Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

ELSEVIER Computer Vision and Image Understanding 93 (2004) 195–205

Computer Vision and Image Understanding

www.elsevier.com/locate/cviu

Note

Fast Euclidean distance transformation in two scans using a 3×3 neighborhood

Frank Y. Shih* and Yi-Ta Wu

Computer Vision Laboratory, College of Computing Sciences, New Jersey Institute of Technology, Newark, NJ 07102, USA

Received 10 September 2002; accepted 18 September 2003

Abstract

Cuisenaire and Macq [Comp. Vis. Image Understand., 76(2) (1999) 163] proposed a fast Euclidean distance transformation (EDT) by propagation using multiple neighborhoods and bucket sorting. To save the time for bucket sorting and to reduce the complexity of multiple neighborhoods, we propose a new, simple and fast EDT in two scans using a 3×3 neighborhood. By recording the relative x- and y-coordinates, an optimal two-scan algorithm can be developed to achieve the EDT correctly and efficiently in a constant time without iterations.

Keywords: Distance transformation; Euclidean distance; Image processing; Object representation

1. Introduction

Distance transformation (DT) is to convert a digital binary image that consists of object (foreground) and non-object (background) pixels into another image in which each object pixel has a value corresponding to the minimum distance from the background by a distance function. The interior of a closed boundary is considered as object pixels and exterior as background pixels in this paper. By simply exchanging the roles of object and background, DT can be applied on the outside pixels of a closed boundary as in [2].

Let S be a set of pairs of integers. The function d mapping from $S \times S$ to non-negative integers is called *a distance function*, if it is

* Corresponding author.
E-mail address: shih@njit.edu (F.Y. Shih).

1077-3142/$ - see front matter
doi:10.1016/j.cviu.2003.09.004

(a) *Positive definite*: That is $d(p,q) \geqslant 0$, and $= 0$, if and only if $p = q$, for all $p, q \in S$.
(b) *Symmetric*: That is $d(p,q) = d(q,p)$, for all $p, q \in S$.
(c) *Triangular*: That is $d(p,r) \leqslant d(p,q) + d(q,r)$, for all $p, q, r \in S$.

Among different kinds of *distance transformation*, the Euclidean distance transform (EDT) is often-used because of its rotation invariance property, but it involves the time-consuming calculations such as square, square-root, and the minimum over a set of floating-point numbers. In [3–5,10–12], a *mathematical morphology* approach was proposed to realize the EDT using gray-scale erosions with successive small distance *structuring elements* by decomposition. Furthermore, a squared Euclidean-distance structuring element was used to perform the squared Euclidean distance transform (SEDT). Shih and Wu [4] decomposed the squared *Euclidean-distance structuring element* into successive dilations of a set of 3×3 structuring components. Hence, the SEDT is equivalent to the successive erosions of the result at the preceding stage by each structuring component. EDT can be finally obtained simply by a square-root operation over the image. In particular, image analysis tasks can directly take the result of SEDT as input.

The approaches to achieve distance transformation do not adopt directly the definition of the minimum distance from an object pixel to all background border pixels, since their computations are extremely time-consuming. Previous researches in [1–9] represent a sampled set of successful efforts in improving the speed efficiency. In general, the algorithms of DT can be categorized into two classes: one is the *iterative* method which is efficient in a cellular array computer since all the pixels at each iteration can be processed in parallel, and the other is *sequential* (or *recursive*) method which is suited for a conventional computer by avoiding iterations with the efficiency to be independent of object size. Using the general machines that most people working in digital image processing have access to, sequential algorithms are often much more efficient than iterative ones.

Although many techniques have been presented for obtaining *Euclidean distance transform*, most of them are either inefficient or complex to implement and understand. Cuisenaire and Macq [1] proposed a fast EDT by propagation using multiple neighborhoods and bucket sorting. In this paper, a size-independent two-scan algorithm is presented to achieve the EDT correctly and efficiently. It is organized as follows. In Section 2, we reviews the distance transforms. Section 3 introduces the technique for the two-scan algorithm by using a 3×3 neighborhood. Its mathematical proof is shown in Section 4. Some examples of the two-scan algorithm is presented in Section 5. Finally, the conclusions are made in Section 6.

