The Generalized Fast Marching Method and applications to image segmentation

Nicolas Forcadel[1]

January 20, 2017

Abstract

In [3, 6], the authors have proposed a generalization of the classical Fast Marching Method of Sethian for the eikonal equation in the case where the normal velocity depends on space and time and can change sign. This methods have been applied in [7] to the problem of image segmentation. The goal of this note is to explain this method and its application.Some numerical tests are also proposed.

AMS Classification: 65M06, 65M12, 49L25 .

Keywords: Eikonal equation, fast marching scheme, monotone scheme, convergence.

1 Introduction

A very popular method to describe the evolution of a front is the Level Sets method (see the seminal paper by Osher and Sethian [10] as well as the books [13, 14], [9]), where the front is represented by the zero level set of a continuous function u which solves an Hamilton-Jacobi equation. This function u is compute by a discretization of the equation using a finite difference method with a CFL condition of the type $\Delta t\|c\|_\infty \leq \Delta x$ for explicit schemes, where Δx is the space step and Δt is the time step.

When the velocity c only depends on x and it is constant in sign (positive or negative) the evolution of the front is monotone (increasing or decreasing, respectively) and the problem can be solved via an associated stationary problem corresponding to a generalized eikonal equation (see [5] for more details and for the relations with the minimum time problem). Once the problem is reduced to the stationary eikonal equation we can use the Fast Marching Method (FMM) (see Sethian [12, 14]), where the unknown of the problem is the time $T(x)$ the front reaches the point x. This method works for non negative (non positive) velocities and provides a very efficient scheme which concentrates the computational effort on a neighbourhood of the front. If c cannot change sign we have a monotone (increasing or decreasing) evolution and the front passes just one time on every point of the computational domain. The corresponding arrival time of the front is univalued so that the evolutive problem reduces to a stationary problem (the eikonal equation). Note that in this method, there is no time step, because the time is itself the unknown of the problem so that the original evolutive problem reduces to a stationary problem as remarked in [5] and [8], *i.e.*

$$|\nabla T| = \frac{1}{c(x)}. \tag{1.1}$$

Very recently, the method has been extended to the case of general velocities $c(x,t)$ without sign restrictions (see [3, 6]). In this case, the evolution is not necessarily monotone and the time of arrival of the front can be multivalued.

In [7], the authors have proposed a segmentation method based on the GFMM to detect and visualize the boundaries of the objects contained in an image. The main advantags of this model is that it is fast, efficient, and robust as well as the well-known FMM. Moreover, the initialization step of the GFMM is straightforward and flexible: the initial contour can be taken outside or inside the contour to be detected.

[1]CEREMADE, UMR CNRS 7534, université Paris-Dauphine, Place de Lattre de Tassigny, 75775 Paris Cedex 16, France

The goal of this note is to present the basic ideas of the FMM as well as the Generalized Fast Marching Method (GFMM) and to show how to adapt this method to image segmentation. All the details of the results presented here can be found in [3, 6, 7].

2 The classical Fast Marching Method

2.1 The level set formulation

Let us consider a curve Γ_t (t represents the time) moving according to its normal (see Figure 1).

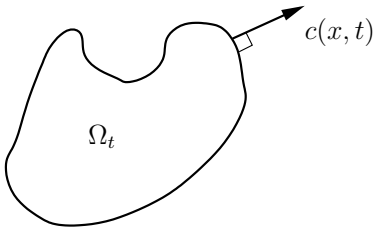

Figure 1: Front propagations in the normal direction

This means that

$$\frac{\partial \Gamma_t}{\partial t} = c(x) \cdot \vec{n}_{x,t} \tag{2.2}$$

where $\vec{n}_{x,t}$ is the unit outward normal vector to the curve Γ at point (x,t). The basic idea of the level set methods is to represent the curve Γ_t by the zero level set of a continuous function u (we assume that $u(\cdot,t) > 0$ in Ω_t and $u(\cdot,t) < 0$ in Ω_t^c), i.e.

$$\Gamma_t = \{x,\ u(x,t) = 0\}.$$

Very formally, using the fact that

$$u(\Gamma_t, t) = 0,$$

we deduce that

$$u_t + \frac{\partial \Gamma_t}{\partial t} \cdot \nabla u = 0.$$

Using (2.2) and the well-known fact that $\vec{n}_{x,t} = -\frac{\nabla u(x,t)}{|\nabla u(x,t)|}$, we finally deduce that u solve the following Hamilton-Jacobi equation

$$u_t = c(x)|\nabla u| \tag{2.3}$$

This computation is formal but it can be make completely rigorous (see for instance Barles [2])

2.2 The minimal-time problem and the first algorithm

The basic idea of the FMM is to see the evolution problem as a stationary problem and more precisely as minimum-time problem (see Falcone [5]). More precisely, we will search stationary solution of Equation (2.3) of the form

$$u(x,t) = T(x) + t.$$

The function T represents the arrival time of the front at the point x (see Figure 2) and solves the eikonal equation

$$|\nabla T| c(x) = 1$$

with $T = 0$ on Γ_0.

