Miro Wakounig, BSc

Extraktion digitaler Karten von Schigebieten aus GPS-Tracks von Schifahrern

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

Masterstudium Geomatics Science

eingereicht an der

Technischen Universität Graz

Betreuer

Ao.Univ.-Prof.i.R. Dr.phil. tit.Univ.-Prof. Norbert Bartelme

Institut für Geodäsie

Graz, 03 2015

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Datum

Unterschrift

Widmung - Posvetilo

Die vorliegende Arbeit ist das Ergebnis einer mehrjährigen Zeitspanne. Entlang dieses Weges gab es viele Begleiter, die mit dem Entstehen dieses Werkes zu tun hatten und denen ich gerne danken möchte.
Ein besonderer Dank gilt meinem Betreuer Ao.Univ.-Prof.i.R. Dr.phil. tit.Univ.-Prof. Norbert Bartelme, der mir die Freiheit und das Vertrauen aussprach, meine Masterarbeit nach meinen Ideen und Ermessungen umsetzen zu dürfen. Hilfreich stand er mir bei aufkommenden Fragen zur Seite und bot mir ein Forum, um Ideen und Ansätze zu diskutieren.
Na koncu študijskega obdobja je čas pogledati nazaj. Najprej se zahvalim staršem. Omogočila sta mi študij v Gradcu in mi bila moralna opora. Nikoli ne bom pozabil prvih dveh let študija. Živela sta z mano in verjela vame pri vsakem izpitu.
Zahvala velja bratu Danielu in ženi Miriam. Daniel mi je vedno spet odpiral nove perspektive med študijem. Omogočil mi je praktikum in iz tega sodelovanja se je rodila ideja za mastersko delo.
Hvala lepa Samu in Doris, ki sta me vedno spet spodbujala, da ob študiju delam še kaj drugega, bodi si na športnem ali kulturnem področju, kar me je in me bo vedno veselilo.
V posebno veselje so mi vedno bili obiski otrok obeh bratov. Za vse sem si smel vzeti čas, da nisem stalno sedel ob študiju.
Na koncu bi se rad zahvalil tudi Igorju, Christianu in Damijanu. Dnevno smo se srečali v Gradcu pri kuhanju, kavi in pogovoru.

Kurzfassung

Schifahren ist eines der beliebtesten Hobbys in Österreich. Große Schigebiete bieten mehr als 100 Kilometer präparierter Pisten und einige dutzend Seilbahnanlagen. Viele Schifahrer, vor allem solche, die sich in einem Schigebiet nicht gut auskennen, würden sich ein Navigationssystem für Schigebiete wünschen.
Damit ein Navigationssystem umgesetzt werden kann, müssen dafür zunächst die Grundlagen geschaffen werden. Mit einer in dieser Masterarbeit entwickelten Software wird aus einem Satz von GPS-Trajektorien eine digitale Karte für ein Schigebiet extrahiert. Die Inputdaten stammen von Schifahrern, die ihren Schitag mit Hilfe eines Smartphones getrackt haben und die Daten zur Verfügung stellten. Die Software extrahiert mit Hilfe von verschiedensten Algorithmen Lifte und Pisten eines Schigebietes. Der Aufbau des Programmes ist so gestaltet, dass die Berechnungen automatisch durchgeführt werden können. Dem Benützer ist die Möglichkeit eingeräumt, die Eingabeparameter der einzelnen Berechnungsschritte manuell zu ändern. Die Berechnungszeit für eine digitale Karte eines Schigebietes hängt von der Datenanzahl und der Wahl der Parameter ab. Das Endergebnis der Berechnungen sind Lifte, Pisten und ein Netz von Verbindungen. Das berechnete Netz kann als digitale Karte für ein Navigationssystem verwendet werden.
Der praktische Teil wird mit der Software Matlab umgesetzt. Als Beispiel werden die digitalen Karten für zwei Schigebiete berechnet und im Bericht interpretiert.

Abstract

Skiing is one of the most popular hobbies in Austria. Large ski areas offer more than 100 kilometers of slopes and a few dozen of lifts. Many skiers who are not familiar with a ski resort, would like to use a navigation system.
First the foundation for a navigation system had to be created. In this thesis a software was developed, which extracted a digital map of a ski area from a set of GPS-trajectories. The GPS-trajectories were provided by skiers who tracked their ski trips with smartphones. The software extracted lifts and slopes with the help of different algorithms.
Although the software can work automatically, the user has the option to change the input parameter of every individual step of calculation. The computation time for a digital map of a ski area depends on the number of data points and the choice of the input parameters. The final results of the whole procedere were lifts, slopes and a network of connections about the whole ski area. A possible navigation system could be built onto the calculated digital map.
The practical part of the master thesis was written with the software Matlab. As an example, in the master thesis digital maps of two skiing areas were calculated and interpreted.

Povzetek

V Avstriji je smučanje je zelo priljubljen konjiček. Velika smučišča ponujajo več kot 100 kilometrov urejenih smučarskih prog in vrsto smučarskih naprav. Nekateri smučarji, predvsem tisti, ki smučišča še ne poznajo dobro, si želijo navigacijskih sistemov.
Za uresničitev navigacijskega sitema pa je treba najprej pripraviti osnove, saj je podlaga za navigacijo digitalna karta smučišča. Glavna naloga diplomske naloge je razvitje programske opreme, ki ustvari iz zbranih tirnic GPS digitalno karto. Tirnice, uporabljene v delu, so bile posnete s pomočjo mobitelov. Na razpolago so jih dali smučarji.
V delu predstavljeni softver s pomočjo raznih algoritmov iz posnetih tirnic izvleče smučarske proge in naprave. Programiran je tako, da lahko računa samostojno. Kljub temu omogoča uporabniku, da lahko tudi samodejno nastavi parametre posameznih korakov. Čas izračunanja je odvisen od obsežnosti zbranih podatkov in izbire parametrov. Končni rezultat so smučarske proge in smučarske naprave ter mreža povezav, ki je razpreda po celem smučišču. Izračunano omrežje je podgradnja za navigacijski program.
Praktični deli so programirani s pomočjo softvera Matlab. Vzorčno sta izračunani digitalni karti dveh smučišč, ki sta opisani in interpretirani v poročilu.

Inhaltsverzeichnis

1. Einleitung

Ursprünglicher Ausgangspunkt für diese Arbeit war das Ausarbeiten einer Navigationslösung für Schigebiete, um Schifahrern die Orientierung in großen Schigebieten zu ermöglichen. Im Laufe der Recherche wurde ersichtlich, dass es keine digitalen Karten von Schigebieten gab bzw. diese nicht frei verfügbar waren. Aus diesem Grund wurde die Zielsetzung auf die Erstellung einer digitalen Karte eines Schigebietes geändert.

Im Internet gibt es viele Anbieter, die Schigebiete darstellen. Eine bekannte Software ist *Google Earth*, welche ein virtuelles Abbild der Erde anbietet. Die Lifte und Pisten könnten anhand der Luftbilder „per Hand“ ausgelesen werden. Zum einen wäre dieser Vorgang mühsam und die Durchführung für eine größere Anzahl von Schigebieten unmöglich, da dieser Vorgang nicht automatisierbar ist und zum anderen sind die Luftbilder nur Momentaufnahmen des Aufnahmezeitpunktes. Im Schigebiet Saalbach stammten die Luftbilder aus den Jahren 2000 bzw. 2009, im Schigebiet Sölden stammten sie aus den Jahren 2000 bzw. 2002. Als Beispiel, warum nur ein aktuelles Luftbild korrekte Ergebnisse liefern würde, sei das Schigebiet Saalbach angeführt, in welchem in der Wintersaison 2014/15 zwei neue Lifte eröffnet wurden, die aus den Luftbildern nicht detektierbar wären. Daher wurde der Ansatz, Lifte und Pisten aus Luftbildern zu entnehmen, verworfen. *Google Earth* wird als Visualisierungswerkzeug für die Ergebnisse des parallel zu dieser Arbeit erstellten Programmes verwendet, indem die Ergebnisse in „*.kml-Files“ geschrieben werden, die in *Google Earth* betrachtet werden können.

Ein weiterer namhafter Anbieter von Karten ist die Internetseite *„www.openstreetmap.org“*. Auf dieser Seite werden Schigebiete grafisch dargestellt. Die Daten werden von Besuchern der Seite generiert, editiert und stehen ihnen zur freien Verfügung. Das Problem bei der Darstellung von Schigebieten auf dieser Seite ist, dass kaum Pisten eingetragen sind. Weiters bestehen viele Lifte nur aus zwei Knoten - dem Eintritts- bzw. Austrittspunkt. Die Daten dieser Homepage werden für den Vergleich der berechneten Ergebnisse verwendet.

Die Seite *„www.gpsies.com“* ist ein Forum für Sportbegeisterte. Jeder Benützer kann GPS-Trajektorien auf die Seite laden. Die Seite bietet eine ansprechende Visualisierung der Trajektorien an. Die Daten können von anderen Personen betrachtet, kommentiert und heruntergeladen werden. Die Verwendung der Daten ist für nicht kommerzielle Zwecke erlaubt. Auf dieser Seite gibt es je nach Größe des Schigebietes unterschiedlich

viele Trajektorien von Schifahrern, die ihre Daten zur Verfügung stellen. Die Daten für zwei Schigebiete wurden heruntergeladen und als Grundlage für die Berechnung einer digitalen Karte verwendet. Die Ergebnisse sind in Kapitel 4 beschrieben.

1.1. Aufgabenstellung

Aus einem Satz von Trajektorien, die von Schifahrern aufgenommen werden, soll ein digitales Schigebiet extrahiert werden. Ziel ist es, so viele Lifte und Pisten wie möglich aus den Daten zu ermitteln. Die Berechnung soll automatisiert werden.
Lifte sind zumeist starre Objekte, die bei genügend großer Datenanzahl sehr gut bestimmbar sind. Die Pisten können auf zwei unterschiedliche Arten definiert und extrahiert werden. Einerseits gibt es die Definition von Pisten als kürzeste Verbindung zwischen zwei Liften, andererseits gibt es die Definition der Piste als ein breites großes Objekt, welches viele unterschiedliche Varianten von Pistenverläufen erlaubt. Ziel ist es, beide Arten von Pisten zu detektieren. Vor allem die zweite Art liefert ein über das gesamte Schigebiet verteiltes Netz von Pistenpunkten, auf welchem weitere Applikationen aufgebaut werden können.
Ziel der Arbeit ist weiters, ein Programm zu schreiben, welches automatisch Pisten und Lifte aus einem Datensatz von Trajektorien extrahiert. Die einzelnen Schritte sollen übersichtlich gestaltet werden um auch anderen Personen die Arbeit mit dieser Software zu ermöglichen.

2. Theoretische Grundlagen

In diesem Kapitel wird auf einige Definitionen und Überlegungen bezüglich Objekten, die in einem Schigebiet vorkommen, eingegangen. In den Abschnitten 2.1 bis 2.3 sind die Eigenschaften von Liften und Pisten beschrieben und wie sie in einem GPS-Track aussehen. In den Abschnitten 2.4 bis 2.6 sind die mathematischen und programmiertechnischen Grundlagen beschrieben. Die Abschnitte 2.7 und 2.8 befassen sich mit den Grundlagen, wie aus einer Vielzahl von einzelnen Pfaden auf eine gemittelte Piste oder Lift geschlossen werden kann. Immer, wenn in diesem Kapitel von einem nicht näher spezifizierten Algorithmus die Rede ist, bezieht sich diese Formulierung auf den Algorithmus, der in Kapitel 3 beschrieben ist.

2.1. Seilbahnen

In jedem Schigebiet gibt es unterschiedliche Arten von Seilbahnen. Die meisten können in drei Kategorien unterteilt werden:

- Luftseilbahnen
- Schienenseilbahnen
- Schlepplifte

Jede Seilbahn besteht aus zumindest zwei Stationen und einer Lifttrasse. Vor allem die Stationen von Schienenseilbahnen und größeren Luftseilbahnen sind überdacht. Schlepplifte und kleinere Luftseilbahnen haben selten ein überdachtes Stationsgebäude. Die Schwierigkeit bei Messungen in überdachten Stationen besteht darin, dass in Gebäuden keine GPS Messungen getätigt werden können. Die Trajektorien in Gebäuden sind entweder verfälscht oder es gibt keine.

Luftseilbahnen

Zu den Luftseilbahnen, die in Schigebieten vorkommen, zählen Gondelbahnen, Sessellifte und Kabinenlifte. Die Fahrbetriebsmittel klemmen bei Luftseilbahnen auf zumindest einem Tragseil. Auf Grund des Gewichtes der Fahrbetriebsmittel und der mitfahrenden Personen hängt das Tragseil von Luftseilbahnen zwischen zwei Liftstützen durch.

Zumeist transportieren Seilbahnen in Schigebieten Schifahrer über einen größeren Höhenunterschied vom Tal auf den Berg. Es gibt aber auch Seilbahnen, die die Aufgabe haben, zwei Schigebiete zu verbinden, z.B. die Gondelbahn *Peak-2-Peak* in Whistler (Kanada), die zwei Schigebiete über ein Tal hinweg verbindet (siehe Abb. 2.2). Gleiches gilt für die Gondelbahn *Gletscherexpress* in Sölden, die den Schifahrern die Verbindung zwischen zwei Pistenhängen ermöglicht, die sonst nur über große Umwege erreichbar wären. Dem Lift selber kann keine Piste zugewiesen werden über welche die Schifahrer vom Ausstiegspunkt zum Einstiegspunkt des Liftes gelangen können. Bei den meisten Liften gibt es eine solche Verbindung, ohne dass weitere Lifte dazwischen verwendet werden müssen. Seilbahnen, die Teile von Schigebieten verbinden, werden in beide Richtungen - talaufwärts und bergabwärts - genutzt. Eine weitere Besonderheit von Luftseilbahnen ist, dass die Fahrbetriebsmittel in der Luft befördert werden, die Lifttrassen also von Pisten gekreuzt werden können. Außer beim Umbau oder Neubau ändern sich die Lifttrassen von Luftseilbahnen nicht. GPS-Messungen können generell vom Tragseil gestört werden (Mehrweg-Effekte) und im Speziellen bei Gondeln vor allem, weil die Fahrbetriebsmittel geschlossen sind. Die Bestimmung der Höhentrajektorie einer Luftseilbahn ist problematisch, einerseits wegen der gerade erwähnten Probleme und andererseits wird ein Großteil der verwendeten GPS-Daten mit Geräten wie Handys aufgenommen, die eine exakte Bestimmung vor allem der Höhenkoordinaten nicht zulassen.

Schienenseilbahnen

Schienenseilbahnen lassen sich in 3 Bauarten unterteilen:

- Schiefe Seilebenen - wurden in den Anfangszeiten der Eisenbahnen verwendet, werden nicht mehr gebaut und in Schigebieten auch nicht mehr verwendet
- Standseilbahnen - ein Fahrbetriebsmittel wird an Schienen gebunden und wird von einem oder mehreren Seilen gezogen, Bsp.: Graz - *Schlossbergbahn*, Kaprun - *Gletscherbahn 2*, Bad Gastein - *Schlossalmbahn*
- Kabelbahnen - werden in Schigebieten nicht verwendet, die berühmteste Kabelbahn ist in San Francisco in Betrieb

Die einzige Bauart von Schienenseilbahnen, die in Schigebieten vorkommt, ist die Standseilbahn. Standseilbahnen transportieren Fahrbetriebsmittel auf Schienen. Die Fahrbetriebsmittel in Schigebieten im Winter sind geschlossene Kabinen oder Waggons,

die keine oder nur schlechte GPS-Messungen zulassen. Dies ist in den Tal- und Bergstationen von Standseilbahnen der Fall. Da Standseilbahnen entlang von Schienen verlaufen, sind die Trassen fixiert und vor äußeren Einflüssen, wie Wind, geschützt. Standseilbahnen sind oftmals Zubringer vom Tal auf den Berg, daher werden diese Bahnen teilweise oder ganz unterirdisch gebaut. Als trauriges Beispiel einer unterirdischen Schienenseilbahn gilt die *Gletscherbahn 2* in Kaprun, wo es im Jahr 2000 im 3.3. Kilometer langen Tunnel zu einer schweren Brandkatastrophe kam. In einem Tunnel können nur erschwert oder keine GPS-Messungen aufgenommen werden und daher können unterirdisch verlaufende Standseilbahnen vom Algorithmus nicht detektiert werden. Die *Schlossalmbahn* in Bad Gastein ist eine kombinierte Standseilbahn-Pendelbahn. Sie ist von der Talstation bis zur Mittelstation eine Standseilbahn und ab dort bis zur Bergstation eine Pendelbahn. Die meisten Lifte mit einer Zwischenstation werden vom Algorithmus als selbstständige Lifte detektiert: ein Lift von der Tal- zur Mittelstation und ein Lift von der Mittel- zur Bergstation.

Schlepplifte

Schlepplifte sind die am häufigsten verbreitete Seilbahnart. Sie sind am leichtesten zu bauen und auch am wartungsfreundlichsten. Die bekanntesten beiden Unterarten von Schleppliften sind der Tellerlift und der Ankerlift. Die Zugbügel werden an das Beförderungsseil geklemmt. Der Schifahrer hält bzw. setzt sich auf den Zugbügel und wird so bergauf befördert. Anders als bei Standseilbahnen und Luftseilbahnen sind die Stationen zumeist nicht überdacht, was bei der Ermittlung der Ein- bzw. Austrittspunkte hilfreich ist. Die Lifttrassen von Schleppliften müssen nicht geradlinig verlaufen, sondern können durchaus auch Knicke oder Kurven aufweisen. Steile oder abrupte Richtungsänderungen sind wegen der dadurch entstehenden Überbeanspruchung des Förderseiles nicht zu erwarten. Das Förderseil eines Schleppliftes kann, wie auch bei allen anderen Seilbahnen, bei GPS-Messungen Mehrwegeffekte erzeugen, was zu einer Verschlechterung der Messungen führen kann. Beim Benützen von Schleppliften kann der Schifahrer aus der Spur fahren, was ein Schlingern in den Daten zur Folge hat. Weiters kann es bei der Benutzung von Schleppliften zu Stürzen von Schifahrern kommen. Die Unterscheidung zwischen einer Lifttrasse und einer Sturzstelle kann nur über die Anzahl von Personen, die an einer Stelle die Lifttrasse verlassen, unterschieden werden. Eine weitere Besonderheit von Schleppliften ist, dass sie auf nicht ruhendem Gelände gebaut werden können, was zu Differenzen bei der Betrachtung von Ergebnissen führen kann. Ein nicht ruhendes Gelände ist z.B. ein Gletscher. Auf Gletschern können

die Lifteintrittspunkte je nach Jahreszeit variieren, wenn z.B. im Sommerbetrieb, oder wegen des Gletscherrückganges, nur Teile des Gletschers befahren werden können.

2.1.1. Eigenschaften der Lifte

Eine Eigenschaft von Lifttrassen ist, dass diese nahezu geradlinig verlaufen. Die GPS-Trajektorien der verschiedenen Schifahrer werden eingelesen und einzeln untersucht. Eine GPS-Trajektorie besteht aus einer größeren Anzahl von hintereinander folgenden Punkten. Zwischen zwei hintereinander folgenden Punkten wird ein Richtungswinkel berechnet. Wenn sich ein Richtungswinkel innerhalb eines Toleranzbereiches des Richtungswinkels eines vorherigen Punktepaares befindet, werden die involvierten Punkte als zusammengehörende Teiltrajektorie definiert. Dieses Verfahren splittet eine Trajektorie in kleinere Teile. Die Start- und Endpunkte einer Teiltrajektorie werden Knickpunkte genannt. Größere Knicke sind bei einem Lift aus technischer Sicht nicht zu erwarten, da das Tragseil sonst übermäßig strapaziert werden würde.
Eine weitere Eigenschaft der Lifte ist die Tatsache, dass diese Personen bergauf befördern, vor allem Schlepplifte. Personen, die mit dem Lift talwärts fahren, werden im Algorithmus nicht berücksichtigt. Es wird die Annahme getroffen, dass Bereiche, in denen ein Lift bergab fährt im Vergleich zu Bereichen, in denen er bergauf fährt, kürzer sind. Dieser Umstand wird im Algorithmus so berücksichtigt, dass in den Trajektorien nach Bereichen positiver Anstiege gesucht wird. Der Bereich zwischen zwei Anstiegen wird ebenfalls untersucht und gegebenenfalls zu einem gemeinsamen langen Anstieg verschmolzen, da es Seilbahnen gibt, die zwischen zwei Stützpfeilern bergab fahren wenn z.B. ein Tal überbrückt wird (siehe Abb. 2.2).
Schifahrer, die mit Schleppliften oder Standseilbahnen befördert werden, bleiben am Boden und dadurch entsprechen die Höhen der Trajektorienpunkte den Höhen des Geländes. Bei Luftseilbahnen wird der Schifahrer durch die Luft befördert. Es gibt drei kritische Bereiche bei der Detektion solcher Lifte. Der erste Bereich sind die Ein- bzw. Austrittspunkte von Luftseilbahnen. Ein Fahrbetriebsmittel, wie z.B. ein Sessel, ist an ein Tragseil geklemmt und fährt mit konstanter Geschwindigkeit. Wenn sich die Schifahrer beim Eintrittspunkt auf den Sessel setzen, dann wirken die Gewichte der Schifahrer gegen die konstante Vorwärtsbewegung des Förderseiles. Die Gewichtskraft der Personen wirkt gegen die Zugkraft, welche vom Seil ausgeübt wird. Solange die Schifahrer Bodenkontakt haben, ändert sich die Höhentrajektorie nicht. Sobald der Sessel die Liftstation verlässt, sackt der Sessel ab, bis die Schubkraft des Liftes den Sessel

anhebt und bergauf befördert. Angenommen eine Messung würde in der Talstation und die nächste in der Absenkung getätigt werden, dann würde der Algorithmus den Startpunkt eines Liftes in der Absenkung detektieren, da es ab diesem Punkt einen konstanten Höhenanstieg gäbe. Damit wäre ein Lift bereits um einige Meter falsch detektiert worden. Gleiches geschieht bei Bergstationen, wo der Sessel vor dem Ausstiegspunkt kurz angehoben und danach gesenkt wird um ein sanfteres Aussteigen zu ermöglichen. Am stärksten tritt dieser Effekt bei fix geklemmten Sesselliften (zumeist älteren Modellen) auf, da die Fahrbetriebsmittel fix ans Umlaufseil geklemmt sind und in den Eintritts- und Austrittspunkten ungebremst durchfahren. Modernere Sessellifte sind kuppelbar und werden beim Eintritt in die Station vom Seil genommen, über eine Schiene abgebremst und beim Verlassen der Station wieder ans Seil geklemmt. Je langsamer das Anfahren geschieht, desto geringer ist das Durchsacken beim Start des Liftes.

Abbildung 2.1.: Bergstation eines Liftes, Quelle: `www.pixabay.com`

In Abbildung 2.1 ist die Absenkung vom letzten Stützpfeiler im Auslaufbereich des Liftes dargestellt. Ein weiterer kritischer Bereich sind die Liftstützen. Das Förderseil hängt zwischen den Liftstützen durch und wenn der Sessel bei den Stützen über die Förderrollen läuft, kommt es kurzzeitig zur Anhebung des Sessels und danach wieder zur Absenkung. Dieser Effekt ist weniger stark ausgeprägt als der Effekt bei den Ein- und Austrittspunkten von Liften. Der dritte Bereich sind Luftseilbahnen, die nicht das Ziel haben Schifahrer in die Höhe zu befördern, sondern z.B. zwei Teile eines Schigebietes zu verbinden. Die Seilbahn *Peak-2-Peak* in Whistler (Kanada) verbindet zwei Bergspitzen über ein Tal hinweg. Die Gipfel sind 4.4 Kilometer voneinander entfernt und die größte

Spannweite beträgt mehr als 3 Kilometer. Der absolute Höhenunterschied zwischen den beiden Stationen beträgt nur 36 Meter, da die Seilbahn jedoch absinkt, ist der gesamte Höhenunterschied um einiges höher. Da das Tragseil durch die Spannweite einen sehr starken Durchhang hat, hat der Algorithmus Schwierigkeiten, die korrekte Lifttrasse zu finden. Der Algorithmus detektiert den tiefsten Punkt des Durchhanges als Talstation, da der Lift ab diesen Punkt einen konstanten Anstieg aufweist. Der erste Teil des Liftes wird als Piste detektiert, da der „Lift“talabwärts verläuft. An der tiefsten Stelle des Durchhanges befindet sich das Förderseil 436 Meter über dem Talboden (siehe Abb. 2.2). Solche Spezialfälle können nur mit Vorwissen behandelt werden, indem die Piste und der Lift manuell zusammenfügt und als ein gemeinsamer Lift definiert werden.

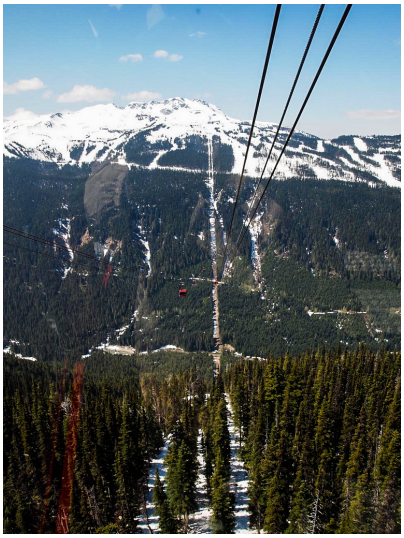


Abbildung 2.2.: Peak-2-Peak-Lift in Whistler Mountain Kanada, Quelle: `www.flickr.com`, Autor: Walter Lim
Lizenz: `www.http://creativecommons.org/licenses/by/2.0/at/legalcode`

Die Anstiege werden aus den Teiltrajektorien anhand ihrer Höhenunterschiede gefiltert. Neben den langen, zusammenhängenden Anstiegen, die tatsächlich Lifte darstellen, sind auch viele kleine Anstiege, die keine Lifte sind. Diese können mit Hilfe eines Grenzwertes gefiltert werden. Als Näherung für den Grenzwert kann die Länge des kürzesten Liftes eines Schigebietes laut Homepage des Seilbahnbetreibers gewählt werden. Viele kleinere Anstiege stammen von langen Schwüngen, Pistenkuppen oder kurzen Gegenanstiegen beim Schifahren und haben mit Liften nichts zu tun.
Beim Detektieren von Liften geht es im Allgemeinen um das Finden der Lifttrassen und im Speziellen um das Finden der Liftanfangs- und Endpunkte. Als Liftanfangspunkt kann der Punkt definiert werden, an dem sich das Umlaufseil in die umgekehrte Richtung wendet. Eine weitere Möglichkeit ist, den Punkt an der Stelle der Antriebs- bzw.

Umlegestation zu definieren. Für den Algorithmus wird der Liftanfangspunkt als der Punkt, an dem die Schifahrer einsteigen, definiert. Bei Schleppliften ist dieser Punkt besser detektierbar als bei Stand- oder Luftseilbahnen, da die Einstiegspunkte nicht überdacht sind. Große Seilbahnen (vor allem Gondelbahnen) haben Einstiegspunkte in großen Stationsgebäuden (siehe Abb. 2.3).

Abbildung 2.3.: Stationsgebäude der Mittelstation der Gaislachkogelbahn in Sölden, Quelle: `www.skiline.com`

In Sölden gibt es den Schlepplift *Seiterjöchl*, welcher den Einstiegspunkt am Gletscher hat. Die Antriebsstation wurde aus praktischen Gründen einige hundert Höhenmeter unterhalb des Eintrittspunktes auf festem Grund gebaut. Der untere Teil zwischen Einstiegspunkt und Antriebsstation konnte noch nie befahren werden. Daher kann der Lift nur vom Eintrittspunkt weg detektiert werden.
Ungeachtet der Problematiken, die beim Detektieren von Seilbahnen auftreten, sind die Informationen, die man sich auf den Homepages der Seilbahnbetreiber abrufen kann, eine hilfreiche Quelle. Pro Jahr werden in größeren Schigebieten ein bis zwei Seilbahnen ausgetauscht oder neu gebaut. Seit dem Jahr 2014/15 sind im Schigebiet Skicircus Saalbach Hinterglemm Leogang zwei neue Seilbahnen in Betrieb:

- die Seilbahn *Polten 8er Sessellift* ersetzt den *Polten 4er Sessellift* (€7 Millionen)
- die *Steinbergbahn* ist neu gebaut und verbindet Leogang mit dem Asitzkogel (€17 Millionen)

„Der Skicircus Saalbach Hinterglemm Leogang investierte seit dem Jahr 2000 über 300 Millionen Euro in die Modernisierung und Komfortverbesserung des Skigebietes“

(www.saalbach.com). Aus diesem Grund ist die Aktualität der Daten entscheidend für die Korrektheit und Vollständigkeit der ermittelten Lifte. Wenn veraltete Daten verwendet werden, werden auch alte Lifttrassen oder Lifte detektiert. Diese müssen manuell aus den Ergebnissen gelöscht werden.
Schwierigkeiten beim Ermitteln von Liften verursachen Lifte, die auf nicht ruhenden Böden gebaut sind. Zumeist werden auf Gletschern nur Schlepplifte gebaut. Da die Fundamente der Stützen nicht betoniert werden können, müssen diese im Eis verankert werden. Dafür werden spezielle Gletscherstützen gebaut, die man je nach Bedarf neigen oder versetzen kann. Im Schweizer Schigebiet Saas Fee steht der *2er-Schlepplift Feechatz* auf dem Feegletscher. Der Feegletscher bewegt sich pro Jahr 50-90 Meter oder 14-25 cm pro Tag. Um diese Bewegung auszugleichen, wird der gesamte Schlepplift nach einer gewissen Gletscherbewegung wieder bergauf gezogen. Der Antrieb des Liftes kann bis zu 34 Meter verschoben werden und der Einstiegspunkt des Liftes kann bis zu 40 Meter pro Saison variieren. Folgend gibt es beim Detektieren des Anfangs- und Endpunktes große Schwankungen. Eine andere Möglichkeit, Liftstützen auf einem Gletscher zu bauen, ist es ein Loch zu graben und das Fundament im festen Boden zu verankern. Dabei besteht die Gefahr, dass nachrückendes Gletschereis die Stützen beschädigt. Weltweit gibt es nur wenige Luftseilbahnen, die über oder auf einem Gletscher gebaut sind, so z.B. die Gondelbahn *Schwarze Schneidbahn 2* in Sölden. Diese Bahn befördert Schifahrer über den Rettenbach-Ferner. Das Fundament der Stütze acht ist am Grund des Gletschers betoniert und die Stütze mit einem beheizbaren Stahlkeil versehen, welcher bei Bedarf angeheizt werden kann, um das Gletschereis um die Stütze schmelzen zu lassen.

2.2. Pisten

Schipisten sind gesicherte und für den Tourismus vorgesehene Bereiche eines Berges, die befahren werden dürfen. In jedem Schigebiet gibt es unterschiedliche Arten von Schipisten, die sich nach ihrer Steilheit, Länge, Präparation und Schwierigkeitsgrad unterscheiden. Das bekannteste Unterscheidungskriterium ist anhand der Schwierigkeitsgrade:

- grüne Pisten - Übungshänge
- blaue Pisten - leichte Abfahrten mit nicht mehr als 25% Längs- und Quergefälle
- rote Pisten - mittelschwere Abfahrten mit nicht mehr als 40% Längs- und Quergefälle

- schwarze Pisten - schwere Abfahrten mit mehr als 40% Längs- und Quergefälle

Obwohl Pisten unterschiedlichen Schwierigkeitsgraden zugewiesen werden können, können zusammenhängende Abfahrten mit durchgehender Steilheit von mehr als 40% nicht automatisch als schwarze Pisten definiert werden. Es muss die gesamte Piste detektiert werden, da Teile von Pisten steiler bzw. weniger steil als die Vorgaben der Schwierigkeitsgrade sein können. Weiters können Pisten Varianten haben, die nicht alle dem gleichen Schwierigkeitsgrad angehören. In dieser Arbeit wird nicht versucht, die Pisten nach Kategorien zu filtern, sondern die Pisten als Gesamtes zu detektieren.
Pisten haben, ähnlich wie Lifte, ihre speziellen Eigenschaften. Sie können von Liften anhand der Höhenunterschiede, die sie überwinden, unterschieden werden, obwohl sie Teilstücke haben, an denen man bergauf fahren muss (siehe Abb. 2.4). Der Anstieg eines Liftes ist viel länger als die kleinen Anstiege, die einer Piste zugewiesen werden. Schifahrer fahren selten eine Piste geradlinig von oben nach unten. Sie fahren Schwünge, bleiben stehen und drehen sich öfters um, was auch in den Daten sichtbar ist. An gewissen Punkten (z.B. Übergänge oder Skihütten) gibt es viele Messungen nah beieinander, ohne dass sich die Höhe stark ändert. Die meisten Pisten sind nicht gerade und gleichmäßig steil, sondern eine Mischung aus Hängen und Übergängen. Viele Schifahrer fahren zu den Rändern solcher Übergänge und bleiben dort stehen. Dadurch entstehen terassenförmige Trajektorien wie sie in Abbildung 2.4 zu sehen sind.

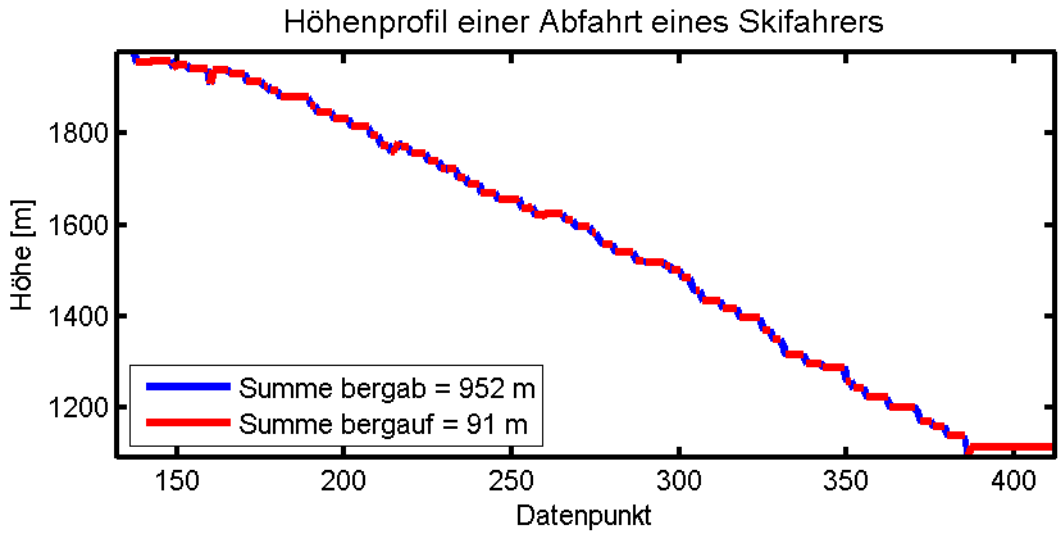


Abbildung 2.4.: Höhenprofil einer Abfahrt eines Skifahrers, Quelle: eigene Darstellung

Würde jeder positive Höhenanstieg das Ende einer Piste definieren, gäbe es sehr viele kurze Pisten. Darum werden Pisten im Algorithmus nicht anhand negativer

Höhendifferenzen gefiltert, sondern zunächst die Lifte detektiert und die restlichen Daten als Teile von Pisten angenommen, ungeachtet der positiven Anstiege, die sich darin befinden. Pisten ändern sich, ähnlich wie Lifte, pro Jahr. Für die bestmögliche Detektion von Pisten sind aktuelle Daten notwendig. Einige Pisten oder Teile von Pisten werden vom Algorithmus trotzdem nicht detektiert. Zum Beispiel verläuft eine Piste in Sölden durch einen Tunnel. Im Tunnel können keine GPS Messungen getätigt werden. Daher detektiert der Algorithmus zwei Pisten, eine Piste zum Tunneleingang hin und eine Piste vom Tunnelausgang weg. Weil aber zwischen den beiden Pisten kein genügend großer Höhenunterschied überwunden wird, um die Definition eines Liftes zu erfüllen, werden die Pisten an den Tunnelportalen nicht geteilt, sondern als eine gemeinsame Piste detektiert. Im Bereich des Tunnels gibt es Pistenpunkte nur an den Enden des Tunnels. Informationen über Besonderheiten kann man auf den Homepages der Seilbahnbetreiber erhalten, oder durch Zuhilfenahme von anderen Visualisierungstools (*Google Earth*). Kurz zusammengefasst werden Anstiege anhand von positiven Anstiegen gefiltert und alle übrig bleibenden Daten sind Abfahrten.

2.3. Datenbeschaffung und Datenstruktur

Die Daten, die für die Berechnung verwendet wurden, stammten von der Homepage „www.gpsies.com“. Auf besagter Homepage stellen User selbstständig aufgenommene GPS-Tracks zur Verfügung. Die Seite bietet im Gegenzug mehrere Optionen der Visualisierung und der Datentransformation an. Die Daten können von anderen angemeldeten Benützern betrachtet, kommentiert und heruntergeladen werden. Die Nutzungsbedingungen der Seite räumen dem Besucher das Recht ein, die Tracks herunterzuladen und weiter zu verwenden, wenn kein kommerzieller Nutzen daraus gezogen wird. Die Seite hat mehr als 200000 Tracks (Stand: November 2013). Die Seite bietet auch eine API-Schnittstelle [3] über welche Daten automatisch heruntergeladen werden können.
Da die Seite vor allem im europäischen Raum angesiedelt ist, gibt es für die Skigebiete in Österreich, Deutschland, Schweiz, Italien und Frankreich eine große Anzahl von Daten. Die Datenqualität der zur Verfügung gestellten Daten ist nicht homogen, die Datenstruktur ist es. Da die Korrektheit der Lagekoordinaten nicht beurteilt werden kann, werden die Höhendaten visuell überprüft. Manche Tracks haben keine Höhendaten, bei manchen Tracks steigen oder fallen die Höhendaten sprunghaft. Bei Tracks mit fehlenden Höhenangaben werden für die fehlenden Höhen aus den vorhergehenden und

nachkommenden Höhen gemittelt. Tracks ohne eine einzige Höhenangabe können nicht verwendet werden und werden aus den Berechnungen ausgeschlossen.
Eine alternative Möglichkeit fehlende Höhendaten zu ersetzen ist, die Höhendaten mittels einer Schnittstelle, die von Google zur Verfügung gestellt wird (*google elevation api [5]*), automatisch abzufragen. Die Anzahl der Abfragen ist auf 2500 Abfragen pro Tag mit maximal 512 Standorten pro Abfrage limitiert. Insgesamt können mit einer freien Lizenz 25000 Standorte pro Tag abgefragt werden. Ein Problem in der Praxis ist, dass bei Abfragen von Höhen diese auf die Oberfläche des Höhenmodelles bezogen sind, Luftseilbahnen jedoch in beträchtlicher Höhe verlaufen und somit Fehler vorprogrammiert sind.
Eine alternative Möglichkeit eines freien Anbieters einer Schnittstelle für Höhenabfragen ist die Seite *mapquest.com* [6], welche auf Basis des Höhenmodelles von *OpenStreetMap* Abfragen von Höhendaten anbietet.

2.4. Koordinatentransformationen

Ein Großteil der GPS-Koordinaten, die mit einem Handy aufgenommen werden, sind ellipsoidische Koordinaten im WGS-84 System. Beim Ausführen des Algorithmus werden Schrägdistanzen benötigt. Dazu müssen die ellipsoidischen Koordinaten in kartesische Koordinaten transformiert werden. Weiters werden die ellipsoidischen Koordinaten in UTM-Koordinaten transformiert. Die Verebnung wird angewendet, weil die Knickwinkel beim Splitten von Trajektorien nur mit verebneten Koordinaten berechnet werden können. Ein Großteil der Teilschritte wird mit verebneten UTM-Koordinaten berechnet und nur das Ergebnis wieder in ellipsoidische Koordinaten transformiert. Nachfolgend sind die Transformationen zwischen den verschiedenen Systemen im Detail beschrieben.

