Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Hyperloop Network Design

Group Project

Thomas Filippo Tavares Marinho
Mark Vero
tathomas@student.ethz.ch
mveroe@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Roland Schmid
Prof. Dr. Roger Wattenhofer

January 8, 2020

Acknowledgements

First and foremost, we want to thank our supervisor Roland Schmid for his excessive support, thoughtful insights and ideas, guidance, and patience during the long hours of the weekly meetings. We also want to express our gratitude towards Prof. Roger Wattenhofer and the whole of DISCO for making such projects possible.

We would like to offer our special thanks to Mr. Thomas Pferdekämper, who offered us his time and helped us take a leap towards a more realistic cost model.

Last but by no means least, we would like to thank our mates, Cristian Cioflan and Aron Szakacs for always being there for both technical and emotional support. Finally we would like to thank each other for the tireless work and occasional banter.

Abstract

This Group Project refines an existing project that searches for the shortest and cheapest path between two points in Switzerland for constructing a Hyperloop. The improvement is the result of a more realistic cost model which does not assume tunnelling to be equal everywhere. In the previous project, the path finding algorithm considers only rail- and motorways as regions where a tube can be built above the surface. Everywhere else, tunnelling is assumed to be required and incorporated at a constant cost per kilometre. This is a vague approach since the costs depend on very complex factors. After a discussion with an expert we came to 3 main aspects: the geological attributes of the ground, the total mass above the tunnel and the type of land use on the surface. We then extracted the available information from Open Street Maps to create a bitmap on which the path finding can be done. At last, we came up with a cost model for each type of pixel leading to improved results.

Contents

Chapter 1

Introduction

Switzerland's inhabitants depend more and more on its railway service and with the arising climate challenges this trend only seems to increase, setting a demand for radical improvements of the public transportation system. The hyper-frequented and enormously cheap short-distance flights also do not seem to be able to stand the test of time. Additionally, rising concerns about the climate make flying rather unpopular. For all this, the Hyperloop proposes a viable alternative. While many people are doing research on the physical realisation of the Hyperloop, it is also important to think about where and how to use the Hyperloop in the future to address the transportation problems our society is facing.

1.1 Existing Work

In this Group Project we want to extend an existing program, a product of a prior Master Thesis [1], which given a start and end point, automatically computes an optimal Hyperloop route. Its aim is to minimise building cost while maximising reduced travel time over its current alternative, the public railway service, in other words, mainly the SBB. The main idea was to build Hyperloop tubes over the surface exclusively where railroads and motorways are located, as these already belong to the state and do not raise bureaucratic complications. The combination of building on the surface and not having any additional costs means that at these locations routes can be built cheaply. If we have to deviate from existing infrastructural elements, it is assumed that a tunnel has to be built, which costs more. For the passengers' convenience, accelerations in all directions shall be kept under a certain threshold. The path finding is done with the A* algorithm on a precomputed bitmap, containing information only about the railway and motorway networks of Switzerland.

Shortcomings

Due to the complexity of the problem the existing program is still lacking in some areas. The yet not perfect heuristic function, the spline-fitting that aims to create a smooth path out of the inherently discrete information we can obtain from the bitmap and the embryonic cost model all hurt the robustness of the program regarding optimality. Further elaboration of the still present problems can be found in Chapter 5.

1.2 Our Work

We mainly focused on elaborating on the tunnelling aspects of the current algorithm, creating a much more realistic cost model, while also keeping or explicitly making further enhancements easily implementable and scalable. On a sheer theoretical basis we have also tried to come up with some possible starting points for solutions to problems that arise from the way the A* is operating on the current discrete raster.

Chapter 2

Background on Tunnelling

Tunnelling is one of the most challenging areas in civil engineering, due to the high level of complexity involved. It is a multi-dimensional problem, where not only static and mechanical considerations have to be made, but also complex infrastructural and logistic planning is involved. As in any infrastructural project, the main objective is to achieve the highest possible gain in mobility for the amount of money invested.

2.1 Main Aspects

When planning a tunnelling project, various aspects have to be considered, such as the social and economic growth won by the shortened travelling times, the total cost of the project and, of course, the sheer possibility of overcoming the technological challenges.

After the start and end points of a tunnel have been more or less fixed, the next step is to find the exact route, for which a simple straight line is often not feasible, due to the geological condition of the ground, heavy overload (i.e. mountains over loose ground), bureaucratic and other complications caused by built-up areas, and ground/surface water that may enter the tunnel, thus creating a need for a more sophisticated sealing and also making the construction process more challenging.

