Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 221 (2023) 346–353

Procedia Computer Science

www.elsevier.com/locate/procedia

Information Technology and Quantitative Management (ITQM 2023)

Shortest path graphs for semi-directed InDoor navigation

Tatiana Babicheva[a], Dmitry Gushchin[b], Nikita Sopochkin[c]

[a]*RATP Smart Systems, 54 quai de la Rapée, 75012 Paris, France*
[b]*Moscow Institute of Physics and Technology, 9 Institutskiy per., Dolgoprudny, 141701, Russia*
[c]*Babson College, 231 Forest St, Babson Park, MA 02457, United States*

Abstract

In this article, we explore methods for finding shortest paths for indoor navigation. We focus on the construction of the roadmap and thus, the graph representation of the public transport station's open space. Likewise, we propose a method which takes into account the possible directions of corridors. The resulting graph is generally smaller that the graph obtained using only bidirectional entry points, which helps to diminish memory consumption in big networks as well as to speed-up the calculation of the shortest path in the resulting network. Reducing the graph of connections will thus speed up the work of navigation algorithms.

© 2023 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the Tenth International Conference on Information Technology and Quantitative Management

Keywords: Indoor Navigation, Visibility Graph, Shortest Path

1. Introduction

Mobility is one of the key issues in prospective smart city development [1]. Transport is ever-changing, according to the requirements of the time and the trend towards digitalization. The related changes in transportation are referred to as "smart mobility".

One of the key tasks of the city authorities is the introduction of technological solutions that allow for the creation of a reserve in the capacity of the transport infrastructure with regards to the growing load. Keeping this in mind, it is important not only to modernize and build new elements of the transport ecosystem, but also to effectively utilize the existing ones. There is a big difference between the mere presence of smart systems in cities, and their effective joint operation.

Comfort is important for the average city dweller. For example, it is crucial to be able to plan a route with convenient transfers from one mode of transport to another while also being aware of the exact travel time.

This article addresses the issue of facilitating the use of public transport and station information for all traveler profiles. The *problem setting* can be describes as follows: Given the public transport station with its existing geometry and facilities, how can we provide the customer with an efficient path?

Simply, by solving the pathfinding problem. All algorithms build a navigation graph, but none of the available algorithms takes into account the fact that it is better to cross some pedestrian spaces rather than move along their sides.

*Corresponding author. Tel.: +33-754-43-7467.
E-mail address: tatiana.babicheva@ratpsmartsystems.com.

1877-0509 © 2023 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the Tenth International Conference on Information Technology and Quantitative Management
10.1016/j.procs.2023.07.047

Modern pathfinding algorithms are based on a connection graph, which is often a bottleneck when calculating the shortest route. To improve upon these methods, we focused on simplifying the graph by entering both entry-type and exit-type incoming vertices into it.

The remainder of the paper is organised as follows. In Section 2 we provide an overview of previous research about indoor and plaza navigation. In Section 3 we describe the indoor geometry and how me model it, and afterwards formulate how we transfer these data into a graph. In Section 4 we describe the computational experiments design and evaluate the algorithms on the surfaces of different stations, one artificial one and one based on Paris metro station hall. To conclude, Section 5 discusses limitations of this work and future research directions.

2. Related work

Finding the best route at stations with a complex architectural plan plays an important role since many different public transport lines connect at large stations. It is often difficult to get to the desired destination, even with instructions from the employees present at the stations. Therefore, an application for navigating inside the station can be of great help in finding the shortest path. When we refer to ”path optimality,” we not only mean the minimum distance but also the minimum time spent on the path. For example, this can be important in the case of moving walkways. Furthermore, the right approach can allow for the consideration of different groups of passengers, such as suggesting routes for passengers with limited mobility.

Given a flat environment with two points defined on it — point A (the origin of the path) and point B (the destination of the path), and a set of polygonal objects acting as obstacles. It is required to find the minimum possible route between two given points. However, somewhat surprisingly, most state-of-the-art pedestrian navigation services route around such walkable areas, not through them.

As a rule, pathfinding consists of two main stages: graph generation and a pathfinding algorithm. Roadmap techniques are among the first methods to address path-planning problems. These methods map the free space to a connectivity graph that is later searched to find the shortest path. Obstacles are represented as closed polygons, and a graph is created using the vertices of the obstacles as graph nodes, which are connected by graph edges if the obstacles do not block a straight line between them.

