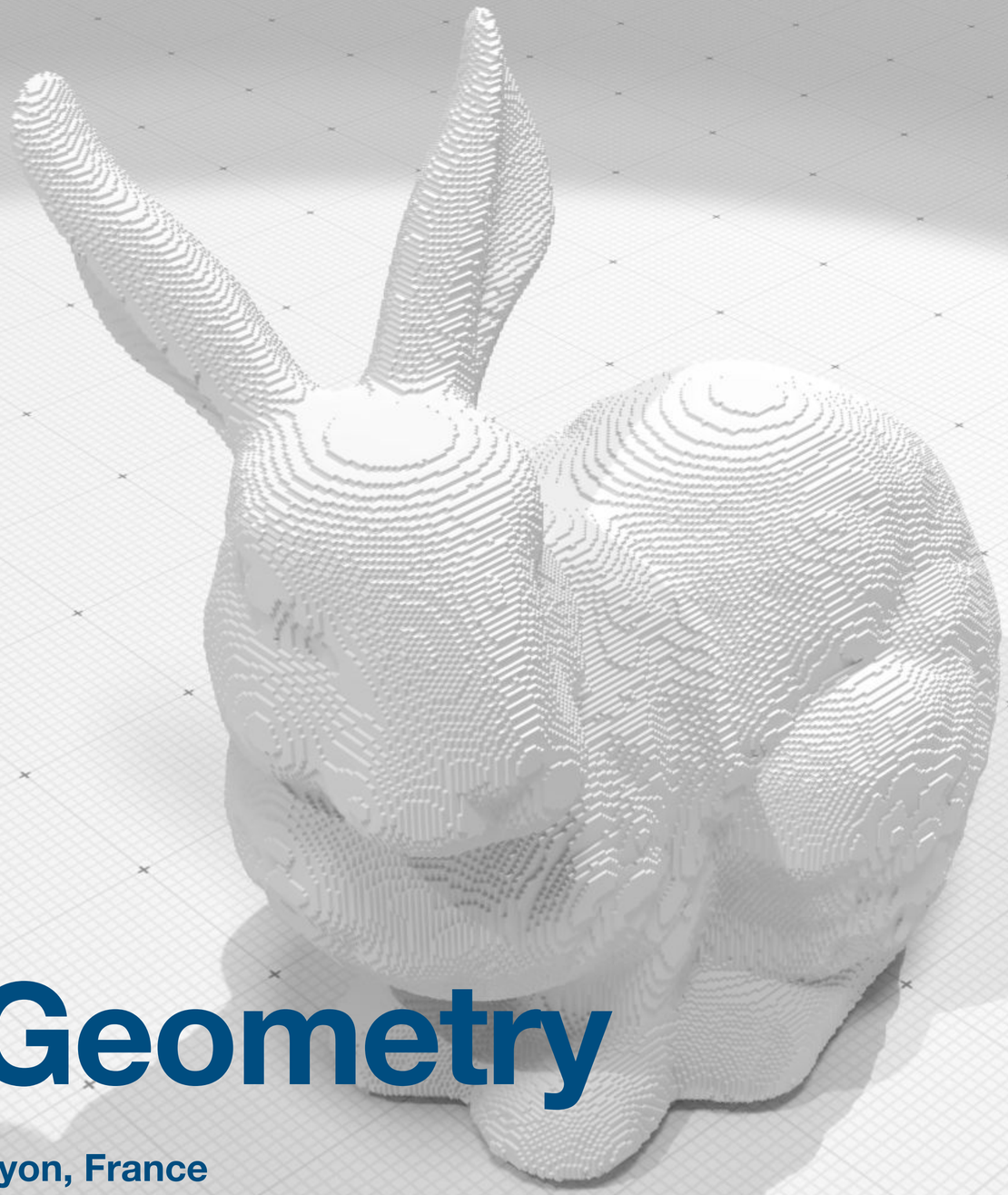


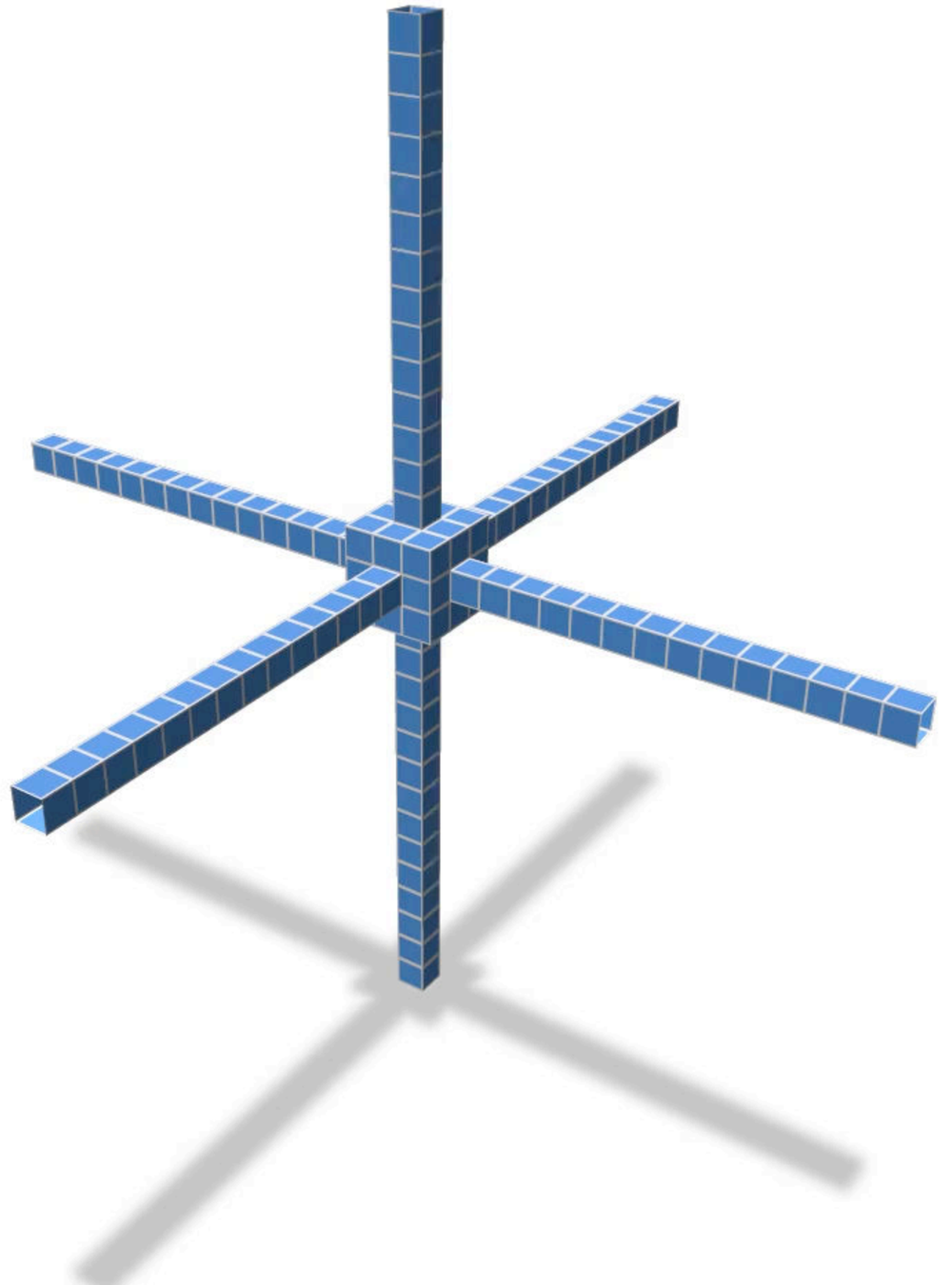
Digital Geometry

David Coeurjolly, CNRS, Lyon, France

Jacques-Olivier Lachaud, Université Savoie Mont-Blanc, France

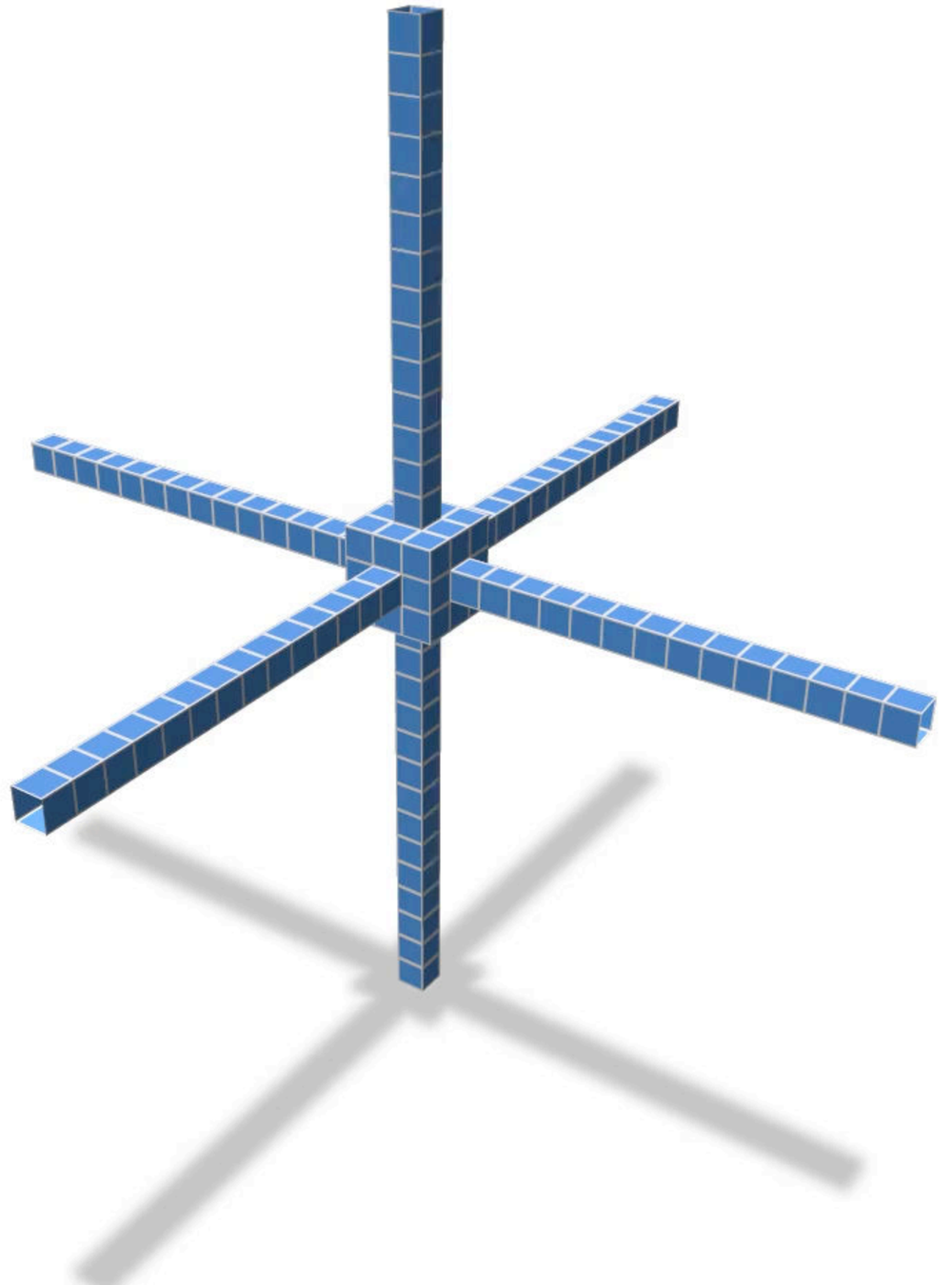
Outline

- context
- dgtal.org
- geometry with integers
- geometry processing on grids
- digital surface processing
- conclusion



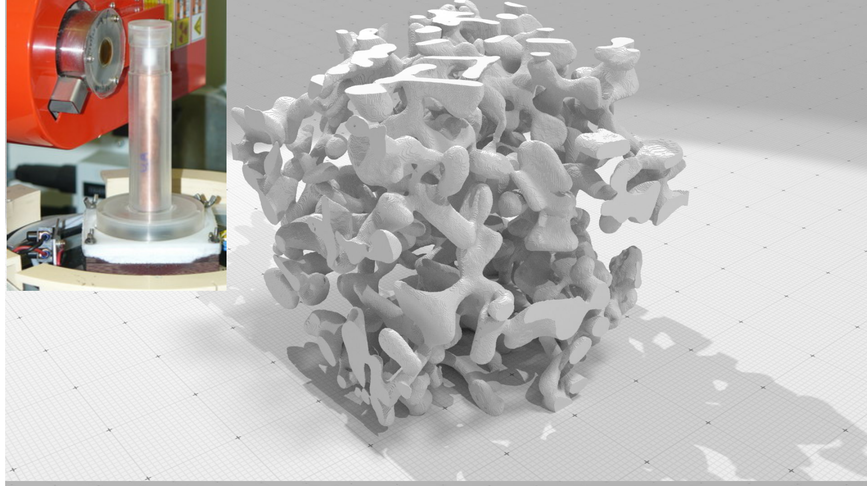
Outline

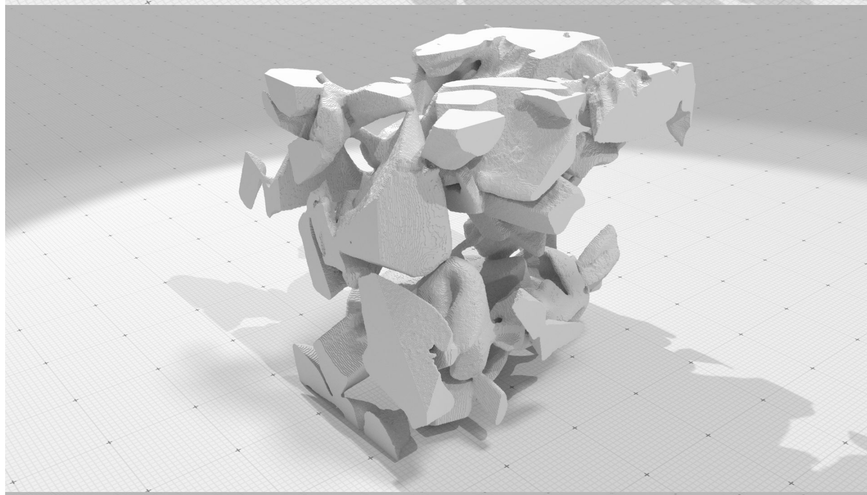
- context
- dgtal.org
- geometry with integers
- geometry processing on grids
- digital surface processing
- conclusion

