Herausgeber

F. HOFFMANN
E. HÜLLERMEIER
R. MIKUT

Dortmund | 28. – 29. November 2019

PROCEEDINGS **29. WORKSHOP** COMPUTATIONAL INTELLIGENCE

KIT Scientific Publishing

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

Proceedings. 29. Workshop Computational Intelligence

Dortmund, 28. – 29. November 2019

PROCEEDINGS **29. WORKSHOP** COMPUTATIONAL INTELLIGENCE

Dortmund, 28. – 29. November 2019

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 2019 – Gedruckt auf FSC-zertifiziertem Papier

ISBN 978-3-7315-0979-0
DOI 10.5445/KSP/1000098736

Inhaltsverzeichnis

Evaluation of Features for Change Detection in Unstructured Image Data

Friedrich Rieken Münke[1], Andreas Bartschat[1], Yujing Chen[2], Ralf Mikut[1], Markus Reischl[1]

[1] Institute for Automation and Applied Informatics
Karlsruher Institute of Technology, Karlsruhe, Germany
E-Mail: friedrich.muenke@kit.edu

[2] University of Stuttgart
Stuttgart, Germany

1 Introduction

Modern mobile devices (e.g. smartphones) are a comfortable means to gather large amounts of image data differing in place and time and can be even taken by multiple photographers. In the context of city development recently introduced deep-learning strategies make use of image data to extract objects of interest (e.g. houses, traffic, city plantation) within each image. Not only the object itself is of interest, but also its change over time, being contained in image series at other time-points (maybe taken by other users). However, images from time series of undefined imaging conditions are often not structured or systematically taken and have a high variance due to differing camera parameters (e.g. exposure, focus, positions, orientations), environment conditions (e.g. daytime, weather, light) and changes of the scene itself. Thus, the relocation of formerly found objects is challenging, even if additional GPS-coordinates are available.

In this article we evaluate features to quantify the grade of change in a detected object: Using object detection and structure from motion (SFM) techniques, objects of interest can be detected, identified and tracked over different images. With the gathered information we assess different features to describe changes

over time. We discuss and evaluate features regarding consistency and fragility with respect to camera positions, lightning conditions and grade of change. Features like Scale Invariant Feature Transform (SIFT) and KAZE Features (KAZE) are designed to be robust against different views, in this paper we discuss their suitability to detect changes of objects after the initial detection is performed with deep learning approaches.

To prove functionality, we introduce a benchmark data set that depicts an urban scene. This data set consists of unstructured images with GPS location and timestamp with a wide variety of weather conditions and camera orientations. The wide variety makes it difficult for SFM to match images. Furthermore we categorize challenges and problems forcing the algorithm to fail, such as failure of the initial object detection.

This paper serves as guideline regarding the detection of changes in unstructured image data. Assuming that a pipeline already has found objects of interest, we discuss the influences of object-changes on the representation in feature space. Therefore, we introduce the common feature-extractors SIFT, KAZE Features and Local Binary Pattern (LBP) and discuss their robustness regarding changes in objects, surroundings, image parameters etc.

We aim to:

- evaluate the performance of feature extraction strategies (SIFT, KAZE and LBP) to describe and detect changes of objects of interest in unstructured image data,
- discuss common failures in interpreting the results of feature extraction strategies if object changes occur and
- recommend best practices to cope best with object changes in unstructured image data.

In Section 2 we give a short overview about the techniques we use and the overall state of change detection. The concept of this work is described in Section 3. To evaluate the concept we build a benchmark data set, which is describe and presented in Section 4. In Section 5 the parameters and detailed

processing steps are described. The results are presented and discussed in Section 6.

2 Related Work

2.1 Overview

Change detection of objects of interest in general consists of three main steps as shown in Fig. 1. First it is necessary to use state of the art object detection algorithms to segment and classify the objects of interest in all available images [13, 14]. Afterwards all detections showing the same object are grouped, utilizing e.g. structure from motion (SFM) techniques [15, 20]. This paper focuses on the last step, where we extract features which are suitable to describe the change of the objects over time. We evaluate the feature extraction strategies for monitoring the change of unique target objects. The previous steps as shown in Fig. 1 are considered done in this paper.

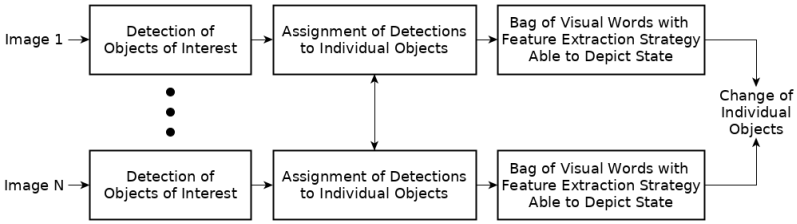
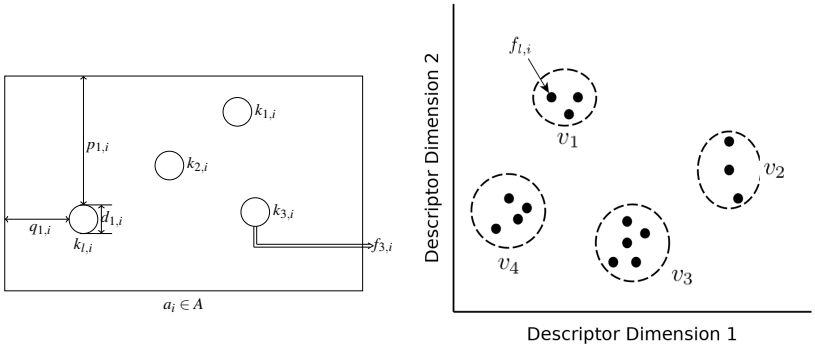


Figure 1: Pipeline to detect changes of individual objects in unstructured image data

2.2 Feature Extraction Strategy

When processing a set of images $A = \{a_1, a_2, ..., a_{N_{img}}\}$ the bag of visual words approach [5] is a method to represent an image a_i as a histogram of visual words V_i. This approach is used in [6] for unsupervised texture classification and in [17] for supervised image classification. For an image a_i a set of key points $K_i = \{k_{1,i}, k_{2,i}, ..., k_{N_K,i}\}$ is defined using a key point sampling method

(KSM). Each key point $k = (p,q,d)^T$ is defined by the location p,q on the image a_i in pixel coordinates and the size d of the key point as shown in Fig. 2a.



(a) Visualization of key points (b) Visual word dictionary creation

Figure 2: Visualization of the bag of visual words approach. a) Visualization of key points $k_{l,i} \in K_i$ and descriptors $f_{l,i} \in F_i$ on an image $a_i \in A$. b) An example of the creation of a visual word dictionary by localizing cluster centers (visual words v_w) in the set of all descriptors F. In the example the descriptors $f_{l,i}$ are $\in \mathbb{R}^2$ and the number of clusters is $N_w = 4$.

For each key point $k_{l,i} \in K_i$ a corresponding descriptor $f_{l,i}$ is computed with the descriptor computation method (DCM). A descriptor $f_{l,i}$ is defined as:

$$f_{l,i} = \begin{bmatrix} x_{l,i,1} \\ x_{l,i,2} \\ \vdots \\ x_{l,i,N_{DCM}} \end{bmatrix}_{DCM} . \tag{1}$$

As result we yield a set of descriptors $F_i = \{f_{1,i}, f_{2,i}, ..., f_{N_K,i}\}$ for a given image a_i. We define the combination of key point sampling method (KSM) and descriptor computation method (DCM) as feature extraction strategy (FES), with FES(KSM, DCM).

A visual words dictionary is created as shown in Fig. 2b. First, a target number of words N_v is selected. Then we apply k-means clustering to the set of all descriptors $F = \{F_1, F_2, ..., F_{N_{img}}\}$ to locate the N_v clusters. Every located

cluster center is called visual word v_w with $w \in N_v$ and the knowledge about all located visual words is called visual words dictionary. Every computed descriptor $f_{l,i} \in F$ is assigned to one cluster center v_w.

The feature vector V_i used to represent the image a_i is built in four steps:

1. Detecting a set of key points K_i on the image a_i using the KSM
2. Computing the set of descriptors F_i for the set of key points K_i using the DCM
3. Assigning every descriptor $f_{l,i} \in F_i$ to its closest visual word v_w in the visual dictionary
4. Counting all occurring visual words to a histogram V_i

There are already feature sampling strategies (FES) implementing key point sampling methods (KSM) and descriptor computation methods (DCM). In this paper state of the art FES such as SIFT [3] and KAZE [8] are evaluated. In addition, we evaluate the DCM opponent color histograms (OCH) [4] and local binary patterns (LBP) [2]. Since OCH and LBP do not implement a KSM, we use dense sampling to generate an artificial set of key point K_i per image a_i. Dense sampling defines a fixed grid on an image a_i with fixed key point size d [9].

We propose the bag of visual words approach to detect changes in unstructured image data, since it can be used unsupervised and has proven to perform in the similar case of unsupervised image classification. The bag of visual words approach is especially suitable in this case, since the image is reduced to visual words, which are able to represent an image without encoding local information.

2.3 Dimension Reduction and Evaluation

For the evaluation of the distribution of high dimensional data points, dimension reduction methods can be utilized. In this paper we use the state of the art Uniform Manifold Approximation and Projection (UMAP) as presented in [19]. The method is presented as a general purpose method for visualization

and preprocessing of data. The algorithm can be used supervised or unsupervised.

The Kullback-Leibler divergence as described in [1] is a measure to evaluate the similarity or distance of two probability distributions. The Kullback-Leibler divergences β can be computed for two sets of points $P^{\xi} = \{p_1, p_2, ..., p_{M^{\xi}}\}$ and $P^{\zeta} = \{p_1, p_2, ..., p_{M^{\zeta}}\}$, where M^{ξ} and M^{ζ} are the number of points in each set and each point $p_b \in P$ is a vector. If the set of points P is normal distributed, where μ is the center of P and Σ is the covariance matrix of P, the Kullback-Leibler divergence $\beta_{\xi,\zeta}$ for P^{ξ} and P^{ζ} is defined as:

$$\beta_{\xi,\zeta} = (\mu_{\xi} - \mu_{\zeta})^T \frac{\Sigma_{\xi}^{-1} + \Sigma_{\zeta}^{-1}}{2} (\mu_{\xi} - \mu_{\zeta}) + \frac{1}{2} \mathrm{sp}(\Sigma_{\xi}\Sigma_{\zeta}^{-1} + \Sigma_{\xi}^{-1}\Sigma_{\zeta} - 2I). \quad (2)$$

2.4 Change Detection

In this paper we divide changes in three categories: environmental, scenic and object-related. Environmental changes refer to changes in exposure, view point and angle, weather, daytime and seasons. Scenic changes are dominated by disappearing, appearing and moving objects within the scene. The last category of object-related changes describes the change of the state (size, color, shape, texture, ...) of an object. While a scenic change can be located on the image, object-related changes do not have a fixed location on the image. The following methods are trying to locate scenic changes and aim to be robust against environmental changes.

In [18] the changes of an urban environment are monitored. To recognize and detect changes in this environment a deconvolutional neural network (CDNet) was trained. The CDNet processes an image pair and marks areas with scene changes. Due to the architecture of the neural network only images which show the exact same scene can be processed and compared. Besides this limitation the network is able to detect changes and discriminate them from noise (e.g. exposure, weather, ...). The performance was evaluated on 152 sequences of images of the VL-CMU data set. The data set was created to evaluate a self localization system. The CDNet can not be used for the use case presented

in this paper since the necessary same position and orientation of the camera is not given. Due to the perspective change it is not possible to use image registration methods for compensation.

In [22] a recurrent convolutional neural network is used to analyze satellite images. Satellite images make it possible to survey a large area in defined time intervals. This end-to-end system is able to compare satellite images. The recurrent neural network encodes temporal dependencies and can reliably detect changes in the satellite images. In satellite images the distance and orientation of the camera is similar and small changes are corrected using image registration.

The Network architecture CosimNet as presented in [21] aims to detect changes in scenes with a siamese network. CosimNet is able to detect changes and segment them even with small viewpoint changes and is able to outperform CDNet. Large viewpoint changes still challenge the network. This approach focuses on scenic changes rather than changes of target objects.

All presented change detection approaches do not address object-related changes. Large viewpoint changes are still an issue for the presented change detection approaches. The topic of object-related changes is the focus of this paper. We propose a bag of visual words approach to detect object-related changes and in this paper we evaluate different feature extraction strategies to perform this task.

3 Concept

The aim is to detect changes of objects of interest O_k with $k \in N_O$ where N_O is the number of all observed objects in a set of unstructured images $A = \{a_1, a_2, ..., a_{N_{img}}\}$. Each image a_i with $i \in N_{img}$, where N_{img} is the number of all images in A, has a GPS location $l_i = (\phi, \lambda)$ in Latitude and Longitude and a corresponding time stamp t_i. The images in A can be divided in subsets A_j with $j \in N_{insp}$. These subsets A_j are called inspections, which is a series of images taken while moving through an area of interest.

The first two steps of change detection, as described in Fig. 1, are considered done. We recommend a Faster-RCNN frame work for object detection and SFM to reconstruct the view point of each image for assigning the separate detections to individual objects. At this point we have a set of bounding boxes (detections) D. Each bounding box (detection) $d_{k,i}$ shows an object O_k and is part of an image a_i. All bounding boxes (detections) $d_{k,i}$ assigned to one object O_k are called a set of detections $D_k = \{d_{k,1}, d_{k,2}, ..., d_{k,N_k^d}\}$, where N_k^d is the number of detections of object O_k. Since this paper focuses only on the comparison of feature extraction strategies to depict object-related changes, we assume, that the set of detections D_k for each object O_k is complete and that each detection $d_{k,i}$ is assigned to the correct object O_k. An object O_k has a set of unique states $S_k = \{s_{k,1}, s_{k,2}, ..., s_{k,N_{k,S}}\}$, where $N_{k,S}$ is the unknown number of states of the object O_k. Every detection $d_{k,i}$ shows the object O_k in a state $s_{k,z} \in S_k$, fuzzy-states are not considered in this paper. With the bag of visual words approach we extract a feature vector from a detection $d_{k,i}$ as representation $r_{k,i}$ of the state $s \in S_k$. The parameters of the process are defined in Section 5. We define the set of representations R_k of an object O_k as $R_k = \{r_{k,1}, r_{k,2}, ..., r_{k,N_k^d}\}$. A change of an object O_k from State s_ξ to State s_ζ should be noticeable in the distribution of the sets of representations R_k^ξ and R_k^ζ of both states.

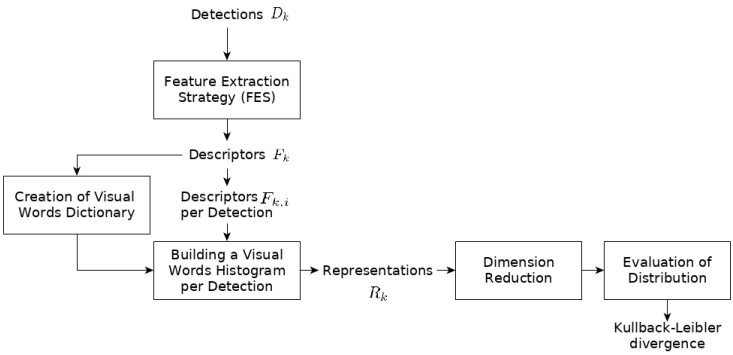


Figure 3: Evaluation of a Feature Extraction Strategy used in combination with the bag of visual words approach to detect changes in unstructured image data

This paper focuses on the performance of different feature extraction strategies (FES) used in the bag of visual words approach. The FES are evaluated separately for different objects O_k. The evaluation process of one FES is shown in Fig. 3. The bag of visual words approach (Section 2.2) is applied to all objects O_k with all feature extraction strategies (FES) to be evaluated. In the first step the FES extracts a set of descriptors $F_{k,i}$ per detection $d_{k,i}$. The set containing all available descriptors is called F_k. The visual words dictionary is created with F_k, while each representation $r_{k,i}$ is build using $F_{k,i}$. The dimensions of each representation $r_{k,i} \in R_k$ is reduced to $\hat{r}_{k,i} \in \mathbb{R}^2$ with UMAP for visual assessment. In the end the FES performance is evaluated using the Kullback-Leibler divergence between all representations $\hat{R}_k$ grouped by their states $s \in S_k$.

To the best of our knowledge, we are the first to compare the performance of SIFT, HSV-SIFT, KAZE, HSV-KAZE, OCH and LBP in a bag of visual words approach to detect object-related changes of an object O_k. The performance is tested considering differing viewing points, viewing angles, exposures, weather, daytime, motion blur and different types of object-related changes.

4 Dataset

There are many data sets dedicated to change detection [7, 11, 12]. They mostly focus on the scenic changes, where the appearance, disappearance and movement of objects are the most dominant changes. The data sets features different seasons and daytime, while having fixed or only slight changing viewpoints for each scene.

This paper focuses on object-related changes, where the state of an object itself changes. Additionally the change detection needs to be robust against environmental changes especially for different viewpoints. As a result the location of the change on the image is not important, since the viewpoint could have changed and the location of the change not necessarily reflects a change of the object state $s \in S_k$.

(a) Image: 222
24.06.19, 07:37:33

(b) Image: 223
24.06.19, 07:37:36

(c) Image: 226
24.06.19, 07:37:45

Figure 4: Sample images from inspection A_{10}

Figure 5: Example annotation of the benchmark data set A

To evaluate the performance of feature extraction strategies (FES) to separate the states $s \in S_k$ of an object of interest O_k, we introduce a benchmark data set A with $N_{img} = 1121$ and $N_{insp} = 38$. The benchmark data set A covers 300m of a street in Karlsruhe, Germany. The first inspection A_1 was collected on the 16.05.2019 and the last inspection A_{38} was collected on the 12.08.2019. Every inspection A_j is a series of images, which was collected by taking pictures in random intervals while walking down the street. Therefore an inspection A_j has constant weather, daytime and general viewing direction. Each image a_i has a resolution of 4032px $\times$ 3024px. In addition to the constant external influences during an inspection A_j the images $a_i \in A_j$ have different view points and viewing directions. Therefore the objects O_k are viewed from different angles, distances and are influenced by exposure, weather and daytime. Sample images are shown in Fig. 4. The only meta-information provided is the GPS

(a) Blooming Flower

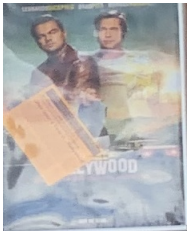
(b) Faded Flower

(c) Blooming Rose

(d) Not Bloom. Rose

(e) Poster Seniorenkino

(f) Poster Fim x Musik



(g) Poster Hollywood

(h) Poster Met Opera

Figure 6: All observed objects of interest in the data set

signal from the smartphone and the time stamp. As ground truth different states $s \in S_k$ of the objects of interest and their bounding box (detection) $d_{k,i}$ on the image are annotated, as seen in Fig. 5. It is assumed that the state $s \in S_k$ of each object O_k remains constant for each inspection A_j. The state of each object is annotated in a separate text file.

In the data set we have three objects of interest O_1, O_2 and O_3. The first object O_1 is a flower with two states $S_1 = \{0,1\}$, where 0: “Blooming” (Fig. 6a) and 1: “Faded” (Fig. 6b). The second object O_2 is a rose with the states $S_2 = \{0,1\}$, where 0: “Blooming” (Fig. 6c) and 1: “Not Blooming” (Fig. 6d). The third object O_3 is a poster stand with the states $S_3 = \{0,1,2,3\}$, where 0: “Seniorenkino” (Fig. 6e), 1: “Film x Musik” (Fig. 6f), 1: “Hollywood” (Fig. 6g) and 2: “Met Opera” (Fig. 6h). All objects changed significantly during the time of observation. The object of interest O_1 "flower" is compact. The information about the object is focused in the center of the bounding box. The observed change refers to the color difference, while the texture itself

remains constant. The second object of interest O_2 "rose" is not as compact and spread out in the bounding box. The change refers to the appearance of red blossoms. The last object of interest O_3 "poster stand" changes with different posters displayed in the same stand. In this case the bounding boxes show nearly no background. The change itself is dominant and features different colors and textures.

5 Methods

5.1 Preprocessing

The preprocessing consists of the first two steps mentioned in Fig. 1. For object detection any deep learning object detection framework (e.g. Faster RCNN) can be used. The performance of object detection algorithms depends on the type of object and the trainings dataset.

The detections $d_{k,i}$ are assigned to individual objects O_k in two steps. First, we use SIFT key point matching and the image GPS coordinates to reconstruct the view point and view angle of every image in 3D coordinates. Afterwards the true position of each detection can be estimated by projecting it into the 3D space. Detections which share the same location are assigned to a common object. With this method we can assure that even after significant changes all detections of an object are assigned to the same object.

5.2 Image Preprocessing

Since each detection $d_{k,i}$ has a different scale, due to differing distances of view point of image a_i to the object O_k, all detections $d_{k,i}$ are resized to a fixed size of 300px $\times$ 300px. This is useful if the observed object O_k does not change its actual size during the period of observation. Afterwards the detection $d_{k,i}$ is standardized to reduce the effects of exposure and daytime. The preprocessed detection $d_{k,i}$ is then processed by the bag of visual words approach (Section 2.2).

Table 1: The specifications of all tested feature extraction strategies (FES)

Abbr.	FES	KSM	DCM
sift	FES(SIFT, SIFT)	SIFT	SIFT
kaze	FES(KAZE, KAZE)	KAZE	KAZE
hsv-sift	FES(SIFT, HSV-SIFT)	SIFT	HSV-SIFT
hsv-kaze	FES(KAZE, HSV-KAZE)	KAZE	HSV-KAZE
och	FES(DENSE, OCH)	DENSE	OCH
lbp	FES(DENSE, LBP)	DENSE	LBP

Depending on the requirements of the feature extraction strategy (FES) the detection $d_{k,i}$ is transformed from RGB color space to gray scale $d_{k,i}^{gray}$ or HSV color space $d_{k,i}^{hsv}$ as described in [4].

5.3 Representation of the Object

As described in Section 3 the bag of visual words approach is used to build a representation $r_{k,i}$ for each detection $d_{k,i}$. While the feature extraction strategies are evaluated and changed, the creation of the visual words dictionary and the building of the histogram as representation is constant for all tests. We use k-means clustering to create the visual words dictionary and to build the histogram of visual words. We have empirically chosen $N_v = 400$ as size of the visual words dictionary. We evaluate six feature extraction strategies FES(KSM, DCM): *sift*, *kaze*, *hsv-sift*, *hsv-kaze*, *och* and *lbp*. The specifications of each FES are shown in Tab. 1. SIFT and KAZE both provide a native key point sampling method (KSM) and a native descriptor computation method (DCM). Both work with the processed detection $d_{k,i}^{gray}$. HSV-SIFT and HSV-KAZE are a specialisation of the basic SIFT and KAZE descriptor computation method (DCM) which is applied to the preprocessed detection $d_{k,i}^{hsv}$. OCH and LBP are descriptor computation methods without a key point sampling method. OCH uses $d_{k,i}$ to compute descriptors and LBP computes descriptors based on $d_{k,i}^{gray}$. As key point sampling method we have chosen dense sampling, where a fixed grid of key points is defined. The grid parameters were chosen empirically with a step size of 10px and a key point size of 15px.

5.4 Evaluation of Feature Extraction Strategy Performance

The representations $\hat{R}_k$ of an object O_k are built as described in Section 3. $\hat{R}_k$ contains representations $r_{k,i}$ each with a defined state $s \in S_k$, which was annotated in the benchmark dataset (Section 4). A FES is suitable to detect changes of an object O_k, when the representations of each state can be separated in feature space. The Kullback-Leibler divergence $\beta_{\xi,\zeta}$ is used to measure the distance between two sets of representations $\hat{R}_k^{\xi}$ and $\hat{R}_k^{\zeta}$ with the corresponding states s_ξ and s_ζ. We assume that $\hat{R}_k^{\xi}$ and $\hat{R}_k^{\zeta}$ are normal distributed and compute $\beta_{\xi,\zeta}$ as described in Section 2.3. When an object O_k has more than two states the Kullback-Leibler divergence is calculated for each state pair and the mean over all divergences is used measure the performance.

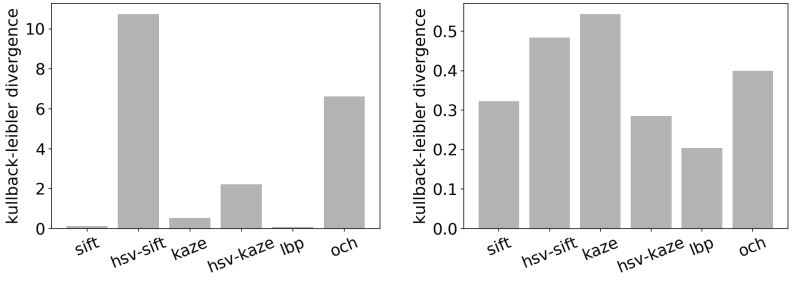
6 Results

All feature extraction strategies FES (Tab. 1) were tested on the three chosen objects of interest O_1, O_2 and O_3. The performance of all FES is evaluated as described in Section 3 and the results are shown in Fig. 7. The Kullback-Leibler divergence β of each FES is evaluated per object. Depending on the object O_k the interval of β varies. A small β represents a strong similarity for two sets of representations $\hat{R}_k^{\xi}$ and $\hat{R}_k^{\zeta}$. This is not desirable, since the change detection relies on the separation of both sets.

The β of the first object O_1 "flower" is shown in Fig. 7a. For object O_1 *hsv-sift* and *och* reach the highest β with $\beta_{hsv-sift} = 10.7$ and $\beta_{och} = 6.1$. As shown in Fig. 8 both FES are able to separate both states of object O_1. The change of the object O_1 is mainly color-related, as expected FES considering colors (*hsv-sift* and *och*) perform best. The FES *hsv-kaze* performs worse with $\beta_{hsv-kaze} = 2.2$. In the visual evaluation *hsv-kaze* still shows tendency to separate both states. All FES (*sift*, *kaze*, *lbp*) which do not consider color information, are not able to make a separation and therefore are not able to detect the change between State 0 and State 1.

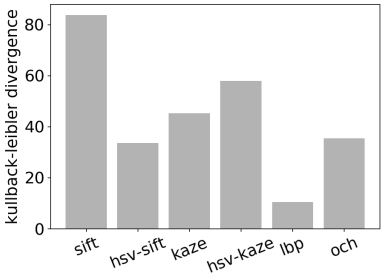
In Fig. 7b the Kullback-Leibler divergences β of all tested FES regarding the second object O_2 "rose" are presented. The highest β reached for object

O_2 is $\beta_{kaze} = 0.5$. The visual evaluation confirms that no FES was able to separate the two states 0: “Not Blooming” and 1: “Blooming” from each other. Object O_2 is due to the sparse information in the bounding box and the small changes especially difficult. In Fig. 9 the distribution of all representations $\hat{R}_2$ is shown.



(a) Kullback-Leibler divergence for all tested FES for O_1 (flower)

(b) Kullback-Leibler divergence for all tested FES for O_2 (rose)



(c) Kullback-Leibler divergence for all tested FES for O_3 (poster stand)

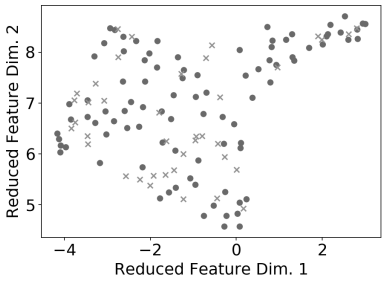
Figure 7: The performance to separate the states $s \in S_k$ of an object O_k measured in Kullback-Leibler divergence β of all tested feature extraction strategies (FES) for each object O_k

The Kullback-Leibler divergences β of the object O_3 are shown in Fig. 7c. O_3 has four states 0: Seniorenkino, 1: Film x Musik, 2: Hollywood and 3: Met Opera. Every state has dominant texture and color characteristics, as result all FES are able to separate the states from another to some degree. The minimum

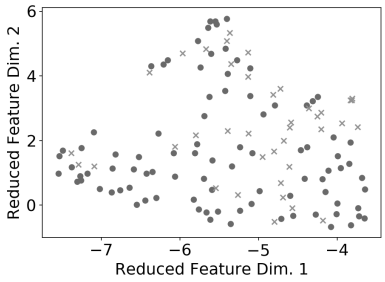
β is $\beta_{lbp} = 10.5$ and the maximum is $\beta_{sift} = 83.8$. The visual evaluation in Fig. 10 confirms that all FES are able to separate the state.

We tested the other FES combining different KSM and DCM. The FES(KAZE, SIFT) had the noticeable ability to separate the viewing/walking directions from each other. The distributions of representations R_1 and R_2 of the objects O_1 and O_2 are shown in Fig. 11. The two viewing directions (North and South) are clearly separated for both objects. The outliers in Fig. 11 were evaluated. Group A are detections of occluded objects. Group B are detections, which were taken while passing the object. This leads to a differing viewing and walking direction. Group C represents images which were taken from across the street.

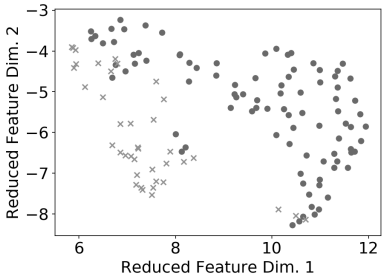
Further evaluations to measure the performance on new validation data is planed, such as analyzing the classification performance to separate different states.



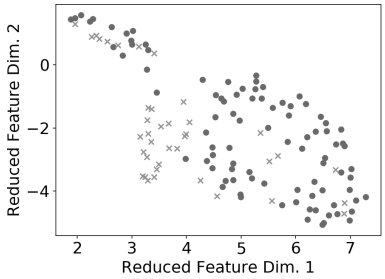
(a) Feature Extraction Strategy: *sift*



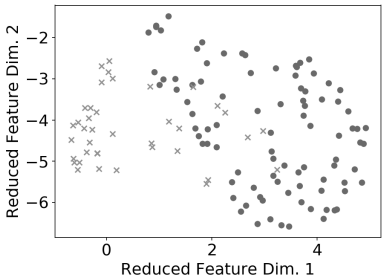
(b) Feature Extraction Strategy: *kaze*



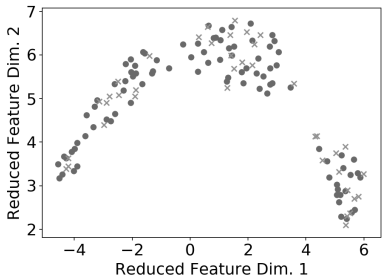
(c) Feature Extraction Strategy: *hsv-sift*



(d) Feature Extraction Strategy: *hsv-kaze*

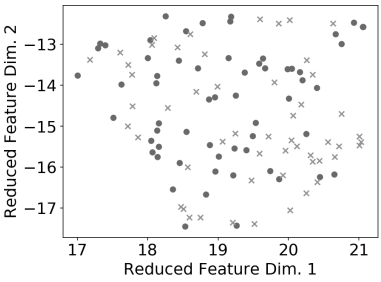


(e) Feature Extraction Strategy: *och*

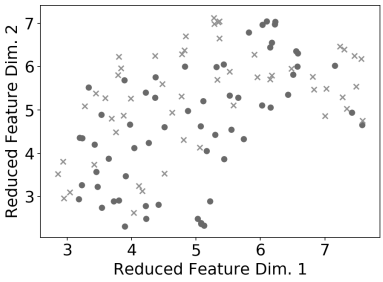


(f) Feature Extraction Strategy: *lbp*

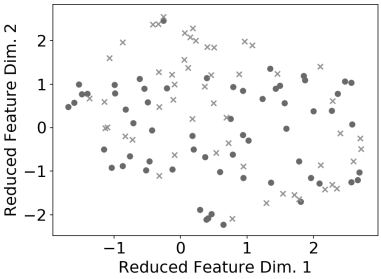
Figure 8: Distribution of representations $\hat{R}_1$ of the object O_1 (flower) in feature space [Circle: “Blooming”, Cross: “Faded”]



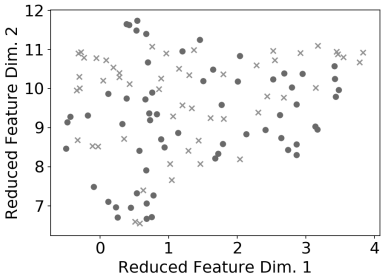
(a) Feature Extraction Strategy: *sift*



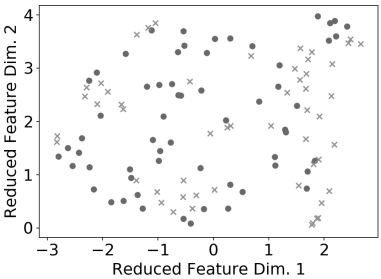
(b) Feature Extraction Strategy: *kaze*



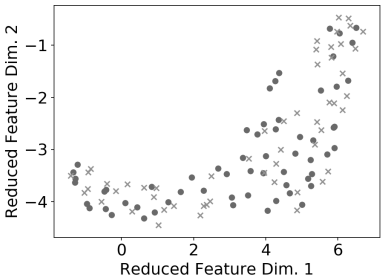
(c) Feature Extraction Strategy: *hsv-sift*



(d) Feature Extraction Strategy: *hsv-kaze*

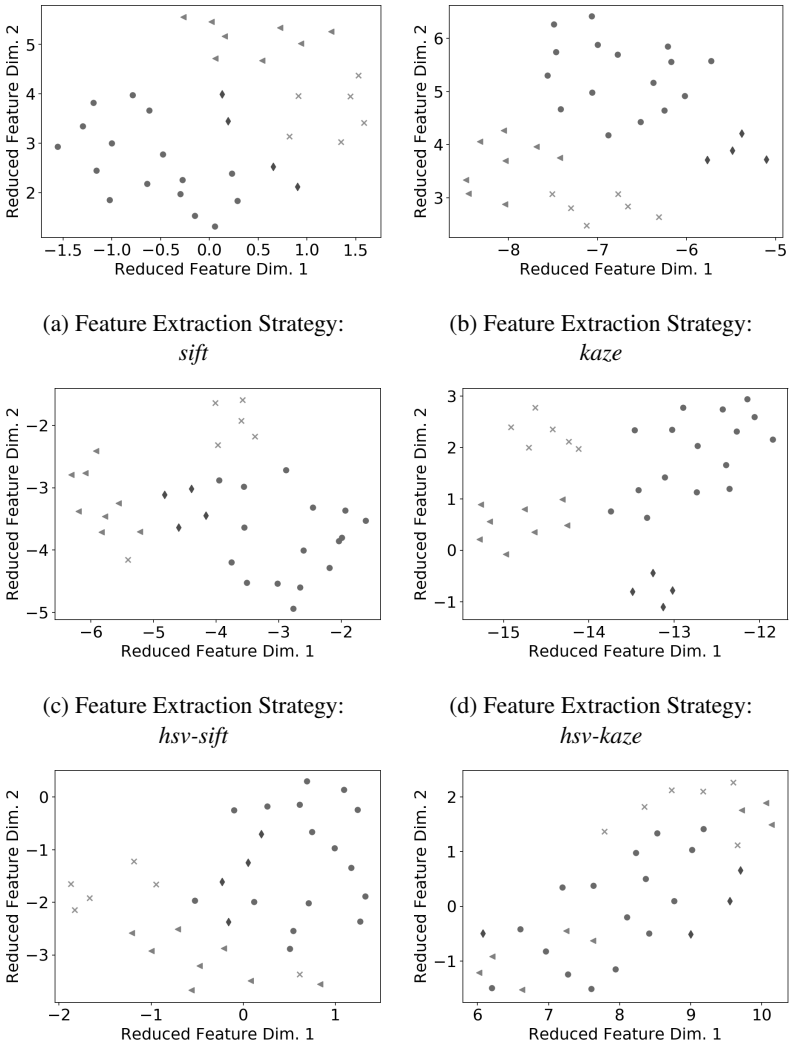


(e) Feature Extraction Strategy: *och*



(f) Feature Extraction Strategy: *lbp*

Figure 9: Distribution of representations $\hat{R}_2$ of the object O_2 (rose) in feature space [Circle: "Blooming", Cross: "Not Blooming"]



(a) Feature Extraction Strategy: *sift*

(b) Feature Extraction Strategy: *kaze*

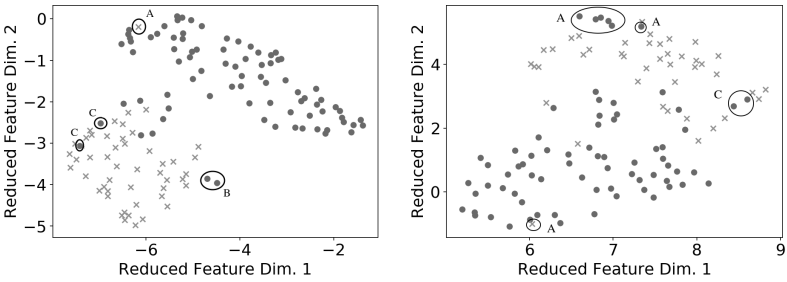
(c) Feature Extraction Strategy: *hsv-sift*

(d) Feature Extraction Strategy: *hsv-kaze*

(e) Feature Extraction Strategy: *och*

(f) Feature Extraction Strategy: *lbp*

Figure 10: Distribution of representations $\hat{R}_3$ of the object O_3 (poster stand) in feature space [Circle: "Seniorenkino", Cross: "Film x Musik", Triangle: "Hollywood", Diamond: "Met Opera"]



(a) Representations R_1 of object O_1 (b) Representations R_2 of object O_2

Figure 11: Distribution of representations R_k created with FES(KAZE, SIFT) marked by viewing direction [Circle: North, Cross: South]

7 Conclusion

In this paper we evaluate the suitability of the bag of visual words approach for object-related change detection. The results indicate that the approach is viable and able to handle unstructured image data. Overall the FES *hsv-sift*, *hsv-kaze* and *och* have proven to be robust against environmental changes and capable to detect different object-related changes. The FES *lbp* performed worst for all objects. The combination of the independent FES *kaze* and *och* are able to detect changes of an object O_k and give information whether the change is color or texture related. This combination is suitable for a wide variety of changes.

Since the change of object O_2 could not be detected, more tests with new FES are needed. Test related to other color transformations, other DCM and optimization of the parameters (e.g. number of words, clustering-method, ...) of the visual words dictionary could further improve the performance. In addition more robust representations could be built from many detections instead of one. For future projects the visual words dictionary could be evaluated to gain insights in the meaning of changes.

References

[1] S. Kullback and R. Leibler “Information and Sufficiency” In *Annals of Mathematics and Statistics* 22:79–86, 1951.

[2] T. Ojala, M. Pietikäinen and T. Mäenpää “Gray Scale and Rotation Invariant Texture Classification with Local Binary Patterns” In *European Conference on Computer Vision, LNCS* 1842:404–420, 2000.

[3] D. Lowe “Distinctive Image Features from Scale-Invariant Keypoints” In *International Journal of Computer Vision* 60(2):91–110, 2004.

[4] S. Sergyan “Color Content-based Image Classification” In *Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence and Informatics*, 5:427–434, 2007.

[5] J. Yang, Y. Jiang, A. Hauptmann and C. Ngo “Evaluating Bag-of-Visual-Words Representations in Scene Classification” In *MIR Proceedings of the International Workshop on Multimedia Information Retrieval*, 7:197–206, 2007.

[6] L. Qin, Q. Zheng, S. Jiang, Q. Huang and W. Gao “Unsupervised Texture Classification: Automatically discover and classify Texture Patterns” In *Image and Vision Computing* 26(5):647–656, 2008.

[7] H. Badino, D. Huber and T. Kanade “Visual Topometric Localization” In *Intelligent Vehicles Symposium*, 2011

[8] P. Alcantarilla, A. Bartoli and A. Davison “KAZE Features” In *European Conference on Computer Vision, LNCS* 7577(4):214–227, 2012.

[9] T. Tuytelaars “Dense Interest Points” In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2281–2288, 2013.

[10] J. Kasecka and G. Mason “Detecting Changes in Images of Street Scenes” In *Computer Vision - ACCV*, 11(4):590–601, 2013.

[11] Y. Wang, P. Jodoin, F. Porikli, J. Konrad, Y. Benezeth and P. Ishwar "CDnet 2014: An Expanded Change Detection Benchmark Data Set" In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014

[12] K. Sakurada and T. Okatani "Change Detection from a Street Image Pair using CNN Features and Superpixel Segmentation" In *BMVC* 61:1–12, 2015

[13] S. Ren, K. He, R. Girshick and J. Sun "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" In *arXiv: 1506.01497v3*, 2016

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu and A. Berg "SSD: Single Shot MultiBox Detector" In *arXiv: 1512.02325v5*, 2016

[15] J. Schönberg and J. Frahm "Structure-from-Motion Revisited" In *Conference on Computer Vision and Pattern Recognition*, 2016

[16] L. N. Thin, L. Y. Ting, N. A. Husna and M. H. Husin "GPSSystems Literature: Inaccuracy Factors and Effective Solutions" In *The International Journal of Computer Networks & Communications (IJCNC)* , 8(2):123–131, 2016.

[17] A. Bartschat, J. Stegmaier, S. Allgeier, K. Reichert, S. Bohn, O. Stachs, B. Koehler and R. Mikut "Augmentations of the Bag of Visual Words Approach for Real-Time Fuzzy and Partial Image Classification" In *Workshop Computational Intelligence, Dortmund*, 27:227–242, 2017.

[18] P. Alcantarilla, S. Stent, G. Ros, R. Arroyo and R. Gherardi "Street-View Change Detection with Deconvolutional Networks" In *Autonomous Robots*, 42:1301-1322, 2018.

[19] L. McInnes, J. Healy and J. Melville "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction" In *The Journal of Open Source Software*, 3(29):861, 2018.

[20] P. Gargallo, O. Lorentzon, Y. Noun and Y. Noutary "OpenSFM Mapillary" *https://github.com/mapillary/OpenSfM 2018*.

[21] E. Guo, X. Fu, J. Zhu, M. Deng, Y. Liu, Q. Zhu and H. Li “Learning to Measure Changes: Fully Convolutional Siamese” Metric Networks for Scene Change Detection In *arXiv: 1810.09111v3*, 2018.

[22] L. Mou, L. Bruzzone and X. Zhou “Learning Spectral-Spatial-Temporal Features via a Recurrent Convolutional Neural Network for Change Detection in Multispectral Imagery” In *IEEE Transactions on Geoscience and Rémote Sensing*, 57(2)924–935, 2019.

Learning to Aggregate: The Aggregation/Disaggregation Problem for OWA Operators of Fixed Arity

Vitalik Melnikov[1], Eyke Hüllermeier[2]

Department of Computer Science
Paderborn University, Germany
E-Mail: [1]melnikov@mail.upb.de, [2]eyke@upb.de

1 Introduction

The problem of "learning to aggregate" has recently been introduced as a novel machine learning task [7]. Roughly speaking, learning-to-aggregate (LTA) problems are supervised learning problems, in which instances are represented in the form of a composition of a (variable) number on constituents; such compositions are associated with an evaluation, score, or label, which is the target of the prediction task, and which can presumably be modeled in the form of a suitable aggregation of the properties of its constituents. As an example, consider the case where compositions are playlists consisting of constituents in the form of songs. Here, it is plausible to assume that the overall rating of a playlist is an aggregation of the evaluations of the individual songs.

An especially interesting class of learning problems arises when the evaluations of the constituents are not available at training time, and instead ought to be learned simultaneously with the aggregation function. As this involves the problem of *disaggregation* [6], we refer to this scenario as the "aggregation/-disaggregation problem".

In this paper, we tackle the aggregation/disaggregation problem for a specific type of aggregation function, namely the Ordered Weighted Averaging (OWA) operator [9]. OWA is a family of parametrized averaging functions

with several appealing properties. Although the learning of OWA operators from data has already been addressed in the literature [5, 1, 2], the problem of simultaneously learning OWA weights as well as its arguments has not been considered so far.

2 Learning the OWA Operator

Compositions (such as playlists) are normally of varying size. Here, we make the simplified assumption that all compositions are of the same size, and refer to [8] for a generalization in which this assumption is relaxed. More specifically, we proceed from the following setting: Compositions $\mathrm{vec}(c)_i$ are bags of size n but have no additional structure. Moreover, constituents are taken from a finite set

$$\mathscr{O} = \{o_1, \ldots, o_M\} \tag{1}$$

of objects, which are only identified by their name (e.g., the name of a song) but not described in terms of any other properties. Perhaps most importantly, no information about local evaluations, i.e., scores of the constituents, is supposed to be given.

An aggregation operator that appears to be suitable in this setting is the Ordered Weighted Averaging (OWA) operator [9]. For a vector of local scores $(z_1, \ldots, z_n)$, the OWA operator is defined as follows:

$$A_\lambda(\mathrm{vec}(z)) = A_\lambda(z_1, z_2, \ldots, z_n) = \sum_{i=1}^{n} \lambda_i z_{(i)}\,, \tag{2}$$

where $(\lambda_1, \ldots, \lambda_n) \in [0,1]^n$ is a weight vector such that $\lambda_1 + \ldots + \lambda_n = 1$, and $(\cdot)$ a permutation $\{1, \ldots, n\} \longrightarrow \{1, \ldots, n\}$ such that $z_{(1)} \leq z_{(2)} \leq \cdots \leq z_{(n)}$.

2.1 Learning Model

For the training data $\mathscr{D} = \{(\mathrm{vec}(c)_i, y_i)\}_{i=1}^{N}$ consisting of compositions $\mathrm{vec}(c)_i = \{c_{i,1}, \ldots, c_{i,n}\}$ together with scores y_i we assume that the constituents $c_{i,j}$ are taken from the set of objects (1). The local scores of these constituents,

$\mathrm{vec}(s) = (s_1, \ldots, s_M) \in \mathbb{R}^M$, are assumed to be unknown and therefore considered as parameters of the model. More specifically, our (noisy) OWA model assumes the following dependency:

$$\begin{aligned} y_i &= F_{\mathrm{vec}(\lambda),\mathrm{vec}(s)}\big(\{o_{i,1}, \ldots, o_{i,n}\}\big) + \varepsilon_i \\ &= \sum_{j=1}^{n} \lambda_j \, s_{i,(j)} + \varepsilon_i \ , \end{aligned} \tag{3}$$

where ε_i is an additive noise term. Moreover, $s_{i,(j)}$ is the j^{th} largest score among the constituents $\{o_{i,1}, \ldots, o_{i,n}\}$, and $\mathrm{vec}(\lambda) \in \mathbb{R}_+^n$ the (unknown) parameters of the OWA operator.

As for the learning process, our goal is to find the model parameters $\mathrm{vec}(s) \in [0,1]^M$ and $\mathrm{vec}(\lambda) \in \mathbb{R}_+^n$ minimizing (a regularized version of) the squared error loss

$$L(\mathrm{vec}(s), \mathrm{vec}(a)) = \sum_{i=1}^{N} \Big(y_i - F_{\mathrm{vec}(a),\mathrm{vec}(s)}\big(\{o_{i,1}, \ldots, o_{i,n}\}\big) \Big)^2 . \tag{4}$$

To this end, we make use of (standard) gradient-based optimization techniques, which essentially presumes an efficient way to compute the empirical risk (4). The major challenge in this regard is the sorting operation within the OWA operator. Indeed, for a sufficiently small vector $\mathrm{vec}(\varepsilon) \in \mathbb{R}^m$, the sorted sequence $\mathrm{sort}(\mathrm{vec}(s))$ is almost surely the same as for $\mathrm{sort}(\mathrm{vec}(s) + \mathrm{vec}(\varepsilon))$. The corresponding Jacobian matrix $\partial \mathrm{sort}(\mathrm{vec}(s)) / \partial \mathrm{vec}(s)$ is zero-filled almost everywhere. In our current solution, we avoid this problem by using the optimization algorithm L-BFGS [3], which approximates the gradient numerically.

3 Experimental Evaluation

As we are not aware of existing learning methods for the problem of aggregation/disaggregation, we compare our method to a simple linear (additive) model as a baseline. This model defines the overall score y_i of a composition $\mathrm{vec}(c)_i$ by the sum of the local scores of the constituents involved, normalized by the

composition size n. Thus, we formally arrive at the following normalized linear model (NLM):

$$y_i = \frac{1}{n} \sum_{j=1}^{M} \beta_j \mathbb{I}\{o_j \in \mathrm{vec}(c)_i\} + \varepsilon_i \ , \tag{5}$$

where $\mathbb{I}$ denotes the indicator function. Since the model is linear in the constituent scores β_j, these coefficients can be estimated easily by standard least squares regression. Interestingly, (5) can be obtained as a special case of the OWA model (3) for $\lambda_1 = \ldots = \lambda_n = 1/n$.

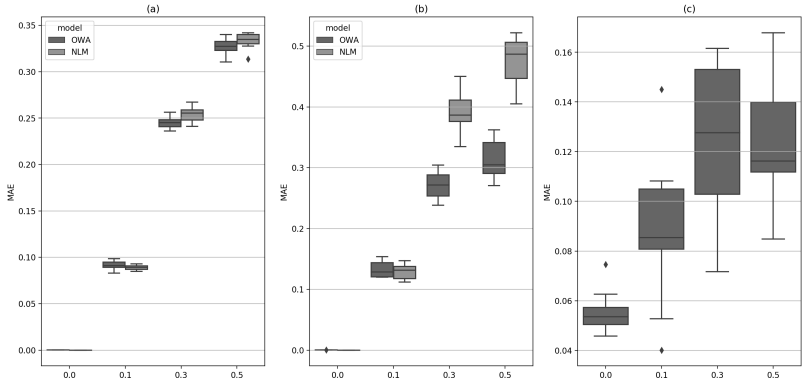


Figure 1: Mean absolute error in estimation of the ground truth parameters of the data-generating process for different noise levels σ (x-axis): (a) composition scores, (b) constituent scores, (c) OWA weight vectors $\mathrm{vec}(\lambda)$.

In the experiment with synthetic data, we assume the data to be produced according to our model (3) with normally distributed noise[1] (zero expectation and standard deviation $\sigma \in \{0, 0.1, 0.3, 0.5\}$). We generate a set of $M = 100$ constituent scores as random numbers in $[0,1]$. These scores are fixed and do not change throughout the experiment. In the second step, 1000 compositions are generated uniformly at random by sampling without replacement. In the last step, we randomly generate 100 OWA weight vectors $\mathrm{vec}(\lambda)$ of size $n = 10$, such that every λ_i is chosen independently from the unit interval, and

[1] Due to the noise, it may happen that y_i falls outside the range $[0,1]$. In this case, it is clamped at the boundary points.

the vector $\mathrm{vec}(\lambda)$ is then rescaled. The hyper-parameters of the OWA model are set to $k = 2$, $m = 21$ and $\gamma = 10^{-5}$.

Both models are trained on 500 compositions and tested on the remaining ones. Fig. 1 compares the model performance in terms of the parameter fit. The estimates of the composition scores are comparable across all noise levels, with a slight advantage in favor of the OWA model. More interesting is the comparison of the constituent score estimates. Here, the OWA model clearly outperforms the baseline, especially for higher noise levels — even then, the ground truth vectors $\mathrm{vec}(\lambda)$ are estimated quite well by OWA.

4 Summary and Outlook

We tackled the "aggregation/disaggregation problem", a novel type of machine learning problem, for the case of the OWA operator, and proposed a method for simultaneously learning the OWA parameters and evaluations of constituents. First experimental results on synthetic data are very encouraging.

One promising direction for future work is to develop a customized learning algorithm for the OWA model. To this end, a differentiable approximation of the sorting function, like the one recently proposed by [4], is required.

Acknowledgment

This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Center On-The-Fly Computing (GZ: SFB 901/3) under the project number 160364472.

References

[1] G. Beliakov. Monotone approximation of aggregation operators using least squares splines. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(6), 2002.

[2] G. Beliakov. Learning weights in the generalized owa operators. *Fuzzy Optimization and Decision Making*, 4(2), 2005.

[3] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208, 1995.

[4] M. Cuturi, O. Teboul, and J. Vert. Differentiable sorting using optimal transport: The sinkhorn CDF and quantile operator. *CoRR*, abs/1905.11885, 2019.

[5] D. Filev and R. R. Yager. Learning owa operator weights from data. In *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*, 1994.

[6] E. Jacquet-Lagreze and Y. Siskosb. Preference disaggregation: 20 years of mcda experience. *European Journal of Operational Research*, 130(2):233–245, 2001.

[7] V. Melnikov and E. Hüllermeier. Learning to aggregate using uninorms. In *Machine Learning and Knowledge Discovery in Databases*, 2016.

[8] V. Melnikov and E. Hüllermeier. Learning to aggregate: Tackling the aggregation/disaggregation problem for OWA. In *Proceedings ACML, Asian Conference on Machine Learning (Proceedings of Machine Learning Research, 101)*, 2019.

[9] R. R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1), 1988.

Methoden zur aktiven Schwingungsdämpfung und deren Anwendung in der Industriepraxis

Tobias Loose

Hochschule Heilbronn, Fakultät für Technische Prozesse
Max-Planck-Str. 39, 74081 Heilbronn
E-Mail: tobias.loose@hs-heilbronn.de

1 Einführung

Schwingungen können technisch durch passive und aktive Maßnahmen gedämpft werden. Passive Maßnahmen sind mechanisch durch Reibung realisiert, z. B. bei Fahrwerken oder Maschinen-Anschläge mit viskosen Öl-Dämpfern. Aktive Maßnahmen sind regelungstechnische Lösungen, bei denen z. B. Massen gezielt bewegt werden, um Schwingungen zu dämpfen. Anwendungsbeispiele aktiver Schwingungsdämpfungen sind bei Gebäuden, Kran-Anlagen, Regalbediengeräten, Hydraulik-Anlagen, Anbaugeräten von mobilen Arbeitsmaschinen zu finden. In diesem Beitrag werden Grundlagen der technischen Schwingungsdämpfung und eine Reihe von aktiven Dämpfungsmaßnahmen vorgestellt sowie an Modell-Anlagen implementiert. Dabei werden klassische und selbstlernende Methoden diskutiert. Klassischen Methoden sind beispielsweise modellbasierte Regler-Entwurfsverfahren. Bei den selbstlernenden Methoden werden Ideen und Realisierungen vorgestellt, indem automatisiert Reglerparameter im laufenden Betrieb einer Anlage eingestellt werden.

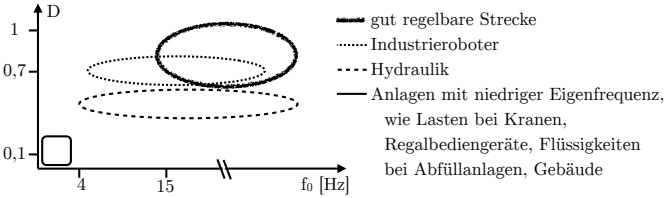


Bild 1: Einordnung schwingungsfähiger Anlagen hinsichtlich Dämpfungsgrad D und Eigenfrequenz des ungedämpften Systems $f_0 = \omega_0/(2\pi)$ [6].

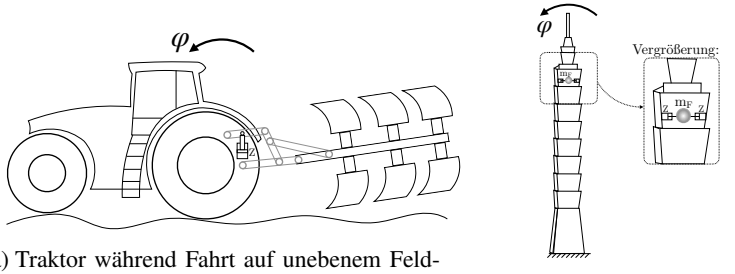
2 Dämpfung technischer Systeme und industriepraktische Anwendungen

Durch Übertragungsfunktionen bzw. Differentialgleichungen (DGL) höherer Ordnung können reale mechatronische Systeme beschrieben werden. Alle mechanischen Systeme haben eine Masse m und deren Materialien haben alle eine bestimmte Elastizität c. Elektrische Systeme besitzen alle eine Induktivität L, die durch den Aufbau eines Magnetfeldes bei einem Stromfluss i entsteht (außer Anwendungen mit Supraleiter). Wenn beispielsweise ein Elektromotor eine Masse m antreibt, so entsteht zwischen der Induktivität L und der Massenträgheit m ein System von mindestens zweiter Ordnung. Diese Systeme koppeln damit zwei unterschiedliche Energieformen, womit sie prinzipiell schwingungsfähig sind. Sie werden durch die Übertragungsfunktion

$$G_S(s) = \frac{K_S \cdot \omega_0^2}{s^2 + 2 \cdot D \cdot \omega_0 + \omega_0^2} \tag{1}$$

mit dem Verstärkungsfaktor K_S, dem Dämpfungsgrad D und der Eigenkreisfrequenz des ungedämpften Systems ω_0 beschrieben, siehe Bild 1.

Natürlich gibt es einige Systeme, bei denen z. B. aufgrund von Reibungen ein großer Dämpfungsgrad $D > 1$ vorherrscht und damit keine Schwingungen auftreten. Des Weiteren gibt es Systeme, bei denen ein charakteristischer Parameter bzw. eine Energieform vernachlässigbar klein ist und somit in der Praxis keine Schwingungen auftreten, weil sie damit weit unterhalb der Eigenfrequenz betrieben werden, z. B. bei kleinen Elektromotoren mit vernachlässigbar kleiner Induktivität. Daneben gibt es eine Vielzahl von indus-



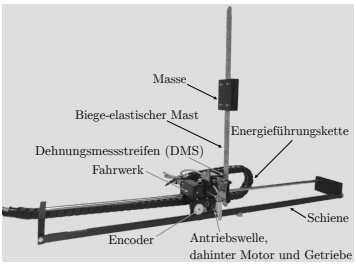
(a) Traktor während Fahrt auf unebenem Feldweg, der Zylinder Z hebt bzw. senkt den Pflug

(b) Wolkenkratzer mit beweglicher Masse m_F

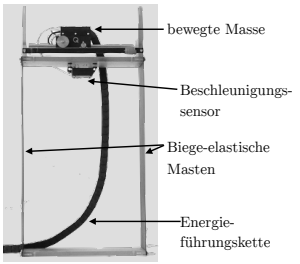
Bild 2: Schwingungsfähige Anlagen und Anwendungen aktiver Dämpfung

triepraktischen Systemen, die ein ausgeprägtes Schwingungsverhalten mit einem Dämpfungsgrad $D < 1$ und relativ niedrigen Eigenfrequenz $f_0 = \omega_0/(2\pi)$ haben, z. B. Werkzeugmaschinen wie Pressen, mobile Arbeitsmaschinen, siehe Bild 1 und 2a. Ein Traktor als mobile Arbeitsmaschine neigt bei der Fahrt über einen unebenen Weg zu Schwingungen, aufgrund der Elastizität des Fahrwerks und der Masse des Anbaugerätes, z. B. ein Pflug. Die Schwingungen sind mitunter so stark, dass die Vorderräder abheben können. Hierzu existieren Systeme, die die Schwingung durch den Hydraulikzylinder Z aktiv dämpfen können, siehe [8]. Weitere Beispiele sind das Erreichen von Endlagen, z. B. das schwingungsfreie Schließen von Türen einer Werkzeugmaschine. Ein anschauliches Beispiel ist zudem das Schwingen von Gebäuden, siehe Skizze in Bild 2b in Anlehnung an „Taipeh 101“. Durch geeignete aktive oder passive Maßnahmen sind diese Anwendungen zu dämpfen.

Aktive Maßnahmen dämpfen aktiv durch ein Regler betätigtes Stellglied. Passive Maßnahmen dämpfen durch mechanisch eingebrachte Reibungen, üblicherweise mit viskosen Öl-Dämpfern. Die Anwendungen sind hier durch Labormodelle im kleinen Maßstab nachgestellt, um die Methoden zu testen und zu zeigen, siehe Bild 3 und 4. Die Traktor-Schwingung wird hier mit dem vertikalen Schwingungssystem nachgeahmt, siehe Bild 3a.

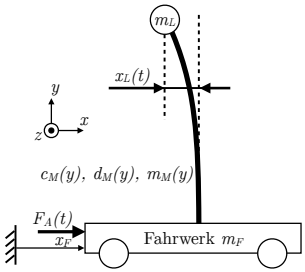


(a) Modell eines vertikalen Schwingungssystems

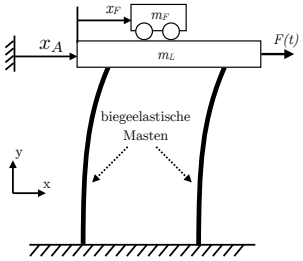


(b) Modell eines „Wolkenkratzers“

Bild 3: Realisierung von Labormodellen [6]



(a) Prinzip eines Labormodells „Vertikalschwingung“



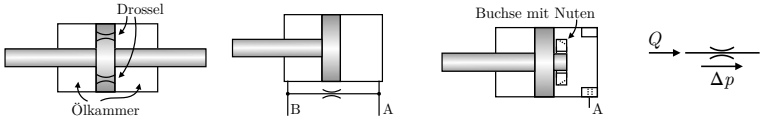
(b) Prinzip eines Labormodells „Wolkenkratzer“

Bild 4: Prinzipien der Labormodelle [6]

3 Lösungen zur Dämpfung technischer Systeme

3.1 Passive Maßnahmen

Passive Maßnahmen zur Schwingungsdämpfung werden über mechanische und / oder fluidische Reibung realisiert. Wegen der hohen Leistungsdichte werden oftmals hydraulische Dämpfer eingesetzt, bei denen die beiden Kammern über Drosseln verbunden sind, siehe Bild 5a. Daraus ist das in der Regelungstechnik verwendete Dämpfer-Symbol ⊟⊢ abgeleitet. Früher wurden Industrie-Hydraulikanlagen teilweise über fest eingestellte Drosseln gedämpft, indem z. B. die Zylinderkammern A und B über einen Bypass verbunden sind, siehe



(a) Öl-Dämpfer (b) Bypass-Dämpfung (c) Endlagendämpfung (d) Drossel

Bild 5: Prinzip passiver Dämpfung mit viskoser Reibung durch Volumenstrom Q und Druckverlust Δp

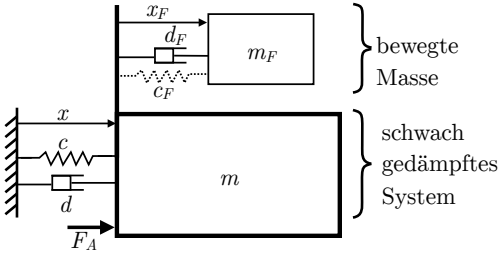
Bild 5b. Bei einfachen Hydraulik-Anlagen gibt es auch die sog. Endlagendämpfung, bei der der Öl-Volumenstrom über eine Buchse mit Nuten gedrosselt wird, siehe Bild 5c. Bei modernen Lösungen wird hingegen die Anlage über das Proportionalventil aktiv gedämpft und das Bremsen beim Heranfahren an die Endlagen wird aktiv über das Proportionalventil geregelt, siehe auch [2, 5]. In der Industriepraxis gibt es zahlreiche Beispiele, bei denen Öl-Dämpfungen zum Einsatz kommen, z. B. beim Schließen von Türen und Klappen, Puffer bei Eisenbahnen. Eine weitere, berühmte Anwendung von Öl-Dämpfern ist das Auto-Fahrwerk. Bei hochwertigen Lösungen werden Bypass-Drosseln je nach Fahrsituation adaptiert, siehe [10].

Durch eine Drossel entsteht ein Volumenstrom Q aufgrund einer Druckdifferenz Δp, es gilt $Q^2 \sim \Delta p$, siehe Bild 5d und [2]. Bei der viskosen Dämpfung ist somit die Dämpfungskraft $F_d \sim v^2$ proportional zum Quadrat der Geschwindigkeit v bzw. als linearisierte Gleichung $F_d = d \cdot v = d \cdot \dot{x}$. Die Differentialgleichung

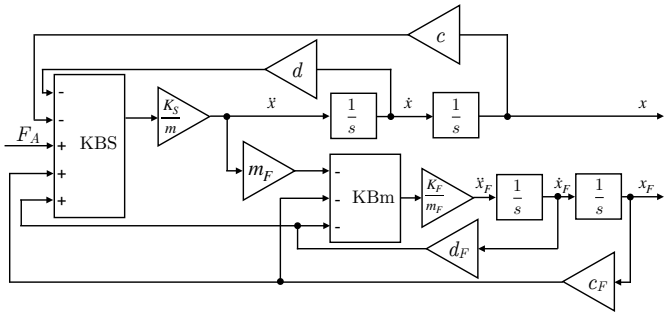
$$m \cdot \ddot{x} + d \cdot \dot{x} + c \cdot x = F_A \tag{2}$$

$$\ddot{x} + 2 \cdot D \cdot \omega_0 \cdot \dot{x} + \omega_0^2 \cdot x = \omega_0^2 \cdot F_A \tag{3}$$

beschreibt ein einfaches gedämpftes Feder-Masse-System mit den Parametern $\omega_0 = \sqrt{c/m}$ und $D = 1/2 \cdot d \cdot \sqrt{m/c}$, siehe Bild 6a allerdings ohne bewegte Masse mit $m_F = 0$. Eine passive Dämpfungsmethode ist es, weitere Dämpfer einzubauen, um d zu vergrößern (was teilweise bei Gebäuden durch in Verstrebungen angebrachten Dämpfern gemacht wird). Eine weitere Möglichkeit besteht darin, eine zusätzliche bewegte Masse m_F über einen Dämpfer d_f an das System anzubringen, siehe Bild 6a. Die (schwache) Feder c_F dient lediglich



(a) Modellskizze



(b) Simulation als Blockschaltbild

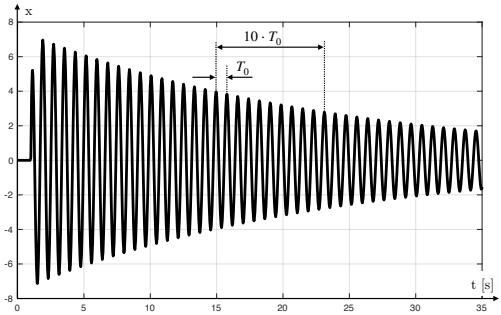
Bild 6: Vereinfachte Modellskizze mit einem schwach gedämpften System m (= Wolkenkratzer) und einer beweglichen Masse m_L(links in 6a) und dessen Simulation als Blockschaltbild (rechts in 6b)

einer Rückstellung und Zentrierung der Masse m_F. Durch einen Kräftefreischnitt kann das DGL-System

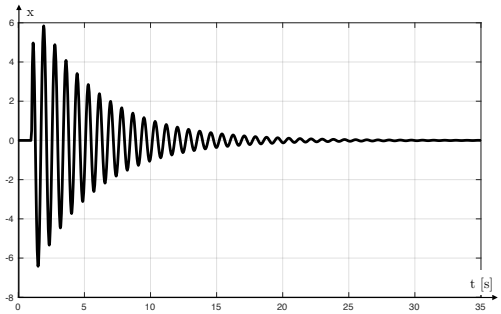
$$m \cdot \ddot{x} + d \cdot \dot{x} + c \cdot x - d_F \cdot \dot{x}_F - c_F \cdot x_F = F_A \tag{4}$$

$$-m_F \cdot \ddot{x} - d_F \cdot \dot{x}_F - c_F \cdot x_F = m_F \cdot \ddot{x}_F \tag{5}$$

bestimmt und simuliert werden, siehe Bild 6b. Interessant und logisch nachvollziehbar ist es, dass eine derartige passive Maßnahme gewisse Grenzen hat, siehe Ergebnisse in Bild 7 im Vergleich zu den Ergebnissen einer aktiven Dämpfung in Abschnitt 3.2. Aus praktischen Gesichtspunkten kann die zusätzliche, bewegte Masse m_F in Relation zur Systemmasse m nicht beliebig groß werden. Sie ist üblicherweise viel kleiner als die Systemmasse m. Der zusätzliche Dämpfer d_F darf dann nicht zu klein und nicht zu groß werden,



(a) Ohne bewegte Masse, d. h. ohne (zusätzliche) passive Dämpfung mit $d_F = c_F = 0$ (Dämpfungsgrad $D \approx 0,0056$)

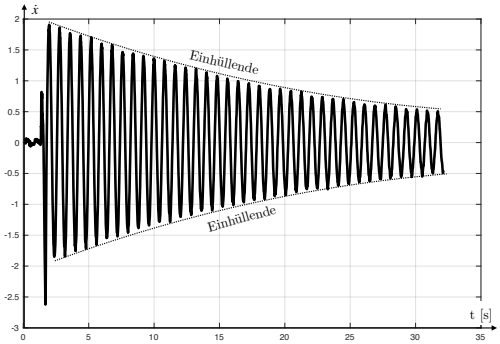


(b) Mit beweglicher Masse m_F als passive Dämpfung und nahezu optimale Dämpfung d_F (Dämpfungsgrad $D \approx 0,0274$)

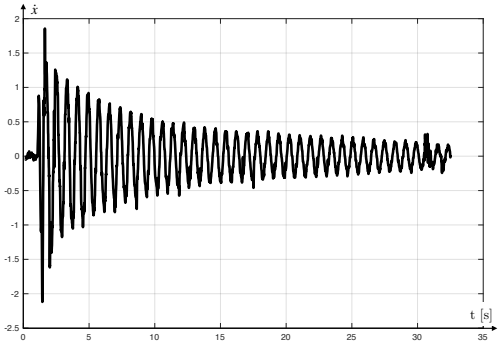
Bild 7: Simulationsergebnisse mit Modell „Wolkenkratzer“

so dass die Masse m_F eine maximal mögliche Relativgeschwindigkeit zum System erreicht und die Dämpfung so groß wie möglich wird. Ist der Dämpfer d_F zu klein, dann wir die Schwingungsenergie zu wenig abgebaut. Bei zu großem Dämpfer d_F ist die Masse m_F zu stark am System gekoppelt, so dass die Relativgeschwindigkeit klein ist und damit zu wenig Schwingungsenergie umgewandelt wird.

Mit dem Labormodell „Wolkenkratzer“ lässt sich dieses Ergebnis gut demonstrieren. Die Ausgangslage ist das schwingungsfähige System ohne bewegte Masse mit einem Dämpfungsgrad $D \approx 0,0056$, siehe Bild 8a. Mit der zusätz-



(a) Ohne zusätzliche Dämpfung, Dämpfungsgrad $D \approx 0,0056$



(b) Mit beweglicher Masse m_F als passive Dämpfung durch Reibung, erzielter Dämpfungsgrad $D \approx 0,021$

Bild 8: Messungen mit Labormodell „Wolkenkratzer“

lichen, bewegten Masse m_F ist eine passive Dämpfung realisiert, der Dämpfungsgrad ist mit $D \approx 0,021$ deutlich um den Faktor $3,75$ größer geworden, wobei das Gesamtsystem absolut immer noch schwach gedämpft ist. Die bewegte Masse ist dabei passiv, d. h. sie wird durch keine Motorkraft betrieben. Die dadurch gewonnene Dämpfung resultiert lediglich aus der Reibung, die zwischen System und Masse m_F auftritt (Lagerreibungen, Getriebereibung, usw.). Über die allgemeine Lösung z. B. der Impulsantwort

$$x(t) = K^* \cdot e^{-D \cdot \omega_0 \cdot t} \cdot \sin\left(\omega_0 \cdot \sqrt{1 - D^2} \cdot t\right) \tag{6}$$

einer DGL aus Gleichung (3) lässt sich der Dämpfungsgrad

$$D = \frac{1}{\omega_0} \cdot \frac{\ln(x_1(t_1)) - \ln(x_2(t_2))}{t_2 - t_1} \tag{7}$$

aus der Einhüllenden, d.h. mit zwei Stützpunkten $x_1(t_2)$ und $x_2(t_2)$ (zwei lokale Maxima oder Minima) zu den Zeitpunkten t_1 und t_2, bestimmen.

Die ungedämpfte Eigenfrequenz $f_0 = 1/T_0$ lässt sich bei schwach gedämpften Systemen näherungsweise aus dem Kehrwert der Schwingungsperiodendauer T_0 bestimmen, siehe Bild 7a. Zur Verbesserung der Ablesegenauigkeit ist es empfehlenswert über mehrere Perioden deren Dauer abzulesen, z.B. über 10 Perioden und daraus hier $T_0 \approx 0,8$ Sekunden zu bestimmen. Die (ungedämpfte) Anlage hat somit die Eigenfrequenz $f_0 = 1/T_0 \approx 1,25$ Hz bzw. die Eigenkreisfrequenz $\omega_0 = 2 \cdot \pi \cdot f_0 \approx 7,85\,\mathrm{s}^{-1}$.

3.2 Aktive Maßnahmen

Bei aktiven Dämpfungsmaßnahmen wird über ein Regler und Stellglied aktiv eine zusätzliche Kraft in das zu dämpfende System eingeleitet. Aus der allgemeinen DGL eines schwingungsfähigen Systems in Gleichung (3) ist ersichtlich, dass proportional zur Geschwindigkeit $\dot{x}$ die Dämpfung erzielt wird. Diese Erkenntnis lässt sich bereits als aktive Maßnahme umsetzen, indem eine Kraft proportional entgegen der Geschwindigkeit eingeleitet wird.

Ein Regelkreis zur Dämpfung von Systemen lässt sich formal mit der Eingangsgröße Null aufbauen, d. h. keine Schwingungen sollen auftreten, siehe Bild 9 und [1, 11] sowie Grundlagen der Regelungstechnik in [3, 7, 9]. Der Regler hat die Übertragungsfunktion G_R und gibt als Stellgröße y beispielsweise ein Spannungssignal für den Elektromotor aus. Der Motor liefert eine Kraft F, wobei er vereinfacht bei vernachlässigter Induktivität durch die Übertragungsfunktion

$$G_M = \frac{K_M \cdot s}{T_M \cdot s + 1} \tag{8}$$

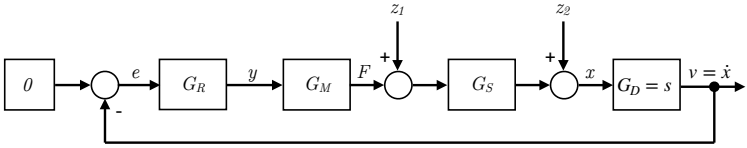


Bild 9: Allgemeiner Regelkreis zur aktiven Schwingungsdämpfung mit der Übertragungsfunktion des Reglers G_R, des Stellgliedes bzw. Elektromotors G_M, des zu dämpfenden Systems G_S und einem Differenzierglied G_D.

beschrieben werden kann, siehe auch [4, 12]. Die Übertragungsfunktion beschreibt das Motorverhalten, indem eine Kraft durch *Änderung* der Spannung erreicht wird (das s im Zähler) und eine „Anfahr-“Kraft zur Beschleunigung auf eine bestimmte Geschwindigkeit auftritt (Nenner).

