Distance Transform in Images and Connected Plane Graphs

Majid Banaeyan(✉) and Walter G. Kropatsch

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

Abstract. The distance transform (DT) serves as a crucial operation in numerous image processing and pattern recognition methods, finding broad applications in areas such as skeletonization, map-matching robot self-localization, biomedical imaging, and analysis of binary images. The concept of DT computation can also be extended to non-grid structures and graphs for the calculation of the shortest paths within a graph. This paper introduces two distinct algorithms: the first calculates the DT within a connected plane graph, while the second is designed to compute the DT in a binary image. Both algorithms demonstrate parallel logarithmic complexity of $\mathcal{O}(log(n))$, with n representing the maximum diameter of the largest region in either the connected plane graph or the binary image. To attain this level of complexity, we make the assumption that a sufficient number of independent processing elements are available to facilitate massively parallel processing. Both methods utilize the hierarchical irregular pyramid structure, thereby retaining topological information across regions. These algorithms operate entirely on a local level, making them conducive to parallel implementations. The GPU implementation of these algorithms showcases high performance, with memory bandwidth posing the only significant constraint. The logarithmic complexity of the algorithms boosts execution speed, making them particularly suited to handling large images.

Keywords: Distance transform · Connected plane graph · Parallel logarithmic complexity · Irregular graph pyramids · Parallel processing

1 Introduction

The concept of distance transform (DT) [34], a cornerstone technique in pattern recognition and geometric computations, has a pivotal role in an array of methods. Its utility spans a wide spectrum of applications, including but not limited to skeletonization [31], robotic self-localization through map matching [36], image registration [19], template matching [26,33], line detection in manuscripts [21], and weather forecasting and analysis [9]. Employed primarily on binary images composed of background and foreground regions, the DT produces a new gray-scale image. In this transformed image, the intensity of each foreground pixel reflects the minimum distance to the background.

Supported by the Vienna Science and Technology Fund (WWTF), project LS19-013.

The application of distance transform extends beyond binary images and into the realm of connected plane graphs, enhancing its utility further [27]. A connected plane graph consists of vertices and edges, analogous to the pixels and adjacency relationships in an image. By applying distance transform on such a graph, each vertex obtains a value that signifies the shortest distance to a set of predefined background vertices.

This transformation on connected plane graphs finds immense value in network analysis [28] and routing applications [20]. It serves as a crucial step in path-finding algorithms, where identifying the shortest paths between nodes [18] can result in optimized routing [14,15]. This is particularly useful in applications such as transport logistics, telecommunications routing [12], and even in social network analysis. Thus, distance transform not only offers valuable insights into image analysis but also aids in optimizing network structures and enhancing routing efficiency [11].

In the computation of the Distance Transform (DT), conventional algorithms applied to binary images [13,32,34] or connected plane graphs [18,22] typically propagate distances in a linear-time complexity, denoted as $\mathcal{O}(N)$. Here, N represents the quantity of pixels in a 2D binary image or the count of vertices in a connected plane graph.

Differently, this paper introduces an innovative approach that propagates distances exponentially, thereby computing the DT with a parallel time complexity of $\mathcal{O}(log(n))$. In this instance, n signifies the diameter of the most extensive connected component found within the binary image or the connected plane graph. In the pursuit of achieving parallel logarithmic complexity, it is imperative to note a key assumption we make. We presume that a sufficient number of independent processing units are readily accessible. This assumption is crucial to the successful execution of our proposed method, as it is heavily reliant on simultaneous processing capabilities.

Our proposed method builds upon the concepts presented in a recent publication [7], which leverages the hierarchical structure found in the irregular graph pyramid [4,8,24]. The methodology outlined in [7] focuses on computing the distance transform on a grid structure. Conversely, our paper introduces a novel approach for calculating the distance transform in a general, non-grid connected plane graph, an approach that also effectively solves the shortest path problem.

In this study, we propose two algorithms that exhibit parallel logarithmic complexity. In Sect. 3, we elaborate on the first method for computing the DT in a connected plane graph. Subsequently, Sect. 4 presents the second method for computing the DT in a binary image. Prior to that, in Sect. 2, we provide a recap of the background and definitions. Lastly, in Sect. 5, we evaluate and compare the results obtained from both methods.

2 Background and Definitions

The image pyramid is a stack of images, each presented at a progressively reduced resolution [8]. It is a methodology prevalent in various domains, including image processing [10] and pattern recognition [34]. This approach effectively encapsulates both local and global information across the different levels. The procedure starts with high-resolution data at the base level and progressively transmutes the local details into more abstract, global information as one ascends the pyramid [29].

Pyramids are essentially of two categories: regular and irregular [8]. In regular pyramids, the resolution decreases in a consistent pattern from one image to the next. However, the resolution reduction in irregular pyramids does not follow a fixed rate. A notable drawback of regular pyramids is their lack of shift invariance - even a minor deviation in the initial image can potentially induce significant alterations in the subsequent pyramid [8]. Irregular pyramids offer a solution to this issue, being data-driven hierarchical structures that inherently bypass the shift variance problem.

