Part 2 : Digital Geometry for Image Analysis

Anne Vialard

LaBRI, Université de Bordeaux

Contents

Discrete distance

Definition

A *discrete distance* is a function $d : \mathbb{Z}^n \times \mathbb{Z}^n \to \mathbb{N}$ verifying $\forall P, Q, R \in \mathbb{Z}^n$

- $d(P, Q) \geq 0$ [positive]
- $d(P, Q) = 0 \Leftrightarrow P = Q$ [definite]
- $d(P, Q) = d(Q, P)$ [symmetry]
- $d(P, Q) \leq d(P, R) + d(R, Q)$ [triangle inequality]

Two discrete distances :

- $d_1(P, Q) = \sum_{i=0}^{n-1} |P_i - Q_i|$
- $d_\infty(P, Q) = \max_i\{|P_i - Q_i|\}$

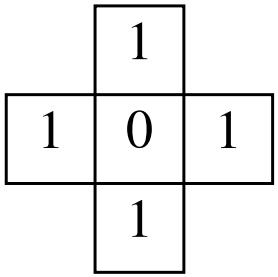
In 2D : d_4 and d_8
In 3D : d_6 and d_{26}
very different from euclidean distance...

Chamfer distances

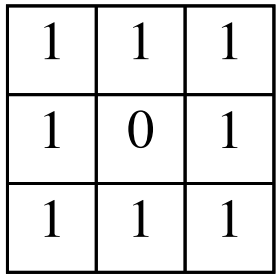
- An integer value is associated with each elementary move in a given neighborhood. The elementary moves and their values (cost) are defined by a mask.
- The distance between two points is the cost of the minimum cost path linking the two points and composed of available elementary moves.

Example : d_4 vertical and horizontal moves of cost 1

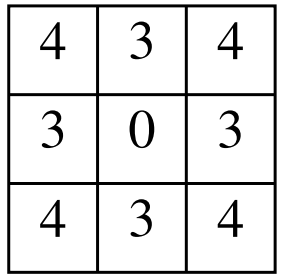
2D chamfer masks



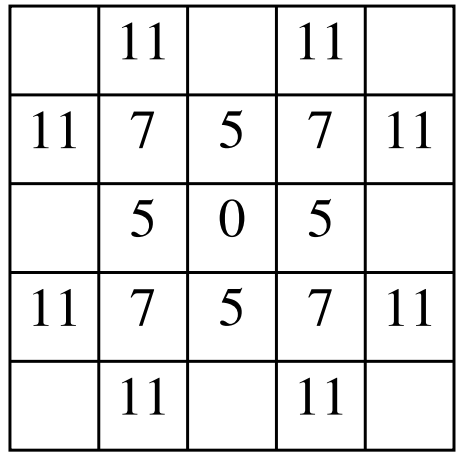
d_4



d_8



$d_{3,4}$



$d_{5,7,11}$

3D chamfer masks

	f		f	
f	e	d	e	f
	d		d	
f	e	d	e	f
	f		f	

z=-2 / z=2

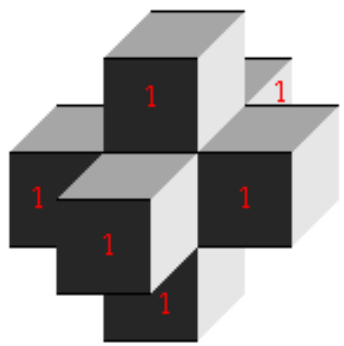
f	e	d	e	f
e	c	b	c	e
d	b	a	b	d
e	c	b	c	e
f	e	d	e	f

z=-1 / z=1

	d		d	
d	b	a	b	d
	a	0	a	
d	b	a	b	d
	d		d	

z=0

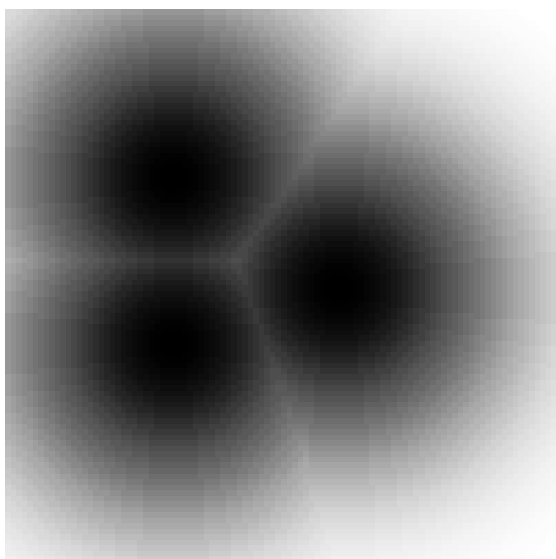
	a	b	c	d	e	f
d_1	1					
d_{18}	1	1				
quasi-euclidean 3×3	1	$\sqrt{2}$				
optimal 3×3	0.92644	1.34065	1.65849			
quasi-euclidean 5×5	1	$\sqrt{2}$	$\sqrt{3}$	$\sqrt{5}$	$\sqrt{6}$	3

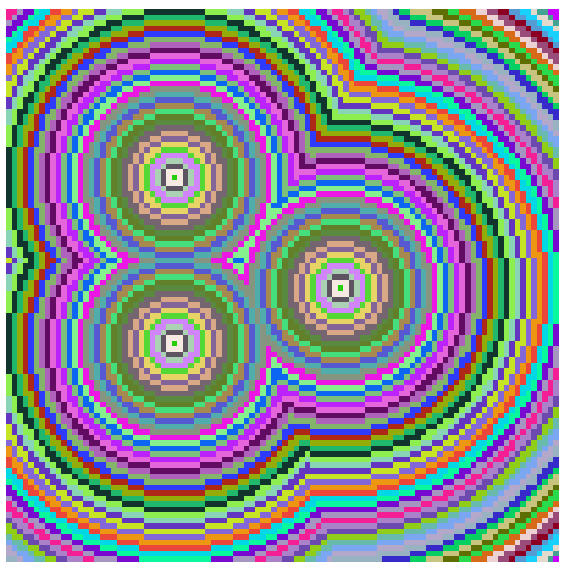


Distance map : definition

Let $R \subset \mathbb{Z}^n$ be a set of points called reference points. The corresponding *distance map* is obtained by associating with each point its distance to the closest reference point.

$$\begin{cases} \mathbb{Z}^n \to \mathbb{N} \\ P \to min(d(P, P_r), P_r \in R) \end{cases}$$





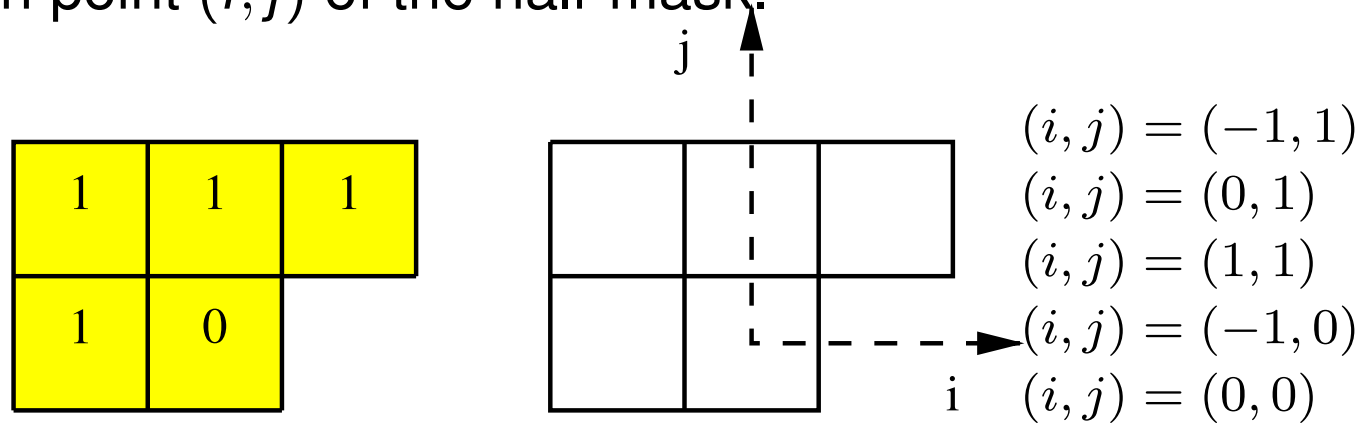
Sequential computation with half-masks I

2D chamfer distances

Principle : spread of information

- Initialization : 0 for reference points, ∞ elsewhere
- First scan of the image from left to right and from top to bottom by using the first half mask.

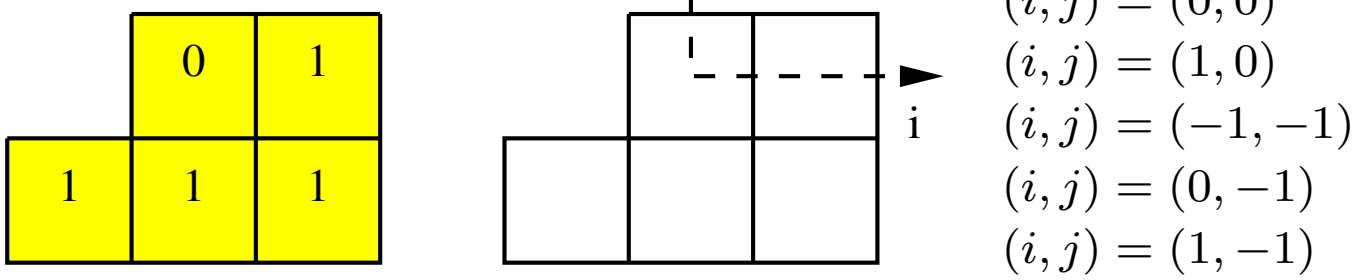
  Computation : $DM(x, y) = min(DM(x + i, y + j) + m_{i,j})$ for each point (i, j) of the half-mask.



Sequential computation with half-masks II

2D chamfer distances

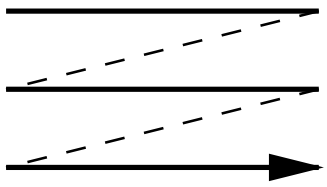
- Second scan of the image from right to left and from bottom to top
  same computation with the second half mask.



Computation example (d_8)

Balayage avant

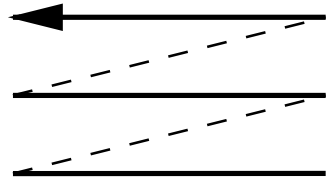
1	1	1
1	0	



∞	∞	∞	∞	∞		1	2
∞	∞	∞	∞	1	1	1	2
∞	∞	∞	2	2	2	2	2
∞		1	2	3	3	3	3
1	1	1	2	3	4	4	4
2	2	2	2	3	4	5	5

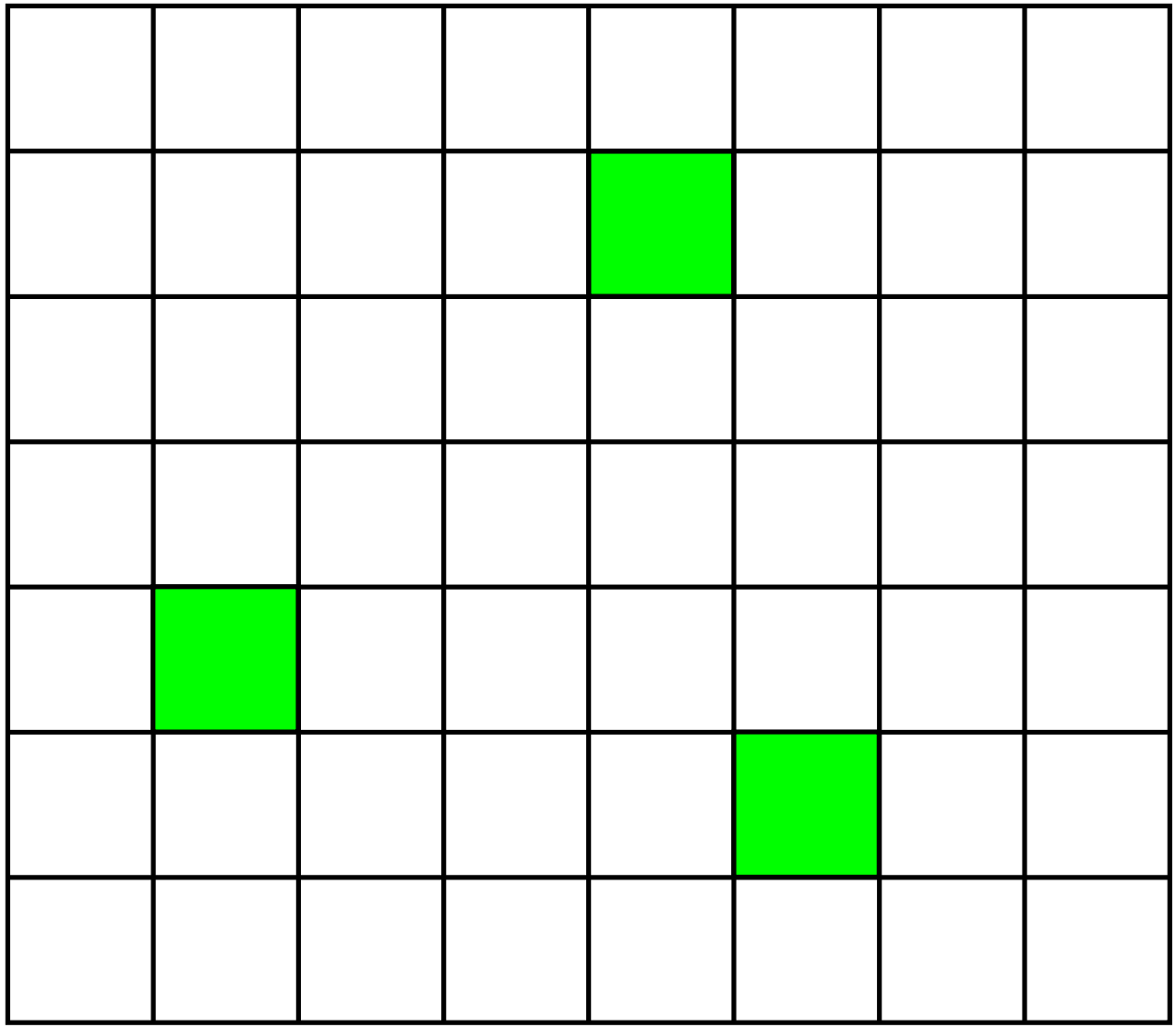
Balayage arrière

	0	1
1	1	1



3	3	3	2	1		1	2
2	2	2	2	1	1	1	2
1	1	1	2	2	2	2	2
1		1	2	3	3	3	3
1	1	1	2	3	4	4	4
2	2	2	2	3	4	5	5

To fill in



Sequential computation with half-masks

3D chamfer distances

```
// Forward pass
for (z=0; z < Sz; z++)
   for (y=0; y < Sy; y++)
     for (x=0; x < Sx; x++)
       DM(x, y, z) = min∀i,j,k∈fp(DM(x + i, y + j, z + k) +
m[i, j, k])

// Backward pass
for (z = Sz − 1; z ≥ 0; z−)
   for (y= Sy − 1; y ≥ 0; y−)
     for (x= Sx − 1; x ≥ 0; x−)
       DM(x, y, z) = min∀i,j,k∈bp(DM(x + i, y + j, z + k) +
m[i, j, k])
```

SED : Danielson Algorithm (2D) I

Question : How to efficiently compute an euclidean distance map ?

- Initialization : 3 integer values dx, dy and $d2$ are associated with each pixel. The vector (dx, dy) gives the location of the current point relatively to the closest reference point known at a given moment. The squared distance $d2 = dx^2 + dy^2$ can also be stored to avoid recomputation.
- Top to bottom pass : for each line
  - forward pass : comparison with upper point / left point
  - backward pass : comparison with right point
- Bottom to top pass : for each line
  - backward pass : comparison with lower point / right point
  - forward pass : comparison with left point

Problem : the result may be wrong (in rare cases)

SED : Danielson Algorithm (2D) II

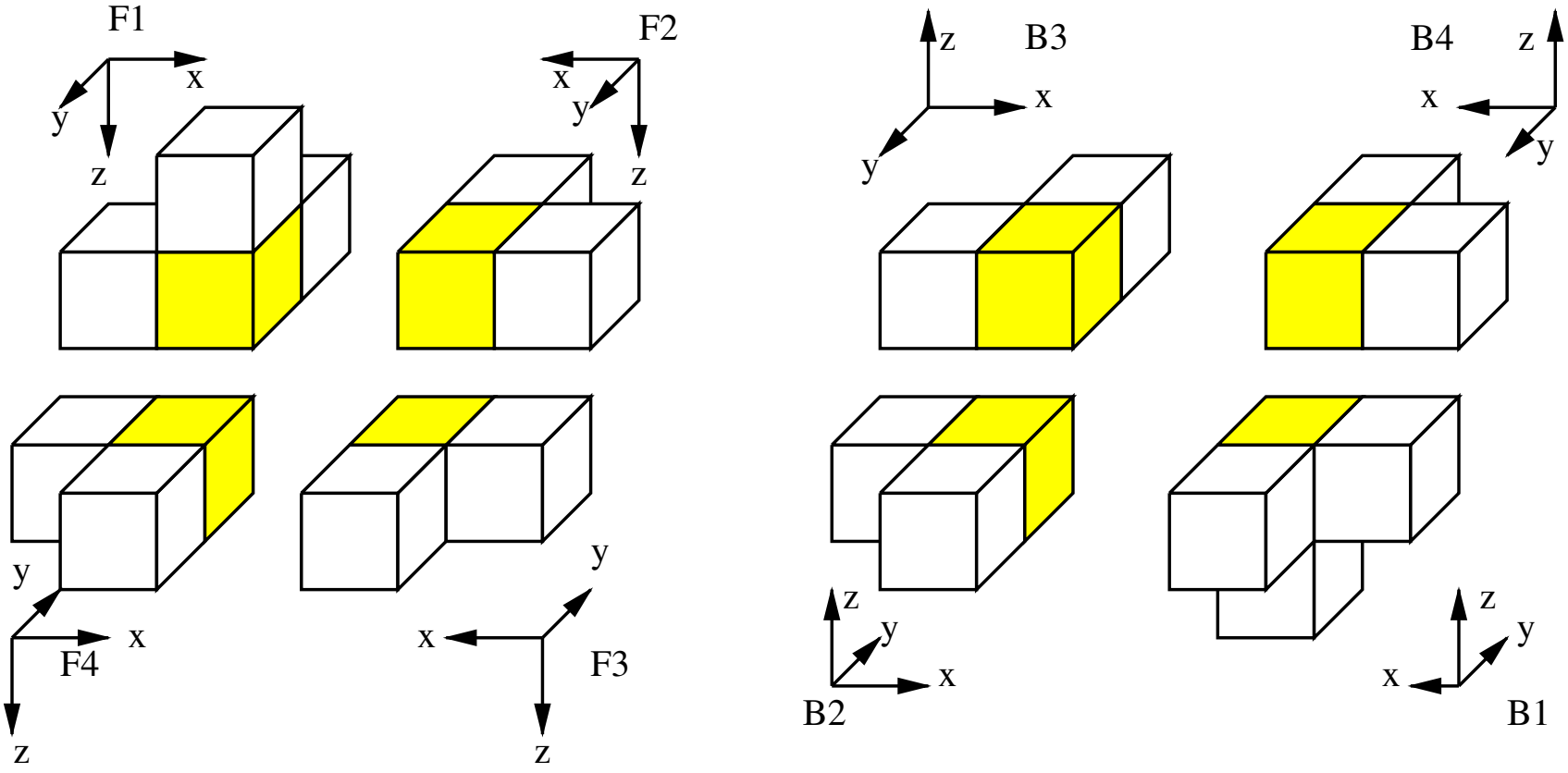
```
void Update (int x, int y, int deltaX, int deltaY)
 int d2 = DM[x+deltaX][y+deltaY].d2
        + 2*deltaX*DM[x+deltaX][y+deltaY].dx
        + 2*deltaY*DM[x+deltaX][y+deltaY].dy + 1;

 if (d2 < DM[x][y].d2)
   DM[x][y].dx = DM[x+deltaX][y+deltaY].dx + deltaX;

   DM[x][y].dy = DM[x+deltaX][y+deltaY].dy + deltaY;

   DM[x][y].d2 = d2;
```

SED : 3D Algorithm



SED : 3D Algorithm

Detail of a forward sub-step

```
for (z=0; z < Sz ; z++)
// Forward pass F1
   for (y=0; y < Sy ; y++)
     for (x=0; x < Sx ; x++)
       p=(x, y, z)
       pos = argmin_i ||vec[p + dir_i] + dir_i||
       vec[p] = vec[p + dir_pos] + dir_pos
       DM(x, y, z) = ||vec[p]||
// Forward pass F2
...
// Forward pass F3
...
// Forward pass F4
...
```

