SIAM J. IMAGING SCIENCES
Vol. 13, No. 1, pp. 497–534

A High-Order Scheme for Image Segmentation via a Modified Level-Set Method[*]

Maurizio Falcone[†], Giulio Paolucci[†], and Silvia Tozza[‡]

Abstract. In this paper, we propose a high-order accurate scheme for image segmentation based on the level-set method. In this approach, the curve evolution is described as the 0-level set of a representation function, but we modify the velocity that drives the curve to the boundary of the object in order to obtain a new velocity with additional properties that are extremely useful to develop a more stable high-order approximation with a small additional cost. The approximation scheme proposed here is the first 2D version of an adaptive "filtered" scheme recently introduced and analyzed by the authors in one dimension. This approach is interesting since the implementation of the filtered scheme is rather efficient and easy. The scheme combines two building blocks (a monotone scheme and a high-order scheme) via a filter function and smoothness indicators that allow one to detect the regularity of the approximate solution adapting the scheme in an automatic way. Some numerical tests on synthetic and real images confirm the accuracy of the proposed method and the advantages given by the new velocity.

Key words. image segmentation, level-set method, Hamilton–Jacobi equations, filtered scheme, smoothness indicators

AMS subject classifications. 68U10, 35F21, 35Q68, 65M06, 65M25

DOI. 10.1137/18M1231432

1. Introduction.

The level-set (LS) method was introduced by Osher and Sethian in the 1980s [36, 31] and then used to deal with several applications, e.g., front propagation, computer vision, and computational fluid dynamics (see the monographs by Sethian [35] and by Osher and Fedkiw [30] for several interesting examples). This method is nowadays very popular for its simplicity and for its capability to deal with topological changes. In fact, the main advantages of the LS method are the possibility to easily describe time-varying objects, follow shapes that change topology, for example when a shape splits in two, develop holes, or to do the reverse of these operations. For the image segmentation problem, the application of the LS method is based on the evolution of a curve according to a normal velocity based on the gray levels of the image; typically, the curve is described as the 0-level set of a representation function (or LS function).

In $\mathbb{R}^2$, the LS method corresponds to defining an initial closed curve Γ_0 using an auxiliary function v_0 which has to change sign on Γ_0. The evolution of that curve at time t is denoted

[*]Received by the editors December 10, 2018; accepted for publication (in revised form) December 30, 2019; published electronically March 25, 2020. The three authors are members of the INdAM Research Group GNCS.
https://doi.org/10.1137/18M1231432
Funding: This work was supported by INdAM-GNCS.

[†]Dipartimento di Matematica, "Sapienza" Università di Roma, 00185 Rome, Italy (falcone@mat.uniroma1.it, paolucci@mat.uniroma1.it).

[‡]Istituto Nazionale di Alta Matematica, U.O. Dipartimento di Matematica, "Sapienza" Università di Roma, 00185 Rome, Italy (tozza@mat.uniroma1.it).

by Γ_t and is represented by the 0-level set of a function v, i.e.,

$$\Gamma_t := \{(x,y) : v(t,x,y) = 0\}. \tag{1.1}$$

This function v is the unique viscosity solution of the following evolutive nonlinear equation of Hamilton–Jacobi type:

$$\begin{cases} v_t + c(t,x,y)|\nabla v| = 0, & (t,x,y) \in (0,T) \times \mathbb{R}^2, \\ v(0,x,y) = v_0(x,y), & (x,y) \in \mathbb{R}^2, \end{cases} \tag{1.2}$$

where $\nabla v := (v_x, v_y)$ denotes the spatial gradient of v. Usually, the velocity $c(t,x,y)$ does not change sign during the evolution and the orientation depends on the type of evolution (outward for an expansion and inward for a shrinking). Typically, v_0 must be a *proper representation* of the initial front Γ_0, satisfying

$$\begin{cases} v_0(x,y) < 0, & (x,y) \in \Omega_0, \\ v_0(x,y) = 0, & (x,y) \in \Gamma_0, \\ v_0(x,y) > 0, & (x,y) \in \mathbb{R}^2 \setminus \overline{\Omega}_0, \end{cases} \tag{1.3}$$

where Ω_0 is the region delimited by Γ_0 (or the reverse inequalities).

The LS method can handle velocities also depending on physical quantities in order to describe several phenomena. Typical examples are

(a) $c(t,x,y)$, isotropic growth with time varying velocity;
(b) $c(t,x,y,\eta)$, anisotropic growth, dependent on normal direction;
(c) $c(t,x,y,k)$, mean curvature motion, with $k(t,x,y)$ mean curvature to the front at time t.

The literature on the LS method is huge, as well as the range of applications where the method has been successfully applied. We refer the interested reader to [13, 28] for the isotropic case, to [25] for the anisotropic growth, and to [14, 10] for the curvature case (see also the monographs [35, 30] and the references therein). Depending on the choice of the velocity, the front evolution will be described by first- or second-order partial differential equations; e.g., case (c) gives rise to a degenerate second-order equation. Here we limit ourselves to first-order problems (1.2) and velocities which depend only on time and position, and we consider a velocity sign constant in time since this is the more relevant case for image segmentation (a change in sign is relevant for dislocations in material science and has been studied in [9, 12]).

To put our contribution into perspective, let us mention that the segmentation problem has been solved by various techniques which mainly rely on two different approaches: variational methods and active contour methods. For the first approach, the interested reader can see [8, 7] and the references therein. For the link between the two classes of methods, see [14]. To have an idea of other segmentation methods, we refer the reader to the surveys [40, 11] and to the papers [16, 38] for the so-called *balloon model* first introduced in [15]. This method is based on the introduction of a potential giving a driving force to the segmentation process and typically leads to second-order partial differential equations that are at present outside the reach of the filtered scheme proposed in this paper. For high-order Runge–Kutta methods in the framework of image segmentation, we refer the reader to [34]. As already said, we will apply the LS method based on (1.2) looking for an accurate numerical method. High-order methods have been proposed for (1.2), and most of them are based on nonoscillatory local interpolation techniques that allow one to avoid spurious oscillations around discontinuities of the solution and/or jumps in the gradient. These techniques were originally developed for conservation laws (see the seminal paper [21] and the references therein), the research activity on essentially nonoscillatory (ENO) methods has been rather effective, and a number of improvements have been proposed in, e.g., [27, 24, 2], so that now ENO and weighted ENO (WENO) techniques are rather popular in many applications (see [39] for a recent survey). We should also mention that later on these techniques were successfully applied to the numerical solution of Hamilton–Jacobi equations [32], opening the way to other applications. The reader interested in image processing will find an application to image compression in [1, 3] and an application to image segmentation in [37]. However, a general convergence theorem for ENO/WENO schemes is still missing and their application is a rather delicate issue. These limitations have motivated further investigations, and a new class of high-order methods for evolutive Hamilton–Jacobi equations has been proposed looking for a different approach based on *filtered schemes* introduced in this framework by Lions and Souganidis [26] (it is important to note that the name does not refer to filtering a noise, as is common in the imaging community, but rather to the presence of a filter function, as we will explain later). The class of filtered schemes is based on a simple coupling of a monotone scheme with a high-order scheme. Monotone schemes are convergent to the weak (viscosity) solution, but they are known to be at most first-order accurate, whereas high-order schemes give a higher accuracy but in general are not stable. The crucial point is the coupling between the two schemes which is obtained via a *filter function* that selects which scheme has to be applied at a node of the grid in order to guarantee (under appropriate assumptions) a global convergence. The construction of these schemes is rather simple, as explained by Oberman and Salvador [29], because one can couple various numerical methods and leave the filter function deciding the switch between the two schemes. A general convergence result has been proved by Bokanowski, Falcone, and Sahu [5] and recently improved by Falcone, Paolucci, and Tozza [18] with an adaptive and automatic choice of the parameter governing the switch. Note that the adaptation of the parameter depends on some regularity indicators in every cell; these indicators are computed at each iteration, and this guarantees convergence. Some contributions to extend filtered schemes to second-order problems can be found in [20] for the stationary case and in [6] for the evolutive case and financial applications.

Our contribution. The contribution of this paper is twofold: from a theoretical point of view, we propose a modified velocity for the segmentation problem in the LS approach. This new velocity is important (and necessary) for the numerical approximation of high-order schemes, like the filtered scheme considered here, and its properties will allow one to avoid the re-initialization procedure used, e.g., in [28]. We also improve the accuracy of the method by applying an adaptive high-order filtered scheme to the segmentation problem. This requires a 2D extension of the scheme proposed in one dimension in [19], for which convergence has been proved under rather general assumptions in [18], thanks to the definition of new full 2D smoothness coefficients (see section 3). We will show that this choice is competitive with respect to other high-order schemes, such as the WENO scheme, in terms of accuracy and computational cost, analyzing several experiments. This contribution is more efficient in terms of CPU time and is interesting from a theoretical point of view since a precise convergence result for the WENO scheme is still missing.

Paper organization. The paper is organized as follows: In section 2, we give the idea behind the new definition of the velocity function, with details on its construction and explaining why it is so important to introduce it when applying high-order approximation schemes. In section 3, we briefly present the adaptive filtered (AF) scheme and recall some of its basic elements. In section 4, we give some information on the implementation and give a sketch of Algorithm 4.1 for the solution of the segmentation problem via the new AF scheme. Finally, the performances of this new method for the segmentation problem are illustrated in section 5, where we compare it with a classical monotone scheme (first-order accurate) and another high-order scheme (the WENO scheme) on a series of virtual and real images presenting a detailed error analysis of the numerical experiments.

2. Image segmentation via a modified LS method.

The boundaries of one (or more) object(s) inside a given image are characterized by an abrupt change of the intensity values $I(x,y)$ of the image, so that the magnitude of $|\nabla I(x,y)|$ can be used as an indication of the edges. Let us assume that ∇I exists at least almost everywhere; the definition of the velocity $c(x,y)$ in (1.2) plays a crucial role and must be defined in a proper way in order to guarantee that the curve evolution is close to 0 when the front is close to an edge: this will stop the evolution. The normal velocity will be positive or negative depending on the case, expanding or shrinking, respectively, and if the velocity is just given in terms of I, it will not change sign during the evolution, as it could happen when the velocity depends on other geometrical properties of the curve (e.g., the curvature). We will focus on velocities which ignore the curvature since the numerical approach we present here is well adapted to first-order evolutive Hamilton–Jacobi equations (the extension to second-order problems goes beyond the scope of this paper). Note that, in order to reduce the noise, several methods have been proposed in the literature. A very simple one is to use the convolution with a Gaussian kernel. This can be obtained by evolving the original function I of the gray levels according to the heat equation for a short time interval (see section 4 for more details). Due to the regularizing effect of the heat equation this also guarantees that ∇I exists.

Several definitions of the velocity function $c(x,y)$ have been proposed in the literature. A typical example is

$$c_1(x,y) = \frac{1}{(1 + |\nabla(G * I(x,y))|^{\mu})}, \qquad \mu \geq 1, \tag{2.1}$$

where μ is used to give more weight to the changes in the gradient, if necessary. In [13], the authors proposed that velocity with $\mu = 2$, and in [28] with $\mu = 1$. According to this definition, the velocity takes values in $[0,1]$ and has values that are close to zero at points where the image gradient is high and equal to 1 where I is constant.

Another possible choice has been proposed in [28] and has the form

$$c_2(x,y) = 1 - \frac{|\nabla(G * I(x,y))| - m}{M - m}, \tag{2.2}$$

where m and M are, respectively, the minimum and the maximum values of $|\nabla(G * I(x,y))|$. This velocity has similar properties with respect to (2.1) and takes values in $[0,1]$ but is close to 0 if the magnitude of the image gradient is close to its maximal value, and equal to 1 otherwise.

It is clear that both definitions have the desired properties but with slightly different features. More precisely, in the first case the velocity depends more heavily on the changes in the magnitude of the gradient, allowing for an easier detection of the edges but possibly producing false edges inside the object (e.g., when specular effects are present in the image). The velocity (2.2) is smoother inside the objects, being less dependent on the relative changes in the gradient, but might present some problems in the detection of all the edges if at least one of those is "more marked."

2.1. Extension of the velocity function. The edge-stopping function which is defined choosing one of the above mentioned velocities has a physical meaning only on the front Γ_t since it was designed precisely to force the 0-level set to stop close to the edges. As has been observed in [28], its meaning does not come from the geometry of v but only from the configuration of the front Γ_t. Using one of the classical velocity functions introduced before, as will be clarified by the numerical tests in section 5, high-order schemes produce unstable results since the numerical approximation of v can start to produce spurious oscillations near the edges where the front should stop. This problem can be solved by adding a limiter as in [5], but here we present a different technique that avoids the use of a limiter and adapts automatically the scheme according to the regularity of the solution in order to produce more accurate results. To this end, we need to extend the image-based velocity function $c(x, y)$ to all the level sets of the representation v in order to give a physical meaning to the speed used in the whole domain.

Following some of the ideas discussed in [28], we want to extend in a simple way the velocity to the whole domain. Our approach exploits the choice of the initial condition v_0 (which is free) and allows us to avoid all the heavy computations required by the numerical approach solution proposed by the authors in [28]. The modification has an interesting interpretation in terms of the method of characteristics, as we will see later in this section, and allows for stable numerical results. Thus, recalling their approach, the first property that the velocity has to satisfy is the following.

Property 2.1. *Any external (image-based) speed function that is used in the equation of motion written for the function v should not cause the level sets to collide and cross each other during the evolutionary process.*

To present the main idea, let us consider the signed distance to the initial 0-level set as the initial representation function. This is the classical choice

$$v_0(x,y) = \begin{cases} -dist\,\{(x,y),\Gamma_0\} & \text{in } \Omega_0, \\ dist\,\{(x,y),\Gamma_0\} & \text{outside } \Omega_0, \end{cases} \tag{2.3}$$

where Ω_0 is the internal region delimited by Γ_0. Therefore, with this choice we can define the velocity extension as follows.

Property 2.2. *The value of the speed function $c(x, y)$ at a point P lying on a level set $\{v = C\}$ is exactly the value of $c(x, y)$ at a point Q, such that the point Q is a distance C away from P and lies on the level set $\{v = 0\}$.*

Note that the point Q is uniquely determined whenever the normal direction in P is well defined. In fact, $Q = P - c(x,y)\eta(P)$, where η is the outgoing normal, and this will provide a definition also when the level sets are nonconvex. To develop a formal argument, we will assume that the normal is sufficiently regular.

In order to compute the point Q on the 0-level set associated to each point P of any level set, the authors introduce in [28] on pages 162–164 a neighborhood of the 0-level set called *narrow-band*. They fix the width δ of the band and the related number of iterations l needed for the 0-level set to reach the boundary of the narrow-band (this is done by the procedure explained in [28] on page 164). This procedure forces one to modify the representation function since the solution is updated only inside the band, and a re-initialization step becomes strictly necessary in order to restore the meaning of the distance function. Referencing their paper, such a procedure either requires at least $O(N^3)$ computations for a grid of N points in each direction or requires one to compute the solution of an associated stationary eikonal equation. In order to reduce the computational complexity, the method proposed here is based on a direct assignment of the associated point on the 0-level set, which requires only $O(N^2)$ operations to compute the modified velocity at each iteration, and greatly simplifies the problem. Moreover, it allows for the use of representation functions (i.e., initial conditions) even more regular than the signed distance function. The idea is straightforward and is based on the fact that the evolution is oriented in the normal direction to the front. If the reciprocal position of the level sets is also known (that is why we must choose wisely the initial condition) and we make all the points in the normal direction to the 0-level set evolve according to the same law, it is reasonable to expect that all such points will keep their relative distance unchanged as time flows.

To illustrate and motivate our modification, let us still consider the distance to Γ_0 (2.3) as the initial condition and let us consider the shrinking case as example. Then, by construction, all the C-level sets are at a distance C from the 0-level set, as stated by Property 2.2. Hence, if we consider a generic point (x_c, y_c) on a C-level set, then it is reasonable to assume that the closest point on Γ_0 should be

$$(x_0, y_0) = (x_c, y_c) - v(t, x_c, y_c)\frac{\nabla v(t, x_c, y_c)}{|\nabla v(t, x_c, y_c)|}. \tag{2.4}$$

Therefore, it seems natural to define the extended velocity $\widetilde{c}(x,y)$ as

$$\widetilde{c}(x, y, v, v_x, v_y) = c\left(x - v\frac{v_x}{|\nabla v|}, y - v\frac{v_y}{|\nabla v|}\right), \tag{2.5}$$

which coincides with $c(x,y)$ on the 0-level set, as it is needed. The same approach can be applied as long as the initial distance between the level sets is known. In that case, if we want higher regularity to the evolving surface, which would be preferable in the case of high-order schemes such as those we use in the numerical tests, we can define an appropriate initial condition, for example, by simply rotating a regular function in one space dimension. More precisely, let us consider a regular function $\overline{v}_0 : \mathbb{R}^+ \to \mathbb{R}$ such that $\overline{v}_0(r_0) = 0$, where r_0 is the radius of the initial circle Γ_0 (e.g., the right branch of a parabola centered in the origin), and let us define $v_0(x,y)$ rotating its profile, that is,

$$v_0(x, y) = \overline{v}_0\left(\sqrt{x^2 + y^2}\right). \tag{2.6}$$

Then, it is clear that the C-level sets of v_0 are located at a distance

$$d(C) := \overline{v}_0^{-1}(C) - r_0, \qquad \text{with } \overline{v}_0^{-1}(C) \geq 0, \tag{2.7}$$

from the 0-level set and, according to our previous remarks, they should keep this property as time evolves. Consequently, also in this case we can define

$$\widetilde{c}(x, y, v, v_x, v_y) = c\left(x - d(v)\frac{v_x}{|\nabla v|}, y - d(v)\frac{v_y}{|\nabla v|}\right). \tag{2.8}$$

More details on the function $d(v)$ will be given in section 2.2. For simplicity, in the last construction we assumed the representation function to be centered in the origin, but it is straightforward to extend the same procedure to more general situations. Note also that if we have only one object to be segmented (or we are considering the shrinking from the frame of the picture), we can always use a representation function centered in the origin since we can choose freely the domain of integration, given by the pixels of the image.

