Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Vision and Image Understanding 95 (2004) 317–333
Computer Vision and Image Understanding
www.elsevier.com/locate/cviu

The "dead reckoning" signed distance transform

George J. Grevera*

Medical Image Processing Group, Department of Radiology, University of Pennsylvania, 4th Floor Blockley Hall, 423 Guardian Drive, Philadelphia, PA 19104-6021, USA

Received 3 February 2003; accepted 17 May 2004
Available online 7 July 2004

Abstract

Consider a binary image containing one or more objects. A signed distance transform assigns to each pixel (voxel, etc.), both inside and outside of any objects, the minimum distance from that pixel to the nearest pixel on the border of an object. By convention, the sign of the assigned distance value indicates whether or not the point is within some object (positive) or outside of all objects (negative). Over the years, many different algorithms have been proposed to calculate the distance transform of an image. These algorithms often trade accuracy for efficiency, exhibit varying degrees of conceptual complexity, and some require parallel processors. One algorithm in particular, the Chamfer distance [J. ACM 15 (1968) 600, Comput. Vis. Graph. Image Process. 34 (1986) 344], has been analyzed for accuracy, is relatively efficient, requires no special computing hardware, and is conceptually straightforward. It is understandably, therefore, quite popular and widely used. We present a straightforward modification to the Chamfer distance transform algorithm that allows it to produce more accurate results without increasing the window size. We call this new algorithm *Dead Reckoning* as it is loosely based on the concept of continual measurements and course correction that was employed by ocean going vessel navigation in the past. We compare Dead Reckoning with a wide variety of other distance transform algorithms based on the Chamfer distance algorithm for both accuracy and speed, and demonstrate that Dead Reckoning produces more accurate results with comparable efficiency.

Keywords: Signed distance transform; Chamfer distance; Euclidean distance

* Fax: 1-215-898-9145.
E-mail address: grevera@mipg.upenn.edu.

1077-3142/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2004.05.002

1. Introduction

Given a binary image consisting of one or more objects and a (possibly disjoint) background, we define a signed distance transform as a transform that assigns to every point (to both those in objects as well as those in the background) the minimum distance from that particular point to the nearest point on the border of an object. The sign of the assigned distance value indicates whether the point is either inside (positive) or outside (negative) objects. Many distance transform algorithms have been proposed with [18] and [25] most likely being the earliest. In general, distance transform algorithms exhibit varying degrees of accuracy of the result, computational complexity, hardware requirements (such as parallel processors), and conceptual complexity of the algorithms themselves. In [7], the author proposed an algorithm that produces extremely accurate results by propagating vectors which approximate the distance of 2D images by sweeping through the data a number of times by propagating a local mask in a manner similar to convolution. In [1], the author presented the Chamfer distance algorithm (CDA) that propagates scalar, integer values to efficiently and accurately calculate the distance transform of 2D and 3D images (again in a manner similar to convolution). Borgefors [1] also presented an error analysis for the CDA for various neighborhood sizes and integer values. More recently [2], an analysis of 3D distance transforms employing $3 \times 3 \times 3$ neighborhoods of local distances was presented. In [3], an analysis of the 2D Chamfer distance algorithm using 3×3, 5×5, and larger neighborhoods employing both integer and real values was presented. Marchand-Maillet and Sharaiha [26] also present an analysis of Chamfer distance using topological order as opposed to the approximation to the Euclidean distance as the evaluation criteria. Because of the conceptual elegance of the CDA and because of its widespread popularity, its improvement is the motivation for this work.

Of course, distance transforms outside of the Chamfer family also have been presented. A technique from Artificial Intelligence, namely A* heuristic search, has been used as the basis for a distance transform algorithm [27]. A multiple pass algorithm using windows of various configurations (along the lines of [7] and other raster scanning algorithms such as the CDA) was presented in [28] and [34]. A method of distance assignment called ordered propagation was presented in [29]. The basis of this algorithm and others such as A* (used in [27]) is to propagate distance between pixels can be represented as nodes in a graph. These algorithms typically employ sorted lists to order the propagation among the graph nodes. Guan and Ma [30] and Eggers [31] employ lists as well. In [35], the authors present four algorithms to perform the exact, Euclidean, n-dimensional distance transform via the serial composition of n-dimensional filters. Algorithms for the efficient computation of distance transforms using parallel architectures are presented in [32] and [36]. In [36], the authors present an algorithm that consists of two phases with each phase consisting of both a forward scan and a backward scan. In the first phase, columns are scanned; in the second phase, rows are scanned. They note that since the scanning of a particular column (or row) is independent of the scanning of the other columns (or rows), each column (row) may be scanned independently (i.e., in parallel). A distance transform

employing a graph search algorithm is also presented in [33]. Distance transforms continue to be interesting and have also been the focus of at least one recent Ph.D. dissertation [4].

Since the early formulation of distance transform algorithms [18,25], applications employing distance transforms have become widespread. For example, distance transforms have been used for skeletonization of images [6,14,19,21]. Distance transforms are also useful for the (shape-based) interpolation of both binary images [11,15] as well as gray image data [8]. In [12], the authors employ distance transform information in multidimensional image registration. An efficient ray tracing algorithm also employs distance transform information [13]. Distance transforms have also been shown to be useful in calculating the medial axis transform with [16,17] employing the Chamfer distance algorithm specifically. In addition to the usefulness of distance transforms for the interpolation of 3D gray medical image data [9,10], they have also been used for the automatic classification of plant cells [22] and for measuring cell walls [24]. The Chamfer distance was also employed in a method to characterize spinal cord atrophy [20]. Because distance transforms are applicable to such a wide variety of problems, it is important to develop accurate and efficient distance transform algorithms.