Die Strecke G_S ist das schwach gedämpfte System mit der Kraft F als Eingangs- und Position x als Ausgangsgröße, siehe Gleichung (1). Durch die Ableitung $G_D = s$ wird die Geschwindigkeit zurück geführt (ebenso vereinfacht durch Vernachlässigung von Verzögerungszeiten beim Ableiten). Störgrößen können u. a. als Kräfte wirken (z_1) oder auf die Position als Auslenkung (z_2). Im Folgenden wird die Auslenkung als Störgröße z_2 betrachtet.

Durch die Störübertragungsfunktion

$$G_{z2} = \frac{\mathscr{L}\{v\}}{\mathscr{L}\{z_2\}} = \frac{s}{1 + G_R \cdot G_M \cdot G_S \cdot G_D} \tag{9}$$

lässt sich zeigen, dass es mehrere Reglereinstellungen gibt, um die Störung zu Null auszugleichen (z. B. mit dem Grenzwertsatz für $t \to \infty$ wird $s \to 0$).

Ein reiner I-Regler führt zu der Übertragungsfunktion

$$G_{z2,I} = \frac{s(T_M s + 1)(\frac{1}{\omega_0^2} s^2 + \frac{2D}{\omega_0} s + 1)}{\frac{T_M}{\omega_0^2} s^3 + (\frac{2D}{\omega_0} T_M + \frac{1}{\omega_0^2}) s^2 + \underbrace{(T_M + \frac{2D}{\omega_0} + K_I K_M K_S)}_{\text{Dämpfungsterm}} s + 1} \tag{10}$$

mit einem Dämpfungsterm, womit der I-Anteil K_I den Dämpfungsgrad einstellt, siehe Messergebnisse in Bild 10.

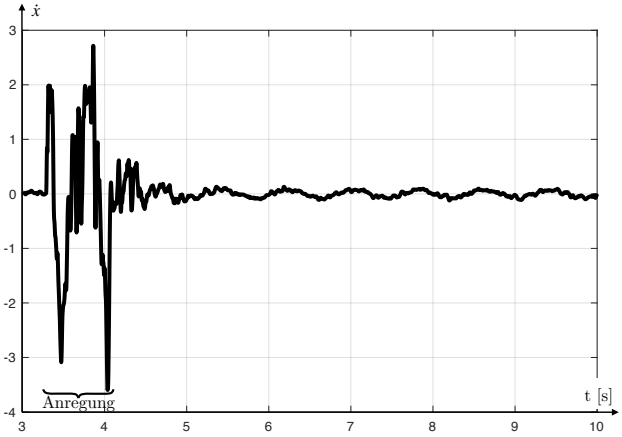


Bild 10: Aktive Dämpfung des Labormodells „Wolkenkratzer“ mit einem I-Regler.

Die Polstellen der Strecke aus Gleichung (1) lassen sich auch mit einem PID-Regler

$$G_{R,PID} = K_P + \frac{K_I}{s} + K_D \cdot s = K_P \cdot \frac{1 + T_N \cdot s + T_N \cdot T_V \cdot s^2}{T_N \cdot s} \tag{11}$$

kompensieren, indem die Nachstellzeit $T_N = 2D/\omega_0$ und Vorhaltzeit $T_V = 1/(2D\omega_0)$ auf die Strecke angepasst werden. Das führt theoretisch zu einem nicht schwingungsfähigen System mit der Gesamtübertragungsfunktion

$$G_{z2,komp} = \frac{s \cdot (T_M \cdot s + 1)}{\underbrace{(T_M + \frac{K_P \cdot K_M \cdot K_S \cdot \omega_0}{2 \cdot D})}_{\text{Verzögerungszeit}} \cdot s + 1} \tag{12}$$

und *einer* (reellen) Verzögerungszeit. Allerdings ist die Übertragungsfunktion nicht realisierungsfähig, aufgrund des größeren Grades im Zähler gegenüber dem im Nenner. In der Praxis ist es zudem so gut wie unmöglich die Streckenpole mit dem Regler exakt „zu treffen“, siehe Ergebnis in Bild 11 im Vergleich zum ungedämpften sowie passiv gedämpften System in Bild 8.

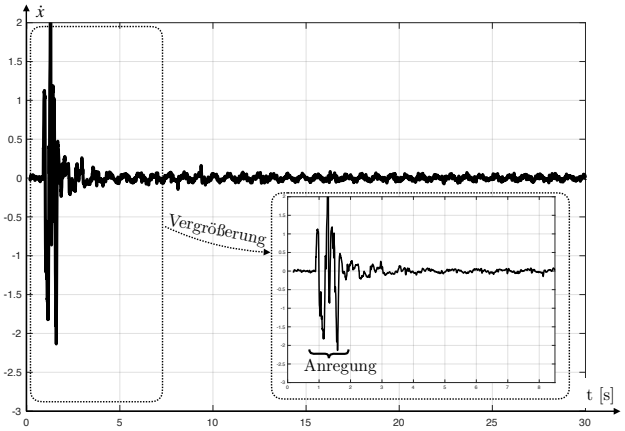
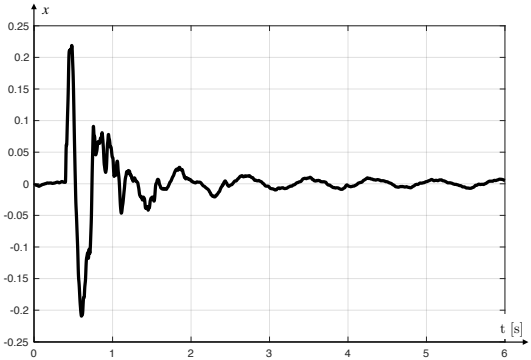


Bild 11: Aktiv geregelte Dämpfung, erzielter Dämpfungsgrad $D \approx 1$.

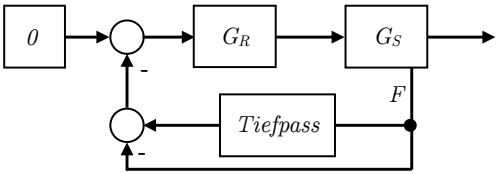
Zudem führt Messrauschen zu teilweise großen Stellgrößen durch den P- und D-Regleranteil, so dass die Anlage „nervös“ wird. Aus technischer Sicht muss daher besonders geprüft werden, ob ein Dämpfungsgrad $D = 1$ bzw. eine vollständige Schwingungskompensation über eine aktive Dämpfung sinnvoll ist. Insbesondere bei schnell ändernden Störgrößen kann dabei die Mechanik der Stellglieder extrem stark beansprucht werden.

Aufgrund der Übertragungsfunktion G_M aus Gleichung (8) des Elektromotors mit seinem differenzierenden Anteil (s im Zähler, technisch liefert der E-Motor eine Kraft bei einer Spannungs*änderung*), ist es auch möglich die Position x zurückzuführen und mit einem P-Regler die aktive Dämpfung zu erzielen, siehe Bild 12a. Der geschwindigkeitsproportionale Eingriff in das System bleibt damit dennoch bestehen. Der Vorteil dabei ist, dass das reale Sensorsignal ggf. nicht differenziert werden muss und somit der Rauschanteil im Signal geringer ist.

Eine weitere Möglichkeit zur aktiven Dämpfung ist die Zustandsregelung, siehe [4, 11]. Aufgrund der aufwändigen Anpassung bzw. Justierung mit realen Anlagen, wurde sie in diesem Beitrag nicht betrachtet. Eine weitere Lösung ist die Rückführung von Kraftspitzen, die bei der Dämpfung von Anbaugeräten mobiler Arbeitsmaschinen angewandt wird, siehe Bild 12b und [8].



(a) Ergebnis mit Positionsrückführung und P-Regler



(b) Rückführung von Kraftspitzen zur Dämpfung, siehe [8].

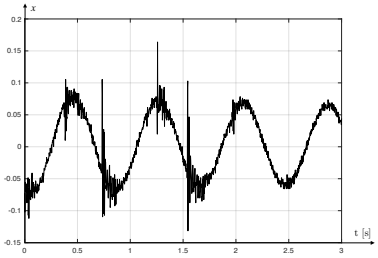
Bild 12: Lösungen zur aktiven Schwingungsdämpfung

4 Implementierung und automatisierte Parametrierung

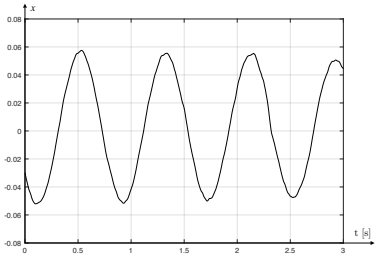
Vor der Implementierung sind oftmals Sensordaten aufzubereiten, indem (hochfrequentes) Messrauschen herausgefiltert wird, siehe Bild 13a. In diesem Beitrag wurden pragmatisch P-T_1 Glieder

$$G_{Filter} = \frac{1}{(T_F \cdot s + 1)^n} \tag{13}$$

als Tiefpassfilter verwendet, z. B. mit der Filterverzögerungszeit $T_F = 0,01$ und dem Exponenten $n = 3$. Bei der Wahl eines Filters ist es bei den hier vorgestellten Anwendungen wichtig, das hochfrequente Messrauschen zu unterdrücken und gleichzeitig die Phasenverschiebung des gefilterten Signals so

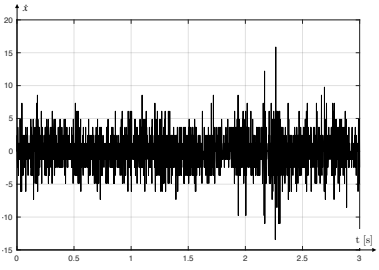


(a) Original Sensor-Messdaten (Rohdaten)

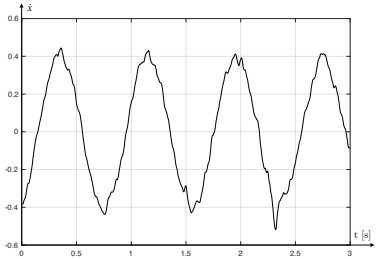


(b) Durch Filter aufbereitete Daten

Bild 13: Roh- und Filterdaten



(a) Zeitliche Ableitung der Rohdaten



(b) Zeitliche Ableitung gefilterter Daten

Bild 14: Abgeleitete Roh- und Filterdaten

klein wie möglich zu halten, siehe Bild 13b. Das Messrauschen verursacht insbesondere bei zeitlichen Ableitungen (D-Regler oder Berechnung der Geschwindigkeit) zu unbrauchbaren Stellsignalspitzen, siehe Bild 14a. Durch eine zu groß gewordene Phasenverschiebung kann der Regler nicht mehr zeitnah der Schwingung entgegen wirken. Ein Sensorsignal, wie in Bild 14a gezeigt, ist für eine Regelung nicht brauchbar und muss entsprechend aufbereitet werden, wie es in Bild 14 gezeigt ist.

Eine weitere Schwierigkeit bei der Verarbeitung realer Sensordaten ist die Sensordrift, die insbesondere durch zeitliche Integration auftreten kann, siehe Bild 15. Die Integration ist notwendig, wenn z. B. die Beschleunigung gemessen wird und die Geschwindigkeit damit berechnet werden soll. Durch

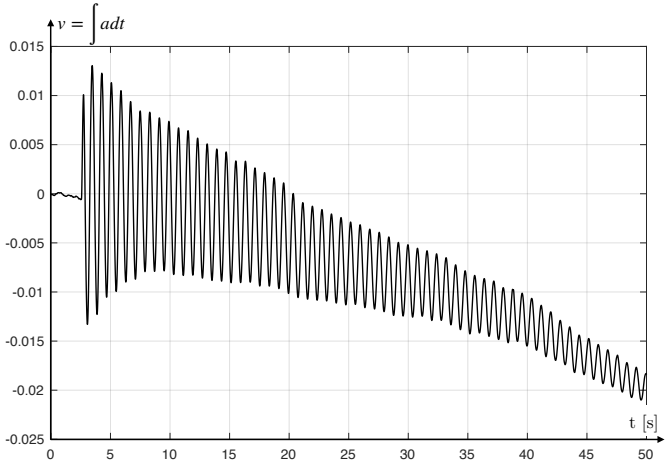


Bild 15: Drift beim Integrieren.

Addition eines Offsets vor der Integration kann die Drift als pragmatische Lösung beseitigt werden.

Zur erfolgreiche Implementierung eines Reglers ist eine gewisse Expertise notwendig, z. B. die Filterung von Messdaten oder die Auswahl eines geeigneten Reglers inkl. dessen Parametrierung wie in Abschnitt 3 gezeigt. Zur Unterstützung bei der Inbetriebnahme und insbesondere zum Finden guter bzw. optimaler Reglerparameter wird in diesem Beitrag eine automatisierte Parameteroptimierung gezeigt. Dabei wird die reale Anlage mit einem Regler betrieben und ein Algorithmus justiert dessen Parameter während dem Betrieb. Anhand eines Gütekriteriums kann z.B. mit einem Gradientenabstiegsverfahren eine passende Reglerparametrierung gefunden werden. Die nachfolgenden Ergebnisse sind anhand des Gütekriteriums

$$Q^2 = \int_{t_0}^{t_1} e^2(t) \cdot dt \tag{14}$$

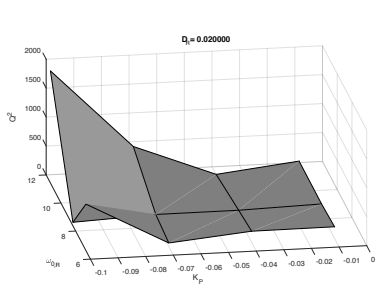
mit der quadratischen Regeldifferenz e^2 (siehe Bild 9) über ein festgelegtes Zeitintervall zwischen t_0 und t_1 erzielt worden. Das Ziel ist nun, durch Einstellung der Regler-Parameter das Gütekriterium Q^2 zu minimieren. Die Anlage kann dabei durch fest vorgegebene Störfunktionen angeregt werden, z. B. in-

dem der Elektromotor kurz, sprunghaft oder durch eine Sinus-Funktion mit der Eigenkreisfrequenz ω_0 der Anlage angeregt wird. Für reale, insbesondere industriepraktische Auslegungen sind die Anlagen allerdings auch durch randomisierte Störfunktionen anzuregen, um den Regler in Feldversuchen zu testen.

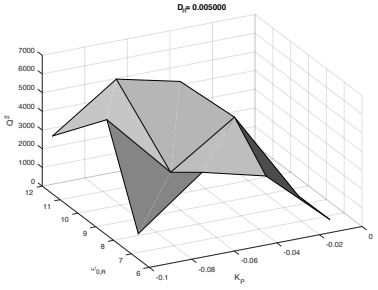
In diesem Beitrag sind die drei PID-Regler-Parameter mit der Reglerverstärkung K_P, der Nachstellzeit T_N und der Vorhaltzeit T_V automatisiert optimiert worden. Die Verzögerungszeit des realen D-Reglers ist dabei fest angenommen worden, kann aber für weitere Arbeiten auch noch optimiert werden. Der PID-Regler soll dabei als Schwingungskompensationsregler ausgelegt werden, siehe Gleichung (11) mit der Nachstellzeit $T_N = 2D/\omega_0$ und Vorhaltzeit $T_V = 1/(2D\omega_0)$, die auf die Strecke anzupassen sind. Daher wird im Folgenden, zur besseren Interpretation, die Regler-Parameter in K_P und $\omega_{0,R}$ sowie D_R als „Suchraum“ vorgegeben (die dann in T_N und T_V umgerechnet werden können).

Die Anlage wurde hier durch klassisches Ingenieur-Vorgehen zur Modellbildung und Parameteridentifikation bereits untersucht, siehe Abschnitt 3. Darin wurde ein Dämpfungsgrad $D \approx 0,0274$ und eine Eigenkreisfrequenz $\omega_0 \approx 7,85\ \mathrm{s}^{-1}$ ermittelt. Durch die nachfolgend gezeigte automatisierte Suche konnten diese Parameter ebenso gefunden werden.

Wurden durch den Algorithmus ein zu kleiner Dämpfungsgrad D_R angenommen, führte das zu schlechten Ergebnissen. Das Gütekriterium nahm Werte von über 1.000 an, siehe Bild 16. Interessant ist, dass trotz kleinen Werten von D_R ein lokales Minimum bei der Eigenkreisfrequenz von $\omega_{0,R} \approx 0,8$ gefunden wurde, die ziemlich genau mit der der Anlage übereinstimmt, siehe Bild 16a. Es ist auch logisch nachvollziehbar, dass bei größer werdendem $|K_P|$ das Gütekriterium größer wird (= schlechtere Regler-Einstellung). Durch das klein eingestellte D_R ist insbesondere der Regler-D-Anteil falsch eingestellt. Wenn nun der Regler durch größere K_P mehr und mehr „aktiviert“ wird, wird das schlechte Ergebnis sichtbar. Bei kleinen Regler-Verstärkungen ist der Regler so gut wie ausgeschaltet und die mechanische Reibung „hilft mit“, um die Anlage zu dämpfen. Anmerkung: Die Reglerverstärkung ist in den Gütegebirgen negativ, um der Anlagen-Geschwindigkeit entgegen zu wirken, siehe Abschnitt 3.

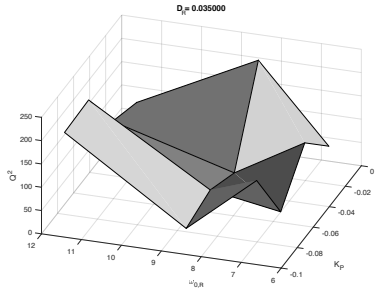


(a) $D_R = 0,002$

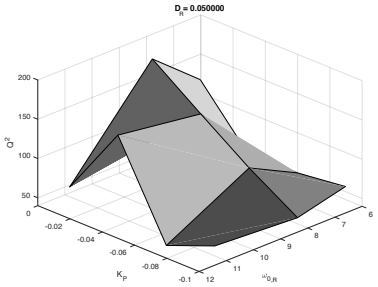


(b) $D_R = 0,005$

Bild 16: Gütegebirge mit zu kleinem Dämpfungsgrad D_R



(a) Dämpfungsgrad $D_R = 0,035$, damit etwas kleiner als der der realen Anlage



(b) Dämpfungsgrad $D_R = 0,05$ passend zur Anlage.

Bild 17: Gütegebirge mit (tendenziell) passendem Dämpfungsgrad D_R

Gute regelungstechnische Ergebnisse konnten erzielt werden, sobald das eingestellte D_R in die Nähe des realen Dämpfungsgrades D der Anlage kam, siehe Bild 17 mit den Werten $Q^2 < 250$. Interessant ist wieder, dass die Eigenkreisfrequenz ω_0 der Anlage ungefähr gefunden wurde, siehe Bild 17a. Ebenso ist es gut zu sehen, dass nun eine Vergrößerung von K_P zu besseren Ergebnissen führt, das Q^2 wird kleiner, siehe Bild 17b.

Bei der automatisierten Parameteroptimierung wurde bei dieser Anlage auch deutlich, dass mehrere und auch randomisierte Versuchsdurchführungen notwendig sind, um gute Regler-Parameter zu finden. Dieses Ergebnis deckt sich auch beim Vergleich mit einer Regler-Inbetriebnahme durch einen erfahrenen Ingenieur. Hierbei sind viele Kriterien bei einer guten Regler-Einstellung zu berücksichtigen, z. B. gute und vor allem robuste Ergebnisse bei vielen unterschiedlichen Störgrößen.

5 Zusammenfassung

In diesem Beitrag wurden Grundlagen der aktiven und passiven Dämpfung von technischen Systemen gezeigt und an Industriebeispielen diskutiert. Zudem wurden verschiedene Möglichkeiten gezeigt, um Regler für eine aktive Dämpfung auszulegen. Abschließend ist die automatisierte Einstellung passender Reglerparameter beschrieben, die im laufenden Betrieb einer Anlage gefunden werden.

Literatur

[1] H. Dresig und A. Fidlin. „Schwingungen mechanischer Antriebssysteme“. Springer Vieweg. 2014.

[2] D. Findeisen und S.Helduser. „Ölhydraulik“. Springer Vieweg. 2015.

[3] O. Föllinger. „Regelungstechnik“. VDE Verlag. 2016.

[4] T. Loose: „Benchmark-Anwendung zur Schwingungsanalyse und -dämpfung von Regalbediengeräten am Beispiel eines Labormodells“. In: *Proc., 25. Workshop Computational Intelligence* (Hoffmann, F.; Hüllermeier, E., Hg.), S. 127-144. Universitätsverlag Karlsruhe. 2015.

[5] T. Loose und T. Pospiech: „Benchmark-Untersuchung zur Regelung schwach gedämpfter Systeme bei industriepraktischen Anwendungen“.

In: *Proc., 27. Workshop Computational Intelligence* (Hoffmann, F.; Hüllermeier, E.; Mikut, R., Hg.), S. 175-194. Universitätsverlag Karlsruhe. 2017.

[6] T. Loose und T. Pospiech. „Regelungstechnische Labormodelle mit industriepraktischen Anwendungen für die Hochschullehre“. In: *at – Automatisierungstechnik* Vol. 67, Nr. 2. S. 157–168. 2019.

[7] H. Lutz, W. Wendt. „Taschenbuch der Regelungstechnik“. Europa-Lehrmittel. 2014.

[8] S. Noack (Vorwort), Bosch Automation. „Hydraulik in mobilen Arbeitsmaschinen“. Christiani. 2001.

[9] W. Oppelt. „Kleines Handbuch technischer Regelvorgänge“. Weinheim / Bergstr.: Verlag Chemie. 1972.

[10] S. Pischinger und U. Seiffert. „Vieweg Handbuch Kraftfahrzeugtechnik“. Springer Vieweg. 2016.

[11] C. Rapp. „Aktive Dämpfung der Lastschwingungen bei Containerkranen“. Dissertation, Technische Universität Hamburg. 2012.

[12] D. Schröder. „Elektrische Antriebe – Regelung von Antriebssystemen“. Springer Vieweg. 2015.

Data Mining im geschlossenen Regelkreis basierend auf adaptiven Kennfeldern mit integriertem Anti-Windup-Mechanismus

Stephan Godt, Martin Kohlhase

Center for Applied Data Science Gütersloh, FH Bielefeld
Schulstraße 10, 33330 Gütersloh
E-Mail: {stephan.godt, martin.kohlhase}@fh-bielefeld.de

1 Einführung

Kennfelder stellen im Bereich der Automatisierung von Produkten und Maschinen eine einfache Lösung dar, nichtlineare Zusammenhänge abzubilden. Sie können einfach bedatet werden und sind vom Applikateur gut zu interpretieren. Kennfelder werden beispielsweise als statische (Vor-) Steuerungen, als Korrekturkennfelder oder als Speichermöglichkeiten für Sollwerte bzw. Parameter eingesetzt, wie in Bild 1 zu sehen ist. Mit dem Einsatz von adaptiven Kennfeldern kann ein zeitvariantes Prozessverhalten berücksichtigt werden, indem Stützstellen online adaptiert werden. Es können nichtlinearen Zusammenhänge online gelernt werden, sodass Einflüsse durch Alterung, Abnutzung und Bauteiltoleranzen im laufenden Betrieb kompensiert werden können.

In diesem Beitrag steht ein Aspekt bei adaptiven Kennfeldern besonders im Fokus. Es wird gezeigt, dass der Einsatz eines adaptiven Kennfeldes im geschlossenen Regelkreis als Data-Mining-Prozess angesehen werden kann. Der Begriff des Data-Minings beinhaltet die Erfassung, Speicherung und Weiterverarbeitung von Daten, sodass weiteres Prozesswissen gewonnen werden kann. Genau dies findet bei der Anwendung von adaptiven Kennfeldern im geschlossenen Regelkreis statt. Prozessinformationen werden verdichtet in einzelnen Stützstellen gespeichert. Durch Adaption der Stützstellen kann neues

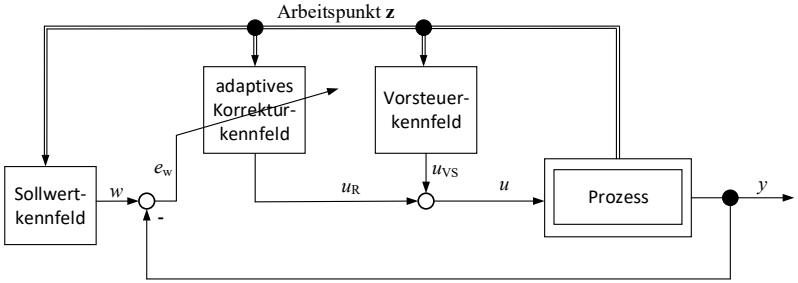
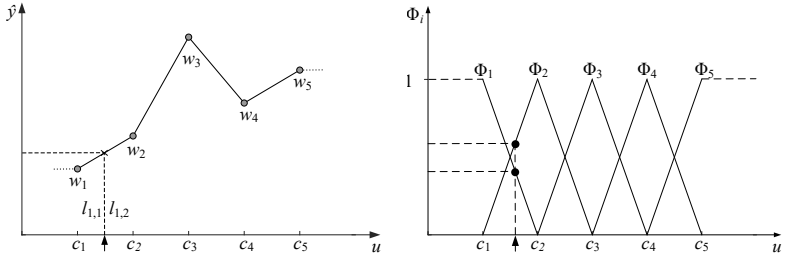


Bild 1: Regelungsstruktur bestehend aus einem Sollwertkennfeld, einem Vorsteuerkennfeld und einem adaptiven Kennfeld, das in Abhängigkeit der Regelabweichung e_w die Vorsteuerung korrigiert.

Prozesswissen generiert werden und durch Interpolation, d.h. der Verknüpfung des in den Stützstellen gespeicherten Wissens, dargestellt werden. Der Speicheraufwand ist dabei sehr gering im Vergleich zu z.B. neuronalen Netzen, da nur die Gitterpositionen und Stützstellenhöhen gespeichert werden müssen. Das Prozesswissen kann zudem extrahiert und für weitere Optimierungsschritte im Gesamtkontext der Automatisierungsaufgabe genutzt werden. Aus den gespeicherten Informationen lassen sich weitere Modelle ableiten. So können mit den gewonnenen Informationen beispielsweise kritische Bereiche im Eingangsraum erkannt und in die Versuchsplanung mit einbezogen werden.
Um mit adaptiven Kennfeldern Data-Mining betreiben zu können, müssen alle Aspekte der Regelungstechnik berücksichtigt werden. So wird eine Anti-Windup-Prävention benötigt, die das unbeabsichtigte „Aufspulen“ („Windup“) verhindert, da das adaptive Kennfeld bei festem Arbeitspunkt wie ein herkömmlicher I-Regler wirkt, der arbeitspunktabhängig den Integratorwert speichert. Die in dieser Arbeit vorgestellte Anti-Windup-Funktionalität gewährleistet, dass die Adaptionsgeschwindigkeit bei Berücksichtigung der Stellgrößenbegrenzung der des herkömmlichen Adaptionsalgorithmus entspricht. Sobald eine der Stützstellenhöhen des Kennfeldes in die Stellgrößenbegrenzung läuft, wird die Adaptionsschrittweite bei den entsprechenden Stützstellen vergrößert, sodass bei gleichbleibender Regelabweichung eine gleichbleibende Integrationsgeschwindigkeit (Integrationskonstante bleibt gleich) sichergestellt ist.



(a) Kennlinie mit Stützstellenpositionen c_i und Stützstellenhöhen w_i

(b) Zugehörige Basisfunktionen Φ_i,zur Gewichtung der Stützstellenhöhen.

Bild 2: Eindimensionales Kennfeld

2 Adaptive Kennfelder

Die ein - bzw. zweidimensionalen Rasterkennfelder dienen als Interpolationsspeicher und bilden einen nichtlinearen Zusammenhang durch Stützstellen ab, welche auf einem orthogonalen Gitter angeordnet sind. Ein Kennfeld lässt sich durch eine nichtlineare Funktion $y = f(\mathbf{u})$ mit p Eingangsgrößen $\mathbf{u} = [u_1, u_2, \ldots, u_p]^T$, durch einen Interpolationsansatz mit M gespeicherten Informationseinheiten, die aus den Positionen c_i und den dazugehörigen Stützstellenhöhen w_i bestehen, darstellen. Die Schätzwerte $\hat{y}$ für den nachzubildenden Zusammenhang werden durch Interpolation mittels der diskreten Positionen und Höhen im p-dimensionalen Eingangsraum repräsentiert [1]. Der Speicheraufwand im ein- bzw. zweidimensionalen Fall ist sehr gering und ergibt sich aus Gleichung (1). Die Anzahl der gesamten Stützstellen

$$M = \prod_{i=1}^{p} M_i \tag{1}$$

ergibt sich aus der Anzahl der Stützstellen M_i pro Eingangsgröße und steigt exponentiell in Abhängigkeit der Eingänge p, sodass auch vom „Fluch der Dimensionen“gesprochen werden kann [2]. Eindimensionale Kennfelder bzw. Kennlinien stellen die einfachste Form eines Kennfeldes dar. Die Informationen werde in Form von Stützstellenpositionen c_i und Stützstellenhöhen w_i

gespeichert, wobei $i = 1,\ldots,M$ und M die Anzahl der Stützstellen sind. Eine solche Kennlinie mit $M = 5$ Stützstellen ist in Bild 2a dargestellt. Der Ausgang

$$\hat{y} = w_i \cdot \frac{l_{i,2}}{\Delta c_i} + w_{i+1} \cdot \frac{l_{i,1}}{\Delta c_i} \tag{2}$$

der Kennlinie berechnet sich für eine Eingangsgröße im Bereich $c_i \leq u \leq c_{i+1}$, mittels linearer Interpolation aus den nächstliegenden linken und rechten Stützstellenhöhen w_i und w_{i+1}. Dabei werden die Stützstellenhöhen mit den gegenüberliegenden Längen

$$l_{i,1} = u - c_i \tag{3}$$

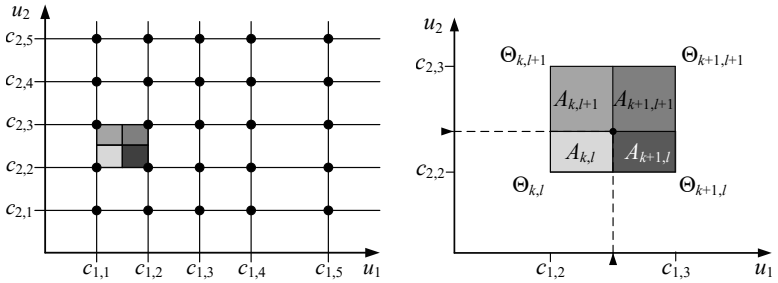
$$l_{i,2} = c_{i+1} - u, \tag{4}$$

gewichtet und auf die Gesamtlänge $\Delta c_i = l_{i,1} + l_{i,2} = c_{i+1} - c_i$ normiert. Fällt der Kennlinieneingang genau auf eine Stützstellenposition, d.h. $u = c_i$, entspricht der Kennlinienausgang $\hat{y}$ der Stützstellenhöhe $\hat{y} = w_i$ [3]. Sollte der Kennlinieneingang außerhalb des Interpolationsgebietes liegen, d.h. $u < c_1$ oder $u > c_M$, so ist der Kennlinienausgang nicht definiert. In der Praxis wird hier der Ausgang gleich der nächstliegenden Stützstelle gesetzt. Für eine allgemeine Interpretierbarkeit können Rasterkennfelder auch durch einen Basisfunktionenansatz beschrieben werden [3, 4, 1]. Der Ausgang der Kennlinie

$$\hat{y} = \sum_{i=1}^{M} (w_i \cdot \Phi_i(\mathbf{u}, \mathbf{c})), \tag{5}$$

berechnet sich aus den Stützstellenhöhen w_i multipliziert mit der jeweiligen Basisfunktion $\Phi_i(\mathbf{u}, \mathbf{c})$ der i-ten Stützstelle, die vom Kennlinieneingang $\mathbf{u}$ und von den Stützstellenpositionen $\mathbf{c} = [c_1, c_2, \ldots, c_M]^T$ abhängt. Im eindimensionalen Bereich ist der in Gleichung (5) und Gleichung (6) als Vektor dargestellte Kennlinieneingang als Skalar zu betrachten. Die Basisfunktionen sind in Bild 2b dargestellt. Das Zentrum einer Basisfunktion Φ_i stellt die jeweilige Stützstellenposition c_i dar. Die Summe aus allen Basisfunktionswerten beträgt für jeden Eingangswert eins.

$$\sum_{i=1}^{M} \Phi_i(\mathbf{u}, \mathbf{c}) = 1 \tag{6}$$



(a) Rasterkennfeld mit entsprechenden Stützstellenpositionen $c_{1,k}$ und $c_{2,l}$.

(b) Aktives Interpolationsgebiet mit den entsprechenden Flächen $A_{k,l}$.

Bild 3: Zweidimensionales Kennfeld

Die Basisfunktionen bei eindimensionalen Kennfeldern lassen sich allgemein definieren durch

$$\Phi_i(u,\mathbf{c}) = \begin{cases} \frac{l_{i-1,1}}{\Delta c_{i-1}} & \text{für} \quad c_{i-1} \leq u \leq c_i \\ \frac{l_{i,2}}{\Delta c_i} & \text{für} \quad c_i < u \leq c_{i+1} \\ 0 & \text{sonst.} \end{cases} \tag{7}$$

Bei einem zweidimensionalen Kennfeld (p=2) befinden sich die Stützstellen auf orthogonalen Gitterlinien, die nicht äquidistant sein müssen. Nach Gleichung (1) ergibt sich somit ein Rasterkennfeld der Größe $M_1 \times M_2$. Das Interpolationsgebiet bei einem Kennfeld dieser Art mit den Eingangsgrößen u_1 und u_2, den Stützstellenpositionen $\mathbf{c} = [c_{1,k}, c_{2,l}]^T$ und den Stützstellenhöhen $\Theta_{k,l}$, wobei $k = 1,2,...,M_1$ und $l = 1,2,...,M_2$ ist, wird in Bild 3 dargestellt. Zur Berechnung des Kennfeldausgangs

$$\begin{aligned} \hat{y} = \Theta_{k,l} \cdot \frac{A_{k+1,l+1}}{a_{k,l}} + \Theta_{k+1,l} \cdot \frac{A_{k,l+1}}{a_{k,l}} + \Theta_{k,l+1} \cdot \frac{A_{k+1,l}}{a_{k,l}} \\ + \Theta_{k+1,l+1} \cdot \frac{A_{k,l}}{a_{k,l}} \end{aligned} \tag{8}$$

im Punkt $\mathbf{u} = [u_1, u_2]^T$ werden nur die benachbarten Stützstellen $\Theta_{k,l}$, $\Theta_{k+1,l}$, $\Theta_{k,l+1}$, $\Theta_{k+,1\,l+1}$ berücksichtigt und mit den Teilflächen $A_{k,l}$, $A_{k+1,l}, A_{k,l+1}$ und $A_{k+1,l+1}$ gewichtet, welche auf die Gesamtfläche $a_{k,l}$ nor-

miert werden [5, 6]. Die Teilflächen ergeben sich aus den Stützstellenpositionen und den Größen u_1 und u_2. Die Gesamtfläche wird aus den einzelnen Teilflächen bestimmt. Die Indizes k und l deklarieren die hinsichtlich der Eingangsgrößen u_1 und u_2 nächst kleineren Stützstellenpositionen. Der berechnete Kennfeldausgang kann wie im eindimensionalen Fall (Gleichung (5)) als Basisfunktionenansatz vereinheitlicht dargestellt werden. Die Basisfunktionen $\Phi_i(\mathbf{u},\mathbf{c})$ entsprechen dabei den jeweilig aktiven normierten Teilflächen. Wird der Kennfeldausgang mittels der vier Stützstellenhöhen und der jeweiligen Basisfunktionen berechnet, ist eine *bilineare* Interpolation notwendig [3].

2.1 Online-Adaption der Kennfeld-Stützstellen

Bei einer Offline-Bedatung von Rasterkennfeldern können die Stützstellenhöhen beispielsweise mit der *Methode der kleinsten Quadrate* in einem Schritt geschätzt werden, wobei eine ausreichend große Datengrundlage zur Verfügung stehen muss. Bei der Online-Bedatung werden Adaptionsalgorithmen wie das *Recursive-Least-Squares* (RLS)-Verfahren und das *Normalized-Least-Mean-Squares* (NLMS)-Verfahren verwendet. Das RLS-Verfahren weist im Vergleich zum NLMS-Verfahren eine höhere Adaptionsgeschwindigkeit auf, besitzt aber einen hohen Speicherbedarf, da mit jedem Adaptionsschritt die Diagonale der Kovarianzmatrix abgespeichert werden muss [10]. Das RLS-Verfahren wird im Folgenden nicht weiterverwendet, da es im geschlossenen Regelkreis nur schwer zu interpretieren ist. Zur Online-Adaption der Stützstellen wird in dieser Arbeit das *Normalized-Least-Mean-Squares* (NLMS)- Verfahren verwendet. Dieses Gradientenverfahren besitzt den Vorteil, dass der Speicherbedarf des adaptiven Kennfeldes gegenüber dem herkömmlichen Rasterkennfeld nicht signifikant ansteigt. Bei dieser Methode wird nicht die Fehlerquadratsumme minimiert, sondern der quadratische Fehler in den einzelnen Abtastschritten als Gütekriterium herangezogen. Der jeweils aktuelle quadratische Fehler wird somit als Schätzwert für den mittleren quadratischen Fehler verwendet. Anhand des Verfahrens werden die neuen Stützstellenhöhen

$$\hat{w}_i(k+1) = \hat{w}_i(k) + \beta \cdot e(k) \cdot \Phi_i(\mathbf{u},\mathbf{c}) \tag{9}$$

in Abhängigkeit der alten Stützstellenhöhen $\hat{w}_i(k)$, einer einzustellenden Lernrate bzw. Schrittweite β, dem momentanen Fehler $e(k) = y(k) - \hat{y}(k)$ sowie anhand der Basisfunktion $\Phi_i(\mathbf{u}, \mathbf{c})$ berechnet. Ein Nachteil dieser Form der Adaption der Stützstellenhöhen ist, dass es zu einer ungleichmäßigen Fehlerreduktion in den einzelnen Adaptionsschritten kommt. Diese Probleme werden in [7, 8] behandelt und durch eine geeignete Normierung behoben, sodass eine gleichbleibende Fehlerreduktion resultiert. Die Stützstellenhöhen berechnen sich mit der Normierung zum jeweiligen Abtastschritt mit der Formel:

$$\hat{w}_i(k+1) = \hat{w}_i(k) + \beta \cdot e(k) \cdot \frac{\Phi_i(\mathbf{u}, \mathbf{c})}{\sum_{j=1}^{M} (\Phi_j^2(\mathbf{u}, \mathbf{c}))}. \tag{10}$$

Für eine gleichbleibende Fehlerreduktion sorgt die Normierung der i-ten Basisfunktion auf die Summe aller quadrierten Basisfunktionen. Der Vorteil einer gleichbleibenden Fehlerreduktion im Hinblick auf eine Regelung wird im folgenden Abschnitt deutlich.

3 Adaptive Kennfelder im geschlossenen Regelkreis

Werden Kennfelder für einen Data-Mining-Prozess im geschlossenen Regelkreis eingesetzt, müssen regelungstechnische Aspekte berücksichtigt werden. Der Fehler $e(k)$ ist somit die Regelabweichung e_w. Der Kennfeldausgang liefert die Stellgröße $u(k)$ bzw. u_R. Der Kennfeldeingang u wird jetzt durch den arbeitspunktbestimmenden Vektor $\mathbf{z} = [z_1, z_2]^T$ dargestellt (siehe Bild 1). Bei einer Anwendung im geschlossenen Regelkreis ist besonders die Interpretierbarkeit der Adaptionseigenschaft wichtig. Hier kommt der Vorteil des NLMS-Verfahrens zu tragen, bei dem die Fehlerreduktion pro Abtastschritt konstant bleibt und somit interpretierbar ist. Dies kann dadurch gezeigt, indem die Gleichung (10) in Gleichung (5) eingesetzt wird. Nach Umformung ergibt sich somit:

$$
\begin{aligned}
u_R(k+1) &= \Theta_{k,l} \cdot \frac{A_{k+1,l+1}}{a_{k,l}} + \Theta_{k+1,l} \cdot \frac{A_{k,l+1}}{a_{k,l}} + \Theta_{k,l+1} \cdot \frac{A_{k+1,l}}{a_{k,l}} \\
&\quad + \Theta_{k+1,l+1} \cdot \frac{A_{k,l}}{a_{k,l}} + \beta \cdot e_w(k) \\
&= u_R(k) + \beta \cdot e_w(k).
\end{aligned} \tag{11}
$$

Das adaptive Kennfeld, basierend auf dem NLMS-Algorithmus, wirkt bei konstantem Arbeitspunkt im geschlossenen Regelkreis wie ein herkömmlicher I-Regler mit $u(k+1) = u(k) + \frac{T_0}{T_1} \cdot e(k)$. Dabei bilden die ersten vier Summanden die Stellgröße $u(k)$. Die Lernrate β ist umgekehrt proportional zur Integrationskonstanten T_1. Der Vorteil bei dieser Regelungsstruktur liegt darin, dass der Integratorwert synchron zum Regelalgorithmus in Form von Stützstellenhöhen in Abhängigkeit des Betriebspunkts $\mathbf{z} = [z_1, z_2]^T$ gespeichert wird.

4 Online-Adaption der Kennfelder mit Windup-Prävention

Werden adaptive Kennfelder als I-Regler im geschlossenen Regelkreis eingesetzt, müssen folgende Anforderungen berücksichtigt werden:

- Es muss eine Anti-Windup-Funktionalität vorhanden sein, um das unbeabsichtigte „Aufspulen“ des Kennfeldes zu verhindern.
- Es muss eine gleiche Fehlerreduktion in allen Arbeitspunkten bei konstanter Regelabweichung gewährleistet werden, auch wenn sich Stützstellen in der Begrenzung befinden.

Ansätze zur Windup-Prävention gibt es für die unterschiedlichsten Reglerstrukturen [9]. Diese sind allerdings nicht ohne Weiteres auf adaptive Kennfelder zu übertragen, da bei diesen die Stellgröße $u(k)$ aus den benachbarten Stützstellen durch einen Interpolationsansatz berechnet wird. Hierbei kann der Fall eintreten, dass eine oder mehrere Stützstellen den zulässigen Bereich bereits verlassen haben, wohingegen die berechnete Stellgröße noch im Stellbereich liegt. Bei einem sprungförmigen Arbeitspunktwechsel ($\mathbf{z}(k-1) \neq \mathbf{z}(k)$) in die Nähe der Stützstellen, die außerhalb des Stellbereichs liegen, wird sich

die neue Stellgröße, die vom Kennfeld ausgegeben wird, auch außerhalb des zulässigen Bereichs befinden. Dies bedeutet, dass sich die Stellgröße in der Begrenzung befindet und sich das adaptive Kennfeld „aufgespult“ hat. Die Problematik wird im Bild 4 am Beispiel eines eindimensionalen Kennfeldes verdeutlicht.

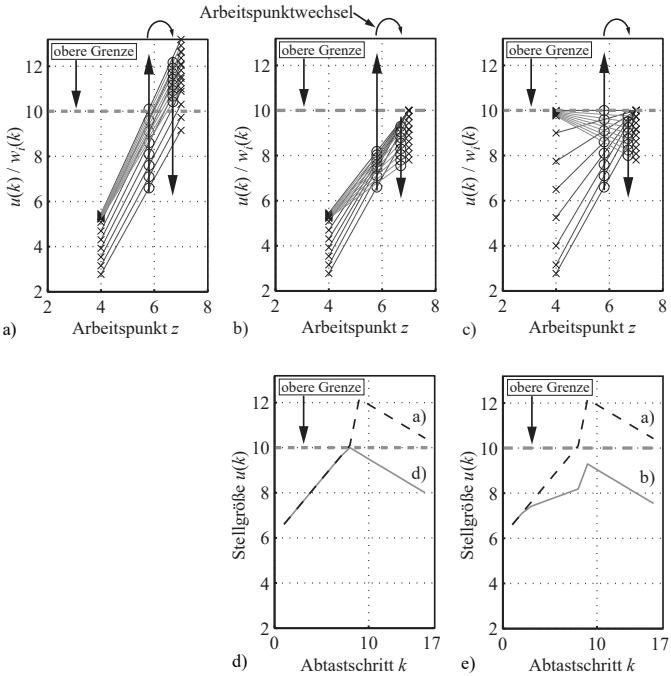


Bild 4: Vergleich der Adaptionsverfahren mit und ohne Windup-Prävention: a) Adaption der Stützstellen ohne Anti-Windup; b) mit Anti-Windup, aber ohne gleichbleibende Fehlerreduktion; c) mit Anti-Windup und gleichbleibender Fehlerreduktion d) Vergleich der, mit den Algorithmen aus a) und c) generierten, Ausgangsgrößen der Kennfelder und e) Vergleich der, mit den Algorithmen aus a) und b) generierten, Ausgangsgrößen der Kennfelder.

Werden keine Maßnahmen zur Windup-Prävention getroffen, so wird die Stellgröße solange in der Begrenzung bleiben, bis sich die Stützstelle, in Abhängigkeit der Regelabweichung e_w, an den gültigen Bereich angepasst hat (siehe Bild 4a). Dies wird durch den Stellgrößenverlauf a) in Bild 4e dargestellt. Eine

schnelle und einfache Lösung, das „Aufspulen“ des adaptiven Kennfeldes zu verhindern, ist die Adaption der sich in der Stellgrößenbegrenzung befindlichen Stützstelle zu stoppen. Diese Lösung wird in Bild 4b dargestellt. Der Nachteil dieser Lösung ist, dass die Regelgeschwindigkeit bei gleichbleibender Regeldifferenz e_w abnimmt, sobald sich eine Stützstelle in der Begrenzung befindet. Dieser Effekt ist im Verlauf b) der Stellgröße $u(k)$ in Bild 4e zu erkennen. Abhilfe schafft die entworfene Anti-Windup-Funktionalität [10]. Diese ist speziell für adaptive Rasterkennfelder ausgelegt, die wie in Bild 1 im geschlossenen Regelkreis betrieben werden. Die Windup-Prävention gewährleistet, dass die Stellgröße sowie die Stützstellen den gültigen Bereich nicht verlassen und die Adaptionsgeschwindigkeit bei Berücksichtigung der Stellgrößenbegrenzung der des herkömmlichen Adaptionsalgorithmus (Gleichung 10) entspricht. In Bild 4c wird der Einfluss der Anti-Windup-Funktionalität gegenüber dem ursprünglichen Algorithmus aus Bild 4a und dem Algorithmus aus Bild 4b abgebildet. Sobald eine der beiden Stützstellen in die Stellgrößenbegrenzung läuft, wird die Adaptionsschrittweite der anderen Stützstelle vergrößert, sodass bei gleichbleibender Regelabweichung die Ausgangsgröße des Kennfeldes, d.h. die Stellgröße, eine gleichbleibende Integrationskonstante aufweist. Dies kann durch einen Vergleich der Verläufe a) und b) der Stellgrößen $u(k)$ aus Bild 4e mit dem Verlauf d) in Bild 4d gesehen werden. Im Folgenden wird der Anti-Windup-Adaptionsalgorithmus am Beispiel eines eindimensionalen Rasterkennfeldes vorgestellt.

4.1 Anti-Windup für adaptive Kennfelder

Bei der Adaption der Stützstellen im geschlossenen Regelkreis wird die Ausgangsgröße $u(k)$ bei eindimensionalen Kennfeldern durch eine lineare Interpolation zwischen zwei Stützstellen berechnet. Die Hebelwirkung aus den Längen l_1 und l_2 zwischen den Stützstellen gewährleistet eine gleichbleibende Adaptionsgeschwindigkeit. Für die Berechnung der Stützstellenhöhen w_i wird der zuvor beschriebene NLMS-Algorithmus (Gleichung (10)) verwendet, wobei in diesem Fall das Argument der Stützstelle $w_i(k|l)$ den k-ten Adaptionsschritt und den l-ten Berechnungsschritt während eines Adaptionsschrittes

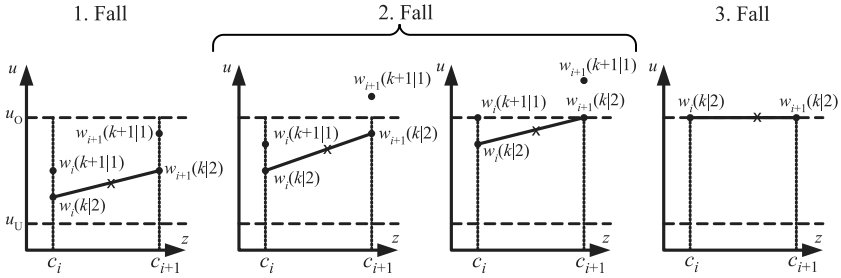


Bild 5: Exemplarische Darstellung unterschiedlicher Adaptionsfälle bei Stellgrößenbegrenzung am Beispiel der Stützstellen w_i und w_{i+1}, wobei w_{i+1} als erste Stützstelle die obere Grenze u_o erreicht und begrenzt werden muss.

darstellt. Es müssen drei Fälle (Bild 5) unterschieden werden, die im Folgenden näher betrachtet werden [10].

1. Beide Stützstellen $w_i(k+1 \mid 1)$ und $w_{i+1}(k+1 \mid 1)$, zwischen denen interpoliert wird, befinden sich nach dem ersten Berechnungsschritt ($l = 1$) innerhalb des gültigen Stellbereichs.

2. Eine der beiden Stützstellen, zwischen denen interpoliert wird, befindet sich nach dem ersten Berechnungsschritt außerhalb (über der oberen Grenze u_o oder unter der unteren Grenze u_u) des gültigen Stellbereichs.

3. Beide Stützstellen $w_i(k+1 \mid 1)$ und $w_{i+1}(k+1 \mid 1)$, zwischen denen interpoliert wird, befinden sich nach dem ersten Berechnungsschritt außerhalb des gültigen Stellbereichs.

Zur Unterscheidung dieser Fälle werden drei Aktivierungsfaktoren definiert:

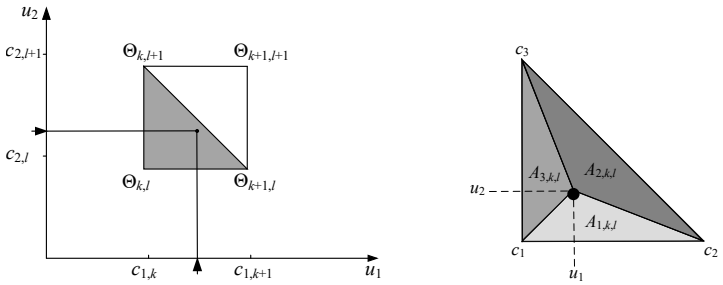
$$
\begin{aligned}
g_{i,1} &= \begin{cases} 1 \;\; \forall \;\; u_u \le w_i(k+1 \mid 1) \le u_o \\ 0 \;\; \text{sonst.} \end{cases} \\
g_{i,2} &= \begin{cases} 1 \;\; \forall \;\; [u_u \le w_i(k+1 \mid 1) \le u_o] \;\wedge\; [w_i(k+1 \mid 1) \ge u_o] \\ 0 \;\; \text{sonst.} \end{cases} \\
g_{i,3} &= \begin{cases} 1 \;\; \forall \;\; w_i(k+1 \mid 1) \ge u_o \\ 0 \;\; \text{sonst.} \end{cases}
\end{aligned}
\tag{12}
$$

Mit diesen Faktoren werden die jeweiligen Summanden für eine gleichbleibende Fehlerreduktion bei Berücksichtigung der Stellgrößenbegrenzung aktiviert oder deaktiviert. Der Adaptionsalgorithmus mit Anti-Windup-Funktionalität zur Adaption der i-ten Stützstelle w_i ergibt sich im eindimensionalen Bereich zu

$$\begin{aligned} w_i(k+1 \mid 2) = w_i(k \mid 2) + \beta \cdot e(k) \cdot \frac{\Phi_i(\mathbf{z},\mathbf{c})}{\sum_{j=1}^{M}(\Phi_j^2(\mathbf{z},\mathbf{c}))} \cdot g_{i,1} \\ -[u_\mathrm{o} - w_{i+1}(k+1 \mid 1)] \cdot \frac{\Phi_{i+1}(\mathbf{z},\mathbf{c})}{\Phi_i(\mathbf{z},\mathbf{c})} \cdot g_{i,2} \\ +[u_\mathrm{o} - w_i(k \mid 2)] \cdot g_{i,3}. \end{aligned} \tag{13}$$

Die ersten beiden Summanden des Adaptionsalgorithmus aus Gleichung (13) repräsentieren dabei den herkömmlichen NLMS-Algorithmus zur Adaption der Stützstellen ohne Stellgrößenbeschränkung (1. Fall in Bild 5). Der dritte Summand stellt den 2. Fall dar. Er addiert die nicht ausgeführte Adaptionsschrittweite der begrenzten Stützstelle w_{i+1} zur Stützstelle w_i hinzu und gewährleistet dabei eine gleichbleibende Fehlerreduktion, indem die nicht ausgeführte Adaptionsschrittweite von w_{i+1} mit dem Verhältnis $\frac{\Phi_{i+1}(\mathbf{z},\mathbf{c})}{\Phi_i(\mathbf{z},\mathbf{c})} = \frac{l_{i,1}}{l_{i,2}}$ gewichtet wird. Der vierte Summand behandelt den 3. Fall und gewährleistet die Stellgrößenbeschränkung für die Stützstelle w_i. Analog zur Stützstelle w_i wird die Stützstelle w_{i+1} berechnet. Ebenfalls werden die Aktivierungsfaktoren $g_{i,1}$ bis $g_{i,3}$ für die untere Grenze u_u genauso wie die für die obere Grenze u_o definiert.

Der Adaptionsalgorithmus mit Anti-Windup-Funktionalität kann vom ein- auf das zweidimensionale Rasterkennfeld übertragen werden. Dafür ist eine lineare Interpolation notwendig. Um dies zu ermöglichen, wird jeweils zwischen drei Stützstellen interpoliert. Hierfür wird die vorgestellte rasterförmige Anordnung (Bild 3a) herangezogen, wobei jeder Quadrant durch eine zuvor festgelegte Diagonale in zwei Dreiecke geteilt wird (Bild 6a). Das Bild 6b zeigt die Anordnung der Stützstellen und die in Abhängigkeit der Eingangsgrößen u_1 und u_2 aktiven Teilflächen. Eine lineare Interpolation besitzt neben der einfachen Übertragbarkeit des Anti-Windups den Vorteil, dass im geregelten Betrieb möglichst wenig Stützstellen adaptiert werden. Ebenso kann durch



(a) Aktives Dreiecksgebiet eines Quadranten.

(b) Aktives Dreieck mit den jeweiligen Teilflächen.

Bild 6: Kennfeldzugriff zwischen drei Stützstellen im zweidimensionalen Fall

die Aufteilung der Quadranten des Rasterkennfeldes in Dreiecksflächen der Anpassungsfreiheitsgrad erhöht werden [10].

Zur Herleitung der Adaptionsvorschrift werden exemplarisch die Fallbeispiele in Bild 7 betrachtet. Die Aktivierungsfaktoren können analog zum eindimensionalen Fall definiert werden. Im ersten Fall aus Bild 7 $w_{i+2} = u_o$ befinden sich die Stützstellen w_i und w_{i+1} im gültigen Bereich und w_{i+2} in der Stellgrößenbegrenzung. Die nicht ausgeführte Adaptionsschrittweite von w_{i+2} wird zu den Stützstellenhöhen w_i und w_{i+1} hinzu addiert. Die zusätzliche Schrittweite wird mit Δw_{i+2} angegeben, wobei im allgemeinen Fall $\Delta w_{i+2}(k+1 \mid 1) = w_{i+2}(k+1 \mid 1) - u_0$ oder $\Delta w_{i+2}(k+1 \mid 1) = u_u - w_{i+2}(k+1 \mid 1)$ gelten kann. Die Stützstellenhöhe

$$\begin{aligned} w_i(k+1 \mid 2) = w_i(k \mid 2) + \beta \cdot e(k) \cdot \frac{\Phi_i(\mathbf{z},\mathbf{c})}{\sum_{j=1}^{M}(\Phi_j^2(\mathbf{z},\mathbf{c}))} \\ + \Delta w_{i+2}(k+1 \mid 1) \cdot \frac{\Phi_{i+1}(\mathbf{z},\mathbf{c})}{\Phi_i(\mathbf{z},\mathbf{c}) + \Phi_{i+1}(\mathbf{z},\mathbf{c})} \end{aligned} \tag{14}$$

wird in diesem Fall aus der alten Stützstellenhöhe $w_i(k \mid 2)$, aus dem gewichteten Fehler $e(k)$ und der nicht realisierten, gewichteten Schrittweite $\Delta w_{i+2}(k+1 \mid 1)$ der begrenzten Stützstelle w_{i+2} berechnet. Mit der Gewichtung $\frac{\Phi_{i+1}(\mathbf{z},\mathbf{c})}{\Phi_i(\mathbf{z},\mathbf{c})+\Phi_{i+1}(\mathbf{z},\mathbf{c})} = \frac{A_{i,i+1}}{A_{i,i+2}+A_{i+1,i+2}}$ wird die geforderte konstante Fehlerreduktion gewährleistet. Dabei errechnet sich hier die Basisfunktion $\Phi_i(\mathbf{z},\mathbf{c})$ einer Stützstelle aus der gegenüberliegenden Fläche bezogen auf die Summe der anderen

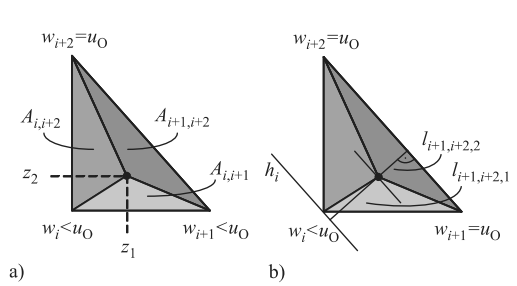


Bild 7: Darstellung der entsprechenden Geometriemaße (Flächen A und Längen l), die für eine Adaption mit Anti-Windup zu berücksichtigen sind. Das Beispiel berücksichtigt die Fälle, in denen eine bzw. zwei der Stützstellen w_i ,w_{i+1} und w_{i+2} die obere Grenze u_o durch Adaption erreicht haben.

Flächen [10]. Analog zur Adaption von w_i wird die andere Stützstellenhöhe w_{i+1} berechnet. Im zweiten Fall (Bild 7b) befindet sich nur die Stützstelle w_i im gültigen Bereich, sodass nur diese zu adaptieren ist und der Adaptionsalgorithmus aus dem eindimensionalen Bereich herangezogen werden kann. Die Stützstelle wird nach der Adaptionsvorschrift

$$\begin{aligned} w_i(k+1 \mid 2) = w_i(k \mid 2) &+ \beta \cdot e(k) \cdot \frac{\frac{l_{i+1,i+2,2}}{l_{i+1,i+2}}}{\left(\frac{l_{i+1,i+2,1}}{l_{i+1,i+2}}\right)^2 + \left(\frac{l_{i+1,i+2,2}}{l_{i+1,i+2}}\right)^2} \\ &+ \beta \cdot e(k) \cdot \frac{\frac{l_{i+1,i+2,1}}{l_{i+1,i+2}}}{\left(\frac{l_{i+1,i+2,1}}{l_{i+1,i+2}}\right)^2 + \left(\frac{l_{i+1,i+2,2}}{l_{i+1,i+2}}\right)^2} \cdot \frac{l_{i+1,i+2,1}}{l_{i+1,i+2,2}} \end{aligned} \tag{15}$$

angepasst. Dabei berechnen sich die Längen $l_{i+1,i+2,1}$ und $l_{i+1,i+2,2}$ aus dem orthogonalen Abstand von Punkt $\mathbf{z}$ zur Geraden, die durch die Stützstellenpositionen w_{i+1} und w_{i+2} verläuft sowie zu dieser parallel verlaufenden und durch die Stützstellenposition w_i führenden Geraden h_i. Die Gesamtlänge $l_{i+1,i+2}$ ergibt sich aus den Teillängen.

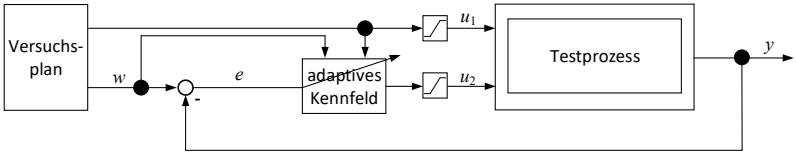


Bild 8: Data-Mining-Prozess unter Verwendung eines Testprozesses.

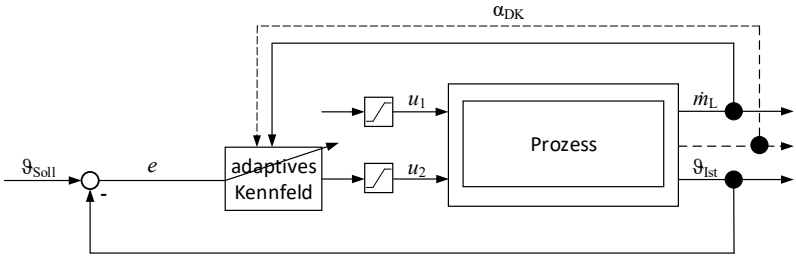
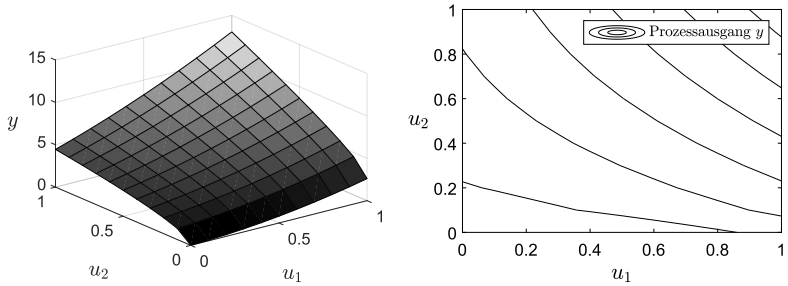


Bild 9: Störgrößenkompensation unter Verwendung einer Heizstrecke

5 Anwendung

Zur Entwicklung und Validierung der vorgestellten Algorithmen werden zwei Aspekte genauer betrachtet. Der im Beitrag fokussierte Data-Mining-Aspekt, basierend auf adaptiven Kennfeldern im geschlossenen Regelkreis, wird unter Verwendung eines Testprozesses untersucht. Die dafür verwendete Regelungsstruktur ist in Bild 8 abgebildet. Die Anregung erfolgt auf Basis eines Versuchsplans im geregelten Betrieb. Es werden zwei Eingangsgrößen u_1 und u_2 definiert. Des Weiteren wird ein gültiger Stellgrößenbereich im Intervall $[0,1]$ vorgegeben. Für die Untersuchungen wird ein zweidimensionales Kennfeld herangezogen, wobei die Führungsgröße w als zweiter Eingang des Kennfeldes dient.

In einem zweiten Schritt wird die Störgrößenkompensation auf Basis adaptiver Kennfelder mit Anti-Windup-Funktionalität, unter Verwendung eines Mehrgrößensystems untersucht. Die dafür verwendete Regelungsstruktur ist in Bild 9 dargestellt. Es werden zwei Eingangsgrößen u_1 und u_2 definiert. Des Weiteren wird ein gültiger Stellgrößenbereich im Intervall $[0,10]$ vorgegeben. Für die Ableitung des Störgrößenmodells wird ein eindimensionales Kennfeld heran-



(a) Darstellung des Prozessausgangs y im dreidimensionalen Raum

(b) Darstellung es Prozessausgangs y in Form von Höhenlinien

Bild 10: Darstellung der Testfunktion mit zwei Eingangsgrößen u_1 und u_2.

gezogen, wobei der gemessene Luftstrom als Kennfeldeingang (Arbeitspunkt) dient.

5.1 Prozessvorstellung

Für Die Untersuchungen der Regelungsstruktur aus Bild 8 wurde ein Testprozess verwendet. Die zugehörige Testfunktion

$$y = 0.6 \cdot u_1 + 2 \cdot u_1^2 + 0.45 \cdot u_2 + 4 \cdot \sqrt{u_2} + 6 \cdot u_1 \cdot u_2 \tag{16}$$

wird für $\{(u_1, u_2) \in \mathbb{R}^2 \mid 0 \leq u_j \leq 1 \ \forall \ j \in \{1,2\}\}$ betrachtet und ist in Bild 10 in Abhängigkeit der Eingangsgrößen u_1 und u_2 abgebildet. In Bild 10a ist der Verlauf der verwendeten Testfunktion im dreidimensionalen Raum dargestellt. Bild 10b stellt diesen Verlauf in Form von Höhenlinien dar. Dabei repräsentieren die dargestellten Höhenlinien den jeweiligen Prozessausgang y in Abhängigkeit des verwendeten Arbeitspunkt. Es ist bei der Wahl des Arbeitspunktes und der Führungsgrößen zu beachten, dass ein möglichst großer Bereich innerhalb der Stellgrenzen von u_1 und u_2 angefahren werden kann.

Für Die Untersuchungen der Regelungsstruktur aus Bild 9 wurde eine eine Heizstrecke der Firma „ELWE Technik“ verwendet (Bild 11). Es handelt sich dabei um einen Labormodell, das über ein xPC Target-System unter Matlab-Simulink angesteuert werden kann. Verbaut sind neben einer Heizung und

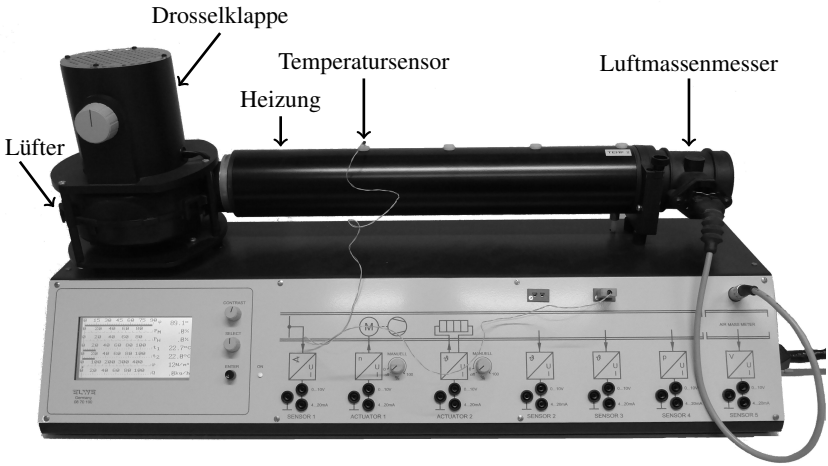


Bild 11: Labormodell der Heizstrecke

einem Lüfter, 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. Folglich kann das System als Mehrgrößensystem mit zwei Eingangs-, zwei Ausgangs- und einer Störgröße angesehen werden. Bei den Eingangsgrößen handelt es sich um die Lüfterleistung P_{L} und die Heizleistung P_{H}. Als Ausgangsgrößen wurden der Luftmassenstrom $\dot{m}_{\mathrm{L}}$ und die Temperatur T erfasst. Letztlich könnte noch der Öffnungswinkel der Drosselklappe α als Störgröße gemessen werden. Dies findet allerdings in dieser Arbeit nicht statt, da der gemessene Luftstrom $\dot{m}_{\mathrm{L}}$ als Kennfeldeingang (Arbeitspunkt) herangezogen wird.

5.2 Durchführung

Die Untersuchungen am Testprozess werden anhand einer Simulationsfallstudie durchgeführt. Die für die Fallstudie gewählte Schrittweite wird fest vorgegeben und beträgt $\beta = 0.01$, sodass eine eher langsame Adaption der Stützstellen mit einer großen Rauschunterdrückung erfolgt. Für die Untersuchungen

wird eine Anzahl von 15 Stützstellen verwendet, die im festgelegten Eingangsbereich u_1 zwischen 0.2 und 0.8 das Prozessverhalten widerspiegeln sollen. Bei der Wahl der Stützstellenanzahl wurde darauf geachtet, mit einer möglichst geringen Anzahl an Stützstellen das Prozessverhalten bestmöglich wiederzugeben. Weitere Parameter, wie der jeweilige Arbeitspunkt der Eingangsgrößen u_1 und u_2, die beide ins Kennfeld einfließen, werden stufenweise im Intervall [0,1] innerhalb der vorgegebenen Stellgrößenbegrenzungen variiert.

Die Untersuchungen an der Heizstrecke finden im eindimensionalen Bereich statt, mit der Lüfterleistung P_L als Eingangsgröße u_1 und dem gemessenen Luftmassenstrom $\dot{m}_\mathrm{L}$ als Störgröße, die ins Kennfeld eingeht. Bei den Untersuchungen wird eine Anzahl von 5 Stützstellen verwendet. Die Temperatur als Führungsgröße wird bei 40°C konstant gehalten. Die Anregung des Systems erfolgt dabei stufenweise, indem die Lüfterleistung u_1 wiederholt zwischen zwei Arbeitspunkten in den Grenzen von 0 bis 10 variiert wird. Aufgrund des dynamischen Verhaltens der Heizstrecke wird das Kennfeld mit einer Schrittweite von $\beta = 0.008$ recht langsam ausgelegt. Der zum Vergleich herangezogene I-Regler dagegen deutlich schneller. Dies erfolgt, um zu zeigen, dass das adaptive Kennfeld mit Anti-Windup-Funktionalität trotz dieses Nachteils ein besseres Störverhalten erzielen kann als der I-Regler.

5.3 Ergebnisauswertung

In Bild 12 werden die Positionen und Höhen der adaptierten Stützstellen im zuvor präsentierten Testprozess dargestellt. Es ist deutlich zu erkennen, dass sich die Stützstellen für alle Kombinationen von u_1 und u_2 im festgelegten Bereich entsprechend der Höhenlinien im Testprozess adaptieren. Da die Höhenlinien den geregelten Prozessausgang y widerspiegeln, kann festgehalten werden, dass sich die Stützstellen mit jedem Adaptionsschritt immer stärker dem Prozessverhalten annähern, bis sie dieses im vorgegebenen Bereich vollständig durch Interpolation beschreiben können. Aufgrund der in jedem Adaptionsschritt durchgeführten Anpassung der in den Stützstellen gespeicherten Informationen wird das hinterlegte Modell immer weiter verbessert, bis das tatsächliche Prozessverhalten bestmöglich wiedergegeben wird.

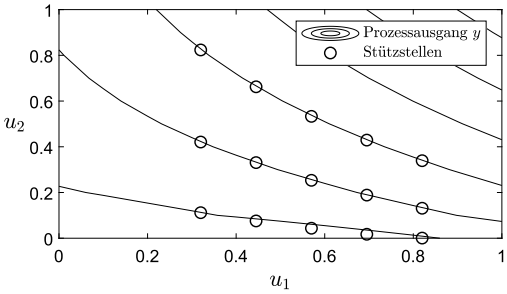


Bild 12: Darstellung der Regelgröße y und der Stützstellen in Abhängigkeit des gewählten Arbeitspunkts.

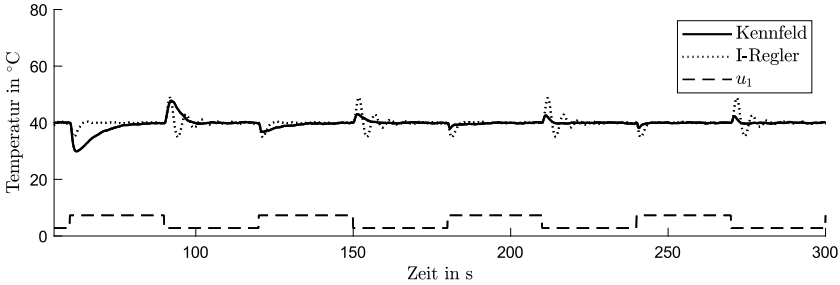


Bild 13: Vergleich des Störgrößenverlaufs beim adaptiven Kennfeld und beim klassischen I-Regler bei unterschiedlichen Arbeitspunkten.

Die Störgrößenkompensation basierend auf einem adaptiven Kennfeld mit Anti-Windup-Funktionalität wird in Bild 13 abgebildet. Dargestellt sind die Verläufe der am Prozessausgang gemessenen Temperatur über die Zeit, wobei der Einsatz eines adaptiven Kennfeldes und der Einsatz eines klassischen I-Reglers miteinander verglichen werden. Zudem wird die konstante Variation in der Lüfterleistung u_1 dargestellt. Es ist deutlich zu erkennen, dass bei einer Erhöhung oder Verkleinerung der Lüfterleistung das adaptive Kennfeld mit zunehmender Zeit ein immer besseres Einschwingverhalten aufweist. Dies ist dadurch zu erklären, dass sich die Stützstellen durch zunehmende Adaption immer besser an den Prozessverlauf anpassen, da die entsprechenden Stellgrößenwerte im Kennfeld gespeichert werden und diese innerhalb der Stellgrößenbegrenzungen liegen. Der I-Regler dagegen weist bei jeder Veränderung der Lüfterleistung ein über die Zeit konstantes Einschwingverhalten auf. Schon

nach 150 Sekunden wird der Vorteil des Kennfeldes gegenüber dem I-Regler deutlich. Mit zunehmender Prozessdauer und einhergehender Anpassung der Stützstellen wird diese Entwicklung immer deutlicher, bis die Stützstellen bestmöglich an den Prozess angepasst sind.

6 Zusammenfassung und Ausblick

In diesem Beitrag wird gezeigt, dass der Einsatz eines adaptiven Kennfeldes im geschlossenen Regelkreis als Data-Mining-Prozess angesehen werden kann. Komplexes Prozessverhalten wird verdichtet in einzelnen Stützstellen gespeichert und durch Interpolation wiedergegeben. Durch Adaption der Stützstellen wird neues Prozesswissen generiert, was die Regelgüte verbessert. Dafür wird eine für adaptive Kennfelder entworfene Anti-Windup-Funktionalität vorgestellt, die gewährleistet, dass die Adaptionsgeschwindigkeit bei Berücksichtigung von Stellgrößenbegrenzungen dem des herkömmlichen Adaptionsalgorithmus entspricht.

Die in den Stützstellen gewonnenen Informationen können nicht nur zur Verbesserung der Regelgüte herangezogen werden. Darüber hinaus lassen sich in einem weiteren Schritt aus den gespeicherten Informationen zur weiteren Prozessverbesserung neue Modelle ableiten. So können mit den gewonnenen Informationen beispielsweise kritische Bereiche im Eingangsraum erkannt und in die Versuchsplanung mit einbezogen werden, sodass mögliche Beschädigungen an Maschinen vermieden werden. Dies kann in einem nächsten Schritt beispielsweise an der beschriebenen Heizstrecke untersucht werden.

Förderung

Dieses Vorhaben wurde aus Mitteln des Europäischen Fonds für regionale Entwicklung (EFRE) gefördert (Förderkennzeichen 34.EFRE-0300119).

Literatur

[1] T. Ullrich. „Untersuchungen zur effizienten interpolierenden Speicherung von nichtlinearen Prozeßmodellen und Vorsteuerstrategien: Methodik und Anwendungen in der Automobilelektronik“. Dissertation, Technische Universität Darmstadt, Aachen: Shaker-Verlag. 1999.

