Coding cells of multidimensional digital spaces a framework to write generic digital topology algorithms

Jacques-Olivier Lachaud

LaBRI - Laboratoire Bordelais de Recherche en Informatique
Bordeaux, France

Outline

- Motivation
- Digital space representation
  - coding cells, adjacency, incidence
  - oriented cells, boundary operators
- Data structures for subsets of digital space
- Application to digital surface tracking
  - adjacency between boundary elements
  - tracking algorithms
  - benchmarks
- Conclusion and perspectives

Motivation

- Analyzing digital images (2D, 3D, more).
- Writing digital topology and geometry algorithms with application to discrete deformable models.
  - modelling sets of pixels and voxels and their boundaries.
  - tracking digital surfaces; visualizing them.
  - computing geometric characteristics.

Motivation

- Analyzing digital images (2D, 3D, more).
- Writing digital topology and geometry algorithms with application to discrete deformable models.
- *digital surface*: set of surface elements with topology.
- r*-cells* and *sets of* r*-cells* in n-dimensional space
  - How to represent them ?
  - How to compute their topology: neighborhood, incidences, boundary operators ?
  - How to get simple geometric characteristics: centroid, normals ?

Motivation

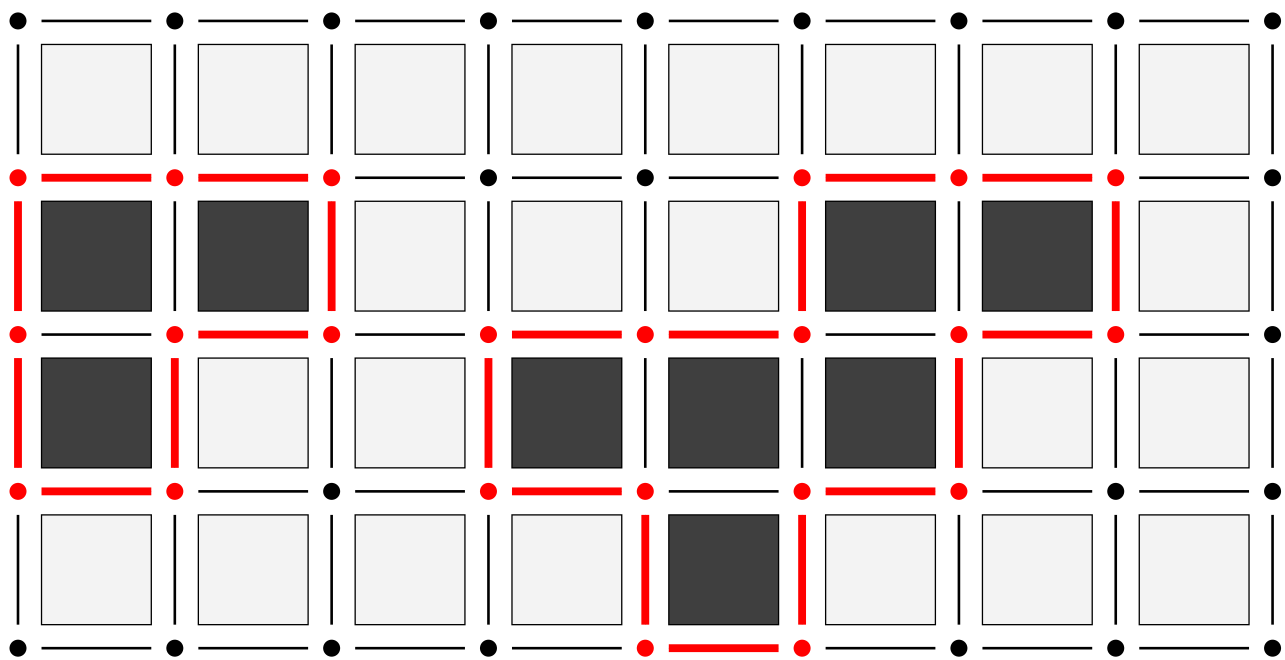
- Analyzing digital images (2D, 3D, more).
- Writing digital topology and geometry algorithms with application to discrete deformable models.
- *digital surface*: set of surface elements with topology.
- r*-cells* and *sets of* r*-cells* in n-dimensional space
- Objective: generic answer to digital cell representation.
  - independent of space dimension and of cell topology and dimension.
  - efficient in practice.

Digital space

- Main objective: analyzing digital images (2D, 3D, more)

$\Rightarrow$ finite regular space of dimension n and coordinate upper bounds M^i.

- *digital space $\mathbb{C}^n$*: cellular decomposition of $\mathbb{R}^n$ into a regular grid.



Digital space

- Main objective: analyzing digital images (2D, 3D, more)

$\Rightarrow$ finite regular space of dimension n and coordinate upper bounds M^i.