A digital image can be visualized as a neighborhood graph. Consider $G = (V, E)$ as the neighborhood graph representing image P. Here, V correlates to P, and E links neighboring pixels. The 4-neighborhood representation is typically favored to avoid intersection of edges between diagonal neighbors within 2×2 pixels, thus maintaining the graph's planarity, which would be compromised in an 8-connected graph.

Irregular graph pyramids [24] are a sequence of consecutively reduced graphs, each iteratively constructed from the graph below it through the selection of specific vertices and edges subsets. The pyramid's construction utilizes two fundamental operations: edge contraction and edge removal. In the case of edge contraction, an edge $e = (v, w)$ undergoes contraction, with the endpoints v and w merging and the edge itself being eliminated. Post-operation, the edges originally connected to the combined vertices become incident to the resulting singular vertex. Conversely, edge removal simply entails the removal of an edge without modifying the count of vertices or interfering with the incidence relationships of remaining edges. Throughout the pyramid, the vertices and edges that do not persist to the next level are termed *non-surviving*, while those that do make it to the subsequent level are classified as *surviving*.

A ***plane*** graph [37], is a graph embedded in a two-dimensional plane where its edges exclusively intersect at their endpoints. In the plane graph there are connected spaces between edges and vertices and every such connected area of the plane is called a *face*. A face's degree is quantified by the number of edges that enclose it. Further categorization introduces the notion of an *empty* face, specifically referring to a face that is demarcated by a cycle. For non-empty faces, traversal of the boundary necessitates multiple visits to certain vertices or edges [23]. Empty faces that encompass only a single edge are distinguished as empty *self-loops*. Considering an empty face of degree 2, it would encompass a pair of edges with identical endpoints. These parallel edges are called as *multiple* edges.

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 [5].*

An edge of a CK is denoted by the directed edge and points towards the survivor.

A *Maximal Independent Vertex Set* (**MIVS**) [29] represents a collection of independent vertices within a connected plane graph. Here, independence implies that no two vertices within the set are neighbors [29]. The MIVS method employs an iterative stochastic process based on a uniformly distributed random variable [0, 1] assigned to each vertex [8]. Vertices corresponding to a local maximum of this random variable are surviving vertices, whereas their neighboring vertices are non-survivors. There may be some isolated vertices left, which will be connected to the local maximum in subsequent iterations for the construction of the independent set. The survival probability of a vertex is correlated to the size of its neighborhood [30], which influences the height of the pyramid and the number of iterations required to construct the pyramid [8]. A challenge with the MIVS approach is that the average degree of vertices tends to increase throughout the pyramid [25]. This leads to a reduction in the number of surviving vertices, subsequently decreasing the decimation ratio along the pyramid [16], which inadvertently increases the pyramid's height. To address this drawback, the *Maximum Independent Edge Set* (MIES) was introduced.

The **MIES** [16] applies the MIVS method to an *edge-graph* derived from the original graph G. This edge-graph comprises a vertex for each edge in G, with vertices in the edge graph being connected if their corresponding edges in G are incident to the same vertex. Consequently, the MIES introduces a maximal matching [17] on the initial graph vertices. A matching on a graph refers to a subset of its edges wherein no two edges share a common vertex. Such a matching is deemed *maximal* if it is impossible to add any additional edge without breaching the matching condition [16]. *Maximum* matching, on the other hand, represents the matching scenario that encompasses the largest possible number of edges for a given graph [16].

3 DT in a Connected Plane Graph

In a planar graph $G = (V, E)$, where V and E represent the vertices and edges respectively, distances are determined by the shortest path length [35]. Each vertex $v \in V$ has a set of neighbors denoted by $\mathcal{N}(v) = \{v\} \cup \{w \in V | e = (v, w) \in E\}$ linked via edges. Partition the vertices $V = B \cup F$ with $B \cap F = \emptyset$ into background and foreground vertices in the graph.

In the process of computing the distance transform (DT) on the graph, vertices in the set $b \in B$ act as *seed* vertices, with their respective distances initialized to zero, denoted as $DT(b) = 0$. The foreground $f \in F$ have their distances initialized to infinity, or $DT(f) = \infty$. The calculation of the distance transform utilizes an irregular graph pyramid structure [3,7]. Within this hierarchical structure, information is propagated in two primary directions: (1) **bottom-up** and (2) **top-down**. The bottom-up construction involves computing the DT for a subset of vertices, referred to as surviving vertices, progressively up the pyramid until its apex is reached. Conversely, the top-down process computes the DT for the remaining vertices, proceeding downwards until the DT for all vertices has been calculated at the base level.

3.1 Bottom-Up Construction in the Irregular Pyramid

The irregular pyramid is constructed from the original graph, G, which forms the base of the pyramid, through a bottom-up procedure. This process comprises four iterative steps at each level of the pyramid: (1) propagation of distances, (2) selection of Contraction Kernels (CKs), (3) contraction of the selected CKs, and, (4) simplification.

Distance Propagation. At each level of the pyramid, vertices with previously computed Distance Transform (DT) propagate their distances to their adjacent vertices. This propagation is described by the following relation:

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

Here, d_c is the shortest length of the cumulative lengths of the subpaths with the same end vertices resulting from several steps of contractions.

