Wavefront Marching Methods: a unified algorithm to solve Eikonal and static Hamilton-Jacobi equations.

Brais Cancela, and Amparo Alonso-Betanzos

Abstract—This paper presents a unified propagation method for dealing with both the classic Eikonal equation, where the motion direction does not affect the propagation, and the more general static Hamilton-Jacobi equations, where it does. While classic Fast Marching Method (FMM) techniques achieve the solution to the Eikonal equation with a O(M log M) (or O(M) assuming some modifications), solving the more general static Hamilton-Jacobi equation requires a higher complexity. The proposed framework maintains the O(M log M) complexity for both problems, while achieving higher accuracy than available state-of-the-art. The key idea behind the proposed method is the creation of 'mini wave-fronts', where the solution is interpolated to minimize the discretization error. Experimental results show how our algorithm can outperform the state-of-the-art both in precision and computational cost.

Index Terms—Fast Marching, Eikonal equation, static Hamilton-Jacobi, Isotropic, Anisotropic.

1 Introduction

A common problem that needs to be faced in the field of computer vision is that there is an important number of applications which require defining a method for solving the optimal-trajectory problem. First attempts to solve this issue involved the popular graph-search algorithms such as A* and F*, that were employed in various applications such as road detection [1] or computing distances in maps [2]. However, this kind of algorithms have the shortcoming that they turn images into graphs (and therefore each pixel becomes a node), thus causing errors in some directions that will be invariant to the grid resolution. Cohen and Kimmel [3] demonstrated this fact and also proposed a solution to deal with the continuous optimal trajectory problem. This solution was based on the idea that the motion is governed by the static Hamilton-Jacobi partial differential equation(PDE). In particular, computer vision researchers found the Eikonal equation

$$\|u(x)\| = f(x), \quad x \in \mathcal{R}^N, \tag{1}$$

with boundary condition $u(x) = q(x), x \in \partial\mathcal{R}^N$, to be very useful to solve path planning problems. In order to obtain u (f and q are known), it is necessary to solve a first-order nonlinear PDE, whose solution tracks the motion of a monotonically advancing front. There have been in the literature some attempts to solve this equation, being the Fast Marching Method (FMM), the Fast Sweeping Method (FSM) and their variants the most stable and consistent ones [4], [5].

There are some works which presented more complex algorithms trying to solve a more general equation (called the static Hamilton-Jacobi equation) [6], [7], [8],

$$\begin{aligned} H(\nabla u(x), x) = 1, \quad x \in \mathcal{R}^N, \\ u(x) = q(x), \quad x \in \partial\mathcal{R}^N, \end{aligned} \tag{2}$$

• B. Cancela and A. Alonso-Betanzos are with the CITIC Research Centre, Universidade da Coruña, Spain, 15008.
E-mail: brais.cancela@udc.es

Manuscript received December 23, 2019; revised –.

where where q and H are assumed to be Lipschitz-continuous, and the Hamiltonian H is also assumed to be convex and homogeneous of degree one in the first argument: $H(\nabla u(x), x) = \|\nabla u\| F(\nabla u / \|\nabla u\|, x)$ for some function F. This equation introduces the concept of *anisotropic forces*, meaning that the direction of the motion is also considered. Although some works in medical imaging have started to deal with this problem, they used a simplified version [9], [10], [11], [12]. In the following, we define the Eikonal and static Hamilton-Jacobi solvers.

Eikonal Equation Solvers: The FMM introduced by Sethian [4] defines a wavefront propagation method that is consistent with the continuous case, while introducing order in the propagation causes this one-pass algorithm to maintain the classic graph search algorithm complexity, $\mathcal{O}(M \log M)$, being M the number of nodes in the model. Several approaches improve either the complexity ($\mathcal{O}(M)$ [13], [14]) or the accuracy of the model [15], [16], [17], [18], [19], [20].

FSM (Fast Sweeping Method) [5] is an iterative algorithm that can solve the Eikonal equation by removing the FMM one-pass condition. In essence, this algorithm searches for the numerical solution by alternating sweepings in predetermined directions, and at the same time it computes the solution employing a nonlinear upwinding method. As happens with FMM, accuracy can be improved when high order schemes are introduced into the finite difference upwinding scheme [21]. FMM and FSM return the same results since they are both using the same upwinding procedure. The only difference between them relies in the method they use to decide the node that needs to be updated. Hassouna et al. [19] state that FMM is advantageous because it keeps order in the selection of which point should be computed next.

Static Hamilton-Jacobi Equation Solvers: The Ordered Upwind Methods (OUM) [6] were created in order to address this problem, although algorithm complexity increases to $\Upsilon^2 \mathcal{O}(M \log M)$, being $\Upsilon = \frac{F_1}{F_2}$ the so-called *anisotropy coeffi-*

TABLE 1
2D Isotropic Analytical Functions (Closed form solution and gradient). Accuracy and Complexity, being M the number of Nodes in the model

Family	Complexity	Eikonal accuracy	Static Hamilton-Jacobi accuracy
FMM	$\mathcal{O}(M \log M)$	Very High	Very low
FSM	$\mathcal{O}(M)$	Very High	Very low
OUM	$\mathcal{O}(\Upsilon^2 M \log M)$	Low	High
WMM (Proposed)	$\mathcal{O}(M \log M)$	Very High	Very high

cient, where F_1 and F_2 are the upper and lower bounds on the speed. Mirebeau [22], [23], [24] improved the computational cost by precomputing a smaller grid, taking into account the speed direction. Recently, some researches in computer vision made use of anisotropic techniques, mainly focused in medical segmentation [9], [10], [11], [12]. They employed an anisotropic approximation method [25], [26], because, although OUM method has a better accuracy, its computational cost makes it unsuitable for this kind of problems.

Table 1 shows a summary of all the proposed methods. When dealing with Eikonal equations, both FMM and FSM techniques achieve state-of-the-art scores, but failed to provide a good solution on static Hamilton-Jacobi problems. On the contrary, the OUM family methods are able to solve the latter, but at the expenses of a higher computational complexity and a low accuracy when dealing with the Eikonal equations.

In this paper we present a novel unified algorithm, called the Wavefront Marching Method (WMM), that solves both the isotropic (Eikonal) and the anisotropic (static Hamilton-Jacobi) equations with the same computational cost ($\mathcal{O}(N \log N)$), while achieving state-of-the-art accuracy. Our idea is based on the FMM algorithm (an FSM variant can also be implemented but, as its results will only differ in the computational time, this point is beyond the scope of this paper), but introducing knowledge about the propagation orientation. To do so, our idea is to create "mini" propagation sections that are used to update the wavefront. This work is an extension of the one presented in [27], in which several relevant improvements have been added:

1) We expand our previous work from 2D to 3D cartesian grids.
2) We expand the algorithm scope to anisotropic cases, obtaining remarkable results.
3) We also provide a novel super-resolution technique, that is able to increase the algorithm's accuracy without increasing its complexity. This can be accomplished since the extra computations can be easily parallelized.
4) We provide a complete framework including all these algorithms, which can be used in two different programming languages (C++ and Python).[1]

This paper is organized as follows: section 2 describes our WMM algorithm; section 3 presents our super-resolution technique; section 4 introduces the experimental results obtained, and finally section 5 offers some conclusions.

2 Wavefront Marching Method

In order to solve the Eikonal equation, we propose a different approach called Wavefront Marching Method (WMM), where 'mini-wavefront' sections are created and combined to obtain the unique physically relevant solution.

1. Code will be avaible at: https://github.com/braisCB/WMM

Algorithm 1 Wavefront Marching Method (WMM)

Definitions:

- U: Minimal action surface.
- *Trial* set: triplets (p, S_p, r_p), consisting in the next nodes to be computed. p is the node, S_p is its associated wavefront section, and r_p is the score used to determine which set should be computed next (in which lower is better).
- *Alive* set: the points of the grid for which U is already computed.

Input:

- G: grid containing all nodes and the ∇U value.
- P_0: set containing all initial nodes.

Initialization:

- For each point in the grid, let $U_p = \infty$.
- Set the starting points $p_0 \in P_0$ to be zero:
  $U_{p_0} = 0, \forall p_0 \in P_0$, create the starting triplet $(p_0, p_0, 0)$ and put it in the *Trial* set (since there is only one point, the wavefront section is just itself).

Marching loop:

- Select (p, S_p, r_p) to be the triplet from *Trial* with the lowest value of r_p. Remove it from the set.
- For each 8-connectivity neighbour p_n that is not in the *Alive* set, compute the mini wavefront sections S_{p_n} and their quality measure r_{p_n}. Put them into the *Trial* set.

Algorithm 1 shows the skeleton of our algorithm. It works in the same way, no matter if we are dealing with 2D or 3D, or isotropic or anisotropic scenarios. It only differs in the way the wavefront segments S_p and the tentative values U_p are computed. For the sake of simplicity,

$$r_p = U_p, \tag{3}$$

that is, similar to other marching algorithms, we use the tentative value U_p to select which is the next node to be evaluated.

2.1 Creating a Wavefront Section

As it was already mentioned, our algorithm is based on the creation of small wavefront sections that are used to propagate the solution over the grid. The idea behind this schema is to create these small sections so they will be placed perpendicular to the propagation direction, simulating its front (that is why we call if wavefronts). Thus, we have defined two different wavefront approximations, depending on whether we are dealing with 2D or 3D schemes.

2.1.1 2D approach

As our system is developed to be used over Cartesian grids, a regular 8-connectivity grid is used. Formally, having a wavefront section S_p, centered at node p, we can propagate the solution to a neighbour node p_N by creating a new wavefront section S_{p_N} with segments that satisfy:

$$S_{p_N} = \bigcup_{p_R \in \mathcal{A}_p \cap \mathcal{A}_{p_N}} s_{p_N}^{p_R} = [p_N, p_R], \tag{4}$$

where $\mathcal{A}_p$ and $\mathcal{A}_{p_N}$ are the sets containing all p and p_N neighbours, respectively. To put it in different words, we take all segments in which one border belongs to the S_{p_N} center node (p_N), and the other border is at the same time a neighbour to

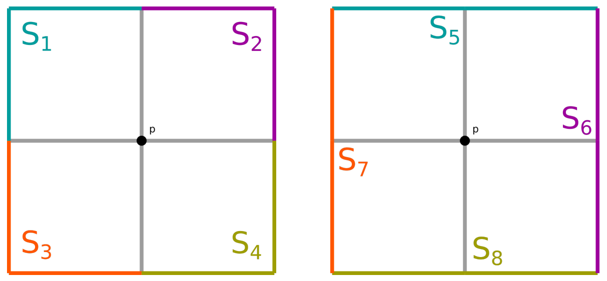


Fig. 1. 2D wavefront sections. Starting at a pixel p, and using an 8-connectivity architecture, eight different sections can be computed. On the left, corner wavefront sections. On the right, lateral sections. All wavefront centers and borders are neighbours to the propagation center p.

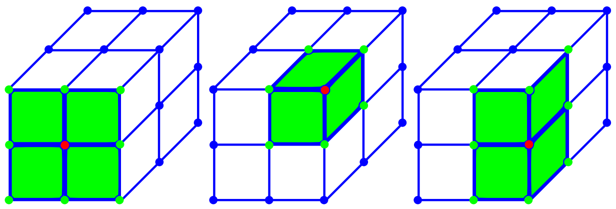


Fig. 2. 3D wavefront sections for different center points (in red). In green, wavefront planes with their respective boundary nodes. From left to right: face, corner and lateral center nodes.

both the previous and the new wavefront section center (p and p_N, respectively). Each segment is referenced by its other border ($s_{p_N}^{p_R}$).

Note that, depending on the direction of the propagation, a different wavefront can be created. Fig. 1 shows how, given a point p that is marked as *Alive*, we can create 8 different wavefront sections, each one including two segments. Assuming an 8-connectivity, we can check that the boundary nodes are always neighbors to both the wavefront center and the node from which the propagation is started.

