Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)

Institut für Test- und Simulation für Gasturbinen

Bachelor-Thesis

Untersuchung zur Bestimmung und Vorhersage der Borstenanordnung in einer Bürstendichtung

Fakultät für Maschinenbau, Fahrzeug- und Flugzeugtechnik
Fachgebiet für Künstliche Intelligenz im Maschinenbau
Hochschule München

von

cand. B.Sc.
Maksym Meyer
Matrikelnummer: 21227719
Wintersemester 2023/2024

Betreuer:

Prof. Dr.-Ing. Marcin Hinz
Moritz Ursprung M.Sc. (DLR)

Tag der Anmeldung: 14. Juni 2023
Tag der Abgabe: 13. Dezember 2023

Hochschule München University of Applied Sciences

Erklärung

gemäß § 26 Abs. 7 ASPO

Name: Meyer

Vorname: Maksym

Studiengang: Luft- und Raumfahrttechnik

Matrikel-Nr.: 21227719

Betreuer/in: Prof. Dr.-Ing. Marcin Hinz

Hiermit erkläre ich, dass ich die Bachelorarbeit selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt, sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

München, 13.12.2023 Meyer

Ort, Datum Unterschrift

Danksagung

Ich möchte meinen Betreuern, Herrn Professor Hinz, für den regelmäßigen und fachlich tiefgehenden Austausch und Moritz Ursprung für die Ermöglichung und wertvolle Hilfe bei der Erstellung der Arbeit danken.

Ebenso bedanke ich mich bei meinen Eltern für die kontinuierliche Unterstützung während des Studiums.

Abstract

Bürstendichtungen bestehen aus einer Vielzahl von Borsten, die unregelmäßig und in Abhängigkeit von der anliegenden Druckdifferenz verschieden geordnet sein können. Dabei hat die Anordnung einen Einfluss auf die Strömungsvorgänge in der Bürstendichtung. Im Rahmen dieser Arbeit werden Methoden der klassischen Computer Vision verwendet, um in Mikroskop-Aufnahmen einzelne Borsten zu erkennen und das resultierende Gefüge statistisch zu erfassen. Durch die bildbasierte Erkennung sollen Merkmale einer realen Borstenanordnung Einfluss gewinnen, die bei bisherigen Modellierungsansätzen nicht berücksichtigt wurden. Dies bildet die Grundlage für eine Simulation von Borstenanordnungen. Die daraus resultierenden Geometrien können zum Aufbau eines Bürstendichtungsmodells zur CFD-Analyse verwendet werden, um anordnungsabhängige Strömungseigenschaften zu untersuchen.

Brush seals consist of a large number of bristles, which can be arranged irregularly and in different ways depending on the pressure difference applied. The arrangement has an influence on the flow processes in the brush seal. In this work, methods of classical computer vision are used to recognize individual bristles in microscope images and to statistically capture the resulting structure. The image-based recognition is intended to consider features of a real bristle arrangement that have not been taken into account in previous modeling approaches. This forms the basis for a simulation of bristle arrangements. The resulting geometries can be used to set up a brush seal model for CFD analysis in order to investigate arrangement-dependent flow properties.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungs- und Symbolverzeichnis

BP-0	Borstenpaket mit 0 Grad Inklinationswinkel
BP-5	Borstenpaket mit 5 Grad Inklinationswinkel
BP-8	Borstenpaket mit 8 Grad Inklinationswinkel
CAD	Computer-Aided Design
CFD	Computational Fluid Dynamics
CLAHE	Contrast Limited Adaptive Histogram Equalization
FP	False Positive
FN	False Negative
MLE	Maximum Likelihood Estimation
ROI	Region of Interest
SFC	Spezifischer Kraftstoffverbrauch
TP	True Positive
TN	True Negative

B	Intensität des blauen Farbkanals
d	Distanzfunktion
D	Borstendurchmesser, Distanztransformation
$\hat{D}$	Kolmogorov-Smirnov-Teststatistik
E	Kantenstärke
$f_{eq}, f_{s,t}$	Transformationsfunktionen
f_{00}, f_{11}	Transformationsfunktionen zur bilinearen Interpolation
F	Verteilungsfunktion
$\hat{F}$	Levene-Teststatistik
F^{-1}	Inverse einer Verteilungsfunktion
F_0	Theoretische Verteilungsfunktion
G	Intensität des grünen Farbkanals
G_x, G_y	Sobel-Filter
h	Histogramm, Kernel
H	Kumulatives Histogramm, Strukturelement
H_x, H_y	Einfache lineare Filter
H_0	Nullhypothese
i	Intensität
I	Bild
I_B	Binärbild

I_x, I_y	Partielle Ableitung eines Pixels
k	Schwellenwert
L	Likelihood-Funktion
m, n	Stichprobenumfang
M	Dimension eines Bildes in x-Richtung
n_i	Intensität eines Pixels
N	Dimension eines Bildes in y-Richtung
p_i	Wahrscheinlichkeit eines Intensitätswerts
Δp	Druckdifferenz in bar
P	Wahrscheinlichkeit
P_i	Wahrscheinlichkeitsdichte
r	Abstand des Ursprungs zu einer Geraden
R	Intensität des roten Farbkanals, Rotationsmatrix, Rangsumme
u, v	Koordinaten eines Pixels
U	U-Wert
w	Gewicht
x_i	Realisierung einer Zufallsvariable
x, y	Koordinaten eines Pixels
X_i	Zufallsvariable
Y	Luminanz

α	Signifikanzniveau
λ	Legewinkel in deg
μ	Mittelwert
φ	Inklinationswinkel in deg
ρ	Packungsdichte in 1/mm
σ	Standardabweichung, Varianz
σ_B	Varianz zwischen Klassen
σ_W	Varianz in einer Klasse
θ	Normalenwinkel, Rotationswinkel in deg
ω	Klassenwahrscheinlichkeit

∇	Gradient
$\mathbb{R}$	Menge der reellen Zahlen
$\oplus$	Dilationsoperator
$*$	Faltungsoperator

1. Einleitung

Der Luftverkehr hat in den letzten Jahrzenten ein starkes Wachstum erlebt. Dabei ist der Betrieb von Luftfahrzeugen mit dem Ausstoß klimawirksamer Emissionen wie Kohlenstoffdioxid, Stickoxiden oder Wasserdampf verbunden. So wurde der Beitrag der globalen Luftfahrt auf den anthropogenen Klimawandel innerhalb eines Jahres bereits mit 3,5% berechnet. (Lee et al., 2021)
Damit wird die Luftfahrtindustrie zunehmend mit der Forderung einer Nachhaltigkeitssteigerung konfrontiert. Klimaschutzabkommen und das steigende Bewusstsein in Politik und Gesellschaft gestalten die Optimierung von Luftfahrzeugen als unausweichlich.
2022 einigte sich die International Civil Aviation Organization (ICAO) der vereinten Nationen unter Beteiligung von 184 Staaten auf das kollektive Ziel bis 2050 klimaneutral zu sein (ICAO, 2022). Dies offenbart jedoch auch die Frage, auf welche Weise diese Ziele erreicht werden können. Potentiale zur Senkung des Kraftstoffverbrauchs und der auftrenden Emissionen liegen so unter anderem in der Weiterentwicklung des übergeordneten Antriebskonzepts, aber auch in der Optimierung von Komponenten, sowohl von bereits im Betrieb befindlichen als auch zukünftigen Flugzeugantrieben. Ein Bauteil, welches zur Erfüllung dieser steigenden Anforderungen dienen kann, sind Bürstendichtungen, die beispielsweise im Bereich des Verdichters, der Turbine oder Ölsümpfen verwendet werden können (Aksit, 2012). Die Leckage kann hierbei auf bis zu 10-20% des Wertes bisher verwendeter Labyrinthdichtungen gesenkt werden. Dadurch sind Reduktionen des spezifischen Kraftstoffverbauchs (SFC) von bis zu 2,5% möglich (Bruce M. Steinetz, 1996). Dabei werden höhere Druckunterschiede, Temperaturen und Umfangsgeschwindigkeiten in der Umgebung der Dichtung gegenüber bisherigen Lösungen erlaubt (Chupp, 1991).
Neben Einsätzen in der militärischen Luftfahrt sind Bürstendichtungen mittlerweile in der Serienproduktion von Triebwerken der zivilen Luftfahrt weit verbreitet. Anwendungen in anderen Industriezweigen, wie beispielsweise stationären Gasturbinen, sind ebenfalls möglich (Fuchs et al., 2018).

2. Stand der Technik

Bürstendichtungen sind Dichtungen, die in der Umgebung rotierender Bauteile verwendet werden.
Die Dichtwirkung wird mithilfe einer großen Zahl einzelner Borsten gebildet. Das daraus resultierende Borstenpaket stellt ein flexibles Gefüge dar, welches eine Bewegung des Rotors, etwa durch Wärmeausdehnung tolerieren kann, ohne beschädigt zu werden. (Fuchs et al., 2018)
Damit ist auch eine reduzierte Einstellung des Dichtspalts möglich, was zu einer Verringerung der Leckage führt. Die Flexibilität des Borstenpakets sorgt auch dafür, dass der Dichtspalt über einen längeren Zeitraum klein gehalten werden kann und die Verlustleistung folglich gegenüber Labyrinthdichtungen reduziert wird (Chupp, 1991). Abbildung 2.1 zeigt eine Bürstendichtung aus axialer Sicht.

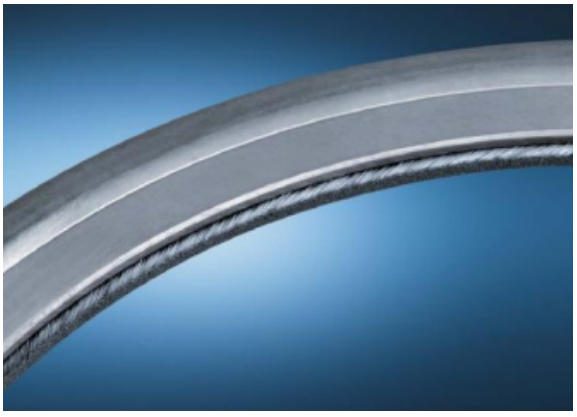


Abbildung 2.1: Axialer Blick auf eine Bürstendichtung (Fuchs et al., 2018)

Bürstendichtungen können dabei, je nach Anforderungsprofil, unterschiedlich gefertigt werden. Die nachfolgenden Abschnitte liefern einen Überblick über den Aufbau üblicher Ausführungen, Geometrieparameter, deren Einflüsse und auftretende Strömungseffekte. Ebenso wird der Stand der Technik im Bezug auf die Modellierung von Bürstendichtungen aufgezeigt.

2.1 Aufbau der Bürstendichtung

Die einfachste Ausführung der Bürstendichtung ist in Abbildung 2.2 dargestellt. Hierbei wird das Borstenpaket zwischen einen Halte- und Stützring gedrückt und anschließend verschweißt. Der Schweißprozess ist anspruchsvoll und setzt voraus, dass

geeignete Werkstoffe verwendet werden. Im Bereich der Wärmeeinflusszone kann es zu Versprödung und Hinterschneidungen kommen, wodurch sich einzelne Drähte im Laufe der Zeit lösen können. (Gail & Beichl, 2000)

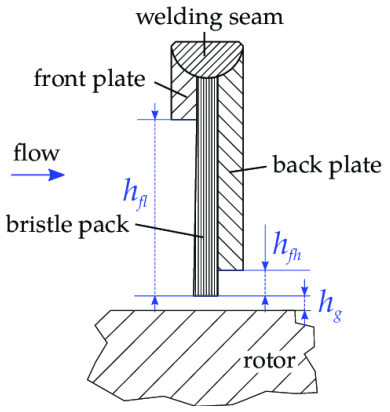


Abbildung 2.2: Bürstendichtung in geschweißter Ausführung (Hildebrandt et al., 2018): Die bemaßten Geometrien sind Einflussgrößen auf das Betriebsverhalten.

Ein Fortschritt konnte durch die geklemmte Ausführung der Bürstendichtung erreicht werden (Gail & Beichl, 2000). Abbildung 2.3 zeigt diese Variante der Bürstendichtung.

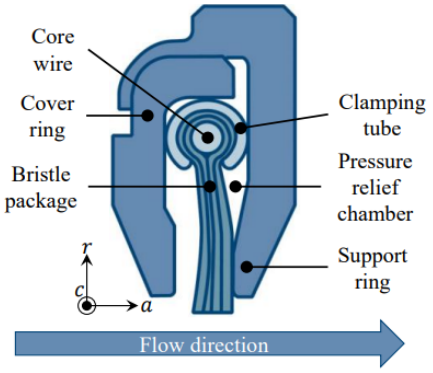


Abbildung 2.3: Bürstendichtung in der geklemmten Ausführung (Fuchs et al., 2018)

Die Gehäuseteile, d.h. der Deck- bzw. Stützring, werden über einen Dreh- oder Tiefziehprozess gefertigt (Gail & Beichl, 2000).
Der Deckring erlaubt dabei einen Schutz vor Störungen in der Anströmung. Der Stützring dient der Verhinderung des Biegens der einzelnen Drähte in Folge der sich einstellenden Druckdifferenz. (Fuchs & Haidn, 2017)
Die Borsten werden hierbei zunächst um einen Kerndraht gewickelt und durch den Klemmring fixiert. Der Klemmring wird zwischen Deck- und Stützring positioniert und die entstandene Einheit miteinander verpresst, sodass eine axiale oder radiale Bewegung des Kerndrahts ausgeschlossen ist. (Gail & Beichl, 2000)
Die Anwinkelung des Stützrings dient einer Steifigkeitserhöhung, welche eine Reduktion der Wandstärke ermöglicht. Die geklemmte Ausführung der Bürstendichtung

kann auch ohne einen Schweißprozess gefertigt werden, wodurch dünnere Bauteile und damit Gewichtseinsparungen möglich sind. Ferner wird neben metallischen Werkstoffen der Einsatz von Borsten aus Keramik oder Kunststoff ermöglicht. (Gail & Beichl, 2000)

2.2 Geometrische Einflussgrößen

Um das gewünschte Betriebsverhalten herbeizuführen, lassen sich an der Bürstendichtung Geometrieparameter gezielt einstellen. Abbildung 2.4 zeigt exemplarisch den Legewinkel λ im Axial- und den Inklinationswinkel φ im Umfangsschnitt der Bürstendichtung.

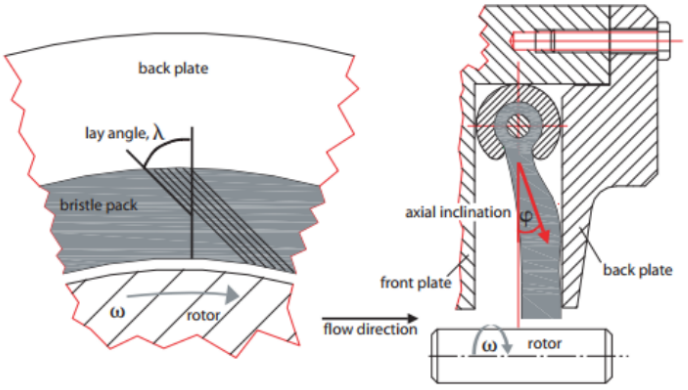


Abbildung 2.4: Winkel der Bürstendichtung im Axial- und Umfangsschnitt (Schur & Friedrichs, 2019)

Der Legewinkel λ in Rotationsrichtung ermöglicht eine Reduktion des Verschleißes sowie der Reibung. Bei einem Anstreifen des Rotors können sich die Borsten elastisch biegen, statt zu knicken (Chupp, 1991). Dadurch können auch größere Interferenzen des Rotors mit dem Borstenpaket toleriert werden (Aksit, 2012). Dies trägt zu einem Erhalt der Dichtwirkung im Laufe des Betriebs bei (Chupp, 1991).
Über den Inklinationswinkel φ kann die Vorspannung des Borstenpakets und Reibung zwischen den Borsten eingestellt werden. So neigen wenig inklinierte Bürstendichtungen bei niedrigen Drücken zu Oszillationen und somit zu Verschleiß. Hier weisen die höher inklinierten, steiferen Bürstendichtugen ein stabileres Verhalten auf. (Schwarz et al., 2014)
Der Durchmesser der Borsten beeinflusst die Wärmeentstehung und das Verformungsverhalten der Borsten. Dünne Borsten neigen zu höheren Verformungen und Instabilitäten bei hohen Drücken, wohingegen dicke Borsten verformungsresistenter sind, aber zu erhöhter Reibung und Verschleiß neigen. (Zheng et al., 2013)
Eine erhöhte Packungsdichte ρ der Borsten, d.h. eine Erhöhung der Zahl der Borsten pro Milimeter in Umfangsrichtung, kann die auftretende Leckage reduzieren. Ein

dichteres Borstenpaket führt aber auch zu erhöhter Reibung zwischen den Borsten und somit zu einer erhöhten Steifigkeitswirkung, also einer Resistenz gegen hohe Drücke. (Crudgington, 1998)
Im Falle des Anstreifens durch den Rotor ist bei einem steiferen Borstenpaket (z.B. durch Inklination oder hohe Packungsdichte) mit erhöhtem Verschleiß der Dichtung zu rechnen (Aksit, 2012).

Bürstendichtungen unterliegen zusammengefasst vielen geometriebedingten Einflüssen auf ihr Betriebsverhalten, was eine gezielte Auslegung für den jeweiligen Anwendungsfall und ein Verständnis von Ursache und Wirkung erfordert.

2.3 Blow-Down und Hang-Up

Der Blow-Down-Effekt beschreibt, dass sich infolge eines radialen Druckgradients und der zugehörigen Kraft, der Legewinkel λ verringert. Dadurch stellt sich eine Minderung des Dichtspalts zwischen Borstenpaket und Rotor ein und die Dichtwirkung kann gesteigert werden (Dogu, 2005). Abbildung 2.5 zeigt schematisch den Strömungsverlauf in einer Bürstendichtung. So erfährt die axiale Anströmung im Bereich des Stützrings eine radiale Komponente, ursächlich für den Blow-Down-Effekt, bevor sie mit Annäherung an den Dichtspalt wieder axial wird (Aksit, 2012).
Der Deckring der geklemmten Bürstendichtung, beschrieben in Kapitel 2.1, kann diesem Effekt entgegenwirken (Gail & Beichl, 2000).

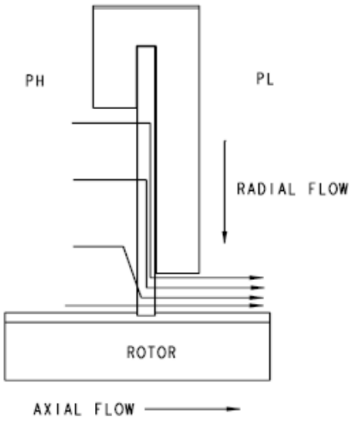
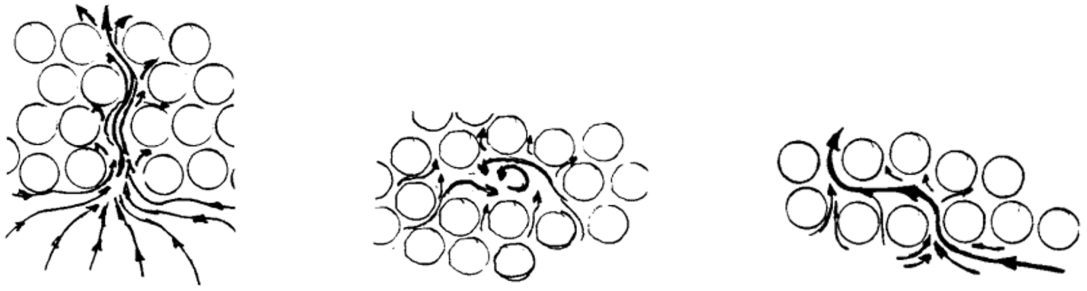


Abbildung 2.5: Blow-Down-Effekt einer Bürstendichtung (Aksit, 2012 nach Aksit, 1998): Die radiale Strömungskomponente kann den Dichtspalt reduzieren, indem Borsten nach unten ausgelenkt werden.

Der Hang-Up-Effekt bildet das Gegenstück zum Blow-Down-Effekt.
Nachdem es zu einem Anstreifen des Rotors kommt, überwiegen die Reibungskräfte des Borstenpakets gegenüber den Rückstellkräften. Es verbleibt ein Dichtspalt, der größer ist, als der vor dem Anstreifen. (Lelli et al., 2005)

2.4 Radiale Strömungsmuster

Bei radialem Blick auf das Borstenpaket können typische Strömungsmuster beobachtet werden. Beispiele hierfür sind in Abbildung 2.6 zu sehen.



(a) Rivering (b) Vortical Flow (c) Lateral-Flow

Abbildung 2.6: Beispiele für typische radiale Strömungsmuster (Braun et al., 1990)

Die einzelnen Muster werden durch eine Änderung der Anordnung des Borstenpakets hervorgerufen und zeigen jeweils einen Einfluss auf die Leckage und Stabilität der Bürstendichtung. Einheitliche Strömungen werden so durch unregelmäßige Lücken im Gefüge in nichteinheitliche Strömungen gewandelt. Ebenso können unterschiedliche Packungsdichten, Druckverluste oder die Steifigkeit der Borsten für die auftretenden Phänomene ursächlich sein. (Braun et al., 1990)
Rivering beschreibt hierbei einen Zustand, bei welchem sich in Folge eines unregelmäßigen Borstenpakets und des Kontakts einzelner Borsten, die Strömung teilweise vollständig versperrt wird. In anderen Bereichen bildet sich hingegen der namensgebende Strömungskanal, der nun einen wesentlich höheren Massestrom erfährt. (Fuchs & Haidn, 2017)
Ebenso ist die Bildung von zufälligen, chaotischen Verwirbelungen als *Vortical Flow* möglich (Braun et al., 1990). *Lateral Flow* äußert sich als Strömung, die senkrecht auf der axialen Anströmung liegt und für eine zusätzliche Dichtwirkung sorgt.

2.5 Modellierung von Bürstendichtungen

Um vorhandene Mechanismen und deren Auswirkungen, wie jene der Geometrieparameter in Kapitel 2.2 oder die Strömungseffekte aus Kapitel 2.3 und 2.4 vorherzusagen, ist neben experimentellen Ergebnissen mit geeigneten Prüfständen eine rechnergestützte Modellierung sinnvoll.
Durch die Vielzahl einzelner Borsten, deren Verformung sowie Interaktion mit der Strömung, gestaltet sich dies jedoch als sehr herausfordernd (Fuchs & Haidn, 2017). Die Literatur umfasst mehrere Ansätze zur Modellierung der Strömung im Borstenpaket, welche sich jeweils in der Geometrie, den verwendeten Gleichungen und den berücksichtigten Effekten unterscheiden. Ein häufig verwendeter Ansatz ist der eines globalen porösen Mediums, wodurch die zugehörigen Berechnungen vereinfacht

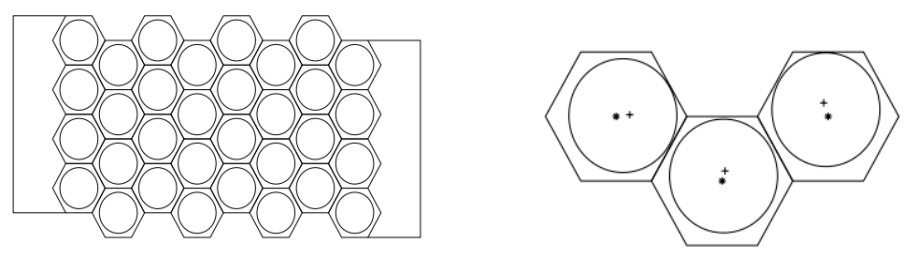
werden. Dieser Ansatz erfordert meist eine Kalibrierung der Gleichungsparameter mit Hilfe von empirischen Daten, beispielsweise anhand von Leckage-, Druck-, oder Kraftmessungen. (Pugachev, 2013)
Ein anderer Ansatz ist die gelöste Modellierung der einzelnen Borsten. Dreidimensionale CFD-Modelle dieser Art erlauben theoretisch eine Abbildung aller physikalischen Effekte. In der Regel wird hier wegen des hohen Rechenaufwands nur ein Ausschnitt der Bürstendichtung betrachtet. Besonders geeignet ist diese Art der Modellierung für die Abbildung der Interaktion zwischen Fluid und der Strukturverformung. (Neef et al., 2007)
Die kontinuierlich steigenden Rechenkapazitäten der letzten Jahre stellen hierbei ein großes Potential dar.

Fuchs und Haidn (2017) haben für diesen Modellierungsansatz bereits den Einfluss einer Abweichung von einer perfekten radialen Anordnungsgeometrie untersucht. Als Ausgangslage wurde der dichtest gepackte Zustand der Bürstendichtung verwendet.

Anschließend wurde zufällig anhand einer normalverteilten Wahrscheinlichkeitsdichtefunktion:

$$P_i = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \tag{2.1}$$

jeder der Borsten eine Abweichung von diesem Zustand beaufschlagt, sodass der Mittelpunkt einer Borste jeweils in axialer Richtung und Umfangsrichtung von der Position im dichtest gepackten Zustand abweicht. Abbildung 2.7 zeigt das Prinzip, mit welchem die Anordnungen generiert wurden. Die hexagonale Borstenanordnung stellt dabei den theoretisch dichtesten Zustand einer Anordnung zylinderförmiger Borsten dar, bei welcher die Lücken zwischen den Borsten minimiert und die Packungsdichte maximiert werden. (Fuchs & Haidn, 2017)



(a) Hexagonal dichtest gepackter Zustand der Borstenanordnung

(b) Einzelne Borsten nach zufälliger Verschiebung

Abbildung 2.7: Erzeugung einer quasi-chaotischen Borstenanordnung (Fuchs & Haidn, 2017)

Abbildung 2.8 zeigt exemplarisch eine so gebildete Verteilung der Mittelpunktabweichungen einzelner Borsten. Der Kreis beschreibt die Grenze der Verschiebungen, sodass es zu keiner Überschneidung zweier Borsten kommen kann (Fuchs & Haidn, 2017).