Selection of Contraction Kernels and Contraction Process. The selection of Contraction Kernels (CKs) is a crucial step in the construction of the pyramid. To ensure a fully parallel construction of the pyramid with logarithmic height, it is necessary for the selected CKs to be independent of each other [6]. To achieve this, we employ the Maximum Independent Edge Set (MIES) method [25], where the chosen CKs share no common vertices. Once the CKs are selected, they are contracted, resulting in a smaller, reduced graph at the subsequent higher level of the pyramid. The *reduction* function [8] computes the lengths of the newly contracted edges and the shortest distance of the surviving vertices.

Simplification. The contraction of selected independent Contraction Kernels (CKs) yields a smaller graph, which might contain parallel edges or empty self-loops. Having no topological information [5] these superfluous edges are referred to as *redundant* edges [2,5]. To simplify the resulting graph, these redundant edges are removed. Only the shortest of a set of multiple edges is kept.

The pyramid's construction concludes when all surviving vertices have received their corresponding DT. This graph represents the apex of the pyramid. Figure 1 provides an example of computing the DT via the bottom-up procedure. The pyramid's base level (Fig. 1a) is the original connected plane graph, which comprises a single filled vertex as the background and the remaining unfilled vertices as the foreground. During the initialization step, the background vertex and its adjacent vertices receive their DT. The edges incident to the background are colored in red, while the remaining foreground edges are shown in dark blue. The selected CKs are indicated by black arrows at various pyramid levels (Fig. 1b, d, f). Post-contraction, the resulting graph (Fig. 1c, e) contains redundant edges. For instance, in Fig. 1c, an empty self-loop is represented by "s". The edges labeled b, c, e, g, l, n, and o are redundant parallel edges. The cumulative sum of previous contractions, d_c, as denoted in Eq. (1), is illustrated by $\boxed{\text{i}}$, where $i = 1, 2$ beside vertices and their corresponding incident edges.

3.2 Top-Down Propagation

The computation of the Distance Transform (DT) for the remaining vertices with unknown distances involves two main steps, executed level by level from the apex to the base of the pyramid: (1) distance propagation, and (2) correction.

Distance Propagation. In the top-down procedure, the hierarchical structure of the pyramid is reconstructed from the apex down to the base level. This reconstruction involves the re-insertion of edges to re-establish connectivity and expand the graph. To compute the DT, each vertex with a previously computed DT propagates its distance to

