2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

An Efficient Solution to the 2D Visibility Problem in Cartesian Grid Maps and its Application in Heuristic Path Planning

Ibrahim Ibrahim†, Joris Gillis†, Wilm Decré†, Jan Swevers†

***Abstract*— This paper introduces a novel, lightweight method to solve the visibility problem for 2D grids. The proposed method evaluates the existence of lines-of-sight from a source point to all other grid cells in a single pass with no preprocessing and independently of the number and shape of obstacles. It has a compute and memory complexity of $\mathcal{O}(n)$, where $n = n_x \times n_y$ is the size of the grid, and requires at most ten arithmetic operations per grid cell. In the proposed approach, we use a linear first-order hyperbolic partial differential equation to transport the visibility quantity in all directions. In order to accomplish that, we use an entropy-satisfying upwind scheme that converges to the true visibility polygon as the step size goes to zero. This dynamic-programming approach allows the evaluation of visibility for an entire grid orders of magnitude faster than typical ray-casting algorithms. We provide a practical application of our proposed algorithm by posing the visibility quantity as a heuristic and implementing a deterministic, local-minima-free path planner, setting apart the proposed planner from traditional methods. Lastly, we provide necessary algorithms and an open-source implementation of the proposed methods.**

I. Introduction

Visibility is crucial for robot applications as it enables the robot to make informed decisions and allows it to interact with the world in a meaningful way. Tasks such as obstacle avoidance and path planning require knowledge about the regions accessible from the robot's position at any time instant, while other tasks such as object recognition, tracking, and manipulation necessitate maintaining a line-of-sight with the target at all times. The robot's visibility of a point in an environment — or the robot's accessibility to a point — can be defined as the existence of an uninterrupted line-of-sight between the robot and the point. The set of points that are visible to the robot at any time instant constitutes the robot's visibility polygon. An example of a visibility polygon in a cluttered environment is illustrated in Fig. 1. In robotic applications, there are two predominant approaches for representing environments: grid maps and polygonal domains. In this introduction, we delve into the issue of visibility within each of these representations.

Polygonal domains constitute a simplified representation of environments where the robot, the obstacles, and the internal and external boundaries are assumed to be polygons [1]. This assumption, although restrictive, makes tackling the problem of 2D visibility tractable by working with polygon vertices to construct visibility polygons and/or visibility graphs. As such, 2D visibility has predominantly been explored within the context of polygonal domains, as extensively documented in the literature [2], [3], [4], [5]. Nonetheless, it is essential to acknowledge the inherent limitations of these methods: they all scale with the number of obstacle vertices and edges. Furthermore, these methods often exhibit determinism and static behavior, making them less suitable for dynamic and probabilistic environments, which impose substantial computational and memory burdens. Lastly, it is noteworthy that the prevailing techniques for constructing visibility polygons and graphs cannot be readily extended to 3D environments, primarily due to the intractability associated with handling vertices in three dimensions.

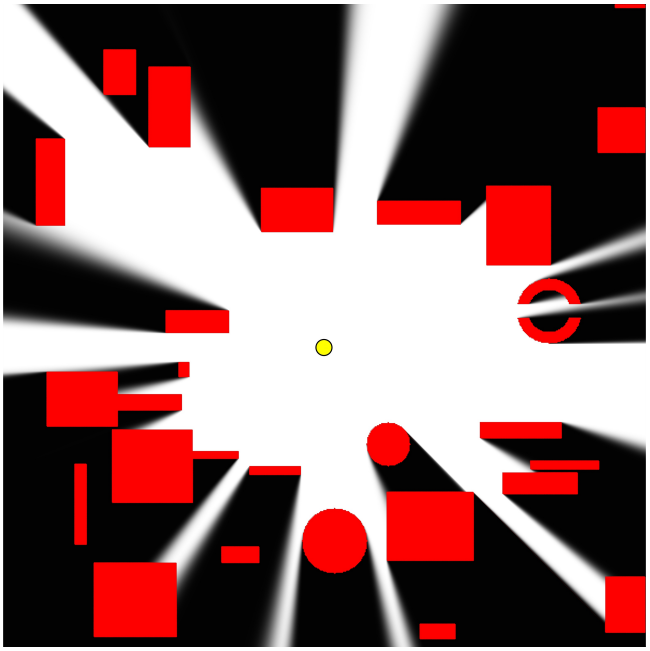

Fig. 1. The 2D visibility polygon $\mathcal{U}$ computed using our algorithm. The robot position is the light source shown as a yellow dot. The white region is visible to the robot, constituting the visibility polygon. Dark regions are invisible to the robot. Obstacles are shown in red. 1000×1000 grid.

Grid maps find favor among roboticists due to their inherent simplicity, user-friendly characteristics, and robustness. They provide granularity and precision, particularly in tasks demanding meticulous localization, path planning, and obstacle avoidance. They are also easy to visualize and interpret, particularly in dynamic environments. However, the realm of 2D visibility in grid map representations remains a relatively underexplored domain in the current literature. Existing methods often focus on evaluating visibility for individual

This work has been carried out within the framework of Flanders Make's SBO project ARENA (Agile & Reliable Navigation). Flanders Make is the Flemish strategic research centre for the manufacturing industry.