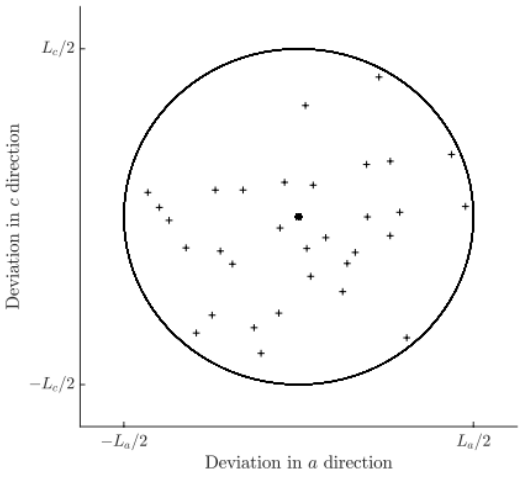


Abbildung 2.8: Exemplarische Abweichung der einzelnen Mittelpunkte vom dichtest gepackten Zustand (Fuchs & Haidn, 2017)

Eine anschließende numerische Simulation zeigte, dass eine steigende Varianz σ der Verschiebungen eine Senkung der Leckage zur Folge hat. Der Einfluss auf die Leckage war jedoch eher gering, sodass er in der Fehlertoleranz der Simulation lag. Die Zufälligkeit der Anordnung hatte im Bezug auf die Leckage bei gleichbleibender Varianz keinen feststellbaren Einfluss. Ferner wurde hervorgehoben, dass diese Methode zur Abbildung der radialen Strömungsmuster aus Kapitel 2.4 geeignet ist, und verwendet werden sollte, wenn eine detaillierte Analyse der Strömung erwünscht ist. (Fuchs & Haidn, 2017)

3. Zielsetzung

Untersuchung zur Bestimmung und Vorhersage der Borstenanordnung in einer Bürstendichtung

Ziel der vorliegenden Arbeit ist es, auf Grundlage von Bildern realer Bürstendichtungen die Anordnung des Borstenpakets zu erkennen, statistisch zu untersuchen und vorherzusagen. Die Bilder unterscheiden sich jeweils hinsichtlich des vorliegenden Betriebspunkts, des Fertigungszustands und der Lage in der Bürstendichtung. Zur Erkennung sollen geeignete Methoden der Computer Vision zur Anwendung kommen. Auf Grundlage der Ergebnisse der Erkennung soll eine repräsentative CAD-Modellierung einer Borstenanordnung ermöglicht werden. Dies bildet die Grundlage für eine CFD-Simulation, welche in dieser Arbeit jedoch nicht thematisiert wird.

4. Grundlagen

Im Nachfolgendem werden die Grundlagen zu verwendeten Methoden beschrieben, welche der Erkennung, Analyse und Vorhersage der Borstenanordnung dienen.

4.1 Verwendete Software

Zur Umsetzung der Borstenerkennung wird Python mit OpenCV benutzt. OpenCV ist eine Open-Source-Bibliothek, die Algorithmen zur Verarbeitung von Bildern mit klassischer Computer Vision und Machine Learning beinhaltet. Sie wird sowohl in der Wissenschaft, als auch in kommerziellen Lösungen genutzt, und erfreut sich einem globalen Bekanntheitsgrad. (OpenCV, 2023)

Zusätzlich wurden die Bibliotheken NumPy für die Verarbeitung numerischer Daten, Matplotlib zur Visualisierung ebendieser und SciPy für die Umsetzung statistischer Methoden verwendet.

4.2 Bildverarbeitung

Der nachfolgende Abschnitt liefert Grundlagen zu den Eigenschaften eines digitalen Bildes und eine Erklärung der verwendeten Verarbeitungsschritte, welche es einem Rechner ermöglichen, automatisiert, vergleichbare Ergebnisse wie ein menschlicher Betrachter zu erzielen.

4.2.1 Eigenschaften eines Bildes

Bilder können als regelmäßige Matrizen mit diskreten Koordinaten beschrieben werden (Burger & Burge, 2015). Das konventionelle Koordinatensystem eines Bildes, mit Ursprung oben links, ist in Abbildung 4.1 dargestellt. Neben der Koordinatenbezeichnung (u, v) für einen Pixel im Bild I ist auch (x, y) gängig. Sie werden in den nachfolgenden Formeln und Bildern synonym verwendet.

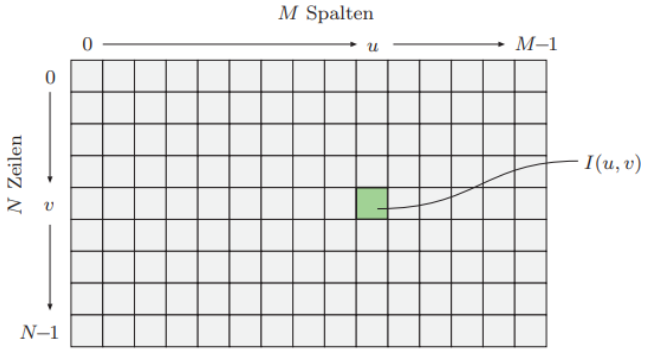


Abbildung 4.1: Konventionelles Koordinatensystem in der Bildverarbeitung (Burger & Burge, 2015)

Digitale Bilder haben eine endliche Menge von Zahlenwerten, um ein analoges Kamera-Signal auf einem Computer verarbeiten zu können. Die einzelnen Pixel sind typischerweise natürliche Zahlen, die 2^k unterschiedliche Werte annehmen können, wobei k die sogenannte Bit-Tiefe darstellt. Farbbilder werden üblicherweise in den drei Komponenten Rot, Grün und Blau mit 24-Bit, also jeweils 8-Bit pro Komponente kodiert. Demnach können die Intensitäten der einzelnen Intensitätsbilder I_R,I_G,I_B Werte von $[0, ..., 255]$ annehmen. Abbildung 4.2 zeigt eine typische Anordnung eines Farbbildes. Hierbei werden die Farbkomponenten in getrennten Arrays gleicher Dimension gespeichert. (Burger & Burge, 2015)

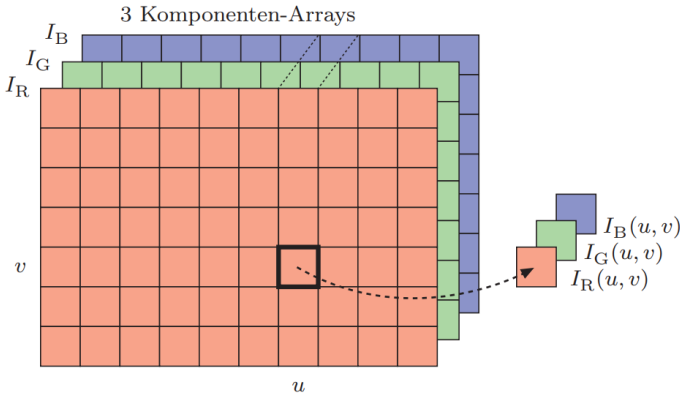


Abbildung 4.2: Drei-Komponenten-Array eines Farbbildes (Burger & Burge, 2015)

Eine Möglichkeit der Konvertierung eines Farbbildes mit drei Kanälen in Graustufen ist die Mittelung der Intensitätswerte des jeweiligen Kanals. Da die subjektive Wahrnehmung der Farben Rot, Grün und Blau unterschiedlich ist, werden die drei Kanäle aber in der Praxis sinnvollerweise gewichtet und der äquivalente Intensitätswert Y (auch Luminanz genannt) nach folgender Formel für jeden Pixel berechnet (Burger & Burge, 2015):

$$Y = 0,299 \cdot R + 0,587 \cdot G + 0,114 \cdot B \tag{4.1}$$

Die Werte der einzelnen Gewichte $w_R = 0,299$, $w_G = 0,587$ und $w_R = 0,114$ entstammen hierbei der standardisierten Kodierung von analogen TV-Farbsignalen (Burger & Burge, 2015). Nach der Konvertierung ist nur noch ein Kanal vorhanden. Abbildung 4.3 zeigt beispielhaft die Intensitätswerte eines Grauwert-Bild-Ausschnittes.

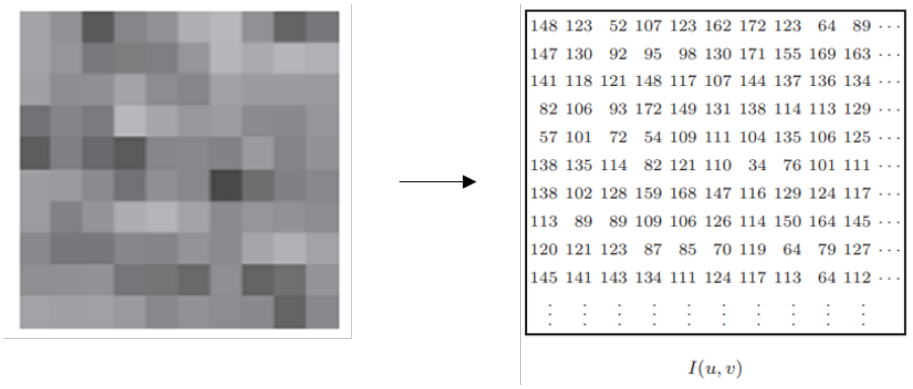


Abbildung 4.3: Ausschnitt eines Grauwert-Bildes mit zugehörigen Intensitätswerten (modifiziert nach Burger und Burge, 2015): Je höher der Intensitätswert, desto heller der Pixel.

4.2.2 Sobel-Filter

Der Sobel-Filter ist ein linearer Kantenerkennungs-Filter. Zunächst wird kurz das Prinzip der linearen Filterung erläutert.
Lineare Filterung bedeutet, dass Pixelwerte innerhalb der Umgebung eines Pixels linear miteinander verknüpft werden (Burger & Burge, 2015). Die Verarbeitung eines Pixels unter Berücksichtigung seiner Umgebung kann genutzt werden, um Bilder zu glätten, Details und Kanten (im Falle des Sobel-Filters) hervorzuheben oder das Rauschen zu mindern. Häufig wird hier die gewichtete Summe der Pixel in seiner Nachbarschaft $\mathcal{N}$ berechnet. Hierzu wird ein Kernel h verwendet, der jeweils die diskreten Pixel eines Bildes I elementweise summiert. Dieser Prozess wird als Faltung ($g = I * h$) bezeichnet (Szeliski, 2022):

$$g(x, y) = I * h = \sum_{k,l} I(x - k, y - l) \cdot h(k, l) \tag{4.2}$$

Das k und l stellen die Verschiebungen im Kernel und x, y die Koordinaten eines Pixels im Bild I dar, von welchem die Berechnung ausgeht.

Abbildung 4.4 zeigt die Faltung eines Bildes und damit berechnete Pixelwerte im Zielbild. Das Zielbild ist hierbei kleiner als das Ursprungs-Bild, da die Nachbarschaft $\mathcal{N}$ an den Rändern des Ursprungs-Bildes über die Grenze reicht und die Faltung an den betroffenen Pixeln nicht berechnet werden kann.

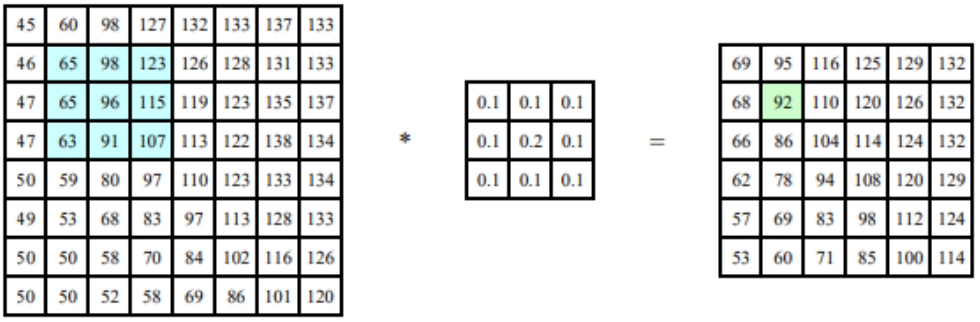


Abbildung 4.4: Faltung eines Bildes mit einem 3x3-Kernel (Szeliski, 2022): Die blauen Pixel beschreiben die Nachbarschaft $\mathcal{N}$ (blau markiert) und der grüne Pixel stellt den berechneten Zielpixel nach der elementweisen, gewichteten Summenbildung dar.

Allgemein lassen sich Kanten als Bereiche großer Intensitätsänderungen definieren. Der Betrag und die Richtung des Gefälles eines Pixels $I(x, y)$ in einem Bild wird hierbei über die partiellen Ableitungen beschrieben (Szeliski, 2022):

$$\nabla I(x, y) = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})(x, y) \tag{4.3}$$

Daraus lässt sich wiederum auf eine Kante sowie deren Orientierung im vorliegenden Bild schließen (Szeliski, 2022). Da die Intensitätswerte nur für Pixel, also diskrete Punkte in den Bildern bekannt sind, müssen die partiellen Ableitungen approximiert werden. Für ein Kamerasignal f und dem betrachteten Punkt u gilt (Burger & Burge, 2015):

$$\frac{df}{dx}(u) \approx \frac{f(u+1) - f(u-1)}{(u+1) - (u-1)} = \frac{f(u+1) - f(u-1)}{2} \tag{4.4}$$

Abbildung 4.5 zeigt das Prinzip der Approximierung der Tangentensteigung an einem betrachteten Punkt u.

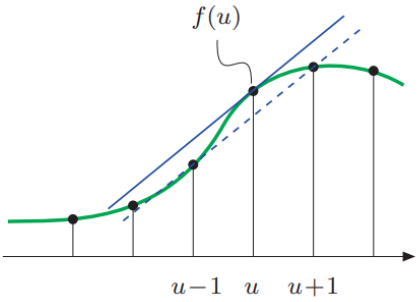


Abbildung 4.5: Approximierung der Tangentensteigung (Burger & Burge, 2015): Die Tangentensteigung an $f(x) = u$ (siehe blaue Linie) wird über den vorherigen ($x = u - 1$) und nachfolgenden Wert ($x = u + 1$) approximiert (siehe gestrichelte Linie). Die grüne Funktion ist ein möglicher Intensitätsverlauf des analogen Kamerasignals, welches über die Punkte diskretisiert wird.

Eine Approximierung der Ableitung nach Gleichung 4.4 lässt sich mit einem linearen Filter H_x für vertikale Kanten und H_y für horizontale Kanten realisieren (Burger & Burge, 2015):

$$H_x = 0.5 \cdot \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}, \quad H_y = 0.5 \cdot \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \tag{4.5}$$

Dieses Prinzip findet auch in den Sobel-Filtern G_x und G_y seine Anwendung (Burger & Burge, 2015):

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{4.6}$$

Aufgrund der Rauschanfälligkeit des einfachen Filters H, bestehen die Sobel-Filter aus mindestens drei Zeilen und drei Spalten. Bei der Berechnung des Gradienten wird besonderes Gewicht jeweils auf die mittlere Zeile beziehungsweise Spalte gelegt. (Burger & Burge, 2015)

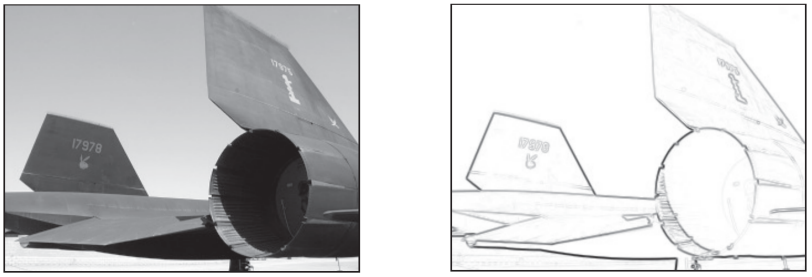
Nach der Faltung des Bildes I mit den Sobel-Operatoren G_x und G_y lässt sich die partielle Ableitung approximieren (Burger & Burge, 2015):

$$\nabla I(x, y) \approx \frac{1}{8} \cdot \begin{pmatrix} I * G_x \\ I * G_y \end{pmatrix} = \begin{pmatrix} I_x \\ I_y \end{pmatrix} \tag{4.7}$$

Für die Kantenstärke E folgt (Burger & Burge, 2015):

$$E(x, y) = \sqrt{I_x^2(x, y) + I_y^2(x, y)} \tag{4.8}$$

Ein beispielhaftes Ergebnis einer Sobel-Filterung ist in Abbildung 4.6 dargestellt.



(a) Originalbild (b) Kantenbild

Abbildung 4.6: Ergebnis einer Sobel-Filterung mit sichtbar verschiedenen Kantenstärken (Burger & Burge, 2015): Das Bild (b) wurde invertiert.

4.2.3 Median-Filter

Werden ein Pixel und seine Umgebung nicht über eine gewichtete Summenbildung verarbeitet, in der die Summe zweier Signale die gleiche Antwort hervorruft, wie die Summe der Antworten des Filters auf die Einzelsignale, so wird von nichtlinearen Filtern gesprochen. Nichtlineare Filter haben gegenüber linearen Filtern spezifische Charakteristika, die einen Einsatz lohnenswert gestalten können. Ein Beispiel hierfür ist der Median-Filter. Dieses Verfahren ist insbesondere bei hochfrequentem *shot noise* effektiv, also sehr intensiven Rauschpixeln. (Szeliski, 2022)

Für eine gegebene Nachbarschaft $\mathcal{N}$ eines Pixels $I(x, y)$ gilt bei einer Median-Filterung (Burger & Burge, 2015):

$$I'(x, y) \leftarrow median(I(x + i, y + j | (i, j) \in \mathcal{N})) \tag{4.9}$$

Der Medianwert beschreibt hierbei den Wert, der die beobachteten Intensitäten in der Pixelumgebung in zwei gleich große Hälften teilt, wobei jeder Teil 50% der Verteilung beinhaltet (Hedderich & Sachs, 2020). Der Prozess zur Medianfindung in der Umgebung eines Pixels ist in Abbildung 4.7 dargestellt.

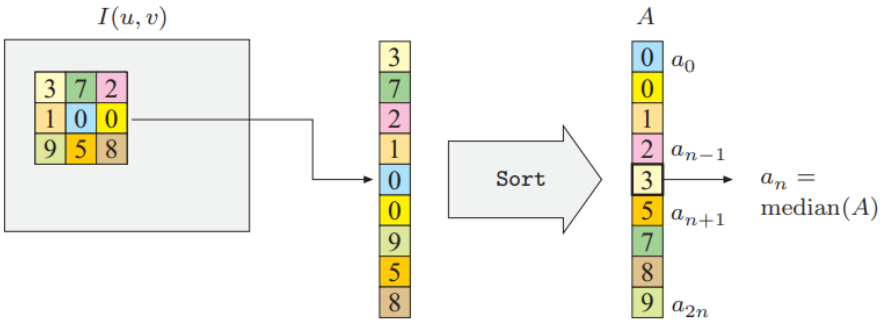


Abbildung 4.7: Medianfindung in der 3x3-Nachbarschaft eines Pixels $I(u, v)$ (Burger & Burge, 2015): Die Werte werden zunächst sortiert und dann in zwei gleich große Hälften geteilt.

Abbildung 4.8 zeigt das Ergebnis einer Median-Filterung an einem verrauschten Bild eines Fahrzeugs.

(a) Originalbild mit Rauschen

(b) Bild nach Median-Filterung

Abbildung 4.8: Ergebnis einer Median-Filterung (Chen et al., 2020)

4.2.4 Adaptiver Histogramm-Ausgleich

Das Ziel des Histogramm-Ausgleichs ist die Anpassung eines Bildes auf eine annähernd übereinstimmende Intensitätsverteilung und Änderung des Kontrasts (Burger & Burge, 2015). Dabei wird das kumulative Histogramm H des Bildes verwendet, welches die Summe der Bildintensitäten i des Histogramms h darstellt (Burger & Burge, 2015):

$$H(i) = \sum_{j=0}^{i} h(j) \text{ für } 0 \leq i < k \tag{4.10}$$

Im Falle einer übereinstimmenden Intensitätsverteilung besteht das Histogramm des Bildes aus gleich großen Intensitäten. Das resultierende kumulative Histogramm ist keilförmig und das Ziel des Histogramm-Ausgleichs. (Burger & Burge, 2015)
Hierzu wird die Transformationsfunktion f_{eq} genutzt, die in Abhängigkeit von dem

Intensitätswert eines Pixels i und seiner relativen Häufigkeit den Wert $f_{eq}(i)$ zuweist (Burger & Burge, 2015):

$$f_{eq}(i) = \frac{H(i) \cdot (K-1)}{M \cdot N}) \tag{4.11}$$

Hierbei stellen M, N die Dimensionen des Bildes dar und $[0,\ K-1]$ den zugehörigen Wertbereich der Pixel. Mittels der Transformation können die Intensitäten des kumulativen Histogramms nach links oder rechts verschoben werden, sodass sich ein annähernd keilförmiges kumulatives Histogramm bildet. (Burger & Burge, 2015)
Der Ausgleichsprozess ist in Abbildung 4.9 dargestellt.

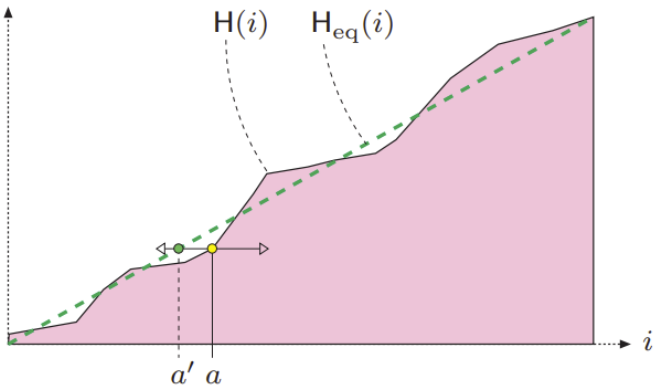


Abbildung 4.9: Ausgleich eines kumulativen Bildhistogramms $H(i)$ (Burger & Burge, 2015): Pixel mit der Intensität a erhalten den Wert a'.

Wird zur Ermittlung der Transformationsfunktion für die Pixel nicht das gesamte Bild, sondern nur lokale Pixelblöcke verwendet, so wird von einem adaptiven Histogrammausgleich gesprochen. Um das Bild trotz der Berechnung in lokalen Pixelblöcken zu harmonisieren, wird die für einen Pixel geltende Ausgleichsfunktion über eine Interpolation ermittelt. Hierfür werden auf einem Gitter befindliche Stützstellen, die die jeweilige Ausgleichsfunktionen ihrer Nachbarschaft beinhalten, verwendet. (Pizer et al., 1987)
Für einen Pixel mit den Koordinaten (s, t) relativ zu seinen Stützstellen und den Transformationsfunktionen $(f_{00}, ..., f_{11})$ ergibt sich dabei die Ausgleichsfunktion $f_{s,t}$ (Szeliski, 2022):

$$f_{s,t}(i) = (1-s)(1-t)f_{00}(i) + s(1-t)f_{10}(i) + (1-s)tf_{01}(i) + stf_{11}(i) \tag{4.12}$$

Abbildung 4.10 zeigt eine graphische Beschreibung dieser Interpolation.

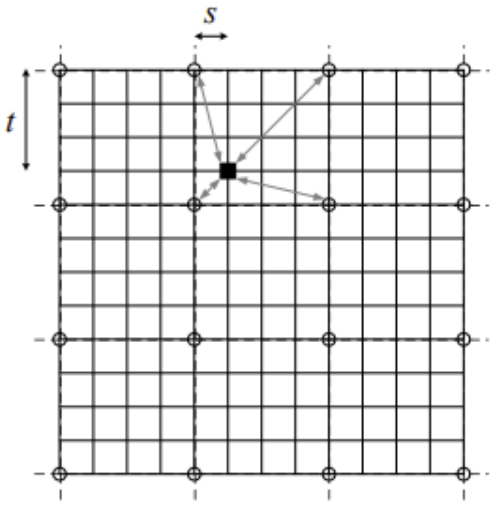


Abbildung 4.10: Lokaler Histogrammausgleich für einen Pixel (Szeliski, 2022): Der Pixel hat die relativen Koordinaten (s, t) zu seiner Stützstellenumgebung, dargestellt durch vier Kreise.

Da ein adaptiver Histogrammausgleich neben einer Verstärkung des Signals zu einer Verstärkung des Rauschens führen kann, ist eine Begrenzung der Kontrastverstärkung teils sinnvoll. Dies ist gleichbedeutend mit der Begrenzung der Steigung der Transformationsfunktion, also des kumulativen Histogramms. Da die Ableitung des kumulativen Histogramms auch das ursprüngliche Histogramm darstellt, kann hierbei eine Grenze eingeführt werden, das *Clip Limit*. Die betroffenen Pixel werden gleichförmig verteilt. Dieses Verfahren wird CLAHE (*Contrast Limited Adaptive Histogram Equalization*) genannt. (Pizer et al., 1987)
Abbildung 4.11 zeigt die Auswirkung von CLAHE auf ein Histogramm.

