Pattern Recognition Letters 32 (2011) 252–257

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Anti-aliased Euclidean distance transform

Stefan Gustavson [a,*], Robin Strand [b,**]

[a] Media and Information Technology, Linköping University, Sweden
[b] Centre for Image Analysis, Uppsala University, Sweden

ARTICLE INFO

Article history:
Received 8 July 2009
Available online 7 September 2010
Communicated by T.K. Ho

Keywords:
Distance transform
Vector propagation
Euclidean metric
Sub-pixel accuracy

ABSTRACT

We present a modified distance measure for use with distance transforms of anti-aliased, area sampled grayscale images of arbitrary binary contours. The modified measure can be used in any vector-propagation Euclidean distance transform. Our test implementation in the traditional SSED8 algorithm shows a considerable improvement in accuracy and homogeneity of the distance field compared to a traditional binary image transform. At the expense of a 10$\times$ slowdown for a particular image resolution, we achieve an accuracy comparable to a binary transform on a supersampled image with 16 $\times$ 16 higher resolution, which would require 256 times more computations and memory.

1. Introduction

Traditionally, a distance transform (DT) is defined on a binary image as the mapping from each foreground picture element (pixel) to its distance from the closest background pixel. Distance functions and the corresponding DTs are important tools in image processing and analysis. An early paper on the subject is Rosenfeld and Pfaltz (1966), where a two-scan sequential algorithm was presented to compute the DT based on the city-block and chessboard distance functions. These distance functions are *path-generated*, since the distance between two pixels is defined as a shortest path between the pixels. Several extensions of path-generated distance functions have been suggested, see, e.g., Borgefors (1996), Rosenfeld and Pfaltz (1968) and Strand (2007).

In Danielsson (1980), an algorithm based on vector propagation for computing the Euclidean DT is presented. The basic idea is to assign the *vector* from the closest background pixel to each foreground pixel. This approach is also considered in, e.g., Mullikin (1992) and Ragnemalm (1993).

There are many other ways to compute (approximate) Euclidean DTs. The most common alternative methods are:

- The *fast-marching* method, (Sethian, 1999). This frequently used method, which can be formulated as a level-set method, approximates a differential equation (the Eikonal equation) to model an evolving wavefront of constant velocity. Scalar values are propagated to compute the arrival time (Euclidean distance) of the wavefront. In Kimmel et al. (1996), sub-pixel distance values are computed by interpolation of distance values from distance values at adjacent grid points.
- Methods based on dimensionality reduction, (Felzenszwalb and Huttenlocher, 2004; Maurer et al., 2003; Saito and Toriwaki, 1994): Since the squared Euclidean distance value is separable, the DT can be computed by one-dimensional scans along each principal direction.
- Grayscale morphology: The (squared) Euclidean DT can be obtained by dilating the binary image with a (large) structuring element. Efficient algorithms are obtained by decomposing the structuring element, see Huang and Mitchell (1994), Mehnert and Jackway (1999) and Shih and Mitchell (1992).

2. Overcoming the constraints of a binary image

The traditional restriction of distance transforms to binary images, where each pixel belongs to either the foreground or the background, results in a well defined discrete problem, and in many applications the constraints of a binary input image can be tolerated. However, a binary image is spatially discretized to a regular grid, and therefore unable to represent general binary shapes correctly. Edges which are not aligned with the sampling grid are discretized with strong stair-step edge aliasing artifacts, referred to in the computer graphics literature as as "jaggies". A binary two-dimensional shape with crisp edges of arbitrary position and orientation contains significant high spatial frequency components and is not band limited. It is not well represented by point sampled, binary-valued pixels and can not be well reconstructed from such data (Köthe, 2008; Schilling, 2001).

A binary image conveys no information on the sub-pixel position of the edge of the sampled shape, as each sample is considered

* Corresponding author.
** Corresponding author.
E-mail addresses: stegu@itn.liu.se (S. Gustavson), robin@cb.uu.se (R. Strand).

0167-8655/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2010.08.010

to be either fully inside or fully outside the contour. Therefore, distance vectors for a binary DT are constrained to an integer grid. Following Sladoje and Lindblad (2009) and common practice in computer graphics, we use a model where the image is instead *area sampled*, whereby the intensity value in a pixel represents the *area coverage*, the fraction of the pixel rectangle that is intersected by an object. The result is that pixels intersecting the edge of a binary contour attain grayscale values, whereby edge aliasing is greatly reduced. The limited sub-pixel information made available through the area coverage can be used to improve precision in, e.g., estimators for edge length and position (Verbeek and van Vliet, 1993), arc length and area (Eberly and Lancaster, 1991), and reconstruction of the digitized object (Kiryati and Bruckstein, 1988). Using the area sampling model, Sladoje and Lindblad (2009) gives a method that provides error-free measurements of the length of straight boundary segments in the case of non-quantized pixel values.