2. Distance transforms

2.1. The definitions of three commonly used distance transforms

Three distance functions are often used in digital image processing. If there exist two points $p = (x, y)$ and $q = (u, v)$ in a digital image, the distance function is defined as follows:

(a) *City-block distance*: $d_4(p,q) = |x-u| + |y-v|$.
(b) *Chessboard distance*: $d_8(p,q) = \max(|x-u|, |y-v|)$.
(c) *Euclidean distance*: $d_e(p,q) = \sqrt{(x-u)^2 + (y-v)^2}$.

Note that the pixels with *city-block distance* 1 counting from p correspond to 4-neighbors of p, and with *chessboard distance* 1 correspond to 8-neighbors of p. These d_4 and d_8 are integer-valued; however, d_e is not.

2.2. Iterative and two-scan methods for achieving city-block and chessboard distance transforms

City-block and *chessboard* distances are very easy to compute since they can be recursively accumulated by considering only 4- or 8-neighbors, respectively, at one time. The iterative algorithm works for *city-block* or *chessboard* distances is as follows. Given an X_s, which is 1 at the pixels of S and 0 elsewhere, X_s^m is defined recursively for $m = 1, 2, \cdots$ as

$$X_s^m(p) = X_s^0 + \min_{d(q,p) \leqslant 1} X_s^{m-1}(q), \tag{1}$$

where $X_s^0 = X_s$ is the initial binary image. Thus X_s^m can be computed by performing a local operation on the pair of arrays X_s^0 and X_s^{m-1} at each point.

The iterative algorithm is very efficient on a cellular array computer since each iteration can be performed at all points in parallel, and the number of iterations is at most the radius of the picture. However in a conventional computer, each iteration requires the processing of the entire image that presents inefficiency. A two-scan algorithm is therefore developed as below.

Assume that the border and background of a picture consist entirely of 0s. Let $N_1(p)$ be the set of (4- or 8-) neighbors that precede p in a row-by-row (left to right, top to bottom) scan of the picture, and let $N_2(p)$ be the remaining neighbors of p. For example, $N_1(x,y)$ consists of the 4-neighbors $(x-1,y)$ and $(x,y+1)$, as well as (if for 8-neighbors) $(x-1,y+1)$ and $(x+1,y+1)$. The algorithm is

$$X'_S(p) = \begin{cases} 0 & \text{if } p \in \bar{S}, \\ \min\limits_{q \in N_1} X'_S(q) + 1 & \text{otherwise}, \end{cases} \tag{2}$$

$$X''_S(p) = \min_{q \in N_2} [X'_s(p), X''_S(q) + 1], \tag{3}$$

where $X'_S(p)$ is computed in a left to right, top to bottom scan, since for each p, X'_S has already been computed for the qs in N_1. Similarly, X''_S is computed in a reverse scan (right to left, bottom to top). Another algorithm only needs two operations in opposite orders of image scanning: one scan is in the left-to-right top-to-bottom direction, and the other is in the right-to-left bottom-to-top direction [8,13].

A disadvantage of *city-block* and *chessboard* distances is that both distance measures are very sensitive to the orientation of an object. The *Euclidean distance* by definition is rotation-invariant. However, its square-root calculation is costly and its global computation of distance computation is very difficult to decompose into small neighborhood

operations due to its non-linearity of *Euclidean distance* variations. Thus, the algorithms to approximate the EDT have been extensively investigated [2–4,6,7].

3. The two-scan algorithm by using a 3×3 neighborhood

Borgefors [6] has presented a two-scan recursive algorithm on EDT by considering a 3×3 neighborhood and has analyzed the errors produced. We further develop the exact EDT to solve the error problem by recording the x- and y-coordinates in a 3×3 neighborhood. The algorithm only requires two image scans that are forward and backward raster scans.

Let 8 neighbors of p be denoted by $q_1, q_2, \ldots, q_8$ as illustrated in Fig. 1. Thus $N_1(p) = \{q_1, q_2, q_3, q_4\}$ and $N_2(p) = \{q_5, q_6, q_7, q_8\}$. Prior to discussing the algorithm, the notations used are first introduced.

f: A binary image.

F: The set of foreground (or object) pixels.

F': The set of background pixels.

O: The set of background boundary pixels (i.e., 8-adjacent to foreground pixels).

Q: The set of foreground pixels which already have the minimum squared Euclidean distances and the closest background boundary pixel is located anywhere.

$R(p)$: The relative-coordinates vector $R(p) = (R_x, R_y)$ of pixel p, which records the horizontal and vertical pixel-distances between p and the closest background pixel. It is initialized as all (0,0). Note that, $R_x(p)$ and $R_y(p)$ indicate the horizontal and vertical pixel-distances, respectively.

