Intelligente Systeme
Aufgabe 2
Dokumentation/Report

Einleitung

Gegeben ist eine Datensatz (~800.000 Daten, sogenannte 'data') im .csv Format. Weiterhin eine weitere Datei mit Koordinaten von gesetzten „Labels“ auf diesen Daten.
Die gegebenen daten sollen nach den möglichen Kriterien zum Setzen von Labels analysiert und ein Algorithmus implementiert werden, der sowohl die Markierung von Labels automatisiert durchführt, ohne bestehende Labels zu kennen, als auch auf weitere unbekannte Datensätze anwendbar ist.
Weiterhin soll der implementiere Algorithmus anhand der gegebenen Testdaten validiert werden, indem der f1Score (Harmonisches Mittel aus Precision (Richtige Treffer / Alle gefundenen Treffer) und Recall (Richtige Treffer / Anzahl möglicher richtiger Treffer)) errechnet wird.

Konkrete Aufgabenstellung

1. Untersuchen Sie die gegebenen Datensätze 0 und 1 (also data0.csv und data1.csv) und versuchen Sie, zu ermitteln, nach welchen Kriterien die zugehörigen Markierungen (label0.csv und label1.csv) vergeben wurden.

2. Entwickeln Sie einen Algorithmus, welcher die Markierung der Datensätze automatisiert.

3. Evaluieren Sie Ihren Algorithmus anhand der manuellen Label und bestimmen Sie Precision, Recall und F-score Ihres Algorithmus.

4. Optimieren Sie Ihren Algorithmus anhand der Datensätze 0 und 1. Im Endergebnis muss auf Datensatz 2 ein F-score von mindestens 0,8 erreicht werden.

Lösung Aufgabe 1

Die gegeben Daten wurden grafisch in einem 3D Modell dargestellt, indem die enthaltenen Werte als Höhe interpretiert wurden (skaliert auf einen Float zwischen 0 und 1, für die graphische Darstellung). Als Nullpunkt der Höhe wird zur Darstellung der minimale enthaltene Datenwert herangezogen.

Hierdurch konnten die Daten graphisch interpretiert werden, was das Auffinden von Kriterien erleichterte.

Gefundene Kriterien sind:

Labels sind lokale Maxima von „Hügeln“ auf der Grundoberfläche

Die Grundoberfläche ist nicht eben, sondern variiert Wellenförmig in eine Richtung

Das „Hügelvolumen“ steigt mit der Höhe der Grundoberfläche

Ein Hügel kann sein Maximum unter der Höhe der Grundoberfläche an anderer Stelle haben

Lösung Aufgabe 2

Zunächst wurden die gegebenen Daten in einem 3D-Model (als Heightmap) dargestellt. Während die Farben der Vertices mit zunehmender Höhe von Blau zu Rot skalieren. Markierte Labels werden als grüne Vertices dargestellt, um sie eindeutig erkennen zu können. Hierdurch konnten wir verstehen und auch mit dem Auge erkennen, welche Kriterien ein Label definieren. Daraufhin haben wir verschiedene Filter implementiert, die die gegebenen Daten so bearbeiten, dass Punkte, die nicht den Kriterien eines Labels entsprechen entfernt, bzw. auf Grundhöhe gesetzt werden.

Der erste Filter 'KillTheNoiseFilter.class' filtert alle Werte heraus, die gleich oder niedriger dem Durchschnittswert der jeweiligen Reihe (Parallel zur Wellenoberfläche) sind. Zur Feineinstellung kann die Grenzhöhe mit einem Grenzwert, proportional zur Reihendurchschnittshöhe, angepasst werden (treshold parameter).

Dies lässt die „Bergspitzen“ stehen, während der übrigbleibende Rest auf eine einheitliche Grundhöhe „gezogen“ wird und es somit nun einfach ist, einen „Berg“ als solches zu erkennen, um sein Maximum zu suchen. Das Auffinden lokaler Maxima wird somit nun zum Auffinden des globalen Maximum pro „Berg“ vereinfacht.

Der zweite Filter 'FloodFillLocalMaximaFilter.class' iteriert den Datensatz und wendet pro Koordinate eine modifizierte prozedurale Version des FloodFill-Algorithmus an, der am aktuellen Punkt startet und den Bereich innerhalb dessen Umrandung von 0.0f-Werten auf 0.0f setzt, gleichzeitig aber die Koordinaten des zuletzt gefundenen Maximalwertes