2.4.1. Ellipsoidische Koordinaten zu kartesischen Koordinaten

Für die Berechnung von Schrägdistanzen müssen die ellipsoidischen Koordinaten in kartesische Koordinaten transformiert werden. Für die Transformation werden zunächst der Normalkrümmungsradius N, aus dem Polkrümmungsradius c und der Hilfsvariable V berechnet. Die Variablen a und b sind die Halbachsen des WGS-84- Ellipsoides und e^2 ist die erste numerische Exzentrizität (siehe Hofmann-Wellenhof und Kienast 2010: Kapitel 4.1 und 4.3) [10].

$$a = 6378137 \text{ [m]}$$

$$
\begin{aligned}
b &= 6356752.3142 \text{ [m]} \\
c &= \frac{a^2}{b} \text{ [m]} \\
e'^2 &= \frac{a^2 - b^2}{b^2} \text{ [ ]} \\
V &= \sqrt{1 + e'^2 \cos^2\varphi} \text{ [ ]} \\
N &= \frac{c}{V} \text{ [m]}
\end{aligned}
$$

Die ellipsoidischen Koordinaten werden anschließend in kartesische Koordinaten transformiert (siehe Hofmann-Wellenhof und Kienast 2010: Kapitel 6.1.2) [10]

$$
\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} (N+H) \cdot \cos\varphi \cdot \cos\lambda \\ (N+H) \cdot \cos\varphi \cdot \sin\lambda \\ (\frac{b}{V} + H) \cdot \sin\varphi \end{bmatrix} \tag{2.1}
$$

2.4.2. Ellipsoidische Koordinaten zu UTM-Koordinaten

Bei der Verebnung von ellipsoidischen WGS-84 Koordinaten zu UTM-Koordinaten wird ein Hauptmeridian *HM* benötigt, der aus den λ-Koordinaten gemittelt wird. Mit Hilfe der gemittelten φ-Koordinate und dem Hauptmeridian kann die UTM Zone und folgend der mittlere Meridian der Zone bestimmt werden. Die ellipsoidischen Koordinaten in den folgenden Formeln werden in Radiant eingefügt. Für die Berechnungen werden einige Hilfsvariablen

$$
\begin{aligned}
\alpha &= 6367449.1458 \text{ [m]} \\
\beta &= -2.51882792 \cdot 10^{-3} \text{ [ ]} \\
\gamma &= 2.64354 \cdot 10^{-6} \text{ [ ]} \\
\delta &= -3.45 \cdot 10^{-9} \text{ [ ]}
\end{aligned} \tag{2.2}
$$

benötigt. Mit diesen Variablen werden die Hilfsgrößen in Gleichung 2.3 berechnet. Die Fußpunktsbreite B ist die Breite vom Äquator zur φ-Koordinate. Die Variable l ist die Differenz der Längenkoordinate zum Mittelstreifen der UTM-Zone

$$
\begin{aligned}
B &= \alpha(\varphi + \beta\sin(2\varphi) + \gamma\sin(4\varphi) + \delta\sin(6\varphi)) \text{ [ ]} \\
t &= \tan\varphi \text{ [ ]}
\end{aligned}
$$

$$
\begin{aligned}
e^{'2} &= \frac{a^2 - b^2}{b^2} \ [\,] \\
\eta_2 &= e^{'2} cos^2\varphi \ [\,] \\
k_0 &= 0.9996 \ [\,] \\
l &= \lambda - HM \ [\,]
\end{aligned}
\tag{2.3}
$$

Weitere Informationen und Erklärungen zur Berechnung der Fußpunktsbreite sind in der Literatur gegeben (Hofmann-Wellenhof und Kienast 2010: Kapitel 4.3.1) [10].
Der Hochwert x wird mit nachfolgenden Formeln

$$
\begin{aligned}
a_1 &= \frac{t}{2} N \cos^2(\varphi) l^2 \ [\mathrm{m}] \\
a_2 &= \frac{t}{24} N \cos^4(\varphi)(5 - t^2 + 9\eta_2 + 4\eta_2^2) l_4 \ [\mathrm{m}] \\
a_3 &= \frac{t}{720} N \cos^4(\varphi)(61 - 58t^2 + t^4 + 270\eta_2 - 330t^2\eta_2) l^6 \ [\mathrm{m}] \\
a_4 &= \frac{t}{40320} N \cos^8(\varphi)(1385 - 3111t^2 + 543t^4 - t^6) l^8 \ [\mathrm{m}] \\
x &= B + a_1 + a_2 + a_3 + a_4 \ [\mathrm{m}] \\
x &= x \cdot k_0 \ [\mathrm{m}]
\end{aligned}
\tag{2.4}
$$

berechnet. Für die Bestimmung des Rechtswertes y werden nachfolgende Formeln verwendet

$$
\begin{aligned}
b_1 &= N \cos(\varphi) l \ [\mathrm{m}] \\
b_2 &= \frac{1}{6} N \cos^3(\varphi)(1 - t^2 + \eta_2) l^3 \ [\mathrm{m}] \\
b_3 &= \frac{1}{120} N \cos^5(\varphi)(5 - 18t^2 + t^4 + 14\eta_2 - 58t^2\eta_2) l^5 \ [\mathrm{m}] \\
b_4 &= \frac{1}{5040} N \cos^7(\varphi)(61 - 479t^2 + 179t^4 - t^6) l^7 \ [\mathrm{m}] \\
y &= b_1 + b_2 + b_3 + b_4 \ [\mathrm{m}] \\
y &= (y \cdot k_0 + 500000) \ [\mathrm{m}]
\end{aligned}
\tag{2.5}
$$

Diese Art der Abbildung wird Gauß-Krüger-Abbildung genannt (siehe Hofmann-Wellenhof und Kienast 2010: Kapitel 5.3.1) [10]. Sie wird für die Umrechnung von ellipsoidischen Koordinaten in Gauß-Krüger-Koordinaten in der Ebene verwendet. Die Ergebnisse x und y werden zusätzlich mit dem Maßstabsfaktor k_0 multipliziert um aus Gauß-Krüger Koordinaten UTM Koordinaten zu bestimmen.

2.4.3. UTM-Koordinaten zu elllipsoidischen Koordinaten

Für die Berechnungen von φ und λ Koordinaten aus UTM Koordinaten werden einige Hilfsvariablen

$$\begin{aligned}
\bar{\alpha} &= 6367449.1458 \text{ [m]} \\
\bar{\beta} &= 2.51882658 \cdot 10^{-3} \text{ [ ]} \\
\bar{\gamma} &= 3.70095 \cdot 10^{-6} \text{ [ ]} \\
\bar{\delta} &= 7.45 \cdot 10^{-9} \text{ [ ]}
\end{aligned} \tag{2.6}$$

und einige Hilfsgrößen

$$\begin{aligned}
B &= x \text{ [m]} \\
\bar{B} &= \frac{B}{\hat{\alpha}} \text{ [ ]} \\
\varphi_f &= \bar{B} + \bar{\beta}\sin(2\bar{B}) + \bar{\gamma}\sin(4\bar{B}) + \bar{\delta}\sin(6\bar{B}) \text{ [ ]} \\
t_f &= \tan(\varphi_f) \text{ [ ]} \\
e^{'2} &= \frac{a^2 - b^2}{b^2} \text{ [ ]} \\
\eta_f^2 &= e^{'2}\cos^2(\varphi_f) \text{ [ ]} \\
N_f &= \frac{a^2}{b\sqrt{1+\eta_f^2}} \text{ [m]}
\end{aligned} \tag{2.7}$$

benötigt (siehe Hofmann-Wellenhof und Kienast 2010: Kapitel 4.3.2.) [10]. Die φ-Koordinate werden mit nachfolgenden Formeln

$$\begin{aligned}
a_1 &= \frac{t_f}{2N_f^2}(-1-\eta_f^2)y^2 \text{ [ ]} \\
a_2 &= \frac{t_f}{24N_f^4}(5+3t_f^2+6\eta_f^2-3\eta_f^4-9t_f^2\eta_f^4)y^4 \text{ [ ]} \\
a_3 &= \frac{t_f}{720N_f^6}(-61-90t_f^2-45t_f^4-107\eta_f^2+162t_f^2\eta_f^2+45t_f^4\eta_f^2)y^6 \text{ [ ]} \\
a_4 &= \frac{t_f}{40320N_f^8}(1385+3633t_f^2+4095t_f^4+1575t_f^6)y^8 \text{ [ ]} \\
\varphi &= \varphi_f + a_1 + a_2 + a_3 + a_4 \text{ [ ]}
\end{aligned} \tag{2.8}$$

berechnet. Die λ-Koordinate werden mit den Formeln aus der Gleichung 2.9 berechnet, wobei λ_0 die Länge des Hauptmeridianstreifens in Radiant ist.

$$
\begin{aligned}
b_1 &= \frac{1}{N_f \cos(\varphi_f)} y \ [\,] \\
b_2 &= \frac{1}{6N_f^3 \cos(\varphi_f)} (-1 - 2t_f^2 - \eta_f^2) y^3 \ [\,] \\
b_3 &= \frac{1}{120N_f^5 \cos(\varphi_f)} (5 + 28t_f^2 + 24t_f^4 + 6\eta_f^2 + 8t_f^2\eta_f^2) y^5 \ [\,] \\
b_4 &= \frac{1}{5040N_f^7 \cos(\varphi_f)} (-61 - 662t_f^2 - 1320t_f^4 - 720t_f^6) y^7 \ [\,] \\
ł &= b_1 + b_2 + b_3 + b_4 \ [\,] \\
\lambda &= l - \lambda_0 \ [\,]
\end{aligned}
\tag{2.9}
$$

Die Variablen φ und λ sind in Radiant gegeben. Diese Art der Abbildung wird Gauß-Krüger-Abbildung von der Ebene aufs Ellipsoid genannt (siehe Hofmann-Wellenhof und Kienast 2010: Kapitel 5.3.2) [10]. Mit diesen Formeln können keine Höhen aus den UTM-Koordinaten abgeleitet werden.

2.4.4. Höhen bestimmen

Bei der Transformation von UTM-Koordinaten zu ellipsoidischen Koordinaten werden nur φ und λ Werte bestimmt, nicht jedoch die Höhen. Im Algorithmus werden zunächst die Koordinaten der Trajektorien in UTM-Koordinaten transformiert und nur die Ergebnisse zum Schluss wieder in ellipsoidische Koordinaten transformiert. Die gemittelten Pisten und Lifte werden zunächst in der Ebene berechnet. Jede Piste besteht aus einer Summe von Teiltrajektorien. Für die Bestimmung der Höhen von Pisten wird aus allen Abfahrten eine Interpolationsfläche berechnet. Für jeden gemittelten Pistenpunkt wird die Höhe der Interpolationsfläche an den ausgewählten Stellen bestimmt. Lifte bestehen ebenfalls aus einer Summe von Anstiegen. Für die Bestimmung der Höhe von Liftpunkten kann nicht aus allen Anstiegen eine Interpolationsfläche bestimmt werden oder die Interpolationsfläche von Pisten verwendet werden, weil nicht jeder Lift entlang der Oberfläche verläuft, wie z.B. Luftseilbahnen. Für jeden gemittelten Lift wird eine eigene Interpolationsfläche berechnet. An den Stellen des gemittelten Liftes in der Ebene werden die Höhen der Interpolationsfläche bestimmt. Weil für die Bestimmung der Fläche für Pisten mehr Punkte als für die Interpolationsfläche für Lifte verwendet werden können, können die Höhen von Pistenpunkten genauer bestimmt

werden, als die Höhen der Liftpunkte. In Abbildung 2.5 ist die Berechnung der Höhen einer Luftseilbahn dargestellt.

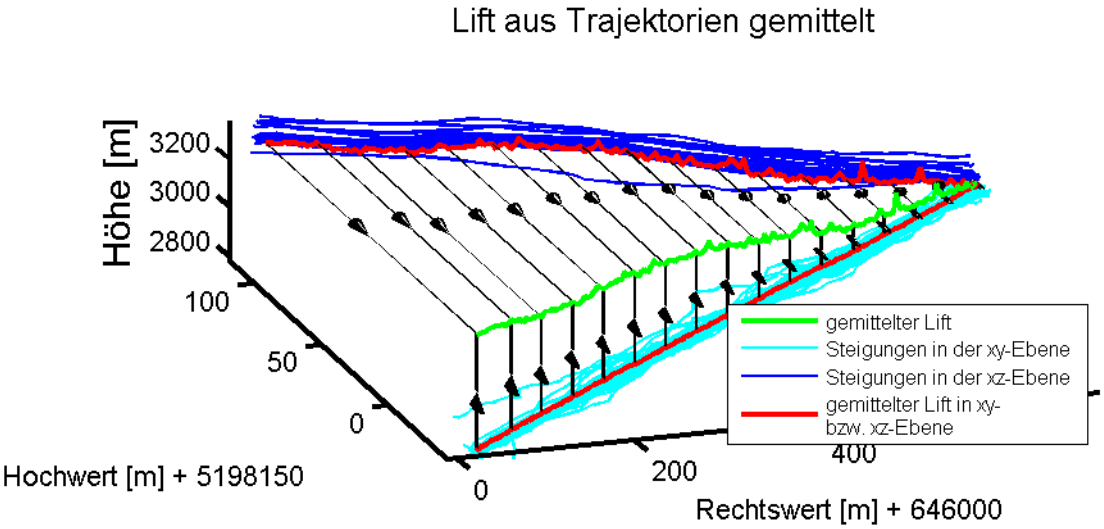


Abbildung 2.5.: Berechnung der Höhen, Quelle: eigene Darstellung

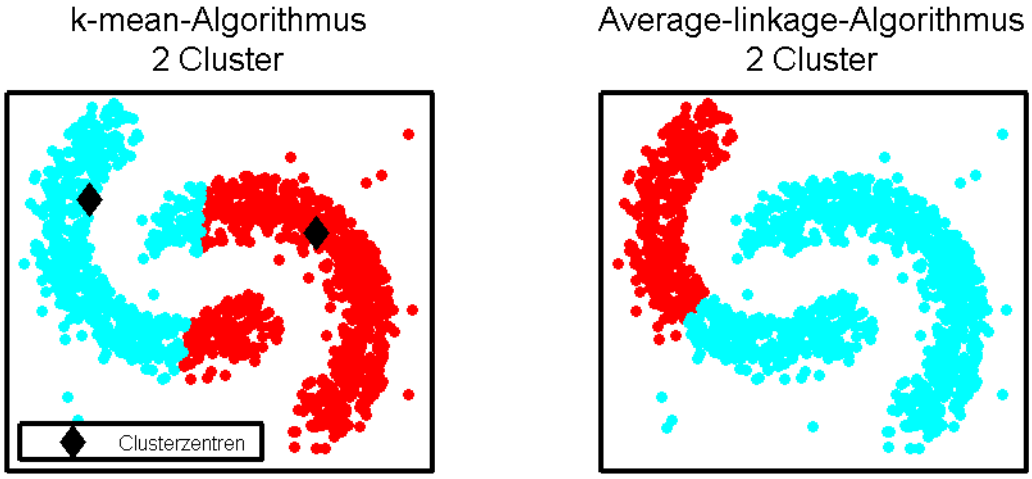
2.5. Clustering

Clustering ist die Sortierung und Ordnung von Datenpunkten zu größeren Punktmengen (Clustern). Daten, die sich in einem Cluster befinden, haben eine oder mehrere ähnliche Eigenschaften. Das Verfahren des Berechnens von Clustern wird Clusteranalyse genannt. *Data mining, the science of extracting useful knowledge from such huge data repositories, has emerged as a young and interdisciplinary field in computer science.* (siehe Fayyad et al. 2006: S. 1) [9]. Das Ziel von *data minig* ist die Suche nach Zusammenhängen und Ähnlichkeiten in großen Datenbanken. Es gibt mehr als 100 verschiedene Clusterverfahren und im Rahmen dieser Arbeit werden die Cluster aus drei Überkategorien (siehe Rokach und Maimon 2005: 330-336) [11] verglichen:

- partitionierende Clusterverfahren (siehe Abb. 2.6a) - Datenpunkte gehören einer begrenzten Anzahl von Clustern an.
- hierarchische Clusterverfahren (siehe Abb. 2.6b) - Die Daten werden geordnet und anhand von Fusionierungsmethoden zu einer Baumstruktur zusammengefügt. Jeder Knoten des Baumes ist ein Cluster.
- dichtebasierte Clusterverfahren - Datenpunkte werden nach ihrer Dichte und Nähe zu anderen Punkten geordnet. Die Cluster werden aus den geordneten Daten

extrahiert.

Nachfolgend sind die verschiedensten Verfahren näher beschrieben. Dadurch können die Vorteile und Nachteile der Verfahren näher erläutert werden. Für das Programm wird der OPTICS-Algorithmus (siehe Abschnitt 2.6) aus der Unterordnung der dichtebasierten Verfahren (siehe Unterabschnitt 2.5.3) implementiert. Der Algorithmus überzeugt durch seine simplen Eingabeparameter und die Möglichkeit, Regionen mit unterschiedlichen Punktedichten gleichmäßig gut aufzubereiten.



(a) partitionierendes Clusterverfahren **(b)** hierarchisches Clusterverfahren

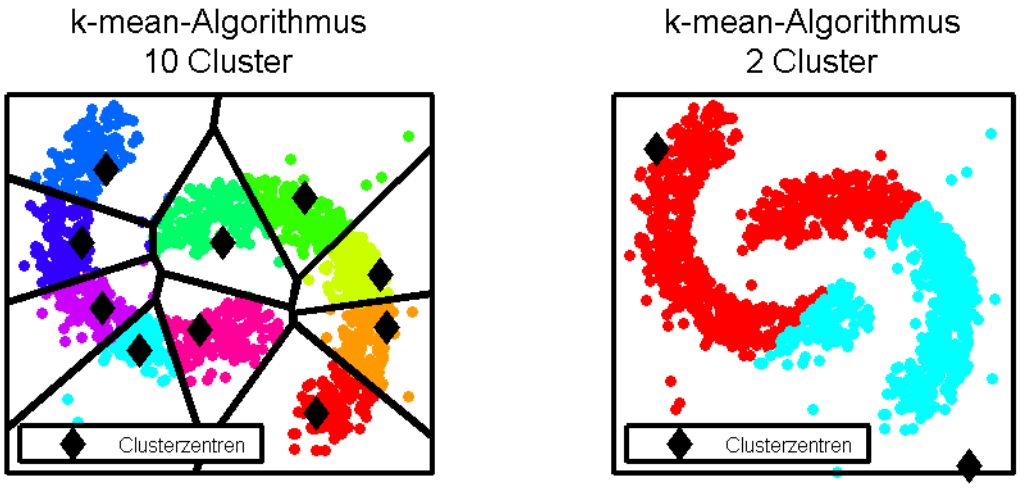
Abbildung 2.6.: unterschiedliche Clusterverfahren, Quelle: eigene Darstellung

2.5.1. Partitionierende Clusterverfahren

Bei partitionierenden Clusterverfahren muss die maximale Anzahl von Cluster pro Datensatz vorgegeben werden. Die Ausgangspunkte (Zentren) des Algorithmus sind manuell zu definieren. Für jeden Punkt werden die Distanzen zu den Clusterzentren berechnet. Der Punkt wird jenem Cluster zugewiesen, zu dem die Distanz am kürzesten ist. Das Ergebnis eines partitionierenden Clusterverfahrens ist ein Voronoi-Diagramm (siehe Abb. 2.7 - linkes Bild).

> „A centroidal Voronoi tessellation is a Voronoi tessellation whose generating points are the centroids (center od mass) of the corresponding Voronoi regions. “
>
> siehe Qiang Du et.al (1999) [7]

Die Clusterzentren sind die Schwerpunkte jeder Zelle eines Voronoi-Diagrammes.



(a) k-mean-Algorithmus und Voronoi Diagramm

(b) k-mean-Algorithmus: Clusterzentren

Abbildung 2.7.: Auswirkungen der Inputparameter, Quelle: eigene Darstellung

Die Vorteile partitionierender Verfahren liegen in der leichten Programmierbarkeit und der schnellen Berechnungszeit. Die Nachteile sind die Eingabeparameter, die pro Datensatz manuell gesetzt werden müssen. Oft gibt es kaum oder keine Informationen über die Dichte und Verteilung der Punkte eines Datensatzes. Somit ist es schwer, die Wahl der Anzahl von Cluster pro Datensatz sowie die Wahl der Clusterzentren zu automatisieren. Weiters weisen diese Arten von Algorithmen jeden Punkt des Datensatzes einem Cluster zu. Daher gibt es keine Möglichkeit, Ausreißer zu ignorieren. Partitionierende Clusteralgorithmen teilen sich in zwei Gruppen:

- k-mean-Algorithmus - weist jeden Datenpunkt anhand seiner euklidischen Distanz einem Cluster zu (siehe linkes Bild in Abb. 2.6 und Abb. 2.7). Der bekannteste k-mean-Algorithmus ist der Lloyd Algorithmus. Weitere Details sind in der Literatur angegeben (Q. Du et.al 1999: S. 637-676) [7].

- EM-Algorithmus (*Expectation-Maximization-Algorithmus*) - funktioniert ähnlich dem k-mean-Algorithmus. Berücksichtigt neben den Distanzen auch die nähere Umgebung jedes Punktes, bevor dieser einem Cluster zugewiesen wird. Nähere Details sind im Artikel von Ralf Sundberg (1974) [14] angegeben.

Im linken Bild der Abbildung 2.7 ist 10 als maximale Clusteranzahl gewählt. Die Ausgangspunkte sind zufällig gewählt. Je nach Datendichte muss die maximale Clusteranzahl angepasst und die Zentren neu bestimmt werden. Dieser Umstand macht das Automatisieren des Clusterverfahrens unmöglich. Auch wenn die Anzahl der Cluster

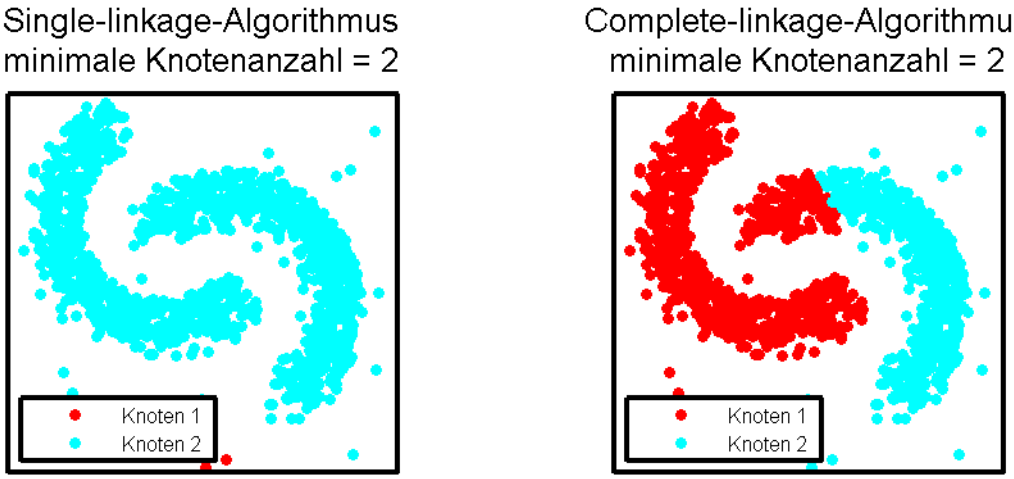
sinnvoll zu ermitteln wäre, ist auch die Wahl der Zentren entscheidend, welcher Punkt welchem Cluster zugewiesen wird (vergleiche linkes Bild in Abb. 2.6 und rechtes Bild in Abb. 2.7).

2.5.2. Hierarchische Clusterverfahren

Anders als bei partitionierenden Clusterverfahren werden die Daten bei hierarchischen Clusterverfahren nicht Clustern zugewiesen, sondern zunächst nur geordnet. Da die Anzahl der Cluster nicht vorgegeben werden muss, sind hierarchische Verfahren flexibler als partitionierende Verfahren. Die geordneten Daten werden mit der Hilfe von Fusionierungsmethoden zu Knoten zusammengefügt und in einem Baum strukturiert. Bei der Erstellung eines Baumes können zwei Verfahren angewendet werden

- diversive Verfahren durchlaufen und ordnen den Datensatz vom Groben ins Detail
- agglomerative Verfahren betrachten jeden Punkt zunächst als einzelnen Knoten. Die Knoten werden schrittweise zusammengefügt (bottom-up).

Die Ordnung der Punkte zu Knoten bzw. die Ordnung der Knoten erfolgt über Distanzmaße, die je nach Datentyp angepasst werden. Bei den Datensätzen, die für diese Arbeit verwendet werden, wird als Distanzmaß die euklidische Distanz gewählt.



(a) Single-linkage-Algorithmus

(b) Complete-linkage-Algorithmus

Abbildung 2.8.: hierarchische Clusterverfahren, Quelle: eigene Darstellung

Beim agglomerativen Verfahren wird zu Beginn jeder Punkt einem eigenen Knoten zugewiesen. Im Laufe des Verfahrens werden immer wieder Knoten zusammengefügt,

sodass ein Knoten Punkte von mehreren kleineren Knoten beinhalten kann. Als erster Schritt werden die Distanzen von allen Punkten eines Knotens zu den Punkten aller anderen Knoten berechnet. Als nächstes wird nach dem Knotenpaar mit der kürzesten Distanz gesucht. Bei der Bestimmung der Distanzen zwischen zwei Knoten unterscheiden sich die unterschiedlichen hierarchischen Clusterverfahren. Beim Single-linkage-Algorithmus wird als Distanz zwischen zwei Knoten die kürzeste Distanz zwischen zwei Punkten der jeweiligen Knoten definiert. Der Single-linkage-Algorithmus wurde 1972 von R. Sibson als SLINK [13] vorgestellt. Beim Complete-linkage-Algorithmus wird als Distanz zwischen zwei Knoten die größte Distanz zwischen zwei Punkten der jeweiligen Knoten definiert. Pro Durchlauf wird das Knotenpaar mit der kürzesten Distanz zusammengefügt. Dieser Schritt wird so lange wiederholt, bis alle Punkte in einem einzelnen Knoten zusammengefügt sind. In Anlehnung an den SLINK-Algorithmus wurde der Complete-linkage-Algorithmus 1977 von D. Defays als CLINK-Algorithmus [4] vorgestellt. Beim Single-linkage-Algorithmus besteht die Gefahr, dass Cluster nur auf Grund einzelner Punktepaare zusammengefügt werden, weil nur anhand eines einzelnen Punktepaares die Zuordnung getroffen wird. Die restlichen Punkte haben keinen Einfluss. Dieser Effekt wird Ketten-Phänomen genannt. Beim Complete-linkage-Algorithmus besteht die Gefahr, dass kompakte Cluster zwar gefunden werden, die gefundenen Cluster aber nicht die Dichte des Datensatzes widerspiegeln. Das Ergebnis eines hierarchischen Clusterverfahrens kann als Dendrogramm dargestellt werden (siehe Abb. 2.9).

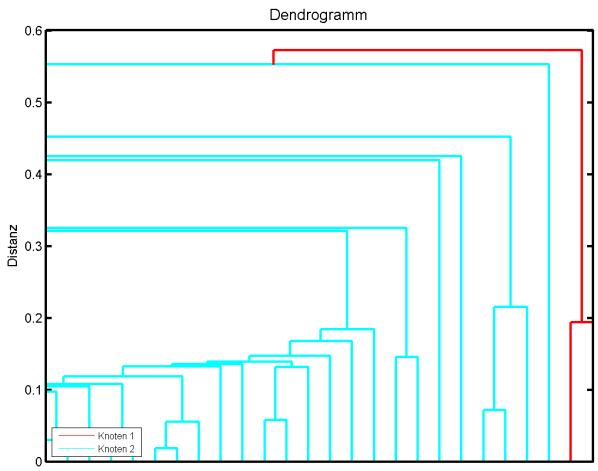


Abbildung 2.9.: Dendrogramm - Ergebnis des Single-linkage-Algorithmus von Abbildung 2.8 - linkes Bild, Quelle: eigene Darstellung

Der oberste Punkt eines Dendrogrammes bildet die Wurzel des Datensatzes und beinhaltet alle Datenpunkte. Jede Aufteilung spaltet den Datensatz. In den Kindknoten sind nur mehr Teile der Daten vorhanden. Diese Aufteilung kann so lange fortgeführt werden, bis jeder Datenpunkt seinen eigenen Knoten hat. Aus dem Dendrogramm können die Cluster auf zwei Arten generiert werden:

- Die Knoten werden anhand ihrer Distanzen einem Cluster zugewiesen - die y-Achse in Abbildung 2.9 zeigt pro Knoten die Distanz zum räumlich nächsten Knoten an. Je höher die Distanz ist, desto weiter entfernt sind die beiden Knoten voneinander. Knoten, die innerhalb einer festgelegten Maximaldistanz liegen, werden zu Clustern zusammengefasst.

- Es wird eine Minimalanzahl von Punkten definiert und Knoten so lange zusammengefügt, bis die Summe der Punkte der zusammengefügten Knoten die minimale Punkteanzahl erreicht. Die Punkte der Knoten werden zu Clustern zusammengefasst.

2.5.3. Dichtebasierte Clusterverfahren

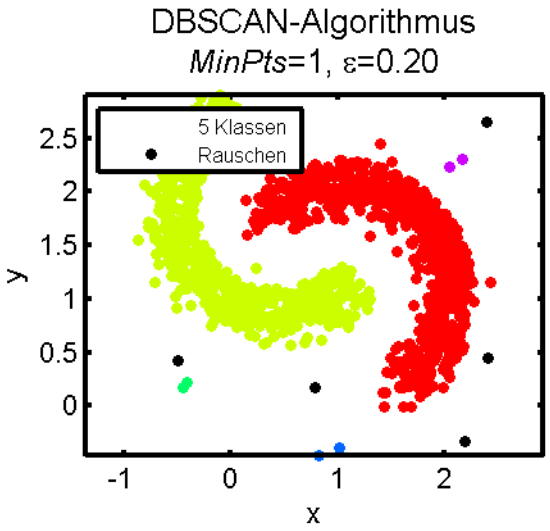


Abbildung 2.10.: DBSCAN-Algorithmus, Quelle: eigene Darstellung

Dichtebasierte Clusterverfahren ordnen Datenpunkte nach ihrer Dichte und Nähe zu anderen Punkten. Dadurch ergeben sich Teilmengen mit nah beieinanderliegenden Punkten. Der bekannteste Vertreter aus diesem Bereich der Clusterverfahren ist der DBSCAN-Algorithmus (siehe Ester et al. 1996) [8]. Das Akronym DBSCAN

steht für „**D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise“ was so viel bedeutet wie dichtebasiertes, räumliches Clusterverfahren für Datensätze mit Rauschen. Die Erweiterung des DBSCAN-Algorithmus ist der OPTICS-Algorithmus (siehe Ankerst et al. 1999) [2]. Der größte Unterschied zu den hierarchischen Algorithmen ist, dass keine Fusionierungsmethoden benötigt werden und somit die Wahl der Clusteranzahl und Clusterzentren entfällt. Beim DBSCAN-Algorithmus müssen 2 Parameter (*MinPts,* ϵ) vom User vorgegeben werden. In Abbildung 2.10 sieht man das Ergebnis eines DBSCAN-Algorithmus. Beim OPTICS-Algorithmus sind ebenfalls zwei Eingabeparameter zu wählen, die jedoch sehr allgemein definiert sind. Dies ermöglicht eine Automatisierung dieses Clusterverfahrens. Der OPTICS-Algorithmus bildet das Kernstück der vorliegenden Arbeit (siehe Abschnitt 2.6).

2.5.4. Kombinierte Verfahren

Die hierarchischen und dichtebasierten Clusterverfahren sind sehr ähnliche Verfahren, die auch kombiniert werden können. Dadurch ergeben sich unterschiedliche Varianten, die Verfahren schneller und genauer machen. Der BIRCH-Algorithmus (Balanced iterative reducing and clustering using hierarchies) ist ein Vertreter der kombinierten Verfahren.

> „BIRCH incrementally and dynamically clusters incoming multi-dimensioned metric data points to try to produce the best quality clustering with the available resources (i.e. available memory and time constraints) “
> siehe Zhang et.al (1996) [15]

Bei sehr großen Datenmengen kann es bei Clusterverfahren zu Überladungen des Arbeitsspeichers kommen. Um dies zu verhindern, strukturiert der BIRCH-Algorithmus die Daten in einem balancierten Baum. Ein balancierter Baum hat eine zu definierende Maximalhöhe und einen Maximalwert von Einträgen pro Knoten. Der Maximalwert entspricht z.B. dem maximalen Arbeitsspeicher. Die Knoten werden so erstellt, dass ein Schwellenwert um einen Startpunkt gelegt wird, und alle Punkte innerhalb des Radius dem Startknoten zugewiesen werden. Sollte der Knoten größer als der Platz im Arbeitsspeicher sein, definieren die beiden am weitesten voneinander entfernten Punkte im Knoten zwei neue Knoten. Auf diese zwei Knoten werden die Daten des aktuellen Knotens verteilt. Zum Schluss ergibt sich ein ausbalancierter Baum und jeder Knoten des Baumes hat seinen eigenständigen Datensatz. Die eigenständigen Datensätze können mit beliebigen Clusterverfahren untersucht werden. Je höher der Baum ist, desto kleiner sind die Datenmengen pro Knoten. Durch die Aufteilung der Daten auf Knoten ist nicht gewährleistet, dass alle Informationen extrahiert werden.

2.6. OPTICS-Algorithmus

In diesem Abschnitt wird der OPTICS-Algorithmus (**O**rdering **P**oints **T**o **I**dentify the **C**lustering **S**tructure) im Detail erklärt. Zunächst werden einige Definitionen und danach der technische Code beschrieben. Der Pseudocode ist in voller Länge angeführt, da er an einigen Stellen gegenüber dem Algorithmus, wie er in der Literatur (siehe Ankerst 1999) [2] beschrieben steht, verändert wurde. Die Kurzbeschreibung des Algorithmus kann als Ordnung von Punkten unterschiedlicher Schifahrer zu Clustern, die eine räumliche Nähe aufweisen, zusammengefasst werden. Im Unterabschnitt 2.6.2 sind die Eingabeparameter für den OPTICS-Algorithmus angeführt. Die restlichen Unterabschnitte bilden das technische Grundgerüst, um die Punkte eines Datensatzes zu ordnen und die entsprechenden Cluster zu extrahieren. Aus den extrahierten Clustern werden im Anschluss Lifte berechnet (siehe Abschnitt 2.7).
Der OPTICS-Algorithmus baut auf dem DBSCAN-Algorithmus (siehe Ester et al. 1996) [8] auf. Daher wird auf den DBSCAN-Algorithmus nicht näher eingegangen, sondern beim Erläutern des OPTICS-Algorithmus auf Ähnlichkeiten verwiesen.

Definition eines Kernpunktes und der ϵ-Nachbarschaft

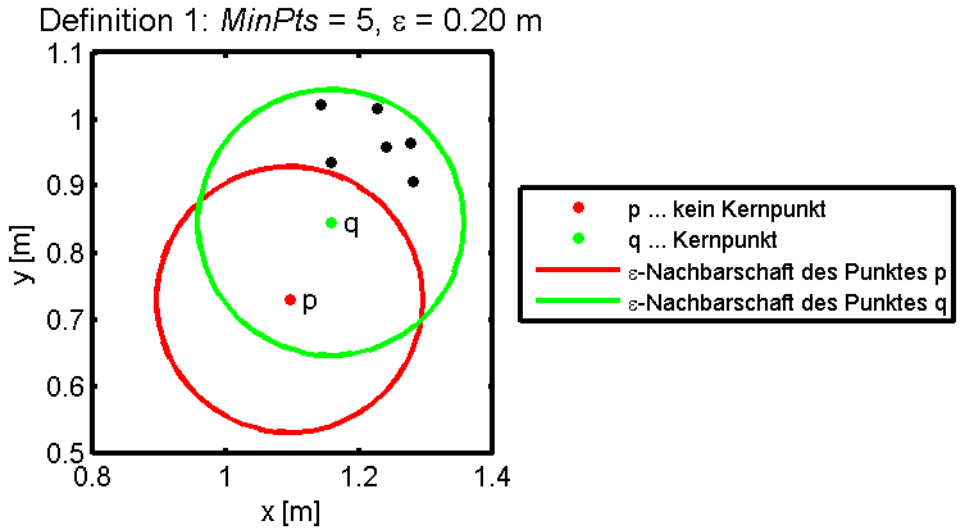


Abbildung 2.11.: Überblicksgrafik für die Definition 1, Quelle: eigene Darstellung

„**Definition 1:** (directly density-reachable)
Object p is directly density-reachable from object q wrt. ϵ and *MinPts* in a

set of objects D if

1.) $p \in N_\epsilon(q)$ ($N_\epsilon(q)$ is the subject of D contained in the ϵ-neighbourhood of q)
2.) $Card(N_\epsilon(q)) \geq MinPts$ ($Card(N)$ denotes the cardinality of the set N)“

siehe Ankerst (1999: Kapitel 3-2) [2]

Zwei Punkte (p und q) sind direkt über die Dichte verbunden, falls sie zwei Teilbedingungen erfüllen. Die erste Teilbedingung besagt, dass ein Punkt p in der ϵ-Nachbarschaft von Punkt q sein muss. ϵ ist der Radius eines Kreises, der um den Punkt q gelegt wird. Alle Punkte, die sich innerhalb dieses Kreises befinden, gehören zur $N_\epsilon(q)$-Nachbarschaft. Die zweite Teilbedingung ist die Definition von Kernpunkten. Ein Kernpunkt hat in seiner ϵ-Nachbarschaft zumindest *MinPts*-Nachbarpunkte. In Abbildung 2.11 ist der Punkt p vom Punkt q direkt über die Dichte erreichbar, da der Punkt q den Punkt p in seiner Nachbarschaft und zugleich mehr als *MinPts*-Nachbarn(7) hat. Der Punkt q ist nicht direkt vom Punkt p über die Dichte erreichbar. Der Punkt q befindet sich zwar in der ϵ-Nachbarschaft des Punktes p, dieser hat aber zu wenige Punkte in seiner ϵ-Nachbarschaft(1) um die Bedingungen eines Kernpunktes zu erfüllen. Alle Nachbarpunkte des Kernpunktes q sind von diesem Punkt aus direkt über die Dichte erreichbar.

Erreichbarkeit über die Dichte

„**Definition 2:** (density-reachable)
An object p is *density-reachable* from an object q wrt. ϵ and *MinPts* in the set of objects D if there is a chain of objects p_1,...,p_n, $p_1 = q$, $p_n = p$ such that $p_i \in D$ and p_{i+1} is directly density-reachable from p_i wrt. ϵ and *MinPts*“
siehe Ankerst (1999: Kapitel 3-2) [2]

Zwei Punkte, die nicht gegenseitig in der ϵ-Nachbarschaft des jeweils anderen liegen, sind über die Dichte erreichbar (*density-reachable*), falls sie über einen oder mehrere andere Kernpunkte verbunden sind. Bis auf den Zielpunkt müssen alle Punkte einer solchen Kette Kernpunkte sein. In Abbildung 2.12 ist der Punkt p vom Punkt q aus über die Dichte erreichbar. Der Punkt p ist über die Dichte nicht vom Punkt q erreichbar, da der Punkt q kein Kernpunkt ist.

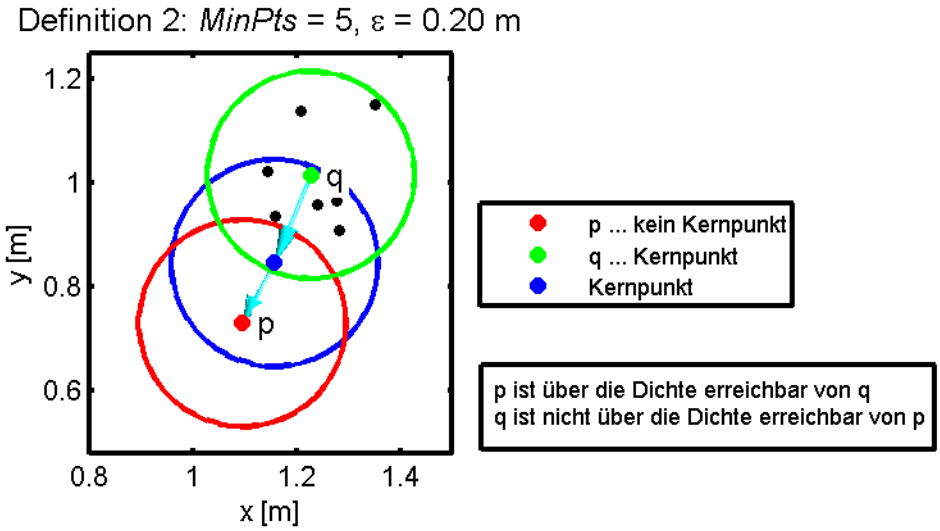


Abbildung 2.12.: Überblicksgrafik für die Definition 2, Quelle: eigene Darstellung

Verbundenheit über die Dichte

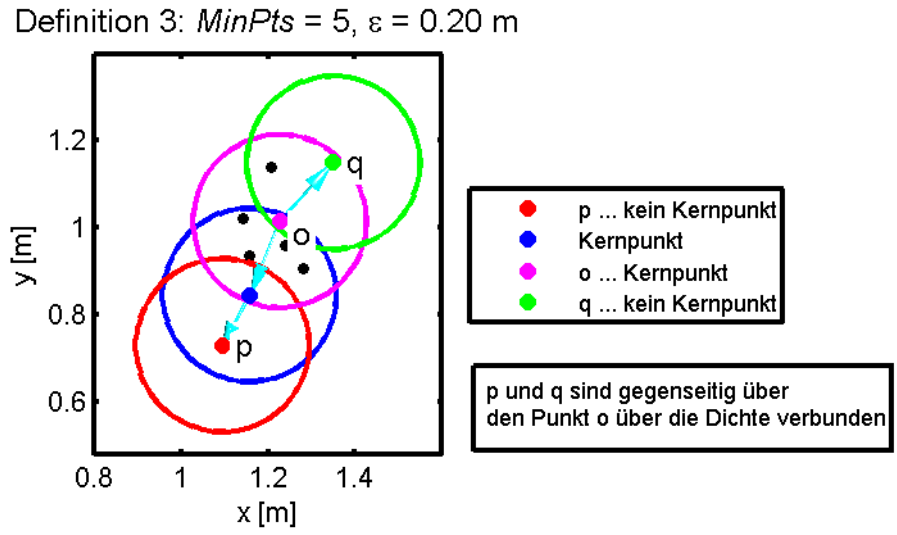


Abbildung 2.13.: Überblicksgrafik für die Definition 3, Quelle: eigene Darstellung

„**Definition 3:** (density-connected)
Object p is *density-connected* to object q wrt. ϵ and *MinPts* in the set of objects D if there is an object $o \in D$ such that both p and q are density-reachable from o wrt. ϵ and *MinPts* in D“
siehe Ankerst (1999: Kapitel 3-2) [2]

Zwei Punkt (p und q) sind über die Dichte verbunden (density connected), falls beide Punkte von einem Punkt o aus über die Dichte erreichbar sind (siehe Abb. 2.13).