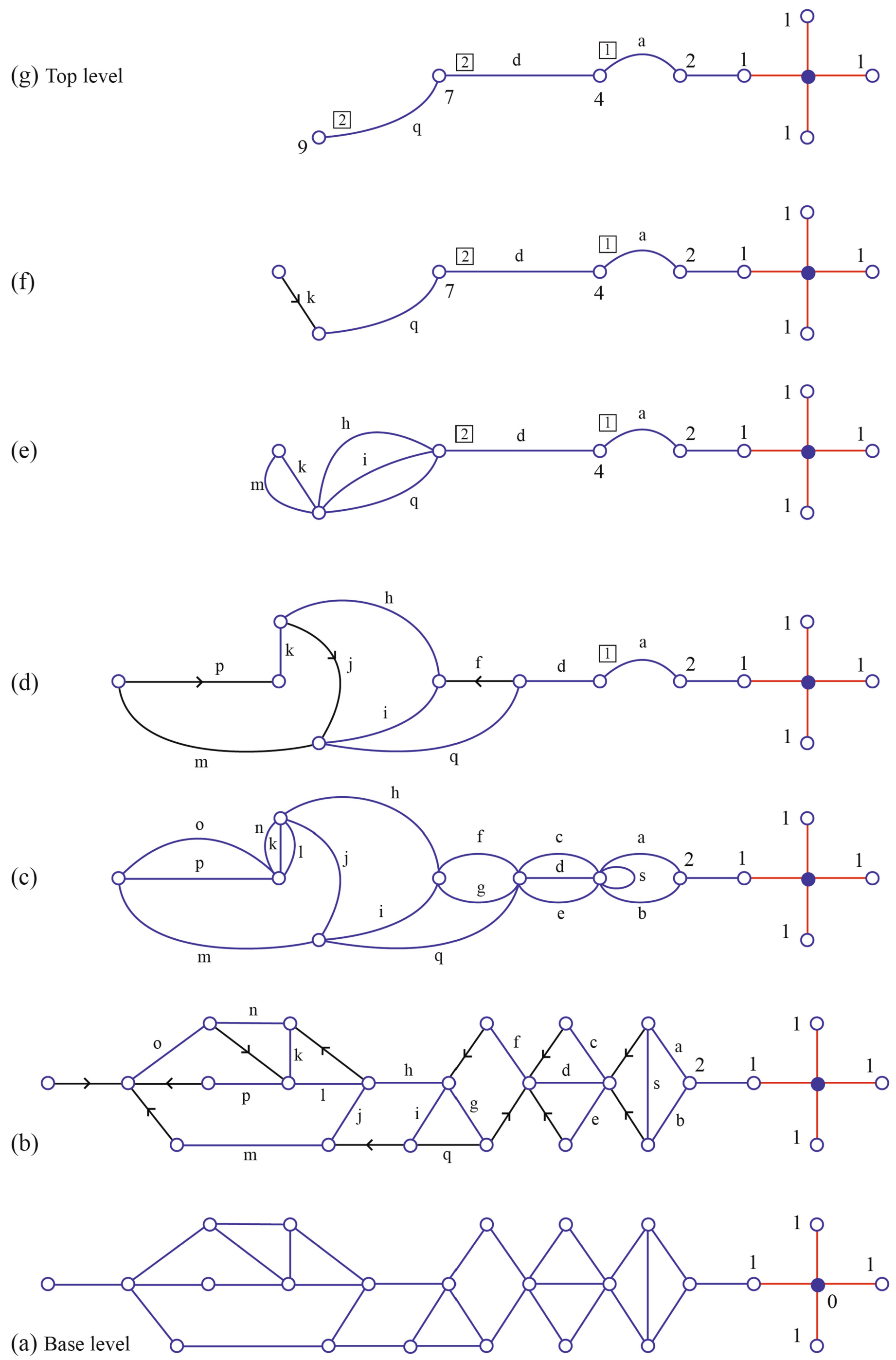

Fig. 1. Bottom-up construction in the pyramid.

its corresponding vertex at the lower level. Following this, one step of DT propagation (as described in Eq. (1)) is conducted, leading to the propagation of distances to the newly inserted vertices. This process continues until the pyramid reaches the base level, at which point all vertices possess their respective DTs.

Correction. According to Eq. (1), the difference between the DT values of any two adjacent vertices at a given pyramid level should be at most $1 + d_c$. However, during the top-down reconstruction of the pyramid, the insertion of a new edge might connect two vertices whose distance apart is greater than $1 + d_c$. In such situations, another application of Eq. (1) is needed to update the new DT and ensure accuracy of the final result.

Figure 2 illustrates the top-down reconstruction of the pyramid, with the correction of certain distances represented by bevelled red lines. Algorithm 1 demonstrates the computation of the Distance Transform (DT) in a connected plane graph.

Algorithm 1. Computing the Distance Transform (DT) in a Connected Plane Graph.

```
1: **Input:** Connected Plane Graph: $G = (V, E) = (B \cup F, E)$   L=pyramid's level
2: **Initialization:** Set $DT(b) = 0; \forall b \in B$, and $DT(f) = \infty,\ \forall f \in F$
3: **While** there are edges to be contracted, perform the following steps for **bottom-up** construction of the pyramid:
4: Propagate the distances using Equ. (1)
5: Select the Contraction Kernels (CKs)
6: Contract the CKs
7: $L \rightarrow L + 1$
8: Record the number of contractions
9: Simplify the resulting reduced graph
10: **End While** (Top of the pyramid is reached)
11: **While** there are vertices with unknown DT in the level below, perform the following steps for **top-down** propagation of distances:
12: $L \rightarrow L - 1$
13: Inherit the computed DT from higher levels
14: Propagate the distances using Equ. (1)
15: Implement corrections if necessary
16: **End While**
```

4 Distance Transform (DT) in a Binary Image

A binary image can be conceptualized as a two-dimensional matrix, with each element assuming a value of either zero or one. The neighborhood graph corresponding to this binary image, denoted by $G = (V, E)$, comprises vertices V representing pixels ($p \in P$) of the image, and edges E signifying the adjacency relationships between these pixels. For the purpose of creating a plane neighborhood graph, we assume 4-nearest neighbor relations between the pixels. The reason is that the 8-connectivity would not be a plane graph [23].

