

Eikonal Equation Adaptation on Weighted Graphs: Fast Geometric Diffusion Process for Local and Non-local Image and Data Processing

Xavier Desquesnes · Abderrahim Elmoataz · Olivier Lézoray

Published online: 26 September 2012
© Springer Science+Business Media, LLC 2012

Abstract In this paper we propose an adaptation of the Eikonal equation on weighted graphs, using the framework of Partial difference Equations, and with the motivation of extending this equation's applications to any discrete data that can be represented by graphs. This adaptation leads to a finite difference equation defined on weighted graphs and a new efficient algorithm for multiple labels simultaneous propagation on graphs, based on such equation. We will show that such approach enables the resolution of many applications in image and high dimensional data processing using a unique framework.

Keywords Eikonal equation · Weighted graph · Non-local image processing · Active contour · PdE · Fast marching · High dimensional data

1 Introduction

The main goal of this paper is to propose an adaptation of the Eikonal equation on weighted graphs, using the framework of Partial difference Equations [17], with the motivation of extending this equation's applications, to any discrete data

that can be represented by graphs. This adaptation leads to a finite difference equation whose coefficients are data geometry dependent, and that leads to an efficient algorithm, as an extension of the Fast Marching algorithm, to propagate multiple fronts without restriction in the direction of their propagations. We will show that the combination of both coefficients and graph topology enables the resolution of many applications in image and high dimensional data processing.

Many applications involve data defined on topologically complex domains. These data can be defined on manifolds (e.g., a sphere) or irregularly shaped domains, defined on network-like structures (e.g., network communities), or defined as high dimensional point clouds such as collections of features vectors. Such unorganized data can be conveniently represented as graphs, where the vertices represent initial data and the edges represent interactions between them. Moreover, the use of a graph representation for usual images also enables to take into account local and non-local interactions and leads to very powerful tools for non-local image processing [8, 19].

Processing and analyzing such structured types of data is a major challenge for both the image and machine learning communities. Hence, it is very important to transfer many tools which were initially developed on usual Euclidean spaces and proven to be efficient for many problems dealing with usual image and signal domains, to graphs and networks.

Classical approaches for graph processing mainly come from graph theory and one can quote two main categories for these methods. Methods of the first category are based on the minimization of an energy with applications in semi-supervised segmentation. One can quote graph cuts [5], random walks [20] or recently the power watershed [14]. A second category groups techniques based on spectral graph theory [12]. They have been successfully used for image filter-

This work was supported under a doctoral grant of the Conseil Régional de Basse-Normandie and the Coeur et Cancer association.

X. Desquesnes (✉) · A. Elmoataz · O. Lézoray
GREYC, UMR CNRS 6972, Image Team, ENSICAEN,
Université de Caen Basse-Normandie, 6. Bvd Marechal Juin,
14050 Caen, France
e-mail: xavier.desquesnes@unicaen.fr

A. Elmoataz
e-mail: abder@unicaen.fr

O. Lézoray
e-mail: olivier.lezoray@unicaen.fr

ing [24], image segmentation [42], data clustering [30], or network communities extraction in complex networks [27], and so on.

There has been also recently much interest in transposing signal processing tools used in image and signal processing on graphs. One can quote the generalization of wavelets approach to graphs, with the work of Coifman et al. on diffusion wavelets [13], Jansen on multiscale methods [31], or recently Hammond et al. on spectral transform [23].

Similarly, there are recent works that aim to transpose Partial Differential Equations (PDEs) on graphs. These works exploit discrete calculus to perform such transcription [25]. Discrete Calculus has been used in recent years to produce a reformulation of continuous problems onto a graph in such a manner that the solution behaves analogously to the continuous formulation. See [22] and references therein for a complete overview on that subject with applications in image processing and machine learning.

To transpose PDEs on graphs, one approach consists in exploiting Partial difference Equations (PdEs) on graphs. Conceptually, PdEs are methods that mimic PDEs on the graphs general domain, by replacing differential operators by difference operator on graphs. Historically, it was first introduced in the seminal paper of Courant, Friedrichs and Lewy [15]. Then, the study of PdEs has appeared to be a subject on its own interest, dealing with existence and qualitative behavior problems [3, 32, 34]. Introduction of such methods for image processing started with the work of Chan et al. [10] who introduced the TV digital filter for image denoising, which is the discrete analogue of total variation on unweighted graphs. Zhou has also used TV on graphs for semi-supervised classification [51].

Following the line of research we developed in previous works, we base the contributions of this paper on difference operators on graphs [24]. The motivation is that these operators allows to simply adapt continuous formulations to graphs by replacing continuous operators by their discrete adaptation. In particular, it allows most techniques based on the p -Laplacian and gradients to be handled with such operators on graphs in a very straightforward, simple and similar manner [17, 47]. Such an approach enables an adaptation to graphs that is not necessarily consistent with the continuous formulation (see in [22]). This point is however not a problem for the paper and will be investigated in future works.

In previous works, using the PdEs formalism, we have introduced non-local difference operators on graphs, and used the framework of PdEs to transcribe PDEs on graphs [4]. In particular, in [17], we have introduced a non-local discrete regularization on graphs of arbitrary topologies as a framework for image and data filtering and clustering. With the same ideas, we have proposed PdE morphological processes on graphs that are a transcription of continuous morphological PDEs [45]. Recently, we have also adapted a time

-dependent version of the Eikonal equation with PdEs morphological processes on graphs [46, 47].

Eikonal Equation Background The Eikonal equation is a special case of the following general continuous static Hamilton-Jacobi equations:

$$\begin{cases} H(x, f, \nabla f) = 0, & x \in \Omega \subset \mathbb{R}^m, \\ f(x) = \psi(x), & x \in \Gamma \subset \Omega, \end{cases} \quad (1)$$

where ψ is a positive function defined on a domain Ω and $f(x)$ is the traveling time or distance from source Γ . Then, the Eikonal equation can be expressed by using the following Hamiltonian:

$$H(x, f, \nabla f) = \|\nabla f(x)\| - P(x), \quad (2)$$

where P is a given potential function. This equation can be linked to the level-set formulation for advancing fronts introduced by Sethian [40]

$$\frac{\partial \phi(x, t)}{\partial t} = \mathcal{F}(x) \|\nabla \phi(x, t)\|, \quad (3)$$

where ϕ is the level-set representation of Γ , and $\mathcal{F} = 1/P$. The relation between such formulation and the Eikonal equation stems from the following change of variable: $\phi(x, t) = t - f(x)$, under the condition that \mathcal{F} is positive on the whole domain Ω .

Solutions of static equation (2) are usually based on a discretization of the Hamiltonian where the approximations are performed by the Godunov methods [28] or with the Lax-Friedrich schemes [40]. Numerous numerical schemes have been proposed and investigated for solving the non-linear system described by (2). Among the existing ones, we can quote the following schemes. (i) An iterative scheme has been proposed by [37] based on fixed point methods that solve a quadratic equation. (ii) The fast sweeping methods [50] that use Gauss-Seidel type of iterations to update the distance function field. The key point of fast sweeping is to update the points in a certain order. (iii) Tsitsiklis [48] was the first to develop a Dijkstra like method and proposed an optimal algorithm for solving the Eikonal equation. Based on the same idea, Sethian [33, 40] proposed the fast marching methods.

Another approach to solve Eq. (2) is to consider a time dependent version of the equation:

$$\begin{cases} \frac{\partial f(x, t)}{\partial t} = -\|\nabla f(x)\| + P(x), & x \in \Omega \subset \mathbb{R}^m, \\ f(x, t) = \psi(x), & x \in \Gamma \subset \mathbb{R}^m, \\ f(x, 0) = \psi_0(x), & x \in \Omega. \end{cases} \quad (4)$$

At steady state, the solution of the system (4) satisfies the Eikonal equation (2). Recently, we proposed in a conference

paper [46] an adaptation of the time dependent formulation of the Eikonal equation over weighted graphs. Based on PdE, the analogue of (4) on a weighted graph $G = (V, E, w)$ is

$$\begin{cases} \frac{\partial f(u, t)}{\partial t} = -\|(\nabla_w^- f)(u)\|_p + P(u), & u \in V, \\ f(u, t) = \psi(u), & u \in V_0 \subset V, \\ f(u, 0) = \psi_0(u), & u \in V, \end{cases} \quad (5)$$

where V corresponds to the set of vertices of the graph and V_0 is the initial set of seed vertices. Operator ∇_w^- corresponds to the weighted internal morphological gradient on graphs (detailed in Sect. 2.2) and $\|\cdot\|_p$ denotes the \mathcal{L}_p -norm. One can see that formulation (5) needs numerous iterations due to finite propagation speed and CFL conditions to converge to the solution of the Eikonal equation.

Contributions This work generalizes and extends significantly our previous works on the Eikonal equation. First, we propose an adaptation of the stationary version of the Eikonal equation over arbitrary weighted graphs. Based on PdEs, our adaptation of (2) is given by this finite difference equation

$$\begin{cases} \mathcal{F}(u)\|(\nabla_w^- f)(u)\|_p = 1, & u \in V, \\ f(u) = 0, & u \in V_0. \end{cases} \quad (6)$$

Explicit solutions of this equation are given for particular values of $p \in \{1, 2, \infty\}$. An efficient algorithm to obtain such solutions, using the Fast Marching's updating scheme, is proposed, and proofs of existence and uniqueness are also provided. This formulation generalizes front propagation methods on weighted graphs and recovers well known schemes as Osher-Sethian discretization or Dijkstra shortest path, for specific graphs and values of p .

Such an adaptation of the Eikonal equation on graphs enables the transcriptions of efficient algorithms from the field of image processing to a huge variety of discrete data that can be represented by a weighted graph. Then, using this adaptation, we also propose a new fast algorithm for propagation and tracking of many concurrent fronts on a weighted graph, the complexity of which is independent of the number of these fronts. Such an algorithm leads to several applications on weighted graphs such as semi-supervised image segmentation or data clustering.

In these two previous contributions, we only considered the case where the front is evolving in the outward normal direction (with the speed \mathcal{F} defined non-negative everywhere), but it is also interesting to consider both inward and outward directions and both positive and negative speeds. In particular, in the case of semi-supervised clustering, such inward and outward evolutions enables to minimize or overcome errors due to wrong initialization. Finally, we generalize the previous algorithm to the case where the speeds of

the different fronts can be either positive or negative, which leads to a graph-based active contour model with many contours. This generalization provides a complete tool for multiple fronts propagation on arbitrary graphs, and offers a novel extension to the classical Fast Marching, when applied to regular grid graphs.

Paper Organization The rest of this paper is organized as follows. In Sect. 2, we provide definitions and notations used in this work. In Sect. 3, we present our new finite difference equation on weighted graphs with proofs for existence and uniqueness, and explicit solutions for values of $p \in \{1, 2, \infty\}$ are also given. Section 4 introduces two efficient algorithms for labels propagation, using the previous equation and an adaptation of the Fast Marching algorithm. Then, Sect. 5 presents several experiments which illustrate the behavior and efficiency of the proposed formulations and algorithms, as geodesic distance, semi-supervised image segmentation (with non-local configuration), active contours or data clustering. Finally, Sect. 6 concludes this paper.

2 Operators on Graphs

As the core structure of our approach, in this section we provide notations and basics on weighted graphs. We recall our formulations of differences, morphological differences and gradients on weighted graphs [4, 17, 45, 46]. The latter formulations constitute the basis of our proposed numerical scheme for solving the Eikonal equation on weighted graphs.

2.1 Notations

We consider the general situation where any discrete domain can be viewed as a weighted graph. Let $G = (V, E, w)$ be a weighted graph composed of two finite sets: vertices V and weighted edges $E \subseteq V \times V$. An edge $(u, v) \in E$ connects two adjacent (neighbor) vertices u and v . The neighborhood of a vertex u is noted $N(u) = \{v \in V \setminus \{u\} : (u, v) \in E\}$. The weight $w(u, v)$ of an edge (u, v) can be defined with a function $w : V \times V \rightarrow \mathbb{R}^+$ if $(u, v) \in E$, and $w(u, v) = 0$ otherwise. For the sake of simplicity, $w(u, v)$ will be denoted by w_{uv} . Graphs are assumed to be simple, connected and undirected implying that function w is symmetric.

Let $f : V \rightarrow \mathbb{R}$ be a real-valued function that assigns a real value $f(u)$ to each vertex $u \in V$. We denote by $\mathcal{H}(V)$ the Hilbert space of such functions and similarly by $\mathcal{H}(E)$, the Hilbert space of functions that assign a real value to each edge of E . These two spaces are endowed with the following inner products:

$$\langle f, h \rangle_{\mathcal{H}(V)} = \sum_{u \in V} f(u)h(u), \quad (7)$$

with $f, h \in \mathcal{H}(V)$, and

$$\langle F, H \rangle_{\mathcal{H}(E)} = \sum_{u \in V} \sum_{v \in V} F(u, v)H(u, v), \tag{8}$$

where $F, G \in \mathcal{H}(E)$.

