})$ eine Regelabweichung bestehen. Da das Schließverhalten der Drosselklappe stark

nichtlinear ist, wurde durch den geringfügig kleineren Winkel der Lufteinlass deutlich stärker verengt, als bei der zweiten Störung. Aufgrund der gegebenen Stellgrößenbegrenzung ist es somit nicht mehr möglich den Sollwert zu erreichen. Eine wichtige Eigenschaft eines Reglers ist, dass eine solche bleibende Regelabweichung aufgrund einer Stellgrößenbeschränkung nicht kontinuierlich aufintegriert wird (Wind-Up-Effekt), was nach abklingen der Störung zu einer verzögerten Reaktion des Reglers führt. Vorteilhaft an dem zustandsbasierten IMC-Regler ist, dass dieser eine solche Anti-Wind-Up Maßnahme implizit berücksichtigt [7]. Dies wird auch anhand der durchgeführten Messung deutlich, da das Überschwingen nach Aufhebung der Störung in beiden Fällen annähernd gleich ist.

## 5 Zusammenfassung und Ausblick

Das in diesem Artikel vorgestellte Verfahren ermöglicht einen systematischen Entwurf von Mehrgrößenreglern auf Basis von lokal-linearen Modellen. Die beschriebenen Erweiterungen der Modellstruktur, sowie die Überführung in eine lokal-lineare Zustandsraumdarstellung resultieren in ein qualitativ hochwertiges Modell, auf dessen Grundlage zahlreiche Entwurfsverfahren für Regler und Vorsteuerungen auf einfache Weise angewendet werden können. Eines dieser Verfahren ist die modellbasierte dynamische Vorsteuerung. Die Eignung dieser Struktur in Kombination mit einem Mehrgrößenregler als Vorsteuerung zu arbeiten, sowie die Erweiterung zu einem zustandsraumbasierten IMC-Regler, wurden aufgezeigt.

Als weiterer Schritt kann eine detaillierte Stabilitätsbetrachtung durchgeführt werden. So kann das erstellte lokal-lineare Zustandsraummodell durch die Überlagerung ein unerwünschtes Verhalten aufweisen, das die Stabilität beeinträchtigten kann [10]. Ein weiterer Aspekt, der die Stabilität der entworfenen Fuzzy-Zustandsregler betrifft, ist die Rückkopplung der Stellgröße zur Vorgabe des Arbeitspunkts. Da sich in Abhängigkeit vom Arbeitspunkt auch die Reglerparameter ändern und diese wiederum zurück auf die Stellgröße wirken, kann dies auch zu Instabilitäten führen. Denkbar wäre hierfür der Einsatz von modellbasierten Ansätzen zur Arbeitspunktbestimmung.

## Literatur


[1] P. Hippe, „Windup in Control: Its Effects and Their Prevention (Advances in Industrial Control). Springer-Verlag, London. 2006.

[2] M. Kohlhase, „Brennraumdruckbasiertes Motormanagement für Otto- und Dieselmotoren zur Verbrauchs und Emissionsreduktion“. In: *Fortschritt-Berichte VDI : Reihe 12: Verkehrstechnik / Fahrzeugtechnik*, VDI-Verlag, Düsseldorf. 2011.

[3] A. Kroll, T. Bernd, S. Trott, „Fuzzy Network Model-Based Fuzzy State Controller Design“. In: *IEEE Transactions on Fuzzy Systems, 8(5):632–644.* 2000.

[4] L. Ljung, „System Identification: Theory for the User“. Prentice Hall,Englewood Cliffs. 1999.

[5] J. Lunze, „Regelungstechnik 1: Systemtheoretische Grundlagen, Analyse und Entwurf einschleifiger Regelungen“. Springer Vieweg Verlag. 2014.

[6] J. Lunze, „Regelungstechnik 2: Mehrgrößensysteme, Digitale Regelung“. Springer Vieweg Verlag. 2014.

[7] S. Mhatre., C. Brosilow, „Multivariable Model State Feedback: Computationally Simple, Easy-to-Tune Alternative to MPC“. In: *AIChE Journal, Vol. 46(8):1566–1580*. 2000.

[8] O. Nelles, „Nonlinear System Identification with Local Linear Neuro-Fuzzy Models“. Shaker Verlag. 1999

[9] K. von Pfeil, „Ladedruck- und Luftmassenregelung von aufgeladenen Dieselmotoren mit lokal linearen Modellen und Optimierung des dynamischen Emissionsverhaltens im Rauchbetrieb“. In: *Fortschrittberichte VDI : Reihe 12, Verkehrstechnik, Fahrzeugtechnik*; VDI-Verlag, Düsseldorf. 2011.

[10] R. Zimmerschied, „Identifikation nichtlinearer Prozesse mit dynamischen lokal-affinen Modellen: Maßnahmen zur Reduktion von Bias und Varianz“. In: *Fortschrittbericht VDI Nr. 1150, Reihe 8.*, VDI-Verlag, Düsseldorf. 2009.

# Gray-Box Regularized FIR Modeling for Linear System Identification

Timm J. Peter, Oliver Nelles

Universität Siegen, Department Maschinebau, Institut für Mechanik und Regelungstechnik - Mechatronik
Paul-Bonatz-Str. 9-11, 57068 Siegen, Germany
E-Mail: {timm.peter,oliver.nelles}@uni-siegen.de

## 1 Introduction

For modeling of linear dynamic systems two different approaches can be distinguished. On the one hand for white-box modeling, first principles are used to describe a system. This approach is not capable to cover undermodeling, as non-modeled aspects of the process cannot be captured. On the other hand, black-box modeling only uses the available data to model a system. Popular structures for black-box modeling are ARX, OE, and FIR. FIR models are the only choice which allows to consistently estimate parameters in the output error case by solving a least-squares problem. The model order has to be set very high for usual systems to cover the significant parts of the impulse response, and thus the variance error of the parameters is extremely high for small amounts of data.

Starting with a new kernel-based method to regularize the impulse response of a system [2], the field of linear dynamic system identification gained new interest. This was achieved by formulating the identification problem with a Bayesian approach and interpreting the impulse response as a Gaussian process [5]. By doing so, the parameters of the covariance matrix are used to integrate prior knowledge about the described system. For smoothness and exponential decay, a stable spline kernel can be chosen [2]. Since the parameters of a FIR model are equal to the impulse response coefficients, this method can be interpreted as regularized FIR identification. A modification of the appro-

ach reformulates the regularization term with a filter matrix [3]. With the filter matrix formulation, a simpler interpretation of the underlying penalty term is possible. As an extension of the filter matrix approach an impulse response preserving (IRP) matrix can be formulated. This matrix has the property that a given impulse response is not penalized by the regularization. This allows to use tailored filter matrices for known systems and to easily integrate almost arbitrary system properties as prior knowledge [1, 9].

The introduced approach combines both black-box and gray-box or white-box approaches for linear systems. To achieve this, the loss function for FIR systems gets extended by a regularization term to penalize deviation from a white-box model. In addition, physical parameters of the observed system and the regularization strength of the regularization term can be treated as hyper-parameters.

## 2 Regularized FIR identification

Using FIR models for linear system identification has several benefits in opposite to the widely used ARX approach. The FIR model does not feed back measured output data, thus as aforementioned consistent parameters are obtained for a realistic noise model. However, a crutial drawback of FIR models is the need for a big number of parameters to cover oscillating behavior of the impulse response. A regularization approach solves this problem.

### 2.1 FIR modeling

The parameters of a FIR model are obtained by solving

$$\min_{\underline{\theta}} \|\underline{y} - \underline{X}\,\underline{\theta}\|_2^2 \tag{1}$$

with the regression matrix

$$\underline{X} = \begin{pmatrix} u(n) & \cdots & u(1) \\ u(n+1) & \cdots & u(2) \\ \vdots & \ddots & \vdots \\ u(N-1) & \cdots & u(N-n) \end{pmatrix}, \tag{2}$$

the output vector $\underline{y}$

$$\underline{y} = \begin{pmatrix} y(n+1) & y(n+2) & \cdots & y(N) \end{pmatrix}^T, \tag{3}$$

and the parameter vector $\underline{\theta}$

$$\underline{\theta} = \begin{pmatrix} g_1 & g_2 & \cdots & g_n \end{pmatrix}^T. \tag{4}$$

The solution of the minimization problem is the well known least squares estimate

$$\underline{\hat{\theta}} = (\underline{X}^T \underline{X})^{-1} \underline{X}^T \underline{y}. \tag{5}$$

## 2.2 Regularization approach

For small data sets big numbers of parameters for FIR identification cause a high variance error. To circumvent this problem, a regularization term can be added to the cost function of the FIR model, which leads to

$$I(\underline{\theta}) = \|\underline{y} - \underline{X}\,\underline{\theta}\|_2^2 + \lambda\, \underline{\theta}^T \underline{R}\, \underline{\theta}. \tag{6}$$

Here $\underline{R}$ denotes a symmetric, positive semidefinite matrix and $\underline{\lambda}$ a scalar to set the strength of the regularization term. The term for setting up $\underline{\theta}$ extends to

$$\underline{\theta} = (\underline{X}^T \underline{X} + \lambda \underline{R})^{-1} \underline{X}^T \underline{y}. \tag{7}$$

Using a Bayesian interpretation of the regularization problem, the regularization matrix $\underline{R}$ can be defined as

$$\underline{R} = \underline{P}^{-1} \tag{8}$$

with the covariance matrix $\underline{P}$. This changes the estimate of the parameters to

$$\underline{\theta} = (\underline{P}\underline{X}^T\underline{X} + \lambda\underline{I})^{-1}\underline{P}\underline{X}^T\underline{y}. \tag{9}$$

This formulation allows to integrate prior knowledge about the system by the setup of $\underline{P}$. A common assumption for impulse responses is a smooth, exponential decay to zero. By using the stable spline kernel [2]

$$P_{ij} = \alpha^{\max(i,j)} \tag{10}$$

these attributes can be achieved. Another common name for this kernel is tuned-correlated (TC) [8].

## 2.3 Filter interpretation of regularization problem

In contrast to the Bayesian viewpoint for the regularization problem the filter interpretation can be used [3]. To obtain this formulation, the penalty matrix $\underline{R}$ is restated as $\underline{R} = \underline{F}^T\underline{F}$. By doing so, the cost function 6 can be rewritten as

$$I(\underline{\theta}) = ||\underline{Y} - \underline{X}\,\underline{\theta}||_2^2 + \lambda||\underline{F}\,\underline{\theta}||_2^2. \tag{11}$$

In this equation, the effect of the filter matrix can be interpreted as a filter operation on the coefficients of the impulse response. The rows of $\underline{F}$ act as a prefiltering of the coefficients in $\underline{\theta}$. Therefore, the matrix $\underline{F}$ should be defined to penalize unwanted behavior of the impulse response.

The estimation of the parameters expands to

$$\hat{\underline{\theta}} = (\underline{X}^T\underline{X} + \lambda\underline{F}^T\underline{F})^{-1}\underline{X}^T\underline{y}. \tag{12}$$

Additionally, the filter matrix can be weighted. This can be achieved by multiplying $\underline{F}$ with weighting matrix

$$\underline{W} = \mathrm{diag}\left(\alpha^0, \alpha^{-1}, ..., \alpha^{-n}\right). \tag{13}$$

For $\alpha < 1$ this weighting causes an exponentially stronger penalty for the latter values of the impulse response.

## 2.4 Optimization of the hyperparameters

In order to tune the hyperparameters of a regularized FIR approach, different methods can be used. Here, the two most common approaches are introduced. The first method maximizes the marginal likelihood

$$L_{ML}(\underline{\eta}) = \underline{y}^T \underline{Z}(\underline{\eta})^{-1} \underline{y} + \log \det \underline{Z}(\eta) \tag{14}$$

with covariance matrix $\underline{Z}(\underline{\eta})$

$$\underline{Z}(\underline{\eta}) = \underline{X}\,\underline{P}(\underline{\eta})\,\underline{X}^T + \sigma^2 \underline{I}_N \tag{15}$$

to tune the hyperparameters gathered in the vector $\underline{\eta}$ consisting of $\alpha$ and all parameters within $\underline{F}$. The second approach is to use the generalized cross-validation (GCV) error

$$L_{GCV}(\underline{\eta}) = \frac{1}{N} \frac{\sum_{i=1}^{N} (y(i) - \hat{y}(i))^2}{(1 - Tr(\underline{S}(\underline{\eta}))/N)^2} \tag{16}$$

with smoothing matrix

$$\underline{S}(\underline{\eta}) = \underline{X}(\underline{X}^T \underline{X} + \lambda \underline{F}^T(\underline{\eta}) \underline{F}(\underline{\eta}))^{-1} \underline{X}^T. \tag{17}$$

A practical advantage of the GCV over the marginal likelihood error is that the variance $\sigma^2$ does not need to be estimated to tune the hyperparameters. See [7] for details about appropriate implementation of tuning algorithms and [4] on rewriting the marginal likelihood maximization for filter matrices.

## 2.5 Impulse response preserving matrices

As mentioned before, the filter matrix should be set up in order to penalize unwanted behavior for the impulse response. At the same time, desired behavior should be maintained. In [1] a method to design filter matrices $\underline{F}$ without penalizing the *true* impulse responses is shown. Filter matrices with this property are called impulse response preserving (IRP), if $\lim_{n \to \infty} \|\underline{F}_n \underline{\theta}_n\|_2 = 0$ holds. The subscript $n$ is used to indicate the order of the FIR model, while $m$ is used to

mark the order of the IRP matrx. In order to underline the meaning of this term, a simple first order process

$$G(z) = \frac{1-\alpha}{z-\alpha} = \frac{Y(z)}{U(z)}. \tag{18}$$

is considered. Transposed and transformed to discrete time, it can be written as

$$y(k) = u(k-1) - u(k-1)\alpha + y(k-1)\alpha. \tag{19}$$

To calculate the impulse response, the input has to be

$$u(k) = \begin{cases} 1 & \text{for } k = 0 \\ 0 & \text{otherwise} \end{cases}. \tag{20}$$

The impulse response $y(k)$ can now be calculated for every time step $k$ according to

$$\begin{aligned}
k =& 0 : y(0) = 0 \\
k =& 1 : y(1) = 1 - \alpha = \theta_1 \\
k =& 2 : y(2) = \alpha - \alpha^2 = \theta_2 = \theta_1 \alpha \\
k =& 3 : y(3) = \alpha^2 - \alpha^3 = \theta_3 = \theta_2 \alpha \\
& \vdots \\
k =& n : y(n) = \alpha^{n-1} - \alpha^n = \theta_n = \theta_{n-1} \alpha
\end{aligned}$$

with FIR order $n$. It can be observed, that impulse response coefficients $\theta_i$ are calculated by the multiplication of previous coefficients and denominator coefficients of the transfer function. Based on this observation, the impulse response preserving equations

$$\underline{F}\,\underline{\theta} = \begin{pmatrix} -\alpha & 1 & 0 & \dots & 0 \\ 0 & -\alpha & 1 & \dots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -\alpha & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{pmatrix} = \begin{pmatrix} -\alpha\theta_1 + \theta_2 \\ -\alpha\theta_2 + \theta_3 \\ \vdots \\ -\alpha\theta_{n-1} + \theta_n \end{pmatrix} \tag{21}$$

with filter matrix $\underline{F}$ as the IRP matrix are set up. If these expressions equal zero, the impulse response is not penalized and called preserved. Setting the IRP expressions (21) equal to zero yield

$$\begin{aligned}
&(1): -(\alpha-\alpha^2)+(\alpha-\alpha^2)=0\\
&(2): -(\alpha^2-\alpha^3)+(\alpha^2-\alpha^3)=0\\
&\quad\vdots\\
&(n): -(\alpha^{n-1}-\alpha^n)+(\alpha^{n-1}-\alpha^n)=0
\end{aligned}$$

for row 1 to $n$. Obviously, this requires the correct choice for the order n of the FIR model. Figure 1 shows exemplary calculations for a first and second order process. The IRP matrix for an arbitrary order $m$ is defined as

$$\underline{F}=\begin{pmatrix}
a_0 & a_1 & \dots & a_{m-1} & 1 & 0 & \dots & 0\\
0 & a_0 & a_1 & \dots & a_{m-1} & 1 & \dots & 0\\
\vdots & & \ddots & \ddots & & \ddots & \ddots & \vdots\\
0 & \dots & 0 & a_0 & a_1 & \dots & a_{m-1} & 1
\end{pmatrix}. \tag{22}$$

Here, $a_0$ to $a_{m-1}$ equal the coefficients of the denominator polynomial of a $m^{th}$-order transfer function. In order to train a model, the coefficients $a_0$ to $a_{m-1}$ can be treated as additional hyperparameters.

## 3 Gray-box regularized FIR models

The gray-box regularized FIR method uses a filter matrix structure and thus (11) as the cost function and (12) for parameter estimation. Matrix $\underline{F}$ is set up with an IRP structure to integrate prior knowledge. In contrast to the standard IRP approach, the parameters of the filter matrix are fixed. The parameters are predefined by a white-box modeling of the observed process. In consequence, the flexibility of the model and also the variance error decrease.

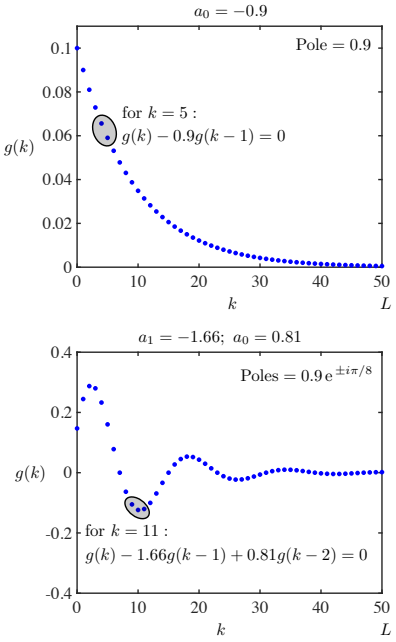



Figure 1: Examples for impulse responses of a first (top) and second (bottom) order process with prior knowledge preserving the shape of the impulse response via incorporation in the FIR model by regularization.

## 3.1 White-box modeling

To model a process, it has to be described by one or more differential equations. Considering linear system identification, the differential equations need to be linear or linearized at the current operating point. For more complex processes, a state space form is beneficial to obtain the transfer function of the process. As a first step, the continuous state-space form is set up. Secondly, the continuous form is transformed to a discrete state-space form

$$\underline{x}(k+1) = \underline{A}(\underline{\theta})\underline{x}(k) + \underline{b}(\underline{\theta})u(k) \tag{23}$$

$$\underline{y}(k) = \underline{c}(\underline{\theta})^{\mathrm{T}}\underline{x}(k) + d(\underline{\theta})u(k) \tag{24}$$

with system matrix $\underline{A}(\underline{\theta})$, input vector $\underline{b}(\underline{\theta})$, output vector $\underline{c}^T(\underline{\theta})$ and feed-through $d(\underline{\theta})$. This can be achieved e.g. in Matlab using the function *c2d*.

Now the transfer function can be written as

$$G(z) = \underline{C}(z\underline{I} - \underline{A})^{-1}\underline{B} + \underline{D} \tag{25}$$

$$= \frac{1}{\det(z\underline{I} - \underline{A})}(\underline{C}\mathrm{Adj}(z\underline{I} - \underline{A})\underline{B} + \underline{D}\det(z\underline{I} - \underline{A})) \tag{26}$$

see [6], with Adj($\underline{A}$)is the adjunct matrix to $\underline{A}$. $G(z)$ in (25) is a $m^{th}$-order transfer function with input $U(z)$ output $Y(z)$ of form

$$G(z) = \frac{b_m z^m + b_{m-1}z^{m-1} + \cdots + b_1 s + b_0}{z^n + a_{m-1}z^{m-1} + \cdots + a_1 z + a_0}. \tag{27}$$

Based on (25), a filter matrix can be set up as described in (22) by using the denominator $\det(z\underline{I} - \underline{A}(\theta))$.

### 3.2 Properties of GBRFIR models

The key quality of the GBRFIR approach is represented by the automatic model trade-off. This trade-off occurs through the hyperparameter tuning, which is done by minimizing the GCV error. By setting $\lambda = 0$, (11) reduces to the standard least squares cost function, thus the model is only dependend on measured data. If $\lambda \to \infty$, deviation from prior knowledge in $\underline{F}$ is penalized infinitely strong. In this case, the impulse response is equivalent to the white-box model. Generally, optimizing the GCV error leads to an impulse response between the two extreme examples.
Another feature of the GBRFIR approach is the possibility to implement physical parameters as additional hyperparameters. This can be useful due to the possibility of inaccurately calculated or estimated physical system properties.

## 4 Real system example

In this section, a real pendulum system, see Fig. 2, is introduced and used to demonstrate the appliciability of the GBRFIR approach. In addition, the system is used to gather data to test the approach. To be able to assess the performance

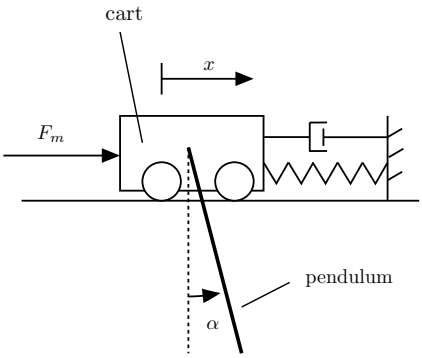



Figure 2: Pendulum system

of the results, the approach is compared to other linear system identification approaches. The observed system is a motor-driven cart, which is spring connected to the test rig. A pivot-mounted pendulum is attached to the side of the cart. Due to the toothed rack lead the cart can move in $x$-direction. To identify the system, input and output have to be defined. The input variable is the voltage $U_m$ applied to the motor. The applied voltage causes a force $F_m$ in $x$-direction to the cart. As output variable, the pendulum angle $\alpha$ is chosen.
In order to excite the dynamic system, a pseudorandom binary signal (PRBS) is used, the sampling rate is set to $T_s = 0.02$. The obtained data set consists of 20000 points. A FIR order of $n = 500$ is chosen. Hyperparameters are tuned by minimizing the GCV error.

## 4.1 White-box modeling

To obtain the differential equations of the process, the equations for movement are set up for cart and pendulum. Therefore, the equilibrium of forces and moments are calculated in $x$ and $y$ direction. The process can be described by a $4^{th}$ order model. These equations have to be rearranged, solved for $\ddot{x}$ and $\ddot{\alpha}$, and linearized. Subsequently, coming from the differential equations the state space model

$$\frac{d}{dt}\begin{pmatrix}\alpha\\ \dot{\alpha}\\ x\\ \dot{x}\end{pmatrix} = \begin{pmatrix}0 & 1 & 0 & 0\\ H_1 & H_2 & H_3 & H_4\\ 0 & 0 & 0 & 1\\ H_5 & H_6 & H_7 & H_8\end{pmatrix}\begin{pmatrix}\alpha\\ \dot{\alpha}\\ x\\ \dot{x}\end{pmatrix} + \begin{pmatrix}0\\ H_5\\ 0\\ H_{10}\end{pmatrix} u$$
$$y = \begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix}\begin{pmatrix}\alpha & \dot{\alpha} & x & \dot{x}\end{pmatrix}^T \tag{28}$$

is built. The values for $H_1, ..., H_{10}$ are listed in Tab. 1 and depend on the physical parameters of cart, pendulum and spring. To discretize the model, the Matlab `c2d` function with a sampling time $T_s = 0.02$ s is used. The denominator of transfer function

$$G(z) = \frac{B(z)}{1 \cdot z^4 - 3.73 \cdot z^3 + 5.27 \cdot z^2 - 3.35 \cdot z + 0.80} \tag{29}$$

is used to set up the filter matrix

$$\underline{F} = \begin{pmatrix}0.8 & -3.35 & 5.27 & -3.73 & 1 & 0 & \dots & 0\\ 0 & 0.8 & -3.35 & 5.27 & -3.73 & 1 & 0 & \vdots\\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots\\ 0 & \dots & \dots & 0.8 & -3.35 & 5.27 & -3.73 & 1\end{pmatrix}. \tag{30}$$

Additionally, the filter matrix is multiplied by an exponential weighting matrix.

## 5 Performance on data set

In order to show the potential of the GBRFIR approach, it is tested on the pendulum system data set. For the purpose of achieving comparable results, the performance of alternative approaches is also evaluated on the pendulum data set. The chosen models are a unregularized FIR model, a TC kernel regularized FIR model and regularized FIR models with $2^{nd}$ and $4^{th}$ order IRP matrices. Moreover, a GBRFIR approach with physical parameters as additional hyperparameters is tested.

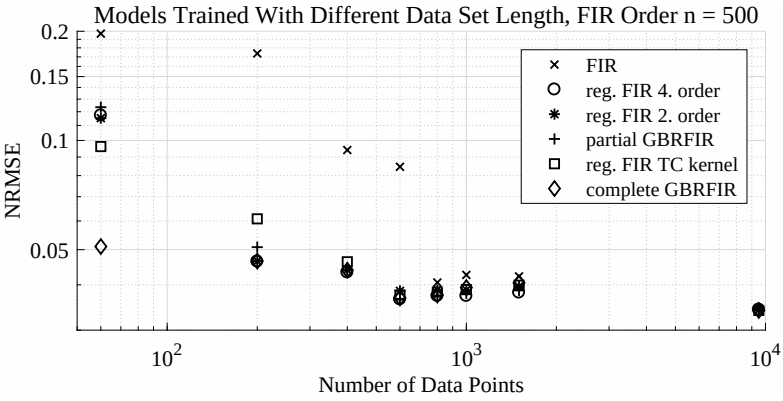



Figure 3: Effect of the amount of data used for training

## 5.1 Influence of data set length

First, the effect of training sets with different amounts of data points is investigated. Figure 3 shows the NRMSE on test data over the number of data points used for training. It can be observed, that with smaller amounts of data used the NRMSE rises due to the lower information content of the data. For 60 data points, the GBRFIR model is solely capable to reproduce the system behavior. The other tested models show a significantly higher NRMSE. This might be reasonably traced back to the implemented prior knowledge in $\underline{F}$ and the low number of hyperparameters for the GBRFIR approach. Going to higher numbers between 200 and 1000 data points, models with higher numbers of parameters and thus higher flexibility perform better. The regularized FIR with $4^{th}$-order IRP matrix and the optimized GBRFIR perform best. For big training data sets, the difference between the models becomes very small. For small to medium sized training data sets, a regularization approach is beneficial in all cases.

## 5.2 Influence of output noise

Secondly, the effects of noise-corrupted training data are investigated while keeping the length of the data set constant at 5000 training samples. Figure 4

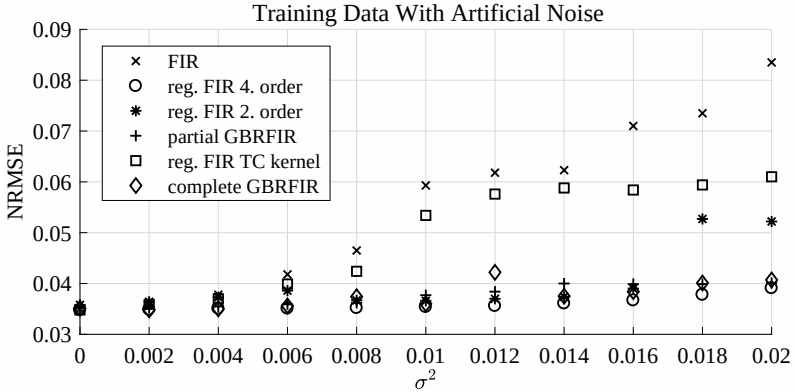



Figure 4: Effect of artificial Gaussian noise $\sigma^2$

shows the NRMSE over the variance of artificial gaussian noise added to the training data set. With rising variance $\sigma^2$, the difference between the models regarding the NRMSE also rises. With a low $\sigma^2$ of 0 to 0.006 there is almost no noticable difference. From $\sigma^2 = 0.008$ on, the NRMSE of the unregularized FIR and the regularized FIR with TC matrix are rising. In contrast, the NRMSE of the regularized approaches with IRP matrices and the GBRFIR models increases negligibly. Based on this observations, it can be concluded that GBRFIR and IRP models are able to automatically trade-off the trust in the data to trust in the white-box model.

# 6 Conclusion

In this contribution, techniques for regularized identification have been employed for gray-box identification. A pendulum system was used to demonstrate the GBRFIR filter matrix in practice. By the use of profound prior knowledge concerning the observed process, the performances of a regularized FIR approach with a TC-kernel matrix and the standard FIR approach were surpassed on test data sets. Two different test scenarios were analyzed. At first, the effect of training data set length was investigated. The GBRFIR model shows advantages for small training data sets compared to the regularized approaches with TC-kernel matrix, IRP matrix and the unregularized FIR. For

Table 1: Formulas for the computation of the white-box-model coefficients.

| Constant | Formula |
|---|---|
| Denominator $D_1$ | $I_p + M_p l_p^2 - \frac{M_p^2 l_p^2}{M_c + M_p}$ |
| Denominator $D_2$ | $M_p + M_c + \frac{M_p^2 l_p^2}{I_p + M_p l_p^2}$ |
| Engine Coefficient $v_1$ | $\frac{-\eta_g K g^2 \eta_M K_t K_m}{R_M r_{mp}^2}$ |
| Engine Coefficient $v_2$ | $\frac{\eta_g K_g \eta_M K_t}{R_M r_{mp}}$ |
| $H_1$ | $-\frac{1}{D_1} l_p M_p g$ |
| $H_2$ | $-\frac{1}{D_1} B_p$ |
| $H_3$ | $\frac{1}{D_1} \frac{-l_p M_p}{M_c + M_p}$ |
| $H_4$ | $\frac{1}{D_1} \frac{-l_p M_p}{M_c + M_p} (v_1 - B_{eq})$ |
| $H_5$ | $\frac{1}{D_1} \frac{-l_p M_p}{M_c + M_p} v_2$ |
| $H_6$ | $\frac{1}{D_1} \frac{M_p l_p}{I_p + M_p l_p^2} - M_p l_p g$ |
| $H_7$ | $\frac{1}{D_1} \frac{M_p l_p}{I_p + M_p l_p^2} B_p$ |
| $H_8$ | $-\frac{1}{D_1} K_s$ |
| $H_9$ | $\frac{1}{D_1} (v_1 - B_{eq})$ |
| $H_{10}$ | $\frac{1}{D_1} v_2$ |

the second scenario, the training data set was corrupted by artificial Gaussian noise. For high variances of the artificial noise, the GBRFIR and the IRP approach showed advantages over the regularized TC-kernel matrix approach and the unregularized FIR approach. Overall, the benefits of regularized FIR estimation in comparison to unregularized FIR modeling for small to medium data set length as well as for noisy data sets were shown.

# 7 Parameters of the Physical Model

The physical parameters in Tab. 2 are used to build the model. To maintain a compact form, the coefficients in Tab. 1 are used.

Table 2: Physical constants for cart and pendulum

| Constant | Value |
|---|---|
| Spring Rate | $Ks = 200\,\frac{N}{m}$ |
| Motor Armature Resistance | $R_M = 2.6\,\Omega$ |
| Motor Armature inductance | $L_m = 180 \times 10^{-6}\,H$ |
| Motor Torque Constant | $K_t = 0.0077\,\frac{N \cdot m}{A}$ |
| Motor Efficiency | $\eta_M = 1$ |
| Back-Electromotive-Force Constant | $K_m = 0.0077\,\frac{V \cdot s}{rad}$ |
| Gear Ratio | $K_g = 3.71$ |
| Planetary Gearbox Efficiency | $\eta_g = 1$ |
| Rotor Moment of Inertia | $J_m = 3.9001 \times 10^{-7}\,kg \cdot m^2$ |
| Cart Mass | $M_{c2} = 0.57\,kg$ |
| Cart Weigth Mass | $M_w = 0.37\,kg$ |
| Motor Pinion Radius | $r_{mp} = 0.0063\,m$ |
| Motor Pinion Number of Teeth | $N_{mp} = 24$ |
| Position Pinion Number of Teeth | $N_{pp} = 56$ |
| Position Pinion Radius | $r_{pp} = 0.0148\,m$ |
| Rack Pitch | $P_r = 0.0017\,\frac{m}{tooth}$ |
| Cart Travel | $T_c = 0.814\,m$ |
| Cart Encoder Resolution | $K_{EC} = 2.2749 \times 10{-}5$ |
| Pendulum Encoder Resolution | $K_{EP} = 0.0015$ |
| Damping Coefficient Engine | $B_{eq} = 5.4\,\frac{N \cdot s}{m}$ |
| Acceleration of Gravity | $g = 9.81\,\frac{m}{s^2}$ |
| Cart Mass Total | $M_c = 1.0731\,kg$ |
| Pendulum Mass | $M_p = 0.18\,kg$ |
| Pendulum Full Length | $L_p = 0.3365\,m$ |
| Pendulum Distance Center of Gravity | $l_p = 0.1778\,m$ |
| Pendulum Moment of Inertia | $I_p = 1.1987 \times 10{-}3\,kg \cdot m^2$ |
| Damping Coefficient Pendulum | $B_p = 0.01\,\frac{N \cdot m \cdot s}{rad}$ |

## References


[1] T. Münker, J. Belz, and O. Nelles “Improved Incorporation of Prior Knowledge for Regularized FIR Model Identification”. In: *American Control Conference (ACC)*. 2018.

[2] G. Pillonetto, A. Chiuso und G. De Nicolao. “Regularized estimation of sums of exponentials in spaces generated by stable spline kernels”. In: *Proceedings of the 2010 American Control Conference* S. 498-503. 2010.

[3] A. Marconato, M. Schoukens und J. Schoukens. “Filter-based regularisation for impulse response modelling”. In: *IET Control Theory; Applications 11* S. 194-204. 2016.

[4] A. Marconato and M. Schoukens “Tuning the hyperparameters of the filter-based regularization method for impulse response estimation”. In: *IFAC-PapersOnLine* S. 12841-12846. 2017.

[5] C. Rasmussen, C. Williams. “Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning series)”. The MIT Press. 2005.

[6] T. Glad and L. Ljung. “Control theory”. CRC press. 2000.

[7] T. Chen and L. Ljung “Implementation of algorithms for tuning parameters in regularized least squares problems in system identification”. In: *Automatica* S. 2213-2220. 2013.

[8] T. Chen, H. Ohlsson and L. Ljung “On the estimation of transfer functions, regularizations and Gaussian processes-Revisited”. In: *Automatica* S. 1525-1535. 2012.

[9] T. Münker, T. Peter, and O. Nelles “Gray-box identification with regularized FIR models”. In: *at - Automatisierungstechnik*. 2018.

# Robustness of Deep Learning Architectures with Respect to Training Data Variation

Andreas Bartschat[1], Tim Unger[1], Tim Scherr[1], Johannes Stegmaier[2], Ralf Mikut[1], Markus Reischl[1]

[1]Institute for Automation and Applied Informatics,
Karlsruhe Institute of Technology, Germany
[2]Institute of Imaging and Computer Vision
RWTH Aachen University, Aachen, Germany
E-Mail: andreas.bartschat@kit.edu

## 1 Motivation

The use of deep neural networks (DNN) revolutionized machine learning tasks like image classification and segmentation [6]. However, DNNs are black box models developed with the aim to generalize a given training set and thus to process unknown data. Due to the high amount of model parameters, it is yet impossible to fully understand the decision making process. The identification of parameters with poor generalization is hard and manual improvement is impossible. Thus, robustness is subject of consideration.

So called adversarial attacks evaluate robustness of trained DNNs by modifying data examples to enforce misclassification [2, 5]. In image classification, the required changes prove to be very subtle and original and modified examples differ only in minimal noise, which would be easily ignored by the human eye [9]. This leads to the question of the robustness of DNN architectures against modifications like noise or blur. Dodge et al. [1] evaluated modifications of the test images. However, the influence of modifications in the training samples remains unclear.

In this contribution we investigate the influence of modifications in the training samples to small DNN architectures with respect to their robustness against modifications in test samples. Furthermore, by introducing three different ar-

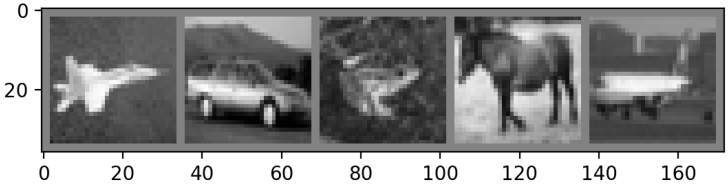



Figure 1: Five CIFAR-10 example images. Classes from left to right: plane, car, frog, horse and plane

chitectures inspired by the popular architectures ResNet [3], Inception-v3 [8] and ResNeXt [10], we investigate the influence of these architectures to their ability to generalize well against modifications. Furthermore, by modification of the training data we investigate the influence of noise and blur in the training process to the robustness of the trained model.

# 2 Methods

## 2.1 Data Set

For training and evaluation the CIFAR-10[2] data set is used [4]. It consists of 60.000 color (RGB) images with a resolution of $32 \times 32$ pixel and a pixel depth of 24 bit (8 bit per color). The data set comprises 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The images are evenly distributed over the classes and a split of 50.000 images is used for training and 10.000 for validation. Figure 1 shows five example images of the data set.

## 2.2 Network Structures

Many different network architectures and modules have been proposed over the last years. Most of these architectures consist of multiple stacked modules with small variations. A module describes a combination of different layers such as

[2] https://www.cs.toronto.edu/~kriz/cifar.html

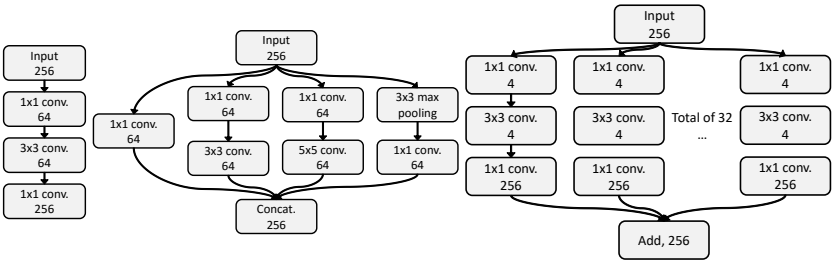



Figure 2: Modules for network architectures: Left: ResNet, middle: Inception, right: ResNeXt. The second line defines the number of feature maps produced by the module.

convolutional, activation, pooling and batch normalization layers. Modules vary in their parameters as well as in the arrangement of layers. Here, we compare three modules based on the popular architectures ResNet [3], Inception-v3 [8] and ResNeXt [10].

To allow a comparison of these modules, we use the same pre- and post-processing and the same amount of modules. The pre-processing consists of two subsequent convolution modules (with a convolutional layer, a rectified linear unit activation function and a batch normalization). After these modules, three stacked modules of the proposed architectures follow. All modules use a residual connection [3] and are followed by a max pooling for downsampling [3]. The post-processing consists of another convolution module, a dense layer with ten output neurons, one for each class of the CIFAR-10 data set and a softmax function. All architectures are trained with the stochastic gradient descent algorithm based on a categorical cross entropy loss of for 90 epochs.

The first module is the *ResNet* module, which is also called bottleneck module [3]. It downsamples the number of feature maps with a $1 \times 1$ convolution module, followed by a $3 \times 3$ convolution module for feature extraction and an upsampling of the feature maps by a $1 \times 1$ convolution module. An example can be seen in Figure 2, left. This module has shown to perform at least similar to a $3 \times 3$ convolution module while using less parameters. The given example in Figure 2, left uses $64 \times 256$ parameters for the first, $64 \times (3 \times 3 \times 64)$ parameters for the second and $256 \times 64$ for the third module, resulting in a total of 69.632 trained parameters. In contrast, a single $3 \times 3$ convolution requires $256 \times (3 \times 3 \times 256) = 589.824$ parameters.

The *Inception* module is the eponym of the Inception network. This module exists in several versions which can be combined to several versions of an Inception network. For simplicity we only investigated the module of the Inception-v3 network [8]. The module is visualized in Figure 2, middle and consists of four parallel paths with convolution modules of size $1 \times 1$, $3 \times 3$ and $5 \times 5$ as well as a max pooling layer. The idea behind the concept is that all paths can focus on features of a certain size and all results are concatenated at the output of the module. Since all of these paths work on the complete input, also the parameter count for such a module is higher. The shown module in the middle of Figure 2 contains 192.512 parameters.

The *ResNeXt* module is based on the ResNet module and introduces parallel paths. Instead of a single bottleneck, multiple bottlenecks with decreased depth are used. In the example shown in Figure 2 right, a total of 32 parallel paths are shown, each consisting of $256 \times 4$, $4 \times (3 \times 3 \times 4)$ and $256 \times 4$ parameters. This results in a total of 70.144 trainable parameters. The idea behind this module is that each bottleneck can focus on different features and each bottleneck can become an expert for different tasks.

## 2.3 Evaluation

To evaluate robustness against adversarial attacks, six models with noisy and six models for blurry inputs are trained for all three architectures. All models are evaluated in terms of accuracy and in terms of the minimal change in pixels to enforce a wrong classification of an image.

For the noisy input, a model without noise and six models with a noise level of $\sigma = \{0.01,\ 0.05,\ 0.1,\ 0.5,\ 1,\ 5\}$. $\sigma$ specifies additive Gaussian noise for each pixel originating from a normal distribution with zero mean and the given $\sigma$. The noise is added to the normalized image with a range of $[-1, 1]$. Thus, also small changes are kept in the image and not discarded due to rounding. An example of two original images and the same images with noise are shown in Figure 3.

For the blurry input, the parameters are exactly the same. Here, the image is blurred with a Gaussian filter with standard deviation sigma.

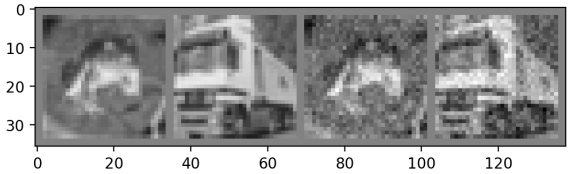



Figure 3: On the left: a frog and a truck from the original data set. On the right: the same frog and truck with added noise ($\sigma = 1$).

# 3 Results

In order to evaluate the influence of noise and blur to the training data, each trained model is evaluated either with all levels of noise or blur. Furthermore, for each model an adversarial attack is computed for 128 images of the original data set. This attack computes a minimal change of an image given a model by applying the backpropagation algorithm to modify the input image and not the model. Based on these gradients a minimal change to the image can be applied to enforce a wrong classification.

The results for the noisy inputs can be seen in Figure 4. The $x$-axis shows the noise level of the training set and is labeled by the $\sigma$ of the models' training data. The plotted lines show the accuracy on the evaluation sets, where each model is evaluated with all noise levels.

For all architectures, the accuracy and their behavior for the different noise levels are similar. The data set with a noise level of $\sigma = 0.01$ behaves like the data set without noise for the training as well as the evaluation. Starting with a model trained on the noise level $\sigma = 0.1$, the evaluation set with the same noise level provides the best accuracy and the overall accuracy of the model starts to drop. Thus, the noise levels of the evaluation sets provide the best accuracy if the model is trained with a similar noise level. These observations are true for all architectures, i.e., none of the architectures generalizes particularly well against noise. Given these results, a small level of noise has no impact to noise-free images but the accuracy for a small level of noise increases. Furthermore, the results indicate that the model should be trained at least with the amount of noise that is expected in the application data.

