FAST AND EXACT SIGNED EUCLIDEAN DISTANCE TRANSFORMATION WITH LINEAR COMPLEXITY.

O. Cuisenaire and B. Macq

Communications and Remote Sensing Laboratory
Université catholique de Louvain
Place du Levant 2, B- 1348 Louvain-la-Neuve, Belgium

ABSTRACT

We propose a new signed or unsigned Euclidean distance transformation algorithm, based on the local corrections of the well-known 4SED algorithm of Danielsson. Those corrections are only applied to a small neighborhood of a small subset of pixels from the image, which keeps the cost of the operation low.

In contrast with all fast algorithms previously published, our algorithm produces perfect Euclidean distance maps in a time linearly proportional to the number of pixels in the image. The computational cost is close to the cost of the 4SSED approximation.

1. INTRODUCTION

A distance map is an image where the value of each pixel is the distance to the nearest pixel from a set of objects. A distance transformation (DT) is the algorithm producing this map from a binary image describing the objects. A Euclidean DT is a transformation that uses the Euclidean metric. A signed DT produces in each pixel the vector of the relative position of the nearest object pixel (NOP). Signed distance maps are a representation of the Voronoi division of the object pixels. The set of all points sharing the same NOP is called the tile of this pixel in the Voronoi diagram.

The DT is an important tool in image processing with many applications in image analysis and pattern recognition [5], [7], [10], [13]. On the other hand, computing a EDT is not straightforward since the direct application of the above definition leads to a prohibitive computational cost. This has lead to the development of many approximate or exact DT algorithms over the last 20 years.

1.1 EDT Approximations

The first idea for producing fast DT algorithms was to use coarse *approximations of the Euclidean metric*, such as the city-block and chessboard metrics [1], and later the Chamfer DT [3], [5]. With these metrics, the value associated with a pixel can be derived from the values of its neighbors, which allows these algorithms to work in two raster scans over the image. Unfortunately these approximations lack such important features as rotational invariance, which makes them unpractical for some applications.

Another approximation was proposed by *Danielsson* [2]. He propagates a vector with the position of the nearest object pixel instead of the scalar distance only. The 4SED and 8SED algorithms – using direct and 3x3 neighborhoods respectively - give the correct Euclidean value for most pixels, but can be erroneous for some configurations of object pixels. The first such error (Fig. 1) happens when the white pixel labeled 22 is disconnected from the tile of A in the Voronoi division. This leads to the pixel to be mislabeled as 30 or 03 by the 4SED algorithm - which considers direct neighbors only. 8SED also considers indirect neighbors and finds the correct value.

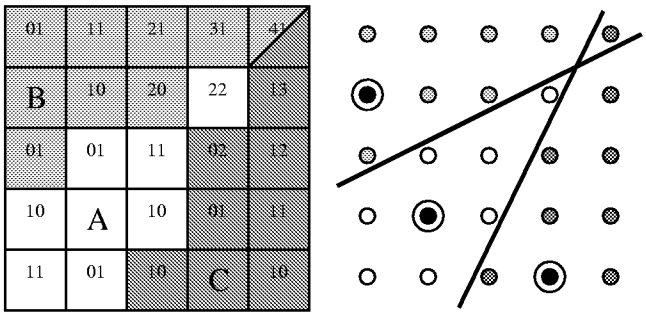

Figure 1: discontinuity of the area of influence of pixel A on the discrete plane, considering direct neighborhoods only. A pixel labeled "xy" means that the relative position of the nearest object pixel (A,B or C) is (x,y) in absolute value. Left: digital image. Right: equivalent continuous plane.

1.2 Exact EDT

In order to produced error-free EDT, *parallel algorithms* [4], [14], [15] allow multiple propagation fronts to follow each other so that the pixel labeled 22 in the previous example will first be reached by the information from A before B or C hides it. Unfortunately, a direct implementation of parallel algorithms leads to a prohibitive computational cost on sequential computers.

Distance transformations by propagation [6], [8], [9], [12], [19], [20] provide efficient implementations of the parallel algorithm principle. All of them have some original way to emulate the multiple propagation front property, and most of the computational time is usually spent in processing these multiple fronts. Finally, *raster-scanning EDT* also provide such a mechanism, either explicitly [11] or implicitly [16].

