Computer Vision and Image Understanding 123 (2014) 53–64

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Efficient algorithm for finding the exact minimum barrier distance ☆

Krzysztof Chris Ciesielski [a,b,*], Robin Strand [c,d], Filip Malmberg [c,d], Punam K. Saha [e]

[a] Department of Mathematics, West Virginia University, Morgantown, WV 26506-6310, USA
[b] Department of Radiology, MIPG, University of Pennsylvania, Blockley Hall – 4th Floor, 423 Guardian Drive, Philadelphia, PA 19104-6021, USA
[c] Centre for Image Analysis, Uppsala University, Sweden
[d] Department of Radiology, Oncology and Radiation Science, Uppsala University, Sweden
[e] Department of Electrical and Computer Engineering and the Department of Radiology, The University of Iowa, Iowa City, IA 52242, USA

ARTICLE INFO

Article history:
Received 10 May 2013
Accepted 10 March 2014
Available online 28 March 2014

Keywords:
Image processing
Distance function
Distance transform
Minimum barrier
Path strength
Segmentation
Fuzzy Connectedness
Fuzzy distance

ABSTRACT

The minimum barrier distance, MBD, introduced recently in [1], is a pseudo-metric defined on a compact subset D of the Euclidean space $\mathbb{R}^n$ and whose values depend on a fixed map (an image) f from D into $\mathbb{R}$. The MBD is defined as the minimal value of the barrier strength of a path between the points, which constitutes the length of the smallest interval containing all values of f along the path.

In this paper we present a polynomial time algorithm, that provably calculates the exact values of MBD for the digital images. We compare this new algorithm, theoretically and experimentally, with the algorithm presented in [1], which computes the approximate values of the MBD. Moreover, we notice that every generalized distance function can be naturally translated to an image segmentation algorithm. The algorithms that fall under such category include: Relative Fuzzy Connectedness, and those associated with the minimum barrier, fuzzy distance, and geodesic distance functions. In particular, we compare experimentally these four algorithms on the 2D and 3D natural and medical images with known ground truth and at varying level of noise, blur, and inhomogeneity.

1. Introduction

The distance transform, *DT*, mappings [1–8] have been widely used as the effective tools for analyzing object morphology and geometry [9–11]. The value $DT(x)$ of a distance transform map at a point x from the domain C of DT is usually defined as a (possibly signed) distance of x from a fixed target set $B \subset C$, the distance measured with respect to some fixed (possibly generalized) metric on C. Most commonly, C is a bounded subset of the Euclidean space $\mathbb{R}^n$ and DT is defined in terms of the Euclidean distance. For the rectangular shape digital domains C, a fast algorithm for finding the approximate values of the Euclidean distance transform was introduced by Rosenfeld and Pfaltz [12]. An algorithm for computing the exact values of such transform in linear time for the n-dimensional images was described in [13,14] and elaborated on in [15]. Such algorithms for the 2-dimensional images were also presented in [16,17].

The distance transforms used in the image processing commonly take into account the image data [6,18–20]. Most of the distance notions used in such setting define the distance between two points in the image's scene as the minimum cost of a path connecting such points, where the path cost functions depend on the image intensity and differ for different methods. Such defined distance measures include the *connection value* (a variant of Rosenfeld's degree of connectivity [21–23]), which allows for an equivalent characterization of topological watersheds, leading to an efficient Watershed (WS) segmentation algorithm [24,25], as well as the *geodesic distance* (see e.g. [26]). Moreover, the *degree of connectivity*, on the basis of which the Fuzzy Connectedness (FC) algorithms are defined, can be also treated as the distance measure defined in the same form. (See e.g. [27,28,20].) Falcão et al. [20] proved that for a general class of the path cost functions, called *smooth*, including the three examples mentioned above, the related distance transform can be found via Dijkstra's algorithm. For the case of FC, this was further elaborated in [29], including the discussion of the related results from the papers [30,31].

The subject of this paper is the study of the distance transform for the *minimum barrier distance*, MBD, and of the segmentation algorithms associated with it. The MBD for an image f is defined from the path cost function, called *barrier strength*, in a manner described above, where the barrier strength of a path constitutes the

☆ This paper has been recommended for acceptance by Michael Bronstein.

* Corresponding author at: Department of Mathematics, West Virginia University, Morgantown, WV 26506-6310, USA.
E-mail addresses: KCies@math.wvu.edu (K.C. Ciesielski), robin@cb.uu.se (R. Strand), filip@cb.uu.se (F. Malmberg), punam-saha@uiowa.edu (P.K. Saha).

http://dx.doi.org/10.1016/j.cviu.2014.03.007
1077-3142/

length of the smallest interval containing all values of f along the path. However, the barrier strength path cost function is not smooth in the sense defined in [20]. In fact, the naturally defined Dijkstra's algorithms for barrier strength path cost function do not need to return the exact values for MBD. Nevertheless, the output of such algorithms approximate MBD, as proved in [1] and shortly described in what follows.

Section 2 introduces a general framework for constructing the generalized distance mappings. We represent within this framework: the Minimum Barrier Distance, geodesic distance, fuzzy distance, as well as the distance notions that stand behind two popular segmentation algorithms, Fuzzy Connectedness (FC) and Watershed (WS). We also discuss, how the generalized distance mappings can be used to naturally define an image segmentation via seeds competition. In the case of FC theory, this gives the Relative Fuzzy Connectedness (RFC) objects. In Section 3 we introduce the new algorithm, that calculates the MBD in polynomial time. We also discuss, how this new algorithm relates to the standard Dijkstra's algorithm, which can be used to calculate the other distance notions mentioned above.

In Section 4 we present our experimental results. In particular, in Section 4.1 we compare different versions of the algorithms that compute MBD (approximately and exactly) with respect to the execution time and accuracy. In Sections 4.2 and 4.3 we compare the segmentation algorithms corresponding to MBD with the segmentation algorithms corresponding to other distance transforms, that is, with Fuzzy Connectedness, and these corresponding to the geodesic and fuzzy distances. The comparison is quantitative (stability to noise, blur, and the choice of seed points) and qualitative.

2. Generalized distance mappings and related segmentations

In this section we review the constructions of the *generalized distance mappings* (or *generalized metrics*) on a set C, which we define as symmetric functions $d : C^2 \to [0,\infty]$ that satisfy the triangle inequality. (We allow possibility that $d(c,c) > 0$ for some $c \in C$.) The construction encompasses the Minimum Barrier Distance as well as several other popular distance mappings used in imaging. We notice that any generalized distance can be used to naturally define an image segmentation via seeds competition for $c \in C$. In particular, two popular segmentation algorithms, Relative Fuzzy Connectedness (RFC) and Watershed (WS), fall into this category.

Most of the theoretical results that follow are independent of image processing interpretation and are presented in a general graph-theoretical setting. Nevertheless, we point out to the image processing applications at each step of our exposition.

In this paper a *digital image* is identified with its intensity (or attribute) function $f : C \to \mathbb{R}^\ell$, where C is its *domain* (whose elements will be referred to as *spels*, short for space elements) and the value $f(c)$ of f at $c \in C$ represents image intensity (an ℓ-dimensional vector, each component of which indicates a measure of some aspect of the signal, like tissue property or color) at the spel c. It is assumed that image domain comes with an *adjacency relation*, that decides which pairs of spels are adjacent. An image domain C together with its adjacency structure is referred to as a *scene*. In the experimental sections we will concentrate on the images on the rectangular scenes $C = \prod_{i=1}^{k}\{1,\dots,n_i\}$, $k = 2,3$, with either 4-adjacency, in 2D, or 6-adjacency, in 3D.

2.1. Path-induced distance mappings

By a graph we understand a pair $G = \langle C, E\rangle$ with C representing a finite set of its vertices and E the set of its edges G, where an edge connecting c and d from C is identified with an unordered pair $\{c,d\}$. In the image processing applications, we concentrate on the graphs associated with the images $f : C \to \mathbb{R}^\ell$ in which the vertices are the spels (the elements of C) and the set E of edges coincides with the adjacency relation of the image's scene.

A *path* in a graph $G = \langle C, E\rangle$ is any sequence $\pi = \langle \pi(0), \pi(1), \dots, \pi(k)\rangle$ of vertices such that $\{\pi(i), \pi(i-1)\} \in E$ for every $i \in \{1,2,\dots,k\}$. A path $\pi = \langle \pi(0), \pi(1), \dots, \pi(k)\rangle$ is from s to c when $\pi(0) = s$ and $\pi(k) = c$. A family of all paths in G from an $s \in S$ to c is denoted by $\Pi_{S,c}$. We will also write $\Pi_{s,c}$ for $\Pi_{\{s\},c}$.

Now, assume that with any path π in G we have associated its cost: a number $\lambda(\pi) \geqslant 0$ treated as the "length" of π. (The examples of such functions λ are given below.) With any such λ we associate a mapping $d_\lambda : C^2 \to [0,\infty]$ (which need not be a generalized distance) defined as

$$d_\lambda(c,d) = \min\{\lambda(\pi) : \pi \text{ is a path in } G \text{ from } c \text{ to } d\}.$$

In what follows we work mostly with the connected graphs, that is, such that for any vertices c and d there is a path in G from c to d. In such case, all values of d_λ are finite.

In general, d_λ need not be a generalized distance. However, it must be under the assumptions of the following easy fact.

Proposition 1. *Assume that for every path* $\pi = \langle \pi(0), \pi(1), \dots, \pi(k)\rangle$

(i) $\lambda(\pi) = \lambda(\langle \pi(k), \pi(k-1), \dots, \pi(0)\rangle)$, *and*
(ii) $\lambda(\pi) \leqslant \lambda(\langle \pi(0), \dots, \pi(i)\rangle) + \lambda(\langle \pi(i), \dots, \pi(k)\rangle)$ *for every* $0 \leqslant i \leqslant k$.

Then d_λ *is symmetric and it satisfies the triangle inequality.*

Since all path length functions we consider here (except for the auxiliary map β_w^-) satisfy the assumptions of Proposition 1, all considered functions d_λ are the generalized metrics.

In the standard examples, the mappings λ are defined in terms of graph's $G = \langle C, E\rangle$ weight functions: either vertex weight $w : C \to [0,\infty)$ or edge weight $w : E \to [0,\infty)$. In image processing, such weight functions are defined in terms of the intensity function $f : C \to \mathbb{R}^\ell$. (See Section 4 for more on the weight mappings.)

2.1.1. Geodesic distance

For an edge weight map $w : E \to (0,\infty)$, where the value $w(\{c,d\})$ is a (geodesic) distance from c to d, and the path length function $\Sigma(\langle \pi(0), \pi(1), \dots, \pi(k)\rangle) = \sum_{i=1}^{k} w(\{\pi(i-1), \pi(i)\})$, the resulted function d_Σ is the *geodesic metric*. (For the length one path $\pi = \langle \pi(0)\rangle$, the formula is interpreted as $\Sigma(\pi) = 0$.)

