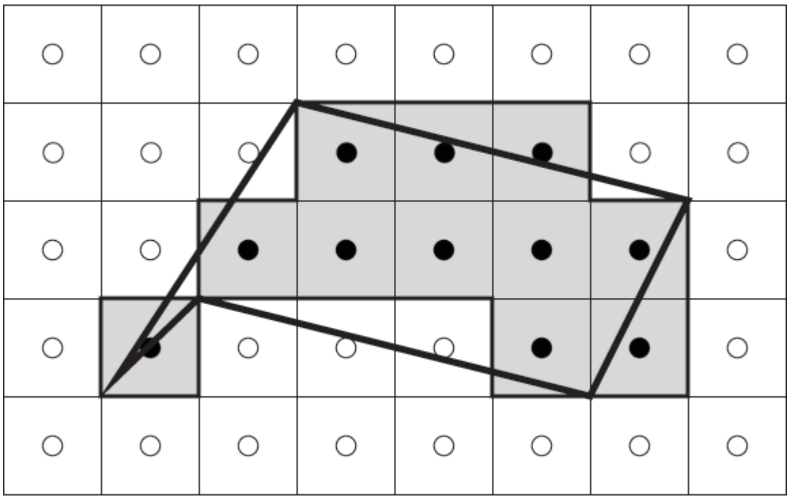
A Linear Online 4-DSS Algorithm

V. Kovalevsky 1990: algorithm **K1990**

We consider the recognition of 4-DSS*s* on the frontier of a region in the 2D cellular grid.



Algorithm **K1990** is one of the simplest (implementation) and most efficient linear online 4-DSS recognition algorithms.

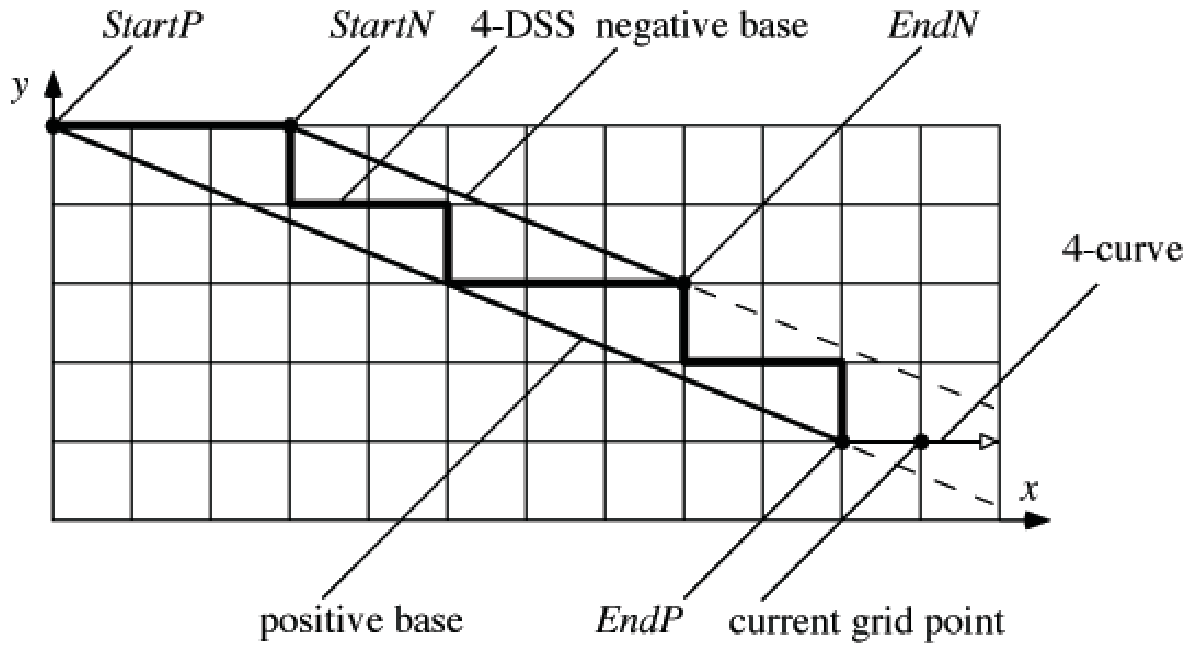
It is based on the calculation of a *narrowest strip* defined by the nearest support below and above (see Theorem 2 on page 4 in Lecture 15, and figure on next page).