- *digital space* $\mathbb{C}^n$: cellular decomposition of $\mathbb{R}^n$ into a regular grid.
  - good topological properties for surfaces *[Kovalevsky89]*
  - geometric characteristics are always defined.
  - many high-level image representation on top of $\mathbb{C}^n$:
    - discrete maps *[Braquelaire, Brun, Desbarats, Domenger]* and *[Bertrand, Damiand, Fiorio]*,
    - cell lists *[Kovalevsky]*,
    - combinatorial pyramids *[Brun, Kropatsch]*.

Usual representations of cells (1/2)

- *Cells*:
  - pixels, voxels, *spels* in nD: static arrays of integer
  - surfels: pairs of adjacent spels *[Herman92]*
  - other cells: implicitly represented in algorithms
- *Set of cells*: characteristic function stored in an “image”; access through offset computation.

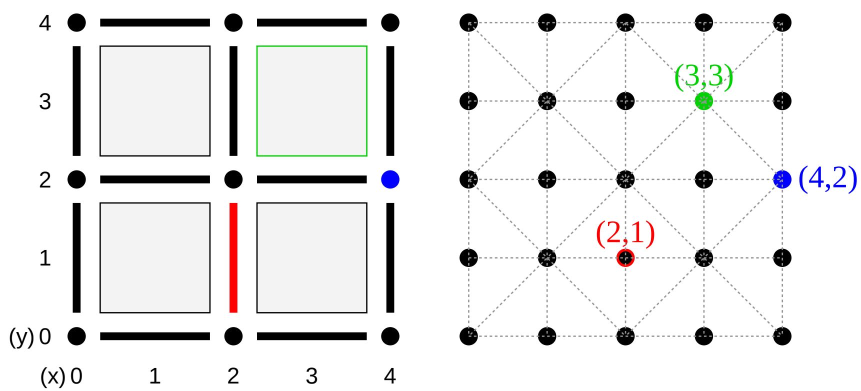
$\Rightarrow$ Very simple and easy to implement.

But non generic approach

  - dimension independence with dynamic allocation (≈ 10 times slower).
  - inhomogeneity between representations of n-cells, $n-1$-cells, etc.

Usual representations of cells (2/2)

- *Khalimsky space* $\mathbb{K}^n$: product of n COTS
  - $\mathbb{Z}$ alternating open and closed points: ●)(●)(●)(●
  - $\mathbb{K}^n$ and $\mathbb{C}^n$ are isomorph *[Kong,Khalimsky]*
  - any r*-cell*: n integer coordinates.
  - cell topology: parity of cell Khalismky coordinates
  - *Sets of cells*: Characteristic function stored in a doubled "image". Access through offset computation.



Usual representations of cells (2/2)

- *Khalimsky space* $\mathbb{K}^n$: product of n COTS

$\Rightarrow$ Homogeneous representation of cells

But
- same implementation problems with dynamic arrays
- memory cost of a set of r-cells is $2^n \prod (M^i + 1)$ bits.
- signed topology operators (upper and lower boundary) are cumbersome to write.

Proposed representation of cells

- any *r-cell* is coded as *one* integer number,
- all the topology (adjacency, incidence) and the geometry (centroid, normal) can be derived from the cell code,
- unoriented and oriented cells can be coded.

$\Rightarrow$ very compact representation of cells and of sets of cells

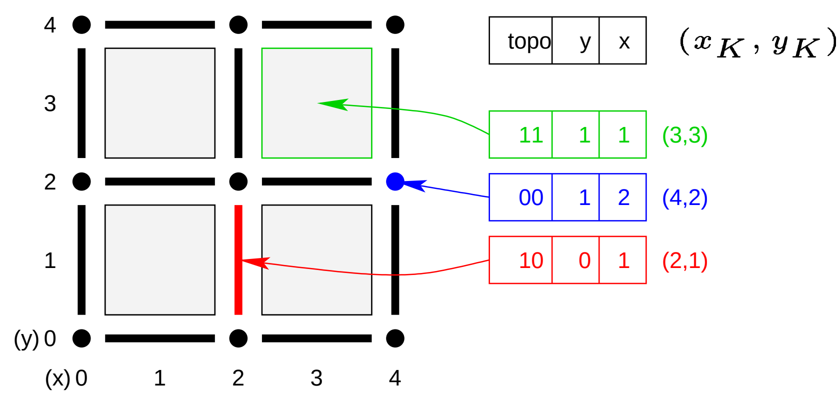
- generic: homogeneous representation that is independent of space dimension.
- efficient (e.g. one cpu register stores any cell)

Coding (unoriented) cells

- Any r-cell c is identified by its Khalimsky coordinates $(x^0_K, \ldots, x^{n-1}_K)$
- c is coded as one integer

α	x^{n-1}	$\ldots$	x^i	$\ldots$	x^0

  - digital coordinate $x^i = x^i_K \text{ div } 2$
  - each coordinate is binary coded on $N^i = \log_2(M_i) + 1$ bits
  - topology $\alpha = \sum_i (x^i_K \bmod 2) 2^i$



