

## To-Do's

■ verweis auf die beschreibungen der Werte . . . . .	3
■ erkläre konfusionsmatrix . . . . .	3
■ verknüpfung zu unten oder darüber die definitionsparameter erwähnen . . . . .	3
■ - Aufbau - Änderungen - Testen als Verifikation der Verbesserungen - Umformulierung zu einem Klassifikationsproblem - Vergleich der Realdaten mit den Kartendaten - Messmetriken für den Algorithmus -> Komplete rework . . . . .	6
■ Ergebnisse sollen in ein File geschrieben werden . . . . .	6
■ Jeder Testfall defineirt iene testregion und ihre paramter oben reinschreiben . . . . .	6
■ tester.define_test_group("Wetterstein", automated_tests_datapeaks_Wettersteinarea.json", automated_tests_resultsWetterstein", geodata_file=ttest-dataWetterstein.tif", test_type=Test_Type.DEFAULT) . . . . .	6



## Overkomplex Title

**T2000**

des Studiengangs „Informatik“

an der Dualen Hochschule Baden-Württemberg, Stuttgart

von

**Max Musterman**

Abgabedatum

8. September 2025

<b>Bearbeitungszeitraum</b>	3 Monate
<b>Matrikelnummer</b>	5321326
<b>Kurs</b>	INF23A
<b>Ausbildungsfirma</b>	TRUMPF SE + Co. KG, Ditzingen
<b>Betreuer</b>	Betreuer
<b>Zweitbetreuer</b>	

# Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit mit dem Titel „Overkomplex Title“ selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

---

Ort

---

Datum

---

Unterschrift

(Max Musterman)

# Inhaltsverzeichnis

<b>Abkürzungsverzeichnis</b>	<b>II</b>
<b>Abbildungsverzeichnis</b>	<b>III</b>
<b>Quellcodeverzeichnis</b>	<b>IV</b>
0.1 Aktuelles Testkonzept . . . . .	1
<b>1 Bewertung des Gipfelfindungsalgorithmus</b>	<b>2</b>
1.1 Motivation . . . . .	2
1.2 Neues Testkonzept . . . . .	2
1.3 chat gpts . . . . .	4

## Abkürzungsverzeichnis

## Abbildungsverzeichnis

# Quellcodeverzeichnis

Dies ist ein Test [mustermann2024].

## 0.1 Aktuelles Testkonzept

# 1 Bewertung des Gipfelfindungsalgorithmus

In diesem Kapitel wird die eine Methode zur weitgehend automatisierten Bewertung des Gipfelfindungsalgorithmus vorgestellt. Es wird gezeigt, wie die Methode funktioniert und welche Vorteile sie bietet. Außerdem werden die Ergebnisse der Bewertung präsentiert und diskutiert.

## 1.1 Motivation

Damit die Optimierungen des Gipfelfindungsalgorithmus tatsächlich zu einer Verbesserung der Laufzeit und Genauigkeit führen, muss zuerst einmal eine Möglichkeit zur Bewertung des Algorithmus geschaffen werden. Es muss eine Methode gefunden werden, um die Ergebnisse des Algorithmus mit den tatsächlichen Gipfeln zu vergleichen und die Genauigkeit zu messen. Außerdem muss eine Möglichkeit gefunden werden, um die Laufzeit des Algorithmus zu messen und zu vergleichen. Dabei ist es wichtig, dass die Bewertung möglichst automatisiert abläuft, um eine große Anzahl von Testfällen schnell und effizient bewerten zu können.

## 1.2 Neues Testkonzept

Im Gegensatz zum alten Testkonzept, bei dem die Ergebnisse des Algorithmus manuell mit den tatsächlichen Gipfeln verglichen wurden, soll nun ein neues Testkonzept entwickelt werden, das eine weitgehend automatisierte Bewertung des Algorithmus ermöglicht. Das Konzept umfasst die automatisierte Ausführung des Algorithmus mit vordefinierten Parametern, die automatisierte Erfassung der Ergebnisse und deren systematischen Vergleich mit den tatsächlichen Gipfeln sowie die automatisierte Berechnung von Metriken zur Bewertung der Genauigkeit und der Laufzeit. Zu diesem Zweck soll ein Python Skript geschrieben werden, dass die obendefinierten Schritte durchführt. Das Skript soll so gestaltet sein, dass es einfach zu bedienen ist und eine große Anzahl von Testfällen schnell und effizient definiert werden können. Dies soll mit der Hilfe einer Konfigurationsdatei

erreicht werden, in der die verschiedenen Testfälle definiert werden können. In dieser Datei soll auch definiert werden, welche Testfälle bei der Ausführung des Testers gestartet werden sollen.

