A Wavefront Marching Method for Solving the Eikonal Equation on Cartesian Grids

B. Cancela, M. Ortega, M. G. Penedo
VARPA Group, Universidade da Coruña
{brais.cancela, mortega, mgpenedo}@udc.es

Abstract

This paper presents a new wavefront propagation method for dealing with the classic Eikonal equation. While classic Dijkstra-like graph-based techniques achieve the solution in $\mathcal{O}(M \log M)$, they do not approximate the unique physically relevant solution very well. Fast Marching Methods (FMM) were created to efficiently solve the continuous problem. The proposed approximation tries to maintain the complexity, in order to make the algorithm useful in a wide range of contexts. The key idea behind our method is the creation of 'mini wave-fronts', which are combined to propagate the solution. Experimental results show the improvement in the accuracy with respect to the state of the art, while the average computational speed is maintained in $\mathcal{O}(M \log M)$, similar to the FMM techniques.

1. Introduction

In computer vision, a large number of applications require the definition of a method for solving the optimal-trajectory problem. In early stages, graph-search algorithms were used to solve the issue. For instance, A* and F* algorithms were applied to compute distance maps [4] or road detection [17]. Both methods suffer from 'metrication errors', since these algorithms consider the image as a graph, where each pixel is a node. As Cohen and Kimmel [8] clearly demonstrate, these methods always cause an error in some direction, that will be invariant to the grid resolution.

To overcome this issue, Cohen and Kimmel [8] proposed to deal with the continuous optimal trajectory problem, whose motion is governed by a partial differential equation known as the Hamilton-Jacobi equation. In computer vision, when dealing with path planning, a Hamilton-Jacobi formula called the Eikonal equation is found to be very useful. It is a first-order nonlinear PDE whose solution tracks the motion of a monotonically advancing front. Several methods have been proposed to solve this equation, being the Fast Marching Method (FMM), the Fast Sweeping Method (FSM) and their variants the most stable and consistent ones.

More complex algorithms can solve a more general equation, the static Hamilton-Jacobi equation. That is the case of the Ordered Upwind Method [22], or, more recently, by the works of Bornemann *et al*. [5] and Mirebeau [18]. This equation introduces the concept of *anisotropic forces*, where the direction of the motion is taken into account. Although recent approaches in medical imaging deals with this problem [14, 19, 16, 3], they deal with a simplified version.

Eikonal Equation Solvers: The FMM introduced by Sethian [20] 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 ($\mathcal{O}(M)$ in [15, 23]) or the accuracy of the model [21, 7, 10, 9, 13, 1].

The Fast Sweeping Method (FSM) [25] was also introduced to solve the Eikonal equation. It is an iterative algorithm that removes the FMM one-pass condition. The algorithm finds the numerical solution by alternating sweepings in predetermined directions, while computing the solution using a nonlinear upwinding method. Similar to the FMM, it is possible to increase its accuracy by introducing high order schemes into the finite difference upwinding scheme [24].

Both FMM and FSM solve the Eikonal equation using the same upwinding procedure, providing the same results. They only differ in the technique used to decide which node should be updated. According to [13], the main advantage of using the FMM over existing techniques is that the technique keeps order in the selection of which point should be computed next.

In this paper we present a new front propagation method to solve the optimal-trajectory problem in cartesian grids. Based in the idea of front-propagation techniques, we

develop an algorithm, the Wavefront Marching Method (WMM), that is able to increase the accuracy of the solution, while the computational complexity is reduced to $\mathcal{O}(M \log M)$ in the average case. To do that, we take advantage of the direction of the speed motion, instead of using a pure isotropic algorithm. Our idea is to create "mini" propagation sections that are used to update the wavefront.

This paper is organized as follows: section 2 explains the fast marching methods, showing its limitations; section 3 describes our WMM methodology; finally, section 4 introduces experimental results and section 5 offers some conclusions.

2. Fast Marching Method

The Eikonal equation is a problem defined by the equation

$$|\nabla T| = F, \quad T(\Gamma_0) = 0, \tag{1}$$

This problem describes the evolution of a curve as a function of time (T), with speed F normal to any given point in the space. p_0 is the initial position, where the propagation starts. The speed of motion has no orientation information, that is, it is an *isotropic* scenario. To introduce order in the selection of the grid points during the computation of the solution, the algorithm takes into account the fact that the time T at any point in the grid only depends on its neighbors with smaller values. To do that, each grid point is tagged with 3 possible labels, as explained in Algorithm 1. Graphically, the *Trial* set contains all the nodes in the grid that have, at least, one neighbour tagged as *Alive*, which are the nodes that already have its minimum value. Finally, the *Far* set contains all the nodes that are not reached yet by using the upwinding procedure.

2.1. Multistencils Fast Marching Method

The two key points in this algorithm are the selection of the neighborhood and the $T_{i,j}$ computation. Several methods were proposed to improve the FMM. In this section we are going to explain one of the most accurate, the Multistencils Fast Marching Method (MSFM) [13]. Whereas classical FMM uses a 4-connected neighbors, MSFM also includes the diagonal ones, increasing the number of neighbors to 8.