Given a function $f \in \mathcal{H}(V)$, the integral of f is defined as

$$\int_V f = \sum_{u \in V} f(u) \tag{9}$$

and it's \mathcal{L}_p norm is given by

$$\|f\|_p = \left(\sum_{u \in V} |f(u)|^p \right)^{1/p}, \quad 1 \leq p < \infty, \tag{10}$$

$$\|f\|_\infty = \max_{u \in V} (|f(u)|), \quad p = \infty.$$

Let \mathcal{A} be a set of connected vertices with $\mathcal{A} \subset V$ such that for all $u \in \mathcal{A}$, there exists a vertex $v \in \mathcal{A}$ with $(u, v) \in E$. We denote by $\partial^+ \mathcal{A}$ and $\partial^- \mathcal{A}$: the *external* and *internal* boundary sets of \mathcal{A} , respectively

$$\begin{aligned} \partial^+ \mathcal{A} &= \{u \in \mathcal{A}^c : \exists v \in \mathcal{A} \text{ with } (u, v) \in E\} \quad \text{and} \\ \partial^- \mathcal{A} &= \{u \in \mathcal{A} : \exists v \in \mathcal{A}^c \text{ with } (u, v) \in E\} \end{aligned} \tag{11}$$

where $\mathcal{A}^c = V \setminus \mathcal{A}$ is the complement of \mathcal{A} .

2.2 Operators and Gradients on Weighted Graphs

The *weighted gradient operator* or *weighted difference operator* of a function $f \in \mathcal{H}(V)$, noted $\vec{\nabla}_w: \mathcal{H}(V) \rightarrow \mathcal{H}(E)$, respectively d_w , is defined on an edge $(u, v) \in E$ by

$$(\vec{\nabla}_w f)(u, v) \stackrel{\text{def.}}{=} (d_w f)(u, v) \stackrel{\text{def.}}{=} \sqrt{w(u, v)}(f(v) - f(u)). \tag{12}$$

The *adjoint of the weighted gradient operator*, noted $\vec{\nabla}_w^* f: \mathcal{H}(E) \rightarrow \mathcal{H}(V)$, is defined by:

$$\langle \vec{\nabla}_w f, H \rangle \stackrel{\text{def.}}{=} \langle f, \vec{\nabla}_w^* H \rangle, \tag{13}$$

with $f \in \mathcal{H}(V)$ and $H \in \mathcal{H}(E)$, and can be expressed as

$$(\vec{\nabla}_w^* H)(u) \stackrel{\text{def.}}{=} \sum_{v \sim u} \sqrt{w(u, v)}(H(v, u) - H(u, v)), \tag{14}$$

where $v \sim u$ means that v is adjacent to u .

This adjoint is linear and measures the flow of a function in $\mathcal{H}(E)$ at each vertex of the graph. Similarly to the continuous case, the *divergence* of a function $F \in \mathcal{H}(E)$ is defined by $\text{div}_w F = -\vec{\nabla}_w^* F$.

These two definitions of the *weighted gradient operator* and it's adjoint allow to define a family of first and second order operators on graphs, as the *p-Laplace operator*. But in this paper we only focus on the first order *weighted gradient operator*.

Based on the *weighted gradient operator* definition, two *weighted directional gradient operators* are defined. The weighted directional *external* and *internal* gradient operators are defined as $\vec{\nabla}_w^\pm: \mathcal{H}(V) \rightarrow \mathcal{H}(E)$, with

$$\begin{aligned} (\vec{\nabla}_w^+ f)(u, v) &\stackrel{\text{def.}}{=} \sqrt{w(u, v)}(f(v) - f(u))^+, \\ (\vec{\nabla}_w^- f)(u, v) &\stackrel{\text{def.}}{=} \sqrt{w(u, v)}(f(v) - f(u))-, \end{aligned} \tag{15}$$

with the following notations:

$$(x)^+ = \max(x, 0) \quad \text{and} \quad (x)^- = -\min(x, 0).$$

The *weighted gradient* of a function $f \in \mathcal{H}(V)$ at vertex u is defined as the vector of all *weighted gradient* with respect to the set of edges $(u, v) \in E$

$$(\nabla_w f)(u) \stackrel{\text{def.}}{=} (\vec{\nabla}_w f(u, v))_{v \in V}. \tag{16}$$

In the sequel, *weighted gradient* will refer to this gradient defined on vertices. Similarly, the *weighted morphological internal and external gradients* at a vertex u are expressed as

$$(\nabla_w^+ f)(u) = (\vec{\nabla}_w^+ f)(u, v)_{v \in V} \quad \text{and} \tag{17}$$

$$(\nabla_w^- f)(u) = (\vec{\nabla}_w^- f)(u, v)_{v \in V}.$$

The corresponding \mathcal{L}_p -norms of gradients (17) and (16) for a vertex u are

$$\begin{aligned} \|(\nabla_w^+ f)(u)\|_p &= \left[\sum_{v \sim u} w_{uv}^{p/2} |(Df(u))^+|^p \right]^{1/p}, \\ \|(\nabla_w^- f)(u)\|_p &= \left[\sum_{v \sim u} w_{uv}^{p/2} |(Df(u))^-|^p \right]^{1/p} \quad \text{and} \tag{18} \\ \|(\nabla_w f)(u)\|_p &= \left[\sum_{v \sim u} w_{uv}^{p/2} |Df(u)|^p \right]^{1/p} \end{aligned}$$

with $0 < p < \infty$ and where $Df(u) = (f(v) - f(u))$. The relation between these directional gradients norms was given in [46] as

$$\|(\nabla_w f)(u)\|_p^p = \|(\nabla_w^+ f)(u)\|_p^p + \|(\nabla_w^- f)(u)\|_p^p. \tag{19}$$

For the \mathcal{L}_∞ -norm, we have

$$\begin{aligned} \|(\nabla_w^+ f)(u)\|_\infty &= \max_{v \sim u} (\sqrt{w_{uv}} |(Df(u))^+|), \\ \|(\nabla_w^- f)(u)\|_\infty &= \max_{v \sim u} (\sqrt{w_{uv}} |(Df(u))^-|) \quad \text{and} \tag{20} \\ \|(\nabla_w f)(u)\|_\infty &= \max_{v \sim u} (\sqrt{w_{uv}} |Df(u)|), \end{aligned}$$

with the following property:

$$\|(\nabla_w f)(u)\|_\infty = \max(\|(\nabla_w^+ f)(u)\|_\infty, \|(\nabla_w^- f)(u)\|_\infty).$$

Properties of these gradients can be found in [45, 46]. One can note that the general definitions presented in this section are defined on graphs of arbitrary topology. They can be used to process any discrete regular or irregular data sets that can be represented by a weighted graph. Moreover, local and non-local settings are directly handled in these definitions and both are expressed by the graph topology in terms of neighborhood connectivity [17].

2.3 Morphological Evolution Equations

Time dependent Hamilton-Jacobi equation formulated in (4) is linked with mathematical morphology processes and can be viewed as morphological evolution equations. We have shown in [44, 45] that morphological gradients constitute numerical schemes for solving time dependent morphological dilation and erosion processes over graphs and therefore solve the time dependent Eikonal equation.

Continuous scale morphology (see for instance in [7, 38] and references therein) defines flat dilations $\delta : \mathbb{R}^m \rightarrow \mathbb{R}^m$ and erosions $\varepsilon : \mathbb{R}^m \rightarrow \mathbb{R}^m$ of a function $f^0 : \mathbb{R}^m \rightarrow \mathbb{R}$ by structuring sets $B = \{x : \|x\|_p \leq 1\}$, with the general PDEs

$$\partial_t f = +\|\nabla f\|_p \quad \text{and} \quad \partial_t f = -\|\nabla f\|_p \tag{21}$$

where f is a modified version of f^0 and with the initial condition $\partial_{t=0} f = f^0$. With different values of p , one obtains different structuring elements: a rhombus for $p = \infty$, a disc with $p = 2$ and a square with $p = 1$.

We have proposed in [45] a graph-based versions of (21) by using gradients defined in (17). Given a weighted graph $G = (V, E, w)$ and a function $f \in \mathcal{H}(V)$, the analogues of (21) on G are

$$\begin{aligned} \partial_t f(u) &= +\|(\nabla_w^+ f)(u)\|_p \quad \text{and} \\ \partial_t f(u) &= -\|(\nabla_w^- f)(u)\|_p. \end{aligned} \tag{22}$$

Intuitively, given a set of vertices $\mathcal{A} \subset V$ and using external and internal graph boundaries (11), equation of dilation over \mathcal{A} can be interpreted as a growth process that adds vertices from $\partial^+ \mathcal{A}$ to \mathcal{A} . By duality, erosion over \mathcal{A} can be interpreted as a contraction process that removes vertices from $\partial^- \mathcal{A}$ to \mathcal{A} .

As mentioned in the introduction, one approach to solve the Eikonal equation (1) is to consider a time dependent version (4) of this equation. The time dependent equation can be viewed as an erosion process regarding the minus sign and a null potential function P . Then, using the corresponding internal gradient (∇_w^-) involved in discrete PdEs based erosion process, one can directly obtain the time-dependent

adaptation of the eikonal equation on graphs. Given a graph $G = (V, E, w)$ and a function $f \in \mathcal{H}(V)$, we have

$$\begin{cases} \frac{\partial f(u, t)}{\partial t} = -\|(\nabla_w^- f)(u)\|_p + P(u), & u \in V, \\ f(u, t) = \psi(u), & u \in V_0 \subset V, \\ f(u, 0) = \psi_0(u), & u \in V, \end{cases} \tag{23}$$

where V_0 corresponds to the set of initial seed vertices. This equation can be solved by steepest gradient descent method, using the following iterative numerical scheme for all $u \in V$, with $f^n(u) \approx f(u, n\Delta t)$:

$$f^{n+1}(u) = f^n(u) - \Delta t (\|(\nabla_w^- f^n)(u)\|_p - P(u)). \tag{24}$$

A complete and precise definition with numerical schemes for $p \in \{1, 2, \infty\}$ can be found in [46].

3 Adaptation of the Eikonal Equation over Weighted Graphs

As mentioned in the introduction, the time dependent version of the Eikonal equation on weighted graphs needs numerous iterations to converge to the solution and our approach, detailed in Sect. 2.3 is not adapted to large graphs. Therefore, we propose a new adaptation of the Eikonal equation and a new algorithm to solve such finite difference equation (6), that overcomes these limitations.

We consider the Eikonal equation that describes the evolution of a propagation front Γ ,

$$\begin{cases} \mathcal{F}(x) \|\nabla f(x)\| = 1, \\ f(x) = 0, \quad x \in \Gamma, \end{cases} \tag{25}$$

where $f(x)$ is the arrival time of the front at x and $\mathcal{F} \geq 0$. The associated level-set function ϕ is defined as

$$\frac{\partial \phi(x, t)}{\partial t} = \mathcal{F}(x) \|\nabla \phi(x, t)\|, \tag{26}$$

and we know that $\phi(x, t) = t - f(x)$, then

$$\frac{\partial(t - f(x))}{\partial t} = \mathcal{F}(x) \|\nabla(t - f(x))\| = \mathcal{F}(x) \|\nabla f(x)\| = 1. \tag{27}$$

Transposed on a weighted graph $G = (V, E, w)$ with functions $f \in \mathcal{H}(V)$ and $\mathcal{F} \in \mathcal{H}(V)$ and using morphological equations defined in (22), the level-set formulation (26) can be rewritten as

$$\frac{\partial \phi(u, t)}{\partial t} = \mathcal{F}(u) \|(\nabla_w \phi)(u, t)\|_p, \tag{28}$$

and can be expressed as a morphological process with the following sum of dilation and erosion

$$\frac{\partial \phi(u, t)}{\partial t} = [(\mathcal{F}(u))^+ \|(\nabla_w^+ \phi)(u, t)\|_p^p + (\mathcal{F}(u))^- \|(\nabla_w^- \phi)(u, t)\|_p^p]^{1/p}. \tag{29}$$

Only considering the case $\mathcal{F} \geq 0$, and with $\phi(u, t) = t - f(u)$, one has

$$\begin{aligned} \frac{\partial \phi(u, t)}{\partial t} &= \mathcal{F}(u) \|(\nabla_w^+ (t - f))(u)\|_p \\ &= \mathcal{F}(u) \|(\nabla_w^- f)(u)\|_p = 1. \end{aligned} \tag{30}$$

Finally, with $P = 1/\mathcal{F}$ we obtain a discrete adaptation of the Eikonal equation on weighted graph, which describes a morphological erosion process, and defined as

$$\begin{cases} \|(\nabla_w^- f)(u)\|_p = P(u), & \forall u \in V, \\ f(u) = 0, & \forall u \in V_0. \end{cases} \tag{31}$$

3.1 Existence and Uniqueness of the Solution

The proof of existence and uniqueness of Eq. (31) can be established considering the equation

$$\begin{cases} \|(\nabla_w^- f)(u)\|_p = P(u), & u \in A \subset V, \\ f(u) = 0, & u \in \partial A \end{cases} \tag{32}$$

which can be expressed as $S_u(u, f, f(v)_{v \sim u}) = 0$ with

$$\begin{cases} S_u(u, f, [f(v)]_{v \sim u}) = \|(\nabla_w^- f)(u)\|_p - P(u) = 0, \\ u \in A, \\ S_u(u, f, [f(v)]_{v \sim u}) = 0, & u \in \partial A. \end{cases} \tag{33}$$

It can be easily shown that this scheme satisfies the following properties for all f .