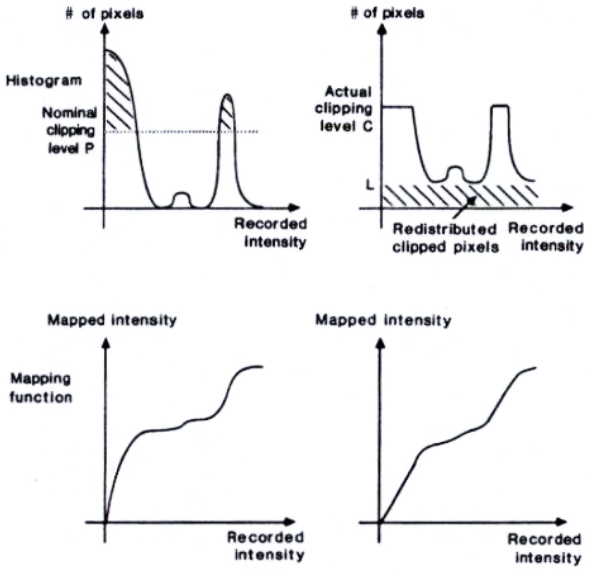
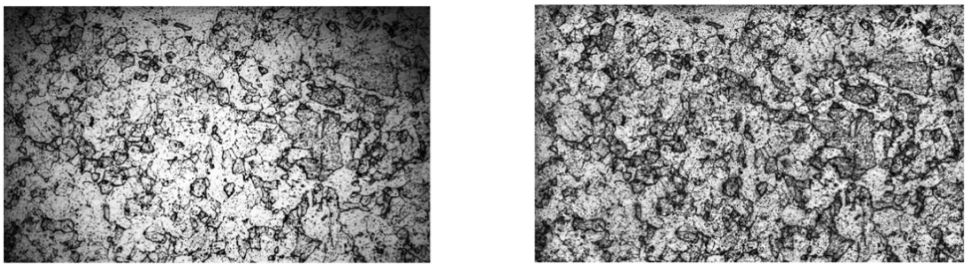


Abbildung 4.11: Einführung eines Clipping-Levels und Konsequenzen für das Histogramm und kumulierte Histogramm (Pizer et al., 1987)

Abbildung 4.12 zeigt den Vergleich eines globalen Histogrammausgleichs und des CLAHE-Verfahrens anhand eines Mikroskop-Bildes eines Kesselrohrs von einem Kraftwerk.



(a) Histogrammausgleich (b) CLAHE

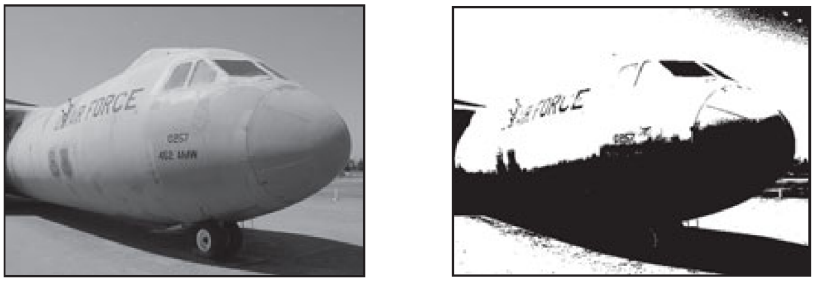
Abbildung 4.12: Vergleich von globalem und lokalen Histogrammausgleich (Choi et al., 2019): CLAHE zeigt ein deutlich gleichmäßigeres Bild.

4.2.5 Thresholding

Die Umwandlung von Grauwertbildern in Binärbilder, die nur weiße oder schwarze Pixel enthalten, ist durch Thresholding (deutsch: Schwellenwertverfahren) möglich (Szeliski, 2022). Ein einfaches Schwellwertverfahren lautet:

$$I_{\mathrm{B}}(x,y) = \begin{cases} 255 & \text{wenn } I(x,y) > \text{Schwellenwert} \\ 0 & \text{sonst} \end{cases} \tag{4.13}$$

Dadurch lassen sich Bereiche, die dieses Kriterium erfüllen, in dem Bild hervorheben und gezielt weiterverarbeiten. Das Ergebnis eines Schwellenwertverfahrens ist in Abbildung 4.13 zu sehen.



(a) Originalbild (b) Bild nach Thresholding

Abbildung 4.13: Ergebnis eines Schwellenwertverfahrens (Burger & Burge, 2015): Pixel, die über dem Schwellwert ($i = 128$) liegen werden weiß, der Rest schwarz markiert.

4.2.6 OTSU-Thresholding

OTSU-Thresholding ist ein nichtparametrisches Verfahren zur Bestimmung eines optimalen Schwellenwerts zur Separierbarkeit zweier Klassen (z.B. Objekten und dem

Hintergrund) anhand der Verteilung von Pixelintensitäten in einem Bild. Grundlage ist das Grauwert-Histogramm eines Bildes und die zugehörigen Wahrscheinlichkeiten p_i jedes Intensitätswerts n_i von 0 bis 255 (Otsu, 1979):

$$p_i = \frac{n_i}{N} \tag{4.14}$$

Eine Unterteilung der Pixel mit einem Schwellenwert k in zwei Klassen C_0 $[1, ..., k]$ und C_1 $[k+1, ..., L]$ liefert so die Wahrscheinlichkeiten der jeweiligen Klasse ω_0 und ω_1 (Otsu, 1979):

$$\omega_0 = \sum_{i=1}^{k} p_i, \quad \omega_1 = \sum_{i=k+1}^{L} p_i = 1 - \omega_0 \tag{4.15}$$

sowie die Mittelwerte der beiden Klassen μ_0, μ_1 des Histogramms (Otsu, 1979):

$$\mu_0 = \sum_{i=1}^{k} ip_i/\omega_0, \quad \mu_1 = \sum_{i=k+1}^{L} ip_i/\omega_1 \tag{4.16}$$

Für Varianzen der beiden Klassen, σ_0 und σ_1, gilt (Otsu, 1979):

$$\sigma_0^2 = \sum_{i=1}^{k} (i - \mu_0)^2 p_i/\omega_0, \quad \sigma_1^2 = \sum_{i=k+1}^{L} (i - \mu_1)^2 p_i/\omega_1 \tag{4.17}$$

Aus diesen Größen wird σ_W, welches die Varianz innerhalb der Klassen darstellt, und σ_B, die Varianz zwischen den Klassen gebildet (Otsu, 1979):

$$\sigma_W^2 = \omega_0\sigma_0^2 + \omega_1\sigma_1^2, \quad \sigma_B^2 = \omega_0\omega_1(\mu_1 - \mu_0)^2 \tag{4.18}$$

Das Optimierungproblem besteht darin, einen Schwellenwert k zu finden, bei dem das Kriterium λ maximiert wird (Otsu, 1979):

$$\lambda = \sigma_B^2/\sigma_W^2 \tag{4.19}$$

Die Varianz sollte optimalerweise innerhalb der Klassen möglichst klein, und die zwischen den Klassen möglichst groß sein.

4.2.7 Dilation

Durch morphologische Filter ist eine gezielte Änderung der Struktur von Binärbildern möglich. Die Dilation ist ein solches Verfahren, und kommt dem Wachsen der

Strukturen gleich. Sie entspricht der Vektorsumme aller Vordergrundpixel p des Binärbilds I mit den Vordergrundpixeln q eines Strukturelements H (Burger & Burge, 2015):

$$I \oplus H \equiv \{(p+q) \mid \text{für alle } p \in I, q \in H\} \tag{4.20}$$

Das Ergebnis einer solchen Summierung ist in Abbildung 4.14 dargestellt.

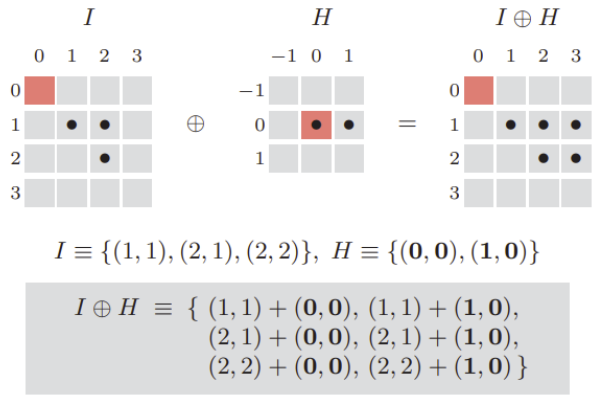


Abbildung 4.14: Dilation eines Binärbilds I mit dem Strukturelement H (Burger & Burge, 2015): Der Ursprung des jeweiligen Koordinatensystems ist rot markiert.

Die Form und Größe des Strukturelements beeinflusst das Ergebnis der Dilation. In der Praxis werden häufig wegen ihrer isotropen Eigenschaften scheibenförmige Strukturelemente verwendet (Burger & Burge, 2015). Die Dilation findet in der Watershed-Segmentierung, welche in Kapitel 4.2.9 erklärt wird, ihre Anwendung.

4.2.8 Distanztransformation

Bei einer Distanztransformation eines Binärbilds I_B wird jedem der Pixel dieses Bildes der Wert D, die Distanz zum nächsten Hintergrundpixel mit Wert 0, statt der Intensität zugeordnet (Szeliski, 2022):

$$D(x,y) = \min_{k,l:I_B(k,l)=0} d(x-k, y-l) \tag{4.21}$$

Hierbei stellt $d(k,l)$ eine Distanzmetrik dar, wie etwa die euklidische Distanz (Szeliski, 2022).

$$d(k,l) = \sqrt{k^2+l^2} \tag{4.22}$$

Ein möglicher Anwendungsfall ist die Marker-Findung bei der Watershed Segmentation in Kapitel 4.2.9.

4.2.9 Watershed Segmentation

Watershed Segmentation ist ein Segmentierungsverfahren, welches auf Region Growing beruht. Dabei entspricht ein Bild dem Modell einer topographischen Oberfläche, die sukzessive und konstant geflutet wird. Die Stellen, an welchen Minima sind, stellen die Ausgangspunkte zur Flutung dar. Das Wasser steigt über die Oberfläche konstant. An den Stellen, wo die Flut zweier Minima zusammenfließt, wird die namensgebende Wasserscheide zur Trennung von Regionen gebildet. (Meyer, 1992)
Dieser Flutungsprozess ist schematisch in Abbildung 4.15 dargestellt:

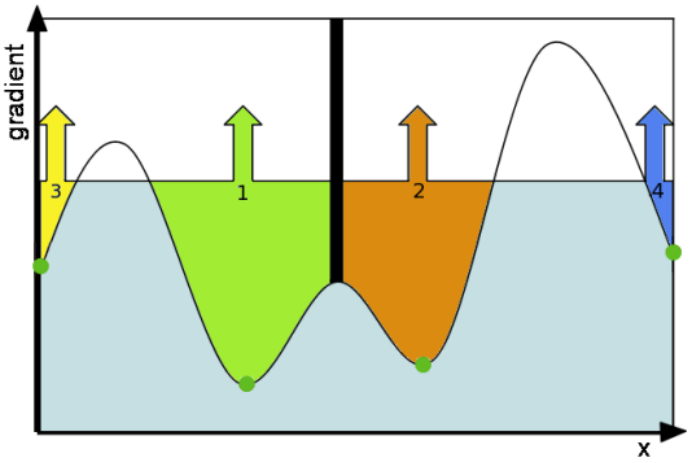
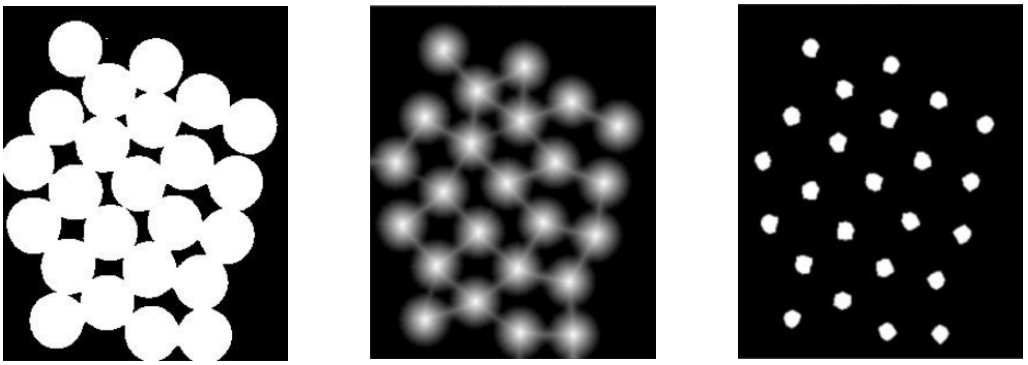


Abbildung 4.15: Flutungsprozess bei der Watershed Segmentation (Roudier et al., 2008)

Neben den Minima können auch sogenannte Marker benutzt werden, die jeweils einem Objekt zugeordnet werden können, da die Nutzung bloßer Minima zu Übersegmentierung, also einer Unterteilung in zu kleine Segmente führen kann (Meyer, 1992). Abbildung 4.16 zeigt ein Beispielbild überlappender Objekte, anhand derer die markerbasierte Watershed-Segmentierung durchgeführt werden soll.

Abbildung 4.16: Beispielbild überlappender Objekte, die mit der Watershed Segmentation getrennt werden (OpenCV, o.D.)

Zunächst werden die Marker der Objekte definiert. Eine Möglichkeit zur Findung der Marker ist die Konvertierung in ein Binärbild und anschließende Anwendung der Distanztransformation aus Kapitel 4.2.8 sowie eines Schwellenwertverfahrens. (OpenCV, o.D.) Dies ist in Abbildung 4.17 zu sehen.



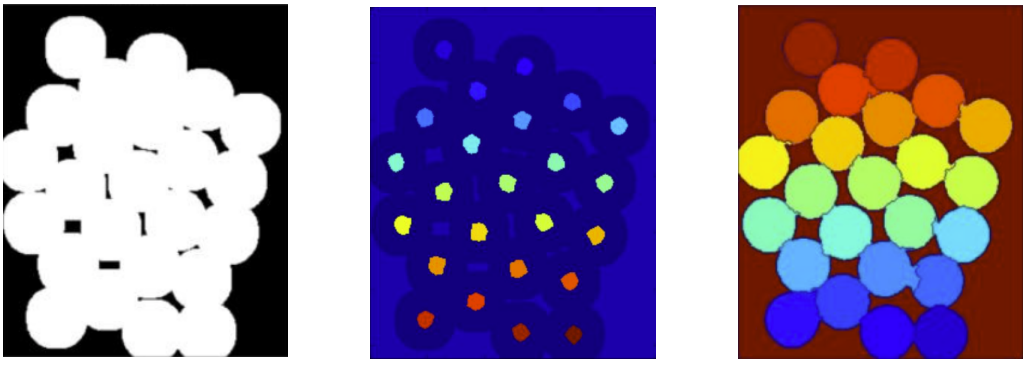
(a) Binärbild

(b) Anwendung der Distanztransformation

(c) Marker nach Anwendung eines Schwellenwertverfahrens

Abbildung 4.17: Prozess der Markerfindung (OpenCV, o.D.)

Zusätzlich zur Markerfindung, kann die Region des Flutungsprozesses definiert werden. Hierzu wird zunächst eine Dilation (vgl. Kapitel 4.2.7) des Binärbildes durchgeführt und von dem Bild der gefundenen Marker abgezogen. Der verbleibende Rest entspricht den Schranken innerhalb welcher die Flutung stattfinden kann. (OpenCV, o.D.) Dieser Prozess und das letztendliche Ergebnis nach der Flutung ist in Abbildung 4.18 zu sehen.



(a) Dilation des Binärbildes

(b) Rest (dunkelblau)

(c) Ergebnis

Abbildung 4.18: Steuerung der Flutung und Ergebnis der Watershed-Segmentierung (OpenCV, o.D.)

Weitere Details über den verwendeten Flutungs-Algorithmus sind Meyer (1992) zu entnehmen.

4.2.10 Hough-Transform-Linienerkennung

Um auf Grundlage eines Bildes Linien zu erkennen, ist der Hough-Transform einsetzbar. Hierbei werden Geraden anhand der normal parametrisierten Form beschrieben (Duda & Hart, 1972):

$$x \cdot cos(\theta) + y \cdot sin(\theta) = r \tag{4.23}$$

Der Parameter r beschreibt hierbei die Distanz zwischen dem Ursprung des Koordinatensystems und der Geraden. θ beschreibt den zugehörigen Normalenwinkel. Jede Gerade in kartesischen Koordinaten (x/y) Bildraum entspricht einem einzigen Punkt im Parameterraum (θ/r). Einzelne Punkte (x_i, y_i) erscheinen im Parameterraum als Sinus-Kurven. Demnach entsprechen Punkte auf einer Geraden im Bildraum, gemeinsamen Schnittpunkten der jeweiligen Sinus-Kurven im Parameterraum. (Duda & Hart, 1972)
Abbildung 4.19 zeigt Geraden, die zugehörigen Winkel und den Vergleich dieser im x-y-Bildraum und θ/r Parameterraum.

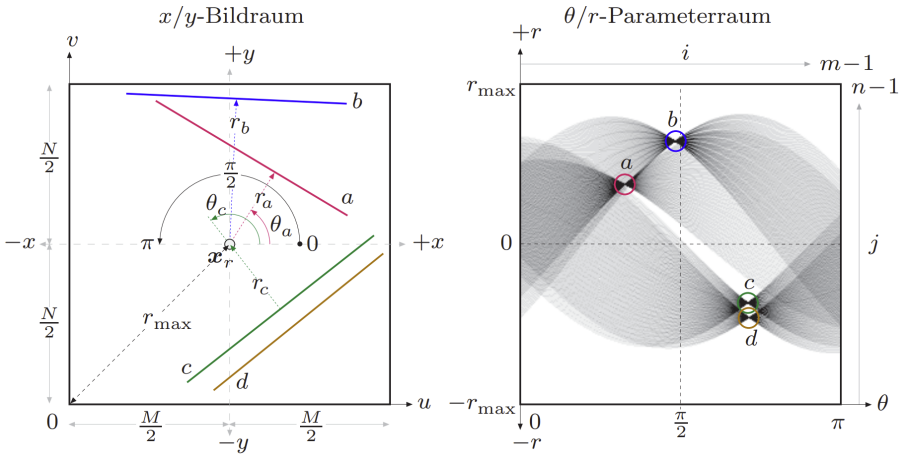


Abbildung 4.19: Vergleich von Geraden im x/y-Bildraum und θ/r-Parameterraum (Burger & Burge, 2015)

Im Rahmen der Linienerkennung werden demzufolge die einzelnen Pixel eines Bildes in den Parameterraum transformiert und aus den resultierenden Kurven die Schnittpunkte (r_i,θ_i) bestimmt (Duda & Hart, 1972). Hierbei kann ein Schwellenwert von der Zahl gemeinsamer Schnittpunkte festgelegt werden, also eine minimale Zahl von Punkten einer Linie, damit sie als solche detektiert wird, um Falscherkennungen zu vermeiden.

4.2.11 Connected Components

Um anhand eines Binärbildes zusammenhängende Objekte zu erkennen, können Connected Components identifiziert werden. Connected Components sind Regio-

nen, in denen benachbarte Pixel den gleichen Intensitätswert haben (Szeliski, 2022). Abbildung 4.20 zeigt exemplarisch die Connected Components eines Binärbilds.

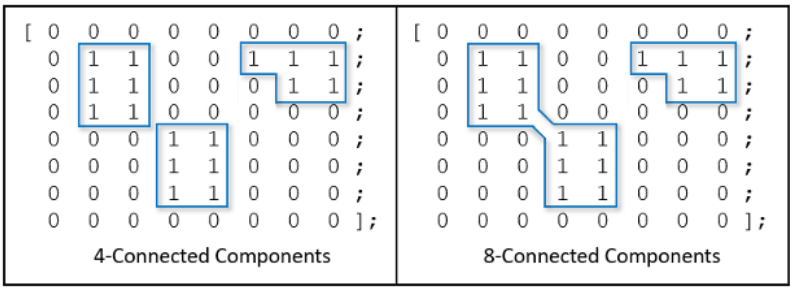


Abbildung 4.20: Connected Components (MathWorks, o.D.): 8-Connected Components erlauben zusätzlich eine diagonale Verbundenheit der Pixel

Aus der Verbundenheit dieser Pixel lassen sich nützliche Eigenschaften für ihre Region ableiten, wie etwa die Fläche (Zahl der Pixel) oder der Schwerpunkt (Durchschnitt der x- und y-Werte) (Szeliski, 2022).

4.2.12 Rotationsmatrix

Im Zweidimensionalen kann für einen Pixel eine Koordinatentransformation durch eine Rotationsmatrix R mit dem Rotationwinkel θ durchgeführt werden:

$$R = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \tag{4.24}$$

Bei dieser Operation bleiben die euklidischen Distanzen der Pixel innerhalb des Bildes erhalten (Szeliski, 2022).

4.3 Statistische Methoden

Die statistischen Methoden dienen der Analyse und Vorhersage der Borstenanordnung anhand der vorliegenden Bilder. Die Grundlage hierzu bilden statistische Tests, deren Eigenschaften zunächst allgemein erläutert werden. Anschließend werden die einzelnen statistischen Tests, das verwendete Verfahren zur Anpassung von Wahrscheinlichkeitsverteilungen an Beobachtungen und das Verfahren zur Simulation von Wahrscheinlichkeitsverteilungen beschrieben.

4.3.1 Eigenschaften statistischer Tests

Mittels statistischer Tests kann geschlussfolgert werden, ob eine festgelegte Nullhypothese angenommen oder abgelehnt wird. Hierbei wird eine Teststatistik (Prüfgröße) auf Basis einer oder mehrerer Stichproben ermittelt. Die Teststatistik folgt

dabei einer Verteilung, aus welcher sich auch der kritische Bereich ableitet. Die Entscheidung über die Annahme oder Ablehnung der Nullhypothese erfolgt anhand der Grenze zum kritischen Bereich, dem Signifikanzniveau α. Der p-Wert stellt dabei die Wahrscheinlichkeit dar, mit welcher eine Aussage zur vorliegenden Stichprobe getroffen wird, und die Wahrscheinlichkeit die gleiche Aussage über die Hypothese bei Betrachtung einer weiteren Stichprobe zu treffen. Liegt er unter dem Signifikanzniveau α, so wird die Hypothese verworfen, die Teststatistik liegt im kritischen Bereich, und die Ablehnung wird als statistisch signifikant bezeichnet. (Bracke, 2022)
Es wird zwischen dem Fehler 1. und dem Fehler 2. Art unterschieden. Der Fehler 1. Art geschieht, wenn die Nullhypothese unberechtigterweise abgelehnt wird. Bei dem Fehler 2. Art wird die Nullhypothese unberechtigterweise beibehalten. (Hedderich & Sachs, 2020)
Die Größe des Fehlers 1. Art stellt das Signifikanzniveau des Tests dar, wohingegen der Fehler 2. Art im Falle eines Signifikanztests nicht berechnet werden kann. Beide Fehlergrößen sind voneinander abhängig. (Bracke, 2022)
Die Wahl eines geeigneten α ist nicht einfach möglich (Hedderich & Sachs, 2020). Im weiteren Verlauf der Arbeit wird $\alpha = 0,05$ verwendet, ein Wert, der in der Praxis in vielen Fällen standardmäßig verwendet wird (Miller & Ulrich, 2019).

4.3.2 Levene-Test

Mithilfe des Levene-Tests kann für k unabhängige Stichproben die Varianzhomogenität getestet werden. Die Nullhypothese H_0 lautet (Hedderich & Sachs, 2020):

$$H_0 : \sigma_1^2 = \sigma_2^2 = ... = \sigma_k^2 \tag{4.25}$$

Die Berechnung der Teststatistik $\hat{F}$ ist Hedderich und Sachs (2020) zu entnehmen.

4.3.3 Mann-Whitney-U-Test

Der Mann-Whitney-U-Test ist ein nichtparametrischer, verteilungsunabhängiger Test zum Vergleich der Schwerpunkte zweier Verteilungen. Dabei wird auch die Unabhängigkeit der Stichproben vorausgesetzt. (Bracke, 2022)
Es werden zunächst n Stichprobenwerte der beiden Verteilungen der Größe nach aufsteigend geordnet (Hedderich & Sachs, 2020):

$$x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(i)} \leq \cdots \leq x_{(n)} \tag{4.26}$$

Hierbei beschreibt i den Rang des Strichprobenwerts. Die Nullhypothese H_0 lautet (Hedderich & Sachs, 2020):

$$H_0 : P(X_1 > X_2) = \frac{1}{2} \tag{4.27}$$

Wobei X_1 eine beliebig gezogene Beobachtung aus der ersten Grundgesamtheit und X_2 eine zufällige Beobachtung aus der zweiten Grundgesamtheit darstellt. (Hedderich & Sachs, 2020)

Die Teststatistik stellt der U-Wert dar, der kleinere Wert von U_1 und U_2 (Hedderich & Sachs, 2020):

$$U_1 = mn + \frac{m(m+1)}{2} - R_1 \qquad U_2 = mn + \frac{n(n+1)}{2} - R_2 \tag{4.28}$$

Dabei sind m, n die jeweiligen Stichprobengrößen und R_1, R_2 die zugehörigen Rangsummen nach kombinierter Sortierung nach Gleichung 4.26.

4.3.4 Kolmogorov-Smirnov-Test

Mit einem Anpassungstest wird untersucht, ob die Ergebnisse eines Zufallsexperiments durch ein stochastisches Modell hinreichend genau beschrieben werden (Kolonko, 2008). Ein möglicher Anpassungstest ist der Kolmogorov-Smirnov-Anpassungstest, der Unterschiede in der Streuung, Schiefe, Exzess und der zentralen Tendenz erfassen kann. Die Nullhypothese lautet hierbei (Hedderich & Sachs, 2020):

$$H_0 : \text{Die Daten folgen einer theoretischen Verteilung } F_0 \tag{4.29}$$

Die Teststatistik $\hat{D}$ berechnet sich aus der maximalen Differenz zwischen den beobachteten Daten und einer theoretischen Verteilungsfunktion F_0 mit dem Rang i, der zugehörigen Beobachtung $x_{(i)}$ und dem Stichprobenumfang n (Hedderich & Sachs, 2020):

$$\hat{D} = \max_{1 \leq i \leq n} |F_0(x_{(i)}) - \frac{i}{n}| \tag{4.30}$$

Wenn sie beibehalten wird, könnte die Stichprobe aus einer Grundgesamtheit stammen, die dem hypothethischen Modell folgt (Bracke, 2022). Vor der Durchführung des Kolmogorov-Smirnov-Tests wird die Maximum Likelihood Estimation in Kapitel 4.3.5 durchgeführt um die jeweiligen hypothetischen Verteilungen an die Beobachtungen anzupassen.