Real world digital imaging sensors consist of finite-area sensor elements, and the sampling performed by such devices is inherently an area sampling, with light being integrated over the sensor element convolved with the point spread function of the optics (Chen et al., 2009). While the actual sampling kernel of a real imaging device does not have a crisp edge or a rectangular shape, it is certainly much closer to an area sampling than to a point sampling. In synthetic, rendered images, point sampling causes aliasing of high frequency features in the image, e.g. edges. Multi-sampling or analytic *anti-aliasing* strategies are employed to effectively replace the point sampling with an approximation of area sampling, bringing the sampling closer to what a physical imaging device would do.

In the novel DT method presented in this paper, the distance vectors point not to the center of the closest pixel in the spatially discretized representation of the object, but instead to the closest position on the estimated *border* of the spatially continuous object. This extension of the traditional definition of the distance in DT algorithms makes the distance vectors unconstrained by the discrete sample grid. The position of the border is estimated by using the area coverage of a single pixel straddling the edge. An improved estimate additionally uses a small region of adjacent pixels for better accuracy near the edge. Scalar propagation methods (listed above) can often be extended to use area sampled grayscale images instead of point sampled binary images as input. Grayscale valued input images are considered for the dimensionality reduction method in Felzenszwalb and Huttenlocher (2004). Another scalar propagation method with sub-pixel accuracy is presented in Krissian and Westin (2005), where interpolation of the computed distance values in a level set method is used to get a better approximation of the wavefront that corresponds to the evolving level set. In Krissian and Westin (2005), the level set is approximated by interpolation using distance values at grid points in a narrow strip and, similarly to our method, the sub-pixel accuracy is obtained by using the distance to the underlying continuous curve instead of the distance to the discrete grid points.

Under the assumption that the edge is locally straight, the pixel coverage and the edge direction together determine the distance from the pixel center to the edge. In this paper, the vector to the closest pixel intersecting the edge is adjusted by a small sub-pixel distance computed from the approximate direction of the edge and the area coverage of the edge pixel. The result is a sub-pixel accuracy extension suitable for use with any vector propagation method for DT computation.

3. Applications

There are a number of image processing applications where the proposed method can be used to provide better results than a binary DT. By modifying the definition of distance to measure the distance to the ideal boundary between foreground and background, signed distance fields created by combining two DT passes over the image, one on the image and one on its inverse, will not exhibit a problematic discontinuity at the edge, and therefore morphological operations like erosion and dilation with small distances of only a few pixels will be significantly more accurate than by using a binary DT. Euclidean erosion and dilation can be performed with arbitrary fractional distances, even less than one pixel. This is not possible with a binary transform (Ragnemalm, 1992). Such sub-pixel erosion and dilation of anti-aliased images are useful for e.g. digital photolithography, where there is a need to compensate for imperfections in the exposure and etching process which yield an erosion or dilation of the final outlines (Wei and Brainard, 2009). Level sets created from DT images with our method will be more accurate, in particular near the edge, even from lower resolution input data. Accurate distance maps for synthetic images can be created from regular anti-aliased images instead of from special, high resolution binary images, which are cumbersome to create and difficult to process (Green, 2007). Accurate distance maps can also be created from (approximately) area sampled digitized images of real world objects, better utilizing the information in images where the spatial resolution is limited by physical constraints.

4. Anti-aliased distance function

A binary distance transform is defined for images with only two pixel values: each pixel is considered to belong either entirely to the foreground or entirely to the background. To lift this restriction, we make the following definitions: Assume a piecewise constant binary function $B(x,y) \in \{0,1\}$ defined over a continuous plane (x,y). $B(x,y)$ contains a set of compact regions of arbitrary shape and with discontinuous step edges. By arbitrary choice, $B = 1$ represents foreground and $B = 0$ the background. (The opposite convention is also quite common.) We create a square grid of pixels and assign a value a, $0 \leqslant a \leqslant 1$, to each pixel based on the fraction of the pixel that is intersected by the foreground, as illustrated in Fig. 1(a) and (b). The grayscale values in the input image are used to approximate the location of the object boundary. The difference between the traditional binary distance measure and our modified measure is illustrated in Fig. 1(c).

