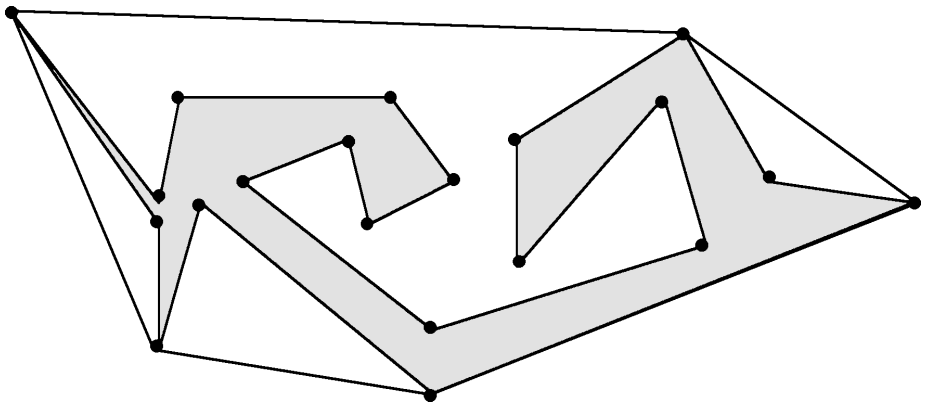
Convex Hull

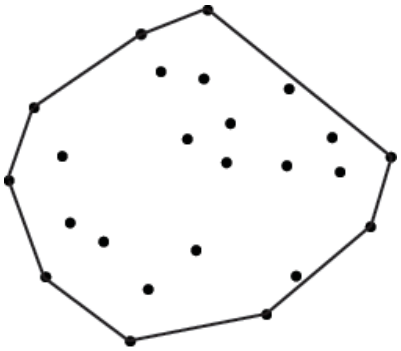
A set S is called *convex* if for any two points p, q of S the straight line segment pq is contained in S.

The *convex hull* $C(S)$ is the intersection of all of the halfspaces of $\mathbb{R}^n$ that contain S; it is the smallest convex set that contains S.

A convex *polygon* (*polyhedron*) is a nonempty bounded set that is an intersection of finitely many half-planes (half-spaces).



A simple polygon (shaded, having 20 vertices) and its convex hull which is a simple polygon with 5 vertives.

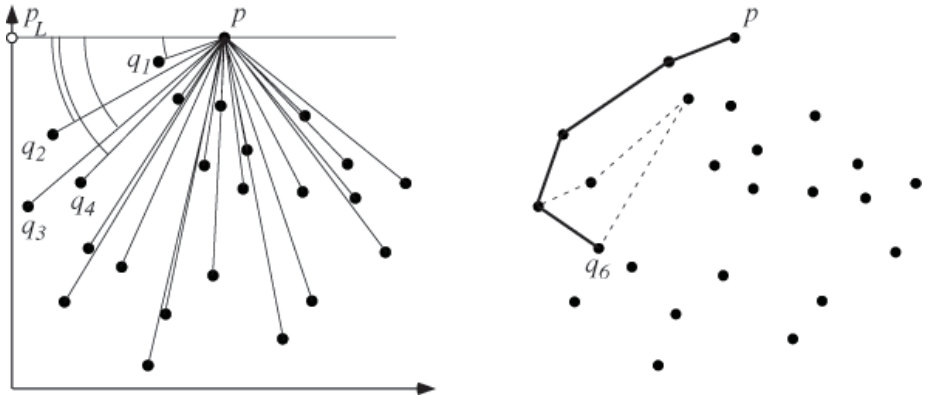


Convex hull of a finite set of points. – The calculation of convex hulls is a basic procedure in geometry-related picture analysis.

Graham's Scan

R.L Graham (1972): $\mathcal{O}(n \log n)$ for n points in $\mathbb{R}^2$

1. Start at a point of S (called the *pivot* p) that is known to be on the convex hull.
2. Sort the remaining points p_i of S in order of increasing angles η_i; if the angle is the same for more than one point, keep only the point furthest from p. Let the resulting sorted sequence of points be $q_1, \ldots, q_m$.
3. Initialize $C(S)$ by the edge between p and q_1.
4. Scan through the sorted sequence. At each left turn, add a new edge to $C(S)$; skip the point if there is no turn (a collinear situation); backtrack at each right turn.