2.1.2. Fuzzy distance

For a vertex weight map $w : C \to [0,\infty)$, associated edge weight map $\hat{w} : E \to [0,\infty)$ defined as $\hat{w}(c,d) = \frac{w(c)+w(d)}{2}$, and the path length function $\widehat{\Sigma}(\langle \pi(0), \pi(1), \dots, \pi(k)\rangle) = \sum_{i=1}^{k} \hat{w}(\{\pi(i-1), \pi(i)\})$, the resulted function $d_{\widehat{\Sigma}}$ is the *fuzzy distance*. (For the length one path $\pi = \langle \pi(0)\rangle$, the formula is interpreted as $\widehat{\Sigma}(\pi) = 0$.) Clearly the fuzzy distance is a *pseudo-metric*, that is, it is symmetric and satisfies the triangle inequality (as the assumptions of Proposition 1 hold for $\lambda = \widehat{\Sigma}$); moreover, $d_{\widehat{\Sigma}}(c,c) = 0$ for every $c \in C$. (However, $d_{\widehat{\Sigma}}(c,d)$ can be equal 0 for $c \neq d$.) See [6,19] for more on the fuzzy distance.

2.1.3. Fuzzy Connectedness strength mapping

This is defined for an affinity weight mapping $\kappa : E \to [0,M]$ (usually with $M = 1$) interpreted: the closer the value of $\kappa(\{c,d\})$ is to M, the stronger the vertices c and d are connected. The standard Fuzzy Connectedness strength of a path $\pi = \langle \pi(0), \pi(1), \dots, \pi(k)\rangle$ is defined as $\mu(\pi) = \min_{i=1,\dots,k} \kappa(\{\pi(i-1), \pi(i)\})$, that is, the weakest link of π; the value of $\mu(\langle \pi(0)\rangle)$ is interpreted

as M. The Fuzzy Connectedness path length is defined as $\lambda(\pi) = M - \mu(\pi) = \max_{i=1,\dots,k} w(\{\pi(i-1), \pi(i)\})$, where $w(\{c,d\}) = M - \kappa(\{c,d\})$; that is, $\lambda(\pi)$ is the biggest w-length of a link in π. Then the Fuzzy Connectedness distance map d_λ becomes $d_\lambda(c,d) = M - \mu(c,d)$, where $\mu(c,d) = \max\{\mu(\pi) : \pi \in \Pi_{c,d}\}$ is the standard FC connectivity strength. Note, that this distance d_λ is also a pseudo-metric. For more on this subject see [27,28]. Compare also [24,25,30].

2.1.4. Watershed and the connection value mapping

This is defined for the vertex weight mapping $w : C \to [0,\infty)$, where the value $w(c)$ is interpreted as an elevation at c. Then, for the path length function $\lambda = \beta^+_w$ defined as

$$\beta^+_w(\langle \pi(0), \pi(1), \dots, \pi(k)\rangle) = \max_{i=0,\dots,k} w(\pi(i)),$$

the related distance function $d_\lambda = d_{\beta^+_w}$ is usually referred to as the *connection value* and it leads to the Watershed (WS) segmentation algorithm [24,25]. The connection value related map d_λ is a generalized metric; however, it is not pseudo-metric, since $d_\lambda(c,c)$ can be greater than 0. In what follows we will use also the dual path length function $\lambda = \beta^-_w$ defined as

$$\beta^-_w(\langle \pi(0), \pi(1), \dots, \pi(k)\rangle) = \min_{i=0,\dots,k} w(\pi(i)).$$

Notice that if $M = \max_{c\in C} w(c)$ and the weight function v is defined as $v(c) = M - w(c)$, then the mapping $d^*_{\beta^-_w} = M - d_{\beta^+_v}$ satisfies

$$d^*_{\beta^-_w} = \max\{\beta^-_w(\pi) : \text{ is a path in } G \text{ from } c \text{ to } d\}.$$

In particular, an algorithm that calculates $d_{\beta^+_v}$ can be also used to find $d^*_{\beta^-_w}$.

2.1.5. Barrier distance transform

This is defined for the vertex weight mapping $w : C \to [0,\infty)$. The path length function $\lambda = \beta_w$, referred to as the *barrier strength*, is defined for a path $\pi = \langle \pi(0), \pi(1), \dots, \pi(k)\rangle$ as

$$\beta_w(\pi) = \beta^+_w(\pi) - \beta^-_w(\pi) = \max_{i=0,\dots,k} w(\pi(i)) - \min_{i=0,\dots,k} w(\pi(i)).$$

The related Barrier Distance map $d_\lambda = d_{\beta_w}$ is a pseudo-metric [1].

2.2. The segmentations associated with the generalized metrics

Let d be a generalized distance on C. For a $c \in C$ and a non-empty $W \subset C$ define $d(c,W) = \min\{d(c,w) : w \in W\}$. For any two non-empty sets $S \subset C$ and $T \subset C$ of seeds, S indicating the object and T indicating the background, we associate the object

$$P_d(S,T) = \{c \in C : d(c,S) < d(c,T)\}.$$

Of course, we would expect that

(ST) $P_d(S,T)$ contains S and is disjoint with T.

This is guaranteed only for a proper choice of the seed sets S and T. In particular, if d is a metric (e.g., geodesic), then (ST) holds precisely when S and T are disjoint. For the pseudo metric d (like the case of FC and MBD), (ST) holds precisely when the number $d(S,T) = \min\{d(s,T) : s \in S\}$ is greater than 0.

For the Fuzzy Connectedness pseudo metric d_λ, as defined above, the object $P_d(S,T)$ is precisely the Relative Fuzzy Connectedness, RFC, object. The Watershed object is also often defined as $P_d(S,T)$ (with respect to the distance $d_{\beta^+_w}$). Similarly, the delineated objects for the geodesic, fuzzy, and MB distances we define as $P_d(S,T)$ for their respective distance functions. In either case, the segmentation into k-objects, indicated by the seed sets $S_1, \dots, S_k$ can be defined as $\{P_d(S_i, T_i) : i = 1, \dots, k\}$, where the set T_i is equal to $(S_1 \cup \cdots \cup S_k) \setminus S_i$.

3. The algorithms for DTs and related segmentations

Let λ be an arbitrary path cost function on a graph $G = \langle C, E\rangle$. We assume that the cost of an empty path is infinite: $\lambda(\emptyset) = \infty$. Also, for any path $\pi = \langle \pi(0), \pi(1), \dots, \pi(k)\rangle$ and c connected by an edge with $\pi(k)$, $\pi \wedge c$ is a *concatenation* path $\langle \pi(0), \pi(1), \dots, \pi(k), c\rangle$.

Notice, that for the path cost functions we consider, the value of $\lambda(\pi \wedge c)$ can be calculated from the value of $\lambda(\pi)$ in $O(1)$ time. (More precisely, in the case of the barrier distance, we need to use the values of $\beta^-(\pi)$ and $\beta^+(\pi)$ to find $\beta^-(\pi \wedge c)$ and $\beta^+(\pi \wedge c)$ in $O(1)$ time.) Therefore, for the complexity considerations, we will assume that the values of $\lambda(\langle r\rangle)$, as well as that of $\lambda(\pi \wedge c)$ using $\lambda(\pi)$, can be found in $O(1)$ time.

Algorithm 1. Dijkstra's algorithm $DA(\lambda, R)$

Input: Path cost function λ on a graph $G = \langle C, E\rangle$, non-empty $R \subset C$.
Output: For every $c \in C$, a path π_c from an $r \in R$ to c and $L(c) = \lambda(\pi_c)$.
Auxiliary: Ordered queue Q: if c precedes d in Q, then $L(c) \leqslant L(d)$.
begin
1: For all $c \in C \setminus R$ initialize $\pi_c = \emptyset$ and $L(c) = \infty$;
2: For all $r \in R$ initialize $\pi_r = \langle r\rangle$ and $L(r) = \lambda(\pi_r)$, push all $r \in R$ to Q;
3: **while** Q is not empty **do**
4: Pop d from Q;
5: **for** every $c \in C$ connected by an edge to d **do**
6: Calculate $\ell = \lambda(\pi_d \wedge c)$ using $L(d)$;
7: **if** $\ell < L(c)$ **then**
8: Put $\pi_c = \pi_d \wedge c, L(c) = \ell$, place c to an appropriate place in Q;
9: **end if**
10: **end for**
11: **end while**
12: Return paths π_c and numbers $L(c) = \lambda(\pi_c)$;
end

3.1. Dijkstra's algorithm

Consider the version of the Dijkstra's algorithm presented as Algorithm 1. The algorithm always stops and, for the connected graphs, the returned paths $F = \{\pi_c : c \in C\}$ form a forest rooted at R (i.e., any path in F starts at an $r \in R$ and F contains any initial segment of a path in F). It is easy to see that if λ is the path cost function for geodesic, Fuzzy Connectedness, or watershed distance, then the algorithm $DA(\lambda, R)$ returns λ-minimal paths, that is, having the property that $d_\lambda(c,R) = \lambda(\pi_c)$ for every $c \in C$. (All these path cost functions are smooth, in the sense of [20].) On the other hand, for the barrier strength cost function $\lambda = \beta_w$, this is not the case, as could be seen on the graph from Fig. 1. Indeed, the two paths $\pi_1 = \langle s,a,d,s\rangle$ and $\pi_2 = \langle s,b,d,s\rangle$ between s and c have the barrier weights $\beta_w(\pi_1) = .8 - .4 = .4$ and $\beta_w(\pi_2) = .8 - .5 = .3$, and so $d_{\beta_w}(s,c) = \beta_w(\pi_2) = .3$. However, the initial restriction $\pi = \langle s,b,d\rangle$ of π_2 is not d_{β_w}-optimal for d (as $\beta_w(\pi) = .7 - .5 = .2$, while $d_{\beta_w}(s,d) = \beta_w(\langle s,a,d\rangle) = .5 - .4 = .1 < \beta_w(\pi)$), which is impossible for the Dijkstra's algorithm.

Even for the good cases, when Algorithm 1 returns λ-optimal paths, it seems that to find the object $P_{d_\lambda}(S,T) = \{c \in C : d_\lambda(c,S) < d_\lambda(c,T)\}$ it is necessary to run DA twice: $DA(\lambda, S)$ to compute $d_\lambda(\cdot, S)$ and $DA(\lambda, T)$ to compute $d_\lambda(\cdot, T)$. To avoid this, in the experiments we used a modification $DA^*(\lambda, S, T)$ of $DA(\lambda, W)$ with $W = S \cup T$ obtained by replacing the condition "$\ell < L(c)$" in line 7 with "either $\lambda(\pi_d \hat{} c) < \lambda(\pi_c)$ or

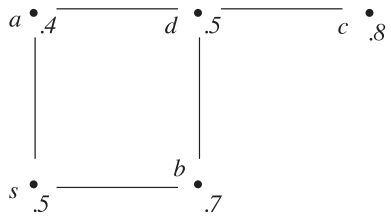

Fig. 1. The minimum barrier distance $d_{\beta_w}(s,c) = .8 - .5$ for the indicated weight function w. However, $DA(\beta_w, \{s\})$ returns suboptimal π_c, with $\beta_w(\pi_c) = .8 - .4$.