Because of this mechanism, none of the above algorithms has a linear time complexity. Simple images such as a diagonal line or a circle of object pixels are counter-examples for which the complexity is at least $o(N^3)$ for NxN images. The only proven

$o(N^2)$ exact EDT algorithm [17] hasn't - to the best of our knowledge - been empirically evaluated, but is likely to be less efficient than some other DTs for images of a reasonable size. Finally, in [18], [21], we proposed a DT by propagation with multiple neighborhoods which approaches the $o(N^2)$ behavior.

In this paper, we propose a new exact EDT algorithm whose computation time is similar to the approximate DT algorithms. In section 2, we produce a first approximation of the signed DT. In section 3, we detect and correct the erroneous pixels. The computational and memory requirements of the algorithm are evaluated at section 4, and compared with other well known DTs.

2. EUCLIDEAN DT APPROXIMATION

As a first step, we produce an approximation of the signed Euclidean DT using the signed version Danielsson's 4SED algorithm.

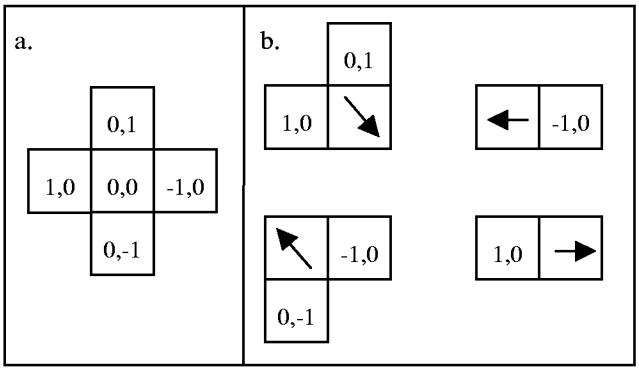

Figure 2: Masks used for the 4SSED algorithm. a) the original mask. b) the propagation masks.

This algorithm relies on the following assumption: "the relative location of the NOP for a pixel can be inferred from the location of the NOP of its direct neighbors". For each pixel, the NOP minimizes the distance among the 5 possible locations found by adding the values of the mask of Figure 2a to the respective neighbors' NOP locations.

This mask is split (Fig. 2b) to allow a propagation in two raster scans, with a backward propagation for each line during the raster scan, as suggested by the arrows. In total, 6 comparisons per pixel are performed.

As pointed out in the introduction, the assumption on which 4SED is based isn't verified for some configurations of object pixels (Fig. 1). This has led Danielsson to propose the 8SED algorithm, a better approximation using a larger (3x3) mask, but which is not error-free either. Actually, whatever the size of the neighborhood N used, one can find a configuration of pixels for which a tile of the Voronoi diagram isn't N-connected, which leads to possible errors in a NSED algorithm.

3. POST-PROCESSING

3.1 Principle

All errors in the map created by the 4SED DT compared to the exact EDT are located at pixels that are disconnected from the tile of pixels that share the same NOP. This only happens on the corners of tiles of the Voronoi diagram of the object pixels.

A corner pixel in the approximate distance map is easily detected since it has at least two direct neighbors belonging to different tiles then itself. An efficient implementation of such a test only requires one comparison per non-corner pixel.

For each corner in the approximate DT, we only need to check that the corner of the tile in the continuous plane includes no disconnected pixel in the discrete lattice. While we obviously do not know the exact shape of this tile in the continuous plane, we can define boundaries that include it and efficiently limit the number of pixels to be checked.

At Figure 3, the pixel labeled (0,0) is such a corner in the signed map created by 4SSED. It belongs to the tile of object pixel X, while its first quadrant neighbors, up and right, are closer to Y and Z respectively (similar developments could of course be done for the other 3 quadrants). The neighbors (i,j) for which we should check if they belong to the tile of X are those in the gray area. Indeed, they must be closer to X than Y, and thus below the line labeled "Max-line", and closer to X than Z, i.e. above "min-line". Max-line and min-line are the mid-perpendiculars of XY and XZ respectively.