It resembles the linear offline algorithm by T.A. Anderson and C.E. Kim (1985) and a linear online algorithm by E. Creutzburg, A. Hübler and O. Sýkora (1988) for 8-arcs.

Notation for K1990

The algorithm follows a digital 4-curve and extends a 4-DSS as long as it has at most two directions and all of its grid points lie between or on a pair of parallel lines that have a main diagonal distance of less than $\sqrt{2}$.

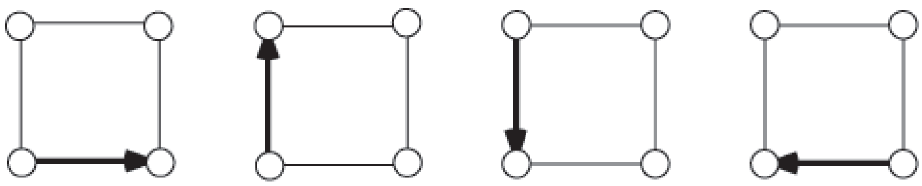
On the parallel line to the left of the digital curve, we define a *negative base* between the grid points p_N=*StartN* and q_N=*EndN*.



On the parallel line to the right of the digital curve, we define a *positive base* between the grid points p_P=*StartP* and q_P=*EndP*.

Initialization of a new 4-DSS

Let the first step be from $r_1 = (x_1, y_1)$ to the 4-adjacent $r_2 = (x_2, y_2)$; let $p_N = p_P = r_1$, $q_N = q_P = r_2$, $a = x_2 - x_1$, $b = y_2 - y_1$, $c = ay_2 - bx_2$, and note the direction of the step from r_1 to r_2.

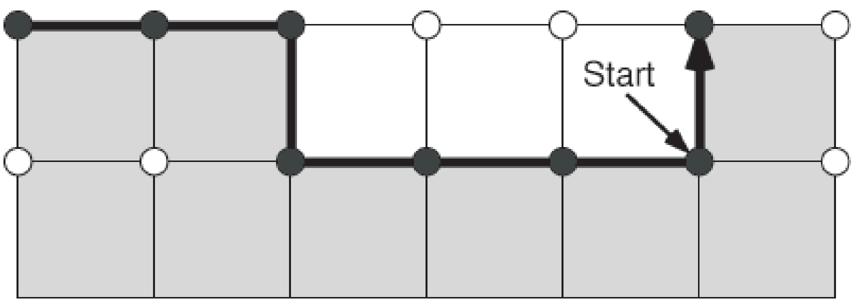


Left to right in figure: $(a, b)^T$ = (1,0), (0,1), (0,-1), or (-1,0).

Here: negative base = positive base, and vector $(a, b)^T$ is parallel to both.

Number of possible directions on a 4-DSS

Whenever a new step is not in one of the (at most two) directions in the current 4-DSS, we start a new DSS.

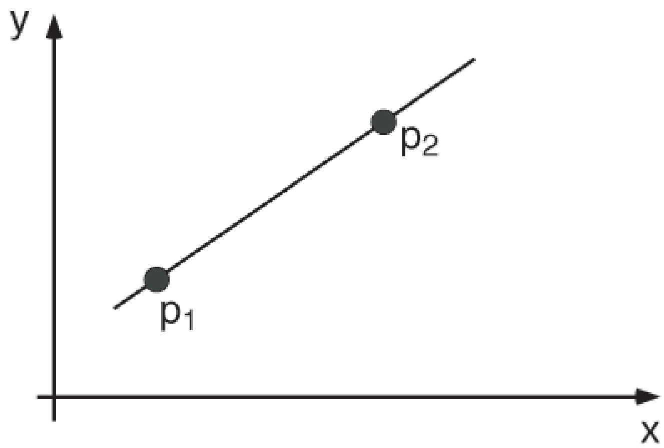


When there has been only one direction so far we continue the current DSS.