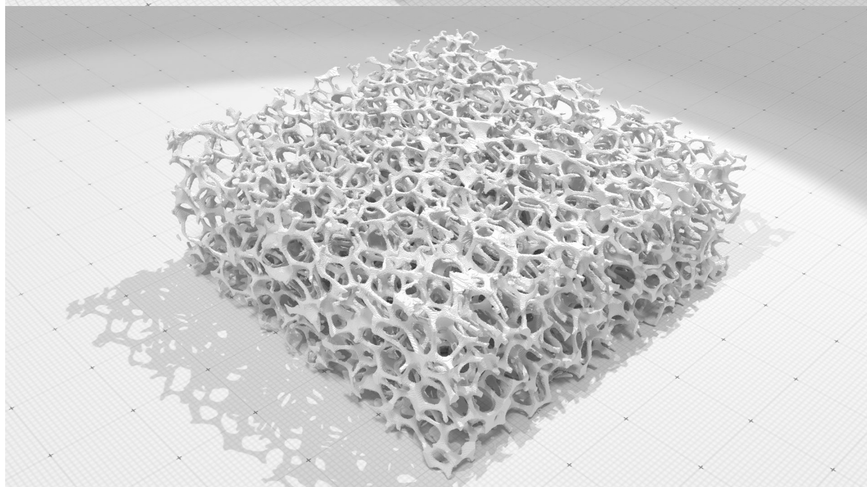


Motivations (1): devices

- Micro-tomographic images
  - material sciences
  - medical images
- Process geometry/topology of images partitions





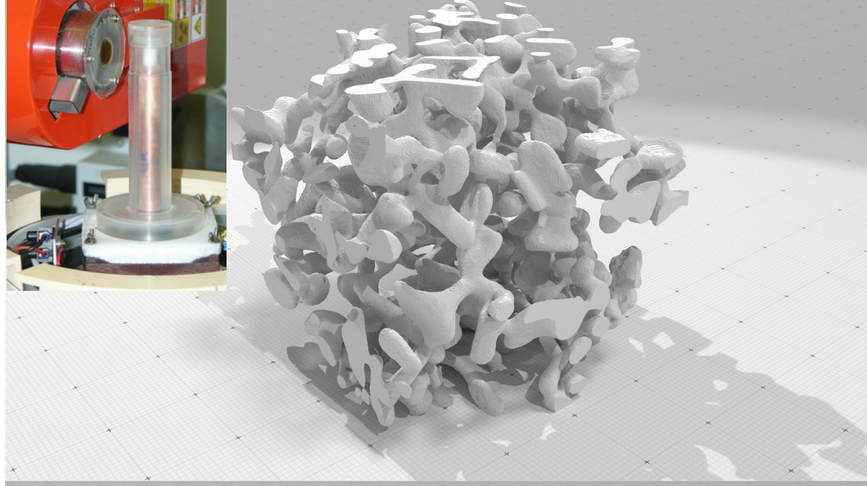


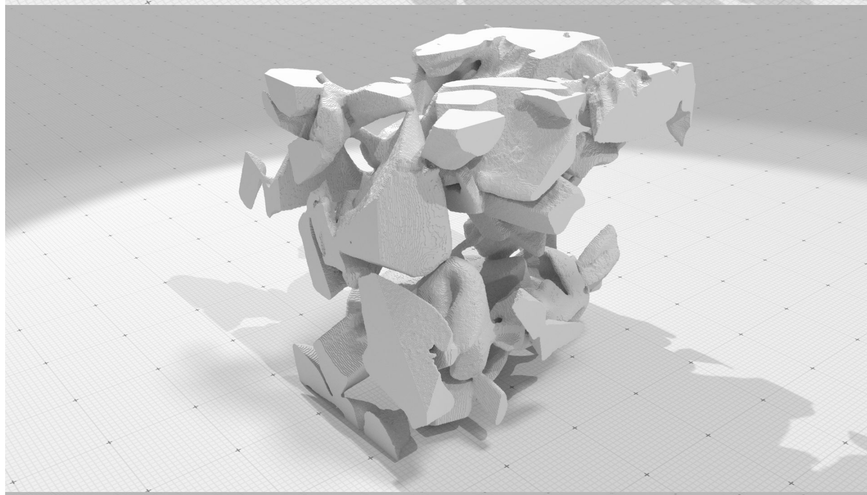
Motivations (1): devices

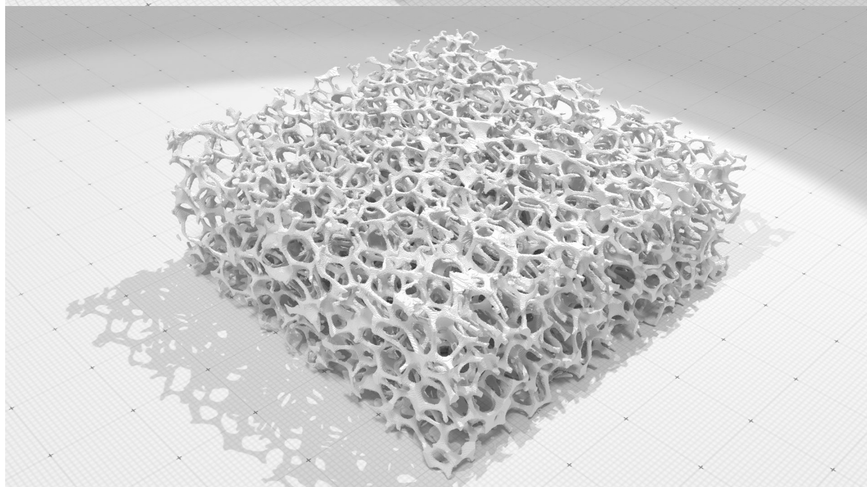
- Micro-tomographic images
  - material sciences
  - medical images

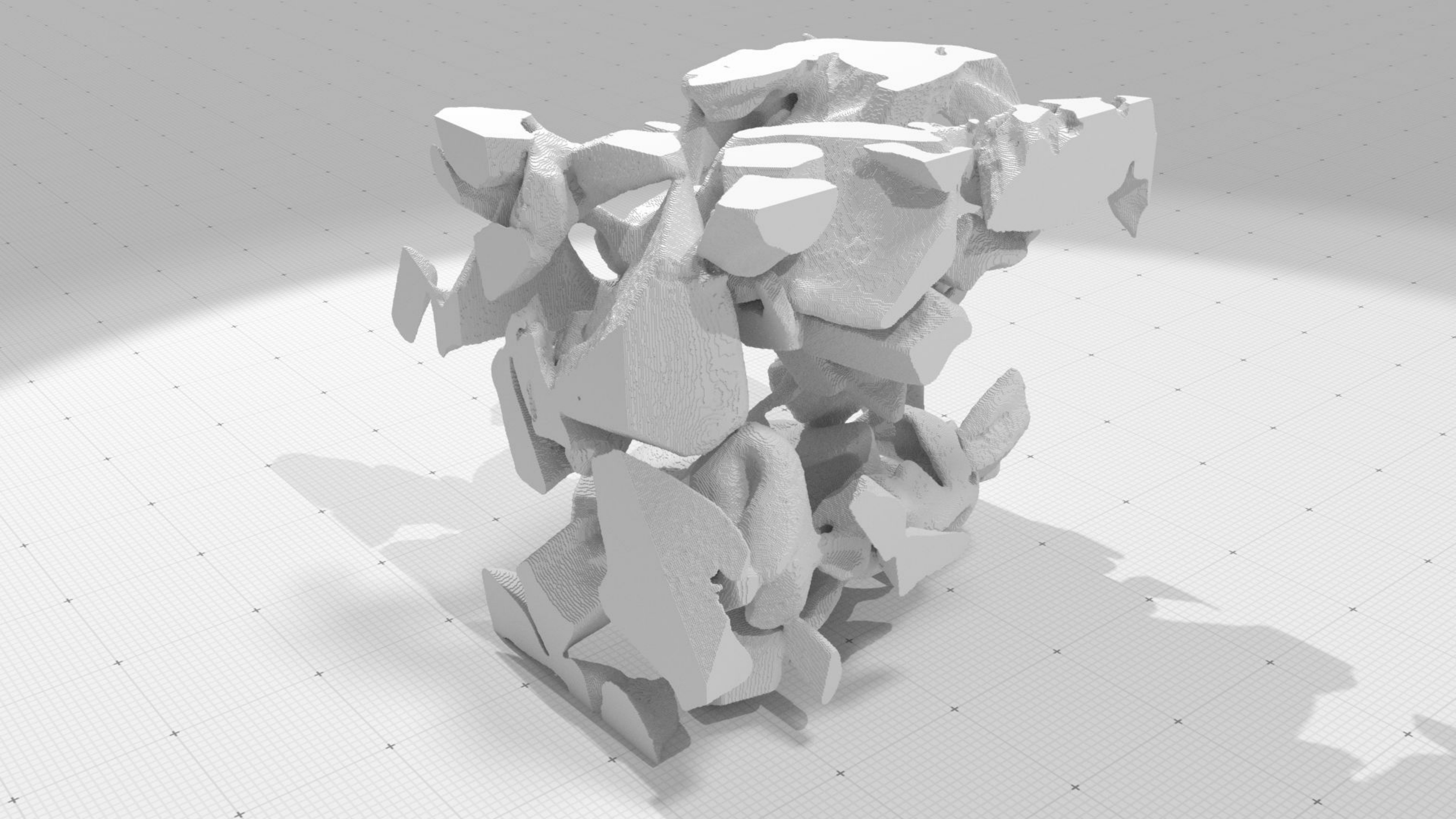
- Process geometry/topology of images partitions

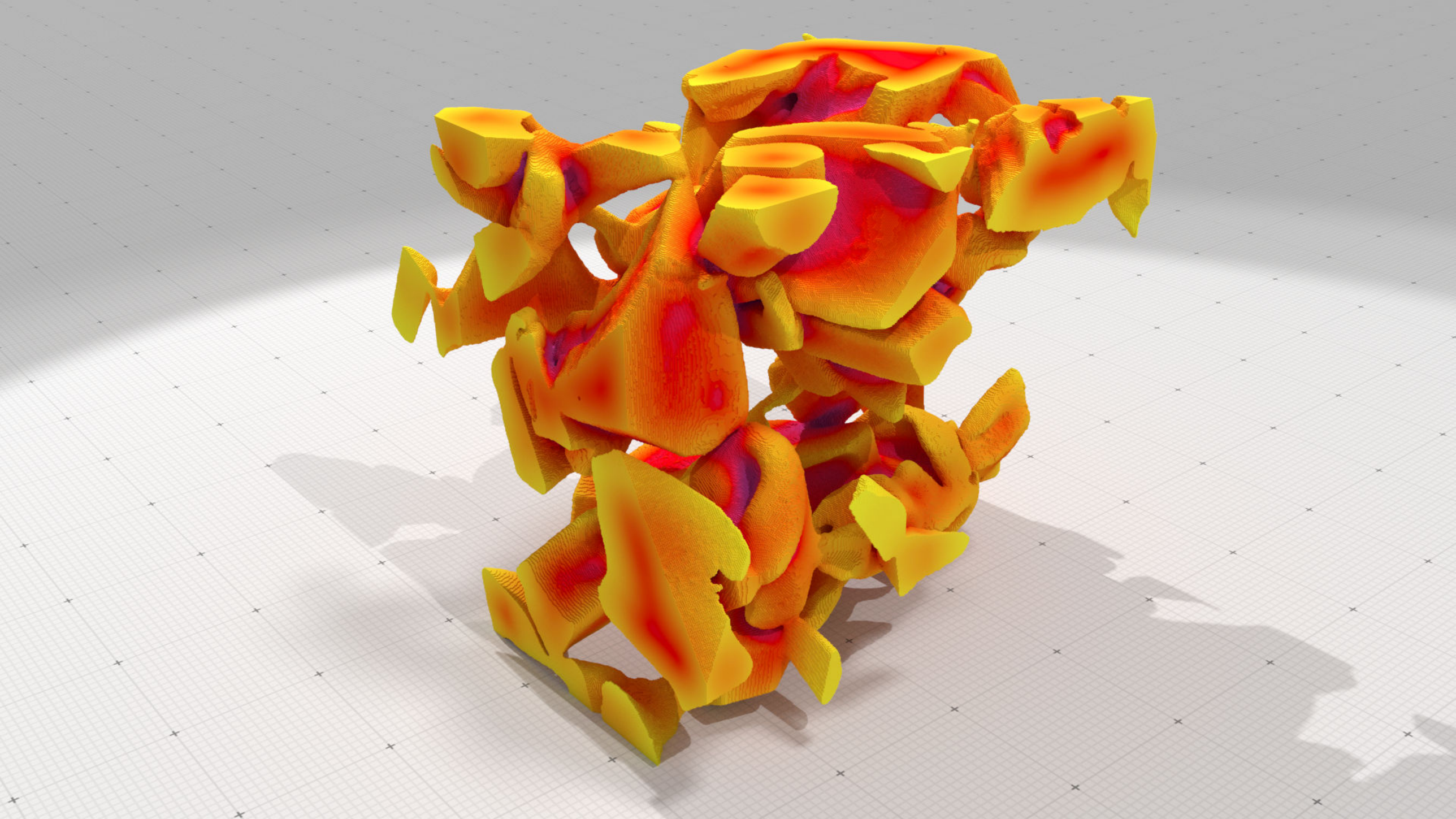
$$\Rightarrow X \subset \mathbb{Z}^3$$









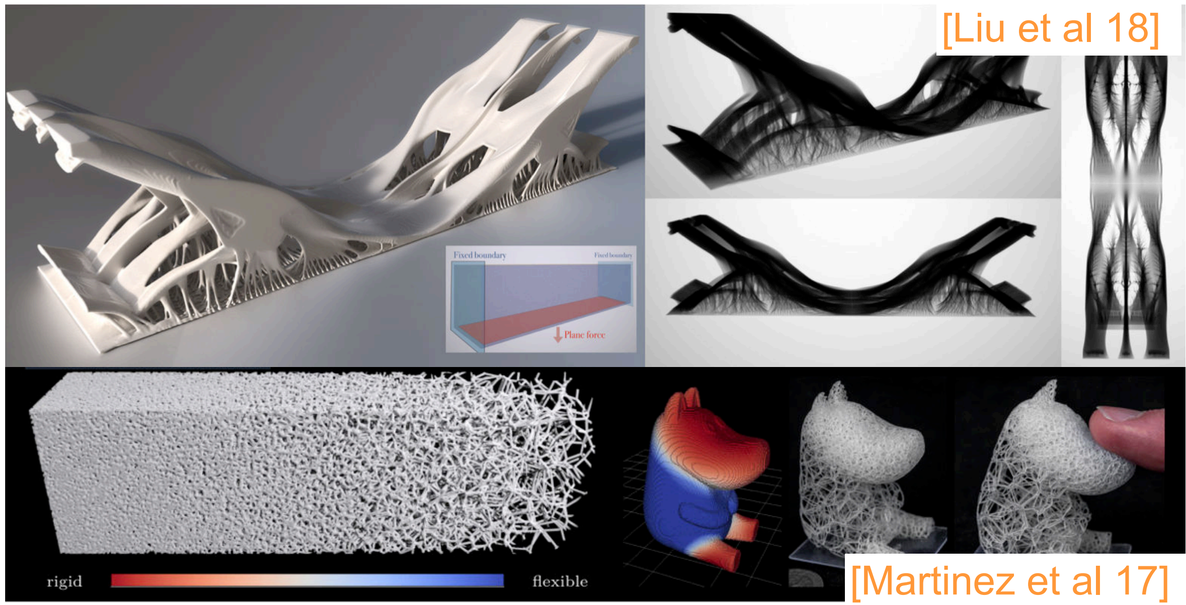


Motivations (2): $\mathbb{Z}^d$ as an efficient modelling space

- Shape optimization / fabrication

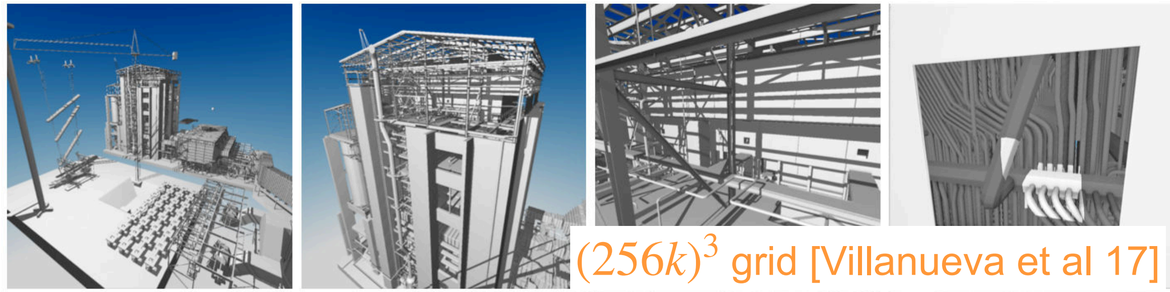
- As a proxy or an intermediate representation

  light transport simulation, booleans, medial axis, distance fields, multiple interfaces/objects tracking in a simulation loop…



