# The Stepping Distance Transformation

Wim H. Hesselink, 2nd November 1998 (whh179)
Dept. of Mathematics and Computing Science, Rijksuniversiteit Groningen
P.O.Box 800, 9700 AV Groningen, The Netherlands
Email: `wim@cs.rug.nl`, Web site: `http://www.cs.rug.nl/~wim` .

**Abstract**

An algorithm *Mscan* is proposed for the computation of the distance transform of a feature in an image with respect to a chamfer distance. The algorithm is naïvely simple and can therefore be easily parallellized. It seems to be new, even for special cases. It is not correct for arbitrary chamfer distances but, by a general analysis of masks and their weights, sufficient conditions for its correctness are obtained. If the image is rectangular with edges parallel to the coordinate axes, all chamfer distances used to approximate the Euclidean distance satisfy these conditions.

**Keywords:** distance transform, chamfer distance, parallel algorithm

## 1 Introduction

For more than thirty years, distance transforms of binary images have played a basic role in computational geometry, with applications in many areas including image processing, graphics and pattern recognition, e.g., cf. [RoP66, RoP68, Pag92]. Since the exact Euclidean distance transform is not always relevant, and is often regarded as computationally too expensive, many approximating algorithms have been proposed, cf. [RoP66, Dan80, Bor84].

The aim of this paper is to present an easy algorithm for the distance transform with respect to a so-called chamfer distance. The task of such an algorithm is to compute in a rectangular grid the shortest distances from a certain set (the source or feature) to all elements of a bigger set (the image) by means of paths consisting of basic steps which are counted according to their weights. The precise definition is given in Section 2.

The problem goes back to [RoP66] for the case of the grid with steps from every point to one of the four neighbour points. In [RoP68], other distance functions are considered which correspond to more general step sets. Later such distance functions have been described as weighted or chamfer or pseudo-Euclidean distance transforms. The main research questions have been parallel computation of these distance transforms, e.g. [BHT90], and how good they approximate the Euclidean distance transform, e.g. [Bor84, Bor86]. Usually, a sequential implementation is taken for granted. The problem is generalized to arbitrary dimensions in [Bor84].

It is our aim to develop a sequential algorithm for arbitrary dimensions that is as simple as possible and therefore also easy to parallellize. In Section 3 we present a general solution in which the additive structure of the ambient space is ignored. Section 4 contains the algorithm *Mscan*. It is simple, has a very regular structure, and can therefore easily be parallellized on a shared memory machine, but it is only correct under specific assumptions on the mask and the image. In Section 5, we give sufficient conditions to imply this correctness, as well as some relevant examples. In particular, we show that all natural chamfer distances satisfy these sufficient conditions.

## 2 Problem description

We let $V$ be the ambient space, which is a vector space over $\mathbb{R}$ or a module over $\mathbb{Z}$. The actual starting point is to choose a set $S$ of allowed steps and a weight $w(r)$ for each allowed step $r \in S$. The combination $(S, w)$ is called a mask, cf. [Bor86]:

**Definition.** A *mask* is a pair $(S, w)$ such that $S$ is a finite subset of $V$ and $w$ is an integer valued function on $S$ with $w(r) \geq 1$ for all $r \in S$. An element $r \in S$ is called a *step*, and $w(r)$ is called its *weight*. Note that taking a step $r$ at position $x$ means replacing $x$ by $x + r$.

Let $F$ be a subset of $V$, to be called the *image*. A *path* from $x$ to $y$ within $F$ is a finite sequence $p(0), \ldots, p(m)$ with $m \geq 0$, such that $x = p(0)$, and $y = p(m)$, and $p(i) \in F$ for all $i$, and $p(i+1) - p(i) \in S$ for all $i$ with $0 \leq i < m$. The *weight* of such a path is the sum of the weights of the steps $\sum_i w(p(i+1) - p(i))$. The *distance* $d_F(x, y)$ is the minimum of the weights of all paths from $x$ to $y$ within $F$. We define $d_F(x, y) = \infty$ if no paths from $x$ to $y$ within $F$ exist. In the special case with $F = V$, we write $d_V(x, y) = d(x, y)$.

