LINEAR TIME ALGORITHM FOR EXACT DISTANCE TRANSFORM

K.C. Ciesielski, J.K. Udupa and G.J. Grevera

Medical Image Processing Group
Department of Radiology
University of Pennsylvania

TECHNICAL REPORT NO. MIPG337

November, 2006

Department of
RADIOLOGY
University of Pennsylvania

Linear time algorithm for exact distance transform

Krzysztof Chris Ciesielski,[a,b,*] Jayaram K. Udupa,[b,†] and George J. Grevera[c,b]

[a]Department of Mathematics, West Virginia University,

Morgantown, WV 26506-6310

[b]Department of Radiology, MIPG, University of Pennsylvania, Blockley Hall – 4th Floor,

423 Guardian Drive, Philadelphia, PA 19104-6021

[c]Mathematics and Computer Science Department, Saint Joseph's University,

5600 City Avenue, Philadelphia, PA 19131

Abstract

In 2003 Mauer *at al.* (see [1]) published a paper describing an algorithm that computes the exact distance transform in a linear time (with respect to image size) for the rectangular binary images in k-dimensional space, where distance is measured with respect to a metric from some class of metrics including Euclidean distance and, in general, L_p-metric distance for $1 \leq p \leq \infty$. However, that algorithm contains an error which was transformed to its Euclidean metric ITK implementation of Tustison *at al.* (see [3]). In this paper we describe a corrected version of the algorithm for $1 < p < \infty$ and *prove* that it indeed has the desired properties. We also discuss in details the error of the algorithm from the paper of Mauer *at al.* The revised version of the algorithm will be made available at ITK and as a part of CAVASS software system developed and maintained at MIPG.

*MIPG Report # 337. Partially supported by NSF grant DMS-0623906.

E-mail: KCies@math.wvu.edu; web page: `http://www.math.wvu.edu/~kcies`; partially supported by NSF grant DMS-0623906.

†Partially supported by DHHS grant NS 37172.

1 Preliminaries

Throughout the paper we fix the following notation: image dimension $k \geq 2$, image scene $C = \{1, \dots, n_1\} \times \cdots \times \{1, \dots, n_k\}$, where we assume that $n_i \geq 2$ for all $i \leq k$, and image scene's size $N = n_1 \times \cdots \times n_k$. A binary image I is a function from C into $\{0, 1\}$, where set $F_I = \{c \in C \colon I(c) = 1\}$ is the image foreground and $B = B_I = \{c \in C \colon I(c) = 0\}$ is its background.

Let Δ be a distance (metric) in the k-dimensional Euclidean space $\mathbb{R}^k$. A *distance transform* DT for image I (with respect to Δ) is a mapping $DT \colon C \to [0, \infty]$ defined as $DT(x) = \Delta(x, B) \stackrel{\text{def}}{=} \inf\{\Delta(x, b) \colon b \in B\}$, where infimum over the empty set $\emptyset$ is defined as ∞. A *closest feature transform* FT is any function $FT \colon C \to B \cup \{\emptyset\} \subset C \cup \{\emptyset\}$ with the property that $FT(c) = \emptyset$ whenever $DT(c) = \infty$, while otherwise $FT(c) \in B$ is such that $\Delta(c, FT(c)) = \Delta(c, B) = DT(c)$. Thus, FT determines DT, and DT can be recovered from FD in $O(N)$ time: for every $c \in C$ calculate $\Delta(c, FT(c))$. (We assume that calculation of $\Delta(c, d)$ is of $O(1)$ time.).

For $x \in C \subset \mathbb{R}^k$ symbol x_i will stand for its i-th coordinate, that is, $x = (x_1, \dots, x_k)$. Also, for an index $1 \leq d \leq k$ and a point $x \in \mathbb{R}^k$ let $\mathbb{R}_d(x) = \{c \in \mathbb{R}^k \colon c_i = x_i \text{ for all } i \neq d\}$ be the one-dimensional row in $\mathbb{R}^k$ parallel to d axis containing x, and let $R_d(x) = C \cap \mathbb{R}_d(x)$ be its restriction to a scene C. The distinction between $R_d(x)$ with $\mathbb{R}_d(x)$ is an important issue, since a confusion of these two notions seems to be at the basis of the error in [1].

The algorithm LTDT (linear time distant transform) that we will construct works properly only for the distance measures Δ that satisfies the following two conditions.

(♣) For every $d = 1, \dots, k$ and $x, u, v \in \mathbb{R}^k$ the following holds. If u is to the left of v with respect to d-th coordinate (i.e., $u_d \leq v_d$), and $\bar{x} \in R_d(x)$ is obtained from x by increasing its d-th coordinate (i.e., $x_d < \bar{x}_d$ and $x_i = \bar{x}_i$ for all $i \neq d$), then $\Delta(x, u) \geq \Delta(x, v)$ implies $\Delta(\bar{x}, u) \geq \Delta(\bar{x}, v)$ and $\Delta(x, u) > \Delta(x, v)$ implies $\Delta(\bar{x}, u) > \Delta(\bar{x}, v)$.

(♠) For every $d = 1, \dots, k$ and $\bar{x}, u, v \in \mathbb{R}^k$ the following holds. If u is strictly left of v with respect to d-th coordinate (i.e., $u_d < v_d$), then there is an $x_{\overline{uv}} \in \mathbb{R}_d(\bar{x})$ such that for every $x \in \mathbb{R}_d(\bar{x})$: if $x_d < (x_{\overline{uv}})_d$, then $\Delta(x, u) < \Delta(x, v)$; and if $x_d > (x_{\overline{uv}})_d$, then $\Delta(x, u) > \Delta(x, v)$.

The conditions are closely related. In particular, (♠) implies (♣) for the case

when $u_d < v_d$.

Recall that, for $1 \le p < \infty$, the L_p metric on $\mathbb{R}^k$ is defined by a formula $\Delta(x,y) = \left(\sum_{i=1}^k |x_i - y_i|^p\right)^{1/p}$. In particular, L_2 metric is the standard Euclidean distance. The fact that the L_p metric satisfies both of these properties is well known. However, for completeness sake, we include its proof in Fact 6. Condition (♣) is a restatement of Property 4 from [1]. Property (♠) was also used in the algorithm from [1]. However, despite the fact that the algorithm from [1] was supposed to work for the L_1- and L_∞-metrics, property (♠) is false for these distances. (For $\mathbb{R}^2$ and row $\mathbb{R}_1(0,0)$, points $u = (0,0)$ and $v = (1,1)$ contradict (†) for L_1 metric, and points $u = (0,1)$ and $v = (1,1)$ contradict it for L_∞ metric. In both these cases there is a non-trivial interval of points in $\mathbb{R}_1(0,0)$ equidistant from u and v.)