Generally, tunnels are classified into three groups according to the construction method used: cut-and-cover method, classical methods and tunnels excavated by so called "Tunnel Boring Machines", which we will further refer to as TBMs. By the cut-and-cover method we mean tunnels that are built by digging up the surface, building the desired infrastructure, and then covering it up. Classical tunnel building incorporates basically any proper underground construction, where the advancement is not made by a boring machine, but for example by controlled explosions and mechanical or manual excavations. TBMs are the machines known from huge tunnelling projects, like the Gotthard Base Tunnel, or urban metro projects. Typically with the help of TBMs a higher advancement rate can

be achieved [2], but they are mostly only deployed at larger tunnelling projects, where the planned tunnel is of considerable length, as the use of one requires a significant initial investment. The adequate method for a project is chosen after carefully evaluating the ground conditions and considering the diameter and the length of the proposed tunnel.

Figure 2.1: A TBM in action[1]

2.2 Cost

We are looking at the construction costs per cubic metre, to have an easily scalable measurement of expenses. From the data from Tunnelling Switzerland [3] we can see that the costs of a cubic metre of excavated mass can have huge changes from project to project, some projects costing up to 4 times as much as others (for details see Appendix A).

The cost per cubic metre depends mainly on two factors: on the engineering complexity of the project and the geological circumstances. The engineering challenge posed by the construction of base tunnels requires much more careful and longer planning than some smaller scale projects. Also, a longer projected tunnel increases the level of uncertainty in the planning, as it is not rare that while digging one faces unforeseeable problems caused by the condition of the ground. A radical change in the state of the ground may also lead to the necessity of changing the excavation method, which can easily cause the costs to explode. What one also has to consider when talking about the costs of a project is the

[1] https://www.openpr.com/news/736645/global%2Dand%2Dchinese%2Dtunnel%2Dboring%2Dmachine%2Dtbm%2Dmarket%2D2017%2Dherrenknecht%2Djimt%2Drobbins%2Dcrte%2Dliaoning%2Dcenscience%2Dcrchi%2Dnhi.html, last accessed on 05/01/20

duration of the construction itself. If a project takes longer, apart from the additional costs of labour and equipment, one may face situations where contracts have to be revised, third party companies pull out of the project or go into bankruptcy, or the regulations/political environment change(s).

All the factors mentioned above take their worst form when a project is planned through mountainous terrain. The fact that at some points one has hundreds of metres of ground above one's head causes significant problems. Although Switzerland has one of the most detailed geological data collected, in the mountainous territories a high level of uncertainty still applies, as it is hard to predict or measure the soil deep inside of a mountain. This combined with the fact that when tunnelling through a mountain, the construction is mainly only reachable from the both ends of the planned tunnel, can cause immense increases in costs when a geological discontinuity is reached. The already long construction times can thus be further increased in case of mountainous tunnelling projects, and as such give rise to astronomic costs. Also due to the restricted accessibility of the main tunnel from the surface, higher safety measures have to be taken both during the construction and for the final tunnel, which further increases the amount of the required investment.

Tunnelling under built-up areas can be confronted by additional challenges. When digging under densely urbanised stretches, for example city centers, the challenges posed by reduced accessibility apply again. Moreover, noise pollution during construction, complicated bureaucracy and the need of ensuring the static consistency of the buildings above, all rise the construction costs.

Chapter 3

Modelling and Implementation

The program resulting from a previous project [1] was able to search a shortest path using an implementation of the A* algorithm. The search was conducted on a bitmap consisting of the railway and motorway networks of Switzerland. The data was retrieved from Open Maps [4]. These features were assigned the value 1, marking the possibility of building cheap tubes on the surface. Everything else was assigned the value 0, meaning that, at these pixels, tunnels must be built; thus, accounting for a higher cost. The algorithm then looked for the shortest path based on the following considerations: trying to minimise the travel time by keeping the route as straight as possible, but at the same time also keeping costs low by trying to follow the underlying infrastructural network as closely as possible. A major shortcoming here was that tunnels were built too lightheartedly and at a fixed cost per kilometre, without taking any other environmental factors into consideration.

As we have pointed out in the last chapter, the art of tunnelling is much more complicated than just drawing straight lines on a map. Thus we have focused on incorporating some new features into the path finding that makes the tunnelling aspect of the algorithm more realistic. In this first step towards more realistic tunnelling, we decided to incorporate the lakes and the mountains of Switzerland into the path finding. We have decided to add these features based on the findings mentioned above, namely that we could face multiple complications when tunnelling through mountainous terrain, and building a tunnel under a lake might also result in a significantly harder engineering challenge.