- H_1 . $\partial S_u[f]/\partial f(v) \leq 0, \forall u \neq v$
- H_2 . $S_u[f + M] = S_u[f], \forall M \in \mathbb{R} \forall u$
- H_3 . $S_u[\lambda f] = \lambda S_u[f] + (\lambda - 1)P(u), \forall \lambda \geq 0, \forall u$
- H_4 . $\lim_{f(u) \rightarrow +\infty} S_u(u, f, f(v)_{v \sim u}) = -P(u)$
- H_5 . $\lim_{f(u) \rightarrow -\infty} S_u(u, f, f(v)_{v \sim u}) = +\infty$

Uniqueness of the Solution Equation (31) has a unique solution.

Proof Let f and g be two distinct solutions of (31) such that

$$\max_A (f(u) - g(u)) > 0. \tag{34}$$

Then for a given $\lambda \geq 1$, we have

$$M_H = \max_A (f(u) - \lambda g(u)) > 0. \tag{35}$$

We denote u_0 the u for which $M_H = f(u_0) - \lambda g(u_0)$. Then we can deduce that, with $h(u) = M_H + \lambda g(u)$:

$$\begin{cases} f(u) \leq h(u) & \forall u \in A, \\ f(u_0) = h(u_0). \end{cases} \tag{36}$$

This implies that

$$\begin{aligned} 0 = S_{u_0}[f] &\geq S_{u_0}[h] \quad (\text{due to } H_1) \\ &= S_{u_0}[\lambda g] \quad (\text{due to } H_2) \\ &= \lambda S_{u_0}[g] + (\lambda - 1)P(u_0) \\ &> 0 \quad (\text{due to } H_3). \end{aligned} \tag{37}$$

There is a contradiction unless $f(u) = g(u)$. □

Existence of Solution The proof of existence for Eq. (31) solution can be easily shown. We know that equation $S_u(u, f, f(v)_{v \sim u})$ is continuous. Due to properties H_4 and H_5 , we can deduce that $\exists -\infty < f(u) < +\infty$ such that $S_u(u, f, f(v)_{v \sim u}) = 0$.

3.2 Numerical Schemes and Algorithms

From (31) and using norms defined in (18) and (20) with the property $\min(x, 0) = -\max(-x, 0)$, we obtain the following equations for the \mathcal{L}_p and \mathcal{L}_∞ norms.

– Case $p \in \{1, 2\}$:

$$\left(\sum_{v \sim u} w_{uv}^{p/2} \max(0, f(u) - f(v))^p \right)^{1/p} = P(u). \tag{38}$$

With a simple transformation of variables and some conventional notations, Eq. (38) can be rewritten as

$$\sum_{i=1}^n \left(\frac{(x - a_i)^+}{h_i} \right)^p = \mathcal{C}^p, \tag{39}$$

where $x = f(u)$, $h_i = \sqrt{1/w_{uv}}$, $n = \text{card}(N(u))$, $a = \{f(v_i) | v_i \in N(u) \text{ with } i = 1, \dots, n\}$, $\mathcal{C} = P(u)$.

– Case $p = \infty$:

$$\max_{v \sim u} (\sqrt{w_{uv}} \max(0, f(u) - f(v))) = P(u). \tag{40}$$

Using the same transformation of variables we obtain

$$\max_i \left(\frac{(x - a_i)^+}{h_i} \right) = \mathcal{C}. \tag{41}$$

Local solutions (i.e., solution for a vertex, assuming the others are held fixed) of (31) over a weighted graph are given by (39) and (41). Both equations are clearly independent of the graph formulation, and can be applied to weighted graphs of arbitrary topology.

Algorithm 1 \bar{x} computation (local solution)

```

Sort increasingly the  $a_i, i = 1, \dots, n$ 
 $a_{n+1} \leftarrow \infty$ 
 $m \leftarrow 1$ 
 $\hat{x} \leftarrow \infty$ 
while  $\hat{x}_m \geq a_{m+1}$  and  $m \leq n - 1$  do
     $\hat{x}_m \leftarrow$  solution of  $\sum_{i=1}^m [\frac{x-a_i}{h_i}]^p = \mathcal{C}^p$  with  $p = 1$  or  $2$ 
     $m \leftarrow m + 1$ 
end while
 $\bar{x} \leftarrow \hat{x}_m$ 
    
```

For the case $p = \{1, 2\}$, local solution \bar{x} at a particular vertex can be easily obtained with an iterative algorithm as described in Algorithm 1. The algorithm is based on the knowledge that there exists a k with $1 \leq k \leq n$ such that $a_k \leq \bar{x} \leq a_{k+1}$, and \bar{x} is the unique solution of the equation. Then, the algorithm consists in sorting increasingly the values a_i and computing temporary solutions \hat{x}_m with the following equations (42) and (43) until the condition $\hat{x}_m \leq a_{m+1}$ is satisfied.

For $p = 1$, the temporary local solution is given by:

$$\hat{x}_m = \frac{\sum_{i=1}^m \frac{a_i}{h_i} + \mathcal{C}}{\sum_{i=1}^m \frac{1}{h_i}} \tag{42}$$

For $p = 2$, one has:

$$\hat{x}_m = \frac{\sum_{i=1}^m \frac{a_i}{h_i^2} + \sqrt{\sum_{i=1}^m [\frac{\mathcal{C}^2}{h_i^2} - \sum_{j>i} \frac{(a_i - a_j)^2}{h_i^2 h_j^2}]}}{\sum_{i=1}^m \frac{1}{h_i^2}} \tag{43}$$

The numerical scheme for the \mathcal{L}_∞ norm is much simpler. Considering Eq. (41), the local solution \bar{x} for a particular node is directly given by

$$\bar{x} = \min_{i=1}^n (a_i + h_i \mathcal{C}), \tag{44}$$

where n corresponds to the number of neighbors. One can remark that this equation is equivalent to Dijkstra like local update equation.

Many solvers can be adapted to compute global solution (i.e., on the whole graph) as, for instance, the Fast Iterative Method introduced by Jeong and Whitaker [49]. In this paper, we prefer the Fast Marching method which has the advantage to be monotonic. On an arbitrary graph, the Fast Marching consists in an *active list* (A) of vertices for which the solution is already known and fixed, and in a *narrow band* (NB) of vertices which are not yet fixed and have at least one neighbor in the *active list*. Vertices which are neither *active* or in the *narrow band* are said *far away* (FA). The *narrow band* is built as a sorted heap, and at each iteration, the first vertex is removed from the *narrow band*, added to

the *active list*, and it's neighbors are updated if they are not yet fixed and added to the *narrow band* if they are *far away*. Each neighbors v are updated simultaneously by computing new local solutions \bar{x}_v from the new value of $f(u)$ (and using Algorithm 1) and updating $f(v) \leftarrow \min(f(v), \bar{x}_v)$ under the condition that the new local solution \bar{x}_v is inferior to previous local solution. This is iterated until the narrow band is empty. More details on the initial Fast Marching algorithm and a proof that the initial algorithm constructs a viable solution are given in [39].

Accuracy and Complexity of the Algorithm Let $G = (V, E, w)$ be a weighted graph. The costs $C_{u,p}$ to update a vertex u (i.e., compute a new value of $f(u)$), according to Algorithm 1 and Eqs. (41), (42) and (43), are given by

$$\begin{aligned} C_{u,1}(u) &= D_u(u)^2 && \text{case } p = 1, \\ C_{u,2}(u) &= D_u(u)^3 && \text{case } p = 2, \\ C_{u,\infty}(u) &= D_u(u) && \text{case } p = \infty, \end{aligned} \tag{45}$$

where the quantity $D_u(u)$ corresponds to the number of neighbors of u which are not *far away*.

In order to generalize the cost functions $C_{u,p}$, we can consider that the previous quantity $D_u(u)$ is constant and corresponds to the maximum degree of a vertex of V . Such constant is defined as $D = \max_{u \in V} (\text{card}(N(u)))$. Thus, in the sequel we will use the following constants to denote the update costs.

$$\begin{aligned} C_{u,1} &= D^2 \geq C_{u,1}(u) && \forall u \in V, \\ C_{u,2} &= D^3 \geq C_{u,2}(u) && \forall u \in V, \\ C_{u,\infty} &= D \geq C_{u,\infty}(u) && \forall u \in V. \end{aligned} \tag{46}$$

Based on the previous constants, the activation costs $C_{a,p}$ of a vertex (i.e., the cost to change the state of a vertex from *far away* to *active*, what implies to update every of it's *not yet activated* neighbors) can be defined as

$$C_{a,p} = DC_{u,p}, \quad p \in \{1, 2, \infty\}. \tag{47}$$

while we know that $N(u) \leq D \forall u \in V$. And we have

$$\begin{aligned} C_{a,1} &= D^3 && \text{case } p = 1, \\ C_{a,2} &= D^4 && \text{case } p = 2, \\ C_{a,\infty} &= D^2 && \text{case } p = \infty. \end{aligned} \tag{48}$$

Finally, the total cost of the algorithm can be summarized as the following complexity

$$\mathcal{O}(C_{a,p} N \log(N)) \tag{49}$$

where N is the number of vertices and the $\log(N)$ factor corresponds to the managing cost of the heapsort. In practice, processing such an algorithm on strongly connected graphs is very rare, and in the most cases we have $C_{a,p} \ll N$.

3.3 Grid Graph Example

In this section, we show that with an adapted graph topology and an appropriated weight function, the proposed formulation recovers the well known Osher-Sethian scheme that has been proposed in literature to solve the Eikonal equation. Let $G = (V, E, w)$ be a weighted graph. In the case where $p = 2$, we have the following scheme

$$\sqrt{\sum_{v \sim u} w_{uv} \max(0, f(u) - f(v))^2} = P(u). \tag{50}$$

If the graph represents a m -dimensional grid in \mathbb{R}^m , this numerical scheme recovers the Osher-Sethian discretization models on a m -dimensional grid.

Let u be a given vertex of V , that defines a vector of m -dimensional spatial such that $u = (i_1 h_1, \dots, i_m h_m)$ where the h_j represent the grid spacing and $i_j \in \mathbb{N}$. The neighborhood of u can then be defined as $N(u) = \{v | v = u \pm h_j e_j \text{ with } j \in 1, \dots, m\}$ where $e_j = (q_k)_{k=1, \dots, m}$ is a vector of \mathbb{R}^m such as $q_k = 1$ if $j = k$ and $q_k = 0$ otherwise. Finally, with the notations

$$\begin{aligned} D_j^+ f(u) &= (f(u + h_j e_j) - f(u)) / h_j, \\ D_j^- f(u) &= (f(u) - f(u - h_j e_j)) / h_j \end{aligned} \tag{51}$$

and with $w_{uv_j} = 1/h_j$ for all $v_j \in N(u)$, (43) can be rewritten as

$$\sqrt{\sum_{j=1}^m \max(0, D_j^- f(u))^2 + \min(0, D_j^+ f(u))^2} = P(u) \tag{52}$$

since $\min(0, a - b)^2 = \max(0, b - a)^2$. This Eq. (52) corresponds to the Osher-Sethian Hamiltonian discretization scheme on a m -dimensional grid, and can be solved by the initial Fast Marching algorithm [39].

4 A New Class of Fast Algorithms for Semi-supervised Graph Clustering

In Sect. 3, we have introduced a new finite difference equation (31) which is an adaptation of the Eikonal equation over weighted graph, and proposed an efficient algorithm based on the Fast Marching method to solve such an equation. In this section, we propose to extend the previous algorithm so as to define a new class of fast algorithms for diffusion processes, that enable the simultaneous propagation of many concurrent labels on a weighted graph. Efficient algorithms for concurrent label propagation on a weighted graph have a great interest for graph partitioning and have led to numerous applications in semi-supervised image segmentation and data clustering. We will first present the algorithm when the

speed is always non-negative ($\mathcal{F} : V \rightarrow [0, +\infty]$), then its extension to the case where the speed can be either positive or negative ($\mathcal{F} : V \rightarrow [-\infty, +\infty]$).

4.1 Label Propagation with Non-negative Speed

Given a graph $G = (V, E, w)$, let $L = \{l_1, \dots, l_n\}$ be a set of labels and $S^0 = S_1^0 \cup \dots \cup S_n^0$ be the set of vertices initially marked by a label, where S_i^0 is the set of vertices initially marked by l_i . We call S_i^0 the seed of l_i .

The objective of label propagation for graph clustering is to mark each vertex u of V with a label l_i , under the condition that u is closer to at least one vertex of S_i^0 than other vertices of S^0 (according to the topology of G , the weight function w and a speed function $\mathcal{F}_{l_i} : V \rightarrow \mathbb{R}, l_i \in L$), so as to obtain a partition of the graph in n consistent clusters. The propagation of each label l_i is driven by a front Γ_i , initialized as the boundary of S_i^0 , and the final partition S_i is given by the set of vertices reached by Γ_i until it is stopped by another front or the boundary of the domain.