$\ell = L(c)$ and $\pi_c(0) \in S$ and $\pi_d(0) \in T$."

The additional condition insures, that for the path of same strength, the algorithm favors those that initiate at T.

If $F = \{\pi_c : c \in C\}$ is a forest returned by $DA^*(\lambda, S, T)$, then the object

$$P^*_{d_\lambda}(S,T) = \{c \in C : \pi_c(0) \in S\}$$

is uniquely defined, in a sense that any two forests returned by $DA^*(\lambda, S, T)$ (which may be different, subject to possible differences in order of processing the vertices from Q) lead to the same set $P^*_{d_\lambda}(S,T)$. Moreover, $P^*_{d_\lambda}(S,T)$ contains $P_{d_\lambda}(S,T)$ and is disjoint with $P_{d_\lambda}(T,S)$; in other words, $P^*_{d_\lambda}(S,T)$ is a union of $P_{d_\lambda}(S,T)$ and some of the vertices from the "boundary" set $\{c \in C : d_\lambda(c,S) = d_\lambda(c,T)\}$. All these facts on $P^*_{d_\lambda}(S,T)$ were rigorously proved in [29] for the FC case, showing, in particular, that $P^*_{d_\lambda}(S,T)$ is the Iterative Relative Fuzzy Connectedness, IRFC, object studied earlier. The arguments presented in [29] easily generalize to the case of a general path cost function.

3.2. Algorithms finding approximation of MBD

Let $\varphi(c,d) = \min\{\beta^+_w(\pi) : \pi \in \Pi_{c,d}\} - \max\{\beta^-_w(\pi') : \pi' \in \Pi_{c,d}\}$, that is,

$$\varphi(c,d) = d_{\beta^+_w}(c,d) - d_{\beta^-_w}(c,d).$$

Clearly, $\varphi(c,d) \leqslant d_{\beta_w}(c,d)$, that is, φ gives a lower bound for the MBD d_β. The equation need not hold, as $\varphi(s,d) = 0 < .1 = d_{\beta_w}(s,d)$ for the example from Fig. 1. Nevertheless, for image induced graphs, the map φ approximates the MBD, as proved in [1] and explained in more details in Theorem 1. The additional advantage of using φ as an approximation of MBD is that its values can be easily computed by the following algorithm introduced in [1].

Algorithm 2. $A^{appr}_{MBD}(\{s\})$

Input: A vertex weight map w on a graph $G = \langle C, E\rangle$, an $s \in C$.
Output: A map $\varphi(\cdot, \{s\})$.
begin
1: Run $DA(\beta^+_w, \{s\})$ and record $d_{\beta^+_w}(c, \{s\}) = \beta^+_w(\pi_c)$ for every $c \in C$;
2: Run $DA(\beta^+_v, \{s\})$, where $v = M - w$ and $M = \max_{c\in C} w(c)$, and record $d_{\beta^-_w}(c, \{s\}) = M - \beta^+_v(\pi_c)$ for every $c \in C$;
3: Return map $\varphi(c, \{s\}) = d_{\beta^+_w}(c, \{s\}) - d_{\beta^-_w}(c, \{s\})$;
end

The following theorem is a variant of a theorem proved in [1]. It specifies an upper bound of a difference between MBD and φ.

Theorem 1. *Let $G = \langle C, E, w\rangle$ be a vertex weighted graph of a rectangular k-D image and let $\varepsilon = \max\{|w(x) - w(y)| : x, y \in C$ are $(3^k - 1)-$adjacent$\}$. Then for every $c, d \in C$ there exists a path $\pi \in \Pi_{c,d}$ with the range in $\left[d_{\beta^-_w}(c,d) - \varepsilon, d_{\beta^+_w}(c,d) + \varepsilon\right]$. In particular, $0 \leqslant d_\beta(c,d) - \varphi(c,d) \leqslant 2\varepsilon$, that is, φ approximates the MBD with at most 2ε error.*

The proof of the theorem translates the discrete MBD to the continuous case and uses the fact that in the continuous case the distance φ coincides with MBD. This last fact is based heavily on a version of Alexander's lemma (see e.g. [32, p. 137]), a deep topological result.

Unfortunately, the number $\varphi(\cdot, S)$ returned by $A^{appr}_{MBD}(S)$ well approximates the MBD distance $d_{\beta_w}(c,S)$ only when S is a singleton. Thus, for larger sets S, to find $\varphi(\cdot, S)$ using A^{appr}_{MBD} within 2ε error, it is necessary to run $A^{appr}_{MBD}(\{s\})$ for every $s \in S$ to find maps $\varphi(\cdot, \{s\})$ and, at the end, compute $\varphi(\cdot, S)$ as $\min_{s\in S}\varphi(\cdot, \{s\})$. Then, the approximation given by Theorem 1 still holds. However, such a procedure essentially increases the computational complexity of finding $\varphi(\cdot, S)$.

Of course, the forest $\{\pi_c : c \in C\}$ returned by the algorithm $DA(\beta_w, R)$ gives an upper bound $\beta_w(\pi_c)$ of MBD map $d_{\beta_w}(c,R)$. However, there is no known theoretical result giving an upper bound for the error for the difference $\beta_w(\pi_c) - d_{\beta_w}(c,R)$. Nevertheless, all three measures, $\varphi(c,s), d_{\beta_w}(c,s)$, and $\beta_w(\pi_c)$, are experimentally compared, see Section 4.1.

3.3. Novel algorithm A_{MBD} for finding the exact MBD

The proof of correctness of the main algorithm presented in this section, A_{MBD}, is a bit involved. However, an idea behind A_{MBD} is relative simple and can be already seen in its simpler version, presented here as the algorithm $A^{simple}_{MBD}(S)$. Therefore, we start here with the discussion of $A^{simple}_{MBD}(S)$.

For a vertex weight map w on a graph $G = \langle C, E\rangle$ and an $a \in \mathbb{R}$ define a modified vertex weight map w_a as

$w_a(c) = w(c)$ provided $w(c) \geqslant a$ and $w_a(c) = \infty$ otherwise.

Theorem 2. *The paths p_c returned by $A^{simple}_{MBD}(S)$ indeed satisfy $\beta_w(p_c) = d_{\beta_w}(c,S)$. Moreover, if $n = |C|$, the size of C, and we assume that $O(|E|) = n$, then $A^{simple}_{MBD}(S)$ stops after at most $O(n^2 \ln n)$ operations.*

In addition, if the range of w is a subset of a fixed set of size $m \leqslant n$, then there exists a small modification of $DA(\beta^+_w, S)$ such that $A^{simple}_{MBD}(S)$ requires at most $O(m^2 n)$ operations.

Algorithm 3. $A^{simple}_{MBD}(S)$

Input: A vertex weight map w on a graph $G = \langle C, E\rangle$, non-empty $S \subset C$.
Output: For every $c \in C$ a path p_c from S to c with $\beta_w(p_c) = d_{\beta_w}(c,S)$.
begin
1: Initialize: $U = \max\{w(s) : s \in S\}$; for $c \in C, p_c = \emptyset$ and $\beta_w(p_c) = \infty$;
2: Push all numbers from $\{w(c) \leqslant U : c \in C\}$ to a queue Q, each only once;
3: **while** Q is not empty **do**
4: Pop a from Q; run $DA(\beta^+_v, S)$ with $v = w_a$, returning π_c's & $\beta_v(\pi_c)$'s;
5: **for** every $c \in C$ **do**
6: **if** $\beta_v(\pi_c) < \beta_w(p_c)$ **then**
7: Put $p_c = \pi_c$ and update the value of $\beta_w(p_c)$ to $\beta_v(p_c)$;
8: **end if**
9: **end for**
10: **end while**
end

Proof. Notice that if there exists a path π from S to c with the range in $[a,\infty)$, then the path π_c returned by $DA(\beta^+_v, S)$ with $v = w_a$ also has this property. This immediately implies correctness of the algorithm.

The first complexity estimate follows from the fact that the complexity of the standard form of $DA(\beta^+_w, S)$ is $O(n \ln n)$, while we run it at most n-many times. The second complexity estimate follows from the fact that, under the assumption, there is a form of $DA(\beta^+_w, S)$ that runs in $O(mn)$ time, as shown in [29] (compare discussion below), and that in this case the loop is executed at most m times. □

Notice that in image processing it is very common that indeed m is a lot smaller than n, in which case the complexity of A^{simple}_{MBD} becomes $O(n)$. However, the constants in this estimate are significant, so A^{simple}_{MBD} runs many times (of order $O(m)$) slower than $DA(\beta^+_w, S)$.

As it can be seen, an idea behind the algorithm A^{simple}_{MBD} is that for every $c \in C$ we consider all possible lower bounds $\beta^-_w(\pi)$ among the paths $\pi \in \Pi_{S,c}$ and, for each such lower bound a, we find a path ${}_a\pi$ minimizing β^+_w; then β_w-minimizer of $\Pi_{S,c}$ is found among such ${}_a\pi$'s. This idea is also present in the algorithm A_{MBD}, though in its dual form: we consider all possible upper bounds $\beta^+_w(\pi)$ among the paths $\pi \in \Pi_{S,c}$ and, for each such upper bound b, we find a path ${}^b\pi$ maximizing β^-_w; a β_w-minimizer of $\Pi_{S,c}$ is one of the ${}^b\pi$'s with the smallest β_w-value. The main difference between A^{simple}_{MBD} and A_{MBD} is in how we choose "all possible one-sided bounds:" in the case of A^{simple}_{MBD} we consider for this purpose all $a \in W, a \leqslant U$. In A_{MBD} this process is considerably more subtle.

More precisely, A_{MBD} can be considered as a Dijkstra's algorithm for finding $\pi_c \in \Pi_{S,c}$ with the minimal value of $\beta^-_w(\pi_c)$, in which "ill-advised order" of the queue Q is used: instead of ordering Q according to the values of β^-_w of already found paths, we order it according the values of β^+_w of such paths. Although this order would be suboptimal if we were only to find the β^-_w-optimal paths, this allows us: (1) to consider as "all possible upper bounds of $\pi \in \Pi_{S,c}$" only the upper bounds of the paths examined during the execution of the algorithm; (2) while considering such an upper bound, say b, to examine all paths $\pi \in \Pi_{S,c}$ with $\beta^+_w(\pi) \leqslant b$, to choose among such paths one, say ${}^b\pi$, with the largest lower bound, and to update our current best β_w-estimate p_c among $\pi \in \Pi_{S,c}$ to ${}^b\pi$, if appropriate.