The remainder of the bitmap is still treated as before, meaning that a fixed cost is assigned per kilometre of tunnel. Furthermore, it is assumed that tunnelling at these areas comes at a fixed and easily predictable cost. We will further call these areas ”areas with friendly tunnelling conditions”. As these are the territories of Switzerland outside of the Alps and without the lakes, we assume that the geological state of the ground is predictable at a high accuracy and tunnels are accessible at multiple points even during construction. This allows lower safety costs and faster construction. Even in case of unexpected changes in the condition of the ground, due to the increased accessibility, the thus arising problems

can be addressed much more easily than in the case of tunnels in mountainous terrain. Therefore we have an overall lower, more stable, and fairly easily predictable overall cost. Here it is important to make the following remark: highly urbanised areas constitute an exception, as they can also result in significantly higher (roughly up to three times) construction costs [A].

3.1 Mountains

To model the Alps in Switzerland we again used Open Maps, where we filtered the data looking for ridges and peaks over 1500m a.s.l. As this provided us with point and line data, we only had the pixels that were directly touched by the data points marked. This meant that a mountain was represented as a single pixel on a bitmap. However, we wanted to capture not just the location of the peak, but also roughly the whole volume of the mountain. To achieve this we decided to create a model for an average mountain in Switzerland, based on the previously described data.

For the creation of said model, the main idea was to create a circular area around the peak, where costs linearily decrease outwards. To choose the radius we proceeded as follows. We averaged the elevation of Switzerland's 20 biggest cities by population[1] and the elevation of the lowest point in Switzerland of the major rivers[2] in Switzerland. This resulted in an average elevation of roughly 500m. Next, we subtracted this value from the mean height of the Alps' peaks (namely 2500 metres[3]), resulting in a relative elevation of 2000 metres. This way we could roughly depict how tall a mountain stands in Switzerland when viewed from the ground. In order to avoid replacing potentially friendly areas with mountains we aimed to underestimate the radius. To keep the radius as an underestimation we have decided to base our model on the famously steep Matterhorn. Hence, to determine the radius we set the steepness of our model at 45° to closely match that of the Matterhorn's average steepness [5], resulting in a radius of 2000m.

Although the model incorporates a lot of approximations, the end product resembles the topology of Switzerland surprisingly well, visibly indicating the Swiss valleys' topology (see Figure 4.1).

[1] https://de.wikipedia.org/wiki/Liste_der_St%C3%A4dte_in_der_Schweiz, last accessed on 04/01/20

[2] https://de.wikipedia.org/wiki/Liste_der_Fl%C3%BCsse_in_der_Schweiz, last accessed on 04/01/20

[3] https://en.wikipedia.org/wiki/Alps, last accessed on 07/01/20

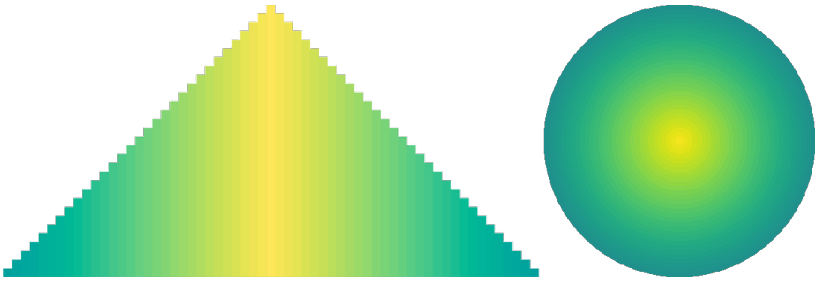


Figure 3.1: Mountain Model: on the left one can see the model from the side, and on the right from the top. The coloring represents the relative elevation of the given pixel. The cost of a pixel is linearily dependent on the elevation.

3.2 Lakes

Building tunnels under lakes, or generally around water resources, increases the engineering complexity and costs significantly. Besides, the sheer depth of a lake poses another problem. Hence, we want to avoid constructing a Hyperloop in such areas.

The major lakes of Switzerland, and generally lakes in the alpine region, tend to be considerably deep. As a comparison, the Channel Tunnel is 115m b.s.l. at its deepest point[4], whereas the deepest lake of Switzerland is 372 metres deep[5]. In order to overcome such depth and yet still not exceed the limitations we have set on vertical acceleration, we have two options: we either approach with a lower speed, resulting in decreased time efficiency, or achieve a smooth transition by starting to build the tunnel far before the lake, thus potentially significantly increasing the distance needed to be covered by tunnels.

To model the lakes we have took the available Open Street Map data for Switzerland's lakes. As a result we were left with almost all standing bodies of water in the country, so we decided to filter the data for a minimal surface area of $1km^2$. After the filtering, only the major lakes were left. We decided to filter the small lakes out, because they were mostly small alpine ponds, which do not influence the tunnelling process, as in these regions a tunnel has to be built hundreds of metres below the surface anyway. We then assigned an exorbitantly high cost to each pixel that coincides with a remaining lake; thus, effectively preventing the algorithm from looking for routes leading through them.

