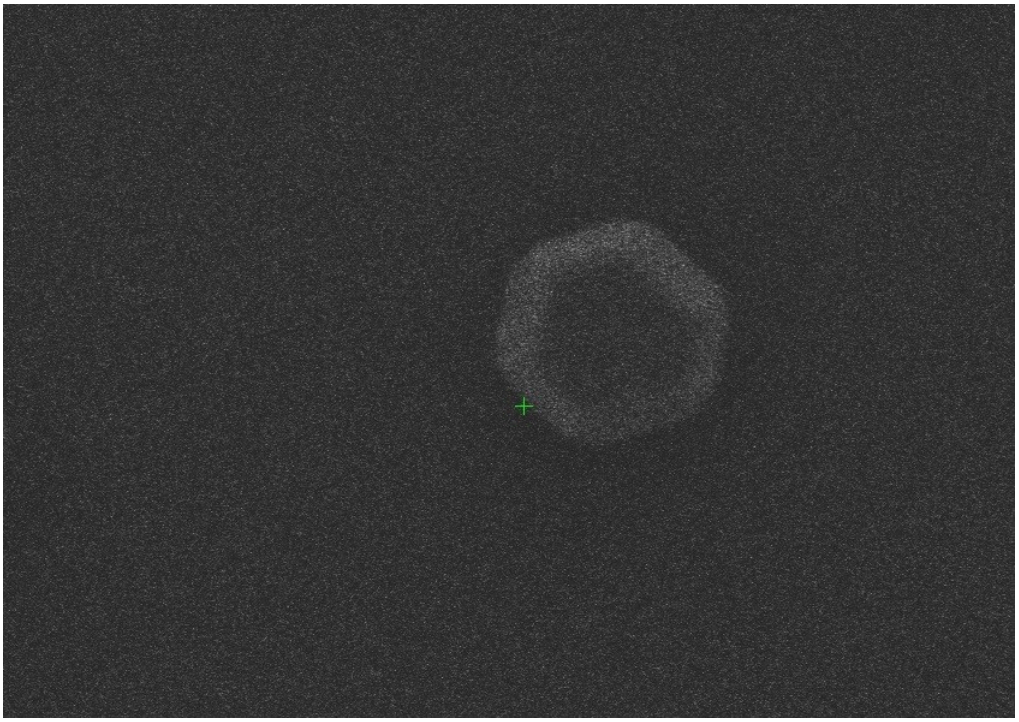
Hector Seminar

Karlsruher Institut für Technologie

NanoPEACH



Abschlussdokumentation der Kooperationsphase 2022/23

Durchgeführt am IAM

Betreut durch Prof. Dr. Christoph Kirchlechner und Konrad Prikoszovich

Iago Bischoff Montenegro, Kilian Quiring (Kurs KA17)

Inhaltsverzeichnis

Abstract

Nanoparticles find increasing utility in a wide variety of fields, with information on their exact size and shape being crucial to optimizing their properties. Scanning electron microscopy (SEM) is the fastest and most straightforward technique for obtaining this information. However, SEM images are distorted by the sample's slow movement during image acquisition. Here, we work on different methods to produce high-quality images without these distorsions. We introduce a new algorithm, inspired by concepts from the fields of astrophotography and machine learning, that combines multiple low-resolution images with high levels of background noise into a single high-quality image. We find that this method yields promising results with little effort, notwithstanding several inherent limitations. Further improvements to this algorithm's performance are possible, and it could be used as the basis for more complex methods allowing to obtain even better results.

1 Einleitung

1.1 Nanopartikel

Als Nanopartikel werden gemeinhin Partikel mit einer Größe zwischen 1 und 100 nm bezeichnet. In einigen Fällen wird der Begriff auch auf größere Partikel angewendet, wie es bei unserer Probe der Fall ist (siehe 2.1). Sie kommen in verschiedensten Varianten vor, die sich jeweils in Material und Form unterscheiden. Dabei weisen die Stoffe auf dieser Größenebene oft andere Eigenschaften auf als in alltäglichen Maßstäben, was sich in vielen Anwendungsbereichnen nutzen lässt.

Für diese Arbeit sind sogenannte Nanokristalle von Bedeutung. Im Gegensatz zu anderen Nanopartikeln zeichnen sie sich dadurch aus, dass ihre Atome in einem regelmäßigen Kristallgitter angeordnet sind. Nanokristalle finden vor allem als Katalysatoren Anwendung, da sie durch ihre geringe Größe ein äußerst hohes Oberfläche-zu-Volumen-Verhältnis besitzen.

Solche Nanokristalle werden durch eine Methode namens *solid-state dewetting* hergestellt. Ausgangspunkt ist dabei ein dünner Metallfilm auf einem Substrat. Dieser agglomeriert unter Wärmeeinwirkung langsam zu Nanopartikeln, die immer weiter wachsen und miteinander verschmelzen, um eine energetisch günstigere Konfiguration zu erreichen. Durch Modifikation der Ausgangsbedingungen lassen sich mit diesem Verfahren gezielt Nanopartikel mit bestimmten Größen und Eigenschaften herstellen.

1.2 Rasterelektronenmikroskopie

Ein Rasterelektronenmikroskop (REM) ist ein Mikroskop, das zur Bildgebung einen Elektronenstrahl statt sichbarem Licht verwendet und dank der sehr kurzen Wellenlänge der Elektronenstrahlung viel höhere Vergrößerungen und eine etwa um den Faktor 1000 bessere Auflösung erzielen kann als herkömmliche Lichtmikroskope. Es besteht aus vier Teilen:

1. Die **Probenkammer** ist luftdicht verschlossen und enthält den in alle Richtungen beweglichen Probenteller. Hier befinden sich auch die Elektronendetektoren.

2. Die **Strahlquelle** erzeugt einen feinen Strahl aus Elektronen, die sich mit etwa 10 Prozent der Lichtgeschwindigkeit fortbewegen. Dieser wird durch Magnetfelder gebündelt und mit hoher Präzision auf die Probe gerichtet

3. Die **Vakuumpumpe** erzeugt ein Hochvakuum in der Probenkammer und verhindert so Störungen des Elektronenstrahls durch Moleküle der Luft.

4. Ein **Computer** wertet die Daten der Detektoren aus und wandelt sie in ein Bild um.

Zur Bildgenerierung erfolgt eine zeilenweise Abtastung des Aufnahmefeldes mittels des Elektronenstrahls. Um den Grauwert für jeden Punkt zu ermitteln, werden mit verschiedenen Detektoren die von der Probe zurückgeworfenen Primär-Elektronen (d.h. Elektronen aus dem Elektronenstrahl), die Sekundär-Elektronen (d.h. von der Probe durch den Beschuss herausgelöste Elektronen) und die

von den Atomen der Probe emittierte Röntgenstrahlung untersucht. Für dieses Projekt kommt fast ausschließlich der Detektor für Sekundär-Elektronen zum Einsatz.