während des Füllvorgangs speichert und zurückgibt, wenn der Algorithmus beendet wurde. Gleichzeitig wird die Anzahl der Iterationen des Algorithmus gezählt und somit die größe der Grundfläche des aktuell zu behandelnden „Bergs“ bestimmt. Diese wird während des Algorithmus dazu herangezogen, sie mit dem Durchschnittswert der auf dieser Reihe stehenden Bergbreite zu vergleichen und zu kleine, falsch positive Werte herauszufiltern. Schließlich werden alle gefundenen Maxima auf eine einheitliche Höhe gesetzt und die Koordinaten der auf der Map verbleibenden Punkte als Set von gefundenen Labels gespeichert. Um die rekursive Natur des FloodFill-Algorithmus zu brechen, wurde ein eigener Stack erstellt, mithilfe dessen der Algorithmus mittels einer Schleife Quasi-Iterativ umgesetzt werden kann, indem Nächstmögliche Punkte auf dem Stack abgelegt und pro Schleifendurchlauf entnommen werden. Durch die gleichzeitige „Planierung“ des aktuellen „Bergs“ wird vermieden, dass der Algorithmus auf dem selben „Berg“ ein zweites Mal während der Iteration ausgeführt wird. Somit kann der Filter die gesamten Daten schnell iterieren und die Maximalwerte auffinden.

Lösung Aufgabe 3 und 4

In der Main-Methode werden die vom Algorithmus gefundenen Labels mit den nicht modifizierten Sollwerten aus den Ursprungsdaten verglichen und dabei Precision, Recall und daraus resultierender f1Score ermittelt. Der Algorithmus wurde mittels setzen von Cut-Off werten und anpassung der Algorithmen optimiert. Die Aufgabenstellung (Mindestens f1Score von 0.8 im zweiten Datensatz) wurde erfüllt (Siehe „Ermittelte Werte“).

Ermittelte Werte

data0/label0:

p:110/144=0.7638888888888888
r:110/139=0.7913669064748201
f1-score:0.7773851590106007

data1/label1:

p:104/116=0.896551724137931
r:104/113=0.9203539823008849
f1-score:0.9082969432314412

data2/label2:

p:98/121=0.8099173553719008
r:98/105=0.9333333333333333
f1-score:0.8672566371681415

