Trends in Computational and Applied Mathematics, **27** (2026), e01845
Sociedade Brasileira de Matemática Aplicada e Computacional
Online version ISSN 2676-0029 www.scielo.br/tcam
ORIGINAL ARTICLE
doi: 10.5540/tcam.2026.027.e01845

Distance Map Associated with Tangent Search as a Pathfinding Strategy: achieving high-quality and safe route maps for crowed movement in built environments

Henrique Costa Braga[1*] and Gray Farias Moita[2]

Received on December 12, 2024 / Accepted on March 16, 2026

ABSTRACT. Several human movement simulators employ pathfinding algorithms to determine the best route to an emergency exit or other targets within built environments. However, the literature often lacks detailed descriptions of some of these algorithms, particularly those of high quality. This paper introduces a new and enhanced pathfinding algorithm, VSP (Visibility Search Pathfinding), designed for movement simulations in built environments. It is based on the creation of distance maps and route maps generated by tangent search. The algorithm's logic is thoroughly detailed, with examples of its application. Key advantages of the VSP algorithm include simplicity of logic, ease of computational implementation, applicability to environments of any design or geometry, low processing time (in non-dynamic scenarios), and full automation, requiring no specialist input. The VSP algorithm consistently identifies routes that closely approximate the optimal paths, independent of the number of people in the environment. Furthermore, the VSP algorithm allows the creation of a region around obstacles that prevents collisions. Thus, the VSP becomes a valuable tool for high-quality human movement simulations in evacuation contexts or similar scenarios.

Keywords: shortest path, pathfinding algorithm, Von Neumann neighborhood, Moore neighborhood, emergency exit, Dijkstra algorithm.

1 INTRODUCTION

Several human movement simulation models employ algorithms to identify the best route between an agent's initial position in an environment and a specific target, such as the nearest emergency exit [18, 25, 26, 34]. These models can be integrated into computational tools, serving as valuable support for architects and engineers during the design process. Such studies are crucial

*Corresponding author: Henrique Costa Braga – E-mail: henriquebraga@cefetmg.br
[1]Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG), Department of Transportation Engineering, MG, Brazil – E-mail: henriquebraga@cefetmg.br https://orcid.org/0000-0001-9504-6156
[2]Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG), Department of Civil Engineering and Post-Graduate Program in Mathematical and Computational Modeling, MG, Brazil – E-mail: gray@cefetmg.br https://orcid.org/0000-0002-6510-1019

for understanding the impact of architectural designs on building safety [29, 51], and reinforce the role of information technology as a fully established field within architectural design [24].

The optimal paths are those routes that minimize (or maximize) some metric between the specified points. The geometric distance is often used mainly as a metric [2, 11]. Although other factors, such as population density [39,42,56], pedestrian distribution [55], personal spaces [28], flow direction [54], information dissemination [31], and specific contexts [5, 33, 48], among others, are also relevant, sometimes even preponderating, understanding the shortest path remains a fundamental aspect.

The shortest path can be found using various methods, including applying search and exploration methods known as pathfinding algorithms. These algorithms identify the best route between two specific points in an environment (or a graph). Several literature reviews on this topic have been conducted [27, 35, 37, 47, 49]. Moreover, modern techniques such as genetic algorithms, neural networks, and fuzzy logic have also been applied [1, 12, 23, 43].

Among the many available algorithms, two stand out for their widespread use: Dijkstra's algorithm [15] and A-Star [16, 19]. Currently, both the Dijkstra and A-Star algorithms have numerous variations and applications [30, 32, 50].

The Dijkstra algorithm searches for the shortest path by assigning a cumulative cost to each node, starting from the initial node. It systematically explores all nodes, ensuring the lowest cost between the source and destination points, and guarantees the shortest path to all reachable nodes in graphs. However, its main limitations include the inability to handle negative edge weights and potential inefficiency in large or dense graphs due to its exhaustive exploration.

As highlighted in the algorithm engineering literature, more advanced methods achieve enormous performance gains: Chimani and Klein [14] note that state-of-the-art point-to-point algorithms can be tens of thousands of times faster than standard Dijkstra; Goldberg and Harrelson [17] motivate the development of more efficient point-to-point search methods than Dijkstra; and Bast et al. [4] demonstrate that modern algorithms may outperform Dijkstra by several orders of magnitude, even up to millions of times faster in transportation networks. These findings reinforce why most of the relevant works in the literature, as well as commercial evacuation simulation software [34, 35] do not adopt Dijkstra's algorithm for crowded scenarios.

On the other hand, the A-Star algorithm is an extension of Dijkstra, incorporating a heuristic function to prioritize nodes closer to the target. This heuristic makes A-Star potentially more efficient in terms of computational cost, as it reduces the number of nodes explored. However, the accuracy of the indicated path depends on the quality of the heuristic used, which may lead to deviations from the optimal path in some cases.

Moreover, in modeling a generic evacuation in a built environment, it is expected to have not just one or a few persons but a large number, potentially hundreds or even thousands, distributed throughout the entire space. Using algorithms individually for each person may not be practical in

a simulation, making it highly valuable to apply a tool capable of finding the best path across the entire environment in a single operation. This can be achieved through a distance map [11, 46].

The distance map provides, for the entire environment – that is, for all points accessible for movement – their respective distances to the nearest exit or target. Thus, a person located at any point in the environment can find the nearest exit by simply moving to the neighboring point with the shortest distance to the exit, systematically continuing this process until reaching the exit.

Once this map is generated, no further analysis is needed for individuals to reach the nearest exit. Additionally, this method maintains the same computational cost regardless of the number of people in the simulation or their locations within the environment. Once the map is obtained, the simulation becomes much more efficient, as the map can be generated prior to the simulation (non-dynamic utilization), thus not affecting the computational cost of the simulation's execution.

Microscopic approaches to modeling pedestrian dynamics, such as those based on Cellular Automata and Floor Fields [13, 36, 54] rely on such algorithms. Consequently, errors and inaccuracies in these pathfinding algorithms can significantly affect the quality of simulation results.

Amongst the algorithms used to create the distance map, two stand out due to their search approach: those based on the Von Neumann neighborhood (VNN) and those based on the Moore neighborhood (MN). These algorithms are well-known and thoroughly detailed in the literature [7].

However, when analyzing the preferred direction profiles generated by some of these algorithms (VNN or MN), considerable deviations from the optimal route are often observed [3, 36, 38, 40]. In some cases, additional algorithms are required to minimize these errors [12].

Furthermore, while certain algorithms may seem to avoid these issues upon superficial analysis, providing highly realistic results [34, 46], the highest-quality algorithms are often not well-documented in the literature. This lack of clear and comprehensive documentation can hinder their broader adoption, often resulting in the use of lower-performance algorithms to generate the distance map instead of superior alternatives.

Thus, the main objective and contribution of this work is to provide a detailed description of the logic and some application examples of a pathfinding algorithm called Visibility Search Pathfinding (VSP), designed for non-dynamic applications. The VSP algorithm is based on creating distance maps and route maps through tangent search. This algorithm combines the advantages of direct search algorithms, such as VNN and MN, while eliminating the errors in distance calculations or route profiles associated with these methods. Furthermore, the VSP algorithm allows the creation of regions around obstacles to prevent collisions.

This feature makes the VSP algorithm suitable for high-performance software to simulate environment evacuations.

2 FUNDAMENTAL BACKGROUND

Pathfinding algorithms for non-dynamic applications often utilize some of the simplest methods, such as breadth-first search, over a discretized environment. These searches typically start from the targets (commonly the available exits) and then expand to cover the entire environment. As mentioned earlier, the search typically employs either the Von Neumann neighborhood (VNN), which considers only the four nearest neighbors of each cell, or the Moore neighborhood (MN), which also includes the diagonal neighbors [52, 53]. Figure 1 illustrates the first neighborhoods for both VNN and MN.