$h(p,q)$: The difference of the *squared Euclidean distances* of p and q, where $q \in N_1 \cup N_2$.

$G(p,q)$: The difference of the relative coordinates of p and q, where $q \in N_1 \cup N_2$.

The $h(p,q)$ and $G(p,q)$ can be computed as follows, where h function is further explained in the proof of the Algorithm.

$$h(p,q) = \begin{cases} 2R_x(q) + 1 & \text{if } q \in \{q_1, q_5\}, \\ 2R_y(q) + 1 & \text{if } q \in \{q_3, q_7\}, \\ 2(R_x(q) + R_y(q) + 1) & \text{if } q \in \{q_2, q_4, q_6, q_8\}, \end{cases} \tag{4}$$

$$G(p,q) = \begin{cases} (1,0) & \text{if } q \in \{q_1, q_5\}, \\ (0,1) & \text{if } q \in \{q_3, q_7\}, \\ (1,1) & \text{if } q \in \{q_2, q_4, q_6, q_8\}. \end{cases} \tag{5}$$

The two-scan algorithm is shown below and its flowchart is shown in Fig. 2.

q_2	q_3	q_4
q_1	p	q_5
q_8	q_7	q_6

Fig. 1. The 8 neighborhoods of p.

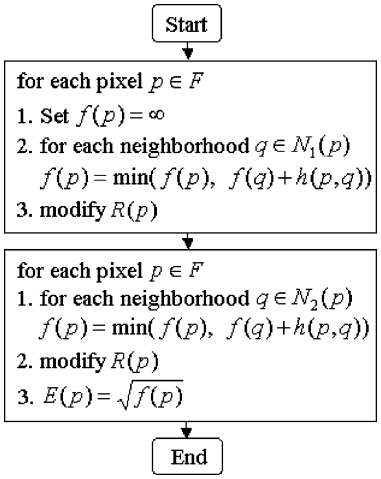

Fig. 2. The flowchart of the two-scan algorithm by using a 3×3 neighborhood.

Algorithm.

1. (Forward) Raster-Scan:
   if $p \in F$ {
   $f(p) = \infty$;
   for $q = q_1$ to q_4
   $f(p) = \min(f(p), f(q) + h(p,q))$;
   if exist q that containing minimum $(f(q) + h(p,q))$ and $(f(q) + h(p,q) < f(p))$
   $R(p) = R(q) + G(p,q)$;
   }
2. Backward (Reverse) Raster-Scan:
   if $p \in F$ {
   for $q = q_5$ to q_8
   $f(p) = \min(f(p), f(q) + h(p,q))$;
   if exist q that containing minimum $(f(q) + h(p,q))$ and $(f(q) + h(p,q) < f(p))$
   $R(p) = R(q) + G(p,q)$;
   $E(p) = \sqrt{f(p)}$;
   }

4. The proof of the two-scan algorithm

4.1. The proof of the equations for achieving SEDT

4.1.1. The nearest background pixel located on the left-top side

For a raster scan in this paper, the concerned neighborhoods can be categorized into four cases when calculating the SEDT of pixel p . Fig. 3 shows a forward raster

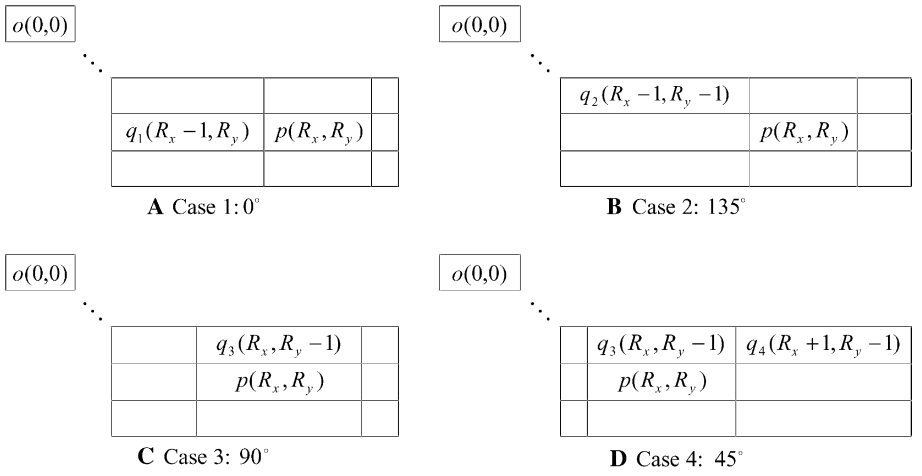

Fig. 3. The four cases when o is located on the left-top side of p.

scan when and the nearest background pixel, o, is located on the left-top part of p. The difference of the *squared Euclidean distances* of p and q in the four cases can be obtained by the following equations.

