Fast line scan algorithm for discrete Euclidean distance transform in three or more dimensions

Yuriy Mishchenko*

***Abstract*—We describe a fast line scan algorithm for calculations of discrete Euclidean distance transform in dimensions three or higher. Presented algorithm is transparent, conceptually simple, and allows efficient implementations using vector processing architectures. Implemented in Matlab as a plain script, this algorithm outperforms standard compiled Matlab's Image Processing Toolbox's function bwdist in terms of the calculation time by a factor of two. Other than being significantly faster than bwdist, described algorithm also allows processing data of much greater size as well as calculations of Euclidean distance transform for the data with anisotropic pixel aspect ratio.**

I. Introduction

EUCLIDEAN distance transform plays an important role in many image processing applications, and efficient algorithms for calculations of the distance transform are of great importance in image processing [1], [2], [3], [4], [5], [6]. In two dimensions, very fast algorithms have been developed in the past and very good implementations are available in standard software packages such as Matlab [7]. However, in dimension three or higher the situation is much less satisfactory [8], [12], [9], [10], [11]. Here, we develop a fast line scan algorithm for calculations of discrete Euclidean distance transform in any number of dimensions. Presented algorithm is inspired by the linear time line scan approaches of [7] and [12], and is transparent, conceptually simple, and allows efficient implementations using vector processing architectures. Implemented in Matlab as a plain script, our algorithm outperforms standard compiled Matlab's Image Processing Toolbox's function *bwdist* [13] in terms of the calculation time at least by a factor of two. Other than being significantly faster than *bwdist*, our algorithm also allows calculations of the Euclidean distance transform for three dimensional datasets of much greater size as well as calculations of the Euclidean distance transform for datasets with anisotropic pixel aspect ratio. These advances are of significant importance for researches working with three and higher dimensional large datasets with anisotropic pixel aspect ratio, such as in confocal microscopy or serial electron microscopy. Matlab implementation of our algorithm is publicly available at the Matlab Central website under the name of "3D Euclidean Distance Transform for Variable Data Aspect Ratio".

II. Method

With a binary image $\mathcal{I}$ defined on a set of pixels $\mathcal{X}$ we can always associate a set of all "on" pixels $\mathbf{X}$. The discrete Euclidean distance transform of $\mathcal{I}$, then, is defined for every point in $\mathcal{X}$ as the smallest distance from that point to an "on" pixel in $\mathcal{I}$, i.e.

$$D(\mathbf{x}) = \min_{\mathbf{y}\in\mathbf{X}} \|\mathbf{x}-\mathbf{y}\|. \quad (1)$$

To calculate (1) in an arbitrary number of dimensions N, we proceed iteratively by constructing first a series of reduced K-dimensional distance transforms $D_{x^*_{K+1:N}}(x_{1:K})$[1] for different K-dimensional cross-sections of $\mathcal{I}$, and gradually increase K from $K=1$ to $K=N$.

For example, in three dimensions, we first pre-compute two dimensional, $K=2$, distance transforms for all cross-sections of $\mathcal{I}$ in different two dimensional Z-slices $Z=z$, $D_z(x,y)$. Then, for every point (x^*,y^*,z^*) we find a slice $Z=z$ such that $D_z(x^*,y^*)^2+(z^*-z)^2$ is the smallest. Using that slice, the smallest distance from (x^*,y^*,z^*) to $\mathcal{I}$ is then $D_z(x^*,y^*)^2+(z^*-z)^2$, Fig. 1.