[2] R.E. Bellmann. „Adaptive Control Processes: A Guided Tour“. Princeton University Press. 1961.

[3] A. Fink. „Nonlinear Control Based on Local Linear Neuro-Fuzzy Models“. In: *Fortschritt-Bericht VDI, Reihe 8*, 1096. Düsseldorf: VDI-Verlag. 2006.

[4] N. Müller. „Adaptive Motorregelung beim Ottomotor unter Verwendung von Brennraumdruck-Sensoren“. In: *Fortschritt-Bericht VDI, Reihe 12*, 545. Düsseldorf: VDI-Verlag. 2003.

[5] J. Hoschek. „Grundlagen der geometrischen Datenverarbeitung (German Edition)“. B.G. Teubner. 1989.

[6] M. Schmitt. „Untersuchungen zur Realisierung mehrdimensionaler lernender Kennfelder in Großserien-Steuergeräten“. In: *Fortschritt-Bericht VDI, Reihe 12*, 246. Düsseldorf: VDI-Verlag. 1995.

[7] M. Brown und C. Harris. „Neurofuzzy Adaptive Modelling and Control“. New York: Prentice Hall. 1994.

[8] M. Heiss, D. Heiss und S. Kampl. „Lernen linear interpolierter Kennlinien“. In: *at - Automatisierungstechnik* 11. S. 497–506. 1994.

[9] P. Hippe. „Windup in Control: Its Effects and Their Prevention (Advances in Industrial Control)“. Springer Nature Switzerland. 2006.

[10] M. Kohlhase. „Brennraumdruckbasiertes Motormanagement für Otto- und Dieselmotoren zur Verbrauchs- und Emissionsreduktion“. Dissertation, In: *Fortschritt-Bericht VDI, Reihe 12*, 743. Düsseldorf: VDI-Verlag. 2011.

Data-driven charging strategies for grid-beneficial, customer-oriented and battery-preserving electric mobility

Karl Schwenk[1,2], Tim Harr[2], René Großmann[2],
Riccardo Remo Appino[1], Veit Hagenmeyer[1], Ralf Mikut[1]

[1] Institute for Automation and Applied Informatics,
Karlsruhe Institute of Technology
Karlsruhe, Germany
E-Mail: karl.schwenk@kit.edu

[2] eDrive Innovations, Daimler AG
Sindelfingen, Germany
E-Mail: karl.schwenk@daimler.com

1 Introduction

Electric Vehicle (EV) penetration, renewable energies, and customer orientation of car manufacturers [1] enables synergies between energy supply, vehicle users, and the mobility sector. However, also new challenges arise [2] which we target to examine with three types of agents and their perspectives: EV user, power supplier and car manufacturer. Although many research papers in the literature describe single perspectives, a threefold contemplation of their connections as shown in Fig. 1, is still missing. In the following, we briefly present the single perspectives and respective interactions.

To satisfy their individual needs, **vehicle users** expect their EVs to be just as reliable and convenient to use, as known from combustion engine cars [3, 4]. However, limited range and longer charging times of EVs complicate individual mobility, manifesting e.g. in *range anxiety* [2, 5]. For EV users also the power supply interaction changes, as not only home appliances but

also mobility requires electric energy. Hence, a convenient and cost-efficient charging process is hard to achieve without automation technology, when using an EV on a daily basis [6].

Power suppliers (distribution grid operators, energy retailers) target an efficient and fail-proof grid operation, for which a reliable forecast and control of electric loads is desired [7, 8]. However, the intermittent nature of renewable energy production endangers the utility grid through voltage and frequency fluctuation [9, 10]. The energy demand caused by charging EVs might further amplify this effect [11]. To avoid these issues, power suppliers may influence the EV users' affection to connect their vehicles via dynamic energy prices [9, 10, 12]. At the same time, multiple EVs connected to the grid—especially if bi-directional charging is enabled [13]—help power suppliers to level out imbalances due to renewable energy generation [14].

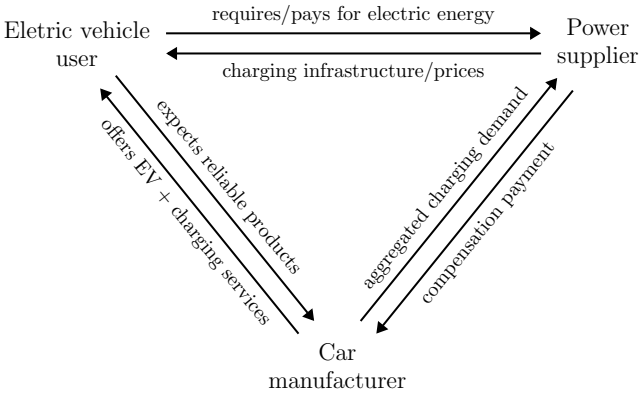


Figure 1: Interaction triangle of electric vehicle user, power supplier, and car manufacturer.

Electric mobility also poses new issues for **car manufacturers**: During charging and discharging of EV batteries a degradation (*battery aging*) occurs [15], that correlates with a value depreciation of the entire EV. However, EV users' satisfaction requires reliable and value-stable products, which car manufacturers aim to achieve by offering services, such as charging assistants [6]. The provided charging strategies target simplified and sustainable EV usage by considering individual customer needs and battery aging.

The remainder of this paper is structured as follows: To identify and develop missing models of the outlined problem, a general approach is presented in Section 4. Then, Section 3 describes an online learning framework for data acquisition, storage and application. In Section 4, we propose two data-driven consumption models. Finally, Section 5 gives a brief summary and outlook on future work.

2 Approach

Despite existing model predictive approaches [16, 17] that support EV users while charging, individualized charging strategies are in general still inadequately dealt with in the literature. We target to identify missing models that quantify perspectives and interactions, for which Fig. 2 shows a schematic map (the green boxes represent unexplored models requiring further evaluation).

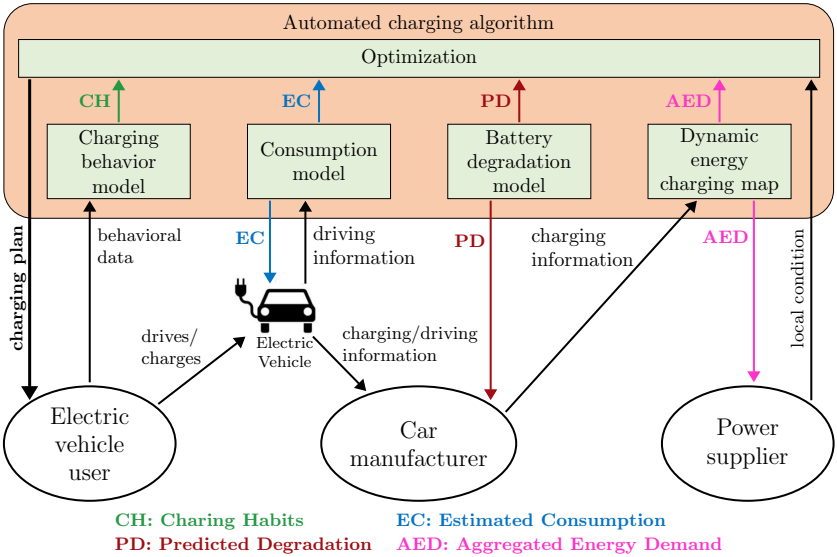


Figure 2: Schematic map of models to describe perspectives and interactions, Automated charging algorithm (orange box) comprises missing models (green boxes)

To automatically create charging plans that satisfy the interests of power suppliers, EV users, and car manufacturers at the same time (cf. Section 1), an **Automated charging algorithm** will be required (Fig. 2, orange box). A **Charging behavior model** (cf. Section 2.1) could determine Charging Habits (CH, Fig. 2, green arrow) based on behavioral data of **Electric vehicle users**. Thereby, charging plans provided to the EV user can be adapted to individual requirements. In order to obtain user- and route-specific Estimated Consumption (EC, Fig. 2, blue arrows) for the EV, a data-driven **Consumption model** (cf. Section 4) may use driving information of the EV. The EC can also help to estimate the remaining driving range of the EV. Further, charging plans should be adapted to be battery preservative by means of a **Battery degradation model** (cf. Section 2.2) that provides a Predicted Degradation (PD, Fig. 2, red arrows) for specific charging strategies. Using the PD, car manufacturers may also monitor the condition of operating EV batteries. A **Dynamic charging energy map** (cf. Section 2.3) could calculate the Aggregated Energy Demand (AED, Fig. 2, pink arrows) based on charging information of several EVs. The **Power supplier** may then make use of the AED for improved load forecast precision. Together with information about momentary grid load conditions obtained from Power suppliers, peak shaving and load balancing applications can be included in the charging plan. An **Optimization** approach may finally conclude all the information and calculate a charging plan for the EV user.

2.1 Charging behavior model

The effectiveness of charging strategies depends on user acceptance, which in turn requires user-specific charging strategies [6]. Therefore, a suitable characterization of users' driving and charging behavior is necessary [18]. Existing approaches in the literature distinguish between qualitative and quantitative ones. Psychological approaches, e.g. deducted from customer surveys [19, 2, 20] allow to characterize basic user types. Therein, *gamification* and *incentivation* [2] methods are developed to increase user acceptance. Quantitative approaches aim at predicting driving and charging behavior of EV users, e.g. to infer energy demand [21, 22, 23, 24]. However, an adequate user

integration, i.e. individualized charging strategies based on EV users' charging habits is not represented.

To attain a more universal characterization of the EV users' charging behavior, different user types need to be determined by analyzing behavioral data of the EV user. Thereby, different user clusters can be identified, for which individual charging habits can be determined (cf. Fig. 2, left). To clarify the EV users' objectives, we plan a customer survey in future work. Therein, the basic requirements towards automated charging strategies are inquired from i) customers using a conventional car, ii) customers about to acquire an EV, iii) customers just recently started using an EV, and iv) customers already using an EV for a longer time. Specific questions on the users' mobility requirements, charging habits, doubts, and expectations are supposed to reveal further requirements for individualized charging strategies. Subsequently, a data-driven analysis of charging processes and EV user information may support these findings and may allow creating a mathematical model.

2.2 Battery degradation model

The interest of car manufacturers focuses on the depreciation process of the EV and its key factors. As the battery considerably contributes to the vehicle value [25], we particularly consider battery degradation. Present-day mobility concepts (e.g. car sharing, leasing) comprise the vehicle battery to remain property of the manufacturer [26]. Value-stable batteries are thus even more relevant to operate economically. To quantify battery aging, usually the State of Health[1] (SOH) is used [27, 28]. Hitherto SOH models have limited practicability due to a complex execution [29] or inaccurate State of Charge (SOC) estimations [30]. Data-driven methods, e.g. Bayesian networks and neural networks [31, 32] hold feasible alternatives for SOH estimation [33]. However, the existing approaches barely use user-related data, for which reason the influence of charging behavior on battery aging is not represented properly.

Hence, we target to evaluate the influence of charging strategies on battery degradation (cf. Fig. 2, center right). By means of a neural network re-

[1] Ratio of actual usable capacity in relation to the nominal battery capacity

gression model, we target to estimate the energy consumption for driving (cf. Section 4), not taking any battery aging influences into account. The model trained on data from batteries without degradation can be used to estimate the energy consumption for EVs with aged batteries. A discrepancy between the estimation and the real consumption indicates a battery aging caused by increased internal losses. Together with a feature-augmented map, this model can additionally estimate energy consumption for charging scheduling, that also considers individual driving styles and environmental conditions. Once this battery aging estimation is mature, also a dependency on charging influences could be inferred. A concept validation has to be proceeded with both simulation and real data.

2.3 Dynamic charging energy map

To match energy usage and generation, power suppliers require information on the energy demand at a certain time and place, targeting a fault-free delivery of electric energy. This includes the demand due to charging EVs. Research papers to model the energy demand caused by charging EVs already exist. They differentiate among approaches to predict and control the grid condition [11, 34, 35, 14], and approaches for dimensioning and locating new charging stations [36, 37]. Stochastic assumptions to describe the EV user behavior are a major deficiency of these approaches.

By combining information about operating EVs, the behavior of their drivers, and the associated charging requirements, we target to aggregate the energy demand caused by charging vehicles and predict vehicles' anticipated charging location.[2] EVs requiring charging can be consolidated to larger energy demands characterized by a load profile over time and location. Then, an "energy map" can be created, containing the aggregated energy demand in real time, or a prediction for future points in time (cf. Fig. 2, right). The required energy can either be acquired directly from the energy market, or the information can be passed to power suppliers to handle the predicted load accordingly. For concept validation, a simulation environment has to be developed.

[2]The access to EV information subjects to the present data security legislation [38].

3 Data framework

In the following, we propose a cloud-based framework to acquire, store and analyze vehicle data, as shown in Fig. 3. A telematic system records selected Controller Area Network (CAN) signals for a fleet of connected EVs during driving and charging.

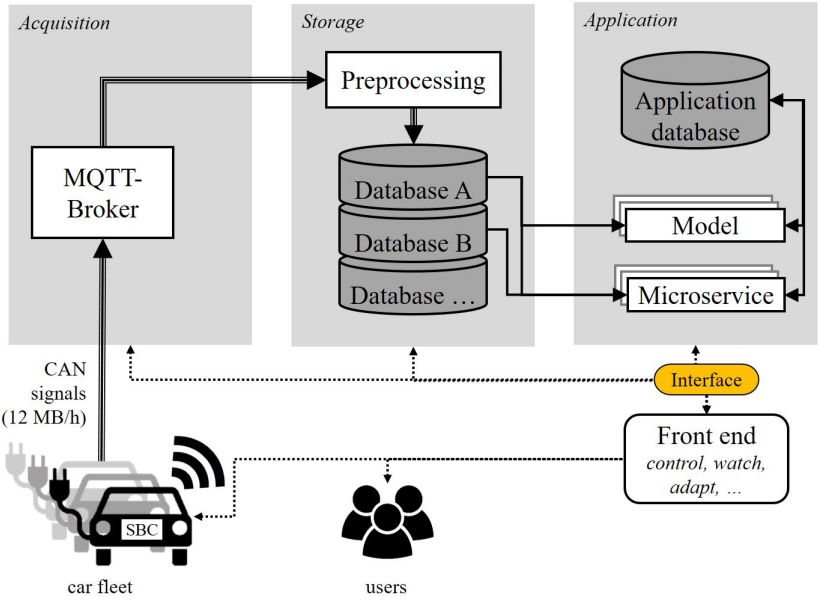


Figure 3: Scheme of framework for data acquisition via a MQTT-broker, storage in several databases and application of data-driven models or microservices.

Applications, e.g. distributed functions structured in microservices require a reliable internet connection to provide historical and live data. Existing devices such as data loggers store data in the car locally. However, data can only be transmitted infrequently via infrastructure-bound connections (e.g. WLAN or LAN). A real-time vehicle status can thus not be established and the large amount of data causes the data transmission to be time-intensive. For this reason, we use a single-board computer (SBC) equipped with a GPS antenna and mobile internet connection as a gateway between vehicle and back-end infrastructure. The SBC initially decodes and filters data with libraries stored

on its internal memory. Via software packages, the SBC can be updated and controlled remotely not requiring a physical contact to the SBC. The data transmission utilizes a Message Queuing Telemetry Transport (MQTT) protocol following a publish and subscribe mechanism[3]. A central MQTT broker (Fig. 3, left) organizes the data distribution. This mechanism also enables communication into the vehicle. Currently ten EVs provide 370 signals of interest, which we record with a 10-Hz sampling rate[4].

We process and store all collected data according to its structure (Fig. 3, middle). A relational database stores vehicle data and meta-information. Further, a document-oriented database allows to store semi-structured signal time series data efficiently. Additionally, a library database allows transforming vehicle specific data into a standardized format.

The stored data can subsequently be used to build models or provide standalone microservices with information (Fig. 3, right). Each model or microservice can make use of several data sources, or communicate among each other, respectively. A separate application database contains configurations and meta-information necessary for the microservice operation. The cloud structure allows continuity in various aspects. Models can be easily implemented, changed, monitored and deleted. Software changes can be applied and deployed continuously. Through a constant availability, the models can be fed with new data each time it is produced. This allows a continuous development of the model with a updated data stream. However, a continuous availability also constitutes a risk in terms of information security [39], that needs to be handled accordingly. The modular structure allows to create and train several models in parallel. An Application Database contains all models and their internal, learned parameters (Fig. 3, right). Via a human machine interface, the models and their learning progress can be monitored and analyzed. This allows to supervise the data acquisition, storage, and application. Furthermore, databases can be examined, microservices and models can be adapted, and errors can be handled accordingly. The interface also provides feedback to the vehicle fleet and vehicle users.

[3] participants can publish information others can subscribe to
[4] equals data stream of approx. 12 Megabyte per hour and vehicle.

Table 1: Input signals grouped by driving behavior, battery state and environmental condition

Driving behavior	Battery state	Environmental cond.
acceleration torque	total electric current	geodetic altitude
brake torque	battery voltage	ambient temperature
recuperation torque	battery temperature	
lateral acceleration		
vehicle speed		
electric current driving		
longitudinal acceleration		

4 Data-driven consumption model

Here, we propose a data-driven consumption model as an exemplary application of the framework presented in Section 3. Consumption models, as used in charging scheduling, help to estimate energy demands for routes the user will drive [40]. However, existing estimators mostly utilize physical models neglecting influences of the driver and the environment. Such consumption models yield relative estimation errors between 2.52 % and 8.3 % [41, 42]. A higher model accuracy allows to compute a more reliable and thoroughly adapted charging strategy. Thus, we aim to design a model estimating the total consumed energy per driven distance for a given driving behavior, environmental condition, and battery state. Note, that we use the dimensionless State of Charge (SOC) as a metric for the consumed energy.

4.1 Data selection

To abstract driving and environmental influences we use time series data recorded during driving of the EVs. We differentiate the input signals in the three categories: driving behavior, battery state, and environmental condition (as shown in Table 1). The *acceleration torque*, *brake torque*, and *recuperation torque* are measured for front and rear axle, i.e. two signals for those values exist. Thus, we obtain a total signal number of $N_{\text{signal}} = 15$.

With a variety of driving situations and environmental conditions as input, we target a universal estimator that generalizes well on arbitrary input data. The samples are sections with duration $t_{\text{sec}} = 6$ min of EV trips representing a typical ride. All trips with insufficient duration or faulty signal values are discarded beforehand.[5] To avoid estimation errors due to random noise, the originally 10-Hz-sampled signals are aggregated before passing them to the model. For each signal and for all values within an aggregation time period $t_{\text{agg}} = 1$ min the mean is calculated.

With 15 signals multiplied by 6 data points per trip section we obtain the extracted features $x_{1,..,90}$. Note that we chose the parameters t_{sec} and t_{agg} according to preliminary test results.[6]

For each data sample representing a trip section of $t_{\text{sec}} = 6$ min we calculate a label

$$\Gamma = \frac{\overline{e} - \underline{e}}{\overline{o} - \underline{o}}. \tag{1}$$

Therein, $\underline{e}$ is the SOC at the start of the trip section, $\overline{e}$ the SOC at the end. Similarly, $\underline{o}$ is the mileage (in kilometers) at the start of the trip section, $\overline{o}$ at the end, respectively.

4.2 Models

To model the dependency between the input features $x_{1,..,90}$ and output label Γ, we design two models.

4.2.1 Model A

For the first Model A we use a linear regression model

$$\hat{\Gamma} = \mathbf{W} \cdot \mathbf{x} + b, \tag{2}$$

[5] In following evaluations, these special cases need to be contemplated separately.
[6] In future work, a more detailed evaluation on the selection of these parameters is required.

with the input features $\mathbf{x} = (x_1, x_2, \ldots, x_{90})^\top$ of the aggregated signal time series, the weight matrix $\mathbf{W} \in \mathbb{R}^{1\times 90}$, and the bias $b \in \mathbb{R}$. The weights $\mathbf{W}$ and the bias b are chosen to yield a minimum mean squared error (cf. Section 4.3) between the training samples and the regression [43].

4.2.2 Model B

For the second Model B we design a neural network regression model. The input layer comprise 90 nodes representing the input features $x_{1,\ldots,90}$. Further, we use five hidden layers in a triangular shape, as shown in Fig. 4. Each of the hidden layer nodes is activated through a Rectified Linear Unit (ReLU). The output layer of the neural network consists of one node representing the estimated consumption $\hat{\Gamma}$.

For training the model we process all trips into training samples (cf. Section 4.1). Then, the data is fed to the models in batches of 32 samples over 100 epochs. The model is implemented in *Python* [44] using *Keras* [45]. We use the *Adam* optimizer [10] with a learning rate $\alpha = 0.001$.

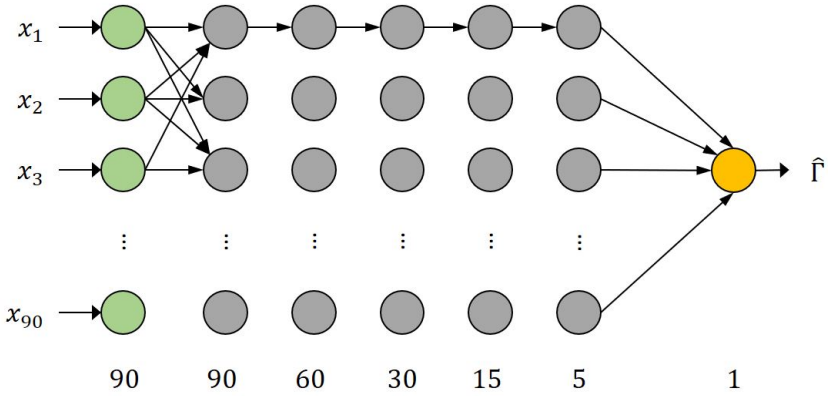


Figure 4: Schematic architecture Model B, green: input nodes, hidden layer nodes in grey, output node in orange, total number of nodes below each layer.

4.3 Evaluation

The models are trained with data obtained from eight out of ten EVs. For validation, the data obtained from the remaining two EVs is used, i.e. we compare the estimated consumption $\hat{\Gamma}$ with the actual consumption Γ. To quantify the model performance, we calculate the metrics
Mean Absolute Error (MAE)

$$\mathrm{MAE} = \frac{1}{N}\sum_{n=1}^{N} |\Gamma_n - \hat{\Gamma}_n|, \tag{3}$$

Relative Mean Absolute Error (RMAE)

$$\mathrm{RMAE} = \frac{1}{N}\sum_{n=1}^{N} |\frac{\Gamma_n - \hat{\Gamma}_n}{\Gamma_n}|, \tag{4}$$

Mean Squared Error (MSE)

$$\mathrm{MSE} = \frac{1}{N}\sum_{n=1}^{N} (\Gamma_n - \hat{\Gamma}_n)^2, \tag{5}$$

and Root Mean Squared Error (RMSE)

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} (\Gamma_n - \hat{\Gamma}_n)^2}. \tag{6}$$

Table 2 reports the validation results according to the error metrics (3)-(16). The results show, that the neural network Model B outperforms the linear Model A in all four metrics. Using Model B, the estimation yields a mean absolute error of 0.00461 km^{-1}, i.e. for a trip of 1 km driven distance, the battery level change is estimated with an average discrepancy of 0.461% SOC compared to the true value. On the contrary, Model A estimates the battery level change for such a trip with an average discrepancy of 52.783% SOC. Note that an SOC of 100% represents a fully charged battery and an SOC of 0% an empty battery, respectively. Considering this fact, the estimations obtained from Model A do not provide useful information on the EV's consumption. Showing the

Table 2: Evaluation of Model A and Model B with MAE, RMAE, MSE, and RMSE.

Model	MAE [km^{-1}]	RMAE [-]	MSE [km^{-2}]	RMSE [km^{-1}]
Model A	0.52783	1.45031	0.44548	0.66744
Model B	0.00461	0.01782	0.00004	0.00651

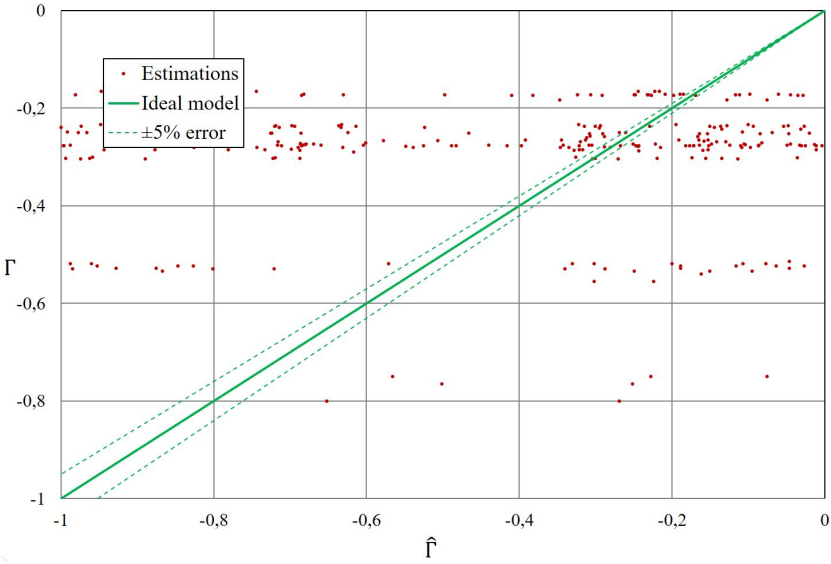


Figure 5: Estimated consumption $\hat{\Gamma}$ (red dots) of Model A with validation data set compared with the true consumption Γ, ideal model output (green line), and 5% relative estimation error (green dashed lines).

results graphically further emphasizes this finding: Figure 5 illustrates the validation results of Model A. For each validation sample with consumption Γ the estimation $\hat{\Gamma}$ that the model provided is depicted as red dot. The green line represents an ideal model behavior for comparison. The green dashed lines represent an RMAE of 5%. It can be seen, that most of the validation samples are not estimated correctly by the linear Model A, shown by the red dots that are far from the green line. Based on the low accuracy of the linear model, we

assume the real relation between the input features $x_{1,..,90}$ and the consumption Γ to be non-linear.

The results of the neural network Model B support this assumption: In the same manner as done for Model A, Fig. 6 shows the estimations for Model B. The results show a higher precision, as the estimations $\hat{\Gamma}$ are located closer to the ideal model representation, i.e. the green line. A mean relative estimation error $\varepsilon_B = 1.78$ % (cf. Table 2) on the validation data supports this result. According to the results, the internal structure of the neural network Model B seems to represent the influences on the energy consumption with much higher precision than the linear Model A.

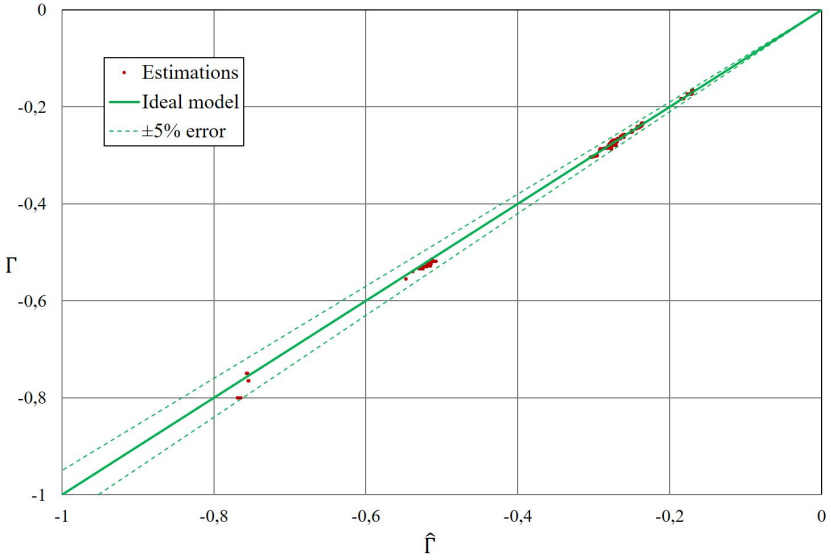


Figure 6: Estimated consumption $\hat{\Gamma}$ (red dots) of Model B with validation data set compared with the true consumption Γ, ideal model output (green line), and 5% relative estimation error (green dashed lines).

A further improvement of the model performance is expected if further data, e.g. the actual weight of the electric vehicle due to baggage and passengers, would be considered. Cross-validation over several vehicles should be proceeded to test the generalization of the developed model.

Note that the input data excludes signals allowing to infer the state of battery degradation. Thus, the consumption model does not consider effects of battery aging. Prospectively, battery aging can be estimated by training a similar model with data solely obtained from EVs, whose batteries have not degraded yet. Then, a consumption can be estimated for vehicles that are assumed to have an aged battery. A discrepancy between the estimation and the real consumption—beyond the known estimation error—indicates a battery aging due to increased internal losses. This method would allow an entirely data-driven battery aging estimation based on data anyway created within the vehicle operation.

5 Conclusion and perspective

In this paper, we examined perspectives and interactions of Electric Vehicle (EV) users, car manufacturers and power suppliers. We proposed a concept to quantify the objectives of all parties, aiming at automated calculation of EV charging strategies. In this context, we outlined a cloud-based framework for data acquisition, storage and application as a common basis for data-driven analyses. As an example use case, we proposed two data-driven models based on linear regression and neural-network regression to estimate specific consumption, i.e. consumed energy per driven distance. Therefore, we used time series data collected from a fleet of ten connected EVs. For the model training only eight out of ten EVs were used, while the remaining two EVs were used for validation.

The linear model seems to inadequately represent the true relation between input features and consumption. However, the neural network model with five hidden layers yields a mean relative absolute error of 1.78%. Considering the underlying data, the proposed model outperforms estimators based on physical models as described in the literature. More elaborate model validation with different data and other models, such as polynomial models, or other neural network architectures may allow further model precision improvement.

In future work, a model as proposed could estimate battery aging, if the training data is solely obtained from EVs, whose batteries have not degraded yet.

Comparing the estimated consumption with the actual consumption for EVs, which have degraded batteries, might indicate battery aging due to increased internal losses. This method would allow an entirely data-driven battery aging estimation.

Furthermore, the remaining model parts of the proposed concept require a more elaborate implementation and evaluation. The interactions among EV user, power supplier, and car manufacturer should be analyzed and described mathematically. Finally, the findings need to be combined in an optimization approach, enabling the automated calculation of individualized EV charging strategies considering momentary grid load and battery preservation. For concept evaluation, we plan to use an experimental EV fleet within measure campaigns in the *Energy Lab 2.0* [47]. Concluding, the feasibility of grid-optimal, battery-preserving and individualized charging strategies needs to be investigated with adequate indexes, such as user acceptance.

References

[1] International Energy Agency, "Global EV outlook 2018: Towards cross-modal electrification," 2018.

[2] M. Eider, M. Stolba, D. Sellner, A. Berl, R. Basmadjian, H. de Meer, S. Klingert, T. Schulze, F. Kutzner, and C. Kacperski, "Seamless electromobility," in *Proceedings of the Eighth International Conference on Future Energy Systems - e-Energy 17*. ACM Press, 2017.

[3] G. Broadbent, G. Metternicht, and D. Drozdzewski, "An analysis of consumer incentives in support of electric vehicle uptake: An Australian case study," *World Electric Vehicle Journal*, vol. 10, no. 1, p. 11, 2019.

[4] M. Lützenberger, N. Masuch, T. Küster, D. Freund, M. Voß, C.-E. Hrabia, D. Pozo, J. Fähndrich, F. Trollmann, J. Keiser, and S. Albayrak, "A common approach to intelligent energy and mobility services in a smart city environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 6, no. 3, pp. 337–350, 2015.

[5] M. Cuchý, M. Štolba, and M. Jakob, "Whole day mobility planning with electric vehicles," in *Proceedings of the 10th International Conference on Agents and Artificial Intelligence*. SCITEPRESS - Science and Technology Publications, 2018.

[6] K. Schwenk, M. Faix, R. Mikut, V. Hagenmeyer, and R. R. Appino, "On calendar-based scheduling for user-friendly charging of plug-in electric vehicles," in *2019 2nd IEEE Connected and Autonomous Vehicles Symposium (CAVS)*, 2019.

[7] L. Liu, F. Kong, X. Liu, Y. Peng, and Q. Wang, "A review on electric vehicles interacting with renewable energy in smart grid," *Renewable and Sustainable Energy Reviews*, vol. 51, pp. 648–661, 2015.

[8] K. M. Tan, V. K. Ramachandaramurthy, and J. Y. Yong, "Integration of electric vehicles in smart grid: A review on vehicle to grid technologies and optimization techniques," *Renewable and Sustainable Energy Reviews*, vol. 53, pp. 720–732, 2016.

[9] M. Allison, E. Akakabota, and G. Pillai, "Future load profiles under scenarios of increasing renewable generation and electric transport," in *2018 5th International Conference on Renewable Energy: Generation and Applications (ICREGA)*. IEEE, 2018.

[10] P. Aliasghari, B. Mohammadi-Ivatloo, M. Alipour, M. Abapour, and K. Zare, "Optimal scheduling of plug-in electric vehicles and renewable micro-grid in energy and reserve markets considering demand response program," *Journal of Cleaner Production*, vol. 186, pp. 293–303, 2018.

[11] R. Jarvis and P. Moses, "Smart grid congestion caused by plug-in electric vehicle charging," in *2019 IEEE Texas Power and Energy Conference (TPEC)*. IEEE, 2019.

[12] M. Alipour, B. Mohammadi-Ivatloo, M. Moradi-Dalvand, and K. Zare, "Stochastic scheduling of aggregators of plug-in electric vehicles for participation in energy and ancillary service markets," *Energy*, vol. 118, pp. 1168–1179, 2017.

[13] M. R. Mozafar, M. H. Moradi, and M. H. Amini, "A simultaneous approach for optimal allocation of renewable energy sources and electric vehicle charging stations in smart grids based on improved GA-PSO algorithm," *Sustainable Cities and Society*, vol. 32, pp. 627–637, 2017.

[14] R. R. Appino, M. Munoz-Ortiz, J. A. G. Ordiano, R. Mikut, V. Hagenmeyer, and T. Faulwasser, "Reliable dispatch of renewable generation via charging of time-varying PEV populations," *IEEE Transactions on Power Systems*, vol. 34, no. 2, pp. 1558–1568, 2019.

[15] L. K. Maia, L. Drünert, F. L. Mantia, and E. Zondervan, "Expanding the lifetime of Li-ion batteries through optimization of charging profiles," *Journal of Cleaner Production*, vol. 225, pp. 928–938, 2019.

[16] A. D. Giorgio, F. Liberati, and S. Canale, "Electric vehicles charging control in a smart grid: A model predictive control approach," *Control Engineering Practice*, vol. 22, pp. 147–162, 2014.

[17] A. Rajaei, M. Jooshaki, M. Fotuhi-Firuzabad, and M. Moeini-Aghtaie, "Enhancing power distribution system flexibility using electric vehicle charging management," in *2019 27th Iranian Conference on Electrical Engineering (ICEE)*. IEEE, 2019.

[18] N. Kühl, M. Goutier, A. Ensslen, and P. Jochem, "Literature vs. twitter: Empirical insights on customer needs in e-mobility," *Journal of Cleaner Production*, vol. 213, pp. 508–520, 2019.

[19] E. Azadfar, V. Sreeram, and D. Harries, "The investigation of the major factors influencing plug-in electric vehicle driving patterns and charging behaviour," *Renewable and Sustainable Energy Reviews*, vol. 42, pp. 1065–1076, 2015.

[20] D. Lüddecke, C. Seidl, J. Schneider, and I. Schaefer, "Modeling context-aware and intention-aware in-car infotainment systems," *Software & Systems Modeling*, vol. 17, no. 3, pp. 973–987, 2016.

[21] R. R. Desai, R. B. Chen, and W. Armington, "A pattern analysis of daily electric vehicle charging profiles: Operational efficiency and

environmental impacts," *Journal of Advanced Transportation*, vol. 2018, pp. 1–15, 2018.

[22] X. Li, Q. Zhang, Z. Peng, A. Wang, and W. Wang, "A data-driven two-level clustering model for driving pattern analysis of electric vehicles and a case study," *Journal of Cleaner Production*, vol. 206, pp. 827–837, 2019.

[23] W. Dong, J. Li, R. Yao, C. Li, T. Yuan, and L. Wang, "Characterizing driving styles with deep learning," IBM Research China, 2016.

[24] S. Sun, J. Zhang, J. Bi, and Y. Wang, "A machine learning method for predicting driving range of battery electric vehicles," *Journal of Advanced Transportation*, vol. 2019, pp. 1–14, 2019.

[25] J. Neubauer, A. Brooker, and E. Wood, "Sensitivity of battery electric vehicle economics to drive patterns, vehicle range, and charge strategies," *Journal of Power Sources*, vol. 209, pp. 269–277, 2012.

[26] G. Smith, J. Sochor, and I. M. Karlsson, "Mobility as a service: Development scenarios and implications for public transport," *Research in Transportation Economics*, vol. 69, pp. 592–599, 2018.

[27] C.-H. Lee and C.-H. Wu, "A novel big data modeling method for improving driving range estimation of EVs," *IEEE Access*, vol. 3, pp. 1980–1993, 2015.

[28] Y. Zheng, J. Wang, C. Qin, L. Lu, X. Han, and M. Ouyang, "A novel capacity estimation method based on charging curve sections for lithium-ion batteries in electric vehicles," *Energy*, vol. 185, pp. 361–371, 2019.

[29] A. Eddahech, O. Briat, N. Bertrand, J.-Y. Delétage, and J.-M. Vinassa, "Behavior and state-of-health monitoring of Li-ion batteries using impedance spectroscopy and recurrent neural networks," *International Journal of Electrical Power & Energy Systems*, vol. 42, no. 1, pp. 487–494, 2012.

[30] M. Hannan, M. Lipu, A. Hussain, and A. Mohamed, "A review of lithium-ion battery state of charge estimation and management system in electric

vehicle applications: Challenges and recommendations," *Renewable and Sustainable Energy Reviews*, vol. 78, pp. 834–854, 2017.

[31] D. D. Susilo, A. Widodo, T. Prahasto, and M. Nizam, "State of health estimation of lithium-ion batteries based on combination of gaussian distribution data and least squares support vector machines regression," *Materials Science Forum*, vol. 929, pp. 93–102, 2018.

[32] Y. Li, S. Zhong, Q. Zhong, and K. Shi, "Lithium-ion battery state of health monitoring based on ensemble learning," *IEEE Access*, vol. 7, pp. 8754–8762, 2019.

[33] M. Berecibar, I. Gandiaga, I. Villarreal, N. Omar, J. V. Mierlo, and P. V. den Bossche, "Critical review of state of health estimation methods of Li-ion batteries for real applications," *Renewable and Sustainable Energy Reviews*, vol. 56, pp. 572–587, 2016.

[34] O. Sundstrom and C. Binding, "Planning electric-drive vehicle charging under constrained grid conditions," in *2010 International Conference on Power System Technology*. IEEE, 2010.

[35] M. Majidpour, C. Qiu, P. Chu, H. R. Pota, and R. Gadh, "Forecasting the EV charging load based on customer profile or station measurement?" *Applied Energy*, vol. 163, pp. 134–141, 2016.

[36] J. González, R. Alvaro, C. Gamallo, M. Fuentes, J. Fraile-Ardanuy, L. Knapen, and D. Janssens, "Determining Electric Vehicle Charging Point Locations Considering Drivers' Daily Activities," *Procedia Computer Science*, vol. 32, pp. 647–654, 2014.

[37] J. Li, X. Sun, Q. Liu, W. Zheng, H. Liu, and J. A. Stankovic, "Planning Electric Vehicle Charging Stations Based on User Charging Behavior," in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2018.

[38] Eurpäisches Parlament und EU Rat, "Datenschutz-Grundverordnung, VERORDNUNG (EU) 2016/679," 2018. [Online]. Available: https://dsgvo-gesetz.de/

[39] D. Chen and H. Zhao, “Data security and privacy protection issues in cloud computing,” in *2012 International Conference on Computer Science and Electronics Engineering*. IEEE, 2012.

[40] M. Baum, J. Dibbelt, A. Gemsa, D. Wagner, and T. Zündorf, “Shortest feasible paths with charging stops for battery electric vehicles,” *Transportation Science*, 2019.

[41] C. Fiori, K. Ahn, and H. A. Rakha, “Power-based electric vehicle energy consumption model: Model development and validation,” *Applied Energy*, vol. 168, pp. 257–268, 2016.

[42] J. Hong, S. Park, and N. Chang, “Accurate remaining range estimation for electric vehicles,” in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2016.

[43] S. Weisberg, *Applied linear regression*, 3rd ed. Wiley-Interscience, 2005.

[44] Python Software Foundation, “Python,” 2019. [Online]. Available: https://www.python.org/

[45] F. Chollet *et al.*, “Keras,” 2015. [Online]. Available: https://keras.io

[46] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014.

[47] V. Hagenmeyer, H. K. Cakmak, C. Düpmeier, T. Faulwasser, J. Isele, H. B. Keller, P. Kohlhepp, U. Kühnapfel, U. Stucky, S. Waczowicz, and R. Mikut, “Information and communication technology in Energy Lab 2.0: Smart energies system simulation and control center with an Open-Street-Map-based power flow simulation example,” *Energy Technology*, vol. 4, pp. 145–162, 2016.

Content Representation for Neural Style Transfer Algorithms based on Structural Similarity

Philip Meier, Volker Lohweg

inIT – Institute Industrial IT
Technische Hochschule Ostwestfalen-Lippe
Campusallee 6, 32657 Lemgo, Germany
{philip.meier,volker.lohweg}@th-owl.de

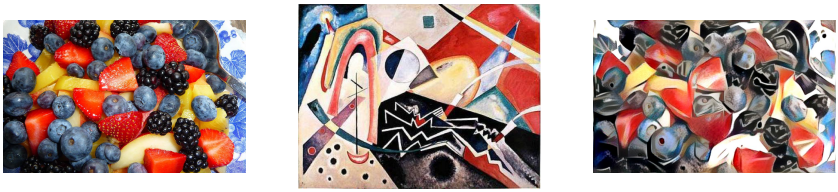
1 Introduction

Neural style transfer (NST) techniques allow to automatically merge the content of one image and the artistic style of another. This goal can be intuitively conveyed to a layperson with only three images (cf. Figure 1). Due to this simplicity, NST has gained some attraction for recreational use. In this context it is desirable to have a general purpose algorithm that is able to yield good results for a large variety of input images. If the quality of the new image is poor, the user might accept it anyway or simply move on without further consequences.

This changes drastically if NST is embedded into a professional environment, since it is usually not possible to swap out the inputs. For a single image and simple artistic styles a skilled artisan might be able to correct the results manually. While this partially defeats the goal to perform the stylisation automatically, it is also not feasible for more complex tasks. For example the stylisation of a video has to be performed for every frame. With a reasonable frame rate even short video clips comprise enough frames to render the manual stylisation practically impossible or at least infeasible. Even for single images the stylisation with a complex style such as the *intaglio* style [1] may

be very time consuming. The transfer of the intaglio style to an image is currently performed manually which takes a skilled artisan around three months to complete.

It is thus crucial to a have a variety of alternative methods to choose from in order to increase the chances that at least one approach yields sufficient results. Within this contribution we propose a drop-in alternative to the way the content is represented within an NST algorithm; an area that so far has received little attention. First, however, we present the work related to our approach.



(a) (b) (c)

Figure 1: Example of a NST by merging the content of (a) and style of (b) into a new image (c) with our proposed approach. Details about the utilised images and methods are presented in Section 4.

2 Related Work

Before presenting the related work in the following subsections, we introduce the notation we utilised throughout this contribution. Matrices are denoted by bold and not-italicised letters, e. g. a greyscale image of height H and width W could be denoted by $\mathbf{I} \in \mathbb{R}^{H \times W}$. Tensors with more than two dimensions are denoted by bold and not-italicised letters without serifs, e. g. an RGB image of the same spatial size as before with $C = 3$ channels could be denoted by $\mathsf{I} \in \mathbb{R}^{H \times W \times C}$. Furthermore, unless explicitly stated otherwise, all elementary arithmetic operations as well as assertions are performed elementwise. Finally, we utilise the sum symbol $\sum$ without indices to denote the sum of elements of the following matrix or tensor. In the same style, the average of a matrix or tensor, i. e. the elementwise sum divided by the number of elements, is denoted by $\overline{\sum}$.

2.1 Neural style transfer

The field of NST emerged in 2016 as a subfield of non-photorealistic rendering (NPR) pioneered by GATYS et al. [2]. NST algorithms allow to merge the content of one image I_C with style of another I_S into a new image I (cf. Figure 1). Opposed to traditional NPR algorithms [3], NST methods do not operate on the pixel space or a handcrafted feature space, but rather on a learned feature space of a pre-trained convolutional neural network (CNN). This CNN is called *encoder* E, where $E^l(\mathsf{I})$ denotes the returned feature map of the image I by passing it through the CNN up to and including layer l.

In general, the synthesis is performed by optimising the image I in order to minimise a loss function $\mathscr{L}$:

$$\operatorname{argmin}_{\mathsf{I}} \mathscr{L}(\mathsf{I}, \mathsf{I}_\mathrm{C}, \mathsf{I}_\mathrm{S}). \tag{1}$$

The loss function $\mathscr{L}$ is called *perceptual loss* and represents how well the synthesized image portrays the targeted content and style. In the most common formulation the perceptual loss $\mathscr{L}$ is a weighted sum of two loss functions $\mathscr{L}_\mathrm{C}$ and $\mathscr{L}_\mathrm{S}$, which represent the content and style agreement independently:

$$\mathscr{L}(\mathsf{I}, \mathsf{I}_\mathrm{C}, \mathsf{I}_\mathrm{S}) = \lambda_\mathrm{C} \cdot \mathscr{L}_\mathrm{C}(\mathsf{I}, \mathsf{I}_\mathrm{C}) + \lambda_\mathrm{S} \cdot \mathscr{L}_\mathrm{S}(\mathsf{I}, \mathsf{I}_\mathrm{S}). \tag{2}$$

The weights λ_C and λ_S are utilised to steer the synthesis towards a focus on content or style. Setting one of the weights to zero, the NST algorithm is transformed into a neural content reconstruction (NCR) or a style synthesis, respectively.

The content agreement is represented with the squared error (SE) between the feature maps of the image I and the content image I_C on the layer l_C of the encoder E:

$$\mathscr{L}_{\mathrm{C,SE}}(\mathsf{I}, \mathsf{I}_\mathrm{C}) = \overline{\sum}\left(E^{l_\mathrm{C}}(\mathsf{I}) - E^{l_\mathrm{C}}(\mathsf{I}_\mathrm{C})\right)^2. \tag{3}$$

The feature maps on layer l_C are sparse representations of the objects or the content of the given image. For the sake of an expressive representation, it is crucial that the encoder is able to recognize the motifs within the images. Thus, it is common to utilise an encoder trained on large datasets to include a high degree of variety of motifs. Furthermore, the layer l_C is usually picked from deep within the CNN. Since the depth of the layer correlates with the abstraction of the recognised objects, a deeper layer ensures that the feature maps represent the actual content invariantly to irrelevant details [4].

GATYS et al. [5] suggest to treat the style of an image I as texture. Thus, NST is closely related to the field of texture synthesis. One possible approach to represent the texture of a feature map is a stochastic one [6]. This assumes that if a given statistic of two feature maps match, the underlying texture and thus style is the same. In the original formulation the authors chose a correlation based approach. Given a set of column vectors arranged in a matrix X, the matrix product, i. e. not the elementwise product, of its transposed version with itself is the GRAM matrix [7]:

$$\text{gram} : \mathbb{R}^{N\times C} \to \mathbb{R}^{C\times C}, \quad \text{gram}(\mathrm{X}) = \mathrm{X}^T \cdot \mathrm{X}.$$

Depending on its position, each element within the GRAM matrix is the inner product of the corresponding column vectors. Thus, the GRAM matrix is an efficient way to calculate the correlation between all combinations of channels of a feature map. With this the style agreement for layer l_S is represented by the SE of the normalised GRAM matrices:

$$\begin{aligned} \mathscr{L}_{\mathrm{S},l_\mathrm{S}}(\mathsf{I},\mathsf{I}_\mathrm{S}) &= \overline{\sum}\left(g\left(E^{l_\mathrm{S}}(\mathsf{I})\right) - g\left(E^{l_\mathrm{S}}(\mathsf{I}_\mathrm{S})\right)\right)^2 \\ \text{with} \quad g : \mathbb{R}^{H\times W\times C} &\to \mathbb{R}^{C\times C}, \quad g(\mathsf{X}) = \frac{\text{gram}(\text{spatvec}(\mathsf{X}))}{H\cdot W}. \end{aligned} \tag{4}$$

Here, spatvec$(\cdot)$ denotes a special case of the vectorisation function [7] that only collapses the spatial dimensions. Opposed to the content loss $\mathscr{L}_{\mathrm{C,SE}}$ in (3), the overall style loss $\mathscr{L}_\mathrm{S}$ includes a weighted sum over multiple layers L_S in order to capture style elements of varying sizes and levels of detail:

$$\mathscr{L}_\mathrm{S}(\mathsf{I},\mathsf{I}_\mathrm{S}) = \sum_{l_\mathrm{S}\in L_\mathrm{S}} \lambda_{l_\mathrm{S}} \cdot \mathscr{L}_{\mathrm{S},l_\mathrm{S}}.$$

In the few years since the emergence of NST a multitude of variants were developed. The following overview does not claim to be exhaustive. The authors refer the interested reader to [8] for a more complete review.

The NST approach explained above is usually dubbed *slow* NST, since the synthesis of a single image is computationally expensive. For each iteration step of (1) the synthesised image I has to be passed through the encoder E. To overcome this JOHNSON et al. [9] and ULYANOV et al. [10] independently proposed to train a *styliser* CNN S with a variant of the perceptual loss in (2):

$$\operatorname{argmin} S \mathscr{L}(\mathsf{I}, \mathsf{I}_\mathrm{C}, \mathsf{I}_\mathrm{S}) \quad \text{with} \quad \mathsf{I} = S(\mathsf{I}_\mathrm{C}).$$

After the training phase, the styliser S performs stylisation of an image I in a single pass. These methods are thus called *fast* NST. Depending on the architecture, the styliser can generate single [9, 10, 11], multiple [12], or arbitrary styles [13]. Although the fast methods achieve remarkable results with respect to the computing cost, the slow or iterative method is still regarded as *gold standard* and is usually utilised as ground truth for comparison [8].

The field of NST suffers from a lack of objective methods to compare the results of different approaches. Usually the evaluation is performed only qualitatively by comparing few selected images [14, 15]. The validity of this approach is fairly low, since the achieved results might not be generalisable to arbitrary images.

SANAKOYEU et al. [11] presented professional art historians with sets of stylisation results generated by different algorithms and let them decide which method performed best. While this excludes the subjectiveness of the authors, due to the human judgement, albeit qualified, it is still not objective. Furthermore this measure is hardly reproducible, since even if other researchers had access to the same professionals, it is questionable if their judgement is invariant.

SANAKOYEU et al. [11] also introduced the *style transfer deception rate* to assess the quality. They trained a CNN to classify paintings according to their artists. The style transfer deception rate is defined as the fraction of stylised images that fool this CNN and are classified as originals from the

respective artist. This is an objective and reproducible measure, but without further investigation it is unknown, if the classifier infers as expected.

We acknowledge that the objective comparison of NST algorithms is an open field of research, but we will not expand on it within this contribution. An objective quality assessment of unstylised images is presented in the following subsection.

This lack of objective comparison did not hinder the development of other NST variants. Many authors deal with alternative formulations of the style loss in (4). Other stochastic approaches include for example the channelwise histogram matching of the feature maps [15]. LI and WAND [16] proposed to represent the style with a structural approach [6] by matching spatial patches extracted out of the feature maps.

In contrast, to the best knowledge of the authors, no drop-in replacement for the content loss has been formulated. SANAKOYEU et al. propose a *style aware content loss* as part of a fast NST architecture [11]. It is still based on the SE of feature maps, but opposed to other methods the feature maps are extracted out of the trainable styliser S instead of a pretrained and fixed encoder E.

This contribution will push into this void and propose an alternative to the content loss in (3).

2.2 Structural similarity

The field of image quality assessment objectively measures the quality of images. For example, given a reference or ground truth image, methods aim to quantify the amount of distortion introduced by a reconstruction from a lossy compression or an NCR. Traditional methods include the SE or the peak-signal-to-noise ratio. Their meaning is however limited, since they align poorly with the human perception, i. e. multiple images that have the same SE towards the ground truth might have vastly different qualities to a human observer.

In order to overcome this, WANG et al. [17] introduced the structural similarity (SSIM) index. The SSIM index compares the luminance, contrast, and structure of two greyscale images I and $\mathrm{I_R}$ independently of each other. The image

I is a distorted version of the reference image $\mathrm{I_R}$. The SSIM components may vary significantly throughout the images and thus the SSIM index is calculated locally with a window w normalised to unit sum $\sum \mathrm{w} = 1$. With this windowed approach the SSIM index is not a scalar measure, but rather a map for local intensity of the distortion. It is calculated as follows:

$$SSIM(\mathrm{I}, \mathrm{I_R}) = l(\mathrm{I}, \mathrm{I_R})^{\alpha} \cdot c(\mathrm{I}, \mathrm{I_R})^{\beta} \cdot s(\mathrm{I}, \mathrm{I_R})^{\gamma}. \tag{5}$$

Here l, c, and s denote the luminance, contrast, and structure component, respectively. The exponents α, β, and γ are used to adjust the weighting between the individual components. The components are calculated as follows:

$$l(\mathrm{I}, \mathrm{I_R}) = \frac{2 \cdot \mathrm{m_I} \cdot \mathrm{m_{I_R}} + \varepsilon_l}{\mathrm{m_I^2} + \mathrm{m_{I_R}^2} + \varepsilon_l}, \tag{6a}$$

$$c(\mathrm{I}, \mathrm{I_R}) = \frac{2 \cdot \mathrm{s_I} \cdot \mathrm{s_{I_R}} + \varepsilon_c}{\mathrm{s_I^2} + \mathrm{s_{I_R}^2} + \varepsilon_c}, \tag{6b}$$

$$s(\mathrm{I}, \mathrm{I_R}) = \frac{\mathrm{s_{I,I_R}} + \varepsilon_s}{\mathrm{s_I} \cdot \mathrm{s_{I_R}} + \varepsilon_s}. \tag{6c}$$

Here $\mathrm{m_I}$, $\mathrm{m_{I_R}}$, $\mathrm{s_I^2}$, $\mathrm{s_{I_R}^2}$, and $\mathrm{s_{I,I_R}}$ are estimators for the local mean, variance, and covariance, respectively. They can be efficiently calculated by a discrete two-dimensional convolution with the window w:

$$\mathrm{m_I} = \mathrm{I} * \mathrm{w}, \quad \mathrm{m_{I_R}} = \mathrm{I_R} * \mathrm{w},$$

In (6) ε_l, ε_c, and ε_s denote small positive constants. They are incorporated to avoid numerical instabilities if the denominators would be close or equal to zero without them. WANG et al. [17] recommend to calculate them with respect to a fixed small constant K and the dynamic range L, i. e. the maximum possible absolute value, of the input values:

$$\varepsilon = (K \cdot L)^2. \tag{8}$$

The SSIM index is symmetric $SSIM(\mathrm{I}, \mathrm{I_R}) = SSIM(\mathrm{I_R}, \mathrm{I})$, is bounded $-1 \leq SSIM(\mathrm{I_R}, \mathrm{I}) \leq 1$, and has a unique maximum for $\mathrm{I_R} = \mathrm{I}$ at $SSIM(\mathrm{I}, \mathrm{I_R}) = 1$, with respect to the input images $\mathrm{I}, \mathrm{I_R} \geq 0$ [17, 18].

One common simplification of the SSIM is to weigh all components equally ($\alpha = \beta = \gamma = 1$) and set $\varepsilon_1 = \varepsilon_l$ as well as $\varepsilon_2 = \varepsilon_s = \varepsilon_c/2$. This results in the simplified SSIM index:

$$sSSIM(\mathrm{I}, \mathrm{I_R}) = S_1(\mathrm{I}, \mathrm{I_R}) \cdot S_2(\mathrm{I}, \mathrm{I_R}).$$

Under these assumptions the non-structural component S_1 is equal to the luminance component l and the contrast and structure component c and s are collapsed into a single structural component S_2:

$$S_1(\mathrm{I}, \mathrm{I_R}) = \frac{2 \cdot \mathrm{m_I} \cdot \mathrm{m_{I_R}} + \varepsilon_1}{\mathrm{m_I^2} + \mathrm{m_{I_R}^2} + \varepsilon_1}, \quad S_2(\mathrm{I}, \mathrm{I_R}) = \frac{\mathrm{s_{I,I_R}} + \varepsilon_2}{\mathrm{s_I^2} + \mathrm{s_{I_R}^2} + \varepsilon_2}.$$

Besides assessing the quality of a reconstructed image I, the SSIM index can also be used to perform the reconstruction [18, 19]. In order to optimise the image quality, $SSIM(\mathrm{I}, \mathrm{I_C})$ has to be maximised. Since optimisations are usually posed as minimisation problems, this is equivalent to minimising $1 - SSIM(\mathrm{I}, \mathrm{I_C})$. BRUNET achieved the best results by minimising the loss

$$\mathscr{L}(\mathrm{I}, \mathrm{I_C}) = \overline{\sum}[(1 - S_1(\mathrm{I}, \mathrm{I_R})) + (1 - S_2(\mathrm{I}, \mathrm{I_R}))], \tag{9}$$

which is the average of the linear approximation of $1 - sSSIM(\mathrm{I}, \mathrm{I_C})$ [18].

3 Approach

The last section explained that the SSIM index is better aligned with human quality perception of images than the SE. The CNNs utilised within NST algorithms are modelled after the human visual system, albeit grossly simplified. The approach we propose within this section is a content loss $\mathscr{L}_{\mathrm{C,\,SSIM}}$ that compares the feature maps $E(\mathrm{I})$ in terms of the SSIM.

Before we go into the details of the proposed approach, we need to prove that the basic properties of the SSIM (cf. Subsection 2.2) also hold for $\mathrm{I}, \mathrm{I}_\mathrm{R} \in \mathbb{R}$. This step is required, since in general and opposed to pixels, feature maps $E(\mathrm{I})$ might contain negative values.

Proposition 1. *The SSIM index is symmetric with respect to the inputs* $SSIM(\mathrm{I}, \mathrm{I}_R) = SSIM(\mathrm{I}_R, \mathrm{I})$ *for* $\mathrm{I}, \mathrm{I}_R \in \mathbb{R}$.

Proof. Since the symmetry of the components in (6) is independent of the extended input space, BRUNETs proof holds [18, p. 40]. □

Proposition 2. *The luminance* $l(\mathrm{I}, \mathrm{I}_R)$ *is bounded by* $-1 < l(\mathrm{I}, \mathrm{I}_R) \leq 1$ *for* $\mathrm{I}, \mathrm{I}_R \in \mathbb{R}$.

Proof. The following is a trivial extension of BRUNETs proof [18, p. 39] for the extended input space. The proposition is proven by contradiction for two cases:

Case 1. $l(\mathrm{I}, \mathrm{I}_\mathrm{R}) \overset{!}{>} 1$

$$
\begin{aligned}
& (\mathrm{m_I} - \mathrm{m_{I_R}})^2 \geq 0 && \Leftrightarrow \quad (\mathrm{m_I} - \mathrm{m_{I_R}})^2 + \varepsilon_l \geq \varepsilon_l \\
\Leftrightarrow \quad & \mathrm{m_I^2} + \mathrm{m_{I_R}^2} + \varepsilon_l \geq 2 \cdot \mathrm{m_I} \cdot \mathrm{m_{I_R}} + \varepsilon_l && \Leftrightarrow \quad \frac{2 \cdot \mathrm{m_I} \cdot \mathrm{m_{I_R}} + \varepsilon_l}{\mathrm{m_I^2} + \mathrm{m_{I_R}^2} + \varepsilon_l} \leq 1 \\
\Leftrightarrow \quad & l(\mathrm{I}, \mathrm{I_R}) \leq 1
\end{aligned}
$$

Case 2. $l(\mathrm{I}, \mathrm{I}_\mathrm{R}) \overset{!}{\leq} -1$

$$
\begin{aligned}
& (\mathrm{m_I} + \mathrm{m_{I_R}})^2 > -2\varepsilon_l && \Leftrightarrow \quad (\mathrm{m_I} - \mathrm{m_{I_R}})^2 + \varepsilon_l > -\varepsilon_l \\
\Leftrightarrow \quad & \mathrm{m_I^2} + \mathrm{m_{I_R}^2} + \varepsilon_l > -2 \cdot \mathrm{m_I} \cdot \mathrm{m_{I_R}} - \varepsilon_l && \Leftrightarrow \quad \frac{2 \cdot \mathrm{m_I} \cdot \mathrm{m_{I_R}} + \varepsilon_l}{\mathrm{m_I^2} + \mathrm{m_{I_R}^2} + \varepsilon_l} > -1 \\
\Leftrightarrow \quad & l(\mathrm{I}, \mathrm{I_R}) > -1
\end{aligned}
$$

□

Proposition 3. *The contrast and the structure components* $c(\mathrm{I}, \mathrm{I}_R)$ *and* $s(\mathrm{I}, \mathrm{I}_R)$ *are bounded by* $0 \leq c(\mathrm{I}, \mathrm{I}_R) \leq 1$ *and* $-1 \leq s(\mathrm{I}, \mathrm{I}_R) \leq 1$ *for* $\mathrm{I}, \mathrm{I}_R \in \mathbb{R}$.

Proof. Since the extended input space does not change the value range of estimated standard deviations $s_I, s_{I_R} \geq 0$ or the estimated covariance $s_{I,I_R} \in [-1, 1]$, BRUNETs proof [18, pp. 39-40] holds. □

Corollary 1. *The SSIM index is bounded by* $-1 \leq SSIM(I, I_R) \leq 1$ *for* $I, I_R \in \mathbb{R}$.

Proof. SSIM index is defined as the product of the individual components in (5). By combining Proposition 2 and Proposition 3, we arrive at the desired conclusion. □

Proposition 4. *The SSIM index has a unique maximum* $SSIM(I, I_R) = 1$ *at* $I = I_R$ *for* $I, I_R \in \mathbb{R}$.

Proof. The proof is structured into two parts: First, we prove that $SSIM(I, I_R) = 1$ if $I = I_R$. Subsequently, we prove that $SSIM(I, I_R) = 1$ is unique for $I = I_R$.

1. By substituting $I = I_R$ in (5), we arrive at the desired conclusion.
2. BRUNET proved that $l(I, I_R) = c(I, I_R) = s(I, I_R) = 1$ and subsequently $SSIM(I, I_R) = 1$ is uniquely obtained by $I = I_R$ independent of the input space. The only other way to obtain $SSIM(I, I_R) = 1$ through a product of the components would be by $l(I, I_R) \stackrel{!}{=} -1$, $c(I, I_R) = 1$, and $s(I, I_R) = -1$. However, this combination is invalid, since after Proposition 2 $l(I, I_R) \stackrel{!}{=} -1$ is not achievable.

□

In summary the SSIM index retains its symmetry, boundedness, and its unique maximum for real-valued inputs. Naturally these properties also apply to the derived simplified SSIM.

Based on (9) we propose

$$\mathscr{L}_{C,\,SSIM}(I, I_C) = \lambda_1 \cdot \mathscr{L}_{C,\,S_1}(I, I_C) + \lambda_2 \cdot \mathscr{L}_{C,\,S_2}(I, I_C)$$

with

$$\mathscr{L}_{\mathrm{C},\,S_1}(\mathsf{I},\mathsf{I}_\mathrm{C}) = \overline{\sum} 1 - S_1\Big(E^{l_\mathrm{C}}(\mathsf{I}), E^{l_\mathrm{C}}(\mathsf{I}_\mathrm{C})\Big),$$
$$\mathscr{L}_{\mathrm{C},\,S_2}(\mathsf{I},\mathsf{I}_\mathrm{C}) = \overline{\sum} 1 - S_2\Big(E^{l_\mathrm{C}}(\mathsf{I}), E^{l_\mathrm{C}}(\mathsf{I}_\mathrm{C})\Big).$$

as a drop-in replacement for the content loss in (3). Opposed to the traditional approach we do not compare every element pair of the feature maps individually, but rather their SSIM in the local neighbourhood of the window w. The weights λ_1 and λ_2 are added to enable an emphasis on the non-structural or structural component, respectively.

We treat each channel pair of the feature maps $E^{l_\mathrm{C}}(\mathsf{I})$ and $E^{l_\mathrm{C}}(\mathsf{I}_\mathrm{C})$ independently, i. e. we calculate an SSIM map for each channel. Since the dynamic range might vary significantly between the channels, the stability constants ε_1 and ε_2 are also calculated channelwise according to (8). This calculation is performed apriori with the feature maps $E^{l_\mathrm{C}}(\mathsf{I}_\mathrm{C})$ of the content image. The other hyperparameters such as the component weights λ_1 and λ_2 as well as the utilised window w are determined empirically in the following section.

4 Evaluation

This section is divided into three subsections. At first, we describe the methods we utilised throughout this section[1]. Secondly, an objective evaluation of the proposed content loss $\mathscr{L}_{\mathrm{C,\,SSIM}}$ compared to the traditional formulation $\mathscr{L}_{\mathrm{C,\,SE}}$ is carried out. Lastly, a qualitatively comparison is performed with the content losses as part of an NST.

4.1 Methods

NCR and NST should be ideally feasible for arbitrary content images given that the motifs are known to the encoder and it thus is able to generate meaningful

[1] We publish the source code to reproduce our results under `https://github.com/pmeier/GMA_CI_2019_ssim_content_loss`

encodings. For the evaluation we utilise the *NPRGeneral* dataset as content images which include wide range of features [20]. The motifs comprise humans, objects, and landscapes. We center-cropped all images to 1024×640 pixels, which is the size of the smallest image in the dataset. An overview of the dataset is depicted in Figure 5 in the appendix. As style images we utilised *White Zig Zags* by KANDINSKY and *Landscape at Saint-Rémy* by VAN GOGH (cf. Figure 6 in the appendix). The former comprises large abstract forms as style elements while the latter is characterized by fine brush strokes. Both images are part of unnamed style image dataset proposed in [8].

For the image synthesis we mostly follow the procedure proposed in [21] which achieved good results and makes our results comparable. We utilise *VGG19* CNN [22] as encoder E. The CNN was trained on the *ILSVRC 2012* dataset [23] which is suitable for the motifs contained in the NPRGeneral dataset. Although the weights were trained with the *Caffe* framework [24], we implement our approach with the *PyTorch* framework [25]. We use $l_\mathrm{C} = \texttt{relu\_4\_2}$ and $l_\mathrm{S} \in L_\mathrm{S} = \{\texttt{relu\_1\_1}, \texttt{relu\_2\_1}, \texttt{relu\_3\_1}, \texttt{relu\_4\_1}, \texttt{relu\_5\_1}\}$ as layers for content and style feature maps, respectively. The style weights λ_{l_S} are calculated by $\lambda_{l_\mathrm{S}} = 1/n_{l_\mathrm{S}}^2$, where n_{l_S} denotes the number of channels of the feature map on layer l_S.

We also utilise an image pyramid with two levels to enhance the synthesis results. For the first level we stick to the procedure of [21] and resize the images to 800×500 pixels. On the second and final level we only increase the image size of the synthesized image I to 1024×640 pixels. The reference images are used in their unaltered form to avoid possibly introducing resampling artifacts in the ground truth. The resizing is performed with bilinear resampling. The synthesis is carried out by the L-BFGS optimisation algorithm [26] with 500 steps on the first and 200 steps on the second level as proposed in [21].

4.2 Neural Content Reconstruction

While NST algorithms lack a method to assess or compare their performance objectively, this is possible for NCR algorithms. Since the target of the reconstruction I_C is known apriori, we define the mean SSIM of the reconstructed

image I and the target image $I_R = I_C$ as *SSIM score*. Both images are priorly converted to greyscale with standard channel weights [27].

The reconstruction is started from a white noise image to avoid any bias from pre-existing content. Furthermore we repeat every reconstruction with five different random seeds and report the median result in order to reduce the influence of random effects. We explicitly state if we deviate from this scheme.

We set $\lambda_{C,\,SE} = 10^{-3}$ and $\lambda_{C,\,SSIM} = 10^{3}$ to achieve approximately the same overall magnitude of the content loss $\mathscr{L}_C$. The stability constants are calculated according to (8) with $K_1 = 10^{-2}$ and $K_2 = 3 \cdot 10^{-2}$ as suggested in [17]. If not stated otherwise we use $\lambda_1 = 0.1$ and $\lambda_2 = 0.9$ as component weights and a 3×3 GAUSSIAN with standard deviation $\sigma = 1/3$ as window w. The determination of these parameters is explained in the following experiments.

Table 1: Median SSIM score for an NCR of selected images in the NPRGeneral dataset with different loss functions.

image	$\mathscr{L}_{C,\,SE}$	$\mathscr{L}_{C,\,SSIM}$			
		$\rho = 0$	$\rho = 3$	$\rho = 9$	$\rho \to \infty$
arch	0.14	0.11	0.12	0.13	0.12
athletes	0.52	0.42	0.48	0.50	0.53
berries	0.33	0.26	0.31	0.33	0.34
mac	0.48	0.37	0.46	0.47	0.45
snow	0.24	0.17	0.21	0.00	0.00

In a first experiment we benchmarked our SSIM based content loss $\mathscr{L}_{C,\,SSIM}$ with different component weights λ_1 and λ_2 against SE based $\mathscr{L}_{C,\,SE}$ with the NPRGeneral dataset. The SSIM component weights λ_1 and λ_1 were chosen, such that $\rho = \lambda_2/\lambda_1$ and $\lambda_1 + \lambda_2 = 1$. Table 1 depicts the results for selected images, while the results for the complete dataset are displayed in Table 2 in the appendix. In general, we observe that for our approach a higher structural weight λ_2 leads to an improved performance. The SSIM score with only the structural component $\rho \to \infty$ is similar compared to the traditional approach, while the better performing approach is dependent on the image. For the images *mac*, *oparara*, and *arch* the SSIM score drops off for higher structural

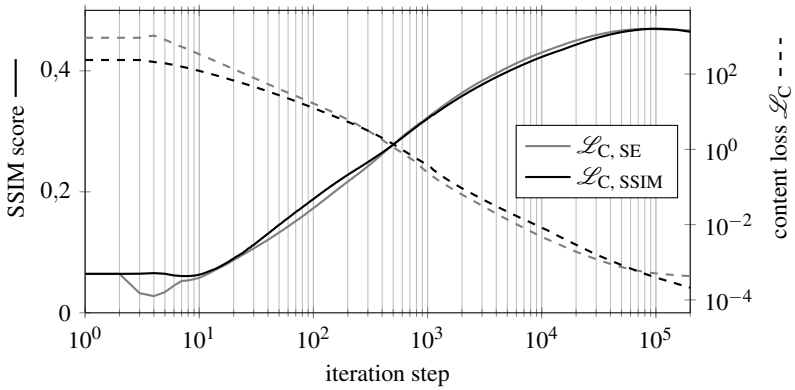


Figure 2: SSIM score and content loss $\mathscr{L}_C$ over the course of an optimisation. The data was recorded in a single run and without utilising the image pyramid.

weights λ_2. This is especially true for the images *snow* and *daisy*: since they comprise large homogenous areas with little structure, our approach is not able to synthesise any content for a high structure weight λ_2 in the majority of runs. The other extreme, i. e. non-structural only $\rho = 0$, is stable, but performs significantly worse than the traditional approach. This implies that a good balance between the component weights has to be found on a per-image basis. For the following experiments we chose $\lambda_1 = 0.1$ and $\lambda_2 = 0.9$ ($\rho = 9$), since it performed reasonably well for most images within NPRGeneral dataset. Furthermore, in order to reduce the evaluation space, we chose the image *berries* as proxy, because both approaches performed very similarly on it. Additionally, its SSIM scores are approximately in the middle between the best and worst reconstructible images *athletes* and *arch*, respectively.

Ideally, the former benchmark should be carried out on steady state behaviour, i. e. further optimisation steps should have negligible effects. Figure 2 depicts that a steady state of the SSIM score is not achieved within the usual range of $\leq 10^3$ steps. Both approaches only plateau after approximately 10^5 steps. Continuing the NCR at this point decreases the SSIM score again. This effect results from unconstrained optimisation: every pixel that gets fit by the optimisation outside of the closed interval $[0, 1]$ becomes salt and pepper noise after

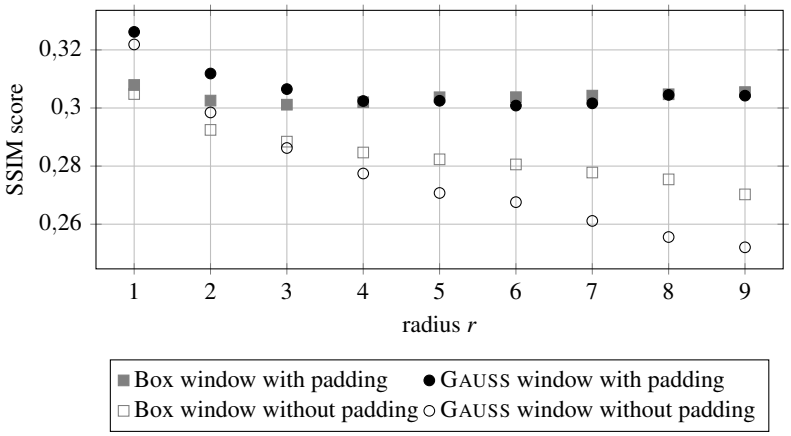


Figure 3: Median SSIM score of the NCR with the image *berries* over different window radii. The window w is square with a side length of $2r+1$ pixels. The standard deviation σ of the GAUSSIAN is set to $\sigma = r/3$.

clipping and thus reduces the quality. Our approach is stronger affected by this, since $\mathscr{L}_{\mathrm{C,\,SSIM}}$ does not stagnate opposed to $\mathscr{L}_{\mathrm{C,\,SE}}$.

Lastly, the window w has to be determined. Figure 3 depicts the SSIM scores for a box or GAUSS window for varying radii. In general, the performance degenerates for a larger radius r. This is a result of the significantly increased *receptive field* of $84+16\cdot r$ pixel of the window w on the layer $l_{\mathrm{C}} = \texttt{relu\_4\_2}$. Within this contribution the receptive field is defined as the largest object in pixels within an image that can completely fit into a local neighbourhood of sequential convolutions or similar operations. The factor 16 in the receptive field calculation is formed by the three prior pooling operations within the encoder E with a stride of 2 and the fact that radius counts twice for the size of the window w. Since larger receptive fields correspond to larger recognisable objects, they tend to suppress finer details, which in turn lowers the performance. Furthermore, Figure 3 shows that padding the feature maps prior to applying the window w is beneficial. If this step is omitted, the area near the edges is under-represented, which leads to a performance loss. This effect is amplified for greater radii. All in all, we chose a GAUSS window with a radius $r = 1$ and padding for the further experiments.