Algorithm 4. $A_{MBD}(S)$

Input: A vertex weighted graph $G = \langle C, E, w\rangle$, non-empty $S \subset C$.
Output: For every $c \in C$ a path p_c from an $s \in S$ to c such that the number $\beta_w(p_c)$ is the minimum barrier distance from S to c.
Auxiliary: For every $c \in C$ a path π_c from S to c being β^-_w-optimal.
A priority queue Q: if c precedes d in Q then either $\beta^+_w(\pi_c) < \beta^+_w(\pi_d)$ or $\beta^+_w(\pi_c) = \beta^+_w(\pi_d)$ and $\beta^-_w(\pi_c) \geqslant \beta^-_w(\pi_d)$.
begin
1: For every $s \in S$ initialize: $p_s = \pi_s = \langle s\rangle$ and $\beta^-_w(\pi_c) = \beta^+_w(\pi_c) = w(s)$;
2: For every $c \in C \setminus S$ put: $p_c = \pi_c = \emptyset, \beta^-_w(\pi_c) = -\infty$, and $\beta^+_w(\pi_c) = \infty$;
3: Push all $s \in S$ to Q;
4: **while** Q is not empty **do**
5: Pop c from Q;
6: **for** every $d \in C$ connected by an edge to c **do**
7: Set $L^- \leftarrow \beta^-_w(\pi_c \wedge d) = \min\{\beta^-_w(\pi_c), w(d)\}$;
8: Set $L^+ \leftarrow \beta^+_w(\pi_c \wedge d) = \max\{\beta^+_w(\pi_c), w(d)\}$;
9: **if** $L^- > \beta^-_w(\pi_d)$ **then**
10: Set $\pi_d \leftarrow \pi_c \wedge d$, $\beta^-_w(\pi_d) \leftarrow L^-$, $\beta^+_w(\pi_d) \leftarrow L^+$, $\beta_w(\pi_d) \leftarrow L^+ - L^-$;
11: Remove d from Q, if needed; place d into (a right place) in Q;
12: **if** $\beta_w(\pi_d) < \beta_w(p_d)$ **then**
13: Set $p_d \leftarrow \pi_d$;
14: **end if**
15: **end if**
16: **end for**
17: **end while**
end

The proof of the complexity (but not of the correctness) of $A_{MBD}(S)$, presented in Theorem 3, requires the assumption that the graph's degree (i.e., the largest number of edges connected to a single vertex) is small, $O(1)$, with respect to the size n of the set C of graph's vertices. This assumption is essentially always true for the graphs associated with images. Moreover, we make some additional assumptions about the structure of the queue Q it utilizes. Actually, the algorithm works correctly if Q has a simple structure of a double linked list. However, in such structure the insertion of a vertex, as in the line 11 of A_{MBD}, would require $O(n)$ operation. Therefore, we will consider two other structures for Q. The first is a binary heap that allows insertion and deletion of any element in $O(\ln n)$ time [33]. However, for the graphs associated with image processing, the set Z of possible values of a weight function w is usually restricted to a fixed set of a modest size, most frequently of a form $Z = \{i/D : i = 0, 1, \ldots, m\}$ for m not exceeding $2^{12} = 4096$. In this case, Q can be defined as an array of buckets indexed by the set $V = \{\langle\beta^+, \beta^-\rangle \in Z^2 : \beta^+ \geqslant \beta^-\}$ and ordered as described in the algorithm A_{MBD}. This is, essentially, the structure described in [29]. Each bucket with an index $\langle\beta^+, \beta^-\rangle \in V$ consists of the pointers to vertices c for which $\beta^+_w(\pi_c) = \beta^+$ and $\beta^-_w(\pi_c) = \beta^-$. An advantage of Q to be represented in such an array format is that this allows $O(1)$-time insertion into Q and deletion from Q of any element c with a fixed label $\langle z, \ell\rangle$. Emptying Q in the priority order from the largest to the smallest vertex in V, as done when executing line 5 of the algorithm, may require $O(|V|) = O(m^2)$ operations during the complete execution of $A^{simple}_{MBD}(S)$. For large images, $O(m^2)$ is usually considered as smaller than $O(n)$, favoring the array of buckets implementation. However, in our implementations, including $A^{simple}_{MBD}(S)$, we use Dijkstra's algorithm with the binary heap queue structure.

Theorem 3 (*On the correctness and complexity of* $A_{MBD}(S)$).

CORRECTNESS: *After* $A_{MBD}(S)$ *terminates, we have* $\beta_w(p_c) = d_{\beta_w}(c, S)$ *for all* $c \in C$.
COMPLEXITY: *Let* n *be the size of the graph and* m *be the size of a fix set* Z, *containing* $W = \{w(c) : c \in C\}$. *The algorithm computational complexity is either*
(BH) $O(m\, n \ln n)$, *if we use binary heap as* Q, *or*
(LS) $O(m(n + m))$, *if we use as* Q *a list structure described above.*

Proof of the Complexity Part of Theorem 3. The complexity of each execution of the *while* loop, lines 4–17, is determined by the line 11, which is either $O(\ln n)$ in case (BH), or $O(m)$ in case (LS). Moreover,

($*$) a vertex d can be popped from the queue, line 5, at most m times.

To see this property, notice that with an ith appearance of d in Q we associate, in line 11, a path π_d^i from S to d. For d to appear in Q for the $(i+1)$-st time, we must have executed the line 11, meaning that $\beta_w^-(\pi_d^{i+1}) > \beta_w^-(\pi_d^i)$. This means that $\langle \beta_w^-(\pi_d^i)\rangle_i$ is a strictly increasing sequence of numbers from W. So, the sequence cannot have more than m elements and $(*)$ is proved.

Now, in the case of (BH), $(*)$ implies that the loop can be executed at most mn-times, meaning that the algorithm's complexity is $O(mn \ln n)$.

In the case of (LS), the true execution of the loop is $O(mn)$. However, in addition, we may need $O(m^2)$ operations for searching, in line 4, for the top of the queue. This gives complexity $O(mn)+O(m^2) = O(m(n+m))$. □

Before we prove the correctness of $A_{MBD}(S)$, it might be useful to follow the execution of $A_{MBD}(\{s\})$ for the graph from Fig. 1. In this example, we use the convention that $\beta_w^+(\emptyset) = \infty$ and $\beta_w^-(\emptyset) = -\infty$. After the execution of lines 1–3, we have $\pi_s = p_s = \langle s\rangle$, and this state does not change, so we will not list it below. The state of the remaining variables is listed as follows, where index i represents the time just before the ith execution of line 4 (so state 1, is just after the initialization).

1: $\pi_x = p_x = \emptyset$ for $x \in \{a,b,c,d\}$ and $Q = \langle s\rangle$;
2: $\pi_a = p_a = \langle s,a\rangle, \pi_b = p_b = \langle s,b\rangle, \pi_c = p_c = \emptyset, \pi_d = p_d = \emptyset$, $Q = \langle a,b\rangle$ $(\beta_w^+(\pi_a) = .5 < .7 = \beta_w^+(\pi_b))$;
3: $\pi_a = p_a = \langle s,a\rangle, \pi_b = p_b = \langle s,b\rangle, \pi_c = p_c = \emptyset, \pi_d = p_d = \langle s,a,d\rangle$, $Q = \langle d,b\rangle$ $(\beta_w^+(\pi_d) = .5 < .7 = \beta_w^+(\pi_b))$;
4: $\pi_a = p_a = \langle s,a\rangle, \pi_b = p_b = \langle s,b\rangle, \pi_c = p_c = \langle s,a,d,c\rangle$, $\pi_d = p_d = \langle s,a,d\rangle, Q = \langle b,c\rangle$ $(\beta_w^+(\pi_b) = .7 < .8 = \beta_w^+(\pi_c))$;
5: $\pi_a = p_a = \langle s,a\rangle, \pi_b = p_b = \langle s,b\rangle, \pi_c = p_c = \langle s,a,d,c\rangle$, $p_d = \langle s,a,d\rangle, \pi_d = \langle s,b,d\rangle, Q = \langle d,c\rangle$ $(\beta_w^+(\pi_d) = .7 < .8 = \beta_w^+(\pi_c))$;
6: $\pi_a = p_a = \langle s,a\rangle, \pi_b = p_b = \langle s,b\rangle, \pi_c = p_c = \langle s,b,d,c\rangle$, $p_d = \langle s,a,d\rangle, \pi_d = \langle s,b,d\rangle, Q = \langle c\rangle$;
7: $\pi_a = p_a = \langle s,a\rangle, \pi_b = p_b = \langle s,b\rangle, \pi_c = p_c = \langle s,b,d,c\rangle$, $p_d = \langle s,a,d\rangle, \pi_d = \langle s,b,d\rangle, Q = \emptyset$;

Proof of the Correctness part of Theorem 3. To facilitate the proof of algorithm's correctness, we insert into its pseudo code the auxiliary variables: a counter j initialized as 0, and a one dimensional array $\bar{M}$ of numbers. We also expand the line 5 to:

5*: Pop c from Q; Set $j \leftarrow j+1$ and $\bar{M}[j] = \beta_w^+(\pi_c)$.

Let $\langle \bar{M}[j]\rangle_j$ be the array recorded during the execution of the algorithm. Then, by the order imposed on Q, it is non-decreasing (i.e., $\bar{M}[k] \leqslant \bar{M}[k+1]$ for all allowable indices k).

For any real number M define the sets $\Pi_M = \{\pi \in \Pi : \beta_w^+(\pi) \leqslant M\}$ and $\Pi_{<M} = \{\pi \in \Pi : \beta_w^+(\pi) < M\}$. We prove that for every $M \in W$ the following property holds. This will finish the proof, since $(\bullet)$ for $M = \max W$ is precisely the desired correctness of the algorithm.

$(\bullet)$ Let k be the largest index with $\bar{M}[k] \leqslant M$. Then, for every $d \in C$ with $\Pi_{S,d} \cap \Pi_M \neq \emptyset$, after the kth execution of the loop 5–16 we have
(a) π_d maximizes β_w^- on $\Pi_{S,d} \cap \Pi_M$, and
(b) p_d minimizes β_w on $\Pi_{S,d} \cap \Pi_M$.

To see (a), for every $m \in W$ and $\ell = 1,2,\ldots$ consider the property:

(a_ℓ^m) For every $d \in C$, if there exists a $\pi \in \Pi_{S,d} \cap \Pi_M$ of length $\leqslant \ell$ such that $\beta_w^-(\pi) \geqslant m$ and π maximizes β_w^- on $\Pi_{S,d} \cap \Pi_M$, then π_d also maximizes β_w^- on $\Pi_{S,d} \cap \Pi_M$.