2D exact euclidean distance I

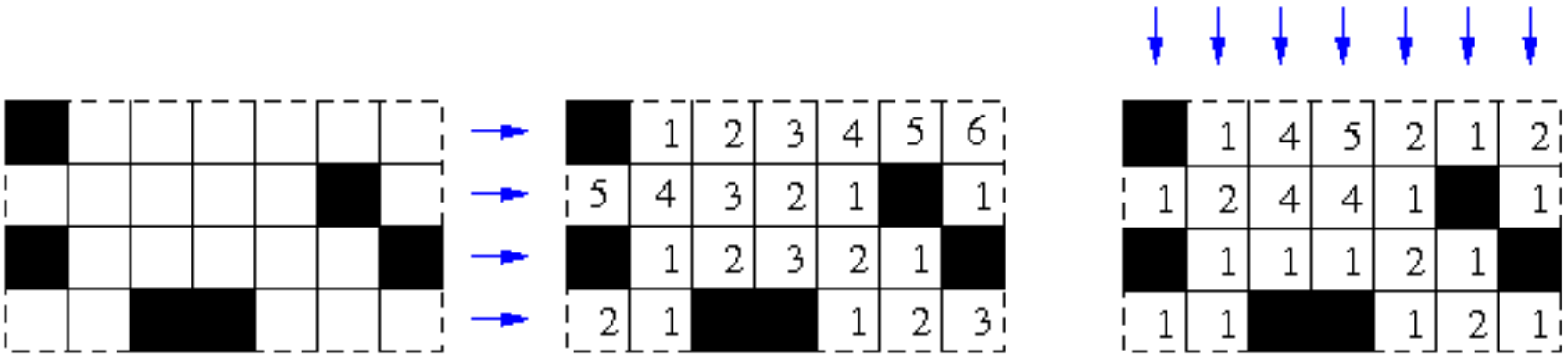
$DM(i,j) = min\{(i-x)^2 + (j-y)^2 : 0 \leq x < W, 0 \leq y < H, (x,y)$ reference point $\}$

1. Line processing : for a line j

$$L(i,j) = min_x\{|i-x| : 0 \leq x < W, (x,j) \text{ reference point } \}$$

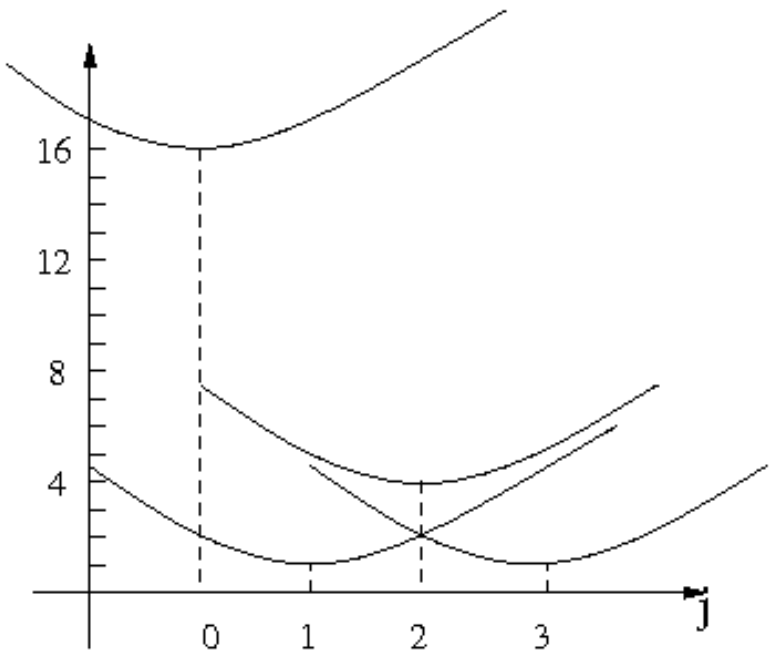
2. Column processing

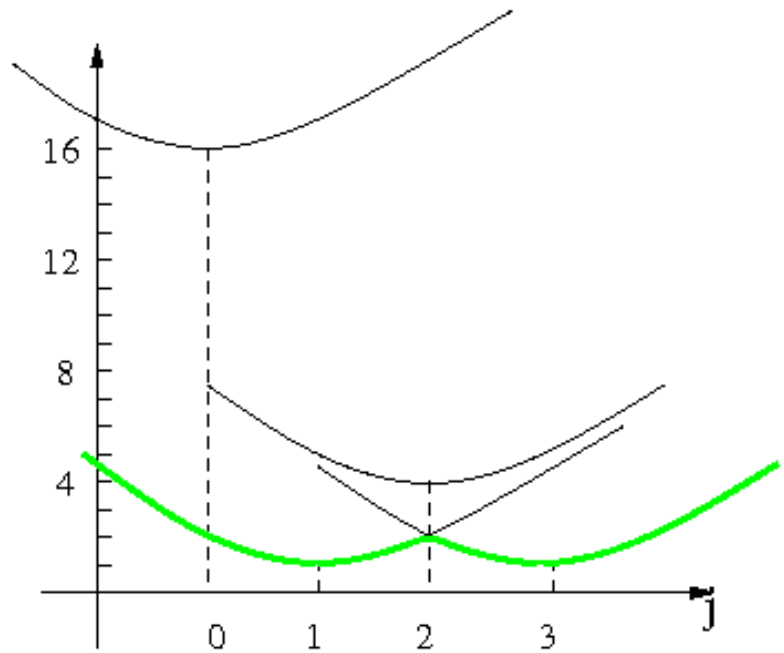
$$DM(i,j) = min_y\{L(i,y)^2 + (j-y)^2 : 0 \leq y < H\}$$



2D exact euclidean distance II

The second part of the algorithm can be linearized by computing the inferior envelop of the parabolas $\mathcal{P}_y^i(j) = L(i,y)^2 + (j-y)^2$





Geodesic distance

Domain : connected region R

Let $P \in R$ be the origin point, any other point $Q \in R$ is labeled with its *geodesic distance* to P, length of shortest path linking P to Q without leaving R.

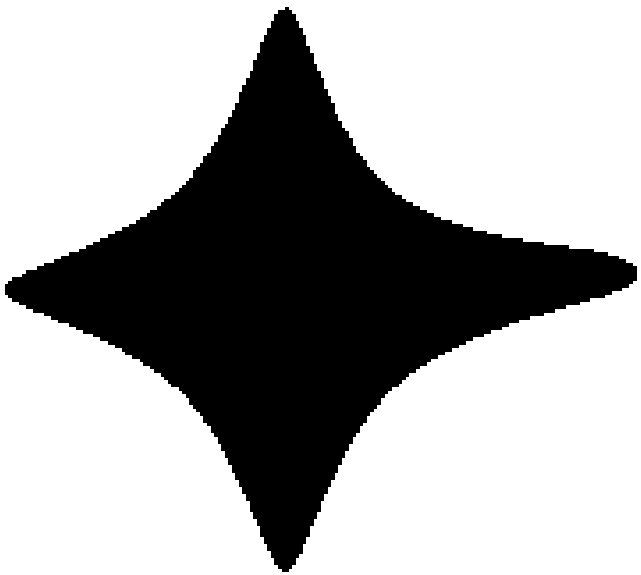
Algorithm with a chamfer mask w :

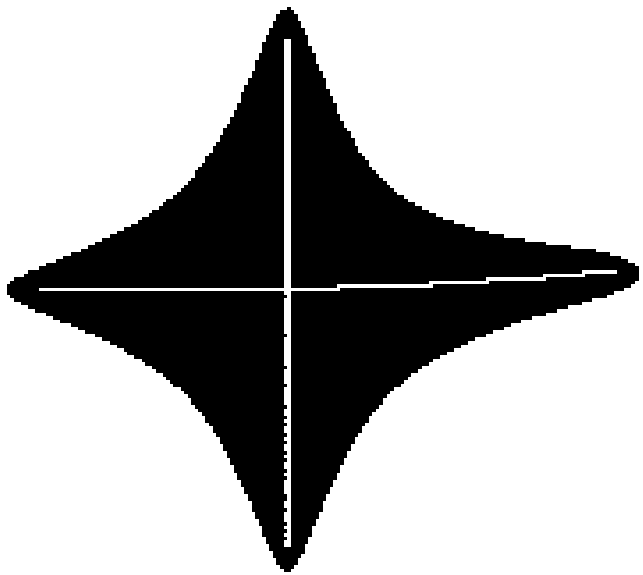
- Data structure : set of queues F_i, all the points in queue F_d are at distance d from point P.
- Initialization : Add P to F_0
- At each step :
  - Pull Q out of $F_{d_{min}}$ (non empty queue with minimum index)
  - For each neighbor N of Q, push N in the queue $F_{d_{min}+w(\vec{QN})}$

Contents

Skeletonization

Aim : shape encoding, simplified representation for shape analysing, recognition and matching (topology preservation, size)





2D binary skeleton

Definition (Chassery) :
The skeleton S_R of a 2D region R verifies :

1. $S_R \subset R$
2. S_R has the same number of connected components and the same number of holes as R
3. S_R is minimal for condition 2
4. The elongated parts of R give the curves of S_R
5. S_R is centered in R

Remark : a binary skeletonization is not reversible

Computation by thinning

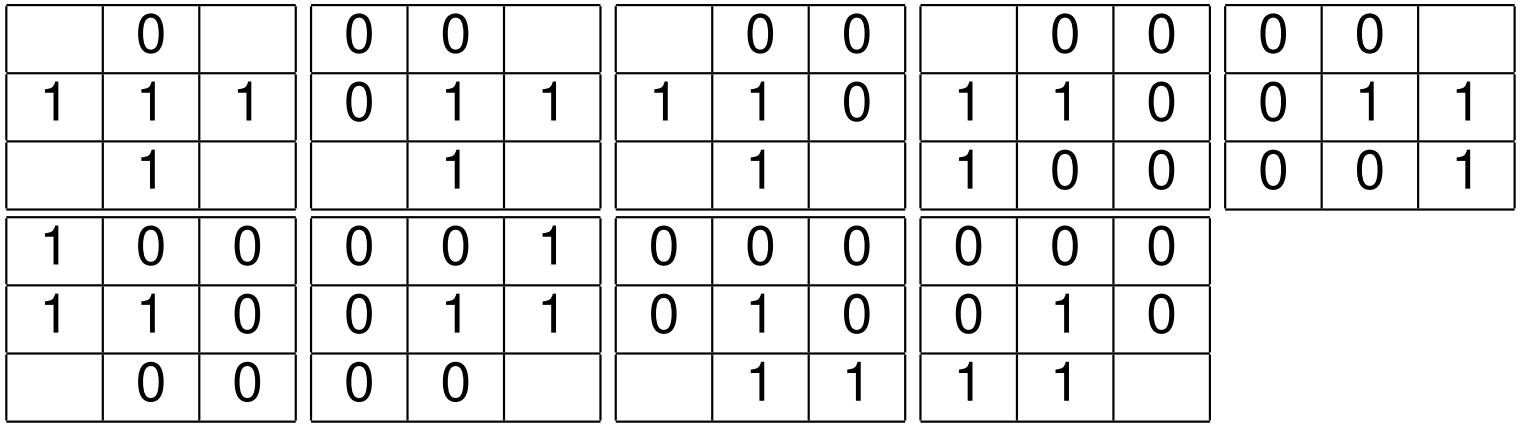
"Non-essential" points are removed

- P *simple point* of $R \Leftrightarrow$
  number of connected components of $R - \{P\} =$
  number of connected components of R
  and
  number of connected components of $\overline{R} \cup \{P\} =$
  number of connected components of $\overline{R}$
- P An *endpoint* of $R \Leftrightarrow P$ has only one neighbor in R

$\Rightarrow$ Idea : remove simple points that are not endpoints
The points to be considered belong to the border of the object.

Thome algorithm

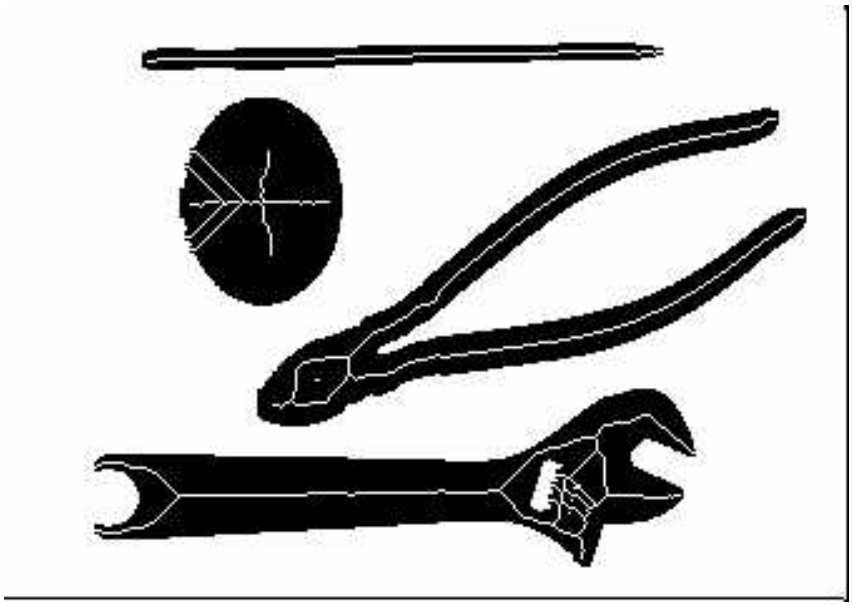
Configurations for a point located north of the shape :



Algorithm :

- Processing of the "north" points : all the points corresponding to the configurations are marked THEN all the marked points are removed
- Same processing for South, East and West
- Iterations until no point can be removed

Thome algorithm : example



Pruning algorithms

Simple point : local 2D characterization

Connectivity number (2D) : $T_k(P,0) = |C_k^P[N_8^*(P) \cap O]|$
where P discret point, O set of discrete points, $C_k^P(X)$ set of k-connected components of X k-adjacent to P.
$(k, \bar{k}) = (8,4)$ connectivities object/background

Point P is a *simple point* of the object O if and only if
$T_k(P,O) = 1$ and $T_{\bar{k}}(P,\bar{O}) = 1$

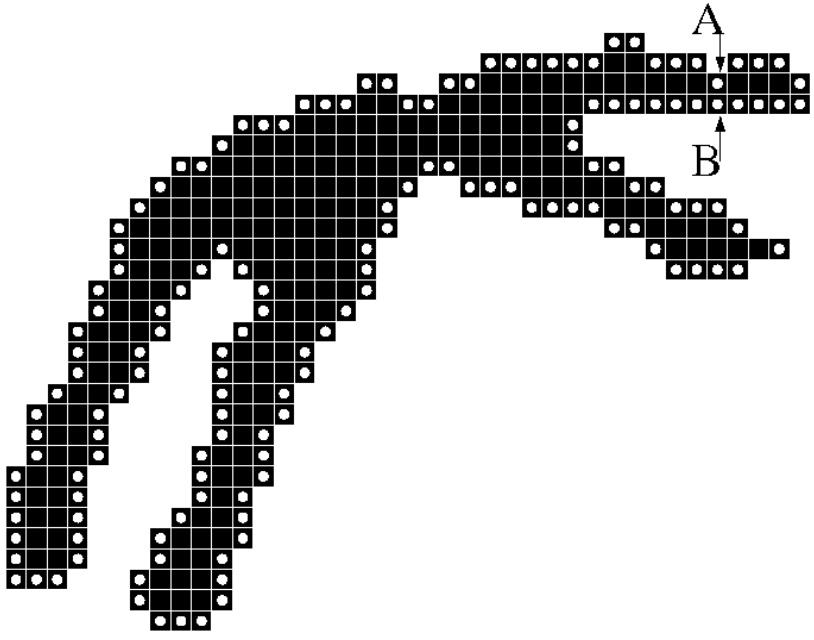
Other results :

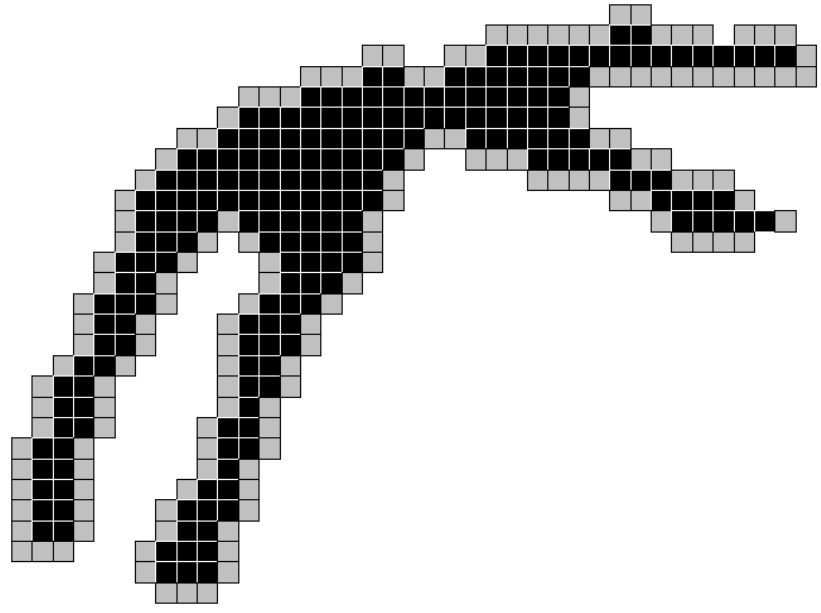
- $T_k(P,O) = 0 \Leftrightarrow P$ is an isolated point
- $T_{\bar{k}}(P,\bar{O}) = 0 \Leftrightarrow P$ is an interior point
- $T_{\bar{k}}(P,\bar{O}) \neq 0 \Leftrightarrow P$ is a boundary point

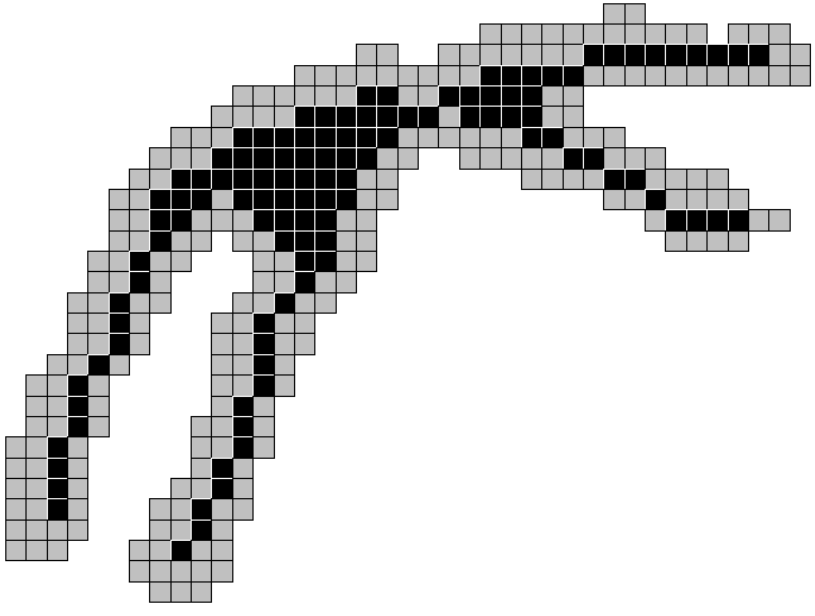
Thinning : first algorithm I

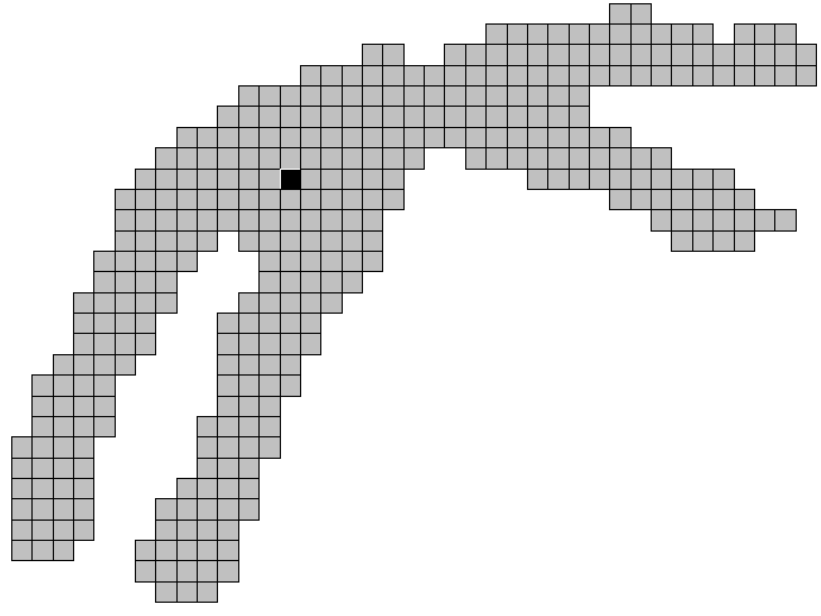
```
ES = set of simple points of O
Until ES is not empty
      E = empty set
      For each P in ES
           If P is a simple point of O
                Remove P from O
                For each Q in N(P)∩ O
                     Add Q to E
      ES = empty set
      For each Q in E
           If Q is simple for O
                Add Q to ES
```

Thinning : first algorithm II









Thinning : directionnal algorithm

```
ES = set of simple points of O
While ES is not empty
      E = empty set
      For t in [North, South, East, West]
           For each P in ES such that type(P)=t
                If P is a simple point of O and
                P is not an end point
                     Remove P from O
                     For each Q in N(P)∩ O
                          Add Q to E
      ES = empty set
      For each Q in E
           If Q is simple for O
                Add Q to ES
```

Thinning : algorithm based on a priority function

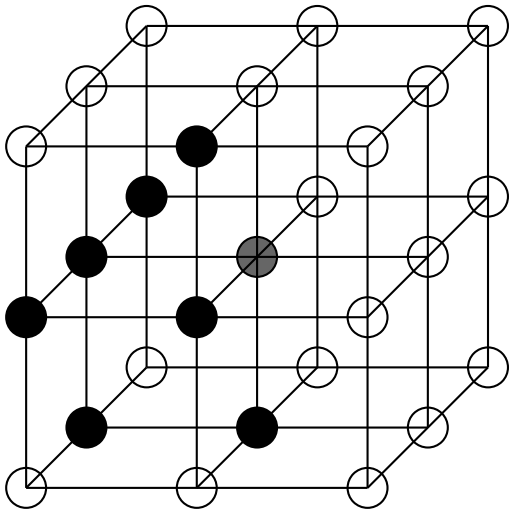
```
Repeat
      Remove a point P of O such that
      P is a simple point of O and that
      P(P) minimum
Until no more possible removal
```

Example of priority function : distance map to the background

Thinning : 3D extension I

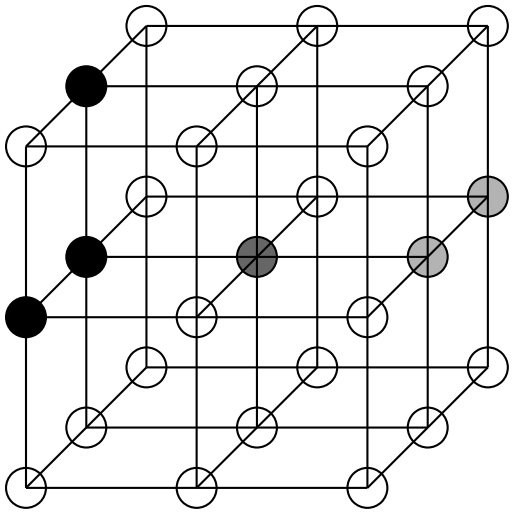
Preserve the number of connected components of the object and of its complement set + preserve the tunnels.