[4] https://en.wikipedia.org/wiki/Channel_Tunnel, last accessed on 04/01/20

[5] https://www.redbull.com/ch-de/die-5-tiefsten-seen-der-schweiz, last accessed on 04/01/20

3.3 Shortcomings of the Model

The imperfections of this approach are relatively obvious. A typical mountain is hardly reducible to a simple cone. We also did not consider the geological data directly, therefore the whole model relies on the assumption that when tunnelling through mountains more geological uncertainty and higher engineering challenges arise. A more accurate approach would have to take the geological map of Switzerland into consideration and associate an additive cost factor for tunnelling wherever mountains and inadequate ground conditions are simultaneously present. We tried to incorporate this into our model, but unfortunately we had no open geological data available to us.

Similarly, an accurate model would incorporate the underwater topology of the lakes and allow for tunnelling in areas with feasible depths. Unfortunately this was once again hindered by the sheer lack of available open data.

The lack of useful data sets also struck us when we were trying to incorporate urban areas into our model. The only city for which we could find the urban zones was Zurich, but our goal was to include at least the 20 biggest cities of Switzerland. Our next idea was to approximate the city limits by looking at only the buildings. For this, there is detailed data available, marking every building's projection onto the ground as a polygon. The problem with this approach was that the density of the buildings was not always consistent with major urban areas. We could also not just look at the density of the buildings and model this statically, as the cost increment comes from the fact of inaccessibility, thus, even though a small village may appear as a densely populated area on a map, it would not hinder our tunnelling as it might only last for just tens of metres. However, urban areas attain only a minor role in the experiments presented in this report, as we did not aim to connect any two locations with major built up areas between them.

To account for these shortcomings and allow for any future improvements, we have modified the existing system to be more flexible, where adding any new features to the path finding can easily be done, allowing for further extensions once appropriate data is available.

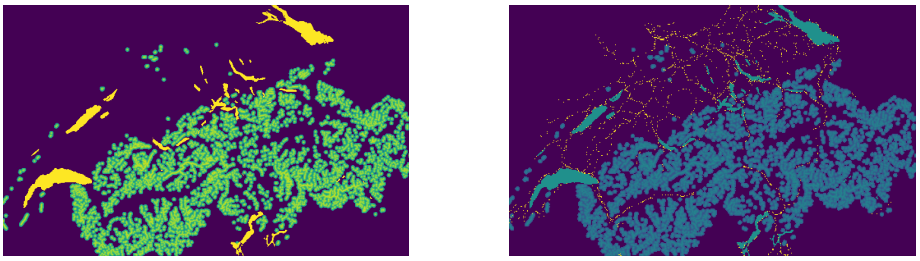
Chapter 4

Results

In this Chapter we show our results on some examples. In order to judge the computed routes, we compare them with the previous routes from the original project. As motorways and railways in mountainous areas are to be found in the valleys already in the previous project, mountains where avoided to some degree. But whenever tunnels had to be built due to the absence of useful motorways and railways, the mountainous terrain was not considered and non-ideal routes were computed. Similarly, lakes were crossed inconsiderately. By our modelling extensions we aimed to address exactly these problematic aspects of the path finding.

We aimed to provide a comparison between the results of the previous algorithm and ours. For this, we recalculated some routes, where the differences are forcefully made obvious. For most of our calculations we chose to connect Sion to Lugano, as these cities are separated by highly mountainous regions and Lago Maggiore, with a peak depth of 372 metres[1]. We used two different setups: first, we did the path finding on a bitmap where the information about motorways and railways was excluded, leaving only lakes and mountains behind. Second, we ran the algorithm on a bitmap where the information of the lakes and mountains was simultaneously present with the infrastructural information. The first setup allowed us to see how efficiently valleys are followed when the algorithm is set to build exclusively under ground. The second is useful for comparisons between routes found by the old algorithm and ours where often more realistic tunnelling solutions are to be observed.

[1] https://www.ascona-locarno.com/en/commons/details/Lago-Maggiore/103496, last accessed on 04/01/20



(a) Setup 1 (b) Setup 2

Figure 4.1: The underlying bitmaps used for the shortest-path search in Setup 1 (only considering underground routes) and Setup 2 (allowing to save on tunnelling costs by following rail- and motorways along the surface).

4.1 Setup 1 - Underground Path Finding

In this section we present the results obtained by a setup where the algorithm was looking for entirely underground routes. To model an environment where tunnelling is considered exclusively, we removed the data of the motorways and railways and only kept the mountains and the lakes.

