Fast Distance Transforms in Graphs and in Gmaps

Majid Banaeyan(✉), Carmine Carratù, Walter G. Kropatsch, and Jiří Hladůvka

Pattern Recognition and Image Processing Group, TU Wien, Vienna, Austria
{majid,krw,jiri}@prip.tuwien.ac.at

Abstract. Distance Transform (DT) as a fundamental operation in pattern recognition computes how far inside a shape a point is located. In this paper, at first a novel method is proposed to compute the DT in a graph. By using the edge classification and a total order [1], the spanning forest of the foreground is created where distances are propagated through it. Second, in contrast to common linear DT methods, by exploiting the hierarchical structure of the irregular pyramid, the geodesic DT (GDT) is calculated with parallel logarithmic complexity. Third, we introduce the DT in the nD generalized map (n-Gmap) leading to a more precise and smoother DT. Forth, in the n-Gmap we define n different distances and the relation between these distances. Finally, we sketch how the newly introduced concepts can be used to simulate gas propagation in 2D sections of plant leaves.

Keywords: nD distance transform · Generalized maps · Irregular pyramids · Parallel processing · Logarithmic complexity · Geodesic distance transform (GDT)

1 Introduction

The distance transform [5] computes for every pixel/voxel of an image/object how far it is from the closest obstacle, boundary, or background. While any valid metric may be involved in the computation of distance transforms, in topological data structures like graph, combinatorial maps [12], or generalized maps (n-Gmap) [7] often the shortest path between the obstacle/boundary and a given point is used. In this study, we first investigate the distance transform (DT) in graphs and then extend it to generalized map. We define different distances for every dimension (1D, 2D,..., nD) in the n-Gmap. This would be useful in many applications. In particular, in study of gas exchange through airspace of a leaf, computing the distances from stomata is very crucial to understand the different diffusion processes needed for photosynthesis.

Supported by the Vienna Science and Technology Fund (WWTF), project LS19-013, https://waters-gateway.boku.ac.at/.

A. Krzyzak et al. (Eds.): S+SSPR 2022, LNCS 13813, pp. 193–202, 2022.
https://doi.org/10.1007/978-3-031-23028-8_20

Computing the DT and propagating the distances is an iterated local operation [8,15]. While local processes (e.g., convolution and mathematical morphology) are important in early vision, they are not suitable for higher level vision, such as symbolic manipulation and feature extraction where both local and global information is needed [9]. Therefore we exploit the advantage of the hierarchical structure of the pyramid [10] that encodes both local and global information similar to the human visual system [14].

In the pyramid there are two directions of processes: bottom-up and top-down. In the bottom-up (fine to coarse) process the information of the input data (e.g. intensity, color, texture) is transformed into global information. In the top-down (coarse to fine) process the global information such as the shape and the size of objects are refined into the base level of the pyramid. Therefore, the main idea of using hierarchical structure in computing the DT is to investigate the connectivity of a connected component in the local and general view within the pyramid. We will show that the connectivity can be checked in parallel logarithmic complexity instead of the linear raster scan commonly utilized in the state-of-the-art algorithms [8].

1.1 Notations and Definitions

An image P can be represented using a 4-adjacent neighborhood graph $G = (V, E)$ where V corresponds to pixels of P and E relates neighboring pixels. 8-Adjacency could be used only if the image is well-formed [11], which is not satisfied in general cases. The gray-value of a pixel $g(p)$ becomes an attribute of the corresponding vertex v, $g(v) = g(p)$ and the $contrast(e) = |g(u) - g(v)|$ becomes an attribute of an edge $e(u, v)$ where $u, v \in V$. In the neighborhood graph of the binary image, the edges have only two values: zero and one. We call them accordingly: **zero-edge** and **one-edge** [1]. Furthermore we denote the set of all zero-edges as E_0 and the set of all one-edges as E_1. In this way, the edges of the graph are partitioned into $E = E_0 \cup E_1$.

Irregular Pyramid. [10] is a stack of successively reduced smaller graphs where each graph is built from the graph below by selecting a specific subset of vertices and edges. In each level of the pyramid, the vertices and edges disappearing in level above are called *non-surviving* and those appearing in the upper level *surviving* ones.

