Fakultät Elektrotechnik, Medien und Informatik
Studiengang Medieninformatik

Bachelorarbeit

von

Aaron K r e u z e r

Möglichkeiten der Automatisierung von benutzergesteuerter Kombination von GPS Tracks

Possibilities of automation of user-controlled combination of GPS tracks

Fakultät Elektrotechnik, Medien und Informatik
Studiengang Medieninformatik

Bachelorarbeit

von

Aaron K r e u z e r

Möglichkeiten der Automatisierung von benutzergesteuerter Kombination von GPS Tracks

Possibilities of automation of user-controlled combination of GPS tracks

Bearbeitungszeitraum: von: 11.08.2022
bis: 10.01.2023

1. Prüfer: Prof. Dr. Dieter Meiller
2. Prüfer: Veit Stephan M. Eng.

Ostbayerische Technische Hochschule Amberg - Weiden

Fakultät Elektrotechnik, Medien und Informatik

Eigenständigkeitserklärung gemäß § 27 (8) ASPO

Name und Vorname
der Studentin/des Studenten: **Kreuzer Aaron**

Studiengang: **Medieninformatik**

Ich bestätige, dass ich die Bachelorarbeit mit dem Titel:

Möglichkeiten der Automatisierung von benutzergesteuerter Kombination von GPS Tracks

selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Datum: 09.01.2023

Unterschrift:

Ostbayerische Technische Hochschule Amberg - Weiden

Fakultät Elektrotechnik, Medien und Informatik

Bachelorarbeit Zusammenfassung

Studentin/Student (Name, Vorname): Kreuzer Aaron
Studiengang: Medieninformatik

Aufgabensteller, Professor: Prof. Dr. Dieter Meiller

Durchgeführt in (Firma/Behörde/Hochschule): evidentmedia

Betreuer in Firma/Behörde: Veit Stephan

Ausgabedatum: 11.08.2022 Abgabedatum: 10.01.2023

Titel:

Möglichkeiten der Automatisierung von benutzergesteuerter Kombination von GPS Tracks

Zusammenfassung:

Durch die Entwicklung einer Webanwendung (Prototyp), soll überprüft werden, ob es möglich ist, automatisiert GPS Tracks zu kombinieren. Der Benutzer soll dabei das Ergebnis durch Präferenzen beeinflussen können und auf einer Karte die Tracks und zugehörige Informationen dargestellt bekommen. Der Prototyp dient zur Evaluierung der Durchführbarkeit einer solchen Kombination.

Schlüsselwörter:
GPS, gpx, Geodatenanalyse, Karten, Routing, Webanwendung

Inhaltsverzeichnis

Abkürzungs- und Begriffsverzeichnis

Routing
: Routing bezeichnet das Finden einer passenden Route zwischen zwei Punkten.

GeoJSON
: GeoJSON ist ein geospatiales Datenformat welches auf Javascript Object Notation basiert. Es dient zur Speicherung und Übermittlung von geographischen Daten (z.B. ein Polygon).

Point of Interest
: Point of Interst (PoI) ist ein Begriff um einen bestimmten/wichtigen Ort oder eine Sehenswürdigkeit zu beschreiben. Für den Prototyp ist es ein Ort der unbedingt ein Teil des Tracks sein soll.

HTTP
: Das Hypertext Transfer Protocol ist ein Netzwerkprotokoll für die Übertragung von Daten zwischen Client und Server im World Wide Web.

API
: Das Application Programming Interface ist eine Schnittstelle für den Datenaustausch zwischen zwei Systemen.

Abbildungsverzeichnis

Listings

Tabellenverzeichnis

1 Einleitung

1.1 Forschungsmotivation

Ein stressiges Berufsleben und Faktoren wie die SARS-CoV-2-Pandemie führen dazu, dass immer mehr Menschen ihre Freizeit in der Natur verbringen. Dies zeigt zum Beispiel auch der extreme Aufschwung des Fahrradmarkts nach dem Ausbruch der Pandemie und einhergehende Regeln (ZIV, 2022). Des Weiteren ist kein Ende der Steigerung bei der Nutzung von Smartphones (Koptyug, 2021) und GPS (Laricchia, 2022) zu sehen. Diese digitalen Tools ersetzen heute häufig analoge Karten und Routenplanung.

Im Gegensatz dazu sind hochwertige Fahrrad- oder Wanderstrecken eine begrenzte Ressource. Offizielle Strecken starten oft an Gondelstationen oder in Stadtzentren, was die Reichweite von Strecken einschränkt. Die meisten Strecken sind für die breite Masse ausgelegt und können so selten allen individuellen Ansprüchen gerecht werden. Den Menschen in diesem Bereich mehr Vielfalt und Anpassungsmöglichkeiten, durch Kombination von Tracks, zu ermöglichen, könnte die Qualität der Freizeit zusätzlich anheben.

Möchte man aktuell zwei Tracks (s. Abb. 1) miteinander verbinden, dann fügen Tools die Datensätze einfach nur zusammen. Durch diese Zusammenführung entsteht eine Verbindung vom Ende (roter Punkt) des ersten Tracks zu dem Start (grüner Punkt) des zweiten Tracks (s. Abb. 2). Wünschenswert wäre aber eine sinnvolle Zusammenführung für die wirkliche Nutzung des Tracks. Dies bedeutet zum Beispiel das Entfernen von Waypoints und das Setzen von neuen Verbindungen (s. Abb. 3). Die neuen Verbindungen können dann mittels Routing durch einen korrekten Pfad ersetzt werden (s. Abb. 4).

Abbildung 1: Ausgangspunkt zweier Tracks

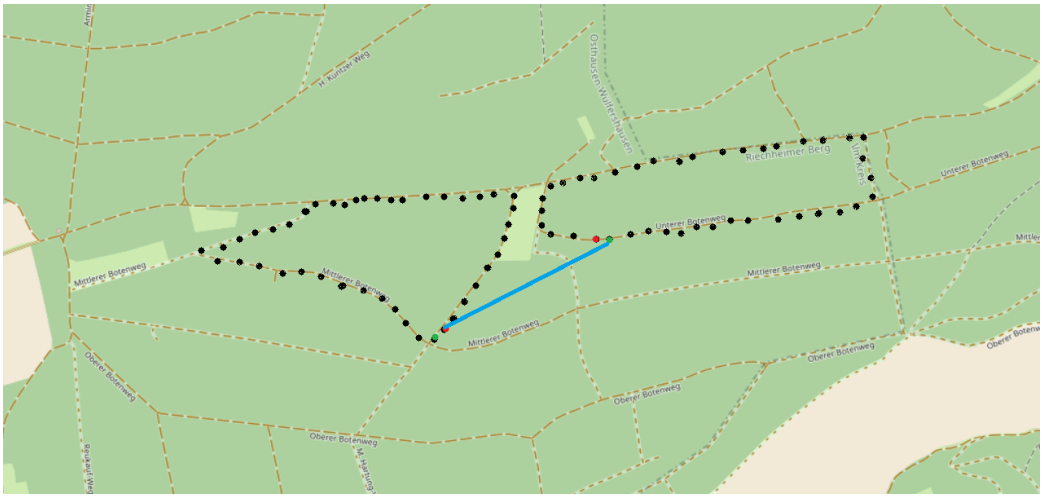

Abbildung 2: Einfache Verknüpfung der Tracks

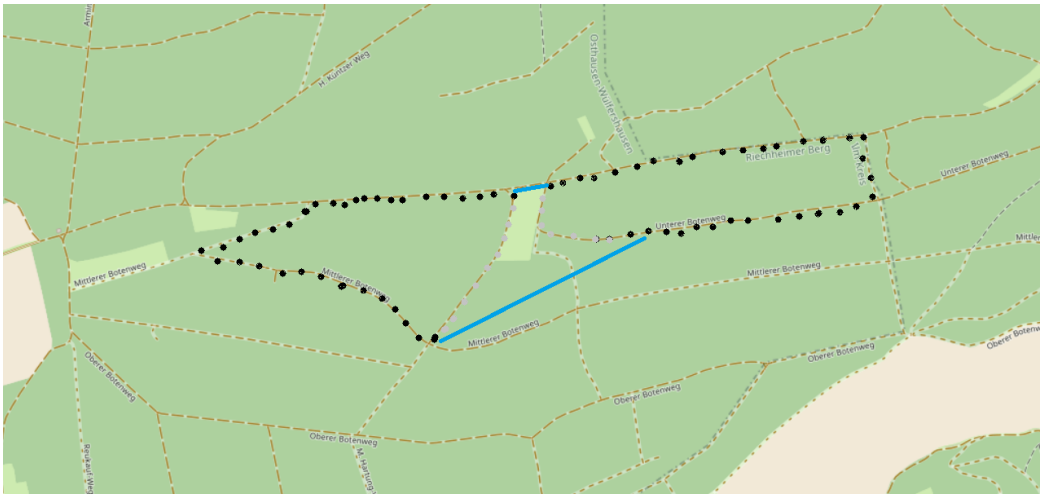

Abbildung 3: Sinnvolle Verknüpfung der Tracks

Abbildung 4: Verknüpfung der Tracks mit Routing

1.2 Ziel der Arbeit

Ziel der Arbeit ist es, sich mit der Frage zu beschäftigen, ob und wie mehrere Tracks sinnvoll nach bestimmten Benutzerkritierien zusammengefügt werden können und wie man dies möglichst automatisieren kann. Für diese Fragestellung wird ein Prototyp mit den entsprechenden Funktionalitäten entwickelt.

Mithilfe des Prototyps sollen die Anforderungen an solch ein Tool analysiert und evaluiert werden. Folgende Fragen sind zu beantworten:

- Wie kann die Kombination von GPS-Tracks durch Automatisierung unterstützt werden?
  - Welche verschiedenen Situationen ergeben sich für die Kombination von Tracks?
  - Welche Prozesse bei der Kombination von Tracks können durch Automatisierung unterstützt werden?
  - Welche mathematischen Algorithmen können genutzt werden?
- Welche Anbieter und Services bieten sich für die Automatisierung an?

2 Aktueller Forschungsstand

In diesem Kapitel soll der aktuelle Forschungsstand im Bezug auf, für diese Arbeit relevanten, Informationen, Applikationen und Dienstleistungsanbieter dargestellt werden.

2.1 Grundlagen

2.1.1 Onlinekarten

Das Geographic Information System ermöglicht Erfassung, Speicherung und Verwaltung von Geodaten. Das System ist daher besonders nützlich um Geodaten zu analysieren. Auch für die Onlinekarten spielt das GIS eine wichtige Rolle, denn die Karten können aus Geodatensätzen generiert werden (s. Abbildung 5). Nur durch solch ein System lassen sich die Onlinekarten effizient aktuell halten.

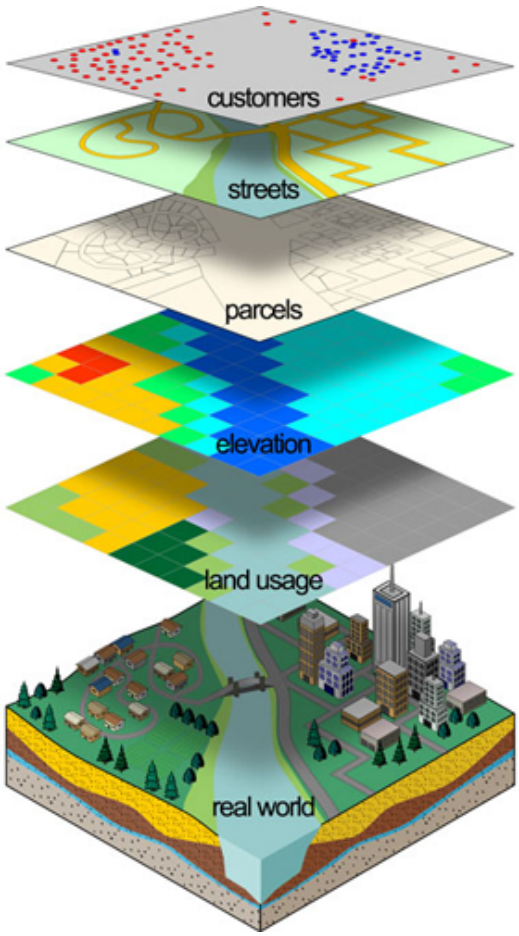

Abbildung 5: Visualisierung von Informationsschichten aus denen eine Karte erstellt wird

Quelle: Westfield State University

Onlinekarten werden meistens, in standardisierter Form, kachelbasiert angeboten (s. Abb. 6). Dies macht es besonders einfach zwischen verschiedenen Anbietern zu wechseln. Generell wird zwischen Referenzkarten und thematischen Karten unterschieden. Eine Referenzkarte zeigt einen relativ großen Abschnitt an und dient dazu, einzelne Bereiche und Objekte in Beziehung zu setzen. Die thematische Karte ist detailreicher und zielt darauf ab, den Nutzer über ein bestimmtes Thema zu informieren. Themen sind bestimmte Informationen die im Fokus der Darstellung liegen, dies könnte zum Beispiel

die Bevölkerungsdichte sein. (DeMers, 2014, S. 27-30) Bekannte Anbieter dafür sind zum Beispiel Google (2022b), Microsoft (2022), Apple (2022) oder OpenStreetMap (2022).

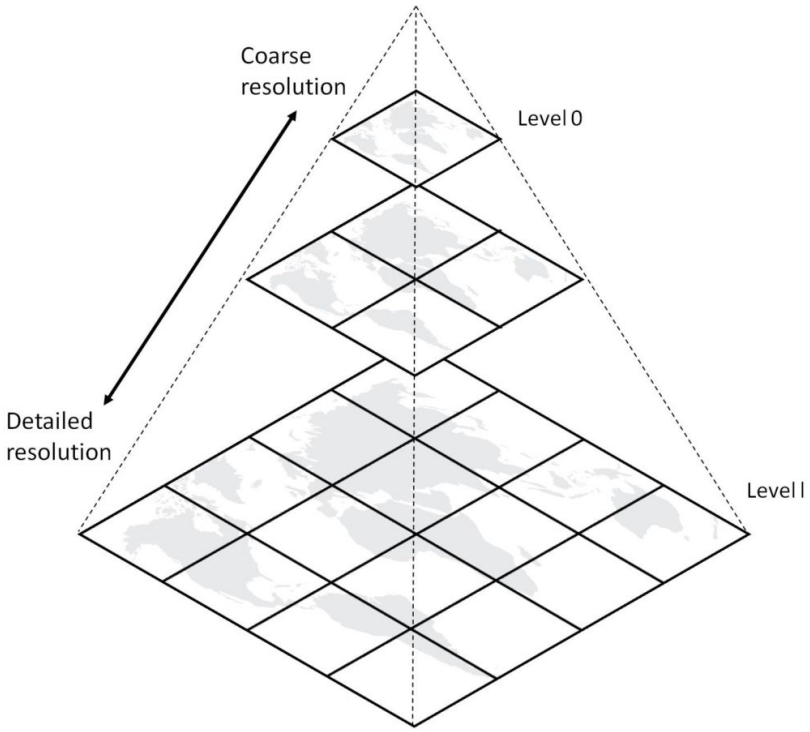

Abbildung 6: Kachelbasierte Karte mit Zoomstufen

Quelle: Open Geospatial Consortium

2.1.2 GPS

Das Global Positioning System wird seit 1973 vom US-Militär entwickelt. Es handelt sich dabei um ein globales Navigationssatellitensystem, welches durch Radiosignale zwischen Empfänger und Satelliten die Positionsbestimmung möglich macht. (El-Rabbany, 2002) Bis zum 2. Mai 2000 fand eine künstliche Signalverschlechterung statt, was die Genauigkeit für zivile Nutzer deutlich senkte. Ohne Signalverschlechterung sind Abweichungen unter 5 Metern die Regel. Im Gegensatz dazu waren vor der Abschaltung Abweichungen von bis zu 45 Metern normal. (National Coordination Office, 2018,

2021) Im Umgang mit GPS sollte man sich also immer im Klaren sein, dass konstant mit kleineren Abweichungen umgegangen werden muss.