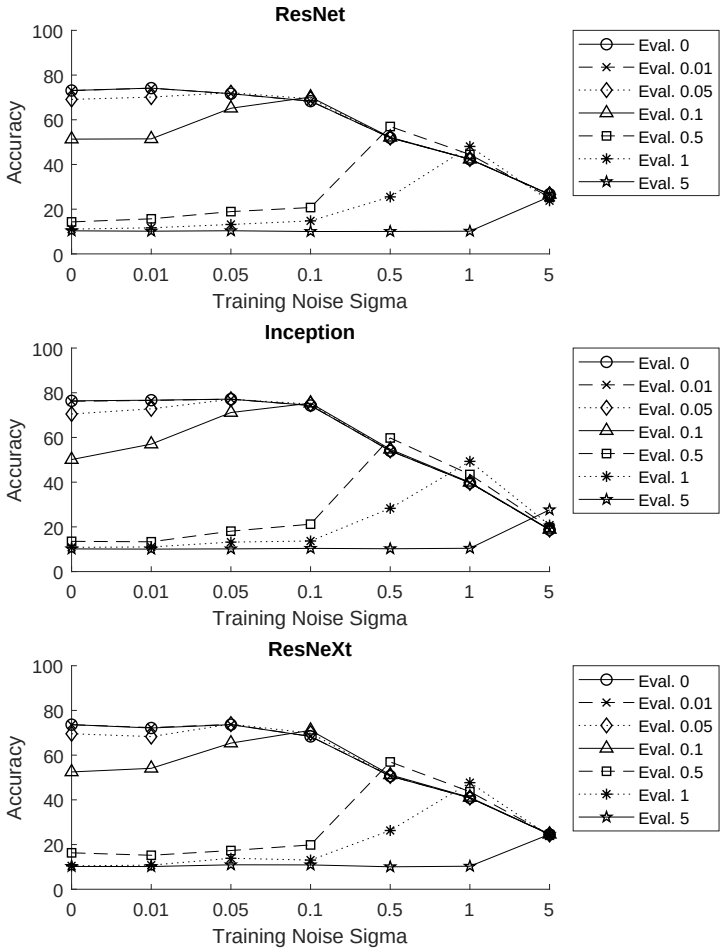



Figure 4: Evaluation of the influence of different noise levels for training ($x$-axis) on the performance of different noise levels for evaluation data sets (lines). The number behind the evaluation data sets is the $\sigma$. Three architectures are shown, the top one is based on ResNet, the middle one on Inception and the bottom one on ResNeXt.

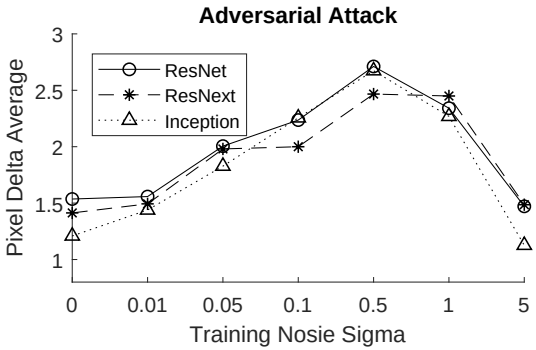



Figure 5: The average amount of required change in pixels to create an adversarial image for different architectures. The $x$-axis shows the noise level of the used training set.

The evaluation of the adversarial attacks in Figure 5 shows the noise level of the models' training set on the $x$-axis and the average delta in pixels required to change the classification of an example image on the $y$-axis. This means that on average every pixel of an image has to be modified by the given delta to change the predicted class of that image. Again, the architectures behave similar and all architectures can be tricked by adversarial attacks with a similar change in pixel values.

It is notable that models trained with noisy data are slightly more robust against adversarial attacks, since the average change in pixels must be higher. For models with a very high level of noise like $\sigma = 1$ or $\sigma = 5$ the required amount of change in the pixels starts to drop. However, these models also have a significantly lower accuracy and do not generalize well to new examples.

Overall, a small level of noise increases the robustness against adversarial attacks and the performance for noise in the evaluation data. Thus, in this case a training with $\sigma = 0.1$ seems to be beneficial for the model.

To put the results on adversarial attacks in perspective, even a change of 2.5 of the pixel value is only a change of 1% in the 8 bit images. Thus, all of the models and architectures are easily tricked by adversarial examples and none of the architectures generalizes well against such attacks.

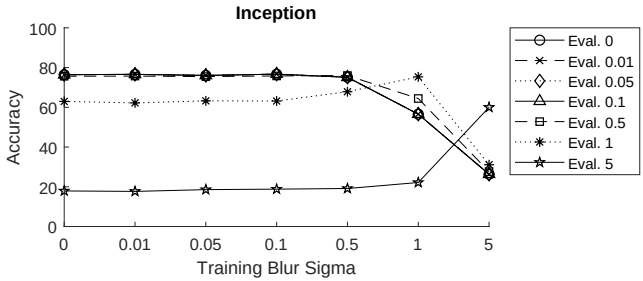



Figure 6: Evaluation of the influence of different blur levels for training ($x$-axis) on the performance of different noise level for evaluation. The figure shows the results for the Inception architecture, the results for ResNet and ResNeXt are similar (data not shown).

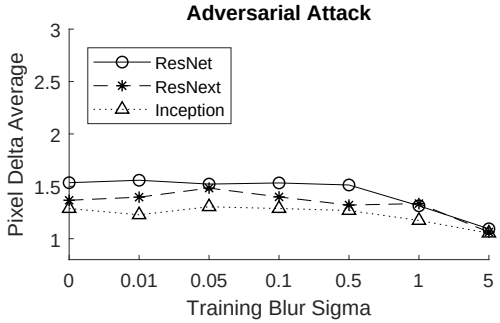



Figure 7: The average amount of required change in pixels to create an adversarial image for different architectures. The $x$-axis shows the blur level of the used training set.

The setup for the evaluation of blurry images is similar to the setup with noise ($\sigma$ defines the size of a Gaussian kernel and thus the extend of blur introduced in the images). Since the results for the different architectures are similar, Figure 6 only shows the results of the best performing models, the Inception models. Up to a blur level of $\sigma = 0.5$ the models accuracy remains unchanged and starts to drop with an increase in blur.

Figure 7 shows the evaluation of adversarial attacks on models with different blur levels. The results show that training with blurry images has no effect on the amount of change that must be introduced to the pixels to create an adver-

sarial example and again all architectures behave similar. For high blur levels of $\sigma = 1$ and $\sigma = 5$ the required change even starts to drop, indicating that models with low accuracy are more easily tricked. Overall, the training with blurry images does not have an effect for small blur levels and on adversarial attacks. Thus, it is not beneficial for this use-case.

# 4 Conclusions

This work investigated the influence of noise and blur to the accuracy and robustness of convolutional neural networks against adversarial attacks. For the training and evaluation the CIFAR-10 image data set was used and modules from three popular architectures are compared. To allow an easy comparison, the pre- and post-processing of the architectures used here are the same and only the intermediate layers are designed according to the proposed modules in the ResNet, Inception and ResNeXt architectures.

All models were trained several times with different levels of noise and blur. In the evaluation, the accuracy for different levels of noise and blur were evaluated as well as the amount of change in the pixels to create an example for an adversarial attack, meaning how much the pixels had to be changed on average to enforce a wrong classification.

The results show that training with a small noise level can increase the robustness against adversarial attacks and is beneficial for the performance of the model on noisy data. The introduction of blur however, has no beneficial effect on adversarial attacks up to a point where the models performance decreases significantly. Regarding the performance of the proposed modules no significant performance difference was observed.

Since augmentations of the training set are known to improve the model performance [7], also other augmentations like rotations and other transformations can be investigated regarding their effects on the robustness.

## References


[1] S. Dodge and L. Karam. Understanding how Image Quality Affects Deep Neural Networks. In *2016 Eighth International Conference on Quality of Multimedia Experience*, pp. 1–6. IEEE, 2016.

[2] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572*, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[4] A. Krizhevsky and G. Hinton. Learning Multiple Layers of Features from Tiny Images. Technical Report, Citeseer, 2009.

[5] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial Examples in the Physical World. *arXiv preprint arXiv:1607.02533*, 2016.

[6] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3), pp. 211–252, 2015.

[7] T. Scherr, A. Bartschat, M. Reischl, J. Stegmaier, and R. Mikut. Best Practices in Deep Learning-Based Segmentation of Microscopy Images. In *Proc., 28. Workshop Computational Intelligence, Dortmund*, 2018.

[8] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing Properties of Neural Networks. *arXiv preprint arXiv:1312.6199*, 2013.

[10] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated Residual Transformations for Deep Neural Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5987–5995, IEEE, 2017.

# Risk Sensitive Decision Making Using Type Reduction Methods

Thomas A. Runkler[1], Chao Chen[2], Robert John[2]

[1]Siemens AG
Corporate Technology
81739 Munich, Germany

[2]University of Nottingham
Wollaton Road
Nottingham, NG8 1BB, UK

E-Mail: Thomas.Runkler@siemens.com

## 1 Introduction

Decision making processes (e.g. in recommender systems) are often based on data containing co–occurences (e.g., which items are often purchased together) [6], ratings (e.g., zero stars to five stars) [1], or preferences (e.g., which option is preferred over which other options) [10]. In this paper we consider decision making based on ratings. Normalized ratings can be considered as membership values in the fuzzy set of suitable options. For example, $n$ of five stars may be mapped to a membership of $u = n/5$. Based on such memberships a fuzzy decision making process [2] will choose the option with maximum membership.

Often experts are not willing or not able to specify exact ratings but only to specify intervals of ratings (e.g., between three and four stars, or satisfaction between 60% and 80%), so the ratings contain uncertainty. Intervals of ratings may be represented as intervals of memberships (e.g., $u \in [0.6, 0.8]$). Such interval ratings can be considered as interval–valued fuzzy sets [4, 5] or interval type–2 fuzzy sets [7, 8, 14]. Interval type–2 fuzzy decision making [12] will choose the most suitable option based on the individual membership intervals, taking into account the degree of risk that the decision maker is willing to take.

Here we consider an interval type–2 fuzzy decision making approach that employs a type reduction process [9] which maps each membership interval (type–2) to a single membership value (type–1) and then chooses the option with maximum membership. Several different methods for type reduction have been proposed in the literature, for example the Nie–Tan method (NT) [9], consistent linear type reduction (CLTR) [11], the uncertainty weight method (UW) [13], and consistent quadratic type reduction (CQTR) [11].

This paper presents an experimental study comparing these four type reduction methods to a decision making process using four alternatives.

## 2 Type Reduction Methods

Given an upper membership $\overline{u} \in [0,1]$ and a lower membership $\underline{u} \in [0,1]$, where $\underline{u} \leq \overline{u}$, the Nie–Tan method (NT) [9] is defined by the type reduction formula that yields the membership

$$u_{\mathrm{NT}} = \frac{\underline{u} + \overline{u}}{2} \tag{1}$$

The consistent linear type reduction (CLTR) [11] is defined by

$$u_{\mathrm{CLTR}} = a \cdot \underline{u} + (1-a) \cdot \overline{u} \tag{2}$$

with a parameter $a \in [0,1]$ that quantifies the degree of caution in the decision making process. The uncertainty weight method (UW) [13] is defined as

$$u_{\mathrm{UW}} = \frac{1}{2}(\underline{u} + \overline{u}) \cdot (1 + \underline{u}(x) - \overline{u}(x))^{\alpha} \tag{3}$$

with the parameter $\alpha \geq 0$. For easier comparison of the different methods we set

$$\alpha = \operatorname{atanh}(a) \tag{4}$$

so we can keep the parameter $a \in [0,1]$ and for $a = 0$ we have $\alpha = 0$ and for $a \to 1$ we have $\alpha \to \infty$. The consistent quadratic type reduction (CQTR) [11] is defined as

$$u_{\mathrm{CQTR}} = a \cdot \underline{u} + (1-a) \cdot \overline{u} - (1-a) \cdot (\overline{u} - \underline{u})^2 \tag{5}$$

# 3 Experiments

In this section we will compare the four type reduction methods presented in the previous section in experiments with a decision making process for four alternatives with the following membership values:

$$u_1 \in [0,1] \tag{6}$$
$$u_2 \in [0.55,0.95] \tag{7}$$
$$u_3 \in [0.7,0.75] \tag{8}$$
$$u_4 = 0.5 \tag{9}$$

We want to emphasize at this point that the numerical values in these experiments have a fuzzy and not a probabilistic character, so they represent memberships, not probabilities [3]. So $u_1$, an interval between zero and one does not represent a lack of information that may be changed after an experiment when further information is provided (e.g. about the director of a movie or the recipe of a drink). Instead, an interval between zero and one indicates a large uncertainty in the rating that is between zero and one. And $u_4$, a singleton membership of 0.5 does not represent a 50% probability whether this option is desirable or not but a 50% degree of desirability. The membership intervals $u_2$ and $u_3$ are both higher than 0.5, $u_2$ has a higher uncertainty than $u_3$, but $u_2$ has a higher average than $u_3$.

Fig. 1 shows the results obtained for these data with the Nie–Tan (NT) method. The three grey vertical bars in this plot represent the membership intervals of the first three options, and the dash–dotted horizontal line represents the membership of the fourth option. The four horizontal lines (solid, dashed, dotted, and dash–dotted) represent the output of the NT method. For the fourth option, the NT method yields the original membership value. In this case, the input is type–1, so type reduction is not necessary. A type reduction operator with this property is called type–1 consistent [11]. For the other three options, NT yields the average of the intervals. The maximum of these averages is for $u_3$, so the decision process will prefer option 3. In this decision the degree of uncertainty of the different options is not taken into account. A cautious decision maker may feel incomfortable that $u_2$ has a lower membership of only

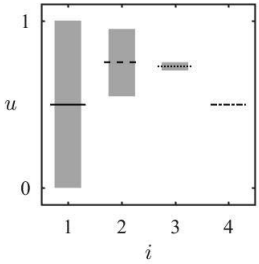



Figure 1: Decision memberships for the ratings $u_i$, $i = 1, \ldots, 4$, obtained by the Nie–Tan (NT) method.

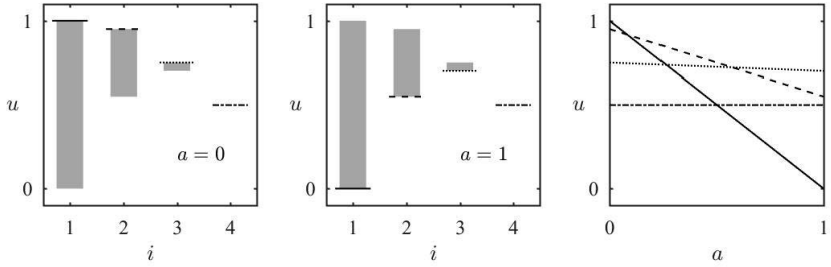



Figure 2: Decision memberships for the consistent linear type reduction (CLTR) method.

0.55 and might prefer $u_3$ with a lower membership of 0.7. On the other hand, a very risky decision maker may not be satisfied that $u_2$ has an upper membership of only 0.95 and might even prefer $u_1$ with an upper membership of 1. The degree of risk that the decsion maker is willing to take is taken into account in the three other type reduction methods.

Fig. 2 shows the results obtained with the consistent linear type reduction (CLTR) method. CLTR is a parametric method with the parameter $a \in [0,1]$. The left plot of Fig. 2 shows the results for $a = 0$, where CLTR always yields the maximum of each interval, with a maximum for option 1. The middle plot of Fig. 2 shows the results for $a = 1$, where CLTR always yields the minimum of each interval, with a maximum for option 3. The right plot of Fig. 2 shows the results for all $a \in [0,1]$ that are plotted along the horizontal axis. So, in contrast to the left and middle plots, the right plot does not display the four

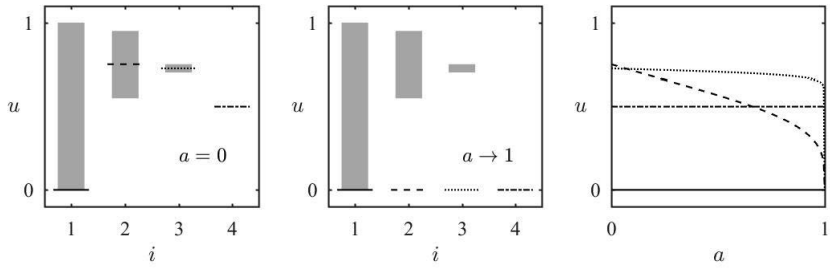



Figure 3: Decision memberships for the uncertainty weight (UW) method.

different options on the horizontal axis but $a$. The four different options are represented by the four different curves: solid for option 1, dashed for option 2, dotted for option 3, and dash–dotted for option 4. For very small values of $a < 0.0833$ the solid curve is on top, so the risky decision is option 1. For $a \in [0.0833, 0.5714]$ the dashed curve is on top, so the medium risk decision is option 2. And for large $a > 0.5714$ the dotted curve is on top, so the cautious decision is option 3. Option 1, the option with complete uncertainty is chosen in the risky decision here. In many applications this is not desirable. Instead, options with complete uncertainty should often be completely ignored. A type reduction operator with this property ignores indifference [11]. The two remaining type reduction operators, UW and CQTR both ignore indifference.

Fig. 3 shows the results obtained with the uncertainty weight (UW) method. Again, the left plot shows the results for $a = 0$, the middle plot shows the results for $a = 1$, and the right plot shows the results for all $a \in [0, 1]$. For option 1 (solid curve) we always obtain zero membership, so complete indifference is ignored, as pointed out above. For $a = 0$ we obtain memberships at the mean of each interval. In the limit for $a \to 1$ all memberships approach zero, which appears counter–intuitive. For small $a < 0.074$ the dashed curve is on top (option 2), and for larger $a > 0.074$ the dotted curve is on top (option 3). For large $a > 0.661$ the dashed curve (option 2) falls below the dash–dotted curve (option 4) so the option $u_4 = 0.5$ would be preferred over $u_2 \in [0.55, 0.95]$ although all values of $u_2$ are higher than $u_4$. Also this behaviour seems counter–intuitive.

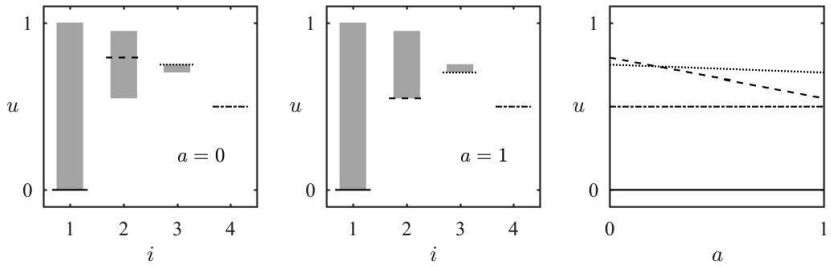



Figure 4: Decision memberships for the consistent quadratic type reduction (CQTR) method.

Fig. 4 shows the results obtained with the consistent quadratic type reduction (CQTR) method. Again we always obtain membership zero for option 1 (solid curve), so complete indifference is ignored. For $a = 1$ we obtain the memberships of the bottom of each interval. For $a < 0.2208$ the dashed curve is on top and for $a > 0.2208$ the dotted curve is on top, so for low degrees of caution option 2 is chosen, and for high degrees of caution option 3 is chosen. The dash–dotted curve (option 4) is always below the dotted curve (option 3), which matches the intuitive expectation.

## 4 Conclusions

In this paper we have considered decision making processes based on interval ratings. Each rating is represented as an interval of memberships in the set of suitable options. Type reduction methods are used to convert the membership intervals to crisp memberships, and then the option with the highest membership is chosen.

We have considered four different type reduction operators from the literature: the Nie–Tan method (NT), consistent linear type reduction (CLTR), the uncertainty weight method (UW), and consistent quadratic type reduction (CQTR). All four type reduction operators were applied in experiments with four alternatives with different degrees of utility and different degrees of uncertainty.

The experiments have shown that

- NT does not consider the degree of uncertainty and is not able to take into account the degree of risk that the decision maker is willing to take.
- CLTR prefers an option with maximum uncertainty, if the risk level is high enough.
- UW prefers an option with a completely certain rating over a more uncertain option although the uncertain rating is always better than the certain one.
- CQTR is the only one of these four type reduction operators that is able to take into account the risk in the decision process, that ignores indifference, and that does not prefer a crisp membership over an interval of higher memberships.

Based on these results we would recommend CQTR as a type reduction method for risk sensitive decision makers.

## References


[1] R. M. Bell and Y. Koren. Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[2] R. Bellman and L. Zadeh. Decision making in a fuzzy environment. *Management Science*, 17(4):141–164, 1970.

[3] J. C. Bezdek. Fuzzy models — what are they, and why? *IEEE Transactions on Fuzzy Systems*, 1(1):1–6, 1993.

[4] M. Gehrke, C. Walker, and E. Walker. Some comments on interval valued fuzzy sets. *International Journal of Intelligent Systems*, 11(10):751–759, 1996.

[5] M. B. Gorzałczany. A method of inference in approximate reasoning based on interval–valued fuzzy sets. *Fuzzy Sets and Systems*, 21(1):1–17, 1987.

[6] M. Hildebrandt, S. S. Sunder, S. Mogoreanu, I. Thon, V. Tresp, and T. Runkler. Configuration of industrial automation solutions using multi–

relational recommender systems. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Dublin, Ireland, September 2018.

[7] Q. Liang and J. M. Mendel. Interval type–2 fuzzy logic systems: Theory and design. *IEEE Transactions on Fuzzy Systems*, 8(5):535–550, 2000.

[8] J. M. Mendel, R. I. John, and F. Liu. Interval type–2 fuzzy logic systems made simple. *IEEE Transactions on Fuzzy Systems*, 14(6):808–821, 2006.

[9] M. Nie and W. W. Tan. Towards an efficient type–reduction method for interval type–2 fuzzy logic systems. In *IEEE International Conference on Fuzzy Systems*, pages 1425–1432, Hong Kong, 2008.

[10] T. A. Runkler. Mapping utilities to transitive preferences. In Jesús Medina, Manuel Ojeda-Aciego, José Luis Verdegay, David A. Pelta, Inma P. Cabrera, Bernadette Bouchon-Meunier, and Ronald R. Yager, editors, *Information Processing and Management of Uncertainty in Knowledge–Based Systems. Theory and Foundations*, volume CCIS 853, pages 127–139. Springer, 2018.

[11] T. A. Runkler, C. Chen, and R. John. Type reduction operators for interval type–2 defuzzification. *Information Sciences*, 467C:464–476, 2018.

[12] T. A. Runkler, S. Coupland, and R. John. Interval type–2 fuzzy decision making. *International Journal of Approximate Reasoning*, 80:217–224, 2017.

[13] T. A. Runkler, S. Coupland, R. John, and C. Chen. Interval type–2 defuzzification using uncertainty weights. In S. Mostaghim, C. Borgelt, and A. Nürnberger, editors, *Frontiers in Computational Intelligence*, pages 47–59. Springer, 2017.

[14] L. A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning. *Information Sciences*, 8:199–249, 9:43–80, 1975.

# Fahrspurerkennung mit Deep Learning für automatisierte Fahrfunktionen

Manuel Schmidt[1], Christian Lienke[1], Malte Oeljeklaus[1], Martin Krüger[2], Till Nattermann[2], Manoj Mohamed[2], Frank Hoffmann[1], Torsten Bertram[1]

[1]Lehrstuhl für Regelungssystemtechnik, TU Dortmund
E-Mail: manuel3.schmidt@tu-dortmund.de

[2]ZF Group TRW Automotive
Automated Driving & Integral Cognitive Safety
40547 Düsseldorf

## 1 Einführung

Moderne Fahrzeuge verfügen über aktive Spurhaltesysteme, die ein ungewolltes Verlassen der Fahrspur erkennen und diesem durch Lenkeingriff entgegenwirken. Solche Systeme basieren häufig auf Kameras und die Funktion ist nur bei Vorhandensein von Spurmarkierungen gegeben. In urbanen Szenarien tritt häufig der Fall ein, dass Spurmarkierungen nicht vorliegen oder durch Schatten und andere Fahrzeuge verdeckt sind. Klassische modellbasierte Verfahren, wie beispielsweise das in [1] vorgeschlagene, versagen dabei. Menschen können mit dieser Problemstellung aufgrund ihres weitreichenden kontextuellen Wissens sowie der Objektkonstanz und Objektpermanenz [2] umgehen. Bereits Kinder im Alter von drei bis vier Monaten verfügen bereits über das Wissen, dass Objekte auch dann weiter existieren, wenn sie durch ein Hindernis verdeckt werden und so aus dem Blickfeld geraten. Aktuelle Bildverarbeitungsalgorithmen sind nicht in der Lage die aufgezeigten Eigenschaften des menschlichen Gehirns vollständig nachzubilden. Durch den Einsatz von Deep Learning (vergleiche [3]) wird versucht, diese Funktionalitäten zu reproduzieren.

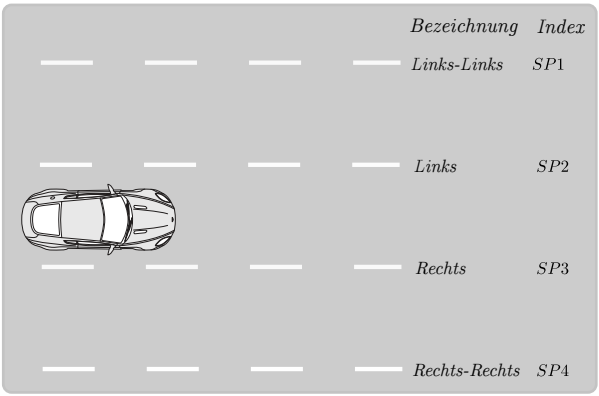



Bild 1: Verwendete Bezeichnungen im Rahmen der Beschreibung zur Schätzung von Spurmarkierungen.

Dies geschieht durch eine Extraktion von Merkmalen aus Kamerabildern und dem statistischen Lernen des Zusammenhangs zwischen den Merkmalen und den Ausgangsdaten. Der vorliegende Beitrag beschäftigt sich mit dem Einsatz von Deep Learning zur Schätzung von Fahrspuren. Besonderes Augenmerk liegt auf der Bereitstellung dieser Funktion für eine möglichst große Klasse an Szenarien. Die Bezeichnungen von Spurmarkierung und Indizierung der zugehörigen Variablen sind in Bild 1 dargestellt. Die Ego-Fahrspur wird durch die Spurmarkierungen mit den Bezeichnungen Links (Index $SP2$) und Rechts (Index $SP3$) begrenzt. Mit diesen befasst sich Abschnitt 2 dieses Beitrages. Hingegen ist die Schätzung der Spurmarkierungen Links-Links (Index $SP1$) und Rechts-Rechts (Index $SP4$) Gegenstand von Abschnitt 3.

## 2 Schätzung der Ego-Fahrspur mittels Deep Learning

In der Literatur existieren mehrere Ansätze, die sich mit der Schätzung von Spurmarkierungen auf Basis von Deep Learning beschäftigen. In [4] wird ein Convolutional Neural Network (siehe [5]) sowie ein rekurrentes Netz zur Strukturprädiktion von Verkehrsszenen eingesetzt. Dies umfasst unter anderem eine Schätzung von Spurmarkierungen. Die Arbeit von [6] nutzt eine Kombination

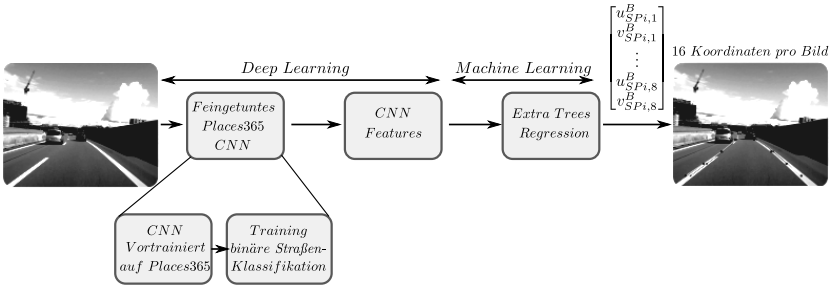



Bild 2: Architektur aus [8] zur Schätzung der Ego-Fahrspur.

aus einem Kantenfilter, einem Convolutional Neural Network und dem RANSAC Algorithmus [7] zur robusten Schätzung von Spurmarkierungen.

In [8] wird ein Convolutional Neural Network zur Merkmalsextraktion verwendet. Einen Überblick über die Architektur gibt Bild 2. Sie wird im Folgenden genauer beschrieben.

Das Netz basiert auf einem *AlexNet* (vergleiche [9]), das auf dem *Places365* Datensatz (vergleiche [10]) zur Szenenklassifikation trainiert wird. Zur Ausprägung der Merkmale zur Spurerkennung erfolgt das Feintuning dieses Netzes im Rahmen einer binären Klassifikation von Straßenszenen. Dazu wird ein Datensatz erzeugt, der aus Bildern von Straßenszenen und Bildern beliebiger anderer Szenen besteht. Das Netz lernt durch das Training relevante Merkmale aus Bildern die Straßen enthalten. Diese dienen anschließend dem Training von Extra Trees (siehe [11]) zur Regression von Bezier-Spline Stützstellen. Dazu wird zuvor ein Bereich im Kamerabild definiert, in welchem die Positionierung der Stützstellen $\mathbf{x}^B_{SPi,j} = \left[u^B_{SPi,j}, v^B_{SPi,j}\right]^\mathsf{T}$ erfolgen soll. Der Index $i$ steht für die Nummer der Spurmarkierung und der Index $j$ charakterisiert die Nummer der Stützstelle. Eine Stützstelle besteht aus den beiden Bildkoordinaten $u^B_{SPi,j}$ und $v^B_{SPi,j}$. Das hochgestellte B $(\cdot)^B$ kennzeichnet das zugrundeliegende Bildkoordinatensystem. Pro Bild erfolgt die Schätzung von zwei Spurmarkierungen mittels jeweils vier Stützstellen. Exemplarisch zeigt dies Bild 3. Darin sind auch die Bezeichnungen der Stützstellen dargestellt.

Die Gesamtzahl der geschätzten Koordinaten pro Bild beträgt damit Sechzehn. Limitierungen des Ansatzes bestehen in zweierlei Hinsicht. Zum einen werden

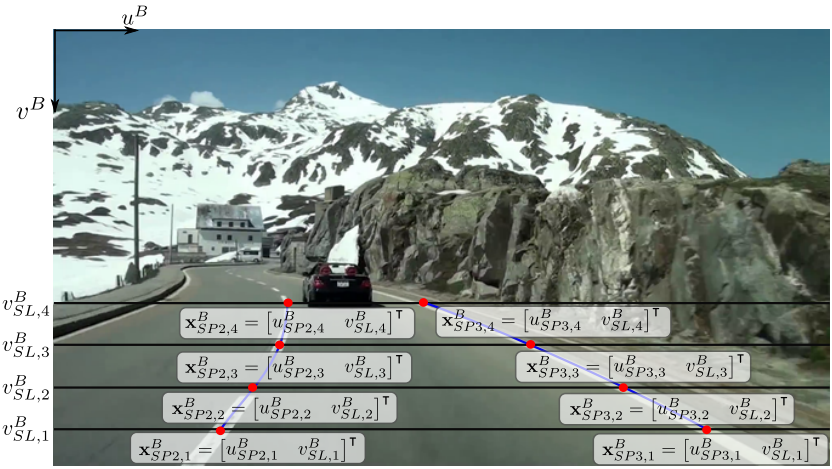



Bild 3: Bezeichnungsweise von Stützstellen $\mathbf{x}^B_{SPi,j}$.

ausschließlich zwei Spurmarkierungen geschätzt. Für Spurwechselvorgänge von automatisierten Fahrzeugen reicht diese Information daher nicht aus. Zum anderen hat das Training der Extra Trees keinerlei Einfluss auf die Merkmalsextraktion durch das Convolutional Neural Network. Der Trainingsvorgang ist sequentiell und nach der binären Straßenklassifikation ist das Netz unveränderlich. Dabei ist davon auszugehen, dass die Fehler, die beim Training der Regression entstehen, sich gut zur Optimierung des neuronalen Netzes nutzen lassen. Beide Mängel werden von dem, in diesem Beitrag vorgeschlagenen Ansatz behoben. Die vorgeschlagene Architektur zur Schätzung der Spurmarkierung der Ego-Fahrspur zeigt Bild 4. Die Repräsentation der geschätzten Spurmarkierungen erfolgt durch die $u^B$-Koordinaten der Stützstellen von zwei kubischen Splines. Zur Approximation werden vier Stützstellen pro Spline verwendet. Linke und rechte Spurmarkierung werden je durch einen Spline dargestellt.

Das Training ist angelehnt an die von [8] vorgeschlagene Architektur. Nachgeschaltet wurde jedoch das Training auf einem eigens erzeugten Datensatz zur Klassifikation der Anzahl befahrbarer Fahrspuren. Die Klassifikation wird im Rahmen dieses Beitrages nicht weiter ausgeführt. Diese Maßnahme sorgt für eine Verbesserung der Schätzung der Spurmarkierung der Ego-Fahrspur. Durch

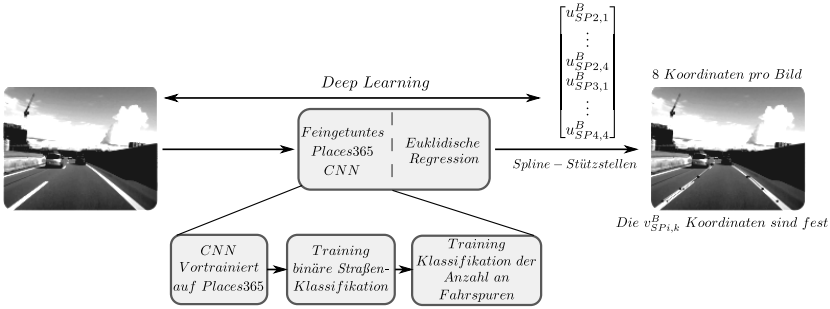



Bild 4: Vorgeschlagene Architektur zur Schätzung der Ego-Fahrspur.

die Repräsentation der Regression mittels Fully-connected Schichten wird erreicht, dass das Training vollständig in der Deep Learning Entwicklungsumgebung ablaufen kann. Während des Trainings können Fehler, die bei der Regression auftreten, im Zuge der Backpropagation durch sämtliche Schichten des Netzes propagiert werden. Dadurch wird die Merkmalsextraktion optimiert, was sich ebenfalls in verbesserten Schätzungen widerspiegelt. Gleichzeitig erfolgt eine Komplexitätsreduktion, indem der von [8] vorgeschlagene Suchbereich im Bild durch vier diskrete Suchlinien mit festen $v^B$-Koordinaten $v^B_{SL,i}$ ersetzt wird. Für jede Stützstelle entfällt so die Notwendigkeit der Schätzung der $v^B$-Koordinate. Zur Schätzung verbleiben so lediglich die acht $u^B$-Koordinaten der Stützstellen, jeweils vier pro Spurmarkierung.

Die Repräsentation mittels kubischer Splines wird nachfolgend, in Anlehnung an [12], eingeführt. Dazu wird die tatsächliche Spurmarkierung als ein funktionaler Zusammenhang $Z_{SPi}$ zwischen den $u^B$- und $v^B$-Koordinaten aufgefasst:

$$u^B_{SPi} = Z_{SPi}(v^B_{SPi}),\ i \in \{2,3\}. \tag{1}$$

Zur Interpolation werden vier feste $v^B$-Koordinaten, zuvor als Suchlinien bezeichnet, $v^B_{SL,1}$ bis $v^B_{SL,4}$ herausgegriffen. Es ist sicherzustellen, dass in dem horizontalen Bildabschnitt, der durch das Intervall $[v^B_{SL,1}, v^B_{SL,4}]$ definiert wird, tatsächlich Markierungen vorliegen. Gewählt werden $\{v^B_{SL,1}, v^B_{SL,2}, v^B_{SL,3}, v^B_{SL,4}\} = \{225, 260, 295, 330\}$. Die Eingangsdimension der Bilder beträgt $640\,\text{px} \times 360\,\text{px}$. Als Stützstellen gelten die Punkte $\left[u^B_{SPi,j}, v^B_{SPi,j}\right]^\mathsf{T}, i \in \{2,3\}, j \in$

$\{1,2,3,4\}$. In jedem Teilintervall $[v^B_{SL,k}, v^B_{SL,k+1}], k \in \{1,2,3\}$ wird ein Polynom $S_{SPi,k}(v^B)$ dritter Ordnung definiert:

$$\begin{aligned} S_{SPi,k}(v^B_{SPi}) =& a_{SPi,k} + b_{SPi,k}(v^B_{SPi} - v^B_{SL,k}) + c_{SPi,k}(v^B_{SPi} - v^B_{SL,k})^2 + \\ &+ d_{SPi,k}(v^B_{SPi} - v^B_{SL,k})^3. \end{aligned} \tag{2}$$

Durch Forderung der gleichzeitigen Erfüllung von sechs Bedingungen werden die Gleichungen zur Bestimmung der Koeffizienten $a_{SPi,k}$, $b_{SPi,k}$, $c_{SPi,k}$ und $d_{SPi,k}$ hergeleitet. Die Bedingungen lauten:

$$\text{Interpolation am linken Rand} : S_{SPi,k}(v_k) = a_{SPi,k} = u^B_{SPi,k}, \tag{3}$$

$$\text{Stetigkeit der Funktion} : S_{SPi,k}(v^B_{SL,k}) = S_{SPi,k-1}(v^B_{SL,k}), \tag{4}$$

$$\text{Stetigkeit der ersten Ableitung} : S'_{SPi,k}(v^B_{SL,k}) = S'_{SPi,k-1}(v^B_{SL,k}), \tag{5}$$

$$\text{Stetigkeit der zweiten Ableitung} : S''_{SPi,k}(v^B_{SL,k}) = S''_{SPi,k-1}(v^B_{SL,k}), \tag{6}$$

$$\text{Krümmung am Anfang Null} : c_{SPi,1} = 0, \tag{7}$$

$$\text{Krümmung am Ende Null} : c_{SPi,4} = 0. \tag{8}$$

Es ergeben sich durch Nutzung der Bedingungen die folgenden Bestimmungsgleichungen für die unbekannten Koeffizienten $c_{SPi,k}$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_{SPi,1} \\ c_{SPi,2} \\ c_{SPi,3} \\ c_{SPi,4} \end{bmatrix} = \frac{3}{h^2} \begin{bmatrix} 0 \\ u^B_{SPi,1} - 2u^B_{SPi,2} + u^B_{SPi,3} \\ u^B_{SPi,2} - 2u^B_{SPi,3} + u^B_{SPi,4} \\ 0 \end{bmatrix}, \tag{9}$$

mit der konstanten Schrittweite $h = (v^B_{SL,k} - v^B_{SL,k-1}) = 35$. Aus den $c_{SPi,k}$ werden die übrigen Koeffizienten berechnet:

$$d_{SPi,k} = \frac{1}{3h}(c_{SPi,k+1} - c_{SPi,k}), \tag{10}$$

$$b_{SPi,k} = \frac{1}{h}(u^B_{SPi,k+1} - u^B_{SPi,k}) - \frac{h}{3}(c_{SPi,k+1} + 2c_{SPi,k}). \tag{11}$$

Das Resultat ist eine Splinefunktion für jedes Teilintervall:

$$S_{SPi,k}(v^B_{SPi}) = u^B_{SPi,k} + b_{SPi,k}(v^B_{SPi} - v^B_{SL,k}) + c_{SPi,k}(v^B_{SPi} - v^B_{SL,k})^2 + \\ + d_{SPi,k}(v^B_{SPi} - v^B_{SL,k})^3, \text{für } v^B_{SL,k} \leq v^B_{SPi} \leq v^B_{SL,k+1}. \quad (12)$$

Mathematisch gilt für eine Spurmarkierung in dieser Spline-Repräsentation:

$$u^B_{SPi} = P_{SPi}(v^B_{SPi}) = \begin{cases} S_{SPi,1}(v^B_{SPi}) & \text{für } v^B_{SL,1} \leq v^B_{SPi} \leq v^B_{SL,2} \\ S_{SPi,2}(v^B_{SPi}) & \text{für } v^B_{SL,2} \leq v^B_{SPi} \leq v^B_{SL,3} \\ S_{SPi,3}(v^B_{SPi}) & \text{für } v^B_{SL,3} \leq v^B_{SPi} \leq v^B_{SL,4} \end{cases}. \quad (13)$$

Zum Vergleich der Leistungsfähigkeit der Regression mittels Extra Trees beziehungsweise der vollständigen Integration in die Deep Learning Umgebung, wird die Architektur nach Bild 2 nachgebildet. Die Implementierung unterscheidet sich vom Original von [8] dadurch, dass die Komplexitätsreduktion durch die Einführung der Suchlinien auch für diesen Ansatz vorgenommen wird. Außerdem kommen auch hier kubische Splines statt der Bezier-Splines zum Einsatz. Dies ermöglicht die Vergleichbarkeit sämtlicher Ergebnisse. Zur Regression der acht Stützstellen werden Extra Trees bestehend aus $N = 50$ Entscheidungsbäumen eingesetzt. Zur Implementierung wird die *scikit-learn* Bibliothek nach [13] verwendet.

Als Basis dienen ein *AlexNet* und ein *VGG16* Netz (vergleiche [14]), die jeweils auf dem *Places365* Datensatz zur Szenenklassifikation trainiert werden. Die Auswahl einsetzbarer Kostenfunktionen ist dadurch beschränkt, dass für die Backpropagation während des Trainings die Gradienten der Kosten nach den Eingängen benötigt werden. Die Eingänge sind die, vom Netz vorgeschlagenen $u^B$-Koordinaten der Stützstellen.

Einerseits wird eine Kostenfunktion auf Basis der Fehlerquadrate:

$$L_2 = L_{2,SP2} + L_{2,SP3}, \tag{14}$$

mit:

$$L_{2,SPi} = \frac{1}{2N}\sum_{i=1}^{N}(u^B_{SPi,j} - \hat{u}^B_{SPi,j})^2 = \frac{1}{8}\sum_{i=1}^{4}(u^B_{SPi,j} - \hat{u}^B_{SPi,j})^2, i \in \{2,3\} \tag{15}$$

eingesetzt. Die tatsächlichen Stützstellen sind darin mit $u^B_{SPi,j}$ beschrieben und die Schätzungen mit $\hat{u}^B_{SPi,j}$. Für die Backpropagation erfolgt die Berechnung des Gradienten der Kostenfunktion $L_{2,SPi}$ nach den Eingängen $\hat{u}^B_{SPi,j},\ j \in \{1,\ldots,4\}$ gemäß:

$$\frac{\partial L_{2,SPi}}{\partial \hat{u}^B_{SPi,j}} = \frac{\hat{u}^B_{SPi,j} - u^B_{SPi,j}}{N} = \frac{\hat{u}^B_{SPi,j} - u^B_{SPi,j}}{4}, i \in \{2,3\}. \tag{16}$$

Andererseits wird eine weitere Kostenfunktion $L_{Spline}$ untersucht. Sie basiert auf den Repräsentationen der echten und geschätzten Spurmarkierungen durch die Splines $P_{SP2}, P_{SP3}$ und $\hat{P}_{SP2}, \hat{P}_{SP3}$ gemäß:

$$L_{Spline} = L_{Spline,SP2} + L_{Spline,SP3}, \tag{17}$$

mit

$$L_{Spline,SPi} = \frac{1}{2N_{sp}}\sum_{j=1}^{N_{sp}}(\hat{P}_{SPi}(v^B_{SPi,j}) - P_{SPi}(v^B_{SPi,j}))^2 \tag{18}$$

$$= \frac{1}{420}\sum_{j=1}^{210}(\hat{P}_{SPi}(v^B_{SPi,j}) - P_{SPi}(v^B_{SPi,j}))^2, i \in \{2,3\}. \tag{19}$$

Die Auswertung erfolgt somit an $N_{SP} = 210$ Stellen.

Für die Backpropagation wird der Gradient der Kostenfunktion $L_{Spline}$ nach den Eingängen $\hat{u}^B_{SPi,j}$ $j \in \{1,\ldots,4\}$ benötigt:

$$\frac{\partial L_{Spline}}{\partial \hat{u}^B_{SPi,j}} = \frac{\partial L_{Spline,SP2}}{\partial \hat{u}^B_{SPi,j}} + \frac{\partial L_{Spline,SP3}}{\partial \hat{u}^B_{SPi,j}}. \tag{20}$$

Dabei gilt:

$$\frac{\partial L_{Spline,SPi}}{\partial \hat{u}^B_{SPi,j}} = \frac{1}{N_{SP}} \sum_{j=1}^{N_{SP}} (\hat{P}_{SPi}(v^B_{SPi,j}) - P_{SPi}(v^B_{SPi,j})) \cdot \frac{\partial \hat{P}_{SPi}(v^B_{SPi,j})}{\partial \hat{u}^B_{SPi,j}},$$
$$i \in \{2,3\}. \tag{21}$$

Für die beiden äußeren Splinestützstellen $\begin{bmatrix} \hat{u}^B_{SPi,1} & \hat{v}^B_{SL,1} \end{bmatrix}^\mathsf{T}$ sowie $\begin{bmatrix} \hat{u}^B_{SPi,4} & \hat{v}^B_{SL,4} \end{bmatrix}^\mathsf{T}$ werden Vorwärts- respektive Rückwärtsdifferenzen erster Ordnung benutzt. An den Stützstellen $\begin{bmatrix} \hat{u}^B_{SPi,2} & \hat{v}^B_{SL,2} \end{bmatrix}^\mathsf{T}$ und $\begin{bmatrix} \hat{u}^B_{SPi,3} & \hat{v}^B_{SL,3} \end{bmatrix}^\mathsf{T}$ werden die Ableitungen mit zentralen Differenzen zweiter Ordnung bestimmt.