[Liu et al 18]

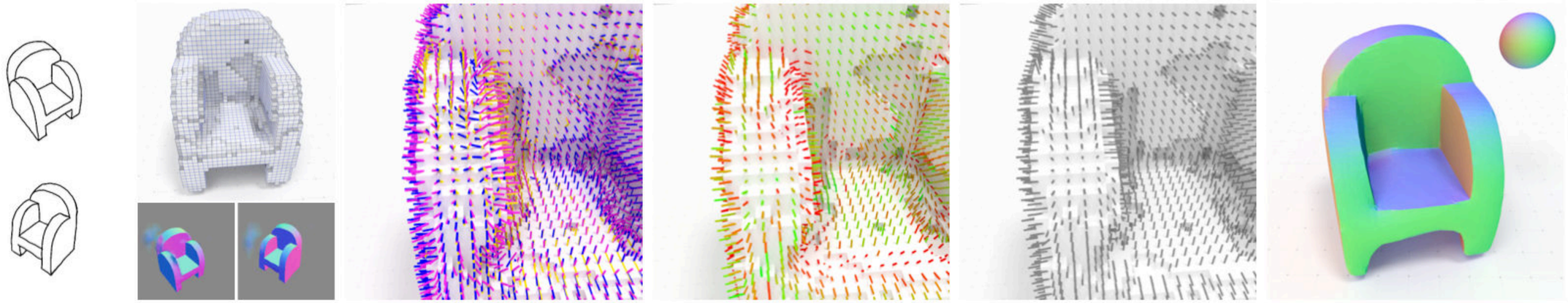
[Martinez et al 17]



$(256k)^3$ grid [Villanueva et al 17]

***Focus**: characteristic functions / labelled images / level sets / …*

Example



(a) Input (b) CNNs predictions (c) Candidate normals (d) Aggregated normals (e) Piecewise-smooth normals (f) Final surface

[Delanoy et al 19]

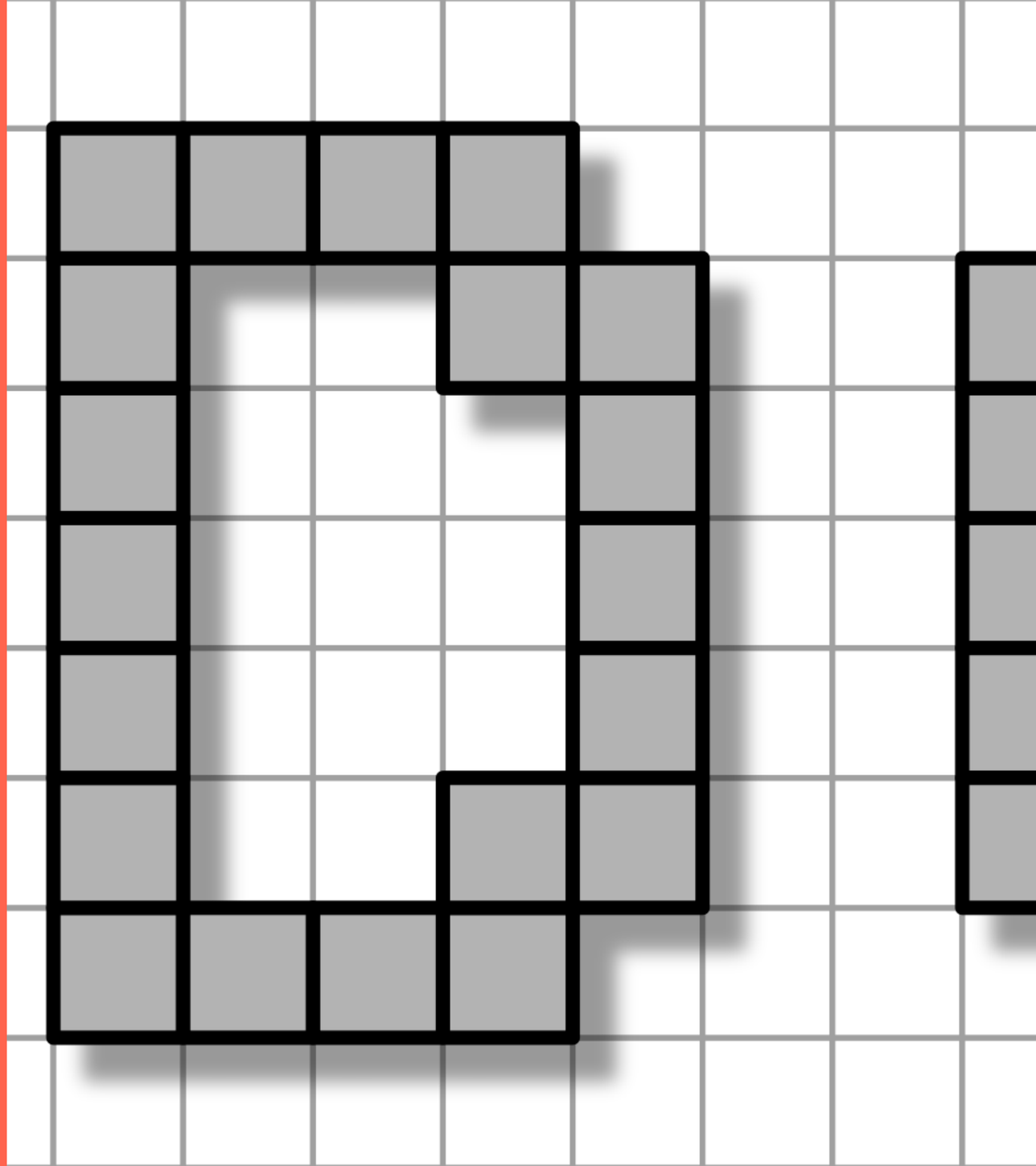
Digital Geometry

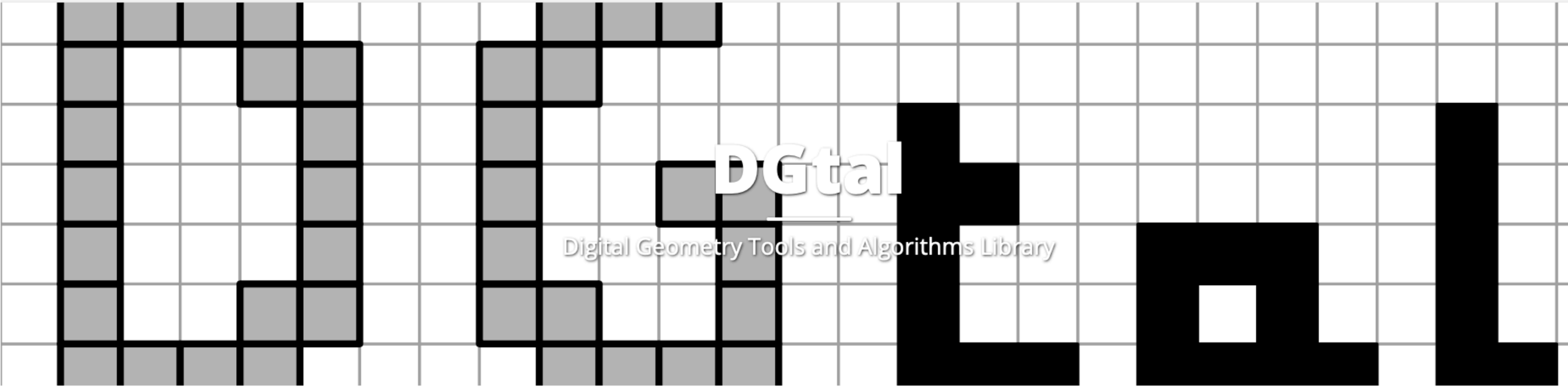
Topology and geometry processing on regular data:

- fast algorithms thanks to the regularity of the data
- simple topological structure
- integer based computations
- advanced surface based geometry processing