Cluster und Rauschen

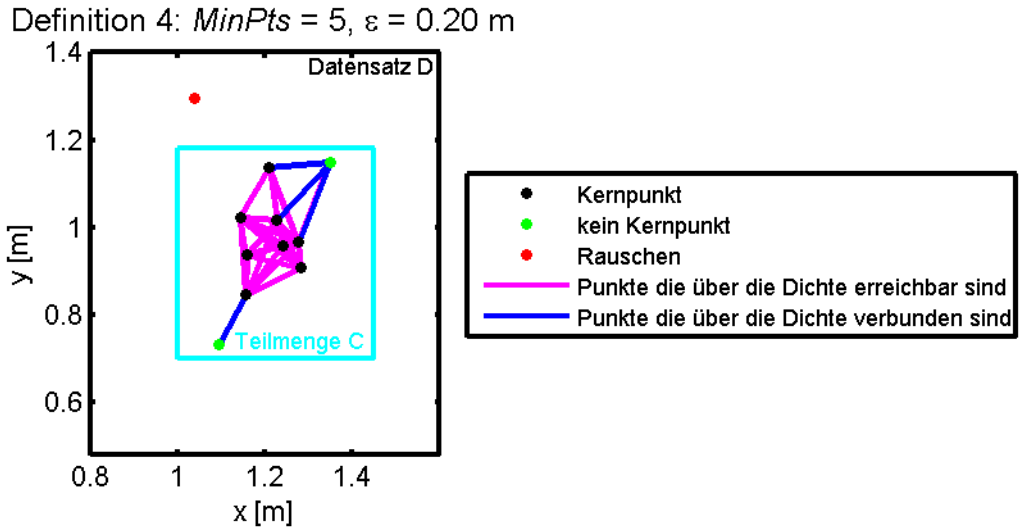


Abbildung 2.14.: Überblicksgrafik für die Definition 4, Quelle: eigene Darstellung

„**Definition 4:** (cluster and noise)
Let D be a set of objects. A *cluster* C wrt. ϵ and $MinPts$ in D is a non-empty subset of D satisfying the following conditions:

1) Maximality :$\forall p, q \in D:$ if $p \in C$ and q is density-reachable from p wrt. ϵ and $MinPts$, then also $q \in C$

2) Connectivity : $\forall p, q \in C:$ p is density-connected to q wrt. ϵ and $MinPts$ in D.

Every object not contained in any cluster is *noise*.“

siehe Ankerst (1999: Kapitel 3-2) [2]

Eine Teilmenge C ist eine nicht leere Teilmenge des Datensatzes D falls die Bedingungen *Maximality* und *Connectivity* gegeben sind. Die erste Teilbedingung besagt, dass alle Punkte, die über die Dichte von einem Punkt aus der Teilmenge C des Datensatzes D erreichbar sind, ebenfalls Punkte der Teilmenge C sind. Die zweite Teilbedingung besagt, dass alle Punkte in einem Cluster C über die Dichte verbunden sind. Alle anderen Punkte im Datensatz D, die keiner Teilmenge angehören, sind Rauschen. Alle Punkte, die in

der ϵ-Nachbarschaft eines Kernobjektes liegen, gehören automatisch einer Teilmenge an. In Abbildung 2.14 ist eine Teilmenge C des Datensatzes D dargestellt. Alle Punkte der Teilmenge C sind über die Kernpunkte miteinander verbunden (siehe Definition 3 S. 29). Alle schwarzen Punkte sind untereinander über die Dichte erreichbar (siehe Definition 2 S. 28).
Beim DBSCAN-Algorithmus werden am Anfang die ϵ- und *MinPts*-Parameter festgesetzt und die Punkte anhand der Eingabeparameter Teilmengen zugewiesen. Eine Teilmenge besitzt immer Kernpunkte. Alle Punkte einer Teilmenge sind über die Dichte von einem Kernpunkt aus erreichbar (siehe Definition 2 S. 28). Teilmengen können Randpunkte haben. Randpunkte sind in der ϵ-Nachbarschaft eines Kernpunktes und sind mit jedem anderen Punkt einer Teilmenge über die Dichte verbunden (siehe Definition 3 S. 29). Beim DBSCAN-Algorithmus werden die Punkte anhand des ϵ-Parameters Teilmengen zugewiesen. Dabei wird die unterschiedliche Punktedichte von Regionen nicht berücksichtigt.
Der OPTICS-Algorithmus verlangt als Eingabeparameter ebenfalls einen *MinPts*- und ϵ-Parameter. Die Daten werden nicht Teilmengen zugewiesen, sondern nur geordnet. Erst im Anschluss werden aus den geordneten Daten Teilmengen extrahiert (siehe Unterabschnitt 2.6.3). Die Wahl der Eingabeparameter für den OPTICS-Algorithmus ist im Unterabschnitt 2.6.2 genauer erläutert. Nachfolgende Definitionen sind die Erweiterungen des DBSCAN-Algorithmus zum OPTICS-Algorithmus.

Definition einer Kerndistanz

„**Definition 5:** (core-distance of an object p)
Let p be an object from a database D, let ϵ be a distance value, let $N_\epsilon(p)$ be the ϵ-neighbourhood of p, let *MinPts* be a natural number and let *MinPts-distance(p)* be the distance from p to its *MinPts*' neighbour. Then, the *core-distance* of p is defined as $\textit{core-distance}_{\epsilon,MinPts}(p)=$

$$\begin{cases} \text{UNDEFINED, if } Card(N_\epsilon(p)) < MinPts \\ MinPts - distance(p), \text{ otherwise“} \end{cases}$$

siehe Ankerst (1999: Kapitel 3-2) [2]
Wie bereits in Definition 1 erwähnt, ist jeder Punkt ein Kernpunkt, der mehr als *MinPts*-Nachbarpunkte in seiner ϵ-Nachbarschaft hat. Ein solcher Punkt erhält eine Kerndistanz zugewiesen. Die Kerndistanz ist die kürzeste Distanz vom Kernpunkt zu

einem seiner Nachbarpunkte.

Definition einer Erreichbarkeitsdistanz

„**Definition 6:** (reachability-distance object p w.r.t. object o)
Let p and o be objects from a database D, let $N_\epsilon(o)$ be the ϵ-neighborhood of o, and let *MinPts* be a natural number. Then, the *reachability-distance* of p w.r.t. o is defined as $\textit{reachability-distance}_{\epsilon,MinPts}(p,o) =$

$$\begin{cases} \text{UNDEFINED, if } |N_\epsilon(o)| < MinPts \\ max(\textit{core-distance}(o), \textit{distance}(o,p)), \text{ otherwise}\text{“} \end{cases}$$

siehe Ankerst (1999: Kapitel 3-2) [2]

Ein Punkt p erhält eine Erreichbarkeitsdistanz falls er sich in der ϵ-Nachbarschaft eines Kernpunktes o befindet. Die Erreichbarkeitsdistanz des Punktes p ist der größere Wert der Kerndistanz von Punkt o oder der Distanz zwischen den Punkten p und o.

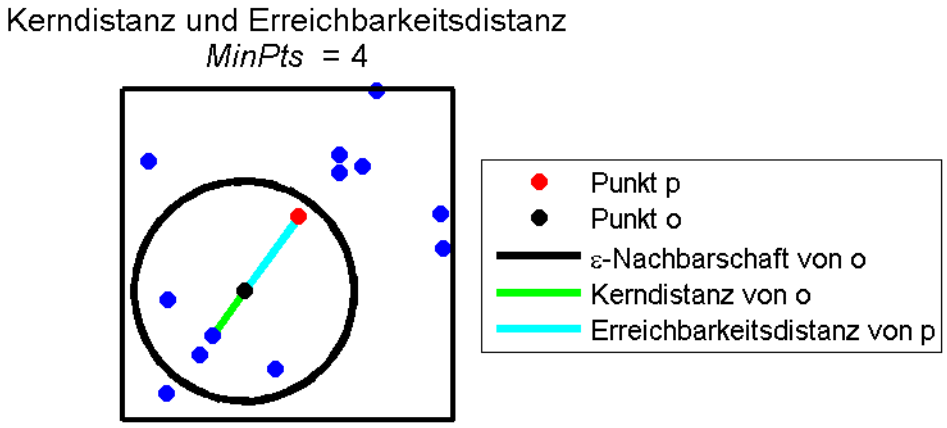


Abbildung 2.15.: Kern- und Erreichbarkeitsdistanz, Quelle: eigene Darstellung

In Abbildung 2.16 hat der Punkt o genügend Nachbarpunkte in seiner ϵ-Nachbarschaft um die Bedingung eines Kernpunktes zu erfüllen. Die Kerndistanz ist die kürzeste Distanz von Punkt o zum nahesten Nachbarpunkt. Die Erreichbarkeitssdistanz des Punktes p ist die Distanz von Punkt p zu Punkt o, da diese Distanz größer als die Kerndistanz des Punktes o ist.

2.6.1. Pseudocode

Der Algorithmus, wie er in der Theorie beschrieben steht (siehe Ankerst 1999) [2], ist für die Aufgabenstellung dieser Arbeit adaptiert. Der größte Unterschied besteht darin, dass die Berechnung der Nachbarpunkte, bzw. die Bestimmung der Distanzen zwischen allen Punkten in Form der Funktion *CalculateDistances*, ausgelagert ist. Als Eingabeparameter benötigt die Funktion *CalculateDistances* einen Datensatz von Punkten *SetOfObjects* und einen Parameter ϵ, welcher den Nachbarschaftsradius definiert.

```
CalculateDistances(SetOfObjects,ε)
    FOR i FROM 1 TO SetOfObjects.size
        d = {dist(p(i), p)|∀ p ∈ SetOfObjects}
        d = d − d(ii)
        d = {d|d > ε}
        DistFile[i].open
        DistFile[i].write(d)
        DistFile[i].close
    END % FOR
END % CalculateDistances
```

Für jeden Datenpunkt werden die Distanzen zu allen anderen Punkten im Datensatz berechnet. Dieser Schritt ist die erste Teilbedingung der Definition 1 (siehe S. 27) zur Bestimmung von Kernpunkten. Alle Punkte, die eine Distanz kleiner als ϵ haben, werden in das Textfile des aktuellen Punktes geschrieben. In der ersten Spalte des Textfiles steht pro Zeile der Index eines Nachbarpunktes und in der zweiten Spalte steht die berechnete Distanz. Weil die Berechnung der Distanzen aus dem Hauptteil des OPTICS-Algorithmus herausgenommen ist, wird der ϵ-Parameter nur in dieser Funktion benötigt. Aus diesem Grund können vor dem eigentlichen OPTICS-Algorithmus die Distanzen für unterschiedliche ϵ-Werte berechnet und die dazugehörigen *DistFiles* erstellt werden. Ohne eine räumliche Einteilung der Daten (spatial indexing) ist die Komplexität für diese Funktion $\mathcal{O}(n^2)$, weil die Distanz von jedem Punkt zu jedem Punkt bestimmt wird (*region query*).

> „Without any index support, to answer such a region query, a scan through the whole database has to be performed. If a tree-based spatial index can be used, the run-time is reduced to $\mathcal{O}(n \log n)$ since region queries are supported

efficiently by spatial access methods. “
siehe Ankerst (1999: Kapitel 3-2) [2]

Weil die Berechnung der Distanzen aus dem Hauptteil des OPTICS-Algorithmus herausgenommen ist, sehen die anschließenden Funktionen anders als der originale Algorithmus aus. Die geänderten Codezeilen sind mit einem Stern versehen und die Unterschiede im Anschluss an die Funktion kurz erklärt. Der OPTICS-Algorithmus beginnt mit der Funktion *OPTICS*. Die Eingabeparameter sind ein Datensatz von Punkten *SetOfObjects*, eine Minimalanzahl *MinPts* von Nachbarn, die für die Definition von Kernpunkten benötigt wird und ein Textfile *OrderedFile*, in welches die geordneten Punkte der Reihe nach geschrieben werden. Das Textfile ist das Endergebnis des OPTICS-Algorithmus. Weil die Berechnung der Distanzen aus dem Hauptteil herausgenommen ist, wird kein ϵ-Parameter als Input benötigt:

```
OPTICS(SetOfObjects, MinPts , OrderedFile)
  OrderedFile.open()
  FOR i FROM 1 TO SetOfObjects.size
    Object=SetOfObjects.get(i);
    IF NOT Object.processed THEN
      SetOfObjects=ExpandClusterOrder(SetOfObjects,Object,MinPts ,OrderedFile)*
    END % IF
  END % FOR
  OrderedFile.close()
END % OPTICS
```

siehe Ankerst (1999: Kapitel 3-2) [2]
**... es wird kein ϵ-Parameter an die Funktion übergeben*
Jeder Punkt ist ein Objekt mit mehreren Eigenschaften:

- *processed*, die Eigenschaft ist *FALSE*, falls der Punkt noch nicht besucht ist und *TRUE*, falls der Punkt bereits besucht ist
- *index*, jedes Objekt hat seinen eigenen Index. Der Index wird im Initialisierungsschritt zugewiesen.
- *core_distance* ist die Kerndistanz. Diese ist bei der Initialisierung der Objekte *nan* und wird im Laufe der Funktion *ExpandClusterOrder* befüllt.
- *reachability_distance* ist die Erreichbarkeitsdistanz. Diese ist bei der Initialisierung der Objekte *nan* und wird im Laufe der Funktion *Orderseeds.update* befüllt.

- *neighbours* sind die Indizes der Nachbarpunkte in der ϵ-Nachbarschaft des ausgewählten Objektes

Im Initialisierungsschritt wird jeder Punkt als noch nicht besucht initialisiert. In der Funktion *OPTICS* wird jeder Punkt des Datensatzes durchlaufen und falls ein Punkt noch nicht besucht ist, wird er an die Funktion *ExpandClusterOrder* übergeben. Diese Funktion bildet die Hauptfunktion des OPTICS-Algorithmus. Die Eingabeparameter der Funktion sind der Datensatz *SetOfObjects*, ein Punkt *Object* aus dem Datensatz, der noch nicht besucht ist, die Variable *MinPts* und das Textfile *OrderedFile*.

```
SetOfObjects=ExpandClusterOrder(SetOfObjects, Object, MinPts, OrderedFile)*
    neighbours=SetOfObjects.neighbours(Object)**
    Object.processed=TRUE
    Object.reachability_distance=UNDEFINED
    Object.setCoreDistance(neighbours, MinPts)
    OrderedFile.write(Object)
    IF Object.core_distance <> UNDEFINED THEN
        OrderSeeds.update(neighbours, Object)
        WHILE NOT OrderSeeds.next() DO
            currentObject=OrderSeeds.next()
            neighbours=SetOfObjects.neighbours(currentObject)**
            currentObject.processed=TRUE
            currentObject.setCoreDistance(neighbours, MinPts)
            OrderedFile.write(currentObject)
            IF currentObject.core_distance <> UNDEFINED THEN
                OrderSeeds.update(neighbours, currentObject)
            END % IF
        END % WHILE
    END % IF
END % ExpandClusterOrder
```

siehe Ankerst (1999: Kapitel 3-2) [2]
**... es wird kein ϵ-Parameter an die Funktion übergeben*
***... für die Bestimmung der Nachbarpunkte wird kein ϵ-Parameter benötigt, sondern nur jener Punkt, für den die Nachbarpunkte bestimmt werden*
In der Funktion *expandClusterOrder* werden zunächst die Nachbarpunkte eines ausgewählten Punktes mit der Funktion *SetOfObjects.neighbours* bestimmt.

```
N=SetOfObjects::neighbours(Object)
    fid=Object.open(Object.index)
    N=fid.read()
    fid.close()
END % SetOfObjects::neighbours
```

Da die Berechnungen der Distanzen ausgelagert ist, werden in dieser Funktion nicht die Nachbarpunkte bestimmt, sondern nur das *DistFile* des jeweiligen Punktes ausgelesen und als Objekt N zurückgegeben. Das Objekt N hat zwei Eigenschaften:

- *index* - Ids der Nachbarpunkte im Datensatz
- *dist* - Distanzen des Objektes zu seinen Nachbarpunkten

Als nächstes wird in der Funktion *ExpandClusterOrder* die Eigenschaft *processed* des Punktes auf *TRUE* und die Erreichbarkeitsdistanz auf *UNDEFINED* gesetzt. Die Kerndistanz wird in der Funktion *Object.setCoreDistance* bestimmt. Als Eingabeparameter werden die Nachbarn N und die Variable *MinPts* übergeben.

```
core_dist=Object::setCoreDistance(neighbours,MinPts)
    core_dist=nan
    IF neighbours.size>=MinPts
        core_dist=min(neighbours.dist)
    END % IF
END % Object::setCoreDistance
```

siehe Ankerst (1999: Kapitel 3-2) [2]
Falls der Punkt genügend viele Nachbarpunkte besitzt, erhält er eine Kerndistanz und erfüllt somit die Bedingung der Kardinalität, die als zweite Teilbedingung eines Kernpunktes gilt (siehe Definition 1 S. 28). Der Punkt wird in das File *OrderedFile* geschrieben. Falls der Punkt kein Kernpunkt ist, wird die Funktion abgebrochen und ein neuer, noch nicht besuchter Punkt, wird von der Funktion *OPTICS* in die Funktion *ExpandClusterOrder* übergeben. Punkte, die an dieser Stelle keine Kerndistanz erhalten, sind Rauschen (siehe Definition 4 S. 30). Punkte, die an dieser Stelle eine Kerndistanz erhalten, sind die Anfangspunkte einer Teilmenge im Datensatz. Falls eine Kerndistanz gegeben ist, wird die Funktion *OrderSeeds.update* aufgerufen. Die Eingabeparameter sind der aktuelle Punkt und die Nachbarpunkte des aktuellen Punktes.

```
OrderSeeds=Orderseeds::update(neighbours, CenterObject)
    c_dist=CenterObject.core_dist
    FORALL Object FROM neighbours DO
        IF NOT Object.processed THEN
            new_r_dist=max(c_dist,CenterObject.dist(Object))
            IF Object.reachability_distance=UNDEFINED THEN
                Object.reachability_distance=new_r_distance
                //Erweiterung der Liste OrderSeeds um einen Punkt
                insert(Object, new_r_dist)
            ELSE
                IF new_r_dist<Object.reachability_distance THEN
                    Object.reachability_distance=new_r_dist
                    //Verkleinerung der Erreichbarkeitsdistanz eines
                    // Punktes in der Liste OrderSeeds
                    decrease(Object, new_r_dist)
                END % IF
            END % IF
        END % IF
    END % FOR
END % Orderseeds::update
```

siehe Ankerst (1999: Kapitel 3-2) [2]
In der Funktion wird zunächst die Kerndistanz des aktuellen Punktes ausgelesen. Danach werden alle Nachbarpunkte des aktuellen Punktes durchlaufen. Nachbarpunkte, die noch nicht besucht sind, erhalten eine temporäre Erreichbarkeitsdistanz. Falls der Nachbarpunkt bereits eine temporäre Erreichbarkeitsdistanz hat und diese größer als die aktuelle temporäre Erreichbarkeitsdistanz ist, wird für diesen Punkt die aktuelle temporäre Erreichbarkeitsdistanz übernommen. Alle Nachbarpunkte, die noch nicht besucht sind, werden in der Liste *OrderSeeds* vermerkt. In der ersten Spalte der Liste stehen die Indizes der noch nicht besuchten Nachbarpunkte und in der zweiten Spalte stehen deren Erreichbarkeitsdistanzen. Falls ein Punkt noch keine Erreichbarkeitsdistanz hat, wird dieser Punkt neu in die Liste aufgenommen. Falls die Erreichbarkeitsdistanz eines Punktes verändert wird, wird auch der Eintrag in der Liste geändert. Der Rückgabeparameter der Funktion ist die Liste. Ein Minimalbeispiel für das Verständnis ist auf der Seite 38 aufgelistet.
Zurück in der Funktion *ExpandClusterOrder* wird die Liste so lange durchlaufen,

bis sie leer ist. Bei jedem Durchlauf wird aus der Liste der Punkt mit der kürzesten Erreichbarkeitsdistanz ausgewählt. Dieser wird von der Liste gestrichen, als besucht markiert, seine Kerndistanz bestimmt, ins Textfile *OrderedFile* geschrieben und gegebenenfalls die Funktion *OrderSeeds.update* aufgerufen. Punkte, die an dieser Stelle der Funktion *ExpandClusterOrder* eine Kerndistanz erhalten, sind später bei der Extraktion von Teilmengen (siehe Unterabschnitt 2.6.4), Teil einer größeren Teilmenge. Solche Punkte sind mit allen anderen Punkten einer Teilmenge über die Dichte verbunden (siehe Definition 2 S. 28). Punkte, die an dieser Stelle keine Kerndistanz erhalten, sind Randpunkte einer Teilmenge. Solche Punkte sind mit allen anderen Punkten einer Teilmenge über die Dichte verbunden (siehe Definition 3 S. 29). Die Liste wird immer wieder erweitert bzw. verändert. Sobald eine Liste geleert ist, wird in der Funktion *OPTICS* der nächste noch nicht bescuht Punkt ausgewählt und an die Funktion *ExpandClusterOrder* übergeben. Punkt, die in einer Liste auftauchen, werden anschließend bei der Extraktion von Clustern (siehe Unterabschnitt 2.6.4) als Teil einer Teilmenge detektiert. Bei einem Schigebiet werden z.B. die Daten, die einem Lift angehören hintereinander ins File *orderedFile* geschrieben. Anhand der Ordnung werden diese Daten im Unterabschnitt 2.6.4 als eine Teilmenge detektiert.
Die Erreichbarkeitsdistanz ist nicht zwangsweise die kürzeste Verbindung zum nahesten Nachbarpunkt. Einem Punkt kann eine Erreichbarkeitsdistanz nur von einem Punkt zugewiesen werden, der gerade besucht wird und die Bedingung eines Kernpunktes erfüllt. Solange ein Punkt mit einer Erreichbarkeitsdistanz selbst noch nicht besucht wurde, ist die Erreichbarkeitsdistanz temporär. Erst wenn der Punkt besucht wird, ist die Erreichbarkeitsdistanz fixiert und kann nicht mehr geändert werden. Weitere Informationen können in der Literatur nachgelesen werden (siehe Ankerst 1999) [2].

Minimalbeispiel

Das Minimalbeispiel rechnet anhand der in Abbildung 2.16 dargestellten Ausgangslage den OPTICS-Algorithmus durch. Der Anfangspunkt der Berechnung ist frei wählbar.

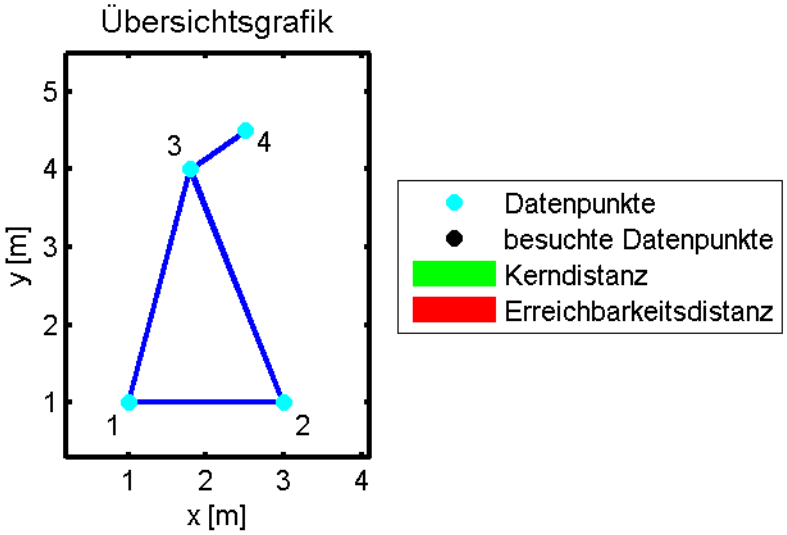


Abbildung 2.16.: Lageplan, Quelle: eigene Darstellung

Die Ausgangsdaten für die Berechnung sind in Tabelle 2.1 gegeben.

ϵ	4 [m]
MinPts	1
s_{12}	2 [m]
s_{13}	3.1 [m]
s_{14}	3.8 [m]
s_{23}	3.2 [m]
s_{24}	3.5 [m]
s_{34}	.9 [m]

Tabelle 2.1.: Ausgangsdaten des Beispieles

In der Funktion *OPTICS* wird der Punkt 1 als erster Punkt an die Funktion *ExpandClusterOrder* übergeben. In der Funktion wird die Kerndistanz bestimmt. Der Punkt 1 wird mit der Erreichbarkeitsdistanz *nan* ins File *OrderedFile* geschrieben (siehe Tabelle 2.2). Als nächstes wird die Funktion *Orderseeds.update* aufgerufen. Für die Nachbarpunkte 2, 3 und 4 wird eine temporäre Erreichbarkeitsdistanz bestimmt, da sich diese Punkte in der ϵ-Nachbarschaft des Punktes 1 befinden und noch nicht besucht sind. Die Liste *OrderSeeds* wird mit den Punkten 2, 3 und 4 befüllt (siehe Tabelle 2.3).

Punktnummer	Kerndistanz	Erreichbarkeitsdistanz
1	s_{12}	*nan*

Tabelle 2.2.: Besuchte Punkte nach Schritt 1

Punktnummer	temporäre Erreichbarkeitsdistanz
2	s_{12}
3	s_{13}
4	s_{14}

Tabelle 2.3.: Liste nach Schritt 1

Da Punkt 2 die kürzeste temporäre Erreichbarkeitsdistanz besitzt, wird dieser Punkt als nächster ausgewählt, seine Kerndistanz bestimmt und in das File *OrderedFile* geschrieben (siehe Tabelle 2.4). Ein weiteres Mal wird die Funktion *Orderseeds.update* aufgerufen. Die Erreichbarkeitsdistanz für Punkt 3 von Punkt 2 aus wird nicht übernommen, da diese größer als die vorhandene Erreichbarkeitsdistanz für Punkt 3 von Punkt 1 aus ist. Für den Punkt 4 wird die temporäre Erreichbarkeitsdistanz von Punkt 2 übernommen (siehe Tabelle 2.5).

Punktnummer	Kerndistanz	Erreichbarkeitsdistanz
1	s_{12}	*nan*
2	s_{12}	s_{12}

Tabelle 2.4.: Besuchte Punkte nach Schritt 2

Punktnummer	temporäre Erreichbarkeitsdistanz
3	s_{13}
4	s_{24}

Tabelle 2.5.: Liste nach Schritt 2

Als nächstes wird der Punkt 3 ausgewählt. Es wird die Kerndistanz zu Punkt 4 bestimmt und der Punkt 3 ins *orderedFile* geschrieben (siehe Tabelle 2.6). Für den Punkt 4 wird die Erreichbarkeitsdistanz bestimmt (siehe Tabelle 2.7).

Punktnummer	Kerndistanz	Erreichbarkeitsdistanz
1	s_{12}	*nan*
2	s_{12}	s_{12}
3	s_{34}	s_{13}

Tabelle 2.6.: Besuchte Punkte nach Schritt 3

Punktnummer	temporäre Erreichbarkeitsdistanz
4	s_{34}

Tabelle 2.7.: Liste nach Schritt 3

In Punkt 4 bricht der Algorithmus ab, weil es keine weiteren Punkte in der Liste *OrderSeeds* gibt und im Datensatz bereits alle Punkte besucht sind (siehe Tabelle 2.8).

Punktnummer	Kerndistanz	Erreichbarkeitsdistanz
1	s_{12}	*nan*
2	s_{12}	s_{12}
3	s_{34}	s_{13}
4	s_{34}	s_{34}

Tabelle 2.8.: Besuchte Punkte nach Schritt 4

Über die Kerndistanz definiert ein Punkt seinen räumlich nächsten Nachbarpunkt. Mit der Erreichbarkeitsdistanz definiert der Punkt den Punkt, von dem aus er besucht wird. Weil aus der Liste *orderSeeds* immer der Punkt mit der kürzesten Erreichbarkeitsdistanz als neuer nächster Punkt gewählt wird, ordnet der Algorithmus Punkte nach ihrer näheren Umgebung und beschreibt somit die Dichte einer Region.
Das Ergebnis des Algorithmus ist das File *OrderedFile* oder im vorliegenden Beispiel die Tabelle 2.8. Pro Zeile steht in der ersten Spalte der Index eines besuchten Punktes, an zweiter Stelle die Kerndistanz und an dritter Stelle die Erreichbarkeitsdistanz. Für die weitere Berechnung wird nur die Erreichbarkeitsdistanz der einzelnen Punkte benötigt.

2.6.2. Bestimmung der ϵ- und *MinPts*-Parameter

Der OPTICS-Algorithmus ist von zwei Eingabeparametern ϵ und *MinPts* abhängig. Bei der optimalen Wahl des ϵ-Parameters ist der Wert zumindest so groß, wie die größte Distanz zwischen zwei Punkten im Datensatz. Falls $\epsilon = \infty$ gesetzt wird, kann die Wahl des *MinPts*-Parameter nur mehr zwei Zustände bewirken:

$$\begin{cases} \text{alle Punkte gehören einer Teilmenge an, falls } MinPts \leq \text{Punkteanzahl} \\ \text{kein Punkt gehört einer Teilmenge an, falls } MinPts > \text{Punkteanzahl} \end{cases}$$

Falls alle Punkte einer Teilmenge angehören, sind alle Punkte Kernpunkte (siehe Abb. 2.17).

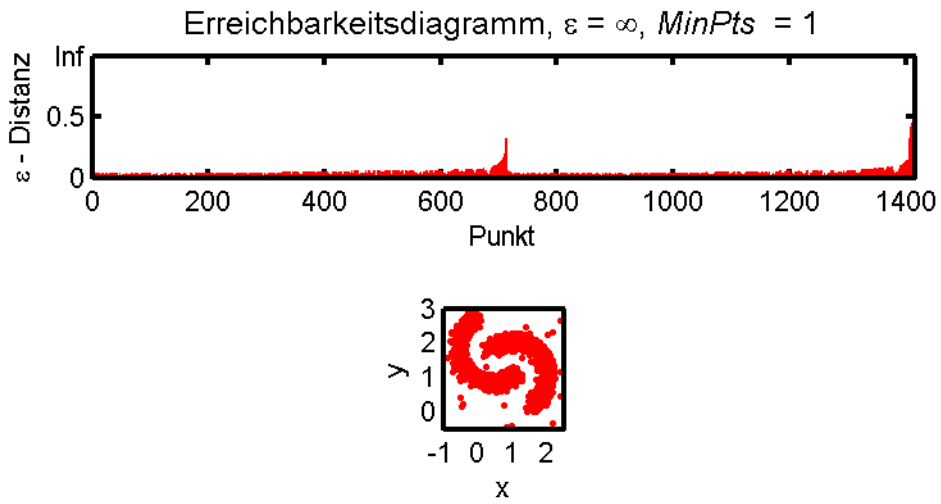


Abbildung 2.17.: Alle Punkte sind Kernpunkte, Quelle: eigene Darstellung

Das obere Bild der Abbildung 2.17 ist ein Erreichbarkeitsdiagramm. Ein Erreichbarkeitsdiagramm ist das grafische Abbild des Textfiles *OrderedFile* und ist im Unterabschnitt 2.6.3 näher erläutert. Die Wahl der Parameter wie sie in Bild 2.17 verwendet werden, ermöglicht die maximale Ausbeute an Information aus dem Datensatz, verbraucht aber auch die meiste Rechenleistung. Wird der ϵ-Parameter kleiner gesetzt, sind nicht mehr alle Datenpunkte Kernpunkte. Ein kleiner ϵ-Wert und ein kleiner *MinPts*-Wert liefern die Grobstruktur im Datensatz. Falls die beiden Halbkreise detektieren werden sollen, sind die Parameter in Abbildung 2.18 in Bezug auf die Datenstruktur gut gewählt. Wird der ϵ-Parameter zu klein gesetzt, dann werden nur mehr einzelne Punktepaare als Teilmengen gefunden. Wenn der *MinPts*-Parameter geändert wird, werden zwar weniger Kernpunkte, aber mehr Randpunkte detektiert. Dadurch wird nicht mehr die gesamte Grobstruktur detektiert, dafür aber die Kernbereiche besser hervorgehoben.

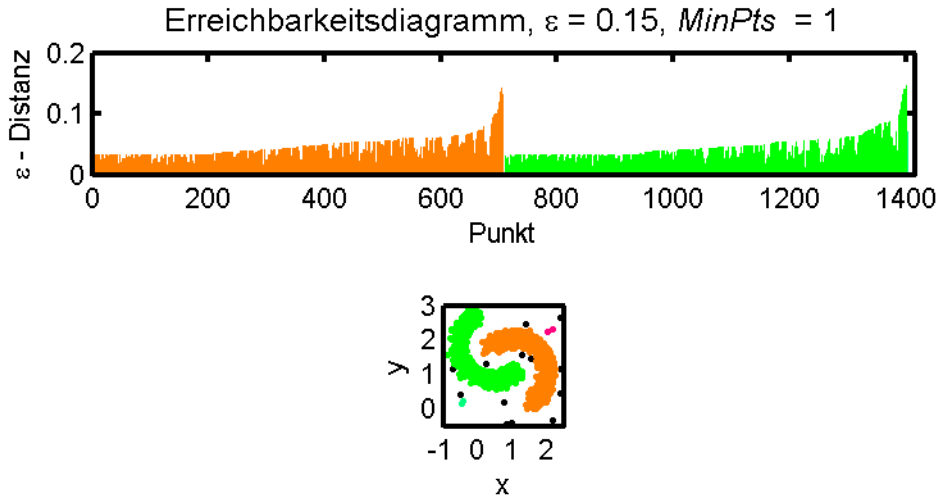


Abbildung 2.18.: Es werden zwei Teilmengen detektiert, Quelle: eigene Darstellung

Je höher der *MinPts*-Parameter ist, desto stärker werden die Daten separiert (siehe Abb. 2.19). Ein kleiner *MinPts*-Parameter birgt die Gefahr, dass bereits einzelne Punkte zwischen zwei Strukturen diese beiden Strukturen verbinden.

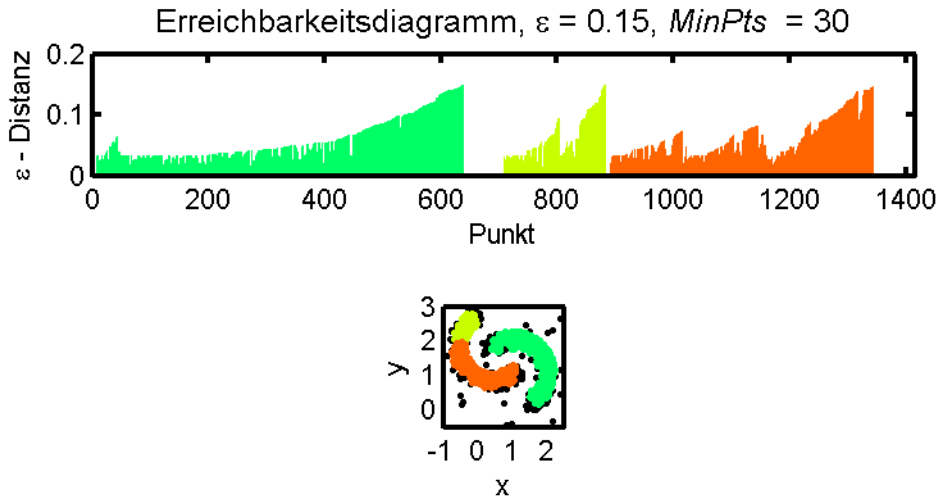


Abbildung 2.19.: Es werden mehrere Teilmengen detektiert, Quelle: eigene Darstellung

In Analogie zu einer Schipiste, detektiert ein großer *MinPts*-Wert an der Stelle einer Piste eine Teilmenge, die den mittleren Bereich einer Piste hervorhebt. Ein kleiner *MinPts*-Parameter detektiert für die gleiche Piste eine Teilmenge, die die gesamte Breite der Piste darstellt. Ein großer ϵ-Parameter detektiert das Schigebiet als eine einzige große Teilmenge, ohne einzelne Pisten. Ein kleiner ϵ-Wert detektiert pro Piste mehrere kleinere Teilmengen.

2.6.3. Erreichbarkeitsdiagramm

Der OPTICS-Algorithmus ordnet die Punkte eines Datensatzes anhand der Punktedichte im Datensatz. Punkte, die nah beieinander liegen, werden hintereinander besucht und stehen so auch im Textfile. In der ersten Spalte stehen die Indizes der besuchten Punkte, in der zweiten Spalte die Kerndistanzen und in der dritten Spalte die Erreichbarkeitsdistanzen. Je näher zusammen die Punkte im Datensatz liegen, desto geringer ist deren Erreichbarkeitsdistanz, was man auch im Erreichbarkeitsdiagramm sieht (siehe Abb. 2.20). Das Diagramm ist die grafische Darstellung der Erreichbarkeitsdistanzen.

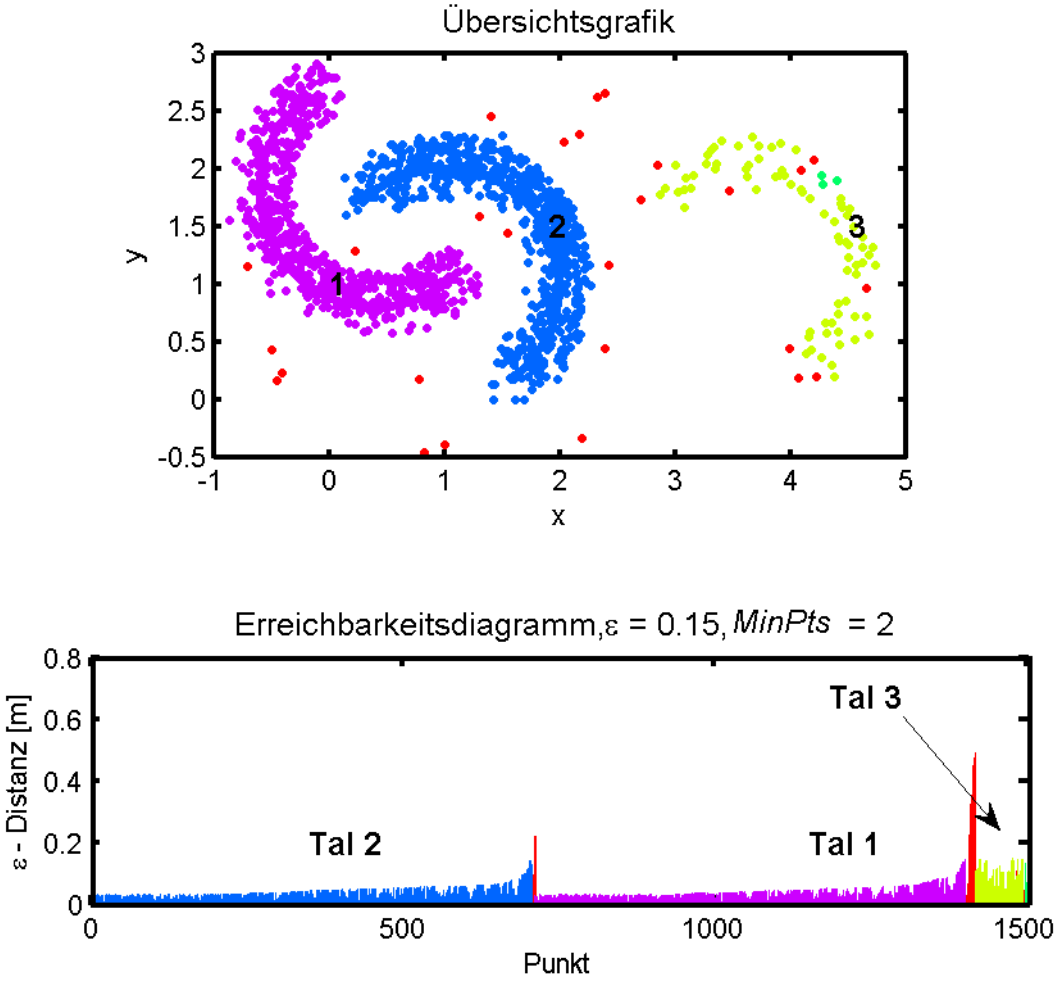


Abbildung 2.20.: Beispiel eines Erreichbarkeitsdiagrammes, Quelle: eigene Darstellung

In Abbildung 2.20 ist der Lageplan und das Erreichbarkeitsdiagramm eines Datensatzes geplottet. Halbkreis 1 und 2 bestehen aus ähnlichen Daten und haben im Erreichbarkeitsdiagramm ein ähnliches Aussehen. Der Halbkreis 3 besteht aus

weniger dicht zusammenliegenden Punkten. Auch er ist im Erreichbarkeitsdiagramm zu erkennen. Da er aus weniger Punkten besteht, ist das dazugehörende Tal im Erreichbarkeitsdiagramm nicht so breit und auch nicht so tief. Je dichter Punkte beieinander liegen, desto geringer ist deren Erreichbarkeitsdistanz.
Um die Extraktion von Clustern aus Datensätzen etwas übersichtlicher zu gestalten, wird im folgenden ein kleinerer Datensatz verwendet. In Abbildung 2.22 ist das Erreichbarkeitsdiagramm für den Datensatz, der in Abbildung 2.21 dargestellt ist, geplottet.

