EleNa: Elevation-based Navigation

The Heisenbug_Reloaded Team
Ashish Ranjan
Abhishek Somani
Srikanth Prabala
Deep Chakraborty
Carlos Daniel Mondragón Chapa

Purpose

The usual Navigation System tries to get the shortest or the fastest path from one point to another. However, in some cases, having the ability to further customize the path is desirable. Such a case could arise if, while working out, following a path that maximizes or minimizes the elevation gain is more convenient than the usual shortest path in order to get better training results. Therefore, the EleNa project tries to find the best path between two points according to the user preferences regarding elevation gain and some distance constraint with respect to the shortest path.

Features

- EleNa calculates the path between two points that maximize or minimize the elevation gain constrained to a percentage increase with respect to the shortest path.
- The origin and destination points along with the distance constraint are set using the command line.
- The origin and destination points are set using coordinate values.
- The resulting path is drawn on a map.

Use cases

The user can set an origin and a destination by using the corresponding coordinates (latitude and longitude) and a distance constraint determined by a percentage that represents the extra distance with respect to the shortest path. Once the coordinates and the distance constraint are set, the user can choose to minimize or maximize the elevation gain. The algorithm will then output the calculated path on a map.

Model Proposals

All models calculate the resulting path within a preselected area in the map. The area contains the origin and the destination and it can be seen as a rectangle that contains all the possible paths.

1. The first proposal does an exhaustive search of all the possible paths from origin to destination. It then finds the elevation gain in each path and chooses the optimal path depending on the user settings. This algorithm works in a reasonable amount of time if the desired extra distance to be travelled with respect to the shortest path is not larger than 10%.

2. The second proposal is much faster than the first one as it uses Dijkstra's algorithm with a slight modification. Instead of taking into consideration only the elevation gain, the distance between nodes, or some impedance function that gives weight to the edges connecting the nodes; the algorithm uses the same greedy approach by selecting the best node based on the lowest elevation gain, but then looks into the next possible node based on the shortest distance. This approach guarantees a shorter elevation gain compared to that of the shortest path, but keeps the increase in distance relative to the shortest path within the desired percentage.
3. The third proposal finds the shortest path from origin to destination and gets the amount of nodes along that path; the amount of nodes in the shortest path is NSP. The algorithm then finds all the paths from origin to destination that have at most MNP nodes such that $MNP > NSP$. The difference between MNP and NSP is some value, v, that was empirically chosen by gradually trying different numbers and checking the performance of the algorithm with each one. Finally, the algorithm picks the path with the lowest elevation gain from all the paths that contain $NSP + v$ or less nodes.

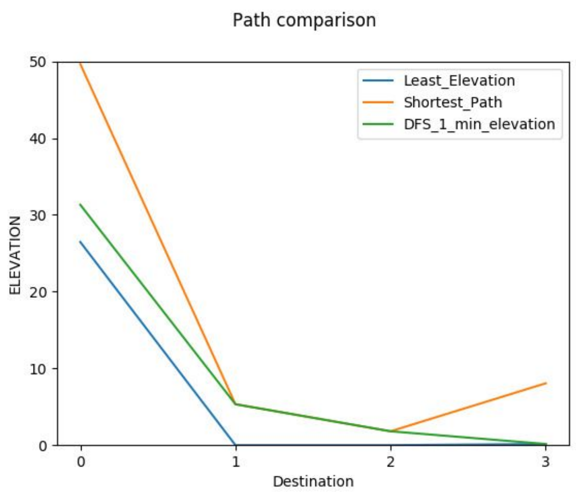
Experimental Design

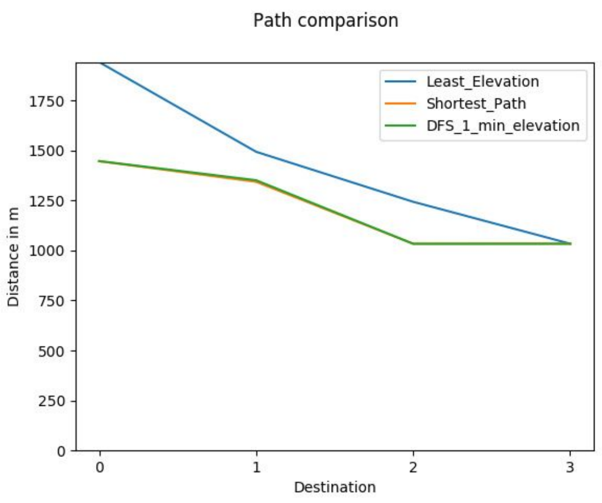
For all the models, the same origins and destinations are randomly generated. Then, the shortest path (SP) and the path with the least elevation gain (LEG) is calculated for each origin and destination point using Dijkstra's algorithm. Afterwards, all algorithms are tested against the output of Dijkstra's algorithm. If the algorithm works properly, then the distances along the paths and the elevation gains should be similar.

