Efficient Algorithms for Image and High Dimensional Data Processing using Eikonal Equation on Graphs

Xavier Desquesnes[1], Abderrahim Elmoataz[1], Olivier Lézoray[1] and Vinh-Thong Ta[2]

[1] Université de Caen Basse-Normandie, ENSICAEN, CNRS, GREYC Image Team
[2] LaBRI (Université de Bordeaux – CNRS) – IPB

Abstract. In this paper we propose an adaptation of the static eikonal equation over weighted graphs of arbitrary structure using a framework of discrete operators. Based on this formulation, we provide explicit solutions for the $\mathcal{L}_1, \mathcal{L}_2$ and $\mathcal{L}_\infty$ norms. Efficient algorithms to compute the explicit solution of the eikonal equation on graphs are also described. We then present several applications of our methodology for image processing such as superpixels decomposition, region based segmentation or patch-based segmentation using non-local configurations. By working on graphs, our formulation provides an unified approach for the processing of any data that can be represented by a graph such as high-dimensional data.

1 Introduction

Initially designed for geometric optics, the eikonal equation has become a very popular approach for computer graphics and computer vision with numerous applications. For example, solution of the eikonal equation is used to compute geodesic distances on discrete and parametric surfaces [1, 2]. In computer vision, one can quote the shape-from-shading problem [3, 4], median axis or skeleton extraction [5], noise removal, feature detection or segmentation [6, 7], which can be solved using the eikonal equation.
The eikonal equation is a special case of nonlinear Hamilton-Jacobi partial differential equations and is given by:

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

where ϕ is a positive function defined on Ω and $f(x)$ is the traveling time or distance from the source Γ. Then, the eikonal equation can be expressed by

This work was supported under a doctoral grant of the Conseil Régional de Basse-Normandie and of the Coeur et Cancer association in collaboration with the Department of Anatomical and Cytological Pathology from Cotentin Hospital Center.

using the following Hamiltonian:

$$H(x, f, \nabla f) = \|(\nabla f)(x)\| - P(x) \tag{2}$$

where P is a given potential function.

Most of resolutions rely on two main ingredients: a numerical scheme for discretization which leads to nonlinear system and an efficient approach to solve this system. Hamiltonian discretization on Cartesian grids is a well-known problem. Many numerical Hamiltonian schemes can be found in literature, such as Godunov or Lax-Friedrich one [8, 6]. In the same way, numerical schemes have been developed in non-Cartesian domain and can be used on both structured and unstructured meshes [1, 6] . Other schemes exist for some particular case of two and three dimensional manifolds [2] .

The static approach to solve the eikonal equation is usually based on a discretization of the Hamiltonian (2). Numerous methods have been proposed: Rouy and Tourin iterative schemes [3] based on a fixed-point method that solves a quadratic equation (but the complexity behaves as $O(N^2)$ in the worst case), Zhao *fast sweeping* method [9] using a Gauss-Seidel update scheme, Tsitsiklis optimal algorithm with sorted lists of active nodes [10], or the *fast marching* algorithm proposed by Sethian which is the most widely used, to name a few.
Another approach to solve (2) is to consider a time-dependent version:

$$\begin{cases} \frac{\partial f(x,t)}{\partial t} = -\|(\nabla f)(x,t)\| + P(x), & x \in \Omega \subset \mathbb{R}^{\mathrm{m}} \\ f(x,t) = \phi(x), & x \in \Gamma \subset \Omega \\ f(x,0) = \phi_0(x). \end{cases} \tag{3}$$

Recently, approaches have been proposed to transcript partial differential equations in a discrete setting by using partial difference equations on graphs [11]. These methods have been applied for image and data processing [12]. With these approaches, the authors in [13] have proposed an adaptation of (3) on weighted graphs of arbitrary topology. Given a graph $G = (V, E, w)$ their discrete analogous of (3) on graphs is

$$\begin{cases} \frac{\partial f(u,t)}{\partial t} = -\|(\nabla_w^- f)(u)\|_p + P(u) & u \in V \\ f(u,t) = \phi(u) & u \in V_0 \subset V \\ f(u,0) = \phi_0(u) & u \in V \end{cases} \tag{4}$$

with V a finite set of vertices , E a set of edges and w is a similarity function defined on edges. $\|(\nabla_w^- f)(u)\|_p$ denotes the $\mathcal{L}_p$ norm of a discrete weighted directional gradient operator defined on graphs (see Section 2 for detailed weighted graphs notations and definitions). One can see that formulation (4) needs numerous iterations due to finite propagation speed and CFL conditions to converge to the solution of the eikonal equation.