Definition 1 (Contraction Kernel (CK)). *A CK is a tree consisting of a surviving vertex as its root and some non-surviving neighbors with the constraint that every non-survivor can be part of only one CK.*

Two basic operations are used to construct the pyramid: **edge contraction** and **edge removal**. In the edge contraction, an edge $e = (v, w)$ is contracted while its two endpoints, v and w, are identified and the edge is removed. The edges that were incident to the joined vertices will be incident to the resulting vertex after the operation. An arrow over an edge is commonly used to indicate the direction of contraction, i.e., from non-survivor to survivor (cf. Fig. 2). Contracting an edge has the enormous advantage of preserving the connectivity of the graph.

During the edge removal, an edge is removed without changing the number of vertices or affecting the incidence relationships of other edges. Constraints are needed to make sure that edge removal does not disconnect the graph [6].

2 Distance Transform in a Graph

In a graph $G = (V, E)$ distances can be measured by the shortest length of paths. In this case the elements are the vertices V and neighbors $\mathcal{N}(v) = \{(v, w) \in E\}$ are related by edges. The distance between two vertices is the shortest path connecting the two vertices.

To compute the DT in a graph $G(V, E)$ with background $B \subset V$ and foreground $F \subset V$ vertices, the shortest distances of foreground vertices from the background should be computed. In this case the *seed* vertices $b \in B$ are initialized by $DT(b) = 0$. The foreground vertices $f \in F$ are initialized by $DT(f) = \infty$. Each one-edge $e = (b, f) \in E_1, b \in B, f \in F$ has two endpoints where $b \in B$ is a seed vertex with $DT(b) = 0$. The other vertex $f \in F$ belongs to the foreground and we initialize its distance by $DT(f) = 1$. The one-edges E_1 are frozen because they have no role in propagating the distances in the graph. Distances are propagated only through the E_0 edges of the foreground.

Using the total order on the foreground F proposed in [1,3], a spanning forest contains only edges E_0 spanning the foreground. The spanning forest is created in a single step with parallel constant complexity. Moreover, to propagate the distances we use the breath-first search (BFS) [4].

Proposition 1. *The parallel complexity of propagating DT is $\mathcal{O}(\delta(T))$ where $\delta(T)$ is the longest path in the spanning forest of the foreground.*

Proof. The complexity of propagating distances in a tree is $\mathcal{O}(|E|)$. Each connected component of the foreground is covered by a spanning tree which is processed independently [3]. Therefore, the longest path in the forest indicates the parallel complexity. □

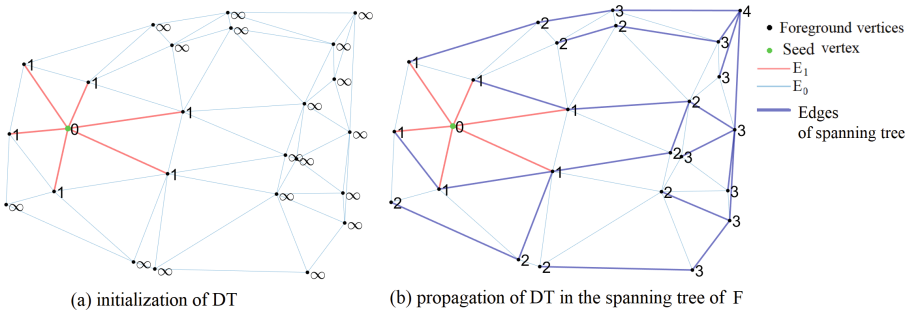
The propagation of the distances to the remaining vertices v of the foreground F follows:

$$D(v) = \min\{D(v), D(v_j) + 1 |\ v_j \in \mathcal{N}(v)\} \quad v \in F \tag{1}$$

where the foreground neighbors $\mathcal{N}(v)$ are defined by:

$$\mathcal{N}(v) = \{v \in F\} \cup \{w \in F | e_0 = (v, w) \in E_0\} \tag{2}$$

The distances are propagated until there is no vertex $v \in F$ with $DT(v) = \infty$ (see Fig. 1). Algorithm 1 shows the steps of computing the DT in a graph.