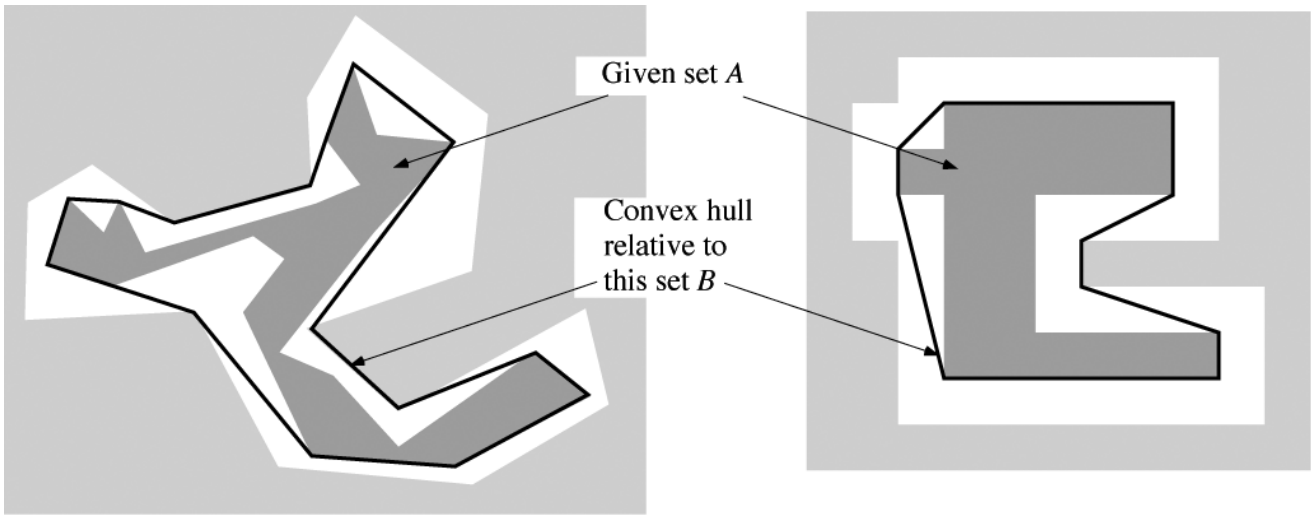


Left: angles η_i for the vectors defined by q_1, q_2, q_5, q_4. Right: backtrack situation at q_6; the dashed edges are removed or not added in Step 4 of the algorithm.

Relative Convex Hull

J. Sklansky, R.L. Chazin, and B.J. Hansen in 1972

Definition 1 *Let $S \subseteq B \subset \mathbb{R}^2$. S is called* B-convex *iff, for all $p, q \in S$, if the straight line segment pq is in B, it is also in S. The B-convex hull of S is the intersection of all B-convex sets that contain S.*



Relative convex hulls. Left: A, B are simple polygons. Right: A, B are isothetic simple polygons.

If A, B are simple polygons and A is contained in B, it can be shown that the frontier of the B-convex hull of A is the (uniquely determined) *minimum-length polygonal curve* (short: MLP) that is contained in B and that circumscribes A.

Relative convex hulls are often used in robotics, computational geometry and picture analysis (digital geometry).

