Abstract—The performance of a number of image processing methods depends on the output quality of a distance transformation process. Most of the fast distance transformation methodologies are not accurate, while other error-free distance transformation algorithms are not very fast. In this paper, a novel, fast, simple and error-free distance transformation algorithm is presented. By recording the relative $x$ and $y$-coordinates of the examined image pixels, an optimal algorithm can be developed to achieve the distance transformation of an image, correctly and efficiently in constant time without any iteration. Furthermore, the proposed method is general, since it can be used by any kind of distance function, leading to accurate image distance transformations.

Index Terms—Distance transformation, Euclidean distance transformation, Euclidean distance, Image processing, Object representation.

I. INTRODUCTION

Distance transformation (DT) [1] is an operation that converts a digital binary image, consisting of feature (object) and non-feature (background) elements, to a map (another image) where each image pixel has a floating value corresponding to the minimum distance from the background provided by a distance function.

The distance transformation methodologies can be divided into two main categories, according to the achieved accuracy. In the first category, approximation distance transformation techniques were presented by Danielson [2], Borgefors [3], [4], [5] and Ragnemalm [6], [7]. These algorithms get a distance map that is accurate on most of the target points, but they can also produce small errors with some configurations of the object pixels. While these approximations are good enough for many applications, in most of the cases an error-free distance transformation is needed [8].

The second category is consists by algorithms that provide error-free distance transformation maps. These algorithms can be further divided into three classes, according to the order used to scan the pixels:

- Parallel algorithms that presented by Yamada [9], Mitchell [10], [11] and Embrechts [12], are efficient in a cellular array computer since all the pixels at each iteration can be processed in parallel. However, these methods cannot be efficiently implemented on a conventional computer.
- Raster scanning algorithms were proposed by Mullikin [13], Saito et al. [14], Breu et al. [15], Guan et al. [16], Maurer et al. [17], Shih [18] or Felzenszwalb et al. [19].
- Propagation or contour-processing methods were introduced by Vincent [20], Ragnemalm [6], Eggers [21] and Cuisenaire [8], [22].

S. Krinidis is with the Information Management Department, Technological Institute of Kavala, Kavala, Ag. Loukas, 65404, Greece, e-mail: stelios.krinidis@mycosmos.gr.

In propagation algorithms, the information is transmitted from each image pixel to its neighbors, starting from the contour of the object and using a dynamic list to store the pixels in the propagation front. For an Euclidean distance transform, the information which is propagated is usually a vector pointing to the nearest object pixel. As shown by Eggers [23], this can be considered as an efficient implementation of the parallel algorithms of Yamada or Mitchell on conventional computers.

Saito and Toriwaki [14] presented an algorithm for computing the exact Euclidean distance transformation based on dimensionality reduction. This method, as well as the propagation algorithms, are very fast and exact Euclidean distance transformation methods for conventional computers. Nevertheless, their computational cost is highly image dependent and requires $O(N^{3/2})$ time for some input images [8], [17].

Breu et al. [15] introduced a method that also exploits the idea of dimensionality reduction in order to compute a feature map of an image in $O(N)$ time, by constructing the intersection of the Voronoi diagram, whose sites are the object pixels with each row of the image. Then, the Euclidean distance transformation is computed from the feature map.

Guan and Ma [16] improved the computational performance of the Breu’s approach by exploiting the fact that neighboring pixels tend to have the same closest object pixel. Thus, they propagate the closest object pixel information in the form of segment lists rather than individual pixels. Later, Mauren et al. [17] also improved the Breu’s algorithm by taking advantage ideas of Guan and Ma’s method [16]. However, their main advantage is that they compute the Euclidean distance transformation directly, rather than first compute the feature map.

Shih and Wu [18] introduce an algorithm that can compute directly the Euclidean distance transformation with only two image scans exploiting the Borgefors masks [3]. However, this method produces small errors in some cases. These errors occurred when the areas of distance transformation map are disconnected with the 4-direct (and 8-direct) neighborhood [8] due to the fact that digital images are in discrete and not in continues plane.

Felzenszwalb and Huttenlocher [19] provided a linear-time algorithm for solving a class of minimization problems involving a cost function with both local and spatial terms. These problems can be viewed as a generalization of classical distance transforms of binary images, where the binary image is replaced by an arbitrary sampled function.

Although many techniques have been presented for obtaining distance transform, most of them are not error-free, or sufficiently fast, while other could not be applied on conventional computers. Furthermore, other algorithms calculate only the Euclidean distance transformation and are insufficient to be applied with any kind of distance function, whilst other are
too complex to be implemented and understood.