… in $\mathbb{Z}^d$

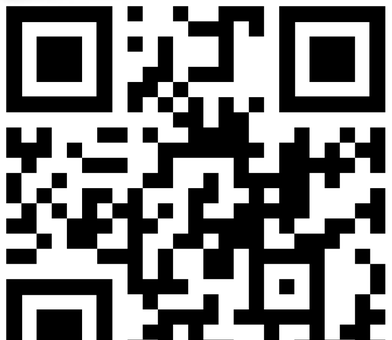
dgtal.org





Fork me on GitHub

https://dgtal.org



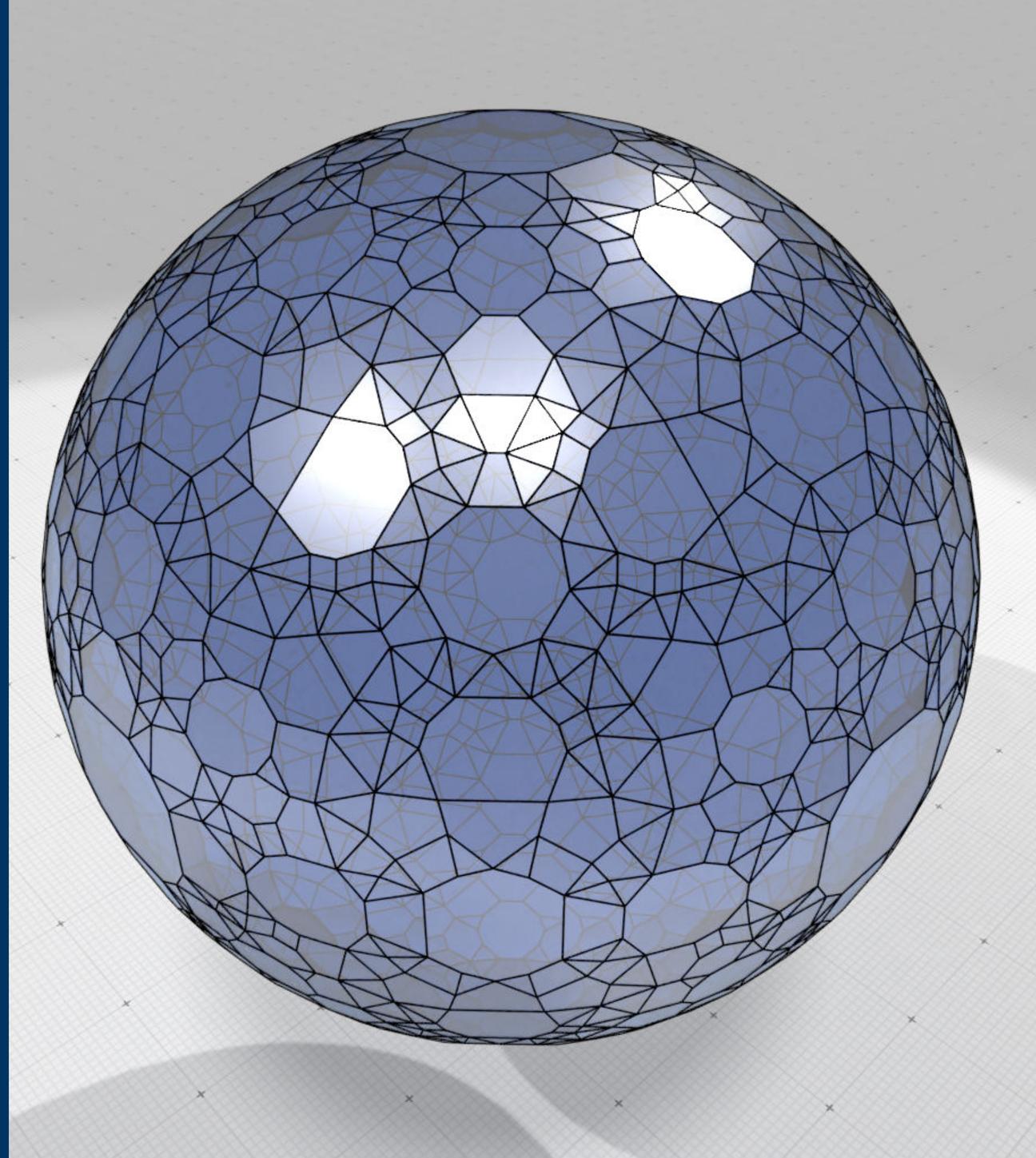
News

DGtal release 1.2

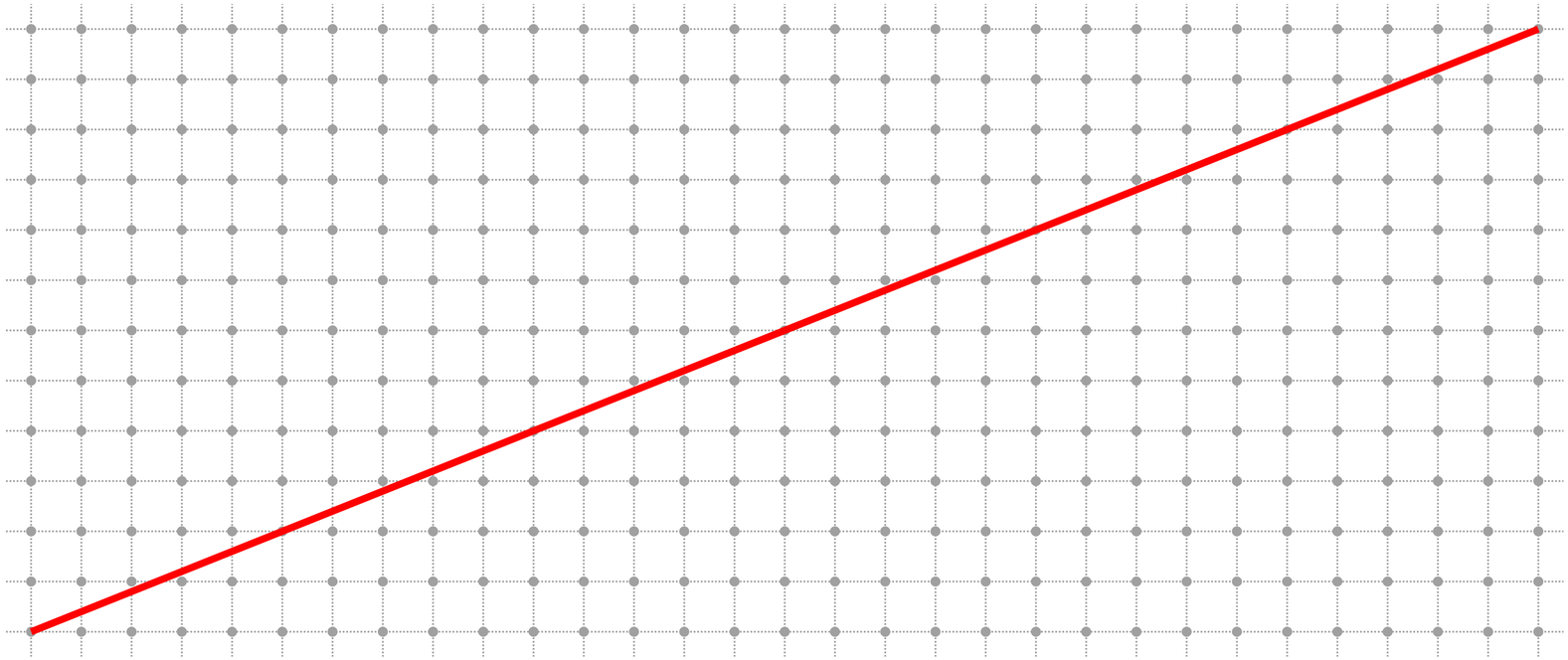
Posted on June 1, 2021

We are really excited to share with you the release 1.2 of DGtal and its tools. As usual, all edits and bugfixes are listed in the Changelog, and we would like to thank all devs involved in this release. In this short review, we would like to focus on only...
[Read More]

DGtal release 1.1



Quick example

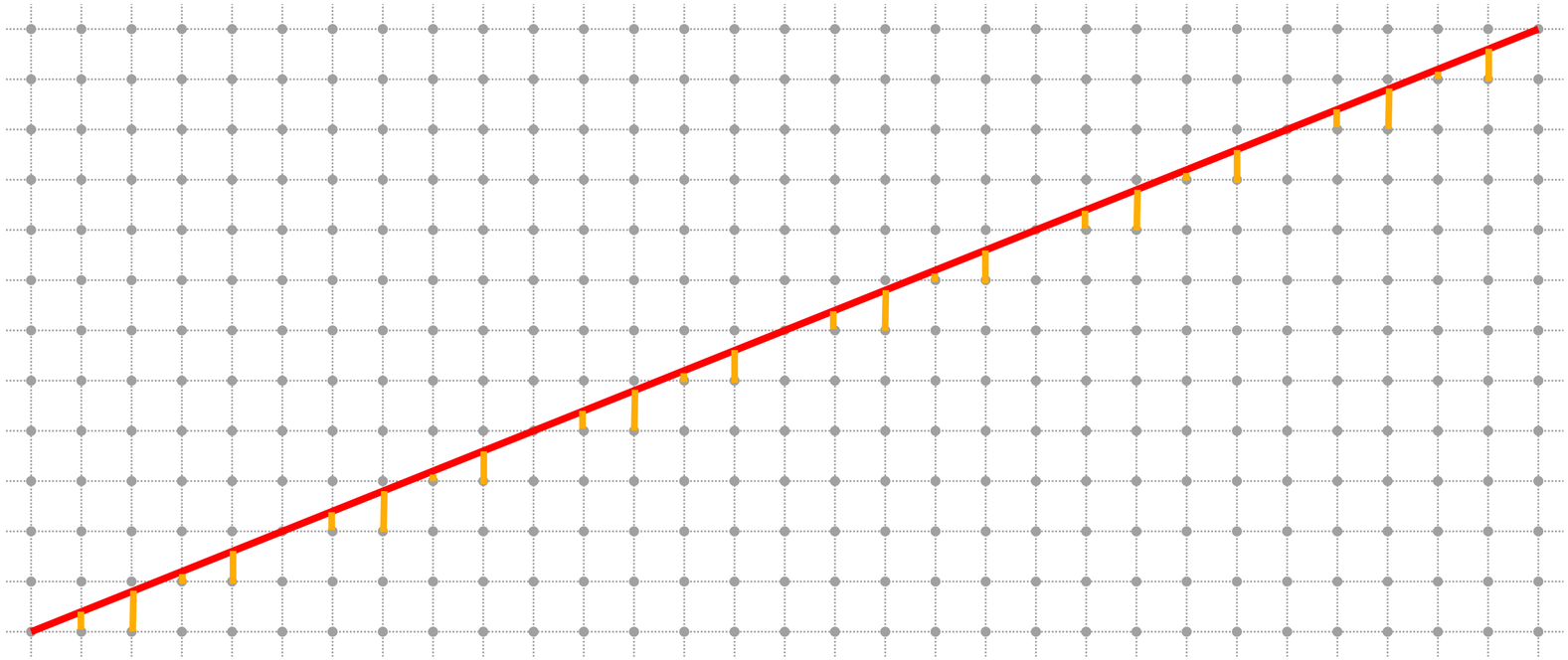


$$a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{a_3 + \ddots}}}$$

- Rational slope ⇒ finite set of remainders ⇒ periodic structure ⇒ canonical pattern from **continued fraction**

- *arithmetization* to speed-up tracing (e.g. fast ray marching on SVO)

Quick example

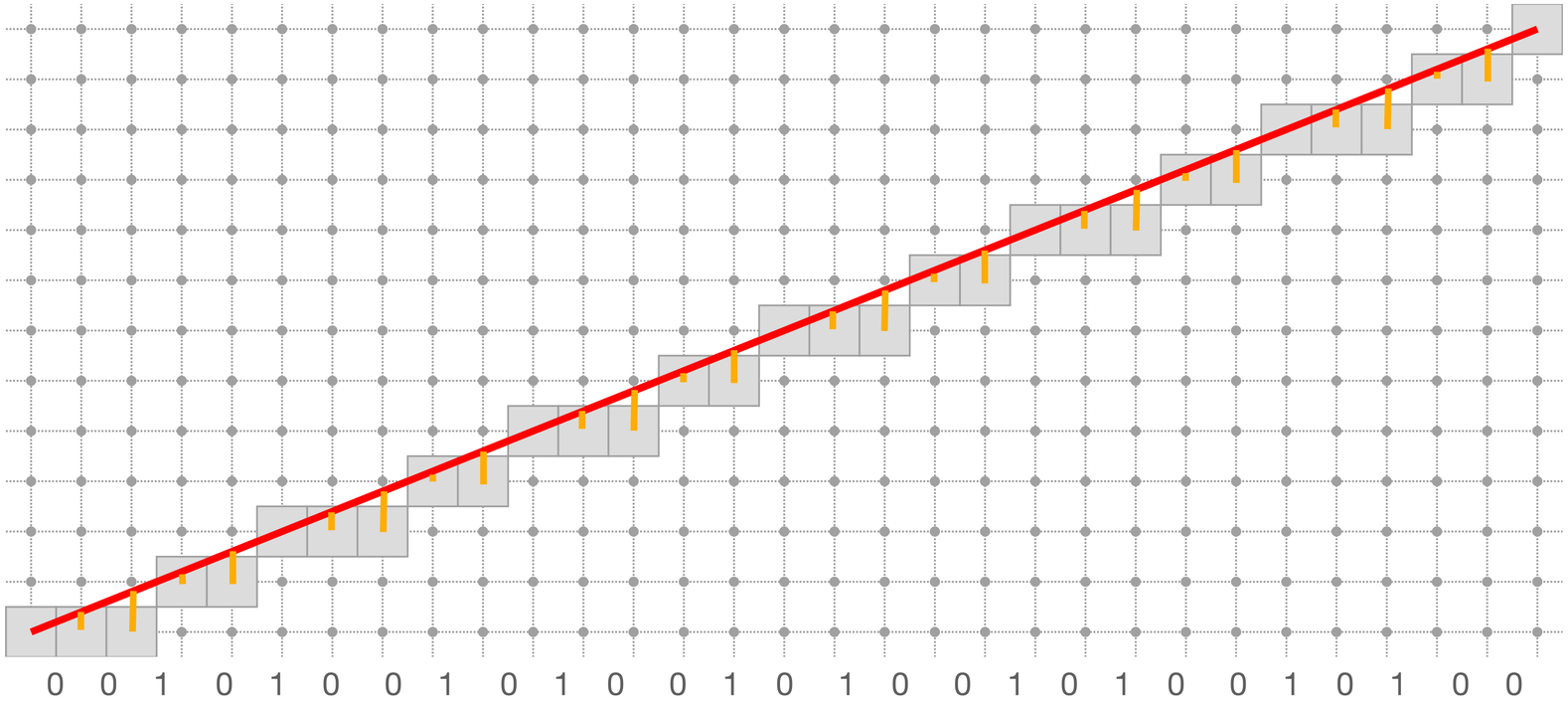


$$a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{a_3 + \ddots}}}$$

- Rational slope ⇒ finite set of remainders ⇒ periodic structure ⇒ canonical pattern from **continued fraction**

- *arithmetization* to speed-up tracing (e.g. fast ray marching on SVO)

Quick example

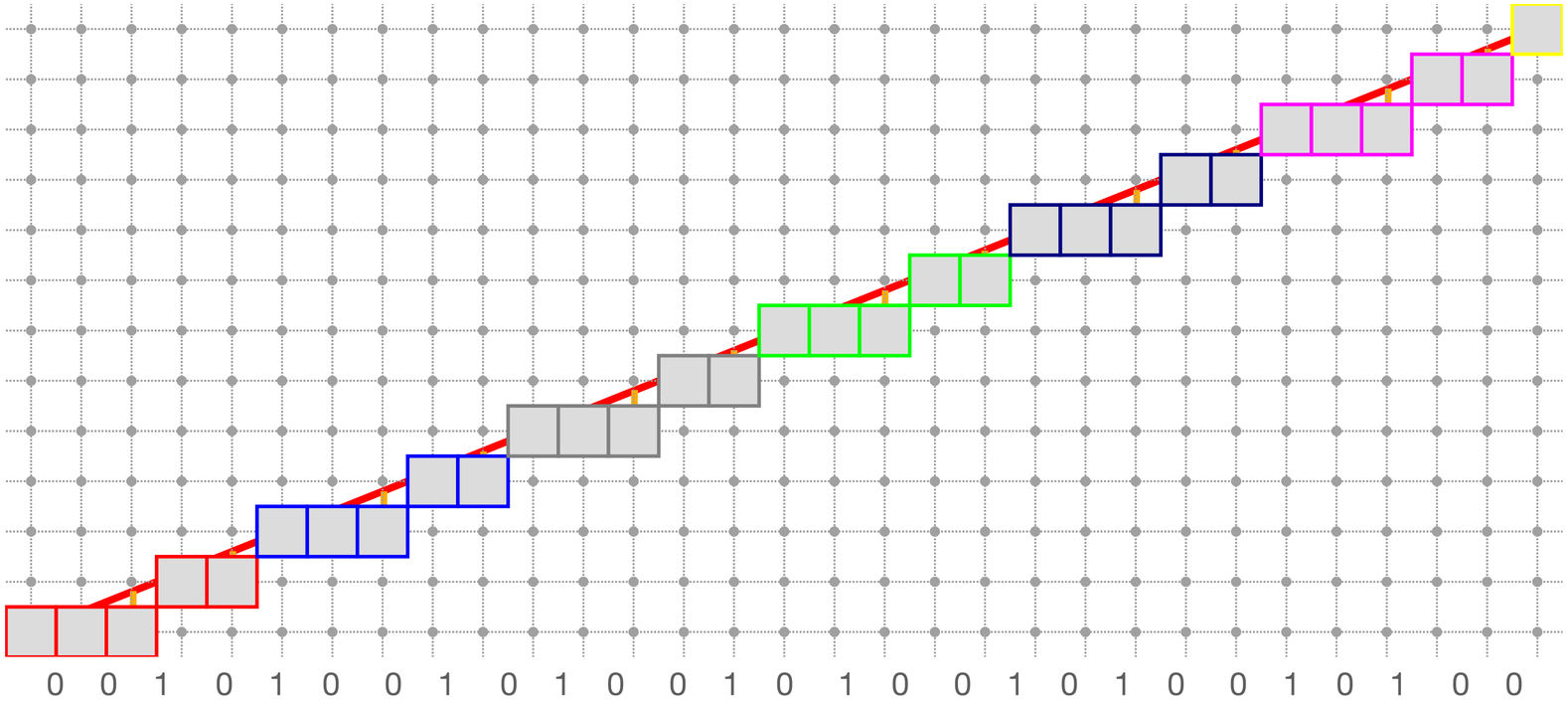


$$a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{a_3 + \ddots}}}$$

- Rational slope ⇒ finite set of remainders ⇒ periodic structure ⇒ canonical pattern from **continued fraction**

- *arithmetization* to speed-up tracing (e.g. fast ray marching on SVO)

Quick example



$$a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{a_3 + \ddots}}}$$

- Rational slope ⇒ finite set of remainders ⇒ periodic structure ⇒ canonical pattern from **continued fraction**

- *arithmetization* to speed-up tracing (e.g. fast ray marching on SVO)

