Pattern Recognition Letters 98 (2017) 103–109

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

The *k*-max distance in graphs and images

Michael Holuša*, Eduard Sojka

Department of Computer Science, FEECS, VŠB - Technical University of Ostrava, 17. listopadu 15, Ostrava-Poruba, 708 33, Czechia

ARTICLE INFO

Article history:
Received 8 December 2016
Available online 5 September 2017

Keywords:
Distance measuring
Shortest path
k-max distance
Geodesic distance
Image processing
Image segmentation

ABSTRACT

In this paper, we propose a new distance called the *k*-max distance that is intended for graphs and images. The length of a path is defined as the sum of the *k* maximum arc weights along the path. The distance between two nodes is the length of the shortest path between them. We show that the *k*-max distance is a metric. The algorithm for computing the *k*-max distance is presented. Certain positive properties of the *k*-max distance are shown, namely in the context of measuring the distances for image segmentation. The comparison with the geodesic distance, the max-arc distance, the minimum barrier distance, and the random walker technique is carried out in the segmentation of real-life images.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

Finding the distance between two graph nodes is an important task in graph theory with practical applications in computer science, including computer vision [1–3]. The images may be viewed as graphs in which the nodes correspond to the pixels, and the arcs connect the nodes corresponding to the neighbouring pixels. The arc weights are determined from the intensity values in the neighbouring pixels.

The common algorithm for solving the shortest path problem was introduced by Dijkstra [4]. If the shortest path in graph is measured in the usual way, it corresponds to measuring the distance along the image surface. Therefore, it is often called the *geodesic distance* [5]. This distance is frequently used in image segmentation [6,7], e.g., in the methods based on the minimization of energy function [8–10]. Some more efficient variations of the algorithm for computing the geodesic distance have been introduced in [11,12].

Although the geodesic distance is used in many tasks of image processing, some other metrics have also been introduced in this area, such as the *minimum barrier distance* [13,14], which is shown to be more resistant to noise and to the changes of seed positions [15]. Another distance that was presented recently is the *resistance-geodesic distance* [16] which reduces the possibility that an accidental unwanted path appearing due to imperfections in an image will be used for determining the distance. In [17], the authors introduced an eccentricity that measures the longest geodesic distance across the object, and they also discussed its robustness to noise.

The sensitivity to noise is a well-known drawback of the geodesic distance. Our goal was to find a distance measure that is capable of reducing the influence of noise. In images, for example, such a measure should take into account only the important changes of intensity, which usually occur on the boundaries between the image segments (the areas with different intensities). In the graph, the important changes correspond to the arcs with high weights. On the other hand, small intensity fluctuations inside the segments giving low weights of arcs are not important and should not be taken into account. The problem with the geodesic distance is that it is difficult to distinguish whether its value arises as a sum of a big number of unimportant arc weights (e.g., the distance between two faraway points lying in the same large segment), or whether it is a sum of a smaller number of important arc weights (e.g., the distance between two points in two different segments).

In the *k*-max distance, the length of the path is defined as the sum of the *k* maximum arc weights along the path. The distance is defined as the length of the path that is the shortest one in this sense. Preliminary ideas and results were introduced in [18]. In this paper, a more detailed view is presented. The properties are presented more rigorously. Special attention is paid to the algorithm with the goal to improve its effectiveness. Further experiments are included.

This paper is organized as follows. The drawbacks of the geodesic distance are presented in Section 2. Section 3 describes the *k*-max distance algorithm and its complexity. Section 4 focuses on experiments, and Section 5 is a conclusion.

* Corresponding author.
E-mail address: michael.holusa@vsb.cz (M. Holuša).

http://dx.doi.org/10.1016/j.patrec.2017.09.003
0167-8655/© 2017 Elsevier B.V. All rights reserved.

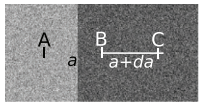

Fig. 1. The synthetic test image with the points between which the distance is measured. We would expect that $d_g(A, B) > d_g(B, C)$. Due to noise, an incorrect result is often provided by the geodesic distance (Fig. 2).

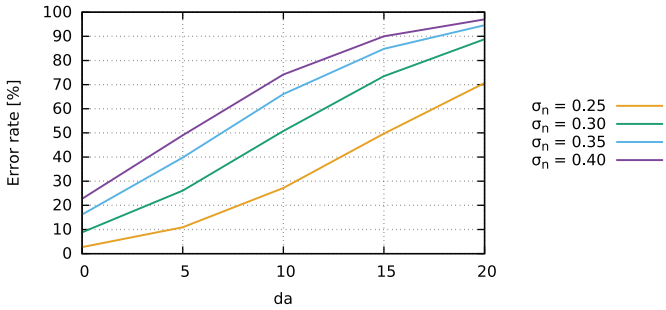

Fig. 2. The error rate of geodesic distance for the test case from Fig. 1, $a = 20$, various values of da, and various values of σ_n are considered.

2. Geodesic distance

This section describes the geodesic distance and its behaviour if the distance is measured in noisy images. Consider a graph and two nodes in it, denoted by A and B, respectively. Let α_{AB} be a path connecting A and B; $\alpha_{AB} = (A \equiv V_1, V_2, \ldots, V_n \equiv B)$; V_i is a node through which the path is running. The length of the path is the sum of the weights of all its arcs $l(\alpha_{AB}) = \sum_{i=1}^{n-1} w_{V_i,V_{i+1}}$, where $w_{V_i,V_{i+1}}$ is the weight of the arc connecting V_i and V_{i+1}. Let $\mathcal{P}_{AB}$ be the set of all existing paths between A and B. The geodesic distance between A and B is the length of the shortest path between A and B, i.e., $d_g(A, B) = \min_{\alpha_{AB} \in \mathcal{P}_{AB}}\{l(\alpha_{AB})\}$. In image processing, the weight of arc is often determined by the equation

$$w_{i,j} = 1.0 - e^{-\frac{(b_i - b_j)^2}{2\sigma_w^2}} + \eta\,, \tag{1}$$

where b_i, b_j are the values of intensities at the pixels corresponding to V_i and V_j, respectively; σ_w is a constant. The value of η determines the price of using the arc regardless of how big the intensity difference between its endpoints is. It can also be $\eta = 0$, which violates the condition $d(A, B) = 0 \Leftrightarrow A \equiv B$, but it reflects the opinion that the distances between any two points in the same image segment should be small or even zero.