(a) initialization of DT (b) propagation of DT in the spanning tree of F

Fig. 1. Computing the DT in a graph

Algorithm 1. Computing the DT in the Neighborhood Graph

1: **Input:** Neighborhood Graph: $G = (V, E) = (B \cup F, E_0 \cup E_1)$
2: Initialization: $DT(b) = 0 \; \forall b \in B, \quad DT(f) = \infty, \; \forall f \in F$
3: $DT(f) = 1 \; \forall (f, b) \in E_1, \quad f \in F, \quad b \in B$
4: **While** $\exists f \in F$ with $DT(f) = \infty$ **do**
5: Propagate the distances by (1)
6: **end**

2.1 Geodesic Distance Transform

Geodesic DT (GDT) computes distances within the connected component of interest in a labeled image (or labeled neighborhood graph). The objects of interest are considered as the foreground objects and the remaining objects with different labels are considered as the background. A subset of points in the foreground are the seeds, $s \in S$, $S \subset F$, initialized by zero, $DT(s) = 0$. The aim is to compute the minimum distance of every point of the foreground to these seeds. The disjoint foreground objects keep the infinite distance if there is no seed in the connected component.

To compute the GDT we employ the irregular graph pyramid with logarithmic complexity. Each vertex receives a unique index and a total order is defined over the indices [1,3] that results in an efficient selection of contraction kernels (CKs). The CKs are only selected from E_0 edges which propagate the distances. The propagating distances are a set of power-of-two numbers. In Fig. 2 edges of CKs are shown by an arrow pointing towards the surviving vertex. The propagating distance i is shown by $\boxed{\text{i}}$ over an edge. By default all edges propagate distances by 1. Each surviving edge propagates the distance equal to 2^i into its adjacent unlabeled vertex. Next, to speed up the propagation of the distances with a power of two, the independent edges of a CK are identified by employing a logarithmic encoding. This logarithmic encoding indicates the priority of contractions through the construction of the pyramid. In Fig. 2a the numbers $1, 2, 3$ and $1', 2', 3'$ indicate the primary priorities that are different for each adjacent edge. The bottom-up construction of the pyramid (Fig. 2(a) to (d)) terminates when there is no edge remaining for the contraction. In top of

the pyramid all surviving vertices have their distance values. At this stage the distances are propagated from top to down where the vertices with $DT(v) = \infty$ receive their distance from their adjacent vertices and adding the distance of an edge (Fig. 2(d) to (g)).

In order to correctly compute the GDT, each surviving vertex counts the number of contractions from its receptive field while this is not needed in computing the DT.

