To-Do's

Inhalt: - Definition der konkreten Testszenarien - Grund für deren Wahl - Daten herbekommen? - Schwierigkeiten, die überprüft werden sollen - Durchlauf der Tests 13

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.

1.3.1 Test_Controler

Diagramm uml of test_controler

Die Klasse `Test_Controler` ist die zentrale Steuerungseinheit für die automatisierten Tests. Sie ist dafür verantwortlich, Testkonfigurationen zu definieren, zu verwalten und auszuführen.

Testdefinition Die Methode `define_test()` ermöglicht die Definition eines einzelnen Testlaufs. Dabei werden Parameter wie der Name des Tests, die zu verwendenden Geodaten, die Ausgabedatei für die Ergebnisse sowie die Schwellenwerte für die Gipfelerkennung (Prominenz, Dominanz, etc.) festgelegt. Diese Konfiguration wird als `Test_Config`-Objekt gespeichert.

Für die Durchführung von Testreihen existiert die Methode `define_test_group()`. Diese Methode generiert eine Gruppe von Tests, die sich in der Regel nur durch einen variierenden Parameter unterscheiden, wie beispielsweise dem *Matching-Radius*. Dies vereinfacht das Testen von verschiedenen Konfigurationen für ein bestimmtes Szenario.

Testausführung

Die Ausführung der Tests wird über die Methoden `run_test()` und `run_test_group()` gesteuert. `run_test()` kann entweder einen einzelnen, spezifischen Test anhand seines Namens oder alle definierten Tests ausführen. `run_test_group()` führt alle Tests aus, die zu einer bestimmten Gruppe gehören.

Intern verwaltet der `Test_Controler` eine Liste von `Test_Runner`-Instanzen. Für jede Testkonfiguration wird geprüft, ob bereits ein kompatibler `Test_Runner` (für die gleiche Geodaten-Datei oder den gleichen Ordner) existiert. Ist dies der Fall, wird dieser wiederverwendet, um das wiederholte Laden und Vorverarbeiten der Geodaten zu vermeiden. Andernfalls wird eine neue `Test_Runner`-Instanz erzeugt.

ist das wirklich so?

Zeitmessung und Parametertests

Zusätzlich zu den funktionalen Tests bietet der `Test_Controler` Methoden zur Zeitmessung (`timetest_single_file()` und `timetest_multiple_file()`). Diese führen die Gipfelanalyse für eine einzelne Datei oder mehrere Dateien durch und messen die benötigte Zeit für die einzelnen Teilschritte des Algorithmus, wie die Suche nach lokalen Maxima, die Prominenz- und die Dominanzberechnung. Die Methode `multiple_execution_test()` ermöglicht es, die Analyse mehrfach hintereinander auszuführen, um die Stabilität und Konsistenz der Laufzeit zu bewerten. Diese Logik wurde hier im `Test_Controler` implementiert, da diese Tests wenig Programmcode zur Ausführung benötigen und eine andere

Ausführlogik sowie Metriken wie die Genauigkeitstests besitzen.

Diagramm uml of test_runner

1.3.2 Test_Runner

Der `Test_Runner` ist für die Durchführung und Auswertung der Genauigkeitstests zuständig. Jede Instanz dieser Klasse ist an ein bestimmtes Set von Geodaten gebunden, welche sie bei der Initialisierung lädt und für alle nachfolgenden Tests wiederverwendet.

Initialisierung und Datenvorbereitung

Beim Erstellen eines `Test_Runner`-Objekts werden die digitalen Höhenmodelldaten (DEM) eingelesen und vorverarbeitet. Wichtige Metadaten wie das Koordinatenreferenzsystem, die Auflösung und die Umrechnungsfaktoren zwischen Pixel und Meter werden extrahiert und für spätere Berechnungen zwischengespeichert. Dieser Schritt ist ressourcenintensiv, weshalb der `Test_Controler` darauf achtet, `Test_Runner`-Instanzen für bereits geladene Geodaten wiederzuverwenden.

ist das wirklich so?

Testdurchführung

Die zentrale Methode ist `execute_test()`, die eine spezifische Testkonfiguration entgegennimmt. Der Ablauf eines Tests ist wie folgt:

1. **Einlesen der Testdaten:** Die realen Gipfel („True Peaks“) und die bekannten Nicht-Gipfel („Fake Peaks“) werden aus der angegebenen JSON-Datei geladen.
2. **Gipfelsuche:** Der Gipfelfindungsalgorithmus wird mit den in der Konfiguration definierten Schwellenwerten (z.B. für Prominenz und Dominanz) auf den Geodaten ausgeführt.
3. **Normalisierung:** Die gefundenen Gipfel werden in ein einheitliches Datenformat überführt, um den Vergleich mit den realen Gipfeln zu ermöglichen.
4. **Matching:** Die gefundenen Gipfel werden mit den realen Gipfeln abgeglichen. Ein gefundener Gipfel gilt als TTreffer"(*True Positive*), wenn er sich innerhalb eines definierten Radius (*Matching-Area*) um einen realen Gipfel befindet. Gefundene Gipfel, die keinem realen Gipfel zugeordnet werden können, gelten als *False Positives*. Reale

Gipfel, die nicht gefunden wurden, sind *False Negatives*.