The goal is now to compute numerically the arrival time function T. We will use a monotone discretization of the gradient, proposed by Rouy and Tourin [11]. Let us denote by $T_{i,j}$ the value of the approximation of

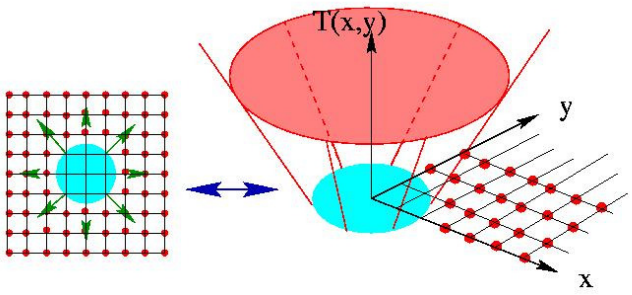

Figure 2: The stationary approach

the function T at the point $(x_i, x_j) = (i\Delta x, j\Delta x)$, where $\Delta x > 0$ is the space step. The approximation $T_{i,j}$ of T is then assume to solve the following scheme:

$$\left(\max\left(T_{i,j} - T_{i+1,j}, T_{i,j} - T_{i-1,j}, 0\right)^2 + \max\left(T_{i,j} - T_{i,j+1}, T_{i,j} - T_{i,j-1}, 0\right)^2\right) = \frac{\Delta x}{c(x_i)}. \tag{2.4}$$

The usual way to solve (2.4) is to use an iterative scheme and to wait for the convergence. In general, this can be very long. The key idea of the Fast Marching is in fact to compute the value of $T_{i,j}$ in such an order so that one can compute the solution of (2.4) in only one iteration. This special order corresponds to the increasing value of $T_{i,j}$.

To do this, the idea of Sethian is to define three regions (see Figure 3):

1. The *frozen* points : this set is composed of all the points already crossed by the front and for which we know the value of T.
2. The *Narrow Band* (NB) : this set is composed of all the point not already frozen and which have a neighbour which is frozen. This is in fact the points that can be immediately reached by the front. In fact, we will solve (2.4) only for this points.
3. The *Far Away* points : this is the other points, that can not be reached immediately by the front.

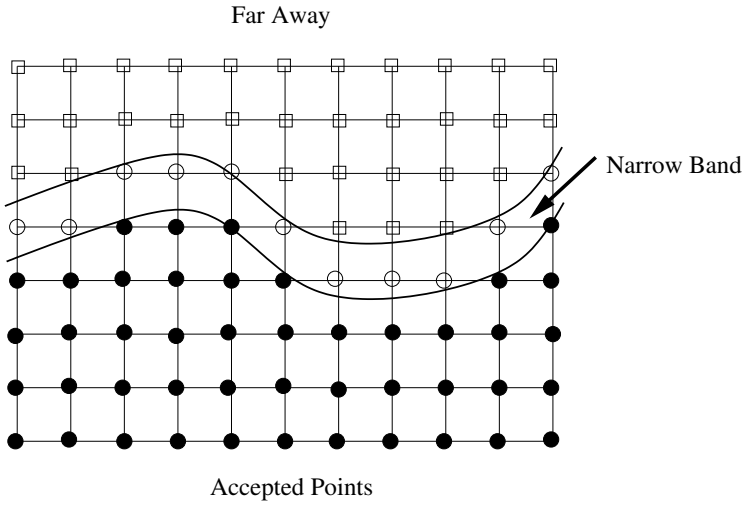

Figure 3: Illustration of the Narrow Band

For (i, j) in the Narrow Band, to compute the value of $T_{i,j}$, the idea is to solve (2.4) using only the value of T of the frozen points (i.e. the points for which we effectively know the value of T). The algorithm is now very simple:
Initialization:
$T_{i,j} = 0\ \forall (i, j)$ such that $(i\Delta x, j\Delta x) \in \Omega_0$.
Loop:

1. For all (i,j) in the Narrow Band, we compute $T_{i,j}$ by solving (2.4).
2. The point(s) (i,j) of the Narrow Band having the smallest value for $T_{i,j}$ is(are) frozen.
3. The Narrow Band is redefined as the boundary of the new frozen region.

In fact, in this scheme, the time is given implicitly by the algorithm. At step n of the algorithm, we define the time t_n as the minimal value of $T_{i,j}$ for (i,j) in the Narrow Band.

2.3 Rewriting of the classical FMM

The notion of Frozen points is not adapted to the case of changing-sign velocity, because a "frozen" point can be crossed again by the front. To overcome this difficulty, Carlini et al. proposed in [3] to introduce a field θ to represent the front. More precisely, they consider a field θ such that

$$\begin{cases} \theta = 1 & \text{in the interior} \\ \theta = -1 & \text{in the exterior} \end{cases}$$

and the front is in fact represented by the discontinuity of θ. Before to give the algorithm of the GFMM, let us rewrite the classical FMM using this setting.

We begin by some definitions.