- *3D number of connectivity* : similar to 2D



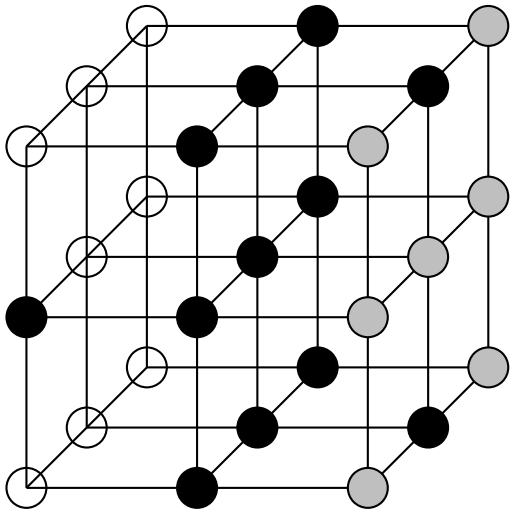
simple point

$T_6(P, O) = T_{26}(P, \bar{O}) = 1$



1D isthmus

$T_6(P, O) = 2$



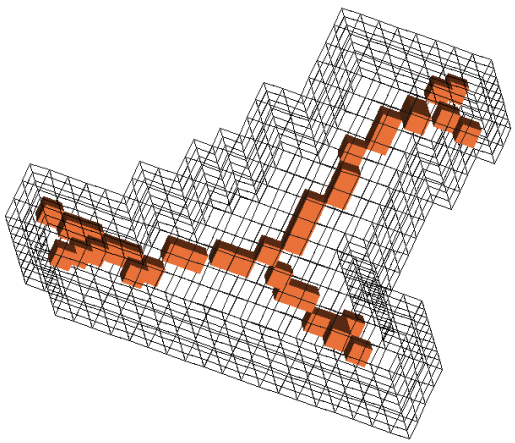
2D isthmus

$T_{26}(P, \bar{O}) = 2$

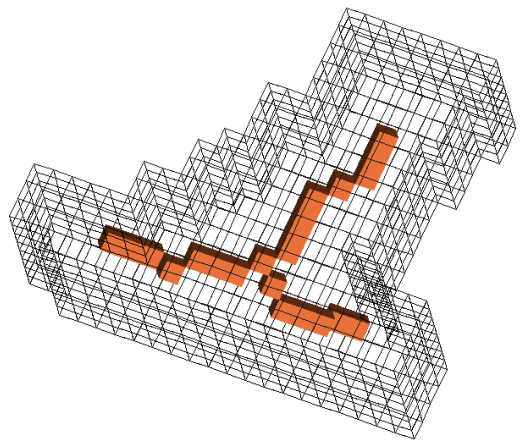
- *simple point* : similar to 2D

Thinning : 3D extension II

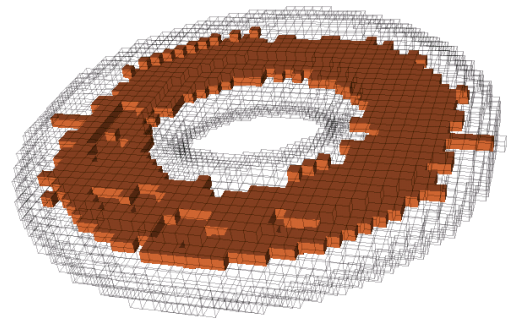
- *end point* : 2 more types
  1. extremity of a curve : adjacent to only one object point
  2. 1D isthmus
  3. 2D isthmus



(1)



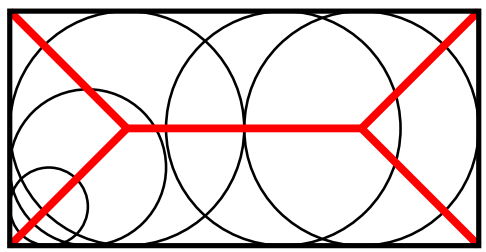
(2)



(3)

Skeletonization : medial axis

- *Discrete ball* of center P (integer coordinates) and of radius r (integer) for the distance d : $B(P, r) = \{Q \ / \ d(P, Q) \leq r\}$
- Let R be a discrete region and $P \in R$, $r_P = d(P, \overline{R}) - 1$
- The *medial axis* of region R is the set of maximal balls covering R :
  $AM(R) = \{P \in R \ / \ \forall Q \in R, B(P, r_P) \not\subset B(Q, r_Q)\}$



Computation for d_4 and d_8 :

Let DM be the distance map to the background. The medial axis is composed of the points corresponding to the local maxima of DM : $P \in AM(R) \Leftrightarrow \forall Q \in R \cap N_8(P), r_P \geq r_Q$

General algorithm : ?

Contents

3 - Digital geometry tools for analyzing object boundaries

Regions and their boundaries

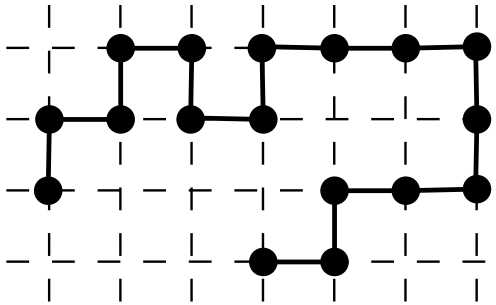
Digital lines and planes

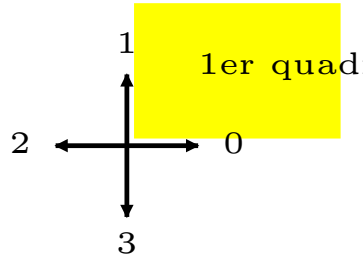
Geometric features of a digital boundary

- 2D digital contour : tangent
- 2D digital contour : length / perimeter
- 2D digital contour : curvature
- 3D Extension

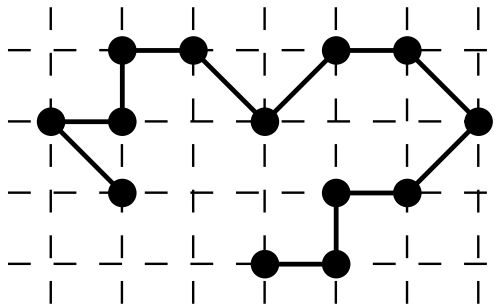
Path

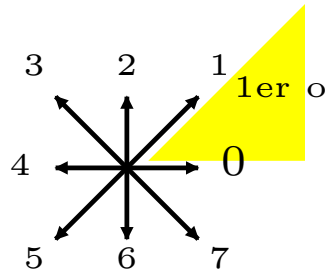
- A k-connected *path* is a sequence of integer points $(P_0, P_1, .., P_n)$ such as $\forall i \in 1..n$, P_{i-1} and P_i are k-connected.
- *Freeman's code* : The path $(P_0, .., P_n)$ is represented by $(P_0, d_0, \ldots, d_{n-1})$. The direction d_i encodes the elementary move from P_i to P_{i+1}.





0100112223212323

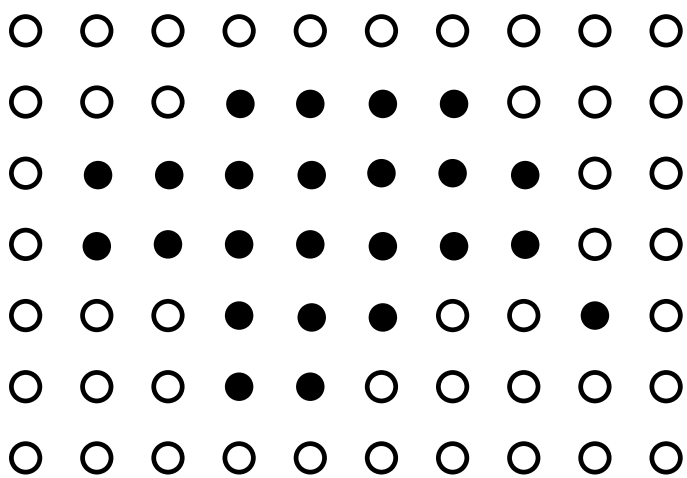




020134534647

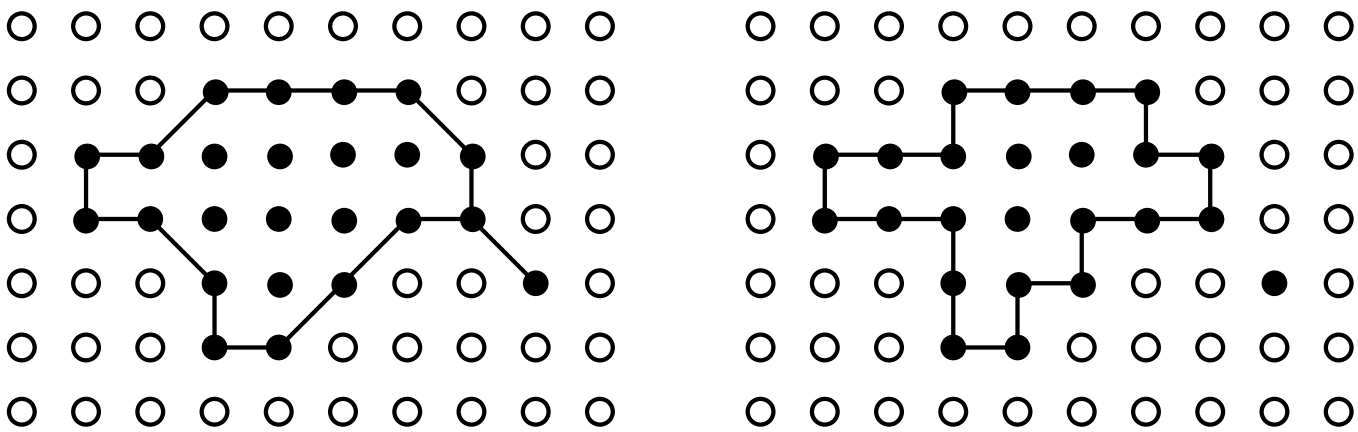
Connected component / region

- *Connected set* : set of integer points E such that $\forall P, Q \in E$, $\exists$ a path $(M_0, .., M_n)$ verifying $M_i \in E$, $M_0 = P$, $M_n = Q$.
- *Connected component* of a set of integer points : maximal connected set (or equivalence class for the adjacency relation).
- Example : set composed of one 8-connected component (of two 4-connected components)



2D boundary, first definition I

The *boundary* of an 8-connected (respectively 4-connected) region R is the set of points of R having at least one 4-neighbor (resp. 8-neighbor) not belonging to R.
$\Rightarrow$ The boundary is composed of 8-connected (resp. 4-connected) paths.

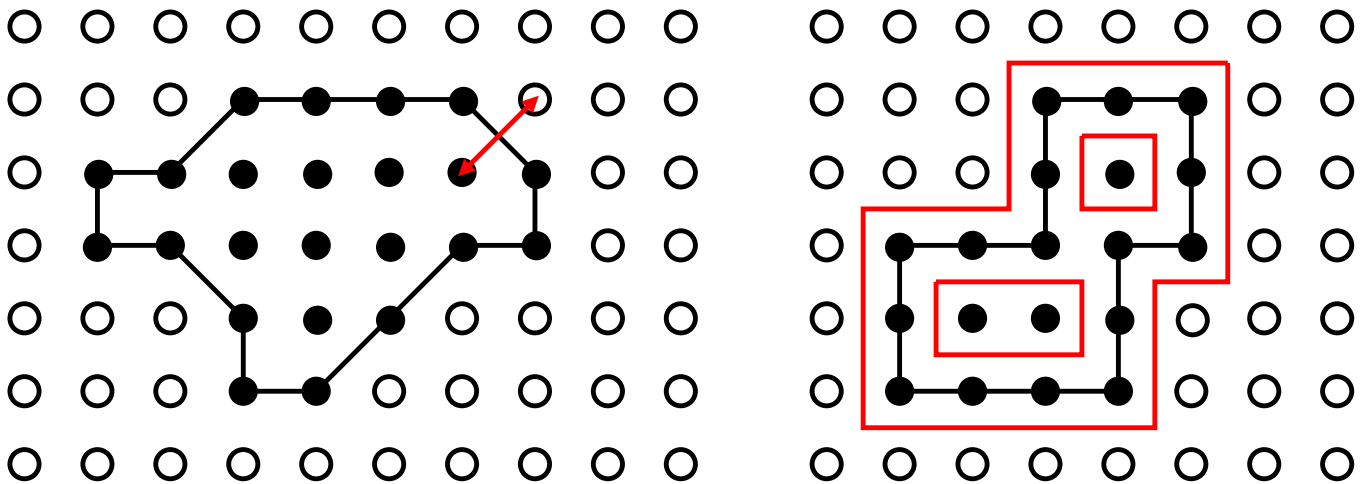


Problems :

- 2 adjacent region share no boundary points

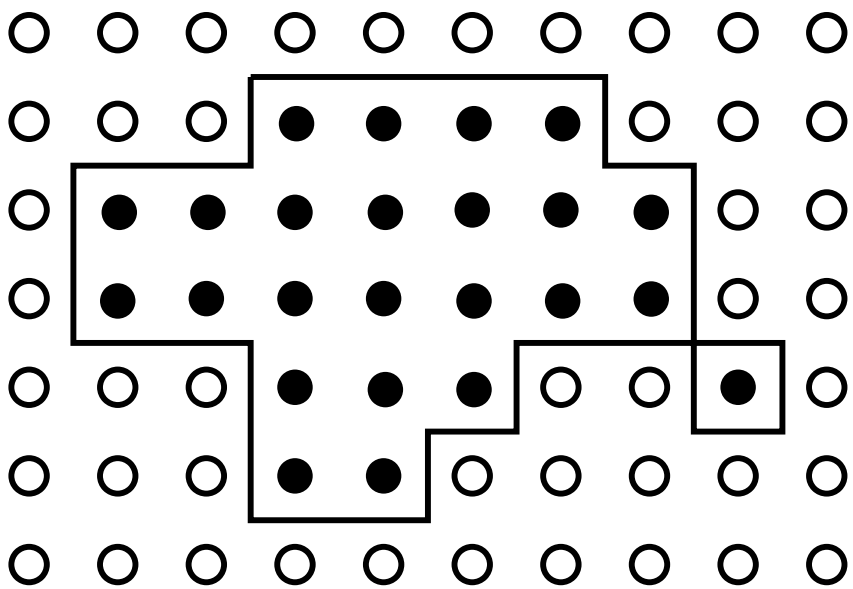
2D boundary, first definition II

- An 8-connected boundary don't split the 2D grid into two distinct 8-connected components.
- A 4-connected boundary can split the 2D grid into more than two distinct 4-connected components.



2D boundary, inter-pixel definition

Each integer point of the region is considered as a pixel (unitary square of side 1 centered on the point).
The inter-pixel boundary is a *sequence of edges* of pixels on the border of the region. This boundary can be represented by a 4-connected digital path (translated in the half-integer grid).

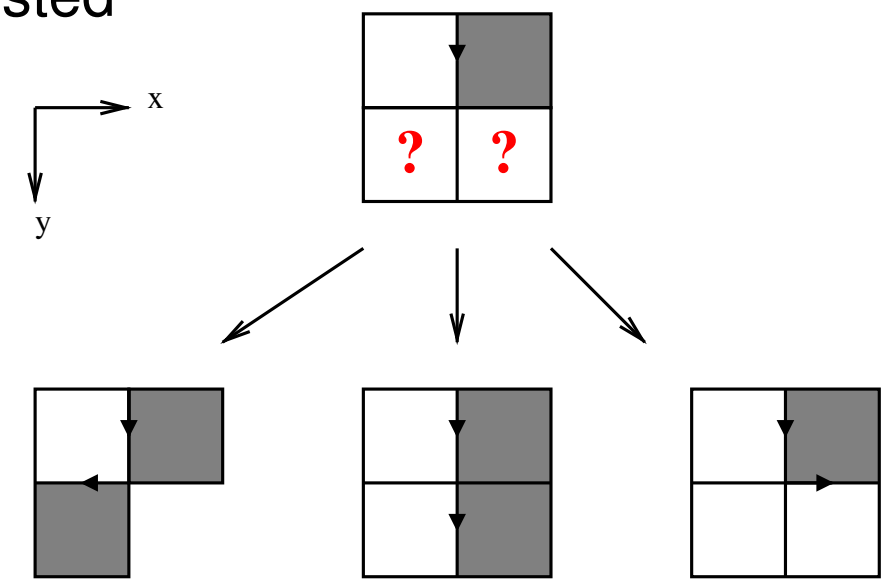


Tracking an inter-pixel boundary I

8-connected object

Hypothesis : the region is at the left side of the contour.

- Search for the first point (x, y) by scanning the image
- Initialize the result with 23 and the current direction $d = 3$
- Search for the next direction : at each step one or two points are tested



- End : reaching the first point.

Tracking an inter-pixel boundary II

Algorithm

$\Delta x = \{1, 0, -1, 0\}$
$\Delta y = \{0, -1, 0, 1\}$

If $P = \{x + \Delta x[d] + \Delta x[d-1], y + \Delta y[d] + \Delta y[d-1]\}$ point of the object

d = d-1

(x, y) = P

Else if $P = \{x + \Delta x[d], y + \Delta y[d]\}$ point of the object

(x, y) = P

Else

d = d+1

Add d to the result

Tracking the border of a region I

8-connected object

Search for the first point (x, y) by scanning the image, $dir = 4$

Search for the next point :

Do

$dir = (dir + 1)$ mod 8

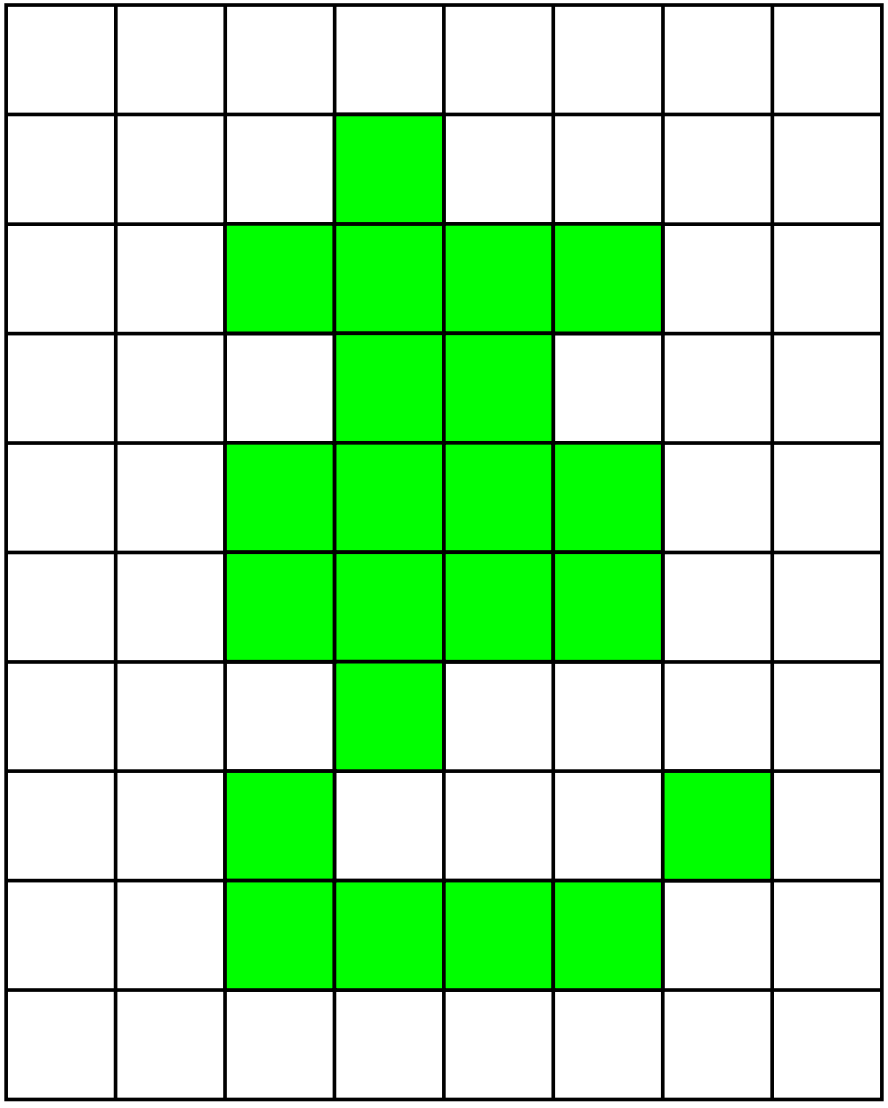
Until the next point (x, y) in the dir direction be a point of the object.

