2010 International Conference on Pattern Recognition

Incremental Distance Transforms (IDT)

Theo E. Schouten
Institute for Computing and Information Sciences (ICIS),
Radboud University Nijmegen
P.O. Box 9104
6500 HE Nijmegen, The Netherlands
t.schouten@cs.ru.nl

Egon L. van den Broek
Human Media Interaction (HMI),
University of Twente
P.O. Box 217
7500 AE Enschede, The Netherlands
Human-Centered Computing Consultancy
http://www.human-centeredcomputing.com/
Vienna, Austria
vandenbroek@acm.org

***Abstract*—A new generic scheme for incremental implementations of distance transforms (DT) is presented: Incremental Distance Transforms (IDT). This scheme is applied on the city-block, Chamfer, and three recent exact Euclidean DT (E^2DT). A benchmark shows that for all five DT, the incremental implementation results in a significant speedup: $3.4 \times -10\times$. However, significant differences (i.e., up to $12.5\times$) among the DT remain present. The FEED transform, one of the recent E^2DT, even showed to be faster than both city-block and Chamfer DT. So, through a very efficient incremental processing scheme for DT, a relief is found for E^2DT's computational burden.**

***Keywords*-Euclidean distance, Fast Exact Euclidean Distance (FEED), distance maps/transforms, incremental implementation, Incremental Distance Transforms (IDT), video processing**

I. Introduction

A distance transformation (DT) [1] converts a binary image into a distance image (DI). In such a DI, the value of each pixel represents its distance to the set O of object pixels o in the binary image. Such a DT is calculated as follows:

$$\mathrm{DI}(b) = \min_{o \in O} \mathrm{D}(b, o), \quad (1)$$

where D can be any metric and b is a background pixel.

Although Equation 1 is straightforward, it is hard to develop an algorithm that calculates the DT in a fast manner [2], [3], [4], [5], [6]. Rosenfeld and Pfaltz [1] introduced the first fast algorithms for the city-block (D_4) and chessboard distance (D_8) metrics. Two decades later, Borgefors [7] extended them to Chamfer DT (D_C), which provide better approximations to the L2 metric. The D_4, D_8, and D_C have in common that they all use raster scans over the image to propagate distance using local information only. As the tiles of the Voronoi diagram are not always connected sets on a discrete lattice, the exact Euclidean distance (ED) cannot be obtained by raster scans.

The last decade, various algorithms have been presented to obtain exact ED transforms (EDT); e.g.,. [4], [5], [6]. We will now introduce three of the most promising EDT. Maurer, Qi, and Raghavan [8] introduced an EDT for arbitrary dimensions (D_M). More recently, Lucet [9] presented several sequential exact EDT algorithms, based on fundamental transforms of convex analysis; e.g., the LLT algorithm. Both approaches use dimensional decomposition to process each row independently of the other rows. Next, they process each column of the resulting image independently of the other columns, to produce the final DI.

An alternative approach for EDT was introduced by Schouten and Van den Broek [10]: the Fast Exact Euclidean Distance (FEED) transformation. With FEED, each object pixel feeds its ED to all pixels in the image, which in turn calculate the minimum of the received EDs. They showed that by reducing the number of pixels that must be processed, a very fast exact EDT can be obtained. Nowadays, DT are not only used for image processing (e.g., [11], [12]) but are also used to process video sequences; e.g., in robot navigation or video surveillance applications [13]. These videos often contain large sequences of frames with stationary objects and one or more moving objects. In this paper, we develop a new class of algorithms, the Incremental Distance Transforms (IDT), to calculate the DI for such frames in a fast way by using the DI of the stationary objects.

In the next section, the principle of IDT will be described through its application on: FEED, D_4, D_C, D_M, and LLT. Next, their incremental implementations are benchmarked in Section III. This paper ends with a discussion in Section IV, reflecting on the work presented.

II. Incremental Distance Transforms (IDT)

With IDT, a new class of algorithms is introduced. This class is inspired by the algorithm of FEED [10], [13], as this enables a direct incremental implementation. Therefore, we will now first briefly describe an adapted version of the FEED algorithm, as this aids the understanding of the incremental implementations for EDT. Second, in Section II-B, the IDT for D_4 and D_C will be defined. Third, in Section II-C, the IDT for D_M and LLT will be introduced.