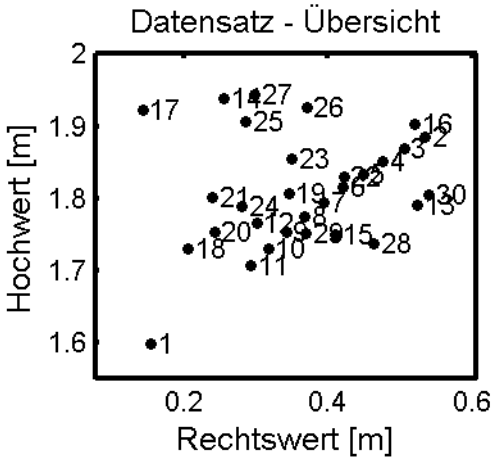


Abbildung 2.21.: Beispiel einer Punkteverteilung, Quelle: eigene Darstellung

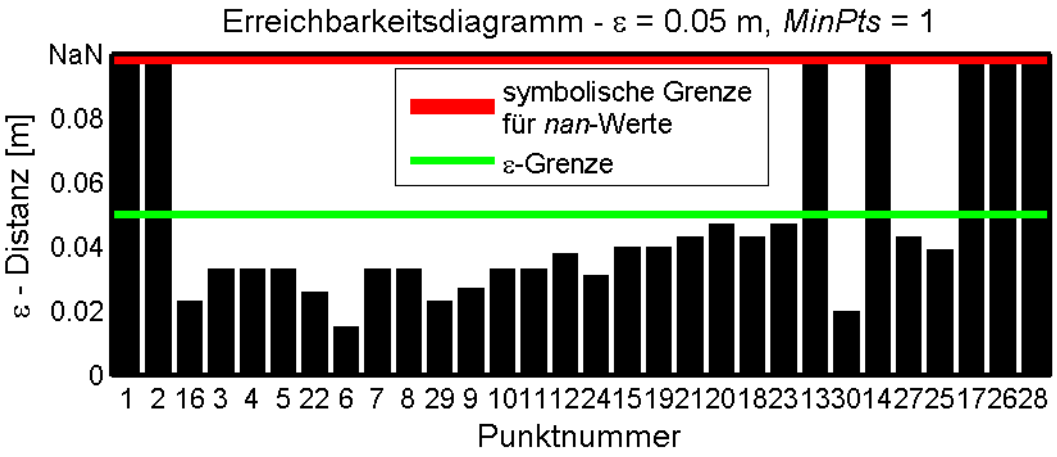


Abbildung 2.22.: Erreichbarkeitsdiagramm, Quelle: eigene Darstellung

Die Reihenfolge, wie die Punkte im Erreichbarkeitsdiagramm dargestellt sind, ist die selbe, wie sie im Textfile ist. Punkte, deren Erreichbarkeitsdistanzen bis zum roten Strich reichen, sind Punkte, die als Erreichbarkeitsdistanz den Wert *nan* im

Textfile stehen haben. Der Punkt 1 hat keinen Punkt in seiner ϵ-Nachbarschaft und auch keine Kerndistanz, also gehört der Punkt keiner Teilmenge an und ist damit Rauschen. Der Punkt 2 hat eine Kerndistanz aber keine Erreichbarkeitsdistanz. Dieser Punkt ist der Startpunkt einer Teilmenge von Punkten, da der nächste Punkt im Erreichbarkeitsdiagramm eine Erreichbarkeitsdistanz hat.

2.6.4. Extraktion von Clustern

Es gibt verschiedenste Varianten, Cluster anhand der Erreichbarkeitsdistanzen zu extrahieren. Alle Varianten beinhalten die Definition einer minimalen Anzahl von Punkten, die gegeben sein muss, um eine Teilmenge zu definieren:

> „To ignore regions that are too small in the reachability plot we assume (as is typically done in other methods as well) a minimum cluster size. In our examples, we set the minimum cluster size to 0.5% of the whole data set, which still allows us to find even very small clusters. The main purpose is the elimination of „noisy“ regions that consists of many insignificant local maxima, which are close to each other.“
>
> siehe Sander (2003: Kapitel 3) [12]

Die Teilmengen können aus dem Erreichbarkeitsdiagramm auf verschiedene Varianten extrahiert werden. Die einfachste Variante Teilmengen zu extrahieren ist, Grenzen an den Stellen zu setzen, wo die Erreichbarkeitsdistanzen *nan* sind. Teilmengen ergeben sich aus Punkten zwischen zwei Grenzen, falls diese aus einer minimalen geforderte Anzahl von Punkten bestehen.

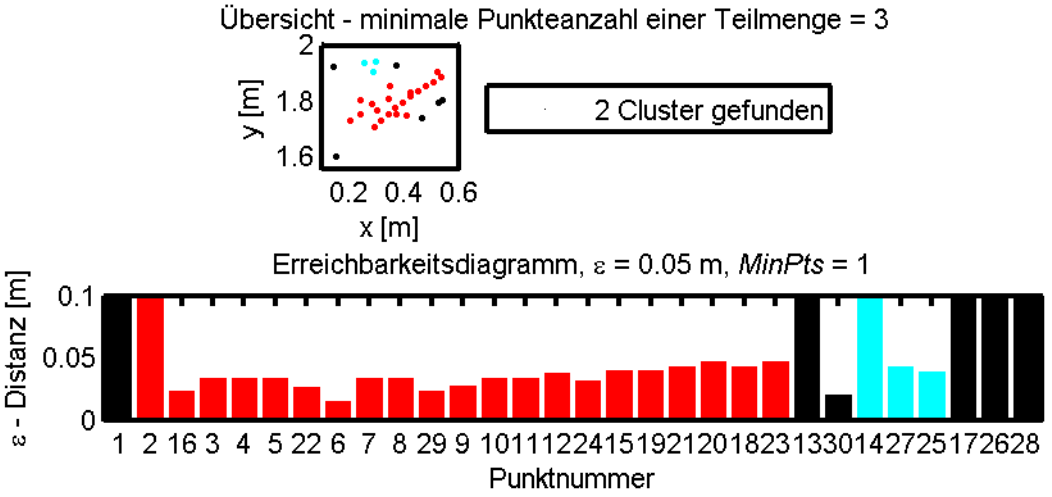


Abbildung 2.23.: Teilmengen extrahieren - Variante 1, Quelle: eigene Darstellung

Die linke Begrenzung einer Teilmenge ist immer ein Punkt mit *nan* als Erreichbarkeitsdistanz. Die rechte Begrenzung ist immer ein Punkt vor dem nächsten Punkt mit *nan* als Erreichbarkeitsdistanz (siehe Abb. 2.23).
Die zweiten Variante beweist, dass der OPTICS-Algorithmus eine Erweiterung des DBSCAN-Algorithmus ist. Statt wie in der vorigen Variante nur Punkte mit Erreichbarkeitsdistanzen *nan* als Grenzpunkte von Teilmengen zu definieren, kann auch ein beliebiger Maximalwert für die Erreichbarkeitsdistanz gewählt werden und Punkte mit selbigen Wert oder darüber als Grenzpunkte zu definieren (siehe Abb. 2.24). Das Ergebnis der zweiten Variante ergibt das gleiche Ergebnis, wie die Berechnung des DBSCAN-Algorithmus mit dem selben *MinPts*-Parameter und dem selben Maximalwert als ϵ-Parameter.

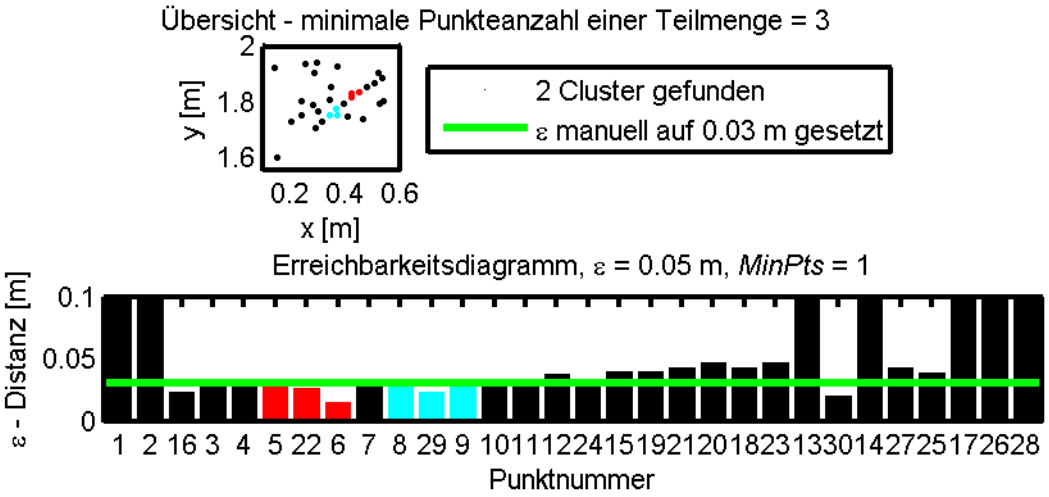


Abbildung 2.24.: Teilmengen extrahieren - Variante 2, Quelle: eigene Darstellung

Bei der dritten Variante werden Teilmengen ohne der Wahl eines fixen ϵ-Parameters extrahiert. Der Algorithmus sucht zusammengehörende Teilmengen. Dabei wird vom Groben ins Detail vorgegangen. Im Vorbereitungsschritt werden die Daten in Teilmengen, wie in der ersten Variante beschrieben, aufgeteilt. Im Anschluss wird jede Teilmenge separat untersucht. Im ersten Schritt werden die lokalen Maxima bestimmt und nach ihrer Größe geordnet. Um die Berechnung aller kleineren Teilmengen einer Menge zu ermöglichen, wird die Funktion *cluster_tree* aufgerufen. Die Eingabeparameter der Funktion sind ein Satz von Daten *N*, ein Elternknoten *parent_of_N*, der bei der Initialisierung leer ist und eine Liste *L* von lokalen Maxima. Der Elternknoten ist immer der übergeordnete Knoten, in welchem alle Punkte seiner Kindknoten enthalten sind.

```
cluster_tree(N,parent_of_N,L)
```

```
    // N is a node; the root of the tree in the first call of the procedure
    // parent_of_N is the parent node of N; nil if N is the root of the tree
    // L is the list of local maxima points, sorted in descending order
    // of reachability
    if (L is empty) return; //parent_of_N is a leaf
    // take the next largest local maximum as possible separation
    // between clusters
    N.split_point = s = L.first_element()
    L=L-{s}
    // create two new nodes and add them to a list of nodes
    N1.points={p ∈ N.points|p is left of s in the reachability plot}
    N2.points={p ∈ N.points|p is right of s in the reachability plot}
    L1={p ∈L|p lies to the left of s in the reachability plot}
    L2={p ∈L|p lies to the right of s in the reachability plot}
    Nodelist NL = [(N1,L1),(N2,L2)]
    // test whether the resulting split would be significant
    if ((average reachability value in any node in NL)/s.r_dist>0.75)
        //if split point s is not significant, ignore s and continue
        cluster_tree(N,parent_of_N,L)
    else// remove clusters that are too small
        if (N1.NoOfPoints)<min_cluster_size)NL.remove(N1,L1)
        if (N2.NoOfPoints)<min_cluster_size)NL.remove(N2,L2)
        if(NL is empty) return// we are done and parent_of_N will be a leaf
    end // if
    // check if the nodes can be moved up one level
    if(s.r_dist and parent_of_N.split_point.r_dist are approximately the same)
        let parent_of_N point to all nodes in NL instead of N
    else
        let N point to all nodes in NL
    end // if
    // call recursively for each new node
    for each element(N_i,L_i)  in NL
        cluster_tree(N_i,parent_of_N,L_i)
    end // if
end // cluster_tree
```

siehe Sander (2003: Kapitel 3) [12]
Die Funktion splittet einen Datensatz an der Stelle des größten lokalen Maximums. Alle Daten, die bildlich gesehen links vom Splitterpunkt liegen, gehören einem neuen temporären Kindknoten *N1* an, alle Daten rechts vom Splitterpunkt gehören einem weiteren Kindknoten *N2* an. Die lokalen Maxima werden ebenfalls auf zwei Listen *L1* und *L2* geteilt. Für beide Kindknoten wird ein Mittelwert aus den Erreichbarkeitsdistanzen berechnet. Beide Mittelwerte werden mit der Erreichbarkeitsdistanz des Splitterpunktes dividiert und beide Werte müssen kleiner als ein signifikanter Wert sein:

> „The ratio for a significant separation is set to 0.75 i.e. for a local maximum p to be considered a cluster separation, the average reachability value of p. This value represents approximately the ratio that a user typically perceives as the minimum required difference in height to indicate a relevant cluster separation.“
> siehe Sander (2003: Kapitel 3) [12]

Mit dieser Bedingung wird verhindert, dass nicht signifikante lokale Maxima eine Teilmenge zu oft teilen. Falls ein Maximum nicht signifikant ist, werden die temporären Kindknoten gelöscht, das Maximum ignoriert und die Funktion *cluster_tree* rekursiv aufgerufen. Falls das lokale Maximum signifikant ist, werden die temporären Kindknoten genauer untersucht. Die Anzahl der Punkte pro Knoten muss einen Minimalwert betragen, sonst wird der Knoten gelöscht. Weiters wird überprüft, ob der Kindknoten ein Kindknoten des aktuellen Elternknotens ist, oder ob der Knoten eigentlich ein „Bruderknoten“ des Elternknotens ist. Für die Überprüfung wird die Erreichbarkeitsdistanz des Splitterpunktes des aktuellen Knotens gegen die Erreichbarkeitsdistanz des Splitterpunktes des Elternknotens verglichen. Falls die Distanzen ähnlich sind, wird der Kindknoten vom Elternknoten gelöst und zum Elternknoten des aktuellen Elternknotens hinzugefügt. In Abbildung 2.25 ist der Fall dargestellt, in dem ein Elternknoten drei Kindknoten hat, da die drei größten Maxima eine ähnliche Größe haben. Die Überprüfung wird durchgeführt, da sonst nur zwei Knoten detektiert werden und dabei ein Knoten aus den Daten eines Halbkreises und ein Knoten aus den Daten zweier Halbkreise besteht.

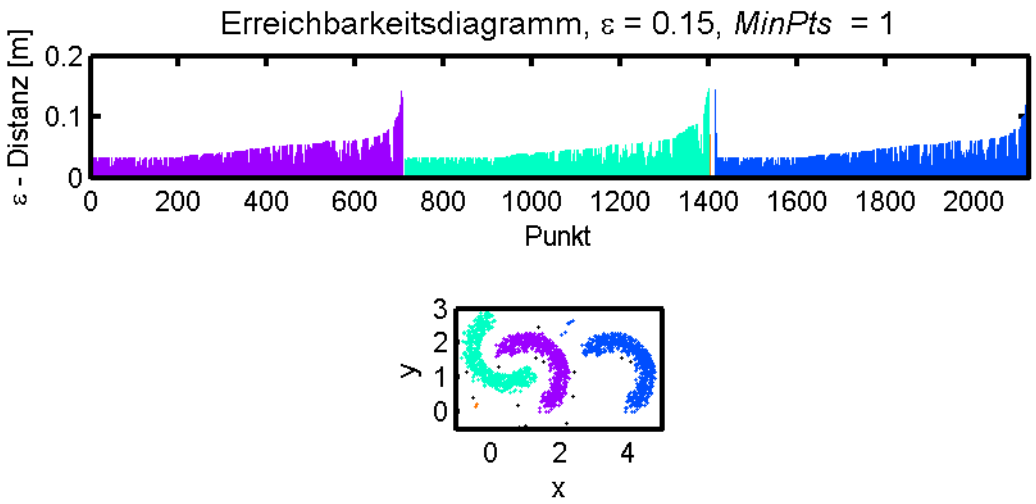


Abbildung 2.25.: Drei „Bruderknoten“, Quelle: eigene Darstellung

Zum Abschluss der Funktion *cluster_tree* wird jeder temporären Kindknoten zu einem permanenten Knoten. Weiters wird mit jedem neuen Knoten die Funktion *cluster_tree* rekursiv aufgerufen.

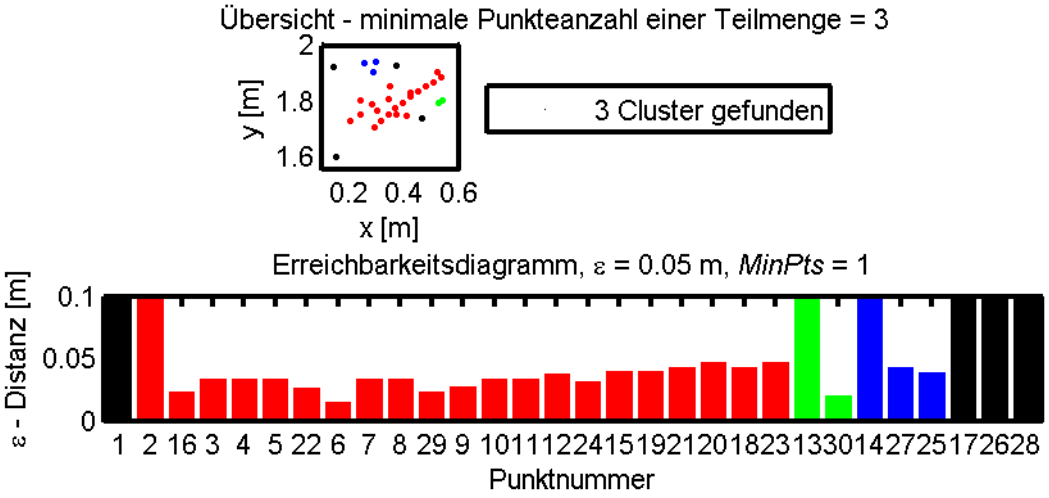


Abbildung 2.26.: Cluster extrahieren - Variante 3, Quelle: eigene Darstellung

Das Ergebnis der dritten Variante (siehe Abb.: 2.26) sind Teilmengen, die nicht weiter geteilt werden können, aber trotzdem aus einer minimalen Anzahl von Punkten pro Teilmenge bestehen. Dies wiederum ermöglicht eine flexiblere Detektion von Teilmengen. In einer Region mit einer großer Punktdichte werden viele Teilmengen gefunden. Auch in Regionen mit geringer Dichte werden Teilmengen gefunden. In Schigebieten sind die

Punkte in der Nähe von Liften sehr dicht verteilt. In diesen Regionen werden mehrere Teilmengen detektiert. Entlang von wenig befahrenen Pisten ist die Punktdichte und auch die Anzahl von gefunden Teilmengen geringer.

Interpretation der Ergebnisse

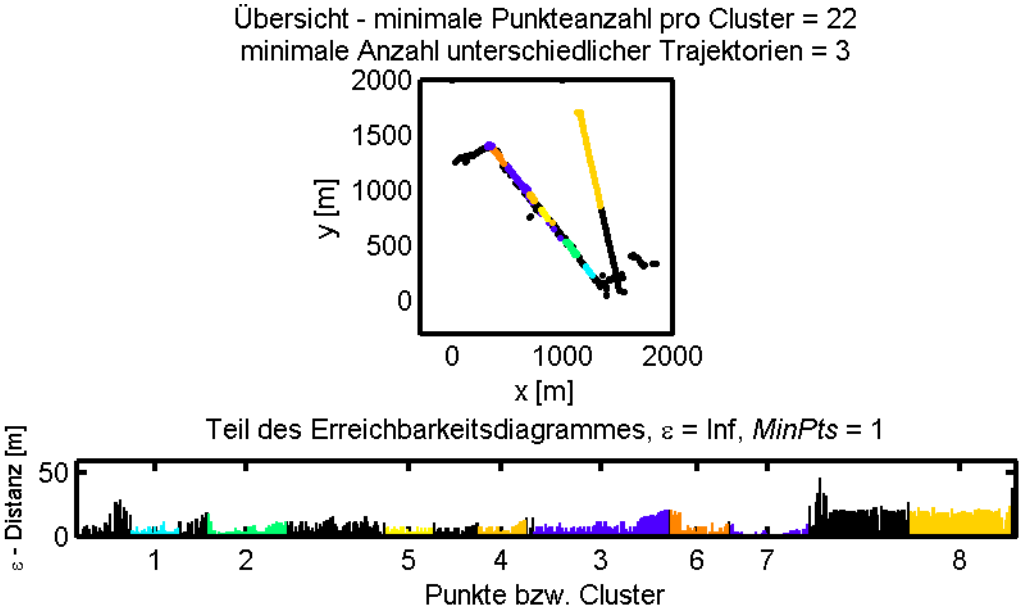


Abbildung 2.27.: Clusterergebnis und Teil des Erreichbarkeitsdiagramms, Quelle: eigene Darstellung

In Abbildung 2.27 ist das Clusterergebnis eines Teiles der Daten aus dem Schigebiet Saalbach dargestellt. Im oberen Bild sind zwei Lifte dargestellt. Für den rechten Lift ist ein Cluster detektiert. Für den linken Lift sind mehrere kleinere Cluster detektiert. Dass es sich beim Detailausschnitt um zwei Lifte handelt, ist auch im Erreichbarkeitsdiagramm, im unteren Bild, am großen Maximalwert, der den Datensatz in zwei Teilmengen teilt, zu erkennen. Der Maximalwert teilt den Ausgangsknoten, in welchem alle Punkte gegeben sind, auf zwei Kindknoten. Der linke Knoten ist einige weitere Male geteilt. Der rechte Knoten ist nur ein weiteres Mal auf den Cluster mit der Nummer 8 und einen restlichen Teil von Daten, der keinem Cluster zugeordnet ist, geteilt. Es hat mehrere Gründe, warum gewisse Daten keinem Cluster zugeordnet werden, wie z.B. signifikante Splitterpunkte. Der Algorithmus durchläuft die Daten vom Groben ins Detail. Man muss zwischen Knoten unterscheiden, die in der Abbildung dargestellt sind und Knoten, die der Algorithmus „kennt“. In Abbildung 2.27 sind nur Knoten dargestellt, die auf keine weitere Knoten verweisen. Der Knoten Nummer 1 ist der erste Knoten, der nicht auf weitere Knoten verweist. Bevor eine Teilmenge als Cluster

definiert wird, muss die Teilmenge eine genügend große Anzahl von Datenpunkten haben. Teilmengen mit einer zu geringen Anzahl an Punkten werden gelöscht. Dadurch entstehen im Erreichbarkeitsplot Bereiche, wo die Punkte keinem Cluster zugewiesen werden. Zwischen den Clustern 2 und 5 sind 38 schwarze Balken, was mehr als die geforderte minimale Punkteanzahl von 22 ist. Der Grund, warum diese Punkte keinem Knoten angehören, ist in der Abbildung 2.28 dargestellt, welche einen Detailausschnittes des Clusters 2 und die besagten nachfolgenden Punkte darstellt.

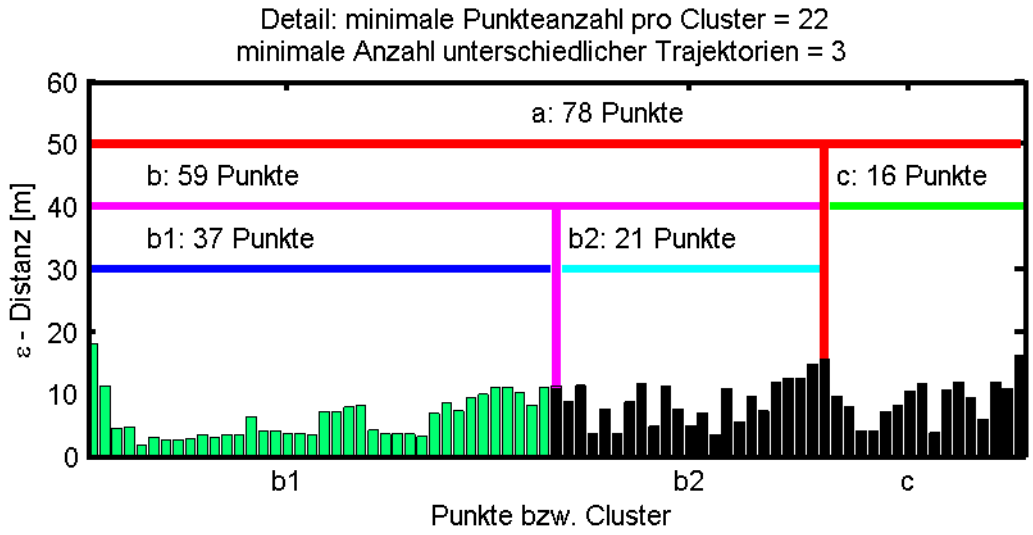


Abbildung 2.28.: Detailausschnitt des Erreichbarkeitsdiagrammes, Quelle: eigene Darstellung

Cluster *a* beinhaltet alle Datenpunkte und ist auf die Cluster *b* und *c* aufgeteilt. Cluster *b* ist auf die Cluster *b1* und *b2* geteilt. Cluster *b2* und *c* haben beide zu wenige Datenpunkte und sind im Erreichbarkeitsplot schwarz eingefärbt. Da die Punkte der beiden Cluster im Textfile hintereinander eingetragen sind, erscheint es, als ob eine große Teilmenge übersehen worden ist.
Ein weiterer Grund, weshalb hintereinander folgende Punkte nicht zu einem Cluster zusammengefasst werden, ist, dass es in der Teilmenge nicht genügend viele Punkte aus unterschiedlichen GPS-Trajektorien gibt. Dieser Umstand ist nicht Teil des originalen Algorithmus, sondern eine Zusatzbedingung, die im Abschnitt 3.8 erklärt ist. Diese Bedingung muss erfüllt werden, um eine Teilmengen in einen Cluster umzuwandeln. Aus diesem Grund sind die Punkte vor dem Cluster 8 in der Abbildung 2.27 schwarz eingefärbt, obwohl es in diesem Bereich eine Teilmenge gibt, die aber die Clusterbedingung nicht erfüllt.

2.7. Lifte berechnen

In diesem Abschnitt wird zunächst auf die Datenstruktur und die Bedeutung der Cluster eingegangen. Darauf aufbauend ist der Pseudocode für die Berechnung der Cluster, die Daten von Liften beinhalten, angeführt. Der Code, ebenso wie die Berechnung eines mittleren Liftes aus mehreren Anstiegen, ist nicht der Literatur entnommen und daher im Detail erklärt. Der schnelle Leser kann diesen Teil überspringen und auf die Seite 58 zum Abschnitt 2.8 weiterblättern.

Datenstruktur

Für die Berechnung von Liften muss als erstes die Datenstruktur, wie sie für die Berechnung von Liften zur Verfügung steht, erläutert werden. Ein Datensatz besteht aus mehreren Trajektorien - GPS-Tracks von Schifahrern. Jede Trajektorie besteht aus potentiellen Liftanteilen (Teiltrajektorien, die bergauf verlaufen) und potentiellen Pistenanteilen (Teiltrajektorien, die bergab verlaufen). Die Anteile werden für jede Trajektorie bestimmt und getrennt, da für die Berechnung der Lifte nur Liftanteile verwendet werden. Jede einzelne Trajektorie besteht meistens aus mehreren einzelnen Liftanteilen. Diese Liftanteile werden als „Anstiege“ bezeichnet. Wie die Anteile aus den Tracks bestimmt werden, ist in den Abschnitten 2.1 und 2.2 erklärt. Der OPTICS-Algorithmus wird nur mit den Daten der Anstiege berechnet. Der Algorithmus ordnet die Datenpunkte so, dass Punkte von Anstiegen, die nah beieinander liegen, im Textfile *OrderedFile* hintereinander gereiht werden. Jede gefundene Teilmenge besteht aus unterschiedlichen Anstiegen und wenn die Anzahl von unterschiedlichen Anstiegen einen minimalen Wert überschreitet, erfüllt die Teilmenge die Bedingung eines Clusters.

Bestimmung der Cluster von Liften

Cluster entstehen vor allem in Regionen mit einer großen Punktdichte. Es müssen pro Cluster auch nicht alle Punkte eines Anstieges vorhanden sein. Es reicht bereits ein Datenpunkt. Daher gibt es Cluster, die aus Daten von Anstiegen bestehen, die unterschiedliche Lifte repräsentieren. Dies ist vor allem bei der Trennung von Liften mit Mittelstationen der Fall, in Bereichen, wo mehrere Lifte zusammen kommen oder bei parallelen Liften. Daher wird ein iterativer Prozess durchgeführt, um „korrekte“ Cluster zu finden, die jeweils einen Lift darstellen. Die Iteration wird so lange wiederholt, bis sich die Anzahl der Cluster nicht mehr ändert oder alle Cluster gelöscht werden. Aus

den übriggebliebenen Clustern werden gemittelte Lifte berechnet.

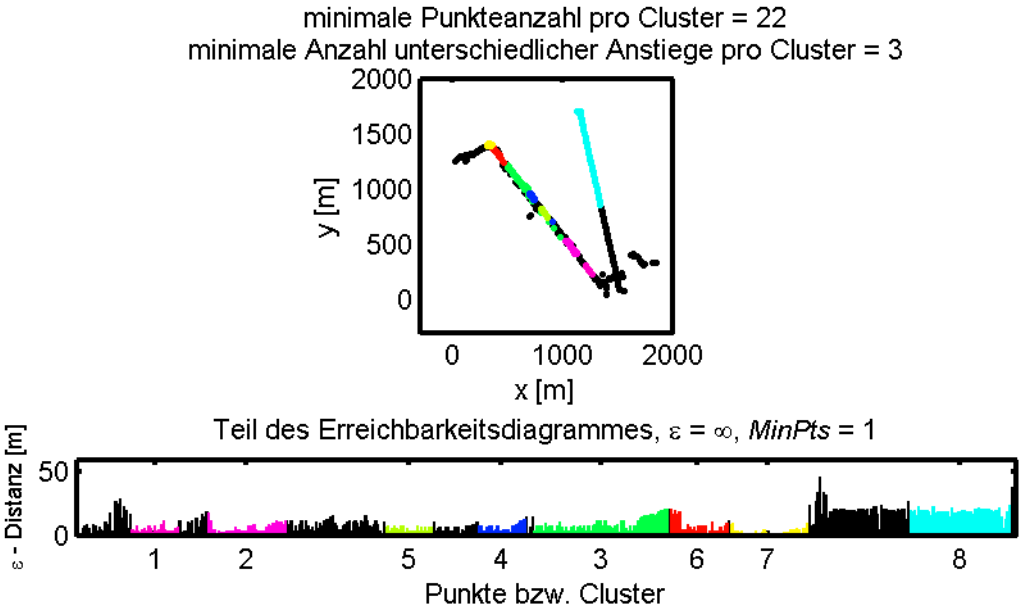


Abbildung 2.29.: Clusterergebnis und Erreichbarkeitsdiagramm, Quelle: eigene Darstellung

In Abbildung 2.29 ist das Clusterergebnis eines Detailausschnittes des Schigebietes Saalbach dargestellt. Um aus den Clustern die zwei Lifte zu berechnen, wird zunächst die Funktion *ermittleLifte* aufgerufen. Als Eingabeparameter benötigt die Funktion vier Parameter:

- *cluster* - ein Satz von Clustern
- *minUeber* - eine Variable, die im Zuge der Funktion *fuegeClusterZusammen* näher erklärt ist
- *abbAbw* - eine Variable, die als Abbruchbedingung in der Funktion *ermittleLifte* dient
- *MinPfadeCluster* - eine Variable, die als Bedingung für die Transformation von Teilmengen zu Clustern steht

```
lifte=ermittleLifte(cluster,minUeber,abbAbw,MinPfadeCluster)
    mitAbw=∞
    WHILE mitAbw>abbAbw ∨ cluster = [ ]
        cluster=fuegeClusterZusammen(cluster,minUeber)
```

```
        FOR i FROM 1 TO cluster.size
            c(i)=cluster(i)
            [c(i).liftkandidat,c(i).abweichungen]=kontrollierePfade(c(i),mitAbw)
        END % FOR
        mitAbw=Mittelwert aus allen Abweichungen in c
        FOR i FROM cluster.size TO 1
            pfadAnzahl=Anzahl von Pfaden in c(i)
            IF pfadAnzahl<MinPfadeCluster
                cluster(i)=[]
            END % IF
        END % FOR
        cluster=c
    END % WHILE
    lifte=c.liftkandidat
END % ermittlerLifte
```

In der Funktion wird zunächst die Variable *mitAbw* auf ∞ gesetzt. Die *while*-Schleife wird so lange ausgeführt, bis die Abweichungen aller Liftkandidaten den Toleranzwert *abbAbw* unterschreiten oder keine validen Cluster übrig bleiben. In der Schleife wird als erstes die Funktion *fuegeClusterZusammen* aufgerufen. Als Eingabeparameter braucht die Funktion einen Satz von Clustern und die Variable *minUeber*.

```
cluster=fuegeClusterZusammen(cluster,minUeber)
    c=cluster
    FOR i FROM c.size TO 1
        FOR j FROM 1 TO ii-1
            pfadAnzahl=Anzahl von Pfaden in c(i)
            uebereinstimmung=Anzahl von gemeinsamen Pfaden in c(i) und c(j)
            IF uebereinstimmung>=minUeber
                c(j)=c(j) ∪ c(i) // c(i) und c(j) zusammenfügen
                c(i)=[] // c(i) entfernen
            END % IF
            cluster=c
        END % FOR
    END % FOR
END % fuegeClusterZusammen
```

Die Funktion fügt Cluster zusammen, die zumindest *minUeber* gemeinsame Anstiege haben. Als nächster Schritt werden die Anstiege der Cluster in der Funktion *kontrolliere Pfade* untersucht. Die Eingabeparameter sind ein einzelner Cluster und die Variable *mitAbw*.

```
cluster=kontrollierePfade(cluster,mitAbw)
    c=cluster
    pfade=c.pfade
    mitAnfang = Mittelwert der Anfangspunkten der Pfade
    mitEnde = Mittelwert der Endpunkten der Pfade
    mitLaenge = Mittelwert der Streckenlängen der Pfade
    FOR i FROM c.pfad.size TO 1
        abweichung=|pfade(i).anfang-mitAnfang| + |pfade(i).ende-mitEnde| +
                  +|pfade(i).laenge-mitLaenge|
        IF abweichung≤mitAbw
            c.abweichung(i)=abweichung
        ELSE
            c.pfade(i)=[]
        END % IF
    END % FOR
    cluster=c
END % kontrollierePfade
```

Aus den Anstiegen eines Clusters wird ein mittlerer Start- und Endpunkt, sowie eine mittlere Streckenlänge bestimmt. Für jeden Anstieg im Cluster wird die Abweichung zu den Mittelwerten berechnet und summiert. Die Summe muss den Wert *mitAbw* unterschreiten, sonst wird der Pfad aus dem Cluster eliminiert. Nachdem die Pfade jedes Clusters gefiltert sind, wird aus allen Abweichungen der Mittelwert gebildet und als neuer Wert für die Variable *mitAbw* definiert. Der neue Wert wird mit der Variable *abbAbw* verglichen und falls der neue Mittelwert den Wert *abbAbw* unterschreitet, wird die Funktion abgebrochen. Falls der Wert nicht unterschritten wird, werden alle Cluster mit zu wenigen unterschiedlichen Anstiegen gelöscht und danach die Berechnung wiederholt. Nach jeder Iteration passen die Pfade pro Cluster besser zusammen, da immer wieder Pfade aufgrund des kleiner werdenden Wertes *mitAbw* aus den Clustern entfernt werden. Aus jedem Cluster, der beim Abbrechen der Funktion *ermittleLifte* übrig bleibt, wird ein mittlerer Lift bestimmt. Obwohl die Variable *mitAbw* pro Iteration kleiner wird, ist es notwendig, eine maximale erlaubte Grenze *abbAbw* für Abweichungen zu setzen. Wenn

die Grenze zu klein gesetzt wird, werden pro Cluster zu viele Pfade eliminiert und in Folge dessen zu viele Cluster gelöscht.

Berechnung von gemittelten Liften

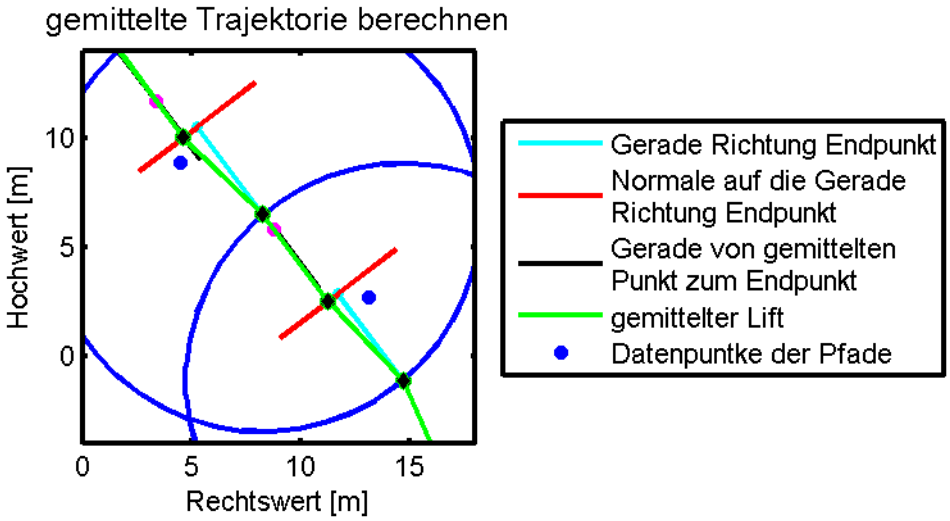


Abbildung 2.30.: Lift aus Pfaden mitteln, Quelle: eigene Darstellung

In Abbildung 2.30 ist ein Teil der Bestimmung von Liften dargestellt. Hilfreich ist die Tatsache, dass Lifte eine sehr geradlinige Trasse haben. Die Eintritts- und Austrittspunkte der Lifte sind als Mittelwerte der Pfadanfangs- und Endpunkte gegeben. Alle anderen Liftpunkte müssen berechnet werden. Zunächst wird vom Startpunkt des Liftes die Gerade (cyane Linie) zum Endpunkt bestimmt. Danach wird vom Startpunkt ein Stück entlang der Geraden ein Punkt definiert und durch diesen die Normale (rote Linie) auf die Gerade gelegt. Als nächstes wird ein Radius um den Startpunkt gelegt und der mittlere Punkt aus den Daten innerhalb des Radius bestimmt. Von diesem Punkt aus wird eine Gerade (schwarze Linie) zum Endpunkt berechnet. Der Schnittpunkt dieser Geraden mit der roten Linie ist ein neuer Punkt des Liftes. An jedem neuen Liftpunkt wird eine neue Gerade zum Endpunkt und ein neuer Punkt entlang dieser Geraden bestimmt, die Normale durch diesen Punkt gelegt, der Radius um den aktuellen Liftpunkt gelegt, der mittlere Wert innerhalb des Radius bestimmt, die Gerade vom Mittelpunkt zum Endpunkt berechnet und der neue Liftpunkt als Schnittpunkt zwischen der Geraden und der Normalen bestimmt. In Bereichen, wo sich keine Trajektorienpunkte innerhalb des Radius befinden, liegt der nächste Liftpunkt auf der Geraden vom aktuellen Punkt zum Endpunkt. Damit können auch in „punktearmen“ Regionen Liftpunkte

bestimmt werden. Wenn ein potentieller Liftpunkt innerhalb des Radius des Endpunktes liegt, wird die Berechnung der gemittelten Lifttrajektorie abgebrochen. Die Berechnung eines gemittelten Liftes ist damit abgeschlossen.