In particular, the last point is relatively straightforward to show. Let us assume that a point $\mathbf{x}=(x,y,z)\in\mathcal{I}$ is the closet point to $\mathbf{x}^*=(x^*,y^*,z^*)$ in $\mathcal{I}$. Clearly then $\|\mathbf{x}^*-\mathbf{x}\|^2=(z^*-z)^2+\|(x^*,y^*)-(x,y)\|^2$, so it only remains to show that (x,y) is also the closest point in $\mathcal{I}$ to (x^*,y^*) in respective Z-slice, $Z=z$. Indeed, if there was another point $(x',y')\in\mathcal{I}$ in slice $Z=z$ that was yet closer to (x^*,y^*), then

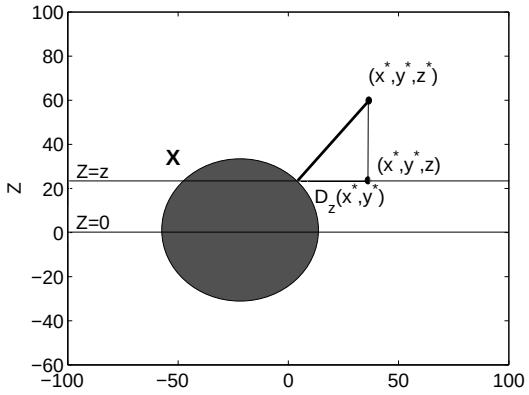

Fig. 1. Euclidean distance in K-dimensions from a point $\mathbf{x}^*_K$ to an object $\mathcal{I}$ can be always represented in terms of the distance from $\mathbf{x}^*_K$ to a certain $(K-1)$-dimensional slice $x_K=y$, and the distance within that slice to $\mathcal{I}$. For example, in three dimensions, the distance from point (x^*,y^*,z^*) to $\mathcal{I}$ can be represented using distance from (x^*,y^*,z^*) to (x^*,y^*,z) for some Z-slice, $Z=z$, and distance from (x^*,y^*,z) to the corresponding cross-section of $\mathcal{I}$ in that slice.

Y. Mishchenko is with the Department of Electrical and Electronics Engineering, School of Engineering, Toros University, 33140, Mersin, Turkey, e-mail: yuriy.mishchenko@gmail.com.

[1] As is customary, we use bold letters to identify vector variables and ":"-notation to denote ranges of variables, $x_{A:B}=(x_A,x_{A+1},x_{A+2},\ldots,x_B)$.

(x', y', z) would be also closer to (x^*, y^*, z^*), thus showing that $D(x^*, y^*, z^*) = \min_z [D_z(x^*, y^*)^2 + (z^* - z)^2]$.

To describe the above process formally, let us first define restriction operator $\pi_{x^*_{K+1:N}} : \mathbb{R}^N \to \mathbb{R}^K$,

$$\pi_{x^*_{K+1:N}} \mathbf{X} = \{\mathbf{y} \in \mathbb{R}^K \ s.t. \ (y_{1:K}, x^*_{K+1:N}) \in \mathbf{X}\}. \quad (2)$$

$\pi_{x^*_{K+1:N}}$ formalizes the notion of a K-dimensional cross-section of $\mathcal{I}$ inside a K-dimensional slice with coordinates $x_{K+1} = x^*_{K+1}, x_{K+2} = x^*_{K+2}, \ldots, x_N = x^*_N$. Then, reduced K-dimensional distance transform $D_{x^*_{K+1:N}}(\mathbf{x}^*_K)$ will be defined as the distance transform of $\pi_{x^*_{K+1:N}} \mathbf{X}$ in the K-dimensional slice $x_{K+1} = x^*_{K+1}, x_{K+2} = x^*_{K+2}, \ldots, x_N = x^*_N$,

$$D_{x^*_{K+1:N}}(\mathbf{x}^*_K) = \min_{\mathbf{y} \in \pi_{x^*_{K+1:N}} \mathbf{X}} \|\mathbf{y} - \mathbf{x}^*_K\|, \quad (3)$$

where we used notation $\mathbf{x}_K = x_{1:K} = (x_1, x_2, \ldots, x_K)$.