The computation of the T value is divided in two different equations, depending if the stencil is aligned with the natural coordinate system (that is, horizontal and vertical), called $\mathbf{S_1}$ or if it is aligned with the diagonal neighbors ($\mathbf{S_2}$). Moreover, two different finite difference schemes can be used, first or second.

$\mathbf{S_1}$ Stencil Computation: Assuming we have a regular grid, the first-order equation of this stencil aligned with the natural coordinate system is

$$\sum_{v=1}^{2} \max\left(\frac{T_{i,j} - T_v}{h}, 0\right)^2 = F_{i,j}^2, \tag{2}$$

where h is the distance between nodes and

$$T_1 = \min(T_{i-1,j}, T_{i+1,j}), \tag{3}$$
$$T_2 = \min(T_{i,j-1}, T_{i,j+1}). \tag{4}$$

In a similar way, for a second-order approximation of the directional derivative, this equation must be solved:

$$\sum_{v=1}^{2} \max\left(\frac{3}{2h}[T_{i,j} - T_v], 0\right)^2 = F_{i,j}^2, \tag{5}$$

where

$$T_1 = \min\left(\frac{T_{i-1,j} - T_{i-2,j}}{3}, \frac{T_{i+1,j} - T_{i+2,j}}{3}\right), \tag{6}$$
$$T_2 = \min\left(\frac{T_{i,j-1} - T_{i,j-2}}{3}, \frac{T_{i,j+1} - T_{i,j+2}}{3}\right). \tag{7}$$

In both cases we have a quadratic equation to solve, resulting in two solutions. However, we must check the solution satisfies the causality relationship, that is, the new value is higher than the value of the neighbors used to compute it.

Algorithm 1 Fast Marching method (FMM)

Definitions:

- *Alive* set: points of the grid for which T has been computed and it will not be modified.
- *Trial* set: next points in the grid to be examined (4-connectivity) for which a estimation of T is computed using the points in *alive* set.
- *Far* set: the remaining points of the grid for which there is not an estimate for T.

Initialization:

- For each point in the grid, let $T_{i,j} = \infty$ (large positive value).
  Put all points in the *far* set.
- Set the start point $(i,j) = p_0$ to be zero:
  $U_{p_0} = 0$, and put it in the *trial* set.

Marching loop:

- Select $p = (i_{min}, j_{min})$ from *trial* with the lowest value of T.
- Put p in *alive* and remove it from the *trial* set.
- For each of the neighboring grid points (k,l) of (i_{min}, j_{min}):
  - If (k,l) belongs to *far* set, then put (k,l) in *trial* set.
  - If (k,l) is not in *alive* set, then compute $T_{k,l}$.

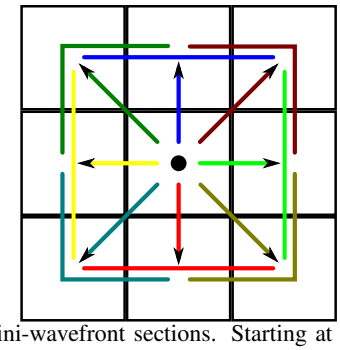


Figure 1. Mini-wavefront sections. Starting at any pixel, eight different sections can be computed, using an 8-connectivity. Each wavefront section center is defined by a neighbor node.

S_2 Stencil Computation: The diagonal equations are similar to the S_1 stencil. The first-order equation is

$$\sum_{v=1}^{2} \max\left(\frac{T_{i,j} - T_v}{\sqrt{2}h}, 0\right)^2 = F_{i,j}^2, \tag{8}$$

where

$$T_1 = \min(T_{i-1,j-1}, T_{i+1,j+1}), \tag{9}$$
$$T_2 = \min(T_{i+1,j-1}, T_{i-1,j+1}), \tag{10}$$

and the second-order equation:

$$\sum_{v=1}^{2} \max\left(\frac{3}{2\sqrt{2}h}[T_{i,j} - T_v], 0\right)^2 = F_{i,j}^2, \tag{11}$$

where

$$T_1 = \min\left(\frac{T_{i-1,j-1} - T_{i-2,j-2}}{3}, \frac{T_{i+1,j+1} - T_{i+2,j+2}}{3}\right), \tag{12}$$

$$T_2 = \min\left(\frac{T_{i+1,j-1} - T_{i+2,j-2}}{3}, \frac{T_{i-1,j+1} - T_{i-2,j+2}}{3}\right). \tag{13}$$

3. 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. Since our system is developed to be used in the computer vision field, regular 8-connectivity grid is used.

Wavefront Structure: Eight different wavefront sections can be defined, as we show in Fig. 1. Each section has an associated pixel node as its center, and other two nodes pointing the mini-wavefront boundaries. The boundary nodes are neighbors to both the center node and the node which starts the propagation.