Definition 2.1 *We define the neighbourhood of a point $I = (i_1, i_2)$ by*

$$V(I) = \{J \in \mathbb{Z}^2,\ |I - J| \leq 1\} \backslash \{I\}$$

Now let us rewrite the definition of the frozen points and of the Narrow Band:

Definition 2.2 *The frozen region at step n is given by*

$$\mathrm{Fr}^n = \{I,\ \theta_I^n = 1\}.$$

The Narrow Band at step n is defined as the neighbourhood of the frozen point, i.e.

$$NB^n = \{I,\ \exists\, J \in V(I), \theta_J^n = -\theta_I^n = 1\}$$

To simplify the presentation of the algorithm, we need to introduce, for $I \in NB^n$, the set of useful points, i.e., the set of points which will be used to compute the tentative value T_I:

Definition 2.3 *Let $I \in NB^n$. We define the set of useful points for I at step n by*

$$\mathcal{U}^n(I) = \{J \in V(I),\ \theta_J^n = 1\}.$$

We also define the set of useful points at step n by

$$\mathcal{U}^n = \cup_{I \in NB^n} \mathcal{U}^n(I).$$

We now rewrite the FMM algorithm:

Initialization

1. Set $n = 1$, $t_0 = 0$
2. *Initialize the field θ^0* as
   $\theta_I^0 = \begin{cases} 1 & \text{for } x_I \in \Omega_0 \\ -1 & \text{elsewhere} \end{cases}$
3. *Initialize the time for points I*
   $$u_I^0 = \begin{cases} 0 & \text{if } I \in \mathcal{U}^0 \\ +\infty & \text{otherwise} \end{cases}$$

Loop

4. *Compute $\tilde{u}^{n-1}$ on NB^{n-1}*

   Let $I \in NB^{n-1}$ ($I = (i_1, i_2)$), then we compute $\tilde{u}_I^{n-1}$ as the solution of the following second order equation:

   (2.5)
   $$\left(\max\left(0, \tilde{u}_I^{n-1} - u_{i_1+1,i_2}^{n-1}, \tilde{u}_I^{n-1} - u_{i_1-1,i_2}^{n-1}\right)^2 + \max\left(0, \tilde{u}_I^{n-1} - u_{i_1,i_2+1}^{n-1}, \tilde{u}_I^{n-1} - u_{i_1,i_2-1}^{n-1}\right)^2\right) = \frac{(\Delta x)^2}{|c_I|^2},$$

5. $t_n = \inf_{I \in NB^{n-1}} \tilde{u}_I^{n-1}$

6. *Initialize the new accepted points*
   $NA^n = \{I,\ \tilde{u}_I^{n-1} = t_n\}$

7. *Reinitialize θ^n*
   $$\theta_I^n = \begin{cases} 1 & \text{if } I \in NA^n \\ \theta_I^{n-1} & \text{otherwise} \end{cases}$$

8. *Reinitialize u_I^n*
   $$u_I^n = \begin{cases} t_n & \text{if } I \in NA^n \text{ and } I \in \mathcal{U}^n \\ u_I^{n-1} & \text{if } I \in \mathcal{U}^{n-1} \backslash NA^n \text{ and } I \in \mathcal{U}^n \\ +\infty & \text{otherwise} \end{cases}$$

9. Set $n := n + 1$ and go to 4

Let us now give a first convergence result concerning the classical FMM. We make the following assumptions:
(A) The velocity $c \in W^{1,\infty}(\mathbb{R}^N)$, for some constant $L > 0$ we have $|c(x') - c(x)| \leq L(|x' - x|)$, and Ω_0 is a C^2 open set, with bounded boundary $\partial\Omega_0$.

We define
$$\theta^\varepsilon(x,t) = \theta_I^n \text{ if } x \in [x_I, x_I + \Delta x[,\ t \in [t_n, t_{n+1}[.$$
We also have to define the half-relaxed limits of θ^ε:

$$\overline{\theta}^0(x,t) = \limsup_{\varepsilon \to 0, y \to x, s \to t} \theta^\varepsilon(y,s), \quad \underline{\theta}^0(x,t) = \liminf_{\varepsilon \to 0, y \to x, s \to t} \theta^\varepsilon(y,s). \tag{2.6}$$

Theorem 2.4 (Convergence Result)
Under assumption (A), $\overline{\theta}^0$ is the unique upper semi-continuous solution of

$$\begin{cases} \theta_t = c(x)|\nabla\theta| & \text{in} \quad \mathbb{R}^N \times (0, +\infty), \\ \theta(\cdot, 0) = 1_{\Omega_0} - 1_{\Omega_0^c} & \text{on} \quad \mathbb{R}^N, \end{cases} \tag{2.7}$$

Let us remark that a slight modification of the algorithm allows us to treat the case $c = c(x,t) > 0$. Indeed, at step n, we know the physical time t_{n+1}. So, for $I \in NB^{n-1}$, we can compute the tentative value $\tilde{u}_I^{n-1}$ in the following way

$$\left(\max\left(0, \tilde{u}_I^{n-1} - u_{i_1+1,i_2}^{n-1}, \tilde{u}_I^{n-1} - u_{i_1-1,i_2}^{n-1}\right)^2 + \max\left(0, \tilde{u}_I^{n-1} - u_{i_1,i_2+1}^{n-1}, \tilde{u}_I^{n-1} - u_{i_1,i_2-1}^{n-1}\right)^2\right) = \frac{(\Delta x)^2}{|c_I^{n-1}|^2},$$