2.1.2 3D approach

The proposed 2D WMM method can be extended to a 3D cartesian domain. The model only differs in the wavefront segment structure. Whereas in 2D we have wavefront segments, now we have wavefront planes. Formally, having the initial wavefront section S_p, the wavefront section S_{p_N} can be defined as

$$S_{p_N} = \bigcup_{p_{R_1},p_{R_2},p_{R_3} \in \mathcal{A}_p \cap \mathcal{A}_{p_N}} s_{p_N}^{p_{R_1},p_{R_2}} = [p_N, p_{R_1}, p_{R_2}, p_{R_3}], \tag{5}$$

that is, each wavefront section is defined as the union of all quadrilaterals that can be formed, taking into account that all corners must be neighbours to both p and p_N. In particular, 4 planes are defined for each neighbor center but for the corners, where we only have 3 planes, as explained in Fig. 2. Each plane is referenced by the two border nodes that are not in the diagonal ($s_{p_N}^{p_{R_1},p_{R_2}}$). The other node is a linear combination of these two ($p_{R_3} = p_{R_1} + p_{R_2} - p_N$).

2.2 Tentative Value Computation

Once the wavefront section is determined, we will define how tentative values are computed. Let $\mathcal{P}_{S_p}$ be the set of all points that are contained in the wavefront section S_p (not only the center and the border nodes, but also any point, p_t, inside the segments connecting them), and let U_{p_n} be the tentative value we aim to compute. The tentative score $U_{p_n}^{S_p}$ is calculated as

$$U_{p_n}^{S_p} = \min_{p_t \in \mathcal{P}_{S_p}} U_{p_t}^{S_p} + \|p_n - p_t\| \; f(\nabla U, S_p), \tag{6}$$

in the case of the isotropic scenario, and

$$U_{p_n}^{S_p} = \min_{p_t \in \mathcal{P}_{S_p}} U_{p_t}^{S_p} + \|p_n - p_t\| \;\; f(\nabla U, S_p, p_n), \tag{7}$$

in the anisotropic scenario. The f function will vary depending on the problem. For the sake of simplicity, only the isotropic approach will be presented in this section, as the anisotropic approach is solved exactly in the same way.

Our method is a Volume Finite approximation between the point p_n and the wavefront section S_p, using the gradient provided by ∇U. Thus, the key points of this approach are: 1) How to select the correct p_t in the wavefront section S_p; and 2) How to compute $U_{p_t}^{S_p}$ and $\nabla U_{p_t}^{S_p}$. Below we are going to answer these questions.

2.2.1 Selecting p_t

As mentioned before, each wavefront section S_p can be decomposed in smaller sections, denoted as $s_{p_N}^{p_R}$ and $s_{p_N}^{p_{R_1},p_{R_2}}$ in the 2D and 3D approaches, respectively. Thus, Eq. 6 can be decomposed in

$$U_{p_n}^{S_p} = \min_{p_T \in S_p} \left\{ \min_{p_t \in s_{p_N}^{p_T}} U_{p_t}^{S_p} + \right.$$
$$\left. \|p_n - p_t\| \frac{\|\nabla U_{p_n}\| + 2\,\|\nabla U_{(p_n+p_t)/2}\| + \|\nabla U_{p_t}\|}{4} \right\} \tag{8}$$

in the 2D approach, and

$$U_{p_n}^{S_p} = \min_{p_T, p_R \in S_p} \left\{ \min_{p_{tr} \in s_{p_N}^{p_T,p_R}} U_{p_{tr}}^{S_p} + \right.$$
$$\left. \|p_n - p_{tr}\| \frac{\|\nabla U_{p_n}\| + 2\,\|\nabla U_{(p_n+p_{tr}/2}\| + \|\nabla U_{p_{tr}}\|}{4} \right\} \tag{9}$$

in the 3D one, being p_T and p_R all border nodes of the rectangle, except the diagonal. Thus, our approach computes the tentative score in all small wavefront sections, selecting the minimum one as the correct answer. p_t is defined as

$$p_t = (1-t)\,p + t\,p_T, \qquad t \in [0,1], \tag{10}$$

in the 2D section, and

$$p_{t,r} = (1-t-r)\,p + t\,p_T + r\,p_R, \qquad t,\ r \in [0,1], \tag{11}$$

in the 3D one.

In order to select the p_t node that minimizes Eq. 8, we propose 3 different approximations:

2.2.1.1 Golden Section Search [28]:: The golden section search is a classic technique for finding an extreme point (minimum in this case) in monotonic functions. Although this is a fair assumption in the isotropic approach, it may not be the case in the anisotropic one. However, it achieves solid results. As a major drawback, this technique requires an iterative search over the segment, reducing the computational speed of the algorithm. In the case of the 3D grid, the Golden Section Search alternatively iterates over t and r variables.

In order to speed up our algorithm, we propose to use other non-iterative approaches: a modified version of the Hopf-Lax update, and the Gradient Formula.

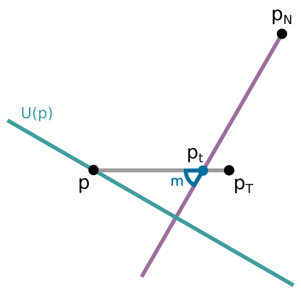


Fig. 3. Modified Hopf-Lax technique. p and p_T are the endpoints of a wavefront section, whereas p_N is the point to be updated. Note that for values $t < 0$ or $t > 1$, t must be bounded to fit within the wavefront section.

2.2.1.2 Modified Hopf-Lax Update:: A way to update U_{p_n} in the Eikonal equation is using the Hopf-Lax formula explained in [7], and extended to 3D in [29]. In our case, we are going to modify this formula to take advantage of the interpolation techniques used in our wavefront section.

2D approach: Let p_N be the node to be update, and p, p_T the endpoints of a wavefront section, as explained in Fig. 3. Having $d = p_T - p$, the t value is computed by solving the system of equations

$$p_t = p + (p_T - p)\, t, \qquad (12)$$
$$p_t = p_N + m\, r$$

where t, r are scalars, and

$$\begin{aligned} m &= R\,(p_T - p), \\ R &= \begin{pmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{pmatrix}, \qquad (13)\\ \delta = \cos\phi &= \frac{U^{S_p}_{p_T} - U^{S_p}_{p}}{\|p_T - p\|} \end{aligned}$$

In essence, the formula calculates the intersection point between the wavefront segment and the line starting in p which follows the direction of the wavefront motion ($\cos\phi$). Note that it must be narrowed, that is, $t \in [0, 1]$. Thus, the best approximation is to solve t, forcing that $t = min(1, max(0, t))$, and then solving p_t.

3D approach: The approximation is similar. Now we have to compute intersection between the line starting in p_N and the plane defined by $s^{p_T, p_R}_{p_N}$.

2.2.1.3 Gradient Formula:: To solve the Eikonal equation, classical trajectory solver algorithms using isotropic forces tend to simplify the speed of motion to its absolute value. However, if this information is provided, the direction of the speed of motion can be used to obtain the t value, as shown in Fig. 4. Thus, we have previously proposed a different approach to obtain the intersection point [27]. Similarly to the modified Hopf-Lax formula, we select the intersection point between the wavefront segment and the line starting in p which follows the direction of the speed of motion. Formally, it is solved in the same way the Hopf-Lax formula (Eq. 12), but modifying

$$m = \nabla U_{p_N}. \qquad (14)$$

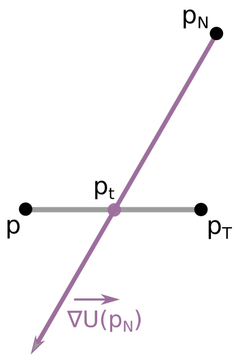


Fig. 4. Gradient technique. p and p_T are the endpoints of a wavefront section, whereas p_N is the point to be updated. p_t is obtained by computing the intersection between the gradient direction and the segment.

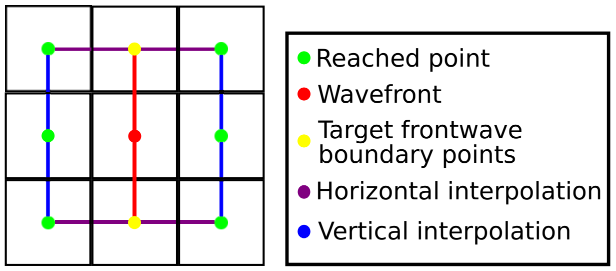


Fig. 5. Interpolation techniques during an upwinding procedure. 4 different interpolation segments are created.

2.2.2 *Computing* $U^{S_p}_{p_t}$

Our idea is to use these mini-wavefront sections to interpolate the value of both U and P over the entire wavefront section. Thus, we can increase the accuracy of the solution by introducing high-order interpolation techniques. The interpolation in 3D Cartesian grids is similar to the 2D model. Instead of a combination of segments, now we have a combination of surfaces. Depending on whether the node is a cube corner or not, a wavefront section will contain 3 or 4 mini surfaces, respectively (see Fig. 2). Thus, starting at any given node p, the solution can be propagated through 26 different direction. Each cube face contains 9 values that are used to create each wavefront surface interpolation.

We have tested our algorithm with different approximations: *linear* and *polynomial* for 2D; *bilinear* and *9-point polynomial interpolation* in 3D; and also the more complex *natural spline*, *Hermite Spline* and *Monotone Piecewise Cubic Hermite Interpolation (PCHIP)* in both dimensions. See Appendix A for more information about how to compute the coefficients.

2.2.2.1 ∇U interpolation:: As early mentioned, ∇U is also interpolated by using the same principle. However, since we are interpolating a vector and not a scalar, we interpolate ∇U in either its polar or its spherical form, depending on whether we are solving a 2D or a 3D scenario, respectively. As described in Eqs. 8 and 9, the tentative score is highly dependent on the gradient norm. Thus, in order to obtain a better solution, we selected a polar interpolation over the classic Cartesian one (see Appendix for a numerical proof). Thus, having a $\nabla U_p = (\nabla u_x, \nabla u_y)$ gradient, our polar representation is defined as

$$\nabla U^{\theta}_p = (n, \cos\rho = \nabla u_x/n, \sin\rho = \nabla u_y/n), \qquad (15)$$

where $n = \sqrt{\nabla u_x^2 + \nabla u_y^2}$ is the magnitude, and the other two parameters control the angle. If we are dealing with a 3D problem

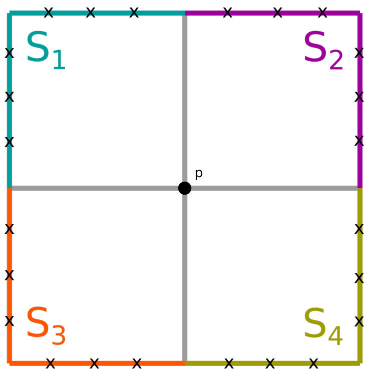


Fig. 6. 2D SR-WMM^3 sections. 3 intermediate points are created within each mini section. U value is computed over these points, but they are not used to feed the main algorithm, just to obtain a more accurate interpolation.

($\nabla U_p = (\nabla u_x, \nabla u_y, \nabla u_z)$), then

$$\nabla U_p^\theta = (n, \cos\rho, \sin\rho, \cos\phi, \sin\phi), \tag{16}$$

where $n = \sqrt{\nabla u_x^2 + \nabla u_y^2 + \nabla u_z^2}$, $\phi = \arccos \nabla u_z / n$ and $\rho = \arctan(\nabla u_y / \nabla u_z)$. Note that a polar representation can be defined by only using two parameters (three in the spherical): magnitude and angle. However, we decided to split the angle in its sine and cosine functions because they are easy to interpolate. For instance, if we interpolate over the angles $(-\pi+\epsilon, \pi-\epsilon)$, which are very close, we can interpolate the large segment $(-\pi+\epsilon, \pi-\epsilon)$ instead of the more suitable (and smaller) $(\pi-\epsilon, \pi+\epsilon)$. By using its trigonometrical functions we ensure this extreme will never happen.

