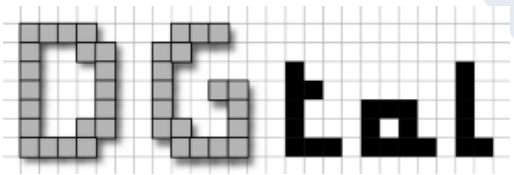
Digital surfaces in DGtal
Topology module (since 0.5)

Jacques-Olivier Lachaud

DGtal Meeting, june 2012

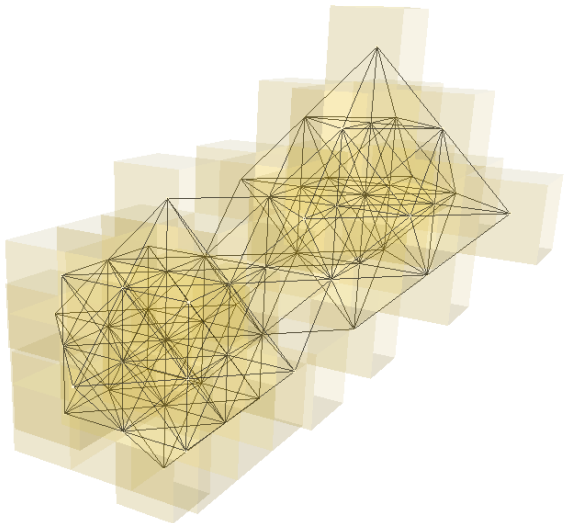


UMR 5127

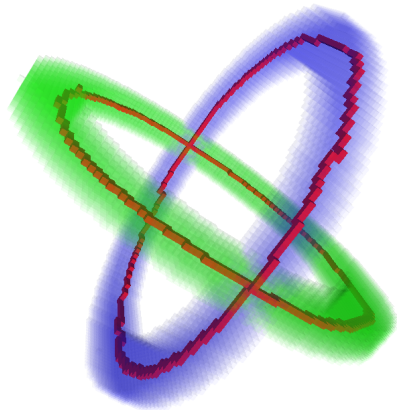
Package Topology, available in DGtal 0.4

1. classical digital topology (*à la Rosenfeld*)

   - Arbitrary adjacencies in $\mathbb{Z}^n$, but also in subdomains
   - Digital topology = couple of adjacencies (Rosenfeld)
   - Object = Topology + Set
   - Operations : neighborhoods, border, connectedness and connected components, decomposition into digital layers, simple points



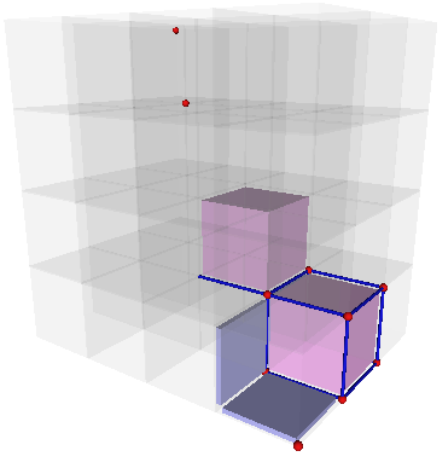
Adjacencies

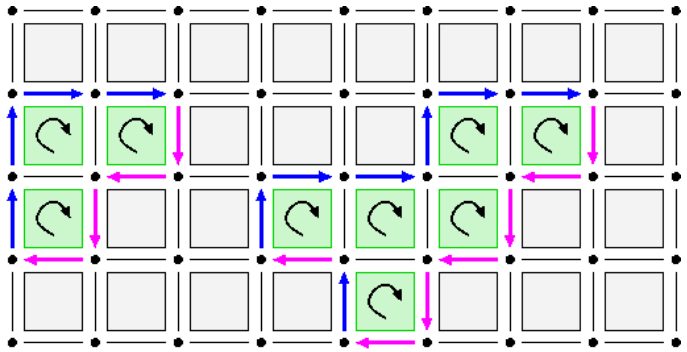


thinning in (6,26)