Für die Speicherung von Geodaten wird häufig das GPS Exchange Format (GPX) verwendet. Dabei handelt es sich um ein offenes und lizenzfreies Datenformat, welches auf dem XML-Standard basiert. Die gpx-Datei kann aus folgenden Koordinatenelementen bestehen (Topografix, 2022):

1. **wptType**: ein individueller Wegpunkt
2. **rteType**: eine nach Ablauf geordnete Liste von Wegpunkten bzw. Wendepunkte für einen beschriebenen Weg
3. **trkType**: eine nach Ablauf geordnete Liste von Wegpunkten, welche eine Linie bzw. einen Pfad/Track bilden

2.1.3 Apps

Es gibt eine Anzahl von Websites und Tools auf Basis von Karten und GPS Tracks. Eine Feature Inspection der gängigsten Tools hat gezeigt, dass wenn überhaupt eine Verbindung von zwei Tracks möglich ist, nur Start- und Endpunkt beider Tracks miteinander verknüpft werden. Die dafür untersuchten Tools waren: GPXStudio (2022), GPSTrackEditor (2022), Komoot (2022), Outdooractive (2022), Garmin (2022), Strava (2022), Wanderreitkarte (2022), GarminBasecamp (2022) und MyGPSFiles (2022).

2.2 Routenplanung

Ein wichtiger Aspekt für den Prototyp ist, zu wissen wie der Mensch in der Regel eine Route plant. Nur durch dieses Wissen kann eine vernünftige Automatisierung angestrebt werden. Zusammenfassen lässt sich die Routenplanung in folgenden vier Schritten:

1. Es wird ein Start-/Endpunkt gewählt. Dabei handelt es sich um einen Standort der gut erreichbar ist und möglichst nah an der gewünschten Strecke liegt.
2. Die Strecke entsteht durch Standorte oder Gebiete von Interesse (Point of Interest), die erreicht werden sollen.

3. Je nach Fortbewegungsmittel und Ziel des Unterfangens gibt es verschiedene Präferenzen für die Art der Strecke. Zum Beispiel möchte man bei einem Wanderausflug in den meisten Fällen die Straße vermeiden.

4. Zusätzliche Präferenzen wie Höhenmeter, Streckenlänge und Hindernisse (Treppen) können weiteren Einfluss auf die Wahl der Streckenabschnitte haben.

3 Anforderungsanalyse und Planung

3.1 Analyse der möglichen Kombination von Tracks

In diesem Abschnitt geht es um die Analyse der möglichen Kombination von Tracks und die Anforderungen die dabei entstehen.
Um das Ziel einer sinnvollen, benutzergesteuerten Kombination von Tracks zu ermöglichen, ergeben sich folgende Anforderungen:

1. Verarbeitung verschiedener Tracktypen und -konstellationen
2. Berücksichtigung von Benutzerpräferenzen
3. Kombination und Erstellung eines neuen Tracks

Der Umgang des Systems mit verschiedenen Tracktypen und deren Konstellation, ist bei der Kombination eine wichtige Anforderung um den Prototyp umfangreich einsetzen zu können. Jeder Tracktyp sowie verschiedene Konstellationen haben Auswirkungen auf die Anforderungen an das System.

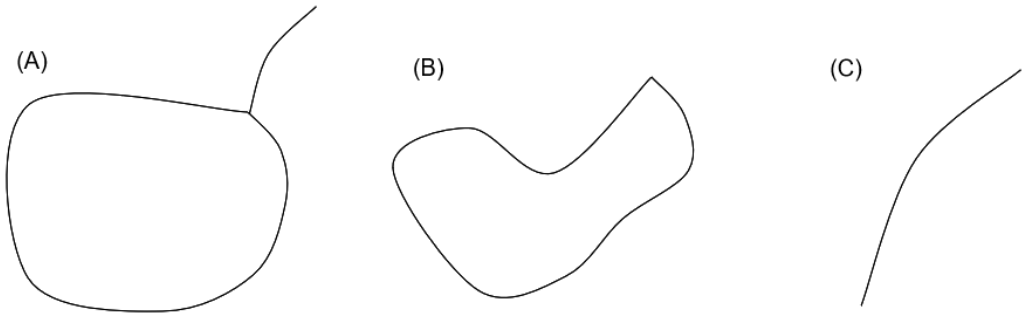

Abbildung 7: Drei verschiedene Tracktypen

In dieser Arbeit sollen die drei verschiedenen Tracktypen aus Abbildung 7 berücksichtigt werden. Bei Tracktyp `Rundstrecke mit Hin- und Rückweg` (A) handelt es sich um einen Rundweg mit zusätzlichem Hin- und Rückweg. Das heißt, der Start- und Endpunkt ist nicht Teil des eigentlichen Tracks. Ein solcher Abschnitt ist nur interessant, wenn der dazugehörige Start- und Endpunkt verwendet werden soll. Das System sollte also in der Lage sein einen Hin- und Rückweg zu identifizieren und bei der Kombination zu vernachlässigen.

Ein klassischer Rundweg bei dem der Start- und Endpunkt ein Teil der Strecke ist, ist folglich Tracktyp `Rundstrecke` (B). Auch dieser Typ muss von dem System erkannt werden und soll als Standard angenommen werden. Dabei sind keine weiteren Berücksichtigungen dieser Strecke nötig.
Tracktyp `Strecke` (C) ist ebenfalls wichtig, da es bei diesem zu einen Spezialfall kommen kann. Wird hier ein Streckensegment in der Mitte entfernt, erhält das System, im Gegensatz zum Rundweg, zwei getrennte Streckenabschnitte. Dies gilt es, für eine sinnvolle Kombination zu vermeiden.

Neben verschiedenen Streckentypen gibt es auch verschiedene Streckenkonstellationen die das System berücksichtigen muss (s. Abbildung 8).

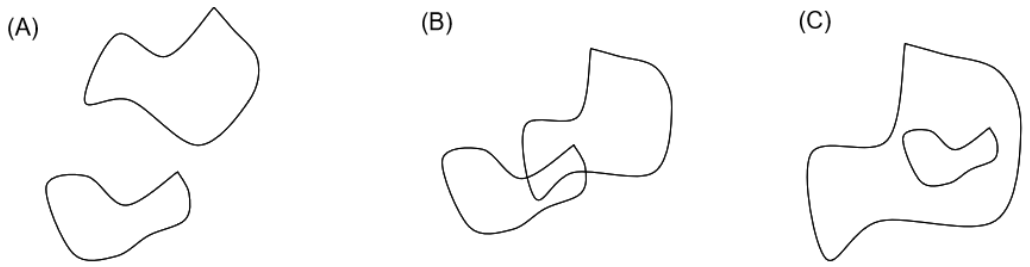

Abbildung 8: Drei verschiedene Trackkonstellationen

Bei Konstellation A gibt es keine Spezialfälle für die Kombination und sie wird als Standard für das System gewählt. Für die weiteren Konstellationen muss das System die Möglichkeit haben sie zu erkennen. Es muss in der Lage sein die Überschneidungen bei Konstellation B sinnvoll aufzulösen und bei Konstellation C sinnvolle Verbindungen zu erstellen. Dies soll für alle Konstellation von unterschiedlichen Tracktypen möglich sein.

Um die Automatisierung benutzergesteuert zu gestalten, muss das System in der Lage sein, bestimmte Benutzerpräferenzen einzulesen. Die Algorithmen des Prototyps müssen dann, anhand der gesetzten Präferenzen, beeinflussbar sein. Wichtige Faktoren stellen die Streckenlänge, Höhenmeter und Points of Interest dar.

Mit all den davor erwähnten Prozessen und Eingaben muss das System die Strecken sinnvoll segmentieren und zusammenfügen, um einen neuen Track, entsprechend der Wünsche des Benutzers, zu erstellen.

3.2 Anforderungen an die Algorithmen

Zusätzlich zu den bereits genannten Anforderungen geht es hier um solche für sehr komplexe Algorithmen, die für den Prototyp entwickelt werden.

Bei den meisten Kombinationen von Tracks müssen neue Streckenabschnitte für die Verbindung erstellt werden. Für diese sehr umfangreiche Aufgabe gibt es viele Tools und Anbieter mit unterschiedlichen Features. Der Prototyp benötigt folgende Features:

1. **Auswahl eines Fortbewegungsmittels** - Um den Prototyp für eine Vielzahl an Benutzern interessant zu machen, müssen erstellte Strecken an das Fortbewegungsmittel angepasst sein.

2. **Alternative Routen** - Um eine hohe Anpassungsfähigkeit zu ermöglichen benötigt der Prototyp Alternativen für Verbindungsabschnitte.

3. **Höhenmeter** - Die Höhenmeter der Tracks sollen dem Benutzer dargestellt werden.

Die geografische Analyse ist ein weiterer wichtiger Bereich für den Prototyp. Dieser benötigt folgende Funktionen:

1. **Erkennung von Überlappungen** - Der Prototyp muss in der Lage sein Überlappungen von Streckenabschnitten zu erkennen.

2. **Erkennung von Überschneidungen** - Eine Erkennung der Schnittpunkte von Tracks ist ebenfalls sehr wichtig.

3. **Erkennung von Umschließungen** - Für die Erkennung verschiedener Konstellationen muss der Prototyp erkennen ob ein Track von einem anderen umschlossen wird.

4. **Richtungserkennung** - Um Streckenabschnitte sinnvoll zusammenzufügen muss bekannt sein in welche Richtung (im Uhrzeigersinn/nicht im Uhrzeigersinn) Abschnitte verlaufen.

3.3 Anforderungskatalog

Die folgenden funktionalen Anforderungen an das System definieren den Umfang des Prototyps und die Funktionen, welche für eine erfolgreiche Umsetzung am relevantesten sind. Nach Fertigstellung des Prototyps werden die Anforderungen der Kern für die Evaluierung sein.

3.3.1 Anforderung FA-1

Ziffer	FA-1
Beschreibung	Das System muss mit den verschiedenen Tracktypen umgehen können.
Zielsetzung	Es muss die verschiedenen Typen identifizieren und diese im weiteren Vorgehen berücksichtigen.
Priorität	Hoch

3.3.2 Anforderung FA-2

Ziffer	FA-2
Beschreibung	Das System muss mit verschiedenen Trackkonstellationen umgehen können.
Zielsetzung	Die unterschiedlichen Konstellationen müssen erkannt und berücksichtigt werden.
Priorität	Hoch

3.3.3 Anforderung FA-3

Ziffer	FA-3
Beschreibung	Das System muss die Benutzerpräferenzen zu Tracklänge und Höhenmeter berücksichtigen.
Zielsetzung	Das Ergebnis muss die Präferenzen, zu Tracklänge und Höhenmeter, des Benutzers, sofern die Tracks dies erlauben, widerspiegeln.
Priorität	Hoch

3.3.4 Anforderung FA-4

Ziffer	FA-4
Beschreibung	Das System soll die Strecken und dazugehörigen Daten (Höhenmeter, Länge der Segmente) visualisieren um dem Benutzer die Entscheidungen verständlich zu machen.
Zielsetzung	Die Strecken und einzelnen Segmente sollen gut erkennbar auf einer Karte visualisiert werden und ein Graph für Höhenmeter muss optional zur Verfügung stehen.
Priorität	Hoch

3.3.5 Anforderung FA-5

Ziffer	FA-5
Beschreibung	Das System muss für die Verbindungen zwischen Strecken eine sinnvolle Route ausgeben.
Zielsetzung	Das Routing sollte für verschiedene Fortbewegungsmittel anpassbar sein (Fuß/Fahrrad) und dem System alternative Routen zur Verfügung stellen.
Priorität	Hoch

4 Konzeption

Bei dem Prototyp wird es sich bereits um ein umfangreiches System handeln. Deshalb sollen grundlegende Bausteine wie Komponenten, Daten und Systemabläufe durch die Konzeption eingegrenzt werden, um eine flüssigere Durchführung zu gewährleisten.

4.1 Komponenten

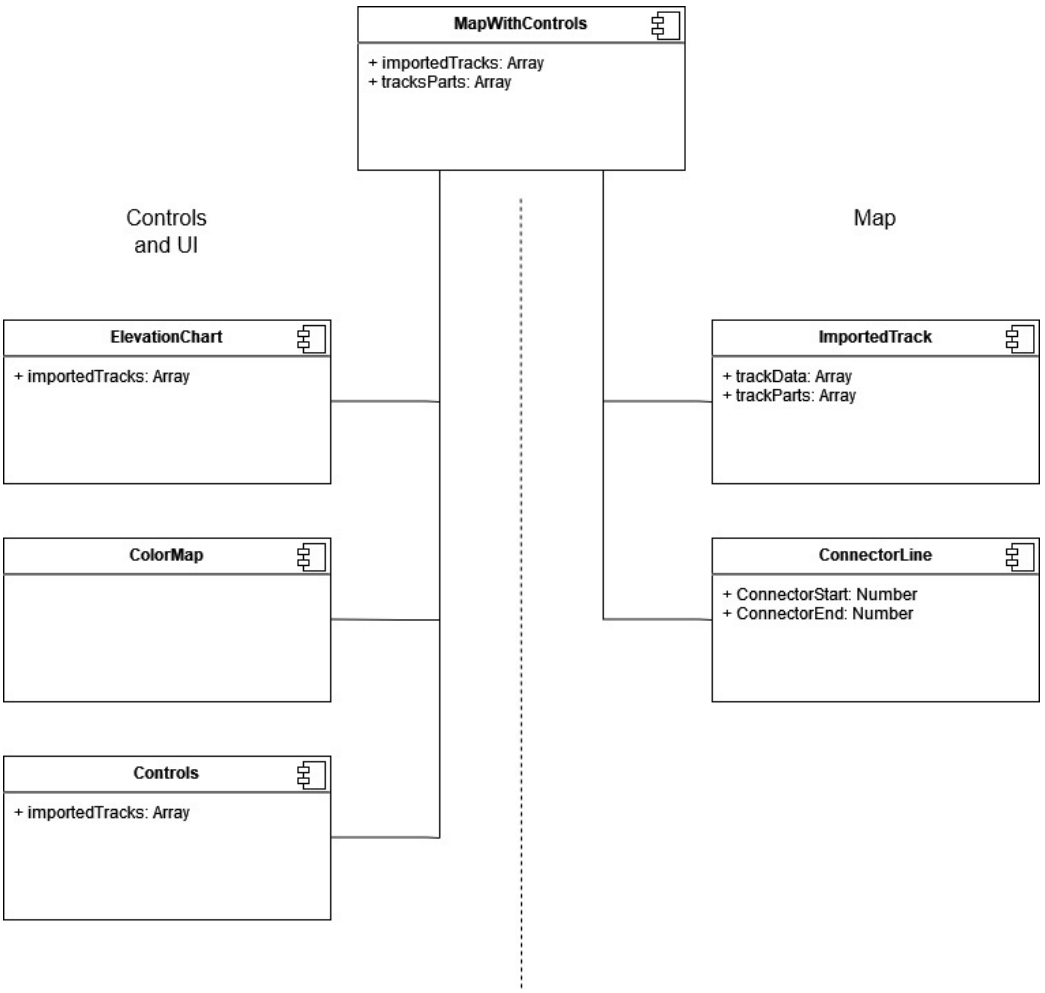

Abbildung 9: Komponenten und ihre wichtigsten Attribute

MapWithControls ist der gemeinsame Parent aller Komponenten (s. Abbildung 9). Er gibt das Layout an, erstellt die benötigten Komponenten und verwaltet die gemeinsamen Daten. Seine Children können in zwei Bereiche unterteilt werden. Karten-bezogene Komponenten sind ImportedTrack und ConnectorLine, sie geben Objekte zurück welche auf der Karte dargestellt werden. Für die Benutzeroberfläche und Steuerung sind ElevationChart, ColorMap und Controls zuständig.

