Journal of Computational Physics 547 (2026) 114538

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A robust method for fast exploration of environments with moving obstacles

Gerardo E. Oleaga, Daniel Ortega-Lozano, Valeri A. Makarov *

Department of Applied Mathematics and Mathematical Analysis, and Instituto de Matematica Interdisciplinar, Universidad Complutense de Madrid, Madrid, 28040, Spain

ARTICLE INFO

Keywords:
Fast marching method
Wavefront
Mobile obstacle
Pathplanning

ABSTRACT

Exploring environments with static and moving obstacles is a fundamental problem with numerous applications in physics and engineering. The Fast Marching Method (FMM) offers a computationally efficient numerical solution to the Eikonal equation, which describes a wavefront propagating through a medium. The FMM is effective in media with static obstacles, but, as we show, it fails in the presence of moving ones. We introduce a novel, robust method for wave exploration of environments of arbitrary dimension and complexity, and prove its convergence numerically. The method accurately handles both dynamic and static obstacles while preserving the computational efficiency of the FMM, ensuring a fast and reliable global search for collision-free trajectories. The algorithm can also serve as an interception strategy for catching a moving target among many obstacles.

1. Introduction

Accurate and efficient simulation of waves propagating in complex time-evolving environments is a long-standing problem with multiple applications in science and engineering [1]. The interdisciplinary nature of this field is particularly intriguing, as waves can also perform computational tasks [2]. For example, in robotics, they can solve the shortest path problem: find an optimal path leading a mobile agent to a target region in a cluttered environment [3]. Even with a single destination point, it is an NP-hard problem [4], while a wave process can solve it for all destinations simultaneously. The wavefront, starting from a departure point, explores the environment, finding possible ways to reach all accessible locations while avoiding obstacles on the path [5,6]. This problem also directly relates to biological cognitive maps, i.e., mental representations of the environment [7,8].

The original path planning problem assumes navigation in static scenarios [9]. Starting from the pioneering Dijkstra's and A-star algorithms on graphs [10,11], numerous refinements have been proposed for exploring static environments (for recent reviews, see e.g. [12,13]). On the other hand, a rapidly expanding sector of autonomous ground, aerial, and marine vehicles, along with traffic control and virtual reality animation, requires navigation in dynamic environments (see, e.g. [14–16]). Then, neglecting moving obstacles severely impairs the safety, effectiveness, and efficiency of motion planning [17]. However, searching for global spatiotemporal solutions in time-evolving media remains a challenging task.

The complexity stems from the nontrivial influence of the time-dependent configuration of obstacles and the environment exploration. Even in apparently simple scenarios, it leads to a nontrivial connectivity of the model space and time-evolving regions of

* Corresponding author.
E-mail address: vmakarov@ucm.es (V.A. Makarov).
URL: http://blogs.mat.ucm.es/vmakarov (V.A. Makarov)

https://doi.org/10.1016/j.jcp.2025.114538
Received 30 April 2025; Received in revised form 17 November 2025; Accepted 18 November 2025
Available online 22 November 2025

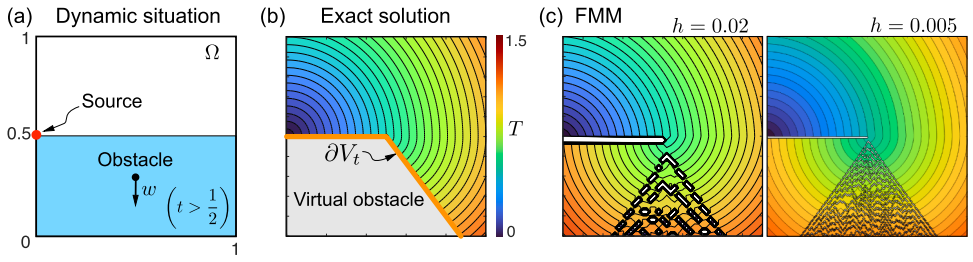

Fig. 1. The FMM fails in dynamic situations. (a) The simplest dynamic situation. A rectangular obstacle (blue) covers half of the domain $\Omega = (0,1)^2$, and it starts moving down at $t = 0.5$ with a constant velocity $w = 0.8$. The wave source is at $(0, 0.5)$. (b) Exact solution. The color corresponds to the arrival time $T(x, y)$. A virtual obstacle in the form of a trapezium (gray) is formed. The orange line shows the obstacle boundary. (c) Examples of solutions provided by the FMM with different mesh sizes h. The wave leaks through the virtual obstacle into the prohibited zone for any h (a supplemental video is available at the link). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

reachability in dynamic environments. Minor changes in the configuration of obstacles can destabilize laminar flows of trajectories and render them unstable and eventually unpredictable in terms of dynamical systems. The problem violates the First-In-First-Out principle [18] and requires introducing time-expanded configurational spaces [19]. The latter can be highly computationally expensive in multidimensional spaces, for example, when working with manipulators that have many degrees of freedom [20]. Usually, the complexity surpasses the algorithm's capacity in a few iterations. Then, many approaches resort to solving the problem locally, i.e., a solution, valid for an immediate vicinity, is replanned periodically along with the agent's movement [21–24]. Searching for global spatiotemporal solutions in predictable environments is a fundamentally different approach. Partially, it can be achieved using lattices of coupled differential equations [25], which is extremely computationally demanding.

Another approach to the static case is rooted in the level-set method [26]. If the wavefront speed does not change sign, in a static environment, the problem can be reduced to the Eikonal equation [27]: $v(x)|\nabla T(x)| = 1$, where $x \in \mathbb{R}^n$ is the space variable, $v(x) \geq 0$ is the agent's velocity, and $T(x)$ is the minimum time required to travel to location x. Thus, we get a problem of motions in a potential field. Note that it naturally assumes that an agent can pass through a point x only once. Although such a setup can admit an infinite number of solutions, in some cases, it is possible to select the relevant one by applying vanishing viscosity methods [28], which, however, introduces unnecessary complexity and requires heavy computations.

Tsitsiklis and Sethian proposed a remarkable algorithm to solve the Eikonal equation [29,30]. The so-called Fast Marching Method (FMM) allows for the numerical determination of a scalar field $T(x)$ by propagating a wave starting from an initial set. Despite its limitation in the class of available solutions, the potential field concept has proven to be highly practical in various applications [31,32]. The FMM's efficiency arises from its ability to evaluate $T(x)$ only in a neighborhood of the wavefront, i.e., at the nodes close to nodes assessed in the previous step, which significantly reduces the computational burden and even has room for further refinements [33,34]. The method complexity is of order $O(N \log N)$, where N is the number of nodes. The Fast Sweeping Method (FSM) enhances computational efficiency to $O(N)$ by employing a distinct strategy to maintain the causality of information transfer between nodes [35,36].

In dynamic situations, the problem becomes much more involved. The solution must fulfill the following equation:

$$|\nabla T(x)| = \frac{1}{v(x, T(x))}. \tag{1}$$

Thus, the agent's velocity $v(x, t)$ depends not only on the position but also on the time instant when the agent arrives at the point. Moreover, at collisions with obstacles, we must formally set $v(x, T(x)) = 0$, which is equivalent to having $T(x) = \infty$. In other words, the agent cannot penetrate an obstacle. We notice that this condition renders numerical methods designed for solving a Hamilton-Jacobi-Bellman equation ineffective (for more details, see, e.g., [37]). Nevertheless, it is tempting to generalize the FMM by setting $T(x) = \infty$ for locations where the wave collides with obstacles. We then checked this hypothesis in a few simple testbeds and found that it failed.

To illustrate the issue, consider a deliberately simplified toy problem (Fig. 1(a)). The departure point of the agent moving at unit speed is on the border of the square domain (red dot). At $t = 0$, a rectangular obstacle occupies the bottom half of the domain, and at $t = 1/2$ it starts moving down with a constant speed $w < 1$. Due to the extreme simplicity, this dynamic situation can be resolved analytically (for details, see Appendix A). Fig. 1(b) shows the exact, unique solution. It has a trapezoidal zone (colored in gray) that is not accessible to the wave. However, the FMM provides solutions that leak into the obstacle (Fig. 1(c)). False gaps appear at the border, and the wave leaks into the prohibited area. Moreover, the gaps do not disappear even if the discretization step $h \to 0$. Thus, a naive application of the FMM to time-evolving situations is deficient and requires further studies.

Using different situations, we observed that other methods (e.g., the FSM) exhibit similar problems. Furthermore, the analysis reveals an unreasonable variety of qualitatively different solutions for simple situations (see, e.g., Fig. 3 below), corroborating a significantly higher level of complexity of the dynamic model. Regarding the leakage problem (Fig. 1), the circularity of the wavefront on small spatial scales (of the grid size) is a paramount feature. A wave starting from a point source in a continuous homogeneous medium immediately forms a circular wavefront. However, the discretization of the medium introduces errors, which can be negligible

in static scenarios (tending to zero as the mesh size decreases) but can significantly distort the solution in dynamic situations. Distortions arise from unavoidable uncertainties when calculating the collision time at discrete nodes. Thus, the discretization scheme determines the accuracy of numerical solutions, and it must satisfy the causality principle (for more details see, e.g., [6,35,36]). Then, noncausal solutions must be filtered out. Another difficulty is dealing with codimension-one *virtual* obstacles, which cannot be captured adequately by a discrete grid. Consequently, dynamic situations are inherently unstable and require a precise numerical approach to approximate continuous solutions.

The term "virtual obstacle" is used in the literature discussing algorithms for navigation in static scenarios, based on building artificial potential fields [38]. Such algorithms often generate local minima, which can trap the agent (see, e.g., [39] and references therein). Then, heuristically adding fake obstacles (also called virtual) can help avoid local minima. We, however, use the term as introduced by Villacorta-Atienza et al. [6], which refers to areas inaccessible to an agent in the future due to the evolving nature of a dynamic situation with real moving obstacles.

In this work, we present a robust method that effectively handles the dynamic case, thus addressing the challenge and bridging the gap in simulating wavefronts in static and dynamic environments. Notably, the algorithm works for state spaces of any dimension (including 3-D and higher) and inherits the computational efficiency of the FMM. In 2-D situations, it provides global solutions almost immediately, even on a modest PC, which makes it attractive for real-time implementation in mobile robots and other AI devices.

2. The problem

Let us consider the exploration of an environment, i.e., an open region $\Omega \subset \mathbb{R}^d$, $d \in \mathbb{N}$, by a wavefront departing from $x_0 \in \Omega$ and moving with unit speed. It is equivalent to searching for trajectories by a point agent. We note that the assumption of a point agent is not restrictive. A shrinking transformation can reduce any problem involving an agent occupying a certain area or volume, or even a multi-joint manipulator, to a problem with a point agent [40,41]. The unit-speed constraint corresponds to a class of trajectories moving at optimal speed, which minimizes energy expenditure in biological and artificial agents (see, e.g., [42,43], and references therein). This energy criterion may be required by specific applications such as outer space, underwater exploration, or military tasks [44]. A broader family of solutions can also be derived using specific configuration spaces of higher dimension.

The environment Ω can have an arbitrary number of static or dynamic objects moving along given trajectories, which the agent must avoid along its path. At the end of its trajectory, the agent can target a single or several objects. For convenience, we consider all objects as obstacles. To model them, we introduce a continuous scalar function $g : \overline{\Omega} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$, which can be related to the distance to obstacles and time-evolving areas with $g(x,t) \leq 0$ delimiting the obstacles. Then, the sets:

$$M_t = \left\{ x \in \overline{\Omega} : \ g(x,t) < 0 \right\}, \partial M_t = \left\{ x \in \overline{\Omega} : \ g(x,t) = 0 \right\}, \tag{2}$$

are the interior and the boundary of the obstacles at time t, respectively. If the environment has static obstacles only, function g and sets (2) are time-independent. Note that the departure point must satisfy: $x_0 \notin \overline{M}_0$. We also assume that the obstacle's speed is finite.

A wave process starting from x_0 and propagating outwards explores $\overline{\Omega}$ and constructs a scalar potential field T defining the time of arrival from x_0 to x:

$$\begin{aligned} T : \overline{\Omega} &\to \mathbb{R}_{\geq 0} \cup \{\infty\} \\ x &\mapsto T(x). \end{aligned} \tag{3}$$

By definition, $T(x_0) = 0$ and $T(x) > 0 \ \forall x \in \overline{\Omega} \setminus \{x_0\}$. Notice that in the presence of obstacles, there can be areas unreachable to the wavefront, and hence $T(x) = \infty$ for such regions. Thus, T is an extended function, defined for the closed domain $\bar{\Omega}$ (i.e., the arena). Given T, we can define the set of *reachable* points as:

$$R := \{ x \in \overline{\Omega} : \ T(x) < \infty \}. \tag{4}$$

Furthermore, no reachable point can be inside an obstacle, but points that lie on the boundary of obstacles can be reachable. Thus, we impose the compatibility condition: $g(x, T(x)) \geq 0$, $\forall x \in R$.