The only difficulty in this case is that, if we do not change the step 5 of the algorithm, we can have $t_n < t_{n-1}$ or $t_n \gg t_{n-1}$. The first case, should be avoid because $(t_n)_n$ represents the physical time and is then expected to be a non-decreasing sequence. The second case should also be avoid because we want to control the time step $t_n - t_{n-1}$ in order to capture the oscillations in time of the velocity. In order to avoid this two situations, we introduce a time step $\Delta t > 0$ (which is independent on Δx) such that we will impose

$$0 \leq t_n - t_{n-1} \leq \Delta t.$$

To ensure this property, we replace Step 5 of the algorithm by

5. $\tilde{t}_n = \inf_{I \in NB^{n-1}} \tilde{u}_I^{n-1}$
   Truncate $\tilde{t}_n$: $t_n = \max(t_{n-1}, \min\{\tilde{t}_n, t_{n-1} + \Delta t\})$
   If $t_n = t_{n-1} + \Delta t$ and $t_n < \tilde{t}_n$ go to 4 with $n := n+1$, $\theta^n = \theta^{n-1}$ and $u^n = u^{n-1}$.

In particular, if $t_n \gg t_{n-1}$, we do not accept any point and we just make advance the time. We also need to change the definition of the new accepted point in the following way:

6. *Initialize the new accepted points*
   $NA^n = \{I,\ \tilde{u}_I^{n-1} = \tilde{t}_n\}$

3 The Generalized Fast Marching Method

Before to give the details of the GFMM, let us show by an example in dimension $N = 1$ that we have to "regularise" in space the numerical velocity. The velocity we will use in this example is not Lipschitz, but, with a slight modification, it is always possible to replace this velocity by a Lipschitz one which is very close in the L^∞ norm. Then the discrete solution is unchanged.
Consider the speed

$$c(x) = \begin{cases} -\delta & \text{if } x < x_I \\ \delta & \text{if } x \geq x_I \end{cases},$$

as plotted in Fig.4.

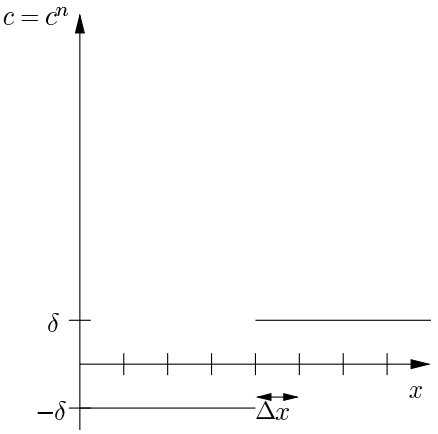

Figure 4: The velocity c^n.

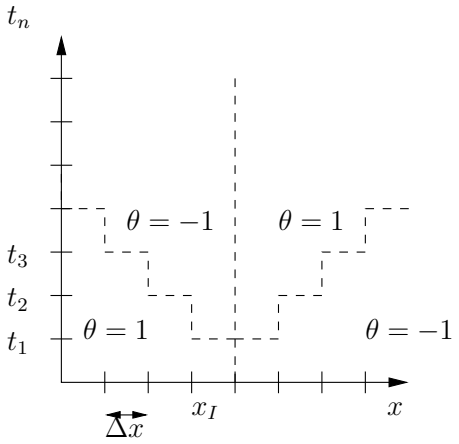

Figure 5: Evolution with the velocity c^n.

Suppose $\frac{\Delta x}{\delta} = \Delta t$, and $\theta_J^0 = 1$ for $J \leq I$, $\theta_J^0 = -1$ for $J > I$. Then the nodes I, $I+1$ will be accepted at the iteration 1, with $t_1 = \Delta t$ and the front will duplicate, see Fig. 5.

This is the reason why we add a band of zero to separate the zone where the velocity is positive from the zone where it is negative. This is done in the following definition:

Definition 3.1 *Given the speed $c_I^n \equiv c(x_I, t_n)$ we define the function*

$$\widehat{c}_I^n \equiv \begin{cases} 0 & \text{if there exists } J \in V(I) \text{ such that } (c_I^n c_J^n < 0 \text{ and } |c_I^n| \leq |c_J^n|), \\ c_I^n & \text{otherwise.} \end{cases}$$

3.1 The GFMM algorithm

We now describe the GFMM algorithm for unsigned velocity.

As in the classical FMM, we define the Narrow Band (NB) and the useful points. The schematic representation is the following one:

The Narrow Band consists on the points $I \in \mathbb{Z}^2$ that can be immediately reached by the front:

$$NB^n = \{I \in \mathbb{Z}^2,\ \exists\, J \in V(I), \theta_I^n = -\theta_J^n \text{ and } \theta_I^n \hat{c}_I^n < 0\}, \quad NB_\pm^n = NB^n \cap \{I, \theta_I^n = \pm 1\}.$$