1. **ImportedTrack** kümmert sich um die Darstellung eines Tracks und den dazugehörigen Funktionen. Die wichtigsten Funktionen sind die Tracktyp-Erkennung und Wahl eines passenden Tracksegments unter Berücksichtigung der Benutzerpräferenzen.

2. **ConnectorLine** visualisiert eine Verbindungslinie zwischen den importierten Tracks.

3. **Controls** stellt die Steuerung und Eingabefelder für den Benutzer dar.

4. **ColorMap** ist für die Darstellung der Informationen zu Farben zuständig.

5. **ElevationChart** visualisiert die Höhenmeter.

4.2 Ablaufdiagramm

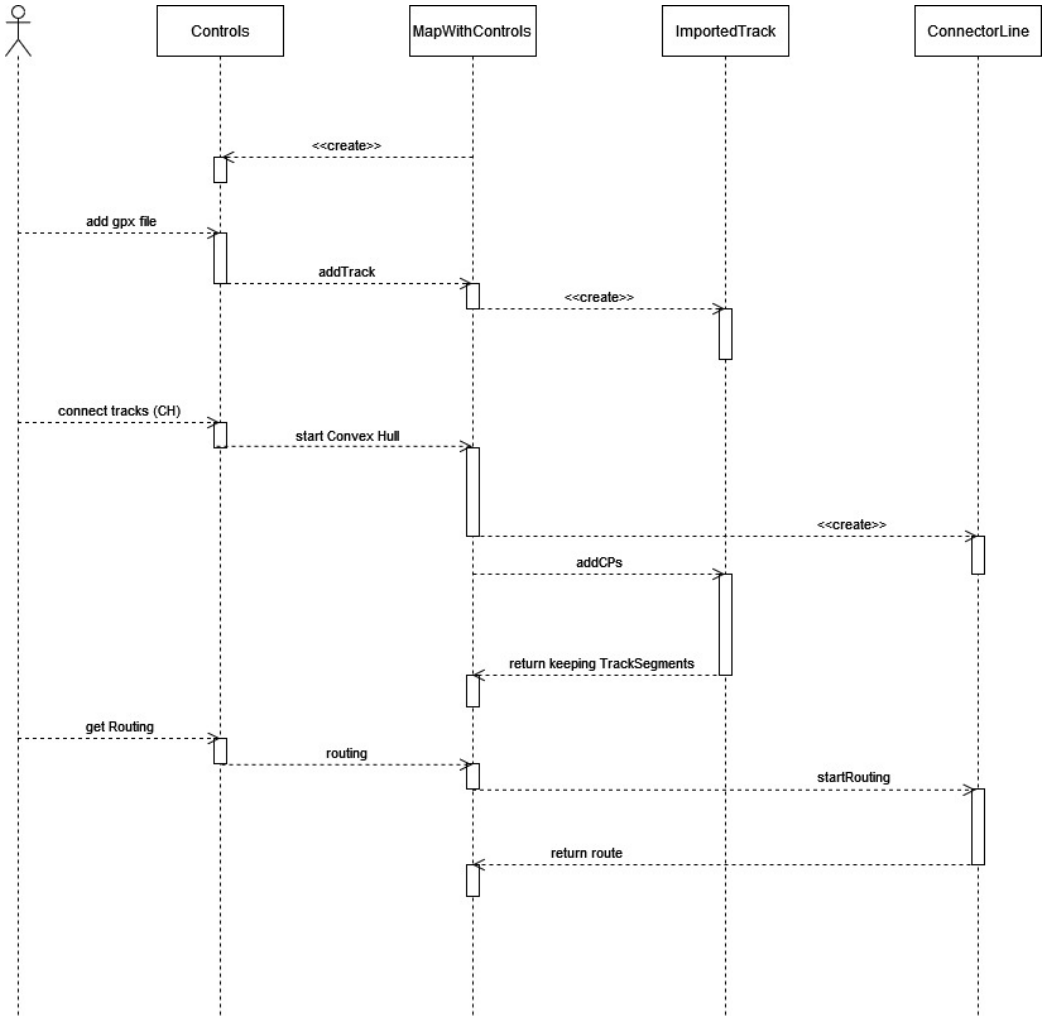

Abbildung 10: Vereinfachtes Ablaufdiagramm einer Trackkombination

MapWithControls ist die zentrale Steuerungskomponente. Sie speichert und verwaltet die Daten von globalem Interesse und reicht sie an die entsprechenden Komponenten weiter. In Abbildung 10 ist ein vereinfachter Ablauf innerhalb des Systems dargestellt. Die Kommunikation von MapWithControls mit den Karten-bezogenen Komponenten ist bidirektional, da hier die Tracktyperkennung, Routing und Anpassungen durch Präferenzen stattfin-

den. Diese Daten sind nicht nur für den dazugehörigen Track relevant und müssen deshalb auch an MapsWithControls übermittelt werden. Komponenten welche für die Steuerung und Darstellung von Benutzeroberflächen zuständig sind, erhalten auch Daten von MapWithControls. In der Regel geht die Kommunikation aber eher in Richtung der Steuerungskomponente.

4.3 Daten

Die wichtigsten Daten bei dieser Anwendung werden die Geodaten zu den Tracks sein. Das System erhält diese in Form von einer gpx-Datei. Diese muss entsprechend geparst werden, um die Daten in einer lesbaren Form für Javascript zu erhalten. Ein Großteil des Systems wird sich mit der Manipulation dieser Daten beschäftigen. Deshalb bietet es sich an die einzelnen Datensätze als Arrays in einem Array zu speichern. Javascript bietet umfangreiche Funktionen für die Bearbeitung von Arrays an.

Folgende Form bietet sich für einen einzelnen Geodatensatz an:

```
[Latitude, Longitude, Informationsobjekt]
```

Dass Latitude und Longitude auf Index 0 und 1 liegen, ist bei den meisten Bibliotheken die sich mit Koordinaten beschäftigen Standard. Dies macht unsere Daten also für viele Bibliotheken kompatibel. In das Informationsobjekt kommen zusätzliche Daten zu den Koordinaten, wie zum Beispiel Höhenmeter. Der Vorteil des Objekts ist es, dass Elemente ohne große Veränderung hinzugefügt und entfernt werden können. So ist es einfacher Formatierungsänderungen beim Prototyp umzusetzen. Die einzelnen Datensätze werden dann in einem Array vereint:

```
[[Latitude, Longitude, Objekt],[Latitude, Longitude, Objekt],...]
```

Ein Beispiel einer Koordinate mit dazugehörigen Höhenmeter:

```
[[47.40508, 12.83162, {alt: 753.90966}],...]
```

5 Implementierung

5.1 Suche und Wahl der Technologien

In diesem Abschnitt wird begründet warum und welche Technologien verwendet werden. Der Prototyp soll als Webanwendung im Browser laufen, da dies eine einfache und unkomplizierte Verwendung sicherstellt. Allgemein gilt, dass das System auf opensource Bibliotheken setzt um eine gewisse Stabilität und Anpassungsfähigkeit zu gewährleisten.

5.1.1 Framework

Als Framework für den Prototyp wird Next.js (Vercel, 2022) verwendet. Es benutzt React (Meta Platforms, 2022b) für das State Management und Rendering. Dies war ein wichtiger Entscheidungspunkt, denn ich habe damit bereits Erfahrung und React ist eine der größten Bibliotheken in diesem Bereich (Potter, 2022). Um eine umfangreiche Applikation zu erstellen, benötigt es weitere Funktionen wie zum Beispiel URL Routing oder API Routen, welche von NextJS mitgeliefert werden. Die Alternative, Create React App (CRA) (Meta Platforms, 2022a), hat dies nicht out of the box. Aus diesem Grund bietet sich NextJS für ein schnelles Prototyping an.

5.1.2 Karte

Für die Darstellung der Strecken wird eine Bibliothek für interaktive Karten benötigt. Die Bibliothek sollte stabil und performant sein, gleichzeitig aber möglichst wenigen Limitierungen unterliegen. All dies spricht für die größte opensource Bibliothek für Karten, Leaflet (Agafonkin, 2022). Leaflet bietet einen guten Support für Browser, hohe Performance und umfangreiche Funktionalität im Bereich Interaktion, Anpassung und Layers. Alternativen wie die Google Maps API (Google, 2022a) bieten im Kern die gleiche Funktionalität wie Leaflet. Allerdings sind diese nicht komplett kostenlos und bieten weniger Anpassungsmöglichkeiten, da es sich um proprietäre Anwendungen handelt.

5.1.3 Routing

Für das Routing verwendet der Prototyp OpenRouteService (openrouteservice, 2022), eine crowed sourced und opensource routing API. Eine wichtige

Funktionalität ist die Streckenfindung mit unterschiedlichen Fortbewegungsmitteln. ORS bietet neben verschiedenen Kraftfahrzeugprofilen auch mehrere Profile für Radfahrer und Wanderer an. Eine weitere gute Routing Engine ist Open Source Routing Machine (OSRM, 2022). OSRM bietet jedoch weniger umfangreiche Fortbewegunsmittel für das Routing an und muss selbst gehosted werden. ORS kann ebenfalls selbst gehosted werden, bietet aber zusätzlich noch eine kostenlose Schnittstelle zu einem bereits gehosteten Service an. Dies entfernt zusätzlichen Overhead für die Nutzung des Prototyps.

5.1.4 Geodaten

Bei einem System wie diesem, wird man nicht lange ohne Geodatenanalyse auskommen. Um grundlegende Berechnungen nicht erneut zu programmieren, bietet sich hierfür die Bibliothek Turf.js (Mapbox, 2022) an. Diese bietet neben Hilfsfunktionen wie das Erstellen eines GeoJSON Objekts von einem Array aus Positionen bis hin zu komplexeren Aufgaben wie die Berechnung von Schnittpunkten oder Überlappungen (s. Listing 1).

Listing 1: Beispiel zu Turf.js

```
1 // Erstellung eines GeoJSON Objekts aus einem Array
2 // mit Positionen
3 let line1 = turf.lineString([[126, -11], [129, -21]])
    ;
4 let line2 = turf.lineString([[123, -18], [131, -14]])
    ;
5
6 // Berechnung der Schnittpunkte beider Linien
7 // Das Ergebnis ist ein GeoJSON Objekts des Typs
8 // FeatureCollection und enthaelt alle Schnittpunkte
9 let intersects = turf.lineIntersect(line1, line2);
```

5.2 Karte und dazugehörige Funktionen

5.2.1 Tiles

Der Benutzer kann zwischen zwei verschiedenen Tilesets (auch Kachelset) wechseln. Zum einen gibt es die vektor-basierten Tiles, welche eine klare Übersicht über Gebiete und Wege bieten. Als weitere Option kann der Benutzer die Google Satellit bild-basierten Tiles auswählen. Die Satellitenbilder

ermöglichen ein besseres Verständnis des Terrains und der Beschaffung von Wegen.

Mit Leaflet kann sehr einfach ein Tileset durch Layers hinzugefügt und verwaltet werden (s. Listing 2). Bei den entsprechenden Domains steht `{z}` für den Zoom, `{x}` und `{y}` für die Koordinaten. Mit `{s}` können verschiedene Subdomains hinzugefügt werden, um die Limitierung von Browsern für die Anzahl von parallelen Anfragen an eine Domain zu umgehen.

Listing 2: Einbindung von Tilesets in Leaflet

```
<LayersControl position="bottomleft">
  <BaseLayer checked name="OpenStreetMap">
    <TileLayer
    url="https://{s}.tile.openstreetmap.org/{z}/{x}/{
       y}.png"
    attribution='&copy;
    <a href="https://www.openstreetmap.org/copyright"
       >
    OpenStreetMap
    </a> contributors'
    />
  </BaseLayer>
  <BaseLayer name="Google Satellite">
    <TileLayer
    url="https://{s}.google.com/vt/lyrs=s&x={x}&y={y
       }&z={z}"
    maxZoom={20}
    subdomains={["mt1", "mt2", "mt3"]}
    />
  </BaseLayer>
</LayersControl>
```

5.2.2 Darstellung von Vektoren auf der Karte

Die Tracks werden auf der Karte durch das Erstellen einer Polyline Leaflet-Komponente angezeigt (s. Listing 3). Wichtig ist, dass die Polyline nur auf der Karte angezeigt wird, wenn das Element innerhalb des MapContainer-Elements von Leaflet steht. Dieses Element ist in MapWithControls definiert. Eine Child-Komponente gibt also nur das Polyline Element zurück, welches dann von MapWithControls innerhalb des MapContainer-Elements platziert

wird (s. Listing 4).

Listing 3: Erstellung eines Polyline Elements

```
<Polyline
  positions={
    ([45.51, -122.68],
    [37.77, -122.43],
    [34.04, -118.2])
  }
/>
```

Listing 4: Einbindung von Polylines in den MapContainer

```
<MapContainer>
  // Polyline Element als Variable gespeichert
  {polyLineElement}
  // Polyline Element erstellt in dieser Komponente
  <Polyline
    positions={
      ([45.51, -122.68],
      [37.77, -122.43],
      [34.04, -118.2])
    }
  />
</MapContainer>
```

5.3 Daten

Dieses Kapitel soll den Ablauf und die Verarbeitung der wichtigsten Daten, den Koordinaten der Tracks, aufzeigen (s. Abbildung 11). Im ersten Schritt erhält das System eine vom Benutzer importierte gpx-Datei. Damit die Daten vernünftig verarbeitet werden können, müssen sie in einem für Javascript lesbaren Format vorliegen. Dieser Schritt wird von einem Parser übernommen, welcher die Daten als GeoJSON Objekt darstellt. Aus diesem Objekt werden dann alle benötigten Daten extrahiert und als Array, wie in Kapitel 4.3 beschrieben, gespeichert. Bevor mit den Daten gearbeitet werden kann, ist ein letzter Schritt nötig. Wie in Kapitel 3.1 beschrieben, gibt es verschiedene Tracktypen und beim Typ `Rundstrecke mit Hin- und Rückweg` spielen nicht alle Koordinaten bei der Kombination von Tracks eine Rolle. Aus diesem

Grund wird überprüft, um welchen Typ es sich handelt und dementsprechend ein neues Array erstellt, welches aus bis zu drei Elementen bestehen kann:

```
[Hauptteil, Hinweg, Rückweg]
```

Nur `Rundstrecke mit Hin- und Rückweg` hat drei Elemente, da die anderen beiden Typen nur aus einem Hauptteil bestehen. Das als Hauptteil identifizierte Koordinaten-Array dient nun als Grundlage für die Kombination der Tracks.