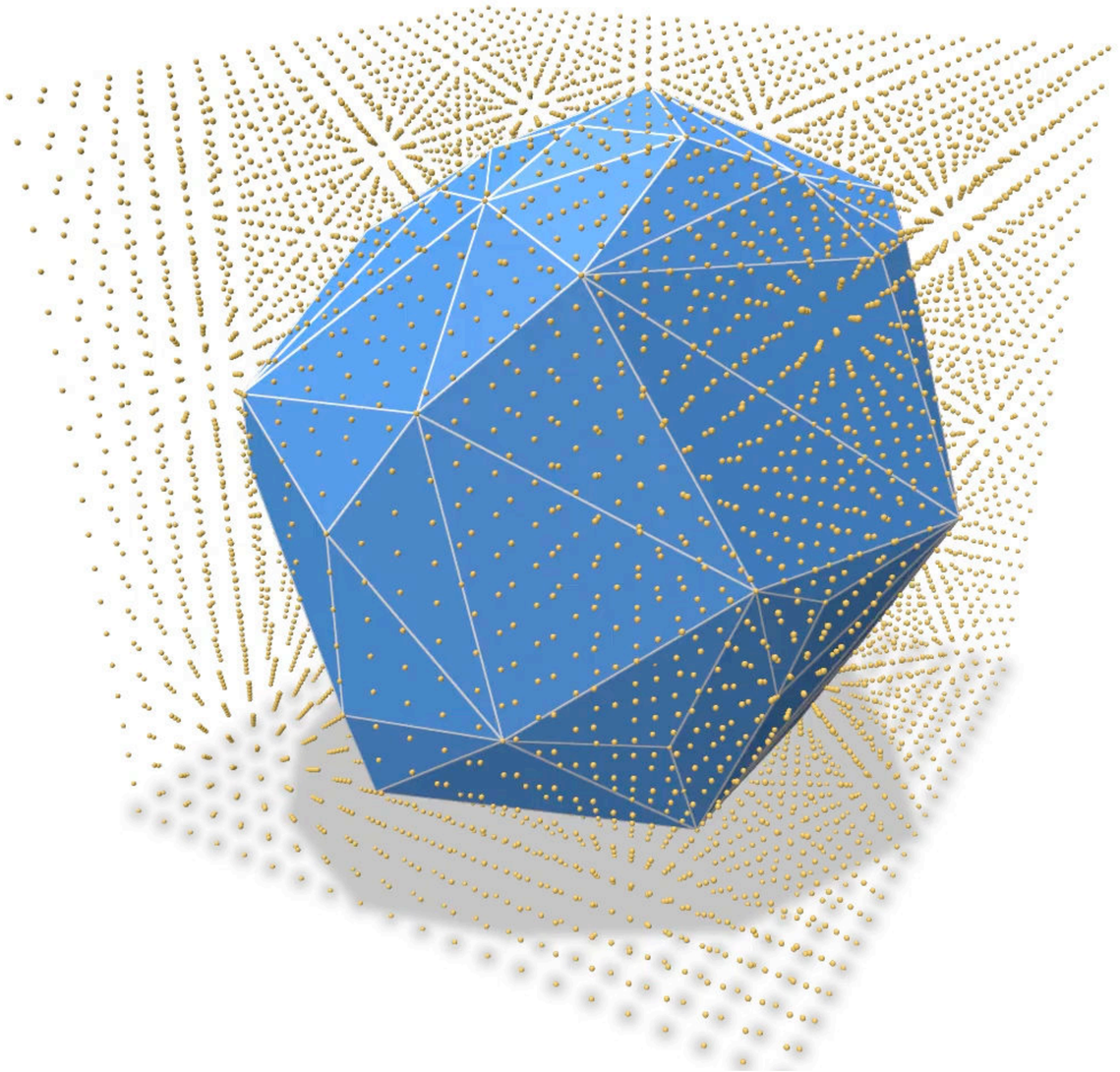
Further elements

Let $P \subset \mathbb{Z}^d$ *a lattice polytope with non-empty interior, then:* $f_k \ll c_d(Vol\,P)^{\frac{d-1}{d+1}}$

Convex on the lattice $[1,n]^2$ *grid has* $O(n^{2/3})$ *edges*

Let $P \subset [1,U]^2$ *(with* $U \leq 2^m$*) and* $n := |P|$*, the expected time for Voronoi diagram / Delaunay triangulation is:*

$$O\left(\min\{n \log n, n\sqrt{U}\}\right)$$



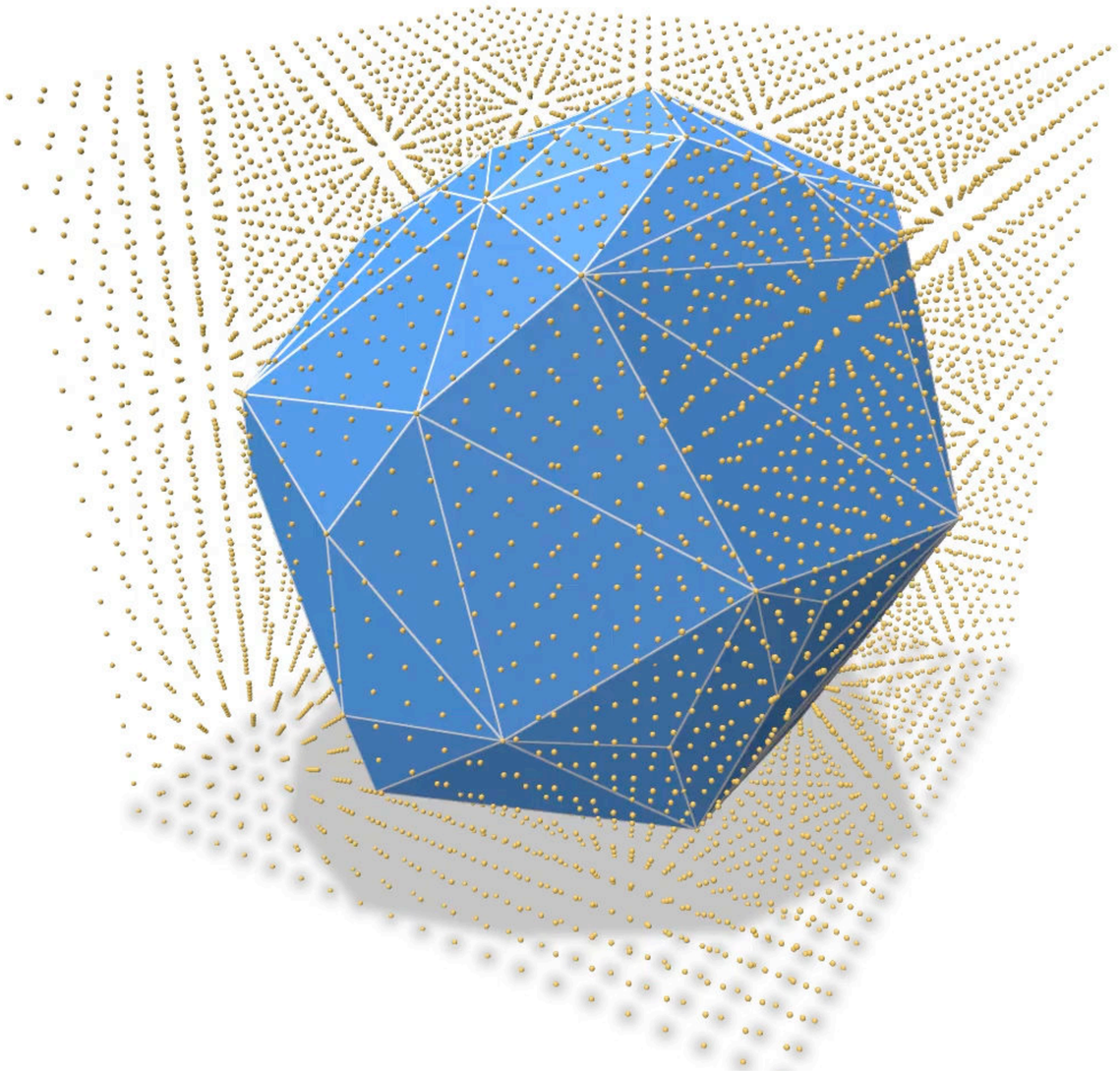
Further elements

Let $P \subset \mathbb{Z}^d$ *a lattice polytope with non-empty interior, then:* $f_k \ll c_d(Vol\,P)^{\frac{d-1}{d+1}}$

Convex on the lattice $[1,n]^2$ *grid has* $O(n^{2/3})$ *edges*

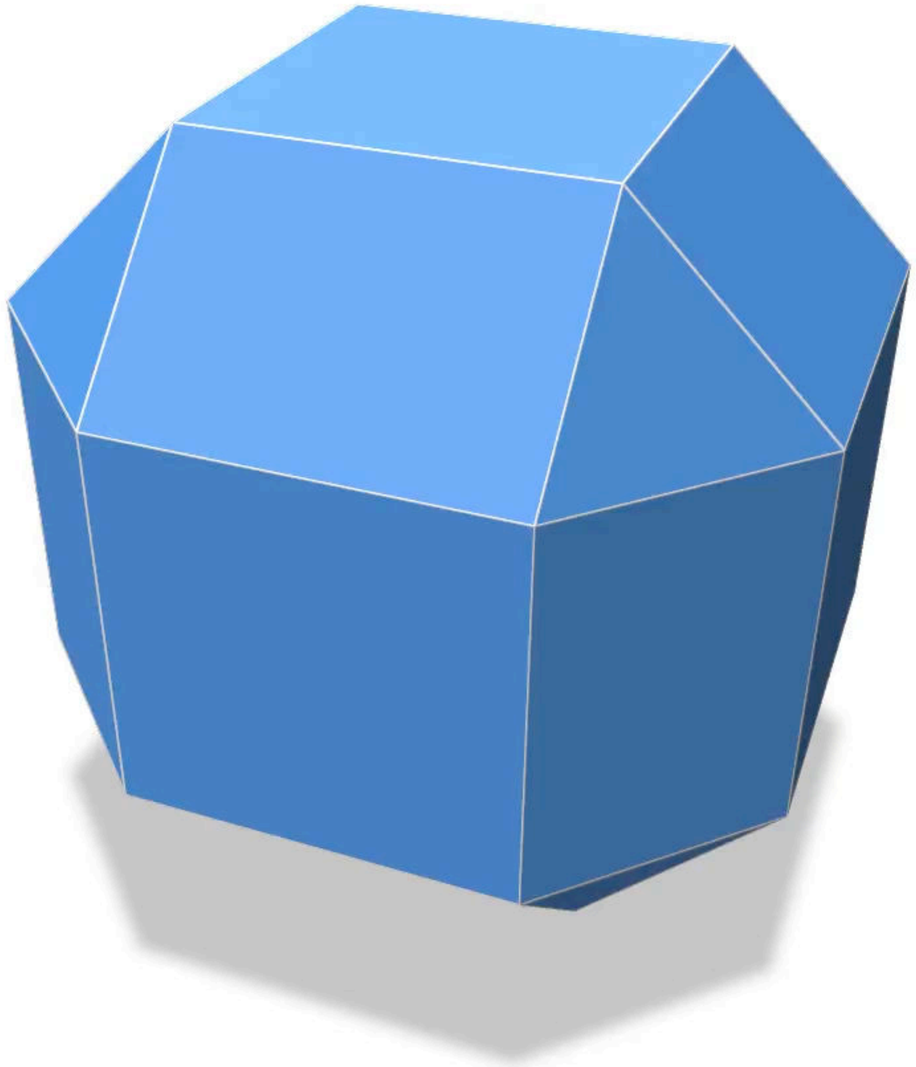
Let $P \subset [1,U]^2$ *(with* $U \leq 2^m$*) and* $n := |P|$*, the expected time for Voronoi diagram / Delaunay triangulation is:*

$$O\left(\min\{n \log n, n\sqrt{U}\}\right)$$

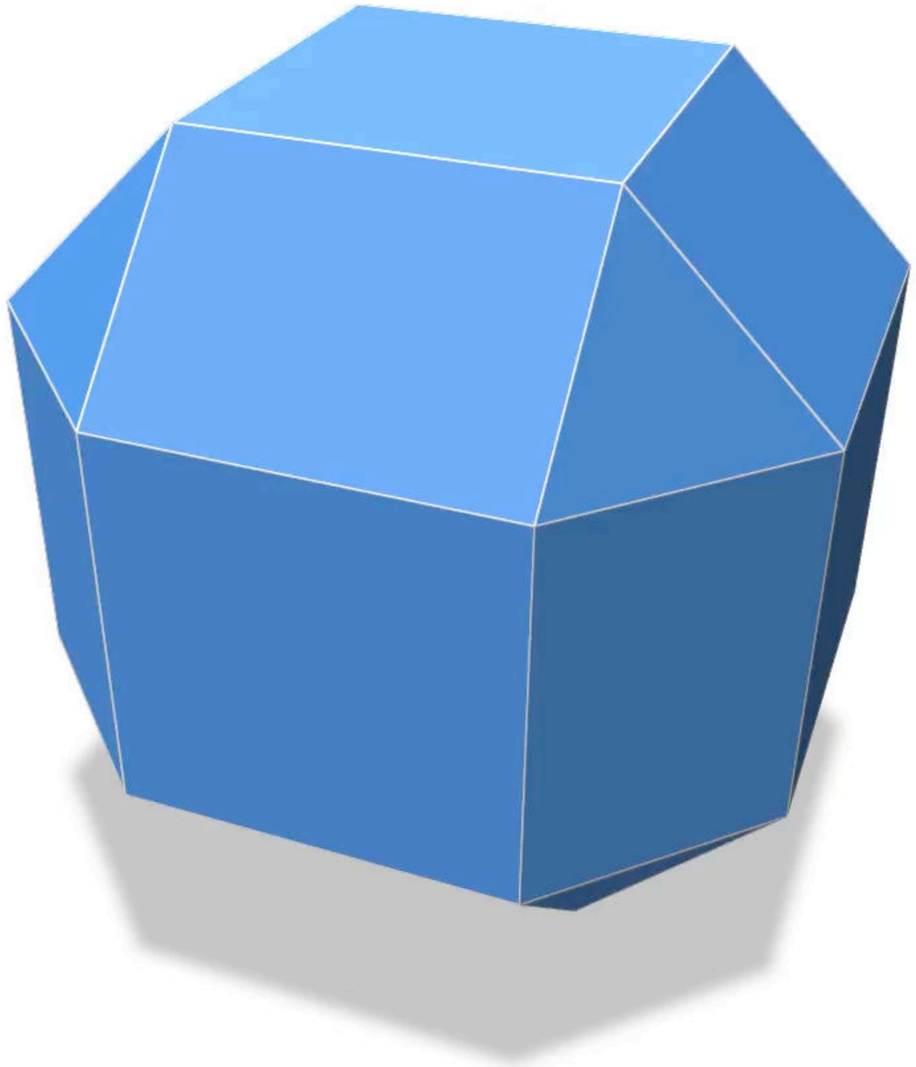


hands on…

```
void oneStep(double myh)
{
  auto params = SH3::defaultParameters();
  params( "polynomial", "sphere1" )( "gridstep", myh )
        ( "minAABB", -1.25 )( "maxAABB", 1.25 );
  auto implicit_shape  = SH3::makeImplicitShape3D  ( params );
  auto digitized_shape = SH3::makeDigitizedImplicitShape3D( implicit_shape, params );

  std::vector<Point> points;
  std::cout << "Digitzing shape" << std::endl;
  auto domain = digitized_shape->getDomain();
  for(auto &p: domain)
    if (digitized_shape->operator()(p))
      points.push_back(p);

  std::cout << "Computing convex hull" << std::endl;
  QuickHull3D hull;
  hull.setInput( points );
  hull.computeConvexHull();
  std::cout << "#points="    << hull.nbPoints()
            << " #vertices=" << hull.nbVertices()
            << " #facets="   << hull.nbFacets() << std::endl;

  std::vector< RealPoint > vertices;
  hull.getVertexPositions( vertices );
  std::vector< std::vector< std::size_t > > facets;
  hull.getFacetVertices( facets );

  polyscope::registerSurfaceMesh("Convex hull", vertices, facets)->rescaleToUnit();
}
```