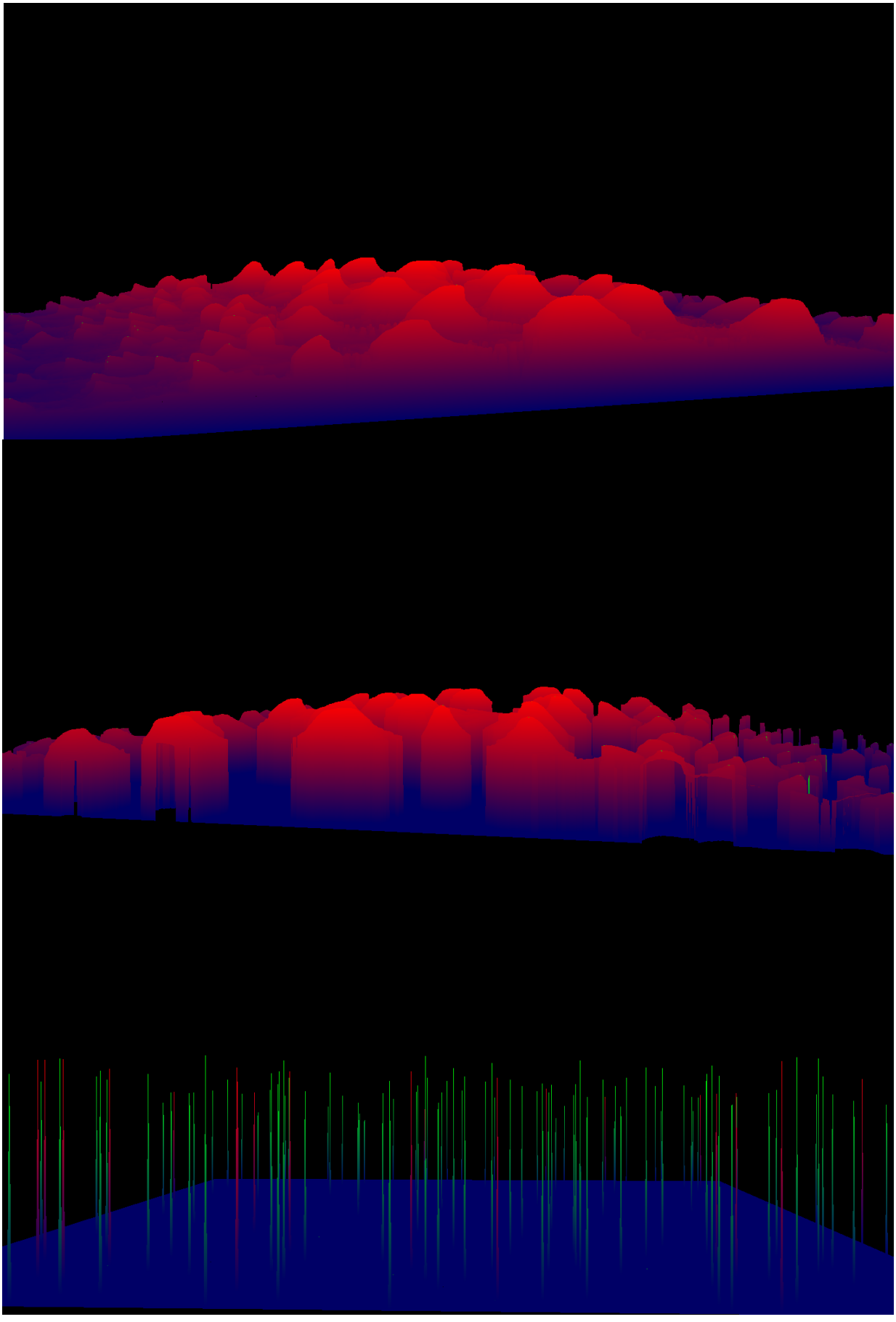
Graphische Algorithmusdarstellung

Auf der nachfolgenden Seite befindet sich die graphische Darstellung des Algorithmus für data1.csv/label1.csv in 3 Schritten (In der Abgabe unter /pics abgelegt sind alle drei T estdatensätze):

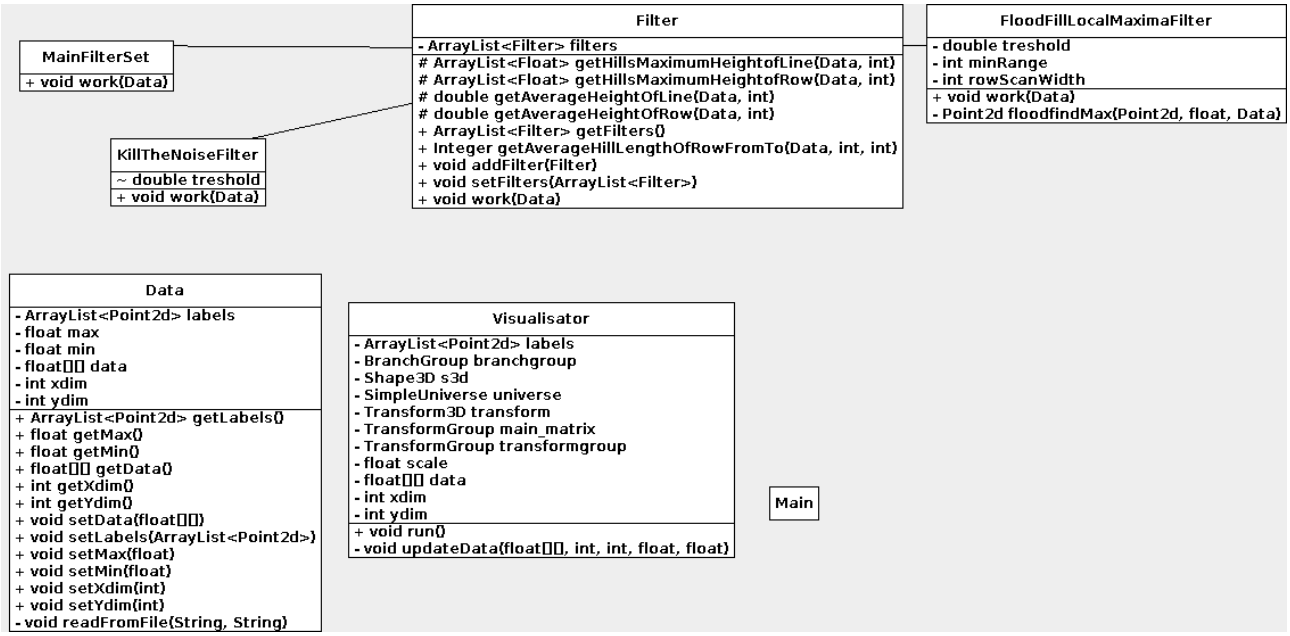
1.Ursprungsdaten mit markierten, durch den Algorithmus gefundenen Labels (grün)

2.Daten nach dem ersten Filter mit Darstellung der gegebenen Validierungslabels

3.Daten nach beendetem Algorithmus (gefundene Labels). Grüne Markierungen bedeuten, dass der Label exakt mit der Position der gegebenen Validierungslabels übereinstimmt.



Klassendiagramm und Benutzung



Filter.class: Enthält alle Grundfunktionen jedes einzelnen Filters

KillTheNoiseFilter.class: Erbt von Filter.class; filtert die gegeben Daten so, dass nur Berge übrigbleiben

FloodFillLocalMaximaFilter.class: Erbt von Filter.class; filtert die gegeben Daten so, dass nur die Maxima übrig bleiben.

MainFilterSet.class: Erbt von Filter.class; hält selber eine Liste mit allen Filtern, die nach und nach die Daten filtern.

Visualisator.class: Generiert das 3D Modell

Data.class: Hält die gegebenen Dateien ('data#' und 'label#')

Main.class: Startet das Programm

Usage: Programm starten, als Parameter werden die 'data' und 'label' Dateien benötigt, Bsp:

'java – jar ###.jar -data0.csv -label0.csv' *

*Die Dateien müssen im Resource Ordner enthalten sein