Package Topology, available in DGtal 0.4

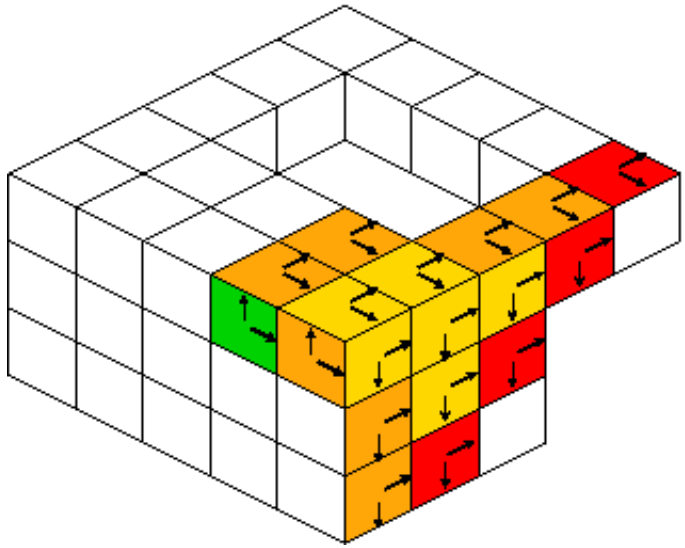
1. classical digital topology (*à la Rosenfeld*)
2. cubical cellular topology + algebraic topology
   - cells, adjacent and incident cells, faces and cofaces
   - signed cells, signed incidence, boundary operators

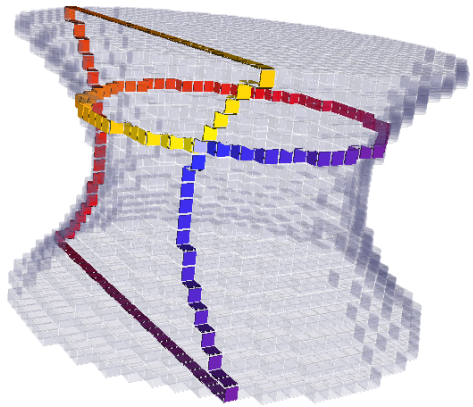




Package Topology, available in DGtal 0.4

1. classical digital topology (*à la Rosenfeld*)
2. cubical cellular topology + algebraic topology
3. digital surface topology (*à la Herman*)
   - surfels, surfel adjacency, surfel neighborhood
   - surface tracking (normal, fast), contour tracking in *n*D



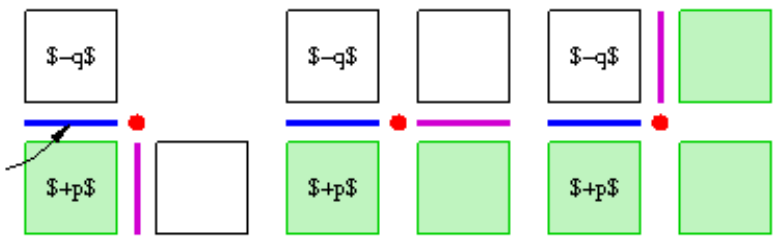


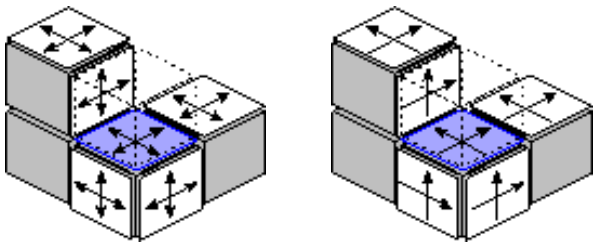
Package Topology, new in DGtal 0.5

Digital Surface

surfels / signed $n-1$-cells
+ adjacencies between surfels

- kind of "dual" graph
- kind of manifold





Package Topology, new in DGtal 0.5

Digital Surface

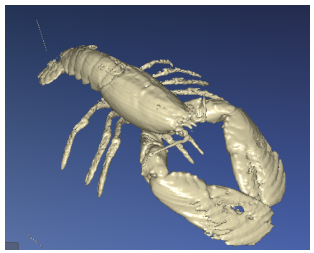
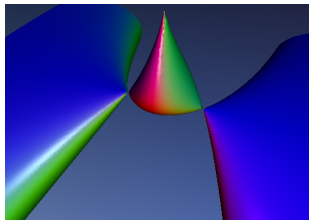
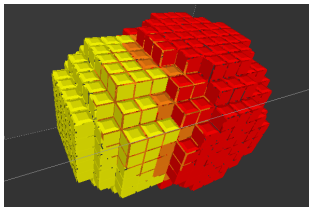
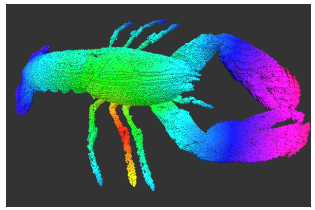
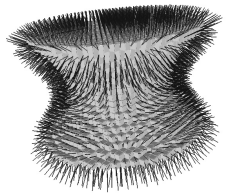
surfels / signed $n-1$-cells + adjacencies between surfels