Hervorzuheben ist, dass die Splinefunktionen:

$$\hat{P}_{SPi}(v^B_{SPi,j}) = \hat{P}_{SPi}(v^B_{SPi,j}, \hat{u}^B_{SPi,1}, \hat{u}^B_{SPi,2}, \hat{u}^B_{SPi,3}, \hat{u}^B_{SPi,4}), \tag{22}$$

von den vier $u^B$-Koordinaten der Stützstellen abhängen.

Insgesamt lauten die Berechnungen der Ableitungen an der Stelle $v^B_{SL,j}$:

$$\frac{\partial \hat{P}_{SPi}(v^B_{SPi,j})}{\partial \hat{u}^B_{SPi,j}}\Big|_{v^B_{SPi,j}=v^B_{SL,j}} = \dots$$

$$\begin{cases} \frac{-\hat{P}_{SPi}(v^B_{SL,1},\hat{u}^B_{SPi,1}+2h_D,\dots)+4\hat{P}_1(v^B_{SL,1},\hat{u}^B_{SPi,1}+h_D,\dots)-3\hat{P}_{SPi}(v^B_{SL,1},\hat{u}^B_{SPi,1},\dots)}{2h_D}, \\ \quad \text{für } j=1 \\ \frac{-\hat{P}_{SPi}(v^B_{SL,,2},\dots,\hat{u}^B_{SPi,2}+2h_D,\dots)+8\hat{P}_{SPi}(v^B_{SL,2},\dots,\hat{u}^B_{SPi,2}+h_D,\dots)}{12h_D}-\dots \\ \frac{8\hat{P}_{SPi}(v^B_{SL,,2},\dots,\hat{u}^B_{SPi,2}-h_D,\dots)+\hat{P}_{SPi}(v^B_{SL,2},\dots,\hat{u}^B_{SPi,2}-2h_D,\dots)}{12h_D}, \\ \quad \text{für } j=2 \\ \frac{-\hat{P}_{SPi}(v^B_{SL,3},\dots,\hat{u}^B_{SPi,3}+2h_D,\dots)+8\hat{P}_{SPi}(v^B_{SL,3},\dots,\hat{u}^B_{SPi,3}+h_D,\dots)}{12h_D}-\dots \\ \frac{8\hat{P}_{SPi}(v^B_{SL,3},\dots,\hat{u}^B_{SPi,3}-h_D,\dots)+\hat{P}_{SPi}(v^B_{SL,3},\dots,\hat{u}^B_{SPi,3}-2h_D,\dots)}{12h_D}, \\ \quad \text{für } j=3 \\ \frac{3\hat{P}_{SPi}(v^B_{SL,4},\dots,\hat{u}^B_{SPi,4})-4\hat{P}_{SPi}(v^B_{SL,4},\dots,\hat{u}^B_{SPi,4}-h_D)+\hat{P}_{SPi}(v^B_{SL,4},\dots\hat{u}^B_{SPi,4}-2h_D)}{2h_D}, \\ \quad \text{für } j=4 \end{cases} \tag{23}$$

mit der Schrittweite $h_D = 1 \cdot 10^{-6}$. Sämtliche Terme der Gleichung 21 sind somit bekannt und der Gradient steht der Backpropagation zur Verfügung.

Die Bezeichnungen der Architekturen sowie die Ergebnisse der unterschiedlichen Architekturen zur Schätzung der Spurmarkierungen der Ego-Fahrspur sind in den Tabellen 1 und 2 dargestellt. Zur Referenzierung wird nachfolgend die Schreibweise {Modell, Vortraining, Kostenfunktion} verwendet. Für das Vortraining gelten die Abkürzungen:

VT1 : Feintuning mittels binärer Straßen-Klassifikation,

VT2 : Feintuning mittels binärer Straßen-Klassifikation und anschließendem Training zur Klassifikation der Anzahl an Fahrspuren.

Die letzten beiden Spalten der Tabelle 2 stellen die mittleren Abweichungen vom geschätzten Spline zum korrekten Spline dar. Dazu werden die Splines an $m = 200$ Stützstellen ausgewertet:

Tabelle 1: Bezeichnungen der unterschiedlichen Architekturen zur Schätzung des Ego-Fahrkorridors.

| Modell | Vortraining | Kostenfunktion |
|---|---|---|
| *VGG16*+Extra Trees | VT1 | - |
| *VGG16* | VT1 | $L_2$ |
| *AlexNet* | VT1 | $L_2$ |
| *AlexNet* | VT1 | $L_{Spline}$ |
| *AlexNet*-Multiscale | VT1 | $L_2$ |
| *AlexNet* | VT2 | $L_2$ |

Tabelle 2: Ergebnisse der unterschiedlichen Architekturen zur Schätzung des Ego-Fahrkorridors (Teil 2).

| Modell | $\overline{\lvert \mathbf{x}^B_{Train} - \hat{\mathbf{x}}^B_{Train} \rvert}$ [px] | $\overline{\lvert \mathbf{x}^B_{Test} - \hat{\mathbf{x}}^B_{Test} \rvert}$ [px] |
|---|---|---|
| {*VGG16*+Extra Trees, VT1, - } | 0,00 | 15,92 |
| {*VGG16*, VT1, $L_2$ } | 3,10 | 10,17 |
| {*AlexNet*, VT1, $L_2$} | 2,38 | 10,55 |
| {*AlexNet*, VT1, $L_{Spline}$ } | 8,70 | 11,81 |
| {*AlexNet*-Multiscale, VT1, $L_2$} | 8,06 | 11,47 |
| {*AlexNet*, VT2, $L_2$} | 1,96 | 8,61 |

$$\begin{aligned}
&\overline{\lvert \mathbf{x}^B_{Train/Test} - \hat{\mathbf{x}}^B_{Train/Test} \rvert} = \\
&= \frac{\sum_{j=1}^{m} \lvert P_{SP2}(v^B_{SP2,j}) - \hat{P}_{SP2}(v^B_{SP2,j}) \rvert + \lvert P_{SP3}(v^B_{SP3,j}) - \hat{P}_{SP3}(v^B_{SP3,j}) \rvert}{2m} \\
&= \frac{\sum_{j=1}^{200} \lvert P_{SP2}(v^B_{SP2,j}) - \hat{P}_{SP2}(v^B_{SP2,j}) \rvert + \lvert P_{SP3}(v^B_{SP3,j}) - \hat{P}_{SP3}(v^B_{SP3,j}) \rvert}{400}. \qquad (24)
\end{aligned}$$

Aus Tabelle 2 ist ersichtlich, dass die Regression mittels Deep Learning auf den Testdaten bessere Ergebnisse erzeugt. Wie zuvor diskutiert ist die hypothetische Begründung für dieses Ergebnis, dass die Einbettung der Regression in Deep Learning eine Optimierung der Merkmale durch die Backpropagation ermöglicht. Ein Vergleich der Modelle {*VGG16*, V1, $L_2$} und {*AlexNet*, V1, $L_2$} zeigt, dass das *VGG16* Netz geringe Vorteile gegenüber dem *AlexNet* hat.

Tabelle 3: Mittlere Ausführungsdauern der unterschiedlichen Architekturen zur Schätzung der Spurmarkierungen der Ego-Fahrspur.

| Modell | Inferenz [ms] |
|---|---|
| {*AlexNet*, VT1/VT2, $L_2$} | 5,44 |
| {*VGG16*, VT1, $L_2$} | 25,82 |
| Extra Trees | 2,037 |

Die *AlexNet* Architektur verfügt über eine geringere Anzahl an Parametern und die Ausführung ist schneller.

Zur Erzeugung des Datensatzes werden Bilder aus Videos extrahiert, die mit einer Dashcam erzeugt worden sind. Jedes Video zeigt eine unterbrechungsfreie Fahrt. Nach Extraktion der Bilder werden diese auf eine Größe von $640\,\text{px} \times 360\,\text{px}$ skaliert. Anschließend werden die Bilder so zugeschnitten, dass lediglich der untere Bildausschnitt mit einer Größe $640\,\text{px} \times 160\,\text{px}$ verbleibt.

Einen Vergleich der Zeitdauern für einen Inferenzschritt gibt Tabelle 3. Die geringe Komplexität der *AlexNet* Architektur spiegelt sich in geringeren Ausführungszeiten wider. Für den Echtzeiteinsatz in automatisierten Fahrzeugen sind kurze Ausführungszeiten eine essentielle Voraussetzung. Dies motiviert, dass im Zuge der verbleibenden Arbeit die *AlexNet* Architektur für sämtliche Entwicklungen verwendet wird.

Die Tabelle 2 zeigt zudem, dass die Verwendung der numerischen Spline-Kostenfunktion $L_{Spline}$ keine Verbesserung der Schätzgüte mit sich bringt. Durch die Spline-Approximation und insbesondere die numerische Differentiation entstehen zusätzliche Fehler. Hingegen hat die quadratische Kostenfunktion $L_2$ eine geschlossene mathematische Darstellung für den Gradienten bezüglich der Eingänge. Es kann keine Verbesserung der Güte festgestellt werden, wenn zur Regression Merkmale unterschiedlicher Schichten des *AlexNets* kombiniert werden, dargestellt durch das Modell {*AlexNet*-Multiscale, V1, $L_2$}.

Für das Modell {*AlexNet*, VT2, $L_2$} ist exemplarisch in Bild 5 der Trainingsverlauf dokumentiert. Die Trainingskosten nehmen bis etwa zur 15000. Iteration ab. Die Kosten auf den Testdaten weisen ab der 5000. Iteration keine signifikanten Veränderungen mehr auf.

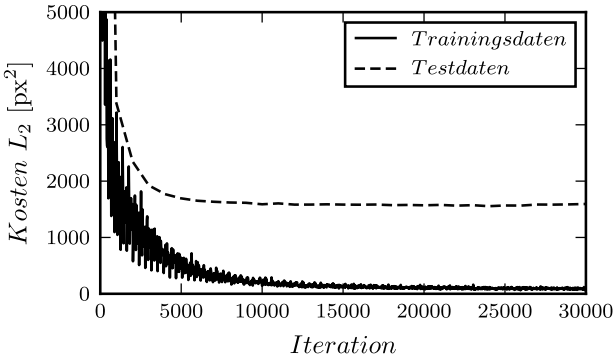
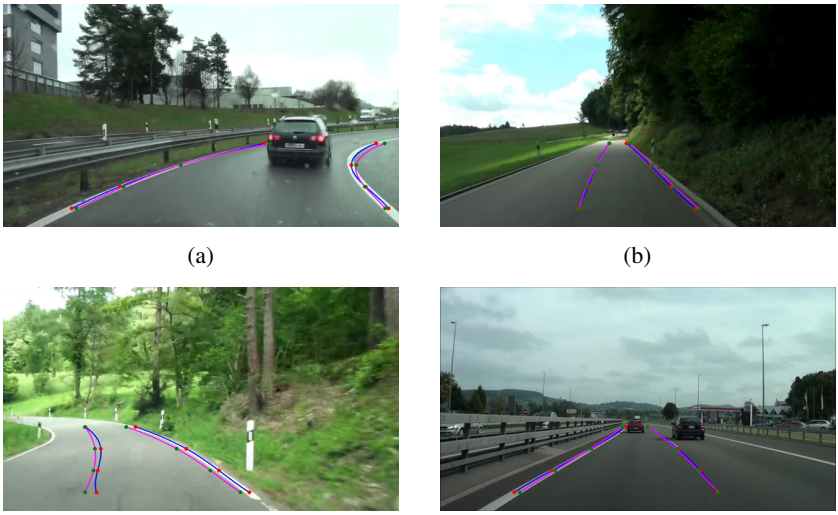



Bild 5: Trainingsverlauf des Modells {*AlexNet*, VT2, $L_2$}.

In Bild 6 sind die manuell annotierten Splinestützstellen als rote Kreise dargestellt. Die blauen Linie ist die Spline Interpolation durch diese Stützstellen. Hingegen stellen die grünen Kreise die vom Modell {*AlexNet*, VT2, $L_2$} ausgegebenen Stützstellen dar. Als violette Linie ist die entsprechende Spline-Interpolation dargestellt. Sowohl in Szenarien mit starken Kurven und bei Fahrten auf der Autobahn liefert die Schätzung gute Ergebnisse. Die besondere Stärke dieses Ansatzes zeigt sich allerdings in den beiden Szenarien, in denen die mittlere Spurmarkierung nicht vorhanden ist. Auch dabei werden gute Ergebnisse erzielt. Modellbasierte Verfahren sind im Vergleich dazu nicht in der Lage die Ego-Fahrspur zu erfassen.

Ein Beispiel einer Schätzung sehr geringer Güte zeigt Bild 7a. Es handelt sich um ein sehr komplexes Szenario, in dem auch ein Abbiegevorgang nach links möglich gewesen wäre. Das Netz scheint einen solchen Vorgang zu präferieren beziehungsweise eher in den Trainingsdaten vorgefunden zu haben. In Bild 7b ist die Fahrt auf einer Landstraße dargestellt. Die Schätzung weist hinsichtlich der linken Spurmarkierung für die ersten beiden Stützstellen größere Abweichungen zur manuellen Annotierung auf. Gleichzeitig ist in diesem Fall fraglich, ob die manuell festgelegten Stützstellen den realen Spurverlauf besser darstellen als die geschätzten.

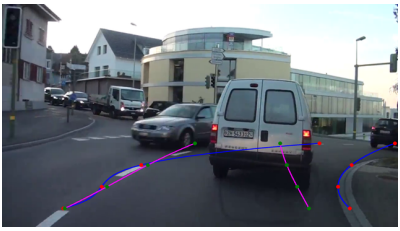


(a) (b)

(c) (d)

Bild 6: Beispiele von Schätzungen der Ego-Fahrspur mit hoher Güte mittels des Modells {*AlexNet*, VT2, $L_2$}.

Ein häufig auftretendes Problem im Kontext von Deep Learning besteht darin, dass selten vollständig nachvollziehbar ist, auf welcher Basis Entscheidungen getroffen werden. In der Automobilindustrie müssen Algorithmen zunächst eine Vielzahl von Sicherheitsüberprüfungen absolvieren, um schließlich für Serienfahrzeuge eine Zulassung zu erhalten. Ein wichtiger Schritt wurde in [15] unternommen, um Convolutional Neural Networks transparenter zu machen. Hier wird eine Architektur bestehend aus transponierten Faltungen und Unpooling Schichten vorgeschlagen. Das Unpooling ist eine approximative Umkehrung der Pooling Operation. Durch Einsatz der Architektur ist es möglich zu visualisieren, welche Anteile des Eingangsbildes zu den stärksten Aktivierungen auf einer Merkmalsschicht führen. Beispielhaft ist dies für das Modell {*VGG16*, VT1, $L_2$} in den Bildern 8, 9 und 10 dargestellt. Dazu werden die 50 stärksten Aktivierungen auf der Pooling 5 Schicht (vergleiche Bild 13) beibehalten und der Rest der Schicht mit Nullen belegt. Diese Aktivierungen propagieren durch eine alternierende Folge von Unpooling und Schichten transponierter Faltungen und rekonstruieren dadurch die Bereiche im Eingangsbild, die zu den stärksten Aktivierungen geführt haben.



(a) Urbane Kurvenfahrt.

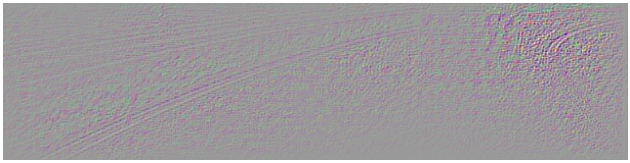
(b) Fahrt auf Landstraße.

Bild 7: (a) Beispiel einer Schätzung geringer Güte und (b) einer akzeptablen Schätzung mittels des Modells {*AlexNet*, VT2, $L_2$}.

In sämtlichen Bildern sind Spurmarkierungen deutlich zu erkennen. Über den gesamten Testdatensatz ist auffällig, dass meist nur eine Spurmarkierung zu starken Aktivierungen führt. Dies deutet darauf hin, dass in einer Vielzahl von Fällen die Kenntnis einer Spur auch den Verlauf der jeweils anderen Spur festlegt. Mutmaßlich berücksichtigt das Netz, ähnlich wie der Mensch, die Parallelität der Markierungen sowie die Perspektive.

Ein weiterer wichtiger Aspekt der Analyse besteht in der Auswertung der Datenverteilungen. Nach [16] besteht das Ziel zur Lösung der vorliegenden Regressionsaufgabe mittels Maschinellen Lernens darin eine Funktion $\mathbf{y}(\mathbf{x})$ zu finden. Aufgabe dieser Funktion ist es, die Eingangsdaten $\mathbf{x}$ in einen Ausgangsvektor $\mathbf{y}$ zu überführen, der über dieselbe Repräsentation wie der Zielvektor verfügt. Die genaue Form der Funktion $\mathbf{y}(\mathbf{x})$ wird während des Trainings auf Grundlage der Trainingsdaten festgelegt. Algorithmen des Maschinellen Lernens basieren auf der Erfassung statistischer Zusammenhänge innerhalb von Datensätzen. Zum Zweck einer genaueren Analyse der Statistik der Daten sind in Bild 11 Histogramme und approximierte Normalverteilungen für die $u^B$-Koordinate der Stützstelle $\mathbf{x}^B_{SP3,2}$ dargestellt. In Bild 11a ist das Histogramm der Pixelwerte der Koordinate über die gesamten Trainingsdaten dargestellt.

(a) Eingangsbild.



(b) Rekonstruktion mittels der 50 stärken Aktivierungen auf der Pooling 5 Schicht.

Bild 8: Beispiel 1 zur Rekonstruktion der Aktivierungen des Modells {*VGG16*, VT1, $L_2$} nach [15].

Das rechte Bild 11b zeigt dies analog für die Testdaten. Die Bilder 11c und 11d zeigen die Histogramme der geschätzten Koordinaten, links für den Trainings- und rechts für den Testdatensatz. Ein Vergleich der ersten und zweiten Zeile des Bildes verdeutlicht, dass der Schätzalgorithmus die Statistik der Daten gut erfasst und modelliert. Noch deutlicher wird dies durch die Approximation der Verteilungen jeweils durch Normalverteilungen gemäß:

$$\mathcal{N}(u^B_{SPi,j}|\mu_{SPi,j},\sigma^2_{SPi,j}) = \frac{1}{\sqrt{(2\pi\sigma^2_{SPi,j})}} e^{-\frac{(u^B_{SPi,j}-\mu_{SPi,j})}{2\sigma^2_{SPi,j}}} \tag{25}$$

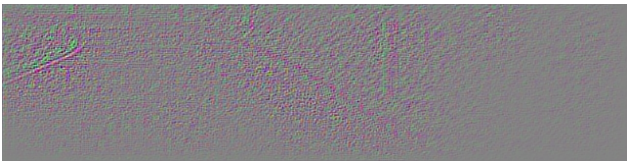
für die $u^B$-Koordinate der Stützstelle $\mathbf{x}^B_{SPi,j}$. Dies zeigen die Bilder 11e und 11f für die Trainings- respektive Testdaten. Die Mittelwerte $\mu_{SPi,j}$ weisen in beiden Fällen eine gute Übereinstimmung auf. Die Varianz der Schätzung ist auf den Testdaten geringer als die Varianz der tatsächlichen Stützstellenkoordinaten. Ein quantitativer Vergleich könnte mittels der Kullback-Leibler Divergenz nach [17] erfolgen.

(a) Eingangsbild.

(b) Rekonstruktion mittels der 50 stärken Aktivierungen auf der Pooling 5 Schicht.

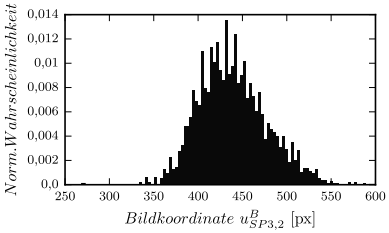
Bild 9: Beispiel 2 zur Rekonstruktion der Aktivierungen des Modells {*VGG16*, VT1, $L_2$} nach [15].

(a) Eingangsbild.

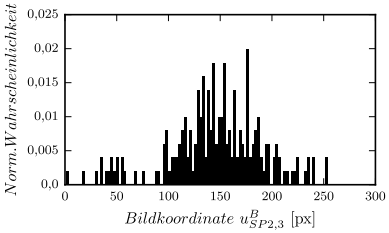


(b) Rekonstruktion mittels der 50 stärken Aktivierungen auf der Pooling 5 Schicht.

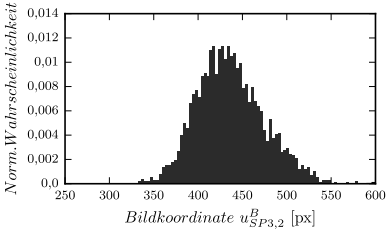
Bild 10: Beispiel 3 zur Rekonstruktion der Aktivierungen des Modells {*VGG16*, VT1, $L_2$} nach [15].



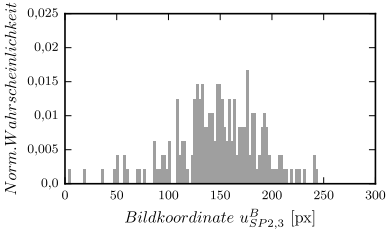

(a) Verteilung der Daten des Trainingsdatensatzes.



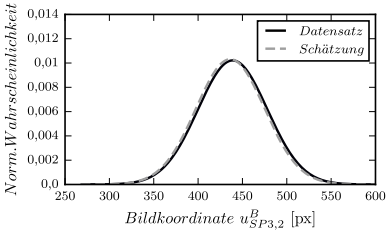

(b) Verteilung der Daten des Testdatensatzes.



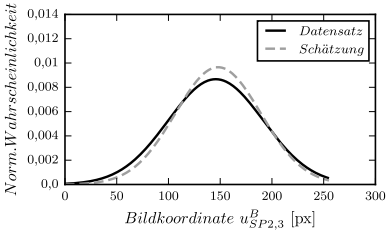

(c) Verteilung der Daten der Schätzung.




(d) Verteilung der Daten der Schätzung.




(e) Normalverteilung auf Basis der geschätzten Daten.




(f) Normalverteilung auf Basis der geschätzten Daten.

Bild 11: Histogramme der Verteilungen der $u^B$-Koordinaten von Stützstelle $\mathbf{x}^B_{SP3,2}$ (links) sowie $\mathbf{x}^B_{SP2,3}$ (rechts) und Approximation durch Normalverteilungen.

# 3 Gesamtarchitektur zur Schätzung von Fahrspuren

Das begrenzte Sichtfeld der Kamera, mit dem der Datensatz erzeugt wurde, ermöglicht eine zuverlässige Schätzung von maximal vier Spurmarkierungen. Auf die Schätzung der Spurmarkierungen links und rechts der Ego-Fahrspur wird in diesem Abschnitt eingegangen. Der Nomenklatur aus Bild 3 folgend werden die Spurmarkierungen als Links-Links und Rechts-Rechts bezeichnet.

Der Schätzvorgang erfolgt sequentiell und basiert auf den Informationen über die Anzahl der Fahrspuren und der vom Ego-Fahrzeug befahrenen Spur. Dadurch ist eindeutig festgelegt, ob die Spurmarkierungen Links-Links und Rechts-Rechts vorhanden sind und somit eine Schätzung dieser erfolgt. Die Gesamtarchitektur ist in Bild 12 dargestellt. Das Eingangsbild wird den beiden Klassifikatoren und die Schätzergebnisse werden einem Umschaltmechanismus zugeführt. Wenn beispielsweise drei Fahrspuren klassifiziert werden und das Ego-Fahrzeug davon die zweite Spur befährt, so erfolgt die Schätzung beider Spurmarkierungen Links-Links und Rechts-Rechts. Existieren nur zwei Spuren und bewegt sich das Fahrzeug auf der ersten der beiden Spuren so ist lediglich die Schätzung der Spurmarkierung Rechts-Rechts möglich. Im unteren Bereich des Bildes ist schließlich die in Abschnitt 2 vorgestellte Schätzung der Spurmarkierungen der Ego-Fahrspur beschrieben. Die Schätzung von Spurmarkierungen erfolgt lediglich auf dem untere Ausschnitt des Eingangsbildes mit der Dimension $640\,\mathrm{px} \times 160\,\mathrm{px}$. Die Schätzergebnisse liegen in der Form von Spline-Stützstellen vor. Sie werden dazu verwendet, die Spurmarkierungen als Splines im Ausgangsbild zu visualisieren. Auch erlaubt die parametrische Form der Splines eine einfache Transformation in das Fahrzeugkoordinatensystem durchzuführen.

Zusammenfassend zeigt Bild 13 sämtliche Architekturen zur Schätzung von Spurmarkierungen, die in diesem Beitrag untersucht respektive genutzt werden.

Bereich 1 zeigt die Nutzung von Merkmalen unterschiedlicher Schichten. Im Bereich 2 ist die Regression mittels Extra Trees dargestellt. Hierbei fungiert das Convolutional Neural Network lediglich als Merkmalsextraktor. Die Bereiche 3 und 4 zeigen die Modelle zur Schätzung von Spurmarkierungen respektive

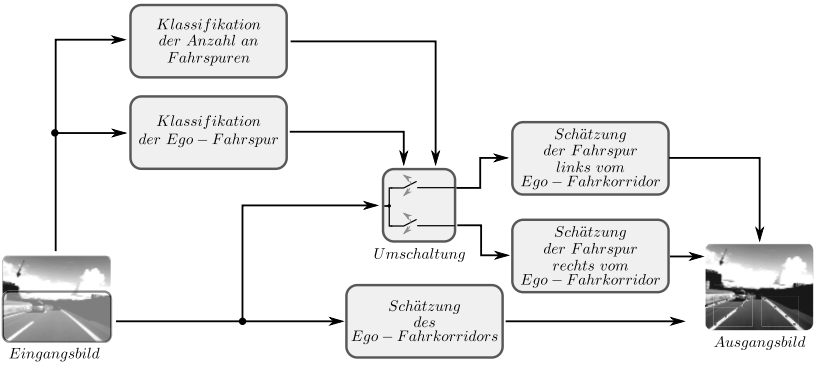



Bild 12: Vorgeschlagene Architektur zur Schätzung von Spurmarkierungen.

zur Lösung der Klassifikationsaufgaben zur lateralen, spurdiskreten Lokalisierung. Beide werden innerhalb dieses Beitrages nicht behandelt.

Das Training zur Schätzung der Spurmarkierungen Rechts-Rechts und Links-Links erfolgt im Sinne eines Feintunings der Architektur zur Schätzung der Spurmarkierungen der Ego-Fahrspur. Dadurch sind Datensätze verminderten Umfangs erforderlich. Für beide Modelle wird ein Datensatz erzeugt, der insgesamt 800 manuell annotierte Bilder umfasst. Die Trainingsverläufe auf diesen Datensätzen sind in den Bildern 14a und 14b dargestellt. Bereits nach circa 1000 Iterationen kommt es in beiden Fällen zur Konvergenz und die Kosten auf den Testdaten verringern sich nachfolgend nur noch sehr langsam.

Die Bezeichnungen, Iterationen sowie die Ergebnisse der jeweils besten Modelle zeigen die Tabellen 4 und 5. In den letzten beiden Spalten sind die Güten der Schätzungen gemäß:

$$\begin{aligned}
&\overline{|\mathbf{x}^B_{Train/Test} - \hat{\mathbf{x}}^B_{Train/Test}|} = \\
&= \frac{\sum_{i=1}^{m} |P_{SP1/SP4}(v^B_{SP1/SP4,i}) - \hat{P}_{SP1/SP4}(v^B_{SP1/SP4,i})|}{m} \\
&= \frac{\sum_{i=1}^{200} |P_{SP1/SP4}(v^B_{SP1/SP4,i}) - \hat{P}_{SP1/SP4}(v^B_{SP1/SP4,i})|}{200}
\end{aligned} \tag{26}$$

dargestellt.

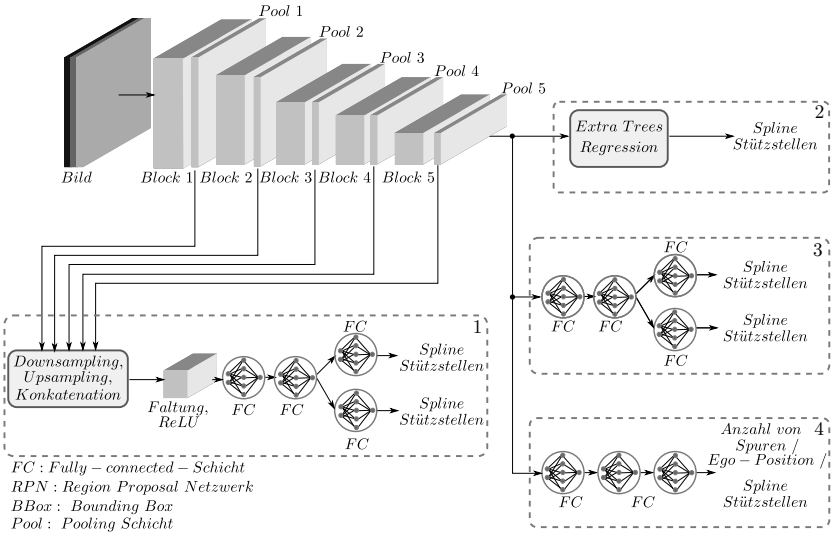


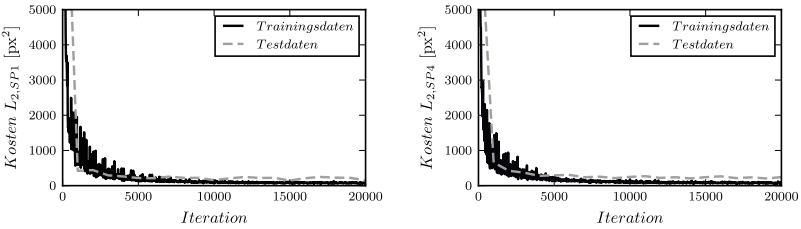

Bild 13: Gesamtarchitektur

Tabelle 4: Bezeichnungen und Iterationen der Netze zur Schätzung der Spurmarkierungen Links-Links und Rechts-Rechts.

| Architektur | Spurmarkierung | Iteration |
|---|---|---|
| *AlexNet* | Links-Links | 15000 |
| *AlexNet* | Rechts-Rechts | 18000 |

Die Berechnung der Güte erfolgt durch Auswertung der Splines $P_{SP1}(v^B_{SP1,i})$ und $P_{SP4}(v^B_{SP4,i})$ für die Markierung Links-Links respektive Rechts-Rechts an je $m = 200$ Stellen.

Dass beide Modelle echtzeitfähig sind, belegt Tabelle 6. Als Hardwareplattform für sämtliche Untersuchungen kommt eine NVIDIA GeForce GTX1060 GPU zum Einsatz.




(a) Spurmarkierung Links-Links. (b) Spurmarkierung Rechts-Rechts.

Bild 14: Trainingsverläufe der Modelle zur Schätzung von linker respektive rechter Spur in Nachbarspuren.

Tabelle 5: Ergebnisse der Netze zur Schätzung der Spurmarkierungen Links-Links und Rechts-Rechts (Teil 2).

| Modell | $\overline{\lvert \mathbf{x}^B_{Train} - \hat{\mathbf{x}}^B_{Train} \rvert}$ [px] | $\overline{\lvert \mathbf{x}^B_{Test} - \hat{\mathbf{x}}^B_{Test} \rvert}$ [px] |
|---|---|---|
| {*AlexNet*, Links-Links} | 1,88 | 4,15 |
| {*AlexNet*, Rechts-Rechts} | 1,78 | 4,37 |

Zur qualitativen Bewertung der Schätzungen dienen die Bilder 15 und 16. Darin sind die manuell annotierten Splinestützstellen als rote Kreise dargestellt. Die blaue Linie steht für die Splineinterpolation durch diese Stützstellen. Hingegen stellen die grünen Kreise die, vom Modell, geschätzten Stützstellen dar. In violett ist zudem die entsprechende Splineinterpolation visualisiert.

Die Spurmarkierung Links-Links wird über den gesamten Testdatensatz konsistent mit sehr hoher Güte geschätzt. Exemplarisch zeigt Bild 15b eine Schätzung, bei der größere Fehler auftreten. Im Bild ist deutlich ein Streifen Teer zu erkennen, wie er durch Reparaturmaßnahmen häufig entsteht. Fälschlicherweise werden die Stützstellen genau entlang des Streifens platziert. Derartige Schätzergebnisse könnten ausgeschlossen werden, indem ein wesentlich umfassender Datensatz zum Training eingesetzt wird. Dadurch ist mutmaßlich eine Erhöhung der Generalisierungsfähigkeit des Netzes möglich.

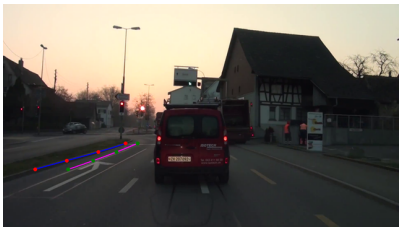
Tabelle 6: Zeitdauer von Inferenz und Backpropagation der Netze zur Schätzung der Spurmarkierungen Links-Links und Rechts-Rechts.

| Architektur | Spurmarkierung | Inferenz [ms] |
|---|---|---|
| *AlexNet* | Links-Links | 5,24 |
| *AlexNet* | Rechts-Rechts | 5,23 |

Für die Spurmarkierung Rechts-Rechts sind Schätzergebnisse in Bild 16 dargestellt. Die Stärke des Deep Learning Ansatzes zeigt sich darin, dass selbst bei Verdeckung der Markierungen durch andere Fahrzeuge dennoch eine zuverlässige Schätzung vollzogen wird. Einen Grenzfall stellt Bild 16b dar. Die vom Netz vorgeschlagenen Stützstellen weichen von den manuell annotierten Stützstellen ab. Das Szenario ist allerdings auch für den Menschen schwierig zu interpretieren und es ist nicht zweifelsfrei sichergestellt, dass die Stützstellen bei Erzeugung des Datensatzes richtig platziert worden sind.

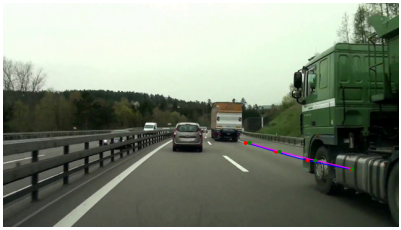
Zum Zweck einer genaueren Analyse der Statistik der Daten zeigt Bild 17 Histogramme und approximierte Normalverteilungen für die $u^B$-Koordinate der Stützstelle $\mathbf{x}^B_{SP1,2}$. Analog zu den Darstellungen im Abschnitt 2 ist in Bild 17a das Histogramm der Pixelwerte der Koordinate über die gesamten Trainingsdaten dargestellt. Das rechte Bild 17b zeigt dies für die Testdaten. Die Bilder 17c und 17d zeigen die Histogramme der geschätzten Koordinaten, links für den Trainings- und rechts für den Testdatensatz. Ein Vergleich der ersten und zweiten Zeile des Bildes verdeutlicht, dass der Schätzalgorithmus die Statistik der Daten auch in diesem Fall gut erfasst und modelliert. Noch deutlicher wird dies durch die Approximation der Verteilungen jeweils durch Normalverteilungen. Dies zeigen die Bilder 17e und 17f für die Trainings- respektive Testdaten. Die Mittelwerte und Varianzen der Stützstellen des Datensatzes und der Schätzungen weisen in beiden Fällen eine gute Übereinstimmung auf. Dies gilt in analoger Weise für die Schätzung der Spurmarkierung Rechts-Rechts.

(a)

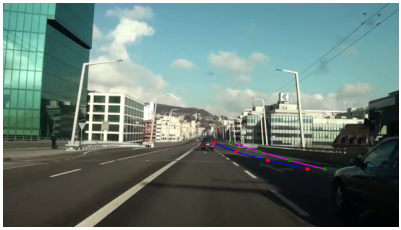


(b)

Bild 15: Exemplarische Schätzergebnisse hoher Güte für die Spurmarkierung Links-Links.

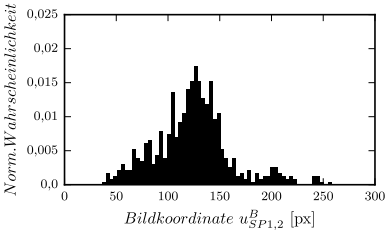


(a)

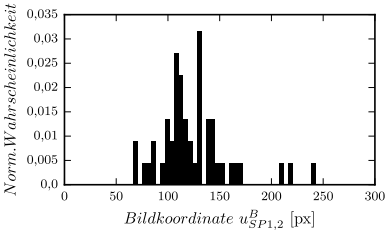


(b)

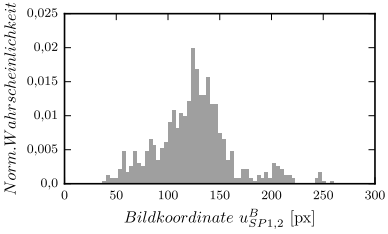
Bild 16: Exemplarische Schätzergebnisse für die Spurmarkierung Rechts-Rechts. Schätzung (a) zeigt, dass die Schätzung auch bei Verdeckung der zu schätzenden Spurmarkierungen gute Ergebnisse liefert. Hingegen zeigt Abbildung (b) eine fehlerhafte Schätzung.



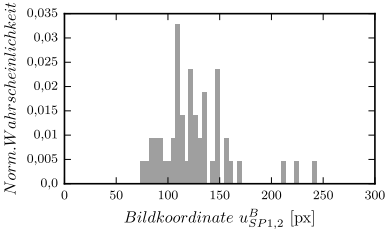

(a) Verteilung der Daten des Trainingsdatensatzes.



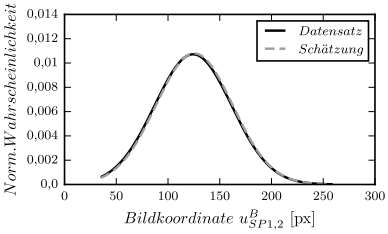

(b) Verteilung der Daten des Testdatensatzes.



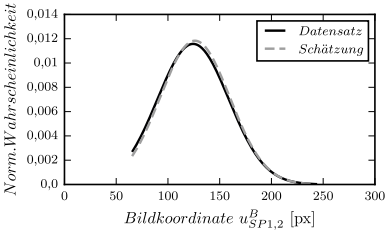

(c) Verteilung der Daten der Schätzung.




(d) Verteilung der Daten der Schätzung.




(e) Normalverteilung auf Basis der geschätzten Daten.




(f) Normalverteilung auf Basis der geschätzten Daten.

Bild 17: Histogramme der Verteilungen der $u^B$-Koordinaten von Stützstelle $\mathbf{x}^B_{SP1,2}$ auf Trainingsdaten (links) sowie Testdaten (rechts) und Approximation durch Normalverteilungen.

# 4 Zusammenfassung

Der vorliegende Beitrag beschäftigt sich mit der Schätzung von Spurmarkierungen zur Nutzung in automatisierten Fahrfunktionen. Hierzu wird ein Convolutional Neural Network entwickelt, welches als Ausgabe Spline-Sütztstellen liefert. Der Trainingsvorgang wird analysiert und die Ergebnisse evaluiert. Zunächst erfolgen die Schätzungen lediglich für die Spurmarkierungen der Ego-Fahrspur. Anschließend wird das Konzept durch Einbringung von Klassifikatoren zur Schätzung der lateralen, spurdiskreten Lokalisierung erweitert. Auf Basis dieser Informationen ist eine Ausdehnung des Konzeptes auf die Schätzung von maximal vier Spurmarkierungen möglich. Die Ergebnisse zeigen, dass es sich um einen vielversprechenden Ansatz handelt, der insbesondere Stärken in urbanen Szenarien hat. Hier fehlt häufig die mittlere Spurmarkierung. Das neuronale Netz liefert dennoch auch in solchen Szenarien gute Ergebnisse und ist in der Lage Fahrspuren korrekt zu schätzen. Zukünftige Arbeiten könnten sich mit der Kombination des vorgeschlagenen Ansatzes mit klassischen Verfahren zur Spurmarkierungsdetektion beschäftigen. Auch die Verknüpfung mit einem Segmentierungsalgorithmus erscheint sehr vielversprechend. So könnten die Stärken der unterschiedlichen Ansätze kombiniert werden um die Robustheit weiter zu erhöhen.

# Literatur


[1] M. Aly, 'Real time Detection of Lane Markers in Urban Streets', *IEEE Intelligent Vehicles Symposium*, pp. 7-12, 2008.

[2] W. Wicki, 'Entwicklungspsychologie', Lehrbuch, UTB GmbH, 2015.

[3] I. Goodfellow et al., 'Deep Learning', Lehrbuch, The MIT Press, 2016.

[4] J. Li, X. Mei, D. Prokhorov und D. Tao, 'Deep Neural Network for Structural Prediction and Lane Detection in Traffic Scene', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 690-703, 2017.

[5] Y. LeCun et al., 'Backpropagation Applied to Handwritten Zip Code Recognition', *Neural Computation*, vol. 1, no. 4, pp. 541-551, 1989.

[6] J. Kim und M. Lee, 'Robust Lane Detection Based On Convolutional Neural Network and Random Sample Consensus', *International Conference on Neural Information Processing*, pp. 454-461, 2014.

[7] M. Fischler und R. Bolles, 'Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography', *Communications of the ACM*, vol. 24, pp. 381-395, 1981.

[8] V. John et al., 'Real-Time Lane Estimation Using Deep Features and Extra Trees Regression', *Image and Video Technology: 7th Pacific-Rim Symposium*, pp. 721-733, Springer International Publishing, 2016.

[9] A. Krizhevsky, I. Sutskever und G. Hinton, 'Imagenet classification with deep convolutional neural networks', *Advances in Neural Information Processing Systems*, vol. 1, pp. 1097-1105, 2012.

[10] B. Zhou et al., 'Places: An Image Database for Deep Scene Understanding', *ArXiv e-prints*, CoRR abs/1610.02055, 2016.

[11] P. Geurts, D. Ernst und L. Wehenkel, 'Extremely Randomized Trees', *Journal of Machine Learning*, vol. 16, no. 1, pp. 3-42, 2006.

[12] R. Mohr, 'Numerische Methoden in der Technik: Ein Lehrbuch mit MATLAB-Routinen', Lehrbuch, Vieweg+Teubner Verlag, 2013.

[13] F. Pedregosa et al., 'Scikit-learn: Machine Learning in Python', *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[14] K. Simonyan und A. Zisserman, 'Very Deep Convolutional Networks for Large-Scale Image Recognition', *ArXiv e-prints*, CoRR abs/1409.1556, 2014.

[15] M. Zeiler und R. Fergus, 'Visualizing and Understanding Convolutional Networks', *European Conference on Computer Vision ECCV 2014*, pp. 818-833, 2014.

[16] C. Bishop, 'Pattern Recognition and Machine Learning (Information Science and Statistics)', Lehrbuch, Springer-Verlag New York, Inc., 2006.

[17] S. Kullback und R. Leibler, 'On information and sufficiency', *Annals of Mathematical Statistics*, vol. 22, no.1, pp. 79-86, 1951.

# Best Practices in Deep Learning-Based Segmentation of Microscopy Images

Tim Scherr[1], Andreas Bartschat[1], Markus Reischl[1],
Johannes Stegmaier[2], Ralf Mikut[1]

[1] Institute for Automation and Applied Informatics,
Karlsruhe Institute of Technology, Karlsruhe, Germany
[2] Institute of Imaging and Computer Vision,
RWTH Aachen University, Aachen, Germany
E-Mail: tim.scherr@kit.edu

## 1 Introduction

Deep neural networks are state-of-the-art methods in image classification [1, 2, 3, 4, 5], single-object localization [1, 2, 3], object detection [2, 4, 6], semantic segmentation [7, 8], combined object detection and instance segmentation [6], and segmentation of 2D/3D biological and medical microscopy images [9, 10, 11]. Common convolutional neural networks consist of fully-connected layers, convolutional layers with fewer, shared weights operating locally, and pooling layers for downsampling [12]. Challenges in image segmentation are, for instance, inherent variation present within and among different data sets, class imbalance [13], a lack of task-specific training data [14], imperfect segmentation labels [15], and clustered and overlapping objects.