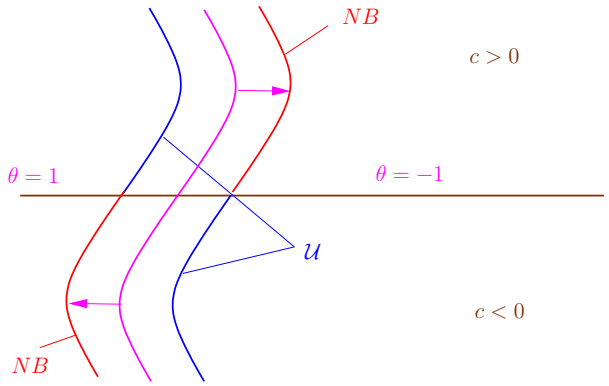

Figure 6: Schematic representation of the Narrow Band and of the useful points

As in the FMM, for all $I \in NB^n$, we have to compute a tentative value (which we denote $\tilde{u}_I^n$ in the sequel) of the arrival time of the front at point I. To compute this tentative value, we define the points that are useful for I, i.e. that we will use in the computation of $\tilde{u}_I^n$:

$$\mathcal{U}^n(I) = \{J \in V(I), \theta_I^n = -\theta_J^n\}, \quad \mathcal{U}^n = \cup_{I \in NB^n} \mathcal{U}^n(I).$$

To compute the tentative value $\tilde{u}_I^n$, we will only use the value of the points that are really useful for I. In particular, we will not use the value of a point $J \in V(I) \cap \mathcal{U}^n$ if $\theta_J^n = \theta_I^n$, that is if $J \notin \mathcal{U}^n(I)$. This is the reason why we introduce

$$u_{J \to I}^n = \begin{cases} u_J^n & \text{if } J \in \mathcal{U}^n(I) \\ +\infty & \text{otherwise} \end{cases}$$

The algorithm is now very similar to the classical Fast Marching Method:

Loop

4. *Compute $\tilde{u}^{n-1}$ on NB^{n-1}*

   Let $I \in NB^{n-1}$ ($I = (i_1, i_2)$), then we compute $\tilde{u}_I^{n-1}$ as the solution of the following second order equation:

$$\begin{cases} & \max\left(0, \tilde{u}_I^{n-1} - u_{(i_1+1,i_2)\to I}^{n-1}, \tilde{u}_I^{n-1} - u_{(i_1-1,i_2)\to I}^{n-1}\right)^2 \\ + & \\ & \max\left(0, \tilde{u}_I^{n-1} - u_{(i_1,i_2+1)\to I}^{n-1}, \tilde{u}_I^{n-1} - u_{(i_1,i_2-1)\to I}^{n-1}\right)^2 \end{cases} = \frac{(\Delta x)^2}{|\widehat{c}_I^{n-1}|^2}$$

5. $\tilde{t}_n = \inf_{I \in NB^{n-1}} \tilde{u}_I^{n-1}$

   Truncate $\tilde{t}_n$: $t_n = \max(t_{n-1}, \min\{\tilde{t}_n, t_{n-1} + \Delta t\})$

   If $t_n = t_{n-1} + \Delta t$ and $t_n < \tilde{t}_n$ go to 4 with $n := n+1$, $\theta^n = \theta^{n-1}$ and $u^n = u^{n-1}$.

6. *Initialize the new accepted points*

   $NA^n = \{I,\ \tilde{u}_I^{n-1} = \tilde{t}_n\}$

7. *Reinitialize θ^n*

$$\theta_I^n = \begin{cases} -\theta_I^{n-1} & \text{for } I \in NA^n \\ \theta_I^{n-1} & \text{otherwise} \end{cases}$$

8. *Reinitialize u_I^n*

$$u_I^n = \begin{cases} t_n & \text{if } I \in NA^n \text{ and } I \in \mathcal{U}^n \\ u_I^{n-1} & \text{if } I \in \mathcal{U}^{n-1} \backslash NA^n \text{ and } I \in \mathcal{U}^n \\ +\infty & \text{otherwise} \end{cases}$$

9. Set $n := n+1$ and go to 4

We now give the convergence result for the GFMM. First, let us point out that the physical sequence of time $\{t_n, n \in \mathbb{N}\}$, defined at the step 5 in the algorithm is non-decreasing and we can extract a subsequence $\{t_{n_k}, k \in \mathbb{N}\}$ strictly increasing such that

$$t_{n_k} = t_{n_k+1} = ... = t_{n_{k+1}-1} < t_{n_{k+1}}.$$

We denote by ε the couple

$$\varepsilon = (\Delta x, \Delta t)$$

and we define θ^ε in the following way

$$\theta^\varepsilon(x,t) = \theta_I^{n_{k+1}-1} \text{ if } x \in [x_I, x_I + \Delta x[,\ t \in [t_{n_k}, t_{n_k+1}[.$$

Finally, we make the following assumption

(A') The velocity $c \in W^{1,\infty}(\mathbb{R}^N \times [0,T])$, for some constant $L > 0$ we have $|c(x',t') - c(x,t)| \leq L(|x'-x| + |t'-t|)$, and Ω_0 is a C^2 open set, with bounded boundary $\partial\Omega_0$.