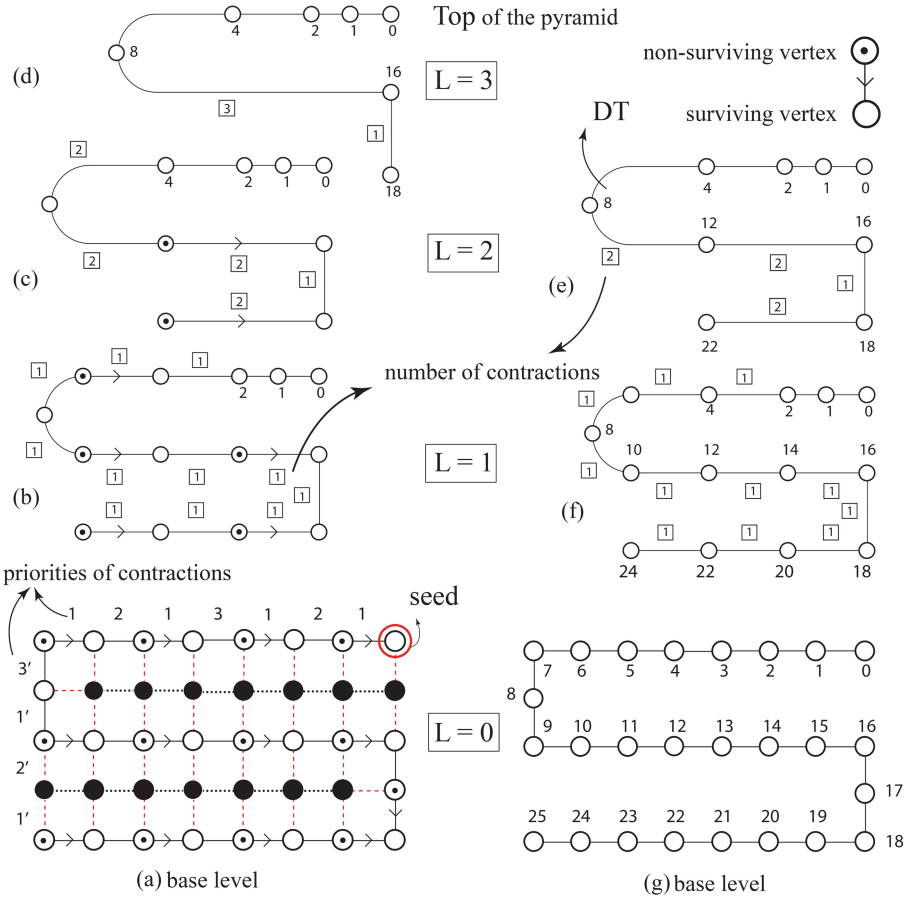


Fig. 2. Logarithmic GDT by irregular pyramid

Proposition 2. *Geodesic distance between two points in the higher dimension is always shorter or equal than in the lower dimension.*

Proof. Assume there is a distance between two points in the lower dimension that is shorter than distance between the same points in higher dimension. Since,

every point in lower dimension is included in higher dimension, the shorter distance in lower dimension exists in the higher dimension as well which is in contradiction with the assumption. □

3 Distance Transforms in n-Gmaps

An n-dimensional generalized map (n-Gmap) is a combinatorial data structure allowing to describe an n-dimensional orientable or non-orientable quasi-manifold with or without boundaries [12]. An n-Gmap is defined by a finite set of darts $\mathcal{D}$ on which act $n+1$ involutions[1] α_i, satisfying composition constraints of the following definition [7]:

Definition 2 (n-Gmap). *An n-dimensional generalized map, or n-Gmap, with $0 \leq n$ is an $(n+2)$-tuple $G = (\mathcal{D}, \alpha_0, ..., \alpha_n)$ where:*

1. *$\mathcal{D}$ is a finite set of darts,*
2. *$\forall i \in \{0, ..., n\}$: α_i is an involution on $\mathcal{D}$*
3. *$\forall i \in \{0, ..., n-2\}$, $\forall j \in \{i+2, ..., n\}$: $\alpha_i \circ \alpha_j$ is an involution.*

Let $(\mathcal{D}, \alpha_0, ..., \alpha_n)$ be an n-Gmap and let us consider its darts $d \in \mathcal{D}$ to be of a unit length. Similar to graphs, we first initialize the distance transform at any nonempty subset of *seed* darts $\mathcal{S} \subseteq \mathcal{D}$ as follows: $\delta(s) := 0 \; \forall s \in \mathcal{S}$ and $\delta(\bar{s}) := \infty \; \forall \bar{s} \in \mathcal{D} \setminus \mathcal{S}$. Scenarios for the initialization (seeding) may include:

- single dart: $\mathcal{S} = \{d_0\}$,
- single i-cell: $\mathcal{S} = \{$all darts of the i-cell$\}$ (e.g., an edge), or
- any multi-combinations of the above, e.g., all edges (1-cells) connecting vertices of different labels resulting from segmentation or connected component labeling.

Similar to graphs, the distances are propagated from the seeds in the breath-first search. The difference to graphs, however, is that the propagation is more general and is driven along (some or all) *involutions* α_i rather than being restricted to the edges of the graph.

Figure 3b shows an example of a 2-Gmap – a 6×6 matrix of vertices (0-cells) of four labels A, B, C, and D where A and B have both two connected components. Edges $(d, \alpha_0(d), \alpha_2(d), \alpha_2(\alpha_0(d))$ connecting different labels[2] are initialized to 0 and distances are propagated following α_0, α_1, and α_2 involutions. Figure 3a illustrates the arrangement of darts around an implicit vertex (X).

The propagation of distances in Fig. 3b is performed equally in all dimensions, i.e., involving all involutions α_i. Excluding a fixed α_j, the propagation is constrained to manifolds of dimensions j. This makes the computation of the *geodesic* distance transforms on n-Gmaps viable.