Analogous to the above considered example, $D_{x^*_{K+1:N}}(\mathbf{x}^*_K)$ for different K are related via a recursive relationship,

$$D_{x^*_{K+1:N}}(\mathbf{x}^*_K)^2 = \min_y [|x^*_K - y|^2 + D_{(y, x^*_{K+1:N})}(\mathbf{x}^*_{K-1})^2]. \quad (4)$$

We observe, therefore, that reduced K-dimensional distance transforms can be constructed iteratively by first building all reduced $(K-1)$-dimensional distance transforms, and then constructing the lower envelops of families of one dimensional parabolas,

$$F_K[y](x^*_K) = |y - x^*_K|^2 + D_{y, x^*_{K+1:N}}(\mathbf{x}^*_{K-1})^2. \quad (5)$$

Such lower envelop can be constructed efficiently using the following Lemma:

Lemma 1: Let $\mathcal{G}(x)$ be a lower envelop of a finite family of one dimensional parabolas $\mathcal{F} = \{F[y](x) = |x - y|^2 + g(y)\}$, and let f be a parabola $f(x) = x^2 + b$. Then, assuming that $\mathcal{G}(x)$ and $f(x)$ are not identically equal, there exist at most two points such that $\mathcal{G}(x) = f(x)$, and these are achieved exactly on at most two parabolas from the family, $F[y^1](x)$ and $F[y^2](x)$, with $y^1 \leq 0$ and $y^2 \geq 0$.

Proof: To prove the above lemma, let us assume that x^* is such a point that $\mathcal{G}(x^*) = f(x^*)$. Then, there exist a parabola in $\mathcal{F}$ such that $F[y](x^*) = f(x^*)$ for some y. By an obvious property of the intersection of two parabolas, $F[y](x) < f(x)$ on a semi-interval $x > x^*$ or $x < x^*$. Specifically, if $y < 0$, then $F[y](x) < f(x)$ for all $x > x^*$, and, if $y > 0$, $F[y](x) < f(x)$ for all $x < x^*$. Since $\mathcal{G}(x)$ is the lower envelope of $\mathcal{F}$, it follows also that $f(x) > F[y](x) \geq \mathcal{G}(x)$ on such respective intervals. If x^1 now is a solution of $\mathcal{G}(x^1) = f(x^1)$, without loss of generality, achieved on parabola $y^1 < 0$, and there exist another solution x^2, $\mathcal{G}(x^2) = f(x^2)$, achieved on parabola $y^2 < 0$, then $f(x) > \mathcal{G}(x)$ on $x > x^1$ and simultaneously $f(x) > \mathcal{G}(x)$ on $x > x^2$. Suppose now that $x^1 < x^2$, then $x^1 < x^2$ also implies that $f(x^2) > \mathcal{G}(x^2)$, leading to contradiction with $f(x^2) = \mathcal{G}(x^2)$. Therefore, there can be at most two solutions to $\mathcal{G}(x) = f(x)$ achieved on two parabolas from $\mathcal{F}$ with $y^1 \leq 0$ and $y^2 \geq 0$, respectively. ■

Corollary 1: A parabola $f(x)$ can be below the envelop $\mathcal{G}(x)$ on at most a single interval (x^1, x^2).

Intuitively, if two parabolas intersect, they can do so at most at a single point (except if the two parabolas are identical), and on one side of the intersection one of the parabolas should always be above the other. Thus, once the envelop $\mathcal{G}(x)$ is crossed by $f(x)$ at some x, by crossing one of the parabolas $F_K[y](x)$ in $\mathcal{F}$, there can be no other crossings for an entire interval $x > x_1$ or $x < x_1$ because $f(x)$ will lie entirely above the respective $F_K[y](x)$ and, therefore, $\mathcal{G}(x)$.