We define $S(p) = \{(p, T_c); (p_l, T_l); (p_r, T_r)\}$ as the mini wavefront core information, where (p, T_c) is the position and the tentative value in the section center, respectively, with (p_l, T_l) and (p_r, T_r) their equivalent in the mini wavefront boundaries. Interpolation techniques can also be used to calculate $T(p)$ along the wavefront. For a more detailed explanation, see Section 3.3. Finally, we define $T_{S(p)}(p_n)$ as the tentative value in p_n if the solution is propagated from the wavefront section $S(p)$.

Algorithm 2 Wavefront Marching Method (WMM)

Definitions:

- *Trial* set: pairs $(S, R(S))$, consisting in the next mini wavefront sections S ready to be evaluated by using the value $R(S)$, which is an estimation of the quality of the section.
- *Far* set: the remaining points of the grid without a calculated estimate for U.

Initialization:

- For each point in the grid, let $U_{i,j} = \infty$ (large positive value). Put all points in the *far* set.
- Set the start point $(i, j) = p_0$ to be zero: $U_{p_0} = 0$, create the starting pair $(S(p_0).R(S(p_0))) = (\{(p_0, 0); (p_0, 0); (p_0, 0)\}, 0)$ and put it in the *Trial* set.

Marching loop:

- Select $S(p) = \{(p, T_c); (p_l, T_l); (p_r, T_r)\}$ from *Trial* with the lowest value of R and remove it from the set.
- For each 8-connectivity neighbor p_n, compute the mini wavefront sections $S(p_n)$ and their quality measure $R(S(p_n))$ using the upwinding criterion, and put them into the *Trial* set.

Algorithm: In algorithm 2 a description of the method is presented. In essence, WMM differs from the FMM by the use of mini wavefront sections instead of nodes to create the minimal action surface U. In our case, the *Alive* set contains all the mini wavefront sections S that were examined.

During this procedure, when the algorithm adds new mini wavefront sections to the *Trial* set, we have to determine the quality measure $R(S(p))$. This is done by using the equation

$$R(S(p)) = \min_{p_s \in S(p)} T(p_s) + \|p_s - p\|\epsilon, \tag{14}$$

where ϵ is an infinitesimal value and $p_s \in \{p, p_l, p_r\}$, the nodes contained within the wavefront section. $R(S(p))$ is, basically, the lower value in the mini wavefront section $S(p)$. The inclusion of the infinitesimal value is related to the idea of introducing order in the selection of the wavefront sections. To illustrate this idea we can think of the following example: two sections, $S(p_a)$, and $S(p_b)$, with the same lower value. In the first section, the lower value

is located into its center node, while in the second one is located into one of its boundaries. The inclusion of the infinitesimal value guarantees that WMM chooses first the mini wavefront sections which lower value is placed closest to its node center.

In the final step, when adding the mini wavefront section $S(p_n)$ into the *Trial* set, we have to check the minimal action surface value $T(p_n)$. If $T_{S(p)}(p_n) < T(p_n)$, we have to follow these steps:

1. Put $T(p_n) = T_{S(p)}(p_n)$.
2. Remove any other mini wavefront section in the *Trial* set where its node center is p_n.
3. Include the pair $(R(S(p_n)), S(p_n))$ into the *Trial* set.

3.1. Tentative value computation

Assume we have a wavefront section $S(p) = \{(p, T_c); (p_l, T_l); (p_r, T_r)\}$ and a reaching point p_s. In order to compute the given tentative value $T_{S(p)}(p_s)$, classical FMM techniques use a linear approximation to compute the accumulated cost, which weak solution is given by Godunov [12] (Both FMM and MSFM use finite difference schemes, resulting in the Eq. 2 or its variants). However, this technique estimates the traveling cost using the speed of motion F given in p_s, resulting in a constant function. Similar to the idea exposed in [2], we compute the solution by approximating the integral between the reaching point and the wavefront section in the direction of the motion. Thus, we define two values, one per each wavefront section

$$T^l_{S(p)}(p_s) = \min_{0 \le t \le 1} T(p^l_t) + \|p_s - p^l_t\| \frac{F(p^l_t) + F(p_s)}{2}, \tag{15}$$

where $p^l_t = (1-t)p + tp_l$, and

$$T^r_{S(p)}(p_s) = \min_{0 \le t \le 1} T(p^r_t) + \|p_s - p^r_t\| \frac{F(p^r_t) + F(p_s)}{2}, \tag{16}$$

where $p^r_t = (1-t)p + tp_r$. Thus, the tentative value is defined as

$$T_{S(p)}(p_s) = \min(T^l_{S(p)}(p_s), T^r_{S(p)}(p_s)). \tag{17}$$

Thus, the key of this model is the way of selecting an appropriate t value and how to compute its respective $T(p_t)$ and $F(p_t)$ values. In this work, we present three different techniques to solve the minimization problem for t selection. Furthermore, four different interpolation techniques are evaluated to estimate both T and F values.

3.2. Minimization Techniques

To select the t value that minimizes Eqs. 15 and 16, we propose 3 different approximations that will be evaluated in experimental section.