To set a unique, physically coherent time field, we define a specific exploration process. For every point $x^* \in R$, there exists a continuous trajectory $\gamma : [0, T(x^*)] \to R$ connecting x_0 and x^* (i.e., $\gamma(0) = x_0$, $\gamma(T(x^*)) = x^*$) with unit speed $|\dot{\gamma}(t)| = 1$, satisfying:

$$g(\gamma(t), t) \geq 0, \quad t \in [0, T(x^*)], \tag{5}$$

and

$$\frac{d\gamma(t)}{dt} = \nabla T(\gamma(t)) \quad \text{if} \ \ g(\gamma(t), t) > 0. \tag{6}$$

Eqs. (5)–(6) constitute a complementarity problem typical of free boundary problems [45]. Notice that the exploration is performed in one of two regimes: a) For $g(\gamma(t), t) = 0$, the gradient of T is not well defined, and the trajectory γ is sliding along the obstacle boundary at unit speed, eventually leaving it. b) For $g(\gamma(t), t) > 0$, the explorer is far from the obstacles, and it moves along the gradient flow of T [Eq. (6)].

Now, we formulate the problem.

Definition 1 (The problem)**.** Given an environment $\Omega \subset \mathbb{R}^d$ ($d \in \mathbb{N}$), an initial point $x_0 \in \Omega$, and a continuous function $g(x,t)$ defining the obstacles' movement [Eq. (2), such that $g(x_0, 0) > 0$], find a scalar time field T [Eq. (3)] such that $T(x_0) = 0$ and for any point

$x^* \in R$ [Eq. (4)] there exists a continuous trajectory $\gamma(t)$ satisfying (5)–(6) (i.e., the agent gets to x^* avoiding collisions with the obstacles along the trajectory). Moreover, the field T must be maximal, i.e., $\forall T'$ s.t. $T(x) = T'(x)\ \forall x \in R$, we have $R' = R$ (i.e., no other field defines a larger reachable set than T under the imposed conditions).

Remark 1. ∇T exists almost everywhere in R. Then, the Eikonal equation holds:

$$|\nabla T(x)| = 1, \quad \forall\, x \in R : g(x, T(x)) > 0 \ \text{ a.e.} \tag{7}$$

Remark 2. By setting the endpoint x^* on the border of a virtual obstacle, we can determine a trajectory that approaches a specific moving object and avoids collisions with others. It can be considered an interception strategy in the presence of dynamic obstacles.

Using the generalized Eikonal Eq. (1), we can formally define the wave velocity:

$$v(x, T(x)) = H(g(x, T(x))), \tag{8}$$

where $H(\cdot)$ is the Heaviside function. The described problem is significantly more complex than the problem of exploring static environments, which is the origin of the failure of straightforward methods, such as a naive extension of the FMM into dynamic situations (Fig. 1). We also note that the provided problem formulation is related to the methods of artificial potential fields, widely discussed in the literature (see, e.g., [39,46] and references therein). The agent also follows the gradient of a potential field; however, at certain parts along the trajectory, the gradient may not be well-defined. Furthermore, there are two key differences: our approach applies to dynamic situations, and it has no issue of local minima.

3. Physical analogy for building the scalar time field

The exploration process is inspired by particle swarm optimization [47]. First, we state a basic property for trajectories outside the obstacles.

Proposition 1. *Let T be a generalized scalar field and $\gamma(t)$ be an exploration trajectory, such that $g(\gamma(t), t) > 0$ for $t \in [t_1, t_2]$, satisfying Definition 1. Then, $\gamma(t)$ is a straight path from $\gamma(t_1)$ to $\gamma(t_2)$.*

Proof. Taking into account that $|\dot\gamma(t)| = 1$, the parameter t is the arc-length of the curve γ. As $g(\gamma(t), t) > 0$, the curvature can be computed from Eq. (6):

$$\left|\frac{d^2\gamma(t)}{dt^2}\right| = \left|\frac{d\nabla T(\gamma(t))}{dt}\right| = \left|\mathrm{Hess}T(\gamma(t)) \cdot \frac{d\gamma(t)}{dt}\right| = |\mathrm{Hess}T(\gamma(t)) \cdot \nabla T(\gamma(t))|, \tag{9}$$

where the dot denotes matrix-vector product. Now notice that $\mathrm{Hess}T \cdot \nabla T = \frac{1}{2}\nabla|\nabla T|^2$ and recalling that $|\nabla T| = 1$, we conclude that $\left|\frac{d\dot\gamma(t)}{dt}\right| = 0$ and hence the path is a straight line for $t \in [t_1, t_2]$. □

For visual clarity, we illustrate the method in 2-D environments. Section 4 provides numerical algorithms for arbitrary dimensions.

3.1. Static environments

Although the study of static situations is widespread, we include it for completeness.

3.1.1. Solution without obstacles

Consider the following thought process. At $t_0 = 0$, x_0 emits particles traveling at unit speed in every direction. The particles do not interact with each other or with the medium. Then, the positions of the particles (see Proposition 1) are:

$$p(t; x_0, u) = x_0 + ut, \quad |u| = 1. \tag{10}$$

At time $t > 0$, the particles will reach a circumference of radius t, centered at x_0. Therefore, in a convex domain, any point $x \in \overline{\Omega}$ will be visited by a particle at time $t = |x - x_0|$. Thus, we get the time field (Fig. 2(a)):

$$T(x) = |x - x_0|, \tag{11}$$

which satisfies the Eikonal Eq. (7) for all $x \neq x_0$.

The level set $T(x) = t$ defines the *traveling wavefront*:

$$F_t := \left\{ x \in \overline{\Omega} :\ T(x) = t \right\}. \tag{12}$$

The front can be related to the "present" of the process of exploration of Ω. Then, the points located "behind" the front can be identified with the "past":

$$A_t := \left\{ x \in \overline{\Omega} :\ T(x) < t \right\}, \tag{13}$$

while the points not yet visited, $T(x) > t$, correspond to the "future". The present and past of the exploration process (12), (13) satisfy the chronological or causal relationship $A_t = \cup_{s<t} F_s$. Such a definition is typical in the mathematical approach to cognitive maps [6].

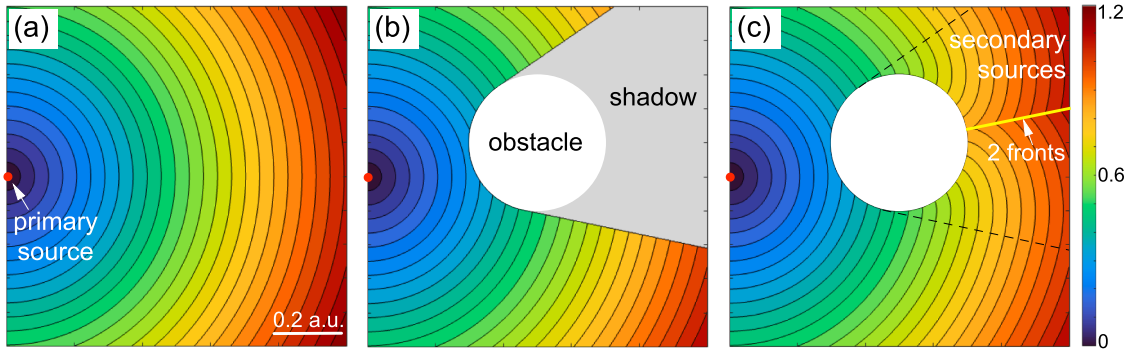

Fig. 2. Time fields in a square domain created by particles emitted by a source at $x_0 = (0, 0)$ (red circle). (a) Obstacle-free situation: $T = |x - x_0|$. The color corresponds to the arrival time. (b) The situation with a circular obstacle of radius 0.2 with the center at $(0.5, 0.1)$. The points in white have infinite arrival time, i.e., they are unreachable for particles. The gray area is also unreachable within the unique source concept. (c) The same as in (b) but under the assumption of secondary sources. The yellow line corresponds to particles reaching the same location from different sides. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.1.2. Solution with a static obstacle

Fig. 2(b) shows a situation with a static obstacle, a disk of radius $r = 0.2$. It acts as a barrier for particles, truncating their trajectories. In accordance with the problem (Section 2), we postulate that a particle "annihilates" when it collides with an obstacle. Therefore, at most, particles can reach the obstacle's boundary but not its interior. Thus, the arrival time for such inaccessible points is infinite: $T(x) = \infty$, $\forall x \in M$. Due to the rectilinear particle movements, obstacles also produce "shadows", i.e., other unreachable locations (Fig. 2(b), gray area). However, the agent could eventually reach such locations. Inspired by Huygens' principle, we extend the thought process to allow for the exploration of shadow zones.

Definition 2 (Secondary sources)**.** Suppose a particle visits position x for the first time $T(x)$. In that case, it "activates" it and becomes a secondary source that emits new particles in all directions at the time of activation. However, in a collision with obstacles, particles annihilate and hence do not activate the corresponding locations.

This definition can be related to the notion of secondary sources used in robotics as a means for extending sensory information beyond direct measurements to help an agent understand an environment (see, e.g., [48]). Here, we also use them to reach points in the shadow from other activated points by rectilinear paths (Fig. 2(c)). However, the utility of secondary sources in our case goes much further; they play a crucial role in dynamic situations (see below).

We define the arrival time introduced in Eq. (3):

$$T(x) = \inf\{t : \text{ a particle arrives at } x \text{ before time } t\}. \tag{14}$$

Here, the infimum accounts for situations where, strictly speaking, the minimum time does not exist (e.g., it may occur on the obstacle's boundary). In terms of (14), all points $x \in M$ in the interior of the obstacle are unreachable for the particles, $T(x) = \infty$. Meanwhile, points $x \in \partial M$ in the obstacles' boundary have a finite time of arrival $T(x) < \infty$. Note that ∇T is discontinuous at points reached simultaneously from different activated locations (Fig. 2(c), 2 fronts).

3.2. Dynamic situations

Moving obstacles challenge the exploration problem. Straightforward application of static algorithms does not work (Fig. 1). We now describe the approach that will later enable a numerical calculation of the time field $T(x)$ for all scenarios.

3.2.1. Virtual obstacles determine the solution

A moving obstacle does not tag locations as unreachable unless it interacts with the particles' wavefront. As above, a unique source at $x_0 \in \overline{\Omega}$ emits particles at $t = 0$ that move along rectilinear paths with unit speed (10). When a particle reaches a location $x \in \Omega$, it can activate it if another particle hasn't done it before (Eq. (14)). An activated location emits particles in every direction.

Consider a particle arriving at x for the first time $T(x)$. If at $T(x)$ the obstacle's border occupies x (i.e., $x \in \partial M_{T(x)}$), then the particle annihilates, and x becomes a *permanent virtual obstacle*. No other particle can pass through x after $T(x)$. We remind that here we refer to virtual obstacles in the sense of sets that the agent cannot cross [6].