To show the behaviour of the geodesic distance, consider the synthetic image containing a unit intensity step, i.e., two segments (Fig. 1). The point A is placed in the left segment, the points B and C are in the right segment. The Gaussian noise is added. The arc weights are computed according to Eq. (1) ($\sigma_w = 0.333$, $\eta = 0.01$). The geodesic distances between A and B, and B and C are computed. Since A and B are in different segments, and B and C are both in the same segment, we expect that $d_g(A, B) > d_g(B, C)$. We check how the expectation is met for various amounts of noise in the image, and for various values of da (see Fig. 1). Even if $da > 0$, we would probably want $d_g(A, B) > d_g(B, C)$.

The fact is that the results are often different. Fig. 2 shows how often the unexpected result $d_g(A, B) \le d_g(B, C)$ is obtained. The error rates are computed as the mean from 10^5 samples. As can be easily understood, the error rate increases with the increasing amount of noise (σ_n stands for the standard deviation of Gaussian noise) and with the increasing value of da. The explanation was given before. The sum of the weights affected by noise may overshadow the useful information, which is the weight of just one arc in this case.

The facts stated above have led us to the idea to define a distance that considers only several arcs with a high weight while the low-weighted arcs are ignored. The Chebyshev distance between two vectors is a distance that takes into account only the maximum difference of vector entries (L_∞ norm). Similarly, only the maximum arc weight along the path can be considered in the graphs, which is known as *max-arc distance*. Apparently, the max-arc distance cannot describe everything what is needed. For example, the length of the path running across two boundaries between segments in an image should be longer than the length of the path running across only one boundary. The k-max distance seems to be the missing part between the max-arc and geodesic distance. For $k = 1$, the k-max distance becomes the max-arc distance. For $k \to \infty$, it becomes the geodesic distance.

3. The *k*-max distance

In this section, the k-max distance is introduced. Let $\sum_{\text{top}_k}(.)$ stand for the sum of the k largest values in a collection of non-negative real numbers. We define the length of a path as the sum of the k largest arc weights along the path, i.e.,$l(\alpha_{AB}) = \sum_{\text{top}_k}(w_{V_1,V_2}, w_{V_2,V_3}, \ldots, w_{V_{n-1},V_n})$, where α_{AB} is a path connecting A and B, V_i are the nodes of the path ($V_1 \equiv A$, $V_n \equiv B$), and $w_{V_i,V_{i+1}}$ are the arc weights. If k is greater than the number of arcs in the path, the distance is given by the sum of all the arc weights, i.e., it equals to the geodesic distance. Let $\mathcal{P}_{AB}$ be the set of all paths between A and B. The k-max distance between A and B is $d_{km}(A, B) = \min_{\alpha_{AB} \in \mathcal{P}_{AB}}\{l(\alpha_{AB})\}$. It can be easily seen that the k-max distance in a weighted undirected graph is a metric since the following is true.

1. $d_{km}(A, B) \ge 0$
   Since the arc weights are positive, their sum is positive.
2. $d_{km}(A, A) = 0$
   The distance $d_{km}(A, A) = 0$ is defined a priori (no edge).
3. $d_{km}(A, B) = d_{km}(B, A)$
   Because $w_{V_i,V_{i+1}} = w_{V_{i+1},V_i}$ and the sum is commutative.
4. $d_{km}(A, C) \le d_{km}(A, B) + d_{km}(B, C)$
   Let $\alpha_{AB} = (A \equiv A_1, A_2, \ldots, A_n \equiv B)$ be the shortest path between A and B (A_i is a path node). Let $\alpha_{BC} = (B \equiv B_1, B_2, \ldots, B_m \equiv C)$ be the shortest path between B and C. The length of α_{AB} is $\sum_{\text{top}_k}(w_{A_1,A_2}, \ldots, w_{A_{n-1},A_n})$, the length of α_{BC} is $\sum_{\text{top}_k}(w_{B_1,B_2}, \ldots, w_{B_{m-1},B_m})$. Consider the path (or a walk in general) $\alpha_{AB} \cdot \alpha_{BC}$ connecting A and C created by concatenating α_{AB} and α_{BC}. Its length is $\sum_{\text{top}_k}(w_{A_1,A_2}, \ldots, w_{A_{n-1},A_n}, w_{B_1,B_2}, \ldots, w_{B_{m-1},B_m})$. The collection of the k top weights determining the length of $\alpha_{AB} \cdot \alpha_{BC}$ contains a certain subcollection from the collection containing the k top weights along α_{AB} and a certain subcollection of the k top weights along α_{BC}. Therefore, the length of $\alpha_{AB} \cdot \alpha_{BC}$ cannot be longer than $d_{km}(A, B) + d_{km}(B, C)$. Furthermore, the shortest path between A and C cannot be longer than the length of $\alpha_{AB} \cdot \alpha_{BC}$ since it would not be the shortest path otherwise.

The rest of this subsection presents our comments on the properties of the k-max distance from the point of view of the classification ($C1$, $C2$, $C4$) introduced in [19,20]. Let α_S be a path to a node S, and let $\langle S, T\rangle$ be the arc connecting S with a node T. Obviously, it makes sense to suppose that $l(\alpha_S) \le l(\alpha_S \cdot \langle S, T\rangle)$, which is the $C1$ condition. Consider two different paths to S, denoted by α_S and β_S, respectively. We claim that the condition $l(\alpha_S) < l(\beta_S) \Leftrightarrow l(\alpha_S \cdot \langle S, T\rangle) < l(\beta_S \cdot \langle S, T\rangle)$ ensures that the optimum path to S is a part of the optimum path to T, which corresponds to the $C2$ condition. Finally, the condition $l(\alpha_S) = l(\beta_S) \Leftrightarrow l(\alpha_S \cdot \langle S, T\rangle) = l(\beta_S \cdot \langle S, T\rangle)$ is the $C4$ condition. In [20], it is shown that the Dijkstra algorithm can be used if either $C1$, $C2$ or $C1$, $C4$ are satisfied, which can be