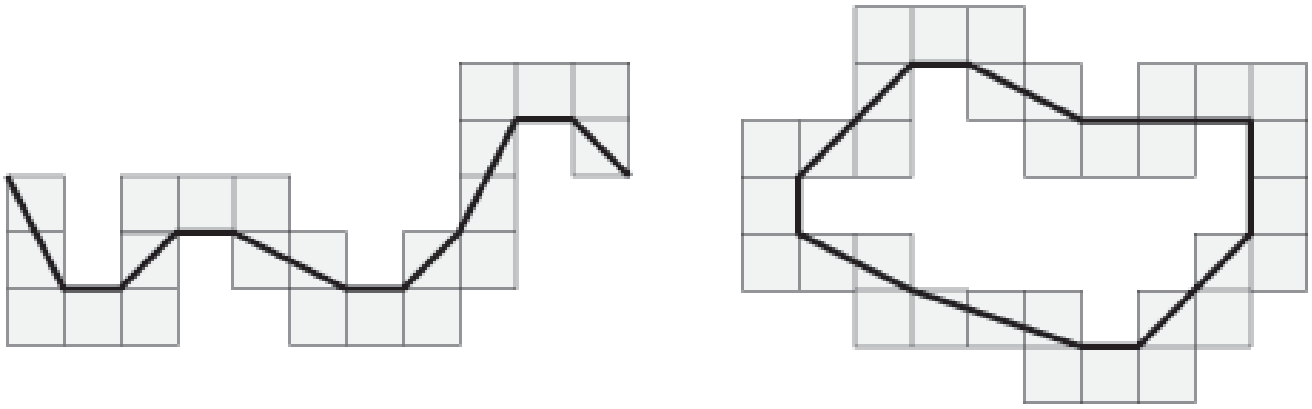
Length in the 2D Cell Model

intrinsic distance between two points in a simple polygon = the length of a shortest arc that connects the points and is contained in the polygon

it follows: this is a polygonal arc

intrinsic diameter of a polygon = the maximum intrinsic distance between any two of its points

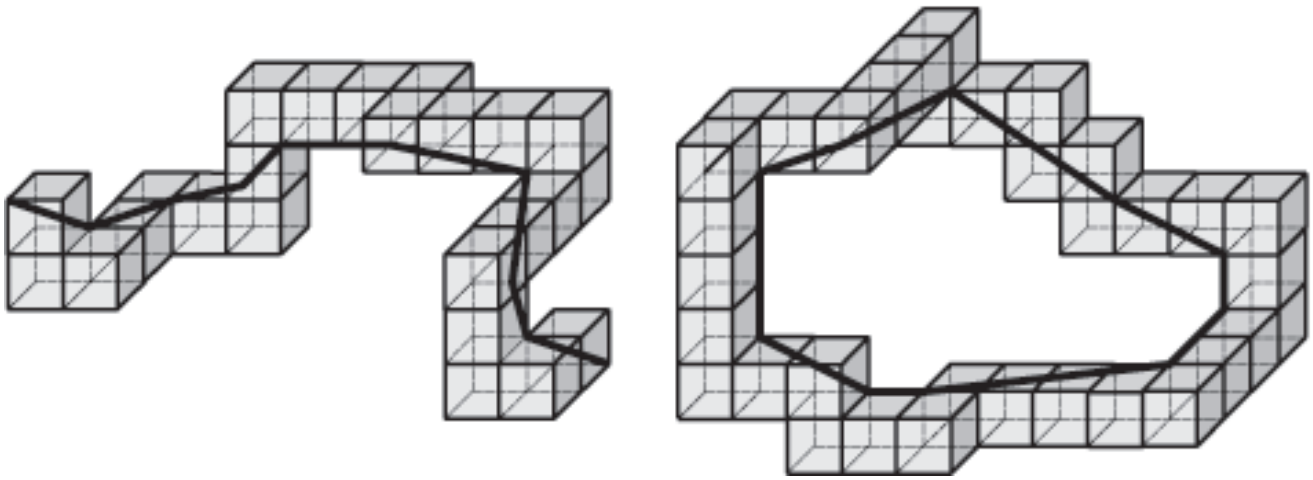
it follows: is between two vertices of the polygon



Left: the *length of a simple 1-arc* (in the 2D grid cell model) can be defined by the length of its intrinsic diameter

Right: the *length of a simple 1-curve* can be defined by the length of that MLP contained in the curve (to be precise: in the union of all 2-cells of this curve), and circumscribing its "inner frontier"

Length in the 3D Cell Model



Left: the *length of a simple 2-arc* (in the 3D grid cell model) can be defined by the length of its intrinsic diameter

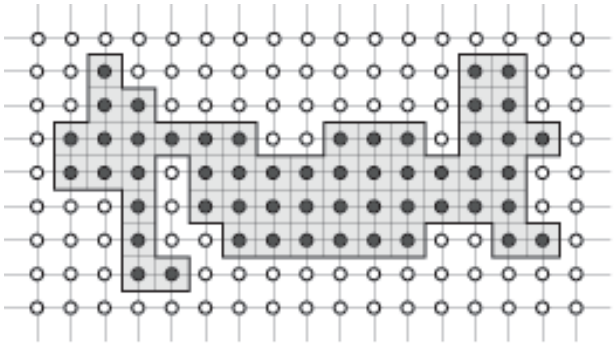
Right: the *length of a simple 2-curve* can be defined by the length of that MLP that is uniquely identified by being contained in the curve and not contractible into a single point within this curve.