Definition 3 (Virtual obstacles)**.** i) The virtual obstacle's boundary is made of all non-activated sites where collisions between particles and obstacles occur:

$$\partial V_t := \{x : T(x) < t \ \& \ g(x, T(x)) = 0\}. \tag{15}$$

The final virtual obstacle's boundary ∂V_∞ delimits unreachable areas. ii) The virtual obstacle is

$$V = \{x : T(x) = \infty\} \cup \partial V_\infty. \tag{16}$$

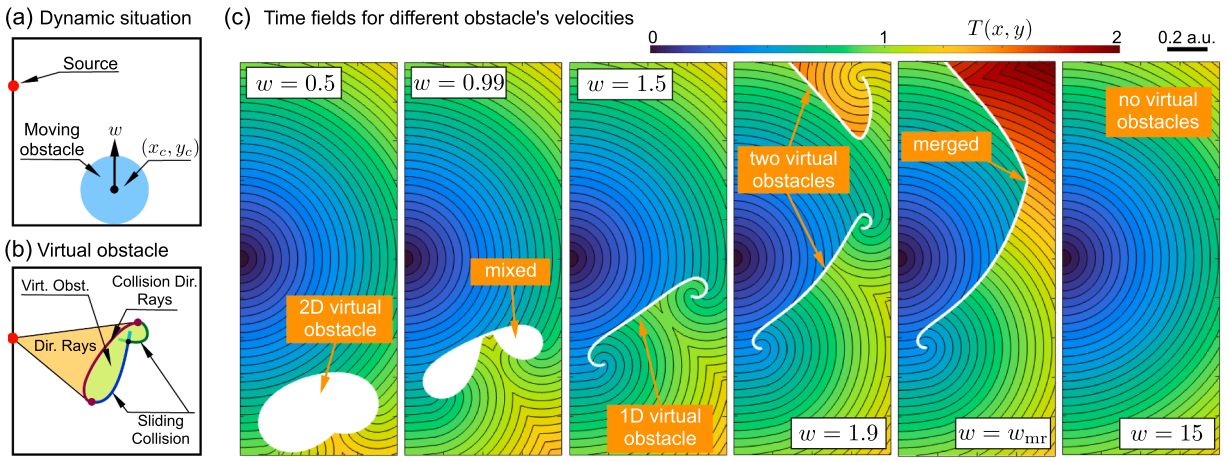

Fig. 3. Exact solutions for a dynamic situation with a moving disk. (a) The dynamic situation. A disk (obstacle) of radius $r = 0.3$ starts moving from $(x_c, y_c) = (0.4, -1.25)$ with constant speed $w > 0$ along the y-axis. The agent (source at $(0, 0)$) explores the environment and builds $T(x, y)$. (b) The boundary of the virtual obstacle typically consists of three parts: collisions with direct rays (brown curve) and two branches for sliding collision (green and blue curves). They appear as solutions of nonlinear algebraic and differential equations, respectively (see Appendix B). (c) Qualitatively different time fields $T(x, y)$, obtained for different speeds w. The critical value when two virtual obstacles merge is $w_{\mathrm{mr}} \approx 1.944$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.2.2. *Surprising variety of virtual obstacles in a simple situation*

As we will see, virtual obstacles are essential for defining admissible paths among moving real obstacles. However, their identification is a challenging task. To illustrate this, let's consider a seemingly simple situation. A circular obstacle starts moving from (x_c, y_c) "upwards" with constant velocity $w > 0$ (Fig. 3(a)). Then, depending on the parameters, we get surprisingly many different configurations of virtual obstacles and time fields $T(x, y)$.

Appendix B gives a complete account of the possible situations and provides exact solutions for each case (Fig. 3). Notice that the term "exact" here doesn't refer to a closed formula. Remarkably, parts of the virtual boundary are solutions of nonlinear algebraic and differential equations. Fig. 3(b) illustrates the main parts of the boundary of a virtual obstacle. Furthermore, finding $T(x)$ involves solving an optimization problem. Therefore, by exact solution, we mean that conventional numerical methods can provide $T(x)$ with arbitrary precision.

If w is low enough, the virtual obstacle (depicted in white in Fig. 3(c), $w = 0.5$) has a shape similar to a deformed disk. Notice that the obstacle's motion reshapes the gradient discontinuity (compare to Fig. 2(c)). As the obstacle's speed increases, we observe a different pattern (Fig. 3(c), $w = 0.99$). The virtual obstacle separates into two distinct shapes connected by a 1-D manifold. Thus, arbitrarily closed points on different sides of the manifold have significantly different arrival times (T is discontinuous). This directly results from how the secondary source particles collide with the moving disk.

When the disk runs faster than the particles ($w > 1$), the virtual obstacle reduces to an open curve (Fig. 3(c), $w = 1.5$). Particles can reach every location in Ω. At the same time, a second virtual obstacle appears descending from infinity (not shown for $w = 1.5$). Thus, a single obstacle can create two virtual obstacles (Fig. 3(c), $w = 1.9$). They appear due to the front and rear collisions of the particle's wavefront with the disk. In the latter case, the disk hits the wavefront from behind.

For $w = w_{\mathrm{mr}} \approx 1.944$, two virtual obstacles merge into one (Appendix B). Then, the virtual wall becomes strongly elongated, precluding particles from moving straight (Fig. 3(c), $w = w_{\mathrm{mr}}$). Observe that it takes a very long time to reach the upper-right corner of the domain. When the obstacle is too fast ($w > w_{\mathrm{cr}}$), particles cannot collide with it, and we get a field $T(x, y)$ the same as if there was no obstacle at all (Fig. 3(c), $w = 15$).

Remark 3. For reachable but not colliding points, the scalar field T is continuous; more precisely:

$$T \in C\big(R \setminus \partial V_\infty\big). \tag{17}$$

Then, due to Remark 1, T is also differentiable almost everywhere in $R \setminus \partial V_\infty$. An example of the discontinuity of T at a 1-D virtual obstacle can be seen in Fig. 3(c), $w = 1.5$. The discontinuity of ∇T can appear "behind" virtual obstacles.

The considered simple example illustrates the complexity of exploring dynamic environments. Little changes in the initial position or the velocity vector of a single obstacle can produce significant changes in the time field. Several obstacles make the problem analytically intractable. Thus, even for relatively simple situations, it is impossible to predict the locations and shapes of virtual obstacles. Their role is crucial in shaping the time field and is the key to understanding a dynamic situation. Once the time field (e.g., Fig. 3) has been found, it can serve as a *static cognitive map* of the dynamic situation (for more details, see [6,25]). In particular, it enables the construction of admissible paths to a target, avoiding both static and moving objects (see below).

3.3. Uniqueness of solutions and boundary conditions

Although a solution to the problem (Definition 1) satisfies the Eikonal equation $|\nabla T(x)| = 1$ outside the virtual obstacle, the inverse statement is false. The Eikonal equation can provide multiple unfeasible solutions. For example, consider the situation shown in Fig. 2(c) with a static disk D. The field $T(x, y) = \sqrt{x^2 + (y - 0.5)^2}$ for $(x, y) \in \Omega \setminus \overline{D}$ satisfies the Eikonal, but it is not a solution of the problem. Besides, ∇T is not well defined for locations reached by particles from different sides (see, e.g., Fig. 2(c)).

Let us now establish conditions for the *virtual boundary* ∂V compatible with the exploration process described above. Consider the motion of a particle that starts from x_s and collides with an obstacle at position x^* and time $T(x^*)$. Then $x^* \in \partial M_{T(x^*)}$ and hence $g(x^*, T(x^*)) = 0$. The particle path, Eq. (10), is:

$$p(t; x_s, u) = x_s + (t - T(x_s))u, \tag{18}$$

where $u = (x^* - x_s)/|x^* - x_s|$. Notice that $T(p(t; x_s, u)) = t$. Taking the time derivative of this equation and using (18), we get:

$$\nabla T \cdot u = 1. \tag{19}$$

Thus, we get from (19) $u = \nabla T$.

When a particle collides with the obstacle, it comes from the exterior of $M_{T(x^*)}$, and therefore:

$$u \cdot \nabla g(x^*, T(x^*)) \le 0, \quad x^* \in \partial M_{T(x^*)}. \tag{20}$$

Note that the collision position x^* appears on both sides of (20). It corresponds to the self-consistent nature of the problem, i.e., the arrival time of the wave to the obstacle depends on its position, and the obstacle's position is defined by the arrival time. This is one of the main difficulties of the problem, inducing unusual demands for numerical treatment. Finally, we get the collision condition:

$$\nabla T(x^*) \cdot \nabla g(x^*, T(x^*)) \le 0. \tag{21}$$

The strict inequality in (21) means that a particle hits the obstacle at a non-zero angle. Such collisions occur on the "illuminated" side of an obstacle by direct rays from the primary particle source (Fig. 3(b)). The equality in (21) occurs for a path tangential to the obstacle boundary, which happens on the shadow side of obstacles. In this case, the trajectory satisfies Eq. (5). However, there is no straightforward way to find it (see Appendix B for a simple example).

4. The algorithm

The environment Ω is discretized by the standard square grid of size h. Without loss of generality, we can set $h = 1$ and get $\Omega = \{1, \dots, n\}^d$, where $n \in \mathbb{N}$ is the linear number of nodes, and $d \in \mathbb{N}$ is the ambient space dimension. Given the initial agent's position $p_0 \in \Omega$ and the obstacles' dynamics $g(p, t)$, our goal is to compute the scalar time field $T(p)$ for $p \in \Omega$ (see Definition 1).

4.1. The fast marching method for static scenarios

First, let us recall the FMM. Its main structure is that of an Ordered Upwind Method, which belongs to a class of Dijkstra-like schemes. At the beginning, set $T(p) = \infty$ for all nodes except p_0, where $T(p_0) = 0$. Define two lists of nodes:

- List 1: Nodes being processed;
- List 2: Nodes whose time values have been accepted and "fixed".

At the beginning, List 1 contains the node p_0, while List 2 is empty. Nodes in List 1 have "tentative" values of time. The following iterative algorithm updates the lists:

1. Take node p from List 1 that has the minimal $T(p)$ and remove it from the list.
2. Add p to List 2, fixing its time value.
3. Add the nearest neighbors q of p (i.e., nodes at a unit distance) that are not in List 2, to List 1 and compute a tentative time t^* for each of them (see below). If they already had time values attached to them (i.e., if they were already in List 1), update $T(q)$ with the minimum between the computed and attached times.
4. Repeat steps 1–3 until List 1 is empty.

The array $\{T(p)\}_{p\in\Omega}$ provided by the algorithm contains the time field, which is a solution to the problem.

This algorithm does not account for obstacles. An extension to the case of static obstacles is the following. At the beginning, attach an infinite time value to all nodes in the set occupied by obstacles and add them to List 2.

To compute the tentative time t^* at Step 3 within the Euclidean metric, the FMM searches for the earliest planar wave that could reach node q. It results in t^* satisfying:

$$\sum_{i=1}^{d} \max\left\{T(q - e_i) - t^*, t^* - T(q + e_i), 0\right\}^2 = 1, \tag{22}$$

where e_i are vectors from the standard orthonormal basis of $\mathbb{R}^d$. The value of t^* can be computed via the Eikonal update formula [30]. Notice that, by applying the maximum in Eq. (22), the causality principle is preserved, ensuring that the flow of information proceeds from the past to the future.

4.2. Dynamic fast marching method

Based on the particle process described in Section 3, we develop a numerical algorithm, called *Dynamic Fast Marching Method* (DFMM). An essential aspect of the DFMM is the interaction of particles with moving obstacles. Nodes on the boundary of the obstacles are reachable points at a given time. Such locations can have a well-defined finite arrival time $T(p)$. If a particle hits an obstacle at p, the node gets the collision time and eventually passes to List 2 (defined in Section 4.1) as any other node. Such nodes form the boundary of virtual obstacles ∂V_t. They participate in the computation of the array T (see below) but don't add new nodes to List 1 as in Step 3 of the FMM.

Algorithm 1: The dynamic fast marching method.

Input: p_0: position of the source, g: function of (p,t) defining obstacles
Output: T: array $\{T(p)\}_{p\in\Omega}$ with first arrival times

```
1  L1 ← List ;                                          /* L1 are the nodes being ''processed'' */
2  L2 ← List ;                                  /* L2 are the nodes with ''fixed'' time values */
3  T ← d-dimensional Array of numbers ;                          /* T(p): time value of node p */
4  T(p) ← ∞ for all nodes p ∈ Ω;
5  append p_0 to list L1;
6  T(p_0) ← 0 ;                                                  /* set time value for (p_0) */
7  while L1 is not empty do
8      p ← take p with minimum time value in L1;
9      delete node p from L1 and append it to list L2 ;                          /* p is fixed */
10     if node p is exterior to the obstacles at time T(p) then
11         for q ∈ {p nearest-neighbors} & q ∉ L2 do
12             t* ← compute_time(p, q, L2, T);
13             if q ∈ L1 then
14                 if t* < T(q) then
15                     T(q) ← t*
16                 end
17             else
18                 append q to L1 and set T(q) ← t*
19             end
20         end
21     end
22 end
```

Thus, we replace Step 2 of the FMM by:

> 2^* Add node p to List 2, fixing its time value. Check if the node is in an obstacle at time $T(p)$ just computed. If true, omit Step 3.

Omitting Step 3 has significant consequences. Nodes inside the virtual obstacles do not emit particles and never reach their closest neighbors, but if they are close to the boundary, they may be reached by some particle and have a finite time value attached. Points that lie more deeply in the interior of the virtual obstacle will never be reached by particles, maintaining infinite values. It is important to note that the modification in Step 2 is not the only difference from the FMM (see below).