A fastest approach to solve the eikonal equation on weighted graphs is to consider the static version of the equation, which can be solved without expensive

iterations. Then (4) can be rewritten as

$$\begin{cases} \|(\nabla_w^- f)(u)\|_p = P(u). & u \in V \\ f(u) = 0 & u \in V_0 \subset V. \end{cases} \tag{5}$$

Main contributions :
In this work, we propose a static version of the eikonal equation over arbitrary weighted graphs, based on a framework of discrete operators. Explicit solutions of that equation are provided for particular values of $p \in \{1, 2, \infty\}$, and efficients algorithms to obtain such solutions are given. Our general arbitrary weighted graphs formulation provides several advantages. Such a formulation enables the processing of a huge variety of discrete data that can be represented by a weighted graph; i.e. data with any structures or topologies, and embed in spaces of arbitrary dimensions (these dimensions could in fact be very high). Considering image processing, one can find a good approximation of euclidean distance with an appropriate weight function and a large neighborhood [14]. Moreover, the given formulation enables to deal with hierarchical segmentation or textured images segmentation within a unique formulation. This paper also proposes an application to high dimensional data clustering.

Paper organization :
The rest of this paper is organized as follows. In Section 2, definitions and notations used in this work are provided. In Section 3, we present our adaptation of the static formulation of the eikonal equation on weighted graphs and provide explicit solutions for $p = \{1, 2, \infty\}$. Finally, efficient algorithms that can consider any graphs are presented to obtain these solutions. Section 4 presents experiments that show the potentialities of the proposed methodology. Section 5 concludes.

2 Graph Definitions and Operators on Weighted Graphs

We begin by briefly reviewing some basic definitions and operators on weighted graphs which are the main components of our adaptation of the static eikonal equation.

Notations and Definitions. We assume that any discrete domain can be modeled by a weighted graph. Let $G = (V, E, w)$ be a weighted graph composed of two finite sets: $V = \{u_1, ...u_n\}$ of n vertices and $E \subset V \times V$ a set of weighted edges. An edge $(u, v) \in E$ connects two adjacent vertices u and v . The weight w_{uv} of an edge (u, v) can be defined by a function $w : V \times V \to \mathbb{R}^+$ if $(u, v) \in E$, and $w_{uv} = 0$ otherwise. We denote by $N(u)$ the neighborhood of a vertex u, i.e. the subset of vertices that share an edge with u. In this paper, graphs are assumed to be connected, undirected and with no self loops.

Let $f : V \to \mathbb{R}$ be a discrete 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 defined on V.

Operators on Weighted Graphs. For better comprehension of the next Section, we now quickly recall some operators on weighted graphs as they are defined in [13]. Considering a weighted graph $G = (V, E, w)$ and a function $f \in \mathcal{H}(V)$, the *weighted discrete partial derivative operator* of f is

$$(\partial_v f)(u) = \sqrt{w_{uv}}(f(v) - f(u)). \tag{6}$$

Two directional partial derivative operators are defined in [13] but, in this work, we restrict ourself to the use of the *internal* one which is defined as

$$(\partial_v^- f)(u) = -\sqrt{w_{uv}}\min(0, f(v) - f(u)). \tag{7}$$

The *weighted directional discrete gradient* $(\nabla_w^- f)(u)$ defined at a vertex $u \in V$ is the vector of all internal partial derivatives:

$$(\nabla_w^- f)(u) = \Big((\partial_v^- f)(u)\Big)_{(u,v)\in E}^T \tag{8}$$

the corresponding $\mathcal{L}_p$ norm is

$$\|(\nabla_w^- f)(u)\|_p = \left(\sum_{v\sim v} w_{uv}^{p/2}\max(0, f(u) - f(v))^p\right)^{1/p}, \tag{9}$$

and for the $\mathcal{L}_\infty$ norm we have

$$\|(\nabla_w^- f)(u)\|_\infty = \max_{v\sim u}(\sqrt{w_{uv}}\max(0, f(u) - f(v))). \tag{10}$$

One can note that these previous definitions are defined on graphs of arbitrary topology, and can be used to design a general method for solving the eikonal equation on any discrete data sets.