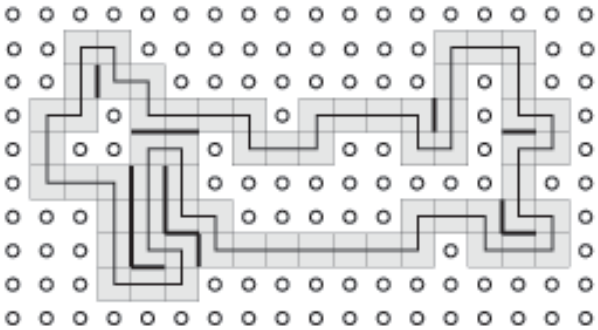
Difference between 2D and 3D

In the 2D case, the MLP coincides with the relative convex hull of the "inner frontier" with respect to the "outer frontier" (and the design of a 2D MLP algorithm is not very hard). In the 3D case, there is no such analogy with a relative convex hull (and the calculation of a 3D MLP within a simple 2-curve is a difficult task).

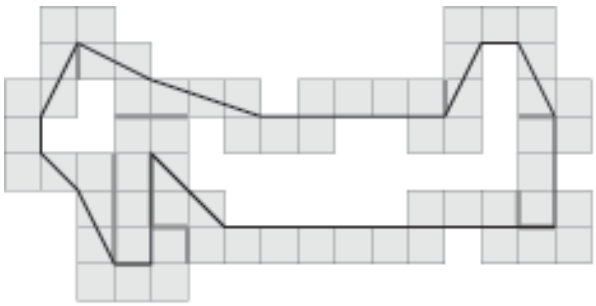
1-Curve around a Digital Region

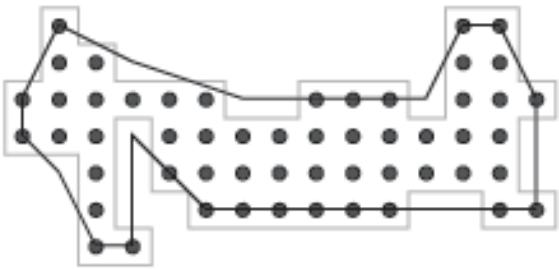
picture grid on the left: a 4-region and its frontier (bold curve)





frontier grid on the right: the original frontier runs through pixels in this grid, and we consider these as being 2-cells; the original frontier defines this way a 1-curve, which is not simple in general, but can be used as input for an MLP algorithm assuming that the successor of a 2-cell in this 1-curve defines the only possible grid edge where the MLP can cross into a next 2-cell (the figure on the right shows in bold grid edges which are not allowed to cross)