The outline of this paper is as follows. First, we present a few definitions from digital topology [23] that will be useful for developing the framework of the algorithm. Then we present the Chamfer distance algorithm and compare and contrast it with the new Dead Reckoning method. To evaluate this new method, we compare it with the Chamfer distance algorithm employing various window sizes (3×3, 5×5, and 7×7) and types (Chamfer, city block, chessboard, and Euclidean with a 3×3 window). An evaluation framework is then presented. This framework evaluates the algorithms with respect to execution time, a quantitative evaluation, and a qualitative evaluation. The qualitative evaluation demonstrates the presence of polygonal isocontours in all of the Chamfer-based distance transform algorithms except Dead Reckoning. Given known and random binary images, the quantitative evaluation demonstrates that the Dead Reckoning algorithm produces the most accurate results with respect to the actual, exact Euclidean distance assignment.

2. Definitions and methods

We first present the CDA for completeness and to demonstrate the similarities and differences between it and the Dead Reckoning algorithm (DRA). Note that CDA has been and DRA may be extended to distance transforms in higher dimensional spaces but we restrict our discussion and analysis to 2D for simplicity. A complete outline of the CDA appears in Fig. 1. A 2D binary input image, I, having X columns and Y rows is given as input to the CDA. Since I is a binary image, for a given point, $C = (x, y)$, either $I(x, y) = 0$ indicating a point outside of any object or $I(x, y) = 1$ indicating a point within an object. Note that the input image may consist of more than one object (and we assume that it contains at least one object). Furthermore, we assume that no object extends to the border of I. More formally,

```
Algorithm: Chamfer
Inputs: I - a 2D binary image of size X by Y
        d1 - distance between two adjacent pixels in either the x or y direction
        d2 - diagonal distance between two diagonally adjacent pixels

Output: d - a 2D grey image of size X and Y representing the distance image

//initialize d
for y = 1 to Y do begin
    for x = 1 to X do begin
        d(x,y) = ∞
    end
end

//initialize immediate interior & exterior elements
for y = 1 to Y do begin
    for x = 1 to X do begin
        //if ( I(x,y) ) then  //uncomment to disable symmetry under complement property
        if ( I(x-1,y) != I(x,y) or I(x+1,y) != I(x,y) or
             I(x,y-1) != I(x,y) or I(x,y+1) != I(x,y) ) then    d(x,y) = 0
    end
end

//perform the first (forward) pass
for y = 1 to Y do begin
    for x = 1 to X do begin
        if (d(x-1,y-1)+d2 < d(x,y)) then     d(x,y) = d(x-1,y-1)+d2
        if (d(x,y-1)  +d1 < d(x,y)) then     d(x,y) = d(x,y-1)  +d1
        if (d(x+1,y-1)+d2 < d(x,y)) then     d(x,y) = d(x+1,y-1)+d2
        if (d(x-1,y)  +d1 < d(x,y)) then     d(x,y) = d(x-1,y)  +d1
    end
end

//perform the final (backward) pass
for y = Y to 1 do begin
    for x = X to 1 do begin
        if (d(x+1,y)  +d1 < d(x,y)) then    d(x,y) = d(x+1,y)  +d1
        if (d(x-1,y+1)+d2 < d(x,y)) then    d(x,y) = d(x-1,y+1)+d2
        if (d(x,y+1)  +d1 < d(x,y)) then    d(x,y) = d(x,y+1)  +d1
        if (d(x+1,y+1)+d2 < d(x,y)) then    d(x,y) = d(x+1,y+1)+d2
    end
end

//indicate inside & outside
for y = Y to 1 do begin
    for x = X to 1 do begin
        if (I(x,y) == 0) then    d(x,y) = -d(x,y)
    end
end
```

Fig. 1. Original Borgefors' Chamfer distance algorithm using a 3×3 window. Typically, $d1 = 3$ and $d2 = 4$. The sections appearing in bold in the first and second pass would be modified to accommodate larger windows. (We note that the two sets of initialization loops may be combined into a single loop for a more efficient implementation.)

$\forall\, xI(x,0) = 0 \wedge \forall\, yI(0,y) = 0 \wedge \forall\, xI(x,Y-1) = 0 \wedge \forall\, yI(X-1,y)$. (If this is not the case, we simply embed I of size $X \times Y$ in I' of size $X + 2 \times Y + 2$). The output will be a gray image, d, also of size $X \times Y$ where the value assigned to a point in the output image represents the distance from that point in the binary image to the nearest point on the border of an object. Using terminology from digital topology [23] we call a border point $b = (x,y)$ an element of the immediate interior, II, iff

$$[I(x,y) = 1] \wedge [I(x+1,y) = 0 \vee I(x-1,y) = 0 \vee I(x,y+1) = 0 \vee I(x,y-1) = 0]. \tag{1}$$

This indicates that for any point to be an element of the immediate interior it must be in some object and at least one of its 4-connected neighbors (i.e., two adjacent pixels with either x coordinates or y coordinates that differ by *exactly* one) must be outside of any object. Similarly, we call a border point $b' = (x, y)$ an element of the immediate exterior, *IE*, iff

$$[I(x,y) = 0] \wedge [I(x+1,y) = 1 \vee I(x-1,y) = 1 \vee I(x,y+1) = 1 \vee I(x,y-1) = 1]. \tag{2}$$

The algorithm proceeds as follows. Similar to [7], the CDA initially assigns $d(x, y) = \infty$ for all points. The next step is to assign 0 to all points belonging to either the *II* or the *IE*. We note that some distance transform algorithms including [7] restrict the definition of border point to elements of *II* only. Our algorithm easily accommodates this (via one line in the initialization of the *II* step in Figs. 1 and 3). If, however, border points consist of $II \cup IE$, the resulting distance transform exhibits the property of symmetry under complement. This means that a distance value assigned to a pixel will be the same regardless of whether or not the pixel is inside or outside of any objects. To illustrate this point, consider a binary image I and its complement $\bar{I}$, where

$$\bar{I}(x,y) = \begin{cases} 0 & \text{if } I(x,y) = 1 \\ 1 & \text{otherwise} \end{cases}.$$

A distance transform with the symmetry under complement property produces the same results regardless of whether I or $\bar{I}$ is used as input. Again, as illustrated in Figs. 1 and 3, our algorithm easily accommodates both definitions.