For example, to build a roadmap, cell decomposition methods are used. The two main partitioning methods, in which the graph is built on the lines of the junction of cells, are trapezoidal decomposition and Morse decomposition. To speed up route calculations on large maps, methods based on sampling have been developed, such as Rapidly-Exploring Random Trees (RRT) [2] and probabilistic roadmaps [3]. The decomposition could be conducted either exactly or approximately [4]. The exact method decomposes the free space into simple cells of triangular and/or trapezoidal shapes [4]. Alternatively, the approximate decomposition starts with discretizing the workspace into a known number of cells of a prespecified shape. The division of the cells is continued recursively until each cell is located completely inside or outside the obstacle space, or a termination criterion is achieved. Neither exact nor approximate cell decomposition is efficient in finding the shortest path since the exact algorithm cannot provide the global solution, and the approximate algorithm is not computationally efficient [5].

Likewise, the Voronoi diagram method is widely used to build a roadmap. This method allows one to build the most secure route since the edges of the graph will be located at the greatest distance from obstacles [6], [7]. In addition, the found vertices of the graph can be used in the [8]. Spline-based algorithm, which makes it possible to calculate the smoothest trajectory of movement, an important criterion for, as an example, the movement of people with limited mobility. Since the edges of the Voronoi diagram are created by points equidistant from the pairs of vertices and/or edges of the two closest obstacles, it results in the maximal clearance path, which is not necessarily the shortest. Researchers have utilized Voronoi diagrams in solving the path-planning problem over the past decades [9]. O'Dunlaing et al. [10] are pioneers of using Voronoi diagrams for solving planning problems using the “Retraction method”. Simultaneously, though independently, Brooks [11] introduced the freeway technique, which is a more empirical version of the retraction by the notion of Voronoi diagrams.

Constructing a ”visibility graph” is the original method in computational geometry for addressing the shortest path problem [12]. Nilsson [13] introduced this method to plan a path for a mobile robot. The visibility graph of a set of polygonal objects consists of visible vertices connected by non-intersecting line segments. Any two nodes that can be connected by a line segment without intersecting obstacles in the workspace are considered visible.

This method is widely applied to various planning problems to reduce the routing problem to a graph search of feasible solutions [14, 15]. The Visibility Graph Connection Points (GCP) only connects points of the surrounding polygon that have a node degree higher than two. Similar to "Visibility All," a connection is established only if there is a direct obstacle-free path. The advantage of "Visibility GCP" is that it connects nodes that are actually linked to other parts of the routing network, such as nodes connecting open spaces with roads.

Another potential improvement to the algorithm is reducing the number of inserted visibility graph edges by removing irrelevant edges. Visibility graph edges are considered relevant if they belong to the shortest path between two entrance/exit/utility points of the space, where the space intersects the existing pedestrian routing network. This is the case in [16], where the authors propose utilizing visibility graphs to create plaza graphs. They assume that the street network database provides traversable plazas as a set of plaza polygons, potentially with obstacles causing holes.

Delaunay triangulation [16] is widely used in indoor navigation for scenarios with numerous rooms. For example, Zhu et al. propose a few-step indoor navigation graph construction. Given a floor plan, the indoor space is extracted as a polygon, and its vertices are used to generate the first triangulated irregular network (TIN). The centroids of each triangle in the first TIN are then extracted and used to generate the second TIN. Finally, the nodes and edges of the second TIN that fall entirely within the indoor space are extracted to create the indoor navigation graph.

The Probabilistic Road Map, first presented by Overmars [17], generates a random graph within the free space. The algorithm starts by adding the start and goal nodes to the graph, then introduces random nodes in the free space until a complete path connecting the start and goal is formed through the randomly generated nodes [18, 19]. PRM is an effective method for dynamic or online path planning problems. However, it may struggle to meet optimization criteria due to the probabilistic nature of graph construction.

Hahmann et al. [20] compared roadmap construction algorithms based on four criteria: number of graph edges, graph creation time, route computation time, and route quality. They found that the Visibility Graph delivers the shortest possible routes and is a promising method. However, it creates many redundant edges and requires optimization. Delaunay triangulation generally produces satisfactory route results, but occasionally yields more modest results.

