HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr

Xiangyu Chen

Bestimmung ausreißerverdächtiger Strukturen in Topografien gestrahlter Oberflächen

Mittweida, 2022

Fakultät: Ingenieurwissenschaften

BACHELORARBEIT

Bestimmung ausreißerverdächtiger Strukturen in Topografien gestrahlter Oberflächen

Autor:
Herr

Xiangyu Chen

Studiengang:
Elektro- und Informationstechnik

Seminargruppe:
El16wA2-BC

Erstprüfer:
Prof. Dr.-Ing. René Pleul

Zweitprüfer:
Prof. Dr.-Ing. Marco Gerlach

Einreichung:
Mittweida, 31.10.2022

Verteidigung/Bewertung:
Mittweida, 2022

Faculty of Engineering

BACHELOR THESIS

Identification of spurious points in topography data of shot blasted surface data

author:
Mr.

Xiangyu Chen

course of studies:
Electrical and Information Technology

seminar group:
EI16wA2-BC

first examiner:
Prof. Dr.-Ing. René Pleul

second examiner:
Prof. Dr.-Ing. Marco Gerlach

submission:
Mittweida, 31.10.2022

defence/ evaluation:
Mittweida, 2022

Bibliografische Beschreibung:

Nachname, Vorname: Chen Xiangyu:

Bestimmung ausreißerverdächtiger Strukturen in Topografien gestrahlter Oberflächen

95 Seiten, Hochschule Mittweida, University of Applied Sciences,

Fakultät Elektro- und Informationstechnik, Bachelorarbeit

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

NM-Punkte	**N**icht-**G**emessene **P**unkt

1 Einleitung

1.1 Motivation

Es gibt mehrere Möglichkeiten zur Oberflächenverbesserung in der additiven Fertigung. Je nach Beschaffenheit der Teile unterscheiden sich auch die eingesetzten Nachbearbeitungstechniken. Sandstrahlen ist ein sehr wichtiger Oberflächenbearbeitungsverfahren, der die Oberflächenqualität gleichmäßiger machen kann. Dies erzeugt jedoch die typische narbige Oberflächenstruktur, die durch den lokalisierten kraterbildenden Einfluss der Strahlkörner verursacht wird.

Durch die mikroskopische Form der Oberfläche, die aus mehreren versetzt übereinander liegenden Kratern besteht, wird eine zufällige Oberfläche gebildet, die aus mehreren unterschiedlich Gestaltelementen besteht. zum Beispiel: runde abgeflachte Strukturen am Grund der Strahl-korn-Krater, steile aufragende Fronten durch Gratbildung an den Kraterrändern, steile schmale Täler aufgrund von Überlagerungen der Krater und Kraterränder usw. Ziel ist es, verschiedene Oberfläche Strukturen zu erkennen. Aber Messgeräte erzeugt Ausreißer, d.h. Spikes, die nicht auf dem eigentlichen Werkstück sind. Deshalb ist die Hauptforschung dieser Arbeit: Erkennung von Ausreißer. Weil man gefunden hat, dass bei der Messung wir erst mal die Ausreißer identifizieren müssen, bevor die Krater bewerten können.

1.2 Aufgabenstellung

Daraus ergibt sich die Aufgabenstellung, die spezifischen ausreißerverdächtige Strukturen aus Oberflächentopografien zu identifizieren, zu isolieren und deren Eigenschaften sowie deren Auftreten numerisch darzustellen. Verwendet werden dazu die Funktionen der Software Mountains Map® (Fa. Digital Surf), der Funktionsumfang von Matlab® sowie bei Bedarf eigene Algorithmen.

- Laden der Topografiedaten von *.sur – Dateien in geeignete Datenstrukturen in Matlab
- Ermittlung und Nutzung von Mountains Map Funktionen für die Extraktion der spezifischen geometrischen Merkmale über die Programmoberfläche und aus Matlab heraus. (Matlab-Mountains-Map-Schnittstelle)
  Dazu gehört der Vergleich der Rauheits-Filterfunktion in Mountainsmap mit einer Implementierung in Matlab

- Umsetzung der automatischen Berechnung dieser Merkmale für viele Topografien (durch Matlab oder Mountains-Map) und Speicherung in Tabellenform

- Erstellung eines Algorithmus zur Erkennung von besonderen Gestaltmerkmalen oder Ausreißern in den Oberflächentopografiedaten

1.3 Kapitelübersicht

In Kapitel 2 werden die Grundlagen der Sandstrahltechnik und Oberflächenmessung vorgestellt. Es enthält den Sandstrahl und Messgräte.

Kapitel 3 beschreibt die Schnittstelle Mountains-Matlab. Es enthält Beispiele für Filterung.

Die Implementierung und Design von Algorithmus des Ausreißer Entfernung wird in Kapitel 4 behandelt. Einführung des Algorithmus und die Software sind ebenfalls enthalten. Die Daten werden aus Mountainsmap exportiert und dann in Matlab importiert. Die Theorie des Algorithmus wird einige Probleme auftreten. Diese Probleme wir Schritt für Schritt in Matlab gelöst, um die Funktion zu realisieren.

Im Kapitel 5 werden Ausreißer Entfernung-Programme getestet und die Ergebnisse gezeigt.

Kapitel 6 befasst sich mit der Zusammenfassung dieser Aufgabe und dem anschließenden Ausblick.

2 Grundlagen und Stand der Technik

2.1 Sandstrahltechnik

„Sandstrahlen ist ein Verfahren der Oberflächenbehandlung und dient der Reinigung von Metalloberflächen sowie dem Mattieren von Glas. Beim Strahlprozess wird ein starker Luftstrahl erzeugt, welcher mit einem Strahlmittel vermischt wird. Der benötigte Luftdruck wird durch spezielle Kompressoren erzeugt und das pulverartige Strahlmittel kann entweder trocken oder mit Wasser aus einem Sammelbehälter beigemischt werden. Dieses spezielle Sand- Luftgemisch wird anschließend mit hoher Geschwindigkeit über ein Schlauch- und Düsensystem auf die zu behandelnde Oberfläche gestrahlt. Dabei werden Oberflächenteilchen aufgrund der abrasiven Strahlwirkung herausgelöst und anschließend entfernt.“ [1]

2.1.1 Sandstrahlverfahren

Es gibt viele Sandstrahlenverfahren, aber in dieser Bachelorarbeit wird die BERNSTEIN-Strahlanlagen verwendet, nämlich das Schleuder-Radverfahren.

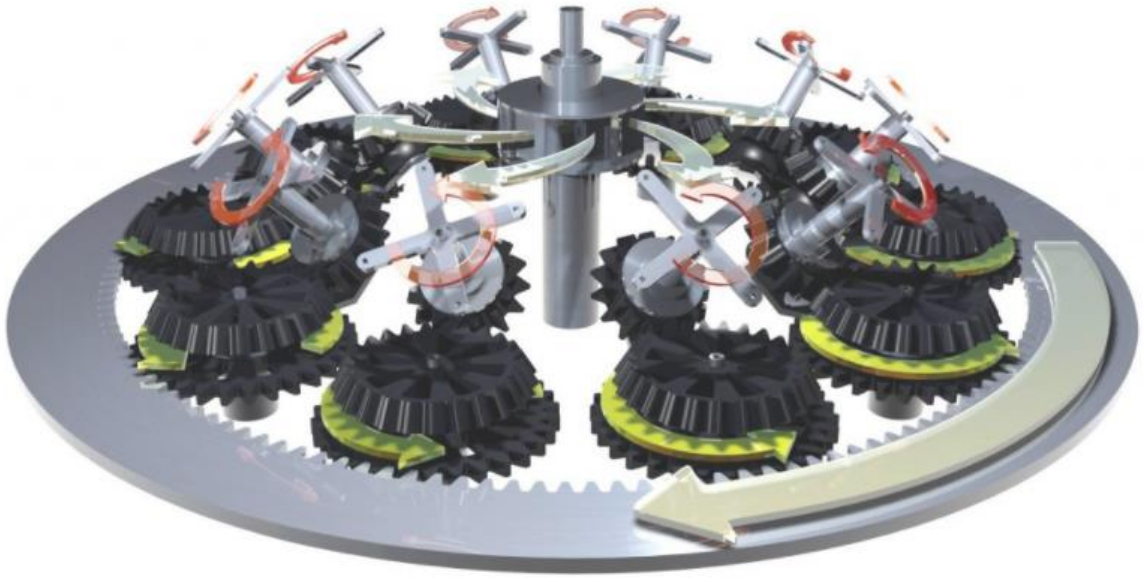


Abbildung 1: Die BERNSTEIN Strahlanlagen [2]

„Die Anlage verfügt über 10 Werkstückträger, die in der Regel mit je 1 -10 Teilen bestückt werden können. Dadurch können je Strahlzyklus bis zu 100 Teile gleichzeitig bearbeitet

werden. Das sichert eine Energieeinsparung bis zu 90% gegenüber dem konventionellen Druckstrahlen bei gesicherter gleichmäßiger Oberflächenqualität. Mit dem entwickelten Prinzip einer schonenden und verschleißarmen Strahlmittelhandhabung lässt sich das Strahlmittel wesentlich länger und effektiver als in bekannten Anlagen einsetzen.“ [2]

Das Sandstrahlen hinterlässt viele Krater auf der Oberfläche, und die unterschiedlichen Merkmale der Krater spiegeln die Eigenschaften der Oberfläche wider. Steile Flanken stellen Messtechniken und die Erkennung von Ausreißern vor Herausforderungen. (Siehe Abbildung 2)

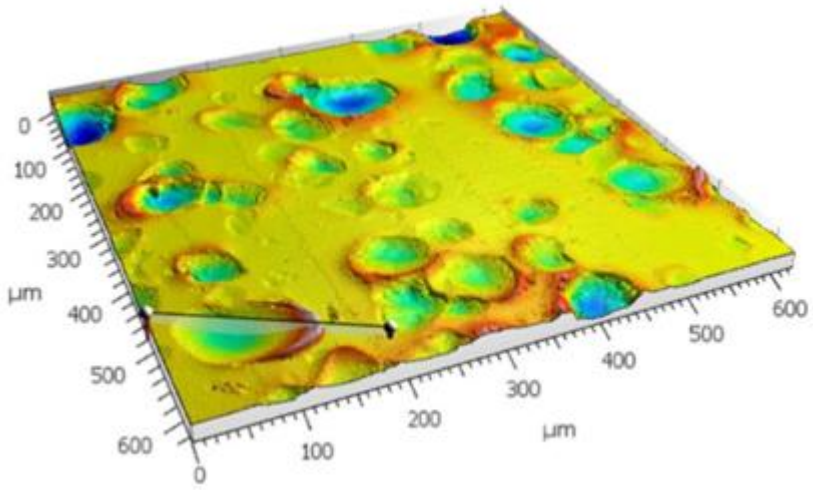


Abbildung 2: Kratern in gestrahlte Oberfläche

2.2 Grundlagen der Oberflächenmessung

Es ist unmöglich, die Oberfläche mit dem Auge zu sehen. Deshalb müssen also Messgeräte verwendet werden, um die Oberfläche zu messen. Es gibt zwei Möglichkeiten, die Oberfläche zu messen. In der industriellen Praxis ist Tastschnittverfahren ein gängiges Verfahren. Er wurde früher entwickelt und eine gute erforschte Methode. Aber da es ein berührendes Messverfahren ist, kann die Oberfläche beschädiget werden. Eine andere Methode ist die optische Messung. Dies ist ein berührungsloses Messverfahren. Es beschädigt die Oberfläche nicht und die Messung ist relativ schnell. Dieser Abschnitt beschreibt Messgeräte und Parameter sowie einige Definitionen.

2.2.1 Messgeräte

Das Tastschnittverfahren ist eine -Methode zur Messung von Oberflächen. Bei der Rauheitsmessung mit mechanischen Tastschnittgeräten nach Abbildung 3 wird eine Tastspitze, mit konstanter Geschwindigkeit über die zu messende Oberfläche geführt.

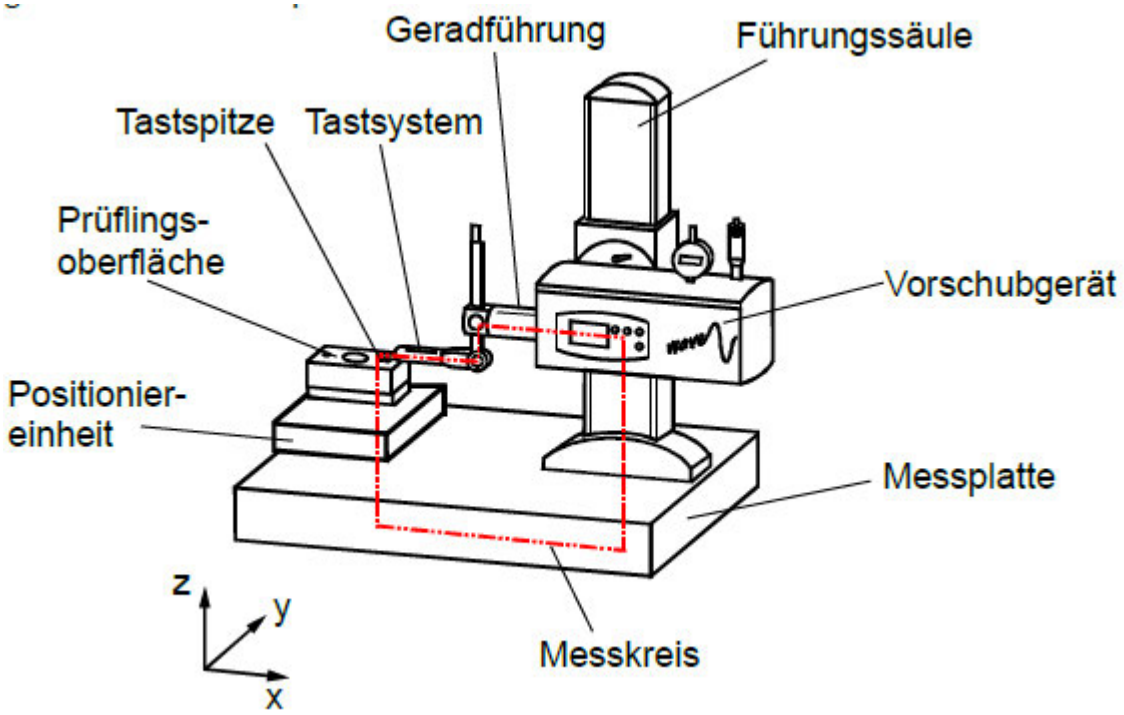


Abbildung 3: Tastschnittverfahren [3]

Das für dieses Projekt verwendete Messsystem ist TOOLinspect der confovis GmbH. Confovis bietet mit dem Messsystem TOOLinspect eine Möglichkeit zur Messung funktionstragender Oberflächen und für die Erfassung und Analyse verschiedenster Oberflächen. Das Messgerät ist in Abbildung 4 dargestellt.

Abbildung 4: Optische Instrument [4]

Die Vorteile sind wie folgt:[4]

- Hohe vertikale Auflösung bis in den einstelligen Nanometerbereich
- Hochpräzise flächige Messergebnisse in 3D
- Auf Normen rückführbare Rauheitsmessungen
- Kombiniertes Messverfahren: Konfokal-Messtechnik und Fokusvariation
- Hoher Automatisierungsgrad

Obwohl optische Instrumente viele Vorteile haben, können während der Messung Ausreißer erzeugt werden. Dies ist, was diese Arbeit untersuchen wird. Der Grund dafür wird unten erwähnt.

2.3 Rauheitskenngrößen

Wenn das Profil gefiltert wird, kann die Rauheitsprofil durch Primärprofil (P-Profil) und Welligkeitsprofil(W-Profil) berechnet werden. Anschließend können auch die Rauheitsparameter berechnet werden. Die Definitionen dieser Parameter können auch auf die beiden anderen Profile angewendet werden. Es sollte angemerkt werden, dass diese Parameter jeweils durch P, W und R dargestellt werden sollten.

Das Messgerät gibt viele Oberflächenparameter aus, die ausgewählte sind unten aufgeführt.

kenngrößen	Definition von Parametern
Ra: arithmetischer Mittenrauwert.	Ra ist der arithmetische Mittenrauwert aus den Beträgen aller Profilwerte. Die Aussagekraft von Ra ist gering, da er unempfindlich gegenüber Spitzen und Riefen reagiert.
Rz: gemittelte Rautiefe	Mittelwert der Rz-Werte aus den Einzelmessstrecken Ir. Rz ist normalerweise der Mittelwert der Ergebnisse von 5 Einzelmessstrecken.
Rq: Quadratischer Mittelwert	Rq gibt die Standardabweichung des Profils bzw. der Verteilung der (diskret vorliegenden) Ordinatenwerte des Profils an.

Rt: Gesamthöhe des Profils	Rt ist die Differenz zwischen der tiefsten Riefe und der höchsten Spitze innerhalb der Gesamtmessstrecke.
RSm: Mittlere Rillenbreite	RSm wird als arithmetischer Mittelwert der Längen der Profilelemente Xsi innerhalb einer Einzelmessstrecke berechnet. Es erfordert die Zählschwellen.
Rsk: Schiefe	Rsk charakterisiert die Asymmetrie der Amplitudendichteverteilung um die Mittellinie.
Rku: Steilheit	Rku charakterisiert die Steilheit der Amplitudendichtefunktion.

Tabelle 1: Ausgewählte Rauheitskenngrößen

kenngrößen	Formel von Parametern
Ra: arithmetischer Mittenrauwert.	$\frac{1}{n}\sum_{i=1}^{n}\lvert z_i\rvert$
Rz: gemittelte Rautiefe	$\frac{R_{z_1}+R_{z_2}+R_{z_3}+R_{z_4}+R_{z_5}}{5}$
Rq: Quadratischer Mittelwert	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}z_i^2}$
Rt: Gesamthöhe des Profils	Zpmax + Zvmax(für eine Messstrecke)
RSm: Mittlere Rillenbreite	$\frac{1}{J}\sum_{J=1}^{j}X\,s_j$

Rsk: Schiefe	$\frac{1}{nR_q^3}\sum_{i=1}^{n} z_i^3$
Rku: Steilheit	$\frac{1}{nR_q^4}\sum_{i=1}^{n} z_i^4$

Tabelle 2: Formeln von Rauheitskenngrößen

2.4 Grenzwellenlänge

Die Grenzwellenlänge λc des Profilfilters überträgt 50 % der Amplitude der Sinuswelle in das Rauheitsprofil und 50 % in das Welligkeitsprofil. Es kann als Grenze zwischen Rauheitsprofil und Welligkeitsprofil angesehen werden.

Die Grenzwellenlänge λs definiert den Übergang von der Rauheit zu kürzeren Wellenlängen, und die Grenzwellenlänge λf trennt die Welligkeit von den längeren Wellenlängen. Das heißt: λs ist der Cutoff-Wert für kurze Wellenlängen. λc und λf repräsentieren die mittlere Wellenlängengrenze bzw. die lange Wellenlängengrenze.

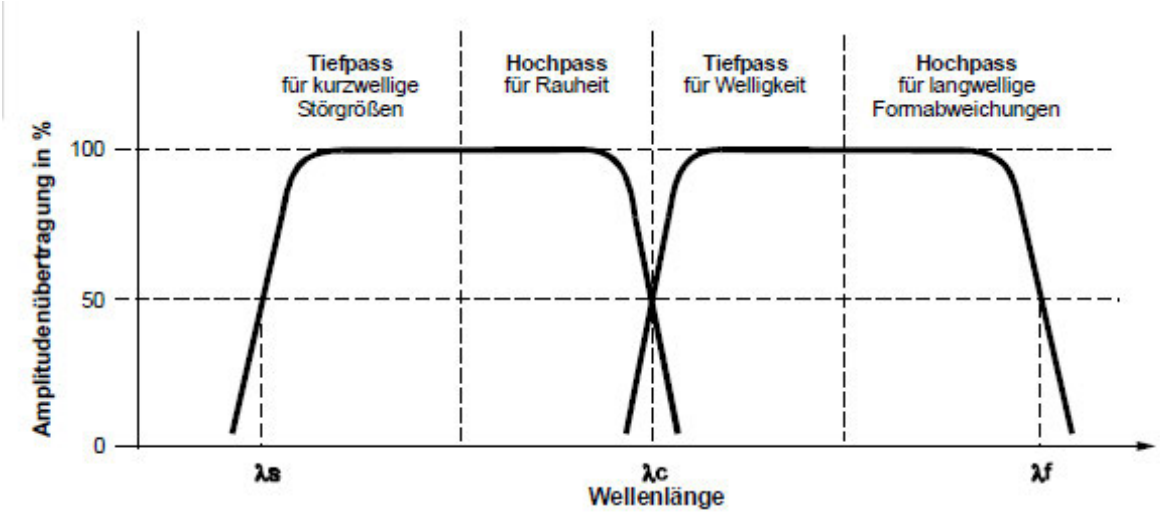


Abbildung 5: Amplitudenübertragung und Wellenlänge [3]

Die richtige Auswahl der korrekten Grenzwellenlänge λc im Algorithmus ist entscheidend, um genaue Messungen in weiteren Rauheitsparametern zu erhalten. Daher müssen wir auf die Beziehung zwischen den Parameter Ra und Grenzwellenlänge λc achten.

Tabelle 3 zeigt diese Korrelation:

Ra(µm)	lc(mm)
> (0.006) …0.02	0.08
> 0.02 …0.1	0.25
> 0.10 …2.0	0.8
> 2.00 …10.0	2.5
> 10.0…80.0	8.0

Tabelle 3: Auswahl von Grenzwellenlänge λc

Üblicherweise wird ein Filter mit λc = 0,8mm verwendet, da die meisten Rauheitswerte in dieser Klasse liegen.

Für die Auswertung des Profils und die Berechnung der Oberflächenparameter müssen bestimmte Profillängen definiert werden, die die Grundlage für die Parameterberechnung bilden.

Die Einzelmessstrecken: lr für das Rauheitsprofil. Gesamtmessstrecke (ln) (Evaluation Length) und Taststrecke(lt).

Die Einzelmessstrecke lr entspricht der Grenzwellenlänge λc des Profilfilters: λc = lr. Die Gesamtmessstrecke ln besteht aus 5 Einzelmessstrecke lr: ln = 5* λc =5*lr. Die Taststrecke lt: lt = 6* lr. Die Auswertung des Profils und die Berechnung der Oberflächenparameter erfolgen in der Gesamtmessstrecke ln.

2.5 Digitale Gauß-Filter

In diesem Artikel ist Filter ein Algorithmus. Es trennt verschiedene Wellenlängen und lässt nur die benötigten Wellenlängen sichtbar werden. Das heißt, ein Filter unterteilt ein Signal in verschiedene Bereiche.

Der Gauß-Filter garantiert dafür, dass die jeweiligen Mittellinienordinate im P-Profil anhand der Profilordinatenwerte vor und nach der jeweiligen Mittelposition berechnet werden. Zur Bestimmung der Mittellinie in der Oberflächenmesstechnik wird ein Gauß-Filter empfohlen. Digitale Gaus-Filter hat solche Vorteile: einfache Implementierung, hohe Präzision, hohe Effizienz und bessere Übertragungseigenschaften.

Ein Profilfilter (digitaler Gaußfilter) trennt das ungefilterte Primärprofil (P-Profil) in Rauheitsprofil (R-Profil) und Welligkeitsprofil (W-Profil) auf. Das Welligkeitsprofil(W-Profil) kann durch einen Gauß-Filter herausgefiltert werden.

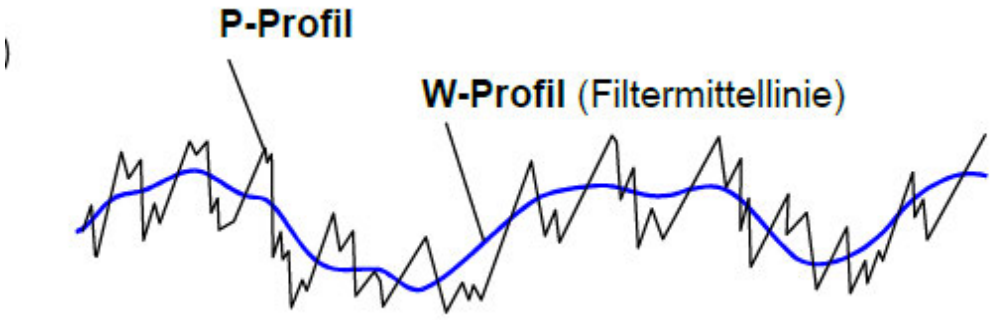


Abbildung 6: Primärprofil und Welligkeitsprofil [3]

Dann kann das Rauheitsprofil(R-Profil) durch Subtrahieren des Welligkeitsprofils(W-Profil) vom Primärprofil (P-Profil) berechnet werden.

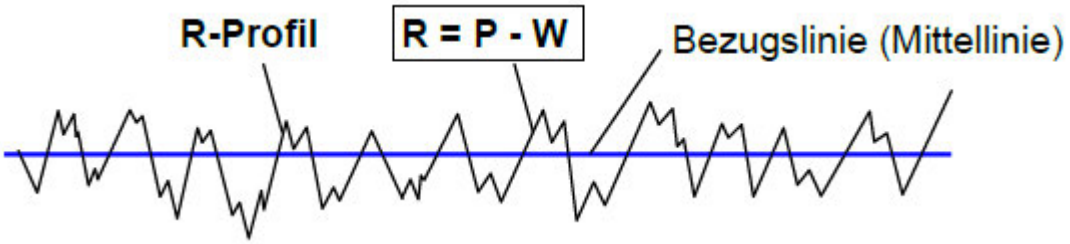


Abbildung 7: Rauheitsprofil [3]

2.5.1 Prinzip

2.5.1.1 Gewichtsfunktion

Gauß-Filter hat viele Vorteile: einfache Implementierung, hohe Präzision, hohe Effizienz und keine Phasenverzerrung. Um einen Gauß-Filter zu konstruieren, muss man zuerst die Gauß-Gewichtsfunktion erfahren.

Die Norm ISO 11562 von 1996 empfiehlt einen Gaußschen Filter zur Bestimmung der Mittellinie in der Oberflächenmesstechnik.

Die folgenden Formeln sind der ISO 11562:1996 entnommen:

$$h_{(x)} = \frac{1}{a\lambda_c} \cdot \exp\left[-\pi\left(\frac{x}{a\lambda_c}\right)^2\right] \qquad (1)$$

In dieser Formel: $\alpha = \sqrt{\ln 2 / \pi}$ ist ein konstanter Wert. λ ist die Wellenlänge und λc ist die Grenzwellenlänge des Gauß-Filters. X ist der Abstand zum Mittelpunkt der Gauß-Gewichtsfunktion. Die zur Mittelwertbildung verwendete Gewichtsfunktion hat bei den genormten digitalen Filtern (digitale phasenkorrekte Gauß-Filter) die Form einer symmetrischen Gaußschen Glockenkurve (Siehe Abbildung 8).

Abbildung 8: Glockenkurve [3]

Um einen Gaußschen Filter in Matlab zu realisieren, muss die Gaußsche Gewichtsfunktion in Matlab zu implementieren. Diese Funktion erstellt eine neue h-Variable. In diesem Algorithmus wird Faltung Methode verwendet, um die W-Profil (Filtermittellinie) zu berechnen.

2.5.1.2 Faltungsmethode

Faltung ist eine wichtige Operation, einfach ausgedrückt: Die sogenannte Faltung zweier Funktionen besteht im Wesentlichen darin, zuerst eine Funktion umzudrehen und dann eine gleitende Überlagerung durchzuführen.

In der Funktionalanalysis, einem Teilbereich der Mathematik, beschreibt die Faltung, auch Konvolution (von lateinisch convolvere „zusammenrollen“), einen mathematischen Operator, der für zwei Funktionen f und g eine dritte Funktion $f * g$ liefert. Anschaulich bedeutet die Faltung $f * g$, dass jeder Wert von f durch das mit g gewichtete Mittel der ihn umgebenden Werte ersetzt wird. Genauer wird für den Mittelwert $(f * g)(x)$ der Funktionswert $f(\tau)$ mit $g(x - \tau)$ gewichtet. Die resultierende „Überlagerung“ zwischen f und gespiegelten und verschobenen Versionen von g (man spricht auch von einer „Verschmierung“ von f) kann z. B. verwendet werden, um einen gleitenden Durchschnitt zu bilden. [5]

Bei der tatsächlichen Messung sind die Daten der gemessenen Oberfläche alle diskrete Werte. Die Formel lautet:

$$y(n) = \sum_{k=-\infty}^{\infty} f(k) \cdot g(n-k) \qquad (2)$$

In diesem Projekt wird die Methode der Faltung verwendet, um die Gaußsche Mittellinie zu erhalten. Denn durch eine gleitende Mittelwertbildung über die Profilordinaten des Primärprofils (P-Profil) kann eine Mittellinie gebildet. Diese Mittellinie ist das Welligkeitsprofil (W-Profil). Die Faltung kann jedoch auch als Verallgemeinerung des „gleitenden Mittelwertbildung“ angesehen werden, d.h. ein gleitender Mittelwertbildung ist eine Art Faltung.