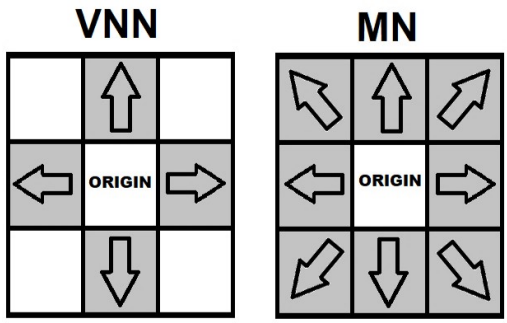
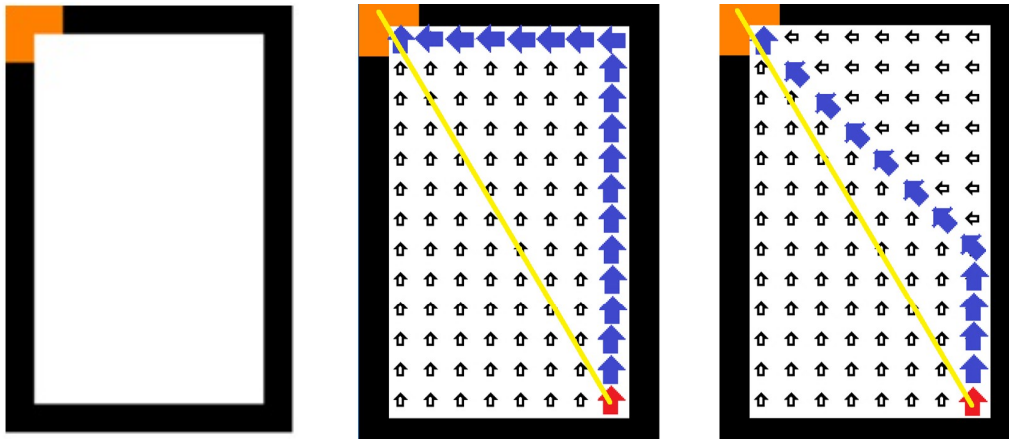


Figure 1: Representation of the first-order neighboring cells based on the Von Neumann (VNN) and Moore (MN) neighborhood models.

These algorithms offer several advantages: simplicity in their internal logic, general applicability (they can be used in environments with any geometry), straightforward computational implementation, low processing time (for non-dynamic applications), comprehensive search of the environment (ensuring no unexplored areas), and full automation (they do not require specialists or human intervention). However, the generated routes can significantly deviate from the true optimal path. These pathfinding algorithms, along with additional aspects, are discussed in the literature [7]. To illustrate, Figure 2 presents a top-view representation of a simple didactic environment (not to scale), consisting of a single rectangular floor with no internal obstructions and one exit located in a corner.

The walls are represented in black, the exit in orange, and the obstacle-free interior in white. Additionally, Figure 2b and Figure 2c display the corresponding route maps that could be generated for the environment depicted in Figure 2a. Different routes with the same total 'best lower' distance can be identified, depending on whether the VNN or MN algorithms are used. The movement directions indicated by the route maps (blue arrows) between an initial point (in red) and the exit (in orange) are highlighted, along with the obvious best theoretical (straight) path (in yellow).



(a) (b) (c)

Figure 2: Top-view representation of a simple environment (a), with examples of the respective route maps showing general routes (black arrows) and movement directions (blue arrows) generated by the VNN (b) and MN (c) algorithms, along with the corresponding ideal straight-line path (in yellow). Adapted from [10].

The difference between the ideal route (in yellow) and those obtained by the algorithms (in blue) depends on the environment geometry, the relative position of the person to the target, and the criteria adopted to generate the route map from the distance map. Unfortunately, errors of approximately 40% in distance calculations can occur when using the VNN algorithm, and up to 8% when using the MN algorithm [7]. Although the results from MN are significantly better, considerable errors persist.

3 METHODOLOGY

The VSP algorithm, like the VNN and MN algorithms, is also based on a breadth-first search from the targets (e.g., the emergency exits) toward the interior of the environment. The search uses the first VNN (due to the logic of the VSP algorithm, the issues caused by VNN will not affect the results, as VNN is used solely to guide the search). However, a route map (or reference map) is also generated simultaneously, in addition to the distance map. The reference map shows minimal sudden changes in the indicated direction toward the exit.

The reference map is constructed based on the principle of visibility, or searches by tangent, or even Euclidean shortest path [22, 44]. Each cell points to another cell closest to the exit, where visibility is possible. Visibility is possible when there is no obstruction along the straight line between two cells.

Instead, the VSP algorithm explores the entire environment without leaving gaps and determines routes based on the shortest Euclidean distance along tangents. It is important to emphasize, however, that the VSP algorithm cannot be considered ”optimal” in the strict mathematical sense. The

discretization of a real built environment into a lattice inevitably introduces geometric approximations that cannot be fully eliminated. Thus, even if an algorithm guarantees the best solution within the discrete graph, the result may still deviate from the true continuous geometry. From an engineering perspective, however, the VSP can be regarded as optimal, since its results consistently remain closer to analytical expectations and superior to alternative algorithms when evaluated in terms of accuracy and efficiency in lattice-based simulations.

From a geometric perspective, visibility-based shortest-path maps in continuous polygonal environments admit optimal solutions, as shown by [22]. Our grid-based VSP can be interpreted as a discrete approximation of this continuous visibility framework. Although a formal discretization-error bound (expressed as a function of the grid resolution r and obstacle geometry) lies beyond the scope of the present work, our empirical results show that, with $r = 0.05\,\mathrm{m}$, the deviation from the analytical solution remained below 1% even in benchmark layouts with multiple angular deflections.

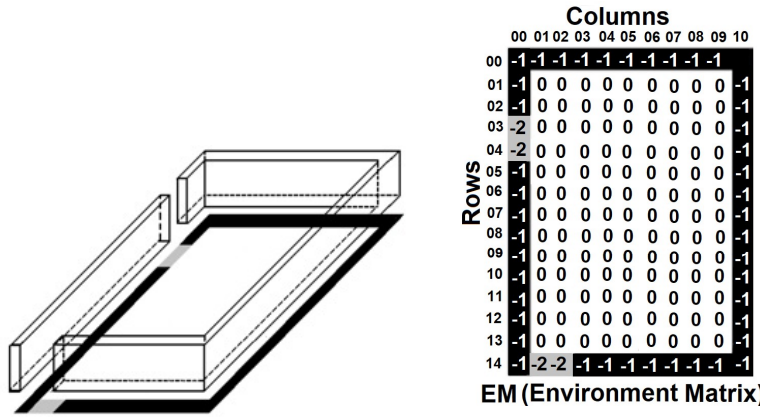
Moreover, uncertainties inherent to human behavior must also be considered. Pedestrian evacuation dynamics are strongly influenced by perception, cognition, decision-making, and social interactions, all of which introduce deviations that cannot be fully captured by purely geometric or physical models [9, 21, 41]. These behavioral factors often dominate the overall uncertainty in evacuation predictions, making small geometric deviations caused by discretization or algorithmic choices secondary in comparison. For these reasons, when assessed in the broader engineering context, the VSP provides results that are sufficiently robust and ”optimal” for practical applications.

The first essential step is to discretize the environment in the form of a two-dimensional rectangular matrix, named the Environment Matrix (EM). Each element of this matrix represents a portion of the environment, with its resolution (r) proportional to the overall dimensions of the environment. All elements of the environment (walls, passages, internal obstructions, and exits) must be an exact multiple of the resolution unit (or rounded to that). As an example, consider the environment depicted in Figure 3a and its corresponding EM in Figure 3b.

Initially, all positions within the environment that are not occupied by walls or exits will have the value ’0’ in the corresponding cell in the EM; the walls will have the value ’-1’; and the exits (or targets) will have the value ’-2’ (these values ’0’, ’-1’, and ’-2’ are used just as references in this example).

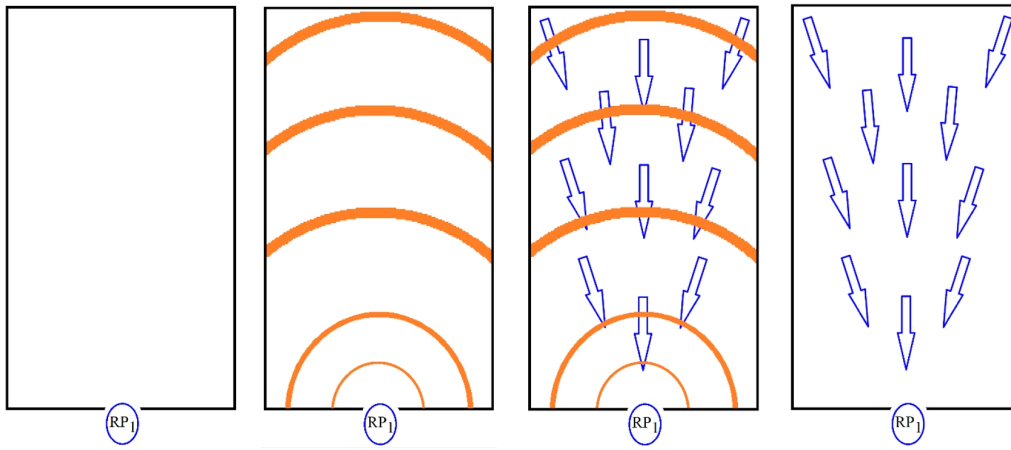
Figure 4 briefly illustrate the steps of the VSP algorithm to better explain the underlying logical principles.