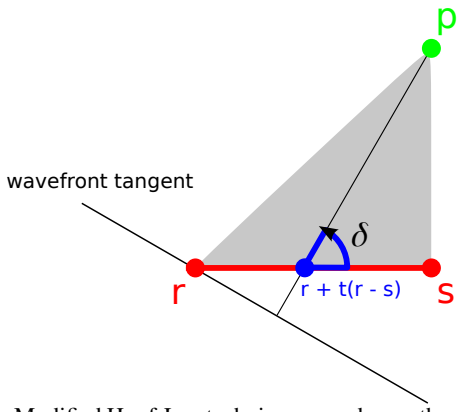


Figure 2. Modified Hopf-Lax technique. r and s are the endpoints of a wavefront section, whereas p is the point to be updated. Note that for values $t < 0$ or $t > 1$, t must be bounded to fit within the wavefront section.

Golden Section Search: Since our method solves the Eikonal equation by propagating monotone advancing fronts, it is clear to assume that the interpolation values within a wavefront segment results in a strictly unimodal function. The golden section search is a classic technique for finding the extreme (minimum in this case) of this kind of functions. However, this technique requires an iterative search over the segment, reducing the computational speed of the algorithm. Thus, we propose the additional evaluation of other non-iterative approaches for finding the best t value.

Modified Hopf-Lax Formula: A way to update $T(p_t)$ in the Eikonal equation using the Hopf-Lax formula is explained in [5]. In our case, we are going to modified this formula to take advantage of the interpolation techniques used in our wavefront section. Let p be the node we are going to update, and r, s the endpoints of a wavefront section, as explained in Fig. 2. Having $d = s - r$, the t value is computed by solving the equation

$$At^2 + Bt + C = 0, \tag{18}$$

where

$$\begin{aligned} A &= d_y^2 \cos^2\delta - d_x^2 \sin^2\delta, \\ B &= 2\big[(r_y - p_y)d_y \cos^2\delta - (r_x - p_x)d_x \sin^2\delta\big], \\ C &= (r_y - p_y)^2 \cos^2\delta - (r_x - p_x)^2 \sin^2\delta, \\ \cos\delta &= \frac{T(s) - T(r)}{\|s - r\|}, \\ \sin\delta &= \sqrt{1 - \cos^2\delta}. \end{aligned} \tag{19}$$

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 ($\cos\delta$). Note that it must be narrowed, that is, $t \in [0, 1]$.

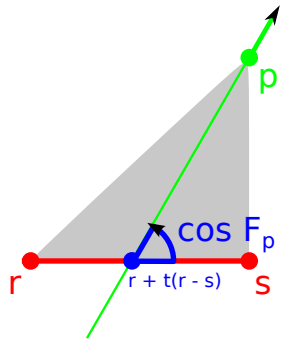


Figure 3. Gradient technique. r and s are the endpoints of a wavefront section, whereas p is the point to be updated. Note that for values $t < 0$ or $t > 1$, t must be bounded to fit within the wavefront section.

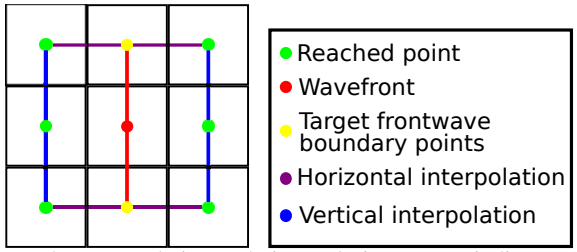


Figure 4. Interpolation techniques during an upwinding procedure. 4 different interpolation segments are created.

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, as shown in Eq. 1. However, the direction of the speed of motion can be used to obtain the t value, as shown in Fig. 3. Similar 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, let $F = (F_x, F_y)$ be the speed of motion, the formula is given by

$$t = \max\left(\min\left(\frac{F_y(r_x - p_x) - F_x(r_y - p_y)}{F_x d_y - F_y d_x}, 1\right), 0\right). \tag{20}$$

3.3. Interpolation Techniques

As mentioned in Algorithm 2, with every mini wavefront section $S(p_{i,j})$, we update the p neighbors using an 8-connectivity. Since we are using a regular grid, we have four different segments with 3 values each, as exposed in Fig 4. Thus, 4 different interpolation segments are defined. Four different interpolation techniques were developed to compute $T(p_t)$. Since the computation differs whether the wavefront is in the corner or not, below we are going to explain how to compute $T(p_t)$ in both cases, that is, $p_{i+1,j}$ and $p_{i+1,j+1}$. $F(p_t)$ value computation is analogue.

Linear Interpolation: $T(p_t)$ value computation is defined as

$$T^l_{S(p_{i+1,j})}(p_t) = (1-t)T_{i+1,j} + tT_{i+1,j-1}, \tag{21}$$
$$T^r_{S(p_{i+1,j})}(p_t) = (1-t)T_{i+1,j} + tT_{i+1,j+1} \tag{22}$$

for the wavefront centered in $p_{i+1,j}$ and

$$T^l_{S(p_{i+1,j+1})}(p_t) = (1-t)T_{i+1,j+1} + tT_{i+1,j}, \tag{23}$$
$$T^r_{S(p_{i+1,j+1})}(p_t) = (1-t)T_{i+1,j+1} + tT_{i,j+1} \tag{24}$$

when it is centered in $p_{i+1,j+1}$.