It is clear that $d_F(x, y) \geq 0$ and $d_F(x, x) = 0$ for all $x$, $y \in F$. Conversely, $d_F(x, y) = 0$ implies $x = y$, since $w(r) \geq 1$ for every step $r$. Since a path from $x$ to $y$ composed with a path from $y$ to $z$ yields a path from $x$ to $z$, we have the triangle inequality

$$d_F(x, z) \leq d_F(x, y) + d_F(y, z) \ .$$

In general, symmetry may fail: $d_F(x, y)$ need not be equal to $d_F(y, x)$. In fact, if $r \in S$, we do not require $-r \in S$.

Now let $B$ be a subset of $F$, to be called the *source*. For $x \in F$, we then define $D_F(x)$ to be the minimal value $d_F(b, x)$ where $b$ ranges over $B$. The function $D_F$ is called the *stepping distance transform* within $F$, with respect to steps $S$, weights $w$, and source $B$. We clearly have $D_F(x) \geq 0$ for all $x$, and $D_F(x) = 0$ for all $x \in B$.

The aim of this note is to present an algorithm to determine $D_F(x)$ for all $x \in F$, assuming that $F$ is a finite set. We thus introduce an array variable $dt$ with index set $F$ and aim at the postcondition

$$Q \ : \quad (\forall\, x \in F :: \ dt[x] = D_F(x)) \ .$$

We refer to [Bor86] for the complementary question how to design the mask $(S, w)$ in such a way that the stepping distance is a good approximation of the Euclidean distance.

# 3 A general algorithm

In view of postcondition $Q$ and the obvious fact that $D_F(x) = 0$ for all $x \in B$, we choose the invariants

(Jq0) $\quad (\forall\, x \in F :: \ D_F(x) \leq dt[x])$ ,
(Jq1) $\quad (\forall\, x \in B :: \ dt[x] = 0)$ .

These invariants can easily be initialized by

Init: **for all** $x \in F \setminus B$ **do** $dt[x] := \infty$ **od** ;
**for all** $x \in B$ **do** $dt[x] := 0$ **od** .

We now note that the triangle inequality implies, for all $x \in F$ and $r \in S$,

$$(0) \qquad x + r \in F \quad \Rightarrow \quad D_F(x + r) \ \leq \ D_F(x) + w(r) \ .$$

It follows that the predicates (Jq0) and (Jq1) are preserved by the assignments, for $x \in F$ and $r \in S$ with $x + r \in F$,

$$dt[x + r] := \ dt[x + r] \ \mathbf{min} \ (dt[x] + w(r)) \ .$$

Now assume a state has been reached in which such assignments have no effect anymore, i.e., that we have

$$T : \quad (\forall\, x \in F, r \in S : x + r \in F : \ dt[x + r] \leq dt[x] + w(r)) \ .$$

The conjunction of $T$ and (Jq1) implies, by the definition of $D_F$ and induction over paths, that $dt[x] \leq D_F(x)$ for all $x \in F$. Therefore, (Jq0) implies postcondition $Q$. This proves that our aim is served by the program (Init; Main), where Main is the loop

Main: **while** $\neg T$ **do**
  *choose* $x \in F, r \in S$ *with* $x + r \in F \ \wedge\ dt[x] + w(r) < dt[x+r]$ ;
  $dt[x+r] := \ dt[x] + w(r)$ ;
**od** .

This would make an excellent UNITY program, cf. [CM88], but we prefer more concrete algorithms. Actually, we aim at an efficient algorithm that is correct for practical masks, but not necessarily for all masks.

# 4 Using the additive structure

We now use the additive structure of the ambient space $V$ to reduce the nondeterminacy.

The set $F$ is said to be $S$–*convex* iff, for every $x \in F$ and $r \in S$, the set of integers $i$ with $x + i * r \in F$ is a "segment", i.e., there exist $p$ and $q$ with $x + i * r \in F$ if and only if $p \leq i < q$.

