Path Advisor: A Multi-Functional Campus Map Tool for Shortest Path

Yinzhao Yan
The Hong Kong University of Science and Technology
yyanas@cse.ust.hk

Raymond Chi-Wing Wong
The Hong Kong University of Science and Technology
raywong@cse.ust.hk

ABSTRACT

The shortest path in both the two dimensional (2D) plane and the three dimensional (3D) terrain is extensively used both in industry and academia. Although there are some map visualization tools for viewing the shortest path in 2D and 3D views, we find two limitations: (1) they are not applicable for map applications with obstacles (such as the wall in a building), and (2) they look unrealistic and strange when a road network approach is blindly adopted. Motivated by this, we developed a web-based multi-functional campus map tool called *Path Advisor*, which allows users to visualize the shortest path in the 2D view, the bird's eye view and the virtual reality view (VR view). Path Advisor uses Dijkstra's shortest path algorithm and breadth-first tree in the 2D view, and the weighted shortest surface path algorithm in the bird's eye view and the VR view. We shot a video for demonstrating Path Advisor at https://youtu.be/ZgdjyXXHwqg.

PVLDB Reference Format:
Yinzhao Yan and Raymond Chi-Wing Wong. Path Advisor: A Multi-Functional Campus Map Tool for Shortest Path. PVLDB, 14(12): 2683-2686, 2021.
doi:10.14778/3476311.3476319

1 INTRODUCTION

The shortest path between a source vertex and a destination vertex in both the 2D plane and the 3D terrain is important in the database community [5, 8] and is extensively used in map applications. It could be monotonous and tedious when the shortest path on a 2D plane map is viewed. But, visualizing the shortest path in the *bird's eye view* and the *virtual reality view* (VR view) could highly improve the map's fun and allow the user to interact with the map more intuitively.

The *bird's eye view* of a 3D object refers to the elevated view of the object from above and the observer is like a bird, which is increasingly popular in maps and floor plans. Besides, the *VR view* is a simulated view that is similar to the real world, and the user could look around, move and interact with the virtual world.

Although there are different visualization tools for 3D shortest path using spatial data [1, 7], none of them are applicable for map applications with obstacles, which means that all the shortest paths are restricted in the presence of some obstacles (such as the wall in a building) and these paths cannot cross such obstacles. In addition, there are also some 3D indoor shortest path visualization tools using a fixed road network for shortest path calculation [4, 6], but the straight and abrupt path calculated by the predefined road network makes the visualization unrealistic and strange. (Note that the term *road network* can be regarded as a *graph network*, we use *road network* simply because it is a common term in spatial database.) Hereby, we set two requirements so that the path should follow: (1) the path should not be too close to the obstacle (*e.g.*, the distance between the path to the obstacle should be at least 0.2 meter), and (2) the path should not have sudden direction changes. Figure 1 (a) shows the *road network* shortest path calculation result (shown in red) and Figure 1 (b) shows the weighted shortest surface path calculation result using *spatial data* (the method used in our system) (shown in green). In these figures, the green line looks more realistic than the red line (because the red line violates the first requirement). Note that there are some other existing methods returning their paths (to be elaborated later in this paper) as shown in Figure 1 (c) and Figure 1 (d), which does not look natural because they violate the first requirement and the second requirement, respectively.

Motivated by the limitations of 3D shortest path visualization tools' obstacle issue and the unrealistic display of the road network on a 3D geometric model, in this paper, we developed a web-based 2D view, bird's eye view and VR view campus map tool for shortest path called *Path Advisor*, which allows users to find and visualize the shortest path between any two rooms (or buildings) in The Hong Kong University of Science and Technology (HKUST) campus. Note that when finding a path between two rooms in two separate buildings, it returns a concrete path involves both the indoor path and the outdoor path. Figure 2 (a) illustrates the 2D view, Figure 2 (b) illustrates the bird's eye view, and Figure 2 (c) and (d) illustrates the VR view at two places in HKUST, namely atrium and lift 25/26, respectively. Besides, we propose to model a 3D model with *weights* and to use the *weighted* shortest surface path calculation algorithm to meet the two requirements as mentioned above. To the best of our knowledge, we are the first to use the weighted shortest surface path calculation algorithm in the presence of obstacles, which could satisfy the above two requirements (this is also our major contribution). None of the existing tools could satisfy these requirements.