For pixels where neither a mountain nor a lake is located, we used the same tunnelling cost as in the previous project's "Model A" [1, p.30, Table 4.1], where a cost of $31 Million per kilometre, taken from Hyperloop Alpha [6], is assumed for a tunnel. This parameter is then multiplied by a scaling factor when mountain or lake pixels are hit. To obtain the results presented below, for the mountains we used a set of scaling parameters that are obtained by comparing the most expensive tunnel and the least expensive tunnel in the table given in the Appendix A. This way the scaling factor for the peak of a mountain, the point from where we linearly decrease until 1, amounts to 4.4. In order to avoid lakes, we chose a scaling factor of one hundred, effectively making the selection of one lake pixel equal to 100 pixels of extra Hyperloop under friendly tunnelling conditions. Although this did not entirely forbid the crossing of lakes, it still resulted in the avoidance of lakes.

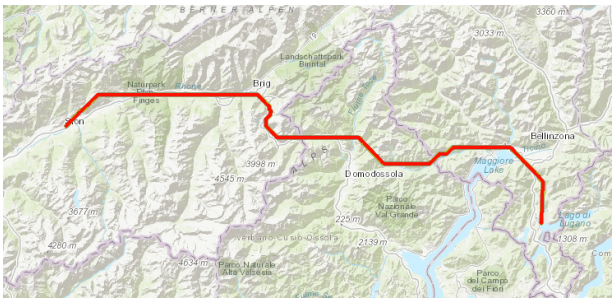


Figure 4.2: Tunnel connecting Sion to Lugano, created with our algorithm. It can be observed that the path approximately follows the natural valleys.

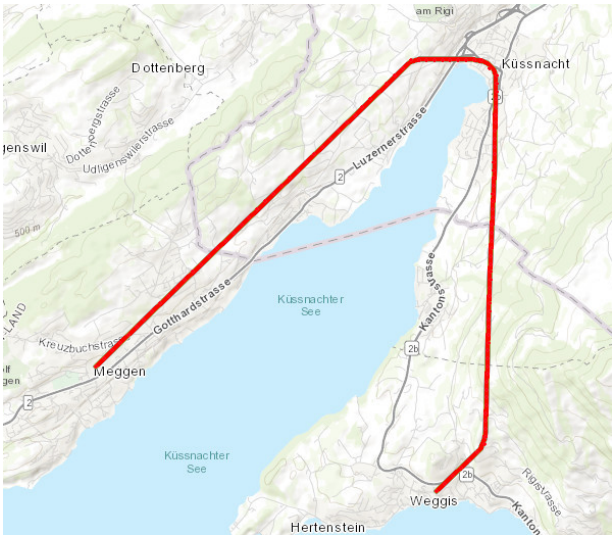


Figure 4.3: Tunnel connecting Meggen to Weggis avoiding crossing lake Lucerne

4.2 Setup 2 - Combined Path Finding

In this setup we added the information about the railways and motorways. This allows the algorithm to consider building options over the surface, as discussed in Chapter 1. For the price of building one kilometre of tubes over rails and motorways, we again took the values from the previous project's "Model A", which again, is based on Hyperloop Alpha. When comparing our results to the results from the previous project, two observations can be made. If the previously generated route already follows the public transportation network tightly, the new route matches the previous one. This is as expected, since no changes were made that would affect the way path finding works above the surface. However, significant differences can be observed when it comes to building tunnels over a

distance. While the old route contains a straight line through the alpine part, after the railway in the Rhone-Valley finishes, it is visible that the new algorithm takes the topology into consideration and creates a tunnel path that roughly follows the valleys along the way.

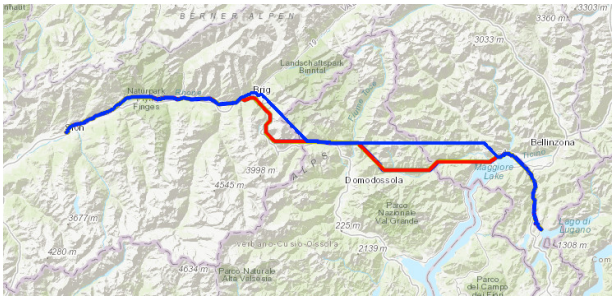


Figure 4.4: Comparison of the routes from Sion to Lugano; the differences of the new route to the old one are marked in red

4.3 Interpretation

In both setups we could observe that the algorithm tried to fit the routes into the valleys. This seems to establish much more believable results, as the tunnelling conditions inside valleys are similar to those of friendly areas. This is due to the fact that in valleys we do not have to deal with the problems of inaccessibility and increased uncertainty; thus allowing for simpler planning, shorter construction time, and generally lower costs.

Chapter 5

Conclusion

We focused on improving the usefulness of the program by improving the underlying cost model. The main finding of our project was the revelation about the sheer complexity of the problem. In order to obtain a realistic model we have seen that the topology, geology and urbanisation data has to be combined, and even with highly detailed data sets assumptions still have to be made, just as in any real tunnelling project. Due to the level of uncertainty that is always connected with a tunnelling project of bigger volume, any cost model will produce results that differ from the real cost as it is not possible to predict all the complications that might occur during the construction phase[1].