Coding (unoriented) cells

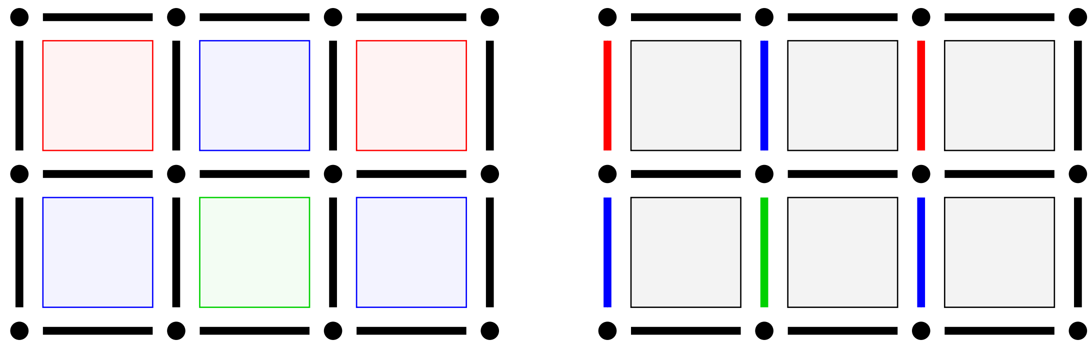
- Any r-cell c is identified by its Khalimsky coordinates $(x_K^0, \ldots, x_K^{n-1})$
- c is coded as one integer

α	x^{n-1}	$\ldots$	x^i	$\ldots$	x^0

- Elementary properties
  - topology of usual cells: spels $(1 \ldots 1)$, pointels $(0 \ldots 0)$, surfels $(1 \ldots 101 \ldots 1)$.
  - 32 bits code any cell of 32768×32768 2D image, $1024 \times 1024 \times 512$ 3D image, 128^4 4D image.
    $\Rightarrow$ enough for most biomedical applications
  - all elementary operations are made with maskings and shiftings, e.g. getting the cell topology or its i-th coordinate.

Topology operations: adjacency

- Two r-cells with same topology are *l-adjacent* iff their coordinates differ of ± 1 on l coordinates.



Cell, 1-adjacent cells, 2-adjacent cells

Topology operations: adjacency

- Two r-cells with same topology are *l-adjacent* iff their coordinates differ of ± 1 on l coordinates.
- Computation of 1-adjacent cells:

α	x^{n-1}	$\ldots$	x^i	$\ldots$	x^0

c // Cell

1-adjacent to

// Space is K

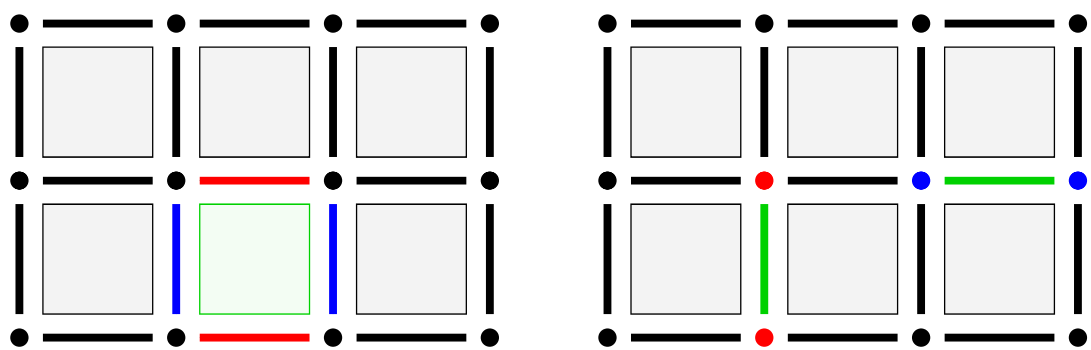
α	x^{n-1}	$\ldots$	$x^i - 1$	$\ldots$	x^0
α	x^{n-1}	$\ldots$	$x^i + 1$	$\ldots$	x^0

K.adjacent(c,i,NEG)

K.adjacent(c,i,POS)

Topology operations: low incidence

- The *low incidence* is the face relation. The 1-low incidence defines the $r-1$-cells that are faces of a r-cell.



Cell, 1-low incident cells along x, 1-low incident cells along y

- Prop. Any r-cell has two 1-low incident $r-1$-cells along each coordinate where the cell is open.