Case 1. The smallest SED is obtained from q_1.

Assume $q_1 \in Q$. Let the relative coordinates of q_1 be $R(q_1) = (R_x - 1, R_y)$ that are counted from o, where $o \in O$. The squared Euclidean distance (SED) at q_1 is $oq_1^2 = (R_x - 1)^2 + R_y^2$. Therefore, the SED at p will be $op^2 = R_x^2 + R_y^2$ since p is located on the one pixel away from q_1. Note that oq_1^2 is the smallest SED from all background pixels and the location of o is not necessarily unique but may not more than 4 places. The difference of the SED of p and q_1 is

$$h(p, q_1) = op^2 - oq_1^2 = [R_x^2 + R_y^2] - [(R_x - 1)^2 + R_y^2] = 2R_x - 1 = 2(R_x - 1) + 1.$$

Since $R_x(q_1) = (R_x - 1)$ and $G(p, q_1) = (1, 0)$, we obtain $h(p, q_1) = 2R_x(q_1) + 1$ and $R(p) = (R_x(q_1) + 1, R_y(q_1))$.

Case 2. The smallest SED is obtained from q_2.

Assume $q_2 \in Q$. Let the relative coordinates of q_2 be $R(q_2) = (R_x - 1, R_y - 1)$ that is counted from o, where $o \in O$. Similarly, the difference of the SED of p and q_2 is

$$\begin{aligned} h(p, q_2) &= op^2 - oq_2^2 = [R_x^2 + R_y^2] - [(R_x - 1)^2 + (R_y - 1)^2] = 2R_x + 2R_y - 2 \\ &= 2((R_x - 1) + (R_y - 1) + 1). \end{aligned}$$

Since $R_x(q_2) = (R_x - 1), R_y(q_2) = (R_y - 1)$ and $G(p, q_2) = (1, 1)$, we obtain $h(p, q_2) = 2(R_x(q_2) + R_y(q_2) + 1)$ and $R(p) = (R_x(q_2) + 1, R_y(q_2) + 1)$.

Case 3. The smallest SED is obtained from q_3.

Assume $q_3 \in Q$. Let the relative coordinates of q_3 be $R(q_3) = (R_x, R_y - 1)$ that are counted from o, where $o \in O$. The difference of the SED of p and q_3 is

$$h(p, q_3) = op^2 - oq_3^2 = [R_x^2 + R_y^2] - [R_x^2 + (R_y - 1)^2] = 2R_y - 1 = 2(R_y - 1) + 1.$$

Since $R_y(q_3) = (R_y - 1)$ and $G(p, q_3) = (0, 1)$, we obtain $h(p, q_3) = 2R_y(q_3) + 1$ and $R(p) = (R_x(q_3), R_y(q_3) + 1)$.

Case 4. The smallest SED is obtained from q_4.

Assume $q_4 \in Q$. Let the relative coordinates of q_4 be $R(q_4) = (R_x + 1, R_y - 1)$ that are counted from o, where $o \in O$. In this case, there exists $R(q_3) = (R_x, R_y - 1)$ which is closer to o than q_4 and derive a smaller SEDT to p. Therefore, q_4 is useless for calculating SEDT of p.

4.1.2. The nearest background pixel located on the right-top side

In Section 4.1.1, we have proved the equations for achieving EDT in a forward raster scan if the nearest background pixel is located on the left-top side of the object pixel p. On the other hand, if the nearest background is located on the right-top side of the object pixel as shown in Fig. 4, the relative-coordinate vectors of q_1, q_2, q_3, and q_4 are $R(q_1) = (R_x + 1, R_y), R(q_2) = (R_x + 1, R_y - 1), R(q_3) = (R_x, R_y - 1)$, and $R(q_4) = (R_x - 1, R_y - 1)$, respectively.