† Motion Estimation Control and Optimization Lab, KU Leuven.

pixels one at a time. However, certain applications necessitate a comprehensive understanding of visibility across the entire grid map, including autonomous path planning, obstacle avoidance, sensor placement, exploration, and area coverage. Consequently, the need arises to quantify visibility for all grid map pixels. Current methodologies rely on ray-shooting, ray casting, or voxel traversal algorithms to compute visibility grid maps, but these approaches suffer from inherent inefficiencies due to the necessity of conducting line-of-sight checks for each grid cell. Moreover, some of these methods exhibit limitations concerning scalability and complexity.

In light of these considerations, this paper addresses the understudied realm of 2D visibility in grid map representations, aiming to contribute novel solutions to this important challenge. To this end, we introduce an efficient method that computes the visibility grid map in a single pass with no assumptions, no preprocessing, and independently of the number and the shape of obstacles. It uses only raw information from a deterministic or probabilistic occupancy grid, which is either given or is constructed from sensor data such as LIDAR. We accomplish that by transporting the visibility quantity using an entropy-preserving, stable, and converging upwind scheme solution to a linear first-order hyperbolic Partial Differential Equation (PDE). The visibility quantity itself is a value ranging from 1, meaning fully visible, to 0, meaning fully invisible. This method extends to 3D and extends to curves, allowing us to quantify *curves-of-sight* — the existence of an unobstructed curve, of a generic shape, between the robot and any other point/s. The result is a *curvilinear polygon* rather than a visibility polygon.

Moreover, we use the produced visibility quantity as a heuristic for path planning as an application to our visibility grid map algorithm. Even though visibility has long been used in planning methods, particularly in polygonal domains, to the best of our knowledge, it has not been directly embedded as a guiding heuristic in a Dijkstra-like algorithm, similar to the distance heuristic that guides an A^* path planner, within the realm of grid maps. The resulting planner builds a path by iteratively placing a new waypoint that is *barely* visible to its parent waypoint while minimizing the overall path length. The heuristic can also be tuned to favor exploration of unlit/unseen regions over driving towards the target. The map is deemed completely explored once each accessible grid cell has a line-of-sight connection with at least one placed waypoint.

In this paper, we use the terms "visibility polygon" and "visibility grid map" interchangeably. The contribution of this paper is threefold:

- Posing visibility as a transportable quantity and transporting it using a stable and converging solution to a linear first-order hyperbolic PDE, allowing us to compute the visibility grid map and the *curvilinear polygon* efficiently for single or multiple light sources.
- As an interesting application to our visibility grid map algorithm, we introduce the visibility heuristic in a way that drives a greedy any-angle path planner towards the target. A planner is said to be any-angle if it allows the turns in the path between two grid points to have any angle. This results in a direct point-to-point path that traverses open areas without being restricted to predefined neighbouring directions/angles at each step. The proposed planner can instead be used to explore the map, ensuring that every point in the grid is seen by (connected to) at least one waypoint.
- Algorithms to compute the visibility polygon and to implement the heuristic path planner as well as numerical experimental results and open-source implementations https://github.com/IbrahimSquared/visibility-heuristic-path-planner.

II. Related Work

The visibility problem is a basic computational geometry problem [6], [7], [8], that has been studied extensively and has multiple applications including computer graphics [9].

A. Polygonal Domains

The computation of the visibility polygon of a point in a simple polygon as studied by Davis and Benedikt [2] has an $\mathcal{O}(V^2)$ time complexity where V is the number of vertices. More efficient algorithms came along afterwards which simplified it down to $\mathcal{O}(V)$ [3]. For the case of a simple polygon with holes, Asano et al. [4] provided a solution that has a $\mathcal{O}(V \log V)$ time complexity. A variant that scales with the number of obstacles h also as $\mathcal{O}(V + h \log h)$ was presented by Heffernan et al. [5].

The solution to answering visibility polygon queries for a generic point, called visibility polygon query problem, is approached by preprocessing the environment polygon and constructing data structures which make querying visibility polygons more efficient. For example, Bose et al. [10] address preprocessing in $\mathcal{O}(V^3 \log V)$ time with data structures of size $\mathcal{O}(V^3)$ to answer visibility polygon queries in $\mathcal{O}(\log V + k)$ where k is the size of the visibility polygon. Recovering the number of vertices visible from the source in this approach requires an added $\mathcal{O}(\log V)$ time.

Visibility graph path planning assumes that obstacles are 2D polygons and creates the visibility graph by using knowledge on obstacle vertices and edges. It then finds the optimal path using efficient graph path planners such as A^* or Dijkstra [11]. There exist multiple ways of constructing the visibility graph of a polygonal domain. Pocciola and Vegter [12] provide a method that constructs a visibility graph in $\mathcal{O}(V + E)$ time and $\mathcal{O}(V)$ space where E is the number of obstacle edges. A method that performs triangulation of the free space of the polygonal domain first was put forth by Kapoor and Maheshwari [13] that has a compute complexity of $\mathcal{O}(T + E + h \log V)$ time and $\mathcal{O}(E)$ space where T is the triangulation time. Other methods relying on triangulation can handle a large number of vertices and edges dynamically [14] but share similar complexity limitations.