Algorithm 1 outlines the core of the DFMM. On line 9, node p is fixed regardless of whether it belongs to the obstacle. Notice that, even if in line 10 only the nodes out of the obstacle are processed, *its nearest neighbors may lie inside the obstacle*. In this way, neighbors of processed nodes have finite times associated with them and eventually belong to List 2. Some of them may be inside, but very close to the obstacle boundary at the computed time. Those are precisely the nodes that form the boundary of the *virtual obstacle*. Since the core scheme is based on the FMM, the computational complexity of the DFMM is of order $O(N \log N)$, where $N = n^d$ is the number of nodes and the $\log N$ factor comes from the heapsort algorithm [35].

4.3. Computation of arrival time

A direct implementation of the Eikonal update formula yields an incorrect computation of the field due to insufficient accuracy in determining causality in the information transfer from one node to another over a limited number of characteristics (Fig. 1). The DFMM relies on a precise computation of the time when a particle reaches a node (Algorithm 1, line 12), which addresses problems inherent to the dynamic nature of the obstacles. Let us now describe the numerical scheme.

To compute the tentative time for a node q, we will now consider all nodes $\|q' - q\|_\infty = 1$ as its neighbors. That is, if q is in the center of a hypercube $\mathcal{H}$ of side 2, its neighbors are all $3^d - 1$ nodes in $\partial\mathcal{H}$. Given time values $t_{q'}$ at each q', we calculate t^* for q.

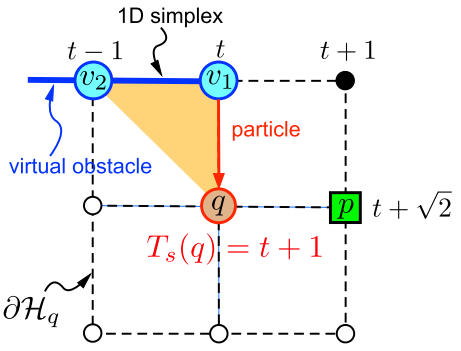

Fig. 4. Particle filtering helps conserve causality of propagation of information. Assume the virtual obstacle (in blue) includes nodes v_1 and v_2, which are in List 2 as well as the black node with time $t+1$ (nodes marked by open black circles have no time assigned). Node p (green square) receives its time from v_1. The addition of node p into List 2 with the fixed time $t+\sqrt{2}$ triggers the update of node q, which is included in List 1 when p goes to List 2. There are 8 simplexes on $\partial\mathcal{H}_q$; one of them (marked by the black arrow) with nodes v_1 and v_2 has the corresponding times t and $t-1$. Since particles cannot cross the virtual obstacle, there is only one option: a particle coming to q from the vertex v_1 (the other noncausal particles from this simplex are filtered out). Then, the corrected arrival time from this simplex is $T_s(q) = t+1$ (but not $t-1+\sqrt{2}$ coming from v_2 as it would be without filtering). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.3.1. *Time filtering*

First, to prevent noncausal interactions between nodes divided by codimension-one virtual obstacles, we will not consider times $t_{q'}$ more than $\sqrt{d}$ units into the past with respect to $T(p)$. This is equivalent to setting $t_{q'} = \infty$ if $t_{q'} < T(p) - \sqrt{d}$. See Algorithm 2. We refer to this step as "time filtering".

Algorithm 2: `filter` function.

Input: t: time of node q', a neighbor of q; T_p: time of the fixed node p
Output: filtered time t

```
1 if t < T_p − √d then
2 |   t ← ∞
3 end
4 return t
```

Let us illustrate this critical step in a two-dimensional example for visual clarity. Fig. 4 shows a typical situation with times set on previous steps of the algorithm. Node p with the fixed time $t+\sqrt{2}$ includes node q in List 1. Consider the simplex marked by nodes v_1 and v_2 with times t and $t-1$, respectively. In such a configuration, the minimal arrival time from the simplex to node q is $t-1+\sqrt{2}$. However, assume that a 1D virtual obstacle (as, e.g., in Fig. 3(c), $w = 1.5$) passes through this simplex. Since particles cannot cross the virtual obstacles, we must filter out all times except one, corresponding to a particle arriving from the vertex v_1. Other times are inconsistent with the particles' motion and the causality principle (for a formal definition see [36]). Thus, this simplex provides $T_s(q) = t+1$. We also note that filtering is effective only in the presence of obstacles in nearby nodes. It does not affect the wave propagation in open space.

Algorithm 3 shows the implementation of calculations of the arrival time (23), which resorts to a call of a `local_solver` (line 35). The local solver (Section 4.3.2) evaluates the time particles need to reach node q from a given simplex. As shown below, this procedure does not depend on a particular simplex and, hence, can be efficiently coded. The `filter` function (line 30) returns an infinite value if the node is not admitted for computation (Algorithm 2).

4.3.2. *Estimated time of arrival*

Now, we calculate the tentative arrival time t^*. To do this, we assume that each particle comes in a straight line from a point $x \in \partial\mathcal{H}$. Then, t^* is given by:

$$t^* = \min_{x\in\partial\mathcal{H}_q} \mathscr{L}T(x) + \|x-q\|_2, \tag{23}$$

where $\mathscr{L}T$ is a linear interpolation of the time value. The expression to be minimized in (23) is the so-called Sethian-Vladimirsky (S-V) metric [49]. Let us now describe an efficient procedure to solve (23).

To find the interpolation $\mathscr{L}T$, we first partition the hypercube $\partial\mathcal{H}$ into $d!2^d$ simplices, using the following one-to-one correspondence between simplices and the set $S_d \times \{-1,1\}^d$, where S_d is the dth symmetric group. For each $\sigma \in S_d$ and $s \in \{-1,1\}^d$, define

$$v_0 := q \tag{24}$$

Algorithm 3: Implementation of the function `compute_time` used by the DFMM.

Input: q: node where time is computed, added by neighbor p; $L2$: list of accepted nodes, T: time array
Output: estimated_time: first time of arrival of a particle to q

```
1  estimated_time ← ∞ ;
2  t ← Array of d time values ;
3  v_0 ← q ;                                              /* v_0 does not belong to the simplex */
4  for σ in S_d, s in {−1, 1}^d do
5      for i = 1 to d do
6          v_i ← v_{i−1} + s_i e_{σ(i)};
7          if v_i ∈ L2 then
8              t_i ← filter(T(v_i), T(p)) ;                          /* admissibility */
9          else
10             t_i ← ∞
11         end
12     end
13     estimated_time ← min{estimated_time, local_solver(t)}
14 end
15 return estimated_time
```

$$v_i := v_{i-1} + s_i e_{\sigma(i)}, i = 1, \dots, d, \tag{25}$$

and then define the simplex $\mathcal{S}_{\sigma,s}$ to be the polygon with vertices $\{v_1, \dots, v_d\}$. Note that

$$\partial\mathcal{H} = \bigsqcup_{\substack{\sigma \in S_d \\ s \in \{-1,1\}^d}} \mathcal{S}_{\sigma,s}, \tag{26}$$

is a partition. This allows us to compute the interpolation term $\mathscr{I}T(x)$ in (23) using barycentric coordinates in each simplex.

4.3.3. Numerical scheme at each simplex

Once a simplex $\mathcal{S}$ with vertices $\{v_1, \dots, v_d\}$ and vector of times $t = (t_1, \dots, t_d)^\intercal$ has been selected, we must evaluate the time of arrival of the first particle from the simplex to node q. This is done by a local solver (line 35, Algorithm 3), which finds the local minimizer in (23). In terms of barycentric coordinates, we get:

$$T(q; \mathcal{S}) = \min_{\lambda \in \mathbb{S}_d} \left\{ \lambda^\intercal t + \|q - V\lambda\|_2 \right\}, \tag{27}$$

where $\lambda = (\lambda_1, \dots, \lambda_d)^\intercal$, $\mathbb{S}_d = \{\lambda \in \mathbb{R}^d : \ \lambda_i \geq 0, \ \sum \lambda_i = 1\}$, and $V = (v_1, \dots, v_d)$.

We now introduce $M = (m_{ij})_{i,j=1}^d$, an upper triangular matrix with $m_{ij} = 1$ for $j \geq i$, and the symmetric positive definite matrix $Q = M^\intercal M$. Let $\mathrm{sp}(Q) = \{\mu_i\}$, $\mu_i > 0$ and $U = (u_1, \dots, u_d)$ be the spectrum and the matrix of orthonormal eigenvectors of Q. Then, the second term in (27) can be reduced:

$$\begin{aligned} &\left\| \sum_{i=1}^d (q - v_i)\lambda_i \right\|_2^2 = \left\| \sum_{i=1}^d \lambda_i \sum_{k=1}^i s_k e_k \right\|_2^2 = \left\| \sum_{k=1}^d s_k e_k \sum_{i=k}^d \lambda_i \right\|_2^2 = \\ &\sum_{k=1}^d \left(\sum_{i=k}^d \lambda_i \right)^2 = \lambda^\intercal Q \lambda = \tilde{\lambda}^\intercal D \tilde{\lambda} = \sum_{i=1}^d \mu_i \tilde{\lambda}_i^2, \end{aligned} \tag{28}$$

where $D = \mathrm{diag}(\mu_1, \dots, \mu_d)$ and $\tilde{\lambda} = U^\intercal \lambda$. Eqs. (27) and (28) yield:

$$\texttt{local\_solver}(t) = \min_{\lambda \in \mathbb{S}_d} \sum \tilde{\lambda}_i \tilde{t}_i + \left(\sum_{i=1}^d \mu_i \tilde{\lambda}_i^2 \right)^{\frac{1}{2}}, \tag{29}$$

where $\tilde{t} = U^\intercal t$.

Eq. (29) defines the local solver. Notably, it does not depend explicitly on the vertices but on their time values only. Thus, the local solver can be efficiently implemented as a function of t that returns the arrival time from a simplex. We now provide particular solutions of (29) for 2-D and 3-D cases.

4.3.4. Implementation for 2-D domains

The local solver receives a vector $(t_1, \dots, t_d)^\intercal$. By construction, t_i corresponds to a node in the simplex that is at distance $\sqrt{i}$ from q. In the 2-D case, the minimum of (29) lies either in the interior of the 1-D simplex $[v_1, v_2]$ or in the 0-D simplexes corresponding to the vertices v_1 and v_2. We then define:

$$f_i(t_i) := t_i + \sqrt{i}, \tag{30}$$

$$f_{12}(t_1,t_2) := \begin{cases} t_1 + \sqrt{1-(t_1-t_2)^2} & 0 < t_1 - t_2 < \frac{\sqrt{2}}{2} \\ \infty & \text{otherwise.} \end{cases} \tag{31}$$

Note that sub-indexes identify the particular simplex. Algorithm 4 shows the local solver's implementation.

Algorithm 4: `local_solver` function for 2-D domains.

Input: (t_1, t_2): times corresponding to nodes v_1 and v_2 defining a 1-D simplex
Output: optimal arrival time to a node q from a simplex

```
a ← f_1(t_1);
b ← f_2(t_2);
c ← f_12(t_1, t_2);
return min{a, b, c}
```

4.3.5. Implementation for 3-D domains

For 3-D domains, the local solver minimizes times over 0-D, 1-D, and 2-D simplexes. For 0-D simplexes, we have (30) with $i = 1, 2, 3$, whereas 1-D simplexes, besides (31), require additional functions:

$$\begin{aligned} f_{23}(t_2,t_3) &= \begin{cases} t_2 + \sqrt{2-2(t_2-t_3)^2} & \text{if } 0 \le t_2 - t_3 \le \frac{\sqrt{3}}{3}, \\ \infty & \text{otherwise} \end{cases} \\ f_{13}(t_1,t_3) &= \begin{cases} t_1 + \sqrt{1-\frac{(t_1-t_3)^2}{2}} & \text{if } 0 < t_1 - t_3 < \frac{2}{\sqrt{3}}, \\ \infty & \text{otherwise} \end{cases} \end{aligned} \tag{32}$$

The minimization over the 2-D simplex yields:

$$f_{123} = \begin{cases} t_1 + \sqrt{1-\Delta_{12}^2-\Delta_{23}^2} & \text{if } 0 < \Delta_{23} \le \Delta_{12} < \sqrt{1-\Delta_{12}^2-\Delta_{23}^2}, \\ \infty & \text{otherwise} \end{cases} \tag{33}$$

where $\Delta_{12} = t_1 - t_2$ and $\Delta_{23} = t_2 - t_3$. Algorithm 5 shows the local solver for 3-D domains.

Algorithm 5: `local_solver` function for 3-D domains.