Die Bildauflösung hängt von der Geschwindigkeit der Abtastung des Bildfeldes durch den Elektronenstrahl ab. Wird der Elektronenstrahl nur sehr langsam über die Probe bewegt sind die Messwerte für die einzelnen Pixel äußerst genau, sodass das Bild hoch aufgelöst und detailreich ist. Die Aufnahme dauert dementsprechend lange; im Extremfall kann die Erzeugung eines einzelnen Bildes mehrere Minuten dauern. Wird der Elektronenstrahl hingegen schneller über die Probe bewegt, entsteht ein starkes Hintergrundrauschen, durch das viele Details verloren gehen - andererseits können Aufnahmen so viel schneller, teilweise nur in Sekundenbruchteilen, angefertigt werden.

1.2.1 Verzerrung der Aufnahmen durch Drift

Als Drift wird die allmähliche Verschiebung der Probe relativ zum Aufnahmeraster des REM bezeichnet. Sie wird durch verschiedene Faktoren verursacht, am wichtigsten sind die mechanische und thermische Setzung des Probentisches in der ersten Zeit nach der Inbetriebnahme des Mikroskops sowie Erschütterungen aus der Umgebung. Die hohe Präzision macht das REM sehr anfällig für solche Störungen. Schon Verschiebungen um wenige Nanometer haben einen großen Einfluss auf das Ergebnis. Durch diese Drift kommen bei der Aufnahme von Bildreihen zwei Arten von Fehlern auf:

- **Bild-zu-Bild-Verschiebung:** Von einer Aufnahme zur nächsten verschieben sich die Partikel im Bild. Bei längeren Aufnahmeserien muss daher regelmäßig das Bildfeld neu zentriert werden.
- **Bewegungsunschärfe:** Innerhalb einer Aufnahme werden die Partikel verzerrt. Dieser Effekt ist beim Arbeiten mit hoher Auflösung und dementsprechend langer Aufnahmedauer besonders problematisch - ähnlich wie auch bei gewöhnlichen Kameras Bilder mit längeren Belichtungszeiten häufig verwackelt sind.

Die Problemstellung dieses Projektes ist es, die Bewegungsunschärfe bei der Aufnahme hoch auflösender Bilder zu korrigieren oder zu umgehen, da es durch sie unmöglich wird, die korrekten Maße eines Nanokristalls direkt aus einem Bild zu ermitteln. Die Bild-zu-Bild-Verschiebung wirkt dabei als erschwerender Faktor, weil dadurch die Bilder einer Aufnahmeserie nicht ohne Zwischenschritte verglichen und kombiniert werden können. Bei beiden Aufnahmefehlern gilt, dass ihre Stärke proportional zur Aufnahmedauer ist.

1.2.2 Sonstige Aufnahmefehler

Zusätzlich zur Drift gibt es bei der Arbeit mit einem Rasterelektronenmikroskop noch weitere Faktoren, die die Aufnahme verschlechtern können. Am wichtigsten ist dabei die elektrostatische Aufladung: Ist die Probe durch ein isolierendes Material (z.B. Staub) verunreinigt, werden die vom Mikroskop verwendeten Elektronen nicht schnell genug abgeleitet und sammeln sich im Aufnahmefeld an. Dadurch entsteht ein unerwünschtes elektrisches Feld, das sowohl den Elektronenstrahl als auch die zu messenden

Elektronen etwas ablenkt und so die Messung verfälscht. Dieser Effekt lässt sich nur durch penible Reinhaltung der Probe und regelmäßiges Verschieben des Aufnahmefeldes minimieren.

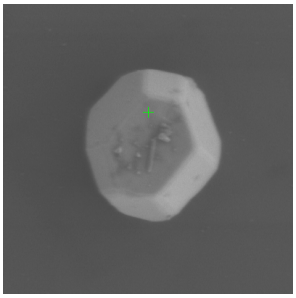
Des Weiteren zeigen sich insbesondere bei der Anwendung des Primärelektronendetektors entlang der Kanten der Nanopartikel horizontale Streifen, die sich über das gesamte Bild erstrecken. Diese Streifen werden auf Elektronen zurückgeführt, die verzögert detektiert wurden und daher eine falsche Position im Bild einnehmen. Aus diesem Grund wurde für den Rest des Projektes ausschließlich der Sekundärelektronendetektor verwendet.

2 Material und Methoden

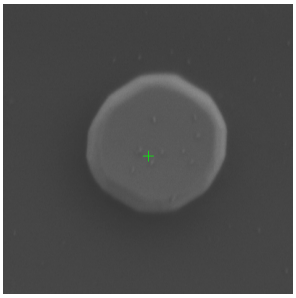
2.1 Probe

Untersuchungsgegenstand des Projekts ist eine Probe von in Metalloxid eingebetteten Platin- Nanokristallen, die von dem am Israel Institute of Technology tätigen Materialwissenschaftler Eugen Rabkin durch die oben erläuterte Methode des *solid-state dewetting* hergestellt wurden [2]. Die Partikel haben in der Regel eine Größe von 500 bis 1000 nm und kommen in unterschiedlichen Formen vor:

1. Eckige, eher höhere Partikel (in der Regel hexagonal)
2. Rundliche, flache Partikel (kreisförmig oder länglich)



(a) eckiges Partikel



(b) rundliches Partikel

2.2 Korrekturverfahren

Im Folgenden werden die verschiedenen Verfahren dargestellt, die zur Erstellung verzerrungsfreier Bilder in Erwägung gezogen werden. Die ersten zwei Ideen basieren allesamt darauf, ein hoch aufgelöstes, aber verzerrtes Bild zu korrigieren; der dritte Ansatz benutzt ein in der visuellen Astronomie verwendetes Verfahren.

2.2.1 Gerichtete Verzerrung aus zwei Bildern

Der erste Ansatz ist es, das Ausmaß der Drift aus zwei unmittelbar nacheinander aufgenommenen Bildern mittlerer bis hoher Auflösung desselben Bildfeldes zu ermitteln. Daraus soll dann die benötigte Entzerrung bzw. Scherung ermittelt werden, um eines der Ausgangsbilder zu korrigieren.

Es besteht allerdings das Problem, dass dabei Veränderungen der Drift während der Aufnahme der beiden Bilder (die je nach Auflösung insgesamt 10-20 Sekunden in Anspruch nimmt) nicht berücksichtigt werden, sodass die Korrektur die Drift nur in Teilen beseitigen würde. Daher wird das Verfahren verworfen; allerdings wird die Idee, die Stärke der Drift von Bild zu Bild zu ermitteln, im endgültigen Verfahren wieder aufgegriffen.

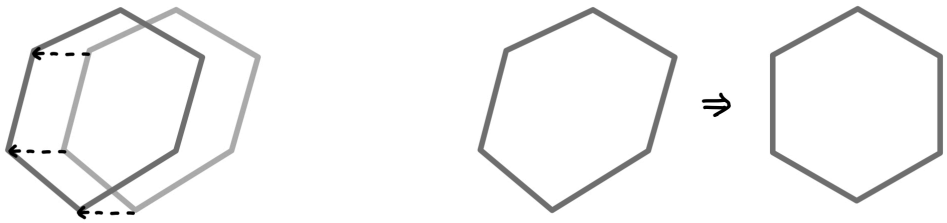


Abbildung 2: Ermitteln des Driftvektors aus zwei Bildern (links); anschließend Entzerrung eines der Bilder anhand des Vektors

2.2.2 Höhe-Breite-Korrektur aus rotierten Bildpaaren

Das REM durchrastert das Aufnahmefeld zeilenweise, sodass die Aufnahme einer einzigen Zeile selbst bei hoch auflösenden Bildern nur wenige Millisekunden dauert. Auf dieser Zeitskala macht sich die Drift kaum bemerkbar, sodass die Breite der Partikel korrekt abgebildet wird. Die Höhe des Partikels hingegen kann durch die Drift stark verändert werden.