- kind of "dual" graph
- kind of manifold

1. High-level `DigitalSurface` class for representing any kind of digital surface
2. Many container classes for digital surfaces
   - boundary of digital shape
   - boundary of implicitly defined shape
   - set of surfels
   - implicitly defined set of surfels
   - light containers
3. a `DigitalSurface` is a graph
4. a `DigitalSurface` is a combinatorial surface (with umbrellas)

Direct applications

- marching cubes algorithm
- tracking implicit polynomial surfaces
- representing boundary of regions and frontier between regions
- breadth-first visiting on surfaces
- estimating normals on surfaces

Necessary concepts and classes for digital surfaces

One must choose

- the representation of cellular grid space : model of CCellularGridSpaceND
  e.g. KhalimskySpaceND< N, int >, Z2i::KSpace, Z3i::KSpace
- the kind of adjacency between surfels, SurfelAdjacency< N >
- the kind of surface container : model of CDigitalSurfaceContainer

```
1  typedef Z3i::Point Point; // 3D digital point
2  typedef Z3i::Domain Domain;
3  typedef Z3i::DigitalSet DigitalSet; // a set of
       digital points
4  typedef Z3i::KSpace KSpace; // 3D cellular grid space
5  typedef SurfelAdjacency<3> SAdj; // surfel adjacency.
6  typedef DigitalSetBoundary<KSpace,DigitalSet>
       Container; // kind of surface container
7  typedef DigitalSurface<Container> MyDigSurf; //
       concrete digital surface
```

Concrete instanciations for digital surfaces

Then, the chosen types are instantiated. Here
digital surface = boundary of two intersecting balls

```
1    Point p1( -20, -20, -20 ), p2( 20, 20, 20 );
2    KSpace K; K.init( p1, p2, true ); // init space
3    DigitalSet someShape( Domain( p1, p2 ) );
4    Shapes<Domain>::addNorm2Ball( someShape, Point
         (-3,0,0), 4 );
5    Shapes<Domain>::addNorm2Ball( someShape, Point
         (3,0,0), 4 );
6    SAdj surfAdj( true );        // the adjacency
7    Container surfContainer( K, someShape, surfAdj );
8    MyDigSurf digSurf( surfContainer ); // digital
         surface
```

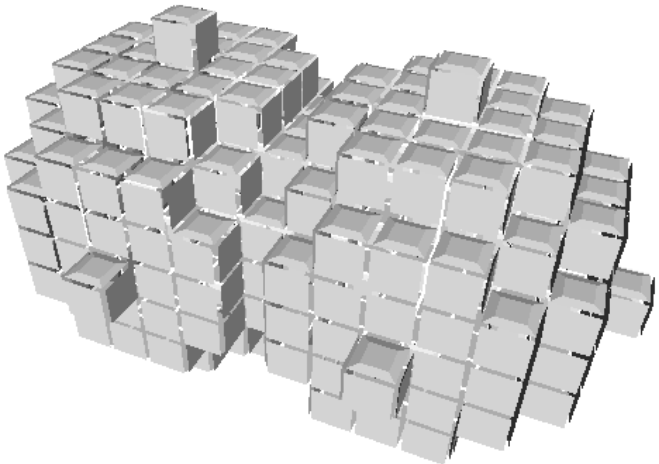
Using the digital surface (displays 518) :

```
1    cout << "- nb surfels/vertices = "
2         << digSurf.size() << endl;
```

How to use digital surfaces (I)

Just enumerating its elements...

```
1    QApplication application( argc, argv );
2    Viewer3D viewer; // QGL viewer
3    viewer.show();
4    for( MyDigSurf::ConstIterator it = digSurf.begin(),
5     itend = digSurf.end(); it != itend; ++it )
6      viewer << *it;
7    viewer << Viewer3D::updateDisplay;
8    return application.exec();
```



Package description

Should contain

- classical digital topology $\tilde{A}$ *la* Rosenfeld
- cartesian cellular topology
- digital surface topology $\tilde{A}$ *la* Herman
- must be the base block of geometric algorithms

Examples

- adjacencies, connected components, simple points, thinning
- cells, boundary operators, incidence, opening, closing
- contours, surfel adjacency, surface tracking
- topological invariants

Location

- `{DGtal}/src/DGtal/topology`
- `{DGtal}/src/DGtal/helpers`
- `{DGtal}/tests/topology`