# A Fast Marching Method for Hamilton-Jacobi Equations Modeling Monotone Front Propagations*

Emiliano Cristiani†

November 15, 2008

**Abstract**

In this paper we present a generalization of the Fast Marching method introduced by J. A. Sethian in 1996 to solve numerically the eikonal equation. The new method, named Buffered Fast Marching (BFM), is based on a semi-Lagrangian discretization and is suitable for Hamilton-Jacobi equations modeling monotonically advancing fronts, including Hamilton-Jacobi-Bellman and Hamilton-Jacobi-Isaacs equations which arise in the framework of optimal control problems and differential games. We also show the convergence of the algorithm to the viscosity solution. Finally we present several numerical tests comparing the BFM method with other existing methods.

**Keywords** Fast Marching methods, front propagation, semi-Lagrangian schemes, Hamilton-Jacobi equations, optimal control problems.

**AMS** Primary, 65N12; Secondary, 49L20.

*This research was partially supported by the MIUR Project 2006 "Modellistica Numerica per il Calcolo Scientifico ed Applicazioni Avanzate" and by INRIA–Futurs and ENSTA, Paris, France.

†E-mail: emiliano.cristiani@gmail.com, cristian@mat.uniroma1.it. Corresponding address: via Nazario Sauro 21A, 00012 Villanova di Guidonia (RM), Italy. Tel. +390774527776, Mobile: +393492639591

# 1 Introduction

The Fast Marching (FM) method is a numerical method for the eikonal equation

$$\begin{cases} c(x)|\nabla T(x)| = 1 & x \in \mathbb{R}^n \backslash \Omega_0 \\ T(x) = 0 & x \in \partial\Omega_0 \end{cases} \tag{1}$$

where $\Omega_0$ is a closed set and $c : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuous and strictly positive. This equation appears in front propagation problems in which the interface propagates in normal direction with speed $c(x)$, more precisely the $t$-level set of its viscosity solution $T$ is the interface at time $t$. The interface at time $t = 0$ is given by $\Gamma_0 = \partial\Omega_0$. The FM method is officially born with the paper of Sethian [22] in 1996 (see also his book [21]). Before that, Tsitsiklis [26] already proposed a slightly different Dijkstra-like algorithm based on a control-theoretic discretization which contains all the basic ideas of the FM technique. The method is very powerful because it is able to compute the viscosity solution of (1) much faster than any other iterative algorithms in which every node of the grid is computed at every iteration. Its computational cost is $O(N \ln N)$ where $N$ is the total number of grid nodes. Since its first appearance, it was applied in many fields like mesh generation, seismology, geodesic computation, image and video segmentation, image enhancement, dislocation dynamics and so on. The original FM method is based on the following *up-wind* first-order finite difference approximation (we choose $n = 2$ to avoid cumbersome notations)

$$\begin{aligned} (\max\{\max\{D_x^-, 0\}, -\min\{D_x^+, 0\}\})^2 + \\ +(\max\{\max\{D_y^-, 0\}, -\min\{D_y^+, 0\}\})^2 = c_{i,j}^{-2} \end{aligned} \tag{2}$$

where $D_x^- = \frac{T_{i,j} - T_{i-1,j}}{\Delta x}$, $D_x^+ = \frac{T_{i+1,j} - T_{i,j}}{\Delta x}$ (and analogous definition for $D_y^+$ and $D_y^+$) and $T_{i,j} = T(i\Delta x, j\Delta y)$ as usual. The FM technique consists in computing the values at the nodes in a special order such that convergence is reached in just one iteration. At a generic step of the algorithm the grid nodes are divided in three sets, *accepted*, *narrow band* and *far* nodes. The accepted nodes are those where the solution has been already computed and where the value can not change in the following iterations. The narrow band nodes are the nodes where the computation actually takes place and their value can still change at the following iterations. Finally, the far nodes are the remaining nodes where an approximate solution has not been computed yet. In physical terms, the far nodes are those in the space region which has not been touched by the front yet, the accepted nodes are those where the front has already passed through and the narrow band nodes are, iteration by iteration, those lying in a neighborhood of the front. The crucial point is how the nodes in the narrow band are chosen to become accepted. This condition must guarantee that the value of those nodes can

not change in following iterations. In the classical FM method the criterion is *picking the node (only one at a time) with the minimal value.*
In the last decade many authors tried to improve the FM method in both velocity and accuracy. Other papers were devoted to the extension of the FM method to more general equation. This is probably the most difficult task since the FM method strictly relies on some particular properties of the eikonal equation as we will see in section 2.2.
Kimmel and Sethian [17] extended the FM method to triangulated domains on manifolds preserving the same computational complexity (see also [24]).
Again Kimmel and Sethian [18] extended the FM method to an equation of the form (1) in which $c$ depends on $x$ and $T$ itself. They apply their result to the solution of the Shape from Shading problem.
Sethian and Vladimirsky [23] extended the FM method to equation of the form (1) in which $c$ depends on $x$ and $\nabla T/|\nabla T|$ on unstructured grids. This equation includes the case of the anisotropic front propagation problem. The authors explain in detail the limitations of the classical FM technique and how they can be overcome. Unfortunately they did not perform many numerical tests and did not present CPU times needed for computations.
Prados [19] proposed an interesting generalization of the FM method to solve Hamilton-Jacobi-Bellman equations. The new method changes the way a node is accepted, it is not the node with the minimal value $T$ in the narrow band but it is the node with the minimal value $T - \phi$ where $\phi$ is a viscosity subsolution of the equation. Of course this can not be considered a real numerical method because a subsolution must be known, nevertheless this procedure can be a useful suggestion for further developments.
The papers [15, 14] made a comparative study of FM method and other existing methods for the eikonal equation, in particular with the Fast Sweeping method (see [25, 20] and references therein) which can overcome FM method in some situations. Carlini et al. [4, 5] extended the FM method to the evolutive eikonal equations modeling non-monotone front propagation problems in which the velocity $c$ of the front can change sign in space and/or in time. The authors apply their results to the simulation of the dislocation dynamics in which the velocity $c$ does not depend locally on a point $x$ but is given in an integral form.
Vladimirsky [27] deals with equations of the form (1) in which $c$ depends on $x$, $\nabla T(x)$ and $T$. He presents a rigorous analysis and some numerical experiments.
The author and Falcone [9] introduced the Characteristic FM method for the eikonal equation which, similarly to Kim [16], accepts more than one node at the same time and it is faster than the FM method in most cases.
Again the author and Falcone [8] (see also [6]) introduced the FM method based on the semi-Lagrangian discretization in the framework of control theory and minimum time problem. The semi-Lagrangian scheme is proved

to be more accurate than the finite difference scheme classically used in FM method although they are both first order schemes.