In Figure 4a, a top-view of a hypothetical environment is presented. It consists of a single rectangular room with no internal obstructions, black walls, and a single passage. This passage serves as both the entrance and the exit of the environment and acts as a reference point, designated as RP_1, located at the center of one of the shorter walls.



(a) (b)

Figure 3: Example of an illustrative environment shown in perspective view (not to scale) along with its top-view projection (a) and the corresponding Environment Matrix (b). Reproduced from [7].



(a) (b) (c) (d)

Figure 4: Top-view images of a hypothetical environment consisting of an unobstructed room, illustrating the steps to derive the route map from any point to the destination (RP_1) using the distance map.

In the VSP algorithm, the environment is discretized into its respective EM. An algorithm is then applied to this matrix, starting from the final objective (the initial reference point, in the example shown, RP_1), scanning the entire environment. This algorithm calculates the distance-to-exit values (D_e), which, in this case, is the straight unobstructed Euclidean distance from each cell center in the EM to the reference point considered (RP_1). In this specific scenario, the unobstructed Euclidean distance is equivalent to the direct Euclidean distance since there are no internal obstructions, and the environment is geometrically convex (in this case, a rectangle).

These D_e values are stored in another matrix, its respective distance map, which has the same dimensions as the EM. The orange lines shown in Figure 4b illustrate several curves where all the elements of the environment along each orange line share the same D_e from RP_1.

Using the distance map, the algorithm generates another matrix called the route map, which also has the same dimensions as the EM. Each cell in the route map contains a vector pointing toward the reference point adopted during its generation, in this case, RP_1. Some of these vectors are represented as blue arrows in Figure 4c, which also displays several curves indicating equal D_e distances from RP_1.

Finally, Figure 4d illustrates the environment with only the arrows pointing toward the reference point (RP_1). Thus, starting with the environment constructed as a matrix (EM), the VSP algorithm generates the distance map, and later, using the distance map, it generates the route map. By simply following the route map, any individual in the built environment can efficiently and directly find the exit (RP_1), avoiding the errors of simpler algorithms such as VNN or MN.

Indeed, the environment in Figure 4 was designed to be very simple for didactic purposes, so the paths could have been found even without any advanced algorithm, as the solution is trivial. From any location in the environment, one could simply point toward the exit, RP_1, and proceed directly. However, real-built environments can become much more complex, making it far from straightforward to identify the best paths. By applying the VSP algorithm, regardless of the environment's complexity, geometry, or number of floors, the routes to the exit will be determined automatically, without human intervention.

To illustrate the VSP algorithm's logic in slightly more complex environments, Figure 5 introduces complementary reference points.

In Figure 5a, an environment similar to that in Figure 4a is presented, but now containing an internal obstruction (in black). When applying the VSP algorithm to the respective new EM, it calculates the distance of each cell from the reference point RP_1, generating the respective new distance map.

In the region between the obstruction and the exit, the distance values obtained from the distance maps in Figure 4b and Figure 5b are identical. Similarly, across much of the area located between the obstacle and the larger walls, there will be no difference between the distance maps.

However, in the region behind the obstruction, the differences between the distance maps from Figure 4b and Figure 5b emerge. This occurs due to the calculation of D_e. In this region behind

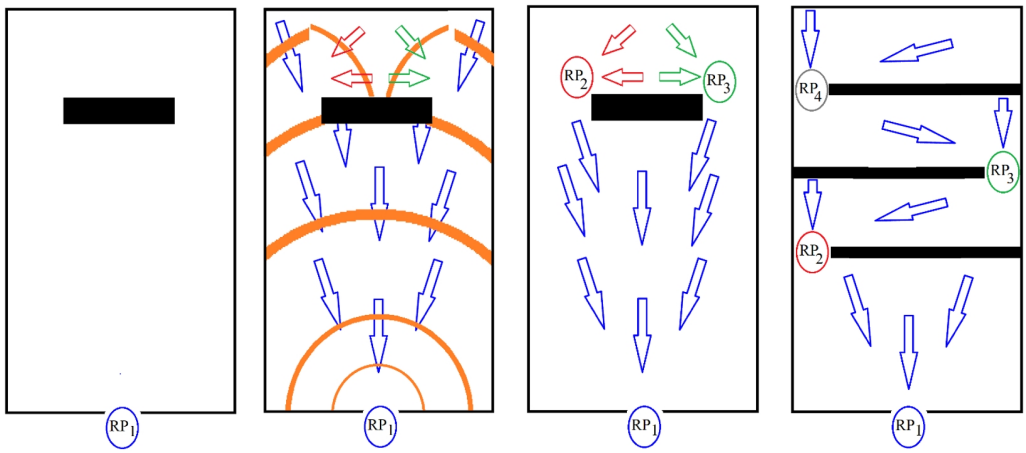


Figure 5: Top-view images of two hypothetical environments. In (a), (b), and (c) an environment with one internal obstruction and two complementary reference points and in (d) an environment with three internal obstructions and three complementary reference points.

the obstruction, there is no straight line that directly connects the reference point RP_1 without intersecting the obstruction. When applying the VSP algorithm to the environment to generate the distance map, the algorithm first checks if a point in the environment has an unobstructed straight line to the reference point (initially RP_1) before calculating the Euclidean distance. If the line is not blocked, the Euclidean distance is calculated directly, and the algorithm proceeds to the next point.

However, if there is an obstruction — meaning there is no longer an unobstructed straight line between the considered point and the adopted reference point — the nearest previous neighbor to this point becomes the new reference point, such as RP_2 (or RP_3), depending on the location of the considered point behind the obstruction. As the calculations are performed from the active reference point, from this point onward, all points in the environment to be checked will consider this new reference, RP_2 (or RP_3).

When such a reference change occurs, the unobstructed Euclidean distance for these specific points is calculated as "1" (the distance between the point and the new neighboring reference), plus the distance from the new reference point to the original one. Thus, the general equation for the distance to the exits can be obtained (Eq. 3.1).

$$D_e = D_{pr} + D_l \tag{3.1}$$

Where: D_e is the distance of any cell to the nearest final exit (this is the value placed in the distance map); D_{pr} is the distance from the specific reference point considered to the nearest final exit; and D_l is the Euclidean distance from each cell to the specific reference point for that cell.

At this point, all these distance values are in number of cells (with the central point of each cell as a reference). To convert these values to a distance unit in the SI - International System, simply multiply the distance values in number of cells by the adopted resolution r at the end of the simulation.

In the example of Figure 4, D_{pr} is zero (no intermediary reference points), so in this case, $D_e = D_l$.

Figure 5b illustrates some of these lines representing equal unobstructed Euclidean distances to the exit (RP_1) and the corresponding routes found. It can be observed that near the obstruction, both the distance profile (orange lines) and the orientation of the arrows (routing paths) change direction.

Removing the distance lines, Figure 5c illustrates some routes indicated by arrows and the two complementary reference points automatically generated by the VSP algorithm, RP_2 and RP_3. The blue arrows point directly to RP_1, the red arrows to RP_2, and the green arrows to RP_3.

Finally, Figure 5d depicts an environment with multiple internal obstructions. Applying the VSP algorithm to this environment, the distance map is obtained, reflecting the unobstructed Euclidean distances for all points within the environment. To achieve this, three additional complementary reference points (RP_2, RP_3, and RP_4) are generated to cover the entire space and create the desired route map.

The algorithm logic described above was implemented computationally using the Python programming language. Complementary resources, including demonstration videos and the pseudocode, are provided in the sections **Supplementary Resources** and **Appendix**, respectively. It is worth noting, however, that there are several ways to develop this algorithm [8].

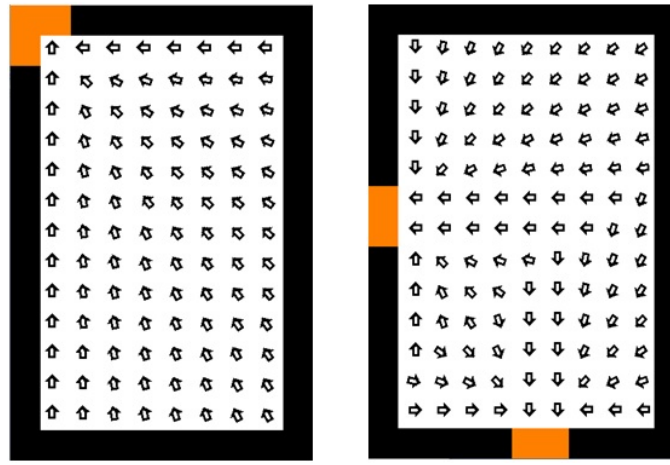
The VSP method represents a geometric refinement of traditional grid-based approaches, improving the accuracy of distance estimation in discrete environments. Having presented and illustrated the main concepts and logic of the VSP algorithm, its practical application is now demonstrated.