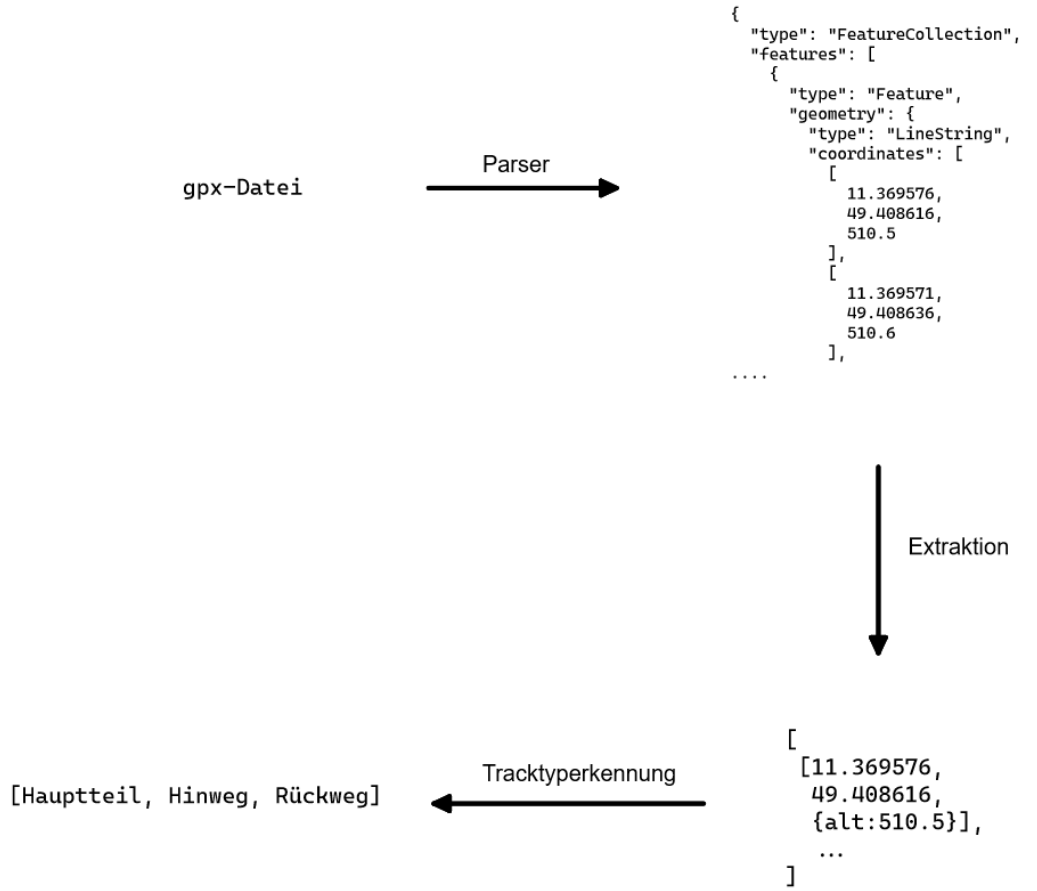

Abbildung 11: Verarbeitung der Trackdaten

Dieser Ablauf findet über mehrere Komponenten statt. Der Import und die Konvertierung zu einem GeoJSON Objekt wird vom Komponenten Con-

trols gestartet. Das GeoJSON Objekt wird dann MapWithControls übergeben, welcher die Daten zu einem Array verarbeitet und anschließend die Komponente ImportedTrack erstellt und dieser das Array übergibt. Dort findet bei der Erstellung die Tracktyp-Erkennung statt. Das daraus entstehende Array wird an MapWithControls übergeben, welcher die Daten für die Kombination verwendet.

5.4 Algorithmen

5.4.1 Erkennung der Tracktypen

Für die Erkennung der drei Tracktypen (s. Abbildung 7, Seite 10), muss das Koordinaten-Array genauer betrachtet werden. Jeder Typ hat Eigenschaften welche zur Identifikation führen. Eigenschaften auf die geachtet werden sind gleiche, beziehungsweise sehr nahe liegende Koordinatenpunkte und ähnlich verlaufende Streckenabschnitte, also Überlappungen und Überschneidungen.

In Abbildung 12 ist das Array vom Typ `Strecke` bzw. `Rundstrecke` dargestellt. Das Merkmal für die Unterscheidung ist der Start- und Endpunkt (markiert mit A und B). Bei der `Rundstrecke` liegen beide Punkte sehr nah beieinander oder stimmen überein, bei der `Strecke` ist dies nicht der Fall.

Beim Typ `Rundstrecke mit Hin- und Rückweg` (s. Abbildung 13) ist vor dem Startpunkt und nach dem Endpunkt ein weiterer Trackabschnitt (blau markiert). Diese sehr ähnlich verlaufenden Abschnitte sind das identifizierende Merkmal dieses Typs.

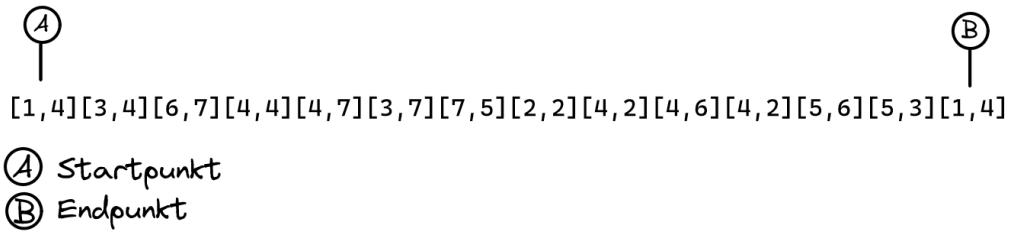

Abbildung 12: Vereinfachte Darstellung von Typ Strecke bzw. Rundstrecke

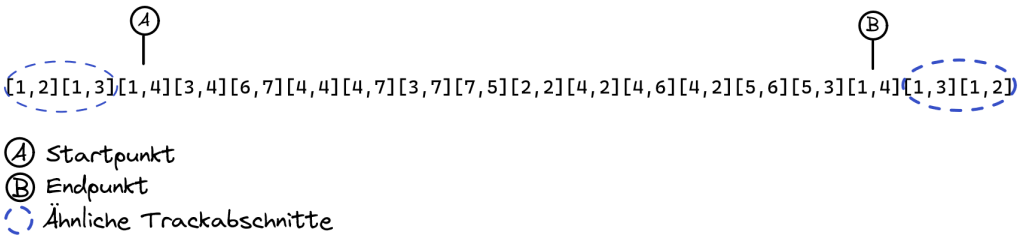

Abbildung 13: Vereinfachte Darstellung von Typ Rundstrecke mit Hin- und Rückweg

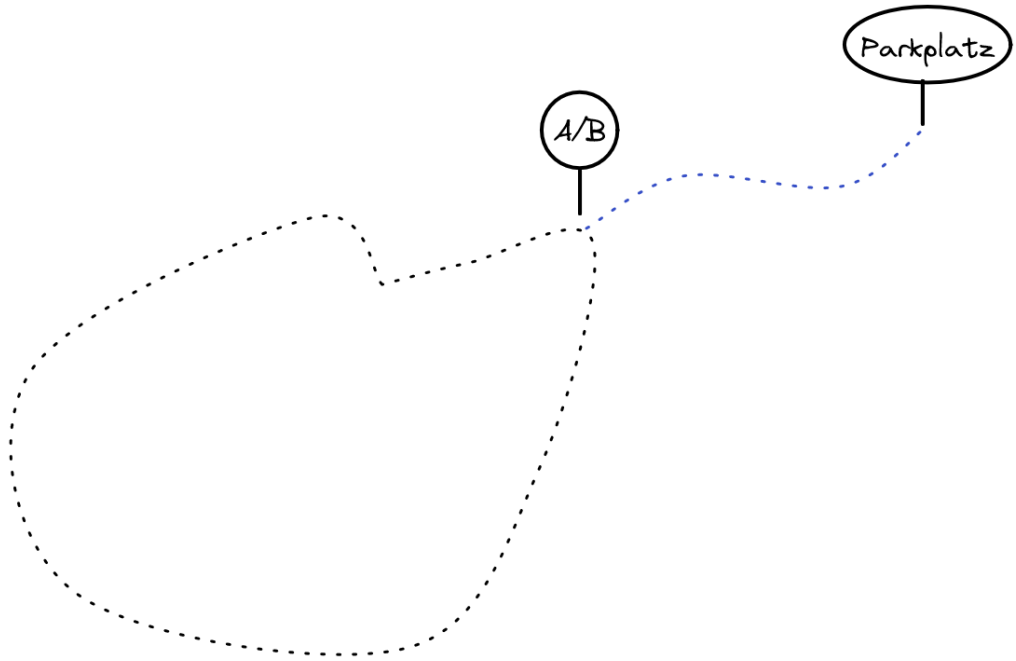

Abbildung 14: Darstellung von Typ Rundstrecke mit Hin- und Rückweg als Track

In Abbildung 14 wird der Typ `Rundstrecke mit Hin- und Rückweg` für das bessere Verständnis als Track dargestellt. Wie bereits zuvor erwähnt, soll der Hin- und Rückweg für die Kombination ignoriert werden, da er nicht Teil des eigentlichen Tracks ist. Dieser Abschnitt könnte der Hin- und Rückweg zu einem Parkplatz oder dem Stadtzentrum sein.

Für die Überprüfung, ob es sich um eine `Strecke` handelt, muss nur die erste und letzte Koordinate auf Gleichheit überprüft werden. Sind beide Koordinaten unterschiedlich, handelt es sich um eine Linie. Da bei mit GPS aufgezeichneten Koordinaten immer mit Abweichungen zu rechnen ist, wird zusätzlich die Entfernung zwischen den beiden Punkte angeschaut. Ist der Abstand größer als 100 Meter wird der Track als Linie erkannt (s. Listing 5).

Listing 5: Überprüfung auf Typ Linie

```
// Betrachtung des Start- und Endpunkts.
// Der erste Index steht fuer Position der Koordinate im Array
// Der zweite Index steht fuer Latitude (0) oder Longitude (1)
if (
coordinates[0][0] !== coordinates[length - 1][0] &&
coordinates[0][1] !== coordinates[length - 1][1]
) {
  // Berechnung der Distanz zwischen Start und Ende
  // falls die Koordinaten nicht uebereinstimmen
  let distanceStartToEnd =
  distanceInKmBetweenEarthCoordinates(
  coordinates[0][0],
  coordinates[0][1],
  coordinates[length - 1][0],
  coordinates[length - 1][1]
  );
  // 0,1 Kilometer ist die Schwelle fuer die
  // Identifizierung als Linie
  if (distanceStartToEnd > 0.1) {
    type = "LINE";
    return type;
  }
}
```

Handelt es sich nicht um Typ `Strecke` dann wissen wir, dass es `Rundstrecke` oder `Rundstrecke mit Hin- und Rückweg` sein muss. Das System überprüft nun also auf `Rundstrecke mit Hin- und Rückweg`. Sollte es sich auch nicht um diesen Typ handeln, dann ist es Typ `Rundstrecke`.

Zuerst wird für die Überprüfung nach einer doppelt vorkommenden Koordinate durchgeführt. Die Untersuchung auf Gleichheit findet in der Funktion

checkForSameCoordinate statt, mit der Bedingung, dass die beiden Koordinaten einen gewissen Indexabstand zueinander haben müssen. Der Abstand ist wichtig, da es durch GPS Aufzeichnungen oder angepasster Formatierung der gpx-Datei zu wiederholt vorkommenden Koordinaten mit keinem oder geringen Indexabstand kommen kann. Zum Beispiel könnte eine Koordinate doppelt hintereinander stehen, wobei die Zweite für ein bestimmtes Programm einen Marker darstellt. Der Prototyp erkennt dies jedoch einfach nur als weitere Koordinate.

Werden zwei ähnliche Koordinaten gefunden, wird der Trackabschnitt vor der ersten Koordinate und der Trackabschnitt nach der zweiten Koordinate betrachtet. Durch Überprüfung auf Überlappung mit Toleranz, erhält das System die Anzahl der überlappenden Koordinaten. Entspricht die Anzahl der Hälfte der durchschnittlich vorkommenden Koordinaten, gilt es als ähnlich verlaufender Abschnitt und somit handelt es sich um Typ `Rundstrecke mit Hin- und Rückweg` (s. Listing 6).

Listing 6: Überprüfung auf Typ Rundstrecke mit Hin- und Rückweg

```
let checkedCoords = [];
coordinates.every((coord, index) => {
  // Ueberprueft ob die Koordinate einer vorherigen
  // Koordinate aehnelt
  let result = checkForSameCoordinate(
  checkedCoords,
  coord,
  index
  );

  if (result.found) {
    // Abfrage der Indizes der doppelten Koordinate
    let firstIndex = getFirstIndex(coordinates, coord
       );
    let lastIndex = getLastIndex(coordinates, coord);

    // Nur wenn es sich nicht um den ersten und
    // letzten Index handelt kann es dieser Typ sein
    if (
    lastIndex < coordinates.length - 1 &&
    result.indeces[0] > 1
    ) {
```

```
22        let hinweg = collection.slice(
23        0,
24        firstIndex + 1
25        );
26        let rueckweg = collection.slice(
27        lastIndex,
28        collection.length - 1
29        );
30
31        // Ueberpueft wie sehr sich der Streckenteil
32        // vor der ersten Koordinate und nach der
33        // zweiten Koordinate aehnlich sind
34        let average =
35        (firstPart.length + secondPart.length) / 2;
36
37        // Formatierung der Koordinaten zu GeoJSON
38        let hinwegGeoJSON = turf.lineString(hinweg);
39        let rueckwegGeoJSON = turf.lineString(rueckweg)
              ;
40
41        // Berechnung der ueberlappenden Koordinaten
42        // mit einer Toleranz von 100 Metern
43        let intersections = turf.lineOverlap(
44        hinwegGeoJSON,
45        rueckwegGeoJSON,
46        {
47          tolerance: 0.1,
48        }
49        );
50
51        // Addiert die Anzahl der ueberlappenden
              Koordinaten
52        let acc = intersections.features.reduce(
53        (prev, curr) => {
54          return (
55          prev + curr.geometry.coordinates.length
56          );
57        }, 0);
58
59        // Erkennt "Rundstrecke mit Hin- und Rueckweg"
```

```
60        // wenn mehr als die Haelfte der
61        // Koordinaten der Abschnitte ueberlappen
62        if (acc >= average / 2) {
63          type = "RUNDSTRECKEmitHINRUECK";
64        }
```

5.4.2 Verbindung der Tracks

Für die erste Verbindung von zwei Tracks gibt es zwei verschiedene Ansätze.

Der erste Ansatz ist der Convex Hull Algorithmus. Dieser nimmt alle Koordinaten der ursprünglichen Tracks (s. Abbildung 15) und wendet auf diese den Algorithmus an. Das Ergebnis ist eine konvexe Hülle um die gesamten Punkte (s. Abbildung 16). Aus der konvexen Hülle werden dann die Verbindungslinien zwischen den Tracks extrahiert und die dazugehörigen Koordinaten stellen den ersten Verbindungsansatz dar, welcher bei der konvexen Hülle durch eine rote Linie hervorgehoben wird.