In this paper we introduce a new FM method based on a semi-Lagrangian discretization which is able to compute an approximate solution of Hamilton-Jacobi equations modeling front propagation problems in which the front does nor pass more than one time on the same point. This class of equations includes Hamilton-Jacobi-Bellman and Hamilton-Jacobi-Isaacs equations which arise in the framework of optimal control problems and differential games. The new method, named *Buffered Fast Marching*, uses a forth set (the buffer) in addition to the sets accepted, narrow band and far to menage the nodes. The buffer is in the middle between the narrow band and the accepted zone and contains the nodes until they can be accepted once and for all. The size of the buffer depends on the anisotropy of the problem. In the case of the eikonal equation (1) (corresponding to an isotropic front propagation) the buffer disappears and the Buffered Fast Marching method changes back into the Fast Marching method.

The paper is organized as follows. In section 2 we introduce the equations we deal with and we recall the FM method for the eikonal equation based on the semi-Lagrangian discretization introduced in [8]. We also show why the FM technique does not work for more general equations. In section 3 we introduce the Buffered Fast Marching (BFM) method, detailing the algorithm and its properties. Finally in section 4 we present some numerical tests on a series of benchmarks commenting the general behavior of the solutions and CPU times.

# 2 Background

In this section we introduce the equations we deal with and we recall the FM method based on the semi-Lagrangian discretization introduced in [8]. It will be the foundation of the Buffered FM method proposed here. We also recall the limitations of the FM method.

## 2.1 Related equations

A front propagation problem consists in recovering the position of a front $\Gamma_t : \mathbb{R}^+ \to \mathbb{R}^n$ (for example the interface between two layers) at any time $t$ starting from an initial configuration $\Gamma_0$. We denote by $\Omega_t$ the region inside the front $\Gamma_t$. One of the most popular method to face this kind of problem is the level set method [21] in which we look for a function $u : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}$ such that $\Gamma_t = \{x \in \mathbb{R}^n : u(x,t) = 0\}$. It is well known that the function $u$

is solution of the following PDE

$$u_t(x,t) + \phi(x,t,u(x,t),\nabla u(x,t)) \cdot \nabla u(x,t) = 0\,, \quad x \in \mathbb{R}^n\,, t > 0 \tag{3}$$

where $\phi$ is the velocity of the front (note that it can also depend on $u(\cdot)$ all over the domain and on higher order derivatives as well) and the initial condition $u(x,0)$ is chosen as the signed distance function from $\Gamma_0$. If the velocity field $\phi$ is such that the front did not pass for any point $x$ more than one time the evolution is said to be *monotone*, *i.e.*

$$\Omega_{t_1} \subset \Omega_{t_2} \quad \text{for any} \quad t_1 < t_2. \tag{4}$$

If (4) is satisfied, it is proved in [21] that the front can be recovered by $\Gamma_t = \{x \in \mathbb{R}^n : T(x) = t\}$ where $T$ is the viscosity solution of the following time-independent equation

$$\phi(x,T,\nabla T) \cdot \nabla T(x) = 1\,, \qquad x \in \mathbb{R}^n \backslash \Omega_0. \tag{5}$$

where we use again the symbol $\phi$ for the velocity with an abuse of notation. If the direction of the velocity is normal to the interface the function $\phi$ has the form $\phi(x,T,\nabla T) = c(x)\frac{\nabla T}{|\nabla T|}$, so equation (5) can be written as $c(x)|\nabla T(x)| = 1$ (eikonal equation).
By the Kružkov transform $v(x) = 1 - e^{-T(x)}$, the equation (5) becomes

$$v(x) + \phi(x,v,\nabla v) \cdot \nabla v(x) - 1 = 0\,, \qquad x \in \mathbb{R}^n \backslash \Omega_0. \tag{6}$$

This equation is very general and is found in many applications, for example in the minimum time problem as follows [1, 6]. Let us consider the controlled nonlinear dynamical system

$$\begin{cases} \dot{y}(t) = f(y(t), a(t))\,, \quad t > 0 \\ y(0) = x \end{cases} \tag{7}$$

where $y(t)$ is the state of the system, $a(\cdot) \in \mathcal{A}$ is the control of the player, $\mathcal{A}$ being the set of admissible controls defined as

$$\mathcal{A} = \{a(\cdot) : [0, +\infty) \to A, \text{ measurable}\},$$

and $A$ is a given compact set of $\mathbb{R}^m$. Assume hereafter $f : \mathbb{R}^n \times A \to \mathbb{R}^n$ is continuous in both variables and Lipschitz continuous with respect to $y$ uniformly in $a$. The unique trajectory solution of (7) will be denoted by $y_x(t; a(\cdot))$. In the minimum time problem the final goal is to find an optimal control $a^*(t)$ such that the corresponding trajectory $y_x(t; a^*(\cdot))$ minimizes over all admissible trajectories the time needed by the system to reach a

given closed *target* $\mathcal{T} \subset \mathbb{R}^n$. The optimal control $a^*(t)$ can be computed by means of the *value function* $T$ defined as

$$T(x) := \begin{cases} \inf\limits_{a(\cdot)\in\mathcal{A}} \min\{t : y_x(t; a(\cdot)) \in \mathcal{T}\} & \text{if } y_x(t; a(\cdot)) \in \mathcal{T} \text{ for some } t \geq 0 \\ +\infty & \text{if } y_x(t; a(\cdot)) \notin \mathcal{T} \text{ for all } t \geq 0. \end{cases} \tag{8}$$

By the Dynamic Programming Principle it can be shown that $v = 1 - e^{-T}$ is the viscosity solution of

$$\begin{cases} v(x) + \max\limits_{a\in A}\{-f(x,a)\cdot\nabla v(x)\} - 1 = 0 & x \in \mathbb{R}^n \backslash \mathcal{T} \\ v(x) = 0 & x \in \partial\mathcal{T}. \end{cases} \tag{9}$$

Equation (9) is known as the Hamilton-Jacobi-Bellman equation for the minimum time problem. Note that $v$ is always in the interval $[0, 1]$ while $T$ is in general unbounded. Finally note that defining

$$a_*(x, \nabla v(x)) := \arg\max_{a\in A}\{-f(x,a)\cdot\nabla v(x)\},$$

$$\phi(x, \nabla v(x)) := -f(x, a_*(x, \nabla v(x))) \quad \text{and} \quad \Omega_0 := \mathcal{T}$$

the equation (9) takes the general form (6).