Our objective is to find the distance to the closest ideal edge from a discrete set of area sampled pixels by analyzing only the pixel values. If we know the orientation of the underlying edge and assume that its local radius of curvature is significantly greater than the pixel size, we can compute the sub-pixel edge position within the pixel from the area coverage a. Denote the (normalized) edge direction φ, which defines $\mathbf{g} = (g_x, g_y) = (\cos\varphi, \sin\varphi)$. The edge direction φ determines the areas a_1 and a_2, the distance values d_1 and d_2 illustrated in Fig. 2.

The signed distance from the edge to the center of the pixel is denoted d_f, see Fig. 2, right. Under the assumption that the edge of the object covering the pixel is straight, there is a relation between d_f, the area coverage a, and the edge direction φ. For all orientations of the edge, $a = 0.5$ means $d_f = 0$, and for $g_x = 0$ or $g_y = 0$, the relation between d_f and a is a simple linear ramp:

$$d_f = 0.5 - a. \tag{1}$$

Note that $d_f = 0$ represents an edge that passes through the center of the pixel. For edges in other than axis-aligned directions, this approximation can be off by as much as $\pm(\sqrt{2} - 1)/2$ but is still useful as a simple, first order approximation. If the edge direction is known, d_f can be more accurately expressed as a function of (g_x, g_y) and a, see Fig. 2. Using elementary trigonometry, we can derive the

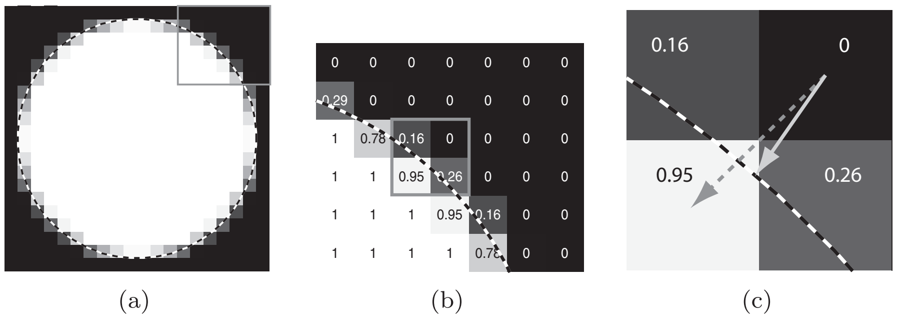

Fig. 1. (a): Pixel representation of a disc by area coverage. True contour shown by dashed line. (b): Detail view of upper right corner of (a). (c): The difference between the traditional integer distance definition (dashed arrow) and the definition proposed in this paper (solid arrow).

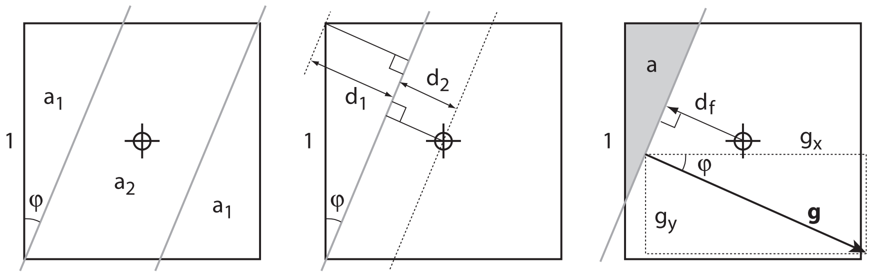

Fig. 2. Notation for working out a as a function of d_f and vice versa. See Eq. 2 for details.

areas a_1 and a_2 of the left-hand side of Fig. 2 and the distances d_1, d_2 in the middle:

$$\begin{aligned} a_1 &= \tan\varphi = g_y/g_x, \\ a_2 &= 1 - 2a_1, \\ d_1 &= \sin\varphi = g_y, \\ d_2 &= (1/\sqrt{2})\sin(\pi/4 - \varphi) = (1/\sqrt{2})\sin(\pi/4 - \arcsin g_y). \end{aligned} \tag{2}$$

The continuous and monotonous function $a(d_f)$ for a given edge direction φ can then be written as:

$$a = \begin{cases} 0, & d_f \leqslant -d_1 - d_2, \\ (d_1 + d_2 + d_f)^2 (a_1/d_1^2), & -d_1 - d_2 \leqslant d_f \leqslant -d_2, \\ a_1 + (a_2/2)(1 + d_f/d_2), & -d_2 \leqslant d \leqslant d_2, \\ 1 - (d_1 + d_2 - d_f)^2 (a_1/d_1^2), & d_2 \leqslant d_f \leqslant d_1 + d_2, \\ 1, & d_f \leqslant d_1 + d_2. \end{cases} \tag{3}$$

