



FACHHOCHSCHULE KIEL
University of Applied Sciences

Entwicklung eines ML-basierten Tools zur
Unterstützung der Bestimmung von
Kornverteilungen in elektronenmikroskopischen
Aufnahmen

Masterthesis
im Studienfach „Data Science“

vorgelegt von:

Max Brede

betreut von:

Prof. Dr. T. Schwörer

Prof. Dr. S. Doerfel

Kiel, im Juli 2022

Inhalt

Abbildungsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation und Problemstellung

In der materialwissenschaftlichen Betrachtung von Werkstücken und deren Eignung für gegebene Anwendungsgebiete ist eine möglichst detaillierte Beschreibung und Charakterisierung derer Eigenschaften eine zentrale Voraussetzung. Je genauer ein Werkstück in seinen Eigenschaften beschrieben werden kann, desto besser kann das Verhalten untersucht und vorhergesagt werden (Askeland, 1996).

Diese Eigenschaften können in verschiedenen Größenordnungen bestimmt und zur Beantwortung unterschiedlicher Fragen genutzt werden. Die erste mögliche Auflösung ist die Beschreibung der atomaren Zusammensetzung des Werkstücks sowie des Verhältnisses der verschiedenen Atome zueinander, sollte mehr als ein Element enthalten sein. Aussagen auf dieser Ebene können zum Beispiel Auskunft über elektrische und magnetische Eigenschaften des Werkstücks ermöglichen (Askeland, 1996). Als nächste Auflösungsstufe kann die Anordnung dieser Atome zueinander betrachtet werden. Diese so genannte Kristallstruktur kann Aussagen über zum Beispiel die Festigkeit eines Metalls liefern. In einem Werkstück können Kristalle sowohl mit unterschiedlicher Struktur und Zusammensetzung als auch in unterschiedlicher Ausrichtung zueinander vorkommen. Diese werden als Körner,

der ganze Verbund als Korn- oder Mikrostruktur bezeichnet. Die Orientierung der Kristalle zueinander und in Bezug zur Ausrichtung des Werkstückes zusammen mit der Größe und Form der Kristallite haben darüber hinaus einen großen Einfluss auf das mechanische Verhalten des Materials.

Die Charakterisierung dieser Mikrostruktur ist ein Teil der Aufgaben des Ausbildungsberufs des Metallographen. Diese Fachkräfte werden zum Beispiel in Stahlwerken eingesetzt, wo sie das Gefüge der im Material vorhandenen Kristalle durch Politur und Ätzung sichtbar machen. Diese Verfahren werden eingesetzt, um die Grenzen zwischen Körnern, die natürlicherweise Gitterfehler und damit Schwachpunkte des Materials darstellen, sichtbar zu machen. Da der Gitterverbund an diesen Grenzen schwächer ist, werden Atome hier leichter von Säuren ausgelöst, was zu einem mit einem Lichtmikroskop darstellbaren Höhenunterschied zwischen Korn und Korngrenze führt (“Gefüge (Werkstoffkunde),” 2021). Ein Beispiel für ein so behandeltes Werkstück ist in Abbildung ?? zu sehen.

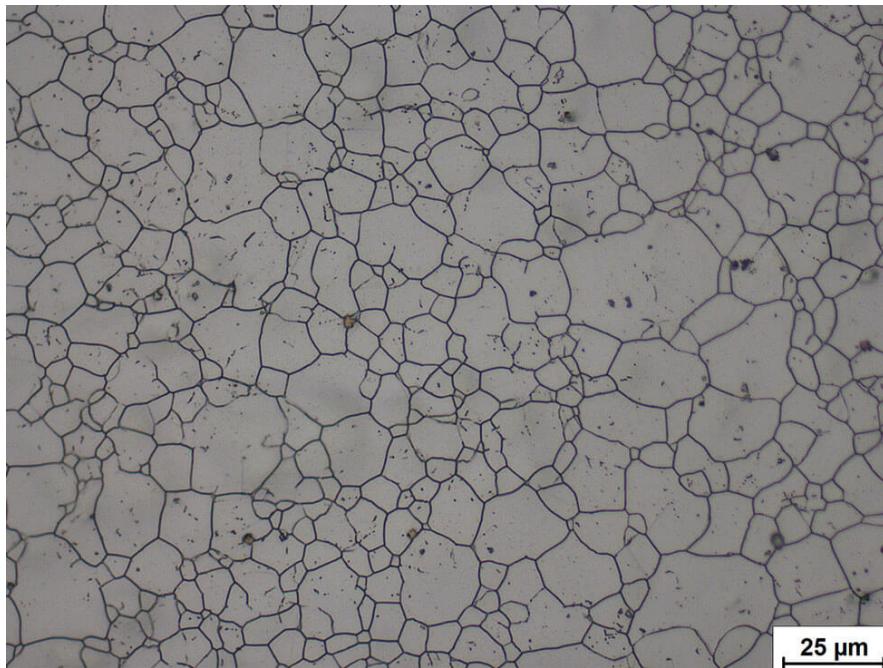


Abbildung 1.1: Lichtmikroskopische Aufnahme von poliertem und geätztem Austenitischem Stahl, Bild von *Metallographie von rostfreiem Stahl* (n.d.).

Um diese Aufnahmen der Schnittbilder zu nutzen, um zu einer quantitativen Be-

schreibung des Materials zu kommen, wurde traditionell und auch mitunter bis heute eins der vielen “Linienschnittverfahren” eingesetzt, wie es zum Beispiel bei Heyn (1903) beschrieben und als Richtlinienverfahren von der Standardisierungsorganisation ASTM international empfohlen wird (*Standard Test Methods for Determining Average Grain Size*, 2021). Neben diesem gibt es noch andere Ansätze zum Durchführen der Linienschnitte, alle diese Verfahren haben aber das folgende Vorgehen gemeinsam: Zuerst wird auf eine je nach Verfahren festgelegten Vorgehensweise eine Reihe von Linien in die Aufnahme vom Lichtmikroskop gezeichnet. Diese Linien werden dann genutzt, um die Körner auszuzählen und/oder zu vermessen, die von der Linie geschnitten werden. Die daraus resultierende Stichprobe an im Werkstück vorhandenen Korngrößen wird abschließend mithilfe einer passenden mathematischen Funktion (z.B. einer log-normalen Verteilungsfunktion) beschrieben, deren Parameter dann als Beschreibung der Kornstruktur genutzt werden.

Neben der verständlichen Ermüdung, die der Bearbeiter bei dieser Methode erfährt, ist die Genauigkeit der Methode grundsätzlich nur approximativ. Daher ist nicht verwunderlich, dass es in diesem Bereich schon Ansätze zur Automatisierung der Materialbeschreibung gibt. Hier wurde bereits über verschiedene Computervision-Methoden (z.B.: Ananyev et al., 2014; Heilbronner, 2000) und Machine-Learning-Ansätze (z.B.: DeCost et al., 2019; Dengiz et al., 2005) versucht, die Korngrenzen zu extrahieren oder auch die Materialien zu klassifizieren (Abouelatta, 2013).

Diese Verfahren funktionieren gut zur Segmentation von mit Lichtmikroskopie gewonnenen Kornbildern, die durch Ätzung gut darstellbare Korngrenzen aufweisen. Da mit dem Fortschritt in der Materialtechnik Körner auf immer kleineren Skalen vorliegen, gewinnt die Anwendung höher auflösender mikroskopischer Verfahren aber zunehmend an Wichtigkeit. Insbesondere bei der Betrachtungen von Materialien in dünnen Schichten, das heißt in einer Dicke im Mikro- oder Nanometerbereich, wird eine deutlich höhere Vergrößerung relevant. Der hier nötige

ge Übergang zur Elektronenmikroskopie stellt die automatische Auswertung der Schnittbilder vor neue Probleme. Zwar können bei ätzbaren Oberflächen die oben genannten automatischen Auswertungsmethoden weiter eingesetzt werden, bei besonders kleinen Körnern führt die Ätzung aber zu einem dermaßen großen Angriff der Kornstruktur, dass eine Identifikation und Detektion der Grenzen geradezu unmöglich wird. Stattdessen werden die Körner über ihre je nach kristallographischer Orientierung unterschiedlich starke Beugung der Elektronen im Rückstreubild in unterschiedlichen Graustufen dargestellt. Diese Graustufenbilder machen das automatisierte Identifizieren der Korgrenzen ungemein schwieriger. Beispiele für solche Aufnahmen sind in Abbildung ?? zu sehen.

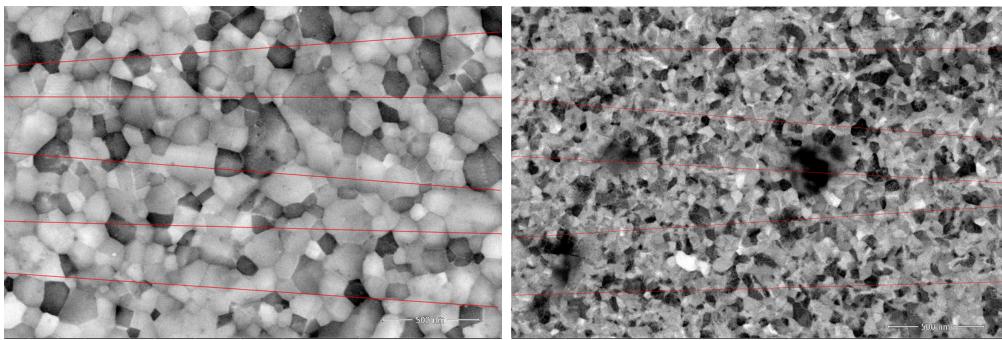


Abbildung 1.2: Elektronenmikroskopische Aufnahmen von Werkstücken. In rot sind die Linien eingezeichnet, die zur Bestimmung der Kornverteilung mit Hilfe eines Linienschnittverfahrens eingesetzt wurden. Das Werkstück links weist wenig Artefakte und klar zu erkennende Kornflächen auf. Rechts ist ein Werkstück abgebildet, dessen Körner weniger deutlich zu erkennen sind, das Gradienten von Grautönen in einem Korn aufweist und dessen Aufnahme deutliche Bildartefakte produziert hat.

Zusätzlich stören Kristalldefekte, Oberflächenartefakte und Spannungen im Material die Auswertung, da sie zu überlagernden Kontrastartefakten führen (Abbildung ?? rechts). Mit Training sind menschliche Bearbeiter zwar weiter in der Lage, Körner und ihre Grenzen zu detektieren und mit Linienschnittverfahren auszuwerten, bestehende Ansätze zur automatischen Detektion von Korgrenzen scheitern aber.

Im Bereich der Mineral-Korn-Erkennung wurden aber bereits erfolgreich vielversprechende Ansätze berichtet (Latif et al., 2022; Maitre et al., 2019). Diese neu-

en Ansätze haben gemeinsam, dass sie statt Korngrenzen Flächen der Körner auszumachen versuchen. Dabei werden Methoden der *Superpixel Segmentation* eingesetzt, bei denen versucht wird, ein Bild in semantisch ähnliche Gruppen von Pixeln zu segmentieren. Das Aufteilen eines Bildes in diese Gruppen von ähnlichen Pixeln oder auch *Superpixel* ist ein Reduzieren der Bildkomplexität für folgende Analyseschritte (Wang et al., 2017). Eine Anwendung von Superpixel-basierten Ansätzen zur Segmentation von mikroskopischen Aufnahmen von Metallstrukturen sind entweder zu hochauflösend (Akers et al., 2021), zu niedrig auflösend (Kim et al., 2020) oder auf andere Arten von Mineralien (DeCost et al., 2019; Latif et al., 2022) oder nur auf Teile der Aufnahme bezogen (li et al., 2020).

Da die Vorbereitung und das spezifische ausgewertete Material stark die Art und Qualität der Bilder beeinflusst, lassen sich diese Ergebnisse nicht direkt auf Aufnahmen von dünnen Schichten übertragen - die Ansätze scheinen aber vielversprechend. Die Auswertung der Größenverteilung möglichst aller Körner über die gesamte elektronenmikroskopische Aufnahme dünner Schichten ist jedoch bisher noch nicht gelöst.

Die vorliegende Masterarbeit soll an diesem Punkt ansetzen und versuchen, auf Basis von Superpixel-Verfahren möglichst alle Körner in einer elektronenmikroskopischen Aufnahme zu erkennen und diese zu vermessen.

1.2 Unternehmensvorstellung

Die Arbeit wird in enger Abstimmung mit dem Institut für Materialphysik der Georg-August-Universität Göttingen umgesetzt und basiert auf dort aufgenommenen Kornbildern. Die Georg-August-Universität Göttingen wurde 1734 gegründet und zählt mit ihren 29.167 Studierenden im WiSe 21/22 (Öffentlichkeitsarbeit, n.d.-b) und den 5.165 Beschäftigten im Jahr 2021 (Öffentlichkeitsarbeit, n.d.-a) zu den größten Hochschulen Deutschlands. Am Lehrstuhl für Materialphysik wird regelmäßig das Verhalten von Materialien in dünnen Schichten untersucht, de-

ren Oberflächen dazu elektronenmikroskopisch aufgenommen und händisch per Linienschnitt ausgewertet werden.

1.3 Zielsetzung

Das Ziel dieser Abschlussarbeit ist die Entwicklung eines Tools, das die Auswertung von Kornbildern so weit es geht unterstützt. Als erster Schritt ist dafür eine Implementierung mit grafischer Oberfläche nötig, die das Einlesen und Verarbeiten von elektronenmikroskopischen Aufnahmen mit dahinter stehendem Datenmodell unterstützt. Dabei soll die Verarbeitung sowohl aus dem Vorverarbeiten als auch der Kornerkennung und -vermessung bestehen. Die Nutzbarkeit des Tools soll durch Angehörige des Instituts getestet und dessen Nutzen überprüft werden. Die bei dieser Überprüfung entstehenden Wünsche an Verbesserungen und Anpassungen des Programms sollen so weit wie möglich umgesetzt werden. Dabei ist insbesondere auch wichtig, das Tool möglichst nachnutzbar zu gestalten und eventuell nötige Erweiterungen so sehr zu vereinfachen wie möglich.

1.4 Aufbau der Arbeit

Im Folgenden wird zuerst auf die Grundlagen der verwendeten Filter- und Auswertungsmethoden und deren Funktionsprinzipien eingegangen. Im darauf folgenden Kapitel wird das Anforderungsprofil der Anwendung formuliert, gefolgt von einer Beschreibung der Entwicklung des Tools. Nutzbarkeit und Nutzen des Tools werden im vorletzten Kapitel evaluiert, wonach eine Schlussbetrachtung sowie ein Ausblick auf im Anschluss weiter zu untersuchende Ansätze folgt.

Kapitel 2

Grundlagen

Dieses Kapitel beschäftigt sich zuerst mit den zur Bildvorbereitung betrachteten und verwendeten Algorithmen. Darauf folgt eine Beschreibung der getesteten Ansätze zur Flächen-Gruppierung, um dann dazu überzugehen, die Implementation des Körner-Tools zu beschreiben. Im Anschluss werden die notwendigen Ansätze zur Extraktion der Kornstrukturen und ihrer Automatisierung erklärt.

2.1 Bildvorverarbeitung

Im Folgenden werden die einzusetzenden Vorverarbeitungsalgorithmen beschrieben. Da das Augenmerk dieser Arbeit insbesondere auf der Erkennung von Körnern und weniger auf dem Vorverarbeiten liegen soll, stehen an dieser Stelle die bereits in der Kornvermessung eingesetzte Algorithmen im Vordergrund der Betrachtung. Alle diese Vorverarbeitungs-Schritte verfolgen zwei Ziele: Zum einen soll versucht werden, die Bildartefakte so gut wie möglich zu entfernen, zum anderen sollen die Grautöne innerhalb eines Korns so gut angeglichen werden wie möglich. Die Reihenfolge, in der die Algorithmen angewandt werden, ist außerdem von entscheidender Bedeutung, ein Histogramm-Equalizer führt nach Anwendung eines Gauss-Filters zu einem deutlich anderen Effekt als davor. Auf diesen Punkt wird aber in der Beschreibung der Implementation noch umfangreicher eingegan-

gen.

Gauss-Filter

Gauss-Filter werden häufig in Anwendungsbereichen mit multivariaten Untersuchungsgegenständen eingesetzt, so zum Beispiel in EEG- und fMRT-Analysen (Harishvijey & Benedict Raja, 2022; Wink & Roerdink, 2004), in Zeitreihenanalysen (Kitagawa, 1994) und nicht zuletzt weit verbreitet in der Bildverarbeitung (Basu, 2002).