3 Proposed Formulation and Algorithms

In this Section, we present our adaptation of the static eikonal equation over weighted graphs and provide explicit solutions of the equation for particular values of $p \in \{1, 2, \infty\}$. Efficient algorithms are provided to obtain such solutions.

3.1 Eikonal Equation on Weighted Graphs

Starting from the continuous formulation (2) and inspired by its time dependent approach over weighted graphs (4), we obtain a discrete adaptation of the static version of the eikonal equation. Given a graph $G = (V, E, w)$ and a function $f \in \mathcal{H}(V)$:

$$\begin{cases} \|(\nabla_w^- f)(u)\|_p = P(u). & u \in V \\ f(u) = 0 & u \in V_0, \end{cases} \tag{11}$$

where $V_0 \subset V$ corresponds to the initial set of seed vertices. Using norms defined in (9) and (10), we obtain the following equations for the $\mathcal{L}_p$ and $\mathcal{L}_\infty$ norms.

$$\left(\sum_{v \sim v} w_{uv}^{p/2} \max(0, (f(u) - f(v)))^p \right)^{1/p} = P(u), \qquad p \in \{1, 2\}. \tag{12}$$

$$\max_{v \sim u} (\sqrt{w_{uv}} \max(0, f(u) - f(v))) = P(u), \qquad p = \infty. \tag{13}$$

In next Sections, we present numerical schemes and algorithms to approximate the solution of these equations.

3.2 Numericals Schemes and Algorithms

As the main contribution of this paper, we propose numerical schemes to solve the static eikonal equation on arbitrary graphs ((12) and (13)). We emphasize that these schemes can be directly obtained without any spatial discretization since all the operators and functions involved in these equations are discrete.

Now, we study the case where $p \in \{1, 2\}$. With a simple transformation of variables, from (12) we have

$$\sum_{v \sim u} \left[\frac{(x - f(v))^+}{h_{uv}} \right]^p = P(u)^p \tag{14}$$

where $x = f(u)$, $huv = \sqrt{1/w_{uv}}$ and $\max(0, x)$ is denoted $(x)^+$. Then (14) can be rewritten as

$$\sum_{i=1}^{n} \left[\frac{(x - a_i)^+}{h_i} \right]^p = C^p \tag{15}$$

with $n = \text{card}(N(u))$, $a_i = \{f(v_i) \mid v_i \in N(u) \text{ with } i = 1, ..., n\}$ and $C = P(u)$. One can remark that (15) is independent of the graph formulation. Let $\overline{x}$ be the unique solution of (15). This solution is obtained with an iterative algorithm which uses a sorted list of neighbors $\{a_i\}$. Algorithm uses a temporary variable $\hat{x}_i$ which is computed at the iteration i with the following equation. In the case where $p = 1$:

$$\hat{x}_i = \frac{\left[\sum_{j=1}^{i+1} \left(\prod_{l \neq j, l=1}^{i+1} h_l \right) a_j \right] + \left(\prod_{l=1}^{i+1} h_l \right) C}{\sum_{j=1}^{i+1} \left(\prod_{l \neq j, l=1}^{i+1} h_l \right)}. \tag{16}$$

For the sake of clarity, solution $\hat{x}_i$ for the case where $p = 2$ is not provided but can be obtained similarly.

Finally, the unique solution $\overline{x}$ is equal to $\hat{x}_i$ when $\hat{x}_i \leqslant a_{i+1}$. The iterative algorithm to compute $\overline{x}$ (for $p = \{1, 2\}$) is summarized in Algo. 1.