B. Grid Map Representation

Ray casting, ray shooting, and voxel traversal are the most widely adopted methods to quantify visibility, albeit in a

binary way, for robot applications in grid environments. All said methods visit each cell more than once and therefore perform redundant computations. For example, in Next-Best-View (NBV) planning schemes, where the purpose is to plan the next camera pose in a way that maximizes the likelihood of obtaining the highest amount of information from sensors such as visibility and matching of image features, authors often fallback to using ray-casting [15], [16], [17]. In [18], the authors perform autonomous scene exploration by encoding visibility or lack thereof in a dual OcTree structure by simulating ray-casting as part of their NBV planning approach.

The notions of visibility and obstruction are also commonly required for multi-agent hide-and-seek or pursuit-evasion applications. In [19], the authors endow the agents with sight by using an array of 30 simulated rays arranged uniformly around the agent that cover a cone of interest, representing the agent's field of view, similar to a simulated lidar. Tandom and Karlapalem [20] also trace uniformly spaced rays that emit from agents to represent the agent's associated visibility region in simulation. Agents not within visibility cones are masked. One commonly adopted voxel traversal algorithm was presented by Amanatides et al. [21].

Occlusion avoidance is another common theme for utilizing the notions of visibility and obstruction. The point in such applications is for the robot to maintain a line-of-sight with the target of interest, e.g., when using manipulators for videography. In [22], the authors present a single pass method to provide a differentiable quantification of visibility in the form of a scalar field for both 2D and 3D grids, but such a method only provides an approximation of visibility which fails in tight or oblique corridors. The formulation in [22] is not formal and is based on a simple geometric approach that fails in some cases. As such, it is deemed unreliable and not robust. On the other hand, Nageli et al. [23] implement a fast visibility check that is limited to evaluating a single point against an ellipsoidal approximation of obstacles. Lastly, Allaire et al. [24] tackle the problem of visibility, or accessibility as they call it, by evaluating the Eikonal equation to every single point and comparing it to the geodesic shortest distance. They extend this to accessibility from a surface by discretizing the surface to a set of points and repeating the process for each point. In order to smoothen their binary solution, they consider obstacles as smooth Heaviside functions. Such an approach entailing a multi-step process is highly computationally demanding.

In [25], Farias and Kallmann present a method to compute optimal path maps with a $\mathcal{O}(\frac{n}{c}V^2)$ complexity where c is the number of GPU cores utilized for the computation. In order to compute the visibility polygon, they use a geometry shader to draw into a stencil buffer three triangles behind every obstacle line-segment that is front-facing with respect to every source point. This process is repeated for all grid cells n for every obstacle line-segment and for every source point. As such, their proposed method scales with the number of obstacles in addition to the number of grid cells n. Moreover, such a procedure involves costly algebra routines and is not limited to simple arithmetic operations as our method.

Once again, this paper proposes a method to compute the visibility grid map for grid-based environment representation. The proposed visibility algorithm is formalized and is independent of V, h, and E. Most importantly, it scales linearly with the size of the grid $n = n_x \times n_y$. To the best of our knowledge, such a formal and efficient method that evaluates visibility for grid maps does not exist in literature.

III. Visibility

A. Definition

Numerically, visibility is a scalar property of a point in a scene consisting of light sources and obstacles, describing the point's degree of illumination by a certain source with a real number in the closed interval $\mathcal{V} = [0, 1]$.

B. Formulation

Consider the linear first-order hyperbolic PDE of the form

$$a(x,y)\frac{\partial u(x,y)}{\partial y} + b(x,y)\frac{\partial u(x,y)}{\partial x} = 0, \tag{1}$$

where $u(x, y) \in V$ is the visibility at every grid position (x, y) and $a(x, y)$ and $b(x, y)$ are non-zero scalar components of the vector field describing the direction of rays starting from the source and going out in all directions. This formulation can be further simplified to

$$\frac{\partial u(x,y)}{\partial y} + c(x,y)\frac{\partial u(x,y)}{\partial x} = 0, \tag{2}$$

where $c(x, y)$ is the quotient of $b(x, y)$ by $a(x, y)$ — a scalar field describing the direction of rays. Equation (2) is often referred to as the linear transport equation or the advection equation, which is a special case of the former [26].

C. Solution

For our case, the vector field describing the directions of the rays has the normalized components

$$a(x,y) = \frac{x}{\sqrt{x^2+y^2}}, \tag{3}$$

$$b(x,y) = \frac{y}{\sqrt{x^2+y^2}}, \tag{4}$$

meaning our scalar field becomes

$$c(x,y) = \frac{y}{x}. \tag{5}$$

The analytic solution for (2) is trivial, but a numerical solution applicable to grids may in some cases be challenging. We adopt the first-order upwind scheme [27] which is a numerical discretization method for solving hyperbolic PDEs. Such a scheme is based on the idea of using the direction of the flow to determine the direction of the numerical approximations. It approximates the solution by taking into account only the information from the side of the flow from which the flow is coming. Applying it to (2) yields

$$\frac{u_i^{j+1} - u_i^j}{\Delta y} + c(x,y)\frac{u_i^j - u_{i-1}^j}{\Delta x} = 0, \tag{6}$$

where i and j are the steps along x and y respectively and Δx and Δy are the step sizes in both dimensions respectively.