In Eq. 3, we can see that, for an arbitrary edge direction φ, the area coverage a for a pixel as a function of d_f starts out as a parabola (proportional to d_f^2) when the edge is in the corner area a_1 shown to the left in Fig. 2, followed by a linear ramp (proportional to d_f) within the center area a_2 and a final parabola in the opposite corner. The function is antisymmetric around $d_f = 0$ and $a = 0.5$, such that $a(-d_f) = 1 - a(d_f)$, and symmetric in φ such that it is enough to look at angles in the first octant $g_x \geqslant g_y \geqslant 0$. Other cases can be evaluated by changing the sign of g_x and/or g_y and if necessary swapping g_x and g_y.

For our application, we want to compute d_f as a function of a. For a fixed edge direction φ, this is the inverse of Eq. 3. The inverse $d_f(a)$ is defined for $0 \leqslant a \leqslant 1$:

$$d_f = \begin{cases} (g_x + g_y)/2 - \sqrt{2 g_x g_y a}, & 0 \leqslant a \leqslant a_1, \\ (0.5 - a) g_x, & a_1 \leqslant a \leqslant 1 - a_1, \\ -(g_x + g_y)/2 + \sqrt{2 g_x g_y (1 - a)}, & 1 - a_1 \leqslant a \leqslant 1. \end{cases} \tag{4}$$

A plot of this function for three different edge directions is presented in Fig. 3. The plot also shows that the linear function $d_f = 0.5 - a$, which is valid for $\varphi = 0$ and $\varphi = \pi/2$, is a reasonable approximation for any edge direction φ.

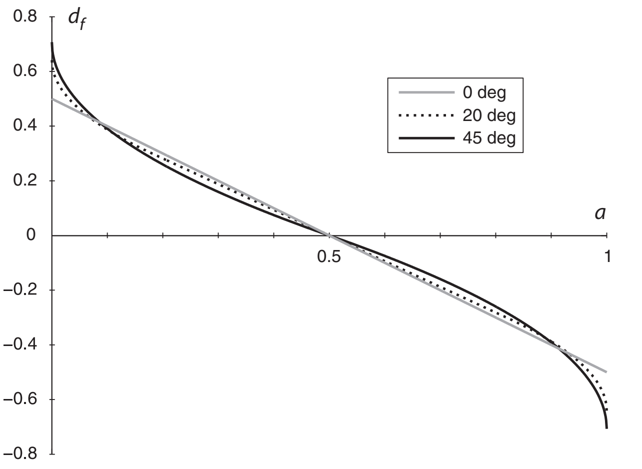

Fig. 3. Plot of $d_f(a)$ for $\varphi = 0°, 20°, 45°$.

5. The anti-aliased vector propagation method

In vector propagation distance transform algorithms, the vector (d_x, d_y) to a candidate for the closest edge is known, and a distance function is computed from the vector. A binary Euclidean distance transform uses the distance function $d_i = \sqrt{d_x^2 + d_y^2}$. A simple but reasonable assumption is that the edge of the object is approximately orthogonal to the vector (d_x, d_y). This assumption does not hold true for small d_i where only a few discrete directions are possible, but it is increasingly accurate for pixels further away from the edge, and asymptotically exact for distant pixels.

In the anti-aliased vector propagation method, d_i is adjusted using d_f from Eq. 4 above:

$$d = d_i + d_f. \tag{5}$$

A vector propagation distance transform using the modified distance measure d requires more computations than merely computing d_i for each candidate tested, but because d_f is bounded, an early-out test on d_i could avoid many computations of d_f in a sequential implementation. Note also that d_f is both bounded and smooth and can be tabulated with good accuracy using a small 2D table and bilinear interpolation if speed is important.

5.1. Further improvements

At edge pixels, the distance vector points to the same pixel and does not contain any information on the direction of the edge. Near the edge, the integer vector from a pixel to the closest edge pixel does not align well with the true direction of the edge. For greater accuracy at those pixels, the local direction of the edge was estimated by two 3×3 gradient filter kernels around each edge pixel. At the edge, the estimated local gradient was always used for (g_x, g_y) in Eq. 4. Near the edge, a distance measure according to Fig. 4 was computed based on the local gradient at the edge and used instead of $d_i + d_f$, but only if it resulted in a well defined orthogonal distance to a linear edge with the same gradient, as explained by Eq. 6.