2.2. Motivations of the new velocity function. The modification of the velocity $c(x, y)$ into $\widetilde{c}(x, y, v, v_x, v_y)$ defined in (2.8) with $d(v) = 0$ if $v = 0$ is to follow the evolution of the 0-level set and then to define the evolution on the other level sets accordingly. This allows one to give a geometrical interpretation of the new velocity and to establish some properties that will also guarantee existence and uniqueness for the first-order evolutive problem. The new velocity can be seen as a *characteristic based velocity*. As a first step, let us analyze the characteristics of the equation; in these computations, we assume that we always have the necessary regularity. In particular, we assume $v \in C^2(\Omega)$ (or at least C^2 in space and C^1 in time) and $c(x, y) \in C^1(\Omega)$. We introduce the notations of the vectors $z := (x, y)$ and $p := (p_1, p_2) = (v_x, v_y)$ that will be used only in this section. Using these notations, the Hamiltonian in our case can be written as

$$H(z, v, p) = \widetilde{c}(z, v, p)|p|. \tag{2.9}$$

Let us introduce the method of characteristics, writing the usual system

$$\begin{cases} \dot{z}(s) = \nabla_p H, \\ \dot{v}(s) = \nabla_p H \cdot p - H, \\ \dot{p}(s) = -\nabla H - H_v p, \end{cases} \tag{2.10}$$

where $\dot{f}$, $f = z, v, p$, denotes the derivative with respect to the variable s, $\nabla_p H$ the gradient with respect to p, H_v the partial derivative with respect to the (scalar) value v, and ∇H the usual spatial gradient of H already introduced in section 1. In our case, defining for brevity the point $(\xi,\zeta) := (x - d(v)\frac{p_1}{|p|}, y - d(v)\frac{p_2}{|p|})$, we obtain

$$\begin{aligned}\frac{\partial H}{\partial p_1} =& \frac{\partial \widetilde{c}}{\partial p_1}|p| + \widetilde{c}\frac{\partial |p|}{\partial p_1}\\ =& \left(\frac{\partial c}{\partial \xi}\cdot\frac{\partial \xi}{\partial p_1} + \frac{\partial c}{\partial \zeta}\cdot\frac{\partial \zeta}{\partial p_1}\right)|p| + \widetilde{c}\frac{p_1}{|p|}\\ =& - d(v)\frac{\partial c}{\partial \xi}\left(\frac{|p| - \frac{p_1^2}{|p|}}{|p|^2}\right)|p| - d(v)\frac{\partial c}{\partial \zeta}\left(-\frac{p_1p_2}{|p|^3}\right)|p| + \widetilde{c}\frac{p_1}{|p|}\\ =& - d(v)\frac{\partial c}{\partial \xi}\frac{p_2^2}{|p|^2} + d(v)\frac{\partial c}{\partial \zeta}\frac{p_1p_2}{|p|^2} + \widetilde{c}\frac{p_1}{|p|},\end{aligned} \tag{2.11}$$

and analogously $\frac{\partial H}{\partial p_2}$, so that we obtain

$$\nabla_p H = \begin{pmatrix} \frac{d(v)p_2}{|p|^2}\left(p_1\frac{\partial c}{\partial \zeta} - p_2\frac{\partial c}{\partial \xi}\right) + \widetilde{c}\frac{p_1}{|p|} \\ \frac{d(v)p_1}{|p|^2}\left(p_2\frac{\partial c}{\partial \xi} - p_1\frac{\partial c}{\partial \zeta}\right) + \widetilde{c}\frac{p_2}{|p|}\end{pmatrix}, \quad \text{implying } \nabla_p H\cdot p = \widetilde{c}(z,v,p)|p|. \tag{2.12}$$

Therefore, the system (2.10) becomes in our case the following:

$$\begin{cases} \dot{z}(s) = \nabla_p H,\\ \dot{v}(s) = \widetilde{c}(z,v,p)|p| - \widetilde{c}(z,v,p)|p| = 0,\\ \dot{p}(s) = -\nabla\widetilde{c}(z,v,p)|p| + d'(v)\nabla\widetilde{c}(z,v,p)|p|^2 = \nabla\widetilde{c}(z,v,p)|p|(d'(v)|p|-1),\end{cases} \tag{2.13}$$

where $d'(v)$ denotes the derivative of the distance d with respect to the value v.

Now, choosing $d'(v)$ such that

$$d'(v) = |p|^{-1}, \tag{2.14}$$

we have the final system

$$\begin{cases} \dot{z}(s) = \nabla_p H,\\ \dot{v}(s) = 0,\\ \dot{p}(s) = 0,\end{cases} \tag{2.15}$$

which states that, as long as the function $\widetilde{c}$ remains smooth enough ($\frac{\partial c}{\partial \xi} \approx 0$ and $\frac{\partial c}{\partial \zeta} \approx 0$), the characteristics are basically directed in the normal direction and along them both the height and the gradient are preserved. Looking at the third relation of (2.15) and at the choice (2.14), since $p(s) \equiv p(0) = \nabla v_0$ along the characteristics, we can choose simply

$$d'(v) = |\nabla v_0|^{-1}, \tag{2.16}$$

which is the trivial case with the function $d(v) = v$ and also for $d(v)$ given by the previous definition (2.7). In fact, using the inverse function theorem, we have

$$d'(v) = \frac{d}{dv}\left(\overline{v}_0^{-1}(v)\right) = \frac{1}{\overline{v}_0'(w)}, \tag{2.17}$$

with w such that $\overline{v}_0(w) = v$. Moreover, recalling the definition (2.6), we can compute

$$\begin{aligned}
(2.18)\qquad |\nabla v_0(x,y)| &= \left|\nabla \overline{v}_0\left(\sqrt{x^2+y^2}\right)\right| \\
&= \left|\left(\frac{\overline{v}_0'\left(\sqrt{x^2+y^2}\right)x}{\sqrt{x^2+y^2}}, \frac{\overline{v}_0'\left(\sqrt{x^2+y^2}\right)y}{\sqrt{x^2+y^2}}\right)\right| \\
&= \frac{\overline{v}_0'\left(\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}}\sqrt{x^2+y^2} = \overline{v}_0'\left(\sqrt{x^2+y^2}\right),
\end{aligned}$$

and then it is enough to consider (x, y) such that $w = \sqrt{x^2+y^2}$. From a numerical point of view, we compute the function $d(v)$ analytically, by using (2.16), which exploits our knowledge of the initial condition v_0, given, e.g., by (2.6). Thanks to the previous computations, we reached a good understanding of the nature of the evolution given by (1.2)–(2.8), but we still have not justified the main motivation that led us to define (2.8), that is, to make all the level sets of v evolve according to the same law. More precisely, we have to show that if we consider the evolution of two points on the same characteristic but on two different level sets, say the 0-level set $z^0(s)$ and a generic level set $z^\ell(s)$, then their relative distance (along the characteristic) does not change during the evolution. This fact would imply that if we choose the level sets of v_0 to be such that

$$(2.19)\qquad z^0(0) = z^\ell(0) - d(v_0(z^\ell))\frac{\nabla v_0(z^\ell)}{|\nabla v_0(z^\ell)|},$$

then the points $\underline{z}(s) := z^\ell(s) - d(v(z^\ell))\frac{p(z^\ell)}{|p(z^l)|}$ are always on the 0-level set of v. In order to prove this last statement, let us proceed by a simple differentiation, dropping the dependence on z^ℓ for brevity:

$$\begin{aligned}
(2.20)\qquad \dot{\underline{z}}(s) &= \dot{z}^\ell(s) - \frac{d}{ds}\left(d(v)\frac{p}{|p|}\right) \\
&= \dot{z}^\ell(s) - \left[d'(v)\dot{v}(s)\frac{p}{|p|} + \frac{d(v)}{|p|^2}\left(\dot{p}(s)|p| - \frac{d}{ds}(|p(s)|)p\right)\right].
\end{aligned}$$

Recalling the relations in the system (2.15) with respect to $\dot{v}(s)$ and $\dot{p}(s)$, we can write

$$\begin{aligned}
(2.21)\qquad \dot{\underline{z}}(s) &= \dot{z}^\ell(s) + \frac{d(v)}{|p|^2}\left(\frac{p\cdot\dot{p}(s)}{|p|}\right)p \\
&= \dot{z}^\ell(s).
\end{aligned}$$

This calculation shows that the points $\underline{z}(s)$ and $z^\ell(s)$ evolve according to the same law along characteristics. Note that if (2.19) holds, then $\underline{z}(s) \equiv z^0(s)$ until the characteristics do not cross. In fact, computing the total derivative with respect to s, we have

$$(2.22)\qquad \frac{d}{ds}v(s,\underline{z}(s)) = v_s + v_x\dot{\underline{z}}(s) = v_s + v_x\dot{z}^\ell(s) = \frac{d}{ds}v(s,z^\ell(s)) = 0$$

since the points of $z^\ell(s)$ are on the same level set, and so are those of $\underline{z}(s)$, as we wanted. This directly implies that the points $\underline{z} := \left(z - d(v)\frac{\nabla v}{|\nabla v|}\right)$ are on the 0-level set of v as long as the gradient is preserved.

In order to reduce the computational cost and to simplify the implementation, as will be better explained in section 4, we decided to split the computation of the Hamiltonian into two steps. First we compute the points $\underline{z}$, and then we update the numerical solution of the full problem (1.2) with the velocity (2.8) via the following *simplified problem* with isotropic velocity:

$$v_t + c(\underline{z})|\nabla v| = 0, \qquad (t,x,y) \in (t_n, t_{n+1}) \times \mathbb{R}^2, \tag{2.23}$$

where, as usual, $t_n = t_0 + n\Delta t$, and Δt is the time step. Using this procedure, the dependencies on the gradient and on the value of v are frozen at every iteration, leaving just an explicit dependence on the variable t. This is why we consider velocities depending only on space variables when defining the numerical schemes.

Let the original velocity $c(x,y)$ be Lipschitz continuous (this is the classical assumption), and let us denote by L_c its Lipschitz constant. We conclude showing that our modified velocity $\tilde{c}$ is still Lipschitz continuous. This point is important to guarantee existence and uniqueness of the characteristics and of the viscosity solution for the evolutive problem (1.2) driven by the new velocity. Let $z = (x,y)$ and $z' = (x', y')$ be two points in the plane and $\underline{z}$ and $\underline{z}'$ the corresponding points used in the definition of the new velocity $\tilde{c}$. Provided that the solution and the normal vector are Lipschitz continuous, we have

$$\begin{aligned} |c(\underline{z}) - c(\underline{z}')| &= \left| c\left(z - d(v(z))\frac{\nabla v(z)}{|\nabla v(z)|}\right) - c\left(z' - d(v(z'))\frac{\nabla v(z')}{|\nabla v(z')|}\right)\right| \\ &\le L_c\left(|z - z'| + |d(v(z'))\eta(z') - d(v(z))\eta(z)|\right) \\ &\le L_c\left(|z - z'| + |d(v(z')) - d(v(z))||\eta(z')| + |d(v(z))||\eta(z') - \eta(z)|\right) \\ &\le L_c(1 + C_1 + C_2)|z' - z|, \end{aligned} \tag{2.24}$$

where η represents the normal unitary vector at the point and C_1, C_2 are two appropriate constants (remember that the distance from the level set stays bounded during the evolution).

3. The adaptive filtered scheme.

In this section, we will introduce and illustrate the AF scheme we will use to approximate the viscosity solution of the problem (1.2). It is important to note that the name does not refer to filtering a noise, as is common in the imaging community, but rather to the presence of a filter function, as we will explain later in this section. For more details on the AF scheme, see [19]. We assume that the Hamiltonian H and the initial data v_0 are Lipschitz continuous functions in order to ensure the existence and uniqueness of the viscosity solution [17]. For a detailed presentation of uniqueness and existence results for viscosity solutions, we refer the reader to [17, 4].

Now let us define a uniform grid in space $(x_j, y_i) = (j\Delta x, i\Delta y)$, $j,i \in \mathbb{Z}$, and in time $t_n = t_0 + n\Delta t$, $n \in [0, N_T]$, with $(N_T - 1)\Delta t < T \le N_T \Delta t$. Then we compute the numerical approximation $u_{i,j}^{n+1} = u(t_{n+1}, x_j, y_i)$ with the simple formula

$$u_{i,j}^{n+1} = S^{AF}(u^n)_{i,j} := S^M(u^n)_{i,j} + \phi_{i,j}^n \varepsilon^n \Delta t F\left(\frac{S^A(u^n)_{i,j} - S^M(u^n)_{i,j}}{\varepsilon^n \Delta t}\right), \tag{3.1}$$

where S^M and S^A are, respectively, the monotone and the high-order schemes dependent on both space variables, $F : \mathbb{R} \to \mathbb{R}$ is the *filter function* needed to switch between the two schemes, ε^n is the switching parameter at time t_n, and $\phi^n_{i,j}$ is the *smoothness indicator function* at the node (x_j, y_i) and time t_n, based on the $2D$*-smoothness coefficients* defined in [33] and briefly recalled later on in this section. The AF scheme introduced here is convergent, as proven in [18]. In what follows, the gradient components will be denoted by the usual notation (p, q) and p^+, p^- will be the right and left discrete derivatives with respect to x (similar notations apply to q, which denotes the discrete partial derivative with respect to y).

The two schemes composing the AF scheme can be freely chosen, provided that they satisfy the following assumptions.

Assumptions on S^M. The scheme is consistent and monotone and can be written in *differenced form*

$$u^{n+1}_{i,j} = S^M(u^n)_{i,j} := u^n_{i,j} - \Delta t\ h^M\left(x_j, y_i, D^-_x u^n_{i,j}, D^+_x u^n_{i,j}, D^-_y u^n_{i,j}, D^+_y u^n_{i,j}\right) \tag{3.2}$$

for a Lipschitz continuous function $h^M(x, y, p^-, p^+, q^-, q^+)$, with $D^\pm_x u^n_{i,j} := \pm\frac{u^n_{i,j\pm1}-u^n_{i,j}}{\Delta x}$ and $D^\pm_y u^n_{i,j} := \pm\frac{u^n_{i\pm1,j}-u^n_{i,j}}{\Delta y}$.

Assumptions on S^A. The scheme has a high-order consistency and can be written in *differenced form*

$$\begin{aligned} u^{n+1}_{i,j} = S^A(u^n)_{i,j} := u^n_{i,j} - \Delta t h^A \Big(& x_j, y_i, D^-_{k,x} u_{i,j}, \dots, D^-_x u^n_{i,j}, D^+_x u^n_{i,j}, \dots, D^+_{k,x} u^n_{i,j}, \\ & D^-_{k,y} u_{i,j}, \dots, D^-_y u^n_{i,j}, D^+_y u^n_{i,j}, \dots, D^+_{k,y} u^n_{i,j} \Big) \end{aligned} \tag{3.3}$$

for a Lipschitz continuous function $h^A(x, y, p^-, p^+, q^-, q^+)$ (in short), with

$$D^\pm_{k,x} u^n_{i,j} := \pm\frac{u^n_{i,j\pm k} - u^n_{i,j}}{k\Delta x} \qquad \text{and} \qquad D^\pm_{k,y} u^n_{i,j} := \pm\frac{u^n_{i\pm k,j} - u^n_{i,j}}{k\Delta y}.$$

Example 3.1. *As examples of monotone schemes in differenced form satisfying the hypotheses stated before, we can consider the simple numerical Hamiltonian*

$$h^M(p^-, p^+, q^-, q^+) := \sqrt{\max\{p^-, -p^+, 0\}^2 + \max\{q^-, -q^+, 0\}^2} \tag{3.4}$$

for the eikonal equation

$$v_t + \sqrt{v_x^2 + v_y^2} = 0, \tag{3.5}$$

or, for more general equations also depending on the space variables, we can use the $2D$*-version of the* local Lax–Friedrichs Hamiltonian

$$\begin{aligned} h^M(x, y, p^-, p^+, q^-, q^+) :=& H\left(x, y, \frac{p^+ + p^-}{2}, \frac{q^+ + q^-}{2}\right) \\ & - \frac{\alpha_x(p^-, p^+)}{2}(p^+ - p^-) - \frac{\alpha_y(q^-, q^+)}{2}(q^+ - q^-), \end{aligned} \tag{3.6}$$

with

$$\alpha_x(p^-,p^+) := \max_{\substack{x,y,q,\\ p\in I(p^-,p^+)}} |H_p(x,y,p,q)|, \qquad \alpha_y(q^-,q^+) := \max_{\substack{x,y,p,\\ q\in I(q^-,q^+)}} |H_q(x,y,p,q)|, \tag{3.7}$$

where $I(a,b) := [\min(a,b), \max(a,b)]$. *This scheme is monotone under the restrictions* $\frac{\Delta t}{\Delta x}\cdot\alpha_x + \frac{\Delta t}{\Delta y}\cdot\alpha_y \le 1$.