5. **Auswertung und Statistik:** Aus den Ergebnissen des Matchings werden Metriken wie Präzision (*Precision*), Sensitivität (*Recall*) und der F1-Score berechnet. Zusätzlich werden die durchschnittlichen Abweichungen für Position, Höhe, Prominenz und Dominanz ermittelt.

6. **Ausgabe und Export:** Die berechneten Statistiken und Ergebnisse werden in der Konsole ausgegeben und zur späteren Analyse in einer JSON-Datei gespeichert.

maybe hier diagramm über genaue ausführreichenfolge und datenübergabelogik und dependencies?

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 Südschwarzwalds. 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 Gebiet bedienen lässt, wo nicht alle Gipfel in der Datenquelle zwingend bekannt sind.

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

zitat zu Tester-Studienarbeit

Die Vergleichsbereich werden bei den Tests jeweils jeweils einmal als $\sqrt{2}$ * *Pixelgröße*, *100m* und *minimale Dominanz* gewählt. Das sorgt dafür, dass schnell erkennbar ist welche Gipfel auf welcher Genauigkeit erkannt wurden. Dabei beschreibt $\sqrt{2}$ * *Pixelgröße* die maximale Genauigkeit mit der eine pixelbassierte Karte einen Gipfel voraussagen kann. Die *100m* sind der Bereich, den wir als gewünschte Genauigkeit des Algorithmus definiert haben. Die *minimale Dominanz* beschreibt nach 1.2 den maximalen Bereich in dem wir die realen Gipfel den gefundenen zuordnen können.

1.4.2 Beschaffung der Testdaten

Die Testdaten wurden aus verschiedenen Quellen beschafft. Die Geodaten (GeoTIFF-Dateien) wurden von verschiedenen Quellen beschafft. Für die Alpen wurden die Daten des Copernicus Programms der EU bezogen. Diese Entscheidung wurde Aufgrund der Ländergrenze zwischen Deutschland und Österreich, welche sich in dem Kartenausschnitt befindet, getätigt. Als Kartenmodell wurde hier die „Digital Elevation Model (DEM) for Europe at 30 meter“ gewählt. Aus ihr können Kartenausschnitte von Europa im mit einer Auflösung von 30m pro Pixel heruntergeladen werden.

abkürzung

Für den Schwarzwald wurden die Daten von der Landesvermessung Baden-Württemberg

bezogen. Der Grund hierfür ist, dass die Daten dort genauer sind. Alle Daten wurden in einer Auflösung von 0.25m pro Pixel bezogen. Da diese Auflösung zu hoch für die Testzwecke ist, wurden die Daten auf eine Auflösung von 10m pro Pixel heruntergerechnet.

Die Daten über die realen Gipfel wurden aus verschiedenen Quellen bezogen. Da in den Daten nicht flächendeckend die Prominenz und Dominaz definiert wruden, mussten bei der Erstellung der Realdatensätze verschiedene API-Quellen und manuelle Suche kombiniert werden. Zum einen wurden die Daten von der OpenStreetMap API bezogen. Diese Daten wurden mit Hilfe der Overpass API abgefragt. Es wurden alle Punkte mit dem Tag *natural=peak* innerhalb der Kartenausschnitte abgefragt. Diese Daten wurden wurden zunächst mit den Daten von Wikidata abgeglichen und durch diese ergänzt. Wikidata bietet eine API an, über die für bestimmte Koordinaten die Gipfel abgefragt werden können. Mit diesem Abgleich konnten über mehrere Kartenbereichen in den Alpen im Durchschitt 80% der Gipfel mit Prominenz über 100m gefunden werden. Dies leigt nicht daran, dass die Gipfel selbst unbekannt sind, sondern daran, dass die Prominenz und Dominanz nicht in den Datenquellen definiert wurden. Daher wurden die Daten von Hand ergänzt und auf Vollständigkeit überprüft. Zum besseren Analyseren der Tests wurden die Gipfel, die nicht den Kriterien für einen Gipfel entsprechen, als *Fake Peaks* definiert und in die Testdaten aufgenommen. Es wurden Punkte ausgewählt, die in der Nähe von realen Gipfeln liegen, aber nicht die Kriterien für einen Gipfel erfüllen oder von den APIs als Gipfel, die den Bedingungen nicht entprechen erkannt wurden. Diese Daten helfen bei dem schnellen finden von Fehlern des Algorithmus, da sie zeigen, welchen Gipfel der Algorithmus fälschlicherweise als Gipfel identifiziert hat und nicht nur seine Koordinaten. Um die Daten manuell zu ergänzen und zu überprüfen muss die Karte von Hand durchsucht werden. Dabei stellt sich die Frage, welche Karte verwendet werden sollte, um die genausten testdaten zu erhalten. Dabei spielen Faktoren wie die allgemeine Genauigkeit der Höhenlinien und Anzahl der makierten Gipfel eine Rolle. Die Höhenlinien sind wichtig um die Prominenz abzuschätzen, wenn keine Realwerte dafür existieren. Bei wie vielen Gipfel keine Realwerte in den Datenquellen existieren hängt von vielen Faktoren ab. Dabei gilt meist, dass desto berühmter, höher die Gipfel und desto höher deren Prominenz und Dominanz ist umso wahrscheinlicher existieren Realwerte für diese Berge und umso besser sind diese. Ersteres spielt nach der Erfahrung eine größere Rolle. So ist beispielsweise die Höhe der