In order to discretize equation (9), we will introduce a structured grid $G$ denoting its nodes by $x_i\,,\ i = 1\ldots, N$, *i.e.* $G = \{x_i,\ i = 1, \ldots, N\}$. It was proved in [2] that the semi-Lagrangian scheme stems from a discrete version of the Dynamic Programming Principle (see f.e. [1]), this leads to the equation

$$\begin{cases} w(x_i) = \min\limits_{a\in A}\{\beta w(x_i - hf(x_i, a))\} + 1 - \beta & \text{for } x_i \in G\backslash\mathcal{T} \\ w(x_i) = 0 & \text{for } x_i \in G \cap \mathcal{T} \end{cases} \tag{10}$$

where $w$ is an approximation of $v$, $\beta = e^{-h}$, $h$ is a discretization step and we defined $w = 0$ also in the internal nodes of $\mathcal{T}$. We use a linear interpolation to approximate the value $w(x_i - hf(x_i, a))$. It has been shown in [11] that equation (10) has a unique solution $w$ in the class of piecewise linear functions defined on the grid. It can be computed by a fixed point technique, iterating until convergence

$$w^{(k+1)} = \Lambda(w^{(k)}) \qquad k = 0, 1, 2, \ldots \tag{11}$$

where $\Lambda(w)_i = \min_{a\in A}\{\beta w(x_i - hf(x_i, a)\} + 1 - \beta$ and $w^{(0)}$ is equal to 0 in $G\cap\mathcal{T}$ and 1 elsewhere. Since we want to use just the three nearest nodes to $x_i$ to compute $w(x_i - hf(x_i, a))$, we choose $h = h(x_i, a) = \Delta x/|f(x_i, a)|$. Of course the constant $\beta = e^{-h}$ must be included in the minimum over $a$'s.

We also recall the Hamilton-Jacobi-Isaacs equation which arises in the framework of differential games [1, 12],

$$\begin{cases} v(x) + \min\limits_{b\in B}\max\limits_{a\in A}\big\{ - f(x,a,b)\cdot\nabla v(x)\big\} - 1 = 0 & x\in\mathbb{R}^n\backslash\mathcal{T} \\ v(x) = 0 & x\in\partial\mathcal{T}. \end{cases} \tag{12}$$

Here $f$ is the dynamics for the game, $A$ and $B$ are two compact sets in $\mathbb{R}^m$ representing respectively the control sets for the first player and the second player. The two players can both steer the system, the first wants the system reaches the target $\mathcal{T}$ in the minimum time while the second player wants the system goes away for ever. The value function $T = -\ln(1-v)$ represents the time to reach the target if both players play optimal nonanticipative strategies.

## 2.2 The semi-Lagrangian FM method and its limitations

It is easy to show that choosing in (9)

$$f(x,a) = c(x)a\,, \quad A = B(0,1)\,, \quad \mathcal{T} = \Omega_0$$

and re-writing the equation in $T = -\ln(1-v)$, we get back to the eikonal equation (1). In the particular framework of the eikonal equation it was introduced in [8] the FM method based on the semi-Lagrangian discretization and it was proved that the new algorithm is slightly slower than the original FM method but much more accurate.
The idea which is behind the semi-Lagrangian FM method is rather simple: we follow the initialization and all the steps of the classical FM method but the step where the value at the node $x_i$ is actually computed where we use the semi-Lagrangian scheme instead of the finite difference scheme.

As we said in the introduction, in the last ten years the extensions of the FM method to more general equations were quite timid and limited to equations very similar to (1). This is due to the fact that the method strictly relies on its physical interpretation based on the isotropic front propagation problem. From the mathematical point of view, it appears that labeling as accepted the node in the narrow band with the minimal value is suitable only in the case the characteristics curves of the equation coincide with the gradient lines of its solution. This is due to the fact that accepting the minimal value in the narrow band means to compute $v$ (or $T$) in the ascending order and then to maintain the right up-winding only

in the case the optimal control $a^*(x) := \arg\max_{a \in A}\{-f(x,a)\cdot\nabla v(x)\}$ satisfies

$$a^*(x) = -\frac{\nabla v(x)}{|\nabla v(x)|}.$$

This is the case for the eikonal equation but it is not true in the general case. As stated in the Criterion 5.1 of [23], the FM method fails exactly where characteristics and gradient lines lie in different simplices (but it is able to compute the right solution elsewhere, even if the two directions does not coincide exactly).
In the case of differential games it is even more clear that the accepting-the-minimum rule of the FM technique can not work. In fact, the optimal trajectory for the second player aims for the higher values of the value function $v$.

# 3 The Buffered Fast Marching method

In this section we present the new method in detail. We also present a convergence result and some considerations about the computational cost.

## 3.1 Main idea of the BFM method

As we explained in section 2.2, the update procedure of FM method is not suitable for the numerical solution of equations different from the eikonal equation. On the contrary the BFM method is designed to solve correctly and rapidly any equation of the form (6).
In the proposed algorithm, the node in the narrow band with the minimum value is not accepted as it happens in the FM method but it is moved in a *buffer*. All the nodes in the buffer are recomputed at each step of the algorithm until it is sure that their value can not change any more, this is guaranteed by a local condition which will be introduced below. After that, they are finally labeled as accepted (see Fig. 1). Note that the size of the buffer strictly depends on the anisotropy coefficient and so the computational cost does.
We choose how the nodes go out from the buffer as follows. In a copy of the matrix where the computation is being performed we substitute the value $v = 1$ for all the values in the narrow band. Then we compute until convergence the nodes in the buffer iterating the computation (for example (11)). After that, we repeat the procedure substituting the value $v = 0$ for all the values in the narrow band. Finally, we look for the nodes whose values have not changed in the two steps. In other words, we treat the narrow band as part of the boundary of the computation domain, imposing at nodes in it two different boundary conditions. The first one is $v = 1$,

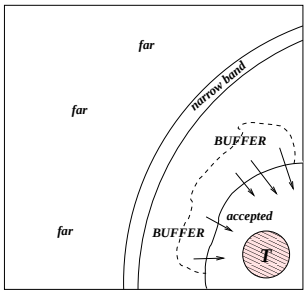


Figure 1: Division of the nodes: target, accepted, part of the buffer which is going to be accepted, buffer, narrow band and far

that is the maximum value a node can assume and the second is $v = 0$ that is the minimum value a node can assume. Clearly, if a node does not change its value after this kind of modification it means that its value does not depend on the next steps of the algorithm whatever it happens and then it can be labeled as accepted.