Suppose that Model 1 outputs a path MoP_1, if we compare this path to SP it should be within the constraint set by the user and have a lower elevation gain, which, at the same time, should be close to the one in LEG.

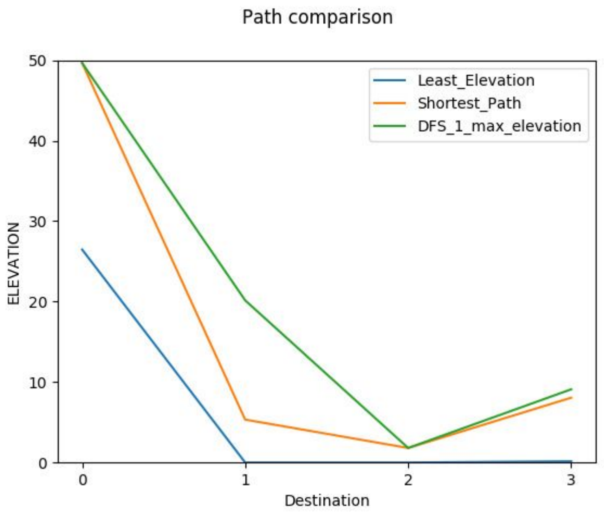
Findings and Conclusions

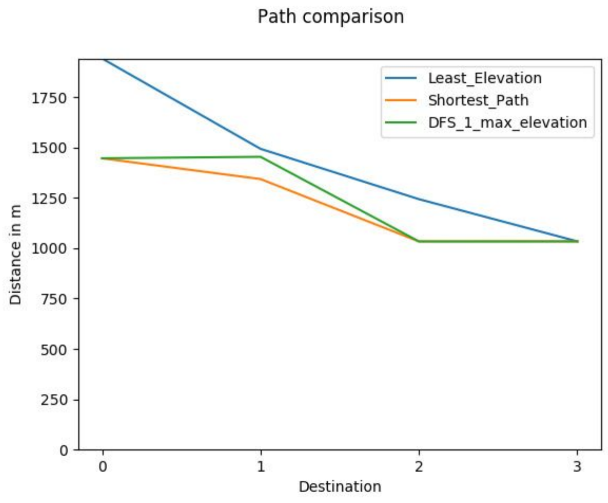
1. Model 1



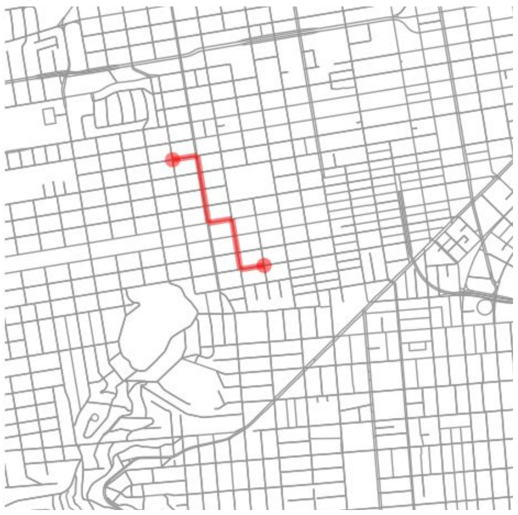


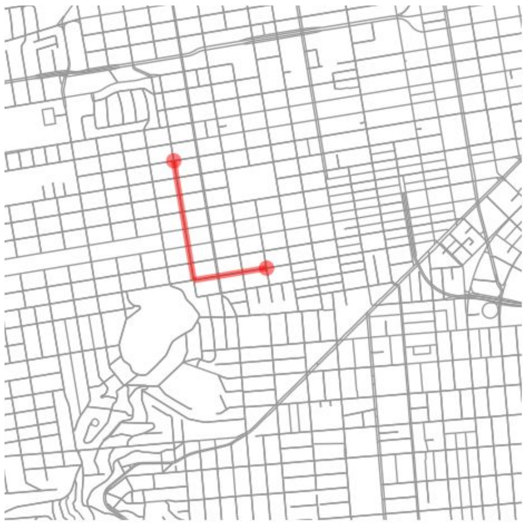
In the case of Minimum Elevation Gain, the first model was able to find the optimal path one time out of four, but was close to the optimal path in the other three tests.