In conclusion, we observed that no approach is objectively better than the respective other one within the carried out experiments. In general, our approach performs best with an emphasised structural component and small windows. After this quantitative evaluation we now compare our approach and the traditional qualitatively by incorporating them in an NST.

4.3 Neural Style Transfer

The synthesis of the stylised image is started from the content image I_C, since the result converges faster to a visually appealing result. We set the style weight to $\lambda_S = 1$, which leads to the suggested weight ratio $\lambda_C/\lambda_S = 10^{-3}$ from [21] for the traditional approach. The style images are also resized with the image pyramid, but are kept in their original aspect ratio.

We performed an NST with all combinations of content images I_C from the NPRGeneral dataset and style images I_S. Within this subsection we only present parts of this experiment, but the interested reader can find all results in the accompanying repository.

The image *White Zig Zags* by KANDINSKY (cf. Figure 6 in teh appendix) features large style elements which are separated by sharp edges. With this style both approaches yield virtually indistinguishable results (cf. Figure 7 in the appendix). *A Landscape at Saint-Rémy* by VAN GOGH on the other hand comprises fine brush strokes with mostly low contrast between them. In the synthesised image, our approach favours high contrast brush strokes (cf. Figure 4): due to the higher structural emphasis of the content loss, a high contrast style element minimises both the content and style loss. The resulting sharp edges are partially dominant enough to distort the content. This is especially visible for human faces, since the human mind is trained to subconsciously notice even slight differences. Although they are not part of this initial evaluation, styles which feature high contrast elements could benefit from our approach. Although both approaches struggle to stylise the homogenous areas in Figure 4, the traditional approach copes better with it. This is again a result of structure emphasis of our approach within an area without any structure.

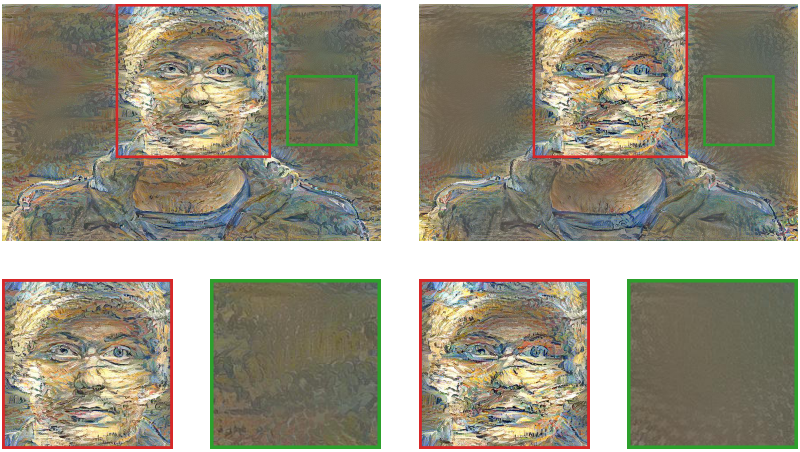


Figure 4: Results of the NST of the content image *rim lighting* and style image *Landscape at Saint-Rémy*. The left column was synthesised with the traditional and the right column with our proposed approach.

5 Conclusion and Outlook

In this contribution we introduced a novel content representation as a drop-in replacement for neural style transfer algorithms. Opposed to the traditional approach it does not rate the content agreement on the squared error of the feature maps, but rather on their structural similarity. In a quantitative comparison, we found that our approach performs similarly to the traditional approach, if utilised within a neural content reconstruction. The better performing approach is determined on a per-image basis, while our approach exhibits some instability for several images depending on the chosen weighting factors. Furthermore, as part of a neural style transfer our approach tends to emphasise high contrast style elements. While this leads to subjectively worse performance for the styles we utilised in our experiments, it might be an advantage for other styles.

Future work can expand on ours by investigating if the results of the qualitative evaluation hold in a more comprehensive study, i. e. performing the neural style transfer with a larger variety of styles. This should include styles with small,

high-contrast elements, e. g. the intaglio style. Additionally, the investigation should evaluate the effect of our proposed content loss in the presence of other style losses.

References

[1] Rudolph L. van Renesse. *Optical Document Security*. Artech House, 3. edition, 2005.

[2] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2001.

[4] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *Computing Research Repository (CoRR)*, 1311, 2013.

[5] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS) 28*, 2015.

[6] Dongxiao Zhou. *Texture analysis and synthesis using a generic Markov-Gibbs image model*. PhD thesis, University of Auckland, 2006.

[7] Leslie Hogben, editor. *Handbook of Linear Algebra*. Chapman & Hall / CRC, 1. edition, 2007.

[8] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, and Mingli Song. Neural style transfer: A review. *Computing Research Repository (CoRR)*, 1705, 2017.

[9] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of the 14th European Conference on Computer Vision (ECCV)*, 2016.

[10] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. *Computing Research Repository (CoRR)*, 1603, 2016.

[11] Artsiom Sanakoyeu, Dmytro Kotovenko, Sabine Lang, and Björn Ommer. A style-aware content loss for real-time hd style transfer. In *Proceedings of the 15th European Conference on Computer Vision (ECCV)*, 2018.

[12] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *Preprint in Computing Research Repository (CoRR)*, 1610, 2016.

[13] Xun Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *Computing Research Repository (CoRR)*, 1703, 2017.

[14] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *Proceedings of the 14th European Conference on Computer Vision (ECCV)*, 2016.

[15] Eric Risser, Pierre Wilmot, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *Computing Research Repository (CoRR)*, 1701, 2017.

[16] Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[17] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4), 2004.

[18] Dominique Brunet. *A Study of the Structural Similarity Image Quality Measure with Applications to Image Processing*. PhD thesis, University of Waterloo, 2012.

[19] Dominique Brunet, Sumohana S. Channappayya, Zhou Wang, Edward R. Vrscay, and Alan C. Bovik. *Optimizing Image Quality*. Springer International Publishing, 2018.

[20] David Mould and Paul L. Rosin. A benchmark image set for evaluating stylization. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling (Expressive) and Non-Photorealistic Animation and Rendering (NPAR)*, 2016.

[21] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. *Computing Research Repository (CoRR)*, 1611, 2017.

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Computing Research Repository (CoRR)*, 2014.

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 2015.

[24] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *Computing Research Repository (CoRR)*, 1408, 2014.

[25] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *Proceedings of the NIPS Autodiff Workshop*, 2017.

[26] Joel Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer-Verlag New York, 2. edition, 2006.

[27] International Telecommunication Union (ITU). Recommendation ITU-R BT.601-7: Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios, 2011.

Appendix

Table 2: Extension of Table 1. Median SSIM score for an NCR of the NPRGeneral dataset with different loss functions.

image	$\mathscr{L}_{\text{C, SE}}$	$\mathscr{L}_{\text{C, SSIM}}$			
		$\rho = 0$	$\rho = 3$	$\rho = 9$	$\rho \to \infty$
angel	0.29	0.20	0.25	0.27	0.29
arch	0.14	0.11	0.12	0.13	0.12
athletes	0.52	0.42	0.48	0.50	0.53
barn	0.27	0.21	0.26	0.27	0.27
berries	0.33	0.26	0.31	0.33	0.34
cabbage	0.41	0.32	0.39	0.41	0.43
cat	0.24	0.20	0.23	0.24	0.25
city	0.33	0.22	0.29	0.31	0.31
daisy	0.50	0.37	0.45	0.46	0.00
dark woods	0.16	0.10	0.14	0.15	0.15
desert	0.33	0.27	0.30	0.31	0.31
headlight	0.49	0.27	0.39	0.43	0.47
mac	0.48	0.37	0.46	0.47	0.45
mountains	0.45	0.32	0.39	0.41	0.44
oparara	0.19	0.12	0.16	0.18	0.17
rim lighting	0.15	0.09	0.12	0.13	0.13
snow	0.24	0.17	0.21	0.00	0.00
tomatoes	0.42	0.25	0.35	0.39	0.41
toque	0.40	0.30	0.37	0.38	0.39
yemeni	0.40	0.26	0.32	0.35	0.37

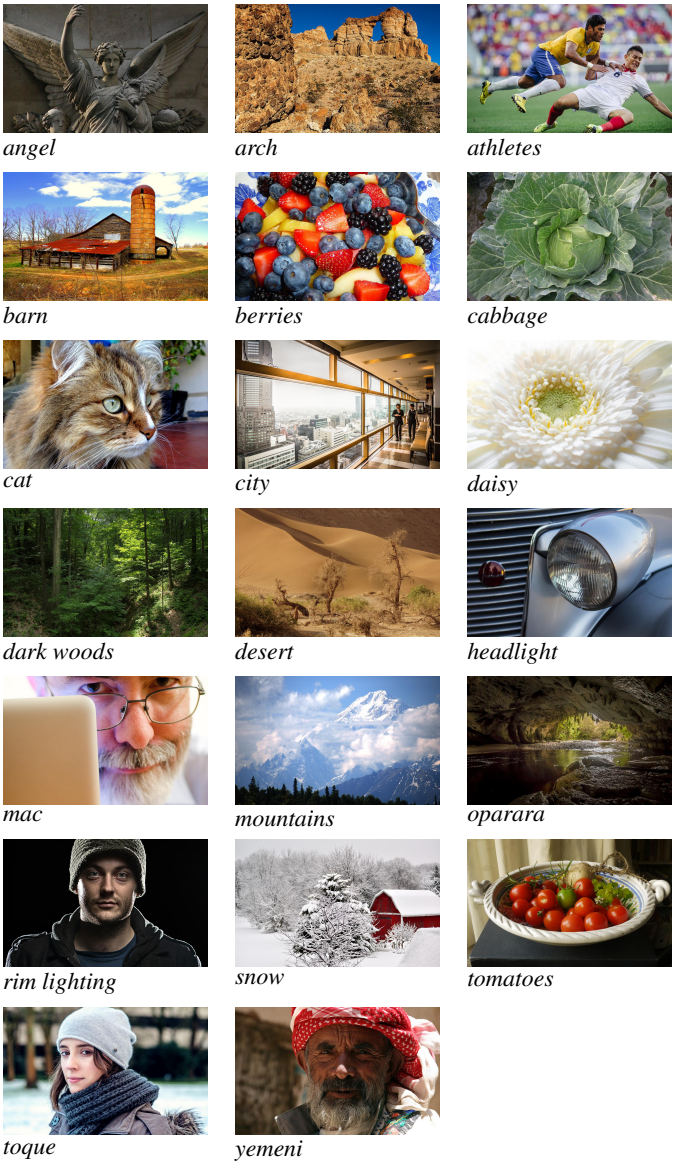


Figure 5: Overview over the images in the NPRgeneral dataset [20], which are utilised as content images.

(a) *White Zig Zags* by KANDINSKY

(b) *Landscape at Saint-Rémy* by VAN GOGH

Figure 6: Overview over the utilised style images.

(a)

(b)

Figure 7: Results of the NST of the content image *berries* and style image *White Zig Zags* with the traditional (a) and our proposed (b) content representation. Without further processing the differences are imperceptible.

Distribution-Based Splitting of Datasets

Timm J. Peter, Dennis Kekec, Oliver Nelles

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

Abstract

The problem of data splitting is discussed and two novel approaches to split datasets based on an efficient evaluation of estimated probability density functions (pdf) are introduced. Due to their structure, the approaches are capable of selecting subsets, that approximate the input point distribution of the original dataset. The first method fills the datasets alternatingly with data points. The second method allocates the points to the dataset with a probability proportional to their fit. Both approaches base on rough a simplistic way of evaluating pdfs, they are estimated using approximate kernel density estimation.

Additionally, an subsequent optimization of the point distribution is introduced, which also relies on pdf estimation. Since datasets are split based on the point distribution of the original dataset, the estimation of pdfs presents a more sophisticated criterion to base a data splitting concept on, compared to existing procedures. In order to validate the functioning of the approaches, their performance is analyzed for two artificial datasets, comparing it to an existing method and the mean performance by random splits. Both approaches and the subsequent optimization lead, on average, to an improved performance. The first method shows the most promising performance improvement.

1 Introduction

The most important determining factor to decide on whether or not a model is of high quality, is its ability to generalize well on unseen data. This ability is decisively influenced by the model complexity. The impact of model complexity occurs as follows: On the one hand, if a model is not complex enough, it is not capable of reproducing the properties of the underlying process properly. The error arising from too simple models among other, less important factors, is called bias error. On the other hand, a too complex model possesses too many parameters to estimate them with the given amount of data, leading to a higher variance error. This context is well-known as the *bias/variance dilemma*. Commonly, hold-out is applied to obtain a trade-off between bias and variance error. In this procedure, the dataset is split up into subsets, e.g. for training and testing.

However, an important detail when performing hold-out is often overlooked: The procedure of how to split up the data into subsets. This task should not be neglected, since model performance variation due to data splitting is greater than variabilities due to other factors, as e.g. model structure selection, random weight initialization or training [8].

Different methods to approach the data splitting problem exist. The most simple but also most frequently used method is to split up data randomly. More advanced approaches use trial-and-error [1] or optimization-based [7] methods to ensure statistical properties, such as mean and standard deviation of the subsets. An alternative is the DUPLEX algorithm [2]. Here, data points are selected based on Euclidean distances, in order to achieve maximum coverage. Another class of methods uses clustering to subdivide the data and sample subset from the resulting clusters [3, 6, 7, 4].

Most of the aforementioned approaches try to replicate statistical features, as mean or standard deviation, of a given dataset or local properties by clustering. In this contribution, the proposed approaches base on the estimation of a more fundamental statistical property for the given input points: The probability density function (pdf). Since the pdf approximates the point distribution of

a dataset and pdf estimation is a well-researched topic, it is a promising technique to base a data splitting method on. Therefore, the goals of this contribution are: (i) to develop a method to split data into subsets of similar estimated pdf values,based on estimating the probability density function, and (ii) to validate the proposed methods concerning their impact on the test error in the model estimation process.

2 Comparison of data splitting methods

Common techniques which are used to develop models are k-fold cross-validation or data splitting in training, test and validation sets or in training and test sets. Here, the focus is on the latter technique.

The task of comparing the performance of different data splitting methods is a rarely discussed topic. Two different procedures to assess data splitting methods exist and are similar in their steps: First, a model is trained. The model training is carried out on the training datasets, according to the evaluated methods. Secondly, the test error is recorded. The data splitting and assessment steps are repeated multiple times in order to build a statistic. Subsequently, (i) the size of the test errors and (ii) the standard deviation of the test errors are assessed.

However, the two procedures differ in their way of evaluating point (i), the size of the test error. The first procedure to rate data splitting methods, used in [4, 6, 7], compares the test errors to zero, assuming a test error of zero being the ideal case. In contrast, [5] proposes a different strategy. This procedure is based on the idea, that if a model is developed and evaluated for each possible split for a given dataset, the population of all validation errors is obtained. They propose, that the expected value of this distribution is the best benchmark or reference to compare the test errors to, since the variability is due to all physical processes and patterns included in the data. Based on the central limit theorem, they formulate an iterative algorithm to approximate the expected value.

The mean of the population of validation errors, as explained in [5], can also be interpreted as the mean performance of a model trained with a random split.

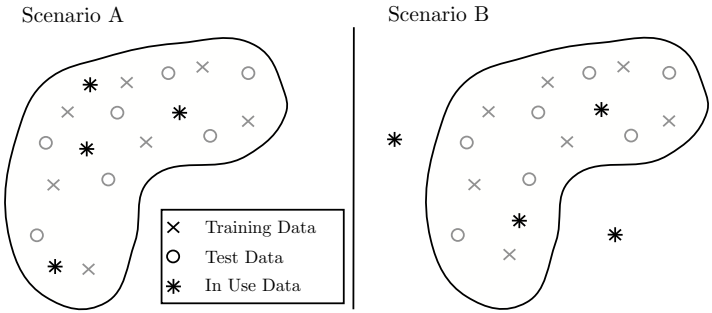


Figure 1: Two scenarios for the realistic estimation of model quality in later use.

Based on this insight, we want to introduce two different goals, that can be distinguished: First, to find the appropriate criterion for performance of data splitting methods and secondly, the realistic estimation of model quality in later use. For the first task, we assume, that the first procedure, comparing the test error to zero, is the superior concept, since we want our data splitting method to pick splits *better* than the mean random split. Nevertheless, the distribution of the data points in training and test set must be taken into account. For the second task, it depends on the application scenario to decide whether the first or second procedure is to prefer. Scenario A, see Fig. 1, is typical for test facilities, where all influence factors stay unchanged or can be actively controlled and new data points always fall into the drawn contour. Here, the first procedure is preferable. Otherwise, scenario B typically appears in industrial environments, thus is the more common, realistic case. New data points fall into the contour, but also outside of it. This is e.g. typical for wearing behavior in production plants. For this task under this scenario, to compare the test error to the performance of the mean random model seems to be more realistic. The reason that randomly split data provokes more extrapolation during testing which realistically captures the real-world behavior in latter model use.

3 Data splitting concepts

In this section, two new approaches to split datasets based on pdf estimation of the input points are presented. Prior to this, the basics of pdf estimation and evaluation are introduced.

3.1 Estimation and evaluation of probability density functions

The first step and basis for distribution-based splitting of datasets is the estimation and evaluation of pdfs. An efficient procedure for this task is proposed in [9]. Here, pdfs are estimated using kernel density estimation and evaluated solely on the points of the given dataset. This results in a drastic reduction of the computational complexity compared to a standard Monte-Carlo evaluation, without decreasing the subset selection performance. The algorithm presented in [9] was proposed originally to select a data subset of an original dataset and proceeds as follows. At first, the pdf of the original dataset is estimated using kernel density estimation. Secondly, two datasets are initialized: One set contains the non-selected data points, one set contains all selected points. At the beginning, the non-selected data points correspond to the original dataset, the dataset of selected points is empty. After the first two initialization steps, the iterative process begins. In each iteration, one data point of the original dataset is selected and added to the selected data points, while the corresponding point is removed from the non-selected data points. Here, the data point, which results in the minimal absolute error between the pdfs of the original dataset and the selected dataset, is chosen. To do so, a pdf is estimated for each candidate dataset and the absolute error between this pdf and the pdf of the original dataset is evaluated. The absolute error is evaluated solely at the points of the selected subset. If the termination criterion, e.g. based on the number of selected points, is met, the iterative procedure is stopped. The procedure of the algorithm is shown in Fig. 2.

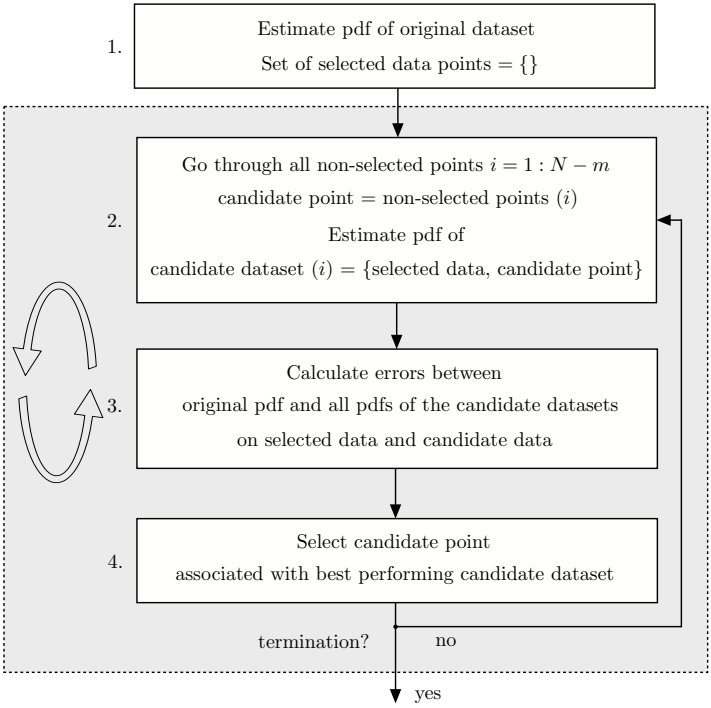


Figure 2: Procedure of the algorithm proposed in [9]

3.2 Method 1: Subset-based data splitting

The first data splitting concept, see Fig. 3, is closely related to the algorithm explained in Sect. 3.1. The differences compared to the original algorithm are pointed out in the following description.

First, the pdf of the original dataset is estimated using kernel density estimation. Secondly, $k + 1$ datasets are initialized: One set contains the yet non-selected data points, k subsets contain the selected points for each subset (training and test datasets in this paper). At the beginning, the non-selected data points correspond to the original dataset, the data subsets of selected points are empty.

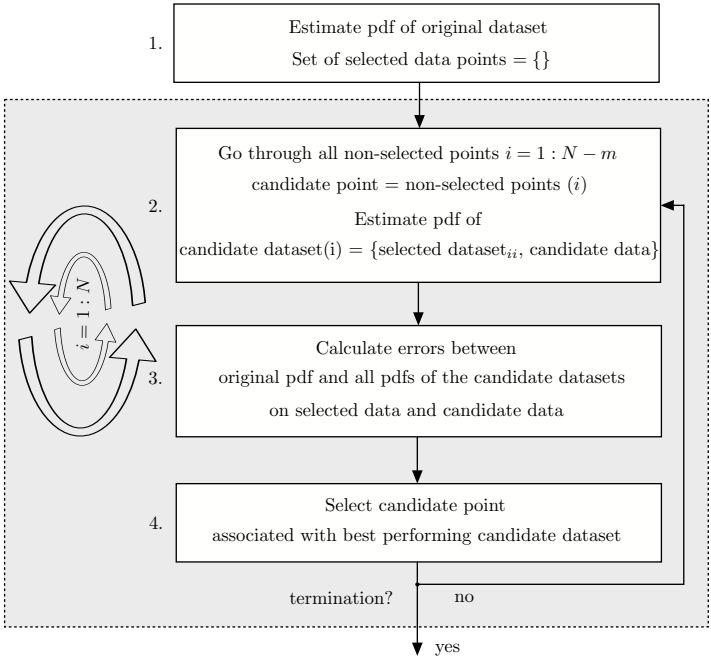


Figure 3: Procedure of the algorithm of method 1

After the initialization steps, the iterative process begins. Contrary to Sect. 3.1, an inner loop, which runs from $i = 1 : k$, is added to the algorithm. The loop activates the ith set of selected points, all remaining $k - 1$ sets stay passive. Now, the algorithm adds one point of the non-selected data points to the active subset of selected data points. Here, the data point which results in the minimal absolute error between the pdfs of the original dataset and the selected dataset, is added to the active subset. To do so, a pdf is estimated for each candidate data subset and the absolute error between this pdf and the pdf of the original dataset is evaluated. The absolute error is evaluated solely at the points of the selected subset. If the inner loop gets to $i = k$, it starts at $i = 1$ again. This procedure repeats, until the termination criterion is met.

Minor adjustments have to be made, if the subsets are unevenly sized, e.g. for an $80/20$ split. Here, after the smaller subset is filled up with the desired

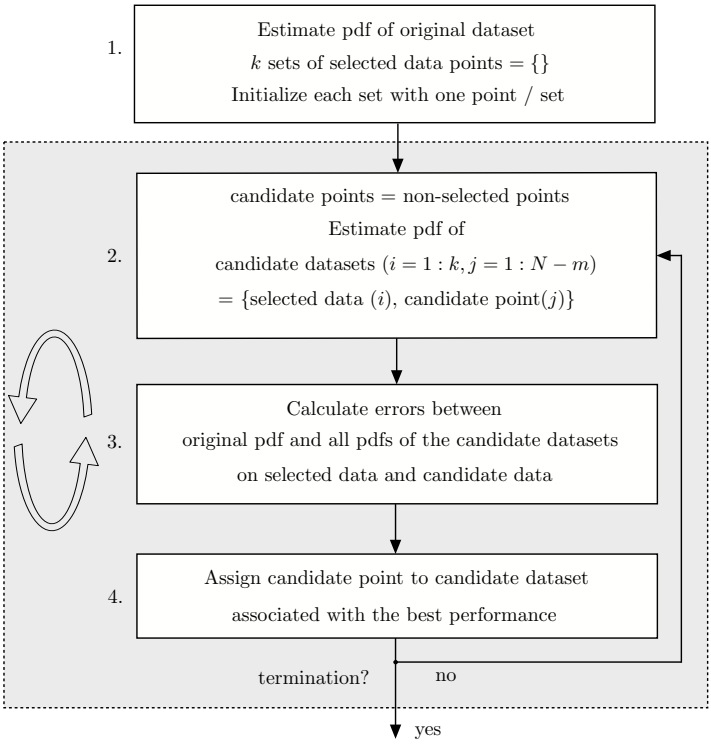


Figure 4: Procedure of the algorithm of method 2

number of data points, the bigger subset is considered in the data selection process solely.

3.3 Method 2: Point-based data splitting

The second data splitting concept uses a different approach to utilize the algorithm introduced in Sect. 3.1. The procedure of the method is depicted in Fig. 4.

At first, the pdf of the original dataset is estimated and the non-selected dataset and the selected data subsets are initialized. For each of the k subsets of selected data points, one initial data point has to be picked before starting an

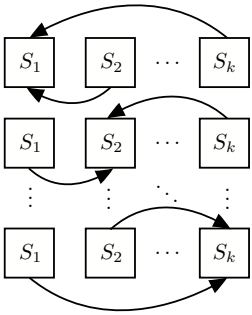


Figure 5: Algorithm scheme for a subsequent data point switching, in order to decrease deficits arising out of the use of forward selection schemes.

iterative process. The first points can be allocated randomly. Evaluations of more complex initialization methods did not show any impact on the performance of this method.

After the initialization steps, the selection process starts. In each iteration, $(N-m)k$ candidate datasets with $m = \#$ points selected are built. The candidate datasets consist of a combination between all points from one data subset (training or test) plus one point from the non-selected dataset. The absolute errors between the original pdf and all pdfs of the candidate datasets are calculated. The data point is allocated using a *probability* proportional to the size of the absolute errors, whereby an prioritization of one subset is prevented. The selection according to the probabilities gives this algorithm a stochastic flavor thereby avoiding getting stuck in bad local optima. The procedure runs until the termination criterion is met. In comparison to subset-based data splitting, this method possesses an even more flexible data allocation process.

3.4 Additional optimization of splitted data

Since both proposed methods, namely point-based and subset-based data splitting, use a forward selection approach to allocate data points, it can be expected that the final split will not correspond to the global minimum of the given loss function. Consequently, a subsequent exchange of points promises to yield

better loss function values. This section introduces a new algorithm to optimize the point distribution for an arbitrary number of subsets.

Figure 5 shows in principle how one step in the optimization is executed. $S_{1...k}$ represent the k selected subsets. The arrows indicate a point moving from one subset to another in a specified pattern. The first row shows the first of k sub-iterations of one iteration, one full iteration consists of k sub-iterations. In the first sub-iteration, subset S_1 is the active subset. By using the algorithm from Sect. 3.1, the active subset selects $k-1$ points, one point from each of the $k-1$ passive subsets. Thus, after one sub-iteration, the active subset possesses $k-1$ additional points, while the remaining $k-1$ passive subsets were reduced by one point. In the following, each subset is active for one time and passive for $k-1$ times. Therefore, after one full iteration, every subset has the same size as before.

In order to increase the potential of the optimization procedure, it is beneficial to add a stochastic element to it. Instead of selecting the best performing point in each iteration, the selection process is altered to choose from a list of the top performing points. Here, the probability to choose a point is proportional to its effect on the absolute error. This stochastic selection enables the procedure to escape local minima and achieve lower loss function values.

4 Evaluation of the approaches

4.1 Structure of the study

In order to confirm the function of the two new data splitting approaches, the Friedman regression function

$$y = 5(2sin(\pi x_1 x_2) + 4(x_3 - 0.5)^2 + 2x_4 + x_5) + \varepsilon \tag{1}$$

is used, similar to the evaluation in [4], to generate artificial data. Two different distributions for the inputs $x_i, i = 1, 2, ...5$, are used to produce two datasets. For dataset 1, the input is drawn from a mixture of two Gaussians, sampling 90 % of data from $x_i \sim \mathcal{N}(1, 0.6)$ and 10 % of data from $x_i \sim \mathcal{N}(-1, 0.6)$. For dataset

2, the input is generated by sampling from a single Gaussian distribution with $x_i \sim \mathcal{N}(0, 0.8)$. For both datasets, $\varepsilon \sim \mathcal{N}(0, 0.8)$ is used.

The LOLIMOT algorithm [10] is used to train models for each splitted dataset. 50 data splits for each of the two datasets are generated by method 1 and 2, each optimized and non-optimized, with a 50/50 splitting ratio. The performance, measured in terms of the root mean squared error (RMSE) on test data, is compared to the DUPLEX algorithm, in order to validate the performance of the new approaches. Additionally, the distribution of random splits with their corresponding mean performance is evaluated, according to [5], to compare the two criteria for data splitting quality.

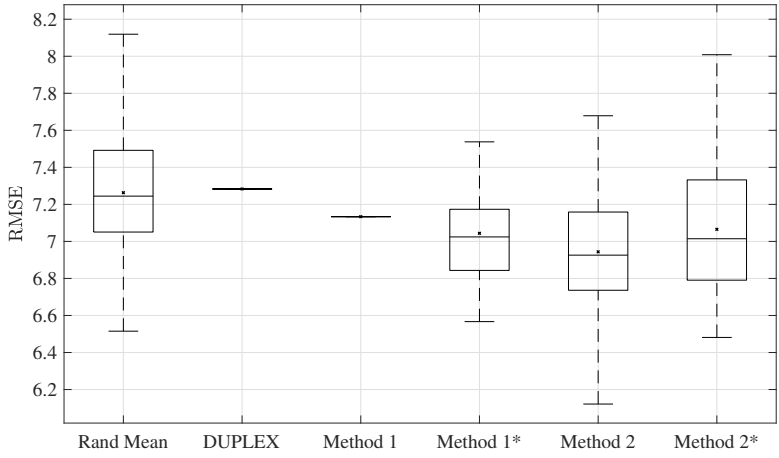


Figure 6: Dataset 1: Evaluation of 50 data splits, with * indicating subsequent splitting optimization

4.2 Performance analysis

Figure 6 shows the test error for 50 evaluations of the proposed splitting methods on dataset 1. The RMSE of DUPLEX and method 1 without optimization do not show variance over the 50 evaluations due to their deterministic algorithmic structure. While the DUPLEX algorithm shows higher RMSE values

compared to the average random split, method 1 achieves an improvement. Method 1*, where the * marks subsequent splitting optimization as described in Sect. 3.1, as well as method 2 and method 2* on average yield significantly lower RMSE values compared to random splitting. Since the DUPLEX algorithm bases on the simplistic approach of maximizing the distances of all input points to each other, an all in all better performance of the more complex pdf-estimation-based approaches could be expected. Looking at the variance, method 2 exhibits a broader spread in performance. This phenomenon can be traced back to the stochastic selection used in the procedure *and* in the optimization process.

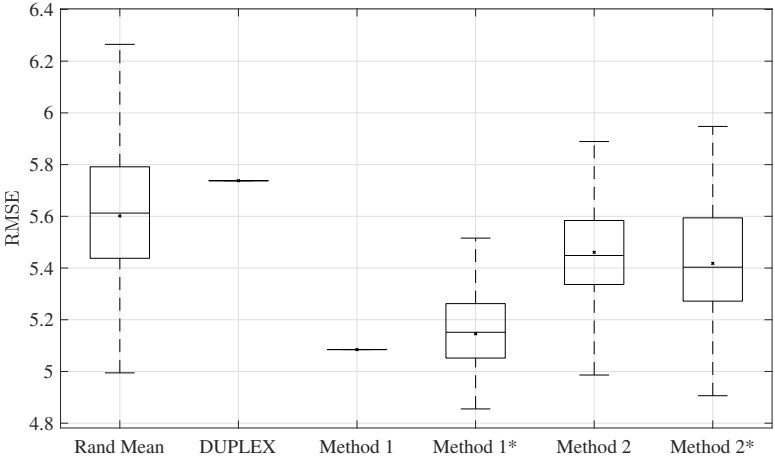


Figure 7: Dataset 2: Evaluation of 50 data splits, with * indicating subsequent splitting optimization

Figure 7 shows the evaluation of dataset 2. Here, method 1 outperforms DUPLEX by far. An explanation for this would be that method 1 handles equally distributed datasets better compared to DUPLEX. This explanation is due to the sampling of the inputs. The input of dataset 2 is produced sampling from one Gaussian instead of two Gaussians, compared to dataset 1.

The average RMSEs achieved by method 1*, method 2 and method 2* are lower than DUPLEX and the mean random split, but higher compared to method

1. Thus, the advantages of stochastic point selection and switching may have a bigger impact, if the underlying point distribution is more complex.

Overall, method 1 and 2 are comparable in mean and better than the average random split and DUPLEX. However, method 1 is preferable due to the lowest variance. The subsequent optimization partially yields better average results, but induces an additional variance error.

Since the rating of value and variance of test errors does not correlate necessarily with the quality of the point distribution, another measure to consult is the training error. Obviously, it is favorable if both training error and test error are low. Furthermore, if the test error is similar (only slightly worse) to the training error this indicates a low amount of overfitting and thus represents an appealing feature as well. The Figs. 8 and 9, corresponding to datasets one and two, show the distribution of training and test errors for 50 random splits, compared to method 1 and DUPLEX. The dashed line marks the average test error for random splitting. The bisecting line marks all points of equal training and test error. For both datasets, the points produced by method 1 are superior to the result of the DUPLEX algorithm and the mean random split performance. More specifically, they are (i) located closer to the line of equal training and test error and (ii) also resulting in lower test errors.

In this context, a straight-forward interpretation of this behavior is, that similarly distributed sets of training and test data should lead to similar performance in training and testing. Since the previously proposed evaluation criteria to compare data splitting methods do not take the quality of the data distribution into account explicitly, a more advanced analysis procedure would try to rate the data distribution of the splits, e.g. by using the test error as well as the training error for evaluation purposes.

5 Conclusion

In this contribution, two new data splitting approaches, based on the estimation and evaluation of pdfs, were introduced. Due to their structure, both approaches are capable of splitting datasets into subsets that approximate the input

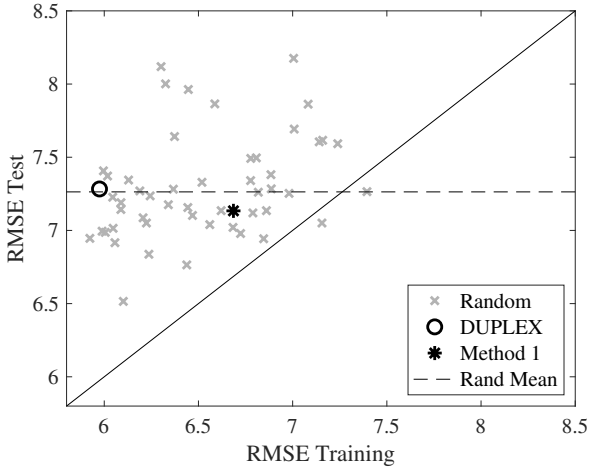


Figure 8: Dependency of training and test error, measured in terms of the RMSE, for dataset 1.

point distribution (pdf) of the original dataset. Method 1 fills the datasets alternatingly with data points. Method 2 allocates the points to the dataset with a probability proportional to their fit, to ensure that no subset is favored. Both approaches are based on an approximate, simplistic way of evaluating pdfs, the pdfs are estimated using kernel density estimation.

In order to validate the functionality of the approaches, they were used to split up two different datasets, that were subsequently used to train models. The evaluation of the RMSE showed an improvement when splitting data with the new approaches, compared to an existing method and the mean performance of random splitting. Method 1 showed the most promising performance.

The evaluation of existing procedures to rate the performance of data splitting methods showed potential to improvements. This will be subject to further research.

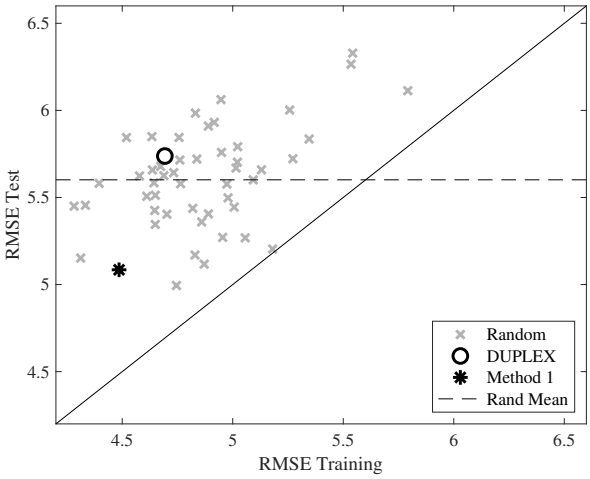


Figure 9: Dependency of training and test error, measured in terms of the RMSE, for dataset 2.

References

[1] M. A Shahin, H. Maier and M. Jaksa. “Data Division for Developing Neural Networks Applied to Geotechnical Engineering” In: *Journal of Computing in Civil Engineering* 18 2004

[2] R. D. Snee. “Validation of regression models: methods and examples” In: *Technometrics* 4.1. S. 415-428. 1977.

[3] C.W. Baxter, S.J. Stanley, Q. Zhang and D.W. Smith. “Developing artificial neural network process models: A guide for drinking water utilities”. In: Proceedings of the 6th Environmental Engineering Society Specialty Conference of the CSCE/, S.376-383. 200.

[4] R. May, H. Maier, and G. Dandy. “Data splitting for artificial neural networks using SOM-based stratified sampling”. In: *Neural networks : the official journal of the International Neural Network Society* 23. S. 283-294. 2009.

[5] W. Wu, R. May, H. Maier, and G. Dandy “A benchmarking approach for comparing data splitting methods for modeling water resources

parameters using artificial neural networks". In: *Water Resources Research* 11 S. 7598-7614. 2013.

[6] A.K. Sahoo, M.J. Zuo, and M.K. Tiwari. "A data clustering algorithm for stratified data partitioning in artificial neural network". In: *Expert Systems with Applications* 39.8. S. 7004-7014. 2012.

[7] G.J. Bowden, H. Maier and G. Dandy. "Optimal division of data for neural network models in water resources applications". In: *Water Resources Research* 38.2. 2002.

[8] B. LeBaron and A. Weigend. "A bootstrap evaluation of the effect of data splitting on financial time series". In: *IEEE Transactions on Neural Networks* 9.1. S. 213-220. 1998.

[9] T. J. Peter and O. Nelles "Fast and Simple Dataset Selection for Machine Learning". In: *at- Automatisierungstechnik*. 2019.

[10] O. Nelles. "Nonlinear System Identification". Berlin, Germany: Springer. 2001.

Algorithm Selection as Recommendation: From Collaborative Filtering to Dyad Ranking

Alexander Tornede, Marcel Wever, Eyke Hüllermeier

Heinz Nixdorf Institute and Department of Computer Science
Paderborn University
Warbuger Str. 100, 33100 Paderborn, Germany
E-Mail: {alexander.tornede,marcel.wever,eyke}@uni-paderborn.de

1 Introduction

Problem classes such as integer optimization, SAT, and classification can be tackled by a large variety of algorithms, the performance of which may differ depending on the concrete problem instance at hand. In fact, theoretical arguments even exclude the existence of a single algorithm that is superior to all other algorithms on all instances of a problem class [17]. Hence, compared to using the algorithm that is best on average across an entire class of problem instances, called the single best solver (SBS), selecting a suitable algorithm for each instance separately should result in an increased overall performance.

This expectation has been confirmed in recent algorithm selection (AS) competitions [1]. Algorithm selection seeks to support and automate the selection of an algorithm that is most suitable for a given problem instance. Meanwhile, quite a number of methods for AS has been proposed in the literature [8]. One interesting idea is to treat AS as a recommendation problem, and to apply techniques such as collaborative filtering [6]. Going beyond standard collaborative filtering, we propose to tackle AS as a problem of so-called dyad ranking [13]. This approach is motivated by at least two potential advantages:

- First, treating problem/algorithm pairs as dyads allows the learner to exploit properties (features) of both the problem instances and the candidate algorithms.

- Second, providing recommendations in the form of rankings of a set of candidate algorithms is presumably easier than evaluating each of them in terms of a precise numerical score.

These advantages are substantiated by first experimental studies in the field of automated machine learning, i.e., the recommendation of machine learning algorithms for model induction on a given dataset.

2 Algorithm Selection

In the setting of (per-instance) algorithm selection, we are given a set of problem instances $\mathscr{I}$, a set of algorithms $\mathscr{A}$, and a performance measure $m : \mathscr{I} \times \mathscr{A} \to \mathbb{R}$. The goal is to find an algorithm selector $s : \mathscr{I} \to \mathscr{A}$ such that, for a given instance $i \in \mathscr{I}$, the selector s chooses the algorithm with best performance according to measure m on instance i. Accordingly, the optimal selector, called oracle, is defined by

$$s^*(i) = \arg\max_{a \in \mathscr{A}} m(i,a) \tag{1}$$

for all $i \in \mathscr{I}$. For simplicity, we subsequently ignore any form of randomness imposed by an algorithm.

In practice, the performance measure m is usually costly to compute. Therefore, the obvious brute-force strategy of evaluating all algorithms for a given instance and returning the one performing best according to m is infeasible. Fortunately, we are usually provided with a subset $\mathscr{I}_D \subset \mathscr{I}$ of the instance space for which several of the algorithms have already been evaluated according to m. Invoking machine learning methods, this information can be used as training data to infer an algorithm selector $s : \mathscr{I} \to \mathscr{A}$ approximating the oracle (1).

Most state-of-the-art approaches to AS are complex systems that involve several steps, such as pre-solvers, portfolios, and other techniques, in addition to their core machine learning component [18, 5]. Here, we only focus on the latter, which is typically realized in the form of a regression model

$h : \mathscr{I} \times \mathscr{A} \to \mathbb{R}$. This model is supposed to track the performance of an algorithm for a given instance, and hence can be seen as a form of surrogate for m. Thus, an algorithm selector s can be constructed by returning the algorithm a that performs best on problem instance i according to h:

$$s(i) = \arg\max_{a \in \mathscr{A}} h(i,a) \tag{2}$$

Note that, in contrast to the performance measure m, the function h is usually cheap to evaluate. In contrast to the oracle (1), the computation of (2) is hence feasible. Nevertheless, one can also cast the learning problem in another form, for example as a recommendation problem in the context of collaborative filtering.

3 Algorithm Selection through Collaborative Filtering

Preference learning methods [3] for predicting rankings of algorithms for a given instance have recently gained attention. This is motivated by the observation that predictions of exact performances are sufficient but actually not necessary for choosing the best algorithm from a set of candidates. Hence, learning a regression model $h : \mathscr{I} \times \mathscr{A} \to \mathbb{R}$ appears to be an unnecessarily difficult problem.

In particular, methods related to collaborative filtering (CF) [6] have recently been studied [9, 10, 19, 2, 15, 4], although this idea was already introduced in [14] about a decade ago. In the standard CF setting, one is confronted with a set of products $\mathscr{P}$ and a set of users $\mathscr{U}$, and given a sparse $|\mathscr{U}| \times |\mathscr{P}|$ rating matrix R. The value contained in $R(u,p)$ is the rating of product p by user u. Common tasks associated with CF include *matrix completion*, which has the goal to infer the missing entries of the matrix R, and the *cold-start problem*, where an entire new row in the rating matrix R has to be predicted for a new user.

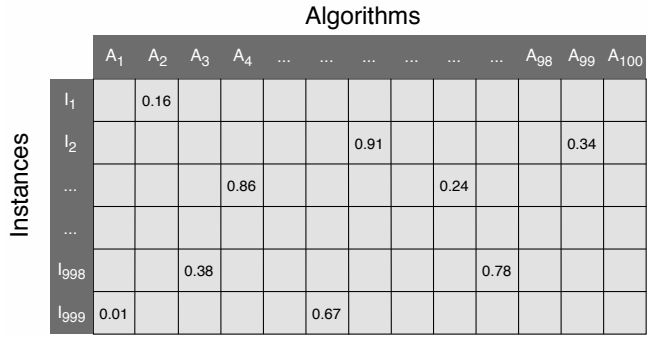


Figure 1: Depiction of a rating matrix filled with performance values from an algorithm selection dataset. Entries $R(i,a)$ contain the known performance of algorithm a on instance i and empty cells indicate unknown performances.

By treating problem instances as users, i.e. $\mathscr{U} = \mathscr{I}$, and products as algorithms, i.e. $\mathscr{P} = \mathscr{A}$, we can construct a rating matrix for algorithm selection in a very similar way, namely by filling the matrix with the evaluations of m available in the training data. An example of a corresponding rating matrix is depicted in Fig. 1.

This setting has two main disadvantages. Firstly, instead of incorporating expert knowledge about the algorithms in an explicit way, only latent characteristics (if at all) are induced (as done in [9]). Secondly, precise numerical information about the performance of algorithms is required. In practice, such information is often difficult to obtain, whereas weaker information in the form of qualitative comparisons between algorithms is more readily available. Imagine, for example, a scenario in which several algorithms are run in parallel until the first one found a solution. Then, if runtime is the performance measure to be optimized, precise numerical information is only generated for the first algorithm, while the knowledge that all other algorithms are worse is not directly used.

Moving from CF to dyad ranking [13] alleviates both of these disadvantages. Firstly, dyad ranking allows algorithm characteristics to be explicitly incorporated into the learning process. Secondly, dyad ranking merely requires qualitative training information in the form of rankings rather than precise numerical performances.

4 Algorithm Selection through Dyad Ranking

In addition to a feature representation for problem instances, we now also assume a feature representation for algorithms. By exploiting this information, there is hope to either speed up the model inference process or derive a more accuarate model. Moreover, instead of a real-valued rating matrix R, we assume a set of rankings over algorithms for the instances in the training set to be given. More precisely, we assume rankings over so-called *dyads*.

In (contextual) dyad ranking, a dyad $(\boldsymbol{x}, \boldsymbol{y})$ consists of a context $\boldsymbol{x} \in \mathscr{X} \subseteq \mathbb{R}^k$ from a context space $\mathscr{X}$ and an alternative $\boldsymbol{y} \in \mathscr{Y} \subseteq \mathbb{R}^r$. The training data we assume to be given is of the form

$$\mathscr{D} = \left\{ (\boldsymbol{x}_i, \boldsymbol{y}_{i,1}) \succ \ldots \succ (\boldsymbol{x}_i, \boldsymbol{y}_{i,l_i}) \right\}_{i=1}^{N} \subset \mathscr{R}(\mathscr{X} \times \mathscr{Y}) \tag{3}$$

and contains rankings with an underlying hidden preference relation $\succ$ over the space of dyads $\mathscr{X} \times \mathscr{Y}$, where l_i is the length of the ith ranking in $\mathscr{D}$, and $\mathscr{R}(\mathscr{X} \times \mathscr{Y})$ is the space of rankings over $\mathscr{X} \times \mathscr{Y}$. The goal is to learn a "dyad ranker"

$$h : \mathscr{P}(\mathscr{X} \times \mathscr{Y}) \to \mathscr{R}(\mathscr{X} \times \mathscr{Y}) \tag{4}$$

which, given an arbitrary set of dyads ($\mathscr{P}$ is the power set), ranks these dyads according to the hidden preference relation $\succ$.

To tackle the dyad ranking problem, we make use of the PLNet algorithm, a neural-network-based algorithm for learning a parametrized probability distribution over rankings, called the Plackett-Luce (PL) model [13].

A corresponding training dataset (3) is constructed by computing the feature representation of each algorithm and sorting the algorithms according to their performance in each row of the rating matrix $R(i, \cdot)$, pertaining to problem instance i. Additionally using the feature representation for instances, one can then extend the ranking to a ranking over dyads.

Problems to be considered during the construction include the sparseness of R as well as ties among algorithms for an instance in the rating matrix. The former can be solved by omitting algorithms with an unknown performance

from the associated ranking. The easiest solution to the latter problem is to treat ties of algorithms by not comparing them directly. This can be achieved by creating a ranking ignoring n tied algorithms, copying it n times and adding each ignored algorithm in one of these copies at the respective position.

As already mentioned, a feature representation is required for both problem instances and algorithms. In the literature, various ways of representing instances via features have been proposed, depending on the problem domain. In the AutoML setting considered in this work, the instances are machine learning datasets and associated feature representations are called meta-features [11]. An example of such meta-features are *landmarkers*, which are performance values of cheap-to-train algorithms on the respective dataset or a subset thereof. As shown in [12], landmarkers can be used successfully in the context of algorithm selection and can yield better results than statistical measures, such as the number of classes in a dataset. Accordingly, for the experiments in this work, we make use of landmarking features for representing datasets. More specifically, we use 45 OpenML landmarkers [16], which are computed based on learning algorithms such as Naive Bayes, One-Nearest Neighbour, Decision Stump, Random Tree, REPTree and J48.

Finding a feature representation for algorithms is more difficult, and the related literature is very sparse. We decided to represent algorithms via their parameters. Given a set of algorithms, we compute the union of their parameters and create a vector that has as many entries as the set of parameters. Then, when given a parametrized algorithm, we set the elements of the vector corresponding to its parameters to the respective values. Furthermore, for each component which can be contained in an algorithm, the vector contains a binary feature indicating whether the component is present or not. While this representation is simple, it has the disadvantage of not generalizing well across different algorithms that do not share any parameters, as they are essentially represented by disjoint subvectors of the original vector.

5 Experimental Results

We evaluated our approach in the AutoML setting, more specifically in the multi-class classification AutoML setting. Accordingly, instances correspond to multi-class classifcation datasets. Furthermore, the set of algorithms $\mathscr{A}$ we consider is a set of machine learning (classification) pipelines. By pipeline we mean the sequential combination of a data preprocessing step (such as a PCA) and a classification algorithm (such as an SVM). We considered 10 preprocessing steps and 7 classification algorithms resulting in a total of 70 classification pipelines. In addition, we considered up to 100 parametrizations for each of these pipelines and in total achieve an algorithm set with 5927 elements. We evaluated each of these parametrized pipelines on 29 classification datasets from OpenML[1]. Due to evaluation timeouts, only 89% of the theoretical amount of performance values is used.

Based on these performance values, we randomly sampled 10 train/test $(70\%/30\%)$ splits on the datasets (i.e., each split features 20 training datasets and 9 test datasets). For each of these splits, we created dyad ranking training datasets by randomly sampling rankings of pipelines of length two, i.e., pairwise comparisons under the condition that the two pipelines do not have the same performance on the respective dataset. In order to estimate how much information the learning algorithm (PLNet) requires to perform well, we evaluated different amounts of pairwise comparisons per dataset.

After training, we evaluated the approach by comparing the predicted ranking over all pipelines (for which we have a performance value) for each test dataset to the ground truth ranking obtained from the true performances using the Kendall's τ rank correlation measure [7], which takes values in $[-1,+1]$. We compared our approach against two instantiations of a nearest neighbor baseline, which, given a new dataset, computes the n closest training datasets according to the Euclidean distance, computes the average performance of all pipelines across these datasets and returns a ranking based on these averages.

[1] `https://www.openml.org/`

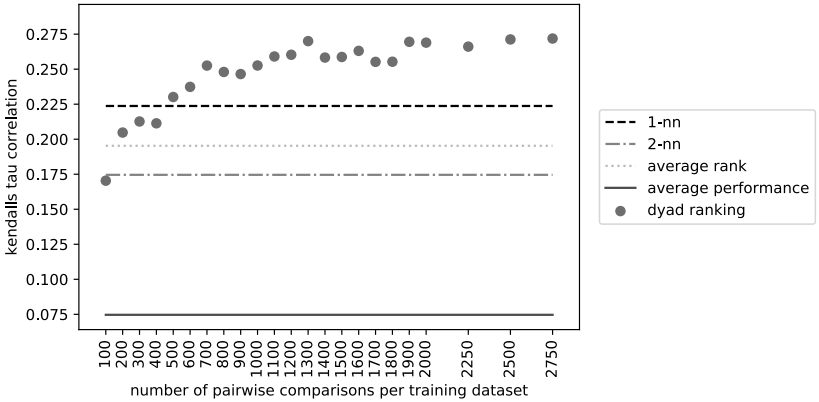


Figure 2: Kendall's τ rank correlation results for our dyad ranking approach based on different training dataset sizes and several baselines. All results are averaged across the test datsets and the train/test splits. On the x-axis, the number of pairwise rankings per training dataset used for training the associated dyad ranker is displayed whereas the y-axis shows the correlation measure value.

Furthermore, we compare against an average performance baseline, which simply returns a ranking based on the average performance of each pipeline across all training datasets, and an average rank baseline which does the same but averages the ranks instead of the performances. We averaged all results across the test datsets and the train/test splits we sampled.

Fig. 2 shows the value of the correlation measure as a function of the amount of training information (number of pairwise rankings per training dataset used for training the associated dyad ranker). As the baselines always consider all data available in the training datasets, their performance does not change with different amounts of rankings.

As expected, the performance of the dyad ranking approach increases with the amount of training data — quite strongly up to around 1300 rankings per training dataset and more slowly thereafter. More importantly, the approach surpasses all baselines with only 500 pairwise rankings per training dataset, which is a remarkable result as the training information used by the dyad ranker is only a tiny fraction of the information made available to the baselines.

Table 1: This table gives the difference between the best pipeline across all pipelines (in terms of accuracy) and the best one of the top-k pipelines returned by the different approaches.

Approach	k	Perf. Diff.	k	Perf. Diff.
DR	3	**0.032**	5	**0.028**
1-nn	3	0.045	5	0.045
2-nn	3	0.053	5	0.052
avg. rank	3	0.045	5	0.044
avg. perf.	3	0.046	5	0.046

Furthermore, since the version of AS we consider in this work is mainly concerned with returning the best pipeline for a new dataset, we compared our approach to the baselines by computing the difference between the best pipeline (in terms of accuracy) and the best one of the top-k pipelines returned by the different approaches. This evaluation gives an idea of how much worse it is to run the top-k pipelines returned by the ranking approach compared to running the best pipeline (according to the oracle) only. The results of the experiment are depicted in Table 1.

The dyad ranking approach (trained with 1400 pairwise comparisons per training dataset for this experiment) outperforms all other baselines by at least 1.3% percent points for $k = 3$ and 1.6% points for $k = 5$. Admittedly, even the baselines achieve a reasonable result in this experiment, as even a performance difference of about 5% to the oracle is still very good. Nevertheless, only the dyad ranking approach is able to achieve a considerably better result when increasing k, which makes us believe that it approximates the ground truth ranking better in the sense that it puts good pipelines in close proximity to their correct rank.

For full details regarding the experiments, we refer the interested reader to the github repository[2] containing all details and code required to reproduce the results presented here.

[2] `https://github.com/alexandertornede/ci_2019_as_via_dyad_ranking`

6 Conclusion and Future Work

We proposed to tackle the algorithm selection problem as a dyad ranking problem and addressed key questions regarding the creation of training datasets and feature representation for both algorithms and datasets.

Our first experimental studies show that dyad ranking outperforms the baselines we used for comparison. In future work, we plan to corroborate these preliminary results by more thorough evaluations of the approach in different scenarios as well as a comparison to state-of-the-art collaborative filtering methods.

Acknowledgement

This work was supported by the German Research Foundation (DFG) within the Collaborative Research Center "On-The-Fly Computing" (SFB 901/3 project no. 160364472).

The authors gratefully acknowledge the funding of this project by computing time provided by the Paderborn Center for Parallel Computing (PC^2).

References

[1] Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, et al. Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58, 2016.

[2] Tiago Cunha, Carlos Soares, and André C. P. L. F. de Carvalho. CF4CF: recommending collaborative filtering algorithms using collaborative filtering. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 357–361, 2018.

[3] Johannes Fürnkranz and Eyke Hüllermeier. *Preference learning*. Springer, 2010.

[4] Nicolo Fusi, Rishit Sheth, and Melih Elibol. Probabilistic matrix factorization for automated machine learning. In *Advances in Neural Information Processing Systems*, pages 3348–3357, 2018.

[5] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Torsten Schaub, Marius Thomas Schneider, and Stefan Ziller. A portfolio solver for answer set programming: Preliminary report. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 352–357. Springer, 2011.

[6] David Goldberg, David A. Nichols, Brian M. Oki, and Douglas B. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, 1992.

[7] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[8] Pascal Kerschke, Holger H Hoos, Frank Neumann, and Heike Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary computation*, 27(1):3–45, 2019.

[9] Yuri Malitsky and Barry O'Sullivan. Latent features for algorithm selection. In *Proceedings of the Seventh Annual Symposium on Combinatorial Search, SOCS 2014, Prague, Czech Republic, 15-17 August 2014.*, 2014.

[10] Mustafa Misir and Michèle Sebag. Alors: An algorithm recommender system. *Artif. Intell.*, 244:291–314, 2017.

[11] Phong Nguyen, Melanie Hilario, and Alexandros Kalousis. Using meta-mining to support data mining workflow planning and optimization. *Journal of Artificial Intelligence Research*, 51:605–644, 2014.

[12] Bernhard Pfahringer, Hilan Bensusan, and Christophe G Giraud-Carrier. Meta-learning by landmarking various learning algorithms. In *ICML*, pages 743–750, 2000.

[13] Dirk Schäfer and Eyke Hüllermeier. Dyad ranking using plackett-luce models based on joint feature representations. *Machine Learning*, 107(5):903–941, 2018.

[14] David H. Stern, Horst Samulowitz, Ralf Herbrich, Thore Graepel, Luca Pulina, and Armando Tacchella. Collaborative expert portfolio management. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010.

[15] Lisheng Sun-Hosoya, Isabelle Guyon, and Michèle Sebag. Activmetal: Algorithm recommendation with active meta learning. In *Proceedings of the Workshop on Interactive Adaptive Learning@ECML-PKDD 2018 Dublin, Ireland, September 10th, 2018.*, pages 48–59, 2018.

[16] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.

[17] David H Wolpert, William G Macready, et al. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[18] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *Journal of artificial intelligence research*, 32:565–606, 2008.

[19] Chengrun Yang, Yuji Akimoto, Dae Won Kim, and Madeleine Udell. OBOE: collaborative filtering for automl initialization. *CoRR*, abs/1808.03233, 2018.

Comparison of Extrapolation Behavior between Different State Space Models

Max Schüssler, Oliver Nelles

Universität Siegen, Department Maschinenbau,
Institut für Mechanik und Regelungstechnik – Mechatronik
Paul-Bonatz-Straße 9-11, 57076 Siegen, Germany
E-Mail: {max.schuessler, oliver.nelles}@uni-siegen.de

1 Introduction

System identification is the research field of modeling dynamic systems based solely on observed input and output data. This paper focuses on a crucial property in nonlinear system identification which is extrapolation. Extrapolation occurs, when the available training data does not cover the whole operating regime, in which the model operates during testing.

The question arises, which extrapolation behavior - constant, linear, polynomial, etc. - is the most desirable in black box modeling. It is obvious that there is no universally valid answer to this question, as the desired extrapolation behavior depends on specific properties of the system under test. Nevertheless, the question shall be answered, if there are differences in how reasonable a certain extrapolation behavior is in black box modeling. Therefore, the extrapolation properties of two nonlinear state space approaches, namely the *polynomial nonlinear state space model* (PNLSS) [7] and the *local model state space network* (LMSSN) [9], are analyzed.

In the case of the PNLSS, polynomials are used for the approximation of the nonlinear state and output equations of the state space model, leading consequently to **polynomial extrapolation behavior**. If the degree of the polynomials exceeds a reasonably small number, the model's output tends

to be erratic near the interpolation boundaries and approaches $\pm\infty$ extremely fast, often yielding unstable models. Extrapolation can be, therefore, quite dangerous for polynomial models [3].

In LMSSNs, affine functions approximate different partitions in the input space, leading therefore to **linear extrapolation behavior**. This property seems to be more reasonable for nonlinear dynamic models and suggests that the LMSSN is in extrapolation "well behaved" in contrast to the PNLSS. To evaluate, whether this assumption holds to be true, a systematic analysis of the two approaches will be carried out on a synthetic test process.

2 Different State Space Model Architectures

A deterministic nonlinear time-discrete state space model is described by

$$\underline{\hat{x}}(k+1) = \underline{h}(u(k), \underline{\hat{x}}(k)) \tag{1a}$$

$$\hat{y}(k) = g(u(k), \underline{\hat{x}}(k)), \tag{1b}$$

with the state vector $\underline{\hat{x}}(k)$, the input $u(k)$, the output $\hat{y}(k)$, the state equations $\underline{h}(\cdot)$ and the output equation $g(\cdot)$ at the time step k.

2.1 Local Model State Space Network

The LMSSN [9] utilizes *local model networks* (LMNs) for the approximation of the state and output equation of a nonlinear state space model. The output $\hat{y}$ of an LMN is calculated by

$$\hat{y} = \sum_{j=1}^{n_m} \underbrace{L_j\left(\tilde{\underline{u}}, \underline{\theta}_j\right)}_{\text{Local Model}} \overbrace{\Phi_j\left(\tilde{\underline{u}}, \underline{c}_j, \underline{\sigma}_j\right)}^{\text{Validity Function}}, \tag{2}$$

where n_m is the number of *local models* (LMs), L_j denotes an affine LM, and Φ_j the validity or activation function which is chosen as a normalized *radial*

basis function (RBF). The arguments of L_j are: $\underline{\tilde{u}}$ representing the extended input vector[1] and $\underline{\theta}_j$ representing the parameters of the affine LMs. The validity functions have as arguments also the extended input vector $\underline{\tilde{u}}$ and the centers $\underline{c}_j$ and standard deviations $\underline{\sigma}_j$ of the RBFs. The identification algorithm of the LMSSN is based on the *local linear model tree* (LOLIMOT) algorithm [4] and on the *best linear approximation* (BLA) [8], ensuring that all estimated nonlinear models are at least as good as the best linear model. For a detailed account on the LMSSN, refer to [9].

2.2 Polynomial Nonlinear State Space Models

The PNLSS was developed by Paduart [6]. It extends a linear state space model by two additional terms which include higher order polynomials to the state and output equation.

For SISO systems, the PNLSS model can be written as

$$\underline{\hat{x}}(k+1) = \underline{A}\,\underline{\hat{x}}(k) + \underline{b}\,u(k) + \underline{E}\,\underline{\zeta}(\underline{\hat{x}}(k), u(k)) \tag{3a}$$

$$\hat{y}(k) = \underline{c}^T\,\underline{\hat{x}}(k) + d\,u(k) + \underline{f}^T\,\underline{\eta}(\underline{\hat{x}}(k), u(k))\,. \tag{3b}$$

The vectors $\underline{\zeta}(\underline{\hat{x}}(k), u(k))$ and $\underline{\eta}(\underline{\hat{x}}(k), u(k))$ contain nonlinear monomials in $\underline{\hat{x}}(k)$ and $u(k)$ of degree two and up to a chosen degree p. The coefficients associated with these nonlinear terms are given by matrix $\underline{E}$ and vector $\underline{f}^T$. Note that the monomials of degree one are included in the linear part of the PNLSS model [7]. The full identification procedure can be found in [6, 7].

Complex PNLSS models do easily become unstable in extrapolation, as a study on the PNLSS for hysteresis identification points out [5]. For a further developed PNLSS model, the decoupled PNLSS, the authors point out that the amplitudes of the input signal for testing had to be smaller than for training as otherwise extrapolation problems occurred [1, 2]. This behavior constitutes a significant limitation to the usefulness of those methods.

[1] $\underline{\tilde{u}}$ denotes the input vector for the LMNs, i.e. $\underline{\tilde{u}} = [u\ \underline{\hat{x}}]$ to distinguish it from the dynamic model and process input u.

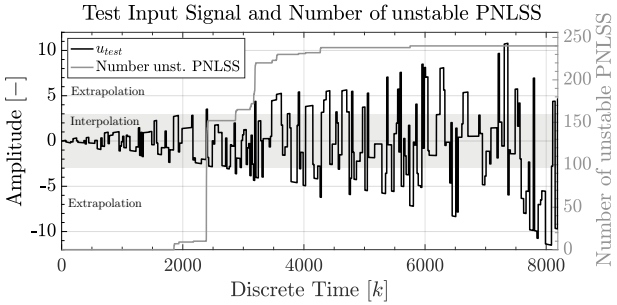


Figure 1: APRBS test signal with increasing maximum amplitude. For amplitudes with absolute value greater than 3 extrapolation occurs.

3 Extrapolation Study on Artificial Test System

The extrapolation behavior of the LMSSN and PNLSS model will be investigated on an artificial second order Wiener test system with an arc tangent output equation. The models will be trained with $N = 2048$ data samples. The model orders are chosen according to the order of the process. For the monomials in state and output equation of the PNLSS all different combinations with a degree up to 5 are evaluated. In total, 256 PNLSS models are evaluated. For example, a model which includes monomial degrees 2 and 3 in the state equation, and no polynomial extension in the output equation will be denoted by $\mathscr{O}(\underline{\zeta}) = [2\ 3]$[2] and $\mathscr{O}(\underline{\eta}) = [\,]$. Note that degree one is always included in the linear model. The training input signal is an *amplitude modulated pseudo random binary signal* (APRBS) which lies in the interval [-3, 3].

What happens, if the maximum amplitude of an APRBS test signal is increased over time from $A_{max} = 0$ (at $N = 0$) to $A_{max} = 12$ (at $N = 8192$)? The test input signal can be seen in Fig. 1. One can see that at $N = 2420$ the amplitude of the input signal is for the first time larger than the training amplitudes with $A_{max} \approx 3.5$. 152 out of the 256 PNLSS models produce from this point on unstable results. 68 other models produce unstable results from $N = 3210$ on where another maximum amplitude is reached of $A_{max} \approx 4.3$. Only 16 out of

[2]For a process with one input u and one state $\hat{x}$, this would mean that $\underline{\zeta} = [\hat{x}^2\ \hat{x}u\ u^2\ \hat{x}^3\ \hat{x}^2u\ u^2\hat{x}\ u^3]$. The argument k is left out for brevity.

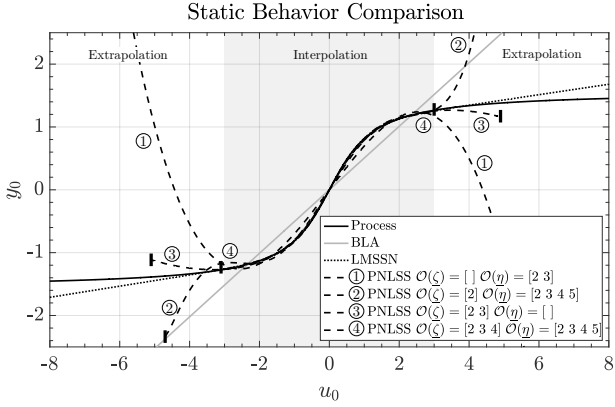


Figure 2: Comparison of static behavior of the Wiener process, BLA, LMSSN, and PNLSS models.

the 256 different PNLSS models do not become unstable on this test signal. Those are exclusively models where $\mathcal{O}(\underline{\zeta}) = [\ ]$ and, therefore, models with linear state equations. Those models nevertheless produce highly inaccurate results in extrapolation, but are robust against outliers. All models with polynomial state equations are highly susceptible to outliers, which is a severe limitation for practical applications of the PNLSS.

The steady state curve of the process, BLA, LMSSN, and a selection of PNLSS models is shown in Fig. 2. It can be seen that all models match the static process characteristics in the interpolation region quite well. It can also be seen that the BLA is a reasonable model for the underlying nonlinear process. The weakness of the PNLSS models (dashed lines) becomes obvious in extrapolation. One can clearly identify the polynomial behavior and for this process quite undesirable extrapolation behavior. The small bars at the end of some of the PNLSS models indicate the points, from where on no stable results could be obtained. For this Wiener process with an arc tangent output equation, the LMSSN has the most desirable extrapolation behavior. The error gets larger the further the input amplitude is away from the extrapolation boundaries, but the general characteristics are captures best.

4 Conclusion and Future Work

The extrapolation behavior of two different nonlinear state space models are compared. On an artificial Wiener test system it is shown that the *local model state space network* (LMSSN) performs superior in comparison to the *polynomial nonlinear state space model* (PNLSS). Superior means in this case that the LMSSN does not become unstable in extrapolation and the general behavior is "well behaved" and reasonable. In contrast, many PNLSS trained models easily become unstable (in 240 of 256 cases of varying polynomial degrees) if close to the interpolation boundaries or in extrapolation. This results shows, that the practical usefulness of the PNLSS is severely limited, while the LMSSN shows favorable properties for real-life application.

References

[1] A. Fakhrizadeh Esfahani, P. Dreesen, K. Tiels, J.-P. Noël, and J. Schoukens. “Polynomial state-space model decoupling for the identification of hysteretic systems”. *IFAC-PapersOnLine*, 50(1):458–463. 2017.

[2] A. Fakhrizadeh Esfahani, P. Dreesen, K. Tiels, J.-P. Noël, and J. Schoukens. “Parameter reduction in nonlinear state-space identification of hysteresis”. *Mechanical Systems and Signal Processing*, 104:884–895. 2018.

[3] T. Hastie, R. Tibshirani, and J. H. Friedman. “The elements of statistical learning”. Springer series in statistics. Springer, New York, NY, 2. ed. edition. 2009.

[4] O. Nelles and R. Isermann. “Basis function networks for interpolation of local linear models”. In *35th IEEE Conference on Decision and Control*, pages 470–475. 1996.

[5] J.-P. Noël, A. F. Esfahani, G. Kerschen, and J. Schoukens. “A nonlinear state-space approach to hysteresis identification”. *Mechanical Systems and Signal Processing*, 84:171–184. 2017.

[6] J. Paduart. " Identification of nonlinear systems using Polynomial Nonlinear State Space models". PhD thesis, Vrije Universiteit, Brussel. 2008.

[7] J. Paduart, L. Lauwers, J. Swevers, K. Smolders, J. Schoukens, and R. Pintelon. "Identification of nonlinear systems using polynomial nonlinear state space models". *Automatica*, 46(4):647–656. 2010.

[8] R. Pintelon and J. Schoukens. "*System identification: a frequency domain approach*". John Wiley & Sons. 2012.

[9] M. Schüssler, T. Münker, and O. Nelles. "Local model networks for the identification of nonlinear state space models". In *58th IEEE Conference on Decision and Control (CDC)*, Nice, France. 2019 (accepted for publication).

Simulating Automotive Radar Sensors Using Lidar and Camera Data

Matthias Greshake, Jonas Schneider

Elektronische Fahrwerksysteme GmbH
Dr.-Ludwig-Kraus-Str. 6, 85080 Gaimersheim
E-Mail: matthias.greshake@efs-auto.de, jonas.schneider@efs-auto.de

1 Introduction

Precise and reliable sensors are essential for the perception in automated driving. For example, driver assistance systems such as adaptive cruise control (ACC) require accurate measurement data to control the velocity and the distance to preceding vehicles. For this reason, automated vehicles are equipped with various high-end sensors that provide a tremendous amount of internal and environmental information. Multiple sensor technologies are combined to handle a wide range of different light and weather conditions. For critical areas, for example in front of the vehicle, the field of view of various sensors deliberately overlaps to exploit the advantages of the different technologies. This leads to redundancies within the received data. However, these redundancies are important to make the system robust against errors and failures of individual sensors.

Nowadays, the environmental sensor setup of an automated vehicle comprises radar (radio detection and ranging), lidar (light detection and ranging) and camera systems. While lidars and cameras operate in the infrared or visible light spectrum, automotive radars employ micro waves in the frequency range between 24 and 77 GHz. They share some characteristics with lidars. Both sensors determine the distance to an obstacle by measuring the time difference between emitting electromagnetic signals and receiving the reflected echoes. Subsequently, the position of the detected object can be calculated from distance and azimuth. In contrast to lidars, the radar additionally obtains the

relative velocity of dynamic objects even under bad weather conditions such as rain or fog. However, radars are much more susceptible to false detections than lidar or camera systems. These are caused by side effects such as multipath propagation, interference or attenuation, which frequently occur in radar scans. [1]

For virtual development and testing accurate models for each sensor are necessary [2]. Unfortunately, the simulation of a physical radar model is a computationally demanding task. Modern GPU technology allows a significantly faster computation, but is still limited to simplified models of the sensor and its environment [3]. Furthermore the modeling process requires detailed knowledge of the sensors. An alternative approach is to exploit the redundancy within the sensor data by statistical learning procedures. These methods learn the internal mechanics of the radar from data instead of manually creating an exact physical model.

For this reason, a statistical model is proposed that simulates an automotive radar sensor using the correlation between radar, lidar and camera data. The advantage of this approach is that the computational effort arises mainly during the training process. Since the training is completed before the application phase, the model only has to be evaluated at runtime. Another advantage is that no further effort is required for engineering a sensor model, as the parameters are learned from data. However, data-driven methods with respect to automotive sensors have not yet been comprehensively explored.

2 Related Work

In recent years, plenty of methods for sensor modeling have been published. They can be roughly classified into model-driven and data-driven approaches. While model-driven approaches design the model from an engineering perspective to physically describe the behavior of the sensor, data-driven approaches statistically estimate the model parameters based on the measurement data.

The use of model-driven approaches requires a detailed model of the sensor and its environment. This model consists of various mathematical equations describing the physical processes within the sensor. The performance depends strongly on the accuracy of these equations. However, the simulation of such highly accurate models is computationally extensive and demands a profound knowledge of the functionality of the sensor to be modeled. For this reason, this paper will focus exclusively on data-driven methods.

A simple approach for statistical model estimation are Gaussian mixture models (GMM). Due to the assumption that most of the real-world data cannot be sufficiently well approximated with a single multivariate normal distribution, a mixture of multiple distributions is required. However, the finite number of mixtures that has to be defined before initialization is a major drawback as it does not necessarily represent the actual number of distributions. Therefore, and due to the limited modeling capacities, GMMs are mainly deployed for simple models or as a benchmark for other algorithms [2][4].

Variational Bayesian Gaussian mixtures (VBGM) counter this problem by incorporating information from prior distributions. Here, the parameters of the mixtures themselves are random variables drawn from the prior distribution to receive a theoretically infinite number of mixture components. Lundgren et al. applied a variational variant of the expectation-maximization (EM) algorithm for estimating radar maps [4]. Their approach outperforms the standard EM algorithm in a simulation scenario as well as with real radar data even in dense clutter. Scheel and Dietmayer simulated radar density maps with variational Gaussian mixtures for tracking subjects [5]. They trained their model on real-world data and achieved better results than the examined model-driven approaches with respect to the tracking ability.

Before the appearance of deep learning architectures, kernel density estimation (KDE) was state of the art for model estimation and data generation. Hirsenkorn et al. applied Gaussian kernels for simulating radar sensors due to simple computation [6]. Their model is able to simulate inherent properties such as occlusion, latency and ghost objects. They tested their approach with real data from an automotive radar system in [7] to demonstrate the real-time capability and extended it to a generic sensor system in [8].