We need to show that (a_ℓ^m) holds for every $m \in W$ and $\ell = 1,2,\ldots$. To prove this, for $\mu \in W$ consider the statement:

(a^μ) The property (a_ℓ^μ) holds for every $\ell = 1,2,\ldots$.

By the power of recursion, it is enough to prove that for every $m \in W$: if (a^μ) holds for every $\mu > m$ with $\mu \in W$, then (a^m) is also true.

So, fix an $m \in W$ and assume that (a^μ) holds for every $\mu > m$ with $\mu \in W$. We must show (a^m), that is, that (a_ℓ^m) holds for every $\ell = 1,2,\ldots$. This will be proven by induction on ℓ.

Clearly, (a_ℓ^m) holds for $\ell = 1$, since in this case we need only to consider d from S and then π_d must be equal $\langle d\rangle$, what is insured in line 1. So, assume that for some $\ell = 1,2,\ldots$ the property (a_ℓ^m) holds. We need to prove $(a_{\ell+1}^m)$. To see $(a_{\ell+1}^m)$, fix a $\pi = \langle c_0,\ldots,c_\ell\rangle \in \Pi_{S,d}$ maximizing β_w^- on $\Pi_{S,d} \cap \Pi_M$, for which $\beta_w^-(\pi) \geqslant m$. We need to show that π_d maximizes β_w^- on $\Pi_{S,d} \cap \Pi_M$.

If $\mu = \beta_w^-(\pi) > m$, then this maximization is insured by (a^μ). So, assume that $\beta_w^-(\pi) = m$ and let $\pi' = \langle c_0,\ldots,c_{\ell-1}\rangle$. Notice that for $c = c_{\ell-1}$,

$(*)$ π_c maximizes β_w^- on $\Pi_{S,c} \cap \Pi_M$.

Indeed, if $\beta_w^-(\pi') > m$, then for any π'' maximizing β_w^- on $\Pi_{S,c} \cap \Pi_M$ (which may have length greater than ℓ) we have $\beta_w^-(\pi'') \geqslant \beta_w^-(\pi') > m$. Thus, $(*)$ is insured by (a^μ). On the other hand, if $\beta_w^-(\pi') = m$, then $(*)$ is insured by (a_ℓ^m).

Finally, notice that, by $(*)$, vertex c, together with the path π_c, must have been placed into Q prior to kth execution of the loop 5–16 (through the execution of either line 3 or line 11). Therefore, for some $k' \leqslant k$, this c, with the same path π_c, is popped from Q and after the consecutive execution of the lines 10–11 we must have $\beta_w^-(\pi_d) = \max\{\beta_w^-(\pi_c), w(d)\} \geqslant m = \beta_w^-(\pi)$. This means, that π_d maximizes β_w^- on $\Pi_{S,d} \cap \Pi_M$, finishing the proof of (a).

We prove part (b) by induction along the increasing order of W. So, let $M \in W$ be such that (b) holds for every $M' \in W$ smaller than M. By the power of induction, it is enough to prove that (b) holds for M. So, fix a $d \in C$ with $\Pi_{S,d} \cap \Pi_M \neq \emptyset$ and let π be a path minimizing β_w on $\Pi_{S,d} \cap \Pi_M$. Let p be equal the value of p_d after the kth execution of the loop 5–16. Clearly $p \in \Pi_{S,d} \cap \Pi_M$. To finish the proof, it is enough to show that p minimizes β_w, that is, that $\beta_w(p) \leqslant \beta_w(\pi)$.

If $M' = \beta_w^+(\pi)$ is less than M, then, by the inductive assumption, for some $k' < k$, after the k'th execution of the loop 5–16 we have $\beta_w(p_d) \leqslant \beta_w(\pi)$. Clearly, the value of $\beta_w(p_d)$ cannot increase during the algorithm's execution, so after the kth execution of the loop 5–16 still $\beta_w(p) = \beta_w(p_d) \leqslant \beta_w(\pi)$, that is, p minimizes β_w on $\Pi_{S,d} \cap \Pi_M$. Therefore, in what follows we can assume that $\beta_w^+(\pi) = M$.

If $\Pi_{S,d} \cap \Pi_{<M} = \emptyset$, then, by part (a), for some $k' \leqslant k$ (with $\bar{M}[k'] = M$), during the k'th execution of the loop 5–16, π_d becomes a maximizer of β_w^- on $\Pi_{S,d} \cap \Pi_M$. Then $\beta_w^+(\pi_d) = M$, since $\pi_d \notin \Pi_{<M}$. Thus, the execution of lines 12–14 during the same execution of the loop insures that, from this point on, p_d also minimizes β_w on $\Pi_{S,d} \cap \Pi_M$. So, in what follows we can assume that $\Pi_{S,d} \cap \Pi_{<M} \neq \emptyset$.

Since $\Pi_{<M} \neq \emptyset$, there exists an $M' \in W$ (the largest number in W smaller than M) such that $\Pi_{M'} = \Pi_{<M}$. In particular, $\Pi_{S,d} \cap \Pi_{M'} \neq \emptyset$. By the inductive assumption, $(\bullet)$ holds for M'. So, let k' be the largest index with $\bar{M}[k'] \leqslant M'$ and let and π' be the path π_d immediately after the k'th execution of the loop 5–16. Then, π' maximizes β_w^- on $\Pi_{S,d} \cap \Pi_{M'}$. Let $m' = \beta_w^-(\pi')$.

Next, notice that $m' < \beta_w^-(\pi)$. Indeed, the inequality $m' \geqslant \beta_w^-(\pi)$ would imply $\beta_w(\pi') \leqslant M' - m' < M - \beta_w^-(\pi) = \beta_w(\pi)$, contradicting the fact that π minimizes β_w on $\Pi_{S,d} \cap \Pi_M$. Therefore, the maximum of β_w^- on $\Pi_{S,d} \cap \Pi_M$ is strictly greater than the maximum of

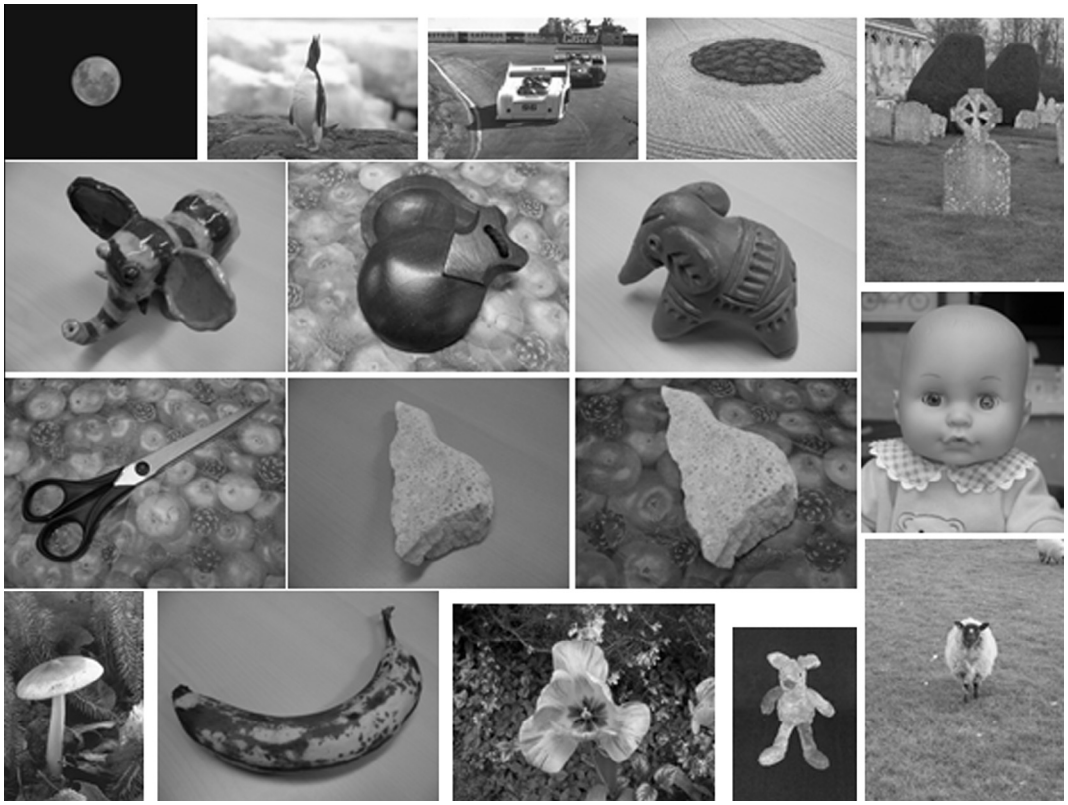

Fig. 2. Images from the grabcut dataset used in the 2D experiments.

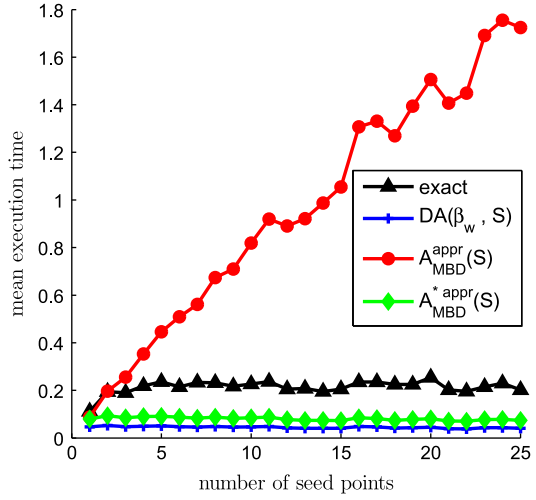

Fig. 3. The mean execution time for the algorithms $A_{MBD}(S)$ (exact), $DA(\beta_w, S)$, $A^{appr}_{MBD}(S)$, and $A^{\star appr}_{MBD}(S)$, for the seed set S having an indicated number of elements.

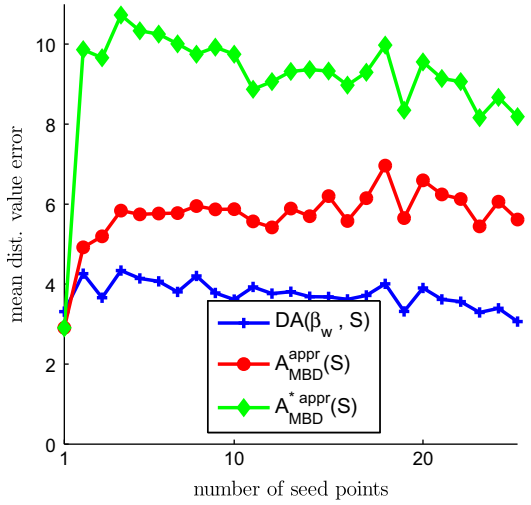

Fig. 4. The mean error of the output values of the algorithms $DA(\beta_w, S)$, $A^{appr}_{MBD}(S)$, and $A^{\star appr}_{MBD}(S)$, as compared with the exact values of MBD returned by $A_{MBD}(S)$. The intensity range of the images is $[0, 255]$.