The light source is initially at $(i, j) = (0, 0)$ in this case. From (6) we write our update equation as

$$u_i^{j+1} = u_i^j - C(u_i^j - u_{i-1}^j), \tag{7}$$

where C is the dimensionless **Courant** number defined as

$$C = \frac{c(x, y)\Delta y}{\Delta x}. \tag{8}$$

For our PDE to converge and maintain stability, it is *necessary* to satisfy the Courant–Friedrichs–Lewy (CFL) condition [28]

$$C = \frac{c(x, y)\Delta y}{\Delta x} \leq C_{max}, \tag{9}$$

where C_{max} depends on the discretization technique. For an upwind scheme, we have stability for $C_{max} = 1$ and $c(x, y) \geq 0$ [29]. It is then *sufficient* to perform numerical experiments to validate stability and convergence.

For the case where we have a uniform grid with $\Delta x = \Delta y = 1$, condition (9) will not be held whenever $y > x$ in (5). In such a case, we adapt our update step (7) as

$$u_{i+1}^j = u_i^j - \frac{1}{C}(u_i^j - u_i^{j-1}). \tag{10}$$

Equations (7) and (10) allow us to propagate initial visibility values, but they do not account for internal obstacles/boundaries which themselves affect visibility. Therefore, after transporting visibility into a new cell, we multiply the resulting value with the complement of the occupancy value held at the cell's coordinates (e.g., if a cell (i, j) is 70% occupied $\implies$ it has a 30% occupancy complement, or visibility). Lastly, we treat the cases where $x = 0$ and $y = 0$ as special boundary conditions, allowing us to write the complete Alg. 1 for the first quadrant relative to the light source position. Other quadrants can be updated in the same fashion with the difference being some sign and bound limit changes. We increment the indices at a step size in each dimension, meaning Δx and Δy are already factored in Alg. 1. The initial light strength can be set as well as a decay factor α that diminishes the visibility exponentially.

Algorithm 1 Visibility Algorithm for the First Quadrant

Inputs: $\mathcal{O} \leftarrow$ Occupancy grid complement
$(l_x, l_y) \leftarrow$ Light source position
$(n_x, n_y) \leftarrow$ Grid dimensions
LightStrength $\leftarrow$ Light source strength, V
$\alpha \leftarrow$ Visibility decay factor, V
Output: $\mathcal{U} \leftarrow$ Visibility polygon, $V^{n_x \times n_y}$

procedure GETVISIBILITYPOLYGON(Inputs)
  $m_x = n_x - l_x$, $m_y = n_y - l_y$
  for $i = 0$ to m_x and $j = 0$ to m_y **do**
    $p_x = l_x + i$, $p_y = l_y + j$
    if $i = 0 \wedge j = 0$ **then**
      $v \leftarrow$ LightStrength
    else if $i = 0$ **then**
      $v \leftarrow \mathcal{U}_{p_x}^{p_y - 1}$
    else if $j = 0$ **then**
      $v \leftarrow \mathcal{U}_{p_x - 1}^{p_y}$
    else if $i > j$ **then**
      $c \leftarrow \frac{p_y - l_y}{p_x - l_x}$
      $v \leftarrow \mathcal{U}_{p_x - 1}^{p_y} - \frac{c\Delta y}{\Delta x}(\mathcal{U}_{p_x - 1}^{p_y} - \mathcal{U}_{p_x - 1}^{p_y - 1})$
    else if $i < j$ **then**
      $c \leftarrow \frac{p_x - l_x}{p_y - l_y}$
      $v \leftarrow \mathcal{U}_{p_x}^{p_y - 1} - \frac{c\Delta x}{\Delta y}(\mathcal{U}_{p_x}^{p_y - 1} - \mathcal{U}_{p_x - 1}^{p_y - 1})$
    end if
    $\mathcal{U}_{p_x}^{p_y} \leftarrow v \times \mathcal{O}_{p_x}^{p_y} \times \alpha$
  end for
  return $\mathcal{U}$
end procedure

D. Properties, Efficiency and Extension to Curves and 3D

The resultant $\mathcal{U}$ from Alg. 1 contains values ranging from 0 to 1 representing the visibility quantity. Using rough step sizes results in dispersion, which arises naturally from the upwind scheme, but such a dispersion decreases as the step sizes are refined. We can obtain the binary visibility polygon with the operation $\tilde{\mathcal{U}} = \mathcal{U} \geq 0.5$. The threshold value 0.5 can be set as desirable. A lower threshold means an under-estimation of the visibility polygon, whereas a higher one means an over-estimation. For the path planner application discussed in Section IV, a step size of 1 provides a sufficiently good approximation of the visibility polygon.