Using the above lemma, the following procedure can be suggested for constructing envelops $\mathcal{G}(x)$ efficiently. Let $\mathbf{G}$ be a subfamily of $\mathcal{F}$ containing m parabolas, and let $\mathcal{G}_{\mathbf{G}}(x)$ be the lower envelop of the parabolas in $\mathbf{G}$. Let $\mathbf{G}' = \mathbf{G} \bigcup \{F[y^{m+1}](x)\}$ for some new y^{m+1}. Observe that the new envelope $\mathcal{G}_{\mathbf{G}'}(x)$ can differ from $\mathcal{G}_{\mathbf{G}}(x)$ at most on an interval (x^1, x^2), by the above corollary, such that $F[y^{m+1}](x) < \mathcal{G}_{\mathbf{G}}(x)$. In order to find such an interval, let x^0 be any point such that $F[y^{m+1}](x^0) < \mathcal{G}_{\mathbf{G}}(x^0)$. Then, we can efficiently recover the interval by finding two intersections, $F[y^{m+1}](x^{1,2}) = \mathcal{G}_{\mathbf{G}}(x^{1,2})$, in the left and the right semi-axes of x^0. In particular, if the algorithm maintains the parameter $Y(x)$ of the parabola achieving the lower envelop at each point, $\mathcal{G}(x) = F[Y(x)](x)$, then a good guess about the intersections can be made by solving a simple linear equation $F[y^{m+1}](x) = F[Y(x)](x)$. By repeating the above process until all parabolas in $\mathcal{F}$ are exhausted, $\mathcal{G}(x)$ can be constructed. Note that $\mathcal{G}(x)$ is constructed in this process at once for all x_K. That is, the above process does not need to be performed for each value of x^*_K separately, $D_{x^*_{K+1:N}}(\mathbf{x}^*_K)$ can be produced from $D_{x^*_{K:N}}(\mathbf{x}^*_{K-1})$ for all x^*_K at the same time.

Our algorithm allows for trivial modifications to perform calculations of the distance transform for the datasets with anisotropic pixel aspect ratio, such as typically obtained with confocal light or electron microscopy imaging. In these applications, the images are typically acquired with high X and Y resolution but Z resolution is much worse, leading to pixels effectively elongated in Z direction. If pixel aspect ratio in such an image in K^{th} dimension is a_K, by replacing (5) with

$$F_K[y](x^*_K) = a_K |y - x^*_K|^2 + D_{y_K, x^*_{K+1:N}}(\mathbf{x}^*_{K-1})^2, \quad (6)$$

our algorithm can be obviously adopted to calculations of the distance transform for datasets with arbitrary pixel aspect ratios.

III. Results

We tested described algorithms on random arrangements of simple geometric figures such as spheres, triangles and squares, see Fig. 2. Implemented as a plain script function in Matlab, our algorithm showed a factor of two better calculation time than standard Matlab's Image Processing toolbox's function *bwdist*, version 7.7, utilizing nearest neighbor search on an optimized kd-tree [13], Fig. 3. Overall, our algorithm implemented as a Matlab function showed excellent performance, not only performing calculations of the distance transform for different 3D arrangements faster than built-in *bwdist*, but also for 3D images of significantly larger size and the datasets with anisotropic pixel aspect ratio.

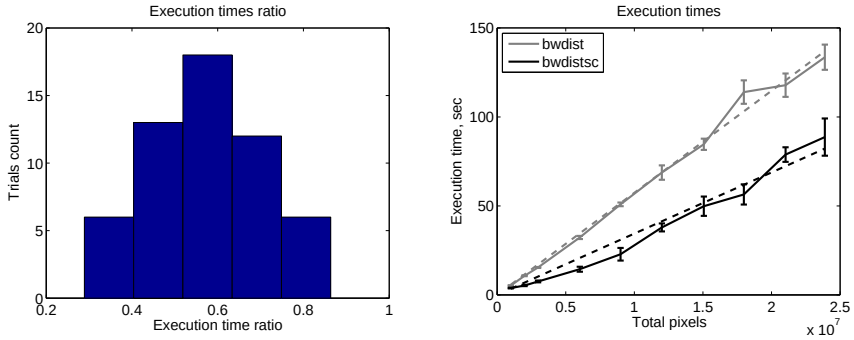

Fig. 3. Histogram of execution time ratios for *bwdist* and this algorithm, for a sample of random 3D arrangements of simple geometrical shapes (left), and plot of execution times versus the total number of pixels in the image (right). While both *bwdist* and this algorithm demonstrated linear time scaling, *bwdist* on average required at least two times more time than the described algorithm.