Annahmen: das Primärprofil ist P(x), das Welligkeitsprofil ist W(x) und das Rauheitsprofil ist R(x).

Die Formeln zum Ermitteln von W(x) und R(x) lauten wie folgt:

$$W(x) = P(x) * h(x) \qquad (3)$$

$$R(x) = P(x) - W(x) \qquad (4)$$

2.5.2 Einführung in den Algorithmus

In diesem Abschnitt wird die Gesamtstruktur des Algorithmus erklärt, wobei der Schwerpunkt darauf liegt, wie der Algorithmus Schritt für Schritt implementiert wird. Aber erwähne nicht die Codierung.

Die Hauptfunktion des Algorithmus besteht darin, die Rohdaten (Profildatei) zu lesen und das Primärprofil, das Welligkeitsprofil und das Rauheitsprofil durch Berechnung und Verarbeitung der Daten zu erhalten. Und alle Daten werden zur visuellen Analyse geplottet.

2.5.2.1 Schritte des Algorithmus

1. Laden der gesamten Daten (Profildatei) in Matlab und Isolieren der Profildaten mit Matlab-Kommandos.
2. Grafische Darstellung der Profildaten mit z-Daten auf der Diagramm-y-Achse in µm und x-Daten auf der Diagramm x-Achse.
3. Dabei wird eine Regressionsgerade z(x)=a1*x+a0 berechnet die Profildaten von dieser Geraden subtrahiert (Berechnung der Residuen). Die Differenzen bilden das neue Profil. Das neue Profil ist das Primärprofil P(x).
4. Darstellung des neuen Profils P(x) wie in 2.
5. Durch eine gleitende Mittelwertbildung über die Profilordinaten des Primärprofils (P-Profil) wird mit Hilfe einer so genannten Gewichtsfunktion zuerst eine Mittellinie gebildet, die das Welligkeitsprofil (W-Profil) darstellt, wenn am Filter die genormte Grenzwellenlänge eingestellt wurde. Diese Mittellinie wird am Tiefpass Ausgang des Filters zur Berechnung der W-Kenngrößen bereitgestellt. Der Filter im Algorithmus liegt in der Mitte der horizontalen Länge der Probe (der Mitte der x-Daten).
6. Berechnung des Rauheitsprofils durch Subtrahieren des Welligkeitsprofils W(x) vom Primärprofil P(x): R(x)=P(x)-R(x).
7. Darstellung von Primärprofil P(x), Welligkeitsprofil W(x) und Rauheitsprofil R(x) mit der Plot-Funktion.

2.6 Der Stand der Ausreißer Forschung

In der Topografie können Ausreißer auftreten, die nachfolgende Parameterbestimmungen erheblich negativ beeinflussen können. Die meisten Messfehler können durch Ausreißer Entfernung identifiziert und korrigiert werden. Dieser Teil ist der Hauptgrund für die folgende Untersuchung.

2.6.1 Die Ursachen und Auswirkungen von Ausreißern

Wie bereits erwähnt, gibt es Ausreißer bei den Messungen. Doch was sind die Gründe für diese Ausreißer: Optische Messgeräte spielen eine immer wichtigere Rolle, was aber auch zu Ausreißern führen wird.

„Um diese optischen Geräte benutzen zu können, muss das Licht des Werkstückes auf dem Detektor reflektiert werden. Die Zeichnung oben zeigt auf der horizontalen Achse die Stärke des Signals am Detektor. Wenn das Signal gut genug ist (rechts), kann die Erfassungssoftware die Höhe korrekt und genau berechnen. Wenn kein Signal zum Detektor zurückgesendet wird (links), z.B. wenn das Messstück zu dunkel ist, oder wenn das Signal außerhalb des Objektiven reflektiert wird, dann ist es unmöglich, eine Höhe zu erhalten; das Gerät erzeugt dann NM-Punkte, oder fehlende Punkte. Am problematischsten wird es dann, wenn ein Signal erkannt wird, dieses aber nicht deutlich genug ist, um eine korrekte Berechnung zu ermöglichen (schlechtes Signal-Rausch-Verhältnis), und das Gerät daher falsche Werte liefert, die viel höher oder niedriger als die Umgebung sein können. Diese Punkte werden Ausreißer genannt.“ [6]

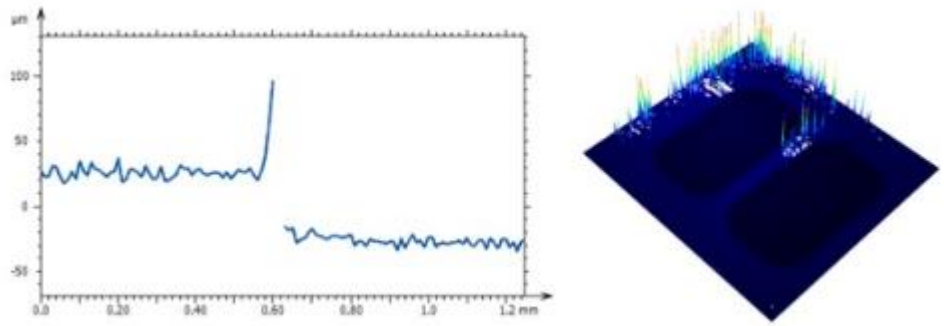


Abbildung 9: Diagramm des Signalvergleichs [6]

Das Vorhandensein von Ausreißern kann einen erheblichen negativen Einfluss auf die Parameterbestimmung haben.

2.6.2 Aktuelle Methoden zum Entfernen von Ausreißern

Gegenwärtig sind viele Methoden entwickelt worden, um Ausreißer zu entfernen. Unterschiedliche Methoden eignen sich für unterschiedliche Situationen. Hier sind einige Beispiele:

- Standardabweichung.

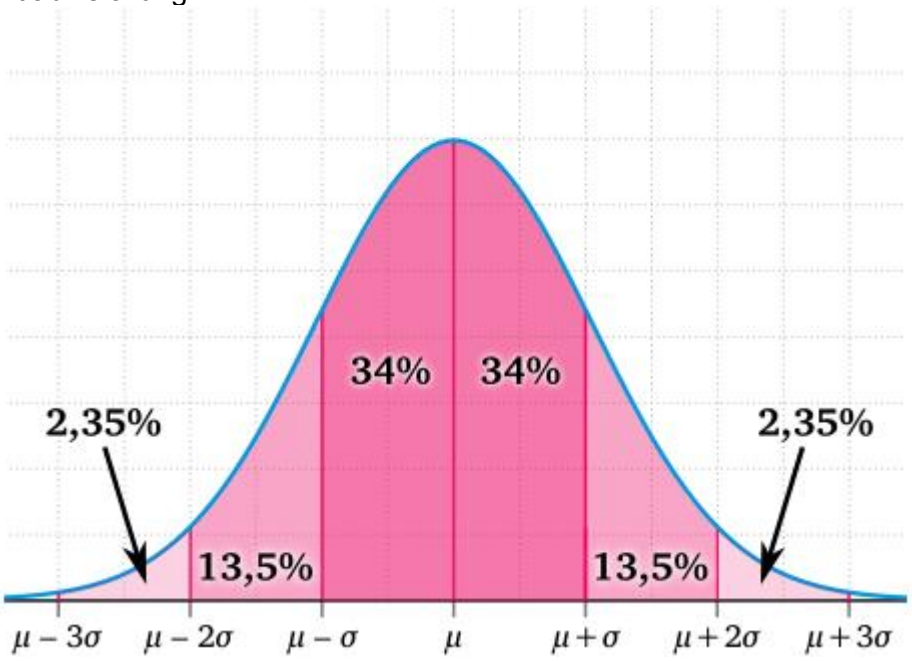


Abbildung 10: Basierend auf der Standardabweichung [7]

Wenn die Oberfläche Daten einer Normalverteilung folgen, kann die statistische Methode benutzen. Die 3-Sigma-Regel werden verwendet, um Ausreißer zu identifizieren und zu entfernen. Etwa 99,7 % der Datenwerte liegen innerhalb von drei Standardabweichungen davor und danach, und Werte außerhalb dieses Bereichs ($\pm 3\sigma$)gelten als Ausreißer.

- Box Plot

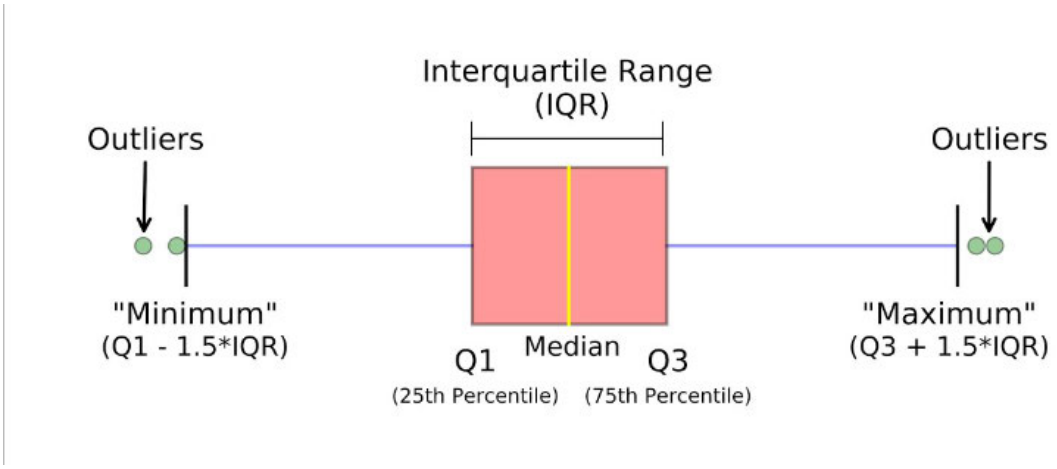


Abbildung 11: Box-Plot [8]

Der Interquartilsabstand ist wichtig, da bei schiefen Verteilungen, empfiehlt es sich, den Grenzwert für Ausreißer aus dem Interquartilsabstand zu berechnen. Es stellt die Differenz zwischen dem dritten Quartil und dem ersten Quartil (IQR=Q3-Q1) dar. Ausreißer sind in diesem Fall definiert als unter (Q1-1,5 IQR) oder über (Q3 +1,5 IQR) Beobachtungen.

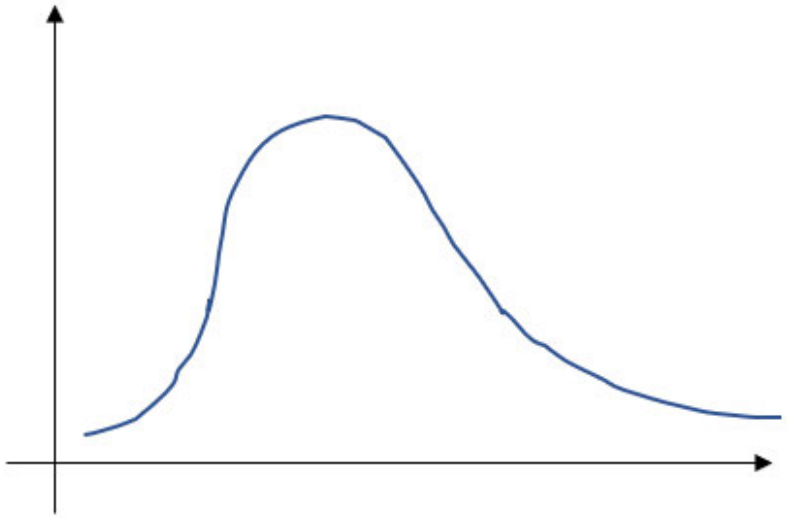


Abbildung 12: schiefe Verteilung

- Basierend auf der Steigung

Die zur Oberflächenmessung verwendeten Instrumente sind optische Instrumente. Optische Instrumente haben normalweise eine maximale Steigung in ihrer Spezifikation. Dies kann als Grenzwert verwendet werden, ob ein Punkt als Ausreißer betrachtet wird. Wenn die lokale Neigung größer als der eingegebene Winkel ist, wird ein Punkt als Ausreißer betrachtet und entfernt. Je kleiner der Wert, desto mehr Ausreißer. Die Steigung wird sehr lokal mit den umliegenden Nachbarn berechnet.

2.6.3 Funktion zum Entfernen von Ausreißern in Mountainsmap

In Mountainsmap gibt es eine Funktion zum Entfernen von Ausreißern, um Ausreißer automatisch zu erkennen, zu entfernen. (Siehe Abbildung 13) Einführung in den Algorithmus: Optische Instrumente haben oft eine maximale Steigung in ihrer Spezifikation. Dies kann als Grenzwert verwendet werden. Wenn die lokale Neigung größer als der eingegebene Winkel ist, wird ein Punkt als Ausreißer betrachtet und entfernt.

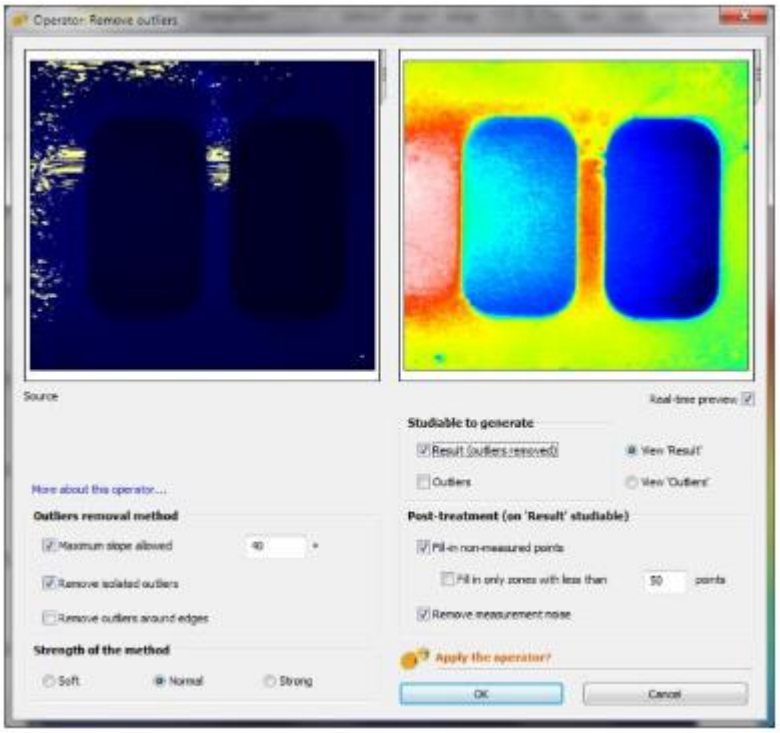


Abbildung 13: Funktion zum Entfernen von Ausreißern [6]

Aber diese Funktion ist fehlerhaft. Das heißt: Der Algorithmus in Mountainsmap war manchmal nicht zweckmäßig. (Siehe Abbildung 14)

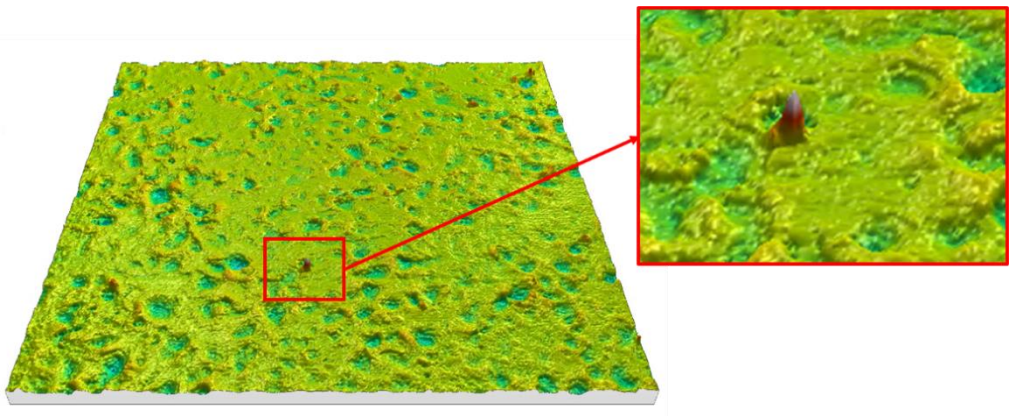


Abbildung 14: Nicht-entfernter Ausreißer auf der Topografie [9]

Die Oberfläche, die schon mit Ausreißer entfernen Funktion verarbeitet, wird auch aus der Mountainsmap exportiert und in matlab dargestellt, um den Fehler zu zeigen. (Siehe das folgende Bild (200x200-Tpografie)

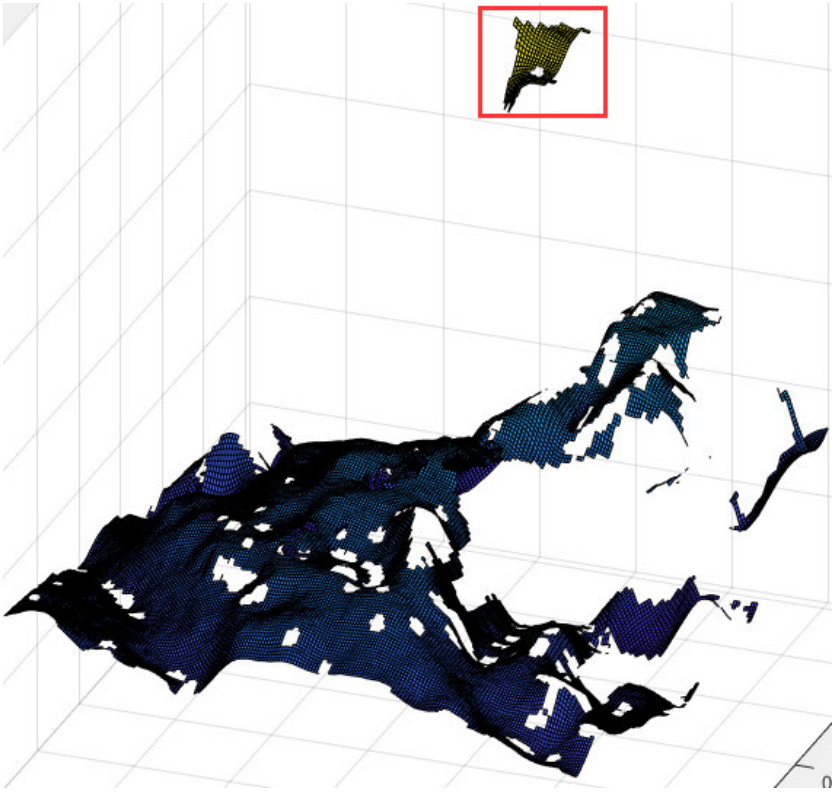


Abbildung 15: Verarbeitete Topografie von Mountainsmap

Wie oben erwähnt, verwendet der Algorithmus in der Mountainsmap-Software den Wert von Steigung, um Ausreißer zu ermitteln. Je kleiner der Wert, desto mehr Punkte sind Ausreißer. Daher werden auch einige Punkte entfernt, die zur normalen Topografie gehören. Deshalb müssen neue Algorithmen entwickelt werden, um dieses Problem zu verbessern. Das heißt, es werden mehr Ausreißer entfernt, während mehr normale Punkte beibehalten werden.

2.7 Software

Man kann die Funktionen der Software MountainsMap® benutzen, um die Topografie zu verarbeiten. In Mountains kann die Messdaten in Text-Daten gespeichert und dann in MATLAB importieren. Dabei kann eigene Algorithmen programmiert, um die Daten zu verarbeiten und analysieren. In diesem Projekt wurden MountainsMap® und MATLAB® zur Analyse der Daten verwendet.

2.7.1 MATLAB®

MATLAB ist eine Sprache für technisches Rechnen und eine interaktive Umgebung für die Entwicklung von Algorithmen, Datenvisualisierung, Datenanalyse und numerische Berechnungen. Matlab ist hauptsächlich geeignet für numerische Berechnungen mit Matrizen.

Da es in Mountainsmap eine Mountains-Matlab-Schnittstellenfunktion gibt, kann der Code in Matlab auf Mountainsmap angewendet werden. Deshalb wird Matlab in diesem Artikel verwendet, um den Algorithmus zu testen und zu simulieren, wie die Topografie vom Algorithmus verarbeitet wird. Um diesen Effekt zu erzielen, kann man die Topografie aus Mountainsmap in Form von Daten exportieren (siehe Abbildung 16).

0.000000e+00	0.000000e+00	-2.346650e+03
1.242981e-04	0.000000e+00	-2.346697e+03
2.485962e-04	0.000000e+00	-2.346742e+03
3.728943e-04	0.000000e+00	-2.346760e+03
4.971924e-04	0.000000e+00	-2.346778e+03
6.214905e-04	0.000000e+00	-2.346796e+03
7.457887e-04	0.000000e+00	-2.346814e+03
8.700868e-04	0.000000e+00	-2.346832e+03
9.943849e-04	0.000000e+00	-2.346838e+03
1.118683e-03	0.000000e+00	-2.346838e+03
1.242981e-03	0.000000e+00	-2.346839e+03
1.367279e-03	0.000000e+00	-2.346840e+03
1.491577e-03	0.000000e+00	-2.346840e+03
1.615875e-03	0.000000e+00	-2.346835e+03
1.740174e-03	0.000000e+00	-2.346822e+03
1.864472e-03	0.000000e+00	-2.346808e+03
1.988770e-03	0.000000e+00	-2.346794e+03
2.113068e-03	0.000000e+00	-2.346780e+03

Abbildung 16: Format der exportierten Beispieldaten

Dann importiere es in Matlab. Dann wird die Surface-funktion verwendet, damit kann eine 3D-topografie erstellen (siehe Abbildung 17).

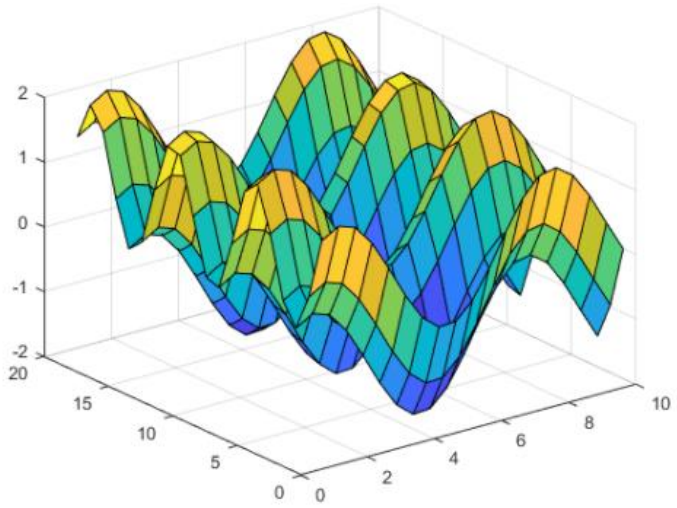


Abbildung 17 : Beispiel Topografie [10]

2.7.2 MountainsMap®

„MountainsMap ist eine Software zur Darstellung und zur messtechnischen Analyse von Oberflächen, die von der Firma Digital Surf entwickelt und vertrieben wird. Das Hauptanwendungsgebiet ist die Mikrotopografie – die Analyse von Oberflächenbeschaffenheit und Form in 3D im mikroskopischen Bereich. Die Software wird hauptsächlich in Verbindung mit taktilen oder optischen Profilometern, optischen Mikroskopen und Rastersondenmikroskopen eingesetzt.“ [11] Abbildung 18 zeigt Topografie mit Ausreißern.

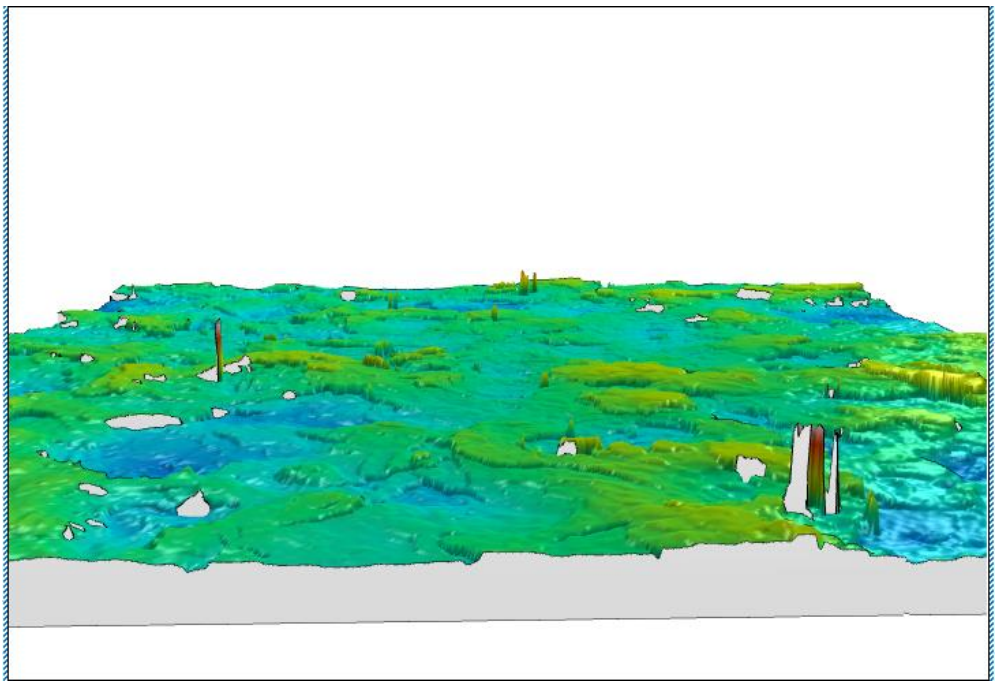


Abbildung 18: 3D-Ansicht

In diesem Dokument muss die Matlab-Funktion in Mountainsmap verwendet werden, um Matlab-Code auf Mountainsmap anzuwenden und damit Topografie zu verarbeiten (siehe Abbildung 19).