[1] Self-inverse permutations.
[2] red separators in Fig. 3(b).

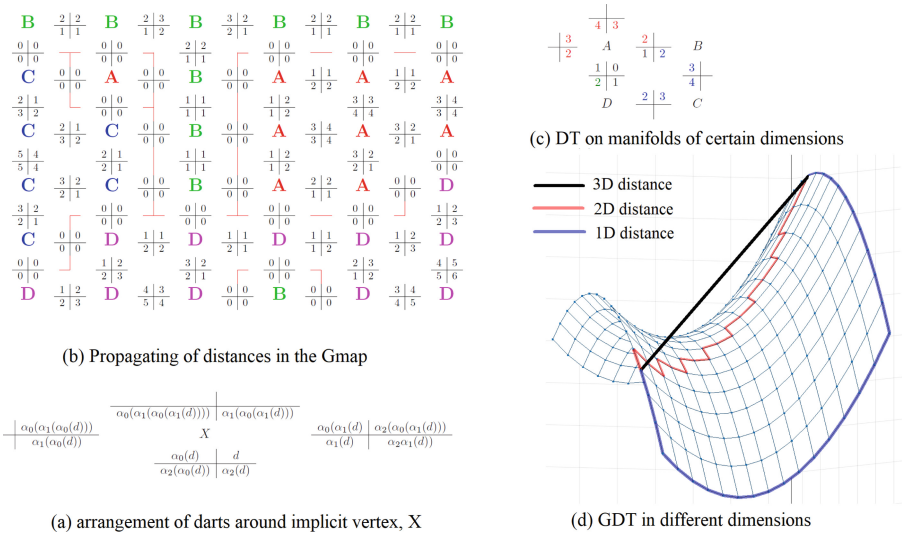


Fig. 3. DT in a Gmap (Color figure online)

We illustrate the effect by a simple 2D example (see Fig. 3c) where we initialize a single dart by zero and propagate distances only by pairs of involutions:

1. $\langle \alpha_0, \alpha_1 \rangle$ denotes the propagation[3] of the orbit $(\alpha_0^*, \alpha_1^*)^*(d_0)$ and identifies the (dual) 2-cell between A,B,C,D. α_2 does not propagate the distance.
2. $\langle \alpha_0, \alpha_2 \rangle$ denotes the propagation of $(\alpha_0^*, \alpha_2^*)^*(d_0)$ and identifies the 1-cell consisting of the four darts between A and D. In this case α_1 does not propagate the distance.
3. $\langle \alpha_1, \alpha_2 \rangle$ denotes the propagation of $(\alpha_1^*, \alpha_2^*)^*(d_0)$ and identifies the 0-cell (a point), the eight darts surrounding A. In this case α_0 does not propagate the distance.

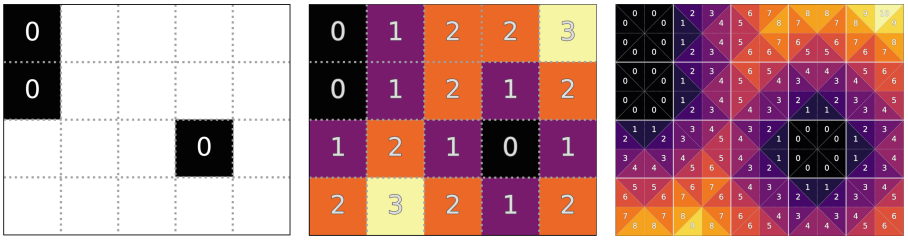
Depending on the initialization and the choice of involutions, distances can thus be propagated along the *boundaries* of any i-cells, $i > 0$. For 3-Gmaps, in addition to 3-cells (volume elements), propagation of distances along their (2D) bounding surfaces or along (1D) curves bounding these surfaces becomes possible. Based on Proposition.2 the GDT in the higher dimension is shorter or equal than in lower dimension (Fig. 3d).

4 Results

As an example of the calculation of distance transforms on 2-Gmaps we refer to Fig. 4. The three black, zero-labeled pixels of the 4×5 image (Fig. 4a) are

[3] Blue distance values belong to the 2-cell, black distances to two types of cells (Fig. 3c).