From the signed DT, we know the positions of X,Y and Z relatively to (0,0). They are $(-x_1,-x_2)$, $(-y_1,-y_2+1)$ and $(-z_1+1,-z_2)$ respectively. Thus, we can determine the equations of both lines, i.e. the coefficients of the two half planes defining the gray area.

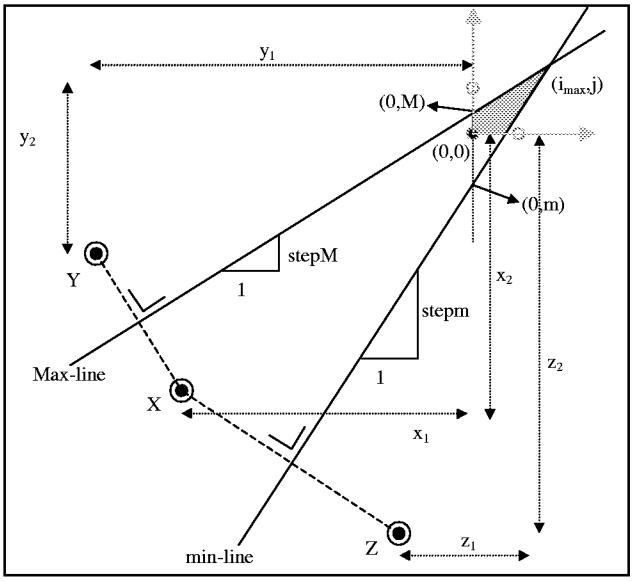

Figure 3: Only the gray region needs to be searched for improvements to 4SSED.

$$j \le M + i * stepM$$
$$j \ge m + i * stepm$$

The coefficients for Max-line are

$$stepM = \frac{y_1 - x_1}{x_2 - y_2 + 1}$$
$$M = \frac{1}{2}.(stepM * (x_1 + y_1) - x_2 - y_2 + 1)$$

and the coefficients of min-line

$$stepm = \frac{z_1 - 1 - x_1}{x_2 - z_2}$$
$$m = \frac{1}{2}.(stepm * (x_1 + z_1 - 1) - x_2 - z_2)$$

Finally, we determine the value (i_{max} ,j) at the intersection of the lines.

$$i_{max} = \frac{M - m}{stepm - stepM}$$

Singular cases happen when $x_2 = z_2$ or stepm = stepM. In the first case, min-line is vertical and no pixel is ever included in the gray area. In the second case, there are two possibilities. If Y and Z are identical, both lines are merged and no pixel is included in the gray area. Otherwise, stepm = stepM = 1 and the lines are parallel. In this case, the pixels are checked until one does not belong to the tile of X, and no further.

3.2 Algorithm

The first step of the algorithm is of course the 4SSED double raster scan over the image. The second step can be written as follows

```
For all pixel p but the last column
  if NOP(p) != NOP(p+(0,1))
  then
    if NOP(p) != NOP(p+(-1,0)) then check (p,1);
    if NOP(p) != NOP(p+(1,0)) then check (p,2);
    if NOP(p+(0,1)) != NOP(p+(1,1)) then check (p+(0,1),3);
    if NOP(p+(0,1)) != NOP(p+(-1,1)) then check (p+(0,1),4);
```

For most pixels, the first test is not passed and only one comparison is needed. For corner pixels, the procedure check(p,n) is called, where p is the pixel and n the quadrant in which to check. For instance, for the first quadrant used in section 3.1, we have the following procedure

```
get (x1,x2), (y1,y2) and (z1,z2) from the distance map;
check for singular cases;
compute m,M,stepm,stepM;
compute imax ;
for i = 1 → imax
    m = m+stepm;  M = M+stepM
    for j = m → M
          testp = p + (i,j);
          if d( testp , NOP(p) ) < d ( testp, NOP(testp))
               NOP( testp ) = NOP ( p);
```

where d(p,q) is the Euclidean distance between pixels p and q. We call this algorithm Corrected Sequential Signed Euclidean Distance, or CSSED.