In this paper, a novel distance transformation algorithm solving all the aforementioned problems is introduced. The proposed algorithm can be used for a wide class of distance functions, including the Euclidean distance. The proposed method can compute directly the distance transform very fast without errors, running in $O(N)$ time. This happens because the algorithm needs only four image scans to compute the error-free distance transformation of any kind of distance function. Also, this method may be of practical value, since it is very simple to be implemented and understood.

The remainder of the paper is organized as follows. In Section II, the four-scan distance transformation algorithm is introduced. Some examples of the proposed algorithm are presented in Section III and conclusions are drawn in Section IV.

II. THE FOUR-SCAN DISTANCE TRANSFORMATION ALGORITHM

Danielsson [2] presented a four-scan recursive algorithm for distance transformation by considering a $3 \times 3$ neighborhood and also analyzed the produced errors. In this paper, an error-free distance transformation method to solve the error problem by recording the free distance transformation method to solve the error problem requires four image scans, two forwards and two backwards.

By recording the free distance transformation by considering a $3 \times 3$ foreground (or object) pixels

- In this paper, a novel distance transformation algorithm presented in Section III and conclusions are drawn in Section IV.

**II. THE FOUR-SCAN DISTANCE TRANSFORMATION ALGORITHM**

Danielsson [2] presented a four-scan recursive algorithm for distance transformation by considering a $3 \times 3$ neighborhood and also analyzed the produced errors. In this paper, an error-free distance transformation method to solve the error problem by recording the free distance transformation method to solve the error problem requires four image scans, two forwards and two backwards.

The four-scan distance transformation algorithm is introduced. Some examples of the proposed algorithm are presented in Section III and conclusions are drawn in Section IV.

**II. THE FOUR-SCAN DISTANCE TRANSFORMATION ALGORITHM**

Danielsson [2] presented a four-scan recursive algorithm for distance transformation by considering a $3 \times 3$ neighborhood and also analyzed the produced errors. In this paper, an error-free distance transformation method to solve the error problem by recording the free distance transformation method to solve the error problem requires four image scans, two forwards and two backwards.

**II. THE FOUR-SCAN DISTANCE TRANSFORMATION ALGORITHM**

Danielsson [2] presented a four-scan recursive algorithm for distance transformation by considering a $3 \times 3$ neighborhood and also analyzed the produced errors. In this paper, an error-free distance transformation method to solve the error problem by recording the free distance transformation method to solve the error problem requires four image scans, two forwards and two backwards.

**II. THE FOUR-SCAN DISTANCE TRANSFORMATION ALGORITHM**

Danielsson [2] presented a four-scan recursive algorithm for distance transformation by considering a $3 \times 3$ neighborhood and also analyzed the produced errors. In this paper, an error-free distance transformation method to solve the error problem by recording the free distance transformation method to solve the error problem requires four image scans, two forwards and two backwards.

**II. THE FOUR-SCAN DISTANCE TRANSFORMATION ALGORITHM**

Danielsson [2] presented a four-scan recursive algorithm for distance transformation by considering a $3 \times 3$ neighborhood and also analyzed the produced errors. In this paper, an error-free distance transformation method to solve the error problem by recording the free distance transformation method to solve the error problem requires four image scans, two forwards and two backwards.

**II. THE FOUR-SCAN DISTANCE TRANSFORMATION ALGORITHM**

Danielsson [2] presented a four-scan recursive algorithm for distance transformation by considering a $3 \times 3$ neighborhood and also analyzed the produced errors. In this paper, an error-free distance transformation method to solve the error problem by recording the free distance transformation method to solve the error problem requires four image scans, two forwards and two backwards.
in the relative coordinate threshold \( thr \) (3) is the image size, which, as shown in the experimental result Section, it must be the size of the largest squared empty (background) area of the image. Thus, a preprocessing step which is estimating the aforementioned area could be added to the proposed algorithm.

The proposed algorithm not only calculates the exact distance transformation \( E \) of the input image, but it also computes the signed relative coordinates \( R \) of each image pixel to its nearest background pixel. Furthermore, the proposed method can be used, as already mentioned, with any distance function (city-block, chessboard or Euclidean distance) resulting to the corresponding distance map \( E \).