Daraus ergibt sich der Ansatz, aus der maximalen Breite eines Partikels auf die eigentliche Höhe in einer um 90° gedrehten Aufnahme zu schließen und diese entsprechend zu dehnen oder zu stauchen. Dieser Prozess soll mit mehreren Bildpaaren mit jeweils unterschiedlichen Winkeln wiederholt werden. Aus jedem Bildpaar ließe sich so eine bestimmte Verzerrung in vertikaler Richtung des ersten Bildes berechnen; die Vektorsumme all dieser einzelnen Verzerrungen sollte dann die benötigte Gesamtverzerrung ergeben.

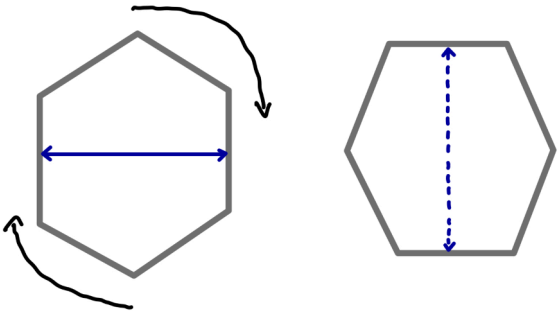


Abbildung 3: Die gestrichelte blaue Linie (verzerrt) muss genauso lang sein wie die durchgezogene blaue Linie. Daraus kann ein Verzerrungsvektor ermittelt werden

Dieser Ansatz ist zwar eigentlich wesentlich präziser als die gerichtete Verzerrung aus zwei Bildern, löst aber nicht das Problem der zeitlichen Veränderung der Drift. Wenn die zu Grunde liegende Verzerrung nicht in allen Bildpaaren genau dieselbe ist, lassen sich auch die Ergebnisse nicht einfach miteinander verrechnen. Zwar wäre es möglich, diese Probleme durch geschickte Gestaltung des Aufnahmeprozesses zu minimieren (indem man beispielsweise Bilder mit möglichst unterschiedlichen Winkeln direkt

nacheinander aufnimmt); es erscheint jedoch sinnvoll, nach einem weniger fehleranfälligen und einfacher umzusetzenden Verfahren zu suchen.

2.2.3 Shift-and-Add-Verfahren

Dieses Verfahren basiert auf der in der Amateurastronomie üblichen Methode des *lucky imaging*. Dabei werden hunderte von mit einer Hochgeschwindigkeitskamera aufgenommenen Bildern eines Himmelsobjekts gemeinsam zentriert und überlagert, um ein einziges hochaufgelöstes und unverzerrtes Ergebnis zu erzielen [1].

In der vorliegenden Arbeit werden dazu niedrig aufgelöste Bilder mit kurzer Aufnahmezeit (siehe Abschnitt 2.3.2) zu einem höher aufgelösten Summenbild überlagert. Dieses Bild wäre ähnlich wie die Einzelbilder nahezu verzerrungsfrei. Dazu muss jedoch ein Algorithmus entwickelt werden, um den genauen Betrag der von der Drift verursachten Verschiebung von Bild zu Bild zu ermitteln und so eine sinnvolle Überlagerung zu ermöglichen.

In diesem Zusammenhang wird das Konzept der *cost function* aus dem Bereich der neuronalen Netzwerke genutzt: einer Funktion, deren Wert in Abhängigkeit von der Bildverschiebung ein Maß für die Güte der Überlagerung wäre. Dazu erscheint zunächst die mittlere Differenz zwischen dem Wert der übereinanderliegenden Pixeln p_1 und p_2 sinnvoll, da diese größer wäre, wenn das Pixel in der Überlagerung in einem der Bilder zum Hintergrund und im anderen zum Partikel gehörte - Hintergrund und Partikel haben ja unterschiedliche Graustufenwerte. Je schlechter die Überlagerung, desto häufiger ist das der Fall (d.h. desto größer sind die roten Bereiche in 4), und desto höher ist der Wert der *cost function*.

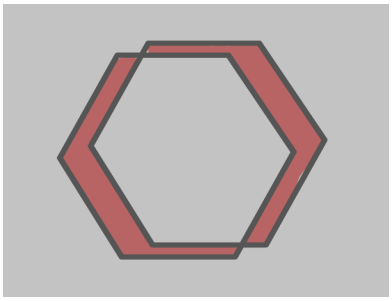


Abbildung 4: In den grauen Bereichen gehören die Pixel beider Bilder jeweils zur selben Kategorie. In den roten Bereichen gehört das Pixel eines Bildes zum Partikel und das des anderen zum Hintergrund

Allerdings wird dabei das durch den Aufnahmeprozess bedingte Hintergrundrauschen, genauer die dadurch entstandenen geringen Abweichungen bei allen Pixeln, zu stark gewichtet, wodurch die Werte der cost function kaum aussagekräftig sind. Um dieses Problem zumindest teilweise zu beheben wurden drei verschiedene Maßnahmen ergriffen:

1. Vorverarbeitung der Bilder mit einem Gauß'schen Filter, durch den das gesamte Bild ein wenig unschärfer wird. So wird der Graustufenwert der einzelnen Pixel ein wenig an den Mittelwert

in ihrer unmittelbaren Umgebung angepasst. Die Standardabweichung in der Verteilung der Graustufenwerte nimmt ab.

2. Verwendung der potenzierten Differenz $(p_2 - p_1)^4$ anstatt der einfachen Differenz $|p_2 - p_1|$ zwischen den Pixeln. So fallen kleine Abweichungen, die durch das trotz Vorverarbeitung verbliebene Hintergrundrauschen verursacht werden, weniger ins Gewicht als größere Differenzen zwischen Partikel und Hintergrund.

3. Berechnung eines gewichteten Mittelwertes mit den Werten der *cost function* für die Verschiebungswerte in unmittelbarer Umgebung der eigentlichen Position.

Bei der Suche nach dem Minimum dieser Funktion, das die optimale Überlagerung darstellt, stellt sich aufgrund der diskreten Natur der Eingabewerte heraus, dass Methoden wie *gradient descent* nicht effektiv angewandt werden können. Stattdessen wird das Minimum durch das systematische Absuchen eines quadratischen Suchfelds ermittelt. Dabei wird das Gebiet um das bisher gefundene Minimum in immer kleineren Suchintervallen abgesucht, bis das Suchraster so eng ist, dass die gemessenen Pixel direkt benachbart sind und das Ergebnis somit die maximale Präzision erreicht hat.