In our experiments, using the local gradient always improved the accuracy for edge pixels, where no other edge direction information was available. Away from the edge, using the local gradient sometimes increased the error but still improved the average result within one or two pixels distance from the edge. Beyond that, the gradient information was frequently less accurate than the integer distance vector. Therefore, the local gradients were used to improve the result only at the edge and very near it, where most of the larger absolute errors were located.

```
$(g_x, g_y)$ is local gradient at edge pixel
$(d_x, d_y)$ is vector to the edge pixel
compute $d_f$ from (4) using $(g_x, g_y)$
$t = d_i \sin\beta = g_x d_y - g_y d_x$ (cross product, $|\mathbf{g}| = 1$)
$u = -d_f g_x + t g_y$
$v = -d_f g_y - t g_x$
if $|u| \leqslant 0.5$ and $|v| \leqslant 0.5$ ("hit inside pixel")
$d = \sqrt{(dx + u)^2 + (dy + v)^2}$
 else
 compute $d_f$ from (4) using $(d_x, d_y)$
$d = \sqrt{d_x^2 + d_y^2} + d_f$.                                  (6)
```

5.2. Results

We implemented our modified distance measures in a classic sequential Euclidean distance transform (EDT) vector propagation algorithm using eight neighbours (8SED), see Danielsson (1980). The implementations were not optimized for speed, as we were only interested in showing the performance relative to a classic binary DT. The exact same propagation code was used for both the reference binary DT and the improved anti-aliased DT, so the performance figures in Table 1 are good indicators of the relative slowdown due to using the new floating-point distance measure in the propagation loop instead of the integer-only evaluation of d_i^2 for the binary DT. The performance figures show that the much improved accuracy comes at a reasonable cost in execution time, even without any early-out tests or look-up table speedups. The exact slowdown for the last improvement (Eq. 6) depends somewhat on the amount of edges in the image, as the extra effort is spent only on pixels near edges. However, the main algorithm requires several sweeps and updates across the image, and the final adjustment of distances using the pre-computed local gradient information constituted only a minor part of the computations. In a worst case scenario where all pixels in the image were close enough to an edge to be post-processed in this manner, the relative slowdown was only around 3%.

For experimental evaluation of the performance of our algorithm, we used a large set of test images of different complexity. For this presentation, we choose to focus on one carefully chosen representative contour, shown in Fig. 5. This contour contains edges in all directions with a wide range of positive and negative curvatures, and constitutes a difficult and representative test case. Using our modified distance measure and our proposed algorithm resulted in significant improvement in accuracy over a traditional binary DT, as shown in Fig. 6. The error is greatly reduced even with the simple linear approximation of Eq. 1 (Fig. 6, lower left), and it is further reduced with the more accurate measure of Eq. 4 taking the vector direction into consideration (Fig. 6, lower middle). The remaining errors near edges are significantly reduced by using gradient filters to supply an improved estimate of the edge direction, as presented in Eq. 6 (Fig. 6, lower right).

The average absolute error of our final result, the lower right image of Fig. 6, is 0.02 pixels. This is significantly better than the upper middle image which was computed by a binary EDT on an image of 8×8 higher resolution, and comparable to the upper right image using a binary EDT of 16×16 higher resolution. 99% of the pixels in our result are accurate to within ± 0.2 pixels. A few isolated pixels exhibit absolute errors of up to 0.5. These worst case errors occur where the closest contour has a radius of curvature smaller than one pixel, which invalidates our assumption of a locally straight edge. A region of such worst case errors can be seen extending from the pointed left fin of the dolphin countour. This single point of strong curvature is in fact the main source of error in this test image.

The anti-aliased EDT yields a smoother gradient field, as illustrated in Fig. 8 by the divergence of the gradient of the distance field. While the modulus of the gradient is quite close to 1 at most positions even for a binary EDT, the gradient *direction* deviates significantly from the true direction in an unpredictable manner. There are noticeable directional discontinuities in the computed

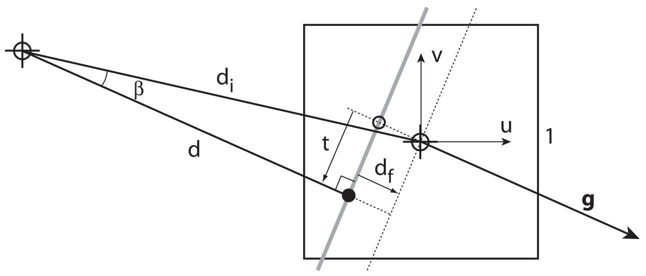

Fig. 4. Notation for determining the true orthogonal distance d to an edge, given d_x, d_y, g_x, g_y, d_f. See also Eq. 6. If the hit point (u, v) (black circle) is inside the pixel, the true distance d is used instead of $d_i + d_f$.