We have conducted a user study and more than 97% of users (out of 50 participants) agree that the bird's eye view and the VR view are better than a simple planar map, and could enhance the user experience. There are also over 95% of users rating their satisfactory level of the bird's eye view and the VR view at 4 and 5 scores (the highest satisfactory level is 5 scores), respectively. In addition, more than 60% of users agree that the path shown in Figure 1 (b) (the path used in our system) is more realistic than the other paths shown in Figure 1 (a), (c) and (d).

Proceedings of the VLDB Endowment, Vol. 14, No. 12 ISSN 2150-8097.
doi:10.14778/3476311.3476319

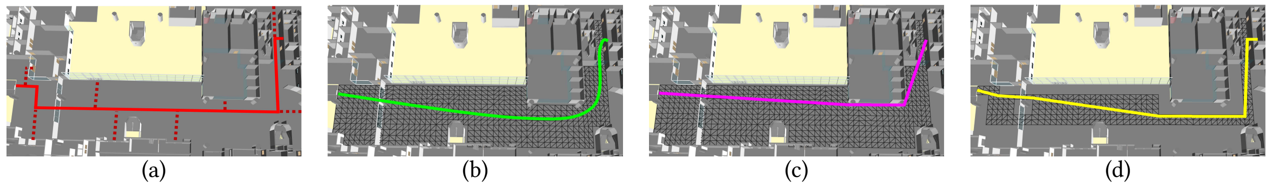

Figure 1: (a) Road network shortest path calculation result (the red line is the calculated result and the dark red dot line is the predefined road network), (b) weighted shortest surface path calculation result (the green line is the calculated result and the black mesh is the spatial surface for computing the weighted shortest surface path), (c) unweighted shortest surface path calculation result (the purple line is the calculated result and the black mesh is the spatial surface for computing the unweighted shortest surface path), and (d) grown obstacle shortest surface path calculation result (the yellow line is the calculated result and the black mesh is the grown obstacle spatial surface for computing the shortest surface path)

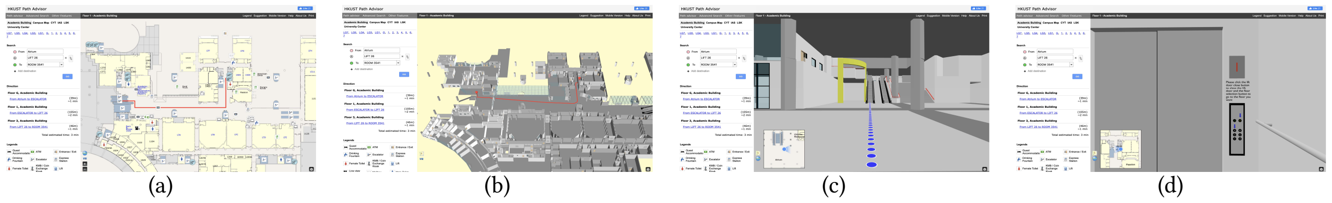

Figure 2: Demonstration of (a) the 2D view, (b) the bird's eye view, (c) the VR view at atrium (one place in HKUST) and (d) the VR view at lift 25/26 (one place in HKUST)

In the remainder of this paper, we introduce the design of Path Advisor in Section 2, show the demonstration of Path Advisor in Section 3, and conclude with future developments in Section 4.

2 DESIGN

In this section, we provide the background knowledge about terrain in Section 2.1, illustrate the architecture of Path Advisor in Section 2.2, and show three main components of Path Advisor, namely the *2D view*, the *bird's eye view* and the *VR view* in Section 2.3, 2.4 and 2.5, respectively.

2.1 Background Knowledge

A *Digital Elevation Model* (DEM) refers to a terrain's 3D geometric representation. Generally, a DEM is stored in *Object File Format* (OFF), which represents the geometric structure of a model by describing the polygons on the surface of the object, and contains the *vertex*, *edge* and *face* information of a given 3D object.

A DEM can be represented as a *Triangular Irregular Network* (TIN) and it can construct a *Triangulated Graph*, which is a triangulated grid and it has no simple cycles of size four or more. Figure 3 shows an example of TIN model. In this figure, $v_1, v_2, ..., v_9$ are *vertices*, and (v_1, v_2), (v_1, v_4) and (v_2, v_4) are examples of *edges*. $\triangle v_1v_2v_4$ and $\triangle v_2v_4v_5$ are examples of *faces* (note that a face in TIN is a triangle). In addition, $< v_1, p_1, v_5, p_2, v_9 >$ is one possible surface path from a source point $s (= v_1)$ to a destination point $t (= v_9)$ in this TIN model (p_1 and p_2 are the intersection points between the surface path and the terrain's edge), which passes through four faces.