Image segmentation challenges have shown that, besides an adapted network architecture, further improvements such as task-specific data augmentation, customized loss functions, and specialized post-processing is needed, e.g., [16]. For the design of a deep learning-based segmentation, the developer has to choose the network architecture and the training process settings [17], including regularization [18], activation and loss functions, gradient descend optimization algorithms [19, 20, 21], batch normalization [22], and the corresponding hyperparameters. In contrast to shallow networks, deeper networks are able to

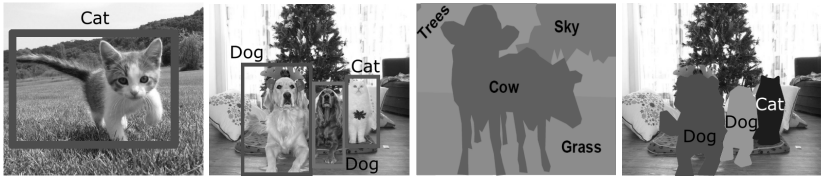
use far fewer parameters per layer to fit the training set and often generalize to the test set, but are also harder to optimize [17]. Task-specific ideal network architectures must be found experimentally, guided by the validation set error [17].

Despite the great success of deep learning in image segmentation tasks, there are only a few software tools for non-specialists, e.g., CellProfiler 3.0 [23]. This is due to the many possible combinations of network architectures, demands on the segmentation (binary, semantic, instance), fields of applications (e.g., biology, medicine), data formats (e.g., gray scale, RGB, 2D, 3D), and training process settings.

In this contribution we show how to start segmenting 2D microscopy images using a selected deep learning framework and architecture. We give recommendations to support developers in the selection of the architecture and training process settings, and show how the segmentation can be improved on the basis of the already finished Kaggle 2018 Data Science Bowl segmentation challenge [16]. This contribution is limited to 2D data, but the outcome of 2D data is useful for 3D or 3D+t data as well.

## 2 Network Architectures for 2D Image Segmentation

Neural networks for image segmentation are often inspired by image classification and object detection networks. Figure 1 shows the tasks of image classification (a), object localization (a), object detection (b), and of the image segmentation subclasses semantic (c) and instance (d) segmentation. In image classification, an image is assigned to one single class. The image should ideally contain only one object. Localization predicts a bounding box of the object. If there are multiple objects, the task is called object detection. Semantic segmentation partitions an image pixel-wise. Touching objects of the same class cannot be distinguished. In contrast, in instance segmentation different instances of the same class have separate labels and touching objects should be distinguished. Usually, two-stage networks with object proposals are used for instance segmentation. Common to all tasks is the requirement to find class-specific features.




(a) Classification w. Localization (b) Object Detection (c) Semantic Segmentation (d) Instance Segmentation

Figure 1: Tasks of classification with localization (a), object detection (b), and semantic (c) and instance (d) segmentation. Modified from [24].

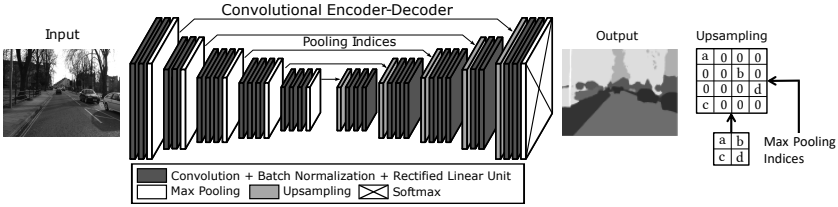



Figure 2: SegNet is a fully convolutional network for semantic segmentation [8]. The pooling indices (right) are used for the upsampling. Modified from [8].

In the following, three popular state-of-the-art architectures for image segmentation are shortly described:

**SegNet** is an encoder-decoder architecture commonly used for semantic segmentation [8]. Figure 2 shows that the encoder is topologically equivalent to the convolutional layers in the VGG16 network for image classification [1]. A novelty was the decoder which upsamples the low resolution input feature maps to full input resolution feature maps using pooling indices computed in the pooling step of the corresponding encoder [8]. Thus, there is no need for learning to upsample. SegNet uses the pre-trained weights of the VGG16 part.

**U-Net** is an encoder-decoder network that was initially developed for biological and medical image data [9]. In contrast to SegNet, corresponding encoder and decoder feature maps are concatenated. This allows successive convolutional layers to assemble a more precise output than without [9]. Figure 3 shows a slightly modified U-Net with batch normalization to fix the means and the variances of the layer inputs [22], and learnable transposed convolutions for the upsampling. If zero padding is used in the convolutional layers, there is

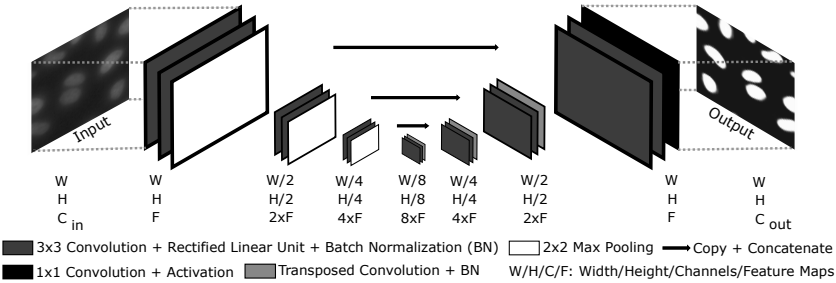



Figure 3: U-Net architecture for image segmentation. The shown network is named 3-block U-Net, as three pooling layers are used in total. Modified from [25].

no need for cropping the feature maps before concatenating as in the original architecture.

**Mask R-CNN** combines object detection and segmentation enabling instance segmentation [6]. It is a two-stage architecture. In contrast to ResNet und U-Net, the architecture is more complex due to its different branches: a branch for predicting segmentation masks on each region of interest in parallel with a branch for classification and bounding box regression. Recently, Mask R-CNN has also been used for the instance segmentation of nuclei in biological microscopy images [26].

# 3 Getting Started with Microscopy Image Segmentation

This section covers basic information for the segmentation of roundish objects in microscopy images using Python and a deep learning framework. Deep learning frameworks provide efficient implementations and GPU support for highly parallelized computations. Popular frameworks are PyTorch and TensorFlow. The high-level API Keras is capable to run on top of TensorFlow and is designed for easy and fast prototyping. This allows easy realization of concepts and enables a fast implementation, e.g., for challenges [27]. Thus, we recommend to start with a high-level API.

Table 1: Freely available training data sets for microscopy image segmentation.

| **Data Set** | **Description** |
|---|---|
| Broad Bioimage Benchmark Collection (BBBC)[1] | Biological image data sets with various labels (counts, outlines, masks). |
| ISBI Cell Tracking Challenge[2] | 2D and 3D data sets covering a wide range of biological cell types and image quality. |
| Masaryk University Cell Image Collection[3] | 3D synthetic benchmark data sets generated using a virtual microscope. |
| Benchmarks for Embryomics [28] | Semi-synthetic benchmark generator. |

[1] `https://data.broadinstitute.org/bbbc/image_sets.html`
[2] `http://www.celltrackingchallenge.net/datasets.html`.
[3] `https://cbia.fi.muni.cz/datasets/`

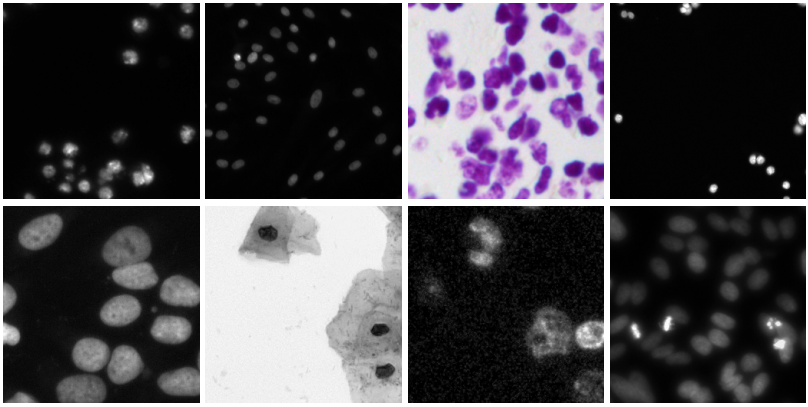


Figure 4: BBBC038v1 data set [29]. The shown images are cropped to $256 \times 256$ px.

## 3.1 Data Sets for Microscopy Image Segmentation

In general, larger training data sets tend to prevent the network from overfitting, but labeling images is time-consuming. Common approaches to enlarge the training set are the use of additional free data sets, e.g., those listed in Table 1, and of data augmentation (data generation using artificially transformed versions of the original data) [17]. Using 3D data slice by slice can also be useful to train a network for 2D segmentation.

In the following, the BBBC038v1 image set is used, which was part of the Kaggle 2018 Data Science Bowl segmentation challenge [29, 16]. The diversity of cell types and density in combination with a low abundance of some cell types, heavy varying image resolutions, and the rather small size of the data set, are challenges of this data set. Figure 4 shows some exemplary images.

## 3.2 Network Architecture

We start with a U-Net as shown in Figure 3. It is easy to implement, trainable from scratch, and expandable, e.g., using stacking and residual connections [30] or specialized encoders [31]. In [32], a U-Net model trained on only two images outperformed an advanced CellProfiler pipeline.

The strides for the convolutional, transposed convolutional and max pooling layers are set to 1, 2 and 2 respectively. Additionally, zero padding is used in these layers. The input image size of the network is fixed to $256 \times 256$ px since this is the smallest image size in the used data set. Another possibility is the use of zero padding for non-supported image sizes (due to the pooling/upsampling) and a non-fixed input image size. Using a 4-block U-Net allows the training on an Nvidia Quadro P4000 GPU using F=64 feature maps in the first layer as mentioned in [9].

## 3.3 Loss Functions

Choosing the loss function is important for an accurate segmentation, especially for unbalanced data sets adapted weights or loss functions are needed. Using an inadequate loss function for those data sets can result in a high false positive or high false negative rate, e.g., in background prediction for all pixels if the training data contains almost only background. Table 2 gives an overview of various loss functions for microscopy image segmentation.

## 3.4 Training Process Settings

A common approach is to divide the data set into a training, a validation, and a test set [17]. The training set is used to learn the parameters. The validation

Table 2: Loss functions for microscopy image segmentation.

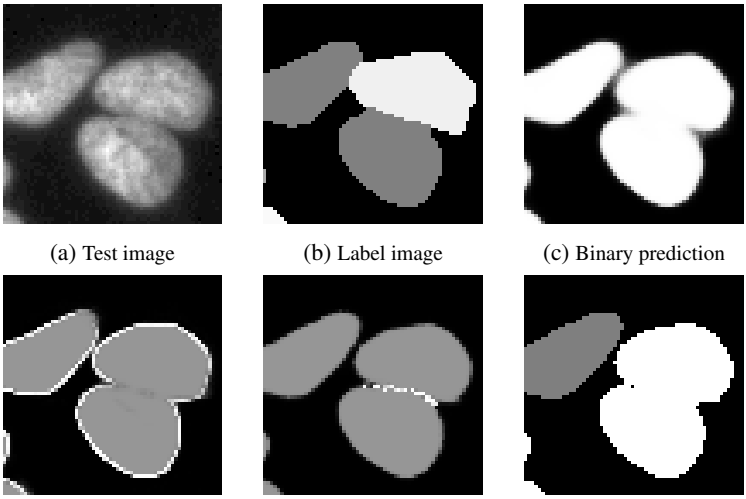
| Loss function | Description |
|---|---|
| Binary/categorical cross entropy (`Bce`/`Cce`) | Measure for dissimilarity between prediction and label. Predictions may be a bit fuzzy. |
| Weighted `Bce`/`Cce` | Using weights to deal with class imbalance. |
| `Dice`/F1 loss [33] | Harmonic mean of precision and recall. No fuzzy predictions but problem of probabilities close to 0 or 1 even for wrong pixels [31]. |
| Generalized `Dice` loss [13] | Generalized `Dice` loss for unbalanced data using weights based on the label area to reduce the correlation between region size and Dice score. |
| `BceDice`/`CceDice` loss | Idea: overcome the limitations of pure `Bce`/`Cce` or `Dice` loss [31]. Also a weighted sum can be used. |

set is used to estimate the generalization error during the training and to guide the selection of the hyperparameters. The test set held back during training can then be used to estimate the generalization error after training. Since there are no labels for the provided BBBC038v1 test set, the last 120 of the 670 training images are used as test set. Before a training process starts, 20% of the remaining 550 training images are selected randomly as validation set. Small unnatural holes in the label images of the training, the validation and the test set are filled using a morphological closing.

We use the adaptive Adam optimizer [21] in the AMSGrad [34] variant (parameters: learning rate $lr = 1 \times 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.99$, $decay = 0$) and a batch size of 8. Without AMSGrad, a higher learning rate is needed. The use of callbacks enables to stop the training process after a fixed number of epochs without validation loss decrease and to save model checkpoints and intermediate results, e.g., loss and validation loss.

## 3.5 Data Pre-Processing

Images larger than the network input size of $256 \times 256$ px are cropped into subimages. After the training an overlap between subimages enables the combination of the predictions without boundary effects. Subimages with less than two objects are excluded from the training process to avoid a bias towards false



(a) Test image (b) Label image (c) Binary prediction

(d) Boundary method (e) Border method (f) Post-processed

Figure 5: First results using the 4-block U-Net. The predictions (c)-(e) of the test image show merged objects. The misclassification of two pixels in the border (white) in (e) causes the merging of the objects (gray). Shown are $64 \times 64$ px crops. The label and the post-processed image are color-coded.

negatives. Every single 8-bit color image $\mathbf{A}$ is converted to single precision and normalized according to:

$$\mathbf{A}' = \frac{\mathbf{A}}{127.5} - 1 \,. \tag{1}$$

## 3.6 First Results

Figure 5c shows first results using the 4-block U-Net and one output channel. If the data analysis demands no instance segmentation, such an approach may be fine. The sum of the `Bce` and the `Dice` loss was used (`BceDice`) and the sigmoid activation function. In tests, no clear trend towards better predictions than with pure `Bce` or `Dice` loss is visible. However, sometimes small improvements on single objects are possible. To get the final binary segmentation a simple thresholding post-processing can be applied. Selecting a fixed threshold on an exemplary prediction is fine most of the time.

If the data analysis demands instance segmentation, weight maps can be used to force the network to learn small separation borders introduced between touching objects as in [9]. Problems occur if a predicted separation border is not perfect.

Another idea is to generate boundaries from the training labels (cf. [32]), and train a network with three one-hot-encoded output classes: background, interior, boundaries. Challenges arise from missing and non-closed boundaries between the touching objects as in Figure 5d. This results in merged objects after post-processing. The reason is that a nearly closed border is quite a good result for this class. As loss function, the sum of the `Cce` loss for every class and of the channel-wise `Dice` losses for the object class and the boundary class was used (`CceDice`). The activation function was the softmax function.

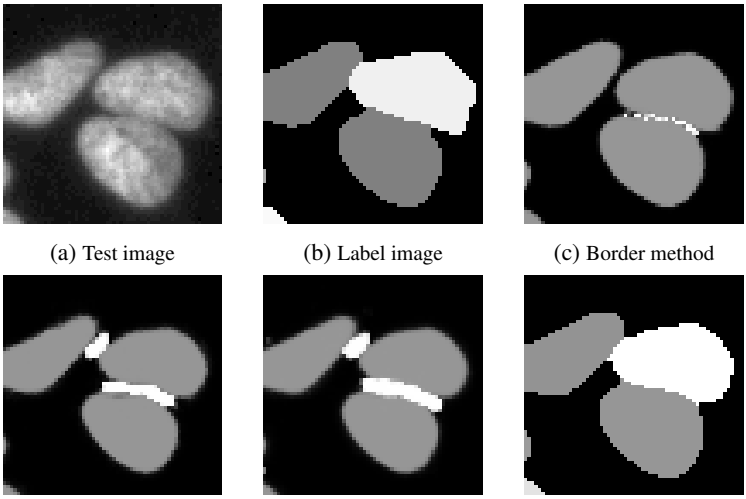
In [31], it is suggested to use the output classes: background, object, border between touching objects. Thus, the network is enforced to learn the border, where it is useful. Figure 5e shows an exemplary erroneous result using the `CceDice` loss and the softmax activation. Now there is a border in between the two objects, but it is not closed. This again can result in merged objects after post-processing. However, the idea of the border method seems to offer an elegant way to train the network and to tweak its output to a desired direction. The training borders can be generated from the provided label images.

For the instance segmentation in Figure 5h, a marker-controlled watershed post-processing was used on the border method result (Figure 5e). The thresholded ($th = 0.3$), inverted background channel was used as input image and a thresholded ($th = 0.6$) with the border channel **B** processed object channel **O** as markers:

$$o'_{ij} = o_{ij} * (1 - b_{ij}) \ . \tag{2}$$

# 4 Improving the Segmentation of Microscopy Images

In the previous section, it was shown how to start segmenting microscopy images using standard architectures, methods and loss functions. If the demands on the segmentation accuracy are not fulfilled yet, there is need for modificati-



(a) Test image (b) Label image (c) Border method

(d) Two-network approach (e) Two-network approach with TTA (f) Prediction (e) post-processed

Figure 6: Improving the results of the 4-block U-Net using a two-network approach. The approach improves the borders (white) and the prediction. Shown are 64×64 px crops. The images (b) and (f) are color-coded. TTA: test-time augmentation (see section 4.2).

ons of the architecture, the training process, and the post-processing. A good source of information besides publications are forums of segmentation competitions, e.g., [31] a post of the Kaggle Data Science Bowl 2018 winner.

## 4.1 Two-Network Approach

The border method in Figure 5e shows a reasonably good prediction but the border is not closed. A simple but powerful trick can overcome this limitation in many cases: train the network on morphologically dilated borders and eroded objects. Figure 6d shows that this simple trick provides much better seeds for the watershed post-processing. Now, borders are closed, thick and wide enough. Additionally, even on regions where borders are not necessarily needed, the network predicts borders. Advantages of this approach, in contrast to a centroid prediction as marker, are less wrongly predicted seeds and split objects since the markers are nearly as big as the real object.

Since morphological erosion is used for the training of the network, the use of the inverted background channel for the watershed post-processing would result in too small predicted objects. Thus, a second network with a one channel output trained on the original labels is needed to obtain a post-processed result with realistic object size. The thresholded ($th = 0.3$) output of this network can then be used for the post-processing. Figure 6f shows the final result combining the network used for the prediction in Figure 5c with that for the predictions in Figure 6d and Figure 6e.

## 4.2 Ensembling

Following the idea in the last section of using more than one network to improve the post-processing, the averaged output of multiple networks can be used to improve the final prediction. This results in an increase of training and prediction time. A simpler approach is the so-called test-time augmentation (TTA). In TTA, one single network is trained, but multiple outputs are estimated, e.g., by flipping the test image, making a prediction, and flip the prediction back. The back-flipped prediction should be nearly the same as the original prediction, but may provide additional information about borders or prediction errors. The resulting object channel **O** and the boundary channel **B** are then the pixelwise mean or maximum respectively of the corresponding ensemble channels.

For the prediction in Figure 6e flipping (up-down, left-right) and a 90°-rotation were used. The TTA prediction shows more clear borders. In cases, where the border is not closed, TTA offers a simple but powerful tool for corrections and better markers (cf. Figure 7). In some special cases, it may worsen the prediction, e.g., if two shifted borders are predicted instead of one big border, resulting in two markers and wrongly split objects. However, this should not occur with the two-network approach.

## 4.3 Evaluation Metrics

When the obvious segmentation errors are identified and minimized, it is beneficial to have some measure for the segmentation accuracy. This measure is

(a) Test image (b) Label image (c) Without TTA (d) With TTA

Figure 7: Improving predictions using test-time augmentation (TTA). With TTA the border (white) is complete and the objects (gray) can be separated.

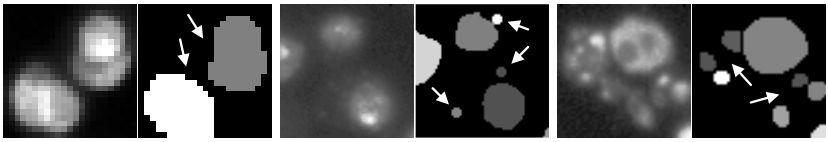
called a metric. However, a look onto the raw predictions is still useful since error sources such as too short boundaries cannot be identified in the metric score. A metric is applied to a final, post-processed result. Thus, it cannot represent the potential of a model.

Common metrics are recall, precision and F-Score. In the Kaggle segmentation challenge [16], the mean average precision at different intersection over union (IoU) thresholds is used. The thresholds $t$ range from 0.5 to 0.95 with a step size of 0.05. A true positive is counted when a single predicted object matches a ground truth object with an IoU above the threshold. The mean average precision $P_{\text{IoU}}$ is then calculated as [16]:

$$P_{\text{IoU}} = \frac{1}{N_t} \sum_t \frac{TP(t)}{TP(t) + FP(t) + FN(t)} , \tag{3}$$

with the true positives $TP$, the false positives $FP$, the false negatives $FN$, and the number of thresholds $N_t$. The final score $\overline{P}_{\text{IoU}}$ is the mean taken over the individual average precisions of each test image.

In [15], a critical analysis about challenges as standard validation for biomedical image analysis methods is provided. It is shown that in challenges the rank of an algorithm is generally not robust to the test data, the ranking scheme, and the observers making the reference labels. For segmentation, rankings can change a lot by using another metric. Additionally, different annotators may produce different winners. Figure 8 shows some examples for label errors in the test set.



(a) Possibly merged (b) Possibly artifacts (c) Possibly split

Figure 8: Possible label errors (arrows) in the used test set. The shown label images are color-coded which enables to distinguish touching objects.

## 4.4 Improving the Training Process

Data augmentation is a common method to increase the training set and to improve the robustness of a model [35]. Obvious augmentations are rotation and flipping of images since characteristic image properties are preserved. Other augmentations are, for instance, blurring, contrast changes, noise adding and affine transformations. In [31], it is suggested to copy and paste nuclei within an image to introduce more borders to the training process. Anyway, augmentations can also result in unnatural images and it is not clear whether the robustness will improve or not.

Tuning the hyperparameters of the training process can improve the segmentation accuracy as well. To tune the learning rate, schedulers can be used, which set the learning rate after a specified number of epochs to a decreased value. The reduction of the learning rate on a validation loss plateau may improve the segmentation accuracy as well. Keras provides various callbacks to modify the learning rate during training. Instead of Adam, e.g., stochastic gradient descent [20] can be used. Additionally, some regularization can be added, e.g., spatial dropout [36].

## 4.5 Architecture Adaption

If the segmentation accuracy is still not high enough, architecture changes may be needed. A reason for choosing the U-Net architecture was the number of existing modifications. Another promising approach is the exchange of the encoder with a pre-trained specialized image classification network and using end-to-end training as suggested in [31]. Anyway, for that also the hardware

Table 3: Overview of the trained networks used in Figure 9. The `BceDice` loss and the sigmoid activation were used to train the networks B, and the `CceDice` loss and the softmax activation for the networks M. In a two-network approach, the networks B provide the needed thresholded binary (B) images and the networks M the markers (M). The training process is stopped after 10 epochs without validation loss improvement.

| Network | Blocks, F | Classes | $lr$ | Augment. | Erosion/Dilation |
|---|---|---|---|---|---|
| B1 | 4, 64 | 1 | 1e-4 | - | - |
| B2 | 4, 64 | 1 | 1e-4 | ×[a] | - |
| B3 | 4, 64 | 1 | 1e-4 | ×[b] | - |
| B4 | 4, 64 | 1 | 2e-4* | ×[b] | - |
| B5 | 5, 32 | 1 | 2e-4* | ×[b] | - |
| M1 | 4, 64 | 3 (boundary) | 1e-4 | - | - |
| M2 | 4, 64 | 3 (border) | 1e-4 | - | - |
| M3 | 4, 64 | 3 (border) | 1e-4 | - | × |
| M4 | 4, 64 | 3 (border) | 1e-4 | ×[a] | × |
| M5 | 4, 64 | 3 (border) | 1e-4 | ×[b] | × |
| M6 | 4, 64 | 3 (border) | 2e-4* | ×[b] | × |
| M7 | 5, 32 | 3 (border) | 2e-4* | ×[b] | × |

[a] flipping (left-right), flipping (up-down), 90°-rotation, noise, scale, blur.
[b] flipping (left-right), flipping (up-down), 90°-rotation.
* learning rate is quartered on a validation loss plateau (5 epochs without improvement).

has to be available. Finding the optimal cut-off layer in transfer learning may also be useful [37].

## 4.6 Evaluation of the Segmentation Accuracy

Figure 9 shows the averaged test set precision score $\overline{P}_{\mathrm{IoU}}$ of the networks in Table 3 which were initialized and trained only once. As expected, the two-network approach M3B1 using the networks M3 and B1 outperforms the simple border method M2 and the boundary method M1. TTA improves the precision for every network. A comprehensive study of training augmentation types with multiple initializations is planned for future work. To validate the improvement of a higher learning rate that is reduced on a plateau, more initializations are needed too. The Kaggle Data Science Bowl winner [31] reached a score of 0.631 on the official test set which includes new cell types. However, since no labels are provided for that data, a comparison is not possible. Training a similar model as used in [31] is also planned but demands a GPU with more memory than the used Nvidia Quadro P4000.

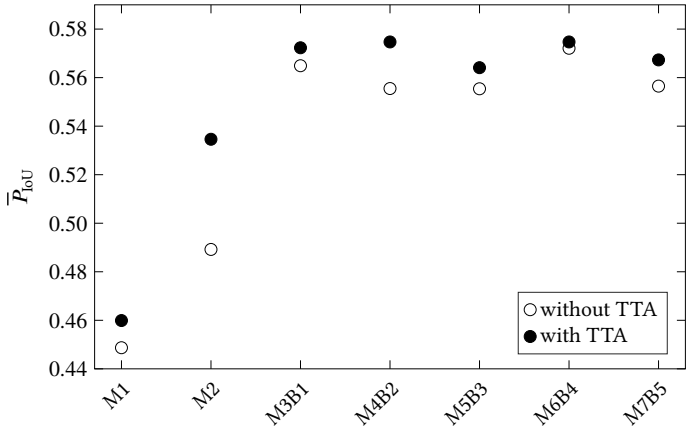



Figure 9: Mean average precision on the test set of the trained networks in Table 3. The combination of two networks B and M means that the two-network approach was used.

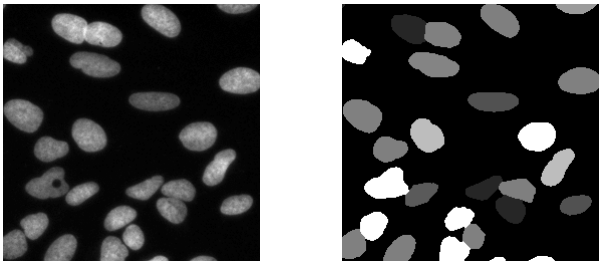
The training time on a single cropped training image is between about 0.1 s (4-block U-Net) and 0.05 s (5-block U-Net) on the used Nvidia GPU. The number of needed epochs ranges from 40 to 100 for every network. The use of a 5-block U-Net with reduced feature maps (M7B5) reduces the mean training and prediction times without losing accuracy.

Figure 10 and Figure 11 show some exemplary segmentations of test images. The network M6B4 with the highest mean averaged precision shows a better generalization to the test set as the first result M2. However, errors still occur. Further improvements may be possible with more training data similar to Figure 11a.

The use of a-priori information about the object sizes, may improve the post-processing and segmentation accuracy. However, in this data set the false negative rate may increase due to artifacts like in Figure 8.

## 5 Towards Segmentation of 3D Microscopy Images

One approach for the segmentation of 3D or 3D+t microscopy data is to apply 2D segmentation slice by slice and to fuse the segments afterwards. Using



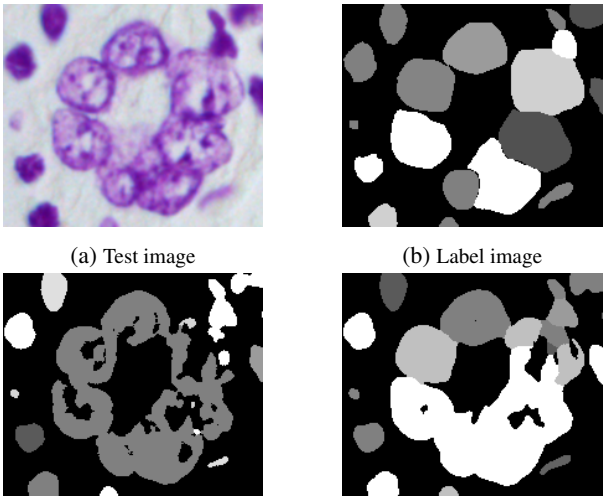
(a) Test image (b) Label image

(c) First result: M2 (d) Improved result: M6B4

Figure 10: Exemplary segmentation of a test image. This image type is frequent in the training set. Shown are $256 \times 256$ px crops and the labeled images are color-coded which enables to distinguish touching objects. In contrast to the first result (c), the improved result (d) shows no merged objects.

3D convolutional layers such as in [10] or [33], segment fusion can be omitted. Instead of border lines in 2D segmentation, border areas may provide a powerful tool for instance segmentation. Furthermore, a combination of three 2D-U-Nets for the *xy*-, *xz*-, and *yz*-slices of the 3D volume can boost memory efficiency [38]. Weight sharing of these 2D-U-Nets is possible and may reduce the training time.

A comparison of the possible ways towards 3D segmentation with a classical segmentation algorithm, e.g., [39], is planned. If for 3D or 3D+t data the use of additional information, e.g., the structure tensor which can also be used for segmentation [40], can improve the training process is an open question we also want to investigate.



(a) Test image (b) Label image

(c) First result: M2 (d) Improved result: M6B4

Figure 11: Exemplary segmentation of a test image. Objects similar to the big cells are not present in the training data. Shown are $180 \times 210$ px crops and the labeled images are color-coded which enables to distinguish touching objects. The improved result (d) shows a better generalization than (c).

## 6 Conclusion

In this contribution, we showed a possible workflow for starting and improving the segmentation of roundish objects in microscopy images. The hereby gained expertise should also help readers in 3D segmentation tasks and in segmentation of arbitrarily shaped objects. Summarized, our suggestions are:

1. Start with a one channel output U-Net to get a feeling how challenging the data are and if there are difficulties with specific objects.
2. Be aware of class imbalance.
3. Use borders instead of boundaries for instance segmentation.
4. Try erosion and dilation to get better markers for the watershed post-processing. Consider the two-network approach in this case.
5. Do not use training data augmentation naively. Some augmentations may worsen the accuracy.
6. Try to boost performance with test-time augmentation.

7. Use metrics but do not rely on them solely. Look at your data to find bottlenecks and difficulties.
8. Look for additional training data sets. This may improve the generalization of the network to the test set.
9. Adjust your training process hyperparameters.
10. Use a-priori information, e.g., about the object sizes, to improve the post-processing.
11. Modify your architecture and try other encoders if the accuracy is not high enough and if the required hardware is available.

A future goal is to improve the segmentation quality in such a way that in sophisticated medical and biological analyses, e.g., of biological zebrafish data using EmbryoMiner [41], no more manual corrections are needed.

## References


[1] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proc. International Conference on Learning Representations*, 2014. arXiv: 1409.1556v6.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, ..., L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[5] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *AAAI Conference on Artificial Intelligence*, pages 4278–4284, 2017.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[7] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[8] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[9] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241. Springer International Publishing, 2015.

[10] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, pages 424–432. Springer International Publishing, 2016.

[11] V. Ulman, M. Maška, K. EG Magnusson, O. Ronneberger, C. Haubold, N. Harder, ..., C. Ortiz-de-Solorzano. An Objective Comparison of Cell-Tracking Algorithms. *Nature Methods*, 14(12):1141–1152, 2017.

[12] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, 521:436–444, 2015.

[13] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso. Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 240–248. Springer, 2017.

[14] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. W. M. van der Laak, B. van Ginneken, and C. I. Sánchez. A Survey on Deep Learning in Medical Image Analysis. *Medical Image Analysis*, 42:60–88, 2017.

[15] L. Maier-Hein, M. Eisenmann, A. Reinke, S. Onogur, M. Stankovic, P. Scholz, ..., A. Kopp-Schneider. Is the Winner Really the Best? A Critical Analysis of Common Research Practice in Biomedical Image Analysis Competitions, 2018. arXiv: 1806.02051v1.

[16] Kaggle. 2018 Data Science Bowl, 2018. `https://www.kaggle.com/c/data-science-bowl-2018`.

[17] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[19] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. *Efficient BackProp*, pages 9–50. Springer Berlin Heidelberg, 1998.

[20] L. Bottou. *Stochastic Gradient Descent Tricks*, pages 421–436. Springer Berlin Heidelberg, 2012.

[21] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization, 2014. arXiv: 1412.6980v9.

[22] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 448–456. PMLR, 2015.

[23] C. McQuin, A. Goodman, V. Chernyshev, L. Kamentsky, B. A. Cimini, K. W. Karhohs, ..., A. E. Carpenter. Cellprofiler 3.0: Next-Generation Image Processing for Biology. *PLOS Biology*, 16(7):1–17, 2018.

[24] F.-F. Li, J. Johnson, and S. Yeung. CS231n: Convolutional Neural Networks for Visual Recognition. Lecture 11: Detection and Segmentation, 2018. `http://cs231n.stanford.edu/syllabus.html`.

[25] X.-Y. Zhou, C. Riga, S.-L. Lee, and G.-Z. Yang. Towards Automatic 3D Shape Instantiation for Deployed Stent Grafts: 2D Multiple-Class and Class-Imbalance Marker Segmentation with Equally-Weighted Focal U-Net, 2018. arXiv: 1711.01506v4.

[26] J. W. Johnson. Adapting Mask-RCNN for Automatic Nucleus Segmentation, 2018. arXiv: 1805.00500v1.

[27] F. Chollet et al. Why Has Keras Been So Successful Lately at Kaggle Competitions?, 2016-08. `https://www.quora.com/Why-has-Keras-been-so-successful-lately-at-Kaggle-competitions`.

[28] J. Stegmaier, J. Arz, B. Schott, J. C. Otte, A. Kobitski, G. U. Nienhaus, ..., R. Mikut. Generating Semi-Synthetic Validation Benchmarks for Embryomics. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 684–688, 2016.

[29] V. Ljosa, K. L. Sokolnicki, and A. E. Carpenter. Annotated High-Throughput Microscopy Image Sets for Validation. *Nature Methods*, 9(7):637, 2012.

[30] A. Sevastopolsky, S. Drapak, K. Kiselev, B. M. Snyder, and A. Georgievskaya. Stack-U-Net: Refinement Network for Image Segmentation on the Example of Optic Disc and Cup, 2018. arXiv: 1804.11294v1.

[31] S. Seferbekov. *[ods.ai] Topcoders, 1st Place Solution*, 2018-04. Winner of Kaggle 2018 Data Science Bowl. `https://www.kaggle.com/c/data-science-bowl-2018/discussion/54741`.

[32] J. C. Caicedo, J. Roth, A. Goodman, T. Becker, K. W Karhohs, C. McQuin, ..., A. E. Carpenter. Evaluation of Deep Learning Strategies for Nucleus Segmentation in Fluorescence Images. *bioRxiv*, 2018.

[33] F. Milletari, N. Navab, and S. A. Ahmadi. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571, 2016.

[34] S. J. Reddi, S. Kale, and S. Kumar. On The Convergence of Adam and Beyond. In *International Conference on Learning Representations*, 2018.

[35] A. Bartschat, T. Unger, T. Scherr, J. Stegmaier, R. Mikut, and M. Reischl. Robustness of Deep Learning Architectures with Respect to Training Data Variation. In *Proc., 28. Workshop Computational Intelligence, Dortmund*, 2018.

[36] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient Object Localization Using Convolutional Networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 648–656, 2015.

[37] N. Prodanova, J. Stegmaier, S. Allgeier, S. Bohn, O. Stachs, B. Köhler, ..., A. Bartschat. Transfer Learning with Human Corneal Tissues: An Analysis of Optimal Cut-Off Layer, 2018. arXiv: arXiv:1806.07073v2.

[38] J. Wasserthal, P. Neher, and K. H. Maier-Hein. TractSeg - Fast and Accurate White Matter Tract Segmentation. *NeuroImage*, 183:239–253, 2018.

[39] J. Stegmaier, J. C. Otte, A. Kobitski, A. Bartschat, A. Garcia, G. U. Nienhaus, ... R. Mikut. Fast Segmentation of Stained Nuclei in Terabyte-Scale, Time Resolved 3D Microscopy Image Stacks. *PLOS ONE*, 9(2):1–11, 2014.

[40] B. Jähne. *Digital Image Processing*. Springer, 6th ed., 2005.

[41] B. Schott, M. Traub, C. Schlagenhauf, M. Takamiya, T. Antritter, A. Bartschat, ..., J. Stegmaier. EmbryoMiner: A New Framework for Interactive Knowledge Discovery in Large-Scale Cell Tracking Data of Developing Embryos. *PLOS Computational Biology*, 14(4):1–18, 2018.

# Schätzung der Körperpose von Autofahrern aus Tiefenbildern

Franz Albers[1], Cecilia De la Parra, Jan Braun[1], Frank Hoffmann[1] und Torsten Bertram[1]

[1]Lehrstuhl für Regelungssystemtechnik,Technische Universität Dortmund
Otto-Hahn-Straße 8
E-Mail: Franz.Albers@tu-dortmund.de

## 1 Einführung

Aktuell werden in den ersten Fahrzeugen Assistenzfunktionen eingeführt, die den Fahrer während einer bedingt automatisierten Fahrt gemäß SAE Level 3 von seiner permanenten Überwachungsaufgabe entbinden und ihm so die Ausführung fahrfremder Tätigkeiten erlauben. Stößt das automatisierte Fahren in einer komplexen Fahrsituation an seine Grenzen, muss der Fahrer nach Aufforderung jedoch in der Lage sein, die Fahraufgabe zu übernehmen. Beim Delegieren der Fahraufgabe wird dem Fahrer eine garantiert ausreichende Zeitspanne zur Erfassung der Verkehrssituation und zur Übernahme der Fahrzeugführung eingeräumt. In diesem Zusammenhang werden Driver-Monitoring Systeme eingesetzt, die nicht nur wie bisher üblich die Müdigkeit und Ablenkung des Fahrers, sondern auch dessen Übernahmebereitschaft während einer automatisierten Fahrt erfassen.

Der aktuelle Fahrerzustand wird auf motorischen, sensorischen und kognitiven Ebenen beschrieben [1], welche jeweils die Übernahmebereitschaft des Fahrers beeinflussen. Dabei gibt der motorische Zustand an, wie ein Fahrer durch eine motorische Reaktion auf Umweltreize reagiert. Der sensorische Zustand definiert, zu welchem Grad ein Mensch über visuelle, auditive oder haptische Kanäle seine Umwelt erfasst. Der kognitive Zustand beschreibt die aktuelle Fähigkeit eines Menschen Umweltreize mental zu verarbeiten und wahrzunehmen.

Dieser Beitrag stellt eine Methode zur Schätzung des motorischen Fahrerzustands anhand der Körperpose durch die visuelle Überwachung des Fahrzeuginnenraums mit einer Tiefenkamera vor.

Bekannte Verfahren des Skelett-Trackings (wie beispielsweise die *Microsoft Kinect* [2], näher beschrieben in Abschnitt 2) erfordern üblicherweise einen Mindestabstand zur Kamera. Da dieser im Fahrzeuginnenraum nicht gegeben ist, wird ein neuer Datensatz von Tiefenkamerabildern in einem nachgebildeten Cockpit aufgenommen und mithilfe eines hochgenauen Motion-Capture-Systems mit den Referenzpositionen der Körpergelenke annotiert. Dafür wird eine *Intel RealSense SR300* Tiefenkamera mit einem Mindestabstand von ca. 0.2 m verwendet. Die genaue Vorgehensweise zur Datengewinnung wird detaillierter in Abschnitt 3.1 beschrieben.

Die Methode zum Skelett-Tracking beruht auf einem zweistufigen Ansatz. In einem ersten Schritt werden zunächst die Bildkoordinaten der Körpergelenke mit einem *Convolutional Neural Network* (CNN), das sich an der bekannten *DenseNet* Architektur [3] orientiert, geschätzt. Die Schätzung der Gelenkpositionen in Bildkoordinaten wird in Abschnitt 3.2 erläutert.

Der zweite Schritt besteht aus der Schätzung der räumlichen Körperpose anhand der Distanzen zwischen den einzelnen Körpergelenken im Bild. Dafür werden ausgehend von den euklidischen 2D-Distanzmatrizen die räumlichen Abstände zwischen den Körpergelenken mit einem weiteren neuronalen Netz geschätzt. Das Verfahren zur räumlichen Körperposenschätzung wird in Abschnitt 3.3 beschrieben.

In Abschnitt 4 erfolgt eine Evaluierung der Skelett-Tracking Methoden und ein Vergleich mit bekannten Verfahren aus der Literatur.

## 2 Stand der Technik

Ein weit verbreiteter Ansatz zum Skelett-Tracking findet in den *Microsoft Kinect* Kamerasystemen Anwendung und basiert auf *Random Decision Forests* [2]. Dabei wird die Aufnahme eines Menschen in einem Tiefenbild pixelweise in vordefinierte Körperteile segmentiert. Diese Regionen liegen jeweils in der Nähe von Gelenken. Schätzungen der genauen Positionen der Körpergelenke

und ihrer zugehörigen Wahrscheinlichkeiten werden anhand der Verteilungen der einzelnen Segmentierungen der Körpergelenke mit einem *Gaussian Mean-Shift* Algorithmus berechnet. Für das Training des Klassifikators wird eine Datenbank von mehreren hunderttausend synthetisch erzeugten Tiefenbildern mit sich deutlich unterscheidenden Körperposen erstellt, um die Robustheit des Klassifikators zu erhöhen und eine Überanpassung zu vermeiden.

Aufbauend auf diesem Verfahren entwickelt [4] einen Ansatz zur automatischen Klassifikation von Nebentätigkeiten von Autofahrern während einer automatisierten Fahrt. Dafür wird ebenfalls die *Microsoft Kinect* Kamera verwendet, die aufgrund des benötigten Mindestabstands von 1.4 m und der eingeschränkten Platzverhältnisse im Fahrzeug suboptimal an der Beifahrer-A-Säule platziert ist.

Motiviert durch die hohen Erkennungsraten neuronaler Netze bei der bildbasierten Klassifizierung von Objekten, präsentiert [5] einen Ansatz zur Schätzung von Körperposen in Bildkoordinaten mit tiefen neuronalen Netzen. Dabei verwendet ein kaskadiertes CNN das vollständige RGB-Eingangsbild für die Schätzung der Gelenkpositionen und verfeinert mit jeder Ebene die Schätzung der einzelnen Gelenkpositionen.

Die von [6] vorgestellten *Convolutional Pose Machines* bauen auf diesem Konzept auf und verwenden mehrere sequentielle CNN, die als Eingang einer Stufe jeweils die von der Vorstufe geschätzten Wahrscheinlichkeiten für Körpergelenkpositionen als Heat Map nutzen.

Verschiedene Ansätze erweitern die Körperposenschätzung im Bild um eine räumliche Dimension. [7] schätzt die 3D-Körperpose unter der Annahme konstanter Länge der einzelnen Gliedmaßen zwischen den Gelenken durch die Minimierung der $L_1$ Norm zwischen der räumlichen Pose und der Detektion im 2D-Bild.

Ein weiteres Verfahren zur Schätzung der räumlichen Pose aus einer initialen Detektion im Bild wird von [8] präsentiert. Dieser Ansatz codiert die Entfernungen zwischen den einzelnen Gelenken in euklidischen Distanzmatrizen (EDM) und trainiert verschiedene einfach strukturierte neuronale Netze auf die Schätzung der Abbildung der Distanzen im Bild auf die räumlichen Entfernun-

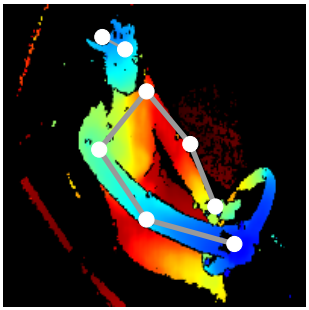


Bild 1: Skelett-Tracking der Gelenke des Oberkörpers

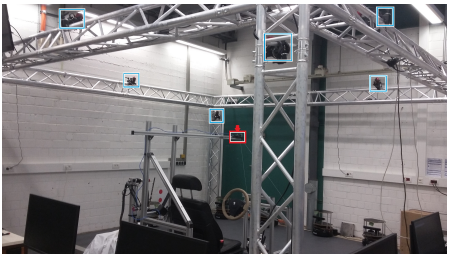
gen. Vorteile der EDMs liegen in der natürlichen Kodierung der Pose und der Invarianz gegenüber Rotationen, Translationen und Normalisierungen.