We will show a toy example to illustrate the reason behind our decision. Suppose you have a 2D grid and two neighbour nodes: p_1 and p_2. Suppose that $\nabla U_{p_1} = (1,0)$ and $\nabla U_{p_2} = (0,1)$ and we are using a linear interpolation. Clearly, $\|\nabla U_{p_1}\| = \|\nabla U_{p_2}\| = 1$. However, if we interpolate the result in the middle section between p_1 and p_2, it results in $\nabla U_{p_{1.5}} = (0.5, 0.5)$, and $\|\nabla U_{p_{1.5}}\| = \sqrt{0.5} \leq \|\nabla U_{p_1}\|$.

However, if using the polar representation, we have $\nabla U_{p_1}^\theta = (1,1,0)$ and $\nabla U_{p_2}^\theta = (1,0,1)$. Thus, $\nabla U_{p_{1.5}}^\theta = (1, 0.5, 0.5) \approx (1, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$, as we have to normalize sine and cosine to have norm 1. Reverting it to cartesian coordinates, we have $\nabla U_{p_{1.5}} = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$ and $\|\nabla U_{p_{1.5}}\| = \|U_{p_1}\| = \|U_{p_2}\|$, which we believe is a more precise approach.

3 Super-Resolution WMM

As we have shown in the previous section, we need to make use of nodes outside the wavefront section to use high-order interpolation techniques. This is not a huge problem when using Cartesian grids, as all points are correctly aligned. However, this approach cannot be used when using unstructured meshes (only linear interpolation techniques can be used). In order to overcome this issue, and also aiming at increasing the precision of our algorithm without increasing its computational cost, we have developed a new version of our algorithm, called Super-Resolution WMM, or SR-WMM.

The structure of the algorithm is the same as explained in Algorithm 1, it only differs in the way wavefront sections are created. We introduce a set of equispaced 'virtual' nodes, where the solution is also computed, as depicted in Fig. 6. Formally,

$$S_{p_N}^{(i)} = \bigcup s_{p_N}^{p_R} = \left\{ p_j : p_j = \left(1 - \frac{j}{i+1}\right) p_N + \frac{j}{i+1}\, p_R, \right.$$
$$\left. j = [0, 1, \ldots, i+1] \right\}, \tag{17}$$

where i is the number of those 'virtual' nodes. This approach has several advantages:

1) It does not need a Cartesian grid, as the interpolation is performed using only the values provided by either the 'virtual' and the border nodes.
2) It does not add extra complexity to the algorithm. These nodes are only used to obtain the interpolated value, they are not used to propagate the solution. That is, these 'virtual' nodes do not exist for Algorithm 1. All virtual scores can be computed in parallel, and if and only if the wavefront centered in p_N is updated.
3) It does not add complexity to the $U_{p_t}^{S_p}$ computation. Since all nodes are equispaced, selecting which pair of nodes are surrounding p_t is straightforward.

As a minor disadvantage, polynomial interpolation techniques are not available, as they could led to high polynomial degrees which are prone to overfitting the solution.

3.1 Tentative Value Computation

In order to improve the accuracy or our model, we also made some slightly modifications to Eqs. 8 and 9. Since we are creating virtual nodes within the wavefront sections, we plan to use the same approach over the segment that joins the candidate node p_n and the wavefront segment p_t. Formally, it is defined as

$$U_{p_n}^{S_p} = \min_{p_T \in S_p} \left\{ \min_{p_t \in s_{p_N}^{p_T}} U_{p_t}^{S_p} + \right.$$
$$\left. \|p_n - p_t\| \frac{\|\nabla U_{p_n}\| + 2\sum_{j=1}^{i} \|\nabla U_{pos(p_t, p_n, i, j)}\| + \|\nabla U_{p_t}\|}{(2+1) * i} \right\} \tag{18}$$

for the 2D approach, and

$$U_{p_n}^{S_p} = \min_{p_T, p_R \in S_p} \left\{ \min_{p_{tr} \in s_{p_N}^{p_T, p_R}} U_{p_{tr}}^{S_p} + \right.$$
$$\left. \|p_n - p_{tr}\| \frac{\|\nabla U_{p_n}\| + 2\sum_{j=1}^{i} \|\nabla U_{pos(p_{tr}, p_n, i, j)}\| + \|\nabla U_{p_{tr}}\|}{(2+1) * i} \right\} \tag{19}$$

for the 3D one, where

$$pos(p, p_n, i, j) = \left(1 - \frac{j}{i+1}\right) p + \frac{j}{i+1}\, p_n, \tag{20}$$

is the virtual node between p and p_n. We perform a linear volume finite approach between these two nodes, using as many virtual nodes as we use in the wavefront segment.

TABLE 2
2D Isotropic Analytical Functions (Closed form solution and gradient).

	U	∇U
T_1	$\sqrt{x^2+y^2}$	$(x,y)/\sqrt{x^2+y^2}$
T_2	$\frac{x^2}{25}+\frac{y^2}{9}$	$\left(\frac{2x}{25},\frac{2y}{9}\right)$
T_3	$\frac{x^2}{100}+\frac{y^2}{20}$	$\left(\frac{x}{50},\frac{y}{10}\right)$
T_4	$T_1-2\sin\left(\frac{T_1}{2}\right)$	$\nabla T_1\left(1-\cos\left(\frac{T_1}{2}\right)\right)$
T_5	$T_1-8\sin\left(\frac{T_1}{8}\right)$	$\nabla T_1\left(1-\cos\left(\frac{T_1}{8}\right)\right)$

TABLE 3
3D Isotropic Analytical Functions (Closed form solution and gradient).

	U	∇U
T_6	$\sqrt{x^2+y^2+z^2}$	$(x,y,z)/\sqrt{x^2+y^2+z^2}$
T_7	$\frac{x^2}{25}+\frac{y^2}{9}+\frac{z^2}{36}$	$\left(\frac{2x}{25},\frac{2y}{9},\frac{z}{18}\right)$
T_8	$\frac{x^2}{100}+\frac{y^2}{20}+\frac{z^2}{20}$	$\left(\frac{x}{50},\frac{y}{10},\frac{z}{10}\right)$
T_9	$\frac{9}{8}T_6-2\sin\left(\frac{T_6}{2}\right)$	$\nabla T_6\left(\frac{9}{8}-\cos\left(\frac{T_6}{2}\right)\right)$
T_{10}	$\frac{9}{8}T_6-20\sin\left(\frac{T_6}{20}\right)$	$\nabla T_6\left(\frac{9}{8}-\cos\left(\frac{T_6}{20}\right)\right)$

3.1.1 Selecting p_t

As in the WMM algorithm, we have 3 different approaches to select p_t: the Gradient method, the Golden Search method and the modified Hopf-Lax formula. While the first two remain the same, as adding inner nodes do not affect their computation, we needed to modify the Hopf-Lax formula by computing it in all segments that are created by the new virtual nodes. The reason behind this decision is to obtain a more accurate wavefront motion direction. Although in this way we increase the complexity of the algorithm, this step is easily parallelized, so it does not highly affect the final performance.

4 Experimental Results

In order to show the versatility of our approach, we have conducted several experiments, including 2D and 3D meshes, using both isotropic and anisotropic approaches.

4.1 Isotropic Equations

First of all, we aim to test our algorithm against the Eikonal equation, which is the problem that is traditionally solved by methods like FMM, MSFM or FSM. In order to provide a fair experimental setup, we will conduct the same experiments performed in [30], as their results prove to be the current state-of-the-art. Ten different equations are tested, five in 2D (T_1 to T_5, see Table 2 and Fig. 7) and another five in the 3D domain (T_6 to T_{10}, see Table 3 and Fig. 8). These experiments cover a broad range of situations like unit speed propagation (T_1 and T_6), elliptical propagation (T_2, T_3, T_7 and T_8) and high gradient variation (T_4, T_5, T_9 and T_{10}). Since the closed form solution is known, we can provide an exact accuracy measure.

4.1.1 Experiments with 2D and 3D functions

The configuration is the same for all of these experiments: a grid of 101×101, $h = 1$ and $-50 \leq x, y \leq 50$, with the front propagation starting from $\Gamma_0 = (0, 0)$, in the case of the 2D solutions; a grid of $51 \times 51 \times 51$, $h = 1$ and $-25 \leq x, y, z \leq 25$. For the sake of simplicity, as the MS_c approach in [30] provides multiple configurations, we will only show its best performance in each test. The gradient values are discretized over the grid nodes. Thus, we do not assume to know the analytical gradient function. In case it is known, SR-WMM results will be way better than the ones we are showing, as the gradient values over the inner nodes will be exact scores rather than an interpolation attempt.

The results in both Table 4 and Table 5 show how our WMM algorithm is able to obtain similar scores to those obtained when using the MS$_c$ approach (or better in some cases), which is the current state-of-the-art. However, our SR-WMM method is able to achieve better results in all configurations. Our algorithm proves to be specially remarkable in 3D scenarios, as even the WMM algorithm is able to overcome the MS$_c$ method in all equations.

4.1.2 Invariance with respect to the Axis Permutation

We also aim at testing the stability of our algorithm when rotating the grid axis. To that end, we selected four different angles (21, 39, 57 and 75 degrees). Table 6 shows how the axis permutation affects the computation of T_2 equation. Although the results are remarkable, they are not as good as the ones obtained when the axis is not rotated. We believe this problem is more related with the gradient discretization rather than an algorithm issue. When the axis is not rotated, we always have the maximum gradient value located at the nodes. Thus, it is easy to perform an interpolation over it, as the maximum value is bounded over one edge. However, this condition is not satisfied when the axis is rotated. Therefore, it is more difficult to perform an interpolation so accurate.

The same experiment was performed over a 3D grid by using T_7 equation. Table 7 shows once more that our algorithm is very stable under 3D scenarios. Again, we believe the gradient discretization is the major factor in the quality of our results.

4.1.3 Invariance with respect to an Anisotropic Grid

We also tested the behavior of our algorithm when used over anisotropic grids, that is, when the h value is different depending on the grid orientation. To illustrate this, we performed the same experiment that was presented in [31]: we tested how the unit speed function (T_1 and T_6 for 2D and 3D, respectively) propagates from the center of the grid. For the 2D approach, we used a 101×101 grid and $h = (\Delta x, \Delta y) = (0.1, 0.2)$, whereas a $41 \times 41 \times 41$ grid is used in the 3D approach, where $h = (\Delta x, \Delta y, \Delta z) = (0.1, 0.1, 0.2)$. Table 8 shows how our methods achieves considerably lower errors than those obtained by classic methods like FMM or MSFM. As our method does not perform a quadratic approximation like the methods mentioned above, an anisotropic grid size does not affect its performance.

4.2 Anisotropic Equations

The previous results demonstrate how our proposed method can achieve higher accuracy than state-of-the-art methods like MS$_c$ under isotropic functions. Over the next experiments, we will also show how our method can be used to solve anisotropic cases by only changing the function $f(\nabla U, S_p, p_n)$ described in Eq. 7. To

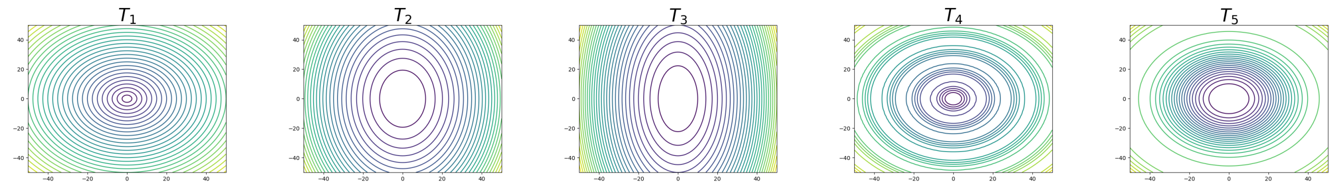


Fig. 7. 2D Analytical function surfaces.