If we continue with two directions, we decide based on values a and b; see next pages for details.

Two-Point Equation of a Line

Assume two points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ on a straight line.



The equation of the straight line is

$$\frac{y - y_1}{y_2 - y_1} = \frac{x - x_1}{x_2 - x_1}$$

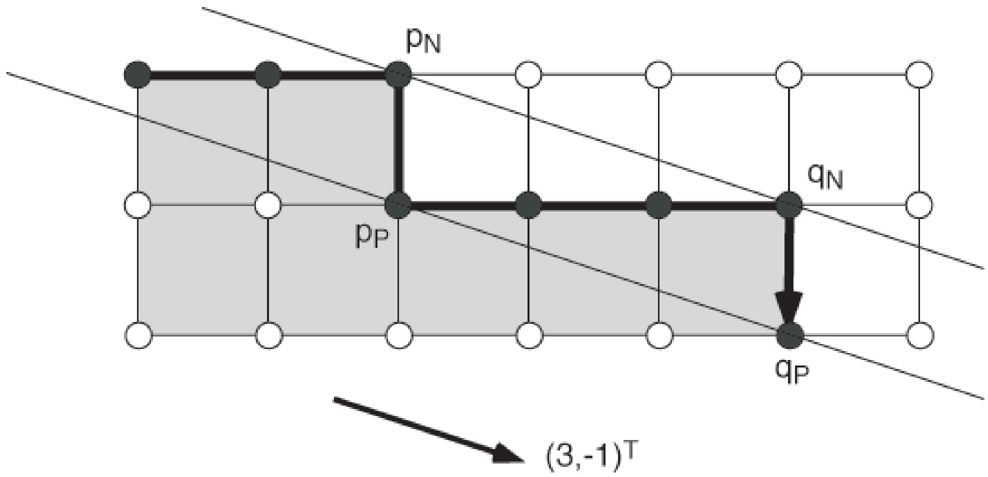
or

$$y = \frac{y_2 - y_1}{x_2 - x_1} x + \frac{y_1(x_2 - x_1) - x_1(y_2 - y_1)}{x_2 - x_1}$$

$$= \frac{b}{a} x + \frac{c}{a}$$

If p_1 and p_2 are grid points, then a, b, c are integers.

Case of Two Directions



Let points p_N, q_N be on straight line

$$y = \frac{b}{a}x + \frac{c}{a}$$

Note: because the straight line is incident with two grid points, it follows that a, b, c are integers, and we can choose a, b such that they are relatively prime.

Points p_P, q_P in the example above are on straight line

$$y = \frac{b}{a}x + \frac{c}{a} - 1$$

Note: Theorem 2 in Lecture 15 allows that this line moves further away from line $y = \frac{b}{a}x + \frac{c}{a}$, namely by $\sqrt{2}$ in main diagonal distance.

Two Inequalities

All grid points on the 4-DSS have coordinates (x, y) which are on or between two straight lines. They satisfy