Ein echtzeitfähiger Ansatz zur Erfassung der räumlichen Körperpose mit einer RGB-Kamera wird von [9] vorgestellt. Ein einziges CNN schätzt gleichzeitg die 2D- und 3D-Gelenkpositionen. Anschließend wird die Pose anhand eines kinematischen Skelett-Modells optimiert, um eine zeitlich stabile Körperposenschätzung zu erreichen.

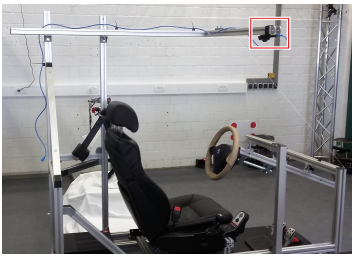
Die Schätzung der Gelenkpositionen hat sich in den bisher vorgestellten Methoden etabliert. Eine andere Möglichkeit ist die Schätzung der Gelenkwinkel, die beispielsweise von [10] verwendet wird. Für die Rekonstruktion des Skeletts sind in diesem Fall allerdings zusätzlich die Distanzen zwischen den einzelnen Gelenken notwendig.

## 3 Skelett-Tracking Konzept

Für die Erkennung der Aktivitäten des Fahrers ist insbesondere die Haltung des Oberkörpers entscheidend. Da die gesamte Körperpose des Fahrers in einem Fahrzeug mit einer einzigen Kamera schwierig zu erfassen ist, fokussiert sich der in diesem Beitrag beschriebene Ansatz auf die Schätzung der Positionen der Hände, der Ellenbogen, der Schultern und des Kopfes (siehe Abbildung 1).



(a) Motion-Capture-System



(b) Fahrersitz Mock-up

Bild 2: Umgebung für die Aufnahme der Referenzdaten

Es existieren verschiedene Datensätze von Körperposen in annotierten Tiefenbildern, wie beispielsweise der bekannte *Human3.6m* Datensatz, der 3.6 Millionen gelabelter RGB-D Bilder von Menschen bei verschiedenen Aktivitäten enthält [11]. Für den speziellen Anwendungsfall der Fahrerüberwachung gibt es zum jetzigen Zeitpunkt unseres Wissens nach noch keinen frei verfügbaren Datensatz. Daher werden mithilfe eines Motion-Capture-Systems Referenzdaten mit verschiedenen Probanden aufgenommen. Mit Hilfe des Datensatzes wird ein auf der *DenseNet* Architektur basierendes neuronales Netz trainiert.

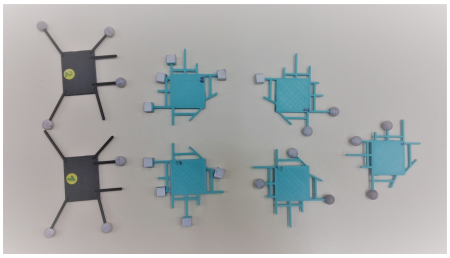
Aufbauend auf den so bestimmten 2D-Skeletten wird die räumliche Körperpose mit Hilfe von euklidischen Distanzmatrizen und einem zweiten neuronalen Netz bestimmt. Für eine robuste Schätzung berücksichtigt die Optimierung die räumlich zeitliche Kontinuität der Gelenkposen.

Die Implementierung erfolgt in *Python* unter Verwendung des TensorFlow Frameworks [12].

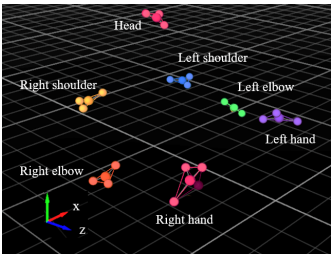
## 3.1 Generierung von Referenzdaten und Preprocessing

Die Aufnahme der Referenzdaten erfolgt mit Hilfe eines hochgenauen Marker-basierten Motion-Capture-Systems der Firma *Optitrack* mit 13 Kameras[3] (in Abbildung 2a blau markiert). Die Software *Optitrack Motive* bietet prinzipiell die Möglichkeit des Skelett-Trackings, benötigt dafür jedoch Marker am

[3] Optitrack Flex 13.
http://optitrack.com/products/flex-13/



(a) Verschiedene Markerstrukturen




(b) Visualisierung der Marker

Bild 3: Markerstrukturen für die automatische Annotierung von Gelenken

Rücken des Probanden. Diese werden durch den Fahrersitz verdeckt, weshalb die direkte Nutzung der Skelett-Tracking Funktion nicht möglich ist. Stattdessen werden Marker mit verschiedenen Strukturen (siehe Abbildung 3a) an den Körpergelenken befestigt. Die jeweiligen Strukturen werden den einzelnen Gelenken zugeordnet, wodurch sich Tiefenbilder automatisch mit den korrekten Positionen der Gelenke annotieren lassen, wie in Abbildung 3b dargestellt.

Für die Aufnahme wird eine Nachbildung eines offenen Cockpits (Fahrersitz Mock-up, siehe Abbildung 2b) verwendet, um Verdeckungen der an den Gelenken der Probanden angebrachten Marker zu vermeiden. An dem Mock-up ist eine *Intel RealSense SR300* Tiefenkamera (in Abbildung 2 rot markiert) angebracht, die Tiefenbilder in Entfernungen zwischen 0.2 m und 1.5 m aufnimmt.

Um die Tiefenbilder mit den korrekten Gelenkpositionen $\mathbf{p}_{\text{label}} = [x, y, z]^\intercal$ zu annotieren, wird die homogene Transformation $\mathbf{T}_{\text{C}}$ vom Koordinatensystem des Motion-Capture-Systems in das Kamerakoordinatensystem bestimmt. Eine erste Schätzung $\mathbf{T}_{\text{est}}$ dieser Transformation erfolgt über die Positionsbestimmung eines auf der Kamera angebrachten Markers, der vom Motion-Capture-System erfasst wird. Der *Iterative Closest Point* Algorithmus [13] identifiziert anhand dieser initialen Schätzung der Gelenkpositionen $\mathbf{p}_{\text{est}}$ die gemäß der euklidischen Distanz nächsten Punkte aus der 3D Punktwolke. Abbildung 4 verdeutlicht die Notwendigkeit der Korrektur der Abweichungen zwischen der ersten Schätzung $\mathbf{p}_{\text{est}}$ und den korrekten Positionen $\mathbf{p}_{\text{label}}$. Zu sehen sind die deutlichen Abweichungen der initialen Schätzung der Gelenkposition von der

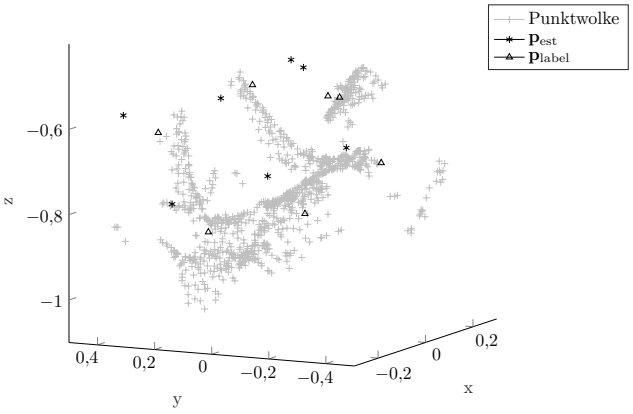



Bild 4: Abweichung zwischen initialer Schätzung und korrigierter Schätzung

Punktwolke, die den Fahrer darstellt und die korrigierten Positionen der Gelenke. Für die Bestimmung der Gelenkpositionen im Bild werden diese 3D Gelenkpositionen mit einem Lochkameramodell in die Bildkoordinaten projiziert, wodurch sich die Annotierungen ergeben. Um die CNN Architektur kompakt zu halten und eine resultierende geringe Laufzeit für die Schätzung der Gelenkpositionen zu erreichen, werden die Bilder auf eine Auflösung von $200 \times 200$ Pixeln skaliert.

Auf diese Weise wurden mehr als 50.000 Tiefenbilder mit vier Probanden und drei Probandinnen aufgenommen und annotiert, von denen 80 % als Trainings- und 20 % als Validierungsdaten dienen. Die Referenzdaten enthalten sowohl Tätigkeiten mit Bezug zur Fahraufgabe (z. B. Lenken, Schalten, etc.) als auch fahrfremde Nebentätigkeiten (z. B. Essen, Trinken, Nutzung des Smartphones, Buch lesen, etc.).

Vor dem eigentlichen Skelett-Tracking erfolgt eine Segmentierung des Tiefenbildes, um den statischen Hintergrund der Szene aus dem Bild zu filtern. Um den Einfluss des Rauschens zu minimieren, werden mehrere Bilder des unbesetzten Fahrersitzes aufgenommen und über die Zeit gemittelt. Aufnahmen werden von dem so detektierten Hintergrund der Szene subtrahiert (*Background Subtraction*). Liegt die Differenz zwischen einer Aufnahme und dem

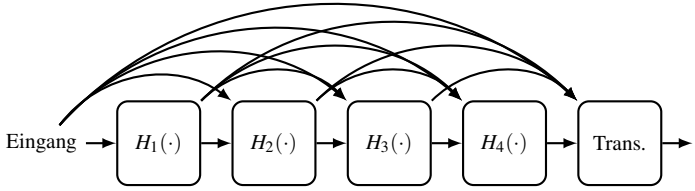



Bild 5: Dichter Block mit $n = 4$ Schichten

Hintergrund oberhalb einer Grenze, wird der jeweilige Pixel als Vordergrund klassifiziert. Das auf diese Weise vorverarbeitete Tiefenbild des Fahrers dient im Folgenden der Schätzung der Gelenkpositionen.

## 3.2 Schätzung der Gelenkpositionen in Bildkoordinaten

Die 2D-Körperposenschätzung in Bildkoordinaten nutzt die *DenseNet* Architektur [3] für neuronale Netze, die durch das Verbindungsmuster der Schichten innerhalb von mehreren dichten Blöcken charakterisiert wird. Die Eingänge der einzelnen Schichten eines Blocks sind nicht nur die Ausgänge (Feature Maps) der jeweils vorherigen Schicht, sondern die verketteten Feature Maps aller vorherigen Schichten (inklusive der Input Feature Maps, siehe Abbildung 5). Jede Schicht im dichten Block stellt eine aus mehreren Operationen bestehende nichtlineare Transformation $H_\ell$ dar, die aus verschiedenen Funktonen zusammengesetzt ist. Der Index $\ell \in [1 \dots n]$ bezeichnet den Index der Schicht und der Parameter $n$ definiert die Anzahl der Schichten des dichten Blocks.

Bei $k$ Feature Maps, die eine einzelne Schicht ausgibt (Wachstumsrate des Blocks), ergeben sich auf diese Weise insgesamt $k_0 + k(\ell - 1)$ Feature Maps als Eingang der $\ell$-ten Schicht. Dabei entspricht der Parameter $k_0$ der Anzahl der Kanäle in der Eingangsschicht. Auf diese Weise wird der Informationsfluss zwischen den Schichten eines Blocks maximiert.

Weitere Vorteile dieser Architektur sind die Abschwächung des Problems des verschwindenden Gradienten beim Aktualisieren der Gewichte durch den Back-Propagation-Algorithmus, eine geringere Anzahl von Parametern und der Regularisierungseffekt, der die Tendenz des Netzes zum Overfitting verringert [3].

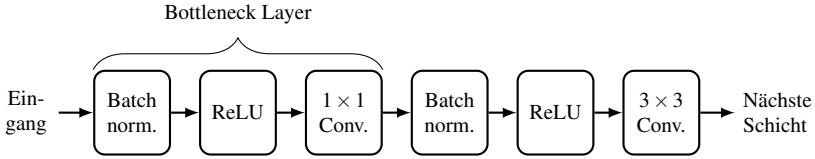



Bild 6: Nichtlineare Transformation $H_\ell(\cdot)$

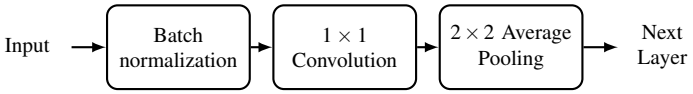



Bild 7: Übergangsschicht zwischen den Blöcken

Die einzelnen Operationen der nichtlinearen Transformationen $H_\ell$ eines dichten Blocks sind in Abbildung 6 dargestellt. Bottleneck Schichten, bestehend aus Batch-Normalisierung, Rectified Linear Unit (ReLU) und einer $1 \times 1$ Convolution reduzieren die Anzahl der Feature Maps innerhalb des Blocks und steigern dadurch die Berechnungseffizienz. Üblicherweise werden $4k$ $1 \times 1$ Filter bei einer Wachstumsrate $k$ eingesetzt [3]. Anschließend erfolgen eine Batch Normalisierung, ReLU und eine $3 \times 3$ Convolution.

Eine Übergangsschicht (Transition Layer) am Ende eines Blocks realisiert das Downsampling der Feature Maps. Der Aufbau der Übergangsschichten ist in Abbildung 7 dargestellt. Die Schichten bestehen aus Batch Normalisierung, $1 \times 1$ Convolution und $2 \times 2$ Average Pooling.

Abbildung 8 gibt einen Überblick über das komplette neuronale Netz für die Körperposenschätzung. Nach einem Abschnitt zum Downsampling, bestehend aus $7 \times 7$ Convolution und einer $2 \times 2$ max. Pooling Schicht folgen $d$ dichte Blöcke. Die Regression der 2D-Körperpose nutzt eine Flatten Schicht und zwei Fully Connected (FC) Schichten. Die letzte vollvernetzte Schicht besitzt sechszehn Ausgänge und klassifiziert so die Koordinaten der acht Gelenke des Oberkörpers im Bild. Die Prädiktionen des *DenseNets* werden anschließend auf die ursprüngliche Größe des Bildes skaliert.

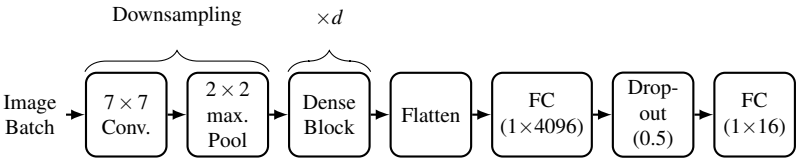



Bild 8: Überblick DenseNet zur Körperposenschätzung

Im Folgenden werden die Gelenke $\mathbf{p}_i$ zu einem Vektor $\mathbf{q} = [\mathbf{p}_1^\intercal, \ldots, \mathbf{p}_8^\intercal]^\intercal$ zusammengefasst. Die geschätzten Gelenkpositionen werden als Vektor $\mathbf{q}^*(\boldsymbol{\theta})$ und die durch das Motion-Capture-System bereitgestellten korrekten Gelenkpositionen als Vektor $\mathbf{q}_{\text{ref}}$ bezeichnet. In Anlehnung an [5] werden die Parameter $\boldsymbol{\theta}$ des Netzes während des Trainings durch die Minimierung der L2-Norm Verlustfunktion zwischen den Schätzungen und korrekten Gelenkpositionen gemäß Gleichung 1 aktualisiert.

$$\arg\min_{\boldsymbol{\theta}} \sum \|\mathbf{q}_{\text{ref}} - \mathbf{q}^*(\boldsymbol{\theta})\|_2^2 \tag{1}$$

Aufgrund der beschränkten Reichweite der Tiefenkamera und der vorangehenden Vordergrundsegmentierung besitzen nicht alle Pixel des Tiefenbildes eine Tiefeninformation. Für den Fall, dass ein Gelenk auf einem solchen Pixel erkannt wird, wird der nächstgelegene Pixel mit Tiefeninformation als Position des Gelenks innerhalb des segmentierten Bilds gewählt. Auf diese Weise ergibt sich eine initiale Schätzung der 3D Pose. Der im folgenden Abschnitt beschriebene Ansatz strebt eine genauere Schätzung an.

## 3.3 Schätzung der räumlichen Gelenkpositionen aus 2D Posen

Euklidische Distanzmatrizen (EDM) bieten die Möglichkeit implizit die jeweiligen Abstände zwischen den einzelnen Gelenken zu berücksichtigen [8]. Dabei stellt jeder Eintrag $m_{i,j}$ der Distanzmatrix den Abstand zwischen den Gelenken mit den Indizes $i$ und $j \in [1 \ldots 8]$ dar:

$$\text{EDM}(\mathbf{q})_{i,j} = \|\mathbf{p}_i - \mathbf{p}_j\|_2^2 \tag{2}$$

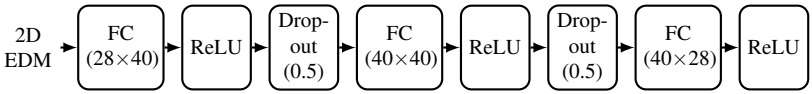



Bild 9: Neuronales Netz zur 3D-EDM Schätzung

EDM besitzen verschwindende Elemente auf der Hauptdiagonalen und sind symmetrisch aufgebaut [14]. Bei acht zu identifizierenden Gelenken ergeben sich somit insgesamt 28 Elemente, welche die paarweisen Distanzen zwischen den jeweiligen Gelenken bestimmen.

Die in diesem Abschnitt vorgestellte Methode verwendet das in Abbildung 9 dargestellte einfache neuronale Netz mit drei vollvernetzten Schichten, um die Transformation der euklidischen Distanzmatrix von 2D in 3D abzubilden [8]. Den Eingang bzw. Ausgang des Netzes bilden jeweils die 28 Elemente oberhalb der Hauptdiagonalen der 2D- bzw. 3D-EDM. Die abschließende ReLU Operation stellt die notwendige Bedingung positiver Elemente der EDM sicher [14]. Das Training des Netzes erfolgt ebenfalls anhand der mit dem Motion-Capture-System aufgenommenen Referenzdaten.

Um die implizit in den EDM enthaltenen Längen der Gliedmaßen in der Körperposenschätzung zu berücksichtigen, wird eine Optimierung der zuvor geschätzten Körperpose durchgeführt. Die 3D EDM wird im Folgenden in der Kostenfunktion $E_{\text{total}}$ verwendet, um die Distanzen zwischen den geschätzten räumlichen Positionen der Gelenke und den korrespondierenden Elementen der 3D EDM zu minimieren (siehe Gleichung 4). Um die räumlich zeitliche Kontinuität der Posen zu gewährleisten werden in Anlehnung an [9] weitere mit Kostenfaktoren $w$ gewichtete Terme bei der Optimierung berücksichtigt, so dass sich die folgende Kostenfunktion ergibt:

$$E_{\text{total}} = w_{\text{edm}} E_{\text{edm}}(\mathbf{q}) + w_{\text{ik}} E_{\text{ik}}(\mathbf{q}) + w_{\text{s}} E_{\text{s}}(\mathbf{v}) \tag{3}$$

Die darin enthaltenen Kostenterme sind folgendermaßen definiert:

$$E_{\text{edm}}(\mathbf{q}) = \sum_{m,n} |\|\mathbf{p}_m - \mathbf{p}_n\|_2^2 - \text{EDM}(\mathbf{q}(k-1))_{m,n}| \quad (4)$$

$$E_{\text{ik}}(\mathbf{q}) = \sum_{m} \|\mathbf{p}_m(k) - \mathbf{p}_m(k-1)\|^2 \quad (5)$$

$$E_{\text{s}}(\mathbf{v}) = \sum_{m} \|\mathbf{v}_m(k) - \mathbf{v}_m(k-1)\|^2 \quad (6)$$

Dabei bezeichnet $k$ den jeweiligen Frame und $m$ den Index der Gelenke. Der Term $E_{\text{ik}}$ bestraft große Abweichungen zwischen optimierter Pose $\mathbf{q}(k)$ (bzw. deren Gelenkpositionen $\mathbf{p}_m(k)$) und der EDM der zuvor geschätzten Pose. Der Term $E_{\text{s}}$ vermeidet Diskontinuitäten zwischen zeitlich aufeinander folgenden Posen indem er hohe Geschwindigkeiten $\mathbf{v}_m$ der Gelenke bestraft.

# 4 Evaluierung

Die Evaluierung des Verfahrens erfolgt zunächst für die in Abschnitt 3.2 vorgestellte 2D Schätzung und anschließend für das in Abschnitt 3.3 beschriebene räumliche Skelett-Tracking. Der Generalisierungsfehler der Posenschätzung wird anhand zweier unabhängiger Metriken evaluiert:

- Mean Per Joint Position Error (MPJPE): Mittlerer absoluter Fehler zwischen den annotierten 2D Labeln und den Prädiktionen der verwendeten *DenseNet* Architektur. Der MPJPE wird für 2D Schätzungen in Pixeln und für 3D Schätzungen in Millimetern angegeben. Der MPJPE ist ein in der Literatur verbreitetes Gütemaß für Skelett-Tracking Algorithmen [11] und erlaubt somit den Vergleich der vorgestellten Methode mit anderen Verfahren.
- Percentage of Correct Keypoints (PCK): Die Prädiktion eines Gelenks gilt als korrekt klassifiziert, falls die geschätzte Position innerhalb eines Radius von $r = \alpha \cdot \max(h,w)$ der Ground Truth Position liegt. Die Parameter $h$ und $w$ bezeichnen dabei die Höhe und Weite einer Bounding Box um das jeweilige Gelenk. Der Faktor $\alpha$ definiert die relative Schwelle, bis zu der die Klassifikation als korrekt gilt. Analog zu [15] wird $\alpha$ für die Berechnung der PCK zu 0.2 gewählt. Um die Detektion

Tabelle 1: Gütemaße der 2D Körperposenschätzung

| Gelenk | MPJPE [px] | PCK [%] |
|---|---|---|
| Hand links | 7,89 | 68,14 |
| Hand rechts | 7,44 | 71,49 |
| Ellenbogen links | 3,76 | 92,20 |
| Ellenbogen rechts | 4,90 | 97,39 |
| Schulter links | 3,16 | 95,70 |
| Schulter rechts | 3,57 | 96,18 |
| Kopf links | 6,13 | 84,72 |
| Kopf rechts | 5,63 | 87,32 |
| Mittelwert | 5,31 | 85,34 |

von Bounding Boxen um die Gelenke zu vermeiden, wird $h = w = 40$ Pixel gewählt, was der Größe des Kopfes der Person entspricht. Somit wird eine Prädiktion als korrekt klassifiziert, falls sie innerhalb eines Radius von acht Pixeln um die Ground Truth Position des jeweiligen Gelenks liegt.

## 4.1 Evaluierung der Körperposenschätzung in Bildkoordinaten

Die Gütemaße MPJPE und PCK der 2D Körperposenschätzung sind in Tabelle 1 dargestellt. Die Berechnung der Gütemaße beruht auf der Evaluation von 4096 Bildern zweier Probanden. Die Aufnahmen dieser Probanden sind nicht in den Trainingsdaten enthalten, um die Invarianz des Verfahrens gegenüber Geschlecht und Körperform der Probanden zu belegen.

Zuverlässig detektiert werden insbesondere die Ellenbogen und die Schultern, die jeweils mittlere Fehler von unter fünf Pixeln pro Schätzung aufweisen. Die Prädiktionen dieser Gelenkpositionen liegen zu teilweise deutlich über 90 % innerhalb des Suchradius von acht Pixeln um die korrekten Werte. Deutlich schlechter erkannt wird die Position der Hände, die lediglich zu ca. 70 % korrekt detektiert werden.

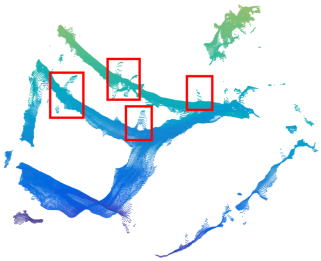


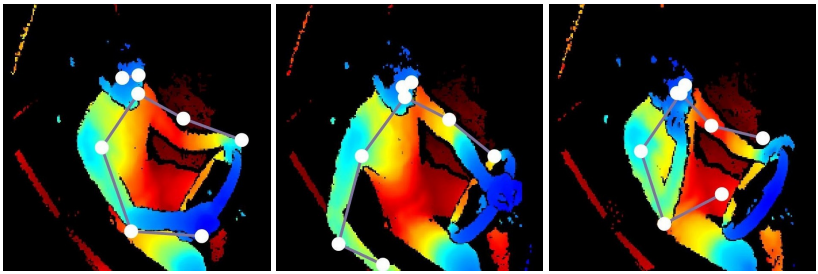
Bild 10: Rauschen aufgrund von reflektierenden Markern an den Gelenken

Die Methode weist eine vergleichbare Genauigkeit mit Werten aus der Literatur auf. [2] berichtet von einer mittleren PCK von 53.9 % für Bilder von Silhouetten, detektiert jedoch auch die Gelenke des kompletten Körpers. [5] erreicht eine PCK von ca. 90 % für Ellenbogen und ca. 81 % für Handgelenke. Die Convolutional Pose Machines [6] klassifizieren Ellenbogen zu 97.59 % und Handgelenke zu 95.03 % korrekt.

## 4.2 Qualitative Evaluierung der Posenschätzung ohne Marker

Die reflektierenden Marker des Motion-Capture-Systems führen zu deutlichen Störungen bei der Aufnahme der Tiefenbilder (siehe Abbildung 10), da sowohl die Kamera als auch das Motion-Capture-System Infrarot-basiert arbeiten. Um zu überprüfen, ob das Netz zur Posenschätzung auf diese Marker reagiert, wird eine Bewertung der 2D Posenschätzung ohne die Marker durchgeführt. Aufgrund der somit fehlenden Referenzwerte für die Gelenkpositionen ist lediglich eine qualitative Bewertung möglich.

In Abbildung 11 sind mehrere geschätzte Körperposen und die zugehörigen Tiefenbilder abgebildet, die ohne Motion-Capture Marker aufgezeichnet wurden. Abbildung 11a zeigt die typische Pose eines Fahrers, der die Fahraufgabe ausführt. Die Pose wird größtenteils korrekt geschätzt, lediglich die Position des linken Ellenbogens wird etwas zu weit oben angenommen. Fehlschätzungen sind in beiden anderen Abbildungen zu sehen. In Abbildung 11b wird die Position der linken Hand an einer falschen Stelle des Lenkrads geschätzt. Ab-



(a) (b) (c)

Bild 11: Qualitative Evaluierung der Posenschätzung

bildung 11c zeigt, wie die Position der rechten Hand bei einem Telefonat nicht korrekt zugeordnet wird.

Fehlende Marker verschlechtern die Genauigkeit der Posenschätzung, was darauf zurückzuführen ist, dass das CNN teilweise auf die durch die reflektierenden Marker entstehenden Features in den Tiefenbildern reagiert. Die Schätzung beruht jedoch nicht ausschließlich auf diesen Features, wodurch die grundsätzliche Schätzung der 2D-Körperpose dennoch möglich ist.

## 4.3 Evaluierung der räumlichen Körperposenschätzung

Dieser Abschnitt bewertet das auf der 2D-Posenschätzung aufbauende komplette Verfahren für die räumliche Körperposenschätzung. Die Resultate für den MPJPE und die PCK sind in Tabelle 2 für eine Sequenz von 500 Bildern dargestellt. Für die PCK werden Schätzungen als korrekt klassifiziert, falls sie innerhalb einer Sphäre mit einem Radius von 100 mm um die Referenzpositionen liegen. Es wird deutlich, dass auch bei der räumlichen Positionsschätzung die Schultern sehr gut detektiert werden. Der Grund liegt in der verglichen mit anderen Gelenken geringeren Varianz der Schulterpositionen. Die übrigen Gelenke werden deutlich schlechter erkannt. Der Grund für diese Abweichungen der Schätzung liegt unter Anderem in den durch die Marker vom Motion-Capture-System an den Gelenken stark verrauschten Tiefenbildern (siehe Abschnitt 4.2).

Tabelle 2: Gütemaße der 3D Körperposenschätzung

| Gelenk | MPJPE [mm] | PCK [%] |
|---|---|---|
| Hand links | 159,26 | 34,6 |
| Hand rechts | 266,50 | 16,0 |
| Ellenbogen links | 109,16 | 70,2 |
| Ellenbogen rechts | 271,10 | 16,4 |
| Schulter links | 45,26 | 100 |
| Schulter rechts | 79,44 | 92,2 |
| Kopf links | 228,01 | 5,2 |
| Kopf rechts | 140,87 | 6,2 |
| Mittelwert | 162,45 | 42,6 |

Vergleichswerte aus der Literatur liegen für den MPJPE bei durchschnittlich 85.64 mm [8] bzw. bei durchschnittlich 162.20 mm [11]. Der in der *Microsoft Kinect* verwendete Ansatz erreicht eine PCK von durchschnittlich ca. 90.8 % [2].

## 5 Zusammenfassung und Ausblick

Dieser Beitrag stellt einen zweistufigen Ansatz zum Skelett-Tracking von Autofahrern anhand von Tiefenbildern vor, der zunächst die Gelenkpositionen im Tiefenbild detektiert und anschließend die räumliche Körperpose schätzt. Der Ansatz beruht auf einer Kombination von Deep Learning und numerischer Optimierung.

Für das Training der neuronalen Netze dienen mit einem Motion-Capture System aufgenommene Referenzdaten mehrerer Probanden. Um die Tiefenbilder mit den korrekten Gelenkpositionen zu annotieren, wird die Transformation in das Kamerakoordinatensystem bestimmt und so optimiert, dass die Gelenkpositionen stets auf einem Punkt in der Punktwolke des Fahrers liegen.

Die Referenzdaten werden für das Training eines auf der *DenseNet* Architektur [3] basierenden neuronalen Netzes verwendet, das die Gelenkpositionen in Bildkoordinaten schätzt. Eine initiale Schätzung der Gelenkpositionen im Raum ergibt sich aus den Tiefendaten. Die plausible und gefilterte Schät-

zung des Skeletts beruht auf einer nachgeschalteten numerischen Optimierung, um nicht plausible Entfernungen zwischen den Gelenken und Diskontinuitäten zwischen zeitlich aufeinander folgenden Posen zu vermeiden.

Die Ergebnisse zeigen für die 2D Schätzung eine Genauigkeit, die mit Werten aus der Literatur vergleichbar ist. Die Genauigkeit der Schätzung der räumlichen Körperpose fällt hinter dem Stand der Technik aus der Literatur zurück. Eine Verbesserung wird durch die Verwendung des neuen Tiefenkameramodells Intel RealSense D435 erwartet. Die Wahl einer anderen Struktur des neuronalen Netzes für die 3D-EDM Schätzung, beispielsweise mithilfe einer CNN Architektur anstelle eines einfachen Fully Connected Netzes, bietet Potential zur Verbesserung der Genauigkeit.

Zukünftig erlaubt es die Schätzung des motorischen Fahrerzustands die Bereitschaft und Fähigkeit des Fahrers zu evaluieren, um bei einer Unterbrechung der automatisierten Fahrt die Übergabe der Fahraufgabe möglichst komfortabel und sicher zu gestalten. In diesem Zuge ist die Klassifizierung der jeweiligen Aktivität des Fahrers aus seiner Körperpose ein Ansatz für die Bewertung der Fahrerverfügbarkeit.

## Förderung

Dieser Beitrag entstand im Zuge des Forschungsprojekts „MoFFa - Holistisches Modell zur Beschreibung der Aufgabenverteilung und der Aufgabenübergabe zwischen menschlichem Fahrer und Fahrerassistenzsystem beim automatisierten und vernetzten Fahren“, das durch das Bundesministerium für Verkehr und digitale Infrastruktur (BMVI) im Rahmen des Forschungsprogramms „Automatisiertes und vernetztes Fahren“ unter dem Kennzeichen FKZ 16AVF2005 gefördert wird.

Bundesministerium
für Verkehr und
digitale Infrastruktur

## Literatur


[1] C. Marberger, H. Mielenz, F. Naujoks, J. Radlmayr, K. Bengler und B. Wandtner. „Understanding and applying the concept of 'driver availability' in automated driving“. In: International Conference on Applied Human Factors and Ergonomics, S. 595–605. 2017.

[2] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman und A. Blake. „Real-time human pose recognition in parts from single depth images“. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), S. 1297–1304. 2011.

[3] G. Huang, Z. Liu, L. v. d. Maaten und K. Q. Weinberger: „Densely Connected Convolutional Networks“. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), S. 2261–2269. 2017.

[4] M. Martin, F. Diederichs, K. Li, M. Voit, V. Melcher, H. Widlroither und R. Stiefelhagen: „Klassifikation von Fahrerzuständen und Nebentätigkeiten über Körperposen bei automatisierter Fahrt“. In: 32. VDI/VW Gemeinschaftstagung "Fahrerassistenzsysteme und automatisiertes Fahren". 2016.

[5] A. Toshev und C. Szegedy. „Deeppose: Human pose estimation via deep neural networks“. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), S. 1653–1660. 2014.

[6] S.E. Wei, V. Ramakrishna, T. Kanade und Y. Sheikh: „Convolutional pose machines“. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), S. 4724–4732. 2016.

[7] C. Wang, Y. Wang, Z. Lin, A.L. Yuille und W. Gao. „Robust estimation of 3d human poses from a single image“. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), S. 2361–2368. 2014.

[8] F. Moreno-Noguer: „3D human pose estimation from a single image via distance matrix regression“. In: Proceedings of the IEEE Conference

on Computer Vision and Pattern Recognition (CVPR), S. 1561–1570. 2017.

[9] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.P. Seidel, W. Xu, D. Casas und C. Theobalt. „Vnect: Real-time 3d human pose estimation with a single rgb camera“. In: ACM Transactions on Graphics (TOG), 36(4), 44. 2017.

[10] L. A. Schwarz, A. Mkhitaryan, D. Mateus und N. Navab:2D „Human skeleton tracking from depth data using geodesic distances and optical flow“. In: Image and Vision Computing, 30(3), 217–226. 2012.

[11] C. Ionescu, D. Papava, V. Olaru und C. Sminchisescu: „Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments“. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(7). 2014.

[12] M. Abadi et al. „Tensorflow: a system for large-scale machine learning“. In: OSDI. S. 265–283. 2016.

[13] Y. Chen und G. Medioni. „Object modelling by registration of multiple range images“. In: Image and vision computing, 10(3), S. 145–155. 1991.

[14] J. Dattorro: „Convex optimization and Euclidean distance geometry“. Meboo USA. 2010.

[15] Y. Yang und D. Ramanan: „Articulated human detection with flexible mixtures of parts“. In: IEEE transactions on pattern analysis and machine intelligence, 2013, 35(12), S. 2878–2890. 2013.

# Detecting Sensor Dependencies for Building Complementary Model Ensembles

Alexander Dockhorn, Rudolf Kruse

Fakultät für Informatik, Otto von Guericke Universität Magdeburg
Universitätsplatz 2, 39106 Magdeburg, Germany
E-Mail: {alexander.dockhorn, rudolf.kruse}@ovgu.de

## Abstract

Forward Model Approximation is a recently developed method for learning how to play unknown games. In this method game state transitions are analyzed with the target of predicting the result of the agent's actions. Instead of using a single complex model, complementary model ensembles can be used. This does not affect the accuracy in case each simple model focuses on independent components of the game. However, it is an open question how to retrieve these independent components without domain knowledge of the game being studied. In this work we explore the usage of bayesian belief network structure learning algorithms for the detection of sensor dependencies. Multiple structure learning algorithms are evaluated in a case-study involving two games of the GVGAI framework. Our results indicate that the analyzed algorithms are able to detect necessary dependencies for generating an approximated forward model.

## 1 Introduction

Popular training processes for artificial intelligence in games are based on either studying expert play or training an agent from scratch using reinforcement learning. Both method classes can achieve impressive results, but the learning process often converges slowly and results in a model that is hard to interpret (e.g., Deep Neural Networks).

In contrast, the Forward Model Approximation framework [6] learns to describe relevant components of an unknown game by learning either a complex model or a set of complementary models for the prediction of future states. Each of those sub-models focuses on describing the behavior of a single game component. Aggregating the results of all sub-models lets us predict future game states with higher accuracy than using a single model describing the behavior of all game components, while maintaining interpretability of each sub-model. Furthermore, the approximated forward model can be used to apply simulation-based search methods, such as Monte Carlo Tree Search [4], which proved to be among the top-performing algorithms in automated game-playing [7, 8, 13].

Previous attempts for building complementary model ensembles for Forward Model Approximation were based on exploiting the limitations of game description languages [9], such as the video game description language [14]. These description languages describe games as a set of distinct game components with limited interactions. On the one hand, knowing those limitations lets us create a conditionalized set of observations for the prediction of each distinct game component. This filtering of the data set's attributes reduces the amount of data for training each sub-model, speeds up the learning process, and generally assists in achieving a higher accuracy with the resulting aggregated model. On the other hand, knowing the limitations between variable interactions is not always ensured, which is why alternative methods for detecting relations between game components and available game observations need to be found.

In this work we study the application of bayesian belief network structure learning algorithms to detect dependencies among game objects and the available sensory information. These dependencies can further be used to create conditionalized databases for the model construction without the need of knowing the game's description. This will be the first step for creating an autonomous learning approach for modeling unknown games with the help of Forward Model Approximation.

In Section 2 we will briefly review general game playing. Section 3 presents a summary of the Forward Model Approximation framework and its benefits and drawbacks. To overcome the latter we study structure learning algorithms and their application to data that has been collected by studying games of the

GVGAI framework in Sections 4 and 5. In Section 6 we present the results of our case-study and a discussion of the usability of applied algorithms for detecting sensor dependencies. We conclude our work in Section 7 in which we shortly summarize implications of this case-study and provide an outlook for future work.

## 2 General Game Playing

General game playing poses the problem of creating an agent that is capable to learn how to successfully play multiple games. Here, games are represented as the combination of an interactive environment and a controllable agent that interacts with it based on a set of pre-defined actions. Each interaction can modify the environment and, therefore, result in a new state. After an interaction the agent receives a response in form of a numerical reward and the updated state.

The environment can be represented as a probability distribution:

$$Pr\{R_{t+1} = r, S_{t+1} = s' | S_0, A_0, R_1, ..., S_{t-1}, A_{t-1}, R_t, S_t, A_t\} \tag{1}$$

where $S_t$ represents the state of the environment at time step $t$, $A_t$ is the action that the agent chose at time $t$, and $R_{t+1}$ is the reward received as the environment's response. Thus, the future reward and the future state depend on all previous interactions between the environment and the agent. Hence, the complexity of this probability distribution grows exponentially over time.

For this reason, general game analysis is often restricted to environments that fulfill the Markov Property [18], in which the environment's response depends only on the current state and the chosen action. This reduces the probability distribution to:

$$Pr\{S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a\} \tag{2}$$

The success of an agent and its related policy $\pi$ can be measured by the return $G$. The return measures the accumulated reward till the end of an episode.

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T$$

If either the episodes are non-ending or the agent needs to value immediate rewards higher than rewards received at a later point of time, the discounted return is used.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{T} \gamma^k R_{t+k+1}$$

The parameter $\gamma \in [0,1]$ is the discount rate.

Reinforcement learning techniques try to estimate the expected return $q$ given a state-action pair and choose their action accordingly.

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi \left[ G_t | S_t = s, A_t = a \right] \\ &= \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right] \end{aligned} \tag{3}$$

For every game we search for an optimal policy $\pi$ that maximizes the expected return of the agent. Given the expected return, the optimal action can be determined by:

$$\pi'(s) = \arg\max_a q_\pi(s,a) \tag{4}$$

## 3 Forward Model Approximation

The General Video Game AI (GVGAI) competition [13] offers two tracks, which focus on different aspects of general game playing. In the single-player planning track agents can use a forward model to predict the outcome of an action sequence. Simulation-based search algorithms such as Monte Carlo Tree Search [4] use this forward model to choose an action sequence that maximizes their expected score. Agents of the single-player learning track cannot access such a model, but can learn from repeated interactions with the environment.

While simulation-based search algorithms show great performance in the planning track, they cannot be applied without a forward model. Therefore, the missing forward model led to multiple submitted agents using reinforcement learning techniques, which learn the value of state-action pairs by playing the game many times. Recently, we introduced Forward Model Approximation [6] as a different solution to the special requirements of the learning track. In Forward Model Approximation we model the changes of the observed game state after applying an action using either a single or a set of classifiers. In case the agent uses a single classifier it tries to predict the future state based on the information on the current state and the action to be applied. Because this approach can result in a very complex classifier, we can also split it into multiple sub-models, of which each of these classifiers predicts the behavior of a single game entity or changes of a single observable value.

Both approaches become feasible in case the Markov Property holds, whereupon, the game state does not depend on all previous interactions, but only on the previous game state and the applied action. The transition to the future state can then be modeled using either a deterministic or a probabilistic model. In contrast to reinforcement learning approaches, learning an approximated forward model can be done using few iterations, which can later be used during simulation-based search. In case the model predicts the game's future states good enough, similar performance to the single player planning track can be achieved.

However, learning the approximated forward model is a challenging task due to the availability of a lot of sensor values, which might or might not have an influence on the different components of the game. Additionally, special attention should be paid on efficiency of the model building process to ensure short learning times. In a previous work [9] we explained how a complex forward model can be split in multiple sub-models to achieve not only higher accuracy, but also reduce the time spent for model construction. This approach is often justified due to independent components of a game being studied. A solution for model splitting exploits specific restrictions of the video game definition language [14] to restrict the sub-model learning process [9].

Despite the advantages of this method, we see two major constraints:

1. **Assumptions on the framework:** The split into multiple sub-models is based on the type definition of the video game definition language. This type definition lets the agent detect similar behaving entities using a type identifier. It cannot be assumed that this information is accessible in games outside of the GVGAI framework. Therefore, a natural extension would be to detect similar entities based on their visual representation or their behavior at run-time.
2. **Limitation to local influence factors:** Each of the created sub-models assumes independence of global influence factors, such that only entities close to the modeled entity are taken into account during the model building process. This is sufficient to detect interactions between neighboring entities. However, it is not able to model interactions between entities that are farther away, e.g., a switch opening a door which is far away cannot be represented.

In this work we want to study methods for automatizing the filtering of data sets to remove these assumptions.

## 4 Structure Learning of Bayesian Belief Nets

Studying the dependence of variables among each other can be implemented by learning the structure of a bayesian belief net using the observed data set. A bayesian belief net is a directed acyclic graph, denoted by $\mathscr{G}$, with parameters $\Theta$. Structure learning approaches try to find the DAG $\mathscr{G}$ that encodes the dependence structure of the data (see [2, 11] for a general description of bayesian belief networks). We differentiate three categories of structure learning algorithms, namely score-based, constraint-based, and hybrid approaches.

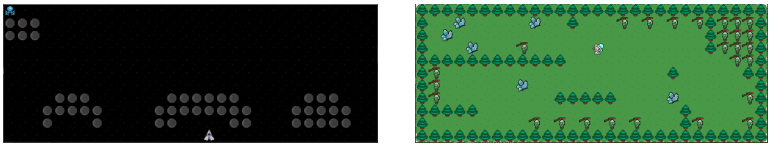
Scoring based approaches choose the best performing belief net structure over a set of candidate structures by measuring the goodness of fit between the structures implied probability distribution and the observed data. However, finding a belief net structure over a given set of variables using exhaustive search is super-exponential in the number of variables. Due to the high number of sensors and their variability of values in games of the GVGAI framework, this

approach is infeasible. For this reason, local search algorithms restrict the search in each iteration to a subspace of network structures. For example the Hill Climbing (*hc*) algorithm [5, 10] modifies the current structure by either adding, removing, or reversing an edge. As a result of each possible modification the given graph structure is rated using a network score, such as the Bayesian Information Criterion [15]. The greedy *hc* algorithm is known to get trapped in local optima. Tabu-Search (*tabu*) [3] maintains a tabu list of previously performed modifications to avoid getting stuck.

Constraint-based approaches use conditional independence tests to learn the dependence structure of the attributes given the observed sample. One example is the Grow-Shrink (*gs*) algorithm [12], which first uses markov blankets to fix an undirected structure of the graph and then orients edges, removes existing cycles, and propagates directions to yet undirected edges. The semi-interleaved Hiton Parents and Children (*si-hiton-pc*) algorithm [1] learns local causal and markov blanket relationships. It is especially suited for large attribute sets with only limited data available. Another algorithm, which was developed with large attribute sets in mind, is the Max-Min Parents and Children (*mmpc*) algorithm [19]. Similar to the other two algorithms *mmpc* creates an undirected graph structure by detecting all possible parents and children per node.