Since $oq_1^2 = (R_x + 1)^2 + R_y^2$ is larger than $op^2 = R_x^2 + R_y^2$, it is obviously that q_1 cannot derive the SED of p. The relative-coordinates vector of q_2 is $R(q_2) = (R_x + 1, R_y - 1)$ that is the same as q_4 in Section 4.1.1. Therefore, q_2 is useless in this condition. The relative-coordinates vector of q_3 is $R(q_3) = (R_x, R_y - 1)$ that is the same as in Section 4.1.1. Therefore, we can derive SED of p by the same equation.

Now, considering pixel q_4, the relative coordinates of q_4 is $R(q_4) = (R_x - 1, R_y - 1)$ that is the same as q_2 in Section 4.1.1. Therefore, the difference of the SED of p and q_4 is

$$h(p, q_4) = op^2 - oq_4^2 = 2((R_x - 1) + (R_y - 1) + 1).$$

Since $R_x(q_4) = (R_x - 1)$, $R_y(q_4) = (R_y - 1)$, and $G(p, q_4) = (1, 1)$, we obtain $h(p, q_4) = 2(R_x(q_2) + R_y(q_2) + 1)$ and $R(p) = (R_x(q_4) + 1, R_y(q_4) + 1)$.

Go over the main points in Sections 4.1.1 and 4.1.2, no matter the background pixel is located on the left-top side or right-top side in the forward raster scan procedure, the SED of p can be obtained by its four neighborhoods, q_1, q_2, q_3, and q_4

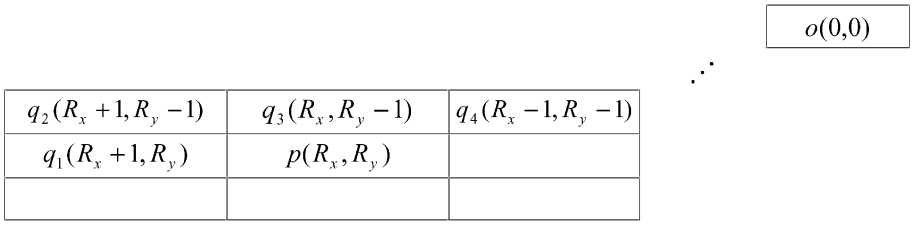

Fig. 4. The four cases when o is located on the right-top side of p.

using Eqs. (4) and (5). In the same way, when processing a backward raster scan, the SED of p can also be obtained by another four neighborhoods, $q_5, q_6, q_7,$ and q_8 using Eqs. (4) and (5).

4.2. The proof of achieving the minimum SEDT

The two-scan algorithm by using a 3×3 neighborhood can be proved by mathematical induction. Suppose l indicates the *chessboard distance* of the background pixel o. For example, in Fig. 5, if $l = 1$, the concerned pixels are the eight neighborhoods of o, that is, the first layer from o. Note that, only one background pixel o in Fig. 5.

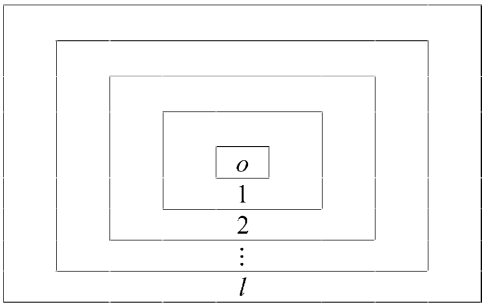

Fig. 5. The representation of object pixels and o.