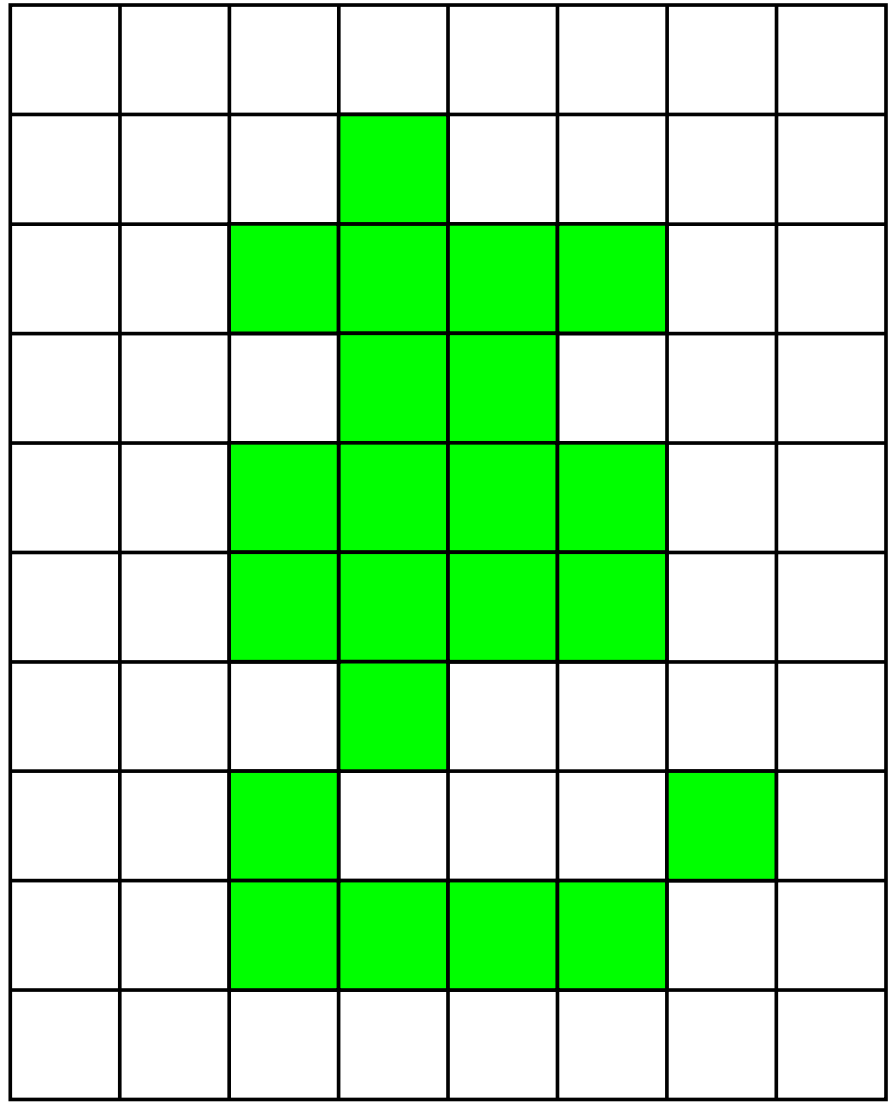
Add dir to the Freeman's code

$dir = MAJ[dir]$ $(MAJ = \{6, 6, 0, 0, 2, 2, 4, 4\})$

End : reaching the two first points.

Tracking the border of a region II





Some theory I

- *Digital space* : (V, W) where V is the set of integer points (pixels in 2D, voxels in 3D) and W an adjacency relation between integer points (represented by pixel edges in 2D, voxel faces or surfels in 3D)
- *Surface S* : non empty subset of W
  - $II(S) = \{u / \exists v \in V$ such that $(u, v) \in S\}$
  - $IE(S) = \{v / \exists u \in V$ such that $(u, v) \in S\}$
  - $I(S) = \{p \in V / \exists$ a W-path from p to $II(S)$ not intersecting $S\}$
  - $E(S) = \{p \in V / \exists$ a W-path from p to $IE(S)$ not intersecting $S\}$
- A surface is *almost-Jordan* if and only if any W-path from a point of $II(S)$ to a point of $IE(S)$ intersects S.
  $\Leftrightarrow I(S) \cap E(S) = \emptyset$
- A surface is $\kappa\lambda$*-Jordan* if and only if it is almost-Jordan and its interior is κ-connected and its exterior is λ-connected.

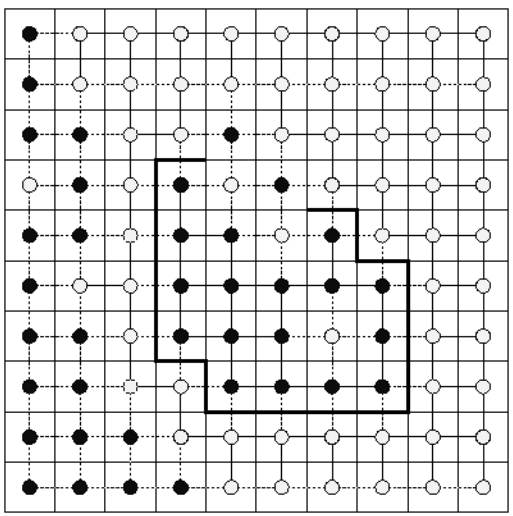
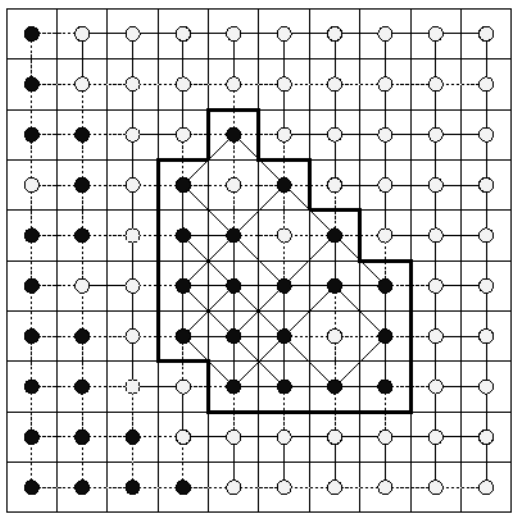
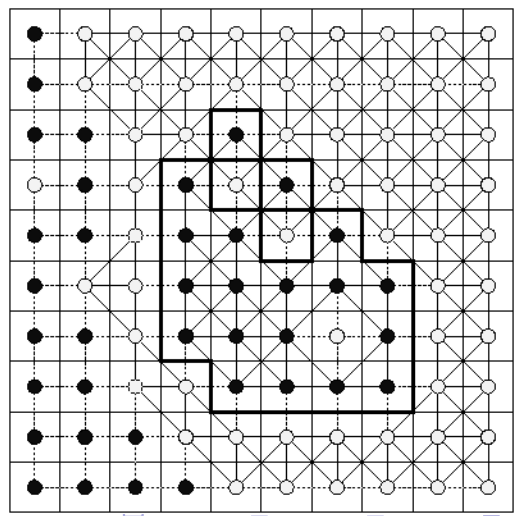
Some theory II

- *Boundary* of O and Q subsets of V :
  $\partial(O, Q) = \{(u, v) \in W / u \in O \text{ and } v \in Q\}$.

In a binary image :

- S is a $\kappa\lambda$-*border* if there exists a black κ-connected object O and a white λ-connected object Q such that $S = \partial(O, Q)$.
- *Jordan pair* :
  - 2D : $(8, 4)$, $(8, 8)$
  - 3D : $(18, 6)$, $(26, 6)$, $(14, 6)$

For a Jordan pair (κ, λ), any $\kappa\lambda$-border is $\kappa\lambda$-Jordan.

3D boundary tracking I

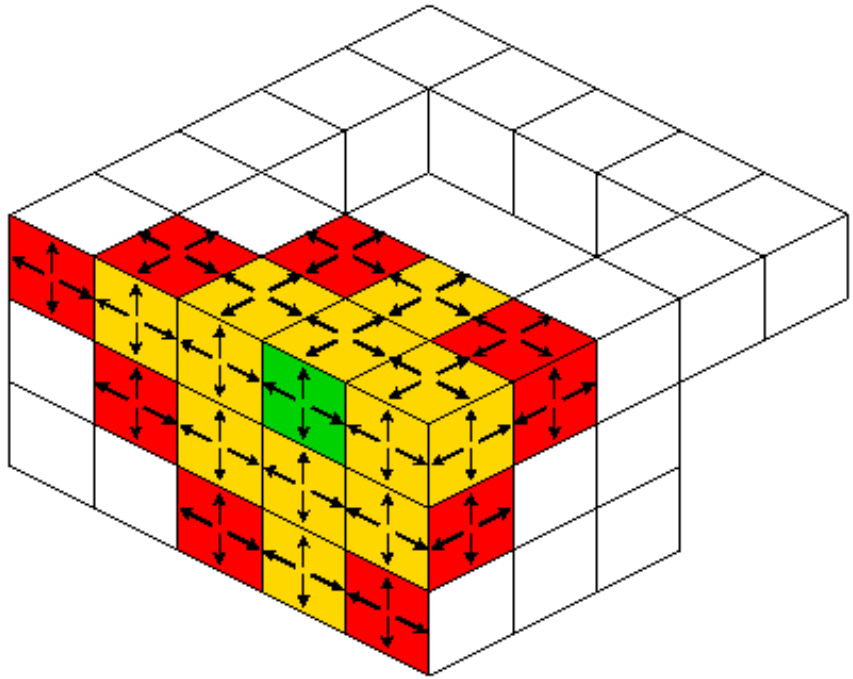
Binary image I
Bel : surfel (u, v) such that u is black and v is white
Set of the bels of I : $B(I)$
Initial bel : b_0
Bel adjacency : β

Algorithm

E : set of processed bels
Q : queue of bels adjacent to E and to be processed
L : result list

Put b_0 in Q and in E
While Q is non empty
    Get b out of Q and put it in L
    For each $b' \in B(I)$ such that $\beta(b, b')$
        If $b' \notin E$
            Put b' in E and in Q

3D boundary tracking II



3 - Digital geometry tools for analyzing object boundaries

Regions and their boundaries

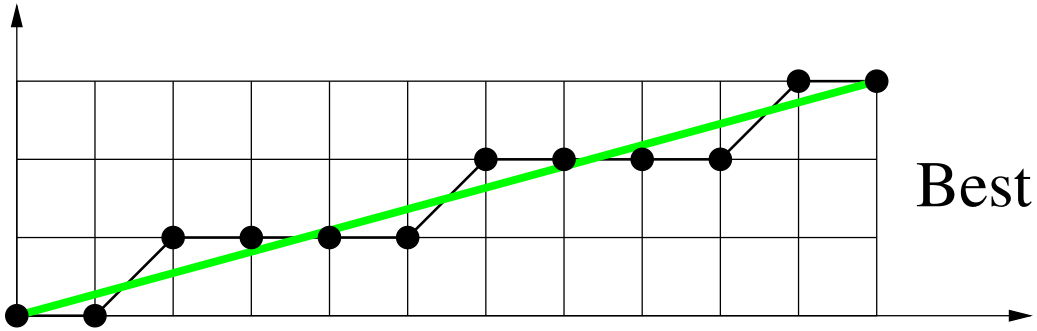
Digital lines and planes

Geometric features of a digital boundary

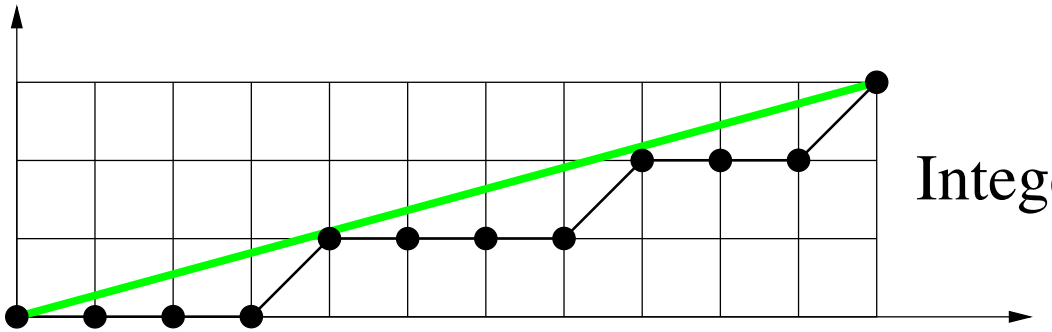
- 2D digital contour : tangent
- 2D digital contour : length / perimeter
- 2D digital contour : curvature
- 3D Extension

Digital line - Definition

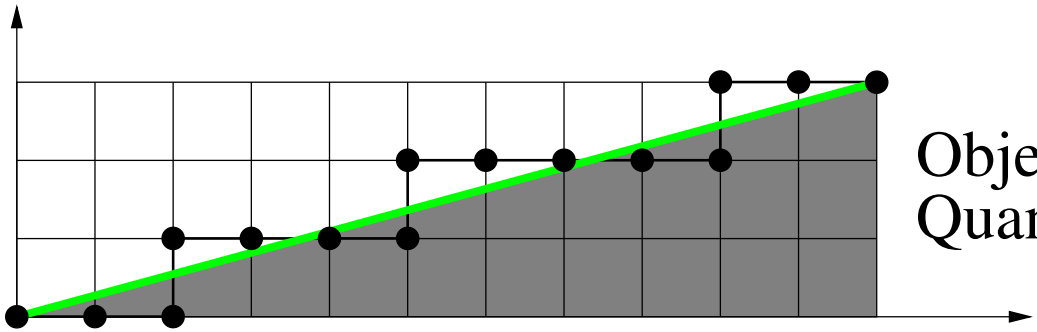
Digitization of a continuous line



Best fit



Integer part

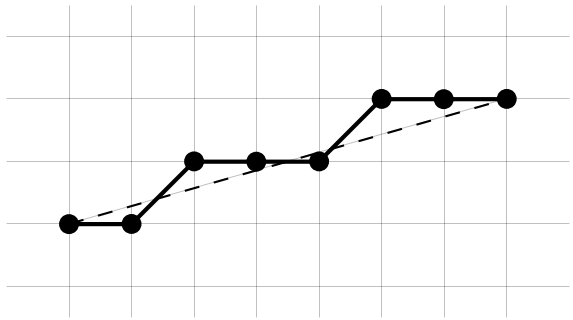


Object Boundary
Quantization (OBQ)

Rosenfeld's characterization (74)

An 8-connected path C is a digital line segment if and only if it verifies the cord property :

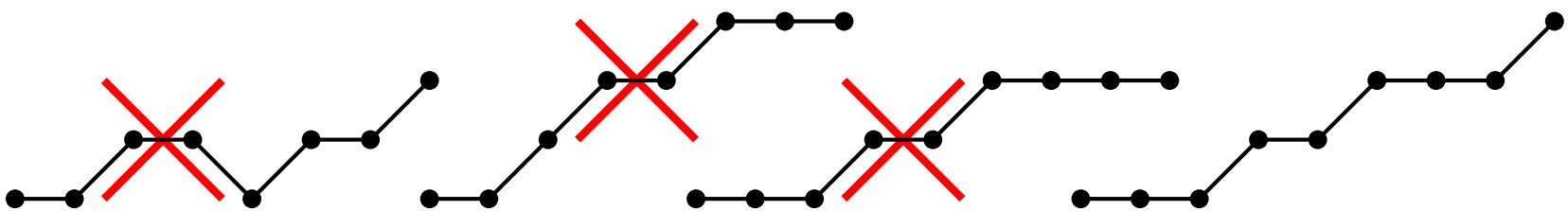
$$\forall P, Q \in C, \forall m \in [P, Q], \exists M \in C / max(|x_M - x_m|, |y_M - y_m|) < 1$$



Freeman's characterization (74)

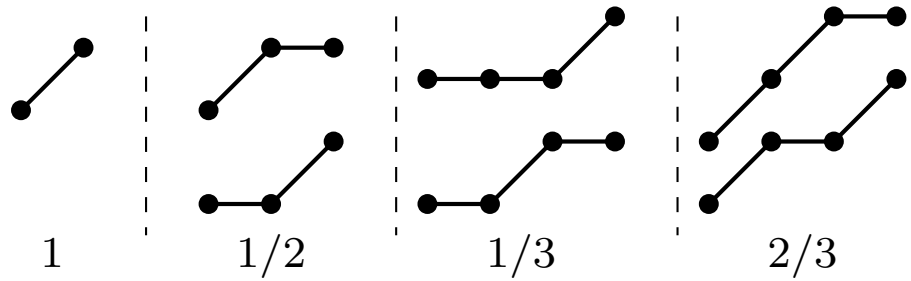
An 8-connected path is a digital line segment if and only if

- Its Freeman code contains at most 2 different elementary directions which differ by 1 modulo 8.
- If the code is composed of two directions, one of them occurs singly.
- Successive occurrences of the singly occurring direction are as uniformly spaced as possible.

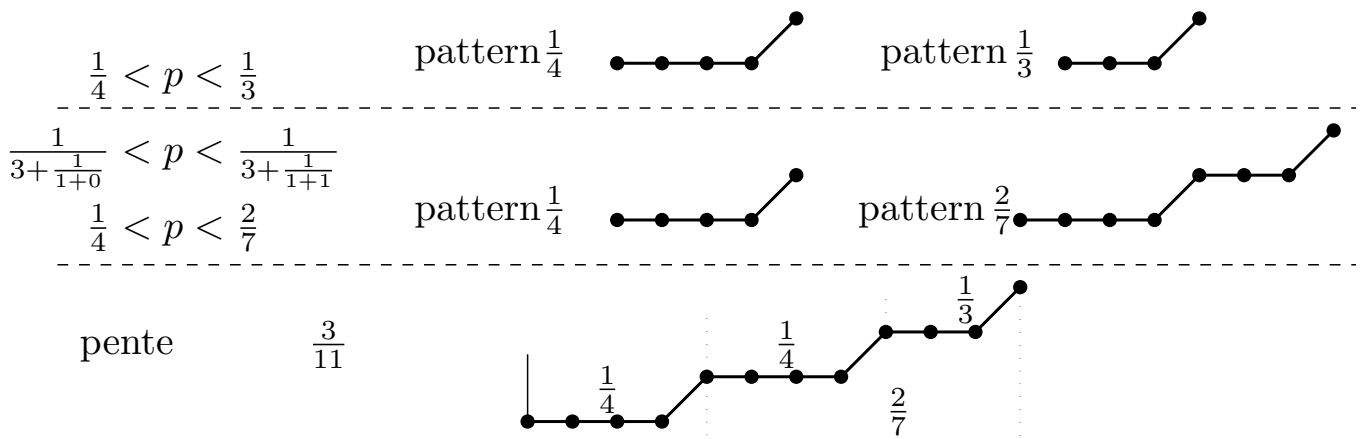


Pattern of a digital line

A digital straight line segment with a rational slope is composed of a repeated pattern



Example : line of slope $p = \frac{3}{11} = \cfrac{1}{3 + \cfrac{1}{1+\frac{1}{2}}}$



Arithmetic definition I

[Réveillès 91]

Line's characteristics : $(a, b, \mu) \in \mathbb{Z}^3$, $\omega \in N$

slope : $\frac{a}{b}$, lower bound : μ, thickness : ω

The line (a, b, μ, ω) is the set of integer points verifying :

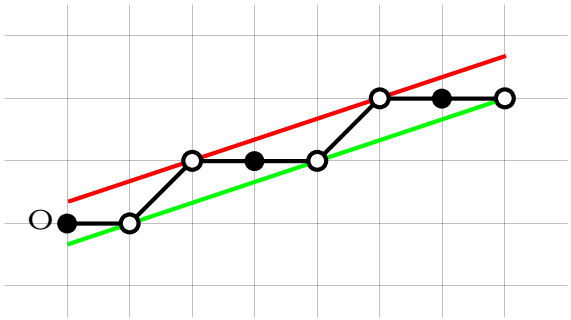
$$\boxed{\mu \leq ax - by < \mu + \omega}$$

Leaning lines : $ax - by = \mu$ and $ax - by = \mu + \omega - 1$

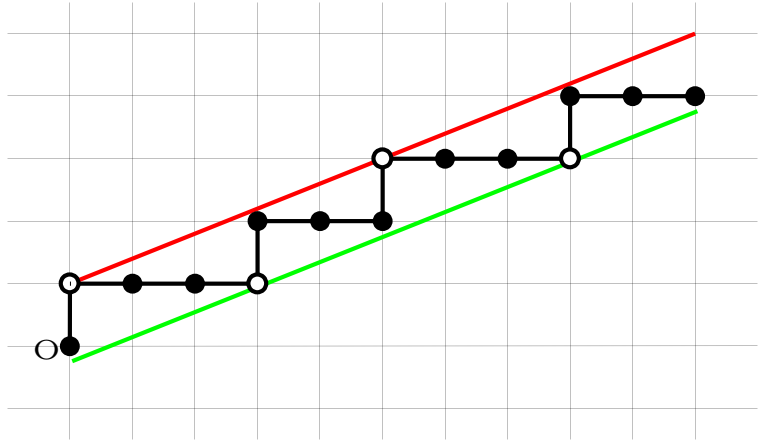
Leaning points : integer points belonging to the leaning lines.

$\omega = max(|a|, |b|) \Rightarrow$ 8-connected line
$\omega = |a| + |b| \Rightarrow$ 4-connected line

Arithmetic definition II



$(a, b, \mu) = (1, 3, -1)$



$(a, b, \mu) = (2, 5, -5)$

Upper leaning line

Lower leaning line

Leaning point

Recognition algorithm I

Incremental algorithm for the recognition of a digital straight segment along an 8-connected path [Debled 95]

Idea : The points of the path are added one by one. At each step the characteristics of the recognized segment are updated.