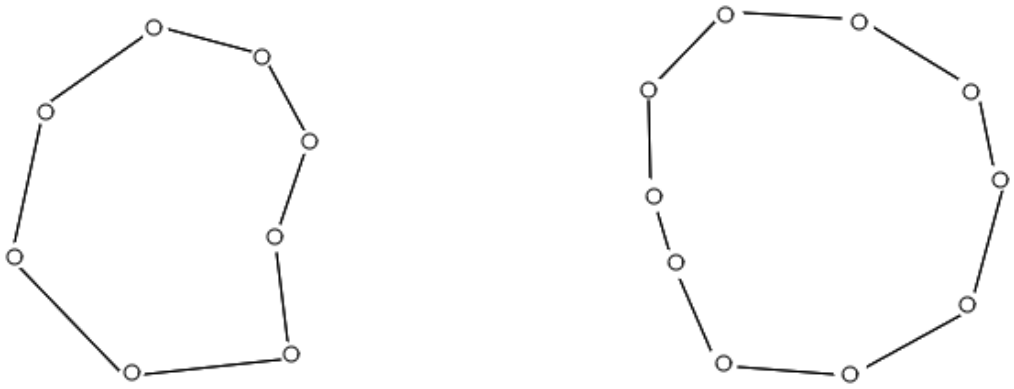

Abbildung 15: Darstellung von zwei Tracks

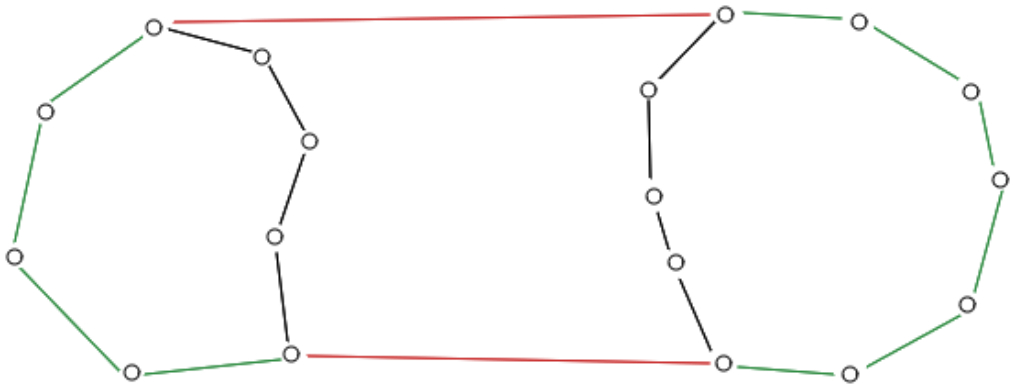

Abbildung 16: Zwei Tracks mit konvexer Hülle und Verbindungslinien (rot)

Der Convex Hull Algorithmus wird von einer Bibliothek übernommen und das System extrahiert aus dem Ergebnis die Verbindungen (s. Listing 7). Für das Extrahieren geht eine for-Schleife über das Array aller Koordinaten der konvexen Hülle und sieht sich immer aufeinanderfolgende Punkte an. Jede Koordinate hat zusätzlich die Nummer ihres Tracks gespeichert. Haben also zwei aufeinanderfolgende Koordinaten eine unterschiedliche Tracknummer, dann handelt es sich um eine Verbindung zwischen den Tracks.

Listing 7: Extraktion von Verbindugen in der konvexen Hülle

```
1  export function getConnectorsFromHull(hull) {
2    let connectors = [];
3    // es werden immer aufeinaderfolgende
       Koordinatenpaare
4    // ueberpueft
5    for (let i = 0; i < hull.length - 2; i++) {
6      // Unterscheidet sich die TrackNummer ist es eine
7      // Verbindung
8      if (hull[i][2].trackNummer !==
9        hull[i + 1][2].trackNummer)
10     {
11       let coord1 = hull[i];
12       let coord2 = hull[i + 1];
13
14       connectors.push([coord1, coord2]});
15     }
16   }
```

```
17    // Ueberpruefung der ersten und letzten Koordinaten
          als Paar
18    if (hull[0][2].trackNummer !==
19      hull[hull.length - 1][2].TrackNummer)
20    {
21      let coord1 = hull[i];
22      let coord2 = hull[i + 1];
23
24      connectors.push([coord1, coord2]});
25    }
26    return connectors;
27  }
```

Der zweite Ansatz für eine Verbindung ist es, den nähesten Punkt zwischen den Tracks zu finden (s. Listing 8), um dort an den umliegenden Koordinaten die Verbindungslinien zu setzen. Um die Suche optimiert durchzuführen, wird aus einem der Tracks ein k-d-Baum erstellt. Es handelt sich dabei um einen Suchbaum, welcher verhindert, dass für jede Koordinate aus Track 2 alle Koordinaten aus Track 1 überprüft werden müssen. Dies führt bei großen Tracks zu extrem verringerten Suchzeiten.

Listing 8: Finden der am naheliegensten Punkte

```
export function checkClosest(track1, track2) {
  // Formatierung der Track Arrays zu Objekten, da
  // der KD-Tree Objekte benoetigt
  var track1Object = track1.map((point, index) =>
  Object.assign(
  {},
  { lat: point[0], long: point[1], i: index })
  );

  var track2Object = track2.map((point, index) =>
  Object.assign(
  {},
  { lat: point[0], long: point[1], i: index })
  );

  // Erstellung eines k-d Trees fuer track1
  const kdTree = new S3.kdTree(track1Object, distance
     , [
  "lat",
  "long",
  ]);

  let nearest = [];

  // Fuegt zu jeder track2 Koordinate die naeheste
     Koordinate
  // aus track1 zur Variablen nearest
  track2Object.forEach((object) => {
    nearest.push([object, kdTree.nearest(object, 1)])
       ;
  });
```

```
29
30    // Sortieren der nearest Elemente, damit das am
31    // naehesten liegende Paar auf Index 0 steht
32    nearest.sort(function (a, b) {
33      return a[1][0][1] - b[1][0][1];
34    });
35
36    // Auslesen der Indizes des Koordinatenpaars
37    let indeces = [nearest[0][1][0][0].i, nearest
          [0][0].i];
38
39    // Zurueckgeben des Indizes
40    return indeces;
41  }
```

Da eine Verbindungslinie nicht ausreichend ist, wählt das System umliegende Punkte für die Verbindungslinien. Damit sich die Linien nicht überschneiden, muss die Richtung des Tracks betrachtet werden um die Punkte sinnvoll zu verknüpfen (s. Abbildung 17). In Abbildung 18 ist das Problem nochmal an zwei Trackabschnitten dargestellt. Die umliegenden Indizes müssen also abhängig davon wie die Richtung der Tracks ist, unterschiedlich miteinander verknüpft werden. Ein vereinfachter Ablauf dieser Überprüfung ist in Listing 9 zu sehen.

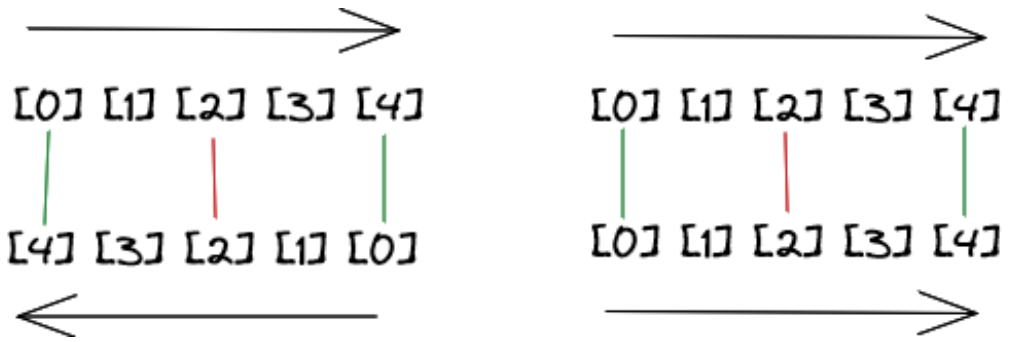

Abbildung 17: Richtungsproblem bei der Verbindung anhand vom nähesten Punkt

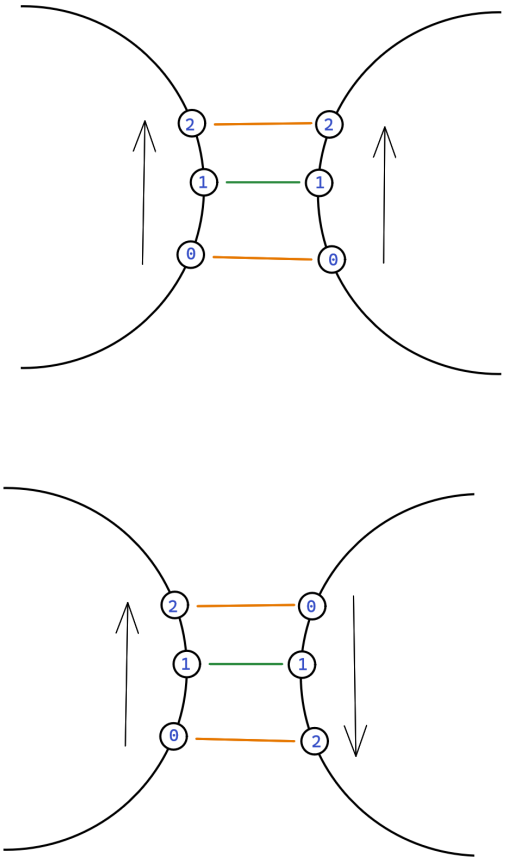

Abbildung 18: Weitere Darstellung des Richtungsproblems

Listing 9: Korrekte Verbindung der umliegenden Punkte des nähesten Punktes

```
1 |  // Ueberpruefung auf den naehesten Punkt zwischen
  |     den Tracks
2 |  let closest = checkClosest(track1[0],track2[1]);
3 |  let indexTrack1 = closest.point0Index;
4 |  let indexTrack2 = closest.point1Index;
5 |
```

```
6   // Formatierung der Koordinaten Arrays zu GeoJSON
7   let t1 = turf.lineString(importedTracks[0][0]);
8   let t2 = turf.lineString(importedTracks[1][0]);
9
10  // Ueberpruefung der Richtung fuer die korrekte
11  // Verbindung der Koordinaten
12  let clockwise1 = turf.booleanClockwise(t1);
13  let clockwise2 = turf.booleanClockwise(t2);
14
15  let con11, con12, con21, con22;
16
17  // Berechnung der Verbindungspunkte mit vom
18  // Benutzer gesetzen Abstand (closestRange).
19  // Abstand: Elemente im Array
20  if (clockwise1 === clockwise2) {
21    con11 = importedTracks[0][0][indexTrack1 +
        closestRange];
22    con12 = importedTracks[1][0][indexTrack2 -
        closestRange];
23    con21 = importedTracks[1][0][indexTrack2 +
        closestRange];
24    con22 = importedTracks[0][0][indexTrack1 -
        closestRange];
25  } else {
26    con11 = importedTracks[0][0][indexTrack1 +
        closestRange];
27    con12 = importedTracks[1][0][indexTrack2 +
        closestRange];
28    con21 = importedTracks[1][0][indexTrack2 -
        closestRange];
29    con22 = importedTracks[0][0][indexTrack1 -
        closestRange];
30  }
31
32  // Speicherung der Koordinaten fuer die
      Verbindungspunkte
33  addConnectors([
34  [con11, con12],
35  [con21, con22],
36  ]);
```

Der Convex Hull Algorithmus strebt eine möglichst breite Verbindung der Tracks an, also weit auseinanderliegende Verbindungspunkte. Der Ansatz mit den nähesten Punkten führt zu näherliegenden Verbindungspunkten und damit auch zu einer weniger ausgeglichenen Teilung des Tracks.

5.4.3 Segmentauswahl anhand Benutzerpräferenzen

Nachdem ein Track in zwei Segmente unterteilt wurde, muss das System entscheiden welches Segment übernommen wird. Es muss dabei die Benutzerpräferenzen zu Tracklänge und Höhenmeter berücksichtigen. Um die richtige Entscheidung zu treffen, wird zuerst überprüft, ob sich die Längen oder die Höhenmeter der Segmente stärker voneinander unterscheiden. Es werden also die Faktoren berechnet:

```
Faktor der Länge = Länge von Segment1 / Länge von Segment2
Faktor der Höhenmeter = Höhenmeter von Segment1 / Höhenmeter von Segment2
```

Die Präferenz mit der größeren Abweichung, also mit dem größeren Faktor zwischen den Segmenten, wird dann für die Segmentauswahl verwendet.

5.4.4 Erkennung der Trackkonstellationen

Für die Überprüfung der Trackkonstellation kann die Turf.js Bibliothek alle Berechnungen übernehmen. Sie bietet dafür die Funktionen *booleanIntersects* und *booleanWithin*. Ist das Ergebnis beider Funktionen falsch, dann handelt es sich um zwei Tracks die nebeneinander liegen (s. Listing 10).

Listing 10: Überprüfung der Trackkonstellation

```
1  // Standard auf NEBENEINANDER bis eine andere
2  // Ueberpruefung es widerlegt
3  let konstellationType = "NEBENEINANDER";
4
5  // Ueberpruefung auf Ueberschneidung der Tracks
6  let intersects = turf.booleanIntersects(track1,
       track2);
7  if (intersects) {
8    konstellationType = "UEBERSCHNEIDUNG";
9  } else {
10   // Wenn sich die Tracks nicht ueberschneiden dann
```

```
11    // ueberpruefen ob ein Track innerhalb des anderen
          liegt
12    if (
13      turf.booleanWithin(track1, track2) ||
14      turf.booleanWithin(track2, track1)
15    ) {
16      konstellationType = "ENTHALTEND";
17    }
18  }
19  // Ergebnis speichern
20  setKonstellation(konstellationType);
```

5.4.5 Auflösung von Überschneidungen der Tracks

Bei der Kombination von zwei Tracks die sich überschneiden, wird zuerst überprüft welcher der beiden Tracks der längere ist. Anschließend werden die äußersten Schnittpunkte auf dem längeren Track berechnet (s. Abbildung 20). An diesen Koordinaten werden beide Tracks zerschnitten. Das Segment welches sich innerhalb des kleineren Tracks befindet, wird verworfen. Die Benutzerpräferenzen bestimmen nun, welches der beiden Segmente, des kleineren Tracks, den fehlenden Abschnitt ersetzt (s. Abbildung 21).

Es kann dabei vorkommen, dass ein Schnittpunkt nicht als Element in dem Koordinaten-Array des Tracks existiert. Damit das System die Tracks an diesem Punkt zerschneiden kann, muss der Schnittpunkt als Element vorhanden sein. Es muss also überprüft werden, ob die Koordinate bereits im Koordinaten-Array vorkommt. Ist dies nicht der Fall, muss herausgefunden werden, an welchem Index der Schnittpunkt eingefügt werden muss. Dafür wird zuerst die am nähesten liegende Koordinate gesucht. Dies sagt jedoch noch nicht aus auf welcher Seite sie eingefügt werden muss. Um dies zu überprüfen, bilden wir eine Linie aus der am nähesten liegenden Koordinate und der Koordinate rechts davon (aus Index-Sichtweise) (s. Listing 11). Liegt der Schnittpunkt auf dieser Linie, liegt er rechts neben der Koordinate. Ist dies nicht der Fall, dann liegt sie auf der linken Seite. Visualisiert ist dieser Lösungsansatz in Abbildung 19.