For the $\mathcal{L}_\infty$ norm formulation (13), the unique solution $\overline{x}$ can be simply computed by the following equation:

Algorithm 1

We know $\exists\ k,\ 1 \leqslant k \leqslant n$ such that $\overline{x}$ is the unique solution of the equation and $a_k \leqslant \overline{x} \leqslant a_{k+1}$
Sort the $a_i,\ i = 1, ..., n$ from the lowest to the greatest values.
$a_{n+1} \leftarrow \infty$
$m \leftarrow 1$
$\hat{x} \leftarrow \infty$
while $\hat{x} \geqslant a_{m+1}$ and $m \leqslant n-1$ **do**
  $\hat{x} \leftarrow$ solution of $\sum_{i=1}^{m} \left[\frac{(x-a_i)^+}{h_i}\right]^p = C^p$ with $p = 1, 2$
  $m \leftarrow m + 1$
end while
$\overline{x} \leftarrow \hat{x}$

$$\overline{x} = \min_{j=1}^{n} (a_j + h_j C) \tag{17}$$

where n corresponds to the number of neighbors. One can remark that this equation is a shortest path algorithm (Dijkstra like).

With this formulation, we need a fast and efficient algorithm to compute the solution at each vertex of an arbitrary graph. Many Hamilton-Jacobi solvers can be used to solve (15). The Fast Marching's updating scheme can be used, but in this paper, we prefer using Jeong and Whitaker Fast Iterative Method (FIM) [15]. The main advantage of this method is to solve the Hamilton-Jacobi equation without expensive data structures.

On an arbitrary graph, FIM consists in an active list of vertices to be updated and initialized with source vertices. Initial solutions are set to 0 for source vertices, ∞ otherwise. At each iteration t, all vertices in the list are updated, i.e. we compute the new solution by solving (11), until convergence: $|\overline{x}_{t+1} - \overline{x}_t| \leq \epsilon$ with $\epsilon \to 0$. Converged points are removed and their neighbors are added if further updates are needed.

Label propagation
Additionally to the previous algorithm, we propose a simple way to propagate an initial set of labels (from a set of source vertices V_0) through the graph, following the evolution of the propagating fronts. Because our approach allows to compute a distance map with many sources, our distance map becomes a nearest-source distance map on each vertex. Then, the propagating front which arrives at a vertex is necessarily the front coming from the nearest source of the vertex (according to the weight function sense). So, each time a distance is updated on a vertex u, we find the neighbor v of u which is the closest to both u and a seed of V_0 and extend the label of v to the current vertex u. The labeling process can be summarized by the following formula: Each time $f(u)$ is updated, the label $L(u)$ is given by

$$L(u) = L(v) \mid v \in N(u), f(v) < f(u) \text{ and } \frac{f(v)}{w_{uv}} = \min_{z \sim u} \left(\frac{f(z)}{w_{uz}}\right) \tag{18}$$

Complexity
If the graph is totally connected, the complexity of the proposed method is $O(N^3)$ (worst case), where N is the number of nodes in the graph. In practice, we use a sparse k-nearest neighbors graph with $k \ll N$, and the worst case complexity decreases to $O(Nk^2)$. For comparison, the iterative method [13] depends on two additional parameters: the number of iterations I needed to reach the steady state and the number of initial seeds S. This yield to a complexity of $O(NkIS)$ that is much more higher than ours in practice (since $IS \gg k$).

Relation with other Schemes
As proposed above, our formulation is independent of the graph structure. One can remark that with adapted graph topology and weight function, the proposed formulation is linked to well-known schemes that have been proposed in literature to solve the eikonal equation such as Osher-Sethian or Dijkstra like schemes. In fact, with $p = 2$ and an m-dimensional grid graph, (12) corresponds to the Osher-Sethian discretization scheme. With an unweighted graph and the $\mathcal{L}_\infty$ norm, the Dijkstra like shortest path formulation on graphs can be recovered. Interested readers can refer to [13] for a similar discussion which provide a detailed demonstration.