Example 3.2. *An example of numerical Hamiltonian* h^A *satisfying the assumptions required is the* Lax–Wendroff Hamiltonian

$$\begin{aligned} h^A(x,y,D_x^{\pm}u,D_y^{\pm}u) &:= H(x,y,D_xu,D_yu) \\ &\quad - \frac{\Delta t}{2}\left[H_p(x,y,D_xu,D_yu)\left(H_p(x,y,D_xu,D_yu)D_x^2u + H_x(x,y,D_xu,D_yu)\right)\right. \\ &\quad + H_q(x,y,D_xu,D_yu)\left(H_q(x,y,D_xu,D_yu)D_y^2u + H_y(x,y,D_xu,D_yu)\right) \\ &\quad \left. + 2H_p(x,y,D_xu,D_yu)H_q(x,y,D_xu,D_yu)D_{xy}^2u\right], \end{aligned} \tag{3.8}$$

where $D_x^{\pm}u$, D_xu, D_x^2u *are, respectively, the usual one-sided and centered one-dimensional finite difference approximations of the first and second derivatives in the* x*-direction (analogously for the* y*-direction), whereas for the mixed derivative we use*

$$D_{xy}^2u_{i,j} := \frac{u_{i+1,j+1} - u_{i-1,j+1} - u_{i+1,j-1} + u_{i-1,j-1}}{4\Delta x\Delta y}. \tag{3.9}$$

Note that the derivatives of H *can be computed either analytically or by some second-order numerical approximation. In particular, to compute the derivative* H_x*, we can simply use*

$$(H_x)_{i,j} := \frac{H(x_{j+1},y_i,D_xu_{i,j},D_yu_{i,j}) - H(x_{j-1},y_i,D_xu_{i,j},D_yu_{i,j})}{2\Delta x}, \tag{3.10}$$

and analogously for H_y.

For more details on the construction of S^M and S^A and other examples of possible numerical Hamiltonians, see [33, 18].

In our approach, in order to couple the two schemes, we need to define three key quantities:

1. The *filter function* F, which must satisfy the following:
   (a) $F(r) \approx r$ for $|r| \le 1$ so that if $|S^A - S^M| \le \Delta t\varepsilon^n$ and $\phi_{i,j}^n = 1$, then $S^{AF} \approx S^A$;
   (b) $F(r) = 0$ for $|r| > 1$ so that if $|S^A - S^M| > \Delta t\varepsilon^n$ or $\phi_{i,j}^n = 0$, then $S^{AF} = S^M$.

   Several choices for F are possible, different for regularity properties. In this paper, we will consider the discontinuous filter already used in [5] and defined as follows:

$$F(r) := \begin{cases} r & \text{if } |r| \le 1, \\ 0 & \text{otherwise,} \end{cases} \tag{3.11}$$

   which is clearly discontinuous at $r = -1, 1$ and satisfies trivially the two required properties.

2. If we want the scheme (3.1) to switch to the high-order scheme when some regularity is detected, we have to choose ε^n such that

$$\left|\frac{S^A(v^n)_{i,j} - S^M(v^n)_{i,j}}{\varepsilon^n \Delta t}\right| = \left|\frac{h^A(\cdot,\cdot) - h^M(\cdot,\cdot)}{\varepsilon^n}\right| \le 1 \qquad \text{for } (\Delta t, \Delta x, \Delta y) \to 0 \tag{3.12}$$

in the *region of regularity at time* t_n, that is,

$$\mathcal{R}^n := \left\{(x_j, y_i) : \phi^n_{i,j} = 1\right\}. \tag{3.13}$$

Proceeding by Taylor expansion for the monotone and the high-order Hamiltonians, by (3.12) we arrive at a lower bound for ε^n. The simplest numerical approximation of that lower bound is the following:

$$\begin{aligned}\varepsilon^n = \max_{(x_j,y_i)\in\mathcal{R}^n} K\Bigg|&\frac{\Delta t}{2}\left[H_p\left(H_x + H_p D_x^2 u^n\right) + H_q\left(H_y + H_q D_y^2 u^n\right) + 2H_pH_qD_{xy}^2u^n\right)\Big] \\ &+ \left(\widetilde{h}^M_{p^+} - \widetilde{h}^M_{p^-}\right) + \left(\widetilde{h}^M_{q^+} - \widetilde{h}^M_{q^-}\right)\Bigg|,\end{aligned} \tag{3.14}$$

in which we have used the usual notation for the gradient, i.e., $(p,q) := (v_x, v_y)$ and

$$\widetilde{h}^M_{p^+} := h^M\left(x, y, D_xu^n, D_x^+u^n, D_yu^n, D_yu^n\right) - h^M\left(x, y, D_xu^n, D_x^-u^n, D_yu^n, D_yu^n\right). \tag{3.15}$$

The definition of $\widetilde{h}^M_{p^-}, \widetilde{h}^M_{q^+}, \widetilde{h}^M_{q^-}$ follows from (3.15) in an analogous way. All the derivatives of H are computed at (x, y, D_xu^n, D_yu^n) and the finite difference approximations around the point (i,j), using $K > \frac{1}{2}$. See [33] for more details.

3. For the definition of a function ϕ, needed to detect the region $\mathcal{R}^n$, we require

$$\phi^n_{i,j} := \begin{cases} 1 & \text{if the solution } u^n \text{ is regular in } I_{i,j}, \\ 0 & \text{if } I_{i,j} \text{ contains a point of singularity,}\end{cases} \tag{3.16}$$

with $I_{i,j} := [x_{j-1}, x_{j+1}] \times [y_{i-1}, y_{i+1}]$. In order to proceed with the construction, we split the cell $I_{i,j}$ into four subcells, denoted by the superscript "$\vartheta_1\vartheta_2$," for $\vartheta_1, \vartheta_2 = +, -$, according to the shift with respect to the center (x_j, y_i). Then, extending the classical WENO approach proposed in [23] to multiple spatial dimensions, we measure the regularity of the solution inside each subcell by computing the *smoothness coefficients* as rescaled L^2 norms of the Lagrange polynomial $P_k^{\vartheta_1,\vartheta_2}(x,y)$ interpolating the values of u^n on the considered stencil, that is,

$$\begin{aligned}{\beta_k}^{\vartheta_1\vartheta_2} &= (-1)^{|\vartheta|} \sum_{\substack{\alpha_1,\alpha_2=0\\|\alpha|\ge 2}}^{2} \int_{\vartheta_1\Delta x}^{0}\int_{\vartheta_2\Delta y}^{0} \Delta x^{2(\alpha_1-1)}\Delta y^{2(\alpha_2-1)}\left(\partial_x^{\alpha_1}\partial_y^{\alpha_2}P_k^{\vartheta_1\vartheta_2}(x,y)\right)^2 dxdy \\ &= \frac{1}{\Delta x\Delta y}\Bigg[{u_{[2,0]}}^2 + {u_{[0,2]}}^2 + {u_{[1,1]}}^2 + \frac{17}{12}\left({u_{[2,1]}}^2 + {u_{[1,2]}}^2\right) + \frac{317}{720}{u_{[2,2]}}^2 + u_{[2,0]}u_{[2,1]} \\ &\quad + u_{[0,2]}u_{[1,2]} - \frac{1}{6}\left(u_{[2,0]}u_{[2,2]} + u_{[0,2]}u_{[2,2]}\right) - \frac{1}{12}\left(u_{[2,1]}u_{[2,2]} + u_{[1,2]}u_{[2,2]}\right)\Bigg],\end{aligned} \tag{3.17}$$

where $|\vartheta|$ denotes the number of "$-$" in $(\vartheta_1, \vartheta_2)$ for $k = 0, 1$. Note that we have dropped the dependence on the time step t^n for brevity and we have used the shorter notation $u_{[t,s]}$ to denote the multivariate undivided difference of u of order t in x and s in y. The previous formula can be used to obtain all the needed quantities as long as the following *ordered* stencils are used to compute the undivided differences:

- $\mathcal{S}_0^{--} = \{x_{j-1}, x_j, x_{j+1}\} \times \{y_{i-1}, y_i, y_{i+1}\}$, $\mathcal{S}_1^{--} = \{x_j, x_{j-1}, x_{j-2}\} \times \{y_i, y_{i-1}, y_{i-2}\}$;
- $\mathcal{S}_0^{+-} = \{x_{j+1}, x_j, x_{j-1}\} \times \{y_{i-1}, y_i, y_{i+1}\}$, $\mathcal{S}_1^{+-} = \{x_j, x_{j+1}, x_{j+2}\} \times \{y_i, y_{i-1}, y_{i-2}\}$;
- $\mathcal{S}_0^{++} = \{x_{j+1}, x_j, x_{j-1}\} \times \{y_{i+1}, y_i, y_{i-1}\}$, $\mathcal{S}_1^{++} = \{x_j, x_{j+1}, x_{j+2}\} \times \{y_i, y_{i+1}, y_{i+2}\}$;
- $\mathcal{S}_0^{-+} = \{x_{j-1}, x_j, x_{j+1}\} \times \{y_{i+1}, y_i, y_{i-1}\}$, $\mathcal{S}_1^{-+} = \{x_j, x_{j-1}, x_{j-2}\} \times \{y_i, y_{i+1}, y_{i+2}\}$.

Since these coefficients are such that
- $\beta_k = O(\Delta^2)$, with $\Delta := \max\{\Delta x, \Delta y\}$, if the solution is smooth in $\mathcal{S}_k$ and
- $\beta_k = O(1)$ if there is a singularity in $\mathcal{S}_k$,

according to the usual WENO procedure we weight the obtained information and focus on the "inner" stencil, denoted by the subscript "0," by computing

$$\omega^{\vartheta_1\vartheta_2} = \frac{\alpha_0^{\vartheta_1\vartheta_2}}{\alpha_0^{\vartheta_1\vartheta_2} + \alpha_1^{\vartheta_1\vartheta_2}}, \tag{3.18}$$

where $\alpha_k^{\vartheta_1\vartheta_2} = \frac{1}{(\beta_k^{\vartheta_1\vartheta_2} + \sigma_\Delta)^2}$, with $\sigma_\Delta = \Delta x^2 + \Delta y^2$, which represents the measure of smoothness of the solution in each subcell. Once we have computed the four indicators, we couple the information by defining

$$\omega = \min\{\omega^{--}, \omega^{+-}, \omega^{-+}, \omega^{++}\}, \tag{3.19}$$

which, as can be shown by exploiting the properties of the coefficients β_k, is such that

$$\omega_{i,j} = \begin{cases} O(\Delta^4) & \text{if } \rho \in I_{i,j}, \\ \frac{1}{2} + O(\Delta) & \text{otherwise,} \end{cases} \tag{3.20}$$

where ρ is a point of discontinuity in the gradient. At this point, in order to reduce the amplitude of the oscillations around the optimal value $\frac{1}{2}$ in regular regions (the $O(\Delta)$ term), we use the *mapping* first introduced in [22] to propose a modification of the original WENO procedure, called *M-WENO*, that is,

$$\omega^* = g(\omega) = 4\omega\left(\frac{3}{4} - \frac{3}{2}\omega + \omega^2\right), \tag{3.21}$$

which, using Taylor expansion around $\frac{1}{2}$, directly gives $\omega^* = \frac{1}{2} + O(\Delta^3)$ when the solution is regular in $I_{i,j}$. Finally, in order to define our function ϕ, it is enough to take

$$\phi(\omega^*) = \chi_{\{\omega^* \geq M\}}, \tag{3.22}$$

with $M < \frac{1}{2}$ (e.g., $M = 0.1$ in the tests reported in section 5), a number that can also depend on Δx and Δy . For more details and other possible constructions for the definition of ϕ, we refer the interested reader to [33, 18].

Remark 3.3. Choosing $\varepsilon^n \equiv \varepsilon \Delta x$, with $\varepsilon > 0$ and $\phi_{i,j}^n \equiv 1$, we get the filtered scheme of [5], so here we are generalizing that approach to exploit more carefully the local regularity of the solution at every time t^n and cell $I_{i,j}$.

4. Numerical implementation of the modified LS method.

Before illustrating the numerical tests, let us first give some comments on the numerical schemes composing the AF scheme adopted for the tests in section 5. The main issue concerning the *local Lax–Friedrichs* and the *Lax–Wendroff schemes* defined by (3.6) and (3.8), respectively, is the need to compute the one-directional velocities H_p and H_q which depend also on $\frac{\partial c}{\partial \xi}$ and $\frac{\partial c}{\partial \zeta}$, as visible in (2.12). Moreover, in order to implement the local Lax–Friedrichs scheme we should be able to compute the maximum of $|H_p|$ (resp., $|H_q|$) uniformly with respect to p (resp., q), which is a very intricate matter due to the (possible) low regularity of $\widetilde{c}$. In fact, if we focus on the usual behavior of $c(x, y)$ in the proximity of a relevant edge, we can expect the derivatives $\frac{\partial c}{\partial \xi}$ and $\frac{\partial c}{\partial \zeta}$ to be really big. This is not surprising since the front decelerates rapidly in the neighborhood of an edge. In addition, in order to solve the full model (1.2)–(2.8), we should take into account also the remaining dependence of $H(x, y, v, \cdot, \cdot)$ when deriving the second-order Lax–Wendroff scheme and, clearly, the formula to compute the threshold ε^n. Finally, concerning the Courant–Friedrichs–Lewy (CFL) condition, it is necessary to compute $\max\{|H_p|, |H_q|\}$ with the full formula (2.12). Consequently, λ could be excessively small due to the low regularity of $\widetilde{c}$. In this latter case, we would clearly need an adaptive mesh refinement technique to reduce the computational cost.

In order to avoid most of these complications in the numerical implementation, we choose to approximate the solution of the simplified problem (2.23), adjusting the velocity $\widetilde{c}$ according to (2.8) at each time step. Using the simplified problem (2.23), we can use the simple relation

$$\max_p \max_q |H_p(\cdot, p, q)| = \max_p |H_p(\cdot, p, 0)|, \tag{4.1}$$

avoiding having to take the maximums over all the possible values of p and q, which is instead required for the resolution of the full problem (1.2)–(2.8) with anisotropic velocity $\widetilde{c}$. An analogous comment holds for H_q. Last, from the numerical point of view, the use of this simplification brings another fundamental consequence: when we apply the numerical schemes to solve (2.23), we are considering, formally, a problem with bounded velocities $\max\{|H_p|, |H_q|\} \le 1$. This implies that we can choose the following CFL condition:

$$\lambda := \max\left\{\frac{\Delta t}{\Delta x}, \frac{\Delta t}{\Delta y}\right\} \le \frac{1}{2}\max\{|H_p|^{-1}, |H_q|^{-1}\}, \tag{4.2}$$

using the relation (4.1), which is a less restrictive condition with respect to the original one coming from the full problem (1.2)–(2.8).

In the following, we will use the same notations introduced in section 3, except for the number of time steps N_T, which will be replaced by the total number of iterations N_i used by the scheme, since now we are looking for an asymptotic solution (in some stationary sense). The maximum number of iterations, which is fixed at the beginning of the procedure, will be denoted by $N_{\max}$.

Let us give some details on the precise numerical implementation, commenting on the main procedures involved in (the sketched) Algorithm 4.1.

Let us set the parameters of the simulation, which are the power μ in (2.1), the number of iterations K_{reg} of the heat equation for the Gaussian filter, the tolerance *tol* > 0 of the

Algorithm 4.1 Segmentation via the LS Method.

```
Input: $\mu$, $K_{reg}$, $tol$, $N_{\max}$, $I$, $u_0$
  $E^0 = 1$, $n = 0$
  regularize the matrix $I$ (apply the Gaussian filter)
  compute the velocity matrix $c$ using (2.1) or (2.2)
  store the position of the front in the matrix $F^0$
  while ($E^n \geq tol$) and ($n < N_{\max}$) do
    Step 1: compute the modified velocity matrix $\widetilde{c}^n$ using (2.8)
    Step 2: update the solution $u^n \to u^{n+1}$
    $n = n + 1$
    Step 3: store the front $F^n$
            compute the error $E^n$
  end while
  $N_i = n$
Output: $N_i$, $u^{N_i}$.
```

stopping criterion, the amplitude of the pixels $(\Delta x, \Delta y)$, and, subsequently, the time step Δt according to the CFL condition (4.2).