In practice, the propagation is performed in the same time than the computation of the arrival-time (given by resolution of (31)): each time a vertex u is reached by a front, the label of the front is propagated to u . The proximity condition is respected while in the case of several front propagation, the front that arrives at a vertex is necessarily the front coming from the nearest source of that vertex (according to weight and speed functions). This is a consequence of the Fast Marching algorithm which *activates* vertices from the smallest to the greatest distance (or arrival-time). Indeed, a vertex u is considered reached by a front when it is *activated* by the algorithm, and the propagated label comes necessarily from it's neighbor v which is already *activated* and such that

$$s(u) = \frac{f(v)}{w_{uv}} = \min_{z \sim u} \left(\frac{f(z)}{w_{uz}} \right). \tag{53}$$

The weight $1/w$ is used to penalize neighbors which are close to a source but have very weak link with u . The whole algorithm for label propagation on weighted graphs is given in Algorithm 2.

Remark 1 In the case where several neighbors have the same contribution and where these neighbors have different labels, several labels could be affected to node u . But, as this situation is very rare and happens only where fronts collapse, we choose, in this work, to label the vertex with only one arbitrary label from the list. This choice is application dependent and does not act as a rule.

Remark 2 Due to the multitude of fronts, we have to define one speed per front and the speed of a front Γ_i at a vertex

Algorithm 2 Labels propagation with non-negative speed

0. List of variables

S^0 : the set of seed vertices.
 A : the set of *active* vertices.
 NB : the set of vertices in the *narrow band*
 FA : the set of vertices said as *far away*
 lab : the label indicator function

1. Initialization:

$lab(u) =$ Initial label of u .
 $f(u) = 0 \forall u \in S^0 ; f(u) = +\infty \forall u \in V \setminus S^0$
 $s(u) = +\infty \forall u \in V \setminus S^0$
 $A = S^0 ; NB = \{u \mid \exists v \in A \text{ and } v \in N(u)\}$
 $FA = V \setminus A \cup NB$

2. Process:

while $FA \neq \emptyset$ **do**

$u \leftarrow$ first element of NB
 remove u from NB and add u in A .

for all $v \in N(u) \cap \overline{A}$ **do**

compute local solution $t \leftarrow f(v)$

if $t < f(v)$ **then**

$f(v) = t$

if $v \in FA$ **then**

remove v from FA and add v in NB .

else

update position of v in NB

end if

if $f(u)/w_{uv} < s(v)$ **then**

$s(v) = f(u)/w_{uv}$

$lab(v) = lab(u)$

end if

end if

end for

end while

u is given by $\mathcal{F}_{l_i}(u)$. Consequently, each time a node u is *activated*, it's neighbors v such that $v \notin A$ are updated by:

$$\|(\nabla_w^- f)(v)\|_p = P_{l_i}(v), \tag{54}$$

where l_i is the label of u and $P_{l_i}(v) = 1/\mathcal{F}_{l_i}(v)$.

One can remark that the algorithm complexity does not depend on the number of labels (the complexity is the same than for the classical Fast Marching algorithm described in Sect. 3). Then, this algorithm opens the way to a new class of fast algorithms for semi-supervised graph clustering, which allows to cluster data or segment images in many partitions, without loss of efficiency.

As noticed in Sect. 3, the label propagation is performed with a speed \mathcal{F} which is always non-negative. In the following section, we propose an adaptive version of the algorithm that can consider a front evolution (and then label propagation) with positive or negative speed.

4.2 Label Propagation Without Restrictions on Speed Sign

The case where the front's speeds can be either positive or negative is very interesting because it enables the labels to be propagated when the speed is positive and removed when the speed is negative, This also enables the tracking of multiple fronts subjected to changing sign speeds.

In the continuous case, Carlini has proposed an approach for tracking an hypersurface with non always positive velocity, called Generalized Fast Marching Method (GFMM) [9].

Let Γ be a front evolving on a continuous domain noted Ω , an represented by a characteristic discontinuous function $\theta(x, t)$ defined as

$$\begin{cases} \theta_t(x, t) = \mathcal{F}(x, t) |\nabla(\theta(x, t))|, \\ \theta(x, 0) = 1_{\Omega_0} - 1_{\Omega_0^c}, \end{cases} \tag{55}$$

where $\mathcal{F}(x, t)$ is a changing-sign speed and $1_{\Omega_0} - 1_{\Omega_0^c}$ is equal to 1 on Ω_0 and -1 on its complementary. The position of the front Γ is then given by the discontinuities of function θ and the time evolution is obtained by solving the stationary Eikonal equation (25).

Then, the main idea of the Generalized Fast Marching Method is to split the front in two fronts: $\Gamma_+^t = \partial\{x|\theta(x, t) > 0\}$ and $\Gamma_-^t = \partial\{x|\theta(x, t) < 0\}$ and perform two Fast Marching along two narrow bands ($NB_+^t = \Gamma_+^t \cap \{x, \mathcal{F}(x, t) < 0\}$ and $NB_-^t = \Gamma_-^t \cap \{x, \mathcal{F}(x, t) > 0\}$), with a modified version of (55). Interested readers can refer to [9] for a detailed presentation.

Then, we propose an extension of such an approach for front tracking on weighted graphs, along with a generalization of the previous algorithm (Algorithm 2) with positive and negative speeds. This can be done by representing each front Γ_i by two fronts Γ_i^+ and Γ_i^- that describe the part of front Γ_i subjected to a positive speed and respectively negative speed. Then, the propagation of a label l_i is performed through two evolving fronts: Γ_i^+ which describes the growth of the set S_i (where the speed \mathcal{F}_{l_i} is positive) and Γ_i^- which describes the decay of the set S_i (where the speed \mathcal{F}_{l_i} is negative). We have $\Gamma_i = \Gamma_i^+ \cup \Gamma_i^-$.

On the first hand, the front Γ_i^+ is initialized by the set $\{u|u \in \partial^- S_i^0 \text{ and } \mathcal{F}_{l_i}(u) > 0\}$. In other words, this corresponds to the set of vertices of S_i^0 which lies in the inner narrow band of S_i^0 and where the speed \mathcal{F}_{l_i} of the front is positive. Neighbors of the front are added to the *narrow band* under an additional condition: as the speed of the front has to be positive at their position. This additional condition can be easily justified since the front can't grow where its speed is negative.

On the other hand, the front Γ_i^- is initialized by the set $\{u|u \in \partial^+ S_i^0 \text{ and } \mathcal{F}_{l_i}(u) < 0\}$. Neighbors are added to the *narrow band* under the condition than the speed \mathcal{F}_{l_i} is negative. Because this front describes the decay of S_i , each time

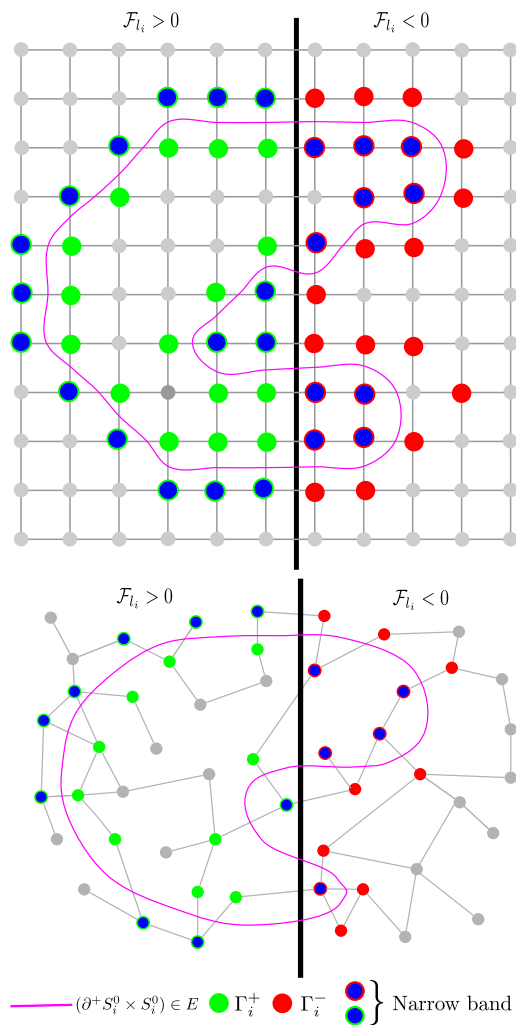


Fig. 1 Label propagation with changing-sign speed. The *first graph* is a 4-adjacency grid graph, and the *second* a non-Cartesian graph. Vertices inside the *purple line* represent S_i^0 . The speed is positive on the *left side* and negative on the *right side*. *Green points* and *red points* represent vertices in initial configuration of positive front Γ_i^+ , respectively vertices in initial configuration of negative front Γ_i^- . *Blue points* represent vertices in the narrow band from the front Γ_i^+ (surrounded in *green*) and from the front Γ_i^- (surrounded in *red*) (Color figure online)

a vertex is reached by Γ_i^- , the vertex is *unmarked* instead of being marked by l_i .

Finally, in order to keep the distance (or arrival time) always positive, the signed version of the speed is replaced by its modulus and the potential is given by $P_{l_i}(u) = 1/|\mathcal{F}_i(u)|$.

Such an approach of label propagation with changing-sign speed for two fronts is illustrated in Figs. 1 and 2. The entire process is summarized in Algorithm 3.

Remark 3 In the case where $p = \infty$, and according to Eq. (44), the arrival-time of a vertex is computed with the contribution of exactly one edge. Geometrically, this means that the fronts propagation follows the edges (one can see

Algorithm 3 Labels propagation with changing-sign speed

0. List of variables:

- S^0 : the set of seed vertices
- A : the set of *active* vertices.
- NB : the set of vertices in the *narrow band*
- FA : the set of vertices said as *far away*
- lab : the label indicator function
- $\partial^+ S_i^0$: outer boundary of the set of seeds S_i
- $\partial^- S_i^0$: inner boundary of the set of seeds S_i
- \mathcal{F}_i : speed function associated to label l_i .

1. Initialization:

- $A = \{u \mid u \in \partial^+ S_i^0 \text{ and } \mathcal{F}_i(u) < 0\}$
- $\cup \{u \mid u \in \partial^- S_i^0 \text{ and } \mathcal{F}_i(u) > 0\}$
- $NB = \{u \mid \exists v \in A \cap N(u) \text{ and } \mathcal{F}_i(v) \times \mathcal{F}_i(u) > 0\}$
- $FA = V \setminus (A \cup NB)$
- $f(u) = 0; lab(u) = l_i \quad \forall u \in A$
- $f(u) = \min(1/w_{uv}), v \in A \cap N(u)$
- $lab(u) = l_i; s(u) = f(u) \quad \forall u \in NB$
- $f(u) = +\infty; lab(u) = 0; s(u) = +\infty \quad \forall u \in FA$

2. Process:

```

while FA ≠ ∅ do
    u ← first element of NB
    remove u from NB and add u in A.
    for all v ∈ N(u) ∩ Ā and F_i(v) × F_i(u) > 0 do
        compute local solution t ← f(v)
        if t < f(v) then
            f(v) = t
        if v ∈ FA then
            remove v from FA and add v in NB.
        else
            update position of v in NB
    end if
    if f(u)/w_uv < s(v) then
        s(v) = f(u)/w_uv
        lab(v) = lab(u)
    end if
end if
end for
end while
    
```

edges as pipes in which the fronts evolve), and the front that reaches a particular vertex comes from exactly one of its neighbors. The label assignment is then automatically deduced.

On the contrary, in the case where $p = \{1, 2\}$ and according to Eqs. (42), (43), the arrival-time of a vertex is computed with the contribution of several edges. Thus, only considering the graph structure, the front propagation is interpreted as coming from several edges. In practice, we consider that the front comes from the edge with the highest contribution, and labels are propagated in this way. If we consider a sufficiently higher-dimensional domain in which

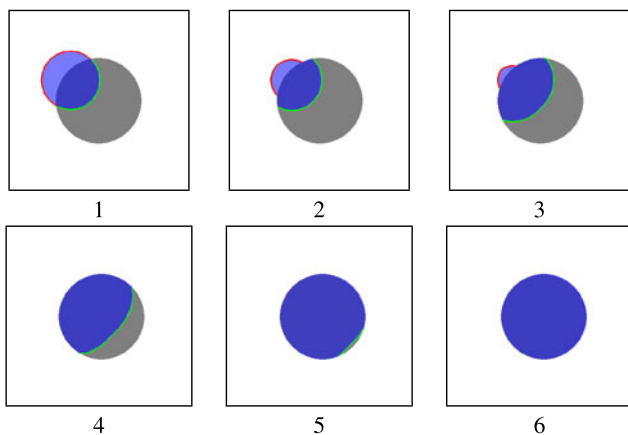


Fig. 2 Illustration of label propagation with changing-sign speed, with a toy example. The set S_i is represented in *blue*. The inner narrow band which represents the position of front Γ_i^+ is superimposed in *green*, and the outer narrow band which represents the front Γ_i^- is superimposed in *red*. The *six* images show different steps of the evolution of the set S_i and both two fronts, with the speed \mathcal{F}_i , which is positive on the *gray* background and negative on the *white* background (Color figure online)

the graph is embedded (as it is the case for images, meshes or manifolds), the front is coming between all the involved edges.