Input: (t_1, t_2, t_3): times corresponding to nodes v_1, v_2 and v_3 defining a 2-D simplex
Output: optimal arrival time to a node q from the simplex

```
a ← f_1(t_1);
b ← f_2(t_2);
c ← f_3(t_3);
d ← f_12(t_1, t_2);
e ← f_23(t_2, t_3);
f ← f_13(t_1, t_3);
g ← f_123(t_1, t_2, t_3);
return min{a, b, c, d, e, f, g}
```

The executable code for the DFMM and examples are freely available in Julia and Matlab programming languages [50].

5. Numerical results

5.1. Method's performance in static scenarios

Before addressing dynamic situations, we first evaluate the method's performance in static settings and compare it with other approaches that solve the Eikonal equation. We consider a square arena $[0,1]\times[0,1]$ with a rectangular obstacle defined by the vertices: $(0.1, 0.4)$, $(0.9, 0.4)$, $(0.9, 0.5)$, and $(0.1, 0.5)$. At $t = 0$, the agent is at the origin. We employ the FMM, FSM, and DFMM to compute the time field $T(x)$ and evaluate their accuracy and execution time.

Table 1 summarizes the results. The FMM and FSM achieve comparable accuracy relative to the exact solution, with errors decreasing as the grid step size is refined. The DFMM, however, provides substantially higher accuracy in estimating the time field, owing to its more precise computation of the characteristic directions and wavefront orientation based on eight simplexes in the

Table 1
Performance of methods for different mesh sizes in a static scenario. Error is computed by $\|T_{\text{num}} - T_{\text{exact}}\|_\infty$.

	Absolute Error			Execution Time (s)		
	100×100	200×200	400×400	100×100	200×200	400×400
FMM	0.0269	0.0153	0.0086	0.0544	0.2193	0.8835
FSM	0.0178	0.0108	0.0064	**0.0181**	**0.0756**	**0.2960**
DFMM	**0.0081**	**0.0047**	**0.0029**	0.1284	0.5222	2.0727

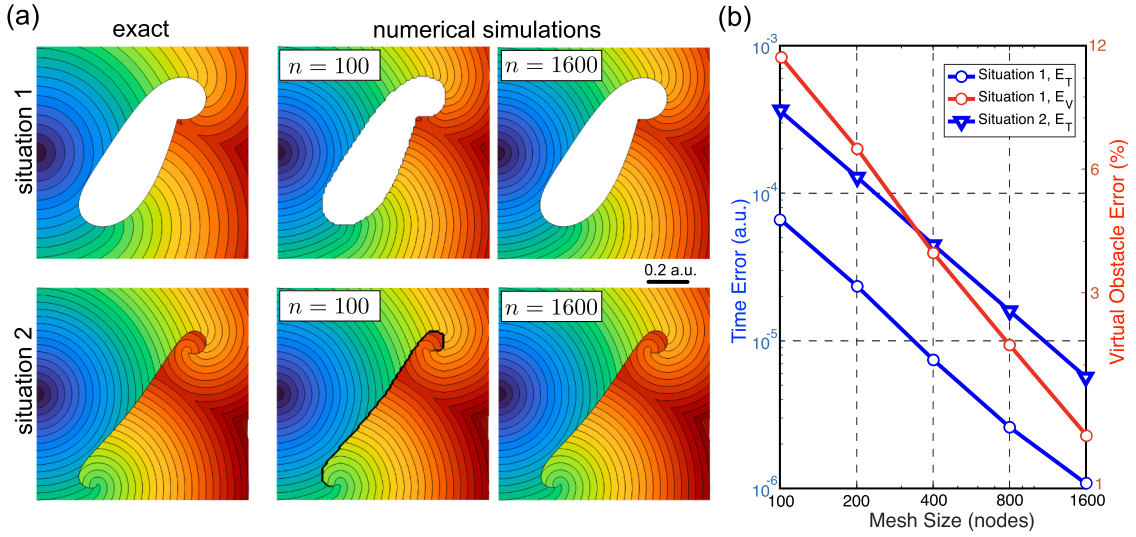

Fig. 5. Convergence of the DFMM to the exact solution. (a) Examples of time field T for two dynamic situations: a disk of radius $r = 0.3$ moves upwards with $x_0 = (0.5, -0.5)$, $w = 0.8$ (top) and $x_0 = (0.5, -1)$, $w = 1.4$ (bottom). *Left:* Exact solutions for the time field. *Right:* T computed for different mesh sizes ($n = 100$ and $n = 1600$ nodes) by the DFMM. (b) Error rates for different mesh sizes. The red curve corresponds to the Jaccard comparison of 2-D virtual obstacles (34), and the blue curves are for the Frobenius metric for the time field (35). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

2-D problem considered. As expected, the FSM is the fastest due to its efficient treatment of nodes. The DFMM requires additional computations to ensure causality in dynamic situations, which increases the computational cost per node. Nevertheless, the data fitting reveals an almost linear dependence of execution time on the number of nodes, confirming that its numerical complexity does not exceed $O(N \log N)$.

5.2. Convergence to exact solutions in dynamic situations

Recall that the FMM and FSM do not work well in dynamic situations. Then, let us analyze the convergence of the DFMM using two dynamic situations admitting exact solutions (Fig. 3, the cases with 2-D and 1-D virtual obstacles).

In the case of a 1-D virtual obstacle, the time field is finite in the whole domain, $T_{i,j} < \infty$. For 2-D virtual obstacles, we have regions with $T = \infty$. Moreover, such areas can differ between the exact and numerical solutions. Thus, to quantify the accuracy of a numerical solution, we need special metrics. Let $T^{\text{cm}}, T^{\text{ex}} \in \mathbb{R}^{n \times n}$ be matrices of the numerically computed and discretized exact solutions, respectively. Then, we also define the sets of indexes corresponding to nodes belonging to 2-D virtual obstacles: ω^{cm} and ω^{ex}, i.e., $T^{\text{cm}}_{ij} = \infty$, $\forall (i,j) \in \omega^{\text{cm}}$ and $T^{\text{ex}}_{ij} = \infty$, $\forall (i,j) \in \omega^{\text{ex}}$. The relative error of numerically found virtual obstacles is measured by the Jaccard distance [51]:

$$E_V := 1 - \frac{|\omega^{\text{cm}} \cap \omega^{\text{ex}}|}{|\omega^{\text{cm}} \cup \omega^{\text{ex}}|}, \tag{34}$$

where $|\cdot|$ is the cardinality of a set. To compare time fields, we use the modified Frobenius metric:

$$E_T = \frac{1}{n^2 - |\omega^{\text{cm}} \cup \omega^{\text{ex}}|} \left(\sum_{\substack{i,j=1 \\ (i,j) \notin \omega^{\text{cm}} \cup \omega^{\text{ex}}}}^{n} (T^{\text{cm}}_{i,j} - T^{\text{ex}}_{i,j})^2 \right)^{1/2} \tag{35}$$

Fig. 5(a), *left*, shows examples of the time fields for two different situations corresponding to exact solutions. Both dynamic situations describe the exploration process with a moving disk (Appendix B). Situation 1, with the disk's velocity $w < 1$, produces a 2-

D virtual obstacle, whereas Situation 2 corresponds to $w > 1$, and we get a 1D virtual obstacle. Thus, we can estimate the performance of the DFMM in two qualitatively different cases.

Fig. 5(a), *right*, illustrates time fields computed by the DFMM for different mesh sizes (100×100 and 1600×1600 nodes). We observe some minor deviations from the exact solutions for $n = 100$, while visual inspection reveals no differences with the fine grid. Then, we repeated the calculations with different mesh sizes and evaluated the error metrics, as defined in Eqs. (34) and (35). Fig. 5(b) shows that both errors decay exponentially with an increase in the number of nodes used to discretize the problem. Moreover, the exponent for E_T is about 1.5, corresponding to a superlinear convergence of the method.

5.3. Examples of exploring complex dynamic situations

5.3.1. Simulations in 2-D environments

Fig. 6(a), *left*, shows a situation with two small obstacles, disks of radius $r_d = 0.2$, moving over curved trajectories in an arena of $[-0.5, 5] \times [-2.5, 2.5]$ arbitrary units. The obstacle function is

$$g(x,t) = \min\left\{|Q_d(x - c_+(t))|, |Q_d(x - c_-(t))|\right\} - 1, \tag{36}$$

where $Q_d = \mathrm{diag}(2r_d, 2r_d)^{-1}$ is the scaling matrix and $c_\pm(t) = (\rho(t)\cos(\theta(t)), \pm\rho(t)\sin(\theta(t)))^T$ define the positions of the centers of the obstacles with

$$\theta(t) = \begin{cases} 0.3\sin(0.9(t-1)), & \text{if } t \in [1, 1 + \frac{\pi}{0.9}) \\ 0, & \text{otherwise} \end{cases}, \qquad \rho(t) = \begin{cases} 1, & \text{if } t \in [0, 1) \\ 1 + 0.78(t-1) & \text{if } t \in [1, 1 + \frac{\pi}{0.9}) \\ 1 + \frac{0.78\pi}{0.9} & \text{otherwise.} \end{cases} \tag{37}$$

Four consecutive snapshots (Fig. 6(a), *middle*) illustrate how the wavefront propagates through the environment and interacts with obstacles (for better visualization, see supplementary videos). The final time field (*right*) displays a large virtual obstacle that is unreachable to the agent. Thus, the number, size, and location of virtual obstacles strongly depend on the fine structure of the situation (see also Fig. 3).

The second situation mimics the problem of pathfinding in a labyrinth with moving obstacles (Fig. 6(b)). The construction of the obstacle function $g(x,t)$ is similar to case (a). However, now it includes static obstacles, two moving disks, and a rotating bar. Rectangular obstacles can be obtained by using the infinity norm and the corresponding scaling matrix Q_d in (36). The rotating bar employs a rotation matrix applied prior to scaling: $g_{\mathrm{bar}}(x,t) = |Q_d R(\phi(t))(x - c)|_\infty$. Multiple obstacles, case (b), make the front propagation intricate. The agent may need to follow significantly different trajectories to reach close points in the labyrinth.

The third example considers a domain with periodic boundary conditions, i.e., Ω is a torus (Fig. 6(c)). Such conditions are typical in problems involving driving manipulators (e.g., artificial arms), where the configuration space is defined by the angles of the joints [41,52]. The DFMM can find paths in this space and provide driving control to the manipulator that avoids obstacles and reaches a target (that could be one of the obstacles). Notice that periodic boundary conditions can lead to new trajectories unavailable in a standard space or ban existing ones.

5.3.2. Simulation of a 3-D environment

Fig. 7 illustrates how the DFMM works in 3-D environments. To maintain visual clarity, we simulated a relatively simple dynamic situation with two moving obstacles: a cylinder traversing the (x, y)-plane and a ball moving across the scene. The figure presents only snapshots of the wavefront at different time instants, i.e., the time field $T(x, y, z)$ is not shown (Fig. 7(a), a video is available). The grid included 8×10^6 nodes, and the simulation lasted less than 5 min on a standard PC (a Mac mini, M1, 8 GB).

Although the situation is rather simple, the front's interaction with the obstacles is entangled. While there is a resemblance between the interaction of the front with the cylinder in 3-D and a disk in 2-D (Fig. 5), the 3-D picture is a whole new level of complexity. The 2-D trace changes in the z-direction, adding a new dimension to the dynamics. The most intricate dynamics emerge when the front interacts with the two objects, breaking the front and leaving a hole, which is later filled by the front. The final virtual obstacle is a blend of 3-D and 2-D parts.

Fig. 7(b) shows the final configuration of the time field $T(x, y, z)$. We observe a diversity of different regions in the 3-D space. They can be reached by an agent at significantly different times and following complex trajectories.

5.4. Exploration of nonhomogeneous environments

Trajectory planning typically assumes that movement occurs in homogeneous environments, such as a wheeled robot rolling over a flat floor in a building [53] or a drone flying in 3-D space [54]. However, in heterogeneous environments, such as when a rover traverses different terrains, the agent's travel speed can vary significantly depending on its location.