4. COMPUTATIONAL COST

In order to assess the computational efficiency of our algorithm, we compare it to the most efficient unsigned exact EDT known to this day, those of Eggers [19], Saito [16] and our own PMN [21]. The tests performed are those suggested by Saito and Eggers themselves. Test 1 is an image including a maximal inscribed disk of non-object pixels. The size of the image varies from 200x200 to 1600x1600. Test 2 is a 1024x1024 image with 15% of object pixels made of random placed squares with a common orientation varying from 0 to 45°. The criteria used for comparison are the CPU time per pixel and the absolute CPU time, respectively.

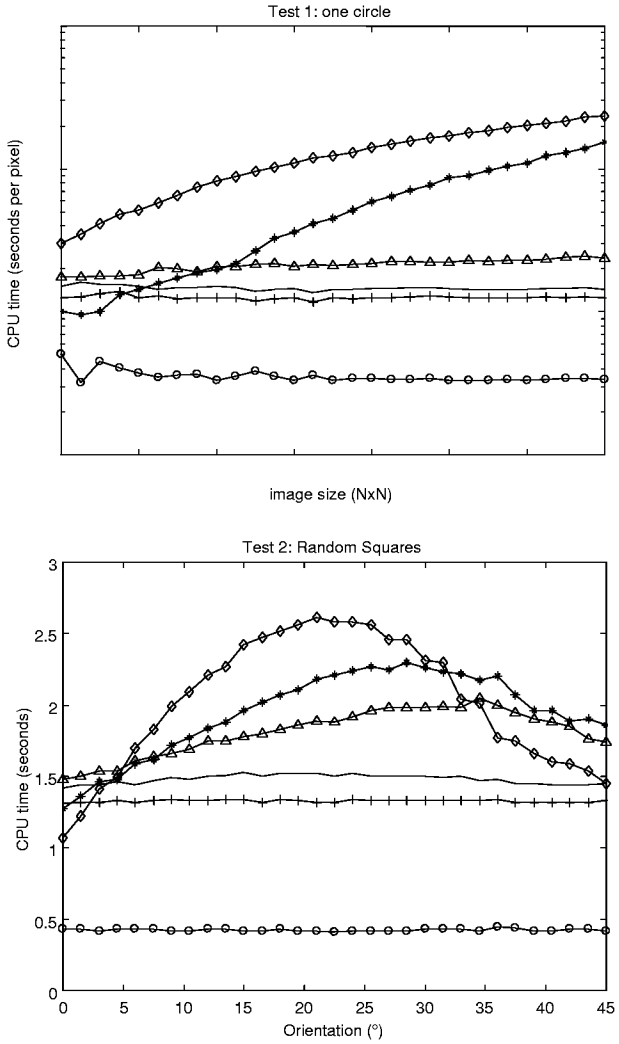

Figure 4: Comparison of CPU time required for each algorithm. "o" Chamfer 34, "+" 4SSED, "◊" Eggers, "*" Saito, "Δ" PMN, "_" CSSED.

Test 1 shows that Chamfer, 4SSED, CSSED and PMN are $o(N^2)$ for NxN images, while Eggers is $o(N^3)$ and Saito $o(N^4)$. CSSED only has a 15% additional cost compared to the approximate 4SSED, whatever the size of the image. Besides, CSSED is the absolute fastest exact EDT for images larger than 400x400 pixels. Our algorithm is even an order of magnitude faster than Saito's or Eggers for the largest images of the test.

Test2 shows that the computational cost of Chamfer, 4SSED and CSSED are orientation independent. PMN, Saito and Eggers are all orientation dependant, with a maximal variability for Eggers' algorithm. CSSED is the absolute fastest exact EDT apart from the exceptional case where all edges in the image are either vertical or horizontal.

As a conclusion, although CSSED provides an exact EDT, its cost is similar to those of the approximate algorithms.

5. CONCLUSION AND PERSPECTIVES

We have developed an extension of the 4SSED algorithm that provides an exact Euclidean distance transformation in a time typical of the DT approximations. Its complexity is optimal, i.e. linear relatively to the number of pixels that the image contains. Besides, it is fast even for small images. This result is close to the theoretical optimum for such algorithms.

Other versions of this algorithm could be produced by using a different approximate DT for the first step. For instance, one could use Ragnelmam [12] without the delayed updating mechanism, or our PSN algorithm [21], if those prove to be faster on the computer used. This is largely machine dependant.