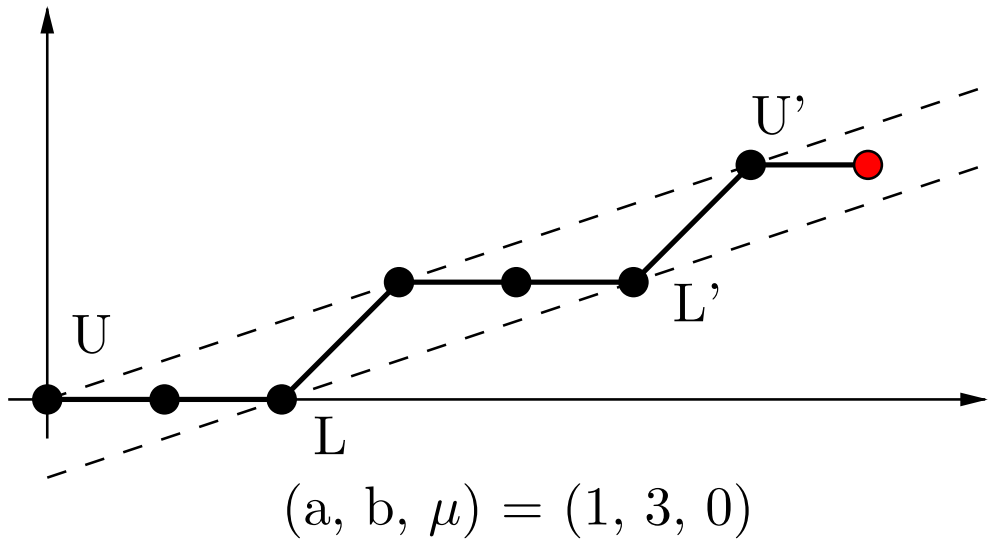
Let $S = (P_0, ..., P_n)$ be a line segment of the 1rst octant and M the point to be added
Is $S + M$ a line segment ? If it is, which are its characteristics ?

Necessary condition : M is a neighbor of P_n and the elementary direction from P_n to M is compatible with the two directions composing $S \Rightarrow d(P_n, M) = O$ or 1

Recognition algorithm II

Case 1 : $\mu \leq ax_M - by_M < \mu + b$

M extends *S* (same characteristics)

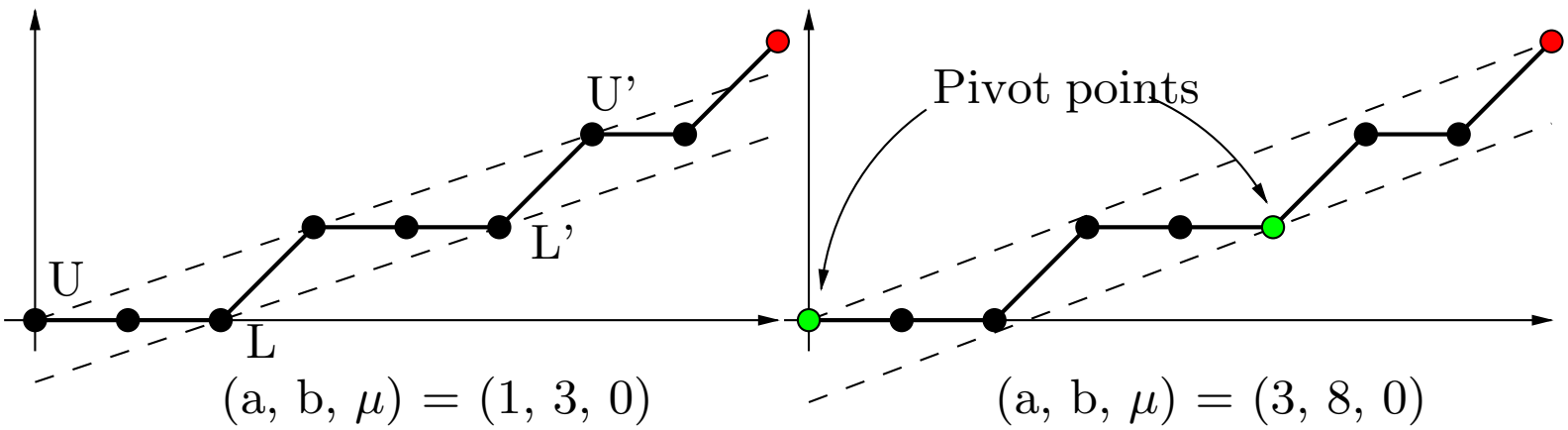


U upper leaning point with minimum abscissa
U′ upper leaning point with maximum abscissa
L lower leaning point with minimum abscissa
L′ lower leaning point with maximum abscissa

Recognition algorithm III

Case 2.1 : $ax_M - by_M = \mu - 1$

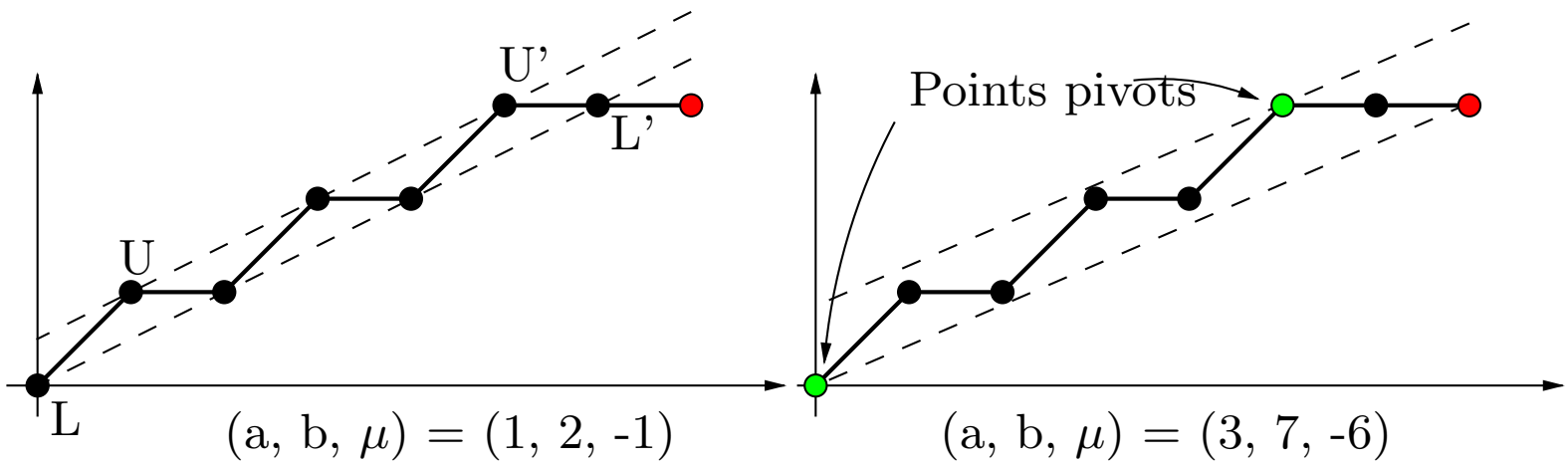
$S + M$ is a segment which characteristics are different from S's : the slope of $S + M$ is **greater** than the slope of S.



Recognition algorithm IV

Case 2.2 : $ax_M - by_M = \mu + b$

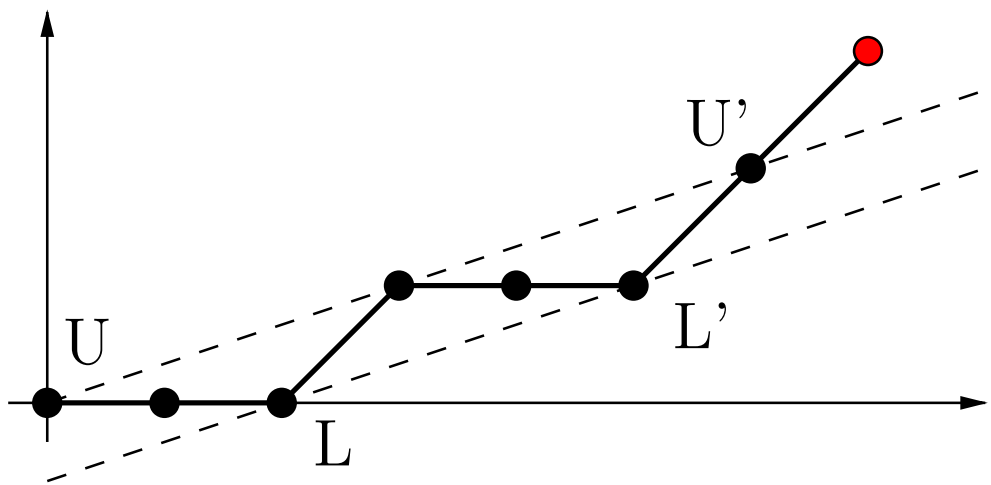
$S + M$ is a segment which characteristics are different from S's : the slope of $S + M$ is **lower** than the slope of S.



Recognition algorithm V

Case 3 : $ax_M - by_M < \mu - 1$ or $ax_M - by_M > \mu + b$

$S + M$ is not a line segment.



$(\mathrm{a},\ \mathrm{b},\ \mu) = (1,\ 3,\ 0)$

Recognition algorithm VI

Adding a point

remainder = $ax_M - by_M$

If ($\mu \leq remainder < \mu + b$)

  If ($remainder == \mu$)

    M is an upper leaning point.

      $U' = M$

  If ($remainder == \mu + b - 1$)

    M is a lower leaning point

      $L' = M$

Else if ($remainder == \mu - 1$)

    The slope increases

    $L = L'$

    $U' = M$

    $a = y_M - y_U$

    $b = x_M - x_U$

    $\mu = ax_M - by_M$

Recognition algorithm VII

Else if ($remainder == \mu + b$)

The slope decreases

$U = U'$

$L' = M$

$a = y_M - y_L$

$b = x_M - x_L$

$\mu = ax_M - by_M - b + 1$

Else M can not be added to the segment

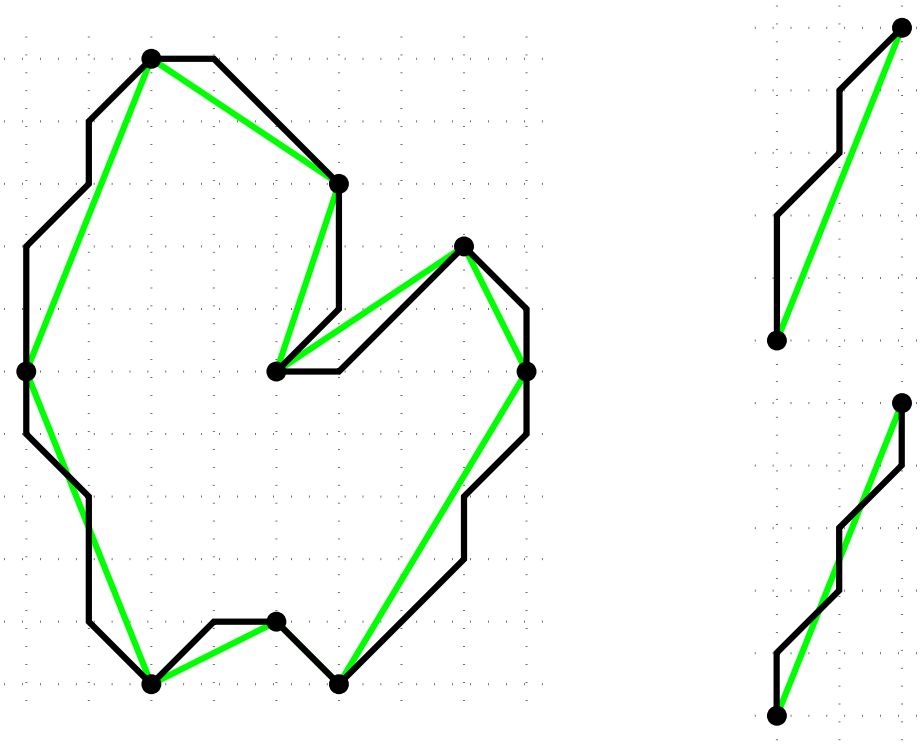
Recognition algorithm VIII

Global algorithm

```
(x, y) = ( 1, code(0))
(a, b, μ) = (y, 1, 0)
U = L = (0, 0)
U′ = L′ = (1, y)
i = 1
Repeat
    x = x + 1
    y = y + code(i)
    i = i + 1
Until add-point(x, y)
```

Direct use : vectorization I

Iterative approach



Remark : loss of information

Other vectorization algorithm I

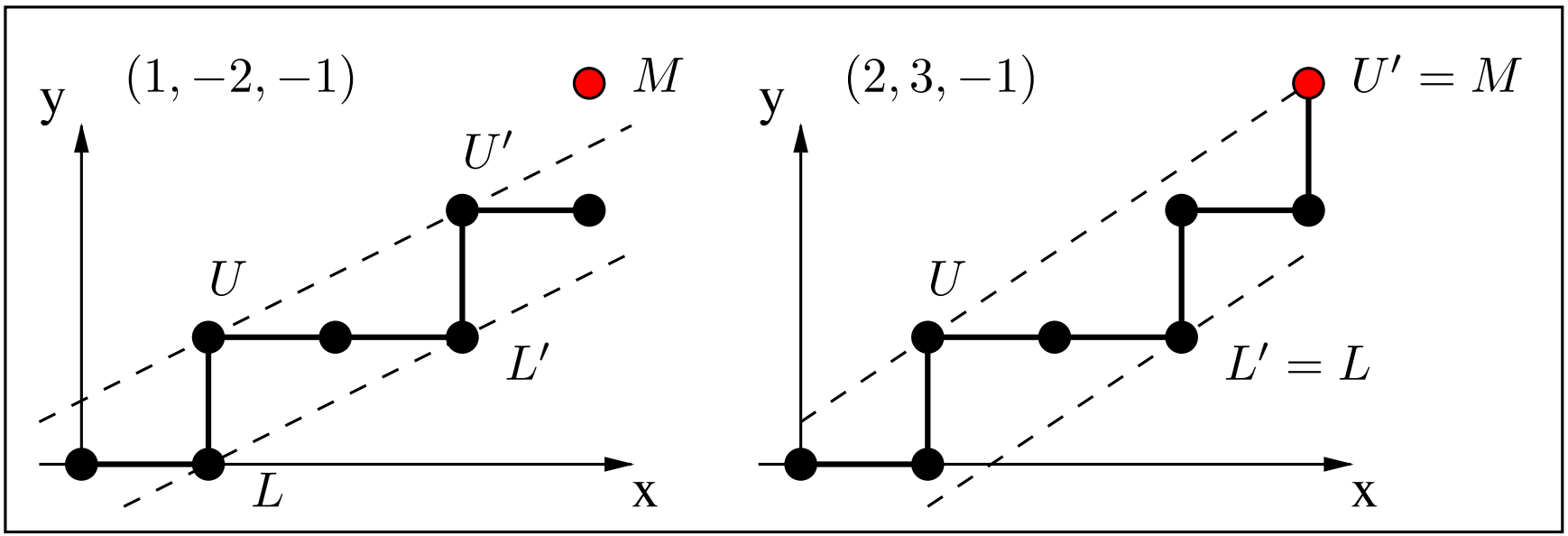
- iterative approach based on an error measure (distance criterion, angular criterion)
- recursive approach : find the most significant point between 2 contour points (Douglas-Peuker, curvature computation)

4-connected case I

Test if a point M can be added to a line segment :

(1) M is between the leaning lines : OK

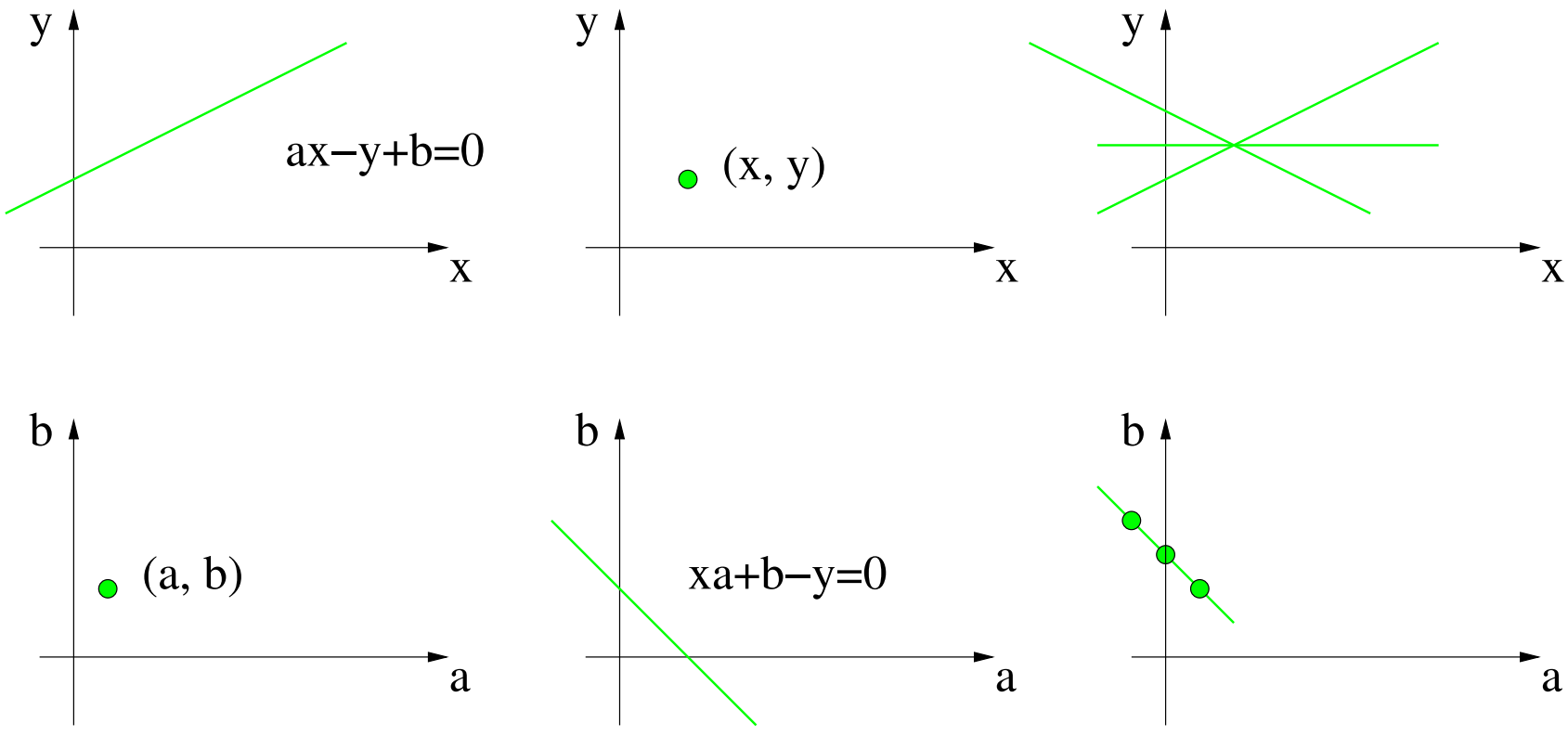
(2) $ax_M - by_M = \mu - 1$: M is "above but not too much"



(3) $ax_M - by_M = \mu + a + b$: M is "under but not too much"

Dual space I

Another approach for recognizing a 2D digital line segment



Dual space II

Another approach for recognizing a 2D digital line segment

Digitization OBQ of $ax - y + b = 0, 0 \leq a < 1$:
set of the points verifying $0 \leq ax - y + b < 1$

A set of points (x_i, y_i) belonging to a digital line is represented in the dual space by the intersection of strips defined by $0 \leq x_i a + b - y_i < 1$

Segment recognition =
verify if a set of linear constraints is valid

Incremental linear algorithm based on this idea

Digital plane I

Arithmetic definition

The *plane of characteristics* $(a, b, c, \mu, \omega) \in \mathbb{Z}^5$ is the set of integer points verifying :

$$\boxed{0 \leq ax + by + cz + \mu < \omega}$$

(a, b, c) : normal vector

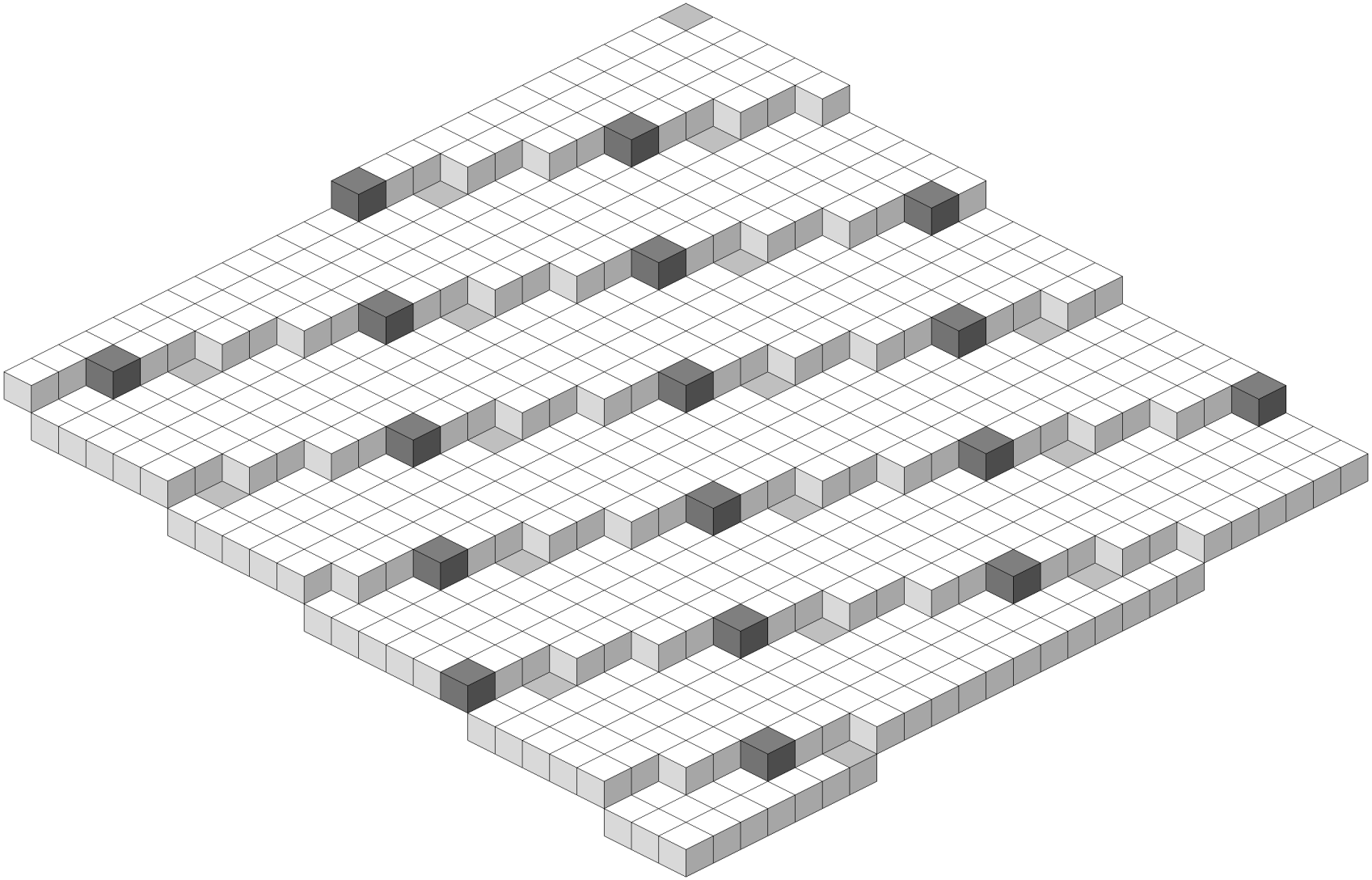
μ : location in the plane

ω : thickness

$\omega = max(|a|, |b|, |c|) \Rightarrow$ naive plane (18-connected)
$\omega = |a| + |b| + |c| \Rightarrow$ standard plane (6-connected)

Digital plane II

Example : naive plane $(3, 7, 37, 0)$



Digital plane : recognition

[Sivignon 04]

V set of voxels containing $(0, 0, 0)$
Question : what is the set S of parameters $(\alpha, \beta, \gamma), 0 \leq \alpha \leq \beta < 1, 0 \leq \gamma \leq 1$ such that all the voxels of V belong to the OBQ digitization of $\alpha x + \beta y + z + \gamma = 0$?