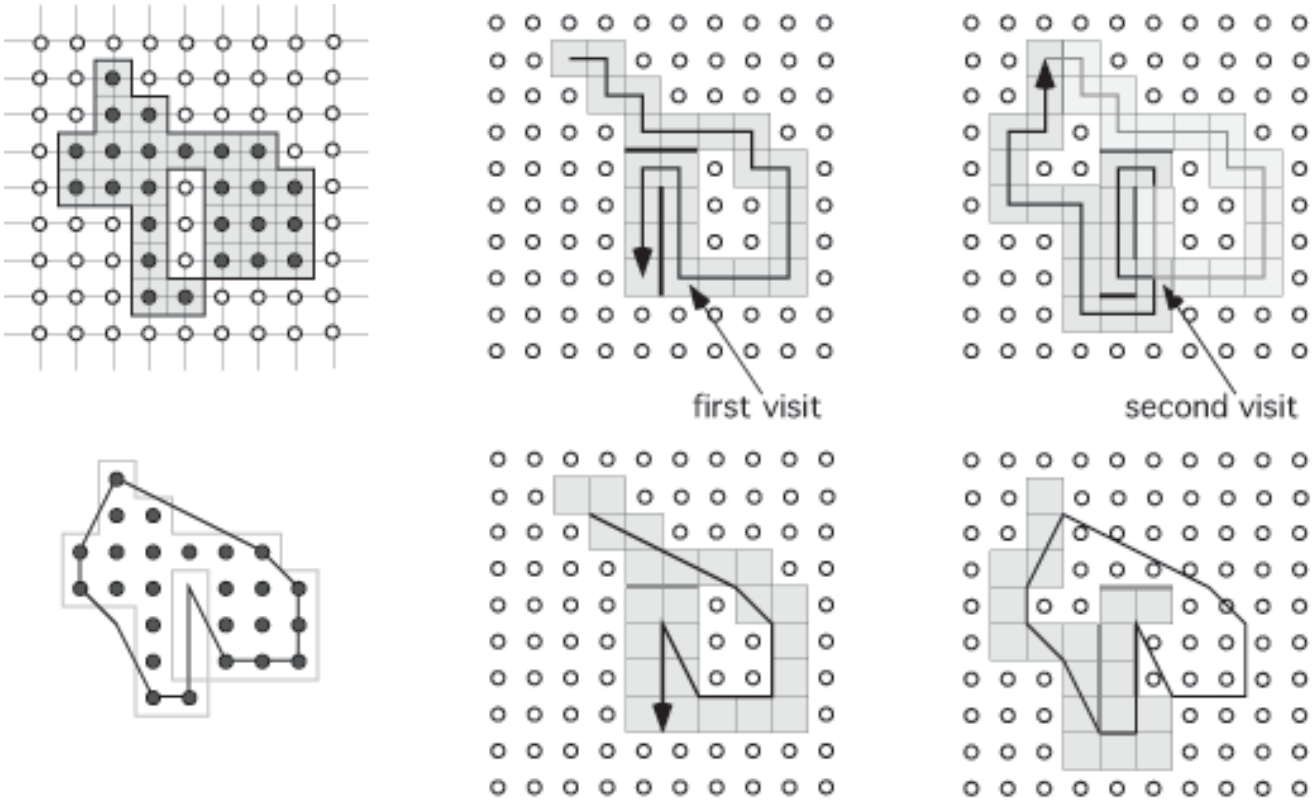




resulting MLP, approximating the frontier of the given 4-region

Self-Intersections

we illustrate possible self-intersections of the frontier of a given simply-connected 4-region



Accordingly, the 1-curve of grid cells in the frontier grid, following a trace around the frontier, will also have self-intersections. The MLP-algorithm needs to handle these cases as well. The figure illustrates (upper row) the repeated visit of one 2-cell, and (lower row) the construction of the MLP in this case. The final result is shown at the bottom on the left.

Inner and Outer Frontier

Following the given 1-curve in counter-clockwise orientation (in a righthand coordinate system), the sequence of 1-cells on the right form a 0-curve γ_1 of 1-cells (the "outer frontier" in case of a simple 1-curve), and those on the left a 0-curve γ_2 of 1-cells (the "inner frontier").

vertex v_i on γ_1 or γ_2 is called a *convex vertex* if the frontier makes a positive turn at v_i, detectable by

$D(v_{i-1}, v_i, v_{i+1}) > 0$, where D is the determinant

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = x_1y_2 + x_3y_1 + x_2y_3 - x_3y_2 - x_2y_1 - x_1y_3$$