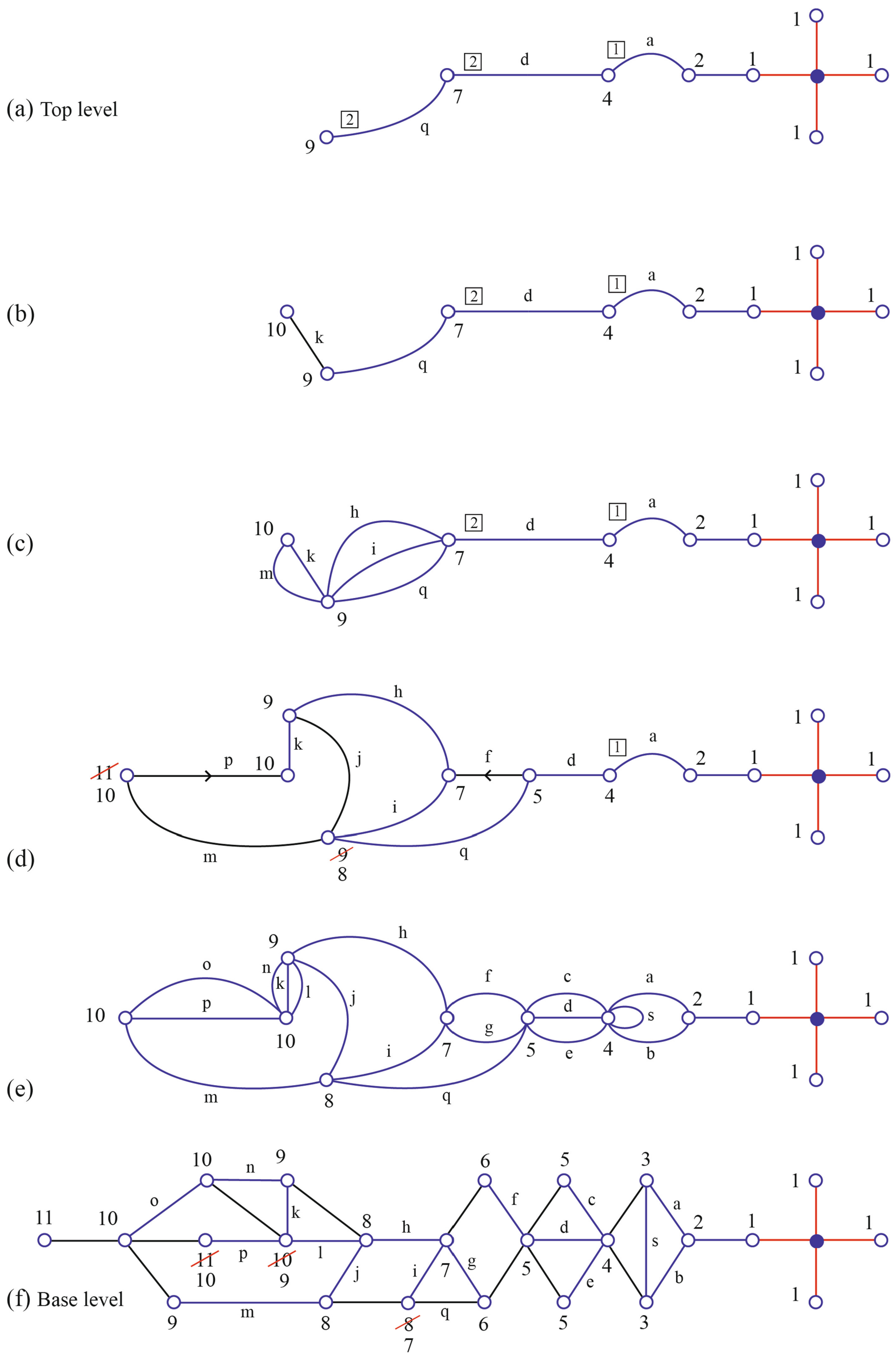

Fig. 2. Top-down propagation of DT.

The computation of the DT in the binary image employs the use of an irregular pyramid as proposed by Banaeyan (2023) [7]. In this hierarchical structure, an efficient selection of Contraction Kernels (CKs) is enabled by defining a *total order* through the indices of the vertices, as suggested in [3].

Suppose the binary image comprises M rows and N columns, with pixel $(1, 1)$ situated at the upper-left corner and pixel (M, N) at the lower-right corner. In such a setup, the vertices of the corresponding graph G are assigned a unique index as follows, based on the approach proposed in [5]:

$$Idx : [1, M] \times [1, N] \rightarrow [1, M \cdot N] \subset \mathbb{N}, \quad Idx(r, c) = (c - 1) \cdot M + r \tag{2}$$

where r and c correspond to the row and column of the pixel, respectively.

The Distance Transform (DT) in the binary image can be calculated using a method similar to Algorithm 1. However, two modifications are incorporated. Firstly, in contrast to computing the DT in the connected plane graph, the simplification step is executed before the contraction by Contraction Kernels (CKs), which expedites the pyramid construction. Secondly, by knowing the coordinates of the vertices, the propagation formula can be directly computed based on the indices of the coordinates, as proposed in [5]:

$$D(v_i) = \min\{D(v_i), D(v_j) \ + \begin{cases} 1 & if\ T = 1 \\ \frac{T}{M} & if\ T \neq 1 \end{cases} \} \tag{3}$$

where

$$v_j \in \mathcal{N}(v_i), \ \ T = |Idx(v_i) - Idx(v_j)| \tag{4}$$

Algorithm 2 outlines the specifics of the proposed method.

Algorithm 2. Computing the Distance Transform (DT) in a binary image.

```
    Input: Neighborhood graph of a binary image: $G = (V, E)$,   L=pyramid's level
 2: Initialization: Set $DT(b) = 0; \forall b \in B$, and $DT(f) = \infty, ; \forall f \in F$
    While there are edges to be contracted, perform the following steps for bottom-up construction of the pyramid:
 4: Propagate the distances using Equ. (1)
    Select the Contraction Kernels (CKs)
 6: Identify the Redundant Edges
    Remove the Redundant Edges
 8: Contract the CKs
    $L \rightarrow L + 1$
10: End While (Top of the pyramid is reached)
    While there are vertices with unknown DT in the level below, perform the following steps for top-down propagation of distances:
12: $L \rightarrow L - 1$
    Inherit the computed DT from higher levels
14: Propagate the distances using Equ. (1)
    Implement corrections if necessary
16: End While
```

It is noteworthy to mention that in calculating the DT of the binary image, the selection of Contraction Kernels (CKs) is executed differently. While in the plane graph the

Maximum Independent Edge Set (MIES) method selects independent CKs at each pyramid level, the binary image employs a different approach. Here, equivalent contraction kernels (ECKs) [5] are selected initially, followed by a logarithmic encoding process which divides the ECKs into a set of independent edges.