2.8. Pisten berechnen

In diesem Abschnitt wird zunächst die Datenstruktur für die Pistenberechnung erläutert, danach der Suchalgorithmus für Pisten zwischen zwei Liften und zuletzt der Algorithmus für die Berechnung eines Netzes von Verbindungen für ein Schigebiet.

Datenstruktur

Der Algorithmus kennt zum Zeitpunkt der Berechnung von Pisten die Lifte. Als Daten werden alle Trajektorienteile, die bei der Berechnung von Liften nicht verbraucht werden, verwendet. Ein Großteil der Abfahrten beginnt oder endet bei einem Lift. Ein Großteil der Pistenanteile kann einem Start- oder Endpunkt eines Liftes zugewiesen werden. Die Abfahrten werden in vier Kategorien eingeteilt:

- Abfahrten mit einem Lift als Startpunkt
- Abfahrten mit einem Lift als Zielpunkt
- Abfahrten mit einem Lift als Start- und einem Lift als Zielpunkt
- Abfahrten ohne Lift als Start- oder Endpunkt

Falls es zwischen zwei Liftpunkten mehrere Abfahrten gibt, kann davon ausgegangen werden, dass zwischen diesen beiden Punkten eine Piste besteht. Je höher die minimale Anzahl von Abfahrten zwischen zwei Punkten gesetzt wird, desto kleiner wird die Anzahl von potentiellen detektierten Pisten sein. Eine minimale Anzahl von Abfahrten muss gesetzt werden, damit nicht bereits die Abfahrten eines einzelnen Schifahrers eine Piste begründen, was z.B. bei der Verwendung von Daten von Schifahrern, die sich im freien Gelände bewegen, möglicherweise Folgen hätte. Nachdem die Abfahrten den Liftpunkten zugewiesen sind, kann zwischen zwei Liftpunkten eine Piste detektiert werden. Anders als bei der Berechnung von potentiellen Liften, kann bei Pisten nicht davon ausgegangen werden, dass diese geradlinig verlaufen. Vielmehr sind die Abfahrten eine Schar von Linien, die im besten Fall eine Hülle um die mittlere Piste bilden. Es kann zwischen zwei Liftpunkten eine Vielzahl von Pisten geben. Daher kann nicht die Hülle bestimmt

werden und aus dieser die mittlere Piste abgeleitet werden. Bei Pisten können zwei Abfahrten gleich lang sein, bei den selben Liftpunkten beginnen und enden und trotzdem unterschiedliche Pisten darstellen. Der Algorithmus findet einen Bündel von Abfahrten, die beim selben Liftpunkt starten und beim gleichen Liftpunkt enden.
Eine Piste kann sowohl als Linie in Vektorform betrachtet werden, als auch als Fläche in Rasterform. Wenn man einen Schigebietsplan betrachtet, dann sind Pisten immer als Verbindung (Vektorform) zwischen zwei Liften dargestellt. Tatsächlich gibt es pro Piste zumeist mehrere Varianten, welche die Schifahrer wählen können. Manche Schifahrer fahren lieber lange Schwünge, wohingegen andere eher die direkte Linie bevorzugen. Der beste Ansatz wäre es, die Hülle aus einer Schar von Abfahrten zu bestimmen. Die Hülle ist sozusagen die Begrenzung einer Piste. Das Problem ist, dass die Abfahrten pro Bündel unterschiedlichen Pisten angehören können. Daher kann man sich bei der Bestimmung einer Hülle niemals sicher sein, dass das Ergebnis tatsächlich eine Piste ist. Daher wird dieser Ansatz verworfen und ein anderer Ansatz verwendet. Über das Schigebiet wird ein Raster gelegt und die Punkte der Abfahrten den Rasterzellen zugewiesen. Im Raster wird danach nach Rasterzellen gesucht, die benachbart sind und Daten von gewissen Abfahrten beinhalten. Der Rasteransatz bietet mehrere Vorteile. Es kann für jede Rasterzelle ein Mittelwert bestimmt werden und entlang der Mittelwerte eine Piste in Vektorform bestimmt werden. In der Praxis findet man pro Piste mehrere Pistenvarianten. Benachbarte Pistenvarianten können mit dem Rasteransatz zu einer gemeinsamen Piste verschmolzen werden. Dadurch ist es möglich, unterschiedlich breite Pisten zu detektieren. Für den Fall, dass zwischen zwei Abfahrten mehrere Pisten verlaufen, werden diese mit dem Rasteransatz als separate Pisten detektiert, was mit dem Vektoransatz nicht möglich wäre.
Für die Berechnung der Lifte wurde aus den Anstiegen pro Lift eine gemittelte Piste aus den Vektordaten bestimmt. Da Pisten breiter sind und es viel mehr Varianten geben kann, werden Pisten mit dem Rasteransatz bestimmt. Die Piste in Vektorform als auch die Piste als zusammengehörende Anzahl von Rasterzellen widersprechen sich nicht, da man für unterschiedliche Anwendungen unterschiedliche Formen benötigt. Für die Navigation eines Schifahrers würde die Piste in Vektorform ausreichen, für die Optimierung der Pistenbeschneiung wird die Piste als Fläche benötigt.

Pisten entlang von Rasterzellen berechnen

Über das gesamte Schigebiet wird ein Raster gelegt. Als nächstes werden alle Abfahrten, die beim selben Liftpunkt eines Liftes starten und beim selben Liftpunkt eines anderen

Liftes enden, aus dem Satz von Abfahrten entnommen. Jeder Datenpunkt einer entnommenen Abfahrt wird einer Rasterzelle zugewiesen.

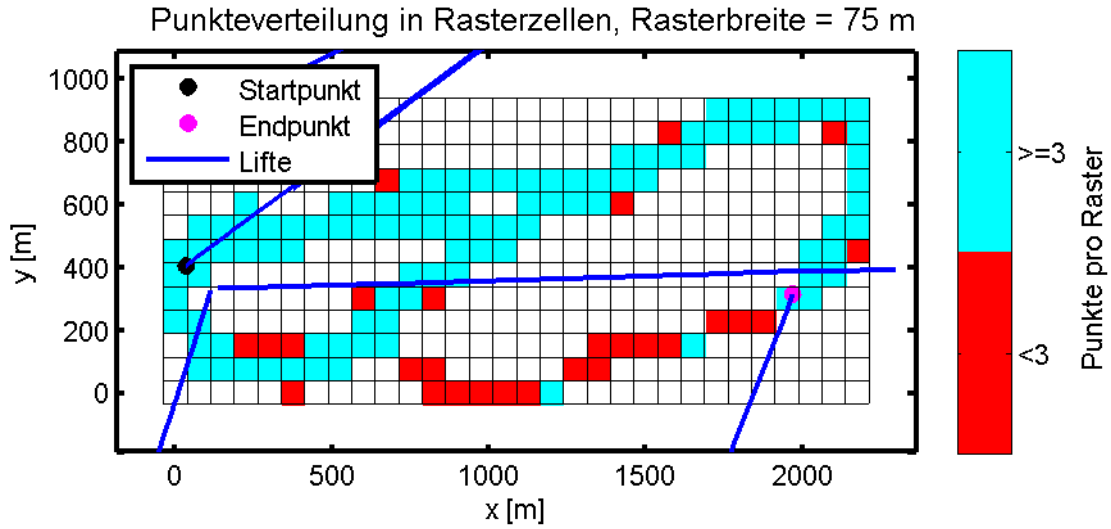


Abbildung 2.31.: Datenverteilung in Raster, Quelle: eigene Darstellung

Als erstes wird die Größe einer Suchmaske definiert, z.B. eine 3x3 Rasterzellen große Maske. Diese wird auf den Startpunkt einer Piste gelegt. Der Start- und Endpunkt einer Piste ist bekannt, da nur Abfahrten betrachtet werden, die zwischen zwei Liftpunkten verlaufen. In den Rasterzellen innerhalb der Maske wird pro Rasterzelle die Anzahl von unterschiedlichen Pfaden bestimmt und damit die Rasterzelle bewertet. Die Rasterzelle mit der besten Bewertung wird als nächster Punkt ausgewählt. Jede Abfahrt besteht aus einer gereihten Abfolge von Punkten. Pro Rasterzelle wird der niedrigste und größte Index eines Pfades, der in der Zelle auftaucht, bestimmt. Der niedrigste Wert bestimmt den „Zeitpunkt“ an welchem der Schifahrer zum ersten Mal eine Rasterzelle besucht, der größte Wert bestimmt, wann der Schifahrer die Rasterzelle verlässt. Pro Pfad erhält diejenige Rasterzelle einen Punkt für die Bewertung, die den niedrigsten Index des Pfades besitzt.

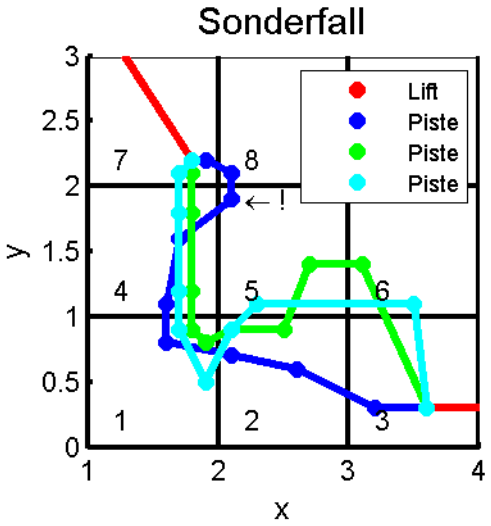


Abbildung 2.32.: Sonderfall, Quelle: eigene Darstellung

In Abbildung 2.32 ist die Begründung dargestellt, warum die Indizes der Pfade entscheidend für die Wahl der Piste entlang der Rasterzellen sind. Die Piste, die in der Abbildung detektiert ist, verläuft entlang der Rasterzellen: 7,4,1,2,3. Auf den Startpunkt in Zelle 7 wird die Suchmaske gelegt. Als mögliche weitere Zellen bieten sich die Zellen 4 und 5 an. Die Zelle 4 erhält zwei Punkte, weil der grüne und cyane Pfad in der Zelle einen niedrigeren Index als in der Zelle 5 hat. Die Zelle 5 erhält auf Grund des blauen Pfades einen Punkt. In diesem Fall wird die Zelle 4 als nächste Zelle ausgewählt. Für alle Pfade werden die niedrigsten und höchsten Indizes, die bisher in irgendeiner besuchten Zelle verwendet werden, vermerkt. Der entscheidende Punkt ist, dass die Zelle 5 aus den weiteren Berechnungen ausscheidet, weil der einzige Punkt des blauen Pfades in dieser Zelle kleiner ist, als der größte bisher verwendete Index (vermerkt in Zelle 4). In der Realität bedeutet dies, dass ein Schifahrer die Zelle 5 von der Reihenfolge her früher besucht hat als die Zelle 4. Damit muss auch dem Suchalgorithmus vorgegeben werden, dass der Punkt der blauen Abfahrt in Zelle 5 zu einem späteren Zeitpunkt als der Index, der in der Zelle 4 vermerkt ist, auftaucht. Dieser Umstand wird mit der Verspeicherung der Indizes berücksichtigt. Nach diesem Suchalgorithmus werden Rasterzellen in Richtung des Endpunktes gesucht. Damit ein Weg entlang der Rasterzellen gefunden wird und daraus eine Piste wird, müssen in allen Zellen entlang des Weges die gleichen Pfade vertreten sein. Daher werden pro Rasterzelle die unterschiedlichen Pfade vermerkt und bei jeder neuen Suche einer Rasterzelle wird nur mehr nach den bereits vermerkten Pfaden gesucht. Falls in einer Zelle ein Pfad nicht mehr auftritt, wird dieser Pfad aus der aktuellen Berechnung ausgeschlossen. Dadurch

wird gewährleistet, dass vom Anfang bis zum Ende nur Rasterzellen gesucht werden, die die selben Pfade beinhalten. Aus den Rasterzellen, die besucht werden, ergibt sich zum Abschluss die gemittelte Piste entlang der Rasterzellen.

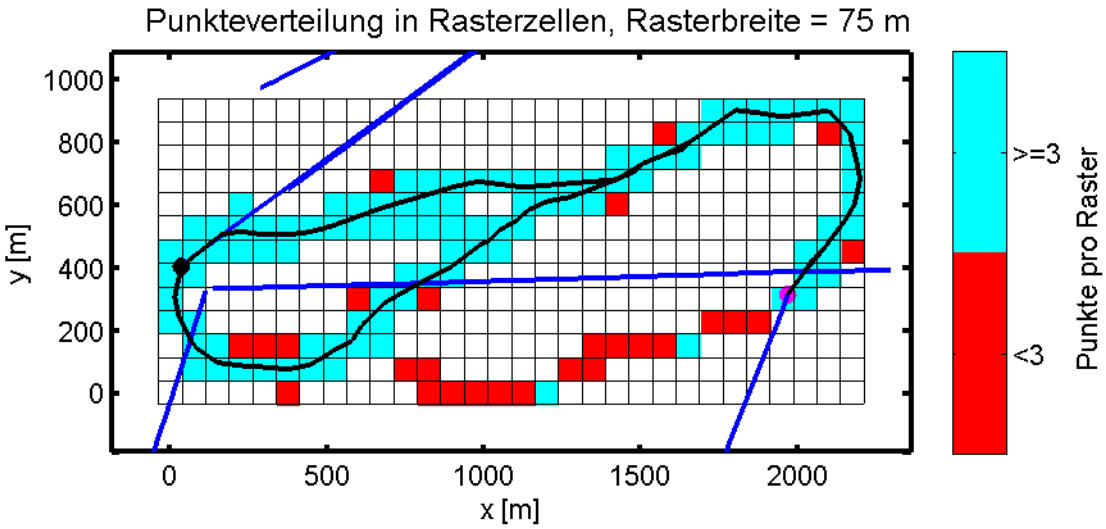


Abbildung 2.33.: 2 gemittelte Pisten, Quelle: eigene Darstellung

Da zwischen zwei Liftpunkten mehrere Pisten verlaufen können und für die Berechnung einer Piste nicht alle Pfade zwischen zwei Liftpunkten verbraucht werden, wird nach jeder Berechnung einer Piste die Suche nach einer weiteren Piste mit den selben Daten abzüglich den bereits verwendeten, gestartet. Die Berechnung einer Piste wird abgebrochen, sobald eine minimale Anzahl von unterschiedlichen Pfaden unterschritten wird. Pro Rasterzelle wird aus den Daten der Pfade, die für die Berechnung der Piste verwendet werden, ein mittlerer Wert bestimmt. Die gemittelten Punkte ergeben die gemittelte Piste. In Abbildung 2.33 sind zwei gemittelte Pisten dargestellt, die zum Teil die selben Rasterzellen verwenden, daher werden in solchen Rasterzellen auch die selben gemittelten Punkte verwendet.

Rasterverbindungen berechnen

In der Abbildung 2.33 ist der Umstand dargestellt, dass nicht jede Pistenvariante bestimmt werden kann, da nicht genügend viele Abfahrten entlang der Rasterzellen verlaufen. Trotzdem können die Verbindungen zwischen diesen Rasterzellen detektiert werden. Dazu muss die Bedingung, dass eine Piste nur aus Pfaden detektiert wird, aufgehoben werden. Der Vorteil dieser Variante ist, dass es viel mehr Verbindungen zwischen Rasterzellen gibt, der Nachteil ist, dass die Detektion von Pisten, wie sie in

Abbildung 2.33 zu sehen ist, nicht möglich ist.

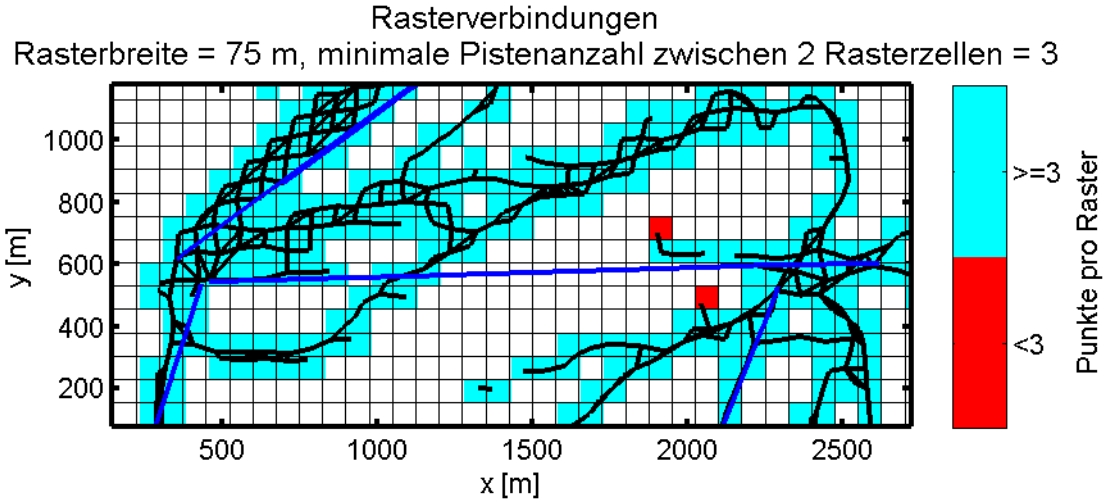


Abbildung 2.34.: Verbindungen zwischen Rasterzellen, Quelle: eigene Darstellung

In Abbildung 2.34 sind viele kleinere Verbindungen zu sehen, die vorher nicht detektiert wurden. Auch die Verbindungen der zwei Pisten, wie sie in Abbildung 2.33 zu sehen sind, sind gegeben. Weiters sieht man im oberen linken Bildrand einen weiteren Vorteil dieser Variante. Die Breite der Pisten kann besser bestimmt werden. Je schmaler die Rasterbreite gewählt wird, desto besser kann eine Piste abgebildet werden. Bei zu kleinen Rasterbreiten muss bedacht werden, dass eine genügend große Anzahl von unterschiedlichen Pfaden pro Raster detektiert werden muss. Weiters muss es eine weitere Rasterzelle in der Suchmaske geben, die die gleichen Pfade besitzt, damit eine Verbindung detektiert werden kann. Die Wahl der Rasterbreite hängt von der Datenanzahl ab. In einem Schigebiet in dem nur wenige Daten gegeben sind, muss die Rasterbreite höher gesetzt werden. Dadurch wird aber auch die Genauigkeit verschlechtert. Rasterbreiten zwischen 50 und 100 Metern liefern durchwegs gute Ergebnisse.

3. Praktische Umsetzung

Das parallel zum Schreiben der Arbeit erstellte Programm zur Berechnung von Pisten und Liften aus einem Datensatz ist gleich aufgebaut wie die nachfolgend beschriebenen Schritte. Dabei ist zu beachten, dass jeder Teilschritt in der Reihenfolge, wie die Schritte in diesem Kapitel aufgelistet sind, ausgeführt werden muss. Alle Algorithmen, die im Kapitel 2 beschrieben sind, finden sich in den Teilschritten wieder. In diesem Kapitel sind die Eingabeparameter der einzelnen Schritte beschrieben und deren Auswirkungen auf das Ergebnis. Die Ergebnisse und die Interpretationen der einzelnen Schritte sind im Kapitel 4 anhand zweier Beispieldatensätze beschrieben. Die Schritte 3.1 bis 3.6 befassen sich mit der Suche von Anstiegen in den einzelnen Trajektorien. Dabei werden die theoretischen Überlegungen der Abschnitt 2.1 bis 2.4 umgesetzt. Die Schritte 3.7 und 3.8 sind die Umsetzung des OPTICS-Algorithmus und die Extraktion von Clustern. Die Grundlagen dazu sind in den Abschnitten 2.6 und 2.7 näher erläutert. Der Schritt 3.9 ist die Berechnung von Liften, wie es in Abschnitt 2.7 beschrieben steht. Die Schritte 3.10 bis 3.15 befassen sich mit der Berechnung von Pisten.

3.1. Daten einlesen

In diesem Schritt werden die Files eingelesen.

3.2. Daten glätten

In diesem Schritt werden die Koordinaten eines Files geglättet.

Jeder File besteht aus unzähligen hintereinander folgenden Koordinatentriple. Die Eingabeparameter für diesen Schritt sind ein Datensatz, die Anzahl an Iterationen, der Grad des Polynoms und die Fensterbreite. Mit der Fensterbreite wird die Anzahl von Datenpunkten definiert, durch die ein Polynom n-ten Grades gelegt wird. Der mittlere Punkt im Fenster wird als geglätteter Wert übernommen. Der Parameter Iteration bestimmt, wie oft die Daten geglättet werden. Der im Schritt 3.7 implementierte OPTICS-Algorithmus arbeitet sowohl mit geglätteten als auch mit nicht geglätteten Daten gleich.

3.3. UTM-Koordinaten berechnen

Die ellipsoidischen Koordinaten werden, wie im Unterabschnitt 2.4.2 beschrieben, in UTM-Koordinaten transformiert.

3.4. Trajektorie spalten

In diesem Schritt werden die GPS-Trajektorien in Teilstücke gespalten.

Jeder Track besteht aus einer Anzahl von hintereinander aufgenommen Punkten. Zwischen zwei hintereinander folgenden Punkten wird der Richtungswinkel

$$\alpha = \text{atan}\frac{DY}{DX} \tag{3.1}$$

berechnet. Eine Lifttrasse ist zumeist eine geradlinig verlaufende Strecke ohne stärkere Knicke oder abrupte Richtungsänderungen. Wenn ein Schifahrer einen Lift benützt, dann ist der Lift in der dazu gehörende GPS-Trajektorie in diesem Abschnitt eine Abfolge von Punkten, die alle einen ähnlichen Richtungswinkel haben.

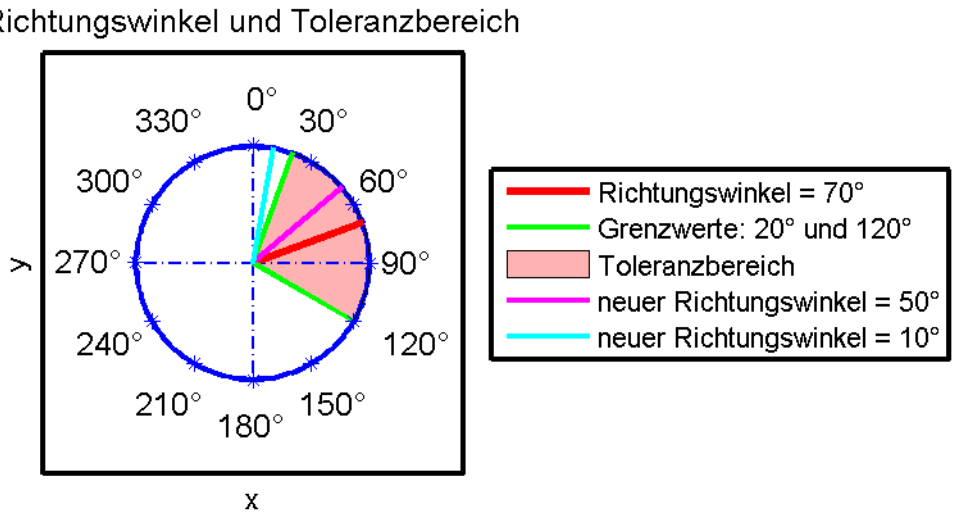


Abbildung 3.1.: Richtungswinkel und Grenzwert, Quelle: eigene Darstellung

In Abbildung 3.1 ist beispielhaft die Teilung einer Trajektorie anhand der Richtungswinkel dargestellt. Falls ein Richtungswinkel (magenta Linie) im Toleranzbereich (rot gefärbte Fläche) des ersten berechneten Richtungswinkels (rote Linie) eines Teilstückes liegt, dann werden die Punkte dem Teilstück des Ausgangsrichtungswinkels zugewiesen. Jeder weitere Punkt, der einen Richtungswinkel

begründet, welcher innerhalb des Toleranzbereiches des ersten Richtungswinkels liegt, wird zum Teilstück hinzugefügt. Falls ein Punkt einen Richtungswinkel begründet, der außerhalb des Toleranzbereiches liegt, wird das bisherige Teilstück abgeschlossen und ein neues Teilstück mit neuem Ausgangsrichtungswinkel begonnen. In Abbildung 3.2 sieht man die Teilung einer Trajektorie auf Teilstücke. Das Teilstück mit der Nummer 46 besteht aus einer größeren Anzahl von Punkten, deren Richtungswinkel sich alle innerhalb des selben Ausgangsrichtungswinkels befinden.

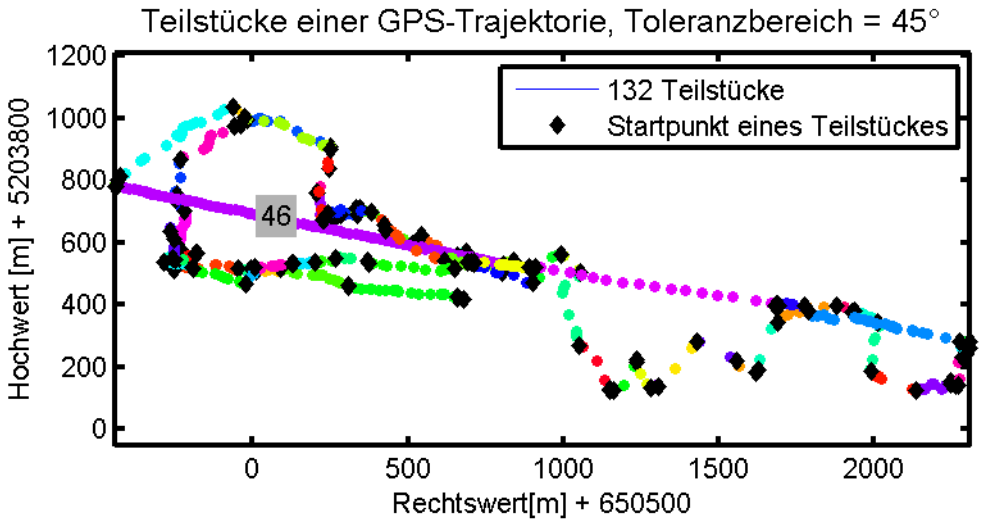


Abbildung 3.2.: Teilung einer Trajektorie in Teilstücke, Quelle: eigene Darstellung

Je kleiner der Toleranzwinkel gewählt wird, desto häufiger wird eine Trajektorie geteilt.

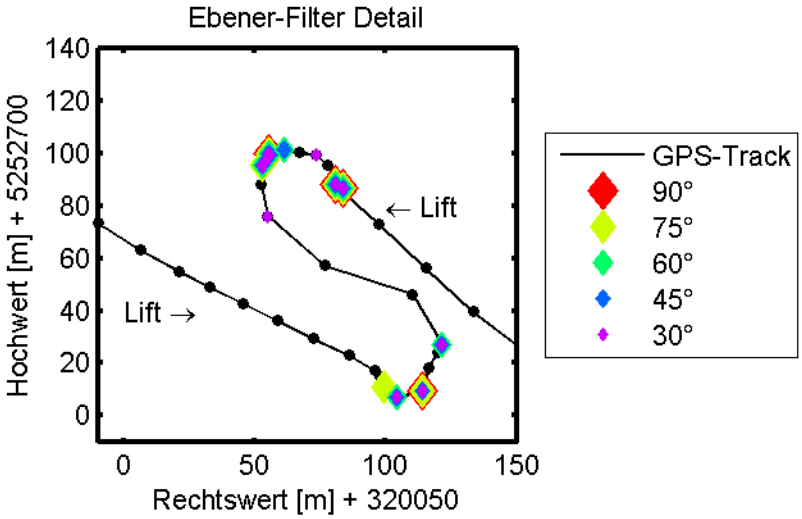


Abbildung 3.3.: Auswirkung unterschiedlicher Toleranzbereiche, Quelle: eigene Darstellung

In Abbildung 3.3 ist das Ergebnis für unterschiedliche Toleranzwinkel und wie sie sich auf die Spaltung einer Trajektorie auswirken, dargestellt. In der Abbildung sind zwei Lifte aus Saalbach dargestellt, einer kommt von rechts unten zur Mitte (Bernkogelbahn) und einer geht von der Mitte nach links oben weg (Bernkogelbahn III). Diese Information steht dem Algorithmus nicht zur Verfügung und wird an dieser Stelle nur vorweg genommen. Zwischen den beiden Liften gibt es ein Verbindungsstück. Unterschiedliche Toleranzbereiche führen zur Spaltung der Trajektorien an unterschiedlichen Stellen.
Ein zu großer Toleranzbereich hat zur Folge, dass Abschnitte nicht streng genug geteilt werden und somit im nachfolgenden Schritt Anstiege gefunden werden, die sehr lang sind und eine sehr große Höhendifferenz überwinden. Im Beispiel, dass in Abbildung 3.3 dargestellt ist, verursacht ein zu groß gewählter Toleranzbereich, dass das Teilstück nicht geteilt wird und somit beide Lifte als ein langer Anstieg detektiert werden. Ein zu kleiner Toleranzbereich zerstückelt eine Trajektorie stark. Die Gefahr besteht, dass eine Trajektorie zu oft gespalten wird und damit auch lange, zusammengehörende Anstiege, geteilt werden.

3.5. Positive Steigungen suchen

In diesem Schritt wird in den gespaltenen Teilstücke nach positiven Höhenunterschieden gesucht. Diese werden gefiltert und in den weiteren Schritten als Anstiege bezeichnet.

Neben der Eigenschaft, dass Lifte geradlinig verlaufen, ist ein weiteres Merkmal, dass Lifte Personen zumeist vom Tal auf den Berg befördern, also eine positive Höhendifferenz überwinden. Mit dieser Annahme fallen Talfahrten mit dem Lift aus den weiteren Berechnungen heraus. Positive Steigungen können auf zwei unterschiedlichen Arten bestimmt werden.

Variante 1

Ein Teilstück besteht aus mehreren hintereinander folgenden Punkten und zwischen zwei hintereinander folgenden Punkten wird der Höhenunterschied bestimmt. Variante 1 berechnet alle möglichen Kombinationen von Gesamthöhenunterschieden und sucht anhand nachfolgend aufgelisteter Kriterien die beste Lösung für positive Anstiege:

- maximaler Höhenunterschied
- geringste Anzahl an verwendeten Punkten

Im obersten Bild der Abbildung 3.4 ist das Ergebnis des ersten Kriteriums dargestellt. Durch unterschiedliche Kombinationen werden alle Punkte zu irgendeinem positiven Anstieg dazugezählt. Das zweite Kriterium, aufbauend auf den Kombinationen des ersten, dargestellt im mittleren Bild, teilt das Teilstück in zwei Anstiege, die beide den gleichen Gesamthöhenunterschied überwinden. Mit den beiden Kriterien wird gewährleistet, dass die positiven Anstiege eines Teilstückes einen maximalen Höhenunterschied überwinden und trotzdem kompakt bleiben.

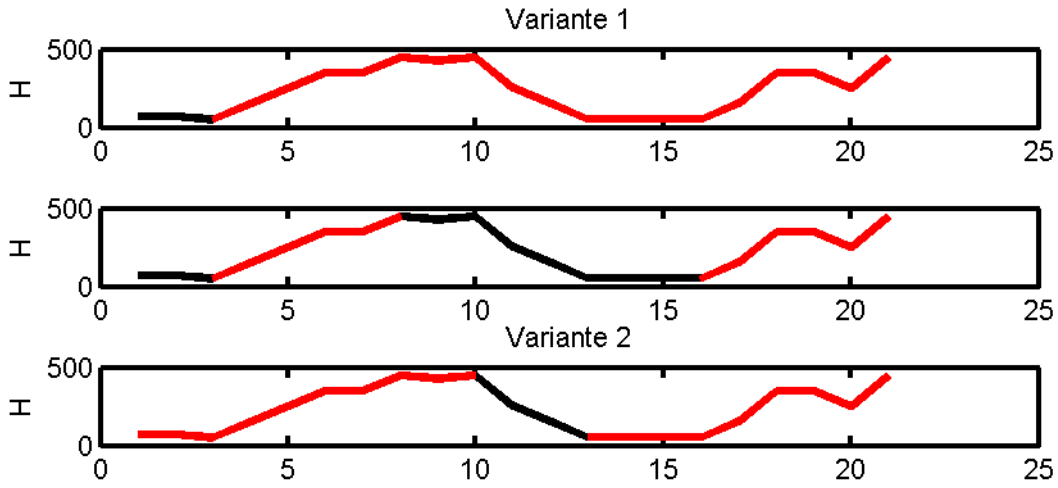


Abbildung 3.4.: Die ersten zwei Bilder gehören zur Variante 1 und das letzte zu Variante 2, Quelle: eigene Darstellung

Variante 2

Bei dieser Variante wird die Teilstrecke Punkt für Punkt durchlaufen. Pro Punkt wird untersucht, ob es einen positiven oder negativen Höhenanstieg zum Vorgängerpunkt gibt. Es gibt vier Arten von Höhenunterschieden

- negativer Höhenunterschied - Beginn eines negativen Teilstückes
- negativer Höhenunterschied - Ende oder Erweiterung eines negativen Teilstückes
- positiver Höhenunterschied - Beginn eines positiven Teilstückes
- positiver Höhenunterschied - Ende oder Erweiterung eines positiven Teilstückes

Falls eine Trajektorie mit einem negativen Teilstück beginnt, gefolgt von einem positiven Teilstück, dann wird der Gesamthöhenunterschied beider Teilstücke berechnet. Falls der negative Höhenunterschied geringer als der positive ist und geringer als ein vom User vorgegebener Grenzwert, dann wird der negative Anstieg zum positiven Anstieg

hinzugezählt. Gleiches gilt, wenn am Ende eines positiven Anstieges ein negatives Teilstück angrenzt. Im untersten Bild der Abbildung 3.4 sieht man den Unterschied zur Variante 1. Die negativen Höhenunterschiede am Anfang und Ende des ersten Anstieges werden zum positiven Anstieg hinzugezählt, da sie den Grenzwert unterschreiten. Der Grund für den Grenzwert ist im Unterabschnitt 2.1.1 beschrieben. Auch der Anfang des zweiten Anstieges ist anders, da Teilstücke ohne Höhendifferenz zum Anstieg hinzugezählt werden. Bei der Variante 1 werden gerade oder kurz abschüssige Teilstücke auf Grund des zweiten Kriteriums nicht zu den Anstiegen gezählt. Allgemein sollte der Grenzwert bei der zweiten Variante nicht zu groß gewählt werden und sich bei Liften im einstelligen Meterbereich befinden. Wird der Grenzwert zu null gesetzt, berechnet die 2. Variante das selbe Ergebnis wie die erste Variante.

Zusammenhang zwischen den Schritten *Trajektorie spalten* und *positive Steigungen suchen*

Im unteren Bild der Abbildung 3.5 sind die Höhendaten von zwei Liften und dem Verbindungsstück dazwischen gezeichnet. Es ist sind die selben Lifte, wie sie in Abbildung 3.3 in der Lage dargestellt sind.

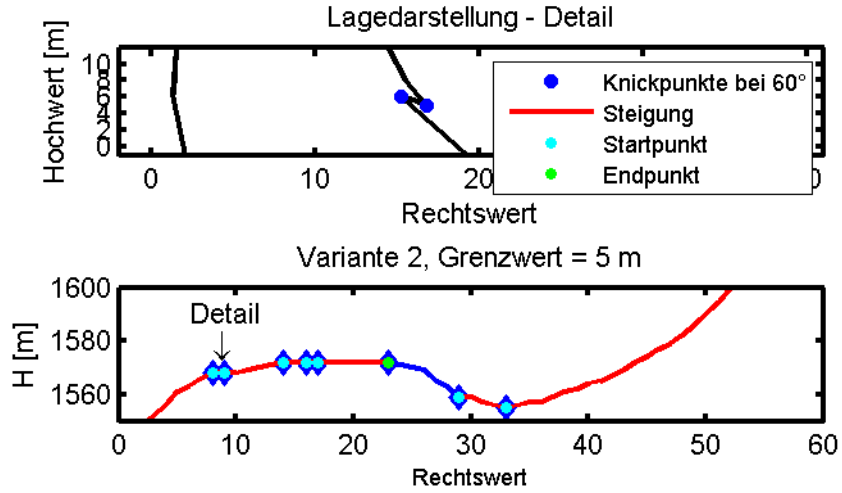


Abbildung 3.5.: Variante 2, Quelle: eigene Darstellung

Die blau gefärbten Punkte im oberen Bild der Abbildung 3.5 stellen die Teilungspunkte dar, wie sie im Schritt *Trajektorie spalten*, berechnet wurden. Positive Steigungen werden nur pro gespaltenem Teilstück gesucht. Damit wird verhindert, dass Teilstücke, die in der Ebene einen viel zu großen Knickpunkt aufweisen, als ein gemeinsamer Anstieg detektiert

werden. Der Schritt *Trajektorie spalten* bildet die Grundlage für den anschließenden Schritt *positive Steigungen suchen*. Je kleiner der Toleranzbereich gewählt wird, umso häufiger wird eine Trajektorie gespalten, desto größer ist die Gefahr, dass Teilstücke entstehen, die beide Teile von Anstiegen des selben Liftes beinhalten. Diese beiden Steigungen werden jedoch getrennt betrachtet, da sie in getrennten Teilstücken auftreten und damit wird die Detektion des Liftes aus diesen beiden Anstiegen unmöglich. Wird der Toleranzbereich zu groß gewählt, werden Anstiege gefunden, die eigentlich zwei Lifte darstellen. Auch in diesem Fall ist die Detektion eines einzelnen Liftes nicht möglich. Für jeden Anstieg wird die Länge und die überwundene Höhendifferenz bestimmt.
Jedes Schigebiet hat seine Eckdaten für Lifte, die den Homepages der Seilbahnbetreiber entnommen werden können. Interessant sind dabei der kürzeste und der längste Lift sowie der Lift, der die maximale Höhendifferenz im Schigebiet überwindet. Anhand der Eckdaten können die berechneten Anstiege in drei Bereiche sortiert werden:

- im ersten Bereich sind Anstiege, die kürzer als der kürzeste Lift sind
- im zweiten Bereich sind Anstiege, die kürzer als der längste Lift sind, sowie eine kleinere Höhendifferenz überwinden als die größte Höhendifferenz, die von einem Lift im Schigebiet überwunden wird
- im dritten Bereich sind Anstiege, die länger als der längste Lift sind und/oder eine größere Höhendifferenz überwinden als die größte Höhendifferenz die von einem Lift im Schigebiet überwunden wird

Die Anzahl von Anstiegen in den unterschiedlichen Bereichen ändert sich mit dem Toleranzbereich, der im Schritt *Trajektorie spalten* verwendet wird. Um den besten Toleranzbereich zu bestimmen, werden die Anstiege für unterschiedliche Toleranzbereiche bestimmt und sortiert.
Im ersten Bereich sammeln sich Anstiege, die zu kurz sind. Kurze Anstiege kommen vor allem deshalb zu Stande, weil es bei den Abfahrten auch kleinere Gegenanstiege geben kann. Im zweiten Bereich sammeln sich alle Anstiege, die potentielle Anstiege für Lifte darstellen. Je höher der Wert in diesem Bereich ist, desto wahrscheinlicher ist es, dass in den anschließenden Berechnungsschritten Lifte detektiert werden. Der Toleranzbereich mit den meisten Anstiegen in diesem Bereich wird als bester Toleranzwert für den Schritt *Trajektorie spalten* definiert. Die Anzahl der Anstiege im dritten Bereich sollte so niedrig wie möglich sein. Die Anzahl an Pfaden in diesem Bereich nimmt mit der Größe des Toleranzbereiches zu, da Trajektorien nicht mehr oft genug gespalten werden. Die Anzahl

von Anstiegen im dritten Bereich eines großen Toleranzbereiches muss immer in Relation zur Anzahl der Anstiege im dritten Bereich eines kleinen Toleranzbereiches gesehen werden. Die Anstiege im dritten Bereich, die bereits bei einem kleinen Toleranzbereiche auftauchen, sind größtenteils auf Datenfehler bzw. schlecht gemessene Trajektorien zurückzuführen.

3.6. Positive Steigungen eliminieren

In diesem Schritt werden zu kurze Anstiege eliminiert.

Es gibt zwei Arten, wie dieser Schritt angewendet werden kann.