Then two passes, one forward and one backward, are made through the image data. Each pass employs local (typically 3×3, 5×5, or 7×7) neighborhood operations (roughly analogous to convolution) in which one attempts to minimize the current distance value assigned to the pixel, $C = (x, y)$, at the center of the window by comparing the current distance value with the distance values assigned to its neighbor, n, plus the distance from C to the given neighbor, n, as specified in the window. Various windows are shown in Fig. 2.

For example, consider the forward pass 3×3 window, w, as shown in Fig. 2. Let $w(0, 0)$ indicate the value of the center of the 3×3 window which is centered over the point $C = (x, y)$ in I and d. Let $d(x, y)$ be the current distance to a border point. The algorithm then makes the following assignment:

$$d(x,y) = \min \left\{ \begin{array}{lll} d(x-1,y-1)+w(-1,-1), & d(x,y-1)+w(0,-1), & d(x+1,y-1)+w(1,-1), \\ d(x-1,y)+w(-1,0), & d(x,y) & \end{array} \right\}. \tag{3}$$

The forward window is moved through d in a forward pass. Then a backward window is moved through d in a backward pass to propagate minimum distances throughout d. As in [15] and [34], we adopt the convention of $d(x, y) > 0$ indicating a point within an object, i.e., $I(x, y) = 1$ and $d(x, y) < 0$ indicating a point outside of any object ($I(x, y) = 0$).

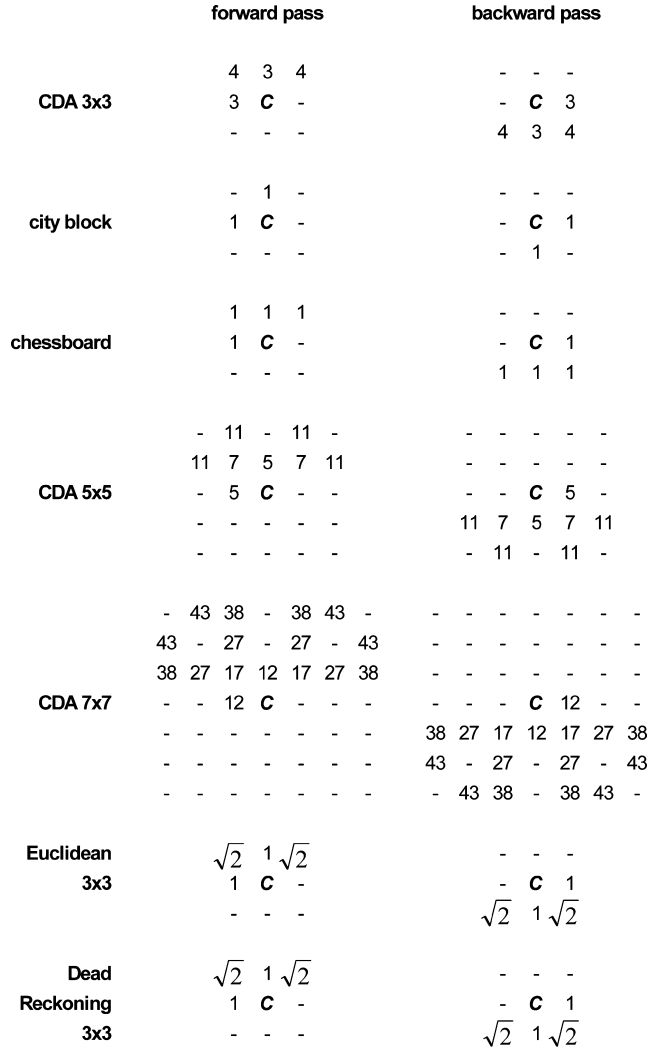

Fig. 2. Various windows used by the Chamfer Distance algorithm. Euclidean 3×3 is included for comparison, as is the 3×3 window employed by the Dead Reckoning algorithm. '*C*' indicates the center of the window; '-' indicates a point that is not used.

Borgefors cleverly demonstrated: (i) using a small window and propagating distance in this manner introduces errors in the assigned distance values even if double precision floating point is used to represent distance values, (ii) these errors may be minimized by using values other than 1 and $\sqrt{2}$ for the distances between neighboring pixels, and surprisingly, (iii) using integer window values such as 3 and 4 yields more accurate results than using window values of 1 and $\sqrt{2}$ and does so with much better performance, and (iv) larger windows with appropriate values minimize errors even further at increased computational cost. Although, as in our algorithm as well,

```
Algorithm: Dead Reckoning
Inputs: I - a 2D binary image of size X by Y
        d1 - distance between two adjacent pixels in either the x or y direction
        d2 - diagonal distance between two diagonally adjacent pixels

Output: d - a 2D grey image of size X and Y representing the distance image
        P - for each pixel, the corresponding border point

//initialize d

for y = 1 to Y do begin
    for x = 1 to X do begin
        d(x,y) = ∞
        p(x,y) = -1
    end
end

//initialize immediate interior & exterior elements

for y = 1 to Y do begin
    for x = 1 to X do begin
        //if ( I(x,y) ) then  //uncomment to disable symmetry under complement property
        if ( I(x-1,y) != I(x,y) or I(x+1,y) != I(x,y) or
             I(x,y-1) != I(x,y) or I(x,y+1) != I(x,y) ) then begin
            d(x,y) = 0
            p(x,y) = (x,y)
        end
    end
end

//perform the first pass

for y = 1 to Y do begin
    for x = 1 to X do begin
        if (d(x-1,y-1)+d2 < d(x,y)) then begin
            p(x,y) = p(x-1,y-1)
            d(x,y) = sqrt( (x-p(x,y).x)*(x-p(x,y).x) +
                           (y-p(x,y).y)*(y-p(x,y).y) )
        end
        if (d(x,y-1)  +d1 < d(x,y)) then begin
            p(x,y) = p(x,y-1)
            d(x,y) = sqrt( (x-p(x,y).x)*(x-p(x,y).x) +
                           (y-p(x,y).y)*(y-p(x,y).y) )
        end
        if (d(x+1,y-1)+d2 < d(x,y)) then begin
            p(x,y) = p(x+1,y-1)
            d(x,y) = sqrt( (x-p(x,y).x)*(x-p(x,y).x) +
                           (y-p(x,y).y)*(y-p(x,y).y) )
        end
        if (d(x-1,y)  +d1 < d(x,y)) then begin
            p(x,y) = p(x-1,y)
            d(x,y) = sqrt( (x-p(x,y).x)*(x-p(x,y).x) +
                           (y-p(x,y).y)*(y-p(x,y).y) )
        end
    end
end
```

Fig. 3. The Dead Reckoning algorithm (using only a 3×3 window). Sections in bold indicate areas that differ from the Chamfer Distance algorithm. (We note that the two sets of initialization loops may be combined into a single loop for a more efficient implementation.)

the computational complexity remains the same, $O(X * Y)$, even for larger and larger window sizes.