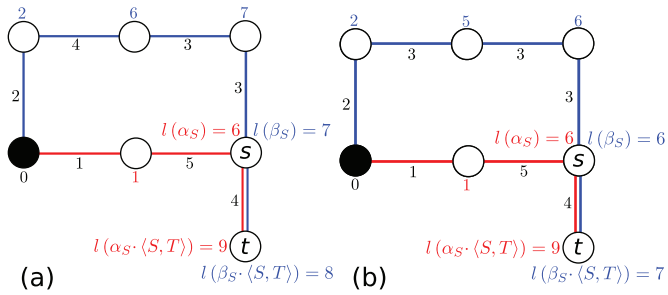

Fig. 3. Counter-examples showing that the k-max distance ($k = 2$ in this case) does not meet the conditions presented in [20]: (a) The C2 property is not satisfied since $l(\alpha_S)$ is optimal while $l(\alpha_S \cdot \langle S, T\rangle)$ is not. (b) The C4 property is not satisfied since $l(\alpha_S \cdot \langle S, T\rangle) > l(\beta_S \cdot \langle S, T\rangle)$ although $l(\alpha_S) = l(\beta_S)$.

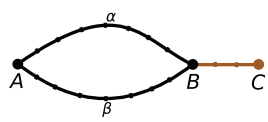

Fig. 4. Measuring the distance $d_{km}(A, C)$.

easily understood. In the Dijkstra algorithm, only a single value that could become the distance, and only a single pointer back are stored in each node at each moment. The conditions $C1$, $C2$ or $C1$, $C4$ ensure that no other values are needed for a correct computation.

The path length in the k-max distance satisfies only the $C1$ condition. Fig. 3 proves that neither C2 nor C4 are satisfied. Therefore, the paths neither create the forest introduced in [19], nor the Dijkstra algorithm can directly be used. Further details are given in the next section.

3.1. The k-max algorithm

In this section, we present the algorithm for computing the k-max distance in graphs. In the sense that the distances are computed from the shortest to the longest, the algorithm may be regarded as similar to well-known Dijkstra's algorithm. On the other hand, substantial modifications are necessary, which is due to the fact that the k-max distance satisfies neither the condition C2 (Bellman's principle of optimality [21]) nor the condition C4 from the previous subsection.

Consider the graph in Fig. 4. If the shortest path is found between A and C and if B is a node on this path, it may happen that the shortest path between A and B is not a part of the shortest path between A and C. Generally, there is a risk that the problems that do not obey Bellman's principle cannot be solved effectively. Firstly, we present the observations that make the computation feasible. Then the algorithm is presented. Say that the distance $d_{km}(A, C)$ between the nodes A and C is to be measured. Since two paths, denoted by α and β, respectively, exist between A and B, the distance between A and B is $d_{km}(A, B) = \min\{l(\alpha_{AB}), l(\beta_{AB})\}$. To determine the path lengths $l(\alpha_{AB})$ and $l(\beta_{AB})$, two vectors of k maximum weights along the particular paths are considered: $\vec{a}_B = (a_1, a_2, \ldots, a_k)$, $\vec{b}_B = (b_1, b_2, \ldots, b_k)$. According to the definition $l(\alpha_{AB}) = \sum_{i=1}^{k} a_i$, and $l(\beta_{AB}) = \sum_{i=1}^{k} b_i$. Let us now explore the path from B to C. Let $c_1, c_2, \ldots$ be the maximum weights along this path. We suppose that the entries of the vectors are always ordered such that $a_i \geq a_{i+1}$, $b_i \geq b_{i+1}$, $c_i \geq c_{i+1}$ in our notation. If r values are retained ($k - r$ values are replaced) in $\vec{a}_B$ during the way from B to C, the vector $\vec{a}_B$ is modified to the vector $\vec{a}_C = (a_1, a_2, \ldots, a_r, c_1, c_2, \ldots, c_{k-r})$, which happens if $c_1 > a_{r+1}$, $c_2 > a_{r+1}$, $\ldots$, $c_{k-r} > a_{r+1}$. Along the same path, s values, $0 \leq s \leq k$, can be similarly retained in $\vec{b}_B$.

Theorem 1. *Consider the situation described in the previous paragraph. If r values are retained in $\vec{a}_C$ from $\vec{a}_B$ (i.e., $k - r$ values are replaced) and if $\sum_{i=1}^{r} a_i < \sum_{i=1}^{r} b_i$, then $\vec{a}_C$ (not $\vec{b}_C$) will determine the distance $d_{km}(A, C)$.*

Proof. Let s be the number of values that are retained when $\vec{b}_B$ changes into $\vec{b}_C$. Three cases should be distinguished: $s = r$, $s > r$, and $s < r$. For brevity, only the latter case is presented. The remaining cases can be proven in a similar way.

Say that $s < r$. We have $\vec{a}_C = (a_1, \ldots, a_r, c_1, \ldots, c_{k-r})$, and

$$l(\alpha_{AC}) = \sum_{i=1}^{r} a_i + \sum_{i=1}^{k-r} c_i \,. \tag{2}$$

The vector $\vec{b}_C$ can be written in the form of $\vec{b}_C = (b_1, \ldots, b_s, c_1, \ldots, c_{k-r}, c_{k-r+1}, \ldots, c_{k-s})$. Considering the fact that $c_{k-r+1} > b_{s+1}, \ldots, c_{k-s} > b_{s+1}$ (since b_{s+1} was completely removed from $\vec{b}_C$ during updating $\vec{b}_B$ to $\vec{b}_C$), and consequently $c_{k-r+1} > b_{s+1}, \ldots, c_{k-s} > b_r$ (since $b_i \geq b_{i+1}$), we have

$$l(\beta_{AC}) = \sum_{i=1}^{s} b_i + \sum_{i=1}^{k-r} c_i \; + c_{k-r+1} \ldots + c_{k-s} \geq \sum_{i=1}^{r} b_i + \sum_{i=1}^{k-r} c_i \,. \tag{3}$$

Remembering the assumption $\sum_{i=1}^{r} a_i < \sum_{i=1}^{r} b_i$, we conclude that $l(\alpha_{AC}) < l(\beta_{AC})$. □

Since Bellman's optimality principle does not hold for the k-max distance, a list of vectors (each containing the k maximum weights, and each corresponding to a certain path to the node) must be stored in each node. Although only one of them determines the distance to the node, the remaining vectors may be needed for determining the distances to other nodes. It is threatening that the total number of vectors that should be stored during the computation may increase above the acceptable limits. The theorem shows that the number of vectors can be reduced.