5.1 Future Work

Here we share the shortcomings of the project and our ideas for further improvement, additionally to the ones already mentioned in the prior Master Thesis. We will not further elaborate on the flaws of our current cost model as they have been thoroughly discussed in Section 3.3.

5.1.1 Time Complexity

One major issue of the current implementation is the time it takes for the computations. The run-time grows quadratically with the distance of the points to be connected. At a resolution of 10 metres the computation takes several days (conducted on one 2GHz clock frequency core of a virtual machine with 128 GB RAM). Also, the underlying bitmap of the said resolution has a size of 609 Megapixels, which after being converted to float for the path finding, is roughly 4.8GB in size, straining the memory of the used machine. For these reasons, we decided to conduct all our calculations in a 100x100m setting, which allowed us to experiment more with different parameter settings, at the cost of decreased

[1] According to a discussion with Mr. Thomas Pferdekämper, PhD Candidate at the Chair of Underground Construction at the Swiss Federal Institute of Technology in Zurich

accuracy. In order to overcome the problem posed by the high time complexity of the program, parallel programming could be used. Speed-up can also be achieved by improving the heuristic function, although it has to be made sure that it is upper-bounded by the actual costs such that the correctness of the algorithm is ensured. Nevertheless, the running time only plays a significant role if the program is meant to be used in real-time. Thus, if one aims to provide a basis for an actual Hyperloop construction project, real-time usability does not play a huge role. However, for the website introduced in the underlying Master Thesis, a future improvement on the running time is unavoidable.

5.1.2 Ensuring Optimality

The currently used heuristic function can be improved further and lacks a formal proof. Although we searched for corner-cases where the current heuristic function fails, we could not find such an example.

As mentioned in the prior project [1, Section 4.3.3], the way the nodes are defined currently can produce cases where the optimal path is missed. Ideally one pixel should correspond to more nodes in the produced graph in which the path finding is conducted; these nodes also should incorporate the arrival angle. However this is hardly feasible, because it means that each pixel would correspond to a continuous set of nodes with all possible arrival angles. Thus a solution where certain arrival angle intervals are represented by one node could be considered, this would lend more flexibility while still allowing for discretisation.

Another problem that arises from the current nature of the pixels can be demonstrated by the following example. We attempted to route a tunnel from Sion to Lugano on a bitmap where every pixel has the same cost, and set the smoothing parameter to 0, not allowing for post processing. In Figure 5.1, we can see the result of the attempt; however, this route is clearly not optimal. This problem arises from the fact that with the current discretisation and spline fitting methods, straight lines can only be created by the algorithm if they are either vertical, horizontal or tilted at exactly 45°. In any other case splines are fitted such that they produce unnecessary curves, thus slowing down the pod and making the algorithm perceive this option as sub-optimal. However an enhanced real-time smoothing or spline fitting could straighten the route during the search, hence allowing for straight-line routing. We elaborate more on this in the next subsection.

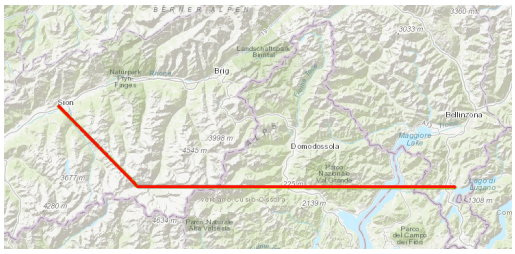


Figure 5.1: Route from Sion to Lugano; it can be observed that instead of a direct straight line the algorithm returns a combination of a 45° and a horizontal line as the optimal path

5.1.3 Smoothing and Post-processing

The currently used spline fitting, smoothing and route post-processing causes problems regarding optimality. As already mentioned in the prior Master Thesis [1, Section 4.3] the current tunnel post processing does not ensure that after the tunnels have been straightened and joined the resulting route is still optimal regarding cost over time won.

As the post-processing is optional, this does not pose a big problem regarding the core of the algorithm. More problematic is the fact already mentioned in Subsection 5.1.2. This actually influences the path finding during its execution in real time, which can lead to results containing major deviations from the optimal path. To fix this issue, the spline fitting can be reviewed and the smoothing enhanced. However working with splines will always mean that we are only looking for smoothness, but are not explicitly trying to minimise curvature. Alternatively, we thought about the possibility of interpreting the posed problem as finding a racing line on the "racetrack" that is formed by converting the pixels making up the current path into one continuous set. Finding the racing line on a racetrack is generally a hard problem, and incorporates the optimisation of the conflicting attributes against each other; we want to cover the shortest distance possible, while still trying to maximise velocity, thus minimising curvature [7]. To find or at least approximate the ideal racing line several different algorithms and techniques are used; in simpler racing games some heuristics are developed to find an almost ideal path automatically, whereas more advanced solutions include the use of machine learning techniques, mostly reinforcement learning, and genetic algorithms.