## 3.2 The algorithm

Let us introduce the algorithm. In the following, the narrow band will be denoted by *NB* and the buffer by *BUF*. We also introduce the following

**Definition 3.1** (neighboring nodes for the semi-Lagrangian scheme). Let the dimension $n$ be 2 and let $x_{i,j}$ be a node. We define the set of neighboring nodes to $x_{i,j}$ as

$$N_{SL}(x_{i,j}) := \{x_{i\pm1,j}, x_{i,j\pm1}, x_{i+1,j+1}, x_{i+1,j-1}, x_{i-1,j+1}, x_{i-1,j-1}\}.$$

The nodes in $N_{SL}(x_{i,j})$ are the nodes that appear in the stencil of the first order semi-Lagrangian discretization. The above definition can be easily extended to the $n$-dimensional case.

*The BFM algorithm*
*Initialization*

- Locate the nodes belonging to the initial front $\Gamma_0 = \partial\Omega_0 = \partial\mathcal{T}$ and label them as accepted. They form the set $\widetilde{\Gamma}_0$. Impose $v(\widetilde{\Gamma}_0) = 0$ (corresponding to $T(\widetilde{\Gamma}_0) = 0$).

- Define *NB* as the set of the nodes belonging to $N_{SL}(\widetilde{\Gamma}_0)$, external to $\Gamma_0$.

- Iterate the computation in $NB$ until convergence (as in the classical fixed point method).
- Label the remaining nodes as far, setting their values to $v = 1$ (corresponding to $T = +\infty$).

*Main Cycle*

1. Let $A$ be the node with the minimum value among all the nodes in $NB$. Find $A$, remove it from $NB$ and insert it in $BUF$. Move the far nodes of $N_{SL}(A)$ into $NB$ and (re)compute the nodes in $N_{SL}(A)$ which are not accepted.
2. Compute all the nodes in $BUF$ until convergence (as in the classical fixed point method).
3. In a copy of the matrix where the computation is being performed, substitute $v = 1$ for the value of the nodes in $NB$. Then iterate the computation on all the nodes in $BUF$ until convergence.
4. Again in the copy of the matrix, substitute $v = 0$ for the value of the nodes in $NB$. Iterate again the computation on all the nodes in $BUF$ until convergence.
5. Remove from $BUF$ and label as accepted the nodes whose value is not changed in the two previous steps.
6. If $NB$ is not empty go back to step 1, otherwise iterate the computation on all the nodes in $BUF$ until convergence and stop computation.

*Remark* **3.1** It is possible that in step 1 of the main cycle the node $A$ is not unique. In this case, the procedure described for the node $A$ can be repeated for all other nodes with the same value before passing to step 2.

**Real implementation**
Unfortunately, the described algorithm can be slower than the classical iterative scheme in many situations. Nevertheless, some tricks can speed up the computation. The real implementation of the method includes all the following modifications.

M1. Assuming that the solution is increasing along characteristics (this is the typical situation in the minimum time problem, for example) we use the current minimal value $v_{min}$ in the $NB$ instead of $v = 0$ in step 4. In fact the values of not-yet-accepted nodes will be greater than $v_{min}$ in the following iterations.

M2. We perform steps 2-3-4-5 every $p > 1$ executions of the main cycle. It seems that $p = p(N) = \sqrt{N}/2$ is a good choice in most cases ($N$ being the total number of nodes and the dimension of the problem being 2). This leads to a larger buffer but to a faster algorithm.

M3. It is not really needed to store a second matrix to perform intermediate computations described in steps 3 and 4. Saving the values of narrow band and buffer nodes in the same dynamic lists which contain their indexes, we can modify the full matrix and then easily restore the old (right) values. This leads to a faster algorithm and a gain in memory requirement.

M4. In step 5 we accept the nodes such that their variation is smaller than a given quantity $\varepsilon$.

M5. In steps 3 and 4 it is not needed to be very accurate because we are just interested if the values in *BUF* change or not. So we iterate the computation for each node $x_{i,j}$ until

$$|w^{(k+1)}(x_{i,j}) - w^{(k)}(x_{i,j})| < \varepsilon', \quad k = 0, 1, \dots$$

M6. The step 3 is completely skipped because the far zone already plays the role of a moving boundary condition with values $v = 1$. Moreover, in step 2 all the nodes in *BUF* are computed just once.

*Remark* **3.2** Modifications M1, M2 and M3 does not modify the solution with respect to the ideal algorithm. They are just tricks to speed-up the convergence. Modifications M4 and M5 produce an error in the solution which is expected to vanish as $\varepsilon$ and $\varepsilon'$ tend to zero (as confirmed by numerical tests). Modification M6 seems to not affect the solution, but this is just an experimental evidence.

*Remark* **3.3** (*BFM as generalization of FM*) Let us consider the case of the eikonal equation (1) (isotropic front propagation) and include in the algorithm the "natural" modification M1, but not M2. Then the buffer's size is never greater than one and the BFM method changes back into the FM method. To prove this, let us define the node $A$ as in step 1. We easily note that it is not possible that the value $v(A)$ changes in step 2. In fact, if it changes, it means that it can be still improved, and this is in contradiction with the prove of convergence of the FM method given in [8]. We can conclude by contradiction as follows. If the node $A$ inserted in *BUF* in step 1 does not exit *BUF* in step 5, then the value $v(A)$ has changed in step 3 or step 4 modified. This implies that the value $v(A)$ was influenced by values greater or equal than itself, and this is not possible for the eikonal equation.

### 3.3 Properties of the BFM

To prove the convergence of the algorithm to the viscosity solution of equation (6) we adopt the following strategy. We show that the BFM computes the same solution of the classical iterative algorithm and then we recover convergence and *a priori* estimates by the results already available for that scheme (see f.e. [1] for Hamilton-Jacobi-Bellman equations).

**Proposition 3.1 (Convergence to the viscosity solution)** Assume equation (6) has a unique viscosity solution. Let $\bar{v} : G \to [0,1]$ be its discrete solution computed by the classical iterative (fixed point) scheme based on a convergent numerical scheme and $v : G \to [0,1]$ be the discrete solution of the same equation computed by the BFM method based on the same scheme. Then $\bar{v}(X) = v(X)$ for any node $X \in G$.