The DRA on the other hand is a straightforward modification to the CDA that, employing equal sized windows, produces more accurate results at a slightly increased computational cost. Furthermore, DRA using only a 3×3 window typically

```
//perform the final pass

for y = Y to 1 do begin
    for x = X to 1 do begin
        if (d(x+1,y)  +d1 < d(x,y)) then begin
            p(x,y) = p(x+1,y)
            d(x,y) = sqrt( (x-p(x,y).x)*(x-p(x,y).x) +
                           (y-p(x,y).y)*(y-p(x,y).y) )
        end
        if (d(x-1,y+1)+d2 < d(x,y)) then begin
            p(x,y) = p(x-1,y+1)
            d(x,y) = sqrt( (x-p(x,y).x)*(x-p(x,y).x) +
                           (y-p(x,y).y)*(y-p(x,y).y) )
        end

        if (d(x,y+1)  +d1 < d(x,y)) then begin
            p(x,y) = p(x,y+1)
            d(x,y) = sqrt( (x-p(x,y).x)*(x-p(x,y).x) +
                           (y-p(x,y).y)*(y-p(x,y).y) )
        end
        if (d(x+1,y+1)+d2 < d(x,y)) then begin
            p(x,y) = p(x+1,y+1)
            d(x,y) = sqrt( (x-p(x,y).x)*(x-p(x,y).x) +
                           (y-p(x,y).y)*(y-p(x,y).y) )

        end
    end
end

//indicate inside & outside
for y = Y to 1 do begin
    for x = X to 1 do begin
        if (I(x,y) == 0) then    d(x,y) = -d(x,y)
    end
end
```

Fig. 3. (*continued*)

produces more accurate results (see Table 5) than CDA with a 7×7 window (with similar execution times, see Table 2). In addition to d which for a given point, (x, y), is the minimum distance from (x, y) to the nearest border point, the DRA introduces an additional data structure, p, which is used to indicate the actual border point, $p(x, y) = b$ such that $b \in II \cup IE$ and $d(x, y)$ is minimum similar to that employed Danielsson in [7]. (Danielsson [7] in 4SED employs 3 minimization iterations in both the forward and backward passes. Our method as in the CDA [1,25] employs only 1 in each pass). Note that as the CDA progresses, $d(x, y)$ may be updated many times. In the DRA, each time that $d(x, y)$ is updated, $p(x, y)$ is updated as well. We note that the order in which the *if* statements in Fig. 3 are evaluated may influence the assignment of $p(x, y)$ and subsequently, the value assigned to $d(x, y)$. Regardless, our results demonstrate that our algorithm remains more accurate using only a 3×3 neighborhood than CDA using a 7×7 neighborhood. Although the DRA employs a 3×3 (or larger) window to guide the update/minimization of distance process as

does CDA, the actual values assigned to d are not the same as CDA as shown in Eq. (3). Let $p_x(x,y)$ denote the x component of b, and $p_y(x,y)$ denote the y component of b. DRA uses instead the actual Euclidean distance from the border to the point (x,y) at the center of the window as shown in Eq. (4).

Using only a 3×3 window, the DRA typically determines a more accurate estimation of the exact Euclidean distance within the framework of the CDA. Details of the DRA are shown in Fig. 3.

$$d(x,y) = \min \left\{ \begin{array}{l} \sqrt{(p_x(x-1,y-1)-x)^*(p_x(x-1,y-1)-x)+(p_y(x-1,y-1)-y)^*(p_y(x-1,y-1)-y)}, \\ \sqrt{(p_x(x,y-1)-x)^*(p_x(x,y-1)-x)+(p_y(x,y-1)-y)^*(p_y(x,y-1)-y)}, \\ \sqrt{(p_x(x+1,y-1)-x)^*(p_x(x+1,y-1)-x)+(p_y(x+1,y-1)-y)^*(p_y(x+1,y-1)-y)}, \\ \sqrt{(p_x(x-1,y+1)-x)^*(p_x(x-1,y+1)-x)+(p_y(x-1,y+1)-y)^*(p_y(x-1,y-1)+y)}, \\ d(x,y) \end{array} \right\}. \tag{4}$$

3. Results and discussion

We compared DRA with other distance transform algorithms, CDA 3×3, city block, chessboard, CDA 5×5, CDA 7×7, and Euclidean 3×3, on the basis of both execution speed and accuracy of the resulting distance transforms for known images.

3.1. Execution times

To determine execution speeds, we compiled and executed all programs on a 2 GHz Pentium 4 based Dell Precision 340 with 1 Gb of RAM running under Red Hat Linux release 7.1. All algorithms were implemented in C++ and were compiled with g++ version 2.96 using the –O3 option for maximum optimization for speed. Test images consisted of a number of input binary images of various sizes containing a single object point at the center of each image. For input test images of sizes less than 5000×5000, execution times were averaged over 100 iterations. For the 5000×5000 image, execution times were averaged over 10 iterations (we define an iteration to be one complete execution of a distance transform algorithm). Execution times appear in Tables 1 and 2. For those distance transform windows that use an integer representation for distance (viz., CDA 3×3, city block, chessboard, CDA 5×5, and CDA 7×7), an optional normalization step may be performed to convert non-unit adjacent window values to unit 1 as shown in Eq. (5).