4 Experiments and Applications

As previously mentioned, our general graph-based formulation allows to deal with any discrete data once they can be represented by graphs. In this Section, we propose some experiments to illustrate that genericity as well as the behavior and the potentiality of such formulation and derived algorithms. All these experiments are processed with a constant potential function $P(u) = 1$. Other potential functions could obviously be used for particular applications.

4.1 Weighted Distances Computation on Graphs

Figure 1 presents the behavior of our formulation for weighted distance computation on arbitrary graphs. For the sake of clarity, the graph used in these experiments was obtained from a grayscale image (Fig.1(a)), with different neighborhoods and weighted functions. Except for the last column, all results are computed with a given weighted function that does not depend on the original image and show the propagating front from the node corresponding to the central pixel of the image (white line are superimposed level-sets). Results (b), (f) and (g) are computed with a 4-adjacency graph, with a constant weight function $f_1 = 1$ and $p = 1, 2, \infty$, respectively. The third and fourth columns show the same distance computation, with respectively 8-adjacency and 16-adjacency for $p = 1$ on the first line and $p = \infty$ on the second. The weight function is provided from [14] and weights each edge in order to reduce the regular-grid metrication errors. One can remark that such graph construction with large neighborhood allows to better approximate the euclidean distance on regular grid. The last column

illustrates the weighted distance computation on a 4-adjacency graph using a weight function $f_2 = e^{\frac{-d_{uv}^2}{\sigma^2}}$ which holds the similarity between two nodes u and v. The distance is computed from the node corresponding to the top left pixel of the image and the resulting propagating front for $p = 2$ and $p = \infty$ are shown at Fig.1(e) respectively Fig.1(j). Because the weight function is designed to catch the topology of the image, the propagating front on the associated graph evolves in a constant way on regular areas (as background or the interior of the apple) and slows down on boundaries. Now, using appropriate graph construction and weight function, we will illustrate the interest of computing such propagating fronts on graphs for image segmentation.

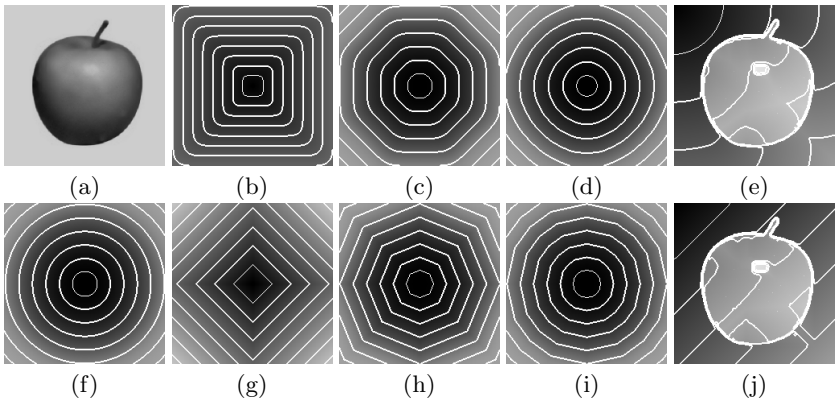


Fig. 1. Illustration of front propagation and weighted distances computation on graph with different configurations (neighborhood), weight functions and p values. Each image represents a weighted distance map from a pixel seed at the center of the image or at the top left corner. See text for details.

4.2 Image Processing

Image Segmentation using Graphs

The objective of the following examples is to illustrate both the genericity of our formulation and the potential of graphs for image processing. Figure 2 presents two images to segment and the associated initial sets of labels. One can remark that each image owns sub-regions of a same class which are not spatially connected (the sky in the first image, or the herd of elephants in the second one). The segmentation is performed on a reduced version of images, a RAG, obtained from a superpixel decomposition. The decomposition and the segmentation are both produced using graphs and our proposed algorithms. Initially developed by Ren and Malik [16], superpixels are an efficient way to reduce image complexity by grouping pixels in a region map while preserving contours. With TurboPixels