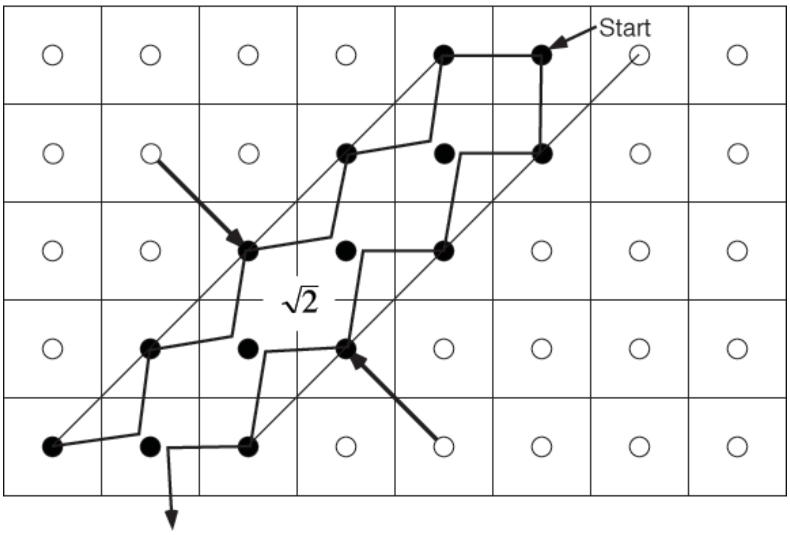
$$0 \leq bx - ay + c \tag{1}$$

(i.e., they are right of, or on the "negative straight line") and

$$bx - ay + c \leq |a| + |b| - 1 \tag{2}$$

(i.e., they are left of, or on a second straight line which is $\sqrt{2}$ away from the "negative straight line" in main diagonal distance.)

Note: (2) allows $bx - ay + c = |a| + |b| - 1$ (i.e., a second line in distance $\sqrt{2}$). This defines a "symmetric" algorithm; a trace of a



"double staircase" (see example of a subsequence of a border cycle) would contain more than two different directions, leading to a start of a new segment where expected.

Continuation of Algorithm

(Case of two Directions)

Suppose inequalities (1) and (2) are true for $n-1$ accepted grid points $r_1 = (x_1, y_1), \ldots, r_{n-1} = (x_{n-1}, y_{n-1})$ of the 4-DSS.

Let $r_n = (x_n, y_n)$ be the next grid point to be tested, which is 4-adjacent to $r_{n-1} = (x_{n-1}, y_{n-1})$.

c is an integer such that

$$c = ay - bx$$

for any (up to point r_{n-1}) grid point (x, y) on the negative base.

(For example, if $q_N = (6, 1)$ in the figure on page 4, then $c = 9$.)

Sometimes c needs to be updated during the algorithm.

Let $h(x, y) = bx - ay + c$.

Because $r_n = (x_n, y_n)$ is 4-adjacent to $r_{n-1} = (x_{n-1}, y_{n-1})$ (i.e., it differs from r_{n-1} only in one coordinate by 1), the value of $h(r_n)$ can only differ either by b or by a from $h(r_{n-1})$.

If $0 < h(r_n) < |a| + |b| - 1$, then r_n is accepted and no parameter needs to be updated. Otherwise consider the cases on the next page.

(i) $h(r_n) = 0$: r_n is on the negative base, and the n vertices form a 4-DSS; let $q_N := r_n$.

(ii) $h(r_n) = |a| + |b| - 1$: r_n is on the positive base, and the n vertices form a 4-DSS; let $q_P := r_n$.

(iii) $h(r_n) = -1$ or $h(r_n) = |a| + |b|$: the n vertices form a 4-DSS because the new grid point r_n is still within the distance limits from the points between the two supporting lines but the values a, b, and c need to be updated:

(A) if $h(r_n) = -1$ then

begin

$q_N := r_n; \quad p_P := q_P; \quad (a, b) := r_n - p_N;$

end

(B) if $h(r_n) = |a| + |b|$ then

begin

$q_P := r_n; \quad p_N := q_N; \quad (a, b) := r_n - p_P;$

end

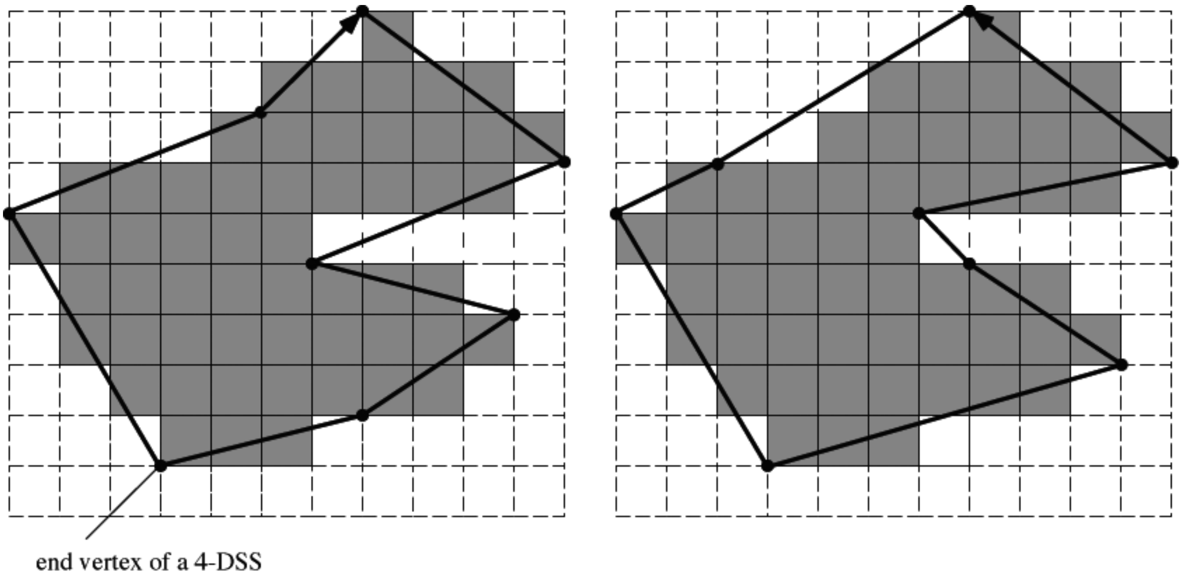
and for $q_N = (x, y)$ let $c = ay - bx$.

(iv) otherwise, the n vertices do not form a 4-DSS; stop at the previous vertex r_{n-1}, and initialize a new 4-DSS.

In cases (iii.A) and (iii.B), we have new values a, b, and c and new endpoints of positive or negative base.

In cases (i) and (ii), we have to move either q_N or q_P forward into position r_n.

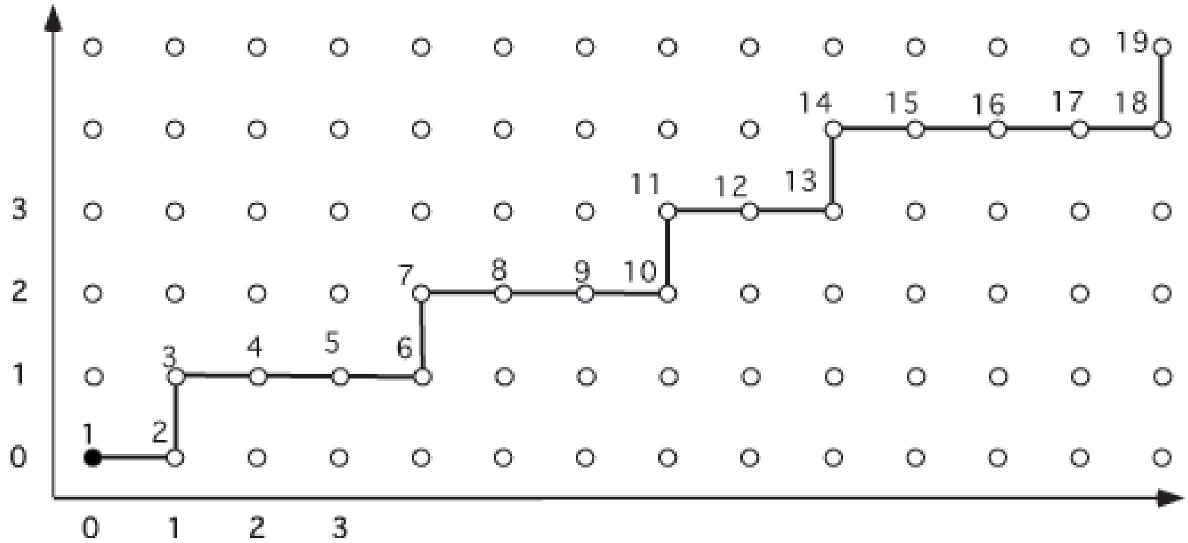
Non-Unique Segmentation Results



A clockwise and an anticlockwise traversal of the frontier of a digital region that produce different segmentations into maximum-length 4-DSSs. The start point of the traversal will also influence the result in general.