β^-_w on $\Pi_{S,d} \cap \Pi_{M'}$. So, by (a), there exists a $k'' \in (k', k]$ such that during the k''th execution of the loop 5–16, the condition on line 9 is satisfied. In particular, directly after the execution of the line 10, $m' \leqslant \beta^-_w(\pi_d) \leqslant M$, implying that after the execution of lines 12–14, $\beta_w(p_d) \leqslant M - m' = \beta_w(\pi)$. In particular, $\beta_w(p) \leqslant \beta_w(\pi)$, finishing the proof. □

3.4. Discussion of A_{MBD}

The algorithms $A^{simple}_{MBD}(S)$ and $A_{MBD}(S)$ have the same computation efficiency, when measured in terms of the worst case scenario. So, why do we bother with a more complicated version $A_{MBD}(S)$? Actually, it is easy to argue that the execution time of $A_{MBD}(S)$ is never worse than that of $A^{simple}_{MBD}(S)$ and, in most cases, $A_{MBD}(S)$ is more efficient. To see this, let δ be the degree of the graph (for the images, $\delta = 4$ in 2D and $\delta = 6$ in 3D), put $U = \max\{w(s) : s \in S\}$, and let μ be the size of the set $W_U = \{w(c) \leqslant U : c \in C\}$. In $A^{simple}_{MBD}(S)$ algorithm, the β_w-strength of a path from S to c is checked always (often unnecessarily) between μ- and $\delta\mu$-many times.

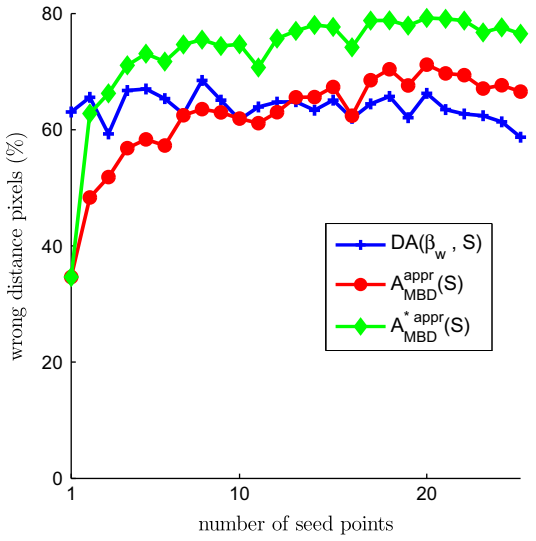

Fig. 5. The mean number of pixels with incorrect value of MBD for the output of the algorithms $DA(\beta_w, S)$, $A^{appr}_{MBD}(S)$, and $A^{\star appr}_{MBD}(S)$, as compared with MBD returned by $A_{MBD}(S)$.

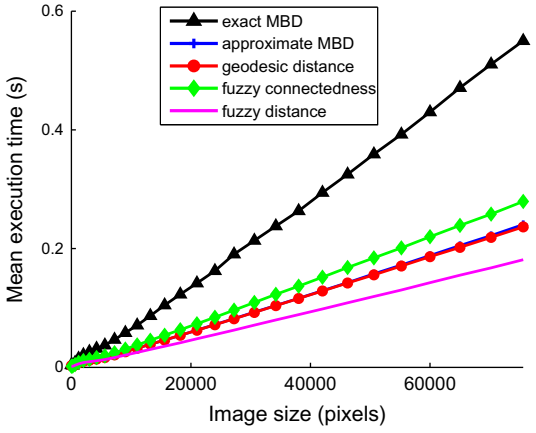

Fig. 6. Comparison of mean execution time on small images obtained by cutting out subimages from the images in Fig. 2. A single seed point was used for each image.

On the other hand, the number of similar checks in $A_{MBD}(S)$ for each spel may be considerably smaller than μ. Certainly, this is the case for each seed. But also, if a spel c is connected to an $s \in S$ with a large value v of β^-_w (meaning that the size μ_c of the set $\{w(d) \in [v, w(s)] : d \in C\}$ is smaller than μ), then the β_w-strength of c will be updated at most $\delta\mu_c$ many times, an improvement from $\delta\mu$.

Also, it is good to mention here that, for some seed sets S and T, the resulted MBD object $P(S,T)$ remains unchanged upon small changes of sets S and T. More specifically, this is the case for the seed pairs $\langle s,t\rangle \in S \times T$ that are essentially separated in the barrier sense, that is, when for any β_w-optimal path p from s to t, $\beta^-_w(p) < \min\{w(s), w(t)\} \leqslant \max\{w(s), w(t)\} < \beta^+_w(p)$. Of course, this robustness for the seed choice is not as potent as the robustness of RFC; however, it is considerable better than for the segmentations associated with the geodesic or fuzzy distances.

4. Experimental evaluation of the algorithms computing MBD and of the related segmentation algorithms

All experiments presented in this section were conducted on a computer HP Proliant ML350 G6 with 2 Intel X5650 6-core processors (2.67 Hz) and 104 GB memory.

4.1. Experimental comparison of the algorithms that compute MBD

In these experiments we have compared four different versions of the algorithms returning MBD: the novel exact MBD algorithm $A_{MBD}(S)$, the interval Dijkstra's algorithm $DA(\beta_w, S)$ approximating MBD from above, the $A^{appr}_{MBD}(S)$ executed once for each seed point, which approximates MBD from below with an error $\leqslant 2\varepsilon$ (see Theorem 1), and $A^{\star appr}_{MBD}(S)$ executed only once even for multiple seeds. The aim for these experiments was to evaluate the practical usefulness of each of these algorithms and to use this information to decide which of them to use in the next set of experiments, comparing MBD with other distance measures.

For the experiments we used 2D images from the grabcut dataset [35], converted to gray scale by using the mean of the three color band values. The images come with the true segmentations. The examples of the images are given in Fig. 2. Their sizes range from 113,032 pixels (for 284×398 image) to 307,200 (for 640×480 image), while the intensity range of the images is $[0, 255]$. The experiments were conducted as follows. For each number $s = 1, \ldots, 25$,

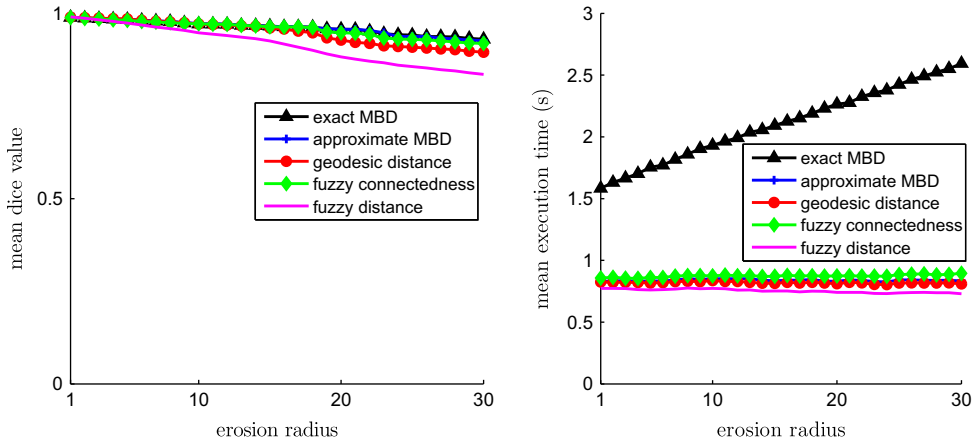

Fig. 7. Comparison of the segmentation of the images from Fig. 2, produced with the distances: exact MBD $A_{MBD}(S)$, approximate MBD $DA(\beta_w, S)$, geodesic $DA(\Sigma, S)$, fuzzy $DA(\widehat{\Sigma}, S)$, and Fuzzy Connectedness $DA(M - \kappa, S)$. The displayed value for each algorithm, for the seeds chosen for the indicated erosion radius, represents the average over the 17 images.

Fig. 8. Example of seed points given by (from left to right) users 1–4, respectively.

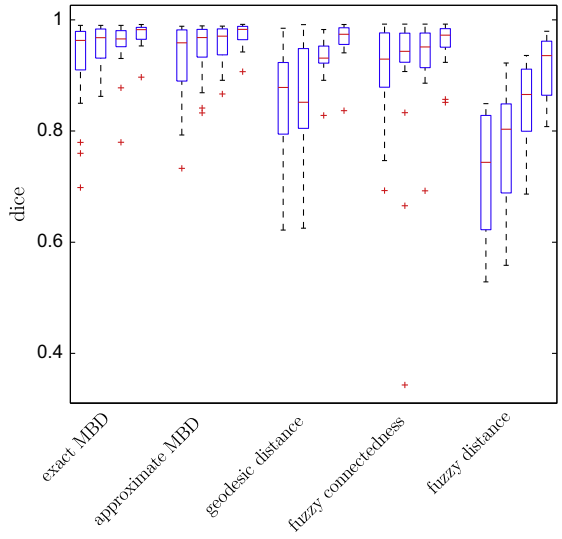

Fig. 9. Boxplots of Dice coefficient for the indicated algorithm. For each distance function, the four boxes correspond to the seed points given by users 1–4, respectively.

the following procedure was repeated 100 times: (1) extract a random image from the subset of the images in the grabcut database; (2) generate randomly the set S of s-many seed points in the image; and (3) run each of the four MBD algorithms on this image with the chosen set S. The averages, for each value of s and each of the algorithms, of the execution time and error in computed distance values are presented in Figs. 3 and 4, respectively. See also Fig. 5.

Based on the presented results, we concluded that the algorithms $A^{\star appr}_{MBD}(S)$ and $A^{appr}_{MBD}(S)$ are not worth pursuing any further: the first one because of its high computing time cost in the presence of multiple seeds, while the second because its higher level of errors, in comparison with the remaining two algorithms.

The experimental performance of the other two algorithms was better than theoretically insured worst case scenarios: (1) The time performance of the exact MBD algorithm $A_{MBD}(S)$ seems to be independent of the number of seeds and is only a bit worst than the execution time of the linear time algorithms $A^{\star appr}_{MBD}(S)$ and $DA(\beta_w, S)$. As expected (Theorem 3), we see in Fig. 6 that, in practice, the execution time of $A_{MBD}(S)$ depends on the image size in a linear manner. (2) The error level of $DA(\beta_w, S)$ is clearly smaller than that of the other two algorithms approximating MBD. Moreover, $DA(\beta_w, S)$ is the most efficient in terms of the computing time.

As a result, in the remaining experiments we used only two MBD algorithms: $A_{MBD}(S)$ and $DA(\beta_w, S)$.