In what follows we will show that for the distances satisfying (♣) and (♠) our algorithm LTDT returns the exact distant transform. Moreover, it does it in $O(N)$ time, as long as finding $\Delta(c,d)$ and $x_{\overline{uv}}$ can be done in $O(1)$ time. Thus, all of this is true for the L_p distances for $1 < p < \infty$, including the Euclidean distance, since all the assumptions are satisfied by these L_p metrics.

2 The algorithm outline: dimension step-up

In this section we will construct the LTDT algorithm using a subroutine, called DimUp, which will be described in the next section. We will also prove (assuming the right properties of DimUp) that LTDT indeed returns the distant transform and that it runs in time $O(N)$.

For $0 \le d \le k$ and $x \in C$ let $H_d(x) = \{c \in C\colon c_i = x_i \text{ for all } d < i \le k\}$ be the d-dimensional hyperplane containing x and fixing the terminal $k-d$ coordinates, that is, the coordinates with indices greater than d. We say that a function $F\colon C \to B \cup \{\emptyset\}$ is a *d-dimensional approximation of FT at $x \in C$* provided $F(x)$ a correct of value of a closest feature transform for $I \restriction H_d(x)$, the image I restricted to $H_d(x)$, that is, $F(x) = \emptyset$ when $B \cap H_d(x) = \emptyset$; otherwise $F(x) \in B \cap H_d(x)$ and $\Delta(x, F(x)) = \Delta(x, B \cap H_d(x))$. Such an F is a *d-dimensional approximation of FT* provided it is a d-dimensional approximation of FT at every $x \in C$, that is, when for every $x \in C$ its restriction $F \restriction H_d(x)$ to $H_d(x)$ is a true feature transform for $I \restriction H_d(x)$. Notice that k-dimensional approximation of FT (for a k-dimensional image)

is its true FT.

We assume that we have a procedure (subroutine) DimUp that has the following properties. (These are the same properties that the procedure VoronoiFV from [1] was supposed to have.)

DimUp input: Row $R_d(x)$ indicators: $x \in C$ and $1 \leq d \leq k$; a function $F\colon C \to B \cup \{\emptyset\}$ which is a $(d-1)$-dimensional approximation of FT at every $c \in R_d(x)$.

DimUp output: A modified $F\colon C \to B \cup \{\emptyset\}$ which is a d-dimensional approximation of FT at every $c \in R_d(x)$. The values of F at spels $c \notin R_d(x)$ remain unchanged.

DimUp running time cost: $O(n_d)$, where n_d is the size of the row $R_d(x)$.

For $1 \leq d \leq k$ let $C_d = \{x \in C\colon x_d = 0\}$ be the hyperplane passing through $(0, \ldots, 0)$ and perpendicular to $R_d(x)$. Note that C_d has size N/n_d.

Algorithm LTDT

Input: Dimension $k \geq 2$ of the image; a sequence $n_1, \ldots, n_k$ representing the size of the scene $C = \{1, \ldots, n_1\} \times \cdots \times \{1 \ldots, n_k\}$; a binary image $I\colon C \to \{0, 1\}$;

Output: A distance transform $DT\colon C \to [0, \infty]$ for the image I.

Auxiliary Data Structures: A feature transform function $F\colon C \to C \cup \{\emptyset\}$. A queue Q of spels from C. Dimension counter d.

begin

1. *for* all $x \in C$ *do*
2. *if* $I(x) = 0$ *then* $F(x) = 0$ *else* $F(x) = \emptyset$;
3. *endfor*;
4. *for* $d = 1$ *to* k *do*
5. push all spels from C_d to Q;
6. *while* Q is not empty *do*
7. remove a spel x from Q;
8. invoke DimUp with x, d, and current F;
9. *endwhile*;
10. *endfor*;
11. *for* all $x \in C$ *do*
12. *if* $F(x) = \emptyset$ *then* $DT(x) = \infty$ *else* $DT(x) = \Delta(x, F(x))$;
13. *endfor*;

end

Lines 1-10 of this algorithm represent procedure ComputeFT from [1]. Our main contribution here is the proof of the following lemma.

Lemma 1 *If algorithm DimUp works correctly, then for every binary k-dimensional image I on $C = \{1, \dots, n_1\} \times \cdots \times \{1 \dots, n_k\}$ algorithm LTDT returns the exact distance transform F for the image I. It does it in time $O(N)$, where N is the size of C.*

Proof. After execution of lines 1-3 the map F represents the 0-dimensional approximation of FT for I, as $H_0(x) = \{x\}$. It part runs in $O(N)$ time.

Next notice that for every $d = 1, \dots, k$, when program enters lines 5-9, then F is a $(d-1)$-dimensional approximation of FT for I; when it exits lines 5-9, then F is a d-dimensional approximation of FT for I.

This statement is proved by mathematical induction on d. For $d = 1$ the entry requirement is guaranteed by lines 1-3. For $d > 1$ this is ensured by the inductive assumption. To finish the argument it is enough to show that the execution of lines 5-9 transforms $(d-1)$-dimensional approximation F of FT for I to the d-dimensional approximation of FT. This is guaranteed by the assumptions on DimUp: when executing lines 5-9, each row $R_d(x)$ of C is considered precisely once, and running DimUp for this row changes the values of F on this (and only this) row from $(d-1)$-dimensional approximation of FT to d-dimensional approximation of FT.

Next note that for each d the while loop from lines 6–9 is executed precisely N/n_d many times (the size of C_d) and each time the execution cost of DimUp is of order $O(n_d)$. Thus, each execution of lines 5-9 runs in time of order $N/n_d O(n_d) = O(N)$. This, the total time of running lines 1-10 is of order $O(N) + kO(N) = O(N)$.

Finally, note that after the execution of the loop 4-10 F represents k-dimensional approximation of FT for I, which is the true FT for I.