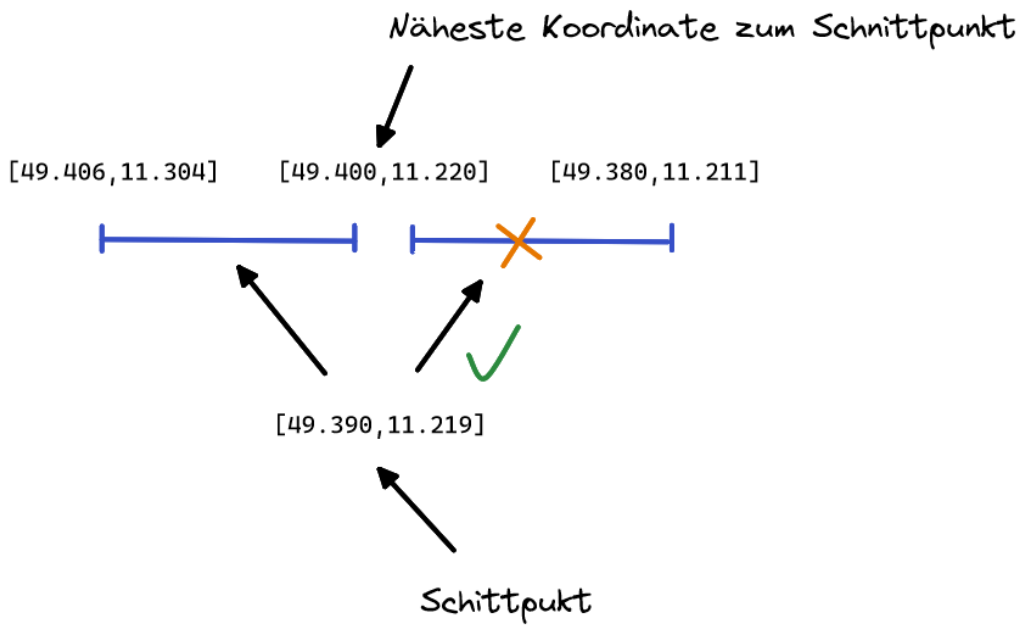

Abbildung 19: Einfügen einer neuen Koordinate an der richtigen Position

Listing 11: Vereinfachte Version des Einfügens einer neuen Koordinate

```
1  // Erstellen der Linie zwischen der naehesten
2  // Koordinate und der Koordinate auf der rechten
       Seite davon
3  let lineToTheRight = turf.lineString([
4  tracksPartsClone[trackNumber][0][closestPointIndex
       ],
5  tracksPartsClone[trackNumber][0][closestPointIndex
       + 1],
6  ]);
7  // Ueberpruefe ob Schnittpunkt auf Linie liegt
8  if (turf.booleanPointOnLine(intersectionCoord,
9      lineToTheRight))
10 {
11   // Punkt liegt rechts von der naehesten
         Koordinate
12 } else {
13   // Punkt liegt links von der naehesten Koordinate
14 }
```

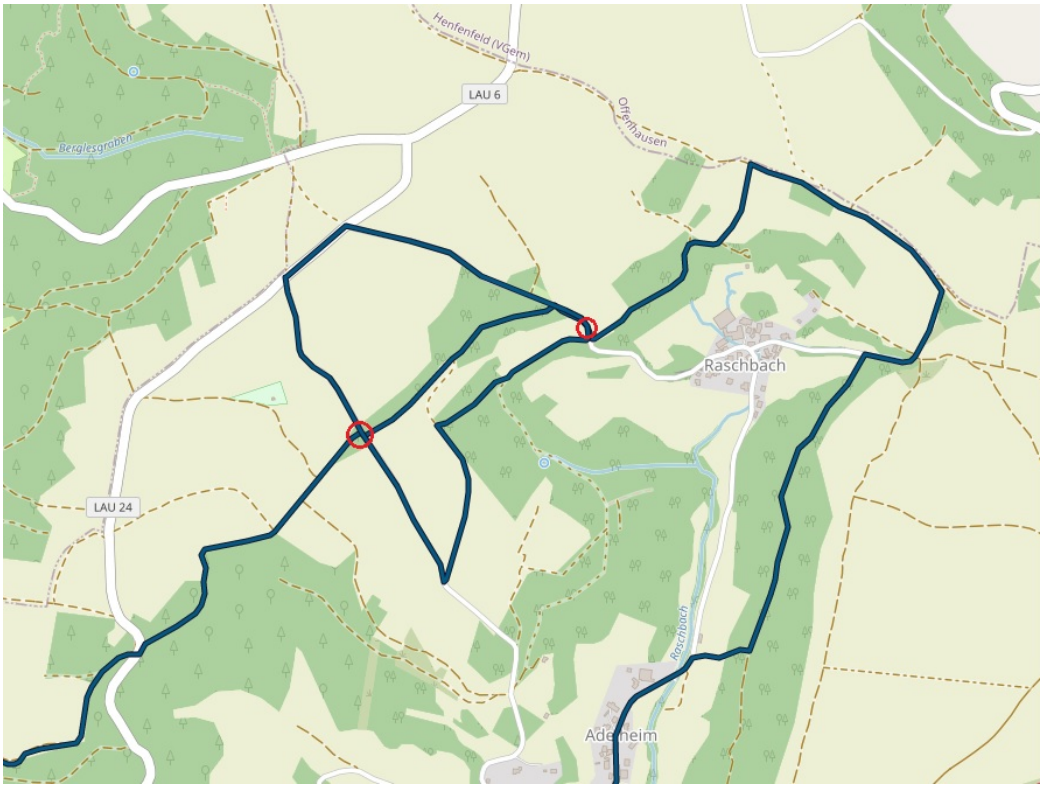

Abbildung 20: Darstellung der äußeren Schnittpunkte bei einer Überschneidung

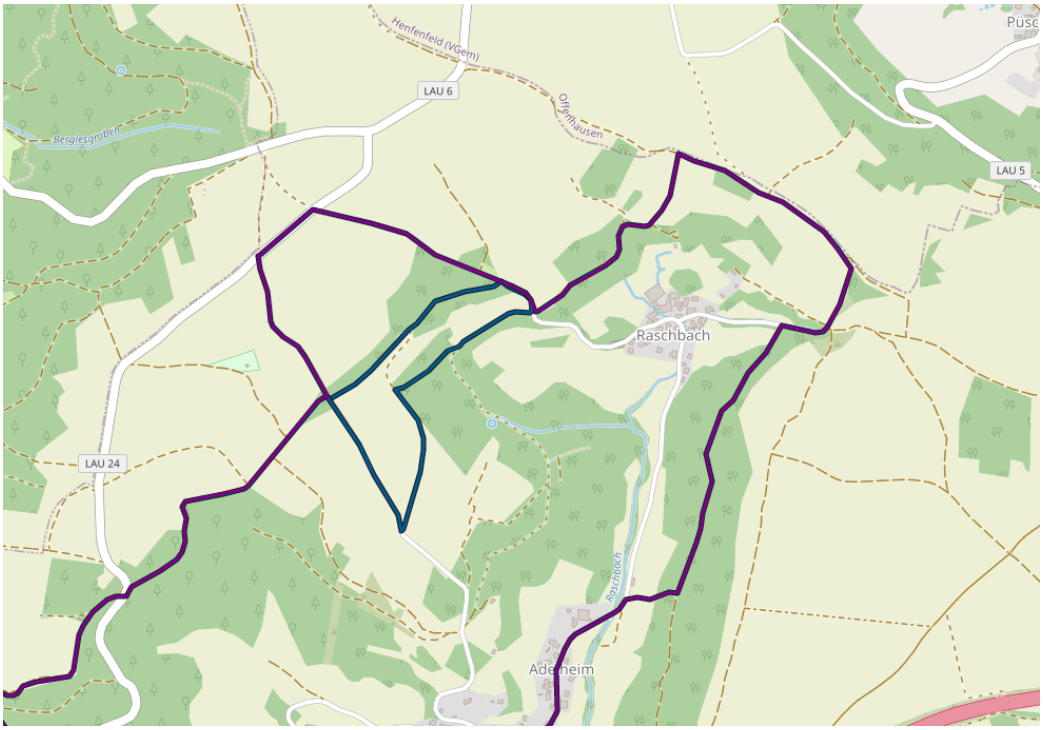

Abbildung 21: Ergebnis einer Kombination von zwei sich überschneidenden Tracks

5.4.6 Berücksichtigung der Points of Interest

Ein Point of Interest ist ein Wegpunkt eines Tracks, welchen der Benutzer auf jeden Fall besuchen möchte. Dafür überprüft das System, ob alle POIs ein Teil des zu übernehmenden Segments sind. Ist dies nicht der Fall, dann muss ein Connectorpoint verschoben werden. Das System ermittelt dafür den nähesten Connectorpoint und verschiebt diesen auf den Punkt des zu besuchenden Wegpunktes.

5.4.7 Routing

Für das Routing wird, wie zuvor erwähnt, Openrouteservice verwendet. Der Benutzer kann dabei über das Profil entscheiden, ob er das Routing für Wanderungen oder Fahrradtouren optimieren möchte. Die genutzen ORS Profile dafür sind derzeit *foot-hiking* und *cycling-mountain*. Zusätzlich hat der Benutzer auch die Wahl beim Routing die gewählten Segmente zu vermeiden (s.

Abbildung 22 und Abbildung 23). Es werden dafür zwei verschiedene Endpunkte der API, mit einem HTTP POST Request, angesprochen (s. Listing 12). Die zu vermeidenden Bereiche müssen als Polygon mitgesendet werden. Die Segmente, welche in diesem Fall vermieden werden sollen, sind jedoch einfache Linien. Um diese als Polygon darzustellen, muss das Segment mit umgekehrter Reihenfolge an das ursprüngliche Segment angefügt werden. Danach kann mit Turf.js ein GeoJSON Polygon erstellt werden.

Allgemein ist auch zu beachten, dass die API die Koordinaten in umgekehrter Reihenfolge benötigt, also `[long,lat]` statt `[lat,long]`.

Listing 12: POST Anfrage an die Routing API

```
export function getRouting(startCoord, endCoord,
   routingProfile) {
  return fetch(
  `https://api.openrouteservice.org/v2/directions/${
     routingProfile}/geojson`,
  {
    method: "POST",
    headers: {
      "Content-Type": "application/json;
         charset=utf-8",
      Accept:
      "application/json, application/geo+json,
         application/gpx+xml, img/png; charset=utf-8"
         ,
      Authorization: apiKey,
    },
    body: JSON.stringify({
      coordinates: [startCoord, endCoord],
      instructions: false,
      elevation: true,
    }),
  }
  )
  .then((response) => response.json())
  // ... Zurückgeben der Daten
}
```

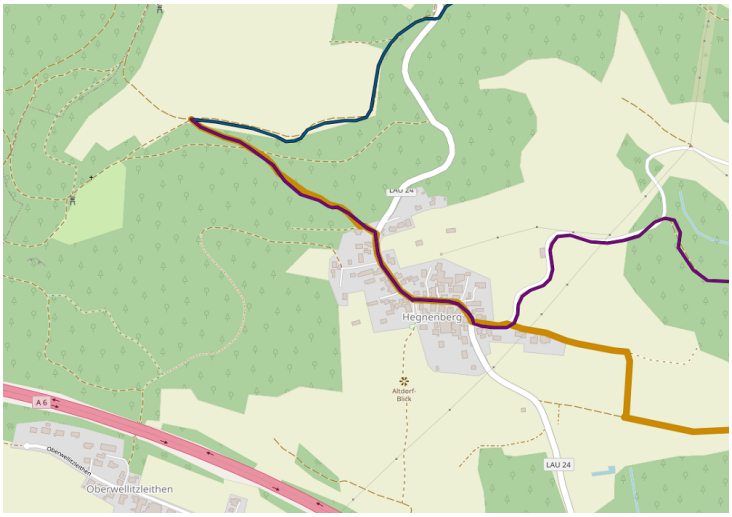

Abbildung 22: Routing ohne Segmentvermeidung (die berechnete Route ist in orange dargestellt)

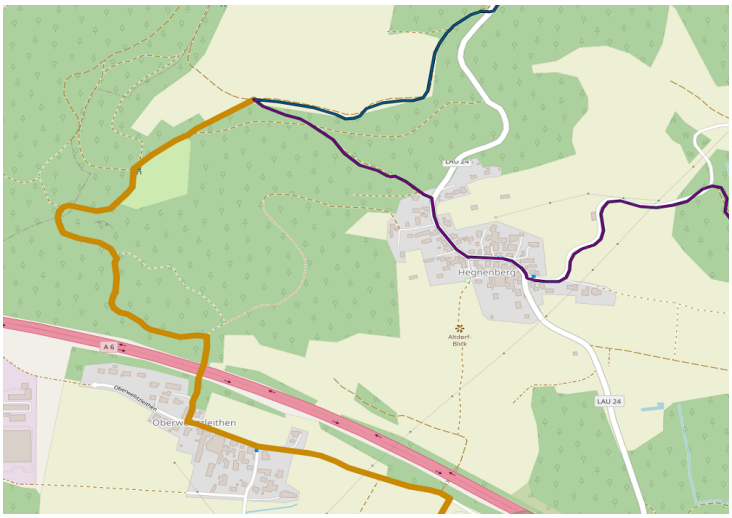

Abbildung 23: Routing mit Segmentvermeidung (die berechnete Route ist in orange dargestellt)

5.4.8 Darstellung der Höhenmeter

Für die Darstellung der Höhenmeter wird die Bibliothek Recharts verwendet. Zusätztlich zu einem Tooltip, soll auf der Karte der entsprechende Punkt hervorgehoben werden. Um Performanceprobleme zu besteitigen, zeichnet die Komponente direkt auf die Karte (s. Listing 13).

Listing 13: Direktes zeichnen auf die Karte

```
1  function drawRect(center) {
2    // Loeschen der vorherigen Markeriung
3    elevationLayer.current.clearLayers();
4
5    // Erstellung der Markierung (Rectangle)
6    const bounds = L.latLng(center).toBounds(100);
7    const square = new L.Rectangle(bounds);
8
9    // Markierung wird zur LayerGroup hinzugefuegt
10   elevationLayer.current.addLayer(square);
11
12   // LayerGroup wird zur Karte hinzugefuegt
13   props.map.addLayer(elevationLayer.current);
14 }
```

Für die Vereinfachung der Verwaltung von Markierungen werden *LayerGroups* verwendet. Damit ist es in Leaflet möglich Elemente zu gruppieren und allgemein anzusprechen. In Listing 14 erstellen wir dafür die Variable *elevationLayer*, welche für die Lebenszeit der Komponente auf unsere *LayerGroup* verweist. Bei Erstellung der Komponente wird auch die *LayerGroup* erstellt. Bei der Löschung der Komponente werden alle Layers der Gruppierung von der Karte entfernt.

Listing 14: Erstellung und Cleanup der Komponente

```
1 const elevationLayer = useRef();
2
3 useEffect(() => {
4   // Erstellung einer LayerGroup bei der Erstellung
5   // der Komponente
6   elevationLayer.current = new L.LayerGroup();
7
8   // Vor Loeschung der Komponente alle Elemente der
9   // LayerGroup loeschen
```

```
10    return () => {
11      elevationLayer.current.clearLayers();
12    };
13  }, []);
```

5.4.9 Erstellung und Export des neuen Tracks

Für die Erstellung des neuen Tracks müssen alle Segmente und die Routingabschnitte miteinander verbunden werden. Besitzen zwei Teile gleiche Koordinaten an erster oder letzter Stelle, können sie an diesen Punkten zusammengefügt werden. Der Benutzer hat die Möglichkeit jede Koordinate der übernommenen Segmente als Start- und Endpunkt zu wählen. Das System geht dann mit einer Schleife über alle Segmente sowie Routingabschnitte und fügt diese entsprechend zusammen (s. Listing 15).