4.2. Comparison of the segmentation algorithms on 2D natural images

In this section, we compare the segmentations, as described in Section 2.2, associated with the following distance functions (see Section 2.1) for the 2D gray-scale digital images $f : C \to [0, \infty)$ obtained from the grabcut dataset, see Fig. 2.

- The *exact MBD* computed with $A_{MBD}(S)$, where $w(c) = f(c)$.
- An *approximate MBD* computed with $DA(\beta_w, S)$, where $w(c) = f(c)$.
- The *geodesic distance* computed with $DA(\Sigma, S)$, where, for adjacent $c, d \in C$, $w(c, d) = |f(c) - f(d)|$.

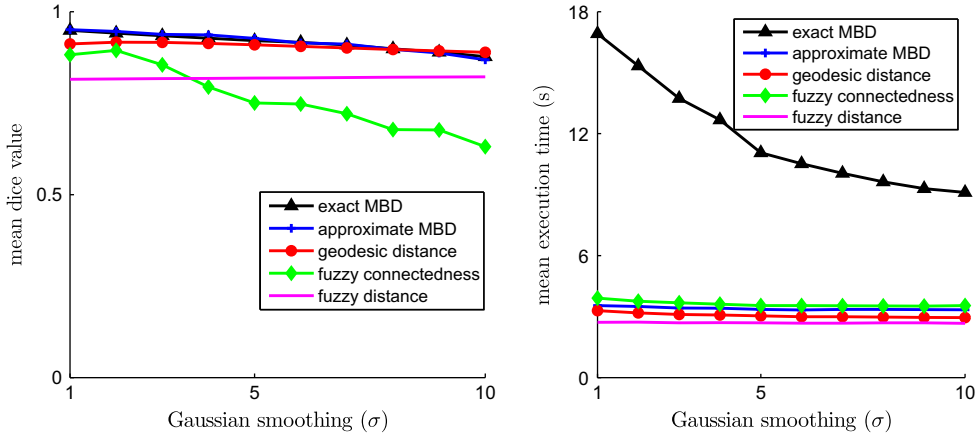

Fig. 10. The performance of the five algorithms as a function of smoothing the images.

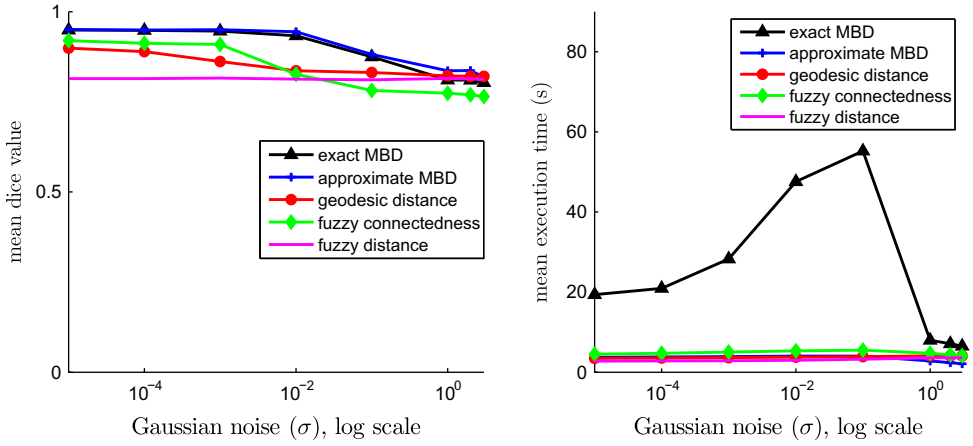

Fig. 11. The performance of the five algorithms as a function of adding noise to the images.

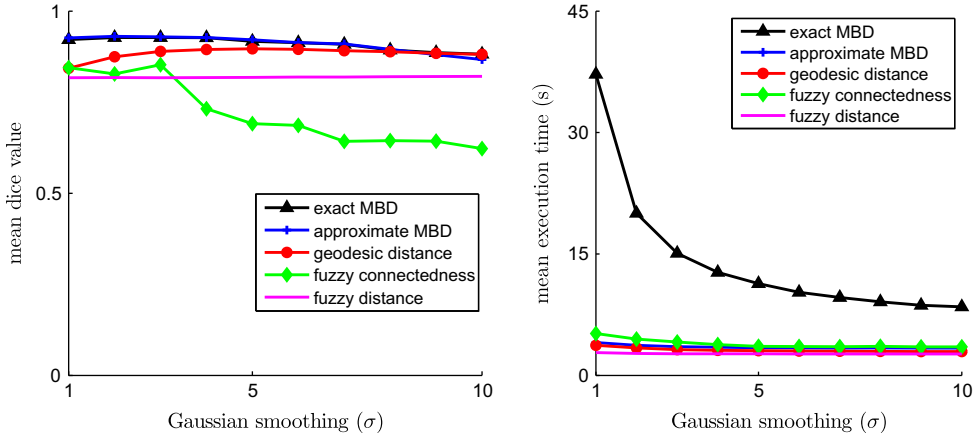

Fig. 12. The performance of the five algorithms as a function of smoothing, applied to the images with added fixed level of noise.

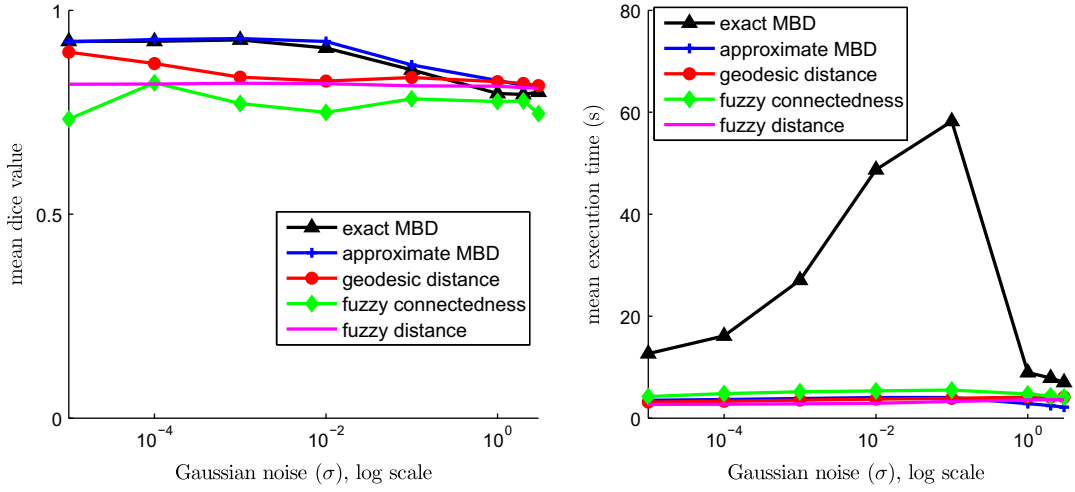

Fig. 13. The performance of the five algorithms as a function of adding noise, applied to the smoothed images.

- The *fuzzy distance* computed with $DA(\widehat{\Sigma}, S)$, where $w(c) = f(c)$.
- The *Fuzzy Connectedness* computed with $DA(w, S)$, where, for adjacent $c, d \in C$, $w(c,d) = M - \kappa(c,d) = |f(c) - f(d)|$.

We start with comparing how the execution time of these algorithms depends on the image size. The summary of our results is displayed in Fig. 6. The algorithms were executed on the small subimages of the images in Fig. 2. For each side length between 1 and 316, a square centered subset of the original images was extracted. A single seed point was placed in the center pixel in the small images. By this procedure, the frequency representation does not depend on image size as would be the case if the images were upsampled or downsampled.

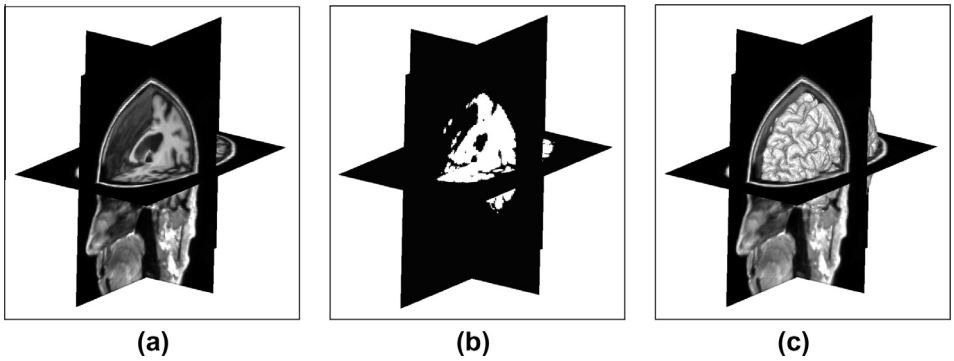

Fig. 14. The 3D T1-weighted MRI image of the brain, smoothed by Gaussian blur with sigma value 0.5. (a) Three perpendicular slices; (b) reference segmentation of the same slices; and (c) surface rendering of the reference segmentation.

Notice that, according to these experiments, the execution time of $A_{MBD}(S)$ depends on the image size in a linear manner, in agreement with Theorem 3.

4.2.1. Seeds chosen by erosion

In these experiments, the seed sets were chosen in the images via erosion of different magnitude of the known true segmentations, see e.g. [34]. Such choice allows varying the seed sets in a more controlled manner, as compared to the alternative of operators specifying seeds interactively or the random seed choice, and thereby we can study the influence of seed sets on results also in a controlled manner. However, there is a concern that such choose may favor the distance measures similar to the Euclidean distance. Although this concern does not seem to be present in our experiments, presented in Fig. 7, we restricted this approach only to the presented small study to avoid any possible bias. (But see also Section 4.3.)

4.2.2. User selected seeds; noise and smoothing influence

In these experiments, involving the images from Fig. 2, the user defined seeds are used. Four different users have placed seed points in the object and background of each image. A sample of such choice is shown in Fig. 8.

Fig. 9 shows boxplots, where the central mark of the box is the median and the edges of the box represent the 25th and 75th percentiles. The whiskers extend to the most extreme data points not considered outliers, which are marked by plus-signs. Four different users have provided object and background seed points for all 17 images. These seed points are used to compute the object for the five different distance function.

The images are degraded by Gaussian smoothing with σ values between 1 and 10. Fig. 10 shows the averaged Dice coefficient results and the execution times for the images with added indicated level of smoothing.

In the experiments presented in Fig. 11 the images were degraded by the additive Gaussian noise with zero mean and variance σ as indicated on the horizontal axis. The figure shows the averaged Dice coefficient results and the execution times for the images with indicated level of noise.

Finally, Figs. 12 and 13 show the similar results for the images with added noise followed by the indicated level of smoothing and for the images with added smoothing followed by the indicated level of noise, respectively.