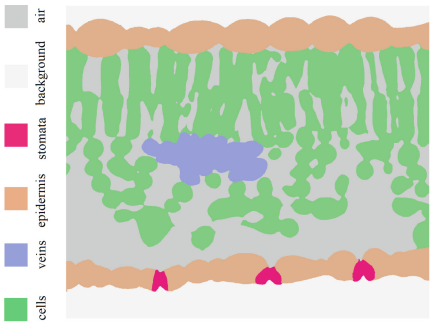
used to seed the distance transform. Figure 4b represents the result of a graph-based distance transform where pixels correspond to vertices of the graph and its edges model the 4-connectivity of the image. The result of the 2-Gmap distance transform is displayed in Fig. 4c. Each pixel corresponds to eight darts which we choose to display by triangles colored by the minimum distance from the seeds. The axes-parallel and the diagonal lines between the triangles of one pixel correspond to α_0 and α_1, respectively. The axes-parallel *pixel-separating* lines correspond to α_2. The 3 seeds of Fig. 4a are represented by total of 24 black, zero-labeled triangles in Fig. 4c. It can be observed that in the 2-Gmaps the distances are propagated in a smoother and a more detailed way.



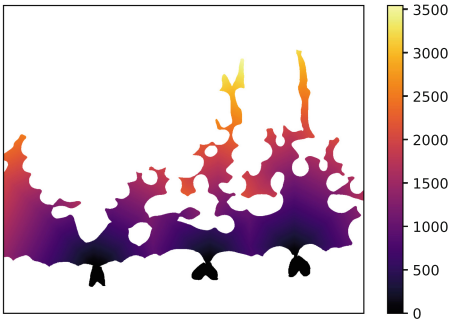
(a) mask with 3 seeds (b) DT (c) DT in Gmap

Fig. 4. Comparison of a graph-based (b) and Gmap-based (c) distance transforms of a binary image (a). Best viewed in color and magnified.

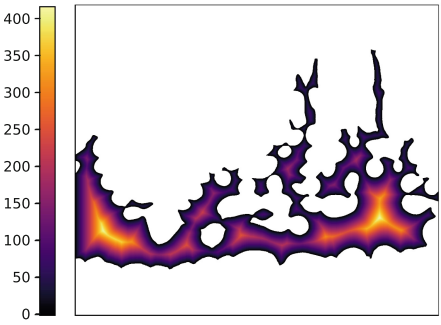
To exploit the advantage of the proposed method in a real application, several geodesic distance transforms (GDTs) are computed through a labeled 2D cross slice of a leaf scan (Fig. 5). The input image (Fig. 5a) has six different labels illustrating different regions inside the leaf. In this figure, the stomas act as gates to control the amount of CO_2 that is entering the leaf. The CO_2 propagates through the airspace to reach the cells and by combining with water and heat the photosynthesis takes place. To model various aspects of the photosynthesis, GDTs may aid in several ways. First, since we are interested in simulations of gas exchange in the leaf [13], we compute the GDT from the stomata through the airspace (Fig. 5b). This is intended to approximate how long it takes to reach the necessary CO_2 concentration. Second, bottlenecks of the airspace supposedly slow down the diffusion processes. We therefore compute the widths of the bottlenecks by the GDT inside the airspace seeded at ist boundary (Fig. 5c). Finally, the GDT from the stomata along the boundary of airspace is calculated (Fig. 5d). This is motivated by the observation that a longer boundary accommodates more cells to perform photosynthesis.



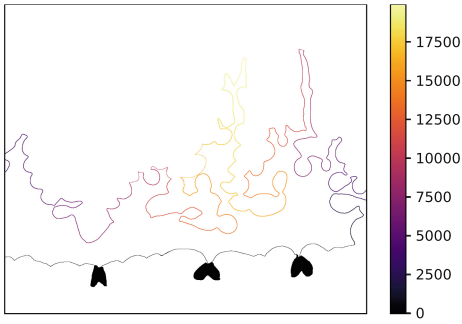
(a) Labeled cross-section of a plant leaf.



(b) GDT in air seeded at stomas.



(c) GDT in air seeded at its boundary.



(d) GDT in air boundary seeded at stomas.

Fig. 5. Computing GDT in a leaf.

