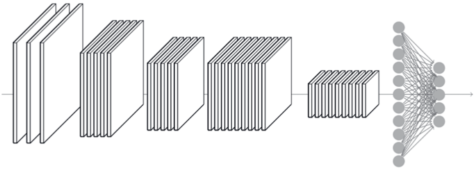
Abdullah Karatas

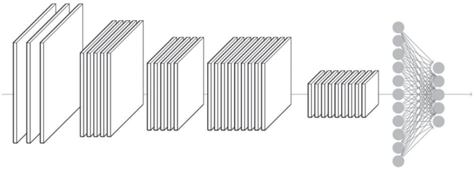


Comparison of model-based methods with machine learning strategies for defect reconstruction, classification, and regression in the field of measurement technology

Berichte aus dem Lehrstuhl für Messtechnik und Sensorik
Band 14
Herausgeber: Prof. Dr.-Ing. Jörg Seewig

TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Abdullah Karatas



Comparison of model-based methods with machine learning strategies for defect reconstruction, classification, and regression in the field of measurement technology

Berichte aus dem Lehrstuhl für Messtechnik und Sensorik
Band 14
Herausgeber: Prof. Dr.-Ing. Jörg Seewig

TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über **http://dnb.d-nb.de** abrufbar.

Berichte aus dem Lehrstuhl für Messtechnik und Sensorik - Band 14

Herausgeber: Prof. Dr.-Ing. Jörg Seewig
Lehrstuhl für Messtechnik und Sensorik
Fachbereich Maschinenbau und Verfahrenstechnik
Technische Universität Kaiserslautern
Gottlieb-Daimler-Straße
67663 Kaiserslautern

Verfasser: Abdullah Karatas
Verlag: Technische Universität Kaiserslautern

Druck: Technische Universität Kaiserslautern
Hauptabteilung 5 | Abteilung 5.6 Foto-Repro-Druck

D-386

© Abdullah Karatas · Kaiserslautern 2022

Alle Rechte vorbehalten, auch das des auszugsweisen Nachdrucks, der auszugsweisen oder vollständigen Wiedergabe (Photographie, Mikroskopie), der Speicherung in Datenverarbeitungsanlagen und das der Übersetzung.

Als Manuskript gedruckt. Printed in Germany.

ISSN 2365-9742
ISBN 978-3-95974-183-5

Short Abstract

Automation, Industry 4.0 and artificial intelligence are playing an increasingly central role for companies. Artificial intelligence in particular is currently enabling new methods to achieve a higher level of automation. However, machine learning methods are usually particularly lucrative when a lot of data can be easily collected and patterns can be learned with the help of this data. In the field of metrology, this can prove difficult depending on the area of work. Particularly for micrometer-scale measurements, measurement data often involves a lot of time, effort, patience, and money, so measurement data is not readily available. This raises the question of how meaningfully machine learning approaches can be applied to different domains of measurement tasks, especially in comparison to current solution approaches that use model-based methods. This thesis addresses this question by taking a closer look at two research areas in metrology, micro lead determination and reconstruction. Methods for micro lead determination are presented that determine texture and tool axis with high accuracy. The methods are based on signal processing, classical optimization and machine learning. In the second research area, reconstructions for cutting edges are considered in detail. The reconstruction methods here are based on the robust Gaussian filter and deep neural networks, more specifically autoencoders. All results on micro lead and reconstruction are compared and contrasted in this thesis, and the applicability of the different approaches is evaluated.

German Version

Automatisierung, Industrie 4.0 und künstliche Intelligenz spielen für Unternehmen eine immer zentralere Rolle. Insbesondere künstliche Intelligenz ermöglicht derzeit neue Methoden, um einen höheren Automatisierungsgrad zu erreichen. Allerdings sind Methoden des maschinellen Lernens in der Regel dann besonders lukrativ, wenn viele Daten einfach gesammelt werden können und mit Hilfe dieser Daten Muster gelernt werden können. Im Bereich der Messtechnik kann sich dies je nach Arbeitsgebiet als schwierig erweisen. Insbesondere bei Messungen im Mikrometermaßstab sind Messdaten oft mit viel Zeit, Mühe, Geduld und Geld verbunden, so dass Messdaten nicht ohne weiteres verfügbar sind. Dies wirft die Frage auf, wie aussagekräftig Ansätze des maschinellen Lernens auf verschiedene Bereiche von Messaufgaben angewendet werden können, insbesondere im Vergleich zu aktuellen Lösungsansätzen, die modellbasierte Methoden verwenden. Dieser Beitrag befasst sich mit dieser Frage, indem zwei Forschungsbereiche in der Messtechnik, die Bestimmung von Mikrodrall und die Rekonstruktion, näher beleuchtet werden. Es werden Methoden zur Mikrodrallbestimmung vorgestellt, die Textur und Werkzeugachse mit hoher Genauigkeit bestimmen. Die verwendeten Methoden basieren auf Signalverarbeitung, klassischer Optimierung und maschinellem Lernen. Im zweiten Forschungsbereich werden Rekonstruktionen für Schnittkanten im Detail betrachtet. Die Rekonstruktionsmethoden basieren hier auf dem robusten Gauß-Filter und tiefen neuronalen Netzen, genauer gesagt auf Autoencodern. Alle Ergebnisse zu Mikrodrall und Rekonstruktion werden in dieser Arbeit verglichen und gegenübergestellt und die Anwendbarkeit der verschiedenen Ansätze bewertet.

Comparison of model-based methods with machine learning strategies for defect reconstruction, classification, and regression in the field of measurement technology

Vom Fachbereich Maschinenbau und Verfahrenstechnik
der Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)
genehmigte
Dissertation

vorgelegt von

Abdullah Karatas
aus Frankenthal (Pfalz)

Kaiserslautern 2022
D 386

Dekan:	Prof. Dr.-Ing. Tilmann Beck
Vorsitzender:	Prof. Dr.-Ing. Jens C. Göbel
Berichterstatter:	Prof. Dr.-Ing. Jörg Seewig
	Prof. Dr.-Ing. Martin Ruskowski

Tag der mündlichen Prüfung: 29. April 2022

Dear Nicole,

I will never forget how you walked closer in my direction when we first met. From that moment on, we are inseparable. During this time we have visited many places and walked them together. You jumped out of joy at all the places we visit, and I learned from you that paradise is not a place to visit. It is the shape of your heart. I thank you for your love and your patience. I love you with all my heart. No lines can express my thanks. Without you, I would not be me and this thesis would not be what it is.

Yours Abdullah

Preface and Acknowledgment

Most of this thesis was written during my stay at the TU Kaiserslautern and at the Institute for Measurement and Sensor Technology in the Department of Mechanical and Process Engineering. At this point, I would like to thank in particular the state of Rhineland-Palatinate, which hired me as a research assistant and provided the necessary funds. Overall, I would like to thank the TU Kaiserslautern, especially the departments of Mathematics, Physics and Mechanical and Process Engineering, for the good education I received during my time at the university.

A very special thank you goes to my doctoral supervisor, Prof. Dr.-Ing. Jörg Seewig. In addition to his constant professional supervision and support, I would especially like to emphasize his human and social components. The so consolidated, natural atmosphere in the institute allows scientific freedom and the introduction of many ideas and their appropriate implementation. Without it, the development of one's own competencies to this extent would not have been possible. A big thank you also goes to my second supervisor Prof. Dr.-Ing. Martin Ruskowski, whose feedback had a great influence on this thesis. Also to Prof. Dr.-Ing. Jens C. Göbel, who chaired the doctoral thesis.

One point I would like to emphasise is the cooperation with all the students who have accompanied me over the years. Without all of you, the whole time would not have been half as meaningful and beautiful. However, the term students is not chosen correctly here, many of you have become friends.

I would like to dedicate this section to my very close friends Christian, Mark and Thomas. Your unconditional support and the time spent together contribute a lot to my own development and thus to this thesis.

An Mia: Meine Freude und deine Geschichte sind ebenfalls Teil dieser Dissertation. Ich könnte nicht glücklicher sein, dein Patenonkel zu sein.

Do moich teściów: Bardzo dziękuję za przyjęcie mnie do swoich serc i rodziny od pierwszego dnia.

Aileme: Sevginiz ve desteğiniz bana çalışma fırsatı verdi. Bu sizin için.

Ablam için: Her şeyi kaybetsek bile, her zaman birbirimize ve atlıkarıncaya sahibiz.

An Kjell, Lara, Lina und Noah: You and I have found a friend in each other.

Abdullah Mannheim 14 September 2020.

Abstract

Automation and related technologies are playing an increasingly important role in our lives. An important pillar of today's automation and research is the field of artificial intelligence. Artificial intelligence is being used very successfully for various applications and automation processes that could not be solved before. However, its use is not always profitable, as there are already many good approaches that do not follow a machine learning approach. In this thesis we want to look at different and current research fields in the field of measurement and try to solve them in two different ways. In the first approach, only model-based methods are used. In the second, machine learning approaches are used. The results obtained with both methods are compared and evaluated. Among other things, it is discussed whether machine learning methods in the field of measurement technology in the micrometre range can be usefully applied to current research areas. Also with regard to the strong dependence on training data and the high degree of automation.

Essentially, two current research areas in the field of metrology are considered. The first research area deals with micro lead and includes two core tasks, texture direction estimation and tool axis estimation. Micro lead occurs when grinding a shaft and can lead to undesirable properties of the shaft. While there is a standardised procedure for another type of lead, the so-called macro lead, which is defined in the Mercedes-Benz standard MBN 31007-7, there is currently no standardisation for micro lead. If the so-called thread method is not used to determine lead, micro lead parameters are necessary to characterise the texture direction and the workpiece axis. In this thesis a methodology for the determination of texture direction is derived, which is based

on a model approach. This methodology, developed by Prof. Dr.-Ing Jörg Seewig, is validated on simulated and real surfaces. Results with low scatter were obtained for both simulated and real surfaces. For simulated surfaces, the texture direction could be determined with an accuracy of less than one arc minute. In a second approach, methods from the field of artificial intelligence were used, more precisely a convolutional neural network. However, only simulated data was used for training, as real data for supervised learning is difficult to realise for this application. Validation is still needed here to investigate whether good results can also be achieved for real measured texture directions. In most cases, a resolution accuracy of less than one arc minute could be achieved for simulated surfaces. However, it was also observed that the neural network performs less well with untrained and thus unknown data. It is therefore recommended to use a neural network that has been trained with a wide range of arcminute texture directions. When comparing the results for texture direction determination of both methods, the model-based solution is recommended for texture direction determination for real surfaces, as with the machine learning approach the results for untrained surfaces are poor and validation for this is lacking.

The determination of the tool axis for the micro lead angle requires the measurement of a different number of measurement areas along the circumference and height of the shaft. These data points are used to fit the idealised geometry of a cylinder. No real measurement data points were used here, only simulations. The number of measurement areas was reduced from the 108 recommended by the National Institute for Standards and Technology (NIST) until only one measurement area remained in the end. This area, like all other simulated measurement areas, had a size of $0.8\mu m \times 0.8\mu m$. Fitting using methods with and without machine learning showed similarly good results for up to four simulated measurement areas. For a single measurement range, the fitting with machine learning showed a smaller deviation from the existing ground truth than the model-based method, especially for the fitting radius of the cylinder. This is due to the fact that the training data provide a kind of a priori knowledge. This can also be realised by a Bayesian estimator for the methodology without machine learning. A comparison of the two methods in 6.2 has shown that the model-based approach is clearly preferable to the machine learning approach presented, taking into account various aspects.

The last area analysed in this thesis is the reconstruction of measurement data with unwanted defects. Here, an ideal reference is to be generated, where as input a measurement with defects and imperfections is loaded. The ideal reference allows the evaluation of parameters for later evaluation of the measured component. In this thesis cutting edges were chosen as an example. By reconstructing the shape of cutting edges it is possible to determine parameters which in turn can give an indication whether the used cutting edge can be reused by reworking. Parameters to determine the reusability of the cutting edge are not part of this thesis. In the first approach without machine learning a Gaussian filter was used for reconstruction. This filter interprets the occurring defects as measuring errors and thus corrects them. In combination with a bicubic smoothing spline the samples are smoothed accordingly. With this approach very good results were achieved for simulated and real cutting edges. For samples with stronger curvature and defects the reconstruction was not always successful. The method with convolution-based neural networks, more precisely with an autoencoder, also achieved very good results for simulated and real samples. Furthermore, several types of cutting edges could be learned with one model. These were able to reconstruct any shape of the learned cutting edge. However, this approach also has some disadvantages. Since the density of the recorded points can be different for cutting edges and thus also for the trained model with the desired weights, a scaling of these points is necessary in some cases. Furthermore, as with all approaches for machine learning, the data is highly dependent on the existing training data. Although this thesis describes an approach to generate training data by using measurement, the use of real data is always preferable. Even if the preliminary work of data collection or generation and its possible scaling are taken into account, it can be summarised that for this last task the machine learning approach achieves better results. This is not only true for the presented results, but as a result for the considered evaluation points in section 6.3.

In a further comparison, all results of the previous comparative analyses based on machine learning and the model-based approach were compared. When comparing the model-based approaches and machine learning, the overall comparison is so balanced that it makes more sense to select a method depending on the respective task area than to derive a general statement.

German Version

Die Automatisierung und die damit verbundenen Technologien spielen eine immer wichtigere Rolle in unserem Leben. Ein wichtiger Pfeiler der heutigen Automatisierung und Forschung ist der Bereich der künstlichen Intelligenz. Künstliche Intelligenz wird sehr erfolgreich für verschiedene Anwendungen und Automatisierungsprozesse eingesetzt, die vorher nicht gelöst werden konnten. Allerdings ist ihr Einsatz nicht immer gewinnbringend, da es bereits viele gute Ansätze gibt, die nicht einem maschinellen Lernansatz folgen. In dieser Arbeit wollen wir verschiedene und aktuelle Forschungsfelder im Bereich der Messtechnik betrachten und versuchen, diese auf zwei verschiedene Arten zu lösen. Im ersten Ansatz werden nur modellbasierende Methoden verwendet. Im zweiten Teil werden Ansätze mit maschinellem Lernen eingesetzt. Die mit beiden Methoden erzielten Ergebnisse werden verglichen und bewertet. Es wird unter anderem diskutiert, ob Methoden des maschinellen Lernens im Bereich der Messtechnik im Mikrometerbereich sinnvoll auf aktuelle Forschungsgebiete angewendet werden können. Auch im Hinblick auf die starke Abhängigkeit von Trainingsdaten und den hohen Automatisierungsgrad.

Im Wesentlichen werden zwei aktuelle Forschungsgebiete im Bereich der Metrologie betrachtet. Das erste Forschungsgebiet beschäftigt sich mit Mikrodrall und umfasst zwei Kernaufgaben, die Schätzung der Texturrichtung und der Werkzeugachse. Mikrodrall tritt beim Schleifen einer Welle auf und kann zu unerwünschten Eigenschaften der Welle führen. Während es für eine andere Art von Drall, dem sogenannten Makrodrall, ein genormtes Verfahren gibt, das in der Mercedes-Benz Norm MBN 31007-7 definiert ist, gibt es für die Mikrodrall derzeit keine Normung. Wird die so genannte Fadenmethode zur Bestimmung von Drall nicht angewendet, sind Mikrodrall Parameter zur Charakterisierung der Texturrichtung und der Werkstückachse notwendig. In dieser Arbeit wird eine Methodik zur Bestimmung der Texturrichtung abgeleitet, die auf einem Modelansatz basiert. Diese Methodik, entwickelt von Prof. Dr.-Ing Jörg Seewig, wird an simulierten und realen Oberflächen validiert. Sowohl für simulierte als auch für reale Oberflächen wurden Ergebnisse mit geringer Streuung erzielt. Für simulierte Oberflächen konnte die Texturrichtung mit einer Genauigkeit von weniger als einer Bogenminute bestimmt werden. In einem zweiten Ansatz wurden Methoden aus dem Bereich der künstlichen Intelligenz verwendet, genauer gesagt ein Faltungsneuronales

Netz. Allerdings wurden für das Training nur simulierte Daten verwendet, da reale Daten für überwachtes Lernen für diese Anwendung schwer zu realisieren sind. Hier ist noch eine Validierung erforderlich, um zu untersuchen, ob auch für real gemessene Texturrichtungen gute Ergebnisse erzielt werden können. Für simulierte Oberflächen konnte jedoch in den meisten Fällen eine Auflösungsgenauigkeit von weniger als einer Bogenminute erreicht werden. Allerdings wurde auch beobachtet, dass das neuronale Netz bei untrainierten und damit unbekannten Daten weniger gut abschneidet. Es wird daher empfohlen, ein neuronales Netz zu verwenden, das mit einem breiten Spektrum von Bogenminuten-Texturrichtungen trainiert wurde. Beim Vergleich der Ergebnisse für die Texturrichtungsbestimmung beider Methoden wird die modellbasierte Lösung zur Texturrichtungsbestimmung für reale Oberflächen empfohlen, da beim maschinellen Lernansatz die Ergebnisse für untrainierte Oberflächen schlecht sind und die Validierung hierfür fehlt.

Die Bestimmung der Werkzeugachse für den Mikrodrallwinkel erfordert die Messung einer unterschiedlichen Anzahl von Messflächen entlang des Umfangs und der Höhe der Welle. Diese Datenpunkte werden verwendet, um die idealisierte Geometrie eines Zylinders einzupassen. Hier wurden keine realen Messdatenpunkte verwendet, sondern nur Simulationen. Die Anzahl der Messflächen wurde von den vom National Institute for Standards and Technology (NIST) empfohlenen Anzahl 108 reduziert, bis am Ende nur noch eine Messfläche übrig war. Dieser Bereich hatte, wie alle anderen simulierten Messbereiche, eine Größe von $0,8\mu m \times 0,8\mu m$. Die Einpassung durch Methoden mit und ohne maschinelles Lernen zeigte für bis zu vier simulierte Messbereiche ähnlich gute Ergebnisse. Für einen einzelnen Messbereich zeigte die Anpassung mit maschinellem Lernen eine geringere Abweichung von der vorhandenen Grundwahrheit als die modellbasierte Methode, insbesondere für den Anpassungsradius des Zylinders. Dies ist darauf zurückzuführen, dass die Trainingsdaten eine Art A-priori-Wissen liefern. Dies kann auch durch einen Bayes'schen Schätzer für die Methodik ohne maschinelles Lernen realisiert werden. Ein Vergleich der beiden Methoden im Abschnitt 6.2 hat gezeigt, dass der modellbasierte Ansatz dem vorgestellten Ansatz des maschinellen Lernens unter Berücksichtigung verschiedener Aspekte deutlich vorzuziehen ist.

Der letzte in dieser Arbeit analysierte Bereich ist die Rekonstruktion von Messdaten mit unerwünschten Defekten. Hier soll eine ideale Referenz erzeugt werden, wobei als Input eine Messung mit Defekten und Imperfektionen eingeladen wird. Die ideale Referenz ermöglicht die Auswertung von Parametern zur späteren Beurteilung des gemessenen Bauteils. In dieser Arbeit wurden als Beispiel Schneidkanten gewählt. Durch die Rekonstruktion der Form von Schneidkanten ist es möglich, Parameter zu bestimmen, die wiederum einen Hinweis darauf geben können, ob die verwendete Schneidkante durch Nacharbeit wiederverwendet werden kann. Parameter zur Bestimmung der Wiederverwendbarkeit der Schneidkante sind nicht Bestandteil dieser Arbeit. Im ersten Ansatz ohne maschinelles Lernen wurde ein Gauß-Filter zur Rekonstruktion verwendet. Dieser Filter interpretiert die auftretenden Defekte als eine hohe Messabweichungen und korrigiert sie somit. In Kombination mit einem bikubischen Glättungsspline werden die Proben entsprechend geglättet. Mit diesem Ansatz wurden sehr gute Ergebnisse für simulierte und reale Schnittkanten erzielt. Bei Proben mit stärkerer Krümmung und Defekten war die Rekonstruktion nicht immer erfolgreich. Das Verfahren mit faltungsbasierten neuronalen Netzen, genauer gesagt mit einem Autoencoder, erzielte auch bei simulierten und realen Proben sehr gute Ergebnisse. Außerdem konnten mit einem Modell mehrere Arten von Schneidkanten erlernt werden. Diese waren in der Lage, jede beliebige Form der erlernten Schneidkante zu rekonstruieren. Dieser Ansatz hat jedoch auch einige Nachteile. Da die Dichte der aufgezeichneten Punkte für Schneidkanten und damit auch für das trainierte Modell mit den gewünschten Gewichten unterschiedlich sein kann, ist in einigen Fällen eine Skalierung dieser Punkte notwendig. Außerdem sind die Daten, wie bei allen Ansätzen des maschinellen Lernens, stark von den vorhandenen Trainingsdaten abhängig. Obwohl in dieser Arbeit ein Ansatz zur Erzeugung von Trainingsdaten durch eine vorhandene Messung beschrieben wird, ist die Verwendung von realen Daten immer vorzuziehen. Selbst wenn die Vorverarbeiten der Daten bzw. -generierung und deren mögliche Skalierung berücksichtigt werden, lässt sich zusammenfassend sagen, dass für diese letzte Aufgabe der Ansatz des maschinellen Lernens bessere Ergebnisse erzielt. Dies gilt nicht nur für die vorgestellten Ergebnisse, sondern im Ergebnis auch für die betrachteten Bewertungspunkte im Abschnitt 6.3.

In einem weiteren Vergleich wurden alle Ergebnisse der bisherigen vergleichenden Analysen auf Basis des maschinellen Lernens und des modellbasierten Ansatzes gegenübergestellt. Beim Vergleich von modellbasierten Ansätzen und maschinellem Lernen ist der Gesamtvergleich so ausgewogen, dass es sinnvoller ist, eine Methode je nach Aufgabenstellung auszuwählen, als eine allgemeine Aussage abzuleiten.

Keywords: Lead, Micro Lead, Texture Orientation, cylinder fitting, cutting edges, reconstruction, machine learning, deep learning, autoencoder.

Contents

Nomenclature **xv**

Motivation and task description **1**

1 Lead in Sealing Systems **4**
- 1.1 Categorisation of leads 5
  - 1.1.1 Macro lead 5
  - 1.1.2 Micro lead 6
- 1.2 Lead measurement method 7
  - 1.2.1 Thread method 8
  - 1.2.2 Algorithmic evaluation of macro lead 8
  - 1.2.3 Algorithmic evaluation of micro lead 9
  - 1.2.4 Evaluation of lead besides texture direction 11
- 1.3 Measurement to determine micro lead 12
  - 1.3.1 Optical areal measuring systems for micro lead 13
  - 1.3.2 Structure of the measuring system 14

2 Cutting edge rounding **17**
- 2.1 Manufacture 17
  - 2.1.1 Preparation method 18
- 2.2 Shape and Parameters for Characterisation 19
- 2.3 Measuring method for cutting edges 20
- 2.4 Reconstruction of Cutting Edges 21

3 Mathematical and Machine Learning Fundamentals 24
3.1 Mathematical Fundamentals 24
3.1.1 Signal Processing 26
3.1.2 Optimization 32
3.2 Basic Terms Machine Learning 34
3.2.1 Training Set, Validation Set and Test Set 35
3.2.2 Overfitting and Underfitting 36
3.3 Deep Learning und Autoencoder 37
3.3.1 Multi-Layer Perceptron 38
3.4 Convolutional Neural Network 41
3.4.1 Convolution Architectures 41
3.4.2 Autoencoder for CNNs 42

4 Model-based Approach for determining Micro Lead and Reconstruction 43
4.1 Determination of Texture Direction 44
4.1.1 Cross-correlation based Texture Orientation Detection 44
4.1.2 Results and Discussion 47
4.2 Determine the Direction of a Shaft for Micro lead 52
4.2.1 Nonlinear Fitting for 3D Cylinder Points 53
4.2.2 Application and Validation for Simulated 3D Cylinder Points . 56
4.3 Reconstruction of Cutting Edges using a Robust Filter 62
4.3.1 Generation of Simulated Cutting Edges 62
4.3.2 Determination of Ideal References for Cutting Edges using model-based methods 68
4.3.3 Application and Validation of the model-based Approach for Reconstruction of Simulated Cutting Edges 70

5 Machine learning based approach for determining Micro Lead and Reconstruction 78
5.1 Determining the texture direction using neural networks 79
5.1.1 Architecture of a Convolutional Neural Network for determining Texture Direction 79
5.1.2 Results and Discussion 81
5.2 Determining the direction of a shaft with neural networks 86

5.2.1 Generation of 3D Cylinder Points 87
5.2.2 Neural Network Model for Fitting a Cylinder 88
5.2.3 Application and Validation for Simulated 3D Cylinder Points of the used Model 90
5.3 Reconstruction of Cutting Edges using Deep Autoencoder 90
5.3.1 Preparation of the Data and Idea Description for using Autoencoder for Reconstruction 91
5.3.2 Architecture of the used Deep Autoencoder for generating ideal Cutting Edges 92
5.3.3 Application and Validation of an Autoencoder Approach to Reconstruct Simulated Cutting Edges 93

6 Comparing model-based and machine learning approaches 102
6.1 Texture Direction for Micro Lead 103
6.2 Cylinder Fitting for Micro Lead 107
6.3 Reconstruction of Cutting Edges 112
6.4 Comparison and interpretation of the Results 118

7 Summary and Outlook 119

A Preprocessing to determine the Texture Direction for Micro Lead 122
A.1 Median Filter 122
A.2 S-Filter 123
A.3 F-Operator 123

B Alternative Options for choosing an Autoencoder to realise a Ideal Reconstruction of a Cutting Edge 124

C Basics of Mathematics Foundations and Machine Learning 126
C.1 Basic Definitions and Notations 126
C.2 Basics of linear Algebra 128
C.3 Basics of Probability Theory 134
C.4 Optimization 136
C.5 Fundamentals Machine Learning 138
C.5.1 Learning 138

C.5.2 Regularization 140
C.5.3 Ensemble learning 141
C.5.4 Additional Note 143
C.6 Fundamentals Deep Learning 143
C.6.1 What is a Neural Network? 144
C.6.2 Back-propagation 148
C.7 Fundamentals of Convolution Neural Network 152
C.7.1 Convolution Matrix 152
C.7.2 Convolution Layer 154
C.7.3 Convolution Transposed 157
C.8 Autoencoder 158
C.8.1 Deep Autoencoders 158
C.8.2 Tying Weights 158
C.8.3 Denoising Autoencoders 159
C.8.4 Other Autoencoders 159

Bibliography **166**

Nomenclature

Blackboard Bold

$\mathbb{K}$	Arbitrary field
$\mathbb{N}$	Set of natural numbers
$\mathbb{P}_n$	Polynomial ring with degree $n \in \mathbb{N}$
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^n$	n-dimensional space of real numbers
$\mathbb{R}^{m \times n}$	Set of all matrices with m rows and n columns where each element of the matrix is a real number and $m, n \in \mathbb{N}$
$\mathbb{S}_n$	Set of spline polynomials with degree $n \in \mathbb{N}$

Vaiables defined with Calligraphic font

$\mathcal{F}[\cdot](\cdot)$	Fourier-Series, see definition 3.1.1
$\mathcal{F}\{\cdot(\cdot)\}(\cdot)$	Fourier-Transformation, see definition 3.1.1
$\mathcal{F}^{-1}\{\cdot(\cdot)\}(\cdot)$	Inverse Fourier-Transformation, see definition 3.1.1

Variables defined with ancient Greek symbols

$\langle \mathbf{w}, \mathbf{x} \rangle$	Active function, see figure C.4
(Ω, Σ, P)	Probability space
Ω	Sample Space, see figure C.2

Ψ	Micro lead angle, see section 1.1.2
Σ	σ-Algebra over Ω, see figure C.2.
Θ^{ax}	Axis direction, used parameter for micro lead. See section 1.1.2
Θ^{circ}	Circumferential direction, used parameter for micro lead. See section 1.1.2
Θ^{t}	Texture direction, used parameter for micro lead. See section 1.1.2
$\varphi(x)^{ELU}$	Active function, see figure 3.4 and equation 3.29
$\varphi(x)^{ReLU}$	Active function, see figure 3.4 and equation 3.29
$\varphi(x)^{sig}$	Active function, see figure C.4 and equation 3.29
$\varphi(x)^{tanh}$	Active function, see figure 3.4 and equation 3.29
P	Measure, see figure C.2.

Vaiables defined with Latin symbols

(L)	Any arbitrary layer $(\cdot)$
$\bar{x}$	$\bar{x}$ as conjugated complex function of x
$\mathbf{A}$	A matrix is notated with a large, bold Latin letter.
$\mathbf{A}^{+}$	Moore-Penrose Pseudoinverse of a matrix $\mathbf{A}$, see also theorem C.2.3.
$\mathbf{v}$	A vector is notated with a small, bold Latin letter.
$\mathbf{W}$	Input for convolution neural network, see section C.7
$\mathbf{X}$	Kernel for convolution neural network, see section C.7
$\mathbf{x}^{(cri)}$	critical point, see theorem C.4
$\tilde{y}_j$	Desired activation of neuron j in the last Layer (L)
C_j	Cost value between $\tilde{y}_j$ and $y_j^{(L)}$
c_k	Fourier-coefficients, see definition 3.1.1
$I^{(act)}(\mathbf{x})$	Active index set, see definition 3.1.2

MSE	Mean square error, see definition 3.1.3
$n^{(\cdot)}$	number of neurons in any arbitrary layer $(\cdot)$
p	Zero padding for convolution neural network, see section C.7
$p(x \mid \mu, \sigma^2)$	The normal distribution, see equation 3.4
$P(x)$	Probability mass function, see definition C.3.2
$p(x)$	Probability density function, see paragraph C.3
$p^{\text{(unif)}}(x \mid a, b)$	The uniform distribution, see equation 3.5
PCA	principal component analysis, see section C.2
PDF	Acronym for Probability density function, see paragraph C.3
s	Stride for convolution neural network, see section C.7
w_0	The first weights value with index 0 is called bias
w_i	Weights with index i
$w_{(L-1),k}^{(L),j}$	Weight of neuron k in layer $(L-1)$ to neuron j in (L)
$y_j^{(L)}$	Activation of neuron j in Layer (L)
$y_k^{(L-1)}$	Activation of neuron k in Layer (L-1)

Mathematical symbols

$\arg\max f(x)$	The arguments of the maxima are the points at which the function values are maximized.
$\arg\min f(x)$	The arguments of the maxima are the points at which the function values are minimized.
$(\cdot * \cdot)(\mathbf{t})$	Convolution, see definition 3.1.4.
$(\cdot \star \cdot)(\mathbf{t})$	Cross-correlation, see definition 3.1.5
$\mathbf{det}(A)$	The determinant of a matrix $\mathbf{A}$.

$\mathbf{diag}\left(\begin{pmatrix}\lambda_0 & \dots & \lambda_{n-1}\end{pmatrix}\right)$	Notes the diagonal elements of a matrix. All other entries outside the diagonal are zero.
$\mathbf{H}\left(f(\mathbf{x}_0)\right)$	Hessian matrix of a scalar field f, see definition C.1
$\mathbf{J}\left(f(\mathbf{x}_0)\right)$	Jacobian matrix of a vector field f, see definition C.1.7
$\mathbf{rank}(A)$	Rank of a matrix $\mathbf{A}$.
$\max f(x)$	Largest value of a function f.
$\min f(x)$	Smallest value of a function f.
$\nabla\left(f(\mathbf{x}_0)\right)$	Gradient of a scalar field f, see definition C.1
$C^q(b)$	q-times continuously differentiable functions of a domain b

Motivation and Task Description

Model-based approaches based on underlying assumptions and described by mathematical models are often used to solve tasks in the field of micro measurement. When looking at tasks within and outside metrology, there is a tendency towards more and more solutions based on machine learning. In this thesis, the question is investigated whether learning models based on data and using machine learning approaches can provide comparably good or even better results than the methodology of classical modelling, which is still frequently used in micro-measurement technology. At the end of this thesis, an almost balanced result can be derived, whereby the approaches with machine learning have a small advantage in the overall evaluation due to the still existing optimisation potential.

This comparison is interesting because model-based methods often require a deep knowledge of the task to be solved and also a high degree of mathematical understanding in order to combine meaningful approaches with this deep knowledge. In the machine learning approach, the weights to be determined for functions are learned from data and thus a modelled description is derived without the use of partially complex approaches. However, it should be mentioned that a well-chosen architecture and approach of e.g. deep neural networks is necessary to achieve good and meaningful results. In order to enable an evaluation of this different approach, two current research areas were considered.

The first field of research concerns the investigation of leakage that can be caused by lead. Here, the so-called micro lead is studied in more detail, which in turn consists of two sub-task. The first associated task enforces the determination of the texture direction of ground surfaces in arc minute accuracy. The second associated task determines the position of the measured shaft with high accuracy so that the axis of the workpiece can be specified. If both results are combined, the so-called micro lead angle can provide information about an existing leakage. A more detailed description of lead and micro lead in particular can be found in chapter 1. Methods using a classical modelling approach and solving the described tasks related to micro lead are presented in section 4.1 and section 4.2. The machine learning approach for determining the texture direction and the location of the corresponding measured shaft in space is described in section 5.1 and 5.2. The results of the classically based modelling and the learned modelling with machine learning are compared and evaluated in the sections 6.1 and 6.2 under consideration of different aspects.

The second research area in this thesis is the reconstruction of cutting edges. Cutting edges and their rounding, parameters for characterisation and a possible measurement method for capturing these cutting edges are described in more detail in chapter 2. Approaches to reconstruct the cutting edges based on classical modelling approaches are presented and validated in section 4.3. The machine learning architecture and thus the approach of learning the modelling weights are presented in section 5.3. Both methods for reconstructing the cutting edge are contrasted in section 6.3 and evaluated and compared with each other from different points of view.

All comparisons for the investigated research areas of micro lead and cutting edge reconstruction are also compared with each other in the section 6.4 to allow an assessment of whether it makes sense to learn modelling weights in the field of micrometrology. This is often particularly critical in the field of micrometrology, as measurement data and thus an important building block for learning meaningful weights are often not readily available.

Chapter 7 gives an outlook and summary of the thesis and thus also other possible contents and tasks for further investigation.

Another motivation and challenge of this thesis lies in the social sphere and in the realisation of this thesis. Science should be freely accessible to all people in this world. This does not only include the written elaboration of this thesis. The tools used for this thesis are also free of charge. The algorithms and methods developed were therefore realised with free programming languages, mainly Python 3. The integrated development environments used are free, as are the tools for creating images. Some of the developed approaches can be viewed and tested for free on my Git repository. However, there is an exception for a developed model-based method. It uses a robust filter from ISO-11610 called the robust Gaussian filter. This filter gave the best results in an application. Alternatively, free robust algorithms can be used for the solution.

1 Lead in Sealing Systems

In the Mercedes-Benz factory standard MBN 31007-7, lead is defined as a texture occurring over the entire circumference of the shaft sealing surface [1]. Lead can interfere with the proper functioning of the sealing system and can cause the component to leak by affecting the axial fluid transport. Other sources define lead as any surface feature that occurs and affects fluid transport. Thus, in contrast to the Mercedes-Benz standard, it is not limited to the condition of the shaft sealing surface. Another good definition that includes all surface features that occur can be found in [2]: From a sealing technology point of view, lead is basically all types of surface structures that are able to actively convey fluid axially into sealing contact during rotation of the shaft to be sealed. These include, in particular, characteristic, anisotropic surface structures that originate specifically from the manufacture of the seal mating surface. If these deviate in their orientation from the circumferential direction of the shaft, a micropumping effect is created, similar to that of a screw pump or a threaded shaft seal.

In this thesis, the definition is taken from the Mercedes-Benz works standard MBN 31007-7 and is based on the texture direction of the shaft sealing surface. In the Mercedes-Benz factory norm MBN 31007-7 itself, but also in other publications such as [3], groove-ground surfaces are differentiated into a “fine” and a “large” texture, which overlap each other.

On this basis, the different types of lead are presented in section 1.1. This section describes the macrostructure, which is referred to as macro lead in the Mercedes-Benz factory norm, and the microstructure, which is referred to as micro lead in the Mercedes-Benz factory norm. The possibility to measure and determine the lead values is presented in section 1.2. This section presents algorithms that can be used to determine texture direction for micro lead, reflecting the current state of the art. In addition to texture direction, a known workpiece axis is also elementary for micro lead determination. The last section 1.3 will show why this is important and what approaches are available.

1.1 Categorisation of leads

Different lead features can be superimposed on one and the same surface. Krahe [4] and Kersten [3], for example, give a basic categorisation for groove-ground surfaces. Both distinguish between a fine and a macro structure [2]. Therefore, a single analysis method is not yet sufficient to fully perform a lead analysis [2], [1]. For this reason, it makes sense to introduce a categorisation for leads [2], [1]. According to the Mercedes Benz factory norm MBN 31007-7, lead structures are classified into macro and micro lead based on their properties and characteristics. In MBN 31007-7, the macrostructure described is called macro lead and the corresponding fine structure is called micro lead.

1.1.1 Macro lead

Axially periodic circumferential structures with one or more thread beginnings are called macro lead [5], [6]. Macro lead is in turn subdivided into four subcategories: Dressing lead, feed lead, periodic zero lead and non-periodic zero lead [1], [6]. Although the non-periodic zero lead contradicts the definition here, it is classified as macro lead due to its macro structure. For the classification of macro leads, parameters are defined in MBN 31007-7, which can also be seen in figure 1.1. These are lead angle D_γ, lead depth D_t, number of threads D_G, theoretical pump cross-section D_F, theoretical pump cross-section per revolution D_{Fu}, axial period length D_P and contact length ratio in percent D_{Lu}. The lead angle itself is a signed parameter value. The parameters described in MBN 31007-7 can be used to characterise the macro lead well, which is why MBN 31007-7 is often used for evaluation. This serves as the status quo of the current evaluation of macro leads.

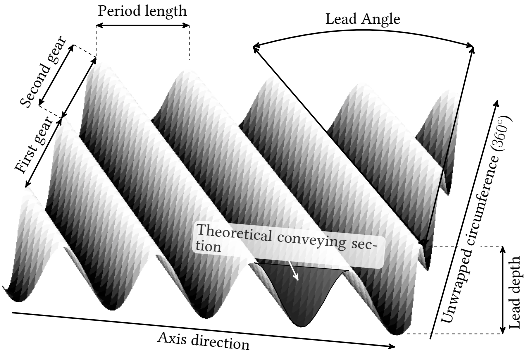


Figure 1.1: Characteristics of a macro lead structure. Illustration inspired by [1]

1.1.2 Micro lead

Micro lead can be defined as stochastically arranged, anisotropic structures with a width of the structures below $20\,\mu m$ [2]. The parameter describing the micro lead in MBN 31007-7, with the so called setting lead angle DS_γ, referred to in this thesis as micro lead angle Ψ, is a difference of the main texture orientation of the ground micro texture and the circumferential direction of the shaft [3], [7], [1]. This micro lead angle Ψ is signed and can be calculated from the difference of the main orientation of the micro texture of the surface Θ^t and the circumferential direction Θ^{circ} whose projection is perpendicular to the shaft axis Θ^{ax} [1], [6]. In this thesis, the micro lead angle is denoted by Ψ and can be calculated with

$$\Psi = \Theta^t - \Theta^{\text{circ}}. \tag{1.1}$$

A visual representation of the micro lead can be found in figure 1.2.

The name setting lead from MBN 31007-7 is derived from a setting or misalignment between the workpiece and the grinding wheel during production, which is an undesirable condition [6]. This is usually very small and is in the range of a few minutes of arc. A larger deviation would be easy to detect and can therefore be avoided by the operator. In addition, grinding with axial feed can produce sloping structures. Details and formulae for these two mechanisms of microleading can be found in [3]. A definition describing all describable causes of microlead has been described by [5].

In this thesis, micro lead is determined by the deviation of the ground texture. For this purpose, methods are presented to determine the texture direction Θ^t and the tool axis Θ^{ax} and thus the circumferential direction Θ^{circ}. Scratches and other features that encourage leakage are not examined further.

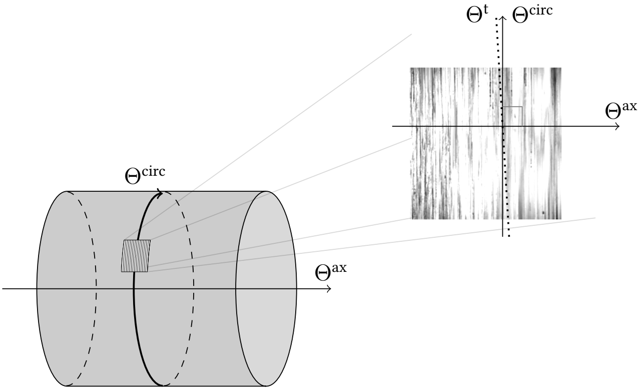


Figure 1.2: Shown is a shaft with an axis direction Θ^{ax} and a circumferential direction Θ^{circ}. A section of the shaft was measured and the texture direction Θ^{t} was determined. The difference between $\Psi = \Theta^{\text{t}} - \Theta^{\text{circ}}$ returns the micro lead angle. The figure also shows that if Θ^{ax} is known, the circumferential direction Θ^{circ} can be calculated. This illustration is inspired by [8] and previously published in [9].

1.2 Lead measurement method

A number of methods have been developed for evaluating lead, which are also widely used in industry. The thread method and the Mercedes-Benz factory norm MBN 31007-7 [1] are among the most commonly used methods. While the thread method can make a statement about macro and micro lead, the Mercedes-Benz factory norm MBN 31007-7 is used to determine macro lead. Currently there is no standardised method for the micro lead angle Ψ, but there are very good approaches and developments for its determination.

In section 1.2.1 the thread method and its advantages and disadvantages are briefly described. In section 1.2.2 the evaluation of the macro lead with the Mercedes-Benz factory standard MBN 31007-7 is described and in the last section 1.2.3 methods for the determination of the micro lead.

1.2.1 Thread method

The thread method is one of the oldest measuring methods for determining the pitch on seal running surfaces. For this purpose, the shaft to be examined is fixed so that it can rotate horizontally at a defined, adjustable speed. It can also be rotated in both directions. A thread with a defined weight and a wrap angle of approximate 220° to 240° is then placed around this shaft [2]. If the shaft rotates in the axial direction and the thread moves, it indicates the presence of lead. This observation can be done visually with the eyes, but often observation with the help of a camera or optics is preferred.

A clear advantage of this method is to get a simple and quick indication of whether there may be leackage for the shaft. The disadvantage of this method is that the results are partly dependent on different settings. For example, smooth nylon threads can be used in addition to cotton threads, whereby cotton threads often react to microscopic lead structures and nylon threads to macroscopic lead structures. In addition to the thread, the defined weight and/or circumferential speed can also be varied. In addition, the measurement can be changed by using oil that is between the shaft and the thread. This can make the application of the thread method problematic, as a comparison of results from different institutes is very difficult or not possible at all [2].

1.2.2 Algorithmic evaluation of macro lead

The current state of the art for macro lead evaluation is the latest version of the Mercedes-Benz standard MBN 31007-7 [1]. It is described in [5], [10] and is referred to as the second generation lead evaluation [6]. The first generation of macro lead determination uses the evaluation of Rau et al [7], [11]. It is called the first generation macro lead evaluation [10] or sometimes the CARMEN method [12].

The measurement data is generated by capturing 72 axial profiles over a circumferential range of 360°, i.e. every 5°. In addition, a further 72 axial profiles are required over a circumferential range of 36 $^\circ$, i.e. every 0.5 $^\circ$. The macro lead parameters obtained from these measurement data, cf. figure 1.1, are generally accepted and serve as a reference.

The use of a single implementation in all companies also enables the comparability of the parameters. In addition to the tactile measurement of these 72 axis profiles, the measurement can also be carried out optically and the macro lead parameters can be evaluated with MBN 31007-7. Essentially, two prerequisites must be fulfilled for this. The measuring length of $2\,\mathrm{mm}$ of a profile length must be ensured so that the topography in axial direction is sufficiently long. Furthermore, the axial scanning distance must be sufficiently fine, as also required in MBN 31007-7. However, caution is also required here, as in older publications before 2009 the Mercedes-Benz standard MBN 31007-7 differs from the current one. Thus, the values from older publications are most likely not comparable with the new values.

In addition to the widely used Mercedes-Benz norm and the briefly mentioned CARMEN method, there are other methods for determining macro leads. An overview of many older applications or different approaches can be found in [6].

1.2.3 Algorithmic evaluation of micro lead

The micro lead angle Ψ, see equation 1.1, is generally very small. From equation 1.1 it becomes evident that besides the texture direction Θ^t also the determination of the circumferential direction Θ^{circ}, whose projection is perpendicular to the shaft axis Θ^{ax}, is very important for the determination of the micro lead angle Ψ. At this point it should be mentioned that many authors neglect the determination and the calibration of the tool axis in addition to the determination of the texture direction, as for example in the works of Buhl [13] and [14]. A detailed analysis of the determination of the tool axis alongside the texture direction can be found in the works of Kersten [3], Cohen et al [15], P. Arnecke [6] and M. Baumann [2].

Kersten [3] investigated whether and at which micro lead angle leakage is present. According to Kersten, no leakage is present if the deviation is smaller than $5'$. However, if the deviation is greater than $10'$, the shaft has leakage. If the micro lead angle is in between, no statement can be made. Methods to detect texture orientation are discussed in many literature sources, e.g. in [16], [6]. A standardised measure for **texture orientation** is the parameter surface texture direction Std from ISO 25178-2 [17] [18]. The parameter Std is calculated by the direction of the maximum of the angular spectrum of a surface [19]. The authors of [8] were able to show that the parameter Std

does not provide a satisfactory resolution in arc minutes. However, there are methods, often based on the Radon transform or the evaluation of the areal power spectral density, which are able to realise the desired resolution. Arnecke [6] uses an approach based on the Radon transform, which uses an algorithm to gradually divide the texture direction into finer intervals. This is according to [6] and [2] able to realise the determination of the texture direction within one arc minute. In addition to the standardized parameter Std, a method based on the power density spectrum is also used by the company Bruder to determine a micro lead angle [2]. Another method based on power density spectrum is used by Puente Leon, who suppresses noise by averaging several images and makes the result robust [12]. Not an integral method, but a structure-based lead measurement was proposed by Kunstfeld [14] and developed by Baitinger [20]. Baumann showed in [20] a high repeatability on real samples and a reliable statement about micro lead with this method. The raw measured data is processed and the high-frequency structural components are separated from the low-frequency components. Then only the high-frequency micro-lead structures are remaining. These data are based on measurements of a shaft on 10 different days and the calculation of the arithmetic mean of the micro lead angle. Baitinger mentions the direction-dependent cumulative structural volume of the grinding grooves as a decisive feature of micro lead. Baitinger presents the results in a distribution diagram. To create the diagram, the volumes of all structures oriented in a certain angular position are summed up and plotted over the angular position, see figure 1.3. The zero degree position of the distribution diagram then corresponds to the circumferential direction Θ^{circ} of the shaft. If the distribution of the cumulated structure volume is symmetrically no leakage occurs. An uneven distribution of the accumulated structural volume indicates leakage. In order to determine the micro lead angle Ψ satisfactorily the determination of the circumferential direction Θ^{circ} using Θ^{ax} is inevitable. A statement about the micro lead angle in arc minutes is only possible by a conclusion with a very known position of the tool axis and thus of the parameter in circumferential direction Θ^{circ}. Arnecke use the basic idea of a calibration shaft. After adjusting the eccentricity and tilt, the axis of rotation is ideally identical for all shafts, with a small deviation in concentricity. Thus a good approximation of the shaft axis is achieved [6]. The described method by Baumann to determine the circumferential direction Θ^{circ} is using a cylinder fit. After the shaft has been fixed in a measuring device, eccentric and tilt are set and several high-resolution measurements in circumferential

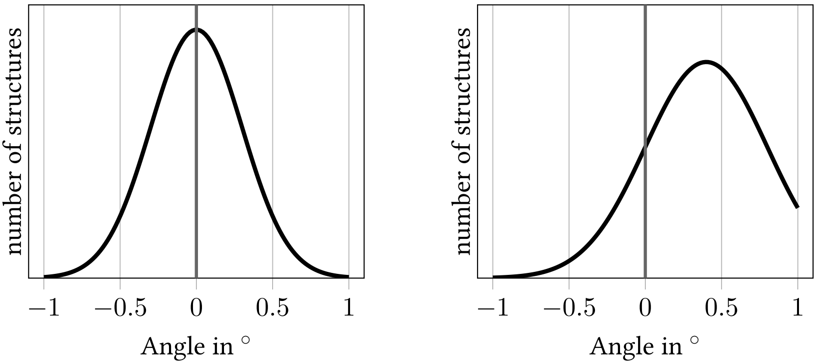


Figure 1.3: Schematic representation of the cumulated structure volume. An ideal symmetrically distribution for oriented structure is given left. This will not cause leakage. The figure on the right shows an asymmetrical distribution. Leakage is to be expected. Illustration inspired by [2].

direction and for three different heights are carried out. Baumann took a total of 108 individual measurements for a satisfactory result. For this purpose, Baumann also relied the number of measurements on the results of the National Institute for Standards and Technology (NIST) [21] [2].