Furthermore, since the Euclidean distance function involves time-consuming calculations, such as square root over a set of floating-point numbers, the Euclidean distance map calculation can be computationally reduced by exploiting the squared Euclidean distance. Hence, the squared Euclidean distance can be used in all, initial and intermediate steps, and only in the last step (last loop), where the distance map \( E \) is formed, the Euclidean distance can be adopted. Thus, the algorithm can be computationally reduced by this way, since it is released from a large number of square root calculations without losing its accuracy and efficiency.

The proposed distance transformation algorithm, using a distance function \( D(\cdot) \), can be summarized as shown in Figure 3.

Figure 4 illustrates a simple example of using the proposed distance transformation algorithm exploiting the Euclidean distance function on an image (Figure 4a). In Figure 4(b) the four image raster scans with their temporary extracted distance maps (first set of columns), as well as, their relative \( x \) and \( y \)-coordinates matrices (second set of columns), are depicted. The first set is extracted by the first forward image scan (left-to-right and top-to-bottom), the second by the second (semi-backward) scan (right-to-left and top-to-bottom), the third set by the third (backward) image scan (right-to-left and bottom-to-top), while the last set, which is the final result, is extracted by the fourth (semi-forward) raster scan (left-to-right and bottom-to-top). The intermediate distance maps (as shown in Figure 4b) are not necessary to be calculated and stored in the intermediate steps, however, they are shown here for the convenience of the reader. The distance map is necessary to be calculated and stored, only in the last step (last scan) of the proposed algorithm. The relative coordinates threshold \( (thr) \) that was used by the proposed algorithm in this example, due to the fact that it is very simple, was set equal to zero \( (thr = 0) \). Thus, the proposed algorithm stores and uses only the relative coordinates of the image pixel that has the minimum distance.

Furthermore, Figure 5 illustrates the relation among the relative coordinates threshold \( thr \) and the corresponding image dimension \( N \). One can notice that as the image dimension \( N \) is getting larger, the threshold \( thr \) is also getting larger, but it still remains in lowness. We observe that for an image of size \( 2048 \times 2048 \), a threshold \( thr \) equal to 29.55 \( (thr = 29.55) \) is enough. Thus, the relative coordinates threshold \( thr \) remains always in lowness leading the proposed algorithm to run in a linear time.

III. EXPERIMENTAL RESULTS

This Section demonstrates some examples and their distance transformation map provided by the proposed technique. In
order to analyze any difficult case of distance transformations, a special object (Figure 6(a)) is created for the distance transformation map.

Figures 6(b) - 6(d) depict the results (distance transformation maps) of the proposed algorithm applied on Figure 6(a). Figures 6(b) and 6(c) show the distance transformation maps which use chessboard and city-block distance functions respectively, while Figure 6(d) presents the Euclidean distance transformation map.

Figure 7 depicts two examples of real images (Figure 7a) and their corresponding distance transformation maps by using different distance functions. The images of size $256 \times 256$ consist of an object (white pixels) and the background (black pixels). Figures 7(b) and 7(c) show the distance transformation maps extracted by the proposed algorithm applied on images depicted in Figure 7(a) by using chessboard and city-block distance functions respectively. Finally, Figure 7(d) illustrates the Euclidean distance transformation maps of the same images (Figure 7a) as they were extracted by the proposed method.

Also, Figure 8 illustrates two examples of test images (Figure 8a) and theirs corresponding distance transformation maps by using different distance functions. Test images of size $512 \times 512$ were generated by suitably combining squares and circles of different radii, and by randomly choosing midpoints of:

- black pixel squares with random radii $4 \leq r \leq 40$, and
- black pixel circles with random radii $4 \leq r \leq 40$,

respectively, until the number of black pixels exceeded the 15% area of the image (Figure 8a). Objects generated by squares represent objects with straight boundaries and corners, while objects generated by circles represent objects with rounded boundaries and without corners.

Furthermore, two difficult examples in Euclidean space are depicted in Figure 9. In those two examples is clearly shown the usage of the storing list in the proposed algorithm, since in Euclidean space could be formed discontinuities of the influence area of each pixel driving some pixels to be disconnected from a tile in the Voronoi division. Hence, if the algorithm is based only at the eight nearest neighbors, then it could lose those disconnected pixels reaching to a wrong distance transformation map. Thus, the usage of the storing list in the proposed algorithm is critical in order to receive the correct distance transformation map (Figures 10 and 11). Those discontinuities could be occurred in an image only when its empty (background) area is large enough (Figure 9) and it is not influenced by other neighboring pixels. Figure 10 shows the relative $x$ and $y$-coordinates matrix (with list of $R(p)$) of the transposed image depicted in Figure 9(a) after the first two image raster scans, while Figure 10 illustrates the final $x$ and $y$-coordinates matrix. The relative coordinates threshold used in that case was $thr = 2.57$, since the largest empty area in the image has size $17 \times 11 (N = 17)$.