5 Evaluation and Results

To underscore the benefits of the introduced logarithmic algorithm, we conducted performance analyses comparing its execution times to those of two other GPU-based methods, MeijsterGPU and FastGPU, as presented in [1]. The experimental environment consisted of MATLAB software running on an AMD Ryzen 7 2700X, 3.7GHz CPU, and an NVIDIA GeForce GTX 2080 TI GPU. The experiments comprised three distinct types of images: Random (Ran.), Mitochondria (Mit.), and MRI. Table 1 elucidates the results obtained from these experimental scenarios. Table 1 [7] presents the image size in its first column, followed by execution times (in ms) for our proposed Logarithmic DT (Log DT) algorithm across three distinct image types, and concludes with the execution times of two other methods. Figure 3 provides a visual comparison of the logarithmic algorithm's performance across the image types. As the Random images have smaller foreground elements compared to the other types, they are processed more rapidly. In Fig. 4, the logarithmic DT's performance is contrasted with that of the MeijsterGPU and FastGPU methods. Our proposed algorithm not only outpaces the others but also displays superior handling of larger images.

Table 1. Execution (ms) of proposed Logarithmic DT, MeijsterGPU and FastGPU [7].

Image-size	Mit. Log DT	MRI Log DT	Ran. Log DT	Ran. MeijsterGPU	Ran. FastGPU
256×256	0.0953	0.1209	0.0645	3.8524	1.7844
512×512	0.410	0.7310	0.3152	14.2005	4.2309
1024×1024	2.6308	5.1501	0.9364	25.8475	12.4668
2048×2048	4.1088	8.9506	1.8577	110.7817	44.9560

It is worth highlighting that all operations within the proposed algorithms are independent and localized, allowing each GPU thread within the shared memory to be allocated to a distinct local process. This distribution creates a performance bottleneck, determined by the capacity of the shared memory. Nevertheless, with an ample number of independent processing elements, the algorithms can operate in full parallelism and maintain logarithmic complexity.

Currently, our team is engaged in the "Water's Gateway to Heaven" project[1], which focuses on high-resolution X-ray micro-tomography (μCT) and fluorescence microscopy. The 3D images involved in this project have dimensions exceeding 2000 in each direction, and the DT plays a crucial role in the challenging task of cell separation. We anticipate that our proposed method will significantly enhance the computational efficiency required for processing such large-scale 3D images. As part of our future work, we plan to evaluate the effectiveness of our method on diverse graph datasets.

[1] https://waters-gateway.boku.ac.at/.

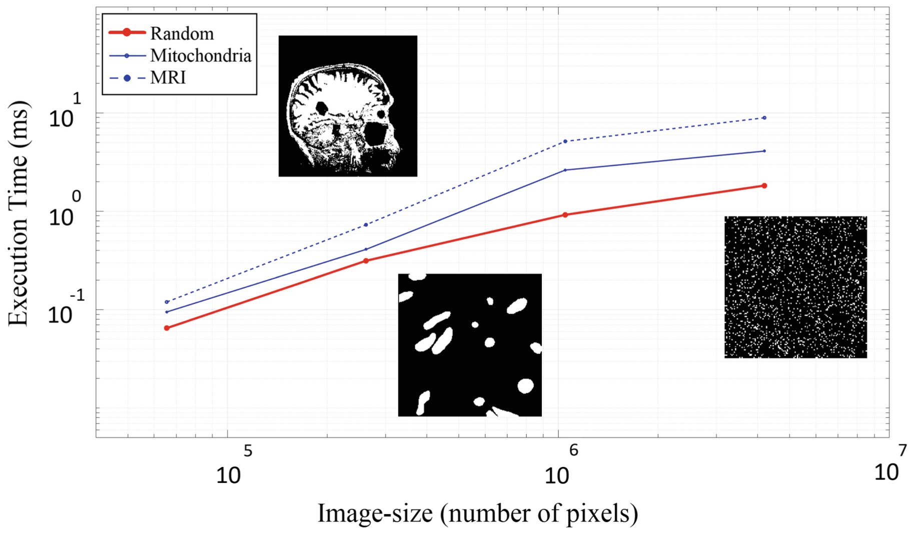

Fig. 3. The proposed logarithmic DT over different images [7].