Bei diesem Verfahren wird das Bild auf Basis einer zweidimensionalen Gauss-Dichte-Funktion gefaltet, wodurch jedes Pixel durch die gewichtete Summe der umliegenden Pixel ersetzt wird. Die dabei als Gewichtung eingesetzte, zweidimensionale Dichte-Funktion wird auch als der *Kernel* bezeichnet. In Abb. ?? ist beispielhaft das Ergebnis eines Gauss-Filters zu sehen. In der Abbildung wird dessen primäre Funktion deutlich. Die harten Kanten des Bildes werden zunehmend weicher mit größerer Kernel-Einstellung, außerdem werden die einzelnen, herausstechenden Pixel im unteren Viertel der Abbildung ihrem Umfeld angeglichen und stechen weniger deutlich hervor. Dieser auch als Glättungs-Filter bezeichnete Vorverarbeitungs-Algorithmus wird mit dem Ziel eingesetzt, Pixel mit extremen Werten an ihr direktes Umfeld anzugeleichen und damit Bildartefakte zu entfernen.

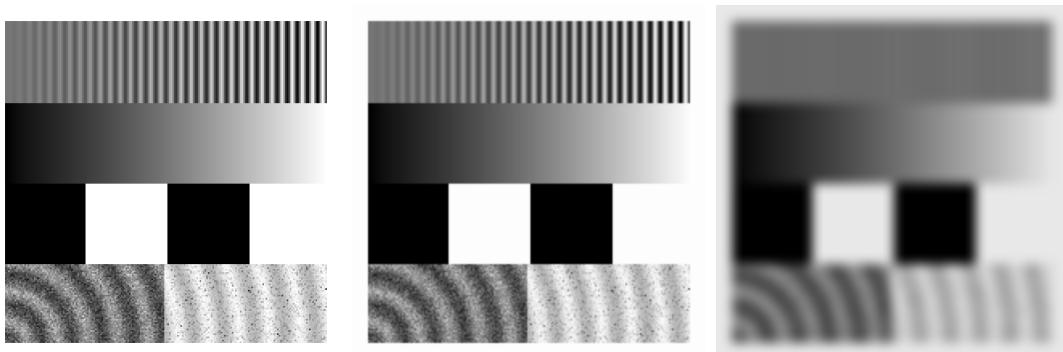


Abbildung 2.1: Beispiel eines Gauss-Filters. Links ist das Bild vor dem Filtern, in der Mitte nach dem Filtern mit einem Gaußkernel mit Durchmesser von 15 Pixeln (1% der Bildkantenlänge) und rechts einem mit Durchmesser von 151 Pixeln (~10% der Kantenlänge) zu sehen.

Histogramm-Equalizer

Ein Histogramm-Equalizer gleicht die Helligkeitswerte eines Bildes so an, dass die kumulative Häufigkeitsverteilung der Helligkeitswerte linear ansteigt (Prince, 2012). Je nach Bild kann es also passieren, dass Teile der Helligkeitswert-Verteilung gestreckt, andere Teile gestaucht werden. Diese nicht-lineare Transformation führt dazu, dass bei nicht-linearen Helligkeitsverteilungen, wie zum Beispiel im Falle eines Bildes mit vielen ähnlichen Grautönen, vorher kleine Unterschiede deutlicher akzentuiert werden, während stark unterschiedliche Helligkeitswerte angenähert werden können. In Abb. ?? ist der Effekt des Histogramm-Equalizers beispielhaft dargestellt. Der Farbverlauf im unteren Viertel der oberen Hälfte der Abbildung wurde verdunkelt, genau wie die erste, auf dem Cosinus basierenden Zeile mit Ausnahme einer weißen Spalte. Unter den geometrischen Muster sind die Helligkeitsverteilungen vor und nach der Anwendung des Filters zu sehen, die Linearisierung der Helligkeitsverteilung ist deutlich zu erkennen.

Non-Local-Mean-Denoiser

Ein Non-Local-Mean Denoiser versucht, Bildrauschen durch das Ersetzen der Grauwerte ähnlich grauer, nicht unbedingt nebeneinander liegender Pixel durch deren Grauwert-Mittelwert zu reduzieren. Dazu wird in einem angegebenen Suchfenster für alle ähnlich grauen Pixel der Mittelwert der Grauwerte als neuer Grauwert gesetzt. Die Pixel müssen dabei explizit nicht lokal nebeneinander liegen, sondern nur einen ähnlichen Grauwert aufweisen (Buades et al., 2011).

Der Filter wird dabei über eine Filter-Stärke, die Größe des Suchfensters und die Größe des “Templates”, also der Menge an Gruppen als gleich gezählter Pixel, bestimmt.

In Abb. ?? ist der Effekt des Denoisers zu sehen, das Bildrauschen im unteren Viertel wird deutlich reduziert, auch wenn das Muster dadurch leicht undeutlich wird. Der Denoiser stößt jedoch an seine Grenzen bei der Entfernung nicht-normalen Rauschens, wie zum Beispiel in der rechten Hälfte der letzten Zeile zu sehen

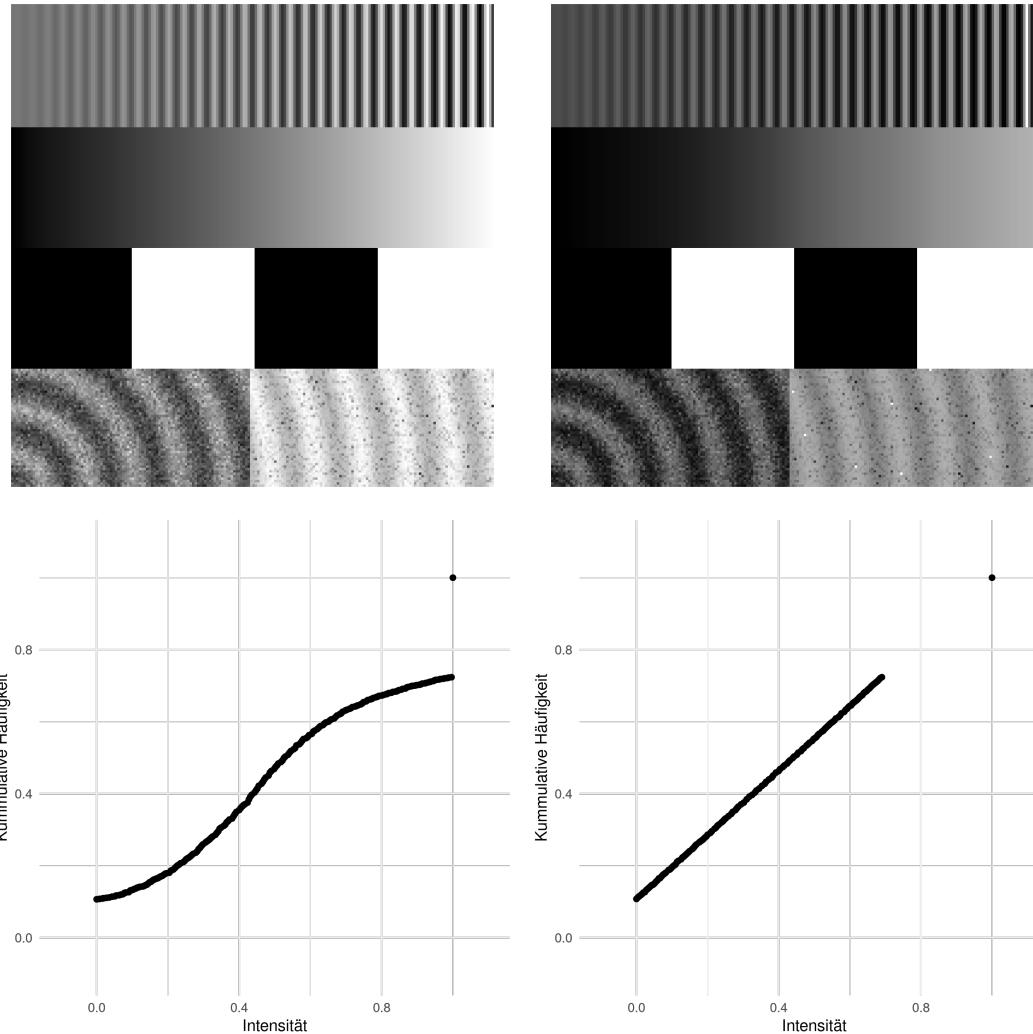


Abbildung 2.2: Beispiel für die Anwendung eines Histogramm-Equalizers. Links ist das Bild vor, rechts nach der Anpassung der Helligkeitsverteilung zu sehen. Der Farbverlauf wurde verdunkelt, genau wie die erste, auf dem Cosinus basierenden Zeile mit Ausnahme einer weißen Spalte. Unten sind die Helligkeitsverteilungen abgebildet, die Linearisierung ist deutlich zu erkennen.

ist. Dort wurde im Gegensatz zur linken Hälfte kein normalverteiltes, sondern F-verteiltes Rauschen auf das Bild addiert. Durch die schiefe Form der F-Verteilung sind so deutlich dunklere Pixel entstanden, die als Ausreißer nicht vom Non-Local-Mean Denoiser entfernt werden, da sie nicht ähnlich genug zu den restlichen Pixeln sind.

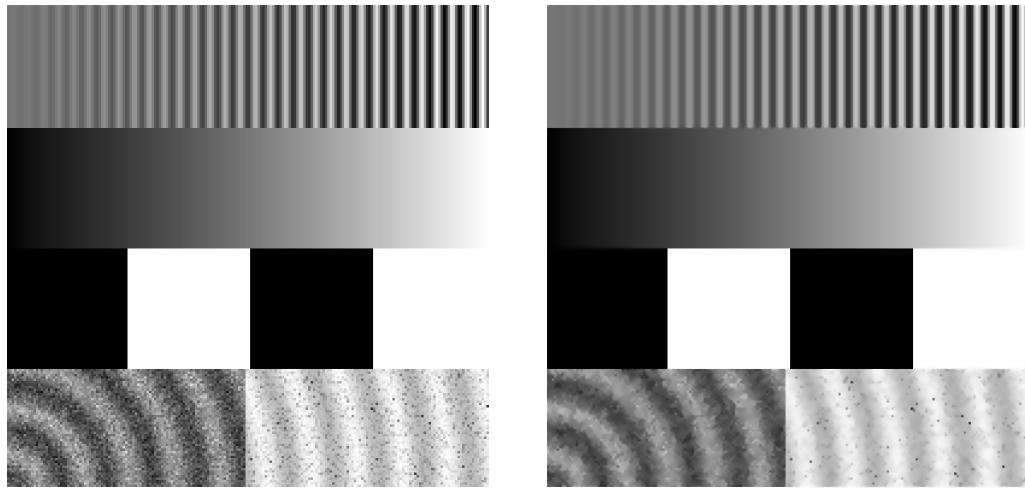


Abbildung 2.3: Beispiel für die Anwendung eines Non-Local_Mean Denoisers. Links ist das Bild, rechts nach dem Denoising mit einer Template-Größe von 7, einem Suchfenster von 21 zu sehen, der Filter wurde verhältnismäßig stark gewichtet um den Einfluss deutlich zu machen. Das Rauschen wurde zwar reduziert, die konzentrischen Kreise aber auch unschärfer.

Savitzky-Golay-Filter

Ein Savitzky-Golay-Filter wird in allen Bereichen der Signalverarbeitung zur Signalglättung eingesetzt. Hierzu wird eine Fenstergröße und ein Grad für das zu nutzende Polynom festgelegt. Das Verfahren fitted iterativ in je einem Abschnitt in der angegebenen Fenstergröße über die gesamte Zeitreihe eine polynomiale Regression des angegebenen Grades. Das Mittel der Vorhersagen dieser Regressionen über die Datenreihe wird dann zurückgegeben. Je nach Größe des Fensters und der Höhe des Polynoms wird dadurch unterschiedlich stark geglättet.

In Abb. ?? ist der Effekt je eines horizontalen und vertikalen Savitzky-Golay-Filters mit jeweils einer Fenstergröße von 51 Pixeln und einem Polygon der 1.,

5. und 11. Ordnung beispielhaft dargestellt. Wie zu sehen ist, schlägt sich der Filter bei Polynomen niedrigeren Grades eher als Unschärfe nieder, je höher die Grade sind, desto mehr Varianz im Signal wird nicht entfernt. Je höher der Grad des Polynoms für den Filter also gewählt wird, desto weniger stark wird das Bild durch den Filter verändert - desto weniger Rauschen wird aber auch entfernt.

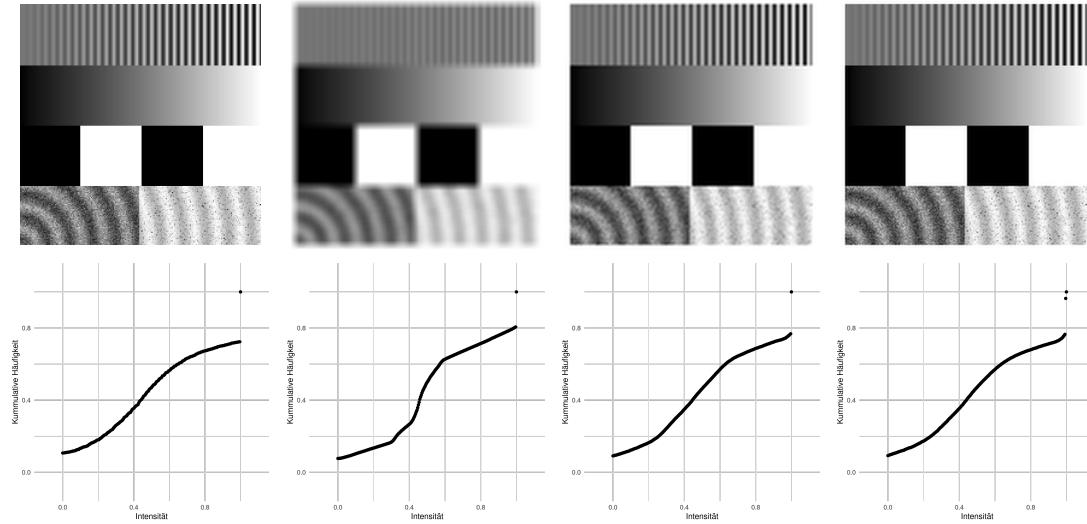


Abbildung 2.4: Beispielhafte Darstellung eines Savitzky-Golay-Filters. Links ist das Bild vor, rechts nach der Glättung der Helligkeitsverteilung mit einem Polynom 1., 5. und 11. Grades zu sehen. Unten sind die Helligkeitsverteilungen abgebildet.

2.2 Extraktion der Kornstrukturen

Der folgende Abschnitt beschäftigt sich mit den möglichen Ansätzen zur Erkennung und Extraktion der Kornflächen.

DBSCAN

Der ursprünglich zur Gruppierung großer Datenbanken entwickelte *Density-based Spatial Clustering of Applications with Noise* (DBSCAN) Algorithmus (Ester et al., 1998) konnte bereits erfolgreich zur Superpixel-Segmentierung in Echtzeit eingesetzt werden (Shen et al., 2016).

Der DBSCAN-Cluster-Algorithmus überprüft für jeden einzelnen Datenpunkt, wie viele Datenpunkte in einem vorgegebenen Radius ϵ um den betrachteten Punkt liegen. Das Verfahren sortiert so Datenpunkt für Datenpunkt lokal über den Radius verknüpfte Einträge zusammen, die als zu einem Cluster zugehörig erklärt werden, sobald eine angegebene Mindestanzahl an Samples erreicht ist.

Durch das iterative Vorgehen bei diesem Verfahren ist die Form der erkannten Cluster nicht festgelegt. Dadurch können auch komplexe Formen im Feature-Raum abgebildet werden. Ein weiterer Vorteil ist die geringe Anfälligkeit für Ausreißer, da sich nicht im Radius um andere Punkte befindliche Datenpunkte einfach keinem Cluster zu sortiert werden.

In Abb. ?? ist der Einfluss verschieden großer Radien und Mindest-Clustergrößen dargestellt. Die Werte wurden vor dem Clustern so skaliert, dass x- und y-Achse sowie die Farbwerte des ursprünglichen Bilds zwischen 0 und 100 liegen.

SLIC

Beim als Superpixel-Segmentierungsansatz weit verbreiteten *Simple Linear Iterative Clustering* (SLIC) (Achanta et al., 2012) wird ein k-Means basiertes Verfahren genutzt, um näherungsweise gleich große Superpixel im Bild zu detektieren. Dabei wird sowohl für den Farb- als auch den Pixelraum je eine euklidische Distanz aller Pixel zueinander berechnet, die dann gewichtet aufsummiert wird. Dabei wird in der Arbeit von Achanta et al. (2012) die räumliche Distanz gewichtet, wodurch mit hohen Werten des Parameters m die räumliche Nähe, mit niedrigen Werten die farbliche Nähe stärker gewichtet wird.