The execution of the loop 11-13 is still of order $O(N)$ (we assume that calculation of $\Delta(x, y)$ is $O(1)$) and the resulted DT is indeed an exact distance transform for I. ∎

Remark 2 Note that if metric Δ is the L_p distance where p is a natural number (in particular, when Δ is the Euclidean distance), and when in line 12 we replace $DT(x) = \Delta(x, F(x))$ with its pth power: $DT(x) = \Delta(x, F(x))^p = \sum_{i=1}^{k} |x_i - y_i|^p$, then LTDT algorithm can be run with only integer arithmetic, as long as this can be done for DimUp subroutine.

3 DimUp procedure: further reduction

The main theoretical feature responsible for the correctness of the algorithm is the following fact.

Lemma 3 *Let I be a binary image on $C = \{1, \dots, n_1\} \times \cdots \times \{1 \dots, n_k\}$, $R = R_d(z)$ be a row in I, and $F\colon C \to B \cup \{\emptyset\}$ be a $(d-1)$-dimensional approximation of FT at every $x \in R$. If metric Δ has property (♣) and $G = \{F(x) \in B\colon x \in R\}$, then $G \subset B \cap H_d(z)$ and $\Delta(x, G) = \Delta(x, B \cap H_d(z))$ for every $x \in R$. In particular, for every $x \in R$ the value of a d-dimensional approximation of FT at x can be chosen from $G \cup \{\emptyset\}$.*

Proof. To see that $G \subset H_d(z)$ pick a $y \in G$ and let $x \in R$ be such that $y = F(x) \in H_{d-1}(x)$. Then $y_i = x_i$ for all $i \geq d$. Since $x \in R = R_d(z)$ implies that $x_j = z_j$ for every $j \neq d$, we have $y_\ell = z_\ell$ for all $\ell > d$. So, $y \in H_d(z)$.

Inclusion $G \subset B \cap H_d(z)$ clearly implies $\Delta(x, G) \geq \Delta(x, B \cap H_d(z))$. To show the other inclusion, choose an arbitrary $u \in B \cap H_d(z)$. We need to find a $v \in G$ such that $\Delta(x, v) \leq \Delta(x, u)$. Let $\bar{x} \in R$ be such that $\bar{x}_d = u_d$. Combining this with $u \in H_d(z) = H_d(\bar{x})$ we conclude that $u \in H_{d-1}(\bar{x})$. In particular, $B \cap H_{d-1}(\bar{x}) \ni u$ is non-empty, so $v = F(\bar{x})$ belongs to $B \cap H_{d-1}(\bar{x})$ and G and has a property $\Delta(\bar{x}, v) = \Delta(\bar{x}, B \cap H_{d-1}(\bar{x})) \leq \Delta(\bar{x}, u)$, since F is a $(d-1)$-dimensional approximation of FT at $\bar{x} \in R$. So, $\Delta(\bar{x}, u) \geq \Delta(\bar{x}, v)$. Also, $u, v \in H_{d-1}(\bar{x})$ implies that $u_d = \bar{x}_d = v_d$. We will consider three cases.

If $\bar{x}_d < x_d$, then, by (♣) (with roles of x and $\bar{x}$ exchanged), we have $\Delta(x, u) \geq \Delta(x, v)$, as $u_d \leq v_d$.

If $\bar{x}_d = x_d$, then $\bar{x} = x$ and clearly inequality $\Delta(\bar{x}, u) \geq \Delta(\bar{x}, v)$ implies $\Delta(x, u) \geq \Delta(x, v)$.

Finally, if $x_d < \bar{x}_d$, then inequality $\Delta(x, v) > \Delta(x, u)$ is impossible, since then, by (♣) and $v_d \leq u_d$, we would conclude $\Delta(\bar{x}, v) > \Delta(\bar{x}, u)$, contradicting $\Delta(\bar{x}, u) \geq \Delta(\bar{x}, v)$. Thus, in this case we also have $\Delta(x, u) \geq \Delta(x, v)$. ■

Lemma 3 tell us that if we like to upgrade F from being $(d-1)$-dimensional approximation of FT on $R = R_d(z)$ to being a d-dimensional approximation of FT on R, the values of this new F can be chosen from the values of old F on R, that is, from $F[R] = \{F(x)\colon x \in R\}$. In our upgrade procedure we will need first to further restrict our choice of the values of new F on R to a subset of $G \cup \{\emptyset\}$. This will be done with the TRIM procedure having the following properties.

TRIM input: Row $R = R_d(c)$ indicators: $c \in C$ and $1 \le d \le k$; a function $F\colon C \to B \cup \{\emptyset\}$ which is a $(d-1)$-dimensional approximation of FT at every $x \in R_d(c)$.

TRIM output: A list $(q_1, \dots, q_m)$, $0 \le m \le n_d$, of spels from the set $G = \{F(x) \in B\colon x \in R\}$ such that

(i) $(q_j)_d < (q_{j+1})_d$ for every $1 \le j < m$;

(ii) $\Delta(x, \{q_j\colon 1 \le j \le m\}) = \Delta(x, G)$ for every $x \in R$;

(iii) for every $1 \le j \le m$ there is an $x \in R$ with $\Delta(x, q_j) = \Delta(x, G)$.

TRIM running time cost: $O(n_d)$, where n_d is the size of the row $R_d(c)$.

Using TRIM it is easy to describe the DimUp algorithm. This is actually a part (lines 15-24) of the VoronoiFT procedure from [1].

Algorithm DimUp
Input: A $(d-1)$-dimensional approximation F of FT on $R_d(c)$.
Output: A d-dimensional approximation F of FT on $R_d(c)$.
Auxiliary Data Structures: A queue Q of spels from C. A counter ℓ.
begin

```
1.  invoke TRIM for R_d(c) and F to get list (q_1, ..., q_m);
2.  if m > 0 then
3.      push all spels from R_d(c) to Q in the increasing order
                                  (i.e., with x_d = 1 for the first removed spel);
4.      initialize ℓ = 1;
5.      while Q is not empty do
6.          remove a spel x from Q;
7.          while ℓ < m and Δ(x, q_ℓ) ≥ Δ(x, q_{ℓ+1}) do
8.              ℓ = ℓ + 1;
9.          endwhile;
10.         F(x) = q_ℓ;
11.     endwhile;
12. endif;
```

end

Lemma 4 *Assume that Δ satisfies (♣). If algorithm TRIM works correctly and the input function F for DimUp is a $(d-1)$-dimensional approximation F of FT on $R_d(c)$, then the output version of F for DimUp is a d-dimensional approximation of FT on $R_d(c)$. Moreover, DimUp is running in $O(n_d)$ time.*

PROOF. By our assumptions on TRIM, the execution time of line 1 is of order $O(n_d)$. The total number of times the lines 7-9 can be executed during the entire program run is bounded by $m \leq n_d$. Since Q has a size n_d, this means that lines 5-11 are executed with $O(n_p)$ operations. So, DimUp requires only $O(n_d)$ operations.