Alg. 1 requires at most 2 additions, 4 subtractions, 2 divisions, and 2 multiplications operations per grid cell and requires one array of the grid size in memory, making it extremely cheap and scalable relative to methods in literature and attractive for applications with low computational and memory capacity. A sample 100×100 grid can be evaluated at a constant time of around $18\,\mu s$ in C++ on an Intel® Core™ i9–13980HX , which scales *linearly* with the grid size $n_x \times n_y$, unlike similar methods in literature that additionally scale with the number of obstacle vertices in the environment. In order to demonstrate the latter, we carried out 60 sets of experiments, with each set comprising 20 repetitions, covering a range of logarithmically-sampled environment sizes. The range of sizes varied from 50×50 to 5000×5000. The resulting C++ average compute time is shown in Fig. 2 and is **independent** of the number and shape of obstacles. This single-pass dynamic-programming approach is also considerably faster and more efficient than performing typical line-of-sight checks to every single pixel in the grid map. The proposed method significantly reduces the number of required floating-point operations for evaluating visibility across the entire grid compared to adopting a voxel traversal algorithm like [21]. We illustrate the result of performing the same set of experiments using typical ray-casting in Fig. 2. Our proposed method runs up to $100\times$ faster than ray-casting for an empty 1000×1000 grid, and up to $400\times$ faster for a 5000×5000 one. Such speedups decrease for denser environments since ray-casting terminates on collisions. In this case, one can adapt (1) with termination conditions. More implementation details are available in our provided C++ open source code.

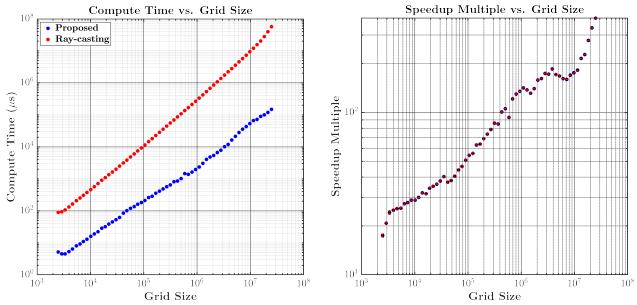

Fig. 2. Log-log plot of average compute time of Alg. 1 (left) with respect to the grid size $n_x \times n_y$ and in comparison to that of ray-casting. Error bars are insignificant/negligible over the 20 repetitions of each experiment. The speedup achieved using Alg. 1 over ray-casting is illustrated to the right.

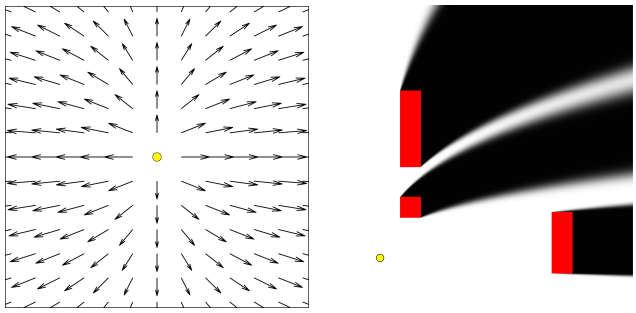

Fig. 3. Quiver plot on left showing the vector field, centered around the light source, that governs the resulting behaviour of the *curves-of-sight*. Resulting scalar field is $\frac{y}{2.5 \times x}$. The resulting *curvilinear polygon* is illustrated on the right with the source of light being the yellow dot.

Our approach can be extended to generic curves by editing the vector field that produces (5) while preserving (9). An example is illustrated in Fig. 3. As such, our approach then checks for the existence of *curves-of-sight* rather than lines-of-sight. This can be useful when dealing with a robot that moves in curves rather than straight lines. More complex forms of the underlying vector field can be adopted to possibly reflect robot kinematic constraints. The linear transport equation (2) can also be generalized to N dimensions [26].

Lastly, the approach can check the existence of lines-of-sight and *curves-of-sight* to multiple source points or even a surface S. By discretizing the latter, one can evaluate multiple polygons, one for each point, then perform

$$\mathcal{U}_{\cap} = \min_{p \in S} \mathcal{U}(p), \tag{11}$$

where $\mathcal{U}_{\cap}$ is the visibility or *curvilinear* polygon common to all points p in S, the set of queried points. $\mathcal{U}(p)$ is the polygon per queried point p.

IV. Path Planning Using Visibility

Visibility through lines-of-sight is the geodesic shortest path between two points and the algorithm introduced in Section III quantifies visibility for the whole grid. It is thus natural to apply Alg. 1 to a path planner.

A. The Visibility Heuristic

Let $\mathcal{S}_0$ be the set of points having visibility values ≥ 0.5 in the visibility polygon $\mathcal{U}(p_0)$ of a starting point p_0. As such, $\mathcal{S}_0$ represents the set of points visible to p_0. Let p_1 be a point in $\mathcal{S}_0$ having the minimum visibility value and evaluate the visibility polygon $\mathcal{U}(p_1)$. Performing the union of $\mathcal{U}(p_0)$ and $\mathcal{U}(p_1)$ results in the cumulative visibility polygon $\mathcal{U}_{\cup}$ that captures the collective visibility of points p_0 and p_1. Repeating the same procedure iteratively allows us to build