However, with the proposed method one faces issues regarding run-time. To recalculate the optimal racing line for each newly opened node will drastically increase the time-complexity of the algorithm. At the current stage, running times of a few seconds were achieved [8], which is still not satisfactory for our purposes. This means that this idea is still something to explore on, and offers the possibility of some exciting results and findings.

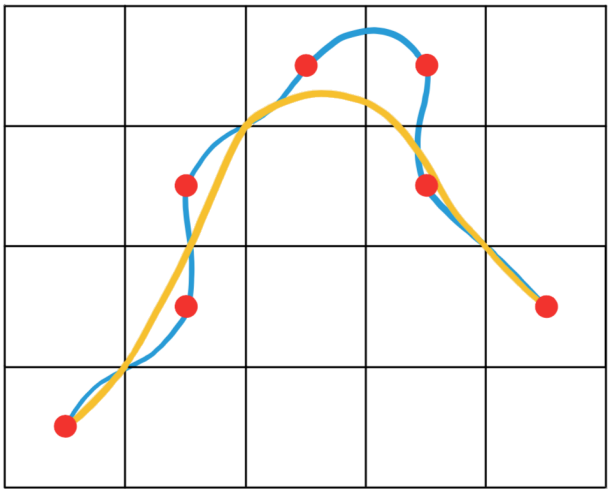


Figure 5.2: A rough sketch to compare splines fitted with zero deviation (marked in blue) and the approximation of the ideal racing line (marked in orange). The orange path aims to achieve minimal curvature while also keeping the distance as short as possible.

Bibliography

[1] S. Jeker, "Hyperloop Network Design," Feb. 2019.

[2] "Annex 13 – Case Study on Tunnels," https://ec.europa.eu/regional_policy/sources/docgener/studies/pdf/assess_unit_cost_rail/annex_13_case_study_tunneling.pdf, last accessed on 04/01/19.

[3] G. Anagnostou and H. Ehrbar, *Tunnelling Switzerland*. vdf Hochschulverlag ETH Zurich, 2013.

[4] "Openstreetmap," https://wiki.openstreetmap.org/, last accessed on 10/12/19.

[5] *Climbing the Matterhorn - A Collection of Historical Mountaineering Articles on the Brave Attempts to Scale One of the Highest Peaks in the Alps*. Read Books Limited, 2016. [Online]. Available: https://books.google.ch/books?id=guEmDQAAQBAJ

[6] E. Musk, "Hyperloop Alpha," 2013.

[7] L. Cardamone, D. Loiacono, P. L. Lanzi, and A. P. Bardelli, "Searching for the Optimal Racing Line Using Genetic Algorithms," 09 2010, pp. 388 – 394.

[8] "Genetic Programming for Racing Line Optimization - part 1," https://medium.com/adventures-in-autonomous-vehicles/genetic-programming-for-racing-line-optimization-part-1-e563c606e502, last accessed on 25/12/19.