Nevertheless, computing the visibility graph and post-processing the routing can be time-consuming, particularly due to nodes with high degrees, i.e., many connections to other nodes. In contrast, Visibility GCP can be computed relatively quickly. Its routing quality is mostly good, but sometimes important connections are missing, especially on concave open spaces. The authors did not consider a reduced graph of relevant visibility edges. However, as shown by Graser [21], this approach can significantly reduce the number of additional edges in the graph (e.g., reducing the number of edges from 282 to 19 on Josefsplatz).

Considering the advantages and disadvantages of the discussed algorithms, a reduced or modified version of the Visibility Graph can provide an optimal solution.

3. Modelling the indoor navigation

We represent each space at the station as a polygonal domain D, which consists of a simple polygon P containing disjoint simple polygonal holes, also known as obstacles. Alternatively, it can be considered as the "free space" F, which is the polygon Pwithout the interior of the holes. Let the number of vertices be denoted as n, and the number of holes as h. The set of possible entries, exits, and utilities is denoted as U, where $|U| = u$ (see Figure 1a).

The objective of indoor navigation is as follows: given D with two points $s, t \in U$, find the **Euclidean shortest path (SP)**. For instance, SP_{st} from s to t such that SP_{st} lies in F (see Figure 1b).

It is proven in [12] that no internal vertex of a SP can lie in either the free space or the interior of an edge.

We define a weighted undirected graph $VG_D(V, E')$ over D with set U as follows:

- V represents the set of vertices in D, merged with U
- An edge $e(u, v) \in E'$ exists whenever u and v are visible to each other in D, and
- The weight of each edge $e(u, v)$, e, is the Euclidean distance along the line segment uv in D.

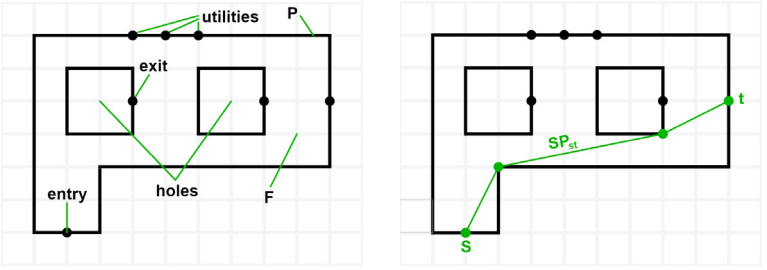

Fig. 1. (a) Polygonal domain; (b) Shortest path in polygonal domain

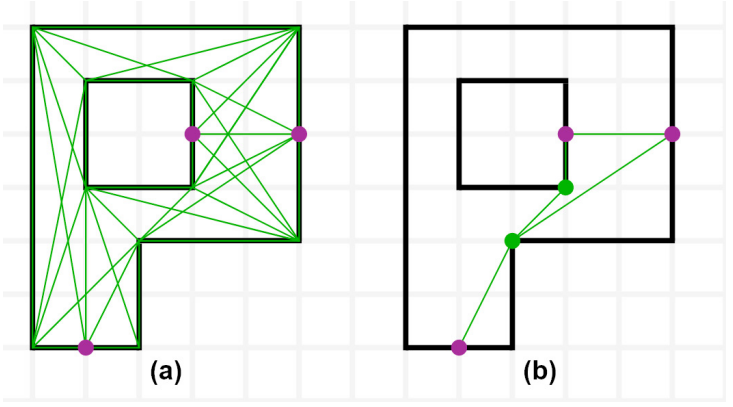

Fig. 2. (a) Small polygon Q with a hole and all visibility edges ; (b)Visibility edges that are also on shortest paths (green vertices are the inner parts of paths, the purple vertices are entry points).

The time complexity is determined by VG_D computation and $|E'|$.

To compute the *entry and exit nodes*, given P and the adjacent station structure, we identify the common boundary parts between different polygons within the same station. We add each entry/exit node as a vertex to the corresponding polygon. Next, for each polygon P, we then compute the *visibility graph* $VG(P)$. If P has no holes, the computation requires quadratic time [22, 23]. However, if P contains holes, it requires cubic time. Although this calculation is performed as a precalculation step, we do not implement a more efficient algorithm, such as the one described in [24].