Table 1
Relative performance of traditional vs. new measures. Image size 2048×2048 pixels, test pattern according to Fig. 5, CPU Intel Core2 Duo 2.4 GHz, single thread execution. First row is execution time, second row is relative slowdown.

Binary EDT	AA using Eq. 1	AA using Eq. 4	AA using Eq. 6
0.33 s	0.83 s	3.2 s	3.9 s
1.0×	2.5×	9.7×	11.8×

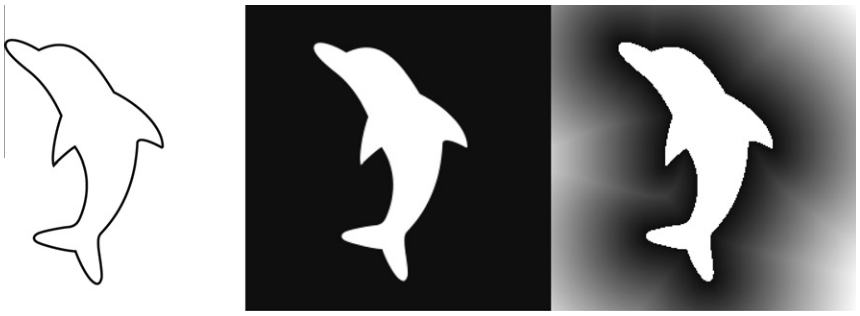

Fig. 5. (left) Test contour for the experiments. (middle) 256 × 256 pixels area-sampled rendering of the contour. (right) Euclidean distance transform of the middle image.

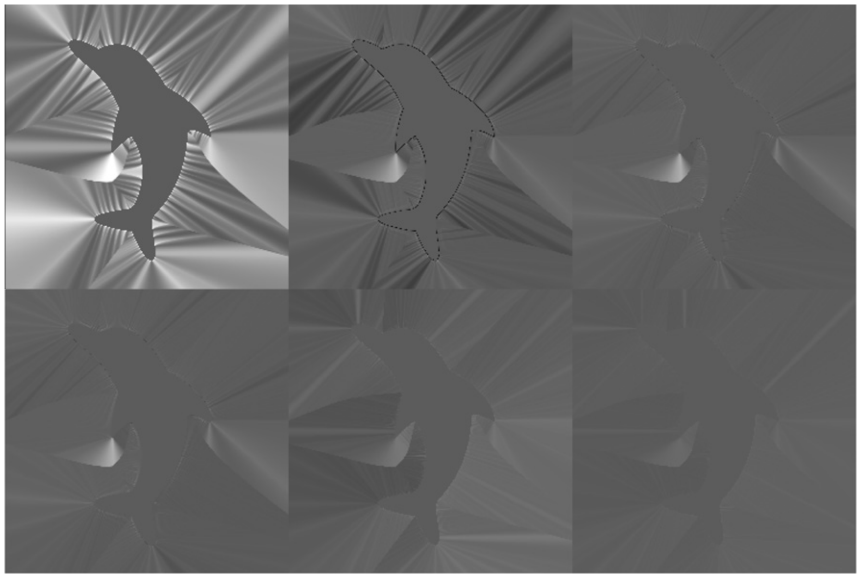

Fig. 6. Error images for 256 × 256 pixels EDT results: transform result minus true distance. Zero is medium gray, white is +1, black is −1. *Top row:* binary EDT, subsampled binary EDT from 8 × 8 higher resolution, subsampled binary EDT from 16 × 16 higher resolution. *Bottom row:* Approximate AA EDT according to Eq. 1, AA EDT according to Eq. 4, further improved AA EDT according to Eq. 6. Errors are discussed in Fig. 7 and in the text.