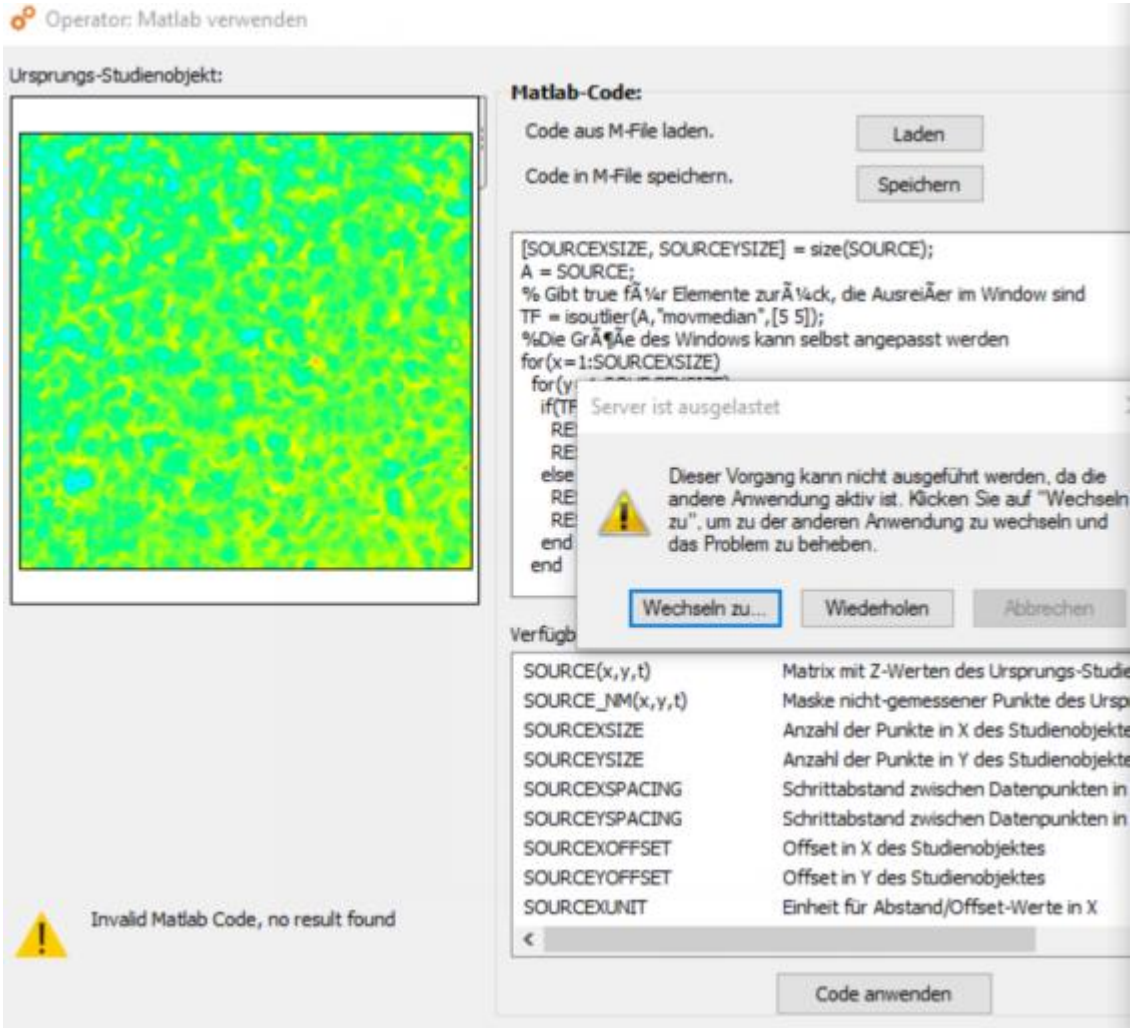


Abbildung 19: Matlab-Funktion in Mountainsmap

Außerdem kann man die Topografie in Mountainsmap auch in Form von Daten exportieren. Beim Bearbeiten von Oberflächen mit Matlab-Funktion ist es langsam und gibt Fehlermeldung. Deshalb müssen die Daten aus Mountainsmap exportiert, dann in Matlab verarbeitet und simuliert werden. Während des Exportvorgangs kann man auswählen, welcher Text exportiert werden soll, z. B. Text mit X-, Y-, Z-Werten oder Text mit nur Z-Werten. (Siehe Abbildung 20)

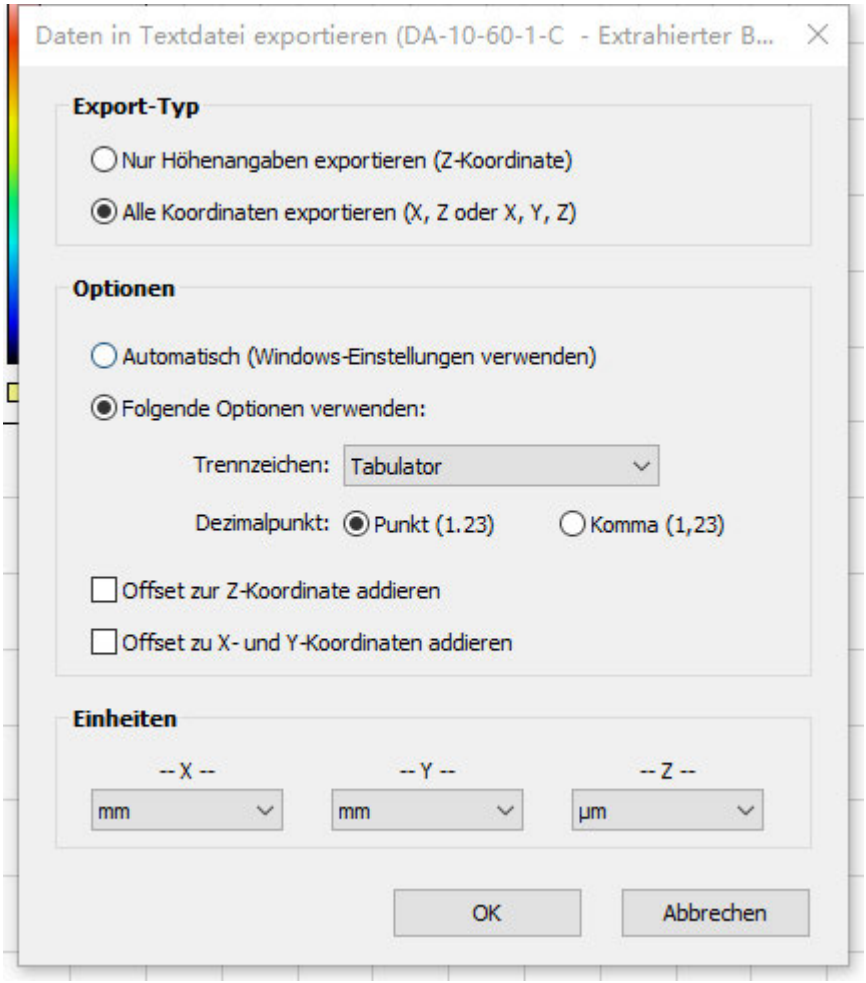


Abbildung 20: Daten exportieren

3 Untersuchung der Schnittstelle Mountains-Matlab

In diesem Projekt war das zum Codieren und Ausführen jeder Funktion oder jedes Algorithmus verwendete Programm Matlab R2021a. Alle Funktionen werden erklärt. Code spielt eine entscheidende Rolle in dem Projekt und stellt daher einen wichtigen Teil dieser Arbeit dar. Die Mountains-Matlab-Schnittstelle wird ebenfalls untersucht.

3.1 Vergleich Gaußfilter Mountains-Matlab

Zur Darstellung der Funktionsweise der Schnittstelle wird anhand der Rauheitsfilterung ein Vergleich der Ergebnisse zwischen Mountains und Matlab durchgeführt.

3.1.1 Lesen von Textdateien in Matlab

Zuerst können wir die Daten verarbeiten und dabei die benötigten z- und x-Daten beibehalten. Die erforderlichen Daten sind die erste Spalte (x-Daten) und die dritte Spalte(z-Daten).

```
31.542000000 0.000000000 0.043626000 0.000000000 0
31.543500000 0.000000000 0.043680000 0.000000000 0
31.545000000 0.000000000 0.043583000 0.000000000 0
31.546500000 0.000000000 0.043680000 0.000000000 0
31.548000000 0.000000000 0.043685000 0.000000000 0
31.549500000 0.000000000 0.043655000 0.000000000 0
31.551000000 0.000000000 0.043830000 0.000000000 0
31.552500000 0.000000000 0.043500000 0.000000000 0
31.554000000 0.000000000 0.044025000 0.000000000 0
31.555500000 0.000000000 0.043466000 0.000000000 0
31.557000000 0.000000000 0.043830000 0.000000000 0
31.558500000 0.000000000 0.043738000 0.000000000 0
31.560000000 0.000000000 0.043621000 0.000000000 0
31.561500000 0.000000000 0.043823000 0.000000000 0
31.563000000 0.000000000 0.043592000 0.000000000 0
31.564500000 0.000000000 0.043753000 0.000000000 0
31.566000000 0.000000000 0.043748000 0.000000000 0
31.567500000 0.000000000 0.043641000 0.000000000 0
31.569000000 0.000000000 0.043796000 0.000000000 0
31.570500000 0.000000000 0.043670000 0.000000000 0
31.572000000 0.000000000 0.043738000 0.000000000 0
31.573500000 0.000000000 0.043685000 0.000000000 0
31.575000000 0.000000000 0.043748000 0.000000000 0
31.576500000 0.000000000 0.043665000 0.000000000 0
```

Abbildung 21: Beispieldaten

In der obigen Abbildung werden die Zahlen in der ersten Spalte, dem x-Vektor der horizontalen Werte, in Millimetern ausgedrückt. In der zweiten Spalte wird die Höhe oder der Z-Wert in Mikrometern ausgedrückt. Die X-Koordinate startet nicht bei 0.0. Die Werte haben aber immer den gleichen Abstand. D.h. es liegt ein äquidistantes Signal vor. Für die Bildung des Signal-Vektors für die Filterung können wir die Z-Werte als Signalwerte verwenden.

3.1.2 Darstellung der Profildaten

Nachdem wir die Daten erfolgreich gelesen haben, extrahieren wir zunächst die Daten der ersten und dritten Spalte.

```
%% Grafische Darstellung der Profildaten
x = A(:,1);
y = A(:,3);
subplot(2,2,1);
plot(x,y);
ylabel('um');
```

Abbildung 22: Datenverarbeitung und grafische Darstellung

Da als nächstes andere Bilder zu zeigen sind, verwenden wir hier die subplot-Funktion, um ein 2x2-Fenster zu erstellen. Damit können mehrere Bilder gleichzeitig angezeigt werden. Schließlich ist die Achseneinheit Mikrometer auf der Abbildung markiert.

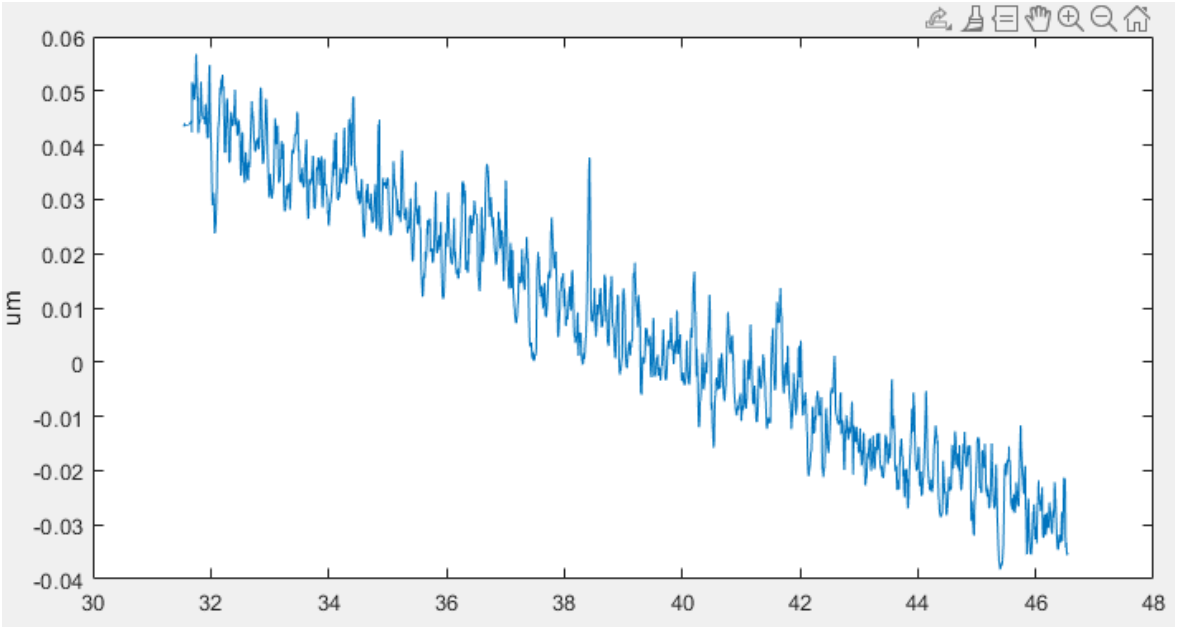


Abbildung 23: Profildaten-diagramm

3.1.3 Berechnung und Darstellung von Regressionsgerade und P(x)

Als nächstes werden wir das obige Bild horizontalisieren. Nähern Sie zunächst den Y-Wert in der Abbildung einer linearen Regression an. Dazu müssen wir die Funktion polyfit in der Matlab-Bibliothek aufrufen.

```
%% Regressionsgerade
p = polyfit(x,y,1);
f = polyval(p,x);
subplot(2,2,2);
plot(x,f);
```

Abbildung 24: Verwendung von Polyfit- und Polyval-Funktionen

Dann verwenden wir die Funktion polyval in Matlab,um den Wert von p an jedem Punkt von x (dem angepassten y-Wert) zu berechnen. Das Ergebnis ist wie unten gezeigt.

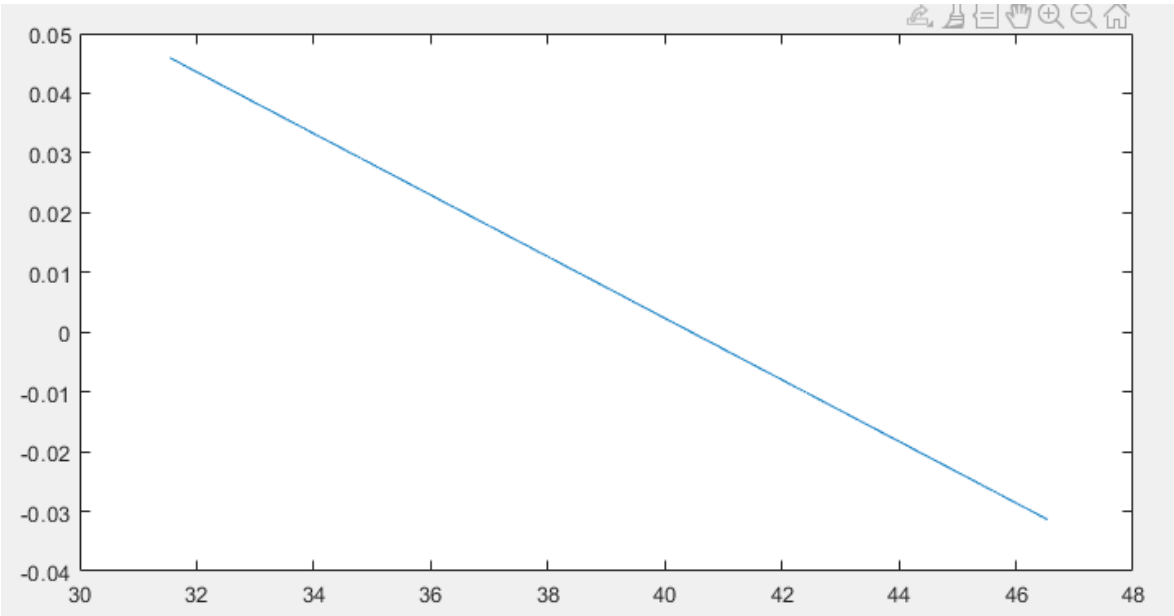


Abbildung 25: Regressionsgerade

Als nächstes subtrahieren wir die Profildaten von der Regressionsgerade und stellen die resultierenden Werte erneut graphisch dar. Das neue Profil ist das Primärprofil P(x).

```
%% Primärprofil P(x)
z = f - y;
subplot(2,2,3);
plot(x,z);
```

Abbildung 26: Zeichnung des Primärprofils

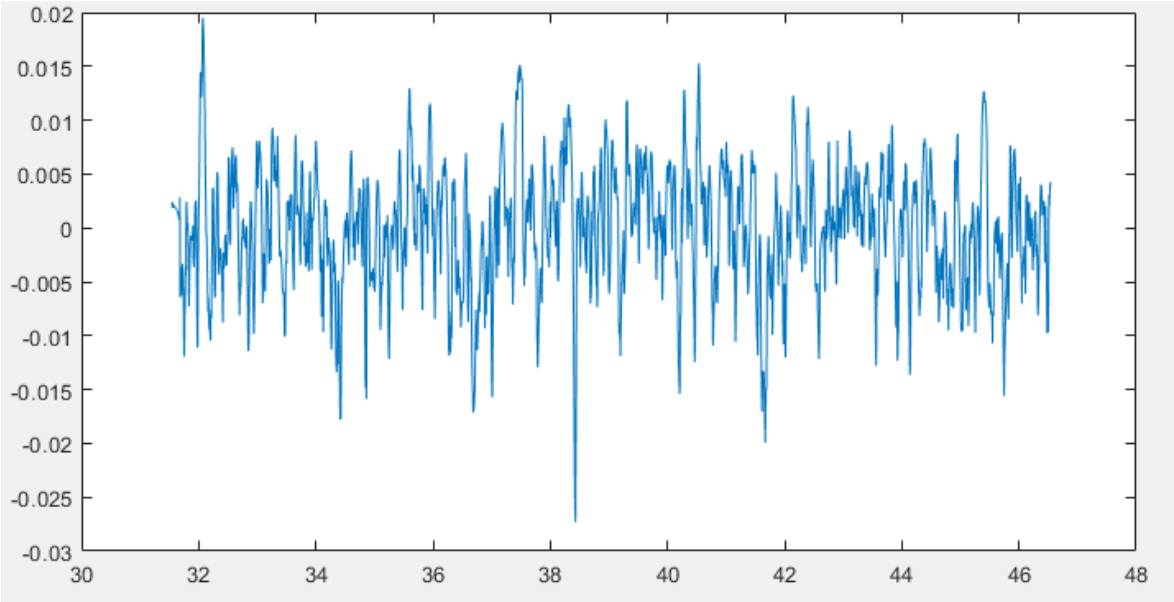


Abbildung 27: Das Primärprofil

3.1.4 Der Aufbau des Filters

Als nächstes müssen wir den Gaußschen Filter konstruieren, weil dies das Welligkeitsprofil W(x) erzeugt. Dieser Teil ist auch der Kernteil des gesamten Codes.

Hier muss die Gaußsche Gewichtsfunktion verwendet werden, wie in Formel 1 erwähnt. Es wurde auch im Design des Gaußschen Filters erwähnt: Der Filter im Algorithmus liegt in der Mitte der horizontalen Länge des Samples (der Mitte der x-Daten). Die Filtergröße ist also dieselbe wie die X-Daten. Dann berechnen wir den Mittelpunkt der Gaußschen Gewichtsfunktion und berechnen die Entfernung von jedem Punkt zum Mittelpunkt. Schließlich können wir den Wert des h-Vektors erhalten, da alle Parameter berechnet wurden.

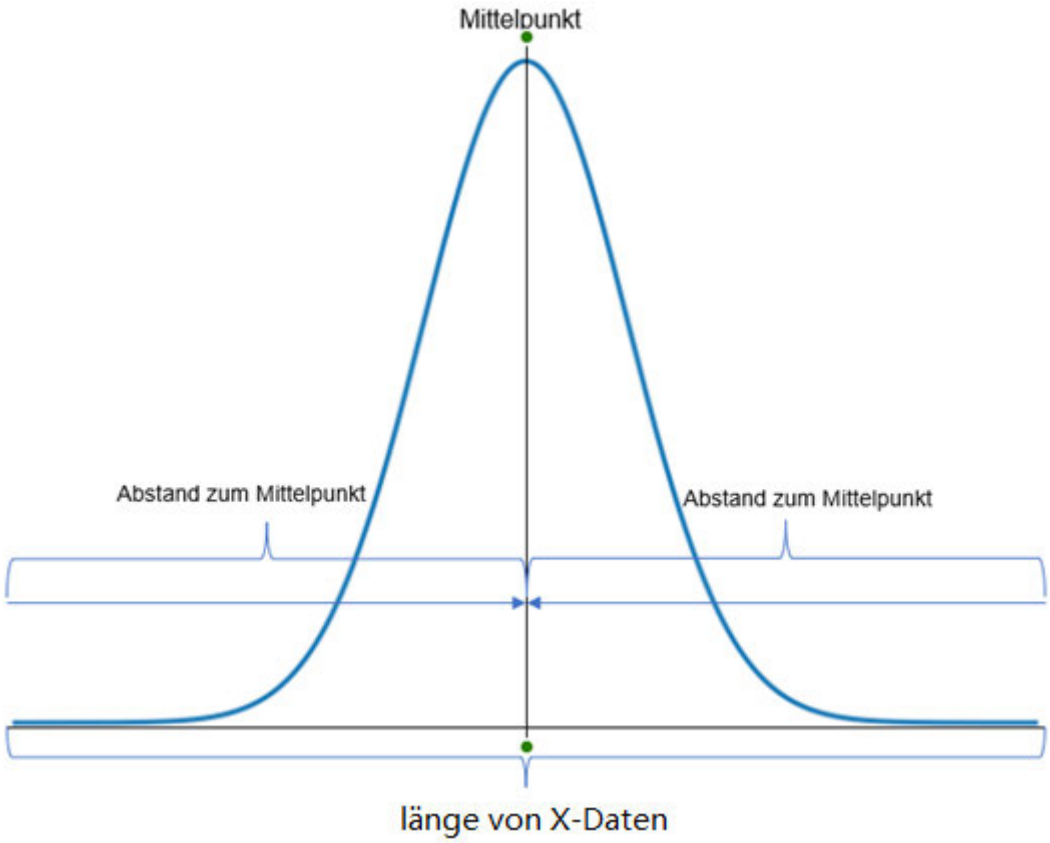


Abbildung 28: Interpretation der Gauß-Gewichtsfunktion

Der Code ist wie folgt geschrieben: Zuerst berechnen wir den Mittelpunkt der Gewichtsfunktion und bestimmen den Wert von λc. α in der Formel ist ein fester Wert, die Größe ist 0,4697. Da die Größe der Schleifenvariablen (M und h) jedes Mal iteriert wird, können wir Speicher für ein Array oder eine Matrix in Matlab vorbelegen. Auf diese Weise können Sie für große Datenstrukturen genügend Platz im Voraus reservieren, um die Geschwindigkeit zu erhöhen.

```
24    n = length(x);
25    Mittelpunkt = (x(1)+x(n))/2;
26    lc = 0.8;
27
28    M = zeros(1,n);
29    h = zeros(1,n);
30    for i = 1:n
31    M(i) = x(i) - Mittelpunkt;%in die Mitte des Filters
32    h(i) = 1/(0.4697*lc)*exp(-pi*(M(i)/(0.4697*lc))^2);
33    end
34    h = h ./ max(h);
35    plot(M,h)
```

Abbildung 29: Erstellung des Filters

Durch Schleifen kann der h-Vektor berechnet werden. Dann muss der h-Vektor normalisiert werden, wie in Zeile 34 des Codes gezeigt. h wird auf der y-Achse auf eine Skala von 0 bis 1 herunterskaliert. Die Glockenkurve sieht so aus:

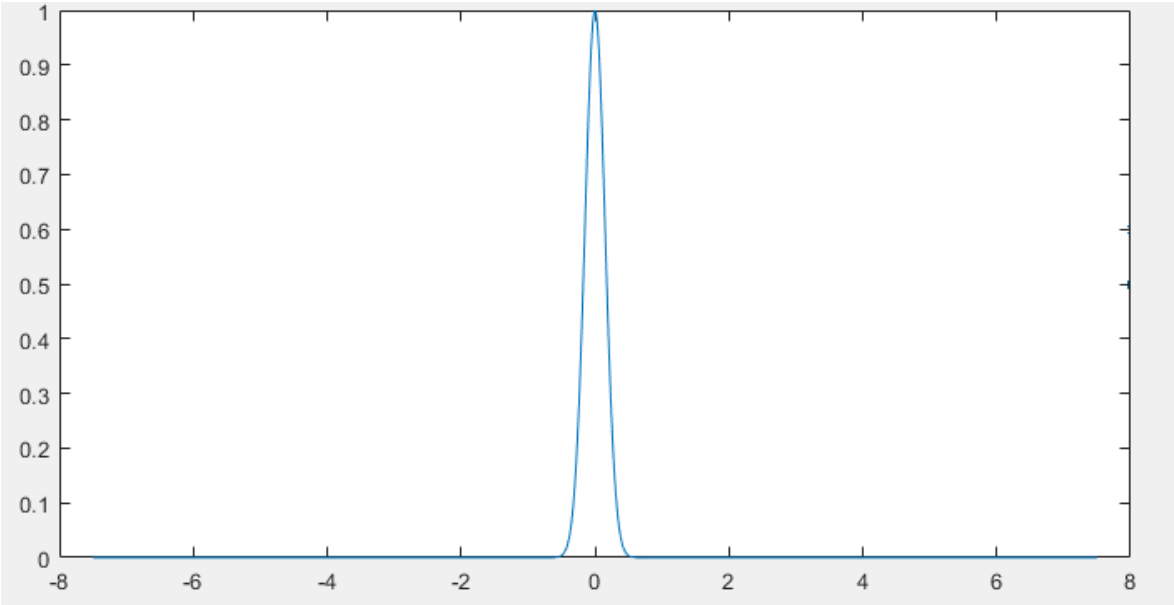


Abbildung 30: Gaußschen Glockenkurve

Wie gesagt: die Gaußsche Mittellinie ist das Welligkeitsprofil. Faltung kann verwendet werden, um Mittellinie zu berechnen. Hier wird die conv-Funktion verwendet und die Form der Funktion „same“ angegeben. Gibt nur den zentralen Teil der Faltung mit der gleichen Größe wie h zurück. Das Ergebnis nach Faltung ist W-Profil.

```
37    w = conv(h',z,"same");
38    w = w/100;
39    subplot(2,2,4);
40    plot(x,w);
41
42    r = z - w;
43    plot(x,r);
44    %% Plot mit P-Profil und W-Profil
45    plot(x,z,"-b",x,w,"-r");
```

Abbildung 31: W-Profil und R-Profil

3.1.5 Darstellung von W(x) und R(x)

Nachdem die w-Variable erhalten wurde, kann zuerst das Welligkeitsprofil angezeigt werden. Vergleichen Sie das Welligkeitsprofil mit dem Primärprofil.

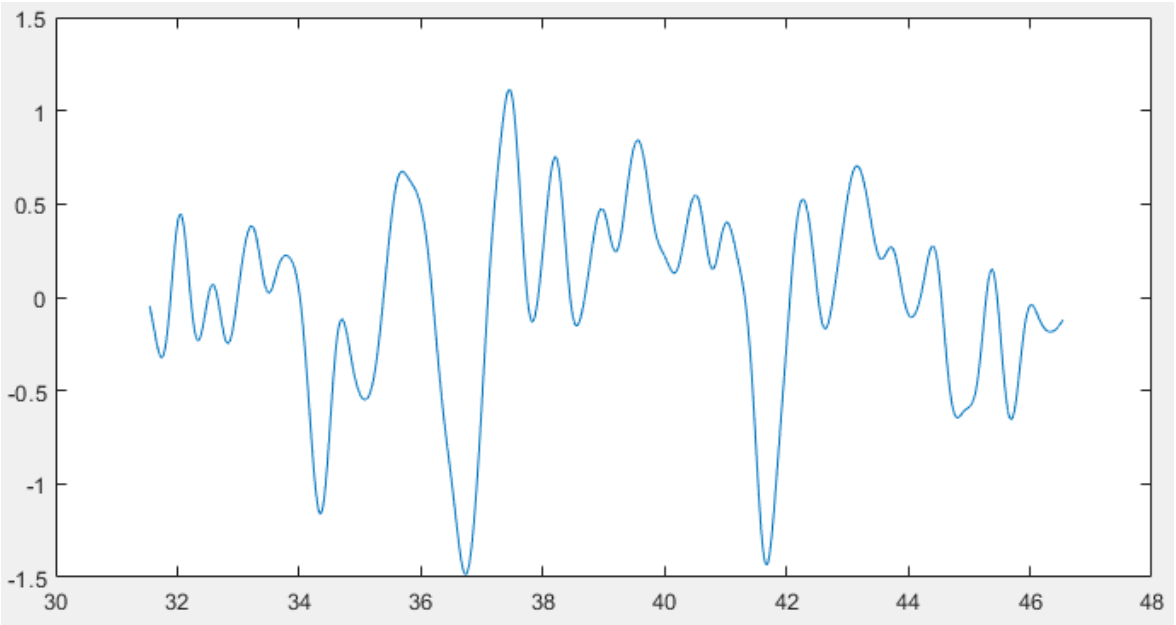


Abbildung 32: Das Welligkeitsprofil

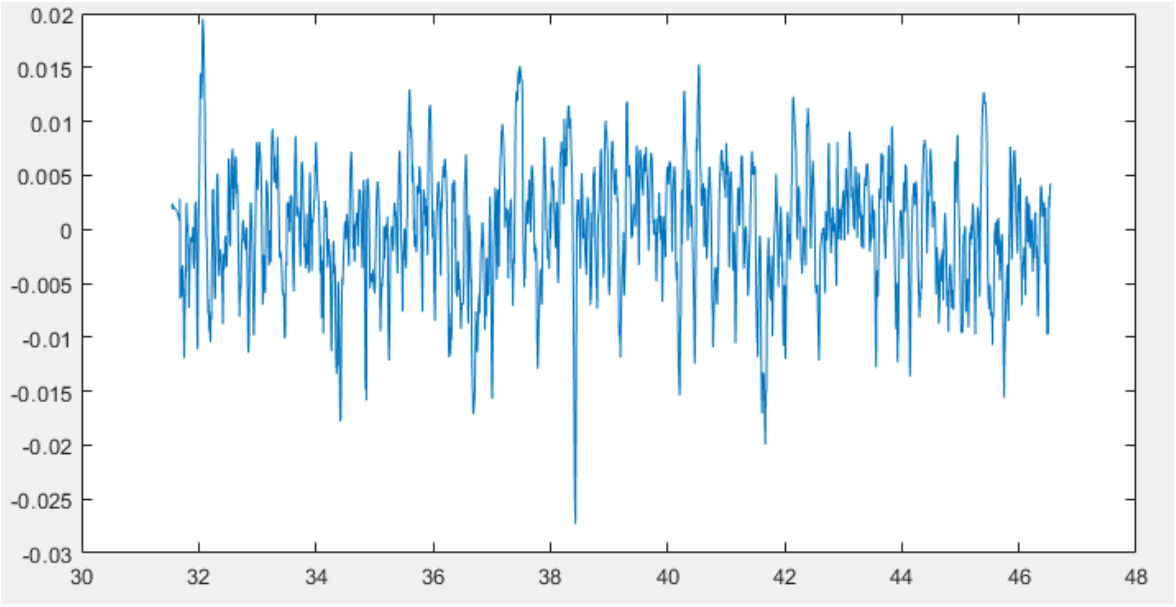


Abbildung 33: Das Primärprofil

Die Amplitude des Welligkeitsprofils wird um den Faktor 100 verstärkt, d.h. das Signal wird um den Faktor 100 verstärkt. Da der Verstärkungsfaktor 100 ist, wird die Variable w durch 100 dividiert. Das neue Welligkeitsprofil sieht wie folgt aus:

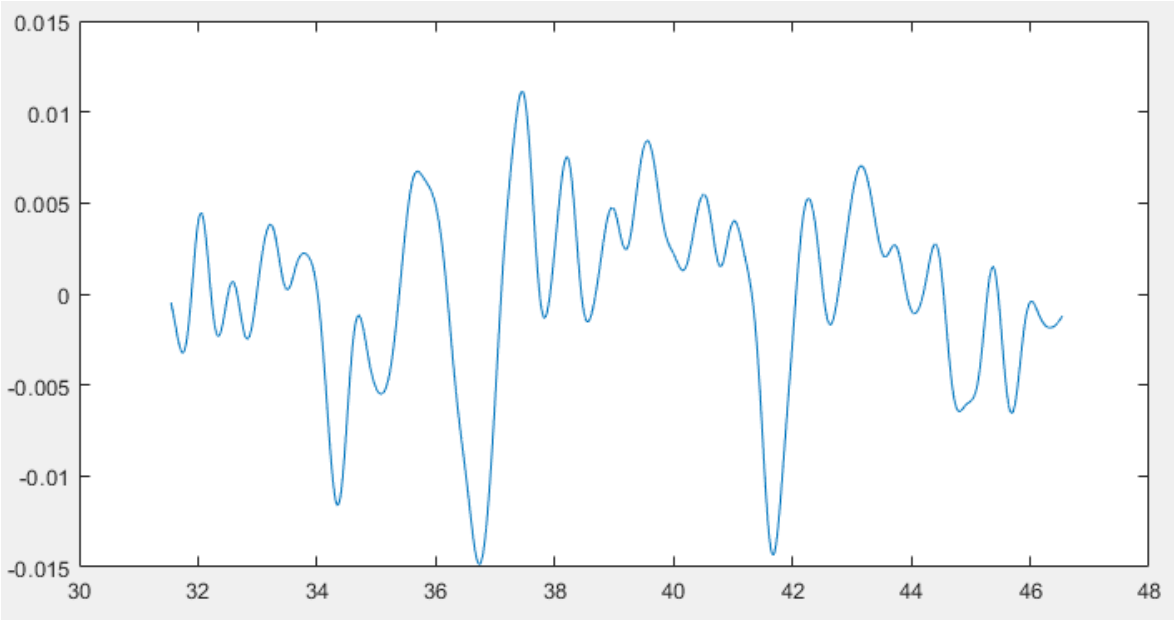


Abbildung 34: Das neue Welligkeitsprofil

Das Rauheitsprofil wird weiterhin durch Subtraktion berechnet. Darstellung von W(x) und P(x) in einem Plot mit zwei Farben und R(x) in einem extra Plot.