In the past few years, deep neural networks have become a popular method for generative modeling, as they are able to describe models of high complexity. Wheeler et al. trained a conditional variational autoencoder (CVAE) for simulating radar power fields with real-world data [2]. They fed a spatial raster of distance and azimuth as well as a radar object list into the network and compared the results with a GMM. Their approach is able to model realistic radar data including fundamental effects such as clutter while outperforming the GMM. In addition, they improved the results by combining the autoencoder loss with the loss of a generative adversarial network (GAN). Suhre and Malik developed a similar approach that generates radar object lists with a CVAE using measurements from a real-world dataset [9]. They mapped data measured by a differential global positioning system (DGPS) to the output of the real radar sensor. The results demonstrate that their model can predict radar object lists with high accuracy.

The presented approaches above are designed for processing data without any temporal behavior. The assumption is often made as the consideration of temporal dependencies drastically increases the computation time. Hidden Markov models (HMM) are a method that is able to operate on sequential data. Zec et al. proposed an autoregressive input-output HMM for modeling dynamic behavior of a lidar sensor [10]. Their approach can also be transferred to other sensor systems.

Most of these methods have been validated either virtual [4] or in specific artificial environments that have been explicitly adapted for these scenarios [2][4][9]. Our approach is evaluated on public roads with real traffic. This has the advantage that the model is trained under the same conditions as it will be applied later.

3 Statistical Radar Model

The process from the data acquisition to the deployment of the trained model can be separated into different tasks. Figure 1 illustrates the whole procedure. First, the data are acquired by test drives on public roads. Subsequently, the data have to be prepared for the training. The preprocessing comprises the

transformation into a unified sensor coordinate system (USC), the data synchronization including the compensation of dynamics and the projection into a structured representation. The training takes multiple iterations until the loss function of the model converges. Once the training is completed, the model can be deployed to generate radar scans solely from the input data of lidar and camera. The radar sensor model is represented by a statistical model. The training of the model parameters is done before the application. During training, the model is provided with samples of radar, lidar and camera. In the application phase, only the input data of lidar and camera are available. Each sample within the dataset consists of an image frame of a high-definition video camera, a lidar point cloud and a radar object list. Object lists are high-level representations of sensor data including the detected objects. The objects can hold different features such as position, relative velocity or radar cross section in the case of a radar sensor.

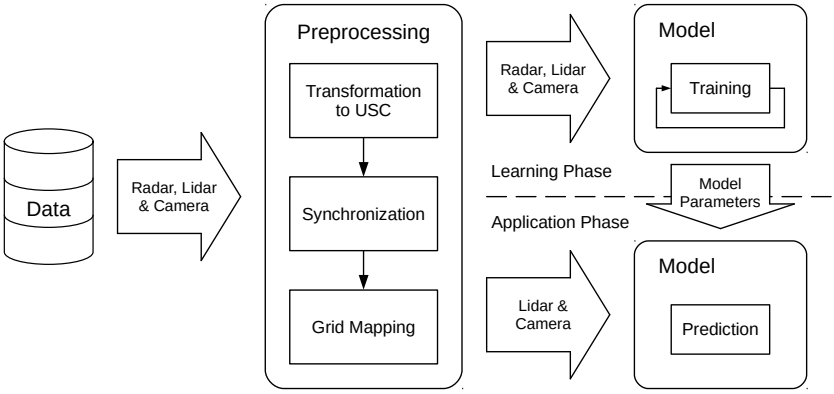


Figure 1: Data flow within the learning and application phase.

For a first proof of concept, only data from highway scenarios are considered due to their restricted dynamics and uncertainty compared with urban environments. Moreover, all scenes are recorded under clear light and weather conditions. Currently, the model exclusively employs dynamic objects, as data analysis has revealed that the correlation between lidar and radar data is insufficient for static objects.

In order to fit the requirements of the training process, the raw data have to pass various preprocessing steps (cf. Fig. 1). First, they are converted into the USC, whose origin is located on the rear axle of the ego vehicle. The unification facilitates the training as the model does not have to learn the transformation between the individual sensor coordinate systems.

After the geometric unification, a temporal synchronization is necessary due to the high dynamics in the automated driving process. Each sensor has a different latency between receiving input signals and sending them to the control unit. This latency has to be corrected to ensure correlation within the data. In addition, the movement of the ego vehicle and the other dynamic objects in the environment affect the synchronization. Consequently, they have to be compensated by estimating their velocity and then calculating and correcting the distance between their actual and perceived positions.

Another problem is the order of the individual objects in the object lists. The position depends on the occurrence of the object within the scan. Unfortunately, neural networks are not invariant to the order of the inputs and outputs. For this reason, a structured representation for lidar and radar data is necessary. In our approach, the lidar and radar data are projected onto a grid map. A grid map is a two-dimensional representation that discretizes the geometric space in grid cells. There exist classical occupancy grid maps where each grid cell $c \in \{0,1\}$ is either occupied or not, and soft occupancy maps where the presence of an object in each cell $c \in [0,1]$ is indicated by a probability value [11]. In contrast to the radar object list, the lidar data are measured as a binary point cloud. Consequently, the lidar point cloud is represented as a classical occupancy maps, while the radar objects are projected onto a soft occupancy map.

4 Variational Autoencoder

The sensor model is represented by a VAE [12]. The VAE is a directed graphical model consisting of a recognition part (encoder) and a generative part (decoder). In order to generate a data sample x, the VAE first draws a sample z from the latent probability distribution $p(z)$. Subsequently, z is decoded by

the generative model G so that $x = G(z)$. In other words, x is sampled from $p_\theta(x; G(z)) = p_\theta(x|z)$, where θ is a set of trainable parameters approximating p. The latent probability distribution $p_\theta(z)$ is inferred by the conditional probability $p_\theta(z|x)$ defined in equation (1).

$$p_\theta(z|x) = \frac{p_\theta(x|z)\, p_\theta(z)}{p_\theta(x)} \tag{1}$$

Unfortunately, $p_\theta(x) = \int p_\theta(x|z)\, p_\theta(z)\, dz$ is intractable. For this reason, $p_\theta(z|x)$ is approximated by another distribution $q_\phi(z|x)$, where ϕ are the trainable parameters of the recognition model. This method is called variational inference. To ensure that $q_\phi(z|x)$ is similar to $p_\theta(z|x)$, the Kullback-Leibler (KL) divergence between both distributions defined in equation (2) is minimized.

$$D_{KL}[q_\phi(z|x) \parallel p_\theta(z|x)] = \sum_z q_\phi(z|x) \log \frac{q_\phi(z|x)}{p_\theta(z|x)} \tag{2}$$

The KL divergence measures the dissimilarity between two probability distributions. By rearranging the definition, the variational lower bound $\mathscr{L}$ is obtained in equation (3).

$$\begin{aligned} \mathscr{L} &= \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}[q_\phi(z|x) \parallel p_\theta(z)] \\ &\leq \log p_\theta(x) \end{aligned} \tag{3}$$

$\mathscr{L}$ is also called evidence lower bound (ELBO). The log likelihood calculates the reconstruction error between the generated and the original input data, while the KL divergence acts as a regularization term. Since both the recognition and the generative model are differentiable, the ELBO can be optimized by any gradient descent method after applying the reparameterization trick described in [12].

A common problem with ELBO is the imbalance between reconstruction error and regularization. For this reason, Higgins et al. extended the ELBO by an additional hyper-parameter [13]. Equation (4) demonstrates the extended lower bound. The β weights the capacity of the latent space depending on the reconstruction accuracy.

$$\mathscr{L} = \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - \beta\, D_{KL}[q_\phi(z|x) \parallel p_\theta(z)] \tag{4}$$

Another problem is that ELBO is often too restrictive and tends to overfit the data. Zhao et al. countered this problem with a variation called InfoVAE [14]. Similar to the β-VAE, it weights the components of the target function and adds mutual information between x and z. Equation (5) illustrates the updated lower bound. Here, λ serves as a scaling parameter to counteract the dimensional imbalance between input space $\mathscr{X}$ and latent space $\mathscr{Z}$, while α weights the influence between regularization and mutual information.

$$\begin{aligned}\mathscr{L} = \mathbb{E}_{z\sim q_\phi(z|x)}&[\log p_\theta(x|z)] \\ &- (1-\alpha)\, D_{KL}[q_\phi(z|x) \parallel p_\theta(z)] \\ &- (\alpha+\lambda-1)\, D_{KL}[q_\phi(z) \parallel p_\theta(z)]\end{aligned} \tag{5}$$

Furthermore, $D_{KL}[q_\phi(z) \parallel p(z)]$ can be replaced by any other divergence that is differentiable. Zhao et al. recommend to apply maximum mean discrepancy (MMD) over ELBO. MMD measures the similarity of two probability distributions by comparing all of their moments. It can be efficiently implemented by equation (6) using any positive definite kernel $k(\cdot,\cdot)$ such as the Gaussian kernel.

$$\begin{aligned}D_{MMD}[q_\phi(z) \parallel p_\theta(z)] = \mathbb{E}_{q(z),q(z')}&[k(z,z')] \\ &- 2\cdot\mathbb{E}_{q(z),p(z')}[k(z,z')] \\ &+ \mathbb{E}_{p(z),p(z')}[k(z,z')]\end{aligned} \tag{6}$$

The major drawback of VAEs is that the output tends to get blurred [15]. One possible explanation for this phenomenon could be an intrinsic effect of the maximum likelihood estimation. This means that features in the input data, which might be important but occupy only a small area in the input space, are ignored. Previous works have figured out that providing additional information such as class labels mitigates the issue of blurry outputs [16]. The CVAE extends the prior and posterior probability by a class label c as illustrated in equation (7).

$$\mathscr{L} = \mathbb{E}_{z\sim q_\phi(z|x,c)}[\log p_\theta(x|z,c)] - D_{KL}[q_\phi(z|x,c) \parallel p(z|c)] \tag{7}$$

Another extension that reduces blurriness is adversarial learning. Here, an adversarial loss is optimized rather than the maximum likelihood. The original algorithm by Goodfellow [17] samples z from a standard normal distribution and generates a data sample x by passing z through a generator network G so that $x = G(z)$. Subsequently the generated data sample x is processed by a discriminator network D, which outputs a probability y whether x is either generated or part of the training set. The whole procedure can be considered as a zero-sum game where the discriminator attempts to maximize the cost function $v(G,D)$ and the generator attempts to minimize it. The complete target function of the so-called GAN is demonstrated in equation (8).

$$\mathscr{L} = \min_G \max_D \left[\mathbb{E}_{x\sim p(x)}[\log D(x)] - \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))]\right] \tag{8}$$

The optimal point of the loss function is the Nash equilibrium, where G produces the real data and D always returns the probability 0.5, meaning that it is no longer able to distinguish between real and generated data.

Unfortunately, $\max_D v(G,D)$ is not convex. This makes the training of GANs difficult, as the network often gets stuck in local minima or underfit the data. Mode collapse is another issue that frequently occurs during training. If the discriminator is too strong, it perfectly distinguishes between real and generated samples. In this case, the generator has no chance to minimize its target function and maps all z to a single representation.

In order to train a CVAE with an adversarial loss, the generator network of the GAN is replaced by the decoder of the CVAE. For the application, encoder and discriminator are pruned and only the conditioned decoder is evaluated to obtain the output.

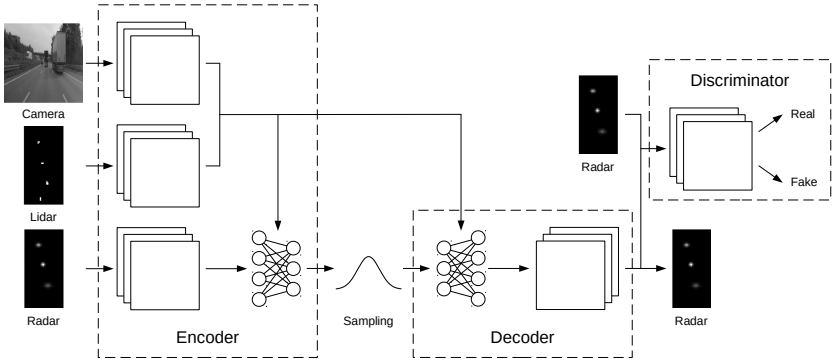


Figure 2: Network architecture including encoder, decoder and discriminator.

5 Network Architecture

For a better understanding, the objective is decomposed into several subproblems. Starting from the reconstruction of the radar input, the problem is expanded step by step to the prediction of radar grid maps by lidar and then camera data. The network architecture consists of three consecutive modules. Figure 2 illustrates the individual parts of the network. In the first step, the radar grid maps are reconstructed by a standard VAE. Subsequently, the VAE is conditioned by lidar and camera data to predict the associated radar input. Finally, the structure can be extended by a GAN to train the network with an adversarial loss. The network hyper-parameters are based on some empirical research and our experience with generative models. The VAE includes an encoder and a decoder part. The encoder consists of six convolutional layers and two fully-connected layers estimating means and variances of the Gaussian distributions spanning the latent space. The latent space is sampled by a 32-dimensional standard normal distribution. The decoder comprises a fully-connected layer and three transposed convolutional layers to expand the latent representation to the size of the original input. Both convolutional and transposed convolutional layers achieve dimension reduction and expansion using a stride of two.

The CVAE requires labels to control the data generation. For this purpose, lidar and camera data are encoded into a smaller representation of important

features and integrated into both parts: encoder and decoder. The encoding of the lidar grid maps is similar to that of the radar maps. They are processed by five convolutional layers. For image encoding, a pre-trained network such as VGG16 [18] can be employed. The use of a pre-trained network for image processing is recommended, as the duration of training and the amount of required data are significantly reduced.

All layers, apart from the last layers of the encoder and decoder includes rectified linear units (ReLU) as activation function. ReLU activations are preferred over sigmoid functions as they mitigate the vanishing gradient problem in deep networks [19]. In addition, each layer implements batch normalization before the activation. This addition normalizes the input batch of the activation function by mean and variance of the batch. It reduces the effect of extremely different input scales and prevents the gradients from oscillating too much, resulting in faster convergence [20]. The output layer of the decoder omits batch normalization and apply a sigmoid activation function to normalize the output data to the range $[0, 1]$. The fully-connected layers in the encoder feature linear activation functions. All convolutional and transposed convolutional layers have quadratic kernels of size 3×3.

In the next step, the adversary approach can be implemented by setting a discriminator network of multiple convolutional layers on top of the CVAE. While the CVAE generates the output data, the discriminator determines the adversarial loss by distinguishing between the generated output and the real radar grid maps from the dataset.

6 Experiments

In order to examine whether a complete radar scan is artificially reproducible using sensor data correlation, various experiments are performed. In the following, first, the radar input is reconstructed and then the lidar data are integrated to predict radar grid maps. In future steps, the influence of camera images and adversarial loss on the prediction results will be investigated.

All experiments employ a real-world dataset recorded from test drives. The results are averaged over 20 runs to minimize statistical variance. The network architecture remains fixed for all experiments. Each run features 100,000 training iterations splitting the dataset in mini-batches of 64 samples. For the training, the Adam optimizer with a learning rate $\eta = 5 \cdot 10^{-4}$ is applied. The training process is validated every 1,000 iterations.

For calculating the reconstruction error, binary cross entropy (BCE) and mean squared error (MSE) are examined as a loss function. The BCE is defined as $\sum_n y_n \cdot \log \hat{y}_n + (1 - y_n) \cdot \log(1 - \hat{y}_n)$, where y is the true label and $\hat{y}$ the prediction. Analogous, the MSE is calculated as $\sum_n (y_n - \hat{y}_n)^2$. Both loss functions are averaged over the input dimensionality and the number of samples in each mini-batch to facilitate comparability. The weighting of the KL divergence depends strongly on the applied loss function. Preliminary experiments have revealed that $\beta = 0.01$ for BCE and $\beta = 0.001$ for MSE provide reasonable results.

6.1 Reconstruction

During the reconstruction phase, the network is fed exclusively with radar data and attempts to reconstruct the input. The validation loss is calculated by applying the two different loss functions to the network output and the radar maps from the validation set.

Figure 3 illustrates the validation loss with BCE and MSE over the complete training process. The root function is applied to the MSE to remove the square from the function and make the results comparable. The BCE loss curve features minor variances, in particular in the later iterations. In contrast, the MSE provides a smoother curve with a convex shape. The values are slightly lower than the BCE. BCE is frequently used in conventional VAEs. Nevertheless, it was originally designed for classification tasks, while the reconstruction of radar grid maps is a generative learning problem. Here, the MSE provides a more accurate evaluation of the loss between the generated grid map of the network and the radar grid map from the dataset, resulting in a more stable training process.

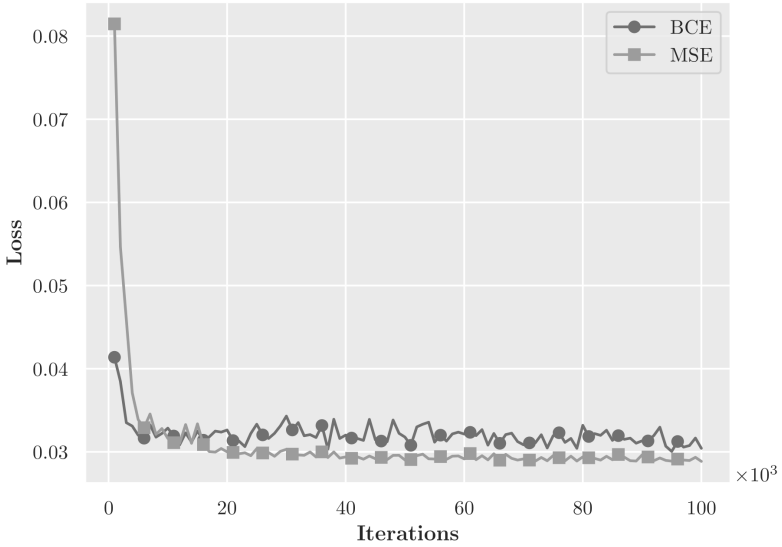


Figure 3: Valdiation loss of a variational autoencoder trained with binary cross entropy and mean squared error. The curves indicate the losses averaged over multiple runs.

6.2 Prediction

In order to generate specific radar grid maps, the network has to be conditioned. For the first step, we started just using the lidar data. The validation loss is calculated by drawing samples from a standard normal distribution, decoding and comparing them to the radar maps from the validation set.

Figure 4 illustrates the validation loss with both functions. In contrast to the reconstruction phase, the loss curve of the BCE has no longer a convex shape. Furthermore, the loss features high variances across all iterations. The local optimum is located after 3,000 iterations, while the loss becomes worse in later iterations. The MSE is highly erratic, but roughly preserves its convex shape. The variances in the first steps are significant, while the prediction becomes more confident in later iterations. The values are again below the optimum of the BCE curve after applying the root function. Both loss functions perform worse than for the reconstruction task. This behavior was to be expected as the samples for the validation are now randomly drawn from a standard normal

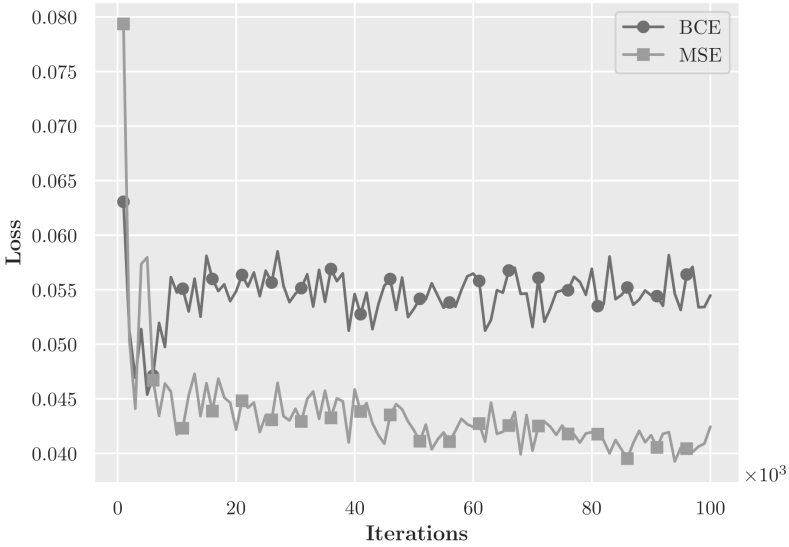


Figure 4: Valdiation loss of a conditional variational autoencoder trained with binary cross entropy and mean squared error. The curves indicate the losses averaged over multiple runs.

distribution. Eventually, the BCE is not really suitable to predict radar scans from lidar data. The MSE is also not a optimal metric for dealing with this learning problem. It is able to generate realistic radar data, but cannot be properly evaluated by validation techniques such as early stopping. Other metrics can be developed that match the learning problem better than MSE or BCE. However, designing a proper loss function is a challenging task and requires prior knowledge of the specific application. The MSE can be easily calculated and is sufficient to evaluate whether a method works well or not in this case. For this reason, the following experiments focus exclusively on MSE. To obtain a visual impression of the results, Figure 5 demonstrates a few exemplary grid maps from the best trained model of the CVAE with MSE. Surprisingly, the loss does not necessarily correspond to the visual evaluation.

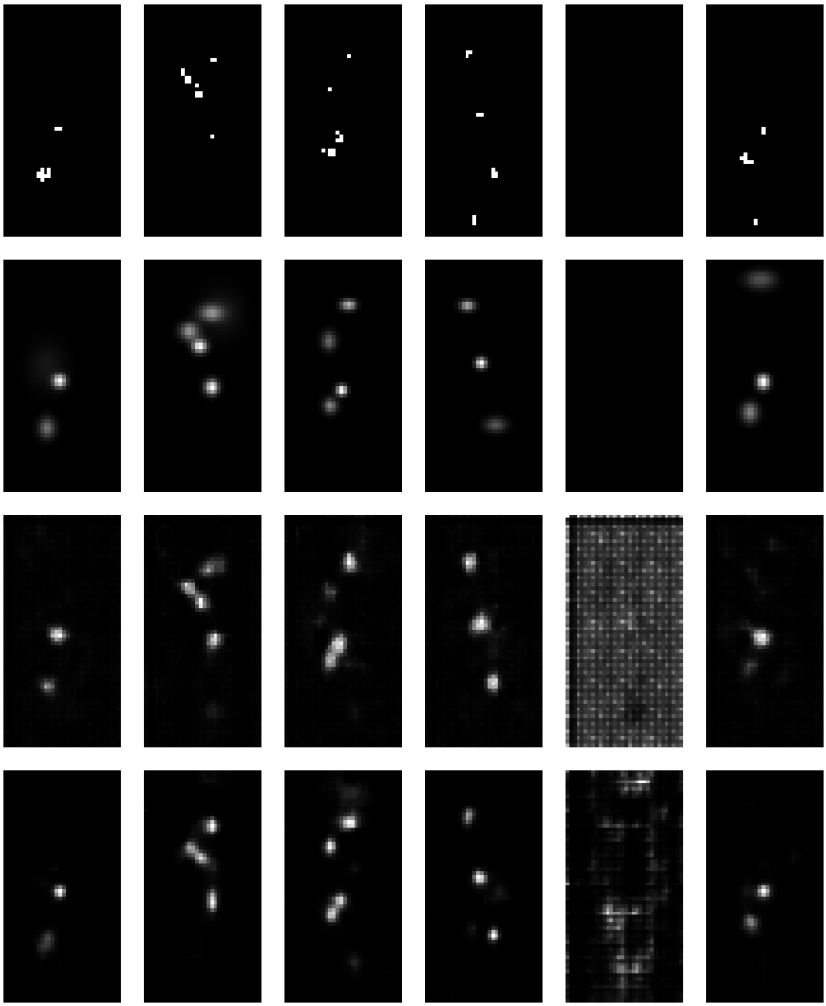


Figure 5: Exemplary grid maps generated by a conditional variational autoencoder trained on lidar data and with mean squared error. *First row:* Lidar grid maps from the test set. *Second row:* Radar grid maps from the test set. *Third row:* Generated radar grid maps of the model trained 7,000 iterations. *Fourth row:* Generated radar grid maps of the model trained 100,000 iterations.

Table 1: Runtimes for a single prediction step of the trained models. The results were averaged over 10,000 measurements with the highest and lowest 5 % omitted. All measurements were calculated in milliseconds on a NVIDIA Quadro M2000M GPU.

Model	Minimum	Average	Maximum	Variance
VAE	2.000	2.278	3.000	0.153
CVAE	4.000	4.624	5.000	0.160

The object detections after 100,000 iterations in the fourth row are smoother and the variances are better predicted. Moreover, the noise in the fifth example is significantly reduced over the model after 7,000 iterations. However, the MSE for the model after 100,000 iterations is at $1.049 \cdot 10^{-3}$, for the model after 7,000 iterations at $0.829 \cdot 10^{-3}$. The processing of camera images is difficult due to the high information density. Since the model conditioned with lidar data already provides reasonable results, we will first focus on this model and include camera images in future experiments. The runtime of a single prediction step is evaluated for both presented models in Table 1. All values are averaged over 10,000 measurements to eliminate irregularities within the computation. A prediction step in the CVAE using lidar data takes between 4 and 5 ms. In comparison, the pure reconstruction without lidar data takes approximately 2 ms. This was to be expected, as the more operation the network has to process, the more computation time is required. After implementing the model on a control unit, the computation time increases tenfold as a rule of thumb. Since current radar systems are clocked with a frequency of approximately 10 Hz, the algorithm could be deployed in an automated driving vehicle.

6.3 Regularization

Previous experiments have demonstrated that the KL divergence is a key hyperparameter for the training of the CVAE. Figure 6 illustrates the training process with different parameterizations of the regularization. Four different weights

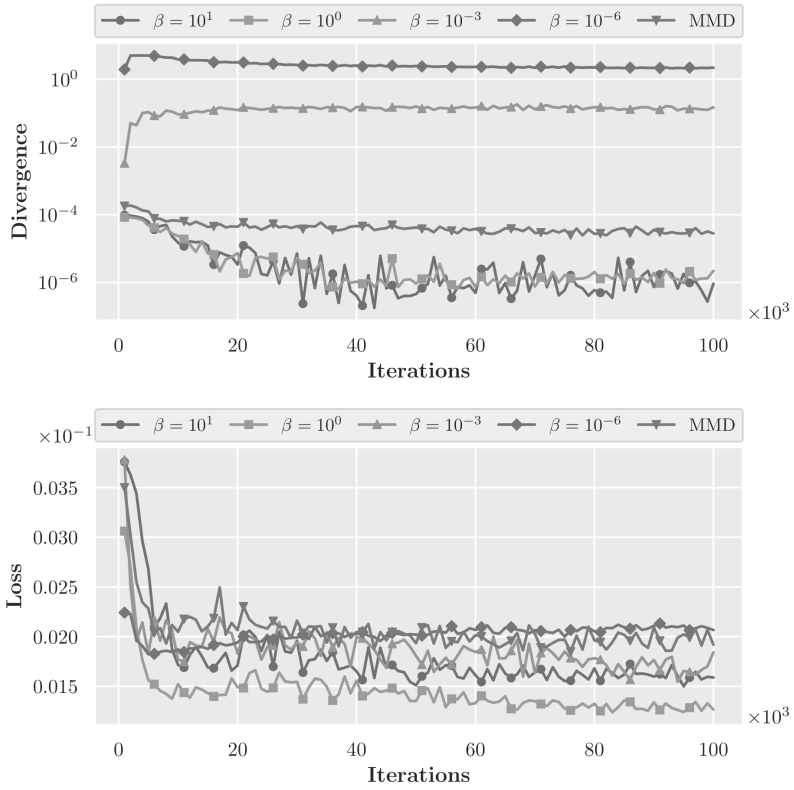


Figure 6: Validation of a conditional variational autoencoder using regularization with different weights. *Top:* Line plot with logarithmic scale on the *y*-axis indicating Kullback-Leibler divergences averaged over multiple runs. *Bottom:* Line plot indicating losses averaged over multiple runs.

for the KL divergence are selected. In addition, mutual information with MMD is provided in a fifth example. The best results are achieved by the models with $\beta \geq 1$. However, these models also features the smallest KL divergence. If the divergence converges to zero, regularization is vanishing and the network reconstructs the input independently from the condition. Unfortunately, the dataset seems to be too small to demonstrate this effect. It is sufficient to simply reconstruct the input, while the loss, due to zero KL divergence, is negligible. For this reason, the optimal β is hard to define. Another observation is that the

KL divergence is much less erratic for $\beta < 1$. Alternatively, the same effect can be achieved by providing mutual information, where $\beta = 1$. Furthermore, if β is close to zero, the KL divergence increases and regulates the loss too much, so that no reasonable output is possible. This behavior is also expressed by an increasing loss for the model with $\beta = 10^{-6}$.

Higgins et al. suggest to select $\beta > 1$ [13]. This is an interesting point as our empirical studies figured out that a parameterization of $\beta \leq 1$ also works very well. Furthermore, $\beta \gg 1$ pushes the KL divergence close to zero, which implicates a vanishing regularization.

7 Discussion

The experiments revealed that a complete radar scan can be simulated by a CVAE using lidar data. For the reconstruction, both loss functions BCE and MSE are applicable. If the radar scan is predicted solely by lidar data, the model with BCE will yield poor results. Moreover, the loss does not necessarily represent the visual quality of the radar grid maps. This is an indication that there might be better loss functions than MSE to evaluate this learning problem.

In the next step, camera images can be incorporated to the network structure. The architecture requires high modeling capacities in order to process the information density of the image data properly. Otherwise, the model only learns features randomly and runs in local minima without generalizing the data. However, very deep networks with a significant number of trainable parameters, such as VGG16, demand for a high amount of data. In addition, those models suffer from vanishing gradients, meaning that the updates in the early layers are shrinking exponentially. There are several possibilities to counteract these problems. Potential solutions could be to project the lidar point cloud into the images or to operate on preprocessed high-level objects extracted from the camera images. In this cases, even smaller architectures would be sufficient to process the image data, as the information density is reduced.

In order to improve the results of the CVAE, the use of GANs is recommended. GANs do not suffer from blurry outputs as they optimize an adversarial loss rather than the maximum likelihood. Nevertheless, they belong to the most difficult networks to train. This characteristic is due to the fact that the adversarial loss reacts sensitively on small changes. Mode collapse is a frequent result of such poorly parameterized GANs. There exists a wide range of additions that attempt to counter these problems. In general, an adversary approach provides better results, but requires more data, longer training and fine tuning of the hyper-parameters.

8 Conclusion

The objective of this paper was to examine whether an automotive radar sensor can be simulated using the correlation between the input data of different sensors. For exploring this task, a CVAE was implemented and tested in various experiments. We have demonstrated that our model generates reasonable radar grid maps that solely result from data of a lidar sensor. For future work, the use of camera images and adversarial learning promise improvements over the presented methods.

References

[1] H. Winner. *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*, Chap. 17, pp. 325–403. Springer, 2016.

[2] T. A. Wheeler *et al.* "Deep Stochastic Radar Models". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 47–53, 2017.

[3] M. J. Barney. "GPU Advancements Reduce Simulation Times for 25 GHz Automotive Radar Models". In: *2015 9th European Conference on Antennas and Propagation (EuCAP)*, pp. 354–356, 2015.

[4] M. Lundgren, L. Svensson, and L. Hammarstrand. "Variational Bayesian Expectation Maximization for Radar Map Estimation". *IEEE Transactions on Signal Processing*, Vol. 64, No. 6, pp. 1391–1404, 2016.

[5] A. Scheel and K. Dietmayer. "Tracking Multiple Vehicles Using a Variational Radar Model". *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[6] N. Hirsenkorn *et al.* "A Non-Parametric Approach for Modeling Sensor Behavior". In: *2015 16th International Radar Symposium (IRS)*, pp. 131–136, 2015.

[7] N. Hirsenkorn *et al.* "Virtual Sensor Models for Real-Time Applications". *Advances in Radio Science*, Vol. 14, pp. 31–37, 2016.

[8] N. Hirsenkorn *et al.* "Learning Sensor Models for Virtual Test and Development". In: *11. Workshop Fahrerassistenzsysteme und automatisiertes Fahren*, pp. 115–124, 2017.

[9] A. Suhre and W. Malik. "Simulating Object Lists Using Neural Networks in Automotive Radar". In: *2018 19th International Conference on Thermal, Mechanical and Multi-Physics Simulation and Experiments in Microelectronics and Microsystems (EuroSimE)*, pp. 168–172, 2018.

[10] E. L. Zec, N. Mohammadiha, and A. Schliep. "Statistical Sensor Modelling for Autonomous Driving Using Autoregressive Input-Output HMMs". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1331–1336, 2018.

[11] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[12] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

[13] I. Higgins *et al.* "β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[14] S. Zhao, J. Song, and S. Ermon. “InfoVAE: Balancing Learning and Inference in Variational Autoencoders”. In: *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pp. 5885–5892, 2019.

[15] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[16] K. Sohn, H. Lee, and X. Yan. “Learning Structured Output Representation Using Deep Conditional Generative Models”. In: *Advances in Neural Information Processing Systems 28*, pp. 3483–3491, 2015.

[17] I. Goodfellow *et al.* “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems 27*, pp. 2672–2680, 2014.

[18] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[19] X. Glorot, A. Bordes, and Y. Bengio. “Deep Sparse Rectifier Neural Networks”. In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.

[20] 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*, pp. 448–456, 2015.

Robot-Human Object Handover with Haptic Input and Visual Feedback

Rodrigo J. Velasco-Guillen[1], Maximilian Krämer[2], Frank Hoffmann[2], Torsten Bertram[2], Philipp Beckerle[1]

[1]Elastic Lightweight Robotics Group, TU Dortmund

[2]Institute of Control Theory and Systems Engineering, TU Dortmund

Abstract

When working together, human and robots complement each other's abilities yielding more efficient task execution. The cognitive abilities of humans are complemented by physical robot support, reducing fatigue and stress, particularly in repetitive tasks. A robot-human handover is an example of interaction in which human and robot act in temporal and spatial concurrence, with particular relevance in scenarios without temporary object placement, such as service robotics. This physical and cognitive interaction between human and robot requires coordination and effective communication.

This work concerns a robot-human object handover with a robot arm equipped with a two-finger robotic gripper. Mutual task awareness between human and robot is attained by both haptic and visual feedback. A force/torque sensor measures the grasp forces imposed by the human which eventually trigger the release of the object. The handover behavior releases the object once force and torque thresholds are exceeded. The decision rule aggregates direct and integral measures of the force/torque signal. In other words, the gripper releases either if the human imposes a significant force or a medium force over an extended period of time. Furthermore, a lighting ring in the robot wrist provides the user with visual feedback on the force exerted and indicates a forthcoming grip release. The evaluation of the algorithm considers the qualitative individual human perception of the naturality of the handover with

and without visual feedback. Results show that merely static thresholds w.r.t. direct force and torque measures provide an effective handover but parameterization proved difficult as the algorithm does not adapt to the conditions of the handover. An adaptive threshold based on integral measures attained better results when parameterized for maximum adaptability. Furthermore, visual feedback has a positive impact on the human perception of the naturality of the handover as users receive information on the required force for attaining an object release.

1 Introduction

Collaborative interaction between human and robot is advantageous as humans contribute expertise, knowledge and understanding for the correct execution of tasks, while the robot assistance reduces fatigue and stress and increases human capabilities and precision [1]. A robot-human handover task, namely the handing of an object from the robot to a human, is an example of human-robot interaction where haptic feedback is particularly important in order to attain communication between human and robot by means of touch. In case of physical interaction it is common to employ haptic sensors such as force/torque [2]. Strategies for achieving an effective handover algorithm often rely on the qualitative human perception of the task [3, 4, 5, 6], while analyzing human intent scenarios and overcoming undesired perturbations that might cause an erroneous gripper release [7]. Furthermore, social cues and communication can further improve the coordination between human and robot [8, 9].

This work investigates the human-robot interaction in an object handover task performed with a robot arm equipped with a two-finger robotic gripper. Haptic feedback is attained with a force/torque sensor mounted between the wrist and the gripper. A lighting ring mounted at the wrist provides the user with visual feedback. The gripper setup for experimentation is shown in Figure 1. During interaction, the force/torque sensor provides information on the forces and torques exerted by the user, while the lighting ring indicates the robot's readiness to release the object by changing color (Figure 2).

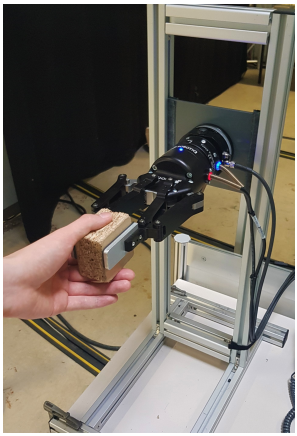


Figure 1: Experimental setup

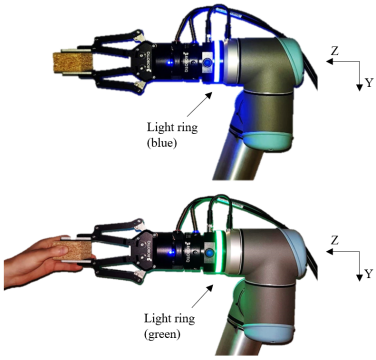


Figure 2: Lighting ring changes from blue (top) to green (bottom) when close to release

Section 2 presents the implementation of the handover algorithms, while Section 3 shows the results obtained from the qualitative assesment of a number of volunteers surveyed. The research is concluded and future work is presented in Section 4.

2 Handover Algorithm

Let $f \in \mathbb{R}$ denote a force or torque obtained from the force/torque sensor, a logical trigger $a_1 \in \mathbb{B}$ is defined when a direct threshold $\vartheta_d \in \mathbb{R}$ is met, such that

$$a_1 = \begin{cases} 1, & \text{if } f \geq \vartheta_d \\ 0 & \text{otherwise} \end{cases}. \tag{1}$$

Yet, such a trigger becomes active even if the signal exceeds the threshold only for a brief moment. Thus a static trigger is susceptible to noise on the force/torque sensor measurements or accidental contact with the gripper or object. To improve the robustness, the handover trigger $a_h \in \mathbb{B}$ is only activated

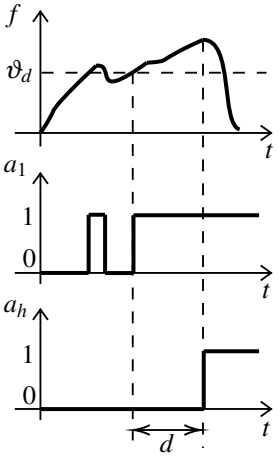


Figure 3: Handover algorithm with a fixed duration

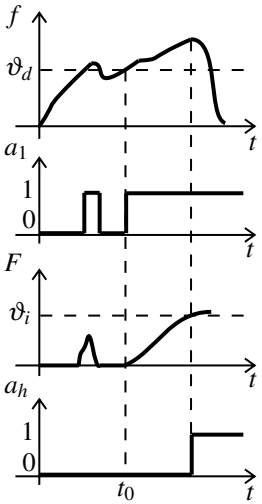


Figure 4: Handover algorithm with integral threshold

if the condition a_1 is fulfilled for a minimum time duration $d \in \mathbb{R}$. This algorithm is depicted in Figure 3.

However, for a fixed duration d, the decision rule does not consider the rate of change of the force or torque exerted by the user. If the force or torque is exerted slowly, the human's experience might differ as if the same force or torque is applied rapidly. In order to overcome this problem, an adaptive threshold is implemented to replace the fixed time duration d of the previous algorithm: Consider an integral force or torque $F \in \mathbb{R}$ which starts to accumulate once a_1 becomes active, such that

$$F = \begin{cases} \int_{t_0}^{t} (f - \vartheta_d)\,\mathrm{dt}, & \text{if } a_1 = 1 \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

in which $t_0 \in [0,t]$ denotes the instance of activation of a_1. Now, the handover trigger a_h becomes active once F meets an integral threshold $\vartheta_i \in \mathbb{R}$, such that

$$a_h = \begin{cases} 1, & F \geq \vartheta_i \\ 0 & \text{otherwise} \end{cases} . \tag{3}$$

Notice that since F is a cumulative measure of f over time, the threshold ϑ_i is met earlier if f is exerted more rapidly at a higher rate of change. This algorithm is depicted in Figure 4.

The algorithms only refer to a scalar force or torque signal. However, a force/torque sensor measures a force $\mathbf{f} = (f_x, f_y, f_z)^{\mathrm{T}}$ and torque $\boldsymbol{\tau} = (\tau_x, \tau_y, \tau_z)^{\mathrm{T}}$ vector which can be aggregated in different ways. In such a case, individual handover triggers release separately along the six dimensions and are combined with logical operators such as AND and OR. Instead of a logical combination the trigger can also refer to the absolute magnitude of either force $||\mathbf{f}||_2$ or torque signal $||\boldsymbol{\tau}||_2$.

3 Experimental Results and Evaluation

In a handover scenario, the object release is triggered by different combination and aggregation of forces exerted by the user. On a preliminary experimentation, it was found that users usually execute a combination of pulling and weight compensating when grasping the object. Twisting, on the other hand, proofs to be an effective method for triggering the object release, however it is non-natural for the user. The evaluation in this work considers the scenario which requires the user to pull the object out of the gripper in order to trigger its release (force in the positive Z axis). Experimentation shows that users typically exert up to 30 N of force in this direction when expecting a gripper release.

This section presents the parameter selection and user study in order to evaluate the naturality and qualitative user experience of the handover obtained by the algorithms implemented.

Table 1: Parameters for handover evaluation

Algorithm with fixed duration	$[\vartheta_d, d]$
Parameter setting A1	$[5\,\mathrm{N}, 100\,\mathrm{ms}]$
Parameter setting B1	$[1\,\mathrm{N}, 300\,\mathrm{ms}]$
Algorithm with integral threshold	$[\vartheta_d, \vartheta_i]$
Parameter setting A2	$[5\,\mathrm{N}, 0.2\,\mathrm{kg\,m\,s^{-1}}]$
Parameter setting B2	$[1\,\mathrm{N}, 0.5\,\mathrm{kg\,m\,s^{-1}}]$

3.1 Parameter selection

It is evident how for both algorithms the parameter sets $[\theta_d, d]$ and $[\theta_d, \theta_i]$ affect the user experience of the gripper release. The activation threshold θ_d determines the absolute magnitude, whereas d and θ_i determine the temporal response. Even though there are plausible ranges for both, it is difficult to estimate which parameter setting corresponds to a natural release.

Experiments with volunteers determined that the perception of naturality in a handover is affected by the overall time of release and the magnitude of the force exerted by the user at the moment of release: Prompt releases are preferred as they are more similar to a human-human handover, while low forces at the moment of release create lower backlash for the user. This means that the parameter settings for the algorithms should be set as sensitive as possible, but not too sensitive that fluctuations in force, caused by noise or undesired motions triggers an non-desired object release. With this in mind, notice that a higher direct threshold ϑ_d benefits from a lower temporal setting, while a lower ϑ_d requires a higher temporal setting to overcome the sensitivity.

Different parameters are tested and tuned in order to obtain a compromise between sensitivity and naturalness. Table 1 presents the selected parameters. A typical handover time, the duration between the moment the user initiates pulling and the gripper release, with these parameter settings range between 150 ms and 1000 ms.

Table 2: Naturality comparison results

Comparison		No. of	No. of times chosen		
H. 1	H. 2	tests	H. 1	H. 2	Neither
A1	B1	30	10	9	11
A2	B2	30	5	18	7
A1	B2	15	4	8	3
B1	B2	15	5	7	3
A1	A1 (light)	15	3	7	4
B1	B1 (light)	15	5	5	5
A2	A2 (light)	15	3	9	4
B2	B2 (light)	15	4	9	4

3.2 User Study

The evaluation considers a user study in which 15 volunteers are presented with different combinations of parameter settings. The subjects are not informed about the algorithms and merely obtain the instruction to pull the object in order to release it. The subjects perform two handovers with different settings and afterwards decide whether one handover felt more natural or if they did not experience any significant difference. Results for this comparisons are summarized in Table 2. At each handover, the force torque sensor signals are recorded and analyzed to determine the object release time and the final force at the moment of release. The force rate of change is calculated by dividing the final force between the total time from the moment that the user starts pulling. Figure 5 shows a typical handover force signal.

When comparing parameterizations A1 and B1 for the fixed duration algorithm, A1 is found more natural 33.3% of the times, while B1 is considered more natural 30% of the times. Results show that at each test, the handover chosen as the more natural is the one that shows a faster release time and/or a lower force at the moment of release. However, there is no conclusive winner between the two parameterization, as users prefer one or the other based on the way they exert forces during the handover. On tests in which A1 is deemed more natural, users exert force at an average rate of $42.04\,\mathrm{N\,s^{-1}}$, whereas on tests in which B1 seems preferable, users exert the force at an average rate of

$20.83\,\mathrm{N\,s^{-1}}$. This implies that users who exert the force at a higher rate prefer a higher direct threshold with a short delay, while users who exert the force at a lower rate prefer a lower threshold with a longer delay.

When comparing parameterizations A2 and B2 for the integral threshold algorithm, A2 is considered more natural 16.7% of the times, while B2 seems more natural 60% of the times. This implies that the B2 parameterization is more commonly preferred by users, as it has maximum adaptability with a higher integral threshold. Results show that tests in which B2 is chosen as more natural, the force rate of change varied from $3.48\,\mathrm{N\,s^{-1}}$ to $115.63\,\mathrm{N\,s^{-1}}$, meaning that even for a high range of force rate of change, the B2 parameterization is more commonly preferred.

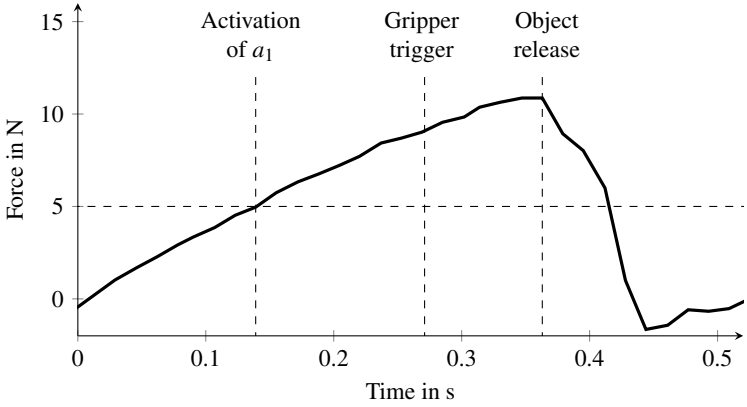


Figure 5: Force analysis on typical handover with 5 N force threshold

In order to compare the algorithms with fixed duration and integral threshold, tests were performed to compare the naturalness between A1 and B2, and B1 and B2. Results show that the parameterization B2 is considered as more natural 50% of the times, while A1 or B1 are chosen as more natural 30% of the times. This implies that the adaptability of the algorithm when using a high integral threshold is more commonly preferred regardless of the conditions during the handover.

Visual feedback changes the lighting to a full green color when the direct threshold ϑ_d is met. When comparing all parameterizations with and without lighting ring, results show that the light feedback improves the naturalness of the handover on 50% of the tests, with 25% of the tests showing more naturalness without it. Furthermore, 66.7% of volunteers express that the light feedback is helpful for the handover, while 20% express indifference to its usefulness, and 13.3% state that the light is rather distracting. This shows that visual feedback has a positive impact on the handover assessment. By giving feedback on the force exerted, users are capable of exerting only the necessary amount of force required. However, since the light depends only on the force and not on the temporal response of the algorithm, the feedback is experienced as more useful for parametrizations with high force thresholds, such as A1 and A2.

Throughout the experimentation, users express difficulty in making a clear verdict w.r.t. the naturalness of a pair of parameter settings. This might be due to an undesired and somewhat unpredictable delay between the gripper trigger command and the real object release observed on the handover analysis. This delay is observed and analyzed for a total of 220 handovers, with a mean of 210.6 ms, a standard deviation of 105.9 ms and ranging from 25.3 ms to 416 ms.

4 Conclusions and Further Work

Results show that a generally natural handover between the user and a robotic gripper equipped with a force torque sensor is possible when implementing an algorithm that adapts to the conditions at which the force is exerted on the gripper. A single force threshold and fixed duration is effective but its parametrization is difficult as the qualitative assessment strongly depends on the conditions of the handover. The integral threshold improves the algorithm by adapting the release time with respect to the rate of change of the force exerted by the user. The parameterization of this algorithm is most effective with a lower force threshold and a higher integral threshold for maximum adaptability.

Visual feedback has positive impact on the perception of naturalness of the handover. During experimentation most users agree that the lighting feedback of the force exerted is useful to regulate the force necessary to trigger an object release.

The evaluation method proposed in this work aims to analyze the characteristics and effectiveness of the handover algorithms presented. However, more robust methods for parameter tuning and statistical analysis should be considered in future work. Furthermore, the handover algorithm could benefit from additional sensors for haptic feedback, such as tactile sensors on the gripper's fingers. Additionally, given the typical duration of a handover, the delay found for the gripper release might have a negative impact on the handover perception. Thus, future work should consider improving the gripper response time or using a different gripper with a faster response.

References

[1] A. De Santis, B. Siciliano, A. De Luca and A. Bicchi. "An Atlas of physical human-robot interaction". In: *Mechanism and Machine Theory*, 43.3. S. 253-270. 2008.

[2] M. Grunwald. (Ed.). "Human haptic perception: Basics and applications". Springer Science & Business Media. 2008.

[3] M. Cakmak, S. Srinivasa, M. K. Lee, J. Forlizzi and S. Kiesler. "Human preferences for robot-human hand-over configurations". In: *International Conference on Intelligent Robots and Systems*, S. 1986-1993. 2011.

[4] W. P. Chan, C. A. Parker, H. M. Van der Loos and E. A. Croft. "A human-inspired object handover controller". In: *The International Journal of Robotics Research*, 30.8. S. 971–983. 2013.

[5] N. Hendrich, H. Bistry, J. Liebrecht and J. Zhang. "Natural robot-human handover combining force and tactile sensors". *Workshop on Assistance and Service Robotics in a Human Environment*, IROS 2014. Chicago, USA. 2014.

[6] M. Huber, M. Rickert, A. Knoll, T. Brandt and S. Glasauer. “Human-robot interaction in handing-over tasks”. In: *International Symposium on Robot and Human Interactive Communication* , S. 107-112. Munich. 2008.

[7] A. G. Eguilúz, I. Rañó, S. A. Coleman and T. M. Mcginnity. “Reliable object handover through tactile force sensing and effort control in the Shadow Robot hand”. In: *International Conference on Robotics and Automation*, S. 372-377. 2017.

[8] A. Edsinger and C. C. Kemp. “Human-robot interaction for cooperative manipulation: Handing objects to one another”. In: *International Symposium on Robot and Human Interactive Communication*, S. 1167-1172. 2007.

[9] K. Strabala, M. K. Lee, A. Dragan, J. Forlizzi, S. S. Srinivasa, M. Cakmak, and V. Micelli. “Toward seamless human-robot handovers”. In: *Journal of Human-Robot Interaction*, 2.1. S. 112-132. 2013.

Robuste Online-Klassifikation kritischer Netzereignisse unter Berücksichtigung von Störgrößen

André Kummerow

Fraunhofer IOSB, Institutsteil Angewandte Systemtechnik (AST), Fraunhofer Center for Machine Learning
Am Vogelherd 50, Ilmenau
E-Mail: andre.kummerow@iosb-ast.fraunhofer.de

1 Einführung

Die Führung elektrischer Netze bildet die Grundlage für eine sichere Versorgung zukünftiger Energiesysteme. Infolge der zunehmenden Verdrängung konventioneller Kraftwerke durch hohe Mengen an fluktuierenden, dezentralen Erzeugungskapazitäten erhöhen sich die Anforderungen zur Sicherstellung eines stabilen Netzbetriebs. Eine Schlüsselrolle kommt hierbei dem Einsatz zeitsynchronisierter Phasormessungen zu (engl.: *phasor meaurement unit*, PMU) durch welche Strom- und Spannungsphasoren, Frequenzen sowie Frequenzänderungen im dynamischen Zeitbereich (typischerweise zwischen 10 und 50 Messungen pro Sekunde) an mehreren Messstationen im Netz erfasst werden können zur Bereitstellung neuartiger Überwachungs- und Steuerungsapplikationen in zukünftigen Netzleitwarten. Hierzu zählen z.B. dynamische Stabilitätsbewertung, Modellvalidierung, Inselerkennung, Echtzeitalarmierungen oder auch Leitungsüberwachung. [1, 2, 3, 4]
Als einer der Hauptschwerpunkte aktueller Forschungstätigkeiten werden Phasormessungen verwendet zur automatischen Klassifikation kritischer Netzereignisse (z.B. Ausfälle von Netzbetriebsmitteln, Netzpendelungen, Fehlfunktionen) mittels Verfahren des maschinellen Lernens. Dies ermöglicht die

weitestgehend automatisierte und schnelle Einleitung vordefinierter Gegenmaßnahmen zur Verhinderung von Versorgungsausfällen oder Netzinstabilitäten. Messdaten sind zum Anlernen der Klassifikationsverfahren dabei in der Regel nicht ausreichend, da sie nur Informationen über bereits eingetretene kritische Netzsituationen enthalten und diese erst aus den historischen Prozessdaten aufwändig extrahiert werden müssen. Mittels dynamischer Netzsimulationen können unter Berücksichtigung der Netztopologie für einen vordefinierten Netzzustand die für die Betriebsführung relevanten, kritischen Situationen nachgestellt werden. Infolge der hohen Systemkomplexität (z.B. fluktuierende, dezentrale Erzeuger) sowie der zunehmenden Anzahl an Freiheitsgraden (z.B. leistungselektronische Betriebsmittel, HGÜ[1]-Systeme) heutiger elektrischer Netze ist nur eine begrenzte Menge an möglichen Betriebssituationen mit vertretbarem Aufwand modellierbar. Daraus folgende Abweichungen zwischen Trainings- und Messdaten wirken als Störgrößen negativ auf den Klassifikationsprozess ein und können zu Fehlentscheidungen in der Anwendungsphase führen mit u.U. weitreichenden, negativen Auswirkungen auf die Netzbetriebsführung. Zur robusten Klassifikation von PMU-Daten sind diese Störgrößen bereits in der Modellentwurfsphase zu berücksichtigen z.B. durch Hinzufügen einer Rückweisungsoption im Klassifikator oder der Hinzunahme externer Einflussgrößen (z.B. der Netzzustand). Dies erfordert die korrekte Zuweisung bzw. Rückweisung noch nicht erlebter Beobachtungen, welche durch das Einwirken von Störgrößen mehr oder weniger stark von der bekannten Trainingsmenge abweichen und sowohl zu bekannten als auch unbekannten Ereignissen bzw. Klassen gehören können. Bisherige Forschungsansätze vernachlässigen derartige Betrachtungen und erschweren somit den Praxiseinsatz von Verfahren zur PMU-basierten Klassifikation kritischer Netzereignisse.

Innerhalb dieses Beitrags werden erste Modellansätze zur robusten Identifikation und Lokalisierung kritischer Netzereignisse auf Basis rekurrenter neuronaler Netze sowie Erweiterungen u.a. durch Einsatz siamesischer Modellarchitekturen vorgestellt. Kapitel 2 beschreibt bisherige Ansätze zur Detektion, Identifikation und Lokalisierung kritischer Netzereignisse auf Basis von PMU-Daten sowie vorhandene, allgemeine Ansätze zur robusten Klassifikation bei Anwesenheit unbekannter Klassen. Anschließend erfolgt in Kapitel 3 die

[1] Hochspannungs-Gleichstrom-Übertragung

Systematisierung der Störgrößen bei der Klassifikation von PMU-Daten unter Verwendung dynamischer Netzsimulationen sowie die formale Beschreibung innerhalb eines systemtheoretischen Modells. In Kapitel 4 werden Modellentwürfe zur Klassifikation von PMU-Daten vorgeschlagen und in Kapitel 5 erste Klassifikationsergebnisse für ein simuliertes Übertragungsnetz beschrieben. Kapitel 6 fasst die Ergebnisse zusammen und liefert Rückschlüsse für weiteren Untersuchungen sowie Modellstrukturen.

2 Stand der Technik

2.1 PMU-basierte Erkennung kritischer Netzereignisse

In den letzten Jahren wurden umfassende Untersuchungen zur Klassifikation kritischer Netzereignisse auf Basis von PMU-Daten durchgeführt. Diese umfassten unterschiedliche Kombinationen aus Verfahren zur Merkmalsextraktion im Zeit- bzw. Frequenzbereich sowie zur linearen oder auch nicht-linearen Klassifikation. Derzeit verfügbare Ansätze können grob drei Hauptaufgaben bzw. Kategorien zugeordnet werden:

- *Detektion*: Erkennung von Abweichungen zum ungestörten Betrieb,
- *Identifikation*: Erkennung des Ereignistyps sowie
- *Lokalisierung*: Erkennung des Ereignisorts.

So wurden in [5, 6, 7] Spannungs- und Frequenzsignale ausgewertet zur Detektion dreiphasiger Netzfehler sowie niederfrequenter Oszillationen u.a. auf Basis von Wavelet-Dekomposition, Hauptkomponentenanalyse sowie MVEE-Transformation (engl.: *minimum volume enclosing ellipsoids*). Die Unterscheidung zwischen gestörten und ungestörten Betriebssituationen erfolgte hierbei u.a. auf Basis von Support-Vektor-Maschinen. Sehr umfangreiche Untersuchungen zur Identifikation und Lokalisierung verschiedener Ausfalltypen wie z.B. Generatorausfälle, Kurzschlüsse oder Lastabwürfe liefern Arbeiten in [8, 9, 10, 11, 12, 13, 14]. Dabei wurden neben den oben genannten Ansätzen auch auf leistungsfähigere Methoden zur Extraktion von Zeitreihenmerkmalen

zurückgegriffen, wie z.B. *shapelets* oder S-Transformation[2], und mit konventionellen Klassifikationsverfahren verknüpft. Darüber hinaus wurden in [15] *deep learning* basierte Ansätze unter Verwendung von *multi-layer perceptrons*, *convolutional neural networks* sowie *deep belief networks* für den Einsatz zur PMU-basierten Identifikation von Netzereignissen untersucht.
Neben dem Erreichen hoher Klassifikationsgenauigkeiten spielt die Ausführungszeit des Klassifikationsmodells eine große Rolle für den Online-Einsatz in Netzleitwarten. Weitere, hier nicht adressierte Herausforderungen ergeben sich aus dem Umgang mit Überlagerungseffekten, d.h. dem zeitgleichen bzw. kaskadiertem Eintreten mehrerer Netzereignisse, sowie der Reaktion auf Struktur- bzw. Topologieänderungen im Netz.

2.2 Verfahren zur robusten Klassifikation

Zur Berücksichtigung von Störgrößen durch eine begrenzte und ggf. nichtrepräsentative Trainingdatenbasis sind konventionelle Klassifikationsverfahren sowie deren Generalisierungsfähigkeit i.d.R. nicht ausreichend. Auf mehreren Forschungsgebieten werden aktuell unterschiedliche Klassifikationsansätze beschrieben bzw. diskutiert zur korrekten Behandlung noch nicht erlebter, von der Trainingsmenge abweichender Beobachtungen bekannter oder auch unbekannter Klassen.
Im Bereich der Datenstromanalyse (engl.: *data stream mining*) werden Konzepte untersucht zur Reaktion auf Änderungen im Merkmalsraum (engl.: *concept drift*) sowie in der Klassenkonfiguration (engl.: *concept evolution*) und der entsprechenden Online-Anpassung des Klassifikationsmodells im laufenden Betrieb. Speziell zur Detektion neuer Klassen auf Basis noch nicht erlebter Beobachtungen werden u.a. Konzepte unter Verwendung von Clusterverfahren bzw. Verfahren zur Anomalieerkennung, Kombination aus lokalen und globalen Merkmalsräumen oder auch Ensemble-Ansätzen eingesetzt [16, 17, 18]. Als wesentliche Einschränkung werden i.d.R. überwachte bzw. teilüberwachte Datenströme während der Anwendungsphase vorausgesetzt. Zudem ist die Ergebnisqualität der beschriebenen Verfahren stark abhängig von der gewählten

[2] Stockwell-Transformation

Merkmalstransformation, welche speziell im Fall von PMU-Daten als hochdimensionale Zeitreihen sowie der hohen Klassenanzahl ggf. nicht optimal ermittelt werden kann.

Als weiterer Forschungsschwerpunkt befasst sich die offene Klassifikation (engl.: *open-set recognition*, *open-world classification* bzw. *open-category classification*) mit dem Einsatz von Rückweisungsmechanismen zur Behandlung unbekannter Klassen. Hierzu wird ein offener Merkmalsraum (engl.: *open space*) mit geringer Sicherheit über die wahre Klassenzugehörigkeit definiert und die Diskriminanzfunktion auf Basis der räumlichen Verteilung bekannter Trainingsinstanzen (engl.: *abating*) angepasst. Im Gegensatz zur Datenstromanalyse können Verfahren aus dem Bereich der offenen Klassifikation in neuronale Netze integriert und ganzheitlich gelernt werden mit entsprechenden Vorteilen bei der Verarbeitung komplexer Datensätze. Eine nachträgliche Anpassung bzw. Optimierung des Klassifikationsmodells in der Anwendungsphase ist demgegenüber nicht möglich. [19, 20, 21, 22]

Der dritte Forschungsschwerpunkt umfasst den Bereich des Lernens auf Basis einzelner bzw. weniger Beispiele (engl.: *one-shot learning* bzw. *few-shot learning*) unter der Verwendung siamesischer neuronaler Netze und entsprechender Erweiterungen. Im Gegensatz zu den vorherigen Ansätzen erfolgt die Verarbeitung der Trainingsinstanzen in relationaler Form, d.h. durch Vergleich einer Beobachtung mit mehreren bekannten Stützbeispielen bzw. Prototypen über einen Komparator. Die Multiklassen-Problemstellung wird somit auf mehrere binäre Klassifikationsprobleme aufgeteilt und ermöglicht gleichzeitig die Generalisierung auf neue Klassen ohne Neutrainieren des Klassifikationsmodells. Darüber hinaus ermöglichen derartige Modellstrukturen eine inherente Rückweisungsoption (fehlende Übereinstimmung einer Beobachtung mit allen Stützbeispielen). Als Komparatoren können definierte Metriken (z.B. Manhattan-Distanz) oder parametrische Funktionen verwendet werden [23, 24, 25]. Der Einsatz Komparator-basierter Klassifikatoren speziell zur Rückweisung nicht erlebter Beobachtungen wird bisher kaum untersucht, bietet jedoch Vorteile im Umgang mit einer hohen Anzahl bekannter sowie unbekannter Klassen und einer verminderten Trainingsdatenbasis.

3 Systemtheoretische Betrachtung

Infolge von Unvollständigkeiten, Ungenauigkeiten sowie Vereinfachungen in dynamischen Netzsimulationen wirken unterschiedliche Störgrößen auf den Klassifikationsprozess ein und können zu größeren Abweichungen zwischen Trainings- und Messdaten führen. Eine allgemeines Systemmodell für die PMU-basierte Klassifikation kritischer Netzereignisse liefert Bild 1. Darin wird zwischen einer Simulations- und Prozessdomäne (S bzw. P) sowie dem Klassifikationsmodell (M) unterschieden, dessen Ziel die korrekte Schätzung des Netzereignisses $\hat{y}$ umfasst. Das Klassifikationsmodell ist zunächst vereinfacht dargestellt bestehend aus den Modulen Merkmalsextraktion zur Erzeugung eines Merkmalsvektors $\underline{f}$, dem Klassifikator zur Bestimmung des Klassifikationsvektors $\underline{c}$ und einem Entscheidungsmodul zur finalen Klassenzuweisung auf Basis der aktuellen Beobachtung. Die Parameter des Klassifikationsmodells werden innerhalb der Trainingsphase auf Basis simulierter Beispieldatensätze $\mathbf{X}^S$ optimiert unter Minimierung des empirischen Fehlers e als Abweichung aus den geschätzten und wahren Klassenzugehörigkeiten y. Die hierzu benötigten Trainingsdaten werden für vordefinierte Ausfallszenarien unter Vorgabe des Netzzustands $\underline{u}_S^S$ sowie der Netztopologie (inkl. der Netzparameter) $\underline{u}_T^S$ erzeugt. In der Anwendungsphase wird das austrainierte Klassifikationsmodell mit den Messdaten $\mathbf{X}^P$ aus dem Realprozess konfrontiert. Netzzustand $\underline{u}_S^P$ und Netztopologie $\underline{u}_T^P$ sind hierbei nur noch teilweise bekannt bzw. weichen von den Simulationsannahmen ab. Infolgedessen kommt es zur Ausprägung unterschiedlicher Störgrößen $\underline{z}$, welche zu Beobachtungen mit starker Abweichung zur Trainingsmenge sowie mit Zugehörigkeit zu bekannten oder auch unbekannten Klassen führen können. Allgemein können folgende drei Störungstypen unterschieden werden:

- *Zufällige Fehler* $\underline{z}_R$: Ungenauigkeit in der Sensorerfassung bzw. Phasorschätzung (idealerweise weißes Rauschen),
- *Approximative Fehler* $\underline{z}_A$: vom Simulationsmodell abweichende Netzparameter, Ersatzmodellierung angrenzender Netze, Simulation einer begrenzten Anzahl an Netzzuständen, Simulation zeitlich konstanter Lasten und Erzeuger sowie

- *Unbekannte Ereignisse* $\underline{z}_E$: neue kritische oder unkritische Ereignisse im eigenen Netz bzw. in angrenzenden Netzen.

Im Fall zufälliger und approximativer Fehler sind die gestörten Beobachtungen ausschließlich bekannten Klassen zuzuweisen wohingegen im Fall unbekannter Ereignisse die gestörten Beobachtungen vom Klassifikator zurückzuweisen bzw. einer unbekannten Klasse zuzuordnen sind.

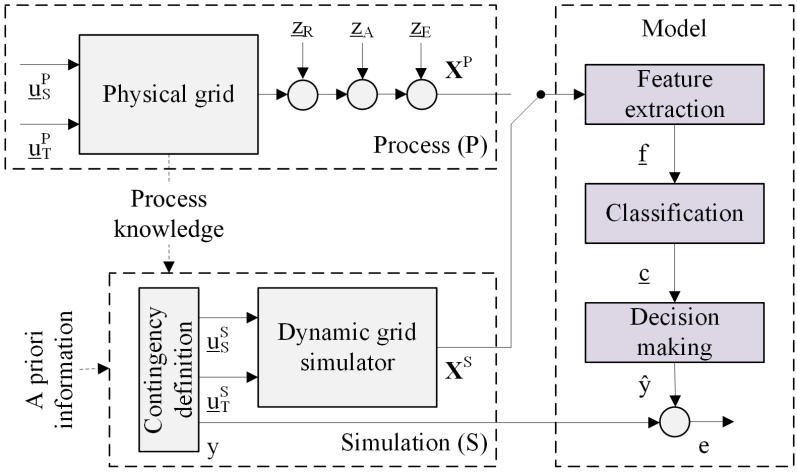


Bild 1: Allgemeines Systemmodell für die PMU-basierte Klassifikation kritischer Netzereignisse

4 Modellentwürfe

4.1 Konventioneller Klassifikator auf Basis rekurrenter neuronaler Netze

Ausgangsbasis für den Entwurf der Klassifikationsmodelle bildet die schematische Übersicht aus Bild 1, wonach Klassifikationsverfahren prinzipiell aus den Modulen Merkmalsextraktion, Klassifikation und Entscheidung zusammengesetzt werden können. Aufgrund der Vorteile bei der Verarbeitung hochdimensionaler Zeitreihen werden zur Merkmalsextraktion *gated recurrent units*

(GRU), einer speziellen Form rekurrenter neuronaler Netze, eingesetzt. Die Verwendung von *gateing*-Mechanismen ermöglicht dabei das Trainieren neuronaler Netze über eine hohe Anzahl an Zeitschritten sowie die Erfassung autoregressiver Zusammenhänge bei variierender Rhythmik und nicht-linearem Charakter. Weiterführende Literatur findet sich in [26]. Trotz des erhöhten Trainingsaufwands bietet der Einsatz neuronaler Netze den zusätzlichen Vorteil geringer Ausführungszeiten in der Anwendungsphase, da z.B. auf rechenintensive Merkmalsextraktionsverfahren (z.B. Wavelet-Dekomposition) verzichtet werden kann. Die allgemeine Struktur des konventionellen Klassifikationsmodells (*convGRU*) zeigt Bild 2.

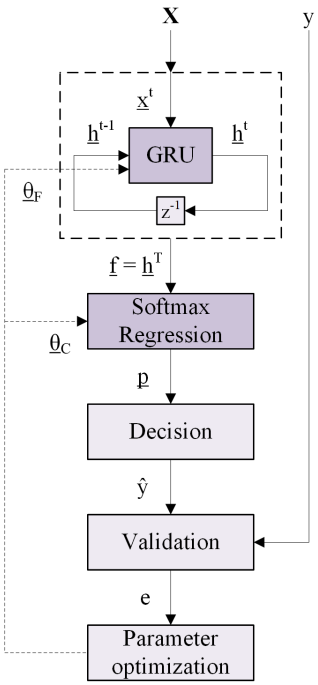


Bild 2: Konventionelles GRU-basiertes Klassifikationsmodell (*convGRU*)

Dabei werden simulierte bzw. gemessene PMU-Daten $\mathbf{X}$ bestehend aus p Messpunkten (z.B. Frequenzen oder Spannungsamplituden) über eine festgelegte Zeitspanne T dem Klassifikationsmodell zugeführt - siehe (1):

$$\boldsymbol{X} = [\underline{x}^t]_{t=0}^{t=T} \text{ mit } \boldsymbol{X} \in \mathbb{R}^{T \times p} \tag{1}$$

Innerhalb dieser Untersuchungen erfolgt die Identifikation und Lokalisierung des Netzereignisses über ein Einzelmodell, womit sich die Zielklasse y aus dem Ereignisort y_{Loc} und dem Ereignistyp y_{Type} zusammensetzt - siehe (2):

$$y = [y_{Loc}; y_{Type}] \tag{2}$$

In der Datenvorverarbeitung (in Bild 2 vernachlässigt) werden die PMU-Signale normalisiert und anschließend im GRU-Netzwerk weiterverarbeitet. Dabei wird zu jedem Zeitpunkt t ein Zustandsvektor $\underline{h}^t$ mit der Dimension q unter Verwendung des vorangegangenen Zustands $\underline{h}^{t-1}$ sowie der aktuellen Beobachtung $\underline{x}^t$ auf Basis der Aktivierungen innerhalb der GRU-Zelle ermittelt. Die Berechnung des Klassifikationsvektors $\underline{c}$ erfolgt auf Basis des Zustandsvektors des letzten Zeitschritts $\underline{h}^T$ bzw. Merkmalsvektors $\underline{f}$ unter Verwendung multinomialer, logistischer Regression (auch *softmax*-Regression genannt). Dabei wird mit Hilfe eines linearen Modells ein Zuweisungswert c für jede Klasse C aller K Klassen ermittelt und anschließend nach (3) über die *softmax*-Normierung in einen Wahrscheinlichkeitsvektor $\underline{p}$ überführt:

$$p_C = \frac{exp(c_C)}{\int_C exp(c_C)} \text{ mit } \underline{c} = [c_C]_{C=1}^{C=K} \text{ und } \underline{p} = [p_C]_{C=1}^{C=K} \tag{3}$$

Nach (4) erfolgt im Entscheidungsmodul die Klassenzuweisung auf Basis der aktuellen Beobachtung durch Bestimmung der maximalen Wahrscheinlichkeit:

$$\hat{y} = \arg\max_C \underline{p} \tag{4}$$

Die Modellparameter des GRU-Netzwerks $\underline{\theta}_F$ sowie des Klassifikators $\underline{\theta}_C$ werden klassisch mittels dem Prinzip der Fehlerrückführung (engl.: *backpropagation*) unter Minimierung des empirischen Fehlers e durch Berechnung der Kreuzentropie zwischen geschätzter und wahrer Wahrscheinlichkeitsverteilung der Klassenzugehörigkeiten erlernt.

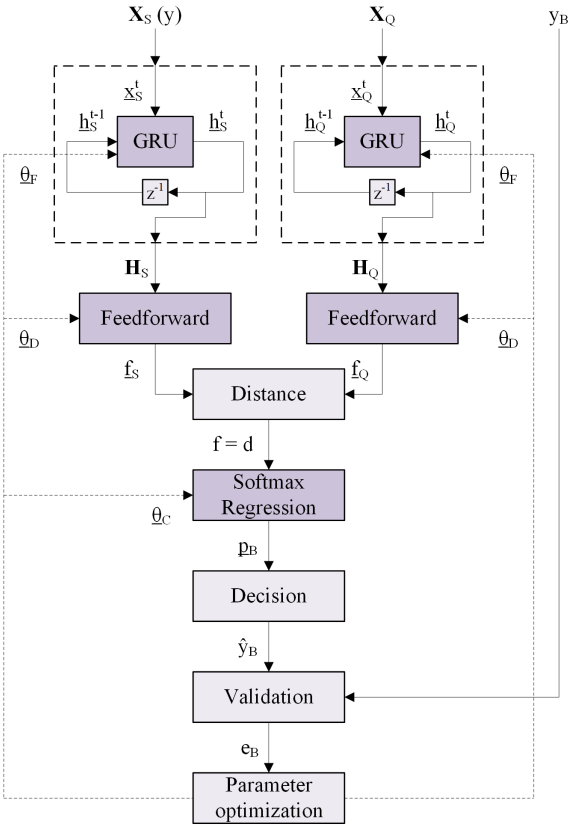


Bild 3: Klassifikationsmodell auf Basis siamesischer GRU (*siamGRU*)

4.2 Komparator-basierter Klassifikator auf Basis rekurrenter neuronaler Netze

Die Komparator-basierte Klassifikation ermöglicht im Gegensatz zum vorherigen Ansatz die Rückweisung nicht erlebter Beobachtungen. Hierzu werden je Klasse ein oder mehrere Stützinstanzen $\mathbf{X}_S$ definiert und mit einer Abfrageinstanz $\mathbf{X}_Q$ unbekannter Klassenzugehörigkeit verglichen. Die Klassenzuweisung von $\mathbf{X}_Q$ erfolgt auf Basis der maximalen Übereinstimmung zu einer der Stützinstanzen und kann über einen binären Klassifikator ermittelt werden.