This allows us to compare calculated distance values with actual, known Euclidean distance values for test images.

$$d'(x,y) = \frac{d(x,y)}{\min\{w(i,j) | w(i,j) > 0 \wedge w(i,j) < \infty\}}. \tag{5}$$

Execution times with and without this conversion are reported in Table 1, and are included in all of the times in Table 2. We note that although city block and chessboard use an integer representation, they do not require normalization since

Table 1
Results of timing comparison for various distance transform algorithms applied to test images of sizes 1000×1000 and 5000×5000

Algorithm	Representation	Window size	1000×1000		5000×5000	
			Normalization	No normalization	Normalization	No normalization
CDA 3×3	Integer	3×3	0.09	0.06	2.30	1.50
City block	Integer	3×3	0.08	0.05	2.01	1.19
Chessboard	Integer	3×3	0.09	0.06	2.34	1.47
CDA 5×5	Integer	5×5	0.12	0.09	3.02	2.19
CDA 7×7	Integer	7×7	0.18	0.14	4.42	3.62
Euclidean 3×3	Double	3×3	0.09		2.22	
Dead Reckoning 3×3	Double	3×3	0.15		3.94	
Dead Reckoning 7×7	Double	7×7	0.26		6.86	
4SED	Double	3×3	0.12		3.11	
8SED	Double	3×3	0.15		3.88	
8SED (improved)	Double	3×3	0.13		3.38	

All input images consisted of a solitary point at the center.

Table 2
Time in seconds to perform one complete 2D distance transform for various image sizes and algorithms

Algorithm	Image size			
	256×256	512×512	1000×1000	5000×5000
CDA 3×3	0.01	0.02	0.09	2.30
City block	0.01	0.02	0.08	2.01
Chessboard	0.01	0.02	0.09	2.34
CDA 5×5	0.01	0.03	0.12	3.02
CDA 7×7	0.01	0.05	0.18	4.42
Euclidean 3×3	0.01	0.02	0.09	2.22
Dead Reckoning 3×3	0.01	0.05	0.15	3.94
Dead Reckoning 7×7	0.02	0.08	0.26	6.86
4SED	0.01	0.04	0.12	3.11
8SED	0.01	0.04	0.15	3.88
8SED (improved)	0.01	0.04	0.13	3.38

All times reported for 3×3, city block, chessboard, 5×5, and 7×7 include normalization of integer distance values to doubles. All input images consisted of a solitary point at the center.

they employ unit distances already. Since city block is the fastest with or without normalization, this point is moot. It is the fastest because our implementation entirely eliminates the unnecessary computation of $d(x-1,y-1)+w(-1,-1)$ and $d(x+1,y+1)+w(1,1)$ in the forward and backward passes, respectively. Since CDA 3×3, chessboard, and Euclidean 3×3 all employ 3×3 windows, they exhibit approximately the same execution times even though the Euclidean 3×3 method represents distances using floating point. Actually, chessboard was faster than CDA 3×3 and Euclidean 3×3 because normalization was not required. Also CDA 3×3

was slightly faster than Euclidean 3×3 which can be attributed to integer vs. floating point performance. (In our implementation, the calculation of $\sqrt{2}$ in Euclidean 3×3 and in DRA only occurs once and not repeatedly.) The next fastest was CDA with a 5×5 window, DRA 3×3, CDA with a 7×7 window, and DRA 7×7 which was the slowest. Although DRA 3×3 uses floating point and the extra steps of the DRA, it was faster than CDA with a 7×7 window and normalization (and only slightly slower than CDA with a 7×7 window without normalization). In summary, the algorithms from fastest to slowest were city block, chessboard, CDA 3×3, Euclidean 3×3, CDA 5×5, DRA 3×3, CDA 7×7, and DRA 7×7 (all including normalization if necessary).

3.2. Test images and quantitative evaluation

To numerically evaluate the accuracy of the various distance transforms, we defined a number of input binary images of various sizes. The first test consists of images of sizes 256×256, 512×512, 1000×1000, and 5000×5000 consisting of a single object point at the center of each image. Let (x, y) be this center point. Then $II = \{(x, y)\}$ and $IE = \{(x-1, y), (x+1, y), (x, y-1), (x, y+1)\}$. As mentioned previously, initially $d(x', y') = 0$ for all $(x', y') \in II \cup IE$ (x', y'). Then for any point u in the image we know that the actual Euclidean distance, D, from u to the boundary can be calculated directly by,

$$\min \{D(u, v) | v \in II \cup IE\}. \tag{6}$$

This is the value that *should* be assigned by any distance transform algorithm for any point in this particular test image. To assess the accuracy of a particular distance transform algorithm, we calculate the root mean squared error (RMSE). The quantitative results are reported in Table 3, respectively, for various input image sizes. From this table, one can see that the DRA was the most accurate with an RMSE of 0. The second and third most accurate were CDA using a 7×7 window

Table 3
Root mean squared error for a particular distance transform from the known Euclidean distance for input test images consisting of a single point/object at the center of the image

Algorithm	Image size			
	256×256	512×512	1000×1000	5000×5000
CDA 3×3	3.82	7.66	14.98	74.97
City block	34.89	70.19	137.49	689.08
Chessboard	17.58	35.17	68.69	343.45
CDA 5×5	0.99	2.00	3.93	19.74
CDA 7×7	0.62	1.26	2.48	12.50
Euclidean 3×3	5.76	11.66	22.90	115.03
Dead Reckoning 3×3	<10e − 14	<10e − 14	<10e − 13	<10e − 13
Dead Reckoning 7×7	<10e − 14	<10e − 14	<10e − 13	<10e − 13
4SED	0.39	0.39	0.39	0.39
8SED	<10e − 2	<10e − 2	<10e − 2	<10e − 3
8SED (improved)	<10e − 14	<10e − 14	<10e − 13	<10e − 13

and CDA with a 5×5 window, respectively. The accuracy of CDA 7×7 over CDA 5×5 over CDA 3×3 was demonstrated in general by Borgefors' analysis and this analysis demonstrates it for a specific image. CDA with a 3×3 window had the fourth best RMSE while Euclidean with a 3×3 window had the fifth best RMSE. The sixth and seventh (least) accurate methods were chessboard and city block, respectively. This is not surprising because chessboard employs an incorrect unit diagonal distance and city block never employs the diagonal distance so it tends to overestimate them.