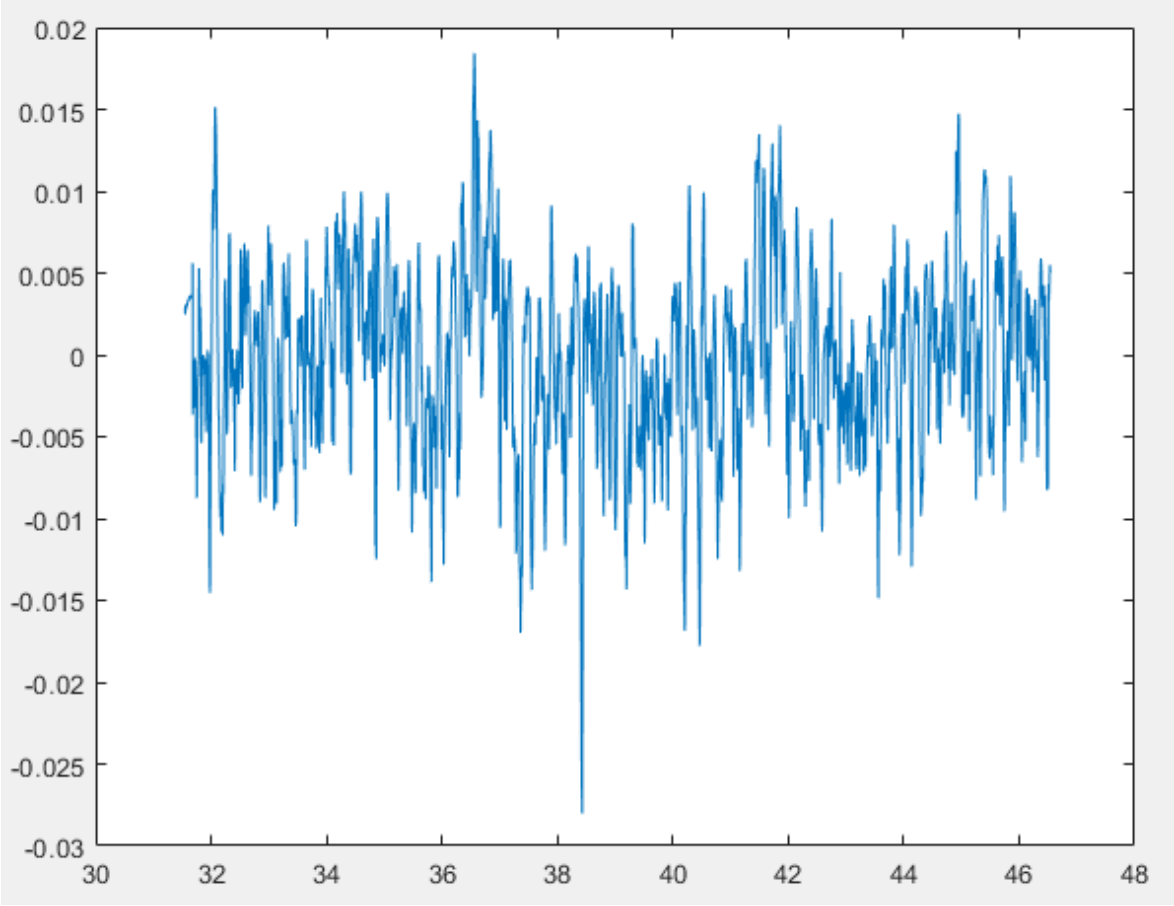


Abbildung 35: Das Rauheitsprofil

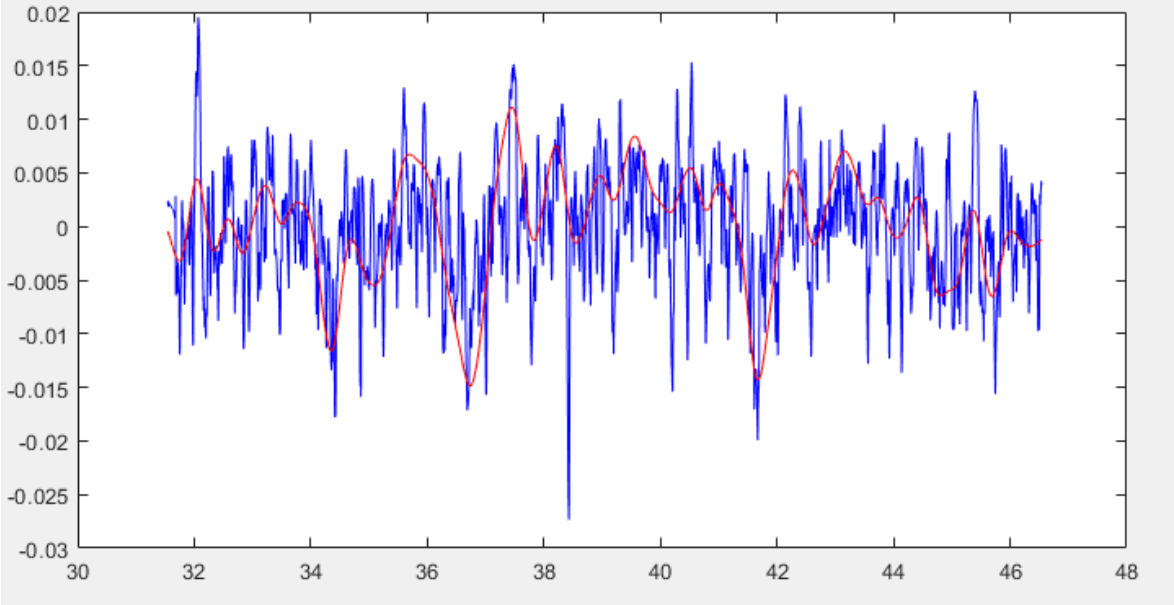


Abbildung 36: Das Welligkeitsprofil und Primärprofil

3.1.6 Test in Mountainsmap

Dieser Abschnitt stellt vor, wie Zusammenarbeit Mountainsmap und Matlab funktioniert. Zum Beispiel: Filterung. Wenn wir Matlab-Code und Softwarefunktionen in Mountainsmap vergleichen möchten, müssen wir zuerst einige Anforderungen an den Code in Mountainsmap kennen. Auf diese Weise können einige Änderungen am Matlab-Code vorgenommen werden, damit der Code in Mountainsmap richtig ausgeführt werden kann.

Als nächstes werden einige Anweisungen und Variablen in Mountainsmap vorgestellt:

- Das Ergebnis muss mithilfe die Variable RESULT angezeigt werden.
- Die von Matlab zurückgegebenen Ergebnisse werden im Fenster „Nachricht“ angezeigt.
- Wenn der Matlab-Code einen Fehler enthält, zeigt das Fenster den von Matlab zurückgegebenen Fehler an.

Variablen für das Ursprung-Studienobjekt:

SOURCE	Matrix mit Z-Werten des Ursprungs-Studienobjektes. Die Z-Werte sind absolute reelle Zahlenwerte.

	- Für ein Profil: SOURCE(x) - Für eine Oberfläche: SOURCE(x,y)
SOURCEXSIZE **SOURCEYSIZE**	Anzahl der Punkte in X/Y des Ursprungs-Studienobjektes.
SOURCEXSPACING **SOURCEYSPACING**	Schrittabstand zwischen Datenpunkten in X/Y.
SOURCEXOFFSET **SOURCEYOFFSET**	Offset in X/Y des Studienobjektes.
SOURCEXUNIT **SOURCEYUNIT** **SOURCEZUNIT**	Einheit für Abstand/Offset-Werte in X/Y. Einheit der Z-Werte.
SOURCE_NM	Maske nicht-gemessener Punkte des Studienobjektes. 1: Nicht-gemessener Punkt 0: Gemessener Punkt - Für ein Profil: SOURCE_NM(x) - Für eine Oberfläche: SOURCE_NM(x,y)

Tabelle 4: Variablen für das Ursprung-Studienobjekt [12]

Variablen für das Ergebnis-Studienobjekt:

RESULT	Matrix mit Z-Werten des Ergebnis-Studienobjektes. Die Z-Werte sind absolute reelle Zahlenwerte.
RESULTXSPACING **RESULTYSPACING**	Schrittabstand zwischen Datenpunkten in X/Y (*).
RESULTXOFFSET **RESULTYOFFSET**	Offset in X/Y des Studienobjektes (*).
RESULTXUNIT **RESULTYUNIT** **RESULTZUNIT**	Einheit für Abstand/Offset-Werte in X/Y (*). Einheit der Z-Werte.
RESULT_NM	Maske nicht-gemessener Punkte des Ergebnis-Studienobjektes. 1: Nicht-gemessener Punkt 0: Gemessener Punkt Diese Matrix muss die gleiche Größe wie das Ergebnis-

	Studienobjekt haben, sonst wird sie nicht auf das Ergebnis angewandt.

Tabelle 5: Variablen für das Ergebnis-Studienobjekt [12]

Der Code in Matlab wird mit diesen Hinweisen modifiziert, damit der neue Code in Mountainsmap ausgeführt werden kann. Mountainsmap hat eine eigene Gaußsche Filterfunktion. Ziel ist es, die Ergebnisse der beiden Filter zu vergleichen. Deshalb müssen wir also nur den Code des Gaußschen Filterteils umschreiben.

Code für Mountaninsmap:

```
[SOURCEXSIZE, SOURCEYSIZE]= size(SOURCE);
Mittelpunkt = SOURCEXSPACING*(1+SOURCEXSIZE)/2;
lc = 0.8;
M = zeros(1,SOURCEXSIZE);
h = zeros(1,SOURCEXSIZE);
for i = 1:SOURCEXSIZE
M(i) = SOURCEXSPACING*i - Mittelpunkt;
h(i) = 1/(0.4697*lc)*exp(-pi*(M(i)/(0.4697*lc))^2);
end
h = h ./ max(h);
w = conv(h',SOURCE,"same");
w = w/100;
r = SOURCE - w;
RESULT = r
```

Abbildung 37: Code in Mountainsmap

Der Code muss nicht viel geändert werden. Die einzige Änderung besteht darin, die Funktion SOURCEXSPACING (Schrittabstand zwischen Datenpunkten) zu verwenden, um X-Daten und Mittelpunkt zu konstruieren. Denn die SOURCE-Variable repräsentiert nur Matrix mit Z-Werten des Ursprungs-Studienobjekts.

Dann kann den Matlab Code in Mountainsmap eingeladen werden. (Siehe Abbildung 38)

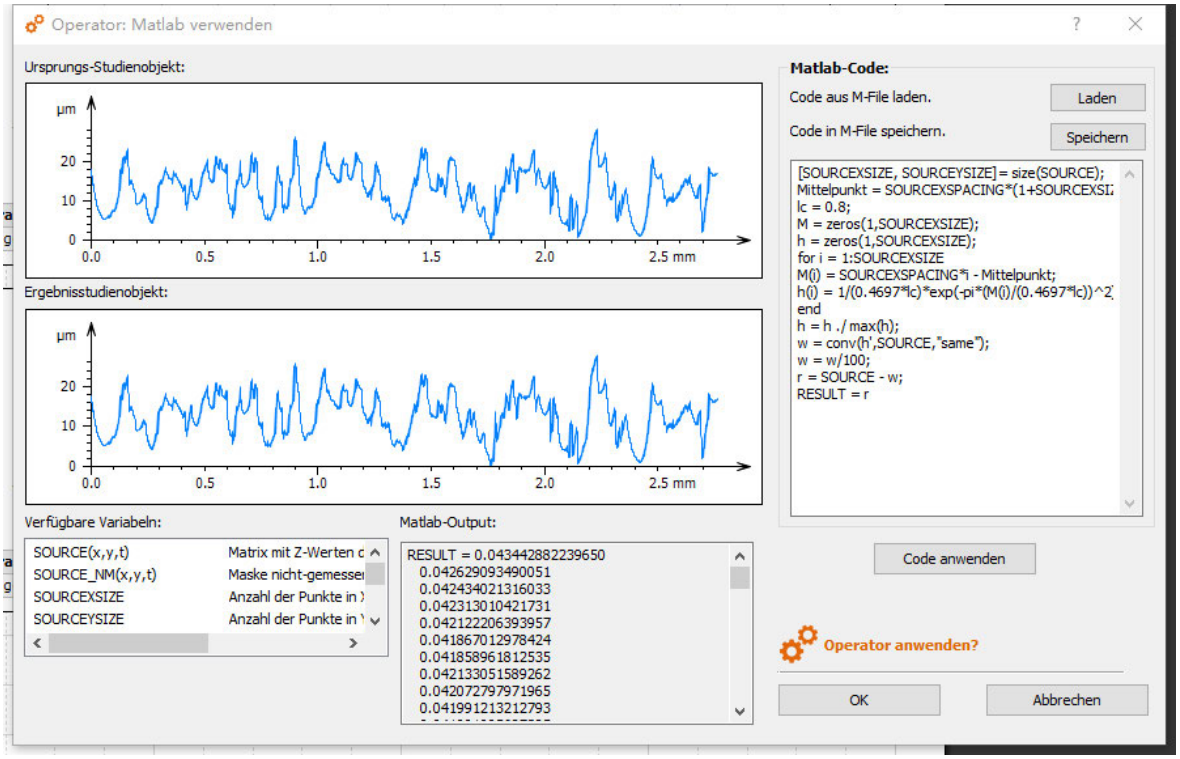


Abbildung 38: Schnittstelle Mountains-Matlab

Wie in der Abbildung gezeigt, können wir den Code aus M-File direkt in die Mountainsmap laden. Es ist auch möglich, direkt in das Fenster zu schreiben, da es einige Variablen gibt, die es in Matlab nicht gibt. Praktischerweise können Funktionen, die zum Fenster Verfügbare Variablen gehören, direkt per Doppelklick verwendet und dem Code-Writing-Fenster hinzugefügt werden.

Wenn der Code geschrieben ist, klicken Sie auf Code anwenden, um zu überprüfen, ob der Code falsch ist. Wenn der Code korrekt ist, zeigt das Ergebnisstudienobjekt-Fenster das verarbeitete Bild an. Das Matlab-Output Fenster zeigt den Wert der RESULT-Variablen an. Klicken Sie dann auf OK, um alle Vorgänge abzuschließen. Das Endergebnis ist in der Abbildung 39 dargestellt.

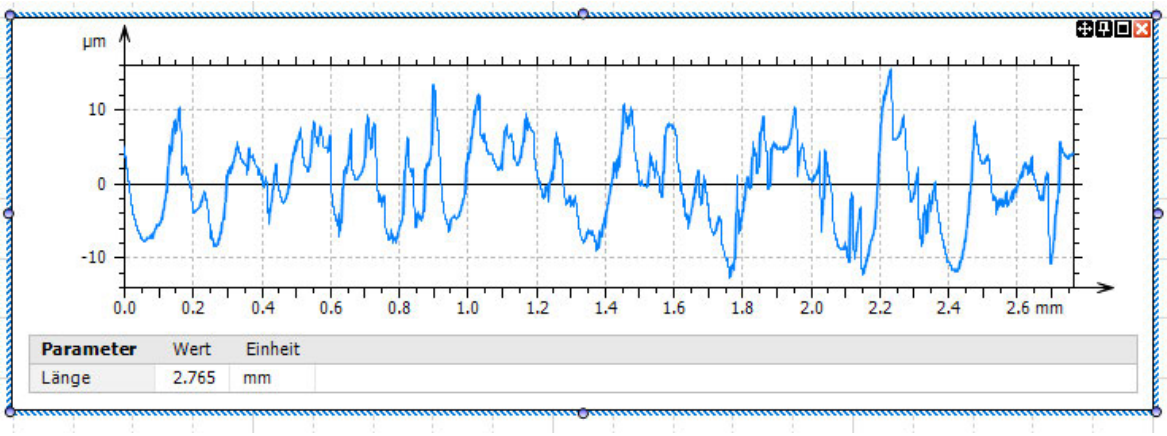


Abbildung 39: Rauheitsprofil von selbst Code

Als nächstes verwenden wir die Funktionen in Mountainsmap, um das Rauheitsprofil zu erhalten. (Siehe Abbildung 40)

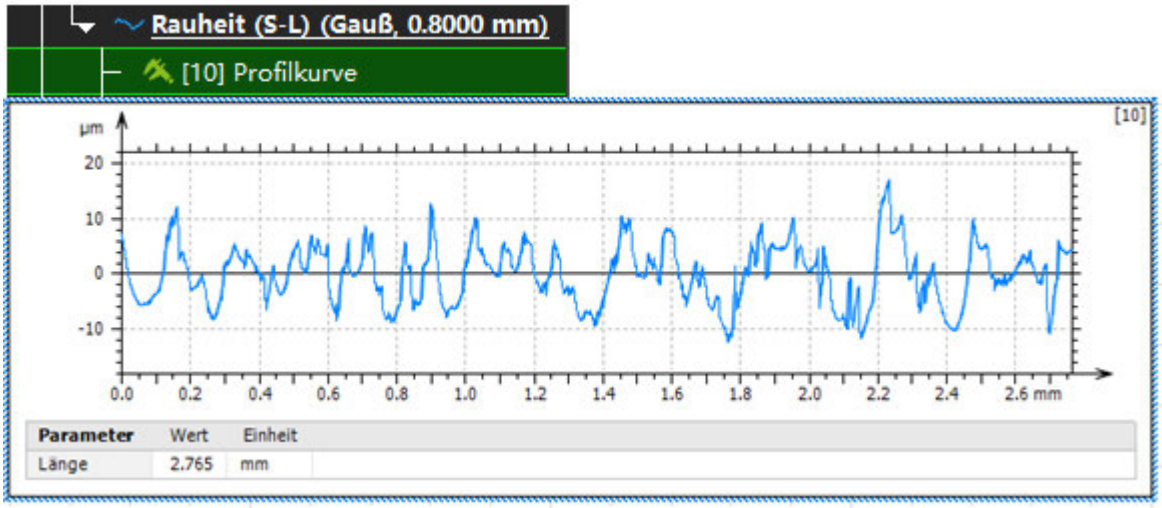


Abbildung 40: Rauheitsprofil von Mountainsmap

Schließlich können wir die Funktion Parametertabelle in Mountainsmap (siehe Abbildung 41 und 42) verwenden, um die erforderlichen Parameter auszuwählen und zu berechnen.

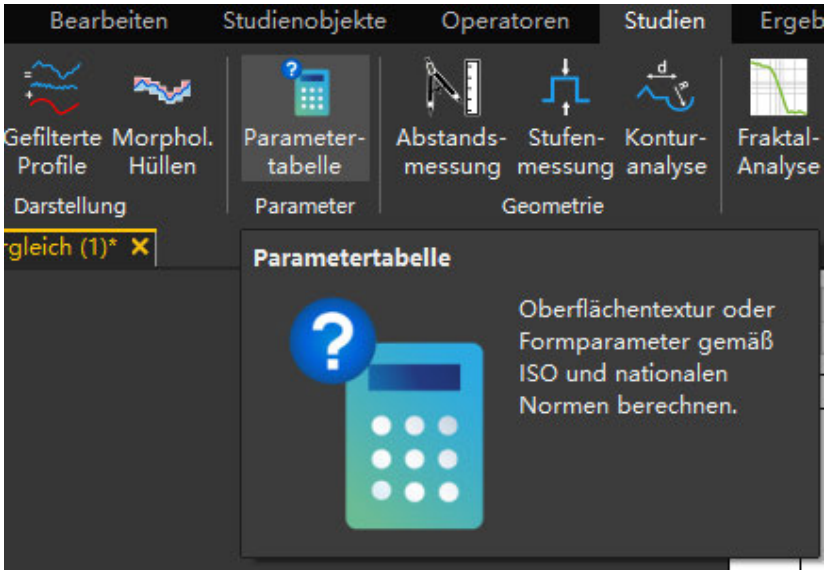


Abbildung 41: Parametertabelle-Funktion

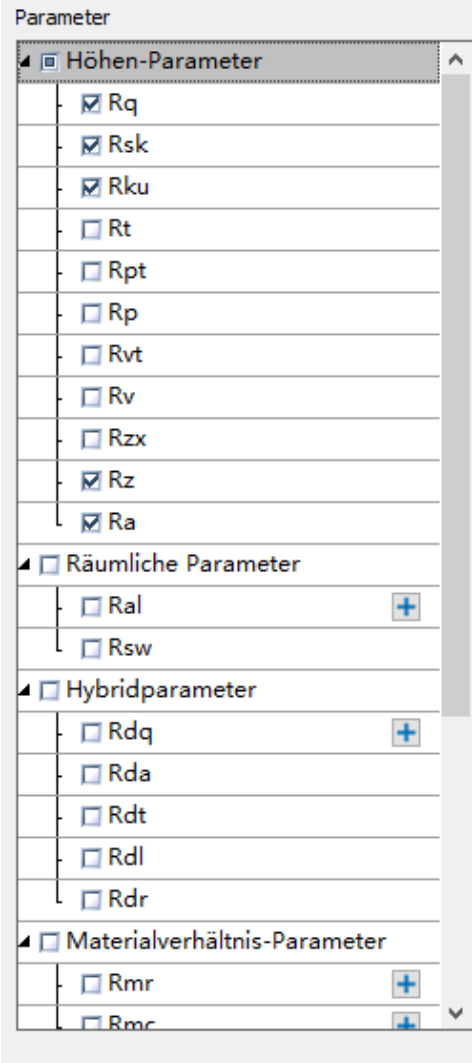


Abbildung 42: Auswahl der Parameter

ISO 21920 - Rauheit (S-L)			
F: Keine			
S-Filter (λs): Gauß, 2.5 µm			
L-Filter (λc): Gauß, 0.8 mm			
Messstrecke: Alle λc (3)			
Höhen-Parameter			
Rq	5.074	µm	
Rsk	0.1674		
Rku	2.609		
Rz	22.73	µm	*Durchschnittswert auf: Alle λc (3)*
Ra	4.148	µm	

Abbildung 43: Ergebnisse des eigenen Algorithmus

ISO 21920 - Rauheit (S-L)			
F: Keine			
S-Filter (λs): Gauß, 2.5 µm			
L-Filter (λc): Gauß, 0.8 mm			
Messstrecke: Alle λc (3)			
Höhen-Parameter			
Rq	5.289	µm	
Rsk	0.1673		
Rku	2.610		
Rz	23.68	µm	*Durchschnittswert auf: Alle λc (3)*
Ra	4.323	µm	

Abbildung 44: Ergebnisse von Mountainsmap

Die Abbildungen 43 und 44 zeigen jeweils die Ergebnisse von Mountainsmap und unserem eigenen Algorithmus. Es ist ersichtlich, dass die Parameter leicht unterschiedlich sind.

4 Design und Implementierung von Algorithmus des Ausreißer Entfernung

Das zweite Kapitel stellt die Ursachen und Auswirkungen von Ausreißern vor. In diesem Kapitel werden Algorithmen zum Entfernen von Ausreißern entworfen. Dieses Programm ist in 3 Teile gegliedert, die im Folgenden nacheinander vorgestellt werden.

4.1 Algorithmus-Design

Ausreißer sind Datenpunkte im Datensatz, die weit von anderen Beobachtungen entfernt sind. Es kann auch gesagt werden, dass es außerhalb der Gesamtverteilung des Datensatzes liegt. Um Ausreißer zu entfernen, können wir selbst ein Programm schreiben. Der Algorithmus dieses Programms wird hier zuerst vorgestellt. Aber erwähne nicht die Codierung.

Die Hauptfunktion dieses Algorithmus ist: Die Werte in der gesamten Matrix werden automatisch gescannt und die problematische Werte werden als Ausreißer markiert und eliminiert. Alle Daten werden zur visuellen Analyse in 3D-Bilder umgewandelt.

Zuerst erstellen wir einen Beispielgraphen der Oberfläche (Siehe Abbildung 45):

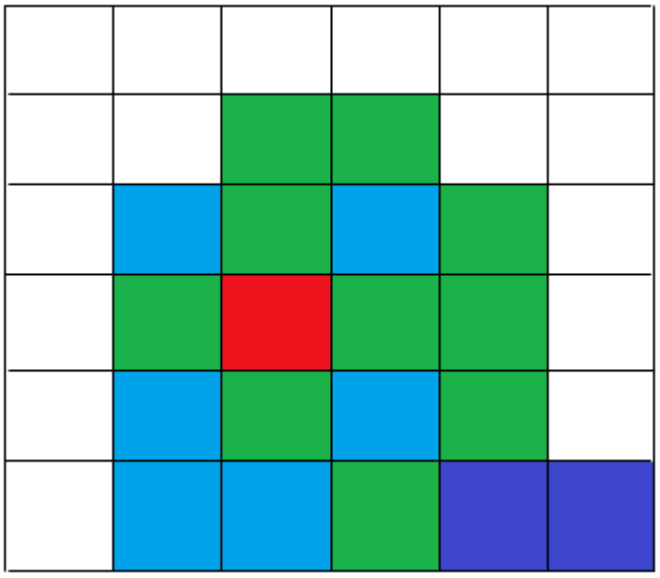


Abbildung 45: Beispieldiagramm 1

Auf dem Bild sehen wir einige Bereiche mit unterschiedlichen Farben. Die roten Bereiche sind die höheren Punkte, die oft als Peaks bezeichnet werden, und die grünen, blauen und dunkelblauen Bereiche sind die niedrigeren Punkte. Der Zweck besteht darin, diese höheren Punkte zu entfernen. Die Höhen dieser Punkte werden durch ihre entsprechenden spezifischen Werte (Z-Werte) bestimmt. Nachdem dieser rote Punkt positioniert ist, können auch die umliegenden Punkte gefunden werden. Wenn man die Z-Werte dieser Punkte kennt, kann die Differenz zwischen dem roten Punkt und seinen umliegenden Punkten berechnet werden. Es wird Δz genannt (Siehe Abbildung 46). Die Methode basiert darauf, wie groß ist DeltaZ im Unterschied zu einem Peak. Dies kann erkennen, ob die Umgebung dieses Punktes steil ist. Wenn die Umgebung steil ist, dann ist dieser Punkt wahrscheinlich ein Ausreißer.