[17], Levinshtein et al. have proposed an implementation of superpixels in which image decomposition is obtained by dilating a regular grid of seeds so as to adapt to local image structure. As TurboPixels, our implementation uses a regular-grid of seeds but seeds dilation is controlled by our label propagation method (18) instead of iterative evolving equations. In these examples, we use a 4-adjacency grid graph weighted from the similarity between each connected pixels and we compute the propagating front with $p = \infty$. Resulting partitions are shown in Fig.2. From the obtained partition we associate a weighted RAG coupled with a 2-nearest-symmetric-neighbors graph (2-NSNG) to allow labels to grow beyond local neighborhood. In other words, each node of the RAG is also connected to its two most similar nodes in the whole RAG. Second column of Fig.2 shows these supplementary edges for few nodes (obviously all nodes have additional edges). Then, the same labeling method is performed on these new graphs in order to obtain the final segmentation (third column of Fig.2).

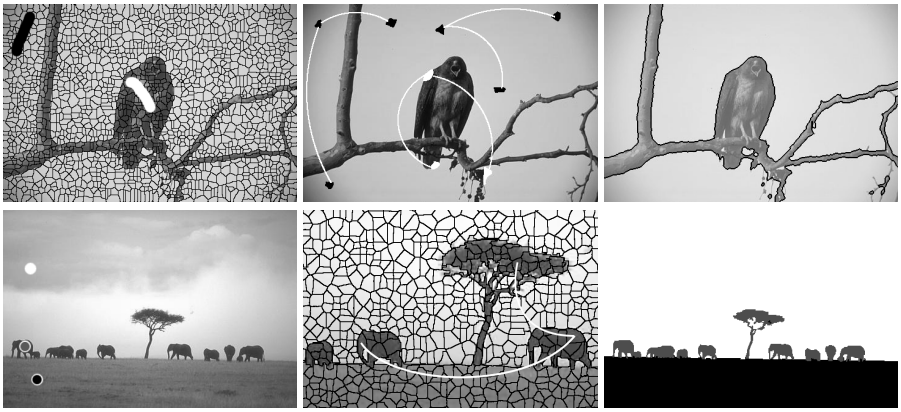


Fig. 2. Image segmentation using graphs (Images are extracted from the Berkeley Segmentation DataSet). The first column presents initial sets of label. The second one shows the RAGs obtained from superpixel decomposition and their 2-NSNG edges (for the sake of clarity additional edges are shown for only few nodes). Third column gives the final segmentations.

One can remark the particularly segmented herd of elephants from a unique label on the left elephant. Finally, we have performed these segmentations only using two successive graph representations of the images and the eikonal equation's based label propagation method proposed in this paper.
Readers interested in semi-supervised segmentation algorithms based on graphs should refer to other recent approaches, as those implementing random walk [18] or graph cuts [19].

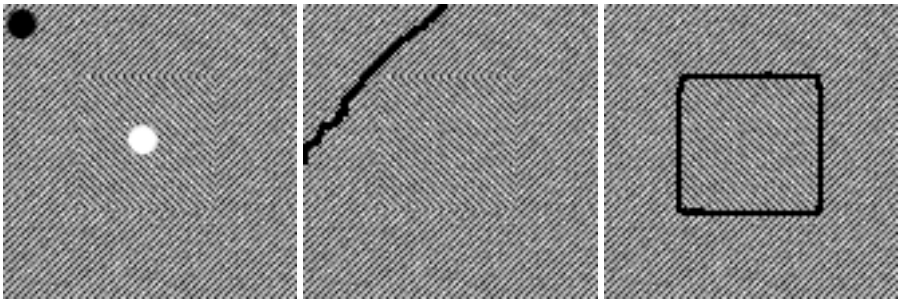
Textured Image Segmentation using High Dimensional Pixel Characterization and Large Neighborhood
The following experiments now show advantages of our method to segment images with texture using high-dimensional pixel characterization and large neighborhood i.e. non-local patch based configurations (interested readers can refer to [13] for more details). Figure 3 shows semi-supervised segmentation using two different graphs. These graphs are computed from the initial image (Fig.3(a)) where initial labels are superimposed. The first graph is a weighted 4-adjacency grid graph where each pixel is characterized by its single intensity which only holds a limited local structure of the image. Figure3(b) shows the segmentation result with this graph. To avoid local structure restriction we build a second graph as a 24-neighbors graph, where each pixel is connected to every pixel in a 5×5 window centered on the pixel (excepting itself). In order to characterize texture, each pixel is represented by a vector of $\mathbb{R}^{25}$, filled with intensity of every pixels in a 5×5 patch centered on the pixel. Figure 3.c shows the segmentation result with this graph which incorporates non-local interactions. One can remark advantages of non-local configurations in order to extract the desired object as compared to the local ones.