4 RESULTS AND DISCUSSIONS

4.1 The VSP algorithm

To illustrate some of the characteristics of the VSP algorithm and its differences from the classic VNN and MN, several experiments were devised. Initially, the didactic environments shown in Figure 2 (one exit) and Figure 3 (two exits) are now presented in Figures 6a and b, respectively, but with indications of the best routes obtained by the VSP algorithm (route map). As expected, Figure 6 typically exhibits a smoother discrepancy in the orientations between adjacent cells that are directed towards the same exit (e.g., observe the differences between Figure 2 and Figure 6a).

Now, square-shaped environments without internal obstacles, with walls of one cell thickness and an exit of two cells width positioned at the center of one side, are simulated. These environments have widths ranging from 10 to 1,000 cells. Considering a resolution of $r = 5$ cm, the value of



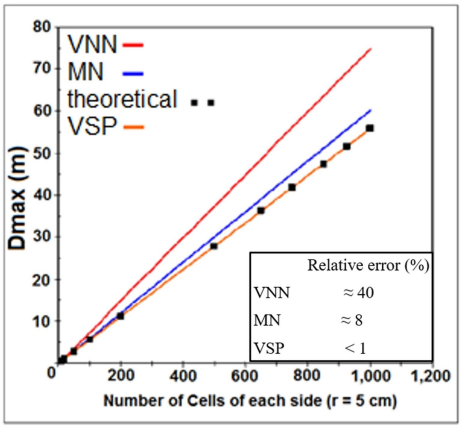
(a) (b)

Figure 6: Route maps with arrows indicating the path to the exit (in orange) as obtained by the VSP algorithm. In (a), the map corresponds to the environment shown in Figure 2 (one exit), while in (b), it corresponds to the environment shown in Figure 3 (two exits).

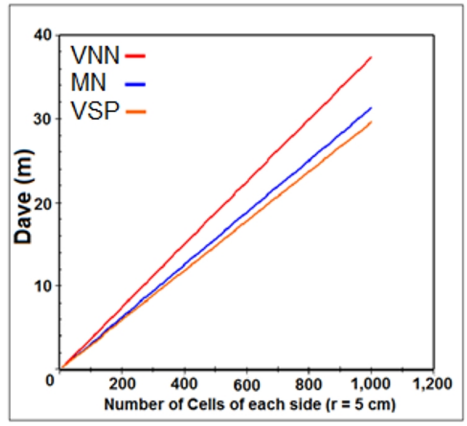
D_{max} (maximum distance indicated in the environment to the exit) and the value of D_{ave} (the average distance in the environment, considering the values of all free cells in the distance map) for each square environment were obtained from the distance map.

Figure 7 shows the results obtained using the VSP algorithm, as well as the VNN and MN algorithms in the same environments for comparison. Figure 7a clearly shows that there are no noticeable differences between the theoretical (exact) distance and that obtained by the VSP algorithm. Additionally, according to Figure 7b, the values for the distances (D_{max} and D_{ave}) obtained using the VSP in square environments with no internal obstructions are 8% lower than those obtained by MN and about 34% lower than those obtained by VNN. Finally, Figure 7c compares the computational times of the three algorithms, showing that VSP is approximately one order of magnitude slower than MN, which in turn is slower than VNN. All reported processing times correspond to wall-clock time, measured under identical hardware conditions. However, since the processing is performed prior to the evacuation simulation (non-dynamic application), this additional cost is not operationally significant and is fully justified by the substantial improvement in both the accuracy and quality of the results. It should be noted that absolute processing times naturally depend on implementation details and hardware conditions; thus, the results are presented to highlight relative trends rather than exact values.

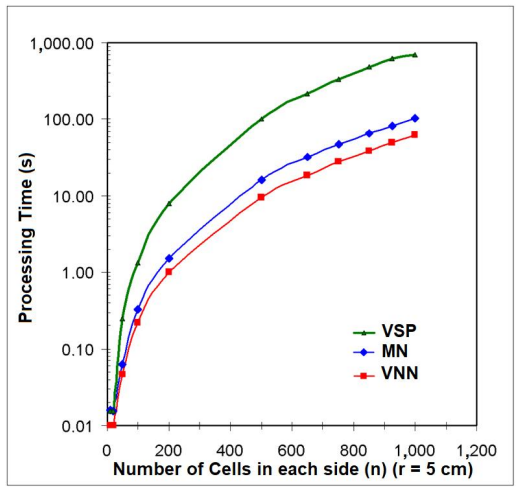
It is important to note that the computational cost shown in Figure 7c refers exclusively to the precomputation stage. During the evacuation simulation itself, the cost becomes practically negligible: routes are retrieved through direct table lookups, with no search, no gradient propagation



(a)



(b)



(c)

Figure 7: (a) Values of D_{max}, (b) values of D_{ave}, and (c) computational processing times obtained using the VSP, VNN, and MN algorithms for square environments with varying numbers of cells per side ($r = 5$ cm) and a single exit located at the center of one side. Data for VNN and MN are based on [7], while VSP results are presented here and further developed in this work (see [8]).

and no additional geometric processing. Hence, once the precomputation has been performed, the pathfinding cost within the simulation is effectively zero in practical terms.

Figure 8 shows the distance map profiles for these square environments, with a 50 m side (2,500 m² of area), for all three algorithms (MN, VNN, and VSP). Each different color range indicates a distance of 1 m from the exit. Figure 9 shows the stylized representation of some distance line

profiles obtained from Figure 8 for all three algorithms, providing a better visualization of the impact resulting from the differences.

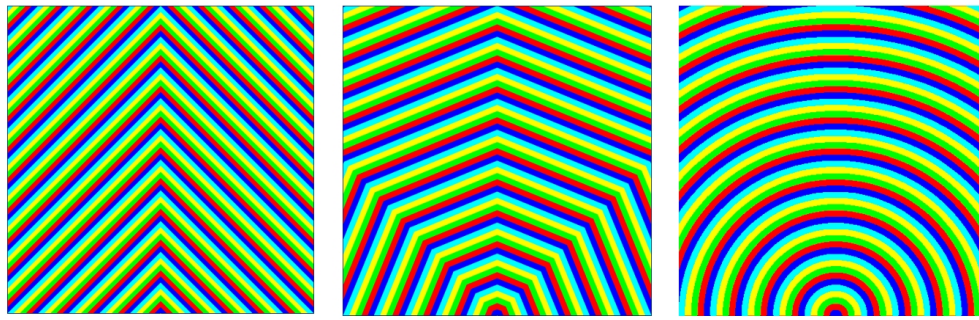


Figure 8: Distance map profiles for square environments with sides of 50 m, according to the VNN, MN, and VSP algorithms (exit located in the middle of the bottom side; each color range represents a distance of 1 m from the exit). VSP based on [8].

In Figure 9, the red semi-lines were obtained from VNN, the blue semi-lines from MN, and the faded background is the distance map obtained by the VSP, where each color range indicates a distance of 5 m from the exit. Still, in Figure 9, the thumbnail on the right shows a reproduction of the main image with the identification of regions where the discrepancy between the distance profiles obtained by the three algorithms is smaller.

Figure 10 shows the distance map profiles for a more sophisticated hypothetical environment with two independent exits and several internal rooms for all three algorithms (VNN, MN, and VSP). Each different color range indicates a distance of 1 m from the nearest exit. For comparison, the values of D_{max} and D_{ave} were obtained for all situations.

It is worth noting that the limitations of pathfinding are not restricted to direct neighborhood-based algorithms. Even classical methods such as Dijkstra's algorithm, while theoretically exact on a graph, rely on the quality of the underlying graph representation of the built environment. When a continuous environment is discretized into a rectangular graph, distortions may occur: with only orthogonal connections, Dijkstra reproduces results similar to Von Neumann search, whereas with additional diagonal links it tends to reproduce Moore-like behavior.