Der weitere Algorithmus ist im Prinzip der des normalen k-Means Clustering, das heißt Cluster-Kerne werden definiert (bei SLIC in regelmäßigen Abständen), Pixel werden den nächsten Clustern zugeordnet und die Zentroide dann auf die Zentroide der neuen Cluster verschoben. Im Gegensatz zum regulären k-Means wird bei SLIC aber nicht im gesamten Datenraum nach möglichen Cluster-Angehörigen gesucht, sondern nur im doppelten Suchraum der vorgegebenen Clustergröße. Achan-

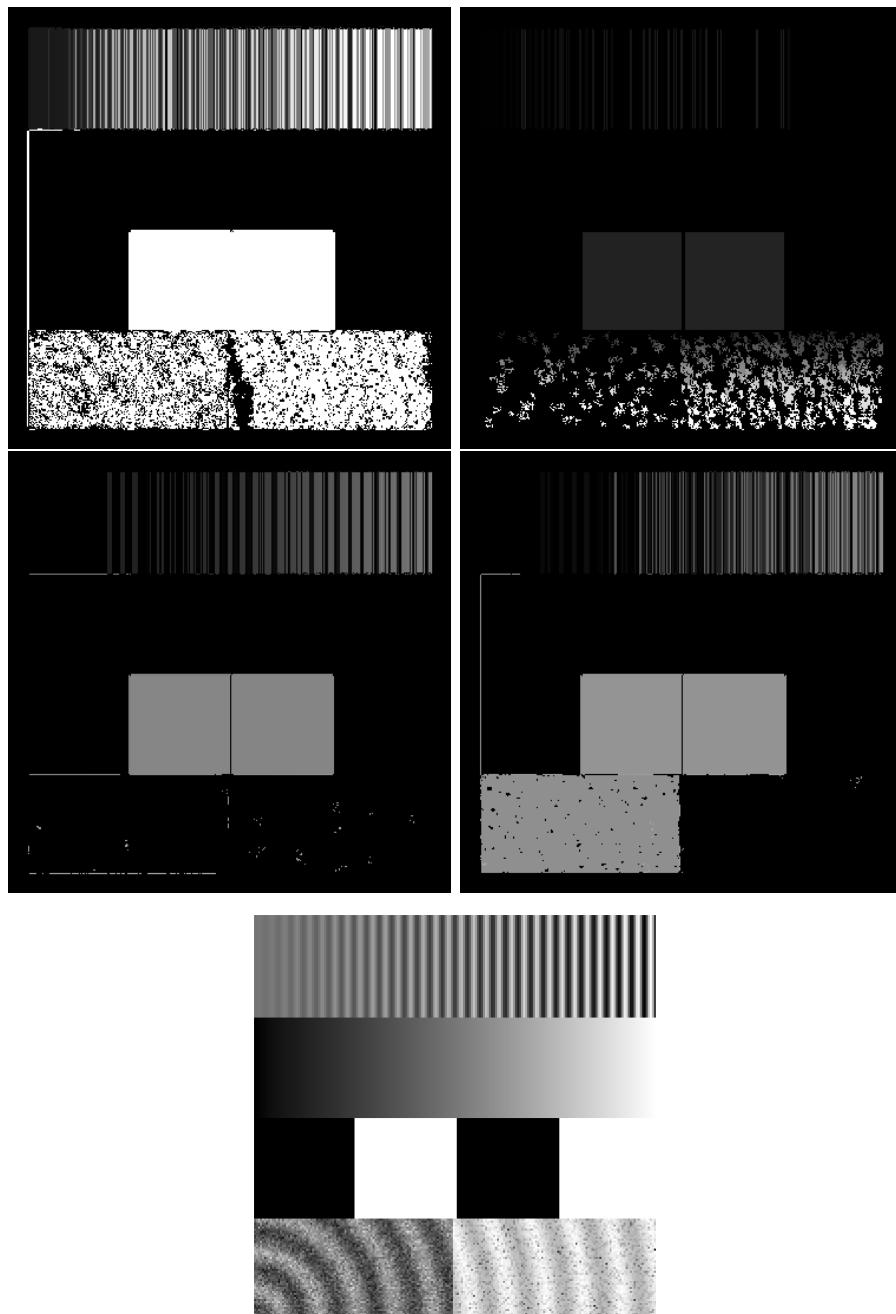


Abbildung 2.5: Beispiel für Ergebnisse des DBSCAN-Algorithmus. Die Graphen stellen dasselbe Muster dar, wie zur Illustration der Vorverarbeitungsmethoden genutzt wurde. In der ersten Zeile wurde ϵ auf 1, in der zweiten Zeile auf 2 gesetzt. Oben wurde die Mindestzahl an Pixeln auf 5, unten auf 10 gesetzt. Die Werte sind so skaliert, x- und y-Achse sowie die Grauwerte des ursprünglichen Bilds zwischen 0 und 100 liegen. Darunter ist das ursprüngliche Bild dargestellt.

ta et al. (2012) beschreiben außerdem eine adaptive Version des SLIC, bei dem die Cluster-Größe (und Anzahl) und die Gewichtung von räumlicher und farblicher Distanz clusterweise angepasst wird. Dazu wird in jeder Iteration an der maximalen Distanz der vorhergehenden Iteration normalisiert. In Abb. ?? ist ein Beispiel für den ASLIC-Algorithmus mit unterschiedlichen m -Gewichten und anfänglicher Cluster-Anzahl zu sehen.

Normalized Cuts

Beim *Normalized Cuts* Verfahren zur Extraktion von Superpixeln (Shi & Malik, 2000) wird das Bild in einen Graphen übersetzt, bei dem jeder Pixel durch einen Knoten repräsentiert wird. Jeder dieser Knoten wird dann mit jedem anderen Knoten über eine Kante verbunden, die über ein Maß gewichtet wird, das die Wahrscheinlichkeit der “Zugehörigkeit zu einem Objekt” (Shi & Malik, 2000) ausdrückt. Im Original-Paper wird die Wahrscheinlichkeit durch eine gewichtete Exponential-Funktion mit den Kontrastwerten und den geometrischen Distanzen im Exponenten ausgedrückt. Für beide Werte ist je eine Gewichtung vorgesehen, die angepasst werden kann - außerdem ein maximal zu berücksichtigender Radius für den geometrischen Abstand.

Unabhängig von der genauen Gewichtung der Kanten wird anschließend derjenige Schnitt gesucht, der die geringste Summe an Kantengewichten “durchtrennt”. Dieser *Normalized Cut* wird so bestimmt, dass die durch den Schnitt getrennten Gruppen möglichst unterschiedlich und die verbleibenden Gruppen in sich möglichst homogen sind. Die Lösung dieses Problems für ein ganzes Bild kann sehr aufwendig sein, weswegen zur Optimierung nur im geometrischen Umfeld der betrachteten Knoten gesucht wird. Statt die Kanten mit der Wahrscheinlichkeits-Formel aus dem Papier von Shi & Malik (2000) zu gewichten, können auch beliebige andere Gewichte gesetzt werden. So ist in der Implementation von NCuts, die in der Dokumentation des scikit-image-Python-Moduls(*Normalized Cut — Skimage V0.20.0.Dev0 Docs*, n.d.) präsentiert wird, der Graph mit den Ergebnissen

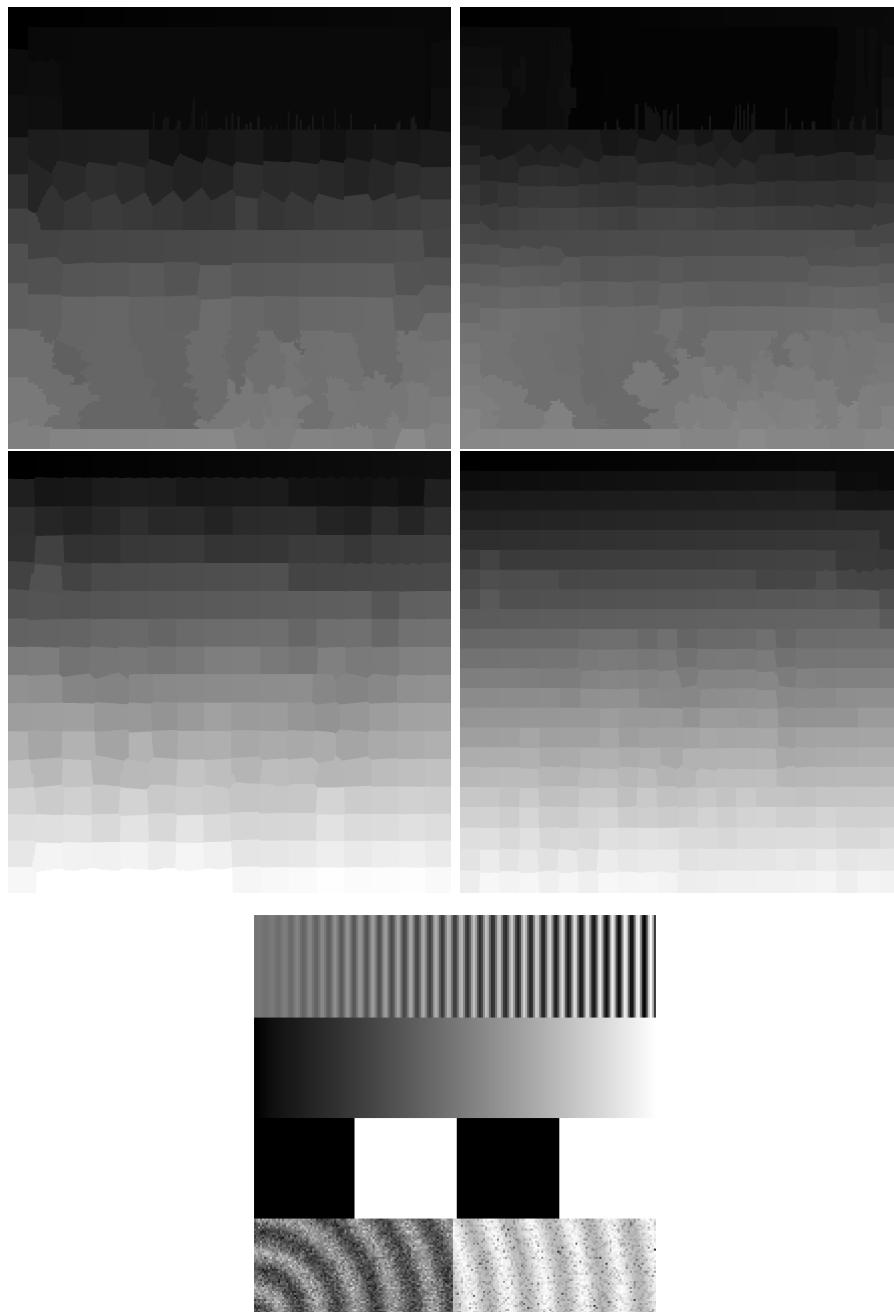


Abbildung 2.6: Beispiel für eine Superpixel-Segmentierung mit SLIC. Dargestellt sind die Cluster-Label als Grauschattierung. Links ist die Segmentierung mit 250 Start-Cluster-Zentren, rechts mit 500 dargestellt, oben je mit $.05$, unten mit 2 als Wert für m zu sehen. In beiden Beispielen wurde die Adaptive Version des SLIC-Algorithmus wie bei Achanta et al. (2012) beschrieben eingesetzt. Unter den verarbeiteten Bildern ist das ursprüngliche Bild dargestellt.

einer SLIC-Voranalyse gewichtet. Vor allem bei großen Bildern wird die von Shi & Malik (2000) präsentierte Berechnung nahezu unmöglich, ohne große Optimierung ist allein eine Ergebnis-Matrix mit mindestens $\frac{N_{Pixel}^2}{2}$ Einträgen nötig. Da SLIC aber in gewisser Form auch eine „Wahrscheinlichkeit der Zugehörigkeit zum selben Objekt“ abdeckt und besonders bei großen Bildern die Findung einer Lösung für den optimalen Schnitt aufwendig wird, ist die Vorverarbeitung in diesem Sinne eine sinnvolle Optimierungs-Option. Bei der Findung der optimalen Lösung können eine Reihe von Einstellungen gesetzt werden. Neben den Einstellungen für die SLIC-Vorsegmentierung kann die Anzahl der gesuchten Schnitte, die maximale Distanz zwischen zwei Farben, um als ähnlich erkannt zu werden, und zuletzt der minimale Wert, den ein Schnitt haben darf, um durchgeführt zu werden.

In Abb. ?? sind Beispiele für NCuts-Anwendungen mit unterschiedlichen Einstellungen zu sehen.

Hyperband und Bayesianische Optimierung

Die Performance von Verfahren des maschinellen Lernens, zu denen auch Cluster-Verfahren wie DBSCAN und k-Means zählen, ist stark abhängig von der Wahl der richtigen Einstellungen des Verfahrens (z.B. Snoek et al., 2012). Rodriguez et al. (2019) konnten zum Beispiel zeigen, dass die zufällige Auswahl von Einstellungen in einer besseren Performance bei ihrem Test gängiger Clusterverfahren resultierte, als die Verfahren mit ihren Grundeinstellungen anzuwenden. Die zufällige Auswahl dieser auch *Hyperparameter* genannten Einstellungen ist ein gängiges, auch *random search* genanntes Verfahren zum Tuning, also der Verbesserung der Performance von Anwendungen im Deep Learning. Hier ist die Auswahl geeigneter Einstellungen für das Modell insbesondere wichtig, aber auch zeitaufwändig, da die möglichen Stellschrauben für ein genügend komplexes Modell so viele mögliche Kombinationen bieten können. Bergstra et al. (2011) nennen die geeignete Wahl an Parametern deswegen auch “*more of an art than a science.*”. Das Problem wird zusätzlich vergrößert, da Hyperparameter-Optimierungen in den meisten Fällen

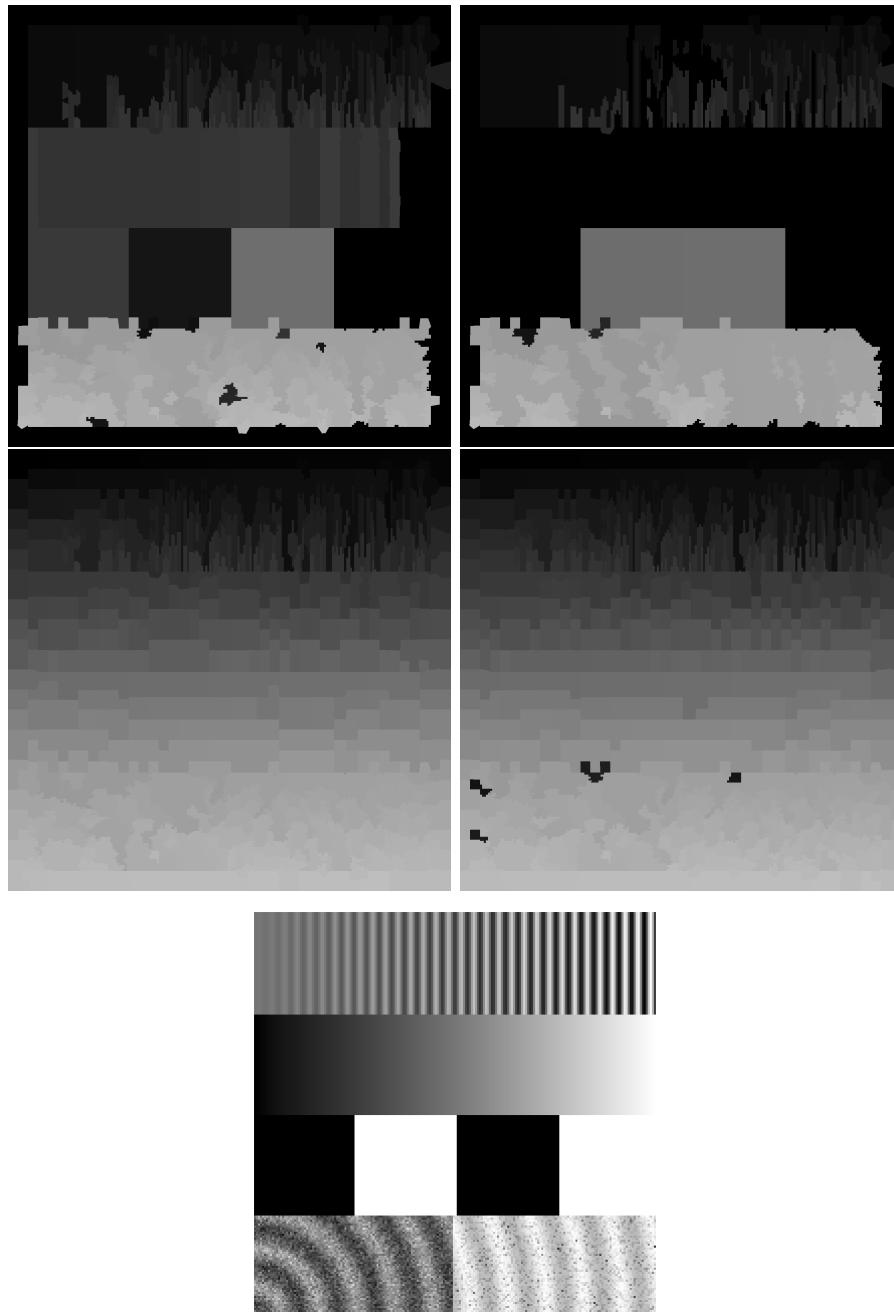


Abbildung 2.7: Beispiel für eine Superpixel-Segmentierung mit NCuts mit vorgeschalteter SLIC-Segmentierung. Die SLIC-Segmentierung wurde mit .05 als geometrisches Gewicht und 2000 Start-Clustern durchgeführt. In der oberen Zeile wurde der minimale Schnittwert auf .001, in der unteren Zeile auf .5 gesetzt. Die linken Bilder wurden mit 10, die rechten mit 100 Schnitten geteilt. Zusätzlich sind die Bilder links oben und recht oben mit einer maximalen Farbdistanz von 20, die anderen beiden mit je 100 erstellt. Unter den verarbeiteten Bildern ist das ursprüngliche Bild dargestellt.

len Black-Box Optimierungen sind. Das heißt, dass keine verlässliche Annahmen über die Form und Art der Zielfunktion (zum Beispiel in Form eines Gütemaßes der Modellvorhersage) getroffen werden kann, außer durch die systematische Eingabe und Aufzeichnung der Ausgabe ausgewählter Hyperparameter. Neben dem Random Search sind in diesem Forschungsbereich auch eine Reihe von wesentlich effizienteren Algorithmen zur Auswahl von Modell-Einstellungen publiziert worden (Eggensperger et al., 2021 für eine erste Übersicht). Ein Beispiel für eine solche Vorgehensweise zum Optimieren der Modelleinstellungen mit dem Ziel, die optimale Vorhersage zu treffen, ist die von Falkner et al. (2018) vorgeschlagene *Bayesian Optimization and Hyperband* (BOHB)-Methode. Diese Kombination von Hyperband- und Bayesianischen Hyperparameter-Optimierungslösungen stellt eine effizientere Optimierungs-Alternative zur Random Search dar. Dieser Effizienzgewinn resultiert aus der Nutzung von Zwischenergebnissen zur Verbesserung von Optimierungszeit und Effizienz im Gegensatz zur komplett Zwischenergebnis-agnostischen Black Box-Methode Random Search.