Quadratic Interpolation: $T(p_t)$ value computation value is defined as

$$T^d_s(p_t) = a^d_s t^2 + b^d_s t + c^d_s, \tag{25}$$

where $d = \{l, r\}$ and $s = \{S(p_{i+1,j}), S(p_{i+1,j+1})\}$. In our examples,

$$\begin{aligned} a^l_{S(p_{i+1,j})} &= T_{i+1,j-1} - b^l_{S(p_{i+1,j})} - T_{i+1,j}, \\ b^l_{S(p_{i+1,j})} &= \frac{T_{i+1,j-1} - T_{i+1,j+1}}{2}, \\ a^r_{S(p_{i+1,j})} &= T_{i+1,j+1} - b^r_{S(p_{i+1,j})} - T_{i+1,j}, \\ b^r_{S(p_{i+1,j})} &= \frac{T_{i+1,j+1} - T_{i+1,j-1}}{2}, \\ c^d_{S(p_{i+1,j})} &= T_{i+1,j} \end{aligned} \tag{26}$$

for the wavefront centered in $x_{i+1,j}$ and

$$\begin{aligned} a^l_{S(p_{i+1,j+1})} &= T_{i+1,j-1} - b^l_{S(p_{i+1,j+1})} - T_{i+1,j+1}, \\ b^l_{S(p_{i+1,j+1})} &= \frac{4(T_{i+1,j} - T_{i+1,j+1}) - T_{i+1,j-1}}{2}, \\ a^r_{S(p_{i+1,j+1})} &= T_{i+1,j+1} - b^r_{S(p_{i+1,j+1})} - T_{i+1,j+1}, \\ b^r_{S(p_{i+1,j+1})} &= \frac{4(T_{i,j+1} - T_{i+1,j+1}) - T_{i-1,j+1}}{2}, \\ c^d_{S(p_{i+1,j+1})} &= T_{i+1,j+1} \end{aligned} \tag{27}$$

in the upper-right corner case.

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, as it can be seen in Fig. 5. Thus, we have to check if the result is higher than any of its endpoints. If not, linear interpolation is used to compute the $T(p_t)$ value.

Monotone Piecewise Cubic Hermite Interpolation: 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 [11]. The model is similar to the classic Cubic Hermite Interpolation. It only differs in the tangents computation.

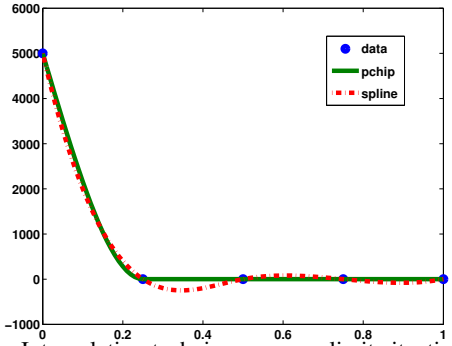


Figure 5. Interpolation techniques over a limit situation. Spline technique produces negative results, which is forbidden in this kind of algorithms. On the contrary, PCHIP preserves monotonicity.

4. Experimental Results

To make a significant evaluation of our method, we tested it against state-of-the-art methods using two experimental approaches present in the literature. In first place, we provide some analytical results that shows both the accuracy and the velocity of our method. Finally, we also provide a segmentation experiment, which is more related to the computer vision field. The code and test of both methods can be downloaded in our webpage[1].

Analytical Results: In order to test their accuracy, different speed models are performed, where the exact analytical solution (which we call T_a) is known. The speed is derived from the following equation

$$F_1 = \nabla T_a \tag{28}$$

for our WMM model using gradient minimization, and

$$F_2 = \nabla |T_a| \tag{29}$$

for the other models. In the following tables, our WMM follows this nomenclature: the minimization technique is mentioned in the superscripts:

- g: gradient
- h: modified hopf-lax
- s: golden section search

On the other hand, interpolation techniques are mentioned in the subscript, as follows:

- l: linear
- q: quadratic
- s: spline
- p: pchip

[1] Available at https://github.com/braisCB/WMM/

As first experiment, we compute the accuracy of the models under a speed function that corresponds to an moving front from the starting point (x_0, y_0) with an unit speed, and we compare the results against two well-known techniques, the 2nd order FMM version (FMM_2), and its most recent variant, the 2nd order MSFM (MSFM_2) developed Hassouna and Farag [13]. To make a fair comparison, both methods were implemented in the same language, c++. The size of the grid is 101×101 and $h_x = h_y = 1$. In Table 1 some error norms are shown from different starting points. Although the linear interpolation using hopf-lax minimization is not able to overcome the MSFM method, all the other interpolation techniques can overcome them. Note that, contrary to the FMM and the MSFM versions we are testing, our method does not use any second order scheme.