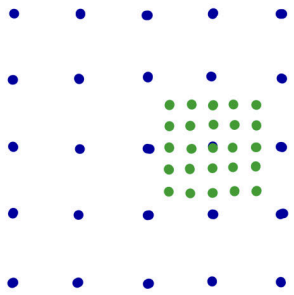


Abbildung 5: Im ersten Schritt (blau) wird in einem großen Suchfeld an einigen Punkten der Wert der cost function bestimmt. Dort, wo die cost function minimal ist, wird erneut mit einem kleineren Raster gesucht (grün)

Dieser Prozess wird für alle Bilder einer Aufnahmeserie wiederholt. Dadurch erhält man die genauen Bild-zu-Bild-Verschiebungen für alle Einzelbilder, sodass man die Drift herausrechnen und die Bilder anschließend zu einem Gesamtbild überlagern kann, indem man ihren Mittelwert bildet. Auf diese Weise wird das durch die geringe Aufnahmedauer bedingte Hintergrundrauschen weitestgehend beseitigt und es stehen verhältnismäßig klare Bilder für die weitere Untersuchung zur Verfügung.

2.3 Aufnahme der Rohbilder

2.3.1 Automatisierung des Aufnahmeprozesses

Für erste Tests werden die Bilder per Hand aufgenommen; für die Messreihen selbst ist jedoch ein automatischer Aufnahmemechanismus notwendig, um die Präzision und Reproduzierbarkeit der Daten

sicherzustellen. Das erweist sich jedoch zunächst als schwierig, da die Herstellersoftware des REM es lediglich erlaubt, sehr einfache Prozesse über Makros zu automatisieren; selbst die wiederholte Aufnahme von Bildern mit for- oder while-Schleifen ist damit nicht möglich.

Dieses Problem kann durch die Verwendung einer spezialisierten Bibliothek mit Python-Skripten umgangen werden, über die sich das REM ansteuern lässt. Das erlaubt viel komplexere und detaillierte Aufnahmeprozesse: Alle Aufnahmeparameter können direkt festgelegt werden, sodass bis auf das Zentrieren der Partikel im Gesichtsfeld der gesamte Aufnahmeprozess vollautomatisch ablaufen kann.

2.3.2 Erste Messreihen

In der Anfangsphase des Projekts wurden mehrere Aufnahmeserien erstellt. Diese Serien sind noch nicht mit einem vollständig ausgearbeiteten Korrekturalgorithmus verknüpft und sind daher nicht unmittelbar zur Auswertung geeignet. Sie dienen vor allem dem Zweck, Erfahrung im Umgang mit dem REM zu sammeln und die Einstellungen zu finden, bei denen Artefakte (siehe Abschnitt 1.2.2) möglichst minimiert werden können. Meist werden dabei verhältnismäßig niedrige *scan-speeds* mit Aufnahmedauern von 10-20 Sekunden verwendet.

Die ersten systematischen Messreihen sind auf ein anfangs in Erwägung gezogenes Verfahren, das auf dem Vergleich von rotierten Bildern derselben Partikel basiert (siehe Abschnitt 2.3), abgestimmt. Sie bestehen daher aus 12 bis 36 Bildern, die mit jeweils um 5 bis 15 Grad veränderten Rotationswinkeln aufgenommen werden. Sie haben eine eher geringe Vergrößerung und enthalten jeweils mehrere Partikel. Dies erschwert die Auswertung und lässt auch den Einfluss der Drift nicht deutlich zutage treten.

In weiteren Aufnahmeserien wird versucht, diese beiden Probleme durch höhere Vergrößerung und noch längere Aufnahmezeiten zu beheben, was allerdings nur in Teilen gelingt - einer der Gründe, warum dieses Korrekturverfahren schließlich verworfen wird.

2.3.3 Endgültige Messreihe

Zur Verwendung des shift-and-add-Verfahrens (siehe 2.2.4) sind Aufnahmeserien mit völlig unterschiedlichen Aufnahmeparametern erforderlich: Serien mit zahlreichen, niedrig aufgelösten Bildern, deren Aufnahmedauern jeweils kurz genug sind, um die Verzerrung durch die Drift in jedem Einzelbild auf einem vernachlässigbar geringen Niveau zu halten. Schließlich werden 6 Aufnahmeserien mit je 20 Einzelbildern angefertigt, die jeweils in etwa 380 ms aufgenommen wurden. Die Vergrößerung variierte von Reihe zu Reihe; detaillierte Daten sind im (Anhang 2) zu finden.

2.4 Vorverarbeitung der Rohbilder

Während des gesamten Projekts kommt ein Rastelektronenmikroskop des Modells ZEISS GeminiSEM 460 zum Einsatz. Da die untersuchten Partikel verhältnismäßig groß sind (siehe Abschnitt 2.1), werden die Grenzen dieses Geräts nicht vollständig ausgereizt: Die Aufnahmen werden üblicherweise bei Vergrößerungen von 30.000- bis 60.000-fach und einer Beschleunigungsspannung von 2kV durchgeführt, was innerhalb der Kapazitäten des Mikroskops für hochauflösende Bilder liegt. Die Bilder werden durch

die Software des Mikroskops als verlustfreie Bilder im TIFF-Format gespeichert. Jedem Pixel ist ein Graustufenwert zwischen 0 und 255 zugewiesen.

Für einen der Korrekturansätze ist es nötig oder zumindest hilfreich, die Rohbilder zunächst in Bitmaps umzuwandeln, bei denen die Pixel des Partikels den Wert 255 und die Pixel des Hintergrunds den Wert 0 haben. Das erleichtert beispielsweise die genaue Bestimmung von Höhe, Breite und Form der Nanopartikel auf dem Bild. Diese erscheinen in den Bildern meist etwas heller als der Hintergrund; sodass zur Trennung ein einfacher Grenzwert verwendet werden kann - alle Pixel mit einem Graustufenwert oberhalb dessen zählen mit hoher Wahrscheinlichkeit zu einem Partikel, alle mit einem Wert darunter zählen zum Hintergrund.

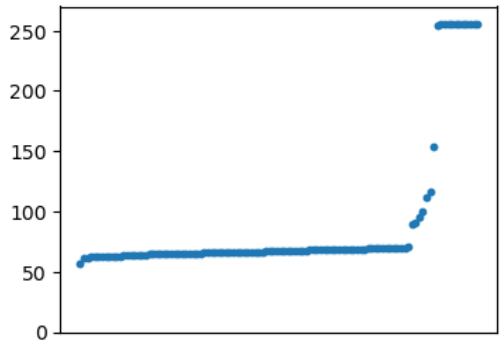


Abbildung 6: Grauwerte für 110 Pixel in einem Bild

Um diesen Grenzwert zu ermitteln, werden zunächst repräsentativ für das gesamte Bild einige Pixel (in diesem Fall 110) ausgewählt. Deren Grauwerte, die zwischen 0 und 255 liegen, werden in aufsteigender Reihenfolge geordnet. Wie in Abbildung 6 zu erkennen, können dabei die Pixel unterschiedlichen Teilen des Bildes zugeordnet werden: Die meisten Werte liegen etwa zwischen 60 und 70 und gehören zum Hintergrund des Bildes. Einige Werte liegen beim Maximum von 255 und gehören zur Leiste, die in den Rohbildern Informationen zu den Aufnahmeparametern enthält. Die wenigen Werte dazwischen sind Pixeln des Partikels zuzuordnen.