The second set of test images consists of a specific configuration of three points that are known to produce errors for distance transform algorithms that employ 3×3 windows [4,5]. We generated test images of various sizes (32×32 to 5000×5000) as follows. Let the input image be of size $X \times X$ and let $c = X/2$. Then we set the following points: $I(c, c+2) = 1, I(c+1, c+6) = 1,$ and $I(c+2, c+8) = 1$. The results are shown in Table 4 which shows that for extremely small images (32×32 to

Table 4
Root mean squared error for a particular distance transform from the known Euclidean distance for the input test image consisting of three separate points known to be particularly problematic (see text)

Algorithm	Image size						
	32×32	64×64	128×128	256×256	512×512	1000×1000	5000×5000
CDA 3×3	0.42	0.88	1.84	3.76	7.61	14.93	75.92
CDA 7×7	0.06	0.13	0.29	0.60	1.24	2.46	12.48
Dead Reckoning 3×3	0.11	0.24	0.38	0.46	0.51	0.54	0.56
Dead Reckoning 7×7	0.00	0.00	0.03	0.06	0.08	0.09	0.10
8SED (improved)	0.00	0.01	0.01	0.01	0.01	0.01	0.01

Table 5
Average root mean squared error for a particular distance transform from the known Euclidean distance for 100 input test images (each 256×256) consisting 1000 random points

Algorithm	Avg rmse
CDA 3×3	0.16
City block	1.19
Chessboard	0.69
CDA 5×5	0.04
CDA 7×7	0.02
Euclidean 3×3	0.22
Dead Reckoning 3×3	0.01
Dead Reckoning 7×7	0.00003
4SED	0.36
8SED	0.35
8SED (improved)	0.0008

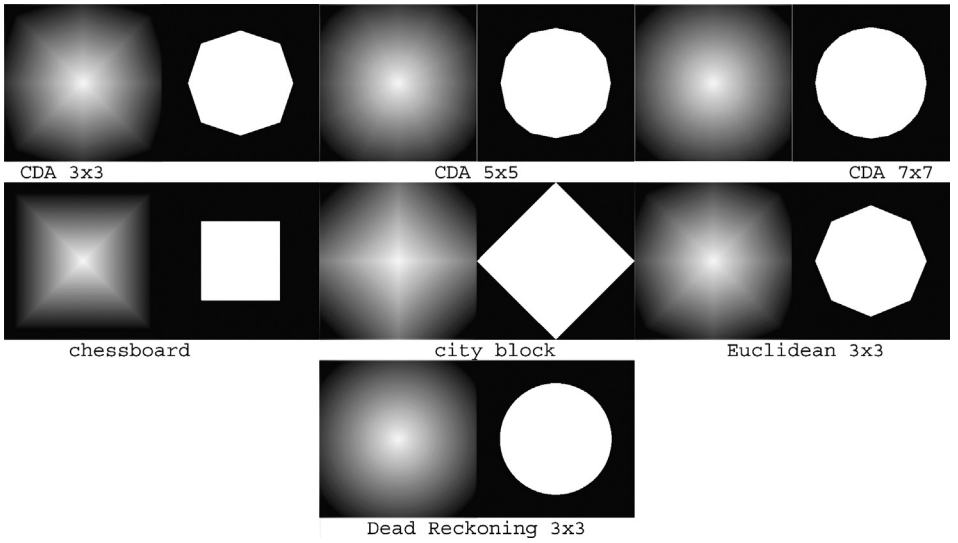

Fig. 4. Results of various distance transform algorithms applied to an input test image containing an object consisting of a single point at the center of the image. The result of each distance transform (left in each pair) is thresholded (right in each pair). The result of a perfect distance transform should be circular for this particular input image.

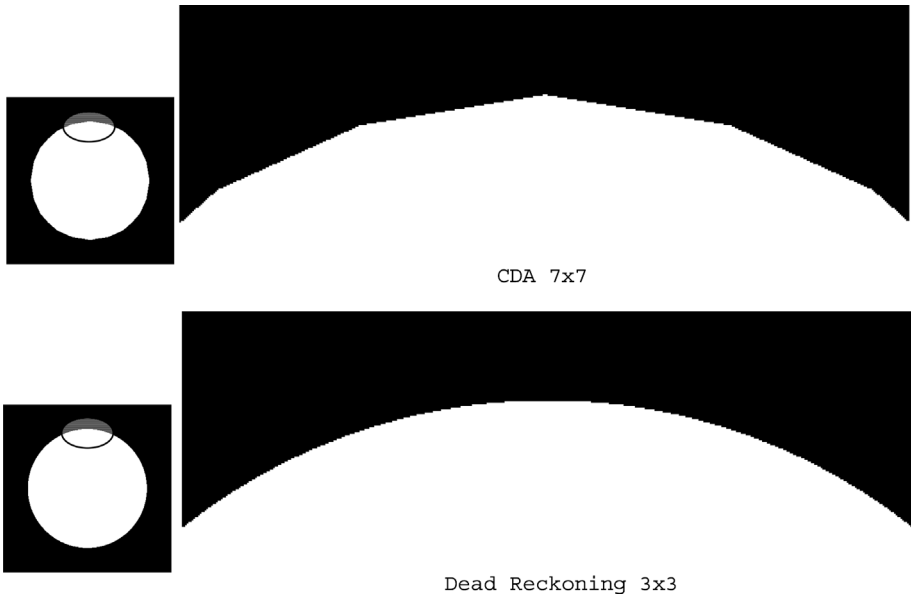

Fig. 5. Comparison of artifacts present in CDA 7×7 and absent in DRA. Although the results appear similar, this figure demonstrates that artifacts are still present in CDA 7×7.