Die Rückweisung einer Beobachtung erfolgt bei Nicht-Übereinstimmung der aktuellen Beobachtung mit allen Stützinstanzen. Einen entsprechenden Modellentwurf auf Basis siamesischer GRUs (*siamGRU*) zeigt Bild 3.
Dabei werden die normierten PMU-Eingangsdaten auf Stütz- und Abfrageseite über zwei identische GRU-Netzwerke verarbeitet zur Ermittlung der Zustandsmatrizen $\mathbf{H}_S$ bzw. $\mathbf{H}_Q$ - siehe (5):

$$\boldsymbol{H}_S = [\underline{h}_S^t]_{t=0}^{t=T} \text{ bzw. } \boldsymbol{H}_Q = [\underline{h}_Q^t]_{t=0}^{t=T} \text{ mit } \boldsymbol{H}_{S,Q} \in \mathbb{R}^{T \times q} \tag{5}$$

Über ein einschichtiges *feed-forward*-Netz mit Sigmoid-Aktivierung werden die Merkmalsvektoren $\underline{f}_S$ bzw. $\underline{f}_Q$ der Dimension r auf Basis der vektorisierten Zustandsmatrizen berechnet und mittels einer Distanzfunktion (hier: Manhattan-Distanz) miteinander verglichen - siehe (6) und (7):

$$\underline{f}_{S,Q} = \boldsymbol{\sigma}(\boldsymbol{W}^T vec(\boldsymbol{H}_{S,Q}) + \underline{b}) \text{ mit } \underline{f}_{S,Q} \in \mathbb{R}^r \tag{6}$$

$$d = \|\underline{f}_S - \underline{f}_Q\|_1 \tag{7}$$

Das *feed-forward*-Netz dient vorrangig der Dimensionsreduktion sowie der Ermittlung der relevanten Merkmale zur Unterscheidung der Zustandsmatrizen. Der Distanzwert d wird anschließend einem binären Klassifikator übergeben zur Ermittlung des Wahrscheinlichkeitsvektors $\underline{p}_B$ auf Basis der *softmax*-Regression sowie der Klassenzuweisung $\hat{y}_B$ im Entscheidungsmodul. Über eine geeignete Initialisierung das Klassifikators wird dabei sichergestellt das nach (8) gilt:

$$\hat{y}_B = \begin{cases} 0 \text{ (andere Klasse wie } y) & \text{wenn: } d > d_{Threshold} \\ 1 \text{ (gleiche Klasse wie } y) & \text{wenn: } d \leq d_{Threshold} \end{cases} \tag{8}$$

Die Grenzdistanz $d_{Threshold}$ folgt aus den erlernten Gewichts- und Biaswerten des linearen Klassifikationsmodells. Ähnlich zum vorherigen Ansatz werden die Parameter der GRU-Netzwerke $\underline{\theta}_F$, der *feed-forward*-Netze $\underline{\theta}_D$ sowie des

binären Klassifikators $\underline{\theta}_C$ unter Minimierung des empirischen Fehlers e durch Berechnung der binären Kreuzentropie erlernt. Infolge der siamesischen Modellarchitektur werden die Gewichte der GRU-Netzwerke sowie *feed-forward*-Netze gleichermaßen auf Stütz- und Abfrageseite aktualisiert. Eine zentrale Rolle spielt hierbei die Bereitstellung eines balancierten Trainingsdatensatzes bestehend aus repräsentativen Stütz- und Abfragebeispielen, welche dem Klassifikationsmodell paarweise übergeben werden.

5 Erste Ergebnisse und Diskussion

5.1 Datenbasis und Hyperparameter

Zur Bereitstellung der Trainingsdaten wird ein synthetisches Übertragungsnetz in der 400kV-Ebene simuliert bestehend aus 33 Netzknoten (Stationen). Hierzu werden RMS-Simulationen mit einer Schrittweite von 1 ms durchgeführt zur Nachstellung von Generator- und Leitungsausfällen an allen Netzpunkten. Die Simulationsdaten umfassen insgesamt 130 Ausfallszenarien für vier unterschiedliche Netzbetriebspunkte bzw. Netzzustände. Die PMU-Signale werden anschließend als gleitende Mittelwerte mit einer zeitlichen Auflösung von 100 ms aus den Simulationsergebnissen extrahiert. In Übereinstimmung mit vorliegenden Rechercheergebnissen beschränken sich die Eingangssignale auf Frequenzwerte und Spannungsamplituden zur Identifikation und Lokalisierung der Netzereignisse.
Zur Bewertung beider Klassifikationsmodelle *convGRU* und *siamGRU* im Umgang mit approximativen Fehlern $\underline{z}_A$ sowie unbekannten Ereignissen $\underline{z}_E$ werden für die Trainings-, Validierungs- und Testdaten unterschiedliche Ausfallszenarien erzeugt. Die Störgrößen werden dabei ausschließlich in den Testdaten eingespeist und sind somit zum Zeitpunkt des Trainings unbekannt. Die entsprechenden Ergebnisse werden in den Abschnitten 5.2 und 5.3 beschrieben. Zufällige Fehler $\underline{z}_R$ werden zunächst nicht weiter betrachtet, da ihr Einfluss auf die Klassifikationsentscheidungen als eher gering angenommen wird. Die gewählten Hyperparameter beider Modelle zeigt nachfolgend Tabelle 1. Die Optimierung erfolgt in beiden Fällen auf Basis von *stochastic gradient descent*.

Tabelle 1: Gewählte Hyperparameter der Klassifikationsmodelle *convGRU* und *siamGRU*

Modell	Hyperparameter	Wert
convGRU	Anzahl Neuronen GRU-Zelle q	8
convGRU	Lernrate	0.002
convGRU	Schrittweitenregelung	*RMSprop*
convGRU	Batchgröße	32
siamGRU	Anzahl Stützvektoren je Klasse	1
siamGRU	Anzahl Neuronen GRU-Zelle q	10
siamGRU	Anzahl Neuronen FF-Netz r	5
siamGRU	Lernrate	0.002
siamGRU	Schrittweitenregelung	*RMSprop*
siamGRU	Batchgröße	32

5.2 Ergebnisse im Umgang mit approximativen Fehlern

Approximative Fehler werden innerhalb dieser Untersuchungen durch Variation des Betriebs- bzw. Netzzustands abgebildet. Dabei beziehen sich die Trainingsdaten auf zwei Netzzustände und die Testdaten auf einen davon abweichenden Betriebspunkt. Ziel der Klassifikation sind die folgenden Ausfallszenarien:

- Ausfälle von Dampfkraftwerk-Generatoren (DKW) an den Stationen 1D1 und 3D2,
- Ausfälle von Großkraftwerk-Generatoren (GKW) an den Stationen 4D2 und 6D1 sowie
- Ausfälle der Leitungen 11, 15, 26 und 32.

Nach Tabelle 2 weißen beide Modelle bei der Klassifikation der Trainings- und Validierungsbeispiele sehr geringe Fehlerraten über alle Ausfallszenarien bzw. Klassen auf. Zusätzlich aufgeführt sind die Ergebnisse der binären Klassifikation des *siamGRU*-Modells zur korrekten Unterscheidung der Stütz- und Abfrageinstanzen. Im Testfall kommt es zu einem starken Einbruch der Klassifikationsgenaugigkeiten um ca. 22.50 % (*convGRU*-Modell) bzw. 11.85 %

(*siamGRU*-Modell) infolge des veränderten Netzzustands bzw. Betriebspunkts der Testdaten.

Eine detaillierte Betrachtung der Testergebnisse liefern die Konfusionsmatrizen in Bild 4. Zu beachten ist hierbei die zusätzliche Klasse *unknown* zur Rückweisung von Beobachtungen im Fall des *siamGRU*-Modells. Beide Modelle zeigen relativ ähnliche Klassifikationsentscheidungen: eine prinzipiell korrekte Unterscheidung des Ausfalltyps ist erkennbar bei jedoch mangelhafter Erkennung des Ausfallorts. Beide Klassifikationsansätze sind demnach nicht in der Lage bei Veränderungen des Netzzustands korrekte Klassifikationsentscheidungen zu treffen. Eine Ausweitung der Trainingsdatenbasis zur Erhöhung der Generalisierungsfähigkeit der Klassifikatoren könnte dem entgegenwirken und ist in weiterführenden Arbeiten zu untersuchen. Darüber ist ein statistischer Vergleich zwischen verschiedenen Netzzuständen sinnvoll zur gezielten Auswahl repräsentativer Testfälle.

Tabelle 2: Ergebnisse der Klassifikationsgenauigkeiten der *convGRU* und *siamGRU* Klassifikatoren im Fall approximativer Fehler

Modell	Training	Validierung	Test
convGRU	100.00 %	99.38 %	77.50 %
siamGRU	100.00 %	100.00 %	88.15 %
siamGRU (binär)	100.00 %	100.00 %	81.25 %

5.3 Ergebnisse im Umgang mit unbekannten Ereignissen

Im Fall unbekannter Ereignisse bleibt der Netzzustand bzw. Betriebspunkt zwischen Trainings- und Testdaten unverändert. Trainings- und Validierungsdaten entsprechen somit den Untersuchungen aus Abschnitt 5.2. Der Testdatensatz setzt sich demgegenüber aus Generator- und Leitungsausfälle mit unbekanntem Ausfallort zusammen:

- Unbekannte Ausfälle von Dampfkraftwerk-Generatoren (DKW) an den Stationen 2D1 und 4D2 sowie
- Unbekannte Ausfälle der Leitungen 12 und 31.

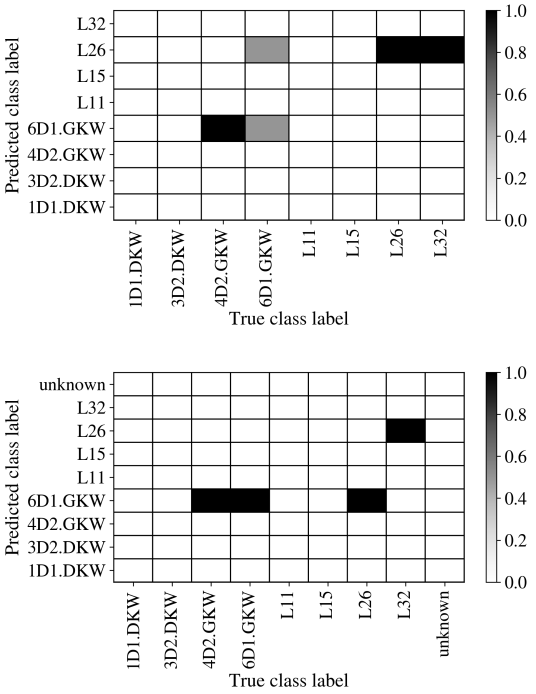


Bild 4: Konfusionsmatrix (Testphase) für das *convGRU*-Modell (oben) bzw. *siamGRU*-Modell (unten) im Fall approximativer Fehler

Analog zum vorherigen Abschnitt fasst Tabelle 3 die Ergebnisse beider Modelle für die Trainings-, Validierungs- und Testdaten zusammen. Infolge der Anwesenheit unbekannter Klassen ist eine sinnvolle Auswertung der Testbeispiele auf Basis der Klassifikationsgenauigkeit nur im binären Fall des *siamGRU*-Modells möglich.

Mehr Aufschluss bietet die Auswertung der Konfusionsmatrizen in Bild 5 für beide Ansätze. Im Fall des *convGRU*-Modells werden sowohl die unbekannten DKW-Ausfälle an den Stationen 2D1 und 4D2 als auch die unbekannten Leitungsausfälle L12 und L31 hauptsächlich den GKW-Ausfällen an den Stationen 4D2 und 6D1 zugewiesen. Im Fall des *siamGRU*-Modells erfolgt zumindest teilweise die Rückweisung einiger Beispiele der unbekannten Klasse

2D1.DKW wohingegen die restlichen Beobachtungen fälschlicherweise bekannten Klassen zugeordnet werden. Im Vergleich zum *convGRU*-Modell ist jedoch eine stärkere Trennung der Ausfalltypen (Generator- und Leitungsausfälle) erkennbar und weißt auf eine bessere Merkmalsbildung hin. Dies kann auf die siamesische Modellstruktur zurückgeführt werden; erfordert jedoch die Validierung anhand weitere Datensätze und Kombination verschiedener Ausfallszenarien. Der geringen Rückweisungsrate im Fall des *siamGRU*-Modells kann wie auch in Abschnitt 5.2 durch Vergrößerung der Trainingsdatenbasis oder auch der Anzahl an Stützbeispielen zum Erlernen besserer Merkmalsrepräsentationen entgegengewirkt werden.

Tabelle 3: Ergebnisse der Klassifikationsgenauigkeiten der *convGRU* und *siamGRU* Klassifikatoren im Fall unbekannter Ereignisse

Modell	Training	Validierung	Test
convGRU	100.00 %	100.00 %	-
siamGRU	100.00 %	99.92 %	-
siamGRU (binär)	100.00 %	99.46 %	88.28 %

6 Zusammenfassung und Ausblick

Innerhalb dieses Beitrags werden erste Ansätze zur robusten Identifikation und Lokalisierung kritischer Netzbetriebssituationen auf Basis von PMU-Daten vorgestellt und diskutiert. Über die Einführung von Störgrößen in Form zufälliger Fehler, approximativer Fehler sowie unbekannter Ereignisse wird der Klassifikationsprozess unter Berücksichtigung von Abweichungen zwischen Simulations- und Messdaten formal definiert. Die Auswirkungen der Störgrößen auf die Klassifikationsentscheidungen werden für zwei Klassifikationsansätze auf Basis rekurrenter, neuronaler Netze demonstriert. Sowohl im Fall approximativer Fehler als auch unbekannter Ereignisse können starke Einbrüche der Klassifikationsgenauigkeiten bzw. eine hohe Anzahl falscher Klassenzuweisungen nicht erlebter Beobachtungen festgestellt werden.

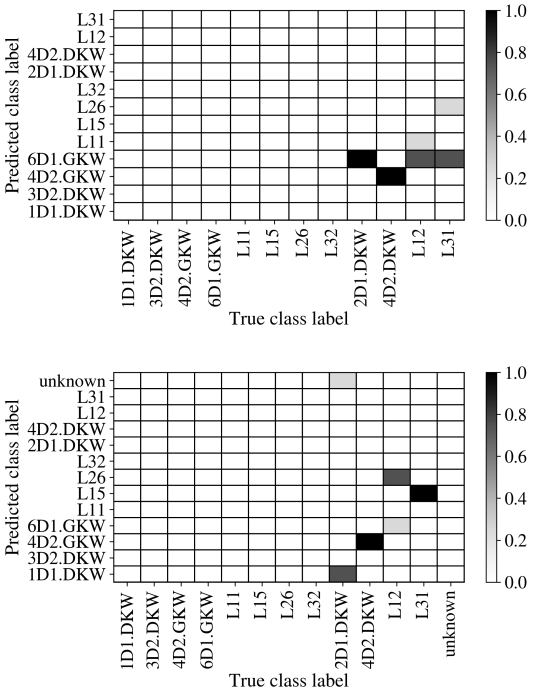


Bild 5: Konfusionsmatrix (Testphase) für das *convGRU*-Modell (oben) bzw. *siamRU*-Modell (unten) im Fall unbekannter Ereignisse

Der Einsatz Komparator-basierter Klassifikatoren (z.B. siamesische neuronale Netze) erscheint dabei aufgrund der inhärenten Rückweisungsoption sowie der hohen Generalisierungsfähigkeit auf Basis weniger Beispiele als vielversprechender Modellierungsansatz und ist bisher wenig erforscht für den Einsatz zur Klassifikation von PMU-Daten. Eine abschließende Beurteilung erfordert weiterführende Untersuchungen unter Einbezeihung weiterer Testszenarien sowie alternativer Ansätze wie Drillings- oder Vierlings-Netze, die Verwendung kontrastiver Verlustfunktionen bzw. der Ersatz von Distanzmetriken mit parametrischen Funktionen.

Literatur

[1] North American Electric Reliability Corporation, "Real-time application of synchrophasors for improving reliability," 2010. Available at `https://www.smartgrid.gov/document/real_time_application_synchrophasors_improving_reliability`.

[2] C. Brosinsky, A. Kummerow, A. Naumann, A. Kronig, S. Balischewski, and D. Westermann, "A new development platform for the next generation of power system control center functionalities for hybrid ac-hvdc transmission systems," in *2017 IEEE Power & Energy Society General Meeting*, (Piscataway, NJ), pp. 1–5, IEEE, 2017.

[3] A. Kummerow, S. Nicolai, and P. Bretschneider, "Ensemble approach for automated extraction of critical events from mixed historical pmu data sets," in *2018 IEEE Power & Energy Society General Meeting (PESGM)*, pp. 1–5.

[4] A. Kummerow, S. Nicolai, and P. Bretschneider, "Outlier detection methods for uncovering of critical events in historical phasor measurement records," *E3S Web of Conferences*, vol. 64, no. 1, 2018.

[5] A. Kumar, S. S. Negi, N. Kishor, and K. Uhlen, "Signal processing and classification of synchro-phasor data," in *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, pp. 1–6.

[6] S. A. Lavand, G. R. Gajjar, S. A. Soman, and R. Gajbhiye, "Mining spatial frequency time series data for event detection in power systems," in *13th IET International Conference on Developments in Power System Protection (DPSP 2016)*, IET conference publications, p. 6, Curran Associates Inc, 2016.

[7] J. Ning and W. Gao, "Multi-feature extraction for power system disturbances by wavelet transform and fractal analysis," in *Energy Society General Meeting*, pp. 1–7.

[8] M. Biswal, Y. Hao, P. Chen, S. Brahma, H. Cao, and P. de Leon, "Signal features for classification of power system disturbances using pmu data," in *19th Power Systems Computation Conference*, pp. 1–7, IEEE, 2016.

[9] D. Nguyen, R. Barella, S. A. Wallace, X. Zhao, and X. Liang, "Smart grid line event classification using supervised learning over pmu data streams," in *2015 Sixth International Green and Sustainable Computing Conference*, pp. 1–8, IEEE, 2015.

[10] Z. Wang, Y. Zhang, and J. Zhang, "Principal components fault location based on wams/pmu measure system," in *Energy Society General Meeting*, pp. 1–5.

[11] S. Fries and C.-P. Rückemann, eds., *Finding Needles in a Haystack: Line Event Detection on Smart Grid PMU Data Streams: ENERGY 2016 The Sixth International Conference on Smart Grids, Green Communications and IT Energy-Aware Technologies : June 26-30, 2016, Lisbon, Portugal*. Red Hook, NY: Curran Associates Inc, 2016.

[12] A. T. Mathew and M. N. Aravind, "Pmu based disturbance analysis and fault localization of a large grid using wavelets and list processing," in *Proceedings of the 2016 IEEE Region 10 Conference (TENCON)*, pp. 879–883, IEEE, 2016.

[13] S. S. Negi, N. Kishor, K. Uhlen, and R. Negi, "Event detection and its signal characterization in pmu data stream," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 3108–3118, 2017.

[14] A. K. Singh and M. Fozdar, "Supervisory framework for event detection and classification using wavelet transform: Department of electrical engineering, malaysia national institute of technology, jaipur, india," in *Institute of Electrical and Electronics Engineers, Power & Energy Society et al. 2017 – IEEE PES General Meeting*, pp. 1–5.

[15] Pedro Emanuel Almeida Cardoso, *Deep Learning Applied to PMU Data in Power Systems*. PhD thesis, Universidade do Porto, Porto, Portugal, July 2017.

[16] A. Haque, L. Khan, M. Baron, B. Thuraisingham, and C. Aggarwal, "Efficient handling of concept drift and concept evolution over stream data," in *ICDE 2016 Conference*, pp. 481–492, IEEE, 2016.

[17] M. M. Masud, Q. Chen, L. Khan, C. Aggarwal, J. Gao, J. Han, and B. Thuraisingham, "Addressing concept-evolution in concept-drifting data streams," in *2010 IEEE International Conference on Data Mining*, pp. 929–934, IEEE, 2010.

[18] Xin Mu, Feida Zhu, Juan Du, Ee-Peng Lim, and Zhi-Hua Zhou, "Streaming classification with emerging new class by class matrix sketching," in *AAAI*, 2017.

[19] A. Bendale and T. Boult, "Towards Open Set Deep Networks," *arXiv e-prints*, p. arXiv:1511.06233, Nov 2015.

[20] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, "Toward open set recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 7, pp. 1757–1772, 2013.

[21] L. Shu, H. Xu, and B. Liu, "DOC: Deep Open Classification of Text Documents," *arXiv e-prints*, p. arXiv:1709.08716, Sep 2017.

[22] Y. Yu, W.-Y. Qu, N. Li, and Z. Guo, "Open-Category Classification by Adversarial Sample Generation," *arXiv e-prints*, p. arXiv:1705.08722, May 2017.

[23] Gregory R. Koch, "Siamese neural networks for one-shot image recognition," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

[24] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical Networks for Few-shot Learning," *arXiv e-prints*, p. arXiv:1703.05175, Mar 2017.

[25] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching Networks for One Shot Learning," *arXiv e-prints*, p. arXiv:1606.04080, Jun 2016.

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, Massachusetts and London, England: MIT Press, 2016.

Variablenreduktion für Surrogat-Modell basierte Optimierung

Frederik Rehbach, Lorenzo Gentile, Thomas Bartz-Beielstein

Institut für Data Science, Engineering, and Analytics, TH Köln
Steinmüllerallee 1, 51643 Gummersbach
E-Mail: firstname.lastname@th-koeln.de

Kurzfassung

Eine Standardmethode zur effizienten Optimierung von rechenintensiven Problemen ist die *Surrogat-Modell basierte Optimierung* (SMBO). Allerdings steigt bei vielen Surrogat Modellen mit zunehmender Dimensionalität der Rechenaufwand stark an. SMBO ist daher oft nur sehr ineffizient auf hochdimensionalen Problemen anwendbar. In diesem Beitrag wird der Einsatz von Regularisierungsverfahren (Shrinkage) untersucht, um die Dimensionalität des Suchraums zu reduzieren. Wir schlagen einen Hybrid-Algorithmus vor: Regularisierte-Surrogat-Optimierung (RSO). Experimente, die auf künstlichen Testfunktionen und industriellen Anwendungen durchgeführt wurden, zeigen, dass der RSO-Ansatz die Dimensionalität der Modelle erfolgreich reduziert und somit erheblich weniger Rechenzeit benötigt. Gleichzeitig liefert der Algorithmus Ergebnisse, die eine ähnlich hohe Güte aufweisen wie das nicht reduzierte Modell.

1 Einführung

Industrielle Probleme besitzen Charakteristiken, die sie von künstlich erzeugten Testfunktionen unterscheiden. Unter anderem sind sie oft hoch-dimensional und teuer zu evaluieren. Ein Standardverfahren für die Optimierung von teuren

Problemen ist Surrogat-Modell basierte Optimierung (SMBO). Eine detaillierte Erklärung zum SMBO Verfahren ist in [1] gegeben.

Im Rahmen laufender Kooperationen mit Industriepartnern reicht es meist nicht einfach nur eine einzelne sehr gute Lösung für ein Problem zu geben. Um auf Akzeptanz in der Industrie zu stoßen, ist es vielmehr notwendig ein Prozessverständnis entwickeln zu können, welches diese Lösung bestätigt und begründet. In dieser Arbeit werden daher zwei Forschungsfragen betrachtet:

(Q-1): Wie kann die Interpretierbarkeit von SMBO in industriellen Anwendungen erhöht werden, um offenere Akzeptanz zu erreichen?

(Q-2): Wie kann SMBO effizient auf hochdimensionalen Problemen eingesetzt werden?

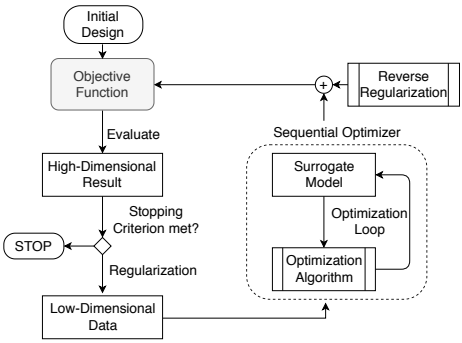


Bild 1: Ablaufdiagramm des RSO Algorithmus

2 Regularisierte-Surrogat-Optimierung

Zur effizienten Nutzung von SMBO auf hochdimensionalen Problemen schlagen wir einen zweischrittigen Hybrid-Algorithmus vor: Regularisierte-Surrogat-Optimierung, siehe Abb. 1. Ein initiales Design wird auf der Zielfunktion ausgewertet. In jeder Iteration werden im Regularisierungsschritt die wichtigsten Features des hochdimensionalen Optimierungsproblems erkannt und die Daten für den Optimierungsschritt reduziert. Der Optimierer schlägt

ein auf dem Surrogat gefundenes Optimum als neuen Punkt zur Evaluierung vor. Die fehlenden Features des Punktes werden aufgefüllt, um wieder die ursprüngliche Dimensionalität zu erlangen. Der neue Punkt wird auf der Zielfunktion ausgewertet und die nächste Iteration des Algorithmus beginnt. Es wurden verschiedene Spacefilling-Verfahren getestet, ein Auffüllen mit Zufallszahlen erzeugte hierbei aber die besten Ergebnisse. Als Regularisierungsmethoden werden LASSO (Least Absolute Shrinkage and Selection Operator) [7] und Random Forests (RF) [8] verglichen. RSO ist anwendbar mit verschiedenen Regularisierungsmethoden und Surrogat Modellen. In dieser Arbeit kam Kriging als Surrogat zum Einsatz.

3 Experimente

Das erste Industrie-Problem beschäftigt sich mit der Optimierung von elektrostatischen Staubabscheidern (englisch: electrostatic precipitator - ESP), siehe [6]. Als zweite Testfunktion betrachten wir das Sandwich-Structured Composite Plate Design Problem (SCPD). Ein Sandwich Panel ist ein Verbundwerkstoff der aus mehreren Schichten verschiedener Materialien zusammengesetzt wird. Durch die Auswahl der jeweiligen Materialien und deren Eigenschaften wird das Verhalten des Werkstoffes in einer Computersimulation verändert. Ziel der Optimierung ist ein Material zu erzeugen, dass möglichst viel Kraft aufnehmen kann.

Für eine tiefergehende statistische Analyse wurde zusätzlich eine Reihe von künstlichen Testfunktionen verwendet. In allen Testfunktionen ist ein Teil der Variablen wichtig, ein anderer Teil unwichtig für die Optimierung. Tabelle 1 zeigt eine Übersicht der Testfunktionen.

Als Vergleich zu RSO wird eine Standard Implementation von SMBO mit Kriging herangezogen. Da auf den teuren Industrieproblemen nur sehr wenige Evaluierungen möglich sind, erhalten alle Algorithmen ein Budget von 200 Evaluierungen pro Lauf. Zur statistischen Auswertung werden alle Algorithmen je 20 mal wiederholt.

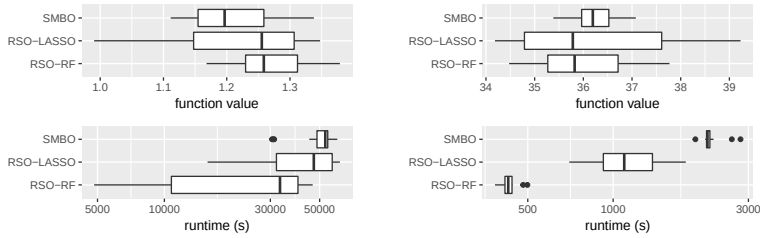
Tabelle 1: Übersicht über alle Testfunktionen und deren Dimensionalität. Angegeben ist die Dimensionalität des Problems, sowie die Anzahl der wichtigen, weniger wichtigen sowie komplett irrelevanten Variablen. Für das ESP Problem sind die wichtigen Variablen nicht bekannt.

Problem	Dimensionen	wichtig	relevant	irrelevant
linketal06dec [2]	10	2	4	4
linketal06sin [2]	10	2	0	8
moon10hd [3]	20	5	10	5
moon10hdc1 [3]	20	5	0	15
moon10hdc2 [3]	20	5	0	15
moon10hdc3 [3]	20	5	0	15
oakoh04 [5]	15	5	5	5
morretal06 [4]	30	2	0	28
ESP-Problem	49	?	?	?
SCPD-Problem	19	2	4	13

4 Ergebnisse und Diskussion

Die Ergebnisse der Algorithmen auf den beiden Industrieproblemen sind in Bild 2a und 2b veranschaulicht. Die Ergebnisse zeigen, dass auf beiden Problemen ein Wechsel zu RSO keinen Qualitätsverlust der gefundenen Lösung verursacht. Gleichzeitig ist aber insbesondere auf dem SCPD-Problem zu sehen, dass die nötige Rechenzeit für den Algorithmus drastisch reduziert werden konnte. Die RSO-RF Variante scheint eine striktere Feature-Selektion vorzunehmen als LASSO. Daher wird in dieser Variante auch weniger Rechenzeit benötigt.

Die Ergebnisse aller Experimente sind in Tabelle 2 zusammengefasst. Obwohl auf den zwei Industrieproblemen kein Qualitätsunterschied der gefundenen Lösungen ersichtlich war, ist dies auf manchen künstlichen Testfunktionen der Fall. Unter anderem haben diese Funktionen zu komplexe Zusammenhänge zwischen den Variablen, sodass wichtige und irrelevante Variablen nicht korrekt voneinander getrennt werden können. Insbesondere die RF Variante, die mehr Variablen ausschließt, ist daher von einem Qualitätsverlust betrof-



(a) Ergebnisse auf dem ESP-Problem (b) Ergebnisse auf dem SCPD-Problem

Bild 2: Der jeweils obere Boxplot zeigt die Qualität der besten gefunden Lösung der Optimierungsläufe mit den links beschriebenen Algorithmen. Niedrigere Werte sind besser. Der untere Plot zeigt die vom jeweiligen Algorithmus benötigte Rechenzeit in Sekunden.

Tabelle 2: Übersicht der Ergebnisse aller Algorithmen auf allen Testfunktionen. Die aufgezeigten Zahlen ergeben sich aus einem rangbasierten, paarweisen Test. Der beste Rang (1) ist jeweils fett-gedruckt.

	ESP	SCPD	linketal06dec	linketal06sin	moon10hd	moon10hdc1	moon10hdc2	moon10hdc3	oakoh04	morretal06	Mean-Rank
SMBO	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
RSO-LASSO	**1**	**1**	**1**	**1**	**1**	2	**1**	2	2	2	1.4
RSO-RF	**1**	**1**	2	**1**	2	2	2	2	2	2	1.7

fen. Dennoch ist auf allen Funktionen eine deutliche Reduktion der Algorithmuslaufzeit, ähnlich der Ergebnisse vom SCPD-Problem zu messen. Abschließend möchten wir wieder zu den anfangs definierten Forschungsfragen zurückkehren.

(Q-1): In der Industrie ist es oft sehr schwer komplexe Surrogat Modelle umzusetzen, da sie zu wenig interpretierbare Ergebnisse liefern und daher keine Akzeptanz finden. Dahingegen sind die durch RSO gebauten niedriger dimensionalen Modelle einfacher zu erklären und spiegeln (im Falle des ESP-Problems) beispielsweise die Vermutungen der Ingenieure wider.

(Q-2): Die Ergebnisse zeigen, dass die benötigte Algorithmuslaufzeit, insbesondere die Rechenzeit zum Modellieren der Surrogate, durch RSO stark reduziert werden kann. Gleichzeitig kann RSO auf den meisten Testproblemen Ergebnisse von äquivalenter Qualität finden. Hinzu kommt, dass RSO auch bis zu einer sehr hohen Dimensionalität skaliert werden kann, wohingegen SMBO mit steigender Dimensionalität schlechtere Ergebnisse liefern wird. Ebenso kann die reduzierte Modellierungsdauer RSO für Grenzfälle anwendbar machen, in denen sich SMBO nicht gelohnt hätte.

Literatur

[1] T. Bartz-Beielstein und M. Zaefferer „Model-based methods for continuous and discrete global optimization“. Applied Soft Computing, Volume 55, Pages 154-167, 2017.

[2] C. Linkletter, D. Bingham, N. Hengartner, D. Higdon and K.Ye„Variable selection for Gaussian process models in computer experiments“. Technometrics, Volume 48, Pages 478-490, 2006.

[3] H. Moon „Design and analysis of computer experiments for screening input variables“. PhD Thesis - The Ohio State University, 2010.

[4] M. Morris, L. Moore and M. McKay „Sampling plans based on balanced incomplete block designs for evaluating the importance of computer model inputs“. Journal of Statistical Planning and Inference, Volume 136, Pages 3203-3220, 2006.

[5] J. Oakley und A. O'Hagan „Probabilistic sensitivity analysis of complex models: a Bayesian approach“. Journal of the Royal Statistical Society: Series B (Statistical Methodology), Volume 66, Pages 751-769, 2004.

[6] F.Rehbach, M. Zaefferer, J. Stork und T. Bartz-Beielstein„Comparison of parallel surrogate-assisted optimization approaches“. Proceedings of the Genetic and Evolutionary Computation Conference 2018.

[7] R. Tibshirani „Regression shrinkage and selection via the lasso“. Journal of the Royal Statistical Society. Series B (Methodological), 1996, Pages 267-288.

[8] L. Breiman „Random forests“. Machine learning, Volume 45, Pages 5-32, 2001.

Multi-Modell-Ansatz für die Nachjustierung von Black-Box-Reglern in der Praxis

Christian Dengler, Boris Lohmann

Technische Universität München
Boltzmannstr. 15, Garching bei München
E-Mail: {c.dengler, lohmann}@tum.de

1 Einführung

Aufgrund der stetig steigenden Rechenkapazitäten werden Reglerentwürfe in Form von nichtlinearen, parametrierten Funktionsapproximatoren immer attraktiver gegenüber einem analytischen Reglerentwurf. Diese Regler können z.B. über Policy-Gradient-Algorithmen (z.B. Proximal Policy Optimization [1]) oder Imitationslernen [2] trainiert werden. Während die Regler in der Theorie auch am realen System trainiert werden können, werden in der Praxis Modelle zum trainieren verwendet. Zum einen um Beschädigungen am realen System zu vermeiden, zum anderen aufgrund der hohen Anzahl an benötigten Trainingsdaten.

Die Übertragung eines in der Simulation trainierten Reglers auf das reale System liefert jedoch oft nicht die gewünschte Regelgüte aufgrund von Modellfehlern. Die Autoren von [3] vermuten hierfür, dass hochgradig nichtlineare Funktionsapproximatoren wie z.B. Neuronale Netze, sich stark auf das Trainingsmodell und die Simulationsumgebung spezialisieren. Um die Übertragbarkeit des Reglers vom Modell in die Realität zu verbessern, können die Regler über Methoden des Reinforcement Learning mittels Modellen mit variablen Parametern trainiert werden. Einige Methoden basieren auf einem Min-Max Ansatz, welcher versucht die Regelgüte der jeweils schlechtesten Modellparameter zu minimieren, z.B. [4]. Andere Ansätze zielen darauf aus die

mittlere Regelgüte über alle Parameter zu maximieren (z.B. [5, 6]). Ähnliche Ansätze für einen Entwurf über Imitationslernen sind den Autoren bisher nicht bekannt.

Der Ansatz in diesem Beitrag ist, eine Nachjustierung des Reglers am System zu erlauben, wodurch die Regelgüte sowie das Verhalten des Reglers für bestimmte Modellparameter angepasst werden kann. Hierdurch soll ein vorhandener Regler am instabilen System inverses Pendel auf Rädern verbessert werden, da dieser zu Schwingungen bei der Stabilisierung neigt.

Im folgenden Abschnitt wird das Trainieren eines Reglers über Imitationslernen sowie die verwendeten Reglerstrukturen genauer erläutert. Danach folgt in Abschnitt 3 eine empirische Untersuchung an einem einfachen Simulationsbeispiel und im Abschnitt 4 wird die Methode am realen System inverses Pendel auf Rädern angewandt.

2 Multi-Modell-Ansatz für Imitationslernen

Wir gehen von einem zeitdiskreten Modell $\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t; \mathbf{p}_{\text{model}})$ einer realen Strecke aus. Dabei bezeichnet $\mathbf{x}_t$ den Zustandsvektor zum diskreten Zeitpunkt t, $\mathbf{u}_t$ den Vektor der Stellgrößen und $\mathbf{p}_{\text{model}}$ ein Vektor mit konstanten, evt. nicht genau bekannten Systemparametern, z.B. Masse, Reibkoeffizienten etc.

Das zugrunde liegende Vorgehen ist bereits in [7] aufgeführt und ist eine vereinfachte Variante der Reglersynthese aus [8], wobei hier alle Modellparameter $\mathbf{p}_{\text{model}}$ bekannt und konstant angenommen wurden. Um einen annähernd optimalen Regler in Form einer Black-Box-Funktion für bekannte Modellparameter $\mathbf{p}_{\text{model}}$ über Imitationslernen zu entwerfen, kann man Trainingsdaten über eine Trajektorienoptimierung erzeugen. So erhält man optimale Stellgrößenverläufe $\mathbf{U}^* = \left[\mathbf{u}_0, \ldots, \mathbf{u}_{T-1}\right]$ und Zustandstrajektorien $\mathbf{X}^* = \left[\mathbf{x}_0, \ldots, \mathbf{x}_T\right]$.

Hierzu definiert man eine Kostenfunktion $c(\mathbf{x},\mathbf{u},t)$ und löst mehrfach für unterschiedliche Startzustände $\mathbf{x}_0$ das Optimierungsproblem

$$(\mathbf{X}^*,\mathbf{U}^*) = \underset{\mathbf{X},\mathbf{U}}{\operatorname{argmin}} \sum_{t=0}^{T-1} c(\mathbf{x}_t,\mathbf{u}_t,t) + c_T(\mathbf{x}_T) \tag{1a}$$

$$s.t. \mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t,\mathbf{u}_t;\mathbf{p}_{\text{model}}) \tag{1b}$$

$$\mathbf{x}_{t=0} = \mathbf{x}_0 \tag{1c}$$

$$\mathbf{eq}(\mathbf{X},\mathbf{U}) = \mathbf{0} \tag{1d}$$

$$\mathbf{ieq}(\mathbf{X},\mathbf{U}) \leq \mathbf{0}. \tag{1e}$$

Die Nebenbedingungen (1d) und (1e) können z.B. verwendet werden um Stellgrößenbeschränkungen einzuhalten oder einen gewünschten Endzustand $\mathbf{x}_T$ zu erzwingen. Im Anschluss an die Trajektorienoptimierungen wird eine parametrierte Funktion $\mathbf{g}(\mathbf{x};\boldsymbol{\Theta})$, z.B. ein Neuronales Netz, trainiert, die optimalen Stellgrößen in den jeweiligen Zuständen möglichst gut wiederzugeben sowie dazwischen zu interpolieren. Die Parameter des Reglers $\boldsymbol{\Theta}$ werden durch minimieren von $\sum_i ||\mathbf{u}_i^* - \mathbf{g}(\mathbf{x}_i^*;\boldsymbol{\Theta})||$ über Gradientenabstieg optimiert. Die Tupel $\mathbf{x}_t^*,\mathbf{u}_t^*$ sind dabei aufgrund der Bedingungen (1d) sowie des endlichen Horizonts T u.U. nicht eindeutig, d.h. aus verschiedenen optimalen Trajektorien können unterschiedliche $\mathbf{u}_t^*$ für das gleiche $\mathbf{x}_t^*$ entstehen.

Im Falle von unbekannten Modellparametern $\mathbf{p}$, wobei $\mathbf{p}$ aus Einträgen von $\mathbf{p}_{\text{model}}$ besteht, werden für die folgenden Ansätze die optimalen Trajektorien über unterschiedliche Modellparameter erstellt. Der Effekt einer nicht eindeutigen optimalen Stellgröße für jeden Zustand wird verstärkt dadurch, dass die optimale Stellgröße nun zusätzlich von den Modellparametern abhängt. Im Folgenden werden zwei Ansätze vorgestellt, diesen Effekt zu mildern.

Der erste, triviale Ansatz ist, den Regler von allen unbekannten Modellparametern $\mathbf{p}$ abhängig zu machen. Hierzu werden die Modellparameter auf ein Interval $p_i \in [-1,1]$ normiert und neben $\mathbf{x}_t$ als zusätzliche Eingänge des Reglers hinzugefügt. In der späteren Anwendung müssen diese zusätzlichen Eingänge $\mathbf{p}$ manuell eingestellt werden. Dies wird schon bei wenigen unbekannten Parametern schnell schwierig, weshalb ein weiterer Ansatz folgt.

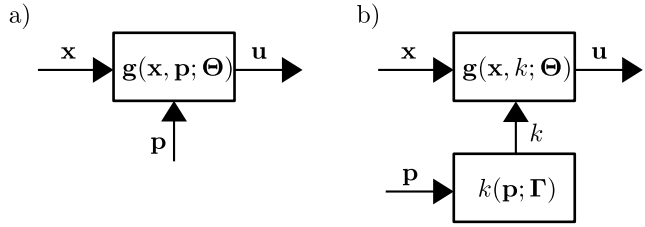


Bild 1: a) Des Regler hat so viele zusätzliche Eingänge wie unbekannte Modellparameter **p**. b) Der Regler hat einen zusätzlichen Eingang k, welcher während des Trainings von den unbekannten Modellparameters **p** abhängt.

Anstatt alle unbekannten Modellparameter als zusätzliche Eingänge zu verwenden fügen wir einen einzigen skalaren Eingang k zu dem Regler $\mathbf{g}(\mathbf{x}, k; \boldsymbol{\Theta})$ hinzu. Der Parameter k hängt während des Trainings von den gewählten Modellparametern $\mathbf{p}_{\text{model}}$ ab und soll in der späteren Anwendung per Hand eingestellt werden. Die Abhängigkeit zwischen k und $\mathbf{p}_{\text{model}}$ ist nicht bekannt und soll sich beim Training auch selbst ergeben, weshalb hierfür wieder eine Black-Box-Funktion angesetzt wird, welche nach dem Training nicht mehr benötigt wird (siehe Bild 1). Der Parameter k muss später im Betrieb auch wieder per Hand eingestellt werden, weil das echte **p** unbekannt ist. Allerdings fällt dies bei nur einem Parameter leichter als alle Einträge von **p** zu variieren.

Um die manuelle Einstellung des Parameters k einfacher zu gestalten, soll dessen Werteverteilung normiert werden und über die gewählte Verteilung oder das gewählte Interval von **p** einer Normalverteilung entsprechen. Hierzu wird eine Standardisierung der Verteilung von **x** und k über Trainingsbatches gemäß [9] eingefügt. Somit ist der Bereich in welchem k später zu wählen ist bekannt.

Die Ansätze zusätzliche Eingänge zum Regler hinzuzufügen lassen sich auch mit Policy-Gradient Verfahren verwenden über eine Betrachtung der unbekannten Modellparameter **p** als Erweiterung des Zustands $\mathbf{x}_t$ während des Trainings. Letztere Algorithmen weisen allerdings häufig Konvergenzprobleme auf und können keine Neben- und Endbedingungen (Gleichungen (1d), (1e)) berücksichtigen, weshalb diese hier nicht weiter untersucht werden. Da die Interpolation von optimalen Stellgrößen im geschlossenen System keine Garantie

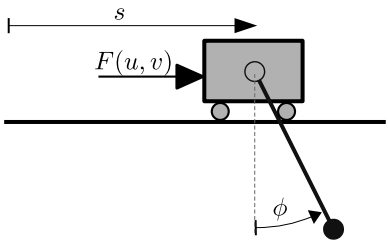


Bild 2: Schematische Darstellung eines Brückenkrans

oder Beweise für gute Performance liefert, wird im folgenden Abschnitt eine empirische Untersuchung an einem einfachen Beispiel durchgeführt.

3 Empirische Analyse am Beispiel Brückenkran

Als Beispiel-System wird im Folgenden ein Brückenkran mit angehängter Masse gewählt, dargestellt in Bild 2. Der Zustandsvektor $\mathbf{x} = \left[s, v, \phi, \dot{\phi}\right]$ beinhaltet die Position des Wagens s, die Geschwindigkeit des Wagens v sowie Winkel ϕ und Winkelgeschwindigkeit $\dot{\phi}$ des gespannten Seils. Die skalare Stellgröße u beschreibt eine Motorspannung eines Antriebs, welches den Wagen bewegt.

Die Modellgleichungen lauten

$$F(u,v) = \frac{ik_2}{rR}\left(u - \frac{ik_1}{r}v\right) - \mu v \tag{2}$$

$$K_1(\phi) = M_{\text{cart}} + M_{\text{load}}\sin(\phi)^2 \tag{3}$$

$$K_2(\phi) = M_{\text{load}}\sin(\phi)\cos(\phi) \tag{4}$$

$$\frac{d}{dt}\begin{bmatrix} s \\ v \\ \phi \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} v \\ \frac{M_{\text{load}}L_{\text{cable}}\dot{\phi}^2\sin(\phi)+K_2(\phi)g+F(u,v)}{K_1(\phi)} \\ \dot{\phi} \\ \frac{-K_2(\phi)L_{\text{cable}}\dot{\phi}^2-(M_{\text{cart}}+M_{\text{load}})g\sin(\phi)-F(u,v)}{K_1(\phi)L_{\text{cable}}} \end{bmatrix} \tag{5}$$

$$-40 \leq u \leq 40 \tag{6}$$

Tabelle 1: Modellparameter für das Brückenkran Beispiel

Variable	Wert	Einheit	Beschreibung
M_{cart}	$29.691 \pm 30\%$	kg	Masse des Wagens
M_{load}	0.135	kg	Angehängte Masse
L_{cable}	$0.589 \pm 30\%$	m	Länge des Seils
μ	0.589	$\mathrm{N\,s\,m^{-1}}$	Reibkoeffizient
k_1	0.263	V s	Motorkonstante
k_2	0.268	$\mathrm{N\,m\,A^{-1}}$	Motorkonstante
R	1.35	Ω	Motorwiderstand
r	0.0239	m	Radius Antriebsrad
i	3	/	Übersetzungsverhältnis

mit den Parameters wie in Tabelle 1 beschrieben. Die beiden Größen M_{cart} und L_{cable} werden dabei als unsicher angenommen und der Regler soll einstellbar und robust gegenüber diesen Parametern sein. Wie in Tabelle 1 angedeutet nehmen wir hier an, dass die wahren M_{cart} und L_{cable} sich in einem Interval von $\pm 30\%$ um einen Sollwert befinden.

Die gewählte Kostenfunktion für die Erstellung der Trainingsdaten über die Trajektorienoptimierung lautet

$$c(\mathbf{x},u) = 2(s+0.589\sin(\phi))^2 + 0.1(v+0.589\cos(\phi)\dot{\phi})^2 + 0.0001u^2 \qquad (7)$$

und zielt darauf ab die Position der angehängten Masse in den Nullzustand zu überführen.

Es werden vier verschiedene Regler trainiert:

- Ein Neuronales Netz ohne zusätzlichen Eingang $g_0(\mathbf{x})$, welches anhand von Daten ohne Variation der Modellparameter $\mathbf{p}$ trainiert wird ($\mathbf{p} = [M_{\text{cart}}, L_{\text{cable}}]^T = [29.691, 0.589]^T$) .
- Ein Neuronales Netz ohne zusätzlichen Eingang $g_1(\mathbf{x})$, welches anhand von Daten mit Variation der Modellparameter $\mathbf{p}$ trainiert wird.
- Ein Neuronales Netz mit zusätzlichem Eingang $g_k(\mathbf{x},k)$, welches anhand von Daten mit Variation der Modellparameter $\mathbf{p}$ trainiert wird.

Tabelle 2: Mittlerer quadr. Fehler der verschiedenen Neuronalen Netze nach dem Training für das Brückenkran-Beispiel.

Regler	$g_0(\mathbf{x})$	$g_1(\mathbf{x})$	$g_k(\mathbf{x},k)$	$g_\mathbf{p}(\mathbf{x},\mathbf{p})$
Trainingsdaten	$1.13 \cdot 10^{-3}$	0.365	0.260	$7.08 \cdot 10^{-3}$
Testdaten	$1.11 \cdot 10^{-3}$	0.363	0.215	$7.45 \cdot 10^{-3}$

- Ein Neuronales Netz $g_\mathbf{p}(\mathbf{x},\mathbf{p})$ mit zwei zusätzlichen Eingängen für beide Modellparameter.

Es werden zwei mal 10000 optimale Trajektorien erstellt, einmal ohne Variation der Modellparameter und einmal mit zufälligen $M_\text{cart} \sim \mathscr{U}(20.78, 38.6)$, $L_\text{cable} \sim \mathscr{U}(0.412, 0.765)$. Alle Neuronalen Netze haben eine ähnliche Anzahl an Parametern und zwei versteckte Layer mit tanh als Nichtlinearität. Die Strukturen der Regler sind in Bild 3 veranschaulicht.

Die Daten werden jeweils in 80% Trainingsdaten und 20% Testdaten aufgeteilt. Die Neuronalen Netze werden für jeweils 1000 Episoden mittels Gradientenabstieg und Adam [10] trainiert. Der mittlere quadratische Fehler nach dem Training ist in Tabelle 2 aufgetragen. Die beiden Neuronalen Netze $g_0(\mathbf{x})$ und $g_\mathbf{p}$ erreichen eine deutlich kleinere Abweichung auf beiden Datensätzen als $g_1(\mathbf{x})$ und $g_k(\mathbf{x},k)$. Dies lässt sich dadurch erklären, dass $g_1(\mathbf{x})$ und $g_k(\mathbf{x},k)$ nicht alle benötigten Informationen als Eingang haben um die Abhängigkeit der optimalen Stellgröße von den Modellparametern aufzulösen. Der Restfehler von $g_k(\mathbf{x},k)$ fällt kleiner aus als der von $g_1(\mathbf{x})$ da hier über k zumindest etwas Informationen über die Modellparameter vorliegen.

Um die Regelgüte bezüglich des Gütemaß in Gleichung (7) zu testen werden anschließend für mehrere Parametertupel die zu erwartenden Kosten in der Simulation ermittelt. Die Resultate für die verschiedenen Regler, sowie der Mittelwert über alle betrachteten Simulationen für den jeweiligen Regler sind in Bild 4 zu sehen. Für jeden Gitterpunkt sind die Kosten gemittelt über die gleichen 10000 Startzustände wie bei der Datengenerierung über die Trajektorienoptimierung. Anschließend wurde der Mittelwert der Kosten der optimalen Trajektorien ($\mathbb{E} = 45.41$) abgezogen um die Güte relativ zur optimalen Lösung anschaulicher zu machen. Um ein manuelles Einstellen für jeden Gitterpunkt

zu vermeiden wurde der Parameter k für den Regler $g_k(\mathbf{x},k)$ aus den Modellparametern und der temporären Funktion $k(\mathbf{p})$ berechnet. Die zusätzlichen Eingänge für den Regler $g_{\mathbf{p}}(\mathbf{x},\mathbf{p})$ werden ebenfalls ideal übergeben.

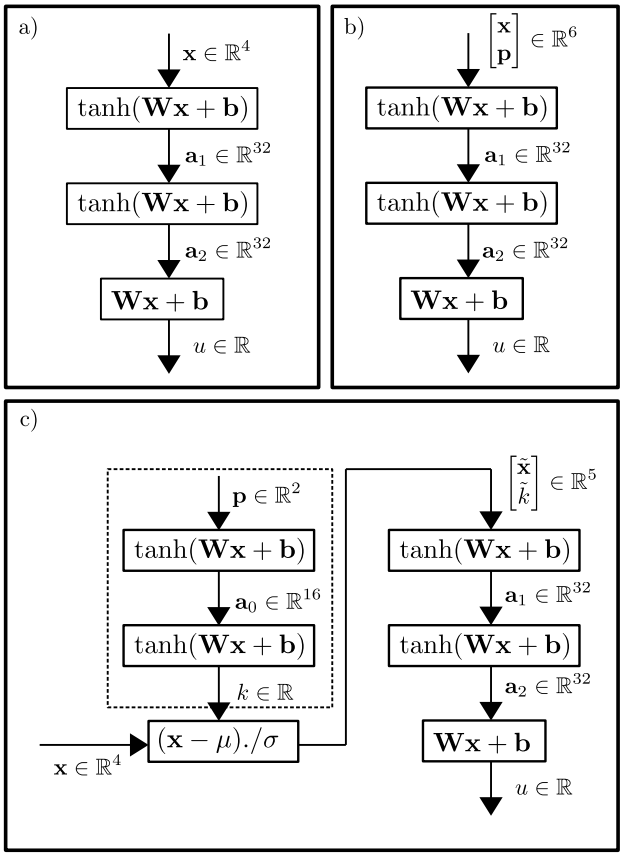


Bild 3: Struktur der verschiedenen Regler. Die Aktivierungsfunktionen tanh werden dabei elementweise angewendet und der Operator ./ soll eine elementweise Division darstellen. Die angegebene Dimension der Vektoren bezieht sich auf das Brückenkran Beispiel in Abschnitt 3. a) Struktur von $g_0(\mathbf{x})$ und $g_1(\mathbf{x})$; b) Struktur von $g_{\mathbf{p}}(\mathbf{x},\mathbf{p})$; c) Struktur von $g_k(\mathbf{x},k)$. Der punktierte Teil wird nur für das Training benutzt.

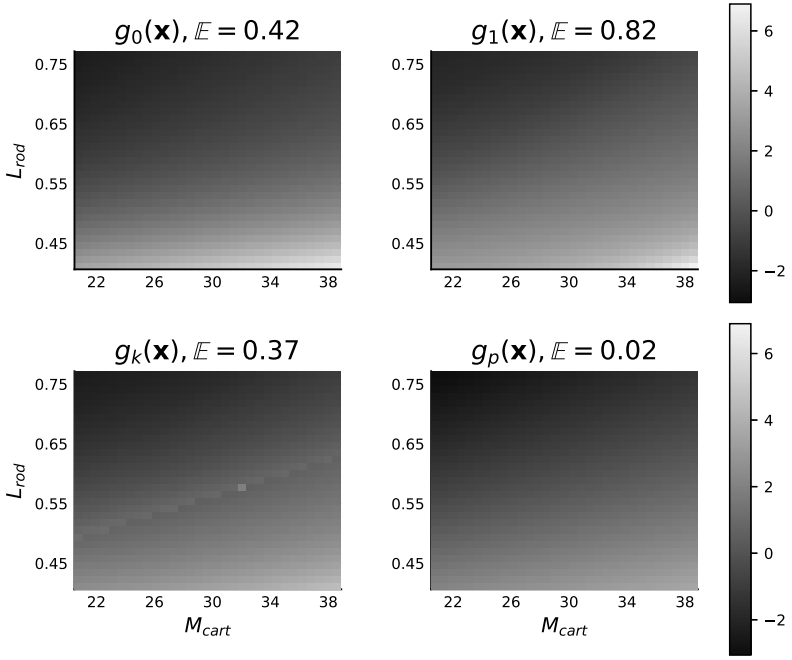


Bild 4: Gemittelte Kosten am Brückenkran-Beispiel für vier verschiedene Regler in Form von Neuronalen-Netzen. Pro Gitterpunkt, welcher jeweils einem Parametertupel $(M_{\text{cart}}, L_{\text{cable}})$ entspricht, sind die Kosten über 10000 Simulationen gemittelt, und der Mittelwert der Kosten über alle Parameter und Simulationen ist für jeden Regler zusätzlich angegeben.

Der Regler mit allen Modellparametern als Eingang $g_{\mathbf{p}}(\mathbf{x}, \mathbf{p})$ hat im Mittel die geringsten Kosten, welche nur um 0.02 höher ausfallen als die mittleren Kosten der optimalen Trajektorien. Der Regler $g_k(\mathbf{x})$, mit nur einem zusätzlichen Eingang, liegt bezüglich der mittleren Kosten auf zweiter Stelle. Die Kosten zeigen eine Unstetigkeit auf dem Gitter der Modellparameter in Bild 4. Diese resultiert aus der Funktion $k(\mathbf{p}_{\text{model}})$, welche hier einen starken Gradienten aufweist. Die beiden Regler ohne zusätzliche Informationen über die Modellparameter schneiden wie zu erwarten schlechter ab, wobei der Regler $g_0(\mathbf{x})$, welcher an Daten ohne Variation von $\mathbf{p}$ trainiert wurde deutlich besser abschneidet als der Regler $g_1(\mathbf{x})$, welcher an optimalen Trajektorien mit unterschiedlichen Modellparametern trainiert wurde. Eine Erklärung hierfür fehlt uns bisher.

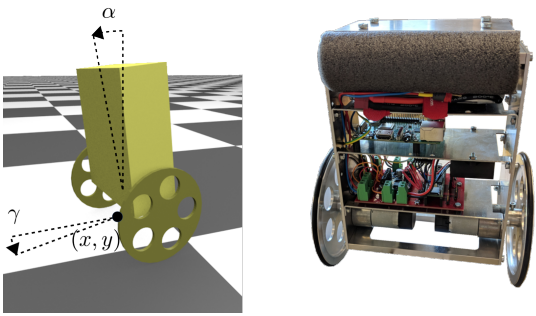


Bild 5: Das inverse Pendel auf Rädern.

4 Anwendung am inversen Pendel auf Rädern

Im Folgenden werden Black-Box-Regler mit zusätzlichen Eingängen für das System des inversen Pendel auf Rädern trainiert mit dem Ziel, eine Black-Box Regelung für die Stabilisierung und für Positionswechsel zu erzeugen.

Das inverse Pendel auf Rädern, abgebildet in Bild 5, ist ein instabiles und nichtholonomes nichtlineares System. Der Regler soll zu beliebigen Sollpositionen in einem Umkreis von 1m um die aktuelle Position fahren können. Im Falle von Sollpositionen außerhalt dieses Radius, welche später auch getestet werden, wird eine virtuelle Sollposition auf den Radius von 1m interpoliert.

Das für den Regelentwurf verwendete Modell beinhaltet die Modellgleichungen aus [11], erweitert um ein Motormodell ohne Stromdynamik und ein nichtlineares Reibmodell. Das Modell hat sieben Zustände $\mathbf{x} = \left[x, y, \gamma, \alpha, \dot{\alpha}, v, \dot{\gamma}\right]^T$: die aktuelle Position x, y, der Gierwinkel γ, der Kippwinkel α, die Vorwärtsgeschwindigkeit v, sowie die Winkelgeschwindigkeiten $\dot{\gamma}, \dot{\alpha}$. Die beiden Eingänge ins System sind die Motorspannungen am linken und rechten Motor, normiert auf einen Wertebereich $u_1, u_2 \in [-1, 1]$. Die Modellgleichungen aus [11] sollen an dieser Stelle nicht nochmal wiederholt werden. Das Motor- sowie Reibmodell lauten

$$i = u - k_1 \cdot \omega \tag{8}$$

$$F_{\text{fric}} = c_{\text{fric},1} \cdot \tanh(c_{\text{fric},2} \cdot \omega) \cdot e^{-c_{\text{fric},3} \cdot \omega^2} + c_{\text{fric},4} \cdot \omega \tag{9}$$

$$\tau = k_2 \cdot i - F_{\text{fric}} \tag{10}$$

wobei die benötigte Winkelgeschwindigkeit des Rads ω aus v und $\dot{\gamma}$ berechnet werden kann. Die Modellparameter sind in Tabelle 3 aufgelistet. Der Regler soll robust sein gegenüber den Modellparametern $\mathbf{p} = [I_{yy}, c_{\text{fric},1}, c_z]^T$ welche in Tabelle 3 als Wertebereiche aufgelistet sind.

Tabelle 3: Modellparameter für das inverse Pendel auf Rädern

Variable	Wert	Einheit	Beschreibung
M_{body}	1.76	kg	Masse des Körpers
M_{wheel}	0.147	kg	Masse eines Rades
R	0.07	m	Radius der Räder
c_z	$0.07 \pm 20\%$	m	Höhe des Körperschwerp. über Radachse
b	0.09925	V s	Halbe Länge zw. den Rädern
I_{xx}	0.0191	N m A^{-1}	Trägheitsmoment, x-Achse
I_{yy}	$0.0158 \pm 20\%$	Ω	Trägheitsmoment, y-Achse
I_{zz}	0.0048	m	Trägheitsmoment, z-Achse
I_{wa}	$3.6 \cdot 10^{-4}$	/	Trägheitsm. Rad, y-Achse
I_{wd}	$1.45 \cdot 10^{-3}$	/	Trägheitsm. Rad, z-Achse
k_1	0.0.18	V s	Motorkonstante
k_2	0.61	N m A^{-1}	Motorkonstante
$c_{\text{fric},1}$	$0.24 \pm 20\%$	N m^{-1}	Konstante im Reibmodell
$c_{\text{fric},2}$	2.0	/	Konstante im Reibmodell
$c_{\text{fric},3}$	0.4	/	Konstante im Reibmodell
$c_{\text{fric},4}$	$8 \cdot 10^{-4}$	N m^{-1}	Konstante im Reibmodell

Im ersten Schritt werden von 10000 unterschiedlichen Startzuständen in einem Bereich von 1.1m um die Wunschposition **0** optimale Trajektorien berechnet. Als Nebenbedingung der Optimierung wird der Endzustand im Wunschzustand erzwungen, sowie die Einhaltung der Stellgrößenbeschränkungen. Der Gierwinkel wird in allen Kostenberechnungen und Nebenbedingungen in die Anteile $\sin(\gamma), \cos(\gamma)$ aufgeteilt, damit der Regler gegenüber ganzen Umdrehungen

um die z-Achse invariant ist. Die Kostenfunktion, sowie Nebenbedingen der Trajektorienoptimierung lauten

$$\begin{aligned} c(\mathbf{x},\mathbf{u},t) = &\frac{t}{T}\left(2x^2+2y^2+(1-\cos\gamma)+v^2+2\dot{\gamma}^2\right) \\ &+0.3\alpha^2+0.3\dot{\alpha}^2+0.05(u_1^2+u_2^2) \end{aligned} \tag{11}$$

$$\begin{array}{ccc} x_T=0 & \alpha_T=0 & \dot{\gamma}_T=0 \\ y_T=0 & \dot{\alpha}_T=0 & -1\leq u_{1,t}\leq 1;\forall t \\ \cos(\gamma_T)-1=0 & v_T=0 & -1\leq u_{2,t}\leq 1;\forall t. \end{array}$$

Es werden wieder vier Regler anhand zweier Datensätze erstellt

- Ein Regler $g_0(\mathbf{x})$ ohne zusätzliche Eingänge, trainiert auf einem Datensatz ohne Variation der Parameter.
- Ein Regler $g_1(\mathbf{x})$ ohne zusätzliche Eingänge.
- Ein Regler $g_k(\mathbf{x})$ mit einem zusätzlichen Eingang k.
- Ein Regler $g_{\mathbf{p}}(\mathbf{x})$ mit drei zusätzlichen Eingängen.

Die Anzahl an Neuronen für die Regler wurde erhöht gegenüber dem Brückenkran-Beispiel aufgrund der erhöhten Komplexität und Dimension. Die Grundstruktur der Neuronalen Netze bleibt allerdings die gleiche wie in Bild 3 mit den Änderungen $\mathbf{x}\in\mathbb{R}^7$, $\mathbf{u}\in\mathbb{R}^2$, $\mathbf{p}\in\mathbb{R}^3$, $\mathbf{a}_0\in\mathbb{R}^{64}$, $\mathbf{a}_1\in\mathbb{R}^{128}$ und $\mathbf{a}_2\in\mathbb{R}^{128}$.

Die Datensätze werden wieder in 80% Trainingsdaten und 20% Testdaten aufgeteilt und über 2000 Episoden mittels Adam trainiert. Die Restfehler auf den Datensätzen sind in Tabelle 4 aufgetragen. Hierbei fällt vor allem der Fehler von $g_k(\mathbf{x},k)$ auf. Dieses Neuronale Netz sollte aufgrund seiner Struktur einen kleineren Fehler erreichen können als $g_1(\mathbf{x})$, da es zusätzliche Parametern enthält. Wir vermuten hier, dass die Batch-Normalisierung und die Abhängigkeit von k das Training erschwert.

Tabelle 4: Mittlerer quadr. Fehler für die Regler des inversen Pendel auf Rädern.

Regler	$g_0(\mathbf{x})$	$g_1(\mathbf{x})$	$g_k(\mathbf{x},k)$	$g_p(\mathbf{x},\mathbf{p})$
Trainingsdaten	$3.91 \cdot 10^{-4}$	$6.49 \cdot 10^{-4}$	$1.03 \cdot 10^{-3}$	$4.685 \cdot 10^{-4}$
Testdaten	$4.21 \cdot 10^{-4}$	$6.97 \cdot 10^{-4}$	$8.51 \cdot 10^{-4}$	$5.367 \cdot 10^{-4}$

Die Regler werden am realen System auf Balancieren sowie das Fahren zu einer Sollposition getestet. Die Sollposition $\mathbf{x}_{\text{soll}} = [x_{\text{soll}}, y_{\text{soll}}, \gamma_{\text{soll}}]^T$ wechselt dabei alle 10 Sekunden entsprechend:

$$\mathbf{x}_{\text{soll}}(t) = \begin{cases} [0,0,0]^T \text{ , wenn } (t \bmod 40) < 10 \\ [\frac{3}{2},0,-\frac{\pi}{2}]^T \text{ , wenn } (t \bmod 40) < 20 \\ [\frac{3}{2},\frac{3}{2},\pi]^T \text{ , wenn } (t \bmod 40) < 30 \\ [0,\frac{3}{2},\frac{\pi}{2}]^T \text{ , sonst.} \end{cases} \tag{12}$$

Der Regler $g_1(\mathbf{x})$ schafft es in der Anwendung nicht das System zu stabilisieren. Da der Regler $g_0(\mathbf{x})$ das System ohne Probleme stabilisiert bestätigt dies die Ergebnisse vom Brückenkran-Beispiel, dass ein einfaches Supervised-Learning auf Stellgrößen, welche mittels unterschiedlichen Modellparametern generiert wurden, zu schlechten Ergebnissen im geschlossenen System führt. Der Regler $g_k(\mathbf{x},k)$ schafft es zwar das System nahe der Ruhelage zu stabilisieren, jedoch kippt das System mit diesem Regler beim Wechsel der Sollposition aus Gleichung (12) um. Das bessere Verhalten von $g_k(\mathbf{x},k)$ gegenüber $g_1(\mathbf{x})$ zeigt dennoch, dass der Fehler auf den Trainingsdaten nicht unbedingt mit der Regelgüte im geschlossenen System übereinstimmen muss. Ein Einstellen des Parameters k verändert zwar sichtlich das Verhalten des Systems beim Balancieren, es konnte jedoch kein Parameter gefunden werden, welcher ein Folgen der Sollposition in (12) erlaubt. Während die Ergebnisse für diese Reglerstruktur beim Brückenkran-Beispiel noch vielversprechend waren, scheint die Methode nicht auf mehrere unbekannte Modellparameter zu skalieren.

Die beiden Regler $g_0(\mathbf{x})$ und $g_{\mathbf{p}}(\mathbf{x})$ stabilisieren das System nahe der Ruhelage und haben auch beim Wechsel der Sollposition keine Probleme. Messungen der Zustände des Systems sind für den Regler $g_0(\mathbf{x})$ in Bild 6, sowie für zwei unterschiedliche Einstellungen von $g_{\mathbf{p}}(\mathbf{x})$ in Bild 7 und 8 zu sehen.

Während das Verhalten von $g_0(\mathbf{x})$ nach dem Training festgelegt ist, führt eine Variation der Einträge aus $\mathbf{p}$ zu merklichen Änderungen im Verhalten des Systems nahe der Ruhelage, wie im Vergleich der Messungen in Bild 7 und 8 zu erkennen ist. Die gewählte Konfiguration entspricht der aus Bild 8; das Verhalten wurde per Hand so eingestellt, dass das System sich subjektiv ruhig verhält und sich trotzdem nicht zu weit von der Ruheposition entfernt. Dies ist vor allem am Kippwinkel α zu erkennen, welcher in Bild 6 mit hoher Frequenz das Vorzeichen wechselt, in Bild 8 ist die Frequenz deutlich kleiner.

5 Zusammenfassung

Es wurde eine Methode vorgestellt, einen Regler in Form einer parametrierten Funktion zu erstellen, welcher nach dem Erlernen noch nachjustiert werden kann. Hierzu wird die parametrierte Funktion an optimalen Trajektorien, generiert an Modellen mit unterschiedlichen Modellparametern, trainiert. Eine empirische Untersuchung ergab, dass Informationen über die Modellparameter mit in den Regler übergeben werden sollten, falls diese nicht konstant gehalten wurden während der Trajektoriengenerierung. Die Anzahl an variierenden Modellparametern sollte außerdem bei diesem Ansatz klein gehalten werden, um eine Anpassung des Reglers in der Praxis zu vereinfachen. Ein Versuch, mehrere Modellparameter in einem Skalar zusammenzufassen um die Anpassung zu erleichtern funktionierte nur für ein Beispiel mit zwei unbekannten Modellparametern, jedoch nicht mehr beim realen Anwendungsfall mit drei Parametern. Der Ansatz einen zusätzlichen Eingang für jeden unbekannten Modellparameter hinzuzufügen erwies sich als nützlich für den Fall von unbekannten Parametern oder einfach bei Bedarf einer späteren Anpassung des Verhaltens der Regelung.

Weiterführend soll versucht werden, zusätzliche Eingänge der Regler für unbekannte Modellparameter während des Betriebs automatisch bestimmen zu lassen, z.B. mittels rekurrenten Neuronalen Netzen. Weiterhin soll jedoch ein skalarer Eingang zur Einstellung des Verhaltens beibehalten werden, da sich die Möglichkeit zum Nachjustieren als nützlich erwies im realen Anwendungsfall.

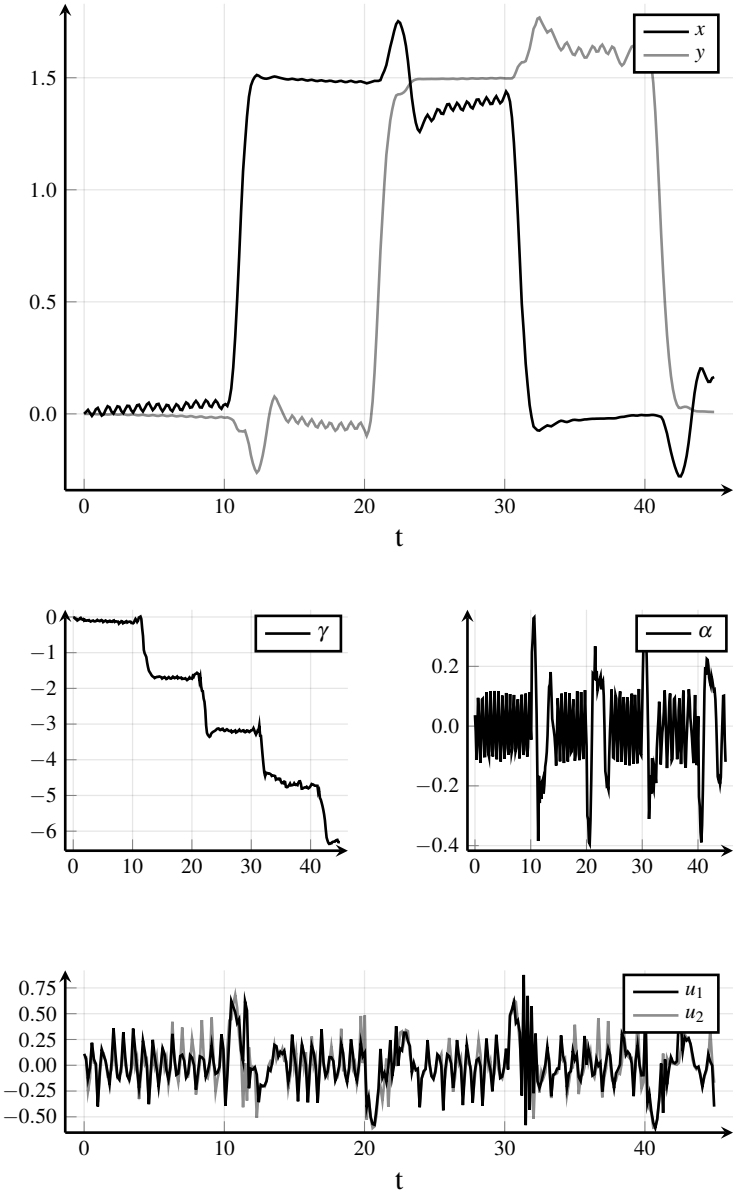


Bild 6: Messung für $g_0(\mathbf{x})$.

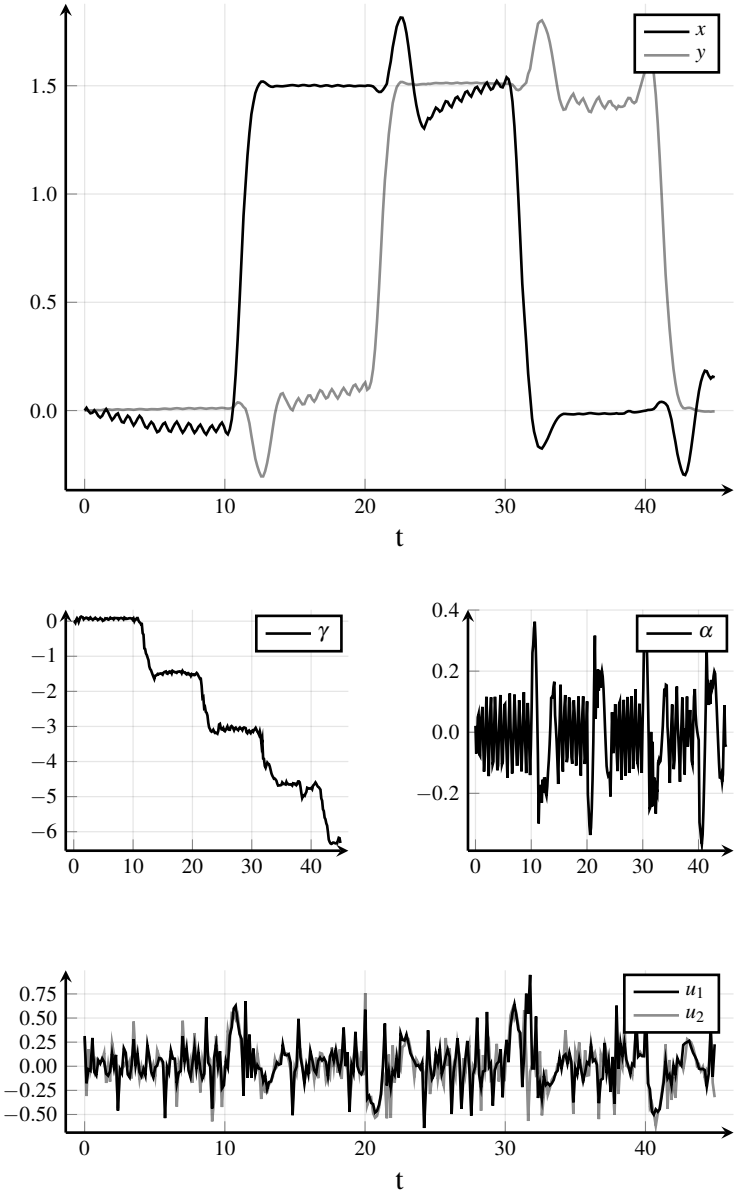


Bild 7: Messung für $g_{\mathbf{p}}(\mathbf{x},\mathbf{p})$ mit Eingang $\mathbf{p} = [0, 0.5, 0.8]^T$.