Example

$n = 2$: We start tracing at points $r_1 = (0, 0)$ and $r_2 = (1, 0)$. We have $p_N := p_P := (0, 0)$, $q_N := q_P := (1, 0)$, $a := 1$, $b := 0$, and $c = 0$. Only one direction (RIGHT) so far.



$n = 3$: With $r_3 = (1, 1)$ comes a second direction (UP). Point $q_N = (1, 0)$ on the negative base defines $c = 0$. We have $h(r_3) = -1$. We are at case (iii.A) on page 8:
$q_N := (1, 1);\;\; p_P := (1, 0);\;\; (a, b) := (1, 1);\;\; c = 0$

$n = 4$: For $r_4 = (2, 1)$ we continue to have two directions (RIGHT, UP). We have $h(r_4) = 1 = |a| + |b| - 1$. Case (ii) defines $q_P := (2, 1)$.

$n = 5$: With $r_5 = (3, 1)$ we get $h(r_5) = 2 = |a| + |b|$. Case (iii.B) defines $q_P := (3, 1)$, $p_N := (1, 1)$, $(a, b) := (2, 1)$, and $c := 1$.

$n = 6$: With $r_6 = (4, 1)$ we get $h(r_6) = 3 = |a| + |b|$. Case (iii.B) defines $q_P := (4,1)$, $p_N := (1,1)$, $(a,b) := (3,1)$, and $c := 2$.

$n = 7$: With $r_7 = (4,2)$ we get $h(r_7) = 0$, $q_N := (4,2)$ from (i).

$n = 8$: With $r_8 = (5,2)$ we get $h(r_8) = 1$. Because of $0 < 1 < |a| + |b| - 1 = 3$, we can continue without any update.

$n = 9$: With $r_9 = (6,2)$ we get $h(r_9) = 2$. Nothing to do.

$n = 10$: $r_{10} = (7,2)$ gives $h(r_{10}) = 3$, $q_P := (7,2)$, see (ii).

$n = 11$: $r_{11} = (7,3)$ gives $h(r_{11}) = 0$, $q_N = (7,3)$, see (i).

$n = 12$: With $r_{12} = (8,3)$ we get $h(r_{12}) = 1$. Nothing to do.

$n = 13$: With $r_{13} = (9,3)$ we get $h(r_{13}) = 2$. Nothing to do.

$n = 14$: With $r_{14} = (9,4)$ we get $h(r_{14}) = -1$. Case (iii.A) defines $q_N := (9.4)$, $p_P := (7,2)$, $(a,b) := (8,3)$, and $c = 5$.

$n = 15$: With $r_{15} = (10,4)$ we get $h(r_{15}) = 3$. Nothing to do.

$n = 16$: With $r_{16} = (11,4)$ we get $h(r_{16}) = 6$. Nothing to do.

$n = 17$: With $r_{17} = (12,4)$ we get $h(r_{17}) = 9$. Nothing to do.

$n = 18$: With $r_{18} = (13,4)$ we get $h(r_{18}) = 12 > |a| + |b|$. Step not possible; start a new 4-DSS at $r_1 7$.

Coursework

Related material in textbook: Section 9.6.4.

A.16. [7 marks][a] Discuss algorithm **K1990**.

(i) Assume equal in inequalities (1) and (2) on page 6. Show that the main diagonal distance between these two straight lines is less or equal to $\sqrt{2}$, for any three integers a, b, and c.

(ii) Implement the algorithm (Note: there are free downloads on the Internet; see, for example, TC18 of the IAPR).

(iii) Provide general arguments (i.e., a proof) that this is a linear time algorithm and specify this further by providing a run-time analysis for 4-DSS*s* of varying length (see also Exercise 10 on page 337).

(iv) This algorithm provides a polygonal approximation of a traced border cycle. Calculate the total length of this polygonal curve for estimating the perimeter of digitized objects (e.g., a disk, an ellipse, or a "halfmoon"). Discuss the accuracy of this method for estimating the length of these digitized curves (i.e., frontiers of these objects).

[a] You can do either **A.16** or **A.17**, but not both.