Theorem 3.2 (Convergence Result)
Under assumption (A'), $\overline{\theta}^0$ (resp. $\underline{\theta}^0$) is a viscosity sub-solution (resp. super-solution) of (2.7). In particular, if (2.7) satisfies a comparison principle, then $\overline{\theta}^0 = (\underline{\theta}^0)^*$ and $(\overline{\theta}^0)_* = \underline{\theta}^0$ is the unique discontinuous viscosity solution of

$$\left\{ \begin{array}{lll} \theta_t = c(x)|\nabla\theta| & \text{in} & \mathbb{R}^N \times (0,+\infty), \\ \theta(\cdot,0) = 1_{\Omega_0} - 1_{\Omega_0^c} & \text{on} & \mathbb{R}^N. \end{array} \right. \tag{3.8}$$

3.2 Computational complexity and remarks on the implementation

In this subsection, we give some rough asymptotic bounds on the computational complexity of our GFMM algorithm. Let us assume that the velocity is constant on each time interval $[k\Delta T, (k+1)\Delta T)$ for some ΔT. Of course, the velocity is not Lipschitz in time, but can always be seen as the discretization of some Lipschitz velocity. We work on a (spatial) grid box of width $M^{\frac{1}{N}}$ in dimension N, and therefore with a total number of grid points equal to M. We assume that the velocity is normalized $|c| \simeq 1$ and then the time T for the front to pass one times on the whole grid box is roughly $T \simeq 1$. Moreover, we normalize the space step with $\Delta x = 1$. The typical size (as a number of grid points) of the front is $M^{\frac{N-1}{N}}$ in the box. We can distinguish several cases depending on the value of ΔT:

Case 1: Constant in time velocity (*i.e.* $\Delta T = +\infty$).
Here the situation is very similar to the one of the classical FMM and we can use a binary heap. Because the velocity is independent on time, we only need to recompute the value of the time at the points I whose neighbours have been accepted (*i.e.*, $I \in V(NA^n)$). This means in practice to slightly modify point 4 of our algorithm GFMM. This implies that our GFMM is equivalent to two FMM algorithms and so we recover the complexity in $O(M \log M)$.

Case 2: $O(\frac{1}{M^{\frac{1}{N}}}) \leq \Delta T < +\infty$.
In the spirit, it is equivalent to Case 1 (indeed on the time interval $[k\Delta T, (k+1)\Delta T)$). Since the time T for the front to pass one times on the whole grid box is roughly $T \simeq 1$, we deduce that the number of time interval $[k\Delta T, (k+1)\Delta T)$ is $\frac{T}{\Delta T} = \frac{1}{\Delta T}$. On each interval $[k\Delta T, (k+1)\Delta T)$, the complexity is $M_k \log M_k$ (as in Case 1) where M_k is the number of points crossed by the front during this interval of time (with $\sum_k M_k = M$). This gives a complexity

$$\sum_{k=1}^{\frac{1}{\Delta T}} M_k \log M_k.$$

Moreover, at each $k\Delta T$, we have to recompute the candidate times and the binary heap (since the velocity changes). The complexity for these operations is $O(M^{\frac{N-1}{N}} \log M)$. Therefore, the total complexity is then

$$\sum_{k=1}^{\frac{1}{\Delta T}} M_k \log M_k + \sum_{k=1}^{\frac{1}{\Delta T}} M^{\frac{N-1}{N}} \log M \leq M \log M + \frac{1}{\Delta T} M^{\frac{N-1}{N}} \log M.$$

If $\Delta T \geq \frac{1}{M^{\frac{1}{N}}}$, we then get a complexity in $O(M \log M)$ (as in the classical case).

Case 3: Variable velocity (*i.e.* $\Delta T = 0$) or $0 < \Delta T < O(\frac{1}{M^{\frac{1}{N}}})$.
From a complexity point of view, it is not interesting to use a binary heap to sort the time of the points on the front. Because the velocity changes at each step n of the algorithm, it is much more efficient to recompute all the times on the front and extract the minimum of these times at each iteration. The complexity for these operations is $O(M^{\frac{N-1}{N}})$. On the time necessary for the front to pass one times on the whole grid box, we need to do this computation $O(M)$ times. Therefore, the total complexity is $O(M^{\frac{2N-1}{N}})$.

In this case, it seems that the Narrow Band Level Set Method can be more interesting from the complexity point of view (see [1]) although it is rather difficult to make a precise statement on this point.

We can then implement the algorithm depending on the variability of the velocity in time (see Case 1, 2, 3 above). Finally, we want to point out that Case 2, where ΔT is not so small, is acceptable in practice if the velocity is Lipschitz in time with a reasonable Lipschitz constant (and this is what we assume theoretically in our convergence theorem).

4 Application to image segmentation

4.1 modelling

In this section, we present the model introduced in [7] to applied the GFMM to image segmentation problem. The question that remains to be dealt with is how to build a normal velocity suitable for the segmentation problem which should allow to simultaneously have parts moving inward and others moving outward.

Let Ω be a bounded open subset of $\mathbb{R}^2$ (the higher dimension case can be treated similarly), $\partial\Omega$ its boundary and let I be a given bounded image function defined by $I : \bar{\Omega} \to \mathbb{R}$. We recall that at a discrete time t_n, the front will be represented by the boundary of an open set Ω_n and by a phase field θ^n defined equal to 1 on Ω_n and -1 on its complementary set. The normal velocity is then based on the fitting component of the Chan-Vese model [4] and is defined as follows : at time t_n, with the previous notations,

$$c(x, t_n) = (I(x) - c_2)^2 - (I(x) - c_1)^2,$$

with

$$c_1 = \frac{\displaystyle\int_\Omega I(x) 1_{\Omega_n} dx}{\displaystyle\int_\Omega 1_{\Omega_n} dx} \quad \text{and} \quad c_2 = \frac{\displaystyle\int_\Omega I(x) 1_{\Omega_n^c} dx}{\displaystyle\int_\Omega 1_{\Omega_n^c} dx},$$

c depending on time through c_1 and c_2 that are both time-varying. (A weighted combination of $(I(x) - c_2)^2$ and $(I(x) - c_1)^2$ could be used).