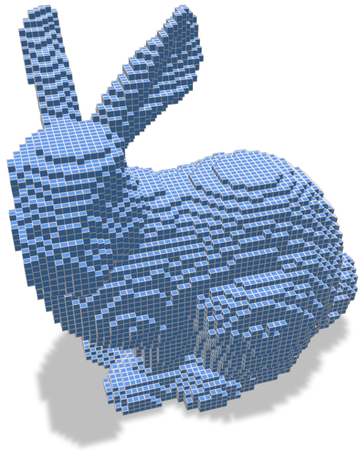


```
void oneStep(double myh)
{
  auto params = SH3::defaultParameters();
  params( "polynomial", "sphere1" )( "gridstep", myh )
        ( "minAABB", -1.25 )( "maxAABB", 1.25 );
  auto implicit_shape  = SH3::makeImplicitShape3D  ( params );
  auto digitized_shape = SH3::makeDigitizedImplicitShape3D( implicit_shape, params );

  std::vector<Point> points;
  std::cout << "Digitzing shape" << std::endl;
  auto domain = digitized_shape->getDomain();
  for(auto &p: domain)
    if (digitized_shape->operator()(p))
      points.push_back(p);

  std::cout << "Computing convex hull" << std::endl;
  QuickHull3D hull;
  hull.setInput( points );
  hull.computeConvexHull();
  std::cout << "#points="    << hull.nbPoints()
            << " #vertices=" << hull.nbVertices()
            << " #facets="   << hull.nbFacets() << std::endl;

  std::vector< RealPoint > vertices;
  hull.getVertexPositions( vertices );
  std::vector< std::vector< std::size_t > > facets;
  hull.getFacetVertices( facets );

  polyscope::registerSurfaceMesh("Convex hull", vertices, facets)->rescaleToUnit();
}
```



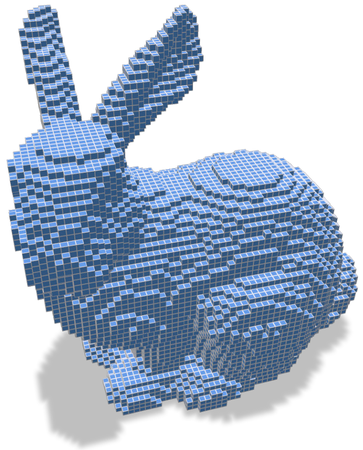
$\mathbb{Z}^d$

Volumetric analysis



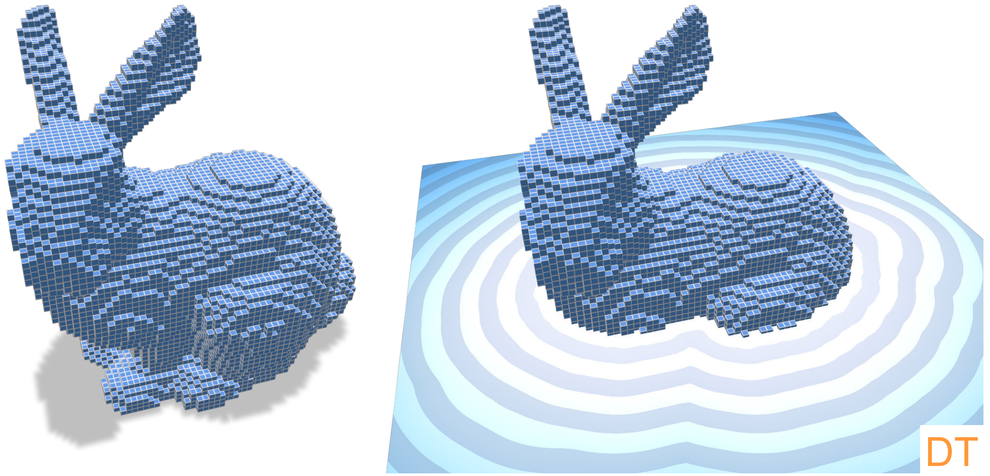
Given $X \subset \mathbb{Z}^d$ and a domain $[0,n]^d$, compute:

Volumetric analysis



Given $X \subset \mathbb{Z}^d$ and a domain $[0,n]^d$, compute:

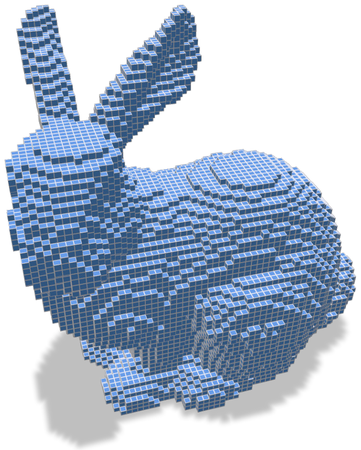
Volumetric analysis

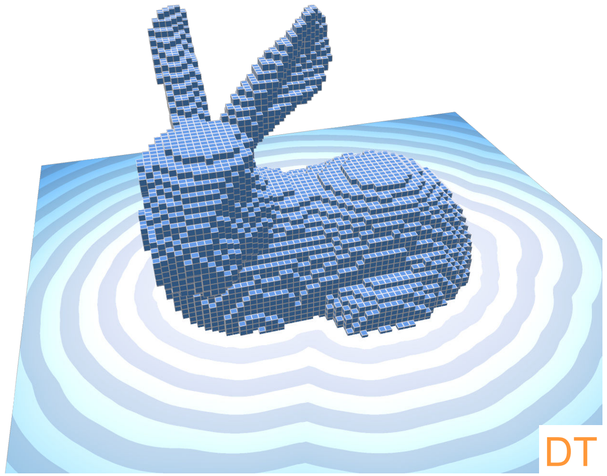


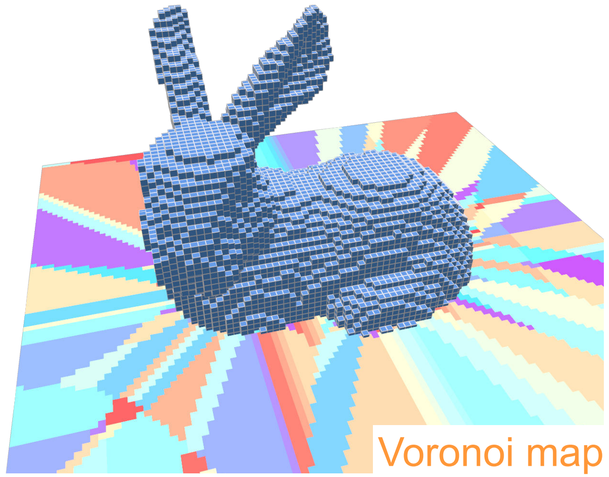
Given $X \subset \mathbb{Z}^d$ ***and a domain*** $[0,n]^d$***, compute:***

$$DT(x) = min_{y \in D \setminus X} \; d(x,y)$$ *(aka distance map)*

Volumetric analysis





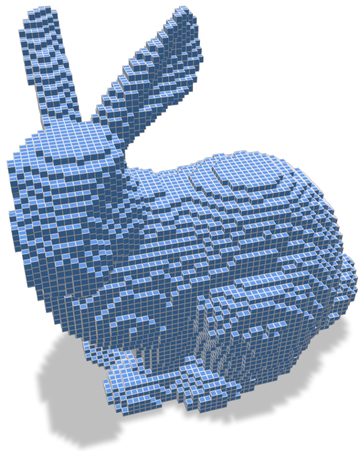


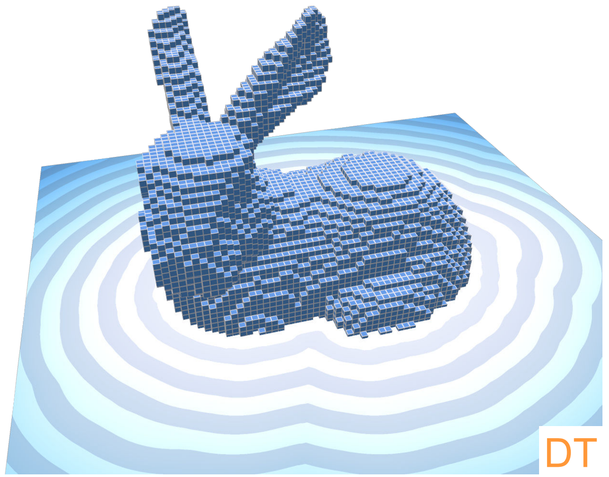
Given $X \subset \mathbb{Z}^d$ ***and a domain*** $[0,n]^d$***, compute:***

$DT(x) = min_{y \in D \setminus X} \; d(x,y)$ *(aka distance map)*

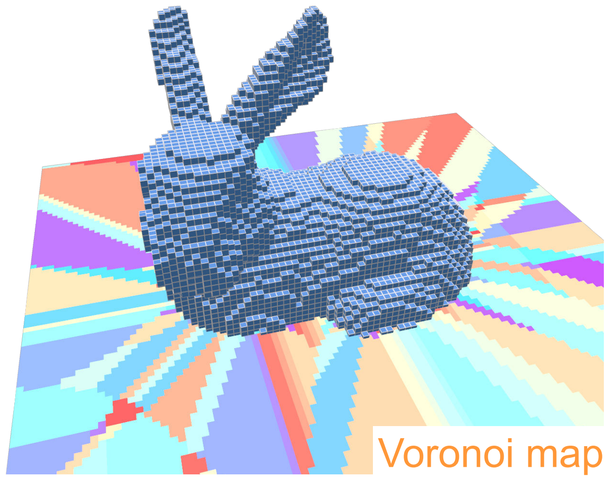
$\sigma(x) = \text{argmin}_{y \in D \setminus X} \; d(x,y)$ *(aka Voronoi map* $\mathscr{V}(X) \cap \mathbb{Z}^d$*)*

Volumetric analysis

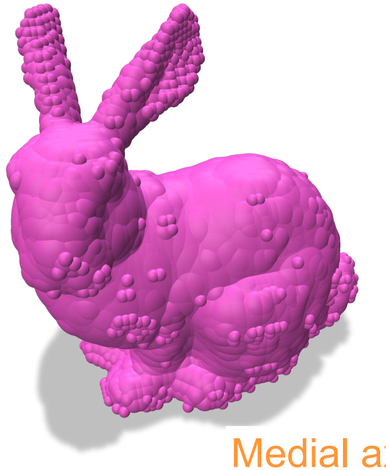




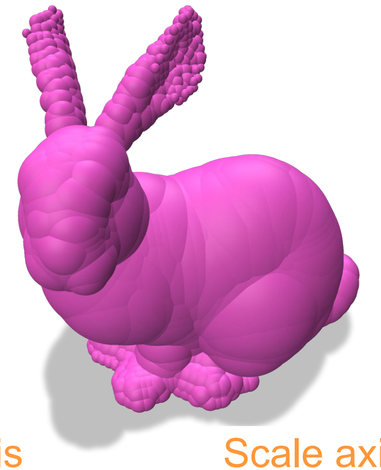
DT



Voronoi map



Medial axis



Scale axis

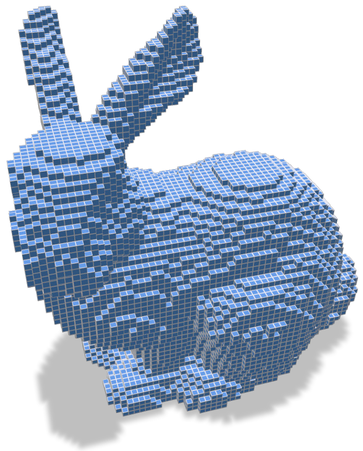
Given $X \subset \mathbb{Z}^d$ ***and a domain*** $[0,n]^d$***, compute:***

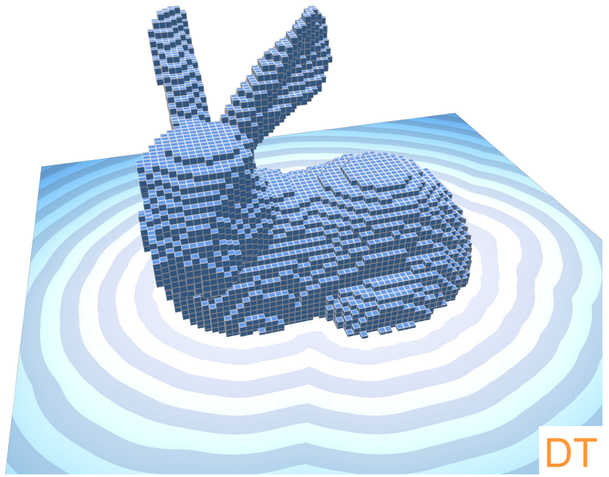
$DT(x) = min_{y \in D \setminus X} \; d(x,y)$ *(aka distance map)*

$\sigma(x) = \text{argmin}_{y \in D \setminus X} \; d(x,y)$ *(aka Voronoi map* $\mathscr{V}(X) \cap \mathbb{Z}^d$*)*

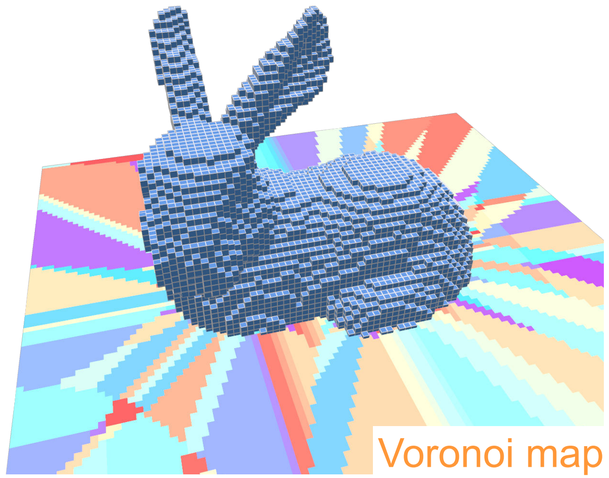
$M = \{(x,r) \in \mathbb{Z}^{d+1} \mid \mathscr{B}(x,r) \cap \mathbb{Z}^d \subset X$, *there is no* (x',r') *s.t.* $\mathscr{B}(x,r) \subset \mathscr{B}(x',r')\}$ *(aka discrete medial axis)*