The *unweighted shortest surface path* on a terrain refers to the shortest path between the source point s and the destination point t that passes the face on the terrain, and the *weighted shortest surface path* refers to the shortest path between s and t where each face has a *different* weight. Figure 5 (a) and (b) shows the unweighted and weighted shortest surface path between two red points. In Figure 5 (b), the terrain faces near the boundary have a larger weight (represented by darker color), so the path will be forced to include the face with a smaller weight. So, compared with Figure 5 (a), it could stay away from the terrain boundary and it becomes smoother.

2.2 Architecture of Path Advisor

The architecture of Path Advisor is shown in Figure 4. There are three major components in Path Advisor, which are the *2D view*, the *bird's eye view* and the *VR view*, respectively.

Path Advisor is a web-based campus map tool, which contains the database, front-end, and back-end parts. The 2D floor plan image (stored in PNG file format), 3D building model (stored in OBJ file format), and spatial data (stored in OFF file format) are stored in the database. The web interfaces are implemented in the front-end and the algorithm for finding the shortest path is implemented in the back-end.

The 2D floor plan image shows the detailed room locations used for the *2D view* shortest path. The 3D building model and spatial data contain the building/floor geometric model and spatial information for the usage of displaying the *bird's eye view* and the *VR view*, and for calculating the shortest surface path, respectively.

2.3 2D View

Calculating the shortest path in the 2D view is implemented by *Dijkstra's shortest path algorithm* together with a breadth-first tree in the predefined road network. Prior to the calculation, we need to

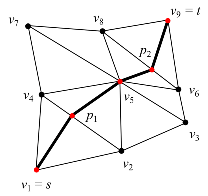

Figure 3: An example of TIN model

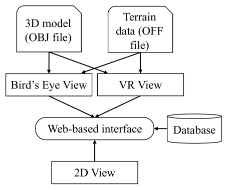

Figure 4: Architecture of Path Advisor

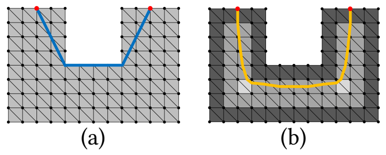

Figure 5: (a) Unweighted shortest surface path, and (b) weighted shortest surface path

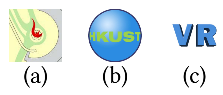

Figure 6: Icon for switching to (a) the 2D view, (b) the bird's eye view and (c) the VR view

build the road network on the map first (where a portion of the road network is shown in Figure 2 (a)). Due to the page limit, we will not go through the details of this algorithm because it is commonly used in map applications.

2.4 Bird's Eye View

The calculation of the bird's eye view shortest path involves two stages, namely the *unweighted* shortest surface path calculation and the *weighted* shortest surface path calculation (which were originally designed for outdoor data but could also be used inside a building). The spatial data in a building is slightly different from other 3D geographic spatial data. For example, there are turns and corners for the floor spatial data. We would like the calculated shortest path not to stick to the boundary of the aisle in a building when the shortest paths are restricted in the presence of an obstacle (such as the wall on both sides of the aisle boundary in a building) and these paths cannot cross such the obstacles. We also want the path to be smooth and realistic. So we set two requirements so that the path should follow: (1) the path should not be too close to the obstacle (*e.g.*, the distance between the path to the obstacle should be at least 0.2 meter), and (2) the path should not have sudden direction changes. If we only use the *unweighted* shortest surface path for calculation, then the path would stick to the aisle boundary. For example, Figure 1 (c) shows the unweighted shortest surface path calculation result. The path is sticky to the aisle boundary (*i.e.* wall) when there is a corner, which looks unrealistic (since usually, humans do not walk like this). Nevertheless, Figure 1 (b) illustrates the weighted shortest surface path calculation result, in which, when there is a corner, the path can avoid sticking to the aisle boundary, which is closer to the real path that humans will walk. Motivated by this, we applied the *weight* idea (to be illustrated later) to the shortest surface path calculation algorithm.

Since the calculation of the weighted shortest surface path is based on the unweighted shortest surface path, we will first introduce the unweighted shortest surface path calculation, and then introduce the weighted shortest surface path calculation.

2.4.1 Unweighted Shortest Surface Path Calculation. The computation of the unweighted shortest surface path between a source point s and a destination point t is one basic and essential operation in the applications of spatial database, online 3D virtual game, and geographic information systems (GIS).