Hybrid algorithms combine a restriction and a maximization phase. During the restriction phase the search space is reduced to a subset of possible DAG structures, e.g., by searching for all pairs of connected nodes. In the maximization phase a score-based algorithm is used to direct the edges of the determined graph structure. The Max-Min Hill-Climbing (*mmhc*) algorithm [20] uses *mmpc* to restrict the network structure and uses the *hc* algorithm to find the best configuration for all edge directions. A more general framework is the 2-phase Restricted Maximization (*rsmax2*), which lets the user choose the settings for both phases independently [16, 17].

In this work we will test the algorithms mentioned above to learn the structure of a belief net on the basis of all observable variables in the GVGAI framework. Using this approach we want to identify dependent and independent components in the belief net structure. While independent components help us to reduce the complexity of sub-models in the approximated forward model, dependent components point out interactions between variables that should not



(a) the *aliens* game (b) the *butterflies* game

Figure 1: GVGAI games used for evaluation

be ignored. These interactions can either be included in a sub-model building process or need to be added during the aggregation process.

## 5 Building a Data Set for GVGAI Games

We evaluate our process using two exemplary games of the GVGAI competition to compare the results of different structure learning algorithms.

In our first game *aliens* (see Figure 1a) the player steers a small space-ship at the bottom of the screen. In the boundaries of the screen it can either move left or right. The spaceship can also shoot to create a bullet over the player's avatar, which is moving to the top of the screen, where it will be destroyed. Only one bullet can exist at a time. At the top of the screen multiple alien spaceships will move from left to right. In case an alien reaches the end of its row, instead of leaving the screen it will move down one row and change its flying direction. This process repeats until it reaches the players row in which case the player loses the game by touching the alien. This can only be avoided by shooting the alien spaceships before they reach the bottom row. Further, multiple boulders can be destroyed by shooting them. The player wins the game in case all aliens are destroyed, but loses if its own spaceship is hit by an alien.

The second game *butterflies* (see Figure 1b) tasks the player to catch all the butterflies flying around. These butterflies fly at random and can free more butterflies in case they are touching one of the cocoons. The player loses the game once all cocoons were destroyed. This can only be avoided by catching all the butterflies before they destroy all the cocoons.

Table 1: Framework specific sensor values stored during the play session.

| Player Sensors | NPC Sensors | Game State Sensors |
|---|---|---|
| X and Y pos | X and Y pos | game tick |
| X and Y grid pos | X and Y grid pos | destroyed/created instances of type x |
| X and Y pos change | X and Y pos change | score change |
| 8-directional neighborhood | 8-directional neighborhood | win/lose/continuing |
| selected action | previous X and Y position change | |
| | delay since last movement | |

Both games were played by a random agent not knowing any of the game's mechanics. These games were selected, because they are two of the few games a random agent might be able to win. During the play sessions we stored the sensor values listed in Table 1.

## 6 Evaluation of Structure Learning Algorithms

For the evaluation we created a data set of playthroughs using the 2017 Java version of the GVGAI framework. Our evaluation of structure learning algorithms was performed using the *bnlearn* R-package [16]. During the preprocessing phase we nominalized all attributes even if they are on a numerical scale. This was done to ensure that all algorithms can be applied equally. The data set consists of the listed attributes (see Table 1) for every observable game element (the player and all non-player elements) as well as the general game state information for every game tick. In case an element cannot be observed during a game tick the value of all its attributes is set to *NA*.

Depending on the number of non-player elements (like butterflies or alien spaceships) the dataset contains more than 500 attributes. Attributes that only depict a single value over the time of a complete playthrough are removed during the preprocessing phase. We additionally deleted the observations of all objects that never change their position (X change, Y Change), are present from

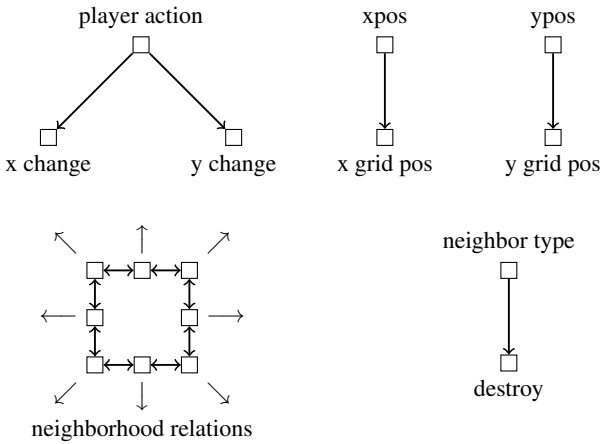



Figure 2: Retrieved movement specific dependence structures; (top left) the player's action and movement; (top right) position and grid position; (bottom left) neighborhood relations; (bottom right) threat of being destroyed by neighboring object

the beginning of the game, and are never removed during the playthrough. This was done to exclude background objects, which are observed and reported, but do never influence any of the game's components. All remaining attributes are used for learning the structure of a bayesian belief net.

We qualitatively compare the structure learning algorithms *hc, tabu* (score-based), *gc, mmpc, si-hiton-pc* (constraints-based), and *rsmax2* for all combinations of mentioned score-based and constraint-based algorithms. Interesting dependency structures are shown in Figure 2. They highlight game specific dependencies that proved useful in previously generated approximated forward models, such as modeling the player's movement, understanding the dependency of an object's neighborhood sensors, and predicting the destruction of an object. The following sections present the results of processing data of the *aliens* and *butterflies* game.

Table 2: Summary of learned structures for the game; #E: number of edges, #V: number of vertices, #C: number of connected components; *the algorithms *gs*, *mmpc* and *si-hiton-pc* return an undirected graph, the listed number of edges is the number of undirected edges

| algorithm | *alien* | | | *butterflies* | | |
|---|---|---|---|---|---|---|
| | #E | #V | #C | #E | #V | #C |
| *hc* | 361 | 370 | 9 | 339 | 318 | 2 |
| *tabu* | 361 | 370 | 9 | 343 | 318 | 1 |
| *gs* | 13* | 26 | 13 | 28* | 57 | 27 |
| *mmpc* | 21* | 42 | 21 | 40* | 76 | 36 |
| *si-hiton-pc* | 93* | 184 | 91 | 148* | 237 | 95 |
| *rsmax2 (gs, hc/tabu)* | 6 | 12 | 6 | 30 | 57 | 27 |
| *rsmax2 (mmpc, hc/tabu)* | 14 | 28 | 14 | 40 | 76 | 36 |
| *rsmax2 (si-hiton-pc, hc/tabu)* | 92 | 182 | 90 | 139 | 230 | 94 |

## 6.1 Evaluation of the Game *Aliens*

The analysis of the game *aliens* is based on a single random playthrough, which consists of 715 observed game ticks and a total of 405 unfiltered attributes. Table 2 summarizes the resulting graph structures of applied structure learning algorithms.

The *hc* and the *tabu* algorithm return the same graph, which nearly includes all attributes and only a few components. Its biggest component consists of 293 vertices and includes movement, neighborhood, and destroy relations of nearly all aliens and alien shots. Even if unpractical, such a single big component is justified by the simultaneous movement of all alien objects. For this reason, the algorithm cannot detect that these objects act independently from each other. One of the smaller components including only 6 vertices represents the player's movement, which in this game is independent of other game objects. Remaining components include all attributes related to a single shot object, which did not collide with any other game object. For this reason, the observed shot was independent of all other observed game objects.

The *gs*, *mmpc*, *rsmax2(gs,hc/tabu)*, and *rsmax2(mmpc,hc/tabu)* algorithms all fail in detecting most attribute dependencies. Their resulting graphs only con-

sist of a few edges correctly connecting the position and the grid position attributes of some instances. Additionally the result of the *rsmax2* algorithm only depends on the chosen restriction algorithm, as switching from *hc* to *tabu* did not change the resulting graph.

Applying the *si-hiton-pc* and *rsmax2(si-hiton-pc, hc/tabu)* returns a graph with many small communities, which represent characteristics of the game, but do not include all related attributes. These communities include parts of the neighborhood relation or movement based attribute interactions, but overall failed to detect necessary dependencies.

## 6.2 Evaluation of the Game *Butterflies*

In the second part of our case study we analyze the *butterflies* game. Our data set is also based on a single random playthrough, which consists of 667 observed game ticks and a total of 334 unfiltered attributes. Table 2 summarizes the resulting graph structures of applied structure learning algorithms.

As it was the case in the previous game, both the *hc* and the *tabu* algorithm return nearly similar graphs. The resulting graph structure includes all game object dependencies in a single component. In the case of the *butterflies* game, this is overly specific, since many of the game's objects act independent from each other. However, in case a butterfly comes into contact with a cocoon, it will destroy it and spawn a new butterfly. This interaction can be found in a specific playthrough, but might not appear in the next one. Currently, it is not possible to relate sensor values from differing playthrough, such that it is impossible to render interacting objects independent from each other.

The *gs* and *mmpc* algorithms as well as their *rsmax2* counterparts tend to form very small groups of attributes. Once again the output of the *rsmax2* algorithm only depends on the chosen restriction phase. The resulting graphs include parts of the neighborhood dependencies. Movement and destroy interactions were not included.

*si-hiton-pc*, *rsmax2(si-hiton-pc, hc)*, and *rsmax2(si-hiton-pc, tabu)* all result in similar graph structures including 95 connected components. These components model neighborhood interactions, but rarely connect attributes of differing objects.

## 6.3 Discussion

The results of our two case studies showed that the applied algorithms largely vary in the resulting graph structures. Generally, the constraint based algorithms *gs* and *mmpc* and their hybrid counterparts created very sparse graph structures. The generated structures map important attribute dependencies of the game, but do so very unreliably. Therefore, they are unsuited for a general framework for detecting object dependencies.

The tested score-based approaches (*hc* and *tabu*) as well as the constraint-based *si-hiton-pc* algorithm detected dependence structures on a reliable basis. The *hc* and *tabu* algorithms result in graphs with higher density, due to connections in between attributes of different game objects. These connected game objects interact only very rarely, therefore, learning a model on the returned graph structure may be over-specific to the single observed playthrough. In return, the *si-hiton-pc* algorithm and the *rsmax2* algorithm using *si-hiton-pc* during its restriction phase both detect most necessary attribute dependencies for modeling the behavior of the games' objects. However, inter-object attribute dependencies are not taken into account.

Using the generated graph structure of *si-hiton-pc* for the generation of an approximated forward model will result in a large number of simple models. The results of these models need to be aggregated and will rarely result in mistakes in case of interactions between multiple objects exist. Higher accuracy can be achieved by using the graph structures of the *hc* or the *tabu* algorithm. Those models will be able to achieve a higher accuracy at the cost of a higher complexity.

# 7 Conclusion and Future Work

In this work we studied the suitability of belief net structure induction algorithms for detecting object dependencies in games of the GVGAI framework. Out of the tested algorithms score-based algorithms such as *hc* and *tabu* seem to detect the most object dependencies. Even rare interactions between objects can be modeled using attribute sets of the resulting graph structures. Applying the *si-hiton-pc* or *rsmax2(si-hiton-pc, hc/tabu)* algorithm results in graphs with reduced density. Forward Model Approximation might benefit from this by a reduced complexity for each sub-model, which may also reduce the model's accuracy.

Our small case study showed the applicability of belief net structure induction algorithms for an automated analysis of dependency structures. Forward Model Approximation will benefit from this by allowing a divide of the agent's sensor values into independent sets. For each of these sets a simple model can be trained. Automatizing the division into multiple sets of independent sensor values makes it possible to apply Forward Model Approximation without information about the game's attributes' dependency structure.

One of the next steps will be to incorporate the structure learning algorithm to Forward Model Approximation. Thus, creating a unified framework for the analysis of unknown games without further domain knowledge. For this purpose, we will need to study the generalizability of the studied structure learning algorithms. Extending our case-study will provide further insights in characteristics of games and how they influence the applicability of varying structure learning algorithms.

## References


[1] Constantin F Aliferis, Alexander Statnikov, Subramani Mani, and Xenofon D Koutsoukos. Local Causal and Markov Blanket Induction for Causal Discovery and Feature Selection for Classification Part I: Algorithms and Empirical Evaluation Ioannis Tsamardinos. *Journal of Machine Learning Research*, 11(March):171–234, 2010.

[2] Christian Borgelt, Matthias Steinbrecher, and Rudolf Kruse. *Graphical Models*. John Wiley & Sons, Ltd, Chichester, UK, August 2009.

[3] R.R. Bouckaert. *Bayesian Belief Networks: from Construction to Inference*. PhD thesis, Utrecht, Netherlands, 1995.

[4] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, March 2012.

[5] Gregory F Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, October 1992.

[6] Alexander Dockhorn and Daan Apeldoorn. Forward Model Approximation for General Video Game Learning. In *2018 IEEE Conference on Computational Intelligence and Games, CIG 2018*, 2018.

[7] Alexander Dockhorn, Christoph Doell, Matthias Hewelt, and Rudolf Kruse. A decision heuristic for Monte Carlo tree search doppelkopf agents. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, November 2017.

[8] Alexander Dockhorn, Max Frick, Ünal Akkaya, and Rudolf Kruse. Predicting opponent moves for improving hearthstone ai. In Jesús Medina, Manuel Ojeda-Aciego, José Luis Verdegay, David A. Pelta, Inma P. Cabrera, Bernadette Bouchon-Meunier, and Ronald R. Yager, editors, *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations*, pages 621–632, Cham, 2018. Springer International Publishing.

[9] Alexander Dockhorn, Tim Tippelt, and Rudolf Kruse. Model Decomposition for Forward Model Approximation. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018.

[10] José A. Gámez, Juan L. Mateo, and José M. Puerta. Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood. *Data Mining and Knowledge Discovery*, 22(1-2):106–148, January 2011.

[11] Rudolf Kruse, Christian Borgelt, Christian Braune, Sanaz Mostaghim, and Matthias Steinbrecher. *Computational Intelligence*. Texts in Computer Science. Springer London, London, 2nd edition, 2016.

[12] Dimitris Margaritis, Sebastian Thrun, Christos Faloutsos, Andrew W Moore, and Gregory F Cooper. *Learning Bayesian Network Model Structure from Data*. PhD thesis, Carnegie Mellon University, 2003.

[13] Diego Perez-Liebana, Spyridon Samothrakis, Julian Togelius, Tom Schaul, Simon M. Lucas, Adrien Couetoux, Jerry Lee, Chong U. Lim, and Tommy Thompson. The 2014 General Video Game Playing Competition. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(3):229–243, 2016.

[14] Tom Schaul. An extensible description language for video games. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(4):325–331, 2014.

[15] Gideon Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461–464, March 1978.

[16] Marco Scutari. Learning bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22, 2010.

[17] Marco Scutari and Jean-Baptiste Denis. *Bayesian Networks with Examples in R*. Chapman and Hall, Boca Raton, 2014. ISBN 978-1-4822-2558-7, 978-1-4822-2560-0.

[18] Publisher Taylor. The Oxford Dictionary of Statistical Terms. *Technometrics*, 46(2):266–266, May 2004.

[19] Ioannis Tsamardinos, Constantin F. Aliferis, and Alexander Statnikov. Time and sample efficient discovery of Markov blankets and direct cau-

sal relations. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '03*, page 673, New York, New York, USA, 2003. ACM Press.

[20] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

# Local Gaussian Process Model Networks

Tim Decker, Oliver Nelles

Universität Siegen, Department Maschinenbau, Institut für Mechanik und Regelungstechnik – Mechatronik
Paul-Bonatz-Str. 9-11, 57068, Siegen, Germany
E-Mail: {tim.decker,oliver.nelles}@uni-siegen.de

## 1 Introduction

Modeling nonlinear processes can be a difficult task, especially when the number of input dimensions is large. In this contribution, local Gaussian process model networks (LGPMNs), are proposed. This model structure aims to combine the advantages of local model networks and Gaussian processes (GPs) for identification of nonlinear static functions.

Identification of local model networks with tree-structured algorithms, like LOLIMOT, is a robust and powerful approach, in particular due to the individual estimation of each local model [1]. These algorithms rely on the partitioning of the input space and thus depend strongly on the dimensionality of the input. One limitation is that linear local models have a low flexibility within their validity region.

In contrast to that, estimating GPs depends less severely on the dimensionality of the input space. GPs are kernel-based, powerful and smooth nonlinear approximators. The hyperparameters inside the kernel function of a GP reflect properties of the assumed underlying process, such as the measurement noise or the lengthscale which can be different for every input dimension. These hyperparameters enable a GP to adapt the kernel to the global behavior of the process. Since GPs are Bayesian models [8] it is also possible to calculate an uncertainty interval. This gives insights to the regional reliability of the model. Problems with GPs arise when the amount of data is large because the computational effort is $\mathcal{O}(N^3)$ with data size $N$. Another issue is that hyperpa-

rameters are valid for the global model structure and cannot adapt to specific local behaviors.

To circumvent these drawbacks and to unite the advantages of both approaches, several attempts have been made. In [9] an algorithm for training a mixed Gaussian experts network is introduced. This algorithm is very powerful but requires a lot of computation. It is based on [5]. Several models are used to build a global model. Each model contributes the most to the global output where it performs the best. One problem that arises when trying to blend local Gaussian processes is that discontinuities can arise at the boundaries between local models [10]. The DDM approach proposed in [6] focuses on this aspect by setting constraints at these boundaries that have to be fulfilled in the optimizations. This solves the problem of discontinuities but requires more computational effort and restricts the problem to be low-dimensional. In [7] an improved version of this can be found.

The LGPMNs proposed here, are less sensitive with respect to data size and number of dimensions. By using the normalized Gaussian validity functions together with a threshold of the required local model's minimum validity, local Gaussian process models are trained with small subsets of the whole data. This saves computational effort. With the assignment of local Gaussian process models to a local region, their hyperparameters can be optimized by maximizing the marginal likelihood (ML) or pseudo likelihood (PL) to adapt well to the processes behavior in these regions. This is necessary, when for example different sensors are used in different regions of the input space. In this case, LGPMNs can identify different process noise levels as individual hyperparameters. The applicability of the presented method is analyzed and demonstrated on artificial data sets.

# 2 Gaussian Process Regression and Local Model Networks

In this section the structure of Gaussian processes (GPs) and local model networks are given.

## 2.1 Gaussian Process Regression

This section explains how the parameters $\underline{\theta}$ that are used to predict the output

$$\underline{\hat{y}} = \underline{K}\,\underline{\theta} \tag{1}$$

are derived. Here $\underline{K}$ represents the kernel matrix containing the activities of all kernel functions at all data points.

Gaussian processes are Bayesian models [8]. In Gaussian process regression (GPR) it is assumed that the closer points are in the input space the more related their output should be. These similarities are described by the kernel matrix $\underline{K}$. Its entries

$$\underline{K}_{pq} = \sigma_f^2 e^{\left(-(\underline{u}(p)-\underline{u}(q))^T \underline{L}^{-2} (\underline{u}(p)-\underline{u}(q))\right)}; \quad \begin{matrix} p = 1,\ldots,N \\ q = 1,\ldots,N \end{matrix} \tag{2}$$

are based on the Euclidean distance between $\underline{u}(q)$ and $\underline{u}(p)$. For $N$ samples in a $P$-dimensional input space, the input values are stored in the matrix

$$\underline{U} = \begin{bmatrix} \underline{u}^T(1) \\ \underline{u}^T(2) \\ \vdots \\ \underline{u}^T(N) \end{bmatrix} \tag{3}$$

$$\underline{u}^T(1) = \begin{bmatrix} u_1(1) & u_2(1) & \ldots & u_P(1) \end{bmatrix}. \tag{4}$$

Training of the GPs depends on the hyperparameters $\sigma_f^2$ which represents the output variance of our model, $\sigma_n^2$ which is the measurement noise of the recorded data and the lengthscale matrix

$$\underline{L} = \begin{bmatrix} l_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & l_P \end{bmatrix} \tag{5}$$

of the kernel function with a lengthscale for every of the $P$ input dimensions. These hyperparameters are trained as explained later. Similar to the least squares procedure, in Gaussian process regression the parameters $\underline{\theta}$ are calculated. For every point of the training data a parameter is calculated. These parameters are used to predict an output of the model for a given input. To calculate the probability density of the parameters $\underline{\theta}$ for given input $\underline{U}$ and measured output $\underline{y}$ its equation

$$p\left(\underline{\theta}|\underline{y},\underline{U}\right) \propto p\left(\underline{y}|\underline{U},\underline{\theta}\right) \cdot p\left(\underline{\theta}|\underline{U}\right) \tag{6}$$

has to be set up according to Bayes theorem [3, 4, 2].

The likelihood function is assumed to be a normal distribution with a mean $\underline{K}\,\underline{\theta}$ and a measurement noise variance $\sigma_n^2\underline{I}$ for the measured outputs $\underline{y}$

$$p\left(\underline{y}|\underline{U},\underline{\theta}\right) \sim \mathcal{N}\left(\underline{K}\,\underline{\theta},\sigma_n^2\underline{I}\right) \tag{7}$$

$$p\left(\underline{y}|\underline{U},\underline{\theta}\right) \sim \frac{1}{\sqrt{2\pi\det\left(\sigma_n^2\underline{I}\right)}} \cdot e^{\left(-\frac{1}{2}\left(\underline{y}-\underline{K}\,\underline{\theta}\right)^T \frac{1}{\sigma_n^2}\left(\underline{y}-\underline{K}\,\underline{\theta}\right)\right)}. \tag{8}$$

Without any further information, the prior distribution of the parameters $\underline{\theta}$ is assumed to be a normal distribution with zero mean and a variance $\underline{K}^{-1}$

$$p\left(\underline{\theta}|\underline{U}\right) \sim \mathcal{N}\left(\underline{0},\underline{K}^{-1}\right) \tag{9}$$

$$p\left(\underline{\theta}|\underline{U}\right) \sim \frac{1}{\sqrt{2\pi\det\left(\underline{K}^{-1}\right)}} \cdot e^{\left(-\frac{1}{2}\underline{\theta}^T\underline{K}\,\underline{\theta}\right)}. \tag{10}$$

In order to find the parameters that explain the training data the best, the probability distribution

$$\max_{\underline{\theta}} p\left(\underline{\theta}|\underline{y},\underline{U}\right) \tag{11}$$

has to be maximized with respect to the parameters. By taking the partial derivative with respect to the parameters and setting it to zero

$$\frac{\partial}{\partial \underline{\theta}} \log p\left(\underline{\theta}|\underline{y},\underline{U}\right) = 0, \tag{12}$$

the optimal parameters

$$\underline{\theta}_{opt} = \left(\underline{K} + \sigma_n^2 \underline{I}\right)^{-1} \underline{y} \tag{13}$$

are obtained. By multiplying the parameters $\underline{\theta}_{opt}$ with the kernel matrix $\underline{K}$, the mean of the predicted output can be calculated (1). Since the mean is the most likely value, this is considered to be the output of the GP. In addition to that the expected variance of the output can be calculated at every point in the input space. For further information on this see [8].

## 2.2 Local Model Networks

The partitioning procedure of the LOLIMOT algorithm has proven to be very powerful. It is a heuristic tree-structure algorithm. This algorithm creates partitions for local models (LMs) in the input space and successively divides regions of the worst performing LM into two regions for new LMs that substitute the worst performing one. Partitioning is done by placing the centers of Gaussians into the centers of the assigned region. By normalizing these Gaussians after every split, the validity functions $\Phi_m$ are derived. Therefore at every point in the input space all validity functions add up to 1 as can be seen in Fig. 2. In LOLIMOT every local model is trained with the whole training data set of size $N$. The output is calculated by summing up the weighted outputs of all LMs. The weighting is done with the respective validity function as can be seen in Fig. 1.

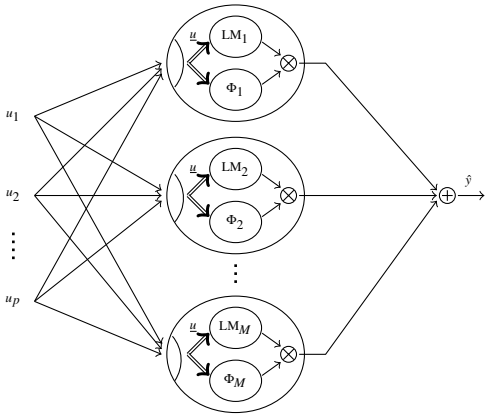



Figure 1: Local Gaussian process network with input $u$, local models (LM), validity functions $\Phi$ and global model output $\hat{y}$

# 3 Local Gaussian Process Model Networks

In order to combine the advantages of local model networks and Gaussian process regression, this section proposes three different methods for estimating the parameters $\underline{\theta}$ of the local models. Since the parameters depend heavily on the values of the hyperparameters, two approaches of optimizing these hyperparameters are presented as well.

In order to keep it simple in all test cases only two LMs are used for a one-dimensional input. Thus, for each local model $m$ an individual scalar lengthscale $l_m$ has to be determined. Their validity functions are normalized Gaussians as used in LOLIMOT. In addition to the validity functions $\Phi_m$, a threshold $t$ is defined. The training data of a local model is defined as the subset of the whole training data, where the value of the respective validity function is larger or equal than the threshold $t$. Figure 2 shows the two validity functions $\Phi_1$ and $\Phi_2$ that are used for every training in this contribution as well as the two thresholds $t_1 = 0.05$ and $t_2 = 0.5$. The threshold $t_1$ assigns 2/3 of the data to each local model for training. Therefore 1/3 of the data is used to train both models. With the threshold $t_2$ each LM receives 1/2 of the whole training data and no data is used twice in the model training.

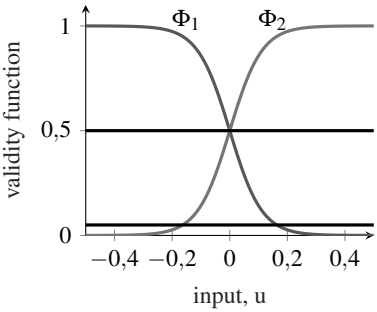



Figure 2: Validity functions $\Phi_1$ for $\mathrm{LM}_1$ and $\Phi_2$ for $\mathrm{LM}_2$ with thresholds $t_1 = 0.05$ and $t_2 = 0.5$

## 3.1 Method 1 – Unweighted Regression

This method is called blended Gaussian process regression. It relies on the assumption that all data points assigned to a local model are of the same importance. The error

$$\underline{e} = \underline{y} - \underline{\hat{y}} \tag{14}$$

is assumed to be normally distributed with a zero mean and variance $\sigma_n^2 \underline{I}$

$$\underline{e} \sim \mathcal{N}\left(\underline{0}, \sigma_n^2 \underline{I}\right). \tag{15}$$

The validity function is relevant for data selection and blending of the models.

The local model parameters

$$\underline{\theta}_m = \left(\underline{K}_m^{(t)} + \sigma_{n_m}^2 \underline{I}\right)^{-1} \underline{y} \tag{16}$$

are estimated with their subset of training data and Gaussian process regression [8]. It can be seen that the noise variance $\sigma_{n_m}^2$ is regularizing the parameters.

Properties for training like the kernel matrix entries

$$\underline{K}_{m_{rq}}^{(t)} = \sigma_{f_m}^2 e^{\left(-\left(\underline{u}_m^{(t)}(r)-\underline{u}_m^{(t)}(r)\right)^T \underline{L}_m^{-2}\left(\underline{u}_m^{(t)}(r)-\underline{u}_m^{(t)}(q)\right)\right)} \quad \begin{matrix} r=1,\ldots,N_m^{(t)} \\ q=1,\ldots,N_m^{(t)} \end{matrix} \tag{17}$$

are denoted with a "t" in the superscript, while properties for prediction like the prediction kernel matrix

$$\underline{K}_{m_{rq}}^{(p)} = \sigma_{f_m}^2 e^{\left(-\left(\underline{u}_m^{(t)}(r)-\underline{u}_m^{(p)}(q)\right)^T \underline{L}_m^{-2}\left(\underline{u}_m^{(t)}(r)-\underline{u}_m^{(p)}(q)\right)\right)} \quad \begin{matrix} r=1,\ldots,N_m^{(t)} \\ q=1,\ldots,N_m^{(p)} \end{matrix} \tag{18}$$

are denoted with a "p" in the superscript. The weighting matrix for predicted outputs

$$\underline{Q}_m^{(p)} = \begin{bmatrix} \Phi_m^{(p)}\left(\underline{u}_m^{(p)}(1)\right) & 0 & 0 & 0 \\ 0 & \Phi_m^{(p)}\left(\underline{u}_m^{(p)}(2)\right) & 0 & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \ldots & \Phi_m^{(p)}\left(\underline{u}_m^{(p)}\left(N_m^{(p)}\right)\right) \end{bmatrix} \tag{19}$$

consists of all weights of the validity function $\Phi_m^{(p)}$ on main the diagonal. The outputs of $\mathrm{LM}_m$

$$\underline{y}_m^{(p)} = \underline{K}_m^{(p)} \underline{\theta}_m \tag{20}$$

are weighted with their validity function and summed up to build the global output of the local model network

$$\underline{\hat{y}} = \sum_{m=1}^{M} \underline{y}_m^{(p)} \cdot \underline{Q}_m^{(p)}. \tag{21}$$

The specific kernel matrix of this method

$$\underline{\underline{K}}^{(1)} = \underline{\underline{K}}_m^{(t)} + \sigma_{n_m}^2 \underline{\underline{I}} \tag{22}$$

is important for the optimization of hyperparameters.

## 3.2 Method 2 – Weighted Variance Regression

This method is called blended weighted Gaussian process regression. The main assumption for this method is that points with a lower validity function provide less certain information for estimating a local model. So the noise variance of each point is multiplied with the inverse of the validity functions value at this point

$$\underline{e} \sim \mathcal{N}\left(\underline{0}, \sigma_n^2 \underline{\underline{Q}}^{-1}\right). \tag{23}$$

By assuming a higher variance at points with a low validity value the parameters at these points are stronger regularized. The resulting parameters

$$\underline{\theta}_m = \left(\sigma_{n_m}^2 \underline{\underline{I}} + \underline{\underline{Q}}_m^{(t)} \underline{\underline{K}}_m^{(t)}\right)^{-1} \underline{\underline{Q}}_m^{(t)} \underline{y}_m^{(t)} \tag{24}$$

are used as in the previous section to calculate the output of local models. Also the calculation of the global output does not differ. The method specific kernel matrix

$$\underline{\underline{K}}^{(2)} = K_m^{(t)} + \sigma_{n_m}^2 \left(\underline{\underline{Q}}_m^{(t)}\right)^{-1} \tag{25}$$

differs in weighting the measurement noise.

## 3.3 Method 3 – Weighted Kernel Function Regression

One main idea of Gaussian processes is that points, which are close to each other in the input space should produce outputs that are correlated. Therefore the kernel matrix is a measure of similarity between points. The idea of this method is, that points with a low validity value are less correlated with other

points and therefore should contribute less to the parameter estimation of other points. So in the calculation of the parameters

$$\underline{\theta}_m = \left( \underline{Q}_m^{(t)} K_m^{(t)} \underline{Q}_m^{(t)} + \sigma_{n_m}^2 \underline{I} \right)^{-1} \underline{y}_m^{(t)} \tag{26}$$

the kernel matrix is multiplied with the weights of the contributing inputs. This is done in the same manner for the local output prediction

$$\underline{y}_m^{(p)} = \underline{Q}_m^{(p)} K_m^{(t)} \underline{Q}_m^{(t)} \underline{\theta}_m \tag{27}$$

where it has to be noted that the weighting matrix for training data $\underline{Q}_m^{(t)}$ and the weighting matrix for prediction data $\underline{Q}_m^{(p)}$ are used. The method specific kernel matrix corresponds to

$$\underline{K}^{(3)} = \underline{Q}_m^{(t)} K_m^{(t)} \underline{Q}_m^{(t)} + \underline{I} \sigma_{n_m}^2. \tag{28}$$

## 3.4 Marginal Likelihood and Pseudo Likelihood Optimization

In the following $\underline{K}$ stands for the method specific kernel matrices from (22), (25) and (28). The marginal likelihood (ML) expresses the likelihood of the outputs $\underline{y}$ for present input data $\underline{U}$ and the hyperparameters

$$\underline{\eta}_m = \begin{bmatrix} \sigma_{n_m} \\ \sigma_{f_m} \\ l_m \end{bmatrix} \tag{29}$$

presuming that the model creation assumptions are correct. In the following explanations, indices for the $\mathrm{LM}_m$ and training case are omitted for simplicity. Both optimization methods depend solely on training data; for deeper insight and a discussion of the equations below see [8]. Gradients are given for the implementation of gradient descent algorithms to search for optimal hyperparameters.

The processes true output is denoted as $\underline{f}$ while $\underline{y}$ is the noisy measured output. With these, the *likelihood* and the *prior*, we receive the *marginal likelihood*

$$p\left(\underline{y}|\underline{U},\underline{\eta}\right) = \int p\left(\underline{y}|\underline{U},\underline{\eta},\underline{f}\right) p\left(\underline{f}|\underline{U}\right) d\underline{f}. \tag{30}$$

Since one seeks to optimize this ML and the logarithm does not change the position of the optima, it is valid to use to log marginal likelihood (LML) instead:

$$\log\left(p\left(\underline{y}|\underline{U},\underline{\eta}\right)\right) = -\frac{1}{2}\underline{y}^T\left(\underline{K}\right)^{-1}\underline{y} - \frac{1}{2}\log|\left(\underline{K}\right)| - \frac{N}{2}\log 2\pi. \tag{31}$$

In order to know the way of optimization for each parameter one needs to be able to compute the gradient for each hyperparameter $\eta_j$

$$\begin{aligned}\frac{\partial}{\partial\eta_j}\log p\left(\underline{y}|\underline{U},\underline{\eta}\right) &= \frac{1}{2}\underline{y}^T\underline{K}^{-1}\frac{\partial\underline{K}}{\partial\eta_j}\underline{K}^{-1}\underline{y} - \frac{1}{2}\mathrm{tr}\left(\underline{K}^{-1}\frac{\partial\underline{K}}{\partial\eta_j}\right)\\ &= \frac{1}{2}\mathrm{tr}\left(\left(\underline{\alpha}\,\underline{\alpha}^T - \underline{K}^{-1}\right)\frac{\partial\underline{K}}{\partial\eta_j}\right)\end{aligned} \tag{32}$$

$$\underline{\alpha} = \underline{K}^{-1}\underline{y}, \tag{33}$$

where $\underline{\alpha}$ is an equivalent to the parameters $\underline{\theta}$.

The pseudo likelihood (PL), also known as leave-one-out (LOO) cross validation (CV) approach, explains the likelihood of one output $y(i)$ for present hyperparameters $\underline{\eta}$, the input data $\underline{U}$ and the output data $\underline{y}_{-i}$. The vector $\underline{y}_{-i}$ consists of all entries of $\underline{y}$ except for the $i$-th entry $y(i)$. Therefore it is computed how well a single output fits to input data, hyperparameters and all the other computed outputs. Since one is not interested in just the likelihood of a single point, the computation is made for every output and summed up to the PL. Again, one rather takes the logarithm of the likelihood since it is easier to compute and does not change the position of the optima.

For a better overview, the $i$-th entry of $\underline{y}$ will also be denoted as $y_i$ instead of $y(i)$ in following equations. The predictive *log* probability for leaving out the $i$-th training case

$$\log p\left(y_i|\underline{U},\underline{y}_{-i},\underline{\eta}\right) = -\frac{1}{2}\log\sigma_i^2 - \frac{(y_i-\mu_i)^2}{2\sigma_i^2} - \frac{1}{2}\log 2\pi \tag{34}$$

is summed up to the PL

$$L_{LOO}\left(\underline{U},\underline{y},\underline{\eta}\right)=\sum_{i=1}^{N}\log p\left(y_i|\underline{U},\underline{y}_{-i},\underline{\eta}\right). \tag{35}$$

In order to compute (34) the LOO-CV predictive mean

$$\mu_i=y_i-\frac{\left[\underline{K}^{-1}\underline{y}\right]_i}{\left[\underline{K}^{-1}\right]_{ii}} \tag{36}$$

and variance

$$\sigma_i^2=\frac{1}{\left[\underline{K}^{-1}\right]_{ii}} \tag{37}$$

are needed. To know the direction for the optimization, the gradient of the LOO-CV predictive mean

$$\frac{\partial\mu_i}{\partial\eta_j}=\frac{\left[\underline{Z}_j\alpha\right]_i}{\left[\underline{K}^{-1}\right]_{ii}}-\frac{\alpha_i\left[\underline{Z}_j\underline{K}^{-1}\right]_{ii}}{\left[\underline{K}^{-1}\right]_{ii}^2} \tag{38}$$

and the variance

$$\frac{\partial\sigma_i^2}{\partial\eta_j}=\frac{\left[\underline{Z}_j\underline{K}^{-1}\right]_{ii}}{\left[\underline{K}^{-1}\right]_{11}^2} \tag{39}$$

are required in the computation of the gradient of the PL. With the parameters $\underline{\theta}$ equivalent expression

$$\underline{\alpha}=\underline{K}^{-1}\underline{y} \tag{40}$$

and the expression

$$\underline{Z}_j=\underline{K}^{-1}\frac{\partial\underline{K}}{\partial\eta_j}, \tag{41}$$

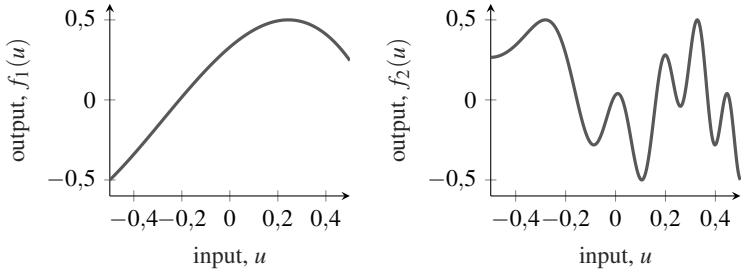



Figure 3: Illustration of artificial functions $f_1(u)$ (left) and $f_2(u)$ (right)

one can calculate the gradient of PL for each hyperparameter $\eta_j$

$$\begin{aligned}\frac{\partial L_{LOO}}{\partial \eta_j} &= \sum_{i=1}^{N} \frac{\partial \log p\left(y_i|\underline{U},\underline{y}_{-i},\underline{\eta}\right)}{\partial \mu_i}\frac{\partial \mu_i}{\eta_j} + \frac{\partial \log p\left(y_i|\underline{U},\underline{y}_{-i},\underline{\eta}\right)}{\partial \sigma_i^2}\frac{\partial \sigma_i^2}{\partial \eta_j} \\ &= \sum_{i=1}^{N} \frac{\left(\alpha_i \left[\underline{Z}_j \underline{\alpha}\right]_i - \frac{1}{2}\left(1+\frac{\alpha_i^2}{\left[\underline{K}^{-1}\right]_{ii}}\right)\left[\underline{Z}_j \underline{K}^{-1}\right]_{ii}\right)}{\left[\underline{K}^{-1}\right]_{ii}}.\end{aligned} \tag{42}$$

# 4 Results

The presented results are evaluated with artificial data. A slowly changing 4-th order polynomial function $f_1(u)$ with an almost linear part and a strongly nonlinear sinusoidal function $f_2(u)$ are used to create separate sets of data as presented in Fig. 3. In later calculations noise is added to the outputs in order to create training data.

The quality of the models is evaluated with the squared error ($SE$)

$$SE = (y_r - \hat{y})^2 \tag{43}$$

which is the squared difference between noiseless test output $y_r$ and the global model output $\hat{y}$.

## 4.1 Application of Local Gaussian Process Networks

For all three presented methods the result of training LGPMNs highly depends on the choice of validity function $\Phi$, threshold $t$ and optimization of hyperparameters $\underline{\eta}$. Therefore, one example is presented for every method to get a better insight.

These examples use data from function $f_1$ with additional normally distributed noise with zero mean and a standard deviation of 0.1 for training. The chosen threshold for data selection is $t_1 = 0.05$. ML optimization (MLO) is used for the optimization of hyperparameters. In order to show similarities and differences the outputs of the local models $y_1^{(p)}$ and $y_2^{(p)}$, the global model $\hat{y}$ and the noiseless test output $y_r$ are presented in Fig. 4. In order to show the extrapolation behavior, the input $u^{(p)}$ is extended beyond the range of the training input (-0.5 to 0.5). For a comparison of the performance the squared error at each point of the relevant input space is shown as well.

Method 1 and 2 produce very similar results. In both cases $\mathrm{LM}_1$ performs better than $\mathrm{LM}_2$. Only in $\mathrm{LM}_2$ is a noticeable difference between these methods results, where the SE is more evenly distributed for method 2. Both perform with a mean squared error of $MSE = 0.0011$.

Significant differences in the results are produced by method 3. Its LMs have a different extrapolation behavior; they converge slower towards the zero mean assumption of the prior. This extrapolation behavior leads to high error peaks at the left and right limits of the training input space. At the input $u_p = -0.2$ the $\mathrm{LM}_2$ already has a validity value close to zero but would improve the performance with a higher validity around this point. With $MSE = 0.0016$ this method delivered the worst performing global model.

## 4.2 Applied Optimization of Hyperparameters

For optimizing hyperparameters the MATLAB function `fminunc` is used. This function utilizes a gradient descent method to find a local optimum. It needs initial values for the variables that shall be optimized. When optimizing the local hyperparameters, it can happen that the optimizer gets stuck in a bad local optimum. This is illustrated in Fig. 5. A sinusoidal function was used for

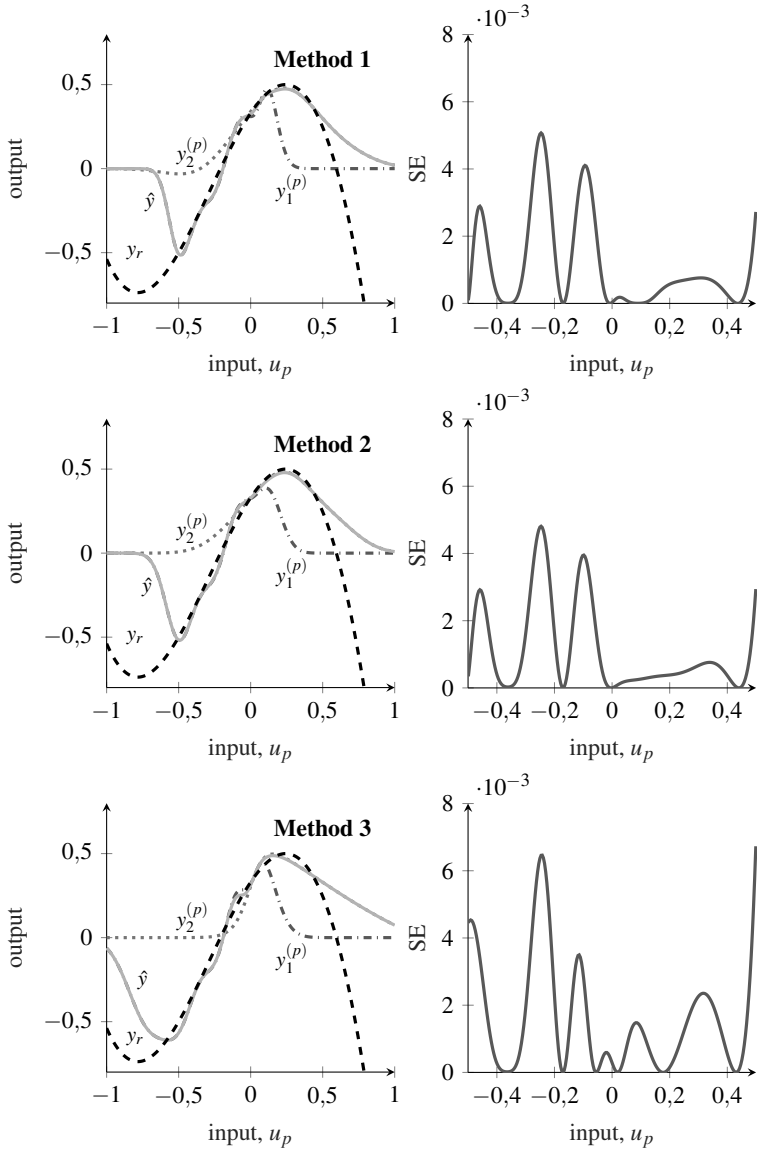



Figure 4: Model estimations of function $f_1$ (left) for methods 1-3 (from top to bottom) and their squared error (right)

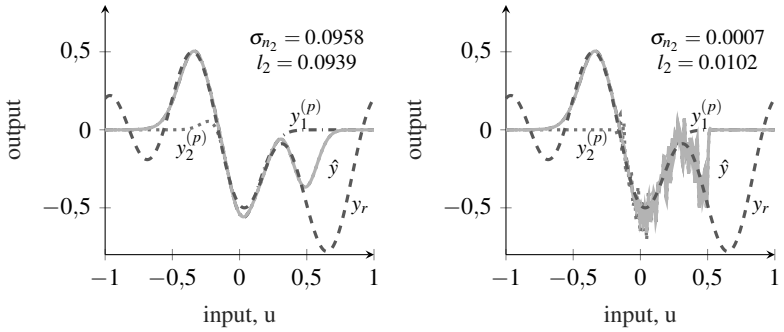


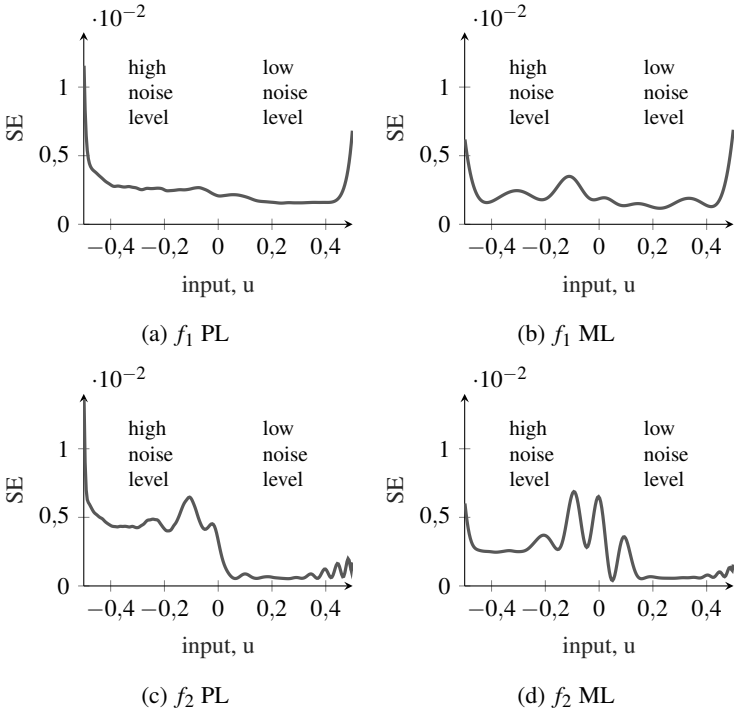

Figure 5: Model estimation of a sinusoidal function; Result of good initialization (left) and result of bad initialization (right)

training with the method 3 and PL optimization (PLO). Two different sets of initial hyperparameters $\underline{\eta}_2$ lead to different final values. In the second result the high frequent change in data was explained by a small lengthscale $l_2$ instead of the correct noise level $\sigma_{n_2}$ which was estimated too low. In contrast to that the first solution yields a good set of estimated hyperparameters and a smaller prediction error.