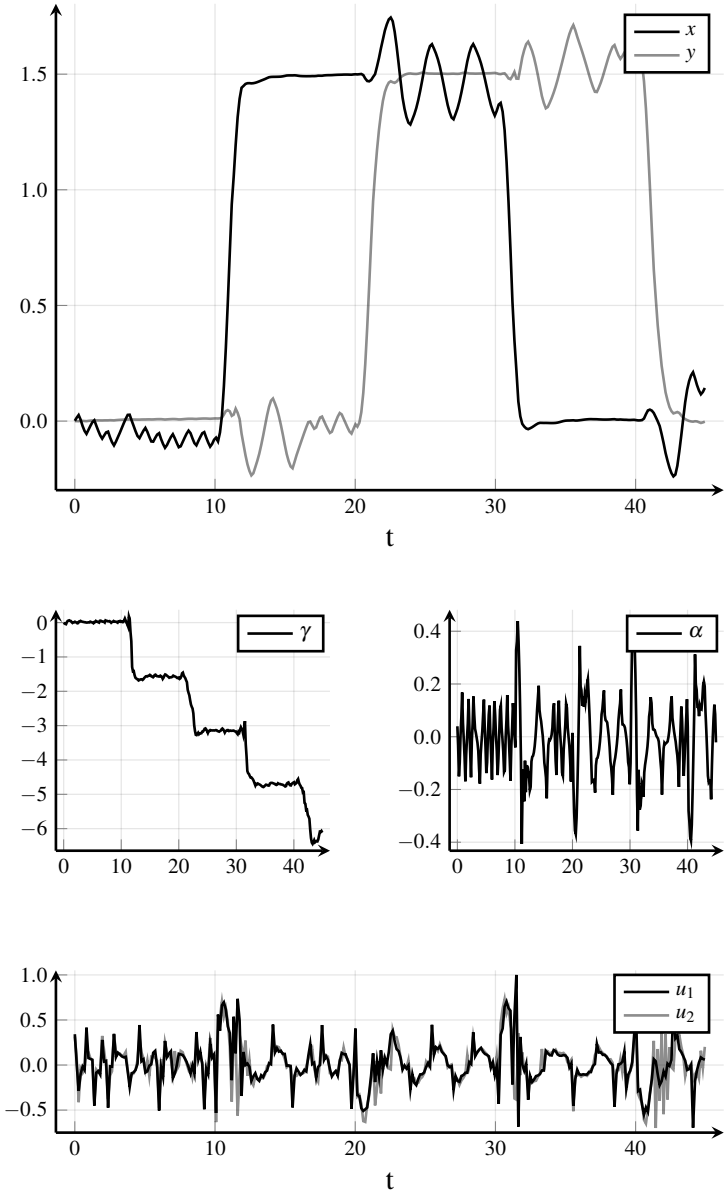


Bild 8: Messung für $g_{\mathbf{p}}(\mathbf{x},\mathbf{p})$ mit Eingang $\mathbf{p} = [-0.1, 0.2, 0.3]^T$.

Danksagung

Die Autoren danken 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] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O.Klimov: „Proximal Policy Optimization Algorithms“. CoRR. 2017.

[2] B.D. Argall, S.Chernova, M. Veloso and B. Browning: „A survey of robot learning from demonstration“. Robotics and autonomous systems, vol. 57. 2009.

[3] A. Rajeswaran, S. Ghotra, B. Ravindran and S.Levine: „Epopt: Learing robust neural network policies using model ensembles“. ICLR. 2017.

[4] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan and G. Chowdhary „Robust deep reinforcement learning with adversarial attacks“. 17th International Conference on Autonomous Agents and MultiAgent Systems 2018.

[5] J. M. Wang, D. J. Fleet, and A. Hertzmann, „Optimizing walking controllers for uncertain inputs and environments“ ACM Transactions on Graphics, vol. 29 2010.

[6] I. Mordatch, K. Lowrey and E. Todorov: „Ensemble-CIO: Full-body dynamic motion planning that transfers to physical humanoids“. International Conference on Intelligent Robots and Systems. 2015.

[7] R. Dessort and C. Chucholowski, „Explicit model predictive control of semi-active suspension systems using Artificial Neural Networks (ANN)“ 8th International Munich Chassis Symposium 2017

[8] I. Mordatch, K. Lowrey, G. Andrew, Z. Popovic, and E. V. Todorov „Interactive control of diverse complex characters with neural networks“ Advances in Neural Information Processing Systems 2015

[9] S. Ioffe and C. Szegedy, „Batch normalization: Accelerating deep network training by reducing internal covariate shift“ International Conference on Machine Learning. 2015.

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

[11] K. Pathak, J. Franch, and S. K. Agrawal: „Velocity and position control of a wheeled inverted pendulum by partial feedback linearization“ IEEE Transactions on robotics, vol. 21 2005

Zur Übertragbarkeit der Ähnlichkeitstransformation von LTI auf Takagi-Sugeno Fuzzy Systeme

Horst Schulte, Stephan Kusche

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

Kurzfassung

In diesem Beitrag wird die Übertragbarkeit der Ähnlichkeitstransformation von linearen zeitinvarianten Zustandsraummodellen auf Takagi-Sugeno Fuzzy Systeme untersucht. Dabei wird die für den modellbasierten Regelungsentwurf wichtige Klasse der Takagi-Sugeno Fuzzy Systeme mit jeweils einem Zustandsraummodell pro Regel in der Konklusion behandelt. Die Ähnlichkeitstransformation für Takagi-Sugeno Fuzzy (TSF) Systeme lässt sich entweder nur auf die einzelnen lokalen Zustandsraummodelle (I) oder auf das gesamte Takagi-Sugeno Fuzzy System (II) anwenden. Bei der Variante II handelt es sich um eine nichtlineare oder lineare parameter-variable Transformation, wodurch das transformierte System auch von der Ableitung der Prämissenvariablen abhängt, die im allgemeinen jedoch unbekannt ist und abgeschätzt werden muss. Untersuchungen zu II wurden bereits in [1] durchgeführt. Wird die Ähnlichkeitstransformation dagegen nur auf die Konklusion der TSF Systeme angewandt, erhält man lokale Modelle, die äquivalent bezüglich des Eingangs/Ausgangsverhaltens und der Eigendynamik des Ursprungssystems sind, sich allerdings nicht mehr wie bei TSF Systeme üblich auf einen gemeinsamen Zustandsvektor beziehen lassen. In dieser Arbeit wird ein Näherungsverfahren für die lokale Ähnlichkeitstransformation (I) vorgestellt, bei der die wichtigsten

Anteile aller Zustandsraumtransformationen in einem gemeinsamen Zustandsvektor zusammengefasst werden. Die Anwendbarkeit dieses Verfahren wird anhand von typischen Aufgabenstellungen der Kontrolltheorie aufgezeigt.

1 Einführung

Die Koordinatentransformation

$$\tilde{x} = Tx \tag{1}$$

mit $T \in \mathbb{R}^{n\times n}$, $\det(T) \neq 0$ und den Zustandsvektoren $x, \tilde{x} \in \mathbb{R}^n$ wird in der Kontrolltheorie angewandt auf lineare zeitinvariante (LTI) Systeme

$$\dot{x} = Ax + Bu, \qquad y = Cx \tag{2}$$

um die Analyse und den Entwurf durch das Bilden von bestimmten Normalformen zu vereinfachen. Dabei überführt die Koordinatentransformation (1) zusammen mit der Ableitung $\dot{\tilde{x}} = T\dot{x}$ das LTI System (2.1) in ein neues System

$$\dot{\tilde{x}} = \underbrace{TAT^{-1}}_{\tilde{A}}\tilde{x} + \underbrace{TB}_{\tilde{B}}u, \qquad y = \underbrace{CT^{-1}}_{\tilde{C}}\tilde{x}. \tag{3}$$

Hierbei bleibt das Ein-/Ausgangsverhalten des neuen (3) gegenüber dem gegebenen System (2.1) und die Eigenwerte der Systemmatrizen unverändert, d.h. $\mathrm{eig}(A) = \mathrm{eig}(\tilde{A})$. Aus diesem Grund spricht man bei der Transformation (1) angewandt auf LTI Systeme von einer Ähnlichkeits- oder auch Äquivalenztransformation. Bekannte Normalformen, die durch Anwendung von (1) auf ein LTI System der Form (2.1) angewandt werden, sind z.B. die Diagonalform[1]

$$\dot{x}_d = A_d x_d + B_d u, \quad y = C_d x_d \tag{4}$$

[1] Voraussetzung ist, dass alle Eigenwerte von A einfach auftreten und reellwertig sind.

mit

$$A_d = \begin{pmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_n \end{pmatrix}, \quad B_d = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \quad C_d = \begin{pmatrix} b_1, b_2, \cdots, b_n \end{pmatrix},$$

die Regelungsnormalform[2]

$$\dot{x}_r = A_r x_r + B_r u, \quad y = C_r x_r \tag{5}$$

mit

$$A_r = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{pmatrix}, \quad B_r = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}, \quad C_r = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}^T \tag{6}$$

und die Beobachtungsnormalform[3]

$$\dot{x}_b = A_b x_b + B_b u, \quad y = C_b x_b \tag{7}$$

mit $A_b = A_d^T$, $B_b = C_d^T$ und $C_b = B_d^T$. Neben den konventionellen Normalformen für SISO (Single-In, Single-Out) und MIMO (Multi-Input, Multi-Output) Systeme (vgl. [2]) werden z.B. in der Sliding Mode Theorie für Mehrgrößensysteme verkette Transformationen für den Regler-und Beobachterentwurf eingesetzt [3]. Beim Entwurf von Sliding Mode Beobachtern werden vier Subtransformationen mit

$$T = T_c \, T_b \, T_a \, T_L \tag{8}$$

[2] Voraussetzung ist, dass das System vollständig steuerbar ist.
[3] Voraussetzung ist, dass das System vollständig beobachtbar ist.

hintereinander ausgeführt [4]. Jede Transformation erfüllt einen besonderen Zweck [6]:

- T_c: Unterteilt das System in messbare und nicht messbare Systemzustände
- T_b : Erzeugt eine spezielle Struktur in den Störungs- und Ausgangsmatrizen, so dass die unbekannten Störungen nur auf die messbaren Zustände wirken.
- T_a: Transformiert die Systemmatrix A in eine spezielle Struktur, so dass eine Submatrix immer beobachtbar ist
- T_L: Führt eine Designmatrix L zur Stabilisierung der Beobachtergleichung ein

Um dieses Entwurfsverfahren für TSF Systeme stringent und allgemein anwenden zu können, muss die Übertragbarkeit der Äquivalenztransformation für diese Systemklasse im Detail untersucht werden.

2 Übertragung der Ähnlichkeitstransformation auf T-S Fuzzy Systeme

2.1 Problemstellung

Für die weiteren Untersuchungen betrachten wir das TSF System [5] bestehend aus $i = 1, \ldots, N_r$ linguistischen Regeln R_i

$$\begin{aligned} R_i : \textbf{If } & z_1 \textbf{ is } M_{i,1} \textbf{ and } z_2 \textbf{ is } M_{i,2} \textbf{ and } \cdots \textbf{ and } z_p \textbf{ is } M_{i,p} \\ & \textbf{then} \\ & \dot{x} = A_i x + B_i u\,, \quad y = C_i x \end{aligned} \tag{9}$$

mit den Fuzzymengen $M_{i,j}$, den zugehörigen Prämissenvariablen z_j, die als Vektor $z \in [z_1, \cdots, z_{N_l}]$, $N_l \in \mathbb{N}^0$ zusammengefasst werden und den linearen Teilsystemen

$$\dot{x} = A_i x + B_i u\,, \quad y = C_i x$$

in der Konklusion. Die Prämissenvariablen können auf Systemzustände, Systemeingänge, externe Variablen und zeitveränderliche Parameter $\theta(t)$ verweisen: $z_j \in \{x_i, u_k, \theta_q\}$. Es stellt sich nun die Frage, inwieweit sich die Äquivalenztransformation (1) auf (9) übertragen lässt und ob das Ein-/Ausgangsverhalten gegenüber dem ursprünglichen System, wie bei LTI Systemen erhalten bleibt. Für einen analytischen Zugang wird eine übliche mathematische Interpretation von (9) gewählt, vgl. [7]

$$\dot{x} = \sum_{i=1}^{N_r} h_i(z)\left(A_i x + B_i u\right), \quad y = \sum_{i=1}^{N_r} h_i(z) C_i x \tag{10}$$

Die Form des TSF Systems (10) ergibt sich aus der Übertragung der linguistischen Terme der Regelbasis R_i, $i = 1,\ldots,N_r$ in eine quantifizierte mathematische Darstellung. Hierzu werden die einzelnen linguistischen Ausdrücke, wie in Tabelle 1 angegeben, ersetzt und zusammengefügt. Die abschließende Zusammenfassung aller Regeln folgt aus der mit $h_i(z)$ gewichteten Summe aller $i = 1,\ldots,N_r$ LTI Modelle, vgl. Tabelle 1.

Tabelle 1: Interpretation der linguistischen Ausdrücke

Linguistischer Ausdruck	Quantifizierung	Beschreibung
z_j **is** $M_{i,j}$	$\mu_{M_{i,j}}(z_j)$	Zugehörigkeitsfunk.
z_1 **is** $M_{i,1}$ AND … … AND z_{N_l} **is** M_{i,N_l}	$h_i(z) := \prod_{j=1}^{N_l} \mu_{M_{i,j}}(z_j)$	Gewichtung mittels Larsen Implikation
$\cup_{i=1}^{N_r} R_i$	$\sum_{i=1}^{N_r} h_i(z)\left(A_i x + B_i u\right)$	Aggregation der Regeln R_i, $\forall i$

Soll nun die Transformation (1) auf das TSF System (9) oder (10) angewendet werden ergeben sich zwei Varianten: Entweder verwendet man eine gemeinsame lineare Transformation für alle Teilsysteme oder für jedes $i-$te Teilsystem wird eine spezielle Transformation T_i ausgeführt. Um z.B. jedes Teilsystem in eine Normalform wie die Beobachtungsnormalform (7) überführen zu können, ist nur in speziellen Fällen T für alle Modelle identisch: $T = T_i, \forall i$, siehe das illustrative Beispiel in [1]. Vielmehr gilt allgemein

$$T_1 \neq T_2 \neq \cdots \neq T_{N_r}$$

Dieser Sachverhalt mit $\tilde{x}_i = T_i x$ und $\dot{\tilde{x}}_i = T_i \dot{x}$ hat zur Konsequenz, dass die transformierten Teilsysteme

$$\dot{\tilde{x}}_i = T_i A_i T_i^{-1} \tilde{x}_i + T_i B_i u\,, \qquad y_i = C_i T_i^{-1} \tilde{x}_i \tag{11}$$

sich nicht mehr wie bei (9), (10) auf einen gemeinsamen Zustandsvektor x beziehen lassen, wodurch der TSF Rahmen verlassen wird.

2.2 Lösungsansatz

Um die im allgemeinen unterschiedlichen Zustandsräume der Teilsysteme wieder zusammenführen zu können, werden mit einer zweiten Transformation

$$\bar{x} = F_i \tilde{x}_i\,, \tag{12}$$

die wichtigsten Anteile aller Transformationen T_i in einem gemeinsamen Zustandsvektor $\bar{x}$ näherungsweise erfasst. Die Matrizen F_i ergeben sich dabei wie folgt

$$F_i = R^T T_i^{-1}\,, \tag{13}$$

wobei R mit der Eigenschaft $RR^T = I$ die Haupttransformation beinhaltet. Diese wird aus der Singulärwertzerlegung der Matrix aller Transformationen

$$\bar{T} = [\,T_1\; T_2\; \cdots\; T_{N_l}\,] \in \mathbb{R}^{n \times n \cdot N_r} \tag{14}$$

mit

$$\bar{T} = U S V^T \tag{15}$$

berechnet mit $R = U$, die aus den Singulärwertvektoren gebildet wird. In (15) ist $U \in \mathbb{R}^{n \times n}$ eine unitäre Matrix, die Matrix $S \in \mathbb{R}^{n \times n \cdot N_r}$ enthält n Singulärwerte und $V^T \in \mathbb{R}^{n \cdot N_r \times n \cdot N_r}$ ist die Adjungierte einer unitären Matrix.

3 Illustratives Beispiel

Zur Veranschaulichung des Ansatzes betrachten wir nun das folgende Zustandsraummodell aus [8], Beispiel 2.2:

$$\begin{aligned} \dot{x}_1 &= -x_1 + x_1 x_2 \\ \dot{x}_2 &= \quad x_1 - 3x_2 \,, \qquad y = x_1 \end{aligned} \tag{16}$$

mit $x_1, x_2 \in [-1, 1]$. Das nichtlineare Modell wird innerhalb der gegebenen Grenzen für x_1 und x_2 exakt durch das TSF Äquivalent (10) beschrieben

$$\begin{aligned} \dot{x} &= h_1(x_2) \underbrace{\begin{bmatrix} -2 & 0 \\ 1 & -3 \end{bmatrix}}_{A_1} x + h_2(x_2) \underbrace{\begin{bmatrix} 0 & 0 \\ 1 & -3 \end{bmatrix}}_{A_2} x \,, \\ y &= \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C} x \,. \end{aligned} \tag{17}$$

Die Beobachternormalformen (7) der Teilsysteme ergibt sich aus den Transformationen

$$T_1 = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}, \qquad T_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} .$$

Mit (13) und der Singulärwertzerlegung der zusammengefassten Transformationsmatrix erhält man

$$F_1 = \begin{bmatrix} -0.9239 & 1.4651 \\ -0.3827 & 1.6892 \end{bmatrix}, \qquad F_2 = \begin{bmatrix} -0.9239 & -0.3827 \\ -0.3827 & 0.9239 \end{bmatrix}.$$

Daraus folgen die Teilsysteme des transformierten TSF Systems, die sich auf einen gemeinsamen neuen Zustandsvektor beziehen.

4 Zusammenfassung

Vorgestellt wurde ein Verfahren, wie durch eine Erweiterung der Ähnlichkeitstransformation von LTI Systemen, welche die Hauptanteile aller Teil-Transformationen zusammenfasst, die TSF Systemeigenschaft eines gemeinsamen Zustandsvektors aller Teilsysteme erhalten bleibt. In weiteren Untersuchungen werden Kriterien zur Bewertung der Güte der Approximation hinsichtlich des Ein-Ausgangsverhaltens und der Erhaltung der Dynamik der Eigenbewegung zusammen mit den Stabilitätsuntersuchungen aus [1] entwickelt.

Literatur

[1] H. Schulte, S. Georg. „Coordinate Transformation of Takagi-Sugeno Models: Stability Conditions and Observer Canonical Forms“. In: 2014 IEEE International Conference on Fuzzy Systems, 2472-2476, Bejing, China, 2014.

[2] G. Ludyk. „Theoretische Regelungstechnik 1“. Springer Verlag, Berlin. 1995.

[3] C. Edwards und S. K. Spurgeon (1998). „Sliding Mode Control: Theory and Applications“. Taylor & Francis, Boca Raton. 1998.

[4] C. Edwards und S. K. Spurgeon, and R. J. Patton (2000). „Sliding mode observers for fault detection and isolation“. Automatica 36, S. 541–553. 2000.

[5] T. Takagi und M. Sugeno. „Fuzzy Identification of Systems and Its Application to Modeling and Control“. IEEE Transactions on Systems, Man, and Cybernetics 15(1), S. 116–132. 1885.

[6] S. Georg. „Fault Diagnosis and Fault-Tolerant Control of Wind Turbines“. Dissertation HTW Berlin/Fakultät für Maschinenbau und Schiffstechnik der Universität Rostock. 2016

[7] R. Palm, D. Driankov und H. Hellendoorn. „Model Based Fuzzy Control“. Springer-Verlag, New York, 1997.

[8] Z. Lendek, T. M. Guerra, R. Babuska und B. De Schutter, „Stability Analysis and Nonlinear Observer Design Using Takagi-Sugeno Fuzzy Models“. Springer-Verlag Berlin Heidelberg, 2010.

Zur Schätzung zulässiger Parametermengen nichtlinearer Takagi-Sugeno-Multi-Modelle mit garantierten Fehlerschranken

Felix Wittich, Matthias Kahl, Andreas Kroll

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

1 Einführung

Bei der datengetriebenen Modellbildung ist neben der Punktschätzung der Modellparameter auch eine Beschreibung deren Unsicherheit von großem Interesse. Damit lassen sich die Modellgüte und -vertrauenswürdigkeit besser einschätzen. Zudem kann die Information für einen robusten modellbasierten Reglerentwurf genutzt werden. Üblicherweise wird hierfür eine Parameterschätzung in einem statistischen Framework herangezogen; so kann beispielsweise durch die Inverse der Fisher-Informations-Matrix bei der Maximum-Likelihood-Schätzung die Unsicherheit des Schätzwertes quantifiziert werden [1]. Diese Unsicherheitsbeschreibungen sind allerdings nur unter restriktiven Annahmen bezüglich der Wahrscheinlichkeitsdichtefunktion des Fehlers und bei großen Datenmengen gültig. Bei praktischen Anwendungsfällen werden diese Voraussetzungen häufig nicht erfüllt. Alternative Ansätze zur Beschreibung von Parameterunsicherheiten, die unabhängig von probabilistischen Annahmen arbeiten, bieten mengenbasierte Ansätze; so genannte Bounded-Error-(BE-)- beziehungsweise Set-Membership-(SM-)-Schätzverfahren [2]. Hierbei wird die Annahme getroffen, dass der Prädiktionsfehler in einem Intervall mit festgelegten Schranken liegt und entsprechend jene Parametermenge bestimmt, mit der die Modellausgabe innerhalb dieser garantierten Fehlerschranken liegt.

Die Fehlerschranke als vom Nutzer festzulegender Parameter bestimmt die Größe der zulässigen Parametermenge.

In diesem Beitrag werden Verfahren zur Schätzung von zulässigen Parametermengen mit beschränkten Fehlern im Hinblick auf ihre Eigenschaften wie Rechenkomplexität und konzeptionelle Eignung für lokal-affine Multi-Modelle vom Typ Takagi-Sugeno (TS) untersucht und bewertet. Bisher wurden die Verfahren nur für Probleme mit wenigen Parametern untersucht. Neben Verfahren zur exakten Beschreibung der Parametermenge wird betrachtet, wie Approximationen – beispielsweise durch Ellipsoide – zur Reduktion der Rechenkomplexität und zur kompakten Beschreibung der Parametermengen eingesetzt werden können. Dabei wird untersucht, wie die TS-Modellstruktur für die Bestimmung einer zulässige Parametermenge mit garantierten Fehlerschranken bei der datengetriebenen Modellbildung nichtlinearer Systeme ausgenutzt werden kann.

2 Takagi-Sugeno-Multi-Modelle

Bei der Modellierung von statischen oder dynamischen Systemen mit einem TS-Modell werden lokal-affine Teilmodelle identifiziert und diese über im Schedulingraum definierte Zugehörigkeitsfunktionen gewichtet überlagert. Durch diese Überlagerung der Teilmodelle ist es möglich, mit TS-Modellen nichtlineare Funktionen zu approximieren [3].

Die in diesem Beitrag betrachteten statischen Multi-Input-Single-Output-TS-Gesamtmodelle

$$\hat{y}(\boldsymbol{x}, \boldsymbol{\theta}_{\mathrm{MF}}, \boldsymbol{\theta}_{\mathrm{LM}}) = \sum_{j=1}^{c} \phi_j(\boldsymbol{x}, \boldsymbol{\theta}_{\mathrm{MF}}) \hat{y}_j(\boldsymbol{x}, \boldsymbol{\theta}_{\mathrm{LM}}) \tag{1}$$

mit den Eingangsgrößen $\boldsymbol{x} \in \mathbb{R}^n$ setzen sich aus c überlagerten lokal-affinen Teilmodellen

$$\hat{y}_j(\boldsymbol{x}) = \theta_{j,0} + \theta_{j,1} x_1 + \cdots + \theta_{j,n} x_n \tag{2}$$

zusammen. Als Zugehörigkeitsfunktionen werden orthogonale Fuzzy-Basisfunktionen vom FCM-Typ eingesetzt:

$$\mu_j(\boldsymbol{x}) = \left[\sum_{l=1}^{c} \left(\frac{\|\boldsymbol{x} - \boldsymbol{v}_j\|_2}{\|\boldsymbol{x} - \boldsymbol{v}_l\|_2} \right)^{\frac{2}{\nu-1}} \right]^{-1}. \tag{3}$$

3 Bounded-Error-Identifikation

Die Schätzung der Modellparameter bei der datengetriebenen Modellbildung lässt sich im BE-Framework [4] darstellen. Die gemessenen Daten $\{(y_k, \boldsymbol{x}_k)\},\ \boldsymbol{x}_k \in \mathbb{R}^N,\ y_k \in \mathbb{R}^N,\ k = 1, \ldots, N$ sollen durch ein parametrisches Modell $\hat{y}(\boldsymbol{x}) = f(\boldsymbol{x})$ beschrieben werden. Das Ziel ist es, die Menge aller zulässigen Parameter $\mathbb{S}_{\text{FPS}} \in \mathbb{R}^n$ zu finden, die zu Ausgangsfehlern in den festgelegten Schranken führen:

$$\mathbb{S}_{\text{FPS}} = \{\boldsymbol{\theta} \in \mathbb{R}^n | y_k - \hat{y}_k(\boldsymbol{\theta}) \in [e_k^-, e_k^+]\}, \tag{4}$$

wobei FPS für „feasible parameter set “steht.

Sollen die Parameter $\boldsymbol{\Theta}_{\text{MF}}$ der Fuzzy-Basisfunktionen mitgeschätzt werden, müssen BE-Methoden für nichtlineare Modelle eingesetzt werden. Eine etablierte Methode für nichtlineare Modelle ist die Mengeninversion mittels Intervallarithmetik (SIVIA) [5], die in einem Branch-and-Bound-Verfahren Zulässigkeitstests durch Anwendung von Intervallarithmetik durchführt. Versuche mit SIVIA konnten aber bereits bei $n = 6$ Parametern und $N = 20$ Datenpunkten auf dem verwendeten Rechner (Intel i5-6500 3,2 GHz CPU, 16 GB RAM) nicht mehr berechnet werden, da der Rechen- und Speicheraufwand exponentiell mit n zunimmt.

Kann die Lage der Teilmodelle durch Vorwissen oder über eine Punktschätzung a-priori festgelegt werden, womit das Schätzproblem linear in den Parametern (LiP) $\boldsymbol{\Theta}_{\text{LM}}$ ist, so können entsprechende BE-Methoden eingesetzt werden können. Als LiP-Modell lässt sich die zulässige Parametermenge als Lösungsmenge von $2N$ Ungleichungen beschreiben:

$$y_k - e_k^- \leq \hat{y}_k(\boldsymbol{\theta}) \leq y_k - e_k^+,\ k = 1, \ldots, N, \tag{5}$$

bzw. als Lösung eines linearen Ungleichungssystems. Dadurch ergibt sich die Lösungsmenge für LiP-Modelle als ein konvexes Polytop.

Zur exakten Lösung des Ungleichungssystems wird bspw. in [5] ein rekursiver Algorithmus beschrieben, bei dem die Eckpunkte des Polytops zusammen mit den Eckpunkten, die auf gemeinsamen Kanten liegen, und den angrenzenden $(n-1)$-dimensionalen Hyperflächen gespeichert werden. Für jedes neue Datum k wird das Polytop $\mathbb{S}_{\mathrm{FPS}}^{k-1}$ mit aus den Ungleichungen resultierenden Hyperebenen geschnitten und die Listen aktualisiert. Alternativ können aus der algorithmischen Geometrie bekannte Verfahren für die Vertex-Enumeration eingesetzt werden. Hierbei werden ebenfalls aus der Beschreibung eines Polytops als lineares Ungleichungssystem die Eckpunkte berechnet. Hierzu zählt die Double-Description-Methode [7] als Verfahren der Vertex-Enumeration, deren Rechenkomplexität exponentiell in n ist und stark von der Anzahl der resultierenden Eckpunkte abhängt. Im Vergleich zu dem selbst implementierten rekursiven Algorithmus konnten mit der Double-Description-Methode in einer effizienten Implementierung nach [7] deutlich kürzere Rechenzeiten erreicht werden.

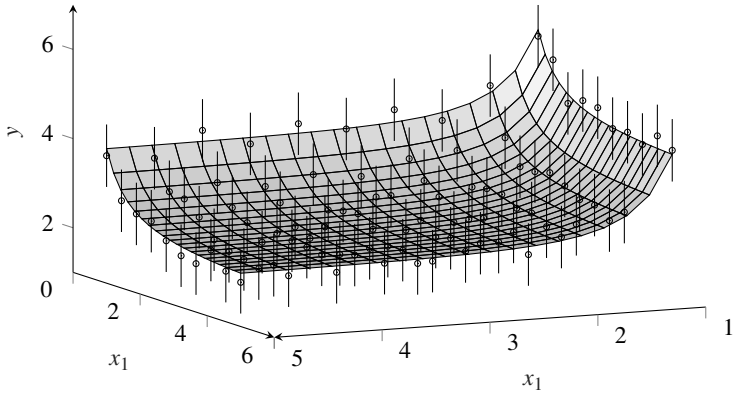
Da die exakten Methoden bei großem n zu sehr komplexen und schwierig zu berechnenden Lösungen führen können, bietet sich der Einsatz von approximativen Methoden zur Beschreibung von $\mathbb{S}_{\mathrm{FPS}}$ an. Bei dem Recursive-Outer-Bounding-Ellipsoid-(ROBE)-Verfahren nach [8] wird eine äußere Approximation als Hyperellipsoid $\hat{\mathbb{E}}_{\mathrm{FPS}} \supset \mathbb{S}_{\mathrm{FPS}}$ berechnet. Dabei wird rekursiv im k-ten Schritt ein volumenminimales äußeres Hyperellipsoid $\hat{\mathbb{E}}_{\mathrm{FPS}}^{k}$ berechnet. Dieses Hyperellipsoid approximiert die Schnittmenge der aus den beiden Ungleichungen resultierenden Hyperebenen $\mathbb{H}_k^+$ sowie $\mathbb{H}_k^-$ mit dem Hyperellipsoid $\hat{\mathbb{E}}_{\mathrm{FPS}}^{k-1}$.

4 Fallstudie

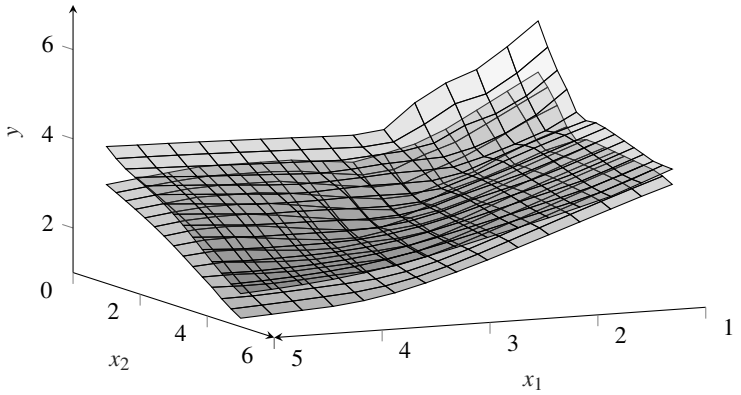
Für eine Fallstudie wurden mit einem Testsystem $y(x_1,x_2) = 2(1+x_1^{-3}+x_2^{-1,5})$ $N = 100$ Datenpunkte in einem äquidistanten Raster erzeugt und mit einem gleichverteilten Rauschen im Intervall $[-0,3;0,3]$ überlagert. In Bild 1a sind die rauschfreie Systemausgabe und die verrauschten Messpunkte mit den exemplarisch vorgegebenen zulässigen Fehlerschranken $[-0,7;0,7]$ dargestellt. Als Modellansatz wurde ein TS-Modell mit $c = 3$ und $\nu = 1,4$ gewählt und eine exakte BE-Parameterschätzung für $\mathbb{S}_{\mathrm{FPS}}$ mit der Double-Description-Methode durchgeführt. In Bild 1b sind die resultierenden begrenzenden Hyperflächen der zulässigen Modellausgabe zu sehen, zwischen denen sich die garantierte Systemausgabe mit der Parametrierung aus $\mathbb{S}_{\mathrm{FPS}}$ bewegt. Zudem wurde für das gleiche Testsystem eine BE-Parameterschätzung des TS-Modells mit dem approximativen ROBE-Verfahren durchgeführt. In Bild 2 ist exemplarisch die Projektion der exakten und der approximierten Parametermenge auf die $(\theta_{1,0},\theta_{1,1})$-Ebene dargestellt. Es kommt dabei zu einer sehr starken Überschätzung der zulässigen Parametermenge.

5 Zusammenfassung und Diskussion

Die Bounded-Error-Schätzung bietet sich als Alternative zu den üblichen statistischen Verfahren zur Punktschätzung bei der datengetriebenen Modellbildung bei Takagi-Sugeno-Multi-Modellen an. Die Vorteile liegen darin, auch bei wenig Vorwissen über den Fehlercharakter interpretierbare Ergebnisse zur Parameterunsicherheit zu erhalten. Zudem lässt die zulässige Parametermenge garantierte Aussagen zum Systemverhalten zu, womit sie sich insbesondere für den Einsatz beim Entwurf robuster Vorsteuerungen und Reglungen eignet. Der Fokus weiterer Untersuchungen liegt darauf, verbesserte Verfahren zur Approximation der zulässigen Parametermenge zu entwickeln, um auch bei hoher Parameteranzahl in angemessener Rechenzeit eine Schätzung zu erhalten. Zudem ist die Anwendung der Methoden auf die Modellierung eines Hartdrehprozesses [9] vorgesehen.



(a) Systemausgabe des Testsystems (Gitterfläche) und Beobachtungen (o) mit zulässigen Fehlerschranken als vertikale Balken



(b) Hyperflächen (Gitterflächen) zwischen denen sich die garantierte Systemausgabe bei Parametrierung des TS-Modells mit $\mathbb{S}_{\mathrm{FPS}}$ bewegt

Bild 1: Ergebnisse der exakten BE-Parameterschätzung für ein TS-Modell des Testsystems

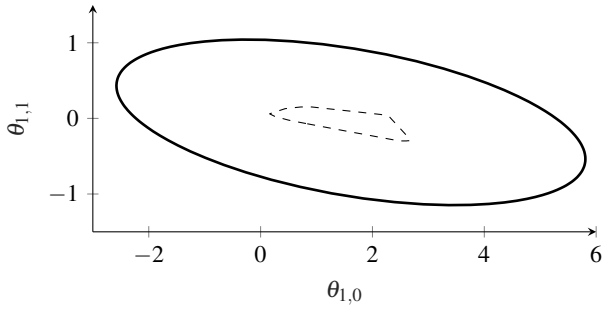


Bild 2: Projektion der exakten zulässigen Parametermenge $\mathbb{S}_{\text{FPS}}$ (--) und der Approximation als Ellipsoid mit ROBE $\hat{\mathbb{E}}_{\text{FPS}}$ (-) auf die $(\theta_{1,0}, \theta_{1,1})$-Ebene

Literatur

[1] E. Walter, L. Pronzato. „Identification of Parametric Models“. Springer, London. 1997.

[2] E. Walter, H. Piet-Lahanier. „Estimation of parameter bounds from bounded-error data: a survey“. In: *Mathematics and Computers in Simulation*, Vol. 32, Nr. 5-6, S. 449–468. 1990.

[3] A. Kroll. „Computational Intelligence. Probleme, Methoden und technische Anwendungen“. De Gruyter Oldenbourg, Berlin. 2016.

[4] F. Schweppe. „Recursive state estimation: Unknown but bounded errors and system inputs“. In: *IEEE Transactions on Automatic Control*, Vol. 13, Nr. 1, S. 22–28. 1968.

[5] L. Jaulin, M. Kieffer, O. Didrit, E. Walter. „Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust Control and Robotics“. Springer, London. 2012.

[6] F. Schweppe. „Fast and robust algorithm to compute exact polytope parameter bounds“. In: *Mathematics and Computers in Simulation*, Vol. 32, Nr. 5-6, S. 481–493. 1990.

[7] K. Fukuda, A. Prodon. „Double description method revisited“. In: *Combinatorics and Computer Science*, S. 91–111. 1996.

[8] E. Fogel, Y.F. Huang. „On the value of information in system identification: Bounded noise case“. In: *Automatica*, Vol. 18, Nr. 2, S. 229–238. 1982.

[9] Wittich F., Gringard M., Kahl M., Kroll A., Niendorf T., Zinn W. „ Zur datengetriebenen Modellierung von Eigenspannungen bei einem Hartdrehprozess“. In: *28. Workshop Computational Intelligence*, S. 61 – 81, KIT Scientific Publishing. 2018.

Entwicklung eines Algorithmus zum Entwurf neuronaler Regler mittels wachsender Netzstrukturen

Simon Harasty, Alessio Cavaterra, Steven Lambeck

Fachbereich Elektrotechnik und Informationstechnik, Hochschule Fulda
Leipziger-Str. 123, 36037 Fulda
E-Mail: {Simon.Harasty, Alessio.Cavaterra, Steven.Lambeck}@et.hs-fulda.de

1 Einführung

Der klassische Entwurf von Reglern für nichtlineare Systeme erfordert die Entwicklung eines genauen Systemmodells, was bei komplexen Systemen einen hohen Entwicklungsaufwand bedeutet. Durch den Einsatz nichtlinearer modellprädiktiver Regelungen mit datengetriebenen Modellen (NL-MPC) kann dieser Aufwand reduziert werden. Allerdings erfolgt diese Reduktion zu Lasten von zusätzlichem Rechenaufwand zur Optimierung zukünftiger Stellsignale, welche zeitkritisch abgeschlossen sein muss [1]. Als Methode mit deterministischem Rechenaufwand bieten sich Neuronale Netze als Regler an, welche anhand eines (datengetriebenen) Systemmodells trainiert werden. Dynamische Systeme mit ausgeprägten Nichtlinearitäten erfordern hierbei eine hohe Komplexität der Netzstruktur mit mehrstufigen rekurrenten Verbindungen. Die Auswahl einer genügend großen Netzstruktur erscheint sinnvoll, da in diesem Fall das vollständig verknüpfte KNN näherungsweise die Eigenschaften eines universellen Approximators aufweist. Solch eine überdimensionierte Struktur ist aber nicht zwangsläufig optimal, weil für das vorliegende Problem möglicherweise überflüssige Neuronen oder Verbindungen vorhanden sind. Um optimale Strukturen zu erreichen, sind reduzierende (Pruning, siehe [2]) Algorithmen eingesetzt worden, um aus einer überdimensionierten Netzstruktur die passende Struktur zu entwickeln. Hierbei ist allerdings die

notwendige Komplexität der Struktur im Voraus nicht bekannt. Wenn neben verschiedenen Netzwerkstrukturen auch rekurrente Verbindungen berücksichtigt werden sollen, wird der Trainingsaufwand sowie die Bestimmung von entfernbaren Elementen extrem aufwändig. Zur Bestimmung von optimalen Strukturen können des Weiteren wachsende (Growing, siehe [3, 4]) Algorithmen eingesetzt werden. Diese bieten den Vorteil, dass sie zunächst mit einfachen Netzstrukturen beginnen (kleiner Rechenaufwand) und erst im fortschreitenden Prozess an Komplexität zunehmen. Eine große Herausforderung bei wachsenden Strukturen ist allerdings die Bestimmung einer sinnvollen Erweiterung des Netzwerks. Gerade bei den zu Beginn überschaubaren Netzwerken bedeuten bereits kleine Änderungen in der Struktur erhebliche Änderungen im Systemverhalten. Eine rein zufällige Erweiterung des Netzwerks führt zu vielen nachteiligen Änderungen, welche die Optimierungszeit wiederum wesentlich erhöhen.

2 Aufbau NEAT

Mit der Entwicklung des NEAT-Algorithmus (NEAT = NeuroEvolution of Augmented Topologies, siehe [5]) steht eine Methode zur Struktur- und Parameteroptimierung von KNN zur Verfügung, welche die Growing- und Pruning-Verfahren in sich vereint. In dem vorliegenden Beitrag wird zunächst der Aufbau des grundlegenden NEAT-Verfahrens erläutert und anschließend die Erweiterung des Verfahrens thematisiert. Anhand eines simulativen Beispiels werden beide Verfahren hinsichtlich verschiedener Bewertungskriterien miteinander verglichen.

2.1 Genetische Codierung

Innerhalb des NEAT Algorithmus wird eine genetische Codierung des zu evolvierenden Netzes genutzt (siehe [6]), welche eine Verfolgung der Innovationen innerhalb des Netzes ermöglicht. Es wird zwischen zwei Gentypen unterschieden. Die einzelnen Knotenpunkte bzw. Neuronen innerhalb des Netzes werden durch Node-Gene bestimmt. Solch ein Gen legt fest, ob es sich um ein

Eingangs- (Sensoren), Ausgangs- (Aktoren) oder verstecktes Neuron handelt. Die Verbindungen innerhalb des Netzwerkes werden mit Connection-Genen angegeben. Ein Connection Gen beschreibt die Punkte der Verbindung innerhalb des Netzwerkes zwischen zwei Neuronen. Durch eine Beschränkung von Verbindungen zu ausschließlich nachfolgenden Neuronen, sowie dem Ausschließen von Eingangsneuronen als Endpunkte und Ausgangsneuronen als Startpunkte der Verbindungen, lässt sich die Form des Netzwerkes als reine Feed-Forward-Struktur limitieren. Des Weiteren wird innerhalb eines Connection-Gens die Gewichtung der Verbindung, bzw. das Eingangsgewicht eines Neurons abgelegt, wie in Bild 1 dargestellt. Jedes Verbindungs-Gen wird mittels einer Innovationsnummer verfolgt, so dass dessen Herkunft verfolgt werden kann und zufällig mutierte Gene mit gleichen Verbindungspunkten dem gleichen Gen zugeordnet werden können. Jedes Gen kann aktiv (dominant) oder inaktiv (rezessiv) sein; rezessive Gene werden wie nicht vorhandene Gene behandelt, bleiben allerdings im Genom enthalten.

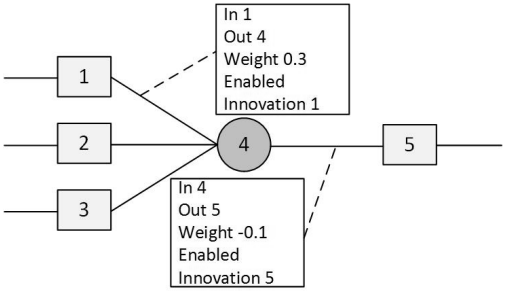


Bild 1: Genetische Codierung

2.2 Spezies

Um Innovationen innerhalb der Entwicklung zu schützen, wird eine Unterteilung der Netzwerkstrukturen in Spezies vorgenommen. Eine einzelne Spezies muss dann nicht global mit allen Individuen einer Generation konkurrieren, sondern lediglich mit seiner eigenen Spezies innerhalb einer Nische. Die Unterteilung der einzelnen Spezies erfolgt über ein Entfernungsmaß δ, welches

über drei gewichtete Eigenschaften der unterschiedenen Genstrukturen berechnet wird. Durch die Innovationsnummern können diese Eigenschaften einfach unterschieden werden. Die Anzahl der entkoppelten (Disjoint) Gene D, die Anzahl an zusätzlichen (excess) Genen E und die durchschnittliche Differenz der Gewichte der geteilten Gene $\overline{W}$ werden zu dem Entfernungsmaß aufaddiert. Zur Normalisierung werden entkoppelte und zusätzliche Gene durch die Anzahl an Genen des größeren Individuums geteilt.

$$\delta = \frac{c_1 \cdot E}{N} + \frac{c_2 \cdot D}{N} + c_3 \cdot \overline{W} \tag{1}$$

Wird nun eine Generation von Individuen festgelegt, wird das erste Individuum der ersten Spezies zugeordnet und repräsentiert nun diese Spezies. Von folgenden Individuen wird nun das Entfernungsmaß zum repräsentativen Individuum berechnet. Überschreitet dieses Entfernungsmaß eine festgelegte Grenze δ_m so wird dieses Individuum einer neuen Spezies zugeordnet und dieses wiederum als Repräsentation dieser Spezies festgelegt.

2.3 Mutation

Bei der Mutation werden mit struktureller Mutation und Verhaltensmutation zwei Arten unterschieden. Bei der strukturellen Mutation kann die Struktur eines Netzes wiederum auf zwei Arten verändert werden. Es kann entweder eine neue Verbindung zwischen zwei Neuronen entstehen oder ein neues Neuron in eine vorhandene Verbindung eingesetzt werden (siehe Bild 2). Das Gen einer neu entstehenden Verbindung wird als dominantes Gen mit einer zufälligen Gewichtung eingeführt. Wird eine Verbindung durch ein neues Neuron erweitert, wird das alte Verbindungsgen rezessiv geschaltet. Die neue Verbindung zum neuen Neuron erhält die Gewichtung 1 und die neue Verbindung vom neuen Neuron zum Endpunkt der alten Verbindung erhält die Gewichtung der alten Verbindung. Hierdurch kann die Auswirkung der Erweiterung des Netzes minimiert werden, um der initialen Verschlechterung des Netzwerkverhaltens entgegen zu wirken. Solche eine neue Verbindung oder Neuron werden als neue Innovation mit Ihren Eigenschaften (Start- und Endpunkte) abgespeichert, um diese im Laufe der Entwicklung nachverfolgen und vergleichen zu können.

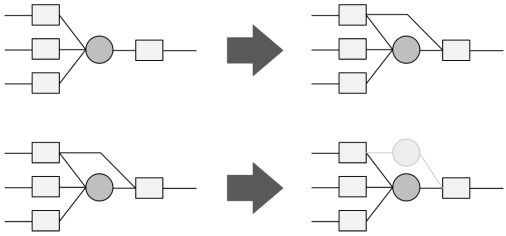


Bild 2: Mutation; oben: neue Verbindung; unten: neues Neuron innerhalb einer Verbindung

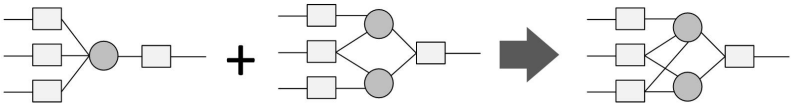


Bild 3: Kombination von zwei Individuen

Jedeneue Verbindung oder jedes neue Neuron wird mit bisherigen Innovationen verglichen und entweder einer bereits vorhandenen Innovation zugeordnet oder als neue Innovation nummeriert. Bei der Verhaltensmutation wird das Gewicht von zufällig bestimmten Verbindungen manipuliert. Ausgewählte Verbindungen werden dabei um einen Wert zwischen 0,01 und 0,2 erhöht oder verringert.

2.4 Rekombination

Bei der Rekombination werden zwei Individuen als Eltern ausgewählt. Gene, die beide Eltern sich teilen, werden zufällig von einem der beiden gewählt. Entkoppelte und zusätzliche Gene werden von dem Elternteil mit besserer Fitness übernommen. Zufällig werden gemeinsame rezessive Gene dominant oder dominante Gene rezessiv gesetzt. In Bild 3 ist ein mögliches Beispiel für eine solche Rekombination aufgeführt.

2.5 Erweiterung der Mutationsstrategien

Bei der Nutzung des Neuronalen Netzes als Regler erfolgt die Bestimmung der Fitness mit dem Ergebnis der Simulation des zu regelnden Systems. Da die Fitness innerhalb einer Nische bzw. einer Spezies nur untereinander verglichen wird, bietet es sich an, innerhalb der einzelnen Spezies verschiedene Schwerpunkte der Weiterentwicklung anzustreben [7]. Hierfür werden nun zur Bestimmung der Fitness unterschiedliche Maße genutzt, welche die Wahrscheinlichkeiten für eine Mutation und die Auswahl von Elternindividuen beeinflussen. Zur Bestimmung der Gesamtperformance wird der *NMSE* (normalized mean square error) genutzt, welcher sich in der Bestimmung von Eltern niederschlägt.

$$NMSE = \frac{\sum_{i=1}^{N} (x_i - w_a)^2}{\sum_{i=1}^{N} (x_i - \bar{x})^2} \tag{2}$$

Des Weiteren wird der maximal auftretende Fehler E_{max} als ein Maß für die Gewichtungen innerhalb eines Netzwerkes genutzt. Sollte der absolute Fehler einer Spezies besonders hoch sein, so erhöht sich die Wahrscheinlichkeit der Mutation der Verbindungsgewichte.

$$E_{max} = \max (w - x_i) \tag{3}$$

Die Tendenz T wird bestimmt, indem bei verschiedenen Arbeitspunkten des Systems ein positives oder negatives Feedback erreicht wird. Die Tendenz wirkt sich auf die Wahrscheinlichkeit der Strukturmutation aus. Ist die Tendenz einer Spezies in einem Arbeitspunkt sehr niedrig, so wird die Wahrscheinlichkeit zur Strukturmutation erhöht. Mit konstanter Sollgröße w ergibt sich hierfür:

$$T = \sum_{i=2}^{N} \frac{y_i - y_{i-1}}{w - x_i} \tag{4}$$

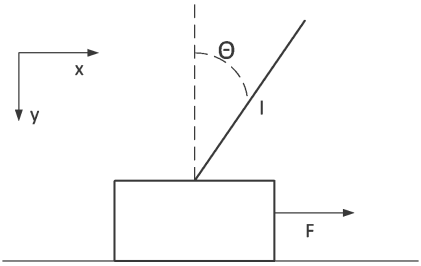


Bild 4: Inverses Pendel auf Wagen

3 Versuchsdurchführung

Um die Algorithmen vergleichen zu können wurde eine Regelstrecke mit nichtlinearem Verhalten gesucht. Als klassisches Benchmark Problem bietet sich das Inverse Pendel an.

3.1 Modell

Als Modell für die verschiedenen Ansätze wird ein inverses Pendel auf einem angetriebenen Wagen genutzt, wie in Bild 4 dargestellt. Hierfür wird das Modell von B. Berger herangezogen [8]. Die hier verwendeten Parameter sind l_e, die effektive Länge des Pendels $\left(\frac{2}{3} \cdot l\right)$, m die Masse des Pendels und μ der Reibungskoeffizient des Pendels. Dann gilt für die Winkelbeschleunigung:

$$\ddot{\Theta} = \frac{1}{l_e}\left(g\sin\Theta - \ddot{x}\cos\Theta - \frac{4\mu}{3ml_e}\dot{\Theta}\right) \tag{5}$$

Die Position des Wagens wird mittels eines $IT1$-Gliedes approximiert. Dieses bildet das Verhalten einer angetriebenen Masse mit einem Motor hinreichend ab. Der Einfluss der Masse des Wagens schlägt sich hierbei in der Zeitkonstanten τ nieder. Für die Beschleunigung des Wagens in x-Richtung wird daher angenommen:

$$\ddot{x} = -\frac{1}{\tau}\dot{x} + \frac{K}{\tau}u \tag{6}$$

3.2 Durchführung

Zum Vergleich der beiden Algorithmen wurden 20 Variationen des Pendelmodells mit zufällig gewählten Systemparametern erstellt. Die Länge des Pendels wurde zwischen 10 cm und 1 m, die Masse des Pendelstabs zwischen 1 g und 200 g, der Startwinkel zwischen $\pm 45°$, der Reibungskoeffizient des Pendels zwischen 0,1 % und 2 % und die Zeitverzögerung des $IT1$-Glieds zwischen 0,01 s und 0,5 s. Die Simulationszeit betrug 20s. Da zufällige Parameter bestimmt wurden, wurde eine uneingeschränkte Stellgröße zugelassen. Die Individuen der Startpopulation für beide Algorithmen wurde ebenfalls für jeden Fall einmal anhand der Mutationsregeln erstellt und in beiden Algorithmen gleich genutzt. Die Größe der Population betrug hierbei 200 Individuen. Es wurde die Wahrscheinlichkeit für die strukturelle Mutation auf 70 % und für Verhaltensmutation auf 90 % erhöht um eine möglichst hohe Diversität in der Startpopulation zu erhalten. Das Problem wurde als gelöst angesehen, wenn nach 10 s der maximale Fehler unterhalb von $1°$ lag. Zur Bestimmung der Fitness der Individuen wurde der *RMSE* von einem Winkel um $0°$ gewählt, um das Balancieren des Pendels zu gewährleisten. Um die Durchführung zu beschleunigen und den Evolutionsdruck zum Balancieren des Pendels zu erhöhen, wurde eine Abbruchbedingung eingeführt. Wenn der Winkel des Pendels $\pm 90°$ überschritt, wird die Simulation abgebrochen und der restliche Fehler als maximal angenommen. Eine Lösung des Problems konnte mit beiden Algorithmen bei jedem Versuch vor Erreichen der maximalen Generationenanzahl von 10.000 erreicht werden. Hierfür sind exemplarisch jeweils das beste Systemverhalten in Bezug auf den zu regelnden Winkel in Bild 5 aufgeführt. Hierbei waren die gewählten Parameter: $\Theta_0 = -18°$; $l = 22{,}3\,\text{cm}$; $\mu = 1{,}2\%$; $\tau = 55\,\text{ms}$.

Im Durchschnitt benötigte der NEAT Algorithmus 1408 Generationen und der adaptierte NEAT 1053 Generationen. In jedem Fall erreichte der adaptierte NEAT die Lösung schneller und zwar durchschnittlich mit $23{,}91\,\%$ weniger Generationen. Die Ergebnisse der einzelnen Durchläufe sind in Tabelle 1 aufgeführt.

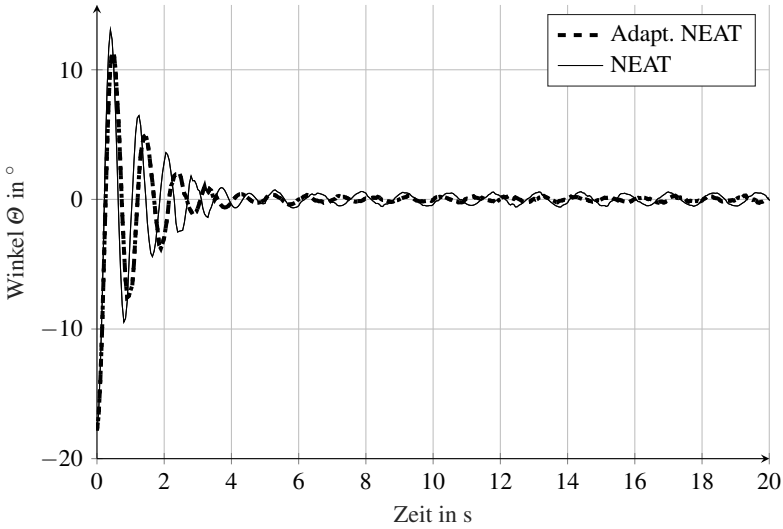


Bild 5: Zeitverhalten der besten Lösungen beider Algorithmen

Durch die adaptierte Mutationsstrategie konnten schneller vorteilhafte Strukturen innerhalb des Netzwerks entwickelt werden. Dies wird deutlich, wenn man die Entwicklung des besten Individuums der jeweiligen Generation betrachtet. Der Durchschnittliche *RMSE* der besten Individuen, dargestellt in Bild 6, verdeutlicht diese Entwicklung.

4 Fazit und Ausblick

In dem hier vorgestellten Beitrag konnte der Vorteil einer adaptierten Mutationsstrategie bei der Verwendung des NEAT Algorithmus gezeigt werden. Die Verwendung solch eines Algorithmus ist vor allem bei einer unbekannten Komplexität des zu regelnden Systems sinnvoll, wenn darüber hinaus eine möglichst kleine Dimension des Netzwerks zur Einsparung von Ressourcen relevant ist. Durch die einfache Erweiterung der Mutationswahrscheinlichkeit für Strukturen und Systemverhalten durch zusätzliche Fehlermaße, konnte die Anzahl an nötigen Simulationen und damit die Zeit zur Erstellung einer Lösung wesentlich verringert werden. In zukünftigen Arbeiten ist die Implementierung

von unterschiedlichen Aktivierungsfunktionen innerhalb des Netzwerkes geplant, deren Mutationswahrscheinlichkeit ebenfalls in Abhängigkeit von Fehlermaßen bestimmt werden soll.

Tabelle 1: Abbruchsgenerationen

Nr	Generationen NEAT	Generationen adaptiert	Faktor
1	1683	1083	0,3565
2	1089	956	0,1221
3	1232	1018	0,1737
4	1708	1179	0,3097
5	1295	1183	0,0865
6	1289	1016	0,2118
7	1581	1050	0,3359
8	1083	990	0,0859
9	1162	886	0,2375
10	1177	1050	0,1079
11	1664	1044	0,3726
12	1147	916	0,2014
13	1741	1131	0,3504
14	1472	1077	0,2683
15	1317	1111	0,1564
16	1297	1051	0,1897
17	1574	1012	0,3571
18	1462	1122	0,2326
19	1713	1194	0,3030
20	1475	998	0,3234

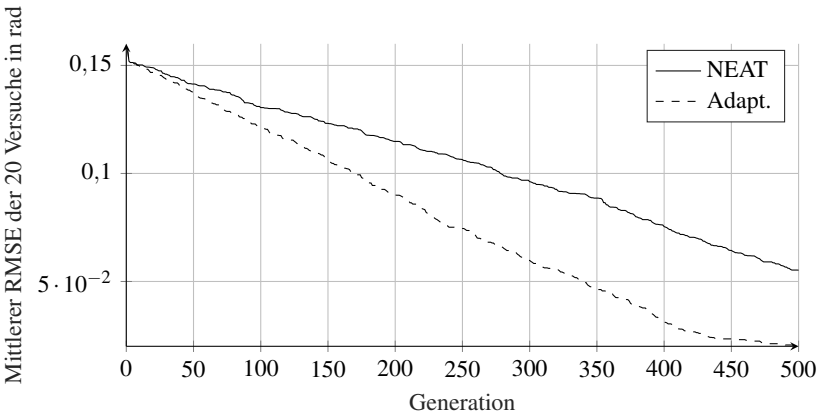


Bild 6: Mittelwert der besten Individuen über die ersten 500 Generationen

Literatur

[1] G. De Souza, D. Odloak, A. Zanin. „Real time optimization (RTO) with model predictive control (MPC)“. In: *Computers & Chemical Engineering, Volume 34, Issue 12*, S. 1999-2006. IEEE 2010.

[2] C. L. Giles, C. W. Omlin. „Pruning recurrent neural networks for improved generalization performance“. In: *IEEE Transactions on Neural Networks, Volume 5 , Issue 5*, S. 848-851. 1994.

[3] S. Nolfi, D. Parisi. „Growing Neural Networks“. In: *The HANDBOOK OF BRAIN THEORY AND NEURAL NETWORKS*, S. 431-434. Addison-Wesley 1991.

[4] A. Sakar, R. J. Mammone. „Growing and pruning neural tree networks“. In: *IEEE Transactions on Computers, Volume 42, Issue 3* , S. 291-299. IEEE 1993.

[5] K. O. Stanley, R. Miikkulainen. „Evolving Neural Networks through Augmented Topologies“. In: *Evolutionary Computation* S. 99-127. MIT Press Journal 2002.

[6] D. Whitley, T. Starkweather, C. Bogart. „Genetic algorithms and neural networks: optimizing connections and connectivity“. In: *Parallel Computing* S. 347-361. Elsevier 1990.

[7] S. Whiteson, P. Stone, K. O. Stanley, R. Mikkulainen, N. Kohl. „Automatic feature selection in neuroevolution“. In: *GECCO '05 Proceedings of the 7th annual conference on Genetic and evolutionary computation* S. 1225-1232. ACM 2005.

[8] B. Berger. „Realisierung einer prototypischen Hardwarelösung fur ein inverses Pendel“. Diplomarbeit, S. 50-55. Technische Universität Chemnitz 2004.

Inkrementelle Modellbildung von statischen Prozessen auf Basis von Latin Hypercube Designs

Tim Voigt[1], Martin Kohlhase[1], Oliver Nelles[2]

[1]Center for Applied Data Science Gütersloh, FH Bielefeld
Schulstraße 10, 33330 Gütersloh, Germany
E-Mail: {tim.voigt, martin.kohlhase}@fh-bielefeld.de

[2]Mess- und Regelungstechnik - Mechatronik, Universität Siegen
Paul-Bonatz-Str. 9-11, 57068 Siegen, Germany
E-Mail: oliver.nelles@uni-siegen.de

1 Einführung

Die Optimierung bestehender Produktionsanlagen gewinnt im Zuge einer ganzheitlichen Einführung von Industrie 4.0 Technologien in produzierenden Unternehmen zunehmend an Bedeutung. Zu diesem Zweck werden Modelle benötigt, die das Verhalten produktiv eingesetzter Anlagen abbilden. Da Anlagen meist in festen Arbeitspunkten betrieben werden, ist der Informationsgehalt der Messdaten gering. Zur Steigerung des Informationsgehalts und Erzielung einer hohen Modellgüte ist der Prozess entsprechend anzuregen, wozu Methoden der Versuchsplanung verwendet werden können [7].

Übliche Verfahren der Versuchsplanung sind an produktiv betriebenen Fertigungsanlagen nur bedingt einsetzbar. Versuchspläne werden zumeist im Vorfeld der Versuche definiert und regen den gesamten Eingangsraum an. Ob ein Versuchsplan jedoch zu einer Überschreitung des zulässigen Eingangsraums führt, ist bei vielen variierten Eingangsgrößen im Vorfeld schwer zu überblicken. In der Praxis wird daher häufig die einfacher zu überwachende One-factor-at-a-time Methode angewandt, bei der nacheinander einzelne Eingangs-

größen variiert und deren Auswirkungen beobachtet werden. Diese Methode führt jedoch im Allgemeinen zu einer hohen Anzahl an benötigten Versuchspunkten und zu einer unzureichenden Abdeckung des Eingangsraums [1]. Das nachfolgend beschriebene Verfahren ermöglicht eine schrittweise Erschließung des Prozessverhaltens und vereinfacht damit die Überwachung der Einhaltung des zulässigen Betriebsbereichs. Gleichzeitig wird durch eine systematische Generierung der Versuchspunkte die Anzahl an Datenpunkten klein gehalten und der Eingangsraum gleichmäßig abgedeckt, wobei eine inkrementelle Versuchsplanung und Modellbildung vereint werden.

Beginnend mit einer Eingangsgröße, wird die Anzahl an betrachteten Eingangsgrößen sukzessiv erweitert. Der Eingangsraum wird damit, ausgehend von einem vorgegebenen Arbeitspunkt, schrittweise erschlossen und das Prozessverhalten durch ein Modell abgebildet. Auf diese Weise bietet diese Methodik folgende Vorteile:

- Sukzessive Abbildung aller Einflüsse und Wechselwirkungen von Eingangsgrößen auf einen Prozessausgang,
- Erzielung einer hohen Modellgüte, in einem Bereich um einen Arbeitspunkt, der besonders relevant für den Betrieb eines Prozesses ist,
- gleichbleibende Modellgüte im Arbeitspunkt, trotz Erweiterung des Modells und
- vereinfachte Erkennung kritischer Betriebsbereiche.

Das inkrementelle Vorgehen zur Modellbildung mit einem parallel zur Modellstruktur aufgebauten Versuchsplan wird anhand verschiedener Testfunktionen und unterschiedlich verteilter Generalisierungsdaten demonstriert. Zur Validierung des Verfahrens wird ein in einem Schritt erstellter Versuchsplan mit derselben Anzahl an Punkten herangezogen, auf dem ein globales Modell geschätzt wird.

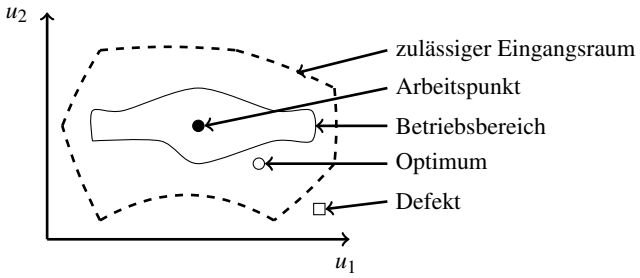


Bild 1: Qualitative Darstellung des Arbeitsbereichs und des zulässigen Eingangsraums eines Prozesses.

2 Methodik

Bei bestehenden Produktionsanlagen sind durch den laufenden Betrieb Informationen verfügbar, die für die Versuchsplanung genutzt werden können. Üblicherweise existiert ein Arbeitspunkt, der bei der Anlagenauslegung vorgegeben wurde. Durch Prozessschwankungen ergibt sich ein Betriebsbereich um den Arbeitspunkt, in dem der Prozess produktiv betrieben wird. Des Weiteren existieren Grenzen des Eingangsraums, die durch Stellgrößenbegrenzungen und den zulässigen Wertebereich der resultierenden Ausgangsgrößen des Prozesses festgelegt sind. Werden diese Grenzen überschritten, kann dies zu einem fehlerhaften Prozessverhalten oder sogar zu einem Defekt der Anlage führen. Bild 1 zeigt schematisch den Eingangsraum eines Prozesses mit zwei Eingangsgrößen. Ein gesuchtes Optimum findet sich aufgrund der Auslegung der Anlage in der Nähe des Arbeitspunktes. Aus diesem Grund ist dieser Bereich besonders relevant für die Modellierung.

Die verschiedenen Eingangsgrößen des Prozesses unterscheiden sich in ihrem Einfluss auf den Prozessausgang. Ein nutzbares Prozessmodell bildet insbesondere die Einflüsse der bedeutendsten Eingangsgrößen mit einer hohen Güte ab. Aus diesem Grund wird im Folgenden eine Rangfolge der Eingangsgrößen verwendet. In dieser Rangfolge sind die p Eingangsgrößen u_j mit $j \in \{1, 2, \ldots, p\}$, gemäß ihres Einflusses im Arbeitspunkt, absteigend sortiert. Die Größe u_1 hat den größten Einfluss auf den Prozess, während die weiteren Eingangsgrößen mit ansteigendem Eingangsindex j an Bedeutung verlieren. Die Sortierung der

Eingangsgrößen kann einerseits auf Grundlage von Prozesswissen und dem bekannten Betriebsbereich (Bild 1) erfolgen. Andererseits können automatische Verfahren verwendet werden, die auf Grundlage verschiedener Kriterien, wie z.B. der Korrelation zwischen Ein- und Ausgangsgrößen, eine Rangordnung von Eingangsgrößen erzeugen [5].

Die Modellierung des Prozesses wird in mehreren Schritten durchgeführt. Dabei wird in jedem Schritt eine weitere Eingangsgröße hinzugefügt und dadurch der Eingangsraum des Modells erweitert. In jedem der $p^* \leq p$ Vermessungs- und Modellierungsschritte $i \in \{1,2,\ldots,p^*\}$ werden die in Bild 2 dargestellten Aktionen durchgeführt. So wird für alle betrachteten Eingangsgrößen ein Versuchsplan in Form eines Latin Hypercubes (LHC) [9] aufgestellt und dieser anschließend hinsichtlich seiner Raumabdeckung optimiert. Auf Grundlage der generierten Versuchspunkte werden Versuche durchgeführt und die Messdaten erhoben. Anschließend wird ein neues Teilmodell geschätzt und dieses additiv zu dem Gesamtmodell hinzugefügt. Schließlich erfolgt eine Generalisierung zur Beurteilung der Modellgüte und zur Bewertung des vorgestellten Verfahrens. Dieses Vorgehen wird für p^* Eingangsgrößen durchgeführt.

2.1 Inkrementelle Modellbildung

Die schrittweise Erschließung des Prozessverhaltens, ausgehend von einem vorgegebenen Arbeitspunkt, spiegelt sich auch in der Modellbildung wieder. Das Modell wird in jedem Schritt angepasst, um die Einflüsse der zusätzlichen Eingangsgrößen zu berücksichtigen. Dazu wird eine Modellstruktur verwendet, die inkrementell erweitert wird. Als Gesamtmodell ergibt sich ein statisches MISO-Modell (Multiple Input - Single Output), das sich aus mehreren Teilmodellen zusammensetzt.

Die gewählte Modellstruktur ist in Bild 3 dargestellt. In jedem Schritt i wird ein weiterer Eingang u_i hinzugefügt und das bestehende Gesamtmodell um ein Teilmodell f_i mit allen aktuell ausgewählten Eingangsgrößen $(u_1, u_2, \ldots, u_i)$ erweitert. Eine Überlagerung von Teilmodellen kann prinzipiell auf verschiedene Arten erfolgen, wie zum Beispiel durch Addition oder Multiplikation. In dieser

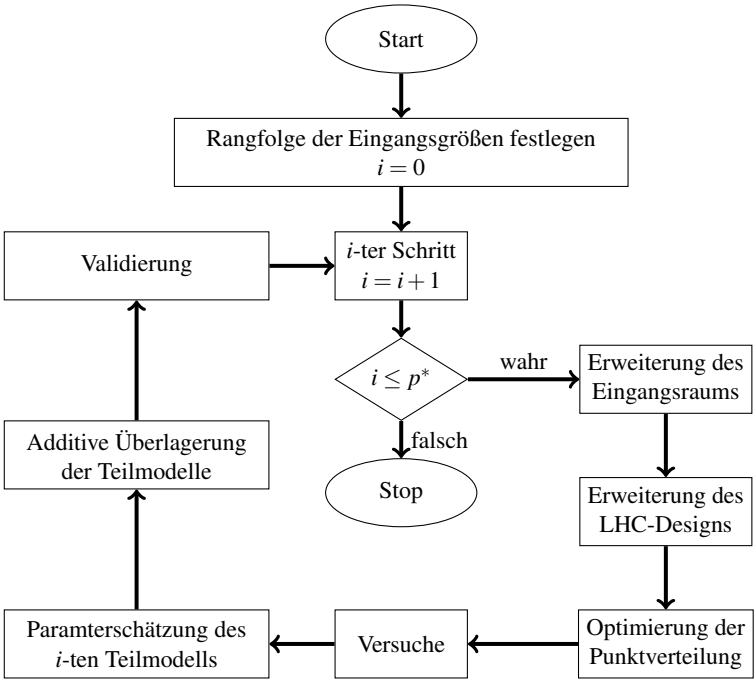


Bild 2: Ablauf des Verfahrens zur inkrementellen Modellbildung.

Arbeit werden alle Teilmodelle additiv überlagert, da aus dieser Überlagerungsart eine einfache Modellstruktur resultiert, die alle auftretenden Wechselwirkungen abbilden kann. Jedes zusätzliche Teilmodell bildet den Einfluss der neu hinzugefügten Eingangsgröße auf das Prozessverhalten und die Wechselwirkungen mit den weiteren Eingangsgrößen ab. Das erste Teilmodell enthält somit nur 1-fach-Interaktionen, das Zweite zusätzlich 2-fach-Interaktionen, das Dritte zusätzlich 3-fach-Interaktionen, usw.

Der Ausgang des Gesamtmodells

$$\hat{y} = \sum_{j=1}^{p^*} \hat{y}_j \tag{1}$$

ergibt sich aus der Addition der p^* Teilmodellausgänge $\hat{y}_j$.

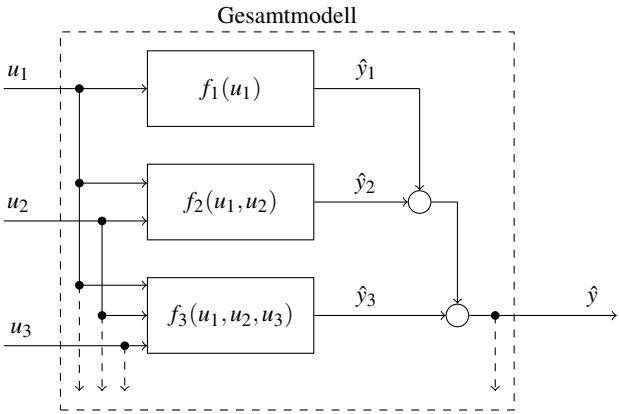


Bild 3: Struktur des Gesamtmodells bestehend aus additiv überlagerten Teilmodellen für den dritten Modellierungsschritt $i = 3$.

Die gewählte Vorgehensweise verwendet ähnlich wie Boosting-Verfahren schrittweise erstellte Modelle, die zu einem Gesamtmodell überlagert werden. Beim Boosting wird in jedem Schritt ein neues Modell mit derselben Struktur geschätzt, das den Fehler der zuvor erstellten Modelle minimiert [4]. Im Unterschied zum Boosting wird bei der gewählten Vorgehensweise in jedem Schritt die Modellstruktur erweitert und außerdem eine weitere Eingangsgröße mit neuen Informationen für das Modell verwendet.

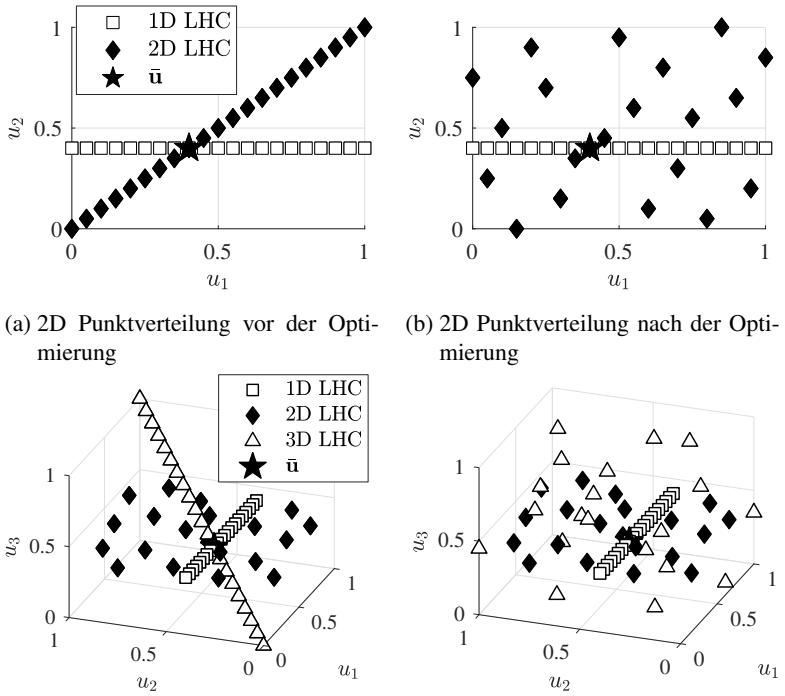
2.2 Versuchsplanung

Zur Erstellung der Teilmodelle wird ein geeigneter Versuchsplan benötigt, der als Datengrundlage für die Schätzung verschiedener Modellstrukturen der Teilmodelle verwendet werden kann. Des Weiteren ist in vielen realen Anwendungen kein Wissen über die Art des Einflusses einer Eingangsgröße auf den Prozessausgang gegeben, so dass im Allgemeinen von einem nichtlinearen Zusammenhang ausgegangen wird. Aus diesen Gründen ist es zweckmäßig, eine gleichmäßige Abdeckung des Eingangsraums mit Versuchspunkten (Space Filling Design) zu erzeugen [8].