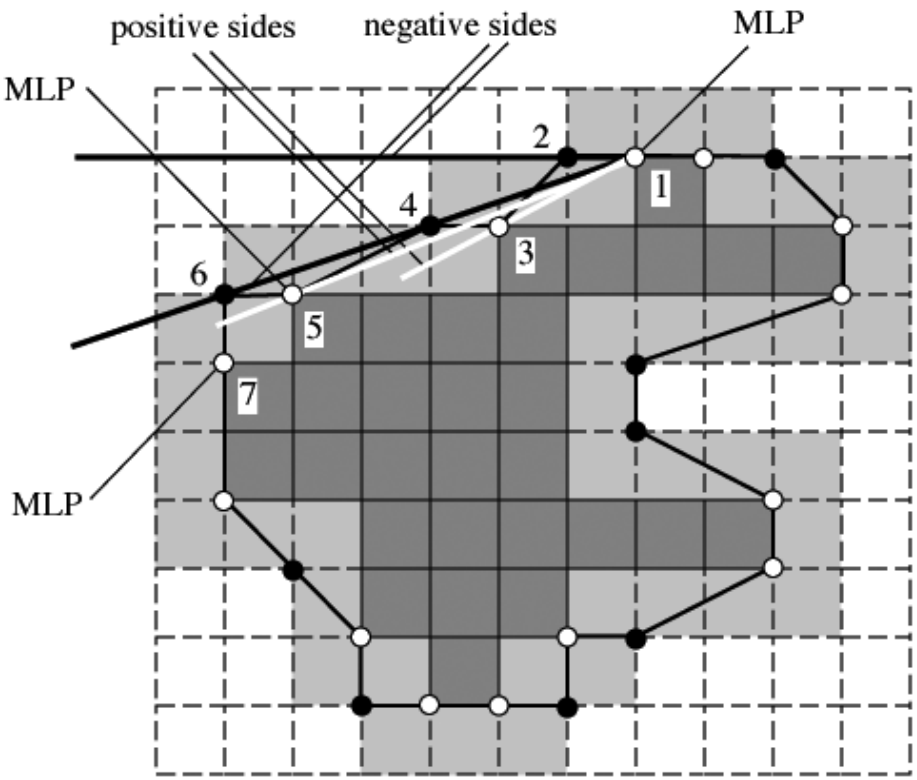
v_i is called a *concave vertex* if the frontier makes a negative turn ($D(v_{i-1}, v_i, v_{i+1}) < 0$); a *collinear vertex* if $D(v_{i-1}, v_i, v_{i+1}) = 0$

Our algorithm traces γ_1 (or γ_2), detects convex and concave vertices, puts their coordinates into a list L, and marks them as convex or concave.

For simplicity assume that the coordinates are integers; the coordinates of two successive vertices with indices i and $i+1$ satisfy $|x_{i+1} - x_i| + |y_{i+1} - y_i| = 1$.

Only convex vertices of the inner curve γ_2 and only concave vertices of the outer curve γ_1 can be vertices of the MLP.

There exists a mapping from the set of all concave vertices of γ_2 onto the set of all concave vertices of γ_1 such that each concave vertex of γ_1 corresponds to at least one concave vertex of γ_2.



Numbers in the figure denote successive vertices in the list L:

1 is a start vertex (i.e., an already known MLP vertex); 3 and 5 are successive convex vertices on γ_2; 2, 4, and 6 are successive concave vertices on γ_1. Vertex 7 is not between the negative (black line) and positive (white line) sides of sector (6,1,5); therefore 5 is the next MLP vertex and is a new start vertex.

Start: put all of the vertices of γ_2 into L, then replace each concave vertex of γ_2 in L with its corresponding concave vertex of γ_1 by modifying its coordinates by ± 1, where the sign depends on the orientations of the incident edges.

L = all (plus others) of the vertices that will form the MLP

MLP calculation and calculation of its length $\mathcal{L}$

1. Initialize list $L = (v_1, \ldots, v_n)$ as described above; it contains all of the vertices on γ_2 except the concave vertices, which are replaced by concave vertices on γ_1. Each v_i in L is labeled by the sign of $D(v_{i-1}, v_i, v_{i+1})$.
2. Let $k := 1$, $a := 1$, $b := 1$ and $i := 2$. Let $\mathcal{L} := 0$ and $p_1 := v_1$.
   //v_1 is the first MLP vertex//
3. If $i > n + 1$, stop.
4. If $i \leq n$, then $j := i$; else $j := 1$. //go back to v_1//
5. If $D(p_k, v_b, v_j) > 0$, then //v_j lies on the positive side//
   $\{k := k + 1,\ p_k := v_b,\ i := b,\ a := b,$ and $\mathcal{L} := \mathcal{L} + d_e(p_{k-1}, p_k)\}$;
   else
   (a) If $D(p_k, v_a, v_j) \geq 0$, then //v_j is in the sector//;
   if v_j has a positive label, then $b := j$, else $a := j$;
   else //v_j lies on the negative side//
   (b) $\{k := k + 1,\ p_k := v_a,\ i := a,\ b := a,$ and $\mathcal{L} := \mathcal{L} + d_e(p_{k-1}, p_k)\}$
6. Let $i := i + 1$ and go to Step 3.