4.3.5 Maximum Likelihood Estimation

Maximum-Likelihood-Estimation (MLE) ist ein Schätzverfahren zur Bestimmung unbekannter Parameter eines Verteilungsmodells. Dabei soll die Likelihood-Funktion L in Abhängigkeit ihres Parameters ϑ maximiert werden (Hedderich & Sachs, 2020):

$$L = L(\vartheta) = \prod_{i=1}^{n} P(X_i = x_i|\vartheta) \tag{4.31}$$

Hier stellt X_i eine Zufallsvariable und x_i die Realisierung einer Zufallsvariablen dar. (Hedderich & Sachs, 2020). Das Maximum tritt bei jenem Parameter ϑ auf, bei dem die größte gemeinsame Dichte vorliegt. Abbildung 4.21 zeigt die Idee der MLE für eine einzelne Realisierung x und den drei Parametern $\vartheta \in \{0, 1, 3\}$. (Messer & Schneider, 2019)

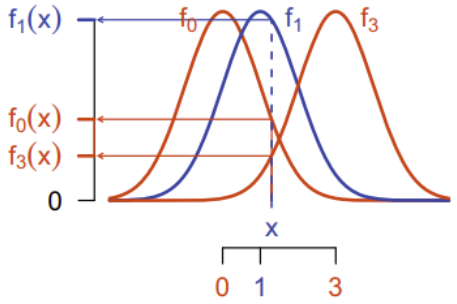


Abbildung 4.21: Idee der MLE (Messer & Schneider, 2019): Für $\vartheta = 1$ liegt die größte Dichte der Realisierung x vor.

Im Falle einer fehlenden analytischen Lösung (nach partieller Ableitung der Likelihood-Funktion), muss die Lösung teils unter Anwendung nicht trivialer numerischer Verfahren erfolgen (Hedderich & Sachs, 2020). Diese Verfahren werden im Rahmen dieser Arbeit nicht weiter thematisiert.

4.3.6 Inversionsmethode

Zur Simulation einer Wahrscheinlichkeitsverteilung F kann das Inversionsprinzip angewendet werden. Dabei wird die Inverse der Verteilungsfunktion F, $F^{-1}(r)$ verwendet. Es gilt allgemein: (Kolonko, 2008)

$$F^{-1}(r) := \inf\{t \in \mathbb{R} \,|\, F(t) \geq r\}, \quad r \in [0, 1] \tag{4.32}$$

Ist U eine im Intervall von [0,1] gleichverteilte Zufallsvariable, so gilt: (Kolonko, 2008)

$$P(F^{-1}(U) \leq t) = F(t), t \in \mathbb{R} \tag{4.33}$$

Abbildung 4.22 zeigt das Prinzip der Inversionsmethode, bei welchem über eine gleichverteilte Zufallsvariable (siehe y-Achse) und die Inversion der Verteilungsfunktion F_X exponentialverteilte Zufallszahlen (siehe x-Achse) erzeugt werden.

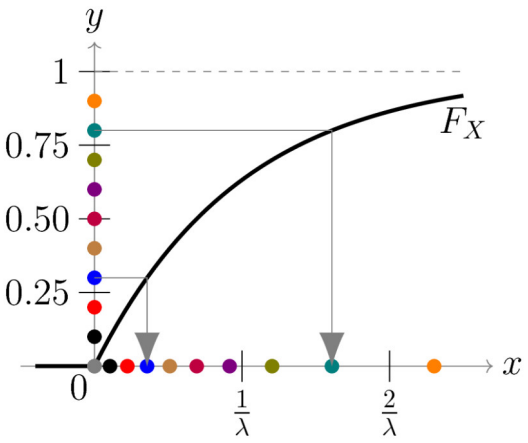


Abbildung 4.22: Inversionsmethode bei einer Exponential-Verteilung (Wikimedia-Commons, 2016)

5. Rahmenbedingungen

In diesem Kapitel wird die Bildentstehung beschrieben, die Bilder gesichtet und auf ihre Merkmale untersucht. Aus dieser Untersuchung leiten sich die Anforderungen an das Erkennungssystem ab, welche die Grundlage für die Implementierung und Evaluation der Algorithmen darstellen.

5.1 Aufbau des Prüfstands

Der Aufbau des Prüfstands, welcher radiale Aufnahmen der Bürstendichtung ermöglichte, ist in Abbildung 5.1 dargestellt. Die Bilder entstammen der Arbeit von Schwarz et al. (2014). Hierzu wurde ein angewinkelter Spiegel verwendet, der auf die Unterseite der Bürstendichtung gerichtet ist. Die Aufnahmen erfolgten mit einem Digitalmikroskop. Die Bürstendichtung kann rotiert werden, um Aufnahmen an mehreren Stellen des runden Bauteils zu ermöglichen. Zusätzlich kann die Druckdifferenz verändert werden. Diese Bilder stellen die Grundlage dar, um die Anordnung des Borstenpakets zu erkennen und zu verarbeiten.

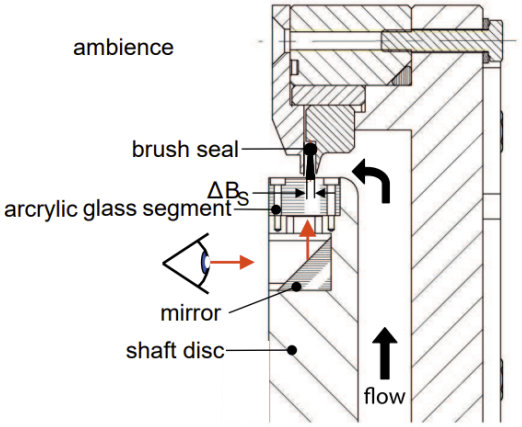


Abbildung 5.1: Aufbau des Prüfstands zur Aufnahme radialer Bilder (Schwarz et al., 2014)

5.2 Varianten der Dichtungen

Es liegen jeweils Bildersammlungen dreier Dichtungsvarianten vor, die sich hinsichtlich des Inklinationswinkels der Borsten unterscheiden. Die Bilder des jeweiligen Da-

tensets mit ihren unterschiedlichen Inklinationswinkeln werden im Nachfolgenden gemäß der Bezeichnungen in Tabelle 5.1 referenziert.

Bezeichnung	**BP-8**	**BP-5**	**BP-0**
Inklinationswinkel φ	8°	ca. 5°	0°

Tabelle 5.1: Varianten der Dichtung

Abgesehen von dem Inklinationswinkel sind die Geometrieparameter, sichtbar in Tabelle 5.2, aller Bürstendichtungen identisch. (Schwarz et al., 2014)

Borstendurchmesser D	$0,07$mm
Legewinkel λ	45°
Packungsdichte ρ	200 Borsten/mm

Tabelle 5.2: Geometrieparameter aller Dichtungen

Die Bilder haben eine Auflösung von 4800 x 3600 Pixeln. Die Aufnahme der Bürstendichtungen erfolgte in Teilstücken, sodass für einen gegebenen Betriebspunkt einer Dichtung jeweils 24 Bilder entstanden.

5.2.1 Beispielbilder der Dichtungsvarianten

Im Nachfolgenden werden anhand eines Beispielbilds die Bilder der jeweiligen Dichtungsvariante beschrieben und etwaige Besonderheiten hervorgehoben.

Die BP-8-Bildersammlung zeigt eine gute Konsistenz hinsichtlich des Auflichts, wobei einige Borsten heller sind, als andere. Die Borsten sind visuell gut trennbar. Einige der Borsten überragen den Stützring. Dies lässt sich beispielsweise auf Transportschäden und bleibende plastische Biegung der feinen Drähte zurückführen. Abbildung 5.2 zeigt ein Beispielbild der Bildersammlung.

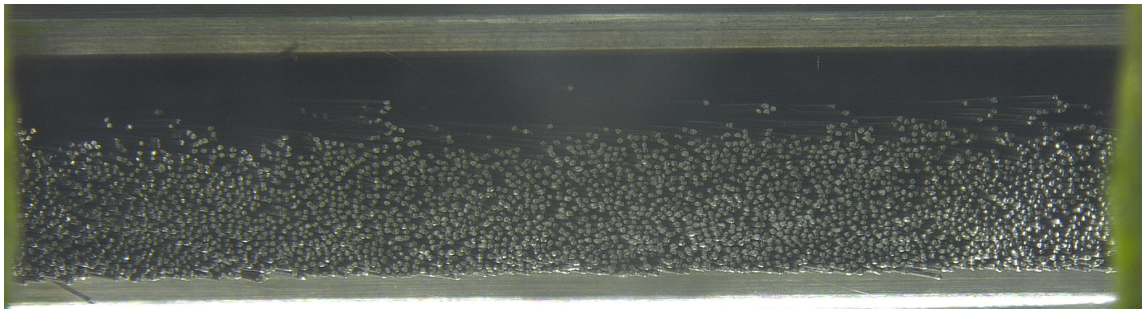


Abbildung 5.2: Beispielbild Borstenpaket BP-8: Im linken unteren Bereich ist eine Borste zu erkennen, die den Stützring überragt.

Die Bilder der BP-5-Bildersammlung (sichtbar in Abbildung 5.3) zeigen eine große Verschiedenheit im Hinblick auf die Helligkeit der Borsten. Der Bereich in der Mitte des Bildes ist in der Bildersammlung ausnahmslos dunkel. Links ist teilweise eine

höhere Belichtung zu erkennen. Die Trennung von Borste und Hintergrund ist in Teilbereichen möglich. Zusätzlich sind hier mehrere vertikale Artefakte durch Spiegelungen oder Kratzer zu erkennen, welche sich ungünstigerweise mit den Borsten überlagern.

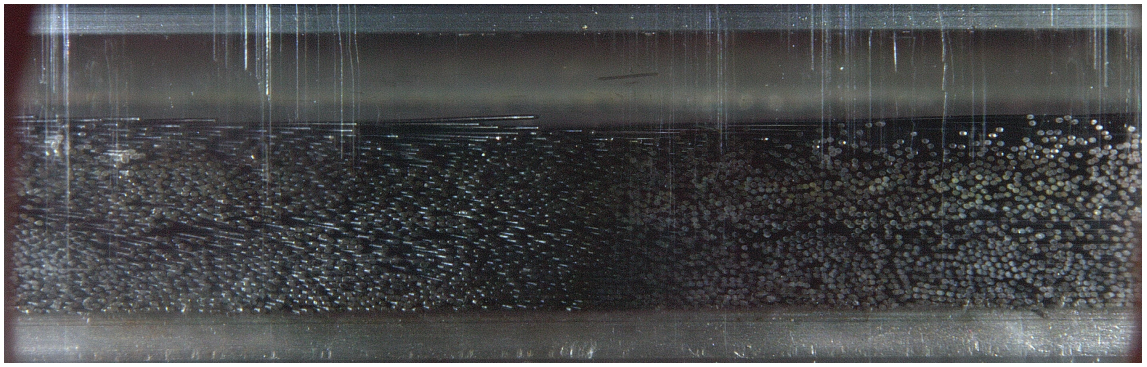


Abbildung 5.3: Beispielbild Borstenpaket BP-5

Die Bilder der nicht inklinierten Borsten (BP-0-Bildersammlung) zeigen größtenteils ein gleichmäßiges Auflicht. Im linken Teilbereich ist das Auflicht recht stark. Die Bürstendichtungen unterlagen in den vorliegenden Betriebspunkten wegen der fehlenden Anstellung teils Oszillationen, wie sie auch in Abbildung 5.4 erkennbar sind und eine Erkennung einzelner Borsten erschweren. In einigen Bildern oszilliert der Großteil der Borsten, wohingegen andere Bereiche einen geringen Einfluss zeigen. Bei nicht oszillierenden Borsten ist eine Erkennung gut möglich.

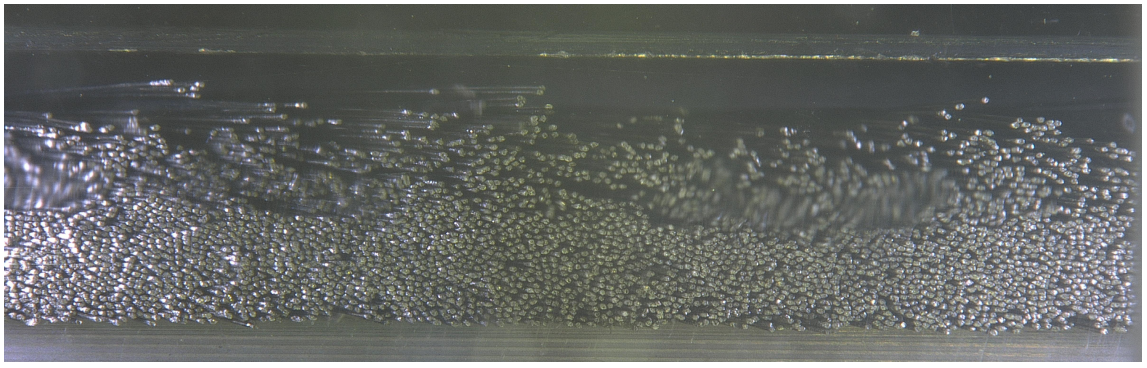
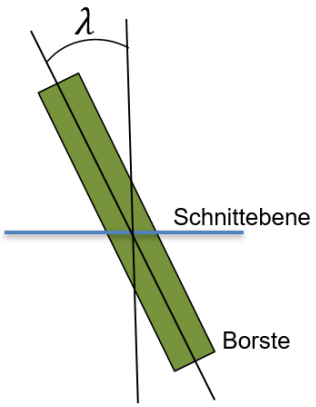


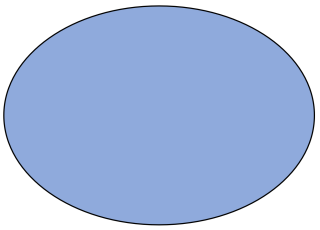
Abbildung 5.4: Beispielbild Borstenpaket BP-0

5.2.2 Borstenform der Dichtungen

Die theoretische Form der Borstenspitzen nach der Fertigung ist kreisförmig. Durch das Einlaufen der Bürstendichtungen beim Betrieb oder Blow-Down ist Kontakt mit dem Rotor möglich. Durch den Legewinkel λ von 45° stellt sich hierbei an den Spitzen der Borsten theoretisch eine elliptische Geometrie ein. Dies ist in Abbildung 5.5 zu sehen.



(a) Schnitt der Borste mit Legewinkel λ



(b) Resultierende Ellipsengeometrie

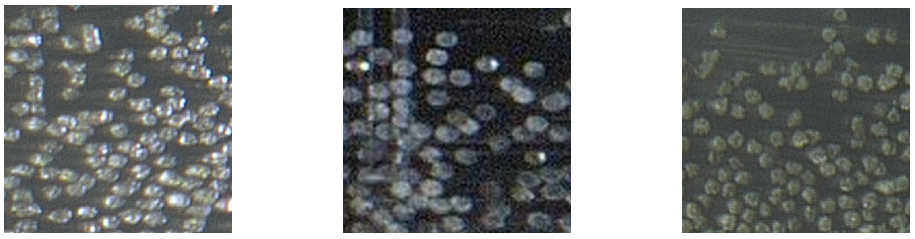
Abbildung 5.5: Schematische Darstellung des Borstenschnitts

Die große Halbachse a und kleine Halbachse b der Ellipsen lassen sich dabei folgendermaßen berechnen:

$$a = \frac{1}{2} \cdot \frac{D}{sin(\lambda)} = 0,0495mm \tag{5.1}$$

$$b = \frac{1}{2} \cdot D = 0,035mm \tag{5.2}$$

Die Spitzen der Borsten in den vorliegenden Bildern der unterschiedlichen Varianten der Bürstendichtung, siehe Abbildung 5.6, zeigen kreis- und ellipsenähnliche Formen. Die Form der Borsten ist aber auch teils unregelmäßig und voneinander verschieden. Denkbare Einflüsse auf die Form sind sehr vielseitig. Unterschiede im Legewinkel oder das Anleuchten einer Borste von der Seite können ursächlich sein. Zudem sind Verschleißerscheinungen nicht auszuschließen, z.B. in Folge des Blow-Down-Effekts (vgl. Kapitel 2.3) und eines Abtrags der Borsten. Ebenso können einzelne Borsten sich stark annähern oder überlappen. Eine visuelle Trennung dieser Borsten-Cluster ist teils sehr schwierig. Neben einer tatsächlichen Annäherung der Borsten ist auch ein Einfluss durch Messunsicherheiten der Kamera, z.B. bei bewegten Borsten in Folge geringer Stabilität durch Oszillationen denkbar.

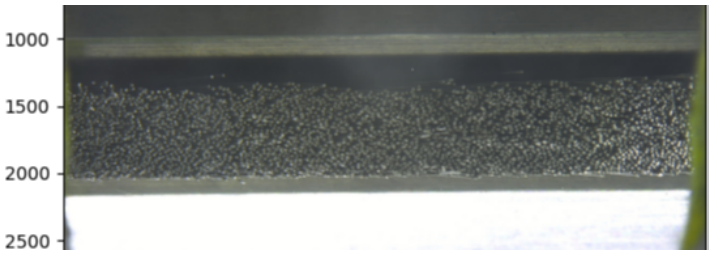


(a) BP-0 (b) BP-5 (c) BP-8

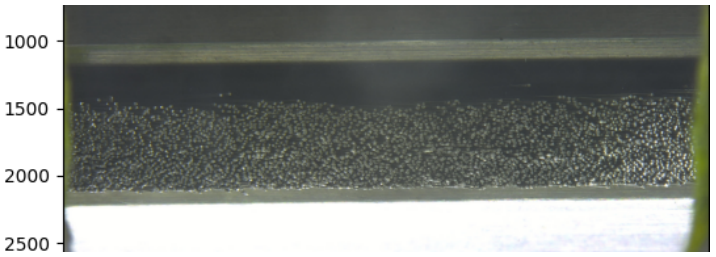
Abbildung 5.6: Form der Borsten in den Bildersammlungen

5.2.3 Betriebszustände der Dichtungen

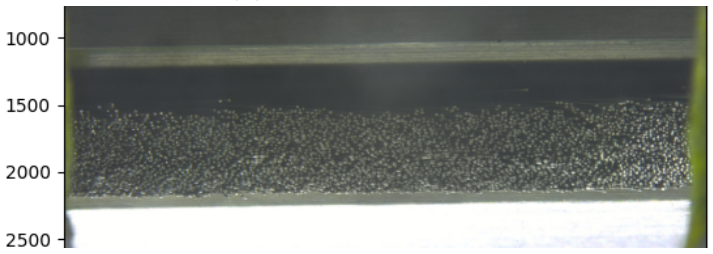
Je Bürstendichtungsvariante wurden vier bis zu sieben Betriebszustände, die sich jeweils um eine Druckdifferenz von $\Delta p = 1\text{bar}$ unterscheiden, fotografiert (Schwarz et al., 2014). Mit zunehmender Bedruckung ist eine Wanderung nach unten, sowohl des Borstenpakets, als auch des Deck- und Stützrings wahrzunehmen. Abbildung 5.7 zeigt den nullten (unbedruckten), den dritten ($\Delta p = 3\text{bar}$) und sechsten Betriebszustand ($\Delta p = 6\text{bar}$) der BP-8-Dichtung.



(a) BP-8 $\Delta p = 0\text{bar}$



(b) BP-8 $\Delta p = 3\text{bar}$



(c) BP-8 $\Delta p = 6\text{bar}$

Abbildung 5.7: Betriebszustände der Bürstendichtung: Mit zunehmender Bedruckung ist eine Verdichtung des Borstenpakets und eine leichte Wanderung des Deckrings (oben) und Stützrings (unten) festzustellen.

5.3 Wahl der Dichtungsvariante

Die BP-0 Bildersammlung wird nicht weiter betrachtet, da die Bürstendichtung im bedruckten Zustand instabil ist, und in vielen Bereichen des Bildes oszilliert. Zum einen hat die Borstenposition im abgebildeten Zustand hierdurch wenig Aussagekraft und zum anderen ist auch ein Verwischen der Borsten bis hin zur vollständigen Unkenntlichmachung einzelner Borsten in den Bildern möglich. Die angestellten Bürstendichtungen (BP-5, BP-8) haben diesbezüglich gutmütigere Eigenschaften. Dabei ist die BP5-Bildersammlung allerdings von vielen Artefakten durchzogen, und weist sehr inhomogene Beleuchtungseigenschaften auf, die mit klassischen Computer-Vision-Algorithmen sehr aufwändig zu korrigieren sind, und ein reproduzierbares Ergebnis über alle Bilder hinweg erschweren. Daher wird im Rahmen dieser Arbeit das BP-8-Datenset mit seinen sieben Betriebszuständen (Δp = 0-6bar) verwendet und analysiert. Der Vergleich der Borstenanordnung zwischen BP-5 und BP-8 wird ausgesetzt.

5.4 Anforderungen an das Erkennungssystem

Aus den Rahmenbedingungen der vorliegenden Bilder leiten sich die Anforderungen an das Erkennungssystem in Tabelle 5.3 ab, deren Erfüllung im Rahmen der Evaluation diskutiert wird.

Nr.	Anforderung	Beschreibung
1	Quantität	In einem gegebenen Bild soll die gleiche Menge von Borsten erkannt werden, die ein menschlicher Betrachter sehen würde, trotz unterschiedlicher Borstenformen sowie Cluster.
2	Genauigkeit	Die ermittelte Kontur soll möglichst mit der tatsächlichen Kontur übereinstimmen.
3	Reproduzierbarkeit	Die Erkennung soll trotz lokal verschiedener Lichtverhältnisse und Störungen, wie der Wanderung des Borstenpakets, funktionieren.

Tabelle 5.3: Anforderungen des Erkennungssystems

6. Bestimmung der Borstenanordnung

Der Borstenerkennungs-Algorithmus gliedert sich in drei Teile. So wird zunächst die *Region of Interest* (ROI) im Bild bestimmt, also der Bereich des Borstenpakets lokalisiert. Der gebildete Ausschnitt wird dann der eigentlichen Borstenerkennung unterzogen. Etwaige Cluster-Bereiche werden mittels der Watershed Segmentation gelöst. Der Ablauf der Bildverarbeitung ist in Abbildung 6.1 abgebildet, und wird nun tiefergelegt.

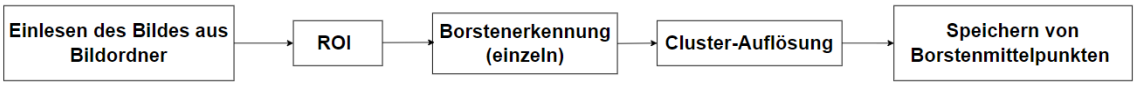


Abbildung 6.1: Flowchart des Borstenerkennungsalgorithmus: Das Verarbeitung der Bilder erfolgt, bis jedes Bildes im Quellordner (des jeweiligen Betriebszustandes) verarbeitet wurde.

Durch das Verfahren werden die insgesamt 168 Bilder (24 Bilder, 7 Betriebszustände) automatisiert verarbeitet. Die gespeicherten Borstenmittelpunkte nach Ablauf der Borstenerkennung dienen als Grundlage zur Analyse und Vorhersage der Borstenanordnung in Kapitel 7.

6.1 Region of Interest

Durch das Erkennen der Region of Interest lässt sich das Risiko von Falscherkennungen durch ähnliche Muster in den vorliegenden Bildern verringern und der Rechenaufwand der nachfolgenden Operationen durch die verringerte Bildgröße reduzieren. Hierzu wird die y-Koordinate ermittelt, an welchem der festgelegte Ausschnitt des Bildes erfolgt. Ein in den Bildern markantes und wiederkehrendes Merkmal sind Kanten, insbesondere des Stützrings. Abbildung 6.2 zeigt das Schema, nach welchem der Stützring und damit die Lage des Borstenpakets bestimmt wird.

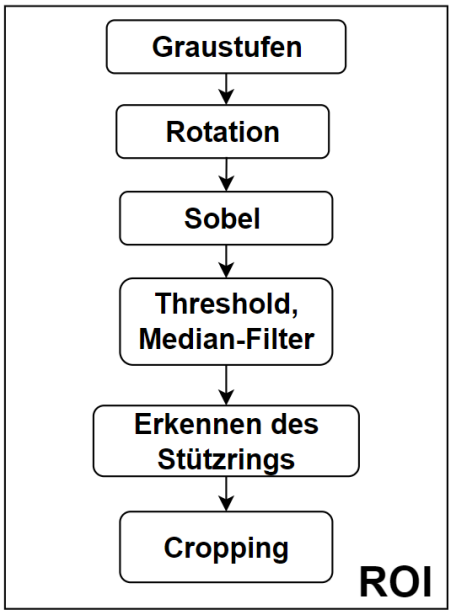
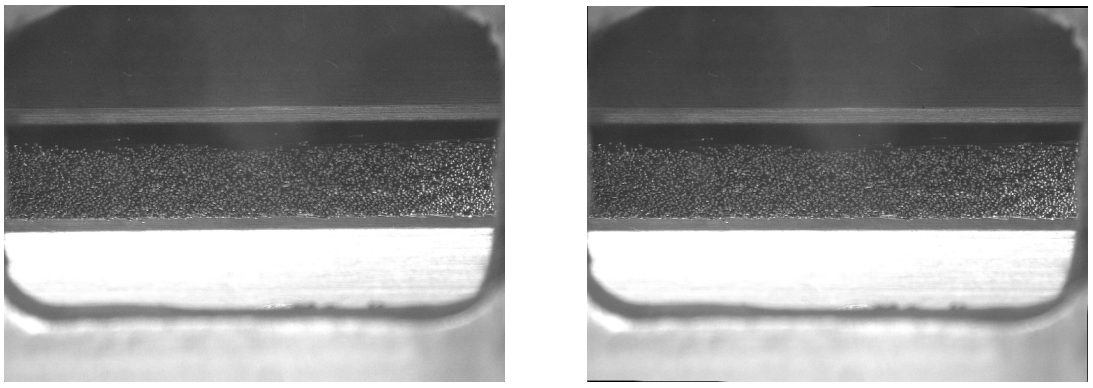


Abbildung 6.2: Flowchart der ROI-Findung

Zunächst wird das Bild in Graustufen konvertiert, um die nachfolgenden Operationen zu erleichtern, indem sie auf einen Kanal reduziert werden. Anschließend wird das leicht schiefe Bild unter Anwendung einer Rotationsmatrix (vgl. Kapitel 4.2.12) um seinen Mittelpunkt rotiert, sodass die Kanten des Decks- und Stützrings horizontal orientiert sind und die Koordinaten der Borsten nicht durch die Schiefe verfälscht werden. Die Rotation zum Ausgleich der Schiefe erfolgt mit einem Winkel von $-0,5°$, der über alle Bilder hinweg konsistent ist.