Volumetric analysis

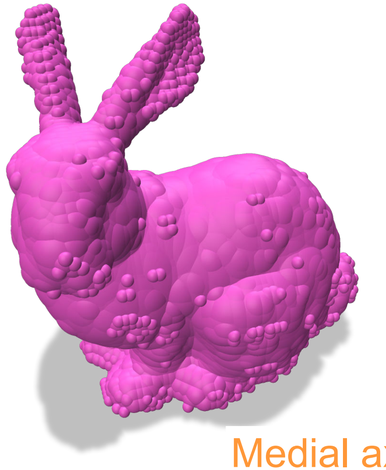




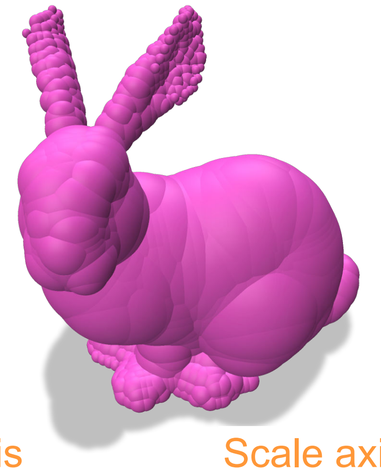
DT



Voronoi map



Medial axis



Scale axis

Given $X \subset \mathbb{Z}^d$ ***and a domain*** $[0,n]^d$***, compute:***

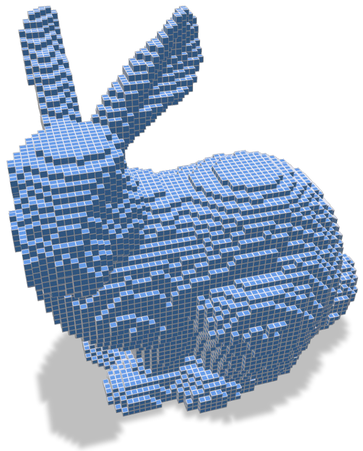
$DT(x) = min_{y \in D \setminus X} \; d(x,y)$ *(aka distance map)*

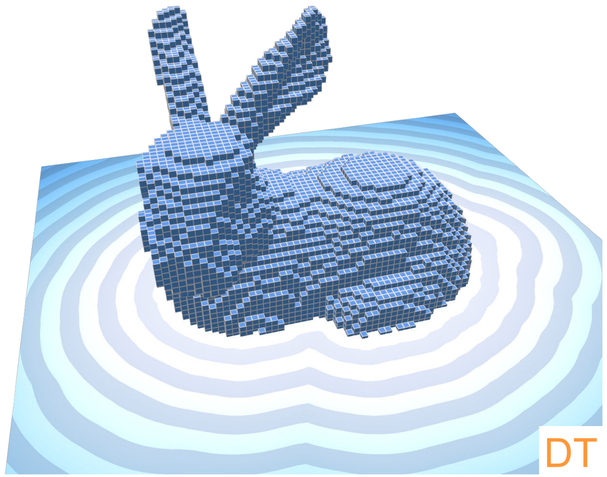
$\sigma(x) = \text{argmin}_{y \in D \setminus X} \; d(x,y)$ *(aka Voronoi map* $\mathscr{V}(X) \cap \mathbb{Z}^d$*)*

$M = \{(x,r) \in \mathbb{Z}^{d+1} \mid \mathscr{B}(x,r) \cap \mathbb{Z}^d \subset X,$ *there is no* (x',r') *s.t.* $\mathscr{B}(x,r) \subset \mathscr{B}(x',r')\}$ *(aka discrete medial axis)*

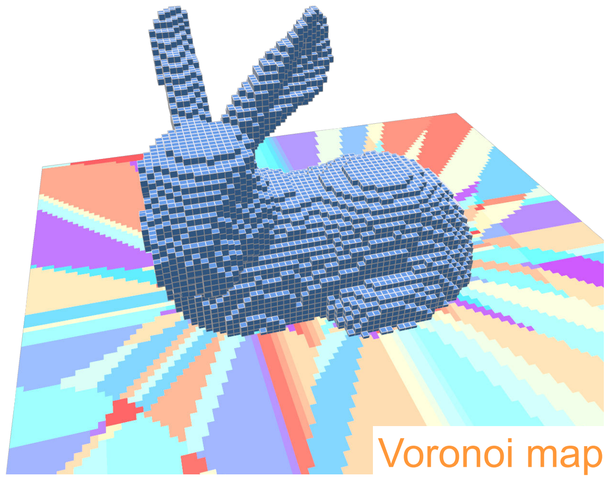
$\pi(x) = \text{argmin}_{(y,r) \in M} \; \|x-y\|_2^2 - r^2$ *(aka* l_2 *Power map* $\mathscr{P}(M) \cap \mathbb{Z}^d$*)*

Volumetric analysis

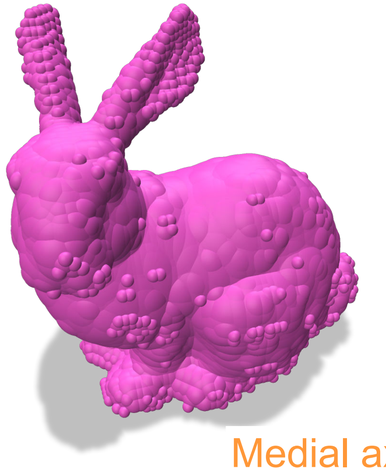




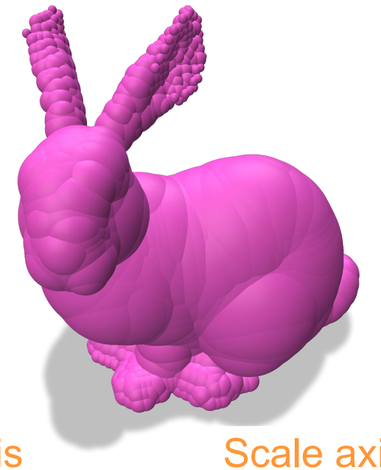
DT



Voronoi map



Medial axis



Scale axis

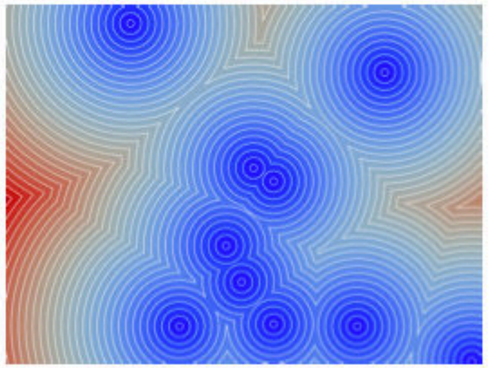
Given $X \subset \mathbb{Z}^d$ ***and a domain*** $[0,n]^d$***, compute:***

- $DT(x) = min_{y \in D \setminus X} \; d(x,y)$ *(aka distance map)*
- $\sigma(x) = \text{argmin}_{y \in D \setminus X} \; d(x,y)$ *(aka Voronoi map* $\mathscr{V}(X) \cap \mathbb{Z}^d$*)*

$M = \{(x,r) \in \mathbb{Z}^{d+1} \mid \mathscr{B}(x,r) \cap \mathbb{Z}^d \subset X$, *there is no* (x',r') *s.t.* $\mathscr{B}(x,r) \subset \mathscr{B}(x',r')\}$ *(aka discrete medial axis)*

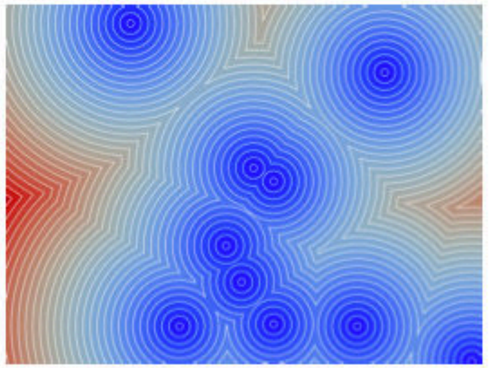
$\pi(x) = \text{argmin}_{(y,r) \in M} \; \|x-y\|_2^2 - r^2$ *(aka* l_2 *Power map* $\mathscr{P}(M) \cap \mathbb{Z}^d$*)*

Separable distance field



$$DT(x) = \min_{y \in D \setminus X} \; ||x - y||_2$$

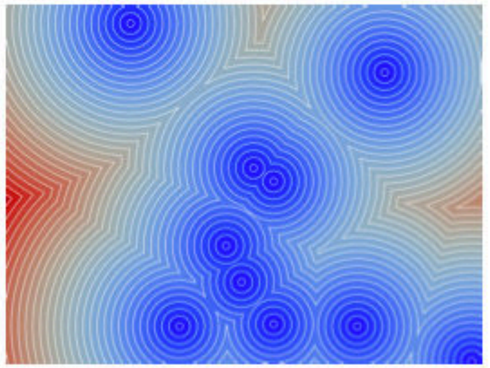
Separable distance field



$$DT(x) = \min_{y \in D \setminus X} ||x - y||_2$$

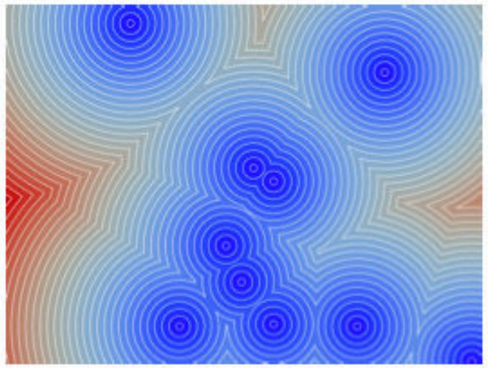
$$= \min_{(u,v) \notin X} (i-u)^2 + (j-v)^2$$

Separable distance field



$$
\begin{aligned}
DT(x) &= \min_{y \in D \setminus X} \; ||x - y||_2 \\
&= \min_{(u,v) \notin X} \; (i-u)^2 + (j-v)^2 \\
&= \min_{v} \left((\min_{u} \; (i-u)^2) + (j-v)^2 \right)
\end{aligned}
$$

Separable distance field



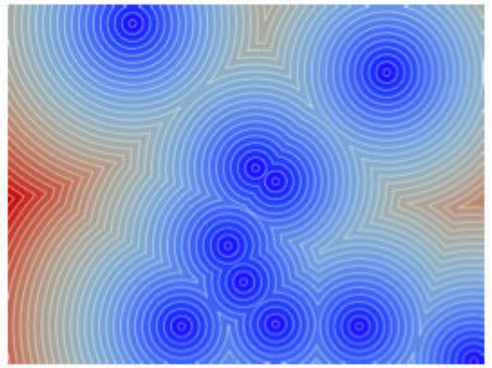
$$DT(x) = \min_{y \in D \setminus X} ||x - y||_2$$

$$= \min_{(u,v) \notin X} (i-u)^2 + (j-v)^2$$

$$= \min_{v} \left(\left(\min_{u} (i-u)^2 \right) + (j-v)^2 \right)$$

per line double-scan = $O(n)$

Separable distance field



$$DT(x) = \min_{y \in D \setminus X} ||x - y||_2$$

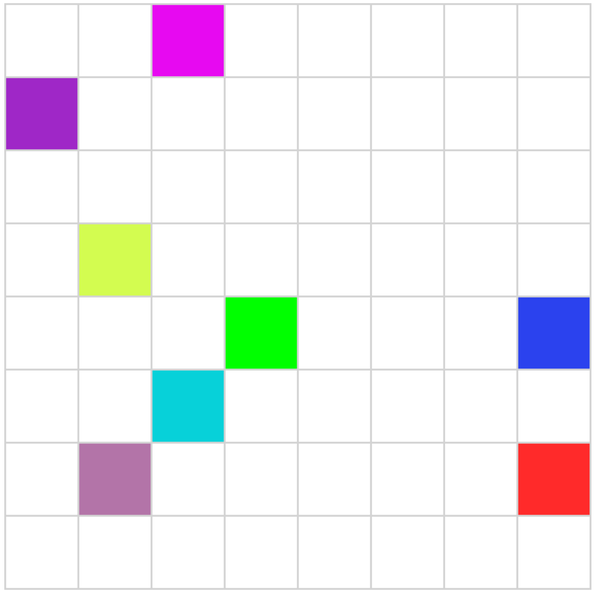
$$= \min_{(u,v) \notin X} (i-u)^2 + (j-v)^2$$

$$= \min_{v} \left(\left(\min_{u} (i-u)^2 \right) + (j-v)^2 \right)$$

per line double-scan = $O(n)$

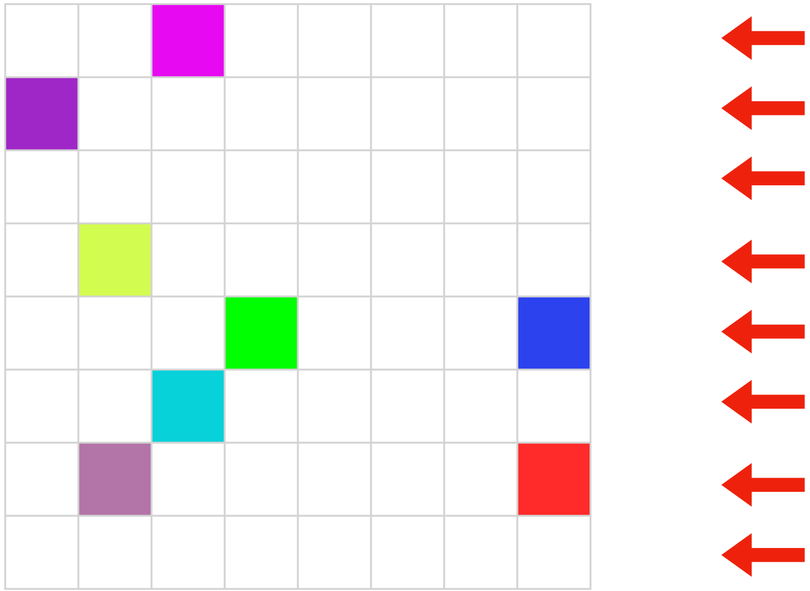
1D lower enveloppe computation of a set of parabolas = $O(n)$