*Proof.* By induction on the cycles of the algorithm. Let us denote respectively by $ACC^{(s)}$, $BUF^{(s)}$, $NB^{(s)}$ and $FAR^{(s)}$ the sets of nodes accepted, buffer, narrow band and far at the generic cycle $s$ of the algorithm. Let $A^{(s)} = \arg\min_{X \in NB^{(s)}} \{v(X)\}$ be the node with the minimal value in $NB^{(s)}$.
We will prove that, for any $s \geq 0$,

$$N_{SL}(BUF^{(s)}) \cap FAR^{(s)} = \emptyset \tag{13}$$

and

$$v(X) = \bar{v}(X) \quad \text{for any } X \in ACC^{(s)} \tag{14}$$

and then we easily conclude. For $s = 0$, assertion (13) is clearly true since $BUF^{(0)} = \emptyset$. Also assertion (14) is true because $ACC^{(0)} = \widetilde{\Gamma}_0$ and the values $v(\widetilde{\Gamma}_0)$ and $\bar{v}(\widetilde{\Gamma}_0)$ are imposed by the boundary condition on $\widetilde{\Gamma}_0$.
Going from cycle $s$ to cycle $s+1$, we have (see step 1 and 5)

$$BUF^{(s+1)} = (BUF^{(s)} \cup \{A^{(s)}\}) \setminus (ACC^{(s+1)} \setminus ACC^{(s)})$$

and

$$FAR^{(s+1)} = FAR^{(s)} \setminus (N_{SL}(A^{(s)}) \cap FAR^{(s)}).$$

As a consequence, the only new node in $BUF^{(s+1)}$ is surrounded by nodes not in $FAR^{(s+1)}$. Then we have

$$N_{SL}(BUF^{(s+1)}) \cap FAR^{(s+1)} = \emptyset.$$

In order to show that (14) is true for $s+1$, let us define the numerical boundaries of $BUF^{(s)}$ as follows (see Fig. 2)

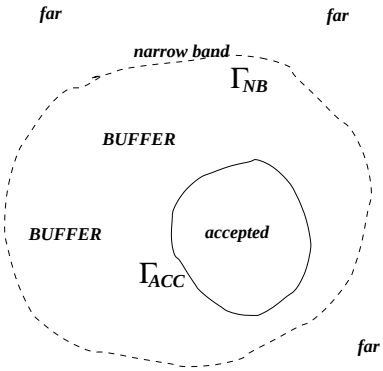


Figure 2: The buffer zone and its boundaries.

$$\Gamma_{ACC}^{(s)} = \{X \in ACC^{(s)} : N_{SL}(X) \cap BUF^{(s)} \neq \emptyset\},$$

$$\Gamma_{NB}^{(s)} = \{X \in NB^{(s)} : N_{SL}(X) \cap BUF^{(s)} \neq \emptyset\}.$$

Clearly we have $\Gamma_{ACC}^{(s)} \cap \Gamma_{NB}^{(s)} = \emptyset$ since $ACC^{(s)} \cap NB^{(s)} = \emptyset$. By (13), the nodes in $BUF^{(s)}$ are surrounded only by nodes in $ACC^{(s)}$ and $NB^{(s)}$ (and $BUF^{(s)}$ itself, of course) so that the values of the nodes in $\Gamma_{ACC}^{(s)}$ and $\Gamma_{NB}^{(s)}$ play the role of two boundary conditions for $BUF^{(s)}$. The algorithm produces three solutions in $BUF^{(s)}$,

$$\begin{array}{ll} v^{(s)} & \text{with } v(\Gamma_{NB}^{(s)}) \text{ and } v(\Gamma_{ACC}^{(s)}) \text{ unchanged,} \\ v_1^{(s)} & \text{with } v(\Gamma_{NB}^{(s)}) = 1 \text{ and } v(\Gamma_{ACC}^{(s)}) \text{ unchanged,} \\ v_0^{(s)} & \text{with } v(\Gamma_{NB}^{(s)}) = 0 \text{ and } v(\Gamma_{ACC}^{(s)}) \text{ unchanged.} \end{array}$$

and we have

$$ACC^{(s+1)} = ACC^{(s)} \cup \{X \in BUF^{(s)} : v^{(s)}(X) = v_1^{(s)}(X) = v_0^{(s)}(X)\}.$$

By (14) we know that $v(\Gamma_{ACC}^{(s)}) = \bar{v}(\Gamma_{ACC}^{(s)})$ while any boundary condition on $\Gamma_{NB}^{(s)}$ has not influence for the new accepted nodes. As a consequence, the iterative algorithm and the BFM method must compute the same solution in $ACC^{(s+1)} \backslash ACC^{(s)}$ and then $v(ACC^{(s+1)}) = \bar{v}(ACC^{(s+1)})$. ■

## 3.4 Some considerations on the computational cost

We always denote by $N$ the total number of nodes in the grid. We assume for simplicity that we are working on a square grid in dimension 2 so each

dimension has $\sqrt{N}$ nodes.

*Classical iterative method.* The iterative (fixed point) method consists in computing the solution of the equation on every node of the grid until convergence is reached. Since it needs $O(\sqrt{N})$ iterations to converge, its complexity is of order $O(N\sqrt{N})$

*FM method.* The FM method has a complexity of order $O(N \ln N_{NB})$ where $N_{NB}$ is the number of nodes in the narrow band (it varies at each step). $N_{NB}$ is bounded by $N$ but in general it is expected to be of order $\sqrt{N}$ because the front has dimension 1, so its computational cost is $O(N \ln \sqrt{N})$. The term $O(\ln \sqrt{N})$ comes from the need of keeping an order in the list containing the nodes of the narrow band (f.e. by an heap-tree structure), so that it is fast to pick the node with the minimal value at each step. To this end it is important to note that the sequence of the minimal values of the narrow band is in many cases very close to be increasing, this means that a simple insertion sort is not so costly as in the randomly-ordered case. For this reason we did not implement an heap structure to store the nodes but a simple dynamic linked list. By experiments it seems that the factor $O(\ln \sqrt{N})$ is in fact $O(1)$ for two dimensional problems and a relatively small number of nodes (otherwise this is not true, see for example [3]).

*BFM method.* As we already remarked, in the case of isotropic front propagation problems the BFM method changes back into the FM method. So we expect a computational cost of the same order in the best case. In the worse case the buffer becomes larger and larger and we need to solve an iterative problem on the buffer to accept just few nodes or even any node at all. This leads to a computational cost greater than that of the iterative algorithm. Experiments says (see next section) that BFM method behaves like FM method in most cases, although the constant in front of $N \ln \sqrt{N}$ is larger for the BFM method (of course BFM method can deal with more general equations).

