To-Do's

TRUMPF

DHBW
Duale Hochschule
Baden-Württemberg

Overkomplex Title

T2000

des Studiengangs „Informatik“

an der Dualen Hochschule Baden-Württemberg, Stuttgart

von

Max Musterman

Abgabedatum

8. September 2025

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

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

Abkürzungsverzeichnis

Abbildungsverzeichnis

Quellcodeverzeichnis

Dies ist ein Test [**mustermann2024**].

0.1 Aktuelles Testkonzept

1 Bewertung des Gipfelfindungsalgorithmus

In diesem Kapitel wird der 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. Ein testfall

ist eine testweise Ausführung des Algorithmus auf einem Kartenstück mit vordefinierten Parametern. 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 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. Dadurch kann der Testfall nachdem er definiert wurde mit einer weiteren Funktion gestartet werdenDie Input-Datei enthält reale Gipfeldaten 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.

Testfälle können auch gebündelt als Gruppe initialisiert sowie ausgeführt werden. Hierfür kann beim Erstellen jedem Testfall ein Gruppenname hinzugefügt werden. Ganze Testfallgruppen lassen sich anschließend über eine eigene Funktion definieren. Diese Funktion verwendet außer des Testgruppennamen die gleichen Parameter wie die Testfall Generierung.

Testgruppen erklären als gleiche tests mit unterschiedlichen Vergleichsbereichen

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).

Für die Analyse mittels Konfusionsmatrix werden ausschließlich diejenigen Punkte betrachtet, die entweder in der Input-Datei als reale Gipfel definiert sind oder vom Algorithmus als Gipfel ausgegeben werden. Es werden somit nicht alle einzelnen Pixel bzw. Geländepunkte der GeoTIFF-Datei als Klassifikationsobjekte herangezogen. Stattdessen beschränkt sich die Bewertung auf die Menge der tatsächlich existierenden Gipfel (Ground Truth) sowie auf die vom Algorithmus detektierten Gipfel. Dadurch wird die Analyse auf die für die Aufgabenstellung relevanten diskreten Objekte fokussiert und eine Verzerrung durch die sehr große Anzahl an Nicht-Gipfel-Pixeln vermieden. Zudem sind Puntke die keine Gipfel sind und vom Algorithmus nicht als solche erkannt werden für die aus den Konfusionsmatrix bestimmten Metriken nicht relevant ().

verweis auf formeln Precision und Recall

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.

Im Folgenden wäre hier ein Beispiel für einen Solchen Testfall im Wetterstein Gebirge.

	Vorhergesagt: Gipfel	**Vorhergesagt: Kein Gipfel**
Tatsächlich: Gipfel	18	4
Tatsächlich: Kein Gipfel	10	0

In diesem Beispiel wurden 18 reale Gipfel korrekt erkannt (Wahre Positive). Diese Gipfel sind in der Input-Datei und der Vorhersage des Algorithmus enthalten. Vier Gipfel aus der Input-Datei waren nicht in der Ausgabe des Algorithmus enhalten, was bedeutet, dass der Algoprithmus diese nicht erkannt hat (Falsche Negative). Zehn Gipfel wurden fälschlicherweise als Gipfel klassifiziert (Falsche Positive). Null Gipfel wurden von Algorithmus sowie Input-Datei nicht als Gipfel klassifiziert, da diese immer ignoriert werden.

Auf Basis dieser Matrix können verschiedene Qualitätsmaße berechnet werden. Bei diesem Tester werden die Präzision (Precision) und die Sensitivität (Recall), sowie der F-Score (f1-Score) verwendet. Die können mit den folgenden Formeln berechnet werden:

$$\text{Precision} = \frac{\text{True Positiv (TP)}}{\text{True Positiv (TP)} + \text{False Positiv (FP)}} \tag{1.1}$$

$$\text{Recall} = \frac{\text{True Positiv (TP)}}{\text{True Positiv (TP)} + \text{False Negativ (FN)}} \tag{1.2}$$

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{1.3}$$

maybe in Theorieteil

Diese Metriken geben Aufschluss über die Genauigkeit des Algorithmus und zeigen wo der Algorithmus genauer ist und wo schlechter . Dabei zeigt die Precision wie sehr der Algorithmus zum finden neuer nicht echter Gipfel neigt und der Recall zeigt wie häufig der Algorithmus reale Gipfel nicht erkennt. Beide Werte sind ≥ 0 und ≤ 1. Der F-Score verbindet beide Werte und zeigt wie genau der Algorithmus allgemein ist.

neu formulieren den satz

Dabei gilt desto näher die Werte an eins sind, desto besser ist der Algorithmus. Am Wichtigsten ist, dass der Recall möglichst nahe an der eins ist, da es für einen Suchalgorithmus besser ist unötige Werte zu finden als Werte zu übersehen.

dabei austauschen

Andere Metriken zur Bestimmung der Genauigkeit sind die durchschittlichen Abweichungen der Position, Höhe, Prominenz und Dominanz pro Testfall. Diese Werte zeigen die Genauigkeit des Algorithmus für die vom Algorithmus gefundenen realen Gipfel. Sie werden in Metern angegeben. Zusätzlich wird bei der Positionsabweichung eine Abweichung in X- und Y-Richtung bestimmt. Alle Metriken werrden in die im Testfall definierte Output-Datei geschrieben. Abgesehen von diesen Metriken werden zudem die Zuordnungen der Gipfel und die Einzelabweichungen pro Gipfel in die Output-Datei geschrieben.