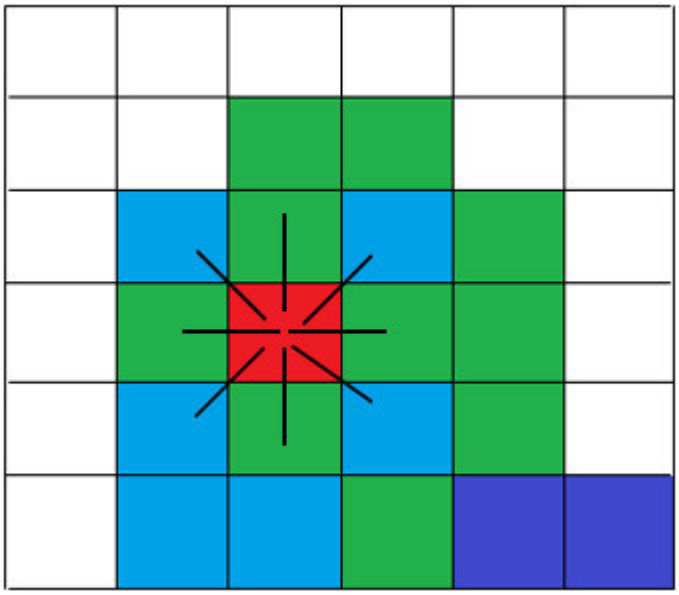


Abbildung 46: Beispieldiagramm 2

Sobald man diese Δz -Werte bekommt, kann ein Grenzwert (z.B. 10 Mikrometer) eingestellt werden. Wenn einer dieser Δz -Werte größer als dieser Grenzwert ist ($\Delta z >$ Grenzwert), wird der Punkt als Ausreißer betrachtet und entfernt.

Es gibt einen häufigen Fall (Siehe Abbildung 47): Es gibt mehrere Peaks. In diesem Fall müssen alle Ausreißer entfernt werden.

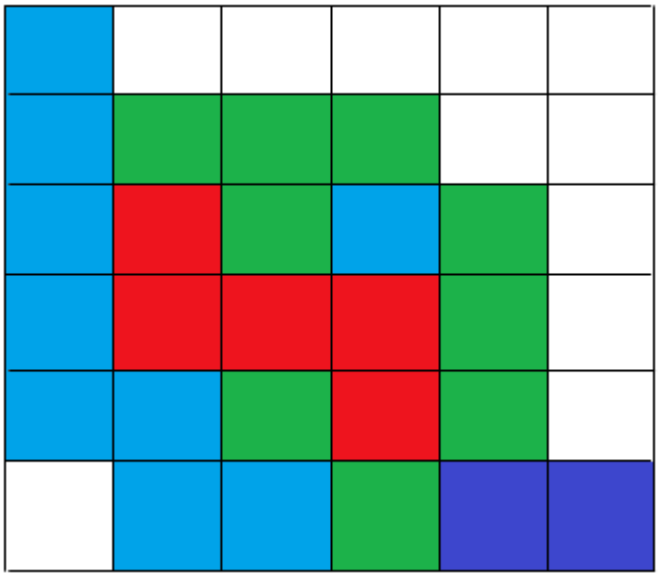


Abbildung 47: Beispieldiagramm 3

Um dieses Problem zu lösen, kann ein Schleifenprogramm geschrieben werden. So kann jeder Z-Wert identifiziert und mögliche Werte als Ausreißer klassifiziert und entfernt werden.

4.1.1 Bestimmung von Grenzwerten

Das Verfahren zum Entfernen von Ausreißern ist wie oben beschrieben. Ein wichtiger Parameter wird erwähnt: der Grenzwert. Durch den Grenzwert können wir feststellen, ob ein Wert die Bedingungen eines Ausreißers erfüllt. Als nächstes wird die Methode zur Berechnung des Grenzwerts vorgestellt.

Denn in der Topografie gibt es zwei Bereiche: Ausreißer Bereich und Normale Topografie. (Wie in Abbildung 48 gezeigt) Zur Bestimmung des Grenzwertes kann ein talbasierter Minimalwert verwendet werden, damit die Trennlinie gefunden werden kann. Dann kann der Abstand zwischen den beiden Bereichen berechnet werden. Diese Methode eignet sich für Histogramme mit bimodalen Verteilungen.

Ähnlich wie bei der Methode nach dem Talminimum kann auch der Mittelwert der Doppelpeaks als Trennlinie genommen werden. Das heißt, der letzte Schritt besteht nicht darin, den Tal wert zwischen den Doppelspitzen zu erhalten, sondern den Mittelwert der Doppelspitzen als Trennlinie zu nehmen.

Da die Codeausführungsumgebung Matlab ist, können viele Parameter direkt durch Befehle in Matlab berechnet werden. Dann folgt die mathematische Beschreibung (Siehe Abbildung 48):

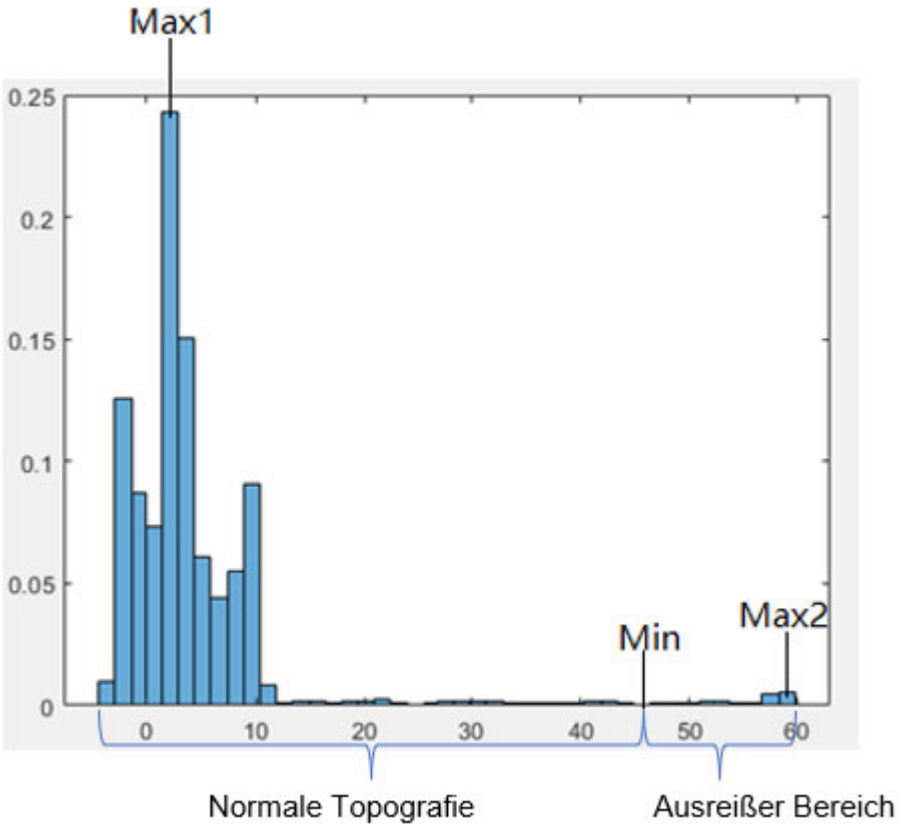


Abbildung 48: Beispiel Histogramm

Finden Sie zuerst die kleinste Häufigkeit(min) als Trennlinie. Es werden zwei Bereiche aufgeteilt (Normale Topografie und Ausreißer Bereich). Dann finden Sie für beide Bereiche die jeweilige maximale Häufigkeit (Max1 und Max2) und den Mittelwert der Z-Werte beider Bereiche (Mittelwert1 und Mittelwert2).

$$\text{Abstand} = \bar{x}_2 - \bar{x}_1 \qquad (5)$$

Die zweite Methode muss nur die Trennlinie neu berechnen, andere Schritte bleiben unverändert:

$$\text{Trennlinie} = (Max_1 + Max_2) \,/\, 2 \qquad (6)$$

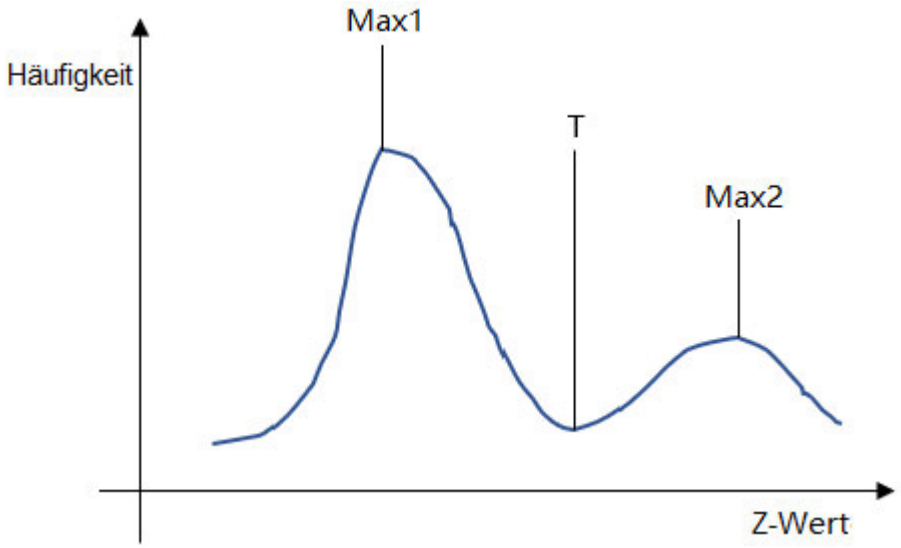


Abbildung 49: bimodale Verteilung

4.2 Implementierung des Codes zum Entfernen von Ausreißern

Der Inhalt dieses Teils ist: Implementieren Sie die oben genannte Methode in Form eines Algorithmus in Matlab, um zu beurteilen, ob jeder Z-Wert den Standard von Ausreißern erfüllt. Der in Matlab importierte Z-Wert lautet wie folgt (siehe Abbildung 50).

1	-2.346650e+03	-2.346697e+03	-2.346742e+03	-2.346760e+03	-2.346778e+03	-2.346796e+03	-2.346814e+03	-2.346832e+03	-2.346838e+03	-2.346838e+03
2	-2.346722e+03	-2.346764e+03	-2.346804e+03	-2.346826e+03	-2.346848e+03	-2.346870e+03	-2.346892e+03	-2.346914e+03	-2.346913e+03	-2.346903e+03
3	-2.346794e+03	-2.346830e+03	-2.346866e+03	-2.346892e+03	-2.346918e+03	-2.346944e+03	-2.346970e+03	-2.346996e+03	-2.346989e+03	-2.346968e+03
4	-2.346867e+03	-2.346898e+03	-2.346929e+03	-2.346958e+03	-2.346988e+03	-2.347018e+03	-2.347048e+03	-2.347078e+03	-2.347064e+03	-2.347033e+03
5	-2.346942e+03	-2.346971e+03	-2.346999e+03	-2.347028e+03	-2.347056e+03	-2.347084e+03	-2.347112e+03	-2.347140e+03	-2.347125e+03	-2.347093e+03
6	-2.347017e+03	-2.347044e+03	-2.347070e+03	-2.347097e+03	-2.347123e+03	-2.347150e+03	-2.347177e+03	-2.347203e+03	-2.347187e+03	-2.347152e+03
7	-2.347092e+03	-2.347117e+03	-2.347141e+03	-2.347166e+03	-2.347191e+03	-2.347216e+03	-2.347241e+03	-2.347266e+03	-2.347248e+03	-2.347212e+03
8	-2.347168e+03	-2.347190e+03	-2.347212e+03	-2.347235e+03	-2.347259e+03	-2.347282e+03	-2.347306e+03	-2.347329e+03	-2.347309e+03	-2.347272e+03
9	-2.347243e+03	-2.347263e+03	-2.347283e+03	-2.347305e+03	-2.347326e+03	-2.347348e+03	-2.347370e+03	-2.347392e+03	-2.347371e+03	-2.347332e+03
10	-2.347323e+03	-2.347347e+03	-2.347371e+03	-2.347390e+03	-2.347409e+03	-2.347428e+03	-2.347447e+03	-2.347466e+03	-2.347443e+03	-2.347403e+03
11	-2.347405e+03	-2.347436e+03	-2.347467e+03	-2.347482e+03	-2.347498e+03	-2.347513e+03	-2.347529e+03	-2.347544e+03	-2.347519e+03	-2.347478e+03
12	-2.347488e+03	-2.347526e+03	-2.347562e+03	-2.347574e+03	-2.347586e+03	-2.347598e+03	-2.347610e+03	-2.347623e+03	-2.347595e+03	-2.347553e+03
13	-2.347570e+03	-2.347615e+03	-2.347658e+03	-2.347666e+03	-2.347675e+03	-2.347684e+03	-2.347692e+03	-2.347701e+03	-2.347672e+03	-2.347627e+03
14	-2.347653e+03	-2.347704e+03	-2.347753e+03	-2.347759e+03	-2.347764e+03	-2.347769e+03	-2.347774e+03	-2.347779e+03	-2.347748e+03	-2.347702e+03
15	-2.347707e+03	-2.347764e+03	-2.347819e+03	-2.347827e+03	-2.347835e+03	-2.347843e+03	-2.347852e+03	-2.347860e+03	-2.347828e+03	-2.347780e+03
16	-2.347714e+03	-2.347773e+03	-2.347830e+03	-2.347853e+03	-2.347876e+03	-2.347899e+03	-2.347922e+03	-2.347945e+03	-2.347915e+03	-2.347865e+03
17	-2.347720e+03	-2.347781e+03	-2.347841e+03	-2.347879e+03	-2.347917e+03	-2.347954e+03	-2.347992e+03	-2.348030e+03	-2.348002e+03	-2.347949e+03
18	-2.347726e+03	-2.347790e+03	-2.347853e+03	-2.347905e+03	-2.347957e+03	-2.348010e+03	-2.348062e+03	-2.348114e+03	-2.348089e+03	-2.348033e+03
19	-2.347732e+03	-2.347798e+03	-2.347864e+03	-2.347931e+03	-2.347998e+03	-2.348065e+03	-2.348132e+03	-2.348199e+03	-2.348176e+03	-2.348118e+03
20	-2.347735e+03	-2.347804e+03	-2.347874e+03	-2.347955e+03	-2.348037e+03	-2.348118e+03	-2.348200e+03	-2.348281e+03	-2.348261e+03	-2.348200e+03
21	-2.347576e+03	-2.347677e+03	-2.347778e+03	-2.347862e+03	-2.347946e+03	-2.348031e+03	-2.348115e+03	-2.348199e+03	-2.348190e+03	-2.348144e+03
22	-2.347417e+03	-2.347551e+03	-2.347681e+03	-2.347769e+03	-2.347856e+03	-2.347943e+03	-2.348030e+03	-2.348117e+03	-2.348120e+03	-2.348088e+03
23	-2.347258e+03	-2.347424e+03	-2.347585e+03	-2.347675e+03	-2.347765e+03	-2.347855e+03	-2.347945e+03	-2.348035e+03	-2.348049e+03	-2.348032e+03
24	-2.347099e+03	-2.347297e+03	-2.347489e+03	-2.347582e+03	-2.347675e+03	-2.347768e+03	-2.347861e+03	-2.347953e+03	-2.347978e+03	-2.347976e+03
25	-2.346940e+03	-2.347170e+03	-2.347393e+03	-2.347489e+03	-2.347584e+03	-2.347680e+03	-2.347776e+03	-2.347871e+03	-2.347907e+03	-2.347920e+03
26	-2.346687e+03	-2.346932e+03	-2.347170e+03	-2.347276e+03	-2.347382e+03	-2.347488e+03	-2.347594e+03	-2.347700e+03	-2.347753e+03	-2.347785e+03
27	-2.346388e+03	-2.346640e+03	-2.346884e+03	-2.347005e+03	-2.347125e+03	-2.347245e+03	-2.347366e+03	-2.347486e+03	-2.347558e+03	-2.347611e+03
28	-2.346090e+03	-2.346348e+03	-2.346599e+03	-2.346734e+03	-2.346868e+03	-2.347002e+03	-2.347137e+03	-2.347271e+03	-2.347363e+03	-2.347437e+03

Abbildung 50: Z-Werte einer Oberfläche

Der allgemeine Gedankengang des Algorithmus wurde im vorigen Kapitel vorgestellt:

1. Suchen Sie den Maximalwert.
2. Suchen Sie die Zeile und Spalte des Maximalwerts.
3. Finden Sie die umliegenden Werte anhand der Zeile und Spalte des Maximalwerts
4. Berechnen Sie die Differenz (Δz) zwischen dem Maximalwert und den umliegenden Werten.
5. Vergleichen Sie Δz mit dem eingestellten Grenzwert. Wenn Δz größer als der Grenzwert ist, wird es als Ausreißer betrachtet und entfernt.
6. Programmieren Sie die obigen Schritte in eine Schleife, um die verbleibenden Werte zu verarbeiten.

Dann befolgen Sie die Schritte zum Schreiben eines vorläufigen Programms. Im Programm: r, c repräsentieren die Zeile und Spalte des ausgewählten Wertes.

```
[SOURCEXSIZE, SOURCEYSIZE] = size(SOURCE);
Grenzwert = 5;

for i = 1:20
    MAX = max(max(SOURCE))
    [r,c]=find(SOURCE==MAX)
    Umliegendwert = SOURCE(r-1:r+1,c-1:c+1);
    DeltaZ = MAX - Umliegendwert

  if (DeltaZ(DeltaZ > Grenzwert))
       SOURCE(r,c) = nan;
  end

end
```

Abbildung 51: vorläufiges Programm

Um zunächst der allgemeine Rahmen des Algorithmus zu testen, können wir zunächst einen 100x100-Ausschnit aus MountainsMap exportieren. Dieser 100×100-Text wird als Testobjekt verwendet. Dadurch können wir den Code Schritt für Schritt verbessern, während wir verschiedene Probleme lösen.

Programmtest:

```
MAX =

   59.4540


r =

    57


c =

    47


Umliegendwert =

   59.4160   59.3150   59.2130
      NaN   59.4540   59.3400
   59.3400   57.9650   56.5900


DeltaZ =

    0.0380    0.1390    0.2410
       NaN         0    0.1140
    0.1140    1.4890    2.8640


MAX =

   59.4540


r =

    57


c =

    47


Umliegendwert =

   59.4160   59.3150   59.2130
      NaN   59.4540   59.3400
   59.3400   57.9650   56.5900
```

Abbildung 52: Problem 1

Aus der obigen Abbildung können wir Folgendes erkennen: Wenn der aktuell erkannte Wert die Bedingungen für Ausreißer nicht erfüllt, bzw. nicht entfernt wird, kann das Programm diesen Wert nicht überspringen und den nächsten Wert erkennen. Das bedeutet, dass das Programm nicht weiter ausgeführt werden kann.

Um das Problem zu lösen, können wir die Variable B einführen und die unique-Funktion verwenden, um die SOURCE-Matrix zu sortieren. Wie in Zeile 5 des Codes gezeigt. Zeile 6 soll das Array in umgekehrter Reihenfolge sortieren, also vom größten zum kleinsten. Der Maximalwert wird nacheinander aus Variable B ausgewählt. Wenn ein Wert nicht als Ausreißer betrachtet und entfernt wird, kann der nicht gelöschten Wert übersprungen werden. Die Anzahl der Schleife ist die Länge der Variablen B.

```
5   B = unique(SOURCE)
6   B = B(end:-1:1)
7   n = length(B);
8
9   for i = 1:n
10      MAX = B(i)
```

Abbildung 53: Lösung für Problem 1

Nach der Lösung dieses Problems gibt es ein neues Problem: Es gibt mehrere gleich große Werte über die gesamte Oberfläche. Wie in der Abbildung gezeigt, gibt es zwei Punkte mit einem Wert von 59.3400. Es wird aber nur einer der Werte verarbeitet.

```
MAX =

   59.3400


r =

    58
    57


c =

    46
    48


Umliegendwert =

   52.3860       NaN   59.4540
   52.5390   59.3400   57.9650
   52.7000   58.8640   55.4180


DeltaZ =

    6.9540       NaN   -0.1140
    6.8010         0    1.3750
    6.6400    0.4760    3.9220


MAX =

   59.3150
```

Abbildung 54: Problem 2

Die Anzahl von gleichen Maximalwerten kann durch die 14. Codezeile bekannt sein. Die Schleifenstruktur in Zeile 15 verarbeitet die wiederholten Werte nacheinander. Das Problem der doppelten Werte kann gelöst werden, indem der Hauptteil des Algorithmus in diese Schleife geschrieben wird.

```
13          [r,c]=find(SOURCE==MAX)
14          gleicheMAX = length(r)
15        for j = 1:gleicheMAX
16            rj = r(j)
17            cj = c(j)
18
19
20        Umliegendwert = SOURCE(rj-1:rj+1,cj-1:cj+1)
21        DeltaZ = MAX - Umliegendwert
22
23         if (DeltaZ(DeltaZ > Grenzwert))
24           SOURCE(rj,cj) = nan;
25         end
26        end
```

Abbildung 55: Lösung für Problem 2

```
MAX =

   59.3400


r =

    58
    57


c =

    46
    48


gleicheMAX =

     2

rj =

    58

cj =

    46

Umliegendwert =

   52.3860       NaN   59.4540
   52.5390   59.3400   57.9650
   52.7000   58.8640   55.4180

DeltaZ =

    6.9540       NaN   -0.1140
    6.8010         0    1.3750
    6.6400    0.4760    3.9220

rj =

    57

cj =

    48

Umliegendwert =

   59.3150   59.2130   59.1110
   59.4540       NaN   59.2260
   57.9650       NaN   55.2140

DeltaZ =

    0.0250    0.1270    0.2290
   -0.1140       NaN    0.1140
    1.3750       NaN    4.1260
```

Abbildung 56: Ergebnis der Lösung des Problems 2

Die nächste Frage lautet: Es ist möglich, dass die höheren Punkte auf der Grenze der gesamten Oberfläche liegen. In diesem Fall werden die umliegenden Punkte nicht mehr 8 sein und Array-Grenzen werden 100 überschreiten.

```
MAX =

   10.7140


r =

   100


c =

    24


gleicheMAX =

     1


rj =

   100


cj =

    24

Index in position 1 exceeds array bounds (must not exceed 100).
```

Abbildung 57 : Problem 3

Um das Problem zu lösen, müssen wir ein Programm schreiben, um die Zeilen und Spalten für Sonderfälle zu begrenzen.

```
19            if(rj == 1 && cj == 1)
20              Umliegendwert = SOURCE(1:2,1:2);
21
22             elseif(rj == 1 && cj == SOURCEYSIZE)
23               Umliegendwert = SOURCE(1:2,SOURCEYSIZE-1:SOURCEYSIZE);
24
25             elseif(rj == SOURCEXSIZE && cj == 1)
26               Umliegendwert = SOURCE(SOURCEXSIZE-1:SOURCEXSIZE,1:2);
27
28             elseif(rj == SOURCEXSIZE && cj == SOURCEYSIZE)
29               Umliegendwert = SOURCE(SOURCEXSIZE-1:SOURCEXSIZE,SOURCEYSIZE-1:SOURCEYSIZE);
30
31             elseif(rj == 1)
32                 Umliegendwert = SOURCE(1:rj+1,cj-1:cj+1);
33
34             elseif(rj == SOURCEXSIZE)
35                 Umliegendwert = SOURCE(rj-1:SOURCEXSIZE,cj-1:cj+1);
36
37             elseif(cj ==1)
38                 Umliegendwert = SOURCE(rj-1:rj+1,1:cj+1);
39
40             elseif(cj ==SOURCEYSIZE)
41                 Umliegendwert = SOURCE(rj-1:rj+1,cj-1:SOURCEYSIZE);
42
43             else
44              Umliegendwert = SOURCE(rj-1:rj+1,cj-1:cj+1);
45             end
```

Abbildung 58: Programm zum Begrenzen von Zeilen und Spalten

Es gibt zwei sogenannte Spezialfälle: Der erste ist, dass der Maximalwert in den vier Ecken liegt. Die Zeilen 19, 22, 25, 28 sind für die Lösung dieser Situation verantwortlich. Zweitens liegt das Maximum an den vier Grenzen. Die Zeilen 31, 34, 37, 40 sind für die Lösung dieser Situation verantwortlich. Mit dem Programm in obiger Abbildung werden die dem jeweiligen Fall entsprechenden umliegende Werte aufgelistet.

Um die Ergebnisse klarer zu sehen, können wir eine 5x5-Matrix erstellen, um den Code zu testen.

```
SOURCE =

   100     2    92     3    99
     4     8     6     6     7
    97     8     9    10    95
     3     6     8     9     6
    94     6    98     3    96
```

Abbildung 59: 5x5-Matrix

Programmtest:

```
MAX =

   100

Umliegendwert =

   100     2
     4     8

MAX =

    99

Umliegendwert =

     3    99
     6     7

MAX =

    94

Umliegendwert =

     3     6
    94     6

MAX =

    96

Umliegendwert =

     9     6
     3    96

MAX =

    98

Umliegendwert =

     6     8     9
     6    98     3

MAX =

    97

Umliegendwert =

     4     8
    97     8
     3     6

MAX =

    95

Umliegendwert =

     6     7
    10    95
     9     6

MAX =

    92

Umliegendwert =

     2    92     3
     8     6     6
```

Abbildung 60: Ergebnis vom Programmtest

4.2.1 Bestimmung von Grenzwerten

Der Hauptcode zum Erkennen von Ausreißern wurde fertiggestellt, und der nächste Schritt besteht darin, den Code für die Berechnung des Grenzwerts zu schreiben, um das gesamte Programm abzuschließen.

Die Methode wurde in Kapitel 2 beschrieben. Der nächste Schritt ist die Implementierung und Verbesserung des Programms in Matlab.

Der erste Schritt ist das Auslesen der Daten. Als nächstes verwenden wir hauptsächlich die Histogram-Funktion in Matlab, um ein Histogramm zu erstellen. Ein Histogramm ist eine Art Balkendiagramm für numerische Daten, das die Daten in Bins gruppiert. Nachdem ein Histogramm Objekt erstellt wurde, können verschiedene Aspekte des Histogramms geändert werden, indem seine Eigenschaftswerte geändert werden.

Die Eigenschaften des Histogramms werden wie folgt eingestellt: Bereiche für z-Koordinate R= max -min. Erstellung des Histogramms mithilfe der „probability“-Normalisierung. Nachdem das Histogramm erstellt wurde, können die Bin-Breite (L) und Balkenhöhen (Häufigkeit) berechnet werden. Bei dieser Normalisierung ist die Höhe jedes Balkens gleich der Häufigkeit, dass eine Beobachtung innerhalb dieses Bin-Intervalls ausgewählt wird, und die Höhen aller Balken summieren sich zu 1.

```
clear ,clc;
A = load("M.txt");
z = A(:,3);
z = z +2350;
h = histogram(z,'BinLimits',[min(z),max(z)],Normalization="probability")
L = h.BinWidth;
Haeufigkeit = h.Values;
```

Abbildung 61: Erstellung des Histogramms und Parameter

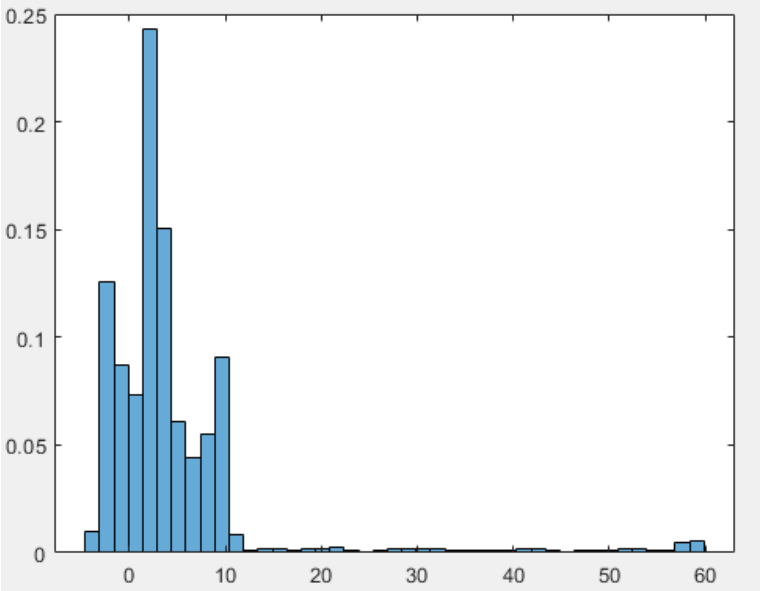


Abbildung 62: Histogramm

Mit den berechneten Parametern kann die minimale Häufigkeit gefunden werden. Dieses Intervall kann als Trennlinie verwendet werden, und der mittlere Wert des Intervalls kann als Trennpunkt verwendet werden. Das gesamte Histogramm ist in zwei Klassen unterteilt. Links ist die Normale Topografie, rechts der Ausreißer Bereich. Dann wird der Abstand DeltaZ zwischen den Klassen von Häufigkeiten als Grenzwert benutzten. Aber dieser Wert ist zu groß, nach Diskussion wurde entschieden, die 4-fache Standardabweichung von diesem Wert abzuziehen. Dieser Wert wird als neuer Grenzwert verwendet.

```
 9    Trennlinie = find(Haeufigkeit == min(Haeufigkeit))
10    Trennpunkt = h.BinEdges(Trennlinie)+L/2
11    NormaleTopografie = z(z <= Trennpunkt)
12    Ausreisser = z(z > Trennpunkt)
13    Abstand = mean(Ausreisser) - mean(NormaleTopografie)
14    Grenzwert = Abstand - 4*std(z)
```

Abbildung 63: Berechnung von Grenzwerten

Es ist möglich, dass das Histogramm Lücken aufweist, d. h. das Intervall, in dem die Häufigkeit gleich 0 ist. In diesem Fall kann man in dem Intervall weitersuchen, um das nächste Intervall findet, wo die Häufigkeit größer als 0 ist.

```
 9    Trennlinie = find(Haeufigkeit == min(Haeufigkeit))
10    if (min(Haeufigkeit) == 0)
11        Trennlinie = Trennlinie(end)+1
12    end
```

Abbildung 64: Lösungen für Sonderfälle

4.2.2 Erstellung der Topografie

Um die Änderungen der Topografie durch den Algorithmus intuitiv zu simulieren, können wir zunächst eine dreidimensionale Oberfläche durch die Surf-Funktion konstruieren.