As the number of visibility edges $E_{vis}(P)$ of a space polygon P can be quite high (see Figure 2a), routing through open spaces can become costly. To address this, we mark a subset $E^{sp}_{vis} \subset E_{vis}$ of visibility edges that are part of the shortest paths between any pair of utility nodes. This allows the routing algorithm to ignore unmarked edges (see Figure 2b). To determine E^{sp}_{vis}, we run Dijkstra's algorithm between each pair of utility nodes. It's important to note that E^{sp}_{vis} is sufficient for routing across the station and reaching any utility. Additionally, calculating E^{sp}_{vis} requires knowledge of the routing cost function (all other preprocessing does not), but since the necessary shortest path queries are limited to each station space and the number of utility nodes is typically small, this step is not significantly time-consuming compared to the overall preprocessing effort.

However, the proposed model does not account for an important feature of station indoor navigation: the possible directions of the corridors, even when disregarding the standard instructions to keep to the right.

To address this, we can divide the station corridors into separate rooms, some of which may have one direction. This enables us to modify the visibility graph. Similar to the method described above, we search for the shortest paths between different origin-destination pairs. However, some points can only serve as either an origin or a destination, depending on whether they lead to a one-way corridor.

4. Computational experiments

4.1. Case study

For the first case study, we examine an artificial room without any structural holes, as shown in Figure 3 a.

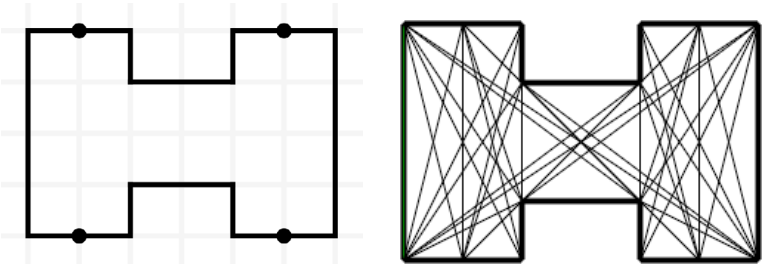

Fig. 3. (a) Case study 1: artificial room; (b) Full visibility graph

By applying the visibility graph algorithm, we obtain the structure depicted in Figure 3 b which consists of 54 edges and 16 vertices.

Simplifying this graph into a non-directed network yields the graph shown in Figure 4 a containing 8 vertices and 10 edges. Considering the directions of entries and exits, the resulting graph is depicted in Figure 4 b with 8 vertices and 8 edges.

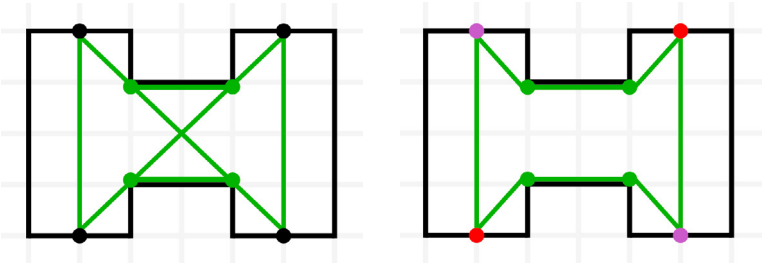

Fig. 4. (a) Shortest path visibility edges; (b) Semi directed shortest path visibility edges **(the red vertices are enries, the pink are exits)**.

Moving on to the second case study, we analyze the hall of a metro station in Paris (refer to Figure 5). To introduce more complexity to the pathfinding process, we have added walls to the actual station geometry. The entries, exits, and utility points have been approximately positioned.

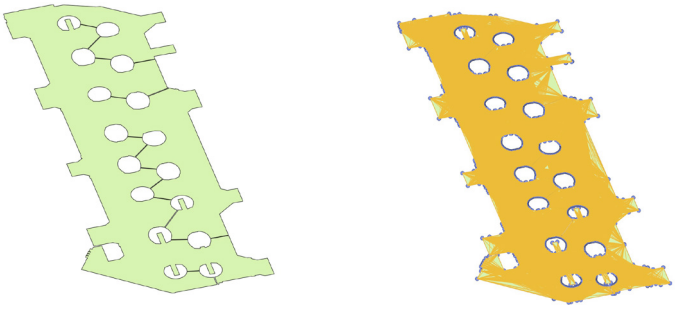

Fig. 5. (a) Hall of a metro station in Paris with modifications; (b) All visibility edges of the proposed station

We will compare the resulting graphs obtained through different methods.

4.2. Numerical results

We have 1118 vertices in the selected hall. Therefore, we can consider that the utilities are located at the vertices of the polygon. The complete visibility graph remains unchanged with respect to the utilities' locations and contains $|E_{vis}| = 26688$ edges.