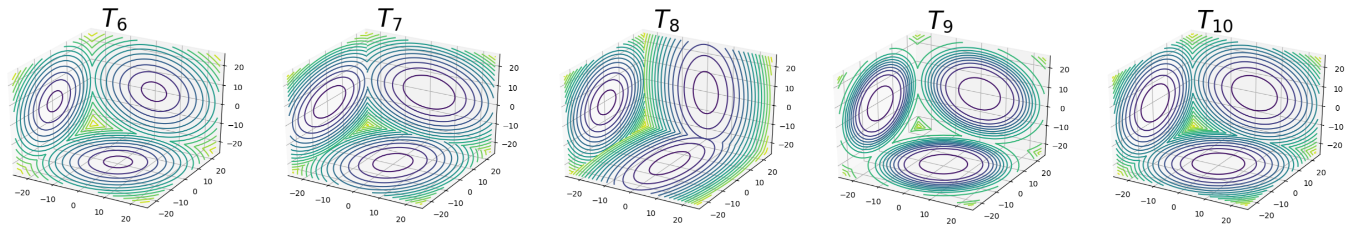


Fig. 8. 3D Analytical function surfaces.

TABLE 4
2D Isotropic Analytical Functions results. Our WMM algorithm achieves state-of-the-art results, whereas the SR-WMM obtains a better performance. Experiment configuration: $h = \Delta x = \Delta y = 1.$, $-50 \leq x, y \leq 50$, grid size $= 101 \times 101$.

	T_1		T_2		T_3		T_4		T_5	
	L_1	L_∞	L_1	L_∞	L_1	L_∞	L_1	L_∞	L_1	L_∞
MS_1 [30]	0.092	0.216	3.815	7.556	1.515	3.000	0.661	1.440	0.539	1.280
MS_2 [30]	0.228	0.307	0.090	0.253	0.022	0.097	1.318	2.521	0.259	0.712
MS_c [30]	0.005	0.016	0.009	0.049	0.003	0.023	0.079	0.131	0.020	0.031
WMM_{gr}	0.021	0.047	0.010	0.028	0.007	0.012	0.048	0.166	0.009	0.027
WMM_{hl}	0.016	0.036	1.126	5.594	0.496	2.403	0.145	0.449	0.170	0.756
WMM_{gs}	0.019	0.037	0.007	0.021	0.002	0.009	0.106	0.295	0.010	0.026
SR-WMM^3_{gr}	3e-4	8e-4	0.009	0.017	0.003	0.013	0.023	0.074	0.001	0.008
SR-WMM^3_{hl}	5e-4	0.002	0.194	1.255	0.081	0.522	0.018	0.070	0.031	0.141
SR-WMM^3_{gs}	3e-4	0.001	0.004	0.003	0.017	0.002	0.030	0.093	0.001	0.004
SR-WMM^5_{gr}	2e-4	5e-4	0.008	0.018	0.002	0.009	0.015	0.058	7e-4	0.004
SR-WMM^5_{hl}	2e-4	7e-4	0.089	0.594	0.036	0.243	0.015	0.050	0.014	0.090
SR-WMM^5_{gs}	6e-5	7e-4	0.002	0.018	0.002	0.008	0.019	0.063	8e-4	0.004

TABLE 5
3D Isotropic Analytical Functions results. Our WMM algorithm achieves state-of-the-art results, whereas the SR-WMM obtains the best performance. Experiment configuration: $h = \Delta x = \Delta y = \Delta z = 1.$, $-25 \leq x, y, z \leq 25$, grid size $= 51 \times 51 \times 51$.

	T_6		T_7		T_8		T_9		T_{10}	
	L_1	L_∞	L_1	L_∞	L_1	L_∞	L_1	L_∞	L_1	L_∞
MS_1 [30]	0.20	0.37	0.82	4.93	0.98	5.90	0.57	1.78	0.23	1.49
MS_2 [30]	0.39	0.48	0.02	0.16	0.03	0.14	0.39	0.91	0.02	0.04
MS_c [30]	0.04	0.23	0.02	0.05	0.01	0.05	0.13	0.59	0.01	0.05
WMM_{gr}	0.03	0.06	0.01	0.03	5e-3	0.03	0.07	0.27	6e-3	0.01
WMM_{hl}	0.04	0.24	1.00	5.11	0.57	2.67	0.47	1.24	0.29	0.54
WMM_{gs}	0.04	0.06	4e-3	0.21	0.01	0.33	0.06	0.45	6e-3	0.03
SR-WMM^1_{gr}	0.01	0.06	6e-3	0.04	0.01	0.03	0.04	0.26	2e-3	0.03
SR-WMM^1_{hl}	0.07	0.33	0.24	1.30	0.18	0.58	0.14	0.58	0.09	0.21
SR-WMM^1_{gs}	0.01	0.04	4e-3	0.03	5e-3	0.01	0.04	0.28	1e-3	6e-3
SR-WMM^2_{gr}	2e-3	0.07	5e-3	0.03	9e-3	0.03	0.02	0.15	7e-4	0.01
SR-WMM^2_{hl}	0.07	0.17	0.12	0.72	0.12	0.48	0.09	0.31	0.04	0.12
SR-WMM^2_{gs}	3e-3	0.01	4e-3	0.02	4e-3	0.02	0.03	0.13	6e-4	6e-3

TABLE 6
T2 error when rotating the grid axis. Experiment configuration: $h = \Delta x = \Delta y = 0.5$, $-25 \leq x, y \leq 25$, grid size $= 101 \times 101$.

	$0°$		$21°$		$39°$		$57°$		$75°$	
	L_1	L_∞	L_1	L_∞	L_1	L_∞	L_1	L_∞	L_1	L_∞
WMM$_{gr}$	0.003	0.007	0.014	0.026	0.006	0.014	0.007	0.015	0.014	0.026
WMM$_{hl}$	0.269	1.401	0.290	2.379	0.286	2.279	0.288	0.260	0.276	1.957
WMM$_{gs}$	0.002	0.005	0.008	0.019	0.003	0.011	0.005	0.014	0.010	0.026
SR-WMM$^3_{gr}$	0.002	0.004	0.015	0.023	0.008	0.014	0.011	0.016	0.007	0.025
SR-WMM$^3_{hl}$	0.046	0.313	0.047	0.231	0.034	0.157	0.036	0.139	0.041	0.260
SR-WMM$^3_{gs}$	7e-4	0.004	0.012	0.020	0.007	0.011	0.009	0.015	0.006	0.027
SR-WMM$^5_{gr}$	0.002	0.005	0.015	0.023	0.008	0.014	0.008	0.012	0.007	0.018
SR-WMM$^5_{hl}$	0.022	0.150	0.029	0.097	0.016	0.058	0.020	0.077	0.021	0.109
SR-WMM$^5_{gs}$	6e-4	0.005	0.012	0.020	0.004	0.008	0.009	0.015	0.008	0.013

TABLE 7
T7 error when rotating the grid axis. Only the best approach is shown. Experiment configuration: $h = \Delta x = \Delta y = \Delta z = .5$, $-12.5 \leq x, y, z \leq 12.5$, grid size $= 51 \times 51 \times 51$.

γ \ β	$0°$		$21°$		$39°$		$57°$		$75°$	
	L_1	L_∞	L_1	L_∞	L_1	L_∞	L_1	L_∞	L_1	L_∞
$0°$	0.01	0.03	0.01	0.03	0.01	0.04	0.01	0.03	0.01	0.04
$21°$	0.01	0.05	0.01	0.05	0.01	0.05	0.01	0.05	0.01	0.04
$39°$	0.01	0.03	0.01	0.04	0.01	0.04	0.01	0.04	0.01	0.05
$57°$	0.01	0.04	0.01	0.04	0.01	0.05	0.01	0.05	0.01	0.04
$75°$	0.01	0.05	0.01	0.06	0.01	0.05	0.01	0.05	0.01	0.04

TABLE 8
T1 and T6 error when using an anisotropic grid. T_1 experiment configuration: $h = (\Delta x, \Delta y) = (0.1, 0.2)$, grid size $= 101 \times 101$, initial point $= (51, 51)$. T_6 experiment configuration: $h = (\Delta x, \Delta y, \Delta z) = (0.1, 0.1, 0.2)$, grid size $= 41 \times 41 \times 41$, initial point $= (21, 21, 21)$.

	T_1			T_6		
	L_1	L_2	L_∞	L_1	L_2	L_∞
FMM$_1$ [31]	0.093769	0.106988	0.177274	0.693620	0.626133	1.610443
FMM$_2$ [31]	0.043759	0.049431	0.083285	0.589116	0.451326	1.374615
MS$_1$ [31]	0.046374	0.059139	0.133076	0.101452	0.012583	0.296870
MS$_2$ [31]	0.017541	0.022933	0.052733	0.024059	0.001020	0.177943
WMM$_{gr}$	0.004060	0.000034	0.016622	0.003665	0.000030	0.020650
WMM$_{hl}$	0.004310	0.000040	0.014713	0.005189	0.000046	0.025917
WMM$_{gs}$	0.004670	0.000034	0.015035	0.003390	0.000023	0.014292
SR-WMM$^3_{gr}$	0.000142	$9.08e{-}8$	0.001146	0.001776	0.000008	0.019174
SR-WMM$^3_{hl}$	0.000107	$5.27e{-}8$	0.000884	0.006164	0.000080	0.041198
SR-WMM$^3_{gs}$	0.000112	$3.71e{-}8$	0.000753	0.001508	0.000003	0.007948
SR-WMM$^5_{gr}$	0.000093	$5.17e{-}8$	0.001166	0.001277	0.000004	0.015363
SR-WMM$^5_{hl}$	0.000038	$5.34e{-}9$	0.000301	0.006747	0.000074	0.032045
SR-WMM$^5_{gs}$	0.000021	$1.58e{-}9$	0.000199	0.000535	5e-7	0.005434

that end, we will propose two anisotropic problems we aim to solve.

The difference between the isotropic and the anisotropic problem is that the direction of the propagation affects the computation. Thus, instead of $f(\nabla U)$, we will have a function $f(\nabla U, a)$, where a is a unit vector pointing into the direction of the propagation.

4.2.1 Static Hamilton-Jacobi Equations

As in [6], we prove the causality of our algorithm by computing the expansion of the ellipse, viewing it as an optimal-trajectory problem, which minimal action surface $u(x)$ is obtained by solving the static Hamilton-Jacobi-Bellman equation

$$\min_{a \in S_1}\{(\nabla u(x) \cdot a) f(x, a)\} + 1 = 0, \quad x \in \Omega, \tag{21}$$

where the speed function in the direction a is given by

$$f(a, x, y) = \sqrt{1 + (c_1 a_1 + c_2 a_2)^2}. \tag{22}$$

Translated to our nomenclature, it is equivalent to solve an anisotropic problem by using

$$f(\nabla U, S_p, p_n) = \frac{1}{\sqrt{1 + (\nabla U_{p_N} \cdot n_t)^2}}, \tag{23}$$

where $n_t = \frac{p_n - p_t}{\|p_n - p_t\|}$. As we can see in Fig. 9, the ellipse expansion computed (using $c = [1, 1]$ in the first row and $c = [\sqrt{2}, 0]$) obtains the same results both in the OUM (Ordered Upwind Methods) [6] and the WMM, meaning that our system can also operate independently of the grid orientation.