(a) Graustufen-Bild (b) Bild nach Rotation

Abbildung 6.3: Leichte Rotation des Bildes zur Ausrichtung des Borstenpakets

Um die horizontal orientierte Kante des Stützrings zu finden, wird der Gradient in y-Richtung berechnet. Dazu wird ein Sobel-Filter (vgl. Kapitel 4.2.2) mit einer 3×3 Faltungsmatrix G_y angewandt. Die Gradienten werden anschließend in den Betrag konvertiert. Abbildung 6.4 zeigt das Ergebnis.

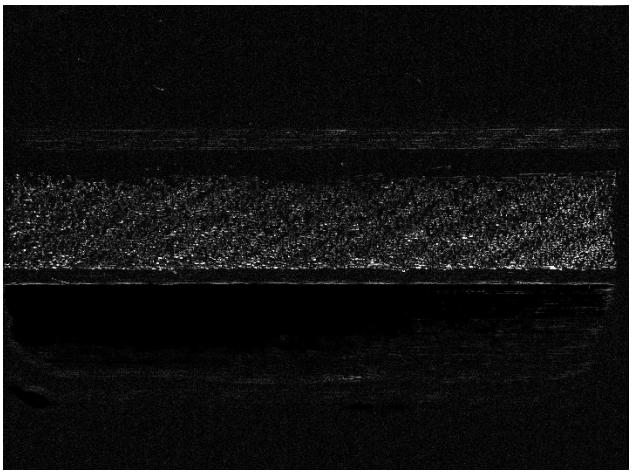
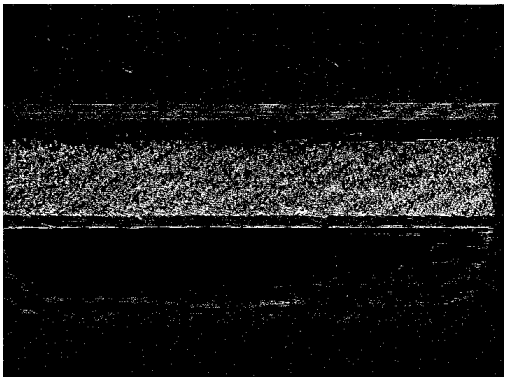
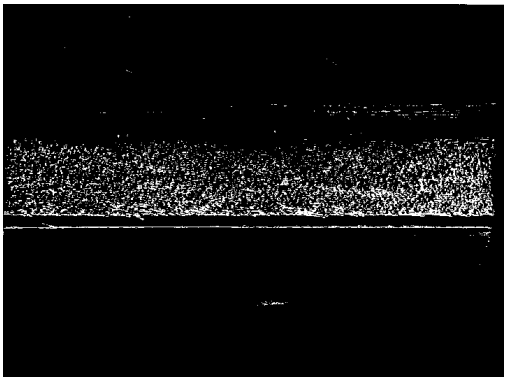


Abbildung 6.4: Bild nach Anwendung des Sobel-Filters

Anhand dieses Gradientenbilds wird nun ein Schwellenwertverfahren (vgl. Kapitel 4.2.5) angewendet. Dieses soll einen Teil der Pixel in dem Gradientenbild von der Linienerkennung ausschließen, z.B. die schwache Kante des Spiegels unterhalb des Stützrings oder einzelne schwache Rauschpixel. Die stärkste Kante ist die des Stützrings, daher wird der Schwellenwert so ermittelt, dass diese definitiv sichtbar ist. Nach mehrfacher Überprüfung unterschiedlicher Schwellenwerte wird er so festgelegt, dass er 15% des Maximalwerts des Kantenbilds entspricht. Zusätzlich wird ein Median-Filter (vgl. Kapitel 4.2.3) angewendet, um einzelne Rauschpixel zu filtern und die Linie des Stützrings zu glätten. Der Vergleich des Binärbilds ohne und mit zusätzlicher Anwendung des Median-Filters ist in Abbildung 6.5 zu sehen.



(a) Schwellenwertverfahren an Sobel-Bild



(b) Schwellenwertverfahren mit zusätzlicher Anwendung eines Median-Filters

Abbildung 6.5: Erzeugtes Binärbild nach der Sobel-Filterung

Anhand dieses Bildes wird nun die Hough-Linienerkennung (vgl. Kapitel 4.2.10) durchgeführt. Hierzu wird der Parameter der Mindest-Zahl von weißen Pixeln, zu der Hälfte der Bildbreite festgelegt. Da eine Vielzahl von Linien erkannt wird, etwa auch solche, die durch das Borstenpaket verlaufen, wird diejenige Linie verwendet, die den größten Abstand r von dem Ursprung des Bildes hat und deren Steigungswinkel sich zusätzlich innerhalb einer Toleranz von $-0,25°$ bis $0,25°$ (zur Berücksichtigung etwaiger Fertigungstoleranzen des Stützrings) befindet.
Abbildung 6.6 zeigt das Ergebnis dieser Linienerkennung.

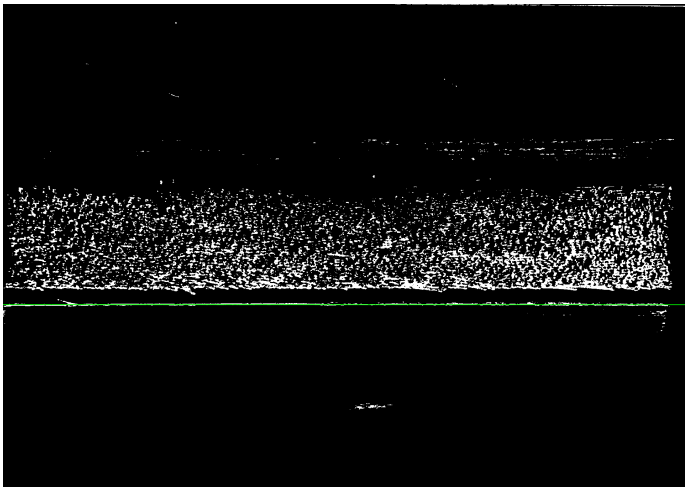


Abbildung 6.6: Erkannte Linie (grün) mit der Hough-Linienerkennung

Anschließend erfolgt das Cropping. Das Mittel der y-Koordinaten der ermittelten Linie wird als Referenzpunkt für die Erzeugung des Ausschnitts verwendet. Dieses erfolgt mit festgelegten Pixelabständen von dieser Koordinate, um den Stützring auszuschließen, sodass der Hintergrund der Borsten in etwa homogen ist. Zusätzlich werden hierbei noch die verblassten Ränder des Spiegels entfernt, in denen die Borsten nicht zu sehen sind (vgl. Abbildung 5.2). Deshalb wird die Breite des Ausschnitts zu nur 90% der ursprünglichen Breite festgelegt. Der resultierende Bildauschnitt ist in Abbildung 6.7 zu sehen.

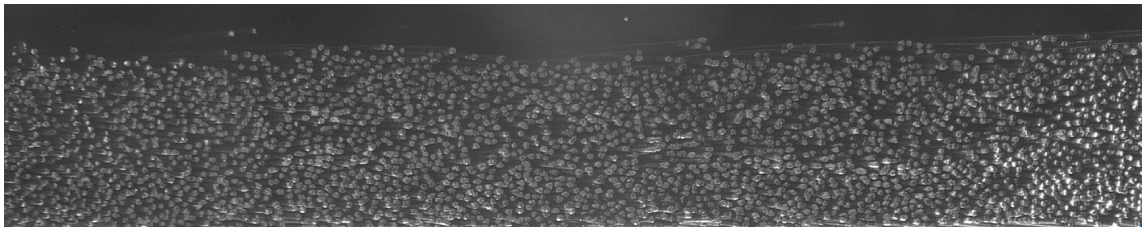


Abbildung 6.7: Resultierender Aussschnitt nach ROI-Verarbeitung

Da das Borstenpaket und der Stützring bei Bedruckung wandert und seine Lage auch innerhalb eines Betriebszustandes zwischen den Bildern variiert, wird dieser Prozess bildweise durchlaufen. Damit soll sichergestellt werden, dass die statistisch relevanten Koordinatensysteme der jeweiligen Bilder ineinander überführbar sind.

6.2 Borstenerkennung

Die eigentliche Borstenerkennung erfolgt zweistufig. Zunächst werden die Strukturen ermittelt, die aller Wahrscheinlichkeit nach einzelne Borsten darstellen. Anschließend werden Cluster-Bereiche innerhalb der erkannten Strukturen ermittelt, um ein Verfahren anzuwenden, welches diese Cluster auflösen kann und so wieder einzelne Borsten liefert. Kann auch dieses Verfahren nicht einzelne Borsten erkennen, so erfolgt eine Schätzung der Borstenzahl anhand der Größe des Clusters.

6.2.1 Erkennung einzelner Borsten

Abbildung 6.8 zeigt den Verlauf des Programms zur Erkennung einzelner Borsten.

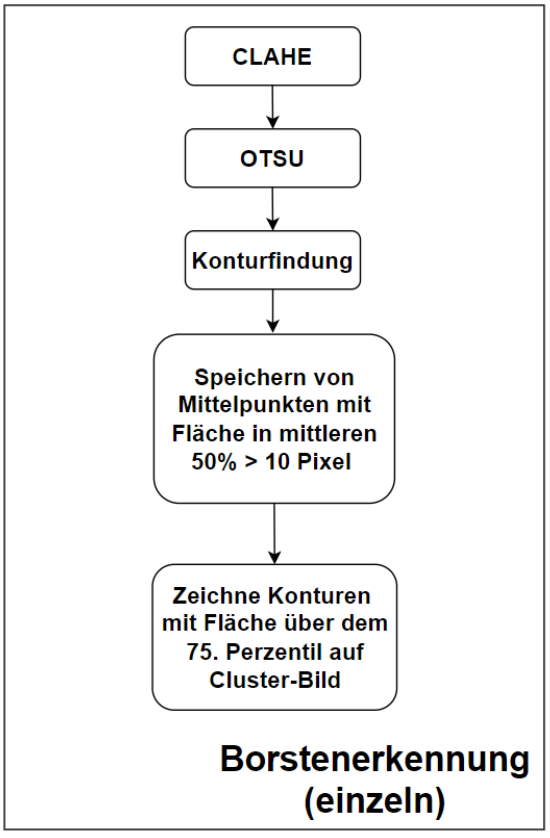
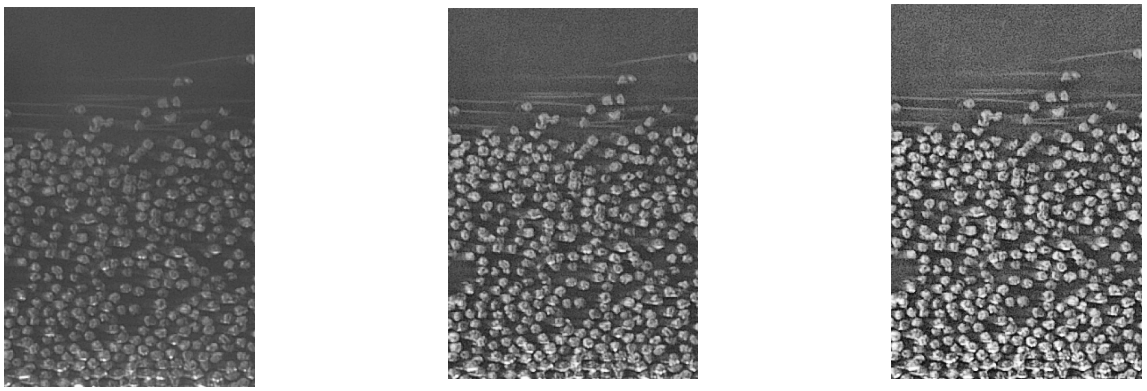


Abbildung 6.8: Ablauf der Erkennung einzelner Borsten

Die nach der ROI-Findung ermittelten Bildausschnitte werden einem CLAHE-Verfahren (vgl. Kapitel 4.2.4) unterzogen. Dies soll den Kontrast in dem Bild verbessern, wodurch sich die Borsten besser gegenüber dem Hintergrund hervorheben. Zusätzlich ist es dadurch möglich innerhalb des Bildes inhomogen ausgeleuchtete Bereiche aneinander anzugleichen. Hierbei können zwei Parameter angepasst werden: Das *ClipLimit*, also die maximale Kontrastverstärkung, und die *TileSize*, also die Größe des für die Berechnung der lokalen Ausgleichsfunktion betrachteten Bereichs. Beide Parameter sollten optimalerweise so gewählt werden, dass das Rauschen oder Artefakte

innerhalb des Bildes kaum verstärkt werden. Die Findung eines optimalen Parametersatzes ist nicht einfach möglich und erfolgt in der Praxis häufig auf heuristische Weise oder unter Anwendung von Machine Learning (Campos et al., 2019). Abbildung 6.9 zeigt das Ergebnis der Verarbeitung eines zufällig gewählten Teilstücks des Borstenpakets bei unterschiedlichen *ClipLimits* und einer *TileSize* von 32×32.



(a) ohne CLAHE

(b) CLAHE mit $cL = 2.0$

(c) CLAHE mit $cL = 4.0$

Abbildung 6.9: Verarbeitung mit unterschiedlichen *ClipLimit* Parametern

Im Nachfolgenden wird eine *TileSize* von (32,32) verwendet, weil diese für den vorliegenden Ausschnitt bereits ein guter Ausgleich zwischen einzelnen Borsten und insgesamt für das Teilstück feststellbar ist. Abbildung 6.9 zeigt bei einem *ClipLimit* von 4,0 zunehmende Rauscherscheinungen im homogenen Hintergrund, allerdings auch eine gute Hervorhebung der Borsten. Daher wird ein *ClipLimit* von 3,0 verwendet, um hierbei einen Kompromiss einzugehen.

Anschließend wird der OTSU-Algorithmus (vgl. Kapitel 4.2.6) angewendet. Dieser liefert einen für das Bild gültigen, globalen Schwellenwert und erzeugt so Binärbilder anhand derer die Erkennung der Borsten erfolgt. Ein solches Binärbild ist in Abbildung 6.10 dargestellt.

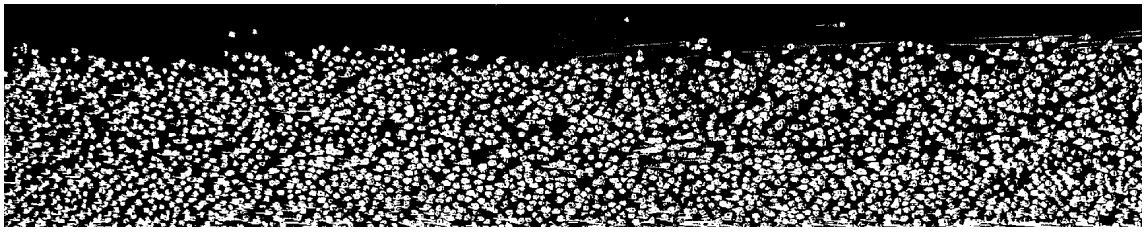


Abbildung 6.10: Binärbild nach Anwendung des OTSU-Algorithmus

Anhand des Binärbilds können nun die Konturen der Borsten ermittelt werden. Hierzu wird der OpenCV-Algorithmus *cv2.findcontours* verwendet, der die Konturen um die weißen Pixel erkennt. Der Algorithmus basiert auf der Arbeit von Suzuki und Abe (1985) zur Grenzfindung von Connected Components (vgl. Kapitel 4.2.11), welcher im Rahmen dieser Arbeit nicht weiter behandelt wird. Das Argument *Countour*

Retrieval Mode wird so gesetzt, dass nur die äußersten Konturen eines Objektes ermittelt werden. Anhand des Merkmals Fläche, also der Zahl der Pixel innerhalb der Konturen, lassen sich gut kleine Rauschpixel entfernen, indem ein Schwellenwert festgelegt wird, ab welchem eine erkannte Kontur als solche markiert und gespeichert wird. Gleichzeitig stellt es ein Indiz für das Vorhandensein eines Clusters, also überlagerter Borsten dar. Einflüsse wie Verschleiß, Fertigungstoleranzen, Strömungseffekte (z.B. Blow-Down) und Unterschiede in der Beleuchtung einer Borste erschweren die Festlegung eines eindeutigen, auf der theoretischen Ellipsengeometrie (vgl. Kapitel 5.2.2) basierenden Schwellenwerts. Daher wird eine Kontur als zulässig bewertet, sofern ihre Fläche sich zwischen dem 25. und 75. Perzentil aller Flächen über zehn Pixeln befindet. Die gefundenen Konturen werden grün markiert. Gleichzeitig wird der Schwerpunkt jeder Kontur ermittelt und gespeichert. Abbildung 6.11 zeigt das Ergebnis der Konturfindung eines Teilstücks des Binärbilds aus Abbildung 6.10.

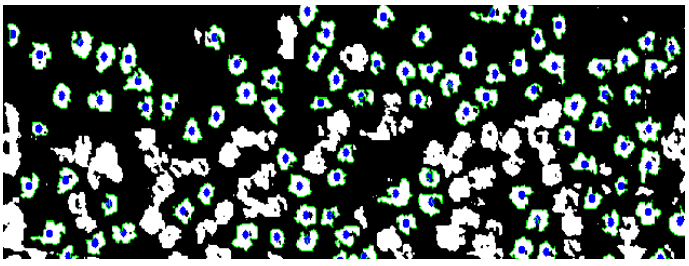


Abbildung 6.11: Ergebnis der Konturfindung: Einzelne Bosten werden grün markiert. Die nicht markierten Borsten entsprechen Borstenclustern, die weiterverarbeitet werden, oder Rauschpixeln, die eine zu kleine Fläche haben.

Alle Konturen, deren Fläche über dem 75. Perzentil der Konturen über zehn Pixeln liegt, werden als Cluster zur weiteren Verarbeitung beibehalten, indem das Innere dieser Konturen in einem neuen Binärbild weiß markiert wird. Abbildung 6.12 zeigt die Cluster-Kandidaten des Binärbildes aus Abbildung 6.10, die anschließend weiterverarbeitet werden.

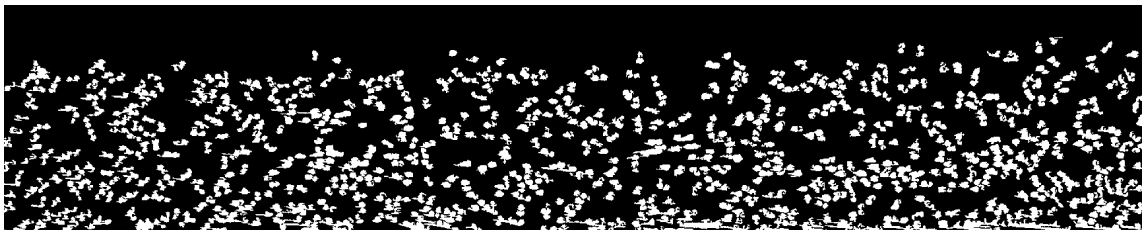


Abbildung 6.12: Cluster-Bild

6.2.2 Verarbeitung von Borsten in Cluster-Bereichen

Abbildung 6.13 zeigt den Ablauf der Trennung von Borsten in Cluster-Bereichen.

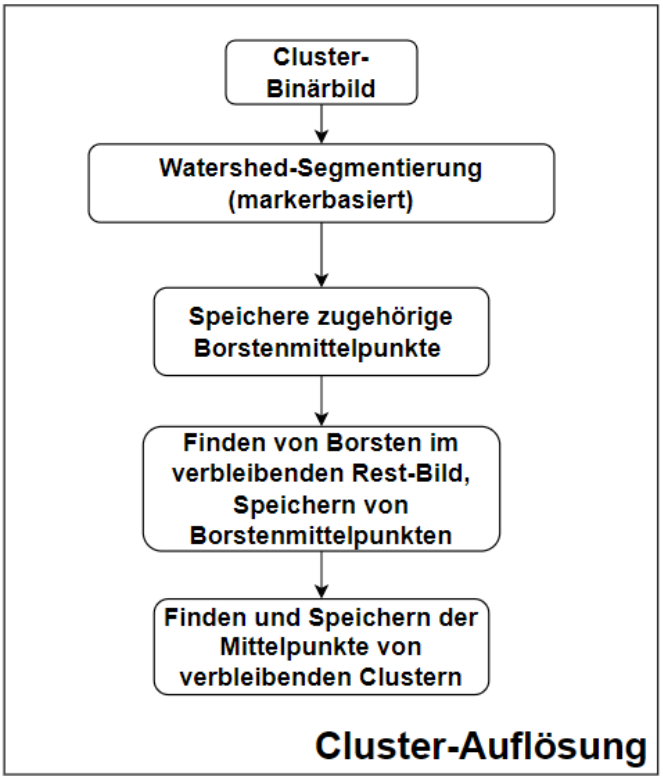


Abbildung 6.13: Flowchart des Verfahrens zur Cluster-Auflösung

Die Borsten-Cluster des Cluster-Bildes (vgl. Abbildung 6.12) werden mittels des markerbasierten Watershed-Segmentierungsverfahrens getrennt, sodass die einzelnen Borsten in Clustern bestimmt werden können. Das Verfahren erfolgt, wie es in Kapitel 4.2.9 geschildert ist, wobei für dieses Verfahren der Ausschnitt (vgl. Abbildung 6.7) in Farbe übermittelt wird. Zusätzlich werden die Parameter des Schwellenwerts nach der Distanztransformation (vgl. Kapitel 4.2.8) und die Größe des Strukturelements bei der Dilation (vgl. Kapitel 4.2.7) für jedes Cluster-Bild variiert, um diejenige Kombination von Parametern zu finden, welche die Zahl der Borsten im Bereich des in Kapitel 6.2.1 bestimmten Flächenbereichs zur Erkennung einzelner Borsten maximiert. Die untersuchten Größen des elliptischen Dilations-Strukturelements sind jeweils (3x3, 5x5, 7x7) und es werden 40 äquidistante Schwellenwerte im Intervall von [0,255] evaluiert. Ein beispielhaftes Ergebnis nach Durchlauf des Verfahrens ist in Abbildung 6.14 zu sehen. Die Trennung der zusammenhängenden Cluster liefert teils kleine Segmente. Der Großteil der Borsten kann allerdings plausibel getrennt werden. Einige Cluster werden nicht getrennt und verbleiben in ihrer ursprünglicher Form. Teils verbleiben nicht segmentierte Reste, die durch fehlende Marker an den relevanten Stellen erklärt werden können.

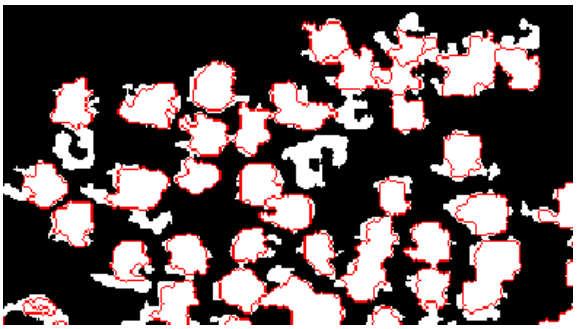


Abbildung 6.14: Ergebnis der Watershed Segmentation in einem Teilbereich des Cluster-Bildes: Die roten Konturen repräsentieren die Grenzen der jeweiligen Segmente. Teils verbleiben nicht segmentierte Reste.

Die gefundenen Segmente werden der Konturfindung aus Kapitel 6.2.1 unterzogen, um wieder einzelne Borsten gemäß der Anforderungen an die Fläche zu erkennen und deren Mittelpunkt zu speichern.

Ebenso werden verbleibende Reste anhand des Rest-Bildes, welches die Differenz aus dem Original- und segmentierten Bild darstellt, siehe Abbildung 6.15, der Konturfindung nach Kapitel 6.2.1 unterzogen, um nicht segmentierte Bereiche auch bei der Erkennung zu berücksichtigen.