Then, at each iteration $n = 0, \ldots, N_i$, which has to be interpreted in the sense "until convergence" (note that N_i is not known a priori, but depends on the stopping criterion described in Step 3 and can be equal to N_{max} in case of not convergence of the scheme), we repeat the following steps.

Step 1. For $i = 0, \ldots, N_y$, $j = 0, \ldots, N_x$, with $(N_x{+}1)\times(N_y{+}1)$ the size of the input image, we precompute the matrix $\widetilde{c}(x_j, y_i, u_{i,j}, D_x u_{i,j}, D_y u_{i,j})$ at the beginning of each iteration using central finite difference approximations for the first-order derivatives $D_x u_{i,j}$ and $D_y u_{i,j}$. Note that the quantities only depend on (i, j) also through u. Clearly, this method is valid only as long as the representation function u remains smooth at all the level sets, and it should be justified in the case of singular edges (although we will not pursue this precise matter). Moreover, in general the point

(4.3)
$$(x_{j_u}, y_{i_u}) := \left(x_j - d(u_{i,j}) \frac{D_x u_{i,j}}{\sqrt{(D_x u_{i,j})^2 + (D_y u_{i,j})^2}}, y_i - d(u_{i,j}) \frac{D_x u_{i,j}}{\sqrt{(D_x u_{i,j})^2 + (D_y u_{i,j})^2}} \right)$$

is not a point of the grid (x_j, y_i).

To reconstruct the correct value (or at least a reasonable approximation), there exist different possible implementations. One example is a simple *bilinear reconstruction* from the neighboring values

$$\mathcal{N}_u := \left\{ \left(x_{\lfloor j_u \rfloor}, y_{\lfloor i_u \rfloor}\right), \left(x_{\lceil j_u \rceil}, y_{\lceil i_u \rceil}\right), \left(x_{\lceil j_u \rceil}, y_{\lfloor i_u \rfloor}\right), \left(x_{\lfloor j_u \rfloor}, y_{\lceil i_u \rceil}\right) \right\}, \tag{4.4}$$

where we have used the notation

$$\lceil j_u \rceil := j - \left\lceil \frac{x_{j_u} - x_j}{\Delta x} \right\rceil \quad \text{and} \quad \lfloor i_u \rfloor := i - \left\lfloor \frac{y_{i_u} - y_i}{\Delta y} \right\rfloor, \tag{4.5}$$

with the other cases following an analogous definition. Another possibility, which we have used in the numerical tests since it seems to give nicer results in terms of the shape of the approximate representation u, consists in taking as (x_{j_u}, y_{i_u}) the point such that

$$|u_{i_u,j_u}| := \min_{(x_j,y_i)\in\mathcal{N}_u} |u_{i,j}|. \tag{4.6}$$

Note that this construction is well defined only if $|\nabla u_{i,j}| \neq 0$. Therefore, we define the updated velocity matrix as

$$\widetilde{c}^n_{i,j} := \begin{cases} c_{i_u,j_u} & \text{if } |\nabla u^n_{i,j}| \neq 0, \\ c_{i,j} & \text{otherwise,} \end{cases} \tag{4.7}$$

and we use $\widetilde{c}^n$ as an isotropic velocity in the next step.

Step 2. We approximate the problem (2.23) using the AF scheme (3.1), with the local Lax–Friedrichs scheme (3.6) as S^M and the Lax–Wendroff scheme (3.8) as S^A. From now on, we will refer to this AF implemented scheme with the acronym AF-LW.

We add homogeneous Neumann boundary conditions to the problem (2.23) in all our experiments in order to not alter the average intensity of the image.

Step 3. In this last step, we describe how to approximate the front Γ_t. Since Γ_t is a curve, it is composed by points that are not all grid points belonging to our mesh. Hence, in order to approximate the position of the front at each time step t, we consider a neighborhood θ_δ of the front Γ_t of radius $\delta = \max\{\Delta x, \Delta y\}$. In this way, stopping the evolution as soon as the front ceases to move will be equivalent to requiring that the neighborhood θ_δ ceases to move. In order to apply that procedure, at each iteration n we store the values of the points (x_j, y_i) such that $u^n_{i,j}$ changes sign in a matrix $F^n_{i,j}$ (we use the closest points on the grid, which are $(i, j \pm 1)$ and $(i \pm 1, j)$), and set $F^n_{i,j} = 0$ otherwise. In this way, we automatically store the disposition of the front with an error of order $\delta = \max\{\Delta x, \Delta y\}$; i.e., we approximate the position of the front, as desired.

We will continue to do that at each iteration until the matrix F at two consecutive iterations will be "close enough." For this reason, we consider two stopping rules:

$$E_\infty := ||u^{n+1} - u^n||_{L^\infty(\theta_\delta)} = \max_{i,j} |F^n_{i,j} - F^{n-1}_{i,j}| < \tau, \tag{4.8}$$

where $\tau > 0$ is the prescribed tolerance a priori chosen, and

$$E_1 := ||u^{n+1} - u^n||_{L^1(\theta_\delta)} = \Delta x \Delta y \sum_{i,j} |F^n_{i,j} - F^{n-1}_{i,j}| < \tau, \tag{4.9}$$

where now a dependence on the discretization parameters appears.

In our implementation, we use one of the two *stopping rules* introduced above combined with a condition on the number of allowed iterations (i.e., $n < N_{max}$).

5. Numerical simulations.

In this section, we present a series of numerical experiments on both synthetic and real images, comparing the results obtained by the AF scheme with those obtained by the simple monotone scheme and a high-order scheme which uses the total variation diminishing (TVD) Runge–Kutta (RK) of third order in time and the WENO scheme of second/third order in space. In more detail, for the WENO scheme we use the same efficient implementation suggested in [23, Remark 1, page 2130], which we extended to the 2D case, adding the improvement presented in [3], which consists in choosing $\sigma = \Delta x^2$ instead of $\sigma = 10^{-8}$, which is the value used in the original paper [23] by Jiang and Peng. Note that we used this scheme instead of the third- to fifth-order scheme used in the numerical tests of [23] for comparison reasons; otherwise, it would be not comparable by order. The first aim is to show the possible improvements of the modified model with respect to the classical formulation. In fact, after extensive numerical simulations, we noticed that the classical model is not well defined when using high-order schemes since they can produce heavy oscillations as soon as the Lipschitz constant of the representation function becomes too big. This effect causes the stopping rule to be practically ineffective (independently of the norm used) in most cases, particularly when using the AF scheme or the WENO scheme, whereas the simple monotone scheme seems to always give stable results. Note that, when the singularity develops, the representation function becomes more and more vertical as time flows.

The numerical tests illustrated in this section will compare the results also varying the initial datum or varying the norm for the stopping rule defined in (4.8) and (4.9). Moreover, for synthetic images, we also vary the space steps Δx and Δy, which we consider equal to each other ($\Delta x = \Delta y$), and, therefore, the total number of pixels.

Now let us specify the initial condition used in each case. When the velocity is defined by the classical model, in the expansion case (Case a) we use the paraboloid

$$u_0(x,y) = \min\left\{x^2+y^2-r^2, \frac{1}{2}r^2\right\}, \tag{5.1}$$

where $r > 0$ is the radius of the initial circle and $\frac{1}{2}r^2$ a value chosen in order to cut the surface from above (therefore, we have a flat surface at the numerical boundary), whereas in the shrinking case (Case b) we use the truncated pyramid (with a square or rectangular base depending on the image frame), that is,

$$u_0(x,y) = \min\{2(x-b_x), 2(a_x-x), 2(y-b_y), 2(a_y-y), -0.2\}, \tag{5.2}$$

where $[a_x, b_x] \times [a_y, b_y]$ is the frame of the image, -0.2 is the value at which we truncate the pyramid, and 2 is the steepness of the faces of the surface (the smaller the value, the less steep the faces). By this choice, we use a slightly more regular front with respect to the discontinuous representation that simply changes value crossing the frame of the image, still being able to keep the whole surface outside the region occupied by the objects to segment.

For the modified velocity $\widetilde{c}$, in the expansion case (Case a) we consider two different initial data: the paraboloid as in (5.1) (Datum 1) or the following signed distance function (Datum 2),

$$u_0(x,y) = dist\{(x,y), \Gamma_0^1\}, \tag{5.3}$$

where Γ_0^1 is the usual circle centered in $(0,0)$ with radius $r = 0.5$ unless otherwise stated. Instead, for the shrinking case (Case b) we only use the signed distance function

$$u_0(x,y) = dist\{(x,y), \Gamma_0^2\}, \tag{5.4}$$

Table 1

Summary of the initial conditions considered in the numerical tests.

	c	$\widetilde{c}$
Expansion case (Case a)	u_0 as paraboloid (5.1)	Datum 1: u_0 as paraboloid (5.1) Datum 2: u_0 as distance from Γ_0^1 (5.3)
Shrinking case (Case b)	u_0 as truncated pyramid (or tent) (5.2)	u_0 as distance from Γ_0^2 (5.4)

with Γ_0^2 representing the frame of the image. We summarize the different initial conditions in Table 1.

In order to give a quantitative evaluation of the performances in addition to the qualitative analysis, in the tables we compare the results in terms of *number of iterations* N_i and *relative error in pixels*, defined as

$$P\text{-}Err_{rel} = \frac{|P_{ex} - P_a|}{P_{ex}}, \tag{5.5}$$

where P_{ex} and P_a are the number of pixels inside the exact and approximated boundaries of the object(s), respectively. Note that we can compute the "exact" object only if the background is really smooth (in the synthetic cases, it is always uniform), because we usually use a comparison with a "threshold" for the values of $I(x, y)$ in order to select the regions occupied by the object (exact object), whereas for the approximated object we will count the pixels for which $u_{i,j} \leq 0$. Moreover, we measure the error also with a closely related quantity, that is,

$$P\text{-}Err_1 = |P_{ex} - P_a|\Delta x \Delta y, \tag{5.6}$$

in order to show some dependence on the discretization parameters.

If the schemes do not converge in the fixed maximum number of allowed iterations $N_{\max}$, we will put a "–" inside the tables, in place of N_i. For all our tests, we will set $N_{\max} = 2000$. Moreover, in case the front does not stop correctly on the boundary of the object, thus giving an unstable and unusable result, we will put an "X" in correspondence of the errors column. For each test, we specify all the values of the parameters involved (μ, K_{reg}, tol, and $\Delta x = \Delta y$ or $\#Nodes$), the norm used in the stopping rule, and the chosen velocity function. For all the numerical tests presented in this paper, we use CFL number $\lambda = \max\left\{\frac{\Delta t}{\Delta x}, \frac{\Delta t}{\Delta y}\right\} = \frac{1}{2}$, $K = 1$ in the formula (3.14) for the computation of ε, and the velocity function c_1 defined in (2.1) for the classical model, referring to it simply as the classical c. All the numerical tests have been implemented in language C++, with plots and computation of the errors in MATLAB. The computer used for the simulations is a Notebook Asus F556U Intel Core i7-6500U with speed of 2.59 GHz and 12 GB of RAM.

5.1. Synthetic tests. Let us begin by a simple synthetic example. The main goal here is to compare the behavior of the three schemes with respect to the use of the classical velocity c and the modified one $\widetilde{c}$, and the performances varying the number of grid nodes.

Test 1. Rhombus. For this first test, we perform the simulations only in the case of an expansion since the results do not vary much in the shrinking case. The rhombus considered is defined by the equation

$$\frac{|x|}{2} + |y| = \frac{3}{4}, \qquad (x, y) \in [-2, 2]^2, \tag{5.7}$$

which produces a final front visible in Figure 1 for each scheme using the velocity $\widetilde{c}$. As clearly visible, this rhombus presents some heavily marked corners, which causes some serious troubles when using the filtered scheme or the WENO scheme for the solution of the classical model. In fact, looking at Table 2, we can see that the two high-order schemes achieve convergence, in the sense of the iterative stopping rule (4.8), but after having lost track of the boundary (the front overcomes the edge of the object and then keeps expanding). This happens also varying the number of grid nodes (from 102 to 402). On the contrary, the monotone scheme converges and the two errors reported decrease by refining, still using the same parameters μ, K_{reg}, and the same tolerance *tol*. The difficulties of the high-order schemes with the classical velocity c are clearly visible looking at Figure 2, in which the contour plots of the representation function obtained by the three schemes are visible: the evolution of the level sets for the monotone scheme expands and finally coincides at the final time (last row on the left) with the boundary of the object. Instead, for the two high-order schemes we can see different times in order to show well when the oscillations on the left and right edges begin to increase, causing the loss of the rhombus boundary in the approximation. For the modified velocity $\widetilde{c}$, the behavior of the schemes is different; see the contour plots in Figure 3. Note that the yellow level set on the last row, particularly for the WENO scheme, is not the 0-level set. The effects of the modified velocity on the evolution are clear comparing Figures 2 and 3. In the first case, all the level sets expand towards the boundary of the rhombus, eventually collapsing onto each other. This gives rise to a discontinuity around the front, which causes instabilities for high-order schemes. On the contrary, using $\tilde{c}$, these schemes are stable since the level sets remain equally spaced around the 0-level set and the representation function is still Lipschitz continuous. Looking at Table 3, we can observe that the monotone scheme converges in a lower number of iterations N_i and with lower errors with respect to the corresponding results obtained by using the classical velocity and reported in Table 2. Note that we used the same parameters, tolerance and initial datum (the paraboloid, Datum 1), for both velocities to make a fair comparison. The AF-LW scheme converges and always gives better results in terms of both errors and number of iterations with respect to the monotone scheme with c or $\widetilde{c}$. Also the WENO scheme converges in a lower number of iterations with respect to the monotone one, even if with greater errors. This is due to the fact that the WENO scheme needs to recompute the velocity $\widetilde{c}$, which makes the errors accumulate (since we use RK of third order, we need to recompute $\widetilde{c}$ three times). Hence, all three schemes benefit from the new definition of velocity and the AF-LW scheme with $\widetilde{c}$ gives the best performances. Regarding the speed of the schemes, looking at Table 4, which contains the CPU times in seconds related to the three schemes needed to obtain the results reported in Table 3, the AF-LW scheme needs a longer CPU time with respect to the monotone one, as expected. However, the AF-LW scheme is faster than the WENO scheme and its best performances are reached in a short time, at most about 40 seconds for the last refinement of the grid, compared to the 1028.51 seconds necessary for the WENO scheme.

A more correct analysis can be made by decreasing the tolerance *tol* together with the

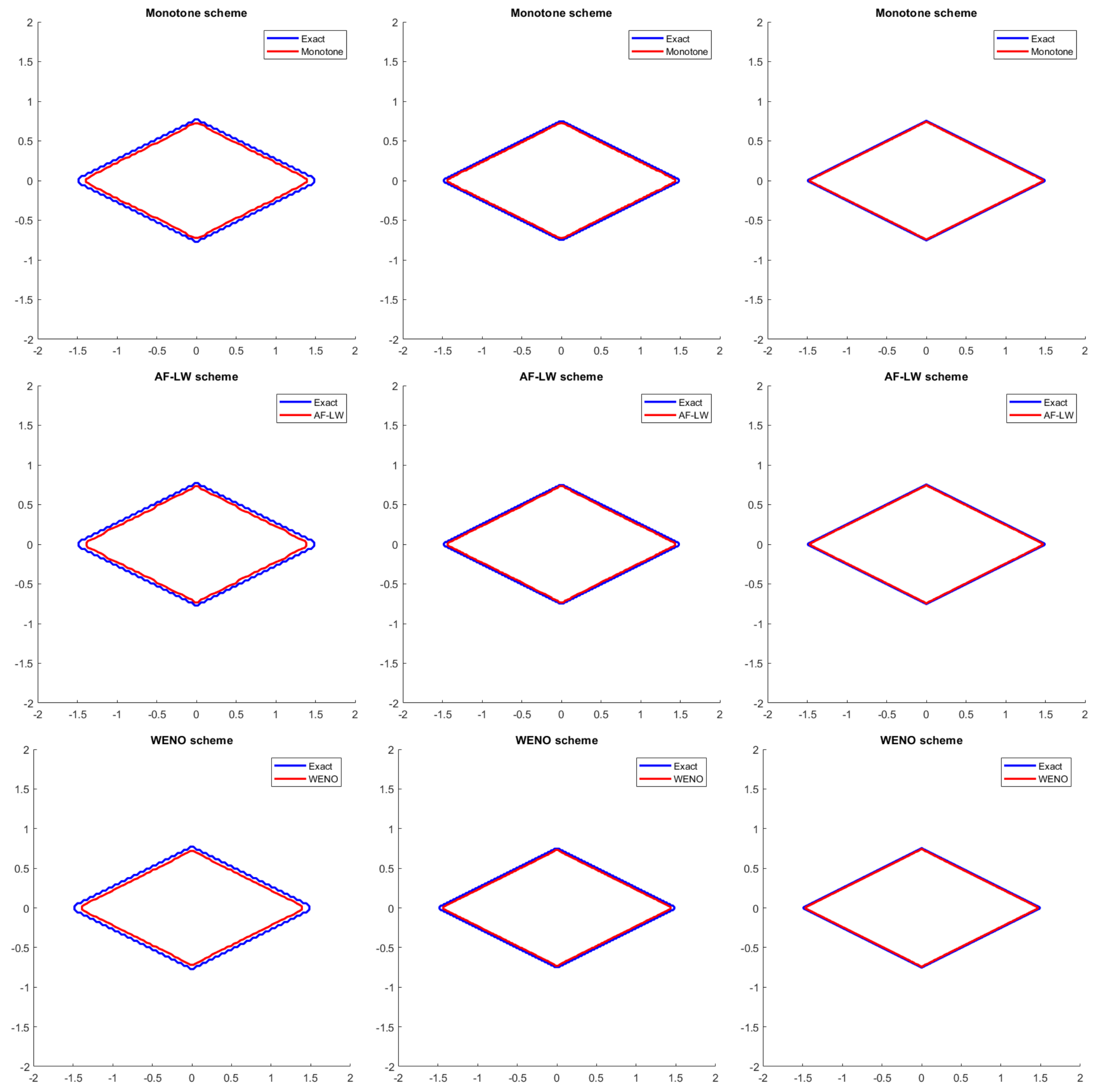

Figure 1. *Test* 1a *with Datum* 1. *Plots of the final front obtained by the monotone scheme (top), the AF-LW scheme (middle), and the WENO scheme (bottom) with velocity* $\widetilde{c}$ *and the parameters reported in Table* 3.