Furthermore, we decided to compare our method using classic optimal-trajectory problems. First, we use the min-time optimal-trajectory equation (Eq. 25). We consider the surface $U = 0.9 \sin(2\pi x) \sin(2\pi y)$, computing the geodesic distance from the origin. The *anisotropy coefficient* for this problem is $\Upsilon \approx 6$. We used the OUM method over a 385×385 mesh to be the true solution, since no analytical solution is available.

In Table 9, we can see the results obtained. WMM is able to obtain better results. As said above,the complexity of our

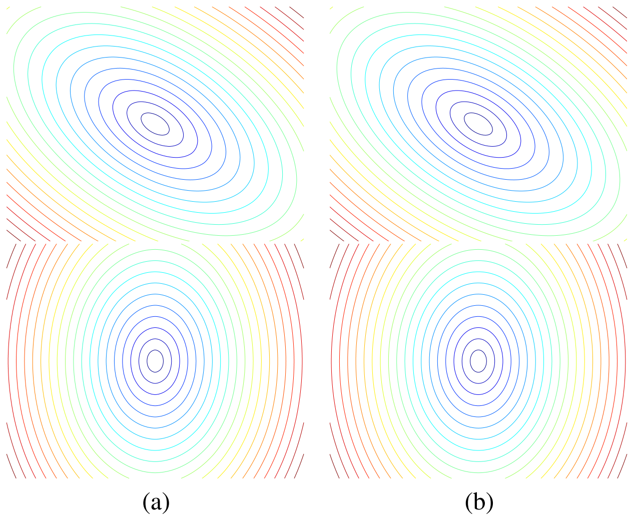


Fig. 9. Ellipse expansion over different angles. (a) OUM. (b) WMM.

algorithm is lower than the OUM one, thus its consumption of resources is always better. However, we also measured the time required to perform the computation. In order to make a fair comparison, we have not used any parallelization technique (although our algorithm is able to do that). Even so, our algorithm is much faster than OUM: our most expensive solution, using the golden search approximation and 5 inner points, matches the OUM cheapest solution.. Note that WMM error do not always decrease when increasing the grid size. We think this effect is caused by both the gradient or modified Hopf-Lax searching techniques we have developed, but this effect do not affect the causality of our approach. As the golden search approach is the most stable and accurate technique, it demonstrates how the accuracy decreases as we reduce the grid size.

In order to test the influence of the number of intermediate points, another experiment is proposed. We use the example of the first arrival travel times computation with applications to seismic imaging. Having $p_N = (y, x)$, we have four layer shapes defined by the function

$$C(x) = A\sin(\frac{m\pi x}{a} + \beta), \tag{24}$$

where A, m, a and β are constants defined over each layer. We use the anisotropic function

$$f(\nabla U, S_p, p_n) = \frac{F_2}{\sqrt{1 + (\nabla U_{p_N} \cdot n_t)^2}}, \tag{25}$$

to model the speed function, where the parameters are

$$\nabla U_{p_N} = \frac{\sqrt{\left(\frac{F_2}{F_1}\right)^2 - 1}}{\sqrt{1 + \left(\frac{\partial C}{\partial x}(x)\right)^2}} \begin{bmatrix} \frac{\partial C}{\partial x}(x) \\ -1 \end{bmatrix}, \tag{26}$$

and F_1 and F_2 are constants in each layer. In Fig. 10 we show the equi-arrival curves obtained on a 385×385 mesh using the following values:

$$a = 0.5, \quad A = 0.1225, \quad m = 2, \quad \beta = 0, \tag{27}$$

with layer offsets $b_i = (-0.25, 0, 0.25)$, the constants $F_1 = (0.2, 1, 1, 0.2)$ and $F_2 = (0.8, 3, 1, 0.8)$. The global *anisotropy coefficient* is $\Upsilon = 15$. Thus, we decided to use 15 intermediate points in a 385×385 mesh to be the true solution, and analysed the performance of our method as the mesh size and number of coefficients are decreased. Fig. 10 shows the obtained results for this test. It is showed that no significant improvement is achieved by using more than 3 intermediate points.

4.2.2 *Riemannian Metrics*

We also tested our algorithm when using Riemannian Metrics. In this case, the tentative score defined Eq. 7 is changed to

$$U_{p_n}^{S_p} = \min_{p_t \in \mathcal{P}_{S_p}} U_{p_t}^{S_p} + \|p_n - p_t\|_{\mathcal{M}_{p_n}} = U_{p_t}^{S_p} + \sqrt{(p_n - p_t)^T\, \mathcal{M}_{p_n}\, (p_n - p_t)}, \tag{28}$$

where $\mathcal{M}_{p_n}$ is a Riemannian tensor. Two different experiments are provided.

4.3 Tubular segmentation

Following [12], [23], we define the curve $\Gamma(t) = t\,(\cos(w_0 t), \sin(w_0 t)),\ \ t \in [0, 1]$ over a grid $[-1, 1] \times [-1, 1]$. We set $\mathcal{M}_{p_n} = \mathbb{I}$ for all grid points, except those where their distance against the curve is lower than r_0. In that case,

$$\mathcal{M}_{p_n} = PDP^{-1}, \quad D = \begin{bmatrix} \delta_0^2 & 0 \\ 0 & 1 \end{bmatrix}, \quad P = \begin{bmatrix} \Gamma_y(t) & -\Gamma_x(t) \\ \Gamma_x(t) & \Gamma_y(t) \end{bmatrix}, \tag{29}$$

where δ_0 is a constant variable, $\Gamma_y(t) = \cos(w_0 t) - t\ \sin(w_0 t)$ and $\Gamma_x(t) = \sin(w_0 t) + t\ \cos(w_0 t)$. The test parameters are $w_0 = 6\pi$ and $r_0 = \delta_0 = 0.01$.

First of all, we tested our algorithm when the spiral thick is just one pixel. To that end, we used a 151×151 grid to compute the result. Fig. 11 shows that, despite the WMM algorithm losses accuracy after a couple of spins, the addition of virtual nodes in the SR-WMM algorithm is able to improve the result by a large margin. Furthermore, Fig. 12 shows that, despite using low resolution grids, the minimal path between the beginning and the end of the spiral (obtained by, starting at the end point, following the maximum gradient descent direction), follows its contour all way down.

4.4 Invariance with respect to the Axis Rotation

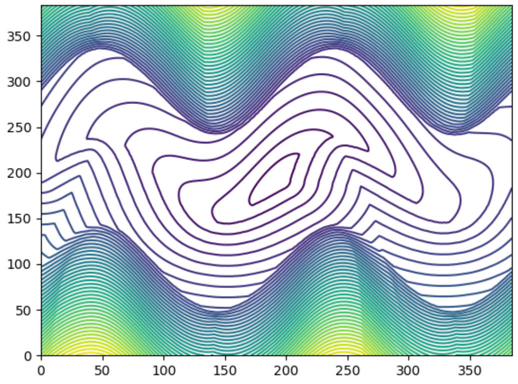
Following the experiment provided in [23], we also tested how the axis rotation affects the result on a parametric surface of height map $z(p_n) = \frac{3}{4}\ \sin(3\pi y)\ \cos(3\pi x)$, where $p_n = (y, x)$. In this case, the Riemannian metric is defined as $\mathcal{M}_{p_n} = \mathbb{I} + \nabla z(p_n) \nabla z(p_n)^T$. Since there is no true solution to this equation, we provide the result of a 4089×4089 grid to be the true result (see Fig. 13), while testing how the rotation affects the result over a 292×292 grid. Fig. 14 shows how the axis rotation affects both L_1 and L_∞ errors. The Hopf-Lax variant is proven to behave slightly unstable. However, every golden search variant is very stable against the axis rotation. Note also the errors are indeed very low, in the same way all previous experiments have demonstrated.

5 Conclusions

In this work we have presented a new method, named Wavefront Marching Method (WMM), to deal with the task of solving an optimal-trajectory problem through directional forces. Our main

TABLE 9
Error estimation on the surface $U = 0.9\sin(2\pi x)\sin(2\pi y)$, produced on refined meshes taking the corresponding method on a 385^2 mesh to be the true solution.

size / method	25^2			49^2			97^2			193^2		
	L_1	L_∞	Time(s)	L_1	L_∞	Time(s)	L_1	L_∞	Time(s)	L_1	L_∞	Time(s)
OUM$_{gr}$	0.111	0.425	0.09	0.082	0.320	0.34	0.047	0.183	1.30	0.014	0.047	4.58
OUM$_{hl}$	0.109	0.425	0.13	0.082	0.320	0.33	0.050	0.185	1.35	0.020	0.051	4.91
OUM$_{gs}$	0.108	0.425	0.36	0.084	0.32	1.21	0.050	0.187	4.85	0.015	0.055	19.22
WMM$_{gr}$	0.037	0.153	$3e-3$	0.013	0.047	$4e-3$	0.021	0.051	0.02	0.026	0.069	0.08
WMM$_{hl}$	0.053	0.231	$4e-3$	0.043	0.147	$5e-3$	0.036	0.123	0.02	0.038	0.121	0.09
WMM$_{gs}$	0.050	0.232	$8e-3$	0.017	0.096	0.02	0.004	0.022	0.07	0.005	0.018	0.29
SR-WMM$^3_{gr}$	0.038	0.142	$9e-3$	0.019	0.051	0.02	0.024	0.063	0.08	0.027	0.071	0.57
SR-WMM$^3_{hl}$	0.055	0.246	0.01	0.020	0.095	0.03	0.009	0.027	0.15	0.012	0.028	0.78
SR-WMM$^3_{gs}$	0.055	0.253	0.05	0.021	0.103	0.18	0.004	0.025	0.70	0.004	0.019	3.34
SR-WMM$^5_{gr}$	0.038	0.139	0.01	0.018	0.051	0.03	0.024	0.063	0.17	0.026	0.070	0.69
SR-WMM$^5_{hl}$	0.055	0.248	0.02	0.019	0.051	0.07	0.006	0.023	0.32	0.008	0.020	1.39
SR-WMM$^5_{gs}$	0.056	0.252	0.09	0.021	0.103	0.33	0.004	0.025	1.20	0.004	0.018	4.86



L_∞ Error	25^2	49^2	97^2	193^2
WMM	0.197	0.118	0.057	0.028
SR-WMM^3	0.173	0.082	0.037	0.013
SR-WMM^5	0.203	0.087	0.033	0.012
SR-WMM^{15}	0.208	0.086	0.038	0.014
L_1 Error	25^2	49^2	97^2	193^2
WMM	0.041	0.018	0.008	0.002
SR-WMM^3	0.038	0.015	0.007	0.002
SR-WMM^5	0.039	0.015	0.007	0.002
SR-WMM^{15}	0.040	0.015	0.007	0.002

Fig. 10. Error estimation on a seismic image, using a different number of intermediate points. A 385^2 mesh is used as the true solution, computed in the square $[-0.5, 0.5] \times [-0.5, 0.5]$. Using a number $n \approx 3$ we obtained similar results as when using higher values.

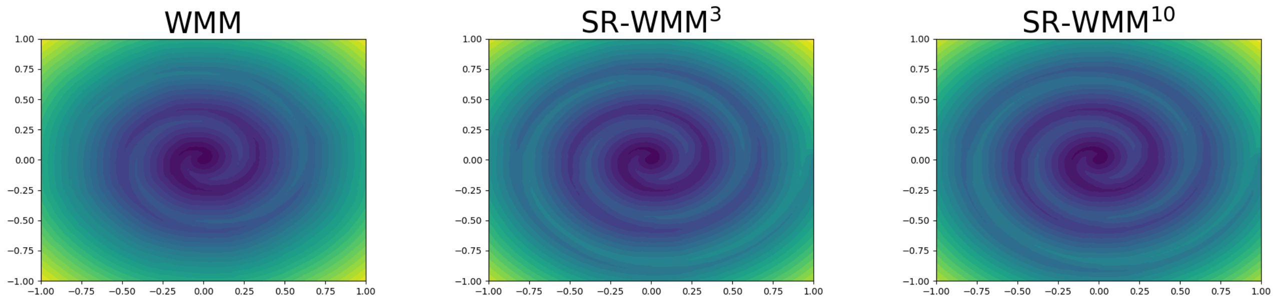


Fig. 11. Tubular segmentation surface over a 151×151 grid. Although the WMM algorithm losses the spiral after a couple of spins, the SR-WMM algorithm can successfully cope with it.