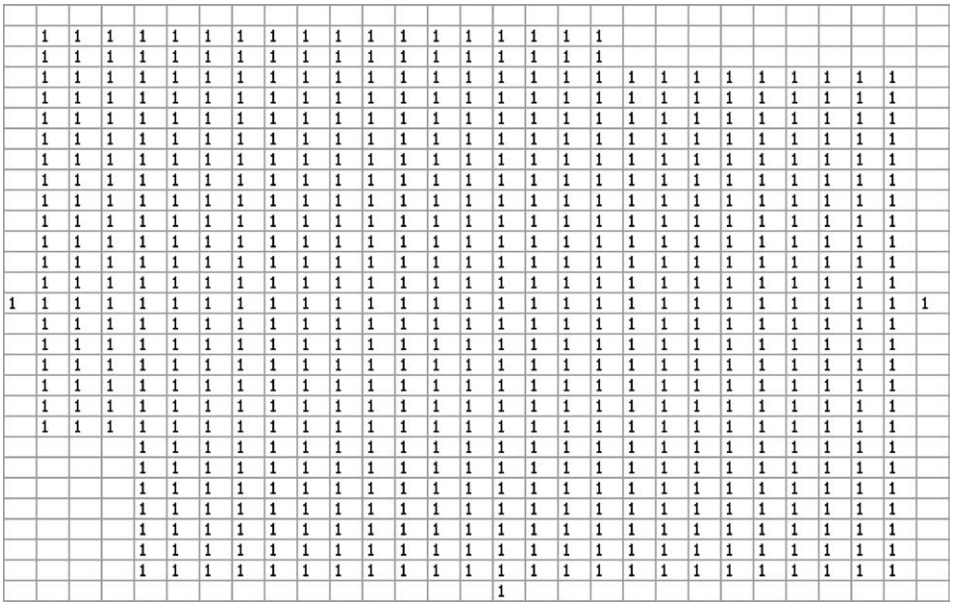

Fig. 6. The difficult case for EDT.

A. *Base case*. When $l = 1$. It is obviously that the minimum SED of the eight neighborhoods of the background pixel *o* can be obtained.

B. *Induction hypothesis*. When $l = k$. An assumption is made that the object pixels located on layer *k* can have their minimum SEDs from those pixels on layer $k - 1$.

A

	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1										
	1	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	2										
	1	4	9	9	9	9	9	9	9	9	9	9	9	9	9	9	8	2	1	1	1	1	1	1	1	1	1	
	1	4	9	16	16	16	16	16	16	16	16	16	16	16	16	16	8	5	4	4	4	4	4	4	4	4	2	
	1	4	9	16	25	25	25	25	25	25	25	25	25	25	25	18	13	10	9	9	9	9	9	9	9	8	2	
	1	4	9	16	25	36	36	36	36	36	36	36	36	36	32	25	20	17	16	16	16	16	16	16	16	8	2	
	1	4	9	16	25	36	49	49	49	49	49	49	49	49	41	34	29	26	25	25	25	25	25	25	18	8	2	
	1	4	9	16	25	36	49	64	64	64	64	64	64	61	52	45	40	37	36	36	36	36	36	32	18	8	2	
	1	4	9	16	25	36	49	64	81	81	81	81	81	74	65	58	53	50	49	49	49	49	49	32	18	8	2	
	1	4	9	16	25	36	49	64	81	100	100	100	100	89	80	73	68	65	64	64	64	64	50	32	18	8	2	
	1	4	9	16	25	36	49	64	81	100	121	121	117	106	97	90	85	82	81	81	81	72	50	32	18	8	2	
	1	4	9	16	25	36	49	64	81	100	121	144	136	125	116	109	104	101	100	100	98	72	50	32	18	8	2	
	1	4	9	16	25	36	49	64	81	100	121	144	157	146	137	130	125	122	121	121	98	72	50	32	18	8	2	
1	2	5	10	17	26	37	50	65	82	101	122	145	170	169	160	153	148	145	144	128	98	72	50	32	18	8	2	1
	1	4	9	16	25	36	49	64	81	100	121	144	169	194	185	178	173	170	162	128	98	72	50	32	18	8	5	
	1	4	9	16	25	36	49	64	81	100	121	144	169	196	212	205	200	197	162	128	98	72	50	32	18	13	2	
	1	4	9	16	25	36	49	64	81	100	121	144	169	196	225	234	229	200	162	128	98	72	50	32	25	8	2	
	1	4	9	16	25	36	49	64	81	100	121	144	169	196	225	256	242	200	162	128	98	72	50	41	18	8	2	
	1	4	9	16	25	36	49	64	81	100	121	144	169	196	225	256	242	200	162	128	98	72	61	32	18	8	2	
	1	4	9	16	25	36	49	64	81	100	121	144	169	196	225	256	242	200	162	128	98	85	50	32	18	8	2	
				1	4	9	16	25	36	49	64	81	100	121	144	169	196	200	162	128	113	72	50	32	18	8	2	
				1	4	9	16	25	36	49	64	81	100	121	144	169	196	200	162	145	98	72	50	32	18	8	2	
				1	4	9	16	25	36	49	64	81	100	121	144	169	196	200	181	128	98	72	50	32	18	8	2	
				1	4	9	16	25	36	49	64	81	100	121	144	169	196	221	162	128	98	72	50	32	18	8	2	
				1	4	9	16	25	36	49	64	81	100	121	144	169	196	200	162	128	98	72	50	32	18	8	2	
				1	4	9	16	25	36	49	64	81	100	121	144	169	196	200	162	128	98	72	50	32	18	8	2	
				1	4	9	16	25	36	49	64	81	100	121	144	169	196	200	162	128	98	72	50	32	18	8	2	
															1													