In the algorithm, the vectors are examined in pairs to decide whether it is necessary to store both of them for further computation. We let r run through all possible values $1 \leq r \leq k$. If $\sum_{i=1}^{r} a_i < \sum_{i=1}^{r} b_i$, we say that $\vec{a}$ is better for that particular value of r. If $\sum_{i=1}^{r} b_i < \sum_{i=1}^{r} a_i$, then $\vec{b}$ is better. If $\vec{b}$ is never better (i.e., is not better for any value of r), then only $\vec{a}$ is stored. We say that $\vec{b}$ is *overshadowed* by $\vec{a}$. (If $\vec{a}$ is never better, we proceed analogously.) If sometimes (for some values of r) $\vec{a}$ is better, and sometimes $\vec{b}$ is better, both the vectors are stored. The algorithm can now be formulated as follows.

Algorithm The k-Max Distance in Graph

Input: The graph and its node S from which the distances are to be computed.

Output: The k-max distances to all nodes in the graph. The shortest path between S and each node if required.

1. For each node, set its actual distance to a big value (infinity), and create an empty list of its attached max vectors.
2. Create an empty priority queue that will be used for organising the vectors according to the distance.
3. Set the actual distance $d_{km}(S, S) = 0$, insert the max vector $(0, 0, \ldots, 0)$ into the list of the vectors attached to S, and insert the vector into the queue (only the pointers are stored in the queue).
4. While the queue is not empty do
5. Read and remove from the queue the vector, denoted by $\vec{m}_T$, whose sum of entries is minimal. (The subscript T indicates that the vector is attached to the node T, i.e., it is present in the list of its max vectors.) If the candidate distance value determined for T before is greater than the sum, replace it by the value of the sum, and remember the pointer to the vector since it may become the first vector for backtracking the path from T to S.

6. For each neighbour of T (including the neighbours for which the distance has already been determined), do the following (let U be such a neighbour, let $w_{T,U}$ be the weight of the arc connecting T and U, and let $\mathcal{M}_U$ be the list of the vectors attached to U): Update $\vec{m}_T$ with $w_{T,U}$ into $\vec{m}_{T,U}$. (If the minimum value in $\vec{m}_T$ is less than $w_{T,U}$, replace it by $w_{T,U}$. Otherwise, $\vec{m}_{T,U} = \vec{m}_T$). Check the updated vector with all vectors in $\mathcal{M}_U$. If no vector overshadowing $\vec{m}_{T,U}$ exists in $\mathcal{M}_U$, insert $\vec{m}_{T,U}$ into $\mathcal{M}_U$, insert $\vec{m}_{T,U}$ into the queue, remove from $\mathcal{M}_U$ all the vectors that are overshadowed by $\vec{m}_{T,U}$, remove them also from the queue, establish the backtracking pointer from $\vec{m}_{T,U}$ to $\vec{m}_T$. If a vector overshadowing $\vec{m}_{T,U}$ exist in $\mathcal{M}_U$, do nothing.
7. Output the distances of all nodes. The backtracking pointers attached to the max vectors can be used for outputting the shortest paths.

3.2. Time complexity

In this subsection, an outline of the time complexity analysis is presented. Let V be the number of nodes, and let D be the maximum degree of node. We recall that k is the number of maximum values determining the distance. For now, say that the number of vectors that are attached to one node simultaneously at a certain moment during computation is not greater than M. Therefore, no more than VM items are present in the queue simultaneously. Say that no more than R vectors are inserted into the queue in total (since the vectors can also be deleted from the lists attached to the nodes, R may be greater than M). Also, the minimum vector is read no more than R-times. For each minimum vector, no more than D node neighbours are checked/updated, which requires $O(kDM)$ time for the test of overshadowing and updating the lists of the vectors attached to the neighbouring nodes. For inserting/deleting the vectors into/from the queue (only the pointers are inserted), $O(R\log(VM))$ time is required. It follows that the overall time complexity is $O(R(kDM + \log(VM)))$. Since V, D, and k are a priori known constant values, the efficiency of algorithm depends on the values of M and R.

It is obviously difficult to explore the values of R and M theoretically, Therefore, an experiment has been used to illustrate possible practical values. 10^5 images of size 101×101 pixels have been generated containing Gaussian noise ($\sigma_n = 0.333$). The arc weights have been computed using Eq. (1) ($\sigma_w = 0.333$, $\eta = 0.01$). In each image, the k-max distances from the centre of the image to all image points have been determined. The number of inserts into the queue, denoted by R, and the number of vectors attached to the nodes simultaneously have been examined. Namely, the average maximum number of the vectors attached to one node (i.e., the average over the maximum numbers in particular nodes that were achieved during computation), denoted by $\hat{M}$, and the absolute maximum number of vectors (i.e., the maximum over the maxima in the particular nodes), denoted by M. The distance between the image centre point and the point in the top-left corner has been checked too. The mean values together with the standard deviations of the mentioned quantities are shown in Table 1. The practical mean running time on one core of an Intel Pentium 3.4 GHz computer is also shown. It can be seen that the values of R, $\hat{M}$, and M increase with k. In spite of this fact, Theorem 1 can be regarded as useful. The computation would not be feasible without it.

3.3. Further optimization

To further reduce the number of the vectors that must be stored, we introduce a criterion that makes it possible to remove the vectors with a low probability of being important for determining the distance. However, it is achieved at the expense that only approximate distances are computed.