1051-4651/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPR.2010.67

A. IDT for FEED

The FEED algorithm calculates the EDT starting directly from the definition (see Equation 1), or rather its inverse. FEED on an image is initialized with distance DI($\cdot$) for (background) pixels b:

$$\text{DI}(b) = \text{ if } (b \in O) \text{ then } 0 \text{ else } \infty. \tag{2}$$

Subsequently, the adapted naive FEED algorithm is:

$$\begin{array}{l} \text{foreach } o \in O \\ \quad \text{determine: } A_o \\ \quad\quad \text{update: foreach } a \in A_o \text{ do} \\ \quad\quad\quad \text{DI}(a) \;=\; \min\{\text{DI}(a), \text{ED}^2(o,a)\}, \end{array} \tag{3}$$

where A_o is the area where o should feeds distances to. ED^2 instead of ED is used as this avoids square roots. In fact, all calculations in our FEED implementation are done in integers.

With $o \in O$ in Equation 3, only the border pixels of O have to be considered because

$$\min\{\text{ED}(b,o)\} \;==\; \text{ED}(b,o_b), \tag{4}$$

where o_b is a border pixel of O; i.e., having at least one of its four 4-connected pixels in the background.

Having this implementation of FEED, its incremental implementation can be developed as follows:

$$\text{DI}_{s+m}(b) \;=\; \text{DI}_s(b) \;=\; \text{DT with } O_s, \tag{5}$$

where DT is in this case FEED and O_s is the set of stationary objects. This initializes the DI_{s+m}, the combined DI for the stationary (s) and moving (m) objects. Subsequently, FEED as defined in Equation 3, is applied again:

$$\begin{array}{l} \text{foreach } o \in O_m \\ \quad \text{determine: } A_o \\ \quad\quad \text{update: foreach } a \in A_o \text{ do} \\ \quad\quad\quad \text{DI}_{s+m}(a) \;=\; \min\{\text{DI}_{s+m}(a), \text{ED}^2(o,a)\} \end{array} \tag{6}$$

where O_m is the set of moving objects.

As with the original FEED algorithm, also its incremental implementation (FEED*) can be speeded up. Most importantly, the area A_o can be restricted immediately to a circle with radius d_{max} (i.e., the outer bounding box), being the maximum ED in DI_s. This is, because combining O_m and O_s can only decrease the maximum ED, compared to the maximum ED in O_s only.

B. IDT for city-block / Chamfer distances

The IDT for the city-block (D_4) and Chamfer (D_C) DT are similar. Therefore, they are introduced together.

D_4 is initialized as follows:

$$\text{if I}[0][0] \;\in\; O \text{ then DI}[0][0] = 0, \text{ else DI}[0][0] = \infty, \tag{7}$$

where $\text{I}[y][x]$ is the input image and $\text{DI}[y][x]$ its DI. Next, a forward raster scan is conducted:

$$\begin{array}{l} \text{if I}[y][x] \;\in\; O \text{ then DI}[y][x] = 0 \text{ else} \\ \text{DI}[y][x] = \min\{1 + DI[y][x-1], 1 + DI[y-1][x]\} \end{array} \tag{8}$$

Subsequently, a backward raster scan is applied:

$$\begin{array}{l} \text{if DI}[y][x] > 0 \text{ then DI}[y][x] = \\ \min\{\text{DI}[y][x], 1 + \text{DI}[y][x+1], 1 + \text{DI}[y+1][x]\} \end{array} \tag{9}$$

For the IDT based on D_4, first the area is determined over which O_m can change the DI. For this the bounding box (bb) of O_m is determined and, subsequently, extended in all directions with d_{max}, the maximum distance in DI_s. Then, inside this bb, the maximum occurring distance is determined. Next, the bb of O_m is enlarged with that distance to provide the local area over which the modified D_4^* algorithm must be applied.