BOHB arbeitet dabei mit einer Kombination eines *Tree-structured Parzen Estimators* (Bergstra et al., 2011) und eines durch dieses Bayesianische Schätzverfahren informierten Hyperband-Verfahrens (Li et al., 2017). Beim Tree-structured Parzen Estimators (TPE) wird der Parameter-Raum als Entscheidungsbaum formalisiert. Aus diesem Baum werden dann die Äste ausgewählt, für die ein möglichst minimaler Wert der Zielfunktion mit dem nicht-parametrischen Parzen-Fenster-Kerndichteschätzer vorhergesagt wird. Die so ausgewählte Parameter-Kombination wird genutzt, um das Modell zu trainieren und die Performance des Ergebnisses genutzt, um die Dichteschätzung zu aktualisieren. An dem Punkt des nötigen Modelltrainings zur Aktualisierung der Performance-Prognose setzt der Hyperband-Teil des BOHB-Verfahrens an. Statt für jede zu testende Parameterkombination das ganze Modell zu trainieren, führen Li et al. (2017) ein Budget als weiteren Parameter ein. Dieses Budget ist so definiert, dass geringere Werte zu einem Modell führen, das zeitlich und hinsichtlich des Rechenaufwands weniger aufwendig ist, dabei aber eine Idee für die Performance eines

Modells auf Basis dieser Werte gibt. Bei Modellen aus dem Deep Learning kann das Budget zum Beispiel als die Anzahl der Trainingsepochen operationalisiert werden, bei Cluster-Algorithmen zum Beispiel als eine Beschränkung der jeweils zur Distanzberechnung genutzten Datenpunkte. Beim Hyperband-Verfahren wird also auf niedrigem Budget zuerst eine Reihe an schnellen, “nicht vollständigen” Tests durchgeführt, deren Settings im Original zufällig ausgewählt wurden. Die Settings der besten n dieser Tests wird im Folgenden genutzt, um eine Reihe höher budgetierter Tests durchzuführen. Diese fortschreitende Verkleinerung des Suchraums auf größerem Budget wird fortgeführt, bis eine kleine Anzahl von Modellen bei vollem Budget mit den vielversprechendsten Parametern getestet wird. Im Gegensatz zum Original nutzt nun BOHB nicht Zufallsauswahl zur Auswahl der jeweiligen Parameter pro Budget, sondern die mit dem Parzen-Kernel als optimal vorhergesagten Parameter - benötigt dafür aber nicht wie die reine Bayesianische Optimierung das gesamte Budget für jeden Test. Um eine Über-Betonung eines Teils des Parameter-Raums zu verhindern wird außerdem ein festgelegter Anteil der im Hyperband-Teil versuchten Kombinationen wie im ursprünglich beschriebenen einfachen Hyperband-Verfahren zufällig gezogen.

In Abb. ?? ist die Performance von BOHB im Vergleich zu Random Search, Hyperband und TPE zu sehen. In Biedenkapp & Hutter (2018) führen einige der Autoren von BOHB als Grenzen des Algorithmus zwei Einschränkungen ein. Zum einen muss die Optimierungsaufgabe ein sinnvolles Budget möglich machen. Sollte kein sinnvolles Budget zu definieren sein, wäre BOHB im schlimmsten Fall um die Anzahl der Halbierungen schlechter als die einfache TPE-Optimierung. Zum anderen kann ein eher unrealistischer Fall konstruiert werden, in dem das globale Optimum der Zielfunktion in einem sonst schlecht performenden Bereich der Hyperparameter-Kombinationen liegt. In diesem Fall konvergiert BOHB dank des festen Anteils zufälliger Parameterkombinationen aber auch zur selben Lösung wie Hyperband allein, braucht aber um den Kehrwert des Anteils an zufälligen Kombinationen länger als Hyperband allein.

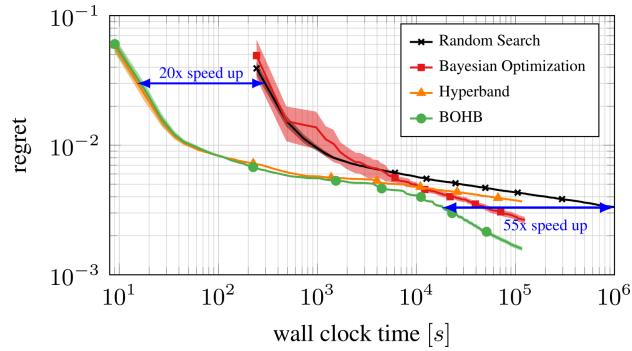


Abbildung 2.8: Performance von BOHB im Vergleich mit anderen Hyperparameter-Optimierungs-Methoden. Abbildung aus Biedenkapp & Hutter (2018). Zu sehen ist der Vergleich von BOHB mit Random Search, dem TPE allein und Hyperband allein. BOHB ist durch den Einsatz von Hyperband wesentlich schneller (x-Achse) zu Beginn der Optimierung als Random Search und TPE und dabei deutlich besser im Ergebnis als alle anderen Verfahren (Ziel-Funktion auf der y-Achse).

Kolmogorov-Smirnov Teststatistik

Die Kolmogorov-Smirnov (KS) Teststatistik ist die Grundlage für den nonparametrischen KS-Test. Dieses Verfahren ist dafür konstruiert, eine Stichprobe auf die Verträglichkeit mit einer gegebenen Verteilung zu überprüfen. Allerdings wird das Verfahren nur noch selten als Voraussetzungstest eingesetzt, da es im Vergleich zu anderen Verfahren eine relativ geringe statistische Power aufweist (Stephens, 1974). Die Teststatistik D ist als die maximale vertikale Distanz zweier Verteilungsfunktionen definiert und ein sehr praktisches Werkzeug zum Anpassen einer geschätzten Verteilung, da es im Gegensatz zu zum Beispiel der χ^2 -Statistik die gesamte beobachtete Häufigkeitsverteilung berücksichtigt und gleichzeitig die Ähnlichkeit zweier Verteilungen in einer Kennzahl ausdrückt. Außerdem ist D relativ einfach zu berechnen. In Abb. ?? ist D illustriert.

Maximum-Likelihood-Methode

Bei der Maximum-Likelihood-Methode wird versucht, für empirische Werte einer Zufallsvariable die Verteilung zu finden, die am wahrscheinlichsten die wahre Verteilung der Zufallsvariable ist. Dazu wird für eine festgelegte Verteilungsklasse

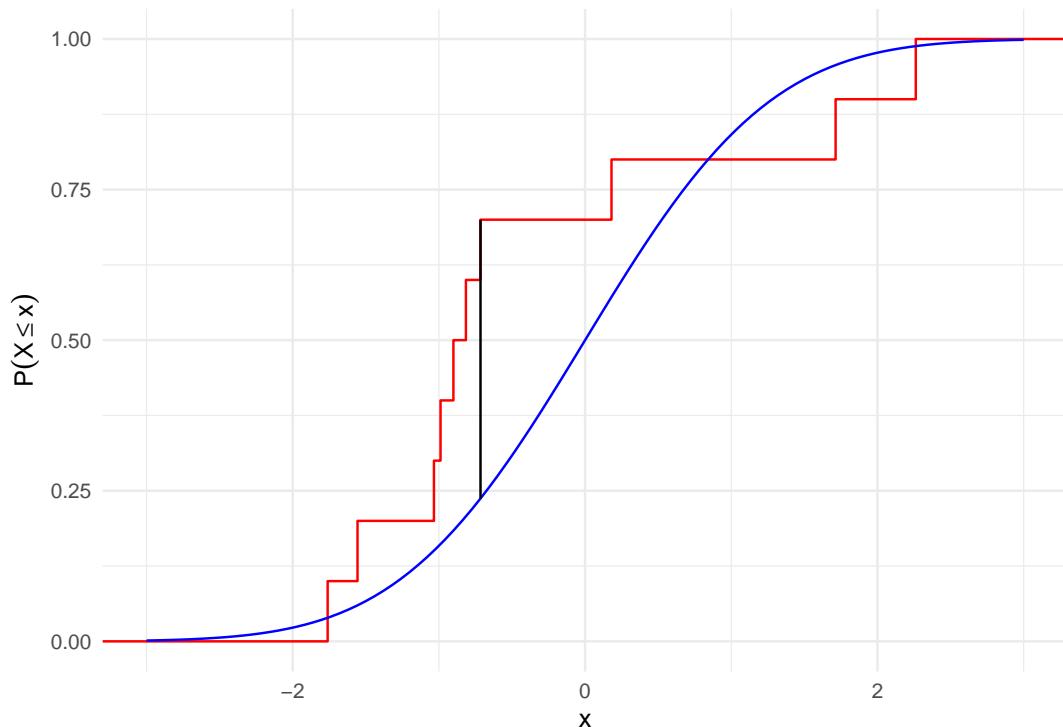


Abbildung 2.9: Beispiel zur Illustration der KS-Teststatistik D . Dargestellt ist eine zufällig generierte empirische Verteilungsfunktion (rot), verglichen mit der Verteilungsfunktion anhand der sie generiert wurde (blau). Die Teststatistik D , als maximaler Abstand zwischen zwei Verteilungsfunktionen definiert, ist in schwarz eingezeichnet.

nach den beschreibenden Parametern gesucht, die das Auftreten der beobachteten Daten unter der Voraussetzung der Verteilung mit den gegebenen Parametern maximiert. Beispielsweise könnte unter Voraussetzung von Normalverteiltheit nach denjenigen Parametern μ und σ gesucht werden, die eine Normalverteilung beschreiben, bei der das Auftreten der beobachteten Ausprägungen der Zufallsvariable mit maximalen Dichtewerten einhergehen. Um eine Maximum-Likelihood-Methode einzusetzen, muss vorausgesetzt werden, dass die Beobachtungen alle unabhängig und alle auf Basis derselben Verteilung geschehen sind. Diese Unabhängigkeit heißt aber für das Schätzverfahren auch, dass die Wahrscheinlichkeit aller Beobachtungen zusammen als Produkt der Einzelwahrscheinlichkeiten ausgedrückt werden kann. Dieses Produkt der Einzelwahrscheinlichkeiten gilt es nun zu maximieren. Da das Finden eines Maximums im relativ komplexen Fall von Dichten mit mehreren Ziel-Parametern aber schwer zu lösen ist, wird stattdessen zum natürlichen Logarithmus des Produkts gegriffen. Diese logarithmische Wahrscheinlichkeit - oder häufiger Log Likelihood genannt - wird dann entweder wo möglich analytisch oder per numerischer Integration und root-finding maximiert. Im Normalverteilungsbeispiel ergibt die analytische Lösung gerade, dass die wahrscheinlichsten Parameter Mittelwert und Streuung der Stichprobe sind.

Kapitel 3

Anforderungsanalyse

In enger Abstimmung mit dem Lehrstuhl für Materialphysik der Georg-August-Universität Göttingen wurde eine Reihe von Anforderungen an das zu entwickelnde Tool aufgestellt. So soll die endgültige Lösung, damit sie als Ersatz für das Linienschnitt-Verfahren eingesetzt werden kann, vergleichbare Ergebnisse wie das traditionelle Verfahren liefern. Vergleichbar heißt hier, dass die gefundenen Kornverteilungen in Form und Verteilungsklasse für jedes Bild so gut wie identisch zu den händisch gefundenen sein sollen, der Abstand zwischen den empirischen Verteilungsfunktionen der Korngrößen soll also minimal sein. Außerdem wäre wünschenswert, dass die Verarbeitung mit Hilfe des Tools einen deutlichen zeitlichen Vorteil gegenüber der mühsamen händischen Auswertung bedeutet. Um das Erreichen dieser Ergebnisse komfortabler zu gestalten, sollen in einer grafischen Oberfläche mindestens die Vorverarbeitungsschritte implementiert werden, die bereits zur Vorbereitung der Linienschnitte genutzt werden. Außerdem sollte zur Überprüfung der Ergebnisgüte eine Möglichkeit dazu bestehen, Ergebnisse händischer Auswertungen zu importieren und mit den mit Unterstützung generierten zu vergleichen. Neben diesen Grundfunktionalitäten soll das Programm in der Lage sein, verschiedene Dichtefunktionen an die gefundenen Korngrößen oder Radien anzupassen. Des weiteren ist natürlich erforderlich, dass generierte Ergebnisse und Größen-Verteilungen exportiert werden können.

Das ganze Tool soll dabei in Python implementiert und so einfach erweiterbar wie möglich sein - sollten weitere Algorithmen zur Vorbereitung oder Auswertung hinzukommen, sollte einfach nur eine Funktion hinzugefügt, nicht das ganze Skript angepasst werden müssen. Python ist deswegen die Sprache der Wahl, da sie bereits breit am Lehrstuhl für Materialphysik eingesetzt wird und so eventuelle Erweiterungen erleichtert werden.

Kapitel 4

Design und Entwurf

Das Tool soll unter der grafischen Oberfläche eine Reihe von Features anbieten. Dies sind zum einen Lösungen für Daten-Handling (Import von Kornbildern und Linienschnitt-Ergebnissen, Export von Resultaten), dann Vorverarbeitungs-Schritte und zuletzt die eigentliche Korn-Detektion. Das Tool soll deshalb in vier Modulen aufgebaut werden:

1. Einem Kornbild-Modul, das eine Klasse zur Abbildung der Kornbilder und je ein Interface zur Vorverarbeitung und Auswertung bietet.
2. Einem Vorverarbeitungs-Modul, das leicht erweiterbar die Vorverarbeitungen in einem möglichst einheitlichen Interface definiert.
3. Einem Analyse-Modul, das auch leicht erweiterbar wieder mit einem einheitlichen Interface Analysemethoden definiert.
4. Einem GUI-Modul, das die Darstellung der Kornbilder, der Korngrößen und die Interaktion mit dem Kornbild ermöglicht.

In Abb. ?? ist das geplante Tool-Design zur Abbildung der Anforderungen an einem Beispiel dargestellt.