The experiments presented in this section show that the quality of the segmentations associated with both versions of MBD algorithms compare favorably with those associated the other three methods. This is particularly well visible in the case of blurred images. But the same pattern is also present at the lower level of noise. In the case of the exact MBD distance algorithm, the price of this improvement is a (slightly) higher execution time. (Though, this disadvantage quickly decreases, as a function of level of applied smoothing.) Therefore, if the execution time is an issue, the approximate MBD algorithm $DA(\beta_w, S)$ is the best performer, unless the image is very noisy.

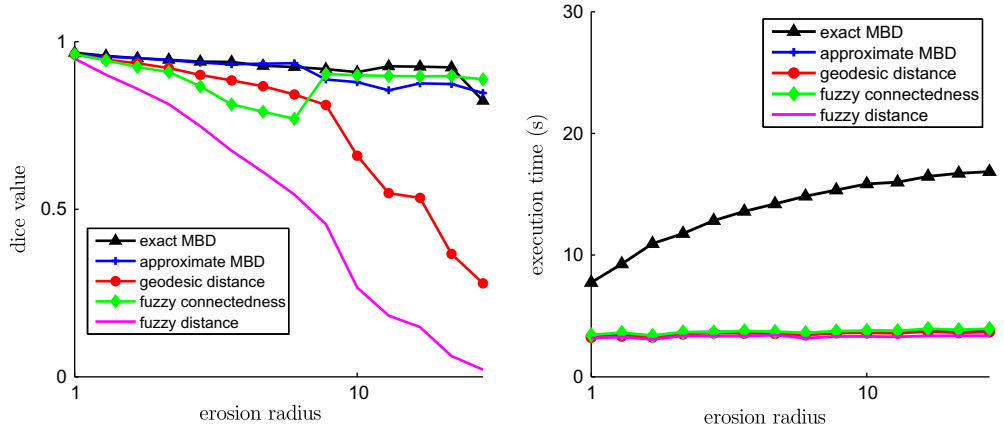

Fig. 15. The performance of the five algorithms on the image from Fig. 14, for the seeds chosen, at the indicated erosion radius, asymmetrically in a way explained in the text.

4.3. Comparison of the segmentation algorithms on a 3D medical image

In the last experiment, presented in this section, we compared the performance of the five algorithms on the 3D T1-weighted MRI image of the brain, shown in Fig. 14.

The presented experiments were performed on the image that has been slightly blurred. We added blur, since the segmentation results performed on the original image were so close to the ground truth for all five algorithms, that there was no basis to differentiate between them.

The seeds were chosen by erosion, with respect to the ground truth. However, to avoid a concern expressed in Section 4.2.1, the erosion was done in an asymmetric manner: increasing radius of the structuring element (with origin located at the border of the structuring element). More specifically, an asymmetric erosion of the ground truth object P of radius r was defined as the set of all spels $c = \langle c_1, c_2, c_3\rangle \in C$ such that the (structuring) set $S(c,s) = \{\langle d_1, d_2, d_3\rangle \in C : c_i \leqslant d_i \leqslant c_i + r \text{ for } i = 1,2,3\}$ is a subset of P.

The results of the experiments are presented in Fig. 15. They show that, in a 3D medical image, the quality of the MBD based segmentations is at least as good (the case of FC) or clearly better (geodesic and fuzzy distances) then those produced by the other methods. This advantage increases, with the decrease of the seed sets.

5. Conclusions

In this paper we introduced a new efficient algorithm, A_{MBD}, that computes the exact values of the Minimum Barrier Distance transform, introduced by the authors in [1]. We provided a detailed proof that A_{MBD} indeed returns the exact MBD and that its execution time is, in the worst case scenario, of order $O(n^2 \ln n)$, n being the size of the image. Moreover, the experimental results indicate that, in practice, the execution time of A_{MBD} is actually linear with respect to n, and comparable to the execution time of the standard Dijkstra's algorithm.

We also investigated an algorithm $DA(\beta_w, S)$ which is faster than A_{MBD} (has, provably, the same complexity as Dijkstra's algorithm) but returns only approximate values of MBD. The presented experiments show that the quality of the output of $DA(\beta_w, S)$ is remarkably similar to that of A_{MBD}.

Finally, we experimentally compared the segmentations associated with both versions of MBD algorithms with that associated with geodesic distance, fuzzy distance, and Fuzzy Connectedness. The segmentation results associated with MBD compare favorable with the other three methods. In particular, MBD is considerable more robust to smoothing than the other algorithms. The same can be also observed when the lower level of noise is added.

References

[1] R. Strand, K.C. Ciesielski, F. Malmberg, P.K. Saha, The minimum barrier distance, Comput. Vis. Image Und. 117 (4) (2013) 429–437.
[2] A. Rosenfeld, J.L. Pfaltz, Distance functions on digital pictures, Pattern Recognit. 1 (1968) 33–61.
[3] G. Borgefors, Distance transformations in arbitrary dimensions, Comput. Vis. Graphics Image Process. 27 (1984) 321–345.
[4] G. Borgefors, On digital distance transforms in three dimensions, Comput. Vis. Image Und. 64 (3) (1996) 368–376.
[5] P.-E. Danielsson, Euclidean distance mapping, Comput. Graphics Image Process. 14 (1980) 227–248.
[6] P.K. Saha, F.W. Wehrli, B.R. Gomberg, Fuzzy distance transform: theory, algorithms, and applications, Comput. Vis. Image Und. 86 (2002) 171–190.
[7] R. Strand, Distance Functions and Image Processing on Point-Lattices: with Focus on the 3D Face- and Body-Centered Cubic Grids, Ph.D. thesis, Uppsala University, Sweden, 2008. <http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-9312>.
[8] R. Strand, B. Nagy, G. Borgefors, Digital distance functions on three-dimensional grids, Theoretical Comput. Sci. 412 (15) (2011) 1350–1363.
[9] T. Hildebrand, P. Rüegsegger, A new method for the model-independent assessment of thickness in three-dimensional images, J. Microsc. 185 (1) (1997) 67–75.
[10] P.K. Saha, Z. Gao, S.K. Alford, M. Sonka, E.A. Hoffman, Topomorphologic separation of fused isointensity objects via multiscale opening: separating arteries and veins in 3-D pulmonary CT, IEEE Trans. Med. Imaging 29 (3) (2010) 840–851.
[11] P.K. Saha, F.W. Wehrli, Measurement of trabecular bone thickness in the limited resolution regime of in vivo MRI by fuzzy distance transform, IEEE Trans. Med. Imaging 23 (1) (2004) 53–62.
[12] A. Rosenfeld, J.L. Pfaltz, Sequential operations in digital picture processing, J. ACM 13 (4) (1966) 471–494.
[13] T. Saito, J.I. Toriwaki, New algorithms for Euclidean distance transformation of an n-dimensional digitized picture with applications, Pattern Recognit. 27 (1994) 1551–1565.
[14] C.R. Maurer Jr., R. Qi, V. Raghavan, A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions, IEEE Trans. Pattern Anal. Mach. Intell. 25 (2) (2003) 265–270.
[15] K.C. Ciesielski, X. Chen, J.K. Udupa, G.J. Grevera, Linear time algorithm for exact distance transform, J. Math. Imaging Vis. 39 (3) (2011) 193–209.
[16] T. Hirata, A unified linear-time algorithm for computing distance maps, Inf. Process. Lett. 58 (3) (1996) 129–133.
[17] A. Meijster, J.B.T.M. Roerdink, W.H. Hesselink, A general algorithm for computing distance transforms in linear time, in: Mathematical Morphology and its Applications to Image and Signal Processing, Kluwer, 2000, pp. 331–340.
[18] J.A. Sethian, Level Set Methods and Fast Marching Methods, Cambridge University Press, 1999.
[19] C. Fouard, M. Gedda, An objective comparison between gray weighted distance transforms and weighted distance transforms on curved spaces, in: A. Kuba, L.G. Nyúl, K. Palágyi, (Eds.), Discrete Geometry for Computer Imagery, 13th International Conference, DGCI 2006, Szeged, Hungary, October 25–27, 2006, Proceedings, Lecture Notes in Computer Science, vol. 4245, Springer, 2006, pp. 259–270.
[20] A.X. Falcão, J. Stolfi, R.A. Lotufo, The image foresting transform: theory, algorithms, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 26 (1) (2004) 19–29.
[21] A. Rosenfeld, Fuzzy digital topology, Inf. Control 40 (1979) 76–87.
[22] A. Rosenfeld, On connectivity properties of grayscale pictures, Pattern Recognit. 16 (1983) 47–50.
[23] A. Rosenfeld, The fuzzy geometry of image subsets, Pattern Recognit. Lett. 2 (1984) 311–317.
[24] G. Bertrand, On topological watersheds, J. Math. Imaging Vis. 22 (2-3) (2005) 217–230.
[25] M. Couprie, L. Najman, G. Bertrand, Quasi-linear algorithms for the topological watershed, J. Math. Imaging Vis. 22 (2-3) (2005) 231–249.
[26] P.J. Toivanen, New geodesic distance transforms for gray-scale images, Pattern Recognit. Lett. 17 (1996) 437–450.
[27] K.C. Ciesielski, J.K. Udupa, Affinity functions in fuzzy connectedness based image segmentation I: equivalence of affinities, Comput. Vis. Image Und. 114 (2010) 146–154.
[28] K.C. Ciesielski, J.K. Udupa, Affinity functions in fuzzy connectedness based image segmentation II: defining and recognizing truly novel affinities, Comput. Vis. Image Und. 114 (2010) 155–166.
[29] K.C. Ciesielski, J.K. Udupa, A.X. Falcão, P.A.V. Miranda, Fuzzy connectedness image segmentation in Graph Cut formulation: a linear-time algorithm and a comparative analysis, J. Math. Imaging Vis. 44 (3) (2012) 375–398.
[30] J. Cousty, G. Bertrand, L. Najman, M. Couprie, Watershed cuts: thinnings, shortest path forests, and topological watersheds, IEEE Trans. Pattern Anal. Mach. Intell. 32 (5) (2010) 925–939.
[31] Camille Couprie, Leo Grady, Laurent Najman, Hugues Talbot, Power watersheds: a unifying graph-based optimization framework, IEEE Trans. Pattern Anal. Machine Intell. 33 (7) (2011) 1384–1399.
[32] M.H.A. Newman, Elements of the Topology of Plane Sets of Points, Cambridge, 1964.
[33] L.G. Nyul, A.X. Falcão, J.K. Udupa, Fuzzy-connected 3D image segmentation at interactive speeds, Graphical Models Image Process. 64 (2003) 259–281.
[34] A.K. Sinop, L. Grady, A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm, in: Proc. of ICCV 07, 2007.
[35] C. Rother, V. Kolmogorov, A. Blake, GrabCut: Interactive Foreground Extraction using Iterated Graph Cuts, in: ACM Transactions on Graphics, SIGGRAPH'04, 2004.