Topology operations: low incidence

- The *low incidence* is the face relation. The 1-low incidence defines the $r-1$-cells that are faces of a r-cell.
- Computation of 1-low incidence cells:

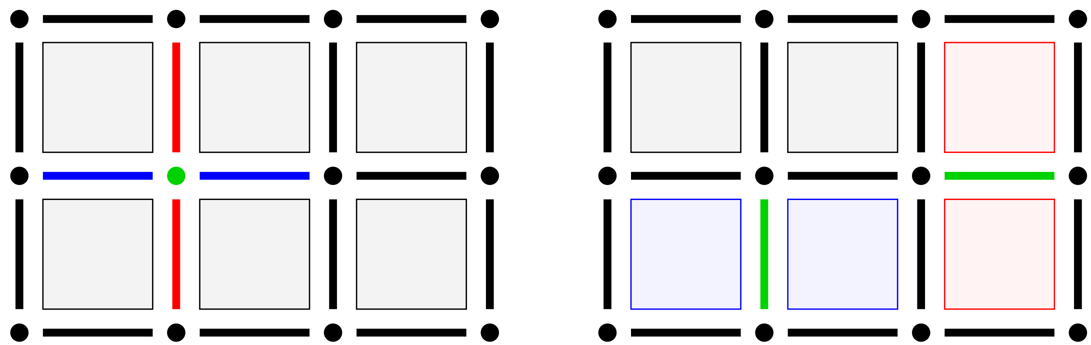
$\dots 1 \dots$	x^{n-1}	$\dots$	x^i	$\dots$	x^0	c

has two 1-low incident cells // Space is K

$\dots 0 \dots$	x^{n-1}	$\dots$	x^i	$\dots$	x^0	K.lowIncident(c,i,NEG)
$\dots 0 \dots$	x^{n-1}	$\dots$	x^i+1	$\dots$	x^0	K.lowIncident(c,i,POS)

Topology operations: up incidence

- The *up incidence* is the coface relation. The 1-up incidence defines the $r+1$-cells that are cofaces of a r-cell.



Cell, 1-up incident cells along x, 1-up incident cells along y

- Prop. Any r-cell has two 1-up incident $r+1$-cells along each coordinate where the cell is closed.

Topology operations: up incidence

- The *up incidence* is the coface relation. The 1-up incidence defines the $r+1$-cells that are cofaces of a r-cell.
- Computation of 1-up incidental cells:

$\ldots 0 \ldots$	x^{n-1}	$\ldots$	x^i	$\ldots$	x^0	c

has two 1-up incident cells // Space is K

$\ldots 1 \ldots$	x^{n-1}	$\ldots$	$x^i - 1$	$\ldots$	x^0	K.upIncident(c,i,NEG)
$\ldots 1 \ldots$	x^{n-1}	$\ldots$	x^i	$\ldots$	x^0	K.upIncident(c,i,POS)

Cost of elementary cell operations

nb ops required	code	topo, coord	==	set coord	adj.	is l-adj.?	inc.	is l-inc.?
bits ops	0	1	0	2	0	$\leq 2n$	1	≤ 3
shifts	n	1	0	1	0	0	0	≤ 6
integer ops	n	0	1	0	1	$\leq 2n$	≤ 1	$\leq l + 4$
lut access	n	1	0	2	1	$\leq n$	≤ 2	$\leq l + 2$
cond. tests	0	0	0	0	1	$\leq 2n$	1	$\leq 3l + 1$

- the dimension n is generally low, and $l \leq n$.
- all these operations on cell codes compete with or are faster than the same operations on cells represented as integer arrays.

Coding oriented cells

- motivation for orienting cells
  - useful for defining digital surfaces, cubical cell complexes and boundary operators.
  - first step to r*-dimensional chains*
  - necessary for giving a local orientation to cells and for defining consistent adjacencies between surfel elements.

Coding oriented cells

- motivation for orienting cells
- Code of an *oriented cell*:

α	s	x^{n-1}	$\ldots$	x^i	$\ldots$	x^0

  with *orientation* s (0 positive, 1 negative)

$\Rightarrow$ most elementary operations are similar.

Coding oriented cells

- motivation for orienting cells
- Code of an *oriented cell*: $\begin{array}{|c|c|c|c|c|c|c|}\hline \alpha & s & x^{n-1} & \ldots & x^{i} & \ldots & x^{0} \\ \hline \end{array}$
- *Boundary operator* Δ: signed 1-low incidence

  if $c = \begin{array}{|c|c|c|c|c|c|c|}\hline i_k \ldots i_j \ldots i_0 & s & x^{n-1} & \ldots & x^{i_j} & \ldots & x^{0} \\ \hline \end{array}$

  $\Delta_{i_j} c =$

  $$\left\{ \begin{array}{l} \begin{array}{|c|c|c|c|c|c|c|}\hline i_k \ldots \overline{i_j} \ldots i_0 & (-1)^{k-j} s & x^{n-1} & \ldots & x^{i_j} & \ldots & x^{0} \\ \hline \end{array}, \\ \begin{array}{|c|c|c|c|c|c|c|}\hline i_k \ldots \overline{i_j} \ldots i_0 & (-1)^{k-j+1} s & x^{n-1} & \ldots & x^{i_j}+1 & \ldots & x^{0} \\ \hline \end{array} \end{array} \right\}$$

  and $\Delta c = \cup_{j=0}^{k} \Delta_{i_j} c$.