To achieve path quality comparable to VSP, one would need to construct a graph with a very large number of connections per node, which would dramatically increase computational cost and complexity, making Dijkstra's already inefficient performance even worse. Furthermore, generating such a rich graph from a generic environment would itself require a complex algorithm, adding another layer of computational burden and potential sources of error. This highlights that the issue is not with Dijkstra's logic itself, but with the mismatch between graph abstraction and

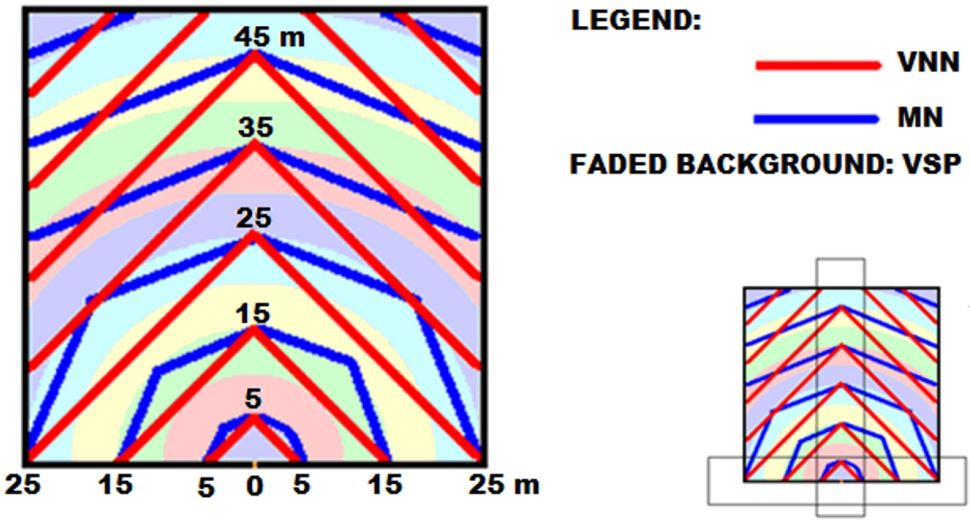
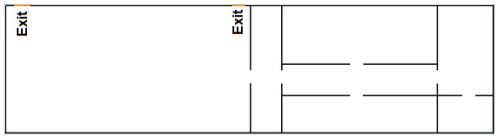
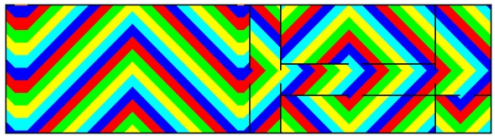


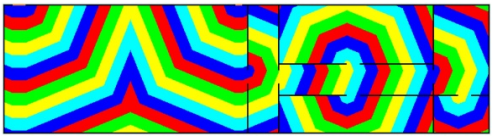
Figure 9: Stylized representation of some distance lines obtained from Figure 8. The red semi-lines were obtained from VNN, the blue semi-lines from MN, and the faded background represents the distance map obtained by the VSP, where each color range indicates a distance of 5 m from the exit.



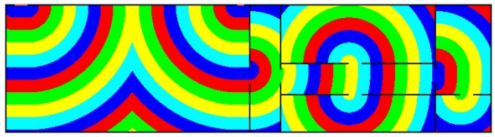
(a) Environment considered, with two exits (in orange) and several inners rooms.



(b) VNN: D_{max} of 30.3 m and D_{ave} of 13.9 m.



(c) MN: D_{max} of 27.5 m and D_{ave} of 12.4 m.



(d) VSP: D_{max} of 27.0 m and D_{ave} of 12.0 m (see [8]).

Figure 10: Representation of distance maps profile for an environment (a) with two independent exits (in orange): (b) for VNN, (c) for MN, and (d) for VSP. Each different colour range indicates a distance of 1 m from the nearest exit.

the geometric reality of built environments. In this sense, the VSP method avoids such distortions by operating directly on the geometry, ensuring both efficiency and fidelity in the resulting paths.

It is also important to clarify that VSP, VNN, MN, and Dijkstra do not share the same design objectives. VNN and MN prioritize simplicity and local uniform expansion, Dijkstra optimizes paths in an abstract graph, and VSP explicitly optimizes geometric fidelity. For this reason, comparisons among them are meant to highlight structural differences rather than suggest equivalence of purpose. That is, the fact that Dijkstra can approximate VNN or MN behaviour under specific connectivities does not imply that it can approximate VSP results under realistic grid constraints.

As discussed in Section 3, all lattice-based formulations introduce geometric approximations due to discretization. However, the sensitivity to this approximation is not uniform across methods. In neighbourhood-based searches (VNN/MN), discretization is embedded directly in the local expansion pattern, and its geometric effect is predictable and limited to the chosen first-order neighborhood. In contrast, when Dijkstra is applied to the same grid graph, its geometric fidelity depends entirely on the connectivity of that graph: with only four neighbours, it necessarily reproduces VNN behaviour; with eight neighbours, it approximates MN; improving geometric accuracy would require substantially denser connectivity. Such dense connectivity rapidly increases memory consumption and computational cost, making Dijkstra comparatively more sensitive to discretization as grid resolution grows. By comparison, VSP relies on geometric visibility rather than neighbourhood connectivity, retaining high geometric accuracy even on coarse grids.

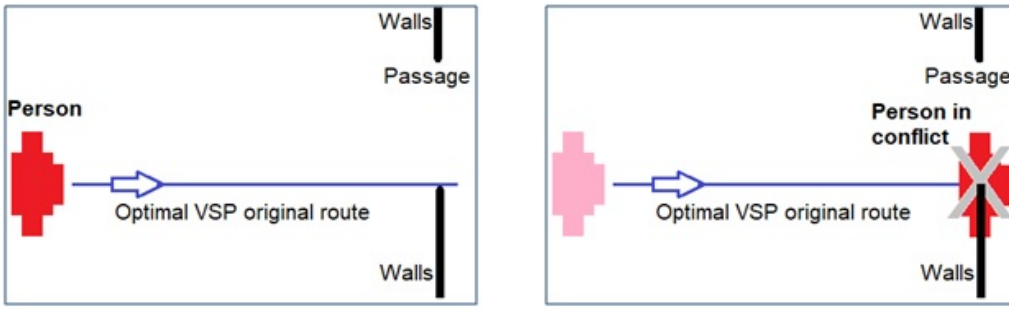
The quality of the results ensures that the profiles of the distance maps generated by the VSP algorithm are predominantly characterized by smooth, effectively differentiable curves, exhibiting gradual transitions rather than abrupt changes at the macroscopic level. This behavior can be likened to segments of circular arcs. This characteristic does not occur in the distance maps generated by the VNN or MN algorithms, which are macroscopically composed of straight lines and non-collinear segments [7]. Another advantageous aspect of the VSP algorithm is the impossibility of local minima occurring, except for the global minimum, as can happen in methods such as potential fields [20]. These characteristics contribute to the algorithm's simplicity and efficiency.

The precomputed traversal information provided by VSP could also be incorporated into hybrid architectures or learning-based evacuation models, e.g., as geometric priors for agent decision models or as constraints in metaheuristic search.

Finally, even if a classical method such as Dijkstra could hypothetically achieve high geometric accuracy for this application (which would require graph densities far beyond practical limits), this would not diminish the scientific value of presenting VSP. Algorithmic research routinely benefits from complementary approaches: a method may be optimal for one class of problems while another excels in a different structural setting. VSP contributes by offering a geometry-driven, globally consistent, and computationally efficient solution tailored to evacuation contexts—an area where classical graph-based algorithms have not been designed to operate.

4.2 Correcting the effect of body sizes

The route generated by the VSP algorithm is indeed very close to optimal, but it does not consider the effect of body size. If the resolution of the route is smaller than the body size, which is a typical situation in high-level modeling, it will be impossible for an agent to follow the route previously defined as the best by the VSP algorithm. This is due to constraints and collisions near fixed obstacles such as walls. Figure 11 illustrates this situation, where a person follows the defined route and collides with the wall. To avoid this, an alternative is to consider the influence of body size when generating the optimal route using the VSP algorithm.



(a) (b)

Figure 11: Representation of a person (in red) walking towards a passage, following the initially defined optimal VSP route (in blue), and hitting the wall (in black).