$$\mathcal{U}_{\cup} = \max_{w \in \mathcal{W}} \mathcal{U}(w), \tag{12}$$

where $\mathcal{W}$ is the set of waypoints that have been placed so far and $\mathcal{U}_{\cup}$ is the cumulative visibility seen by waypoints in $\mathcal{W}$. Using (12) allows us to overcome local minima by choosing the next waypoint to be the point in the visible set $\mathcal{S}$ of $\mathcal{U}_{\cup}$ that has minimum visibility value. As such, upcoming waypoints will not be placed in regions that have already been fully explored, but rather in regions that are *barely* visible to $\mathcal{W}$ (meaning points having visibility v such that threshold $= 0.5 \leq v \ll 1$). The algorithm will keep exploring until the entire map is visible to at least one waypoint. Therefore, the algorithm is guaranteed to establish visibility with all points in the grid that are reachable by straight lines, eventually leading to establishing visibility with the target. Assuming that the planner's target point p_t does not trivially belong to $\mathcal{S}_0$, a terminating criterion is $\mathcal{U}_{\cup_{p_t}} \geq 0.5$. This notation indicates that the target point p_t has a collective visibility value of at least 0.5 making it visible to at least one waypoint in $\mathcal{W}$. An upper bound on the number of iterations is also set as a terminating criterion for the case where the grid has inaccessible regions.

We add to the scheme a distance bias towards the target in order to pick the next waypoint in a manner that minimizes the path length covered by $\mathcal{W}$ so far, similar to an A^* path planner. By saving each waypoint's parent, we also keep track of the path as it is being constructed. The next chosen waypoint w_{i+1} need not have the predecessor waypoint w_i as its parent due to the fact that waypoints are chosen based on $\mathcal{U}_{\cup}$ rather than $\mathcal{U}(p_i)$. Every newly explored point seen by w_i but not previously seen by any predecessor gets w_i as its parent. This is done in step 5 in Alg. 2. Step 7 in Alg. 2 entails picking the point which minimizes the heuristic $\mathcal{H}$

$$\mathcal{H}_p = d_{\text{total}_\text{p}} + \bar{\mathcal{U}}_{\cup_p}, \forall p : \mathcal{U}_{\cup_p} \geq 0.5, \tag{13}$$

where $d_{\text{total}} = d_{\text{parent}} + d_{\text{target}}$, with d_{parent} being the distance between a point p and its parent and d_{target} the distance between p and p_t. $\bar{\mathcal{U}}_{\cup_p}$ is the properly scaled visibility value at p. Our heuristic implementation (13) may not be the best, but the purpose of this paper is not to produce the best implementation, rather, it is to introduce the proposed visibility **quantity** as a heuristic for path planning applications.

B. Visibility Heuristic Path Planning Results

The progression of a solution utilizing Alg. 2 with a threshold of 0.5 is illustrated in Fig. 4 for a 1000×1000 grid. In step 1, the visibility polygon for p_0 is evaluated and the cyan waypoint is chosen based on (13). In steps 2-5, the process is repeated while avoiding local minima due to 12. The stopping criterion is reached by step 6. The resulting path is a point-to-point path. A much more complex

Algorithm 2 Visibility Heuristic Path Planner

Inputs: $\mathcal{O} \leftarrow$ Probabilistic occupancy grid complement
$p_0 \leftarrow$ Start position, $p_t \leftarrow$ Target position
$(n_x, n_y) \leftarrow$ Grid dimensions, $\max_i \leftarrow$ max iterations
Output: $\mathcal{E} \leftarrow$ Map containing the parents of explored points

procedure VISIBILITYPATHPLANNER(Inputs)
  Initialize waypoint $w \leftarrow p_0$, $\mathcal{W}$ empty set, $i = 0$
  while $\mathcal{U}_{p_t} <$ threshold $\wedge\, i <$ max$_i$ **do**
    add w to $\mathcal{W}$
    compute $\mathcal{U}(w)$ and update $\mathcal{E}$
    $\mathcal{U}_{\cup} \leftarrow \max_{\forall w \in \mathcal{W}} \mathcal{U}(w)$
    $w \leftarrow \operatorname{argmin}_p \mathcal{H}$ (maintained by a heap)
    $i \leftarrow i + 1$
  end while
  return $\mathcal{E}$
end procedure

maze environment of size 322×322 can be explored and a solution can be found quickly with a threshold of 0.2 as illustrated in Fig. 5. The solution to the left side maze was obtained in 51 ms in C++ on an Intel® Core™ i7-9750H Processor. Based on our tests, such an environment is more challenging than a highly-cluttered randomly-generated environment containing curved obstacles.

Alg. 2 scales *linearly* with the number of exploration iterations, which itself depends on the heuristic implementation and tuning choice (13). In our implementation, and at every iteration, we are evaluating visibility for the entire grid, even for regions far beyond the visible ones (e.g., dark regions in Fig. 5). One improvement could be having an inner-loop exist strategy where the algorithm stops evaluating visibility once it goes deep beyond visible regions. The proposed planner is distance sub-optimal when compared to other state-of-the-art any-angle path planners such as Anya [30], [31].

V. Conclusion & Future Work

In this paper, we introduced a fast and efficient method to evaluate 2D visibility for grid maps using the linear transport equation — a linear first-order hyperbolic PDE. The proposed method computes both the visibility polygon by evaluating lines-of-sight and the *curvilinear polygon* by evaluating *curves-of-sight*. We demonstrated the efficiency of such an algorithm and we demonstrated its efficacy by introducing an interesting application that uses visibility as a heuristic for path planning. We produced simulation results using the path planner. We also provided pseudo-codes and sample implementations as an open-source code.