We now assume that $F$ is $S$–convex and finite. Every step $r \in S$ induces a partition of $F$ in segments: for every step $r \in S$, we define the low $r$–boundary $Low(r)$ and, for every $y \in Low(r)$, we define the $r$–bound $upb(r, y)$ by

$$\begin{aligned} Low(r) &= \{x \in F \mid x - r \notin F\} \ , \\ upb(r,y) &= (\texttt{MIN}\ i : i > 0 \ \wedge\ y + i * r \notin F : i) \ . \end{aligned}$$

Then $F$ is the disjoint union of line segments starting in the low boundary, in the sense that

$$(\forall\, x \in F :: (\exists!\, y, i ::\ y \in Low(r) \ \wedge\ x = y + i * r)) \ ,$$
$$(\forall\, y, i : y \in Low(r) :\ y + i * r \in F \quad \equiv \quad 0 \leq i < upb(r, y)) \ .$$

As a systematic way to organize the assignments considered in Section 3, for each step $r \in S$ separately, we now introduce

**procedure** *scan* $(r) =$
  **for all** $y \in Low(r)$ **do**
    $x := \ y$ ;  $k := \ 1$ ;
    **while** $k < upb(r, y)$ **do**
      $dt[x + r] := \ dt[x + r]$ **min** $(dt[x] + w(r))$ ;
      $x := \ x + r$ ;  $k := \ k + 1$ ;
    **od**
  **od** .

We specify procedure *scan* by means of a Hoare triple, using a function $H$ to represent the initial value of array $dt$:

$$\{\ (\forall\, x : x \in F :: H(x) = dt[x]) \ \wedge\ (\forall\, x : x \notin F :: H(x) = \infty)\ \}$$
$$scan\ (r)$$
$$\{\ (\forall\, x \in F :: dt[x] = (\texttt{MIN}\ i : i \geq 0 : H(x - i * r) + i * w(r)))\ \} \ .$$

We claim that procedure *scan* satisfies this specification. The proof goes as follows. One first proves the following invariant for the inner loop:

(Kq)
$$\begin{aligned} & 0 < k \leq upb(r,y) \quad \wedge \quad x = y + (k-1) * r \\ \wedge \quad & (\forall\, j : 0 \leq j < k : dt[y + j * r] = (\texttt{MIN}\ i : i \geq 0 : H(y + j * r - i * r) + i * w(r))) \\ \wedge \quad & (\forall\, j : k \leq j < upb(r,y) : dt[y + j * r] = H(y + j * r)) \ . \end{aligned}$$

This yields the term of the postcondition for all elements $x$ in the $r$-segment starting in $y$. One then concludes using that $F$ is the disjoint union of these segments.

The nice point about procedure *scan* is that it can be parallellized in a straightforward way: the vectors $y$ can be distributed over different processors.

We now want to replace the algorithm of Section 3 by

*Mscan :* Init ;
**for all** $r \in S$ **do** *scan* $(r)$ **od** .

The loop in *Mscan* clearly preserves the invariants (Jq0) and (Jq1). The question is whether *Mscan* establishes postcondition $Q$. In general it does not, but before going into that, we have to deal with another point. In previous cases, each **for** loop was independent of the order in which the loop variable was selected from its range. Since *scan* $(r)$ modifies array $dt$, however, the order of selecting the steps to scan influences the result of *Mscan*. Yet, we assume that an arbitrary order is chosen nondeterminately.

For the analysis of the postcondition of *Mscan* we have to fix the order. So we assume that *Mscan* selects the steps according to some enumeration $r_0, \ldots, r_{n-1}$ of $S$. Then, by induction in $n$ and using the specification of *scan*, one sees that *Mscan* gives $dt[x]$ the minimal weight of the paths within $F$ from any point of $B$ to $x$ in which the steps taken have ascending step numbers. Now, addition in $V$ is commutative, so the order of the steps is irrelevant for the destination of the path, but the order may well influence the question whether all points of the path are in $F$.

Let us call a path in which the steps taken have ascending step numbers, an *ordered path*. Let $d'_F(x, y)$ be the minimum weight of the ordered paths from $x$ to $y$ within $F$, and let $D'_F(x)$ be the minimum value of $d'_F(b, x)$ for all $b \in B$. Algorithm *Mscan* has the postcondition that $dt[x] = D'_F(x)$ for all $x \in F$. Therefore, *Mscan* establishes $Q$ if the functions $d'_F$ and $d_F$ are equal.