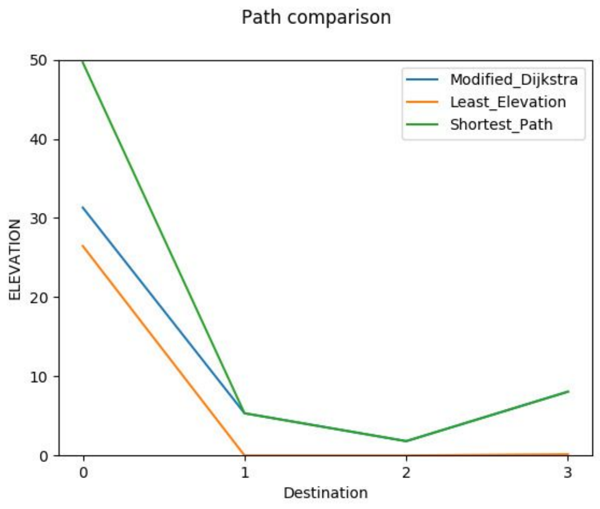
In the case of Maximum Elevation Gain, the first model was able to find a path that had more elevation gain than the shortest path on all the tests while keeping the distance from origin to destination within the desired percentage.

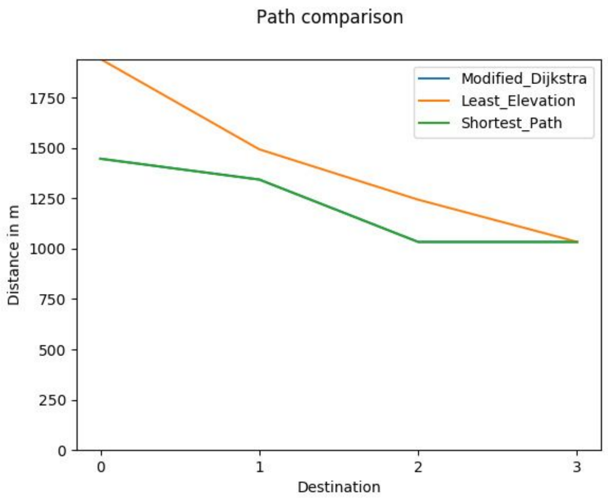




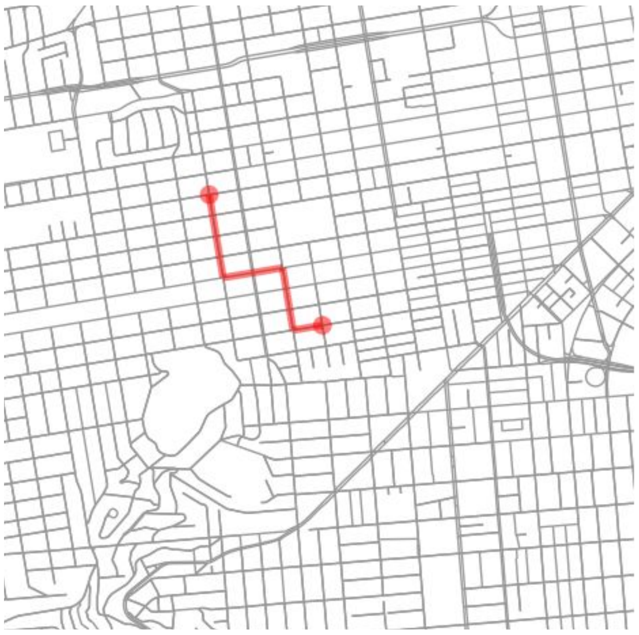
Here can be seen the paths predicted by the algorithm. On the left side is the path with the least elevation gain and on the right side is the path with the maximum elevation gain.