Die von Objekten der Kornbild-Klasse aufrufbaren Interfaces zu den Analyse- und Vorverarbeitungsmodulen sollen dabei über je eine zentrale Funktion ablaufen, die die jeweiligen Auswertungs- oder Vorverarbeitungsschritte bei Weiterreichen der

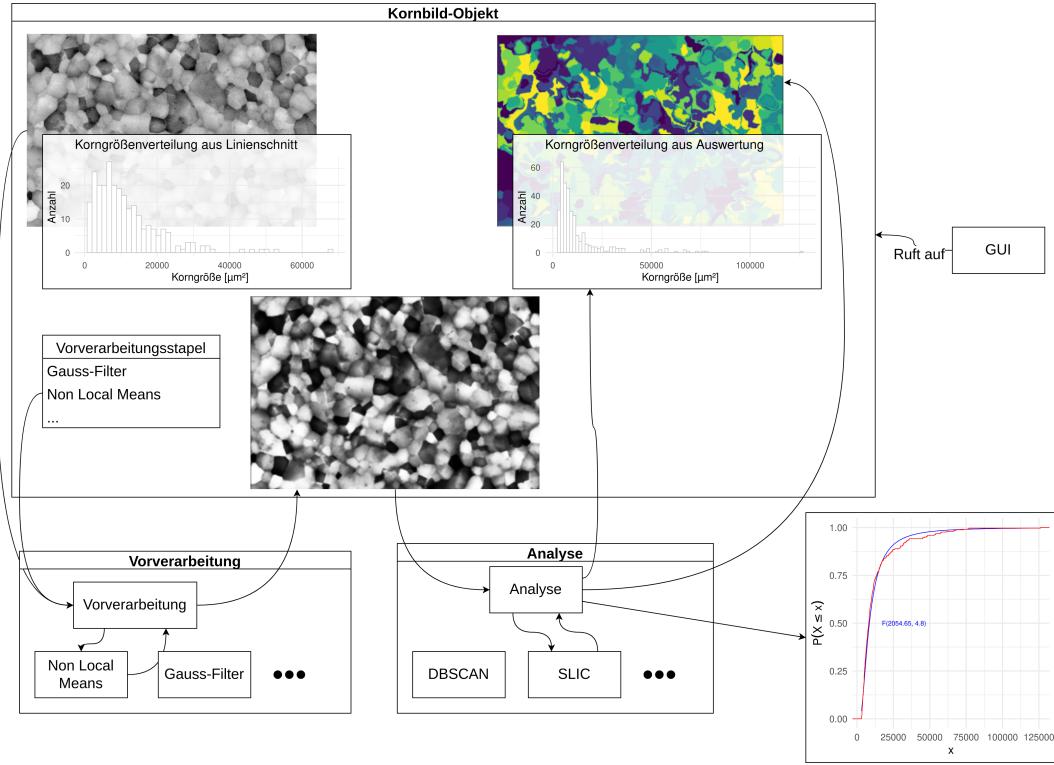


Abbildung 4.1: Illustration des Tool-Designs.

gegebenen Argumente als Unterfunktionen aufrufen. So kann zur Erweiterung des Funktionsumfangs einfach im entsprechenden Modul eine Funktion hinzugefügt und der zentrale Wrapper erweitert werden, ohne dass das Kornbild-Modul angepasst werden muss. Solange die Funktionen dabei in einem standardisierten System benannt und dokumentiert werden, können die Funktionsumfänge der Analyse- und Vorverarbeitungsmodule dann auch zur flexiblen Erstellung von GUI-Schaltflächen genutzt werden, so dass auch hier bei einer Erweiterung des Funktionsumfangs ein Minimum an Anpassungen nötig wird.

Objekte der Kornbild-Klasse sollen außerdem Methoden zum Einlesen der vom Elektronenmikroskop gelieferten .tif-Bilder und der etwaig vorliegenden Linienschnitt-Auswertungen liefern. Für beide Inhalte muss je ein Attribut in der Klasse vorgehalten werden. Um verschiedene Auswertungsschritte und -reihenfolgen ausprobieren zu können, muss für Bilder und Korngrößen-Verteilung außerdem je ein Attribut für die veränderten und ausgemessenen Größen angelegt

werden.

Da bei der Vorverarbeitung die Reihenfolge der angewandten Algorithmen entscheidend ist, sollen die geplanten Vorverarbeitungen außerdem nicht einfach bei Aufruf ausgeführt werden, sondern in Form eines Stapels geplant und dann auf Wunsch hintereinander ausgeführt werden. Ein Beispiel für den Einfluss der Reihenfolge der Vorverarbeitung ist in Abb. ?? zu sehen. Auch dieser Stapel muss in der Kornbild-Klasse als Attribut angelegt werden.

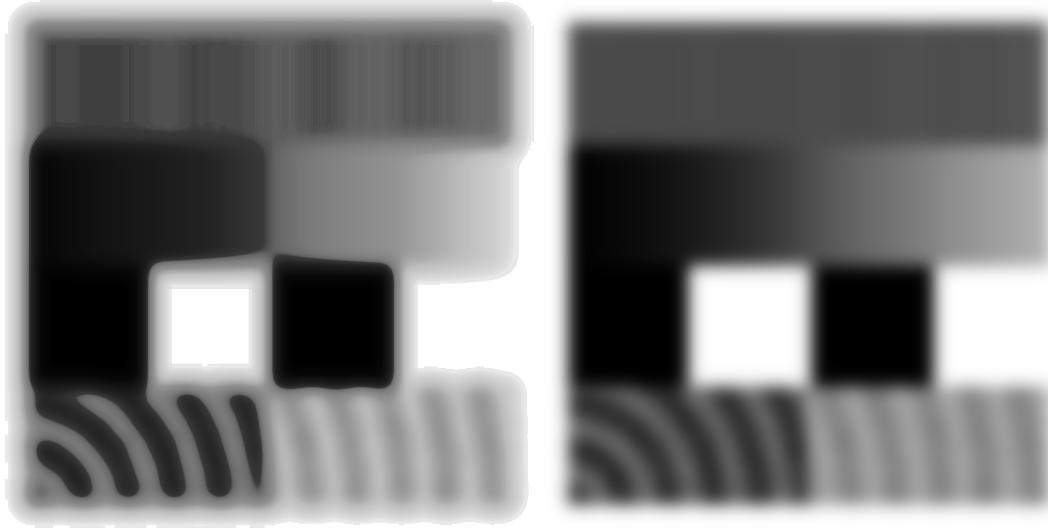


Abbildung 4.2: Einfluss der Vorverarbeitungs-Reihenfolge. Links ist zuerst ein Histogramm-Equalizer, dann ein Gauss-Filter angewandt worden, rechts dieselben Filter in umgekehrter Reihenfolge. Der Gauss-Filter hat mit seiner Kernelgröße von $\approx 10\%$ der Bildgröße eine relativ extreme Einstellung, der Effekt der Reihenfolge ist aber deutlich zu erkennen.

Für die Analyse- und Vorverarbeitungsmodule gilt ein zentrales Design-Prinzip:

Das Ziel der Aufrufbarkeit der Verarbeitungsschritte über eine zentrale Funktion heißt im Detail, dass alle Verarbeitungsfunktionen zum einen möglichst ähnliche Interfaces, zum anderen dasselbe Outputformat aufweisen müssen. Im Idealfall sollten die einzelnen Verarbeitungsschritte jeweils in eine feste Pipeline integriert werden, um Lesbarkeit und Erweiterbarkeit der Lösungen zu verbessern.

Alle Vorverarbeitungsschritte, die bisher durch das Institut für Materialwissenschaften standardmäßig eingesetzt werden, sollten in das Vorverarbeitungsmodul

integriert werden. Für das Analysemodul sollen die in den Grundlagen diskutierten, vielversprechenden Superpixel-Methoden implementiert werden. Außerdem soll das Analysemodul für eine gegebene empirische Korngrößen-Verteilung die Möglichkeit bieten, eine Verteilungsfunktion anzupassen und die Verteilungsparameter zurückzugeben.

Als letzter zu beschreibender Punkt bleibt die GUI. Neben dem Zugriff auf die oben genannten Funktionen zur Datenverarbeitung, stapelweisen Vorverarbeitung und der Analyse sollte die GUI über Grafiken Feedback zum vorliegenden Bild, dem Ergebnis der (Vor-)verarbeitung und der händischen oder ausgemessenen Korngrößen-Verteilungen bieten. Es ist also ein Kontroll-Bereich zur vollständigen Nutzung und Steuerung der Funktionalität und ein Feedback-Bereich mit grafischen Darstellungen der Verteilungs- und Bildattribute des Kornbild-Objekts einzuplanen.

Kapitel 5

Umsetzung und Entwicklung

Wie vom Institut für Materialforschung gefordert, wurde das Tool in der Programmiersprache Python umgesetzt. Im folgenden Kapitel wird anhand der grafischen Oberfläche auf eine Reihe von Umsetzungsaspekten eingegangen. Eine Reihe von Designentscheidungen ist im Prozess nach Rücksprache mit den Mitarbeitenden der Uni Göttingen entstanden, dies wird an entsprechender Stelle erwähnt.

Die GUI wurde mit PySimpleGUI (PySimpleGUI, n.d.) implementiert und ist exemplarisch im Einsatz für eine Kornauswertung in Abb. ?? dargestellt. Links sind die Steuerelemente für Datenhandling, Vorverarbeitung, Analyse und GUI zu sehen, rechts die Bilder- und Kornverteilungen vor und nach der Analyse.

Mit der Import-Schnittstelle im File-Handling-Segment (in Abb. ?? blau umrandet) wird in einem Dialog der Pfad zu dem Kornbild angegeben, das geöffnet werden soll. Dieser Pfad wird dann, sollte er in '.tif' enden, mit Hilfe des OpenCV-Moduls (*Opencv-Python*, n.d.) genutzt, um das gewählte Kornbild zu importieren. Die Aufnahmen werden vom im Institut für Materialphysik eingesetzten Elektronenmikroskop im **tif**-Format herausgeschrieben und mit Meta-Informationen wie zum Beispiel der Kantenlänge eines Pixels versehen. Beim Import werden die Datenfelder für die Höhe des Datenbalken und die Kantenlänge eines Pixels genutzt, um zum einen den Datenbalken auszublenden und zum anderen den Umrechnungs-

faktor von Pixelzahl in Fläche in μm^2 in einem dafür vorgesehenen Attribut des beim Öffnen initialisierten Kornbild-Objektes abzulegen. Zusätzlich wird im angegebenen Pfad nach einer .csv-Datei gesucht, die dieselbe Probenbeschreibung im Namen trägt, wie das tif-Bild. Sollte eine solche Datei vorliegen, wird sie mit Hilfe des pandas-Moduls(*Pandas*, 2010/2022) importiert und in einem dafür vorgesehen Attribut vorgehalten. Nach dem Import werden Kornbild und - wenn vorhanden - Korngrößenverteilungen außerdem rechts oben in der GUI mit Hilfe des matplotlib-Moduls(*Matplotlib/Matplotlib*, 2011/2022) dargestellt.

Die Vorverarbeitungs- und Analysetools (in Abb. ?? rot umrandet) sind beide nicht-statisch beim Programm-Start generiert. Dazu werden für beide Module mit Hilfe des inspect-Moduls alle Funktionen aufgelistet, die einer entsprechenden Namens-Konvention folgen. Deren Docstrings und Signaturen werden daraufhin ausgewertet, um Benennung und Argumentliste zu erstellen und als Text- und Input-Bausteine in die GUI einzufügen.

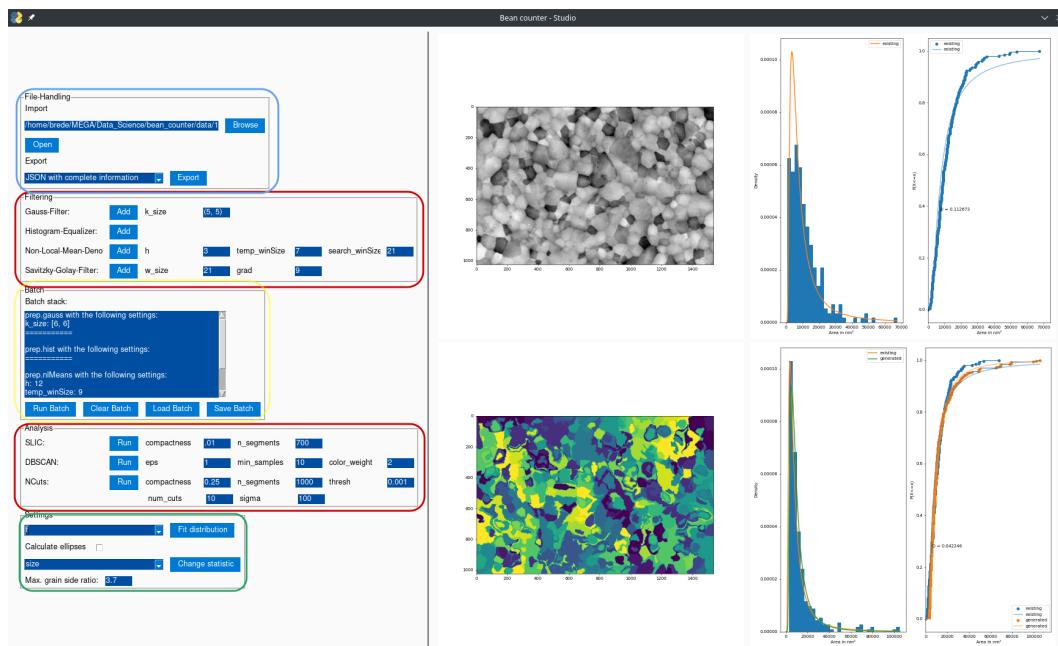


Abbildung 5.1: GUI des Tools. Links sind die Steuer-Elemente, geteilt nach File-Handling, Bildvorverarbeitung, Analyse und generellen Einstellungen, rechts sind oben Bild vor Verarbeitung und händisch gemessene Korngrößen und unten das Bild nach Verarbeitung und die gefundenen Korngrößen im Vergleich mit den händisch gemessenen dargestellt.

Die Vorverarbeitungsschritte werden bei Klick auf “Add” als Zeichenkette nach einem standardisierten Format an einen Parser übergeben, der als Teil des Kornbild-Moduls ein `dict` erstellt und dieses der Batch-Stack Liste hinzufügt. So wird zum Beispiel ein Gauss-Filter mit 5 Pixeln horizontaler und vertikaler Kernel-Größe als “`prep.gauss.(k_size=(5, 5))`” an den Parser übergeben und als “`{'category': 'prep', 'type': 'gauss', 'k_size': (5, 5)}`” zum Batch-Stapel hinzugefügt. Eine andere Methode des Kornbilds wird danach von der GUI aufgerufen, die den gesamten Batch-Stack in menschenlesbarer Form in der dafür vorgesehene Batch-Liste ausgibt (in Abb. ?? gelb umrandet). Das Gauss-Filter Beispiel wird so zu dem Eintrag:

```
“prep.gauss with the following settings: k_size: (5, 5)”
```

Die Schaltflächen unter der Batch-Liste erlauben, die erstellte Liste als JSON zu exportieren (**Save Batch**), aus einer JSON zu laden (**Load Batch**), alle Einträge zu löschen (**Clear Batch**) und den Stapel an Vorverarbeitungsschritten auszuführen (**Run Batch**). Die Ausführung ist dann so umgesetzt, dass die Listeneinträge iterativ an die zentrale Funktion des Vorverarbeitungsmodul weitergegeben werden. Hier wird der “`type`”-String im Aufrufs-`dict` genutzt, um die richtige Funktion aufzurufen, die darauf folgenden Einträge werden der Funktion als `**kwargs` übergeben. Um neue Vorverarbeitungsmöglichkeiten anzufügen, muss dem Modul nur die Funktion und der zentralen Funktion die Option eines neuen `type`-String hinzugefügt werden. In der letzten Version sind die folgenden Vorverarbeitungsschritte implementiert:

- Savitzky-Golay-Filter in der Implementation aus dem `scipy.signal`-Modul (*Scipy/Scipy*, 2011/2022), so angepasst, dass das Polynom immer größer als das betrachtete Fenster und das Fenster immer eine ungerade Pixelgröße hat. Außerdem wird bei jeder Ausführung der Filter zweimal angewandt, einmal über die Bild-Horizontale, dann über die Bild-Vertikale,
- Histogram-Equalizer in der Implementation aus dem `OpenCV`-Modul (*Opencv-Python*, n.d.),

- Gauss-Filter in der Implementation aus dem OpenCV-Modul (*Opencv-Python*, n.d.), dabei um eine Überprüfung auf ungerade Pixelzahlen und entsprechender Anpassung in beiden Kernel-Dimensionen erweitert und
- Non-Local-Means-Denoiser in der Implementation aus dem OpenCV-Modul (*Opencv-Python*, n.d.).

Diese vier Vorverarbeitungsschritte und ihre Implementationen wurden deswegen ausgewählt, weil sie bereits zur Vorbereitung des Linienschnitt-Verfahrens im Institut für Materialforschung eingesetzt wurden. Dabei wird bei allen Vorbereitungsschritten das für das modifizierte Bild vorgesehene Attribut als numpy-Array eingelesen, gefiltert und abschließend überschrieben. Durch das Setzen dieses Attributs auf das Originalbild vor jeder Durchführung der Vorverarbeitungsliste wird sichergestellt, dass keine vorausgegangenen Analyseschritte das Ergebnis beeinflussen.