Finally, a quick complexity analysis for the proposed algorithm is presented. It is rather preferred to perform a quick complexity analysis than a running time experiment which depends on the programming style and the language used. The proposed algorithm has a complexity of $O(N)$. It includes only four image scans, with each scan carrying at most five pixel
lists depending on the relative coordinates threshold (3). As it is proven in the Appendix, the relative coordinates threshold (3) does not worsen the complexity of the algorithm, and as a consequence the proposed method is of linear complexity.

IV. Conclusion

In this paper, a novel, very fast, simple to be implemented and understood error-free distance transformation algorithm, is introduced. The proposed method requires only four
scans of the input image by using $3 \times 3$ neighborhood masks in order to compute the distance transformation map exploiting any distance function. Moreover, the proposed algorithm is easy to be implemented and understood and it does not require any time consuming process, as well as any extra pre-processing steps. The error-free distance transformation map of the input image is correctly and efficiently achieved in constant time by recording the relative $x$ and $y$-coordinates of each image pixel. Furthermore, the proposed method is general, since it could be used with any kind of distance function leading to accurate 2D distance transformations.

Finally, the extension of the proposed method to higher dimensions is an open problem and being studied.

**APPENDIX**

The proof of the relation among the relative coordinate threshold $\text{thr}$ (3) used by the proposed algorithm and the image dimension $N$ lies in this Appendix. The problem to be solved occurs when a pixel is disconnected from a tile in the Voronoi division, as shown in Figure 12.

Firstly, some assumptions are made. The image under consideration is of $N \times N$ size, and is composed by three background pixels $A$, $B$ and $C$ as shown in Figure 13. The influence area of each pixel is defined by the perpendicular bisector of the line defined by the pixel and its nearest neighbor pixel (lines $PA$ and $PE$ in Figure 13). Thus, the disconnected pixels in an influence area of a pixel could lie in the triangle $PZH$. Our goal is to determine the distance $d_1 = d_{PZ}$ of the line $PZ$ according to the image dimensions. The triangle $PZH$ is defined to be isosceles, and according to law of cosines one can obtain:

$$d_1^2 = \frac{d_{ZH}^2}{2(1 - \cos(\theta))}. \quad (A.1)$$

where $\theta$ is the angle $ZPH$ and $d_{ZH}$ is the distance between points $Z$ and $H$.

Furthermore, according to the geometry theory, point $P$ is the circumcenter of the circumcircle defined by the triangle $PZH$.
Fig. 8. Distance transformation maps on random circle and square images. (a) The random circle and square images. The distance transformation map extracted by the proposed algorithm exploiting (b) chessboard, (c) city-block and (d) Euclidean distance function.

Thus, distances $d_{PA}$, $d_{PB}$ and $d_{PC}$ are equal to the radius of the circumcircle $R$ ($d_{PA} = d_{PB} = d_{PC} = R$). Furthermore, the triangle $ACP$ is also isosceles ($d_{PA} = d_{PC}$), with the angle $APC$ to be equal to $APC = 2\theta$. Thus, according to law of cosines for the triangle $ACP$, one can obtain:

$$\cos^2(\theta) = 1 - \frac{d_{AC}^2}{4R^2}.$$  \hspace{1cm} (A.2)

Once the above notations have been made, it can be easily observed that while rate $\frac{d_{AC}}{R}$ and angle $\theta$ is getting smaller the desired distance $d_1$ is getting larger. The above notations and a connection between distance $d_1$ and the image dimension $N$, will be studied in the rest of the Appendix.

All points $A$, $B$ and $C$ are image pixels, while point $P$ lies in the domain of the image under examination. In order to complete our study, the “worst” case will be studied, i.e., the case that the circumcircle of the triangle $ABC$ has the largest radius $R$, while the angle $\theta$ is the minimum. Thus, the circumcenter must be at the upper right corner of the image, and as a consequence points $B$ and $C$ are at the lower right corner of the image. This happens, since the circumcenter $P$ lies on the perpendicular bisector of the line $BC$. Point $A$ and the desired distance $d_1$, will be located based on those
In order to have the smaller angle $\theta$, where points $A$, $B$, and $C$ are defined as:

$$B = \begin{bmatrix} N-1 \\ N \end{bmatrix},$$

$$C = \begin{bmatrix} N \\ N \end{bmatrix},$$

where $N$ is the dimension of the image under consideration. In order to have the smaller angle $\theta$, the point $A$ is located at the $N-1$ row of the image ($A = [A_x, N-1]^T$). On the other hand, the circumcenter $P$ lies on:

$$P = \begin{bmatrix} N-1 \\ N+2A_y-N-A_x^2-A_x \end{bmatrix}.$$ (A.4)