To evaluate the proposed algorithms based on the station geometry, we vary the number of utilities. We can recall that $|U|$ represents the total number of utilities, N_0 is the number of utilities that can only serve as origins, N_d is the number of utilities that can only serve as destinations, and N_{od} is the number of utilities that can function as both origins and destinations.

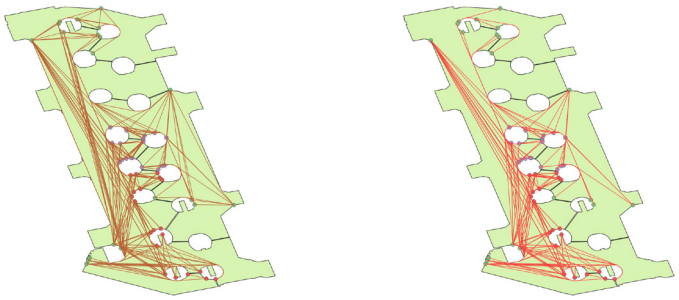

Fig. 6. (a) The roadmap without taking into account the directions of the utilities for 60 utilities; (b) The roadmap with taking into account the directions of the utilities for 20 origins, 20 destinations and 20 bidirected poins.

Figure 6 illustrates the difference in graphs obtained for non-directed and directed utilities. We can observe that considering the directed flows of passengers, certain edges will never be traversed. As a result, the resulting graph becomes smaller, and calculating the shortest path using the proposed approach can be faster.

In this context, we are constructing different roadmaps while incorporating randomly generated utilities. $|E_{vis}^{sp}|$ represents the average number of edges in the visibility graph that are included in at least one non-directed shortest path between utilities. On the other hand, $|E_{vis}^{spdir}|$ represents this value in the case of directed utilities. Since we consider random utility points, we can compute the average value over 50 iterations.

Table 1. The statistics for different amounts of utilities.

$\|U\|$	N_0	N_d	N_{od}	$\|E_{vis}^{sp}\|$	$\|E_{vis}^{spdir}\|$	% of decrease
6	2	2	2	99.92	95.65	4.27
15	5	5	5	226.93	206.66	10.25
30	10	10	10	383.03	313.89	18.05
60	20	20	20	643.7	484.23	25.08
75	25	25	25	752.66	550.27	26.89
90	30	30	30	887.37	635	28.44
70	25	25	20	720.5	516.93	28.25
65	25	25	15	680.1	486.87	28.41
60	25	25	10	632.73	446.77	29.39
50	25	25	0	557.63	369.67	33.71
60	30	30	0	649.67	413.1	36.41

At Table 1, it is evident that as the number of utilities increases, the difference between the amount of shortest path visibility edges and directed shortest path visibility edges also increases. If the categories of utilities are distributed non-uniformly, this difference becomes even more significant.

Figure 7 illustrates the number of edges in shortest path graphs with and without edge orientation for various case studies. The different blue points represent the same total number of utilities but distributed differently. Consequently, the increase in the number of edges in the oriented utility case can vary significantly. It is observed that the graph's edge count grows at a slower rate in the case of oriented utilities.

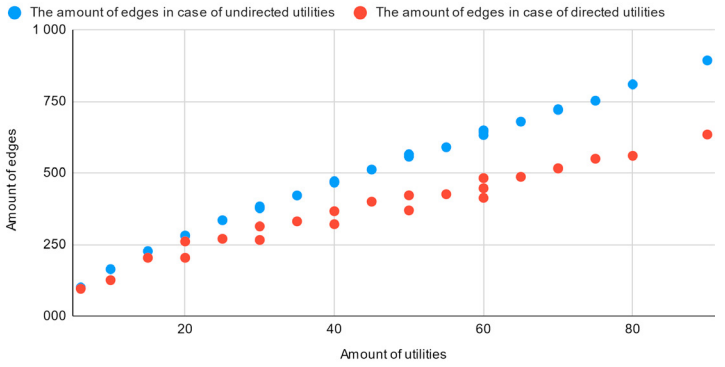

Fig. 7. The dependency of the amount of edges from the amount of utilities for different distribution of utility types

5. Conclusion

In this article, we propose a modification to the visibility graph algorithm for indoor navigation. The proposed modification takes into account the possibility of one-directional passenger flows in specific corridors and treats the junction points between different rooms of the public transport station as either bidirectional or only allowing for one-way travel.