number of nodes $\#Nodes$. In Tables 5, 6, and 7, we analyze the behavior of the schemes with respect to the initial data. Looking at Table 5, we can note that the monotone scheme is not influenced by the change of the initial datum in terms of number of iterations. In fact, comparing the two columns related to N_i, only one iteration in the last row is different. With respect to the errors, small changes are visible, with a small improvement using Datum 2 for the last two rows. The AF-LW scheme (Table 6) has better performances with lower (or equal) errors and lower (or equal) N_i when using Datum 2. For the WENO scheme (see Table 7), no

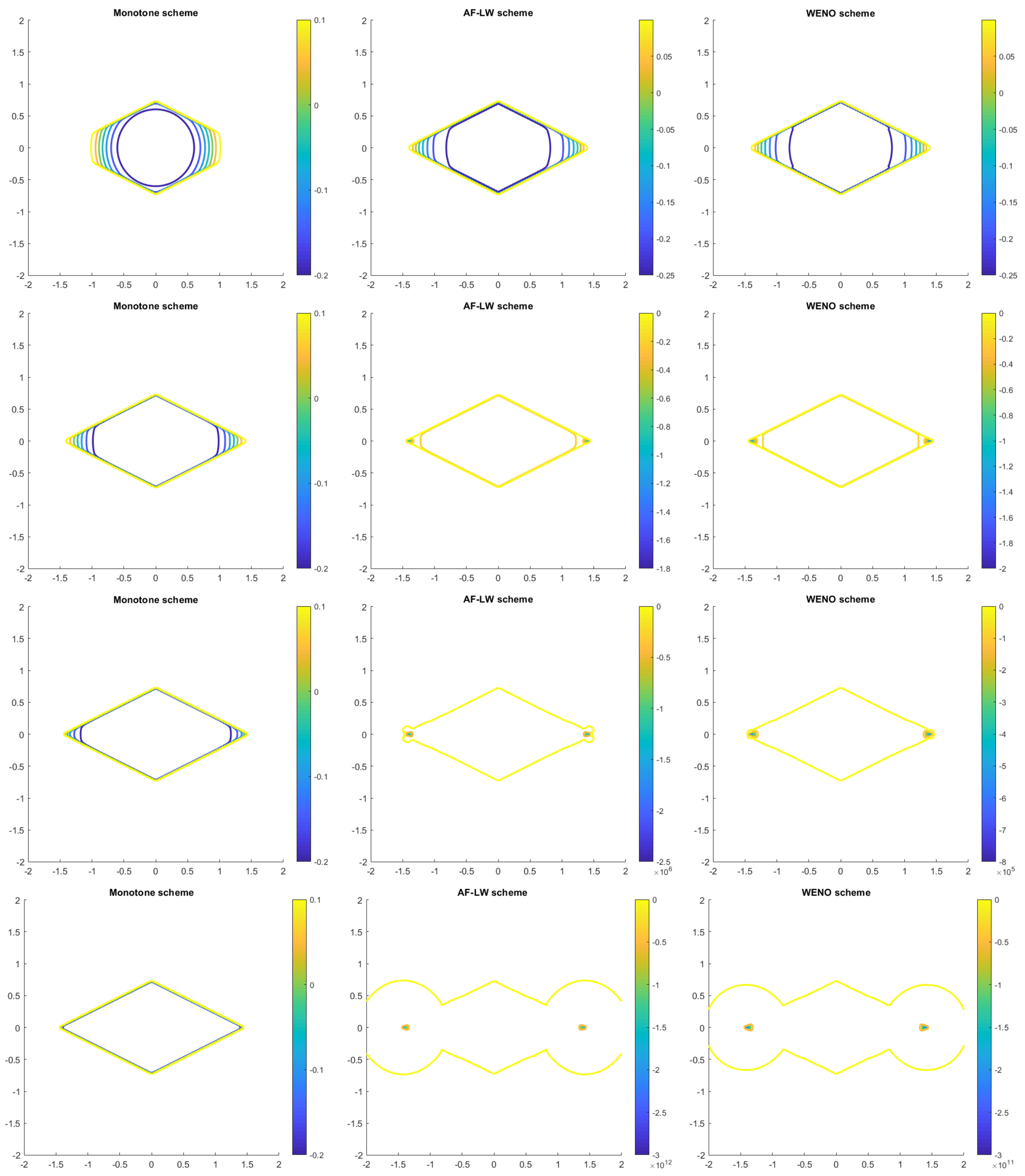

Figure 2. *Test* 1a *with Datum* 1. *Contour plots of the representation function obtained by the monotone scheme (left) at* $N_i = 40$, 80, 120 *and at final time, and by the AF-LW scheme (middle) and the WENO scheme (right) at* $N_i = 80$, 100, 160, 220, *using velocity* c *with* $\#Nodes = 202$.

changes are visible in terms of N_i for all the refinements and both the initial data. In terms of errors, only a small change in the last row is visible, with a slight preference for Datum

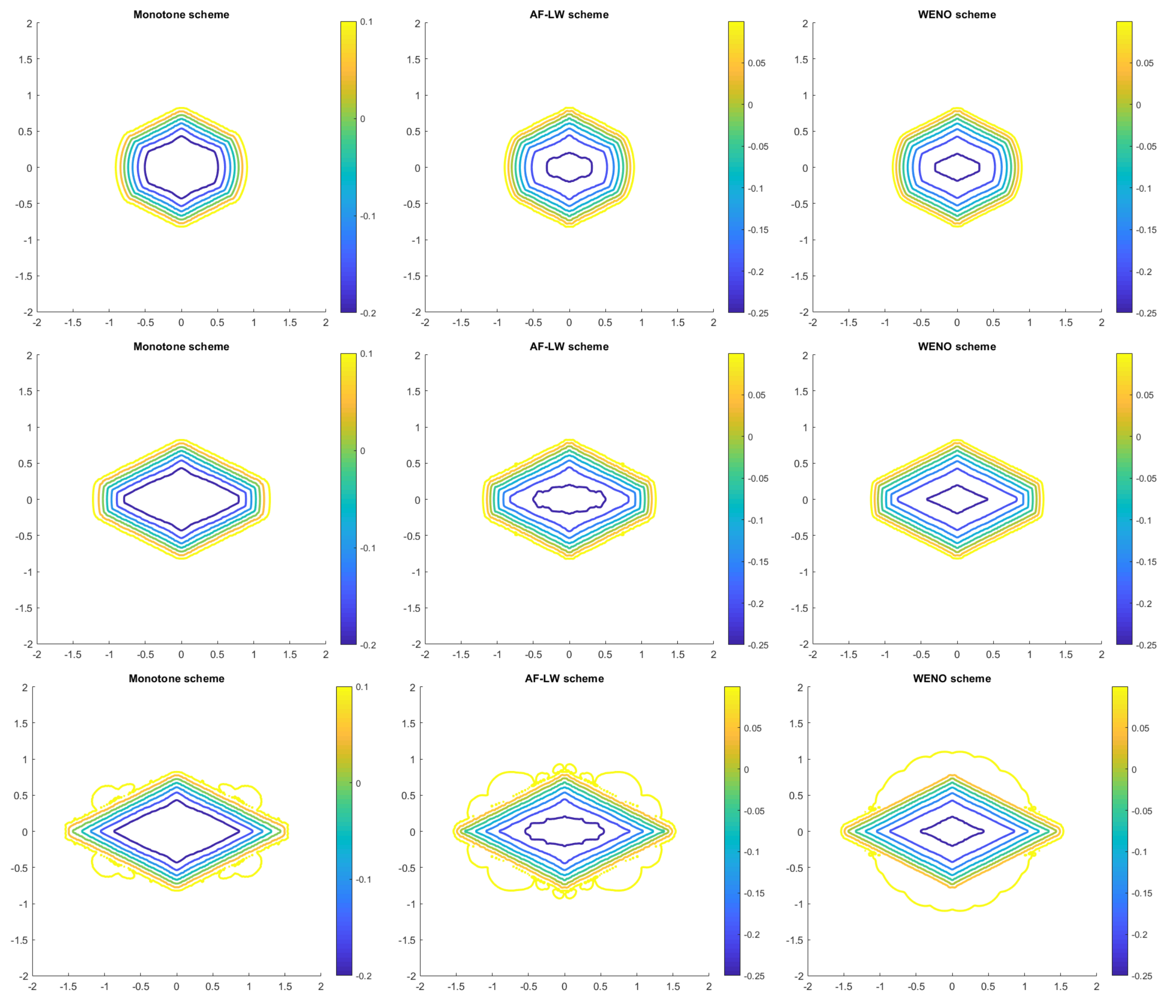

Figure 3. *Test* 1a *with Datum* 1*. Contour plots of the representation function obtained by the monotone scheme (left), the AF-LW scheme (middle), and the WENO scheme (right) at* $N_i = 30$*,* 60 *and at final time, using velocity* $\widetilde{c}$ *with* $\#Nodes = 202$*.*

Table 2

Test 1a*. Errors and number of iterations using the* L^∞ *norm, Datum* 1*, and the parameters* $\mu = 2$*,* $K_{reg} = 0$*,* $tol = 0.0005$*, varying the number of nodes. Best results are in bold.*

c	Monotone			AF-LW			WENO		
$\#Nodes$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$
102	**84**	**0.1025**	**0.2321**	213	X	X	217	X	X
202	**152**	**0.0526**	**0.1172**	394	X	X	399	X	X
402	**288**	**0.0265**	**0.0593**	745	X	X	669	X	X

1. For all the schemes, the errors decrease when we refine the grid, as expected. Comparing the three tables, Tables 5, 6, and 7, we can note that the AF-LW scheme always gets better results in terms of number of iterations and errors with both initial data with respect to the

Table 3
Test 1a. *Errors and number of iterations using the* L^∞ *norm, Datum* 1*, and the parameters* $\mu = 2$*,* $K_{reg} = 0$*,* $tol = 0.0005$*, varying the number of nodes. Best results are in bold.*

$\widetilde{c}$	Monotone			AF-LW			WENO		
$\#Nodes$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$
102	50	0.0748	0.1694	48	**0.0693**	**0.1568**	**47**	0.0886	0.2008
202	100	0.0427	0.0950	**96**	**0.0363**	**0.0808**	**96**	0.0469	0.1045
402	199	0.0208	0.0466	**196**	**0.0203**	**0.0454**	**196**	0.0240	0.0537

Table 4
Test 1a. *CPU times in seconds related to Table* 3.

$\#Nodes$	Monotone	AF-LW	WENO
102	0.19	0.64	4.75
202	1.35	5.61	68.38
402	9.77	40.20	1028.51

monotone scheme, except for the last refinement ($\#Nodes = 402$) with Datum 2, in which the monotone scheme seems to be a little bit better, due to round off errors. The WENO scheme get always the greatest errors using both initial data, for the reasons explained before, in a number of iterations which differ from those of the AF scheme for at most one iteration in some cases. This simple synthetic example illustrates very well the limits of a high-order scheme like the WENO one, when some singularities occur, and how the proposed AF scheme is able to deal with them.

Table 5
Test 1a. *Errors and number of iterations varying* $\#Nodes$ *(*L^∞ *norm). Best results are in bold.*

$\widetilde{c}$	Monotone			Datum 1			Datum 2		
$\#Nodes$	tol	μ	K_{reg}	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$
102	0.001	2	0	**50**	**0.0748**	**0.1694**	**50**	0.0776	0.1757
202	0.0005	2	0	**100**	0.0427	0.0950	**100**	**0.0420**	**0.0935**
402	0.00025	2	0	**199**	0.0208	0.0466	200	**0.0199**	**0.0446**

Table 6
Test 1a. *Errors and number of iterations varying* $\#Nodes$ *(*L^∞ *norm). Best results are in bold.*

$\widetilde{c}$	AF-LW			Datum 1			Datum 2		
$\#Nodes$	tol	μ	K_{reg}	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$
102	0.001	2	0	48	0.0693	0.1568	**47**	**0.0658**	**0.1490**
202	0.0005	2	0	**96**	**0.0363**	**0.0808**	**96**	**0.0363**	**0.0808**
402	0.00025	2	0	196	0.0203	0.0454	**195**	**0.0201**	**0.0450**

5.2. Real tests. In this section, we consider real images also coming from biomedical applications. Thanks to the error formulas (5.5) and (5.6), we can give a sort of quantitative evaluation of the performances of the schemes in terms of "pixels error" (i.e., the number of pixels composing the area of the object to be segmented).

Table 7
Test 1a*. Errors and number of iterations varying* $\#Nodes$ *(*L^∞ *norm). Best results are in bold.*

$\widetilde{c}$	WENO			Datum 1			Datum 2		
$\#Nodes$	tol	μ	K_{reg}	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$
102	0.001	2	0	**47**	**0.0886**	**0.2008**	**47**	**0.0886**	**0.2008**
202	0.0005	2	0	**96**	**0.0469**	**0.1045**	**96**	**0.0469**	**0.1045**
402	0.00025	2	0	**196**	**0.0240**	**0.0537**	**196**	0.0242	0.0541

Test 2*. Brain (*340×340 *pixels).* The first real test focuses on a biomedical image of a human brain. For this test, we approximate the relevant "external" boundary of the brain via a front expansion (Case a) or a shrinking (Case b), so that we start from inside or outside as visible in Figure 4.

