STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 2, 2011

NUMERICAL COMPUTATION METHOD OF THE GENERAL DISTANCE TRANSFORM

SZIDÓNIA LEFKOVITS[(1)]

Abstract. The distance transform is a mathematical operator with a wide range of applications in computer vision. In the state-of-the art the mostly discussed distance transform is the Euclidean distance transform of binary images. In this paper we propose an algorithm for general distance transform of sampled functions. In our method the only restriction is that the defined distance has to be an increasing function for each component. The proposed algorithm is compared with the low parabolas algorithm which is one of the most performant. Experimental results show advantages in computation speed even for the Euclidean distance transform of high resolution images. The most important property of our method is the usability for any kind of distances.

1. Introduction

The distance transform is defined as a mathematical operator which computes the distance map of an image. In the classical formulation only binary images are considered. Thus, the distance transform is an image D obtained from the original image I, where each pixel value is the nearest distance from this to the object O.

$$D(y) = \min_{x \in O}\{d(x, y) | y \in I\} \tag{1}$$

Numerous applications of the distance transform are known in computer vision, image analysis, pattern recognition, shape analysis, feature detection techniques. It can also be associated with the shortest-path algorithm, medial axes or skeleton extraction and other image segmentation techniques.

The paper is organized as follows: In the next section the most important distance transform algorithms are summarized. The third section presents the formal definition for understanding the theoretical background. The fourth

Received by the editors: March 15, 2011.

2010 *Mathematics Subject Classification.* 68, 68T45.

1998 *CR Categories and Descriptors.* I.4.10 [**Artificial Intelligence**]: Image Processing and Computer Vision – *Image Representation.*

Key words and phrases. general distance transform, Euclidean distance transform, binary search tree, sampled functions.

section illustrates the proposed algorithm presenting an example and the pseudocode implementation. In the last section the algorithm is compared with the low parabolas algorithm.

2. Related work

There are many ways to compute the distance transform. The trivial method which is valid for all types of distances is called the brute force algorithm. This algorithm computes for each background pixel its distance to all the object pixels and out of these it selects only the minimum.

Instead of the brute force algorithm a lot of scientists have tried to solve this algorithm in a faster way. Fabbri et al. in their survey [1] makes a classification and compare the most important algorithms. They classify the existent algorithms in three categories.

The propagation algorithms proposed by Eggers[2] and Cuisenaire [3] compute the distance of the background pixels, starting form the boundary pixels of the object, from the closest to the farthest.

The raster scanning algorithms, first used by Rosenfeld et al.[4] and developed by Borgefors et al. [5] and Daniellson [6] , compute the distance transform algorithms supposing that a value of a pixel can be computed from the value of distance of its neighbours.

The independent scanning algorithms compute the distance transform by scanning the image in each direction. Paglieroni [7] extends Rosenfeld et al.'s method[4], by independent scans for each direction appliable for Euclidean distance transform too. In the same category some morphological operators can were used by Shih at al. [8] and Lotufo et al.[9]. One of the most interesting algorithms are based on parabola intersection, by constructing the lower envelope of the parabolas invented by Meijster et al.[10]. This has been further developed by Felzenszwalb et al. [11] in order to compute l_1 and l_2 distance transform of sampled functions which give a faster solution to the cost minimization problem.

3. Theoretical background

In the previous section the distance transform was defined in the classical way, but in several applications it is useful to combine the distance with an other property. This extends the definition (1) with an additive term $a(x)$. We define the measure function as: $M : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_+$, where $\mathcal{P}$ is a finite discrete domain.

$$M(x, y) = d(x, y) + a(x) \tag{2}$$

The distance transform is defined as the minimum of the measure function $D : \mathcal{P} \to \mathbb{R}_+$

$$D_a(y) = \min_{x \in \mathcal{P}} M\left(x, y\right) = \min_{x \in \mathcal{P}} \left(d\left(x, y\right) + a\left(x\right)\right) \tag{3}$$

In this article the functions take arbitrary positive real values, in order to be summable with the distance function. Several methods can not handle this additional term.

Our algorithm is able to compute the distance transform for any kind of distances with the same restrictions given by Paglieroni [7]:

(4)

$$d(x, y) = f(|x_1 - y_1|, |x_2 - y_2|, \ldots, |x_n - y_n|)$$

$$|x_i - y_i| < |z_i - t_i| \Rightarrow f(m_1, \ldots, |x_i - y_i|, \ldots, m_n) < f(m_1, \ldots, |z_i - t_i|, \ldots, m_n)$$

These properties mean that the distance function is increasing for each component separately. These conditions are valid for usual distances. Our algorithm computes the distance transform using the independent scanning method, thus the distance is computed in each dimension independently.

4. Presentation of the algorithm

In this section we propose an algorithm adequate for computing the general distance transform (3). The main idea comes form the observation that in order to find the minimum of the value of the measure function (2) in a point y it is not necessary to compute all the possible distances $d(x, y)$ to it. The points x which can be eliminated, are those in which the value of the dissimilarity function $a(x)$ is greater than the actual value and are further than the actual point. It is assumed that the distance is an increasing function, thus the order relation between the arguments assigns the same order for the values (4). The dissimilarity function values are ordered increasingly and we operate only with the indices of them: $a(1), a(2), \ldots a(n)$.