# 4 Numerical tests

In this section we perform some tests on equations of the form (9) and (12). The aim is to compare numerical results and CPU times for the classical iterative method, the FM method and the BFM method (with modifications M1-M6). For the classical iterative method we use the Fast Sweeping (FS) technique to speed up the convergence. This technique consists in computing over the grid in four alternate directions (for example from North to South, from South to North, from East to West and from West to East) until convergence is reached (see [25, 20] and references therein for details). The algorithms are implemented in C++ on a PC with a Pentium IV 2.60 GHz processor and 256 MB RAM. In the following we will consider the

solution of the iterative method as the exact solution and we compute the error of the other two methods with respect to that solution. Although this is obviously not true (semi-Lagrangian scheme can produce bad results in some cases due to the numerical diffusion) we can not expect neither FM nor BFM overcome the iterative method since the three methods are based on the same numerical scheme. We compare the methods on $51^2$, $101^2$ and $201^2$ structured grids (the number of nodes is chosen to have a grid node corresponding to the point $(0,0)$).
Computation is done in a square domain Q on a structured grid. As stopping criterion for the iterative Fast Sweeping method we used

$$\|v^{(k+1)} - v^{(k)}\|_\infty < 10^{-16}.$$

The $L^1$ error is defined as

$$E_1 = \frac{1}{|Q|} \iint_Q |T - T^{FS}|$$

where $T$ is the solution computed by FM or BFM and $T^{FS}$ is the solution computed by FS. Note that the difference with respect to the relative error $\widetilde{E}_1 = \frac{1}{|Q|} \iint_Q |T - T^{FS}|/T^{FS}$ is not significant so it will be not reported.

Except for the first test (eikonal equation), the FM method is known to compute an incorrect solution and then *it is not a real alternative to FS and BFM*. Nevertheless we think it is interesting to show the error and the CPU time for the FM method in order to study its robustness with respect to the anisotropy of the problem.

For the BFM, the values of the parameters $\varepsilon$ (see step M4) and $p$ (see step M2) are reported in the tables test by test while the value of $\varepsilon'$ (see step M5) is fixed to $10^{-6}$.

## Test 1: Eikonal equation

In this test we solve the eikonal equation $|\nabla T(x,y)| = 1$ in $[-2,2]^2$ coupled with a Dirichlet boundary condition $T(0,0) = 0$. This equation can be written in the form (9) choosing $f(x,y,a) = a$ and $A = B(0,1) \subset \mathbb{R}^2$. We discretized the unit ball with 16 points equally spaced on the boundary. The level sets of the solution $T(x,y) = x^2 + y^2$ correspond to an isotropic front propagation so the FM method can be used. By Table 1 we can see that the three methods compute the same solution. Here FS method needs just four iterations to reach convergence. Nevertheless, FM is the fastest method. BFM is slower than FM due to the time spent to menage the buffer (even if here the buffer contains only one node at each step). The

Table 1: Errors and CPU times for Test 1

| method | $\Delta x$ | $N$ | $p$ | $\varepsilon$ | $E_1$ | CPU time (sec) | CPU $N \to 4N$ |
|---|---|---|---|---|---|---|---|
| FS | 0.08 | $51^2$ | – | – | – | 0.04 | – |
| BFM | 0.08 | $51^2$ | 25 | $10^{-3}$ | 0 | 0.07 | – |
| FM | 0.08 | $51^2$ | – | – | 0 | 0.02 | – |
| FS | 0.04 | $101^2$ | – | – | – | 0.17 | 4.2 |
| BFM | 0.04 | $101^2$ | 50 | $10^{-3}$ | 0 | 0.27 | 3.8 |
| FM | 0.04 | $101^2$ | – | – | 0 | 0.1 | 5.0 |
| FS | 0.02 | $201^2$ | – | – | – | 0.7 | 4.1 |
| BFM | 0.02 | $201^2$ | 100 | $10^{-3}$ | 0 | 1.12 | 4.1 |
| FM | 0.02 | $201^2$ | – | – | 0 | 0.38 | 3.8 |

last column reports the ratio between the CPU time for a $4N$ grid and a $N$ grid. The value 5.0 for the FM with $N = 101^2$ is not completely correct because the FM with $N = 51^2$ is too fast to be precisely measured.

## Test 2: Eikonal equation on a manifold

In this test we solve an anisotropic front propagation problem, choosing in (9)

$$f(x, y, a_1, a_2) = \frac{(a_1, a_2)}{\sqrt{1 + (5a_1 + 5a_2)^2}}, \quad (a_1, a_2) \in B(0, 1) \subset \mathbb{R}^2$$

and Dirichlet boundary condition $v(0, 0) = 0$. This choice corresponds to solving the eikonal equation on the plane $z = 5x + 5y$ (see [23]). The unit ball is discretized in 16 points and $Q$ is again $[-2, 2]^2$. In Fig. 3 we show the exact solution and the solution computed by the FM method. As shown in [23], the FM method is not able to compute the right solution even if we increase the number of nodes.
By Table 2 and Fig. 4 we see that the behavior of BFM is quite good since it preserves the order of the scheme with a significant difference in CPU time.
Note that the error of BFM does not converge to zero as the grid is refined because we used a fixed $\varepsilon$ in all cases.

## Test 3: Lunar landing

In this test we solve equation (9) with

$$\begin{cases} f_1(x, y, a) = y \\ f_2(x, y, a) = a \end{cases}$$

Table 2: Errors and CPU times for Test 2

| method | $\Delta x$ | $N$ | $p$ | $\varepsilon$ | $E_1$ | CPU time (sec) | CPU $N \to 4N$ |
|---|---|---|---|---|---|---|---|
| FS | 0.08 | $51^2$ | – | – | – | 0.37 | – |
| BFM | 0.08 | $51^2$ | 25 | $10^{-3}$ | 0.02 | 0.09 | – |
| FM | 0.08 | $51^2$ | – | – | 0.87 | 0.02 | – |
| FS | 0.04 | $101^2$ | – | – | – | 2.49 | 6.7 |
| BFM | 0.04 | $101^2$ | 50 | $10^{-3}$ | 0.01 | 0.45 | 5.0 |
| FM | 0.04 | $101^2$ | – | – | 1.02 | 0.09 | 4.5 |
| FS | 0.02 | $201^2$ | – | – | – | 13.55 | 5.4 |
| BFM | 0.02 | $201^2$ | 70 | $10^{-3}$ | 0.02 | 1.67 | 3.7 |
| FM | 0.02 | $201^2$ | – | – | 1.01 | 0.4 | 4.4 |