Wie aus dem deutlichen Übergang im Diagramm von Hintergrund zu Partikel hervorgeht, kann somit ein ziemlich genauer Grenzwert für die Bitmap-Erzeugung festgelegt werden. Mit dem Verfahren des *connected-component labelling*, bei der das gesamte Bild in große, zusammenhängende Flächen mit demselben Wert unterteilt wird, können dann Pixel, die durch Fehlmessung des REM einen im Vergleich zu ihren Nachbarpixeln viel zu hohen oder niedrigen Wert haben und damit falsch zugeordnet wurden, korrigiert werden.

2.5 Informatische Umsetzung

2.5.1 Programmierumgebung

Die Korrekturalgorithmen werden in Python in der Jupyter-Notebook-Programmierumgebung implementiert; derselben Schnittstelle, die auch zur Ansteuerung des Mikroskops verwendet wird. Python ist

sehr gut für solche Datenverarbeitungsaufgaben geeignet und kann durch zahlreiche externe Bibliotheken (*libraries*) an jeden Verwendungszweck angepasst werden. Für dieses Projekt werden die folgenden *libraries* verwendet:

- **OpenCV** für Import und Export der Bilder (d.h. Umwandlung der Rohdaten von .tiff-Dateien zu Numpy-Arrays und später Umwandlung des Ergebnisses zu .png-Datei) sowie für Vorbereitung durch Gauß'schen Filter
- **Numpy** für die Korrekturalgorithmen selbst (Formatierung der Daten als Numpy-Arrays ermöglicht schnellere Rechenzeiten)
- **OS** für Import und Export von Dateien
- **Matplotlib** für die Ausgabe von Diagrammen

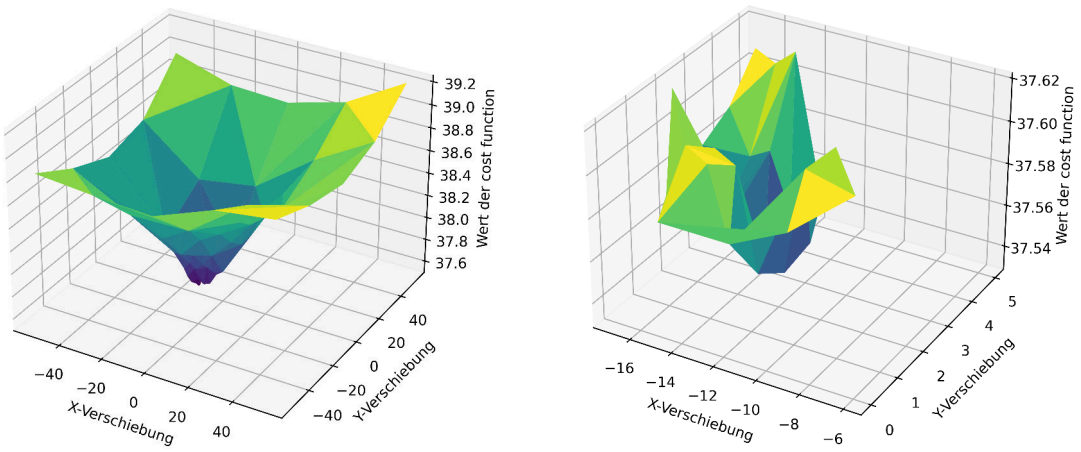
2.5.2 Einsatz von KI-Assistenten

Die Verwendung des KI-Assistenzsystems ChatGPT ist vor allem in der späteren Phase des Projektes systematisch Teil des Programmierprozesses. Zum einen wird es dazu verwendet, eine konkrete Vorstellung zu einem (Teil-)Algorithmus in Python-Code umzusetzen und die dazu nötigen *libraries* und Befehle zu finden, zum anderen kommt es beim debuggen des Codes zum Einsatz.

3 Ergebnisse

3.1 Untersuchung des Algorithmus

Die Diagramme 7 zeigen alle bei einer systematischen Suche nach der optimalen Überlagerung gemessenen Werte der *cost function*. Anhand dessen lässt sich beurteilen, wie gut das Konzept tatsächlich funktioniert hat.



(a) Globalansicht (b) Umgebung des Minimums

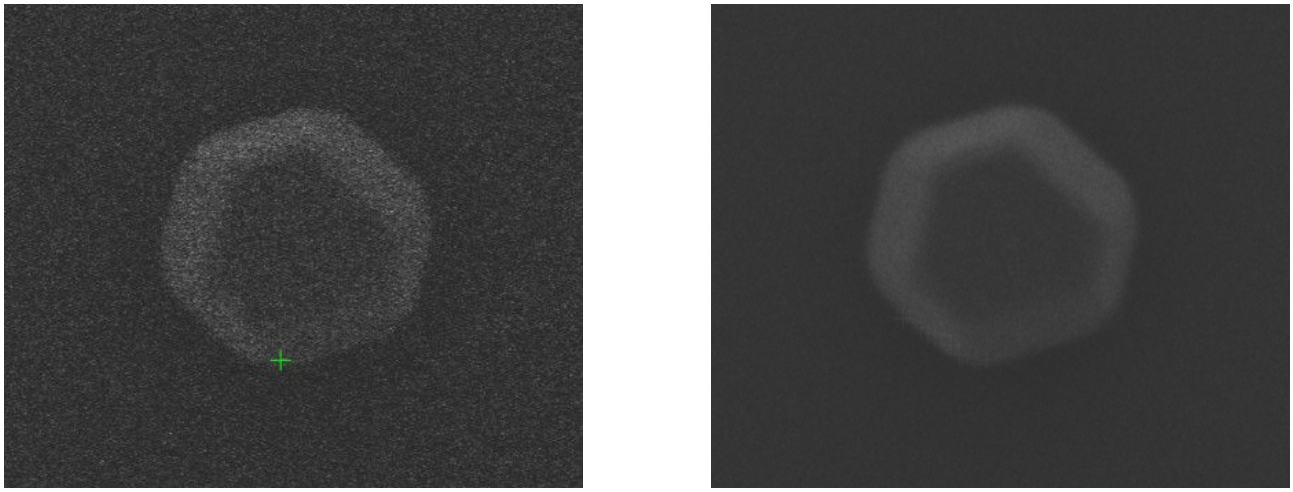
Abbildung 7: Beispielhafter Verlauf der *cost function* bei der Überlagerung von zwei Bildern in Abhängigkeit der Verschiebung in x- und y-Richtung

Das linke Diagramm (a) zeigt eine deutliche trichterartige Form, wobei der tiefste Punkt die optimale Überlagerung darstellt, von dem aus die Werte in alle Richtungen zunehmen. Das beweist, dass der Ansatz im Prinzip funktioniert; die Summe der potenzierten Differenzen zwischen den Pixeln hat ein klares Minimum bei einer bestimmten Verschiebung.

Im rechten Diagramm (b) mit näher zusammenliegenden Werten wird jedoch eine Einschränkung des Verfahrens deutlich: In der direkten Umgebung des Minimums sind die Werte der *cost function* viel chaotischer verteilt, da die Unterschiede teilweise nur 0.01 % betragen - ein deutlicher Trichter ist nicht erkennbar.

3.2 Qualitätsgewinn durch Überlagerung

Im Vergleich zu den Rohbildern ist das Rauschen im überlagerten Bild sehr stark reduziert. Die einzelnen Flächen wirken homogener, sodass sich das Partikel deutlicher vom Hintergrund abhebt. Das erleichert auch die weitere Arbeit mit dem Bild, z.B. die Umwandlung in eine Bitmap. Trotzdem ist der Rand des Partikels immer noch leicht verschwommen und nicht so klar definiert wie bei Bildern mit längerer Aufnahmedauer und höherer Auflösung.