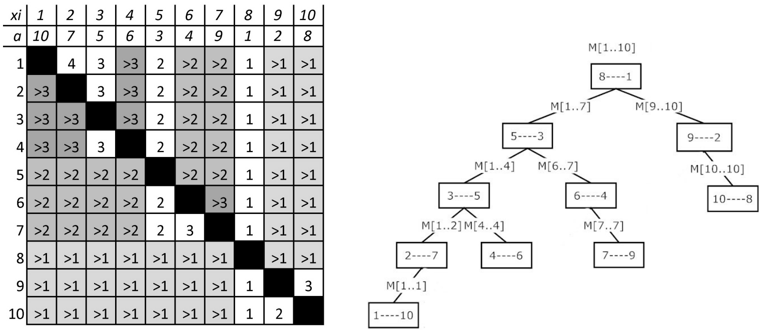
The algorithm principle is the sequential determination of the potential minimum points for each reference point. The reference point y is the point for which the distance transform is computed. The next potential minima have to satisfy the following two conditions:

$$(5) \qquad d(y, x_{pm_i}) > d(y, x_{mp_{i+1}}) \qquad\qquad (6) \qquad a(x_{mp_i}) < a(x_{mp_{i+1}})$$

Based on these conditions all the potential minimum points can be computed from the reference point. The value of the distance transform will give the minimum of the measure function computed only in the potential minimum points. Thus the algorithm sequentially computes the potential minima set for each reference point.

The algorithm implementation resides form the observation that every potential minimum point splits one matrix into four parts (figure 1(a)). Two of these matrices represent the elements which are not taken into consideration (shaded points in the figure), because the measure function value of these is surely greater than the value of the split point. The other two matrices are split recursively by the next minimum, which satisfies the condition . Each potential minimum point splits one matrix in the same way.

This algorithm suggests a recursive implementation. The search for the minimum in the main matrix can be traced back to the minimum of the superior and inferior matrices. This method can be rephrased with binary search tree (figure 1(b)). The nodes of the tree are inserted in the ascending order of the dissimilarities and their final position in the tree is determined by their distance to the position of the previous split. The root is always the $a(1)$ and every element $a(i)$ is inserted into the left subtree, if $position(a(i)) < position(a(i_{parent}))$ or into the right subtree otherwise. Thus every node represents a split in the matrix. The construction of the tree ends with the insertion of all the potential minimum points. From the preorder traversal of the tree the distance transform can be computed. In each node we have to compute the distance transform only for the indicated interval (see figure 1(b)).



xi	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
a	*10*	*7*	*5*	*6*	*3*	*4*	*9*	*1*	*2*	*8*
1		4	3	>3	2	>2	>2	1	>1	>1
2	>3		3	>3	2	>2	>2	1	>1	>1
3	>3	>3		>3	2	>2	>2	1	>1	>1
4	>3	>3	3		2	>2	>2	1	>1	>1
5	>2	>2	>2	>2		>2	>2	1	>1	>1
6	>2	>2	>2	>2	2		>3	1	>1	>1
7	>2	>2	>2	>2	2	3		1	>1	>1
8	>1	>1	>1	>1	>1	>1	>1		>1	>1
9	>1	>1	>1	>1	>1	>1	>1	1		3
10	>1	>1	>1	>1	>1	>1	>1	1	2	

(a) potential minima (b) corresponding binary search tree

FIGURE 1. Principle of the algorithm

5. APPLICATIONS AND EXPERIMENTS

Exemplifying the functionality of the algorithm we refer to figure 1(a). In this case the general distance transform is determined for 10 points. The first row represents the natural order of the distances, the second row represents the ordered values of the dissimilarity function, the matrix represents the method of the calculus.

Algorithm 1 DT*(a,d)*

```
SORT($[a,d]$) {ascending order of $a$}
$tree \leftarrow$ CREATE($[a,d](1)$)
for $i \leftarrow 2..n$ do
   INSERT($tree, [a,d](i)$)
end for
$min \leftarrow$ COMPUTEMINIMUM($tree, 1, dim([a,d])$) {preorder traversal}
```

Subalgorithm 1a COMPUTEMINIMUM(*tree, idx_topleft, idx_rightbottom*)

```
if $\exists$ $node$ then
   for all $y \geq idx\_topleft$ and $y \leq idx\_rightbottom$ do
      $tf \leftarrow$ COMPUTEDT($node.a, dist(node.x, y)$); $min \leftarrow$ UPDATEMINIMUM($tf$)
   end for
end if
return $min$
```