Table 1
The distance (see the text), the mean of the maximum numbers (during one computation) of the vectors that were present in the list attached to one node ($\hat{M}$), the absolute maximum number of vectors attached to one node (M), the number of the inserts into the queue in the form of the ratio R/V, and the real running time in ms. The values were computed from 10^5 samples; the mean values with standard deviations (the values in the brackets) are presented.

k	dist	$\hat{M}$	M	R/V	Time
5	2.04 (0.32)	1.90 (0.52)	8.29 (2.03)	1.99 (0.56)	17
10	3.56 (0.40)	4.13 (1.44)	24.36 (8.09)	4.45 (1.54)	55
15	4.87 (0.47)	7.79 (2.99)	52.67 (20.15)	8.38 (3.17)	170
20	6.01 (0.55)	13.15 (5.28)	96.25 (40.02)	14.08 (5.55)	523

Theorem 2. *Consider the situation as in Theorem 1. Say that the probability distribution of the arc weights in the graph is known and is described by its cumulative distribution function, denoted by $F(.)$. The particular arc weights are supposed to be independent. The probability, denoted by π_r, of the event that, during the move along a path containing L arcs, the vector $\vec{a} = (a_1, \ldots, a_r, a_{r+1}, \ldots, a_k)$ will be updated in such a way that just r entries will be retained in $\vec{a}$ (i.e., just $k - r$ entries will be replaced) can be expressed as (we introduce $k - r \equiv q$ for brevity)*

$$\pi_r = \sum_{i=0}^{q} \binom{L}{i} (1 - F(a_r))^i F(a_r)^{L-i} - \sum_{i=0}^{q-1} \binom{L}{i} (1 - F(a_{r+1}))^i F(a_{r+1})^{L-i}. \tag{4}$$

Proof. Consider two situations: (1) Less than q weights on the path will be greater than a_{r+1}. Therefore, less than q entries in $\vec{a}$ are replaced (more than r are retained). (2) More than q weights will be greater than a_r. Therefore, more than q entries are replaced (less than r are retained). The probabilities of the mentioned events are denoted by $\pi_{>r}$ and $\pi_{<r}$, respectively. The probability that just q entries are replaced (r entries are retained) is $\pi_r = 1 - \pi_{>r} - \pi_{<r}$. The first case occurs if $0, 1, 2, \ldots, q-1$ weights on the path are greater than a_{r+1}. Therefore, we have

$$\pi_{>r} = \sum_{i=0}^{q-1} \binom{L}{i} (1 - F(a_{r+1}))^i F(a_{r+1})^{L-i}. \tag{5}$$

The second case occurs if $q+1, \ldots, L$ weights on the path are greater than a_r. It follows that

$$\pi_{<r} = \sum_{i=q+1}^{L} \binom{L}{i} (1 - F(a_r))^i F(a_r)^{L-i} = 1 - \sum_{i=0}^{q} \binom{L}{i} (1 - F(a_r))^i F(a_r)^{L-i}. \tag{6}$$

The second formula can be derived if we realise that the items in the sum in the first formula have the form that is known from the binomial theorem. In this case, $((1 - F(a_r)) + F(a_r))^L = 1^L = 1$. Therefore, the terms in the sum in the second formula are the terms that are not present in the first formula. The formula in the theorem can now be obtained as $\pi_r = 1 - \pi_{>r} - \pi_{<r}$. □

The method for determining whether or not a vector should be stored (Section 3.1) can now be modified. The condition from Theorem 1 tells whether, for a given value of r, a vector may be needed for determining the distance in the future. We strive to determine the probability that it will happen. Namely, we determine the probability of the event that just r entries are retained in the vector during the future updates. If this probability is lower

Table 2
On the behaviour of the method using the probability threshold 0.3. The meaning of the symbols is as in Table 1. $F(.)$ was determined as the distribution corresponding to the Gaussian noise of pixel intensities (Eq. (1), $\sigma_n = \sigma_w = 0.333$, $\eta = 0.01$).

k	dist	$\hat{M}$	M	R/V	Time
5	2.05 (0.32)	1.73 (0.44)	7.41 (1.70)	1.81 (0.48)	17
10	3.57 (0.40)	2.58 (0.71)	14.44 (3.73)	2.77 (0.77)	36
15	4.89 (0.47)	3.44 (1.01)	21.60 (6.12)	3.72 (1.08)	82
20	6.03 (0.55)	4.52 (1.39)	29.73 (8.98)	4.89 (1.48)	201

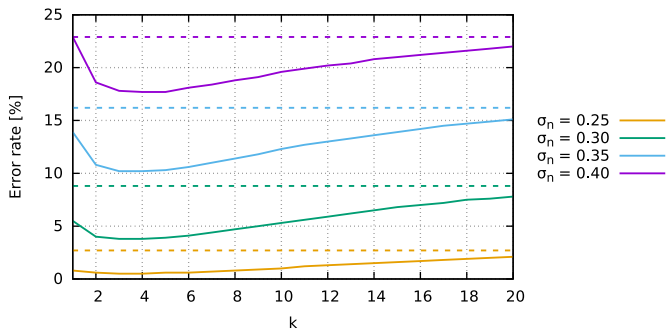

Fig. 5. The error rate of the k-max distance for the test case from Fig. 1, $a = 20$, $da = 0$, $k = 1, \ldots, 20$. Various noise levels (σ_n) are considered. The dashed lines show the error rate for the geodesic distance with the same settings.

than a chosen threshold, we can conclude that the vector is not important for that particular value of r. The discussion following Theorem 1 can now be extended. All values of r, $1 \le r \le k$, are explored. Let n_a be the number specifying for how many values of r, $\vec{a}$ can determine the distance (Theorem 1). Similarly, n_b specifies for how many values of r, $\vec{b}$ can determine the distance (or shortly, is better). If $n_b > 0$ and if n_a is substantially greater than n_b, $\vec{b}$ is checked whether, for the value of r for which $\vec{b}$ is better, the probability is big enough of the event that just r entries are retained in $\vec{b}$ in the future. If the probability is low for all r for which $\vec{b}$ is better, $\vec{b}$ is not stored. Various chosen values of L are used to estimate the maximum possible probability. The function $F(.)$ must be known or estimated.

An example of how the introduced probabilities can influence the numbers of vectors in nodes is presented in Table 2. The test environment was the same as described in Section 3.2. The values $L \in \{q, 2q, 3q\}$ with the step $\max\{1, q/2\}$ were used to determine the maximum probability for each checked value of r. As can be seen, the distances remain almost unchanged while the numbers of vectors are reduced (compare with the results in Table 1). Although computing the probabilities takes some additional time, the overall computation time is reduced if the probability threshold is sufficiently high.

4. Experiments

In this section, the performance of the k-max distance is shown in comparison with the geodesic distance and the max-arc distance (which is equivalent to the 1-max distance). In the experiments on real-life images, the mentioned distance-based methods are also compared with the minimum barrier distance [14], and with the random walker segmentation algorithm [22]. In all experiments, the parameters from Eq. (1) are set to $\sigma_w = 0.333$ and $\eta = 0.01$ unless otherwise noted.