Listing 15: Zusammenfügung der einzelnen Trackbausteine

```
1  // Ein Array fuer Segmente und Routingabschnitte
2  let all = sectionsToKeep.concat(route);
3
4  let allLength = allParts.length;
5  for (let i = 0; i < allLength; i++) {
6    for (let j = 0; j < allParts.length; j++) {
7      // Start der Abschnitts stimmt mit Koordinate
8      // überein
9      if (turf.distance(allParts[j][0],
           coordinateToFind) < 0.01) {
10       // Koordinaten in das Array des neuen Tracks
11       // einfügen
12       for (const obj of allParts[j]) {
13         newTrack.push(obj);
14       }
15       // Abschnitt aus dem Such-Array entfernen
16       allParts.splice(j, 1);
17       // Neue Koordinate für die Suche definieren
18       coordinateToFind = newTrack[newTrack.length -
             1];
19       // Aktueller Schleifendurchgang kann
             abgebrochen werden
20       break;
```

```
21          // Ende der Streckenabschnitts stimmt mit
                Koordinate überein
22      } else if (
23      turf.distance(allParts[j][allParts[j].length -
            1], coordinateToFind) <
24      0.01
25      ) {
26          const reversed = allParts[j].reverse();
27          for (const obj of reversed) {
28              newTrack.push(obj);
29          }
30          allParts.splice(j, 1);
31          coordinateToFind = newTrack[newTrack.length -
                1];
32          break;
33      }
34    }
35  }
```

Für den Export wird aus dem neuen Koordinaten-Array eine GeoJSON Linie erstellt. Aus diesem Objekt generiert das System anschließend ein Blob (binary large object). Ein Blob stellt rohe binäre Daten dar, zu welchem mit Hilfe von *createObjectURL* eine URL erstellt wird, die das Objekt repräsentiert. Daraufhin kann das System dem Benutzer das Objekt als gpx-Datei herunterladen lassen.

5.5 Ergebnisse

In diesem Kapitel soll ein Überblick über die wichtigsten Ergebnisse verschafft werden.

In Abbildung 24 ist die Aufteilung der Benutzeroberfläche zu sehen. Der größte Bereich wird von der Karte abgedeckt und der restliche Teil, auf der linken Seite, mit der *Controls* Komponente. Rechts oben auf der Karte liegt die *ColorMap* Komponente und rechts unten ist der Button um die *ElevationChart* Komponente (s. Abbildung 25) anzuzeigen.

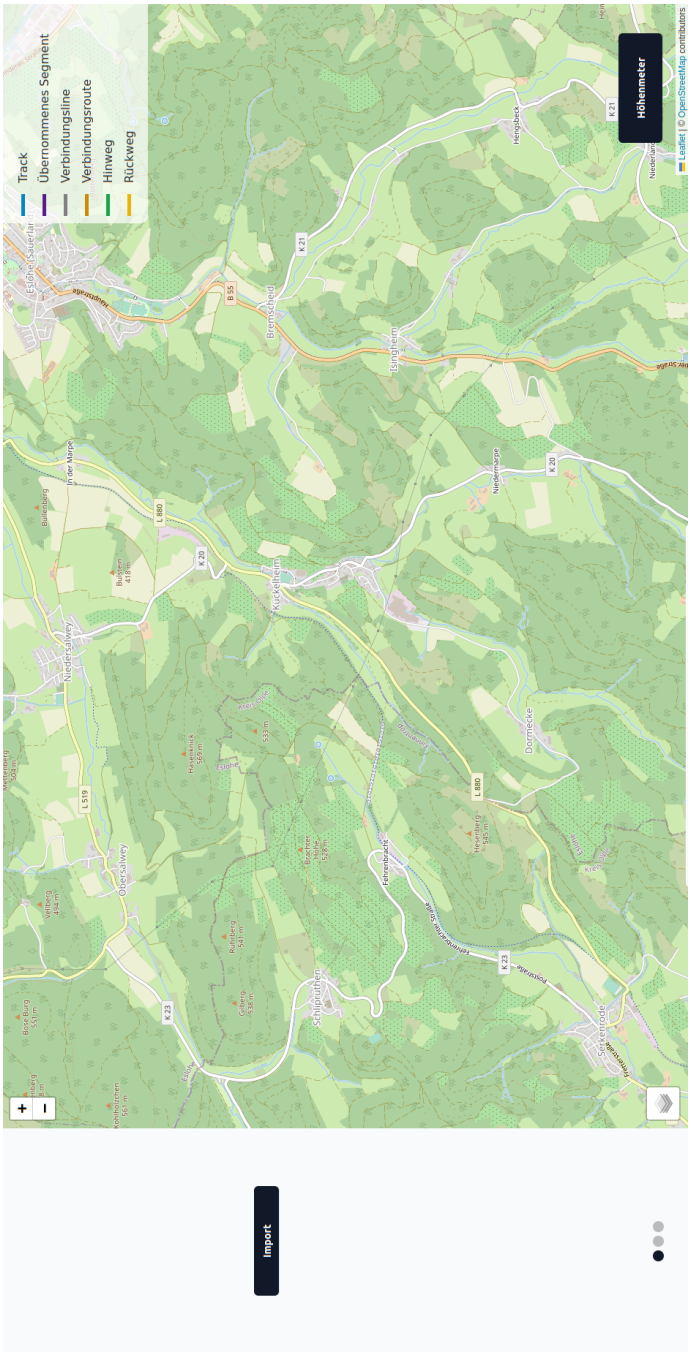

Abbildung 24: Vollständige Benutzeroberfläche

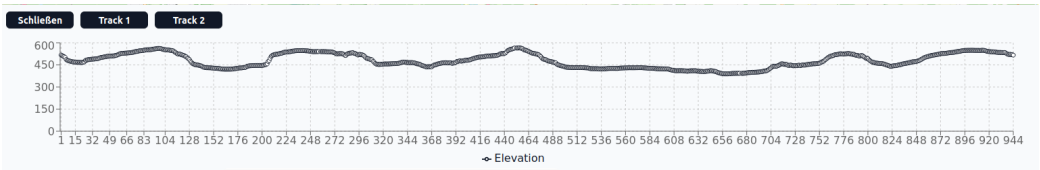

Abbildung 25: Höhenmeterchart

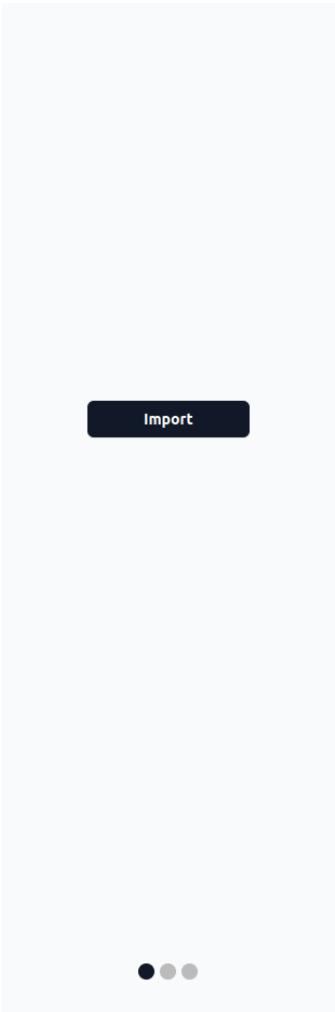

Abbildung 26: UI für Import

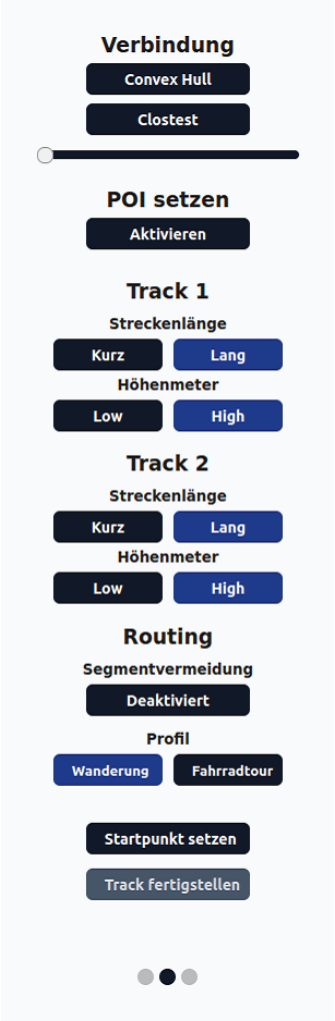

Abbildung 27: UI für die Einstellungen

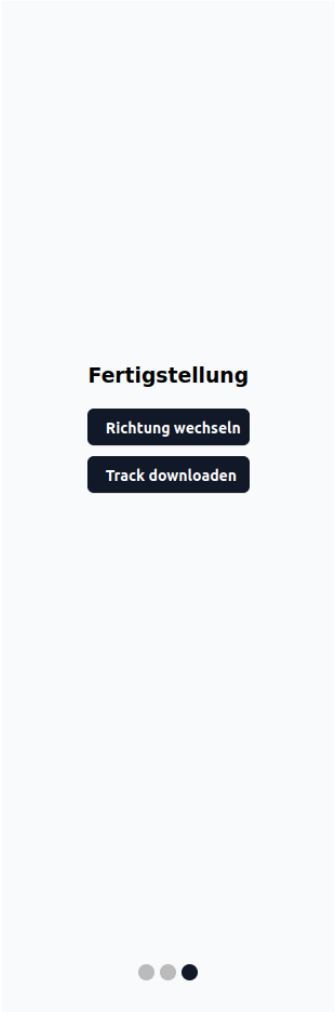

Abbildung 28: UI für die Fertigstellung

Die Steuerungselemente von *Controls* sind in drei Bereiche unterteilt. Der erste Bereich (s. Abbildung 26) ist während des Importvorgangs aktiv. Die Steuerungselemente zur Beeinflussung der Kombination sind aktiv sobald zwei Tracks importiert sind (s. Abbildung 27). Hat der Benutzer seine Präferenzen angepasst, kann er mit dem Drücken auf *Track fertigstellen* in den letzten Bereich wechseln. Der letzte Bereich wird in Abbildung 28 dargestellt und ermöglicht den Richtungswechsel des Tracks sowie das Herunterladen der gpx-Datei.

In Abbildung 29 ist ein Track des Typs `Rundstrecke mit Hin- und Rückweg` zu sehen. Der Hin- und Rückweg wird dabei korrekt erkannt und in grün bzw. orange hervorgehoben.

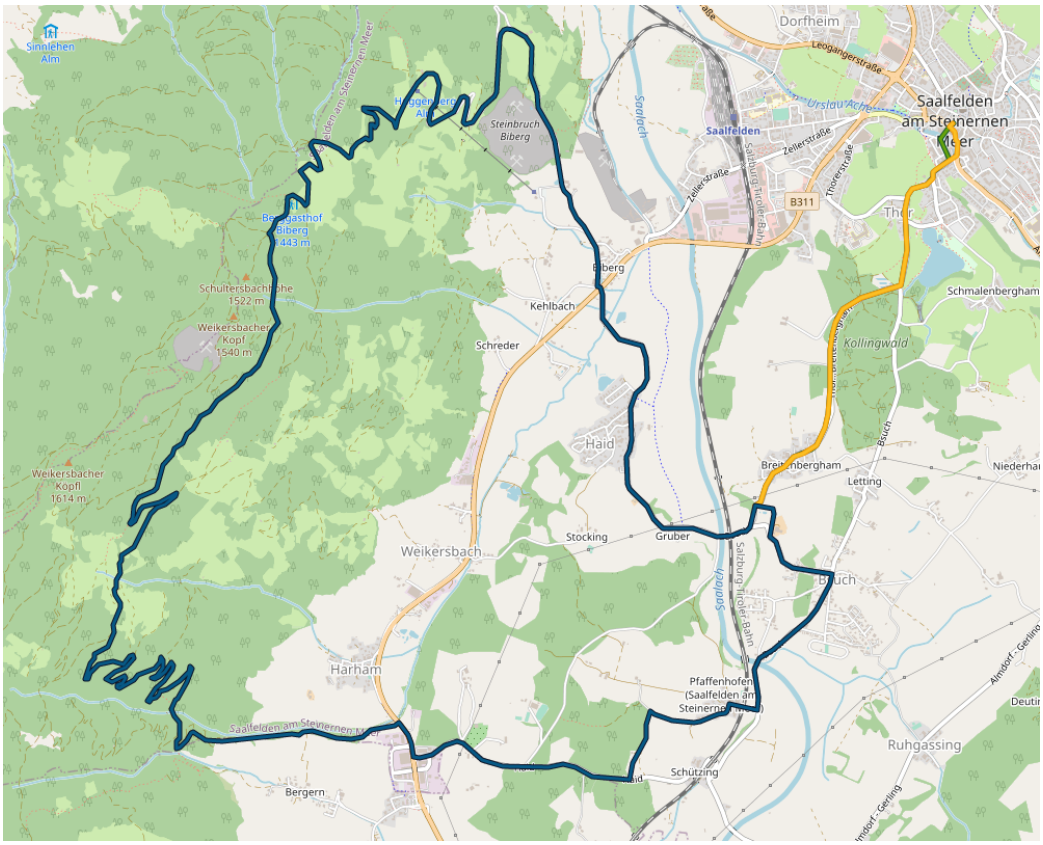

Abbildung 29: Track mit Hinweg (grün) und Rückweg (gelb)

In Abbildung 30 werden zwei nebeneinanderliegende Tracks dargestellt. Eine Variante der Kombination davon, ist in Abbildung 31 zu sehen. Dabei werden die zu übernehmenden Segmente in lila und die berechnete Route in orange hervorgehoben. Der neue vollständig kombinierte Track ist in Abbildung 32 dargestellt.