**Definition.** The set $F$ is defined to be *stepwise good* (with respect to the enumeration of $S$ and the weights $w$) iff $d_F = d'_F$.

It follows that, if the set $F$ is stepwise good, *Mscan* establishes postcondition $Q$. If $F$ is not stepwise good, one can establish postcondition $Q$ by repeating the body of *Mscan* until nothing changes anymore.

*Remark.* Parallellization of *Mscan* in shared memory requires synchronization points after each call of *scan*. If $S$ contains both $r$ and $-r$, the composition of *scan*$(r)$ and *scan*$(-r)$ can be parallellized together since $r$ and $-r$ induce the same partition of image $F$. Therefore, usually, the number of synchronization points can be reduced to $\frac{1}{2}\#S$. □

# 5 Stepwise goodness

Since *Mscan* establishes $Q$ if $F$ is stepwise good, and since *Mscan* can be implemented efficiently, it is important to obtain concrete criteria that imply stepwise goodness.

**Lemma 0.** Image $F$ is stepwise good if and only if function $d'_F$ satisfies the triangle inequality.

*Proof.* If $F$ is stepwise good, then $d'_F$ equals $d_F$ and hence satisfies the triangle inequality. Conversely, let $d'_F$ satisfy the triangle inequality. It is clear that $d_F \leq d'_F$. So it suffices to prove $d'_F \leq d_F$. Consider $x$ and $y \in F$ with distance $u = d_F(x, y) < \infty$. We have to prove $d'_F(x, y) \leq u$. Let $p(0), \ldots, p(m)$ be an $S$-path from $x$ to $y$ within $F$ with weight $u$. Since $d'_F(p(i + 1) - p(i)) \leq w(p(i + 1) - p(i))$ for all $i$, repeated application of the triangle inequality yields $d'_F(x, y) \leq \sum_i w(p(i + 1) - p(i)) = u$. □

We now restrict ourselves to the most important case that $V$ is the module $V = \mathbb{Z}^k$ for $k \geq 1$. Elements of $V$ are called (integral) vectors. We use coordinates on $V$ indexed by $i$ with $0 \leq i < k$. For an integral vector $b$, the block $B(b)$ is defined by

$$x \in B(b) \quad \equiv \quad (\forall\, i : 0 \leq i < k : 0 \leq x_i \leq b_i \quad \vee \quad b_i \leq x_i \leq 0)\ .$$

Note that $B(b)$ is a finite set that contains the elements 0 and $b$ and that the sides of $B(b)$ are parallel to the coordinate axes.

A subset $F$ of $V$ is called a *block* iff there are integral vectors $a$ and $b$ such that

$$x \in F \quad \equiv \quad x - a \in B(b)\ .$$

It is easy to verify that every block is $S$–convex, for any step set $S$. We are interested on conditions on $S$ and $w$ such that every block is stepwise good.

**Definition.** Let an $S$–path $p(0)$, ..., $p(m)$ from $x$ to $y$ be called *monotonic* iff, for every coordinate $i$, the sequence of numbers $p(0)_i$, ..., $p(m)_i$ is ascending or descending. Mask $(S, w)$ is called *monotonic* iff, for every pair $x$, $y \in V$, there is a monotonic $S$–path from $x$ to $y$ with weight $d(x, y)$.

**Lemma 1.** Let $(S, w)$ be a monotonic mask and let $F$ be a block. Then $F$ is stepwise good and satisfies $d_F = d$.

*Proof.* Let $x$ and $y \in F$. Since $(S, w)$ is monotonic, there is a monotonic shortest $S$–path from $x$ to $y$. The corresponding ordered $S$-path from $x$ to $y$ is obtained by permutation of the difference vectors of the path. In this way the path remains monotonic. On the other hand, since $F$ is a block, every monotonic $S$–path from $x$ to $y$ stays within $F$. This shows the existence of a shortest $S$–path from $x$ to $y$, which is an ordered path within $F$. Therefore, the three functions $d'_F$, and $d_F$, and $d$ coincide on $F$. In particular, $F$ is stepwise good. □

*Example 0.* Take $k = 2$. We allow steps of the king in chess. So let $S$ consist of the eight vectors $(\pm 1, 0)$, $(0, \pm 1)$, $(\pm 1, \pm 1)$, with $\pm$ standing for independent choices of $+$ or $-$. Let $w(\pm 1, 0) = w(0, \pm 1) = c$ and $w(\pm 1, \pm 1) = d$.