The resulting visibility graph for a given room is then simplified by retaining only the edges that appear in at least one of the shortest paths between origin and destination points. This simplified graph can be easily utilized in any shortest path algorithm, such as Dijkstra or Contraction Hierarchies. However, it is important to note that using the visibility graph with Contraction Hierarchies has a drawback in that it creates numerous nodes with many connections, which can slow down the precalculation process. To address this, our proposed algorithm imposes limits on the number of edges and the degree of each node.

Moreover, the proposed algorithm enables more precise path finding by incorporating ”keep to the right” instructions. This is achieved by dividing each entry node into two nodes, one designated as the origin and the other as the destination of the surface path. This approach facilitates showing travelers a more accurate trajectory and predicting walking time more efficiently.

In future work, we will explore the application of the proposed algorithm to multi-room public transport stations and develop methods for handling the origins and destinations within such complex structures numerically.

References

[1] S. Breux, J. Diaz, La ville intelligente: origine, définitions, forces et limites d’une expression polysémique.
[2] S. M. LaValle, et al., Rapidly-exploring random trees: A new tool for path planning.
[3] L. E. Kavraki, P. Svestka, J.-C. Latombe, M. H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE transactions on Robotics and Automation 12 (4) (1996) 566–580.
[4] J. Latombe, ìrobot motion planning, î kluwer, Boston, MA.
[5] P. Bhattacharya, M. L. Gavrilova, Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path, IEEE Robotics & Automation Magazine 15 (2) (2008) 58–66.
[6] H. Choset, Sensor based motion planning: The hierarchical generalized Voronoi graph, California Institute of Technology, 1996.
[7] B. Lau, C. Sprunk, W. Burgard, Improved updating of euclidean distance maps and voronoi diagrams, in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2010, pp. 281–286.
[8] E. Magid, D. Keren, E. Rivlin, I. Yavneh, Spline-based robot navigation, in: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2006, pp. 2296–2301.
[9] S. Garrido, L. Moreno, M. Abderrahim, F. Martin, Path planning for mobile robot navigation using voronoi diagram and fast marching, in: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2006, pp. 2376–2381.
[10] C. Ó’Dúnlaing, C. K. Yap, A “retraction” method for planning the motion of a disc, Journal of Algorithms 6 (1) (1985) 104–111.
[11] R. A. Brooks, Solving the find-path problem by good representation of free space, IEEE Transactions on Systems, Man, and Cybernetics (2) (1983) 190–197.
[12] J. o’Rourke, Computational geometry in C, Cambridge university press, 1998.
[13] N. J. Nilsson, A mobile automaton: An application of artificial intelligence techniques, Tech. rep., Sri International Menlo Park Ca Artificial Intelligence Center (1969).

[14] G. E. Wangdahl, S. M. Pollock, J. B. Woodward, Minimum-trajectory pipe routing, Journal of Ship Research 18 (01) (1974) 46–49.
[15] T. Lozano-Pérez, M. A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, Communications of the ACM 22 (10) (1979) 560–570.
[16] M. Xu, S. Wei, S. Zlatanova, An indoor navigation approach considering obstacles and space subdivision of 2d plan, Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci 41 (2016) 339.
[17] M. H. Overmars, et al., A random approach to motion planning.
[18] R. Geraerts, M. H. Overmars, A comparative study of probabilistic roadmap planners, Algorithmic foundations of robotics V (2004) 43–57.
[19] A. H. Qureshi, Y. Ayaz, Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments, Robotics and Autonomous Systems 68 (2015) 1–11.
[20] S. Hahmann, J. Miksch, B. Resch, J. Lauer, A. Zipf, Routing through open spaces–a performance comparison of algorithms, Geo-spatial Information Science 21 (3) (2018) 247–256.
[21] A. Graser, Integrating open spaces into openstreetmap routing graphs for realistic crossing behaviour in pedestrian navigation, GI_Forum 4 (1) (2016) 217–30.
[22] M. H. Overmars, E. Welzl, New methods for computing visibility graphs, in: Proceedings of the fourth annual symposium on Computational geometry, 1988, pp. 164–171.
[23] N. Goren, E. Fogel, D. Halperin, Cgal made more accessible, arXiv preprint arXiv:2202.13889.
[24] H. Alt, E. Welzl, Visibility graphs and obstacle-avoiding shortest paths, Zeitschrift für Operations-Research 32 (1988) 145–164.