  - Prop. If R is a set of oriented r-cells, then $\Delta\Delta R = 0$

  $\Rightarrow$ Any boundary has no boundary (ie. is closed).

Coding oriented cells

- motivation for orienting cells
- Code of an *oriented cell*:

α	s	x^{n-1}	$\dots$	x^i	$\dots$	x^0

- *Boundary operator* $\triangle$: signed 1-low incidence
- *Coboundary operator* ∇: signed 1-up incidence
  - Prop. If R is a set of oriented r-cells, then $\nabla\nabla R = 0$

$\Rightarrow$ Any coboundary has no coboundary (ie. is open).

Oriented cells and boundaries

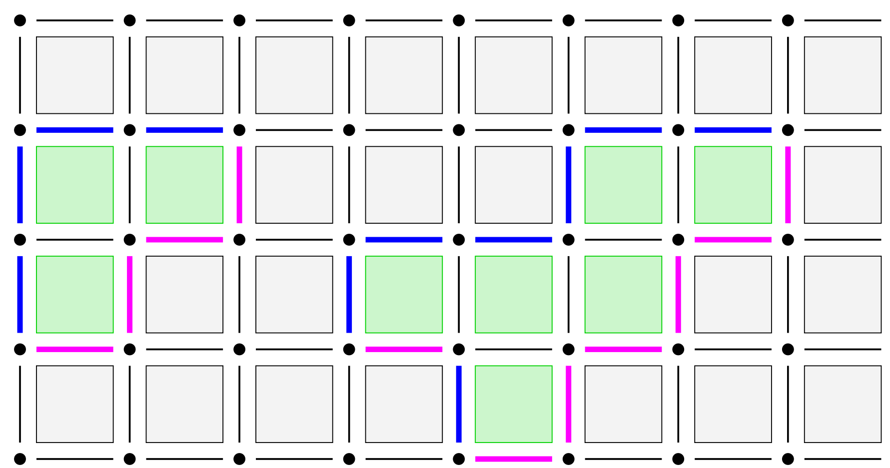
- Def. If a region O is viewed as a set of positively oriented spels, then the boundary of O is ΔO. Any path from O to its complement crosses ΔO.

$\Rightarrow$ a simple scanning extracts the boundary of a region.

- Prop. For any surfel $s \in \Delta O$, $\nabla s = \{+p, -q\}$ with $p \in O$ and $q \notin O$.

$\Rightarrow$ gives locally the inside and outside of a surface.

set of spels O,
positive surfels in ΔO,
negative surfels in ΔO



Data structures for sets in $\mathbb{C}^n$

- classical data structures for small sets

set data struct.	Is in set ?	Set operations	Memory cost (bytes)
dynamic array	$O(m)$	$O(m)+$	$\approx 4m$
linked list	$O(m)$	$O(m)$	$\approx 24m$
RB-tree	$O(\log m)+$	$O(\log m)+$	$\approx 3m$
hashtable	$O(1)+$	$O(1)+$	$\approx 4m' + 20m,$ $m' \gg m$

- rather costly for big sets: e.g. 1.000.000 surfels $\Rightarrow$ requires 28Mb in a hashtable with $m' = 2m$.

Data structures for sets in $\mathbb{C}^n$

- classical data structures for small sets
- characteristic function for big sets of r-cells.
  - code of a cell c gives an offset in array of bits

$$\text{offset}(c) = \text{LUT}[\text{topology}(c)] + \text{coords}(c)$$

  - with LUT:

0 . . . 0011 . . . 1	0	0 . . . 0
1 . . . 0101 . . . 1	1	0 . . . 0
. . .	. . .	. . .
1 . . . 1100 . . . 0	$\binom{n}{r} - 1$	0 . . . 0

$\Rightarrow$ Offset = one LUT access + one masking + one addition

Data structures for sets in $\mathbb{C}^n$

- classical data structures for small sets
- characteristic function for big sets of r-cells.
  - set of r-cells: $\binom{n}{r}2^{\sum N_i}$ bits (e.g set of surfels in a 256^3 image holds ≈ 50 million surfels with 6Mb).
  - set of oriented r-cells are twice bigger.
  - all set operations are in O(1).
  - computation time of the difference of two sets of surfels in a 512^3 image: 2.5s or 6ns / surfel (Celeron 450Mhz).

Data structures for sets in $\mathbb{C}^n$

- classical data structures for small sets
- characteristic function for big sets of r-cells.
- memory comparison with other cell representations (with space sizes 2^{N^i} and $N = \sum N^i$).