If $c \leq d$, then $(S, w)$ is monotonic and every block in $\mathbb{Z}^2$ is stepwise good. See e.g. [BHT90] for the case with $c = 3$ and $d = 4$. For $d < c$, optimal paths tend to use diagonals rather than horizontal and vertical vectors. It is easy to see that then most blocks are not stepwise good. □

It seems likely that Lemma 1 can be reversed, i.e., that, if every block $F$ is stepwise good, the mask is monotonic. We have a proof that the mask is monotonic if every block $F$ satisfies $d_F = d$. Since this is only a partial result, we leave this matter to the interested reader.

In general, monotonicity of mask $(S, w)$ is not easy to decide. We therefore introduce a more concrete criterion.

**Definition.** Mask $(S, w)$ is called *blockwise convex* iff $d(0, u) \leq w(r)$ holds for every step $r$ and every vector $u \in B(r)$.

It is clear that blockwise convexity is decidable. It is likely that all chamfer distances proposed to approximate the Euclidean distance are blockwise convex. E.g., in [Bor86], blockwise convexity is implicitly required (explicitly for the case of Example 0). We show below that blockwise convexity implies monotonicity.

*Example 1.* Take $k = 2$ as above. We allow steps of the knight in chess, and small steps of the rook. So we let $S$ consist of the twelve vectors $(\pm 1, 0)$, $(0, \pm 1)$, $(\pm 2, \pm 1)$, and $(\pm 1, \pm 2)$. Let $w(\pm 1, 0) = w(0, \pm 1) = c$ and $w(\pm 2, \pm 1) = w(\pm 1, \pm 2) = d$. Mask $(S, w)$ is blockwise convex if and only if $2 * c \leq d$. For $c = 1$ and $d = 2$ one gets distance $d_4$ of [RoP68]. □

**Lemma 2.** Let $(S, w)$ be blockwise convex. Then $d(0, u) \leq d(0, x)$ holds for every pair of vectors $x$, $u \in V$ with $u \in B(x)$.

*Proof.* Again we use induction. Let $x$, $u \in V$ with $u \in B(x)$ be given. If $x = 0$ then $u = 0$, and the assertion is trivial. So, assume $x \neq 0$. By induction, we may assume that $d(0, v) \leq d(0, y)$ holds for every pair of vectors $y$, $v \in V$ with $v \in B(y)$ and $d(0, y) < d(0, x)$. Let $p(0)$, ..., $p(m)$ be an $S$–path from 0 to $x$ with weight $d(0, x)$. Since $x \neq 0$ we have $m > 0$. Let $y = p(m - 1)$ and $r = x - y$. Then $r \in S$ and $d(0, x) = d(0, y) + w(r)$.

By Lemma 3 below, there exists a vector $t \in B(u) \cap B(y)$ with $u - t \in B(x-y)$. We now have $d(0,t) \leq d(0,y)$ by induction, and $d(0, u-t) \leq w(r)$ since $(S,w)$ is blockwise convex. It follows that $d(t,u) \leq w(r)$, and hence $d(0,u) \leq d(0,y) + w(r) = d(0,x)$. □

**Lemma 3.** Let $x$, $y$, $u \in V$ be such that $u \in B(x)$. Then there exists $t \in B(u) \cap B(y)$ with $u - t \in B(x-y)$.

*Proof.* All conditions of the form $v \in B(z)$ can be verified componentwise. We use this to construct vector $t$ componentwise. In other words, we may assume that $k = 1$ and $V = \mathbb{Z}$. By symmetry we may assume that $0 \leq u \leq x$. If $y \leq 0$, we can choose $t = 0$. If $0 \leq y \leq u$, we can choose $t = y$. If $u \leq y$, we can choose $t = u$. □

**Lemma 4.** Let $(S,w)$ be blockwise convex. Then $(S,w)$ is monotonic.

*Proof.* By induction on $d(0,x)$ we prove the existence of a monotonic shortest $S$–path from 0 to $x$. If $x = 0$ the assertion is trivial. So assume $x \neq 0$. Let $p(0), \ldots, p(m)$ be a shortest $S$–path from 0 to $x$. If $m \leq 1$, the path consists of at most one step and is therefore monotonic.