(a) Rohbild (b) Ergebnis

Abbildung 8: Vergleich eines Partikes in einem Rohbild und im überlagerten Bild

3.3 Fehlerbetrachtung

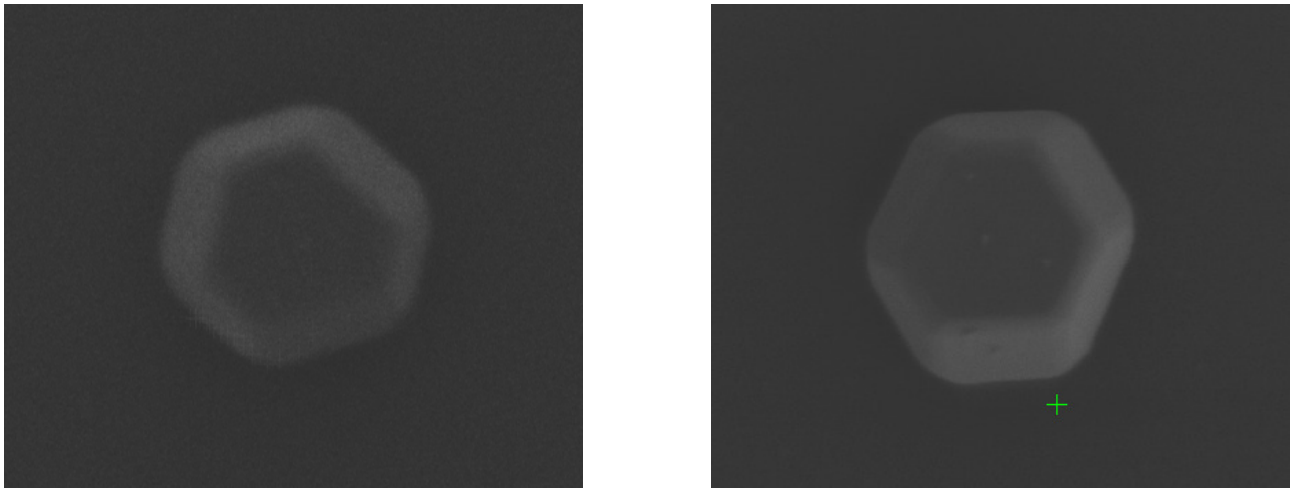
3.3.1 Verbleibende Aufnahmefehler

Der verschwommene Rand ist das größte nach der Überlagerung verbleibende Artefakt in den Bildern; er erschwert eine genaue Vermessung der Partikel. Dadurch kann die Größenbestimmung der Partikel nur mit einer Präzision von ± 5 Pixeln, was je nach Größe des Partikels etwa 1-2 % entspricht - die Vorteile der nahezu driftfreien Abbildung können nicht voll ausgeschöpft werden.

Wichtigster Grund dafür ist, dass bei der Überlagerung die vorverarbeiteten Bilder verwendet werden, die mit einem Gauß'schen Filter weichgezeichnet worden sind - somit ist auch die Überlagerung weichgezeichnet. Würden stattdessen die Originalbilder überlagert und die weichgezeichneten Bilder nur zum Bestimmen der Drift verwendet werden, wäre mit einer deutlich höheren Bildqualität zu rechnen.

Ein Teil des Fehlers ist jedoch auch auf die in Abschnitt 3.1 identifizierten Ungenauigkeiten in der Bestimmung der optimalen Überlagerung mit der *cost function* zurückzuführen. Möglicherweise sorgen die durch das Hintergrundrauschen bedingte Varianz dafür, dass die Überlagerung in manchen Fällen um ein paar Pixel vom eigentlichen Optimum abweicht.

Schließlich trägt auch der in den Einzelbildern verbleibende Drift zur Unschärfe bei. Trotz der sehr geringen Aufnahmedauer kann das Partikel in jedem Einzelbild so um 1-2 Pixel verzerrt werden. Da diese Verzerrung von Bild zu Bild anders sein kann, macht sich das im Ergebnis als Unschärfe am Rand bemerkbar. Dieser Effekt ist jedoch verhältnismäßig schwach und schränkt die Vermessung der Partikel kaum ein.



(a) überlagertes Bild (b) hoch aufgelöstes Bild

Abbildung 9: Vergleich eines überlagerten Bilds und eines hoch aufgelösten Einzelbildes

3.3.2 Einschränkungen des Algorithmus

Beim Testen des Korrekturverfahrens mit verschiedenen Aufnahmereihen machen sich verschiedene Einschränkungen bemerkbar, die bei der Verwendung des Algorithmus bedacht werden müssen:

- Die *cost function* gibt unerwünschte Werte aus, wenn sich mehrere Partikel im Bildfeld befinden. Um das zu vermeiden, sollte nur ein Partikel zentriert und die Vergrößerung möglichst hoch gewählt werden - das ermöglicht auch präzisere Messungen.
- Eine potentielle Drehung des Bildfeldes von Aufnahme zu Aufnahme wird nicht berücksichtigt. Da solche Effekte beim REM jedoch kaum vorkommen, kann dieses Problem weitestgehend vernachlässigt werden.
- Grundlage für das Funktionieren der *cost function* ist ein deutlicher Unterschied zwischen den Graustufen von Partikel und Hintergrund. Ist dieser im Verhältnis zum Rauschen nicht stark genug, kann die Funktion nicht zur Suche nach einer Überlagerung verwendet werden.

4 Ausblick

Insgesamt sind die mit dem entwickelten Verfahren erzielten Ergebnisse nicht völlig zufriedenstellend. Es besteht allerdings noch Potential für Verbesserungen; außerdem kann das Verfahren auch in Kombination mit anderen Methoden weiterentwickelt werden, um noch bessere Aufnahmen von Nanopartikeln zu ermöglichen.

4.1 Optimierungen

Der Algorithmus kann mit gewöhnlichen Computern in recht kurzer Zeit ausgeführt werden; für eine Echtzeitverarbeitung der Bilder während des Aufnahmeprozesses ist er dennoch zu rechenintensiv. Neben der Verwendung von leistungsstärkerer Hardware oder einer schnelleren Programmiersprache (z.B. C++ oder Mojo) gibt es auch Möglichkeiten, den Algorithmus effizienter und damit schneller zu gestalten.

4.1.1 Bild-zu-Bild-Messung beschleunigen

Die systematische Suche nach dem Mimimum der *cost function* durch die sukzessive Verfeinerung des Suchfeldes kann so optimiert werden, dass dabei weniger Messungen durchgeführt werden müssen.