The algorithm has linear time complexity.

Suppose we already know that vertex v_i of L is an MLP vertex. Then v_j ($j > i$) can be an MLP vertex only if all convex vertices v_k^+ such that $i < k < j$ lie on the positive side of (v_i, v_j) or are collinear with it (i.e., $D(v_i, v_k^+, v_j) \geq 0$).

Similarly, all concave vertices v_l^- such that $i < l < j$ must lie on the negative side of (v_i, v_j) or be collinear with it.

Suppose a convex vertex v^+ and a concave vertex v^- both satisfy these conditions. When we consider a vertex v as a candidate MLP vertex, the following situations can occur:

1. v lies on the positive side of (v_i, v^+) (i.e., $D(v_i, v^+, v) \geq 0$);
2. v lies on the negative side of (v_i, v^+) or is collinear with it and also lies on the positive side of (v_i, v^-) or is collinear with it; or
3. v lies on the negative side of (v_i, v^-).

In case (1), v^+ becomes the next MLP vertex.

In case (2), v becomes a candidate for the MLP and must replace either v^+ or v^- depending on the sign of v.

In case (3), v^- becomes the next MLP vertex. This is also correct in the trivial case where v^+, v^-, or both coincide with v_i.

Start: a vertex v_1 that is known to be an MLP vertex (e.g., the uppermost-leftmost vertex of γ_2); set v^+ and v^- equal to v_1; and then test all subsequent vertices as just described.

Whenever the next MLP vertex is detected, it becomes a new start vertex.

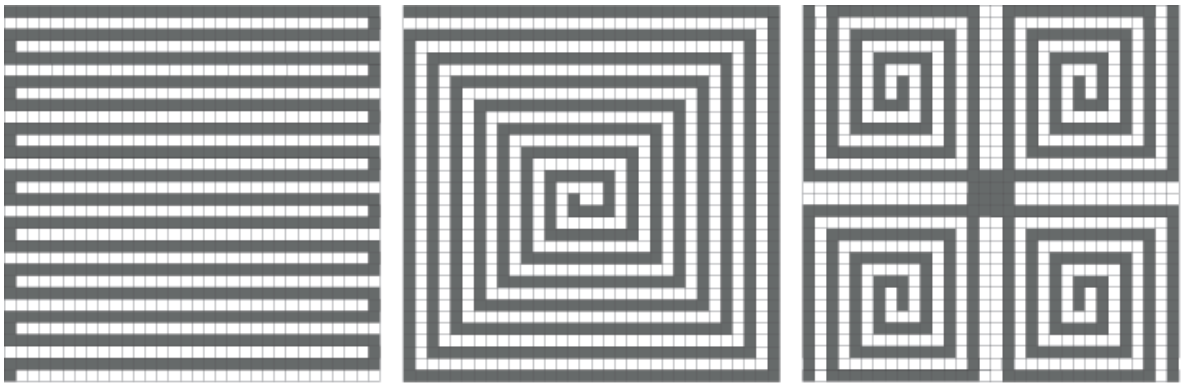
Coursework

Related material in textbook: Sections 1.2.9, 10.1.4, 10.2.3, and 10.2.4.

A.18. [7 marks] Implement the given MLP algorithm for estimating the perimeter of a simply-connected 4-region by the length of the MLP. (The calculated length $\mathcal{L}$ defines the length estimator E_{mlp}.) Note:
http://www.citr.auckland.ac.nz/dgt/Source_Code.php?id=4
offers a free download of an MLP source.

Allow as input simply-connected 4-regions of arbitrary shape and size in a 256×256 binary picture.

(i) Test your algorithm by considering "extreme input cases", such as shown below.



(ii) Generate a diagram which shows run-times of your algorithm in dependence upon the length of the frontier (note: this length is equal to the number of 1-cells on this curve) of the given 4-region.

(iii) Optionally, **[1 mark]** also provide a way (e.g., in a grid of higher resolution) to visualize the calculated MLP.