Now, $m = 0$ precisely when $F(x) = \emptyset$ for all $x \in R_d(c)$, in which case $B \cap H_d(c) = \emptyset$, and program correctly leaves all these values unchanged.

So, assume that $m > 0$, that is, that the set $H = \{q_1, \ldots, q_m\}$ is non-empty. We will enter loop from lines 5-11 precisely n_d times and on ith entering we will have $x_d = i$ for the removed x from the queue Q. We will show, by induction on i, that upon leaving the loop we will have

$$\Delta(x, q_j) \geq \Delta(x, q_\ell) \;\&\; \Delta(x, q_\ell) < \Delta(x, n) \;\text{ for every } 1 \leq j < \ell < n \leq m. \quad (1)$$

In particular, this shows that $\Delta(x, F(x)) = \Delta(x, q_\ell) \leq \Delta(x, \{q_1, \ldots, q_m\})$, which, by Lemma 3 and property (ii), is equal to $\Delta(x, B \cap H_d(x))$. Thus, F becomes a d-dimensional approximation of FT at x, and remains so, since the value of F at x does not change any more.

To prove (1) let ℓ_0 be the value of ℓ upon entering the loop and ℓ_1 its value upon leaving it. First we will argue for the first inequality. Let $1 \leq j < \ell_1$. If $j < \ell_0$, then $i > 1$, since for $i = 1$ we have $\ell_0 = 1$. Let $\bar{x} \in R_d(c)$ be such that $\bar{x}_d = i - 1$. So, by the inductive assumption, $\Delta(\bar{x}, q_j) \geq \Delta(\bar{x}, q_{\ell_0})$. Since $(q_j)_d < (q_{\ell_0})_d$ and $\bar{x}_d < x_d$, condition (♣) implies that $\Delta(x, q_j) \geq \Delta(x, q_{\ell_0})$. Moreover, execution of the loop from lines 7-9 insure that $\Delta(x, q_{\ell_0}) \geq \Delta(x, q_t) \geq \Delta(x, q_{\ell_1})$ for every $\ell_0 \leq t \leq \ell_1$. This implies that $\Delta(x, q_j) \geq \Delta(x, q_{\ell_1})$ for every $1 \leq j < \ell_1$.

To show the second inequality take $\ell_1 < n \leq m$. Then $\ell_1 + 1 \leq n \leq m$ and the fact that loop 7-9 stopped means that $\Delta(x, q_{\ell_1}) < \Delta(x, q_{\ell_1+1})$. By way of contradiction assume that $\Delta(x, q_{\ell_1}) \geq \Delta(x, q_n)$. We will show that this implies that q_{ℓ_1+1} fails to satisfy condition (ii) for the output of TRIM. So, choose an $\bar{x} \in R$. It is enough to prove that $\Delta(\bar{x}, q_{\ell_1+1}) > \Delta(\bar{x}, B \cap H_d(x))$. Let $\bar{x}_d = j$. We will consider three cases.

If $i < j$, then $x_d < \bar{x}_d$. Since $\Delta(x, q_{\ell_1+1}) > \Delta(x, q_n)$ and $(q_{\ell_1+1})_d \leq (q_n)_d$, condition (♣) implies $\Delta(\bar{x}, q_{\ell_1+1}) > \Delta(\bar{x}, q_n) \geq \Delta(\bar{x}, B \cap H_d(x))$.

If $i = j$, then $x = \bar{x}$ and $\Delta(\bar{x}, q_{\ell_1+1}) = \Delta(x, q_{\ell_1+1}) > \Delta(x, q_{\ell_1}) = \Delta(\bar{x}, q_{\ell_1}) \geq \Delta(\bar{x}, B \cap H_d(x))$.

Finally, if $j < i$, then $\bar{x}_d < x_d$. Notice that $\Delta(\bar{x}, q_{\ell_1}) < \Delta(\bar{x}, q_{\ell_1+1})$, since otherwise, by (♣), $\Delta(\bar{x}, q_{\ell_1}) \geq \Delta(\bar{x}, q_{\ell_1+1})$ and $(q_{\ell_1})_d \leq (q_{\ell_1+1})_d$ imply

$\Delta(x, q_{\ell_1}) \geq \Delta(x, q_{\ell_1+1})$, contradicting $\Delta(x, q_{\ell_1}) < \Delta(x, q_{\ell_1+1})$. Thus, once again, $\Delta(\bar{x}, q_{\ell_1+1}) > \Delta(\bar{x}, q_{\ell_1}) \geq \Delta(\bar{x}, B \cap H_d(x))$. ■

4 The TRIM procedure

In what follows we will use a simple relation CHECK(u,v,w), which depends on a row $R = R_d(c)$. It is applied to $u, v, w \in C$ with $u_d < v_d < w_d$ and is true when there is no integer n for which $(x_{\overline{uv}})_d \leq n \leq (x_{\overline{vw}})_d$, or, equivalently, when $\lceil (x_{\overline{uv}})_d \rceil > \lfloor (x_{\overline{vw}})_d \rfloor$, where $\lceil r \rceil$ is the smallest integer greater than or equal to r, and $\lfloor r \rfloor$ is the greatest integer less than or equal to r. Notice that if CHECK(u,v,w) is true, then $\Delta(x, v) > \Delta(x, \{u, w\})$ for every $x \in R_d(c)$. In other words, if $u, v, w \in G$, then v cannot be in the sequence returned by TRIM.