Firstly, we show the behaviour of the k-max distance on the same synthetic image that was used for illustrating the behaviour of the geodesic distance in Section 2 (Fig. 1). We measure the k-max distances $d_{km}(A, B)$ and $d_{km}(B, C)$. The measurement is erroneous if $d_{km}(A, B) \le d_{km}(B, C)$. Fig. 5 shows the error rate of the

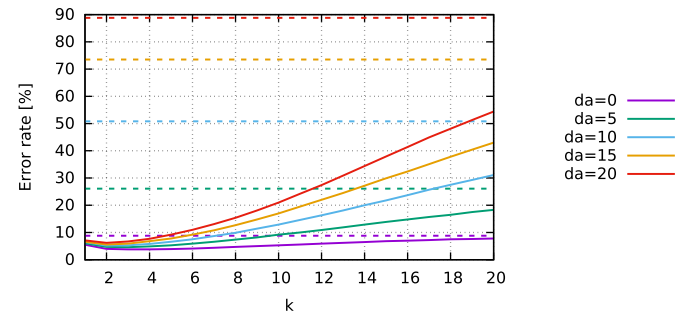

Fig. 6. The error rate of the k-max distance for the test case from Fig. 1, $a = 20$, $\sigma = 0.3$, $k = 1, \ldots, 20$. Various values of da are considered. The dashed lines show the error rate for the geodesic distance with the same settings.

k-max distance for $k = 1, \ldots, 20$, and for various levels of noise. The results show that the error rate of the k-max distance is lower than it is in the case of the geodesic distance. The error rate is similar only for $k = 1$ with $\sigma_n = 0.4$. In this case, the high level of noise causes that the weights of the important arcs that cross the segment boundary are frequently overshadowed by the noisy weights of the unimportant arcs lying inside the segments. This is also the reason why the best results are not achieved for $k = 1$ (max-arc distance), but a greater value of k is required although the image contains only one boundary line between the segments. Fig. 6 shows how the error rate depends on the value of k for various values of da ($a = 20$, $\sigma_n = 0.3$). It can be seen that, beginning from a certain value of k, the error rate increases with k, which is because more unwanted arc weights (affected by noise) are included into the max vector. If many such weights are included, the probability increases that they will decide the final distance. For $k \to \infty$, the error rate of the k-max distance approaches (from below) to the error rate of the geodesic distance, which is expected since the k-max distance becomes the geodesic distance for $k \to \infty$.

4.1. Exploring the shortest paths in images

In this experiment, the shortest path between two chosen image points is explored. A noisy synthetic image with a spiral, a map image, and a fundus photography are used (Fig. 7). The shortest paths found by the geodesic distance, the max-arc distance, and the k-max distance for $k \in \{5, 10, 20\}$ are shown. The spiral image contains two image segments: the spiral, and the background. Since the chosen points are both situated in the background (Fig. 7, column 1, row 1), we expect that the whole shortest path between them should run in the background too. It can be seen that the geodesic distance ignores the spiral shape since the expected shortest path is too expensive due to summing the noisy weights of many arcs. For the max-arc, 5-max, and 10-max distance, the shortest path correctly follows the shape of spiral. For the 20-max distance, the number of noisy weights summed in the distance is too high, similarly as in the case of the geodesic distance. In the map image (Fig. 7, row 2), the goal is to find the shortest path connecting two points located on the sea near the coast. We expect that the shortest path should run on the sea along the coast. This is achieved by the k-max distance for $k \in \{5, 10, 20\}$. Similarly, in the fundus photography (Fig. 7, row 3), the points are placed on a retinal blood vessel. We expect that the path should follow the vessel, which is achieved by the 5-max and 10-max distance.

4.2. Image segmentation

Since the distance measurement is important for image segmentation, the tests are carried out in this area. The seeded segmentation has been selected. It uses a priori provided seeds scribbled into the particular segments. The distance of every image

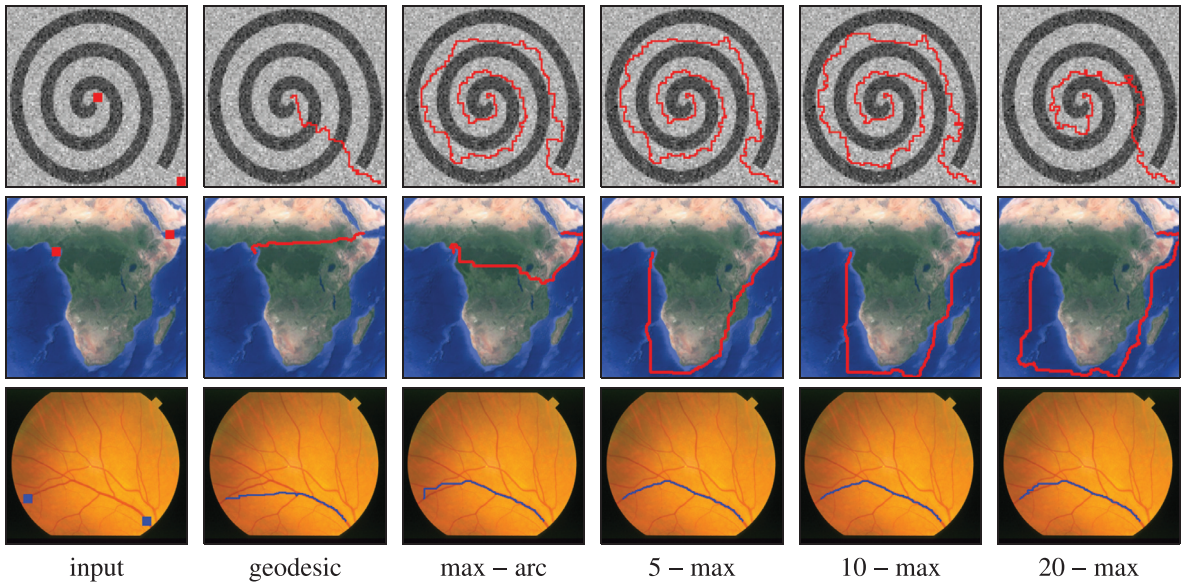

Fig. 7. The shortest paths in images. The input images with the end points of the path are shown in the first column. The shortest paths are shown for the geodesic distance (column 2), the max-arc distance (column 3), and for the k-max distance for $k = 5$ (column 4), $k = 10$ (column 5), $k = 20$ (column 6).