Alle Analyse-Algorithmen wurden wieder über eine zentrale Funktion implementiert. Auch hier wird, wie im Fall der Vorverarbeitungs-Algorithmen, von der GUI ein String aus der Ausführungs-Aufforderung generiert, der nach der Umwandlung in ein `dict` an das Analyse-Modul weitergegeben wird, das nach dem “Type”-String die entsprechende Funktion aufruft. Genau wie im Vorverarbeitungsschritt wird hier auch das modifizierte Kornbild aus dem Kornbild-Objekt ausgelesen, weiterverarbeitet und wieder eingefügt. Zur Analyse der Bilder wurde anfänglich nur DBSCAN aufgenommen, nach dem Test durch Studierende des Instituts für Materialphysik noch SLIC und eine Kombination von NCuts und vorangegangenem SLIC. Die Implementation des DBSCAN-Algorithmus erfolgte auf Basis des `scikit-learn cluster` Moduls (*Scikit-Learn/Scikit-Learn*, 2010/2022). Damit die Analysen möglichst reibungslos erweitert werden können, wurden hier die objektorientierten Interfaces von `scikit-learn` genutzt, um möglichst in der Analyse-Funktion nur das Model-Objekt zu erstellen und alle darauffolgenden Verarbeitungsschritte in verallgemeinerten Funktionen zu verpacken, die dann die `fit_predict`-Methode des jeweiligen Modells aufrufen. Nach ersten Tests mit dem

DBSCAN-Algorithmus stellte sich heraus, dass insbesondere für Bilder mit kleinen, regelmäßigen Kornmustern (für ein Beispiel siehe Abb. ??), die Möglichkeiten dieses Algorithmus an ihre Grenzen kommen. Um diesem Problem zu begegnen, wurde der Funktionsumfang nach Besprechung der bei Stutz et al. (2018) diskutierten Superpixel-Methoden mit Mitarbeitern des Instituts für Materialforschung um **SLIC** und **Normalized Cuts** erweitert. Für beide Methoden wurde die Implementation aus dem **scikit-image**-Modul (*Scikit-Image*, 2011/2022) genutzt, deren Interfaces aber leicht von den bisher genutzten **scikit-learn** Interfaces abweichen. Um die Analyse-Pipeline möglichst wenig anpassen zu müssen, wurde deshalb ein weiteres Modul geschrieben, das je eine **NCuts**- und **SLIC**-Klasse vorhält, die eine an die bisherige Pipeline angepasste `fit_predict` Methode anbieten. Die Anpassung bedeutete insbesondere, dass aus dem standardmäßig von **scikit-image** zurückgegebenen Bild-Matrizen eine Liste mit Cluster-Labels wird, wie sie von **scikit-learn** beim Einsatz von Cluster-Methoden zurückgegeben wird.

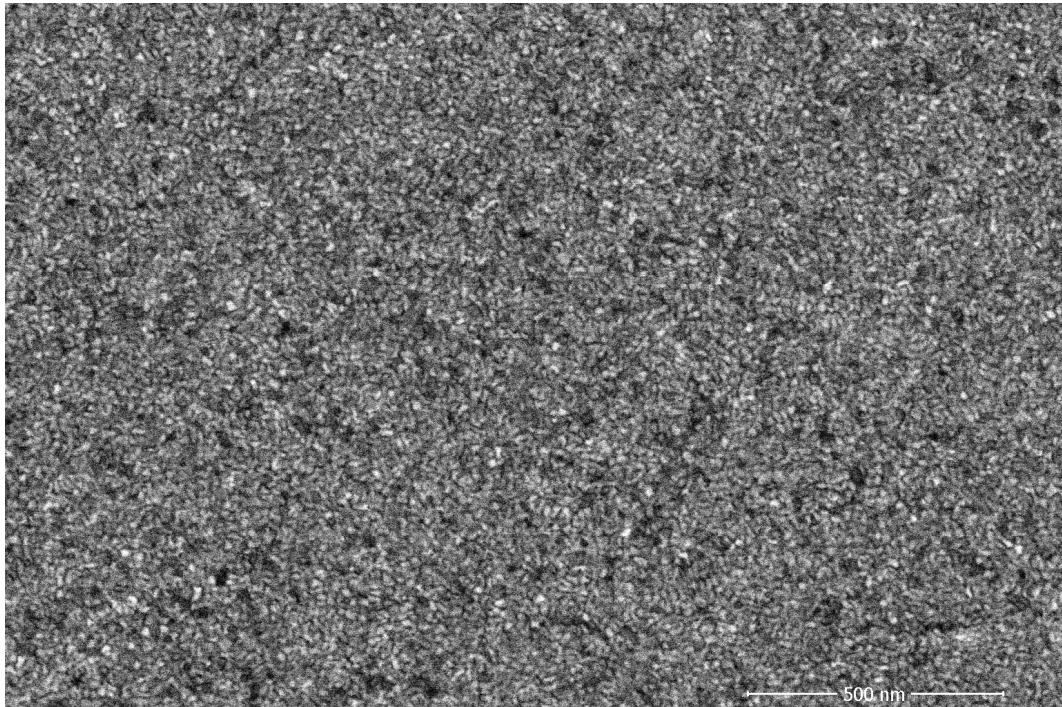


Abbildung 5.2: Kornbild mit besonders kleinen Strukturen.

Nach der Erstellung der Analyse-Modell-Objekte durch die Hilfsfunktionen werden so standardisiert bei allen Algorithmen Cluster-Labels generiert. Anschließend

wird ein Post-Processing angestoßen, das eine Reihe von Beschreibungsgrößen berechnet, die von den Mitarbeitern der Uni Göttingen als hilfreich genannt wurden. Zum einen sind das einfache Beschreibungen wie Anzahl an Pixeln, Größe des Clusters in μm^2 , X- und Y-Koordinaten des Cluster-Mittelpunkts, die Varianz der Farbe der im Cluster befindlichen Pixel und der Durchmesser des Kreises, der der Fläche entsprechen würde. Außerdem wurde die Möglichkeit eingebaut, mit Hilfe des `scikit-image`-Moduls (*Scikit-Image*, 2011/2022) Ellipsen an das Cluster anzupassen, deren Winkel und Länge der Hauptachse und Länge der kürzeren Achse auch ausgegeben werden. Da dieses Fitten aber computational vor allem für Bilder von kleinen Körnern aufwändig ist, wurde im “Settings”-Segment der GUI (in Abb. ?? grün umrandet) ein Steuerungselement eingebaut, mit der diese Berechnung als Boolches Attribut im Kornbild-Objekt unterdrückt werden kann. Neben dieser Zusammenfassung wird nach erfolgreichem Durchlauf der Superpixel-Algorithmen außerdem ein Histogramm und eine kumulierte Häufigkeitsverteilung der Korngrößen berechnet, an die außerdem per Maximum-Likelihood-Schätzung nach der Implementierung im `scipy.stats`-Modul (*Scipy/Scipy*, 2011/2022) eine Verteilungsfunktion angepasst wird. Die Klasse der Verteilungsfunktion, sowie die Auswahl der Korndurchmesser statt der -fläche können auf Wunsch der Mitarbeitenden auch im “Settings”-Segment der GUI ausgewählt werden. Verschiedene Funktionsklassen als Auswahlmöglichkeiten können hierbei als `dict`-Objekt bei Programmstart angegeben werden, auch hier ist eine Erweiterung über den bisherigen Umfang also ohne großen Aufwand möglich. Ein weiteres Ergebnis der Besprechungen mit Mitarbeitenden der Uni Göttingen ist das `Max. grain side ratio`-Feld im `Settings`-Segment. Die vorherigen Versionen des Tools neigten, insbesondere bei Auswertung mit DBSCAN, zur Erstellung von bildübergreifenden Clustern, die die Schatten an den Korngrenzen beinhalteten. Um diese Klassifikation nicht in die Ergebnisse einfließen zu lassen, kann mit dem Feld für das Maximale Seitenverhältnis aus den gefundenen Clustern die Menge herausgefiltert werden, die ein Seitenverhältnis aufweisen, das größer als der angegebene Wert ist. So kann über Expertenvorwissen über das untersuchte Material informiert eine

Reihe von unmöglichen geometrischen Formen ausgeschlossen werden.

Die generierten Korn-Verteilungen werden dann rechts unten in der GUI dargestellt. Um die gefundenen Cluster deutlicher hervorzuheben, wird das Ergebnis statt in Grautönen wie bei dem eingelesenen und dann beim vorverarbeiteten Bild in einer viridis-Farbskala präsentiert. Außerdem können die Ergebnisse und dazu berechneten deskriptiven Statistiken mit dem Export-Tool im File-Handling-Segment (in Abb. ?? blau umrandet) exportiert werden. Dazu stehen der nutzenden Person zwei Export-Formate zur Auswahl - zum einen eine CSV nur mit den deskriptiven Kornstatistiken, zum anderen eine JSON mit zusätzlich Informationen über die vollzogene Vorverarbeitung und den Parametern der mit `scipy.stats` angepassten Verteilung.

Kapitel 6

Evaluation

Im folgenden Kapitel wird die durchgeführte Überprüfung der Umsetzung beschrieben. Dazu wird zuerst auf den Prozess der Rücksprachen und Tests mit und durch die Mitarbeitenden des Instituts für Materialphysik und die dabei entstandenen Testergebnisse eingegangen, darauf folgen die Ergebnisse des Versuchs, bestehende, händisch generierte Kornverteilungen möglichst gut durch die implementierten Methoden nachzuvollziehen.

6.1 Praxistest

Während des Entwicklungs-Prozesses des Tools wurden an insgesamt 8 Terminen Video-Calls mit Mitarbeitenden der Uni Göttingen durchgeführt. In diesen Treffen wurden regelmäßig der Stand des Tools präsentiert und weitere Schritte besprochen. Neben den Treffen wurden asynchron Tests durchgeführt und Änderungswünsche und Bugreports über Github Issues gesammelt. Über diesen Entwicklungsprozess wurde sichergestellt, dass alle Design-Anforderungen an die Bedienmöglichkeiten und oberflächlichen Eigenschaften des Tools erfüllt sind. So wurde während dieses Prozesses die Auswahl an Vorverarbeitungsschritten festgelegt, die möglichst gute Darstellung bisheriger und gefundener Ergebnisse diskutiert und eine Reihe von Anpassungen zur Verbesserung der Nutzbarkeit wie eine höhere

Verbosität bei Speichern und Laden und ein Fortschrittsbalken während laufender Auswertungen hinzugefügt. Nachdem das Tool so weit fertig gestellt war, dass erste manuelle Tests zu vielversprechenden Korngrößen-Verteilungen führten, wurde am 18.5.2022 eine weitere Evaluationsrunde angestoßen. Dazu wurde zwei Studierenden des Instituts das Tool präsentiert und seine Nutzung erklärt - verbunden mit der Bitte, es für ihre Auswertung auszutesten. Von den zwei instruierten Studierenden brach einer kurz nach dem Gespräch seine Abschlussarbeit ab, wodurch der praxisnahe Anwendungstest nur von einer Person durchgeführt wurde. Am 27.5. berichtete diese Testperson, dass die Auswertung mit DBSCAN bei ihren Bildern zu keinem befriedigenden Ergebnis führte. Daraufhin wurde das Tool nach Konsultation mit der Betreuung der Person in Anlehnung an Stutz et al. (2018) und die dort gezeigten Muster der Superpixel um **Normalized Cuts** und **SLIC** erweitert. Am 6.6. berichtete die Testperson, dass sie ihre Bilder erfolgreich mit SLIC auswerten konnte - die Verteilungen seien oberflächlich vergleichbar mit den per Linienschnitt generierten. Weiterhin wurde berichtet, dass die Anpassung des ersten Bildes eines Stapels wegen der Suche von passenden Vorverarbeitungs- und Auswertungseinstellungen etwa 5 Minuten länger gedauert habe, als die Auswertung per Linienschnitt. Bei jedem weiteren Bild von ähnlichem Material sei die Auswertung mit dem Tool aber pro Bild etwa 25 Minuten schneller gewesen.

Die Ergebnisse dieses Feldversuchs können als vorsichtige Belege für die Erfüllung einer Reihe von Anforderungen an das Tool gesehen werden:

1. die Ergebnisse sahen laut Aussage der Testperson ähnlich zu den händisch generierten Kornverteilungen aus.
2. die Auswertung ist ab dem 2. Bild ohne Empfehlungen für Voreinstellungen schneller als die per Linienschnitt.
3. durch die Notwendigkeit der Erweiterung um Normalized Cuts und SLIC konnte die einfache Erweiterbarkeit getestet werden, die weitere Funktionalität konnte reibungslos aufgenommen werden.

6.2 Technische Evaluation

Für die Überprüfung der Validität der ausgewählten Superpixel-Methoden wurde versucht, für jedes zur Verfügung stehende Bild und jede implementierte Auswertungsmethode ein Set an Einstellungen zu finden, das die generierte Kornverteilung möglichst nah an per Linienschnitt ausgemessene Verteilung der Korngrößen bringt. Dazu wurde mit Hilfe des **HPBandster**-Moduls (*HpBandSter*, 2017/2022) versucht, möglichst optimale Einstellungen zu finden, die zu Auswertungsergebnissen mit der minimalen Kolmogorov-Smirnov Teststatistik beim Vergleich mit den Linienschnittergebnissen führen. Während der Masterarbeit wurde das Git-Repository des **HPBandster**-Moduls mit dem Hinweis aktualisiert, dass das Modul nicht weiter gepflegt wird - trotzdem wurden aus Gründen der Vergleichbarkeit die restlichen Tests auch mit diesem Modul durchgeführt. Dabei ist die Wahl des Optimizers aber insofern fragwürdig, da die Vorteile des Hyperband-Aspekts des Implementierten BOHB-Verfahrens bei der Optimierung der Analyseverfahren nicht ausgenutzt werden. Damit dieses Verfahren zu schnelleren Analyseergebnissen führt, muss ein sinnvolles Budget definiert werden können, das weniger aufwändige Auswertungen als Indikatoren für aufwändigere Auswertungsgüten möglich macht. Da die einzige wirkliche Stellschraube zur Reduktion von Cluster-Laufzeiten die Reduktion der Anzahl der berechneten Distanzen ist und dies im gegebenen Anwendungsfall eine Reduktion der Bildgröße bedeuten würde, ist die BOHB-Optimierung nur noch mit einem reinen TPE zu vergleichen. Zwar könnte die Bildfläche vor dem Anpassen reduziert werden, diese Lösung würde zum Beispiel bei SLIC aber auch zu weniger Cluster-Zentren bei der optimalen Lösung führen. Dadurch würde die Hyperband-Optimierung bei niedrigerem Budget zu aktiver Verschlechterung der Zielfunktions-Schätzung für das gesamte Budget bedeuten. Der Parameterraum des Optimierungsversuchs aller drei Analysemethoden bestand dabei aus den Filtern und ihren Einstellungen, die Reihenfolge der Filter und der Argumente der Analysemethoden. Die Grenzen und Auflösungen der Parameterräume wurden dabei so festgelegt, wie sie nach den händischen

Tests sinnvoll erschienen. Nach einem kleinen Test wurde bei Boden- und Deckeneffekten entsprechend der Parameterraum angepasst.

In Abb. ?? ist das Ergebnis der drei Optimierungen zu sehen - die Einstellungen und die Vorverarbeitung jeden Verfahrens wurden dabei in 100 Durchläufen optimiert. Es ist deutlich zu erkennen, dass SLIC und DBSCAN mit Abstand besser abgeschnitten haben als Normalized Cuts. Außerdem entsteht der Eindruck, dass SLIC und DBSCAN als ergänzende Ansätze zu verstehen sind - die die Aufnahmen verbindenden Linien kreuzen sich. Bilder, die höhere und damit schlechtere *D*-Fit-Statistiken mit SLIC ergaben, konnten also tendenziell besser mit DBSCAN ausgewertet werden und andersherum.

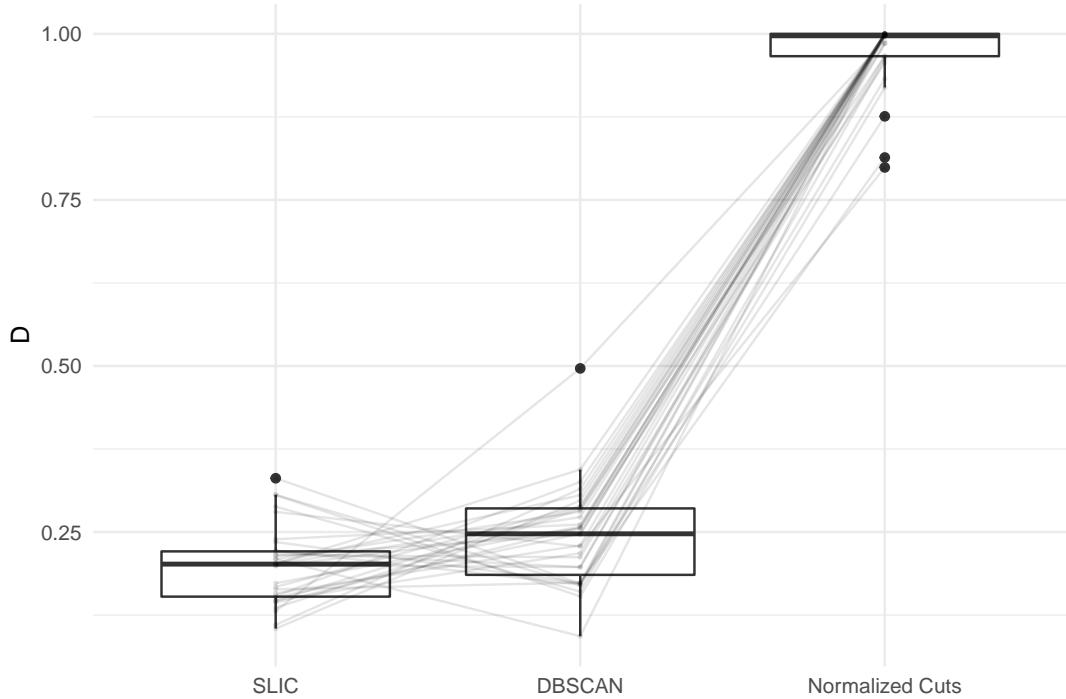


Abbildung 6.1: Ergebnisse der Bayes-Optimierung. Die TPE wurden für jedes Bild auf die *D*-Statistik zwischen der generierten und der händisch gemessenen Kornverteilung optimiert. Der Wert kann normalerweise zwischen 0 und 1 liegen, wurde aber auf 1.5 gesetzt, wenn keine Lösung zu finden war. Dies war insbesondere dann der Fall, wenn bei den Normalized Cuts keine sinnvolle Lösung gefunden werden konnte. Die grauen Linien verbinden die Loss-Werte in den drei Bedingungen pro Bild.