Our model is also capable to work with anisotropic grids. In Table 2 we test our method using a linear speed function, using different grid configurations, whether it is isotropic ($h_x = h_y = 1$) or not ($h_x = 0.1$ and $h_y = 0.2$). As our model approximates the integral between the reaching point and the wavefront section, the use of anisotropic grids does not affect the behavior of the model, unlike MSFM. It is also noticeable that the gradient minimization and the golden section search obtains similar results. This proves gradient minimization as a good choice to be used, since the difference in the computational cost between these two techniques is really high.

Finally, we test our algorithm against a recent technique developed by Appia and Yezzi [2]. Using the test they provide against two different cosine functions, which is shown in Table 3, our algorithm is capable to find similar results, whereas the computational cost of our algorithm is lower. Furthermore, we found that our method, using quadratic interpolation, performs even faster than the MSFM. This is a remarkable achievement, since our method clearly outperforms MSFM accuracy. As in [2], the computational time is achieved on a 500×500 grid. It was measured using a computer with a 3.20 GHz Processor.

We conclude that our method not only obtains the best accuracy results in most of the provided tests, but also is one of the fastest. To our knowledge, the WMM has the best balance between accuracy and efficiency. Note that, although our implementation has a $\mathcal{O}(M \log M)$ complexity, it can be reduced to $\mathcal{O}(M)$ by implementing the trial set as an untidy priority queue [23].

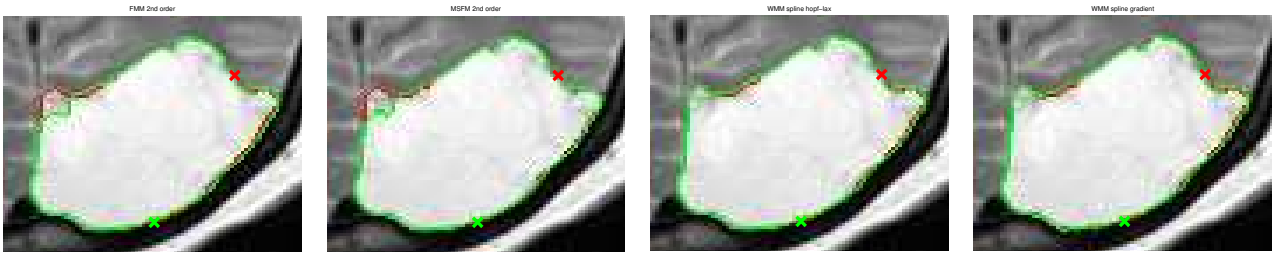
Image Segmentation: We test our methodology in a computer vision scenario related with medical imaging in order to illustrate the applicability of these techniques in real-world problems. In particular, techniques are tested for segmentation of a brain tumor in a brain CT image, as shown

Table 1. Error norms of the equation $\sqrt{(x-x_0)^2+(y-y_0)^2}$ from different source points of a grid of size 101×101.

Time $T(p)$	$\sqrt{(x-x_0)^2+(y-y_0)^2}$								
Source Point(s)	$(51,51)$			$(1,1)$			$(21,24)$ and $(65,74)$		
Method/error	L_1	L_2	L_∞	L_1	L_2	L_∞	L_1	L_2	L_∞
FMM_2	0.35278	0.14826	0.58027	0.34672	0.14823	0.58630	0.30025	0.11017	0.53609
MSFM_2	0.06927	0.00771	0.21550	0.05882	0.00444	0.21233	0.07458	0.00985	0.42054
WMM_l^h	0.11823	0.01978	0.27562	0.13979	0.02829	0.33306	0.13118	0.02359	0.30854
WMM_q^h	**0.01510**	**0.00035**	**0.03493**	0.05378	0.00507	0.12078	**0.01395**	**0.00030**	**0.04375**
WMM_s^h	0.03217	0.00170	0.09128	0.05943	0.00616	0.16179	0.03330	0.00182	0.10126
WMM_p^h	0.01585	0.00039	0.03716	**0.05004**	**0.00343**	**0.09171**	0.02215	0.00118	0.22218

Table 2. Error norms of the equation $\frac{(x-x_0)^2}{100}+\frac{(y-y_0)^2}{20}$ from an anisotropic grid of size 101×101 with $(51,51)$ as source point.