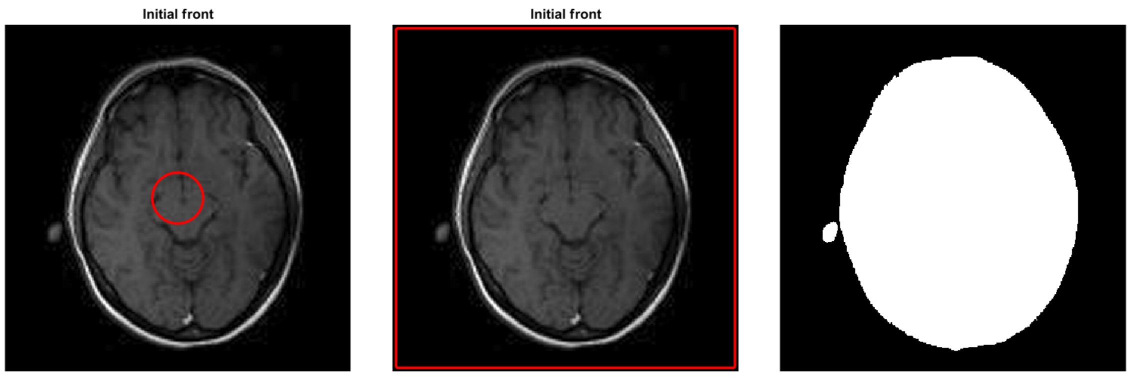

Figure 4. *Test* 2*. From left to right: Initial front for the expansion case (Case* 2a*) composed by a circle of radius* $r = 0.25$*; initial front for the shrinking case (Case* 2b*); mask used for the pixel errors in Case* 2b*.*

Looking at Figures 5 and 6, it is worth noting that the two high-order schemes recognize better the boundary of the object starting from two different initial data using the same tolerance $tol = 0.00001$. The differences and the best resolutions are clearly visible looking at the central part close to the bottom of the brain figures. In fact, the monotone scheme stops too early (see the top-left pictures in both Figures 5 and 6). In order to get better results, a smaller tolerance parameter is necessary for the monotone scheme; see the top-right pictures in both of the considered figures. In that case, the monotone scheme can increase its resolution even if with a higher number of iterations and in any case not more accurate than the high-order schemes. Comparing the AF and the WENO schemes, both obtain more accurate results with a lower number of iterations with respect to the monotone scheme. In more detail, with initial Datum 1, the AF scheme converges in a lower number of iterations; with Datum 2, the WENO scheme uses fewer iterations to get convergence. In any case, the CPU times are really different and the AF scheme is always much faster than the WENO scheme (around 15–20 times faster), as visible looking at Table 8, in which the CPU times in seconds related to the simulations of the three schemes visible in Figures 5 and 6 are reported. Clearly, the AF-LW scheme needs more CPU time with respect to the monotone scheme; this is mainly due to the computation of the smoothness indicators, but is still fast and competitive

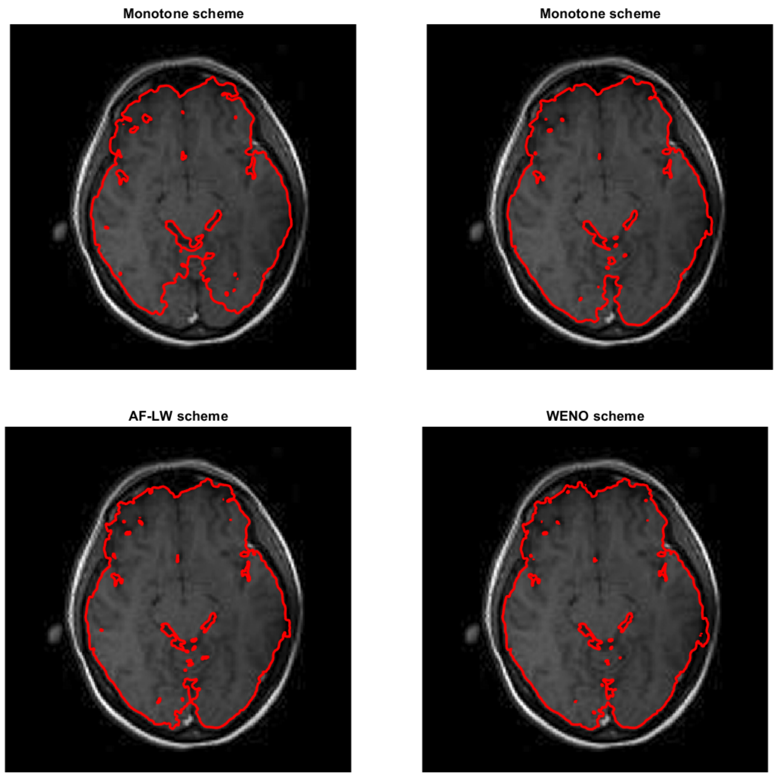

Figure 5. *Test* 2a *with Datum* 1*. On the first row: plots of the final front using the monotone scheme with* $tol = 0.00001$*,* $N_i = 376$ *(left), and with* $tol = 0.000005$*,* $N_i = 468$ *(right). Second row: plot of the final front using the AF-LW scheme,* $N_i = 407$ *(left), and the WENO scheme,* $N_i = 451$ *(right), both with* $tol = 0.00001$*. The four tests have been obtained using the* L^1 *norm in the stopping criterion, with* $\mu = 4$*,* $K_{reg} = 5$*, and velocity* $\widetilde{c}$*.*

since it needs only one minute and half in the worst case (to get a better result). In Case a, hence, the qualitative evaluation is enough to understand which scheme provides the better results in a reasonable CPU time.

Table 8

Test 2a*. CPU times in seconds related to the brain tests (Figures 5 and 6).*

Figure	Mon. (left)	Mon. (right)	AF-LW	WENO
5	12.95	16.54	71.18	1424.18
6	9.63	20.31	91.41	1318.59

For the shrinking case, a quantitative error evaluation is needed in addition to the qualitative one. Looking at Figure 7, we can see that all three schemes recognized the more desired external boundary. Analyzing the errors reported in Table 9, we observe that the AF-LW scheme produces lower values in both errors $P\text{-}Err_{rel}$ and $P\text{-}Err_1$ with respect to the other schemes for both the resolutions considered (input image size 170×170 and 340×340 pixels). The errors in the second row have been computed using the mask visible in Figure 4 on the

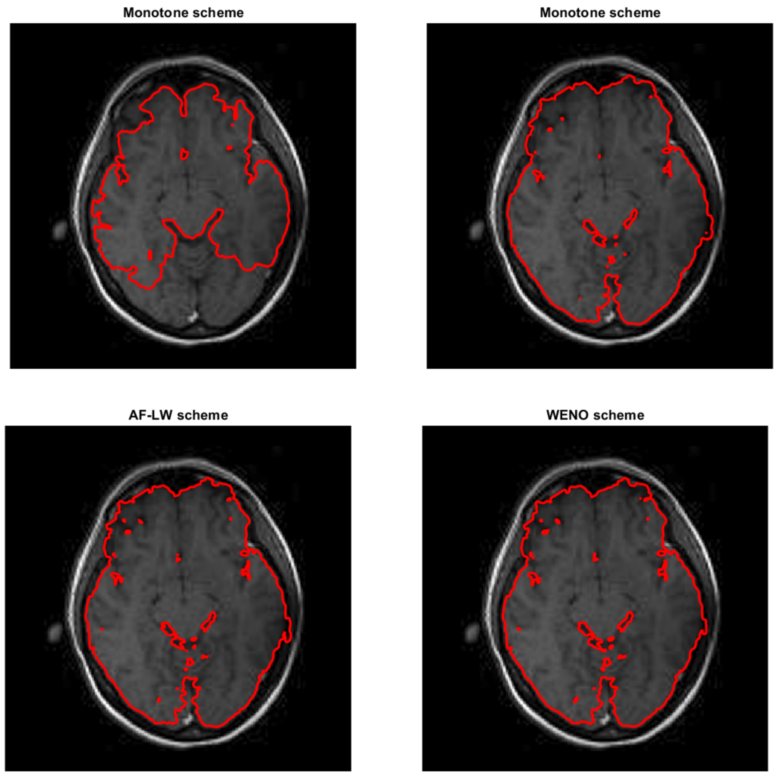

Figure 6. *Test* 2a *with Datum* 2. *First row: plots of the final front using the monotone scheme with* $tol = 0.00005$, $N_i = 264$ *(left), and with* $tol = 0.00001$, $N_i = 495$ *(right). Second row: plots of the final front using the AF-LW scheme,* $N_i = 431$ *(left), and the WENO scheme,* $N_i = 390$ *(right), both with* $tol = 0.00005$. *The four tests have been obtained using the* L^1 *norm in the stopping criterion, with* $\mu = 4$, $K_{reg} = 5$, *and velocity* $\widetilde{c}$.

right and are related to Figure 7. In that test of the brain, Case b, the WENO scheme obtains errors closer to those of the AF scheme, lower with respect to the monotone scheme for both the considered image sizes. Comparing the CPU times with the two different resolutions visible in Table 10, we note that with half the size and less than half the time we can obtain better accuracy with respect to the P-Err_1 error using the AF-LW scheme instead of the monotone one. Comparing only the two high-order schemes, again the CPU times of the WENO scheme are about 11 or 20 times greater than those necessary for the AF scheme.

Table 9

Test 2b. *Errors and number of iterations using the* L^1 *norm,* $\widetilde{c}$, *and the parameters* $tol = 0.00005$, $\mu = 5$, $K_{reg} = 3$, *varying the image size. Best results are in bold.*

	Monotone			AF-LW			WENO		
Image size	N_i	P-Err_{rel}	P-Err_1	N_i	P-Err_{rel}	P-Err_1	N_i	P-Err_{rel}	P-Err_1
170×170	**83**	0.0465	0.2436	87	**0.0439**	**0.2300**	87	0.0448	0.2348
340×340	**228**	0.0118	0.2476	262	**0.0078**	**0.1628**	264	0.0079	0.1648

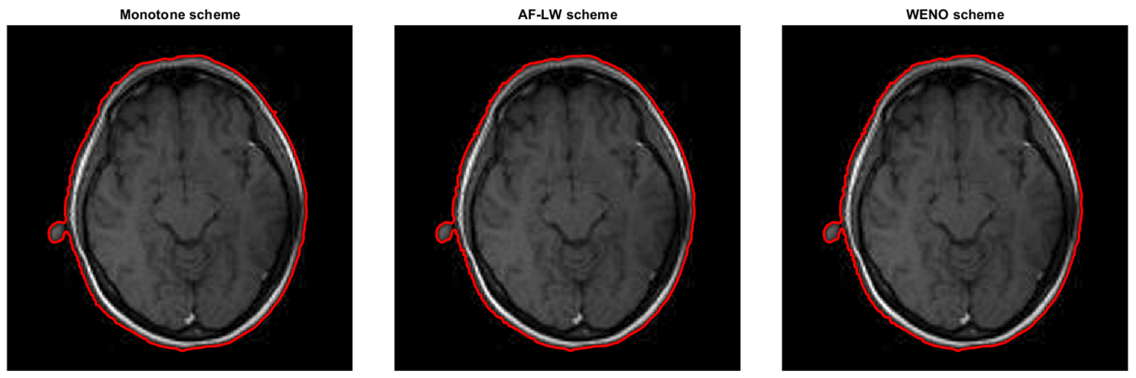

Figure 7. *Test* 2b*. Plots of the final front using the monotone scheme (left), the AF-LW scheme (middle), and the WENO scheme (right), with* $\mu = 5$*,* $K_{reg} = 3$*, and velocity* $\widetilde{c}$*. Image size:* 340×340*.*

Table 10
Test 2b*. CPU times in seconds related to the brain tests (Figure* 7*).*

Image size	Monotone	AF-LW	WENO
170×170	0.80	3.75	44.10
340×340	8.39	46.07	933.90

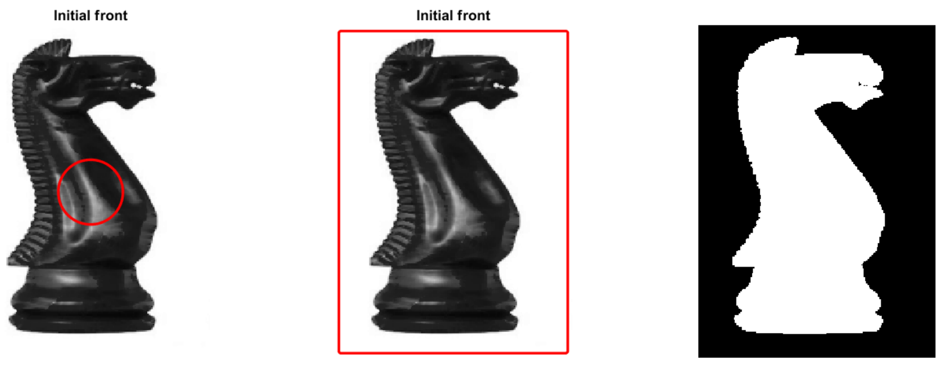

Figure 8. *Test* 3*. From left to right: Initial front for the expansion case (Case* 3a*) composed by a circle of radius* $r = 0.5$*; initial front for the shrinking case (Case* 3b*); mask used for the pixel errors in both cases.*

Test 3*. Horse chess* (184×256 *pixels*)*.* We choose the image of the horse piece in the game of chess visible in Figure 8, and we approximate its boundary from inside (Case a), varying the initial datum, and outside (Case b), always using the modified model with velocity $\widetilde{c}$.

Starting our analysis of the results from the expansion case, looking at Figure 9 we can note some differences between the schemes around the mouth of the horse, at the top at the beginning of the horse's mane, and at the bottom left, i.e., at the end of the horse's mane, where the monotone scheme seems to stop too early. Looking at Figure 10, we note in addition other differences and worse performances of the monotone scheme, due probably to the specularities inside the image. Regarding the errors reported in Table 11, the WENO scheme gets lower errors with Datum 1 and *tol* = 0.00005, even if with a greater number of

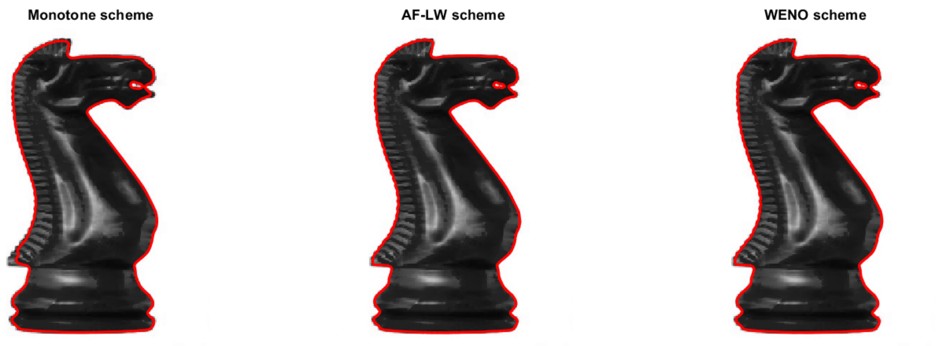

Figure 9. *Test* 3a *with Datum* 1. *Plots of the final front using the monotone scheme,* $N_i = 508$ *(left), the AF-LW scheme,* $N_i = 544$ *(middle), and the WENO scheme,* $N_i = 523$ *(right), with the* L^1 *norm and* $tol = 0.000025$, $\mu = 2$, $K_{reg} = 5$, *and velocity* $\widetilde{c}$.

iterations with respect to the AF scheme, whereas with $tol = 0.000025$ we obtain the inverse situation; i.e., the AF scheme gets lower errors but with more iterations with respect to the WENO scheme. Looking at Table 12, we can note that using the initial Datum 2 the AF scheme always provides the best performances in terms of both errors $P\text{-}Err_{rel}$ and $P\text{-}Err_1$. Regarding the CPU times, Table 13 shows that the WENO scheme is really slow compared to the other schemes, whereas the AF scheme always converges in less than one minute.

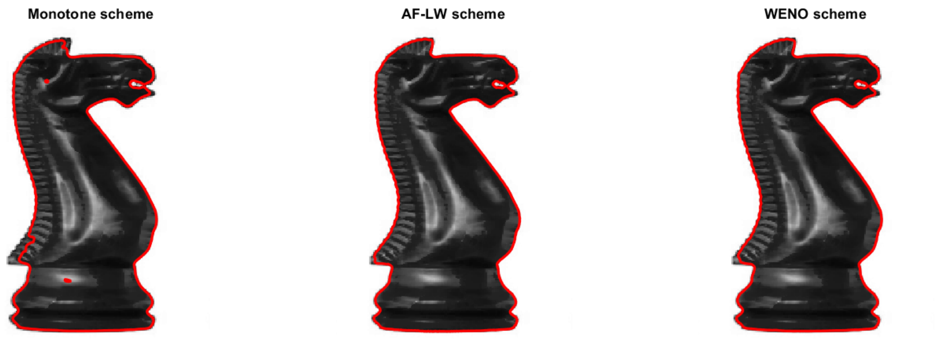

Figure 10. *Test* 3a *with Datum* 2. *Plots of the final front using the monotone scheme* $N_i = 453$ *(left), the AF-LW scheme,* $N_i = 542$ *(middle), and the WENO scheme,* $N_i = 472$ *(right), with the* L^1 *norm and* $tol = 0.00004$, $\mu = 2$, $K_{reg} = 5$, *and velocity* $\widetilde{c}$.