Um die vielversprechenden Ergebnisse für SLIC und DBSCAN zu verdeutlichen, wurde je ein zweiter Optimierungsversuch gestartet, diesmal mit je 500 Optimie-

rungsdurchläufen (Abb. ??). Das Ergebnis bestätigt den Eindruck aus dem ersten Test, die beiden Methoden scheinen zum Einen ergänzend einzusetzen zu sein (die am schlechtesten angepasste Kornverteilung aus der SLIC-Optimierung konnte mit DBSCAN am besten vorhergesagt werden), zum Anderen bestätigt sich der leichte Trend aus den 100 Optimierungsdurchläufen, nach dem SLIC für die zu Verfügung stehenden Bilder ein insgesamt besseres Ergebnis erzielte. Beide Verfahren scheinen aber verlässliche Ergebnisse produzieren zu können. Auch wenn die D -Werte beider Verfahren bei Vergleich mit der entsprechenden Verteilung bei der Anzahl an gefundenen Körnern wahrscheinlich signifikant würden und damit auf unterschiedliche Ursprungs-Verteilungen hinwiesen - die Verteilungen nähern sich deutlich an. Unter Berücksichtigung der aus dem Linienschnitt entstehenden Fehler und dem unterschiedlichen Anteil des ausgewerteten Bildes, sind die Ergebnisse vielversprechend, benötigen aber weitere Evaluation.

??

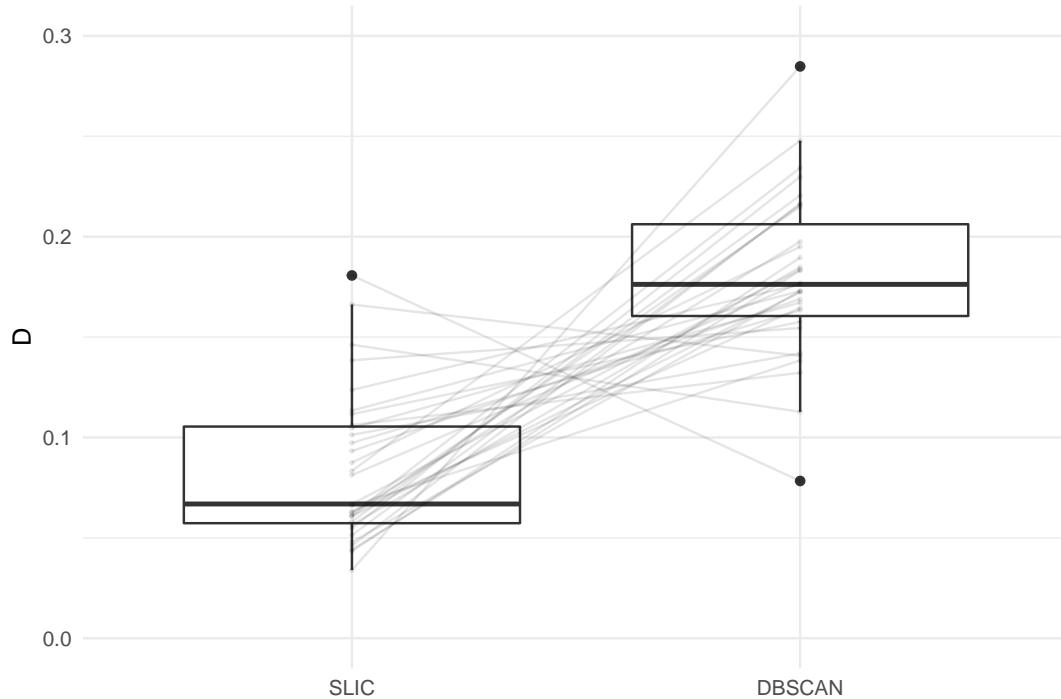


Abbildung 6.2: Ergebnis von 500 Optimierungsdurchläufen für SLIC und DBSCAN. Die grauen Linien verbinden die Loss-Werte in den zwei Bedingungen pro Bild.

Als weiterer Aspekt kann anhand der Optimierungsergebnisse überprüft werden, ob die optimalen Einstellungen der Vorverarbeitung methodenunabhängig sind. Sollte dies so sein, sollten die Optimierungsversuche bei SLIC und DBSCAN zu ähnlichen Ergebnissen pro Bild gekommen sein.

In Abb. ?? sind die Zusammenhänge aller optimierter Vorverarbeitungs-Schritte dargestellt. Es ist kein Zusammenhang bei keinem der Parameter ersichtlich, die Vorverarbeitung scheint also analysespezifisch angepasst werden zu müssen. Insbesondere in Anbetracht der Parametrischen Natur vom SLIC zugrundeliegenden k-Means Clustering und der Non-parametrischen Natur von DBSCAN scheint der Einfluss der Reduktion parametrischen Rauschens einleuchtend.

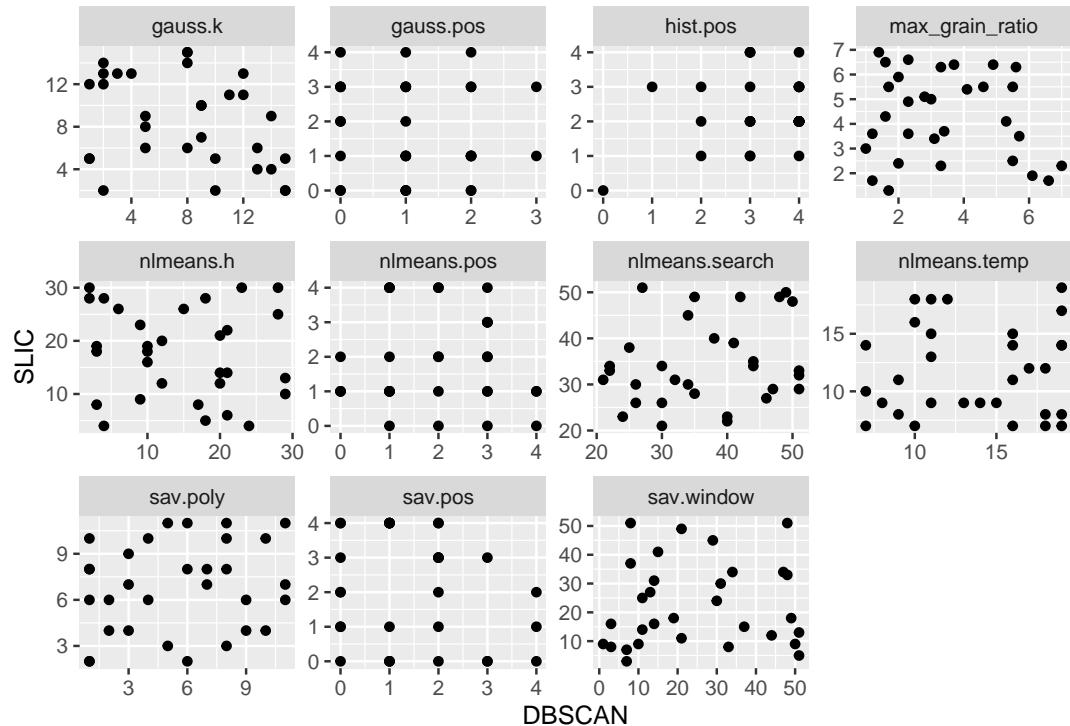


Abbildung 6.3: Scatterplots und Korrelationen der Vorverarbeitungseinstellungen. Eine Unabhängigkeit der optimalen Einstellungen von den Analyseverfahren sollte zu hohen Korrelationen und annähernd diagonalen Punktwolken führen.

In Abb. ?? sind die Kornbilder zu sehen, die den größten Abstand zwischen ihrem Fit in der SLIC und der DBSCAN-Auswertung aufweisen. Das besser von SLIC ausgewertete Bild weist kleinere, regelmäßige Körner auf, die Körner im von DBSCAN besser ausgewerteten Bild sind wesentlich unregelmäßiger in Form und

Größe. Der Eindruck der Überlegenheit SLICs im Erkennen regelmäßiger, kleiner Strukturen aus dem Praxistest bestätigt sich also.

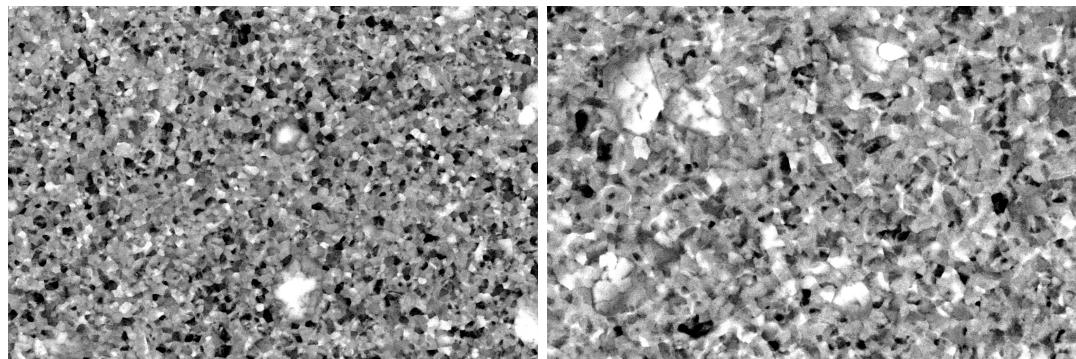


Abbildung 6.4: Bilder mit größtem Unterschied in DBSCAN und SLIC-Fit. Links ist das Bild mit dem schlechtesten SLIC-Fit bei gleichzeitig bestem DBSCAN-Fit zu sehen, rechts umgekehrt.

Abschließend kann über den Versuch mithilfe von TPE-Optimierung die eingesetzten Analysen an die händische Linienschnitt-Auswertung anzupassen gesagt werden, dass zumindest für DBSCAN und SLIC eine Lösung möglich ist. Für Normalized Cuts nach SLIC muss festgestellt werden, dass zumindest mit dem gewählten Hyperparameter-Raum und der gewählten Optimierungsmethode kein wirklich verlässliches Ergebnis für die vorliegenden Bilder gefunden werden konnte. Für SLIC und DBSCAN ist außerdem zu bemerken, dass die Verfahren bei unterschiedlichen Bildern unterschiedlich gut die per Linienschnitt gefundenen Kornverteilungen annähern. So ist beim Vergleich der Bilder mit dem größten Unterschied im Verteilungsabstand exemplarisch zu sehen, dass DBSCAN besser bei unregelmäßigen Korngrößen abschneidet, SLIC bei regelmäßigen, kleinen Körnern aber die Nase vorn hat.

Kapitel 7

Schlussbetrachtung

Ziel dieser Arbeit war es, ein Tool als Alternative zum Linienschnitt-Verfahren zur Auswertung elektronenmikroskopischer Aufnahmen von Kornstrukturen zu entwickeln. Dieses Vorhaben wurde in Kooperation mit dem Institut für Materialphysik der Georg-August-Universität Göttingen angestoßen, an dem bisher breit das Linienschnitt-Verfahren eingesetzt wird, das sehr aufwändig in der Durchführung und zusätzlich anfällig für Fehler ist. Damit das Tool sinnvoll eingesetzt werden kann, sollte es vergleichbare Kornverteilungen wie die händische Auswertung generieren und die gewohnten Vorverarbeitungsschritte abdecken. Außerdem ist der Einsatz erst dann sinnvoll, wenn die Nutzung des Tools zu einer deutlichen Beschleunigung der Auswertung führt. Eine weitere Voraussetzung für die Nachnutzbarkeit des Tools ist die einfache Erweiterbarkeit dessen.

Um diese Anforderungen zu erfüllen, wurde ein Python-Tool entwickelt, das mit Hilfe von verschiedenen Superpixel-Algorithmen versucht, eine Auswertung von Kornbildern zu ermöglichen. Neben der Analyse der Bilder bietet das Tool die Möglichkeit, die Bilder vorzuverarbeiten und Ergebnisse per Maximum-Likelihood-Schätzung mit einer Verteilung der Wahl anzupassen, integriert also weitere Verarbeitungsschritte, die bisher separat geschahen, in ein Werkzeug. Das fertige Tool wurde zum einen in einem Praxistest, zum anderen mit Hilfe eines TPE in

Hinblick auf die Lösungsgüte evaluiert. Während des Praxisstests musste dabei der Umfang der Analyse-Funktionen erweitert werden, um mit einer vorher nicht gesehenen Art an Bildern umzugehen, wobei die Erweiterbarkeit getestet und bewiesen werden konnte. Mit dem neuen Funktionsumfang konnte nach Finden der richtigen Parameter bei der ersten Aufnahme eine deutliche Beschleunigung der Auswertungszeit festgestellt werden. Bei dem Versuch, per TPE die Einstellungen der Vorverarbeitungs- und Superpixel-Algorithmen so zu optimieren, dass die Ergebnisse möglichst nah an den Linienschnitt-Ergebnissen lagen, konnte für zwei der drei implementierten Analyse-Methoden ein sehr ähnliches Ergebnis für alle Testbilder generiert werden. Insgesamt kann der Lösungsansatz damit als Erfolg gewertet werden, die Anforderungen wurden zur Zufriedenheit der Mitarbeitenden der Uni Göttingen erfüllt. Insbesondere der Vergleich der Optimierungsverfahren SLIC und DBSCAN und die Notwendigkeit zur Anpassung des Tools im Praxistest verdeutlichen aber nochmal die Notwendigkeit der Erweiterbarkeit des Tools. Die Auswertungsverfahren waren unterschiedlich gut in der Auswertung unterschiedlicher Kornstrukturen - die Notwendigkeit anderer Verfahren bei neuen Bildern ist also naheliegend. Das dritte implementierte Analyseverfahren konnte im Rahmen dieser Masterarbeit leider nicht erfolgreich zur Replikation der händisch gewonnenen Verteilungen genutzt werden. Wahrscheinlich hängt dieses Problem mit der Anzahl der Freiheitsgrade des Verfahrens und der damit wesentlich komplexeren Optimierungsaufgabe zusammen. Mit mehr Zeit und Ressourcen oder einem anderen Optimierungsansatz könnte dieses Problem aber gelöst werden. Der Einsatz eines anderen Optimierungsverfahrens könnte noch aus zwei anderen kritischen Perspektiven auf das verwendete Verfahren in Betracht gezogen werden. Zum einen schreiben die Autoren von BOHB selbst, dass ein Problem ohne sinnvolle Budget-Definition nicht die Vorteile des Verfahrens voll ausnutzt (Biedenkapp & Hutter, 2018), außerdem würden selbst bei optimalen Bedingungen andere Verfahren bessere Ergebnisse auf Benchmark-Problemen zeigen als BOHB (Eggensperger et al., 2021). Aus Gründen der Vergleichbarkeit der Ergebnisse wurde im Rahmen dieser Arbeit aber auf einen Wechsel des Optimierungsverfahrens verzichtet. Zum