In summary, it can be noted that the methods presented by Arnecke and Baumann give a good indication of the texture direction Θ^{t}. Arnecke determines the texture direction with an accuracy of less than one arc minute. For the evaluation of micro lead, Baumann carried out extensive investigations and achieved well repeated results. Both methods require effort to determine the circumferential direction Θ^{circ}. Arnecke realizes this with a calibration standard and Baumann with 108 measurements on one shaft. Finally, it should be noted that pre-processing work is required to assess the texture direction. These aspects will not be discussed in detail here. A brief description can be found in appendix A.

1.2.4 Evaluation of lead besides texture direction

Besides micro lead, which can be defined as a stochastically arranged anisotropic structure with a width of the structures of less than 20 µm, and the macro lead, as an axial peroidal macroscopic structure, there are also **defects** and a stochastically arranged, aperiodic, isotropic or anisotropic macroscopic structure, which can be defined as a **microwave** and can lead to leakage [2]. Since defects can self-explanatorily lead to leakage, we focus more on the described aperiodic microwave. A comprehensive characterisation of defects can be found in DIN EN ISO 8785 [22].

Aperiodic Microwaviness

Aperiodic waves occur to varying degrees on both ground and alternatively machined surfaces. When groove grinding sealing surfaces, aperiodic microwaves are caused by machine vibrations, imbalance of the grinding wheel or variable or non-constant speed ratios. With a fixed speed ratio, an existing dressing spiral is mapped onto the surface with a constant slope. If the speed ratio changes, the slope of the guide structure changes at the same time. An already existing structure can be ground over with a different slope and thus be “cut off”. The result is a surface structure with macroscopic, interrupted aperiodic structural parts. [2]

Another machining process that favours aperiodic microwaves is shot peening of surfaces. However, these are rarely used for sealing surfaces, which is why they are not listed further here. Further details on machining and further execution can be found in [23].

In this thesis we restrict ourselves to the determination of micro lead and do not go further into the determination of periodic and aperiodic microwaves.

1.3 Measurement to determine micro lead

In order to be able to evaluate the micro lead angle Ψ, defined as the deviation of the main texture direction Θ^t from the circumferential direction of the shaft Θ^{circ}, the lateral surface of the cylinder must be measured. It is therefore obvious that in addition to determining the texture direction Θ^t, the circumferential direction of the shaft Θ^{circ} must also be determined and thus a calibration and adjustment of the measuring device used is of particular importance. Since the circumferential direction and the axis of the workpiece are perpendicular to each other, this term is used as a synonym in this thesis.

In this section 1.3 we would like to motivate optical measurement methods for this purpose and present the measurement method and measurement system used in this thesis, as well as the lens settings used and two ways of determining Θ^{circ}.

1.3.1 Optical areal measuring systems for micro lead

In order to measure and evaluate micro lead, a large number of measuring points and a high optical resolution are required. In addition, the size of the measuring field is crucial. Considering the manufacturing process of grinding the shaft, ideally it is not only lead-free and thus has a micro lead angle of $\Psi = 0'$, but it is also assumed that the grinding structures are evenly distributed over the circumference [1]. Here, depending on the literature, the circumferential length of the marks ranges between maximum values of $0.5\,\mathrm{mm}$ and $1\,\mathrm{mm}$ [6], [24]. These spatial properties of micro lead structures define the requirements for the measuring instruments.

These requirements are not only met by optical measuring devices, as also described in DIN ISO 25178-6 [25], but optical measuring methods offer relatively fast measurements with a manageable amount of preparation. Furthermore, optical measuring devices often provide identical measuring distances in both lateral directions and are well described and technically mature in the literature [6].

In general, the realised images should show a relatively large area of the cylinder to be able to show more grinding marks and other features. However, these should not be realised at the expense of resolution. To obtain such a resolution and image of the surface, a confocal microscope or a coherence scanning interferometer can be used. To achieve all the desired properties, this thesis used a confocal microscope with an objective lens with a magnification factor of 20 and an NA of 0.45. This allows a field size of $0.8\,\mathrm{mm} \times 0.8\,\mathrm{mm}$, an optical resolution of $0.685\,\mathrm{\mu m}$ and a lateral scan distance of $1.5625\,\mathrm{\mu m} \times 1.5625\,\mathrm{\mu m}$. For this thesis, the NanoFocus μ-surf explorer confocal microscope was used. The wavelength of the optical measuring device with LED light source is specified with $505\,\mathrm{nm}$. [26]

1.3.2 Structure of the measuring system

In order to determine the micro lead as accurately as possible, several measurements are taken around the circumference of the cylinder. Since it is a prerequisite that the grinding structure on the shaft behaves the same everywhere, the measurement on the circumference not only enables the detection of errors, but also the indication of the determined micro lead angle Ψ by an average value and its deviation by repeated measurement. To realise this, the rotation of the measuring object or the measuring device is necessary, whereas in our thesis the measuring object was rotated. This in turn was realised by a high-precision turntable that allowed adjustments in tilt and eccentricity.

Setup of the measuring system used

The measuring system used consists of a surface topographic measuring device, in this thesis the NanoFocus μ-surf explorer was used, a support frame and a rotary table in which the workpiece is fixed. It should be noted that the measured workpiece is clamped horizontally in the rotary table, as this facilitates the adjustment of the eccentricity and tilt. A sketch of the set-up can be seen in figure 1.4. The setup used has many degrees of freedom overall, as not only can the eccentricity and inclination be adjusted by the turntable, but positioning is also possible by the support arm to which the measuring device is attached. This support arm is a construction of the Chair of Measurement and Control Engineering at the Technical University of Kaiserslautern and was used unchanged for this thesis, i.e. without own partial construction.

Origin and determination of eccentricity and tilt

Clamping the component in the rotary table inevitably creates an eccentricity and inclination of the workpiece that is too large a deviation for a lead measurement. This can lead to the fact that even with a single measurement the measured surface, even after an alignment correction by an algorithm, the height profile of the surface can be mapped incorrectly or not completely. Furthermore, in real lead measurements, multiple measurements of the ground surface around the circumference can determine a statistically reliable statement. If the eccentricity and tilt of the shaft are not set correctly and due to the rotation of the shaft around its axis, surface images can occur where the

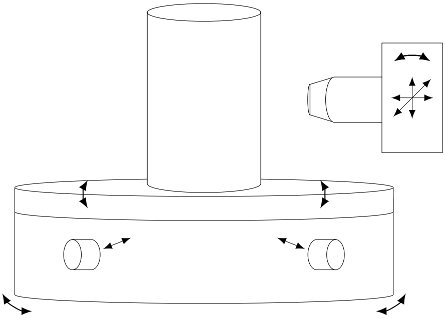


Figure 1.4: Sketch of the measuring setup used. The workpiece is fixed in a high-precision measuring device and is adjustable in its position and inclination. The measuring device on the right-hand side can also be moved translationally in any direction and can also be rotated. Illustration inspired by [6].

workpiece can be particularly close to the measuring device during a measurement and particularly far away from the measuring device after the workpiece has been rotated by 180°. This can even lead to the surface no longer being in the measuring range in both case.

In order to determine the eccentricity and tilt of the shaft and to determine the micro lead angle Ψ from the equation 1.1, conclusions about the parameter Θ^{circ} are necessary. Therefore, the position of the shaft at the end of the measurement must be known or determinable by measurements. The position of the shaft axis can be determined, as in section 4.2 and 5.2 or also in the thesis of [2], [15], by fitting a cylinder into measured surface sections. This approach of fitting a cylinder has the advantage that it does not require much additional effort other than the already collected measurement data. However, a disadvantage of this approach is that many measurements around the circumference and possibly at different cylinder heights are necessary to obtain a fit with very high accuracy, which is unavoidable for a micro lead estimation. P. Arnecke determines the orientation of the circumferential direction relative to the device by measuring the evaluated calibration groove. Arnecke has achieved very good results and tests his calibration scheme on several grooves. This has the advantage that the circumferential direction Θ^{circ} could be reliably determined with this method, but also the disadvantage that a corresponding calibration shaft must be manufactured for this.

This calibration shaft must meet the corresponding criteria and requirements and thus have a high-precision turned shaft with three grooves and sufficient space to adjust the eccentricity and inclination [6].

2 Cutting edge rounding

Cutting edge rounding is the round shape of a cutting edge used for machining [27]. The cutting edge itself is the part of a cutting tool that is effective in machining and where the cutting wedges are located [28]. The most important terms concerning their surfaces, cutting edges, reference systems and angles are standardised in DIN 6581 [29] [28]. Cutting edge rounding is used on cutting tools such as drills, milling cutters and indexable inserts to safeguard the cutting process, whereby many factors, such as tool life or cutting speed, play a role [27]. Exemplary cutting edges are shown in figure 2.1.

Like other tools, cutting edges and especially their rounding are subject to defects and wear in the course of their use. This thesis is particularly concerned with the reconstruction of cutting edges with wear characteristics and defects. To describe cutting edges in general and their condition, the first section 2.1 of this chapter describes how cutting edges are made. The second section 2.2 describes the shape of a cutting edge and its properties, before the third section 2.3 presents the measuring technique for measuring cutting edges. The last section 2.4 of this chapter gives an outlook on the reconstruction of cutting edges.

2.1 Manufacture

During machining, various cutting edge preparations are used to stabilise the cutting edge, reduce notch wear, increase surface quality or relieve the cutting edge with a protective chamfer during roughing. The cutting edge architecture influences the shape of the deformation zones, the temperature distribution, the residual stresses and the

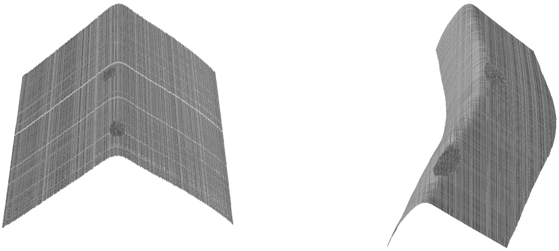


Figure 2.1: Simulated cutting edges are shown, with the cutting edge from the left image having a different profile than the cutting edge on the right.

cutting forces, so that wear resistance and surface integrity of the component depend significantly on the cutting edge design. [30] [31, pp. 12–33]

In particular, the type and extent of edge rounding is an important aspect of the cutting edge that fully describes a cutting edge preparation. In this context, a favourable cutting edge architecture is particularly advantageous because, in addition to increasing the reliability of the cutting edge preparation, it also delays the failure of the cutting edge and thus increases the tool life. In this way, the increased performance can also keep costs down for the customer. [32] [31, pp. 12–33]

A number of requirements are placed on ideal cutting edge rounding. For example, the prepared edges must meet the requirements of the application, such as geometry and variability, while the scatter must be kept within narrow limits. Furthermore, different input conditions must be selectively adjustable. [33] [31, pp. 12–33]

A cutting edge preparation can be achieved not only by rounding the cutting edge, but also by chamfering. However, this has disadvantages compared to applying the rounding to the cutting edge. In addition to the increased effort required for machining, a chamfered edge also leads more easily to breakage during prolonged use.

2.1.1 Preparation method

Through a process chain of grinding, brushing, blasting and coating processes, the micro-geometry is created in conventional edge preparation. Basic tests have shown the potential for process-dependent effects. For example, in drilling the tool life could be increased by 30% and in turning it was recognised that small symmetrical radii are advantageous. According to a Federal Ministry of Education and Research[1] project, optimising this process chain in edge processing also enables significant cost reductions and shorter production times while increasing the performance of the tools. [34] [31, pp. 12–33]

Depending on the desired edge radius, three main preparation methods are used. Sharp edges with an edge radius smaller than 5 µm are produced by grinding the rake faces. Medium cutting edges with a cutting radius between 5 µm and 20 µm can be produced by microblasting the rake and flank surfaces. Brushing the cutting edges produces large cutting radii with an edge radius greater than 20 µm [35] [31, pp. 12–33].

Many different preparation methods such as blasting, brushing [31, p. 34–52], magnetic finishing [31, p. 53–70] or edge preparation on indexable inserts [31, p. 71–86] can be further explored in the literature given.

2.2 Shape and Parameters for Characterisation

In summary, cutting edge rounding can be symmetrical or asymmetrical. If the cutting edge rounding is symmetrical, the rounding can be approximated in a very simplified way with a circle and a corresponding radius, see also figure 2.2. However, this way of characterising the cutting edge rounding using an edge radius r itself is not always sufficient, as the same edge radius may be present for different types of cutting edges, see figure 2.2. Furthermore, findings show that not only the edge radius r but the actual shape has a large influence on the cutting forces and tool wear [31]. If the cutting edge rounding has an asymmetrical shape, the setting of the rounding is often not done with the help of a circle, but with the help of an ellipse. In this case, the two semi-axes of the ellipse are evaluated instead of the edge radius r.

To obtain a complete characterisation of the cutting edge geometry, B. Denkena [36] proposes several parameters to describe the geometry. Here, the straight lines of the rake and flank face are extended to the point of intersection so that the points can be

---

[1] in German, Bundes-Ministerium für Bildung und Forschung (BMBP)

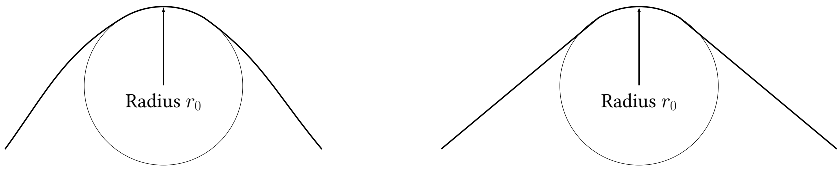


Figure 2.2: Approximating the cutting edge rounding by a circle is not always sufficient. Although the cutting edge is different on the left and right, the same circle with the same radius can be fitted in both cases. Illustration inspired by [36]

determined at which the cutting edge shape deviates from the extended straight lines of the rake and flank face, see also 2.3. The distances between these two points and the theoretical intersection are called cutting edge sections with S_γ and S_α. Now the highest point of the cutting edge can be determined and the distance Δr between this and the theoretical intersection of the rake and flank face can be determined. Furthermore, the ratio K between the cutting edge intersection points S_g and S_a is calculated. With this parameter combination, the geometry and orientation of the cutting edge is completely described and conclusions can be drawn about the application behaviour of the tool.

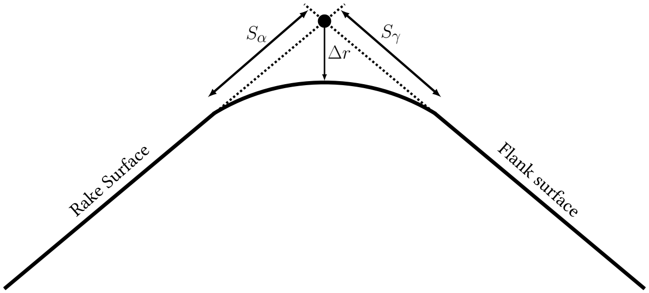


Figure 2.3: Parameters for the complete description of a cutting edge geometry. Illustration inspired by [36]

2.3 Measuring method for cutting edges

One challenge is the measurement technology for cutting edges and thus also for cutting edge rounding. To measure edge rounding, a sensor is needed that enables very high-resolution measurement in both depth and width. To measure a rounding of 10 µm radius, a resolving capacity of less than 1 µm is ideally required. This makes optical sensors necessary. To obtain favourable measuring conditions, the sample should be aligned approximately symmetrically to the light beam and have favourable optical properties. Strongly inclined measuring surfaces are problematic in this respect. In addition, the sensor should react dynamically, as large reflection gradients occur. Tactile measuring devices cause problems here due to the tactile peaks. In the range up to about 30 µm, fringe light projection is advantageous. [34], [31, pp. 12–33].

Focus variation combines the shallow depth of field of an optical system with vertical scanning to extract topographic and colour information from the variation of focus. In the following, the principle of operation is described by means of a system schematically shown in figure 2.4. The main component of the system is a precision optical system that contains various lens systems that can be fitted with different lenses, thus enabling measurements with different resolutions. The light emitted by a white light source is introduced into the system's beam path via a beam splitter mirror and focused on the sample via the lens. All rays emanating from the sample and hitting the objective lens are bundled in the optics and detected by a light-sensitive sensor behind the beam splitter mirror. Due to the shallow depth of field of the optics, only small areas of the object are sharply imaged. To achieve the full depth of field of the surface, the precision optics are moved vertically along the optical axis while continuously capturing data from the surface. This means that every area of the object is in focus. Algorithms convert the captured sensor data into 3D information and a true colour image with full depth of field. This is done by analysing the variation of the focus along the vertical axis. [37]

2.4 Reconstruction of Cutting Edges

Minor wear or even small micro-fractures can be corrected by reworking to such an extent that they can be used again afterwards. In order to obtain criteria for whether reworking is still possible or useful, parameters for characterising the cutting edge and its defects must be determined. This is done by measuring the cutting edge, as realised e.g. in the section 2.3 by focus variation. If the measured cutting edge has an ideal reference in the form of a CAD model, the difference between the ideal cutting edge and the measured cutting edge can be used to create a so-called difference model and analyse the defects in more detail, e.g. according to the existing length, width and depth of a defect. To create the difference model, only the alignment of the ideal cutting edges to each other must be realised in advance. However, if no ideal reference is available, which is often the case, an attempt can be made to create an ideal reference using the existing measurement and an algorithm, which is then used to create a difference model. The parameters can then be calculated to characterise and evaluate the cutting edge. The steps so far are summarised in figure 2.5.

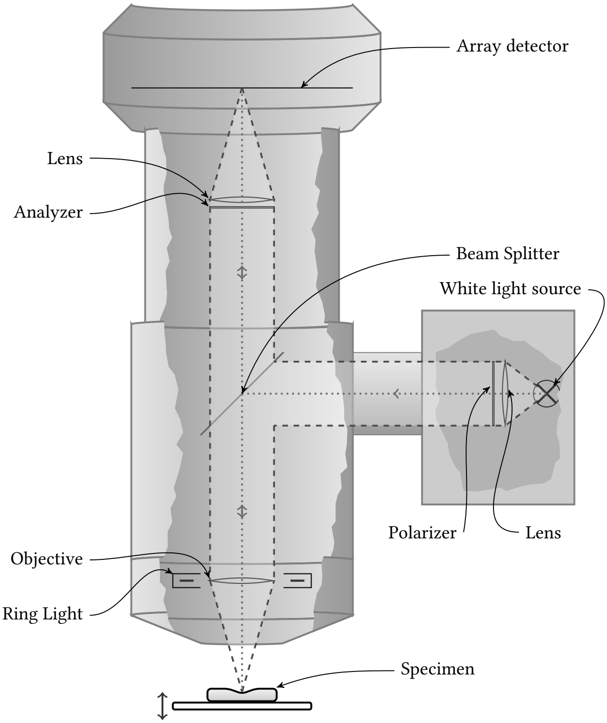


Figure 2.4: Schematic illustration of focus variation based measuring device. Illustration inspired by [37]. Foundation of the illustration based on [38]

Currently, there are not yet many implementations for cutting edge reconstruction, as many publications consider the process chain and the parameters for characterisation [40]. Traditional methods for measuring cutting edge quality are based on the manual

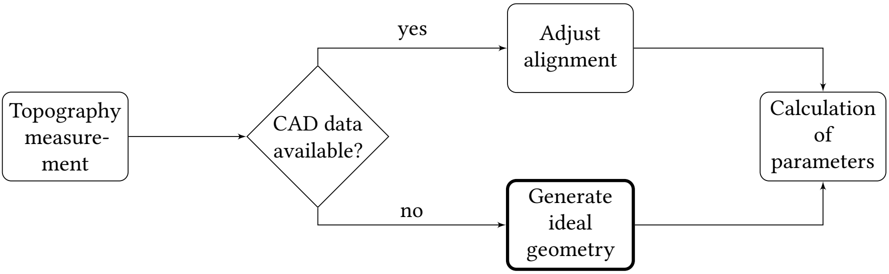


Figure 2.5: There are three steps to assess whether a cutting edge is still usable or not. If an ideal reference is available through a computerised model, a least squares method is used to make a match between the measured cutting edge and the ideal cutting edge and the parameters are determined by calculating the difference. If no computerised model is available, which is usually the case and the critical part, the measured cutting edge must be used to create an ideal reference. [39]

evaluation of high-resolution 2D images from a microscope, which are suitable for quantifying e.g. the defect length along the rake and flank surface, but do not provide 3D information and thus cannot determine e.g. the depth of a surface profile along the cutting edge [40]. The company Alicona has implemented an algorithm in its software that can also generate an ideal reference geometry based on the measured cutting edge. The Alicona algorithm is widely used in the industry and has been shown to give very good results for many different types of cutting edges. An earlier published algorithm by the author in [41], also presented in section 4.3, can also generate an ideal three-dimensional reference geometry based on the measured cutting edge. This gives very good results for many cutting edges and, according to [41], can even give better results than Alicona[2]'s existing solution for some cutting edges. Furthermore, there is an approach presented by the author in [39] that uses convolutional neural networks to generate ideal reference geometries from measurement data, which is also described in more detail in 5.3. This approach is interesting as it gives very good results for many different types of cutting edges. With this solution, all trained cutting edge types could be reconstructed very well, even if the cutting edge had significant defects or curvatures.

---

[2] State 2019

3 Mathematical and Machine Learning Fundamentals

In this chapter 3 the mathematical basics necessary to solve the problems of micro lead and cutting edge reconstruction are explained. Information on lead can be found in chapter 1 and on cutting edges in chapter 2. A very detailed description of this chapter 3 on the mathematical basis of the methods used can be found in appendix C.

This chapter starts with the section 3.1 and the mathematical foundations required to use the model-based solutions. In the next section 3.2, the basic concepts of machine learning are briefly described. A more detailed introduction of this section can be found in appendix C. In the last section 3.3, the methods used in the thesis, which are based on deep learning, are presented.

3.1 Mathematical Fundamentals

This section starts with the definitions of splines and smoothing splines.

Definition 3.1.1 (Set of Splines): Let $C^q(b)$ describes q-times continuously differentiable functions of a domain b. Splines are defined as piecewise polynomials with required global differentiability. The line of breakage between two polynomials is called knots and the set of all knots will be defined with $\tau = \{\tau_0, \ldots, \tau_l\}$. Let n be the degree of the polynomial then the set of all spline functions are given by

$$\mathbb{S}_n = \{s \in C^{n-1}\left([\tau_0, \tau_l]\right) \big| \, s_{|_{[\tau_i, \tau_{i+1})}} \in \mathbb{P}_n, 0 \leqslant i \leqslant l-1\}, \, n \geqslant 1. \tag{3.1}$$

In order to determine all coefficients of the splines uniquely, additional $q-1$ boundary values are necessary. [42]

The definition of $\mathbb{P}_n$ can be found in C.1. The functions of set $\mathbb{S}_n$ have a disadvantage with noisy data. The nodes $\tau = \{\tau_0, \ldots, \tau_l\}$ are interpolated here. However, if a spline method that suppresses noise is required, smoothing splines can be used.

Definition 3.1.2 (Smoothing Spline): Let $f \in \mathbb{S}_3$ be a function of the defined spline set. Furthermore, a series of observations is given by the set $\{x_i, y_i \big| i = 0, \ldots, n-1\}$. The solution $\hat{f}$ to the optimization problem

$$\min g(f, \lambda) = \sum_{i=0}^{n-1} \left(y_i - \hat{f}(x_i)\right)^2 + \lambda \int \frac{\partial^2 f(x)}{\partial x^2}^2, \quad \lambda \geqslant 0 \tag{3.2}$$

provides the desired Smoothing Spline parameter.[43]

The Mean Square Error is used to evaluate individual deviations or often as an evaluation in the machine learning process as a so-called loss function. The mean square error can be defined with definition 3.1.3.

Definition 3.1.3 (Mean squared error): The mean squared error MSE is defined for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ by

$$MSE = \frac{1}{n}\langle \mathbf{x} - \mathbf{y}, \mathbf{x} - \mathbf{y} \rangle. \tag{3.3}$$

When probability density functions are used in this thesis, they are either normally or uniformly distributed.

Example of a probability density function, the normal distribution The normal or Gaussian distribution is probably the most commonly used distribution in science. A continuous random variable $X : \Omega \to \Omega'$ corresponds to a normal distribution with the mean μ and the variance σ^2 if it has the following probability density function

$$p(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{3.4}$$

Example of a probability density function, the uniform distribution A continuous random variable $X : \Omega \to \Omega'$ has a uniform distribution with two boundaries $a, b \in \mathbb{R}$ and $a \neq b$ if it has the following probability density function

$$p^{(\text{unif})}(x \mid a, b) = \begin{cases} \frac{1}{b-a} & \text{for } a \leqslant x \leqslant b, \\ 0 & \text{else} \end{cases}. \tag{3.5}$$

A description of what exactly probability density functions are can be found in the appendix section C.3.

3.1.1 Signal Processing

Signal processing is generally used in the natural sciences and engineering to describe a process that extracts information from simulated or measured data. This very general description of signal processing has a very wide range of applications. An example from the field of medicine is the recording of the electrical activities of all heart muscle fibres with an electrocardiograph. The electrocardiograph provides information about the heart function and possible disturbances. Signal processing also plays a major role in technology. This section presents the preliminary work for the methods that do not require a machine learning approach, but on the other hand are needed for data generation using machine learning methods. The section 3.1.1 of signal processing starts with convolution and cross-correlation, then defines the Fourier series and the Fourier transform as well as the Radon transform. It also describes filtering methods, in particular polynomial regressions over a set of values and the resulting robust Gaussian filter.

Convolution and Cross Correlation

Convolution is a mathematical method of combining two signals into a third signal. It is one of the most important technique in digital signal processing. With the impulse decomposition strategy, systems are described by a signal called the impulse response. Convolution is important because it combines the three signals of interest: the input signal, the output signal and the impulse response. [44] A definition of convolution can be given as follows

Definition 3.1.4 (Convolution): Let $x, w\colon \mathbb{R}^n \to \mathbb{C}$ be two functions with $f, g \in L^1(\mathbb{R}^n)$ [1]. The convolution is defined by [45]

$$\big(x(\mathbf{a}) * w(\mathbf{a})\big)\,(\mathbf{t}) =: (x * w)\,(\mathbf{t}) = \int_{-\infty}^{\infty} x(\mathbf{a})w(\mathbf{t}-\mathbf{a})\,d\mathbf{a}. \tag{3.6}$$

In machine learning the first entry, in definition 3.1.4 the variable $x(\mathbf{a})$, is called **input** and the second argument $w(\mathbf{a})$ is called **kernel**. Often the measurements are not known at all times. Consequently, the use of an integral is not possible. Instead, the integral is approximated by a sum. In machine learning, the input and the kernel are multidimensional arrays, more precisely tensors and not vectors, as noted in definition 3.1.4. The convolution in general is commutative. However, this property is unimportant in machine learning algorithms. Therefore, the related methods of cross correlation is often used.

Definition 3.1.5 (Cross-correlation): Let $x, w\colon \mathbb{R}^n \to \mathbb{C}$ be functions with $x, w \in L^1(\mathbb{R}^n)$. The cross-correlation is defined by [46]

$$(x \star w)\,(\mathbf{t}) := \big(\bar{x}(-\mathbf{a}) * w(\mathbf{a})\big)\,(\mathbf{t}) = \int_{-\infty}^{\infty} \bar{x}(\mathbf{a})w(\mathbf{t}+\mathbf{a})\,d\mathbf{a}, \tag{3.7}$$

with $\bar{x}$ as conjugated complex function of x.

---

[1] $L^p(D)$ is the space that contains all p times integrable functions on one set D. The space is named after Henri Léon Lebesque.

Note that in machine learning, the term convolution is still used, even if cross-correlation is applied and implemented. As this language is commonly spoken, it is also used in this thesis.

Transformations

In signal processing, a distinction is made between analogue and digital signal processing, whereby digital signal processing is becoming more and more important. The advantages are obvious: digital signal processing has a lower temperature influence and post-processing and storage are easier. To name just two advantages. Nevertheless, analogue signal processing and analogue theories form the basis of digital techniques. For these reasons, analogue theory is described here. We start with the Fourier series and the Fourier transform.

Periodic functions and signals can be represented as superposition of harmonic oscillations. Their frequencies must be integer multiples of the fundamental frequency of the periodic signal. The method by which the corresponding components are summed up is the Fourier series. The coefficients of this sum result in a line spectrum. There it can be seen which frequency components make up the time signal [47].

Definition 3.1.6 (Fourier-Series): Let $f\colon \mathbb{R} \to \mathbb{C},\ x \mapsto f(x)$ be a T-periodic function[2] which fulfills the Dichelet condition[3]. This function f can then be represented by the Fourier series

$$\mathcal{F}[f](x) := \sum_{k=-\infty}^{\infty} c_k e^{ikx}. \tag{3.8}$$

The so-called Fourier-coefficients can be estimated by

$$c_k := \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-ikx}\, dx, \quad \text{with } k \in \mathbb{Z}. \tag{3.9}$$

---

[2] $f(x) = f(x+T), \forall x \in \mathbb{R}$.

[3] A T-periodic function fulfills the Dichelet condition if f is bounded and f and $\frac{df}{dx}$ has finite many discontinuities.

The Fourier series cannot be applied to non-periodic functions. The Fourier transformation is therefore required. The Fourier series is transformed into an integral representation. The discrete line spectrum of the Fourier series results in a continuous spectrum called the Fourier spectrum. With the decomposition into the amplitude and phase response, this transformation provides the prerequisite for a frequency analysis of any signals [47].

Definition 3.1.7 (Fourier-Transformation): An integral transformation, defined by

$$\mathcal{F}\{f(x)\}(\omega) = \int_{-\infty}^{\infty} f(x)e^{-i\omega x}\,dx =: \mathcal{F}(\omega) \tag{3.10}$$

is called Fourier-Transformation. The inverse Fourier-Transformation is given with

$$\mathcal{F}^{-1}\{\mathcal{F}\}(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \mathcal{F}(\omega)e^{i\omega x}\,d\omega = f(x). \tag{3.11}$$

The function $f\colon \mathbb{R} \to \mathbb{C},\ x \mapsto f(x)$ must be integrable for this purpose, more accurate $f \in L^1(\mathbb{R})$.

The introduction of the next mathematical transformation is inspired by [48]. As an example, see the image 3.1 on the left, which shows a noisy image with a white straight line. This could e.g. be a noisy measurement of a surface with a deeper scratch. The aim now is to clearly identify the scratch and give information about the depth or shape of the scratch based on the position. The radon transformation is able to locate these lines clearly in the image domain of the transformation, cf. figure 3.1 right. Thus the determination of these scratches can be reduced by the determination of peeks in the radon domain. A feasible way to indicate and compare peaks is to indicate a threshold value. Alternative solutions are e.g. algorithms for edge detection or using cross-correlation, see section 4.1.1. Nevertheless, there are many ways to define the radon transformation. Here a definition with the Dirac delta function[4] is used.

---

[4] The Dirac delta function is defined by $\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases}$ is used. The Dirac delta function also satisfy the identity $\int_{-\infty}^{\infty} \delta(x)\,dx = 1$.

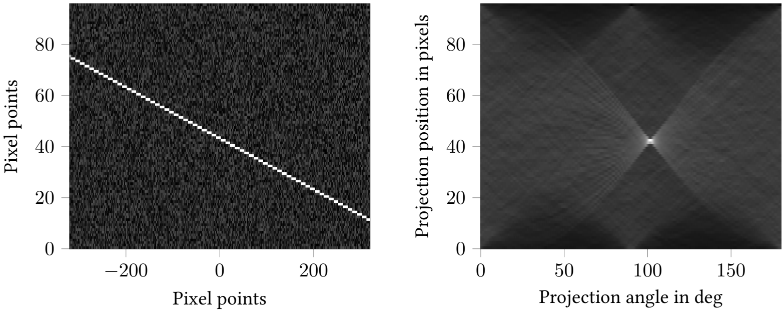


Figure 3.1: The inserted straight line has two weights w_0 and w_1. Both weights can easily be determined from the image domain of the radon transformation using the maximum point $\left(\alpha \mid P\right)$. The slope can be estimated with $w_1 = 1/\tan(\alpha)$ and the shift of the straight line at 0 with $w_0 = P$. In the example shown, the weights $w_0 = 44$ and $w_1 = -0.194$ were calculated. The true values are $w_0 = 44$ and $w_1 = -0.2$.

Definition 3.1.8 (Radon-Transformation): Let $f \in L^1\left(\mathbb{R}^2\right)$ be a scalar field and above a finite radius ρ equal zero. The radon-transform is defined by [49]

$$\mathcal{R}(\rho, \Theta) = \int_{-\infty}^{\infty} f(\mathbf{x})\delta\left(\langle \mathbf{x}, \mathbf{n}_\Theta \rangle - \rho\right)\, d\mathbf{x}, \quad \text{with } \mathbf{n}_\Theta = \begin{pmatrix} \cos(\Theta) & \sin(\Theta) \end{pmatrix}^T. \tag{3.12}$$

The realisation of the radon inverse can be done in several ways. A very common way is the so-called filtered back projection and can be found in C.3.

Filter Operations

There is always a reason why measurements are performed. These measurement data contains some important information. Suppose there are measurement data y_i with $i = 0, \dots, n-1$ and each measurement has a noise term ε_i. In the example, a function $f(x)$ is searched, which is attempted to be determined by measuring $f(x_i)$. However, this can “only” be estimated approximately due to the noise influences. By suppressing noise, the desired function $f(x)$ can be better approximated. It is now assumed that the measurements $\left(x_i \mid y_i\right)$ describe positions along a path. The measurement data can therefore be noted by

$$y_i = f(x_i) + \varepsilon_i. \tag{3.13}$$

Here $f(x)$ describes the desired function. In the simplest case, a digital filter replaces each measured value y_i with a linear combination of itself and the nearby points, i.e.

$$y_i^{(filter)} = \sum_{n=-n_l}^{n_r} c_n y_{i+n}. \tag{3.14}$$

If now $c_n = \frac{1}{n_L+n_R+1}$ and $n_L = n_R$ is selected, a moving average filter is realized. However, this is trivially a problem if, e.g., a local maxima occur. A filter is desired that determines the so-called filter coefficients c_n in a different way. One very famous approach is the Savitzky-Golay filter [50]. It describes the filter coefficients not through a constant window, but by a polynomial approach[5]. Typically, second-order or fourth-order polynomials are used. A table on how to select the filter coefficients can be found in [51]. J. Seewig modified this approach to be phase-true and developed a robust filtering technique by solving the optimisation problem [52]

$$\underset{\beta_k}{\arg\min}\, \rho\left(\mathbf{z} - \mathbf{X}_k \boldsymbol{\beta}_k\right) \cdot \mathbf{s}_k \tag{3.15}$$

for a profile ordinate k where $\boldsymbol{\beta}_k \in \mathbb{R}^n$ describes a parameter vector, $\mathbf{z} \in \mathbb{R}^n$ the measured data, $s_k = \begin{pmatrix} s_{0,k} & s_{1,k} & \cdots & s_{n-1,k} \end{pmatrix}$ with $s_{l,k} = \exp\left(-\alpha(\Delta x_{l,k})^2\right)$ where α is a smoothing value and

$$\mathbf{X}_k = \begin{pmatrix} \mathbf{x}_k^0 & \mathbf{x}_k^1 & \cdots & \mathbf{x}_k^p \end{pmatrix} \tag{3.16}$$

$$\mathbf{x}_k^j = \begin{pmatrix} \left(\Delta x_{0,k}^j\right) & \left(\Delta x_{1,k}^j\right) & \cdots & \left(\Delta x_{n-1,k}^j\right) \end{pmatrix}, \quad \left(\Delta x_{l,k}\right) = \Delta x \cdot (l-k) \tag{3.17}$$

a matrix with equidistant intervals Δx and $j = 0, 1, \ldots, p$. The parameter $\boldsymbol{\beta}_k$ contains the k-th value of the filtered mean line. The other values of $\boldsymbol{\beta}_k$ are the profile derivatives

---

[5] If a zero-order polynomial is chosen, the Savitzky-Golay filter is equivalent to a moving average filter, which is why the moving average filter is a subset of the Savitzky-Golay filter.

up to the p-th derivation. Solving this optimization problem for $p = 2$ and

$$\Psi(x) = x \cdot \left(1 - \left(\frac{x}{c}\right)^2\right)^2, \quad \Psi(x) = \frac{\mathrm{d}\rho(x)}{\mathrm{d}x} \tag{3.18}$$

leads to an robust gaussian regression filter which is in its current form an ISO standard.

3.1.2 Optimization

Optimisation methods are not only important in the application of machine learning, but are also used in this thesis for the generation of data. Here, the Karush-Kuhn-Tucker theorem is necessary, which is used for an optimisation problem with constraints. In this section, we briefly define the optimisation problem and then present an algorithm for the gradient descent method used in machine learning approaches to find minima, before discussing the Karush-Kuhn-Tucker theorem. Further basic definitions can be found in section C.4. Many other parts in this section 3.1.2 are motivated from [53].

The optimal problem can generally be defined as

$$(P): \quad \min f(x) \quad \text{s.t} \quad g_i(x) \leqslant 0, i \in I, \quad h_j(x) = 0, j \in J \tag{3.19}$$

with continuous scalar fields $f, g_i, h_j : \mathbb{R}^N \to \mathbb{R}$ for $i \in I$ and $j \in J$. The index sets I and J are finite and possibly empty. These are defined by

$$I = \{1, \ldots, p\} \text{ and } J = \{1, \ldots, q\}, \quad p, q \in \mathbb{N} \text{ with } q < n. \tag{3.20}$$

If the optimizing target function has no side conditions, the sets I and J are empty. This case will be considered first. If a minimum is now searched, a procedure frequently used in this thesis is the gradient descent method. It is an iterative method which uses a negative gradient. A pseudocode can be found in algorithm 3.1.

Algorithm 3.1: Gradient descent method

```
Input: Function $f \in C^1(\mathbb{R}^n)$ for solving (P), see eq. 3.19
Output: $\mathbf{x}^{(cri)}$ if the algorithm terminates

Set $\mathbf{x}_0$, tolerance $\varepsilon$, $k = 0$
while $\langle \nabla f(\mathbf{x}_k), \nabla f(\mathbf{x}_k) \rangle^{\frac{1}{2}} > \varepsilon$ do
    Set $d_k = -\nabla f(\mathbf{x}_k)$
    Choose step size $t_k$
    Set $x_{k+1} = x_k + t_k d_k$
    $k \leftarrow k + 1$
end
Set $\mathbf{x}^{(cri)} = \mathbf{x}_k$
```

The step size t_k is also fixed in many applications and is called **learning rate** in machine learning. When using fixed step or learning rates, convergence can often not be achieved. Therefore, an adaptive method such as **Adaptive Moment Estimation** (ADAM) is desirable. It is an extension of the stochastic gradient descent method, which in turn is based on the gradient descent. This method is described in detail within section C.6.2. It is called stochastic since the method uses randomly selected samples to evaluate the gradients [54]. ADAM only requires first-order gradients with little memory requirement [55]. The method computes individual adaptive learning rates for different parameters by estimating the first and second moment of the gradients [55]. No pseudo code is noted considering that algorithm 3.1 gives a good overview of the methodology. If the reader is interested, the literature [55] is highly recommended.

Now assume that the index sets I and J from equation 3.20 are not empty and have constraints to solve the optimization problem (P) in equation 3.19. The generalizations of first- and second-order optimality conditions are derived from the unrestricted to the restricted case. As a generalization of the first-order condition $\nabla f(\mathbf{x}) = 0$ from the unrestricted case, one essentially obtains the Karush-Kuhn-Tucker conditions. The Karush-Kuhn-Tucker conditions are also sufficient, not only required, for optimality in convex optimization problems. To note a more elegant Karush-Kuhn-Tucker condition, the active index set is defined. The following definition applies.

Definition 3.1.9 (Active Index Set): Given are the restrictions from equation 3.19. A index set is called active if

$$I^{(act)}(\mathbf{x}) = \{i \in I \mid g_i(\mathbf{x}) = 0\}. \tag{3.21}$$

It can be shown that the consideration of active restrictions is sufficient for the local description of a critical point $\mathbf{x}^{(cri)}$. Before the Karush-Kuhn-Tucker condition is noted, a set of all conditions from equation 3.19 is defined with

$$M = \{\mathbf{x} \in \mathbb{R}^n \mid g_i(\mathbf{x}) \leqslant 0, h_j(\mathbf{x}) = 0\}. \tag{3.22}$$

Theorem 3.1.1 (Karush-Kuhn-Tucker): Let $\mathbf{x}^{\mathbf{(opt)}}$ be a local minimum point of (P) with the scalar fields $f, g_i, h_j \in C^1(\mathbb{R}^n)$ and $i \in I^{(act)}(\mathbf{x}^{(opt)})$, $j \in J$. For $\mathbf{x}^{\mathbf{(opt)}}$ the Mangasarian-Fromowitz condition[6] is also valid. Then multipliers $\lambda_i \geqslant 0, i \in I^{(act)}(\mathbf{x}^{(opt)}), \mu \in \mathbb{R}, j \in J$ exists, with

$$\nabla f(\mathbf{x}^{(opt)}) + \sum_{i \in I^{(act)}(\mathbf{x}^{(opt)})} \lambda_i \nabla g_i(\mathbf{x}^{(opt)}) + \sum_{j \in J} \mu_j \nabla h_j(\mathbf{x}^{(opt)}) = 0. \tag{3.23}$$

At this point it should be noted that Karush-Kuhn-Tucker's theorem only describes the necessary condition. It is only sufficient for convex conditions. Similar to the unrestricted case, a necessary property can be realized with an Hessian matrix. For more information, the following literature is recommended [53].

3.2 Basic Terms Machine Learning

This section defines the terms training set, validation set and test set and then describes overfitting and underfitting. For a detailed introduction, see appendix C.5. Many parts from this section 3.2 are driven by [56]

---

[6] The Mangasarian-Fromowitz condition is fulfilled if for $\mathbf{x} \in M$ the vectors $\nabla h_j(\mathbf{x})$ with $j \in J$ are linear independent and $\exists \mathbf{d} \in \mathbb{R}^n : \langle \nabla g_i(\mathbf{x}), \mathbf{d} \rangle < 0$ and $\langle \nabla h_j(\mathbf{x}), \mathbf{d} \rangle = 0$ for $i \in I^{(act)}(\mathbf{x})$ and $j \in J$.

3.2.1 Training Set, Validation Set and Test Set

The particular challenge of machine learning is to deliver good results for new and unknown data. To ensure this, the data set is divided into three parts: **training set**, **validation set** and **test set**. A possible workflow can be seen in figure 3.2. There, the dataset was split into the three differently named parts. After the model has been trained on the training set, it is evaluated on the validation set. If the results of the validation set are satisfactory, this model is used and confirmed on the test set. If the results of the validation set are not satisfactory, the model must be adjusted. If the data set is large, splitting it into three subsets is not problematic. The test set is then large enough and a statement can be made about the quality of the algorithm used. If this is not the case, there are alternative methods to solve this problem. The best known is cross-validation. In this technique, non-overlapping and repeated sentences are randomly removed from the training set and applied to the test set.

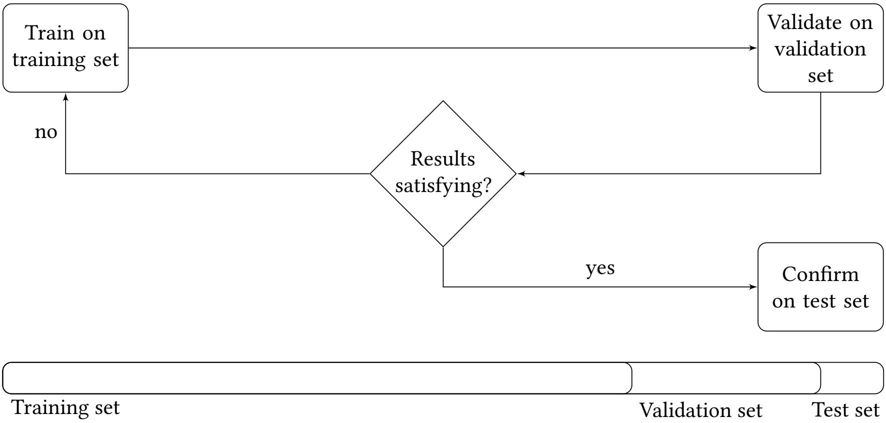


Figure 3.2: Division of the data set into three smaller sets. The data is trained in the training set, validated in the validation set and applied to the test set if the result of validation is satisfactory. If the results of the validation set are not satisfying, the model must be adapted and modified. The bar below indicates the relative proportions into which the data set should be divided.

3.2.2 Overfitting and Underfitting

This section 3.2.2 describes and motivates overfitting and underfitting in machine learning. For this purpose the input data $\mathbf{x}$ and a model function with

$$y^{(model)} = w_0 + w_1 x_1 + \ldots + w_{n-1} x_{n-1} = \mathbf{w}^T \mathbf{x} \tag{3.24}$$

are given. $\mathbf{w} \in \mathbb{R}^n$ is called **weights** and the first element w_0 is called **bias**. The weights $\mathbf{w}$ characterize the behavior of the model function. So if feature x_i is of great importance, w_i will have a large magnitude. Nevertheless, higher order model functions are also allowed, so that $x^2, x^3, \ldots, x^{n-1}$ are new feature to be considered. The model function can be notated using the Hadamard product with

$$y^{(model)} = \mathbf{w}^T \mathbf{x} + \mathbf{w}^T (\mathbf{x} \circ \mathbf{x}) + \ldots . \tag{3.25}$$

Since our task (T) is clearly defined by predicting y using $y^{(model)}$, a suitable performance measurement (P) is required. A good performance measure for this model can be the mean square error MSE as a cost function from the definition 3.1.3. As in figure 3.2, the input data $\mathbf{X} \in \mathbf{R}^{n \times m}$ should be separated in three subsets $\mathbf{X}^{(train)}$, $\mathbf{X}^{(val)}$ and $\mathbf{X}^{(test)}$. However, testing for $\mathbf{X}^{(train)}$ will lead to a small **training error** $MSE^{(train)}$, but actually a small **test error** $MSE^{(test)}$ is of interest, since the machine learning system should be able to handle unknown data. As weights are determined for the training set, weights are learned in such a way that $MSE^{(test)} \geqslant MSE^{(train)}$ is almost everywhere true. The factors that determine how well a machine learning algorithm generally works can be assessed with $MSE^{(train)}$ and $MSE^{(test)}$. This is by its ability to [56]:

1. Make the training error small.
2. Make the gap between training and test error small.

These factors reflect the key challenges: **overfitting** and **underfitting**. Overfitting creates a model that matches the training data so closely that the model cannot make correct predictions given new data [57]. Underfitting results in a model with poor predictive capacity because the model has not captured the complexity of the training data [57]. Changing the capacity can control whether a model is more likely to overfit or underfit the data [56]. One way to control capacity is through **hypothesis space**. This is the set of all admissible functions that may be selected as a solution [56]. Most machine learning algorithms have several setting options that affect capacity. These settings are called **hyperparameters**. A corresponding example in which the capacity is changed can be seen in Figure 3.3.