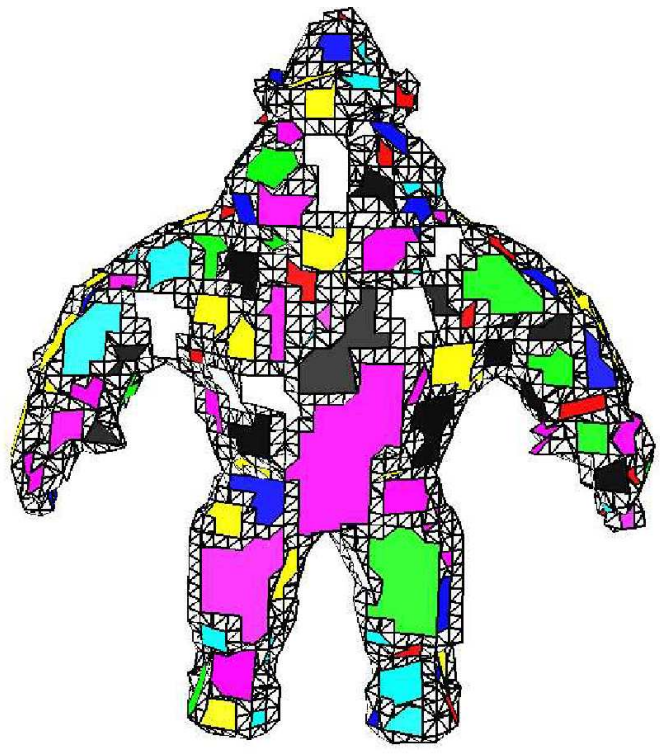
coordinates space	parameter space
plane	point
point	plane
a voxel of a digital plane	area between 2 parallel planes

Computing S : half-spaces intersection at each voxel addition
Result = polyhedron, polygon, line segment or empty set.

Polyhedrization I

First approach : split the surface in digital plane pieces + compute a polygonal curve for each border
Other approach : simplify the result of the marching cube algorithm

Polyhedrization II



3 - Digital geometry tools for analyzing object boundaries

Geometry of a digital boundary I

Issue

An infinite number of shapes have the same digitization $\Rightarrow$ there is not ONE unique value of the geometric features

Hypothesis on the underlying real boundary : smooth curve with bounded curvature for example

Estimators : length/area, tangent/tangent plane, curvature

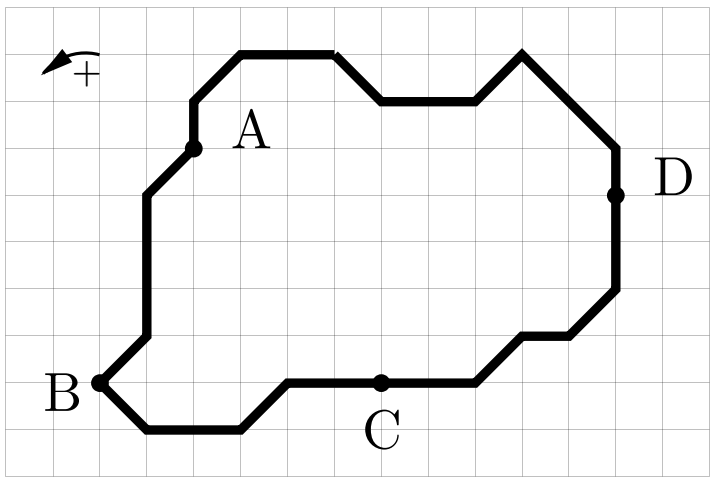
Properties :

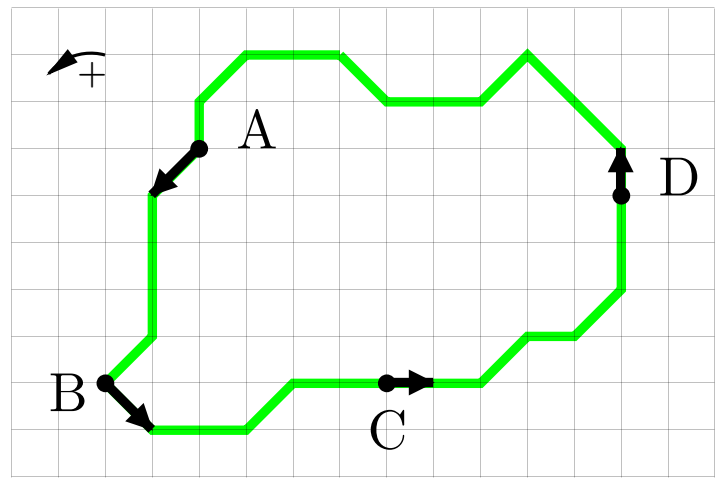
- asymptotic convergence
- good estimation at low resolution
- preservation of the shape properties (convexity for example)

Tangent : basic definition I

$C = (P_0, .., P_n)$
$\vec{t_i}$ tangent vector at the ith point of C
$\widehat{\theta_i}$ orientation of $\vec{t_i}$

First approximation : $\vec{t_i} = P_i\vec{P}_{i+1}$
Different possible orientations : 8 if 8-connected contour





Tangent : median filtering I

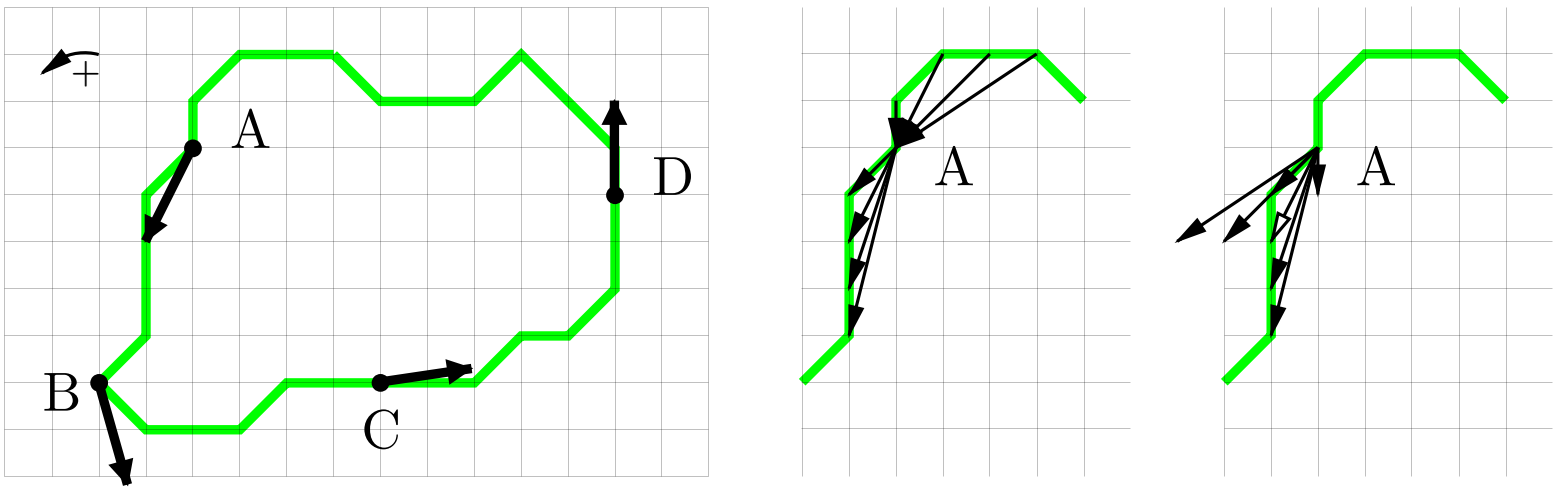
Window of size m around the current point P_i.

$\vec{V}_{i,i+j} = P_i\vec{P}_{i+j} \quad j = 1..m$

$\vec{V}_{i,i+j} = P_{i+j}\vec{P}_i \quad j = -1..-m$

The vectors are sorted according to their angle with the Ox axis. Let θ_k be the orientation of the kth vector in the sorted sequence (numbered from 1 to 2M).

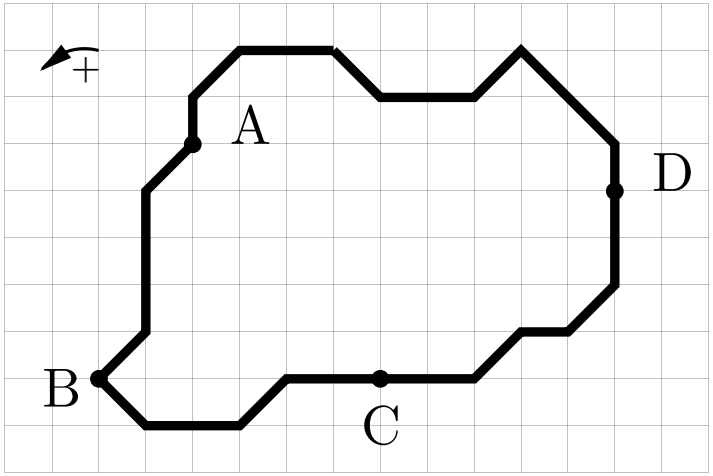
Orientation of the tangent at P_i : $\widehat{\theta_i} = \frac{\theta_M + \theta_{M+1}}{2}$

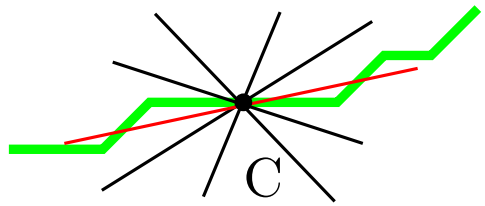


Approximation of the tangent line by a continuous line I

$$\widehat{\theta_i} = argmin_{\Theta}\{\sum_{j=-m}^{m} w(j) d^2(P_{i+j}, l_\theta)\}$$

$w(j)$ weight, for example $w(j) = G_\sigma(j) = \frac{1}{\sigma\sqrt{2\Pi}} e^{\frac{-j^2}{2\sigma^2}}$





Approximation by a digital line I

Definition : *the symmetric digital tangent* at point P_i of the digital curve C is the longest part of C centered at P_i being a digital line segment.

Algorithm : addition of pairs of points around P_i, (P_{i-1}, P_{i+1}).. (P_{i-k}, P_{i+k}) while $(P_{i-k}, .., P_{i+k})$ is a line segment.

$\Rightarrow$ adapt the recognition algorithm of a digital line segment so as to allow the extension of a segment in the 1rst octant/quadrant by a point with negative abscissa.

Digital 4-connected tangent I

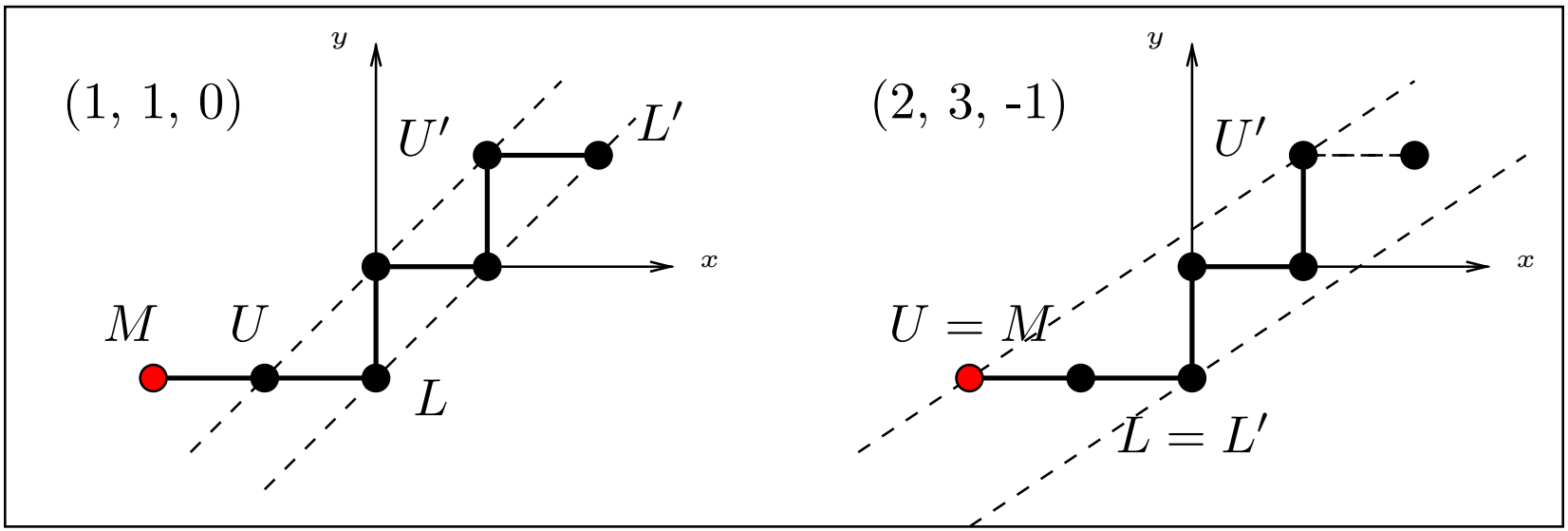
segment recognition

Add a point *M* with positive abscissa : see above

Add a point *M* with negative abscissa

(1) M is in between the leaning lines : OK

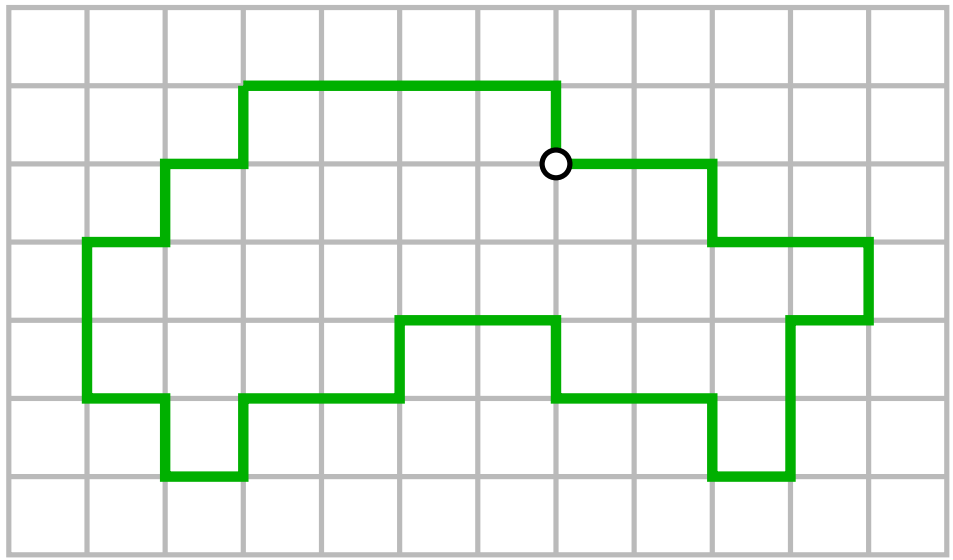
(2) $ax_M - by_M = \mu - 1$: M is "above but not too much"



(3) $ax_M - by_M = \mu + a + b$: M is "under but not too much"

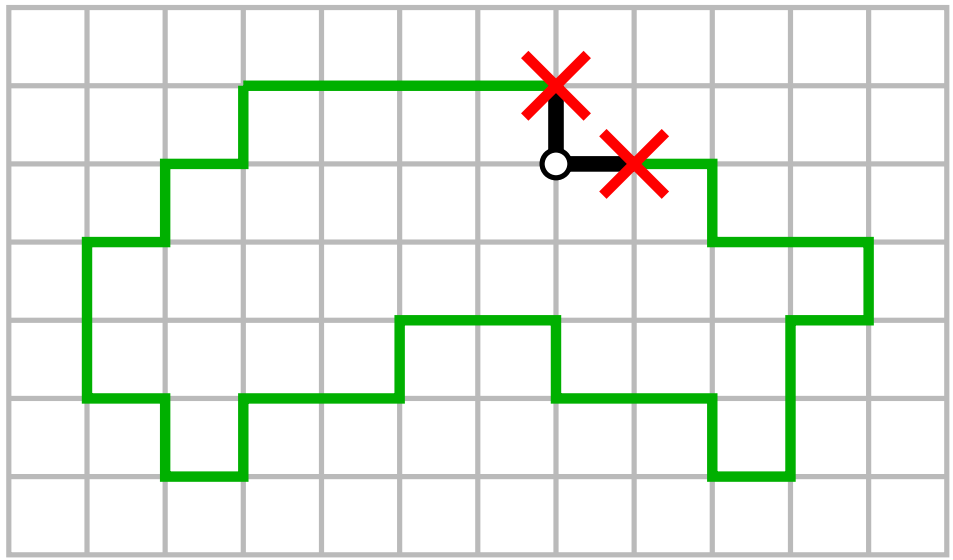
4-connected tangent : example

Symmetric tangent around a point



4-connected tangent : example

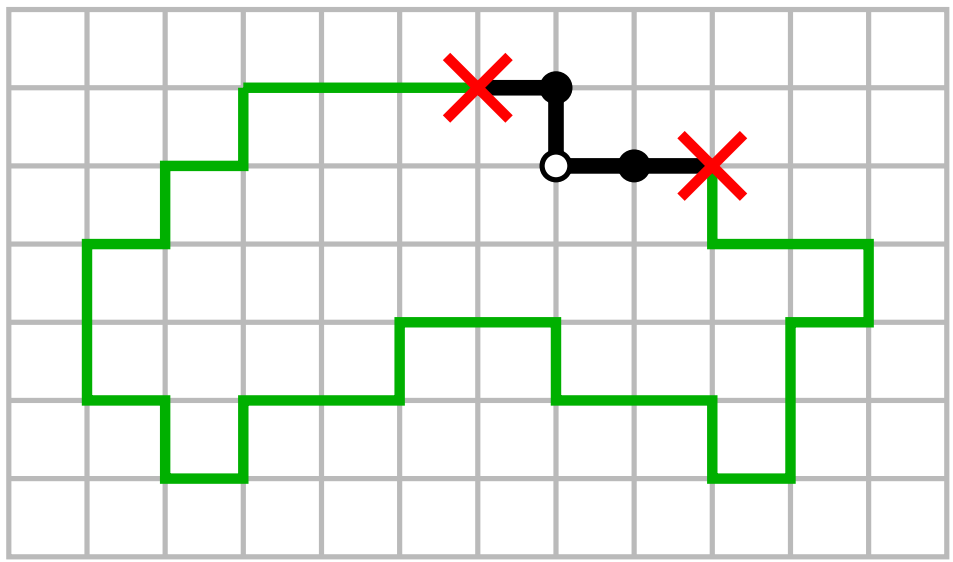
Symmetric tangent around a point



$$(-1, 1, -1)$$

4-connected tangent : example

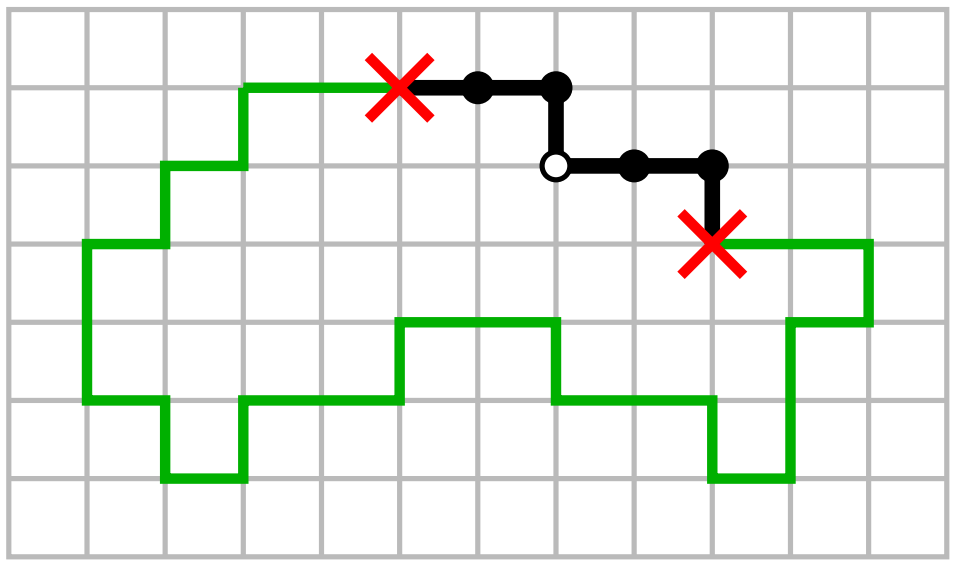
Symmetric tangent around a point



$$(-1, 2, -2)$$

4-connected tangent : example

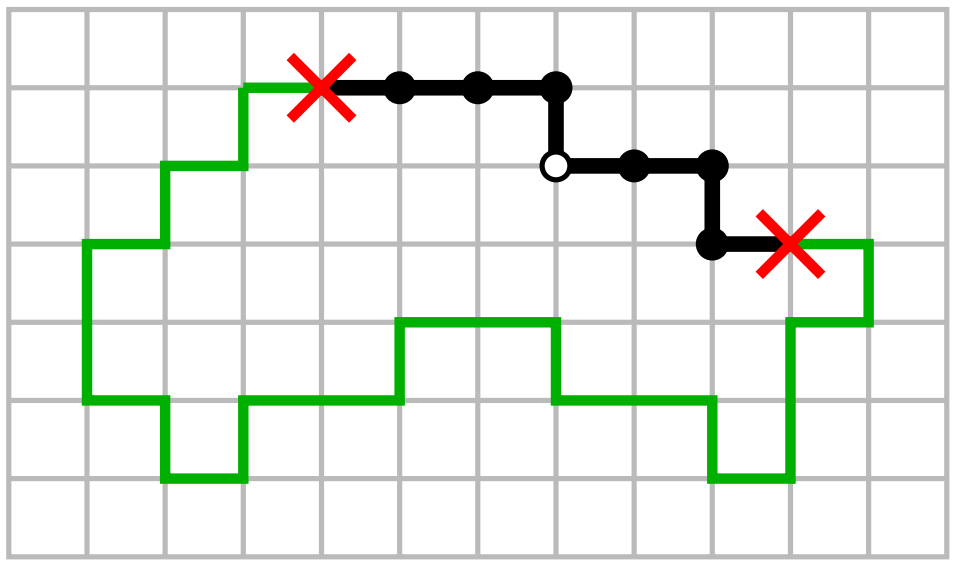
Symmetric tangent around a point



$$(-1, 2, -2)$$

4-connected tangent : example

Symmetric tangent around a point



$$(-2, 5, -5)$$

Digital tangent : conclusion I

Advantages of a definition based on a digital line segment :
- the size of window adapt to the contour shape
- the algorithm is independent from the way of iterating on the boundary and from the starting point
- quite precise estimation
- similar algorithm in 4 or 8-connectivity

Drawbacks :
- compromise localization/precision
- problem for convexity preservation and convergence
-> other approaches (weighted mean of the orientations of digital line segments containing the processed point)

Evaluation of the "real" tangent : line in the middle of the two leaning lines.