Fig. 8(a) illustrates a scenario in which an agent (red circle) must cross a procession of people moving with velocity $w = 1.4$. The domain has terrains with different properties. The upper part allows for a higher velocity, while the lower part resists movement (Fig. 8(a), fast and slow tracks, respectively). We model this situation by assuming that the agent's velocity is

$$v_{\mathrm{agent}} = \begin{cases} 1, & \text{if } \ y < 0.8 \\ 1 + a(y - 0.8), & \text{otherwise,} \end{cases} \tag{38}$$

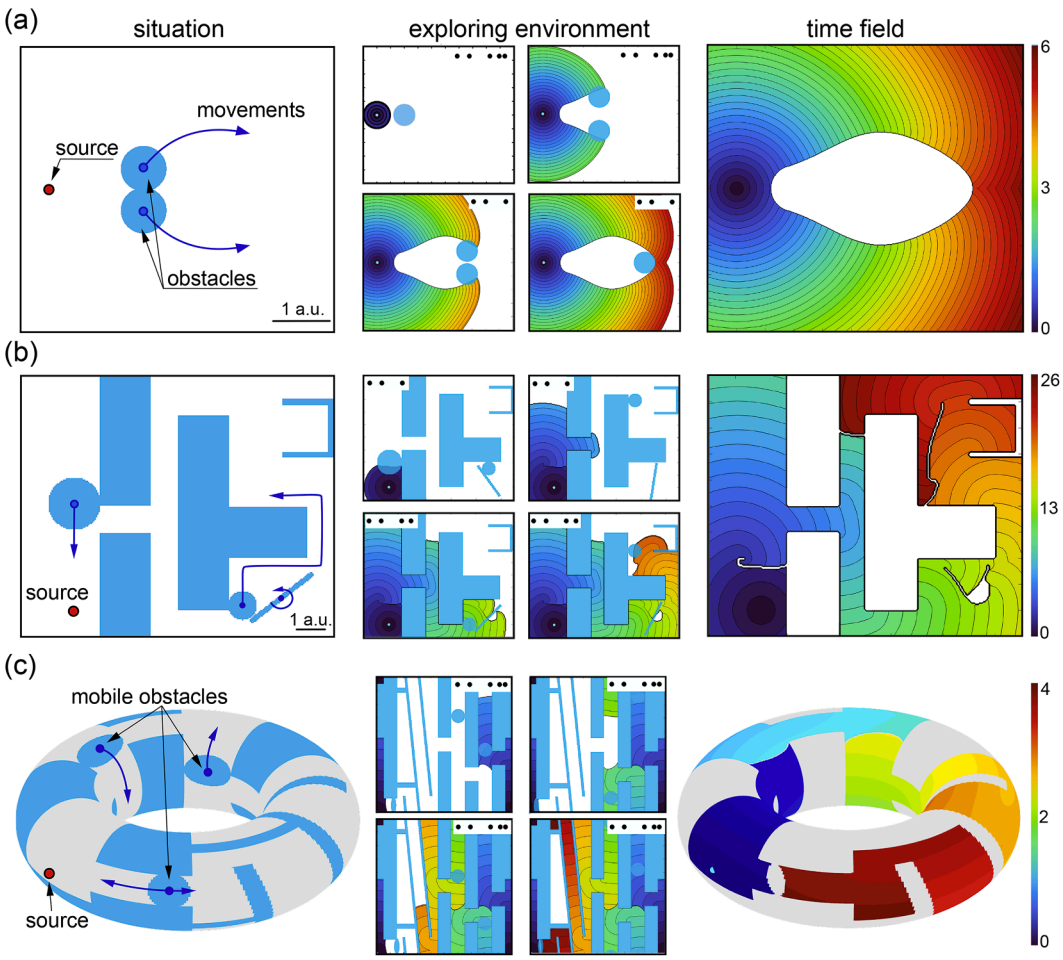

Fig. 6. Examples of application of the DFMM to three dynamic situations in 2-D environments. a) Small obstacles can create a big virtual obstacle. b) Movement in a labyrinth with mobile obstacles. c) Dynamics on a torus (videos are available at the link).

where a is the heterogeneity constant ($a = 0$ for a homogeneous case). Then, the results of exploring such an environment depend on a.

To adapt the DFMM to the nonhomogeneous case, we use the following procedure, generalizing the time filtering function and computation of optimal arrival time (Sections 4.3.1 and 4.3.3, respectively). For each node q, consider the time values T_1 and T_2 of the nodes q_1 and q_2 in a simplex (triangle in 2-D). As before, the nodes are at distances 1 and $\sqrt{2}$ (Δt rescaled). Then, the optimal arrival time is

$$T_q = T_1 + \min_{0\le\lambda\le 1}\left\{\frac{\sqrt{1+\lambda^2}}{v_q} - \lambda\Delta\right\}, \tag{39}$$

where v_q is the agent's speed at node q and $\Delta := T_1 - T_2$. Developing (39), we get the closed formula:

$$T_q = \begin{cases} T_1 + 1/v_q, & \text{if } \Delta \le 0 \\ T_1 + \sqrt{\dfrac{1}{v_q^2} - \Delta^2}, & \text{if } 0 < \Delta < \dfrac{1}{\sqrt{2}v_q} \\ T_2 + \sqrt{2}/v_q, & \text{otherwise.} \end{cases} \tag{40}$$

Filtering is handled in the same way. For instance, once the time values of the neighboring nodes are gathered, we discard those values that are "far" in the past, taking into account the speed value associated with the node. That is, if node p is the "inviting node", and q is the node whose time value is filtered, we discard q if

$$T_p - T_q > \sqrt{2}/v_q. \tag{41}$$

Note that if the local speed is 1 ($a = 0$), we obtain the same filtering function as in Section 4.3.1.

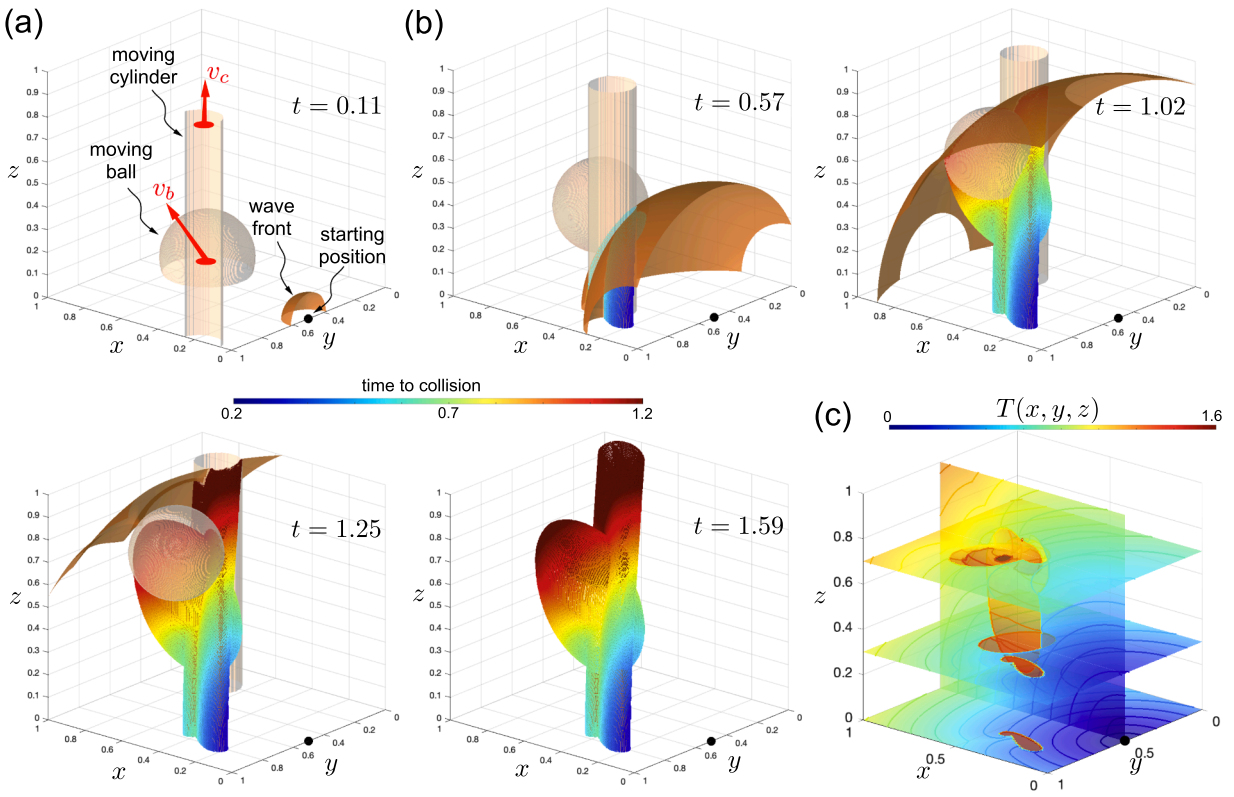

Fig. 7. Example of application of the DFMM to a dynamic situation in a 3-D environment. a) The domain contains two moving obstacles (reddish gray): a cylinder (radius: 0.1; initial position: $(0.1, 1.1, 0)$; velocity: $v_c = (0.5, -0.7, 0)$) and a ball (radius: 0.2; initial position: $(0.7, 0.3, 0)$; velocity: $v_b = (-0.1, 0.3, 0.6)$). The starting point of the agent (a black dot) is at $(0, 0.5, 0)$. The wavefront (brown) propagates from the starting point. b) Four consecutive snapshots for different time instants (only the wavefront is shown, $T(x, y, z)$ is not). Colors from blue to red at virtual obstacles correspond to the time of the collision of the wavefront with real obstacles. c) Final time field $T(x, y, z)$ (four 2-D slices are shown). A video is available at the link. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

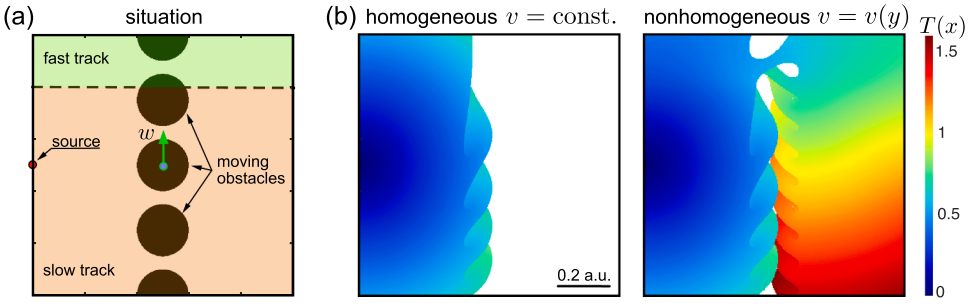

Fig. 8. Application of the DFMM to a dynamic situation in homogeneous and heterogeneous environments. a) Scenario. The agent (red dot) must cross a line of people (black circles) moving with velocity w. The domain consists of two parts, each admitting different velocities for the agent (orange and green areas corresponding to the slow and fast tracks). b) Cognitive maps obtained by the DFMM for the homogeneous (left, $a = 0$) and nonhomogeneous (right, $a = 10$) cases. In the former, the agent's velocity is constant, and it is trapped in the left part of the arena. In the nonhomogeneous case, the fast track allows the agent to slip through the line of people. Thus, it can reach the destination. A video is available at the link. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

We now apply the generalized DFMM to the example shown in Fig. 8(a). Fig. 8(b) illustrates the cognitive maps obtained for the homogeneous ($a = 0$, left) and nonhomogeneous ($a = 10$, right) cases. In the former, a virtual obstacle of a rather complex shape is formed, and the agent becomes trapped in the left part of the arena, unable to cross the line of moving people. However, the existence of the fast track in the nonhomogeneous case (right) creates a hole in the virtual obstacle, allowing the agent to slip through and reach the destination.

6. Conclusion

In this work, we tackled the complex problem of propagating a circular wavefront through a medium with multiple moving obstacles. Although this is a classical problem, its numerical treatment has not been adequately addressed in the literature. As we have seen, it is ill-posed and unstable, presenting significant challenges for numerical algorithms. Even small perturbations to the velocities or initial positions of obstacles can significantly change the solution, i.e., the final time field.

Although the problem is strongly related to the Eikonal equation, its direct application has drawbacks. The generalized Eikonal equation discussed here can be formally satisfied, but it is not instrumental for numerical methods. A viscosity solution can be employed as an alternative; however, it requires additional boundary conditions, which may unnecessarily complicate the problem. Moreover, viscosity solutions are challenging to handle numerically due to the presence of small parameters.

We proposed a particle-like description of the problem, which allowed us to introduce a novel algorithm, the Dynamic Fast Marching Method. The algorithm is designed to search for accurate solutions of problems of arbitrary complexity in configuration spaces of arbitrary dimension, which can be critical for aerial or submarine navigation (3D) and modern manipulators with many degrees of freedom requiring even higher dimensions. Furthermore, the proposed method can work in time-extended workspaces, which allows relaxing the unit speed restriction considered in the current work.