4.3 High Dimensional Unorganized Data Processing

We now provide experiments of our method with semi-supervised real-world data clustering. Database used in Fig.4 is a sample of 400 images from MNIST database. This database consist in handwritten digit images of size 28x28. In order to cluster these data, a weighted 3-KNN graph is constructed where edges are weighted with a Gaussian kernel (each vertex is represented by a vector of $\mathbb{R}^{28\times 28}$, filled with intensity of all image pixels.) A few nodes of each class (1 & 0) are marked with initial labels. First line of Fig.4 shows the graph and the obtained clustering using our label propagation method. In order to introduce some difficulties and show the accuracy of our method, each node of the previous graph is also linked to its most dissimilar node. The resulting graph and the new clusters are shown on the second line of Fig.4. One can remark the good clustering results in both cases, and the interest of the given methodology in order to process high-dimensional unorganized data.

5 Conclusion

In this paper, we proposed a solution of the static eikonal equation over weighted graphs using a framework of discrete operators. We showed that the proposed formulation leads to explicit solutions of the equation for different $\mathcal{L}_p$ norms. Efficient algorithms to compute solutions and a label propagation method using the resolution of the eikonal equation on graphs were also provided. The given experiments have shown the behavior and the potentialities of such methodology applied to image processing and high-dimensional data clustering. Good results for high-dimensional unorganized data clustering could suggest interesting outgoing works as hierarchical graph coarse-gaining in order to simplify databases.



(a) Labels (b) Local (c) Non-Local

Fig. 3. Segmentation using patches. This figure shows the boundaries between each resulting partition with only local configuration using a 4-adjacency grid graph (b), or local and non-local configuration using a large 24-neighborhood and a patch-based pixel representation in a 25-dimensional space (c)

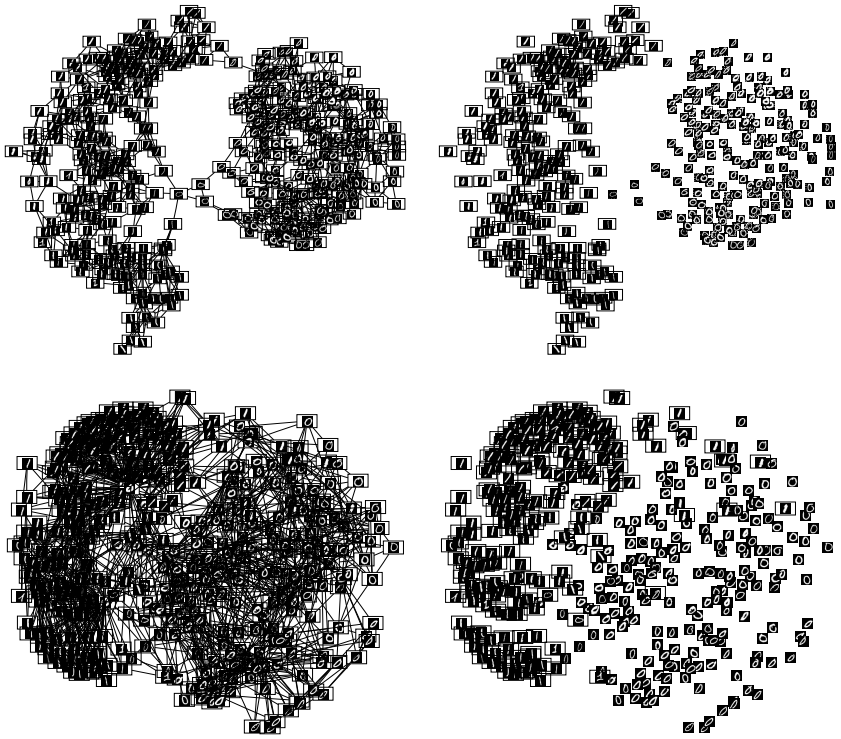


Fig. 4. Handwritten digits database clustering. Each digit is embedded in $\mathbb{R}^{28\times 28}$. The left column shows initial graphs. Second column shows the final classification obtained by using label propagation on graphs. In the resulting graphs, nodes marked with the first label are surrounded with a square.