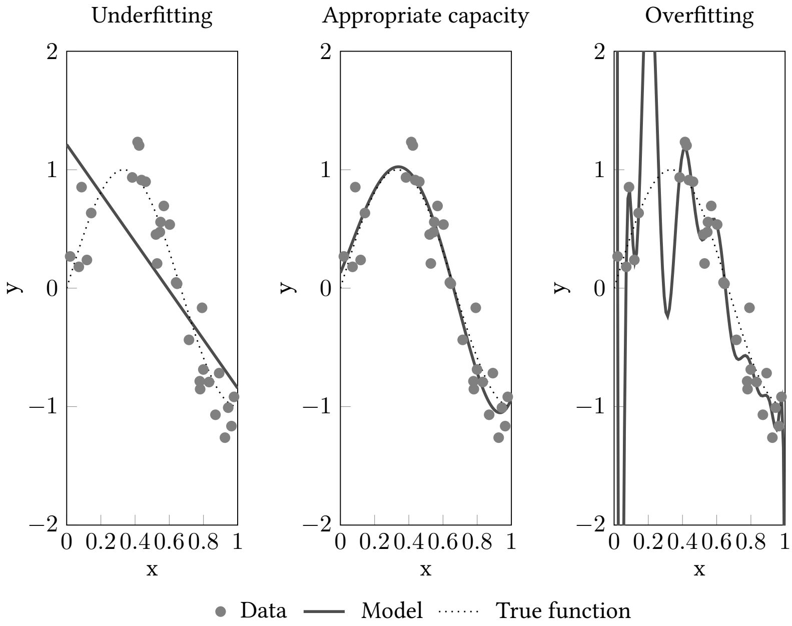


Figure 3.3: The data set is generated by noise from an underlying true function. Three different model functions with changed capacity have been generated. The model function used in the left figure cannot reflect the complexity, while the right figure shows overfitting. The figure in the middle shows the choice of a well selected capacity. [58] [56]

3.3 Deep Learning und Autoencoder

All solutions for the described tasks of micro lead and cutting edge reconstruction, which are solved with machine learning approaches, are based on the fundamentals of neural networks. Due to the high number of features, architectures with neural networks and deep learning were used. A detailed description of neural networks can be found in Appendix C.6. A multi-layer perceptron, a convolutional neural network and an autoencoder were used to solve the tasks for micro lead determination and cutting edge reconstruction. This section 3.3 briefly discusses each of the deep learning approaches, starting with the multilayer perceptron, followed by convolutional neural networks and autoencoders.

3.3.1 Multi-Layer Perceptron

A feedforward neural network or a so called Multi-Layer Perceptron consists of an input layer, one or more layers with artificial neuron called hidden layers, and a final layer with artificial neuron called the output layer, see figure C.5. Each layer except the output layer contains a bias neuron and is fully connected to the next layer. If an Multi-Layer Perceptron has two or more hidden layers, it is called a deep neural network. [59] A more detailed description of an artificial neuron can be found in appendix C.6.1. If we now consider any layer (L) except the input layer, the output of an artificial neuron of the k-th artificial neuron in layer (L) can be determined by

$$y^{(L),k} = \varphi\left(\langle \mathbf{w}_{(L-1)}^{(L),k}, \mathbf{x}_{(L-1)} \rangle + b^{(L),k}\right), \tag{3.26}$$

where $(L-1)$ is the layer before (L). For the described artificial neuron $y^{(L),k}$, many input values x_i exist, which are multiplied by a weight w_i. This leads to a sum $\langle \mathbf{w}_{(L-1)}^{(L),k}, \mathbf{x}_{(L-1)} \rangle$, where $b^{(L),k}$ is the bias term of neuron k in layer (L) to which an activation function φ is applied.

Activation Function

To keep the section 3.3.1 about activation function short, in most current applications rectified linear unit (ReLU) is used as an activation function. It is slightly faster than other activation functions like the logistic sigmoid activation function or the hyperbolic tangent activation function. In addition, the gradient descent does not rest so much on plateaus corridors, since it is not saturated for large input values [59], [56]. However, it should be mentioned that the activation function exponential linear unit (ELU) outperformed all the ReLU variants in Djork-Arné Clevert et al. experiments [59] [60] and is therefore often used in this thesis. This includes training time and better results on the test set. The graphs of the mentioned activation functions are illustrated in figure 3.4. The corresponding equations are given by

$$\varphi(x)^{sig} = \frac{1}{1+e^{-x}}, \quad \varphi(x)^{tanh} = \tanh(x) \tag{3.27}$$

$$\varphi(x)^{ReLU} = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geqslant 0 \end{cases}, \tag{3.28}$$

$$\varphi(x)^{ELU} = \begin{cases} \alpha\left(e^{x}-1\right) & \text{for } x < 0 \\ x & \text{for } x \geqslant 0 \end{cases}, \text{ with } \alpha \in \mathbb{R}_{>0}. \tag{3.29}$$

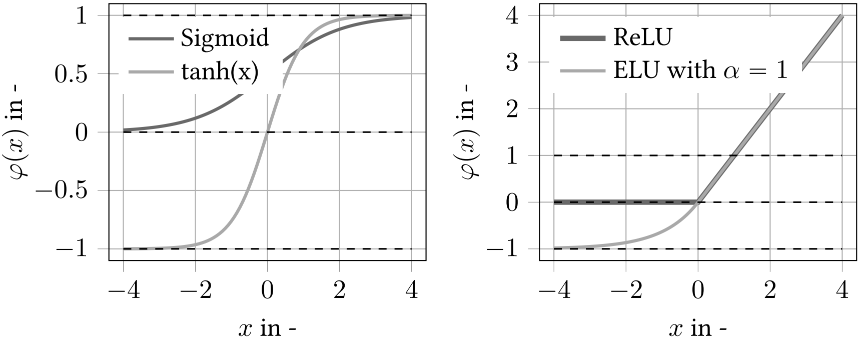


Figure 3.4: The activation functions Sigmoid, Tangens hyperbolicus, rectified linear unit (ReLU) and exponential linear unit (ELU) are shown.

Regularization

In section C.5.2 the importance of regularization in machine learning has already been discussed. As a reminder, a key problem in machine learning is that the data should have a small error not only for the training set but also for an unknown test set [56]. Strategies trying to realize this are called regularization. Indeed, for deep neural networks some new strategies exist besides L_1 and L_2 regularization (see equation C.30). Two further strategies are described, namely **Early Stop** and **Dropout**.

Early Stop

The curve of the training error and the curve of the validation error are often characteristic. While in learning process the error for the training set become smaller and smaller, the validation error increase again. It has a kind of a U-shape. The increase of validation error is a sign for overfitting. Early Stop stops when the error for the validation set has reached its minimum. Typical curves for the training and validation set are shown in Figure 3.5. Even if early stopping works well in practice, performance can be further improved by combining it with other regularization procedures [59]. A very good guideline to use Early Stop in a meaningful way is [61] and [62].

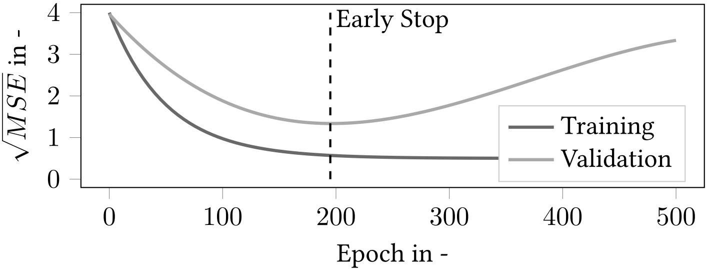


Figure 3.5: Two typical progressions for training error and validation error, evaluated with the performenace measurement $\sqrt{MSE}$, are shown [63]. Early stop is a procedure to avoid overfitting. The epoch with the smallest validation error is selected.

Dropout

The most popular regulatory method for deep neural networks is currently Dropout [59]. Here each neuron has a probability of p being thrown or dropped out during the training step. This also includes the neurons from the input layer, but no neurons from the output layer. In the next step all cards are shuffled again and the ignored neuron can have an influence in the next taining step. The probability p is often set to 50%. After training, no neuron is ignored. G.E. Hinton presented this method in a paper 2012 and Nitish Srivastava et al. showed in another paper the high success of this method [64] [65].

3.4 Convolutional Neural Network

The Convolutional Neural Network (CNN) is a special type of neural network often used for grid-like structures. In contrast to the multilayer perceptron, the input is left unchanged in its structure. For the terms convolutional matrix, convolutional layer and the associated terms convolutional stage, stride, feature map and pooling stage used here, see section C.7.2.

In this section 3.4 we describe the convolutional architectures and autoencoders for CNNs used in this thesis. Further explanatory terms, such as transposed convolution, tying weights, denoising autoencoders and a further list of autoencoders used can be found in appendix C.7.3, C.8, C.8.2, C.8.3 and C.8.4.

3.4.1 Convolution Architectures

A typical CNN architecture stacks multiple convolutional layers, uses a non-linear activation function, applies a pooling layer, performs multiple convolutional layers again, and so on. The size of the current output becomes smaller and smaller, but also deeper and deeper. Typically, the last layer state adds a regular feedforward network. This network consists of some fully connected layers and the last layer gives a prediction. Figure 3.6 illustrates a typical structure for this.

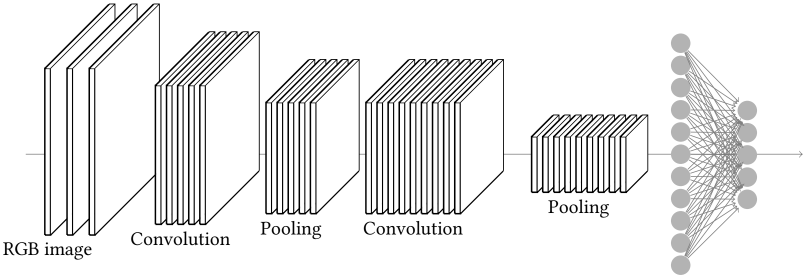


Figure 3.6: A possible CNN structure is illustrated. The input image is reduced to different patterns by the application of several filters and convolution between input and filter. Pooling was used afterwards to store computational load, memory usage and the number of parameters. This is followed by the application of several filters and pooling. At the end a fully connected neural network is considered to obtain a prediction. Template from [66].

It should be noted that many different architectures have been developed over the years. The ILSVRC[7] challenge and the winners give a good overview about successful architectures and are helpful to build up a good understanding for CNN architectures. These include AlexNet (2012) [67], GoogLeNet (2014) [68] and ResNet (2015) [69] just to name some few.

3.4.2 Autoencoder for CNNs

The idea and methodology of autoencoders described in C.8 can also be used for CNNs. The typical stages are used for encoding, but instead of an output layer a decoding part with convolutionally transposed or resizable convolution is used so that the output has the same dimensional size as the input. A possible setup for an autoencoder based on CNN can be seen in figure 3.7. The effective representation of the input is also called **bottle neck** [70].

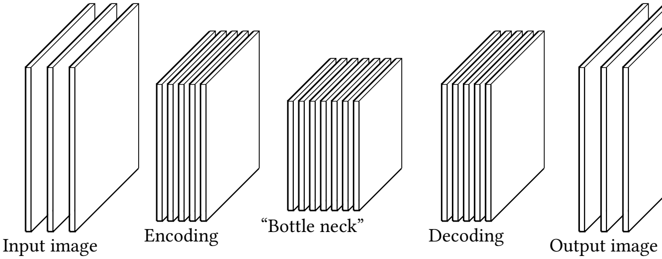


Figure 3.7: In an autoencoder based on CNNs, the input image is firstly encoded as described in section 3.4.1 and brought to an effective representation. The representation is then decoded with convolution transposed or resizing. Ideally, the output is the input or at least a good approximation. Template from [66].

---

[7] short for ImageNet Large Scale Visual Recognition Challenge.

4 Model-based Approach for determining Micro Lead and Reconstruction

This chapter 4 describes the solutions to the problems described in chapter 1 and chapter 2 and the tasks derived from them using model-based assumptions. These described solutions represent the best possible procedures after a status quo analysis. A possible exception is the determination of the micro lead angle Ψ. This shows very good results in simulations, but in contrast to [2], it has not been extensively validated on real samples.

Section 4.1 describes a cross-correlation based approach to determine the texture direction of ground surfaces. The theory used was developed by J. Seewig and is presented in this form for the first time in this thesis in section 4.1.1. The corresponding validation on real samples and simulated surfaces and the comparison with an already validated methodology has been carried out for the approach from section 4.1.1 first in the context of this thesis and is described in section 4.1.2.

Section 4.2 presents a fitting based on nonlinear least squares, where the theory is described in section 4.2.1 and derived from [71]. In section 4.2.2 this approach is tested for the first time for its applicability for micro lead measurements.

The last section 4.3 of this chapter describes the reconstruction of cutting edges using a robust Gaussian filter. The described theory and the approach of this solution were developed by the author and successfully tested not only on simulated but also on real samples. The first associated section 4.3.1 deals with the generation of simulated cutting edges, since due to legal requirements no real samples is used in this thesis. The section

4.3.2 describes the methodology for generating ideal cutting edges based on the robust Gaussian filter. The last section 4.3.3 validates the presented approach on different types of cutting edges.

4.1 Determination of Texture Direction

As known from section 1, equation 1.1 and figure 1.2, the texture direction and the circumferential direction must be known in order to unambiguously determine the micro lead angle Ψ. Here a method is presented that is based on area spectral power spectral density [1] and can achieve a resolution of less than one minute of arc. This approach was developed by J. Seewig and is described in the section 4.1.1. Furthermore, the approach has already been tested and validated on real surfaces. In comparison, the methodology of P. Arnecke [6], which is based on the Radon transform and according to [6] and [2] can achieve a resolution of the texture direction for microlead evaluation below one arc minute, serves as a reference. Both methods by J. Seewig and Philipp Arnecke are compared in section 4.1.2 for simulated and real surfaces. The idea and the description of the method from section 4.1.1 are taken from an internal document by J. Seewig.

4.1.1 Cross-correlation based Texture Orientation Detection

One method to calculate texture orientation of an areal-topographic surface measurement is based on the cross correlation between harmonic waves and the surface. To benefit from an easy description by the Fourier transform the following explanation assumes a sinusoidal topography

$$z(\mathbf{x}, \mathbf{s}) = e^{i\left(\frac{2\pi}{\lambda}\langle \mathbf{x}, \mathbf{s}\rangle\right)}, \tag{4.1}$$

where $\mathbf{x}$ describes the position and $s = \begin{pmatrix}\cos(\varphi) & \sin(\varphi)\end{pmatrix}^T$ the wave vector with an arbitrary wavelength λ and an orientation of the wave front φ. The topography is oriented in a coordinate system with coordinates $x = \begin{pmatrix}x & y\end{pmatrix}^T$. The coordinate systems origin is the topographies lateral centre.

---

[1] which is closely related to circular correlation

Structure detection using a two-dimensional discrete Fourier transform $z(\mathrm{x}, \mathrm{s})$ be a sinusoidal given by equation 4.1. The wave front φ is used in a discrete grid with M steps in x and N in y direction. This grid is defined by

$$x_m = m \cdot \Delta x - \frac{X}{2}, \quad y_n = n \cdot \Delta y - \frac{Y}{2} \tag{4.2}$$

with $m = 0, 1, \ldots, M-1, n = 0, 1, \ldots, N-1$ and $X = M \cdot \Delta x$ and $Y = N \cdot \Delta y$. Inserting equation 4.2 into equation 4.1 and applying a two-dimensional discrete Fourier transformation leads to

$$\begin{aligned} Z_{k,l} = {} & \underbrace{\mathrm{e}^{-i \cdot 2\pi \cdot \left(\frac{X}{2} \cdot \frac{\cos(\varphi)}{\lambda} + \frac{Y}{2} \cdot \frac{\cos(\varphi)}{\lambda} \right)}}_{=:\Phi_z} \cdot \\ & \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \mathrm{e}^{i \cdot 2\pi \left[m \cdot \left(\Delta x \frac{\cos(\varphi)}{\lambda} - \frac{k}{M} \right) + n \cdot \left(\Delta y \frac{\sin(\varphi)}{\lambda} - \frac{l}{M} \right) \right]} \end{aligned} \tag{4.3}$$

with $k = 0, 1, \ldots, M-1 \wedge l = 0, 1, \ldots, N-1$. Now a special case for the wave front φ is considered. Only wavelengths λ with

$$^{X}\!/_{q} = {}^{\lambda}\!/_{\cos(\varphi)} \wedge {}^{Y}\!/_{r} = {}^{\lambda}\!/_{\sin(\varphi)}, \quad q, r \in \mathbb{N} \tag{4.4}$$

are allowed. Therefore, the projection of the wavefront in x- and y direction are multiple integer of the grid size X and Y. Then, equation 4.3 reduces to

$$Z_{k,l} = \Theta_z \cdot N \cdot M, \quad \text{for } k = q \wedge l = r, \tag{4.5}$$

and $Z_{k,l} = 0$ for all other cases, which is a clear peak in the spectrum. Thus, the orientation of the wave front φ can be estimated with

$$\varphi = \arctan \left({}^{X \cdot r}\!/_{Y \cdot q} \right). \tag{4.6}$$

Structure detection using a rotation matrix and an one-dimensional Fourier transform To achieve a one-dimensional Fourier transform approach, the texture orientation of $z(x, y)$ is examined in a new coordinate system (x', y') by rotating the original coordinate system (x, y) by an angle α.

$$\begin{pmatrix} x' & y' \end{pmatrix}^T = \begin{pmatrix} x \cdot \cos(\alpha) + y \cdot \sin(\alpha) & -x \cdot \sin(\alpha) + y \cdot \cos(\alpha) \end{pmatrix}^T. \tag{4.7}$$

Inserting equation 4.7 in equation 4.1 leads to

$$z(x', y') = \mathrm{e}^{i \cdot \frac{2\pi}{\lambda} \cdot \left(x \cdot \cos(\alpha - \varphi)\right) - y \cdot \sin(\alpha - \varphi)}. \tag{4.8}$$

Applying the two-dimensional spatial continuous Fourier transform to the new coordinate system (x', y') and since the measured data consists of discrete points, inserting the grid for equation 4.2 yields to an expression

$$\begin{aligned} &Z(\omega_x, \omega_y, \alpha, \varphi) = \\ &\underbrace{\mathrm{e}^{-i \cdot \frac{\pi}{\lambda} \cdot \left(X \cdot \cos(\varphi) + Y \cdot \sin(\varphi)\right)}}_{=:\Phi_Z} \cdot \underbrace{\mathrm{e}^{\frac{i}{2} \cdot \left[X \cdot \left(\omega_x \cos(\alpha) - \omega_y \sin(\alpha)\right) + Y \cdot \left(\omega_x \sin(\alpha) + \omega_y \cos(\alpha)\right)\right]}}_{=:\Phi_F} \\ &\cdot \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \{\mathrm{e}^{i \cdot 2\pi \cdot m \cdot \Delta x \left[\cos(\varphi)/\lambda - \left(\omega_x \cdot \cos(\alpha) - \omega_y \cdot \sin(\alpha)\right)/2\pi\right]} \\ &\mathrm{e}^{i \cdot 2\pi \cdot n \cdot \Delta y \left[\sin(\varphi)/\lambda - \left(\omega_x \cdot \sin(\alpha) + \omega_y \cdot \cos(\alpha)\right)/2\pi\right]}\}. \end{aligned} \tag{4.9}$$

The angle φ is estimated similar to section 4.1.1. The condition, comparable to equation 4.4, is exactly fulfilled for $\omega_y = 0$ and $\alpha = \varphi$. Therefor the expression $Z(\omega_x, \omega_y, \alpha, \varphi)$ can be reduced to a one-dimensional equation along the x' direction

$$\begin{aligned} Z(\omega_x, \varphi) = \Phi_Z \cdot \Phi_F \cdot \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} &\mathrm{e}^{i \cdot 2\pi \cdot \left[m \cdot \Delta x \cdot \cos(\varphi)/\lambda + n \cdot \Delta y \sin(\varphi)/\lambda\right]} \\ &\cdot \mathrm{e}^{-i \cdot \omega_x \cdot \left[m \cdot X \cdot \cos(\varphi)/M + n \cdot Y \sin(\varphi)/N\right]}, \end{aligned} \tag{4.10}$$

where $\Phi_Z(\lambda, \varphi)$ and $\Psi_F(\omega_x, \varphi)$ are independent of the individual sample points. For finding the frequency ω_x a sampling with

$$\omega_x = 2\pi \cdot \frac{k_{x\prime}}{lt}, \quad lt = \frac{X \cdot Y}{\sqrt{\left(X \cdot \sin(\varphi)\right)^2 + \left(Y \cdot \cos(\varphi)\right)^2}} \tag{4.11}$$

is require. To avoid aliasing, k_x is limited. Inserting equation 4.11 in equation 4.10 leads to an expression $Z(k_x)$. Texture orientation is now found as a global maximum from a range of rotation angles each summed up over a range of frequencies

$$\omega_{cc}(\varphi) = \sum_{p=0}^{P-1} Z(k_{x,p}, \varphi)^2 \tag{4.12}$$

with $k_{x,\mathrm{min}} \leqslant p \leqslant k_{x,\mathrm{max}}$ for $p = 0, 1, \ldots, P-1$ and $\Theta^{\mathrm{t}} = \arg\max\big(\omega_{cc}(\varphi)\big)$. This procedure reduces a global detection problem to a simple one-dimensional peak detection.

4.1.2 Results and Discussion

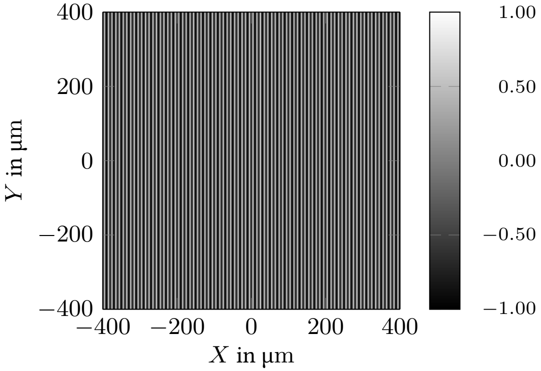
In order to verify the feasibility of the methods for micro lead evaluation, simulated and measured topographies are presented together with the pre-processing steps necessary for measured data. The texture directions of the surfaces are evaluated with the methods based on the radon transformation of [6] and the cross correlation based method of section 4.1.1. The results are discussed and the similarity between the approaches is investigated.

Preliminary description The pre-processing steps for measured data are followed by a description of the test surfaces before the results for simulated and measured topographies are presented.

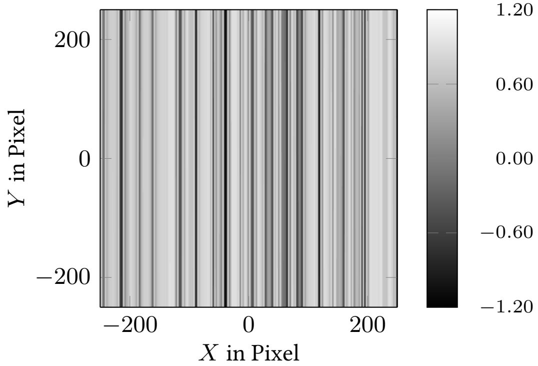
Preprocessing The pre-processing is the same for both methods. A $3\,\mathrm{pixels} \times 3\,\mathrm{pixels}$ median filter is followed by the generation of an S-F surface according to ISO 25178-2 [17]: a Gaussian filter from the ISO 16610-61 standard [17] with an nesting index of $8\,\mathrm{\mu m}$ suppresses short-wavelength structures and then the subtraction of a linearly fitted second-order polynomial serves as an F operation. For more information, see appendix A.

Test surfaces Simulated and measured surfaces are used to verify accuracy, resolution and sensitivity of both methods.

The simulated surfaces consist of two different types. One are purely sinusoidal surfaces with a spatial wavelength $\lambda = 20\,\mu\mathrm{m}$ on a $800\,\mu\mathrm{m} \times 800\,\mu\mathrm{m}$ topography with lateral sampling distances of $1.5625\,\mu\mathrm{m}$. It is defined similar to equation 4.1. Their orientation Θ^{t} ranges from 89.9° to 90.1° in increments of $3.6''$. The topographies have a mean value of zero and an amplitude of 1, which can be considered as $1\,\mu\mathrm{m}$. An exemplary surface is plotted in figure 4.1a. The other type of simulated surface is generated by placing peaks in the Radon domain followed by an inverse Radon transform. A surface has a size of $500\,\mathrm{pixels} \times 500\,\mathrm{pixels}$ and should approximately resemble a ground surface. An example is shown in figure 4.1b.



(a) Sinusoidal surface



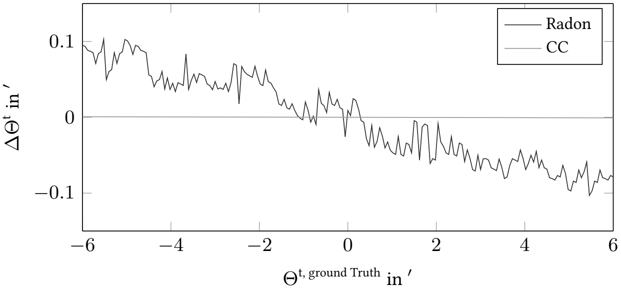
(b) Ground surfaces modeled in the Radon domain

Figure 4.1: Simulated test surfaces with $\phi_{set} = 90^\circ$.

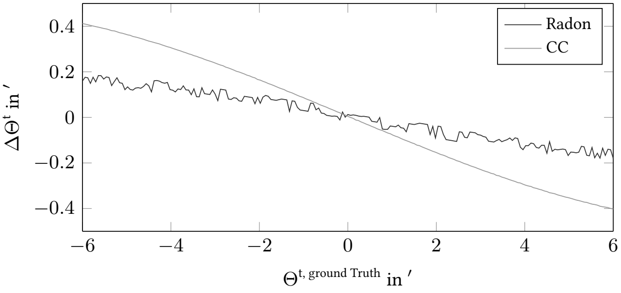
When evaluating the texture direction for both algorithms, the resulting deviation $\Delta\Theta^{\mathrm{t}}$ from the modeled orientation Θ^{t} is calculated with

$$\Delta\Theta^{\mathrm{t}} = \Theta^{\mathrm{t}} - \Theta^{\mathrm{t,\,ground\ Truth}}\,, \tag{4.13}$$

where $\Theta^{\mathrm{t,\,ground\ Truth}}$ describes the true value of the texture direction, the so-called ground truth. The results are shown in figures 4.2a and 4.2b. The deviations in figure 4.2b for the cross-correlation approach will be further analysed. It is obvious that the Radon-based evaluation deviates more on the sinusoidal surfaces, while the opposite is true on the simulated ground surfaces generated by the inverse Radon transformation. Nevertheless, both methods evaluate the texture orientation with small deviations from the ideal sensitivity and are capable of resolving the texture orientation in a very low sub-arc minute range.



(a) $\Delta\Theta^{\mathrm{t}}$ for surfaces from figure 4.1a.



(b) $\Delta\Theta^{\mathrm{t}}$ for surfaces from figure 4.1b.

Figure 4.2: Results from simulated test surfaces from figure 4.1.

Measured ground shaft surfaces A similar study is conducted on measured data. The shaft was rotated around a radial axis in 20 increments of $\approx 16.91'$. One measured surface is plotted in figure 4.3. The obtained Θ^t are plotted in figure 4.4. The data was presented before in [8] and [72]. The surfaces are not related to a reference direction and their absolute values only represent Θ^t and not Ψ (cf. figure 1.2).

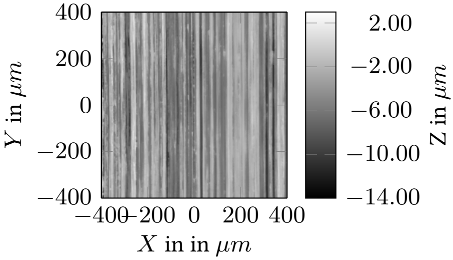


Figure 4.3: Ground shaft surface from measured data

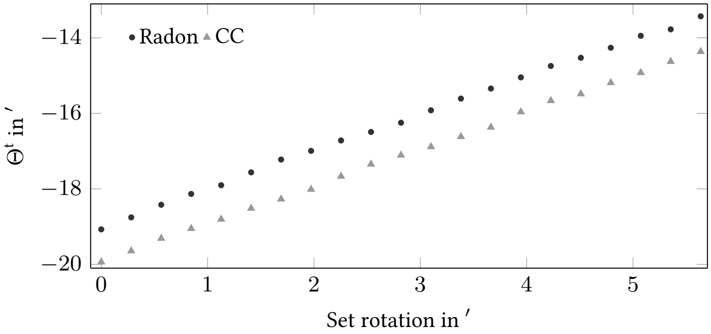


Figure 4.4: Calculated orientation of measured topographies.

Both curves progress nearly identical but deviate in their absolute values by around $1'$.

Measurements around the circumference of a ground shaft are presented as histogram in figure 4.5. The Radon-based method yields a mean of $14.85'$ and a standard deviation of $2.20'$. Cross correlation results in $15.53'$ with a standard deviation of $1.81'$.

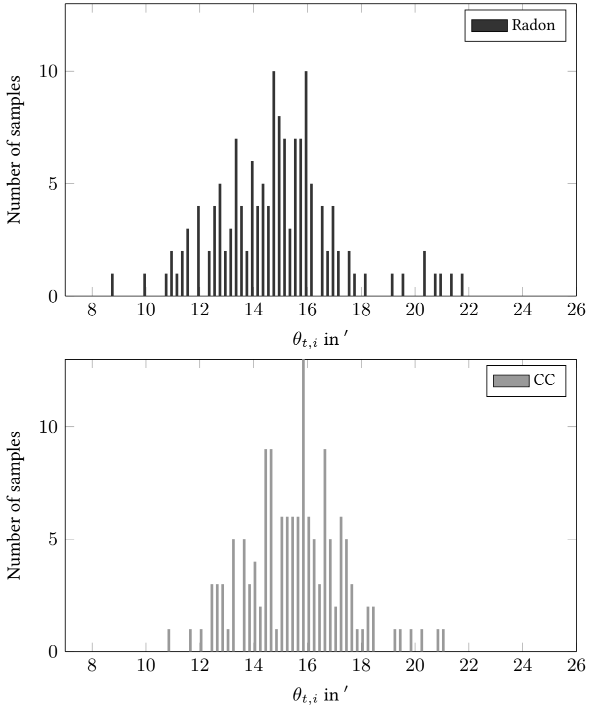


Figure 4.5: Texture orientation of circumferential measurements of ground shaft surfaces (bar width $0.2'$).

Discussion The two methods for evaluating the texture direction Θ^{t} differ to a certain extent. However, the difference is considerably small. It can be seen from figure 4.2 that both procedures offer a low sub-arc minute resolution on both simulated and measured topographies. The cross-correlation algorithm and the sinusoidal surfaces are a perfect match because this evaluation is based on harmonic waves. The Radon-based algorithm differs more but still resolves the simulated texture orientation very well. For ground surfaces simulated with the inverse radon transformation, a higher sensitivity of the cross correlation based method can be seen compared to the radon method from P. Arnecke. But the difference of $\approx\ 0.4'$ at $\pm\ 0.1^\circ$ does not appear to be a concern in common applications. Especially, since both methods produce comparable results on real measurement data in figures 4.4 and 4.5.

The reason for the absolute differences between the methods of about one minute of arc on measured data, but very similar variabilities, is probably due to different structures on the surfaces to which the methods are not equally sensitive. This assumption is supported by figure 4.2, although the methods behave slightly differently on artificial surfaces.

An often neglected but crucial prerequisite in any evaluation of lead and especially micro lead is the agreement on the coordinate system for lead measurement on shafts [1]. Measuring instruments often provide only a vague specification of their coordinate system. By confusing directions, wrong conclusions can be drawn and incorrect conclusions are likely to be made.

Relation between both methods Radon transformation and 2D-Fourier transformation are related by the projection slice theorem [73] and [74]. According to this theorem, the 1D Fourier transform of a section through the radon domain at the angle $\alpha = \phi$ and all ρ (the so-called projection profile) is equal to a central section at $\alpha + \pi/2 = \beta$ by the 2D Fourier transform of a surface. When using the Parseval theorem [75], curves from the squared Radon domain and the squared power spectrum from the 2D Fourier transform contain the same signal energy [76]:

$$\int\limits_{\rho_{min}}^{\rho_{max}} \breve{z}\,(\alpha,\theta)^2\, \mathrm{d}\rho = \int\limits_{\omega_{x,min}}^{\omega_{x,max}} Z(\omega_x,\beta)^2 \quad . \tag{4.14}$$

Regarding the power spectrum and the squared Radon transform it is no surprise that both methods yield essentially the same results, see figure 4.4.

4.2 Determine the Direction of a Shaft for Micro lead

As a starting point the equation 1.1 with $\Psi = \Theta^{\mathrm{t}} - \Theta^{\mathrm{circ}}$ is considered[2]. Using the model-based method described in the section 4.1 it is possible to estimate the texture direction Θ^{t} without interpolation steps and to resolve the texture direction Θ^{t} below one arc minute. What is missing for the determination of the micro lead angle Ψ is the

---

[2] Circumferential direction Θ^{circ}, texture direction Θ^{t} and Ψ micro lead angle.

circumferential direction Θ^{circ} which can be determined with the axis direction Θ^{ax} of the cylindrical shaft, see figure 1.2. The determination of the circumferential direction Θ^{circ} by a cylinder fit has already been tested in [2]. Here, 108 measurements were performed to obtain the desired resolution.

In the first section 4.2.1 a method for fitting a cylinder in point clouds is presented. This method does not use machine learning methods and refers to [71]. The validation of the suitability of the described approach for lead measurements is presented in section 4.2.2. In this section 4.2.2 this method is applied to simulated cylinder points, which describes a simplified micro lead measurement.

4.2.1 Nonlinear Fitting for 3D Cylinder Points

A infinite cylinder is determined unambiguously by its center $\mathbf{c} \in \mathbb{R}^3$, a unit vector $\mathbf{w} \in \mathbb{R}^3$ and a radius $r \in \mathbb{R}_{\geqslant 0}$. If two more unit vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^3$ are introduced, which form an orthonormal base, each point can be uniquely determined by

$$\mathbf{x} = \mathbf{c} + y_0\mathbf{u} + y_1\mathbf{v} + y_2\mathbf{w} = \mathbf{c} + \mathbf{R}\mathbf{y} \tag{4.15}$$

with $\mathbf{R} \in \mathbb{R}^{3\times 3}$ as a rotation matrix and $\mathbf{y} \in \mathbb{R}^3$ containing the row elements $y_0, y_1, y_2 \in \mathbb{R}$. To apply a representation in equation 4.15 for cylinders, the following two conditions

$$\begin{aligned} r^2 &= y_0^2 + y_1^2 \\ &= \left(\mathbf{u}^T\left(\mathbf{w} - \mathbf{c}\right)\right)^2 + \left(\mathbf{v}^T\left(\mathbf{w} - \mathbf{c}\right)\right)^2 \\ &= \left(\mathbf{x} - \mathbf{c}\right)^T \underbrace{\left(\mathbf{I} - \mathbf{w}\mathbf{w}^T\right)}_{=:\mathbf{P}} \left(\mathbf{x} - \mathbf{c}\right)^T \end{aligned} \tag{4.16}$$

$$\|y_2\|_2 = \|\mathbf{w}^T\left(\mathbf{x} - \mathbf{c}\right)\|_2 \leqslant \frac{h}{2} \tag{4.17}$$

must be fulfilled, with $\mathbf{I} = \mathbf{u}\mathbf{u}^T + \mathbf{v}\mathbf{v}^T + \mathbf{w}\mathbf{w}^T$ as unit matrix. The condition 4.17 forces the cylinder to have a finite total height h. To realize the fitting of a cylinder, the radius r, the center $\mathbf{c}$ and the direction of the cylinder $\mathbf{w}$ must be determined. If the measurement data $\mathbf{x}_i, i = 0, \ldots, n-1$ are available, the error function $E\left(r, \mathbf{c}, \mathbf{w}\right)$ can be determined with

$$E\left(r,\mathbf{c},\mathbf{w}\right)=\sum_{i=0}^{n-1}\left(\underbrace{\left(\mathbf{x}_i-\mathbf{c}\right)^T\left(\mathbf{I}-\mathbf{w}\mathbf{w}^T\right)\left(\mathbf{x}_i-\mathbf{c}\right)^T}_{=r_i}-r\right)^2. \tag{4.18}$$

To achieve numerical robustness, the arithmetic mean of the measured data is zero [3]. First the equation for the radius r is introduced, followed by the center $\mathbf{c}$ and the unit direction vector $\mathbf{w}$.

Equation for the radius To optimize the error function theorem C.4.1 is applied and the error function is partially derived to r^2. Therefore with

$$\frac{\partial E\left(r,\mathbf{c},\mathbf{w}\right)}{\partial r^2}\overset{!}{=}0 \tag{4.19}$$

the equation

$$r^2=\frac{1}{n}\sum_{i=0}^{n-1}r_i \tag{4.20}$$

follows. Using this property from equation 4.20, the result is

$$r_i^2-r^2=-2\mathbf{x}_i\mathbf{P}\mathbf{c}+\mathbf{x}_i^T\mathbf{P}\mathbf{x}-\frac{1}{n}\sum_{i=0}^{n-1}\mathbf{x}_i^T\mathbf{P}\mathbf{x}. \tag{4.21}$$

Equation for the center Is theorem C.4.1 applied, then the derivation of the error function $E\left(r,\mathbf{c},\mathbf{w}\right)$ after the center $\mathbf{c}$ leads to the equations

$$\mathbf{0}=-2\cdot\underbrace{\frac{1}{n}\mathbf{P}\left(\sum_{i=0}^{n-1}\mathbf{x}_i\mathbf{x}_i^T\right)\mathbf{P}}_{=:\mathbf{A}}\mathbf{c}+\underbrace{\frac{1}{n}\sum_{i=0}^{n-1}\mathbf{x}_i^T\mathbf{P}\mathbf{x}_i\mathbf{P}\mathbf{x}_i}_{=:\mathbf{B}} \tag{4.22}$$

$$\mathbf{A}\mathbf{c}=\frac{1}{2}\mathbf{B} \tag{4.23}$$

---

[3] If this is not the case, the arithmetic mean value is subtracted from the measurement data and added again after the determination of the desired variables.

where the condition in equation 4.21 was used. Due to the singularity of $\mathbf{P}$, matrix $\mathbf{A}$ cannot be inverted. The linear system of equations has only information about the plane that is perpendicular to the direction $\mathbf{w}$. The system of equations is reduced to two equations with two unknowns. If this idea is continued, the matrix

$$\hat{\mathbf{A}} = \mathbf{S}\mathbf{A}\mathbf{S}^T. \tag{4.24}$$

$\hat{\mathbf{A}}$ can be noted by defining a skew matrix

$$\mathbf{S} = \begin{pmatrix} 0 & -w_2 & w_1 \\ w_2 & 0 & -w_0 \\ -w_1 & w_0 & 0 \end{pmatrix}, \tag{4.25}$$

where $w_i, i = 0, 1, 2$ are the row elements of the direction $\mathbf{w}$. The center $\mathbf{c}$ can now be determined with the solution of

$$\mathbf{c} = \frac{\hat{\mathbf{A}}}{\operatorname{Trace}\left(\hat{\mathbf{A}}\mathbf{A}\right)} \left(\frac{1}{n} \sum_{i=0}^{n-1} \left(\mathbf{x}_i^T \mathbf{P} \mathbf{x}_i \right) \mathbf{x}_i \right). \tag{4.26}$$

Equation 4.26 is valid only by one further assumption. The center point $\mathbf{c}$ could be anywhere along the set

$$\{\lambda \in \mathbb{R} \mid \mathbf{c} = \mathbf{c}^{(\text{Origin})} + \lambda \mathbf{w}\}. \tag{4.27}$$

The calculation of the center point $\mathbf{c}$ from equation 4.26 is independent from the unit direction vector $\mathbf{w}$.

Equation for the direction If the direction $\mathbf{w}\left(\mathbf{s}\right)$ is being parameterized, for example in spherical coordinates, $\mathbf{s}$ is a two-dimensional parameterization variable. Now, if theorem C.4.1 is applied, the derivation of the error function according to the parameterization variables cannot be solved in closed form. One way of determining the direction $\mathbf{w}$ can be done by using the conditions derived from equations 4.21 and 4.26. If these two equations are used in the error function, a convex nonlinear optimization problem with

$$g(\mathbf{w}) = \frac{1}{n}\sum_{i=0}^{n-1}\left(\mathbf{x}_i^T\mathbf{P}\mathbf{x}_i - \frac{1}{n}\sum_{i=0}^{n-1}\mathbf{x}_i^T\mathbf{P}\mathbf{x}_i - 2\mathbf{x}_i^T\frac{1}{\text{Trace}\left(\hat{\mathbf{A}}\mathbf{A}\right)}\left(\frac{\hat{\mathbf{A}}}{n}\sum_{i=0}^{n-1}\left(\mathbf{x}_i^T\mathbf{P}\mathbf{x}_i\right)\mathbf{x}_i\right)\right)^2 \tag{4.28}$$

follows, which can be solved e.g. with the Powell method [77]. David Eberly shows in [71] that the solution of the optimization problem and thus the whole algorithm converges much faster by precomputed summations. A pseudo code without precomputed sums can be viewed in algorithm 4.1. A suitable implementation can be found in [78].

Algorithm 4.1: Least squares fitting of a cylinder data

```
Input: $n$ points $\mathbf{x}_i$ with $i = 0, \ldots, n-1$
Output: radius $r$, center $\mathbf{c}$ and direction $\mathbf{w}$

Preprocess data (arithmetic mean of zero).
Define start point for direction $\mathbf{w}$.
Estimate $\mathbf{w}$ by minimizing $g(\mathbf{w})$ from equation 4.28 with e.g. the Powell method.
Estimate $\mathbf{c}$ using equation 4.26. Add the arithmetic mean value to $\mathbf{c}$.
Estimate $\mathbf{r}$ using equation 4.21.
```

4.2.2 Application and Validation for Simulated 3D Cylinder Points

Various cylinders are fitted in data points to test the reliability and robustness of the algorithm from section 4.2.1. For this purpose the number of measured areas of a cylinder is changed. This is based on [2] and tests the algorithm for 108 measurements and reduces it until a reliable fitting is no longer possible. One measurement is a section of a shaft with the dimension $0.8\,\text{mm} \times 0.8\,\text{mm}$. As a reminder, ideally one measurement area is sufficient to fit a cylinder. However, since the measurement data is noisy and only represents a very small section, this cannot be guaranteed. The implemented generation for cylinder point data allows to change different influences like noise or position of the cylinder. In this work, many adjustments were made to enable a realistic scenario for the simulated measurement of micro lead. However, the algorithms for generating cylinder data were not restricted and can also be used for any cylinders without restrictions. All programs for creating and fitting a cylinder can be found in my git repository [79]. The following settings were made for a simulated micro lead measurement:

$$\begin{aligned}
\tilde{\mathbf{w}} &\in \begin{pmatrix} p^{(\text{unif})}(x \mid -10^{-3}, +10^{-3}) & p^{(\text{unif})}(y \mid -10^{-3}, +10^{-3}) & p^{(\text{unif})}(z \mid 1-10^{-3}, 1+10^{-3}) \end{pmatrix}^T \\
\mathbf{w}^{(train)} &= \frac{\tilde{\mathbf{w}}}{\|\tilde{\mathbf{w}}\|_2} \\
r &\in [12\,\text{mm} - 1\,\mu\text{m}, 12\,\text{mm} + 1\,\mu\text{m}] \\
\mathbf{c} &\in \\
&\begin{pmatrix} p^{(\text{unif})}(x \mid -10^{-3}, +10^{-3}) & p^{(\text{unif})}(y \mid -10^{-3}, +10^{-3}) & p^{(\text{unif})}(z \mid \delta^{(\text{height})} - 10^{-3}, \delta^{(\text{height})} + 10^{-3}) \end{pmatrix}^T \\
n^{(\text{data points})} &\in \{2500, ...\}, \quad n^{(\text{circumference})} \in \{1, 4, 9, 18, 36\} \\
\delta^{(\text{arc length})} &= \frac{0.8\,\text{mm}}{12\,\text{mm}}, \quad \delta^{(\text{height})} = \{0\,\text{mm}, 2\,\text{mm}, 4\,\text{mm}\} \\
h^{(\text{data points})} &= 0.8\,\text{mm}.
\end{aligned}$$

The direction of the cylinder $\mathbf{w}$ ideally corresponds to the z-axis. A slight uncertainty in all directions is caused by an uniformly distributed random variable, which leads to a non-ideal position of the cylinder axis $\mathbf{w}^{(train)}$. The shaft used here has a radius r of ideally $12\,\text{mm}$ with a simulated deviation of $\pm 1\,\mu\text{m}$, which should simulate the deviation of the manufacturing process. The simulated surface on the cylinder, see also figure 4.6, ideally has its origin in $\mathbf{c} = \begin{pmatrix} 0 & 0 & \delta^{(\text{avg, height})} \end{pmatrix}^T$, where $\delta^{(\text{avg, height})}$ is the average of the selected different height values[4]. Here, the uncertainty of the central position $\mathbf{c}$ of the cylinder is changed by a uniform random variable. A measured area in this study consists of at least $n^{(\text{data points})} = 2500$ and a maximum of $n^{(\text{data points})} = 250000$ measuring points. This is in most cases a simplification, since typical measuring instruments, such as a confocal sensor, often have a significantly higher point density than the selected number $n^{(\text{data points})}$. In addition, the number of measuring points on the circumference $n^{(\text{circumference})}$ of the cylinder can be changed. The measured area of the circumferential surface has a dimension of $0.8\,\text{mm} \times 0.8\,\text{mm}$, which is why the central angle $\delta^{(\text{arc length})}$ and the height of the data points $h^{(\text{data points})}$ have the corresponding values. Following [2] 108 measured areas on the shaft are used. These number of measured areas are made for three different height settings $\delta^{(\text{height})}$ and with $n^{(\text{circumference})} = 36$, i.e. 36 measurements around the circumference. To identify the limits of the fit, the number of measurements is reduced. At first still $n^{(\text{circumference})} = 36$ measurements on the circumference are simulated, but only two different heights are selected for $\delta^{(\text{height})}$. The next step use only one height information $\delta^{(\text{height})}$. The described three cases are shown schematically in figure 4.6. To make the fitting task more difficult, only 18 measurements

---

[4] Assumes that two heights with $0\,\text{mm}$ and $2\,\text{mm}$ are selected. Then $\delta^{(\text{avg, height})} = 1\,\text{mm}$.

around the circumference are taken for one height setting, followed by 9, 4 and finally one simulated measurement area around the circumference. According to [21], the use of such a small area of the circumferential surface is not sufficient to perform a meaningful cylinder fit. For an uncertainty analysis, a test set consisting of 20 data for all cases mentioned was simulated and validated.

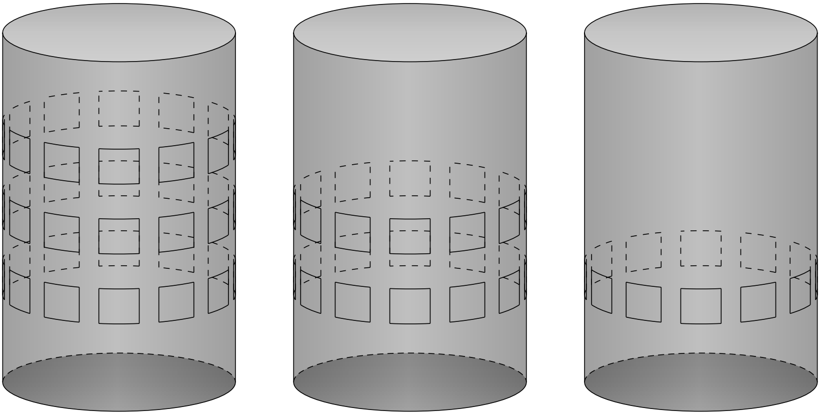


Figure 4.6: The illustration shows the fitting of a cylinder for a simulated micro lead measurement. In the left illustration, three different heights were selected and each height was measured 36 times around the circumference. The visualization in the middle and the visualization on the right also shows 36 measurements on the circumference, but with two respectively one height setting.

Validation Various settings has been made to validate the model-based fitting algorithm described in section 4.2. Table 4.1 shows a summary of the settings used for the generated points as well as the minimum and maximum deviation for the radius r and the unit direction $\mathbf{w}$ as well as the mean value for all deviations and the mean square error MSE. The amount of used test data for validation is 20. To determine the values for the deviations in table 4.1, the algorithms from 4.2 and 4.3 were used. For this, the values radius $r^{(fit)}$ and direction $\mathbf{w}^{(fit)}$ determined by the fitting were compared with the ground truth radius $r^{(real)}$ and the ground truth directions $\mathbf{w}^{(real)}$. Table 4.1 shows that the results for 108-measuring areas up to 4-measuring areas lead to comparable good results. It should be noted, that the comparability between the individual settings and results must also be considered critically, since different measurement points influence the result. For example, significantly fewer data points $n^{(\mathrm{scope})}$ per measurement area were simulated for 108 measurement areas than for 4 measurement areas.