Die Definition der Tests soll die Parameter Name, Input-Datei, Output-Datei, Geodaten-Datei, die Algorithmus-Parameter und den Vergleichsbereich beinhalten. Dabei dient der Name als Identifier für den Testfall. Dadurch kann der Testfall nachdem er definiert wurde einfach mit einer weiteren Funktion gestartet werden. Die Input-Datei soll die realen Gipfeldataen enthalten. Die Gipfeldataen beinhalten die Namen, Koordinaten, Höhe, Prominenz, Dominanz zu jedem einzelnen Gipfel in der Testregion. Diese Daten werden später als Ground Truth für die Bewertung des Algorithmus verwendet. Die Output-Datei ist die Datei in die die Testergebnisse geschrieben werden sollen. Die Geodaten-Datei enthält die Kartendaten der Testregion im Geotiff Format. Diese Daten werden dem Algorithmus später als Input übergeben. Die Algorithmus-Parameter beinhalten die Parameter, die bei der Ausführung des Algorithmus verwendet werden sollen. Diese sind die minimale Dominanz, minimale Prominenze, minimale Höhe, minimale orographische Dominanz sowie die Randbreite. Der Vergleichsbereich definiert den Bereich um die tatsächlichen Gipfel, in dem die vom Algorithmus gefundenen Gipfel den tatsächlichen Gipfeln zugeordnet werden. Bei dem Testablauf soll zuerst der Algorithmus mit den definierten Parametern ausgeführt werden. Danach sollen die Ergebnisse des Algorithmus validiert werden. Dabei wird das Problem der Gipfelbestimmung als Klassifikationsproblem neu formuliert. Da der Algorithmus alle Gipfel auf der Karte bestimmt und zurückgeben soll kann er als Klassifikator betrachtet werden, der alle Punkte in die zwei Klassen: Gipfel (Positiv) und kein Gipfel (Negativ) aufteilt. Um die Genauigkeit eines Klassifikators zu überprüfen kann eine Konfusionsmatrix verwendet werden. Die richtige Klassifikation für die Konfusionsmatrix liefert die Input-Datei. In ihr werden alle realen Gipfel auf dem Kartenausschnitt definiert. Da die bestimmten Gipfel nie direkt auf den realen Gipfeln liegen können (da die Kartendaten pixelbasiert und somit eine Approximation der realen Welt sind) soll ein Vergleichsbereich definiert werden, in dem die vom Algorithmus gefundenen Gipfel den tatsächlichen Gipfeln zugeordnet werden. Dadurch kann bestimmt werden, ob ein Gipfel als solcher erkannt wird oder nicht. Dabei gilt, dass es für jeden realen Gipfel nur einen gefundenen Gipfel geben darf. Daraus folgt, dass es im Vergleichsbereich um einen Gipfel

verweis auf die  
beschreibung  
der Werte

erkläre konfu-  
sionsmatrix

verknüpfung  
zu unten oder  
darüber die  
definitionspa-  
rameter er-  
wähnen

nicht möglich sein darf bei korrekter Funktion des Algorithmus zwei Gipfel zu finden. Um dies sicher zu stellen muss entweder die minimale Dominanz, die dem Algorithmus übergeben wird größer als der Vergleichsbereich sein oder der Kartenausschnitt so gewählt werden, dass das Auftreten eines solchen Falles unmöglich ist. In diesem Falle wird - da es einfacher ist - der Vergleichsbereich immer kleiner als die minimale Dominanz gewählt. Daraus ergeben sich nun die Gipfel aus der Input-Datei als wahre Positive und die Gipfel, die in der Inputdatei nicht definiert wurden als wahre Negative Werte. Die vorausgesagten Positiven Werte sind die Gipfel, die der Algorithmus bestimmt hat und die vorausgesagten Negativen Werte sind die Werte, die der Algorithmus nicht als erkannt hat.

### 1.3 chat gpts

Die Definition eines Testfalls umfasst die Parameter *Name*, *Input-Datei*, *Output-Datei*, *Geodaten-Datei*, *Algorithmus-Parameter* sowie den *Vergleichsbereich*. Der Name dient als eindeutiger Identifier des Testfalls. Nach der Definition kann der Testfall über eine separate Funktion ausgeführt werden.

Die Input-Datei enthält reale Gipfeldata der Testregion. Diese umfassen für jeden Gipfel den Namen, die Koordinaten, die Höhe, die Prominenz sowie die Dominanz. Diese Daten dienen als Ground Truth zur Bewertung des Algorithmus.

Die Output-Datei legt fest, in welche Datei die Testergebnisse geschrieben werden.

Die Geodaten-Datei enthält die Kartendaten der Testregion im GeoTIFF-Format. Diese Daten werden dem Algorithmus als Eingabe übergeben.

Die Algorithmus-Parameter definieren die Bedingungen für die Ausführung. Dazu gehören die minimale Dominanz, die minimale Prominenz, die minimale Höhe, die minimale orographische Dominanz sowie die Randbreite. Diese Parameter beeinflussen, welche Erhebungen als Gipfel klassifiziert werden.