The segmentation criterion in this modelling can be easily understood. Let us consider the following synthetic image (Figure 7) and the evolving contour in red. Let us also consider the green point of the curve. It is obvious that in this case $(I - c_2)^2 \simeq 0$ and $(I - c_1)^2$ is bigger. As a consequence, this part of the curve will move according to the inward normal vector to the curve. On the contrary, the yellow point is such that $(I - c_2)^2$ is bigger thus this part of the curve will move outward.

4.2 Numerical simulations

Test 1 : a rotating line
We choose as initial data a line $P(x, 0) = x_2 + 1.5x_1$ and then as representing function:

$$\theta(x, 0) = \begin{cases} 1 & \text{if } x_2 + 1.5x_1 > 0 \\ -1 & \text{otherwise.} \end{cases} \tag{4.9}$$

We choose as velocity $c(x, t) = x_1$. The exact solution of this evolution can been explicitely computed (see [3] for more details) and then we are able to compute the numerical error (in term of the Hausdorff between the two curves, see [3]). We compute the discrete solution in the numerical domain $D = [-1, 1] \times [-1, 1]$. Table 1 shows the error for the tests run with $50, 100, 200, 400$ number of nodes for each side of the square

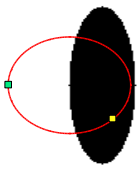

Figure 7: Illustration of the segmentation criterion in the modelling.

domain.
Table 1 show a better performance in term of accuracy for the FD scheme, this can be explained observing that in the scheme the front is represented by the zero level set of a continuous function and this convergence is uniform. On the other hand, the GFMM algorithm represents the front by the interface of a discontinuous function (i.e. the front is where there is a jump between -1 and 1). The advantage here to use the GFMM scheme is in term of CPU time, the GFMM results approximately 10 time faster than the full matrix approach.

	GFMM			**FD**		
Δx	$\lvert\mathcal{A}(\Omega_T^+) - \mathcal{A}(\Omega_m^+)\rvert$	$\mathcal{H}(\mathcal{C}, \tilde{\mathcal{C}})$	CPU	$\lvert\mathcal{A}(\Omega_T^+) - \mathcal{A}(\Omega_n^+)\rvert$	$\mathcal{H}(\mathcal{C}, \tilde{\mathcal{C}})$	CPU
0.04	$1.62 \cdot 10^{-1}$	$5.08 \cdot 10^{-2}$	$0.19s$	$1.15 \cdot 10^{-1}$	$4.10 \cdot 10^{-2}$	$1.82s$
0.02	$8.26 \cdot 10^{-2}$	$2.72 \cdot 10^{-2}$	$0.73s$	$5.16 \cdot 10^{-2}$	$2.05 \cdot 10^{-2}$	$13.2s$
0.01	$4.05 \cdot 10^{-2}$	$1.35 \cdot 10^{-2}$	$3.98s$	$2.80 \cdot 10^{-2}$	$1.03 \cdot 10^{-2}$	$102s$
0.005	$2.05 \cdot 10^{-2}$	$6.80 \cdot 10^{-3}$	$76s$	$9.00 \cdot 10^{-3}$	$2.60 \cdot 10^{-3}$	$810s$

Table 1: Area and Hausdorff distances: GFMM-*case 1* versus Finite Difference (FD), for test 1 with constant time speed

Fig.8 left shows the interface between $\{\theta^\varepsilon = 1\}$ and $\{\theta^\varepsilon = -1\}$ at each time interval 0.1. The line is rotating clockwise and it will reach in infinite time the x_2 axis. Fig.8 right shows the same test computed by the FD scheme, here the lines are the 0-level set of the discrete function $(v_I^n)_{I,n}$. In both cases the test has computed with $\Delta x = 0.01$ and the line has been plotted at times $t_n = n0.1$, $n = 1, 2, 3....$

Test 2 : Segmentation of a brain
We conclude this note by presenting a numerical result for image segmentation. As previously stressed, the computation cost of this method is comparable to the one of the classical Fast Marching Method. We present a test regarding the segmentation of a brain in order to point the ability of the GFMM to handle any kind of initialization (see Figure 9).

References

[1] D. Adalsteinsson and J. Sethian, *A fast level set method for propagating interfaces*, Journal of Computational Physics, 118 (1995), pp. 269–277.

[2] G. Barles, *Remarks on a flame propagation model.* Rapport de Recherche INRIA O464, 1985.