Durch die Erweiterung um eine Eingangsgröße in jedem Schritt, wächst das Hypervolumen des Eingangsraums exponentiell an (Fluch der Dimensionalität). Würde bei der Erweiterung des Eingangsraums die initiale Dichte der Versuchspunkte beibehalten werden, würde dies zu einem exponentiellen Anstieg der benötigten Versuchspunkte führen. Gemäß der ermittelten Rangordnung der Eingangsgrößen, nimmt deren Einfluss und damit auch der Beitrag zum Gesamtfehler in jedem Schritt ab. Aus diesem Grund können die Einflüsse weiterer Eingangsgrößen weniger detailliert abgebildet werden, ohne die Gesamtmodellgüte signifikant zu beeinträchtigen. Im Folgenden wird für jeden Schritt, trotz exponentiell ansteigendem Hypervolumen des Eingangsraums, die gleiche Anzahl an Versuchspunkten betrachtet. Auf diese Weise steigt der Aufwand zur Vermessung linear mit der Anzahl der betrachteten Eingangsgrößen.

In jedem Schritt soll eine Gleichverteilung der Versuchspunkte im betrachteten i-dimensionalen Eingangsraum erzeugt werden. Dazu wird ein Latin Hypercube Design verwendet. Ein LHC mit N Versuchspunkten unterteilt jede der i Dimensionen in N gleich große Bereiche (Levels), in denen jeweils nur ein Punkt liegt [9]. Bild 4 stellt die resultierenden Punktverteilungen im zweiten und dritten Schritt bei der inkrementellen Modellbildung dar. Im Schritt $i = 1$ ergibt sich eine äquidistante Punktverteilung (Quadrate) im eindimensionalen Eingangsraum. Diese verläuft durch den Arbeitspunkt $\bar{\mathbf{u}}$ und parallel zur Achse u_1. Im zweiten Schritt werden die neu hinzugefügten Punkte (Rauten) entlang der ersten Hauptdiagonale initialisiert und erfüllen damit die LHC Eigenschaften (Bild 4a). Da die neu hinzugefügten Punkte den Raum nicht gleichmäßig abdecken, wird eine Optimierung der Punktverteilung benötigt, deren Ergebnis in Bild 4b zu sehen ist. Bild 4c und Bild 4d stellen die Punktverteilungen mit den neu hinzugefügten Punkten (Dreiecke) vor und nach der Optimierung im Schritt $i = 3$ dar.

Zur Optimierung des LHC wird der Extended Deterministic Local Search (EDLS) Algorithmus [2] verwendet. Dieser Algorithmus basiert auf dem *Maximin*-Kriterium. Dieses Kriterium maximiert den minimalen Abstand zwischen den Versuchspunkten, indem wiederholt Koordinaten zweier Punkte in einer Dimension vertauscht werden. Durch ein erweitertes Abbruchkriterium erzielt dieser Algorithmus einen besonders großen Abstand zwischen benach-



(a) 2D Punktverteilung vor der Optimierung

(b) 2D Punktverteilung nach der Optimierung

(c) 3D Punktverteilung vor der Optimierung

(d) 3D Punktverteilung nach der Optimierung

Bild 4: Resultierende Punktverteilungen im zweiten a), b) und dritten Schritt c), d) für $N = 21$ Punkte.

barten Punkten. Bei der Optimierung des Versuchsplans können außerdem alle bereits vermessenen Punkte aus den vorherigen Schritten berücksichtigt werden. Somit wird auch der minimale Abstand zu den vorhandenen Punkten in dem zuvor betrachteten Unterraum maximiert.

Der vorgegebene Arbeitspunkt bleibt in jedem Schritt ein Punkt des neu erstellten LHC. Um einen LHC mit N Punkten zu erhalten, werden jeweils $N-1$ zusätzliche Punkte vermessen. Für den Schritt i ergibt sich folglich die Gesamtzahl der vermessenen Punkte

$$N_i = 1 + i \cdot (N-1). \tag{2}$$

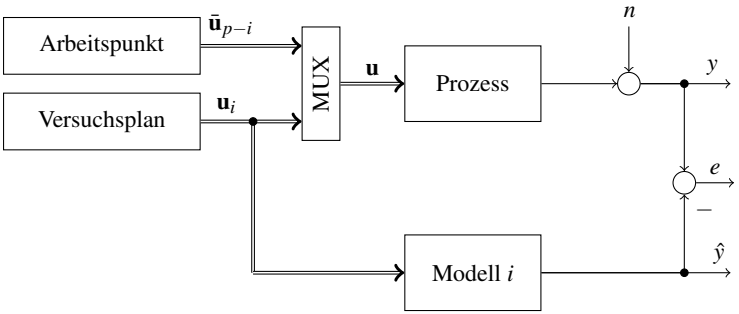


Bild 5: Ein- und Ausgangsgrößen für die Schätzung des i-ten Teilmodells.

2.3 Parameterschätzung

Auf Grundlage des erstellten Versuchsplans und den durchgeführten Messungen kann in jedem Schritt ein Modell geschätzt, bzw. das bestehende Modell erweitert werden. Die betrachteten Ein- und Ausgangsgrößen für die Schätzung des i-ten Teilmodells sind in Bild 5 dargestellt. Ziel ist es, den Prozessausgang y mit einem Gesamtmodell mit der Ausgangsgröße $\hat{y}$ zu approximieren und somit den Fehler $e = y - \hat{y}$ zu minimieren. Das Rauschen n wird als weißes Rauschen mit wählbarer Rauschleistung angenommen, das additiv auf den Systemausgang wirkt.

Alle p Eingangsgrößen des betrachteten Prozesses sind im Eingangsgrößenvektor $\mathbf{u} = [u_1, u_2, \ldots, u_p]^\mathrm{T}$ zusammengefasst. Diese Eingangsgrößen sind gemäß der ermittelten Rangfolge sortiert. Zu Beginn wird ein Prozessmodell mit nur einer Eingangsgröße erstellt und dieses anschließend inkrementell um eine zusätzliche Eingangsgröße in jedem Schritt erweitert. Entsprechend werden bei der Versuchsplanung auch nur die ersten i Eingangsgrößen $\mathbf{u_i} = [u_1, u_2, \ldots, u_i]^\mathrm{T}$ systematisch variiert. Alle weiteren $p - i$ Eingangsgrößen verbleiben auf den konstanten Werten eines gegebenen Arbeitspunkts $\bar{\mathbf{u}}_{p-i} = [\bar{u}_{i+1}, \bar{u}_{i+2}, \ldots, \bar{u}_p]^\mathrm{T}$. Im Eingangsgrößenvektor des Prozesses $\mathbf{u} = [u_1, \ldots, u_i, \bar{u}_{i+1}, \ldots, \bar{u}_p]^\mathrm{T}$ wird in jedem weiteren Modellierungsschritt eine zuvor konstant gehaltene Größe durch die Versuchsplanung systematisch variiert.

2.4 Teilmodellstruktur

Der erstellte Versuchsplan ermöglicht durch das verwendete Space-Filling-Design die Schätzung verschiedener Modellstrukturen der Teilmodelle. In dieser Arbeit werden Polynommodelle

$$\hat{y} = b_0 + b_1 u_1 + b_2 u_2 + \ldots + b_i u_i + b_{i+1} u_1^2 + b_{i+2} u_1 u_2 + \ldots + b_{M_i} \cdot u_i^m \quad (3)$$

des Grades m mit M zu bestimmenden Parametern verwendet.

Die betrachtete Methodik zielt darauf ab, eine besonders hohe Güte im Arbeitspunkt zu erreichen. Zu diesem Zweck könnte eine Gewichtung der einzelnen Punkte eingeführt und das Weighted Least Squares Verfahren (WLS) für die Parameterschätzung verwendet werden. Auf diese Weise können Polynommodelle, wie bei der lokalen polynomialen Regression, auf bestimmte Bereiche des Eingangsraums angepasst werden [3]. Bei dem inkrementellen Vorgehen soll das i-te Teilmodell die Einflüsse der neu hinzugefügten Eingangsgröße sowie deren Wechselwirkungen mit anderen Eingangsgrößen abbilden und bereits modellierte Zusammenhänge aussparen. Dieses Verhalten kann durch eine entsprechend gewählte Gewichtung näherungsweise erreicht werden. Punkte aus den vorherigen Schritten der Versuchsplanung würden dabei geringer gewichtet werden.

Da trotz der Gewichtung alle additiv überlagerten Modelle einen Einfluss auf den Arbeitspunkt haben, wird ein anderes Vorgehen gewählt. Es werden in jedem Schritt nur die Punkte des neu erstellten LHC betrachtet und gleichgewichtet in die Parameterschätzung einbezogen. Der inkrementelle Ansatz basiert darauf, dass die einzelnen Teilmodelle in dem i-dimensionalen Eingangsraum, für den sie geschätzt wurden, die höchste Modellgüte erreichen. Der dazu verwendete Versuchsplan deckt diesen Eingangsraum, der ein Unterraum des insgesamt betrachteten p dimensionalen Raums ist, mit einer hohen Punktdichte ab. Die Überlagerung mit weiteren additiven Teilmodellen, die im Allgemeinen einen Restfehler aufweisen, könnte die Gesamtmodellgüte in diesem Unterraum zunehmend herabsetzen. Aus diesem Grund wird eine Modellstruktur benötigt, die sicherstellt, dass die Gesamtmodellgüte im zuvor

betrachteten Unterraum durch die additive Überlagerung weiterer Teilmodelle nicht verringert wird.

Das Vorgehen wird exemplarisch für ein Polynom zweiten Grades aufgezeigt, kann aber analog für Polynome höheren Grades umgesetzt werden. Im ersten Schritt ergibt sich mit

$$\hat{y}_1 = b_0 + b_1 \cdot u_1 + b_2 \cdot u_1^2 \tag{4}$$

ein Polynommodell zweiten Grades für die erste Eingangsgröße u_1. Für jedes weitere Teilmodell wird ein Basismodell gewählt, das mit einem Linearfaktor multipliziert wird. So ergibt sich im zweiten Schritt mit

$$\begin{aligned}\hat{y}_2 &= \underbrace{\left(b_0 + b_1 \cdot u_1 + b_2 \cdot u_2\right)}_{\text{Basismodell}} \underbrace{\left(u_2 - \bar{u}_2\right)}_{\text{Linearfaktor}} \\ &= (-b_0 \cdot \bar{u}_2) + (-b_1 \cdot \bar{u}_2) \cdot u_1 + (b_0 - b_2 \cdot \bar{u}_2) \cdot u_2 \\ &\quad + (b_1 - b_3 \cdot \bar{u}_2) \cdot u_1 \cdot u_2 + b_2 \cdot u_2^2 \end{aligned} \tag{5}$$

ein Funktionsverlauf, der entlang der in Schritt $i = 1$ vermessenen Gerade durch den Arbeitspunkt den Wert Null aufweist und drei zu schätzende Parameter $\hat{\boldsymbol{\Theta}}_2 = [b_0, b_1, b_2]^{\mathrm{T}}$ besitzt. Als Basismodell wird ein Polynommodell betrachtet, das im Grad um eins reduziert wird. Eine anschließende Multiplikation mit dem Linearfaktor $(u_i - \bar{u}_i)$ erhöht den Grad des Teilmodells wieder um eins. Dieser Linearfaktor erzwingt für Teilmodelle mit $i \geq 2$ in dem zuvor betrachteten Unterraum einen Wert von Null. Aus einer Überlagerung der Teilmodelle resultiert ein Gesamtmodell $\sum_{j=1}^{i} \hat{y}_j$, das eine polynomiale Struktur m-ten Grades aufweist. Damit diese Bedingung für Polynommodelle beliebigen Grades erfüllt ist, müssen die einzelnen Basismodelle entsprechend angepasst werden. Auf diese Weise wird eine handhabbare Anzahl an Parametern und damit auch an benötigten Datenpunkten erzielt. Des Weiteren können höhere Potenzen und Wechselwirkungen der Eingangsgrößen, wie sie ohne die Reduzierung des Grads des Basismodells auftreten würden, in vielen praktischen Anwendungen vernachlässigt werden [7].

Analog wird das Polynommodell im dritten Modellierungsschritt

$$\begin{aligned}\hat{y}_3 &= \underbrace{\left(b_0 + b_1 \cdot u_1 + b_2 \cdot u_2 + b_3 \cdot u_3 + b_4 \cdot u_1 \cdot u_2\right)}_{\text{Basismodell}} \underbrace{\left(u_3 - \bar{u}_3\right)}_{\text{Linearfaktor}} \\ &= b_0 \cdot \left[\left(u_3 - \bar{u}_3\right)\right] + b_1 \cdot \left[u_1 \cdot \left(u_3 - \bar{u}_3\right)\right] + b_2 \cdot \left[u_2 \cdot \left(u_3 - \bar{u}_3\right)\right] \\ &\quad + b_3 \cdot \left[u_3 \cdot \left(u_3 - \bar{u}_3\right)\right] + b_4 \cdot \left[u_1 \cdot u_2 \cdot \left(u_3 - \bar{u}_3\right)\right] \end{aligned} \tag{6}$$

so erweitert, dass in der für $i = 2$ vermessenen Ebene der Teilmodellausgang den Wert Null annimmt. Das Basismodell wurde in (6) so gewählt, das sich aus der Überlagerung mit den beiden vorherigen Teilmodellen ein vollständiges Polynommodell zweiten Grades mit drei Eingangsgrößen ergibt. In der Darstellung in Gleichung (6) wird deutlich, dass die Teilmodelle linear in den Parametern sind. Die zu bestimmenden M_i Parameter

$$\hat{\boldsymbol{\Theta}}_i = [b_0, b_1, \ldots, b_{M_i}]^{\mathrm{T}} \tag{7}$$

werden in jedem Modellierungsschritt für jedes Teilmodell separat mit dem Least Square Verfahren [3] geschätzt. Dazu wird der quadratische Modellfehler minimiert. Der Modellfehler

$$\mathbf{e} = \begin{cases} \mathbf{y} - \hat{\mathbf{y}}_1 = \mathbf{y} - \mathbf{X}_1 \hat{\boldsymbol{\Theta}}_1 & \text{für } i = 1 \\ \mathbf{y} - \underbrace{\sum_{j=1}^{i} \hat{\mathbf{y}}_j}_{\hat{\mathbf{y}}} = \underbrace{\left(\mathbf{y} - \sum_{j=1}^{i-1} \hat{\mathbf{y}}_j\right)}_{\tilde{\mathbf{y}}} - \hat{\mathbf{y}}_i = \tilde{\mathbf{y}} - \mathbf{X}_i \hat{\boldsymbol{\Theta}}_i & \text{für } 1 < i \leq p \end{cases} \tag{8}$$

ergibt sich in Vektorschreibweise aus dem Ausgang des zu schätzenden Teilmodells $\hat{\mathbf{y}}_i$ und dem gemessenen Prozessausgang $\mathbf{y}$. Höhere Teilmodelle $i > 1$ bilden die Differenz zwischen dem Prozessausgang $\mathbf{y}$ und dem vorhandenen niederdimensionalen Prozessmodell $\sum_{j=1}^{i-1} \hat{\mathbf{y}}_j$ ab, die im Vektor $\tilde{\mathbf{y}}$ zusammengefasst ist. $\mathbf{X}_i$ ist die Regressionsmatrix, die aus der gewählten Modellstruktur und den im i-ten Versuch vermessenen Punkten resultiert. Die Lösung des Minimierungsproblems ergibt sich dementsprechend zu

$$\hat{\boldsymbol{\Theta}}_i = \begin{cases} (\mathbf{X}_1^{\mathrm{T}} \mathbf{X}_1)^{-1} \mathbf{X}_1^{\mathrm{T}} \mathbf{y} & \text{für } i = 1 \\ (\mathbf{X}_i^{\mathrm{T}} \mathbf{X}_i)^{-1} \mathbf{X}_i^{\mathrm{T}} \tilde{\mathbf{y}} & \text{für } 1 < i \leq p. \end{cases} \tag{9}$$

3 Fallstudie

Die entwickelte Methode wird anhand verschiedener Testfunktionen untersucht und mit einem global optimierten Latin Hypercube Design verglichen, das in einem Schritt erstellt wird. Dabei wird für das resultierende Gesamtmodell in beiden Fällen dieselbe polynomiale Modellstruktur zugrunde gelegt.

3.1 Testfunktionen

Zur Untersuchung der Methodik werden verschiedene Testfunktionen verwendet, die für $\{(u_1, u_2, \ldots, u_p) \in \mathbb{R}^p \mid 0 \leq u_j \leq 1 \; \forall \; j \in \{1, 2, \ldots, p\}\}$ betrachtet werden. Die Funktionen werden zur Modellerstellung auf Basis des jeweiligen Versuchsplans ausgewertet und zur Generierung der Generalisierungsdatenpunkte genutzt.

Es wird eine Testfunktionen verwendet, die eine starke Krümmung in der Mitte des Eingangsraums aufweisen. Bild 6a zeigt die in der Mitte des Raums zentrierte, dreidimensionale Gaussfunktion

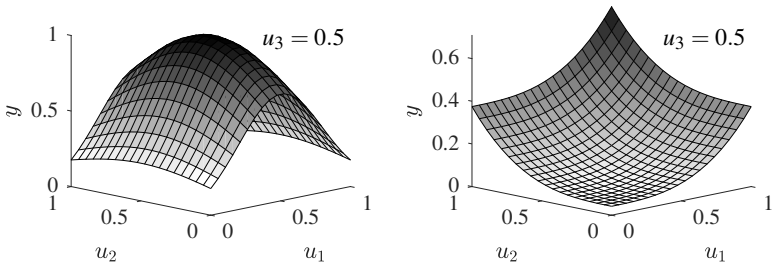
$$y(k) = \exp\Big(-\frac{1}{2}\Big(\frac{(u_1(k)-0.5)^2}{0.3^2} + \frac{(u_2(k)-0.5)^2}{0.6^2} + \frac{(u_3(k)-0.5)^2}{1.2^2}\Big)\Big), \tag{10}$$

bei der die Standardabweichung mit dem Eingangsindex j zunimmt. Dabei ist $k \in \mathbb{N}$ der Index des jeweiligen Datenpunkts.

Des Weiteren werden zwei Funktionen betrachtet, deren Krümmung zum Rand des Eingangsraums hin zunimmt. Dazu zählt die gewichtete Potenzfunktion

$$y(k) = \sum_{j=1}^{p} \frac{1}{j} u_j^3(k)), \tag{11}$$

die in Bild 6b dargestellt ist. Diese Funktion gewichtet Eingangsgrößen mit zunehmendem Eingangsindex geringer. Einen ähnlichen Verlauf, der in der



(a) Gaussfunktion (b) Gewichtete Potenzfunktion

Bild 6: Darstellung der betrachteten Testfunktionen mit $p = 3$ Eingangsgrößen.

Mitte des Eingangsraums vergleichsweise flach ist und zum Rand hin stärker gekrümmt ist, weist die Testfunktion

$$y(k) = \frac{1}{1 + 10 \cdot \sum_{j=1}^{p} \frac{1}{p}(1 - u_j(k))} + \frac{29}{44} \exp\Big(-\frac{1}{2} \sum_{i=1}^{p} \Big(\frac{u_i(k)}{0.25} \Big)^2 \Big) \tag{12}$$

gemäß [6] auf. Diese ergibt sich aus der Überlagerung einer Hyperbel- mit einer Gaussfunktion. Zusätzlich treten Wechselwirkungen zwischen den Eingangsgrößen auf.

3.2 Generalisierungsdaten

Zur Bewertung der Modellstruktur wird ein Generalisierungsdatensatz benötigt, anhand dessen die Vorhersage des Modellausgangs für unbekannte Datenpunkte untersucht wird. Der nachfolgend beschriebene Generalisierungsdatensatz wurde auf die Fallstudie zugeschnitten und verfügt über eine für reale Anwendungen nicht praktikable Anzahl an Datenpunkten. Er wird dazu verwendet, das Vorgehen zu untersuchen und die entworfene Modellstruktur zu verifizieren. An einer realen Produktionsanlage könnten zur Generalisierung beispielsweise die Datenpunkte des Betriebsbereichs verwendet werden.

Generalisierungsdaten sind so zu verteilen, dass sie auf den Verwendungszweck des Modells angepasst sind. Soll ein Modell beispielsweise im vollständigen Eingangsraum verwendet werden, müssen die Generalisierungsdaten diesen auch vollständig abdecken. Die vorgestellte Methodik hat hingegen zum Ziel, eine möglichst hohe Modellgüte in Bereichen um den vorgegebenen Arbeitspunkt zu erzielen. Die Anlage wurde durch ihre Konstruktion so ausgelegt, dass in diesen Bereichen günstige Prozessergebnisse erzielt werden. Ein angestrebtes Optimum wird typischerweise in diesem Bereich liegen, weshalb die Verteilung der Generalisierungsdaten auf diesen Bereich abgestimmt wird.

Es wird eine Punktverteilung verwendet, die sich über einen Parameter von einer näherungsweisen Gleichverteilung über den gesamten Eingangsraum hin zu einer konzentrierten Punktverteilung um den Arbeitspunkt variieren lässt. Da der Einfluss der Eingangsgrößen mit zunehmendem Eingangsindex stetig abnimmt, wird die Streuung der Generalisierungsdaten mit steigendem Eingangsindex stetig verringert. Auf diese Weise kann die Modellgüte in Abhängigkeit der Punktverteilung untersucht werden.

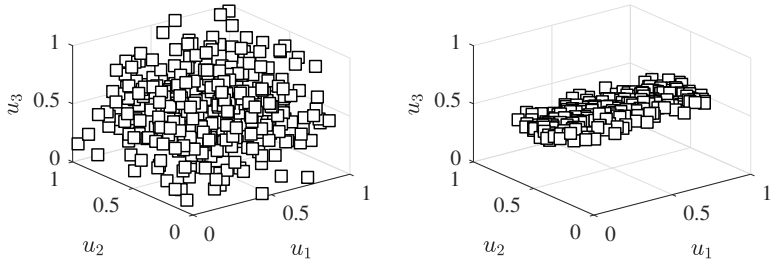
Zur Erzeugung einer solchen Punktverteilung wird jede Koordinate eines zu erzeugenden Datenpunkts $P = (p_1, p_2, \ldots, p_i)$ einer separaten Normalverteilung entnommen, sodass gilt:

$$p_j \sim \mathcal{N}(\mu_j, \sigma_j^2) \text{ mit } j \in \{1, 2, \ldots, i\}. \tag{13}$$

Um eine Extrapolation zu vermeiden, werden nur Punkte in den Generalisierungsdatensatz

$$\mathcal{G} = \{(p_1, \ldots, p_j, \ldots, p_i) \in \mathbb{R}^i | 0 \leq p_j \leq 1 \; \forall \; j \in \{1, 2, \ldots, i\}\} \tag{14}$$

aufgenommen, bei denen alle Koordinaten im vorgegebenen Wertebereich $[0, 1]$ liegen. Die jeweiligen Normalverteilungen sind mit ihrem Mittelwert $\mu_j = \bar{u}_j$ um die Koordinate des Arbeitspunkts zentriert. Um den abnehmenden Einfluss der Eingangsgrößen mit steigendem Eingangsindex zu berücksichtigen, wird



(a) Generalisierungsdaten für $\alpha = 1$ (b) Generalisierungsdaten für $\alpha = 0.2$

Bild 7: Darstellung verschiedener Punktverteilungen zur Generalisierung für den Eingangsindex $i = 3$ mit $\mu_j = \bar{u}_j$ und der initialen Standardabweichung $\sigma_1 = \frac{1}{2}$.

die Standardabweichung

$$\sigma_j = \begin{cases} \sigma_1 & \text{für } j = 1 \\ \sigma_{j-1} \cdot \alpha & \text{für } j > 1, \end{cases} \tag{15}$$

ausgehend von einer initialen Standardabweichung σ_1, verringert. Der dazu eingeführte Faktor $\alpha \in (0, 1]$ bestimmt, wie sehr die Punkte um den Arbeitspunkt und die Haupteinflüsse konzentriert sind. So kann die Punktverteilung mit abnehmendem α von einer annähernden Gleichverteilung der Punkte im gesamten Eingangsraum (Bild 7a) hin zu einem schmalen Schlauch um den Haupteinfluss (Bild 7b) variiert werden.

3.3 Durchführung

Die Methodik wird anhand einer Simulationsfallstudie untersucht. Dabei wird das inkrementelle Vorgehen zur Versuchsplanung und Modellbildung mit einem theoretisch optimalen Vorgehen in einem Schritt verglichen. Die für die Fallstudie gewählten festen Parameter sind in Tabelle 1 zusammengefasst. Weitere Parameter, wie der gewählte Arbeitspunkt $\bar{\mathbf{u}}$, der Parameter α für die Verteilung der Generalisierungsdaten und das Signal-Rausch-Verhältnis *SNR* werden während der Fallstudie variiert.

Tabelle 1: Übersicht über die gewählten Parameter der Fallstudie

konstante Parameter der Fallstudie	
Versuchspunkte des LHC in jedem Schritt	$N = 21$
Durchgeführte Modellierungsschritte	$p^* = 3$
Initiale Standardabweichung zur Generalisierung	$\sigma_1 = \frac{1}{2}$
Anzahl der Generalisierungspunkte	$N_{\text{test}} = 1000$
Wiederholungen pro Parameterkombination	$W = 40$
Grad der Polynommodelle	$m = 2$

Als Vergleichsmodell wird ein Polynommodell des Grades $m = 2$ gemäß (3) verwendet, das dieselbe Struktur aufweist, wie das aus der Addition der Teilmodelle $\sum_{j=1}^{p^*} \hat{y}_j$ resultierende Polynommodell. Damit treten in beiden Modellen dieselben Terme mit insgesamt derselben Anzahl zu bestimmender Parameter auf. Unterschiede in der Modellgüte sind somit nur auf die verwendeten Versuchspläne zurückzuführen. Zur Schätzung des Vergleichsmodells wird ein LHC der Dimension p^* betrachtet, der über N_{p^*} Punkte gemäß (2) verfügt. Dieser LHC wurde ebenfalls mit dem EDLS-Algorithmus [2] optimiert.

Zur Beurteilung der erzielten Modellgüte wird der Root-Mean-Square-Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} (\hat{y} - y)^2} \tag{16}$$

verwendet, wobei N_{test} der Anzahl an Datenpunkten im Generalisierungsdatensatz entspricht.

3.4 Ergebnisse

Als erste Testfunktion wird die Gaussfunktion (10) betrachtet. Für diese Funktion stellt Bild 8 den mittleren RMSE über $W = 40$ Wiederholungen des inkrementellen Modells und des Vergleichsmodells dar. Dabei wurde die Verteilung der Generalisierungsdaten über den Parameter $\alpha \in [0.2, 1]$ variiert. Für

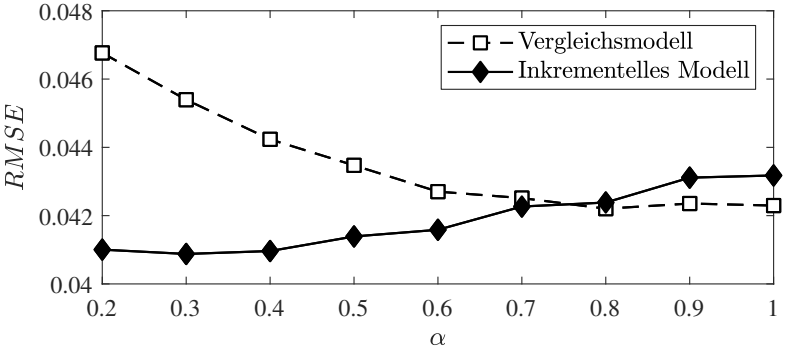


Bild 8: Mittlerer RMSE des inkrementellen Modells und des Vergleichsmodells für (10) über verschiedene Verteilungen der Generalisierungsdaten α bei $SNR = 60$ dB und $\bar{\mathbf{u}} = (0.4, 0.4, 0.4)$.

eine hohe Punktdichte der Generalisierungsdaten um den Arbeitspunkt $\alpha \ll 1$ ergibt sich durch das inkrementelle Vorgehen gegenüber dem Vergleichsmodell ein geringerer Modellfehler. Decken die Generalisierungdaten mit $\alpha = 1$ näherungsweise den kompletten Eingangsgsraum ab, erzielt hingegen das Vergleichsmodell eine höhere Modellgüte. Dieses Verhalten bestätigt die generelle Funktionsweise der inkrementellen Methodik. Das Vergleichsmodell ist aufgrund eines global optimierten Versuchsplans auf den gesamten Eingangsraum ausgelegt und erreicht für diesen Raum die höchste Modellgüte. Das inkrementelle Vorgehen konzentriert hingegen die Versuchspunkte auf bestimmte Bereiche, um Vorteile in der praktischen Anwendung zu erzielen. Trotzdem wird für den gesamten Eingangsraum ein Fehler erzielt, der nicht gravierend von dem Fehler des Vergleichsmodells abweicht. Außerdem wird ersichtlich, dass mit dem inkrementellen Vorgehen unter bestimmten Bedingungen ein Vorteil gegenüber dem globalen optimalen Versuchsplan erzielt werden kann.

Zudem hat die Wahl des Arbeitspunktes einen Einfluss auf die erzielbare Modellgüte des inkrementellen Vorgehens. Bild 9 zeigt Simulationsergebnisse für die Gaussfunktion (10), die gewichtete Potenzfunktion (11) und die Überlagerung einer Hyperbel mit einer Gaussfunktion (12) in verschiedenen Arbeitspunkten $\bar{\mathbf{u}}$. In jedem Feld ist der prozentuale Anteil der Simulationen dargestellt, in denen das inkrementelle Modell einen geringeren Gesamtfehler

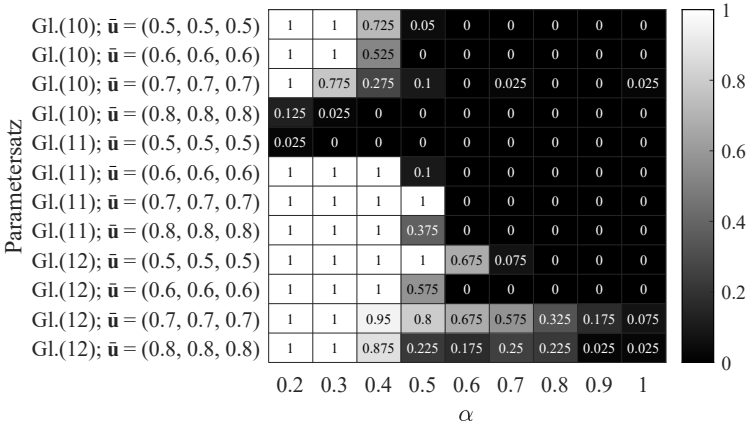


Bild 9: Anteil der Simulationen, in denen das inkrementelle Vorgehen einen geringeren Gesamtfehler als das Vergleichsmodell erzielt. Die Gaussfunktion (10), die gewichtete Potenzfunktion (11) und die Überlagerung einer Hyperbel mit einer Gausfussfunktion (12) werden in verschiedenen Arbeitspunkten $\bar{\mathbf{u}}$ bei $SNR = 100$ dB betrachtet.

als das Vergleichsmodell erzielt. Je höher dieser Anteil ist, desto heller ist das jeweilige Feld. Vom Mittelpunkt $(0.5, 0.5, 0.5)$ des betrachteten Eingangsraums ausgehend, wurden für jede Funktion nacheinander vier Punkte der ersten Hauptdiagonale als Arbeitspunkte ausgewählt.

Je weiter der Arbeitspunkt vom Mittelpunkt entfernt liegt, desto geringer ist bei (10) der Anteil der Simulationen, in denen mit dem inkrementellen Modell ein Vorteil gegenüber dem Vergleichsmodell erzielt wird. Bei (11) zeigt sich ein gegenteiliges Verhalten, da insbesondere im Mittelpunkt das Vergleichsmodell bessere Ergebnisse liefert. Für die betrachteten Funktionen wird deutlich, dass mit dem inkrementellen Vorgehen ein Vorteil erzielt werden kann, wenn der Arbeitspunkt in einem Bereich mit hoher Krümmung liegt. Der Funktionsverlauf (12) weist ähnlich wie (11) eine zunehmende Krümmung mit steigendem Abstand zum Mittelpunkt auf, die bei dieser Funktion jedoch deutlicher ausgeprägt ist. Durch (12) wird ersichtlich, dass das inkrementelle Vorgehen auch für hohe α-Werte durch die Wahl geeigneter Arbeitspunkte eine ähnliche Modellgüte wie das Vergleichsmodell erzielen kann.

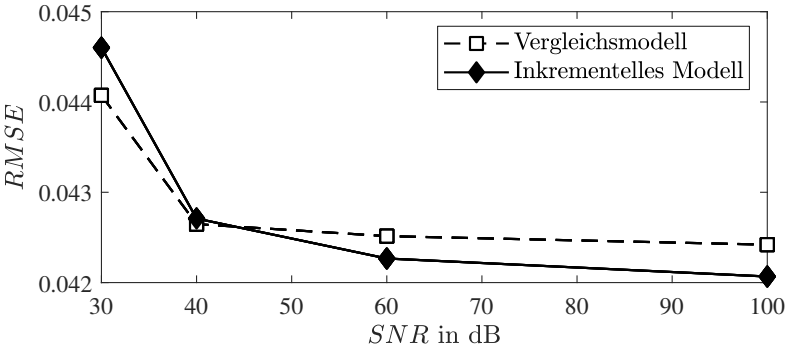


Bild 10: Mittlerer RMSE des inkrementellen Modells und des Vergleichsmodells für (10) über verschiedene Signal-Rausch-Verhältnisse *SNR* bei $\alpha = 0.7$ und $\bar{\mathbf{u}} = (0.4, 0.4, 0.4)$.

Bild 10 stellt den Gesamtfehler für die Gaussfunktion (10) mit unterschiedlichen Signal-Rausch-Verhältnissen dar. Dabei wurden ein konstanter Arbeitspunkt und eine konstante Verteilung der Generalisierungsdaten gewählt. Bei einem niedrigen *SNR*, also stark verrauschten Messdaten, weist das inkrementelle Vorgehen einen größeren Gesamtfehler auf, als das Vergleichsmodell. Dieses Verhalten lässt sich durch die insgesamt konzentriertere Punktverteilung beim inkrementellen Vorgehen erklären. Einzelne abweichende Messwerte haben dabei einen größeren Einfluss auf das Gesamtmodell und können die Modellgüte herabsetzen.

4 Zusammenfassung und Ausblick

Dieser Beitrag zeigt eine inkrementelle Vorgehensweise zur Versuchsplanung und Modellbildung, die für die praktische Anwendung an bestehenden Produktionsanlagen Vorteile aufweist. So wird insbesondere der Bereich um einen vorgegebenen Arbeitspunkt, also ein Bereich, der für den praktischen Einsatz des Modells von großer Bedeutung ist, mit einer hohen Güte abgebildet. Dabei stellt die hier entworfene Modellstruktur sicher, dass die Güte im Arbeitspunkt in jedem weiteren Schritt nicht herabgesetzt werden kann. Das Modell wird in jedem Schritt um weitere Informationen ergänzt und verbessert.

Schließlich liegt in jedem Modellierungsschritt ein Modell vor, das mit geringem
Aufwand und mit einer überschaubaren Anzahl an Datenpunkten in einem Produktivsystem erstellt werden kann. Die Eignung dieser Vorgehensweise wurde anhand einer Simulationsfallstudie überprüft und mit einem global optimalen Versuchsplan verglichen.

Als weiterer Schritt kann die Eignung andere Modellstrukturen für die Teilmodelle untersucht werden. Durch den Einsatz von flexibleren Modellen als den bisher verwendeten Polynommodellen könnten nichtlineare Zusammenhänge besser abgebildet und die Gesamtmodellgüte gesteigert werden. Eine Möglichkeit ist die Verwendung von lokalen Modellnetzen als Modellstruktur. Die Eigenschaft, dass ein neu hinzugefügtes additives Modell die Modellgüte im zuvor betrachteten Unterraum nicht herabsetzt, könnte durch eine geeignete Wahl der Aktivierungsfunktionen erreicht werden.

Förderung

Dieses Vorhaben wurde aus Mitteln des Europäischen Fonds für regionale Entwicklung (EFRE) gefördert (Förderkennzeichen 34.EFRE-0300072).

Literatur

[1] Veronica Czitrom: „One-Factor-at-a-Time versus Designed Experiments“. In: *The American Statistician* 53.2. S. 126–131. 1999.

[2] Tobias Ebert, Torsten Fischer, Julian Belz, Tim Oliver Heinz, Geritt Kampmann and Oliver Nelles: „Extended deterministic local search algorithm for maximin latin hypercube designs“. In: *2015 IEEE Symposium Series on Computational Intelligence*, S. 375–382. IEEE, 2015.

[3] Ludwig Fahrmeir, Thomas Kneib and Stefan Lang: „Regression: Modelle, Methoden und Anwendungen (Statistik und ihre Anwendungen)“. Berlin Heidelberg: Springer. 2008.

[4] David Forsyth: „Applied Machine Learning“. Cham: Springer Nature Switzerland. 2019.

[5] Isabelle Guyon and Andre Elisseeff: „An Introduction to Variable and Feature Selection“. In: *Journal of Machine Learning*, 3, S. 1157–1182. 2003.

[6] Benjamin Hartmann: „Lokale Modellnetze zur Identifikation und Versuchsplanung nichtlinearer Systeme“. Dissertation, In: *Schriftenreihe der Arbeitsgruppe Mess- und Regelungstechnik – Mechatronik, Department Maschinenbau, Universität Siegen*, (Nelles, O., Hg.), 2014.

[7] Karl Siebertz, David van Bebber and Thomas Hochkirchen: „Statistische Versuchsplanung. Design of Experiments (DoE)“. Berlin: Springer Vieweg. 2017.

[8] Thomas J. Santner, Brian J. Williams and William I. Notz: „The Design and Analysis of Computer Experiments“. New York: Springer. 2018.

[9] Felipe A. C. Viana: „Things you wanted to know about the Latin hypercube design and were afraid to ask“. In: *10th World Congress on Structural and Multidisciplinary Optimization, Orlando, Florida, USA*, S. 1–9. 2013.

Influence of Synthetic Label Image Object Properties on GAN Supported Segmentation Pipelines

Moritz Böhland, Tim Scherr, Andreas Bartschat, Ralf Mikut, Markus Reischl

Institute for Automation and Applied Informatics,
Karlsruhe Institute of Technology, Karlsruhe, Germany
E-Mail: moritz.boehland@kit.edu

1 Introduction

Artificial neural networks (ANNs) are a state-of-the-art method for image segmentation [1, 2, 3]. To train a neural network for a specific segmentation task, a dataset containing images and corresponding label images is required [4]. Usually, label images are annotated manually or semi-automatically. This is time consuming, expensive and prone to errors. A recent approach substitutes manual labeling by a Generative Adversarial Network (GAN) [5, 6, 7, 8, 9, 10, 11, 12, 13]. GANs are able to transform data from one domain into another. For example from the domain of horse images to the domain of zebra images, while preserving the general properties of the image [14]. This approach can be used to create a synthetic training dataset based on real-world images and synthetic label images (which have to be created manually or semi-automatically). The main advantage of this approach is, that during the training of the GAN no direct one-to-one relation between image and label is required. The GAN learns the mapping from the label domain to the image domain and vice versa. During inference, the trained model can be used to create a paired synthetic dataset from the synthetic label images [12, 9, 11]. For example, this can be achieved by the CycleGAN architecture [14].

Since the synthetic label images are part of the synthetic training dataset created by the GAN, they highly influence the subsequent segmentation results. This turns the creation of the synthetic label images into a central point of the method. General guidelines on how to create synthetic label images are already given in [9]. However, no specific approach for synthetic label image generation and no quantitative results regarding the quality of the synthetic label images are provided.

This article examines the influence of synthetic cell nuclei label image properties on the segmentation quality. Therefore, the segmentation results of a GAN and an U-Net trained on the generated synthetic training dataset are compared. Furthermore, insights on how to create high quality synthetic label images for the GAN are given. Analyzing the quantitative relation between the synthetic label image properties and the resulting segmentation quality leads to the identification of the relevant object properties. Focusing on these properties reduces the time necessary for synthetic label images creation.

In Section 2 GANs in general, the CycleGAN and frameworks to create synthetic datasets are discussed. A segmentation pipeline utilizing GANs to create a synthetic dataset as well as possibilities to quantify the influence of synthetic label images on the segmentation pipeline are described in Section 3. The experimental setup and the results are shown in Section 4 and Section 5. Section 6 gives a conclusion and an outlook.

2 State-of-the-Art

2.1 Generative Adversarial Networks

A new framework for neural networks called Generative Adversarial Networks was introduced in 2014 by Goodfellow [15]. In the introduced framework, two models are simultaneously trained. The generator model tries to generate samples that matches a given distribution. The discriminator model tries to distinguish between a sample coming from the generator and a sample coming

from the given distribution. They compete against each other and therefore the training goal with the value function V can be described as

$$\begin{aligned}\min_{\theta_G}\max_{\theta_D} V =& \mathbb{E}_{X\sim p_{data}}[\log D(x,\theta_D)] \\ &+\mathbb{E}_{Z\sim p_z}[\log(1-D(G(z,\theta_G),\theta_D))],\end{aligned} \tag{1}$$

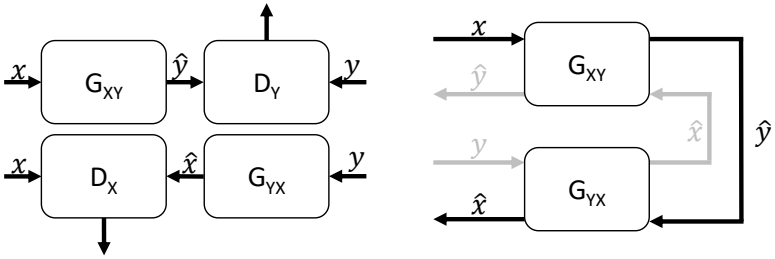
where z is a sample from a noise distribution p_z given to the generator $G(z,\theta_G)$ (with the network parameters θ_G) to generate samples. Respectively x is a sample from the real data distribution p_{data} and $D(x,\theta_D)$ is the discriminator. The output of $D(x,\theta_D)$ is the estimated probability of a sample being from p_{data}. The expected value is denoted by $\mathbb{E}$. When the discriminator is able to distinguish between a sample from the given distribution and the generated distribution, $D(x,\theta_D)$ and $1-D(G(z,\theta_G),\theta_D)$ will be close to 1 and therefore Equation 1 is maximized. When the generator is able to produce samples close to the given distribution and the discriminator is not able to distinguish between both samples, $D(x,\theta_D)$ will be close to 0 and $1-D(G(z,\theta_G),\theta_D)$ will also be close to 0. This leads to the training goals of minimizing Equation 1 for the generator and maximizing it for the discriminator [16].

With the trained generator, additional samples of p_{data} can be generated. This is also possible for highly complex distributions like the distribution of images containing a dog [17, 18]. A drawback is the limited influence on the generated samples, since the generator is only dependent on the random input z.

2.2 CycleGAN

Unlike the framework described in Subsection 2.1, CycleGAN does not create samples from a random input, but is able to perform image-to-image translation which is a generalization of style transfer [16]. In image-to-image translation, the mapping from a source domain of images (a highly complex distribution) to a target domain of images is performed. For example images of horses are transformed to images of zebras, while preserving the overall properties such as number of animals, pose and background. Furthermore, CycleGAN is able to learn the translation in an unpaired manner, therefore no paired training data

is needed (images from the source domain have no corresponding image in the target domain while training) [14].



(a) Generator - discriminator data flow (b) Cycle-consistency

Figure 1: CycleGAN with two generators (G) and discriminators (D) to transform images $x \in X$ to $\hat{y} \in Y$ and vice versa (a). The generators transform the images to the target domain, while the discriminators try to distinguish between generated images and real images from the target domain. To ensure cycle-consistency (generated images match original images), images $\hat{x}$ and $\hat{y}$ are created by passing the corresponding generators (b).

CycleGAN is able to perform unpaired image-to-image translation by extending the original GAN framework. It contains two generators and two discriminators (see Figure 1a). Generator G_{XY} is used to transform images $x \in X$ from the distribution $\mathscr{X} \sim p_x$ to an image $\hat{y}$ of the distribution $\mathscr{Y} \sim p_y$. The distribution $\mathscr{X} \sim p_x$ is approximated by images of the domain X and the distribution $\mathscr{Y} \sim p_y$ is approximated by the images of the domain Y (with images $y \in Y$). The discriminator D_Y is trained to distinguish between images y and images $\hat{y}$ created by G_{XY}. Generator G_{YX} is used to transform images y to the distribution $\mathscr{X} \sim p_x$. The discriminator D_X works accordingly to D_Y. The loss functions can be described equivalent to Equation 1. Therefore, the loss $\mathscr{L}_{GAN}(G_{XY}, D_Y, X, Y)$ for mapping $\mathscr{X} \sim p_x$ to $\mathscr{Y} \sim p_y$ can be defined by

$$\begin{aligned}\mathscr{L}_{GAN}(G_{XY}, D_Y, X, Y) = &\mathbb{E}_{\mathscr{Y} \sim p_y}[\log D_Y(y, \theta_Y)] \\ &+ \mathbb{E}_{\mathscr{X} \sim p_x}[\log(1 - D_Y(G_{XY}(x, \theta_{XY}), \theta_Y))].\end{aligned} \tag{2}$$

The loss for mapping $\mathscr{Y} \sim p_y$ to $\mathscr{X} \sim p_x$ is defined accordingly by $\mathscr{L}_{GAN}(G_{YX}, D_X, Y, X)$.

To further restrain the mapping and force the CycleGAN to preserve important properties of the input image, cycle-consistency is introduced. Otherwise, the input image is only considered as a random input variable by the CycleGAN. As seen in Figure 1b an image x is transformed to $\mathscr{Y} \sim p_y$ by G_{XY} creating image $\hat{y}$. Afterwards the image is transformed back to $\mathscr{X} \sim p_x$ by G_{YX} creating image $\hat{x}$. Ideally x and $\hat{x}$ should be the same. The same is done for the images from images y. Therefore the cycle-consistency loss

$$\begin{aligned} \mathscr{L}_{cyc}(G_{XY}, G_{YX}) = &\mathbb{E}_{\mathscr{X} \sim p_x}[\|G_{YX}(G_{XY}(x)) - x\|] \\ &+ \mathbb{E}_{\mathscr{Y} \sim p_y}[\|G_{XY}(G_{YX}(y)) - y\|] \end{aligned} \tag{3}$$

can be defined. With Equation 2 and Equation 3, the combined loss

$$\begin{aligned} \min_{\theta_{XY}, \theta_{YX}} \max_{\theta_X, \theta_Y} \mathscr{L}(G_{XY}, G_{YX}, D_X, D_Y) = &\mathscr{L}_{GAN}(G_{XY}, D_Y, X, Y) \\ &+ \mathscr{L}_{GAN}(G_{YX}, D_X, Y, X) \\ &+ \lambda \mathscr{L}_{cyc}(G_{XY}, G_{YX}) \end{aligned} \tag{4}$$

can be defined. Again, the discriminators try to maximize Equation 4 (by tuning their parameters θ_X and θ_Y), while the generators try to minimize it (by tuning their parameters θ_{XY} and θ_{YX}). The hyperparameter λ balances the importance of both losses. Further details can be found in [14]. During inference, the generators are able to create paired samples by creating the corresponding images from the target domain. Variations of CycleGAN and GANs able to perform unpaired image-to-image translation can be found in [9, 19, 20, 12, 21].

2.3 Synthetic Dataset Creation

Frameworks to create synthetic datasets are used for training and benchmarking [22, 23, 24, 25]. Datasets containing 3D+t cell images can be created by placing spheres in the images, distorting them to match the variability of the real cells and applying texture. Afterwards, the microscopy image acquisition is simulated by applying a point spread function and different noise sources. This can be done by Cytopacq which is freely available and has an online

benchmark dataset generator available[1] [26, 27]. A different approach does not create the cells by placing spheres, but by extracting a set of cell nuclei form existing images. The cells are placed in a new dataset using cell positions of repaired tracks computed from realistic images [28]. In general, methods simulating the microscopy acquisition have many parameters which need to be tuned to match a specific real-world microscopy image dataset.

Recent approaches try to substitute the simulation of the microscopy acquisition by using different types of GANs [5, 6, 7, 8, 9, 10, 11, 12, 13]. High quality results are achieved for biological data, while simplifying the process drastically. Datasets are created by training the GAN using existing label images. However, there are still open questions: how are the synthetic label images created, which label image properties influence the results as well as how and why they were chosen. Furthermore, the segmentation quality (when evaluating real-world datasets by training with the synthetic dataset) is often compared to different GANs, but not to segmentation networks trained with manually annotated label images.

3 Methods

3.1 Segmentation Pipeline

Segmentation pipelines based on ANNs like the U-Net [3] need paired training samples which is e.g. manually annotated. By the use of CycleGAN, paired training samples can be created without the need for manual annotations. A segmentation pipeline utilizing a CycleGAN for the creation of the training data is shown in Figure 2. However, any GAN able to perform unpaired image-to-image translation can be used.

In the first step of the segmentation pipeline, synthetic label images X need to be created. This can be done by visual inspection of the real-world microscopy images Y or by simulation of the underlying physical model: If the synthetic label images are created by visual inspection, properties describing the labels

[1] Benchmark dataset generator available at `https://cbia.fi.muni.cz/simulator`.

(e.g. the size of objects or the number of objects per image) need to be visually extracted from the microscopy images. These properties can then be used to create synthetic label images. On the other hand, synthetic label images can be created by simulation of a corresponding physical model. For example label images for the segmentation of porous membranes, can be acquired by the simulation of the phase separation of polymer solutions [29].

After creation of the synthetic label images, the CycleGAN needs to be trained with X and Y. The resulting network parameters are used for inference. Inference can be done in two different ways:

1. The CycleGAN is used to directly create the label images $\hat{X}$ from Y.
2. The CycleGAN is used to generate the corresponding synthetic microscopy images $\hat{Y}$ to X. The resulting paired synthetic label and microscopy
   images (X and $\hat{Y}$) are used as a training dataset for a segmentation network (e.g. a U-Net). After training on the synthetic training dataset, the segmentation network is used in inference mode to create the corresponding label images $\hat{X}$ to Y.

3.2 Quantification of the Synthetic Label Image Object Properties Influence

A crucial step in the method described in Subsection 3.1 is the creation of the synthetic label images. If they do not match the properties of the label images of the real microscopy images, CycleGAN learns a wrong mapping and therefore the results are biased. Essential properties for cells are e.g. size or shape. They have to be defined individually for different types of data. If the size of an object is too small in the label image, CycleGAN could learn to increase it. This would result in a synthetic dataset with big cells in the synthetic microscopy images and small cells in the label images. To gain insight into which properties need to be modelled accurately and which properties have no influence on the final segmentation results, the following procedure is introduced:

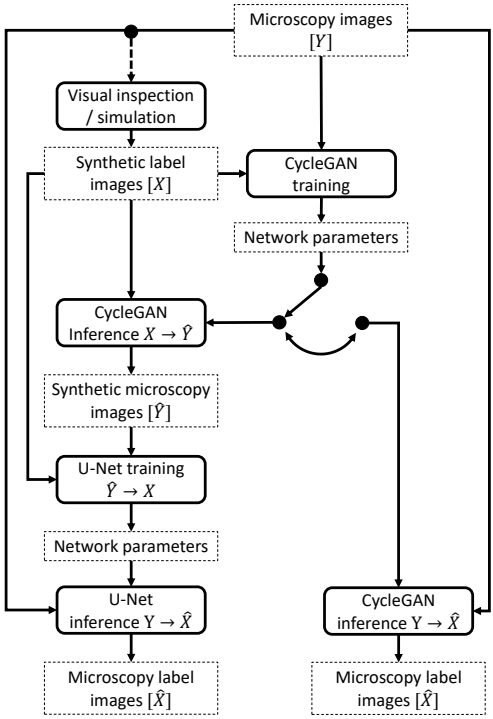


Figure 2: Segmentation pipeline utilizing a CycleGAN. The CycleGAN can directly be used to generate the desired microscopy label images. Furthermore, it can be used to generate a synthetic training dataset for a segmentation network.

1. Extract the real properties from the manually annotated microscopy images (ground truth),
2. create different synthetic label image datasets in which one property varies, while the other properties are fixed,
3. apply the segmentation pipeline from Subsection 3.1 to each of the datasets,
4. evaluate the segmentation results for the different datasets,
5. repeat steps 2-4 for each property.

Although optimization of the properties is a multivariate optimization problem, the proposed univariate optimization can be used to discuss effects. The univariate optimization is done to reduce the computational expense. Property extraction and reducing the amount of relevant properties can be challenging for complex scenarios (e.g. simulation of street scenes). In this study, a relatively easy dataset is used (see Subsection 4.1) to reduce the risk of errors in the property extraction step and have a straightforward insight into the resulting influence of the object properties. However, the pipeline can also be applied to more complex datasets.

4 Experimental Setup

4.1 Dataset

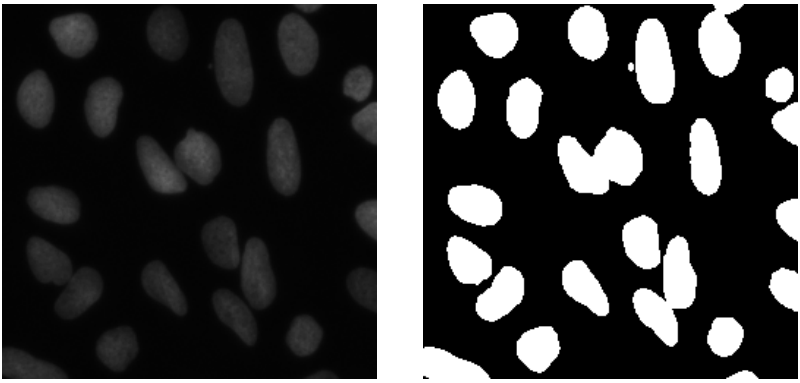
The influence of synthetic label image properties has been evaluated by applying the segmentation pipeline from Subsection 3.1. The BBBC039v1[2] dataset has been used for the experiments [23].

The dataset contains 200 images of U2OS cells acquired by fluorescence microscopy. The images are stored as 520×696 px 16-bit grayscale images. The cell nuclei in the corresponding label images are manually annotated. An exemplary crop with 256×256 px and the corresponding label image are shown in Figure 3. The different sizes and shapes of the cells are visible. It is notable, that ellipses are approximating the cells well, even though the borders of the real cells are distorted.

4.2 Synthetic Label Image Generation

To create the synthetic label images, label image properties have been extracted from the ground truth containing all 200 images. For the used cell dataset, the properties are cells per image (P_C), size of the cells (P_S) (diameter of a circle with the same area) and the eccentricity of ellipse that have the same

[2]The dataset is available at `https://data.broadinstitute.org/bbbc/BBBC039/`.



(a) Microscopy image (b) Corresponding ground truth label image

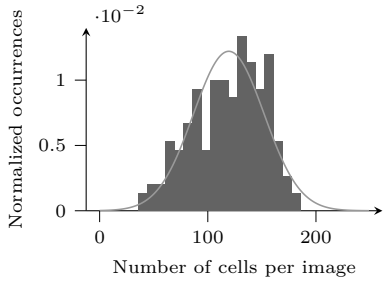
Figure 3: Exemplary 256×256 px crop from a BBBC039v1 cell microscopy image (a) and the corresponding label image (b).

second-moment as the cells (P_E). For each property, the mean and the standard deviation are calculated. To handle outliers, the distributions have been clipped between the 5% and 95% percentiles. Afterwards, each distribution is approximated by a normal distribution. Histograms of the properties and the approximations are shown in Figure 4.

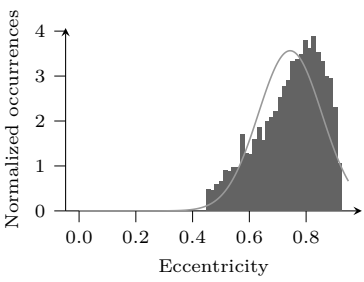
The resulting means and standard deviations of the fitted normal distributions are 120 ± 33 for P_C, 27 ± 6 for P_S and 0.74 ± 0.11 for P_E. Especially for P_S, using a normal distribution does not result in a perfect fit to the data. Nevertheless, it can be used to evaluate the influence of the different properties on the segmentation performance.

A synthetic dataset is created by drawing from the normal distributions. It has to be assured, that no invalid values (e.g. cell size < 0) are drawn. An example with different means for the cell eccentricity is shown in Figure 5.

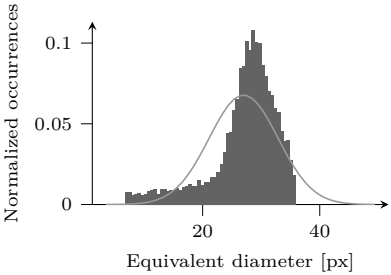
For each image in a dataset the number of cells per image and for each cell in the images the size and the eccentricity are individually drawn from the corresponding distributions. As introduced in Subsection 3.2, different datasets are created by varying the mean of the normal distributions while keeping the standard deviation constant (being the ground truth value). For each property, 8 datasets with the following means:



(a) Cells per image distribution



(b) Cell eccentricity distribution



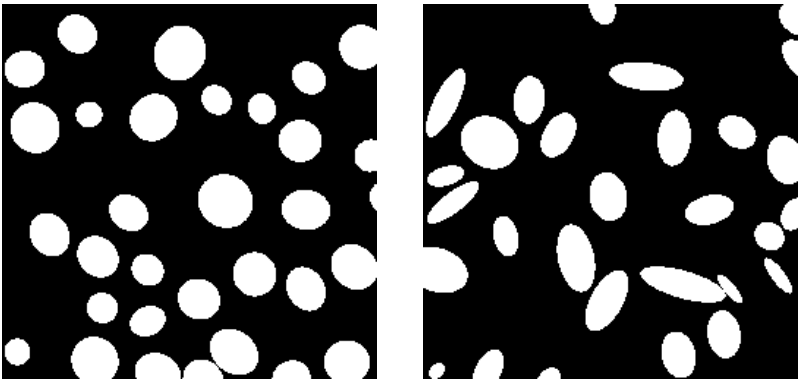
(c) Equivalent cell diameter distribution

Figure 4: Distributions of different image parameters extracted from the ground truth images. The distributions are clipped between the 5% and 95% percentiles and normalized to an area of 1. A normal distribution is fitted.

- $\mu_{P_C} = 40, 60, 80, ..., 180,$
- $\mu_{P_S} = 7, 12, 17, ..., 42$ and
- $\mu_{P_E} = 0.34, 0.44, 0.54, 0.64, 0.69, 0.74, 0.79, 0.84,$

have been created. The mean values were selected to have datasets containing both smaller and larger values than the true mean values.

Each of the datasets contains 560 label images with 520×696 px, which is equivalent to the BBBC039v1 dataset. The dataset is split as follows: 160 images are used for CycleGAN training (alongside 160 of the BBBC039v1 images). The remaining 400 label images are used for CycleGAN inference creating the training dataset for the U-Net.



(a) Ellipses with eccentricity of 0.44 ± 0.11 (b) Ellipses with eccentricity of 0.79 ± 0.11

Figure 5: Comparison of two 256×256 px areas extracted from synthetic datasets with different mean eccentricity. The left image (a) has a mean eccentricity of 0.44 and therefore the ellipses are more round than the ellipses on the right image (b) with a mean of 0.79. Both have a standard deviation of 0.11.

4.3 Neural Network Configurations, Training and Post-Processing

The PyTorch implementation[3] of [14] is used for the CycleGAN. The generators consist of a downsampling block, 9 ResNet blocks and an upsampling block. The discriminators each consist of a 70×70 px PatchGAN [30]. The CycleGAN is trained with random crops (256×256 px) for 200 epochs. For better results, each image of the BBBC039v1 dataset has been scaled to the minimum and maximum intensity in the dataset. The inference is done on the full image, resulting in output images of 520×696 px.

The applied segmentation pipeline consists of a modified U-Net with batch normalization and transposed convolutions. In the output layer the sigmoid activation function is used. The loss function is a weighted sum of Dice loss [31] and binary crossentropy loss. Training is terminated after 100 epochs or

[3]The CycleGAN PyTorch implementation is available at `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`.

when the loss on the validation data has not improved for 10 epochs. Post-processing is done by binarizing the network output with a threshold, applying the Euclidean distance transform and thresholding the Euclidean distance transform to create seeds. The seeds and the binarized network output are used to create an instance segmentation (touching cells can be distinguished) with a watershed algorithm. The dataset created by the CycleGAN is used (synthetic binary label images and synthetic microscopy images) for training. The 400 images are split in 80% training (320 images) and 20% validation (80 images).

Furthermore, a reference segmentation is acquired by training the U-Net with 60 of the images used to train the GAN and the manually annotated label images. This represents the best performance achievable by manual labeling with the applied U-Net.

The 40 images from the BBBC039v1 dataset (and the corresponding manually annotated label images) not used during CycleGAN training (respectively training of the reference segmentation) are used for evaluation. When using the CycleGAN for processing the 40 evaluation images instead of the U-Net (see Subsection 3.1), the same post-processing as for the U-Net is applied.

5 Results

The performance of the segmentation is measured by the mean average precision at different intersection over union thresholds (P_{IoU}) and the F-Score (true positive: predicted cell matches ground truth cell with $\mathrm{IoU} > 0.5$) [32, 33]. The results for P_C, P_S and P_E are shown in Figure 6. For each property the performance of the segmentation with the CycleGAN, the CycleGAN and U-Net (CG+U-Net) and the reference segmentation are compared.

For the mean number of cells, the U-Net performs well near the mean value of the dataset (120 cells per image). It performs even better if the number of cells is slightly higher, but falls back, if the number is way higher. When compared to Figure 4c it is notable, that the maximum of the histogram of the data is also higher, than the mean of the approximated normal distribution. When μ_{P_C}

is too low, the P_{IoU} decreases drastically. This is even more visible for the F-Score. While the U-Net performs better for values in the range of the real distribution, the CycleGAN performs better for very small values of μ_{P_C}. It is notable, that the U-Net trained with the CycleGAN is able to outperform the reference for $\mu_{P_C} = (140, 160)$. Although, it has to be further evaluated whether this is significant (see Section 6).

For the mean equivalent circle diameter, the performance of the P_{IoU} drops quickly as the mean moves away from the optimum. The difference between the U-Net segmentation and the CycleGAN segmentation is especially notable in areas around the optimum. The F-Score does not drop as quickly as the P_{IoU}. This is due to the fact, that the P_{IoU} depends heavily on the correct size estimation.

Also for the mean cell eccentricity, the F-Score is less dependent on the correct choice of the mean value. Only for very small values, it drops rapidly. This could also be due to the dependence on the correct size estimation. The maximum performance is achieved for $\mu_{P_E} = 0.74$. That corresponds to the mean of the normal distribution fitted to the real data.

The mean P_{IoU} of the best performing CG+U-Net segmentation for each of the properties is 0.64 while it is 0.46 for the CycleGAN segmentation. This encourages usage, of the more complex CG+U-Net segmentation. As shown in Figure 7, the visual quality of the synthetic images is high. Even for synthetic label images with $P_S = 12 \pm 6$, viable images are created by the CycleGAN. It is obvious, that a U-Net, trained with the dataset shown in the third row in Figure 7, will fail to produce good results, since all labels are too small. This underlines the results from Figure 6c.

6 Conclusions and Outlook

In this article, the influence of object properties of synthetic label images on a segmentation pipeline using a CycleGAN instead of manually annotated label images has been examined. As an example, the BBBC039v1 dataset with 200 images of U2OS cells acquired by fluorescence microscopy has been used.

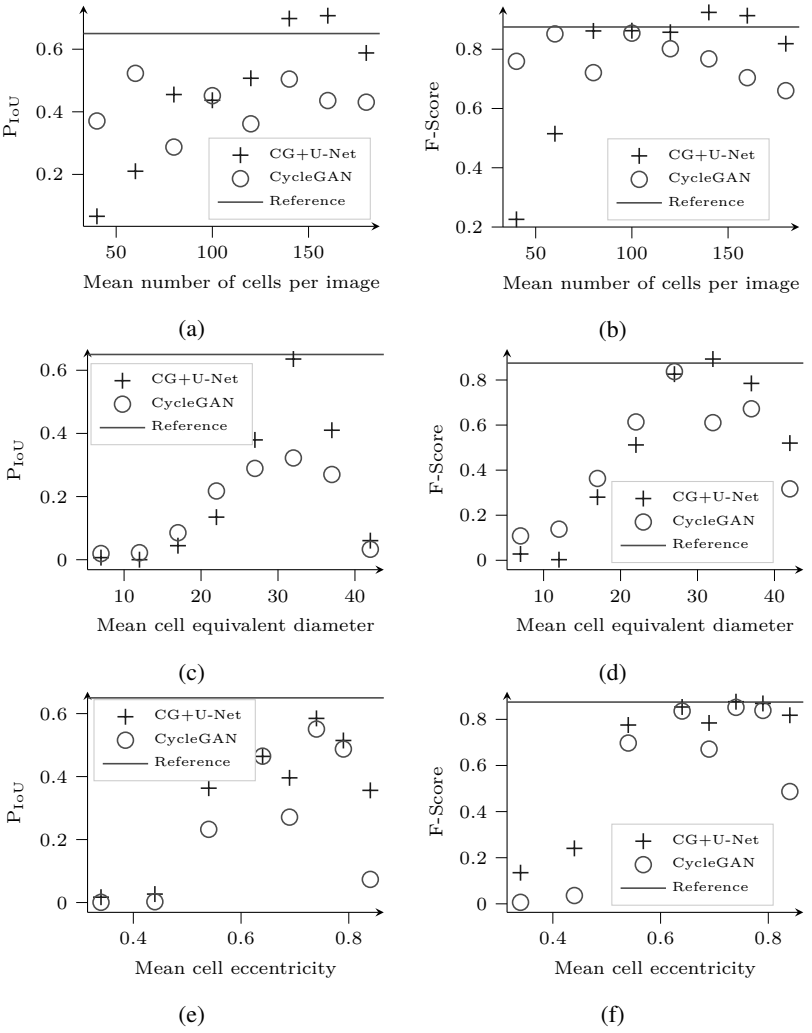


Figure 6: Resulting performance for variation of properties in the synthetic label images. Variation of the mean number of cells per image is shown in (a, b). Variation of the mean equivalent diameter of a circle is shown in (c, d) and variation of the mean cell eccentricity is shown in (e, f). The P_{IoU} is used as a performance measurement in (a, c, e) and the F-Score is used in (b, d, f).

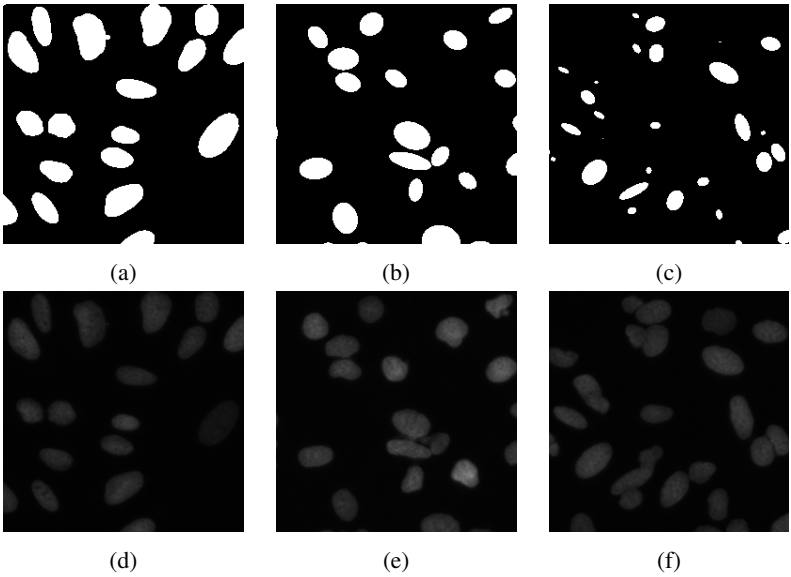


Figure 7: The left column shows a crop from a BBBC039v1 microscopy image with 256×256 px (d) and the corresponding label image (a). The middle column (b, e) shows a synthetic label image and the corresponding synthetic microscopy image from a synthetic dataset with normal distributions $P_C = 120 \pm 33$, $P_S = 27 \pm 6$ and $P_E = 0.74 \pm 0.11$. For the right column (c, f), the mean diameter P_S is changed to 12 ± 6.

The following conclusions can be drawn:

1. The quality of the segmentation results is highly dependent on the label image properties. Choosing the wrong parameters, results in unusable results.

2. A U-Net trained with a synthetic dataset generated by a GAN can perform as well as a U-Net trained with real-world manually annotated data. However, this has to be further examined regarding significance and application on other datasets.

3. The effort that has to be put into the synthetic image generation depends strongly on the result to be achieved. For counting, less effort has to be invested, than for generating knowledge about the exact shape.

4. It is worth the computational expense to train a U-Net for segmentation and not use the CycleGAN itself.

5. It is possible to achieve high quality results, without further investment into the GAN training data (e.g. apply noise on the synthetic label images). The GAN is able to reproduce the additional properties of the microscopy image (e.g. noise and illumination) through the properties of the label image itself.

The results show quantitatively, that the introduced pipeline is able to be used in scientific image analysis, without having to sacrifice performance. Therefore, the method enables scientists to reduce the time needed to evaluate datasets by reducing manual annotation. Choosing the right object size seems to be most important and time should be invested to estimate it right. However, it must be investigated individually whether it is worth to manually annotate the microscopy images or to use the GAN supported segmentation pipeline. Especially for 3D or 3D+t datasets, where manual annotation is extremely complex, the GAN supported segmentation pipeline could be useful [34, 35].

Besides the results for this real-world cell dataset, several open questions remain. Choosing the right distributions to create the synthetic dataset is challenging without having the manually annotated label images available. To solve that problem, e.g. a grid-search could be applied. This reduces the time needed by the scientist but strongly increases the computational load. The GAN might not be able to transfer label images to highly complex images (e.g. using the method for street scenes). Therefore the usability has to be explored quantitatively for different scientific areas. There are more advanced methods to train a U-Net than training with the binary label images such as training with cell borders [33]. It has to be examined whether the introduced pipeline benefits from that as much as training a U-Net with the manually annotated images.

References

[1] K. He et al. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

[2] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, 2015.

[3] 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*, pp. 234–241, Cham, 2015. Springer International Publishing.

[4] A. Gupta et al. Deep Learning in Image Cytometry: A Review. *Cytometry Part A*, 95(4): pp. 366–380, 2019.

[5] T. de Bel et al. Stain-Transforming Cycle-Consistent Generative Adversarial Networks for Improved Segmentation of Renal Histopathology. In *Proceedings of the 2nd International Conference on Medical Imaging with Deep Learning*, volume 102, pp. 151–163, 2019.

[6] A. Osokin et al. GANs for Biological Image Synthesis. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2252–2261, 2017.

[7] L. Hou et al. Robust Histopathology Image Analysis: To Label or to Synthesize? In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[8] P. Welander, S. Karlsson, and A. Eklund. Generative Adversarial Networks for Image-to-Image Translation on Multi-Contrast MR Images - A Comparison of CycleGAN and UNIT, 2018. arXiv: 1806.07777.

[9] I. J. Stephan et al. UDCT: Unsupervised Data to Content Transformation with Histogram-Matching Cycle-Consistent Generative Adversarial Networks, 2019. bioRxiv: 563734.

[10] Z. Zhang, L. Yang, and Y. Zheng. Translating and Segmenting Multimodal Medical Volumes with Cycle- and Shape-Consistency Generative Adversarial Network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[11] Y. Huo et al. Adversarial Synthesis Learning Enables Segmentation without Target Modality Ground Truth. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pp. 1217–1220, 2018.

[12] C. Fu et al. Three Dimensional Fluorescence Microscopy Image Synthesis and Segmentation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2018.

[13] S. Han et al. Nuclei Counting in Microscopy Images with Three Dimensional Generative Adversarial Networks. In *Medical Imaging 2019: Image Processing*, volume 10949, p. 105, 2019.

[14] J.-Y. Zhu et al. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[15] I. Goodfellow et al. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. 2014.

[16] H. Huang, P. S. Yu, and C. Wang. An Introduction to Image Synthesis with Generative Adversarial Nets, 2018. arXiv: 1803.04469.

[17] H. Zhang et al. Self-Attention Generative Adversarial Networks, 2018. arXiv: 1805.08318.

[18] A. Brock, J. Donahue, and K. Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis, 2018. arXiv: 1809.11096.

[19] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised Image-to-Image Translation Networks. In *Advances in Neural Information Processing Systems 30*, pp. 700–708. 2017.

[20] Z. Yi et al. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[21] S. Benaim and L. Wolf. One-Sided Unsupervised Domain Mapping. In *Advances in Neural Information Processing Systems 30*, pp. 752–762. Curran Associates, Inc., 2017.

[22] V. Ulman et al. An Objective Comparison of Cell-Tracking Algorithms. *Nature Methods*, 14: pp. 1141–1152, 2017.

[23] V. Ljosa, K. L. Sokolnicki, and A. E. Carpenter. Annotated High-Throughput Microscopy Image Sets for Validation. *Nature Methods*, 9: p. 637, 2012.

[24] S. Sankaranarayanan et al. Learning from Synthetic Data: Addressing Domain Shift for Semantic Segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[25] J. Tremblay et al. Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.

[26] D. Wiesner et al. CytoPacq: A Web-Interface for Simulating Multi-Dimensional Cell Imaging. *Bioinformatics*, 2019.

[27] D. Svoboda, M. Kozubek, and S. Stejskal. Generation of Digital Phantoms of Cell Nuclei and Simulation of Image Formation in 3D Image Cytometry. *Cytometry Part A*, 75A(6): pp. 494–509, 2009.

[28] J. Stegmaier et al. Generating Semi-Synthetic Validation Benchmarks for Embryomics. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pp. 684–688, 2016.

[29] F. Wang et al. Progress Report on Phase Separation in Polymer Solutions. *Advanced Materials*, 31(26): p.1806733, 2019.

[30] P. Isola et al. Image-to-Image Translation with Conditional Adversarial Networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[31] F. Milletari, N. Navab, and S. Ahmadi. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 565–571, 2016.

[32] Kaggle. 2018 Data Science Bowl, 2018.

[33] T. Scherr et al. Best Practices in Deep Learning-Based Segmentation of Microscopy Images. In *Proc., 28. Workshop Computational Intelligence, Dortmund, 2018*, pp. 175 – 195, 2018.

[34] K. A. Philbrick et al. RIL-Contour: A Medical Imaging Dataset Annotation Tool for and with Deep Learning. *Journal of Digital Imaging*, 32(4): pp. 571–581, 2019.

[35] A. Y. Kobitski et al. An Ensemble-Averaged, Cell Density-Based Digital Model of Zebrafish Embryo Development Derived from Light-Sheet Microscopy Data with Single-Cell Resolution. *Scientific Reports*, 5: p. 8601, 2015.

Dieser Tagungsband enthält die Beiträge des 29. 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 28. – 29.11.2019 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 978-3-7315-0979-0