The points of principal diagonal are the consecutive reference points . The potential minimum points are indicated on white background, numbered in the order of appearance. The grey background elements need not be effectively evaluated as described above. The proposed algorithm (1) was compared with the one of the best distance transform algorithm proposed by Meijster [10], known as the low parabolas algorithm. An exhaustive implementation of this can be found in the technical report[11] proposed by Felzenszwalb. The most recent survey of distance transform [1] compares the best-known algorithms. These algorithms were defined and implemented based on the Euclidean distance transform, with only few of them based on the chess-board or city-block distances. According to the described experiments in [1], the Meijster algorithm is one of the most performant. Also here the measures are made only for the Euclidean distance transform. Thus we compared our algorithm with these measures. The presented results are made in the same circumstances using the same hardware. Both of the algorithms are implemented in MatLab. In this case not the absolute values in seconds are relevant, but the comparative results. The measures are made for different image sizes varying the number of points. For every size a set of N ($N = 50$) grayscale images are generated randomly. The measured time represents the average computation time of the distance transform for these images. The $1D$ results are shown in table (1). Due to the experiments we noticed that for large values of n, our proposed algorithm computes faster than that of Meijster's. Our algorithm is based on the construction of a binary tree, so the complexity of it is $\theta(n \log n)$, that of Meijster's is "almost" linear, but from the measures it results, that for

$n > 5000$ our algorithm is faster.

Table 1. Performance of EDT

No. of points	Low parabolas[s]	Proposed algorithm[s]
20	0.02293	0.049972
100	0.027082	0.115051
1000	0.256203	0.765405
5000	7.51091	6.693865
10000	32.600791	21.660176
100000	3149.846706	1940.046934

Table 2. Performance of GDT

No. of points	Low parabolas [s]	Proposed algorithm [s]
20	0.420737	0.048204
100	1.169198	0.105476
1000	12.223505	0.74884
5000	80.379097	6.693865
10000	184.046395	24.87107

The most important advantage of our algorithm is not needing to compute the intersection point of parabolas. For arbitrary distances the graphical representation is not obvious and the intersection points even less so. In this case the lower envelope of curves can be computed by only using numerical methods for determining the intersection point of curves. Consequently, the proposed algorithm is much more efficient (see table (2)) in this case. When there are many equal values, the binary tree becomes very unbalanced. The algorithm can be improved by building a balanced tree.

6. Conclusion

The most important advantages of our algorithm is the applicability for any distance. There is no need to compute the intersection of curves in order to determine the lower envelope. In most cases, the solving of the equation is very time consuming. The bottleneck of our algorithm is the construction of the tree. But in several applications we can use the same tree several times for more images, for example, in image retrieval or detection in video sequences. Based upon the experiments made, we consider our algorithm one of the most performant for general distance transform. The general distance transform is often used in deformable object detection, or when a special distance function needs to be used.

References

[1] R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno, “2d euclidean distance transform algorithms: A comparative survey,” *ACM Comput. Surv.*, vol. 40, pp. 2:1–2:44, February 2008.

[2] H. Eggers, “Two fast euclidean distance transformations in z2based on sufficient propagation,” *Computer Vision and Image Understanding*, vol. 69, no. 1, pp. 106 – 116, 1998.

[3] O. Cuisenaire, *Distance Transformations: Fast Algorithms and Applications to Medical Image Processing.* PhD thesis, Universite Chatolic de Louvain, 1999.

[4] A. Rosenfeld and J. Pfaltz, "Distance functions on digital pictures," *Pattern Recognition*, vol. 1, no. 1, pp. 33 – 61, 1968.

[5] G. Borgefors and I. Nyström, "Efficient shape representation by minimizing the set of centres of maximal discs/spheres," *Pattern Recognition Letters*, vol. 18, no. 5, pp. 465 – 471, 1997.

[6] P. E. Daniellson, "Euclidian distance mapping," *Computer Vision Graphics and Image Processing*, vol. 14, pp. 227–248, 1980.

[7] D. W. Paglieroni, "Distance transforms: Properties and machine vision applications," *CVGIP: Graphical Models and Image Processing*, vol. 54, no. 1, pp. 56 – 74, 1992.

[8] F. Shih and Y.-T. Wu, "The efficient algorithms for achieving euclidean distance transformation," *Image Processing, IEEE Transactions on*, vol. 13, no. 8, pp. 1078 –1091, 2004.

[9] R. A. Lotufo and F. A. Zampirolli, "Fast multidimensional parallel euclidean distance transform based on mathematical morphology," *Graphics, Patterns and Images, SIBGRAPI Conference on*, vol. 0, p. 100, 2001.

[10] A. Meijster, J. B. T. M. Roerdink, and W. H. Hesselink, "A general algorithm for computing distance transforms in linear time," in *Mathematical Morphology and its Applications to Image and Signal Processing* (J. Goutsias, L. Vincent, and D. S. Bloomberg, eds.), vol. 18 of *Computational Imaging and Vision*, pp. 331–340, Springer US, 2002.

[11] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," tech. rep., Cornell Computing and Information Science, 2004.

[1] "Petru Maior" University, Tîrgu-Mureş, Romania
E-mail address: szidonia.lefkovits@science.upm.ro