5 Experiments and Applications

5.1 Graph Construction

There exists several popular methods that transform a set of unorganized discrete data into a neighborhood graph. Considering a set of vertices V such that functions of $\mathcal{H}(V)$ represent the data, the construction of such a graph consists in modeling the neighborhood relationships between the data through the definition of a set of edges E , using a pairwise distance measure $\mu : V \times V \rightarrow \mathbb{R}^+$. In the general case (where data are unorganized) one can quote:

- ε -neighborhood graphs where two data $v_i, v_j \in V$ are connected by an edge of E if $\mu(v_i, v_j) \leq \varepsilon$, with $\varepsilon > 0$.
- k -nearest neighbor graphs (k -NNG) where each vertex v_i is connected with its k -nearest neighbors according to μ . Such construction implies to build a directed graph, as the neighborhood relationship is not symmetric. Nevertheless, an undirected graph can be obtained while adding an edge between two nodes v_i and v_j if v_i is among the k -nearest neighbors of v_j or if v_j is among the k -nearest neighbors of v_i .

In the case of structured data (i.e., images or meshes):

- *Grid graphs* which are the natural structure to describe an image with a graph. Each pixel is connected by an edge to its adjacent pixels.

- *Region adjacency graphs (RAG)*, initially designed for images where vertices correspond to image regions and the set of edges is obtained by considering an adjacency distance. A RAG can be built for any structured data represented by a graph, where a region R_i is defined as a set of connected vertices such that $\bigcup R_i = V$ and $\bigcap R_i = \emptyset$. Two regions R_i, R_j are adjacent if $\exists v_i \in R_i$ and $v_j \in R_j \mid v_i \sim v_j$.

In both cases, weights are computed according to a measure of similarity $g : E \rightarrow \mathbb{R}^+$, which satisfies:

$$w(u, v) = \begin{cases} g(u, v) & \text{if } (u, v) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{56}$$

The similarity is usually based on a pairwise distance measure between data features, where each vertex $u \in V$ is represented by a feature vector $F_u \subset \mathbb{R}^m$. For a given edge $(u, v) \in E$ and a distance measure $\mu : V \times V \rightarrow \mathbb{R}^+$, we can have

$$\begin{aligned} g_0(u, v) &= 1, \\ g_1(u, v) &= (\mu(F_u, F_v) + \varepsilon)^{-1} \quad \text{with } \varepsilon > 0, \varepsilon \rightarrow 0, \\ g_2(u, v) &= \exp(-\mu(F_u, F_v)/\sigma^2) \quad \text{with } \sigma > 0, \end{aligned} \tag{57}$$

where σ depends on the variation of the function μ over the graph and controls the similarity scale. Several choices can be considered for the expression of the feature vectors, depending on the nature of the features to be used for the graph processing. In the context of image processing, one can quote the simplest grayscale or color feature vector F_u , or the patch feature vector $F_u^\tau = \bigcup_{v \in \mathcal{W}^\tau(u)} F_v$ (i.e., the set of values F_v where v is in a square window $\mathcal{W}^\tau(u)$ of size $(2\tau + 1) \times (2\tau + 1)$ centered at a vertex pixel u), in order to incorporate non-local features.

5.2 Weighted Geodesic Distances

In this section, we show the adaptivity of our framework and illustrate the behavior of proposed numerical schemes to compute weighted geodesic distances.

Figure 3 presents the application of the schemes formulated by (39) and (41). Results are obtained with different graph topologies and weight functions. They are all illustrated with color distance maps where iso-levels are superimposed in white.

The potential function P is constant and equal to 1 and the propagation is performed for a unique label which initial seed is located either at the top left corner of the original image or at the center. The original image is a 256×256 grayscale image. First rows show results obtained with a weighted 4-adjacency grid graph and different weight functions w_1, w_2 and w_3 . The first weight function ($w_1 = 1$) is a constant weight function. Using such weight function and

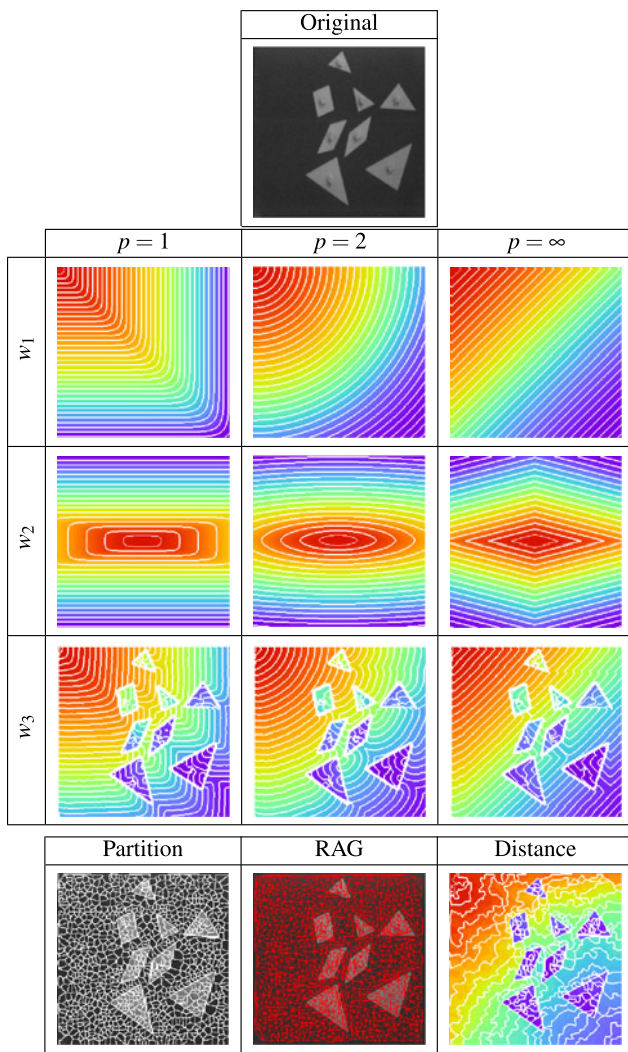


Fig. 3 Adaptive weighted distance computation with different p values, weight functions w and graph topologies G . Results represent color distance maps with iso-level sets. *At top*: original image. *First, second and third columns*: results with $p = 2, 1$ and ∞ . *First row*, the weight w_1 is constant. *Second row*, the weight w_2 is designed to penalize vertical diffusion. *Third row*, the weight w_3 holds original image informations. *First rows*, the graph is built on the whole image. *On the last*, the distance is computed on a Region Adjacency Graph (RAG) from a partition of the original image

in the case where $p = 2$ we obtain the same result as using the initial Fast Marching algorithm [39], what verifies that in such configuration our formulation recovers the Osher-Sethian scheme. Similarly, in the case where $p = \infty$, our algorithm is equivalent to a Dijkstra algorithm. The second weight function is designed to favor horizontal diffusion to vertical diffusion by penalizing the weights of vertical edges and given by

$$w_2(u, v) = \begin{cases} 1 & \text{if } v_x = u_x, \\ 0.25 & \text{if } v_y = u_y. \end{cases} \quad (58)$$

In that latter case, the seed is placed at the center of the image. The third weight function ($w_3 = \exp(-d^2/\sigma^2)$) is designed to hold the similarity between each connected pixel (based on pixel intensity). Shape information is naturally captured by the weights that stop the front evolution at the boundaries. Clearly, in this case, one can obtain a segmentation of the different shapes simply by thresholding the obtained distance map. We provide results for different values of $p = 1, p = 2$ and $p = \infty$ at respectively first, second and third column.

The last row illustrates distance computation on a Region Adjacency Graph (RAG), built from the original image. The first column shows the partition. Second one shows the associated graph where each node represents a region of the RAG and two nodes are linked if their regions are adjacent. Finally, the third column presents the distance map obtained from the top-left region. In that case, the weight function holds the similarity between two adjacent regions in the sense of mean intensity of each region, and the distance was computed with $p = \infty$.

Figure 4 presents illustrations of geodesic distances computed on an irregular 3d mesh and an irregular 2d graph with constant speed/potential. The two graphs are said irregular, because each vertex has a variable number of edges. In these cases, our results does not necessarily coincide with solutions of the Eikonal equation via Fast Marching algorithms. Indeed, the finite difference equation (31) used to compute these geodesics is not the Eikonal equation (but is adapted from) and only consider values of functions defined on graph vertices. There exists some algorithms, using geometric informations and local solvers designed for these particular cases, which better approximate the solution of the Eikonal equation on triangulated domains, as the Fast Marching algorithm for triangulated domains proposed by Kimmel [26], or Fast Sweeping on triangulated meshes by Qian [35].

In the following section, we will present the accuracy of the proposed fast algorithm to compute graph partitions.

5.3 Graph Partitioning

In this section, we present the accuracy of the proposed algorithm (Algorithm 2) for graph partitioning using a distance function (i.e. a metric), with some examples of data simplification. The use of a distance function for graph partitioning is not new, and we recommend interested readers to refer to [1, 2, 18] a references therein for a more complete review on this topic.

First, we recall a definition of metric-based graph partition. Let $G = (V, E, w)$ be a weighted graph and $C_G(u, v)$ be the set of paths connecting two vertices $u, v \in V$. A path $c(u, v)$ is a sequence of vertices (u_1, \dots, u_j) such as $u = u_1$ and $v = u_j$ with $(u_i, u_{i+1}) \in E$ and $i = 1, \dots, j - 1$.

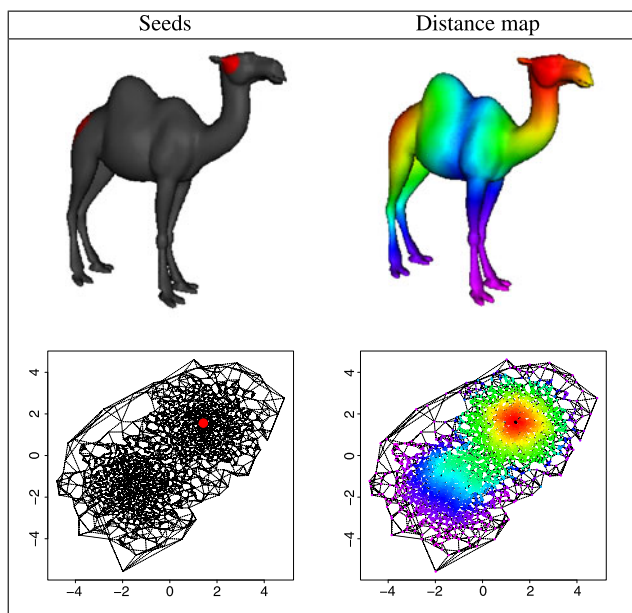


Fig. 4 First row: geodesic distance on a 3d mesh from two distinct seeds with, at left the seeds, at right the rainbow distance map where red means near and blue means far. Second row: geodesic distance on an irregular graph in \mathbb{R}^2 with one seed

Let $\rho : E \rightarrow \mathbb{R}$ be a metric defined as

$$\rho(u, v) = \min_{\rho \in C_G(u, v)} \sum_{i=1}^{j-1} (f(u_{i+1}) - f(u_i)). \tag{59}$$

Given a set of K labels $L = \{l_i\}_{i=1, \dots, K}$ and a set of K seeds $S^0 = \{s_i^0\}_{i=1, \dots, K} \subseteq V$ (where s_i^0 is the seed of l_i), the energy $\rho_S : V \rightarrow \mathbb{R}^m$ induced by the metric ρ for all seed of S^0 is

$$\rho_S(u) = \min_{s_i \in S} \rho(s_i, u) \quad \forall u \in V. \tag{60}$$

The region R_i of a label l_i , is the set of vertices which are closer to s_i^0 than to any other seeds with respect to the metric ρ . R_i can be defined as

$$R_i = \{u \in V : \rho(s_i^0, u) \leq \rho(s_j^0, u), \forall j = 1, \dots, K\}. \tag{61}$$

A partition of G , for a given set of labels (with seeds S^0) and a metric ρ , is finally the set $R(L, \rho) = \{R_i, \forall l_i \in L\}$. For a graph G , to find its partition corresponds to seek a minimal cost path over G , and can be easily computed with a simple Dijkstra like algorithm.

This general formulation of graph partitioning according to a metric can be easily linked with our definition of label propagation for graph clustering (see Sect. 4.1). Indeed, such a partition can be processed using the new finite difference equation (31) and the new class of algorithms proposed in Algorithm 2 or Algorithm 3, since we know that the label propagated at each vertex $u \in V$ is the label which seed is

the nearest of u (see Sect. 4 for more details). Given a vertex $u \in V$ with a label l_i , we have the following relation

$$f(u) = \rho_S(u) = \rho(u, s_i^0) = \min_{s_j^0 \in S^0} (\rho(u, s_j^0)). \tag{62}$$

Moreover, graph partitioning is intrinsic to our algorithm, because of the label propagation according to the distance measure (or arrival time), which naturally cluster the graph. The resulting partition corresponds to a generalized Voronoï diagram. In the next paragraph, we present a methodology to compute graph partition using our equations and algorithms, we call supervertices.