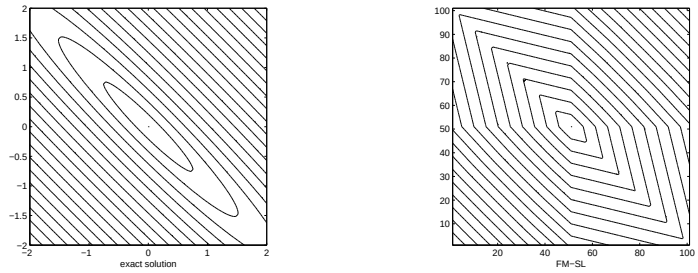


Figure 3: Test 2: exact solution (left) and solution computed by FM method (right), $101 \times 101$ grid.

and Dirichlet boundary condition $v(0,0) = 0$. We chose $A = \{-1, 1\}$. This test correspond to the classical one-dimensional minimum time problem in which the dynamics is $\ddot{x} = a$ and $a$ can be chosen in $\{-1, 1\}$. This is a difficult test because of the strong mutual dependency of nodes. Moreover, the effect of boundary condition is very strong so we decided to perform computation on the domain $[-5, 5]^2$ and to analyze the results on the subdomain $[-2, 2]^2$. Results are shown in Table 3.
As the grid size increases, we need to decrease the constant $\varepsilon$ in order to maintain the same order in the error. Not surprisingly, the FM computes a vary bad solution.

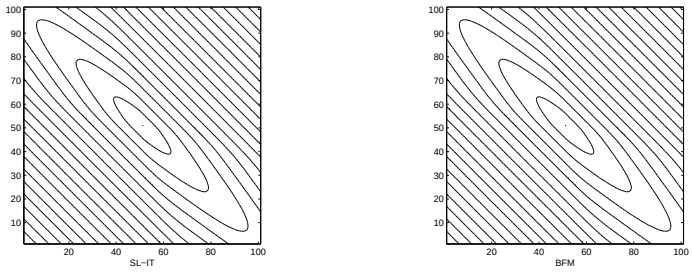


Figure 4: Test 2: solution computed by FS method (left) and BFM method (right).

Table 3: Errors and CPU times for Test 3

| method | $\Delta x$ | $N$ | $p$ | $\varepsilon$ | $E_1$ | CPU time (sec) |
|---|---|---|---|---|---|---|
| FS | 0.2 | $51^2$ | – | – | – | 0.13 |
| BFM | 0.2 | $51^2$ | 12 | $10^{-3}$ | 0.07 | 0.01 |
| FM | 0.2 | $51^2$ | – | – | 1.54 | 0 |
| FS | 0.1 | $101^2$ | – | – | – | 0.67 |
| BFM | 0.1 | $101^2$ | 25 | $10^{-4}$ | 0.07 | 0.15 |
| FM | 0.1 | $101^2$ | – | – | 3.21 | 0.02 |
| FS | 0.05 | $201^2$ | – | – | – | 3.91 |
| BFM | 0.05 | $201^2$ | 50 | $10^{-5}$ | 0.05 | 2.05 |
| FM | 0.05 | $201^2$ | – | – | 6.11 | 0.11 |

## Test 4: Tag-Chase game with state constraints

In this test we solve equation (12) with

$$\begin{cases} f_1(x,y,a) = V_A a \\ f_2(x,y,b) = V_B b \end{cases}$$

where $a, b \in \{-1, 0, 1\}$. This test models the one-dimensional Tag-Chase game where the two players $A$ and $B$ are constrained to run in the segment $[-2, 2]$. The game is set in $Q = [-2, 2]^2 \subset \mathbb{R}^2$. The velocities $V_A$ for the pursuer and $V_B$ for the evader are constant. We choose $V_A = 2$ and $V_B = 1$. The axis of abscissas represents the coordinate $x_A$ of the Pursuer and the axis of ordinate represents the coordinate $x_B$ of the Evader. The target is $\mathcal{T} = \{(x_A, x_B) : x_A = x_B\}$ that is the set of point where the capture occurs (see [12, 10, 7] for more details on the model and recent results on

differential games with state constraints).
In Fig. 5 we show the exact solution with one optimal trajectory starting from the point $(-1.5, 0)$. We show the result only in half domain due to the symmetry of the solution. Clearly in this case characteristics and gradient lines does not lie on the same simplex so the FM fails. In Fig. 6 we show the solution computed by FS (left) and by BFM (right). We can see very well the effect of numerical diffusion due to the scheme but again the two solutions are very similar.

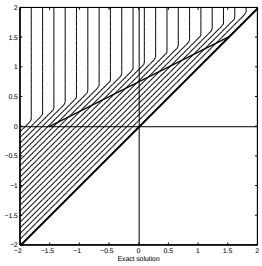


Figure 5: Test 4: exact solution with an optimal trajectory starting from $(-1.5, 0)$.

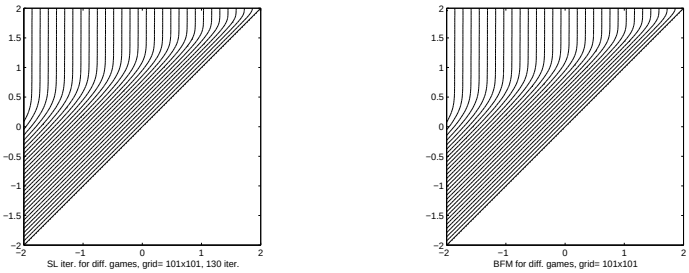


Figure 6: Test 4: solution computed by FS method (left) and BFM method (right).

## FM and BFM methods vs. Fast Sweeping method

In this paper we used the Fast Sweeping technique to compute the solution of the classical iterative scheme (11) because it is in general fast and robust and it is proved to converge to the fixed point. The other advantage

is that it is not restricted to isotropic front propagation problems like FM method. Unfortunately it is very difficult to estimate the number of sweepings needed to reach convergence in the case of a general velocity field but experiments say that FS method is much faster than the classical iterative method where the nodes are visited in only one fixed order.
Comparing FM/BFM and FS methods is not an easy task because their behavior is very case-dependent. For example, test 2 was chosen to be "difficult" for FM/BFM methods because of the strong anisotropy. On the other hand, the same test is not so difficult for FS method because the characteristics directions are straight lines to the origin and few sweepings are enough to compute a good approximation of the viscosity solution. In test 2 the FS method is faster than BFM method allowing the same $L^1$ distance from the exact solution.
The result is reversed in the test described in Fig. 7. The choice of the velocity field corresponds to an isotropic front propagation problem in presence of obstacles. The FS method is slower than BFM method which is slower than FM method.