1. keine Steigungen eliminieren

Bei diesem Ansatz werden keine Steigungen eliminiert. Der Vorteil ist, dass alle Steigungen behalten werden und jede Steigung Teil eines Liftes sein kann. Der Nachteil ist, dass in nachfolgenden Schritten auch Lifte detektiert werden, die nur einige Meter lang sind. Vor allem in Regionen, wo Schifahrer stehen bleiben (Lifteintrittspunkte), kann es eine größere Anzahl von kleinen Anstiegen geben. Diese Variante ist die sichere Variante, da keine Anstiege entfernt werden. In der Praxis macht es wenig Sinn, kurze Anstiege nicht zu entfernen, da spätesten im Schritt 4.10 die Sinnhaftigkeit wenige Meter langer Liften bezweifelt werden kann.

kurze Steigungen eliminieren

Wie bereits in Schritt 3.5 erwähnt, können die Eckdaten der Lifte den Homepages der Liftbetreiber entnommen werden. Als Näherung für zu kurze Steigungen kann die kürzeste Distanz eines Liftes eines Schigebietes herangezogen werden. Alle Anstiege, die kürzer sind als die Näherung, werden eliminiert. Der Vorteil dieser Variante ist, dass alle Anstiege, die übrig bleiben, Teile eines Liftes sein können. Alle Lifte, die in den nachfolgenden Schritten berechnet werden, sind länger als die Näherung. Ein praktischer Vorteil ist, dass die mit der Eliminierung von kurzen Steigungen verbundene Reduktion der Datenpunkte eine Ersparnis in der Rechenzeit für die Berechnung des OPTICS-Algorithmus mitbringt.
Es macht keinen Sinn, Anstiege auf Grund ihrer überwundenen Höhendifferenzen zu filtern, da es Lifte gibt, die sehr lang sind und trotzdem nur wenige Höhenmeter überwinden. Mit diesem Schritt ist die Suche nach Anstiegen in den einzelnen

Trajektorien abgeschlossen. Übrig bleiben Anstiege, die in den nächsten Schritten näher untersucht und zu potentiellen Liften zusammengefasst werden.

3.7. Daten ordnen

In diesem Schritt werden die Datenpunkte aller Anstiege mit Hilfe des OPTICS-Algorithmus geordnet.

In diesem Schritt werden als erstes die Distanzen von jedem Punkt zu allen anderen Punkten im Datensatz bestimmt, wie im Unterabschnitt 2.6.1 erklärt. Die Distanzen werden mit dem vom User vorgegebenen ϵ-Wert gefiltert. Für jeden Punkt wird ein Textfile erstellt, in welches die Indizes der Punkte und die Distanzen zu selbigen in der ϵ-Nachbarschaft des untersuchten Punktes geschrieben werden.

Variante 1: ϵ=∞

Wenn der ϵ-Parameter auf ∞ gesetzt wird, dann liegt jeder Punkt in der ϵ-Nachbarschaft jedes anderen Punktes im Datensatz. Dadurch gehört jeder Punkt der gleichen Teilmenge an.

Variante 2: ϵ wird mit einem Wert festgelegt

Wenn dem ϵ-Parameter ein Wert zugewiesen wird, dann ist nicht jeder Punkt in der ϵ-Nachbarschaft jedes anderen Punktes. Dadurch werden zwar die Daten geordnet, es werden die Punkte aber unterschiedlichen Teilmengen zugewiesen. Der Speicherplatzaufwand verringert sich genauso wie die Berechnungszeit.

Als nächstes wird der OPTICS-Algorithmus ausgeführt und das Ergebnis des Algorithmus im Textfile *OrderedFile.txt* gespeichert.

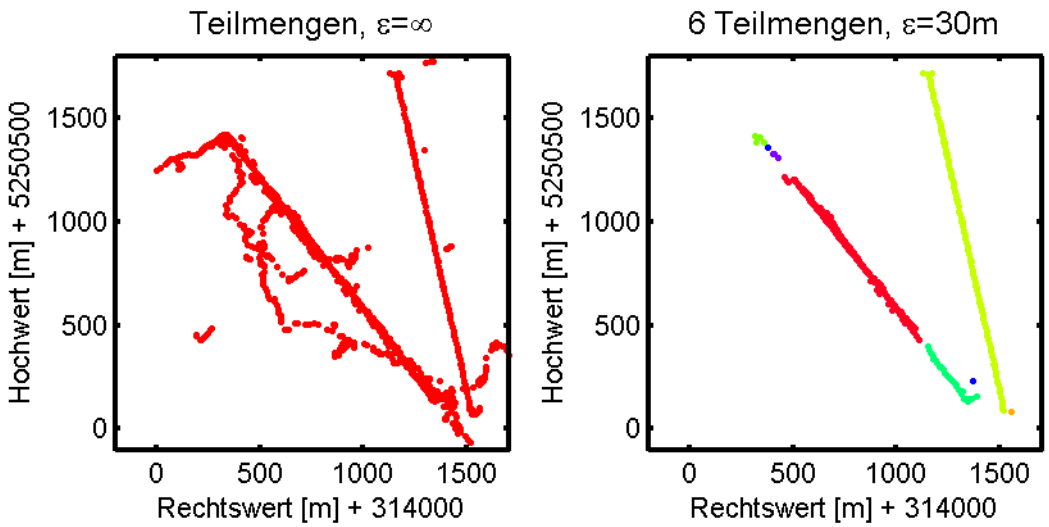


Abbildung 3.6.: Detailausschnitt aus dem Datensatz Saalbach, Quelle: eigene Darstellung

Im linken Bild der Abbildung 3.6 sind beispielhaft Teilmengen, die nach der Berechnung des OPTICS-Algorithmus extrahiert werden, abgebildet. Für das Ergebnis im linken Bild wird der ϵ-Parameter auf den Wert ∞ gesetzt und im Schritt *positive Steigungen eliminieren* keine Steigungen eliminiert. Alle Punkte gehören daher einer Teilmenge an. Für das Ergebnis im rechten Bild werden zu kurze Steigungen eliminiert und der ϵ-Parameter auf den Wert 30 Meter gesetzt.

3.8. Teilmengen berechnen und Cluster bestimmen

In diesem Schritt werden Teilmengen anhand der Erreichbarkeitsdistanzen der Datenpunkte extrahiert. Falls die Teilmengen die Bedingungen für Cluster erfüllen, werden daraus Cluster.

Ausgangspunkt für die Berechnungen in diesem Schritt sind die extrahierten Teilmengen aus Schritt 3.7. In jeder Teilmenge wird nach weiteren Teilmengen gesucht, bis die Anzahl von Punkten pro Teilmenge einen minimalen Wert unterschreitet. Dabei wird nach der dritten Variante, die im Unterabschnitt 2.6.4 beschrieben ist, vorgegangen. Die Näherung für die kleinste Datenanzahl pro Teilmenge wird anhand der Definition auf S. 46 berechnet. Je nachdem, ob positive Steigungen im Schritt 3.6 eliminiert werden oder nicht, verändert sich die kleinste Anzahl von Punkten pro Teilmenge. Wenn der ϵ-Parameter im Schritt 3.7 nicht beschränkt wird, findet der Suchalgorithmus im Datensatz weniger viele Teilmengen, als wenn der ϵ-Parameter beschränkt worden wäre. Dafür ist die Punkteanzahl pro Teilmenge höher, da die Gesamtdatenanzahl größer ist.

Damit aus einer Teilmenge ein Cluster wird, müssen die Teilmengen aus Daten von unterschiedlichen Pfaden bestehen. Die Definition eines Pfades kann unterschiedlich streng ausgelegt werden:

- ein Pfad kann *eine* positive Steigung *einer* Trajektorie sein
- ein Pfad kann *eine* gesamte Trajektorie sein

Wie streng die Unterscheidung durchgeführt wird, kann vom User bestimmt werden und wird als strenge (ein Pfad = eine Trajektorie) oder weniger strenge Lösung (ein Pfad = eine Steigung einer Trajektorie) bezeichnet. Erst aus mehreren unterschiedlichen Anstiegen wird ein Lift berechnet. Ein Schifahrer kann einen Lift öfters als einmal befahren. Somit gibt es in der Trajektorie eines Schifahrers mehrere Anstiege, die dem selben Lift zugeordnet werden. Wenn die Daten, die der Schifahrer aufzeichnet, grob falsch sind, dann können bereits die Daten eines einzelnen Schifahrers einen möglichen Lift definieren, wo es gar keinen geben darf. Mit der strengeren Variante wird bei der Bestimmung von Clustern aus Teilmengen nur eine Steigung pro Trajektorie als Pfad gezählt. Der Wert für die Bedingung, dass aus einer Teilmenge ein Cluster wird, muss vom Benützer der Software vorgegeben werden.
Die Bedingung, dass es eine minimale Anzahl von irgendeinem Wert geben muss, damit ein Objekt oder ein Cluster erkannt wird, hat den Grund, dass gewisse Objekte öfters als einmal in unterschiedlichen Trajektorien auftauchen müssen, damit nicht Einzelfälle selbige begründen. Bei Liften kann sonst ein einzelner Tourengeher, der neben einer Piste geradeaus nach oben geht, einen Lift begründen. Gleiches gilt für Abfahrten im freien Gelände, wo einzelne Schifahrer dem Algorithmus einen validen Cluster und damit eine mögliche Piste vorgaukeln, wo es keine Piste geben darf. Der Grenzwert für die Bedingung der Transformation von Teilmengen zu Clustern sollte mit ansteigender Datenanzahl angepasst werden. Grundsätzlich kann bei jedem Schritt der Wert verändert werden, es sollte aber bei jedem Schritt der gleiche Wert als minimale Anzahl verwendet werden.

3.9. Lifte extrahieren

In diesem Schritt werden aus den Clustern Lifte gemittelt.

Die Lifte werden, wie im Abschnitt 2.7 beschrieben, bestimmt. Der Schritt besteht aus zwei Teilschritten. Im ersten Teilschritt werden iterativ die Cluster gefiltert, die Daten

von potentiellen Liften beinhalten. Pro Iteration werden zwei Funktionen ausgeführt. Die erste Funktion benötigt als Eingabeparameter eine minimale Anzahl von Pfaden, die zwei Cluster gemeinsam haben müssen, um zusammengefügt zu werden. Je höher die Variable gesetzt wird, desto weniger Cluster werden zusammengefügt. In der zweiten Funktion werden alle Pfade pro Cluster untereinander verglichen. Der Wert der maximalen Abweichung, den sich ein Pfad von anderen Pfaden des gleichen Clusters unterscheiden darf damit er nicht aus dem Cluster eliminiert wird (siehe Abschnitt 2.7), ändert sich pro Iteration selbstständig. Aus der Summe der Abweichungen der behaltenen Pfade aller Cluster wird nach jeder Iteration eine neue maximale Abweichung berechnet. Da in der zweiten Funktion immer wieder Pfade aus Clustern eliminiert werden, muss nach der zweiten Funktion jeder Cluster auf die Bedingung eines Clusters getestet werden. Dazu muss die Anzahl der unterschiedlichen Pfade einen minimalen Wert betragen. Als minimale Anzahl wird der gleiche Wert verwendet wie im Schritt 3.8. Wenn die Bedingung nicht erfüllt wird, wird der Cluster gelöscht. Danach wiederholt sich die Berechnung mit den verbliebenen Clustern, bis die neue maximale Abweichung eine vom Benützer der Software definierte maximal erlaubte Abweichung unterschreitet. Im zweiten Teilschritt werden aus den übrig gebliebenen Clustern Lifte berechnet.
Die erste Funktion vereint Cluster, die aus den Daten gleicher Anstiege bestehen. Vor allem, wenn der ϵ-Parameter im Schritt 3.8 beschränkt wird, kann es pro Lift mehrere Cluster geben. In jedem Cluster sind unterschiedliche Anstiege vereint. Nicht jeder Anstieg gehört aber dem gleichen Lift an. Vor allem in Regionen, wo mehrere Lifte zusammenkommen, kann dieser Fall auftreten. Die Anstiege werden in der zweiten Funktion gefiltert. Auch aus den aussortierten Anstiegen wird versucht, einen neuen Cluster zu bilden. Damit wird gewährleistet, dass bei Clustern, die Anstiege von zwei Liften beinhalten, die Anstiege eines Liftes behalten und die Anstiege des anderen Liftes eliminiert werden. Da die kleinste Abweichung pro Iteration aus den Daten der übrig gebliebenen Cluster berechnet wird, wird dieser Wert pro Iteration immer kleiner. Damit passen pro Iteration auch die Anstiege pro Cluster besser zusammen. Die kleinste Abweichung kann so oft verbessert werden, bis pro Cluster nur mehr wenige Anstiege übrig bleiben. Darum wird die kleinste maximale Abweichung festgelegt. Falls die kleinste Abweichung diesen Wert unterschreitet, werden die Iterationen gestoppt. Übrig bleiben Cluster, die aus Anstiegen bestehen, die sehr gut zusammen passen und aus diesen Anstiegen wird eine gemittelte Lifttrasse, wie im Abschnitt 2.7 beschrieben, bestimmt.

3.10. Lifte löschen und Lifte bestimmen

In diesem Schritt können berechnete Lifte gelöscht werden.

In diesem Schritt werden Lifte entfernt, die nicht eine gewisse Länge haben. Dieser Schritt ist vor allem interessant, falls im Schritt 3.6 kurze Anstiege nicht entfernt werden. Als Näherung für die Minimallänge eines berechneten Liftes kann die angegebene Länge des kürzesten Liftes auf der Homepage des Seilbahnbetreibers angenommen werden.
Einige Lifte verlaufen parallel zueinander oder überlagern sich ein Stück. Diese Lifte können anhand ihrer Längen bzw. der Anzahl von Pfaden, aus denen sie berechnet werden, von falsch detektierten Liften unterschieden bzw. gefiltert werden.

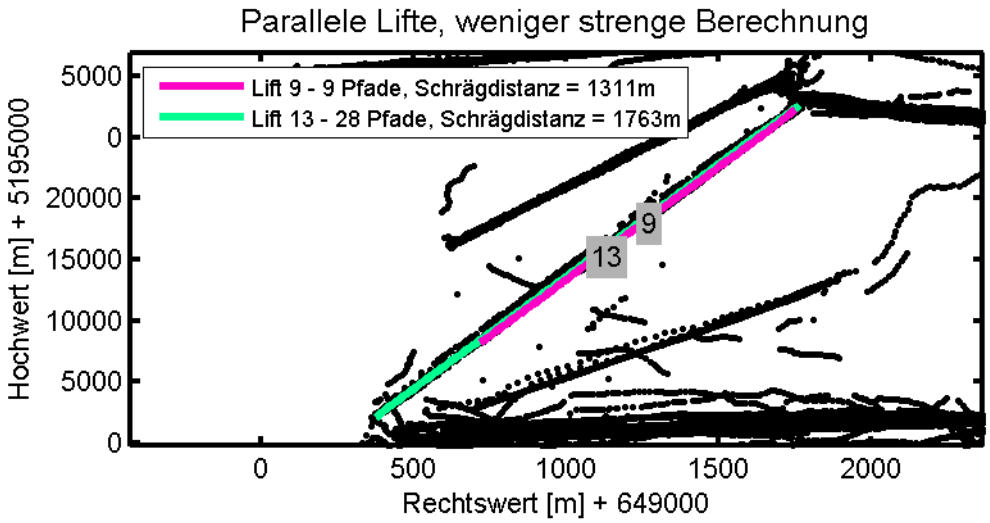


Abbildung 3.7.: Zwei parallel Lifte, Quelle: eigene Darstellung

In Abbildung 3.7 sind zwei parallele Lifte dargestellt, wobei ein Lift kürzer als der andere ist. Im dargestellten Fall sind beide „Liftanwärter“ auch tatsächlich Lifte, was auch anhand der Anzahl von verwendeten Pfaden pro Lift zu vermuten ist.
Es gibt Lifte, die teilweise fast exakt übereinander liegen. Bei Schleppliften kann dies der Fall sein, wenn z.B. in der Trajektorie eines Schifahrers ein Anstieg gegeben ist, wo der Schifahrer beim Liftfahren gestürzt ist. Daher gibt es für einen solchen Lift einen Satz von parallelen, fast gleich langen Anstiegen, wobei ein Anstieg kürzer als die anderen ist. Weil der Algorithmus nach allen möglichen Varianten von Liften im Schritt 3.9 sucht, kann es pro Lift mehr als einen Anwärter geben. Der korrekte „Liftanwärter“ kann zumeist anhand der Anzahl von verwendeten Pfaden für die Berechnung des gemittelten Liftes von falschen „Liftanwärtern“ unterschieden werden (siehe Abb. 3.8).

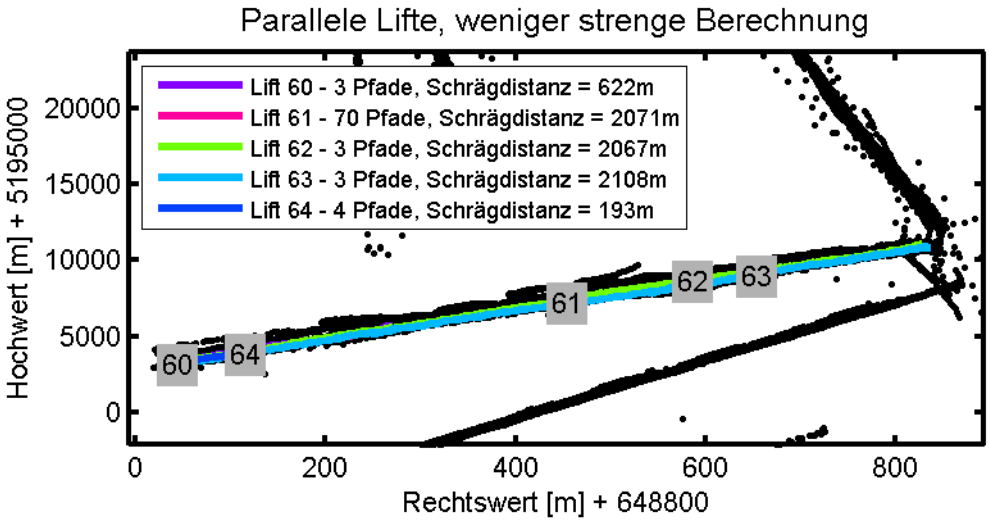


Abbildung 3.8.: Mehrere „Liftanwärter“ pro Lift, Quelle: eigene Darstellung

In Abbildung 3.8 wird der „Liftanwärter“ 61 als korrekter Lift angenommen. Nur anhand der Längen kann nicht auf den richtigen „Liftanwärter“ geschlossen werden, auf Grund der Anzahl von verwendeten Anstiegen zur Berechnung des gemittelten Liftes jedoch schon. In diesem Schritt können auch Lifte gelöscht werden, die es nicht mehr gibt oder die ersetzt wurden. Mit diesem Schritt ist die Berechnung der Lifte abgeschlossen.

3.11. Daten für Pisten vorbereiten

In diesem Schritt werden die Daten für die Berechnung der Pisten vorbereitet.

Für die Berechnung der Pisten werden alle Daten verwendet, die nicht für die Berechnung der Lifte verwendet wurden. Ab diesem Schritt sind „Abfahrten“ die Verbindungsstücke zwischen zwei Anstiegen. Eine einzelne Abfahrt begründet noch keine Piste.

3.12. Abfahrten den Liften zuordnen

In diesem Schritt werden die Abfahrten Liften zugewiesen.

Um jeden Anfangs- und Endpunkt eines Liftes wird ein Radius gelegt. Jede Abfahrt wird separat untersucht. Falls ein Punkt einer Abfahrt innerhalb des Radius um einen Liftpunkt liegt, wird die Abfahrt dem jeweiligen Lift zugewiesen. Eine Abfahrt kann auch mehrere Radien kreuzen und wird somit auch mehreren Liften zugewiesen. Viele Abfahrten werden Liften zugewiesen, die der Schifahrer nicht benutzt hat, da viele Pisten

an Lifteintrittspunkten zusammenlaufen, aber nicht zwangsweise auch dort aufhören. Als Beispiel kann man sich eine Piste vorstellen, die einerseits zu einer Mittelstation hinläuft und andererseits von dort aus weiter ins Tal führt. Ein weiteres Beispiel sind Lifte, deren Ein- oder Austrittspunkte sehr nah beieinander liegen und die Abfahrten eines Schifahrers zwangsweise mehr als einen Radius kreuzen.

3.13. Pistenkandidaten bestimmen

In diesem Schritt werden potentielle Pistenkandidaten zwischen zwei Liftpunkten bestimmt.

Wie im Abschnitt 2.8 beschrieben, gibt es in einem Schigebiet unterschiedliche Arten von Pisten. Die Abfahrten wurden bereits in Schritt 3.12 den Liften zugewiesen. Ein potentieller Pistenkandidat ergibt sich, wenn zwischen zwei Seilbahnen zumindest eine vom Benützer der Software vorgegebenen Anzahl von Pfaden verlaufen. Der Wert sollte nicht zu niedrig angesetzt werden, da sonst bereits einzelne Schifahrer Pisten begründen können. Problematisch kann dies im freien Gelände sein. Nur Abfahrten, die Teil eines Pistenkandidaten sind, kommen für die Berechnung von Pisten in Frage.

3.14. Pisten aus Pistenkandidaten berechnen

In diesem Schritt werden die Pistenkandidaten näher untersucht, gefiltert und gemittelte Pisten berechnet.

Jeder potentielle Pistenkandidat besteht aus einer Anzahl von unterschiedlichen Abfahrten. Wie im Abschnitt 2.8 beschrieben, wird zunächst ein Raster über die Abfahrten gelegt und jeder Punkt einer Abfahrt einer Rasterzelle zugewiesen. Je größer die Datendichte ist, desto kleiner kann die Rasterbreite gewählt werden. Eine kleine Rasterbreite abstrahiert eine Piste genauer als eine große. Allerdings muss bei kleinen Rasterbreiten darauf geachtet werden, dass durch eine Rasterzelle eine minimale Anzahl von unterschiedlichen Abfahrten verlaufen muss. In Gebieten, wo die Datendichte gering ist, kann eine zu kleine Rasterbreite dazu führen, dass Pisten nicht detektiert werden, weil zu wenige Punkte von Abfahrten pro Rasterzelle gegeben sind. Darum muss bei solchen Datensätzen eine größere Rasterbreite verwendet werden. Eine große Rasterbreite kann dazu führen, dass z.B. Gabelungen von Pisten verloren gehen. Eine durchschnittliche Piste ist breiter als 50 Meter.
Zwischen zwei Liftpunkten kann es mehr als eine Piste geben, also ist auch die

Berechnung der Pisten ein iterativer Vorgang. Die Start- und Endpunkte der Abfahrten sind bekannt, da die Abfahrten den Liften zugewiesen werden. Vom Startpunkt einer Piste wird nach dem im Abschnitt 2.8 beschriebenen Verfahren zunächst eine Piste entlang der Rasterzellen bestimmt. Da zwischen zwei Liftpunkten mehr als eine Piste bestehen kann, wird dieser Vorgang so lange wiederholt, bis die Anzahl an übrig bleibenden Abfahrten für die Berechnung einer weiteren Piste nicht mehr ausreichen. Diese Berechnung wird für jeden Pistenkandidaten durchgeführt und daraus die mittleren Pisten pro Pistenkandidat entlang der Rasterzellen berechnet.
Pro Rasterzelle wird ein Mittelwert aus den Datenpunkten der Abfahrten, die durch die Zelle verlaufen und für die Berechnung einer mittleren Piste verwendet werden, berechnet. Somit erhalten unterschiedliche Pisten, wenn sie durch die selbe Rasterzelle verlaufen, den gleichen Pistenpunkt für die jeweilige Rasterzelle zugewiesen. Je dichter die Anzahl von Punkten in einem Gebiet ist, desto kleiner kann das Raster gesetzt werden. Je kleiner die Rasterbreite ist, desto genauer kann eine mittlere Piste extrahiert werden. Aus den mittleren Pisten entlang der Rasterzellen werden zum Abschluss dieses Schrittes mittlere Pisten entlang der Mittelwerte berechnet. Mit diesem Schritt ist die Berechnung der Pisten abgeschlossen.

3.15. Rasterverbindungen bestimmen

In diesem Schritt werden die Verbindungen zwischen Rasterzellen berechnet.

Bis zu diesem Schritt werden nur Pisten bestimmt, die unterschiedliche Bedingungen erfüllen, z.B. können Pisten nur gemittelt werden, falls zwischen zwei Liften eine minimale Anzahl von unterschiedlichen Pfaden verläuft. Eine weitere Einschränkung ist, dass Pisten zumeist nur entlang gewisser Rasterzellen gefunden werden, meistens sind dies die Rasterzellen, die die Mitte einer Piste bedecken. Dieser Effekt entsteht dadurch, dass Schifahrer am häufigsten in der Mitte einer Piste fahren und nicht an den Rändern, also die Rasterzellen, die entlang der Mitte verlaufen, mehr Datenpunkte beinhalten, was sich bei der Bewertung der Rasterzellen bemerkbar macht. Aus diesem Grund wird die volle Breite einer Piste in Schritt 3.14 selten detektiert.
In diesem Schritt wird ein Raster über das Schigebiet gelegt und jeder Punkt einer Abfahrt einer Rasterzelle zugewiesen. Falls zwischen zwei benachbarten Rasterzellen mehrere Pfade verlaufen, dann gibt es zwischen den zwei Rasterzellen eine Verbindung. Zusätzlich kann man die Richtung der Verbindung ermitteln, indem man ebenso die Reihenfolge der Pfadpunkte entlang der Rasterzellen detektiert. Die Anzahl von

unterschiedlichen Pfaden, die zwischen zwei Zellen gegeben sein müssen um eine valide Verbindung zu begründen, kann vom Benützer der Software festgelegt werden. Der Vorteil diese Schrittes ist, dass ein gerichtetes Netz von Verbindungen über das Schigebiet berechnet wird. Weiteres wird der Nachteil des Schrittes 3.14, wo Pisten vor allem entlang der Mitte einer Piste detektiert werden, ausgeglichen, weil mit diesem Verfahren jegliche Verbindungen zwischen zwei Rasterzellen detektiert werden und damit die Breite einer Piste besser extrahiert wird.

4. Ergebnisse

Für die praktische Berechnung wurden zwei Datensätze von der Seite „www.gpsies.com“ heruntergeladen. Die Entscheidung für die Skigebiete Skicircus Saalbach Hinterglemm Leogang (kurz Saalbach) und Sölden wurde bewusst gewählt. Saalbach ist ein sehr großflächiges Skigebiet mit über 200 Pistenkilometern. Damit ist Saalbach gemessen an den Pistenkilometern das drittgrößte Skigebiet in Österreich. Die Datendichte für dieses Gebiet war mit 21 Tracks und etwas mehr als 60000 Punkten „überschaubar“. Der zweite Datensatz war Sölden und bestand aus 78 Tracks mit etwas weniger als 400000 Punkten. Das Schigebiet Sölden hat mehr als140 Pistenkilometer zu bieten.
Der Datensatz Saalbach war um einiges kleiner als der Datensatz Sölden, hatte aber den Vorteil, dass die Berechnungen schneller durchgeführt werden konnten. Weiters gab es keine Sonderfälle, wie etwa Tunnel. Der Datensatz Sölden hatte mehr Daten. Das Schigebiet hat auch im Sommer einen Skibetrieb, da sich aber das Schigebiet im Winter vom Schigebiet im Sommer unterscheidet, wurden nur Daten von Tracks, die in den Wintersaisonen aufgenommen wurden, verwendet.
Durch die Wahl der unterschiedlich großen Datensätze, konnten die Stärken und Schwächen des Algorithmus besser hervorgehoben werden. Das Ergebnis jedes Schrittes konnte in der Software grafisch dargestellt werden. Um die Ergebnisse in *Google Earth* betrachten zu können, wurde jedes Ergebnis in ein KML-File geschrieben. Die nachfolgenden Schritte wurden in der gleichen Reihenfolge gelistet, wie sie im Kapitel 3 beschrieben sind.

4.1. Daten einlesen

Die Daten beider Datensätze wurden nach dem gleichen Verfahren eingelesen. In Abbildung 4.1 wurden die Daten des Schigebietes Saalbach dargestellt. Der Datensatz hatte 20 Trajektorien mit insgesamt 60646 Datenpunkten. Im Schigebiet waren in der Wintersaison 2014/15 56 Seil- und Seilbahnanlagen in Betrieb und mehr als 200 Pistenkilometer befahrbar (siehe „www.saalbach.com“). Die Daten, die verwendet wurden, stammten aus den Wintersaisonen 2008 bis 2013.

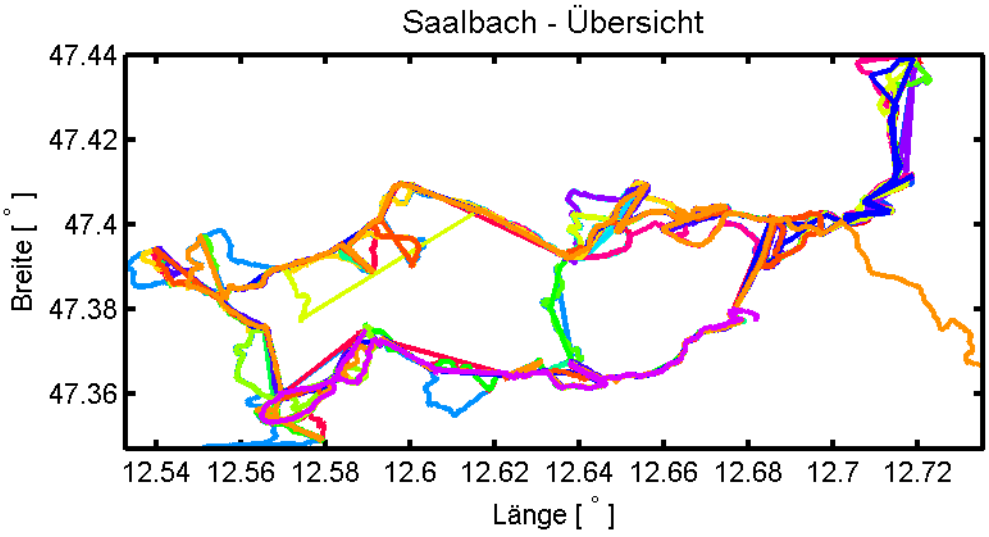


Abbildung 4.1.: Datensatz Saalbach, Quelle: eigene Darstellung

Der Datensatz Sölden (siehe Abb. 4.2) hatte 78 Trajektorien mit 398696 Datenpunkten.

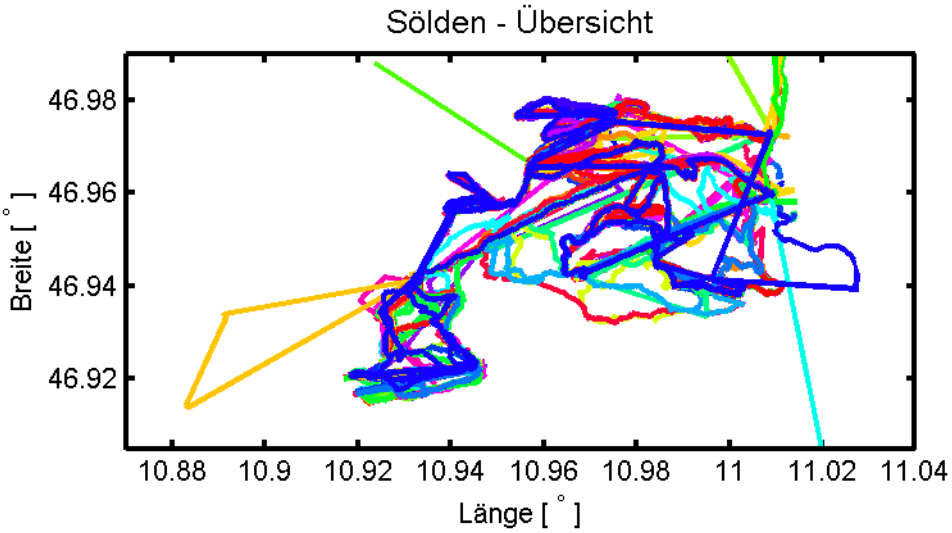


Abbildung 4.2.: Datensatz Sölden, Quelle: eigene Darstellung

Sölden hatte im Jahr 2014/15 33 Seilbahnanlagen in Betrieb und 109 befahrbare Pistenkilometer. Die Daten des Datensatzes Sölden stammten aus den Jahren 2008 bis 2014. Eine Trajektorie beinhaltete sowohl Daten aus dem Schigebiet Sölden als auch aus dem Schigebiet Obergurgl.

4.2. Daten glättem

In beiden Datensätzen wurde keine Trajektorien geglättet.

4.3. UTM Koordinaten berechnen

Die UTM-Koordinaten aus den ellipsoidischen Koordinaten für beide Datensätze wurden mit den Formeln, die im Unterabschnitt 2.4.2 beschrieben wurden, berechnet.

4.4. Trajektorie spalten

Das Ergebnis dieses Schrittes waren zu Teiltrajektorien gespaltene Trajektorien. Teiltrajektorien, die potentielle Lifte darstellten, waren lange Teilstücke mit ähnlich bleibenden Richtungswinkeln (siehe Abb. 4.3). Abfahrten bestanden zumeist aus mehreren kurzen Teiltrajektorien, da Abfahrten Kurven hatten und sich der Richtungswinkel auch dementsprechend stark änderte.

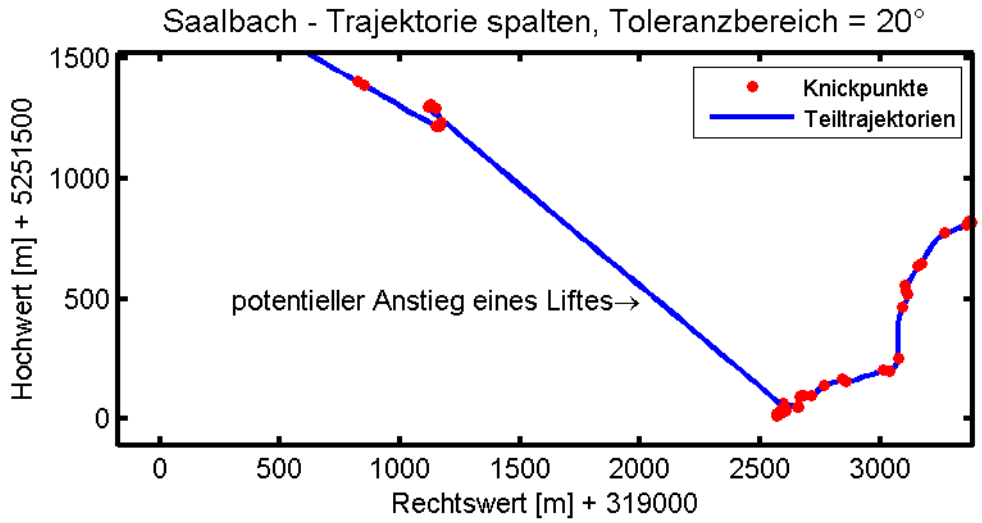


Abbildung 4.3.: Teiltrajektorie, die einen Lift beinhaltet, Quelle: eigene Darstellung

Für den Datensatz Saalbach wurden 20° als Toleranzwert gewählt. Für dem Datensatz Sölden wurden 25° als Toleranzwert gewählt. Die Erklärung für die Wahl der Parameter wurde im Schritt 4.5 beschrieben. In Abbildung 4.3 wurden die selben zwei Lifte dargestellt wie sie in Abbildung 3.3 bereits als Beispiel dargestellt wurden.

4.5. Positive Steigungen suchen

In diesem Schritt wurden aus den im Schritt 4.4 berechneten Teiltrajektorien die positiven Anstiege gefiltert. Die Anstiege für beide Datensätze wurden nach der zweiten Variante (siehe Schritt 3.5) bestimmt. Als Grenzwert für die Variante zwei wurden 5

Meter gewählt.
Der Schritt war entscheidend für die Wahl des Toleranzbereiches im Schritt 4.4.

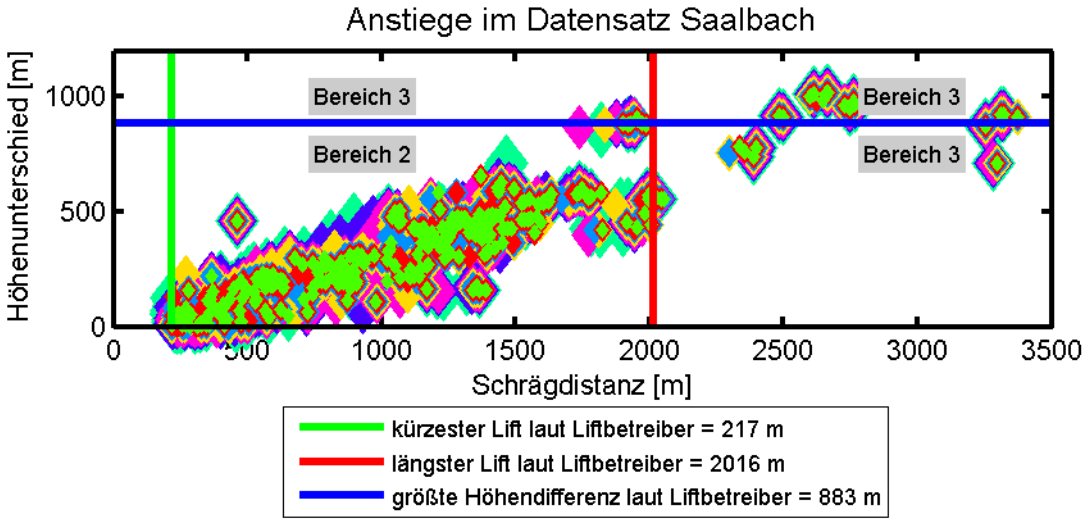


Abbildung 4.4.: Verteilung der Anstiege in Bereiche, Quelle: eigene Darstellung

Die Abbildung 4.4 wurde in zwei Bereiche geteilt. Als Näherung für die Grenzen der Bereiche wurden die Angaben, die der Seilbahnbetreiber auf seiner Homepage angab, verwendet. Die rote Linie war der längste Lift (Bernkogelbahn I) im Schigebiet Saalbach und die blaue Linie stellte die größte überwundene Höhendifferenz eines Liftes (Zwölferkogel Nordbahn) dar. Der Graf begann mit der grünen Linie, die die kürzeste Länge eines Liftes (*Oberschwarzachlift*) im Schigebiet darstellte. Die Steigungen, die berechnet wurden, wurden in drei Bereiche unterteilt. Der erste Bereich umfasste alle Steigungen, die kürzer als der kürzeste Lift im Schigebiet waren - links von der grünen Linie und in der Abbildung nicht dargestellt. Dem zweiten Bereich wurden alle Steigungen zugewiesen, die länger als der kürzeste Lift waren und eine kleinere Höhendifferenz als die größte Höhendifferenz (blaue Linie) überwanden. Im dritten Bereich wurden alle Steigungen gesammelt, die zu lang waren oder eine zu große Höhendifferenz aufwiesen. In der Tabelle 4.1 wurde die Aufteilung von Steigungen pro Bereich für verschiedenste Toleranzbereiche des Datensatzes Saalbach aufgelistet. Die Anzahl der Steigungen im ersten Bereich nahm mit wachsendem Toleranzbereich ab, weil die Trajektorien weniger oft geteilt wurden. Die Anzahl der Steigungen im dritten Bereich war bei kleinen Toleranzbereichen ähnlich hoch, wie bei größeren.

Toleranz [°]	Bereich 1	Bereich 2	Bereich 3
10	16930	327	16

20	13210	335	19
30	11125	318	21
40	9686	305	23
50	8738	297	26
60	7990	300	27
70	7437	298	27

Tabelle 4.1.: Summe der Steigungen pro Bereich - Saalbach

In Abbildung 4.5 wurden die Steigungen, die dem dritten Bereich zugeordnet wurden, beispielgebend für zwei Toleranzbereiche dargestellt.

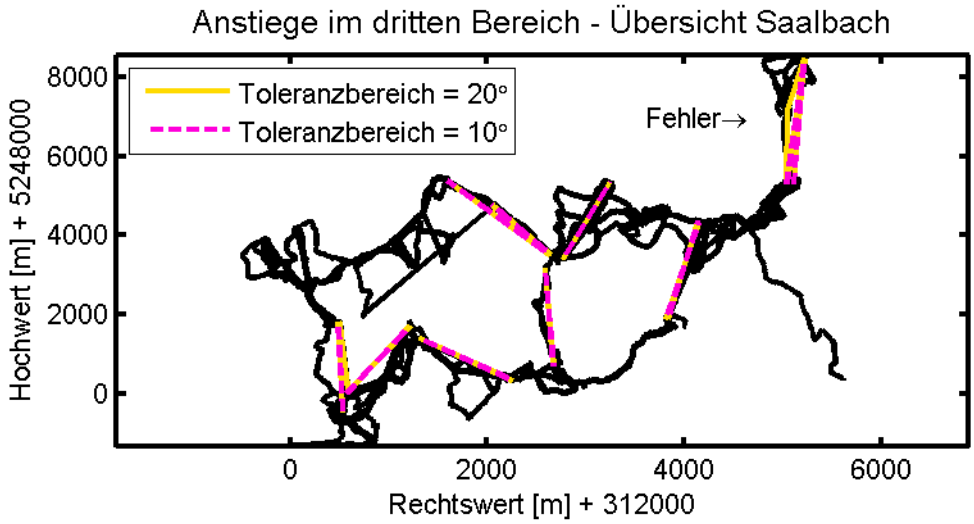


Abbildung 4.5.: Steigungen im dritten Bereich, Quelle: eigene Darstellung

Anstiege im dritten Bereich für den Toleranzbereich von 10°, waren auf Datenfehler zurückzuführen bzw. auf ungenaue Messungen. Mit größer werdendem Toleranzbereich nahm die Anzahl von Steigungen im dritten Bereich zu. In der Abbildung wurde als Beispiel ein „Fehler“ eingezeichnet. Der „fehlerhafte“ Anstieg wurde für den Toleranzbereich von 10° korrekt geteilt, aber für den Toleranzbereich von 20° nicht mehr. Daher wurde statt zwei Anstiegen nur mehr ein gemeinsamer langer Anstieg detektiert, der dem dritten Bereich zugeordnet wurde. Die Anzahl der Steigungen im zweiten Bereich erreichte bei einer gewissen Größe - 20° - ein Maximum und fiel dann wieder ab. Gleichzeitig fielen mit zunehmendem Toleranzbereich mehr Steigungen in den dritten Bereich. Steigungen, die beim Toleranzbereich von 10° als Fehler detektiert wurden, wurden auch bei größeren Toleranzbereichen dem dritten Bereich zugewiesen.