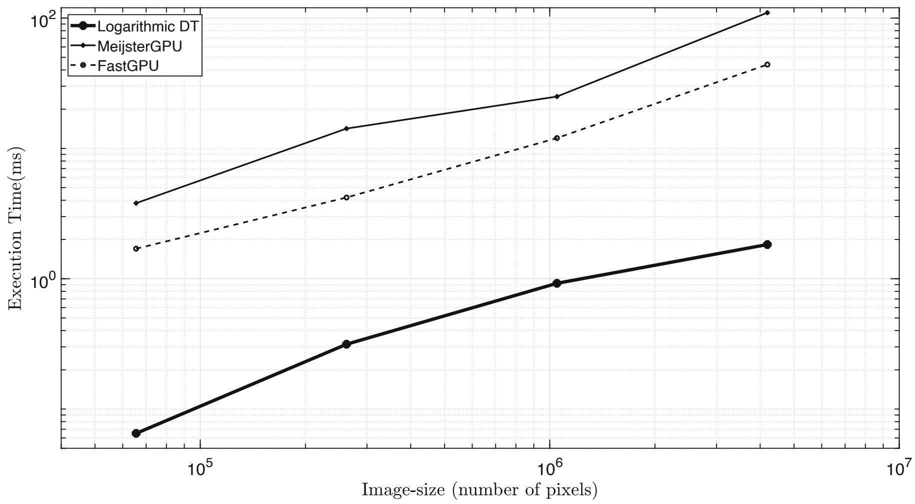

Fig. 4. Comparison of the proposed algorithm with MeijsterGPU and FastGPU [1,7].

6 Conclusion

The distance transform (DT) calculates the shortest distance to reach a point from the set of initial points. In this study, we expanded the concept of DT computation from images (grid structures) to connected plane graphs (non-grid structures). This expansion can enhance the efficiency of solving the shortest path problem in graph theory. The primary advantage of the proposed algorithms lies in their parallel logarithmic complexity, which is achieved through the construction of an irregular pyramid using fully parallel local processing. Evaluation of these algorithms using a database of binary images, and comparison with other contemporary methods, reveals that our proposed algorithm significantly reduces execution time. This efficiency makes it especially advantageous for processing large images.

Acknowledgements. We acknowledge the Paul Scherrer Institut, Villigen, Switzerland for the provision of beamtime at the TOMCAT beamline of the Swiss Light Source and would like to thank Dr. Goran Lovric for his assistance. This work was supported by the Vienna Science and Technology Fund (WWTF), project LS19-013, and by the Austrian Science Fund (FWF), projects M2245 and P30275.

References

1. de Assis Zampirolli, F., Filipe, L.: A fast CUDA-based implementation for the Euclidean distance transform. In: 2017 International Conference on High Performance Computing Simulation (HPCS), pp. 815–818 (2017). https://doi.org/10.1109/HPCS.2017.123
2. Banaeyan, M., Batavia, D., Kropatsch, W.G.: Removing redundancies in binary images. In: Bennour, A., Ensari, T., Kessentini, Y., Eom, S. (eds.) ISPR 2022. Communications in Computer and Information Science, vol. 1589, pp. 221–233. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-08277-1_19
3. Banaeyan, M., Carratù, C., Kropatsch, W.G., Hladůvka, J.: Fast distance transforms in graphs and in gmaps. In: Krzyzak, A., Suen, C.Y., Torsello, A., Nobile, N. (eds.) Structural and Syntactic Pattern Recognition. S+SSPR 2022. LNCS, vol. 13813. pp. 193–202. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-23028-8_20
4. Banaeyan, M., Kropatsch, W.G.: Pyramidal connected component labeling by irregular graph pyramid. In: 5th International Conference on Pattern Recognition and Image Analysis (IPRIA), pp. 1–5 (2021). https://doi.org/10.1109/IPRIA53572.2021.9483533
5. Banaeyan, M., Kropatsch, W.G.: Fast labeled spanning tree in binary irregular graph pyramids. J. Eng. Res. Sci. **1**(10), 69–78 (2022)
6. Banaeyan, M., Kropatsch, W.G.: Parallel $\mathcal{O}(log(n))$ computation of the adjacency of connected components. In: El Yacoubi, M., Granger, E., Yuen, P.C., Pal, U., Vincent, N. (eds.) ICPRAI 2022. LNCS, vol. 13364, pp. 102–113. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-09282-4_9
7. Banaeyan, M., Kropatsch, W.G.: Distance transform in parallel logarithmic complexity. In: Proceedings of the 12th International Conference on Pattern Recognition Applications and Methods - ICPRAM, pp. 115–123. INSTICC, SciTePress (2023). https://doi.org/10.5220/0011681500003411
8. 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)
9. Brunet, D., Sills, D.: A generalized distance transform: theory and applications to weather analysis and forecasting. IEEE Trans. Geosci. Remote Sens. **55**(3), 1752–1764 (2017). https://doi.org/10.1109/TGRS.2016.2632042
10. Burt, P.J., Hong, T.H., Rosenfeld, A.: Segmentation and estimation of image region properties through cooperative hierarchial computation. IEEE Trans. Syst. Man Cybern. **11**(12), 802–809 (1981)