As briefly discussed in Section IV-B, our heuristic implementation choice has room for improvement. The number of computations when evaluating visibility at every iteration may be vastly reduced by having stopping criterion reliant on the amount of visibility information being added (using local or on-demand visibility evaluations). A superior path planning algorithm relying on the concept of visibility that we introduced can also be produced. The potential of customizing the shape of the desired curves when using Alg. 1 will be explored. Alg. 1 can have applications in localization by performing a maximum likelihood estimation based on comparing sensor data, e.g., LIDAR, to computed visibility polygons. Applications in area coverage, sensor placement, and pursuit evasion constitute attractive potential future works relying on Alg. 1 or a 3D version of it.

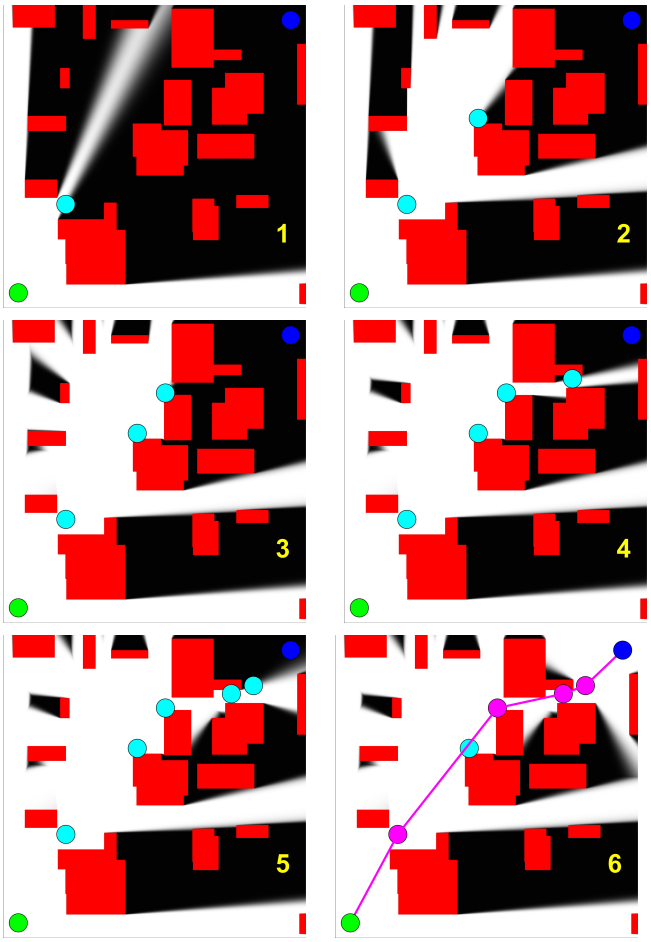

Fig. 4. Progression of the path planner in six steps. Starting point is in green at the bottom left, whereas the target point is the blue one at the top right. Points in cyan are intermediate waypoints, whereas the points and lines in magenta constitute the final path. Obstacles are shown in red.

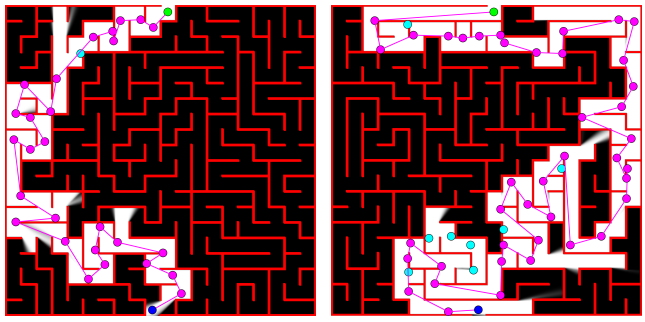

Fig. 5. Visibility heuristic planner solution in mazes. Start point at top center in green, target point at bottom center in blue, intermediate exploration waypoints in cyan, and final path in magenta. Obstacles in red.

REFERENCES

[1] M. Bern and P. Plassmann, "Chapter 6 - mesh generation," in *Handbook of Computational Geometry*, J.-R. Sack and J. Urrutia, Eds. Amsterdam: North-Holland, 2000, pp. 291–332. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444825377500073

[2] L. S. Davis and M. Benedikt, "Computational models of space: Isovists and isovist fields," *Computer Graphics and Image Processing*, vol. 11, pp. 49–72, 1979.

[3] B. Joe and R. B. Simpson, "Corrections to lee's visibility polygon algorithm," *BIT Numerical Mathematics*, vol. 27, pp. 458–473, 1987.

[4] T. Asano, T. Asano, L. J. Guibas, J. Hershberger, and H. Imai, "Visibility of disjoint polygons," *Algorithmica*, vol. 1, pp. 49–63, 2005.

[5] P. J. Heffernan and J. S. B. Mitchell, "An optimal algorithm for computing visibility in the plane," *SIAM J. Comput.*, vol. 24, no. 1, p. 184–201, feb 1995. [Online]. Available: https://doi.org/10.1137/S0097539791221505

[6] F. Bungiu, M. Hemmer, J. Hershberger, K. Huang, and A. Kröller, "Efficient computation of visibility polygons," *ArXiv*, vol. abs/1403.3905, 2014.