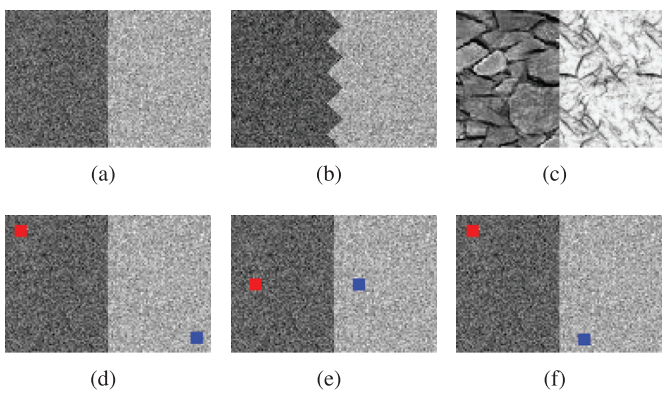

Fig. 8. Synthetic images to be segmented: (a, b) with a unit intensity step with Gaussian noise, (c) with textures, (d, e, f) with various positions of seeds.

Table 3
The comparison of the segmentation results of the synthetic images (Fig. 8). The percentage of incorrectly labelled pixels is presented in the form of the mean values and standard deviations (in the brackets). The minimum errors are depicted in bold.

	Fig. 8a	Fig. 8b	Fig. 8c
Geodesic	7.29 (5.05)	7.38 (4.90)	8.17 (7.28)
Max-arc	4.45 (11.16)	4.68 (10.57)	6.25 (14.77)
5-max	2.78 (6.83)	**2.69** (6.39)	3.69 (5.76)
10-max	**2.75** (4.75)	2.76 (4.39)	**3.57** (4.63)
	Fig. 8d	Fig. 8e	Fig. 8f
Geodesic	5.86 (2.20)	8.73 (2.86)	14.55 (3.06)
Max-arc	8.37 (16.42)	4.05 (10.43)	7.32 (15.32)
5-max	5.09 (10.95)	2.58 (6.43)	5.12 (10.70)
10-max	**3.41** (7.18)	**2.56** (4.42)	**5.08** (7.96)

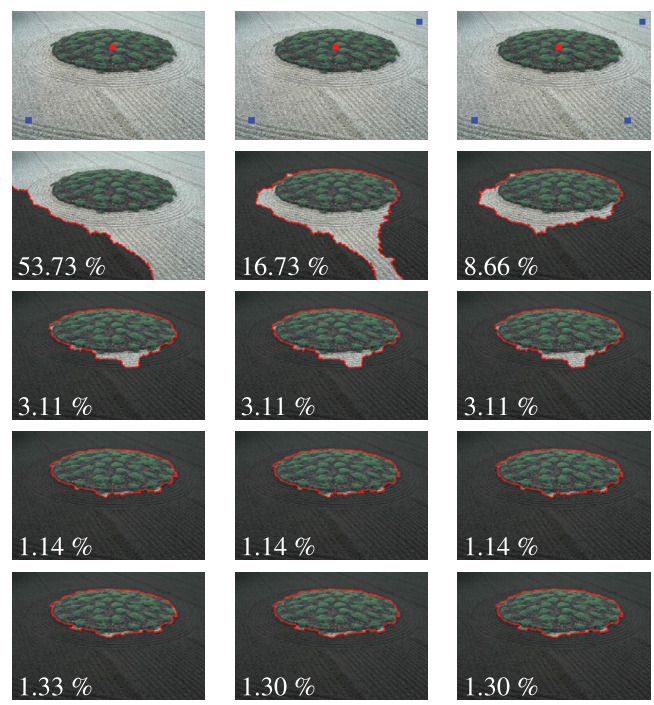

Fig. 9. On the sensitivity to the number and position of seeds. The input image with the seeds (row 1), the segmentation results of the geodesic distance (row 2), the max-arc distance (row 3), the 5-max distance (row 4), and 10-max distance (row 5) are shown. The values in the images show the percentage of incorrectly labelled pixels.

point is measured to the seeds. The point is assigned to a segment if the distance to its seed is shorter than the distance to the seeds of the remaining segments.

In the first test, a synthetic greyscale image with two segments separated by a boundary in the image centre is used. Three versions of the image are considered: a unit intensity step with a line boundary (Fig. 8a), with a more complex boundary (Fig. 8b) (both with added Gaussian noise, $\sigma_n = 0.3$), and with textures in each segment (Fig. 8c). The segmentation error is evaluated as a relative number of incorrectly labelled pixels, which is computed as the mean (together with the standard deviation) from 10^5 samples. In each sample, the seeds (3×3 pixels) are placed randomly within the segment area (one seed in each segment). We also explore how the positions of seeds influence the segmentation results. For this purpose, Fig. 8a is used ($\sigma_n = 0.3$) with the seeds (3×3 pixels) placed on various positions (Fig. 8d–8f). The segmentation error is evaluated from 10^5 samples again. Table 3 shows that the k-max distance performs better than the geodesic distance and the max-arc distance in all the mentioned test cases.

Fig. 10. The images from the Berkeley dataset [23] used in the experiments. The object (red) and background (blue) seeds are depicted.

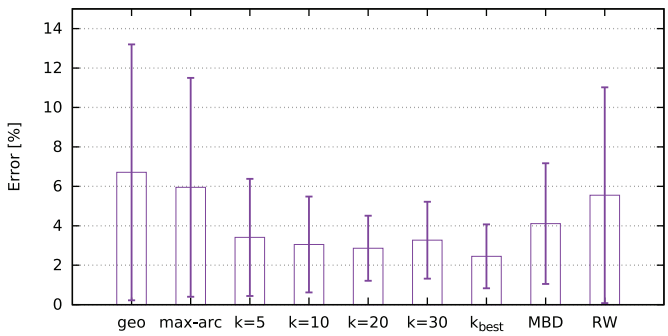

Fig. 11. The mean segmentation errors (with standard deviations) of the methods tested on real-life images shown in Fig. 10. The results of the geodesic distance (geo), the max-arc distance, the random walker (RW), the minimum barrier distance (MBD), and the k-max distance for various values of k are shown; k_{best} is the result obtained in such a way that the best segmentation was found for each image for $k = 1, \ldots, 30$.