First, the incremental D_4^* algorithm is initialized, using Equation 5. Second, Equation 8 and 9 are applied on O_m, instead of O. The forward raster scan starts at the lowest $[y_s][x_s]$ point of the bb of O_m and for the next lines start with that x_s. It can be stopped as soon as a scanline $[y_e]$ with no changes in $\text{DI}[y][x]$ has been found. The backward scan is stopped as soon as a scanline below O_m with no changes in DI is detected.

The IDT for D_C is similar to the IDT for D_4. However, because also the two diagonal neighboring pixels are taken into account in the raster scans, a larger part of the local area has to be covered.

C. IDT for D_M and LLT

Both D_M and LLT produce intermediate images of which the values cannot be regarded as distances, in the sense that they cannot be compared with the finally produced Euclidean distances.

According to the definition of DI (see Equation 1), DI_s and DI_m can be calculated separately. DI_s is initialized through Equation 5, with DT being either D_M or LLT. Next, their full original algorithm is applied on the rectangular area over which the moving object can have an effect, as described in the previous section, which results in DI_m. Subsequently, using the $\min$ operator, both DI are combined to provide the final ED^2 DI.

III. Benchmark

In a benchmark, the five original algorithms and their five IDT implementations, as described in the previous two sections, will be compared with each other: D_4/D_4^* [1], D_C/D_C^* [7], the algorithm of Maurer et al. [8] (D_M/D_M^*), LLT/LLT* [9], and FEED/FEED* [10], [13].

All ten algorithms were implemented using single precision (32 bit) integers, producing their output in a format fastest for each method: ED^2 for FEED/FEED*, D_M/D_M^*, and LLT/LLT*; ED$\times$3 for D_C/D_C^*; and ED for D_4/D_4^*. The

set	standard implementation					incremental implementation				
	FEED	D_M	LLT	D_C	D_4	FEED*	D^*_M	LLT*	D^*_C	D^*_4
A	7.6	34.3	34.7	10.3	7.5	0.5	3.5	3.4	1.2	0.7
B	7.1	32.9	33.5	10.3	7.5	0.6	4.7	4.6	1.5	1.0
C	6.6	24.2	23.1	11.4	7.9	1.3	17.5	17.6	3.8	2.7
D	7.4	29.2	28.0	11.5	7.9	0.6	9.4	9.3	1.9	1.3
average	7.2	30.1	29.8	10.9	7.7	0.7	8.8	8.7	2.1	1.4

Table I

TIMING (IN NS/PIXEL) RESULTS OF THE BENCHMARK ON 4 TEST SEQUENCES AS WELL AS THEIR AVERAGE; SEE ALSO SECTION III. THE BENCHMARK WAS CONDUCTED ON BOTH THE STANDARD IMPLEMENTATION, AS REFERENCE VALUES, AND THE IDT IMPLEMENTATION (DENOTED WITH *) OF THE FOLLOWING ALGORITHMS: FEED, D_M, LLT, D_C, AND D_4; SEE ALSO SECTIONS I AND II.

times measured for the IDT methods include the time for reuse of DI_s for subsequent frames. This is achieved by restoring the part changed by the IDT back from a changed copy of DI_s.

The benchmark was executed on a PC with an Intel Core 2 Duo E6550P® 2.33GHz processor (2x 32KB data and 2 x 32 KB instruction L1 cache, 4096 KB L2 cache) and with 1024 MB memory, using the *gcc* compiler. The benchmark employed four sequences of 120 images, of size 652×492 pixels. Each of the four sequences contained a triangular object moving throughout the frame; see also Figure 1. The input image was split in two images. One for the program parts handling O_s, with 0 and 255 indicating respectively fixed object pixels and the background. The other for the program parts handling O_m, with 0, 127, and 255 indicating a pixel from respectively O_s, O_m, and the background.

Table I shows the results of the benchmark in terms of speed. It shows that all IDT implementations are significantly faster than their original implementation. For example, FEED* is at least $5\times$ faster than FEED. Further, Table I shows that FEED* is at least $7\times$ faster than both D^*_M [8] and LLT* [9]. FEED* is also $>2.4\times$ faster than D^*_C [7] and even $>1.4\times$ faster than D^*_4 [1], with FEED being the only algorithm to give exact results. Please note that the obtained times are dependent on the content of the images.