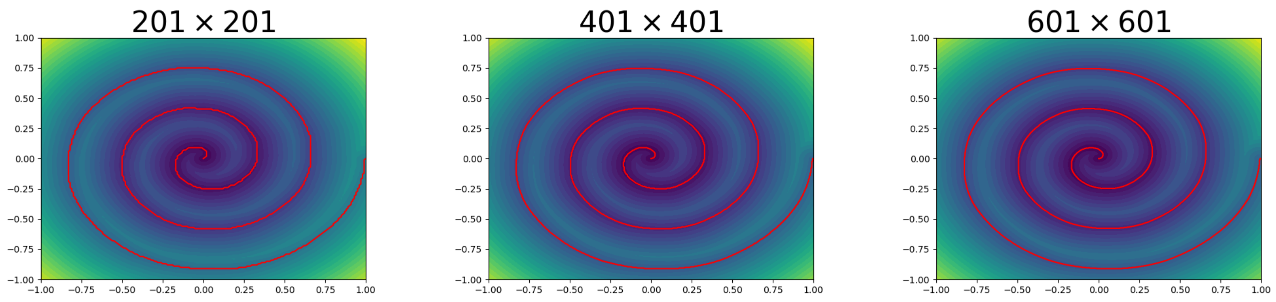


Fig. 12. Tubular segmentation surface when using different grid resolutions. Even when using small grids, our proposal is able to obtain the right answer when computing the minimal path between the spiral borders.

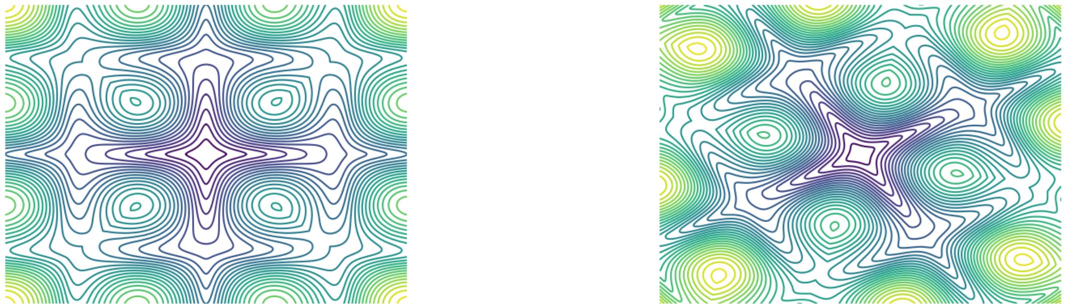


Fig. 13. True solution of the Riemannian metric defined as $\mathcal{M}_{p_n} = \mathbb{I} + \nabla z(p_n)\nabla z(p_n)^T$, where $z(p_n) = z(y,x) = \frac{3}{4}\ \sin(3\pi y)\ \cos(3\pi x)$. On the left, the solution without axis rotation. On the right, the axis is rotated bt $\frac{\pi}{6}$ radians.

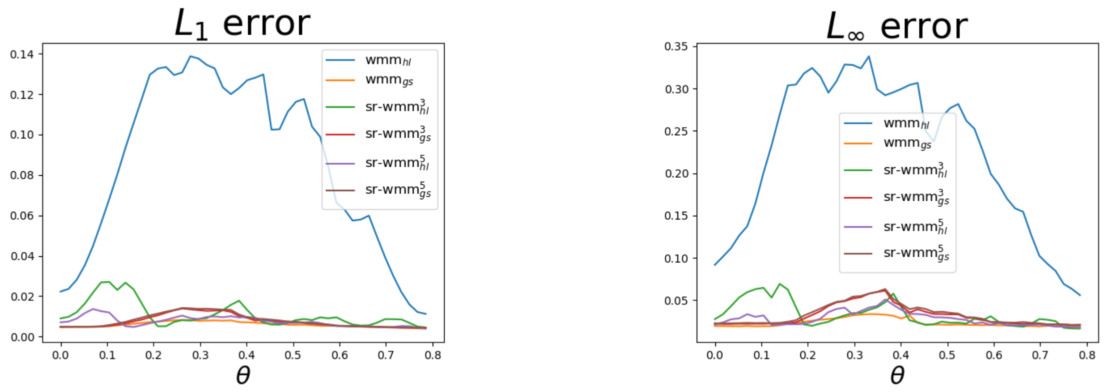


Fig. 14. L_1 and L_∞ errors when rotating the grid axis between 0 and $\frac{\pi}{4}$ radians on a parametric surface of height map $z(p_n) = \frac{3}{4}\ \sin(3\pi y)\ \cos(3\pi x)$. Although the WMM$_{hl}$ algorithm is unstable, the other variants can maintain a stable error under every angle.

idea is to propagate the solution over 'mini wavefront sections', instead of computing it over the nodes so as to reduce computational complexity. The front-propagation procedure only updates the solution over the direct neighbours, reducing the computational time to $\mathcal{O}(\Psi M \log M)$ in the average case, being Ψ the number of intermediate nodes that are used to perform the interpolation technique in the wavefront sections. Experimental results show our method works faster and is more accurate than the state of the art. We also have shown that WMM performs better using no more than 3 intermediate nodes, reducing the complexity algorithm to $\mathcal{O}(M \log M)$. This kind of techniques have many applications in computer vision domains, from which the state of the art techniques used in our comparison study are taken. As shown, our proposal obtains better results with less computational complexity.

Acknowledgments

The authors would like to thank to the financial support of the Spanish Ministerio de Economía y Competitividad (research project TIN2015-65069-C2-1-R), the Xunta de Galicia (research projects ED431C 2018/34 and Centro Singular de Investigación de Galicia, accreditation 2016-2019) and by the European Union (European Regional Development Fund). Brais Cancela acknowledges the support of the Xunta de Galicia under its postdoctoral program.

References

[1] Nicolas Merlet and Josiane Zerubia. New prospects in line detection by dynamic programming. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(4):426–431, apr 1996. 1

[2] G. Borgefors. Distance transformations in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing*, 27(3):321–345, sep 1984. 1

[3] Laurent D. Cohen and Ron Kimmel. Global minimum for active contour models: A minimal path approach. *International Journal of Computer Vision*, 24:57–78, 1997. 1

[4] J A Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States of America*, 93(4):1591–1595, 1996. 1

[5] Hongkai Zhao. A fast sweeping method for eikonal equations. *Mathematics of computation*, 74(250):603–627, 2005. 1

[6] J. A. Sethian and A. Vladimirsky. Ordered Upwind Methods for Static Hamilton–Jacobi Equations: Theory and Algorithms. *SIAM Journal on Numerical Analysis*, 41(1):325–363, 2003. 1, 8

[7] Folkmar Bornemann and Christian Rasch. Finite-element discretization of static hamilton-jacobi equations based on a local variational principle. *Computing and Visualization in Science*, 9(2):57–69, 2006. 1, 4

[8] Jean-Marie Mirebeau. Efficient fast marching with finsler metrics. *Numerische Mathematik*, pages 1–43, 2012. 1

[9] Saâd Jbabdi, Pierre Bellec, Roberto Toro, Jean Daunizeau, Mélanie Pélégrini-Issac, and Habib Benali. Accurate anisotropic fast marching for diffusion-based geodesic tractography. *Journal of Biomedical Imaging*, 2008:2, 2008. 1, 2

[10] Mickaël Péchaud, Renaud Keriven, and Gabriel Peyré. Extraction of tubular structures over an orientation domain. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 336–342. IEEE, 2009. 1, 2

[11] Ender Konukoglu, Olivier Clatz, Bjoern H Menze, Bram Stieltjes, M-A Weber, Emmanuel Mandonnet, Hervé Delingette, and Nicholas Ayache. Image guided personalization of reaction-diffusion type tumor growth models using modified anisotropic eikonal equations. *Medical Imaging, IEEE Transactions on*, 29(1):77–95, 2010. 1, 2

[12] Fethallah Benmansour and Laurent D Cohen. Tubular structure segmentation based on minimal path method and anisotropic enhancement. *International Journal of Computer Vision*, 92(2):192–210, 2011. 1, 2, 9

[13] Seongjai Kim. An o(n) level set method for eikonal equations. *SIAM journal on scientific computing*, 22(6):2178–2193, 2001. 1

[14] Liron Yatziv, Alberto Bartesaghi, and Guillermo Sapiro. O(n) implementation of the fast marching algorithm. *Journal of computational physics*, 212(2):393–399, 2006. 1

[15] James Albert Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999. 1

[16] David L Chopp. Some improvements of the fast marching method. *SIAM Journal on Scientific Computing*, 23(1):230–244, 2001. 1

[17] Per-Erik Danielsson and Qingfen Lin. A modified fast marching method. In *Image Analysis*, pages 1154–1161. Springer, 2003. 1

[18] Paul Covello and Garry Rodrigue. A generalized front marching algorithm for the solution of the eikonal equation. *Journal of computational and applied mathematics*, 156(2):371–388, 2003. 1

[19] M Sabry Hassouna and Aly A Farag. Multistencils fast marching methods: A highly accurate solution to the eikonal equation on cartesian domains. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(9):1563–1574, 2007. 1

[20] Shahnawaz Ahmed, Stanley Bak, Joyce McLaughlin, and Daniel Renzi. A third order accurate fast marching method for the eikonal equation in two dimensions. *SIAM Journal on Scientific Computing*, 33(5):2402–2420, 2011. 1

[21] Yong-Tao Zhang, Hong-Kai Zhao, and Jianliang Qian. High order fast sweeping methods for static hamilton–jacobi equations. *Journal of Scientific Computing*, 29(1):25–56, 2006. 1

[22] Jean-Marie Mirebeau. Anisotropic fast-marching on cartesian grids using lattice basis reduction. *SIAM Journal on Numerical Analysis*, 52(4):1573–1599, 2014. 2

[23] Jean-Marie Mirebeau. Anisotropic fast-marching on cartesian grids using voronoi's first reduction of quadratic forms. 2017. 2, 9

[24] Jean-Marie Mirebeau. Fast-marching methods for curvature penalized shortest paths. *Journal of Mathematical Imaging and Vision*, 60(6):784–815, 2018. 2

[25] Qingfen Lin. *Enhancement, extraction, and visualization of 3D volume data*. PhD thesis, Linköping, 2003. 2

[26] John Melonakos, Eric Pichon, Sigurd Angenent, and Allen Tannenbaum. Finsler active contours. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(3):412–423, 2008. 2

[27] Brais Cancela, Marcos Ortega, and Manuel G Penedo. A wavefront marching method for solving the eikonal equation on cartesian grids. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1832–1840, 2015. 2, 4

[28] Jack Kiefer. Sequential minimax search for a maximum. *Proceedings of the American mathematical society*, 4(3):502–506, 1953. 3

[29] Christina Stocker, Simon Vey, and Axel Voigt. Amdis-adaptive multidimensional simulations: composite finite elements and signed distance functions. *WSEAS Transactions on Circuits and Systems*, 4(3):111–116, 2005. 4

[30] S. Merino-Caviedes, L. Cordero-Grande, M. T. Pérez, P. Casaseca-de-la-Higuera, M. Martín-Fernández, R. Deriche, and C. Alberola-López. A second order multi-stencil fast marching method with a non-constant local cost model. *IEEE Transactions on Image Processing*, 28(4):1967–1979, 2019. 6, 7, 15, 16

[31] M Sabry Hassouna and Aly A Farag. Multistencils fast marching methods: A highly accurate solution to the eikonal equation on cartesian domains. *IEEE transactions on pattern analysis and machine intelligence*, 29(9):1563–1574, 2007. 6, 8

[32] Frederick N Fritsch and Ralph E Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980. 13

[33] Brais Cancela, Alberto Iglesias, Marcos Ortega, and Manuel G Penedo. Unsupervised trajectory modelling using temporal information via minimal paths. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2553–2560, 2014. 16

Brais Cancela received the Ph.D. degree for his work in the area of computer vision in 2015 at the University of A Coruña (Spain). He is currently a Postdoctoral Researcher in the Department of Computer Science, University of A Coruña (Spain). His main current areas are convex optimization and machine learning.