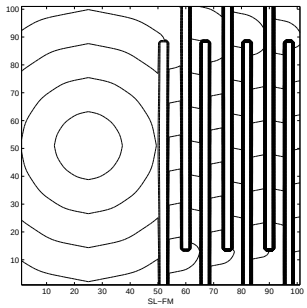


Figure 7: A difficult test for the FS method. The front moves in normal direction with speed 1. The rectangles represent obstacles.

# Conclusions

In this paper we introduced a new fast method to solve Hamilton-Jacobi equations modeling a monotone front propagation problem, including Hamilton-Jacobi-Bellman and Hamilton-Jacobi-Isaacs equations related to optimal control problems and differential games. Although it does not compute exactly the same solution of the standard iterative (fixed point) method based on the same first order semi-Lagrangian scheme, the new method is able to compute a good approximation of the viscosity solution preserving the

order of the scheme. By the experiments, it seems that the computational cost is close to $O(N)$ as for the FM method, at least for two dimensional problems.

## Acknowledgment

The author wishes to thank Maurizio Falcone, Frederic Bonnans, Hasnaa Zidani, Olivier Bokanowski, and Nicolas Forcadel for the useful discussions and suggestions.

## References

[1] Bardi, M., and Capuzzo Dolcetta, I. (1997). *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*, Birkhäuser, Boston.

[2] Bardi, M., and Falcone, M. (1990). *An approximation scheme for the minimum time function*, SIAM J. Control Optim., 28, pp. 950–965.

[3] Bokanowski, O., Cristiani, E., and Zidani, H. (2008). *An efficient data structure to solve front propagation problems*, submitted to Journal of Scientific Computing.

[4] Carlini, E., Cristiani, E., and Forcadel, N. (2006). *A non-monotone Fast Marching scheme for a Hamilton-Jacobi equation modelling dislocation dynamics*, in A. Bermdez de Castro, D. Gmez, P. Quintela, P. Salgado (eds.), Numerical Mathematics and Advanced Applications, Proceedings of ENUMATH 2005 (Santiago de Compostela, Spain, July 2005), pp. 723–731, Springer.

[5] Carlini, E., Falcone, M., Forcadel, N., and Monneau, R. (2008). *Convergence of a Generalized Fast Marching Method for an eikonal equation with a velocity changing sign*, SIAM J. Numer. Anal., 46, pp. 2920–2952.

[6] Cristiani, E. (2007). *Fast Marching and Semi-Lagrangian Methods for Hamilton-Jacobi Equations with Applications*, Ph.D. thesis, Dipartimento di Metodi e Modelli Matematici per le Scienze Applicate, SAPIENZA - Università di Roma, Rome, Italy.

[7] Cristiani, E., and Falcone, M. (2008). *Numerical solution of the Isaacs equation for differential games with state constraints*, Proceedings of 17th IFAC World Congress (Seoul, Korea, July 6-11, 2008).

[8] Cristiani, E., and Falcone, M. (2007). *Fast semi-Lagrangian schemes for the Eikonal equation and applications*, SIAM J. Numer. Anal., 45, pp. 1979–2011.

[9] Cristiani, E., and Falcone, M. (2008). *A characteristics driven Fast Marching method for the Eikonal equation*, in K. Kunisch, G. Of, O. Steinbach (eds.), Numerical Mathematics and Advanced Applications (ENUMATH 2007, Graz, Austria, September 10-14, 2007), pp. 695–702, Springer Berlin Heidelberg.

[10] Cristiani, E., and Falcone, M., *Fully-discrete schemes for the value function of Pursuit-Evasion games with state constraints*, to appear in Annals of International Society of Dynamic Games, vol. 10, pp. 177–206.

[11] Falcone, M. (1994). *The minimum time problem and its applications to front propagation*, in Motion by Mean Curvature and Related Topics, A. Visintin and G. Buttazzo, eds., de Gruyter, Berlin, pp. 70–88.

[12] Falcone, M. (2006). *Numerical methods for differential games based on partial differential equations*, International Game Theory Review, 8, pp. 231–272.

[13] Falcone, M., Giorgi, T., and Loreti, P. (1994). *Level sets of viscosity solutions: Some applications to fronts and rendez-vous problems*, SIAM J. Appl. Math., 54, pp. 1335–1354.

[14] Gremaud, P. A., and Kuster, C. M. (2006). *Computational study of fast methods for the eikonal equation*, SIAM J. Sci. Comput., 27, pp. 1803–1816.

[15] Hysing, S.-R., and Turek, S., *The eikonal equation: numerical efficiency vs. algorithmic complexity on quadrilateral grids*, in Proceedings of ALGORITMY 2005, pp. 22–31.

[16] Kim, S. (2001). *An $\mathcal{O}(N)$ level set method for eikonal equations*, SIAM J. Sci. Comput., 22, pp. 2178–2193.

[17] Kimmel, R., and Sethian, J. A. (1998). *Computing geodesic paths on manifold*, Proc. Natl. Acad. Sci. USA, 95, pp. 8431–8435.

[18] Kimmel, R., and Sethian, J. A. (2001). *Optimal algorithm for shape from shading and path planning*, J. Math. Imaging Vision, 14, pp. 237–244.

[19] Prados, E., and Soatto, S., *Fast marching method for generic shape from shading*, in Proceedings of VLSM 2005, pp. 320–331.

[20] Qian, J., Zhang, Y.-T., and Zhao, H.-K. (2007). *A fast sweeping method for static convex Hamilton-Jacobi equations*, J. Sci. Comput., 31, pp. 237–271.

[21] Sethian, J. A. (1999). *Level Set Methods and Fast Marching Methods. Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, Cambridge, UK.

[22] Sethian, J. A. (1996). *A fast marching level set method for monotonically advancing fronts*, Proc. Natl. Acad. Sci. USA, 93, pp. 1591–1595.

[23] Sethian, J. A., and Vladimirsky, A. (2003). *Ordered upwind methods for static Hamilton–Jacobi equations: Theory and algorithms*, SIAM J. Numer. Anal., 41, pp. 325–363.

[24] Spira, A., and Kimmel, R. (2004). *An efficient solution to the eikonal equation on parametric manifolds*, Interfaces Free Bound., 6, pp. 315–327.

[25] Tsai, Y.-H. R., Cheng, L.-T., Osher, S., and Zhao, H.-K. (2003). *Fast sweeping algorithms for a class of Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 41, pp. 673–694.

[26] Tsitsiklis, J. N. (1995). *Efficient algorithms for globally optimal trajectories*, IEEE Trans. Automat. Control, 40, pp. 1528–1538.

[27] Vladimirsky, A. (2006). *Static PDEs for time-dependent control problems*, Interfaces and Free Boundaries, 8, pp. 281–300.