From the results in table 4.1 for the fitting of radius $r^{(real)}$ and direction $\mathbf{w}^{(real)}$ it is clear that the fitting based on one measuring area is not satisfying. The deviation for this, also recognizable by the MSE in table 4.1, is too high in relative comparison to the other fits with multiple measuring areas. The results for the fit using 4 or up to 108 measuring areas shows a smaller deviation from the ground truth. It should be pointed out that due to the different number of $n^{(datapoints)}$ measurement data points in one measurement area, a direct comparison must be critically considered. The results give indications that for the simulated case 108 measuring ranges are not strictly necessary to achieve the desired resolution accuracy for the micro lead measurement. Good results for radius $r^{(real)}$ and direction $\mathbf{w}^{(real)}$ could also be obtained for 4 or 9 measurement ranges for simulated data.

The fitting of two cylinders, for 4 and 108 measuring areas, are shown in figure 4.7. The darker cylinder and the corresponding darker direction $\mathbf{w}$ represents the ground truth cylinder, the brighter one the calculated cylinder fitting.

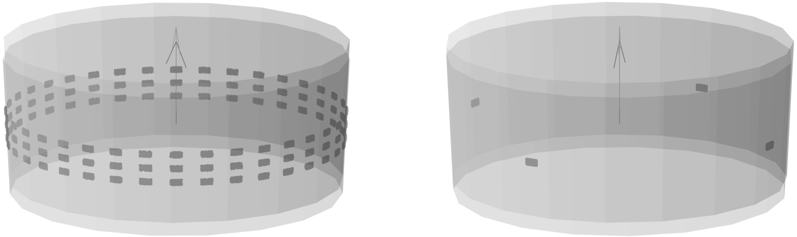


Figure 4.7: Two cylinders are mounted as shown. In the upper figure, 108 measuring areas were used for assembly, while in the lower figure, 4 measuring areas were used. The darker visualized cylinder and arrow, which indicates the direction of the fitted cylinder, represents the actual cylinder and the actual direction. The illustration for the brighter cylinder and the brighter arrow reflect the fit. For better comparability, the point of the centre has been shifted slightly upwards. In the actual result this is not present.

Summary By determining the texture direction Θ^t with a resolution accuracy of less than one arc minute using the methodology in section 4.1 and fitting the cylinder with at least 4 measurement surfaces using the approach from section 4.2.1, the estimation of the direction $\mathbf{w}^{(fit)}$ and thus the determination of the micro lead angle Ψ for simulated measurement data can be determined with sufficient accuracy. These results are based on simulated values and must be verified for real measurements.

Algorithm 4.2: Radius error estimation for cylinder fitting

Input: Vector $\mathbf{r}^{(real)}, \mathbf{r}^{(fit)} \in \mathbb{R}^n$ for n many measuremets

Output: Average of ERR and $MSE(\mathbf{r}^{(real)}, \mathbf{r}^{(fir)})$ from table 4.1

```
Estimate the absolute value of $\mathbf{r}^{(real)} - \mathbf{r}^{(fit)}$ component-wise. Output is $ERR$.
Estimate the square value of $\mathbf{r}^{(real)} - \mathbf{r}^{(fit)}$ component-wise. Output is $ERR2$.
Estimate the mean value of $ERR$. Output is $ERR$.
Estimate the mean value of $ERR2$. Output is $MSE(\mathbf{r}^{(real)}, \mathbf{r}^{(fir)})$.
```

Algorithm 4.3: Direction error estimation for cylinder fitting

Input: Matrix $\mathbf{w}^{(real)}, \mathbf{w}^{(fit)} \in \mathbb{R}^{n\times 3}$ for n many measuremets

Output: Average of ERR and $MSE(\mathbf{r}^{(real)}, \mathbf{r}^{(fir)})$ from table 4.1

```
Estimate the absolute value of $\mathbf{w}^{(real)} - \mathbf{w}^{(fit)} \in \mathbb{R}^{n\times 3}$ component-wise. Output is $ERR$.
Estimate the mean value of every row of $ERR$. Output is a new $ERR \in \mathbb{R}^n$.
Estimate the square value of $\mathbf{w}^{(real)} - \mathbf{w}^{(fit)} \in \mathbb{R}^{n\times 3}$ component-wise. Output is $ERR2$.
Estimate the mean value of $ERR$. Output is average of $ERR$.
Estimate the mean value of $ERR2$. Output is average of $MSE(\mathbf{w}^{(real)}, \mathbf{w}^{(fit)})$.
```

Table 4.1: The table shows the different parameters for a different total number of measurement areas. In addition, the deviations of the fitted parameters, radius $r^{(fit)}$ and direction $\mathbf{w}^{(fir)}$, can be seen. In each case, an interval of the deviation was specified. In addition, the mean value of the deviations and the mean square error MSE were given. The mean square error MSE is used later for comparability with the neural network. The calculation of ERR and all deviation variables are described in algorithm 4.2 and algorithm 4.3.

Settings			
Total number of measuremet areas	$n^{(Datapoints)}$ per measurement area	$n^{(circumference)}$ per height adjustment	Total number of selected heights
108	2500	36	3
72	2500	36	2
36	2500	36	1
18	10000	18	1
9	22500	9	1
4	250000	4	1
1	250000	1	1
Radius	$\mathbf{r}^{(real)}, \mathbf{r}^{(fit)} \in \mathbb{R}^{20}, \quad ERR = \text{abs}\left(\mathbf{r}^{(real)} - \mathbf{r}^{(fit)}\right) \in \mathbb{R}^{20}$		
Total number of measuremet areas	Interval of ERR	Average of ERR	$MSE(\mathbf{r}^{(real)}, \mathbf{r}^{(fit)})$
108	$\left[6.199 \times 10^{-5}, 1.013 \times 10^{-3}\right]$	7.463×10^{-4}	6.135×10^{-7}
72	$\left[7.248 \times 10^{-5}, 9.966 \times 10^{-4}\right]$	5.017×10^{-4}	3.187×10^{-7}
36	$\left[2.861 \times 10^{-6}, 9.365 \times 10^{-4}\right]$	4.474×10^{-4}	2.777×10^{-7}
18	$\left[1.202 \times 10^{-4}, 9.699 \times 10^{-4}\right]$	4.993×10^{-4}	3.221×10^{-7}
9	$\left[4.101 \times 10^{-5}, 8.755 \times 10^{-4}\right]$	4.610×10^{-4}	2.709×10^{-7}
4	$\left[4.482 \times 10^{-5}, 9.260 \times 10^{-4}\right]$	5.553×10^{-4}	3.903×10^{-7}
1	$\left[1.160 \times 10^{1}, 1.161 \times 10^{1}\right]$	1.161×10^{1}	1.348×10^{2}
Direction	$\mathbf{W}^{(real)}, \mathbf{W}^{(fit)} \in \mathbb{R}^{20\times 3}, \quad ERR = \text{mean}\left(\text{abs}\left(\mathbf{W}^{(real)} - \mathbf{W}^{(fit)}\right)\right) \in \mathbb{R}^{20}$		
Total number of measuremet areas	Interval of ERR	Average of ERR	$MSE(\mathbf{W}^{(real)}, \mathbf{W}^{(fit)})$
108	$\left[4.264 \times 10^{-5}, 5.979 \times 10^{-4}\right]$	2.946×10^{-4}	1.097×10^{-7}
72	$\left[7.780 \times 10^{-5}, 5.201 \times 10^{-4}\right]$	3.074×10^{-4}	1.091×10^{-7}
36	$\left[1.100 \times 10^{-4}, 6.093 \times 10^{-4}\right]$	3.499×10^{-4}	1.406×10^{-7}
18	$\left[2.736 \times 10^{-5}, 7.246 \times 10^{-4}\right]$	3.591×10^{-4}	1.650×10^{-7}
9	$\left[8.339 \times 10^{-5}, 5.375 \times 10^{-4}\right]$	2.851×10^{-4}	9.878×10^{-8}
4	$\left[4.862 \times 10^{-5}, 6.626 \times 10^{-4}\right]$	3.206×10^{-4}	1.205×10^{-7}
1	$\left[1.236 \times 10^{-4}, 1.321 \times 10^{-3}\right]$	6.081×10^{-4}	4.625×10^{-7}

4.3 Reconstruction of Cutting Edges using a Robust Filter

As described in chapter 2, the reconstruction of cutting edges is of interest. If the depth or length of a defect is not too large, the cutting edge can still be reworked. Generally, an existing reference is required to determine the necessary parameters. This can either be provided by a computer-based model or must be reconstructed from measured data. A method for generating an ideal reference edge using model-based methods is presented in section 4.3.2. Due to legal restrictions, real measurements of cutting edges are not used in this section. However, the following approach was tested on real cutting edges and also gave good results.

This section is structured in such a way that first in the section 4.3.1 the generation of simulated cutting edges is considered. The last two sections 4.3.2 and 4.3.3 describe the algorithm for reconstruction and validate it against simulated data.

4.3.1 Generation of Simulated Cutting Edges

Regardless of the legal basis and the use of real measurement data for cutting edges, a methodology for generating cutting edge measurements is required. The reason for this is that machine learning methods require a lot of measurement data for training. However, simulating cutting edges is not a trivial problem. They can have different shapes. Some different types of cutting edges are shown in figure 4.8. The plane shown in this figure 4.8 is defined as the x-z plane $E^{(xz)}$, since the basis vector is determined by the direction of the x-axis and the z-axis. The y-direction can be uniquely determined by the right-hand rule. The simulated profiles from $E^{(xz)}$ and figure 4.8 are results from measured cutting edges.

In both figures 4.8 and 4.9 it can be seen that the areas consist of different slopes, with the shape containing positive slopes on the left side of the maximum and negative slopes on the right side of the maximum[5]. Now, it is generally the case that gradients near the maximum are smaller than those further out. This often leads to gradients that drop from a larger positive number to a negative number with a larger magnitude, see also figure 4.9. So when creating cutting edges, it is important to find the areas with

---

[5] see figure 4.8.

different gradients and thus the points between these areas. Once all these points are found, a function $f \in \mathbb{P}_d$ can be used to model the cutting edge in that area. In summary, the modelling and therefore the generation of data depends heavily on these points and areas. These different areas and points can be identified using the steps described next.

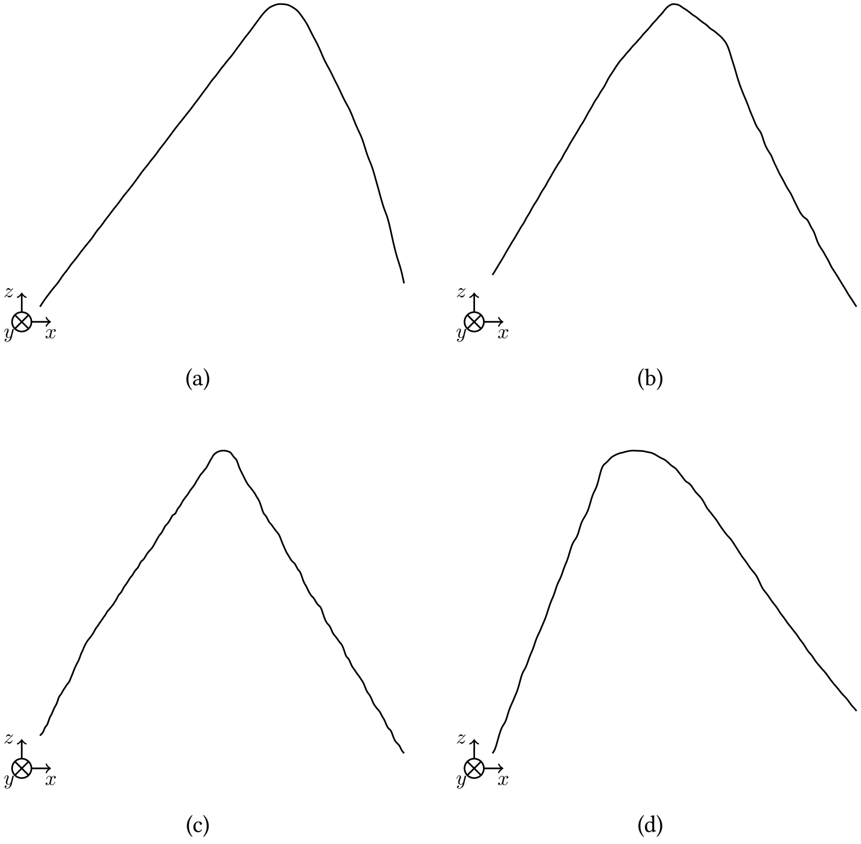


Figure 4.8: Shown are different types of cutting edge profiles within the $E^{(xz)}$ plane. In these figures it can be seen that these profiles are different from each other. This figure also serves to define the used coordinate system.

In a first step, real measurement data are used and thus measurement points of a plane $E^{(xz)}$ are considered, similar to the figure 4.8. In the next step, the noise of the measurement data is reduced with the help of a filter. For our application, a Hanning window was used for filtering. This can be calculated with a convolution analogous to the moving average, using a special discrete window function

$$w(n) = 0.5 \cdot \left(1 - \cos\left(\frac{2\pi n}{L-1}\right)\right), n \in 0, 1, \dots L-1 \subset \mathbb{N}. \tag{4.29}$$

L describes the length of the window. As can be seen in figure 4.9, after filtering, the discrete slopes of the cutting edge are determined within the $E^{(xz)}$-plane. This figure 4.9 also shows the typical gradient character of the cutting edges. In the example shown in figure 4.9, it can be seen that the gradients change exactly when a point P is reached between two areas. However, as the data is still noisy, it is difficult to identify this point P and therefore the area. These areas are identified by using histograms. Here, the gradients are transferred to a histogram in order to identify the areas with the same gradient with the help of a maximum description and neighbourhood suppression.

Another method to determine the points and thus the areas within the plane $E^{(xz)}$ can be done by determining the curvature. The numerically determined curvature and the corresponding histogram can also be seen in figure 4.9. The curvature is investigated by the second derivative, which is determined by the filtered first numerically approximated derivative of the filtered cutting edge plane $E^{(xz)}$. However, since the surfaces of the cutting edge can often be approximated by straight lines, a good interpretation of the different slopes is not always possible by evaluating the curvature alone. It makes sense to combine both pieces of information to verify the different areas. A good reconstruction of an cutting edge plane can be achieved by combining the first and second numerical derivation[6].

If the points are now determined, then the region is modelled by a function $f \in \mathbb{P}_d$. However, the points P found have a high relevance. If a standard least squares algorithm were used to fit the data in this region, the deviation in the specific point P itself would be high, as these points also represent the boundary points of each region. So a constraint is needed that requires interpolation of the points P. This leads to an optimisation problem with constraints and can be solved by the Karush-Kuhn-Tucker theorem. To describe the algorithm for optimisation with constraints, the coordinates

---

[6] Precisely approximated derivation of pre-filtered data

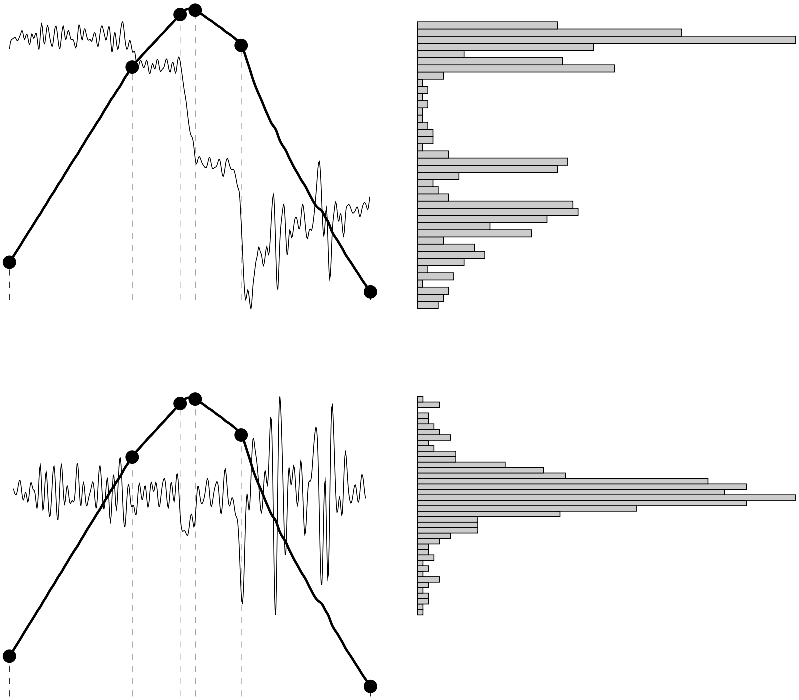


Figure 4.9: An exemplary profile of a cutting edge is shown. With thick circles in this profile, points are defined. The defined area of a cutting edge is located between two of these points. The first and second approximated derivation of the cutting edge profile is shown in thin lines. On the right of the corresponding cutting edge the histograms of the approximated derivatives are visualized. These are used to clearly identify the individual areas.

$\left(x_i^{(P)}, z_i^{(P)}\right), i = 0, 1, \ldots, p$ must be given, where p indicates the number of fixed points P. Furthermore, an optimisation problem[7] based on least square can be specified as follows:

$$(P): \quad \min_a \|\mathbf{Xa} - \mathbf{z}\|^2 \quad \text{s.t} \quad \mathbf{x}_j^{(P)^T}\mathbf{a} - z_j^{(P)} = 0, \quad j \in J, \tag{4.30}$$

---

[7] see equation 3.19

where $\mathbf{z} \in \mathbb{R}^n$ and

$$\mathbf{X} = \begin{pmatrix} 1 & x_0 & \cdots & x_0^d \\ \vdots & & & \vdots \\ 1 & x_{n-1}^1 & \cdots & x_{n-1}^d \end{pmatrix} \in \mathbb{R}^{n \times d+1} \tag{4.31}$$

$$\mathbf{a} = \begin{pmatrix} a_0 & a_1 & \dots & a_d \end{pmatrix}^T \in \mathbb{R}^{d+1} \tag{4.32}$$

$$\mathbf{x}_j^{(P)} = \begin{pmatrix} 1 & x_0^{1^P} & \dots & x_0^{d^P} \end{pmatrix}^T \in \mathbb{R}^{d+1}, \quad z_j^{(P)} \in \mathbb{R}. \tag{4.33}$$

The approach to solve this optimization problem from equation 4.30 is similar to the source [80], by determining the Lagrange function

$$\mathcal{L}\left(\mathbf{a}, \boldsymbol{\lambda}\right) = -\|\mathbf{X}\mathbf{a} - \mathbf{z}\|^2 + \lambda_0(\mathbf{x}_0^{(P)^T}\mathbf{a} - z_0^{(P)}) + \lambda_{|J|-1}(\mathbf{x}_{|J|-1}^{(P)^T}\mathbf{a} - z_{|J|-1}^{(P)}). \tag{4.34}$$

If the equations

$$\frac{\partial \mathcal{L}\left(\mathbf{a}, \boldsymbol{\lambda}\right)}{\partial a_i} = 0, \quad i = 0, \dots, d \tag{4.35}$$

and

$$\frac{\partial \mathcal{L}\left(\mathbf{a}, \boldsymbol{\lambda}\right)}{\partial \lambda_j} = 0, \quad j = 0, \dots, |J| - 1 \tag{4.36}$$

are used to determine the optimum point $\mathbf{x}^{(opt)}$, the following

$$\begin{pmatrix} 2\mathbf{X}^T\mathbf{X} & \mathbf{X}^{(P)^T} \\ \mathbf{X}^{(P)} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{a}^{(opt)} \\ \boldsymbol{\lambda}^{(opt)} \end{pmatrix} = \begin{pmatrix} 2\mathbf{X}^T\mathbf{z} \\ \mathbf{z}^{(P)} \end{pmatrix} \tag{4.37}$$

with

$$\mathbf{X}^{(P)} = \begin{pmatrix} 1 & x_0^{1^P} & \dots & x_0^{d^P} \\ \vdots & & & \vdots \\ 1 & x_{|J|-1}^{1^P} & \dots & x_{|J|-1}^{d^P} \end{pmatrix} \in \mathbb{R}^{|J| \times d+1}, \quad \mathbf{z}^{(P)} = \begin{pmatrix} z_0 \\ \vdots \\ z_{|J|-1} \end{pmatrix} \in \mathbb{R}^{|J|}$$

applies. The solution from equation 4.37 for $\begin{pmatrix}\mathbf{a}^{(opt)} & \boldsymbol{\lambda}^{(opt)}\end{pmatrix}^T$ yields the desired coefficients $\mathbf{a}^{(opt)}$ of the polynomial to be fitted.

To determine the polynomials for the cutting edge areas, $d = 3$ was chosen. The interpolation points P and the number of points to be interpolated within a range are automatically determined, see algorithm 4.4. In our example, the number of automatically determined interpolation points $|J|$ was often either $|J| = 2$ or $|J| = 3$.

The coefficients $\mathbf{a}^{(opt)}$ of all polynomials for all planes $E^{(xz)}$ and the subdivided areas of the cutting edges are the same for a single profile. This assumption is not valid for real measurement data, as it is not possible to always obtain the same polynomial coefficients for manufacturing and statistical reasons. In fact, the shape of the cutting edge can also change along the y-axis[8]. In order to simulate different cutting edges of one type, also with regard to machine learning and the amount of data required, the determined polynomial coefficients were fixed within a range by using a uniform distribution. The limits of this used uniform distribution were determined empirically. Furthermore, it was simplistically assumed that the shape and structure at the contours of the cutting edge do not change along the y-axis for simulated data. However, this is usually not the case for real cutting edges and describes a simplified reality.

For the y-axis it is taken into account that the shape of the cutting edge, see also figure 4.10, is not always a straight line. In a first step, the curvature of the cutting edge along the y-axis is simulated. The following index values are taken into account

$$i^{(argmax)} = \arg\max_i z_i, \quad i = 0, \ldots, n-1 \tag{4.38}$$

and the data points $\left(x_i^{(argmax)}, z_i^{(argmax)}\right)$ for all existing measured n profile sections. However, since the generated section edges are based on real measurements and these may have errors, the determination of $i^{(argmax)}$ alone is not sufficient. It could be that due to a defect on the cutting edge, the position $i^{(argmax)}$ is different from a non-defective cutting edge. Instead, a discrete parameterisation with the parameterisation

---

[8] see also the direction convention from figure 4.8.

variable

$$\begin{aligned} t_{i+1} &= t_i + \| \begin{pmatrix} x_{i+1} & y_{i+1} & z_{i+1} \end{pmatrix}^T - \begin{pmatrix} x_i & y_i & z_i \end{pmatrix}^T \|, \\ t_0 &= 0, \quad i = 0, \dots, n-2 \end{aligned} \tag{4.39}$$

is performed. If the parameterization from equation 4.39 is used for the x, y and z directions, a function $f\colon \mathbb{R} \supset D \to \mathbb{R}^m,\ t \mapsto f(t)$ can be created. Since the n many maximum points $\left(x_i^{(argmax)}, z_i^{(argmax)}\right)$ of the plane $E^{(xz)}$ cannot be sufficiently described by a polynomial fit, spline functions are used for the description. For cutting edges, spline functions were created with $s^{(x)}, s^{(y)}, s^{(z)} \in \mathbb{S}_3$ for the individual directional components. As known from definition 3.1.2 this depends on a so-called smoothing parameter λ. This smoothing parameter was determined empirically and was set to a default value.

In summary, the generation of cutting edges based on a measurement can be described by the pseudo code described in the algorithm 4.4. Possible generated cutting edges are shown in figure 4.10. In this thesis, two types of cutting edges were generated, each with a different edge profile in the y-direction and contour in the $E^{(xz)}$-plane.

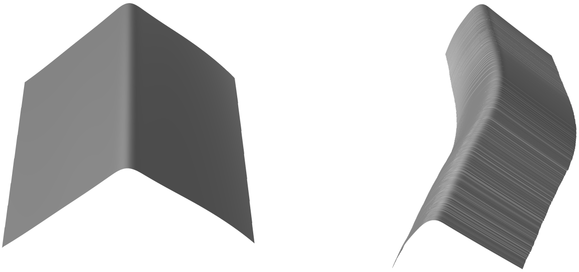


Figure 4.10: Shown are two different cutting edge types. They differ in their profile in the $E^{(xz)}$ plane, as well as in their curvature along the defined y-direction.

4.3.2 Determination of Ideal References for Cutting Edges using model-based methods

This section describes the generation of ideal cutting edges for measured cutting edges with defects. The methodology described here uses only model-based approaches. The ideal reference is necessary to calculate parameters for the validation of a defective cutting edge. It is assumed that no CAD data is available beforehand, see also figure 2.5. To describe the algorithm, equidistant grid points at $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{m \times n}$ and the corresponding topography measurement data $\mathbf{Z} \in \mathbb{R}^{m \times n}$ are used. Here m describes the number of measurement points in y direction and n the number of measurements in x direction. The definition of the direction is noted in figure 4.8. In general, not all measurement points are completely present in $\mathbf{Z}$. Since the cutting edges have steep surfaces, they are sometimes difficult to measure. For this reason, some measurement points in real samples often have undefined values. To fill these gaps, which are usually not marked with a number in the measurement data, the model-based method uses cubic hermetic splines. The implementation of hermitian piecewise polynomials in C++ is based on [81]. This is necessary because a polynomial correction and a spline interpolation would lead to higher oscillations due to partially large missing parts. Another advantage is that, in contrast to smoothing splines[9] the hermetic piecewise interpolation is independent of input parameters and a realistic path of the edge in the $E^{(xz)}$-plane can be realised. Moreover, by using hermetic piecewise polynomials, the measured data are not smoothed in the first step and remain unchanged after filling all gaps.

The next step is to determine the maximum points $\left(x_i^{(argmax)}, z_i^{(argmax)}\right)$ of each plane in $E^{(xz)}$ for $i = 0, \dots, m-1$ using equation 4.38. These maximum points are used to determine three areas, $area_0, area_1$ and $area_2$[10]. Here $area_1$ describes the index values for the cutting edge rounding area in the $E^{(xz)}$-plane and thus also the region with stronger defects. With respect to $area_1$, the two regions $area_0$ and $area_2$ lie to the left and right of $area_1$ and often have no or small defects. In the model-based approach, the defects are interpreted as larger imperfections and therefore corrected with a Gaussian filter, see section 3.1.1. However, for the different areas $area_0, area_1$ and $area_2$, different smoothing parameters α[11] were used for the robust Gaussian filter. By choosing different smoothing parameters α, it is possible to maintain the contour of the cutting edge in the side regions while correcting the errors in $area_1$. To ensure that the generated ideal reference is also mathematically smooth throughout its

---

[9] see definition 3.1.2

[10] these are different areas from those in section 4.3.1

[11] see equation 3.17.

representation, bicubic smoothing splines are used to generate an ideal cutting edge reference. Figure 4.11 shows the main steps of the algorithm for generating cutting edges using model-based methods.

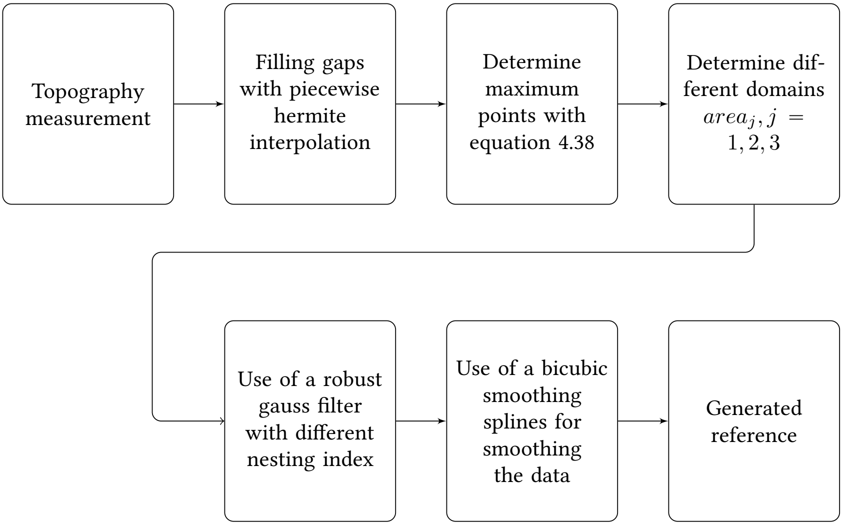

Topography measurement
Filling gaps with piecewise hermite interpolation
Determine maximum points with equation 4.38
Determine different domains $area_j, j = 1, 2, 3$
Use of a robust gauss filter with different nesting index
Use of a bicubic smoothing splines for smoothing the data
Generated reference

Figure 4.11: Several steps are necessary to determine an ideal reference geometry for cutting edges. The figure summarizes the most important steps for generating ideal references with model-based methods.

4.3.3 Application and Validation of the model-based Approach for Reconstruction of Simulated Cutting Edges

To validate the algorithm described in section 4.3.2, real and simulated cutting edges with different defect sizes were used. It should be noted that only the results for simulated cutting edges are presented in this thesis. For legal reasons, the measurements of real cutting edges and their generated reference cannot be shown. However, it should be mentioned that very good results were also obtained for real cutting edges. The results for real cutting edges could keep up with other methods reflecting the state of the art and even provided better results.

In order to test the results on simulated samples, two different cutting edge types are generated from the section 4.3.1 with algorithm 4.1. The two cutting edges have a different curvature, which can also be seen in the course of the maximum points, see figure 4.10. One with a straight-line curvature along the y-direction but with a more complex profile in the $E^{(xz)}$-plane and a second cutting edge shape that has no straight-line curvature in the y-direction but a simpler profile in the $E^{(xz)}$-plane. Both cutting edges are modified and tested with defects of different depth and length. These defects are very similar for both simulated cutting edges. In total, four artificial defects were added, two of which are very large and deep defects along the cutting edge. The other two defects are very small and almost disappear in the added noise. This noise was added in x and y directions with a uniform distribution between $-2\,\mu\text{m}$ and $2\,\mu\text{m}$. Thus, the added noise can have a maximum reduced amplitude of $4\,\mu\text{m}$ and a maximum added amplitude of $4\,\mu\text{m}$. The results for all cutting edges are shown from figure 4.12 to 4.15.

These four illustrations from figure 4.12 to figure 4.15 are showing the two described cutting edge types. Figure 4.12 and 4.13 shows the cutting edge with a straight cutting edge line in y direction and a more complex shape in the $E^{(xz)}$ plane (type A). The other two figures 4.14 and 4.15 illustrate the cutting edge with a curved shape in y direction (type B). However, the interpretation of the results for all figures from 4.12 to 4.15 have the same structure.

- The top figure shows the desired cutting edge, once without and once with noise.
- The second illustration shows the cutting edge with defects.
- The third figure shows the reconstruction using the algorithm described in section 4.3.2.
- The last figure shows the difference between the generated reference and the cutting edge with defects.

It can be seen that the reconstruction for cutting type A works very well. This result is almost unaffected by the noise present. A determination of the parameters for the evaluation of the cutting edge is easily possible for this type. For cutting edge type B with stronger curvature in y-direction the results are not consistently good. The result of the reconstruction with noise is significantly weaker than the reconstruction without noise, see also figure 4.14 and figure 4.15. In general, reconstruction using this approach is not always successful when there is a strong defect near a curved path. This is because the Gaussian filter wants to correct strong defects, which it interprets as imperfections and therefore can no longer follow the curved shape. The same phenomenon occurs with the bicubic smoothing spline, which is subsequently used to smooth the surface of the sample. This influence can be optimised by changing the nesting index or smoothing parameter α[12] for the Gaussian filter and by changing the smoothing parameter for the bicubic smoothing spline. However, with adjustable parameters, it is no longer possible to use a "universal" solution.

---

[12] see equation 3.17

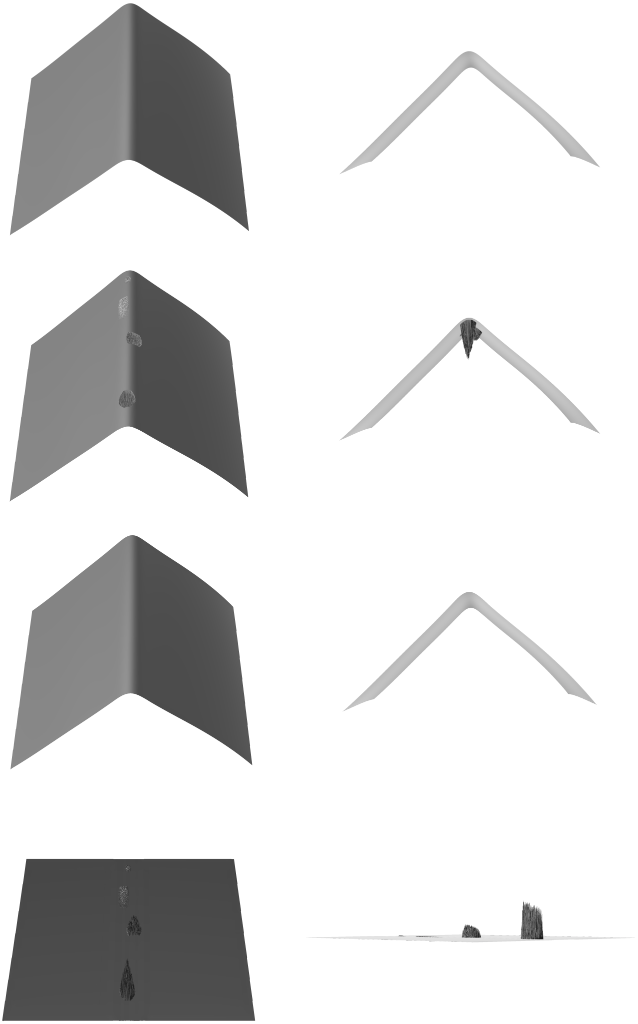


Figure 4.12: Shown is a generated cutting edge of **type A without** additional noise. Furthermore this generated cutting edge is shown once without and once with simulated defects. Also shown is the calculated ideal reference and the difference to the cutting edge with simulated defects.

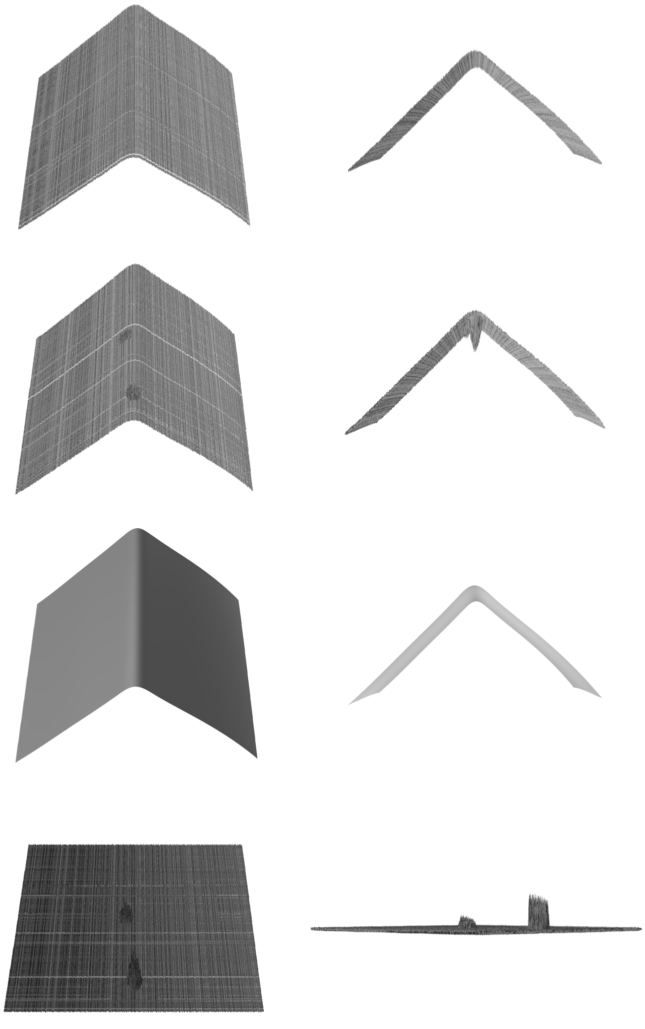


Figure 4.13: Shown is a generated cutting edge of **type A with** additional noise. Furthermore this generated cutting edge is shown once without and once with simulated defects. Also shown is the calculated ideal reference and the difference to the cutting edge with simulated defects.

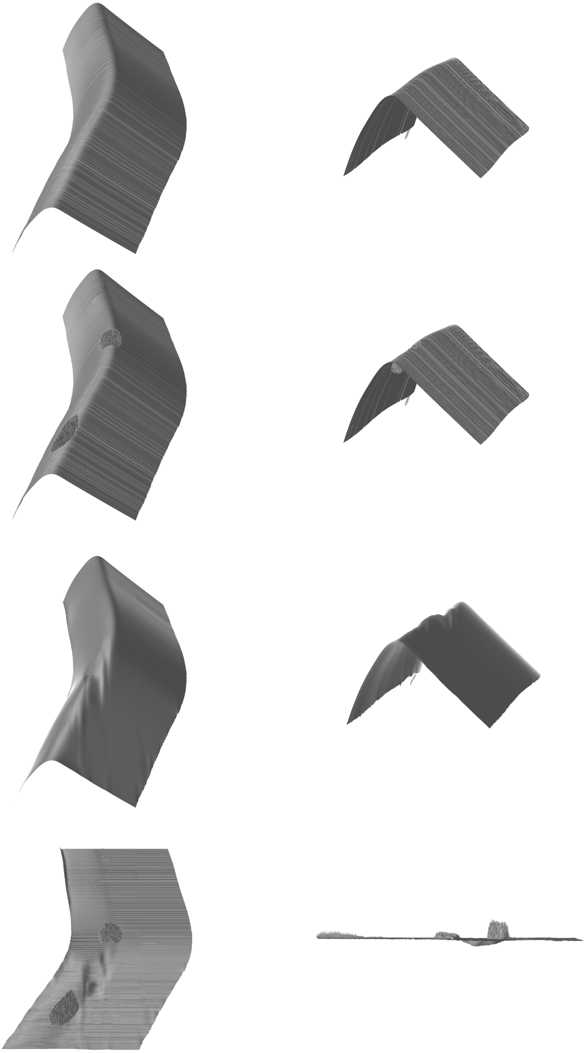


Figure 4.14: Shown is a generated cutting edge of **type B without** additional noise. Furthermore this generated cutting edge is shown once without and once with simulated defects. Also shown is the calculated ideal reference and the difference to the cutting edge with simulated defects.

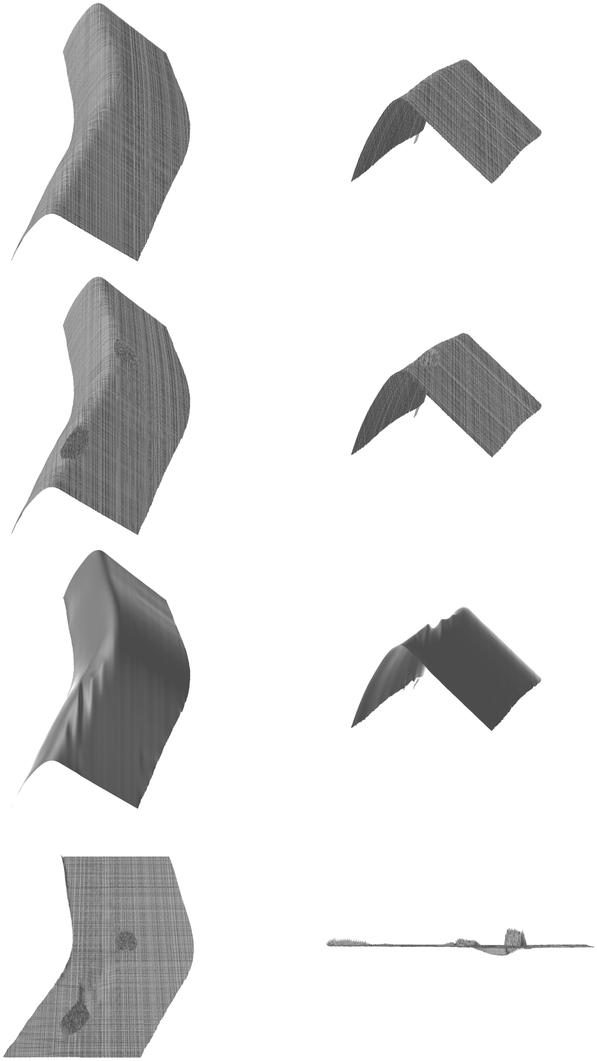


Figure 4.15: Shown is a generated cutting edge of **type B with** additional noise. Furthermore this generated cutting edge is shown once without and once with simulated defects. Also shown is the calculated ideal reference and the difference to the cutting edge with simulated defects.

Algorithm 4.4: Generating cutting edge data

```
Input: Datapoints $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \in \mathbb{R}^{n \times m}$ with $i = 0, \dots, n-1, j = 0, \dots, m$
Output: Points for generated cutting edge $\mathbf{X}^{(sim)}, \mathbf{Y}^{(sim)}, \mathbf{Z}^{(sim)} \in \mathbb{R}^{n \times m}$
Get data points $\begin{pmatrix} x_i & z_i \end{pmatrix}$ in $E^{(xz)}$
Reduced noice of $\begin{pmatrix} x_i & z_i \end{pmatrix}$ using a hanning filter $\leftarrow \begin{pmatrix} x_i^{(filter)} & z_i^{(filter)} \end{pmatrix}$
Build the first derivative of $\begin{pmatrix} x_i^{(filter)} & z_i^{(filter)} \end{pmatrix} \leftarrow \left( d\, x_i^{(filter)}, d\, z_i^{(filter)} \right)$
Filter the first derivative to reduce noise $\leftarrow \left( d\, x_i^{(filter)}, d\, z_i^{(filter)} \right)$
Build the second derivative of
  $\left( d\, x_i^{(filter)}, d\, z_i^{(filter)} \right) \leftarrow \left( d^2\, x_i^{(filter)}, d^2\, z_i^{(filter)} \right)$
Determine the critical points $P$ by histogram analysis of $\left( d\, x_i^{(filter)}, d\, z_i^{(filter)} \right)$
  and $\left( d^2\, x_i^{(filter)}, d^2\, z_i^{(filter)} \right)$
Solve the optimization problem from equation 4.30 $\leftarrow \mathbf{A}_{i,j}$ for $j = 0, \dots, |J| - 2$
Build the $E^{(xz)}$ profile using the solution of the the optimization problem from
  equation 4.30
for $i$ in range$(0, n)$ do
    Estimate $i^{(argmax)} = \arg\max_i z_i$ using
      $\begin{pmatrix} x_i^{(filter)} & z_i^{(filter)} \end{pmatrix} \leftarrow \left( x_i^{(argmax)}, z_i^{(argmax)} \right)$
end
Calculate the parameterized variable $t$ from equation 4.39
Determine the parameterized spline curve using $t \leftarrow \begin{pmatrix} s^{(x)}(t) & s^{(y)}(t) & s^{(z)}(t) \end{pmatrix}^T$
Force the spline curve to grid points by searching the nearest neighbor
  $\leftarrow \begin{pmatrix} s^{(x)}(t) & s^{(y)}(t) & s^{(z)}(t) \end{pmatrix}^T$
Combine the profiles $\mathbf{x}^{(sim)}, \mathbf{z}^{(sim)}$ along the spline curve
  $\leftarrow \mathbf{X}^{(sim)}, \mathbf{Y}^{(sim)}, \mathbf{Z}^{(sim)}$
```

5 Machine learning based approach for determining Micro Lead and Reconstruction

This chapter 5 describes the solutions to the problems described in chapter 1 and chapter 2 and the tasks derived from them using machine learning approaches. Since no work involving machine learning has been published in this field apart from the author's publications, all the approaches and results presented here have been developed in the context of this thesis.

Section 5.1 describes a convolutional neural network based architecture for determining the texture direction of ground surfaces. In addition to the architecture described in section 5.1.1, the corresponding validation on simulated surfaces is described in more detail in section 5.1.2.

Section 5.2 presents a fitting based on feedforward neural networks, whose design is described in section 5.2.2 and validated in section 5.2.3. As a basic building block, the section 5.2.1 shows a way to generate simulated measurement data for a micro lead measurement and to train and validate the corresponding weights.

The last section 5.3 of this chapter 5 describes the reconstruction of cutting edges using a deep autoencoder. The first related section 5.3.1 deals with data preparation and idea description. Section 5.3.2 presents the used architecture of the deep autoencoder. The last section 5.3.3 presents and validates the results for different cutting edge types.

5.1 Determining the texture direction using neural networks

To determine the micro lead angle Ψ, information about the texture direction Θ^t and about the circumferential direction Θ^{circ} is required. Detailed information on both parameters and on micro lead in general is described in chapter 1 and section 4.1. In this section 5.1, the two parameters Θ^t and Θ^{circ} are determined using artificial intelligence methods. First, the determination of the texture direction Θ^t is considered.

5.1.1 Architecture of a Convolutional Neural Network for determining Texture Direction

In order to achieve the accuracy of one arcminute for the texture direction using machine learning and thus obtain an alternative to the cross-correlation method from the section 4.1, a lot of measurement data from ground surfaces are necessary. One way to use machine learning to solve this task is using supervised learning. This type of machine learning requires a lot of measurement data and the corresponding ground truth[1]. However, the texture direction and thus the associated ground truth is often not known during or after processing. In addition to this aspect, the time required and the associated costs of a measurement also play a major role. To remove these hurdles and other challenges associated with real measurements, simulated measurement data is used.

For a reasonable simulation of texture directions for ground surfaces, the inverse radon transformation can be used, see figure 3.1. A good approximation can also be generated by a sine wave. Both approaches to approximate the texture directions were also used in 4.1 to validate the cross-correlation method, see also figure 4.1. If one now compares both approaches with a real measurement, e.g. from figure 4.3, it can be seen that the inverse Radon transformation approximates the ground surface more accurately. Therefore, to determine the texture direction Θ^t, a neural network with 3375 training data and 374 validation data was used to train the model. It should be noted that the

---

[1] training methods could also be used that do not require labelled data and obtain the necessary information through non-linear component analysis of the data sets. However, the other negative points, such as the number of measurements, remain. Therefore, training labelled data is useful

amount of training and validation data for machine learning scenarios is very small. However, since the simulated images have a correspondingly high resolution, it was not possible to use more data with the available hardware. Figure 5.1 shows simulated ground surfaces generated by the inverse Radon transform and their corresponding texture direction in arcminutes.

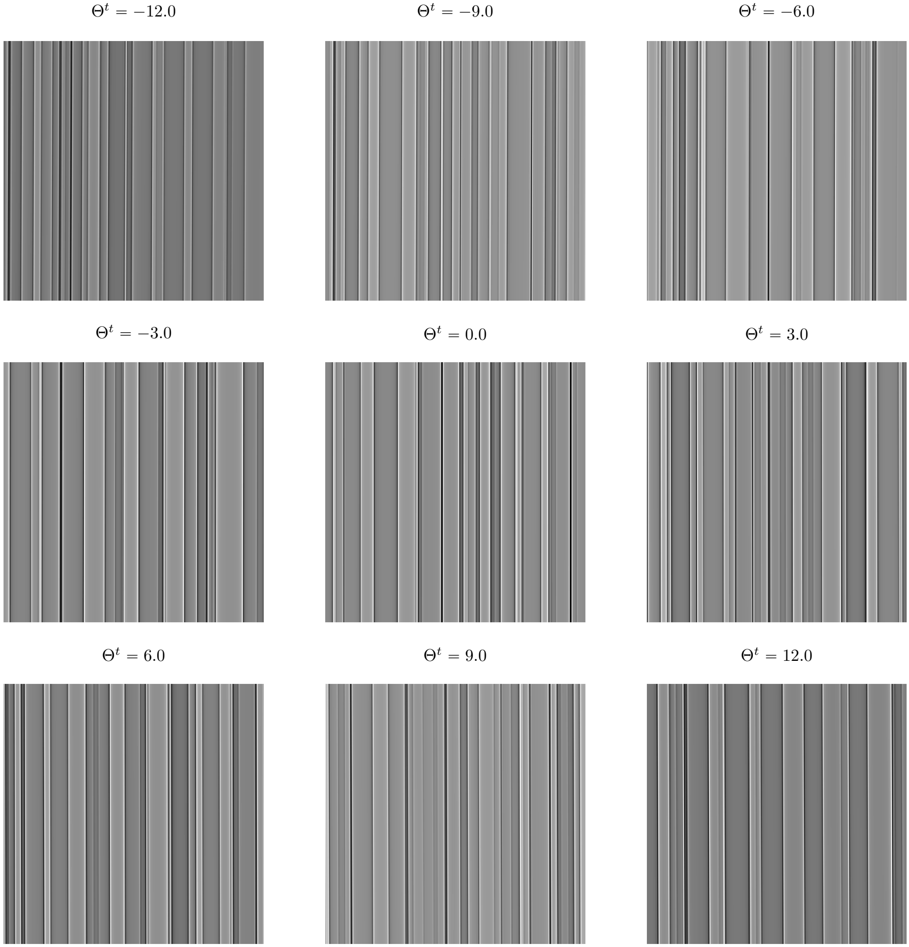


Figure 5.1: The figure shows simulated texture directions created by the inverse Radon transformation.