Algorithm 1 Line scan algorithm for calculating discrete Euclidean distance transform.

Set $K = 1$.
while $K \leq N$ **do**
  Assume $D_{x^*_{K:N}}(\mathbf{x}^*_{K-1})$ had been calculated for all $\mathbf{x}^*$ ($D_{x^*_{1:N}}(\mathbf{x}^*_0) = 0$).
  for all $(x^*_{1:K-1}, x^*_{K+1:N})$ **do**
    Set $F[y](x) = |y - x|^2 + D_{y,x^*_{K+1:N}}(\mathbf{x}^*_{K-1})^2$.
    Set $\mathbf{F} = \{F[y](x) \text{ for all } y\}$.
    Set $\mathcal{G}(x) = +\inf$ and set $\mathbf{G} = \emptyset$.
    while $\mathbf{G} \neq \mathbf{F}$ **do**
      Select a parabola $F[y] \in \mathbf{F} \setminus \mathbf{G}$.
      Select an initial point x^0 such that $F[y](x^0) < \mathcal{G}(x^0)$ (Algorithm 2).
      Scan x away from x^0 in positive and negative direction and set $\mathcal{G}(x) = F[y](x)$ as long as $F[y](x) < \mathcal{G}(x)$.
      Set $\mathbf{G} = \mathbf{G} \bigcup \{F[y]\}$.
    end while
    Set $D_{x^*_{K+1:N}}(x^*_{1:K-1}, x^*_K) = \mathcal{G}(x^*_K)$.
  end for
  Set K=K+1.
end while

Algorithm 2 Selecting initial point for the line scans.

For a parabola $F[y^m](x)$ and a current lower envelop $\mathcal{G}(x)$.
Set $n = 0$ and $x^0 = y^m$.
while $F[y^m](x^{(n)}) > \mathcal{G}(x^{(n)})$ and $x^{(n)}$ is inside the image domain **do**
  Solve $F[y^m](x) = F[Y(x^{(n)})](x)$.
  Set $x^{(n+1)}$ to the found solution.
  Set $n = n + 1$.
end while
If $x^{(n)}$ is outside of the allowed domain, then initial point does not exist — $F[y^m](x)$ is completely above $\mathcal{G}(x)$ — otherwise $x^{(n)}$ is the initial point.

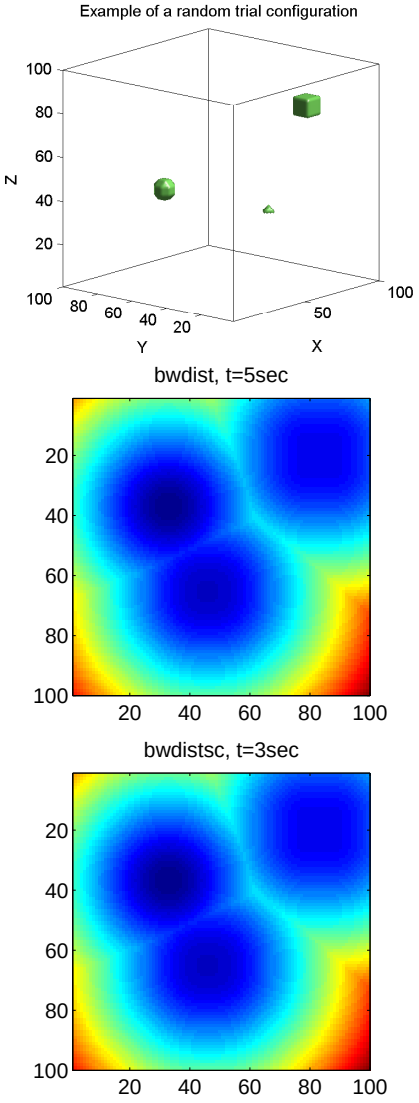

Fig. 2. An example of a random configuration of three simple geometrical shapes — a sphere, a square, and a tetrahedra (top) — and a slice from the corresponding 3D Euclidean distance transform calculated using Matlab's Image Processing Toolbox's standard function *bwdist* (middle) and this paper's line scan algorithm (*bwdistsc*, bottom). The time required for the calculation on a dual core 2GHz laptop is also shown for each case.