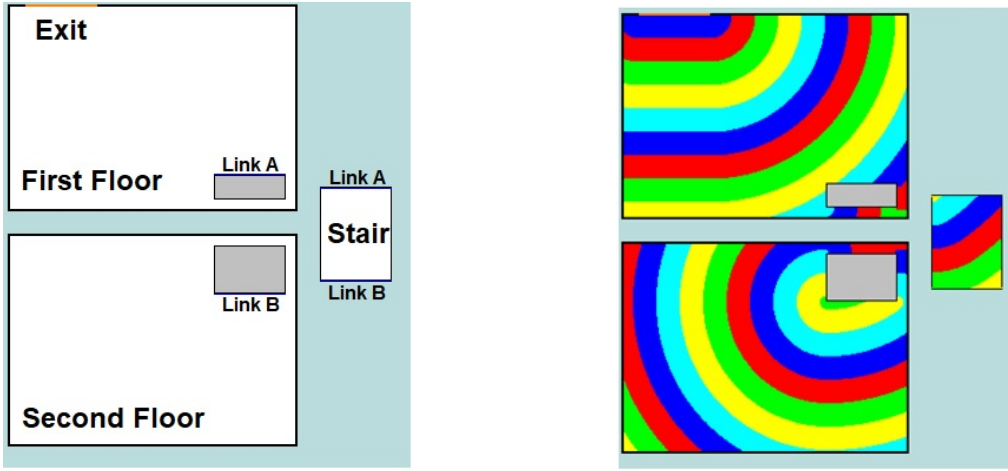
The proposed approach aims to implement the so-called 'shading effect' around obstacles in the discretized built environment. This shading involves marking cells around all fixed obstacles that are, within a linear distance from these obstacles, less than or equal to half the width between the shoulders of a person (considered the minimum space necessary to safely avoid physical collision) to be perceived by the VSP algorithm as fixed obstacles as well. Therefore, when the VSP algorithm is applied to this shaded environment, it will generate corrected optimal routes, avoiding overlaps, without requiring further modification or adaptation.

For the implementation of this shading approach, instead of conducting the distance mapping search from the exits (objectives) into the environment, the search will be initiated from the fixed obstacles themselves. Consequently, the resulting distance map will not provide the distance from any cell in the environment to the nearest exit; rather, it will provide the distance from any cell to the nearest wall (fixed obstacle).

With this established distance map of walls, a simple scan is performed on all free movement spaces (empty cells) in the discretized environment, which are at a distance equal to or less than the human-defined distance to avoid overlaps, as indicated on the previously generated distance map. These elements are then also treated as fixed obstacles. Subsequently, in this newly modified or shaded environment, the best distance map acquisition algorithm (in this case, VSP) is applied, with the real exits considered as objectives.

From this new distance map, optimal movement routes can be obtained, with no risk of physical overlap between individuals and fixed obstacles. The core features of the VSP algorithm remain unchanged. To illustrate the proposed method, a simple two-floor built environment is presented, with no internal obstructions except for the constraints imposed by the stairs and featuring a single exit on the first floor.

Figure 12a depicts this 3D environment according to a planned representation [10,46]. Walls are represented in black, open areas for movement in white, the exit on the first floor in orange, stair constraints in gray, dark blue straight lines represent stair links, and light blue regions designate areas disregarded for simulation. Figure 12b shows the distance map obtained using the original VSP algorithm, where each color band indicates a distance of 1 m from the considered objective, in this case, the exit.



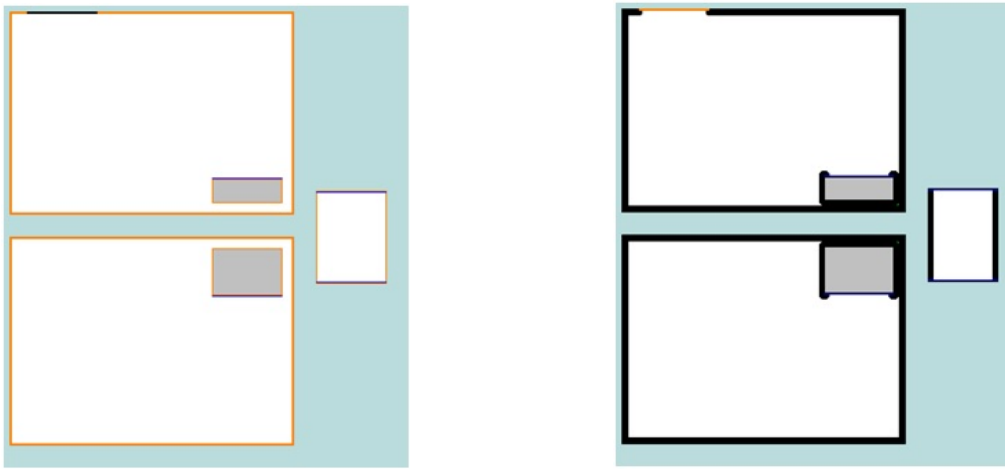
(a) (b)

Figure 12: (a) Planned representation of a simple 3D two-floor built environment, and (b) the original distance map obtained by the VSP algorithm.

The distance map in Figure 12b, as it stands, does not meet the requirements for simulations. It indicates paths that are too close to obstacles, potentially leading to undesirable overlaps if employed directly. For instance, note the narrow regions between the stairs and the walls. To address this, the distance map for generating the shaded environment should be created using the same VSP algorithm, but with the walls as the objectives.

Figure 13a illustrates this scenario, where all elements in the environment are identical to those in Figure 12a, except the walls are now represented in orange and the exit in black. Applying the VSP algorithm to this configuration with the walls as objectives produces a new distance map. Using this distance map, shading can be achieved by scanning all free movement points (empty cells) within a specified distance from the walls and treating them as walls (shadows). In this

example, the distance threshold for generating shadows was set to 25 cm, corresponding to the shoulder width of a typical agent. Figure 13b depicts the resulting shaded built environment.



(a) (b)

Figure 13: (a) Planned 3D environment with walls as objectives (represented in orange); (b) the same environment as Figure 12a, but properly shaded within a 25 cm distance radius.

Finally, the VSP algorithm is applied again to this shaded environment, resulting in the final distance map to be used by the subsequent simulation program. Figure 14 illustrates this distance map for the shaded environment.

As previously noted, the distance map presented in Figure 12b shows routes generated between the stairs and the walls. However, in the distance map shown in Figure 14, this region is blocked (in black) due to the shadowing effect, preventing its use for movement. This occurs because its width is less than 25 cm, which represents the shadow width considered in this example.

Another important point is the variation in body sizes among individuals [45], as well as special conditions, such as a person using a wheelchair [6], which must be considered for a more realistic simulation. Therefore, a single shading configuration may not fully meet all requirements. To achieve the desired realism, multiple distance maps could be generated, each accounting for a specific shading size suitable for different human dimensions and situations. Each simulated individual can then be associated with their respective distance map, a concept that can be easily implemented as previously described.

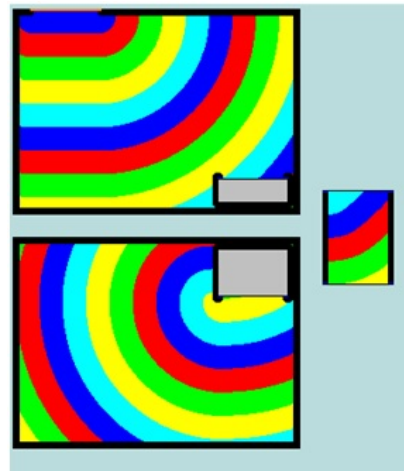


Figure 14: Final distance map obtained after considering the shading effect.

5 CONCLUSION

The Visibility Search Pathfinding algorithm (VSP), described in detail in this paper, offers several advantages, including logical simplicity, ease of computational implementation, broad applicability (independent of the environment's internal or external geometry, or number of floors, including 3D environments), no processing time demands in simulations (for non-dynamic applications), full automation (independent of human intervention), and the absence of multiple local minima.

Furthermore, compared to simpler search algorithms such as the Von Neumann Neighborhood (VNN) and the Moore Neighborhood (MN), the VSP consistently produces nearly optimal results in any built environment, with minimal errors. In contrast, the VNN and MN algorithms may yield, in some situations, differences of up to 40% (VNN) or 8% (MN) in distance calculations.

The distance maps and corresponding route maps generated by the VSP algorithm exhibit fewer artificial abrupt changes in direction within areas influenced by specific exits, with distance ranges that often resemble segments of circular arcs.

Yet, for more realistic simulations, the VSP algorithm can account for variations in body size or include a safety factor through a shading effect, ensuring optimal, safe paths to exits, without overlaps or collisions with fixed obstacles.

The detailed logic of the VSP algorithm presented here, along with its applications, represents a significant contribution and holds considerable potential to support high-performance human movement simulation software, particularly in evacuation scenarios, thereby aiding in the architectural design of safer built environments.

Acknowledgments

The authors would like to express their sincere gratitude to Prof. Dr. Paulo Eduardo Maciel de Almeida (*in memoriam*) for the valuable discussions related to this work and for his enduring friendship.

Data availability

All data generated or analysed during this study are included in this published article.