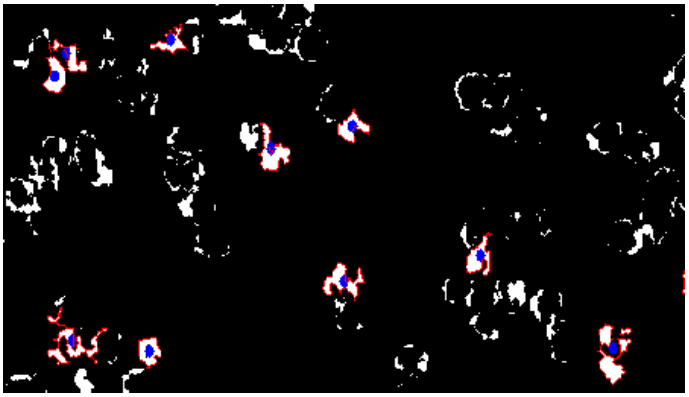
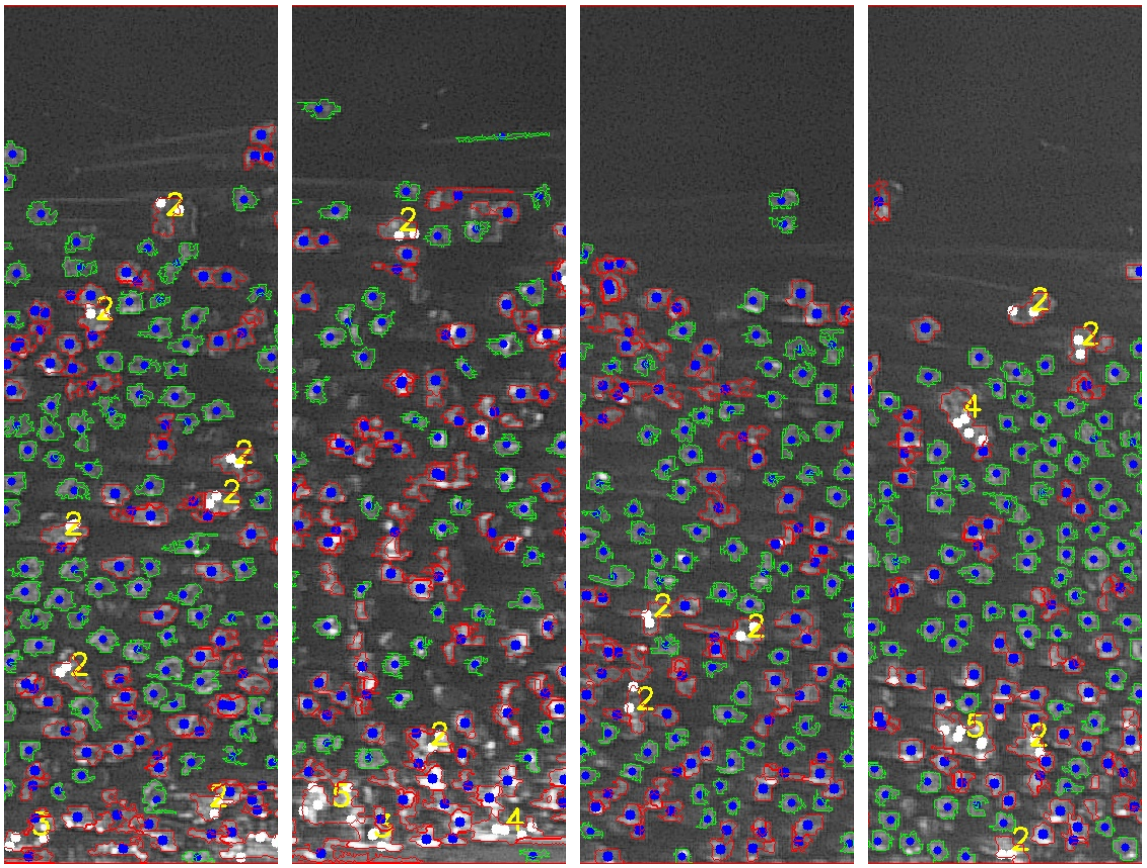


Abbildung 6.15: Markierte Konturen des Rest-Bildes

Verbliebene Cluster nach der Watershed-Segmentierung werden mittels eines Schätzverfahrens verarbeitet. Hierbei wird die jeweilige Konturfläche durch den Median der Flächen bisher gefundener Konturen geteilt und das Ergebnis auf eine ganze Zahl gerundet. Innerhalb des Clusters wird dann eine Zahl von Mittelpunkten entsprechend der gerundeten Zahl zufällig verteilt. So sollen Cluster-Bereiche ihre praktische Bedeutung als Bereiche nahe liegender Punkte in der nachfolgenden Statistik nicht verlieren.

6.3 Evaluation

Jedes Bild wird nach der Erkennung in sechszehn Teile geteilt und in einem Ordner abgelegt, um mit dem Auge bewertet werden zu können. Abbildung 6.16 zeigt vier zufällig gewählte Ausschnitte, anhand derer die Evaluation durchgeführt wird. Grün markierte Borsten werden mit dem in Kapitel 6.2.1 vorgestellten Verfahren zur Erkennung einzelner Borsten ermittelt. Rot markierte Borsten entstammen dem Verfahren aus Kapitel 6.2.2 zur Lösung von Cluster-Bereichen. Verbliebene Cluster sind in den Bildern mit weißen Mittelpunkten und der daneben stehenden gelben Ziffer (Zahl geschätzter Borsten) gekennzeichnet.



(a) Bild 1 (b) Bild 2 (c) Bild 3 (d) Bild 4

Abbildung 6.16: Bilder zur Evaluation der Borstenerkennung: Grün mit blauem Mittelpunkt: Erkannte einzelne Borsten, Rot mit blauem Mittelpunkt: Getrennte Cluster, Rot mit weißem Mittelpunkt und danebenstehender Ziffer: Zufällig verteilte Mittelpunkte in verbleibenden Clustern

Die Leistungsfähigkeit des Erkennungssystems wird anhand der Metriken Precision, Recall und F_1-Score quantifiziert. Dem vorliegenden Erkennungssystem liegt kein pixelgenaues Label (Hintergrund, Borste) zu Grunde. Deswegen wird das Erkennen der zusammenhängenden Regionen, also der einzelnen Borsten, bewertet. Die Definitionen der Metriken lauten jeweils:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6.1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6.2}$$

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.3}$$

Dabei gilt:

- TP (True Positive): Borste wird erkannt wo sie sich auch tatsächlich befindet.
- FP (False Postitive): Borste wird erkannt, befindet sich aber nicht auf dem Bild.
- FN (False Negative): Borste ist auf dem Bild, wird aber nicht erkannt.

TN (True Negative), d.h. das richtige Erkennen des Hintergrunds, kann bei der objektbezogenen Betrachtungsweise nicht eindeutig quantifiziert werden. Die Ergebnisse der vier Ausschnitte aus Abbildung 6.16 sind in Tabelle 6.1 zu sehen.

Bild Nr.	**1**	**2**	**3**	**4**
TP (True Positive)	122	94	110	124
FP (False Positive)	4	4	5	5
FN (False Negative)	5	15	13	16
Recall	96,1%	86,2%	89,4%	88,6%
Precision	96,8%	95,9%	95,7%	96,1%
F_1-Score	96,4%	90,8%	92,4%	92,2%

Tabelle 6.1: Evaluation der Borstenerkennung

Dabei ist der Beitrag der geschätzten und zufällig verteilten Borsten in Cluster-Bereichen in Tabelle 6.2 zu sehen:

Bild Nr.	**1**	**2**	**3**	**4**
Geschätzte Borsten	17	16	6	17
Davon FP	3	5	0	2
Davon FN	0	0	1	2
Anteil geschätzter Borsten	11%	12%	4%	10%

Tabelle 6.2: Evaluation in Bereichen geschätzer Borsten

Die Erfüllung der Anforderungen aus Tabelle 5.3 wird nun anhand der Bilder in Abbildung 6.16 diskutiert. Die Zahl erkannter Borsten (Anforderung: Quantität)

ist vergleichbar mit der tatsächlichen Zahl vorliegender Borsten. Der F_1-Score, das harmonische Mittel aus Precision und Recall, ist über die Bilder hinweg vergleichbar, mit einer größeren Abweichung bei Bild Nummer 1. Die Precision ist hierbei stets höher als das Recall, was durch eine Tendenz zu falsch negativen statt falsch positiven Erkennungen zu erklären ist (siehe Tabelle 6.1). Die Anforderung der Genauigkeit (genauer Konturverlauf) ist in einigen Bereichen, jedoch nicht in allen erfüllt. Insbesondere die rot markierten Konturen, die dem Cluster-Lösungsverfahren entstammen, sind teils ungenau, dafür aber im Hinblick auf Quantität insgesamt nachvollziehbar. Die Schätzung von Borsten in verbleibenden Clustern und zufällige Verteilung der Mittelpunkte ist in diesem Kontext als Genauigkeitsverlust zu bewerten. Dabei ist die Schätzung der Menge vorliegender Borsten als solche plausibel, und die Menge falsch positiver oder falsch negativer Erkennungen sowie der Anteil geschätzter Borsten an insgesamt erkannten Borsten minderheitlich (vgl. Tabelle 6.2). Hinsichtlich der Anforderung der Reproduzierbarkeit sind keine eindeutigen Bereiche schlechter Erkennung innerhalb des Bildes feststellbar. Auch zwischen den Bildern sind die Ergebnisse vergleichbar. Eines der Bilder zeigt einen Teil des Stützrings, was durch eine Ungenauigkeit bei der Erkennung des Stützrings aus Kapitel 6.1 oder einen fertigungsbedingten Unterschied erklärbar ist.

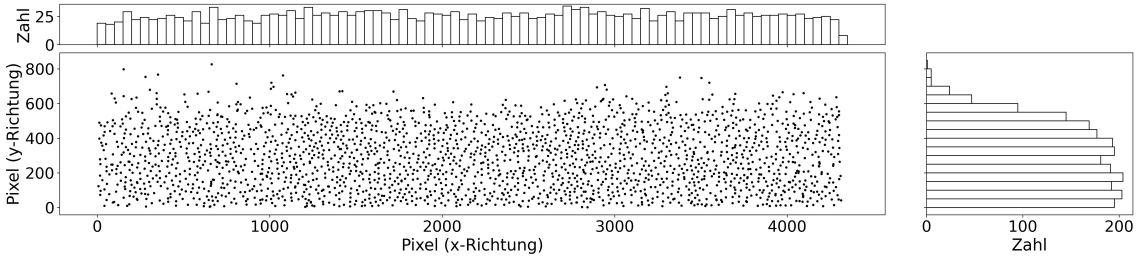
Die Ergebnisse dieser Evaluation unterliegen insgesamt einem subjektiven Einfluss, da in Teilbereichen die Borstenspitzen hinsichtlich ihrer Form Unregelmäßigkeiten aufweisen, die eine objektive Aussage über die tatsächliche Zahl vorliegender Borsten erschweren.

7. Beschreibung und Vorhersage der Borstenanordnung

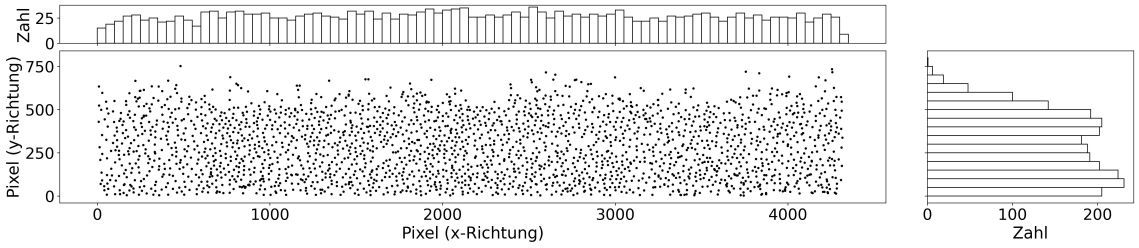
Das Resultat der Erkennung soll nun verarbeitet werden, um eine Beschreibung und Schlussfolgerungen über die Bürstendichtungsanordnung zu ermöglichen. Im Anschluss erfolgt die Erklärung des Programms zur Simulation von Borstenanordnungen anhand der ermittelten Verteilungen.

7.1 Statistische Eigenschaften der Borstenanordnung

Die statistische Modellierung eines Borstenpakets erfolgt anhand der Verteilungen von x- und y-Koordinaten der Mittelpunkte erkannter Borsten. Abbildung 7.1 zeigt exemplarisch zwei Punktwolken nach der Erkennung sowie die zugehörigen Histogramme der x- und y-Koordinatenmittelpunkte einer Borstenanordnung. Der vorliegende Betriebszustand ($\Delta p = 1\text{bar}$) ist bei den Bildern der gleiche und sie sind in Umfangsrichtung voneinander versetzt.



(a) Anordnung 1

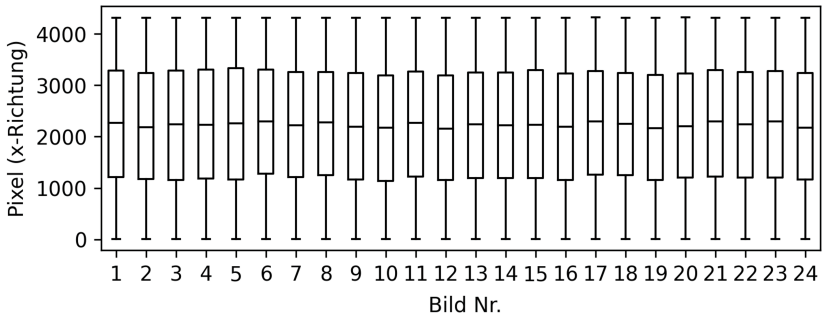


(b) Anordnung 2

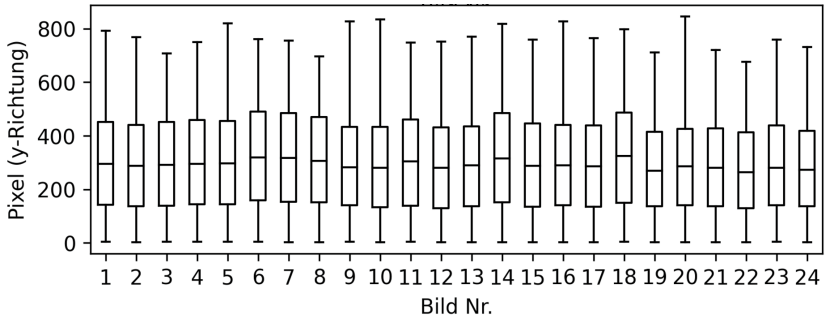
Abbildung 7.1: Scatter-Plots der Mittelpunktanordnungen zweier Bilder des Borstenpakets und zugehörige Histogramme: Die Klassenteilung beträgt jeweils 50 Pixel.

Innerhalb eines Betriebszustands liegen jeweils 24 Bilder und damit unterschiedliche Stichproben vor die das Borstenpaket des runden Bauteils repräsentieren. Abbildung

7.2 zeigt exemplarisch für einen Betriebszustand von $\Delta p = 1$ bar die Box-Plots der x- und y-Koordinaten von unterschiedlichen Bildern dieses Betriebszustands. Hierbei fallen insbesondere in y-Richtung Unterschiede im Hinblick auf Lage des Medians, der Spannweite und des Interquartilabstands auf.



(a) Box-Plots der x-Koordinaten



(b) Boxplots der y-Koordinaten

Abbildung 7.2: Box-Plots der Mittelpunkte unterschiedlicher Bilder innerhalb eines Betriebszustands

Anhand von statistischen Tests sollen nun Schlussfolgerungen über die Verteilungen der x- und y-Koordinaten der Borstenanordnungen, innerhalb eines Betriebszustands gezogen werden. Das Signifikanzniveau wird dabei auf $\alpha = 0,05$ (vgl. Kapitel 4.3.1) festgelegt. Es wird angenommen, dass die in Umfangsrichtung versetzen Stichproben voneinander unabhängig sind.
Zunächst wird der Levene-Test (vgl. Kapitel 4.3.2) durchgeführt. Die Ergebnisse des Levene-Tests (siehe Tabelle 7.1) zeigen, dass die Nullhypothese homogener Varianzen bei allen Betriebszuständen in x-Richtung nicht verworfen wird. Jedoch wird die Nullhypothese bei den y-Koordinaten der Stichproben bei allen Betriebszuständen verworfen.

Betriebszustand	**0**	**1**	**2**	**3**	**4**	**5**	**6**
x-Richtung, p-Wert	$0,494$	$0,415$	$0,787$	$0,523$	$0,928$	$0,356$	$0,078$
y-Richtung, p-Wert	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0

Tabelle 7.1: Ergenisse des Levene-Tests

Der Mann-Whitney-U-Test (vgl. Kapitel 4.3.3) wurde ebenfalls für jeden Betriebszustand durchgeführt. Da dieser einen Zweistichprobentest darstellt, wird hierbei jede einzigartige Kombination von zwei unterschiedlichen Stichproben ausgewertet und auf den p-Wert überprüft. Danach wird der prozentuale Anteil der Tests mit $p \geq 0,05$ durch die Zahl aller durchgeführten Tests geteilt und als Prozent-Wert zurückgegeben. Die Ergebnisse dieses Verfahrens sind in Tabelle 7.2 dargestellt. Hierbei fällt auf, dass die Nullhypothese, insbesondere bei den Kombinationen der Stichproben von y-Koordinaten verworfen wird.

Betriebszustand	**0**	**1**	**2**	**3**	**4**	**5**	**6**
x-Richtung	96,38%	92,75%	86,23%	87,31%	90,58%	88,77%	88,04%
y-Richtung	54,35%	43,48%	48,19%	40,58%	48,91%	45,29%	42,39%

Tabelle 7.2: Ergebnisse des Mann-Whitney-U Tests

Ferner wird der Kolmogorov-Smirnov-Anpassungstest (vgl. Kapitel 4.3.4) durchgeführt, um zu untersuchen, ob die Mittelpunktverteilungen einzelner Bilder durch theoretische Verteilungsmodelle beschreibbar sein könnten. Hierzu werden eine Gleich-, Normal-, Weibull-, Gamma-, Lognorm- und Beta-Verteilung jeweils mittels Maximum Likelihood Estimation (vgl. Kapitel 4.3.5) an die Beobachtungen angepasst. Tabelle 7.3 und Tabelle 7.4 zeigen den Anteil der Tests einzelner Stichproben mit $p \geq 0,05$ für ein gegebenes Verteilungsmodell und Betriebszustand für die x- und y-Koordinaten.

Betriebszustand	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Gleich-Verteilung	50%	37,5%	37,5%	41,67%	41,67%	25%	37,5%
Normal-Verteilung	0	0	0	0	0	0	0
Weibull-Verteilung	0	0	0	0	0	0	0
Gamma-Verteilung	0	0	0	0	0	0	0
Lognorm-Verteilung	0	0	0	0	0	0	0
Beta-Verteilung	95,83%	91,67%	75%	87,5%	91,67%	91,67%	91,67%

Tabelle 7.3: Ergebnisse des Kolmogorov-Smirnov-Anpassungstests in x-Richtung

Betriebszustand	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Gleich-Verteilung	0	0	0	0	0	0	0
Normal-Verteilung	0	0	0	0	0	0	0
Weibull-Verteilung	0	0	0	0	0	0	0
Gamma-Verteilung	0	0	0	0	0	0	0
Lognorm-Verteilung	0	0	0	0	0	0	0
Beta-Verteilung	0	0	0	0	4,17%	4,17%	4,17%

Tabelle 7.4: Ergebnisse des Kolmogorov-Smirnov-Anpassungstests in y-Richtung

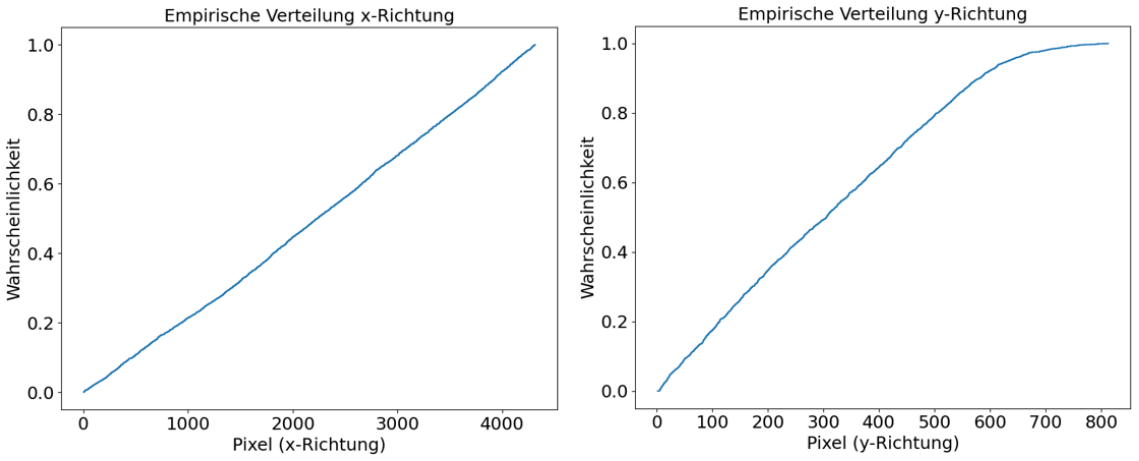
Die Nullhypothese einer Gleichverteilung, aber insbesondere der Beta-Verteilung in x-Richtung lässt sich in den meisten Fällen nicht verwerfen. In y-Richtung wird

die Nullhypothese nur bei einer geringen Zahl von Stichproben und für eine Beta-Verteilung beibehalten.
Für die nachfolgende Simulation wird daher auf die diskreten empirischen Verteilungen zugegriffen. Die durchschnittliche Borstenzahl eines Bildes für einen gegeben Betriebzustand ist in Tabelle 7.5 dargestellt.

Betriebszustand	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Durchschnitt	2417,8	2373,9	2234,3	2207,4	2184,4	2100,21	2075,4

Tabelle 7.5: Durchschnittlich erkannte Zahl von Borsten eines Bildes für einen gegebenen Betriebszustand

Die zunehmende Verkleinerung kann durch eine sukzessive Wanderung von Borsten über den von der Erkennung ausgeschlossenen Stützring, in Folge des zunehmenden Drucks erklärt werden. Ein Beispiel für die empirischen Verteilungen einer Stichprobe ist in Abbildung 7.3 zu sehen.



(a) Empirische Verteilung in x-Richtung (b) Empirische Verteilung in y-Richtung

Abbildung 7.3: Empirische Verteilungen einer Stichprobe

Alle ermittelten Verteilungen werden in einer PKL-Datei abgelegt, auf welche das nachfolgend erklärte Simulationsskript zugreifen kann.

Die zunehmende Bedruckung bei den Betriebszuständen und das physikalisch begründete Kompaktieren des Borstenpakets in y-Richtung (vgl. Kapitel 5.2.3) soll anhand der Verkleinerung der empirischen Kennwerte Mittelwert, Varianz und Median überprüft werden. Tabelle 7.6 zeigt für den Übergang eines Betriebszustands auf den nächsten den Anteil der Stichproben, wo der jeweilige Kennwert nach der Bedruckung kleiner wird, was einem Kompaktieren des Borstenpakets durch die zunehmende Druckdifferenz gleichkommt.

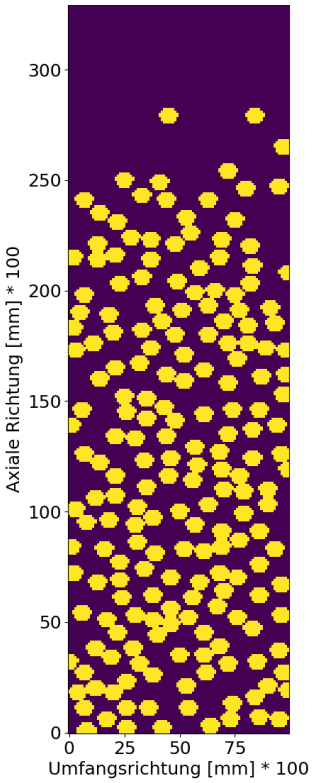
Übergang	**0-1**	**1-2**	**2-3**	**3-4**	**4-5**	**5-6**
Mittelwert	100%	100%	91,67%	95,83%	83,33%	87,5%
Varianz	100%	100%	91,67%	95,83%	87,5%	79,17%
Median	95,83%	100%	87,5%	87,5%	66,67%	79,17%

Tabelle 7.6: Vergleich der empirischen Kennwerte der Bilder in y-Richtung

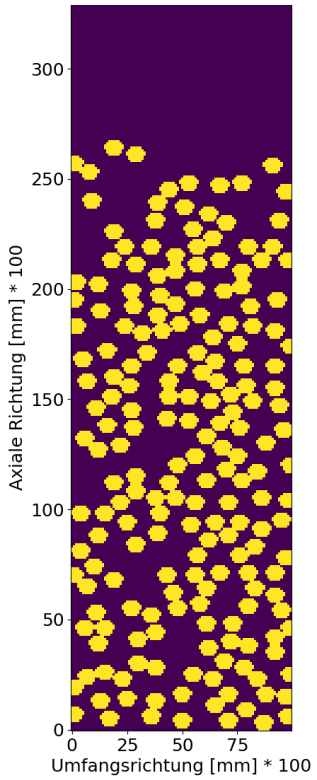
In den empirischen Kennwerten kann eine Verkleinerung insbesondere bei den niedrigen Betriebszuständen festgestellt werden.

7.2 Simulation einer Borstenanordnung

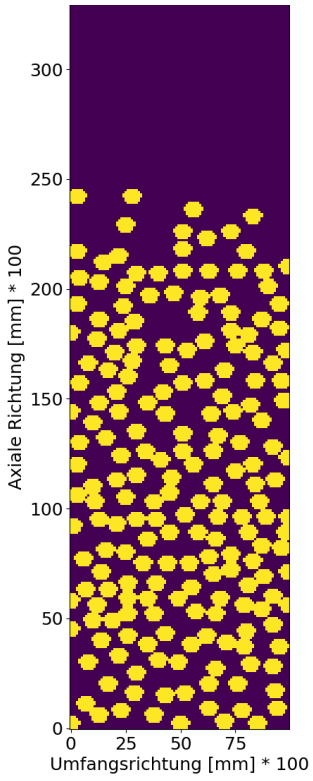
Die ermittelten empirischen Verteilungen dienen als Grundlage zur simulationsbasierten Erstellung von Bürstendichtungsgeometrien für CAD-Modelle. Dazu wurde ein modifizierbares Skript entwickelt.
Hierbei kann die Größe des betrachteten Ausschnitts in Milimetern, jeweils in Axial- und Umfangsrichtung und der betrachtete Betriebszustand (0-6) angegeben werden. Die zugehörigen Beobachtungen der Verteilungen werden auf einen Wertebereich von [0,1] normiert und mit der Größe des jeweiligen Ausschnitts multipliziert, sodass die ermittelten Zufallszahlen sich innerhalb des Ausschnitts befinden. Die Wahl einer Verteilungsfunktion eines Betriebszustands erfolgt dabei zufällig.
Nach Angabe der Größe des betrachteten Ausschnitts wird durch den Fertigungsparameter der Packungsdichte $\rho = 200$ Borsten/mm (nach Tabelle 5.1) die Zahl der zu zeichnenden Borsten ermittelt. Der Mittelpunkt einer Borste entstammt dabei einer zufälligen Ziehung, mittels der Inversionsmethode (vgl. Kapitel 4.3.6). Die Ermittlung der x- und y-Koordinate erfolgt dabei unabhängig voneinander. Die Größe der Halbachsen und zugrundeliegende elliptische Form der Borsten in den Anordnungen entstammt den Überlegungen zur theoretischen Borstenform aus Kapitel 5.2.2. Zusätzlich unterliegt die Ziehung geometrischen Randbedingungen. So muss sichergestellt werden, dass zwei oder mehr Ellipsen nicht überlappen. In diesem Zusammenhang wird überprüft, ob die weißen Pixel eines ermittelten Borstenkandidaten sich nicht mit bisher platzierten Ellipsen-Pixeln überlagern, also ob eine pixelweise Summierung keine Werte größer als 255 liefert. Nur wenn diese Bedingung erfüllt ist, wird die Borste platziert, ansonsten ein neuer Mittelpunkt gezogen. Zusätzlich kann eingestellt werden, ob an den Rändern abgeschnittene Borsten bei der Ziehung erlaubt werden sollen. Hierzu werden die Längen der tatsächlich vorliegenden Halbachsen mit den theoretischen verglichen. Die Mittelpunkte der simulierten Anordnungen werden nach Ablauf des Programms jeweils in einer csv-Datei abgelegt, die zum Einlesen in gängige CAD-Programme und Aufbau eines Modells zur CFD-Simulation genutzt werden kann. Abbildung 7.4 zeigt Beispiele simulierter Borstenanordnungen. Dabei wurde der vorliegende Betriebszustand (Druckdifferenz) variiert.