Algorithm TRIM
Input: Row $R = R_d(c)$ and a function $F\colon C \to B \cup \{\emptyset\}$ which is a $(d-1)$-dimensional approximation of FT at every $x \in R_d(c)$.
Output: A list $(q_1, \ldots, q_m)$ of spels from $\{F(x) \in B\colon x \in R\}$ satisfying (i)-(iii).
Auxiliary Data Structures: Counters i, m and spel pointers u, v.
begin

```
1.  set m = 0;
2.  for i = 1 to n_d do
3.      if F(x_i) ≠ ∅ then
4.          set m = m + 1;
5.          set q_m = F(x_i);
6.          if m > 1 then
7.              set u = q_{m-1};
8.              set v = q_m;
9.              if (x_{uv})_d ≥ n_d then
10.                 set m = m − 1;
11.             else
12.                 while m > 2 and CHECK(q_{m-2}, q_{m-1}, q_m) do[1]
13.                     set q_{m-1} = q_m;
14.                     set m = m − 1;
15.                 endwhile;
```

[1]In the implementation insure that CHECK doesn't freeze the program when $m = 2$.

```
16.              if m = 2 then
17.                 set u = q_1;
18.                 set v = q_2;
19.                 if (x_{\overline{uv}})_d < 0 then
20.                    set q_1 = q_2;
21.                    set m = 1;
22.                 endif;
23.              endif;
24.           endif;
25.        endif;
26.     endif;
27.  endfor;
28.  return sequence (q_1 ..., q_m) for the current value of m;
end
```

Lemma 5 *Assume that Δ satisfies (♣). Then TRIM works correctly and DimUp is running in $O(n_d)$ time.*

Proof. To see that TRIM runs in $O(n_d)$ time note that it enters the loop from lines 2-27 precisely n_d times. The i-th run time of this loop is of order $O(1)+2P_i$, where P_i is the number of runs of the loop from lines 12-15. Since each time this loop is run, one value from the set $\{F(x_i)\colon i = 1, \ldots, n_d\}$ is removed, we have $P_1 + \cdots + P_{n_d} \le n_d$. Therefore, TRIM indeed runs in time $\sum_{i=1}^{n_d}(O(1) + 2P_i) = O(n_d)$.

To prove that the output of TRIM satisfies (i)-(iii), we will show, by induction on $i = 1, \ldots, n_d$, that after completing i-th run a loop from lines 2-27 the following holds, where m_i stands for the value of m at this point program execution and $G_i = \{F(x_j) \in B\colon 1 \le j \le i\}$.

(A_i) $(q_j)_d < (q_{j+1})_d$ for every $1 \le j < m_i$ and all these q_j's belong to G_i;

(B_i) $\Delta(x, \{q_j\colon 1 \le j \le m_i\}) = \Delta(x, G_i)$ for every $x \in R$.

(C_i) for every $1 \le j \le m_i$ there is an $x \in R$ such that $\Delta(x, q_j) = \Delta(x, G_i)$.

This will finish the proof, since then TRIM's output value of m is equal to m_{n_d}, the set G_{n_d} equals to G from TRIM's output description, and so, the conditions (A_{n_d})-(C_{n_d}) are the restatement of (i)-(iii).

Assume that $m_0 = 0$. Then G_0 and the q-sequence are empty, so conditions (A_0)-(C_0) are satisfied. Thus, we just need to show that, for every

$i = 1, \dots, n_d$, if conditions (A_{i-1})-(C_{i-1}) are satisfied upon entering the code lines 2-27, then (A_i)-(C_i) hold upon finishing their execution.

Note that after each execution of lines 2-27 the sequence the q-sequence may have more than m_i elements. However, only the first m_i of its elements are of consequence, and these first m_i elements constitute the q-sequence (possibly empty) satisfying (A_i)-(C_i).

If $F(x_i) = \emptyset$, then $G_i = G_{i-1}$ and none of lines 4-25 is executed, so $m_i = m_{i-1}$ and the q-sequence remains unchanged. This clearly implies (A_i)-(C_i). So, for the rest of the proof assume that $F(x_i) \neq \emptyset$.

The execution of lines 3-4 temporarily extends the q-sequence (by assigning to $m_i = m$ value $m_{i-1} + 1$) and puts $F(x_i)$ at its end. This initial assignment ensures (A_i) and (B_i). However, (C_i) may be false at this stage, and the sequence may need to be trimmed to ensures it. This is done in lines 6-25.

Clearly, by the inductive assumption (A_{i-1}), at this stage the sequence satisfies condition (A_i), since $(F(x_i))_d = i > x_d$ for every $x \in G_{i-1}$. To see that the execution of lines 6-25 preserves (A_i), it is enough to note that the only changes to this sequence in lines 6-25 are either through dropping the last sequence element (in line 10) or by replacing the second to the last of its elements by the last one and shortening the sequence by 1 (lines 13-14 or 20-21). These operations clearly preserve (A_i).

Now, if we enter line 6 with $m = m_i = 1$, then $m_{i-1} = 0$ and, by (B_{i-1}), we have $G_{i-1} = \emptyset$. Although at this case condition in line 6 insures that no other lines are executed, this implies that $m_i = 1$ and $G_i = \{F(x_i)\} = \{q_1\}$, so (B_i) and (C_i) hold. So, assume that at line 6 we have $m_i = m > 1$, that is, that the q-sequence has at least two elements. Next we will decide whether its last element is in the proper position and, if not, modify the sequence.

Thus, entering line 7 we know that our q-sequence has at least two elements. In lines 7-11 we check whether there is any reason to keep q_m in the sequence. If not, we can simply remove it. More precisely, since $G_i = G_{i-1} \cup \{F(x_i)\}$, condition (B_{i-1}) implies that for every $x \in \mathbb{R}$ we have $\Delta(x, G_i) = \Delta(x, G_{i-1} \cup \{q_m\}) = \Delta(x, \{q_j \colon 1 \leq j \leq m\})$. Assume that the condition from line 9 is satisfied. Then, the only executed line in the rest of the loop is line 10, which discards the last element of the sequence. This means that $m_i = m = m_{i-1}$. Now, to show that this sequence satisfies (B_i) and (C_i), note that $x_{\overline{uv}}$ is to the right of every $x \in R$. This means that for $m = m_{i-1} + 1$ we have $\Delta(x, q_{m-1}) = \Delta(x, u) \leq \Delta(x, v) = \Delta(x, q_m)$. In particular, $\Delta(x, G_i) = \Delta(x, \{q_j \colon 1 \leq j \leq m\}) = \Delta(x, \{q_j \colon 1 \leq j \leq m_{i-1}\})$

is equal to $\Delta(x, G_{i-1})$ for every $x \in R$. Therefore, in this case (B_{i-1}) and (C_{i-1}) imply (B_i) and (C_i) for $m_i = m_{i-1}$.

