OSM Ski Resort Routing

Wenzel Friedsam  
st161561@stud.uni-stuttgart.de  
University of Stuttgart

Robin Hieber  
st161972@stud.uni-stuttgart.de  
University of Stuttgart

Alexander Kharitonov  
st160809@stud.uni-stuttgart.de  
University of Stuttgart

Tobias Rupp  
in81143@stud.uni-stuttgart.de  
University of Stuttgart

Figure 1: Visualisation of the Matterhorn ski paradise.

ABSTRACT

We present OSM Ski Resort Routing, an app that combines the concept of pathfinding and navigation with skiing. It provides an interactive 2D and 3D visualisation of arbitrary ski resorts using OpenStreetMap data and can be used to compute and display the optimal route between any waypoints in the resort.

CCS CONCEPTS
- Information systems → Geographic information systems.

KEYWORDS
OpenStreetMap, routing, ski resorts, 3D visualisation

1 INTRODUCTION

Skiing is a popular sport in many countries around the world. In the past, ski resorts have vastly expanded their lift and slope network to increase their region size or to connect with other resorts, creating large ski regions. This has lead to ski resorts with several hundreds of kilometres of connected ski slopes, and therefore also to complex routes which need to be taken to get to a certain location in a ski region.

Ski resort trail maps are schematic maps which contain most of the relevant information, like lifts, slopes, and other infrastructure of a ski resort. They can be used to orient yourself and find a way to a specific location, but in very complex ski regions, finding a good route from one waypoint to another can be difficult. In recent years, smartphones have become so popular that nearly everybody uses one now. Even while skiing it is used to take photos, check the weather, or for navigation on roads from home to the ski region.
However, to our knowledge, smartphones are currently not typically used for routing on the slopes and lifts. Therefore, we created an app which allows the user to visualise arbitrary ski regions and start a navigation to find optimal routes for a given list of waypoints. We use OpenStreetMap (OSM) as our main data source to generate the needed graph, containing all lifts and slopes, for routing.

2 RELATED WORK

For visualisation, replacement of traditional printed piste maps with digital maps that are displayed on smartphones was studied in [12] and there are already websites that use OpenStreetMap to visualise ski related data. openskimap.org [5] is a website that uses the mapping platform Mapbox to visualise ski regions on a tile based map. Additional info like distances or ascent/descent information can be viewed by clicking on a lift or slope.

Another website, opensnowmap.org [6], uses a custom Mapnik OSM tile style to visualise ski regions. This website offers similar features as openskimap.org. In addition to that, basic routing is possible, but we observed some problems while using it. In some ski regions, ways are not always properly connected in the OSM data. In these cases, the navigation fails. In addition to that, the navigation tries to find the shortest path in terms of distance. Therefore, routing sometimes prefers using lifts downwards or slopes upwards. This is usually not wanted during skiing. In our approach, we tried to fix these issues to provide routing that is suitable for practical use.

For visualisation, there are some products like Google Maps [2] and FATMAP [1] but they are closed-source and lack a routing function for skiing. In addition to that, the ski lift data provided by Google Maps is often outdated or incomplete.

3 STRUCTURE

We use the libGDX framework as a basis which allows us to export our app to both desktop and android devices. LibGDX is an open-source Java and OpenGL based framework, mainly used for game development [3].

Our app is structured into several modules, which are explained in more detail in the following sections (see Fig. 2 for a graphical overview). The main components of the app are the management of OSM and elevation data, navigation, map visualisation, and the user interface. All the code can be found at [8].

3.1 OSM Data

Our data source for our ski graph is OpenStreetMap. All data is loaded dynamically using the Overpass API, which is an open source web API that can be used to query data from the OSM database. Queries are written in the Overpass Query Language and sent to an Overpass Instance, which returns the specified data [9].

After starting the app, the user needs to select a ski resort on the map by selecting a point on the map. These resort are modelled in OSM as ways or relations using the tag landuse=winter_sports. Unfortunately, the provided raw OSM data cannot be straightforwardly used for navigation in all ski resorts. We tackle this problem in section 4.

3.2 Elevation Data

OSM map data is restricted to a planar map projection, and so does not generally include information about the terrain elevation. However, elevation data is critical for 3D visualisation, and is also quite important for navigation accuracy when steep slopes are involved.

For our app, we chose to use elevation data from the Shuttle Radar Topography Mission (SRTM) 1 Arc-Second Global dataset by the United States Geological Survey (USGS) [10]. It is publicly available and features elevation data for most of the earth’s surface at 1 arc-second resolution, which roughly corresponds to 30 meters between data points on average.

This data uses spherical mapping (unlike the planar mapping of OSM), so we converted the dataset to a planar format for compatibility. Another problem with the data is that in some areas, there are so-called voids in the data, where elevation data is missing completely. This results in a strong distortion of the resulting visualisation and significant navigation problems. Fortunately there also exists void-filled data[11] to fix this problem (See Figure 3).

Figure 3: Left: Elevation data with voids (red), Right: Same data with voids filled

In total, this elevation data for the Alps for example, is around 3GB in size. We have taken several steps to handle such a large amount of data on memory-limited devices such as phones. First of all, the data is provided by a separate elevation server and only...
the data relevant for displaying is requested through the Internet, with a local cache maintained by the app to reduce the number of repeated network requests. Moreover, the required elevation data on the displaying device is dynamically loaded into a quadtree structure, at a reduced data resolution. As the camera gets closer to a specific part of the terrain, higher resolution data is loaded for that area and the quadtree structure is locally refined, see Figure 5. Small quadtree nodes with equal resolution are automatically combined into larger nodes again to minimise memory usage.