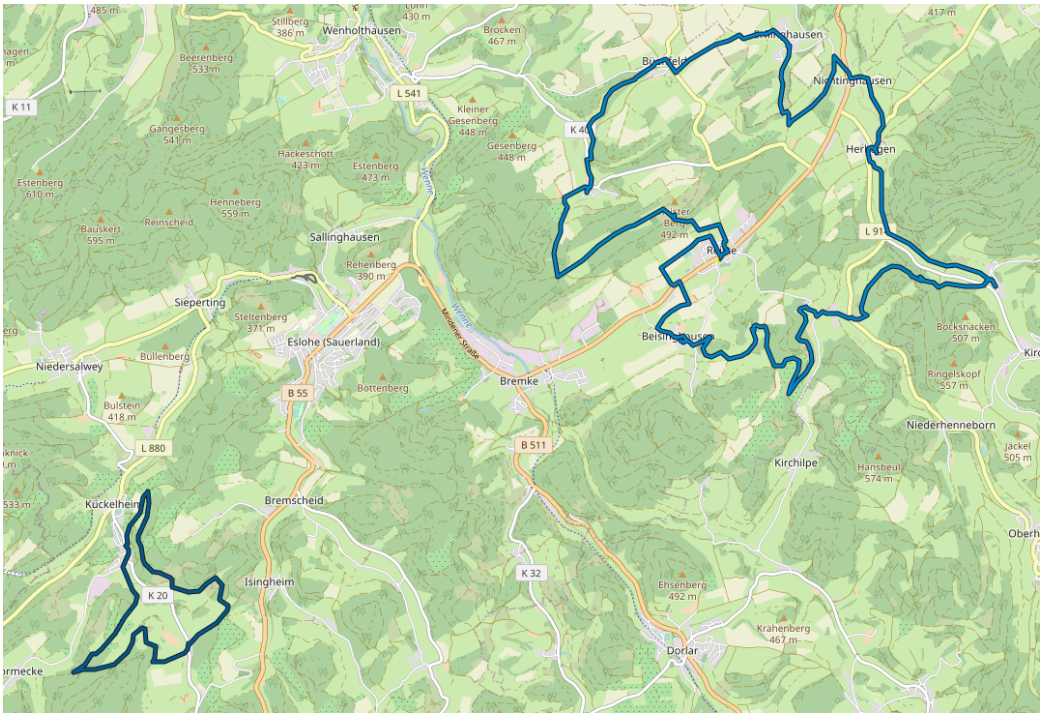

Abbildung 30: Zwei nebeneinander liegende Tracks

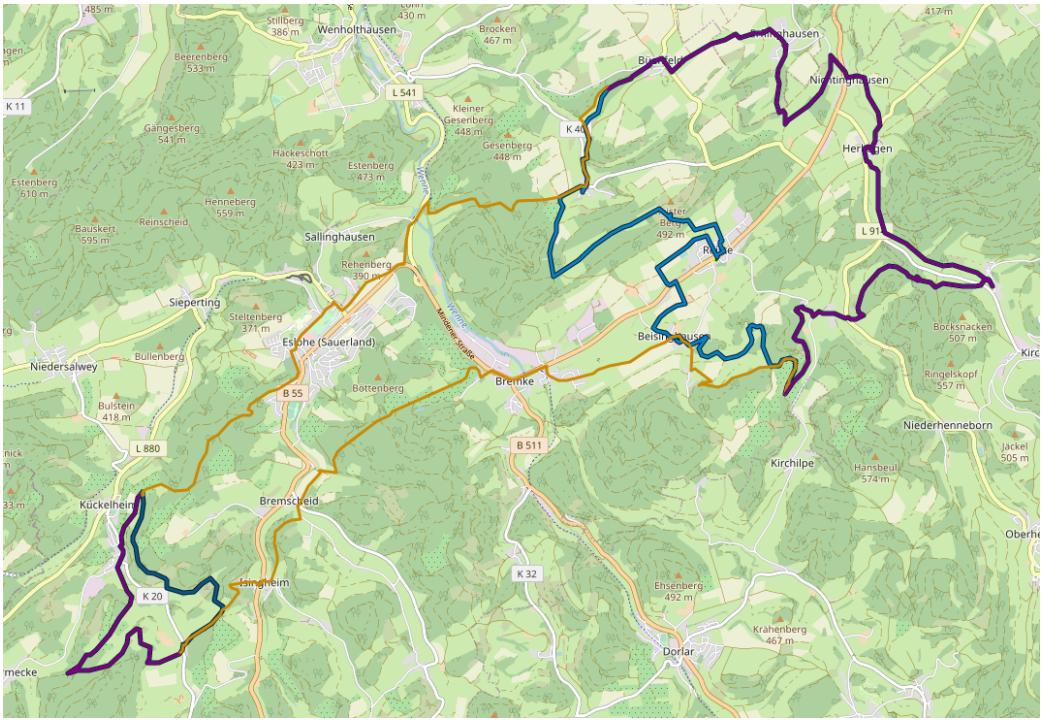

Abbildung 31: Zwei Tracks mit den zu behaltenden Segmenten (lila) und dem Routing (orange)

In den Abbildungen 33 und 34 werden zwei sich überschneidende Tracks dargestellt und die passende Auflösung der Überschneidung gezeigt.

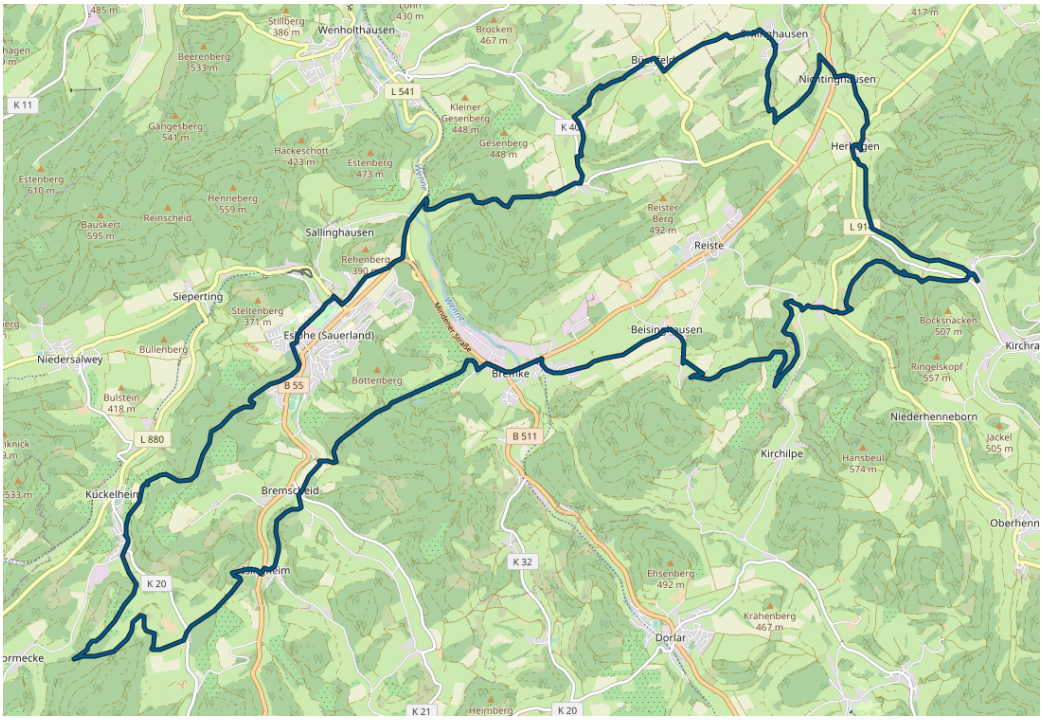

Abbildung 32: Exportierte Kombination der Tracks aus Abbildung 30 und 31

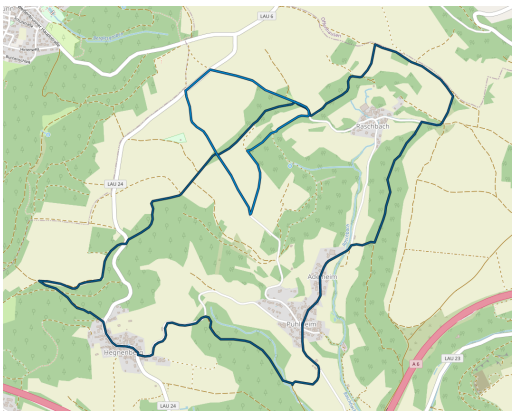

Abbildung 33: Zwei sich überschneidende Tracks

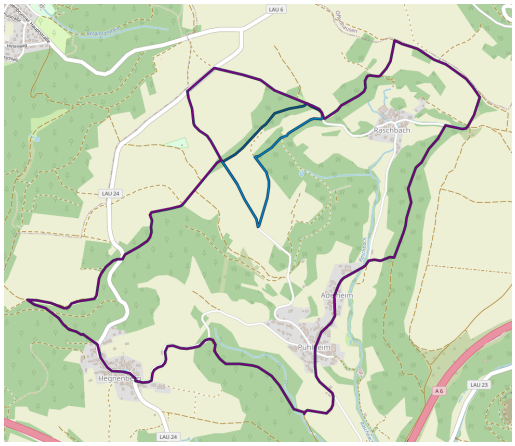

Abbildung 34: Auflösung von zwei sich überschneidenden Tracks

6 Evaluierung

6.1 Auswertung der Anforderungen

6.1.1 FA-1

Die Anforderung FA-1 verlangt, dass das System mit den verschiedenen Tracktypen umgehen kann. Mit dem in Kapitel 5.4.1 beschriebenen Verfahren ist es dem System möglich, alle drei definierten Tracktypen zu erkennen und in weiteren Funktionen zu berücksichtigen.

6.1.2 FA-2

Die Anforderung FA-2 fordert, dass das System mit den verschiedenen Trackkonstellationen umgehen kann. Die Überprüfung auf die Konstellation wurde bereits in Kapitel 5.4.4 beschrieben. Einige Konstellationen haben bestimmte Einschränkungen, welchen den vollen Funktionsumfang des Systems verhindern. Insgesamt gibt es drei verschieden Szenarien, bei denen es Abweichung gibt:

Tracktyp Strecke Zu beachten ist, dass die Connectorpoints bei einer Strecke immer auf der ersten und letzten Koordinate liegen. Die Strecke wird also immer vollständig übernommen. Zusätzlich gilt, dass keine sinnvolle Kombination bei einer Strecke mit Konstellation *Überschneidung* möglich ist.

Konstellation Überschneidung Bei einer Überschneidung findet keine Kombination durch den Convex Hull Algorithmus oder der Closest Methode statt. Diese Konstellation wird wie in Kapitel 5.4.5 beschrieben, gelöst.

Konstellation Enthaltend Liegt ein Track innerhalb des anderen Tracks, ist kein Convex Hull Algorithmus möglich. Diese Methode funktioniert nur, wenn die Hülle aus Koordinaten von beiden Tracks besteht und somit aus der Hülle Verbindunglinien extrahiert werden können.

6.1.3 FA-3

Die Anforderung FA-3 verlangt, dass das System die Tracklänge und Höhenmeter als Benutzerpräferenz bei der Kombination berücksichtigt. Diese Anforderung wurde erfolgreich umgesetzt und ist in Kapitel 5.4.3 genau beschrieben.

6.1.4 FA-4

Die Anforderung FA-4 fordert, dass das System die Höhenmeter für den Benutzer darstellen kann. Diese Anforderung wurde erfolgreich umgesetzt und Informationen zu der technischen Umsetzung sind in Kapitel 5.4.8 zu finden.

6.1.5 FA-5

Die Anforderung FA-5 verlangt, dass das System eine passende Route zwischen den Segmenten ausgibt und eine Alternative anbietet. Dies wurde erfolgreich umgesetzt und wird in Kapitel 5.4.7 erläutert.

6.2 Fazit

Anforderung	Status
FA-1	Erfüllt
FA-2	Erfüllt
FA-3	Erfüllt
FA-4	Erfüllt
FA-5	Erfüllt

Tabelle 1: Status der Anforderungen

Alle Anforderungen wurden erfolgreich in einem Prototyp umgesetzt (s. Tabelle 1). Es werden die verschiedenen Situationen dargestellt und die Prozesse für die Kombination erläutert. Auch werden die verschiedenen Anbieter und Services beleuchtet.

7 Zusammenfassung und Ausblick

Für diese Abschlussarbeit wurde ein Konzept für die Möglichkeiten der Automatisierung von benutzergesteuerter Kombination von GPS Tracks erstellt und anschließend in einem Prototyp umgesetzt. Dabei wurde untersucht welche Szenarien entstehen und was für Algorithmen unterstützend eingesetzt werden können. Der Benutzer sieht bei der Kombination der Tracks wie verschiedene Präferenzen den Prozess beeinflussen und hat auf Grund der Karte einen klaren Überblick über die Tracks.

In einer zukünftigen Version der Umsetzung sollte die Benutzeroberfläche überarbeitet werden, um eine bessere Benutzerführung durch die einzelnen Schritte zu ermöglichen. Des Weiteren sollen Quality of Life Anpassungen wie das Löschen von Tracks und Erklärungen zu bestimmten Funktionen eingefügt werden.

Auch soll die Berücksichtigung der Benutzerpräferenzen angepasst werden. Die aktuelle Umsetzung liefert gute Ergebnisse, jedoch ist eine komplexere und feinere Berücksichtigung möglich.

Neben neuen und verbesserten Features ist auch die Verbesserung des internen Datenaustausches möglich sowie die Optimierung von Funktionen.

8 Literatur

Agafonkin, V. (2022). *Leaflet*. Verfügbar 2. Januar 2023 unter https://leafletjs.com/

Apple. (2022). *Apple Maps*. Verfügbar 2. Januar 2023 unter https://www.apple.com/de/maps/

DeMers, M. N. (2014). *GIS for Dummies* (2. Aufl.). For Dummies.

El-Rabbany, A. (2002). *Introduction to GPS : the Global Positioning System* (2. Aufl.). Artech House.

Garmin. (2022). https://connect.garmin.com

GarminBasecamp. (2022). https://www.garmin.com/de-DE/software/basecamp/

Google. (2022a). *Google Maps API*. Verfügbar 2. Januar 2023 unter https://developers.google.com/maps/documentation/javascript

Google. (2022b). *GoogleMaps*. Verfügbar 2. Januar 2023 unter https://www.google.com/maps/

GPSTrackEditor. (2022). http://www.gpstrackeditor.com/

GPXStudio. (2022). https://gpx.studio/

Komoot. (2022). https://www.komoot.de/plan

Koptyug, E. (2021). *Number of smartphone users in Germany 2009-2021*. Verfügbar 2. Januar 2023 unter https://www.statista.com/statistics/461801/number-of-smartphone-users-in-germany/

Laricchia, F. (2022). *Quarterly revenue of Garmin from 2017 to 2021*. Verfügbar 2. Januar 2023 unter https://www.statista.com/statistics/1008102/quarterly-net-sales-of-garmin-by-segment/

Mapbox. (2022). *Turf.js*. Verfügbar 2. Januar 2023 unter https://turfjs.org/

Meta Platforms, I. (2022a). *Create React App*. Verfügbar 2. Januar 2023 unter https://create-react-app.dev/

Meta Platforms, I. (2022b). *React.js*. Verfügbar 2. Januar 2023 unter https://reactjs.org/

Microsoft. (2022). *Azure Maps*. Verfügbar 2. Januar 2023 unter https://azure.microsoft.com/de-de/services/azure-maps/

MyGPSFiles. (2022). https://www.mygpsfiles.com/app/

National Coordination Office. (2018). *Data From the First Week Without Selective Availability*. Verfügbar 2. Januar 2023 unter https://www.gps.gov/systems/gps/modernization/sa/data/

National Coordination Office. (2021). *Selective Availability*. Verfügbar 2. Januar 2023 unter https://www.gps.gov/systems/gps/modernization/sa/

openrouteservice. (2022). *Openrouteservice*. Verfügbar 2. Januar 2023 unter https://openrouteservice.org/

OpenStreetMap. (2022). *OpenStreetMap*. Verfügbar 2. Januar 2023 unter https://www.openstreetmap.de/

OSRM. (2022). *Open Source Routing Machine*. Verfügbar 2. Januar 2023 unter https://project-osrm.org/

Outdooractive. (2022). https://www.outdooractive.com/de/

Potter, J. (2022). *npm trends*. Verfügbar 2. Januar 2023 unter https://npmtrends.com/@angular/core-vs-angular-vs-react-vs-vue

Strava. (2022). https://www.strava.com

Topografix. (2022). GPX: the GPS Exchange Format. https://www.topografix.com/gpx.asp

Vercel, I. (2022). *Next.js*. Verfügbar 2. Januar 2023 unter https://nextjs.org/

Wanderreitkarte. (2022). https://www.wanderreitkarte.de/

ZIV, Z.-I.-V. (2022). *Marktdaten 2021*. Verfügbar 2. Januar 2023 unter https://www.ziv-zweirad.de/marktdaten/detail/article/marktdaten-2021/

9 Anhang

Übersicht der elektronischen Anhänge:

- Quellcode des Prototyps
- Abschlussarbeit in elektronischer Form