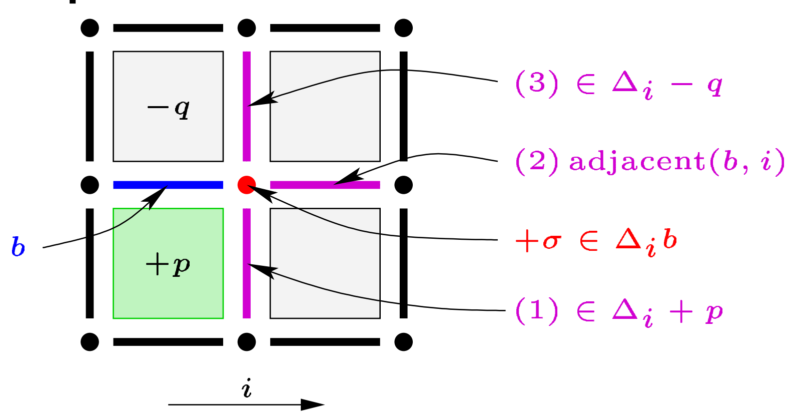
cell rep.	r-cell (int. words)	set of n-cells (bits)	set of r-cells (bits)	genericity
classical rep.	$\geq n$	2^N	no	no
Khalimsky rep.	n	$2^n 2^N$	$2^n 2^N$	dyn. alloc.
proposed rep.	1	2^N	$\binom{n}{r} 2^N$	yes

Application: digital surface tracking in $\mathbb{C}^n$

- Problem: Given an oriented surfel s and a set of spels O with $s \in \Delta O$, find the whole *connected* component $C(\Delta O, s)$ of ΔO that contains s by tracking *adjacent* element of ΔO.
  - the algorithm should be linear with respect to the number of surfels of $C(\Delta O, s)$.
- an adjacency relation must be defined between surfels of ΔO (or *bels*).
  - $2^{\frac{n(n-1)}{2}}$ different *bel adjacencies*.
  - two of them corresponds to the classical $(2n, 2n^2)$ and $(2n^2, 2n)$ bel adjacencies *[Udupa94]*.

Bel adjacency in ΔO (1/2)

1. Def. A *direct follower* of an oriented r-cell b^r is any r-cell $c^r \neq b^r$ such that $\exists$ $r-1$-cell σ^{r-1} with $+\sigma^{r-1} \in \Delta b^r$ and $-\sigma^{r-1} \in \Delta c^r$. The cell b^r is an *indirect follower* of c^r.
2. Prop. Any surfel (or $n-1$-cell) b has 3 direct and 3 indirect followers along each coordinate where it is open.

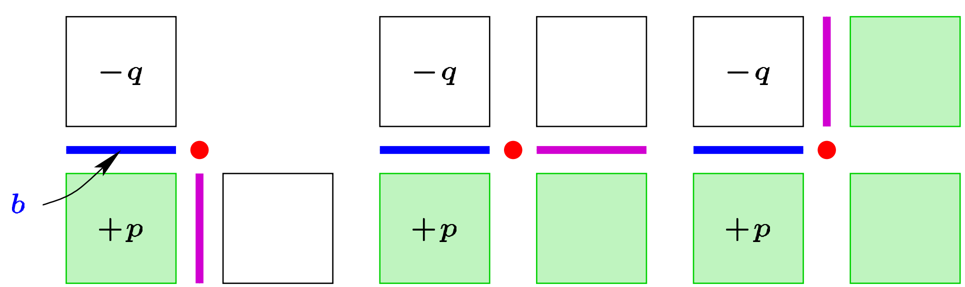


$$\nabla b = \{+p, -q\}$$

followers are ordered

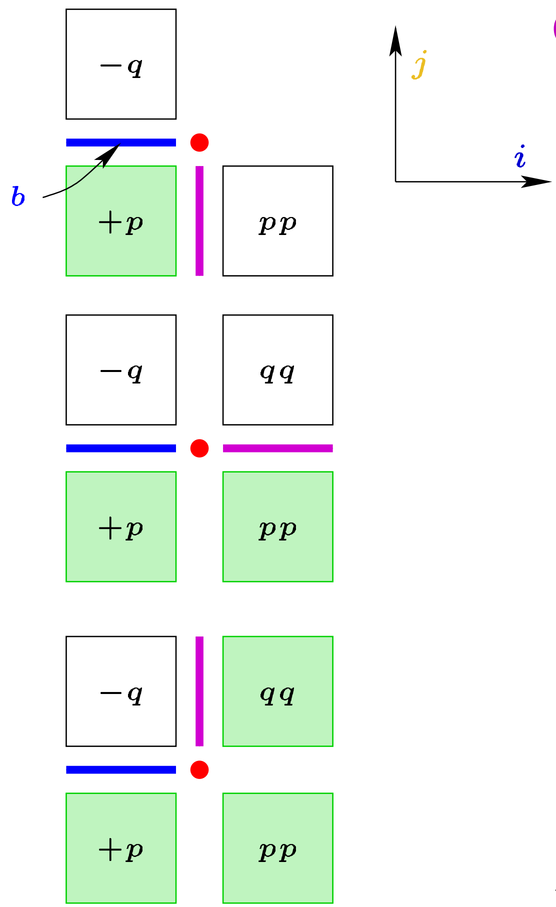
Bel adjacency in ΔO (1/2)

1. Def. A *direct follower* of an oriented r-cell b^r is any r-cell $c^r \neq b^r$ such that $\exists\ r-1$-cell σ^{r-1} with $+\sigma^{r-1} \in \Delta b^r$ and $-\sigma^{r-1} \in \Delta c^r$. The cell b^r is an *indirect follower* of c^r.
2. Prop. Any surfel (or $n-1$-cell) b has 3 direct and 3 indirect followers along each coordinate where it is open.
3. Def. The direct interior (resp. exterior) adjacent bel to b $\in \Delta O$ along coordinate i is the first (resp. last) of the direct followers of b that is $\in \Delta O$.