Associate editor: Kelly Cristina Poldi

REFERENCES

[1] J.M. Alkazzi & A.K. Okumura. A comprehensive review on leveraging machine learning for multi-agent path finding. *IEEE Access*, **12** (2024), 57390–57409. doi:10.1109/ACCESS.2024.3392305.

[2] M. Andayesh & F. Sadeghpour. A comparative study of different approaches for finding the shortest path on construction sites. *Procedia Engineering*, (85) (2014), 33–41. doi:10.1016/j.proeng.2014.10.526.

[3] S. Bandi & D. Thalmann. Path finding for human motion in virtual environments. *Computational Geometry*, **15**(1-3) (2000), 103–127. doi:10.1016/S0925-7721(99)00046-2.

[4] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner & R.F. Werneck. Route Planning in Transportation Networks. In L. Kliemann & P. Sanders (editors), "Algorithm Engineering: Selected Results and Surveys", volume 9220 of *Lecture Notes in Computer Science*. Springer, Cham (2016), p. 19–80. doi:10.1007/978-3-319-49487-6.

[5] E. Bosina, R. Roth & J. Meli. Estimating the Capacity of Railway Platforms and Stations. *Collective Dynamics*, **9** (2024), 1–7. doi:10.17815/CD.2024.158.

[6] K. Boyce. Safe evacuation for all - Fact or fantasy? Past experiences, current understanding and future challenges. *Fire Safety Journal*, **91** (2017), 28–40. doi:10.1016/j.firesaf.2017.05.004.

[7] H.C. Braga, G.F. Moita & P.E.M. Almeida. Comparação entre os Algoritmos de Busca pela Vizinhança de Von Neumann ou de Moore para Geração do Mapa de Distâncias em um Ambiente Construído. *Abakós*, **4**(2) (2016), 20–40. doi:10.5752/P.2316-9451.2016v4n2p20.

[8] H.C. Braga, G.F. Moita & P.E.M. Almeida. High Quality Pathfinder Algorithm to Be Used to Find the Best Route to the Nearest Exit of an Environment. In "Proceedings of the 8th International Conference on Pedestrian and Evacuation Dynamics (PED 2016)", volume 1. University of Science and Technology of China, Hefei, China (2016), p. 306–312.

[9] H.C. Braga, G.F. Moita & P.E.M. Almeida. Abordagem Ergonômica da Movimentação Humana, sua Natureza Complexa e Aspectos para sua Simulação. *Ergodesign & HCI*, **5**(Especial) (2017), 1–13. doi:10.22570/ergodesignhci.v5iEspecial.349.

[10] H.C. Braga, G.F. Moita & P.E.M. Almeida. Mapas de Distâncias para a Segurança contra Incêndio em Edifícios de Interesse Social. *PARC: Pesquisa em Arquitetura e Construção*, **8**(1) (2017), 32–45. doi:10.20396/parc.v8i1.8647977.

[11] H.C. Braga, G.F. Moita & P.E.M. Almeida. The Influence of the Location of Emergency Exits over the Distance to Be Covered to the Exit of an Environment. *Ambiente Construído*, **19**(2) (2019), 219–232. doi:10.1590/s1678-86212019000200318.

[12] H.C. Braga, G.F. Moita, F. Camargo & P.E.M. Almeida. Simulação da Movimentação de Pessoas em Situações de Emergência: Aspectos Ergonômicos e Computacionais com Autômatos Fuzzy e sua Aplicação ao Projeto Arquitetônico. *Ambiente Construído*, **14**(2) (2014), 61–77. doi:10.1590/S1678-86212014000200005.

[13] C. Burstedde, K. Klauck, A. Schadschneider & J. Zittartz. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A*, **295**(3-4) (2001), 507–525. doi:10.1016/S0378-4371(01)00141-8.

[14] M. Chimani & K. Klein. Algorithm Engineering: Concepts and Practice. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete & M. Preuss (editors), "Experimental Methods for the Analysis of Optimization Algorithms". Springer, Berlin, Heidelberg (2010), p. 131–176. doi:10.1007/978-3-642-02538-9_6.

[15] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numer Math (Heidelb)*, **1**(1) (1959), 269–271. doi:10.1007/BF01386390.

[16] D. Foead, A. Ghifari, M.B. Kusuma, N. Hanafiah & E. Gunawan. A systematic literature review of A* pathfinding. *Procedia Computer Science*, (2021), 507–514. doi:10.1016/j.procs.2021.01.034.

[17] A.V. Goldberg & C. Harrelson. Computing the Shortest Path: A* search meets graph theory. In "ACM-SIAM symposium on Discrete algorithms (SODA '05)". Society for Industrial and Applied Mathematics (2005), p. 156–165.

[18] M. Haghani. Optimizing crowd evacuation: mathematical, architectural and behavior approaches. *Safety Science*, **128** (2020), 104745. doi:10.1016/j.ssci.2020.104745.

[19] P.E. Hart, N.J. Nilsson & B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, **4**(2) (1968), 100–107. doi:10.1109/TSSC.1968.300136.

[20] L. He, X.H. Li & J. Lu. A Potential Field Model Avoiding Local Minimum in Pedestrian Simulation. *Applied Mechanics and Materials*, **380–384** (2013), 1660–1663. doi:10.4028/www.scientific.net/AMM.380-384.1660.

[21] D. Helbing, I.J. Farkas & T. Vicsek. Simulating dynamical features of escape panic. *Nature*, **407**(6803) (2000), 487–490. doi:10.1038/35035023.

[22] J. Hershberger & S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, **28**(6) (1999), 2215–2256. doi:10.1137/S0097539795289604.

[23] A. Kherrour, M. Robol, M. Roveri & P. Giorgini. A multi-algorithm pathfinding method: exploiting performance variations for enhanced efficiency. *Annals of Mathematics and Artificial Intelligence*, **93** (2025), 569–588. doi:10.1007/s10472-024-09957-3.

[24] D.C.C. Kowaltowski & D.C. Moreira. As pesquisas sobre o processo de projeto em arquitetura. *Revista Projetar - Projeto e Percepção do Ambiente*, **1**(1) (2016), 42–52. doi:10.21680/2448-296X.2016v1n1ID18495.

[25] T. Kretz & M. Schreckenberg. F.A.S.T. – Floor Field and Agent Based Simulation Tool. In E. Chung & A.G. Dumont (editors), "Transport Simulation: Beyond Traditional Approaches". CRC Press, Lausanne, Switzerland (2009), chapter 8, p. 125–135. doi:10.1201/9780429093258-8.

[26] E.D. Kuligowski. Predicting human behavior during fires. *Fire Technology*, **49** (2013), 101–120. doi:10.1007/s10694-011-0245-6.

[27] S.R. Lawande, G. Jasmine, J. Anbrasi & L.I. Izhar. A systematic review and analysis of intelligence-based pathfinding algorithms in the field of video games. *Applied Sciences*, **12**(11) (2022), 1–30. doi:10.3390/app12115499.

[28] M. Liui, G. Lui & O. Yoshinao. Analyzing factors causing deadlock events of bi-directional pedestrian fow when moving on stairs using a personal space model. *Scientific Reports*, **14**(1) (2024), 10847. doi:10.1038/s41598-024-61007-48.

[29] G. Ma, Y. Wang & S. Jiang. Optimization of Building Exit Layout: Combining Exit Decisions of Evacuees. *Advances in Civil Engineering*, **2021** (2021). doi:10.1155/2021/6622661.

[30] P. Maristany de las Casas, L. Kraus, A. Sedeño-Noda & R. Borndörfer. Targeted multiobjective Dijkstra algorithm. *Networks*, **82**(3) (2023), 277–298. doi:10.1002/net.22174.

[31] C.M. Mayr, S. Schuhbäck, L. Wischhof & G. Köster. Analysis of information dissemination through direct communication in a moving crowd. *Safety Science*, **142** (2021), 105386. doi:10.1016/j.ssci.2021.105386.

[32] M.E. Miyombo, Y. Liu, C.M. Mulenga, A. Siamulonga, M.C. Kabanda, P. Shaba, C. Xi & A. Ayodeji. Optimal path planning in a real-world radioactive environment: A comparative study of A-star and Dijkstra algorithms. *Nuclear Engineering and Design*, **420** (2024), 113039. doi:10.1016/j.nucengdes.2024.113039.

[33] M. Moussaïd, D. Helbing & G. Theraulaz. How simple rules determine pedestrian behavior and crowd disasters. *PNAS*, **108**(17) (2011), 6884–6888. doi:10.1073/pnas.1016507108.