To further remedy issues with low-end devices and slow Internet connections, we designed a specialised compression algorithm for the elevation data which is used for all network transmissions and the local elevation cache. This compression algorithm relies on the specific properties of the elevation data to be significantly more effective and faster than the Deflate/Zip algorithm for mountainous areas, see table 1).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Size reduction</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our compression</td>
<td>46.6%</td>
<td>47 ms</td>
</tr>
<tr>
<td>Deflate (Zip)</td>
<td>32.4%</td>
<td>385 ms</td>
</tr>
</tbody>
</table>

Table 1: Elevation compression algorithm performance compared to Zip for data of a mountainous area with 8MB size

For navigation purposes, it is necessary to calculate the elevation of OSM nodes. For this, elevation data around the nodes is loaded at high resolution and the surrounding data points are used to linearly interpolate an estimate for the elevation of each node.

3.3 Navigation

In order to implement pathfinding, the Dijkstra algorithm is used to determine the shortest route via ski slopes and lifts.

For this purpose, an edge cost function is required, e.g. based on the way length. To calculate the weight of the edges we use the coordinates provided by OSM as well as the elevation data.

Another important factor is the time which is needed to go along the edges. Going upwards on a ski slope is a lot slower than skiing downwards for example (also see section 4.2). Therefore, the pathfinding edge cost function also takes other factors into account, like elevation changes and further OSM information, to approximate the edge traversal time as well as possible. For some lifts, the time required to take a lift to is included in the tags of the OSM way. For all other lifts, this value is estimated based on some heuristics. Not all lifts operate at the same speed, and that speed can even change from day to day, but a rough estimate can easily be calculated from the type and length of the lift, which is sufficient for our purposes.

With these considerations in mind, we designed an edge cost function that roughly estimates edge traversal time, which results in practically useful navigation.

In addition to that, our pathfinding algorithm is adaptive, i.e. it can handle slope and lift restrictions. Users can set the desired difficulty level of the slopes and the permitted lifts, if they prefer not to use some lifts or slope difficulty levels. Another feature of our navigation are multiple waypoints, which allow the user to choose intermediate destinations in addition to the start and end point.

3.4 Map Visualisation

Our app contains two different map visualisation modes: A 2D mode, which displays the standard Mapnik OSM tiles to show the map, as well as a 3D mode, which combines the 2D map with the elevation data to display the environment as a 3D model. The user can switch between these two different visualisation modes at any time.

In both modes, the ski area graph, containing all lifts and slopes, as well as the current navigation are rendered as an overlay on the map. The zoom level of all displayed tiles is equal across the entire screen in 2D mode, but depends on the current map zoom. In the 3D map, the tile zoom level is dynamic. Tiles which are further away from the camera have a lower zoom level. Using this technique, we were able to achieve a visibility range of several hundred kilometres (see Fig. 4 for graphical example of both modes).
3.5 User Interface
The user interface, in addition to the map visualisation, is an important part of the front-end component. Various aspects have been taken into account in order to enable simple and intuitive operation of the application. While the application’s bar in the lower area allows an intuitive and simple interface, the user still has all necessary configuration options. This includes for example the desired configuration of the permitted slopes and lifts for pathfinding.

At the top of the screen there is a search bar, which allows the user to search for slopes and lifts in the current ski region. The names of these ways are extracted from OSM tags.

4 CHALLENGES
In some ski regions, the graph generated by the raw OSM data requires some modifications before it can be used for pathfinding.

4.1 OSM Data incompleteness
The main problem are slopes and lifts which are not properly connected (see Fig. 6).

![Figure 6: An example of a gondola (middle) which is not directly connected to the slopes (left).](image)

We implemented an algorithm to detect those situations and add edges for the purpose of routing. The result of performing the algorithm on the example in Fig. 6 can be seen in Fig. 7.

![Figure 7: A visualisation of the generated edges (green). These edges are not displayed in the app and are only used for navigation purposes.](image)

4.2 Practical Navigation
Unlike normal navigation on streets, the shortest or fastest path is not always the most desired path for skiers.

A skier usually avoids larger ascents on slopes and prefers to take a lift instead, even if the total distance to reach the end point is larger. This is not a problem for our pathfinding algorithm because our cost function is based on the time needed and not the distance. The time, and therefore cost, of ascending a slope is significantly higher than the cost to ski down the slope.

Another problem are routes where it is faster to take a lift downhill than to use the slope. This is not intended in most cases, but sometimes it is only possible to reach a part of the ski region by riding a lift downhill. Therefore, removing the edges from the top to the bottom of all lifts was not an option. Our solution was to increase the cost of these edges by a constant factor. The pathfinding algorithm can still use these edges, but prefers using the slopes.

5 CONCLUSION
Our app allows users to choose an arbitrary location on the map to select and load ski regions. The ski region is loaded dynamically using the Overpass API and required elevation data is loaded from an elevation server instance. The elevation data is processed efficiently by using an adaptive quadtree and our compression algorithm. The loaded raw OSM data is automatically analysed and modified to fix smaller issues which can cause problems with pathfinding.

Users have the ability to start a navigation within the ski region with any number of waypoints. Using waypoints, it is possible to calculate a route for round trips given a set of specified points on the map. This can help people orient themselves, especially in larger ski areas, which have become more and more common in the last few decades. In addition to that, our app allows users to choose which types of lifts and slope difficulties they want to use.

For our routing purposes we already made some adaptions to capture travel experience and pursuing this line of thought in future work by implementing a prize-collecting TSP planning may be the most promising feature.

REFERENCES