In der vorliegenden Version gibt es 4 Durchläufe mit je 25 Messungen. Das erste Suchfeld ist 135 x 135 Pixel groß; bei jedem weiteren Durchlauf wird es um den Faktor 3 kleiner. Werden diese Parameter jedoch anders gewählt, kann die Anzahl an Messungen beträchtlich reduziert werden, wie die 1 zeigt:

Tabelle 1: Anzahl an Messungen für verschiedene Parameter

Erstes Suchfeld (Messpunkte)	Durchläufe	Faktor	**Messungen insgesamt**
135 x 135 (25)	4	3	**100**
80 x 80 (25)	3	4	**75**
128 x 128 (9)	6	2	**54**

Zudem fallen mit der aktuellen Methode einige Mehrfachmessungen an, da z.B. das Optimum des letzten Suchdurchgangs als Mitte des nächsten Durchganges erneut gemessen wird. Hier ließe sich etwas Rechenaufwand einsparen, wenn die Messdaten gespeichert würden und im Vorfeld überprüft werden könnte, ob der zu messende Punkt schon darunter vorkommt.

4.1.2 Weniger Bild-zu-Bild-Messungen

Die Tests des Algorithmus wurden mit Serien von je 20 Einzelbildern durchgeführt; es mussten also 19 Verschiebungen ermittelt werden. Es ist davon auszugehen, dass auch mit kürzeren Serien (beispielsweise mit 10 Einzelbildern) zufriedenstellende Resultate erzielt werden könnten; dadurch würden sowohl die Gesamtaufnahmedauer als auch der Rechenaufwand für die Korrektur reduziert werden. Hier müsste für jede konkrete Anwendung eine Abwägung zwischen Aufwand und Qualität des Ergebnisses getroffen werden.

Da die Drift oft über mehrere Einzelbilder hinweg recht konstant ist, wäre es außerdem denkbar, die Bild-zu-Bild-Verschiebung nicht für jedes Bildpaar, sondern lediglich in regelmäßigen Intervallen in der Aufnahmeserie zu bestimmen und vereinfachend anzunehmen, dass die Verschiebung bei den dazwischen liegenden Bilder dieselbe ist. Würde beispielsweise bei einer Aufnahmeserie von 16 Bildern die Drift nur bei jedem dritten Bild bestimmt werden, wären 6 Bestimmungen ausreichend.

4.2 Mögliche Weiterentwicklung

Als Werkzeug zur schnellen Bestimmung von Stärke und Richtung des Drifts kann der entwickelte Algorithmus gut als Grundlage für andere Korrekturverfahren verwendet werden. Ist nämlich die Drift während der Aufnahme eines Bildes mit höherer Auflösung bekannt, kann dieses im Nachhinein entzerrt werden.

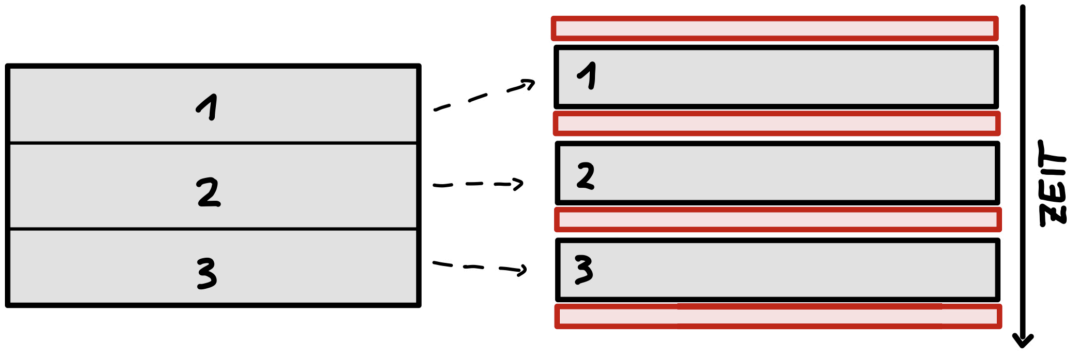


Abbildung 10: Zwischen den Streifen des hoch aufgelösten Bildes (grau) wird jeweils ein niedrig aufgelöstes Bild (rot) zur Driftbestimmung aufgenommen

Am besten wäre das Ergebnis, wenn das höher aufgelöste Bild aus mehreren Streifen zusammengesetzt wird, deren Aufnahme jeweils nur ein oder zwei Sekunden dauert, sodass die Drift während seiner Aufnahme möglichst konstant bleibt. Vor und nach jedem Streifen in hoher Auflösung wird jeweils zur Ermittlung der Drift ein Bild mit niedriger Auflösung angefertigt (siehe Abb. 10). Die Streifen werden dann einzeln entzerrt und schließlich zu einem Gesamtbild zusammengesetzt.

5 Fazit

Für seine primäre Anwendung, das Erstellen verzerrungsfreier Bilder von Nanopartikeln, erscheint der entwickelte Algorithmus gut geeignet. Das Ergebnisbild enthält kaum Rauschen und kann für die weitere Vermessung gut in eine Bitmap umgewandelt werden. Die Aufnahmedauer für die gesamte Serie ist vergleichbar mit der für ein höher aufgelöstes (aber verzerrtes) Einzelbild mit niedrigerer Aufnahmegeschwindigkeit - besonders, wenn die Verarbeitung in Echtzeit geschehen kann.

Ohne die Überlagerung der Einzelbilder kann der Algorithmus allerdings auch verwendet werden, um den Drift im REM über einen bestimmten Zeitraum hinweg mit recht großer Genauigkeit zu ermitteln. Durch eine systematische Untersuchung des Drifts zu verschiedenen Zeitpunkten und unter verschiedenen Bedingungen könnten die konkreten Gründe des Drifts ermittelt und Methoden für seine Begrenzung entwickelt werden - was wiederum Bilder von besserer Qualität ermöglichen würde. Zudem wäre so genauer bekannt, welche Verzerrungen in welchem Maße vorkommen; das würde die Entwicklung von ausgefeilteren und wirksameren Korrekturverfahren erleichtern.

Danksagung

Wir bedanken uns herzlich bei allen, die uns die Arbeit an diesem Projekt ermöglicht haben. Dazu gehört zuallererst unser Betreuer Konrad Prikoszovich, der uns unterstützend zur Seite stand. Wir danken genauso Prof. Dr. Christoph Kirchlechner, dem Institutsleiter des IAM-MMI, der uns die Arbeit am Institut ermöglicht und sich ebenfalls sehr für unser Projekt engagiert hat.

Unser Dank gilt ebenfalls Dr. Hans-Werner und Josephine Hector dafür, dass sie das Hector-Seminar gestiftet und uns dadurch über die Jahre all die vielfältigen Eindrücke und vertiefenden Einblicke in das wissenschaftliche Arbeiten ermöglicht haben.

Zuletzt möchten wir uns bei allen Kursleiterinnen und Kursleitern bedanken, die uns in den letzten sechs Jahren am Hector-Seminar begleitet haben. Besonders hervorheben möchten wir Norbert Krieg, der uns für die Dauer dieses Projektes betreut hat.

Literatur

[1] University of Cambridge. *Amateur Lucky Imaging*. 2011. URL: https://www.ast.cam.ac.uk/research/instrumentation.surveys.and.projects/lucky.imaging/latest.results/amateur.lucky.imaging (besucht am 16. 07. 2023).

[2] Jérôme Carnis u. a. "Twin boundary migration in an individual platinum nanocrystal during catalytic CO oxidation". In: *Nature communications* 12.1 (2021), S. 5385.