Length : simple estimators I

The points of the digital curve are classified according to the local configuration (k classes). The length of the curve is estimated by :

$$\hat{L} = \sum_{i=1}^{k} \psi(C_i) N(C_i)$$

where $N(C_i)$ is the number of points of the class C_i and $\psi(C_i)$ the weight associated with this class.

The weights are estimated for line segments with varying slopes $\Longrightarrow$ implicit vectorization.

The simple estimators are not convergent

Length : simple estimators I

8-connected curve

N number of steps
N_e number of horizontal and vertical steps (even Freeman direction)
N_o number of diagonal steps (odd Freeman direction)
N_c number of corners (between one even and one odd directions)

First estimation of the length of C : $N_e + \sqrt{2}N_o$

Estimators :

$$\hat{L}_1 = 1.1107N$$
$$\hat{L}_K = 0.945N_e + 1.346N_o$$
$$\hat{L}_C = 0.980N_e + 1.406N_o - 0.091N_c$$

Length : simple estimators I

4-connected curve

Rosen-Profitt estimator :

N_c number of corners (between one even and one odd directions)

N_n number of junctions between to identical successive directions

$$\hat{L} = \frac{\Pi(\sqrt{2}+1)}{8} N_n + \frac{\Pi(\sqrt{2}+2)}{16} N_c$$

Koplowitz estimator :

N_{c1} nb of corners with at least a non-corner neighbor

N_{c2} nb of corners with two corner neighbors

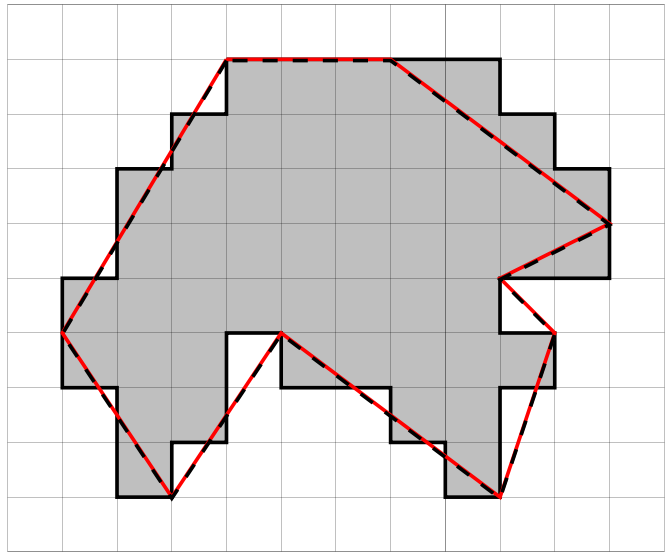
N_{n1} nb of points at the center of a linear sequence of 3 points

N_{n2} nb of points on a linear sequence of more than 3 points

$$\hat{L} = 0.57736 N_{c1} + 0.70251 N_{c2} + 1.06681 N_{n1} + 0.99350 N_{n2}$$

Length : explicit vectorization I

Length = size of a polygonal approximation



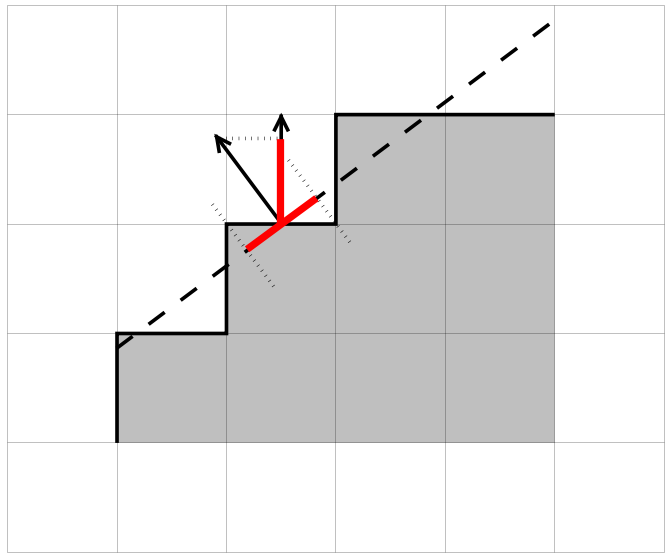
Vectorization based on digital line segments $\Longrightarrow$ convergent length estimator

Length : normal integration I

Edge contribution : $\vec{n} \cdot \vec{e}$
$\vec{n}$: computed normal vector, $\vec{e}$: trivial normal

$\Longrightarrow$ The tangent computation has to be adapted (definition on an edge).

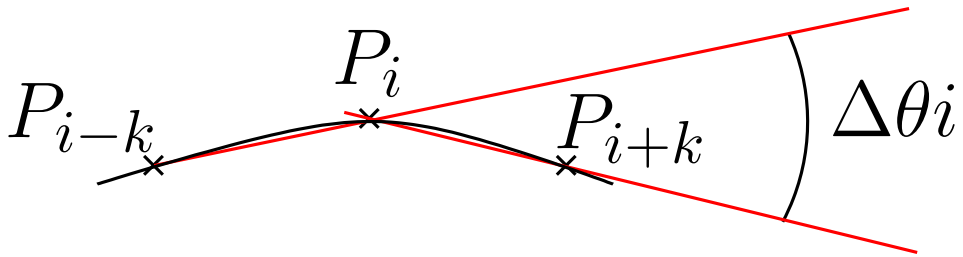
Total length : $\hat{L} = \sum \vec{n}_i \cdot \vec{e}_i$

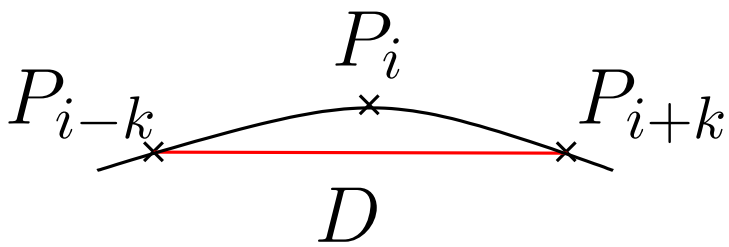


Convergent tangent estimator $\Longrightarrow$ convergent length estimator

Curvature : simple evaluation I

1. Angular variation : $\Delta\theta i$
2. Distance ratio : $\frac{d}{D}$ (d = distance along the curve)





Curvature : tangent derivative I

One curvature definition : tangent orientation derivative [Worring 93].

- Computation by finite differences :

$$\kappa_i = \frac{\widehat{\theta}_{i+k} - \widehat{\theta}_{i-k}}{d(P_{i-k}, P_{i+k})} \quad \text{or} \quad \kappa_i = \frac{||\vec{t}_{i+k} - \vec{t}_{i-k}||}{d(P_{i-k}, P_{i+k})}$$

- Smoothed derivative : convolution by the derivative of a Gaussian function.

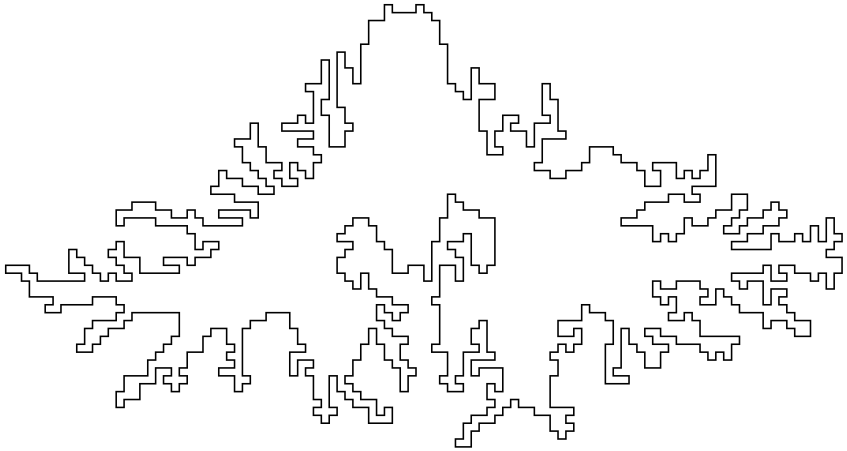
$$\kappa = \frac{\widehat{\theta} * G'_\sigma}{1.1107} \qquad G'_\sigma(x) = \left(\frac{-x}{\sigma^3\sqrt{2\Pi}} e^{\frac{-x^2}{2\sigma^2}}\right)$$

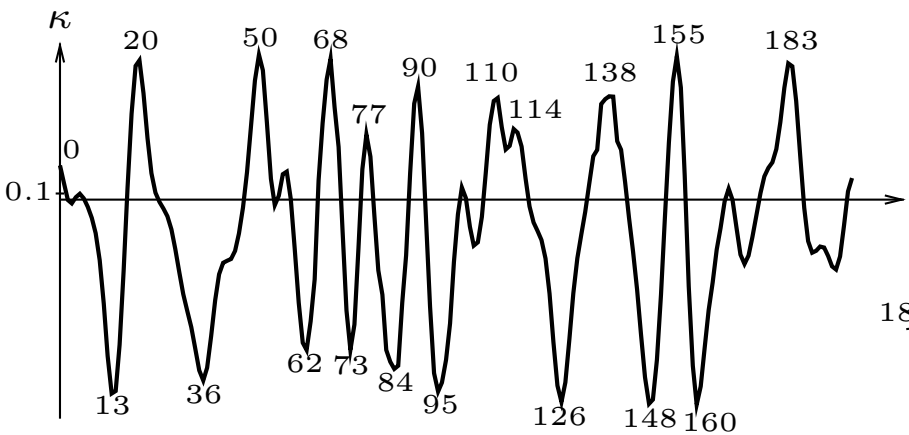
1.1107 : mean distance between two successive points. Size of the computation window : $m = 3\sigma$, $\sigma = 3$ for example.

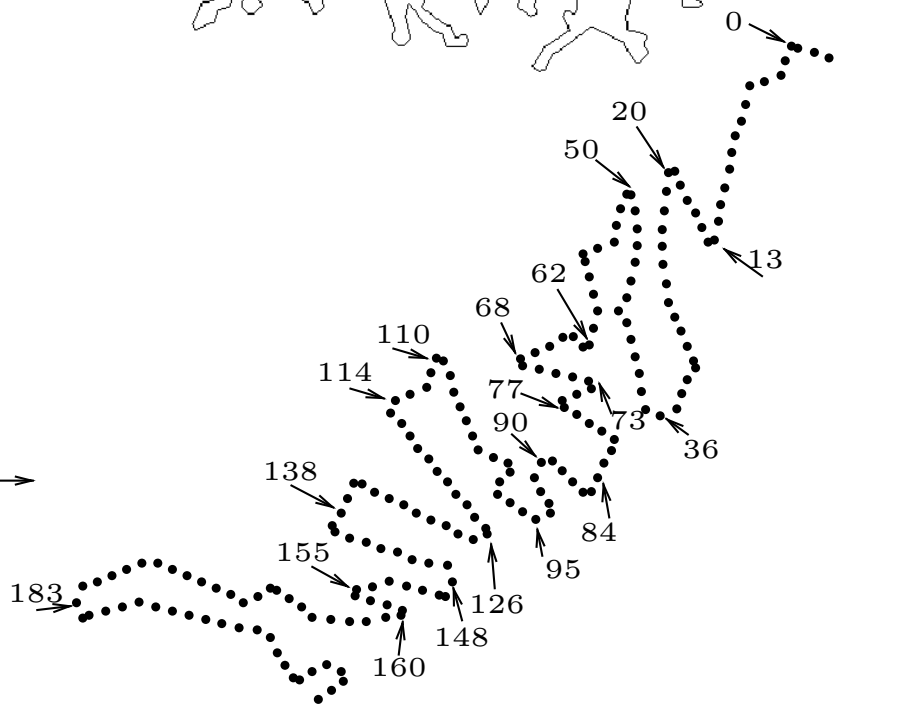
Other curvature computation methods : estimation of the radius of the osculating circle,...

Salient points I

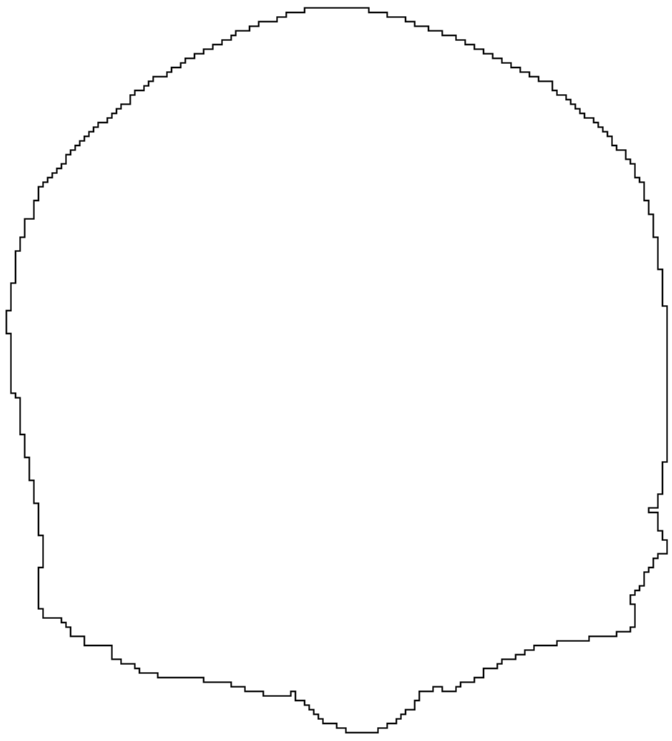
The extrema of the curvature profile of a contour correspond to the dominant points of the contour.

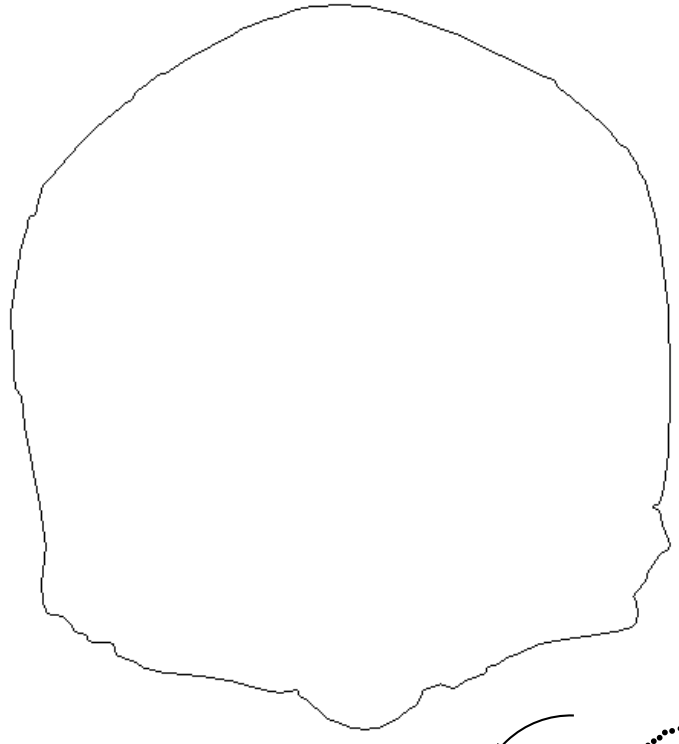


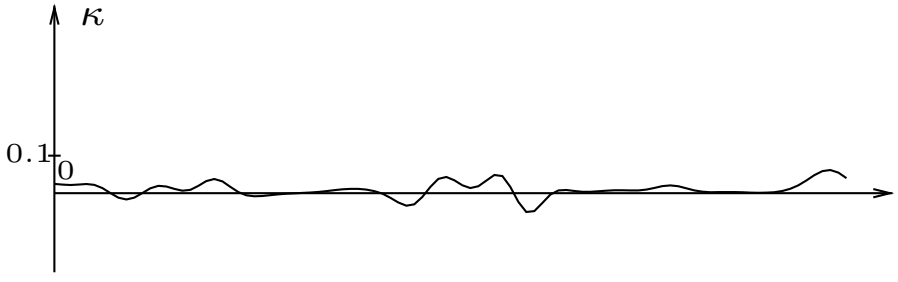




Salient points II



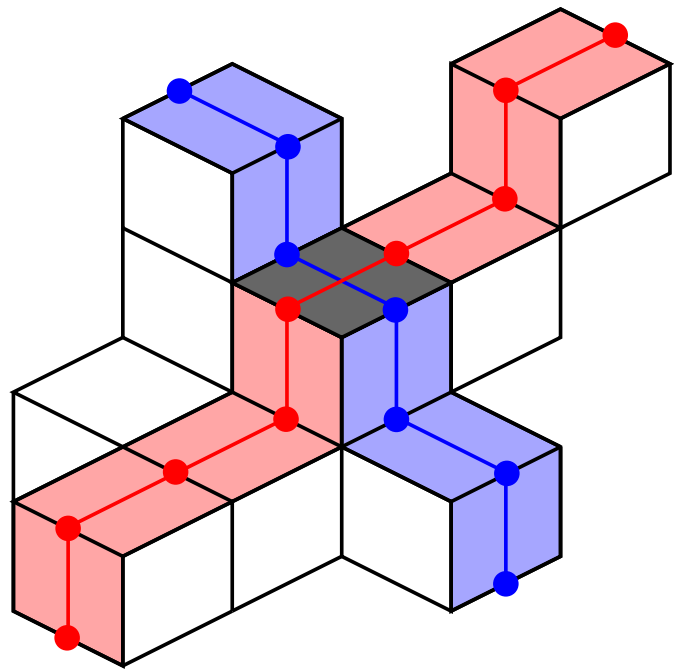




3D extension : normal vector I

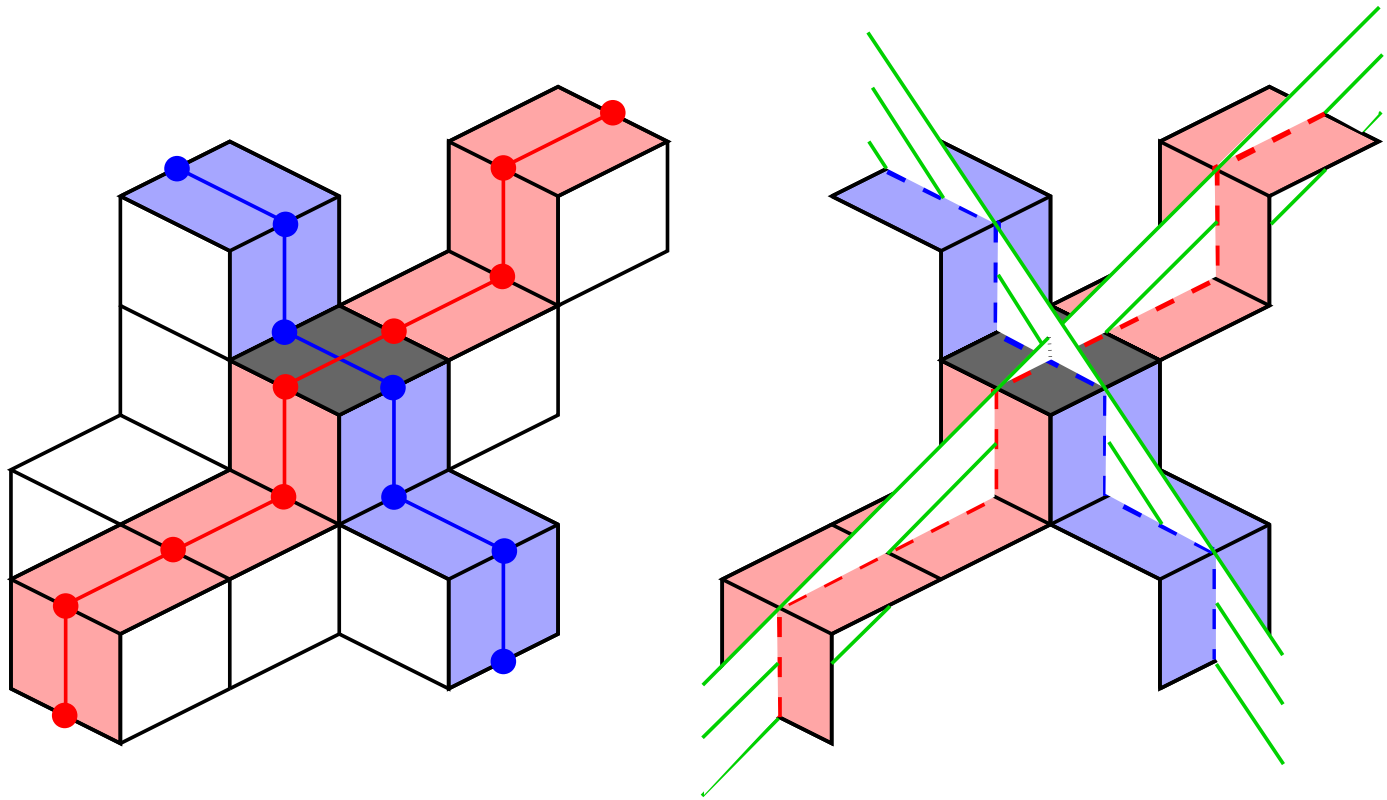
3D region : set of voxels
Surface of a 3D region : set of voxel faces (surfels)