Grundsätzlich galt es, bei der Wahl des Toleranzbereiches eher den Parameter mit den meisten Steigungen im zweiten Bereich zu wählen, als den Parameter, der die wenigsten Steigungen im dritten Bereich hatte. Der Grund war die Tatsache, dass aus Steigungen im zweiten Bereich am ehesten Lifte bestimmt werden konnten.
Für den Datensatz Saalbach wurden 20° als Toleranzbereich gewählt.

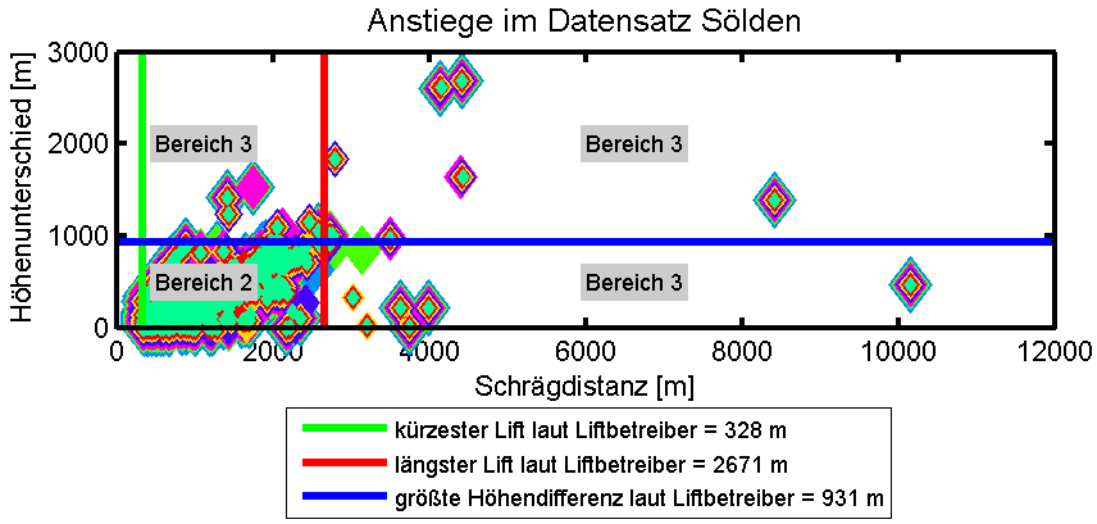


Abbildung 4.6.: Verteilung der Anstiege in Bereiche, Quelle: eigene Darstellung

Für den Datensatz Sölden wurden die gleichen Berechnungen durchgeführt als für den Datensatz Saalbach. Da der Datensatz viel größer war, waren auch die Anzahl der Steigungen pro Bereich für die unterschiedlichen Toleranzbereiche größer. Die kürzesten und längsten Lifte, sowie die maximale Höhendifferenz wurden angepasst (siehe Abb. 4.6). Die numerischen Werte wurden in der Tabelle 4.2 aufgelistet. Als Toleranzbereich für den Datensatz Sölden wurden 25° gewählt.

Toleranz [°]	Bereich 1	Bereich 2	Bereich 3
10	93690	1130	13
20	74073	1176	17
25	68545	1183	19
30	64317	1181	21
35	60851	1179	23
40	57861	1169	24
50	53124	1173	26
60	49080	1166	27

70	45590	1158	29

Tabelle 4.2.: Summe der Steigungen pro Bereich - Sölden

4.6. Positive Steigungen eliminieren

In diesem Schritt wurden für beide Datensätze kurze Steigungen entfernt. Für das Schigebiet Saalbach wurde der *Oberschwarzachlift* mit 217 Meter Länge als kürzester Lift ausgewählt. Von diesem Wert wurden 50 Meter abgezogen und diese Distanz als kürzeste Länge verwendet. Alle Anstiege, die kürzer waren, wurden entfernt. Es wurde keine Distanzen auf Grund der Höhendifferenzen entfernt, da nach diesen nicht gefiltert wurde, weil es Lifte gab, die sehr lang waren und trotzdem nur eine kleine Höhendifferenz überwanden. Für den Datensatz Saalbach ergaben sich für die weiteren Schritte 385 Steigungen mit 12328 Datenpunkten.
Für das Schigebiet Sölden wurde der Lift *Mini Tiefenbach* mit 328 Meter Länge als kürzester Lift ausgewählt. Von der Liftlänge wurden 50 Meter abgezogen und dieser Wert als kürzeste Distanz für den Schritt eingeführt. Für den Datensatz Saalbach ergaben sich 1249 Steigungen mit insgesamt 105580 Datenpunkten für die Berechnung von Liften.

4.7. Daten ordnen

In diesem Schritt wurden als erstes die Distanzen von jedem Punkt zu jedem Punkt im Datensatz bestimmt. Anhand des ϵ-Wertes wurden jedem Punkt seine Nachbarpunkte zugewiesen. Im Sinne des OPTICS-Algorithmus lieferte der Wert ∞ als ϵ-Parameter die korrektesten Ergebnisse. Der Wert $\epsilon = \infty$ verbrauchte bei der Berechnung die meiste Berechnungszeit und benötigte den größten Arbeitsspeicherbedarf. Für den Datensatz Saalbach wurde als Eingabeparameter für ϵ der Wert ∞ gewählt. Auch für den Datensatz Sölden mit mehr als 100000 Punkten war die Berechnung möglich, es trat aber ein anderes Problem in der Praxis auf. Wenn der ϵ-Parameter auf ∞ gesetzt wurde, dann gehörten alle Punkte der selben Teilmenge an. Im Schritt 4.8 wurden aus den Teilmengen rekursiv so lange kleinere Teilmengen bestimmt, bis diese eine kleinste erlaubte Punkteanzahl erreichten. Aus technischen Gründen musste die maximale Anzahl von rekursiven Aufrufen beschränkt werden. Falls das Limit erreicht wurde, wurde an den Benützer eine Warnung herausgegeben, dass die Berechnung der Teilmengen nicht korrekt abgeschlossen werden konnte. Beim Datensatz Sölden mit mehr

als 100000 Punkten wurde mit dem Parameter $\epsilon = \infty$ das Limit erreicht, wobei für den Wert ∞ die größte Distanz zwischen zwei Punkten im Datensatz gewählt wurde. Darum wurde für den ϵ-Parameter der Wert 30 Meter gewählt. Dadurch wurden zwar die Punkte in mehrere große Teilmengen aufgeteilt, bei der Berechnung der kleinsten Teilmengen kam es aber zu keinen Fehlermeldungen.
Die kleinste Anzahl von Nachbarpunkten *MinPts*, für die Identifikation von Kernpunkten, wurde für beide Datensätze auf ein Minimum von 1 gesetzt. Ein niedriger Wert generierte größere Teilmengen, da mehr Kernpunkte detektiert wurden und noch nicht besuchte Punkte konnten im OPTICS-Algorithmus nur über diese einer Teilmenge zugewiesen werden. Die Ergebnisse des OPTICS-Algorithmus wurden in ein Textfile geschrieben.

4.8. Teilmengen berechnen und Cluster bestimmen

Der Algorithmus, der im Unterabschnitt 2.6.1 beschrieben wurde, teilte die Datenmengen so lange, bis die Teilmengen einen minimalen Wert unterschritten oder es keine weiteren lokalen Maxima in einer Datenmenge gab. Die minimale Punkteanzahl für Teilmengen wurde mit 0.5% der Gesamtpunkteanzahl eines Datensatzes angenommen (siehe Begründung S. 46).
Der Datensatz Saalbach hatte 12328 Datenpunkte. Als kleinster Wert für eine Teilmenge ergaben sich somit 61 Punkte. Der Datensatz Sölden bestand schon von vornherein aus mehreren Teilmengen und für jede Teilmenge musste die minimale Anzahl separat berechnet werden. Die größte Teilmenge bestand aus 24014 Punkten und als kleinster Wert für diese Teilmenge wurden 120 Punkte bestimmt.
Nach der Bestimmung der kleinsten Teilmengen, wurde jede Teilmenge auf die Bedingung eines Clusters untersucht. Ein Cluster musste aus Daten von unterschiedlichen Anstiegen bestehen. Der Wert konnte vom Benützer der Software frei bestimmt werden. In den nachfolgenden Schritten wurden aus den Clustern Lifte berechnet. Daher war es bereits bei diesem Schritt notwendig eine minimale Anzahl von Anstiegen zu bestimmen, die als Bedingung für einen Cluster galt. Aus den Clustern wurden im Anschluss an diesen Schritt Lifte berechnet. Der minimale Wert für den Datensatz Saalbach wurde mit drei angenommen. Der Datensatz Saalbach war klein und daher konnte der Wert nicht höher angesetzt werden, da sonst zu viele Cluster entfernt worden wären. Der Wert konnte auch nicht kleiner gesetzt werden, da sonst bereits einzelne Personen einen Lift begründet hätten. Der Datensatz Sölden war um einiges

größer. Daher konnte von mehreren Pfaden pro Lift ausgegangen werden. Die wichtigen Zubringer und neueren Gondelbahnen hatten teilweise mehr als 20 unterschiedliche Anstiege pro Cluster. Bei den weniger stark befahrenen Liften waren trotzdem nur einige wenige Pfade gegeben. Daher wurde für den Datensatz Sölden die kleinste Anzahl mit fünf festgelegt.
Die Werte, die in diesem Schritt verwendet wurden, wurden auch jedes Mal in den folgenden Schritten verwendet, wenn es darum ging, einen Wert festzulegen.
Wie im Schritt 3.8 beschrieben wurde, konnte zwischen der strengen und weniger strengen Lösung entschieden werden. Für beide Datensätze wurde die weniger strenge Bestimmung ausgewählt. In Tabelle 4.3 wurden die Unterschiede in der Clusteranzahl pro Datensatz zwischen strenger und weniger strengen Berechnung aufgelistet.

Schigebiet	Saalbach	Sölden
Ausgangsteilmengen	1	236
Cluster - strenger Ansatz	73	781
Cluster - weniger strenge Ansatz	82	824

Tabelle 4.3.: Unterschied zwischen strenger und weniger strengen Berechnung

In der Tabelle 4.3 wurde für den Datensatz Saalbach nur eine Ausgangsteilmenge aufgelistet. Diese Teilmenge beinhaltete alle Datenpunkte, da der ϵ-Parameter im Schritt 4.7 auf ∞ gesetzt wurde. Anders war dies beim Datensatz Sölden, wo der ϵ-Parameter mit 30 Meter angenommen wurde und daher die Anzahl der Ausgangsteilmengen höher war.
In Abbildung 4.7 wurde die Übersicht der Cluster im Datensatz Saalbach dargestellt. Hervorgehoben wurde der *Schattberg X-press I+II*. Obwohl die Punkte im Erreichbarkeitsdiagramm als Teilmenge detektiert wurden (siehe Abb. 4.8), entsprach die Teilmenge nicht der Bedingung eines Clusters, da sie aus nur 2 unterschiedlichen Pfaden bestand. Der Lift war zwar im Datensatz vorhanden, konnte aber aus diesem Grund nicht detektiert werden. Um diesen Lift zu detektieren, mussten die Eingabeparameter anders gewählt werden. Somit wurden aber auch Lifte gefunden, die gar keine waren. Daher wurde der Wert drei als Bedingung verwendet.

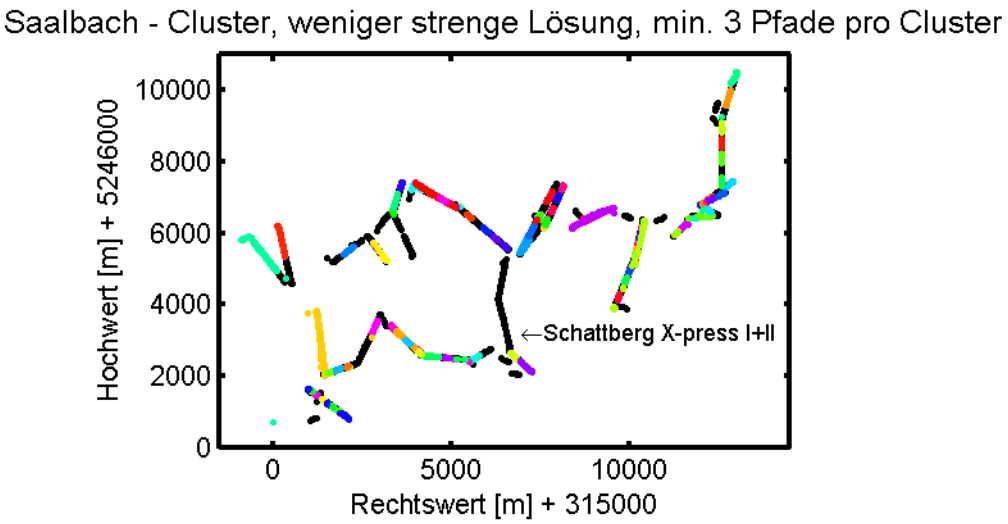


Abbildung 4.7.: Verteilung der Cluster im Schigebiet, Quelle: eigene Darstellung

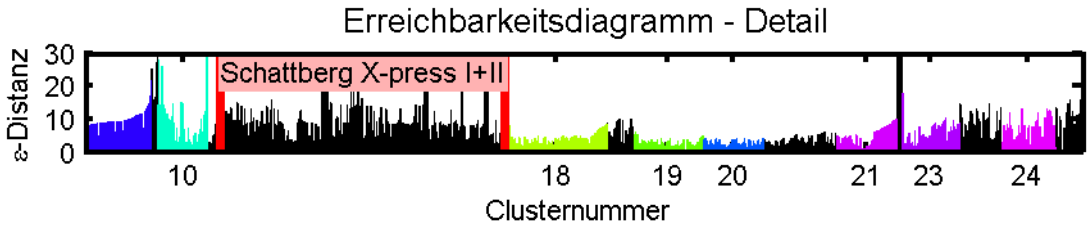


Abbildung 4.8.: Detail im Erreichbarkeitsdiagramm, Quelle: eigene Darstellung

In Abbildung 4.9 wurden die Cluster im Schigebiet Sölden dargestellt. Im Schigebiet Sölden wurde für jeden Lift, der in den Daten vorhanden war, ein Cluster gefunden.

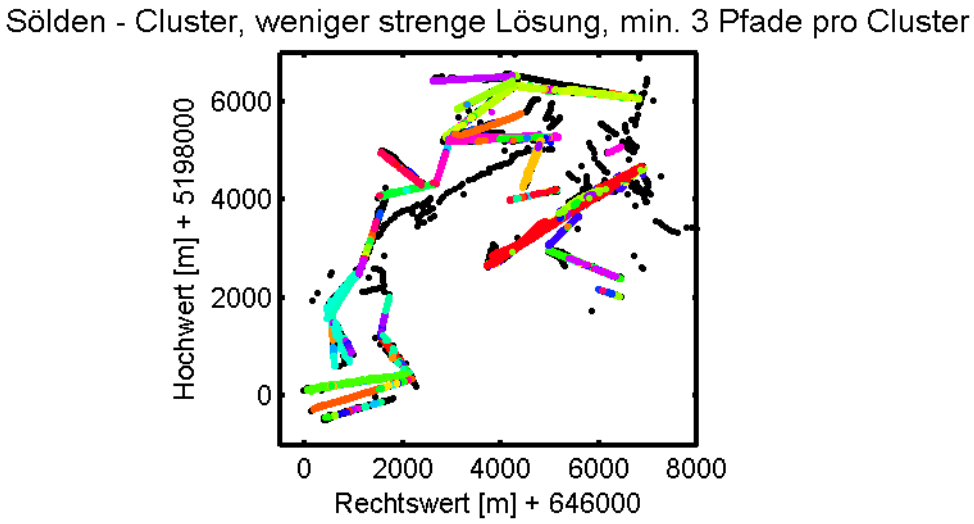


Abbildung 4.9.: Verteilung der Cluster im Schigebiet, Quelle: eigene Darstellung

4.9. Liftkandidaten extrahieren

Die Berechnung von Liften verlief wie im Abschnitt 2.7 beschrieben. Im ersten Teilschritt einer Iteration wurden die Cluster, die aus Daten von gleichen Pfade generiert wurden, zusammengefügt. Als kleinster Wert von gleichen Pfaden wurde der gleiche Wert, wie er im Schritt 4.8 zur Bestimmung von Clustern verwendet wurde, angenommen. Im zweiten Teilschritt wurden pro Cluster die verwendeten Pfade untereinander verglichen. Pfade, die eine zu große Abweichung zu den anderen Pfaden im Cluster aufwiesen, wurden eliminiert. Falls einem Cluster dadurch weniger Pfade übrig blieben, als es die Bedingung für Cluster vorgab, wurde der Cluster gelöscht. Da Cluster oft aus mehreren unterschiedlichen Pfaden bestanden als es die Bedingung für Cluster vorgab, wurden jeweils die aussortierten Pfade eines Cluster untereinander verglichen. Falls sich unter diesen Pfaden zusammenpassende Pfade befanden und deren Anzahl die Bedingung für einen selbstständigen Cluster erfüllten, definierten diese Pfade einen selbstständigen neuen Cluster. Die maximale Abweichung, die sich ein Pfad von anderen Pfaden im Cluster unterscheiden durfte, wurde im ersten Iterationsschritt mit einem so großen Wert angenommen, dass im ersten Iterationsschritt keine Pfade aus den Clustern eliminiert wurden. Mit diesem Wert wurden in der ersten Iteration keine Pfade aus irgendeinem Cluster eliminiert. Nach jedem Iterationsschritt wurde dieser Wert neu berechnet. Er ergab sich als Mittelwert aller Abweichungen eines jeden Pfades in jedem Cluster. Damit wurde der Wert für die maximale Abweichung pro Iteration immer kleiner, da die Pfade in den Clustern immer besser zueinander passten und weil die zulässige Abweichung pro Iterationsschritt immer kleiner wurde.
Die Iteration wurde so lange wiederholt, bis alle Cluster gelöscht waren, weil sie die Bedingung für Cluster nicht mehr erfüllten, oder der neue Mittelwert für die Abweichungen einen maximalen erlaubten Wert unterschritt. Der kritische Wert konnte vom Benützer der Software vorgegeben werden und wurde anhand der nachfolgend beschriebenen Überlegung berechnet. Ein Lift wurde aus mehreren Pfaden gemittelt. Die Anfangs- und Endpunkte eines Liftes wurden aus den Anfangs- bzw. Endpunkten der Pfade eines Clusters gemittelt. Die GPS-Daten wurden ohne Genauigkeitsangaben geliefert, daher wurden für die Genauigkeit s_{UTM} 10 Meter angenommen. Der mittlere Wert ergab sich aus der Formel

$$x_{UTM,Anfang} = \frac{1}{n} \sum_{i=1}^{n} x_{i,Anfang} \tag{4.1}$$

und die Genauigkeit wurde mit der Formel

$$s_{x_{UTM,Anfang}} = \frac{1}{\sqrt{n}} s_{UTM} \tag{4.2}$$

berechnet. Für beide Koordinaten wurden die gleichen Genauigkeiten angenommen

$$s_{x_{UTM}} = s_{y_{UTM}} = s_{x_{UTM,Anfang}} = s_{y_{UTM,Anfang}} = s_{x_{UTM,Ende}} = s_{y_{UTM,Ende}} \tag{4.3}$$

und somit ergab sich der Punktlagefehler

$$s_{P_{UTM}} = \sqrt{s_{y_{UTM}}^2 + s_{x_{UTM}}^2} \tag{4.4}$$

Für die Variable n wurde jeweils der gleiche Wert wie für die Übereinstimmung im Teilschritt eins gewählt. Für den Datensatz Saalbach wurde n mit drei angenommen. Daraus ergab sich ein Punktlagefehler von 8 Metern. Weil einige Lifteintritts- und Austrittspunkte in großen Stationsgebäuden waren und in solchen Häusern keine guten GPS-Messungen erfolgen konnten, wurde der Wert für die Genauigkeit mit dem Faktor fünf multipliziert. Der Faktor ergab sich aus der Überlegung, dass der Punktlagefehler 8 Meter betrug und das fünffache von diesem Wert durchschnittlich einem großen Stationsgebäude entsprach. Zusätzlich mussten an beiden Liftenden Punkte gemittelt werden. Daher wurde der berechnete Wert mit zwei multipliziert. Als kritischer Wert für den Datensatz Saalbach ergaben sich 81 Meter. Für den Datensatz Sölden wurde für die Variable n der Wert fünf verwendet. Daher ergab sich der kritische Wert von 63 Metern für den Datensatz.
Der zweite Teilschritt war ein Grund, warum im Schritt 4.6 kleine Anstiege eliminiert werden sollten, da in diesem Teilschritt aus allen Anstiegen Lifte detektiert wurden, die im nächsten Schritt manuell aus den Ergebnissen entfernt werden mussten.
Im Datensatz Saalbach wurden 29 potentielle Lifte gefunden. Potentiell deshalb, weil z.B. nicht alle gefunden Lifte auch Lifte waren, da der Datensatz noch alte Lifte beinhaltete, die es nicht mehr gab und die gelöscht werden mussten. Im Datensatz Sölden wurden 25 potentielle Lifte gefunden.

4.10. Lifte löschen

Von den 29 potentiellen Liftkandidaten im Datensatz Saalbach wurden zwei potentielle Lifte entfernt. Aus dem Datensatz Saalbach wurden 27 Lifte detektiert (siehe Abb. 4.10).

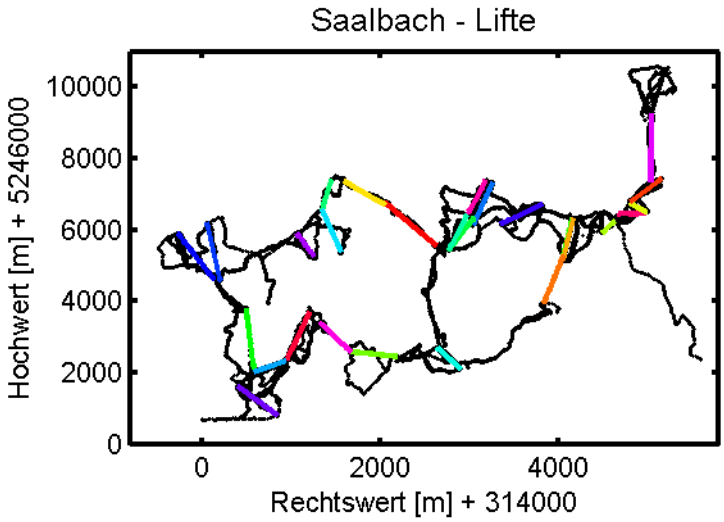


Abbildung 4.10.: detektierte Lifte in Saalbach, Quelle: eigene Darstellung

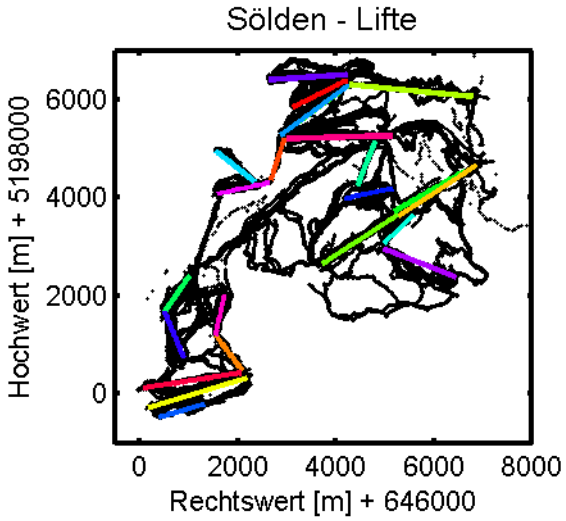


Abbildung 4.11.: detektierte Lifte in Sölden, Quelle: eigene Darstellung

Von den 25 potentiellen Liften im Datensatz Sölden wurde 1 potentieller Lift entfernt. Falls im Schritt 4.9 die Abbruchbedingung zu hoch gewählt wurde oder im Schritt 4.6 keine Steigungen eliminiert wurden, gab es in diesem Schritt pro Lift mehrere potentielle Liftkandidaten. Im zweiten Teilschritt des Schrittes 4.9 wurde pro Cluster nach allen möglichen Kombinationen von Pfaden gesucht, die zusammen einen Cluster definierten. Aus jedem gefundenem Cluster wurde ein Lift berechnet. Dadurch ergaben sich pro Lift mehrere Liftkandidaten, die sich aber alle vom korrekten Lift unterschieden, da sie aus einer kleineren Anzahl von unterschiedlichen Pfaden berechnet wurden, kürzer waren

oder an unterschiedlichen Punkten starteten oder endeten (siehe Abb. 3.8). Für den Datensatz Sölden wurden 24 Lifte detektiert.

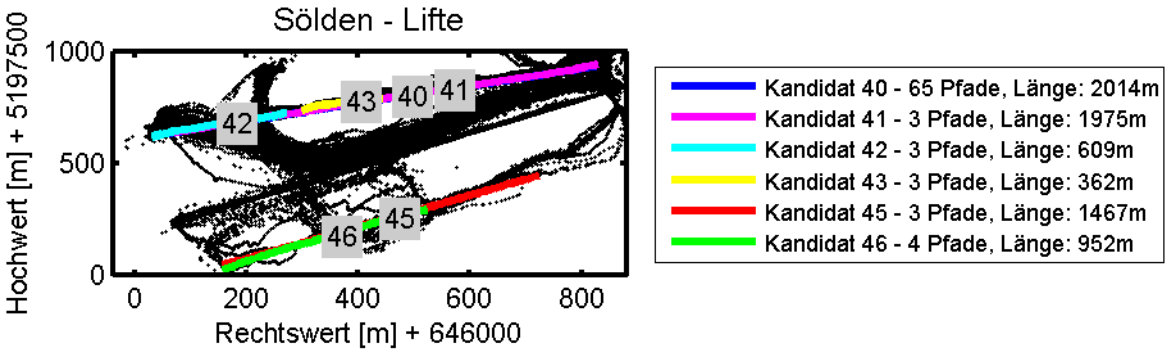


Abbildung 4.12.: potentielle Liftkandidaten, Quelle: eigene Darstellung

In Abbildung 4.12 wurden einige Liftkandidaten im Detail dargestellt. Beide Lifte hatten mehrere potentielle Liftkandidaten. Der „korrekte“ Liftkandidat für den oberen Lift konnte anhand der Anzahl von Pfaden pro Liftkandidat und den Längen der restlichen Anwärter bestimmt werden. Der Kandidat 40 hatte die größte Anzahl an Pfaden und wurde als einziger der vier Liftkandidaten nicht gelöscht. Beim zweiten, unteren Lift war die beste Lösung nicht eindeutig. Der Lift war ein Schlepplift und auf einem Gletscher gebaut. Der Eintrittspunkt konnte nicht eindeutig festgelegt werden. Laut den Angaben auf der Homepage des Seilbahnbetreibers wäre der längere Liftkandidat der „korrektere“, allerdings könnte auch der kürzere Liftkandidat der selbe Lift mit einem veränderten Startpunkt sein, z.B. im Sommerbetrieb. Ein Indiz dafür waren der unterschiedliche Startpunkt aber ein gemeinsamer Endpunkt. Im umgekehrten Fall könnte es sich um einen Schlepplift handeln, an welchem es zu vielen Stürzen kam.
Mit diesem Schritt wurde die Berechnung der Lifte abgeschlossen. Eine detaillierte Auflistung der detektierten Lifte und der entsprechenden Details wurde im Anhang A angehängt.

4.11. Daten für Pisten vorbereiten

In diesem Schritt wurden Abfahrten aus den Trajektorien gefiltert. Zunächst wurden alle Verbindungen zwischen zwei Anstiegen, die für die Berechnung eines Liftes verwendet

wurden, als Abfahrten verspeichert. Diese Abfahrten wurden ein weiteres Mal separat nach Anstiegen gefiltert. Abfahrten, die Steigungen beinhalteten, die größer als ein Minimalwert waren, wurden gespalten und die Anstiege entfernt. Als Näherung für den Minimalwert wurde der gleiche Wert wie in Schritt 4.6 verwendet. Für den Datensatz Saalbach wurden 576 Abfahrten mit insgesamt 41062 Datenpunkten vorbereitet. Der Datensatz Sölden hatte 2235 Abfahrten mit insgesamt 217848 Datenpunkten.

4.12. Abfahrten den Liften zuordnen

Um jeden Liftpunkt wurde ein Kreis mit 50 Meter Radius gelegt. Für jeden Datenpunkt einer Abfahrt wurde untersucht, ob sich dieser innerhalb des Radius um einen Liftpunkt befand. Wurde ein Punkt der Abfahrt innerhalb eines Kreises gefunden, wurde die Abfahrt dem zum Kreis gehörenden Lift zugewiesen. Eine Abfahrt konnte mehreren Liften zugewiesen werden.
Am häufigsten begannen und endeten Abfahrten bei einem Liftpunkt. Dies entsprach dem klassischen Verhalten eines Schifahrers, der von Lift zu Lift fuhr. Mit einem Liftpunkt war der Ein- oder Austrittspunkt eines Liftes gemeint. Abfahrten, die nur bei einem Liftpunkt starteten, waren z.B. Talabfahrten. Abfahrten, die nur bei einem Lift endeten, waren z.B. Anfahrten zum ersten Lift.
Im Datensatz Saalbach gab es 576 Abfahrten. Die Abfahrten verteilten sich auf

- 402 Abfahrten, die bei einem Liftpunkt starteten und bei einem Liftpunkt endeten
- 60 Abfahrten, die bei einem Liftpunkt starteten
- 49 Abfahrten, die bei einem Liftpunkt endeten
- 65 Abfahrten, die weder bei einem Liftpunkt starteten, noch bei einem Liftpunkt endeten

Die 65 Abfahrten ohne Start- und Endpunkt bei einem Lift waren größtenteils Abfahrten, die zu Liften führten, die im Datensatz nicht detektiert wurden, wie z.B. die Pisten, die beim Lift *Schattberg X-press* endeten, der nicht detektiert wurde (siehe Abb. 4.8).
Im Datensatz Sölden gab es 2235 Abfahrten. Die Abfahrten verteilten sich auf

- 1714 Abfahrten, die bei einem Liftpunkt starteten und bei einem Liftpunkt endeten
- 179 Abfahrten, die bei einem Liftpunkt starteten

- 155 Abfahrten, die bei einem Liftpunkt endeten
- 187 Abfahrten, die weder bei einem Liftpunkt starteten, noch bei einem Liftpunkt endeten

Die Anzahl von Abfahrten, die weder bei einem Liftpunkt starteten noch bei einem Liftpunkt endeten, war auch deswegen so hoch, weil sich 34 Abfahrten im Schigebiet Obergurgl befanden und die Lifte in diesem Schigebiet gelöscht wurden. Somit hingen diese Abfahrten „in der Luft“. Der Grund für die Abfahrten im Schigebiet Obergurgl war, dass eine Trajektorie Daten in Sölden als auch in Obergurgl hatte.

4.13. Pistenkandidaten bestimmen

In diesem Schritt wurden Pistenkandidaten bestimmt. Um einen Pistenkandidaten zu detektieren, musste eine minimale Anzahl von unterschiedlichen Abfahrten zwischen zwei Liftpunkten verlaufen. Die Erklärung wurde im Anschnitt 3.13 erläutert. Als minimale Anzahl wurde für beide Datensätze der Wert drei gewählt. Ob die Pistenkandidaten zwischen zwei Liftpunkten eine Piste definierten oder nicht, war nur mit der Detektion von Pistenkandidaten noch nicht beantwortet, da es zwischen zwei Liftpunkten auch mehrere Varianten von Pisten geben konnte.
Für den Datensatz Saalbach wurden

- 52 Pistenkandidaten, die bei einem Liftpunkt starteten und bei einem Liftpunkt endeten
- 2 Pistenkandidaten, die bei einem Liftpunkt starteten
- 0 Pistenkandidaten, die bei einem Liftpunkt endeten
- 2 Pistenkandidaten, die weder bei einem Liftpunkt starteten, noch bei einem Liftpunkt endeten

gefunden.
Für den Datensatz Sölden wurden

- 116 Pistenkandidaten, die bei einem Liftpunkt starteten und bei einem Liftpunkt endeten
- 20 Pistenkandidaten, die bei einem Liftpunkt starteten

- 16 Pistenkandidaten, die bei einem Liftpunkt endeten
- 5 Pistenkandidaten, die weder bei einem Liftpunkt starteten, noch bei einem Liftpunkt endeten

gefunden.

4.14. Pisten aus Pistenkandidaten berechnen

Die Berechnung der Pisten erfolgte wie im Abschnitt 2.8 beschrieben. Über beide Schigebiete wurde ein Raster gelegt. Pro Pistenkandidat wurde jeder Punkt einer Abfahrt des Pistenkandidaten einer Rasterzelle zugewiesen. Da die Pistenkandidaten den Liften bereits zugewiesen waren, waren die Ausgangs- und Endrasterzellen bereits bekannt. Der Algorithmus suchte Wege entlang von Rasterzellen, die zumindest von einer minimalen Anzahl von unterschiedlichen Pfaden besucht wurden. Als minimale Anzahl wurde der gleiche Wert wie im Schritt 4.13 angenommen. Für jede Rasterzelle einer Piste wurde aus den Punkten der Abfahrten, die in den Rasterzelle gegeben waren, ein mittlerer Wert berechnet. Der gemittelte Wert pro Rasterzelle ergab den Verlauf einer Piste. Einige Pisten verliefen bis auf wenige Rasterzellen gleich wie andere Pisten. Pro Liftkandidat konnten mehrere Pisten gefunden werden. Falls zwei Pisten entlang der selben Rasterzelle verliefen, wurde der Wert für solche Rasterzellen aus allen Punkten der Pisten, die in der Rasterzelle vorkamen, gemittelt. Damit erhielten beide Pisten für solche Rasterzellen den gleichen Wert zugewiesen.

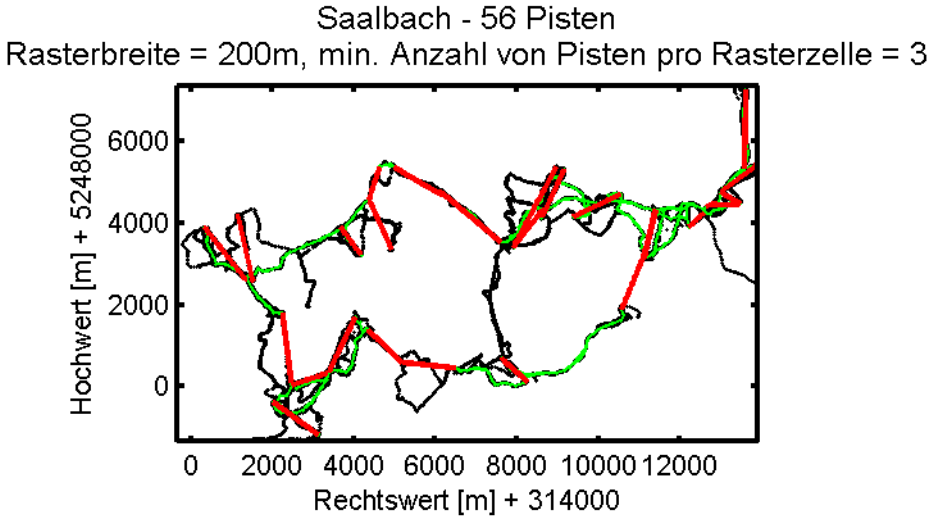


Abbildung 4.13.: detektierte Pisten und Lifte in Saalbach, Quelle: eigene Darstellung

Für den Datensatz Saalbach wurde eine Rasterbreite von 200 Metern gewählt. Aus den 52 Pistenkandidaten wurden 56 Pisten detektiert. Bei einigen Liften wurden aus einem Pistenkandidaten mehrere Pisten bestimmt. Im Vergleich mit den Pisten, die den Homepages der Seilbahnbetreiber entnommen wurden, wurden 27 von den 63 angeführten Pisten gefunden. In Abbildung 4.13 sind alle gefunden Pisten eingezeichnet. Eine detaillierte Auflistung der gefundenen Pisten und den dazugehörenden Pisten im Schigebiet wurde im Anhang B angefügt.
Für den Datensatz Sölden wurden aus den 122 Pistenkandidaten, die den Start- und Zielpunkt bei einem Lift hatten, 154 Pisten gefunden. Als Rasterbreite wurden 100 Meter gewählt. Laut Liftbetreiber gab es in der Wintersaison 2014/15 36 Pisten. Von den 36 Pisten wurden 31 Pisten detektiert. Eine detaillierte Auflistung der gefundenen Pisten wurde dem Anhang B angefügt.

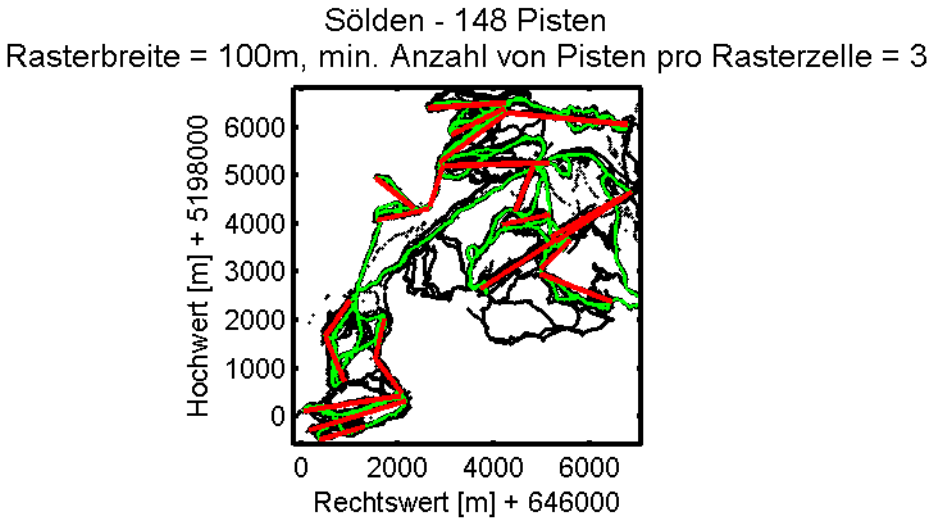


Abbildung 4.14.: detektierte Pisten und Lifte in Sölden, Quelle: eigene Darstellung

In Abbildung 4.14 wurden die gefunden Pisten und Lifte des Datensatzes Sölden dargestellt. Die Berechnung der Lifte wurde mit diesem Schritt abgeschlossen.

4.15. Rasterverbindungen bestimmen

Die Rasterverbindungen wurden wie im Unterabschnitt 2.8 erklärt berechnet.