Anhang

Code

Import der benötigten Module:

```
import cv2
import numpy as np
import os
```

Definition der Funktionen für das *preprocessing* der Bilder:

```
# Infoleiste unten entfernen
def cut_image(img):
    img = img[:670, 0:]
    return img

# Gauß'schen Filter anwenden
def smoothen_image(img):
    img = cv2.GaussianBlur(img, (3, 3), 0)
    return img

# Kombination aus den beiden oberen
def prepare_image(img):
    img = cut_image(img)
    img = smoothen_image(img)
    return img
```

Bilder aus einem lokalen Ordner als Numpy-Arrays importieren und (mit und ohne *Preprocessing*) in Listen speichern

```
folder_path = "images"
original_images = []
```

```
images = []

for folder in os.listdir(folder_path):
    folder_path = "images/" + folder
    for filename in os.listdir(folder_path):
        file_path = os.path.join(folder_path, filename)
        if os.path.isfile(file_path):
            img = cv2.imread(str(file_path), cv2.IMREAD_GRAYSCALE)
            original_images.append(img)
            img = prepare_image(img)
            images.append(img)
    shift_list = get_shift_list(images)
    process_images()
    original_images = []
    images = []
```

Definition der Funktionen für die *cost function* und die systematische Suche ihres Minimums

```
# Überlagerung der verschobenen Bilder
# Zuschnitt auf gemeinsame Fläche
# Berechnung der potenzierten Differenz (Wert der cost function)
def get_shifted_cost(img1, img2, shift_x, shift_y):
    len_y1, len_x1 = img1.shape
    len_y2, len_x2 = img2.shape
    if shift_x > 0:
        img1 = img1[0:, shift_x:]
        img2 = img2[0:, :(len_x2-shift_x)]
    elif shift_x < 0:
        img2 = img2[0:, -shift_x:]
        img1 = img1[0:, :(len_x1+shift_x)]
    if shift_y > 0:
        img1 = img1[shift_y:, 0:]
        img2 = img2[:(len_y2-shift_y), 0:]
    elif shift_y < 0:
        img2 = img2[-shift_y:, 0:]
        img1 = img1[:(len_y1+shift_y), 0:]
    diff = (img2-img1)**4
    cost = np.sum(diff)/np.size(diff)
    return cost
```

```
# Cost function für mehrere benachbarte Positionen
# Ausgabe des Mittelwerts
def precise_cost(img1, img2, x_pos, y_pos):
    cost_list = []
    cost_list.append(get_shifted_cost(img1, img2, x_pos, y_pos)*2)
    for x in [-1, 1]:
        cost_list.append(get_shifted_cost(img1, img2, x_pos+x, y_pos))
    for y in [-1, 1]:
        cost_list.append(get_shifted_cost(img1, img2, x_pos, y_pos+y))
    return sum(cost_list)/6

# Systematisches Absuchen eines immer kleineren quadratischen Suchfeldes
# Ausgabe: Position des lokalen Minimums
def systematic_search(img1, img2):
    favorite_pos = [0, 0]
    for i in range(0, 4):
        step_size = int(27/3**i)
        results = []
        for x in range(-2, 3):
            for y in range(-2, 3):
                x_pos = favorite_pos[0]+x*step_size
                y_pos = favorite_pos[1]+y*step_size
                cost = precise_cost(img1, img2, x_pos, y_pos)
                results.append([x_pos, y_pos, cost])
        sorted_res = sorted(results, key=lambda x: x[2])
        favorite_pos = [sorted_res[0][0], sorted_res[0][1]]
    return favorite_pos
```

Bestimmung der nötigen Verschiebung für alle Bilder in der Aufnahmeserie durch wiederholte Anwendung des Verfahrens

```
def get_shift_list(img_list):
    shift_list = []
    for i in range(len(img_list)-1):
        print("processing image number", i)
        shift_list.append(systematic_search(img_list[i], img_list[i+1]))
    return shift_list

shift_list = get_shift_list(images)
print(shift_list)
```

Zuschneiden der Bilder auf den Bereich, der allen gemeinsam ist; anschließend Berechung eines Mittelwertes

```
# Größe des Ergebnisbildes
def get_output_size(cumulative_shift_list):
    y_size, x_size = images[0].shape
    y_max, x_max = 0, 0
    for a in range(len(cumulative_shift_list)):
        for b in range(a+1, len(cumulative_shift_list)):
            if abs(cumulative_shift_list[b][0]-cumulative_shift_list[a][0]) > x_max:
                x_max = abs(cumulative_shift_list[b][0]-cumulative_shift_list[a][0])
            if abs(cumulative_shift_list[b][1]-cumulative_shift_list[a][1]) > y_max:
                y_max = abs(cumulative_shift_list[b][1]-cumulative_shift_list[a][1])
    x_size -= x_max
    y_size -= y_max
    return [x_size, y_size]


# Grenzen des Ergebnisbildes auf allen vier Seiten
def get_limits_on_all_sides(cumulative_shift_list):
    max_x, max_y = -1000, -1000
    min_x, min_y = 1000, 1000
    for item in cumulative_shift_list:
        if item[0] > max_x:
            max_x = item[0]
        if item[0] < min_x:
            min_x = item[0]
        if item[1] > max_y:
            max_y = item[1]
        if item[1] < min_y:
            min_y = item[1]
    return [max_x, min_x, max_y, min_y]


# Teil eines Rohbildes, der zum Ergebnisbild gehören wird
def get_common_image_part(img, cumulative_shift, max_x, min_x, max_y, min_y):
    y_size, x_size = img.shape
    cut_right = abs(max_x - cumulative_shift[0])
    cut_left = abs(min_x - cumulative_shift[0])
    cut_bottom = abs(max_y - cumulative_shift[1])
```

```
    cut_top = abs(min_y - cumulative_shift[1])
    img = img[0:, cut_left:(x_size-cut_right)]
    img = img[cut_top:(y_size-cut_bottom), 0:]
    return img


# Anwenden der obigen Funktionen
# Erstellen des Ergebnisbildes
def process_images():
    cumulative_shift_list = []

    for i in range(len(images)):
        total_shift = [0, 0]
        for j in range(i, len(shift_list)):
            total_shift[0] += shift_list[j][0]
            total_shift[1] += shift_list[j][1]
        cumulative_shift_list.append(total_shift)

    print(cumulative_shift_list)
    print(get_output_size(cumulative_shift_list))
    print(get_limits_on_all_sides(cumulative_shift_list))

    test_var = get_common_image_part(images[0], cumulative_shift_list[0], 0, -89, 32, -34)
    print(test_var.shape)

    max_x, min_x, max_y, min_y = get_limits_on_all_sides(cumulative_shift_list)

    final_imgs = []

    for i in range(len(images)):
        img = get_common_image_part(images[i], cumulative_shift_list[i], max_x, min_x, max_y,
        final_imgs.append(img)

    average_img = np.mean(final_imgs, axis=0)
    cv2.imwrite("images/result_"+ folder +".png", average_img)
```

Aufnahmereihe

Nr.	Vergrößerungfaktor	Bildfeld in $(\mu m)^2$
1	39660	4,4 x 2,9
2	63330	2,8 x 1,8
3	57070	3,1 x 2,0
4	62020	2,8 x 1,8
5	41780	4,2 x 2,8
6	41350	4,2 x 2,8

Tabelle 2: Detaillierte Daten zu den sechs Aufnahmereihen