Amparo Alonso-Betanzos received the Ph.D. degree for her work in the area of medical expert systems in 1988 at the University of Santiago de Compostela (Spain). Later, she was a postdoctoral fellow in the Medical College of Georgia, Augusta (USA). She is currently a Full Professor in the Department of Computer Science, University of A Coruña (Spain). Her main current areas are intelligent systems, scalable machine learning and feature selection.

Appendix A Interpolation techniques

Below we will show a detailed description about how to compute the interpolation coefficients for each technique.

A.1 2D interpolation

Since we are using a regular grid, we have four different segments with 3 values each, as exposed in Fig 5. Thus, 4 different interpolation segments are defined. Four different interpolation techniques were developed to compute U_{p_t}. The computation of the $\nabla U(p_t)$ value is analogous.

Linear Interpolation: Suppose we have one segment of a wavefront section S_p, defined by the center node p and a border node p_R. $U_{p_t}^{S_p}$ value computation is defined as

$$U_{p_t}^{S_p} = (1-t)\,U_p \;+\; t\,U_{p_R}, \tag{30}$$

independently of whether we are dealing or not with a corner wavefront.

Polynomial Interpolation: $U_{p_t}^{S_p}$ value computation value is defined as

$$U_{p_t}^{S_p} = a^{S_p}\,t^2 \;+\; b^{S_p}\,t \;+\; c^{S_p}, \tag{31}$$

where $a_{p_t}^{S_p}$, $b_{p_t}^{S_p}$ and $c_{p_t}^{S_p}$ are the polynomial coefficients. If the wavefront is a corner,

$$\begin{aligned} a^{S_p} &= \frac{U_{p_T} + U_p - 2U_{p_R}}{2}, \\ b^{S_p} &= U_{p_R} - a^{S_p} - U_p, \\ c_{S_p} &= U_p \end{aligned} \tag{32}$$

where $p_T = 2p_R - p$. Note that this node point does not belong to the wavefront section, but it is required to perform the interpolation. On the contrary, if the wavefront is not a corner,

$$\begin{aligned} a^{S_p} &= \frac{U_{p_U} + U_{p_R} - 2U_p}{2}, \\ b^{S_p} &= U_p - U_{p_U} + a^{S_p}, \\ c_{S_p} &= U_p, \end{aligned} \tag{33}$$

where $p_U = 2p - p_R$. It is worth mentioning that, when creating the wavefront section S_p by propagation from any other wavefront section S_{p_S}, we do not need to perform extra computations for either p_T and p_U nodes, as these points are also neighbours of the wavefront center node p_S, so U_{p_T} and thus U_{p_U} have to be computed before the wavefront creation, in order to know if our solution needs to be propagated through these points. If any of these nodes fits outside the node grid, then we extend the grid size, filling these values by mirroring.

Natural Spline Interpolation: A cubic approximation can be achieved using piecewise interpolation. To do so, we propose to use a natural spline. Unfortunately, the problem of the spline interpolation is that monotonicity is not preserved, causing the interpolation to have lower values than its endpoints. Negative values can also be achieved. Thus, we have to check if the result is higher than any of its endpoints. If not, linear interpolation is used to compute the U_{p_t} value. Formally, $U_{p_t}^{S_p}$ value computation value is defined as

$$U_{p_t}^{S_p} = U_p + t\left(-\frac{b^{S_p}}{6} - \frac{a^{S_p}}{3} \;+\; U_{p_R} \;-\; U_p + t\left(\frac{a^{S_p}}{2} + t\left(b^{S_p} - \frac{a^{S_p}}{6}\right)\right)\right), \tag{34}$$

where

$$\begin{aligned} a^{S_p} &= 0, \\ b^{S_p} &= \frac{3}{2}(U_p - 2\,U_{p_R} + U_{p_T}), \end{aligned} \tag{35}$$

if we are dealing with a wavefront corner section, and alternatively

$$\begin{aligned} a^{S_p} &= \frac{3}{2}(U_{p_U} - 2\,U_p + U_{p_R}), \\ b^{S_p} &= 0 \end{aligned} \tag{36}$$

if not.

Hermite Spline Interpolation: Also another cubic approximation, its value can be computed as

$$U_{p_t}^{S_p} = (2\,t^3 - 3t^2 + 1)\,U_p + (t^3 - 2\,t^2 + t)\,a^{S_p} + (-2\,t^3 + 3\,t^2)\,U_{p_R} + (t^3 - t^2)\,b^{S_p}, \tag{37}$$

where

$$\begin{aligned} a^{S_p} &= U_{p_R} - U_p, \\ b^{S_p} &= \frac{U_{p_T} - U_p}{2}, \end{aligned} \tag{38}$$

if we are dealing with a wavefront corner section, and

$$\begin{aligned} a^{S_p} &= \frac{U_{p_R} - U_{p_U}}{2}, \\ b^{S_p} &= U_{p_R} - U_{p_U} \end{aligned} \tag{39}$$

if not.

Monotone Piecewise Cubic Hermite Interpolation (PCHIP): As mentioned before, monotonicity is not preserved using spline interpolation. A different way to solve this issue is the use of a Monotone Piecewise Cubic Hermite Interpolation [32]. The model is similar to the classic Cubic Hermite Interpolation. $U_{p_t}^{S_p}$ is also computed by using Eq. 37. It only differs in how a^{S_p} and b^{S_p} coefficients are computed. If we are dealing with a wavefront corner section

$$a^{S_p} = \begin{cases} 0 & \text{if } \dfrac{3\,\Delta_0 - \Delta_1}{2}\Delta_0 \leq 0 \\ 3\,\Delta_0 & \text{if } \Delta_1\,\Delta_0 \leq 0 \text{ and } \left|\dfrac{3\,\Delta_0 - \Delta_1}{2}\right| > \left|3\,\Delta_0\right| \\ \dfrac{3\,\Delta_0 - \Delta_1}{2} & \text{otherwise} \end{cases} \tag{40}$$

$$b^{S_p} = \begin{cases} 0 & \text{if } \Delta_0\,\Delta_1 \leq 0 \\ \dfrac{2\,\Delta_0\,\Delta_1}{\Delta_0 + \Delta_1} & \text{otherwise} \end{cases} \tag{41}$$

where $\Delta_0 = U_{p_R} - U_p$ and $\Delta_1 = U_{p_T} - U_{p_R}$. If the wavefront is not a corner,

$$a^{S_p} = \begin{cases} 0 & \text{if } \Delta_0' \, \Delta_1' \leq 0 \\ \dfrac{2\,\Delta_0'\,\Delta_1'}{\Delta_0' + \Delta_1'} & \text{otherwise} \end{cases} \tag{42}$$

$$b^{S_p} = \begin{cases} 0 & \text{if } \dfrac{3\,\Delta_1' - \Delta_0'}{2} \Delta_1' \leq 0 \\ 3\,\Delta_1' & \text{if } \Delta_1'\,\Delta_0' \leq 0 \text{ and } \left|\dfrac{3\,\Delta_1' - \Delta_0'}{2}\right| > \left|3\,\Delta_1'\right| \\ \dfrac{3\,\Delta_1' - \Delta_0'}{2} & \text{otherwise} \end{cases} \tag{43}$$

where $\Delta_0' = U_p - U_{p_U}$ and $\Delta_1' = U_{p_R} - U_p$.

A.2 3D interpolation

Five different interpolation techniques were developed to compute $U_{p_{tr}}$. Again, $\nabla U(p_{tr})$ value computation is analogue.

Bilinear Interpolation: Suppose we have one segment of a wavefront section $S_{p'}$, defined by the center node p and three border nodes (p_T and p_R on its laterals, and p_{TR} on its diagonal), that was created by starting from another wavefront section S_p. $U_{p_{tr}}^{S_p}$ value computation is defined as

$$U_{p_{tr}}^{S_p} = (1-t)\,(1-r)\,U_p \;+\; t\,(1-r)\,U_{p_T} \;+ (1-t)\,r\,U_{p_R} \;+\; t\,r\,U_{p_{TR}}. \tag{44}$$

9-point Polynomial Interpolation: A little bit more complex than the previous one, $U_{p_{tr}}^{S_p}$ value computation is defined as

$$U_{p_{tr}}^{S_p} = a^{S_p}\,t^2 \;+\; b^{S_p}\,t \;+\; c^{S_p}\,r^2 \;+\; d^{S_p}\,r \;+\; e^{S_p}\,t^2\,r^2 \;+ f^{S_p}\,t^2\,r \;+\; g^{S_p}\,t\,r^2 \;+\; h^{S_p}\,t\,r \;+\; i^{S_p}. \tag{45}$$

All coefficients are defined as

$$\begin{aligned} a^{S_p} &= \frac{U_{p_T'} - (1-\Delta_T)\,U_p - \Delta_T\,U_{p_T}}{\Delta_T\,(1-\Delta_T)}, \\ b^{S_p} &= U_{p_T} - a^{S_p} - U_p, \\ c_{S_p} &= \frac{U_{p_R'} - (1-\Delta_R)\,U_p - \Delta_R\,U_{p_R}}{\Delta_R\,(1-\Delta_R)}, \\ d_{S_p} &= U_{p_R} - c^{S_p} - U_p, \\ e_{S_p} &= \frac{\Delta_T\Delta_R\,U_1 - \Delta_R\,U_2 - \Delta_T\,U_3 + U_4}{\Delta_T\Delta_R\,(1-\Delta_R-\Delta_T+\Delta_T\Delta_R)}, \\ f_{S_p} &= \frac{\Delta_T\,U_3 - U_4 - (\Delta_T\Delta_R^2\,(1-\Delta_T))\,e_{S_p}}{\Delta_T\Delta_R\,(1-\Delta_T)}, \\ g_{S_p} &= \frac{\Delta_R\,U_2 - U_4 - (\Delta_T^2\Delta_R\,(1-\Delta_R))\,e_{S_p}}{\Delta_T\Delta_R\,(1-\Delta_R)}, \\ h_{S_p} &= \frac{U_4 - \Delta_T^2\Delta_R^2\,e_{S_p} - \Delta_T\Delta_R^2\,g_{S_p} - \Delta_T^2\Delta_R\,f_{S_p}}{\Delta_T\Delta_R}, \\ i_{S_p} &= U_p \end{aligned} \tag{46}$$

where

$$\begin{aligned} p_T' &= \begin{cases} 2p_T - p & \text{if } 2p_T - p \text{ is neighbour of } p' \\ 2p - p_T & \text{otherwise} \end{cases} \\ p_R' &= \begin{cases} 2p_R - p & \text{if } 2p_R - p \text{ is neighbour of } p' \\ 2p - p_R & \text{otherwise} \end{cases} \\ \Delta_T &= \begin{cases} 2 & \text{if } 2p_T - p \text{ is neighbour of } p' \\ -1 & \text{otherwise} \end{cases} \\ \Delta_R &= \begin{cases} 2 & \text{if } 2p_R - p \text{ is neighbour of } p' \\ -1 & \text{otherwise} \end{cases} \\ U_1 &= U_{p_{TR}} - U_p - a^{S_p} - b^{S_p} - c^{S_p} - d^{S_p}, \\ U_2 &= U_{p_R+(\Delta_T,0)} - U_p - \Delta_T^2\,a^{S_p} - \Delta_T\,b^{S_p} - c^{S_p} - d^{S_p}, \\ U_3 &= U_{p_T+(0,\Delta_R)} - U_p - a^{S_p} - b^{S_p} - \Delta_R^2\,c^{S_p} - \Delta_R\,d^{S_p}, \\ U_4 &= U_{p+(\Delta_T,\Delta_R)} - U_p - \Delta_T^2\,a^{S_p} - \Delta_T\,b^{S_p} - \Delta_R^2\,c^{S_p} - \\ &\quad \Delta_R\,d^{S_p} \end{aligned} \tag{47}$$