The implementation of the simulated texture direction data can be found in my git repository [82]. The determination of the texture direction Θ^t can be classified as a regression problem for machine learning tasks. To keep the number of parameters small when comparing the two methods, it is reasonable to use a convolutional neural network rather than a fully connected neural network. The high number of parameters for a fully connected neural network is due to the size of the input image. This leads to overfitting even for models with few neurons and one hidden layer. Furthermore, an approach that is oriented towards approaches without neural networks, such as a random forest approach, is not useful due to the large number of features. The convolutional network used is shown schematically in figure 5.2.

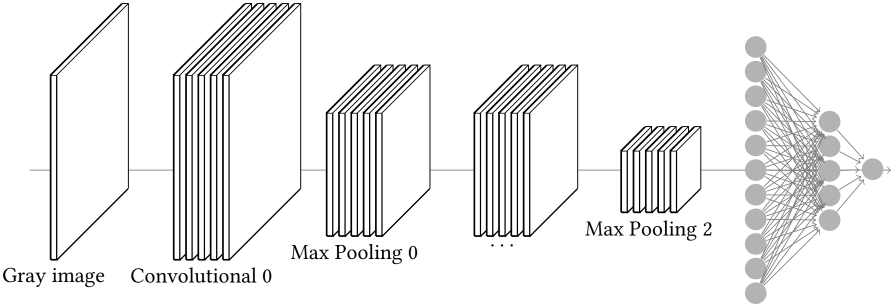


Figure 5.2: The figure shows a schematic representation of the used approach. Details on this structure can be found in table 5.1.

The structure of the individual layers can be found in table 5.1. Furthermore, the convolutional neural network used can also be found in my Git repository [82]. For the corresponding Git repository, a Medium story has been published, which publishes the idea, the approach and some kind of instructions for using and implementing this approach [9]. The training and validation process of the convolutional neural network used is shown in figure 5.3.

5.1.2 Results and Discussion

To check how the learning and validation behaviour works for the given architecture, the mean square error, cf. equation 3.1.3, was used. An excerpt of the training process is shown in figure 5.3. The training and validation process shows a desired outcome

Table 5.1: The architecture of the model is shown in table form. As padding the parameter "same" and as activation function $\varphi(x)^{ELU}$ was always used, see section 3.3.1.

Settings	Used loss function MSE		
Input Layer	Size 301×301		
Convolution Layer 0	Filter size 30	Kernel size 5×5	Strides 3×3
Max Pooling 0	Pool size 2×2	Strides size 2×2	
Convolution Layer 1	Filter size 60	Kernel size 3×3	Strides 2×2
Convolution Layer 2	Filter size 60	Kernel size 3×3	
Max Pooling 1	Pool size 2×2	Strides size 2×2	
Convolution Layer 3	Filter size 120	Kernel size 3×3	
Max Pooling 2	Pool size 2×2	Strides size 2×2	
Flattern			
Fully connected neural Network	110 Neurons		
Dropout	rate 0.4		
Output neural	1 Neuron		

without overfitting, which is prevented by the integration of drop-outs. This desired outcome was observed for two trained machine learning models.

For the first model, defined with the model name $M^{(0)}$, training and validation data of texture directions between $-6'$ and $+6'$ are used. The second model, defined with model name $M^{(1)}$, learns texture direction between $\pm 12'$. The trained and validated data has a step size of $0.5'$ for both models. At the end of the training for 500 epochs, a smaller MSE value was obtained for the first model $M^{(0)}$. This smaller value is also reflected in the MSE value of the validation. As a reminder, in order to make a qualitative statement for micro lead, the accuracy for texture direction must be less than one arc minute. This difference between the two models and the MSE values should therefore be highlighted.

To test the model for unknown data, two types of test data were generated. These are described below in the form of bullet points. As a reminder, for the first model $M^{(0)}$, a texture direction range of $-6'$ and $+6'$ was used as training and validation data. For the second model $M^{(1)}$, a range of $-12'$ and $+12'$ was used as training and validation data. For both cases 101 test data was generated.

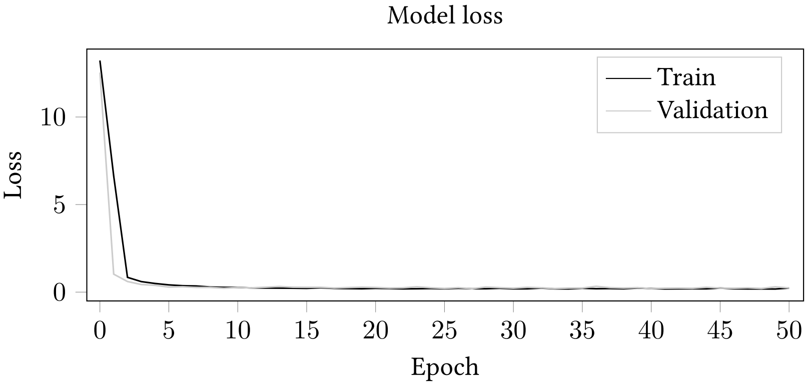


Figure 5.3: Shown is the training and validation error in dependence of the learned epochs. As loss function the mean square error MSE was chosen. It can be seen that there is no overfitting in the first 50 epochs.

- The test data is generated with different step sizes than the training and validation data for both models. The range of texture direction for the test data is between $-6'$ and $+6'$. Thus, the range of texture directions in the test data is always a subset of the generated texture direction of the training and validation data for both models.
- The test data is generated with different step sizes than the training and validation data for both models. The texture direction range for the test data is between $-12'$ and $+12'$. Thus, the texture direction range of the test data for only one model is a subset of the generated texture direction for the training and validation data. For the first model, only a range between $\pm 6'$ was used for training and validation.

Results for test data between $-6'$ **and** $6'$ For these two trained machine learning models $M^{(0)}$ and $M^{(1)}$, the predicted results are compared to the ground truth. The ground truth in this section covers a range between $\pm 6'$. The predicted data results for the model $M^{(0)}$ are shown in figure 5.4 above. The bottom plot in figure 5.4 shows the difference between the predicted results and the ground truth for the model $M^{(0)}$. It can be seen that almost every time a resolution of less than one minute of arc was achieved.

The same result, and hence the results for resolution accuracy less than one arcminute, was also obtained for the model $M^{(1)}$. The corresponding plot of the results can be seen in figure 5.6.

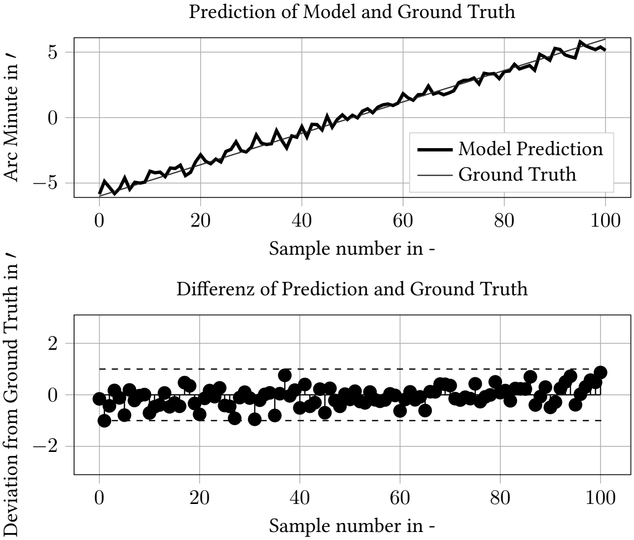


Figure 5.4: Shown are the ground truth values for the simulated texture direction of a ground surface and the predicted texture directions of the machine learning model. Here the model $M^{(0)}$ was used, which uses trained texture directions between $\pm 6'$. The test data here have a range between $\pm 6'$ for the texture direction.

Results for test data between $-12'$ and $12'$ For these two trained machine learning models, the predicted results are compared to the ground truth. The ground truth in this section covers a range between $\pm 12'$. The predicted data results for the model $M^{(0)}$ are shown in figure 5.5 above. The bottom plot in figure 5.5 shows the difference between the predicted results and the ground truth for the model $M^{(0)}$. As a reminder, the model $M^{(0)}$ used a training and validation range of $\pm 6'$ and the model $M^{(1)}$ used a training and validation range of $\pm 12'$. It can be seen that the unknown regions for the model $M^{(0)}$ could not provide a good enough prediction for the texture direction. A different result was obtained for the model $M^{(1)}$. Since the entire texture range of the test data was trained, the predicted texture direction could be determined with better accuracy.

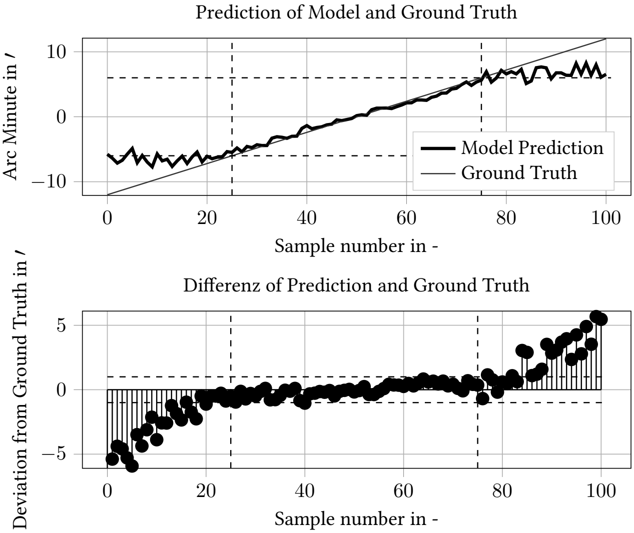


Figure 5.5: Shown are the ground truth values for the simulated texture direction of a ground surface and the predicted texture directions of the machine learning model. Here the model $M^{(0)}$ was used, which uses trained texture directions between $\pm 6'$. The test data here have a range between $\pm 12'$ for the texture direction.

However, it can be seen that with a larger learned texture area, a resolution accuracy of less than one arc minute could often not be maintained, see also figure 5.7 below.

Summary and Discussion In summary, the architecture of the convolutional neural network from table 5.3 requires a large texture area to achieve good resolution accuracy. If this is not the case, untrained areas are only indicated with a large deviation. Figure 5.6 shows that the model cannot correctly predict areas far outside the learned texture direction. The difference between the ground truth and the predicted model is so large that it can lead to a wrong interpretation in terms of leakage and micro lead. One way to better learn these divergent boundaries is to extend the training and validation data. If the training and validation data cover wider ranges as in the $M^{(1)}$ model with up to $\pm 12'$, then these boundary ranges can also be better predicted. It is conceivable that an extended convolutional neural network architecture can be used to learn larger texture direction ranges. In addition to an extended learned texture direction range, which may

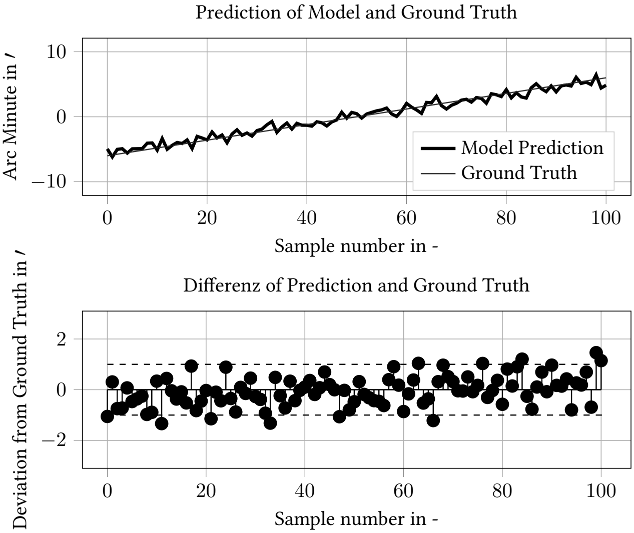


Figure 5.6: Shown are the ground truth values for the simulated texture direction of a ground surface and the predicted texture directions of the machine learning model. Here the model $M^{(1)}$ was used, which uses trained texture directions between $\pm 12'$. The test data here have a range between $\pm 6'$ for the texture direction.

require a modified convolutional neural network architecture, the evaluation needs to be tested for real measured ground surfaces. Ideally, measured training and validation data with known texture directions can be used for this. However, since these are not easy to determine in large numbers, a model with simulated ground surfaces can also be pre-trained. The weights of the trained model can then be refined in a further step and trained again with real measured data.

5.2 Determining the direction of a shaft with neural networks

If now equation 1.1 is recalled with $\Psi = \Theta^{\mathrm{t}} - \Theta^{\mathrm{circ}}$ it can be recognized, that for the determination of the micro lead angle Ψ also the circumferential direction Θ^{circ} is relevant. In the section 5.1 it was shown that for simulated texture directions the resolution below one arc minute for Θ^{t} can be achieved with machine learning approaches. If the

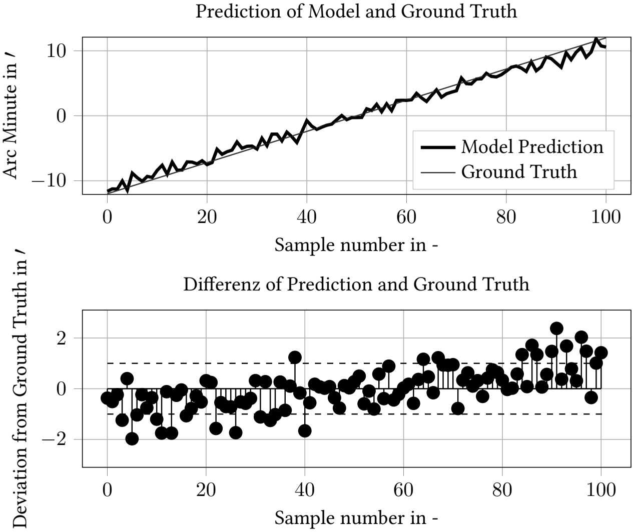


Figure 5.7: Shown are the ground truth values for the simulated texture direction of a ground surface and the predicted texture directions of the machine learning model. Here the model $M^{(1)}$ was used, which uses trained texture directions between $\pm 12'$. The test data here have a range between $\pm 12'$ for the texture direction.

micro lead angle Ψ shall be resolved below one arc minute, the determination of the circumferential direction Θ^{circ} must not differ significantly in the resolution accuracy. A method based on classical modeling is presented in section 4.2. This method gives sufficiently results for four simulated datasets with a simulated area of $800\,\mu\text{m} \times 800\,\mu\text{m}$ around the circumference with a point density of 500×500. However, with only one measurement around the circumference, the circumferential direction Θ^{circ} could not be determined satisfactorily.. A similar investigation is now being carried out for a model using machine learning methods.

5.2.1 Generation of 3D Cylinder Points

The settings for generating cylinder data are not significantly different from the settings for generating cylinder data in section 4.2.2. Therefore, this is only briefly summarised. If more information is needed, see section 4.2.2. The following settings were used to generate the training, validation and test data:

$$\begin{aligned}
\tilde{\mathbf{w}} &= \left(p^{(\text{unif})}(x \mid -10^{-3}, +10^{-3}) \quad p^{(\text{unif})}(y \mid -10^{-3}, +10^{-3}) \quad p^{(\text{unif})}(z \mid 1-10^{-3}, 1+10^{-3})\right)^T \\
\mathbf{w}^{(train)} &= \frac{\tilde{\mathbf{w}}}{\|\tilde{\mathbf{w}}\|_2} \\
r &\in [12\,\text{mm} - 1\,\mu\text{m}, 12\,\text{mm} + 1\,\mu\text{m}] \\
\mathbf{c} &= \\
&\left(p^{(\text{unif})}(x \mid -10^{-3}, +10^{-3}) \quad p^{(\text{unif})}(y \mid -10^{-3}, +10^{-3}) \quad p^{(\text{unif})}(z \mid \delta^{(\text{height})} - 10^{-3}, \delta^{(\text{height})} + 10^{-3})\right)^T \\
n^{(\text{data points})} &= \{225, \ldots\}, \quad n^{(\text{circumference})} = \{1, 4, 9, 18, 36\} \\
\delta^{(\text{arc length})} &= \frac{0.8\,\text{mm}}{12\,\text{mm}}, \quad \delta^{(\text{height})} = \{0\,\text{mm}, 2\,\text{mm}, 4\,\text{mm}\} \\
h^{(\text{data points})} &= 0.8\,\text{mm}
\end{aligned}$$

The labels used are radius r, the cylinder origin $\mathbf{c}$ and the direction of the cylinder $\mathbf{w}^{(train)}$. The number of data points for an area is given by $n^{(\text{data points})}$. The total number of data taken on the surface of the cylinder around the circumference is described by the parameter $n^{(\text{circumference})}$. Since simulated surfaces are of dimension $800\,\mu\text{m} \times 800\,\mu\text{m}$, the central angle $\delta^{(\text{arc length})}$ and $h^{(\text{data points})}$ have the corresponding values. In total, a maximum of 108 measurements are considered, since the parameter $\delta^{(\text{height})}$ can be used to realise a maximum of three adjustable heights in combination with a maximum number of $n^{(\text{circumference})} = 36$ measurements along the circumference.

5.2.2 Neural Network Model for Fitting a Cylinder

For this approach, a fully connected neural network was used. For this purpose, the existing three-dimensional data points were first flattened. This flat structure serves as the input layer. This was then combined with two hidden layers with 70 neurons each. As an output layer, 7 neurons were trained. These 7 neurons correspond to the 7 labels for uniquely identifying the corresponding cylinder in three-dimensional space. All necessary settings for the used model are shown in figure 5.8. The generation of the data and the corresponding model can be downloaded and tested from my Git repository [79]. An exemplary training and validation history of the MSE error is shown in figure 5.9 for the setting parameters.

$$n^{(\text{data points})} = 2500, \quad n^{(\text{circumference})} = 1, \quad \delta^{(\text{height})} = 0\,\text{mm}.$$

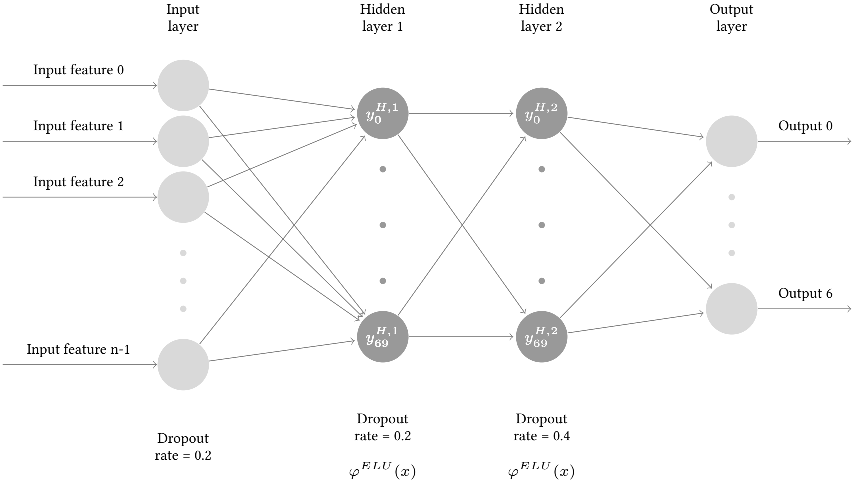


Figure 5.8: The figure shows the neural network used to fit a cylinder into a point cloud. Two hidden layers with 70 neurons each were used. As activation function $\varphi(x)^{ELU}$ was applied, see equation 3.29. To avoid overfitting Dropout was used.

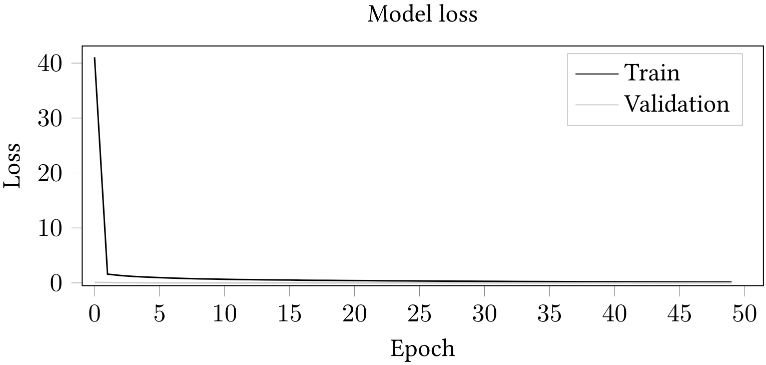


Figure 5.9: Shown is the training and validation error in dependence of the learned epochs. As loss function the mean square error MSE was chosen. It can be seen that there is no overfitting for the first 50 epochs. 6400 training data and 1600 validation data were used.

5.2.3 Application and Validation for Simulated 3D Cylinder Points of the used Model

Various settings were made to test the used model. For this purpose, the table 5.2, found on the penultimate page of this chapter 5, shows a summary of all settings. Besides the different number of simulated areas, the number of points $n^{(\text{data points})}$ was also varied. This number of data points $n^{(\text{data points})}$ is also different from the set of points in the 4.2.2 section. The set of points used for machine learning is significantly reduced due to the lack of capacity of the hardware used. The noise range of the data for the corresponding parameters $\mathbf{w}^{(train)}$, r and $\mathbf{c}$ was not changed. In addition to the settings, the results of the cylinder fitting using machine learning are also shown in table 5.2. In addition to the minimum and maximum deviation, the mean square error MSE is also noted there. This deviation is determined by the difference between the ground truth and the predicted values of the model, see algorithm 4.2 and 4.3. The table 5.2 shows that the results for 108 to 1 measurement areas are almost equally good. Comparability between settings and results is difficult because the total number of measurement points is different and therefore a different number of features affect the result. For example, only slightly more data was simulated for 108 measurement areas in total than for 4 measurement areas. The calculation of the corresponding ERR and MSE error in table 5.2 can be seen in algorithm 4.2 and 4.3. Two example fits of a cylinder, one for 108 and one for 1 measurement areas, are shown in figure 5.10. The darker cylinder and the corresponding darker direction represent the ground truth cylinder.

5.3 Reconstruction of Cutting Edges using Deep Autoencoder

As described in chapter 2, the reconstruction of cutting edges is an important challenge. Reconstruction is a necessary intermediate step in determining parameters for the cutting edge[2]. Further information can be found in the chapter 2. A method for creating an ideal reference edge using model-based methods was presented in section 4.3.

---

[2] if no CAD data is available for the corresponding cutting edge

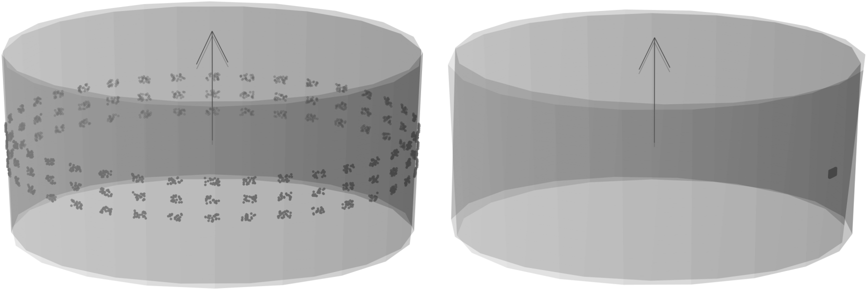


Figure 5.10: Two fitted cylinders and the corresponding point clouds are shown. In the upper figure, 108 measuring areas were applied for fitting, while in the lower figure a single measuring area was used. The darker cylinder and the arrow indicating the ground truth. The brighter illustrated of the cylinder and the arrow represent the fit using a machine learning model. For better comparison, the center has been moved slightly upwards. This is not present in the actual result.

As also described in section 4.3, no real measurements of cutting edges are used for validation due to legal restrictions. Due to these legal restrictions, the generation of simulated cutting edges described in section 4.3.1 is used for training, validation and testing. Section 5.3.2 describes the idea and the deep learning methods used. In section 5.3.3 the results are validated against simulated data. Starting with a description of the methods used for deep learning in section 5.3.1.

5.3.1 Preparation of the Data and Idea Description for using Autoencoder for Reconstruction

The measurement data for cutting edges are often provided with a high point density. This is necessary to determine the parameters for the cutting edge with micrometer accuracy. For example, a measurement can have over 500,000 points. If a lot of training and validation data is generated, this can quickly lead to a full memory[3]. In order to provide more training and validation data, the size of the original image was reduced by resizing. All simulated measurements of cutting edges were changed to a grid size of 476×476 using the library PIL in Python. The ratio was always the same, so there was no stretching along one direction. The remaining areas, since the measured areas are in

---

[3] with today's technology (2020)

general not quadratic, were filled with the number zero. Furthermore, the image area of the functions has been normalized to a range between 0 and 1. This is important for machine learning methods, as no different value ranges need to be learned for the images. If real measurement data are used, it is important that no empty measurement point exist. Technically, there must not be a value with the label “Not a Number”. To solve this problem, the method for filling the “Not a Number” areas from section 4.3.1 can be used. Here piecewise hermite interpolation is used, which leaves all other measured values unchanged. In summary, a flow chart can be seen in figure 5.11 on how to prepare the training, validation and test data.

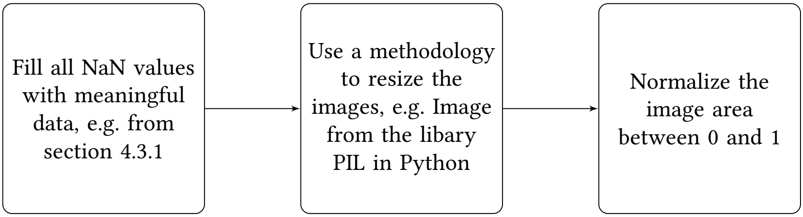

Fill all NaN values with meaningful data, e.g. from section 4.3.1
Use a methodology to resize the images, e.g. Image from the libary PIL in Python
Normalize the image area between 0 and 1

Figure 5.11: Necessary steps before a machine learning model can be applied.

Machine learning models for cutting edges were also realized, which keep the original grid size of the measurements unchanged. The results are shown in appendix B.

5.3.2 Architecture of the used Deep Autoencoder for generating ideal Cutting Edges

The reconstruction for cutting edges is mainly used for cutting edges with defects. The realization of this required reconstruction can be realized by using autoencoders, see section C.8. However, these have the original goal of copying the input to a corresponding output. So if an image information such as cutting edges are learned, the basic idea of the autoencoder would be to try to copy these cutting edge. The challenge here is to load a cutting edge with defects as an input image and create an ideal reference as an output image. Thus an autocoder should be trained that knows the shape and structure of the cutting edge so well that it can be reconstructed and on the other hand so badly that the defects cannot be reconstructed. Here the idea of PCA, see section C.2, is essential. Reduce the information to the most necessary characteristics and try to reconstruct the whole image from them. A good explanation is shown in figure C.1. There it is shown that a number of key properties lead to a better reconstruction. The autoencoder uses the same idea. It tries to reconstruct the image from its bottle neck, see section C.8 and figure C.12. The biggest difference to PCA is that autoencoders can also learn non-linear functions and therefore can be better used for more complex tasks. If the bottle neck is selected to be large, it is very likely that the defects at the output of an autoencoder will also be reconstructed. If, on the other hand, the bottle neck is selected too small, a reconstruction of structure and shape is no longer possible. If a good bottle neck size is chosen, shape and structure can be reconstructed, but not the defects. As shown in section C.12, different approaches can be taken to implement an autocoder. One of them is the method of tying weights, where only in encoding direction the weights are trained and in decoding direction the weights are transposed. An alternative approach would be to learn the weights for encoding and decoding separately. The results for the methodology of separately learned weights for encoder and decoder are presented in section 5.3.3. A further distinction in autoencoding can exist in the design of the decoder. In addition to the so-called convolution transposed, upsampling can also be used. Upsampling was here used in section 5.3.3. In summary, this thesis distinguishes the following cases for this task:

- Tied weights using convolutional transposed for decoding, see appendix B.
- Trained weights for encoding and decoding. Use of upsampling for decoding, see section 5.3.3.

The structure of the autoencoder is very similar to the schematic model in figure 3.7. The table 5.3 on the last page of this chapter 5 shows the settings used. In addition, the implemented autoencoders can be found in my Git repository [83]. This git repository also contains the generation of training, validation and test data. The only requirement is the existence of a sample profile of a cutting edge.

5.3.3 Application and Validation of an Autoencoder Approach to Reconstruct Simulated Cutting Edges

To validate the deep learning approach described in section 5.3.2, cutting edges with different defect sizes were simulated. It is important to note that for legal reasons this thesis only represents simulated cutting edges. However, the results could also be confirmed for real measured data. These were reliably reconstructed to determine the defect parameters. A comparison between both approaches from section 4.3 and 5.3.3 will be evaluated in section 6.3. To validate the deep learning approach, the training and validation data are generated using algorithm 4.4 for two different types of cutting edges. Both types of cutting edges and their ideal references are shown in figure 4.7. The most noticeable point here is the different curvature of the cutting edges. While one cutting edge has a “straight-line” curvature along the y-direction, the second cutting edge has a “non-straight-line” curvature along the y-direction. Another difference is their profile along the $E^{(xz)}$ plane. Here, the profile for the cutting edge with the straight line in y-direction is more complex than the cutting edge with a stronger curvature along the y-direction. The generated test data have defects in comparison to the training and validation data. These defects are very similar for the two simulated cutting edges shown. In total, four artificial defects are generated for the test data, which differ in depth and length. Furthermore, the test data has uniformly distributed noise between $\pm 2\,\mu\mathrm{m}$. The results are shown in figure 5.12 through figure 5.15. The two figures from 5.12 and 5.13 shows the results for the cutting edge with a straight cutting edge direction along the y-direction. The simplified name type A is used. The other two figures 5.14 and figure 5.15 refer to the cutting edge with the stronger curvature along the y-axis. This cutting edge will be called type B. In order to get an identical validation and interpretation of the results, all the figures from 5.12 to 5.15 have the same structure. These have already been described in section 4.3.3. At this point they are briefly repeated:

- The figure above shows the desired cutting edge, once without and once with noise.
- The second figure shows the desired cutting edge with defects.
- The third figure shows the reconstruction with the algorithm described in section 5.3.2.
- The last figure shows the difference between the generated reference and the desired cutting edge with defects.

It is shown that the reconstruction for cutting edge types A and B works well. These results are largely unaffected by existing noise. A determination of the parameters is possible and thus a decision whether a reworking of the cutting edge can be made. It is important to mention that one model was trained with both cutting edge types. Therefore, the weights learned are applicable to both cutting edge types.

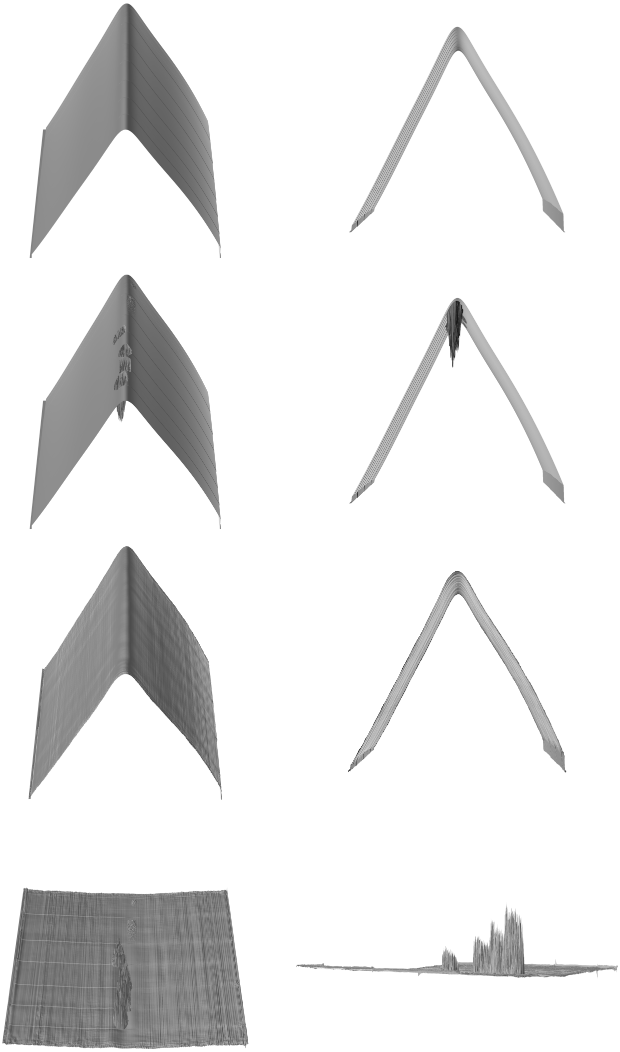


Figure 5.12: Shown is a generated cutting edge of **type A without** additional noise. Furthermore this generated cutting edge is shown once without and once with simulated defects. Also shown is the calculated ideal reference and the difference to the cutting edge with simulated defects.

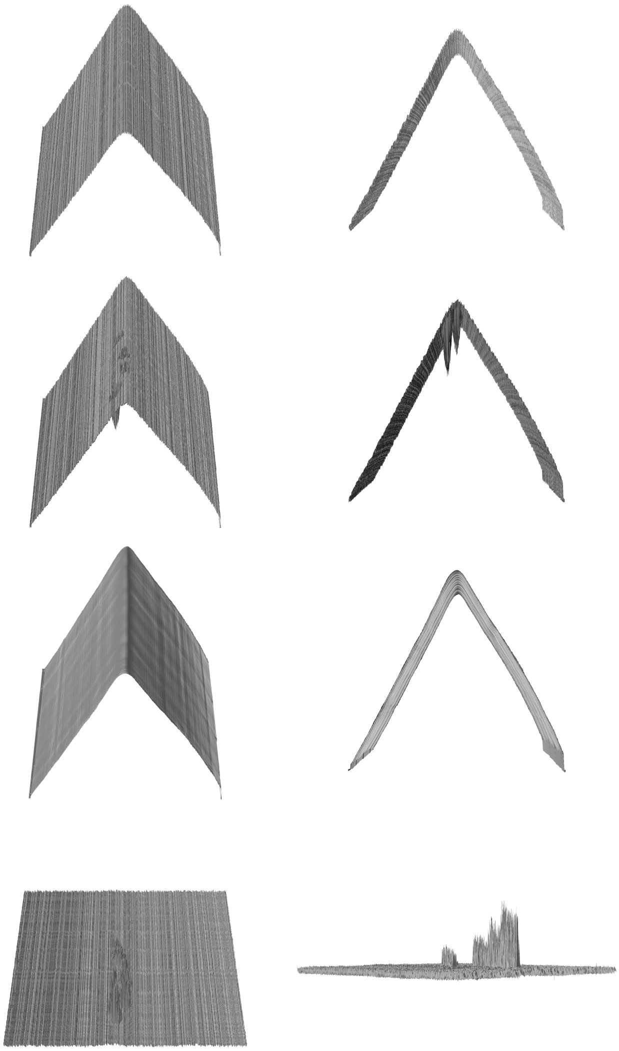


Figure 5.13: Shown is a generated cutting edge of **type A with** additional noise. Furthermore this generated cutting edge is shown once without and once with simulated defects. Also shown is the calculated ideal reference and the difference to the cutting edge with simulated defects.

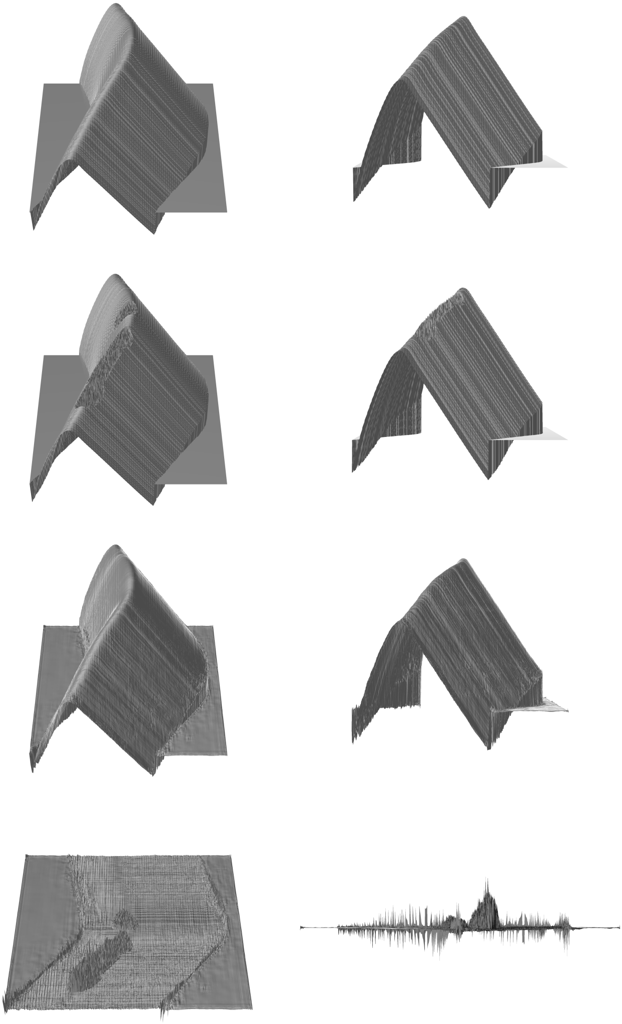


Figure 5.14: Shown is a generated cutting edge of **type B without** additional noise. Furthermore this generated cutting edge is shown once without and once with simulated defects. Also shown is the calculated ideal reference and the difference to the cutting edge with simulated defects.

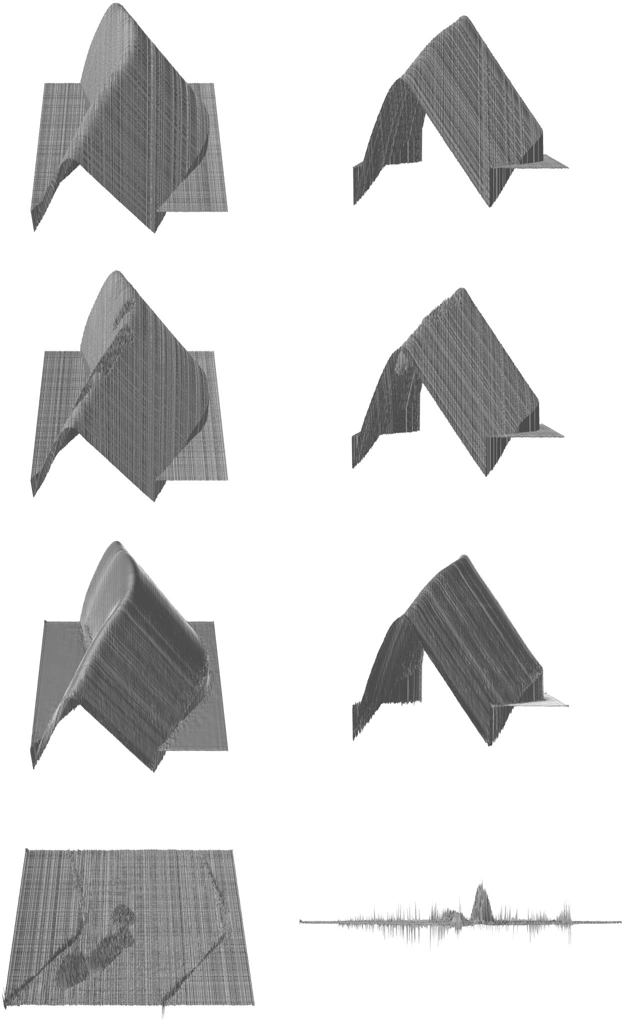


Figure 5.15: Shown is a generated cutting edge of **type B with** additional noise. Furthermore this generated cutting edge is shown once without and once with simulated defects. Also shown is the calculated ideal reference and the difference to the cutting edge with simulated defects.

Table 5.2: Besides the parameters used for data generation, the table also shows the deviations of the parameters $r^{(fit)}$ and $\mathbf{w}^{(fit)}$. An interval was specified in which the difference between the ground truth and the predicted values is given. The mean square error MSE is also noted.

Settings			
Total number of measuremet areas	$n^{(Datapoints)}$ per measurement area	$n^{(circumference)}$ per height adjustment	Total number of selected heights
108	225	36	3
72	324	36	2
36	625	36	1
18	1296	18	1
9	2500	9	1
4	5625	4	1
1	22500	1	1

Radius	$\mathbf{r}^{(real)}, \mathbf{r}^{(fit)} \in \mathbb{R}^{20}, \quad ERR = \text{abs}\left(\mathbf{r}^{(real)} - \mathbf{r}^{(fit)}\right) \in \mathbb{R}^{20}$		
Total number of measuremet areas	Interval of ERR	Average of ERR	$MSE(\mathbf{r}^{(real)}, \mathbf{r}^{(fit)})$
108	$\left[1.528 \times 10^{-5}, 1.953 \times 10^{-3}\right]$	1.128×10^{-3}	1.578×10^{-6}
72	$\left[1.297 \times 10^{-4}, 2.281 \times 10^{-3}\right]$	1.231×10^{-3}	1.880×10^{-6}
36	$\left[1.033 \times 10^{-5}, 1.059 \times 10^{-3}\right]$	4.268×10^{-4}	2.630×10^{-7}
18	$\left[1.044 \times 10^{-5}, 1.061 \times 10^{-3}\right]$	5.216×10^{-4}	3.657×10^{-7}
9	$\left[4.852 \times 10^{-6}, 1.102 \times 10^{-3}\right]$	4.045×10^{-4}	2.692×10^{-7}
4	$\left[9.940 \times 10^{-6}, 1.482 \times 10^{-3}\right]$	7.263×10^{-4}	7.106×10^{-7}
1	$\left[8.900 \times 10^{-6}, 1.040 \times 10^{-3}\right]$	4.560×10^{-4}	1.132×10^{-7}

Direction	$\mathbf{W}^{(real)}, \mathbf{W}^{(fit)} \in \mathbb{R}^{20 \times 3}, \quad ERR = \text{mean}\left(\text{abs}\left(\mathbf{W}^{(real)} - \mathbf{W}^{(fit)}\right)\right) \in \mathbb{R}^{20}$		
Total number of measuremet areas	Interval of ERR	Average of ERR	$MSE(\mathbf{W}^{(real)}, \mathbf{W}^{(fit)})$
108	$\left[2.421 \times 10^{-5}, 1.173 \times 10^{-3}\right]$	6.773×10^{-4}	8.858×10^{-7}
72	$\left[8.104 \times 10^{-4}, 1.756 \times 10^{-3}\right]$	1.229×10^{-3}	2.358×10^{-6}
36	$\left[8.673 \times 10^{-6}, 1.051 \times 10^{-3}\right]$	4.485×10^{-4}	4.611×10^{-7}
18	$\left[2.209 \times 10^{-5}, 8.893 \times 10^{-4}\right]$	3.901×10^{-4}	2.835×10^{-7}
9	$\left[6.234 \times 10^{-5}, 8.104 \times 10^{-4}\right]$	4.485×10^{-4}	3.017×10^{-8}
4	$\left[4.223 \times 10^{-5}, 6.168 \times 10^{-4}\right]$	3.346×10^{-4}	2.199×10^{-7}
1	$\left[2.979 \times 10^{-4}, 1.344 \times 10^{-3}\right]$	7.609×10^{-4}	7.325×10^{-7}

Table 5.3: The architecture of the autoencoder model is shown in table form. As padding the parameter "same" and as activation function $\varphi(x)^{ELU}$ was used, see section 3.3.1. For the output layer the activation function $\varphi(x)^{sig}$ was used.

Settings	Used settings	
Encoder Design		
Input Layer	Size $476 \times 476 \times 1$	
Convolution Layer E-0	Filter size 16	Kernel size 3×3
Max Pooling E-0	Pool size 2×2	
Convolution Layer E-1	Filter size 16	Kernel size 3×3
Max Pooling E-1	Pool size 2×2	
Convolution Layer E-2	Filter size 32	Kernel size 3×3
Max Pooling E-2	Pool size 2×2	
Convolution Layer E-3	Filter size 32	Kernel size 3×3
Max Pooling E-3	Pool size 2×2	
Convolution Layer E-4	Filter size 64	Kernel size 3×3
Max Pooling E-4	Pool size 2×2	
Decoder Design		
Convolution Layer D-0	Filter size 64	Kernel size 3×3
Upsampling D-0	Pool size 2×2	
Convolution Layer D-1	Filter size 32	Kernel size 3×3
Upsampling D-1	Pool size 2×2	
Convolution Layer D-2	Filter size 32	Kernel size 3×3
Upsampling D-2	Pool size 2×2	
Convolution Layer D-3	Filter size 16	Kernel size 3×3
Upsampling D-3	Pool size 2×2	
Convolution Layer D-4	Filter size 16	Kernel size 3×3
Upsampling D-4	Pool size 2×2	
Output with convolution Layer	Filter size 1	Kernel size 3×3

6 Comparing model-based and machine learning approaches

This chapter compares the approaches described in chapter 4 and chapter 5 with the challenges described in chapter 1 and chapter 2. In addition to the interpretation of the results, an essential point of comparison is the evaluation of artificial intelligence approaches. In order to be able to draw a conclusion in this context, it is important to analyse the results under different aspects. All subsequent sections in this chapter are structured in such a way that first the results of the model-based method and then the machine learning method are summarised. As comparison criteria, the following points are contrasted in each section.

- comparison in terms of results
- comparison in terms of degree of automation
- comparison in terms of user-friendliness
- comparison in terms of future potential

It is important to note that this assessment and evaluation only covers a very small area of metrology and no general statement can be made. The topics covered here reflect a part of metrology that deals with a resolution limit in the micrometre range.

The next section 6.1 deals with micro lead, more precisely with the determination of the texture direction Θ^t of a ground surface.

6.1 Texture Direction for Micro Lead

An important challenge in determining the texture direction for micro lead is to achieve a high reproducible resolution accuracy. Since even a deviation of a few arc minutes in the texture direction Θ^t of ground surfaces can lead to a false statement for micro lead, a resolution accuracy of less than one arc minute is desirable. First, the results for the model-based method are summarised.

Summary model-based method The approach shown in section 4.1 achieves a resolution accuracy of the texture direction Θ^t of less than one minute of arc, without additional interpolation methods. The maximum deviation for simulated surfaces of the investigated cases is $0.4'$, see also figure 4.2. This high resolution accuracy could also be demonstrated on real measurement data. In fact, the standard deviation determined for the measured shaft was smaller overall than the method of P. Arnecke, which also achieves a resolution accuracy of less than one minute of arc. An equally important point is that the cross-correlation method in section 4.1 described has been shown to be able to determine texture direction Θ^t with an accuracy of one minute of arc, regardless of the area in which it lies. For real measurement data, for example, an average texture direction Θ^t of about $15'$ was easily determined. For simulated measurement data, texture directions Θ^t between $-6'$ and $+6'$ were generated and could also be determined. This approach thus has very good accuracy and high applicability, which makes the solution very user-friendly. However, when looking at the simulated results of the approach with cross-correlation, an increase in the deviation from the ground truth can be seen. This situation is illustrated in figure 6.1. This figure 6.1 shows the difference of the respective texture direction to the ground truth as a dependency of the ground truth.

The results for the real measurement sample can be seen in figure 4.5. There, a mean texture direction of $15.53'$ with a standard deviation of $1.81'$ was determined for 20 measurements of a shaft.

Summary Machine Learning Method The machine learning based solution, described in more detail in section 5.1, showed good texture direction resolution accuracy for the learned regions for simulated ground surfaces. For the learned texture region of $\pm 6'$, a resolution accuracy of $\pm 1'$ could be achieved as long as the test data is also in this texture direction region between $\pm 6'$. However, if the texture direction Θ^t is outside the learned texture direction, such as $15'$, the texture direction is only determined with a high texture deviation and can no longer be reliably tested for leakage. This situation is shown in figure 5.5. To investigate this problem in more detail, another convolutional neural network was trained with training and validation data between $\pm 12'$. Here, using the same chosen architecture of the convolutional neural network as for learning the range of $\pm 6'$, the texture directions Θ^t could be determined with a resolution accuracy of $\pm 2'$, although there is still potential for optimisation here. The visualised results for this can be seen in Figure 5.7. However, if the texture direction is outside the learned range here too, the specification of the texture direction is no longer possible.