anderen könnte an dem Vorgehen, jedes Analyse-Verfahren für jedes Bild einzeln zu optimieren, kritisiert werden, dass Overfitting der wenigen Daten riskiert wird, die zur Verfügung stehen. Da das Tool aber nicht mit gefundenen Einstellungen generalisiert werden soll, sondern der intendierte Nutzen ist, die Parameter an ein gegebenes Bild anzupassen, ist das Overfitting im Sinne der Beeinträchtigung der prädiktiven Validität aber kein wirkliches Problem. Eine offene Frage ist aber, ob die durch die Optimierung gefundenen Kornverteilungen wirklich aus den ausgemessenen Körnern resultieren, oder - vor allem durch den setzbaren Parameter des maximalen Seitenverhältnisses - einfach eine Lösung gesucht wird, die dem Linienschnitt möglichst ähnliches Rauschen produziert. Dieser Punkt ist nicht von der Hand zu weisen, muss aber von Anwendern mit größerer Expertise beantwortet werden. Die Besprechung von Stichproben der Ergebnisse mit einem Mitarbeiter der Uni Göttingen wies nicht darauf hin, weitere Tests dazu bleiben aber nötig. Neben der hier benutzten Auswahl von klassischen Superpixelmethoden existiert noch ein großer Pool an anderen Möglichkeiten, wie zum Beispiel im Überblicksartikel von Stutz et al. (2018) beschrieben wird. Die Auswahl im Rahmen dieser Arbeit wurde auf Betrachtung der Lösungen von Stutz et al. gestützt, ist aber weder erschöpfend noch unbedingt die optimale, wenn auch die in Rücksprache mit Mitarbeitern der Uni Göttingen als am plausibelsten erscheinende. Weitere, systematische Tests der Anwendbarkeit einer Reihe anderer Verfahren scheint notwendig. Dies ist insbesondere so, da die erfolgreich getesteten Analyse-Verfahren SLIC und DBSCAN beide Cluster-Algorithmen einsetzen, um zu einer Lösung zu gelangen. Daneben werden bei Stutz et al. (2018) noch eine ganze andere Reihe von konzeptionellen Ansätzen beschrieben, wie zum Beispiel der auch ausgetestete Graph-Basierte Normalized Cuts Ansatz oder Ansätze der Energie-Optimierung wie zum Beispiel das Verfahren der Convexity Constrained Superpixels (Tasli et al., 2015), für deren Implementierung und Test die Dauer dieser Masterarbeit nicht ausreichte. Neben den klassischen Methoden zu Superpixel-Extraktion existieren auch Ansätze, Superpixel auf Basis neuronaler Netze zu erkennen und zu extrahieren, zum Beispiel nach dem bei Yang et al. (2020) beschriebenen Verfah-

ren. Dazu kommen noch die bereits beschriebenen Ansätze zur Extraktion von Kornstrukturen mit nicht Superpixel-basierten neuronalen Netzen (z.B. Latif et al., 2022; Podor et al., 2021). Da von der Universität Göttingen aber nur 31 Bilder mit Kornverteilungen zur Verfügung gestellt werden konnten und der erste Schritt eine Entwicklung eines Tools zur Auswertungsunterstützung war, konnte eine Implementation auch dieser Verfahren im Rahmen dieser Arbeit nicht geleistet werden. Als weiterer Ansatzpunkt zur Verbesserung der Analyse bleibt die Beobachtung aus dem Praxistest: Die Analyse mit Tool ist zwar ab dem zweiten Bild wesentlich schneller als per Linienschnitt, die Analyse des ersten Bildes dauert aber länger. Hier könnte angesetzt werden, indem ein Modell trainiert wird, das auf Basis von vorliegenden Bildern und den Einstellungen, die zu optimalen Ergebnissen führen, eine Einstellungs-Empfehlung abgibt. Umzusetzen wäre diese Empfehlung durch eine Ergänzung der Kornbild-Klasse um eine Methode, die aus einer Liste an vom Modell vorhergesagten, numerischen Werten einen Batch-Stack und eine Analyse-Voreinstellung generiert. Damit die Einstellungs-Vorschläge nutzbar wären müsste letzteres dann als Aktualisierung an die GUI zurückgegeben werden. Diese Einstellungen könnten dann von der nutzenden Person als Ausgangspunkt zur Analyse eines vorliegenden Bildes genutzt werden und könnten dabei helfen das zeitaufwändige erste Suchen nach möglichen Einstellungen zu beschleunigen. Als möglicher Kandidat für ein solches Modell könnte eine Modifikation des von Zhu & Wu (2021) vorgestellten Ansatzes zur Multi-Label Recognition mit Anpassung der Integration der Ergebnisse sein. Tests in diese Richtung waren auf Basis der zur Verfügung gestellten 31 Bilder aber leider nicht möglich, selbst bei Spiegeln anhand beider Achsen und damit einer Augmentierung des Datensatzes auf ein Volumen von 124 Bildern wäre nicht ausreichend. Mit mehr Bildern als Trainingsgrundlage könnte hier aber gut angesetzt werden, um den Nutzen des Tools schon beim ersten Bild zu erhöhen. Ein anderer Punkt, an dem zur Verbesserung der erstellten Lösung angesetzt werden kann, ist die Ergänzung der Methoden zur Vorverarbeitung der Bilder. Die Vorverarbeitung ist laut Prince (2012) nicht umsonst als mindestens so wichtig wie die Auswahl

des Analysemodells beschrieben. In dieser Arbeit wurden aus Zeitgründen nur die Vorverarbeitungsschritte implementiert und getestet, die bisher auch für die Vorbereitung des Linienschnitt-Verfahrens am Institut für Materialphysik eingesetzt wurden, daneben existiert aber ein ganzer Blumenstrauß an Alternativen. Auf paperswithcode.com, einer Übersichtswebsite zur Performance von Machine Learning Algorithmen gemessen an einer Reihe von Benchmark-Datensätzen, finden sich alleine für die Aufgabe der Rauschentfernung zum Zeitpunkt des Schreibens 905 Ansätze (*Papers with Code - Denoising*, n.d.). Neben diesen ML-basierten Vorverarbeitungsmethoden existiert natürlich auch eine große Anzahl getesteter und etablierter klassischer Ansätze, auch hier würde eine erschöpfende Betrachtung aber auch den Rahmen sprengen. Da aber das Tool so konstruiert ist, dass sowohl der Umfang der analytischen, als auch der Vorverarbeitungsmethoden ohne viel Aufwand erweiterbar ist, kann hier in Zukunft angesetzt werden, ohne dafür eine andere Lösung konstruieren zu müssen.

Kapitel 8

Literatur

- Abouelatta, O. B. (2013). Classification of copper alloys microstructure using image processing and neural network. *Journal of American Science*, 9(6), 213–223.
- Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11), 2274–2282.
- Akers, S., Kautz, E., Trevino-Gavito, A., Olszta, M., Matthews, B. E., Wang, L., Du, Y., & Spurgeon, S. R. (2021). Rapid and flexible segmentation of electron microscopy data using few-shot machine learning. *Npj Computational Materials*, 7(1, 1), 1–9. <https://doi.org/10.1038/s41524-021-00652-z>
- Ananyev, M., Medvedev, D., Gavrilyuk, A., Mitri, S., Demin, A., Malkov, V., & Tsiakaras, P. (2014). Cu and Gd co-doped BaCeO₃ proton conductors: Experimental vs SEM image algorithmic-segmentation results. *Electrochimica Acta*, 125, 371–379. <https://doi.org/10.1016/j.electacta.2013.12.161>
- Askeland, D. R. (1996). *Materialwissenschaften: Grundlagen, Übungen, Lösungen*. Spektrum Akad. Verlag.
- Basu, M. (2002). Gaussian-based edge-detection methods-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 32(3), 252–260. <https://doi.org/10.1109/TSMCC.2002.804448>

- Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for Hyper-Parameter Optimization. *Advances in Neural Information Processing Systems*, 24. <https://proceedings.neurips.cc/paper/2011/hash/86e8f7ab32cf12577bc2619bc635690-Abstract.html>
- Biedenkapp, A., & Hutter, F. (2018, July 26). *AutoML / BOHB: Robust and Efficient Hyperparameter Optimization at Scale*. AutoML.org. https://www.automl.org/blog_bohb/
- Buades, A., Coll, B., & Morel, J.-M. (2011). Non-Local Means Denoising. *Image Processing On Line*, 1, 208–212. https://doi.org/10.5201/ipl.2011.bcm_nlm
- DeCost, B. L., Lei, B., Francis, T., & Holm, E. A. (2019). High Throughput Quantitative Metallography for Complex Microstructures Using Deep Learning: A Case Study in Ultrahigh Carbon Steel. *Microscopy and Microanalysis*, 25(1), 21–29. <https://doi.org/10.1017/S1431927618015635>
- Dengiz, O., Smith, A. E., & Nettleship, I. (2005). Grain boundary detection in microstructure images using computational intelligence. *Computers in Industry*, 56(8-9), 854–866. <https://doi.org/10.1016/j.compind.2005.05.012>
- Eggensperger, K., Müller, P., Mallik, N., Feurer, M., Sass, R., Klein, A., Awad, N., Lindauer, M., & Hutter, F. (2021). *HPOBench: A Collection of Reproducible Multi-Fidelity Benchmark Problems for HPO* (No. arXiv:2109.06716). arXiv. <https://doi.org/10.48550/arXiv.2109.06716>
- Ester, M., Kriegel, H.-P., Sander, J., Wimmer, M., & Xu, X. (1998). Incremental Clustering for Mining in a Data Warehousing Environment. *Proceedings of the 24rd International Conference on Very Large Data Bases*, 323–333.
- Falkner, S., Klein, A., & Hutter, F. (2018). *BOHB: Robust and Efficient Hyperparameter Optimization at Scale* (No. arXiv:1807.01774). arXiv. <https://doi.org/10.48550/arXiv.1807.01774>
- Gefüge (Werkstoffkunde). (2021). In *Wikipedia*. [https://de.wikipedia.org/w/index.php?title=Gef%C3%BCge_\(Werkstoffkunde\)&oldid=214385861](https://de.wikipedia.org/w/index.php?title=Gef%C3%BCge_(Werkstoffkunde)&oldid=214385861)
- Harishvijey, A., & Benedict Raja, J. (2022). Automated technique for EEG signal processing to detect seizure with optimized Variable Gaussian Filter and Fuzzy

- RBFELM classifier. *Biomedical Signal Processing and Control*, 74, 103450. <https://doi.org/10.1016/j.bspc.2021.103450>
- Heilbronner, R. (2000). Automatic grain boundary detection and grain size analysis using polarization micrographs or orientation images. *Journal of Structural Geology*, 22(7), 969–981. [https://doi.org/10.1016/S0191-8141\(00\)00014-6](https://doi.org/10.1016/S0191-8141(00)00014-6)
- Heyn, E. (1903). Short reports from the metallurgical and metallographical laboratory of the royal mechanical and technical testing institute of charlottenburg. *The Metallographist*, 5, 39–64.
- HpBandSter*. (2022). AutoML-Freiburg-Hannover. <https://github.com/automl/HpBandSter> (Original work published 2017)
- Kim, H., Inoue, J., & Kasuya, T. (2020). Unsupervised microstructure segmentation by mimicking metallurgists' approach to pattern recognition. *Scientific Reports*, 10(1, 1), 17835. <https://doi.org/10.1038/s41598-020-74935-8>
- Kitagawa, G. (1994). The two-filter formula for smoothing and an implementation of the Gaussian-sum smoother. *Annals of the Institute of Statistical Mathematics*, 46(4), 605–623.
- Latif, G., Bouchard, K., Maitre, J., Back, A., & Bédard, L. P. (2022). Deep-Learning-Based Automatic Mineral Grain Segmentation and Recognition. *Minerals*, 12(4, 4), 455. <https://doi.org/10.3390/min12040455>
- Li, L., Jamieson, K. G., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. *ICLR (Poster)*.
- Li, M., Chen, D., Liu, S., & Liu, F. (2020). Metallographic Image Segmentation Method Based on Superpixels Algorithm and Transfer Learning. *2020 Chinese Control And Decision Conference (CCDC)*, 1922–1926. <https://doi.org/10.1093/CCDC49329.2020.9164466>
- Maitre, J., Bouchard, K., & Bédard, L. P. (2019). Mineral grains recognition using computer vision and machine learning. *Computers & Geosciences*, 130, 84–93. <https://doi.org/10.1016/j.cageo.2019.05.009>
- Matplotlib/matplotlib*. (2022). Matplotlib Developers. <https://github.com/matplotlib/matplotlib>

- otlib/matplotlib (Original work published 2011)
- Metallographie von rostfreiem Stahl.* (n.d.). Retrieved December 14, 2021, from <https://www.struers.com/de-DE/Knowledge/Materials/Stainless-Steel#etching>
- Normalized Cut — skimage v0.20.0.Dev0 docs.* (n.d.). Retrieved June 23, 2022, from https://scikit-image.org/docs/dev/auto_examples/segmentation/plot_ncut.html?highlight=normalization
- Öffentlichkeitsarbeit, G.-A.-U. G.-. (n.d.-a). *Personal - Georg-August-Universität Göttingen.* Retrieved May 9, 2022, from <https://www.uni-goettingen.de/de/652520.html>
- Öffentlichkeitsarbeit, G.-A.-U. G.-. (n.d.-b). *Studium und Lehre - Georg-August-Universität Göttingen.* Retrieved May 9, 2022, from <https://www.uni-goettingen.de/de/626487.html>
- Opencv-python: Wrapper package for OpenCV python bindings.* (Version 4.6.0.66). (n.d.). Retrieved July 1, 2022, from <https://github.com/skvark/opencv-python>
- Pandas: Powerful Python data analysis toolkit.* (2022). pandas. <https://github.com/pandas-dev/pandas> (Original work published 2010)
- Papers with Code - Denoising.* (n.d.). Retrieved July 7, 2022, from <https://paperswithcode.com/task/denoising>
- Podor, R., Le Goff, X., Lautru, J., Brau, H. P., Massonnet, M., & Clavier, N. (2021). SEraMic: A semi-automatic method for the segmentation of grain boundaries. *Journal of the European Ceramic Society*, 41(10), 5349–5358. <https://doi.org/10.1016/j.jeurceramsoc.2021.03.062>
- Prince, S. J. (2012). Part IV: Preprocessing. In *Computer vision: Models, learning, and inference* (pp. 321–354). Cambridge University Press.
- PySimpleGUI. (n.d.). *PySimpleGUI: Python GUIs for Humans. Launched in 2018. It's 2022 & PySimpleGUI is an ACTIVE & supported project. Super-simple to create custom GUI's. 325+ Demo programs & Cookbook for rapid start. Extensive documentation. Main docs at www.PySimpleGUI.org. Fun & your success*

- are the focus. Examples using Machine Learning (GUI, OpenCV Integration), Rainmeter Style Desktop Widgets, Matplotlib + Pyplot, PIL support, add GUI to command line scripts, PDF & Image Viewers. Great for beginners & advanced GUI programmers.* (Version 4.60.1) [Computer software]. Retrieved July 2, 2022, from <https://github.com/PySimpleGUI/PySimpleGUI>
- Rodriguez, M. Z., Comin, C. H., Casanova, D., Bruno, O. M., Amancio, D. R., Costa, L. da F., & Rodrigues, F. A. (2019). Clustering algorithms: A comparative approach. *PLOS ONE*, 14(1), e0210236. <https://doi.org/10.1371/journal.pone.0210236>
- Scikit-image: Image processing in Python.* (2022). Image Processing Toolbox for SciPy. <https://github.com/scikit-image/scikit-image> (Original work published 2011)
- Scikit-learn/scikit-learn.* (2022). scikit-learn. <https://github.com/scikit-learn/scikit-learn> (Original work published 2010)
- Scipy/scipy.* (2022). SciPy. <https://github.com/scipy/scipy> (Original work published 2011)
- Shen, J., Hao, X., Liang, Z., Liu, Y., Wang, W., & Shao, L. (2016). Real-Time Superpixel Segmentation by DBSCAN Clustering Algorithm. *IEEE Transactions on Image Processing*, 25(12), 5933–5942. <https://doi.org/10.1109/TIP.2016.2616302>
- Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 888–905. <https://doi.org/10.1109/34.868688>
- Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 25.
- Standard Test Methods for Determining Average Grain Size.* (2021, November 17). <https://www.astm.org/e0112-13.html>
- Stephens, M. A. (1974). EDF Statistics for Goodness of Fit and Some Comparisons. *Journal of the American Statistical Association*, 69(347), 730–737.

- <https://doi.org/10.1080/01621459.1974.10480196>
- Stutz, D., Hermans, A., & Leibe, B. (2018). Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166, 1–27. <https://doi.org/10.1016/j.cviu.2017.03.007>
- Tasli, H. E., Cigla, C., & Alatan, A. A. (2015). Convexity constrained efficient superpixel and supervoxel extraction. *Signal Processing: Image Communication*, 33, 71–85.
- Wang, M., Liu, X., Gao, Y., Ma, X., & Soomro, N. Q. (2017). Superpixel segmentation: A benchmark. *Signal Processing: Image Communication*, 56, 28–39. <https://doi.org/10.1016/j.image.2017.04.007>
- Wink, A. M., & Roerdink, J. B. (2004). Denoising functional MR images: A comparison of wavelet denoising and Gaussian smoothing. *IEEE Transactions on Medical Imaging*, 23(3), 374–387.
- Yang, F., Sun, Q., Jin, H., & Zhou, Z. (2020). *Superpixel Segmentation With Fully Convolutional Networks*. 13964–13973. https://openaccess.thecvf.com/content_CVPR_2020/html/Yang_Superpixel_Segmentation_With_Fully_Convolutional_Networks_CVPR_2020_paper.html
- Zhu, K., & Wu, J. (2021). *Residual Attention: A Simple but Effective Method for Multi-Label Recognition*. 184–193. https://openaccess.thecvf.com/content/ICCV2021/html/Zhu_Residual_Attention_A_Simple_but_Effective_Method_for_Multi-Label_Recognition_ICCV_2021_paper.html