[7] L. Barba, M. Korman, S. Langerman, and R. I. Silveira, "Computing a visibility polygon using few variables," *Comput. Geom. Theory Appl.*, vol. 47, no. 9, p. 918–926, oct 2014.

[8] V. Chvátal, "A combinatorial theorem in plane geometry," *Journal of Combinatorial Theory, Series B*, vol. 18, no. 1, pp. 39–41, 1975. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0095895675900611

[9] M. E. Newell, R. G. Newell, and T. L. Sancha, "A solution to the hidden surface problem," in *Proceedings of the ACM Annual Conference - Volume 1*, ser. ACM '72. New York, NY, USA: Association for Computing Machinery, 1972, p. 443–450. [Online]. Available: https://doi.org/10.1145/800193.569954

[10] P. Bose, A. Lubiw, and J. Munro, "Efficient visibility queries in simple polygons," *Computational Geometry*, vol. 23, no. 3, pp. 313–335, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925772101000700

[11] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, no. 10, p. 560–570, oct 1979. [Online]. Available: https://doi.org/10.1145/359156.359164

[12] M. Pocchiola and G. Vegter, "Topologically sweeping visibility complexes via pseudotriangulations," *Discrete & Computational Geometry*, vol. 16, pp. 419–453, 1996.

[13] S. Kapoor and S. N. Maheshwari, "Efficiently constructing the visibility graph of a simple polygon with obstacles," *SIAM Journal on Computing*, vol. 30, no. 3, pp. 847–871, 2000. [Online]. Available: https://doi.org/10.1137/S0097539795253591

[14] M. Kallmann, "Dynamic and robust local clearance triangulations," *ACM Trans. Graph.*, vol. 33, no. 5, sep 2014. [Online]. Available: https://doi.org/10.1145/2580947

[15] R. Zeng, Y. Wen, W. Zhao, and Y.-J. Liu, "View planning in robot active vision: A survey of systems, algorithms, and applications," *Computational Visual Media*, vol. 6, no. 3, pp. 225–245, Sep 2020. [Online]. Available: https://doi.org/10.1007/s41095-020-0179-3

[16] K. Wu, R. Ranasinghe, and G. Dissanayake, "Active recognition and pose estimation of household objects in clutter," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4230–4237.

[17] C. McGreavy, L. Kunze, and N. Hawes, "Next best view planning for object recognition in mobile robotics," in *Proceedings of the 34th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG2016)*, Nov. 2016, 34th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG2016) ; Conference date: 15-12-2016 Through 16-12-2016.

[18] J. Santos, M. Oliveira, R. Arrais, and G. Veiga, "Autonomous scene exploration for robotics: A conditional random view-sampling and evaluation using a voxel-sorting mechanism for efficient ray casting," *Sensors*, vol. 20, no. 15, 2020.

[19] B. Baker, I. Kanitscheider, T. M. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, "Emergent tool use from multi-agent autocurricula," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[20] A. Tandon and K. Karlapalem, "Medusa: Towards simulating a multi-agent hide-and-seek game," in *International Joint Conference on Artificial Intelligence*, 2018.

[21] J. Amanatides and A. Woo, "A fast voxel traversal algorithm for ray tracing," *Proceedings of EuroGraphics*, vol. 87, 08 1987.

[22] I. Ibrahim, F. Farshidian, J. Preisig, P. Franklin, P. Rocco, and M. Hutter, "Whole-body mpc and dynamic occlusion avoidance: A maximum likelihood visibility approach," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 221–227.

[23] T. Nägeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, "Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1696–1703, 2017.

[24] G. Allaire, M. Bihr, B. Bogosel, and M. Godoy, "Accessibility constraints in structural optimization via distance functions," Nov. 2022, working paper or preprint. [Online]. Available: https://hal.science/hal-03864841

[25] R. Farias and M. Kallmann, "Optimal path maps on the gpu," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 9, pp. 2863–2874, 2020.

[26] L. C. Evans, *Partial Differential Equations*. American Mathematical Society, 2010.

[27] S. V. Patankar, *Numerical heat transfer and fluid flow*, ser. Series on Computational Methods in Mechanics and Thermal Science. Hemisphere Publishing Corporation (CRC Press, Taylor & Francis Group), 1980. [Online]. Available: http://www.crcpress.com/product/isbn/9780891165224

[28] H. Lewy, K. Friedrichs, and R. Courant, "Über die partiellen differenzengleichungen der mathematischen physik," *Mathematische Annalen*, vol. 100, pp. 32–74, 1928. [Online]. Available: http://eudml.org/doc/159283

[29] C. Hirsch, *Numerical computation of internal and external flows: The fundamentals of computational fluid dynamics*. Elsevier, 2007.

[30] D. Harabor and A. Grastien, "An optimal any-angle pathfinding algorithm," in *ICAPS 2013 - Proceedings of the 23rd International Conference on Automated Planning and Scheduling*, 06 2013.

[31] T. Uras and S. Koenig, "An empirical comparison of any-angle path-planning algorithms," in *Symposium on Combinatorial Search*, 2015.