As it was above mentioned, the worst case for point $P$ is to be located at $P = [N-\frac{1}{2}, 1]^T$, which is the upper right corner of the image. Thus, if one sets $P_y = 1$ and $A_y = N-1$ in (A.4), the following equation is obtained:

$$A_x^2 + (1-2N)A_x + (N^2 - 3N + 3) = 0.$$ (A.5)

Solving the quadratic equation defined above, one reaches to the solution:

$$A_x = \frac{2N-1-\sqrt{8N-11}}{2}.$$ (A.6)

Once the coordinates of point $A$ are calculated, it is easy to compute the distance $d_{AC}$ and the radius $R$ and, consequently the desired angle $\theta$ by using (A.2) as follows:

$$\cos^2(\theta) = 1 - \frac{d_{AC}^2}{R^2} = 1 - \frac{(A_x-C_x)^2+(A_y-C_y)^2}{4(N-1)^2} = 1 - \frac{(2N-1-\sqrt{8N-11})^2+(N-1-N)^2}{4(N-1)^2}.$$ (A.7)

Combining (A.1), (A.7) and the assumption that distance $d_{ZH}$ is equal to 1 ($d_{ZH} = 1$), since in that way a pixel will always lie in the triangle $PZH$, the new distance $d_1$ is given by:

$$d_1^2 = \frac{d_{ZH}^2}{4(1-\sqrt{16N^2-40N+22-2\sqrt{8N-11}})} = \frac{2}{4-\sqrt{16N^2-40N+22-2\sqrt{8N-11}}}.$$ (A.8)

As mentioned above, the triangle $PZH$ is isosceles and also has $d_{ZH} = 1$, thus it always contains a pixel. Moreover, triangle $PAM$ is also isosceles and contains no pixels. However, another one distance that could be determined is distance $d_2 = d_{PA}$ of the line $PA$ according to image dimensions.

Point $A$ is defined as $A = [N-1, \Lambda_y]^T$ and is located on line $PA$. Thus, the coordinates of point $\Lambda$ must verify line equation which is determined by points $P$ and $\Lambda$, as follows:

$$0 = (\Delta_y - P_y)\Lambda_x + (P_x - \Delta_x)\Lambda_y + (\Delta_x-P_x-\Delta_y)\Lambda_y$$

$$\Lambda_y = \frac{2N-\Lambda_x-3}{N-A_x^2-N-A_x}. \quad \Lambda_y = \frac{3+\sqrt{8N-11}}{4}.$$ (A.9)

Once the coordinates of point $\Lambda$ are calculated, it is easy to compute the distance $d_2 = d_{PA}$ as follows:

$$d_2^2 = (P_x - \Lambda_x)^2 + (P_y - \Lambda_y)^2$$

$$d_2^2 = (N-1/2 - (N-1))^2 + (1 - 3+\sqrt{8N-11})^2$$

$$d_2^2 = \frac{(\sqrt{8N-11}-1)^2+4}{16}.$$ (A.10)

If the relative coordinate threshold $thr$ of the proposed algorithm is set equal to $d = d_1 - d_2$, it is safe to be claimed that all the discontinuities carried by the Euclidean distance in the discrete domain will be eliminated. Hence, the relative coordinate threshold $thr$ is related to the image dimension $N$ and defined as:

$$thr = \frac{2}{\sqrt{16N^2-40N+22-2\sqrt{8N-11}}} = \frac{(\sqrt{8N-11}-1)^2+4}{16}.$$ (A.11)

**References**


Stelios Krinidis was born in Kavala, Greece, in 1978. He received the B.Sc. in informatics in 1999 and the Ph.D. degree in informatics in 2004, both from the Aristotle University of Thessaloniki, Thessaloniki, Greece. From 1999 to 2004, he was a researcher and teaching assistant in the Department of Informatics, University of Thessaloniki. From 2005 to 2011, he is a temporary lecturer in the Department of Information Management, Technological Institute of Kavala where he is currently a senior researcher. His current research interests include computational intelligence, pattern recognition, digital signal and 2D and 3D image processing and analysis, and computer vision.