It should be noted that the parallel logarithmic complexity of computing the GDT in the proposed method makes it useful for processing the big data. In our data-set each dimension of the 3D input image (leaf) is more than 2000 pixels. Therefore, fast computation of the DT with low complexity is required as shown in [2,3].

5 Conclusions

The paper presents a new algorithm to propagate distances in a graph which is based on a spanning forest of the foreground. The spanning forest is produced in parallel constant complexity and it reduces the linear search space to the length of the longest path in the spanning forest. By preserving the connectivity of connected components (CCs) and the topological information between CCs the proposed algorithm performs the connected component labeling (CCL) and the distance transform (DT) simultaneously. Using the hierarchical structure of the irregular pyramid the new method computes the geodesic distance transform (GDT) with parallel logarithmic complexity that makes it useful for processing of the big data.

We additionally introduce distance transforms for the generalized combinatorial maps (n-Gmaps). We show how they naturally result in a smoother and a higher resolution distance fields. More importantly, however, we show how geodesic distance transforms can efficiently be performed just by omitting relevant involutions from the distance propagation. Finally, we demonstrate how computing GDTs in n-Gmaps may support modelling of the gas exchange in plant leaves.

References

1. Banaeyan, M., Batavia, D., Kropatsch, W.G.: Removing redundancies in binary images. In: International Conference on Intelligent Systems and Patterns Recognition (ISPR), Hammamet, Tunisia, 24–25 March 2022. pp. 221–233. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-08277-1_19
2. Banaeyan, M., Kropatsch, W.G.: Pyramidal connected component labeling by irregular graph pyramid. In: 2021 5th International Conference on Pattern Recognition and Image Analysis (IPRIA), pp. 1–5 (2021)
3. Banaeyan, M., Kropatsch, W.G.: Parallel $\mathcal{O}(log(n))$ computation of the adjacency of connected components. In: International Conference on Pattern Recognition and Artificial Intelligence (ICPRAI), Paris, France, June 1–3, 2022. pp. 102–113. Springer, Cham (2022).https://doi.org/10.1007/978-3-031-09282-4_9
4. Beamer, S., Asanovic, K., Patterson, D.: Direction-optimizing breadth-first search. In: SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. pp. 1–10 (2012). https://doi.org/10.1109/SC.2012.50
5. Borgefors, G.: Distance transformations in arbitrary dimensions. Comput. Vis., Graphics image Process. **27**(3), 321–345 (1984)
6. Brun, L., Kropatsch, W.G.: Hierarchical graph encodings. In: Lézoray, O., Grady, L. (eds.) Image Processing and Analysis with Graphs: Theory and Practice, pp. 305–349. CRC Press (2012)
7. Damiand, G., Lienhardt, P.: Combinatorial Maps: Efficient Data Structures for Computer Graphics and Image Processing. CRC Press (2014)
8. Fabbri, R., Costa, L.D.F., Torelli, J.C., Bruno, O.M.: 2D Euclidean distance transform algorithms: a comparative survey. ACM Comput. Surv. **40**(1), 1–44 (2008)
9. Haxhimusa, Y.: The Structurally Optimal Dual Graph Pyramid and Its Application in Image Partitioning. DISKI, Berlin (2007)
10. Kropatsch, W.G.: Building irregular pyramids by dual graph contraction. IEE-Proc. Visi. Image Signal Process. **142**(No. 6), 366–374 (1995)
11. Latecki, L., Eckhardt, U., Rosenfeld, A.: Well-composed sets. Comput. Image Underst. **61**(1), 70–83 (1995)
12. Lienhardt, P.: Topological models for boundary representation: a comparison with n-dimensional generalized maps. Comput. Aided Des. **23**(1), 59–82 (1991)
13. Momayyezi, M., et al.: Desiccation of the leaf mesophyll and its implications for CO_2 diffusion and light processing. Plant, Cell Enviro. **45**(5), 1362–1381 (2022)
14. Pizlo, Z., Stefanov, E.: Solving large problems with a small working memory. J. Probll. Solv. **6**(1), 5 (2013)
15. Rosenfeld, A., Pfaltz, J.L.: Sequential operations in digital picture processing. Assoc. Comput. Macch. **13**(4), 471–494 (1966)