IV. DISCUSSION

On the one hand, distance transforms (DT) are a basic operation in computational geometry. On the other hand, they are applied within various applications (e.g., [11], [12], [13]), either by itself or as intermediate method; e.g., video surveillance, robot navigation, skeletonization, fMRI data analysis, neuromorphometry, and volume rendering. This article will extent current research by providing a new class of fast and exact algorithms for incremental distance transforms (IDT).

Par excellence, IDT can be used to generate distance images (DI) for video sequences, in a fast manner. It is assumed that each frame of a sequence consists of stationary objects and 1 moving object (see Figure 1), although an extension to multiple moving objects is straightforward. Then, the DI of the stationary objects (DI_s) is calculated once for each sequence, using a certain DT. This DI_s is used in the corresponding IDT to calculate the complete DI for each frame. Such an IDT is an adapted version of the corresponding DT.

The principle of IDT is applied on five DT: city-block DT (D_4) [1], Chamfer DT [7], a recent exact Euclidean DT [8], the LLT algorithm [9], and the FEED algorithm [10]. A benchmark has been employed to enable a controlled comparison among these DT; see also Table I. In general, the observed speedup is large, compared to calculating the DI for each frame separately with the corresponding DT. However, it should be noted that the timing results depend on both the used DT and the content of the image/video sequence.

The lowest speedup is obtained for the D_M and LLT exact ED transformations. From the DI_s and the position of moving object, a rectangular area has been determined in which the DT is applied as the IDT in order to obtain the DI of the moving objects (DI_m) in that area. Next, the two DIs have been combined, using the minimum operator. Note that this method is generic, as it can be used to convert any DT method into a corresponding IDT method.

For the raster scan methods D_C and D_4, the obtained speedup is larger. The IDT is applied to a rectangular area, determined in the same way as for D_M and LLT, and consists of a modified version of the corresponding DT. The starting DI_m is initialized with DI_s, which reduces the number of updates on it. Further, the raster scans can be restricted to only part of the rectangular area. Also, by placing the data for the rectangular area in the same memory locations as DI_s, the final output is directly available.

For FEED the obtained speedup is the largest. This is largely due to the fact that to obtain the final result, only the border pixels of the moving object have to feed their EDs in a restricted area. With FEED on full images already being faster than the other two exact EDs, FEED's IDT implementation showed to be $> 7\times$ faster than the IDTs based on both D_M and LLT. Moreover, FEED* was also faster than both D^*_C and D^*_4.

It would of interest to apply IDT on various other DT, both those that provide exact ED and those that approximate ED. Throughout the last decade, a range of interesting DT

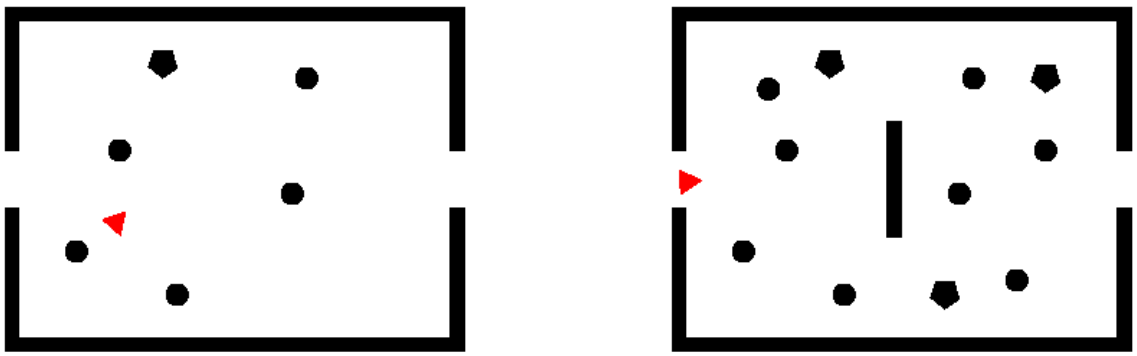

Figure 1. Sample test images from set A (left) and B (right). The triangle is the object O_m, which moves throughout the frame during the sequence. The other objects are all stationary ($o \in O_s$). Sets C and D contain the same images as sets A respectively B with the border objects left out.