Table 11

Test 3a. *Errors and number of iterations using the* L^1 *norm, Datum* 1, *velocity* $\widetilde{c}$, *and the parameters* $\mu = 2$, $K_{reg} = 5$, *varying the tolerance. Best results are in bold.*

$\widetilde{c}$	Monotone			AF-LW			WENO		
tol	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$
0.00005	433	0.0794	0.6424	**396**	0.0672	0.6424	432	**0.0591**	**0.4788**
0.000025	**508**	0.0599	0.4848	544	**0.0438**	**0.3544**	523	0.0462	0.3744

Analyzing the results in the shrinking case, small differences can be noted looking at Figure 11, particularly between the two high-order schemes. For the monotone scheme, we

Table 12

Test 3a. *Errors and number of iterations using the L^1 norm, Datum* 2*, velocity $\widetilde{c}$, and the parameters $\mu = 2$, $K_{reg} = 5$, varying the tolerance. Best results are in bold.*

$\widetilde{c}$	Monotone			AF-LW			WENO		
tol	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$
0.00008	**382**	0.1031	0.8344	451	**0.0640**	**0.5180**	411	0.0684	0.5536
0.00004	**453**	0.0737	0.5964	542	**0.0501**	**0.4052**	472	0.0565	0.4572

Table 13

Test 3a. *CPU times in seconds related to the chess horse tests (Tables* 11 *and* 12*).*

Datum	*tol*	Monotone	AF-LW	WENO
1	0.00005	7.26	26.22	378.07
1	0.000025	8.78	42.23	462.55
2	0.00008	6.51	37.15	377.47
2	0.00004	8.02	44.16	443.66

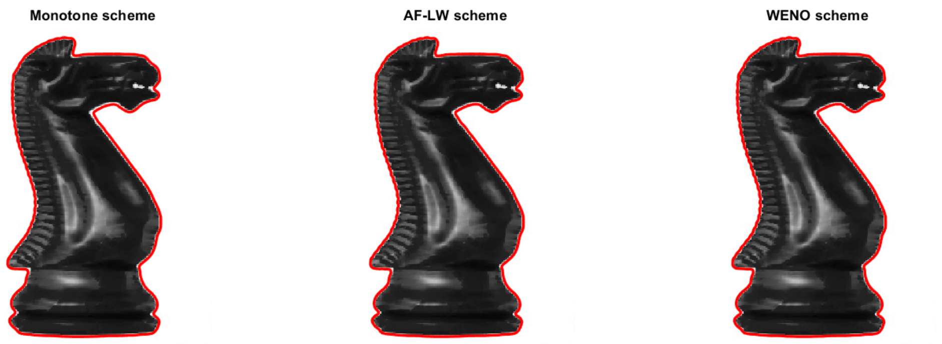

Figure 11. *Test* 3b. *Plots of the final front using the monotone scheme (left), the AF-LW scheme (middle), and the WENO scheme (right), with velocity $\widetilde{c}$ and tol = 0.00005, by using the L^1 norm in the stopping criterion and parameters $\mu = 2$ and $K_{reg} = 3$.*

Table 14

Test 3b. *Errors and number of iterations using the L^1 norm, velocity $\widetilde{c}$, and the parameters tol = 0.00005, $\mu = 2$, $K_{reg} = 3$. Best results are in bold.*

Monotone			AF-LW			WENO		
N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$
187	0.0303	0.2452	197	0.0273	0.2212	196	**0.0262**	**0.2120**

Table 15

Test 3b. *CPU times in seconds related to the horse test (Table* 14*).*

Monotone	AF-LW	WENO
3.14	14.10	195.13

note that the final front stops just before the boundary of the object. So, also in this case, the only qualitative analysis is not enough. A quantitative analysis is shown in Table 14, in which both the high-order schemes are more accurate than the monotone scheme, as expected. In

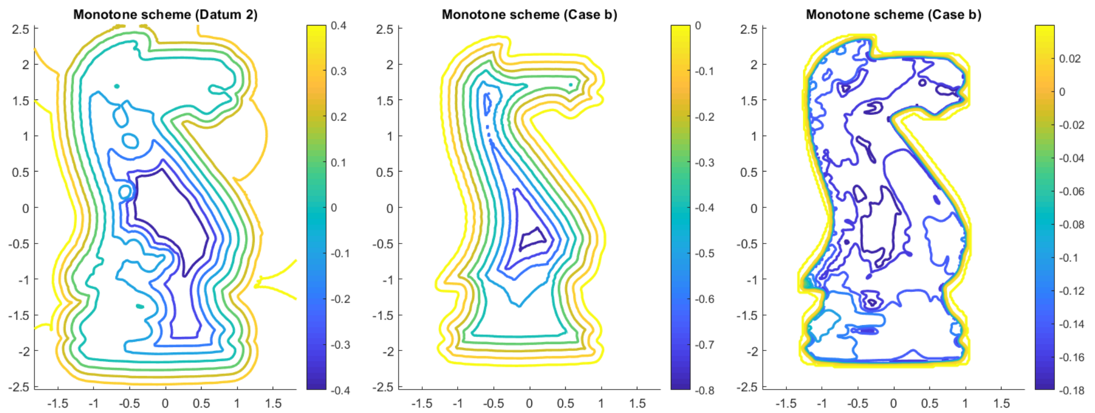

Figure 12. *Test* 3. *Contour plots of the final representations using the monotone scheme with velocity* $\widetilde{c}$*, for Case a with Datum* 2 *(left) and for Case b (middle), and with velocity c for Case b (right).*

this case, WENO is the most accurate scheme, having lower values in both errors, even if with a CPU time around 14 times greater than that of the AF scheme (see Table 15).

In order to show the effectiveness of our implementation of the modified velocity, in Figure 12 we collected the final representations obtained by the monotone scheme. Although in the first case some new fronts arise due to the specularities inside the image, we can still recognize that the gradient of the initial condition is preserved and that the level sets do not collide during the evolution. In particular, in the middle image the distance function to the 0-level set is still clearly visible. The third image is related to the results obtained by using the classical velocity c in Case b, with the same tolerance used with $\widetilde{c}$ and visible in Figure 11. We note that with the classical model oscillations appear throughout the whole horse, even if focusing on the 0-level set the scheme performs well. A better behavior of all the level sets can be obtained but using a greater tolerance tol, e.g., $tol = 0.0005$.

Test 4. *Grains* (300 × 300 *pixels).* In this test, we consider the shrinking front in the presence of multiple separate objects to be segmented. The final fronts obtained by the three schemes are visible in Figure 13, in which almost no differences are visible. Looking at the errors and number of iterations reported in Table 16, we can note that the two high-order schemes are more accurate than the first-order monotone one, with a number of iterations N_i very close between the three schemes; the lowest number is obtained by the AF scheme. In terms of errors, the WENO scheme is the most accurate, even if with a really great CPU time as ever (see Table 17). We report in Figure 14 the contour plots of the initial datum and the final representation obtained by the AF scheme. It is rather interesting to see that, although the representation function has a rather complex evolution and splits in various parts, the final solution is still a distance function to the 0-level set, now composed by different peaks each relative to a single grain.

Test 5. *Geometric shapes* (640 × 480 *pixels).* In some of the previous real tests, we have seen that with a careful tuning of the parameters in the stopping rule, the monotone scheme can get comparable results with respect to those obtained by the high-order schemes, at least from a visible point of view. This is not always possible, especially in more critical situations,

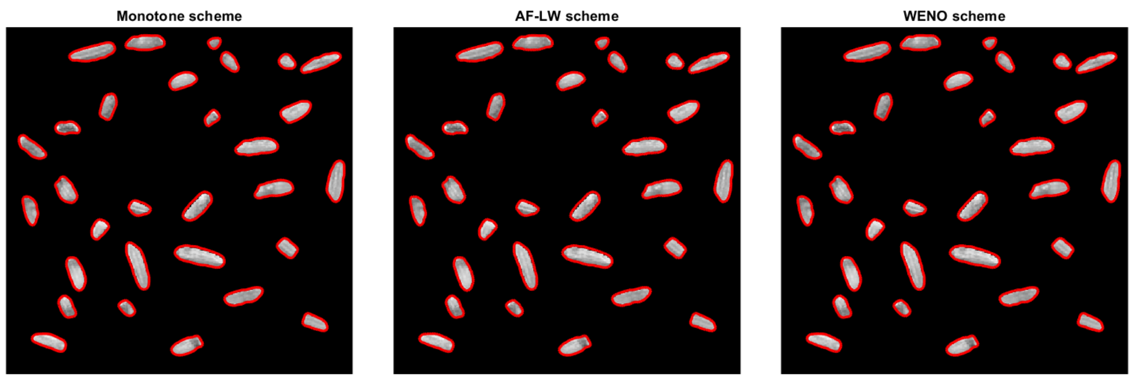

Figure 13. *Test* 4b. *Plots of the final front obtained by the monotone scheme (left), the AF-LW scheme (middle), and the WENO scheme (right), using velocity $\widetilde{c}$ and the parameters reported in Table* 16*.*

Table 16

Test 4b. *Errors and number of iterations with velocity $\widetilde{c}$, using the L^1 norm in the stopping criterion and the parameters $tol = 0.0001$, $\mu = 2$, and $K_{reg} = 2$. Best results are in bold.*

Monotone			AF-LW			WENO		
N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$	N_i	$P\text{-}Err_{rel}$	$P\text{-}Err_1$
312	0.0097	0.0316	**299**	0.0074	0.0240	306	**0.0064**	**0.0208**

Table 17

Test 4b. *CPU times in seconds related to the grains test (Table* 16*).*

Monotone	AF-LW	WENO
8.71	41.26	676.91

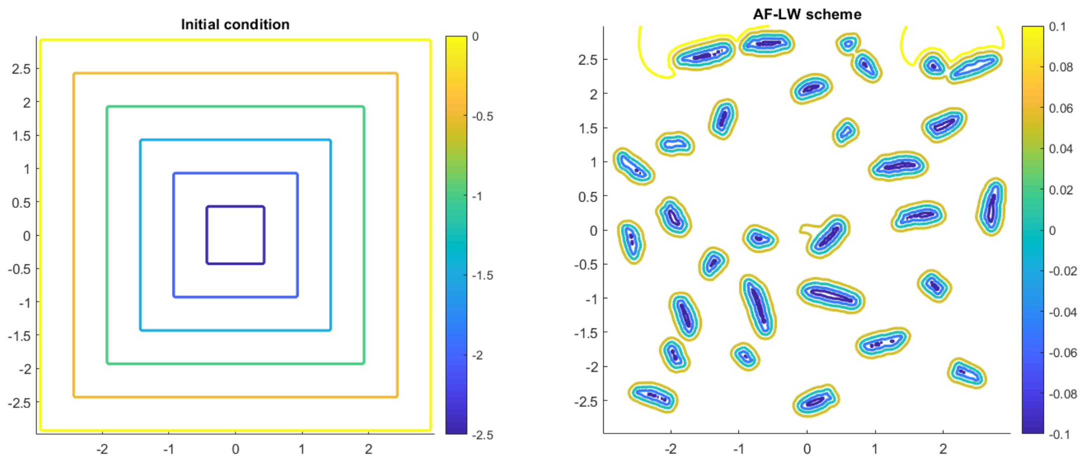

Figure 14. *Test* 4b. *Contour plots of the initial datum (left) and the final representations (right), using the AF-LW scheme with velocity $\widetilde{c}$ and the parameters reported in Table* 16*.*

e.g., when the difference between the background and the objects we want to segment is less marked, as shown in this test. In Figure 15, we reported the initial front in red for the

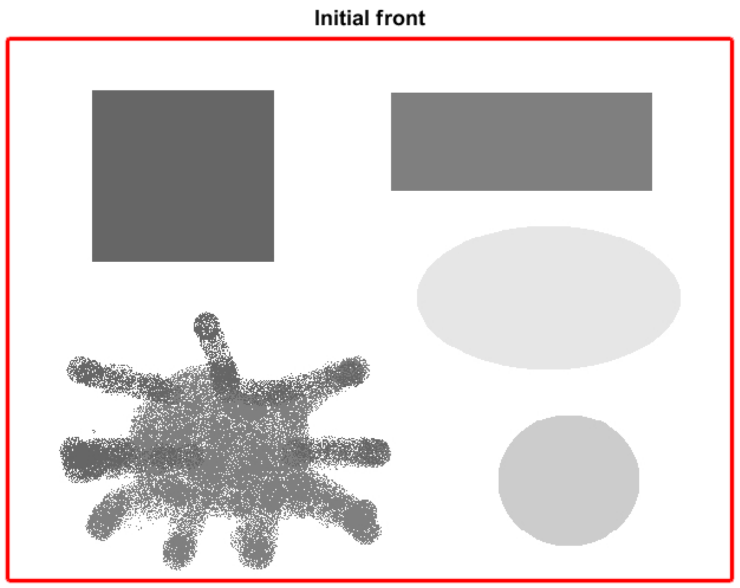

Figure 15. *Test* 5b. *Initial front for the shrinking case (Case* 5b*).*

shrinking case. The more difficult object to detect will be the ellipse, due to the lighter gray levels closer to the white background.

Looking at the final fronts in Figure 16, obtained by varying the tolerance in the stopping criterion, we can note that the AF scheme recognizes very well the boundaries of all the objects, differently from the monotone one. In fact, if we stop the schemes too early, with $tol = 0.0005$, we note that the monotone scheme still has to conclude the recognition of the "spray" shape below on the left. But if we adopt a smaller tolerance, e.g., $tol = 0.0001$ or $tol = 0.00005$, in order to give "more time" to the monotone scheme to achieve all the boundaries, the scheme improves the detection of the spray shape but loses the boundary of the ellipse.

Higher-order approximation of the evolution can clearly give more stability, especially when the contrast is not satisfactory, as in the present situation.

This numerical test clearly shows the difficulties that the monotone scheme can encounter. It needs a really complicated manual tuning of the parameters, and nonetheless not always gives good and reliable results (as in this case), whereas the AF scheme overcomes this difficulty thanks to its properties.

Test 6. *Pneumonia* (191×150 *pixels*). Finally, we conclude our numerical tests considering a Pneumonia image in the expansion case starting from an initial datum here composed by four equal paraboloids (each defined as Datum 1) or cones (defined as Datum 2), placed as visible in Figure 17, with 0-level sets composed by circles of radius $r = 0.125$. In this test, the behavior of the three schemes is confirmed in terms of CPU time, looking at Table 18, and in terms of accuracy, as visible in Figures 18 and 19, where the final fronts obtained by the high-order schemes recognize better the object starting from the two different initial data, Datum 1 and Datum 2, respectively. In fact, in the first figure, Figure 18, it is evident looking

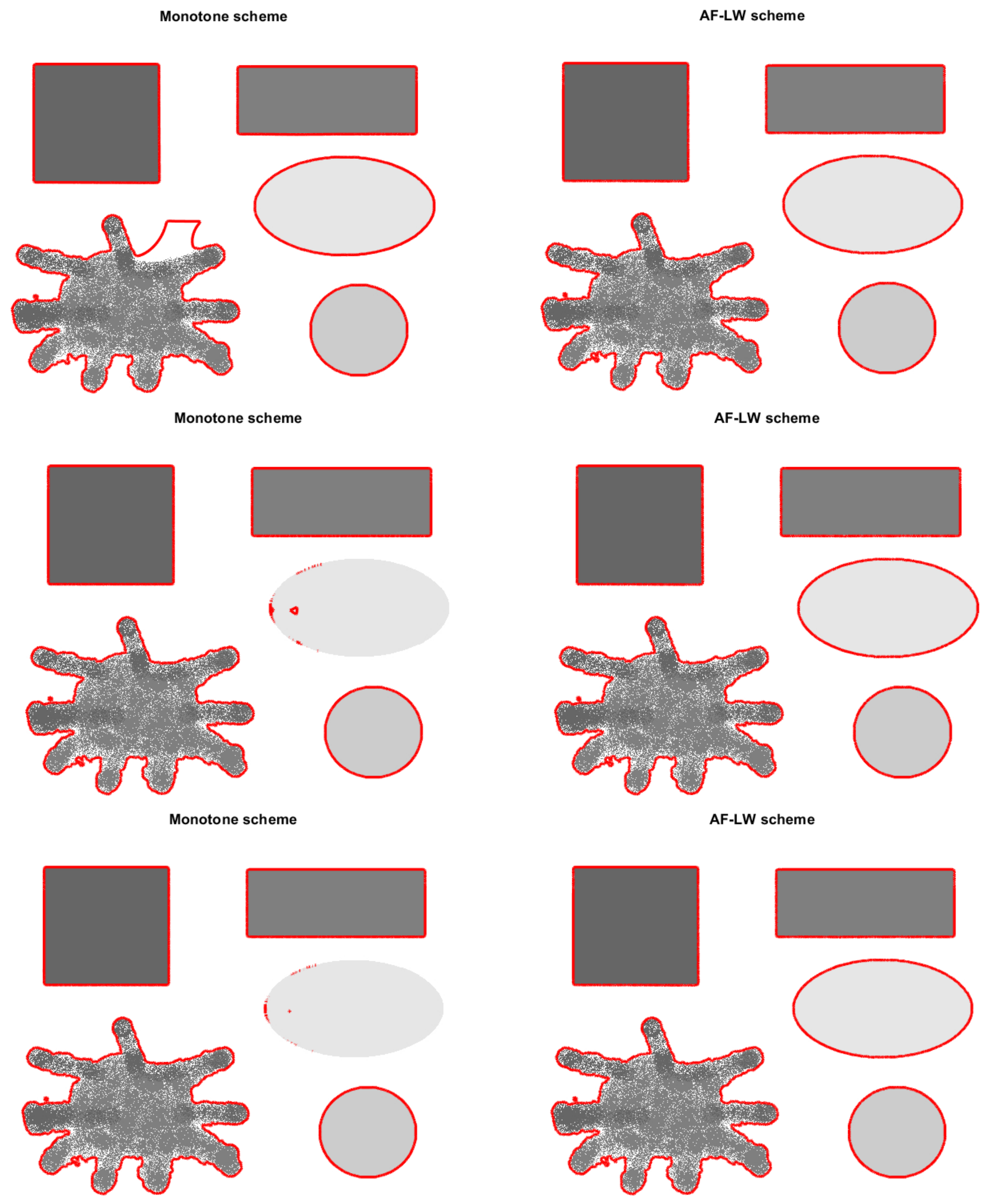

Figure 16. *Test* 5b. *Plots of the final front using the monotone scheme (left) and the AF-LW scheme varying the tolerance (tol* $= 0.0005, 0.0001, 0.00005$*), with* $\mu = 4$*,* $K_{reg} = 1$*, and velocity* $\widetilde{c}$*.*

at the lung on the left; in the second case, Figure 19, it is clear, especially looking at the right lung, mostly for visualizing the better performances of the AF scheme. Moreover, comparing the two figures vertically, focusing on each scheme with a different initial datum, we can note that all three schemes seem to prefer the distance function (Datum 2). The good behavior of the AF scheme and of the modified model in the case of a merging front is confirmed by the contour plots of the final representations, shown in Figure 20. Regarding the WENO scheme (last column), some oscillations arise at some point in the evolution and increase as time flows