The *exact* unweighted shortest surface path calculation algorithm was first raised by Chen and Han (CH) [2]. The algorithm has $O(n^2)$ worst-case time complexity, where n represents the total number of points in the 3D geographic surface. The *planar unfolding* is applied in the CH algorithm, which rotates all the faces on the surface into the same level plane. Instead of using the *Dijkstra* technique, the CH algorithm calculates the minimum distance of the *projections* from the source point s on each edge/face to each *visible* points and stores the shortest path in a *sequence tree*. Nevertheless, the sequence tree involves a tremendous number of unnecessary points, which highly affects the algorithm performance, whereas an improved CH algorithm raised by Xin et al. [9] could minimize the size of unnecessary points to achieve better performance. We applied this improved algorithm in our map tool.

2.4.2 Weighted Shortest Surface Path Calculation. The difference between the unweighted shortest surface path calculation and the weighted shortest surface path calculation is that the latter one has a weight-distance function to be applied to each face of the spatial data. We introduce this weight-distance function as follows.

Let $d(p, q)$ be the surface path between point p and q on a face f_i, and we assign a weight w_i to face f_i in the floor spatial information. We define $D(p, q) = w_i \cdot d(p, q)$ to be the weighted surface path from p to q on face f_i. For each triangulated face f_i, we denote the smallest distance between this face and the obstacle (the boundary of the aisle, *i.e.* wall) as DTO_i (Distance To Obstacle). We also denote DTO_{max} as the maximum DTO among all the faces. Then, the weighted surface will be used to calculate the weighted shortest surface path using the improved CH algorithm. When the face f_i is closer to the boundary of aisle in a building, the weight w_i approaches ∞, whereas when the face f_i closer to the aisle center, the weight w_i approaches 1. Specifically, we have the following weight-distance function for each face f_i:

$$w_i = \begin{cases} \infty & \text{if } DTO_i = 0 \\ \frac{DTO_{max}}{DTO_i} & \text{otherwise} \end{cases} \tag{1}$$

The principle for Equation 1 is that we would like the weight w_i for face f_i to be inversely proportional to its distance to obstacle (DTO_i). By applying the *weighted shortest surface path calculation*, the surface path could avoid sticking to the aisle boundary.

In addition, we would like to highlight that we used the *whole* floor spatial surface for calculating the weighted shortest path on the 3D geometric model instead of using *partial* floor spatial surface (using the *grown obstacle* method to calculate this partial spatial surface [3]) for calculating the shortest path. The *grown obstacle* method virtually enlarges the obstacles so that the calculated shortest path result could maintain a tolerance distance from the obstacles. In other words, virtually enlarging the obstacles means narrowing the floor spatial surface, *i.e.* using a part of the floor spatial surface which is far away from the obstacles (such as the wall in a building) to replace the whole floor spatial surface in the shortest path calculation. For example, in Figure 1 (b), the green line shows the calculated shortest path result using the *whole* spatial surface and the black mesh is the *whole* spatial surface for calculating the weighted shortest path. While in Figure 1 (d), the yellow line shows the calculated shortest path result using the *partial* spatial surface and the black mesh is the *partial* spatial surface for calculating the shortest path. It clearly shows that the green line looks more realistic than the yellow line (since the yellow line involves several *sudden* angle/direction changes). Note that the smoother path generated by our algorithm is achieved implicitly when the faces are assigned with different weights based on their DTO.

2.5 VR View

The VR view is a simulated view that is similar to the real world, so we have created the 3D geographic model for all the buildings on the campus. We used *WebGL* (or so-called *ThreeJS*) to simulate the first-person view by changing the camera position to implement this feature. The user could use the keyboard to move and mouse to interact with the virtual world, where *Raycaster* is used.

3 DEMONSTRATION

In this section, we describe the user interfaces of Path Advisor in Section 3.1 and the main features of Path Advisor in Section 3.2.

3.1 User Interfaces

In Path Advisor, the center area shows the map details and routine details in the 2D view, the bird's eye view and the VR view, whereas the default mode is the 2D view. Figure 2 (a) illustrates the 2D view, Figure 2 (b) illustrates the bird's eye view, and Figure 2 (c) and (d) illustrates the VR view at two places in HKUST, namely atrium and lift 25/26, respectively. The user could switch among different views and zoom in/out of the map by clicking the "+" button or the "−" button at the left-bottom corner of the center area. The left panel allows the user to input the source point and destination point. The user can use the top bar in the left panel to change to different floors or buildings in HKUST, and choose the advanced search so that the shortest path is step-free access or the shortest time/distance path.