IV. Summary

We describe a fast line scan algorithm for computations of discrete Euclidean distance transform for images in any number of dimensions. Described algorithm calculates the distance transform iteratively, first by computing the reduced one-dimensional distance transform for all 1D-slices of the object, than by computing reduced two-dimensional distance transforms for all 2D-slices of the object, and so on. Each iteration step reduces to finding of a lower envelop for a finite family of one dimensional parabolas of the form $\{F[y](x) = |x - y|^2 + g(y)\}$. Using the lemma proved in the text, such envelop can be constructed efficiently for all x.

The algorithm for calculation of Euclidean distance transform, presented in this paper, is transparent and conceptually simple, and can be efficiently implemented using vector processing architectures. We implemented described algorithm in Matlab as a plain script function. In tests performed using random 3D configurations of simple geometric shapes, this function outperformed by a factor of two analogous built-in Matlab's Image Processing Toolbox's function *bwdist*, version 7.7. Other than being substantially faster, our algorithm also allowed calculating distance transforms for 3D datasets of much larger size than *bwdist*, and also allowed calculating distance transforms for images with anisotropic pixel aspect ratio. These advantages are of significant importance for researches working with large three and higher dimensional datasets with anisotropic pixel aspect ratio, such as obtained in confocal microscopy or serial electron microscopy applications. Matlab implementation of our algorithm is publicly available at the Matlab Central website under the name of "3D Euclidean Distance Transform for Variable Data Aspect Ratio".

References

[1] Paglieroni, D., Distance Transforms : Properties and Machine Vision Applications, Graphical models and image processing, 1992, **54**, 56-74.

[2] Baggett, D., Nakaya, M., McAuliffe, M. *et al*, Whole cell segmentation in solid tissue sections, Cytometry Part A, 2005, **67**, 137-143.

[3] Rosenfeld, A., Pfaltz, J., Sequential operations in digital picture processing, Journal of the ACM, 1966, **13**, 471-94.

[4] Rousson, M., Paragios, N., Deriche, R., Implicit active shape models for 3D segmentation in MR imaging, Medical Image Computing and Computer-Assisted Intervention, 2004, **3216**, 209-216.

[5] Town C., Sinclair D., Content Based Image Retrieval using Semantic Visual Categories, Technical report, 2001.

[6] Gavrila, D. M., Pedestrian Detection from a Moving Vehicle, In *Computer Vission - ECCV 2000*, Springer Berlin / Heidelberg, 2000, 37-49

[7] Breu H., Gil J., Kirkpatrick D., Werman M., Linear Time Euclidean Distance Transform Algorithms, IEEE Transactions on Pattern Analysis and Machine Intelligence, **17**, 529-533.

[8] Maurer C., Qi R., and Raghavan V., 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**, 265-270.

[9] Tustison N. J., Siqueira M., Gee J. C., N-D Linear Time Exact Signed Euclidean Distance Transform, The Insight Journal, 2006, http://hdl.handle.net/1926/171.

[10] Guan W., Ma S., A list-processing approach to compute Voronoi diagrams and the Euclidean distance transform, Pattern Analysis and Machine Intelligence, IEEE Transactions on, **20**, 757-761.

[11] Lucet, Y., A linear Euclidean distance transform algorithm based on the linear-time Legendre transform, Computer and Robot Vision, 2005. The 2nd Canadian Conference on, 262-267.

[12] Meijster A., Roerdink J., Hesselink W., A General Algorithm for Computing Distance Transforms in Linear Time, In *Mathematical Morphology and its Applications to Image and Signal Processing*, Springer US, 2002, 331-340.

[13] Friedman J. H., Bentley J. L., and Finkel R. A., An Algorithm for Finding Best Matches in Logarithmic Expected Time, ACM Transactions on Mathematics Software, **3**, 209-226.