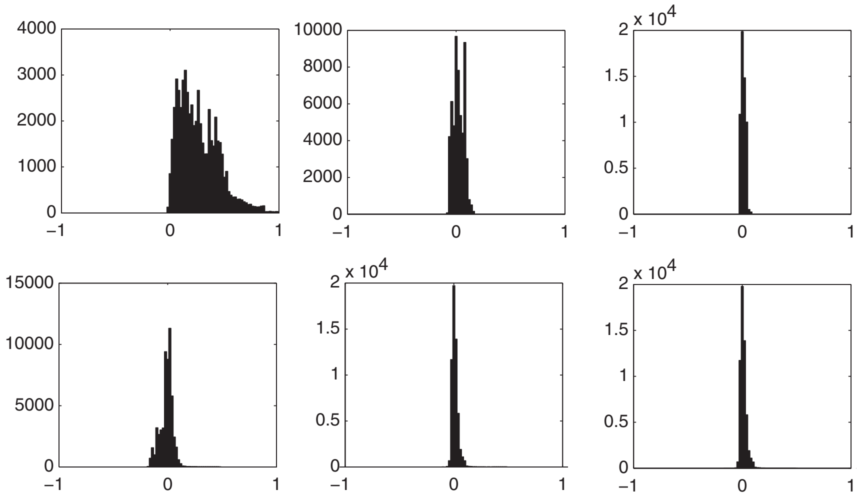

Fig. 7. Histograms for the respective error images in Fig. 6. Systematic errors are eliminated and accuracy is improved by our algorithms. Our final result (lower right) is comparable to using a binary EDT of 16 × 16 higher resolution (upper right). For details, see the text.

distance field even at large distances, as can be seen in Fig. 8, left. These artifacts are reduced to the limit of extinction by using an anti-aliased grayscale input image and our modified distance transform algorithm, see Fig. 8, right.

6. Conclusion

The distance field generated by our improved algorithm has good accuracy even near contour edges. This makes it well suited

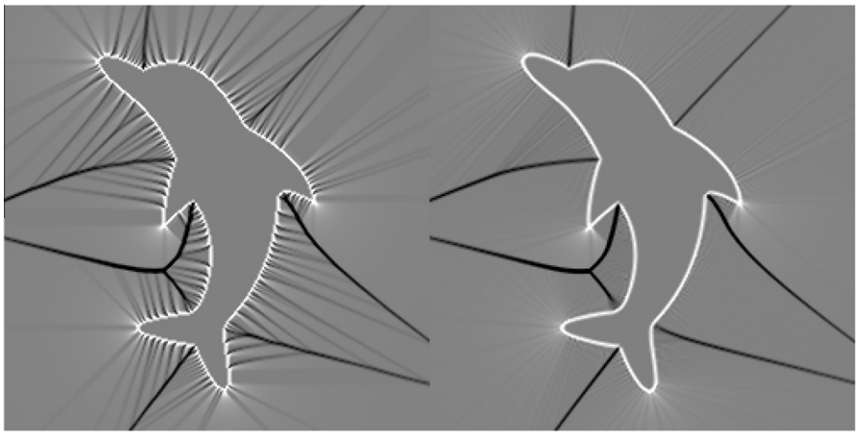

Fig. 8. Divergence of the discrete, filter-estimated gradient of the distance field. White is positive, black is negative, medium gray is zero. *Left:* binary EDT. *Right:* our anti-aliased EDT.

to performing erosion and dilation of contours with fractional and sub-pixel distances.

A distance field with sub-pixel accuracy is also well suited for detailed reconstruction of sharp boundaries of arbitrary shape from a sampled image, while a binary transform is inadequate for this purpose. This reconstruction problem, in fact, was our motivation for developing this transform.

Furthermore, the improved smoothness of the gradient of the generated distance field, as demonstrated by Fig. 8, makes the distance field much better suited for generating level sets, height fields, bump maps, bevels and other data structures for computer graphics where the visual result is dependent on differentials of the distance field.

In theory, the anti-aliased distance transform presented here can be replaced by a traditional binary transform on a higher resolution image, followed by subsampling. This is what we used for the "ground truth" in our experiments to evaluate the accuracy. However, even when a higher resolution binary image can be acquired (which is typically not the case for real world input data), the return in edge position accuracy for each unit of input data is much greater with antialiased grayscale images than with binary images. A linear edge rendered with ideal area sampling and quantized to 8 bits per pixel can theoretically convey the edge position with an accuracy of around 1/255 of a pixel, somewhat depending on the edge position and orientation. If a similar minimum accuracy were to be computed from binary input, the binary image would need about 255×255 as many 1-bit pixels, or $255 \times 255/8 \approx 8000$ times as much data. Even though our results fall short of this theoretical maximum, we show an accuracy equivalent to a 16×16 higher resolution binary EDT, which translates to around $30\times$ savings in execution time and memory requirements. We conclude that it is clearly worthwhile to consider the extra accuracy which can be achieved in a distance transform by utilizing grayscale information where available.