Since there are two LMs to be trained and different hyperparameters shall be estimated, to the first half of the training data noise with zero mean and a standard deviation of 0.15 is added. To the second half of the data noise with zero mean and a standard deviation of 0.05 is added.
To see the impact of the threshold $t$ on the training of LGPMNs, both functions $f_1$ and $f_2$ are used for training with both thresholds $t_1 = 0.05$ and $t_2 = 0.5$ and both optimization methods PLO and MLO for hyperparameters $\underline{\eta}_m$. Each training is done with 5000 different noise realizations. The following Fig. 6 and 7 present the average *SE* of the training with the same 5000 noise variations.

### 4.2.1 Optimization of Hyperparameters with Overlapping Training Data

The threshold $t_1 = 0.05$ results in 1/3 of the training data being shared for the estimation of both LMs. From Fig. 6 it can be seen that for $f_1$ at both ends of the input space the networks produce error peaks. In the transition zone between both LMs there are no significant error peaks. In both cases $\mathrm{LM}_1$



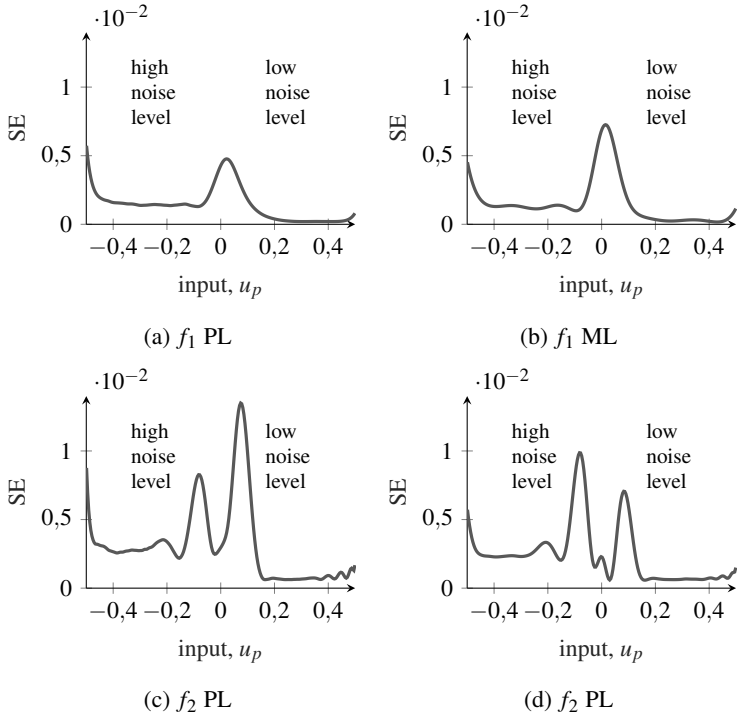

(a) $f_1$ PL (b) $f_1$ ML

(c) $f_2$ PL (d) $f_2$ ML

Figure 6: Average performance of 5000 LGPMN trainings with a threshold $t_1 = 0.05$ for both functions and both optimization methods

produces a slightly larger error. MLO leads to a slightly lower mean error but is less consistent in its prediction across the input space.

For $f_2$ both methods $LM_1$ produces a larger error and again MLO leads to a lower mean error that is more inconsistent across the input space. In these 4 test cases all LMs failed to predict the exact noise level. MLO and PLO predict the noise levels very similarly. The low noise level was overestimated and the high noise level was underestimated. PLO led to a higher variance in hyperparameter optimization.




(a) $f_1$ PL (b) $f_1$ ML

(c) $f_2$ PL (d) $f_2$ PL

Figure 7: Average performance of 5000 LGPMN trainings with a threshold $t_2 = 0.5$ for both functions and both optimization methods

#### 4.2.2 Optimization of Hyperparameters with No Overlapping Training Data

With the threshold $t_2 = 0.5$ the LMs receive each 1/2 of the data for training. This leads to no data being shared for training. The result of this can be seen in Fig. 7. For $f_1$ the largest errors occur in the transition zone of the LMs. The prediction except in this zone has improved in comparison to the results for threshold $t_1$. MLO is in this case not performing better in the average and produces a higher peak in the transition zone. For $f_2$ the error peaks occur in both cases at the edges of the transition zone of the LMs at $u_p = -0.1$ and $u_p = 0.1$. MLO produces a lower error in this case. For all 4 cases $\mathrm{LM}_1$ produces a larger error.

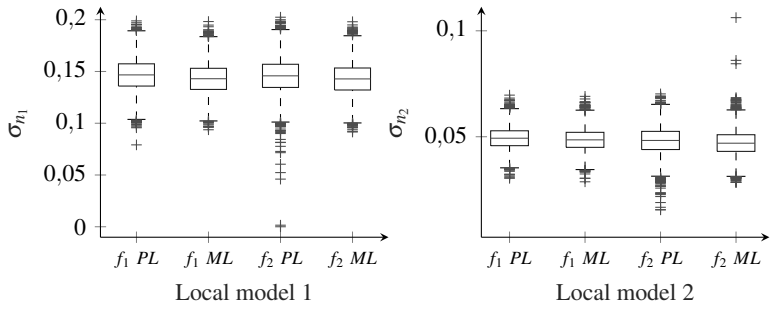



Figure 8: Estimation of measurement noise $\sigma_n$ parameters for both local models with $t_2 = 0.5$

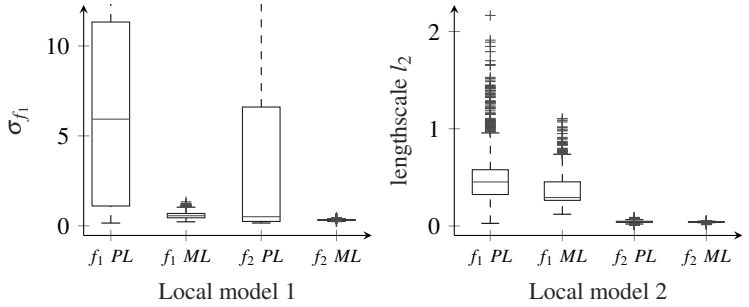



Figure 9: Estimation of output variance $\sigma_f$ for $\mathrm{LM}_1$ and lengthscale $l_1$ for $\mathrm{LM}_2$

### 4.2.3 Differences in Optimization of the Hyperparameters

With the threshold $t_2 = 0.5$ the estimation of hyperparameters becomes more precise. Figure 8 shows the estimated $\sigma_n$ values for both local models. Both optimization methods estimate them accurately. The PLO yields a slightly higher variance in its 5000 predictions of the hyperparameters.

To point out major differences in PLO and MLO, Fig. 9 shows the estimated output noise hyperparameter $\sigma_{f_1}$ values for the first local model $\mathrm{LM}_1$ and the estimated lengthscale $l_2$ for the second local model $\mathrm{LM}_2$. The PLO leads to a much higher variance in its prediction of $\sigma_{f_1}$. The most extreme outlier occurs in the prediction for function $f_1$ with the PLO with an estimated value of $\sigma_{f_1} = 305.6$. As can be seen, the estimation of the MLO is much more consistent.

For the lengthscale optimization (Fig. 9, right) the process behavior has clearly been well captured by the local GP model. This means that high spatial frequencies ($f_1$) lead to small optimal lengthscales, while low spatial frequencies ($f_2$) yield large lengthscales. This observation is valid for both hyperparameter optimization methods.

# 5 Conclusion

In this contribution three methods for estimating local Gaussian process model networks (LGPMNs) are presented. For all three methods a validity function defines where local models are active and how much they contribute to the output of the network $\hat{y}$. The first method (unweighted regression) utilizes Gaussian process regression (GPR) to calculate local models and blends the local models according to the validity functions.

The second method (weighted variance regression) is similar to the first one except that it uses a weighted measurement noise $\left(\sigma_n^2 \underline{\underline{Q}}^{-1}\right)$ in its parameter calculation. This comes from the assumption that points with a lower validity function are regarded as less trustworthy and need to be regularized stronger.

The third method (weighted kernel function regression) weights the kernel matrix with the validity function $\underline{\underline{Q}}\,\underline{\underline{K}}\,\underline{\underline{Q}}$ for its prediction of the parameters $\underline{\theta_m}$ and outputs $y_{p_m}$. The kernel matrix $\underline{\underline{K}}$ is a measure of similarity between data points. The idea is that data points with a low validity function should have less impact on the calculation of other points predictions.

All three methods successfully produce comparable results and none can be said to be superior without further investigation.

For the optimization of their hyperparameters $\underline{\eta}$, which are necessary for the model estimation, two methods namely marginal likelihood (ML) optimization and the pseudo likelihood (PL) optimization are presented and evaluated.

The ML explains how likely the chosen hyperparameters $\underline{\eta}$ explain the measured outputs $\underline{y}$ for given inputs $\underline{\underline{U}}$. The PL is a form of leave-one-out cross validation. This likelihood is calculated for each point separately and then summed up. It describes the likelihood for the output of one point when all other points are given.

The PL optimization leads to a less consistent estimation of the hyperparameters. That means it has a higher variance in its prediction and finds more extreme values than the ML optimization. In the output prediction quality however, the PL optimization leads to a more uniform error distribution than the ML optimization. But the ML optimization finds results with a slightly smaller mean squared error.

The adjustment of the threshold $t$ for data selection has a major impact on the prediction of the networks. If no data is shared ($t = 0.5$) the zone of transition between local models tends to produce higher errors, while sharing data ($t \ll 0.5$) leads to higher computation time and slightly worse performance across the remaining input space. When the data is split between the models without sharing any data, the prediction of the hyperparameters $\underline{\eta}$ is the most successful. This claim holds only when the boundaries of the LMs are determined reasonably. Therefore in real-world applications an important issue is to find suitable regions for the LMs. Which very likely requires a supervised approach such as LOLIMOT.

## References


[1] O. Nelles. “Nonlinear System Identification”. Springer Verlag. 2001.

[2] T. Bayes. “An essay towards solving a problem in the doctrine of chances”. Technical report. 1763.

[3] R. T. Cox. “Probability, Frequency and Reasonable Expectation”. In: *American Journal of Physics* 14(1): 1-13 1946.

[4] R. T. Cox. “The Algebra of Probable Inference”. In: *American Journal of Physics*. 31(1): 66 1963

[5] M. I. Jordan and R. A. Jacobs “Adaptive Mixture of Local Experts”. Cambridge. 1991.

[6] C. Park, J. Z. Huang and Y. Ding. “Domain Decomposition Approach for Fast Gaussian Process Regression of Large Spatial Data Sets”. In: *Journal of machine Learning Research* 12: 1697-1728 2011

[7] C. park and J. Z. Huang. “Efficient Computation of Gaussian Process Regression for Large Spacial Data Sets by Patching Gaussian processes”. In: *Journal of machine learning Research* 17:1-29 2016.

[8] C. E. Rasmussen and C. K. I. Williams. “Gaussian Processes for machine Learning”. 2004.

[9] C. E. Rasmussen and Z. Ghahramani. “Infinite Mixtures of Gaussian Process Experts”. In: *Advances in Neural Information Processing Systems* 2:881-888 2002.

[10] E. Snelson and Z. Ghahramani. “Local and Global Sparse Gaussian Process Approximations”. In. *11th international Conference on Artificial Intelligence and Statistics*. 2007.

# How to Solve the Dilemma of Margin-Based Equality Handling Methods

Samineh Bagheri[1], Wolfgang Konen[1], Thomas Bäck[2]

[1]Department of Computer Science
TH Köln (University of Applied Sciences)
Steinmüllerallee 1
51643 Gummersbach
E-Mail: {wolfgang.konen, samineh.bagheri}@th-koeln.de,

[2]Department of Computer Science
Leiden University, LIACS,
2333 CA Leiden, The Netherlands
Email: T.H.W.Baeck@liacs.leidenuniv.nl

## Abstract

Addressing black-box constrained optimization problems (COPs) in an efficient manner is a challenging task in real-world applications. Although population-based optimization heuristics including evolutionary strategies and genetic algorithms appear to be successful in absence of derivative information, they often require too many function evaluations which may not be affordable in practice. The surrogate-assisted technique SACOBRA [1, 2] aims to reduce the required number of function evaluations by building mathematical models for objective and constraint functions.

But for COPs with equality constraint(s), it is highly unlikely to find a fully feasible solution, because the feasibility ratio $\rho$ of this type of problems is exactly zero. The feasibility ratio $\rho$ is the probability that a point randomly chosen from the search space is feasible. In order to overcome this problem, many equality handling techniques consider a small feasibility margin $\varepsilon$ and try to find the best solution within the given margin. In other words, they transform an equality constraint $h(\vec{x}) = 0$ into an inequality constraint $|h(\vec{x})| - \varepsilon \leq 0$.

The dilemma with most margin-based equality handling techniques is the fact that they often report 'solutions' better than the true optimum which are however infeasible in the equality constraint(s) with a constraint violation of order $\varepsilon$.

The equality-handling approach in SACOBRA [1] uses a decrementing margin. Moreover, a refine mechanism moves the solutions within the given margin towards the subspace of the equality constraint(s). The refine mechanism applies a conjugate-gradient optimizer on the surrogate, so that no extra function evaluation is imposed. In this work we show that SACOBRA with refine step finds solutions very close to the true optimum, needing only relatively few function evaluations. Furthermore, we show that SACOBRA is able to find a range of different solutions (a set of Pareto-optimal solutions minimizing both the objective function and the constraint violation) for a given margin.

## 1 Introduction

An optimization problem can be defined as the minimization of an objective function (fitness function) $f$ subject to inequality constraint function(s) $g_1, \ldots, g_m$ and equality constraint function(s) $h_1, \ldots, h_r$:

$$\begin{array}{lll} \text{Minimize} & f(\vec{x}), & \vec{x} \in [\vec{a}, \vec{b}] \subset \mathbb{R}^d \\ \text{subject to} & g_i(\vec{x}) \leq 0, & i = 1, 2, \ldots, m \\ & h_j(\vec{x}) = 0, & j = 1, 2, \ldots, r \end{array} \tag{1}$$

where $\vec{a}$ and $\vec{b}$ define the lower and upper bounds of the search space (a hypercube). By negating the fitness function $f$ a maximization problem can be transformed into a minimization problem without loss of generality.

The most common way of tackling equality constraints in a black-box manner is to consider a small artificial feasibility margin of size $\varepsilon_0$ as it is done in several works [7, 11, 12]. Differential evolution optimizers [6, 14] often use a decremental margin $\varepsilon$ which gradually shrinks to $\varepsilon_f$. The same mechanism is found in swarm optimizers [13]. Other approaches transform an equality constraint into two inequalities [5] or they consider one side of the the equality constraint as the artificial feasible area [10]. They do not need any feasibility margin, but they cannot solve all types of equality COPs [1, Fig. 2].

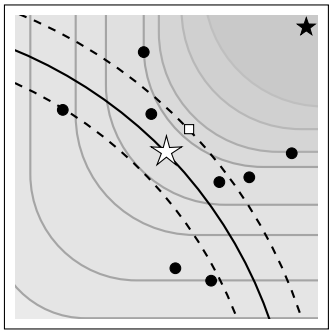


Figure 1: The shaded contours depict the objective function $f$ (darker = smaller). The black curve shows the equality constraint. Feasible solutions are restricted to this line. The black star shows the global optimum of the objective function which is different from the optimum of the constrained problem shown as the white star $\vec{x}^*$. The area enclosed by the dashed curves is the artificially feasible area, given a fixed feasibility margin $\varepsilon_0$. The white square marks the optimal solution $\vec{x}^*_{af}$ in the artificial feasible region. Black dots mark some infill points.

Other works [3, 4] use analytical information about the equality constraints to transform the optimization problem into a feasible subspace where the equality constraints are implicit. But the absence of such analytical information in black-box optimization does not allow us to solve the problem in a feasible subspace.

Now we describe in more detail the dilemma of margin-based equality COP solvers: Once there is a feasibility margin greater than zero, we have the following dilemma: Each COP solver has a whole set of possible solutions to choose from. Should we prefer a solution with better objective value but larger violation of the true equality constraint over another one with worse objective value but smaller violation? There is no clear answer to this, it is a multi-criteria problem. Nonetheless, most of the numerical constrained optimizers are designed to search for the solution with the best objective value $\vec{x}^*_{af}$ inside the artificially feasible region and they avoid approaching the true optimum $\vec{x}^*$ (see Fig. 1). The distance between these two solutions $\vec{x}^*_{af}$ and $\vec{x}^*$ can be pretty large, both in input space and objective space.[4] Of course the dilemma would

[4]The distances depend on $\varepsilon_f$ and the steepness of the equality functions $h_j(\vec{x})$ and the objective function $f(\vec{x})$.

disappear if $\varepsilon_f$ approaches zero, but many optimizer have problems with too small margins (they may not find feasible solutions).

Another aspect of the dilemma is this: The G-function suite used in the CEC2006 competition [8] is a set of COPs which is commonly used as a benchmark for constrained optimizers. Most of the solvers assume a feasibility margin of size $\varepsilon_f = 10^{-4}$ and, as a result, they often report solutions with an objective better than the optimal value for problems with equality constraints. The error measure in objective space becomes negative, which is not very logical. Other optimizers (among them SACOBRA) tend to stick closer to the true optimum, of course with a slightly worse objective value. This makes it difficult to compare solvers for equality COPs. A possible way out is to show and compare a **set** of solutions in the multi-objective space {objective function, constraint violation}.

SACOBRA also uses a decrementing margin strategy. But additionally, each solution will be pushed in the direction of the true feasible subspace by means of a refine mechanism.

This work aims to show the benefits of using the refine step in approaching the true optimum. Moreover, we want to emphasize the importance of reporting a **set** of solutions (Pareto set) instead of one final solution. This should become a good practice for benchmarking optimizers on equality-constrained problems.

This paper is organized as follows: Sec. 2 briefly describes the equality-handling procedure in SACOBRA. After explaining in Sec. 3 the experimental setup, we show in Sec. 4 the impact of the refine step on solving several G-problems by representing a set of solutions in the neighborhood of the optimum. Sec. 5 summarizes the paper.

## 2 Methods

SACOBRA is a surrogate-assisted constrained optimizer which takes advantage of radial basis function interpolation techniques to reduce the expensive evaluations of objective and constraint functions [2].

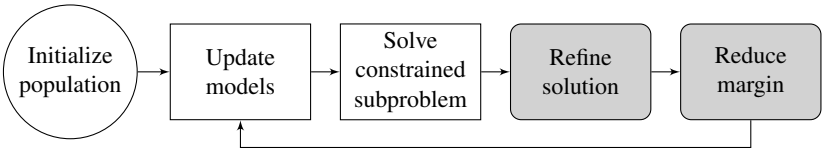



Figure 2: Main steps of SACOBRA with the equality handling mechanism.

SACOBRA tends to approach the optimal solution iteratively. After generating a population of initial points, SACOBRA builds surrogate models for the objective and constraint functions. Based on solving a so-called *internal COP* on the surrogate models, SACOBRA suggests the next infill point (the optimum returned from the internal COP). The new infill point will be evaluated on the real functions and added to the population of points, so the models can be updated. The algorithm goes through this loop until the budget is exhausted, as depicted in Fig. 2. Handling equalities in SACOBRA has two main ingredients: I. a decreasing feasibility margin, II. a refine step which aims to move the solutions towards the subspace of equality constraint(s).

## 2.1 Decrementing Margin

The zero-volume feasible space attributed to the $j$-th equality constraint $h_j(\vec{x}) = 0$ is expanded to a tube-shaped region around it: $|h_j(\vec{x})| - \varepsilon^{(n)} \leq 0$ by the proposed algorithm. By gradually reducing the margin $\varepsilon^{(n)}$ the solutions are smoothly guided towards the real feasible subspace. The equality margin $\varepsilon$ can be reduced in different fashions described in [1]. We use a simple exponential decay scheme. In each iteration $n$ the feasibility margin will be reduced as follows:

$$\varepsilon^{(n+1)} = \max(\varepsilon_f,\ \varepsilon^{(n)}\beta), \tag{2}$$

where the lower bound $\varepsilon_f$ is often set to a small value close to machine accuracy.

### 2.2 Refine Mechanism

The **refine step** is done by minimizing the squared sum of all equality constraint violations by means of a conjugate-gradient (CG) method. This minimization step as described in Eq. (3) is done based on the surrogates of the equality constraints and not the real equality functions; therefore, no extra real function evaluation are imposed by this step.

$$\text{Minimize} \quad \sum_{j=1}^{r} (s_j^{(n)}(\vec{x}))^2, \qquad \vec{x} \in [\vec{a}, \vec{b}] \subset \mathbb{R}^d, \tag{3}$$

where $s_j^{(n)}(\vec{x})$ is the surrogate of the $j$-th equality. Although in black-box COPs we do not know the analytical form of the equalities, the refine step tries to move the best found solution in each iteration towards the feasible subspace with assistance of the estimated models of the equality functions $h(\vec{x})$. Furthermore, the refine step can be helpful in not losing a good feasible solution when decreasing the margin after an iteration. More algorithmic details can be found in [1].

## 3 Experimental Setup

We use $4 \cdot d$ points generated by LHS to initialize SACOBRA, where $d$ is the size of the variable space. According to [1] we know that SACOBRA is not very sensitive to the choice of the decaying factor $\beta$, but it performs best in $\beta \in [0.90, 0.94]$, therefore we use $\beta = 0.92$. The internal COPs are addressed by `Cobyla()` from the NLOPTR package in R [9]. The refine step is done with assistance of the `optim()` function from the STATS package, the method is set to L-BFGS-B and the maximum number of refine iterations is set to $10^4$. A maximum of 400 function evaluations is permitted. SACOBRA sets the final feasibility margin to $\varepsilon_f = 10^{-8}$. We compare two different configurations of SACOBRA with and without the refine step and also a differential evolution (DE) strategy with automatic parameter adjustment as proposed by Brest [6].

Table 1: Characteristics of G-problems with equality constraints: $d$: dimension, LI: the number of linear inequalities, NI: number of nonlinear inequalities, LE: the number of linear equalities, NE: the number of nonlinear equalities, $\alpha$: number of active constraints.

| Fct. | $d$ | type | LI | NI | LE | NE | $\alpha$ |
|---|---|---|---|---|---|---|---|
| G03 | 20 | nonlinear | 0 | 0 | 0 | 1 | 1 |
| G05 | 4 | nonlinear | 2 | 0 | 0 | 3 | 3 |
| G11 | 2 | nonlinear | 0 | 0 | 0 | 1 | 1 |
| G13 | 5 | quadratic | 0 | 0 | 0 | 3 | 3 |

For DE results, we run our experiments using the DEOPTIMR package. The function `JDEoptim()` in DEOPTIMR applies the same adaptive equality margin found in [14] but with a more aggressive updating scheme. The final feasibility margin is set to $\varepsilon_f \in \{10^{-4}, 10^{-8}\}$ in different DE runs.

The characteristics of the G-problems used in this study is listed in Tab. 1. We select here those problems from the G-function suite that have equality constraints.

## 4 Results and Discussion

The result in Fig. 3 shows that SACOBRA with only 400 fe (function evaluations) is able to populate the Pareto front nicely (we take the points with and without refine together, which are both available within the same run). On the other hand, DE needs more function evaluations (3300 fe, upper plot) to come into the vicinity of the Pareto front, however, many DE points have a constraint violation larger than $10^{-4}$. Only if we add more fe to DE (7800 fe, lower plot), then the DE points will more or less densely populate the region near the Pareto front.

Although the refine step is a very simple step, it is essential for our algorithm. As shown in Fig. 3, SACOBRA with refine is doing a better job in reducing the constraint violation in comparison with the SACOBRA without the refine step. On the one hand, the refine step helps us to move the best solution found in each margin towards the real feasible subspace in each iteration. Although a wrong approximation of the equality constraints in the early iterations can cause a

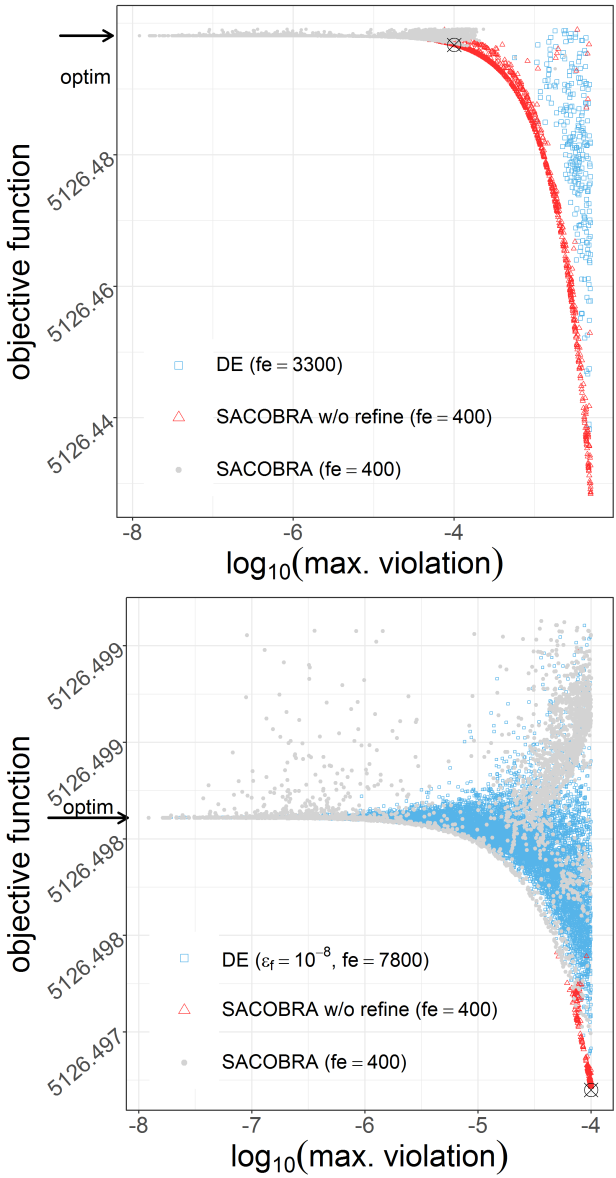



Figure 3: G05 problem: Infill points generated by different algorithms (DE, SACOBRA w/o refine, SACOBRA) during the optimization process. The final equality margin for all three algorithms is set to $\varepsilon_f = 10^{-8}$.

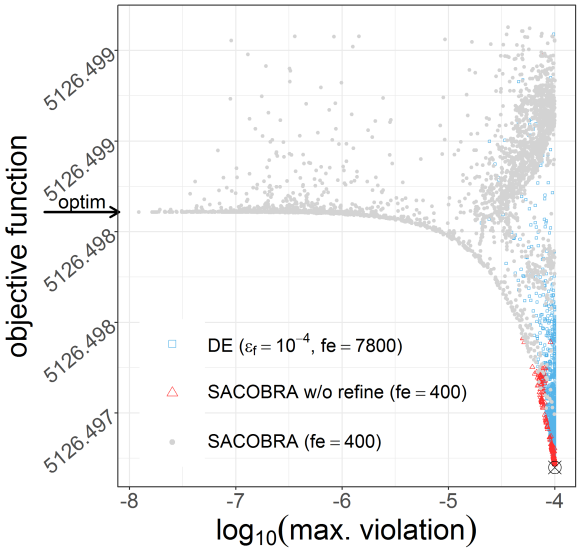



Figure 4: Same as Fig. 3, but for DE the feasibility margin is set to the value suggested by CEC2006 [8] $\varepsilon_f = 10^{-4}$.

shift towards more violated regions, the model(s) of equality constraint(s) gradually improve by learning about these regions and eventually the refine step can guide the solutions towards the correct direction. On the other hand, since only one new point will be added to the population in each iteration, usually this point will sit at the border of the artificial feasible region after the optimization step. If we now shrink the artificial feasible region without refining, we would lose this point and jump to another feasible point, if any, probably with a much larger objective value.

Comparing a set of found solutions instead of only the final one, helps us to have a better comparison for the performance of the equality constrained algorithms. Additionally, providing a set of solutions for COPs with equality constraints can be beneficial in practice, because the user then can decide to take a solution which fits best to his/her application.

Figs. 5–7 show similar results for the other G-problems. Tab. 2 shows the results from all runs in tabular form.

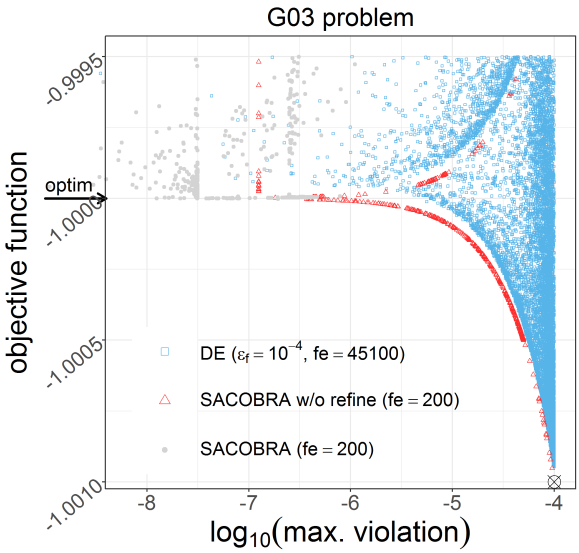



Figure 5: Same as Fig. 3, but for G03 problem.

Fig. 4 compares SACOBRA and DE when the final feasibility margin is set to $\varepsilon_f = 10^{-4}$ for DE. As expected DE even after thousands of function evaluations, converges to the optimum of the artificial feasible area $\vec{x}_{af}$.

## 5 Conclusion

In this work we have shown that SACOBRA is capable of approaching the true solution of equality-constrained optimization problems in less function evaluations than other optimizers. SACOBRA finds significantly better solutions in terms of constraint violation, vicinity to the optimum and efficiency, compared to DE, a well-known algorithm from evolutionary strategies. It avoids – at least to a large extent – the often seen dilemma of equality-constraint optimization that a margin leads to solutions 'better than the optimum' by violating some of the equality constraints. We have seen that the refine step – which may be also a useful building block for other optimization schemes – is essential for achieving this goal.

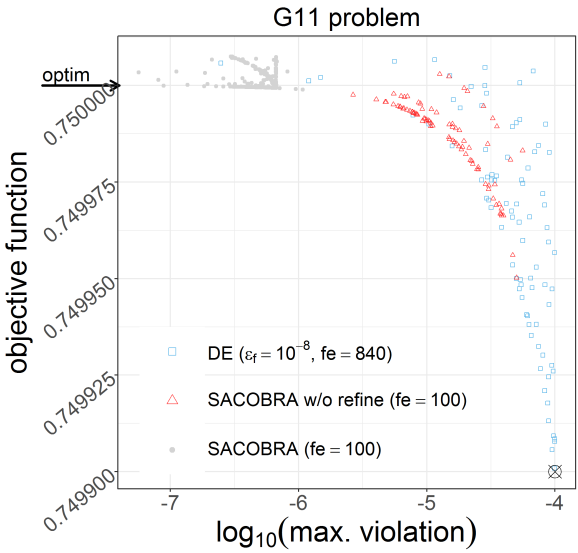



Figure 6: Same as Fig. 3, but for G11 problem.

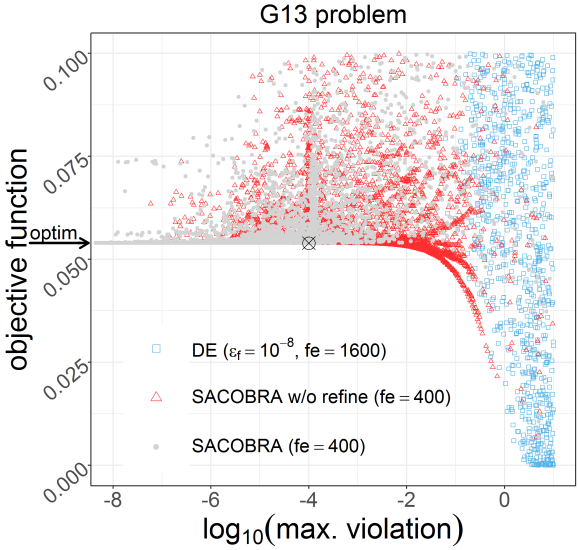



Figure 7: Same as Fig. 3, but for problem G13. As shown in Tab. 2 DE cannot find good solutions with even more function evaluations.

Table 2: Results for the different G-problems: true objective value, median and standard deviation of objective values from 30 runs of the solvers, mean and median (over 30 runs) of the maximum equality constraint violation, averaged number of function evaluations (fe). Note that DE was run here for only $200 \cdot d$ iterations on each problem, which leads to bad objective values in the case of G03 and G13. If more iterations were used for DE, it would find near optimal solutions for G03 but not for G13.

| problem | trueObj | medObj | sd.obj | meanViol | medViol | fe |
|---|---|---|---|---|---|---|
| | | | SACOBRA | | | |
| G03 | -1.0000 | -1.0000 | 5e-07 | 5e-08 | 3e-08 | 200 |
| G05 | 5126.4981 | 5126.4981 | 6e-04 | 1e-07 | 7e-08 | 400 |
| G11 | 0.7500 | 0.7500 | 2e-05 | 5e-06 | 7e-07 | 100 |
| G13 | 0.0539 | 0.0540 | 2e-01 | 1e-08 | 7e-09 | 400 |
| | | | DE ($\varepsilon_f = 10^{-4}$, $200 \cdot d$ iterations) | | | |
| G03 | -1.0000 | -0.8728 | 4e-02 | 6e-05 | 7e-05 | 13124 |
| G05 | 5126.4981 | 5126.4967 | 2e-06 | 1e-04 | 1e-04 | 7391 |
| G11 | 0.7500 | 0.7499 | 6e-08 | 1e-04 | 1e-04 | 1902 |
| G13 | 0.0539 | 0.4267 | 2e-01 | 4e-02 | 4e-02 | 2359 |
| | | | DE ($\varepsilon_f = 10^{-8}$, $200 \cdot d$ iterations) | | | |
| G03 | -1.0000 | -0.8061 | 5e-02 | 4e-05 | 4e-05 | 12428 |
| G05 | 5126.4981 | 5126.4981 | 4e-05 | 3e-07 | 2e-07 | 6612 |
| G11 | 0.7500 | 0.7500 | 6e-11 | 1e-08 | 1e-08 | 2425 |
| G13 | 0.0539 | 0.3670 | 2e-01 | 5e-02 | 5e-02 | 2274 |

Showing a set of best solutions minimizing both, maximum constraint violation and objective function, is helpful to have a fair comparison of different algorithms and also it is more practical for real-world applications because the user has a chance to select the solution which suits the application best.

## References


[1] Bagheri, S., Konen, W., Bäck, T.: Equality constraint handling for surrogate-assisted constrained optimization. In: 2016 IEEE Congress on Evolutionary Computation (CEC). pp. 1924–1931 (July 2016)

[2] Bagheri, S., Konen, W., Emmerich, M., Bäck, T.: Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. Applied Soft Computing 61, 377 – 393 (2017)

[3] Beyer, H.G., Finck, S.: On the design of constraint covariance matrix self-adaptation evolution strategies including a cardinality constraint. IEEE Transactions on Evolutionary Computation 16(4), 578–596 (2012)

[4] Beyer, H.G., Finck, S., Breuer, T.: Evolution on trees: On the design of an evolution strategy for scenario-based multi-period portfolio optimization under transaction costs. Swarm and Evolutionary Computation 17, 74–87 (2014)

[5] Bhattacharjee, K.S., Ray, T.: A novel constraint handling strategy for expensive optimization problems. In: 11th World Congress on Structural and Multidisciplinary Optimization. Sydney, Australia (June 2015)

[6] Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. Trans. Evol. Comp 10(6), 646–657 (Dec 2006)

[7] Deb, K.: An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering 186(2), 311 – 338 (2000)

[8] Liang, J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P., Coello, C.C., Deb, K.: Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. Journal of Applied Mechanics 41, 8 (2006)

[9] R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2014), `http://www.R-project.org`

[10] Regis, R.G.: Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. Engineering Optimization 46(2), 218–243 (2013)

[11] Runarsson, T.P., Yao, X.: Stochastic ranking for constrained evolutionary optimization. IEEE Transactions on Evolutionary Computation 4(3), 284–294 (2000)

[12] Runarsson, T.P., Yao, X.: Search biases in constrained evolutionary optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 35(2), 233–243 (2005)

[13] Xie, X.F., Zhang, W.J., Bi, D.C.: Handling equality constraints by adaptive relaxing rule for swarm algorithms. Congress on Evolutionary Computation (CEC) pp. 2012–2016 (2004)

[14] Zhang, H., Rangaiah, G.: An efficient constraint handling method with integrated differential evolution for numerical and engineering optimization. Computers & Chemical Engineering 37, 74–88 (2012)

# Prognosen für die Vorausschauende Instandhaltung bei Bestandsanlagen der Prozessindustrie

Dr.-Ing. Nico Zobel, Dr.-Ing. Sergii Kolomiichuk,
Dr.-Ing. Andreas Herzog, Andreas Lehwald

Fraunhofer-Institut für Fabrikbetrieb und -automatisierung IFF
Sandtorstr. 22, 39106
E-Mail: nico.zobel@iff.fraunhofer.de

## 1 Ausgangssituation

Eine vorausschauende Instandhaltung von Anlagenkomponenten in der Prozessindustrie (z. B. Pumpen, Gebläse, Wärmeübertrager, etc.) im Sinne der Vision Industrie 4.0 basiert auf der Kenntnis des aktuellen und der Prognose des künftig zu erwartenden technischen Zustandes einer Anlage. Dafür müssen in der Regel große Mengen verfügbarer und oft ungenutzter Sensordaten mit geeigneten Maschinelles-Lernen-Methoden, z. B. künstliche neuronale Netze (ANN), analysiert und bewertet werden, um schleichende Veränderungen zielgerichtet identifizieren zu können und somit konkrete Handlungsempfehlungen für Instandhaltungsmaßnahmen abzuleiten. Daher wird die Nutzung von Ergebnissen von Datenanalysen für eine Erhöhung der Effizienz in der industriellen Produktion, sei es Fertigungs- oder Prozessindustrie, eine immer größere Rolle spielen. Die dazu notwendige Datenanalyse-Kompetenz beschaffen sich die produzierenden Unternehmen entweder durch Beauftragung externer Dienstleister oder durch den Aufbau eigener Kompetenzen im Bereich der Datenanalyse. Es sind mehr und mehr Software-Produkte verfügbar, mit denen auch Nicht-Mathematiker, z. B. Ingenieure aus den Bereichen Maschinenbau oder Verfahrenstechnik, selbst in die Lage versetzt werden, Datenanalysen durchzuführen, ohne auf externe Dienstleister angewiesen zu sein. Da allerdings die Erfahrung und das Hintergrundwissen fehlt, führt der Einsatz solcher Software

oft dazu, dass die Ergebnisse dieser Datenanalysen nur sub-optimal sind. Auf der anderen Seite besitzen ausgebildete Datenanalysten, seien sie unternehmensextern oder -intern, oft nur wenig bis gar keine eigene Betriebserfahrung mit der Produktionsanlage, deren Daten sie analysieren. Das Erfahrungswissen der Anlagenbediener geht, sofern überhaupt, zumeist nur in sehr geringem Umfang in die Datenanalysen ein.

## 2 Lösungsansatz

Der von uns vorgeschlagene Lösungsansatz besteht in einer flexiblen und praktikablen Kombination von Datenanalyse mittels maschinellem Lernen und einer Modellierung des Erfahrungswissens von Mitarbeitern in einem Fuzzy-System. Dadurch soll es Anwendern von Datenanalyse-Tools in Industrieunternehmen ermöglicht werden, mit geringerem Aufwand bessere Analyseergebnisse zu erzielen als bisher.

## 3 Methodik

Um das Erfahrungswissen der Anlagenbediener modellhaft zu erfassen, wird ein neuer IFF-Lösungsansatz mittels Fuzzy-Logik eingesetzt. Im ersten Schritt werden Anlagenbediener, Instandhalter oder Servicetechniker befragt, welche Einflussfaktoren nach Ihrer Erfahrung einen großen Einfluss auf Beanspruchungen und/oder Zustandsveränderungen ausüben. Das führt zu einer Reduzierung der Komplexität der Analysen, da weiterführend nur relevante Einflussgrößen betrachtet werden. Diese Einflussgrößen (Eingangsparameter) werden fuzzifiziert, d. h. beispielsweise wird ein Temperaturwert einer Fuzzy-Variablen „Temperatur“ und darunter der Bereich zwischen 30 und 120 °C einem Fuzzy-Term „hoch“ zugeordnet. Im nächsten Schritt werden Ursache-Wirkungs-Zusammenhänge mittels linguistischen Regeln aus den Aussagen der Anlagenbediener aufgenommen, z. B.:

WENN (Temperatur = hoch) DANN (Beanspruchung = hoch),
WENN (Temperatur = gering) DANN (Beanspruchung = gering).

Damit lassen sich selbst komplizierte Zusammenhänge einfach erfassen und abbilden. Anschließend werden Rechenoperationen zur Schlussfolgerung und Zusammenfassung mehrerer Regeln zu einem Bewertungsergebnis formuliert. Dies geschieht mittels „unscharfer“ Fuzzy-Logik-Verarbeitungsalgorithmen durch Komposition von Regeln und abschließender Defuzzifizierung. Ein einfaches Beispiel für den Einfluss der Wicklungstemperatur von 50 °C auf die Beanspruchung eines Motorlagers ist in Bild: 1 dargestellt. Die im Bild dargestellten Aussagen führen dazu, dass der Temperatur im Bereich 20 bis 80 °C der Fuzzy-Therm „gering“ (Bild: 1, links, Kurve „gering “) und im Bereich 30 bis 60 °C „hoch“ (Bild 1, links, Kurve „hoch“) zugeordnet wird. Für die Temperatur 50 °C (Abszisse) ergeben die Kurvenverläufe dann Zugehörigkeitsgrade von 0,48 (für „gering“) und 0,65 (für „hoch“). Diese werden anschließend in das für die Beanspruchung ausgewählte Modell (Bild: 1, rechts Bild) übertragen und das Flächenverhältnis „gering“/„hoch“ bestimmt. In dem Beispiel errechnet sich das Flächenverhältnis zu 1,05, was einer Beanspruchung des Motorlagers von 105 % bei 50 °C Wicklungstemperatur entspricht. Üblicherweise wird diese Vorgehensweise noch für weitere die Belastung eines Motorlagers betreffende Einflussgrößen (wie z. B. Laufzeit, Drehzahl, Auslastung (Volllast, Teillast), Betriebsstunden, Schwingung Lager A, Schwingung Lager B, etc.) durchgeführt und die Gesamtbeanspruchung ermittelt.