Time $T(p)$	$\frac{(x-x_0)^2}{100}+\frac{(y-y_0)^2}{20}$					
(h_x,h_y)	$(1,1)$			$(0.1,0.2)$		
Method/error	L_1	L_2	L_∞	L_1	L_2	L_∞
FMM_2	0.11405	0.01317	0.12011	0.38843	0.24317	0.09889
MSFM_2	0.04725	0.00250	0.10465	0.01754	0.02293	0.05277
WMM_l^g	0.09001	0.00995	0.17887	0.00254	$9.07e-6$	0.00856
WMM_q^g	0.01667	0.00057	0.12465	0.00415	$2.96e-5$	0.02811
WMM_s^g	0.02584	0.00085	0.05493	0.00071	$6.32e-7$	0.00151
WMM_p^g	0.01999	0.00248	0.47550	**0.00027**	$\mathbf{2.77e-7}$	0.00476
WMM_l^s	0.08928	0.00980	0.17922	0.00221	$6.58e-6$	0.00659
WMM_q^s	**0.01189**	**0.00040**	0.11776	0.00279	$1.38e-5$	0.00922
WMM_s^s	0.02159	0.00060	**0.04451**	0.00059	$4.80e-7$	**0.00138**
WMM_p^s	0.01876	0.00277	0.49845	0.00035	$3.80e-7$	0.00526



(a) FMM_2 (b) MSFM_2 (c) WMM_s^g (d) WMM_s^h

Figure 7. A comparison of segmentation starting in two different points (red and green contours). Each initial point is marked with a cross. 2nd order FMM and MSFM fail to provide an accurate segmentation. Its accuracy depends on the chosen starting point. On the contrary, our method obtains good results in every situation.

in Fig. 6. We use as speed of motion

$$F_x(p) = \frac{\cos(\nabla I^t)}{1+|\nabla I|^2}, \qquad F_y(p) = \frac{\sin(\nabla I^t)}{1+|\nabla I|^2}, \tag{30}$$

where I is the intensity image. This equation helps the fast marching methods to propagate the solution along the zones with higher gradients. Once T is computed, we detect the saddle points as described in [8]. A saddle point is a point such that it is possible to reach the initial point by using back-propagation over two different paths. This allows us to obtain a closed contour. Unfortunately, the number of saddle points is really high in this kind of image. We select only one saddle point: the one that minimizes the Chan-Vese energy [6].

We deliberately decide to use this example because of its simplicity. Nonetheless, 2nd order methods fail to provide a good solution. Fig 7 shows the overlay results of two different segmentations using two different starting points. 2nd order FMM and MSFM fail to provide the same result in both segmentations, because the 2nd order scheme check in both algorithm is too relaxed. Classic higher order techniques require the values that are two pixels away to be lower than the immediate neighbor. The condition is used to make sure the value in the immediate neighbor was obtained by using the two pixels away pixel, which happens in ideal scenarios, where the direction of the motion

Table 3. Error norms from a grid of size 50×50 with $(26, 26)$ as source point, and ordered by their computational times. [1] Times obtained using a slower machine [2]. (Only for ordering purposes). Our method is highlighted with blue rows. The best accuracies are presented in bold type.

Time $T(p)$	$1-\cos(\frac{x-x_0}{20})\cos(\frac{y-y_0}{20})$			$1-\cos(\frac{x-x_0}{20})\cos(\frac{y-y_0}{10})$			Time (s)
Method/error	L_1	L_2	L_∞	L_1	L_2	L_∞	
FMM_2	$2.46e-2$	$6.73e-4$	0.0380	$4.37e-2$	$2.07e-3$	0.1060	0.23
Tsitsiklis	$2.14e-2$	$4.89e-4$	0.0281	$3.81e-2$	$1.57e-3$	0.0825	$(0.26)^1$
WMM_q^g	$1.00e-3$	$1.93e-6$	0.0032	$1.69e-3$	$2.64e-5$	0.0811	0.32
WMM_l^g	$1.45e-3$	$2.76e-6$	0.0032	$3.87e-3$	$4.16e-5$	0.0852	0.33
WMM_q^h	$1.77e-2$	$4.90e-4$	0.0452	$1.83e-2$	$4.97e-4$	0.1076	0.36
WMM_l^h	$1.64e-2$	$4.36e-4$	0.0438	$1.84e-2$	$5.03e-4$	0.1075	0.37
$MSFM_2$	$2.36e-2$	$6.07e-4$	0.0349	$4.23e-2$	$1.94e-3$	0.1007	0.37
WMM_p^g	$4.40e-4$	$3.28e-7$	0.0015	$1.31e-3$	$2.05e-5$	0.0749	0.40
WMM_p^h	$1.64e-2$	$4.35e-4$	0.0437	$1.82e-2$	$4.97e-4$	0.1075	0.43
WMM_s^g	$2.98e-4$	$1.38e-7$	0.0008	$1.71e-3$	$2.68e-5$	0.0819	0.44
WMM_s^h	$1.64e-2$	$4.35e-4$	0.0438	$1.83e-3$	$4.97e-4$	0.1075	0.44
Linear8	$2.25e-3$	$6.82e-6$	0.0046	$4.46e-3$	$3.43e-5$	0.0596	$(0.52)^1$
Bilinear8	$2.74e-3$	$9.42e-6$	0.0052	$5.01e-3$	$4.10e-5$	0.0607	$(0.65)^1$
IsoLinear8	$2.25e-3$	$6.82e-6$	0.0046	$4.03e-3$	$3.11e-5$	0.0596	$(0.91)^1$
WMM_q^s	$8.97e-4$	$1.68e-6$	0.0040	$\mathbf{1.23e-3}$	$2.04e-5$	0.0803	0.93
WMM_l^s	$1.50e-3$	$2.98e-6$	0.0034	$3.70e-3$	$3.67e-5$	0.0819	1.16
WMM_p^s	$4.58e-4$	$3.45e-7$	0.0012	$1.76e-3$	$1.86e-5$	0.0785	1.31
Up8	$2.99e-4$	$1.96e-7$	0.0014	$1.54e-3$	$\mathbf{7.81e-6}$	**0.0289**	(1.42^1)
UpSG	$1.96e-3$	$4.15e-6$	0.0035	$1.20e-2$	$1.94e-4$	0.0566	(1.42^1)
WMM_s^s	$\mathbf{2.70e-4}$	$\mathbf{1.19e-7}$	**0.0008**	$1.41e-3$	$2.14e-5$	0.0813	2.00