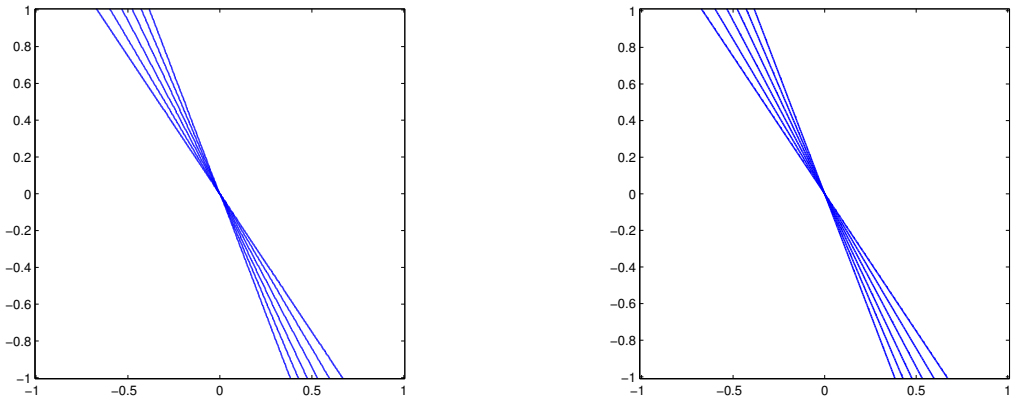

Figure 8: A rotating line by the GFMM algorithm (left) and by DF algorithm (right)

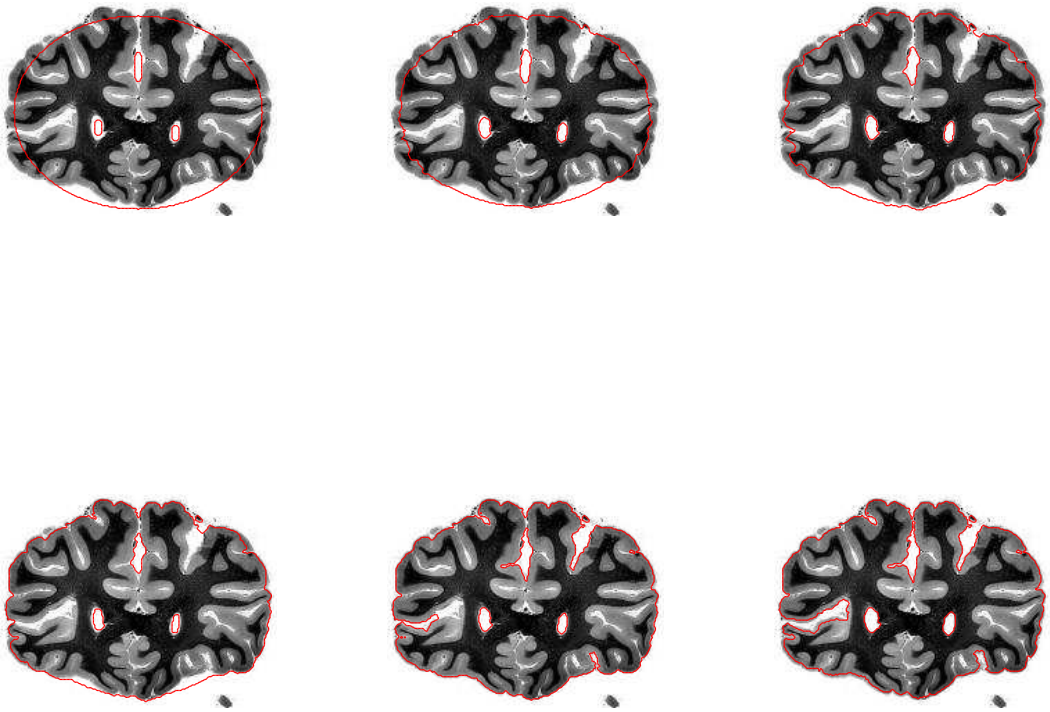

Figure 9: Different steps of the process. The initial set Ω_0 is a shape with three holes.

[3] E. Carlini, M. Falcone, N. Forcadel, and R. Monneau, *Convergence of a generalized fast marching method for a non-convex eikonal equation*, To appear in SIAM J. Num. Anal., (2006).

[4] T. Chan and L. Vese, *Active Contours Without Edges*, IEEE Transactions on Image Processing, 10 (2001), pp. 266–277.

[5] M. Falcone, *The minimum time problem and its applications to front propagation*, in Motion by mean curvature and related topics (Trento, 1992), de Gruyter, Berlin, 1994, pp. 70–88.

[6] N. Forcadel, *Comparison principle for a generalized fast marching method.* Preprint 2008.

[7] N. Forcadel, C. Le Guyader, and C. Gout, *Generalized fast marching method: applications to image segmentation*, Numer. Algorithms, 48 (2008), pp. 189–211.

[8] S. Osher, *A level set formulation for the solution of the Dirichlet problem for Hamilton-Jacobi equations*, SIAM J. Math. Anal., 24 (1993), pp. 1145–1152.

[9] S. Osher and R. Fedkiw, *Level set methods and dynamic implicit surfaces*, vol. 153 of Applied Mathematical Sciences, Springer-Verlag, New York, 2003.

[10] S. Osher and J. A. Sethian, *Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.

[11] E. Rouy and A. Tourin, *A viscosity solutions approach to shape-from-shading*, SIAM J. Numer. Anal., 29 (1992), pp. 867–884.

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

[13] ——, *Level set methods*, vol. 3 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, 1996. Evolving interfaces in geometry, fluid mechanics, computer vision, and materials science.

[14] ——, *Level set methods and fast marching methods*, vol. 3 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, second ed., 1999. Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science.