Der erste Schritt besteht zweifellos darin, die Daten in Matlab einzulesen. Die A-Matrix enthält 3 Datenspalten, nämlich x, y und z. Und die SOURCE-Matrix enthält nur Höhenangeben(Z-Koordinaten). Wie Abbildung 65 gezeigt.

```
1    clear,clc;
2    A = load("1000x1000.txt");
3    SOURCE = load("1000x1000 (z).txt");
```

Abbildung 65: Die Datei lesen

Bei Verwendung der Surf-Funktion müssen die Daten in Rasterform vorliegen. Wir müssen also die meshgrid-Funktion verwenden, um die Extremwerte von x und y zu kombinieren. Dann können die Gitterkoordinaten generieren.

Bei der Verwendung der meshgrid-Funktion muss jedoch zuerst die Größe des Gitters bestimmt werden. Da wir Ausreißer in der Topografie entfernen, entspricht die Gittergröße der Größe der Z-Wert-Matrix. Deshalb können wir die linspace-Funktion verwenden, um die Größe des Gitters direkt anzugeben. Die Größe ist gleich length(SOURCE).

Diese Funktion gibt zweidimensionale Gitterkoordinaten basierend auf den Koordinaten zurück, die in den Vektoren x und y enthalten sind. Sowohl xq als auch yq sind Matrizen. Jede Zeile ist eine Kopie von x, und jede Spalte ist eine Kopie von y.

```
5   x = A(:,1);
6   y = A(:,2);
7   z = A(:,3);
8   n = length(SOURCE);
9
10  [xq,yq] = meshgrid(linspace(min(x),max(x),n),linspace(min(y),max(y),n));
```

Abbildung 66: meshgrid-Funktion

Jetzt haben wir drei eindimensionale Arrays von x, y, z, diese Art von Daten wird "scattered data" genannt. Um eine Oberfläche basierend auf bekannten Streupunkten ohne Oberflächengleichung zu zeichnen, muss die griddata-Funktion verwendet werden, um die Streupunkte zu interpolieren.

Hier wird es mit Hilfe der Funktion [X, Y, Z] = griddata(x,y,z,xq,yq) implementiert. Auf diese Weise erhalten wir die X-, Y-, Z-Werte, die die oben genannten Bedingungen erfüllen. 3D-Topografie kann durch Surf-Funktion erstellt werden.

```
12  [X,Y,Z] = griddata(x,y,z,xq,yq);
13  surf(X,Y,Z)
```

Abbildung 67: Die Funktion griddata und surf

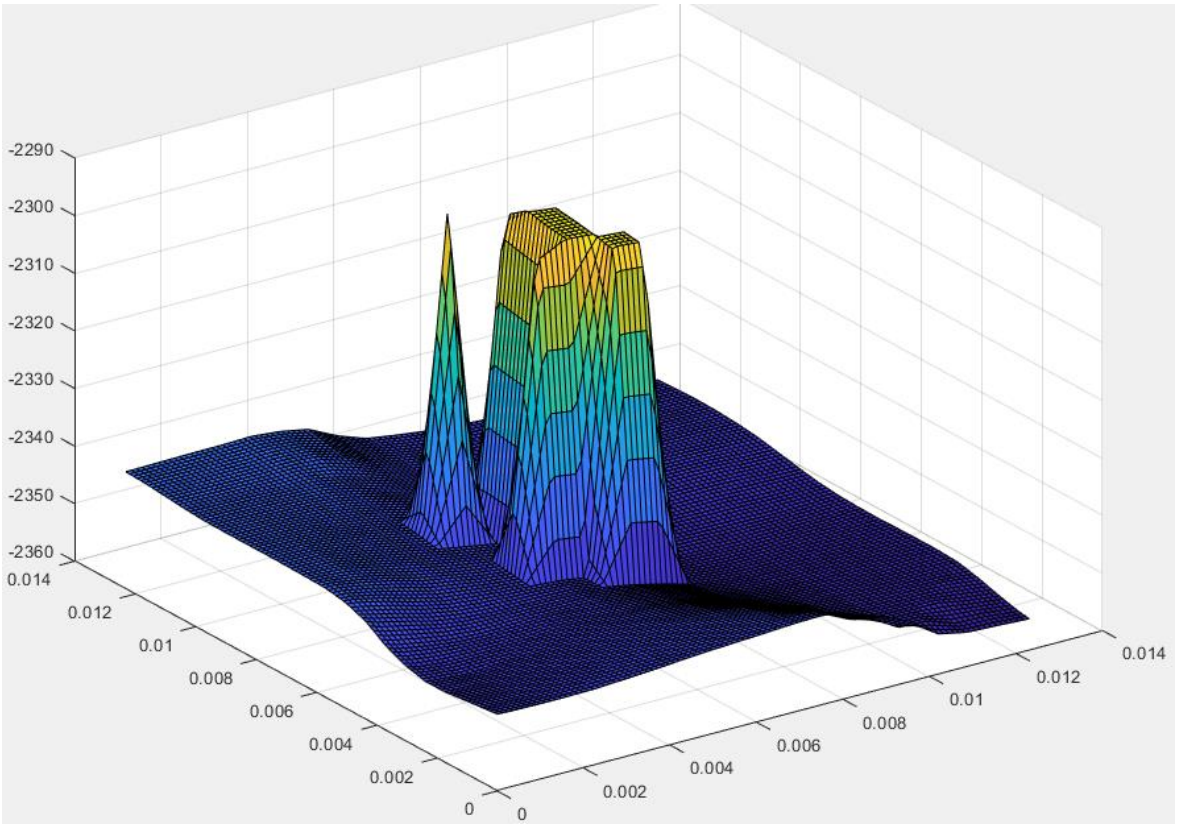


Abbildung 68: Beispiel Topografie(100x100)

Der obige Code hat einen Fehler: Er funktioniert nur für kleinere Topografie. Wenn zum Beispiel die Topografiegröße 1000 x 1000 erreicht, wird die gesamte Topografie sehr dunkel. (Siehe Abbildung 69)

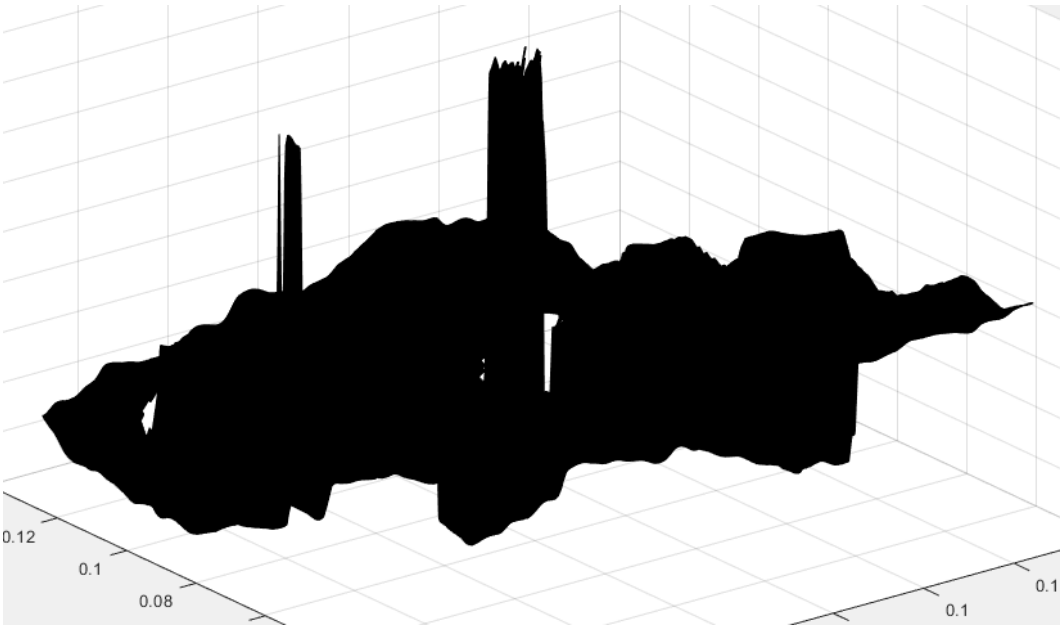


Abbildung 69: dunkele Topografie

Um dieses Problem zu lösen, können wir schwarze Gitterlinien reduzieren, z.B. nur jede fünfte zeichnen. Zuerst verwenden wir die Shading-Interp-Funktion. Ändert die Farbe in jeder Linie oder jedem Bereich durch Interpolieren des Colormap-Index oder TrueColor-Werts in dieser Linie oder diesem Bereich.

Die Shading-Funktion steuert die Farbschattierung von Oberflächen- und Patch-Grafikobjekten. Diese Funktion beinhaltet Shading Flat, Shading Faceted und Shading Interp.

Der Unterschied zwischen diesen drei Funktionen ist wie folgt:

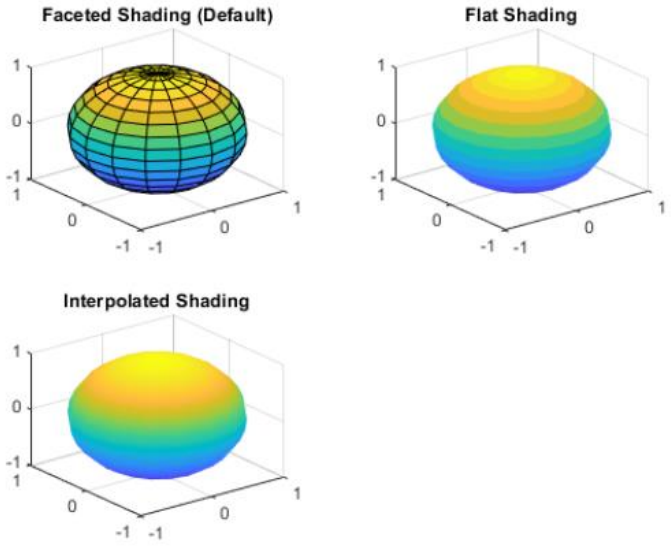


Abbildung 70: Der Unterschied zwischen Shading-funktionen [13]

Die Shading-Interp-Funktion kann den Farbübergang reibungslos gestalten. Dies erleichtert es uns, die Gitterlinien als nächstes neu zu zeichnen.

Die Wirkung der Shading-Funktion ist wie folgt:

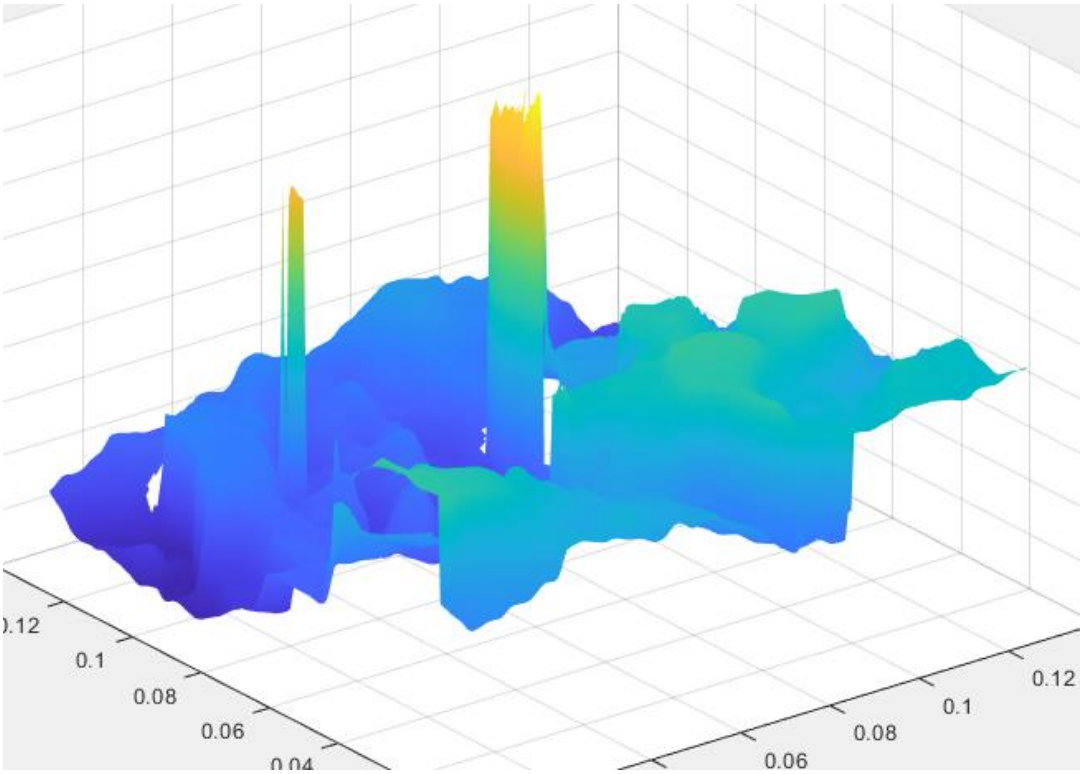


Abbildung 71: Shading-Interp-Funktion

```
15    shading interp   %schwarze Gitterlinien reduzieren, z.B. nur jede fünfte zeichnen
16    line(X(:,1:5:end),Y(:,1:5:end),Z(:,1:5:end),'color','k')
17    line(X(1:5:end,:)',Y(1:5:end,:)',Z(1:5:end,:)','color','k')
```

Abbildung 72: Zeichnung den Gitterlinien

Als nächstes wird die line-funktion verwendet, um jede fünf Punkte in 3D-Koordinaten eine Linie zu zeichnen. Die Farbe des Gitters ist "k", bzw. schwarz.

Schließlich wird die Topografie sichtbar:

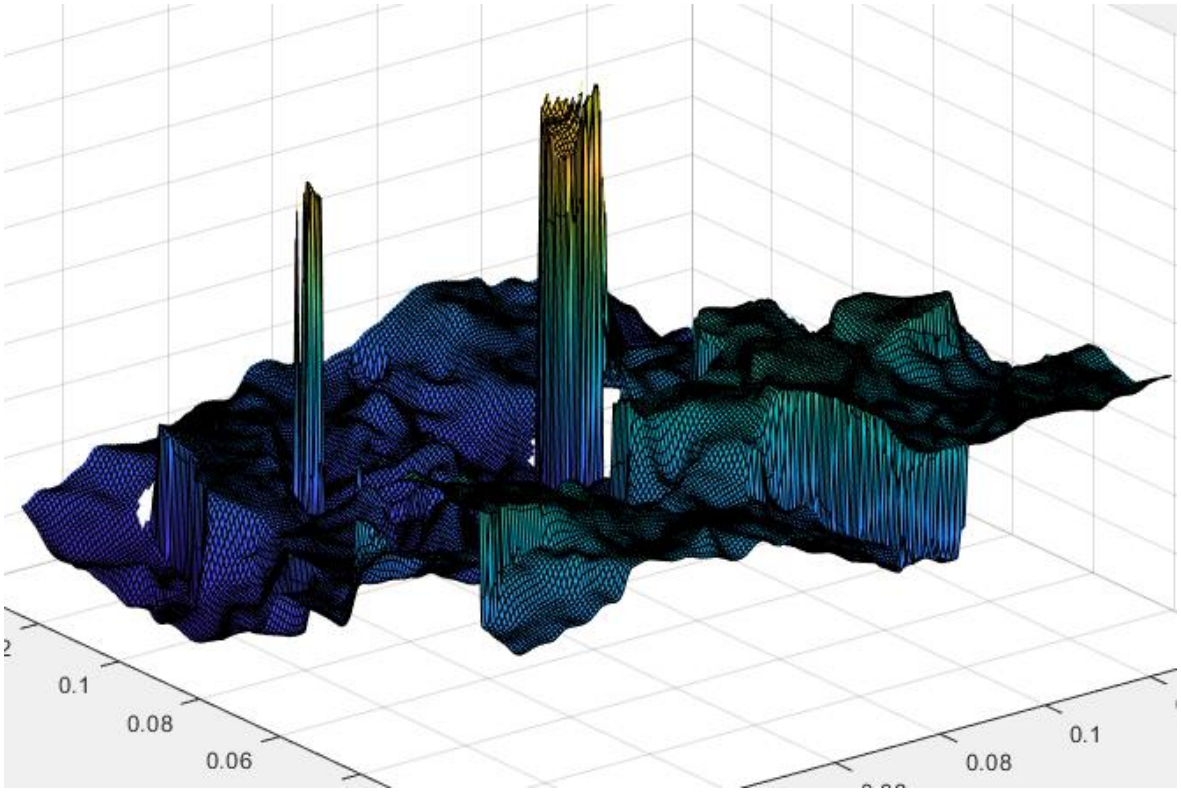


Abbildung 73: verarbeitete Topografie

5 Test von Ausreißer Entfernen

In diesem Abschnitt wird der im vorherigen Kapitel geschriebene Code getestet. Da die Hauptaufgabe dieser Arbeit die Entfernung von Ausreißern ist, zeigt dieses Kapitel nur die Ergebnisse des Ausreißer Algorithmus.

5.1 Test auf Ausreißer Entfernung

Im Testkapitel wird zunächst ein Programmablauf angehängt, um die allgemeinen Schritte zu erläutern. (Siehe Abbildung 74)

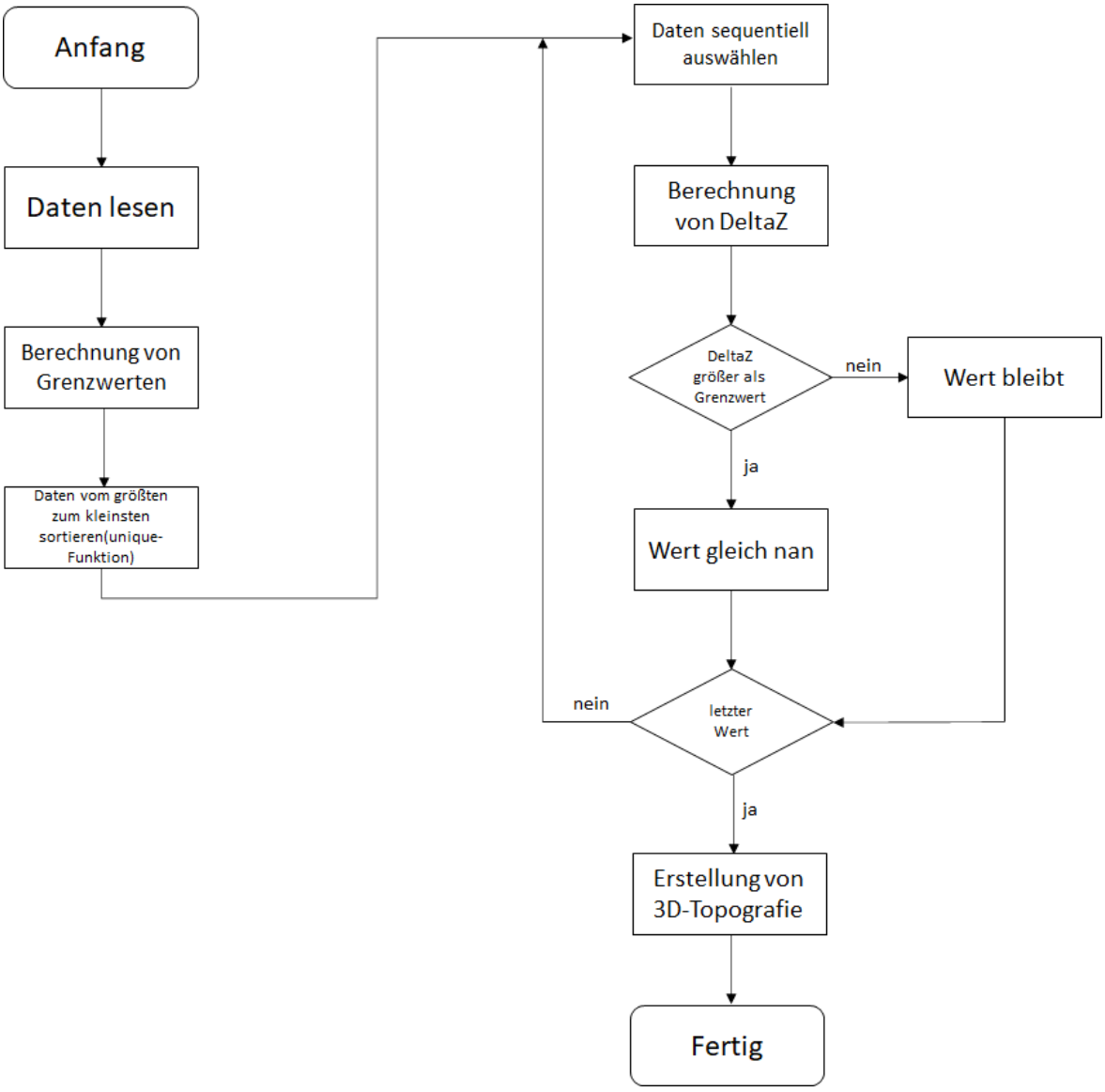


Abbildung 74: Programmablauf

Als nächstes werden einige getestete Topografien und die Ergebnisse gezeigt.

100×100-Topografie:

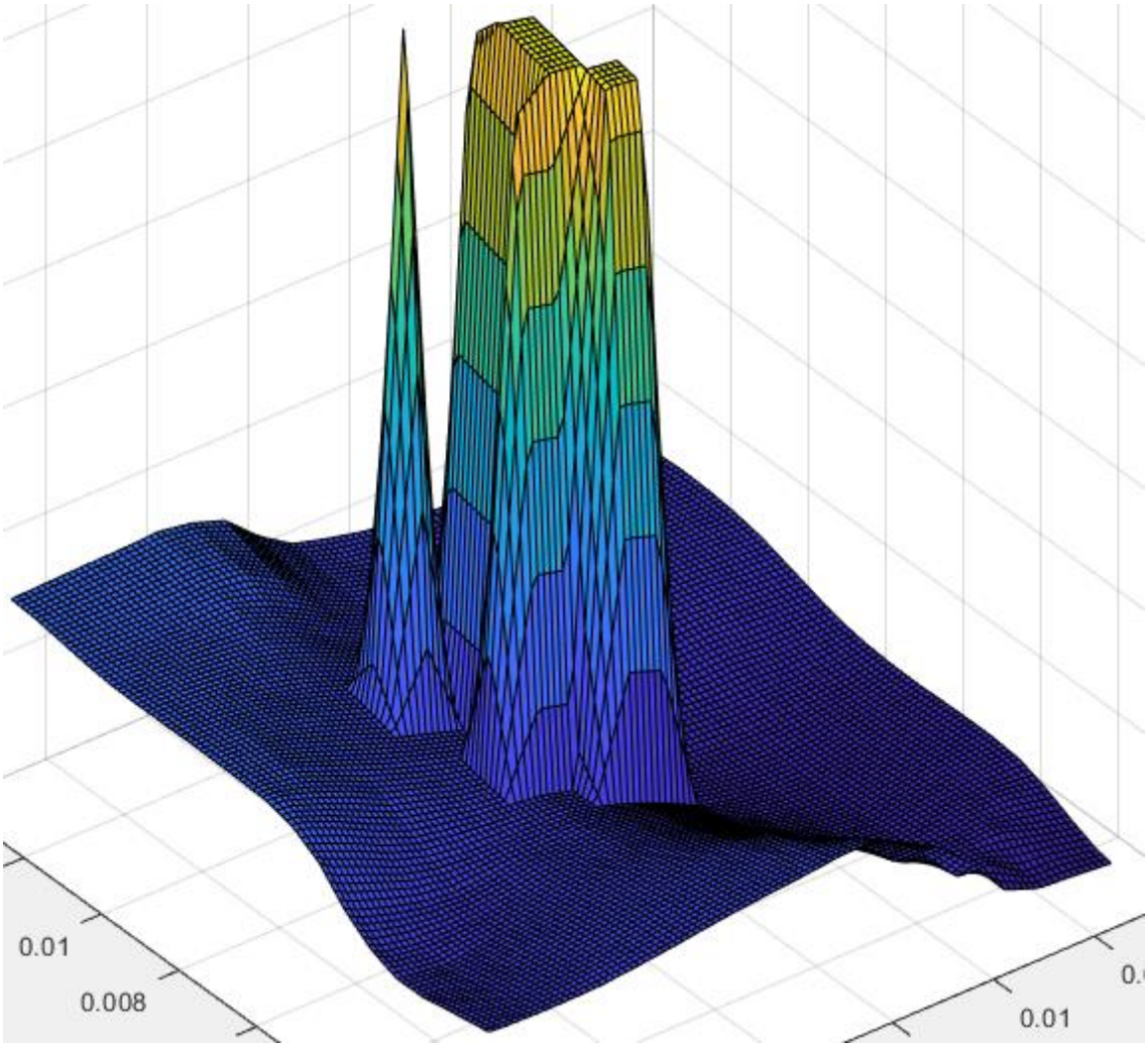


Abbildung 75: originale 100x100-Tpografie

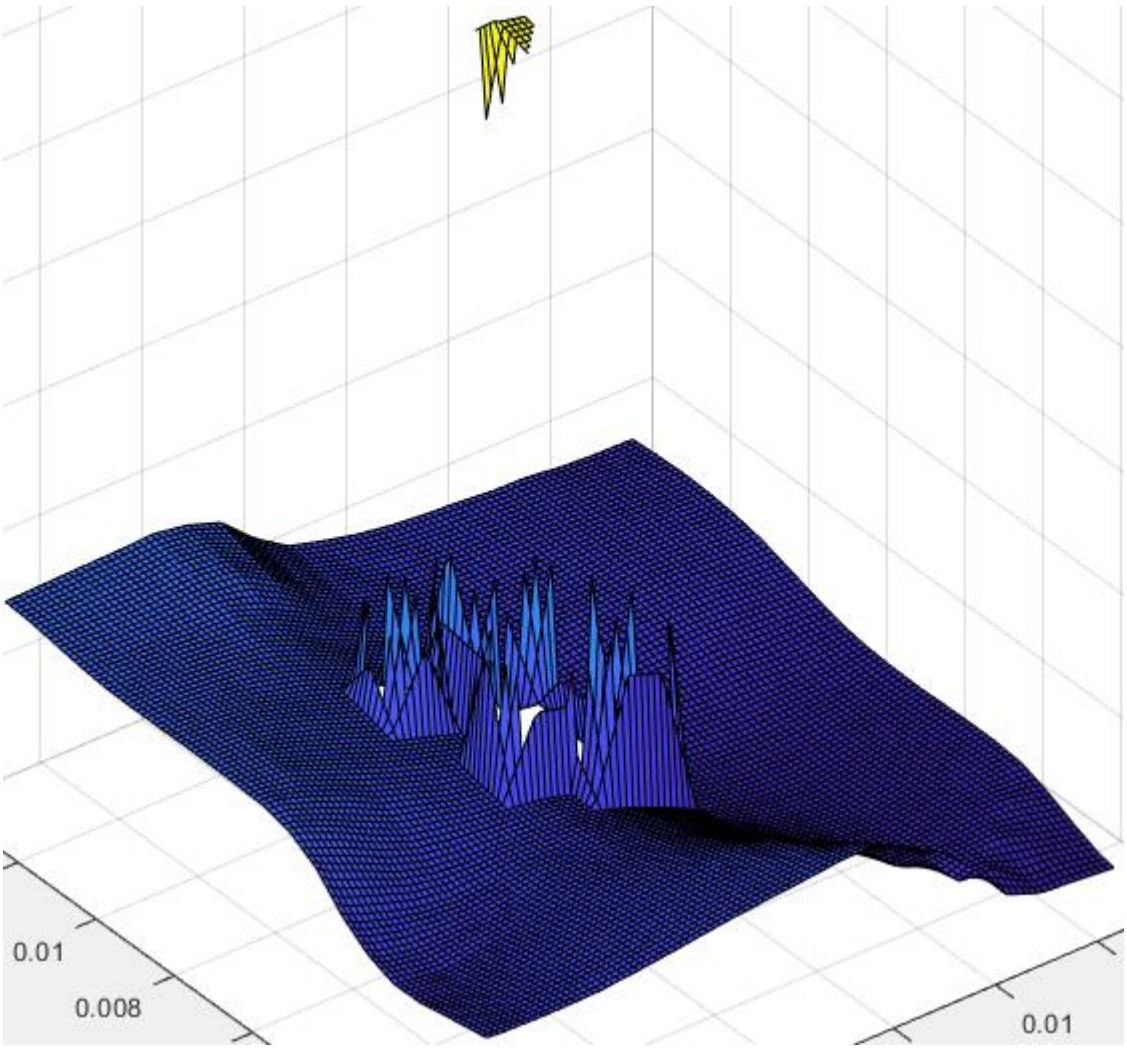


Abbildung 76: verarbeitete 100x100-Tpografie

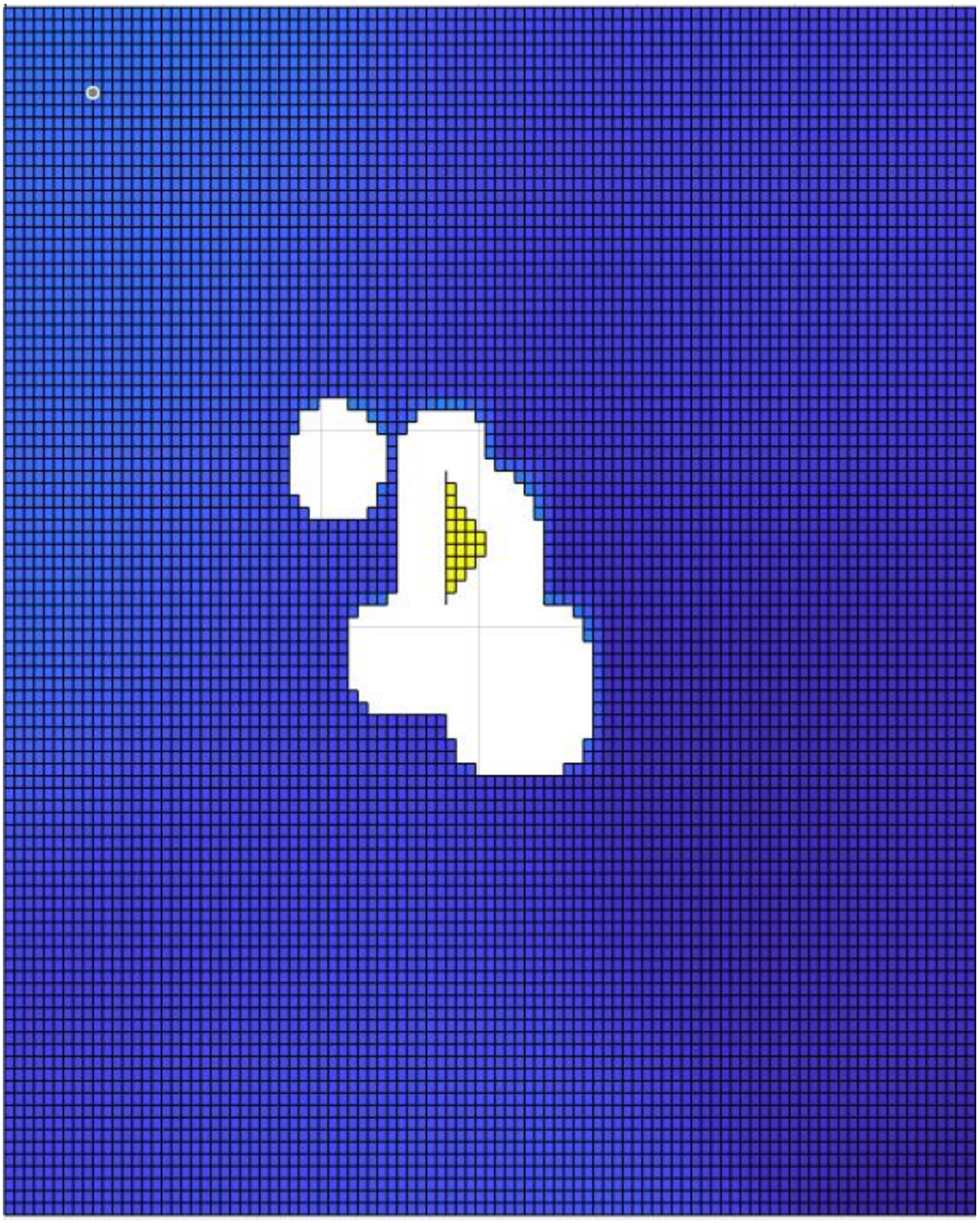


Abbildung 77: verarbeitete 100x100-Tpografie (Aufsicht)

200×200-Topografie:

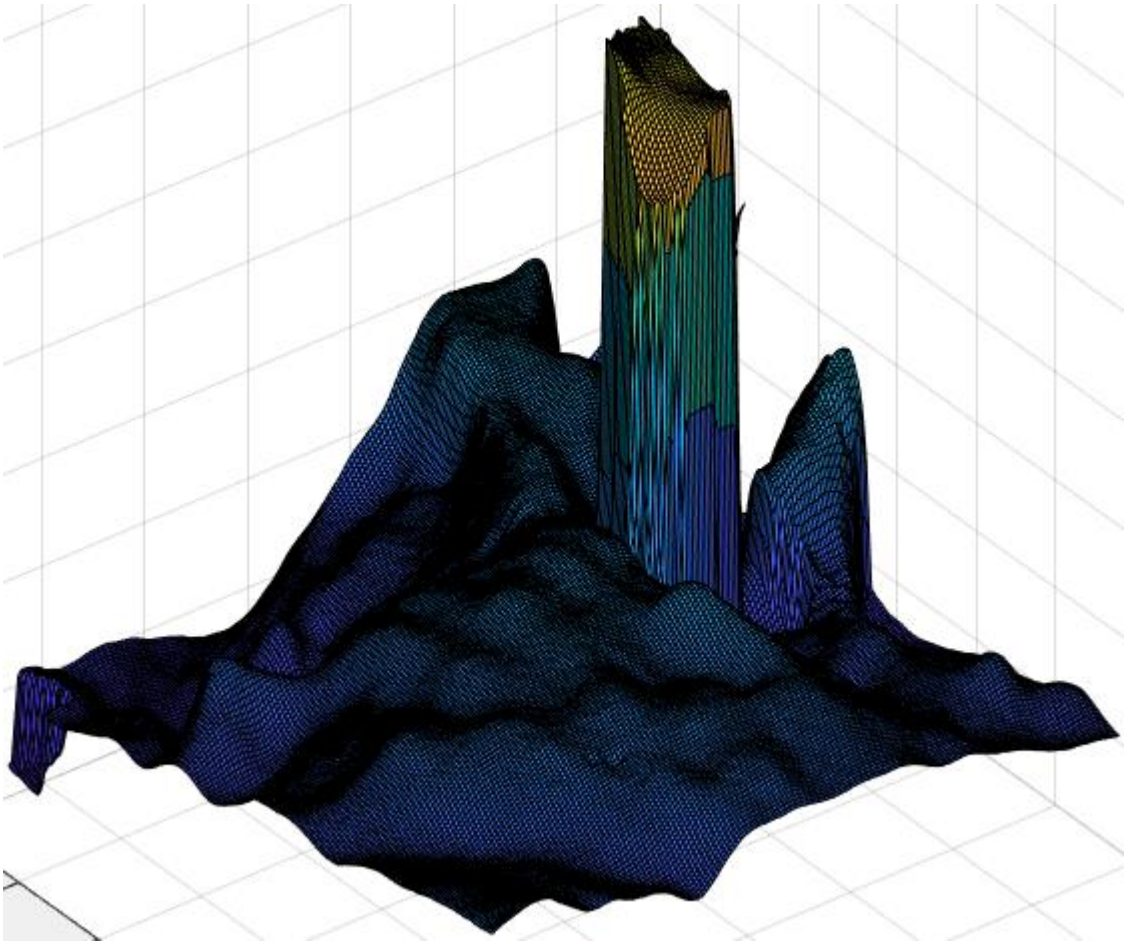


Abbildung 78: originale 200x200-Tpografie

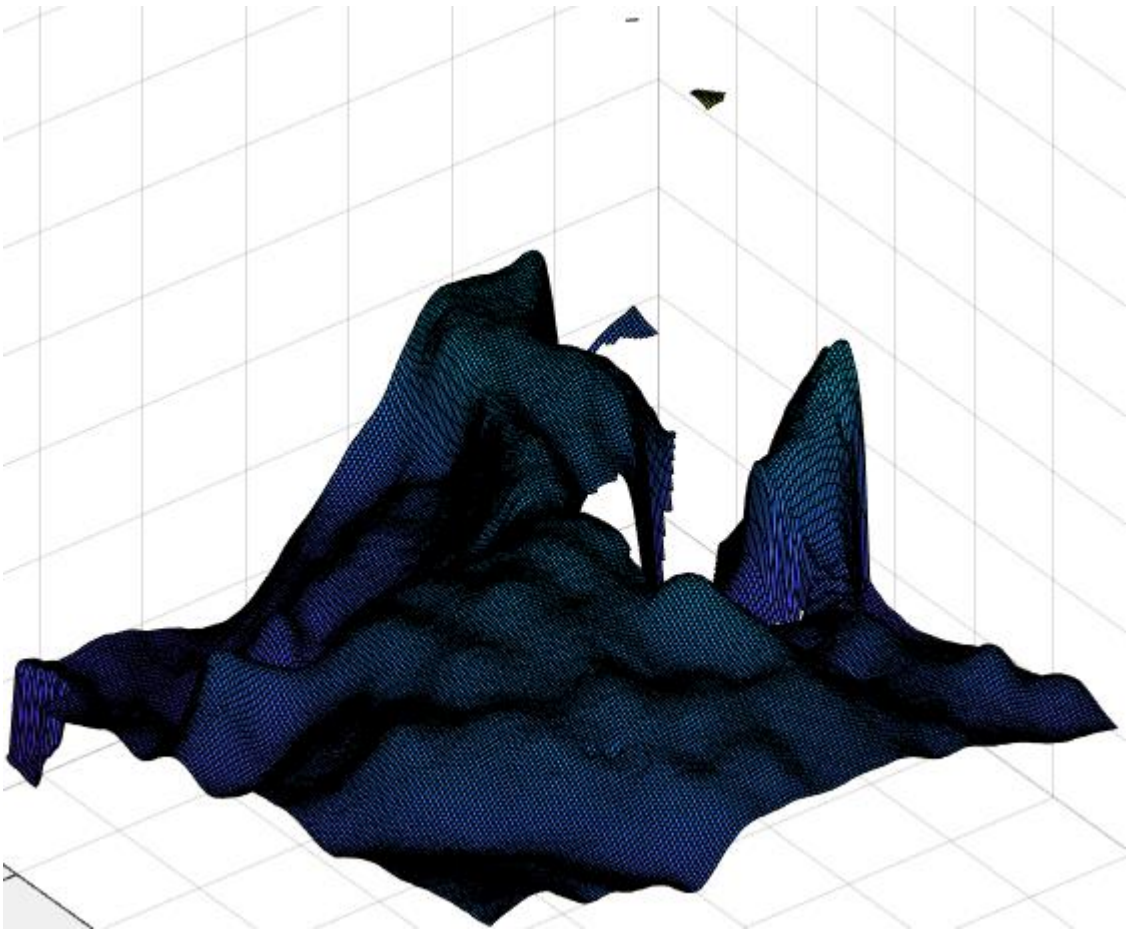


Abbildung 79: verarbeitete 200x200-Tpografie

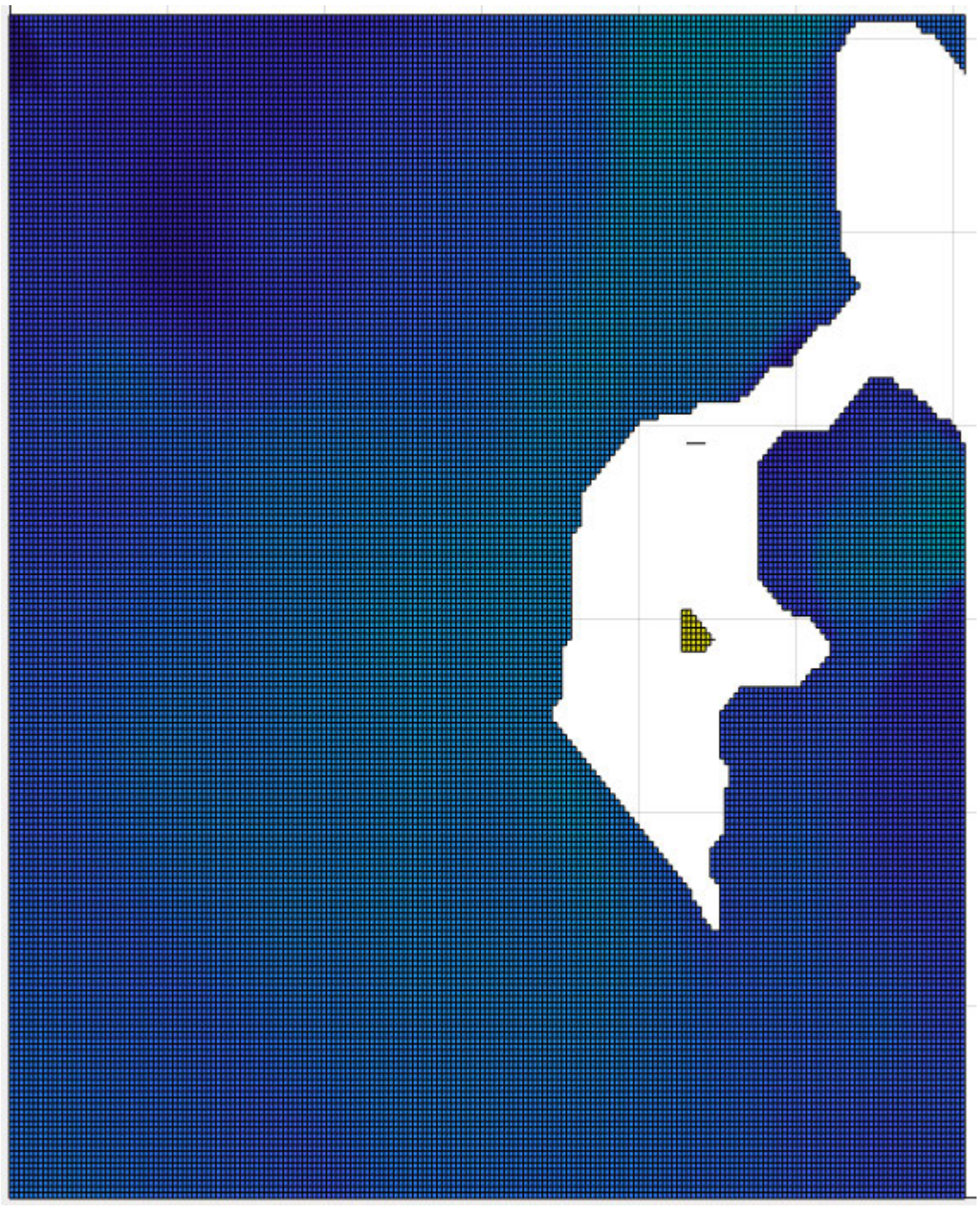


Abbildung 80: verarbeitete 200x200-Tpografie (Aufsicht)

1000×1000-Topografie:

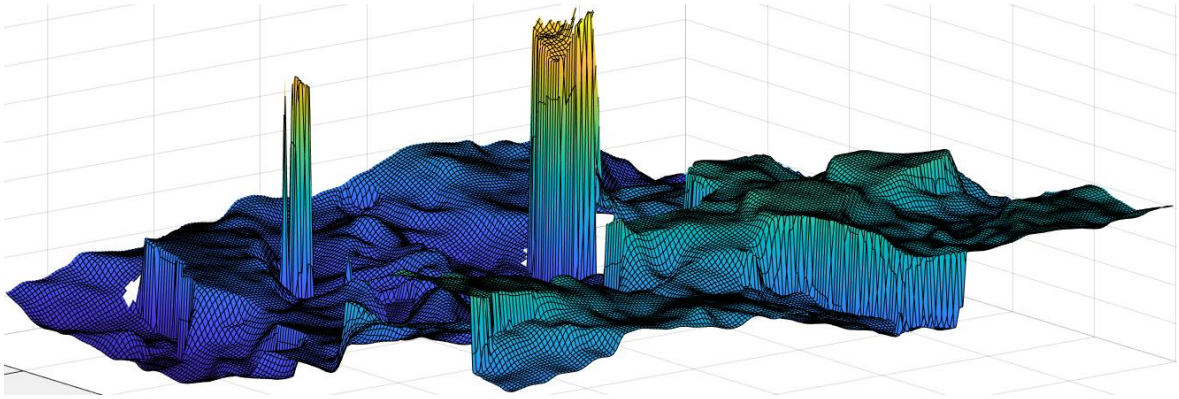


Abbildung 81: originale 200x200-Tpografie

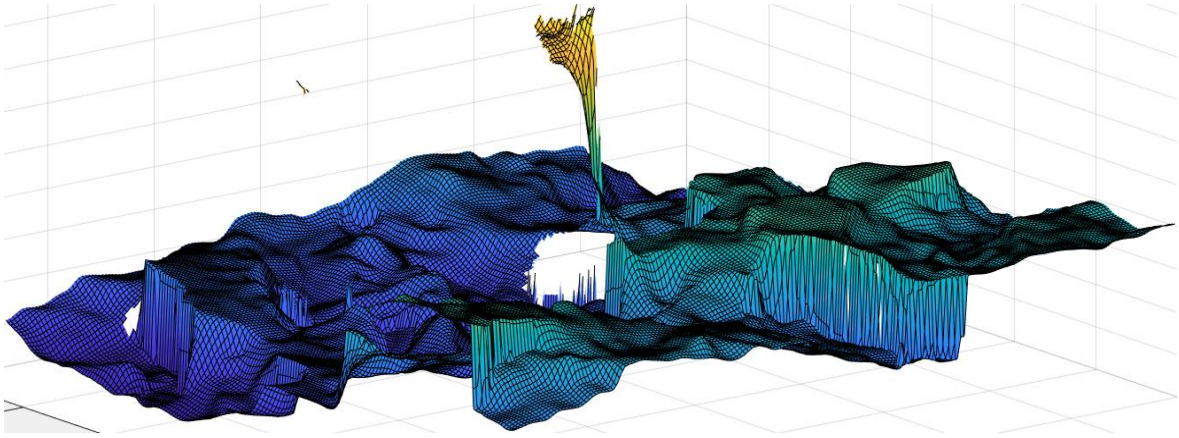


Abbildung 82: verarbeitete 1000x1000-Tpografie

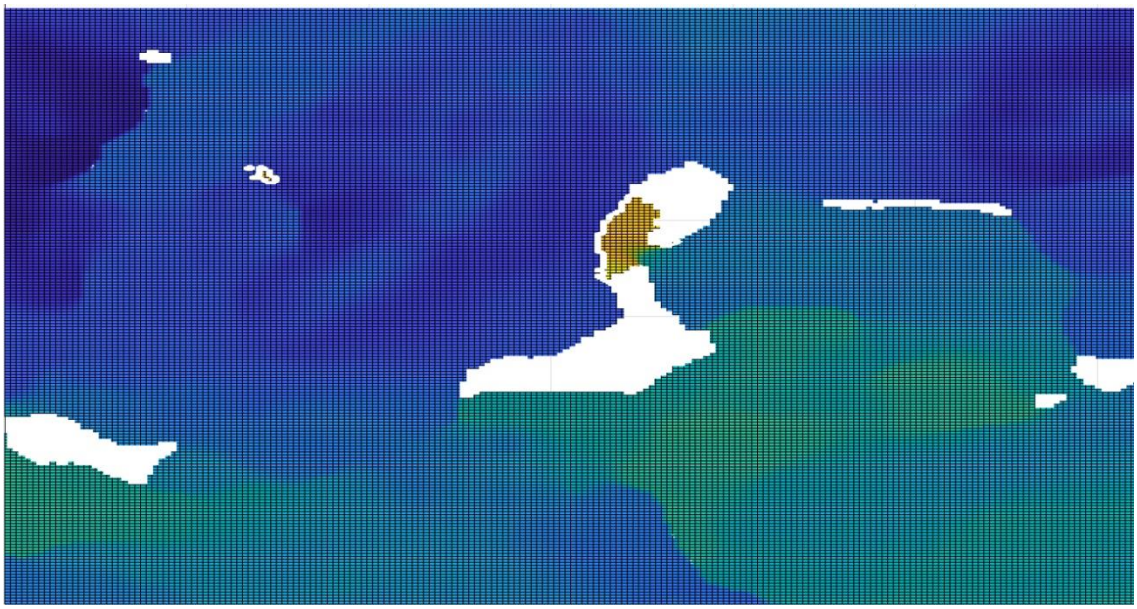


Abbildung 83: verarbeitete 1000x1000-Tpografie (Aufsicht)

Bei 1000x1000-Topografie änderte sich die Methode der Grenzwertberechnung. (Siehe Abbildung 84)

```
clear ,clc;
A = load("1000x1000.txt");
z = A(:,3);
h = histogram(z,'BinLimits',[min(z),max(z)],Normalization="probability");
L = h.BinWidth;
Haeufigkeit = h.Values;
Minimum = min(Haeufigkeit);
Trennlinie = find(Haeufigkeit == min(Haeufigkeit));
peak1 = find(Haeufigkeit == max(Haeufigkeit(1:Trennlinie)));
peak2 = find(Haeufigkeit == max(Haeufigkeit(Trennlinie+1:end)));
MAX1 = h.BinEdges(peak1)+L/2;
MAX2 = h.BinEdges(peak2)+L/2;
Trennpunkt = (MAX2 + MAX1)/2
NormaleTopografie = z(z <= Trennpunkt);
Ausreisser = z(z > Trennpunkt);
Abstand = mean(Ausreisser) - mean(NormaleTopografie)
Grenzwert = Abstand - 2*nanstd(Ausreisser)%(es gibt nan)
```

Abbildung 84: Neue Methode zur Grenzwertberechnung

Hier ist der Mittelwert der Doppelpeaks als Trennlinie genommen. Und nach Diskussion wurde beschlossen, die 2-fache Standardabweichung des Ausreißer Bereichs zu subtrahieren. Dieser Wert wird als neuer Grenzwert verwendet.

Aus den obigen Ergebnissen ist ersichtlich, dass: Gegenüber dem normalen Algorithmus von Mountainsmap ist bei diesem Algorithmus Verbesserung für mehr Topografie Übersicht. Punkte, die zur normalen Topografie gehören, werden so weit wie möglich beibehalten und Ausreißer werden so weit wie möglich entfernt.

Aber es gibt immer noch einige Ausreißer, die nicht entfernt werden können, 100x100-Topografie kann als Beispiel erklärt werden.

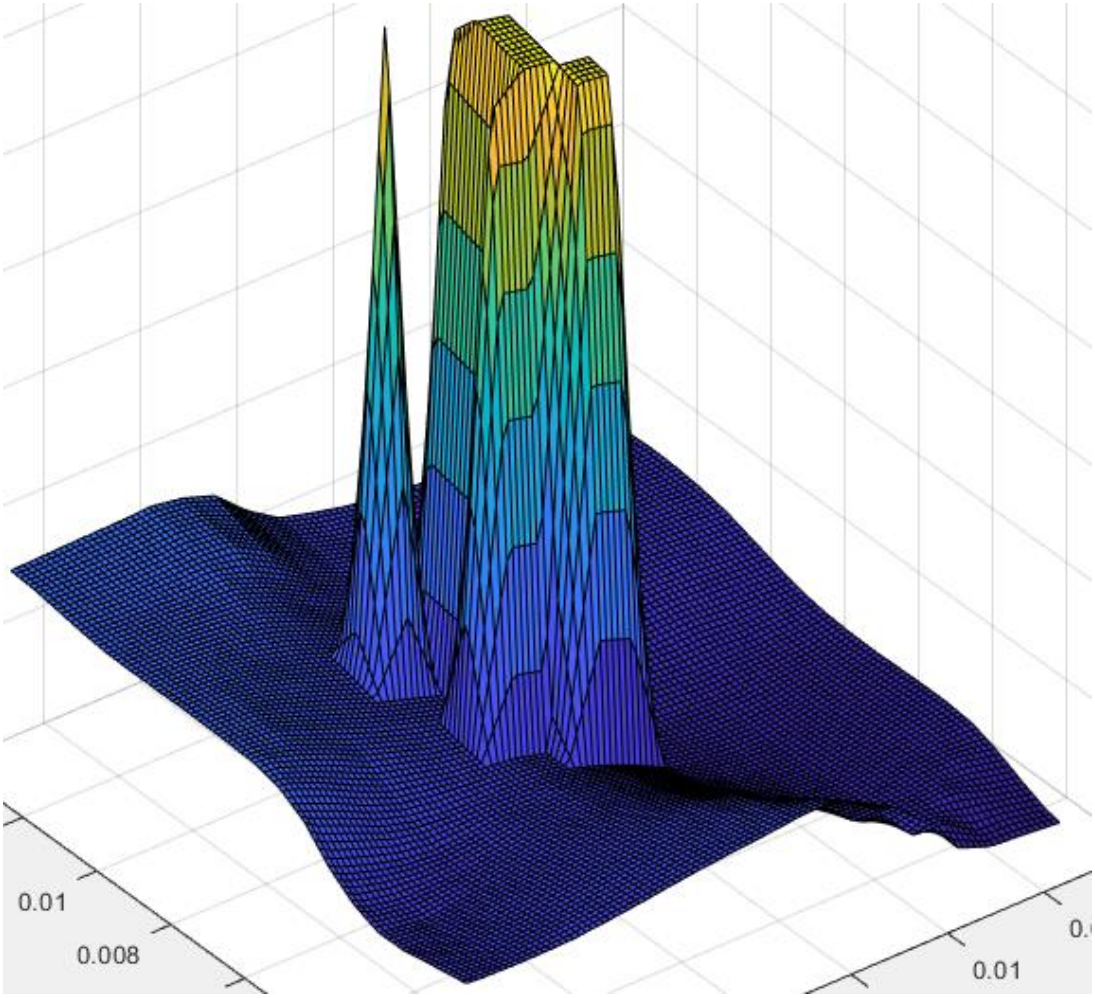


Abbildung 85: 100x100-Topografie

Wie in Abbildung 85 gezeigt: Ausreißer Bereich links ist wie ein Kegel, alle Punkte liegen in der Flanke. DeltaZ ist also groß und die Toleranz von Grenzwert ist relativ groß und lässt sich leichter entfernen. Der Ausreißer Bereich rechts sieht aus wie ein Trapez, wobei einige Hochpunkte eine Plattform bilden. Das DeltaZ zwischen ihnen ist sehr klein, daher ist die Toleranz des Grenzwerts auch sehr klein und es ist schwierig, es zu entfernen. Und es ist sehr wahrscheinlich, dass auch die Punkte, die zur normalen Topografie gehören, stark entfernt werden. Dies ist immer noch ein Problem.