Außerdem sollen Metriken implementiert werden, die die Laufzeit des Algorithmus aufzeigen und verdeutlichen, wo der Algorithmus längere Zeit benötigt als erwartet. Für diesen Zweck soll eine Funktion implementiert werden, die die gesammte Ausführtzeit sowie die einzelnen Ausführzeiten der Teilbereiche (globale Maximasuche, Prominenzberechnung, Dominanzberechnung) zurückgibt.

Zusammengefasst kann gesagt werden, dass das neue Testkonzept, bis auf die Testfalldefinition, automatisiert abläuft und dadurch, dass die Suche als Klassifikationsproblem neu formuliert wird die Genauigkeit des Algorithmus bestimmt werden kann. Außerdem liefert das neue Testkonzept eine zeitliche Abschätzung der Leitung der einzelnen Teiile des Algorithmus.

besserer Satzbnau (verstehe ihn selbst nicht mal richtig)

1.3 Umsetzung

Wie bereits in 1.2 definiert wird der Tester mit einer Konfigdatei und mehreren Pythondateien umgesetzt. Zusätzlich werden die Realen Gipfel für jeden Testfall in einer seperaten Json-Datei definiert und die Ergebnisse in selbigen Format gespeichert.

sicherstellen, dass das in NNeues Testkonzeptßteht

Der Tester besteht aus zwei Klassen. Sie heißen Test_Runner und Test_Controler.

Diagramm uml of test_controler

Der Test_Controler ist für die Haushaltung und initialisierung der Tests zuständig. Er führt außerdem die Tests der Laufzeitdauer durch, da diese wenig Programmcode zur Ausführung benötigen und eine andere Ausführlogik sowie Metriken wie die Genauigkeitstests besitzen.

beschreibung definition der Funktion des Test_Controler

Diagramm uml of test_runner

Der Test_Runner ist für die Ausführung der Genauigkeitstests zuständig.

bescchreibe Test_runner

1.4 Testen des Algorithmus

In diesem Unterkapitel werden die Test-Szenarien auf die der Tester getestet wird beschreiben. Außerdem wird die Beschaffung der Testdaten erläutert und die Testergebnisse bewertet.

1.4.1 Definition der Test-Szenarien

Um die korrekte Funktion des Algorithmus zu prüfen wird auf verschiedenen Kartenausschnitten getestet. Dazu wurden drei Kartenauschnitte ausgewählt, die verscheidenen Fälle abbilden in welchen der Algorithmus eingesetzt werden kann.

bild Wetterstein

Der erste Kartenausschnitt konzentriert sich auf das Wettersteingebirge (Alpen) bei Gamisch-Partenkirchen und große Teile des Karwendelgebirges nördlich von Innsbruck (Österreich), sowie viele umliegende Berge. Dieser Kartenausschnitt wurde ausgewählt, da er viele Bergrücken und Grate beinhaltet, die in diverse Richtungen abfallen und denoch weitgehend freistehende Berge wie die Große Arnspitze oder den Vorderskopf beinhaltet. Dieser Kartenabschnitt bietet zudem vielfältige Berghöhen und gleichzeitig viele Gipfel, die Höhentechnisch nahe bei einander sind.

umformulieren

bild Feldberg Kartenausschnitt

Der zweite Kartenausschnitt zentriert sich um den höchsten Berg des Schwarzwaldes, den Feldberg. Dieser Bereich wurde gewählt, da der Algorithmus bis jetzt weniger in Mittelgebirgen getestet wurde. In Mittelgebirgen sind die Hänge meist weniger steil und die Gipfel weniger eindeutig. Die Whal fiel auf die Region um den Feldberg, da diese Region gut kartographiert ist und daher die Gipfel schon genau bestimmt sind. Außerdem gleicht der Feldberg auf seinem Gipfel mehr einer Hochebene als einem herkömlichen Gipfel. Dieses Terrain ist daher ein gutes Gegenstück zu den schroffen und unebenen nördlichen Kalkalpen.

bild Süd Schwarzwald

Der letzte Kartenausschnitt ist ein großer Kartenausschnitt des Süd Schwarzwalds. Die Wahl fällt auf den Südschwarzwald, da er sehr vielseitig ist. Er fällt in richtung Süden und Westen steil ins Rheintal ab und wandelt sich in den Osten in eine Hügellandschaft. Dieser Test soll, durch das einschließen des Rheintals bei Schaffhausen außerdem zeigen, wie sich der Algorithmus auf weniger gut karthographiertem Gebeit bedienen lässt und

ob er Erhebungen finden, die nicht in den Datenquellen enthalten sind.

Es wurde kein Abschnitt in Schweizer Alpen gewählt, da der Algorithmus dort schon mehr getestet wurde.

Inhalt: - Definition der konkreten Testszenarien - Grund für deren Wahl - Daten herbekommen? - Schwierigkeiten, die überprüft werden sollen - Durchlauf der Tests

- Objekt orientierung? - Optimierung der Programstruktur - 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 Allgorithmus