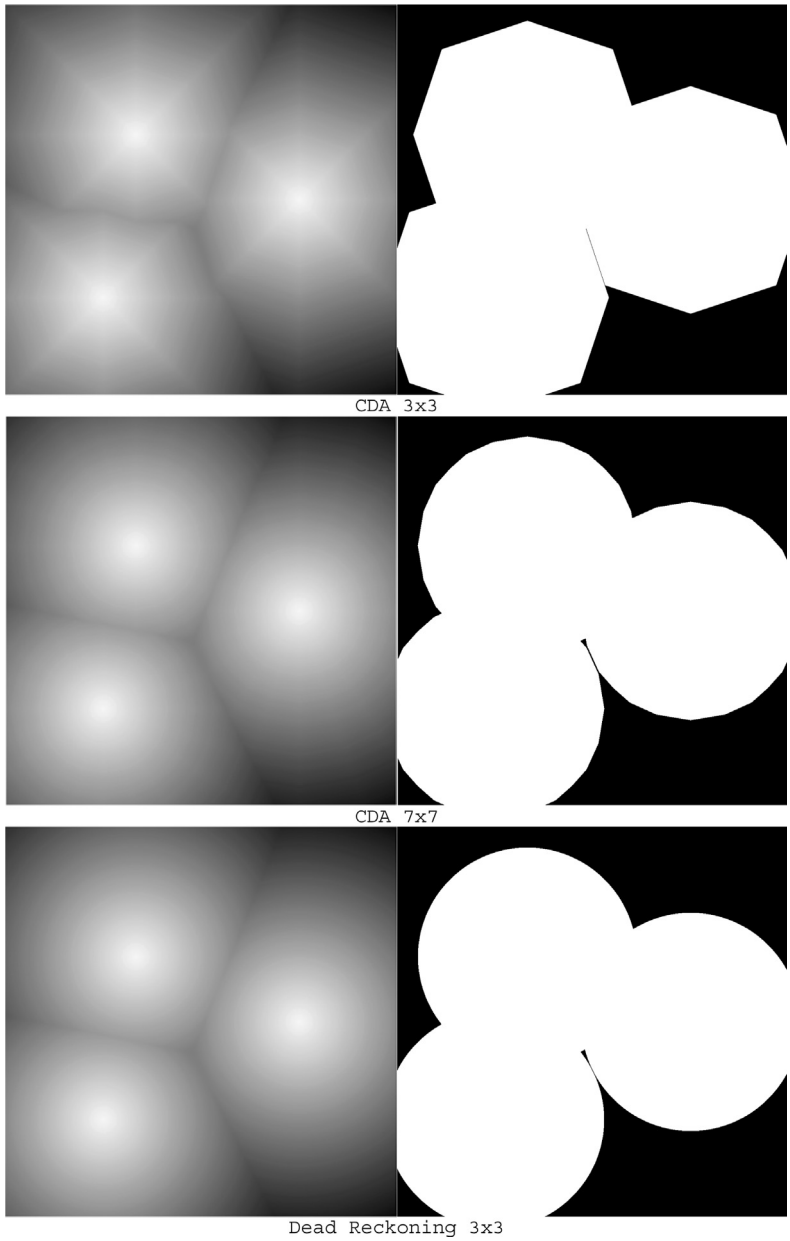

Fig. 6. Distance transform results (left in each pair) for another input test image consisting of 3 single point objects, and the thresholded version illustrating artifacts in all *but* DRA.

128×128) with this specific configuration, CDA 7×7 performs better than DRA 3×3 but DRA 3×3 outperforms CDA 7×7 for more realistic image sizes of 256×256 to 5000×5000. Since our algorithm is readily adaptable to 7×7 windows,

we evaluated DRA 7×7 as well. The results show that DRA 7×7 outperformed CDA 7×7 for all input image sizes.

The third and last set of test images consists of 100 randomly generated 256×256 binary images with 1000 random points in each image set to one. The results of various distance transform algorithms are then applied to each of these images and evaluated in terms of RMSE. Table 5 reports the average RMSE for these 100 images for various algorithms. DRA 3×3 and DRA 7×7 outperform *all* of the other methods with DRA 7×7 only slightly better than DRA 3×3.

3.3. Qualitative evaluation

To demonstrate the artifacts (linear segments at the periphery of the circular isocontour, introduced by the various distance transform algorithms and the lack of artifacts for the DRA), consider again the binary input image consisting of a single point/object at the center of the image. The resulting grey distance transform, viewed as an image, should consist of concentric, circular isovalue contours that are centered about and radiating from the center. Furthermore, if we choose a threshold value and apply it to the gray distance transform result, a circular object centered about the center point should result. The results of this qualitative test are shown in Fig. 4. All distance transform algorithms *except* DRA 3×3 (and DRA 7×7) exhibited artifacts. Even though artifacts appear absent for CDA 7×7, closer inspection reveals that they are still present in CDA 7×7 and absent in DRA 3×3 as shown in Fig. 5.

To illustrate these artifacts further, consider a different input binary image consisting of three objects, each consisting of a single point/object separated by some finite distances. Fig. 6 illustrates the results of applying CDA 3×3, CDA 7×7, and DRA 3×3 to this test image. Again the thresholded CDA 3×3 and 7×7 both exhibit artifacts but artifacts are absent in DRA 3×3.

4. Conclusions

We have presented a new distance transform algorithm, Dead Reckoning, that is a straightforward modification of the well-known Chamfer distance algorithm. We also developed and presented a framework by which one can evaluate distance transform methods with regard to both their quantitative as well as qualitative aspects. We demonstrated that this new algorithm executes with a computational cost approximately the same as the most accurate Chamfer distance algorithm, CDA 7×7, but is much more accurate. In the future, we intend to prove that this new algorithm is indeed a distance metric. We also intend to compare this algorithm with other algorithms (such as Dijkstra's) in terms of speed, computational complexity, and accuracy using a testing framework that we are in the process of developing.

Free source code for DRA as well as CDA and other methods mentioned in this paper may be obtained from http://www.mipg.upenn.edu/~grevera/code.

Acknowledgments

The author gratefully acknowledges NIH Grants NS 37172 and P01-CA85424 for support of this work.