have been presented either in separate papers (e.g., [3], [4]), in special issues of journals or dedicated workshops (e.g., [5], [6]), or as core of applications (e.g., [11], [12]). Consequently, a more exhaustive benchmark would be of interest. The current benchmark could be easily extended through, for example: i) Incorporating a larger and more varying set of image sequences than the current four sets; cf. Section III and Figure 1; ii) Including alternative DTs, such as just mentioned; and iii) Use image sequences with multiple moving objects.

As said earlier, one should note that the timings are also dependent on the content of the images. This effect needs further investigation, especially for FEED, as its founded on a stochastic process and, consequently, its computational complexity cannot be proven. Consequently, FEED has only shown to be fast through experimental results. Although this has been done repeatedly, as also in this article, in contrast with the other methods, it cannot be proved that the arithmetic complexity is $O(n)$.

Taken together, with the class of IDT algorithms, fast exact EDT can be done on video sequences. Hence, no approximations of EDT are needed due to its computational burden [2]. As such, an important new class of video processing algorithms is launched, which will find its way to many applications in both image and video analysis.

ACKNOWLEDGMENT

We gratefully acknowledge the reviewers, for their valuable comments on an earlier version of this article.

REFERENCES

[1] A. Rosenfeld and J. L. Pfaltz, "Sequential operations in digital picture processing," *Journal of the ACM*, vol. 13, no. 4, pp. 471–494, 1966.

[2] A. Hajdu and T. Tóth, "Approximating non-metrical Minkowski distances in 2D," *Pattern Recognition Letters*, vol. 29, no. 6, pp. 813–821, 2008.

[3] O. Cuisenaire, "Locally adaptable mathematical morphology using distance transformations," *Pattern Recognition*, vol. 39, no. 3, pp. 405–416, 2006.

[4] F. Porikli and T. Kocak, "Fast distance transform computation using dual scan line propagation," *Proceedings of SPIE (Real Time Imaging)*, vol. 6496, pp. 649 608–1–649 608–8, 2007.

[5] G. Borgefors, I. Nyström, and G. Sanniti di Baja, "Discrete geometry for computer imagery," *Pattern Recognition Letters*, vol. 23, no. 6, p. [special issue], 2002.

[6] ——, "Discrete geometry for computer imagery," *Discrete Applied Mathematics*, vol. 125, no. 1, p. [special issue], 2003.

[7] G. Borgefors, "Distance transformations in digital images," *Computer Vision, Graphics, and Image Processing: An International Journal*, vol. 34, no. 3, pp. 344–371, 1986.

[8] C. R. Maurer, Jr., R. Qi, and V. Raghavan, "A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 265–270, 2003.

[9] Y. Lucet, "New sequential exact Euclidean distance transform algorithms based on convex analysis," *Image and Vision Computing*, vol. 27, no. 1–2, pp. 37–44, 2009.

[10] T. E. Schouten and E. L. van den Broek, "Fast Exact Euclidean Distance (FEED) Transformation," in *Proceedings of the 17th IEEE International Conference on Pattern Recognition (ICPR 2004)*, J. Kittler, M. Petrou, and M. Nixon, Eds., vol. 3, Cambridge, United Kingdom, 2004, pp. 594–597.

[11] E. L. van den Broek, T. E. Schouten, and P. M. F. Kisters, "Modeling human color categorization," *Pattern Recognition Letters*, vol. 29, no. 8, pp. 1136–1144, 2008.

[12] M. Chen, W. Lu, Q. Chen, K. Ruchala, and G. Olivera, "Efficient gamma index calculation using fast Euclidean distance transform," *Physics in Medicine and Biology*, vol. 54, no. 7, pp. 2037–2047, 2009.

[13] T. E. Schouten, H. C. Kuppens, and E. L. van den Broek, "Video surveillance using distance maps," *Proceedings of SPIE (Real-Time Image Processing)*, vol. 6063, pp. 54–65, 2006.