7. Source code

For easy evaluation and testing, we have wrapped our anti-aliased Euclidean distance transform implementations in MEX files for Matlab or GNU Octave. The source code is published under the GPL and can be downloaded from: http://www.itn.liu.se/~stegu/edtaa/.

References

Borgefors, G., 1996. On digital distance transforms in three dimensions. Computer Vision and Image Understanding 64 (3), 368–376.

Chen, W., Lia, M., Sua, X., 2009. Error analysis about ccd sampling in fourier transform profilometry. Optik - International Journal for Light and Electron Optics 120, 652–657.

Danielsson, P.-E., 1980. Euclidean distance mapping. Computer Graphics and Image Processing 14, 227–248.

Eberly, D., Lancaster, J., 1991. On gray scale image measurements: I.arc length and area.. CVGIP: Graphical Models and Image Processing 53 (6), 538–549.

Felzenszwalb, P., Huttenlocher, D., 2004. Distance transforms of sampled functions. Tech. rep., Cornell Computing and Information Science Technical Report TR2004-1963.

Green, C., 2007. Improved alpha-tested magnification for vector textures and special effects. In: SIGGRAPH 2007 course notes. vol. 28. Advanced Real-Time Rendering in 3D Graphics and Games. pp. 9–18.

Huang, C.T., Mitchell, O.R., 1994. A euclidean distance transform using grayscale morphology decomposition. IEEE Trans. Pattern Anal. Mach. Intell. 16 (4), 443–448.

Kimmel, R., Kiryati, N., Bruckstein, A.M., 1996. Sub-pixel distance maps and weighted distance transforms. Journal of Mathematical Imaging and Vision 6 (2-3), 223–233.

Kiryati, N., Bruckstein, A., 1988. Gray-levels can improve the performance of binary image digitizers. In: Computer Vision and Pattern Recognition, 1988. Proceedings CVPR '88., Computer Society Conference on. pp. 562–567.

Köthe, U., 2008. Reliable low-level image analysis. Department of Informatik, University of Hamburg. Habilitation thesis.

Krissian, K., Westin, C.-F., 2005. Fast sub-voxel re-initialization of the distance map for level set methods. Pattern Recognition Letters 26 (10), 1532–1542.

Maurer, C.R., Qi, R., Raghavan, V., 2003. A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions. IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (2), 265–270.

Mehnert, A.J.H., Jackway, P.T., 1999. On computing the exact Euclidean distance transform on rectangular and hexagonal grids. Journal of Mathematical Imaging and Vision 11, 223–230.

Mullikin, J.C., 1992. The vector distance transform in two and three dimensions. CVGIP: Graphical Models and Image Processing 54 (6), 526–535.

Ragnemalm, I., 1992. Fast erosion and dilation by contour processing and thresholding of distance maps. Pattern Recognition Letters 13 (3), 161–166.

Ragnemalm, I., 1993. The Euclidean distance transform in arbitrary dimensions. Pattern Recognition Letters 14 (11), 883–888.

Rosenfeld, A., Pfaltz, J.L., 1966. Sequential operations in digital picture processing. Journal of the ACM 13 (4), 471–494.

Rosenfeld, A., Pfaltz, J.L., 1968. Distance functions on digital pictures. Pattern Recognition 1, 33–61.

Saito, T., Toriwaki, J.-I., 1994. New algorithms for Euclidean distance transformation of an n-dimensional digitized picture with applications. Pattern Recognition 27 (11), 1551–1565.

Schilling, A., 2001. Festschrift zum 60. Geburtstag von Wolfgang Strasser. Wilhelm-Schickard-Institut für Informatik, Graphisch-Interaktive Systeme, Ch. 10: Some notes on sampling in computer graphics, pp. 185–207.

Sethian, J.A., 1999. Level Set Methods and Fast Marching Methods. Cambridge University Press.

Shih, F.-C., Mitchell, O., 1992. A mathematical morphology approach to Euclidean distance transformation. IEEE Transactions on Image Processing 1 (2), 197–204.

Sladoje, N., Lindblad, J., 2009. High-precision boundary length estimation by utilizing gray-level information. IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2), 357–363.

Strand, R., 2007. Weighted distances based on neighbourhood sequences. Pattern Recognition Letters 28 (15), 2029–2036.

Verbeek, P.W., van Vliet, L.J., 1993. Estimators of 2 d edge length and position, 3 d surface area and position in sampled grey-valued images. Bioimaging, 47–61.

Wei, Y., Brainard, R.L., 2009. Advanced Processes for 193-nm Immersion Lithography. SPIE. Ch. 8.