11. Demaine, E.D., Hajiaghayi, M., Klein, P.N.: Node-weighted Steiner tree and group Steiner tree in planar graphs. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 328–340. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02927-1_28
12. Ellinas, G., Stern, T.E.: Automatic protection switching for link failures in optical networks with bi-directional links. In: Proceedings of GLOBECOM'96. 1996 IEEE Global Telecommunications Conference, vol. 1, pp. 152–156. IEEE (1996)
13. Fabbri, R., Costa, L.D.F., Torelli, J.C., Bruno, O.M.: 2D Euclidean distance transform algorithms: a comparative survey. ACM Comput. Surv. (CSUR) **40**(1), 1–44 (2008)
14. Frederickson, G.N.: Fast algorithms for shortest paths in planar graphs, with applications. SIAM J. Comput. **16**, 1004–1022 (1987)
15. Frey, H.: Scalable geographic routing algorithms for wireless ad hoc networks. IEEE Netw. **18**(4), 18–22 (2004)
16. Haxhimusa, Y.: The Structurally Optimal Dual Graph Pyramid and Its Application in Image Partitioning, vol. 308. IOS Press, Amsterdam (2007)
17. Haxhimusa, Y., Glantz, R., Kropatsch, W.G.: Constructing stochastic pyramids by MIDES - maximal independent directed edge set. In: Hancock, E., Vento, M. (eds.) 4th IAPR-TC15 Workshop on Graph-based Representation in Pattern Recognition, vol. 2726, pp. 24–34. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45028-9_3. http://www.prip.tuwien.ac.at/people/krw/more/papers/2003/GbR/haxhimusa.pdf
18. Henzinger, M.R., Klein, P., Rao, S., Subramanian, S.: Faster shortest-path algorithms for planar graphs. J. Comput. Syst. Sci. **55**(1), 3–23 (1997). https://doi.org/10.1006/jcss.1997.1493. https://www.sciencedirect.com/science/article/pii/S0022000097914938
19. Hill, B., Baldock, R.A.: Constrained distance transforms for spatial atlas registration. BMC Bioinform. **16**(1), 1–10 (2015)
20. Hong, S.H., Tokuyama, T.: Beyond Planar Graphs. In: Communications of NII Shonan Meetings, vol. 1, pp. 11–29. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-6533-5
21. Kassis, M., El-Sana, J.: Learning free line detection in manuscripts using distance transform graph. In: 2019 International Conference on Document Analysis and Recognition (ICDAR), pp. 222–227 (2019)
22. Klein, P.N., Mozes, S., Sommer, C.: Structured recursive separator decompositions for planar graphs in linear time. In: Symposium on the Theory of Computing (2012)
23. Klette, R.: Concise Computer Vision, vol. 233. Springer, London (2014). https://doi.org/10.1007/978-1-4471-6320-6
24. Kropatsch, W.G.: Building irregular pyramids by dual graph contraction. IEE-Proc. Vis. Image Sig. Process. **142**(6), 366–374 (1995)
25. Kropatsch, W.G., Haxhimusa, Y., Pizlo, Z., Langs, G.: Vision pyramids that do not grow too high. Pattern Recogn. Lett. **26**(3), 319–337 (2005)
26. Lindblad, J., Sladoje, N.: Linear time distances between fuzzy sets with applications to pattern matching and classification. IEEE Trans. Image Process. **23**(1), 126–136 (2014). https://doi.org/10.1109/TIP.2013.2286904
27. Lotufo, R., Falcao, A., Zampirolli, F.: Fast euclidean distance transform using a graph-search algorithm. In: Proceedings 13th Brazilian Symposium on Computer Graphics and Image Processing (Cat. No.PR00878), pp. 269–275 (2000). https://doi.org/10.1109/SIBGRA.2000.883922
28. Masucci, A.P., Smith, D., Crooks, A., Batty, M.: Random planar graphs and the London street network. Eur. Phys. J. B **71**, 259–271 (2009)
29. Meer, P.: Stochastic image pyramids. Comput. Vis. Graph. Image Process. **45**(3), 269–294 (1989)

30. Montanvert, A., Meer, P., Rosenfeld, A.: Hierarchical image analysis using irregular tessellations. In: Faugeras, O. (ed.) ECCV 1990. LNCS, vol. 427, pp. 28–32. Springer, Heidelberg (1990). https://doi.org/10.1007/BFb0014847
31. Niblack, C., Gibbons, P.B., Capson, D.W.: Generating skeletons and centerlines from the distance transform. CVGIP: Graph. Models Image Process. **54**(5), 420–437 (1992)
32. Nilsson, O., Söderström, A.: Euclidian distance transform algorithms: a comparative study (2007)
33. Prakash, S., Jayaraman, U., Gupta, P.: Ear localization from side face images using distance transform and template matching. In: 2008 First Workshops on Image Processing Theory, Tools and Applications, pp. 1–8. IEEE (2008)
34. Rosenfeld, A., Pfaltz, J.L.: Sequential operations in digital picture processing. Assoc. Comput. Mach. **13**(4), 471–494 (1966)
35. Sharaiha, Y.M., Christofides, N.: A graph-theoretic approach to distance transformations. Pattern Recogn. Lett. **15**(10), 1035–1041 (1994). https://doi.org/10.1016/0167-8655(94)90036-1. https://www.sciencedirect.com/science/article/pii/0167865594900361
36. Sobreira, H., et al.: Map-matching algorithms for robot self-localization: a comparison between perfect match, iterative closest point and normal distributions transform. J. Intell. Robot. Syst. **93**(3), 533–546 (2019)
37. Trudeau, R.: Introduction to Graph Theory. Dover Books on Mathematics, Dover Pub (1993)