References

[1] G. Borgefors, Distance transformations in digital images, Comput. Vis. Graph. Image Process. 34 (1986) 344–371.
[2] G. Borgefors, On digital distance transforms in three dimensions, Comp. Vis. Image Understand. 64 (3) (1996) 368–376.
[3] M.A. Butt, P. Maragos, Optimum design of chamfer distance transforms, IEEE Trans. Image Process. 7 (10) (1998) 1477–1484.
[4] O. Cuisenaire, Distance transformations: fast algorithms and applications to medical image processing. Ph.D. thesis, Universite Catholique de Louvian, October 1999.
[5] O. Cuisenaire, B. Macq, Fast Euclidean distance transformation by propagation using multiple neighborhoods, Comp. Vis. Image Understand. 76 (2) (1999) 163–172.
[6] L. da Fontoura Costa, Robust skeletonization through exact Euclidean distance transform and its application to neuromorphometry, Real-Time Imag. 6 (2000) 415–431.
[7] P.-E. Danielsson, Euclidean distance mapping, Comp. Graph. Image Process. 14 (1980) 227–248.
[8] G.J. Grevera, J.K. Udupa, Shape-based interpolation of multidimensional grey-level images, IEEE Trans. Med. Imag. 15 (6) (1996) 881–892.
[9] G.J. Grevera, J.K. Udupa, An objective comparison of 3-D image interpolation methods, IEEE Trans. Med. Imag. 17 (4) (1998) 642–652.
[10] G.J. Grevera, J.K. Udupa, Y. Miki, A task-specific evaluation of three-dimensional image interpolation techniques, IEEE Trans. Med. Imag. 18 (2) (1999) 137–143.
[11] G.T. Herman, J. Zheng, C.A. Bucholtz, Shape-based interpolation, IEEE Comp. Graph. Appl. 12 (3) (1992) 69–79.
[12] D. Kozinska, Multidimensional alignment using the Euclidean distance transform, Graph. Models Image Process. 59 (6) (1997) 373–387.
[13] D.W. Paglieroni, Directional distance transforms and height field preprocessing for efficient ray tracing, Graph. Models Image Process. 59 (4) (1997) 253–264.
[14] C. Pudney, Distance-ordered homotopic thinning: a skeletonization algorithm for 3D digital images, Comp. Vis. Image Understand. 72 (3) (1998) 404–413.
[15] S.P. Raya, J.K. Udupa, Shape-based interpolation of multidimensional objects, IEEE Trans. Med. Imag. 9 (1) (1990) 32–42.
[16] E. Remy, E. Thiel, Computing 3D medial axis for chamfer distances, Discrete Geom. Comput. Imagery (2000) 418–430.
[17] E. Remy, E. Thiel, Medial axis for chamfer distances: computing look-up tables and neighbourhoods in 2D or 3D, Pattern Recogn. Lett. 23 (6) (2002) 649–662.
[18] A. Rosenfeld, J.L. Pfaltz, Distance functions on digital pictures, Pattern Recogn. 1 (1) (1968) 33–61.
[19] G. Sanniti di Baja, Well-shaped, stable, and reversible skeletons from the (3,4)-distance transform, J. Vis. Commun. Image Represent. 5 (1) (1994) 107–115.
[20] J.A. Schnabel, L. Wang, S.R. Arridge, Shape description of spinal cord atrophy in patients with MS, Comput. Assist. Radiol. (1996) 286–291.
[21] S. Svensson, G. Borgefors, On reversible skeletonization using anchor-points from distance transforms, J. Vis. Commun. Image Represent. 10 (1999) 379–397.
[22] A.J. Travis, D.J. Hirst, A. Chesson, Automatic classification of plant cells according to tissue type using anatomical features obtained by the distance transform, Ann. Bot. 78 (1996) 325–331.
[23] J.K. Udupa, Multidimensional digital boundaries, Comput. Vis. Graph. Image Process.: Graph. Models Image Process. 56 (4) (1994) 311–323.

[24] G.W.A.M. Van Der Heijden, J.G. Van De Vooren, C.C.M. Van De Wiel, Measuring cell wall dimensions using the distance transform, Ann. Bot. 75 (1995) 545–552.
[25] U. Montanari, A method for obtaining skeletons using a quasi-Euclidean distance, J. ACM 15 (1968) 600–624.
[26] S. Marchand-Maillet, Y.M. Sharaiha, Euclidean ordering via chamfer distance calculations, CVIU 73 (3) (1999) 404–413.
[27] B.J.H. Verwer, P.W. Verbeek, S.T. Dekker, An efficient uniform cost algorithm applied to distance transforms, IEEE Trans. PAMI 11 (4) (1989) 425–429.
[28] F. Leymarie, M.D. Levine, Fast raster scan distance propagation on the discrete rectangular lattice, CVGIP:IU 55 (1) (1992) 84–94.
[29] I. Ragnemalm, Neighborhoods for distance transformations using ordered propagation, CVGIP:IU 56 (3) (1992) 399–409.
[30] W. Guan, S. Ma, A list-processing approach to compute Voronoi diagrams and the Euclidean distance transform, IEEE Trans. PAMI 20 (7) (1998) 757–761.
[31] H. Eggers, Two fast Euclidean distance transformations in Z2 based on sufficient propagation, CVIU 69 (1) (1998) 106–116.
[32] L. Boxer, R. Miller, Efficient computation of the Euclidean distance transform, CVIU 80 (2000) 379–383.
[33] R.A. Lotufo, A.A. Falcao, F.A. Zampirolli, Fast Euclidean distance transform using a graph-search algorithm, IEEE Proc. Comput. Graph. Image Process. (2000) 269–275.
[34] R. Satherley, M.W. Jones, Vector-city vector distance transform, CVIU 82 (2001) 238–254.
[35] T. Saito, J.-I. Toriwaki, New algorithms for euclidean distance transformation of an n-dimensional digitized picture with application, Pattern Recognit. 27 (11) (1994) 1551–1565.
[36] A. Meijster, J.B.T.M. Roerdink, W.H. Hesselink, A general algorithm for computing distance transforms in linear time, in: J. Goutsias, L. Vincent, D.S. Bloombers (Eds.), Mathematical Morphology and its Applications to Image and Signal Processing, Kluwer, 2000, pp. 331–340.