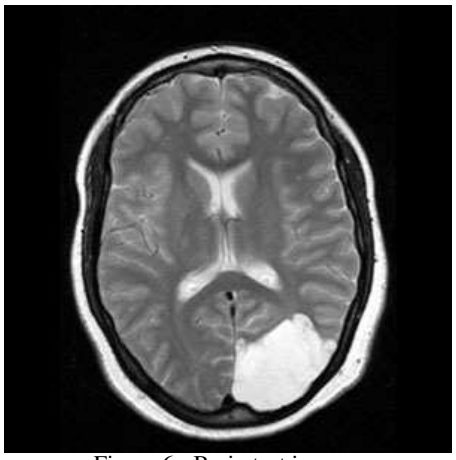


Figure 6. Brain test image.

has no sudden changes. However, when dealing with images this condition is too relaxed, since every edge is just a sudden orientation change, causing 2nd or even 3rd order schemes behavior to choose as values that not belongs to the same propagation orientation. On the contrary, our method, even using interpolation techniques, is able to preserve the same shape, since only the immediate neighbors are taking into account in the updating procedure, making the WMM a more suitable algorithm to be used in the image domain.

5. Conclusion

In this paper we present a new methodology for improving the existent marching method techniques under monotonically advancing fronts. Our approach creates wavefront sections that are used to update each neighbor's value. We introduce three different variants to solve the updating procedure. We also show that interpolation techniques can reduce the error of the algorithm. Experimental results demonstrate how our method achieve the highest balance between robustness and velocity, usually outperforming state-of-the-art in both parameters simultaneously.

As a future development, we plan to extend our algorithm to work with anisotropic forces in a similar way the Ordered Upwind Method does. We think an extension in our model enables it to solve the classical static Hamilton-Jacobi equation without significantly increasing the computational cost of our model. Furthermore, we would like to extend our algorithm to work with 3D images.

References

[1] S. Ahmed, S. Bak, J. McLaughlin, and D. 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

[2] V. Appia and A. Yezzi. Fully isotropic fast marching methods on cartesian grids. In *Computer Vision–ECCV 2010*,

pages 71–83. Springer, 2010. 4, 6, 8

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

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

[5] F. Bornemann and C. 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

[6] T. Chan and L. Vese. An active contour model without edges. In *Scale-Space Theories in Computer Vision*, pages 141–151. Springer, 1999. 7

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

[8] L. D. Cohen and R. Kimmel. Global minimum for active contour models: A minimal path approach. *International Journal of Computer Vision*, 24:57–78, 1997. 1, 7

[9] P. Covello and G. 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

[10] P.-E. Danielsson and Q. Lin. A modified fast marching method. In *Image Analysis*, pages 1154–1161. Springer, 2003. 1

[11] F. N. Fritsch and R. E. Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980. 5

[12] S. K. Godunov. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Matematicheskii Sbornik*, 89(3):271–306, 1959. 4

[13] M. S. Hassouna and A. 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, 2, 6

[14] S. Jbabdi, P. Bellec, R. Toro, J. Daunizeau, M. Pélégrini-Issac, and H. Benali. Accurate anisotropic fast marching for diffusion-based geodesic tractography. *Journal of Biomedical Imaging*, 2008:2, 2008. 1

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

[16] E. Konukoglu, O. Clatz, B. H. Menze, B. Stieltjes, M.-A. Weber, E. Mandonnet, H. Delingette, and N. 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

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

[18] J.-M. Mirebeau. Efficient fast marching with finsler metrics. *Numerische Mathematik*, pages 1–43, 2012. 1

[19] M. Péchaud, R. Keriven, and G. 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

[20] 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

[21] J. A. 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

[22] 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

[23] L. Yatziv, A. Bartesaghi, and G. Sapiro. O(n) implementation of the fast marching algorithm. *Journal of computational physics*, 212(2):393–399, 2006. 1, 6

[24] Y.-T. Zhang, H.-K. Zhao, and J. Qian. High order fast sweeping methods for static hamilton–jacobi equations. *Journal of Scientific Computing*, 29(1):25–56, 2006. 1

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