One surfel $\Rightarrow$ two 4-connected contours



3D extension : normal vector II

Computation of the tangent along each 2D contour



Directions of the 2 tangents $\Rightarrow$ normal vector to the surface

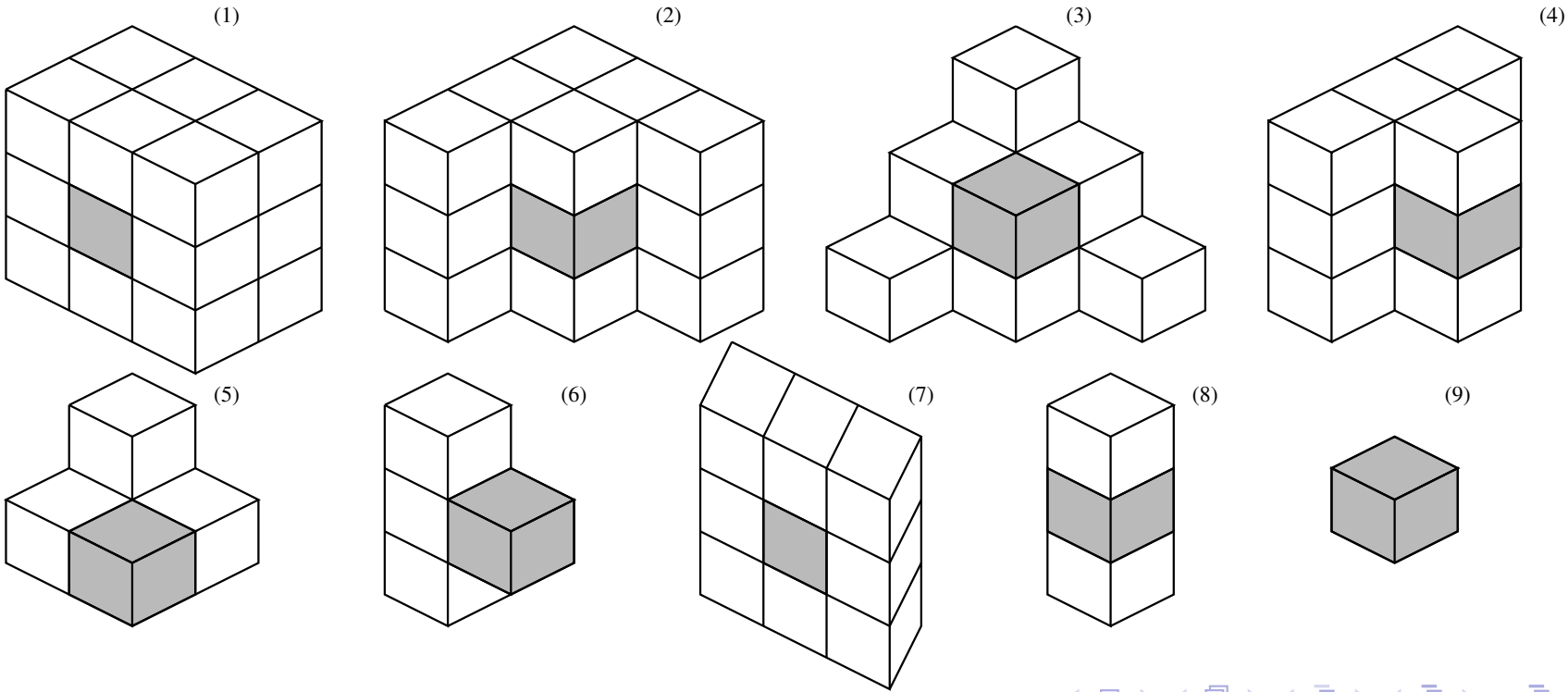
Surface area I

Simple estimator [Mullikin 93]

$\hat{S} = \sum_{i=1}^{6} \psi_i N_i$

$\psi_1 = 0.894$, $\psi_2 = 1.3409$, $\psi_3 = 1.5879$,
$\psi_4 = 2$, $\psi_5 = \frac{8}{3}$, $\psi_6 = \frac{10}{3}$

Configurations of a voxel of the boundary (at least one surfel belongs to a background voxel) :

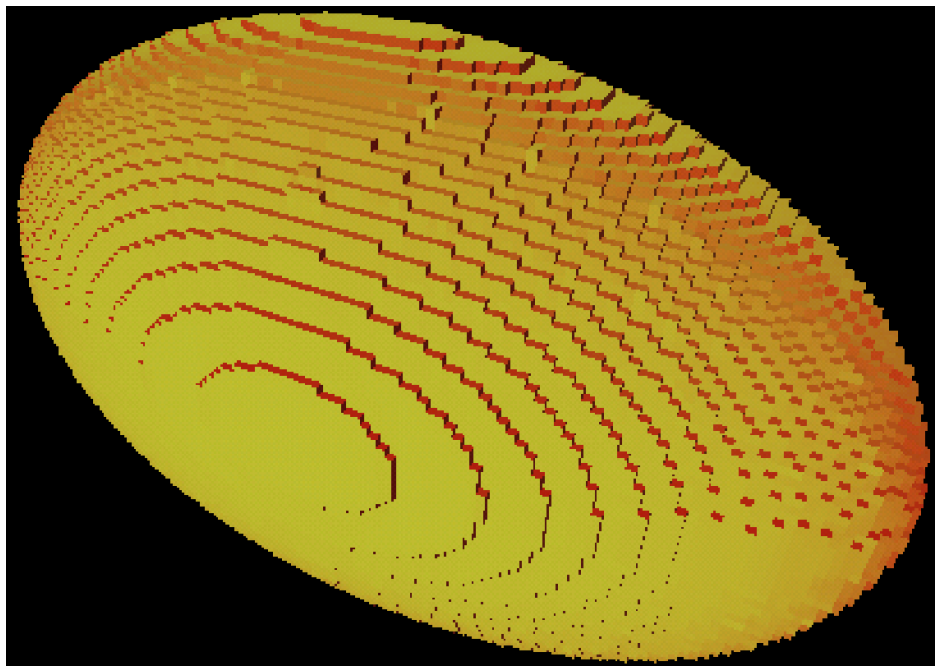


Surface area I

Normal integration

Surfel contribution : $\vec{n} \cdot \vec{e}$

$\vec{n}$: computed normal vector
$\vec{e}$: trivial normal vector



Curvature at a point of a surface I

Integral Invariants

Multigrid Convergent Principal Curvature Estimators in Digital Geometry - D. Coeurjolly, J.-O. Lachaud, J. Levallois - Computer Vision and Image Understanding, 2014

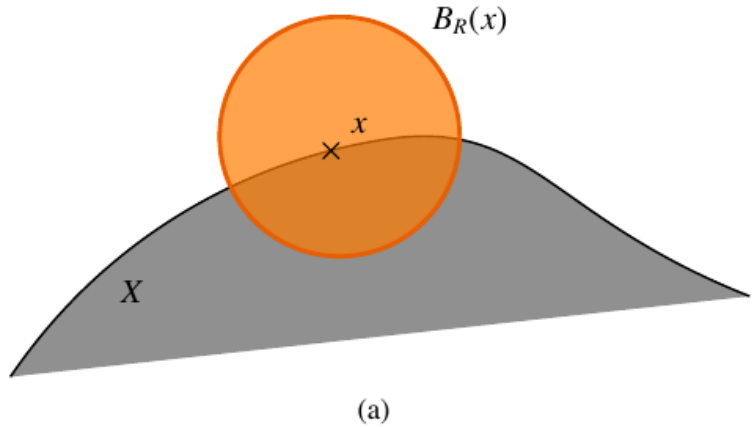
Continuous version :

$$V_R(x) = \int_{B_R(x)} \chi(p) dp$$

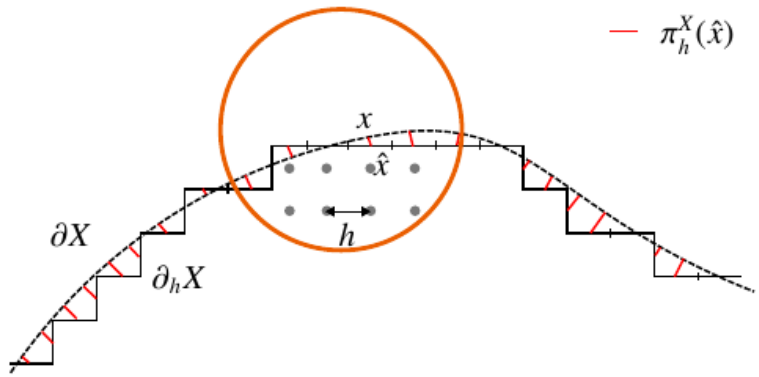
estimated mean curvature :

$$\hat{H}_R(X, x) = \frac{8}{3R} - \frac{4V_R(x)}{\pi R^4}$$

Simple digitization



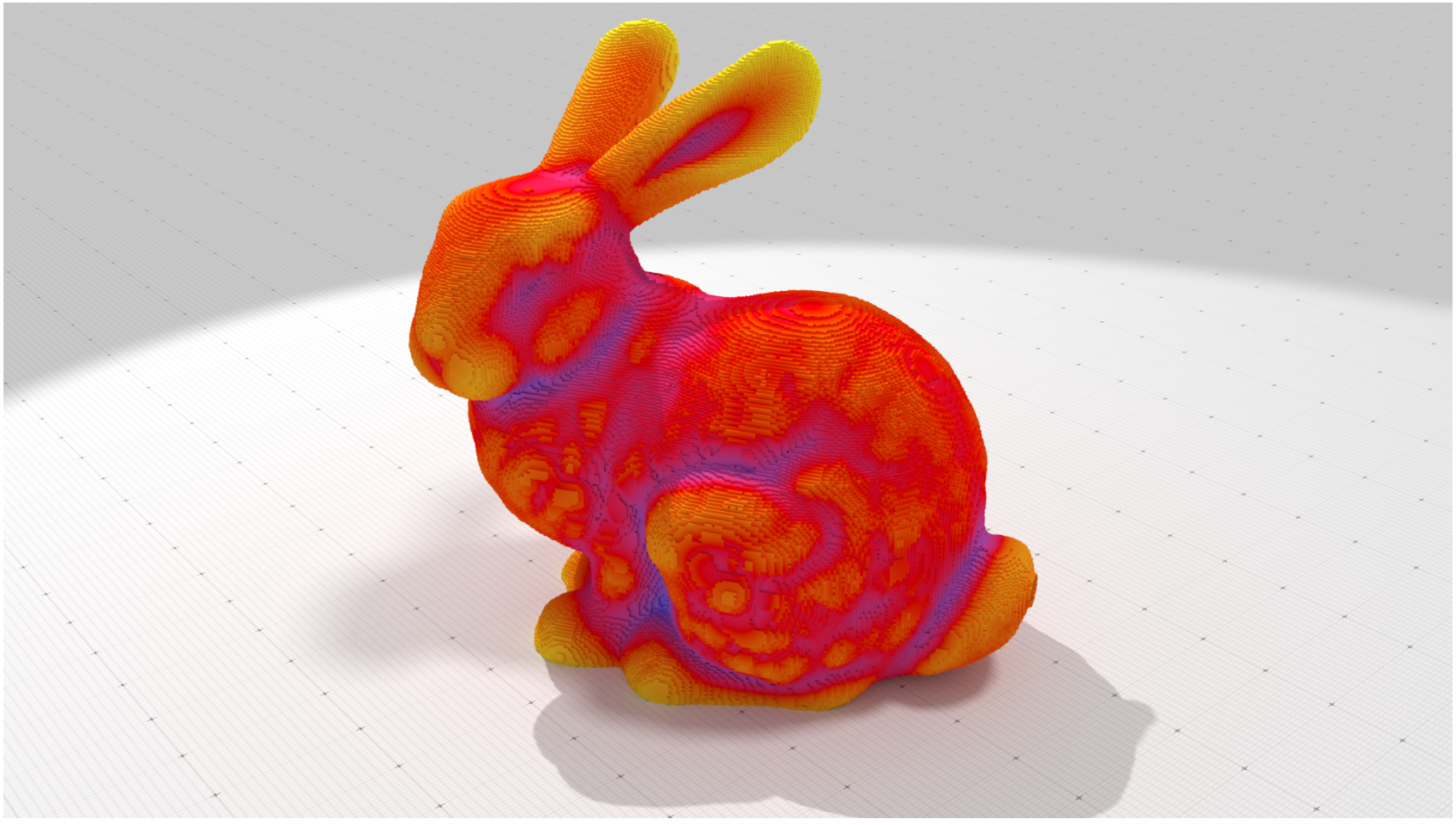
(a)



(b)

Curvature at a point of a surface II

Integral Invariants



Contents

Characterizing a shape

Here shape = 2D or 3D digital region

Defining features :

- robust to noise
- discriminative
- (invariant to geometric transforms)

$\rightarrow$ location, dimensions, orientation, shape descriptors, topological features, contour signatures...

Simple 2D geometric features

The region R is a set of pixels $(x_i, y_i)_{i=1..n}$

Area : number of pixels of R. the staircase effect along the contours does not distort the approximation.

Center of mass : $G = (\bar{x}, \bar{y}) = (\frac{1}{n}\sum_{i=1}^{n} x_i, \frac{1}{n}\sum_{i=1}^{n} y_i)$

Diameter : $D = max(\text{distance}(P, Q)/P, Q \in R)$

Circularity : $\dfrac{4\Pi\ \text{Area}(R)}{\text{Perimeter}^2(R)}$ *Elongation* : $\dfrac{\Pi L_g^2(R)}{4\text{Area}(R)}$

Convexity : $\dfrac{\text{Area}(R)}{\text{Area(convex hull(R))}}$

Simple 3D geometric features

The region R is a set of voxels $(x_i, y_i, z_i)_{i=1..n}$

Volume : number of voxels of R (if cubic voxels)

Center of mass :
$G = (\bar{x}, \bar{y}, \bar{z}) = (\frac{1}{n}\sum_{i=1}^{n} x_i, \frac{1}{n}\sum_{i=1}^{n} y_i, \frac{1}{n}\sum_{i=1}^{n} z_i)$

Sphericity : $\dfrac{36\Pi\ \text{Volume}^2(R)}{\text{Area}^3(R)}$

2D Cartesian moments

General definition : $m_{pq} = \sum_x \sum_y x^p y^q f(x,y)$

Binary region R : $f(x,y) = 1$ if $(x,y) \in R$ else $f(x,y) = 0$

$\Longrightarrow \; m_{pq} = \sum_{(x,y)\in R} x^p y^q$

Area : m_{00}

Center of mass : $(\bar{x}, \bar{y}) = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}})$

Centered moments : $\mu_{pq} = \sum_{(x,y)\in R} (x-\bar{x})^p (y-\bar{y})^q$

Normalized centered moments : $\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\frac{p+q}{2}+1}}$

Covariance matrix :

$$\frac{1}{m_{00}} \begin{pmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{pmatrix}$$

Features invariant to rotation

$\phi_1 = \eta_{20} + \eta_{02} \qquad \phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$

$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$

Principal axes in 2D I

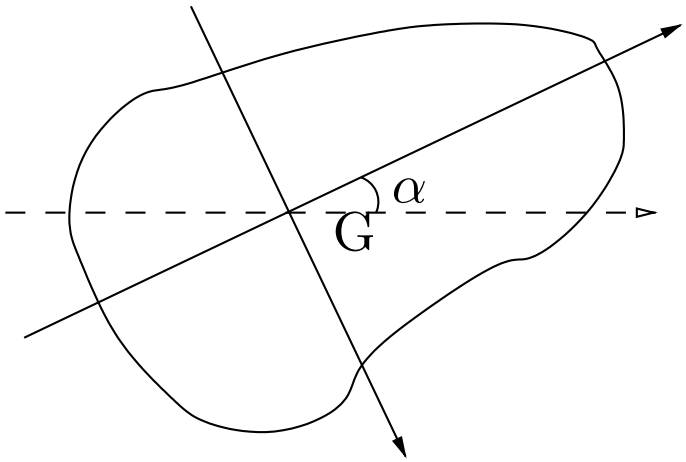
Principal axes (principal directions) : eigen vectors of the covariance matrix of the pixels.

Covariance matrix :

$$M_C = \begin{pmatrix} a & c \\ c & b \end{pmatrix} =$$
$$\begin{pmatrix} \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 & \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \\ \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) & \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2 \end{pmatrix}$$

M_C symmetric $\Rightarrow$ orthogonal eigen vectors

Principal axes in 2D II



$$tg(2\alpha) = \frac{2c}{a-b}$$

α angle between x axis and the first principal direction

$$a = (\frac{1}{n}\sum_{i=1}^{n} x_i^2) - \bar{x}^2, \quad b = (\frac{1}{n}\sum_{i=1}^{n} y_i^2) - \bar{y}^2, \quad c = (\frac{1}{n}\sum_{i=1}^{n} x_i y_i) - \bar{x}\bar{y}$$

Remark : a rectangular bounding box of R can be defined from the principal directions.

3D Cartesian moments

Definition (binary region) : $m_{pqr} = \sum_{(x,y,z)\in R} x^p y^q z^r$

Volume : m_{000}

Center of mass : $(\bar{x}, \bar{y}, \bar{z}) = (\frac{m_{100}}{m_{000}}, \frac{m_{010}}{m_{000}}, \frac{m_{001}}{m_{000}})$

Centered moments : $\mu_{pqr} = \sum_{(x,y,z)\in R} (x - \bar{x})^p (y - \bar{y})^q (z - \bar{z})^r$

Centered normalized moments : $\eta_{pqr} = \frac{\mu_{pqr}}{\mu_{000}^{\frac{p+q+r}{3}+1}}$

Covariance matrix : $\frac{1}{m_{000}} \begin{pmatrix} \mu_{200} & \mu_{110} & \mu_{101} \\ \mu_{110} & \mu_{020} & \mu_{011} \\ \mu_{101} & \mu_{011} & \mu_{002} \end{pmatrix}$

eigen vectors $\leftrightarrow$ principal axes of the shape.

Features invariant to rotation

see : *Geometric moment invariants, Dong Xu, Pattern Recognition 2008*

2D Zernike moments

$A_{nm} = \frac{n+1}{\Pi} \sum_x \sum_y f(x,y)[V_{nm}(x,y)]^*$ for $x^2 + y^2 \leq 1$,
n positive integer, m integer verifying $n - |m|$ even and $|m| \leq n$

$V_{nm}(x,y) = R_{nm}(x,y) e^{imtan^{-1}(\frac{y}{x})}$ basis of complex orthogonal polynomials

$R_{nm}(x,y) = \sum_{s=0}^{\frac{n-|m|}{2}} \frac{(-1)^s (x^2+y^2)^{\frac{n}{2}-s} (n-s)!}{s!(\frac{n+|m|}{2}-s)!(\frac{n-|m|}{2}-s)!}$ radial polynomial

$|A_{nm}|$ invariant to rotation : Let A'_{nm} be the Zernike moment computed after an image rotation of angle θ, $A'_{nm} = A_{nm} e^{-im\theta}$

To obtain invariance to translation and resizing, normalization with the Cartesian moments :
$h(x,y) = f(\frac{x}{a} + \bar{x}, \frac{y}{a} + \bar{y})$ with $a = \sqrt{\frac{\beta}{m_{00}}}$

Reconstruction : $f(x,y) = lim_{N \to \infty} \sum_{n=0}^{N} \sum_m A_{nm} V_{nm}(x,y)$

homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/SHUTLER3/node11.html