From Super Pixels to Super Vertices Initially developed by Ren and Malik [36], superpixels are an efficient way to reduce image complexity by grouping pixels in a region map while preserving contours. With TurboPixels [29], Levinshstein et al. have proposed an implementation of superpixels in which an image partition is obtained by dilating a regular grid of seeds so as to adapt to local image structure, where the dilation is performed using a level-set approach. In this paper, we propose to extend such approach of image simplification to weighted graphs, using our fast algorithm for multilabel propagation on graphs. Because this adaptation groups vertices instead of pixels, we name it supervertices. As TurboPixels, our approach uses a set of labels whose seeds are placed on initial data, but seeds dilation becomes a label propagation which is controlled by our fast algorithm instead of iterative evolving equations. Figure 5 presents two examples of supervertices on 3D meshes. A review of some other mesh partitioning methods can be found in [41] and references therein. In the case of image processing, the process is very similar to superpixels: the simplification is performed from a regular grid of distinct labels whose seeds position can be perturbed in the direction of the descending image gradient to avoid placing seeds close to string boundaries. This is illustrated in Fig. 6 on two images from the Berkeley Database.¹ Such partition can be easily transformed in a Region Adjacency Graph (RAG), which can be used as a simplified version of the initial image that preserves texture information and strong boundaries. Then, it becomes very interesting to perform any graph-based algorithm on such a reduced version of the image. This is illustrated in Sect. 5.4.4 with active contour and semi-supervised image segmentation using RAG.

5.4 Semi-supervised Image Segmentation and Data Clustering

In this section, we present the behavior and efficiency of our algorithm for semi-supervised image segmentation and data

¹<http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/>.

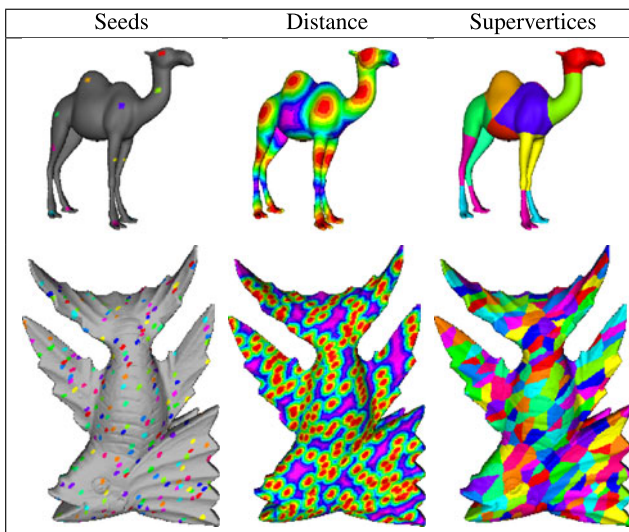


Fig. 5 Supervertices on meshes. Graphs used in this example are faces graphs, where each face of meshes is represented by a vertex and linked by an edge to any adjacent face. In both cases, the weight function is given by $f = 1$ and $P = 1$. *First column* presents initial meshes with superimposed random seeds. *Second column* shows the distance maps computed where red means near a seed and purple means far from a seed. Finally, *last column* shows the resulting partition in super vertices

clustering, using graph representation of these data. First we give qualitative and quantitative comparison with a recent and efficient graph-based approach for semi-supervised image segmentation. Then, we illustrate the algorithm behavior with local or non-local configurations and with non-negative speeds or either positive or negative speeds, through several examples involving different kinds of data.

In the case of image semi-supervised segmentation, graph-based approaches have become very popular in recent years. Many graph-based algorithms for image segmentation have been proposed, such as graph cuts [5], random walk [20], shortest-path, watershed or frameworks that unify some of the previous methods (as powerwatershed) [14, 43]. Recently, these algorithms were all placed into a common framework [14] that allows them to be seen as special cases of a single general semi-supervised algorithms, and leads to the following general formulation: Given a graph $G = (V, E, w)$, the general algorithm consists in finding the function f which minimizes the following energy function:

$$E_{p,q}(f) = \sum_{u \sim v \in E} w_{uv}^p |f(u) - f(v)|^q \tag{63}$$

where f represents the target configuration. Using such a formulation in the case of two classes A and B such that $V_A \cup V_B = V$ and $V_A \cap V_B = \emptyset$, the graph clustering can be performed as follow:

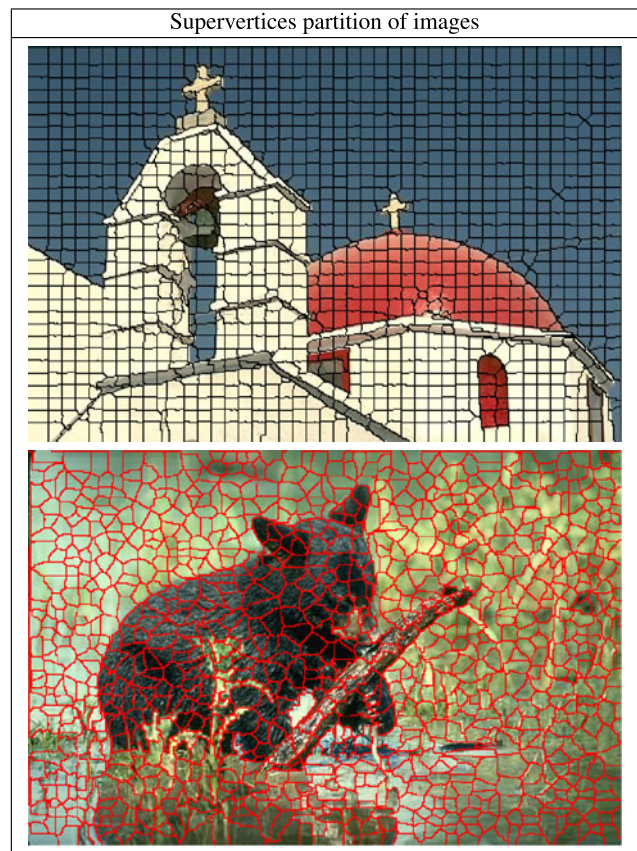


Fig. 6 Image partition with supervertices. *Both images* were partitioned using a regular grid of seeds (perturbed according to the image gradient). See text for details

1. The first step consists in finding an optimum f for (63), which is obtained by solving the equation

$$\begin{cases} f(A) = 1, \\ f(B) = 0, \\ f = \arg \min_f E_{p,q}(f). \end{cases} \tag{64}$$

Such formulation can be interpreted using PdEs as $\arg \min \sum \|(\nabla_w f)(u)\|$ with w depending on p and q , and A and B represents the internal Dirichlet boundary condition.

2. Finally, the segmentation is performed via a threshold and the labeling function $L : V \rightarrow \{1, 0\}$ is given by

$$L(u) = \begin{cases} 0 & \text{if } f(u) < 1/2, \\ 1 & \text{if } f(u) \geq 1/2. \end{cases} \tag{65}$$

In the case of more than two classes, the method needs to find as many optimum f_i as distinct labels, where the shape is given by the set of vertices of the label and the background by the set of vertices of all other labels. Then, the labeling











Seeds	Power watershed	Eikonal $p = 1$	Eikonal $p = 2$	Eikonal $p = \infty$
				
mean computation times:	0.281 s	0.540 s	0.528 s	0.504 s
				
mean computation times:	0.610 s	0.907 s	0.895 s	0.863 s

Fig. 7 Qualitative comparison between power watershed and the proposed approach, using two images from the grabcut database. See text for more details

function is given by

$$L(u) = \arg \max_i f_i(u). \quad (66)$$

Another approach [2, 18] to perform a graph clustering consists in computing a graph partition from the set of user's seeds and a metric, as it is described in the previous section. In this paper, we consider this approach which can be easily performed using Algorithms 2 and 3.

5.4.1 Comparison with Power Watershed

In this paragraph, we propose qualitative and quantitative comparisons between our method and the recent power watershed [14] which is one of the most efficient methods proposed in the previously cited framework. Results for the power watershed are obtained using the source code from Couprie's website.²

Figure 7 presents a qualitative comparison between the two algorithms for semi-supervised image segmentation with two labels (shape and background). The comparison is performed on images from the Microsoft Grabcut database.³ The initial set of seeds is replaced by an eroded version in order to penalize both algorithms (as all propagation algorithms are very efficient when desired boundaries and seeds are very close). Results are presented for values of $p \in \{1, 2, \infty\}$ and the power watershed (given by (63) with $p = 2, q = 2$). The mean computation time is given for each method (although it is quite difficult to compare runtime from two different implementations) and one can remark

that the runtime difference between each p -norm is negligible, while the case $p = 2$ outperform the three other methods ($p = 1, \infty$ and power watershed), with smoother results.

On the contrary to others methods, the proposed algorithm complexity and computation time does not depend of the number of labels (neither seeds). Indeed, whatever the number of labels, each vertex is *activated* exactly once by the algorithm (and we recall that the arrival time and the label of each vertex are fixed when the vertex is *activated*, see Sect. 4.1). This is illustrated on Figs. 8 and 9, which presents the runtime variation of both methods, in function of an increasing number of labels. The labels propagation is performed on a cytological slide photography (size 1280×2960) on which we consider a label for the background and a label per cell. For each experiment, an increasing number of label's seeds are placed on the image, and the non-seeded cells are considered as background. Figures 8 and 9 presents the whole set of labels and the slide segmentation with all labels for both two methods. One can remark that where the power watershed runtime grows as the number of labels, the proposed method runtime is relatively constant (the little increase is implementation dependent and is not a consequence of the algorithm).

5.4.2 Non-local Image Segmentation

Our approach using graphs has the advantage to naturally enable local and non-local configurations in the same formulation. In this paragraph, we show the benefits of non-local schemes as compared to local ones for semi-supervised image segmentation, especially noisy images or images that contain fine and repetitive structures. In order to hold non-local configurations, the graph is construct as a k -grid graph, (i.e. a pixel u is linked to every pixels in a $k \times k$ window centered on u), and each pixel is associated with a patch

²<http://www.esiee.fr/~couprie/>.

³<http://research.microsoft.com/en-us/um/cambridge/projects/visionimagevideoediting/segmentation/grabcut.htm>.

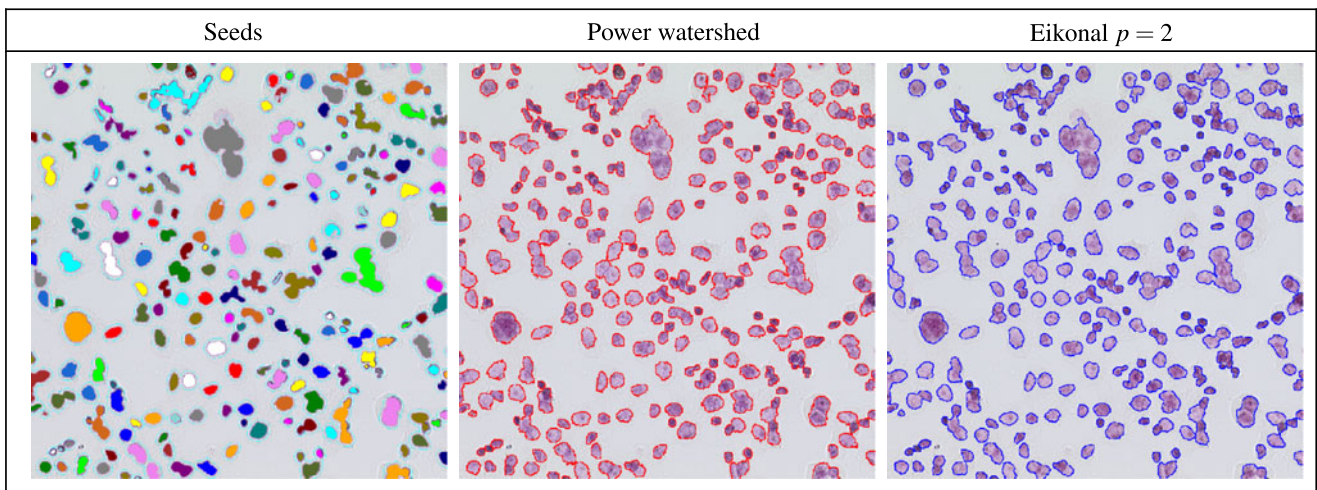


Fig. 8 Runtime variation in function of the number of labels. Variation is given for the power watershed (in red) and the proposed approach (in blue). The sets of seeds for the maximum number of labels, and

segmentation results for both methods are illustrated. See text for a comment (Color figure online)

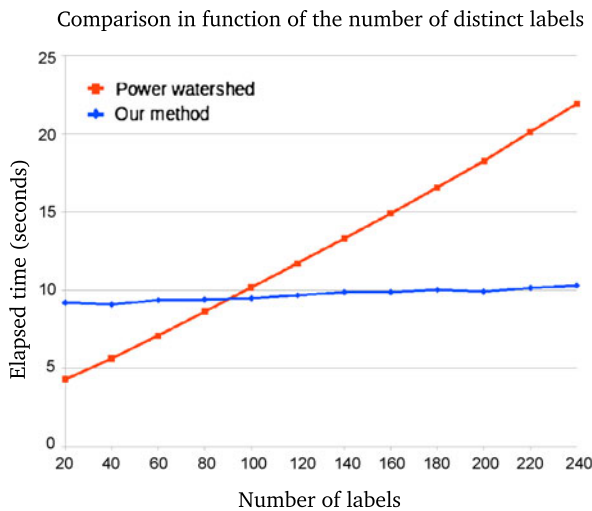


Fig. 9 Runtime variation in function of the number of labels. Variation is given for the power watershed (in red) and the proposed approach (in blue). The graph illustrates the runtime variation while the number of labels increase from 20 to 240. See text for a comment (Color figure online)

feature vector (where each vector can be seen as a patch of texture). Figure 10 presents several examples of textured image segmentation. First column presents initial image and seeds. Second column shows local results obtained with a 4-adjacency grid graph using color feature vectors. Finally, the last column shows non-local results with a 11×11 window and patches of 3×3 . All results were computed with a potential function $P = 1$ and $p = 2$. These results demonstrate the benefits of non-local configurations especially for textured images, where classical methods fail to found correctly the desired object. In non-local configuration cases,

the graph weights better capture fine and repetitive information contained in the image.