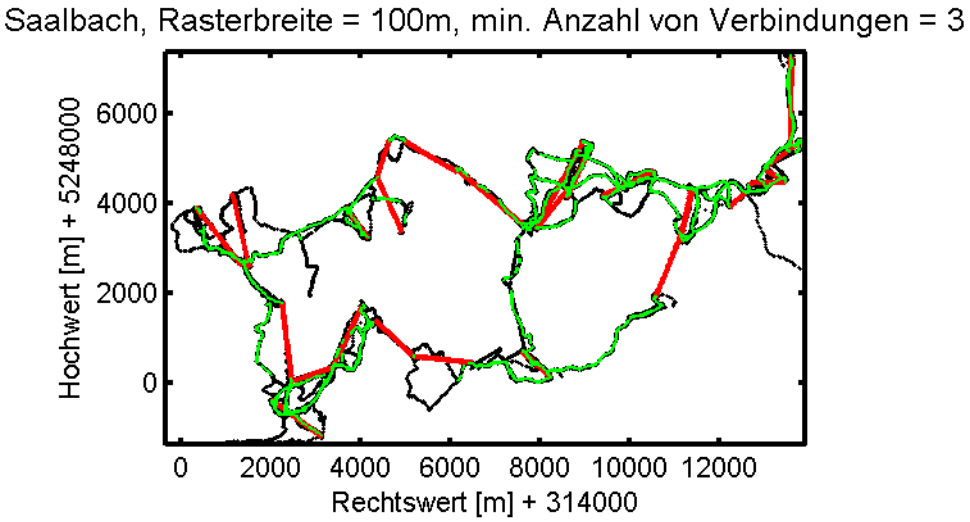


Abbildung 4.15.: Rasterverbindungen im Datensatz Saalbach, Quelle: eigene Darstellung

In diesem Schritt wurde zunächst ein Raster über das Schigebiet gelegt. Als nächstes wurden die Datenpunkte aller Abfahrten den Rasterzellen zugewiesen. Für jede Rasterzelle wurde untersucht, ob Punkte einer Abfahrt in benachbarten Zellen auftauchten. Falls zwischen zwei Rasterzellen eine minimale Anzahl von Abfahrten verlief, wurde dies als Verbindung detektiert.
In Abbildung 4.15 wurde das Ergebnis dieses Schrittes für das Schigebiet Saalbach dargestellt. Da die Datenmenge größer als im Schritt *Pisten berechnen* war, wurde die Rasterbreite auf 100 Meter halbiert. Mit einer kleineren Rasterbreite konnte die Breite der Pisten genauer abstrahiert werden. Auf Grund der spärlichen Verteilung der Daten konnte die Rasterbreite nicht weiter gesenkt werden, da sonst zu wenige Rasterzellen die erforderliche minimale Anzahl von Verbindungen zwischen zwei Rasterzellen erfüllt hätten. Als Minimalwert wurde der gleiche Wert - drei - verwendet, wie für die Berechnung der Lifte in Schritt 4.13.
In Abbildung 4.16 wurde das Ergebnis für das Schigebiet Sölden dargestellt. Als Rasterbreite wurden 50 Meter gewählt. Als minimale Anzahl von Pfaden, die zwischen zwei Rasterzellen verlaufen mussten, um eine Verbindung zu detektieren, wurde der gleiche Wert verwendet, wie für die Bestimmung von Liften - nämlich fünf.

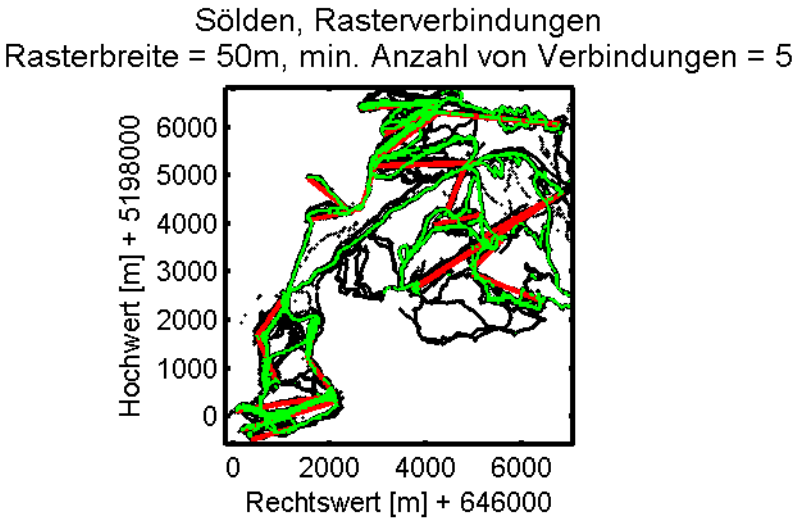


Abbildung 4.16.: Rasterverbindungen im Datensatz Sölden, Quelle: eigene Darstellung

Auf Grund der geringeren Rasterbreite wurden breite Pisten besser detektiert. Die Piste zwischen der *Mutkogelbahn* und der *Tiefenbachbahn* war z.B. breiter als eine normale Piste, daher gab es dort auch eine Vielzahl von Verbindungen (siehe Abb. 4.17).

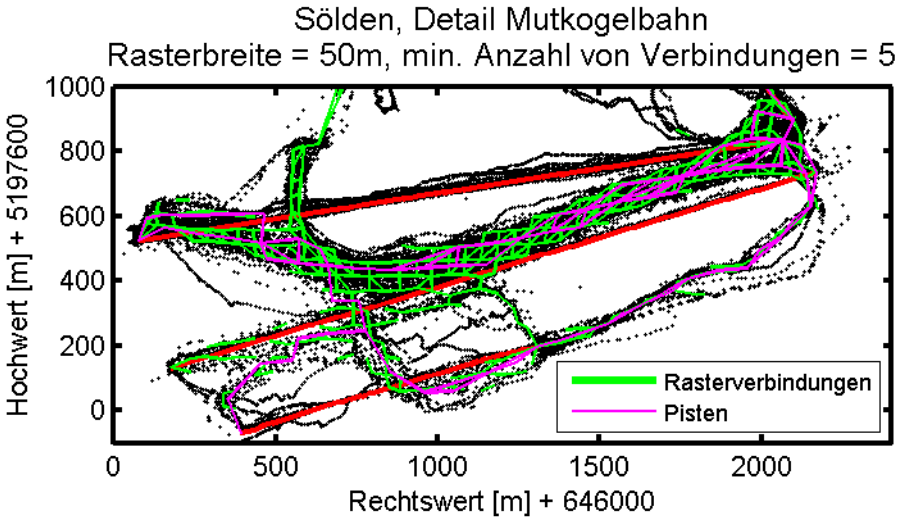


Abbildung 4.17.: Detailausschnitt aus dem Datensatz Sölden, Quelle: eigene Darstellung

In Abbildung 4.17 wurden die Verbindungen zwischen Rasterzellen dargestellt, ebenso die Pisten, die im Schritt 4.14 berechnet wurden. Es wurden deutlich mehr Verbindungen gefunden als Pisten.

4.16. Zusammenfassung und Interpretation der Ergebnisse

Der Algorithmus generierte für beide Datensätze gute Ergebnisse. Die Wahl der Parameter war beim Großteil der Schritte nicht schwer auszuwählen. Entscheidend waren die Parameter *Toleranzbereich* im Schritt 4.4 und ϵ im Schritt 4.7. Für den Parameter *Toleranzbereich* galt, dass dieser nicht zu klein gesetzt werden durfte, da dieser sonst die Detektion von Liften verhindert hätte. Für den ϵ-Parameter galt, dass dieser ab einer gewissen Datenanzahl nicht mehr auf einen Maximalwert gesetzt werden konnte, da es sonst Schwierigkeiten mit dem Rekursionslimit im Schritt 4.8 gab. Je dichter belegt ein Datensatz war, desto kleiner konnte der ϵ-Parameter gesetzt werden. Bei beiden Datensätzen wurden vor dem Schritt 4.7 kurze Anstiege entfernt.
Die Höhentrajektorien der berechneten Lifte konnten nicht gut bestimmt werden, da die Ausgangsparameter nicht ausreichend genau gegeben waren. Darum wurden zunächst sämtliche Schritte in der Ebene berechnet und nur für die Ergebnisse die Höhenkoordinaten bestimmt. Vor allem die Bestimmung der Höhe von Luftseilbahnen war schwierig.
Bei der Berechnung von Pisten ging es einerseits um die Detektion von Verbindungen zwischen zwei Liftpunkten und andererseits um die befahrbaren Abschnitte eines Schigebietes. Letzteres wurde mit dem Schritt 4.15 umgesetzt. Die Vollständigkeit des Schigebietes hing von der Anzahl und Dichte der Trajektorien, die für den Algorithmus verwendet wurden, ab.
Der Vergleich und die Kontrolle der Ergebnisse konnte nur teilweise automatisiert werden, da nur *OpenStreetMap*(OSM) den Download der Koordinaten von Liften anbot. Die berechneten Lifte wurden mit den Liften aus OSM verglichen. Die Daten von OSM waren als UTM-Koordinaten gegeben und daher konnten die Liftlängen nur in der Ebene verglichen werden. Weiters wurden die Liftlängen mit den Liften, die die Seilbahnbetreiber auf ihren Homepages zur Verfügung stellten, verglichen. Das Problem war, dass die meisten Seilbahnbetreiber auf ihren Homepages nicht angaben, wie die Längen von Liften berechnet wurden. Denkbar wären zwei Varianten. Es könnten die Schrägdistanzen zwischen dem Anfangspunkt und dem Endpunkt eines Liftes berechnet worden sein oder die Summe von Teillängen zwischen zwei Liftpunkten. Daher war ein Vergleich schwierig. Die Kontrolle wurde auch visuell vorgenommen, indem die berechneten Pisten mit einem Schigebietsplan verglichen wurden. Mit diesem Verfahren konnte allerdings nur kontrolliert werden, ob Lifte fehlten bzw. zu viele berechnet wurden.

In der Wintersaison 2014/15 hatte das Schigebiet Saalbach 56 Lifte in Betrieb. Von den 56 Liften im Schigebiet waren 7 Babylifte und 7 kürzere Schlepplifte ohne Verbindung zu den größeren Seilbahnen. Von diesen 14 Liften war kein einziger im Datensatz gegeben. Ebenfalls nicht im Datensatz war die *Steinbergbahn I+II*, da diese erst in der Wintersaison 2014/15 eröffnet wurde. Einige Lifte waren von zu wenigen Schifahrern befahren worden und konnte deswegen nicht detektiert werden, wie z.B.: *Schattberg X-press I+II*. Aus dem Datensatz Saalbach wurden 27 Lifte extrahiert.
Für den Vergleich der Lifte, wurden diese von der Internetseite *OpenStreetmap* heruntergeladen und mit den berechneten Pisten verglichen. Alle Eintrittspunkte von Liften des Datensatzes Saalbach befanden sich innerhalb von 40 Metern zu den Angaben, die OSM anbot.
Der Datensatz Saalbach konnte auf Grund der geringen Datendichte nicht vollständiger detektiert werden. Die Lifte, die gefunden wurden, passten mit den Liften in OSM überein.

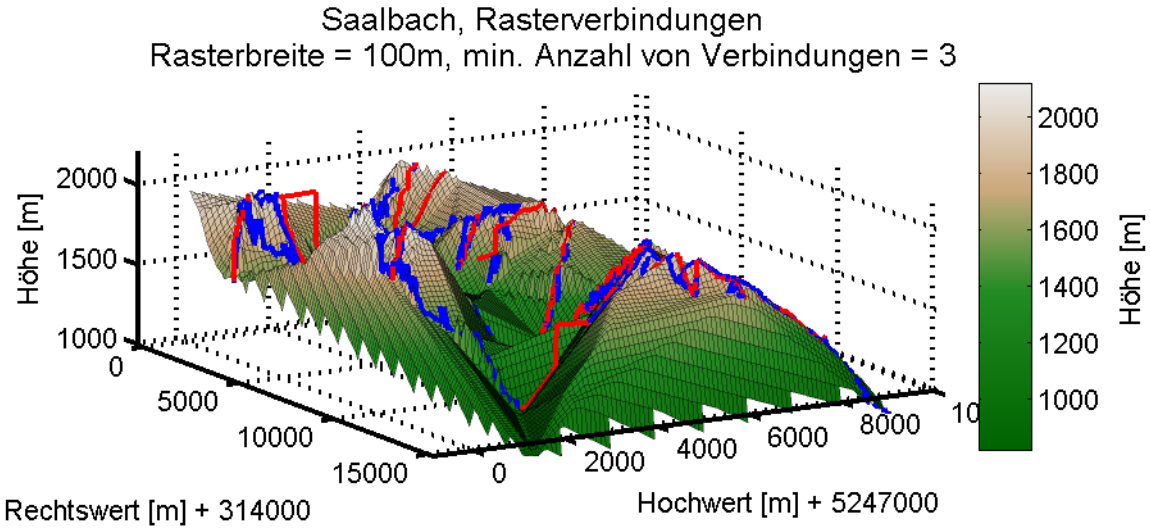


Abbildung 4.18.: 3D-Modell-Übersicht für Saalbach, Quelle: eigene Darstellung

In der Wintersaison 2014/15 hatte das Schigebiet Sölden 33 Liftanlagen in Betrieb. In den Daten nicht vorhanden waren das *Zentrum-Shuttle*, die beiden Schlepplifte *Innerwald I+II* und die *Rotkoglbahn*. Es wurde nicht versucht, die zwei Übungslifte *Mini Giggijoch* und *Mini Tiefenbach* zu finden. Die restlichen 27 Lifte im Schigebiet wurden gefunden. Der Datensatz Sölden war viel größer als der Datensatz Saalbach und daher konnte das Schigebiet vollständiger detektiert werden. Es wurden fast alle Pisten gefunden. Die Rasterverbindungen lieferten ein Netz von Punkten, so dass jeder Liftpunkt im Schigebiet mit allen anderen Liften über das Netz verbunden war. Als Sonderfall erwiesen

sich im Datensatz die Lifte *Seiterjöchl* und *SL Panorama*. Beide Lifte waren Schlepplifte und beide waren am Gletscher gebaut. Beim Lift *Seiterjöchl* passte der Eintrittspunkt im Vergleich mit den Angaben von OSM nicht überein. Der Lift wurde nur aus wenigen Pfaden bestimmt. Daher ist die Qualität dieses Liftes fragwürdig. Der Lift *SL Panorama* hatte den Sonderfall, dass für den Lift zwei Lifttrassen berechnet wurden, wobei beide Trassen am gleichen Punkt aufhörten und nur der Einstiegspunkt sich unterschied. Da beide Lifte aus mehreren Pfaden bestanden, konnte keine der beiden Varianten als Lösung angenommen werden.

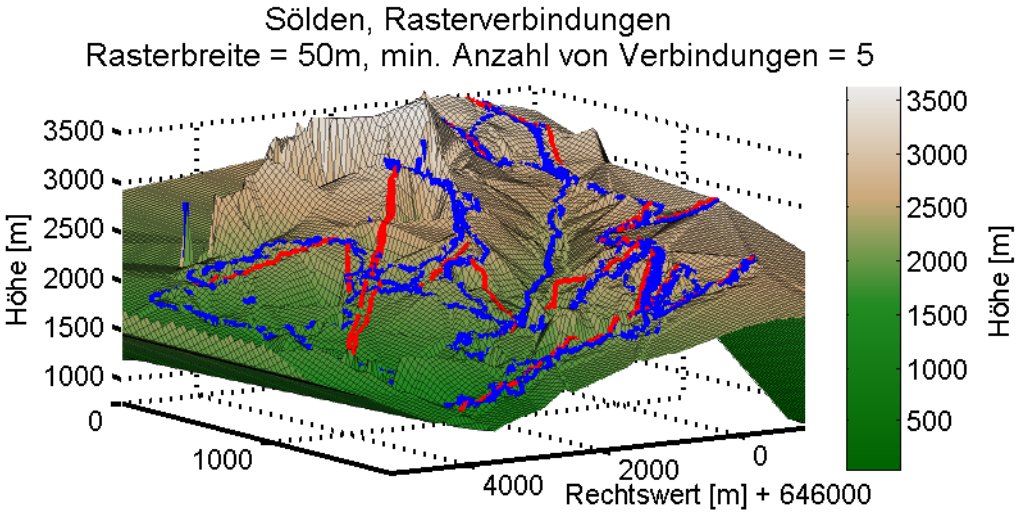


Abbildung 4.19.: 3D-Modell-Übersicht für Sölden, Quelle: eigene Darstellung

Je größer die Datenmenge eines Schigebietes war, desto vollständiger konnte ein Schigebiet detektiert werden. Im Grunde waren die 21 Trajektorien für das Schigebiet Saalbach nur ein kleiner Prozentsatz von Touristen, die täglich das Schigebiet besuchten. Zum Vergleich hatte „*Saalbach im Februar 2015 etwas weniger als 900000 Nächtigungen*“ (www.saalbach.or.at). Ähnlich waren die 81 Trajektorien im Schigebiet Sölden zu werten. Je größer der Datensatz war, desto vollständiger konnten die Pisten und Seilbahnanlagen eines Schigebietes bestimmt werden.

5. Ausblick

Aufbauend auf dem Netz von Pistenpunkten und den extrahierten Liften kann Navigation bzw. Guidance im Schigebiet betrieben werden. Beim OPTICS-Algorithmus geht es um die Suche nach Punkten, die über die Dichte mit anderen Punkten erreichbar bzw. verbunden sind. Gleiches gilt für Navigationslösungen, wo ein Schifahrer eine Startposition hat und eine Zielposition vorgibt. Die Position kann einer Rasterzelle zugewiesen werden. Von der Ausgangsrasterzelle muss ein Pfad entlang von Rasterzellen zum Zielpunkt gefunden werden. Es geht dabei um die Suche nach Wegen, entlang welcher zwei Punkte über ein Netz von Punkten verbunden bzw. erreichbar sind.
Eine weiterer Aufbau auf die Arbeit ist ein System, welches Pistenmanagement ermöglicht. Das Verhalten von Schifahrern kann analysiert werden. Pro Rasterzelle kann die Anzahl von Schifahrern bestimmt werden. Vor allem in der Nähe von Liftpunkten sind überdurchschnittlich viele Schifahrer pro Rasterzelle. Die Kapazität der Lifte ist bekannt und wenn in einer Region eine kritische Masse von Schifahrern erreicht wird, so dass es zu Überlastungen der Lifte kommen könnte, kann diese Informationen über Monitore den Schifahrern weitergeben werden, damit diese andere Regionen des Schigebietes aufsuchen. Mit einem Pistenmanagementsystem werden Schifahrer gesteuert und optimal über das gesamte Schigebiet verteilt um Wartezeiten bei den Liften zu vermeiden.
Die Software bietet sich auch als Analysetool für Seilbahnbetreiber an. Wie bereits erwähnt, kann die Anzahl von Schifahrern pro Rasterzelle bestimmt werden. Wenn bei der Analyse die Zeitkomponente berücksichtigt wird, kann die Pistenpräparierung optimiert werden, da während des laufenden Betriebes kritische Areale erkannt und behoben werden können. Eine weitere Möglichkeit ist die Auslastung von Liften zu bestimmen bzw. dort neue Lifte zu planen, wo die Kapazität nicht ausreicht.
Ein weiteres Anwendungsgebiet ist der Bereich der Security und Überwachung. Schifahrer können entlang der Rasterzellen getrackt werden. Angenommen auf einer Piste geschieht ein Unfall und der Täter flüchtet, dann ist es möglich, den Unfallort einer Rasterzelle zuzuweisen und die Tracks, die sich zum Unfallzeitpunkt in der Rasterzelle befinden, zu ermitteln. Die Pfade können entlang der Rasterzellen verfolgt werden. In Kombination mit den Checkpoints bei Lifteintrittspunkten und den Überwachungskameras kann der Täter ausfindig gemacht werden.
Auch die Schifahrer profitieren von den Daten, die mit diesem Programm extrahiert

werden. Jeder Schifahrer kann sich seine persönliche Statistik des Schitages erstellen lassen, die der Schifahrer downloaden und mit anderen Personen über soziale Netzwerke teilen kann. Bei einer genügend großen Datenanzahl kann aus den Daten der Schifahrer auch 3D-Modelle von Schigebieten berechnet werden. Anhand dieser Modelle ist es möglich virtuelle Überflüge über das Schigebiet oder entlang der Schitrajektorien zu programmieren. Die Karten können auch öffentlich gemacht werden und die User hätten die Möglichkeit, Pisten und Lifte zu editieren und zu bewerten, ähnlich wie es bei *OpenStreetMap* der Fall ist, wo User sowohl die Rolle der Benützer als auch der Moderatoren übernehmen.
Neben dem Aufbau von Applikationen auf das Datennetz sind auch bei der Extraktion der Schigebiete noch Erweiterungen möglich. Ein Ziel ist die Detektion von großen Zubringern - Standseilbahnen, die durch Tunnel Schifahrer auf den Berg bringen. Damit verbunden ist die Suche nach Parkplätzen, da die Trajektorien zumeist an diesen Stellen beginnen. Ein weiteres Element sind Schihütten. Interessant ist auch die Unterscheidung der Lifte zwischen Luftseilbahnen und Liften, die entlang der Oberfläche verlaufen. Die Höhen von Liften, die entlang der Oberfläche verlaufen, können dem Höhenmodell für die Pisten entnommen werden, welches auf Grund der größeren Datenanzahl genauer ist.
Aus technischer Sicht ist es interessant den DeLiClu-Algorithmus (siehe Achtert et al. 2006: S. 199-128) [1] , welcher eine Erweiterung des OPTICS-Algorithmus darstellt, zu implementieren. Der Algorithmus eliminiert den ϵ-Parameter und weist eine verbesserte Performance durch Verwendung eines R-Baumes aus. Damit können die Daten in Teilmengen verlustfrei geteilt werden. Dies verhindert im Umkehrschluss auch, dass bei der Bestimmung von kleinsten Teilmengen und Clustern das Rekursionslimit erreicht wird.
Eine weitere Erweiterung der Software ist es, den Download von Daten des *OpenStreetMap*-Servers einzubauen, um den Vergleich mit diesen Daten zu ermöglichen. Die berechneten Lifte werden im Datensatz von OSM gesucht. Wenn ein Pendant gefunden wird, wird der Name aus den OSM-Daten übernommen. Damit ist es möglich, den berechneten Liften automatisch Liftnamen zuzuweisen.

Anhang

Anhang A.

Auflistung aller detektierten Lifte

Saalbach

In Tabelle A.1 sind alle Lifte aufgelistet, die im Datensatz Saalbach gefunden wurden. In der ersten Spalte ist die Bezeichnung des Liftes im Programm angegeben und in der zweiten Spalte der Name eines Liftes, wie er in *Openstreetmap* verwendet wurde. Die Lifte können anhand der Nummern in der Abbildung B.1 betrachtet werden. In der dritten Spalte sind die berechneten Längen der Lifte angegeben und in der Spalte Δs die Differenzen der berechneten Liftlängen zu den Liftlängen, die aus OSM entnommen wurden. In der fünften Spalte Δs_A stehen die Abstände zwischen den Anfangspunkten der berechneten Lifte und den Anfangspunkten der Lifte in OSM, gleich wie in der sechsten Spalte, nur dass dort die Differenzen für die Endpunkte der Lifte eingetragen sind. In der letzten Spalte stehen die Anzahlen von Pfaden, die für die Berechnung eines Liftes verwendet wurden.

Nr.	Liftname	s [m]	Δs [m]	Δs_A [m]	Δs_E [m]	Pfadanzahl
1	Panorama 6er	1314	19	2	18	16
2	Kohlmais 3er Sessellift	1169	4	14	18	4
4	Kohlmaisgipfelbahn I **	1085	-	-	-	4
3	Kohlmaisgipfelbahn II **	1080	-	-	-	4
5	8er Bernkogelbahn	1919	23	12	12	6
6	Bernkogel Schlepplift III + IV	1335	7	7	3	4
7	Limberg 4er Sessellift	863	24	13	11	5
8	Reiterkogel Ost	858	21	8	16	6
9	Sunliner	1336	11	10	23	3
10	Hasenauer Kopf Sessellift	854	9	8	4	7
11	Westgipfelbahn II	1286	33	27	7	5
12	Westgipfelbahn I	1187	18	11	12	4
13	Zwölferkogelbahn I	1471	27	15	15	4
14	Zwölfer Nordbahn	1695	28	29	5	6
15	Zwölferkogelbahn II	925	44	34	11	5

16	6er Sesselbahn Zehner	1376	11	15	6	5
17	Magic 6er	1275	8	5	5	6
18	Schönleitenbahn	1055	24	18	10	9
19	Schönleiten 6er Gipfelbahn	1199	4	7	11	16
20	Schönleitenbahn	1504	36	29	9	6
21	Poltenlift *	477	4	10	2	7
22	Sportbahn Asitzkogel 2000	807	0	4	10	4
23	Asitzmuldenbahn	534	4	7	8	9
24	Asitzgipfelbahn	1078	3	9	12	14
25	Asitzbahn II **	1908	-	-	-	3
26	Hochalm Sesselbahn	1669	26	16	16	4
27	Spieleck 6er Sesselbahn	1634	26	40	17	4

Tabelle A.1.: berechnete Lifte aus dem Datensatz Saalbach

Der mit einem * gekennzeichnete Lift ist ein alter Lift, der in der Wintersaison 2014/15 ersetzt wurde und so nicht mehr besteht. Trotzdem sind die Differenzen zum OSM-Lift klein, was darauf schließen lässt, dass auch im OSM-Datensatz noch der alte Lift eingetragen ist. Die mit zwei ** gekennzeichneten Lifte sind lange Lifte mit jeweils einer Mittelstation. Diese Lifte wurden im Programm getrennt als zwei Lifte detektiert. Im OSM-Datensatz sind die Lifte als ein gemeinsamer langer Lift eingetragen.

Sölden

In Tabelle A.2 sind alle Lifte aufgelistet, die im Datensatz Sölden gefunden wurden. Die Inhalte der Spalten wurden bereits oben erklärt.

Nr.	Liftname	s [m]	Δs [m]	Δs_A [m]	Δs_E [m]	Pfadanzahl
1	Roßkirpl	1609	40	30	11	17
2	Hainbachkar	1278	16	11	7	10
3	Silberbrünnel	1713	14	15	2	27
4	Giggijochbahn	2506	0	9	3	6
5	Giggijoch	1218	17	5	12	39
8	Gaislachkoglbahn I	1871	33	17	19	11
6	Gaislachkoglbahn II	1815	33	12	25	24

7	DSB-Mittelstation	1657	8	2	14	5
9	Heidebahn	1568	1	5	7	21
10	Langegg	2146	1	5	4	45
11	Seekogel	877	18	9	9	16
12	Stabele	998	10	7	6	17
13	Einzeiger	1080	12	7	10	53
14	Schwarzkogel	1017	19	29	10	10
15	Schwarze Schneid I	880	21	11	13	53
16	Schwarze Schneid II	1013	14	12	9	63
17	Seiterkar	879	0	4	3	47
18	Seiterjöchl **	850	-	-	-	5
19	Tiefenbachbahn	2013	8	4	5	66
20	Mutkogl	2084	2	8	7	10
21	Panorama **	952	-	-	-	5
22	Wasserkar *	829	-	-	-	19
23	Gratl	989	15	6	21	8

Tabelle A.2.: berechnete Lifte aus dem Datensatz Sölden

Der mit einem * gekennzeichnete Lift ist ein alter Lift, der in der Wintersaison 2014/15 ersetzt wurde und so nicht mehr besteht. Die mit zwei ** gekennzeichneten Lifte sind beides Lifte, die auf einem Gletscher stehen. Beim *Panorama*-Lift stimmt der Austrittspunkt mit dem Lift in OSM überein, allerdings ist der Eintrittspunkt des berechneten Liftes auf der halben Strecke des Liftes in OSM. Die Pfadanzahl spricht dafür, dass an der berechneten Stelle ein Eintrittspunkt ist, allerdings zu einer anderen Jahreszeit z.B. im Sommer. Es kann der Eintrittspunkt auch pro Jahr variieren. Der *Seiterjöchl*-Lift steht ebenfalls auf dem Gletscher. Auch für diesen Lift gilt, dass der Austrittspunkt mit dem Austrittspunkt des Liftes in OSM übereinstimmt, aber nicht der Eintrittspunkt. Auch für diesen Lift gilt das Gleiche wie für den *Panorama*-Lift.

Anhang B.

Auflistung aller detektierten Pisten

Saalbach

Saalbach hatte in der Wintersaison 2014/15 63 Pisten. Von den 63 Pisten wurden 27 Pisten gefunden. In Tabelle B.2 sind alle detektierten Pisten im Datensatz Saalbach aufgelistet. In den Spalten „A“ sind die Nummerierungen der Pisten, wie sie vom Seilbahnbetreiber verwendet werden, eingetragen. Jeder Lift in einem Schigebiet erhält üblicherweise eine Nummer und einen Namen. Bei den Pisten sind zumeist nur die Pistennummern in Schigebietsplänen eingetragen. In den Spalten „B“ steht pro Zeile die Anzahl von berechneten Pisten, die zur Pisten mit der Pistennummer in Spalte A dazugehören. Das bedeutet, dass es pro Piste laut Schigebietsplan, eine unterschiedliche Anzahl von berechneten Pistenverläufen gibt. Pistenvarianten wurden jeweils der Ursprungspiste zugewiesen (z.B. Piste *22* und *22a* siehe Abb. B.1). Es gibt berechnete Pisten, die mehr als einer Piste angehören, weil Schifahrer öfters eine Kombination von unterschiedlichen Pisten befahren. Auf Grund der Nummerierung kann nicht auf Pisten geschlossen werden, die vom Programm nicht detektiert wurden, da die Nummerierung der Pisten im Schigebiet Saalbach nicht der Reihenfolge nach nummeriert wird. Die Nummern 0, 10 ,20 usw. werden nie einer Piste gegeben, sondern definieren eine Region, wo alle Pisten mit einer Zahl z.B. zwischen 21 und 29 beginnen und nicht jede Region besteht aus genau neun Pisten. Anders ist die Nummerierung der Pisten im Schigebiet Sölden, die nachfolgend an das Schigebiet Saalbach angeführt sind.

A	B		A	B		A	B		A	B
2	2		28	1		59	1		66	2
4	1		32	1		60	1		81	2
7	1		46	1		61	1		82	2
11	3		48	2		62	2		83	1
15	1		52	2		63	1		87	5
17	1		55	1		64	1		88	1
22	2		57	1		65	3			

Tabelle B.1.: Pisten Saalbach

Aufgrund der zu geringen Datenanzahl konnten nicht mehr Pisten detektiert werden. In Bild B.1 ist ein Abbild des Schigebiets in der Wintersaison 2014/15 dargestellt.

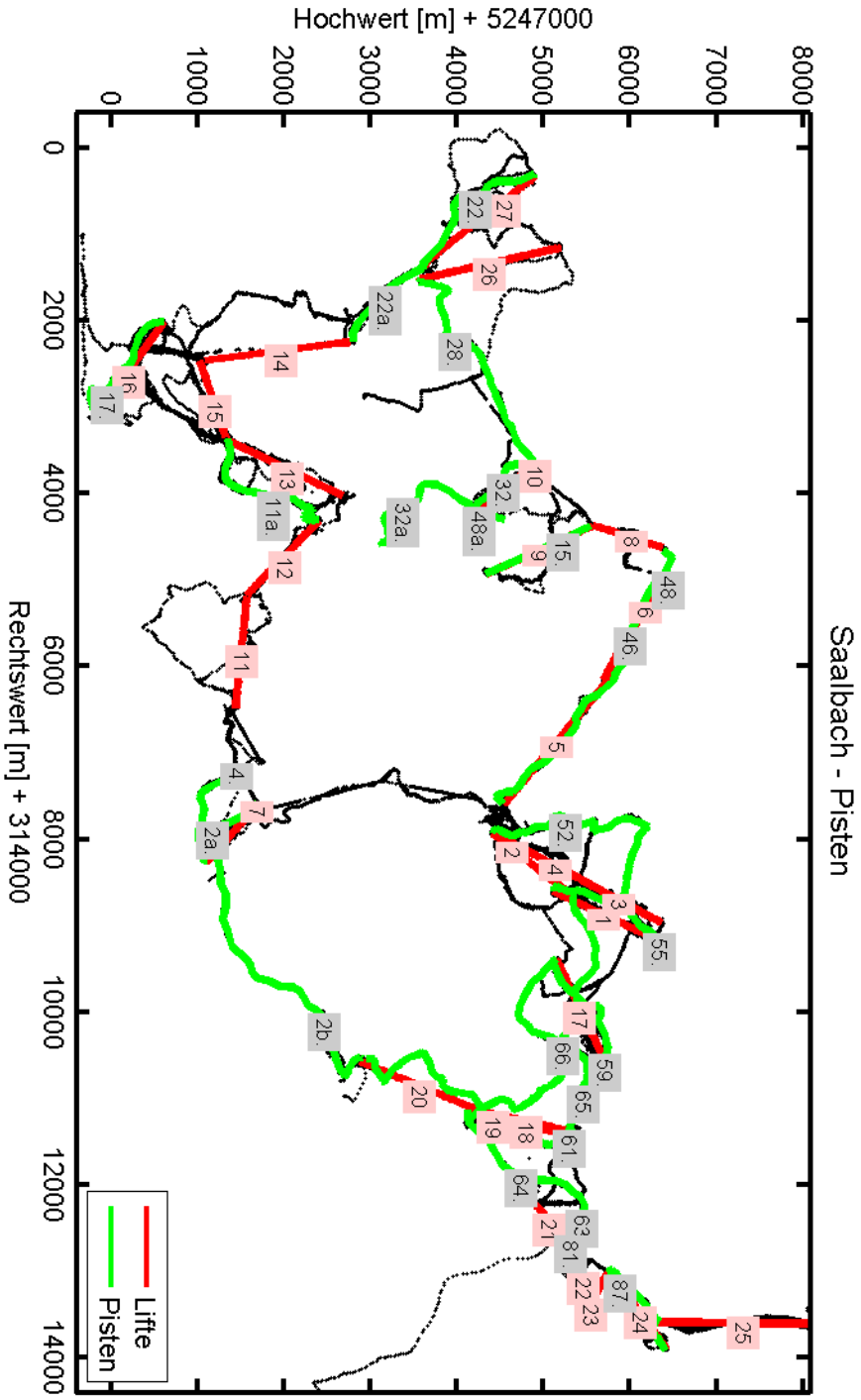


Abbildung B.1.: Übersichtsplan Saalbach 2014/15, Quelle: eigene Darstellung

Sölden

Für den Datensatz Sölden wurden 148 Pisten gefunden. Lauf Liftbetreiber gab es im Jahr 2014/15 36 Pisten, die sich noch auf weitere Varianten spalteten. Viele der berechneten Pisten waren nur ganz kurz z.B. Verbindungen von Liften in Regionen, wo Lifte nahe zusammen gebaut worden sind. Fünf Pisten im Schigebiet Sölden konnten nicht detektiert werden. Nicht jede Piste, die man auf Grund der Reihenfolge der Nummerierung vermuten würde, gab es tatsächlich. Die Pisten 10, 12, 16 und 35 wurden nicht detektiert, weil sie in den Daten nicht vorhanden waren. Die Pisten 17 und 26 bis 29 waren im Schigebietsplan der Wintersaison 2014/15 nicht eingetragen. Diese Pisten befanden sich in der Nähe eines Liftes, der abgebaut und nicht neu aufgestellt wurde. Dadurch kam auch die Nummerierung durcheinander. Die Pisten 40 und 41 waren im Schigebietsplan des Jahres 2014/15 zwar angeführt, aber für den Betrieb geschlossen. Beide Pisten werden nur freigegeben, wenn es die Schneeverhältnisse zulassen.

A	B		A	B		A	B		A	B
1	13		9	5		21	1		33	10
2	6		11	4		22	3		36	1
3	5		13	8		23	5		37	1
4	5		14	3		24	5		38	10
5	4		15	7		25	2		39	5
6	6		18	2		30	5			
7	4		19	3		31	4			
8	1		20	1		32	3			

Tabelle B.2.: Pisten Saalbach

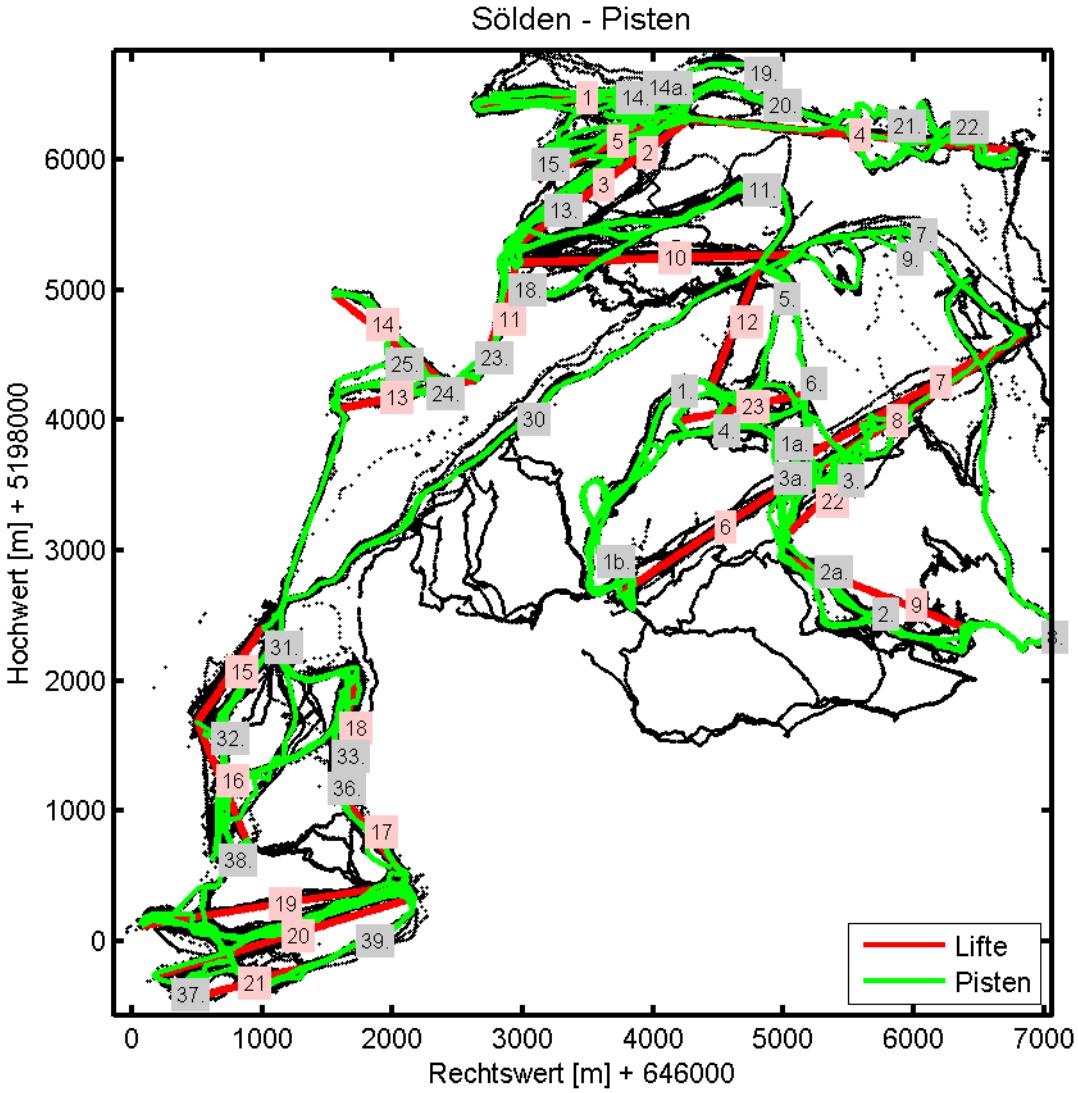


Abbildung B.2.: Übersichtsplan Sölden 2014/15, Quelle: eigene Darstellung

Literaturverzeichnis

[1] Achtert E., C. Böhm, P. Kröger: *DeLi-Clu: Boosting Robustness, Completeness, Usability, and Efficiency of hierarchical Clustering by a Closest Pair Ranking. In Advances in Knowledge Discovery and Data Mining.* Springer Berlin, 2006.

[2] Ankerst M., M. Breunig, H. Kriegel, J. Sander: *OPTICS: Ordering Points To Identify the Clustering Structure.* 1999.

[3] API Schnittstelle der Internetseite *GPSies*: *www.gpsies.com/api/GPSiesAPI.pdf*.

[4] Defays D.: *An efficient algorithm for a complete link method.* 1977.

[5] Dokumentation der Elevation API Schnittstelle der Internetseite *Google*: *developers.google.com/maps/documentation/elevation.*

[6] Dokumentation der Elevation API Schnittstelle der Internetseite *mapquest*: *developer.mapquest.com/web/products/open/elevation-service.*

[7] Du Q.„ V. Faber, M. Gunzburger: *Centroidal Voronoi Tessellations: Application and Algorithms.* SIAM Review, 1999.

[8] Ester M., H. Kriegel, J. Sander, X. Xu: *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.* 1996.

[9] Fayyad U., S. Chakrabarti, M. Ester, J. Gehrke, J. Han, S. Morishita, W. Wang: *Data Mining Curriculum: A Proposal.* 2006.

[10] Hofmann.Wellenhof B., G. Kienast: *Bezugssysteme.* Institut für Navigation und Satellitengeodäsie, TU Graz, 2010.

[11] Rokach L. , O. Maimon: *Data Mining and Knowledge Discovery Handbook.* Springer US, 2005.

[12] Sander J., X. Qin, Z. Lu, N. Niu, A. Kovarsky: *Automatic Extraction of Clusters from Hierarchical Clustering Representations.* 2003.

[13] Sibson R.: *SLINK: An optimally efficient algorithm for the single-link cluster method.* 1972.

[14] Sundburg R.: *Maximum Likelihood Theory for Incomplete Data from an Exponential Family.* Scandinavian Journal of Statistics: Theory and Applications 1, 1974.

[15] Zhang T., R. Ramakrishan, M. Livny: *BIRCH: An Efficient Data Clustering Method for Very Large Databases.* 1996.