Our approach is based on the well-known Fast Marching Method, but it incorporates crucial modifications that enable its application to dynamic situations by preserving causality in the interaction between the wavefront and moving obstacles. The algorithm generates collision-free trajectories and can also serve as an interception strategy for capturing a moving target amid multiple obstacles. We have tested the algorithm in a variety of 2-D and 3-D scenarios, including cases with exact solutions to verify its convergence. We have demonstrated its effectiveness and computational efficiency by comprehensive numerical simulations. The algorithm's computational complexity scales as $O(N \log N)$ with the number of nodes N. Performance can be further improved by incorporating ideas from fast sweeping methods, while ensuring that the causality of wave interactions with moving obstacles is preserved.

CRediT authorship contribution statement

Gerardo E. Oleaga: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization; **Daniel Ortega-Lozano:** Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Valeri A. Makarov:** Writing – review & editing, Writing – original draft, Visualization, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Data availability

The code is freely available to download. The paper provides a link to the code.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Valeri A. Makarov reports financial support was provided by Spanish Ministry of Science and Innovation. Daniel Ortega-Lozano reports financial support was provided by Spanish Ministry of Universities. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been supported by the Spanish Ministry of Science and Innovation (Project No. PID2021-124047NB-I00). D.O-L. benefited from a grant from the Spanish Ministry of Universities (contract FPU22/01848).

Appendix A. Virtual obstacle for a moving rectangle

Consider the dynamic situation shown in Fig. 1(a) with the domain $\Omega = (0,1)^2$. The starting position for exploration of the environment (primary particles' source) is at $(0, 0.5 + \epsilon)$, where $\epsilon > 0$ is arbitrarily small. The dynamics of a rectangular obstacle is given by:

$$g(x,y,t) = y - 1/2 + w \max\{0, t - 1/2\}, \quad (x,y) \in \overline{\Omega},\ t \geq 0. \tag{A.1}$$

Thus, for $t \in [0, 1/2]$, the bottom half of $\overline{\Omega}$ is occupied by the obstacle, and for $t > 1/2$ the obstacle moves down with a constant velocity $w \in (0,1)$.

For $t \in [0, 1/2]$, the obstacle does not move, and hence the boundary of the virtual obstacle is:

$$\partial V_t^a = \{(x,y) : y = 1/2, x < t\}, \quad t \in [0, 1/2]. \tag{A.2}$$

Thus, for $t = 1/2$, we get a secondary source of particles at $(1/2, 1/2 + \delta)$, where $\delta > 0$ and $\delta \to 0$. This source emits particles in all directions with the velocity vector $u = (\alpha, \pm\sqrt{1-\alpha^2})$, with $\alpha \in [-1, 1]$. It generates all $T(x, y)$ for $y \le 1/2$. However, the values $\alpha \le 0$ correspond to particles going into the past and/or colliding with the virtual obstacle (A.2). Thus, we can rule them out.

Consider now $\alpha > 0$ and the negative sign in the y-axis of the velocity vector. If $\delta \to 0$, then after time dt, the particles avoid collisions for α satisfying:

$$\sqrt{1-\alpha^2}\mathrm{d}t < w\mathrm{d}t. \tag{A.3}$$

Therefore, $\alpha = \sqrt{1-w^2}$ is the critical vertical velocity, which defines the boundary of the virtual obstacle:

$$\partial V_t^b = \left\{ (x, y) : x = 1/2 + \sqrt{1-w^2}(t - 1/2),\ y = 1/2 - w(t - 1/2) \right\},\quad t \ge 1/2. \tag{A.4}$$

Eqs. (A.2) and (A.4) define the virtual obstacle's boundary, which is a continuous piecewise straight line. Now, we have the time field for the problem:

$$T(x, y) = \begin{cases} \sqrt{x^2 + (y-0.5)^2}, & \text{if } y \ge 0.5 \\ 0.5 + \sqrt{(x-0.5)^2 + (y-0.5)^2}, & \text{if } y < 0.5,\ y > \frac{x-l}{1-2l} \\ \infty, & \text{otherwise,} \end{cases} \tag{A.5}$$

where $l = 0.5(1 + \sqrt{1-w^2}/w)$. Fig. 1(b) illustrates the solution.

Appendix B. Exact solution for a moving disk

Consider a dynamic situation with a single moving obstacle in the form of a disk of radius r (Fig. 3(a)). The obstacle's function is:

$$g(x, y, t) = (x - x_c)^2 + (y - y_c - wt)^2 - r^2,\quad (x, y) \in \overline{\Omega},\ t \ge 0, \tag{B.1}$$

where $w > 0$ is the disk's velocity along the y-axis and (x_c, y_c) is the position of the disk's center at $t = 0$, such that $x_c > r$. The agent's initial position is $(0, 0)$.

Using (B.1), the boundary of the disk can be parameterized as:

$$\begin{cases} x(t, \varphi) = A(\varphi), \\ y(t, \varphi) = B(\varphi) + wt, \end{cases} \quad \varphi \in [0, 2\pi), \tag{B.2}$$

where we denoted:

$$A(\varphi) := x_c + r\cos\varphi,\quad B(\varphi) := y_c + r\sin\varphi. \tag{B.3}$$

Note that $A > 0$.

Now, we proceed in two steps to find the field $T(x, y)$. First, we calculate the boundary of a virtual obstacle created by collisions of particles with the disk. Then, we can partition Ω into several regions and find $T(x, y)$ for each of them.

Boundary of virtual obstacles.

As we see below, it may occur that no virtual obstacle appears in Ω. Then, the solution is trivial (Fig. 3(c), $w = 15$). Otherwise, the virtual boundary ∂V can have two parts: i) due to collisions with direct rays on the "illuminated" side of the obstacle and ii) due to sliding collisions on the "shadow" side (Fig. 3(b)).

Collision with direct particles' rays. Let us find collisions of the disk with particles emitted at $(0, 0)$ and propagating along straight rays. Such collisions satisfy $g(x, y, T) = 0$ and $T = \sqrt{x^2 + y^2}$. Solving these equations for $w \ne 1$, we get:

$$T(\varphi) = \frac{wB(\varphi) \pm \sqrt{B^2(\varphi) + (1-w^2)A^2(\varphi)}}{1-w^2}, \tag{B.4}$$

and then $x = A(\varphi)$, $y = B(\varphi) + wT(\varphi)$ define the "illuminated" part of the virtual obstacle's boundary. Note that the minus sign in Eq. (B.4) corresponds to a second virtual obstacle that appears at $w > 1$.

Eq. (B.4) has no positive solutions for high speeds $w > w_{\mathrm{cr}}$ (a good upper estimate of the critical velocity is $\sqrt{1 + (|y_c| + r)^2/(x_c - r)^2}$, which gives about 15.5 for the parameters used in Fig. 3(c)). Then, we have the trivial solution for the time field (Fig. 3(c), $w = 15$):

$$T = \sqrt{x^2 + y^2},\quad w > w_{\mathrm{cr}}. \tag{B.5}$$

Let us now consider moderate speeds, $w \le w_{\mathrm{cr}}$. Then, Eq. (B.4) admits positive solutions for $\varphi \in [\varphi_1, \varphi_2]$. Moreover, the limit angles φ_1, φ_2 correspond to rays tangent to a virtual obstacle. Recalling that a normal vector to ∂V is $\nabla g + \frac{\partial g}{\partial t}\nabla T$ and the ray direction is given by $\nabla T = \frac{1}{T}(x, y)^{\mathsf{T}}$, we get:

$$\nabla T \cdot \left(\nabla g + \frac{\partial g}{\partial t}\nabla T \right) = \frac{1}{T}(x, y)^{\mathsf{T}} \cdot \nabla g + \frac{\partial g}{\partial t} = 0,\quad \varphi \in \{\varphi_1, \varphi_2\}. \tag{B.6}$$

Substituting (B.1) into (B.6), we have the following equation for determining $\varphi_{1,2}$:

$$A(\varphi_{1,2})\cos\varphi_{1,2} + B(\varphi_{1,2})\sin\varphi_{1,2} = 0. \tag{B.7}$$

To solve it, we denote $z := e^{i\varphi_k}$ $(k = 1, 2)$ and get:

$$\bar{\alpha} z^2 + 2z + \alpha = 0, \tag{B.8}$$

where $\alpha = \frac{x_c}{r} + \frac{y_c}{r} i$ is the relative position of the disc's center on the complex plane. Then, solving for z in (B.8), we get $z_{1,2} = e^{i(\xi + \eta_{1,2})}$, where

$$\xi = \arctan \frac{y_c}{x_c}, \quad \eta_{1,2} = \pi \pm \arctan \sqrt{|\alpha|^2 - 1}. \tag{B.9}$$

Thus, for moderate obstacle velocities, the direct ray boundary of the virtual obstacle is

$$\partial V_{\mathrm{dr}} = \left\{ (A(\varphi), B(\varphi) + wT(\varphi)) : \varphi \in [\varphi_1, \varphi_2] \right\}, \tag{B.10}$$

where $A(\varphi)$, $B(\varphi)$, $T(\varphi)$ are given by Eqs. (B.3), (B.4), and $\varphi_{1,2} = \xi + \eta_{1,2}$ with ξ, $\eta_{1,2}$ provided by Eq. (B.9).

Sliding collisions. Limit points corresponding to angles $\varphi_{1,2}$ act as secondary sources emitting particles that start the exploration of the dark side of the disk. When a particle reaches a point in the virtual boundary, an infinitesimally close point emits a new particle that reaches another point infinitesimally shifted over the disk's boundary. This way, particles slide around the disk (that keeps moving) and form the "sliding" part of the virtual boundary ∂V_{sl} that consists of two branches starting from φ_1 and φ_2.

To describe this dynamic process, we consider one of the branches, e.g., corresponding to φ_1. Let $s(t)$ be the sliding trajectory. Then, it satisfies:

$$\begin{cases} s(T(\varphi_1)) = (A(\varphi_1), B(\varphi_1) + wT(\varphi_1)) \\ s(t) = (A(\varphi(t)), B(\varphi(t)) + wt), \quad t > T(\varphi_1) \\ \|\dot{s}\| = 1, \end{cases} \tag{B.11}$$

where the dot denotes the time derivative. The last two equations of (B.11) yield:

$$r^2 \sin^2(\varphi)\dot{\varphi}^2 + (r\cos(\varphi)\dot{\varphi} + w)^2 = 1. \tag{B.12}$$

Resolving this equation for $\dot{\varphi}$, we arrive at the initial value problem:

$$\begin{cases} \dot{\varphi} = -\frac{w}{r}\left(\cos\varphi \pm \sqrt{\frac{1}{w^2} - \sin^2\varphi}\right) \\ \varphi(T(\varphi_{1,2})) = \varphi_{1,2}, \end{cases} \tag{B.13}$$

where plus and minus correspond to the branches starting from φ_1 and φ_2, respectively. For small velocities, $w < 1$, the solution of (B.13) is defined for all φ. For high velocities, $1 < w < w_{\mathrm{cr}}$, the solution is restricted to either $[-\arcsin\frac{1}{w}, \arcsin\frac{1}{w}]$ or $[\pi - \arcsin\frac{1}{w}, \pi + \arcsin\frac{1}{w}]$.

Notice that the solutions of (B.13) form two branches of the virtual obstacle boundary and hence must be truncated if they intersect between each other or with ∂V_{dr}. Fig. 3(b) and (c) for $w = 0.99$ show truncations due to the intersection of the two sliding branches between each other and with ∂V_{dr}, respectively. Furthermore, the sliding branches must be truncated if the disk moves faster than the particles, i.e., $\dot{s}(t) \cdot n(s(t)) < -1$, where $n = (\cos\varphi_i(t), \sin\varphi_i(t))$ is the outwards normal to the disk at position $s(t)$ and time t. This condition gives the limit angles for sliding collisions if $w > 1$:

$$\varphi_{\mathrm{lm}} \in \left\{ -\pi + \arcsin\left(\frac{1}{w}\right), -\arcsin\left(\frac{1}{w}\right) \right\}. \tag{B.14}$$

Notice that the limit angles delimit the existence of a solution of (B.13).

If either of the branches satisfies $\varphi(t^*) = \varphi_{\mathrm{lm}}$, then the boundary has an open end, and hence the virtual obstacle reduces to a 1D curve (e.g., Fig. 3(c), $w = 1.5$).