Now assume $m > 1$. As before, put $y = p(m-1)$ and $r = x - y$. By Lemma 3 with $u := x$, there exists $t \in B(x) \cap B(y)$ with $x - t \in B(x-y)$. By Lemma 3, we have $d(0,t) \leq d(0,y)$ and $d(0, x-t) \leq d(0, x-y)$. Using $m > 1$, we infer that $d(0,t) \leq d(0,y) < d(0,x)$ and $d(t,x) \leq d(y,x) < d(0,x)$. So, by induction, we can choose a monotonic shortest path from 0 to $t$ and a monotonic shortest path from $t$ to $x$. Since $d(0,t) \leq d(0,y)$ and $d(t,x) \leq d(y,x)$, the composition of these two shortest paths is a shortest path from 0 to $x$. Since $t \in B(x)$, the composition of these two monotonic paths is monotonic. Therefore, there is a monotonic shortest path from 0 to $x$. □

This proves that, if $(S,w)$ is blockwise convex, every block $F$ in $V$ is stepwise good. It does not imply anything for other images.

*Example 2.* Again take $k = 2$ and let $S$ consist of the four vectors $(1,0)$, $(-1,0)$, $(0,1)$, $(0,-1)$, enumerated in this order. Let $w(r) = 1$ for all $r \in S$. Then $(S,w)$ is blockwise convex. Let $F$ be the triangular region defined by

$$(x_0, x_1) \in F \quad \equiv \quad 0 \leq x_1 \leq x_0 \leq 9 \ .$$

Then $F$ is not stepwise good with respect to the enumeration order of $S$. In fact, we have $d_F(x,y) = 2$ and $d'_F(x,y) = \infty$ for the vectors $x = (3,3)$ and $y = (2,2)$. □

*Example 3.* Lemma 4 cannot be reversed: blockwise convexity for a mask is stronger than monotonicity. Let $S$ consist of the vectors $(\pm 2, 0)$, $(\pm 1, 0)$, $(0, \pm 2)$, $(0, \pm 1)$ with $w(\pm 2, 0) = w(0, \pm 2) = 1$ and $w(\pm 1, 0) = w(0, \pm 1) = 2$. Then $(S,w)$ is monotonic but not blockwise convex. □

# 6 Conclusions

Since the stepping distance transformation is most often used as an approximation of the Euclidean distance transformation, the assumption that the mask is blockwise convex holds in most practical cases. The image is also usually a block in our technical sense, i.e., with sides parallel to the coordinate axes. Then algorithm *Mscan* is applicable. Whenever applicable, *Mscan* is an excellent algorithm, likely to perform better than its competitors. Indeed, its time complexity is linear in the product of $\#S$ and $\#F$, and it needs no additional data structures. It can easily be parallellized on shared memory machines.

# References

[Bor84] G. Borgefors: Distance transformations in arbitrary dimensions. Computer Vision, Graphics, and Image Processing **27** (1984) 321–345.

[Bor86] G. Borgefors: Distance transformation in digital images. Computer Vision, Graphics, and Image Processing **34** (1986) 344–371.

[BHT90] G. Borgefors, T. Hartmann, S.L. Tanimoto: Parallel distance transforms on pyramid machines: theory and implementation. Signal processing (1990) 61–86.

[CM88] K.M. Chandy, J. Misra: Parallel Program Design, A Foundation. Addison–Wesley, 1988.

[Dan80] P.E. Danielsson: Euclidean distance mapping. Comput. Graphics Image process. **14** (1980) 227–248.

[Pag92] D.W. Paglieroni: Distance transforms: properties and machine vision applications. CVGIP: Graphical Models and Image Processing **54** (1992) 56–74.

[RoP66] A. Rosenfeld, J.L. Pfaltz: Sequential operations in digital picture processing. J ACM **13** (1966) 471–494.

[RoP68] A. Rosenfeld, J.L. Pfaltz: Distance functions on digital pictures. Pattern Recognition **1** (1968) 33–61.