Natural Spline, Hermite Spline and PCHIP: 2 consecutive 1D spline interpolations are used to perform a bicubic spline interpolation. That is,

$$U_{p_{tr}}^{S_p} = \text{spline}(x = [0, 1, \Delta_R], y = [U_1, U_2, U_3])(t), \tag{48}$$

where

$$\begin{aligned} U_1 &= \text{spline}(x = [0, 1, \Delta_T], y = [U_p, U_{p_T}, U_{p+(\Delta_T,0)}])(t), \\ U_2 &= \text{spline}(x = [0, 1, \Delta_T], y = [U_{p_R}, U_{p_{TR}}, U_{p_R+(\Delta_T,0)}])(t), \\ U_3 &= \text{spline}(x = [0, 1, \Delta_T], \\ &\qquad y = [U_{p+(0,\Delta_R)}, U_{p_T+(0,\Delta_R)}, U_{p+(\Delta_T,\Delta_R)}])(t) \end{aligned}$$

The spline 1D coefficients are computed as shown in the 2D interpolation section. In order to improve the accuracy, and also to keep consistency in the results, we perform a two-pass approximation: creating the first spline by using t and then using r to compute $U_{p_{tr}}^{S_p}$ (as shown in Eq. 48), and back. Thus, the tentative value $U_{p_{tr}}^{S_p}$ will be given as the mean between these two scores.

APPENDIX B
CARTESIAN VS POLAR INTERPOLATION

As Eqs. 8 and 9 show, the tentative score in the Eikonal equation requires to compute the gradient norm. Thus, in order to achieve a good accuracy, it is mandatory to have a good gradient norm interpolation. As exposed in section 2.2.2.1, the use of the polar coordinates (or spherical in 3D) helps us with this matter. Besides the theoretical motivation, we also wanted to show that using Cartesian coordinates will result in a significant performance drop. To that end, we create a vanilla experiment by replicating the one presented in Table 4, employing both Polar and Cartesian coordinates, but only using the linear interpolation (same results are achieved with other interpolation techniques). Table 10 shows how the Polar coordinates achieve the best accuracy in every configuration. On those cases where the solution is similar we can confirm almost all tentative scores choose $t \in \{0, 1\}$, making the interpolation approach unnecessary.

TABLE 10
2D Isotropic Analytical Functions results when using polar or cartesian approach for interpolation. Polar approach achieves the best scores amongst all different configurations.

	Interpolation	T_1		T_2		T_3		T_4		T_5	
		L_1	L_∞	L_1	L_∞	L_1	L_∞	L_1	L_∞	L_1	L_∞
WMM$_{gr}$	Polar	0.021	0.047	0.010	0.028	0.007	0.012	0.048	0.166	0.009	0.027
WMM$_{hl}$		0.016	0.036	1.126	5.594	0.496	2.403	0.145	0.449	0.170	0.756
WMM$_{gs}$		0.019	0.037	0.007	0.021	0.002	0.009	0.106	0.295	0.010	0.026
WMM$_{gr}$	Cartesian	0.130	0.281	0.291	0.588	0.090	0.180	0.048	0.166	0.070	0.199
WMM$_{hl}$		0.116	0.273	1.269	5.647	0.551	2.492	0.144	0.449	0.223	0.848
WMM$_{gs}$		0.118	0.276	0.276	0.576	0.085	0.168	0.106	0.295	0.061	0.196

TABLE 11
2D Isotropic Analytical Functions results. Our WMM algorithm achieves state-of-the-art results, whereas the SR-WMM obtains a better performance. Experiment configuration: $h = \Delta x = \Delta y = 1.$, $-50 \leq x, y \leq 50$, grid size $= 101 \times 101$.

	T_1		T_2		T_3		T_4		T_5	
	L_1	L_∞	L_1	L_∞	L_1	L_∞	L_1	L_∞	L_1	L_∞
FMM$^3_1$ [30]	0.324	0.561	2.107	4.999	0.838	1.985	0.298	0.911	0.290	0.687
FMM$^3_2$ [30]	0.081	0.123	1.034	3.681	0.430	1.657	0.390	1.062	0.210	0.502
MSFM$^3_1$ [30]	0.135	0.270	2.107	4.999	0.838	1.985	0.381	1.332	0.300	0.598
MSFM$^3_1$ [30]	0.060	0.116	0.836	2.466	0.331	0.972	1.303	2.811	0.336	0.800
FMM$^5_1$ [30]	0.217	0.372	1.762	4.490	0.701	1.783	0.267	0.840	0.260	0.668
FMM$^5_2$ [30]	0.048	0.069	1.096	3.307	0.446	1.488	0.274	0.853	0.227	0.461
MSFM$^5_1$ [30]	0.092	0.185	1.762	4.490	0.701	1.783	0.303	1.053	0.277	0.609
MSFM$^5_1$ [30]	0.035	0.058	0.993	2.967	0.394	1.176	0.787	1.937	0.295	0.685
SR-WMM$^3_{gr}$	3e-4	8e-4	0.009	0.017	0.003	0.013	0.023	0.074	0.001	0.008
SR-WMM$^3_{hl}$	5e-4	0.002	0.194	1.255	0.081	0.522	0.018	0.070	0.031	0.141
SR-WMM$^3_{gs}$	3e-4	0.001	0.004	0.003	0.017	0.002	0.030	0.093	0.001	0.004
SR-WMM$^5_{gr}$	2e-4	5e-4	0.008	0.018	0.002	0.009	0.015	0.058	7e-4	0.004
SR-WMM$^5_{hl}$	2e-4	7e-4	0.089	0.594	0.036	0.243	0.015	0.050	0.014	0.090
SR-WMM$^5_{gs}$	6e-5	7e-4	0.002	0.018	0.002	0.008	0.019	0.063	8e-4	0.004

Appendix C
About Super Resolution

In this paper we tested our super-resolution algorithm (SR-WMM) against the classic fast marching techniques, as we wanted to test our methodology against techniques that have similar complexity. However, it is not possible to create a similar super-resolution technique by using classic Fast Marching Method approaches, as they need a structured mesh to perform that computations. Nevertheless, we can create a naive approximation by resizing the initial grid, and interpolating the gradient in the inner nodes. Note that the computational cost of this approach is way higher than our proposed SR-WMM algorithm: if we imagine a grid with only four nodes, creating a rectangle, our algorithm only creates virtual nodes over the edges, while the naive approach above computes virtual nodes over the whole surface. Furthermore, while our virtual nodes are invisible for Algorithm 1, they have to be taken into account in the vanilla version. Thus, being M the grid size, and γ the number of inner nodes, the complexity of these vanilla approaches scales up to $\mathcal{O}(\gamma M \log \gamma M)$, while our SR-WMM algorithm complexity remains at $\mathcal{O}(M \log M)$.

Thus, we created a vanilla experiment by replicating the one presented in Table 4, using a super-resolution grid for the FMM and MSFM algorithms. Table 11 shows the results obtained, suggesting that, although FMM and MSFM can increase its accuracy, the margin is not high enough to compete with our proposed algorithm.

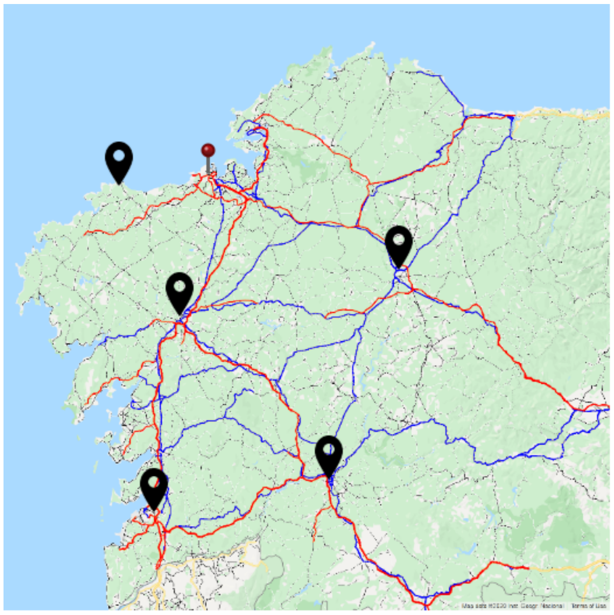


Fig. 15. Galicia's main roads (red: highways; blue: main routes; black: local routes). The red pin sets the initial points, whereas the black pins show five different destinations.

Appendix D
An application to Path Planning

Finally, we aimed at testing our algorithm against a real problem: path planning. The objective is to obtain the fastest route from any given initial point to a destination point. To do so, we have selected six different cities from Galicia, a small region in the north-west

TABLE 12
Path planning results. Our proposed WMM$_{gs}$ algorithm achieves the best result for all image sizes tested.

image size	(3000, 3000)		(1500, 1500)		(1000, 1000)		(750, 750)	
error (km)	L_1	L_∞	L_1	L_∞	L_1	L_∞	L_1	L_∞
FMM$_1$ [30]	4.207	5.185	8.356	12.472	51.457	121.258	64.314	133.243
FMM$_2$ [30]	2.038	3.437	6.667	9.232	51.005	120.494	64.184	132.982
MSFM$_1$ [30]	1.599	2.222	2.910	3.730	4.418	5.821	5.162	7.447
MSFM$_1$ [30]	0.729	1.005	1.968	3.157	3.890	5.266	5.428	7.367
WMM$_{hl}$	2.408	3.130	2.892	3.771	4.140	4.817	3.444	5.003
WMM$_{gs}$	0.581	1.096	0.672	1.419	1.950	3.353	1.666	3.396

of Spain, creating a map containing all routes (see Fig. 15. As the main purpose is to obtain the fastest route, the speed function f will be defined in terms of speed and distance:

$$f_{time}(x) = \frac{\text{distance}}{\text{speed}} = \begin{cases} \alpha/120 & \text{for highways} \\ \alpha/80 & \text{for main roads} \\ \alpha/50 & \text{for local roads,} \end{cases} \tag{49}$$

where α are the distance between grid nodes (in kilometers)

Thus, as the f function uses both distance and speed, the value obtained in U_{time} is time, that is, the time needed to go from the initial to any of the possible destination points. Unfortunately, time is not an exact measure that can be used to compare the performance of our algorithms. For that reason, in a similar way as presented in [33], we simultaneously propagate the solution over two different functions: f_{time} to compute the arrival time, and

$$f_{distance}(x) = \alpha \tag{50}$$

to compute the arrival distance, that is, the exact distance (in kilometers) between the initial position to any other node in the map. This simultaneous propagation is done by using a master-slave approach: f_{time} is used to seek for the correct propagation of motion, whereas $f_{distance}$ is propagated only by the direction provided by f_{time}. In our WMM algorithm, this means that both U_{time} and $U_{distance}$ solutions are propagated by using the same p_t value, and this value is chosen by performing a tentative value computation over U_{time}.

Therefore, as the distances between the initial city and the rest of destination cities are well-known and can be exactly computed, in Table 12 we compared both our WMM algorithms against the classic FMM and MSFM algorithms, using four different image sizes. The results obtained show that our algorithm obtains the best results, independently of the image size.