Zugspitze in der Wikidata Datenbank auf Centimeter genau definiert, während die Höhe der meisten anderen Gipfel nur auf Meter genau definiert ist. Es gibt aber auch Fälle, wie beispielsweise den Daniel, bei der die Höhe nur auf 10 Meter genau definiert ist, obwohl sie ein eine große Prominenz von über 500m besitzt.

Es wurden verschiedene Karten verglichen, wie die OpenStreetMap Karte, die Karte von Google Maps und die Karte von Bing Maps. Es wurde festgestellt, dass die OpenStreetMap Karte die genausten Höhenlinien und die meisten markierten Gipfel enthält. Daher wurde sie als Hauptquelle für die manuelle Ergänzung der Testdaten verwendet. Die Werte für die Prominenz und Dominanz wurden dabei durch Abschätzen der Höhenlinien auf der Karte ergänzt, wenn nicht schon durch die APIs gegeben.

1.4.3 Durchlauf der Tests

Nachdem die Testdaten erstellt wurden, wurden die Tests mit verschiedenen Konfigurationen des Algorithmus durchgeführt. Dabei wurden verschiedene Werte für die Schwellenwerte der Prominenz und Dominanz getestet, um zu sehen, wie sich diese auf die Genauigkeit der Gipfelerkennung auswirken. Dabei ist aufgefallen, dass die Datenqualität aprupt abnimmt, wenn der Schwellenwert für die Prominenz und der Dominanz zu neidrig angesetzt werden. Um einerseits den Gipfelfinder in schwerem Gebiet zu Testen und gleichzeitig die Datenqualität zu erhalten, wurde bei keinem Test der Schwellenwert für die Prominenz unter 100m und der Schwellenwert für die Dominanz unter 1000m gesetzt. Es wurden aber auch Tests mit höheren Schwellenwerten durchgeführt, um zu sehen, wie sich diese auf die Genauigkeit auswirken.

1.4.4 Bewertung der Testergebnisse

Nach dem durchlaufen der Tests wurden die Ergebnisse ananlysiert und bewertet. Dabei wurde nicht nur die Genauigkeit der Gipfelerkennung bewertet, sondern auch die durchschnittlichen und extremen Abweichungen für Position, Höhe, Prominenz und Dominanz, sowie die Laufzeit. Es wrude festgestellt, dass der Algorithmus bei den meisten Tests die Position der Gipfel mit einer hohen Genauigkeit bestimmt. Die Höhe der Gipfel wird ebenfalls mit einer hohen Genauigkeit bestimmt, wobei die durchschnittliche Abweichung

meist unter 10 Metern liegt. Diese Abweichungen lassen sich durch die Auflösung und Qualität der Geodaten, sowie Realdaten erklären.

Es gibt Fälle in welchen der Algorithmus Gipfel erkennt, die in Realität nicht existieren. Das passiert nur in der Nähe des Kartenrandes. Das liegt daran, dass kleine Erhebungen am Hang eines Berges nahe des Kartenrandes nicht als Gipfel ausgeschlossen werden können. In diesem Fall wird ihnen der Hang des gesammten Berges bei der Prominenzberechnung angerechent. Um dies zu verhindern wurde bereits die *border width* eingeführt. Diese scheint aber nicht richtig zu funktionieren, die Randwertfehler nun am Rand der *border width* auftretten.

Die Prominenz und Dominanz werden ebenfalls mit einer guten Genauigkeit bestimmt, wobei die durchschnittliche Abweichung meist unter 20 Metern liegt. Die Laufzeit des Algorithmus variiert je nach Größe des Kartenausschnitts und der Anzahl der gefundenen Gipfel, liegt aber in der Regel im Bereich von einigen Sekunden bis zu einigen Minuten.

1.5 Bewertung des Testers

- 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

diagram mit Zylinder oder berg und dann ne border durch/ höhenlinien an border

das border width problem diagramm

das stimmt nicht, werte ändern

birkkarspitzenpro wannig hat das auch

In dem alten Algorithmus wurde angenommen, dass die Prominece immer zum nächsten Gipfel geht. Das muss sie nicht. Das spiegelt der neue Algorithmus mit einer modifierten Abbruchbedingung und keinem Endpunkt wieder.

analyse auf altem project

abschließend die schwierigkeiten des algorithmus als Topografische bereiche

Inhalt: - Definition der konkreten Testszenarien - Grund für deren Wahl - Daten herbekommen? - Schwierigkei-