We also examine the dependence on the position and the number of seeds in a real-life image (Fig. 9). We suppose that the image contains an object with a background (the image and the ground truth are from the Berkeley dataset [23]). For the object, one seed is used. For the background, one, two, and three seeds, respectively, are used. We examine how many seeds are necessary to obtain satisfactory segmentation results. It can be regarded as a good property of the distance measure if the number and the position of seeds are not important. Fig. 9 shows that the geodesic distance (row 2) requires more seeds in the background. The results of the max-arc (row 3) and k-max distance (row 4 and 5) are similar.

Finally, we compare the segmentations based on the k-max distance, the geodesic distance, the max-arc distance, the minimum barrier distance, and the random walker technique on several real-life images from the Berkeley dataset (Fig. 10). Fig. 11 shows the mean segmentation errors (with the standard deviations) of the particular methods (the ground truth for each image is also from the dataset). The value of σ_w, which is used for computing the arc weights, is chosen individually for each image such that it gives the best possible segmentation result.

5. Conclusion

We have presented a new distance called the k-max distance which can be used in the graphs and images. The length of a path is defined as the sum of the k maximum arc weights on the path. The distance is the length of the shortest path in this sense. We proved that the k-max distance is a metric. Since it does not obey Bellman's principle of optimality, the time and memory complexity is generally critical. We presented the observations and the method making the computation of the k-max distance feasible. The properties of the k-max distance were tested and its comparison with some other distance measures was carried out. The experiments have shown that the k-max distance performs generally better than the geodesic distance. If used in image segmentation, for example, it is less sensitive to the position and to the number of seeds than the geodesic distance. We also achieved better results than the minimum barrier distance, and the random walker technique in the segmentation of real-life images.

Acknowledgments

This work was partially supported by the Grants of SGS Nos. SP2016/58, and SP2017/61, VŠB - Technical University of Ostrava, Czechia.

References

[1] G. Borgefors, Distance transformations in arbitrary dimensions, Comput. Vis. Gr. Image Process. 27 (3) (1984) 321–345.
[2] A. Hajdu, J. Kormos, B. Nagy, Z. Zorgo, Choosing appropriate distance measurement in digital image segmentation, Annales Univ. Sci. Budapest Sect. Comp 24 (2004) 193–208.
[3] I.-M. Sintorn, G. Borgefors, Weighted distance transforms for volume images digitized in elongated voxel grids, Pattern Recognit. Lett. 25 (5) (2004) 571–580. Discrete Geometry for Computer Imagery (DGCI'2002).
[4] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1) (1959) 269–271.
[5] P.J. Toivanen, New geodesic distance transforms for gray-scale images, Pattern Recognit. Lett. 17 (5) (1996) 437–450.
[6] A. Protiere, G. Sapiro, Interactive image segmentation via adaptive weighted distances, IEEE Trans. Image Process. 16 (4) (2007) 1046–1057.
[7] X. Bai, G. Sapiro, Geodesic matting: a framework for fast interactive image and video segmentation and matting, Int. J. Comput. Vis. 82 (2) (2009) 113–132.
[8] A. Criminisi, T. Sharp, A. Blake, GeoS: geodesic image segmentation, in: Proceedings of the Tenth European Conference on Computer Vision: Part I, in: ECCV '08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 99–112.
[9] B.L. Price, B.S. Morse, S. Cohen, Geodesic graph cut for interactive image segmentation., in: Proceedings of the Computer Vision and Pattern Recognition, IEEE, 2010, pp. 3161–3168.
[10] L. Zhou, Y. Qiao, J. Yang, X. He, Learning geodesic CRF model for image segmentation, in: Proceedings of the Nineteenth IEEE International Conference on Image Processing, 2012, pp. 1565–1568.
[11] L. Ikonen, P.J. Toivanen, Shortest routes on varying height surfaces using gray-level distance transforms., Image Vis. Comput. 23 (2) (2005) 133–141.
[12] R. Cárdenes, C. Alberola-López, J. Ruiz-Alzola, Fast and accurate geodesic distance transform by ordered propagation, Image Vis. Comput. 28 (3) (2010) 307–316.
[13] R. Strand, K.C. Ciesielski, F. Malmberg, P.K. Saha, The minimum barrier distance, Comput. Vis. Image Underst. 117 (4) (2013) 429–437. Special issue on Discrete Geometry for Computer Imagery.
[14] K.C. Ciesielski, R. Strand, F. Malmberg, P.K. Saha, Efficient algorithm for finding the exact minimum barrier distance, Comput. Vis. Image Underst. 123 (2014) 53–64.
[15] R. Strand, F. Malmberg, P.K. Saha, E. Linner, Discrete geometry for computer imagery, Springer International Publishing, Cham, 2014, pp. 111–121. September 10–12.
[16] J. Gaura, E. Sojka, Resistance-geodesic distance and its use in image segmentation, Int. J. Artif. Intell. Tools 25 (05) (2016) 1640002–1–164002–16.
[17] W.G. Kropatsch, A. Ion, Y. Haxhimusa, T. Flanitzer, The Eccentricity Transform (of a Digital Shape), Springer, Berlin, Heidelberg, pp. 437–448.
[18] M. Holusa, E. Sojka, A k-max Geodesic Distance and Its Application in Image Segmentation, Springer International Publishing, Cham, pp. 618–629.
[19] A.X. Falcao, J. Stolfi, R. de Alencar Lotufo, The image foresting transform: theory, algorithms, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 26 (1) (2004) 19–29.
[20] L.A.C. Mansilla, P.A.V.V. Miranda, F.A.M. Cappabianco, Image segmentation by image foresting transform with non-smooth connectivity functions, in: Proceedings of the Twenty-Sixth Conference on Graphics, Patterns and Images, 2013, pp. 147–154.
[21] R. Bellman, On a routing problem, Q. Appl. Math. 16 (1958) 87–90.
[22] L. Grady, Random walks for image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 28 (11) (2006) 1768–1783.
[23] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, in: Proceedings of the Eight International Conference on Computer Vision, 2, 2001, pp. 416–423.