In order to obtain a reliable statement about the micro lead angle Ψ and a possible leakage with the described machine learning approach, the tool axis can first be determined. With the help of the equation 1.1, the determined tool axis direction can then be corrected and transferred to the measured ground surface. Here, a fixed learned texture direction range of $\pm 12'$ corrected by the tool axis with a resolution range of $\pm 2'$ could be sufficient to make a statement about leakage. This statement is based on [3], there it is stated that leakage occurs from a texture direction larger $10'$ or smaller then $-10'$ and can possibly occur between a texture direction of $\pm 5'$ to $\pm 10'$. If the texture direction of the ground surface is within $\pm 5'$, leakage will not occur. In general, however, this method is expensive, so it is better to learn an extended texture direction range that can determine a much larger texture direction with a high resolution accuracy.

Comparison related to results A comparison of the texture direction results for ground surfaces between the two approaches is first considered for the simulated texture direction between $\pm 6'$. Here it can be seen in figure 6.1 that both approaches were able to achieve a texture resolution of less than one minute of arc. This result reflects a good resolution accuracy. Furthermore, it can be seen that the absolute mean value for the machine learning approach to ground truth of $0.312'$ is on average higher than for the model-based approach, which has a absolute mean value to ground truth of $0.225'$. The same observation also applies when determining the standard deviation. This is $0.255'$ for the model-based approach and $0.386'$ for the machine learning approach. However, when looking at the data, it is clear that there is a correlation between the simulated surfaces based on an inverse Radon transform and the model-based approach based on a cross-correlation. This correlation is also briefly discussed in section 4.1.2. Nevertheless, it is noticeable that the deviation from the ground truth in the model-based approach is higher at the edges than in the area of the zero texture direction. With the machine learning approach, this is scattered.

Since the machine learning approach could not be tested on real samples, a statement about the practicability of this approach is missing here. The machine learning method for determining the texture direction is trained using simulated data, since the texture direction and the associated ground truth of real surfaces in this quantity are often difficult to determine or can only be determined with expected high deviation. In comparison, the method based on the model-based approach gives good results for real surfaces, see figure 4.4.

Through the validation on simulated and real surfaces, it can be summarised that the results for determining the texture direction with a model-based approach are to be preferred over the presented machine learning architecture.

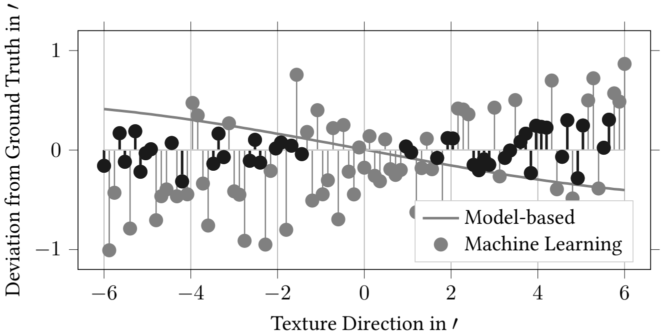


Figure 6.1: Shown is the deviation of the predicted texture direction values from the ground truth for simulated textures of a ground surface. The solid line shows the deviation of the model-based solution. The discrete points show the deviation of the machine learning approach. If a discrete value is grey, the model-based solution shows a smaller absolute deviation from the ground truth than the machine learning approach. If it is black, the deviation for the machine learning approach is smaller.

Comparison related to degree of automation Ideally, after measuring a real ground surface, the texture direction is determined and output directly afterwards. This is possible with both approaches as the algorithm can be applied after some pre-processing of the surface. Detailed information on pre-processing can be found in appendix A. However, both approaches have disadvantages in terms of automation.

With the model-based approach, this is the duration of several minutes until the texture direction can be determined. So with a solution implemented in software, this would probably be started externally by the user. With the machine learning-based approach, the disadvantage in terms of a high degree of automation is that a neural network must first be trained with data. However, once a successful model and the associated weights are stored, the determination of the texture direction can be directly displayed to the user without much time expenditure and thus has a possible high degree of automation. Furthermore, the neural network can also be based on training data that does not require pre-processing, and a pre-trained model can be stored in any case.

Comparison related to user-friendliness The ease of use is reflected in this comparison not only by the degree of automation, but also by the circumstances and settings necessary to obtain the result at the end. One of the most time-consuming tasks is aligning the workpiece to measure the texture direction of the ground surface. This is the same with both methods and does not increase the ease of use with either method. In addition, no further adjustment parameters are required with either method. Since the time advantage of machine learning has already been mentioned in the automation comparison section, no advantage for either approach can be identified for this ease-of-use comparison. Both solutions can be designed to be very user-friendly in their current implementation.

Comparison related to future potential It is also important how much future viability the presented solution has and how much potential the individual approaches still have. The future viability of the model-based approach is very high because the results are very good and validated not only for simulated but also for real measurement samples. Furthermore, the model-based approach has a high degree of automation and good user-friendliness. A possible point of improvement would be the speed of the algorithm, which is already very efficiently programmed and parallelised in its current implementation. For the machine learning approach, not only the validation on real measurement samples is missing, but also the validation of the dimensional dependence of the measured ground surfaces. This can vary from institute to institute or company to company. If these two problems are investigated in future work and positively evaluated for feasibility, the machine learning approach would be able to determine the texture direction through the learned model within seconds and would also have the potential to achieve a resolution of less than one arc minute.

Evaluation of the comparisons After considering the various aspects of the results, the degree of automation, the user-friendliness and the future potential, the model-based solution is to be preferred. Not only has it been successfully tested on real and simulated data, but it also has a lower standard deviation in the evaluation of real data compared to other model-based solutions, such as the one based on the Radon transformation by P. Arnecke.

6.2 Cylinder Fitting for Micro Lead

When determining the micro lead angle Ψ, in addition to the texture direction Θ^t, the determination of the tool axis is also elementary. Since the micro lead angle Ψ must be determined with arc minute accuracy, the tool axis must also be determined very precisely. To determine the tool axis, first the results of the model-based method and then the machine learning results are combined before the methods are compared.

Summary model-based method The model-based approach requires an initial parameter for the fast convergence of an optimal solution for fitting a cylinder to the simulated measurement data. An initial estimator can also be presented here, which no longer requires the input of an initial parameter. This initial parameter is acceptable in this application for determining the tool axis in micro lead angle Ψ determination, since the ideal nominal direction of the tool axis is often known due to the fixation of the workpiece.

The results were determined for the fitting of a cylinder for a maximum of 108 simulated measuring surfaces, which in their measuring area reflect the described setting of the measuring device used from the section 1.3.2. In summary, the results for the determination of the radius and the tool axis reflect good results for up to 4 simulated measuring surfaces. For the model-based approach, the best results in terms of mean square error were obtained for 9 measurement surfaces, although this result should be viewed with caution overall. Due to the hardware limitation, it was partly not possible to load several simulated surfaces with a high point density. It can therefore be assumed that better results are possible for more simulated surfaces with comparable point density. The comparison with a single simulated measuring surface did not provide good results for the model-based approach when determining the tool axis, especially when determining the radius. A detailed presentation of the results can be found in the table 4.1.

Summary Machine Learning Method Analogous to the model-based procedure, results for fitting a cylinder were determined for a maximum of 108 simulated measuring surfaces, whose measuring area reflects the described setting of the measuring device in section 1.3.2. In contrast to the generated results of the model-based solution, this approach did not require an initial state of the tool axis. In addition, significantly fewer measuring points could be used for a measuring area, as the hardware used was not sufficient for a higher point density. In summary, the results for determining the radius and the tool direction reflect good results for all simulated measuring areas. When using the machine learning approach, the results for only one simulated measuring area were not significantly worse than for several simulated measuring areas. This result is a consequence of the existing knowledge. The training data was generated in such a way that when the workpiece is adjusted, the centring and tilting properties lie within a defined range. Thus, similar to Bayesian estimation, a priori knowledge is learned. This means that the adjustment cannot be significantly worse than the prior knowledge used. A detailed presentation of the results can be found in the table 5.2.

Comparison related to results In summary, a comparison shows that both methods give similarly good results. Figure 6.2 and figure 6.3 show the fitting results once for the radius and once for the tool direction. In both figures, the deviation for fitting with model-based methods is shown on the left and the deviation with machine learning methods is shown on the right. The visualised point shows the mean value of the deviation and the line shows the interval of the deviation. The radius for the fit with the model-based method for a measuring area is not shown because the deviation is much larger and no longer lies within the range shown. If the results are now compared with each other, a smaller scatter can be seen for the model-based approach when determining the radius and the tool direction. Overall, the results for radius and tool direction are also closer to the ground truth, whereby the determined mean value was used. In summary, the results with the model-based approach show better accuracy.

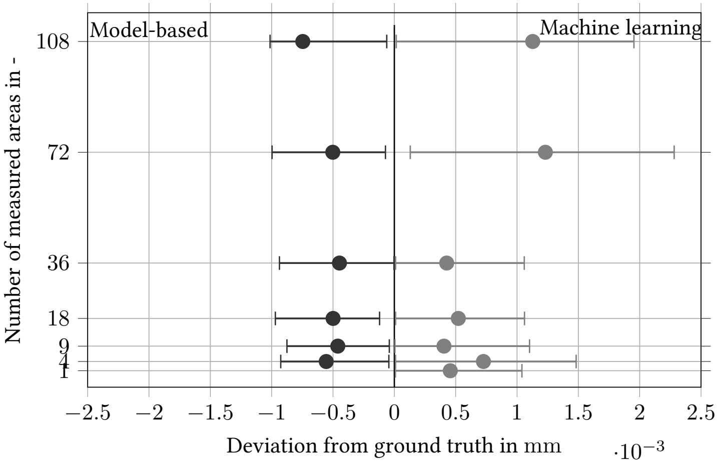


Figure 6.2: Shown is the comparison of a cylinder fit. The ordinate shows the number of measuring areas used, see figure 4.6. The deviation of the predicted value from the ground truth is shown on the abscissa. On the left side the deviation of the radius for the adaptation of a cylinder with the described methodology of section 4.2 is shown. The result for one measuring area is missing because the deviation is far outside the shown deviation range. On the right side the deviation of the fitted radius with the machine learning methods from section 5.2 is shown.

Comparison related to degree of automation Both points have disadvantages in terms of automation. While the presented solution of the model-based method has a dependency on the end user due to the input parameter, the acquired number of point densities is a challenge for the machine learning used model and is not yet generalisable in the current solution. In the training phase, the number of measured or simulated features may lead to insufficient memory of the used graphics processor

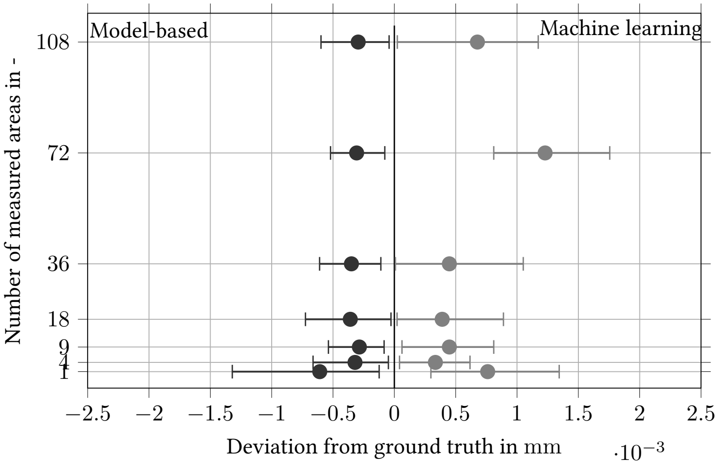


Figure 6.3: Shown is the comparison of a cylinder fit. The ordinate shows the number of measuring areas used, see figure 4.6. The deviation of the predicted value from the ground truth is shown on the abscissa. On the left side the deviation of the cylinder direction for the adaptation of a cylinder with the described methodology of section 4.2 is shown. The result for one measuring area is missing because the deviation is far outside the shown deviation range. On the right side the deviation of the fitted cylinder direction with the machine learning methods from section 5.2 is shown.

and therefore a lower point density has to be used to reduce the number of features[1]. However, in the presented model-based solution, the input parameter is the smallest challenge. Especially in a lead analysis, since here the axis direction usually points in a known and fixed direction. In addition, the machine learning approach used so far has the disadvantage that a known radius is learned in the training data, which makes an automated solution difficult, since this is also present in the training data. Thus, according to the current state of art, a higher degree of automation can be achieved with the model-based methodology.

Comparison related to user-friendliness The usability of the model-based solution is good. The only limitation is the input parameter, which can even be made redundant by implementing an initial estimator. Thus, the tool axis can be derived quickly. In addition, the model-based approach prefers several measurements along the circumference and

---

[1] in 2020

possibly at different heights of the shaft to make a reliable prediction. However, this is also recommended when determining the texture direction and thus hardly contributes to a negative evaluation in terms of user-friendliness.

User-friendliness has several disadvantages in machine learning. In a first step, training data of the desired shaft are necessary, which have to be simulated as measuring them is very costly. In addition, with real measurement data, labelled quantities can also only be determined in advance using model-based approaches. In addition, this simulation method is only well suited with regard to the tool axis for micro lead applications, as this is often known. The situation is different with the radius of the shaft, which can vary greatly for different shafts. Therefore, a pre-trained machine learning model cannot simply be provided. The negative dependence on the number of measurement points and measurement surfaces, which has already been mentioned for the current machine learning approach, also makes it difficult to use arbitrary training data. To keep the quality of the results in the presented machine learning solution high, a model trained by the end user is required. Once the model is trained and stored with pre-trained weights, determining the tool axis and radius is very easy and user-friendly. Therefore, the end user must be openly told what kind of training data has been generated to obtain the pre-trained model. This is especially important as future predictions will be based on this pre-trained model.

In summary, the model-based approach provides the better user experience.

Comparison related to future potential To reliably prepare the presented model-based approach for future applications, two modifications to the current solution are proposed. An initial estimator so that fitting can be expected without end-user input, and a solution based on prior knowledge and thus using Bayes' theorem. If these two requirements are met, realisation for future applications is guaranteed. To future-proof the machine learning approach, a neural network must be trained on the basis of a defined number of measurement surfaces and measurement points. Furthermore, training data must be used that also have different radii so that these radii can also be learned. This feature makes it possible to use the machine learning approach to adapt a cylinder for lead measurements in a broader spectrum.

After the improvements have been realised in both approaches, both can be used for future applications without hesitation.

Evaluation of the comparisons After weighing various aspects of the results, the degree of automation, the user-friendliness and the future potential, the model-based solution is to be preferred. This is also made clear by the fact that the model-based solution solves an optimal problem and the machine learning-based solution aims for an optimal solution by the ADAM optimiser. Thus, the machine learning method used in this thesis approximates against the optimal solution of the model-based approach. However, the advantage of machine learning is that the independence of an initial estimator can be easily obtained from the training data and also an a priori knowledge is automatically learned from the data, which is reflected in the result for the radius of a simulated measurement area.

6.3 Reconstruction of Cutting Edges

Reconstruction is a widespread field of work for various applications. The reconstruction of cutting edges is complicated by their special and different shapes. For example, sometimes larger defects have to be reconstructed along a curved cutting edge. The flanks of an edge can also vary greatly depending on the cutting edge type, which makes reconstruction difficult. This section 6.3 first summarises the results of the model-based method and then the machine learning results before comparing them.

Summary model-based method TBasis for the reconstruction with model-based methods is the Gaussian filter described in section 3.1.1. The Gaussian filter ensures that defects are interpreted as outliers and that the shape of the cutting edge takes an ideally approximated course. The model-based approach gives particularly good results for cutting edges that have defects that do not follow a curved cutting edge. It should be mentioned here that the reconstruction often provides better results than the state of the art solution used in the industry, as the flanks are also reconstructed very well. Only in the case of strong defects along a curved cutting edge shape, the approach does not provide optimal results. This is because the shape and the defects are interpreted as outliers and thus the shape of the cutting edge cannot be ideally followed. Nevertheless, it can be summarised that the presented model-based approach provides very good results for many types of cutting edges.

Summary Machine Learning Method The machine learning approach uses an autoencoder to reconstruct the cutting edge. Here, the autoencoder and thus its bottleneck is trained with intact cutting edges, so that the learned weights reproduce the shape of the cutting edge, but at the same time are not able to reconstruct the defect of this cutting edge. This approach shows very good results for all types of cutting edges, so that for test data that had defects, all defects could be reconstructed. Furthermore, several cutting edge shapes could be learned and applied with one model. A disadvantage of this method is that a lot of training data is needed for the reconstruction and this must either be available or created. However, it is unlikely that many cutting edge measurements will be available for a cutting edge shape, as these measurements are often very time and labour intensive. Therefore, an approach to generating cutting edge data has been presented where, in addition to the profile of the cutting edge, the shape must also be defined as an input parameter. However, both can also be replaced by reading in one cutting edge measurement. This then generates cutting edges with a similar shape and enables the training of a suitable autoencoder. The results of this approach can be summarised as very good, as all desired results could be achieved.

Comparison related to results A comparison of the two approaches for different cutting edges is contrasted in figure 6.4 and 6.5. Figure 6.4 shows the curve for the defined type A of a cutting edge without stronger contour shapes. Figure 6.5 shows an cutting edge with stronger curvature.

Looking first at figure 6.4, it can be seen that both approaches achieve a very good reconstruction. In both approaches the defects are reconstructed and thus a difference model between the ideal and the measured cutting edge can be used to extract the corresponding parameters for the evaluation of the defect. Even though both approaches fulfil this objective, the ideal reference cutting edge generated by the model-based approach is more similar to a generated CAD model due to the very smooth reconstructed surface. With the machine learning approach, individual smaller noise components could not be smoothed as perfectly.

For the cutting edge in figure 6.5, the defects could be reconstructed for the machine learning approach in such a way that a difference model for determining the parameters can be derived from the generated ideal reference geometry. Although the approach with model-based methods enables an ideal reference geometry and the evaluation of individual defects, the derived parameters cannot be determined with high reliability. Nevertheless, it can be seen that the model-based approach produces a very smooth surface that resembles a generated CAD model. With machine learning, the surface is still subject to slight noise, similar to the evaluation of the Type A cutting edge. In addition, the dependence on the training data is clearly visible with the machine learning approach, as individual profile planes are clearly recognisable.

In summary, the machine learning approach gives better results, as any kind of major defects can be reconstructed along different cutting edge types. Nevertheless, the solution with the model-based approach also shows a very good reconstruction. The case shown in figure 6.5 describes one of the most difficult reconstruction cases.

Comparison related to degree of automation The degree of automation of the model-based methodology is high, as it can be applied without major adjustments. Nevertheless, there are dependent parameters that can influence the result. Although these parameters are well designed in their default configuration, sometimes better settings can lead to a better result and thus influence an automation procedure. The most important adjustable parameter is the factor for the robustness of the Gaussian filter and the resulting correction of the defects. If the robustness is set too high, the contour of the cutting edge can no longer be followed. If the robustness is too low, the generated contour follows the defects too closely.

Similar to the model-based approach, a high degree of automation is also possible with the machine learning approach. Although a model must first be trained for a cutting edge type, the reconstruction can then be automated with the pre-trained model. It should be mentioned that the cutting edges were reduced to a special dimension in order to train the model accordingly for different cutting edge types. The edge fields are filled with zero values so that different shapes of cutting edges can be trained at once. However, this has no negative influence on the determination of the defects and their parameters.

Comparison related to user-friendliness In terms of usability, the model-based approach is only limited by the change of possible parameters. However, changing the parameters is most likely not necessary, so that a “click-and-realise” solution for generating an ideal reference geometry can be generated.

In the machine learning solution, the point of model creation has already been discussed and considered in the point of degree of automation. There are no further restrictions for this approach, so that a “click-and-realise” solution is also possible.

Thus, a high degree of user-friendliness is guaranteed for both approaches.

Comparison related to future potential The future viability of the model-based solution for reconstructing cutting edges can be rated highly, as this approach in particular has delivered very good results not only for simulated but also for real cutting edges. However, the approach for cutting edges can still be improved, as shown in 6.5. Also, there are some samples that are much more damaged than the cutting edge shown in 6.5. These samples are so badly damaged that some of the flanks along the entire sample are also badly damaged. This makes it impossible for this method to define and reconstruct the areas required for this algorithm. So this approach is feasible for many cutting edges, but not for all.

The situation is different for cutting edges that are reconstructed by machine learning. Since the training data is trained on the basis of intact samples, the contour can always be learned and thus the defect can also be reconstructed. This makes this approach very promising for the future, as it can also be used positively for very complex cases.

Evaluation of the comparisons After weighing various aspects of the results, the degree of automation, the user-friendliness and the future potential, the machine learning-based solution is to be preferred. This is also made clear by the fact that the architecture of the machine learning solution only learns undamaged cutting edges and thus the shape of the cutting edge is stored in the efficient representation of the autoencoder. Consequently, the learned intact cutting edge data can also be used to reconstruct stronger defects along the cutting edge. Furthermore, the architecture of the autoencoder has made it possible to learn multiple cutting edge types with one training.

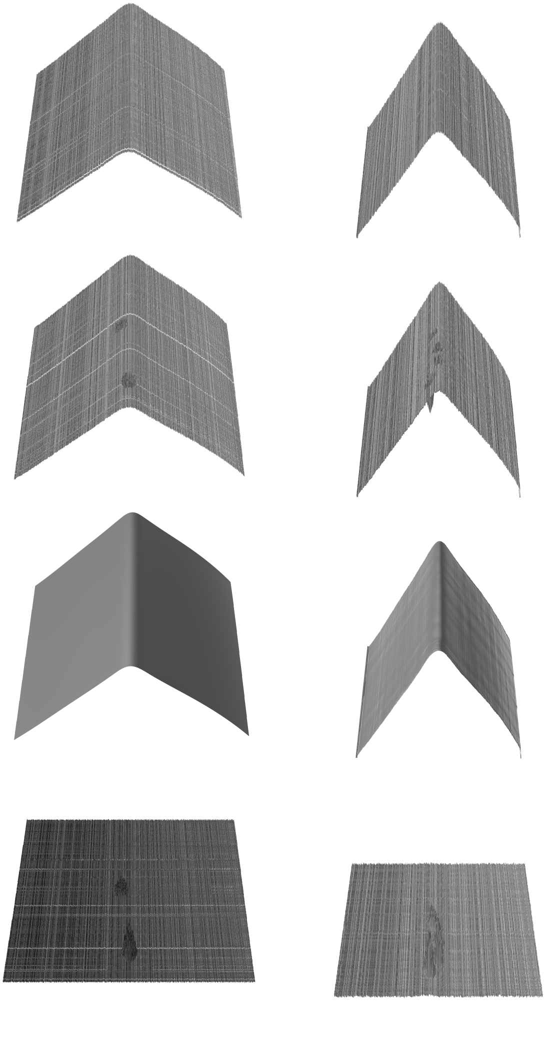


Figure 6.4: Shown is the reconstruction of an ideal cutting edge. Once on the left with the approach described in section 4.3 and once on the right with the machine learning approach described in section 5.3. Above the ideal reference is shown overlaid with noise. Below this figure the ideal noisy reference with defects. Further down the reconstruction of the corresponding methodology and at the bottom the difference between the reconstructed reference and the reference with defects.

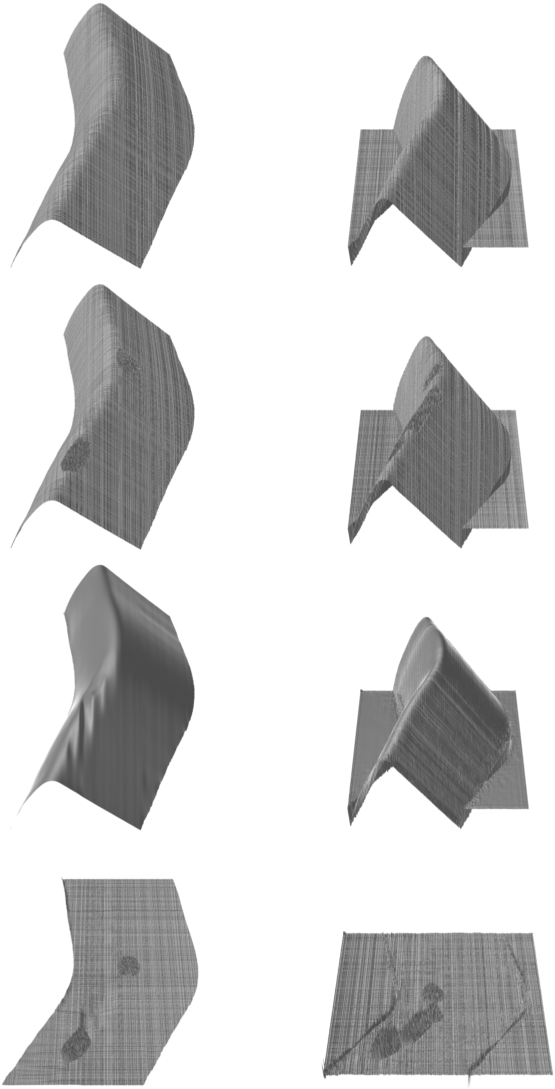


Figure 6.5: Shown is the reconstruction of an ideal cutting edge. Once on the left with the approach described in section 4.3 and once on the right the machine learning approach described in section 5.3. The visualisation of the figures is structured in the same way as in figure 6.4, but for a different type of cutting edge.

6.4 Comparison and interpretation of the Results

After solving all the described micro lead and cutting edge reconstruction problems and comparing them with model-based and machine learning methods, this section 6.4 serves as an overall comparison and conclusion. In total, it can be summarised that depending on the problem, either the model-based solution or the machine learning solution leads to a better result.

In determining texture direction, the machine learning approach clearly shows that the underlying data for real surfaces is difficult to label and therefore difficult to apply to real measured surfaces. In addition, basic manufacturing characteristics and manufacturing differences between companies are difficult to represent in the simulated data, so that either further fine-tuning of the machine learning process by the customer is required or real surfaces with labeled data must be known. All these problems do not occur in this form with the model-based approach.

In order to be able to unambiguously determine the micro lead angle, the direction of the tool axis must be determined in addition to the texture direction. The tool axis was determined for several simulated measuring ranges from a total of 108 simulated measuring areas for one workpiece to one measured measuring area. The results were better for the model-based approach with the exception for one simulated measuring area. Besides the better results for the model-based approach, it is superior to machine learning especially in terms of flexibility, which is why the model-based approach is to be preferred for this application.

When reconstructing ideal cutting edges using measured cutting edges, very good results could be achieved with the model-based approach and the machine learning approach. Since the used architecture of the machine learning solution of the autoencoder was able to store multiple cutting edge models and to store the shape of the cutting edge in its efficient representation, any kind of defect could be reconstructed by using intact cutting edges as training data. Due to these listed advantages, the machine learning approach has an advantage here and should be looked at more closely for further research steps or initial collaborations with companies.

7 Summary and Outlook

In this chapter 7, the tasks, the methods and the results are briefly summarised and an outlook on further research and validation tasks is given. In total, two topics were addressed in this thesis and three tasks were described. Two tasks were related to the determination of the micro lead and the third and last task was related to the reconstruction of cutting edges. Further information on the micro lead tasks can be found in chapter 1 and on cutting edges in chapter 2. For all tasks, solutions were solved using machine learning, described in chapter 5, and model-based approaches, described in chapter 4.

Overall, a positive conclusion can be drawn for the approaches with model-based methodology and with machine learning. In the first task, the determination of the texture direction in the arcminute range, results with a high resolution accuracy could be determined in both cases. The model-based approach using the cross-correlation method is preferable in this case as it was able to achieve better overall resolution accuracy and was tested on real ground surfaces. The next steps for the model-based approach are further investigations on real samples. For the machine learning approach, validation on real samples is lacking and thus needs to be tested. This means whether the approach used with learned weights from simulated data also has sufficient resolution accuracy for real measurement samples. Here, a disadvantage of machine learning becomes clear in relation to measurement technology in the micrometre range. This is a disadvantage because texture directions of real measurement samples are often not known and it is difficult to produce them in production with a defined texture direction. In addition, these measurements are very time-consuming and therefore expensive. The methodology presented is therefore highly dependent on the simulated surfaces and the simulated label values. A more detailed comparison can be found in section 6.1.

To determine the micro lead angle Ψ, knowledge of the tool axis is required. In this case, a good cylinder fit for the workpiece could be achieved with a model-based optimisation approach. Here it has to be investigated how many measurements on the circumference and whether the 108 measurements proposed in [2] are necessary. In particular, there is a lack of evaluation for real measurements. In simulated approaches, good results could be obtained for less than 108 measurements. The same outlook applies to the machine learning approach. Here, it has to be examined how many measurement surfaces for real measurement data are necessary to fit a cylinder using a point cloud. Simulated results here also showed that fewer than 108 measurements could be sufficient. However, it should be noted that the point density for the machine learning approach has to be reduced depending on the hardware capacity. Therefore, both approaches described in this thesis need to be investigated and validated on real samples. A more accurate comparison and evaluation between the two methods can also be better done afterwards. In addition, the model-based approach can be extended with a Bayesian model to obtain a better fit with predefined and reasonably chosen ranges of values. Likewise, a new neural network can be built and trained with different hardware and thus a higher point density. For both approaches, the research work has not yet been completed. However, in this thesis it is shown that these methods can be used for fitting cylinders to determine the micro lead angle Ψ. For a more detailed comparison, see section 6.2.

The reconstruction of cutting edges is an important step in deciding whether the cutting edge can be further used. Since often no suitable CAD model of the cutting edge is available, an ideal reference must be generated from the measurement data. For the reconstruction of cutting edges, the model-based approach is an improvement on the current status quo. This was not only tested on simulated cutting edges, but could also be validated on real samples. With the model-based approach difficulties arise with cutting edges that have a stronger curvature and strong defects. One way to achieve a good reconstruction for these cases is to use an autoencoder. With this method, it could be shown that reconstruction is also possible for the combination of curvature and strong defects. Furthermore, the reconstruction could also be successfully tested on real samples, which is why the machine learning approach is to be preferred. In a further step, a case with several different types of cutting edges can now be trained and validated, so that a broad spectrum can be covered. The presented solution has shown that at least two types of cutting edges can be learned with one model. Nevertheless, the machine learning approach also has disadvantages, in particular the adjustment of a uniform grid size and normalisation of the data. However, these are minor compared to the results obtained and the advantages of this approach. Refer to section 6.3 for a more detailed comparison.

In addition, an overall comparison between all tasks was carried out to compare the applicability of the machine learning methods in the field of micro-measurement technology, see also section 6.4. Here, all previously determined results were compared and once again evaluated against each other. The overall comparison between the model-based approaches and the machine learning approach is so balanced that it makes more sense to choose a method depending on the corresponding task area than to derive a general statement.

With regard to the areas studied, there is no reason to avoid publishing content that is not based on machine learning. In various optimisation tasks and applications, it is not always the most user-friendly and best possible solution. However, there are also cases where it makes a lot of sense to pursue approaches based on machine learning. Especially for reconstruction tasks, which are inevitably important in metrology, the use of autoencoders is very useful. Good results were also obtained for texture direction determination, although further investigation and an extended model are needed for this. Also in the case of fitting, it could be shown that a good fit can be derived with a priori knowledge through existing training and validation data. In summary, it is always advisable to understand the task and the underlying challenges in order to find the best possible approaches. Nevertheless, the field of machine learning offers the possibility of automation in metrology and micrometre scale, especially in the reconstruction of samples.

A Preprocessing to determine the Texture Direction for Micro Lead

In order to determine the texture direction Θ^t for a micro twist measurement, it is necessary to perform preparatory steps on real samples. Essentially this involves two points, firstly the noise must be reduced and secondly the measured shape of the cylinder must be corrected. For this purpose a median filter, an S-filter and an F-operator are used. Starting with the section for median filter.

A.1 Median Filter

The median filter was originally used in statistics and was used by Tukey for time series analysis 1970 [84] and [85]. Later it was also used for digital image analysis and in many other areas [85]. The reason for this is the good performance and the simplicity in calculation. We will not describe it mathematically here, but explain the median filter using figure A.1. This figure shows that a 3×3 median image is applied to a data set. Here the displayed area is sorted and the median is determined. This information is then modified and changed on the underlying data set.

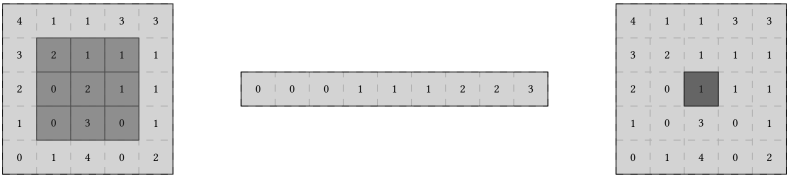


Figure A.1: Schematic representation of the median filter. Illustration is inspired by [86].

A.2 S-Filter

The nesting index of the S-filter λ_s represents the short cut-off wavelength of the lateral structures on the topography [6]. At the cut-off wavelength, the corresponding amplitude is attenuated by 50% [6]. In several literature a recommendation for the correct setting of λ_s is given. For example, VDI/VDE 2655 part 1.1 λ_s recommends that the optical resolution should be set to two to three times the value for diffraction according to the Rayleigh criterion [87] and [6]. From a description in DIN EN ISO 25178-3, the nesting index can be selected from a range of values corresponding to the maximum lateral sample distance [88, 6]. For the measuring instrument used to evaluate the texture direction in this thesis, the next largest maximum sample distance is $2.5\,\mu\mathrm{m}$, which is why λ_s with $8\,\mu\mathrm{m}$ was chosen [88]. According to DIN EN ISO 25178-3, the S-filter is an areal Gaussian filter [88]. It is implemented according to E DIN EN ISO 16610-60 and -61 [89] and [90] as a linear planar filter working in the image area.

A.3 F-Operator

Since the measured surface has a cylindrical shape and this can affect the results of the texture directional estimation, this underlying shape is removed by applying an F-operator [18]. For the evaluation of the texture direction for real data a second order surface polynomial was used, which was realised with least square method. Lemke et al. find that the shape removal by means of the second-order surface polynomial fit is suitable for measurements on a cylinder bore surface [91]. The advantage of the F operation used here is its simple implementation, but it differs from DIN EN ISO 25178-3, which requires a total least squares method [88] and [6].

B Alternative Options for choosing an Autoencoder to realise a Ideal Reconstruction of a Cutting Edge

Section 5.3.3 describes how to use an autoencoder to reconstruct defective cutting edges. This reconstruction is based on an encoder and decoder component, see also section C.8.1. However, the structure of the decoder can also be used differently from the solution described in section 5.3.3. There, after compressing the information within a bottle neck, the information was trained with upsampling and newly learned weights for the decoding step. A common other implementation is the use of a convolutional transposed and tied weights. In fact, it is often the preferred approach, as tied weights can be used to reduce the number of learned weights, see also section C.8.2. This simply uses the transposed weights from the decoder step. Figure B.1 shows the results of the individual cutting edges for the different methods. Here the input of the cutting edge is compared with the methodology of tied weights, with convolutional transposed and learned weights for the decoder and convolutial transposed with tied weights. In this case the best results were obtained for the case with learned weights for the decoder and convolutional transposed. The results of this thesis were presented in [39] by the author and other contributors. The corresponding numerical realisation can be found in the Git repository [83]. The sources [92] and [93] were used to implement tied weights.

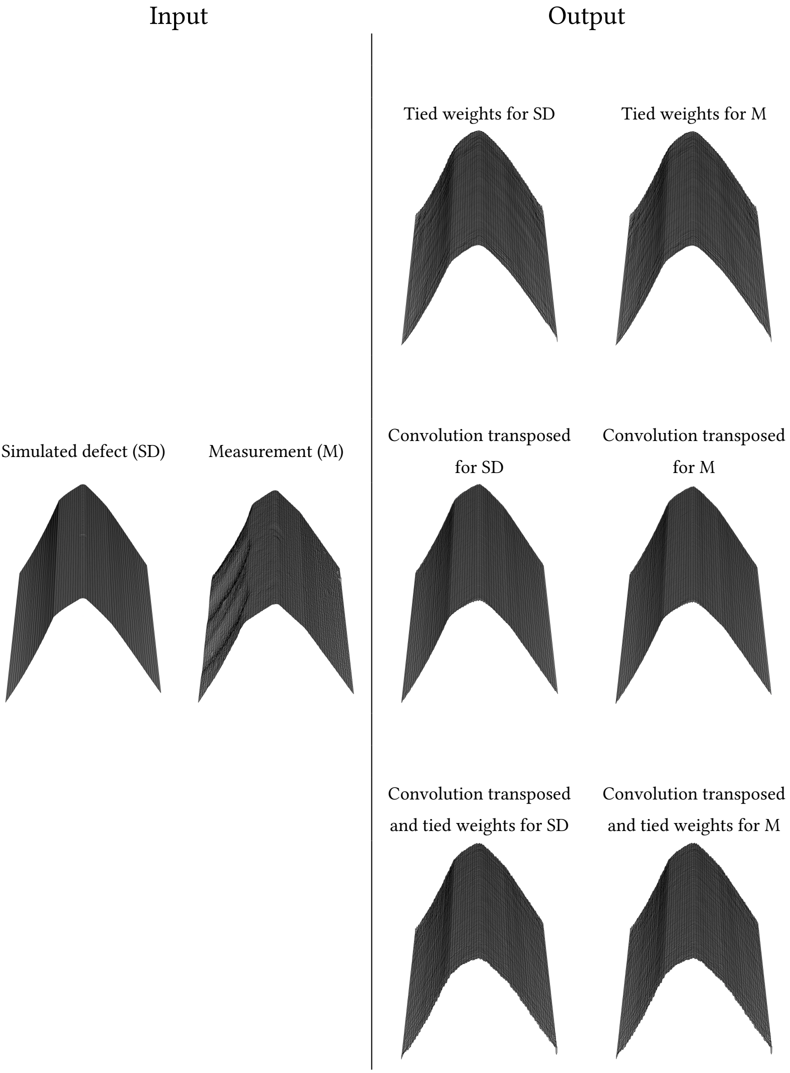

Input
Output
Tied weights for SD
Tied weights for M
Simulated defect (SD)
Measurement (M)
Convolution transposed for SD
Convolution transposed for M
Convolution transposed and tied weights for SD
Convolution transposed and tied weights for M

Figure B.1: Two cutting edges with defects are shown on the left. One defect is artificially added and the second picture shows a real measurement. In the upper right corner the reconstruction of the defective cutting edges with tied weights is shown. Middle right and bottom right the convolutional transposed was used for decoding. Bottom right the image was additionally evaluated with tied weights. [39]

C Basics of Mathematics Foundations and Machine Learning

In this chapter, the mathematical foundations of linear algebra, probability theory, signal processing and optimization are considered. These form the basis for the theories used, which are described in chapter 3. Further notations, such as indexing or operators for linear algebra, are summarized under Notations. Starting from basic definitions and notations used in this thesis.

C.1 Basic Definitions and Notations

The most important fields in this thesis are the natural numbers $\mathbb{N}$ and the real numbers $\mathbb{R}$. In this thesis the field of natural numbers $\mathbb{N}$ contains the element 0. Small and bold Latin characters are used to identify vectors. Large and bold Latin characters are used for matrices. For the n-dimensional space of real numbers $\mathbb{R}^n$ is used and for the set of all matrices by m rows and n columns by $\mathbb{R}^{m \times n}$ is used, where each element of the matrix is a real number and $m, n \in \mathbb{N}$. Now some basic notations are introduced which are needed in the following sections, starting with some basic definitions.

Definition C.1.1 (Polynom): Let $\mathbb{K}$ be a field and the polynomial ring is given with $\mathbb{K}[x]$ over $\mathbb{K}$ with the variable symbol x. A polynom over $\mathbb{K}$ is defined with an expression of the form

$$w_0 + w_1 x + \ldots + x_n x^n, \tag{C.1}$$

with $w_i \in \mathbb{K}$ for all i and $n \in \mathbb{N}$.

The notation $\mathbb{K}[x] = \mathbb{P}_n$ will be used for a polynomial ring. In addition, with $n > 0$ it is assumed that $w_n \neq 0$ [94] [95].

Definition C.1.2 (p-norm or L^p-norm): Let $\| \cdot \| \colon \mathbb{R}^n \to \mathbb{R}_{\geqslant 0},\ \mathbf{x} \mapsto \|\mathbf{x}\|_p$ be a function that satisfies all the conditions of a norm. The so-called p-norm is defined by

$$\|\mathbf{x}\|_p := \left(\sum_{i=0}^{n-1} |x_i|^p \right)^{\frac{1}{p}}, \text{ with } \mathbf{x} \in \mathbb{R}^n. \tag{C.2}$$

Mostly the L^2-norm is used in this thesis. In this case, the alternative notation $\|\mathbf{x}\|_2 = \langle \mathbf{x}, \mathbf{x} \rangle^{\frac{1}{2}} = \left(\mathbf{x}^T \mathbf{x}\right)^{\frac{1}{2}}$ is often used. The reader should be aware of these notations.

Next, the definitions for scalar fields and vector fields are introduced. For the scalar field D is a domain and therefor an open connected subset of a finite-dimensional vector space, in our case $\mathbb{R}^n$.

Definition C.1.3 (Gradient): Let $D \subseteq \mathbb{R}^n$ be a domain and $f \colon D \to \mathbb{R},\ \mathbf{x} \mapsto f(\mathbf{x})$ a scalar field. The gradient in a point $\mathbf{x}_0 \in \mathbb{R}^n$ is defined by

$$\nabla \left(f(\mathbf{x}_0) \right) = \begin{pmatrix} \left.\frac{\partial f(\mathbf{x})}{\partial x_0}\right|_{\mathbf{x}=\mathbf{x}_0} & \cdots & \left.\frac{\partial f(\mathbf{x})}{\partial x_{n1}}\right|_{\mathbf{x}=\mathbf{x}_0} \end{pmatrix}^T. \tag{C.3}$$

Definition C.1.4 (Hessian): Let $D \subseteq \mathbb{R}^n$ be a domain and $f \colon D \to \mathbb{R},\ \mathbf{x} \mapsto f(\mathbf{x})$ a scalar field. The hessian matrix in a point $\mathbf{x}_0 \in \mathbb{R}^n$ is defined by

$$\mathbf{H} \left(f(\mathbf{x}_0) \right) = \begin{pmatrix} \left.\frac{\partial^2 f(\mathbf{x})}{\partial x_0 \partial x_0}\right|_{\mathbf{x}=\mathbf{x}_0} & \cdots & \left.\frac{\partial^2 f(\mathbf{x})}{\partial x_0 \partial x_{n1}}\right|_{\mathbf{x}=\mathbf{x}_0} \\ \vdots & \ddots & \vdots \\ \left.\frac{\partial^2 f(\mathbf{x})}{\partial x_{n1} \partial x_0}\right|_{\mathbf{x}=\mathbf{x}_0} & \cdots & \left.\frac{\partial^2 f(\mathbf{x})}{\partial x_{n1} \partial x_{n1}}\right|_{\mathbf{x}=\mathbf{x}_0} \end{pmatrix}. \tag{C.4}$$

Definition C.1.5 (Convex set): Given $\mathbf{x}, \mathbf{y} \in D \subseteq \mathbb{R}^n$. D is called a convex set, if for any choice of $\mathbf{x}$, $\mathbf{y}$ and $\Theta \in [0,1]$ the statement

$$\Theta\mathbf{x} + (1-\Theta)\mathbf{y} \in D \tag{C.5}$$

is valid.

Definition C.1.6 (Convex function): Given $f\colon \mathbb{R}^n \supseteq D \to \mathbb{R},\ \mathbf{x} \mapsto f(\mathbf{x})$ a scalar field, $\mathbf{x}, \mathbf{y} \in D$ and D a convex set. The scalar field f is convex, if

$$f\left(\Theta\mathbf{x} + (1-\Theta)\mathbf{y}\right) \leqslant \Theta f(\mathbf{x}) + (1-\Theta)f(\mathbf{y}), \quad \forall\Theta \in [0,1] \tag{C.6}$$

is true [96].

Definition C.1.7 (Jacobian): Let $D \subseteq \mathbb{R}^n$ be a domain and $f\colon D \to \mathbb{R}^m$ $\mathbf{x} \mapsto f(\mathbf{x})$ a vector field. Also all partial derivatives of f exists. The Jacobian matrix in a point $\mathbf{x}_0 \in \mathbb{R}^n$ is defined by

$$\mathbf{J}\left(f(\mathbf{x}_0)\right) = \begin{pmatrix} \left.\frac{\partial f_0(\mathbf{x})}{\partial x_0}\right|_{\mathbf{x}=\mathbf{x}_0} & \cdots & \left.\frac{\partial f_0(\mathbf{x})}{\partial x_{n1}}\right|_{\mathbf{x}=\mathbf{x}_0} \\ \vdots & \ddots & \vdots \\ \left.\frac{\partial f_{m1}(\mathbf{x})}{\partial x_0}\right|_{\mathbf{x}=\mathbf{x}_0} & \cdots & \left.\frac{\partial f_{m1}(\mathbf{x})}{\partial x_{n1}}\right|_{\mathbf{x}=\mathbf{x}_0} \end{pmatrix}. \tag{C.7}$$

C.2 Basics of linear Algebra

In this section C.2 the basics of linear algebra are briefly summarized. Not many basic terms like the trace or the determinant of a matrix are repeated here. If the reader is uncertain about these terms, the literature [97] [98] [99] is recommended. Within this section C.2 the eigendecomposition, the Singular value decomposition, the Moore-Penrose pseudoinverse and finally the principal component analysis are discussed.

Eigendecomposition The motivation to introduce the eigenvaluedecomposition is inspired by [56]. The decomposition of large facts into many small ones can often bring many new insights. For example, a puzzle that consists of many individual small pieces, or, to stay in the field of mathematics, the decomposition of integers into smaller integers. For example, if you look at the number 16, this number can be broken down into $16 = 2 \cdot 2 \cdot 4$. Thus, every integer multiple of 16 is also divisible by 4 with a remainder of 0. In addition to the decompositions mentioned above, the decomposition of a matrix can also provide important properties and functions. One of the most important decompositions of matrices is the so-called eigendecomposition. The eigendecomposition is founded and justified on the basis of two definitions.

Definition C.2.1 (Diagonalizable): A matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$ is diagonalizable if $\exists V \in \mathbb{C}^{n \times n} \colon V^{-1}AV$ is a diagonal matrix and $\mathbf{det}(V) \neq 0$.

Definition C.2.2 (Eigenvector, Eigenvalue): Let $\mathbf{0} \neq \mathbf{v} \in \mathbb{C}^n$ be a vector and $\lambda \in \mathbb{C}$ a scalar. $\mathbf{v}$ is called the eigenvector corresponding to the eigenvalue λ if the equation C.8 for $\mathbf{A} \in \mathbb{C}^{n \times n}$ is satisfying

$$\mathbf{A} \cdot \mathbf{v} = \lambda \mathbf{v}. \tag{C.8}$$

The definition C.2.2 and definition C.2.1 is now used to describe the principle of eigendecomposition. Now assume that the certain eigenvectors of the matrix $A \in \mathbb{C}^{n \times n}$ are linearly independent and can be defined by the following expression

$$\mathbf{V} = \begin{pmatrix} \mathbf{v}_0 & \cdots & \mathbf{v}_{n-1} \end{pmatrix}. \tag{C.9}$$

If now equation C.9 is linked with the two definitions C.2.1 and C.2.2, the following theorem can be concluded by a simple proof.

Theorem C.2.1 (Eigendecomposition): $\mathbf{v}_0, \ldots, \mathbf{v}_{n-1} \in \mathbb{C}^n$ are linear independent eigenvector to the corresponding eigenvalues $\lambda_0, \ldots, \lambda_{n-1} \in \mathbb{C}$ of $\mathbf{A} \in \mathbb{C}^{n \times n}$. The matrix

$\mathbf{A}$ can be estimated by

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}, \quad \text{with} \tag{C.10}$$

$$\mathbf{V} = \begin{pmatrix} \mathbf{v}_0 & \dots & \mathbf{v}_{n-1} \end{pmatrix} \text{ and} \tag{C.11}$$

$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_0 & 0 & \cdots & 0 \\ 0 & \lambda_1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_{n-1} \end{pmatrix} =: \mathbf{diag}\left(\begin{pmatrix} \lambda_0 & \dots & \lambda_{n-1} \end{pmatrix}\right). \tag{C.12}$$

Similar to our example and the decomposition of an integer number, the decomposition of a matrix can also provide important insights. For example, the eigenvalues can be used to get information about the singularity of the matrix $\mathbf{A}$. Again, note that the decomposition is not unique. In order to bring a certain uniqueness to the representation, the eigenvalues are sorted in descending order and the corresponding eigenvectors are normalized to the length 1. However, not every matrix can be decomposed into its eigenvalues and eigenvectors. In this thesis, a much simpler case is discussed. Here, the eigenvectors form an orthonormal basis. The columns are orthogonal to each other and normalized to the length 1. In this case the matrix $\mathbf{V}$ is orthogonal and the inverse of the matrix is equal to its transposition $\mathbf{V}^T$. The conclusion from the theorem C.2.1 can therefore be simplified by

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T. \tag{C.13}$$

Before continuing with the singular value decomposition, a theorem is notated to determine the definitiveness of a matrix. Again, no proof is given, since it is found in most books on linear algebra [100].