Bel adjacency in ΔO (2/2)

- Example of direct interior adjacent bel computation.




```
Cell Space::DIAdjBel( Set O, Cell b, int i ) {
    // Extract p and q
    int j = orthDir( b );
    bool orth = direct( b, j );
    Cell p = unsign( incident( b, j, orth ) );
    Cell q = adjacent( p, j, !orth );
    // Extract pp
    bool track = direct( b, i );
    Cell pp = adjacent( p, i, track );
    // Check if first follower ∈ ΔO
    if ( O.isInSet( pp ) )
        return incident( pos( pp ), i, track );
    // Extract qq
    Cell qq = adjacent( q, i, track );
    // Check if second follower ∈ ΔO
    if ( ! O.isInSet( qq ) )
        return adjacent( b, i, track );
    // if not, last follower ∈ ΔO
    return incident( neg( qq ), i, track );
}
```

Bel adjacency in ΔO (2/2)

- Example of direct interior adjacent bel computation.
- Prop. Any bel of ΔO has $n-1$ direct interior (resp. exterior) adjacent bels in ΔO and $n-1$ indirect interior (res.p exterior) adjacent bels.
- A *bel adjacency relation* in ΔO is given by fixing for each coordinate couple (i,j), $0 \leq i < j < n$, whether the bel adjacency is interior or exterior.

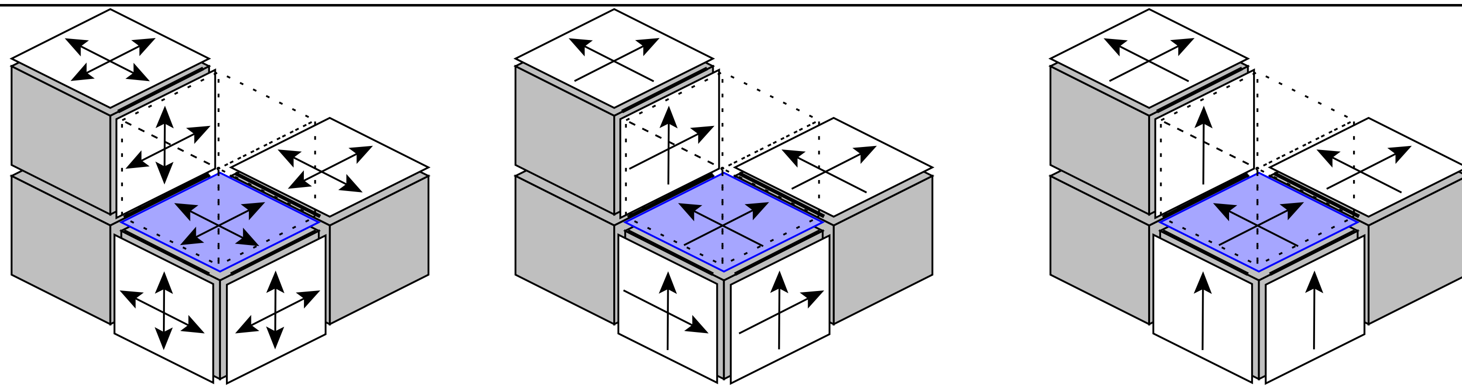
  $\Rightarrow$ they are $\frac{n(n-1)}{2}$ different bel adjacencies.

  $\Rightarrow$ connectedness relations on ΔO.
- From $\Delta\Delta O = 0$ and definition of followers, it is easy to find that tracking along only *direct* adjacent bels is sufficient to get the whole connected component of ΔO containing the starting bel.

Digital (hyper)surface tracking

- Tracking algorithms

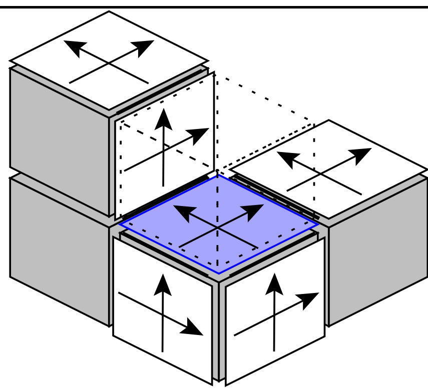
(A)	(B)	(C)
direct and indirect bel adj. tracking	only direct bel adj. tracking	*[Herman,Webster83]* tracking (3D)



- easily implemented with the proposed framework (both nD and 3D algorithms).