Diese Methodik wird in Kombination mit einer Datenanalyse mittels künstlicher neuronaler Netze angewendet. Ein Beispielfall stellt eine Komponente einer verfahrenstechnischen Anlage dar. Die Ergebnisse dieser Anwendung werden im Folgenden kurz zusammengefasst.

# 4 Anwendungsbeispiel: Prognose der Verstopfung einer Zweistoffdüse

Die Zweistoffdüse einer Granulieranlage neigt in unregelmäßigen Abständen zur Verstopfung und führt zum Ausfall der Anlage. Anhand von Betriebsdaten aus der Anlagenhistorie sollte ein Modell entwickelt werden, welches die Verstopfungsneigung bzw. den -grad der Düse quantifiziert. Der aktuelle Verstopfungsgrad und damit die Ausfallwahrscheinlichkeit der Düse ist an der diskontinuierlich betriebenen Anlage mittels Sensoren nicht direkt ermittelbar.

**Aussagen:** „Hohe Wicklungstemperaturen schädigen die Lager."

„Die Wicklungstemperatur ist „gering", wenn sie zw. 20 und 80 °C liegt."

„Die Wicklungstemperatur ist „hoch", wenn sie zw. 30 und 60 °C liegt."

**Frage:** Wie hoch ist denn die Beanspruchung der Lager bei z. B. 50 °C?

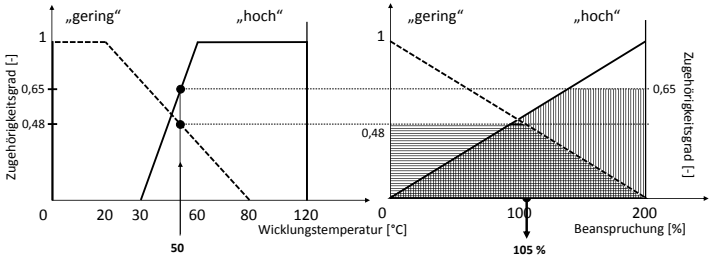



Bild 1: Fuzzy-Logik für die Beanspruchung eines Lagers in Abhängigkeit der Wicklungstemperatur

Im untersuchten Zeitraum von 50 Betriebstagen innerhalb eines Jahres traten 12 Verstopfungsfälle auf, die zum Ausfall der Anlage führten. Zunächst wurden aus der Vielzahl der Sensoren diejenigen identifiziert, in denen sich ein beginnendes Zusetzen der Düse am wahrscheinlichsten darstellt. Im Ergebnis wurden fünf kontinuierlich aufgezeichnete Sensordaten (Druck und Durchfluss der Flüssigkeit sowie der Luft und die Pumpenleistung) untersucht. Neben den Momentanwerten ist auch der zeitliche Verlauf (Anstieg, Abfall) dieser Messdaten von Bedeutung. Deshalb wurden daraus abgeleitete Größen wie gleitender Mittelwert, erste und zweite Ableitung mit in die Auswertung einbezogen. Anschließend wurde eine einfache Zielfunktion für das Maschinelle Lernen mittels künstlichem neuronalen Netz definiert (siehe Bild: 2), welche einen vermuteten Anstieg des Grades der Verstopfung vor einer aufgezeichneten Verstopfung abbildet. Der so approximierte Verstopfungsgrad steigt innerhalb einer vorgegebenen Zeit (1 Stunde) mit einer festgelegten Funktion linear an, sodass dieser zum Zeitpunkt der Verstopfung den Wert 1 (100%) erreicht.

Zudem konnte das Erfahrungswissen der Mitarbeiter über den Druck- und Temperaturverlauf vor einer Verstopfung in einem Fuzzy-Modell beschrieben werden. Die Betriebsdaten wurden anschließend mittels künstlichem neuronalen Netz (Bild: 3), Fuzzy-Modell (Bild: 4) sowie einer Kombination aus bei-

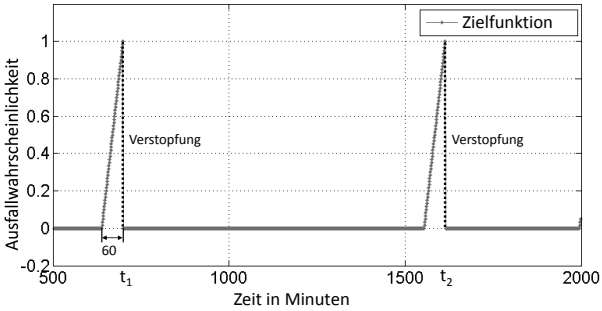



Bild 2: Zielfunktion für das künstliche neuronale Netz

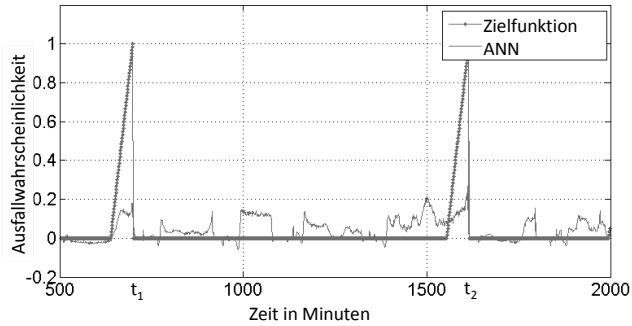



Bild 3: Prognose der Ausfallwahrscheinlichkeit mittels künstlichem neuronalem Netz

den Modellarten (Bild: 5) mit der Zielfunktion korreliert. Im Fall der approximierten Ausfallwahrscheinlichkeit mittels künstlichem neuronalem Netz (Bild: 3) werden im Ergebnis die beiden Verstopfungen detektiert, wenn auch nicht mit voller Amplitude. Zudem werden allerdings auch diverse nicht vorhandene Verstopfungen fälschlicherweise festgestellt. Kurz gesagt: das Ziel ist nicht erreicht. Das ist ein Beispiel für ein typisches Ergebnis einer Datenanalyse mit (zu) wenigen Lernereignissen.

Auch im Fall der Prognose der Ausfallwahrscheinlichkeit mittels Fuzzy-Modell (Bild: 4) ist das Ergebnis nicht zufriedenstellend. Dieses Modell erzeugt ebenfalls zu viele Ereignisse.

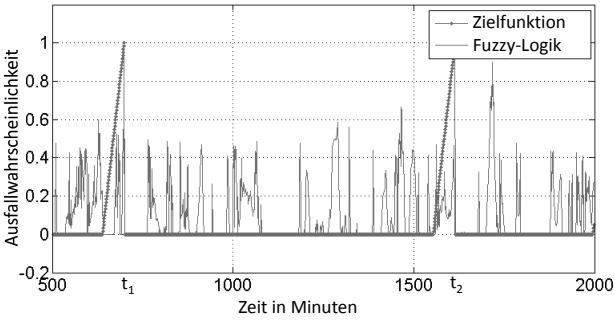



Bild 4: Prognose der Ausfallwahrscheinlichkeit mittels Fuzzy-Modell

Die Kombination beider Modellarten (Bild: 5) hingegen approximiert die Ausfallwahrscheinlichkeit wesentlich besser. Die beiden Verstopfungen werden mit voller Amplitude detektiert. Zwar werden auch noch nicht vorhandene Verstopfungen fälschlicherweise festgestellt, allerdings mit geringerer Amplitude bzw. mit einem Dirac-Impuls-ähnlichen Anstieg, die im Rahmen des Post-Processings z. B. mittels eines Schwellwertes (z. B. 0,3) und einer minimalen Haltezeit (z. B. 10 s) eliminiert werden können. Mit dem trainierten Modell kann während des Betriebes der Anlage eine beginnende Verstopfung der Düse anhand der aufgezeichneten Messdaten prognostiziert und zeitlich detektiert sowie das Einleiten entsprechender Gegenmaßnahmen zum Abwenden des Anlagenausfalls ermöglicht werden.

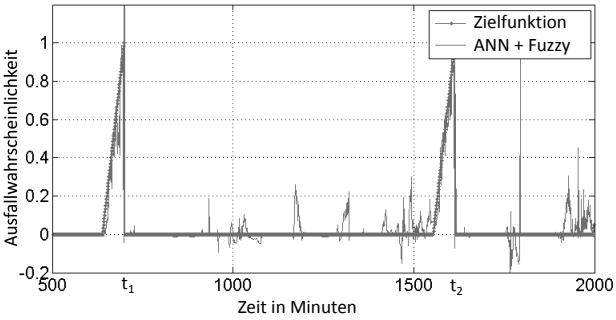



Bild 5: Prognose der Ausfallwahrscheinlichkeit mittels künstlichem neuronalem Netz und Fuzzy-Modell

# Identifying Characteristic Gait Patterns in Real-World Scenarios

Anton Pfeifer, Volker Lohweg

Institute Industrial IT (inIT),
Ostwestfalen-Lippe University of Applied Sciences,
Langenbruch 6, 32657 Lemgo, Germany
E-Mail: {anton.pfeifer, volker.lohweg}@hs-owl.de

## 1 Introduction

The population of many industrialised countries are among the oldest in the world. In Germany, one-fifth of the population is currently over 65 years old, in 2050 it will be more than one-third [1]. Due to this demographic change, there will be more elderly people with chronic disease progressions and the number of neurodegenerative diseases such as dementia or Parkinson's disease will increase [1]. In this context, innovative approaches in medicine and care are particularly relevant for the modernisation of person-oriented services. More and more modern IT solutions in healthcare rely, for example, on sensors of mobile devices (wearables, smartphones, etc.) to record data over a long period of time [1, 2].

Most mobile devices have accelerometers and gyroscopes to measure our physical activity, while wearables additionally use sensors to continuously measure heart rate or blood pressure [3]. This large amount of data provides information on a patient's health state. For instance, one of the most common diseases is persistently high blood pressure (hypertension). A consequence of hypertension is an increased risk of heart diseases, stroke, and dementia [4, 5]. Unfortunately, half of the people with this condition are not diagnosed. Thus, CHANDRASEKHAR et al. [3] proposed a prototype blood pressure sensor which only requires the press of a fingertip. The proposed sensor can be integrated into smartphones and consists of two components: a *photoplethysmo-*

*graphy* (PPG) sensor and a pressure sensor. The first component is a low-cost optical sensor for measuring blood volume changes. The second component is a pressure sensor that measures the pressure applied to it. To measure a user's blood pressure, a finger is pressed on the sensor, on which an algorithm processes the generated data. Pressing the fingertip creates external pressure on the underlying artery, similar to a blood pressure cuff.

Typically these kind of approaches tackle research hot-spots in three categories [6]: (i) the compartmentalisation of the data, i.e. the data is stored in one repository and harmonised for further analysis [7], (ii) the *corruption* of the data during measurement, which results in corrective work on erroneous data, missing data, and imprecise data [8], and (iii) the *complexity* inherent in the system. Once the patient data has been collected and validated, the last step describes the machine learning of complex data. For instance, inferences are drawn about a latent state using measurements (condition monitoring) or multiple types of informations are analysed regarding a patient health state (information fusion) [9].

The main focus of this work will be laid on the third aspect, complexity. It proposes an algorithm to automatically analyse a user's gait. A mobile device monitors the gait of patients with accelerometers. Subsequently, a recorded gait section is evaluated. Here, ideas from indoor navigation and time-series motif discovery are combined. The term motif describes a frequently repeated unknown pattern in a signal [10]. In this context, the motif discovery algorithm is applied during long-term gait analysis to classify meaningful, new, and even unknown movement patterns.

The paper is divided into five main sections. First, the basics of the human gait are introduced. Thereupon, state of the art concepts are discussed. Subsequently, the proposed approach for gait analysis is presented. Finally, the evaluation will be carried out on real-world data, before the conclusion and an outlook for further investigations in the underlying field is indicated.

## 2 Preliminaries

The human gait is considered as an automatic motor task. It requires strategic planning of the best route and continuous interaction with the environment. If an inappropriate path is chosen or the physical abilities are misjudged, a fall may be the consequence. Thus, healthy walking relies on a balance of various interacting neuronal systems. It consists of three components [11]:

- locomotion,
- balance, and
- adaptation to the environment.

Dysfunction in the adaptation process and variations in gaits might be an indication of neurophysiological system malfunction [11]. Therefore, gait analysis could help in diagnosing and evaluating the severity of neurophysiological disorders. In general, three methods are applied to analyse the human gait [11]. The first method describes the kinetic aspects involved in producing movements. The second method denotes the body movement through space. At the present state of practice, both methods require sophisticated gait labs, which are equipped with motion capture systems and force plates. Finally, the third method describes spatiotemporal aspects of gait. This method captures parameters from of a particular leg step by step. Thereupon, characteristics of the right leg can be compared with those of the left leg. This makes it possible to evaluate to what extent the captured parameters differ from the parameters of a reference group with or without gait pathology [12].

Independent of the chosen method, the human gait is divided into two phases for each foot [13]. The individual phases are shown in Fig. 1. The first phase describes the time between the initial contact of the right foot and when the same foot leaves the ground. It is defined as *stance phase*. Thereupon, the *swing phase* begins when the right toe leaves the ground and ends with the initial contact of the right heel. The next two phases are referred to the left foot, respectively. These phases are defined within a *gait cycle*. One cycle starts when one foot contacts the ground and ends with the subsequent floor contact of the same foot. After describing the basic terminology for gait analysis, the following section introduces a review of existing work [13].

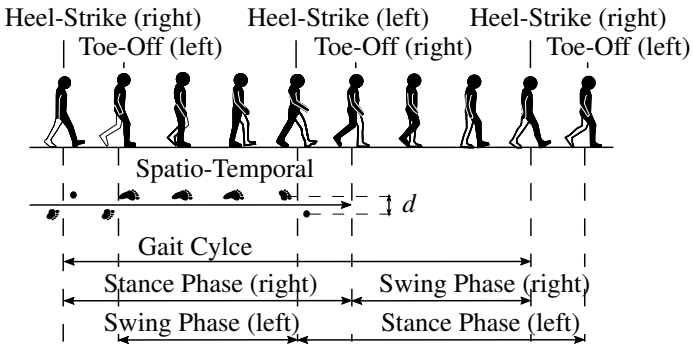



Figure 1: Illustration of individual phases and spatiotemporal aspects of the human gait. The first phase of the right foot is defined by the stance phase. Subsequentially, the swing phase begins when the right toe leaves the ground and ends with the initial contact of the right heel. Similarly, for the left foot the first phase begins with the swing phase and ends with the stance phase. The spatiotemporal aspects capture parameters from step-to-step of a particular leg. Typical parameters are stride time or the distance between both feet. Picture is based on [15].

## 3 Related Work

In the past years, many methods have been introduced to monitor human gait. In clinical environments, gait analysis is performed in controlled labs using motion capture systems or force plates [14]. These systems provide very accurate results but are not suitable for everyday use, since they can usually only be used stationary, are expensive, and require a high level of operational competence. An alternative is, for example, force sensitive resistors (FSR). FSR provide contact information and serve as a reference method to determine the accuracy of the algorithms. Unfortunately, these systems do not provide information about kinematic or spatial information during a walk [16]. However, these are important aspects of pathological gait assessments. For this purpose, body-worn sensors such as accelerometers and gyroscopes are a more suitable alternative. They provide spatiotemporal information and can be used in combination to estimate further parameters [17, 18].

KHANDELWAL and WICKSTRÖM [18] proposed a gait-event detection for real-world applications based on long-term measurements of accelerometers. The approach is based on the recognition of two gait events: heel strike (HS) and toe off (TO). These two events are further used to calculate parameters such as swing time, stance, and stride duration. To detect gait events in long-term observations, domain knowledge about human gait is combined with a time-frequency analysis. The accuracy and robustness of the proposed algorithm is evaluated in indoor and outdoor experiments. During the experiments the sensors are in fixed positions and the participants vary speed and move over different terrain. Although the algorithm provides robust results, the analysed signals are obtained from individual accelerometers in pre-defined locations. Unfortunately, during long-term analysis it is quite likely that external factors might disturb the position of the sensor. GADALETA and ROSSI [19] introduced a gait recognition approach that tackles this problem and can be applied in daily living. The algorithm is based on an orientation-independent gait circle detector. It combines a convolutional neural network feature extractor with a walking cycle classification based on a support vector machine, all included in a coherent multi-stage authentication system.

Based on the above mentioned approaches the acquired gait parameters might be applied for continuous monitoring of elderly patients. Here, the variations of gait parameters over time could provide information about disease development and therefore help to detect syndromes of Parkinson's disease, dementia, and other neurophysiological disorders [11]. Unfortunately, most of the approaches are applied on predefined walking sequences and are therefore not suitable for long-term monitoring. Thus, in the following section an approach is proposed to detect activities from raw sensor signals.

# 4 Approach

As stated, the human gait is divided into three main components: locomotion, balance, and adaptation to the environment [11]. The interaction of the individual components requires a balance between different musculoskeletal and neuronal systems. Dysfunctions in one of the systems usually lead to a gait disorder. The risk of falling and losing much of the mobility increases with age

[18]. Therefore, identifying gait changes in humans' natural environment is a key challenge in gait analysis [20].

In this section an approach for the detection of human movement patterns is introduced. It is divided into two subsections: first, a method which corrects the accelerometer axis by applying a rotational transform is detailed, and second, a method which detects meaningful movement patterns is specified.

## 4.1 Orientation Independent Transform

Accelerometers attached to the human body do not always have the same axis alignment as the axis alignment of the human body. Unfortunately, during long-term analysis it is quite likely that external factors might disturb the position of the sensors and thus have an impact on the measurements [21]. In order to evaluate human gait, the acceleration sensor is virtually rotated so that its axes fit the axes of the body.

In general, a transformation between two three-dimensional vectors is described by a rotation matrix. It describes the alignment of a *user's coordinate system* (UCS) with respect to the *device coordinate system* (DCS). The algorithm is adopted by [22, 23]. A vector $\mathbf{a}$ in the DCS is converted into the UCS by a multiplication with a $(3 \times 3)$ rotation matrix $\mathbf{R}$:

$$\mathbf{a}_{\text{UCS}} = \mathbf{R} \cdot \mathbf{a}_{\text{DCS}}, \tag{1}$$

where $\mathbf{a} \in \mathbb{R}^n$ is a column accelerometer vector $\mathbf{a} = (a_x, a_y, a_z)^T$, and $(\cdot)^T$ is the transpose operator. This orientation correction process involves three steps. Firstly, raw accelerometer signals are measured in the DCS over a period of 5 seconds. Next, the mean gravitational force applied to the sensor is defined as the initial orientation $\mathbf{a_i} = (\bar{\mathbf{a}}_\mathbf{x}, \bar{\mathbf{a}}_\mathbf{y}, \bar{\mathbf{a}}_\mathbf{z})^T$, where $\mathbf{a_x}, \mathbf{a_y}, \mathbf{a_z}$ are accelerometer samples in the defined window. As stated, the UCS is aligned with gravity, and therefore the desired orientation vector is $\mathbf{a_d} = (0, -9.81, 0)^T$ [22].

Secondly, the rotation matrix is constructed. In this work a quaternion rotation is applied to describe a three-dimensional rotation [24]. In order to construct a set of quaternions, the rotation angle $\alpha$ and the axis vector $\mathbf{A}$ have to be calculated. The axis vector $\mathbf{A}$ is defined by the cross product

$$\mathbf{A} = \mathbf{a_i} \times \mathbf{a_d}, \tag{2}$$

where $\mathbf{a_i}$ is the initial orientation, and $\mathbf{a_d}$ is the desired orientation. Subsequently, the vector $\mathbf{A}$ is normalized by

$$\mathbf{A}_{\text{norm}} = \mathbf{A} \cdot \left( \frac{1}{\sqrt{a_x^2 + a_y^2 + a_z^2}} \right). \tag{3}$$

Then the angle between $\mathbf{a_i}$ and $\mathbf{a_d}$ is deducted from the dot product and is given by

$$\begin{aligned} \mathbf{a_i} \cdot \mathbf{a_d} &= ||\mathbf{a_i}|| \cdot ||\mathbf{a_p}|| \cos(\alpha), \\ a_{i,x}a_{d,x} + a_{i,y}a_{d,y} + a_{i,z}a_{d,z} &= ||\mathbf{a_i}|| \cdot a_{d,y} \cos(\alpha), \\ \alpha &= \cos^{-1}\left(\frac{a_{i,y}}{||\mathbf{a_i}||}\right). \end{aligned} \tag{4}$$

By using the axis-angle representation of Eq. (3) and Eq. (4) the quaternion rotation matrix

$$\mathbf{R} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1^2 q_2^2 + q_0^2 q_3^2) & 2(q_1^2 q_3^2 - q_0^2 q_2^2) \\ 2(q_1^2 q_2^2 - q_0^2 q_3^2) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2^2 q_3^2 + q_0^2 q_1^2) \\ 2(q_1^2 q_3^2 + q_0^2 q_2^2) & 2(q_2^2 q_3^2 - q_0^2 q_1^2) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{5}$$

is constructed, where $q_0$ is the scalar part of the quaternion and $q_1, q_2, q_3$ is the vector part

$$\mathbf{q} = [q_0, q_1, q_2, q_3]^T = \begin{bmatrix} \cos(\frac{\alpha}{2}) \\ \sin(\frac{\alpha}{2}) A_{\text{norm},x} \\ \sin(\frac{\alpha}{2}) A_{\text{norm},y} \\ \sin(\frac{\alpha}{2}) A_{\text{norm},z} \end{bmatrix}. \tag{6}$$

After the basic procedure to construct a rotation matrix, in the third step Eq. (5) is applied to Eq. (1). Thus, raw accelerometer data are converted from the DCS to the UCS [22]. Thereupon, a method to detect meaningful movement patterns is applied.

## 4.2 Motif Discovery

In the following, the motif discovery is described. Here, the main goal is to classify meaningful, new, and even unknown movement patterns from long-term accelerometer signals. The acquired signals are often obtained from activities such as walking, running, or sitting. Unfortunately, there is no general approach to relate specific movement patterns to sensor data. Thus, the main goal is to detect subsequences in the acquired long-term accelerometer signals which behave similarly. One approach to detecting similar movement patterns is to apply *motif discovery* algorithms, where the term *motif* is defined as a frequently repeated unknown pattern in a signal [10]. Before describing the design of the motif discovery algorithm in more detail some notations used in this work are briefly introduced.

Let $x[n] = (x_1, x_2, \ldots, x_n)^T, n \in \mathbb{N}$ be a time series, and $M_i$ be a motif of length $m$ from $x[n]$, and let $(s_i, s_j)$ be a pair of subsequences of length $m$, where $i$ and $j$ are the beginning position of each subsequence, then a motif is determined by the smallest distance $d(s_i, s_j)$ compared to all other pairs. The function $d(s_i, s_j)$ describes a similarity measure [25].

Each subsequence is extracted by a sliding window. Let $W$ be a window of length $m$, and let $w$ be the length of a given time series $x[n]$, then by sliding $W$ over $x[n]$ all subsequence $s_i$ are extracted [26].

In order to classify meaningful movement patterns the *Complex Quad Tree Wavelet Packet Transform* (CQT-WPT) is applied. This transform is utilised to detect shifted and multi-scale motifs of different size and is applied in various applications [27, 28, 29]. The CQT-WPT decomposes a signal $x[n]$ into discrete frequency bands (*scales*). Each scale consists of two *wavelet packet trees* (WPT), which are working in parallel to each other. The first tree, WPT A, represents the real part of the signal. The second tree, WPT B, provides information about the imaginary part. In each tree the decomposition is performed by discrete filter banks, where ${}^s h_a, {}^s h_b$ are high-pass and ${}^s g_a, {}^s g_b$ are low-pass filters. The scale is denoted by $s \in \mathbb{N}$. Sequentially, an even $\downarrow 2^e$ and odd $\downarrow 2^o$ decomposition is applied. The corresponding wavelet coefficients are described as follows [27].

Let ${}^{s}C[n] = {}^{s+1}C_{2J}[n], {}^{s+1}C_{2J+1}[n], {}^{s+1}C_{2J+2}[n], {}^{s+1}C_{2J+3}[n]$ be the coefficients of the CQT-WPT for the real part WPT A, where $J = 2j$, $0 \leq j < 2^s \cdot (s-1)$, $M$ is equal to the length of filter ${}^{s}g_b$, and $L = length({}^{s}C_j)$, then each coefficient is obtained by

$$\begin{aligned}
{}^{s+1}C_{2J}[n] &= \sum_{k=0}^{M+L-1} {}^{s}g_a[k]\,{}^{s}C_j[2n-k], \\
{}^{s+1}C_{2J+1}[n] &= \sum_{k=0}^{M+L-1} {}^{s}h_a[k]\,{}^{s}C_j[2n-k], \\
{}^{s+1}C_{2J+2}[n] &= \sum_{k=0}^{M+L-1} {}^{s}g_a[k]\,{}^{s}C_j[2n+1-k], \\
{}^{s+1}C_{2J+3}[n] &= \sum_{k=0}^{M+L-1} {}^{s}h_a[k]\,{}^{s}C_j[2n+1-k].
\end{aligned} \tag{7}$$

The coefficients of the imaginary part WPT B are obtained in a similar way. Here, the filters ${}^{s}g_a, {}^{s}h_a$ are replaced by ${}^{s}g_b, {}^{s}h_b$ [27].

The corresponding wavelet and scaling functions in WPT A are given as $\psi_a(t)$ and $\phi_a(t)$:

$$\begin{aligned}
\psi_a(t) &= \sqrt{2} \sum_{n=0}^{M} {}^{s}h_a[n]\,\phi_a(2t-n), \\
\phi_a(t) &= \sqrt{2} \sum_{n=0}^{M} {}^{s}g_a[n]\,\phi_a(2t-n).
\end{aligned} \tag{8}$$

Similarly for WPT B the wavelet and scaling functions $\psi_b(t)$ and $\phi_b(t)$ are obtained by replacing ${}^{s}h_a[n], {}^{s}g_a[n]$ with ${}^{s}h_b[n], {}^{s}h_b[n]$ [27]. In Fig. 2 a graphical representation of the CQT-WPT is depicted. Here, the decomposition is depicted for e.g. two scales. A further detection of unknown patterns within time-series does not necessarily require to consider all frequency scales. Instead, the scale with the best frequency solution and the largest information content is selected.

One suitable approach to detect the best frequency resolution is to consider the energy density of the low-pass coefficients in each scale as a criterion for decomposition. The low pass coefficients of the superior node are compared with its subnodes. If the energy density $E\{{}^{s+1}C_{2J}\}$ is greater than $E\{{}^{s}C_{2J}\}$,

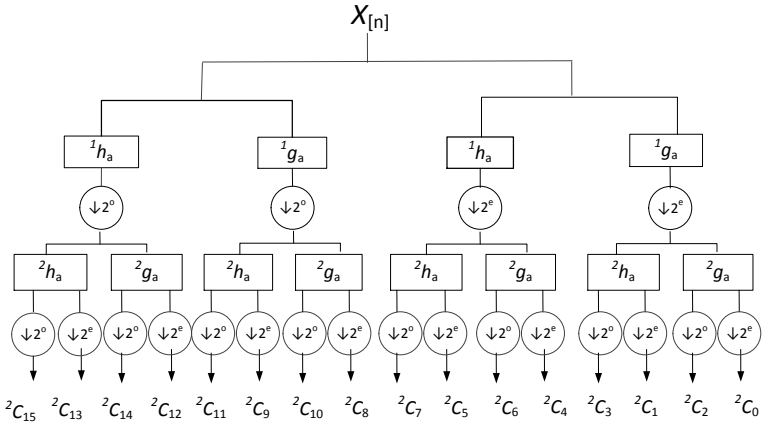



Figure 2: Illustration of the first wavelet packet filter bank. It consists of two scales. The first scale is obtained by the filters ${}^1h_a, {}^1g_a$ and an even $\downarrow 2^e$ and odd $\downarrow 2^o$ decomposition. Sequentially, the second scale is calculated. Here, the filters ${}^1h_a, {}^1g_a$ are replaced by ${}^2h_a, {}^2g_a$ [27].

the decomposition is continued. If the energy density $E\{{}^sC_{2J}\}$ is greater than $E\{{}^{s+1}C_{2J}\}$, the decomposition is completed, and features in the current scale are extracted in order to distinguish classes from each other [28].

**Feature Selection.** Three features are extracted of the normalised coefficients from the CQT-WPT. The first two features are statistical moments: mean value and variance. In addition, energy of the wavelet coefficients is considered as a third feature. Next, the similarity between time-series employing the features is determined.

**Similarity Measures.** As stated, a motif is describes as a pair of subsequences $(s_i, s_j)$, where the distance $d(s_i, s_j)$ is the smallest among all other pairs. In this work the following two methods are applied: Euclidean distance [30] and Canberra distance [30]. Both measurement methods are metric and have linear computational time. After measuring the similarity of all subsequences, the distances are saved in the matrix $\mathbf{D}$. Significant motifs are detected by Alg. 1. The Threshold $T_R$ is defined with prior knowledge. Points where the standard deviation of the signal $\mathbf{D}_i$ changes most significantly are defined as relevant change points.

Algorithm 1.: Search for significant motifs in a time series

**Voraussetzung:** **D**: Similarity measures of all motifs; $T_R$: Threshold
**Abschlussbedingung:** **M**: Set of significant motifs

```
1: function SEARCHSIGNIFICANTMOTIFS(D, T_R)
2:     for all D_i ∈ D do
3:         ipt = findChangePoints(D_i)              ▷ Compute relevant change points
4:         for j = 2 : |ipt| do
5:             meanDistance = mean(D_i[ipt(j), ipt(j−1)])
6:             if meanDistance < T_R then
7:                 Put relevant motif in matrix M
```

# 5 Preliminary Results

The proposed concept is evaluated under consideration of two gait datasets. The first dataset is the MAREA gait database (*Movement Analysis in Real-world Environments using Accelerometers*) [31]. The database comprises gait activities of 20 healthy adults (12 male and 8 female) in different real-world environments. Each subject had an accelerometer attached to their waist, left wrist, and left/right angels using elastic bands. The second gait dataset contains 3-day accelerometer recordings of 71 elder adults, which is originally used to study gait stability and fall risk [32, 33]. Before describing the results the performance measurements used in this work are briefly introduced.

The results are evaluated by five quality measurements [34]: Sensitivity (Sn), Precision (Pr), F-Measure (F-M), and Correct discovery rate (CR). Each quality measurement is based on the following possibilities to qualify a motif: true positive rate (TP), false negative rate (FN), false positive rate (FP), and true negative rate (TN).

Table 1: Preliminary results of the proposed method for the MAREA gait database (*Movement Analysis in Real-world Environments using Accelerometers*) [31]. Sn: Sensitivity, Pr: Precision, F-M: F-Measure, CR: Correct discovery rate.

| Similarity Measures | Sn (%) | Pr (%) | F-M (%) | CR (%) |
|---|---|---|---|---|
| Euclidean distance | 77.6 | 100 | 87.4 | 77.6 |
| Canberra distance | 71.4 | 100 | 83.3 | 71.4 |

Table 2: Preliminary results of the proposed method for the second gait dataset. Sn: Sensitivity, Pr: Precision, F-M: F-Measure, CR: Correct discovery rate.

| Similarity Measures | Sn (%) | Pr (%) | F-M (%) | CR (%) |
|---|---|---|---|---|
| Euclidean distance | 91.9 | 79.2 | 85.1 | 91.9 |
| Canberra distance | 41.9 | 96.3 | 29.2 | 41.9 |

Let $N$ be the number of all motifs, and let $n^+$ be the number of correctly detected motifs, then the correct motif discovery rate is determined by

$$\mathrm{CR} = \left( \frac{n^+}{N} \right). \tag{9}$$

Sensitivity (Sn), Precision (Pr), and F-Measure (F-M) are given by

$$\begin{aligned} \mathrm{Sn} = \left( \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \right), \mathrm{Pr} = \left( \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \right), \text{ and} \\ \mathrm{F\text{-}M} = 2 \cdot \left( \frac{\mathrm{Pr} \cdot \mathrm{Sn}}{\mathrm{Pr}+\mathrm{Sn}} \right). \end{aligned} \tag{10}$$

Table 1 displays the results for a subject from the first dataset. The signal consists of $4.9 \cdot 10^4$ samples (6 minutes). The sampling frequency is 128 Hz. The length of the sliding window is set to $w = 500$. The subject starts walking for 3 minutes at a pace of his choice and then change to jogging or running for 3 minutes at a steady pace. In this example both similarity measures perform equally. In table 2 the results for a subject from the second dataset are given. The signal consists $5.5 \cdot 10^4$ samples. The sampling frequency is 100 Hz. The

sliding window size is set to $w = 500$. In this sequence, the subject follows everyday tasks such as walking. In this example ED outperforms CD.

## 6 Conclusion and Outlook

The approach proposed in this work combines ideas from indoor navigation and time-series motif discovery. It is divided into three subparts. First, raw motion data of a mobile device is transformed into an orientation invariant coordinate system. It remains fixed during a walk and is oriented on the axes of the human body. Thus, the algorithm is robust to external factors during long-term measurements. In the second part, the Complex Quad Tree Wavelet Packet Transform (CQT-WPT) [27] is applied to extract features. The CQT-WPT is utilised to detect shifted and multi-scale motifs of different size [27]. In this context, the CQT-WPT is applied during long-term gait analysis to classify meaningful movement patterns. The evaluation is carried out under consideration of two gait datasets: the MAREA gait database (*Movement Analysis in Real-World Environments using Accelerometers*) [31] and a dataset which contains accelerometer recordings of elder adults. The first dataset comprises gait activities of 20 healthy adults in different real-world environments. The second gait dataset contains 3-day accelerometer recordings of 71 elder adults [32]. The preliminary results demonstrated that the approach is able to detect meaningful movement patterns without any prior knowledge of the sensor data.

In order to further enhance the proposed approach, the following issues need to be addressed in future work. Currently the approach considers only two similarity measurements and one motif discovery algorithm. Here, additional methods need to be evaluated in further steps. Furthermore, motif discovery algorithms leads to a pool of activity patterns. The activity patterns should first be sorted by relevance before they are further processed. A promising idea is to employ semantics to the found motifs in future work.

## Acknowledgment

This work was partly funded by the European Regional Development Fund (EFRE.NRW) within the project Projektwerkstatt Gesundheit 4.0, Grant-No.: 34-EFRE-0300024.

## References

[1] Statistisches Bundesamt. “Ältere Menschen in Deutschland und der Europäischen Union (EU).” https://www.destatis.de/. 2016.

[2] Vienne, A., Barrois, R. P., Buffat, S., Ricard, D., and Vidal, P.-P. “Inertial Sensors to Assess Gait Quality in Patients with Neurological Disorders: A Systematic Review of Technical and Analytical Challenges.” In: *Front. Psychol.*. Vol. 8. 2017.

[3] Chandrasekhar, A., Kim, C. S., Naji, M., Natarajan, K., Hahn, J. O., and Mukkamala, R. “Smartphone-based Blood Pressure Monitoring via the Oscillometric Finger-pressing Method.” In: *Science Translational Medicine*. Vol. 10. No. 431. 2018.

[4] Lackland, D. T., and Weber, M. A. “Global Burden of Cardiovascular Disease and Stroke: Hypertension at the Core.” In: *The Canadian Journal of Cardiology*. Vol. 31. No. 5. Pp. 569–571. 2015.

[5] Lau, D. H., Nattel, S., Kalman, J. M., and Sanders, P. “Modifiable Risk Factors and Atrial Fibrillation.” In: *Circulation*. Vol. 136. No. 6. Pp. 583–596. 2017.

[6] Johnson, A.E., Ghassemi, M.M., Nemati, S., Niehaus, K.E., Clifton, D.A., and Clifford, G.D. “Machine Learning and Decision Support in Critical Care.” In: *Proceedings of the IEEE*. Vol. 104. No. 2. Pp. 444–466. 2016.

[7] Liang, X., Zhao, J., Shetty, S., Liu, J., and Li, D. “Integrating Blockchain for Data Sharing and Collaboration in Mobile Healthcare Applications.” In: *28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE. 2017.

[8] Ghassemi, M. M., Richter, S. E., Eche, I. M., Chen, T. W., Danziger, J., and Celi, L. A. “Robust Heart Rate Estimation from Multiple Asynchronous Noisy Sources using Signal Quality Indices and a Kalman Filter.” In: *Physiological Measurement*. Vol. 29. No. 1. Pp. 15–32. 2008.

[9] Ghassemi, M. M., Richter, S. E., Eche, I. M., Chen, T. W., Danziger, J., and Celi, L. A. “A Data-driven Approach to Optimized Medication Dosing: A focus on Heparin.” In: *Intensive Care Medicine*. Vol. 40. No. 9. Pp. 1332–1339. 2014.

[10] Patel, P., Keogh, E., Lin, J., and Lonardi, S. “Mining Motifs in Massive Time Series Databases.” In: *International Conference on Data Mining*. Pp. 370–377. IEEE. 2002.

[11] Snijders, A. H., van de Warrenburg, B. P., Giladi, N., and Bloem, B. R. “Neurological Gait Disorders in Elderly People: Clinical Approach and Classification.” In: *The Lancet Neurology*. Vol. 6. No. 1. Pp. 63–74. 2007.

[12] Schutte, L. M., Narayanan, U. P., Stout, J. L., Selber, P., Gage, J. R., and Schwartz, M. H. “An index for quantifying deviations from normal gait.” In: *Gait & Posture*. Vol. 11. No. 1. Pp. 25–31. 2000.

[13] Inman, V. T. “Human Locomotion.” In: *Canadian Medical Association Journal*. Vol. 94. No. 20. Pp. 1047–1054. 1966.

[14] Bruening, D. A., and Ridge, S. T. “Automated Event Detection Algorithms in Pathological Gait.” In: *Gait & posture*. Vol. 39. No. 1. Pp. 472–477. 2014.

[15] UT Dallas. “Walking in Graphs.” http://www.utdallas.edu/atec/midori/Handouts/walkingGraphs.htm. 2017.

[16] Everett, T., and Kell, C. “Human Movement: An Introductory Text.” 6th ed. London: Elsevier Health Sciences UK. 2010.

[17] Rebula, J. R. and Ojeda, L. V. and Adamczyk, P. G. and Kuo, A. D. “Measurement of Foot Placement and its Variability with Inertial Sensors.” In: *Gait & posture*. Vol. 38. No. 4. Pp. 974–980. 2013.

[18] Khandelwal, S., and Wickström, N. “Gait Event Detection in Real-World Environment for Long-Term Applications: Incorporating Domain Knowledge into Time-Frequency Analysis.” In: *Transactions on Neural Systems and Rehabilitation Engineering*. IEEE. Vol. 24. No. 12. Pp. 1363–1372. 2016.

[19] Gadaleta, M., and Rossi, M. “IDNet: Smartphone-based Gait Recognition with Convolutional Neural Networks.” In: *Pattern Recognition*. Vol. 74. Pp. 25–37. 2018.

[20] Rueterbories, J., Spaich, E. G., and Andersen, O. K. “Gait Event Detection for use in FES Rehabilitation by Radial and Tangential Foot Accelerations.” In: *Medical Engineering & Physics*. Vol. 36. No. 4. Pp. 502–508. 2014.

[21] Gietzelt, M., Schnabel, S., Wolf, K.-H., Büsching, F., Song, B., Rust, S., and Marschollek, M. “A Method to Align the Coordinate System of Accelerometers to the Axes of a Human Body: The Depitch Algorithm.” In: *Computer Methods and Programs in Biomedicine*. Elsevier. Vol. 106. No. 2. Pp. 97–103. 2012.

[22] Tundo, M. D., Lemaire, E., and Baddour, N. “Correcting Smartphone Orientation for Accelerometer-based Analysis.” In: *International Symposium on Medical Measurements and Applications*. IEEE. Pp. 58–62. 2013.

[23] Deng, Z.-A., Wang, G., Hu, Y., and Di W.. “Heading Estimation for Indoor Pedestrian Navigation Using a Smartphone in the Pocket.” In: *Sensors*. Vol. 15. No. 9. Pp. 21518–21536. 2015.

[24] Diebel, J. “Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors.” In: *Matrix*. Vol. 58. No. 15–16. Pp. 1–35. 2006.

[25] Mueen, A., Keogh, E., and Bigdely-Shamlo, N. “Finding Time Series Motifs in Disk-resident Data.” In: *9th International Conference on Data Mining*. IEEE. Pp. 367–379. 2009.

[26] Castro, N., and Azevedo, P. J. “Multiresolution Motif Discovery in Time Series.” In: *Proceedings of the International Conference on Data Mining*. Pp. 665–676. 2010.

[27] Torkamani, S., Lohweg, V., Hoffmann, F., and Hüllermeier, E. “Shift-invariant Feature Extraction for Time-series Motif Discovery.” In: *25. Workshop Computational Intelligence*. Vol. 54. Pp. 23–45. 2015.

[28] Torkamani, S. Dicks, A., and Lohweg, V. “Anomaly Detection on ATMs via Time Series Motif Discovery.” In: *21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE. 2016.

[29] Deppe, S., and Lohweg, V. “Evaluation of Similarity Measures for Shift-Invariant Image Motif Discovery.” In: *International Journal On Advances in Intelligent Systems*. IARIA Journals. Pp. 434–446. 2018.

[30] Deza, M. M., and Deza, E. “Encyclopedia of Distances.” Springer. 2009.

[31] Khandelwal, S., and Wickström, N. “Evaluation of the Performance of Accelerometer-based Gait Event Detection Algorithms in Different Real-World Scenarios using the MAREA Gait Database.” In: *Gait & Posture*. Vol. 51. Pp. 84–90. 2017.

[32] Weiss, A., Brozgol, M., Dorfman, M., Herman, T., Shema, S., Giladi, N., and Hausdorff, J. M. “Does the Evaluation of Gait Quality During Daily Life Provide Insight into Fall Risk? A Novel Approach using 3-day accelerometer Recordings.” In: *Neurorehabilitation and Neural Repair*. Vol. 27. No. 8. Pp. 742–752. 2013.

[33] Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C, K, and Stanley, H, E. “PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals.” In: *Circulation*. Vol. 101. No. 23. Pp. 215–220. 2000.

[34] Alpaydin, E. “Introduction to Machine Learning.” 2nd ed. Cambridge: The MIT Press. 2010.

Dieser Tagungsband enthält die Beiträge des 28. Workshops „Computational Intelligence“ des Fachausschusses 5.14 der VDI/VDE-Gesellschaft für Mess- und Automatisierungstechnik (GMA) und der Fachgruppe „Fuzzy-Systeme und Soft-Computing“ der Gesellschaft für Informatik (GI), der vom 29. – 30.11.2018 in Dortmund stattfindet.

Der GMA-Fachausschuss 5.14 „Computational Intelligence“ entstand 2005 aus den bisherigen Fachausschüssen „Neuronale Netze und Evolutionäre Algorithmen“ (FA 5.21) sowie „Fuzzy Control“ (FA 5.22). Der Workshop steht in der Tradition der bisherigen Fuzzy-Workshops, hat aber seinen Fokus in den letzten Jahren schrittweise erweitert.

Die Schwerpunkte sind Methoden, Anwendungen und Tools für
- Fuzzy-Systeme,
- Künstliche Neuronale Netze,
- Evolutionäre Algorithmen und
- Data-Mining-Verfahren

sowie der Methodenvergleich anhand von industriellen und Benchmark-Problemen.

Die Ergebnisse werden von Teilnehmern aus Hochschulen, Forschungseinrichtungen und der Industrie in einer offenen Atmosphäre intensiv diskutiert. Dabei ist es gute Tradition, auch neue Ansätze und Ideen bereits in einem frühen Entwicklungsstadium vorzustellen, in dem sie noch nicht vollständig ausgereift sind.

ISBN 978-3-7315-0845-8