2. Model 2



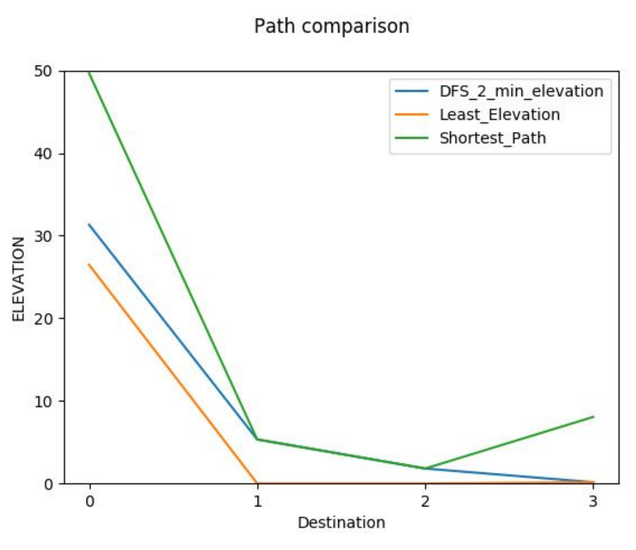


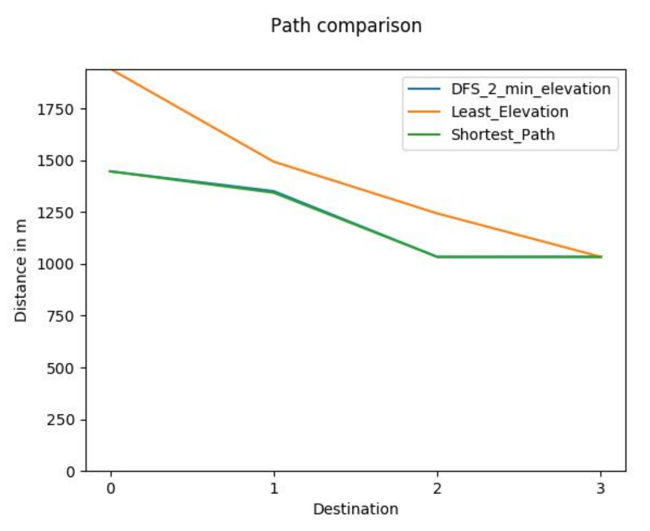
Model 2 can only calculate the Minimum Elevation Gain. In the case of the tests the algorithm was able to find a better path once, since in the other three cases the shortest path was very close of the optimal path with the Minimum Elevation Gain.



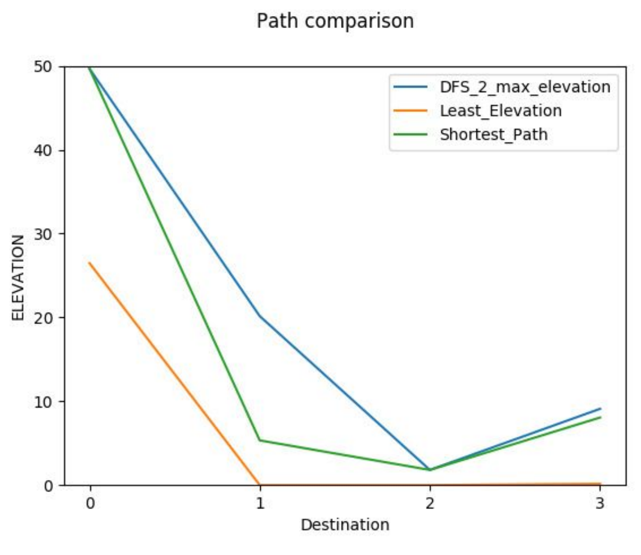
Here can be seen the path predicted by the algorithm.

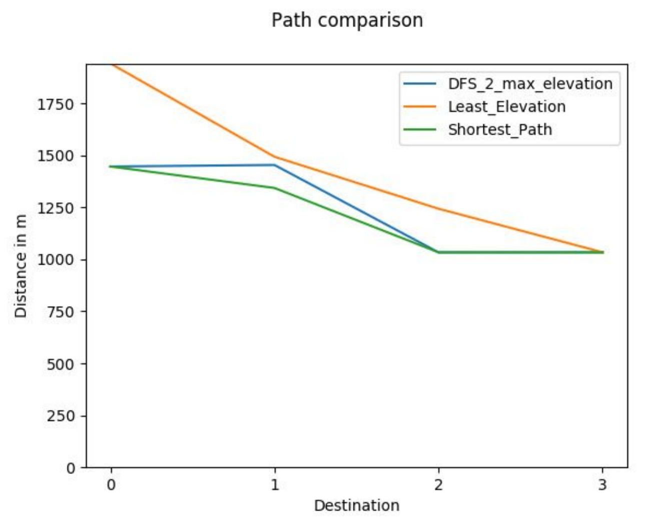
3. Model 3





In the case of Minimum Elevation Gain, the third model was able to find the optimal path one time out of four, but was close to the optimal path in the other three tests. Just like the Model 1.





In the case of Maximum Elevation Gain, the third model was able to find a path that had more elevation gain than the shortest path on all the tests while keeping the distance from origin to destination within the desired percentage. Just like the Model 1.
Even though the first and third model seem to be getting better results for the four chosen points, the second model is able to output a result in less time than the others. So for larger distances, it is the best model