Appendix A

Tunnelling Cost Table

TUNNEL	TYPE	TOPOLOGY	GEOLOGY	EXCAVATION	COSTS CHF	LENGTH m	COSTS CHF/m	PROFILE m^2	COSTS CHF/m^3
Lötschberg Base Tunnel	Railway Tunnel	Base Tunnel	Mixed	Mixed	4'300'000'000	88'000	49'000	68	720
Gotthard Base Tunnel	Railway Tunnel	Base Tunnel	Mixed	Mixed	11'300'000'000	114'000	99'000	66	1'500
Ceneri Base Tunnel	Railway Tunnel	Base Tunnel	Mixed	Conventional	2'500'000'000	30'800	81'000	68	1'190
Weinberg Tunnel	Railway Tunnel	City	Mixed	TBM	480'000'000	9'600	50'000	99	510
Zimmerberg Tunnel	Railway Tunnel	City	Mixed	TBM	630'000'000	12'400	51'000	99	510
Engelberg Tunnel	Railway Tunnel	Mountain	Rock	Conventional	98'300'000	4'060	24'000	27	900
Almend Tunnel	Railway Tunnel	City	Soil	Conventional	30'000'000	900	33'000	68	490
Pinchat Tunnel	Railway Tunnel	City	Soil	Conventional	135'000'000	2'024	67'000	65	1'030
Tunnel de Champel	Railway Tunnel	City	Soil	Conventional	115'000'000	1'631	71'000	62	1'140
Luzernerring Tunnel	Road Tunnel	City	Soil	Conventional	45'000'000	383	117'000	95	1'240
Hafnerberg Tunnel	Road Tunnel	City	Soil	Conventional	155'000'000	1'370	113'000	180	630
Aeschertunnel	Road Tunnel	City	Soil	Conventional	401'000'000	2'150	187'000	180	1'040
Uetlibergtunnel	Road Tunnel	Mountain	Mixed	Mixed	1'169'000'000	8'800	133'000	163	810
Islisbergtunnel	Road Tunnel	Mountain	Rock	TBM	370'000'000	9'900	37'000	110	340
Tunnel Flüelen	Road Tunnel	Mountain	Rock	TBM	190'000'000	2'596	73'000	114	640
Birchi Tunnel	Road Tunnel	City	Soil	Conventional	122'000'000	1'400	87'000	100	870
Längholz Tunnel	Road Tunnel	City	Mixed	TBM	245'000'000	4'980	49'000	126	390
Butterberg Tunnel	Road Tunnel	Mountain	Rock	TBM	245'000'000	2'920	84'000	125	670
Serriéres Tunnel	Road Tunnel	City	Rock	Conventional	135'000'000	1'602	84'000	104	810
Concise Tunnel	Road Tunnel	Mountain	Rock	Conventional	115'000'000	2'750	42'000	102	410
Lance Tunnel	Road Tunnel	Mountain	Rock	Conventional	22'000'000	417	53'000	102	520
Tunnel Lungern	Road Tunnel	Mountain	Rock	Conventional	265'000'000	3'574	74'000	100	740
Tunnel Giswil	Road Tunnel	Mountain	Rock	Conventional	100'000'000	2'100	48'000	100	480
Tunnel Sachseln	Road Tunnel	City	Rock	TBM	294'000'000	5'750	51'000	100	510
Südumfahrung Visp	Road Tunnel	Mountain	Rock	Conventional	1'200'000'000	13'800	87'000	120	720
Tunnel Eyholz	Road Tunnel	Mountain	Mixed	Conventional	400'000'000	8'400	48'000	120	400
San Fedele Tunnel	Road Tunnel	Mountain	Rock	Conventional	189'000'000	2'391	79'000	140	560
Buchrain Tunnel	Road Tunnel	Mountain	Rock	Conventional	28'000'000	740	38'000	100	380
Tunnel Sous le Mont	Road Tunnel	Mountain	Rock	Conventional	113'000'000	1'200	94'000	115	820
Tunnel du Graitery	Road Tunnel	Mountain	Mixed	Conventional	130'000'000	2'450	53'000	90	590
Moutier Tunnel	Road Tunnel	Mountain	Mixed	Conventional	300'000'000	2'389	126'000	107	1'170
Raimeux Tunnel	Road Tunnel	Mountain	Rock	Conventional	150'000'000	3'211	47'000	100	470
Chindez Tunnel	Road Tunnel	Mountain	Mixed	Conventional	180'000'000	3'288	55'000	125	440
Montaigre Tunnel	Road Tunnel	Mountain	Soil	Conventional	76'000'000	1'728	44'000	104	420
Bure Tunnel	Road Tunnel	Mountain	Rock	TBM	135'000'000	3'100	44'000	124	350
Neu-Bois Tunnel	Road Tunnel	Mountain	Rock	Conventional	60'000'000	1'900	32'000	92	340
Küblis Tunnel	Road Tunnel	Mountain	Mixed	Conventional	128'000'000	2'255	57'000	110	520
Saaser Tunnel	Road Tunnel	Mountain	Mixed	Conventional	116'200'000	2'577	45'000	100	450
Gotschnatunnel Tunnel	Road Tunnel	Mountain	Rock	Conventional	211'000'000	4'207	50'000	100	500
Hausmatt Tunnel	Road Tunnel	City	Soil	Conventional	58'000'000	514	113'000	120	940
Flimsterstein Tunnel	Road Tunnel	Mountain	Mixed	Conventional	295'000'000	4'500	66'000	100	660
Vedeggio-Cassarate Tunnel	Road Tunnel	City	Mixed	Conventional	150'000'000	2'630	57'000	120	480
Chienberg Tunnel	Road Tunnel	City	Mixed	Conventional	250'000'000	2'200	114'000	113	1'010
Wihaldentunnel Tunnel	Road Tunnel	City	Soil	Conventional	40'000'000	510	78'000	95	830
Rotwald Tunnel	Road Tunnel	City	Soil	Conventional	20'000'000	385	52'000	100	520
Palatinat Tunnel	Road Tunnel	City	Soil	Conventional	7'000'000	80	88'000	160	550
Tunnel de Grand-Saint-Bernard	Safety Tunnel	Mountain	Rock	TBM	56'000'000	5'774	10'000	14	690

Figure A.1: The table was created by Mr. Pferdekämper from the book Tunnelling Switzerland [3].