Calculation of the time field

Given the virtual obstacle's boundary ∂V, we partition Ω in two regions: accessible to direct rays or directly "illuminated" part L and the remaining or dark-side part D:

$$L = \left\{ x : \ [0, x) \cap \partial V_{\mathrm{dr}} = \emptyset \right\}, \quad D = \Omega \setminus (L \cup V). \tag{B.15}$$

Then the time field is:

$$T(x) = \begin{cases} |x - x_0|, & \text{if } x \in L, \\ \inf\limits_{v \in \partial V_{\mathrm{sl}}} \{T(v) + |x - v| : \ (v - x) \cdot n(v) \geq 0 \ \& \ (v, x] \cap \partial V_{\mathrm{sl}} = \emptyset\} & \text{if } x \in D. \end{cases} \tag{B.16}$$

Remark 4. 1. Note that $\partial V_{\mathrm{dr}} \subset L$ and hence it gets times by direct rays. 2. If a branch of ∂V_{sl} has an open end, the limit point is used for evaluating $T(x)$ in all directions.

References

[1] L. Cavaleri, J.-H.G.M. Alves, F. Ardhuin, et al., Wave modelling - the state of the art, Prog. Oceanogr. 75 (4) (2007) 603–674. https://doi.org/10.1016/j.pocean.2007.05.005

[2] J.A. Villacorta-Atienza, V.A. Makarov, Wave-processing of long-scale information by neuronal chains, PLoS One 8 (2013) 57440. https://doi.org/10.1371/journal.pone.0057440

[3] H. Ortega-Arranz, A. Gonzalez-Escribano, D.R. Llanos, The Shortest-Path Problem: Analysis and Comparison of Methods, Springer Nature, 2022.

[4] J. Canny, J. Reif, New lower bound techniques for robot motion planning problems, in: 28th Annual Symp. on Found. of Comput. Sci. (SFCS 1987), IEEE, 1987, pp. 49–60. https://doi.org/10.1109/SFCS.1987.42

[5] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, D. Lane, Path planning for autonomous underwater vehicles, IEEE Trans. Robot. 23 (2) (2007) 331–341. https://doi.org/10.1109/TRO.2007.895057

[6] J.A. Villacorta-Atienza, M.G. Velarde, V.A. Makarov, Compact internal representation of dynamic situations: neural network implementing the causality principle, Biol. Cybern. 103 (2010) 285–297. https://doi.org/10.1007/s00422-010-0398-2

[7] E.C. Tolman, Cognitive maps in rats and men, Psychol. Rev. 55 (4) (1948) 189. https://doi.org/10.1037/h0061626

[8] J.A. Villacorta-Atienza, C.C. Tapia, S. Díez-Hermano, A. Sánchez-Jiménez, S. Lobov, et al., Static internal representation of dynamic situations reveals time compaction in human cognition, J. Adv. Res. 28 (2021) 111–125. https://doi.org/10.1016/j.jare.2020.08.008

[9] L. Janson, E. Schmerling, A. Clark, M. Pavone, Fast marching tree: a fast marching sampling-based method for optimal motion planning in many dimensions, Int. J. Robot. Res. 34 (7) (2015) 883–921.

[10] E.W. Dijkstra, A note on two problems in connexion with graphs, in: Edsger Wybe Dijkstra: His Life, Work, and Legacy, 2022, pp. 287–290. https://doi.org/10.1007/BF01386390

[11] P.E. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Syst. Sci. Cybern. 4 (2) (1968) 100–107. https://doi.org/10.1109/TSSC.1968.300136

[12] D. Foead, A. Ghifari, M.B. Kusuma, N. Hanafiah, E. Gunawan, A systematic literature review of A* pathfinding, Procedia Comput. Sci. 179 (2021) 507–514. https://doi.org/10.1016/j.procs.2021.01.034

[13] Z. Wang, X. Fan, A systematic review and analysis of A-star pathfinding algorithms and its variations, Fourth Int. Conf. on Signal Process. and Mach. Learn. (CONF-SPML 2024) 13077 (2024) 157–182. https://doi.org/10.1117/12.3027132

[14] M. Martelli, A. Virdis, A. Gotta, P. Cassara, M.D. Summa, An outlook on the future marine traffic management system for autonomous ships, IEEE Access 9 (2021) 157316–157328. https://doi.org/10.1109/ACCESS.2021.3130741

[15] F. Ahmed, J.C. Mohanta, A. Keshari, et al., Recent advances in unmanned aerial vehicles: a review. Arab, J. Sci. Eng. 47 (2022) 7963–7984. https://doi.org/10.1007/s13369-022-06738-0

[16] S.A. Lobov, N.P. Krilova, V.A. Makarov, Arcade game testing of generalized cognitive maps in humans, in: Third Int. Conf. Neurotechnol. Neurointerfac. (CNN), Kaliningrad, 2021, pp. 61–63. https://doi.org/10.1109/CNN53494.2021.9580220

[17] Y. Wang, X. Li, J. Zhang, S. Li, Z. Xu, X. Zhou, Review of wheeled mobile robot collision avoidance under unknown environment, Sci. Progress 104 (3) (2021). https://doi.org/10.1177/00368504211037771

[18] L. Foschini, J. Hershberger, S. Suri, On the complexity of time-dependent shortest paths, in: Proc. 2011 Ann. ACM-SIAM Sympos. Discr.ete Algor, 2011, pp. 327–341.

[19] B.C. Dean, Shortest Paths in FIFO Time-Dependent Networks: Theory and Algorithms Technical Report, MIT 2004.

[20] C. Calvo-Tapia, et al., Semantic knowledge representation for strategic interactions in dynamic situations, Front. Neurorob. 14 (2020) 4.

[21] A. Stentz, The focussed D* algorithm for real-time replanning, in: Int. Joint Conf. Artif. Intell., 1995.

[22] H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, S. Thrun, Principles of Robot Motion, MIT Press, 2007.

[23] M. Moussaid, D. Helbing, G. Theraulaz, How simple rules determine pedestrian behavior and crowd disasters, Proc. Natl. Acad. Sci. U.S.A 108 (2011) 6884–6888.

[24] K. Karur, N. Sharma, C. Dharmatti, J.E. Siegel, A survey of path planning algorithms for mobile robots, Vehicles 3 (2021) 448–468.

[25] J.A. Villacorta-Atienza, C. Calvo, V.A. Makarov, Prediction-for-CompAction: navigation in social environments using generalized cognitive maps, Biol. Cybern. 109 (3) (2015) 307–320. https://doi.org/10.1007/s00422-015-0644-8

[26] J.A. Sethian, Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Science, Cambridge Monographs on Appl. and Comput. Math 3, Cambridge University Press, 1996.

[27] M. Falcone, The Minimum Time Problem and its Applications to Front Propagation, in: Motion by Mean Curvature and Related Topics, de Gruyter, Berlin, 1994. https://doi.org/10.1515/9783110870473.70

[28] M.G. Crandall, P.-L. Lions, Viscosity solutions of Hamilton-Jacobi equations, Trans. Amer. Math. Soc. 277 (1) (1983) 1–42. https://doi.org/10.1090/S0002-9947-1983-0690039-8

[29] J.N. Tsitsiklis, Efficient algorithms for globally optimal trajectories, IEEE Trans. Autom. Contr. 40 (9) (1995) 1528–1538. https://doi.org/10.1109/9.412624

[30] J.A. Sethian, A fast marching level set method for monotonically advancing fronts, Proc. Natl. Acad. Sci. U.S.A 93 (4) (1996) 1591–1595. https://doi.org/10.1073/pnas.93.4.1591

[31] J.A. Sethian, Fast marching methods, SIAM Rev. 41 (2) (1999) 199–235. https://doi.org/10.1137/S0036144598347059

[32] A. Telea, An image inpainting technique based on the fast marching method, J. Graph. Tools 9 (1) (2004) 23–34. https://doi.org/10.1080/10867651.2004.10487596

[33] M. Breuß, E. Cristiani, P. Gwosdek, O. Vogel, An adaptive domain-decomposition technique for parallelization of the fast marching method, Appl. Math. Comput. 218 (1) (2011) 32–44. https://doi.org/10.1016/j.amc.2011.05.041

[34] A. Chacon, A. Vladimirsky, A parallel two-scale method for eikonal equations, SIAM J. Sci. Comput. 37 (1) (2015) A156–A180. https://doi.org/10.1137/12088197X

[35] H. Zhao, A fast sweeping method for eikonal equations, Math. Comput. 74 (250) (2005) 603–627. https://doi.org/10.1090/S0025-5718-04-01678-3

[36] J. Qian, Y.T. Zhang, H.K. Zhao, Fast sweeping methods for Eikonal equations on triangular meshes, SIAM J. Numer. Anal. 45 (1) (2007) 83–107. https://doi.org/10.1137/050627083

[37] A. Vladimirsky, Static PDEs for time-dependent control problems, Interface Free Bound 8 (2006) 281–300. https://doi.org/10.4171/ifb/144

[38] Y. Wang, G.S. Chirikjian, A new potential field method for robot path planning, ICRA. Millennium conference. IEEE Int. Conf. Robotics Automation 2 (2000) 977–982.

[39] Y. Zheng, X. Shao, Z. Chen, J. Zhang, Improvements on the Virtual Obstacle Method, 2020, pp. 1–9. https://doi.org/10.1177/1729881420911763

[40] C.C. Tapia, I.Y. Tyukin, V.A. Makarov, Fast social-like learning of complex behaviors based on motor motifs, Phys. Rev. E 97 (5) (2018) 052308. https://doi.org/10.1103/PhysRevE.97.052308

[41] J.A. Villacorta-Atienza, C. Calvo, S. Lobov, V.A. Makarov, Limb movement in dynamic situations based on generalized cognitive maps, Math. Model. Nat. Phenom. 12 (4) (2017) 15–29. https://doi.org/10.1051/mmnp/201712403

[42] A.G. Gonzalez-Rodriguez, A. Gonzalez-Rodriguez, F. Castillo-Garcia, Improving the energy efficiency and speed of walking robots, Mechatronics 24 (5) (2014) 476–488. https://doi.org/10.1016/j.mechatronics.2014.05.004

[43] J.K. Rathkey, C.M. Wall-Scheffler, People choose to run at their optimal speed, Am J. Phys. Anthropol. 163 (2017) 85–93. https://doi.org/10.1002/ajpa.23187

[44] A. Gasparetto, P. Boscariol, A. Lanzutti, R. Vidoni, Path Planning and Trajectory Planning Algorithms: A General Overview, 29 of Cham, Springer, 2015. https://doi.org/10.1007/978-3-319-14705-5_1

[45] S.C. Billups, K.G. Murty, Complementarity problems, J. Computat. App. Math. 124 (1) (2000) 303–318. https://doi.org/10.1016/S0377-0427(00)00432-5

[46] X. Yan, W. Wang, C. Huang, L. Li, A new path planning method for AUV based on the Navier-Stokes equations for ocean currents, Math. Comput. Simul. 222 (2024) 199–208. https://doi.org/10.1016/j.matcom.2023.08.030

[47] D. Wang, D. Tan, L. Liu, Particle swarm optimization algorithm: an overview, Soft Comput. 22 (2) (2018) 387–408. https://doi.org/10.1007/s00500-016-2474-6
[48] S. Thrun, W. Burgard, D. Fox, P. Robotics, Probabilistic Robotics, MIT Press, 2005.
[49] J.A. Sethian, A. Vladimirsky, Fast methods for the Eikonal and related Hamilton-Jacobi equations on unstructured meshes, Proc. Natl. Acad. Sci. U.S.A 97 (11) (2000) 5699–5703. https://doi.org/10.1073/pnas.090060097
[50] Codes in Julia and Matlab programming languages are available at, http://blogs.mat.ucm.es/vmakarov/downloads/ Supplemental video examples are available at, https://blogs.mat.ucm.es/vmakarov/dfmm_examples/.
[51] P. Jaccard, The distribution of the flora in the alpine zone, New Phytol. 11 (2) (1912) 37–50. https://doi.org/10.1111/j.1469-8137.1912.tb05611.x
[52] T. Sandakalum, M.H. Ang Jr, Motion planning for mobile manipulators-a systematic review, Machines 10 (2) (2022) 97. https://doi.org/10.3390/machines10020097
[53] A.V. Savkin, A.S. Matveev, M. Hoy, C. Wang, Safe Robot Navigation Among Moving and Steady Obstacles, Butterworth-Heinemann, 2015.
[54] S. Aggarwal, N. Kumar, Path Planning Techniques for Unmanned Aerial Vehicles: A Review, Solutions, and Challenges, Computer Communic, 149, 2020. https://doi.org/10.1016/j.comcom.2019.10.014