Note that the condition from line 9 can be viewed as checking whether $\hat{q} = F(x_i)$ can satisfy (C_i). Next, we assume that the condition from line 9 fails. This means that we are entering execution of the lines 12-23 and that at this stage we have $\Delta(q_{m_i}, x_{n_d}) < \Delta(q_{m_{i-1}}, x_{n_d})$. We need to see that

$$\Delta(q_{m_i}, x_{n_d}) < \Delta(q_j, x_{n_d}) \text{ for every } 1 \leq j \leq m_{i-1}. \tag{2}$$

So, take a $j < m_{i-1}$. By (C_{i-1}), there exists an $x \in R$ such that $\Delta(x, q_{m_{i-1}}) = \Delta(x, G_{i-1})$. In particular, $\Delta(x, q_j) \geq \Delta(x, q_{m_{i-1}})$. So, by (♣), $\Delta(x_{n_d}, q_j) \geq \Delta(x_{n_d}, q_{m_{i-1}}) > \Delta(q_{m_i}, x_{n_d})$, proving (2).

Note that (2) shows that no matter what trimming to the q-sequence will be done, $\hat{q} = F(x_i)$ will satisfy (C_i).

Now, we consider execution of the lines 12-23. In their execution we will discard from the sequence $q_1, \ldots, q_{m_{i-1}}$ some of its terminal part. More precisely, we will search for an index $\bar{m} \in \{1, \ldots, m_{i-1} + 1\}$, redefine $q_{\bar{m}}$ as $\hat{q} = F(x_i)$, and define m_i as $\bar{m}$. Our resulting exit q-sequence will be of the form $q_1, \ldots, q_{\bar{m}-1}, q_{\bar{m}} = \hat{q}$, where the initial part, possible empty, consists of spels listed at in the q-sequence from the $i-1$ step. Note that, by (2), we need to insure condition (C_i) only for $j < \bar{m}$. We need also to preserve condition (B_i).

First we concentrate on the loop 12-15. Initially $m = m_{i-1} + 1$. Assume that the condition from line 12 is satisfied. Then the execution of lines 13-14 removes q_{m-1} from the sequence. Note, that under the condition from the line 12, this preserve (B_i). Indeed, to see this fact, we need to argue only that $\Delta(x, \{q_1, \ldots, q_{m-2}, q_{m-1}, q_m\}) = \Delta(x, \{q_1, \ldots, q_{m-2}, q_m\})$ for every $x \in R$, that is, that $\Delta(x, q_{m-1}) \geq \Delta(x, \{q_1, \ldots, q_{m-2}, q_m\})$. Let $u = q_{m-2}$, $v = q_{m-1}$, and $w = q_m$. Since, by the condition from the line 12, no x from R is between $x_{\overline{uv}}$ and $x_{\overline{vw}}$, one of the following two cases apply. (1) $x_d \leq (x_{\overline{uv}})_d$, in which case $\Delta(x, q_{m-1}) = \Delta(x, v) \geq \Delta(x, u) = \Delta(x, q_{m-2}) \geq \Delta(x, \{q_1, \ldots, q_{m-2}, q_m\})$. (2) $x_d \geq (x_{\overline{vw}})_d$ and $\Delta(x, q_{m-1}) = \Delta(x, v) \geq \Delta(x, w) = \Delta(x, q_m) \geq \Delta(x, \{q_1, \ldots, q_{m-2}, q_m\})$. All of this means that upon completion of the loop from lines 12-15, condition (B_i) holds for the current sequence $q_1, \ldots, q_{\bar{m}-1}, q_{\bar{m}} = \hat{q}$.

Upon finishing the loop from lines 12-15, either $m = 2$ or else condition CHECK(q_{m-2}, q_{m-1}, q_m) is false. Assume first that $m > 2$. In this case the sequence is not modified any more in lines 16-23, so we need to prove that

it satisfies (B_i). We know it holds for $q_m = \hat{q}$. To see that it holds for q_{m-1} note that the negation of CHECK(q_{m-2}, q_{m-1}, q_m) implies that there exists an $x \in R$ such that $(x_{\overline{uv}})_d < x_d < (x_{\overline{vw}})_d$. We will show that this x satisfies (B_i) for q_{m-1}. The last two inequalities imply immediately that $\Delta(x, q_{m-1}) = \Delta(x, v) < \Delta(x, w) = \Delta(x, q_m)$ and $\Delta(x, q_{m-1}) = \Delta(x, v) < \Delta(x, u) = \Delta(x, q_{m-2})$. Let $1 \le j \le m-2$. To see that $\Delta(x, q_{m-1}) \le \Delta(x, q_j)$ it is enough to show that $\Delta(x, q_{m-2}) \le \Delta(x, q_j)$. Let $\bar{x} \in R$ satisfy (B_{i-1}) for q_{m-2}. Then $\Delta(\bar{x}, q_j) \ge \Delta(\bar{x}, q_{m-2})$. So, $\bar{x}_d \le (x_{\overline{uv}})_d < x_d$ and, by (♣), $\Delta(x, q_j) \ge \Delta(x, q_{m-2})$. Thus, q_{m-2} satisfies (B_i). To show that (B_i) holds for q_j, let $\hat{x} \in R$ satisfy (B_{i-1}) for q_j. Then, $\Delta(\hat{x}, q_j) \le \Delta(\hat{x}, q_\ell)$ for all $1 \le l \le m-1$. We need to show that also $\Delta(\hat{x}, q_j) \le \Delta(\hat{x}, q_m)$, which follows from $\Delta(\hat{x}, q_{m-1}) \le \Delta(\hat{x}, q_m)$, which, in turn, is equivalent to $\hat{x}_d \le (x_{\overline{vw}})_d$. By way of contradiction, assume that $(x_{\overline{vw}})_d < \hat{x}_d$. Then $\bar{x}_d \le (x_{\overline{uv}})_d < x_d < (x_{\overline{vw}})_d < \hat{x}_d$. Since, by the definition of $\bar{x}$, $\Delta(\bar{x}, q_j) \ge \Delta(\bar{x}, q_{m-2})$, condition (♣) implies that $\Delta(\hat{x}, q_j) \ge \Delta(\hat{x}, q_{m-2})$. Also, $(x_{\overline{uv}}) < \hat{x}_d$ implies that $\Delta(\hat{x}, q_{m-2}) = \Delta(\hat{x}, u) > \Delta(\hat{x}, v) = \Delta(\hat{x}, q_{m-1})$. Therefore, $\Delta(\hat{x}, q_j) > \Delta(\hat{x}, q_{m-1})$, what contradicts the choice of $\hat{x}$. This completes the proof of the case when after completing line 15 we have $m > 2$.