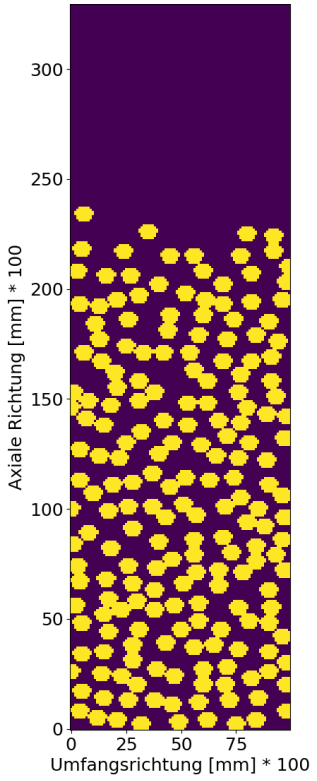
(a) Betriebzustand 0



(b) Betriebszustand 2



(c) Betriebszustand 4



(d) Betriebszustand 6

Abbildung 7.4: Simulierte Anordnungen von Borsten bei unterschiedlichen Betriebszuständen: Der betrachtete Ausschnitt ist 1mm in Umfangsrichtung und 3,3mm in Axialrichtung groß, was dem Maß zwischen Stützring und Deckring der realen Bürstendichtung entspricht.

8. Fazit und Ausblick

In dieser Arbeit wurde ein Erkennungsverfahren erarbeitet, welches anhand von verschiedenen Mikroskop-Bildern einer Bürstendichtung, bei unterschiedlichen Betriebszuständen, die Borsten erkennt. Hierbei wird zunächst das Borstenpaket über eine Linienerkennung lokalisiert, anschließend einzelne Borsten über ein Flächenkriterium erkannt und etwaige Borsten-Cluster einem separaten Verarbeitungsschritt mittels einer Watershed-Segmentation unterzogen. Die Evaluation zeigte im Hinblick auf die Erkennung des Vorhandenseins von Borsten in zufällig gewählten Ausschnitten einen F_1-Score im Bereich von 90,8%-96,4%.
Die Mittelpunkte aus der Erkennung bildeten die Basis für eine statistische Untersuchung der Borstenpositionen. Diese zeigte, dass innerhalb eines Betriebszustands Unterschiede im Hinblick auf die Dispersion, und Schwerpunkte der Verteilungen insbesondere in den y-Koordinaten der Borsten vorliegen könnten. Als theoretisches Modell war in x-Richtung bei der Mehrheit der Stichproben und in y-Richtung nur vereinzelt eine Beta-Verteilung denkbar. Die zugehörigen Programm-Codes sind den Anhängen A, B und C zu entnehmen.
Ferner wurde zur Modellierung einer unregelmäßigen Borstenanordnung ein alternativer Ansatz entwickelt. Dieser basiert auf der Simulation zufällig gewählter empirischer Wahrscheinlichkeitsverteilungen der x- und y-Koordinaten und somit den Ergebnissen zur Position erkannter Borsten. Er ist im Gegensatz zu bisherigen Ansätzen nicht von der Definition eines regelmäßigen Ausgangszustands (vgl. Kapitel 2.5) abhängig. Die ermittelten Geometrien sind in gängige CAD-Programme einlesbar. Der Code hierzu ist dem Anhang D zu entnehmen.
Zur Steigerung der Robustheit und Genauigkeit der Borstenerkennung ist eine Anwendung von Machine Learning sinnvoll, um nicht verwendete oder direkt ersichtliche Merkmale der Bilder, auch ohne einer expliziten Programmierung zu berücksichtigen. Eine Modellierung und Simulation unter Anwendung kombinierter Wahrscheinlichkeiten von x- und y-Koordinaten ist denkbar, um die räumlichen Eigenschaften der Anordnungen besser abzubilden. Möglicherweise lassen sich die Positionen erkannter Borsten über Mischverteilungen beschreiben. Eine nachfolgende CFD-Untersuchung mittels simulierter Bürstendichtungsgeometrien bietet Potenzial, um die Abbildungsfähigkeit radialer, anordnungsabhängiger Strömungsmuster zu untersuchen.

Literatur

Aksit, M. F. (1998). *Evaluation of Brush Seal Performance for Oil Sealing Applications* [Diss., Rensselaer Polytechnic Institute].

Aksit, M. F. (2012). Brush seals and common issues in brush seal applications, pp. 1–14.

Bracke, S. (2022). *Technische Zuverlässigkeit: Datenanalytik, Modellierung, Risikoprognose*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-65015-8

Braun, M. J., Hendricks, R. C., & Canacci, V. (1990). *Flow Visualization in a Simulated Brush Seal* (Bd. Volume 5: Manufacturing Materials and Metallurgy; Ceramics; Structures and Dynamics; Controls, Diagnostics and Instrumentation; General) [V005T16A008]. https://doi.org/10.1115/90-GT-217

Bruce M. Steinetz, R. C. H. (1996). *Engine Seal Technology Requirements to Meet NASA's Advanced Subsonic Technology Program Goals*. NASA Lewis Research Center.

Burger, W., & Burge, M. J. (2015). *Digitale Bildverarbeitung*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-04604-9

Campos, G. F. C., Mastelini, S. M., Aguiar, G. J., Mantovani, R. G., Melo, L. F. d., & Barbon, S. (2019). Machine learning hyperparameter selection for Contrast Limited Adaptive Histogram Equalization. *EURASIP Journal on Image and Video Processing*, *2019*(1). https://doi.org/10.1186/s13640-019-0445-4

Chen, Z., Su, Y., Liu, Y., Huang, J., & Cao, W. (2020). Key technologies of intelligent transportation based on image recognition. *International Journal of Advanced Robotic Systems*, *17*(3), 172988142091727. https://doi.org/10.1177/1729881420917277

Choi, W., Huh, H., Tama, B. A., Park, G., & Lee, S. (2019). A Neural Network Model for Material Degradation Detection and Diagnosis Using Microscopic Images. *IEEE Access*, *7*, 92151–92160. https://doi.org/10.1109/access.2019.2927162

Chupp, R. E. (1991). *Simple Leakage Flow Model for Brush Seals*. American Institute of Aeronautics Astronautics.

Crudgington, P. (1998). *Brush Seal Performance Evaluation*. https://doi.org/10.2514/6.1998-3172

Dogu, Y. (2005). Investigation of Brush Seal Flow Characteristics Using Bulk Porous Medium Approach. *Journal of Engineering for Gas Turbines and Power*, *127*(1), 136–144. https://doi.org/10.1115/1.1808425

Duda, R. O., & Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, *15*(1), 11–15. https://doi.org/10.1145/361237.361242

Fuchs, A., Göttler, J., & Haidn, O. J. (2018). Numerical Investigation On The Leakage Of Brush Seals. https://doi.org/10.5281/ZENODO.1344984

Fuchs, A., & Haidn, O. (2017). Effects of uncertainty and quasi-chaotic geometry on the leakage of brush seals. *17th International Symposium on Transport Phenomena and Dynamics of Rotating Machinery (ISROMAC2017)*. https://hal.science/hal-02410298

Gail, A., & Beichl, S. (2000). MTU brush seal - Main features of an alternative design. *36th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*. https://doi.org/10.2514/6.2000-3375

Hedderich, J., & Sachs, L. (2020). *Angewandte Statistik*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-62294-0

Hildebrandt, M., Schwarz, H., Schwitzke, C., Bauer, H.-J., & Friedrichs, J. (2018). Effects of the Back Plate Inner Diameter on the Frictional Heat Input and General Performance of Brush Seals. *Aerospace*, *5*(2), 58. https://doi.org/10.3390/aerospace5020058

ICAO. (2022). States adopt net-zero 2050 global aspirational goal for international flight operations [Zuletzt aufgerufen am 15.06.2023]. https://www.icao.int/Newsroom/Pages/States-adopts-netzero-2050-aspirational-goal-for-international-flight-operations.aspx

Kolonko, M. (2008). *Stochastische Simulation*. Vieweg+Teubner. https://doi.org/10.1007/978-3-8348-9290-4

Lee, D., Fahey, D., Skowron, A., Allen, M., Burkhardt, U., Chen, Q., Doherty, S., Freeman, S., Forster, P., Fuglestvedt, J., Gettelman, A., León, R. D., Lim, L., Lund, M., Millar, R., Owen, B., Penner, J., Pitari, G., Prather, M., ... Wilcox, L. (2021). The contribution of global aviation to anthropogenic climate forcing for 2000 to 2018. *Atmospheric Environment*, *244*, 117834. https://doi.org/10.1016/j.atmosenv.2020.117834

Lelli, D., Chew, J. W., & Cooper, P. (2005). Combined 3D Fluid Dynamics and Mechanical Modelling of Brush Seals. *Volume 3: Turbo Expo 2005, Parts A and B*. https://doi.org/10.1115/gt2005-68973

MathWorks. (o.D.). Label and Measure Connected Components in a Binary Image [Zuletzt aufgerufen am 07.11.2023]. https://de.mathworks.com/help/images/label-and-measure-objects-in-a-binary-image.html

Messer, M., & Schneider, G. (2019). *Statistik: Theorie und Praxis im Dialog*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-59339-4

Meyer, F. (1992). Color image segmentation. *1992 International Conference on Image Processing and its Applications*, 303–306.

Miller, J., & Ulrich, R. (2019). The quest for an optimal alpha (Y. Li, Hrsg.). *PLOS ONE*, *14*(1), e0208631. https://doi.org/10.1371/journal.pone.0208631

Neef, M., Hepermann, F., Sürken, N., & Schettel, J. (2007). Brush Seal Porosity Modeling – Applicability and Limitations.

OpenCV. (2023). About - OpenCV [Zuletzt aufgerufen am 06.10.2023]. https://opencv.org/about/

OpenCV. (o.D.). *Image Segmentation with Watershed Algorithm: OpenCV 4.5.3 Documentation* [Zuletzt aufgerufen am 07.11.2023]. https://docs.opencv.org/4.x/d3/db4/tutorial_py_watershed.html

Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, *9*(1), 62–66. https://doi.org/10.1109/tsmc.1979.4310076

Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., ter Haar Romeny, B., Zimmerman, J. B., & Zuiderveld, K. (1987). Adaptive

histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, *39*(3), 355–368. https://doi.org/10.1016/s0734-189x(87)80186-x

Pugachev, A. (2013). Predicted performance of brush seals: porous medium versus resolved bristle matrix and comparison with experimental data. *Source of the Document 10th European Conference on Turbomachinery Fluid Dynamics and Thermodynamics, ETC 2013*, 160–170.

Roudier, P., Tisseyre, B., Poilvé, H., & Roger, J.-M. (2008). Management zone delineation using a modified watershed algorithm. *Precision Agriculture*, *9*(5), 233–250. https://doi.org/10.1007/s11119-008-9067-z

Schur, F., & Friedrichs, J. (2019). Operational Performance and Locally Resolved Outflow of Brush Seals. *Volume 3B: Fluid Applications and Systems*. https://doi.org/10.1115/ajkfluids2019-4877

Schwarz, H., Friedrichs, J., & Flegler, J. (2014). *Axial Inclination of the Bristle Pack, a New Design Parameter of Brush Seals for Improved Operational Behavior in Steam Turbines* (Bd. Volume 1B: Marine; Microturbines, Turbochargers and Small Turbomachines; Steam Turbines). https://doi.org/10.1115/GT2014-26330

Suzuki, S., & Abe, K. (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, *30*(1), 32–46. https://doi.org/10.1016/0734-189x(85)90016-7

Szeliski, R. (2022). *Computer Vision*. Springer International Publishing. https://doi.org/10.1007/978-3-030-34372-9

WikimediaCommons. (2016). File:Inverse transformation method for exponential distribution.jpg — Wikimedia Commons, the free media repository [[Zuletzt aufgerufen am 17.11.2023]]. https://commons.wikimedia.org/w/index.php?title=File:Inverse_transformation_method_for_exponential_distribution.jpg&oldid=465197134

Zheng, X., Lu, G., Mack, M., Trivedi, D., Sarawate, N., & Wolfe, C. (2013). Design, Manufacture And Testing Of Variable Bristle Diameter Brush Seals. https://doi.org/10.2514/6.2013-3859

A. Anhang: Programm-Code main.py

```
import cv2
import os
import numpy as np
import Borstenerkennung # siehe Anhang B
import Statistik # siehe Anhang C


path = r'C:\Users\Maksym\Desktop\Subset_Bilder\BP8' # Bildordner zum Einlesen
folder_processed = False # Argument zum Loeschen der Evaluationsordner
all_xy_data = [] # Liste mit Eintraegen aller Punkte
xy_list_1 = [] # Liste mit Mittelpunkten der Erkennung einzelner Borsten
xy_list_2 = [] # Liste mit Mittelpunkten des Cluster-Verfahrens

"""
Einlesen aller Bilder des Ordners bis alle Bilder verarbeitet wurden
"""

# "Debugging" der Bilder nach Verarbeitung jeweils ueber Methode "visualize_img"

for folder in os.listdir(path):
    for i, filename in enumerate(os.listdir(f"{path}\{folder}")):
        img = cv2.imread(os.path.join(path, folder, filename))
        file_path = os.path.join(path, folder, filename)
        print(filename)

        """
        Pre-Processing (Graustufen)
        """
        gray_img = Borstenerkennung.grayscale_img(img)
        Borstenerkennung.visualize_img(gray_img)

        """
        Region of Interest (Lokalisierung des Borstenpakets)
        """
        roi_img, line_img, cropped_img, color_cropped_img = Borstenerkennung.find_roi(gray_img, img)
        Borstenerkennung.visualize_img(color_cropped_img)
        Borstenerkennung.visualize_img(line_img)
        Borstenerkennung.visualize_img(cropped_img)


        """
        Erkennung einzelner Borsten, zeichnen der gruenen Konturen auf Bild
        """
        clahe_img = Borstenerkennung.clahe_img(cropped_img)
        Borstenerkennung.visualize_img(clahe_img)
        thresh_img = Borstenerkennung.threshold_image(clahe_img) # OTSU
        Borstenerkennung.visualize_img(thresh_img)
        overlay_img_small, xy_list_1, overlay_img_thresh, big_contour_img, area_list_1, area_25, area_75 = Borstenerkennung.overlay_img(img_orig = cropped_img, thresh_img = thresh_img)
        Borstenerkennung.visualize_img(overlay_img_small) # Originalbild mit Konturen
        Borstenerkennung.visualize_img(overlay_img_thresh) # Binaerbild mit Konturen
        Borstenerkennung.visualize_img(big_contour_img) # Cluster-Bild zur weiteren Verarbeitung

        """
        Watershed-Segmentierungsverfahren (inkl. Verarbeitung des Rests + Zufaellige Verteilung
        bei verbleibenden Clustern)
```

```
56          """
57
58          new_overlay, xy_list_2 = Borstenerkennung.watershed(cropped_img,
        big_contour_img, overlay_img_small, area_list_1, area_25, area_75,
        color_cropped_img)
59
60
61
62          """
63          Speichern des Bildes mit Konturen und Mittelpunkten zur visuellen
        Evaluation
64          """
65          Borstenerkennung.slice_img(new_overlay, file_path, folder, folder_processed
        )
66
67
68          """
69          Speichern von Mittelpunkten eines Bildes zur weiteren Verarbeitung
70          (Mittelpunkte von Verfahren 1 + 2)
71          """
72          point_cloud = np.array(xy_list_1 + xy_list_2)
73          Statistik.plot_points(point_cloud, filename)  # Plot und Speichern der
        Scatter-Plots jedes Bildes
74          print(f"Bild {i} verarbeitet")
75
76      all_xy_data.append(point_cloud) #Fuege erkannte Mittelpunkte der Liste aller
        Mittelpunkte zu
77
78  """
79  Statistik (nach Verarbeitung aller Bilder)
80  """
81  Statistik.boxplot(all_xy_data) # Boxplots innerhalb eines Betriebszustandes und
        dazwischen
82  Statistik.y_parameter(all_xy_data) # Empirische Parameter der y-Koordinaten
83  Statistik.CDF(all_xy_data) # Empirische Verteilungen aus Beobachtungen
84  Statistik.GoF(all_xy_data) # Kolmogorov-Smirnov-Anpassungstest
85  Statistik.levene(all_xy_data) # Levene-Test
86  Statistik.mwu(all_xy_data) # Mann-Whitney-U-Test
87  Statistik.point_count(all_xy_data) # Durchschnittlich erkante Zahl von Borsten pro
        Bild
```