[34] N. Pelechano, J. Allbeck & N. Badler. "Virtual Crowds: methods, simulation, and control". Morgan & Claypool Publishers (2008). doi:10.2200/S00123ED1V01Y200808CGR008.

[35] N. Pelechano, J. Allbeck, M. Kapadia & N. Badler. "Simulating heterogeneous crowds with interactive behaviors". CRC Press (2016). doi:10.1201/9781315370071.

[36] J. Porzycki & J. Was. Modeling spatial patterns in a moving crowd of people using data-driven approach - A concept of interplay floor field. *Safety Science*, **167** (2023), 106266. doi:10.1016/j.ssci.2023.106266.

[37] A. Rafiq, T.A.A. Kadir & S.N. Ihsan. Pathfinding algorithms in game development. *IOP Conf. Series: Materials Science and Engineering*, **769** (2016), 012021. doi:10.1088/1757-899X/769/1/012021.

[38] C. Rogsch. The Influence of Moore and von-Neumann Neighbourhood on the Dynamics of Pedestrian Movement. In "Traffic and Granular Flow '15", volume 3. Springer International Publishing (2016), p. 129–136. doi:10.1007/978-3-319-33482-0_17.

[39] A. Schadschneider. I'm a football fan ... get me out of here. *Physics World*, **23**(7) (2010), 21–25. doi:10.1088/2058-7058/23/07/30.

[40] M.J. Seitz & G. Köster. Natural discretization of pedestrian movement in continuous space. *Phys Rev E*, **86**(4) (2012), 046108. doi:10.1103/PhysRevE.86.046108.

[41] A. Sieben, J. Schumann & A. Seyfried. Collective phenomena in crowds — Where pedestrian dynamics need social psychology. *PLoS ONE*, **12**(6) (2017), e0177328. doi:10.1371/journal.pone.0177328.

[42] A. Sieben & A. Seyfried. Inside a life-threatening crowd: Analysis of the Love Parade disaster from the perspective of eyewitnesses. *Safety Science*, **166** (2023), 106229. doi:10.1016/j.ssci.2023.106229.

[43] A.R. Soltani, H. Tawfik, J.Y. Goulermas & T. Fernando. Path planning in construction sites: performance evaluation of the Dijkstra, A^*, and GA search algorithms. *Advanced Engineering Informatics*, **16**(4) (2002), 291–303. doi:10.1016/S1474-0346(03)00018-1.

[44] A.M. Thompson. "The navigation system of the JPL robot". Propulsion Laboratory - NASA (1977).

[45] P. Thompson, D. Nilsson, K. Boyce & D. McGrath. Evacuation models are running out of time. *Fire Safety Journal*, **78** (2015), 251–261. doi:10.1016/j.firesaf.2015.09.004.

[46] P.A. Thompson & E.W. Marchant. A computer model for the evacuation of large building populations. *Fire Safety Journal*, **24**(2) (1995), 131–148. doi:10.1016/0379-7112(95)00019-P.

[47] S. Tjiharjadi, S. Razali & H.A. Sulaiman. A systematic literature review of multi-pathfinding for maze research. *Journal of Advances in Informatic Technology*, **13**(4) (2022), 358–367. doi:10.12720/jait.13.4.358-367.

[48] Y. Tong & N.W.F. Bode. An investigation of how context affects the response of pedestrian to the movement of others. *Safety Science*, **157** (2023), 105919. doi:10.1016/j.ssci.2022.105919.

[49] T. Van Vuren & G.R.M. Jansen. Recent developments in path finding algorithms: a review. *Transportation Planning and Technology*, **12**(1) (1988), 57–71. doi:10.1080/03081068808717360.

[50] M.R. Wayahdi, S.H.N. Ginting & D. e Syahputra. Greedy, A-Star, and Dijkstra's Algorithms in Finding Shortest. *International Journal of Advances in Data and Information Systems*, **2**(1) (2021), 45–52. doi:10.25008/ijadis.v2i1.1206.

[51] X. Wei, Z. Lou, H. Song, H. Qin & H. Yao. Exploring the impacts of exit structures on evacuation efficiency. *Fire*, **6**(12) (2023), 462. doi:10.3390/fire6120462.

[52] S. Wolfram. Statistical mechanic of cellular automata. *Reviews of Modern Physics*, **55**(3) (1983), 601–644. doi:10.1103/RevModPhys.55.601.

[53] D.A. Zaitsev. A generalized neighborhood for cellular automata. *Theor Comput Sci*, **666** (2017), 21–35. doi:10.1016/j.tcs.2016.11.002.

[54] F. Zanlungo, C. Feliciani, Z. Yücel, K. Nishinari & T. Kanda. Macroscopic and microscopic dynamics of a pedestrian cross-flow: Part II, modelling. *Safety Science*, **158** (2023), 105969. doi:10.1016/j.ssci.2022.105969.

[55] F. Zanlungo, T. Ikeda & T. Kanda. A microscopic "Social Norm" model to obtain realistic macroscopic velocity and pedestrian distributions. *PlosOne*, **7**(12) (2012), e50720. doi:10.1371/journal.pone.0050720.

[56] N. Zhu, J. Wang & J. Shi. Application of pedestrian simulation in Olympic Games. *Journal of Transportation Systems Engineering and Information Technology*, **8**(6) (2008), 85–90. doi:10.1016/S1570-6672(09)60007-6.

How to cite

H.C. Braga & G.F. Moita. Distance Map Associated with Tangent Search as a Pathfinding Strategy: achieving high-quality and safe route maps for crowed movement in built environments. *Trends in Computational and Applied Mathematics*, **27**(2026), e01845. doi: 10.5540/tcam.2026.027.e01845.

SUPPLEMENTARY RESOURCES

Demonstration videos related to the present study are available within a broader playlist associated with the Fuga framework, a computational tool for pedestrian movement simulation:
Fuga demonstration playlist.
`https://www.youtube.com/playlist?list=PLtNrPc3_xkxDKX7BiOr1dSZfSQJHbh8Zm`

APPENDIX

In the following pseudocode, we state:

- EM (Environment Matrix) is the matrix that represents the discretized built environment,
- OM (Objective Matrix) is the matrix that contains the coordinates of all final targets (e.g., exits),
- DM (Distance Map) is the matrix with the shortest distances between all points and the nearest objective,
- RM (Route Map) is the matrix that contains the best route to be followed from all points to the nearest objective,
- The values in all matrices represent cell counts and have not been converted to SI units (this can be done in a later step using the resolution r), and
- N_F^4 is the set of all 4-neighbors through the faces of a generic cell at (x,y):
  $N_F^4 = \{(x+1,y),(x-1,y),(x,y+1),(x,y-1)\}$.

Algorithm 1 Pseudocode for the search core of the VSP algorithm.

Input: EM, OM

Output: DM, RM

$MBP \leftarrow OM$ // Initialize the search matrix with the final objectives
$DM \leftarrow 0$ // Initialize the Distance Map matrix with the same dimensions as EM, with its cells initially set to 0
$RM \leftarrow (0,0)$ // Initialize the Route Map matrix with the same dimensions as EM, with its cells initially set to (0,0)
while $MBP \neq null$ **do**
    // When MBP = *null* there will be no more cells to search
    $MBS \leftarrow null$
    for $j \leftarrow 1$ **to** length(MBP) **do**
        // Will search through all objectives
        $(x_i, y_i) \leftarrow$ coordinates in $MBP(j)$ $(x_r, y_r) \leftarrow$ coordinates in $RM(x_i, y_i)$
        for $k \leftarrow 1$ **to** *4* **do**
            // searches neighboring face cells (N_F^4) from $EM(x_i, y_i)$
            $(x_k, y_k) \leftarrow$ coordinate of (N_F^4) from $EM(x_i, y_i)$
            if $EM(x_k, y_k)$ *indicates a valid coordinate* **then**
                $MBS \leftarrow MBS \cup EM(x_k, y_k)$
                if *there is a straight unobstructed line between* $EM(x_k, y_k)$ *and* $EM(x_r, y_r)$ **then**
                    $step \leftarrow$ Euclidean distance between $EM(x_k, y_k)$ and $EM(x_r, y_r)$
                    $DM(x_k, y_k) \leftarrow DM(x_r, y_r) + step$ // Equation (3.1)
                    $RM(x_k, y_k) \leftarrow RM(x_r, y_r)$ // Keep the reference
                end
                else
                    $DM(x_k, y_k) \leftarrow DM(x_i, y_i) + 1$
                    $RM(x_k, y_k) \leftarrow RM(x_i, y_i)$ // Change the reference
                end
            end
        end
    end
    $MBP \leftarrow MBS$
end