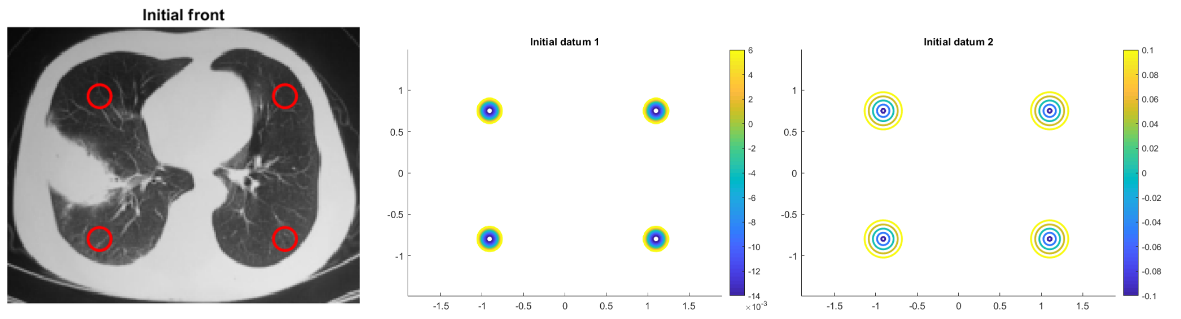

Figure 17. *Test* 6a*. From left to right: Initial front composed of four separate circles of radius* $r = 0.125$*; contour plots of the initial data (Datum* 1 *and Datum* 2*).*

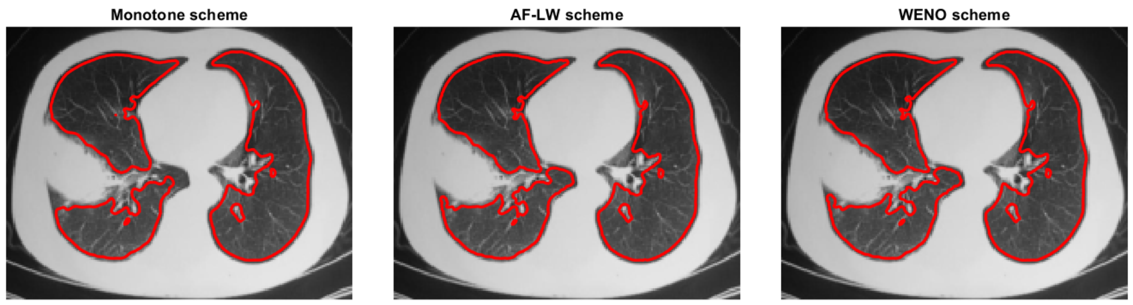

Figure 18. *Test* 6a *with Datum* 1*. From left to right: Plots of the final front using the monotone scheme* $(N_i = 185)$*, the AF-LW scheme* $(N_i = 194)$*, and the WENO scheme* $(N_i = 192)$*, with the* L^1 *norm and* $tol = 0.00001$*,* $\mu = 4$*,* $K_{reg} = 5$*, and velocity* $\widetilde{c}$*.*

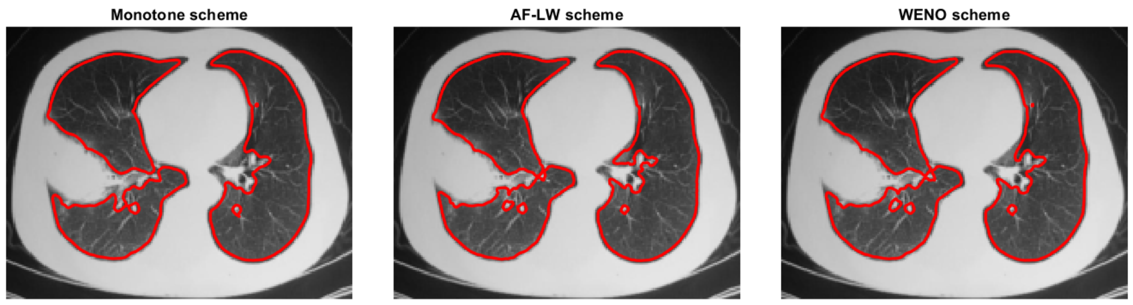

Figure 19. *Test* 6a *with Datum* 2*. From left to right: Plots of the final front using the monotone scheme* $(N_i = 413)$*,* $tol = 0.00001$*, the AF-LW scheme* $(N_i = 589)$ *with* $tol = 0.00001$*, and the WENO scheme* $(N_i = 458)$ *with* $tol = 0.000012$*, all using the* L^1 *norm,* $\mu = 4$*,* $K_{reg} = 5$*, and velocity* $\widetilde{c}$*.*

(see the bottom-left part of the final representation using Datum 2). This is why a slightly greater tolerance has been used in that case to obtain the reported result.

6. Conclusions and perspectives.

In this work, we have proposed a new velocity function for the LS method in order to improve image segmentation and we have extended to two dimensions an adaptive filtered scheme originally developed in one dimension [18]. We have shown that the use of the new velocity function $\widetilde{c}$ allows one to get more accurate results

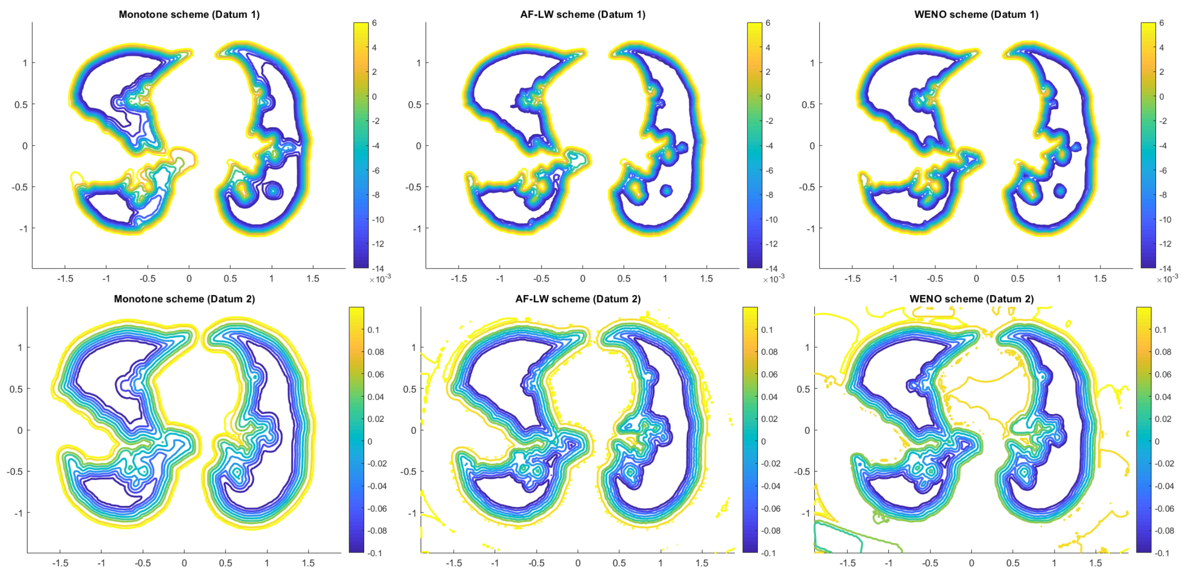

Figure 20. *Test* 6a. *Contour plots of the final representations using the monotone scheme (left), the AF-LW scheme (middle), and the WENO scheme (right). Top: Datum* 1*, bottom: Datum* 2*.*

Table 18

Test 6a. *CPU times in seconds related to the pneumonia tests (Figures* 18 *and* 19*).*

Figure	Monotone	AF-LW	WENO
18	1.80	8.05	83.83
19	5.01	28.91	202.373

stabilizing high-order schemes such as the AF or the WENO schemes, for which the classical velocity introduces some instabilities destroying the convergence. Moreover, the new velocity function can also be applied to the simple monotone scheme, getting better results even in the first-order approximation, and does not require the re-initialization of the front in order to keep an accurate tracking of the 0-level set. From the numerical point of view, the AF scheme is based on two building blocks: a monotone scheme and a high-order scheme. The filter function allows one to couple the two schemes in a rather simple way and easily switches from one scheme to the other according to new smoothness indicators. In terms of CPU time, the AF scheme is less expensive than the WENO scheme and offers a good option to improve the accuracy of the monotone scheme. It is interesting to note that these two changes are rather effective for the segmentation of synthetic and real images according to many simulations; some of them are presented in section 5. A qualitative analysis of the resulting segmentations is illustrated by the pictures of the last section, whereas a more accurate comparison of the schemes is based on a quantitative analysis of the pixel errors. Moreover, the sensitivity of the AF scheme seems to be rather low with respect to the presence of noise and only a few steps of a linear filter are required to obtain the necessary regularization of the gray levels of the input image I. Although in this paper the image segmentation is obtained just using the classical first-order equation, a possible extension to second-order problems can be considered;

this extension is motivated by the inclusion of curvature terms in the evolutive equation, as done in the literature. The analysis of the adaptive filtered scheme to second-order nonlinear equations goes beyond the scope of this paper and will be the object of a further investigation.

REFERENCES

[1] S. Amat, F. Aràndiga, A. Cohen, and R. Donat, *Tensor product multiresolution analysis with error control for compact image representation*, Signal Process., 82 (2002), pp. 587–608.

[2] F. Aràndiga and A. Belda, *Weighted ENO interpolation and applications*, Commun. Nonlinear Sci. Numer. Simul., 9 (2004), pp. 187–195.

[3] F. Aràndiga, A. M. Belda, and P. Mulet, *Point-value WENO multiresolution applications to stable image compression*, J. Sci. Comput., 43 (2010), pp. 158–182, https://doi.org/10.1007/s10915-010-9351-8.

[4] G. Barles, *Solutions de viscositè des équations de Hamilton-Jacobi*, Springer-Verlag, Paris, 1994.

[5] O. Bokanowski, M. Falcone, and S. Sahu, *An efficient filtered scheme for some first order time-dependent Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 38 (2016), pp. A171–A195, https://doi.org/10.1137/140998482.

[6] O. Bokanowski, A. Picarelli, and C. Reisinger, *High-order filtered schemes for time-dependent second order HJB equations*, ESAIM Math. Model. Numer. Anal., 52 (2018), pp. 69–97.

[7] T. Brox and J. Weickert, *Level set based image segmentation with multiple regions*, in Pattern Recognition, C. E. Rasmussen, H. H. Bülthoff, B. Schölkopf, and M. A. Giese, eds., Springer, Berlin, Heidelberg, 2004, pp. 415–423.

[8] X. Cai, R. Chan, and T. Zeng, *A two-stage image segmentation method using a convex variant of the Mumford–Shah model and thresholding*, SIAM J. Imaging Sci., 6 (2013), pp. 368–390, https://doi.org/10.1137/120867068.

[9] E. Carlini, E. Cristiani, and N. Forcadel, *A non-monotone fast marching scheme for a Hamilton-Jacobi equation modeling dislocation dynamics*, in Numerical Mathematics and Advanced Applications, A. B. de Castro, D. Gòmez, P. Quintela, and P. Salgado, eds., Springer, Berlin, 2006, pp. 723–731.

[10] E. Carlini, M. Falcone, and R. Ferretti, *Convergence of a large time-step scheme for mean curvature motion*, Interfaces Free Bound., 12 (2010), pp. 409–441.

[11] E. Carlini, M. Falcone, and R. Ferretti, *Numerical techniques for level set models: An image segmentation perspective*, in Level Set Methods in Medical Imaging Segmentation, A. El-Baz and J. Suri, eds., Taylor & Francis, Boca Raton, FL, 2019.

[12] E. Carlini, M. Falcone, N. Forcadel, and R. Monneau, *Convergence of a generalized fast-marching method for an eikonal equation with a velocity-changing sign*, SIAM J. Numer. Anal., 46 (2008), pp. 2920–2952, https://doi.org/10.1137/06067403X.

[13] V. Caselles, F. Catté, T. Coll, and F. Dibos, *A geometric model for active contours in image processing*, Numer. Math., 66 (1993), pp. 1–31.

[14] V. Caselles, R. Kimmel, and G. Sapiro, *Geodesic active contours*, Int. J. Comput. Vis., 22 (1997), pp. 61–79, https://doi.org/10.1023/A:1007979827043.

[15] L. Cohen, *On active contour models and balloons*, CVGIP: Imag. Understand., 53 (1991), pp. 211–218.

[16] L. Cohen and I. Cohen, *Deformable models for* 3-*D medical images using finite elements and balloons*, in Proceedings of the 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 28, 1992, pp. 592–598.

[17] M. Crandall and P.-L. Lions, *Viscosity solutions of Hamilton-Jacobi equations*, Trans. Amer. Math. Soc., 277 (1983), pp. 1–42.

[18] M. Falcone, G. Paolucci, and S. Tozza, *Convergence of Adaptive Filtered Schemes for First Order Evolutionary Hamilton-Jacobi Equations*, https://arxiv.org/abs/1812.02140, 2018.

[19] M. Falcone, G. Paolucci, and S. Tozza, *Adaptive filtered schemes for first order Hamilton-Jacobi equations*, in Numerical Mathematics and Advanced Applications (ENUMATH 2017), F. Radu, K. Kumar, I. Berre, J. Nordbotten, and I. Pop, eds., Springer, Cham, 2019, pp. 389–398, https://doi.org/10.1007/978-3-319-96415-7_34.

[20] B. D. Froese and A. M. Oberman, *Convergent filtered schemes for the Monge–Ampère partial differential equation*, SIAM J. Numer. Anal., 51 (2013), pp. 423–444, https://doi.org/10.1137/120875065.

[21] A. Harten, S. Osher, B. Engquist, and S. R. Chakravarthy, *Some results on uniformly high-order accurate essentially nonoscillatory schemes*, Appl. Numer. Math., 2 (1986), pp. 347–377.

[22] A. K. Henrick, T. D. Aslam, and J. M. Powers, *Mapped weighted essentially non-oscillatory schemes: Achieving optimal order near critical points*, J. Comput. Phys., 207 (2005), pp. 542–567.

[23] G.-S. Jiang and D. Peng, *Weighted ENO schemes for Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 21 (2000), pp. 2126–2143, https://doi.org/10.1137/S106482759732455X.

[24] G.-S. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.

[25] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi, *Conformal curvature flows: From phase transitions to active vision*, Arch. Rational Mech. Anal., 134 (1996), pp. 275–301.

[26] P. Lions and P. Souganidis, *Convergence of MUSCL and filtered schemes for scalar conservation laws and Hamilton–Jacobi equations*, Numer. Math., 69 (1995), pp. 441–470.

[27] X. Liu, S. Osher, and T. Chan, *Weighted essentially non-oscillatory schemes*, J. Comput. Phys., 115 (1994), pp. 200–212.

[28] R. Malladi, J. A. Sethian, and B. C. Vemuri, *Shape modeling with front propagation: A level set approach*, IEEE Trans. Pattern Anal. Mach. Intell., 17 (1995), pp. 158–175, https://doi.org/10.1109/34.368173.

[29] A. M. Oberman and T. Salvador, *Filtered schemes for Hamilton-Jacobi equations: A simple construction of convergent accurate difference schemes*, J. Comput. Phys., 284 (2015), pp. 367–388.

[30] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer, New York, 2003.

[31] 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.

[32] S. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 28 (1991), pp. 907–922, https://doi.org/10.1137/0728049.

[33] G. Paolucci, *Adaptive Filtered Schemes for First Order Hamilton-Jacobi Equations and Applications*, Ph.D. thesis, Dipartimento di Matematica "Sapienza" Università di Roma, Rome, Italy, 2018.

[34] B. Scheuermann and B. Rosenhahn, *Analysis of numerical methods for level set based image segmentation*, in Advances in Visual Computing, G. Bebis, R. Boyle, B. Parvin, D. Koracin, Y. Kuno, J. Wang, R. Pajarola, P. Lindstrom, A. Hinkenjann, M. L. Encarnaçaõ, and C. T. Silva, eds., Springer, Berlin, Heidelberg, 2009, pp. 196–207.

[35] J. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, 2nd ed., Cambridge University Press, Cambridge, UK, 1999.

[36] J. A. Sethian, *Curvature and the evolution of fronts*, Comm. Math. Phys., 101 (1985), pp. 487–499.

[37] K. Zhang, L. Zhang, K.-M. Lam, and D. Zhang, *A level set approach to image segmentation with intensity inhomogeneity*, IEEE Trans. Cybern., 46 (2016), pp. 546–557.

[38] K. Zhang, L. Zhang, H. Song, and W. Zhou, *Active contours with selective local and global segmentation: A new formulation and level set method*, Image Vision Comput., 28 (2010), pp. 668–676.

[39] Y. Zhang and C.-W. Shu, *ENO and WENO schemes*, in Handbook of Numerical Methods for Hyperbolic Problems, Elsevier/North–Holland, Amsterdam, 2016, pp. 103–122.

[40] H. Zhu, J. Meng, J. Cai, and S. Lu, *Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation*, J. Vis. Commun. Image Represent., 34 (2016), pp. 12–27.