Als nächstes kommt das Problem zur Schnittstellenfunktion. (Siehe Abbildung 86)

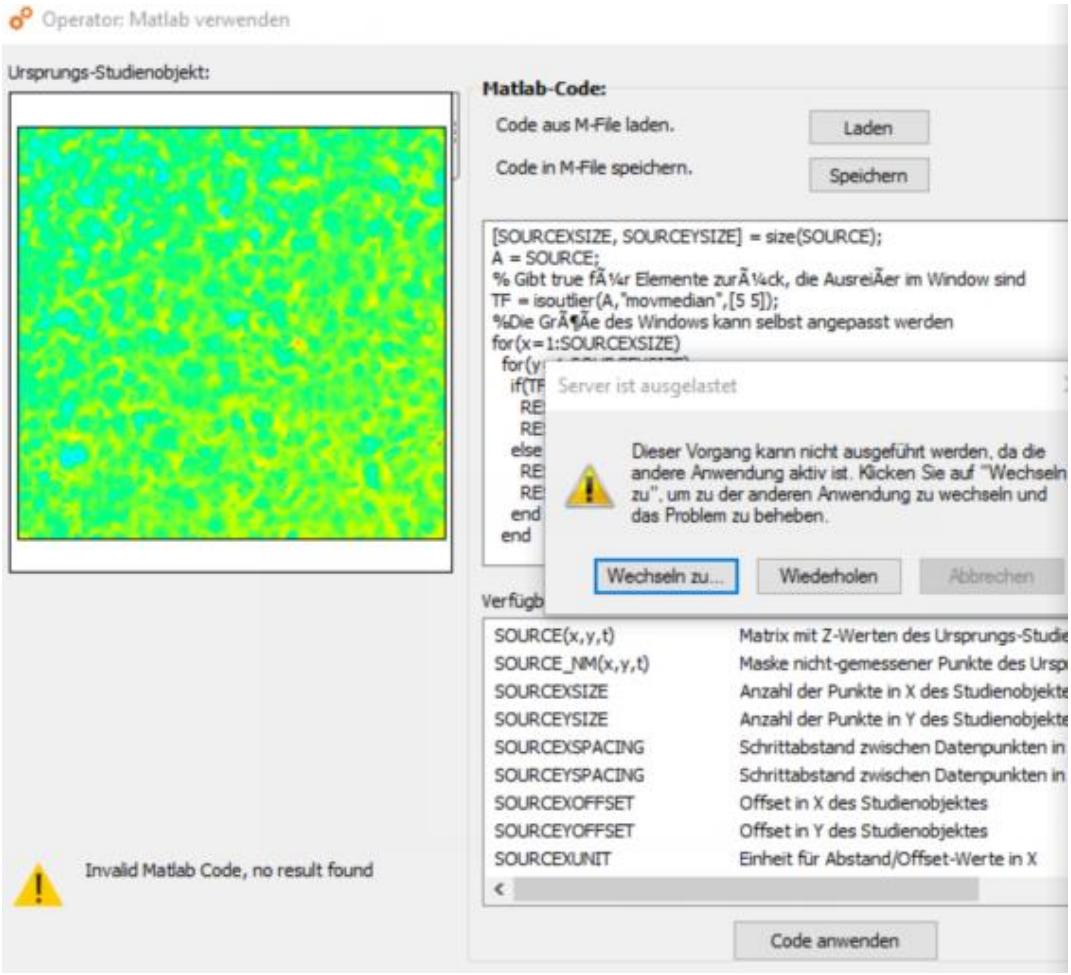


Abbildung 86: Schnittstellenfunktion

Wenn diese Funktion zum Bearbeiten von Oberflächen verwendet wird, ist sie sehr langsam. Der „Server ist ausgelastet“ Fenster erscheint für eine Weile. Das Bild ist ein sehr einfacher Matlab-Code, der für Experimente verwendet wird, aber die Geschwindigkeit ist immer noch sehr langsam.

Ich habe auch die Mitarbeiter der Digital Surf Company nach dieser Frage gefragt, und sie haben 2 Lösungen gegeben: Diese Fehlermeldung wird im Allgemeinen angezeigt, wenn Matlab Zeit braucht, um eine Antwort zu geben. Dies kann daran liegen, dass die Verarbeitung durch Matlab zu komplex ist oder wenn die Oberfläche zu groß ist. Wäre es möglich, es mit einer kleineren Oberfläche oder mit der gleichen Oberfläche zu versuchen, aber neu resampled.

Tatsächlich ist die Wirkung nicht ideal. Und jedes Mal, wenn die mnt-Datei geöffnet wird, erscheint diese Fehlermeldung und muss das Programm erneut ausgeführt werden, d.h. das Ergebnis wird nicht direkt angezeigt. Dies ist sehr zeitaufwändig und mühsam. Daher wird die Ausreißer Entfernung direkt in Matlab simuliert.

6 Zusammenfassung und Ausblick

Dieses Kapitel bietet eine Zusammenfassung der Arbeit. Im Abschnitt Ausblick wird vorgestellt: Was sind die Mängel dieses Programms.

6.1 Zusammenfassung

Diese Arbeit befasst sich mit der Oberflächenvorverarbeitung, und das Entfernen von Ausreißern ist ein wichtiger Punkt, bevor die Krater bewerten können. Optische Messgeräte wird zu Ausreißern führen. Dieser Ausreißer kann einen erheblichen negativen Einfluss auf die Ermittlung der Kenngrößen haben. Aber der Algorithmus in Mountainsmap war manchmal nicht zweckmäßig, weil meist Ausreißer übrigbleiben. Daher wird noch ein neuer Algorithmus benötigt. Diese Punkte sind für unsere Forschung von entscheidender Bedeutung.

Danach wurde die Oberflächentopografie in Mountainsmap verarbeitet und die Daten zur Analyse in Matlab importiert. Die 3D-Topografie wird mit Hilfe der Surf-Funktion von Matlab erstellt, um den neuen Algorithmus zur Entfernung von Ausreißern zu testen. Die Funktion Schnittstelle Mountains-Matlab wird mithilfe des Beispiels von Filterung untersucht. Im Kapitel Test werden die Ergebnisse meines Algorithmus gezeigt: Gegenüber dem normalen Algorithmus von Mountainsmap ist es ersichtlich, dass Punkte, die zu normalem Gelände gehören, erhalten bleiben und Ausreißer so weit wie möglich entfernt werden. Die Funktion Matlab-verwenden in Mountainsmap wurde erfolgreich getestet und fand auch einige Probleme mit dieser Funktion.

6.2 Ausblick

Hinsichtlich der Verbesserung des Algorithmus ist diese Arbeit jedoch nur der Anfang. Obwohl dieser Algorithmus einige Verbesserungen vorgenommen hat, gibt es immer noch Probleme. Wie man mit diesen Höhepunkten umgeht, der Unterschied zwischen ihnen ist sehr gering, ist immer noch ein Problem. Gleichzeitig sollen die zur normalen Topografie gehörenden Punkte so wenig wie möglich beeinträchtigt werden, deshalb ist es besonders wichtig, den geeigneten Grenzwert für verschiedene Topografie zu wählen. Auch dieser Teil muss optimiert werden. Es gibt auch das Problem der Verarbeitungsgeschwindigkeit. Die Geschwindigkeit steigt zwar, wenn die verarbeitete Oberfläche kleiner ist, aber das ist zu ineffizient. Wie man diese beiden Faktoren in Einklang bringt, ist schwierig. Kurz gesagt, die Optimierung des Algorithmus hat noch einen langen Weg vor sich.

Literatur

[1] Die Erklärung von Sandstrahltechnik, online Quelle: https://sandstrahlen-marktplatz.de/ (Abgerufen am 12.09.2022)

[2] TWISTER und TORNADO, online Quelle: http://tornado.twister-sand-strahl-anlage.de/strahlanlagen/ (Abgerufen am 6.04.2022)

[3] Prof. Dr.-Ing. René Pleul: Messung technischer Oberflächen mit Tastschnittgeräten, 2014

[4] Die Vorteile von Optische Instrument, online Quelle: https://www.confovis.com/produkte/optische-messsysteme/toolinspect/ (Abgerufen am 6.04.2022)

[5] Das Konzept der Faltung, online Quelle: https://www.biancahoegel.de/mathe/analysis/faltung.html (Abgerufen am 6.04.2022)

[6] Einführung in die Ausreißer Entfernung, online Quelle: https://guide.digitalsurf.com/de/leitfaden-filtertechniken.html (Abgerufen am 16.09.2022)

[7] 3sigma-Kriterium, online Quelle: https://matheguru.com/stochastik/normalverteilung.html (Abgerufen am 17.09.2022)

[8] Box-Plot, online Quelle: https://de.wikipedia.org/wiki/Box-Plot (Abgerufen am 17.09.2022)

[9] Dominik Aehle：Empirische Modellbildung des Fertigungsergebnisses bei der Herstellung gestrahlter Oberflächen，2022

[10] Einführung in die Surf-Funktion, online Quelle: https://ww2.mathworks.cn/help/matlab/ref/surf.html (Abgerufen am 7.04.2022)

[11] Einführung der Mountainsmap-Software, online Quelle: https://de.wikipedia.org/wiki/MountainsMap (Abgerufen am 7.04.2022)

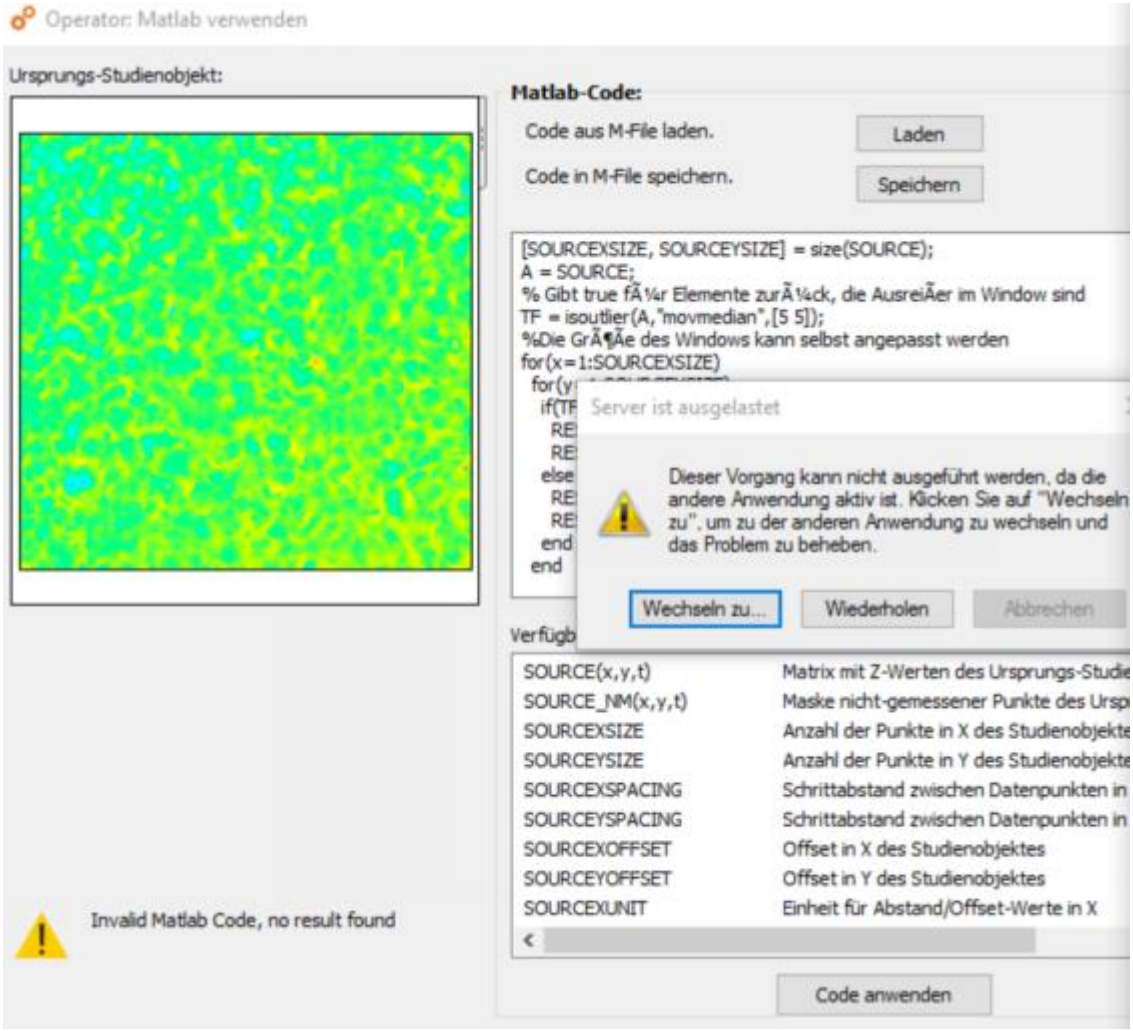
[12] Befehle in Mountainsmap-Software, online Quelle: https://www.digitalsurf.com/ (Abgerufen am 13.09.2022)

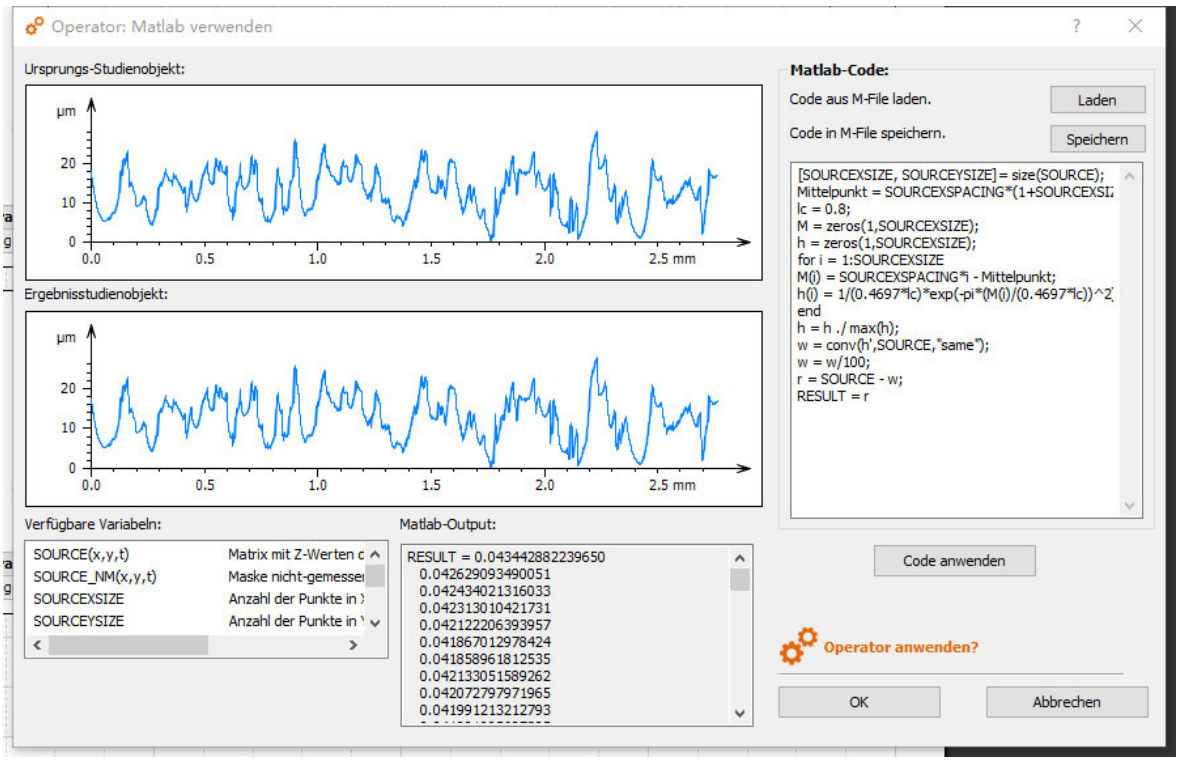
[13] Einführung in Shading-Funktionen, online Quelle: https://ww2.mathworks.cn/help/matlab/ref/shading.html (Abgerufen am 14.09.2022)

Anlagen

Anlagen, Teil 1

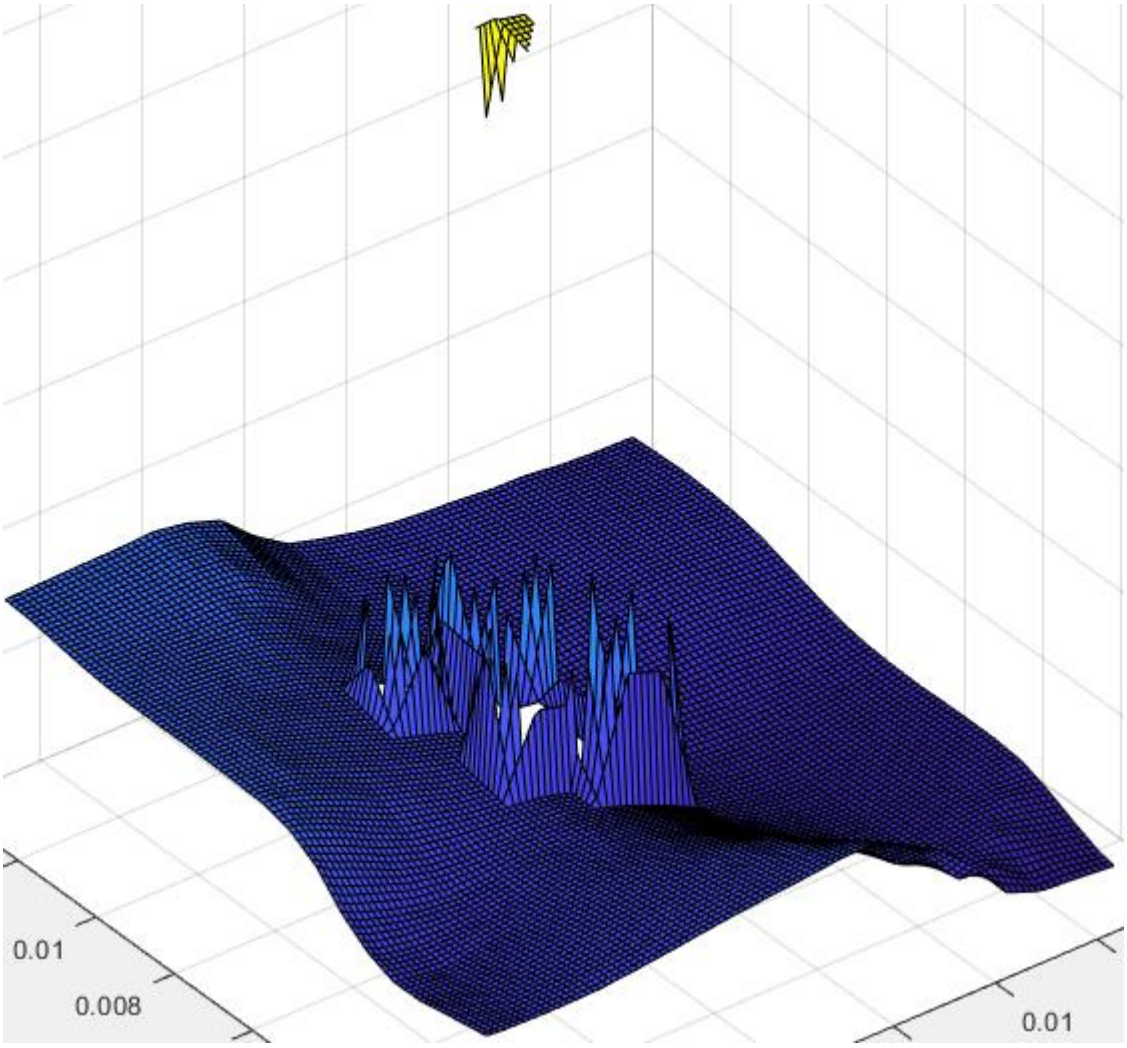
Schnittstelle Mountains-Matlab:

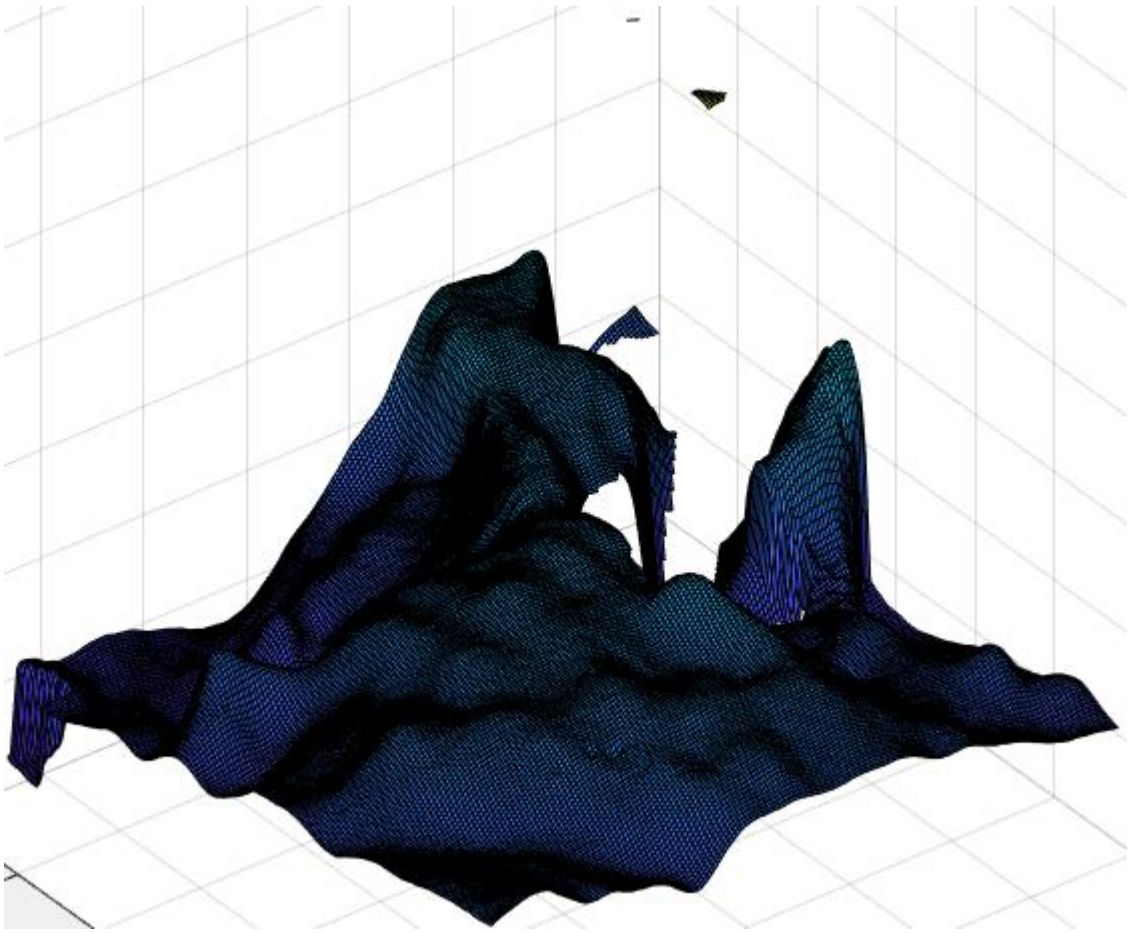


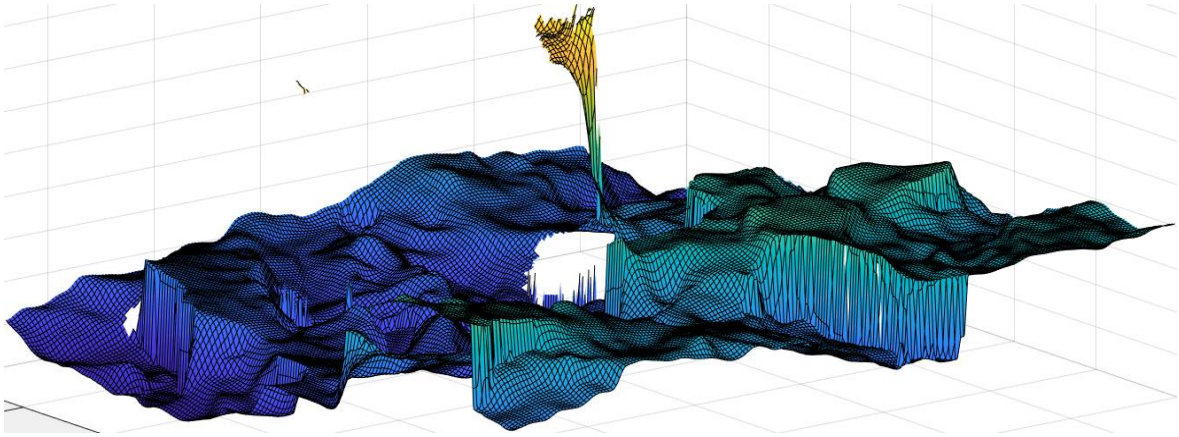

Operator: Matlab verwenden
Ursprungs-Studienobjekt:
µm
20
10
0
0.0
0.5
1.0
1.5
2.0
2.5 mm
Ergebnisstudienobjekt:
µm
20
10
0
0.0
0.5
1.0
1.5
2.0
2.5 mm
Verfügbare Variabeln:
SOURCE(x,y,t) Matrix mit Z-Werten d
SOURCE_NM(x,y,t) Maske nicht-gemesse
SOURCEXSIZE Anzahl der Punkte in
SOURCEYSIZE Anzahl der Punkte in
Matlab-Output:
RESULT = 0.043442882239650
0.042629093490051
0.042434021316033
0.042313010421731
0.042122206393957
0.041867012978424
0.041858961812535
0.042133051589262
0.042072797971965
0.041991213212793
Matlab-Code:
Code aus M-File laden.
Laden
Code in M-File speichern.
Speichern
[SOURCEXSIZE, SOURCEYSIZE] = size(SOURCE);
Mittelpunkt = SOURCEXSPACING*(1+SOURCEXSI
lc = 0.8;
M = zeros(1,SOURCEXSIZE);
h = zeros(1,SOURCEXSIZE);
for i = 1:SOURCEXSIZE
M(i) = SOURCEXSPACING*i - Mittelpunkt;
h(i) = 1/(0.4697*lc)*exp(-pi*(M(i))/(0.4697*lc))^2
end
h = h ./ max(h);
w = conv(h',SOURCE,"same");
w = w/100;
r = SOURCE - w;
RESULT = r
Code anwenden
Operator anwenden?
OK
Abbrechen

Anlagen, Teil 2

Ergebnisse der Ausreißer Entfernung:







Anlagen, Teil 3

Code zum Entfernen von Ausreißern:

```
clear ,clc;

A = load("200x200.txt");
SOURCE = readmatrix('200x200(z).txt');
z = A(:,3);
[SOURCEXSIZE, SOURCEYSIZE] = size(SOURCE);
h = histogram(z,'BinLimits',[min(z),max(z)],Normalization="probability");
L = h.BinWidth;
Haeufigkeit = h.Values;

Trennlinie = find(Haeufigkeit == min(Haeufigkeit));
if (min(Haeufigkeit) == 0)
    Trennlinie = Trennlinie(end)+1;
end
Trennpunkt = h.BinEdges(Trennlinie)+L/2;
NormaleTopografie = z(z <= Trennpunkt);
Ausreisser = z(z > Trennpunkt);
Abstand = mean(Ausreisser) - mean(NormaleTopografie);
Grenzwert = Abstand - 4*nanstd(z)

    B = unique(SOURCE);
    B(isnan(B)) =[];
    B = B(end:-1:1);
    n = length(B);

 for i=1:n
      MAX =B(i);
      [r,c]=find(SOURCE==MAX);
      gleicheMAX = length(r);
      for j = 1:gleicheMAX
          rj = r(j);
          cj = c(j);

           if(rj == 1 && cj == 1)
            Umliegendwert = SOURCE(1:2,1:2);

           elseif(rj == 1 && cj == SOURCEYSIZE)
             Umliegendwert = SOURCE(1:2,SOURCEYSIZE-1:SOURCEYSIZE);

           elseif(rj == SOURCEXSIZE && cj == 1)
             Umliegendwert = SOURCE(SOURCEXSIZE-1:SOURCEXSIZE,1:2);

           elseif(rj == SOURCEXSIZE && cj == SOURCEYSIZE)
             Umliegendwert = SOURCE(SOURCEXSIZE-1:SOURCEXSIZE,SOURCEYSIZE-1:SOURCEYSIZE);

           elseif(rj == 1)
               Umliegendwert = SOURCE(1:rj+1,cj-1:cj+1);
```

```
            elseif(rj == SOURCEXSIZE)
                Umliegendwert = SOURCE(rj-1:SOURCEXSIZE,cj-1:cj+1);

            elseif(cj ==1)
                Umliegendwert = SOURCE(rj-1:rj+1,1:cj+1);

            elseif(cj ==SOURCEYSIZE)
                Umliegendwert = SOURCE(rj-1:rj+1,cj-1:SOURCEYSIZE);

            else
             Umliegendwert = SOURCE(rj-1:rj+1,cj-1:cj+1);
            end

             Umliegendwert(isnan(Umliegendwert))=0;
             DeltaZ = MAX - Umliegendwert;

             if (DeltaZ(DeltaZ > Grenzwert))
                 SOURCE(rj,cj) = nan;
             end
      end
 end

x = A(:,1);
y = A(:,2);
m = length(SOURCE);

[xq,yq] = meshgrid(linspace(min(x),max(x),m),linspace(min(y),max(y),m));
[X,Y,Z] = griddata(x,y,z,xq,yq);

subplot(121);surf(X,Y,Z);
subplot(122);surf(X,Y,SOURCE);
```

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida,12.10.2022

Xiangyu Chen