Der Vergleichsbereich definiert einen räumlichen Umkreis um jeden realen Gipfel. Innerhalb dieses Bereichs werden vom Algorithmus erkannte Gipfel einem tatsächlichen Gipfel zugeordnet.

Der Testablauf besteht aus zwei Schritten. Zunächst wird der Algorithmus mit den definierten Parametern auf den Geodaten ausgeführt. Anschließend werden die Ergebnisse validiert.

Zur Validierung wird das Problem der Gipfelbestimmung als Klassifikationsproblem formuliert. Der Algorithmus bestimmt alle Gipfel innerhalb des Kartenausschnitts und kann daher als binärer Klassifikator betrachtet werden. Er teilt alle Punkte in zwei Klassen ein: *Gipfel* (positiv) und *kein Gipfel* (negativ).

Zur Bewertung eines Klassifikators wird eine Konfusionsmatrix verwendet. Eine Konfusionsmatrix stellt die Anzahl korrekt und falsch klassifizierter Fälle gegenüber. Sie besteht aus vier Kategorien: Wahre Positive (True Positives), Falsche Positive (False Positives), Wahre Negative (True Negatives) und Falsche Negative (False Negatives).

Die korrekte Klassifikation ergibt sich aus der Input-Datei, in der alle realen Gipfel des Kartenausschnitts definiert sind. Da die Kartendaten pixelbasiert sind und somit eine Approximation der realen Welt darstellen, können erkannte Gipfel nicht exakt auf den realen Koordinaten liegen. Daher wird der Vergleichsbereich eingeführt. Befindet sich ein vom Algorithmus erkannter Gipfel innerhalb dieses Bereichs um einen realen Gipfel, wird er diesem zugeordnet.

Für jeden realen Gipfel darf höchstens ein erkannter Gipfel zugeordnet werden. Innerhalb eines Vergleichsbereichs darf daher kein zweiter Gipfel liegen. Um dies sicherzustellen, muss die minimale Dominanz größer sein als der Vergleichsbereich oder der Kartenausschnitt so gewählt werden, dass ein solcher Fall ausgeschlossen ist. In diesem Testsystem wird der Vergleichsbereich stets kleiner als die minimale Dominanz gewählt.

Auf dieser Grundlage gelten die in der Input-Datei definierten Gipfel als tatsächliche positive Fälle. Alle nicht in der Input-Datei definierten Punkte gelten als tatsächliche negative Fälle. Die vom Algorithmus erkannten Gipfel bilden die vorhergesagten positiven Fälle. Nicht erkannte Punkte bilden die vorhergesagten negativen Fälle.

Eine beispielhafte Konfusionsmatrix kann wie folgt dargestellt werden:

	Vorhergesagt: Gipfel	Vorhergesagt: Kein Gipfel
Tatsächlich: Gipfel	42	8
Tatsächlich: Kein Gipfel	5	945

In diesem Beispiel wurden 42 reale Gipfel korrekt erkannt (Wahre Positive). Acht reale Gipfel wurden nicht erkannt (Falsche Negative). Fünf Punkte wurden fälschlicherweise als Gipfel klassifiziert (Falsche Positive). 945 Punkte wurden korrekt als Nicht-Gipfel klassifiziert (Wahre Negative).

Auf Basis dieser Matrix können verschiedene Qualitätsmaße berechnet werden, beispielsweise die Genauigkeit (Accuracy), die Präzision (Precision) oder die Sensitivität (Recall). Die Konfusionsmatrix bildet somit die Grundlage für eine quantitative Bewertung der Algorithmusleistung.

- Definition der konkreten Testszenarien - Grund für deren Wahl - Schwierigkeiten, die überprüft werden sollen - Durchlauf der Tests
- Objekt orientierung? - Optimierung der Programmstruktur - Abbruch nicht bei erreichen des nächsten gipfels - Die Verbesserung der Distanzberechnungen

Das Randwert problem als  $\min\_prominence < x$  und  $\min\_dominanz < y$

Bewertung des neuen Algorithmus

- Aufbau - Änderungen  
- Testen als Verifikation der Verbesserungen - Umformulierung zu einem Klassifikationsproblem - Vergleich der Realdaten mit den Kartendaten - Messmetriken für den Algorithmus -> Komplette rework

Ergebnisse sollen in ein File geschrieben werden

Jeder Testfall definiert eine testregion und ihre parameter oben reinschreiben

```
tester.define_test_group("Wetterstein",
    automated_tests__datapeaks_Wettersteinarea.json",
    automated_tests__resultsWetterstein",
    geodata_file=ttest-dataWetterstein.tif",
    test_type=Test_Type.DEFAULT)
```