B

	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1										
	1	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	1										
	1	4	9	9	9	9	9	9	9	9	9	9	9	9	9	9	5	2	1	1	1	1	1	1	1	1	1	
	1	4	9	16	16	16	16	16	16	16	16	16	16	16	16	13	8	5	4	4	4	4	4	4	4	4	1	
	1	4	9	16	25	25	25	25	25	25	25	25	25	25	25	18	13	10	9	9	9	9	9	9	9	4	1	
	1	4	9	16	25	36	36	36	36	36	36	36	36	36	32	25	20	17	16	16	16	16	16	16	9	4	1	
	1	4	9	16	25	36	49	49	49	49	49	49	49	49	41	34	29	26	25	25	25	25	25	16	9	4	1	
	1	4	9	16	25	36	49	64	64	64	64	64	64	61	52	45	40	37	36	36	36	36	25	16	9	4	1	
	1	4	9	16	25	36	49	64	81	81	81	81	81	74	65	58	53	50	49	49	49	36	25	16	9	4	1	
	1	4	9	16	25	36	49	64	81	100	100	100	100	89	80	73	68	65	64	64	49	36	25	16	9	4	1	
	1	4	9	16	25	36	49	64	81	100	121	121	117	106	97	90	85	82	81	64	49	36	25	16	9	4	1	
	1	4	9	16	25	36	49	64	81	100	121	144	136	125	116	109	104	100	81	64	49	36	25	16	9	4	1	
	1	4	9	16	25	36	49	64	81	100	121	144	157	146	137	130	121	100	81	64	49	36	25	16	9	4	1	
1	2	5	10	17	26	37	50	65	82	98	113	130	149	169	160	145	122	101	82	65	50	37	26	17	10	5	2	1
	1	4	9	16	25	36	49	61	72	85	100	117	136	157	169	144	121	100	81	64	49	36	25	16	9	4	1	
	1	4	9	16	25	34	41	50	61	74	89	106	125	144	145	144	121	100	81	64	49	36	25	16	9	4	1	
	1	4	9	16	20	25	32	41	52	65	80	97	116	121	122	121	121	100	81	64	49	36	25	16	9	4	1	
	1	4	9	10	13	18	25	34	45	58	73	90	100	100	101	100	100	100	81	64	49	36	25	16	9	4	1	
	1	4	4	5	8	13	20	29	40	53	68	81	81	81	82	81	81	81	81	64	49	36	25	16	9	4	1	
	1	1	1	2	5	10	17	26	37	50	64	64	64	64	65	64	64	64	64	64	49	36	25	16	9	4	1	
				1	4	9	16	25	36	49	49	49	49	49	50	49	49	49	49	49	49	36	25	16	9	4	1	
				1	4	9	16	25	36	36	36	36	36	36	37	36	36	36	36	36	36	36	25	16	9	4	1	
				1	4	9	16	25	25	25	25	25	25	25	26	25	25	25	25	25	25	25	25	16	9	4	1	
				1	4	9	16	16	16	16	16	16	16	16	17	16	16	16	16	16	16	16	16	16	9	4	1	
				1	4	9	9	9	9	9	9	9	9	9	10	9	9	9	9	9	9	9	9	9	9	4	1	
				1	4	4	4	4	4	4	4	4	4	4	5	4	4	4	4	4	4	4	4	4	4	4	1	
				1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	
															1													

Fig. 7. The example of the two-scan algorithm (A) Forward, (B) Backward and that is the result of SEDT.

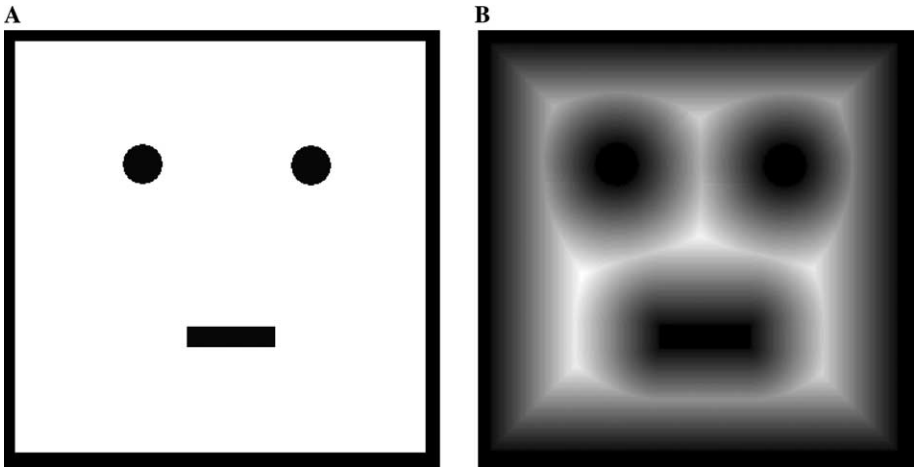

Fig. 8. The EDT on a real image.