To finish the proof, assume that after completing line 15 we have $m = 2$. Thus, the q-sequence consists of just two elements, q_1 and $q_2 = \hat{q}$. We also know that conditions C_i and (2) are satisfied. The final outcome depends on the condition from line 19. If $(x_{\overline{uv}})_d < 0$, then $\Delta(x, \{q_1, \hat{q}\}) = \Delta(x, \{\hat{q}\})$ for every $x \in R$, and we can safely remove q_1 from the sequence preserving (B_i); in this case condition (C_i) trivially holds. If $(x_{\overline{uv}})_d \ge 0$, then the first element of R satisfies (C_i) for q_1, so no change is required, and the two element sequence is returned. ∎

5 Comments on CHECK and LTDT

Assume that Δ the L_p metric. Although the algorithm is set up under the assumption that $C = \{1, \dots, n_1\} \times \cdots \times \{1 \dots, n_k\}$, we can as well take $C = C_1 \times \cdots \times C_k$, where C_i is any finite (say of size n_i) subset of $\mathbb{R}$. Then, when $C_i = \{c_1^i, \dots, c_{n_i}^i\}$ is naturally identified with its index set $\{1, \dots, n_i\}$, and $C = C_1 \times \cdots \times C_k$ with $\tilde{C} = \{1, \dots, n_1\} \times \cdots \times \{1 \dots, n_k\}$. In this case the naturally modified L_p distance $\tilde{\Delta}$ on C is given by a formula

$$\tilde{\Delta}((p_1, \dots, p_k), (q_1, \dots, q_k)) = \Delta((c_{p_1}^1, \dots, c_{p_k}^k), (c_{q_1}^1, \dots, c_{q_k}^k)).$$

This approach is of particular interest, when the image is anisotropic, though uniform with respect to each coordinate. In this case there are weights $w_i \in (0,\infty)$ such that $\tilde{\Delta}((p_1,\dots,p_k),(q_1,\dots,q_k)) = \sqrt[k]{\sum_{i=1}^{k}(w_i|p_i-q_i|)|^p}$. Our algorithm works perfectly in this setting, especially in case of the Euclidean distance. (See comments below.) However, in the rest of the discussion we assume, for simplicity, that $w_i = 1$ for all i's.

Some difficulty with using CHECK relation is in finding points $x_{\overline{vw}}$ for the given line $R = R_d(z)$. Its dth coordinate constitutes the solution of the equation $|x_d - u_d|^p + \sum_{i\neq d}|z_i - u_i|^p = |x_d - v_d|^p + \sum_{i\neq d}|z_i - v_i|^p$. For $p = 2$, the case of Euclidean distance, this equation reduces to a linear equation $(v_d - u_d)(2x - u - v) = \sum_{i\neq d}|z_i - v_i|^p - \sum_{i\neq d}|z_i - u_i|^p$, and so the values of $\lceil x_d \rceil$ and $\lfloor x_d \rfloor$, all that we are looking for, can be found quickly (certainly in $O(1)$ time) and within the integer arithmetic realm. For general p, however (even integer $p > 4$, in which case the equation is a polynomial of degree greater than 4) the problem is slightly more complicated. (Since we are interested in the solutions between $1,\dots,n_d$, a brute-force algorithm can be run in $\log_2 n_d$ time.)

In practical imaging applications the distance transform is often defined as a distance of a foreground point $x \in F$ to the boundary Bd between foreground and the background B (i.e., as $\Delta(x, Bd)$) rather than as a distance between x and B (i.e., as $\Delta(x, B)$). In the non-digital setting, these are the same notions. However, in the digital case you often identify each spel with the unit side k-dimensional cube centered at x, and the boundary as a union of all such cube's faces shared by foreground and background spels. In such interpretation it can be shown that $\Delta(x, Bd) = \Delta(x, Bd_0)$, where Bd_0 is the set of all points that eiher are at the center of a face used in Bd, or one of its corners. (Thus, each face in B_d is replaced by $1 + 2^{k-1}$ points.) Then, one can use the LTDT algorithm to compute $\Delta(x, Bd_0)$ for every $x \in F$. However, the center points belong only to double density grid (with respect to the original C), so before running LTDT to find $\Delta(x, Bd_0)$, ione should double each side of C, increasing the total size of C by a magnitude of 2^k.

6 Error in the algorithm of Mauer *at al.*

DimUp subroutine (which is equivalent to a part of the procedure VoronoiFV from [1]) works properly *only* when its input q-sequence have properties (i)-(iii). For example, algorithm DimUp *always* assigns q_1 to the first element

of the row $R = R_d(z)$. If (iii) is not satisfied for q_1, this assignment *must be incorrect*. This is precisely the error that appears in the algorithm from [1]. More specifically, the version of our CHECK algorithm appears also as a part of the procedure VoronoiFV from [1]. However, the authors insure in it only that between any appropriate u, v, and w, there is a point in $\mathbb{R}_d(z)$ (rather than in $R_d(z)$) between $x_{\overline{uv}}$ and $x_{\overline{vw}}$. (They just check the inequality $(x_{\overline{uv}})_d < (x_{\overline{vw}})_d$. There is no checking for the first element of the sequence or its last element.)