B. Anhang: Programm-Code Borstenerkennung.py

```
1
2  import cv2
3  import numpy as np
4  import os
5  from statistics import mean
6  import math
7  import random
8
9  """
10 Visualisierung eines Bildes fuer "Debugging" (Dauer der Anzeige ueber waitKey
           steuern)
11 """
12
13 def visualize_img(img):
14     cv2.namedWindow('custom window', cv2.WINDOW_KEEPRATIO)
15     cv2.imshow('custom window', img)
16     width = int(img.shape[1])
17     height = int(img.shape[0])
18     dim_window = int(0.2 * width), int(0.2 * height)
19     cv2.resizeWindow('custom window', dim_window)
20     cv2.waitKey(0)
21     cv2.destroyAllWindows()
22
23 """
24 Konvertierung des Bildes in Graustufen
25 """
26
27 def grayscale_img(img):
28     gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
29
30     return gray_img
31
32 """
33 Dilation mit elliptischem Strukturelement und vorzugebender Kernel-Groesse
34 """
35
36 def dilate_img(img, ksize):
37     kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(ksize,ksize))
38     dilated_img = cv2.dilate(img, kernel)
39     return dilated_img
40
41
42 """
43 Erstellen und Speichern von gleich grossen Bildausschnitten
44 """
45
46 def slice_img(img, path_sample, folder, folder_processed):
47     num_cuts = 16 # Zahl der Ausschnitte pro Bilder
48     crop_height, crop_width = img.shape[0], int(img.shape[1]/num_cuts)
49     x = np.linspace(0, img.shape[1], num_cuts, dtype=int, endpoint=False)
50     file_name = os.path.splitext(os.path.basename(path_sample))[0]
51
52     output_dir = r'C:\Users\Maksym\Desktop\Python_Projekt\Evaluationsbilder\BP8' #
           Ordner zur Ablage des geteilten Bildes
53
54     if not folder_processed: # Loeschen der Eintraege des Ordners
55         for item in os.listdir(f'{output_dir}'):
56             item_path = os.path.join(output_dir, folder)
57             if os.path.isfile(item_path):
```

```
58                 os.remove(item_path)
59
60         folder_processed = True
61
62     for i in range(num_cuts): # Speichern der geteilten Bilder mit File_Name +
           Nummer des Schnitts
63         w = x[i]+crop_width
64         crop_image = img[0:crop_height, x[i]:w]
65         out = os.path.join(output_dir, folder,f"{file_name}_num{i}.jpg") #f"{str(i)
           }.jpg"
66         cv2.imwrite(out, crop_image)
67         cv2.waitKey(0)
68         crop_image = None
69
70     return folder_processed
71
72 """
73 Erkennung einzelner Borsten und Erstellen eines Overlays (Originalbild + Konturen)
74 """
75
76 def overlay_img(img_orig, thresh_img):
77     img_orig_color = cv2.cvtColor(img_orig, cv2.COLOR_GRAY2RGB) #Konventierung in
           Farbbild um gruene Konturen zeichnen zu koennen
78     thresh_img_color = cv2.cvtColor(thresh_img, cv2.COLOR_GRAY2RGB) #Konventierung
           in Farbbild um gruene Konturen zeichnen zu koennen
79     contours, _ = cv2.findContours(thresh_img, cv2.RETR_EXTERNAL, cv2.
           CHAIN_APPROX_NONE) #cv2.RETR_EXTERNAL ->> nur aeussere Konturen
80
81     xy_list = []
82     area_list = []
83     black_img = np.zeros_like(img_orig) # Schwarzes Bild zum Zeichnen der Cluster
84     bristle_count = 0
85
86     # Ermittle 25. und 75. Perzentil aller Konturen mit Flaeche > 10 Pixel
87     area_list = [cv2.contourArea(c) for c in contours if cv2.contourArea(c) > 10]
88     area_75 = np.percentile(area_list, 75)
89     area_25 = np.percentile(area_list, 25)
90
91     # Zeichnen der Konturen und Mittelpunkte
92     for c in contours:
93         if cv2.contourArea(c) > area_25 and cv2.contourArea(c) < area_75:
94             cx = int(np.average(c[:,0,0]))
95             cy = int(np.average(c[:,0,1]))
96             xy_list.append((cx, img_orig.shape[0] - cy))
97             area_list.append(cv2.contourArea(c)) # Liste aller Flaechen (zur
               spaetern Bestimmung des Medians)
98             cv2.circle(img_orig_color, (cx, cy), 4, (255, 0, 0), -1)
99             cv2.circle(thresh_img_color, (cx, cy), 4, (255, 0, 0), -1)
100
101            bristle_count = bristle_count + 1
102
103            overlay_img_small = cv2.drawContours(img_orig_color, [c], -1, (0,255,0)
               , thickness=1)
104            overlay_img_thresh = cv2.drawContours(thresh_img_color, [c], -1,
               (0,255,0), thickness=1)
105
106        # Zeichnen des Inneren der grossen Konturen auf Cluster-Bild
107        if cv2.contourArea(c) >= area_75:
108            big_contour_img = cv2.drawContours(black_img, [c], -1, 255, thickness=
               cv2.FILLED)
109
110
111
112    return overlay_img_small, xy_list, overlay_img_thresh, big_contour_img,
           area_list, area_25, area_75
113
114 """
115 Schwellenwertverfahren zur Erzeugung von Binaerbild (OTSU)
116 """
117
118 def threshold_image(img):
119
120     _, thresh_img = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU) #150
121
```

```
122     return thresh_img
123 
124 """
125 Adaptiver Histogrammausgleich (Contrast Limited)
126 """
127 
128 def clahe_img(img):
129     clip_limit_list = [3.0] # bei Bedarf andere Parameter ergaenzen
130     for clip_limit in clip_limit_list:
131         clahe = cv2.createCLAHE(clipLimit=clip_limit, tileGridSize=(32,32))
132         cl1 = clahe.apply(img)
133 
134     return cl1
135 
136 """
137 Finden (Rotation + Sobel + Median-Filter + Hough-Line) und Crop der ROI (Region of
            Interest)
138 
139 Erstellen und Rueckgabe des Ausschnitts: 1. Grau zur Erkennung einzelner Borsten,
            2. in Farbe fuer Watershed
140 """
141 
142 def find_roi(img, color_img_to_crop):
143 
144     # Rotation mit -0,5Grad um den Mittelpunkt
145     M = cv2.getRotationMatrix2D((int(img.shape[1])/2,int(img.shape[0]/2)), -0.5, 1)
146     rotated = cv2.warpAffine(img, M, (int(img.shape[1]), int(img.shape[0])))
147     visualize_img(rotated) # Debugging
148 
149     # Sobel in y-Richtung
150     ddepth = cv2.CV_16S
151     delta = 0
152     scale = 1
153     grad_y = cv2.Sobel(rotated, ddepth, 0, 1, ksize=3, scale=scale, delta=delta,
            borderType=cv2.BORDER_DEFAULT)
154     abs_y = cv2.convertScaleAbs(grad_y) # Konvertierung des Sobel-Bilds in Betrag
155 
156     visualize_img(abs_y) # Debugging
157 
158     _, edge_img_y = cv2.threshold(abs_y, int(0.15 * np.max(abs_y)), 255, cv2.
            THRESH_BINARY) # Kandidaten fuer Linienerkennung
159 
160     visualize_img(edge_img_y) # Debugging
161 
162     # Median-Filter
163     edge_img_y = cv2.medianBlur(edge_img_y, ksize = 9) # Entfernen von Rauschpixeln
            , Glaetten der Linie
164 
165     visualize_img(edge_img_y) # Debugging
166 
167     # Hough-Linienerkennung
168 
169     point_thresh = 1800
170     lines = cv2.HoughLines(edge_img_y, 1, np.pi/180, point_thresh)
171     color_img = cv2.cvtColor(edge_img_y, cv2.COLOR_GRAY2BGR) # Bild zum Zeichnen
            der Konturen
172 
173     lines = np.flipud(lines[lines[:,0,0].argsort()]) #Sortierung, um von am
            weitesten entfernten Linien zu beginnen
174     for i, line in enumerate(lines):
175         rho = lines[i,0,0]
176         theta = lines[i,0,1]
177         schwelle_1 = np.deg2rad(-0.25) # Toleranzen Linie (+/-0.25 deg)
178         schwelle_2 = np.deg2rad(0.25)
179         angle = math.cos(theta) / math.sin(theta) # Steigung Linie
180         if angle > schwelle_1 and angle < schwelle_2: # Pruefen ob Linie gefunden
181             break
182 
183     a = math.cos(theta)
184     b = math.sin(theta)
185     x0 = a * rho
186     y0 = b * rho
187     pt1_max = (int(x0 + 4800*(-b)), int(y0 + 4800*(a)))
188     pt2_max = (int(x0 - 4800*(-b)), int(y0 - 4800*(a)))
```

```
189
190     line_img = cv2.line(color_img, pt1_max, pt2_max, (0, 255, 0), 4, cv2.LINE_AA) #
            Linie mit maximalen Abstand zeichnen und anzeigen
191
192     y_crop = int(mean([pt1_max[1], pt2_max[1]])) # y-Koordinate fuer Crop
193
194     y_crop = y_crop - 130  #Versatz zum Vernachlaessigen des Stuetzrings
195
196     width = int(img.shape[1])
197     cropped_img = img[(y_crop-850):y_crop, int(0.05*width):int(0.95*width)] #CROP (
            GRAU)
198
199     cropped_color_img = color_img_to_crop[(y_crop-850):y_crop, int(0.05*width):int
            (0.95*width)] #CROP (FARBE)
200
201     return edge_img_y, line_img, cropped_img, cropped_color_img
202
203 """
204 Watershed-Verfahren fuer gefiltertes Binaerbild
205 +
206 Schaetzung in Bereichen gebliebener Cluster
207 +
208 Verarbeitung uebergebliebener Reste
209 """
210
211 def watershed(img, thresh_img, overlay_img, area_list, area25, area75,
            color_img_orig):
212
213
214     color_img = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR) #Bild zum Zeichnen der
            Konturen
215     color_img_thresh = cv2.cvtColor(thresh_img, cv2.COLOR_GRAY2BGR) #Bild zum
            Zeichnen der Konturen
216
217     param_list_thresh = (np.linspace(1, 255, 40, endpoint=False)).astype(np.uint8)
            #Parametervariation -> "optimale" Kombination
218     param_list_ksize = [3,5,7]
219
220     markers_list = []
221     good_area_count_list = []
222
223     #Parametervariation -> Waehlen der "optimalen" Kombination mit maximaler Zahl
            von Flaechen im Bereich
224     # > 25. Perzentil und < 75.Perzentil
225
226     for param in param_list_ksize:
227         sure_bg = dilate_img(thresh_img, param)
228         for param in param_list_thresh:
229
230             dist = cv2.distanceTransform(thresh_img, cv2.DIST_L2, 3) #
                Distanztransformation (euklidisch)
231             _, dist_thresh =  cv2.threshold(dist, param, 255, cv2.THRESH_BINARY)
232
233             sure_fg = dist_thresh.astype(np.uint8) #Vordergrund (Marker),
                Hintergrund, unbekannter Bereich
234             unknown = cv2.subtract(sure_bg, sure_fg)
235             _, markers = cv2.connectedComponents(sure_fg)
236             markers = markers + 1
237             markers[unknown==255] = 0
238
239             watershed_markers = cv2.watershed(color_img_orig, markers) # Finden der
                 Watershed-Segmente
240             markers_list.append(watershed_markers)
241             count_list = []
242
243
244             # Finden des "optimalen" Parametersatzes
245             for i in range(2, np.max(watershed_markers)):
246                 x = np.size(np.where(watershed_markers == i))
247                 if x >= area25 and x <= area75:
248                     count_list.append(x)
249             good_area_count_list.append(len(count_list)) # Flaechen in dem
                gewuenschten Intervall
250
```

```
251     watershed_markers = markers_list[np.argmax(good_area_count_list)] #Nehme die
            Parameter / Index an, wo die Zahl der Borsten in den mittleren 50%, der
            bekannten Flaechen maximiert wird
252
253     xy_list = [] # Liste zur Speicherung der Mittelpunkte
254
255     # Bild mit Grenzen der Segmente
256     color_img_thresh[watershed_markers == -1] = (0,0,255)
257     visualize_img(color_img_thresh) # Debugging
258
259     unique_val = list(range(2, np.max(watershed_markers)))
260
261     # Zeichnen den Mittelpunkt derjenigen Konturen, wo das Flaechenkriterium
            erfuellt ist
262
263     for unique in unique_val:
264         x = np.argwhere(watershed_markers == unique)
265         if x.shape[0] > area25 and x.shape[0] < area75:
266             cx = int(np.mean(x[:,1]))
267             cy = int(np.mean(x[:,0]))
268             xy_list.append((cx, thresh_img.shape[0] - cy))
269             area_list.append(x.size)
270
271             color_img = cv2.circle(color_img, (cx, cy), 5, (255, 0, 0), -1)
272             overlay_img = cv2.circle(overlay_img, (cx, cy), 5, (255, 0, 0), -1)
273             color_img_thresh = cv2.circle(color_img_thresh, (cx, cy), 5, (255, 0,
                0), -1)
274
275
276         # Zeichne Grenzen der Konturen
277         overlay_img[watershed_markers == -1] = (0,0,255)
278         color_img[watershed_markers == -1] = (0,0,255)
279
280         # Median aller bisher erkannten Flaechen zur Schaetzung in verbleibenden
                Clustern
281         median_area = np.median(area_list)
282
283         # Verarbeitung von verbleibenden Clustern
284         if x.shape[0] >= area75:
285             cx = int(np.mean(x[:,1]))
286             cy = int(np.mean(x[:,0]))
287
288             weight = round(x.shape[0] / median_area)
289             for i in range(0,weight):
290                 if weight == 1: # Ein Mittelpunkt wird gezeichnet
291                     xy_list.append((cx, color_img_thresh.shape[0] - cy))
292                     color_img = cv2.circle(color_img, (cx, cy), 5, (255, 0, 0), -1)
293                     overlay_img = cv2.circle(overlay_img, (cx, cy), 5, (255, 0, 0),
                     -1)
294                 if weight >= 2: # Mehrere Mittelpunkte werden gezeichnet und
                zufaellig in dem Segment unterteilt
295                     rand = random.randint(0, x.shape[0]-1)
296                     cy_rand, cx_rand = x[rand]
297                     xy_list.append((cx_rand, color_img_thresh.shape[0] - cy_rand))
298
299                     color_img = cv2.circle(color_img, (cx_rand, cy_rand), 5, (255,
                255, 255), -1)
300                     overlay_img = cv2.circle(overlay_img, (cx_rand, cy_rand), 5,
                (255, 255, 255), -1)
301                     cv2.putText(overlay_img, f"{weight}", (cx, cy), cv2.
                FONT_HERSHEY_SIMPLEX, 1, (0,255,255), 2) # Text (Zahl geschaetzter
                Borsten) fuer Evaluation
302
303
304
305     visualize_img(color_img_thresh) # Debugging
306     visualize_img(overlay_img) # Debugging
307
308     found = np.where(np.isin(watershed_markers, unique_val), 255, 0) # Erzeugung
            des Restbilds: Original(thresh) - gefundene Konturen (Watershed)
309     found = np.reshape(found, np.shape(thresh_img))
310     remaining_img = thresh_img - found
311     remaining_img[watershed_markers == -1] = 0
312     remaining_img = remaining_img.astype(np.uint8)
```

```
313
314     remaining_img_color = cv2.cvtColor(remaining_img, cv2.COLOR_GRAY2RGB) #
            Farbbild zum Zeichnen der Konturen
315
316     # Finde verbleibende Borsten im Restbild
317
318     contours, _  = cv2.findContours(remaining_img, cv2.RETR_EXTERNAL, cv2.
            CHAIN_APPROX_NONE)
319
320
321     # Verbleibende Borsten zeichnen, speichern
322
323     for c in contours:
324         if cv2.contourArea(c) > area25:
325
326
327             cx = int(np.average(c[:,0,0]))
328             cy = int(np.average(c[:,0,1]))
329             xy_list.append((cx, remaining_img.shape[0] - cy))
330             cv2.circle(remaining_img_color, (cx, cy), 4, (255, 0, 0), -1)
331             overlay_img = cv2.circle(overlay_img, (cx, cy), 5, (255, 0, 0), -1)
332
333             remaining_img = cv2.drawContours(remaining_img_color, [c], -1,
                (0,0,255), thickness=1)
334             overlay_img = cv2.drawContours(overlay_img, [c], -1, (0,0,255),
                thickness=1)
335
336
337     visualize_img(remaining_img) # Debugging
338     visualize_img(overlay_img) # Debugging
339
340
341     return overlay_img, xy_list
```

C. Anhang: Programm-Code Statistik.py

```
1
2  import numpy as np
3  from scipy import stats
4  import matplotlib.pyplot as plt
5  import pickle
6  from statistics import mean
7
8  """
9  MLE (Fit) + Kolmogoroff-Smirnov-Anpassungstest, fuer jedes Bild eines
               Betriebszustands
10 +
11 Ermittlung Anteil p>=0,05 pro Betriebszustand
12 """
13
14 def GoF(all_xy_data):
15
16
17     dist_names = ['uniform', 'norm','weibull_min','gamma','lognorm','beta']
18
19
20     for dist_name in dist_names:
21         dist = getattr(stats, dist_name)
22         for i, op_state in enumerate(all_xy_data):
23             count_x = 0
24             count_y = 0
25             for image_points in op_state:
26                     x_data = image_points[:,0]
27                     y_data = image_points[:,1]
28
29                     params_x = dist.fit(x_data)
30                     params_y = dist.fit(y_data)
31
32                     res_x = stats.kstest(x_data, dist_name, params_x)
33                     if res_x.pvalue >= 0.05:
34                         count_x = count_x + 1
35
36                     res_y = stats.kstest(y_data, dist_name, params_y)
37                     if res_y.pvalue >= 0.05:
38                         count_y = count_y + 1
39
40             share_x = count_x / 24
41             share_y = count_y / 24
42
43             print(f"KS-TEST, {dist_name}, BETRIEBSZUSTAND {i}, x-Richtung: {share_x
               }")
44             print(f"KS-TEST, {dist_name}, BETRIEBSZUSTAND {i}, y-Richtung: {share_y
               }")
45
46
47 """
48 Mann-Whitney-U-Test, fuer alle einzigartigen Kombinationen zweier Bilder innerhalb
               eines Betriebszustands
49 +
50 Ermittlung Anteil p>=0,05 pro Betriebszustand
51 """
52
53 def mwu(all_xy_data):
54     for o, operating_state in enumerate(all_xy_data):
55         p_x_list = []
```

```
56          p_y_list = []
57          for i in range(0, len(operating_state)):
58              for j in range(0, len(operating_state)):
59                  if j > i:
60                      point_cloud_1 = operating_state[i]
61                      point_cloud_2 = operating_state[j]
62                      _, p_x = stats.mannwhitneyu(point_cloud_1[:,0], point_cloud_2
                  [:,0], method="auto")
63                      _, p_y = stats.mannwhitneyu(point_cloud_1[:,1], point_cloud_2
                  [:,1], method="auto")
64                      p_x_list.append(p_x)
65                      p_y_list.append(p_y)
66
67          percent_x = 100 * len([p for p in p_x_list if p >= 0.05]) / len(p_x_list)
68          percent_y = 100 * len([p for p in p_y_list if p >= 0.05]) / len(p_y_list)
69
70          print(f"{percent_x}%, Betriebszustand {o}, MWU x-Richtung")
71          print(f"{percent_y}%, Betriebszustand {o}, MWU y-Richtung")
72
73  """
74  Levene-Test, fuer alle Bilder eines Betriebszustands
75  """
76
77  def levene(all_xy_data):
78      for i, operating_state in enumerate(all_xy_data):
79          x = [point_cloud[:,0] for point_cloud in operating_state]
80          stat_x, p_x = stats.levene(*x)
81
82          y = [point_cloud[:,1] for point_cloud in operating_state]
83          stat_y, p_y = stats.levene(*y)
84
85          print(f"LEVENE-TEST, BETRIEBSZUSTAND {i}, x-Richtung, statx:{stat_x}, p_x:{
                p_x}")
86          print(f"LEVENE-TEST, BETRIEBSZUSTAND {i}, y-Richtung, staty:{stat_y}, p_y:{
                p_y}")
87
88
89  """
90  Empirische Verteilung (CDF) und Normierung dieser
91  +
92  Ablage als PKL zur Durchfuehrung der Simulation
93  +
94  Plot
95  """
96
97  def CDF(all_xy_data):
98
99      for o, op_state in enumerate(all_xy_data):
100         for i, img_points in enumerate(op_state):
101
102             path = f"C:/Users/Maksym/Desktop/Python_Projekt/
                Verteilungen_Betriebszustand_{o}/{i}.pkl"
103
104             #Originale (ohne Normierung) Verteilungen fuer Plot
105             img_x_orig = img_points[:,0]
106             img_y_orig = img_points[:,1]
107
108             data_x_orig, counts_x_orig = np.unique(img_x_orig, return_counts=True)
109             cdf_x_orig = np.cumsum(counts_x_orig) / np.cumsum(counts_x_orig)[-1]
110             data_y_orig, counts_y_orig = np.unique(img_y_orig, return_counts=True)
111             cdf_y_orig = np.cumsum(counts_y_orig) / np.cumsum(counts_y_orig)[-1]
112
113
114
115             #Normierung [0,1] + Ablage in PKL-Datei
116             img_x_data = img_points[:,0] / 4320
117             img_y_data = img_points[:,1] / 900 # Abstand Stuetzring - Deckring
118
119             data_x, counts_x = np.unique(img_x_data, return_counts=True)
120             cdf_x = np.cumsum(counts_x) / np.cumsum(counts_x)[-1]
121             data_y, counts_y = np.unique(img_y_data, return_counts=True)
122             cdf_y = np.cumsum(counts_y) / np.cumsum(counts_y)[-1]
123
124             with open(path, 'wb') as f:
```

```
125                 pickle.dump([data_x, cdf_x, data_y, cdf_y], f)
126
127             # Plot
128
129             _, (ax1, ax2) = plt.subplots(nrows = 2, ncols = 1)
130             ax1.set_title('Empirische Verteilung x-Richtung', fontsize = 18)
131             ax1.plot(data_x_orig, cdf_x_orig, drawstyle='steps-post')
132             ax1.set_xlabel('Pixel (x-Richtung)', fontsize=18)
133             ax1.set_ylabel('Wahrscheinlichkeit', fontsize=18)
134             ax2.set_title('Empirische Verteilung y-Richtung', fontsize=18)
135             ax2.set_xlabel('Pixel (y-Richtung)',fontsize=18)
136             ax2.set_ylabel('Wahrscheinlichkeit',fontsize=18)
137
138             ax2.plot(data_y_orig, cdf_y_orig, drawstyle='steps-post')
139
140             ax1.tick_params(axis='both', which='both', labelsize= 18)
141             ax2.tick_params(axis='both', which='both', labelsize= 18)
142
143
144             plt.tight_layout()
145             plt.show()
146
147
148
149 """
150 Box-Plots (innerhalb eines Betriebszustands, zwischen den Betriebszustaenden)
151 """
152
153 def boxplot(all_xy_data):
154
155     for o, op_state in enumerate(all_xy_data):
156         fig, (ax1, ax2) = plt.subplots(nrows = 2, ncols = 1)
157         medianprops = dict(color='black')
158         ax1.boxplot([point_cloud[:,0] for point_cloud in op_state], medianprops=
                medianprops)
159         ax1.set_ylabel('Pixel (x-Richtung)')
160         ax1.set_xlabel('Bild Nr.')
161         ax2.boxplot([point_cloud[:,1] for point_cloud in op_state], medianprops=
                medianprops)
162         ax2.set_ylabel('Pixel (y-Richtung)')
163         ax2.set_xlabel('Bild Nr.')
164
165         fig.savefig(f'C:/Users/Maksym/Desktop/Python_Projekt/Box-Plots/
                Innerhalb_Betriebszustand/{o}', dpi=300, bbox_inches='tight')
166
167     for i in range(0,24):
168         fig, (ax1, ax2) = plt.subplots(nrows = 2, ncols = 1)
169
170         ax1.set_xlabel('Betriebszustand', fontsize=18)
171         ax1.set_ylabel('Pixel (x-Richtung)', fontsize=18)
172         ax2.set_xlabel('Betriebszustand',fontsize=18)
173         ax2.set_ylabel('Pixel (y-Richtung)',fontsize=18)
174
175         medianprops = dict(color='black')
176
177         x_list = []
178         y_list = []
179         for op_state in all_xy_data:
180             point_cloud = op_state[i]
181             x_list.append((point_cloud[:,0]))
182             y_list.append((point_cloud[:,1]))
183
184         ax1.set_xticklabels(range(0,7))
185         ax1.tick_params(axis='both', which='both', labelsize=21)
186         ax1.boxplot(x_list, medianprops=medianprops)
187         ax2.set_xticklabels(range(0,7))
188         ax2.tick_params(axis='both', which='both', labelsize=21)
189         ax2.boxplot(y_list, medianprops=medianprops)
190
191
192         fig.savefig(f'C:/Users/Maksym/Desktop/Python_Projekt/Box-Plots/
                Zwischen_Betriebszustand/Bild_{i}', dpi=500, bbox_inches='tight')
193
194
```

```
195 """
196 Durchschnittliche Borstenzahl pro Bild
197 """
198
199
200 def point_count(all_xy_data):
201     for i, op_state in enumerate(all_xy_data):
202         avg = mean([len(point_cloud[:,0]) for point_cloud in op_state])
203         print(f"Betriebszustand{i}: Durchschnittliche Borsten pro Bild: {avg}")
204
205
206 """
207 Ueberpruefen der Verkleinerung der Kennwerte in Abhaengigkeit vom Betriebszustand
              fuer jedes Bild / Stelle in BD
208 """
209
210
211 def y_parameter(all_xy_data):
212
213     for o in range(0,6):
214         count_avg = 0
215         count_var = 0
216         count_med = 0
217         for i in range(0,24):
218             o_state_prev = all_xy_data[o]
219             o_state_next = all_xy_data[o + 1]
220             img1 = o_state_prev[i]
221             img2 = o_state_next[i]
222             y1 = img1[:,1]
223             y2 = img2[:,1]
224
225             y1_average = np.average(y1)
226             y2_average = np.average(y2)
227             y1_variance = np.var(y1)
228             y2_variance = np.var(y2)
229             y1_median = np.median(y1)
230             y2_median = np.median(y2)
231
232             if y2_average < y1_average:
233                 count_avg = count_avg + 1
234             if y2_variance < y1_variance:
235                 count_var = count_var + 1
236             if y2_median < y1_median:
237                 count_med = count_med + 1
238
239         share_avg = count_avg/24
240         share_var = count_var/24
241         share_med = count_med/24
242
243         print(f"AVG, Betriebszustand {o} -> {o+1}, y-Richtung, {share_avg}")
244         print(f"VAR, Betriebszustand {o} -> {o+1}, y-Richtung, {share_var}")
245         print(f"MED, Betriebszustand {o} -> {o+1}, y-Richtung, {share_med}")
246
247 """
248 Scatter-Plot von Mittelpunkten mit jew. Histogramm
249 """
250
251 def plot_points(xy_list, filename): #angelehnt an https://matplotlib.org/stable/
              gallery/lines_bars_and_markers/scatter_hist.html#sphx-glr-gallery-lines-
              bars-and-markers-scatter-hist-py
252
253
254     fig = plt.figure(figsize=(25,10))
255
256     ax = fig.add_gridspec(top=0.95, right=0.75).subplots()
257     ax.set(aspect=1)
258     ax_histx = ax.inset_axes([0, 1.05, 1, 0.25], sharex=ax)
259     ax_histy = ax.inset_axes([1.05, 0, 0.25, 1], sharey=ax)
260
261     ax.tick_params(axis='both', which='both', labelsize= 21)
262     ax_histx.tick_params(labelbottom=False, labelsize = 21)
263     ax_histy.tick_params(labelleft=False, labelsize = 21)
264
265     ax.scatter(xy_list[:,0], xy_list[:,1], c='black', s=5)
```

```
266     ax.set_xlabel('Pixel (x-Richtung)', fontsize=22)
267     ax.set_ylabel('Pixel (y-Richtung)', fontsize=22)
268
269     ax_histx.set_ylabel('Zahl', fontsize=22)
270     ax_histy.set_xlabel('Zahl', fontsize=22)
271
272
273     binwidth = 50 # Klassenteilung
274     xmax = max(xy_list[:,0])
275     ymax = max(xy_list[:,1])
276     x_lim = (int(xmax/binwidth + 1) * binwidth)
277     y_lim = (int(ymax/binwidth + 1) * binwidth)
278
279     x_bins = np.arange(0, x_lim + binwidth, binwidth)
280     y_bins = np.arange(0, y_lim + binwidth, binwidth)
281
282     ax_histx.hist(xy_list[:,0], bins=x_bins, color='white', ec='black')
283     ax_histy.hist(xy_list[:,1], bins=y_bins, orientation='horizontal', color='white
            ', ec='black')
284
285     # Ablageordner fuer Plots
286     fig.savefig(f'C:/Users/Maksym/Desktop/Python_Projekt/Scatter-Plots/{filename}',
             dpi=300, bbox_inches='tight')
```

D. Anhang: Programm-Code Simulation.py

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import pickle
import random
import csv
import os


"""
Finden zufaelliger (nicht ueberlappender) Borsten in spezifizierbaren Ausschnitt
            anhand von normierten empirischen Verteilungen aus der Borstenerkennung
"""


def simulate_brush(data_x, cdf_x, data_y, cdf_y):

    size_x = 1 # Laenge in Umfangsrichtung [mm]
    size_y = 3.3 # Laenge in Axialrichtung [mm]
    size = (int(size_y*100), int(size_x*100), 1) # Dimensionen des Ausschnitts *
            Skalenfaktor (100)
    a = 0.099 / 2 #gro e Halbachse [mm]
    b = 0.07 / 2 # kleine Halbachse [mm]
    bpmm = 200 # Borsten / mm Umfangsrichtung (Packungsdichte)

    x_data = []
    y_data = []
    z_data = []


    black_img = np.zeros(size, np.uint8) # Bild zum Zeichnen der Borsten
    bristle_count = 0



    while bristle_count < (bpmm * size_x):
        x = random.uniform(0,1) # Inversionsmethode
        y = random.uniform(0,1)

        simulated_x = data_x[min(np.argwhere((cdf_x >= x)))] # Naechstmoeglicher
            Wert der empirischen Verteilung
        simulated_y = data_y[min(np.argwhere((cdf_y >= y)))]


        test_x = int(simulated_x * 100 * size_x)
        test_y = int(simulated_y * 100 * size_y)

        test_img = cv2.ellipse(np.zeros(size, np.uint8), (test_x,test_y), (int(a
            *100),int(b*100)), 0, 0, 360, 255, 1)

        # ggf. Ueberpruefung ob die Ellipsen vollstaendig auf den Bildern sind
            ueber ihre Halbachsen
        if prevent_cut == True:
            test_points = np.argwhere(test_img==255)
            if (max(test_points[:,0] - min(test_points[:,0])) < 2 * int(b * 100)):
                continue
            if (max(test_points[:,1] - min(test_points[:,1])) < 2 * int(a * 100)):
                continue

```

```
55          if bristle_count > 0:
56              sum_img = (test_img.astype(np.uint64) + ellipse_img.astype(np.uint64))
57
58              # Ueberpruefen der pixelweisen Summierung einer zu platzierenden
                Borsten und bisher platzierten Borsten (zur Verhinderung des
                Ueberlappens)
59              if np.max(sum_img > 255):
60                  continue # neue Ziehung
61
62
63          ellipse_img = cv2.ellipse(black_img, (int(simulated_x * 100 * size_x),int(
                simulated_y * 100 * size_y)), (int(a*100),int(b*100)), 0, 0, 360, 255,
                thickness=cv2.FILLED)
64
65
66          bristle_count = bristle_count + 1
67          print(f"Borste {bristle_count} gefunden")
68          x_data.append(test_x)
69          y_data.append(test_y)
70          z_data.append(0)
71
72          test_img = []
73
74
75      plt.imshow(ellipse_img)
76      plt.gca().invert_yaxis() #Invertierung wegen anderem Koordinatensystem (Borsten
                sollen unten anliegen)
77      plt.title('Simulierte Anordnung', fontsize=18)
78      plt.xlabel('Umfangsrichtung [mm] * 100', fontsize=18)
79      plt.ylabel('Axiale Richtung [mm] * 100', fontsize=18)
80      plt.tick_params(axis='both',labelsize=18)
81      plt.show()
82
83      with open('CAD_POINTS.csv', 'w', newline='') as csvfile: #Speichern der
                simulierten Anordnung als csv-Datei
84          writer = csv.writer(csvfile)
85          writer.writerows(zip(x_data,y_data,z_data))
86
87  path = "C:/Users/Maksym/Desktop/Python_Projekt" #Pfad des Programm-Ordners
88  Betriebszustand = 6 #Wahl des Betriebszustands
89  prevent_cut = False #Wahl ob abgeschnittene Borsten an den Raendern erlaubt sind
90
91  distributions = os.listdir(f"{path}/Verteilungen_Betriebszustand_{Betriebszustand}"
                )
92  dist = distributions[random.randint(0,23)] #Wahl einer zufaelligen Verteilung in
                dem Betriebszustand
93  with open(f"{path}/Verteilungen_Betriebszustand_{Betriebszustand}/{dist}", 'rb') as
                 f:
94      data_x, cdf_x, data_y, cdf_y = pickle.load(f)
95
96  simulate_brush(data_x, cdf_x, data_y, cdf_y)
```