5.4.3 Active Contour

In this paragraph, we illustrate the behavior of our algorithm to perform active contour on graphs using an adaptation of the Chan and Vese model [11]. Although many recent approaches outperform this model [6, 16, 21], our choice lies in that we are not interested in present a new active contour model on graph, but only illustrate the adaptivity of our finite difference equation and algorithms. Given a front Γ represented by the level-set function ϕ , the Chan and Vese active contours model (without regularization term) can be summarized by the following PDE

$$\frac{\partial \phi(x, t)}{\partial t} = [(F_x - \mu_1(t))^2 - (F_x - \mu_2(t))^2] \|\nabla \phi(x, t)\|, \tag{67}$$

where F_x is the feature vector of x , $\mu_1(t)$ and $\mu_2(t)$ are the mean outside the front Γ at time t , respectively the mean inside. With $\mathcal{F}(x, t) = (F_x - \mu_1(t))^2 - (F_x - \mu_2(t))^2$ such an Eq. (67) is equivalent to the level-set function defined in (26), which is the continuous version of our finite difference equation. Then, transposed on weighted graphs, the active contours model for a front Γ_i can be written as

$$\|(\nabla_w^- f)(u)\|_p = P_i(u, t), \tag{68}$$

with $P_i(u, t) = 1/\mathcal{F}_i(u, t)$ and $\mathcal{F}_i(u, t) = (F_u - \mu_1(t, l_i))^2 - (F_u - \mu_2(t, l_i))^2$. Finally, the front evolution is performed using Algorithm 3 with the advantages given by our formulation as the speed of the Fast Marching and

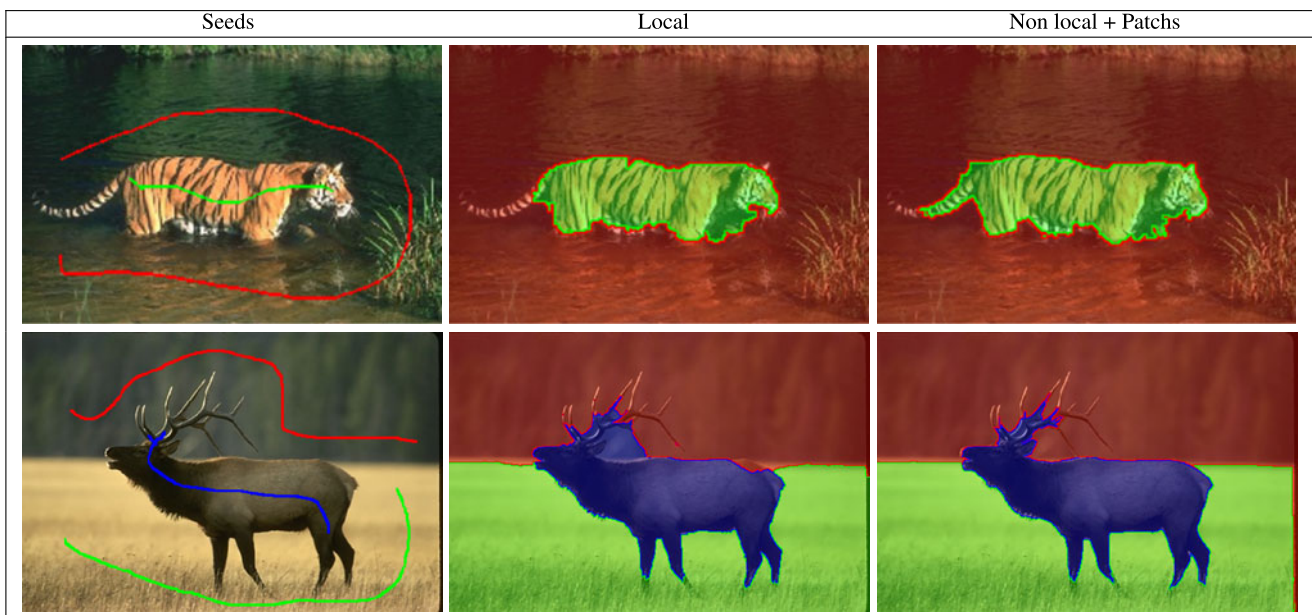


Fig. 10 Semi-supervised image segmentation using the proposed algorithm. For each *image*, results are provided for two different graph topologies and weight functions. The *first column* presents images to segment with superimposed initial labels (respectively 2 and 3). The *second one* shows results for 4-adjacency grid graphs where each pixel

is characterized by its color feature vector. Finally, *third column* shows results for a larger neighborhood (each pixel u is linked with any pixel in a 11×11 window centered on u) and pixels are characterized by patches of 3×3

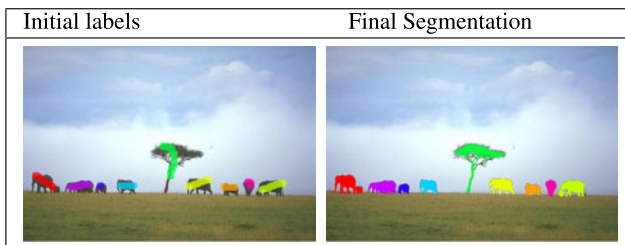


Fig. 11 Another example of active contours on a natural image. The evolution of the 9 contours is performed simultaneously without loss of efficiency

the ability to perform several active contours evolution simultaneously. It is illustrated by Fig. 13, where different steps of the contours evolutions are presented. The graph is a 4-adjacency grid graph, and the 4 initial sets of labels (or in other words the initial position of contours) were deliberately roughly placed, to illustrate the behavior of the algorithm in such configuration. Figure 11 presents another example which involves 9 distinct fronts/contours. Here again, the seeds are deliberately roughly placed to illustrate the robustness of the method to imprecise initialization

Remark 4 In Algorithm 3, in order to transcript the time dependence of speed, the speeds \mathcal{F}_i are periodically updated.

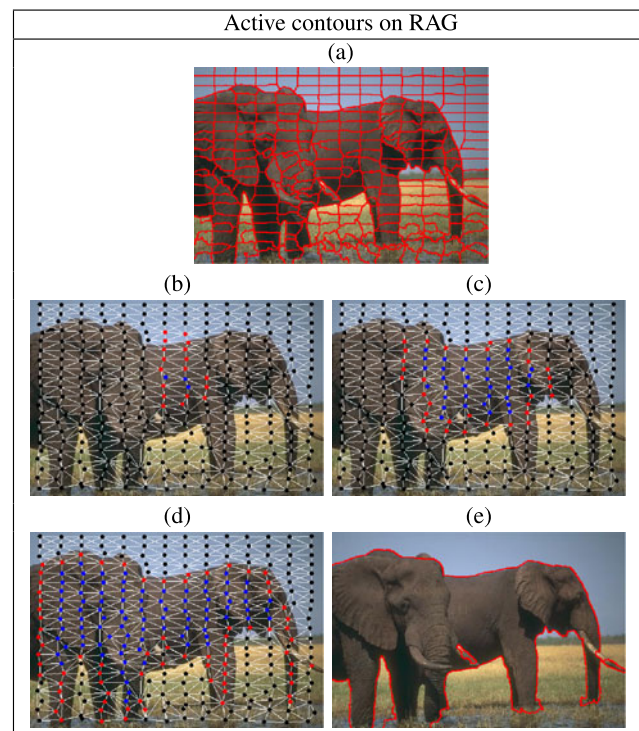


Fig. 12 Active contours on a weighted RAG from one contour. (a) The image partition the RAG is built on. (b, c, d) Different steps of the contour evolution. *Points* represent vertices and *white lines* represent edges. The contour is given by the *red points* and *blue points* represent the vertices which lie inside the contour. (e) The final contour transposed on initial image (Color figure online)

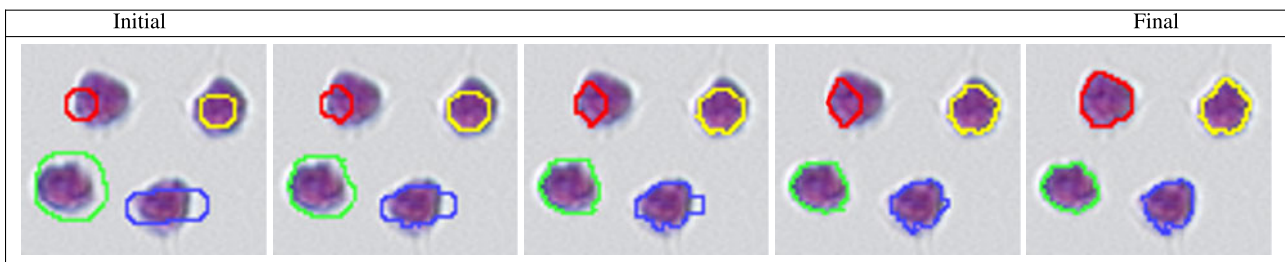


Fig. 13 An application of the proposed algorithm to active contours. Different steps of the simultaneous evolution of 4 contours on the 4-grid graph representation of the given image. See text for details

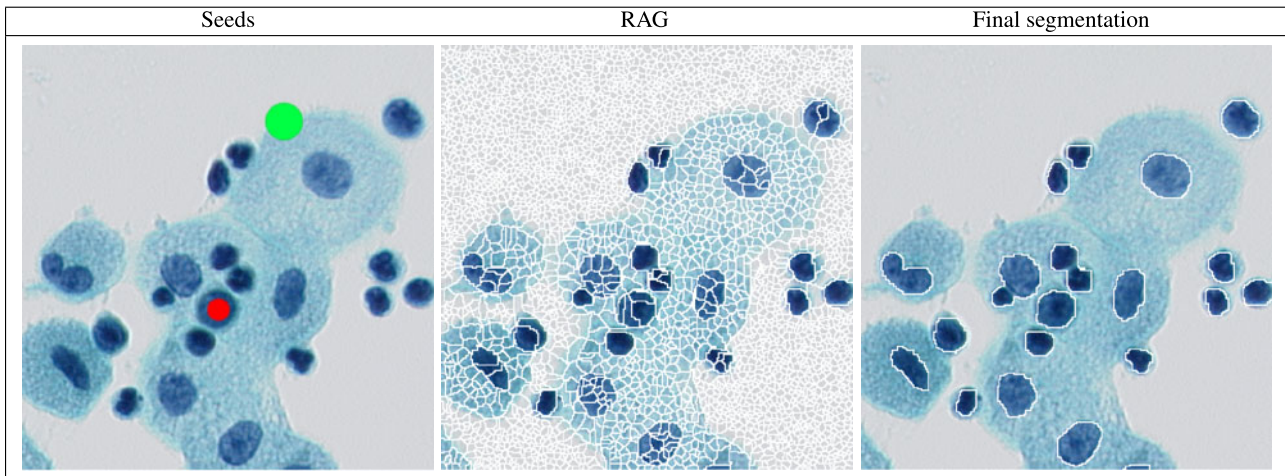


Fig. 14 Semi supervised non-local-region-based segmentation. *First image*: original image with two seeds. *Second image*: image partition with region boundaries superimposed. *Final image*: final segmentation (performed on the RAG). See text for details

5.4.4 Region Adjacency Graph

Another advantage of our graph-based formulation is that the proposed algorithm can be applied to any graph, and therefore any graph representing images. To illustrate such an adaptive behavior, we propose to use other image structures, such as regions maps, instead of pixels grids to build the graph for image segmentation. In the following examples, we use two graphs. The first is a 4-adjacency grid graph used to build an image partition in a supervertices lattice. The obtained region map is then transformed in a second graph, a Region adjacency Graph (RAG), on which the segmentation algorithm is performed. Figure 14 illustrates non-local semi-supervised image segmentation using a RAG. In order to let the labels grow beyond local neighborhood, each vertex neighborhood is extended by a k -nearest neighborhood based on mean color value. The final graph is a $RAG \cup k$ -NNG. One can remark that every cells are extracted even those without initial marker. Another example, presented in Fig. 12, illustrates active contour on a RAG. Finally, using region based graphs has the following advantages.