Digital (hyper)surface tracking

(B)
only direct
bel adj. tracking



```
Set Space::directTracking( Set O, Cell b, BelAdj A ) {
    Set S = emptySurfelSet(); // Ouput
    Queue Q; // Cells to process
    Q.push( b ); // init tracking
    S.add( b );
    while ( ! Q.empty() ) {
        Cell s = Q.pop(); // current Cell
        for ( i = 0; i < dim(); i++ )
            if ( i != orthDir( s ) ) {
                // Get direct adjacent bel along i
                Cell n = A.directAdj( O, s, i );
                if ( ! S.isInSet( n ) ) {
                    S.add( n );
                    Q.push( n );
                }
            }
    }
    return S;
}
```

Benchmarks of boundary extraction algos

- Experimental results: balls of increasing radius (Celeron 450 Mhz)

Space	Rad.	Nb spels	Nb surf.	Scan (A)	Track (A)	Track (B)
4096^2	2000	12566345	16004	2.07s	$<$ 0.01s	$<$ 0.01s
256^3	30	113081	16926	3.12s	0.02s	0.01s
256^3	60	904089	67734	3.12s	0.09s	0.08s
256^3	120	7236577	271350	3.15s	0.36s	0.32s
512^3	240	57902533	1085502	25.15s	1.88s	1.85s
32^4	14	190121	92104	0.26s	0.14s	0.11s
64^4	14	190121	92104	4.17s	0.20s	0.14s
64^4	30	4000425	904648	4.26s	1.91s	1.37s

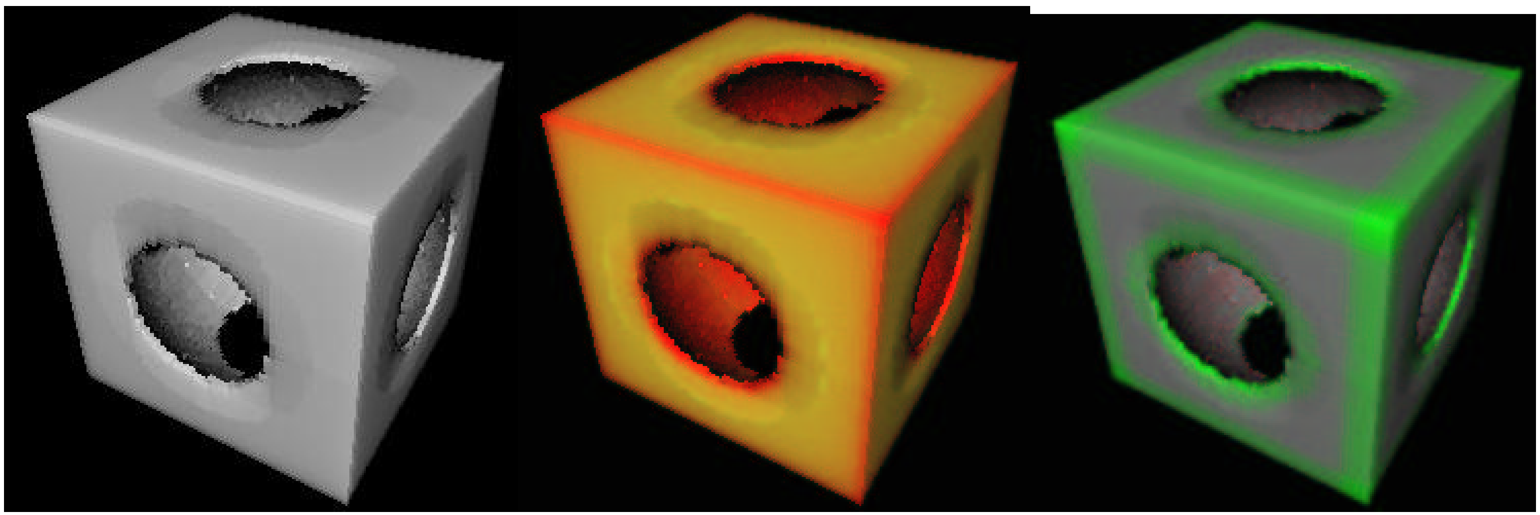
  - scanning linear with number of surfels of space (in nD, 62ns/spel)
  - tracking linear with number of bels of boundary (in nD, 1.5μs/bel)

Conclusion

- generic framework to represent cells and subsets of digital spaces and to write digital topology algorithms
  - unoriented and oriented r-cells
  - compact sets of r-cells
  - boundary operators (topology invariants)
  - bel adjacency on boundaries
- proposed framework fully implemented in an object oriented language
  - formal algorithms close to implementation
- tests have shown its efficiency and scalability

Perspectives

- check extension of Herman and Webster 3D digital surface algorithm to nD spaces
- computing n-dimensional geometric characteristics



- coding cells of hierarchical digital spaces