Formal proofs of the correctness of the algorithm and the extendibility to 3 dimensions or anisotropic data are under study.

6. ACKNOWLEDGEMENTS

The work of Olivier Cuisenaire is supported by the Belgian FRIA Fund.

7. REFERENCES

[1] A.Rosenfeld and J.L. Pfaltz "Distance functions on digital pictures" *Pattern Recognition,* **1** (1) 1968, pp. 33-61.

[2] P.E. Danielsson "Euclidean distance mapping". *Computer Graphics and Image Processing,* **14**, 1980, pp. 227-248.

[3] G. Borgefors "Distance transformations in arbitrary dimensions". *Computer Vision, Graphics and Image Processing*, **27**, 1984, pp. 321-345

[4] H. Yamada, "Complete Euclidean Distance Transformation by Parallel Operation". *Proc. 7th Internationl Conference on Pattern Recognition*, Montreal, pp. 69-71, 1984.

[5] G. Borgefors. "Distance transformations in digital images". *CVGIP,* **34**, 1986, pp. 344-371

[6] J. Piper and E. Granum "Computing Distance Transformations in Convex and Non-Convex Domains". *Pattern Recognition,* **20**(6), 1987, pp. 599-615.

[7] G. Borgefors "Hierarchical chamfer matching: a parametric edge matching algorithm". *IEEE Trans. Pattern Analysis and Machine Intelligence,* **10** (6), 1988, pp. 849,865

[8] B.J.H. Verwer, P.W. Verbeek and S.T. Dekker "An Efficient Uniform Cost Algorithm Applied to Distance Transforms". *IEEE Trans. PAMI,* **11**(4), 1989, pp 425-429.

[9] L. Vincent "Exact Euclidean Distance Function by Chain Propagation". *Proc. Computer Vision and Pattern Recognition Conference,* Hawaii, pp. 520-525, June 1991.

[10] D.W. Paglieroni "Distance transforms: properties and machine vision applications". *CVGIP: Graphical Models and Image Processing,* **54** (1), 1992, pp.56-74.

[11] J. Mullikin "The vector distance transform in two and three dimensions". *CVGIP,* **54**(6), 1992, pp. 526-535

[12] I. Ragnelmam "Neighborhoods for Distance Transformations Using Ordered Propagation" *CVGIP, Image Understanding,* **56**(3), 1992, pp. 399-409

[13] I. Ragnelmam "Fast Erosion and Dilation by Contour Processing and Thresholding of Distance Maps" *Pattern Recognition Letters,* **13**, 1992, pp. 161-166

[14] F.Y-C. Shih and O.R. Mitchell "A Mathematical Morphology Approach to Euclidean Distance Transformation". *IEEE Trans. Image Processing,* **1**(2), 1992, pp. 197-204.

[15] C.T. Huang and O.R. Mitchell "A Euclidean Distance Transform Using Greyscale Morphology Decomposition" *IEEE Trans. PAMI,* **16**(4), 1994, pp. 443-448.

[16] T. Saito and J.I. Toriwaki "New Algorithms For Euclidean Distance Transformations of an n-Dimensional Digitised Picture with Applications". *Pattern Recognition,* **27**(11), 1994, pp. 1551-1565.

[17] H. Breu, J. Gil, D. Kirkpatrick and M. Werman, "Linear Time Euclidean Distance Transform Algorithms", *IEEE Trans. PAMI,* **17**(5), 1995, pp. 529-533.

[18] O. Cuisenaire "Region growing Euclidean distance transforms". *in Proceedings 9th International Conference on Image Analysis and Processing (ICIAP'97),* vol. 1, pp. 263-270, Florence, September 1997.

[19] H. Eggers "Two Fast Euclidean Distance Transformations in Z^2 Based on Sufficient Propagation". *Computer Vision and Image Understanding,* **69**(1) 1998, pp 106-116.

[20] W. Guan and S. Ma. "A List-Processing Approach to Compute Voronoi Diagrams and the Euclidean Distance Transform". *IEEE Trans. PAMI*, **20**(7), 1998, pp 757-761.

[21] O. Cuisenaire and B. Macq, "Fast Euclidean Distance Transformation by Propagation using Multiple Neighborhoods", *submitted to CVIU.*