- Fast computation. By reducing the number of graph vertices (approximately 98 % of reduction in term of vertices

as compared to the pixel based graph), the computing time is reduced due to the reduced number of data to consider.

- Non-local segmentation. The first example shows non-local object segmentation, where only one object is initially marked and the others are found by our method even if the objects are not spatially close or connected.
- Minimal number of initial seeds. With an appropriate graph structure, the segmentation requires a minimal number of initial seeds.

5.4.5 Data Clustering

Finally, this last experiment illustrates the adaptivity and behavior of the proposed algorithm for high-dimensional unorganized data clustering. The clustering is performed similarly to the non-local image segmentation case, using a set of seeds and a weighted graph. The data consists in a set of 500 cells extracted from a cytological slide. Each cell c_i is represented by a feature vector F_{c_i} and the graph is a 7-NNG. Figure 15 presents the graph with 4 initial sets of labels, and the final clustering.

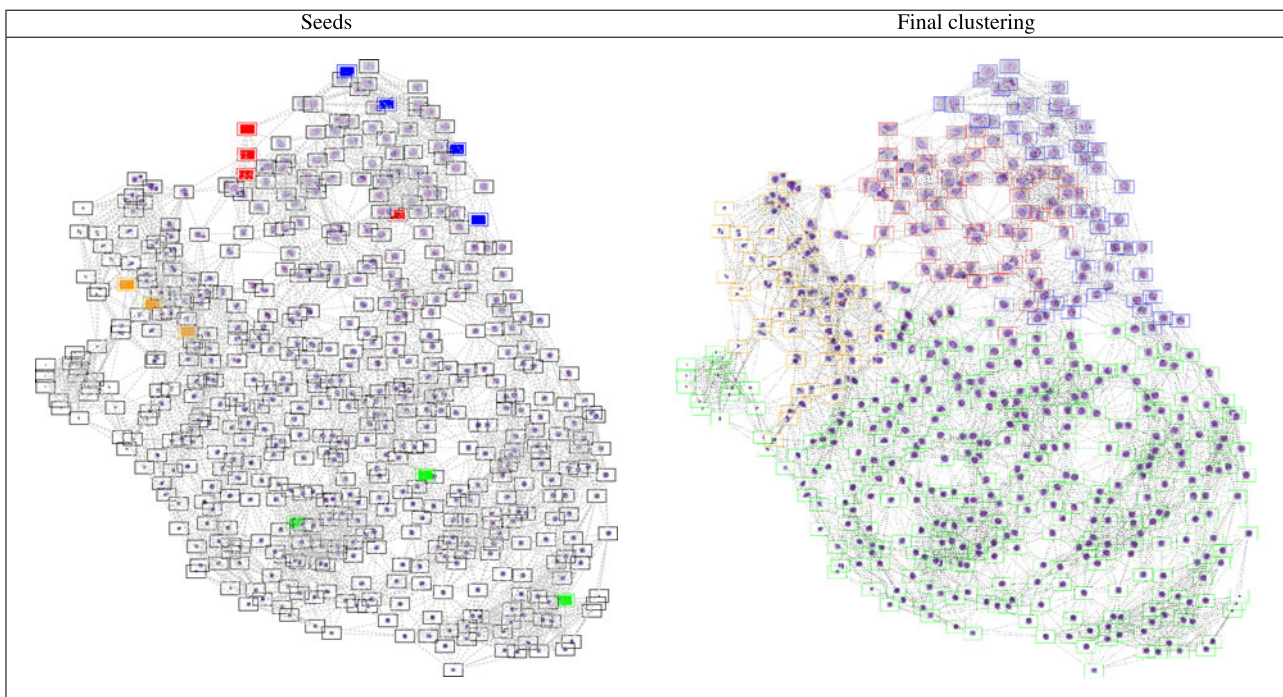


Fig. 15 Unorganized real data clustering. The graph is a 7-nn graph built from a set of 500 cells, and each cell c_i is represented by a feature vector F_{c_i} . At left, the graph with initial labels. At right, the final clustering

6 Conclusion

In this paper, we have considered a new finite difference equation which is an adaptation of the Eikonal equation and defined on weighted graphs. We have shown that this equation has a unique solution and proposed numerical schemes for its resolution with different L_p norms, with $p \in \{1, 2, \infty\}$. Based on such an equation, we have proposed efficient algorithms for multiple labels propagation on graphs of arbitrary topology which enables numerous applications as graph clustering, geodesic distance computation or front tracking. This can be used on many domains such as images, meshes, data or any structure that can be represented by a weighted graph.

References

- Arbelaez, P., Cohen, L.: A metric approach to vector-valued image segmentation. *Int. J. Comput. Vis.* **69**, 119–126 (2006)
- Bai, X., Sapiro, G.: Geodesic matting: a framework for fast interactive image and video segmentation and matting. *Int. J. Comput. Vis.* **82**, 113–132 (2009)
- Bensoussan, A., Menaldi, J.L.: Difference equations on weighted graphs. *J. Convex Anal.* **12**(1), 13–44 (2003)
- Bougleux, S., Elmoataz, A., Melkemi, M.: Discrete regularization on weighted graphs for image and mesh filtering. In: *Proc. SSVM*, pp. 128–139 (2007)
- Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In: *Proc. ICCV*, vol. 1, pp. 105–112 (2001)
- Boykov, Y., Kolmogorov, V., Cremers, D., Delong, A.: An integral solution to surface evolution pdes via geo-cuts. In: *Proc. ECCV*, pp. 409–422 (2006)
- Brockett, R.W., Maragos, P.: Evolution equations for continuous-scale morphology. *Proc. ICASSP* **3**, 125–128 (1992)
- Buades, A., Coll, B., Morel, J.M.: Nonlocal image and movie denoising. *Int. J. Comput. Vis.* **76**, 123–139 (2008)
- Carlini, E., Falcone, M., Forcadell, N., Monneau, R.: Convergence of a generalized fast-marching method for an eikonal equation with a velocity-changing sign. *SIAM J. Numer. Anal.* **46**, 2920–2952 (2008)
- Chan, T.F., Osher, S., Shen, J.: The digital tv filter and nonlinear denoising. *IEEE Trans. Image Process.* **10**, 231–241 (2001)
- Chan, T.F., Vese, L.A.: Active contours without edges. *IEEE Trans. Image Process.* **10**(2), 266–277 (2001)
- Chung, F.: Spectral graph theory. *CBMS Reg. Conf. Ser. Math.* **92**, 1–212 (1997)
- Coifman, R.R., Maggioni, M.: Diffusion wavelets. *Appl. Comput. Harmon. Anal.* **21**(1), 53–94 (2006)
- Coupric, C., Grady, L., Najman, L., Talbot, H.: Power watersheds: a unifying graph-based optimization framework. *IEEE Trans. Pattern Anal. Mach. Intell.* **7**(33), 1384–1399 (2011)
- Courant, R., Friedrichs, K., Lewy, H.: On the partial difference equations of mathematical physics. *Math. Ann.* **100**, 32–74 (1928)
- El Zehiry, N., Xu, S., Sahoo, P., Elmaghraby, A.: Graph cut optimization for the mumford-shah model. In: *Proc. VIIP*, pp. 182–187 (2007)
- Elmoataz, A., Lézoray, O., Bougleux, S.: Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing. *IEEE Trans. Image Process.* **17**(7), 1047–1060 (2008)
- Falcão, A.X., Stolfi, J., de Alencar Lotufo, R.: The image foresting transform: theory, algorithms, and applications. *IEEE Trans. Pattern Anal. Mach. Intell.* **26**, 19–29 (2004)

19. Gilboa, G., Osher, S.: Nonlocal operators with applications to image processing. *Multiscale Model. Simul.* **7**, 1005–1028 (2008)
20. Grady, L.J.: Random walks for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**, 1768–1783 (2006)
21. Grady, L.J., Alvino, C.V.: The piecewise smooth mumford-shah functional on an arbitrary graph. *IEEE Trans. Image Process.* **18**, 2547–2561 (2009)
22. Grady, L.J., Polimeni, J.R.: *Discrete Calculus*. Applied Anal. Graphs Comput. Sc (1999)
23. Hammond, D.K., Vandergheynst, P., Gribonval, R.: Wavelets on graphs via spectral graph theory. *Appl. Comput. Harmon. Anal.* **30**(2), 129–150 (2011)
24. Hein, M., Maier, M.: Manifold denoising. In: *Proc. NIPS*, pp. 561–568 (2006)
25. Hirani, A.N.: *Discrete exterior calculus*. PhD thesis, California Ins. Tech. (2003)
26. Kimmel, R., Sethian, J.A.: Computing geodesic paths on manifolds. In: *Proc. Natl. Acad. Sci.*, pp. 8431–8435 (1998)
27. Kurucz, M., Benczúr, A., Csalogány, K., Lukács, L.: Spectral clustering in social networks. In: *Advances in Web Mining and Web Usage Analysis*, vol. 5439, pp. 1–20 (2009)
28. Leveque, R.J.: *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, Cambridge (2002)
29. Levishtein, A., Stere, A., Kutulakos, K., Fleet, D., Dickinson, S., Siddiqi, K.: Turbopixels: fast superpixels using geometric flows. *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(12), 2290–2297 (2009)
30. von Luxburg, U.: A tutorial on spectral clustering. *J. Stat. Comput.* **17**, 395–416 (2007)
31. Jansen, M.H., Oonincx, P.: *Second Generation Wavelets and Applications*. Springer, Berlin (2005)
32. Neuberger, J.M.: Nonlinear elliptic partial difference equations on graphs. *Exp. Math.* **15**(1), 91–107 (2006)
33. Osher, S., Sethian, J.A.: Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.* **79**, 12–49 (1988)
34. Park, J.H., Chung, S.Y.: Positive solutions for discrete boundary value problems involving the p-Laplacian with potential terms. *Comput. Math. Appl.* **61**, 17–29 (2011)
35. Qian, J., Zhang, Y.T., Zhao, H.K.: Fast sweeping methods for eikonal equations on triangular meshes. *SIAM J. Numer. Anal.* **45**, 83–107 (2007)
36. Ren, X., Malik, J.: Learning a classification model for segmentation. In: *Proc. ICCV*, p. 10 (2003)
37. Rouy, E., Tourin, A.: A viscosity solutions approach to shape-from-shading. *SIAM J. Numer. Anal.* **3**, 867–884 (1992)
38. Sapiro, G., Kimmel, R., Shaked, D., Kimia, B.B., Bruckstein, A.M.: Implementing continuous-scale morphology via curve evolution. *Pattern Recognit.* **26**(9), 1363–1372 (1993)
39. Sethian, J.A.: A fast marching level set methods for monotonically advancing fronts. *Proc. Natl. Acad. Sci.* **41**(2), 199–235 (1999)
40. Sethian, J.A.: Level set methods and fast marching methods: evolving interfaces. In: *Computational Geometry, Fluid Mechanics, Computer Vision and Materials Science*. Cambridge University Press, Cambridge (1999)
41. Shamir, A.: A survey on mesh segmentation techniques. *Comput. Graph. Forum* **27**(6), 1539–1556 (2008)
42. Shi, J., Malik, J.: Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 888–905 (2000)
43. Sinop, A.K., Grady, L.: A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In: *Proc. ICCV*, pp. 1–8 (2007)
44. Ta, V.T., Elmoataz, A., Lézoray, O.: Partial difference equations on graphs for mathematical morphology operators over images and manifolds. In: *Proc. ICIP*, pp. 801–804 (2008)
45. Ta, V.T., Elmoataz, A., Lézoray, O.: Partial difference equations over graphs: morphological processing of arbitrary discrete data. *Lect. Notes Comput. Sci.* **5304**, 668–680 (2008)
46. Ta, V.T., Elmoataz, A., Lézoray, O.: Adaptation of eikonal equation over weighted graphs. In: *Proc. SSVM* (2009)
47. Ta, V.T., Elmoataz, A., Lézoray, O.: Nonlocal pdes-based morphology on weighted graphs for image and data processing. *IEEE Trans. Image Process.* **20**(6), 1504–1516 (2011)
48. Tsitsiklis, J.N.: Efficient algorithms for globally optimal trajectories. *IEEE Trans. Autom. Control* **40**(9), 1528–1538 (1995)
49. Won-Ki, J., Ross, T.W.: A fast iterative method for eikonal equations. *J. Sci. Comput.* **30**(5), 2512–2534 (2008)
50. Zhao, H.: Fast sweeping method for eikonal equations. *Math. Comput.* **74**, 603–627 (2005)
51. Zhou, D., Schölkopf, B.: Adaptive computation and machine learning. In: *Discrete Regularization*, pp. 221–232 (2006)