3.2 Main Features

Due to the page limit, we demonstrate the following three key features of Path Advisor only.

3.2.1 2D View. There are four steps for viewing the shortest path in the 2D view. First, the user could click the button as shown in Figure 6 (a) to change to the 2D view. Second, the user could type and choose the source (*from*) point and the destination (*to*) point in the left panel search area. The user can also click "Add destination" to add pass by (*via*) point. Third, the user could click the "GO" button to view the shortest path in the 2D view at the center area. The detailed routine will also be shown in text with the estimated time and distance in the left panel. Fourth, the user could click the endpoint of the shortest path or the detailed routine text to change floor. Note that the user can also drag the calculated shortest path to add pass by (*via*) point. The query time for calculating the shortest path in the 2D view is less than 0.05 second.

3.2.2 Bird's Eye View. There are four steps for viewing the shortest path in the bird's eye view. Basically, the last three steps are similar to the steps mentioned in Section 3.2.1. First, the user could click the button as shown in Figure 6 (b) to change to the bird's eye view. Then, the user could follow the three steps mentioned in Section 3.2.1 to view the weighted shortest surface path with the 3D floor geographic model at the center area. Changing floor could be done by clicking the red pin (shown along the path) or the detailed routine text (shown in the left panel). The query time for calculating the shortest path in the bird's eye view is less than 0.1 second.

3.2.3 VR View. There are four steps for viewing the shortest path in the VR view, which are still similar to the steps mentioned in Section 3.2.1. First, the user could click the button as shown in Figure 6 (c) to change to the VR view. Then, the user could follow the three steps mentioned in Section 3.2.1 to view the shortest path displayed as the blue dot line in the VR view. Next, the user can use a keyboard to move in the VR view or a mouse to interact with the objects in the VR view (*i.e.* open the lift door, click the lift panel number button to change floor, etc.). The user can also use the thumbnail map (shown at the bottom-left corner of the VR view) to view the current position to avoid getting lost.

4 CONCLUSION

Motivated by the limitations of 3D shortest path visualization tools' obstacle issue and the unrealistic display of the road network on the 3D geometric model, we developed Path Advisor. We demonstrate the techniques used in Path Advisor to visualize the shortest path in the 2D view, the bird's eye view and the VR view. The future work on this demonstration proposal could be that after the user has chosen the source point and destination point, a video simulating the VR view along the shortest path will be generated automatically.

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for their constructive comments on this paper. The research is supported by HKRGC GRF 16214017.

REFERENCES

[1] Prosenjit Bose, Anil Maheshwari, Chang Shu, and Stefanie Wuhrer. 2011. A survey of geodesic paths on 3D surfaces. *Computational Geometry* 44, 9 (2011), 486 – 498.
[2] Jindong Chen and Yijie Han. 1990. Shortest Paths on a Polyhedron. In *SOCG*. New York, NY, USA, 360–369.
[3] Tomás Lozano-Pérez and Michael A. Wesley. 1979. An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles. *Commun. ACM*, 560–570.
[4] Shi Pu and Sisi Zlatanova. 2005. *Evacuation Route Calculation of Inner Buildings.* Berlin, Heidelberg, 1143–1161.
[5] Jagan Sankaranarayanan and Hanan Samet. 2009. Distance Oracles for Spatial Networks. In *ICDE*. 652–663.
[6] Muhamad Uznir Ujang and Alias Abdul Rahman. 2008. Implementing shortest path calculation for 3D navigation system. In *Advances toward 3D GIS*. 153–164.
[7] Qi Wang, Manohar Kaul, Cheng Long, and Raymond Chi-Wing Wong. 2014. Terrain-Toolkit: A multi-functional tool for terrain data. In *VLDB*, Vol. 7. 1645–1648.
[8] Victor Junqiu Wei, Raymond Chi-Wing Wong, Cheng Long, and David M. Mount. 2017. Distance Oracle on Terrain Surface. In *SIGMOD/PODS'17*. New York, NY, USA, 1211–1226.
[9] Shi-Qing Xin, Dao Quynh, Xiang Ying, and Ying He. 2012. A Global Algorithm to Compute Defect-Tolerant Geodesic Distance. In *SIGGRAPH Asia 2012*. New York, NY, USA.