Theorem C.2.2 (Definite Matrix form): Let the matrix $\mathbf{A} \in \mathbb{K}^{n \times n}$ be a hermitian matrix for $\mathbb{K} = \mathbb{C}$ or a symmetric matrix for $\mathbb{K} = \mathbb{R}$ and λ_i for $i = 0, \dots, n-1$ the corresponding eigenvalues of $\mathbf{A}$.

$$\begin{aligned} \mathbf{A} \text{ is positiv definit} &\iff \lambda_i > 0, \forall i && \text{(C.14)} \\ \mathbf{A} \text{ is positiv semidefinit} &\iff \lambda_i \geqslant 0, \forall i && \text{(C.15)} \\ \mathbf{A} \text{ is negativ definit} &\iff \lambda_i < 0, \forall i && \text{(C.16)} \\ \mathbf{A} \text{ is negativ semidefinit} &\iff \lambda_i \leqslant 0, \forall i. && \text{(C.17)} \end{aligned}$$

Singular Value Decomposition The disadvantage of the eigendecomposition in section C.2 is that it works only for square matrices. However, the matrices available to us do not have to be square. They can have a different number of rows and columns. In general, singular value decomposition, similar to eigendecomposition, is a way to factorize matrices. In fact, it can be seen as an extension of the eigendecomposition since singular value decomposition exists for all real matrices. The decomposition is noted as a definition.

Definition C.2.3 (Singular value decomposition): Let the matrix $\mathbf{A} \in \mathbb{K}^{m \times n}$ with $\mathbb{K}$ a field and **rank** $(A) = r$. The Singular value decomposition is defined by

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H \tag{C.18}$$

with

$\mathbf{U} \in \mathbb{K}^{m \times m}$ a unitary matrix if $\mathbb{K} = \mathbb{C}$ and an orthogonal matrix if $\mathbb{K} = \mathbb{R}$

$\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ is a diagonal matrix with non-negative values

$\mathbf{V} \in \mathbb{K}^{n \times n}$ a unitary matrix if $\mathbb{K} = \mathbb{C}$ and an orthogonal matrix if $\mathbb{K} = \mathbb{R}$

$\mathbf{V}^H$ is the hermitian transposed of $\mathbf{V}$ respectively the transposed matrix if $\mathbb{K} = \mathbb{R}$.

The positive entries from $\mathbf{\Sigma}$ are called singular values.

Singular decomposition is used in many fields, for example in the calculation of the rank of a matrix or in the determination of a generalized inverse, the so-called Moore-Penrose-Pseudoinverse.

Moore-Penrose Pseudoinverse A classical introduction and a realistic scenario for the solution of a system of equations is considered. In many cases there is an overdetermined system, i.e. more equations than unknown variables. This problem can no longer be solved by building an inverse, since the matrix $\mathbf{A}$ is no longer quadratic. In general, however, the system can be over- or under-determined. Therefore, assuming that $\mathbf{A}$ is a non-square matrix with $\mathbf{A} \in \mathbb{C}^{m \times n}$ and $m \neq n$. Furthermore, $\mathbf{b} \in \mathbb{C}^{m}$ and $\mathbf{x} \in \mathbb{C}^{n}$. The linear system of equations can now be solved with

$$\mathbf{A}\mathbf{x} = \mathbf{y}. \tag{C.19}$$

Our goal is to solve this equation C.19 on $\mathbf{x}$. But that's not exactly trivial. The Moore-Penrose Pseudoinverse describes one and also the widespread method to generalize an inverse. Many algorithms in this thesis use the Moore Penrose pseudoinverse and this can be determined by a singular value decomposition. Furthermore, the Moore Penrose pseudoinverse is defined for $\mathbb{C}$ or $\mathbb{R}$ and it is also unique. This is followed by a theorem for determining the generalized inverse, which is often mistakenly noted as a definition. The proof from Penrose can be found in [101]. This section C.2 ends with a theorem and an additional note that the numerical realisation of the generalised inverse uses more advanced techniques. A possible numerical realization using singular value decomposition can be found in [99].

Theorem C.2.3 (Moore-Penrose Pseudoinverse): **For any** $\mathbf{A} \in \mathbb{K}^{m \times n}$ with **rank** $(\mathbf{A}) = r$ the four equations C.20 - C.23 have a **unique** solution. $\mathbf{A}^{+}$ is called the Moore-Penrose Pseudoinverse of $\mathbf{A}$.

$$\mathbf{A}\mathbf{A}^{+}\mathbf{A} = \mathbf{A} \tag{C.20}$$

$$\mathbf{A}^{+}\mathbf{A}\mathbf{A}^{+} = \mathbf{A}^{+} \tag{C.21}$$

$$\left(\mathbf{A}\mathbf{A}^{+}\right)^{*} = \mathbf{A}\mathbf{A}^{+} \tag{C.22}$$

$$\left(\mathbf{A}^{+}\mathbf{A}\right)^{*} = \mathbf{A}^{+}\mathbf{A}. \tag{C.23}$$

Principal Components Analysis Principal Component Analysis (PCA) is one of the most important and oldest methods of dimension reduction, and it is also very common in machine learning applications. The range of applications of principal component analysis in general is huge, looking at the whole field of biology, medicine, psychology, etc. With PCA a first machine learning algorithm is described, which only requires the basics of linear algebra. In this section C.2 PCA is briefly motivated, a theorem is presented and a small example of reconstruction with PCA is described.

Motivation Starting point are many measurement data. These measurement data require a lot of storage space, and not all data contribute to a much better understanding of these data. Therefore, the general goal is to “save” as much storage space as possible compared to the original data set while losing as little information as possible. This is the goal of PCA and our motivation. The data is defined with $\mathbf{X} = \begin{pmatrix} \mathbf{x}_0 & \cdots & \mathbf{x}_{m-1} \end{pmatrix} \in \mathbb{R}^{n \times m}$.

Theorem The source [56] serves as a guideline for the theorem. The proof will not be repeated here. However, the reader is strongly advised to look at the proof if PCA is unknown. The proof provides many important aspects of PCA.

Theorem C.2.4 (Principal components analysis): Given $\mathbf{X} = \begin{pmatrix} \mathbf{x}_0 & \cdots & \mathbf{x}_{m-1} \end{pmatrix} \in \mathbb{R}^{n \times m}$ points. Furthermore let $f \colon \mathbb{R}^n \to \mathbb{R}^l \, , \mathbf{x} \mapsto f(\mathbf{x}) = \mathbf{c}$. For each point $\mathbf{x}_i$ a corresponding $\mathbf{c}_i \in \mathbb{R}^l$ with $l < n$ and a decoding function $g \colon \mathbb{R}^l \to \mathbb{R}^n \, , \mathbf{c} \mapsto g(\mathbf{c}) = \mathbf{Dc}$ where $\mathbf{D} \in \mathbb{R}^{n \times l}$ is the matrix defining the decoding is wanted. Furthermore, the columns of matrix $\mathbf{D}$ are linearly independent to each other. The optimal code point $\mathbf{c}^*$ for each point $\mathbf{x}$ can be estimated by

$$f(\mathbf{x}) = \mathbf{c} = \mathbf{D}^T \mathbf{x} \tag{C.24}$$

and

$$\mathbf{D}^* = \arg\max_{\mathbf{D}} \operatorname{Trace}\left(\mathbf{D}^T \mathbf{X}^T \mathbf{X} \mathbf{D}\right) . \tag{C.25}$$

Note It is immediately clear, that the solution to the maximization problem for $\mathbf{D}$ with $l = 1$ can be solved by eigenvalue decomposition. $\mathbf{D}$ is then the eigenvector of

$\mathbf{X}^T\mathbf{X}$ to the corresponding largest eigenvalue. For $l > 1$ the proof is no longer quite trivial, but can be solved with induction.

Example As already mentioned, PCA is a simple machine learning algorithm. To illustrate this and the procedure of PCA, an example for the reconstruction of digits is given. Here the MNIST dataset is used. The MNIST dataset are handwritten digits, available in [102]. The digits have already been size-normalized and centered in a fixed-size image. Some digits and their reconstructions can be seen in figure C.1. To realize the reconstruction a PCA with the two most important components and the 20 most important components was used, i.e. $l = 2$ and $l = 20$. Figure C.1 shows that the reconstruction has already been successful for many digits. The algorithm for generating this image can also be found on my git repository [103]. This example was inspired by [104].

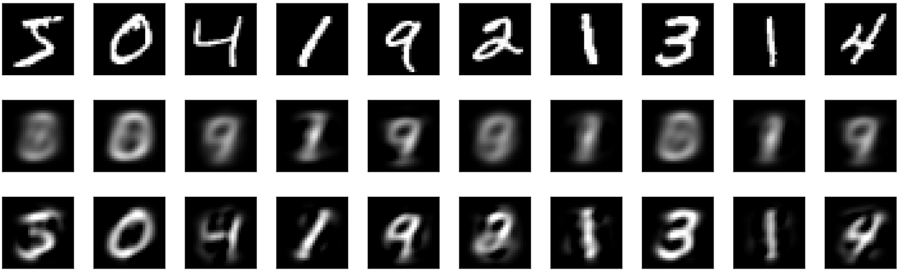


Figure C.1: Reconstruction of digits using PCA. The first row shows the original digits from the MNIST dataset. The second row represents a reconstruction of the digits for $l = 2$. The third row represents a reconstruction with $l = 20$ (see theorem C.2.4).

C.3 Basics of Probability Theory

Probability theory is extremely important in science and industry, especially in the field of metrology. They form the basis for many theoretical descriptions and give us a lot of information about measured data. Many theoretical descriptions of machine learning go hand in hand with probability theory. An obvious example here is that machine learning always has to deal with uncertainties. In this section C.3 the field of probability theory is briefly discussed. The emphasis here is on briefly. Only a very small area of this theory is covered. This section C.3 will also not be divided into further subsections. For further information on probability theory, such as calculating the expected value or the variance of probability distributions, the literature [105] can be recommended. Starting with a first definition of random numbers. If uncertain, the figure C.2 illustrates the definitions of probability space and measurable space. A concrete definition of these terms is not given in this thesis.

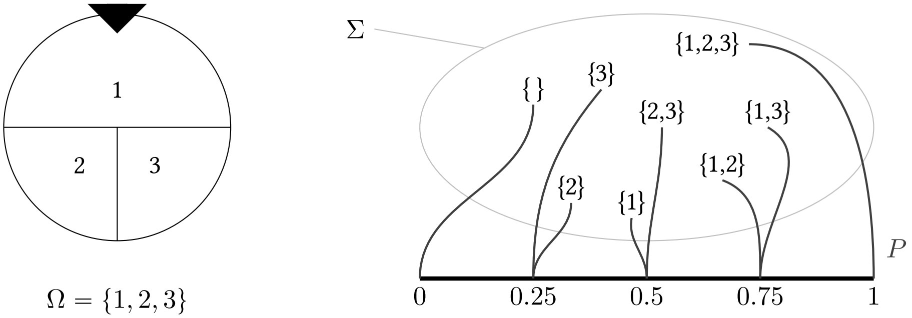


Figure C.2: Illustration of the probability space (Ω, Σ, P) using a rotating disk as example. The sample space is given by $\Omega = \{1, 2, 3\}$. Σ is a σ-Algebra over Ω and P the measure [106].

Definition C.3.1 (Random number): Let (Ω, Σ, P) be a probability space and $\left(\Omega', \Sigma'\right)$ a measurable space. One $\left(\Sigma, \Sigma'\right)$ measurable function $X : \Omega \rightarrow \Omega'$ is called Ω' random variable to Ω [107].

A random variable is called discrete if it assumes only finitely many or countably infinite many values [108].

Probability mass function Simply spoken, a probability distribution over discrete random variables is a probability mass function. More precisely, the probability mass function maps a discrete random number into a range of $[0, 1]$. A very formal definition is covered by the next definition.

Definition C.3.2 (Probability mass function): Let $X\colon \Omega \to \Omega'$ be a discrete random number. The probability mass function $f\colon \Omega' \to [0,1]$ is defined by [109]

$$f(x) = P\left(\{\omega \in \Omega\colon X(\omega) = x\}\right). \tag{C.26}$$

It should also be noted that probability mass functions can affect many variables simultaneously. Such a probability distribution over many variables is called a joint probability distribution [56]. It will also note with $P(x)$ to keep the notation short. A joint probability thus carries the notation $P(x, y)$.

Probability density function When working with continuous variables, a probability density function (PDF) is required instead of a probability mass function. However, an exact definition will not be used at this point. An unambiguous separation is indicated by the use of a small letter, $p(x)$.

Inverse Radon Transformation The implementation of radon inversion can be done in different ways. The so-called filtered backprojection is one of them.

Definition C.3.3 (Filtered backprojection): Let the projection $\mathcal{R}(\rho, \Theta)$ and the conditions given by definition 3.1.8. The inverse Radon transformation can be determined by

$$f(\mathbf{x}) = \int_0^\pi \left(\mathcal{R}(\rho, \Theta) * w(\rho)\right)\left(\langle \mathbf{x}, \mathbf{n}_\Theta \rangle\right)\, d\Theta. \tag{C.27}$$

The weighted function $w(\cdot)$ is defined so that for the Fourier Transformation the statement $\mathcal{F}\{w(\cdot)\}(\omega) = \|\omega\|_2$ is true [49].

C.4 Optimization

The introduction and many other parts in this section C.4 are motivated from [53]. Finite dimensional continuous optimization deals with minimizing or maximizing a target function with a finite number of continuous decision variables. Many optimization applications can be solved by linear optimization, but there exist also various nonlinear problems from natural sciences, engineering and economics. These include e.g. parameter fitting problems. The methods required in this thesis for solving optimisation tasks are mostly determined by the application areas of machine learning. This section only describes methods for nonlinear optimization, but distinguishes between unrestricted and restricted optimization. When optimizing an target function, constraints often have to be considered. Here a distinction is made between inequality and equation restrictions and the restricted problem is considered in a general form[1] .

$$(P): \quad \min f(x) \quad \text{s.t} \quad g_i(x) \leqslant 0, i \in I, \quad h_j(x) = 0, j \in J \tag{C.28}$$

with continuous scalar fields $f, g_i, h_j : \mathbb{R}^N \to \mathbb{R}$ for $i \in I$ and $j \in J$. The index sets I and J are finite and possibly empty. These are defined by

$$I = \{1, \ldots, p\} \text{ and } J = \{1, \ldots, q\}, \quad p, q \in \mathbb{N} \text{ with } q < n. \tag{C.29}$$

If the optimizing target function has no side conditions, the sets I and J are empty. This case will be considered first.

Optimization methods are used, among other things, for algorithms with artificial intelligence. Common numerical methods for minimizing a function $f : \mathbb{R}^N \to \mathbb{R}$ are described, whereby these always fulfill the necessary conditions of continuity and differentiability. For example the gradient descent method to name one. Preliminary work is necessary to be able to apply and understand numerical methods. In this case it will be limited to the most necessary theorems without proving them. The first one describes the necessary first-order optimality condition.

Theorem C.4.1 (Necessary first-order optimality condition): Let $f \in C^1(\mathbb{R}^n)$ be a scalar field. If $\mathbf{x}^{(cri)} \in \mathbb{R}^n$ is a local minimum point, then $\nabla f\left(\mathbf{x}^{(cri)}\right) = 0$ is true. $\mathbf{x}^{(cri)}$ is called a critical point.

---

[1] Any optimization problem, even one that wants to maximize the target function, can be noted as a minimization problem.

Theorem C.4.1 only represents a necessary condition of optimality. It states that a local minimum point of f is necessarily a critical point, but the property of being a critical point is not sufficient for minimalism.

Theorem C.4.2 (Required and sufficient second order optimality condition): Let $f \in C^2\left(\mathbb{R}^n\right)$ be a scalar field. If $\mathbf{x}^{(opt)} \in \mathbb{R}^n$ is a local minimum point, then $\nabla f\left(\mathbf{x}^{(opt)}\right) = 0$ is true and $H\left(f\left(\mathbf{x}^{(opt)}\right)\right)$ is positive semidefinit[2]. If $H\left(f\left(\mathbf{x}^{(opt)}\right)\right)$ is positive definite, then $\mathbf{x}^{(opt)}$ is a strict local minimum of f.

C.5 Fundamentals Machine Learning

Artificial intelligence is an increasingly widespread field of computer science. In addition to automation, another strong pillar in the area of artificial intelligence are methods and procedures of machine learning. Machine learning itself is in turn a superset of deep learning and artificial neural networks. In order to understand and apply artificial neural networks, basics from the field of machine learning are inevitable. It is important to note that for our applications in metrology methods are difficult to use without deep learning. First starting with terminology and basic terms. For an introduction to the Training Set, Validation Set and Test Set, as well as Overfitting and Underfitting, see the section 3.2

C.5.1 Learning

But what exactly does learning in machine learning mean? One possible definition is given by Mitchell [110]:
“A computer program is said to learn from experience (E) with respect to some class of tasks (T) and performance measure (P), if its performance at tasks in (T), as measured by (P), improves with experience (E).”
Individual fields of experience (E), tasks (T) and measurement of performance (P) will be discussed briefly. It primarily serves to introduce terms and conventions.

---

[2] see Theorem C.2.2

Tasks (T) The task itself is not necessarily part of the learning. A task can be, for example, letting a robot learn a certain action, such as walking. It is possible to write a program directly to execute this task or use machine learning. Both approaches would solve this task. Machine learning tasks are usually described in terms of how a system should process an example [56]. An **example** is a collection of **features** that have been quantitatively measured on an object or event that the machine system is to process [56]. An example is typically represented as a vector $\mathbf{x} \in \mathbb{R}^n$, where each x_i entry of the vector is a feature. For example, the features of an image are usually the pixels values [56]. There are a number of tasks that are solved with machine learning. In this thesis only two tasks are discussed, **classification** and **regression**. Simply spoken, a regression model predicts continuous values and classification model predicts discrete values. An example of a regression would be the value of a house in Kaiserslautern (Germany), and an example of a classification is the task of determining whether a cat or a dog is represented in a picture.

Performance (P) After applying methods for machine learning, the question arise on how well the model could predict. Thus, a measure of performance (P) is trivially relevant. Of course, different criteria are helpful for different tasks. A commonly used example of performance is **accuracy**, which simply determines the relative frequency of a correct prediction. It should be mentioned that accuracy is not always a good choice. Especially when an event occurs relatively rarely. Better methods for measure performance here are **Precision** and **Recall**. There are also reference value that can be used to evaluate the quality of a set. To determine the two values Precision and Recall, a distinction is made between actual positive or actual negative and predicted positive or predicted negative. For example, true positive are data points labeled as positive that are actually positive and false positive are data points labeled as positive that are actually negative [111]. Using this the recall value is the ability of a classification model to identify all relevant instances and precision the ability of a classification model to return only relevant instances[111]. Often such approaches for classification are much more meaningful.

Experience (E) Machine learning algorithms can be broadly categorized as **unsupervised** or **supervised**[3] by what kind of experience they are allowed to have during the learning process [56]. **Supervised learning** can be compared well with the learning behavior at a university. The student receives a lot of questions and learns or understands the connection to the corresponding solutions. Through gained knowledge, the student is able to work on new questions in the same subject topic [57]. In supervised model, the corresponding **labels** are learned from the input. The labels are the "answer" or "result" of an **example**. **Unsupervised learning** trains a model to recognize patterns in data sets. This data set is typically unlabeled. An example for unsupervised learning is PCA, see theorem C.2.4. Other models in machine learning have additional interactions with their environment and are not associated with pure experience from a fixed data set [56] One example is **reinforcement learning**. However, this will not be discussed in detail. This is only mentioned for completeness reasons.

C.5.2 Regularization

As in other algorithms and methods, a machine learning system must be well applied to individual tasks. In addition of changing capacity, other techniques can also be used. A very important one is regularization. Regularization adds a penalty on a model's cost function and helps to prevent overfitting. Different kinds of regularization are available:

1. L_2 regularization or Ridge Regression[112]: Let us assume that the cost function is described with the mean square error MSE. This cost function is now extended with L_2 regularization by adding additionally the sum of all weights. The new cost function is therefor given by

   $$MSE + \alpha\langle \mathbf{w}, \mathbf{w}\rangle, \quad \alpha \in \mathbb{R}_{\geqslant 0}. \tag{C.30}$$

   This additional condition to minimize the error ensures that the weights themselves do not become too heavy. In our regression example from figure 3.3, this means that the weights result in less oscillation and reduce overfit, see figure C.3. For a detailed description [113], [114] is recommended.

---

[3] or also semi-supervised

2. L_1 regularization or Lasso Regression[4][112]: L_1 regularization also adds penalties to the cost function similar to L_2 regularization. Instead of the sum of the squared weight terms, the sum of the absolute values is used. However, the biggest and most important difference between the two regularization methods is that L_1 regularization reduces smaller weights, i.e., less important features, to zero and thus completely removes this feature. If a large number of features are available, a selection of important features can be realized. For a detailed description [114] is recommended.

3. Dropout regularization: A useful regularization method while training neural networks. This is discussed in section 3.3.1.

4. Early stopping: The early end of the training process. It is not formally a regularization method, but can avoid overfitting. The training process is aborted when the validation error starts to increase. See section 3.3.1 for more information.

C.5.3 Ensemble learning

Suppose you randomly select a large number of people, ask them a complex question and collect the answers. In many cases, the collected answer will be better than a single expert's answer. This phenomenon is also called group or swarm intelligence. The same applies to predictions, i.e. classification and regression. Here, prediction made by collection is better than a single prediction, determined by a best prediction model. The collection of these predictions is called **ensemble** and the method ensemble learning [59]. Ensemble methods can eliminate the weak parts of individual models by majority decision and are therefore often more robust. One very efficient method is Random Forest. This is discussed in more detail in section C.5.3. It is worth mentioning that despite its simplicity, ensemble learning is one of the most powerful algorithms known for machine learning. In general, ensemble methods are very successful in the field of machine learning competitions [59]. In the following two very well known methods are introduced, bagging and boosting.

Bagging There are different methods to generate diverse sets. One divides the training set into several smaller subsets and trains them all with the same algorithm. If

---

[4] Stands for Least Absolute Shrinkage and Selection Operator.

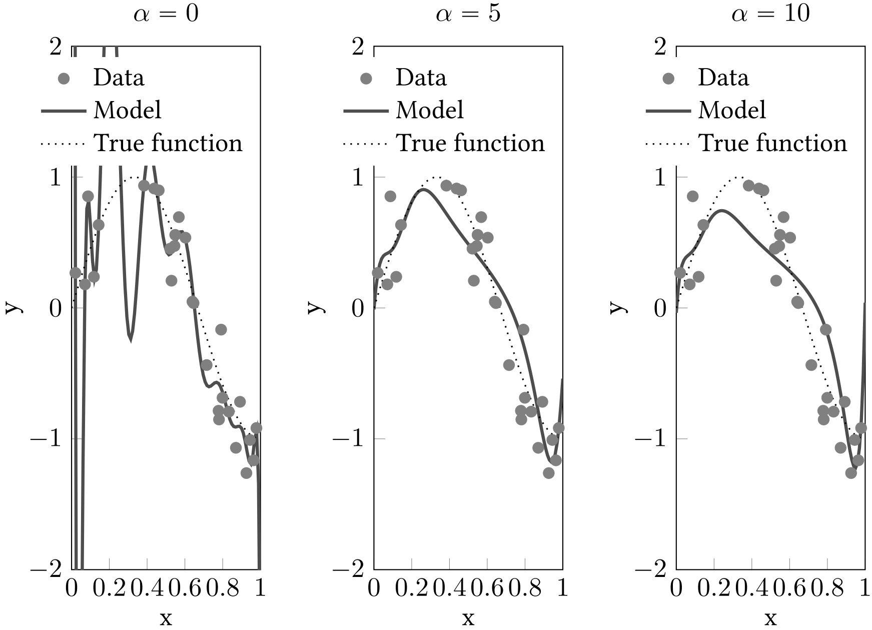


Figure C.3: The figure shows a regression example for measured data and a L_2 regularization. In the left figure, the factor $\alpha = 0$ was used, see equation C.30. The model function is overfitting. If the factor α is increased, the middle and right figure shows that regularization reduce overfitting.

the sampling of the subsets is done with replacement, it is called bagging[5]. After all partitions have been trained, a new prediction can be made by collecting the predictions. Typically, it is the most frequently used prediction for classification or the mean value for regression. One very famous bagging Method is the Random Decision Forest (short **Random Forest**). This method is an ensemble of decision trees. This means that a set of features is randomly taken and a decision tree is created. This results in several decision trees where the average of all decision trees[6] describes the best fit. [59] A decision tree model is a sequence of branch statements [57]. As an example for decision trees, the house price in Kaiserslautern (Germany) can be predicted based on many decisions. For

---

[5] Sampling without replacement is called pasting.

[6] called forest

example, it is possible to predict that a newly built house with a certain square meter size will be more expensive than an average house price in Kaiseslautern.

Boosting Unlike bagging methods, boosting methods use sequential prediction rather than parallel prediction. There are many methods that use boosting approaches. One of the most popular is Adaptive Boosting (short **AdaBoost**). AdaBoost pays more attention to the underfitting predecessors. Thus the sequentially following subsets can concentrate more and more on the difficult cases. The first trained subset generates a prediction that gets many instances wrong and is therefore a weak learner. The corresponding badly predicted weights are increased respectively boosted. The second subset now predicts better than the first in these areas. These steps are now repeated. [59] In short: Through the sequential combination of many weak learners a strong learner can be realized.

C.5.4 Additional Note

The content of this chapter C.5 deals very briefly with machine learning. Through a combination of principal component analysis, ensembling learning methods and parameterized learning, different tasks can be solved. However, there are many other applications where machine learning can be very helpful and useful. It is noted that the above mentioned ensemble methods have been used very successfully for many tasks. However, there are still plenty of other types of classification and regression that use the methodologies of conventional machine learning. Among others there are well-known methods like the k-nearest neighbor, Extra-Tree, Voting Classifier, support vector machine and many more. If the reader is interested in further methods, I can highly recommend the Scikit-learn documentation [115]. There are some great examples to the algorithms available. Another very good literature is [59].

C.6 Fundamentals Deep Learning

Many things from today's science have been inspired or motivated by nature. The most famous example is based on the desire of being able to fly like birds. Also the Lotus effect is a well-known example inspired by nature. Another very important example is the brain. The desire to reproduce and apply this structure led to the first artificial neural networks[7]. In this chapter C.6 the basic concepts of a neural network are discussed and different types are considered, starting with an important question: What exactly is a neural network?

C.6.1 What is a Neural Network?

The digits 5, 0, 4 and 1 in figure C.5 are easy for us to classify as human species. Human clearly recognize the 5 as such and can easily distinguish it from the 9. Writing an algorithm that does exactly this, the whole problem will no longer be trivial. The introduction, what is a neural network, is based on this example. Nevertheless, this section C.6.1 was strongly influenced by [116]. As mentioned above, there are many types of neural networks. In this example a **feedforward** neural network is illustrated in figure C.5. Here there is only one direction of construction, namely forward. This structure is also called **multilayer perceptron**. This raises the question of what exactly a **perceptron** is.

By definition, a perceptron is a binary classifier in supervised learning. To illustrate this more clearly, figure C.4 is discussed in more detail. For the represented artificial neuron many inputs x_i exist which are multiplied with a weight w_i. This results in a sum $\mathbf{w}^T\mathbf{x}$, on which an **activation function** is applied. The activation functions is used to map the sum $\mathbf{w}^T\mathbf{x}$ to a desired target set such as $[0, 1]$ or $[-1, 1]$. If a binary classification is performed, the Heaviside step function[8] can be used as a possible activation function. For example, if the input data is assumed to be pixel values of an image, it can be classified whether the image is light (value 1) or dark (value 0). Since the Heaviside step function with the decision whether the sum $\mathbf{w}^T\mathbf{x} \gtreqless 0$ is not a good way to classify an image is bright or dark, an additional bias term is added.

---

[7] It is important to note that artificial neural networks no longer have much to do with their biological twins. This is not tragic either. Even the realization of airplanes today has not much in common with the way birds fly.

[8] $\varphi(x) = \begin{cases} .0, & \text{for } x < 0 \\ 1, & \text{for } x \geqslant 0 \end{cases}$ [117]

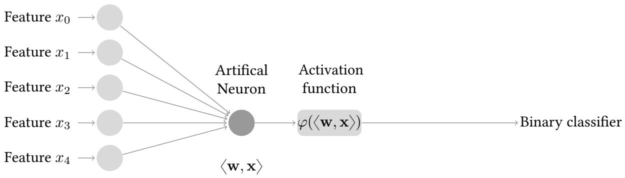


Figure C.4: A single perceptron is shown. Here the activity of the neurons is multiplied by weights and then summed up. The summed result is given as input for an activation function whose output can perform a binary classification. Inspired by [116].

.

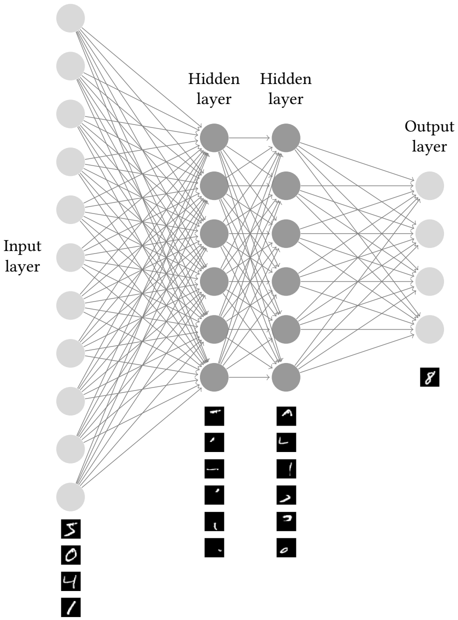


Figure C.5: A multilayer perceptron is illustrated. There is an input layer and an output layer and hidden layers in between. Furthermore the figure shows an example for number classification. The input layer would be the pixel values of the whole image and the output layer the classified number. The output $y^{(L)}$ of each neuron in an arbitrary layer (L) is called activation. The classification is created by assembling neurons from the previous layer, here the second hidden layer. This may include loops and lines that make up a number. The first hidden layer could contain other smaller compositions, like edges. Template from [118].

In this example, a bias term with 0.5 for the classification task bright or dark with $\mathbf{w}^T\mathbf{x} \gtrless 0.5$ is useful. In summary, the output of a single perceptron is called **activation** of an (artificial) neuron [119]. When the artificial neurons are connected as shown in figure C.5, a **multilayer perceptron** is obtained, i.e. a class of feedforward neural networks [119]. After describing a perceptron and a multilayer perceptron, the original example of number classification is described again.

In this example a multilayer perceptron and the activation of an arbitrary artificial neuron in any layer is a value between $[0, 1]$. Our digits, like in example C.1, are taken from the MNIST dataset. This images have only gray value information with the dimension $\mathbf{I} \in \mathbb{R}^{28\times28}_{[0,1]}$. Thus the input layer contains $28 \cdot 28 = 784$ artificial neurons, labeled by $(I)_{0:738}$[9]. Even if not all artificial neurons were drawn in figure C.5 due to space reasons, 10 neurons are to be expected within the output layer $(O)_{0:9}$. The target set of our output layer is therefor a number between 0 and 1 and indicates the activation of this neuron and this in turn indicates which number between 0 and 9 can be related to the input image. Between the input and output layers there are further so called **hidden layers** [120]. If there are two or more hidden layers, it is called a **deep neural network** [59]. Simplified one can say that the activation of the current layer enables the activation of the next layer and the determination of how exactly the activation of one layer causes the activation of the next layer is the core of a neural network, see section C.6.2. For our example this means, that each gray values of the images and the corresponding numbers between 0 and 1 in the input layer (I) activate a very special pattern in the next layer $(H1)$, which in turn creates a very special pattern in layer $(H2)$ and this creates again a very special pattern in the output layer (O). The neuron at the output with the largest activation is the best estimate of the network for recognizing this image. But why are these layers so useful for learning?

When we as humans recognize numbers, individual components are combined with each other. The 9, for example, as a composition of a line and a circle. Another example is the 8, which is built by two circles. Also the 4 which can be interpreted as a special combination of lines. These circles and lines are theoretically found in the hidden layer $(H2)$, see figure C.5. The last step would therefore be to learn which combination

---

[9] The notation $(I)_{s:e}$ where s is the first and e the last used neuron is used. If all neurons of a layer are involved, the notation (I) is used from now on. If only the k-th neuron is considered, a notation with $(I)^k$ is used.

of subcomponents corresponds to which digit. But the whole problem has only been shifted. Recognizing a circle is just as difficult as recognizing the number 0. However, these subcomponents like circles and lines can also be divided into smaller components. One possibility for this would be the detection of smaller edges. For example, take the 0 from figure C.5. This could be built up in maybe four or five parts or edges. These components would again be lines, as they occur for example in the numbers 4 or 7. So summing up, the first hidden layer $(H1)$ could have some different relevant edges. If the network is really trained or works that way is another question. But there is a kind of layer structure to achieve the goal. Returning to our original question. How exactly can one layer activate another? In general, a method is sought that allows us to combine edges from pixel values, patterns from edges and digits from patterns. First of all, an edge description is considered. Therefore only one neuron in $(H1)^k$ is considered. Since every neuron from the input layer (I) is now connected to this one neuron k in $(H1)$, the weighted sum is formed and an activation function is applied. Thus the output for the neuron $(H1)^k$ is given by

$$y^{(H1),k} = \varphi\left(\langle \mathbf{w}_{(I)}^{(H1),k}, \mathbf{x}_{(I)}\rangle + b^{(H1),k}\right) \in [0,1], \tag{C.31}$$

where $b^{(H1),k}$ is the bias term from neuron k in layer $(H1)$ [10]. And this is just valid for one neuron. Every other neuron in the layer $(H1)$ is also connected to all other neurons in the input layer (I) and each one has its own weight and bias term. It is not necessary to use a numerical example to show that many parameters have an influence on the neural network. So learning means finding the right weights and bias terms. A general equation between input layer and first hidden layer can be described by

$$\mathbf{y}^{(H1)} = \varphi\left(\mathbf{W}_{(I)}^{(H1)}\,\mathbf{x}_{(I)} + \mathbf{b}^{(H1)}\right), \tag{C.32}$$

where the activation function is applied to each row and $\mathbf{W}_{(I)}^{(H1)} \in \mathbb{R}^{m\times n}$ where n is the number of neurons in the input layer and m is the number of neurons in the first hidden layer. It is important to mention that a neuron and also the whole network is a function that assigns an output value to many input values. Since this is supervised learning, results are known and used to train the system. In the following steps, the

---

[10] In future the notation $\cdot_{\text{(from layer)}}^{\text{(to layer)}}$ will be used to describe an interaction from one layer to another.

network tries to solve an optimization problem and to learn weights and the bias term in such a way that some performance measure like the mean square error MSE is minimized. The optimization problem can be solved with the gradient descent method[11] or other methods, see section C.6.2. It should be mentioned that the determination of a local minimum in general is “trivial”, but not the determination of a global minimum. It took many years to find a method for training multilayer perceprtons. It was not until 1986 that D. E. Rumelhart et al. published a fundamental paper in which they introduced an algorithm for back-propagation training to effectively determine the necessary gradients [121] [59]. Different types of activation functions can be found in section 3.3.1.

C.6.2 Back-propagation

The described feedforward network learns weights and bias terms by minimizings a cost function. The solution of this optimization task is often solved with the gradient descent method. As the name suggests, the gradients are necessary. Back-propagation is an algorithm for computing this complicated gradient in a very high dimensional space. First, the idea of back-propagation algorithms is explained, before the mathematics is noted afterwards. Index battles can quickly make this unmanageable.

The example from section C.6.1 and figure C.5 is considered. First, it is assumed in a simplified way that only the $(H2)$ and (O) layers interact and that in the first step the weights and bias terms are estimated randomly. Furthermore, the classification for a training example is considered, namely the number 8. Since this is supervised learning, the desired activation of the output layer (O)[12] is known and also the current state of the corresponding activation. An indicator can now be given on how much the weights and bias terms must change. In general, for single neurons in the output layer the weights have to rise, for other neurons they need to fall. Thus there will be a variety of desired weights for each activation in the output layer. What happens is that the desired weights for the neurons are averaged. Once the averaged changes has been determined, they can be applied recursively to the previous layers. This is where the name “backward propagation” comes from. Since this process does not only

---

[11] see algorithm 3.1.

[12] $y^{(O)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$

happen for the number 8, the whole back-propagation routine must be carried out for each further training example. These also have their desired weights, which are also averaged over all training data. This averaged weights and bias term for all training data is proportional to the negative gradient of the cost function. As might be guessed, in practice it takes a very long time for a computer to process the influence of each training example and each gradient descent step. Instead, the training data is randomly mixed and small individual **mini batches** are formed [122]. On these mini batches, the gradient descent step is then determined using back propagation. This does not provide the optimal solution for the optimization problem, but a pretty good approximation and a significant computational speed up. This procedure is then implemented and applied for all other mini batches and this is called **stochastic gradient descent** [123]. This describes the whole idea behind this topic. The appropriate code for back-propagation can be found in [124]. For the math-loving reader, the following is a mathematical description of back-propagation. This is necessary since back-propagation describes the heart of a neural network. It should be mentioned that the description is not a mathematical proof. For a more detailed description, the paper from D. E. Rumelhart et al. is recommended [121].

To simplify the index battle, the notation from figure C.6 is used. First, all variables mentioned in figure C.6 are described.

$y_k^{(L-1)}$: Activation of neuron k in Layer (L-1)
$y_j^{(L)}$: Activation of neuron j in Layer (L)
$\tilde{y}_j$: Desired activation of neuron j in the last Layer (L)
$w_{(L-1),k}^{(L),j}$: Weight of neuron k in layer $(L-1)$ to neuron j in (L)
C_j : Cost value between $\tilde{y}_j$ and $y_j^{(L)}$

Furthermore, the number of neurons in any arbitrary layer $(\cdot)$ is noted with $n^{(\cdot)}$. Our goal is to minimize the cost $C = \sum_{j=0}^{n^L-1} C_j$ with[13]

$$C_j = \left(\tilde{y}_j - y_j^{(L)}\right)^2 \tag{C.33}$$

$$= \left(\tilde{y}_j - \varphi\left(w_{(L-1),0}^{(L),j} \cdot y_0^{(L-1)} + \ldots + w_{(L-1),n^{L1}-1}^{(L),j} \cdot y_{n^{L1}-1}^{(L-1)} + b_j^{(L)}\right)\right)^2 \tag{C.34}$$

$$= \left(\tilde{y}_j - \varphi\left(\underbrace{\sum_{k=0}^{n^{L1}-1} w_{(L-1),k}^{(L),j} \cdot y_k^{(L-1)} + b_j^{(L)}}_{=: z_j^L}\right)\right)^2 . \tag{C.35}$$

Thus the following dependencies apply to C_j

$$C_j \rightarrow \left(\tilde{y}_j \quad y_j^{(L)}\right) \rightarrow \left(\tilde{y}_j \quad z_j^{(L)}\right) \rightarrow \left(\tilde{y}_j \quad \left(w_{(L-1),k}^{(L),j}, y_k^{(L-1)}, b_j^{(L)}\right)\right) \tag{C.36}$$

$$\rightarrow \left(\tilde{y}_j \quad \left(w_{(L-1),k}^{(L),j}, z_k^{(L-1)}, b_j^{(L)}\right)\right) \tag{C.37}$$

$$\rightarrow \left(\tilde{y}_j \quad \left(w_{(L-1),k}^{(L),j}, \left(w_{(L-2),l}^{(L-1),k}, y_l^{(L-2)}, b_k^{(L-1)}\right), b_j^{(L)}\right)\right) \tag{C.38}$$

$$\rightarrow \cdots ,$$

but how do the dependencies relate to each other? What are the changes to the cost function when the weights vary? The application of the chain equation yields to

$$\frac{\partial C}{\partial w_{(L-1),k}^{(L),j}} = \frac{\partial z_j^{(L)}}{\partial w_{(L-1),k}^{(L),j}} \cdot \frac{\partial y_j^{(L)}}{\partial z_j^{(L)}} \cdot \frac{\partial C_j}{\partial y_j^{(L)}} \tag{C.39}$$

$$= \left(y_k^{(L-1)}\right) \cdot \left(\frac{\partial \varphi\left(z_j^{(L)}\right)}{\partial z_j^{(L)}}\right) \cdot \left(2\left(y_j^{(L)} - \tilde{y}_j\right)\right) . \tag{C.40}$$

---

[13] The cost function depends on all weights and bias terms. For reasons of clarity, these are not written down.

Equation C.40 does not cause us major problems since all factors can be determined. The result of the second multiplicand depends on which activation function was used. However, the dependency also affects the bias term and the activation of the previous layer. The following applies to the bias term

$$\frac{\partial C}{\partial b_j^{(L)}} = \frac{\partial z_j^{(L)}}{\partial b_j^{(L)}} \cdot \frac{\partial y_j^{(L)}}{\partial z_j^{(L)}} \cdot \frac{\partial C_j}{\partial y_j^{(L)}} \tag{C.41}$$

$$= (1) \cdot \left(\frac{\partial \varphi \left(z_j^{(L)} \right)}{\partial z_j^{(L)}} \right) \cdot \left(2 \left(y_j^{(L)} - \tilde{y}_j \right) \right) \tag{C.42}$$

and for the activation in the previous layer

$$\frac{\partial C}{\partial y_k^{(L-1)}} = \sum_{j=0}^{n^L-1} \frac{\partial z_j^{(L)}}{\partial y_k^{(L-1)}} \cdot \frac{\partial y_j^{(L)}}{\partial z_j^{(L)}} \cdot \frac{\partial C_j}{\partial y_j^{(L)}} \tag{C.43}$$

$$= \left(\sum_{j=0}^{n^L-1} w_{(L-1),k}^{(L),j} \right) \cdot \left(\frac{\partial \varphi \left(z_j^{(L)} \right)}{\partial z_j^{(L)}} \right) \cdot \left(2 \left(y_j^{(L)} - \tilde{y}_j \right) \right). \tag{C.44}$$

The sum in equations C.43 and C.44 emerges from the fact that the influence of the activation of neuron k in layer $(L-1)$ has an influence on all neurons in the last layer (L). This influence is also well illustrated in figure C.6. Since the gradient descent can be determined by deriving the cost function according to all weights and bias terms using the chain rule expressions from equation C.40 to C.44, back-propagation is clearly possible. Once determining the gradient, finding the minimum cost can be realized by the gradient descent method. Regularization methods specifically used for neural networks can be found in section 3.3.1.

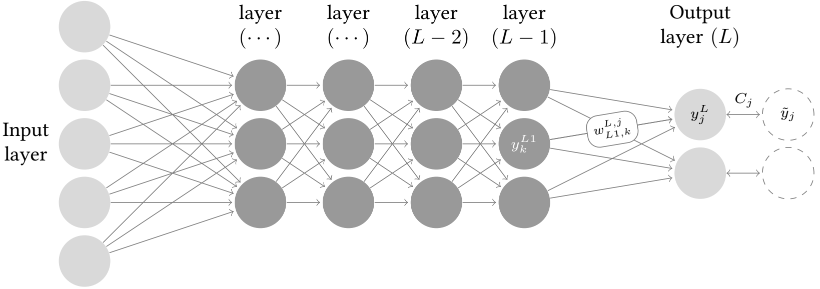


Figure C.6: A multilayer perceptron is shown. The activity of neuron k in any layer $(\cdot)$ is noted by $y_k^{(\cdot)}$. Weighting between any two neurons k and j from different layers is given by $w_{(\cdot-1),k}^{(\cdot),j}$. Cost function between desired activity of the neuron $\tilde{y}_j$ and the actual activity of the neuron in the last layer $\tilde{y}_j^{(L)}$ is defined by C_j. The cost function, $C = \sum_{j=0}^{n^L-1} C_j$, where $n^{(L)}$ is the number of neurons in layer (L). Template by [118].

C.7 Fundamentals of Convolution Neural Network

Convolution network or convolution neural network (CNN) is a special type of neutral networks that is often used for grid-like structure. Unlike multilayer perceptron, the input is left unchanged in its structure. When an image is loaded with 28×28 pixels, no input layer with $28 \cdot 28$ neurons is generated. The structure remains unchanged, even if e.g. an RGB image is present. In section 3.1.1 convolution has already been discussed. This term is extended to describe the convolution matrix or **kernel**. After that, some terms of a convolution layer are described. At the end of this section C.7 convolution transposed is described. A typical architecture of a convolution neural network can be found in section 3.4.1.

C.7.1 Convolution Matrix

Definition 3.1.4 and definition 3.1.5 shows the continuous convolution and cross correlation. For the discrete case the following definition follows.

Definition C.7.1 (Discrete Convolution and Cross Correlation): Let $x, w\colon D \to \mathbb{C}$ be two functions with $D \subset \mathbb{Z}$ discrete. The discrete convolution is defined by [125]

$$(x * w)(n) = \sum_{k \in D} x(k) w(n-k) \tag{C.45}$$

and the discrete cross correlation with [126]

$$(x \star w)(n) = \sum_{k \in D} \bar{x}(k) w(n+k) \tag{C.46}$$

where $\bar{x}(k)$ is the conjugated complex part of $x(k)$.

Our input for CNNs are often images. If it is a grayscale image, a two-dimensional kernel $\mathbf{W}$ is required. For the corresponding extended convolution and its commutative property

$$\begin{aligned}(\mathbf{X} * \mathbf{W})(n,m) &= \sum_{k \in D_1} \sum_{l \in D_2} \mathbf{X}(k,l) \mathbf{W}(n-k, m-l) \\ &= \sum_{k \in D_1} \sum_{l \in D_2} \mathbf{W}(k,l) \mathbf{X}(n-k, m-l)\end{aligned} \tag{C.47}$$

holds. Since inputs for CNNs are often real numbers, the conjugated complex part is omitted for cross-correlation. Furthermore, the commutative property is often useful to perform proofs, but it is not an important property in neural networks. For the cross-correlation in our case

$$(\mathbf{X} \star \mathbf{W})(n,m) = \sum_{k \in D_1} \sum_{l \in D_2} \mathbf{X}(n+k, m+l) \mathbf{W}(k,l) \tag{C.48}$$

can be concluded. All implementations of common machine learning libraries are implemented with cross correlation, but called convolution [56]. We will also follow this convention. The convolving of a 3×3 kernel $\mathbf{W}$ over a 5×5 input $\mathbf{X}$ is illustrate in Figure C.7 (left).

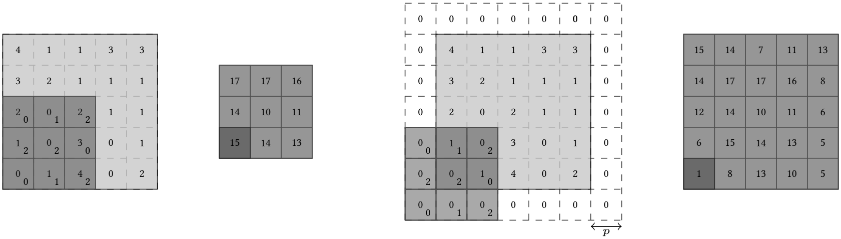


Figure C.7: In the left illustration the convolving of a 3×3 kernel over a 5×5 input is shown. The right figure shows the same convolution, but the input data was previously extended with zeros. This procedure is called zero padding. Illustration is inspired by [127].