The following table contains a comparison of Maurer and Danielsson distance transforms as independently implemented in ITK. The binary input image size is $100 \times 100 \times 100$ pixels with a pixel spacing specified as $1.0 \times 1.0 \times 1.0$ units. This 3D data set consists of 10,000 random points sampled from a normal distribution to simulate a spherical object with a nonuniform (and possibly disconnected) border. The time indicates the elapsed time in seconds required for execution. The count is the number of pixels in the distance transform result that differ from the gold standard. The max is the magnitude of the single, largest difference and the rmse is the root mean squared error (difference). The gold standard is calculated via an exhaustive comparison and is therefore not usable for large data sets.

algorithm	time (sec)	count	max	rmse
gold	41.1	0	0	0.00000
danielsson	7.4	1179	1	0.02704
maurer	1.6	1006	1	0.02701

7 Phantom example

Example of the image that gives error in the algorithm from [1]: center of the image. Consider feature points $P = (-a, a)$, $Q = (0, b)$ and $R = (a, a+1)$ and let $L_0 = \mathbb{R} \times \{0\}$. Let $(0, x_{PQ}) \in L_0$ be a point equidistant from P and Q. Then, solution of equation $(x_{PQ} + a)^2 + a^2 = x^2 + b^2$ gives $x_{PQ} = \frac{b^2-2a^2}{2a}$. Similarly, if point $(0, x_{QR}) \in L_0$ is equidistant from Q and R, then x_{QR} solves $(x_{QR} - a)^2 + (a+1)^2 = x^2 + b^2$, so $x_{QR} = 1 + \frac{1}{2a} - \frac{b^2-2a^2}{2a}$. In particular, if $a = 24$ and $b = 34$, then $x_{PQ} \approx 0.15$ and $x_{QR} \approx 0.94$. This means, that interval (x_{PQ}, x_{QR}) is non-empty, but contains no integer.

If the binary image $I\colon \{-30,-29,\dots,30\}\times\{0,\dots,60\}\to\{0,1\}$ is such that P, Q, and R, are its only background pixels, then Q is not a feature point for $L_0\cap\{-30,-29,\dots,30\}\times\{0,\dots,60\}$, though algorithm VoronoiFT treats it as such, leading to associating it to as a feature point to $(1,0)$.

Example of the image that gives error in the algorithm from [1]: boundary of the image. Run the algorithm for the diagonal binary image $I\colon \{1,\dots,10\}\times\{1,\dots,10\}\to\{0,1\}$ with background being exactly its diagonal: $\{(i,i)\colon i=1,\dots,10\}$. Then VoronoiFT algorithms assign the spel $(1,1)$ as the closest to every spel $(i,1)$, which is incorrect for all $i>1$.

8 Appendix

Fact 6 *For $1<p<\infty$ if Δ is the L_p metric on $\mathbb{R}^k$, then it satisfies the properties (♣) and (♠).*

Proof. When $u_d=v_d$, we have $u=v$, in which case (♣) is trivially satisfied. So, in what follows we will consider only the case when $u_d<v_d$. Let $h(x_d)=\Delta(x,u)^p-\Delta(x,v)^p$ and notice that it is enough to show that function h is strictly increasing.

Indeed, to show that this implies (♣) assume that $\Delta(x,u)>\Delta(x,v)$. Since x^p is strictly increasing, $\Delta(x,u)^p>\Delta(x,v)^p$, and so $h(x_d)>0$. Hence $h(\bar{x}_d)>0$, as $x_d<\bar{x}_d$. But $h(\bar{x}_d)=\Delta(\bar{x},u)^p-\Delta(\bar{x},v)^p$, since $x_i=\bar{x}_i$ for all $i\neq d$. Thus, $\Delta(\bar{x},u)^p-\Delta(\bar{x},v)^p>0$, that is, that $\Delta(\bar{x},u)^p>\Delta(\bar{x},v)^p$. From this we conclude that $\Delta(\bar{x},u)>\Delta(\bar{x},v)$. The argument for the implication $\Delta(x,u)\geq\Delta(x,v)\Longrightarrow\Delta(\bar{x},u)\geq\Delta(\bar{x},v)$ is essentially the same.

To see (♠), note that h is continuous and that $\lim_{u_d\to\pm\infty}h(u_d)=\pm\infty$. (The argument for the limit requires some algebra work. It follows from a simple estimate [2, Lemma 3, page 121], proven with calculus tools, that $(a+b)^p\geq a^p+pba^{p-1}$ for non-negative a and b.) Thus, by Intermediate Value Theorem (see e.g. [2]), there exists an $\hat{x}\in\mathbb{R}$ with $h(\hat{x})=0$. Let $x_{\overline{uv}}\in\mathbb{R}_d(\bar{x})$ be such that $(x_{\overline{uv}})_d=\hat{x}$. Then, for every $x\in\mathbb{R}_d(\bar{x})$ if $x_d<(x_{\overline{uv}})_d$, then $\Delta(x,u)^p-\Delta(x,v)^p=h(\bar{x}_d)<h((x_{\overline{uv}})_d)=0$, so $\Delta(x,u)<\Delta(x,v)$. Similarly, $x_d>(x_{\overline{uv}})_d$ implies $\Delta(x,u)>\Delta(x,v)$, so $x_{\overline{uv}}$ satisfies (♠).

To prove that $h(x_d)=\sum_{i=1}^k|x_i-u_i|^p-\sum_{i=1}^k|x_i-v_i|^p$ is strictly increasing it is enough to show that it has non-negative derivative at all points except possibly for $x_d=u_d$ and $x_d=v_d$. Since $h'(x_d)=\frac{d}{dx_d}\left(|x_d-u_d|^p-|x_d-v_d|^p\right)$,

for $u_d \leq v_d < x_d$ we have $h'(x_d) = p\left((x_d - u_d)^{p-1} - (x_d - v_d)^{p-1}\right) \geq 0$ as $x_d - u_d \geq x_d - v_d > 0$ and function x^{p-1} is non-decreasing on $(0, \infty)$. Similarly, for $x_d < u_d \leq v_d$ we have $h'(x_d) = p\left(-(u_d - x_d)^{p-1} + (v_d - x_d)^{p-1}\right) \geq 0$ as $v_d - x_d \geq u_d - x_d > 0$. Finally, $h'(x_d) = p\left((x_d - u_d)^{p-1} + (v_d - x_d)^{p-1}\right) > 0$ for the remaining case $u_d < x_d < v_d$. ■

References

[1] Maurer, C.R., Jr., 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**(2) (2003), 265–270.

[2] H.L. Royden, *Real Analysis*, MacMillan Publishing Company, New York, 1988.

[3] Tustison, N.J., Siqueira, M., and Gee, J.C.: *N*-D Linear Time Exact Signed Euclidean Distance Transform, Penn Image Computing and Science laboratory, UPenn, February 17, 2006.