C. Induction case. When $l = k + 1$. From inductive hypothesis we know that the object pixels located on layer k have already reached the minimum SED. Now all the object pixels located on layer $k + 1$ can be categorized according to their locations into four classes: 0°, 45°, 90°, and 135°. In both forward and backward raster scans, the minimum SED of those pixels on layer k can be obtained by using Eqs. (4) and (5). Therefore, the two-scan algorithm by using a 3×3 neighborhood can achieve the SEDT correctly.

5. Some examples of the two-scan algorithm by using a 3×3 neighborhood

This Section shows some examples when achieving EDT by our technique. In order to analyze any difficult case of EDT, the special object, as shown in Fig. 6, is created for EDT.

Fig. 7 shows the examples of the two-scan algorithm by using a 3×3 neighborhood. Figs. 7A and B show the forward and backward raster-scans in the direction of left-to-right top-to-bottom and right-to-left bottom-to-top, respectively.

Fig. 8 shows an example when achieving EDT on an image. Fig. 8A is a 400×400 image in which white and black pixels are object and background pixels, respectively. Fig. 8B is the result of EDT of Fig. 8A.

6. Conclusions

Although the method proposed by Cuisenaire and Macq [1] can achieve the EDT in a short time, the propagation using multiple neighborhoods and bucket sorting are the overheads of their algorithm. In this paper, a new, simple and efficient

two-scan algorithm by using a 3×3 neighborhood to compute the exact EDT is presented. We do not need to use multiple neighborhoods in order to achieve exact EDT, but also do not need extra pre-processing. The exact EDT can be obtained in two raster scans under a 3×3 neighborhood by recording the relative coordinates.

References

[1] O. Cuisenaire, B. Macq, Fast Euclidean distance transformation by propagation using multiple neighborhoods, Comp. Vis. Image Understand. 76 (2) (1999) 163–172.
[2] L. Vincent, Exact Euclidean distance function by chain propagations, in: Proc. IEEE Comp. Vis. Pattern Recog., Hawaii (1991) 520–525.
[3] F.Y. Shih, O.R. Mitchell, A mathematical morphology approach to Euclidean distance transformation, IEEE Trans. Image Process. 1 (2) (1992) 197–204.
[4] F.Y. Shih, H. Wu, Optimization on Euclidean distance transformation using grayscale morphology, J. Vis. Commun. Image Represent. 3 (2) (1992) 104–114.
[5] C.T. Huang, O.R. Mitchell, A Euclidean distance transform using grayscale morphology decomposition, IEEE Trans. Pattern Anal. Mach. Intell. 16 (4) (1994) 443–448.
[6] G. Borgefors, Distance transformations in arbitrary dimensions, Comput. Vis., Graph., Image Process. 27 (1984) 321–345.
[7] P.E. Danielsson, Euclidean distance mapping, Comput. Graph. Image Process. 14 (1980) 227–248.
[8] A. Rosenfeld, A.C. Kak, Digital Picture Processing, Academic Press, 1982.
[9] F.Y. Shih, O.R. Mitchell, Decomposition of gray scale morphological structuring elements, Pattern Recog. 24 (3) (1991) 195–203.
[10] F.Y. Shih, O.R. Mitchell, Threshold decomposition of gray-scale morphology into binary morphology, IEEE Trans. Pattern Anal. Machine Intell. 11 (1) (1989) 31–42.
[11] R.M. Haralick, S.R. Sternberg, X. Zhuang, Image analysis using mathematical morphology, IEEE Trans. Pattern Anal. Mach. Intell. 9 (4) (1987) 532–550.
[12] J. Serra, Image Analysis and Mathematical Morphology, Academic, New York, 1982.
[13] F.Y. Shih, C.T. King, C.C. Pu, Pipeline architectures for recursive morphological operations, IEEE Trans. Image Process. 4 (1) (1995) 11–18.