References

1. Kimmel, R., Sethian, J.A.: Computing geodesic paths on manifolds. In: Proc. Natl. Acad. Sci. USA. (1998) 8431–8435
2. Bronstein, A.M., Bronstein, M.M., Kimmel, R.: Weighted distance maps computation on parametric three-dimensional manifolds. J. Comput. Phys. **225** (2007) 771–784
3. Rouy, E., Tourin, A.: A viscosity solutions approach to shape-from-shading. SIAM J. Numer. Anal. **29** (1992) 867–884
4. Bruss, A.R.: The eikonal equation: some results applicable to computer vision. (1989) 69–87
5. Siddiqi, K., Bouix, S., Tannenbaum, A., Zucker, S.W.: The hamilton-jacobi skeleton. In: ICCV'99: Proceedings of the International Conference on Computer Vision-Volume 2, Washington, DC, USA, IEEE Computer Society (1999) 828
6. Sethian, J.A.: Level set methods and fast marching methods - evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science. In: Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science, Cambridge University Press (1998)
7. Malladi, R., Sethian, J.A.: A unified approach to noise removal, image enhancement, and shape recovery. IEEE Trans. On Image Processing **5** (1996) 1554–1568
8. Zhang, Y.T., Shu, C.W.: High-order weno schemes for hamilton-jacobi equations on triangular meshes. SIAM J. Sci. Comput. **24** (2002) 1005–1030
9. Zhao, H.: A fast sweeping method for eikonal equations. Mathematics of Computation **74** (1999) 603627
10. Tsitsiklis, J.N.: Efficient algorithms for globally optimal trajectories. IEEE Transactions on Automatic Control **40** (1995) 1528–1538
11. Elmoataz, A., Lézoray, O., Bougleux, S., Ta, V.T.: Unifying local and nonlocal processing with partial difference operators on weighted graphs. In: Proc. of LNLA. Volume 44. (2008) 11–26
12. Bougleux, S., Elmoataz, A., Melkemi, M.: Local and nonlocal discrete regularization on weighted graphs for image and mesh processing. Int. J. Comput. Vision **84** (2009) 220–236
13. Ta, V.T., Elmoataz, A., Lézoray, O.: Adaptation of eikonal equation over weighted graph. In: SSVM. Volume 5567 of Lecture Notes in Computer Science., Springer (2009) 187–199
14. Boykov, Y., Kolmogorov, V.: Computing geodesics and minimal surfaces via graph cuts. In: ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision, Washington, DC, USA, IEEE Computer Society (2003) 26
15. Jeong, W.K., Whitaker, R.T.: A fast iterative method for eikonal equations. SIAM J. Sci. Comput. **30** (2008) 2512–2534
16. Ren, X., Malik, J.: Learning a classification model for segmentation. In: ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision, Washington, DC, USA, IEEE Computer Society (2003) 10
17. Levinshtein, A., Stere, A., Kutulakos, K.N., Fleet, D.J., Dickinson, S.J., Siddiqi, K.: Turbopixels: Fast superpixels using geometric flows. IEEE Trans. Pattern Anal. Mach. Intell. **31** (2009) 2290–2297
18. Grady, L.: Minimal surfaces extend shortest path segmentation methods to 3D. IEEE Trans. on Pattern Analysis and Machine Intelligence **32** (2010) 321–334
19. Boykov, Y., Funka-Lea, G.: Graph cuts and efficient nd image segmentation. International Journal of Computer Vision **70** (2006) 109–131