C.7.2 Convolution Layer

The convolution layer is the most important pillar in a CNN. Here not all neurons from the input data, only local regions, are connected to the previous convolution layer, see also figure C.8. Each neuron in the next layer is also connected to only one local region from the previous layer. This ordered structure makes it possible to realize so called "low-level" features like edge detection in the first layer and to combine them in the second layer, which maybe includes "mid-level" features or "high-level" features. This hierarchical structure is an important reason why CNN works so well in image recognition. There are three main parts for this important convolution layer [128]:

$$\begin{aligned}\text{Convolution stage} &\rightarrow \text{Detector stage (Nonlinearity like e.g. ReLU }^{14}\text{)}\\ &\rightarrow \text{Pooling stage}\end{aligned}$$

The detector stage is the insertion of a nonlinearity using an activation function. The other two steps, convolution and pooling stage, and there possible setting will be explained in more detail.

Convolution Stage The convolution stage has already been described in section C.7.1 and can be found in equation C.48. In Convolution, the input $\mathbf{X}$ is convolted with a kernel $\mathbf{W}$ and the weights are now present in this kernel $\mathbf{W}$[15]. However, as shown in the left figure C.7, the dimension of the output decreases after convolution. Even more, the information from the edge image area is neglected. One possibility to prevent this is the introduction of **zero padding** [129]. This method adds zero to the input, see also the right illustration in figure C.7. This increases the dimension of the input and the edges of the input are considered more strongly than before. It should be mentioned that zero padding can be designed differently in all existing dimensions. In Figure C.7 $p = 1$ was selected for both directions. Also the dimension of the output does not necessarily have to be smaller or equal than the input. Using $p = 2$ for figure C.7 would increase the dimension of the output.

Stride The discrete cross correlation in equation C.48 describes the sum over a discrete set D_1 and D_2. This set does not always need to have a step size of 1, nor does it need to be theoretically equidistant. The variation of this step size parameter is called **stride** s in CNNs [130]. An illustration can be found in figure C.8. Caution is also advised, not every parameter is allowed, e.g. $s = 3$ in figure C.8. Also the dimension of the output can change. This fact is shown in figure C.8 for $s = 2$. Here it should be noted that the stride parameter can be different in all possible directions.

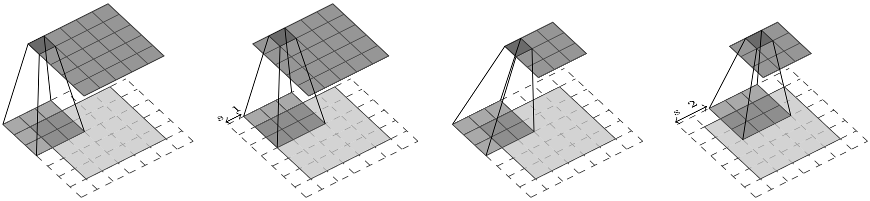


Figure C.8: The figure illustrate the convolution operation for a stride of 1 and a stride of 2. By choosing a stride of $s = 2$ the dimension of the output is reduced. A stride of $s = 3$ would be invalid in this example. Illustration is inspired by [127].

Feature map More than one filter is needed to detect all different types of patterns in an image. In order to extract different local features of a certain region of the input, several filters are necessary. These filters are stored in an **activation map** or **feature map** and can be passed on to the following layers to recognize more complex patterns

---

[15] Because of its filtering property, the kernel is also called filter.

[131]. The depth of the activation map is the number of used filters. Within one feature map, all neurons share the same parameters (weights and bias term) [59]. This is also valid if the input has a “more complex” structure. Images, e.g. are usually a third-order tensor. They have further information like RGB values compared to grayscaled values. In summary, a neuron from a particular feature map has a connection to a local region, but across all depths of the layer (see figure C.9) [132].

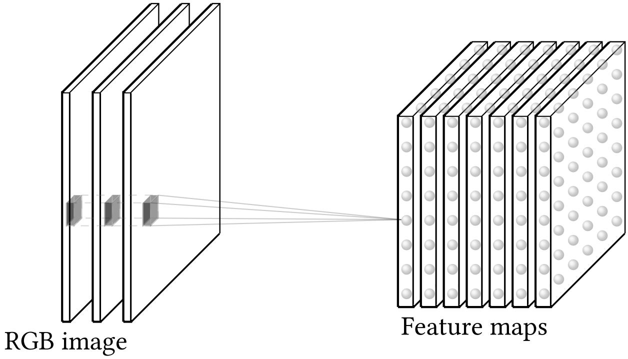


Figure C.9: One neuron from one feature map is assigned to an area of the previous layer (here the input layer). This local area belongs to all depths of a structure. Furthermore, all neurons of one feature map share the same weights and bias term. The number of layers in a feature map is determined by the number of applied filters. Illustration is inspired by [133].

Pooling stage As already mentioned, one layer of a convolutional network often consists of three stages. The first two, convolution and detector stage, were discussed. The third step uses the pooling function to manipulate the output [56]. The goal of pooling is to subsample the current layer, to reduce computational load, memory usage and the number of parameters. Furthermore, the pooling operation helps to make the image more tolerant to small image shifts [59]. The most popular known pooling methods are max pooling and average pooling. Both can be seen in figure C.10. It should be noted that the depth remains unchanged for the output layer.

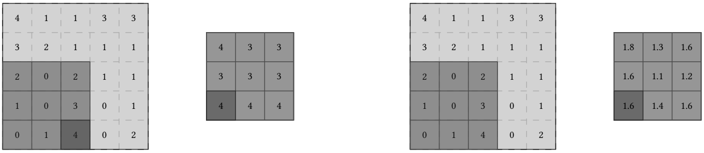


Figure C.10: Two common pooling methods, max pooling and average pooling are illustrated. Max pooling displays the maximum value in the overlapping area between pooling operator and input. Average pooling calculates the mean value of the overlapping area. Illustration is inspired by [127].

Commonly used convolution architectures that use the above techniques are described in section 3.4.1.

C.7.3 Convolution Transposed

What happened so far in section C.7 is that an image was encoded. In other words, the image has been reduced in important feature regions. But what if an image is supposed to be decoded? For this task **convolution transposed** is suitable. This goes in the opposite direction of a normal convolution, i.e., from something that has the shape of the output to something that has the shape of its input while maintaining a pattern [127]. Figure C.11 illustrates this fact and gives the link between ordinary and convolution transposed.

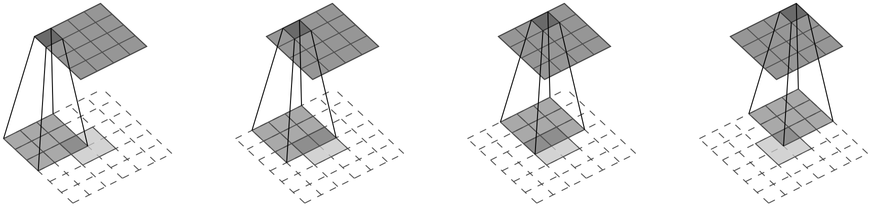


Figure C.11: The convolving of a 3×3 kernel over a 2×2 input padded with a 2×2 border of zeros using unit strides **or** the transpose of convolving of a 3×3 kernel over a 4×4 input. Illustration is inspired by [127].

C.8 Autoencoder

An autoencoder is a neural network that is trained to attempt to copy its input to its output [56]. This might sound a bit irrelevant at first but may result in a lot of useful information. For example, the step of encoding and decoding the input can lead to efficient representations of the input sequence without any supervised learning. Therefore the training data are unlabeled and decoding allows us to reduce the dimension, just like PCA does, but with nonlinearity. The following describes how autoencoders work with multiplayer perceptron. The handling for CNN is similar. These are discussed in section 3.4.2.

C.8.1 Deep Autoencoders

Just like neural networks, autoencoders consists of several hidden layers. If this is the case, it is also called a stacked or **deep autoencoder**. If more hidden layers are added, the autoencoder is able to detect more complex patterns. However, there is a danger that this will become too powerful. The autoencoder then perfectly copies the output to the input, but is unusable for new data and does not produce useful results during the process. A typical architecture of a deep autoencoder can be seen in figure C.12. [59]

Autoencoders based on convolutional neural networks are described in section 3.4.2.

C.8.2 Tying Weights

If an autoencoder is symmetrical, as shown in Figure C.12, the common technique is to connect the weights of the encoder with the weights of the decoder. This halves the number of weights and trivially minimizes the risk of overfitting [59]. To mathematically illustrate the relationship between the encoding and decoding weights, it is assumed that there are N many layers, without the input layer[16]. Let the weights from layer $(L-1)$ to (L) be $\mathbf{W}_{(L-1)}^{(L)}$, than the corresponding weights for the decoding part is $\mathbf{W}_{(N-L-1)}^{(N-L)} = \left(\mathbf{W}_{(L-1)}^{(L)}\right)^T$.

---

[16] N will therefore always be even, because of the symmetric structure.

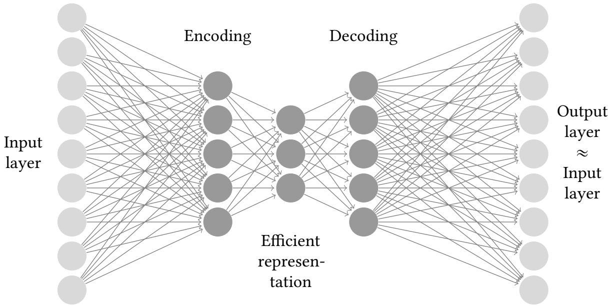


Figure C.12: A typical structure of an autoencoder is shown. Here the input layer is encoded and brought to an efficient representation. This efficient representation is also called **bottle neck** and will be decoded in the next steps. Typically, the output layer is identical to the input layer or a good approximation of it. Template from [118].

C.8.3 Denoising Autoencoders

To prevent the autoencoder from learning the copy function, noise is added to the input data to achieve noise-free output data. Denoising autoencoders are thus an example of how useful properties can emerge as a by-product of minimising reconstruction errors [56]. They are also an example of how over-complete, high-capacity models can be used as autoencoders as long as care is taken to ensure that they do not learn the identity function [56]. Another advantage of this method is that the desired patterns must be learned more effectively. Another method of inserting noise or changing the input data is to randomly switch off input neurons, similar to dropout. This method is even more common in practice. Figure C.13 summarized both methods for denoising autoencoders [59].

C.8.4 Other Autoencoders

Unsupervised learning with autoencoder has been somewhat neglected in recent years due to the success of supervised learning and its architectures. However, something is happening here as well. The following is a small extract, maybe important for your future work[59]:

- Contractive autoencoder [134].

- Stacked convolutional autoencoders [135].
- Generative stochastic network (a good way to generate data) [136].
- Winner-take-all autoencoder [137].
- Adversarial autoencoders [138].

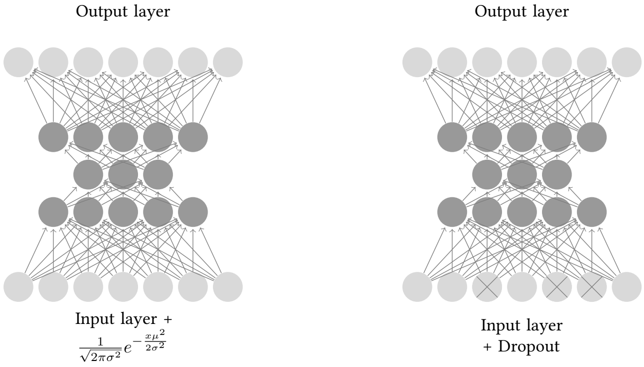

Output layer
Output layer
Input layer +
$\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{x\mu^2}{2\sigma^2}}$
Input layer
+ Dropout

Figure C.13: Two denoising autoencoder methods are presented. In the left figure the input data is overlaid with noise. In the right figure, random neurons are removed from the input layer. Template from [118].

List of Supervised Student Theses

In the following you will find a selection of the student theses I supervised during my time as research assistant at the chair of measurement and sensor technology under Prof. Dr.-Ing. Jörg Seewig. The theses in bold print and their contents had an influence on the dissertation.

Adil Kiper	Master thesis	AI und Deep Learning in der medizinischen Computertomographie: Eine Überführung auf indistrielle Computertomographie im Zuge von Industrie 4.0
Arsalan Jawaid	**Bachelor thesis**	**Vergleich zwischen klassischen Methoden und neuronalen Netzen bei der Einpassung von geometrischen Körpern**
Bastian Schuh	Master thesis	“Sensorlose” Echtzeitpositionsrekonstruktion für einen elektromagnetischen Aktuator
Christian Brill	Diplom thesis	Entwurf und Auswertealgorithmen für Telemetriemessdaten selbstfahrender Erntemaschinen und Ableitung von Lastzyklen
Christian Fischer	Project thesis	Reduktion der Störschwingung durch Geschwindigkeitsmodifikation bei Linearvorschüben zur taktilen Rauheits- und Konturmessung
Dorothea Kölsch	**Diploma thesis**	**Three-Dimensional reconstruction of Measurement Data Using Artifical Neural Network**
Dorothea Kölsch	Project thesis	Entwurf und Realisierung eines Demonstrators für elektrische Schaltungen
Elias Rocklage	Project thesis	Automated Traffic Scenario Generation for Autonomous Vehicle Evaluation
Hikmet Büsra Sahin	Bachelor thesis	Entwicklung von optischen Design Programmen zum Entwurf und Analysieren eines Ellipso-Höhentopometers

Jingyuan Wu	Master thesis	Evaluierung eines Partikelfilters zur Lokalisierung eines Smartphones unter Berücksichtigung von GNSS Rohdatenmessungen und IMU Messungen
Johannes Klunk	Bachelor thesis	Entwurf und Realisierung eines Überholmanövers für autonome Modellfahrzeuge mit Verfahrender Optimaler regelung und eines kalman-Filters
Johannes Merseburg	Diploma thesis	B-View Rundum-Kamerasystem
Katharina Umber	Master thesis	Implementierung einer auf der Radon Transformation basierenden Mikrodrall Charakterisierung für geschliffene Oberflächen in C++
Lidiia Glemba	Master thesis	Entwicklung einer kraftbasierten Geschwindigkeitsregelung zur Steuerung des Leichtbauroboters KUKA LBR iiwa im handgeführten Modus
Lidiia Glemba	Project thesis	Entwicklung eines Oberflächenmessgerätes auf der Basis einer Beschleunigungsmessung
Lukas Müller	Project thesis	Aufbau und Inbetriebnahme einer Erdbebensimulation zur Verwendung in der Lehre
Lukas Müller	Project thesis	Aufbereitung und nachhaltige Weiterentwicklung der Codebasis eines autonomen Tischkickers
Marius Holdstein	Project thesis	Gestaltung, Implementierung und Validierung einer Auswertesoftware zur Charakterisierung von Mikrodrall an geschliffenen Oberflächen

Max Gieselmann	Diploma thesis	Prototyp eines selbstlernenden Torwarts in einem autonomen Tischenfußballspiel mittels bestärkendem Lernen mit künstlichen neuronalen Netzwerken als Funktionsapproximation
Maximilian Theobald	Master thesis	Konzeptentwicklung und Konstruktion einer flexiblen, mechanischen Funktionseinheit zur Verbindung zweier neuartiger Landmaschinen
Robert Kranz	Master thesis	Positionserfassung für die 3D-Terahertz Bildgebung
Samuel Schmidt	**Project thesis**	**Entwicklung eines Convolutional Autoencoder als Anomaliendetektor**
Samuel Schmidt	Master thesis	Simulation eines Stereo-CIS-Sensors und Implementierung von Stereoalgorithmen zur Prozessüberwachung eines 3d-Metalldruckers
Tim Becker	Project thesis	Entwurf und Realisierung einer Lenk- und Abstandsregelung für ein autonom geführtes Modellfahrzeug

List of previous publications

A. Karatas, “Determination of texture directions for ground surfaces with a resolution below one arc minute using CNNs,”, in Medium. [Online]. Available: Medium Link

A. Karatas, D. Koelsch, S. Schmidt, M. Eifler, and J. Seewig, “Development of a convolutional autoencoder using deep neuronal networks for defect detection and generating ideal references for cutting edges,”, in SPIE Proceedings, vol. 11056, 2019.

A. Karatas, and J. Seewig, “Generating ideal reference geometry for cutting edges”, in Journal of Physics: Conference Series, vol. 1065, p. 142017, 08 2018.

E. Rocklage, H. Kraft, A. Karatas, and J. Seewig, “Automated scenario generation for regression testing of autonomous vehicles”, in 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 2017, pp. 476–483.

F. M. Torner, A. Karatas, M. Eifler, I. Raid, and J. Seewig, “Application of GPU-based,highly parallelized algorithms for the estimation of electromagnetic multi-layerinteractions”, in Optical Measurement Systems for Industrial Inspection X, P. Lehmann, W. Osten, and A. A. G. Jr., Eds., vol. 10329, International Society for Optics and Photonics. SPIE, 2017, pp. 521–535.[Online]. Available: `https://doi.org/10.1117/12.2270257`

Bibliography

[1] MBN 31007-7:2009-04, Geometrische Produktspezifikationen (GPS) - Oberflaächenbeschaffenheit - Mess- und Auswerteverfahren zur Bewertung von drallreduzierten dynamischen Dichtflaächen. Mercedes-Benz.

[2] M. Baumann, “Abdichtung drallbehafteter dichtungsgegenlaufflächen - messung, analyse, bewertung und grenzen,” doctoralthesis, Stuttgart : Institut für Maschinenelemente, 2017, accessed: 2021-12-17. [Online]. Available: https://elib.uni-stuttgart.de/handle/11682/9078

[3] G. Kersten, “Optische und antastende prüfung der gegenlauffläche von radialwellendichtringen,” doctoralthesis, Universität Hannover, 1992.

[4] D. Krahe, “Zerstörungsfreie prüfung der textur gehonter und geschliffener oberflächen,” Ph.D. dissertation, 2000.

[5] J. Seewig and T. Hercke, “Lead characterisation by an objective evaluation method,” Wear, vol. 266, p. 530–533, 03 2009.

[6] P. Arnecke, “A measurement method for characterising micro lead on ground shaft surfaces,” doctoralthesis, Technische Universität Kaiserslautern, 2017, accessed: 2021-12-17. [Online]. Available: http://nbn-resolving.de/urn:nbn:de:hbz:386-kluedo-46744

[7] N. Rau and M. Seibold, “Drallstrukturen geschliffener dichtflächen beurteilen, reviewing twist structures of ground sealing surfaces, werkstatt und betrieb,” Werkstatt und Betrieb, vol. 130, no. 11, pp. 1013–1016, 1997,

accessed: 2021-12-17. [Online]. Available: https://www.tib.eu/de/suchen/id/tema%3ATEMAM97120108533

[8] P. Arnecke and J. Seewig, "A strategy for micro-twist characterization on the shaft in a rotary shaft sealing system," in 18th ISC, International Sealing Conference, Internationale Dichtungstagung, 2014, pp. 654–666.

[9] A. Karatas. Determination of texture directions for ground surfaces with a resolution below one arc minute using cnns. Medium. Accessed: 2021-12-17. [Online]. Available: https://medium.com/@abdullah.karatas/determination-of-texture-directions-for-ground-surfaces-with-a-resolution-below-one-arc-minute-870923c47b72

[10] J. Seewig and T. Hercke, "2nd generation lead measurement," 2009.

[11] Rau, Norbert, Seibold, Michael, and Heilbronner, Robert, "Verfahren zur ermittlung einer drallstruktur in der oberflächenrauheit eines feinbearbeiteten wellenzapfens," DE19740141C1, sep 1997.

[12] F. Puente Leó n, "Drallerkennung an gegenlaufflächen von radialwellendichtringen," in Bildverarbeitung im industriellen Einsatz, ser. VDI-Berichte, vol. 1572. Düsseldorf: VDI Verlag, 2000, pp. 117–122.

[13] S. Buhl, Wechselbeziehungen im Dichtsystem von Radial-Wellendichtring, Gegenlauffläche und Fluid. Ed. by Forschungsvereinigung Antriebstechnik e.V., 2000.

[14] T. Kunstfeld, "Einfluss der wellenoberfläche auf das dichtverhalten von radialwellendichtungen," doctoralthesis, Stuttgart : Institut für Maschinenelemente, 2005, accessed: 2021-12-17. [Online]. Available: https://elib.uni-stuttgart.de/handle/11682/4074

[15] Cohen, Donald K., Smith, Stanley N., Novak Erik L., and Masters, Andrew T., "Quantitatively Measuring Surface Texture and Shaft Lead of Dynamic Sealing Systems," http://www.ai-online.com/Adv/Previous/show_issue.php?id=4401, accessed: 2021-07-30.

[16] J. J. Hull, “Document image skew detection: Survey and annotated bibliography,” in Document Analysis Systems II. Word Scientific. World Scientific, 1998, pp. 40–64.

[17] “Geometrical product specifications (gps) — surface texture: Areal — part 2: Terms, definitions and surface texture parameters,” International Organization for Standardization, Geneva, CH, Standard, apr 2012.

[18] Deutsches Institut für Normung e.V.: DIN EN ISO 25178-2:2012-09, Geometrische Produktspezifikation (GPS) - Oberflächenbeschaffenheit: Flächenhaft - Teil 2: Begriffe und Oberflächen-Kenngrößen. Beuth Verlag, Berlin, 2012.

[19] R. Leach, Characterisation of areal surface texture, 06 2013.

[20] G. Baitinger, “Multiskalenansatz mit mikrostrukturanalyse zur drallbeurteilung von dichtungsgegenlaufflächen,” doctoralthesis, 2011.

[21] C. M. Shakarji, “Least-squares fitting algorithms of the nist algorithm testing system,” in Journal of Research of the National Institute of Standards and Technology, 1998, pp. 633–641.

[22] Deutsches Institut für Normung e.V.: DIN EN ISO 8785:1999-10, Geometrische Produktspezifikation (GPS) - Oberflächenunvollkommenheiten - Begriffe, Definitionen und Kenngrößen (ISO 8785:1998). Beuth Verlag, Berlin, 1999.

[23] E. Prem and R. Vogt, “Der simmerring - grundlagen zur schaden- sprävention: Firmenschrift, freudenberg simrit gmbh & co. kg, weinheim,” mar 2008.

[24] Rau, Norbert, Kruppke, Volker, and Seibold, Michael, “Messtechnische beobachtungen zum funktionsverhalten von gedrehten dichtflächen,” in Sealing systems for fluid power applications: sealing technology - a global challenge. Fachverband Fluidtechnik im VDMA, 2004, p. 93–105.

[25] Deutsches Institut für Normung e.V.: DIN EN ISO 25178-6:2010-06, Geometrische Produktspezifikation (GPS) – Oberflächenbeschaffenheit: Flächenhaft – Teil 6: Klassifizierung von Methoden zur Messung der Oberflächenbeschaffenheit (ISO 25178-6:2010). Beuth Verlag, Berlin, 2010.

[26] N. AG, “Berührungsloses 3d-oberflächen-messsystem: µsurf explorer,” 2007.

[27] “Cutting edge rounding,” https://de.wikipedia.org/wiki/Schneidkantenverrundung, accessed: 2021-07-30.

[28] “Cutting part,” https://de.wikipedia.org/wiki/Schneidteil#Fl%C3%A4chen,_Schneiden_und_Ecken, accessed: 2021-07-30.

[29] Deutsches Institut für Normung e.V.: DIN 6581, Begriffe der Zerspantechnik; Bezugssysteme und Winkel am Schneidteil des Werkzeuges.

[30] Y.-C. Yen, A. Jain, and T. Altan, “A finite element analysis of orthogonal machining using different tool edge geometries,” Journal of Materials Processing Technology, vol. 146, pp. 72–81, 02 2004.

[31] F. Tikal, R. Bienemann, and L. Heckmann, Schneidkantenpräparation - Ziele, Verfahren und Messmethoden: Berichte aus Industrie und Forschung. Tikal, Franz, Kassel University Press, 2009.

[32] B. Schaffer, “Advances in edge preparation offer production advantages,” Tools & Production, p. 14–16, 2005.

[33] B. Denkena, T. Friehmuth, S. Fedorenke, and M. Groppe, “An der schneide wird das geld verdient-neue parameter zur charakterisierung der schneidengeometrien an zerspanwerkzeugen, fertigung,” Sonderausgabe Werkzeuge, vol. 12, pp. 24–26, 01 2002.

[34] B. Denkena and P. Baumann, “Lasertechnologie für die generierung und messung der mikrogeometrie an zerspanwerkzeugen : Ergebnisbericht des bmbf-verbundprojektes geospan ; projektzeitraum 01.01.2002 - 31.03.2005,” Projektträgerschaft Produktion und Fertigungstechnologien, Garbsen, Tech. Rep., 2005, accessed: 2021-12-17. [Online]. Available: https://www.tib.eu/de/suchen/id/TIBKAT%3A510576680

[35] K. Risse, “Einflüsse von Werkzeugdurchmesser und Schneidkantenverrundung beim Bohren mit Wendelbohrern in Stahl,” Ph.D. dissertation, Aachen, 2006, aachen, Techn. Hochsch., Diss., 2006. [Online]. Available: https://publications.rwth-aachen.de/record/61346

[36] B. Denkena, N. Kramer, F. Siegel, and J. Kästner, “Leistungsoptimierung an der schneidkante,” VDI-Z, Special Werkzeuge, August 2007, S. 24-26, 2007, 2007, accessed: 2021-07-30. [Online]. Available: http://www.magnetfinish.com/assets/leistungsoptimierung-an-der-schneidkante-2007.pdf

[37] R. Danzl, F. Helmli, and S. Scherer, “Focus variation – a robust technology for high resolution optical 3d surface metrology,” Strojniški vestnik - Journal of Mechanical Engineering, vol. 57, no. 3, pp. 245–256, 2011, accessed: 2021-12-17. [Online]. Available: https://www.sv-jme.eu/article/focus-variation-a-robust-technology-for-high-resolution-optical-3d-surface-metrology/

[38] Clemens Helfmeier, “Focused ion beam system,” https://texample.net/tikz/examples/focused-ion-beam-system/, accessed: 2021-10-21.

[39] A. Karatas, D. Koelsch, S. Schmidt, M. Eifler, and J. Seewig, “Development of a convolutional autoencoder using deep neuronal networks for defect detection and generating ideal references for cutting edges,” SPIE Proceedings, vol. 11056, 2019.

[40] R. Danzl and F. Helmli, “Automatische 3d-auswertung der schneidkantenqualität durch fokus variation,” in Internationales Oberflächenkolloquium - XIV. International Colloquium on Surfaces, 2017.

[41] A. Karatas and J. Seewig, “Generating ideal reference geometry for cutting edges,” Journal of Physics: Conference Series, vol. 1065, p. 142017, 08 2018.

[42] C. De Boor, A practical guide to splines; rev. ed., ser. Applied mathematical sciences. Berlin: Springer, 2001, accessed: 2021-12-17. [Online]. Available: https://cds.cern.ch/record/1428148

[43] F. Liang, “Constrained least squares,” https://www2.stat.duke.edu/courses/Spring06/sta293.3/topic5/spline.pdf, 2006.

[44] Analog Devices, Inc., “Convolution,” https://www.analog.com/media/en/technical-documentation/dsp-book/dsp_book_Ch6.pdf, accessed: 2021-10-25.

[45] I. Hirschman and D. Widder, The Convolution Transform, ser. Dover Books on Mathematics. Dover Publications, 2012.

[46] R. Bracewell, Fourier Analysis and Imaging. Springer US, 2004.

[47] H. Weber and H. Ulrich, Laplace-, Fourier- und z-Transformation: Grundlagen und Anwendungen für Ingenieure und Naturwissenschaftler. Vieweg+Teubner Verlag, 2011.

[48] P. Toft, The Radon Transform: Theory and Implementation. Department of Mathematical Modelling, Section for Digital Signal Processing, Technical University of Denmark, 1996, accessed: 2021-12-17. [Online]. Available: https://books.google.de/books?id=s7EPYAAACAAJ

[49] "Radon Definition," https://statweb.stanford.edu/~candes/math262/Lectures/Lecture09.pdf, accessed: 2019-07-19.

[50] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures," Analytical Chemistry, vol. 36, pp. 1627–1639, 1964.

[51] "Savitzky-Golay Filter coefficients," https://www.wire.tu-bs.de/OLDWEB/mameyer/cmr/savgol.pdf, accessed: 2019-07-19.

[52] J. Seewig, "Linear and robust gaussian regression filters," Journal of Physics: Conference Series, vol. 13, pp. 254–257, jan 2005, accessed: 2021-12-17. [Online]. Available: https://doi.org/10.1088%2F1742-6596%2F13%2F1%2F059

[53] O. Stein, Grundzüge der Nichtlinearen Optimierung. Springer, Berlin, 2018.

[54] "Stochastic Gradient Descent," https://en.wikipedia.org/wiki/Stochastic_gradient_descent, accessed: 2019-05-28.

[55] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," International Conference on Learning Representations, 12 2014.

[56] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016, http://www.deeplearningbook.org.

[57] "Goodle Machine Learning Glossary," https://developers.google.com/machine-learning/glossary, accessed: 2019-05-28.

[58] "Scikit-learn Documentation," https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html, accessed: 2019-07-20.

[59] A. Gron, Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 1st ed. O'Reilly Media, Inc., 2017.

[60] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," CoRR, vol. abs/1511.07289, 2016.

[61] J. Brownlee. A gentle introduction to early stopping to avoid overtraining neural networks. Machine Learning Mastery. Accessed: 2021-12-17. [Online]. Available: https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/

[62] ——. Use early stopping to halt the training of neural networks at the right time. Machine Learning Mastery. Accessed: 2021-12-17. [Online]. Available: https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/

[63] M. Siddhartha, "Regularization techniques in deep learning," 2019, accessed: 2021-12-17. [Online]. Available: https://www.kaggle.com/sid321axn/regularization-techniques-in-deep-learning

[64] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," CoRR, vol. abs/1207.0580, 2012, accessed: 2021-12-17. [Online]. Available: http://arxiv.org/abs/1207.0580

[65] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[66] J. Tan, "Tikz cnn," https://github.com/jettan/tikz_cnn/, 2020, accessed: 2021-12-17.

[67] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105, accessed: 2021-12-17. [Online].

Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[68] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," CoRR, vol. abs/1409.4842, 2014, accessed: 2021-12-17. [Online]. Available: http://arxiv.org/abs/1409.4842

[69] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," CoRR, vol. abs/1512.03385, 2015, accessed: 2021-12-17. [Online]. Available: http://arxiv.org/abs/1512.03385

[70] S. Parmar. Autoencoders - denoising understanding! Medium. Accessed: 2021-12-17. [Online]. Available: https://medium.com/analytics-vidhya/autoencoders-denoising-understanding-b41315fd7fa

[71] D. Eberly, "Least squares fitting of data by linear or quadratic structures," 2018.

[72] P. Arnecke and J. Seewig, "Characterisation of texture orientation on ground surfaces with high accuracy," Journal of Physics: Conference Series. 15th International Conference on Metrology and Properties of Engineering Surfaces (Met & Props 2015), mar 2015.

[73] S. R. Deans, "The radon transform and some of its applications," 1983.

[74] B. Jähne, Digitale Bildverarbeitung, 7th ed. Berlin: Springer Vieweg, 2012.

[75] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, Discrete-Time Signal Processing (2nd Ed.). USA: Prentice-Hall, Inc., 1999.

[76] J. Beyerer, Analyse von Riefentexturen. VDI-Verl. Dusseldorf, 1994.

[77] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," The Computer Journal, vol. 7, no. 2, pp. 155–162, 01 1964, accessed: 2021-12-17. [Online]. Available: https://doi.org/10.1093/comjnl/7.2.155

[78] xingjiepan, "Cylinder fitting," https://github.com/xingjiepan/cylinder_fitting, 2017, accessed: 2021-12-17.

[79] A. Karatas, “Generation of cylinders,” https://github.com/AbdullahKaratas/GenerationOfCylinders, 2020, accessed: 2021-12-17.

[80] S. Boyd, “Constrained least squares,” https://stanford.edu/class/ee103/lectures/constrained-least-squares_slides.pdf, 2017.

[81] F. N. Fritsch and R. E. Carlson, “Monotone piecewise cubic interpolation,” SIAM Journal on Numerical Analysis, vol. 17, no. 2, pp. 238–246, 1980.

[82] A. Karatas, “Texture direction recognition,” https://github.com/AbdullahKaratas/Texture-Direction-Recognition, 2020, accessed: 2021-12-17.

[83] A. Karatas, “Generate Cutting Edges And CNN Reconstruction,” 2020, accessed: 2021-12-17. [Online]. Available: https://github.com/AbdullahKaratas/GenerateCuttingEdgeAndCNNReconstruction

[84] J. W. Tukey, Exploratory Data Analysis. Addison-Wesley, 1977.

[85] I. Pitas and A. Venetsanopoulos, Nonlinear Digital Filters, ser. Kluwer international series in engineering and computer science: VLSI, computer architecture, and digital signal processing. Springer US, 1990.

[86] B. Jähne, Digital Image Processing 6th Edition. Berlin [u.a.]: Springer, 2005.

[87] VDI VDE 2655 Blatt 1.1:2008-03, Optische Messtechnik an Mikrotopographien - Kalibrierung von Interferenzmikroskopen und Tiefeneinstellnormalen für die Rauheitsmessung. Verein Deutscher Ingenieure.

[88] Deutsches Institut für Normung e.V.: DIN EN ISO 28178-3:2012-11, Geometrische Produktspezifikation (GPS) - Oberflächenbeschaffenheit: Flächenhaft - Teil 3: Spezifikationsoperatoren (ISO 25178-3:2012); Deutsche Fassung EN ISO 25178-3:2012. Beuth Verlag, Berlin.

[89] Deutsches Institut für Normung e.V.: DIN EN ISO 16610-60:2013-01, Geometrische Produktspezifikation (GPS) – Filterung – Teil 60: Lineare Flächenfilter: Grundlegende Konzepte (ISO/DIS 16610-60:2012); Deutsche Fassung prEN ISO 16610-60:2012. Beuth Verlag, Berlin.

[90] Deutsches Institut für Normung e.V.: DIN EN ISO 16610-61:2013-01, Geometrische Produktspezifikation (GPS) - Filterung - Teil 61: Lineare Flächenfilter - Gauß-Filter (ISO/DIS 16610-61:2012); Deutsche Fassung prEN ISO 16610-61:2012. Beuth Verlag, Berlin.

[91] H.-W. Lemke, J. Seewig, H. Bodschwinna, and S. Brinkmann, "Kenngrößen der abbott-kurve zur integralen beurteilungdreidimensional gemessener zylinderlaufbahn-oberflächen," MTZ - Motortechnische Zeitschrift 64.5, 2003.

[92] Tying weights. Stack Overflow. Accessed: 2021-12-17. [Online]. Available: https://stackoverflow.com/questions/57827274/keras-autoencoder-tying-weights-from-encoder-to-decoder-not-working

[93] C. Ranjan. Build the right autoencoder — tune and optimize using pca principles. part ii. towards datascience. Accessed: 2021-12-17. [Online]. Available: https://towardsdatascience.com/build-the-right-autoencoder-tune-and-optimize-using-pca-principles-part-ii-24b9cca69bd6

[94] "Groups, Fields and Polynomials," http://www.cs.cmu.edu/~aada/courses/15251s17/www/notes/groups-fields-polynomials-notes.pdf, accessed: 2019-07-18.

[95] I. N. Herstein, Topics in Algebra. Wiley, 1975.

[96] "Convex function," http://www.princeton.edu/~amirali/Public/Teaching/ORF523/S16/ORF523_S16_Lec7_gh.pdf, accessed: 2019-07-18.

[97] J. Hefferon, Linear Algebra, ser. VCU mathematics textbook series. Department of Mathematics & Applied Mathematics/Virginia Commonwealth University, 2009.

[98] S. J. Axler, Linear Algebra Done Right, ser. Undergraduate Texts in Mathematics. Springer, 1997.

[99] G. Strang. Thomson, Brooks/Cole.

[100] G. Fischer, Lehrbuch der Algebra: mit lebendigen Beispielen, ausführlichen Erläuterungen und zahlreichen Bildern ; [Bachelor geeignet!], ser. Vieweg Mathematik. Vieweg, 2008.

[101] B. Y. R. Penrose and C. J. A. Todd, “[a generalized inverse for matrices,” 1954.

[102] C. J. B. Yann LeCun, Corinna Cortes. The mnist dataset. Courant Institute, Google Labs and Microsoft Research. Accessed: 2021-12-17. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[103] A. Karatas, “Reconstruction using pca,” https://github.com/AbdullahKaratas/ReconstructionPCA, 2020, accessed: 2021-12-17.

[104] Autoencoder with pca. Stack Exchange Network. Accessed: 2021-12-17. [Online]. Available: https://stats.stackexchange.com/questions/190148/building-an-autoencoder-in-tensorflow-to-surpass-pca

[105] A. Gut, Probability: A Graduate Course, ser. Springer Texts in Statistics. Springer New York, 2006.

[106] “Probability Space,” https://de.wikipedia.org/wiki/Wahrscheinlichkeitsraum, accessed: 2019-07-19.

[107] K. Hinderer, Grundbegriffe der Wahrscheinlichkeitstheorie, ser. Hochschultext (Berlin). Springer, 1972.

[108] D. Meintrup and S. Schäffler, Stochastik: Theorie und Anwendungen, ser. Statistik und ihre Anwendungen. Springer Berlin Heidelberg, 2006.

[109] U. Kumar, J. Crocker, T. Chitra, and H. Saranga, Reliability and Six Sigma. Springer US, 2010.

[110] T. M. Mitchell, Machine Learning. New York: McGraw-Hill, 1997.

[111] “Recall and Precision,” https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c, accessed: 2019-05-28.

[112] “L_1 and L_2 Regularization Method,” https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c, accessed: 2019-07-20.

[113] A. Müller, S. Guido, and K. Rother, Einführung in Machine Learning mit Python: Praxiswissen Data Science, ser. Animals. O’Reilly, 2017.

[114] P. Dangeti, Statistics for Machine Learning. Packt Publishing, 2017.

[115] "Scikit-learn Documentation," https://scikit-learn.org/stable/documentation.html, accessed: 2019-07-20.

[116] 3Blue1Brown. But what is a neural network? Youtube. Accessed: 2021-12-17. [Online]. Available: https://www.youtube.com/watch?v=aircAruvnKk

[117] H. Montazeri and S. Zandavi, Derivative of Heaviside Step Function Vs. Delta Function in Continuum Surface Force (CSF) Models. Elsevier Science Limited, 2018.

[118] K. M. Fauske. Example: Neural network. texample. Accessed: 2021-12-17. [Online]. Available: https://texample.net/tikz/examples/neural-network/

[119] J. Patterson and A. Gibson, Deep Learning: A Practitioner's Approach. O'Reilly, 2017, accessed: 2021-12-17. [Online]. Available: https://www.safaribooksonline.com/library/view/deep-learning/9781491924570/

[120] K. Wang, Intelligent Condition Monitoring and Diagnosis Systems: A Computational Intelligence Approach, ser. Frontiers in artificial intelligence and applications. IOS Press, 2003.

[121] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds. Cambridge, MA, USA: MIT Press, 1986, ch. Learning Internal Representations by Error Propagation, pp. 318–362, accessed: 2021-12-17. [Online]. Available: http://dl.acm.org/citation.cfm?id=104279.104293

[122] N. Pentreath, Machine Learning with Spark, ser. Community experience distilled. Packt Publishing, 2015.

[123] C. Arankalle, G. Dwyer, B. Geerdink, K. Gera, K. Liao, and A. S, The Artificial Intelligence Infrastructure Workshop: Build your own highly scalable and robust data storage systems that can support a variety of cutting-edge AI applications. Packt Publishing, 2020.

[124] tensorflow, “Tensorflow back-propagation,” https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/eager/backprop.py, 2020, accessed: 2021-12-17.

[125] M. Kerckhove, Scale-Space and Morphology in Computer Vision: Third International Conference, Scale-Space 2001, Vancouver, Canada, July 7-8, 2001. Proceedings, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003.

[126] M. Finne, Methods for Direction-Finding of Direct-Sequence Spread-Spectrum Signals, ser. FOA reports. Diane Publishing Company, 1996.

[127] V. Dumoulin and F. Visin, “A guide to convolution arithmetic for deep learning,” 2016.

[128] J. Ren, A. Hussain, J. Zheng, C. Liu, B. Luo, H. Zhao, and X. Zhao, Advances in Brain Inspired Cognitive Systems: 9th International Conference, BICS 2018, Xi’an, China, July 7-8, 2018, Proceedings, ser. Lecture Notes in Computer Science. Springer International Publishing, 2018.

[129] S. Jansen, Hands-On Machine Learning for Algorithmic Trading: Design and implement investment strategies based on smart algorithms that learn from data using Python. Packt Publishing, 2018.

[130] G. Bonaccorso, Mastering Machine Learning Algorithms: Expert techniques to implement popular machine learning algorithms and fine-tune your models. Packt Publishing, 2018.

[131] N. Purkait, Hands-On Neural Networks with Keras, 1st ed. Packt Publishing, 2019.

[132] “Convolutional Neural Networks for Visual Recognition,” https://cs231n.github.io/convolutional-networks/, accessed: 2020-03-25.

[133] F.-F. Li, J. Johnson, and S. Yeung, “Convolutional neural networks for visual recognition,” http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf, 2017, accessed: 2021-12-17.

[134] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive autoencoders: Explicit invariance during feature extraction," in Proceedings of the 28th International Conference on International Conference on Machine Learning, ser. ICML'11. USA: Omnipress, 2011, pp. 833–840, accessed: 2021-12-17. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104482.3104587

[135] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional autoencoders for hierarchical feature extraction," in Artificial Neural Networks and Machine Learning – ICANN 2011, T. Honkela, W. Duch, M. Girolami, and S. Kaski, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 52–59.

[136] G. Alain, Y. Bengio, L. Yao, J. Yosinski, E. Thibodeau-Laufer, S. Zhang, and P. Vincent, "Gsns : Generative stochastic networks," CoRR, vol. abs/1503.05571, 2015, accessed: 2021-12-17. [Online]. Available: http://arxiv.org/abs/1503.05571

[137] A. Makhzani and B. Frey, "Winner-take-all autoencoders," in Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 2791–2799, accessed: 2021-12-17. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969442.2969552

[138] A. Makhzani, J. Shlens, N. Jaitly, and I. J. Goodfellow, "Adversarial autoencoders," CoRR, vol. abs/1511.05644, 2015, accessed: 2021-12-17. [Online]. Available: http://arxiv.org/abs/1511.05644

Abdullah Karatas

Place of Birth: Frankenthal (Pfalz)

EDUCATION

Research assistants at the Chair of Metrology and Sensor Technology July 2015 – June 2019

PhD Thesis: Comparison of model-based methods with strategies of machine learning for defect reconstruction, classification and regression in the field of measurement technology

TU Kaiserslautern

Berichte aus dem Lehrstuhl für Messtechnik und Sensorik

bereits veröffentlicht wurden

1	Wendel, M.: Qualifizierung eines Streulichtsensors und Untersuchung erster Ansätze zur dreidimensionalen Streulichterfassung 2015, ISBN 978-3-95974-006-7	**€ 39,–**
2	Schäfer, P.: Modellbasierte Entwicklung pneumatischer Abstandssensoren für prozessintegrierte Messungen 2015, ISBN 978-3-95974-009-8	**€ 39,–**
3	Eifler, M.: Modellbasierte Entwicklung von Kalibriernormalen zur geometrischen Produktspezifikation 2016, ISBN 978-3-95974-027-2	**€ 39,–**
4	Arnecke, P.: A measurement method for characterising micro lead on ground shaft surfaces 2017, ISBN 978-3-95974-047-0	**€ 39,–**
5	Kusnezowa, T.: Möglichkeiten und Herausforderungen bei der taktilen 2D Rauheitsmessung auf Konturen 2017, ISBN 978-3-95974-050-0	**€ 39,–**
6	Torner, F. M.: Entwicklung virtueller, optischer Sensoren zur Charakterisierung geometrischer Oberflächen 2018, ISBN 978-3-95974-085-2	**€ 39,–**
7	Alapurath George, B.: Web-Based Reference Software For Characterisation Of Surface Roughness 2018, ISBN 978-3-95974-095-1	**€ 39,–**
8	Wiehr, C.: Anwenderunterstützung bei der Nutzung und Überprüfung von optischen 3D-Oberflächenmessgeräten 2019, ISBN 978-3-95974-110-1	**€ 39,–**
9	Eifler, M.: Tendenzen und Herausforderungen in der geometrischen Produktspezifikation am Beispiel der Rauheitsmesstechnik 2019, ISBN 978-3-95974-117-0	**€ 39,–**
10	Ströer, F.: Modellbasierte Entwicklung der Betriebselektronik für eine Rasterkraftsonde im Frequenzmodulationsverfahren zum Messen technischer Oberflächen 2020, ISBN 978-3-95974-129-3	**€ 39,–**
11	Mansel, H.: Pneumatische Inline-Messung formgehonter Zylinderlaufflächen 2020, ISBN 978-3-95974-141-5	**€ 39,–**

Berichte aus dem Lehrstuhl für Messtechnik und Sensorik

bereits veröffentlicht wurden

12 Sivasothy, P.: Potentialanalyse eines nichtinvasiven Sensorkonzepts zur Füllstandüberwachung bei mobilen Schüttgutsilos
2021, ISBN 978-3-95974-160-6 **€ 39,–**

13 Teto, JO D.: Isolation of a measurement platform for UAV applications
2022, ISBN 978-3-95974-175-0 **€ 39,–**

14 Karatas, A.: Comparison of model-based methods with machine learning strategies for defect reconstruction, classification, and regression in the field of measurement technology
2022, ISBN 978-3-95974-183-5 **€ 39,–**

ISBN: 978-3-95974-183-5
ISSN: 2365-9742

Kurzfassung

Automatisierung, Industrie 4.0 und künstliche Intelligenz spielen für Unternehmen eine immer zentralere Rolle. Insbesondere künstliche Intelligenz ermöglicht derzeit neue Methoden, um einen höheren Automatisierungsgrad zu erreichen. Allerdings sind Methoden des maschinellen Lernens in der Regel dann besonders lukrativ, wenn viele Daten einfach gesammelt werden können und mit Hilfe dieser Daten Muster gelernt werden können. Im Bereich der Messtechnik kann sich dies je nach Arbeitsgebiet als schwierig erweisen. Insbesondere bei Messungen im Mikrometermaßstab sind Messdaten oft mit viel Zeit, Mühe, Geduld und Geld verbunden, so dass Messdaten nicht ohne weiteres verfügbar sind. Dies wirft die Frage auf, wie aussagekräftig Ansätze des maschinellen Lernens auf verschiedene Bereiche von Messaufgaben angewendet werden können, insbesondere im Vergleich zu aktuellen Lösungsansätzen, die modellbasierte Methoden verwenden. Dieser Beitrag befasst sich mit dieser Frage, indem zwei Forschungsbereiche in der Messtechnik, die Bestimmung von Mikrodrall und die Rekonstruktion, näher beleuchtet werden. Es werden Methoden zur Mikrodrallbestimmung vorgestellt, die Textur und Werkzeugachse mit hoher Genauigkeit bestimmen. Die verwendeten Methoden basieren auf Signalverarbeitung, klassischer Optimierung und maschinellem Lernen. Im zweiten Forschungsbereich werden Rekonstruktionen für Schneidkanten im Detail betrachtet. Die Rekonstruktionsmethoden basieren hier auf dem robusten Gauß-Filter und tiefen neuronalen Netzen, genauer gesagt auf Autoencodern. Alle Ergebnisse zu Mikrodrall und Rekonstruktion werden in dieser Arbeit verglichen und gegenübergestellt und die Anwendbarkeit der verschiedenen Ansätze bewertet.

Abstract

Automation, Industry 4.0 and artificial intelligence are playing an increasingly central role for companies. Artificial intelligence in particular is currently enabling new methods to achieve a higher level of automation. However, machine learning methods are usually particularly lucrative when a lot of data can be easily collected and patterns can be learned with the help of this data. In the field of metrology, this can prove difficult depending on the area of work. Particularly for micrometer-scale measurements, measurement data often involves a lot of time, effort, patience, and money, so measurement data is not readily available. This raises the question of how meaningfully machine learning approaches can be applied to different domains of measurement tasks, especially in comparison to current solution approaches that use model-based methods. This thesis addresses this question by taking a closer look at two research areas in metrology, micro lead determination and reconstruction. Methods for micro lead determination are presented that determine texture and tool axis with high accuracy. The methods are based on signal processing, classical optimization and machine learning. In the second research area, reconstructions for cutting edges are considered in detail. The reconstruction methods here are based on the robust Gaussian filter and deep neural networks, more specifically autoencoders. All results on micro lead and reconstruction are compared and contrasted in this thesis, and the applicability of the different approaches is evaluated.