Separable Voronoi map: step 1

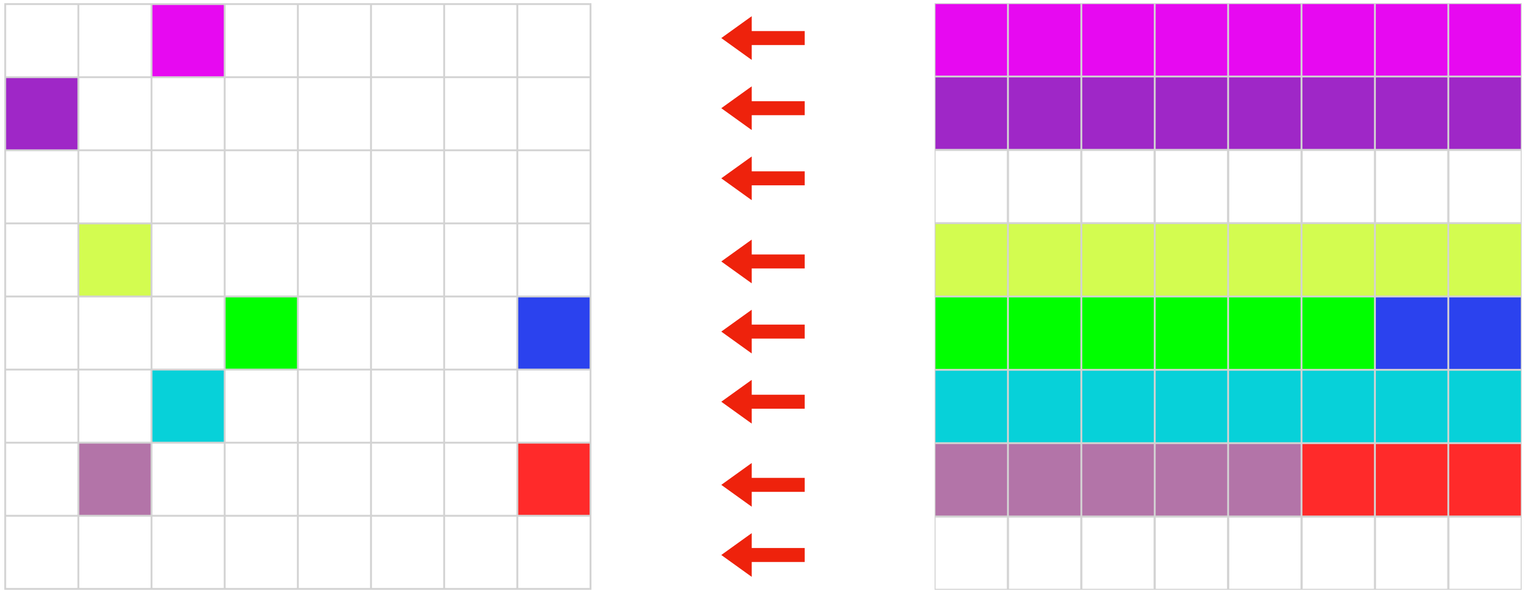


Separable Voronoi map: step 1

Separable Voronoi map: step 1

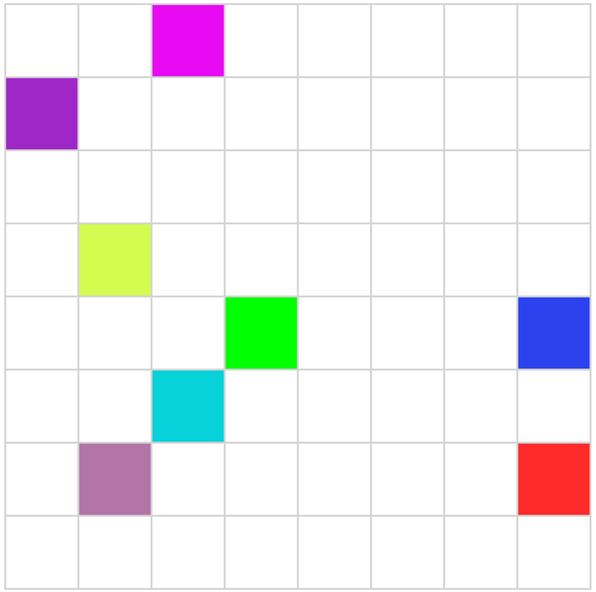


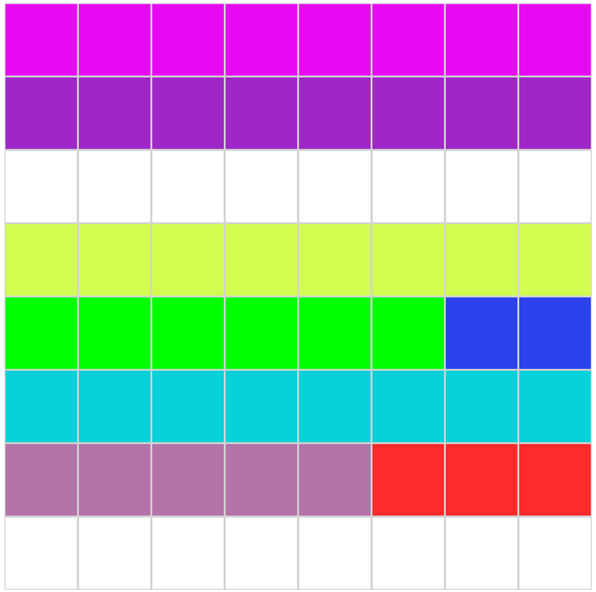
Separable Voronoi map: step 1



$\Rightarrow O(n)$ per row

Separable Voronoi map: step 1

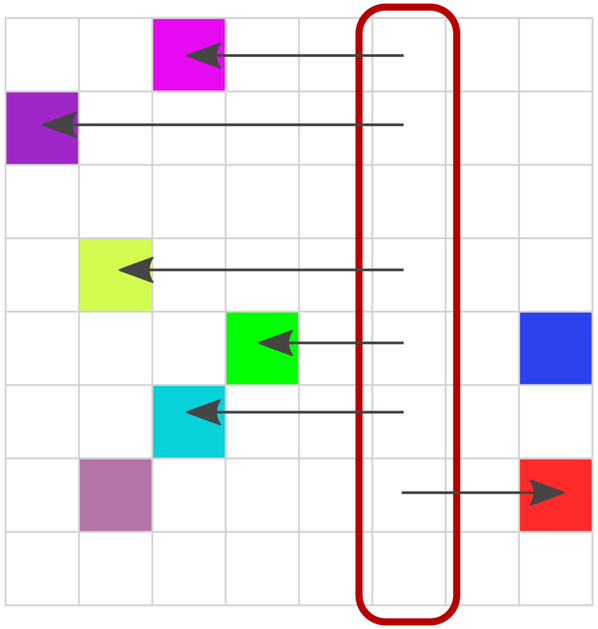




$\Rightarrow O(n)$ per row

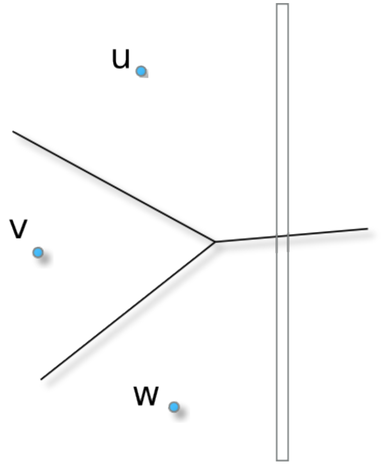
Separable Voronoi map: step 1

Separable Voronoi map: step 1



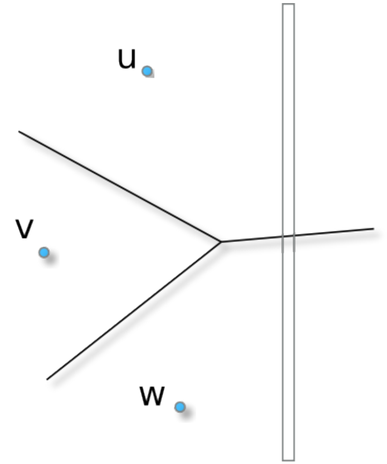
Separable Voronoi map: step 2

Stack based algorithm using a 3-ary *hiddenBy* predicate, *à la* sweep line ⇒ $O(n)$ per column



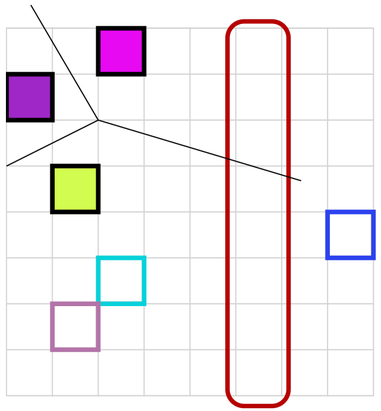
hiddenBy(u,v,w,S)

Separable Voronoi map: step 2

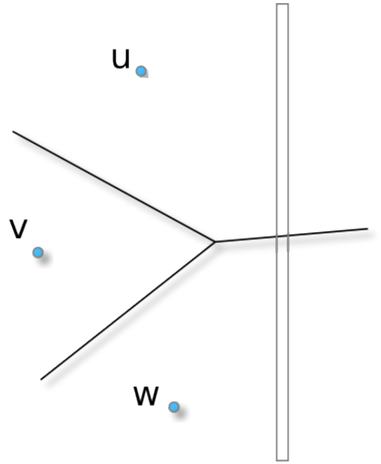


hiddenBy(u,v,w,S)

Stack based algorithm using a 3-ary *hiddenBy* predicate, *à la* sweep line $\Rightarrow O(n)$ per column

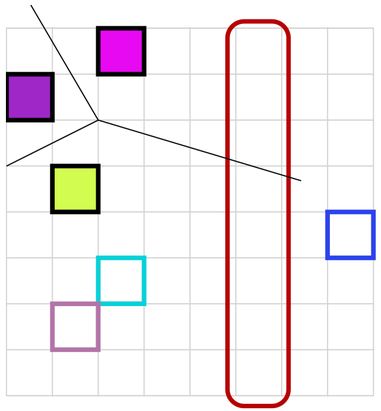


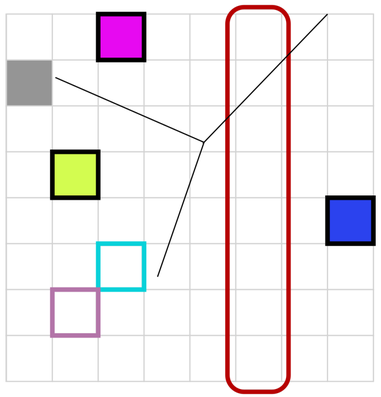
Separable Voronoi map: step 2



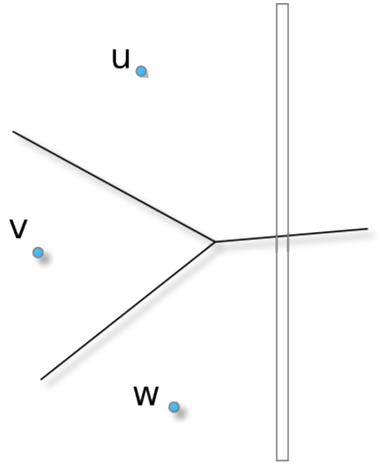
hiddenBy(u,v,w,S)

Stack based algorithm using a 3-ary *hiddenBy* predicate, *à la* sweep line $\Rightarrow O(n)$ per column



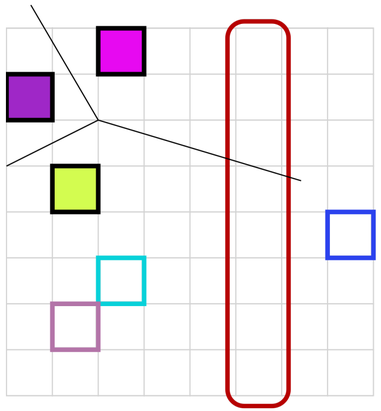


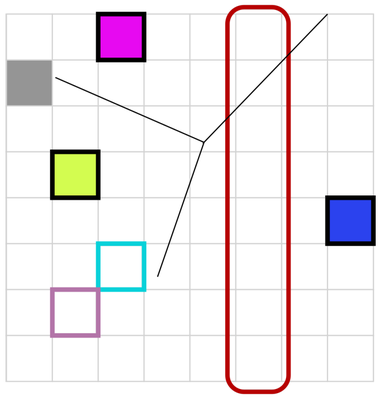
Separable Voronoi map: step 2

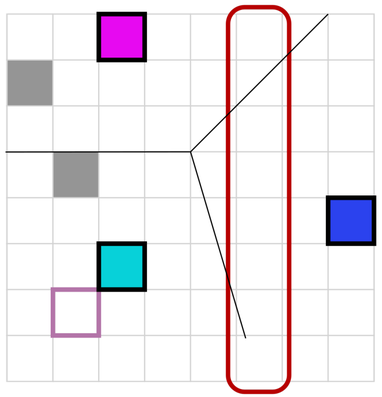


hiddenBy(u,v,w,S)

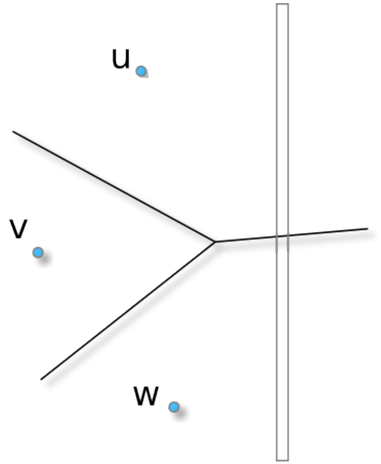
Stack based algorithm using a 3-ary *hiddenBy* predicate, *à la* sweep line $\Rightarrow O(n)$ per column





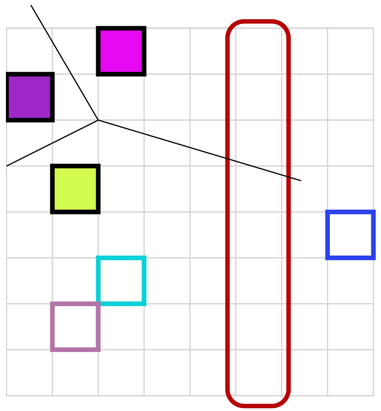


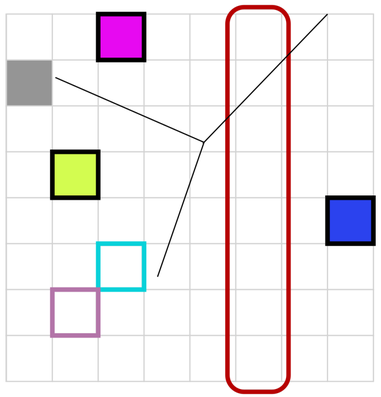
Separable Voronoi map: step 2

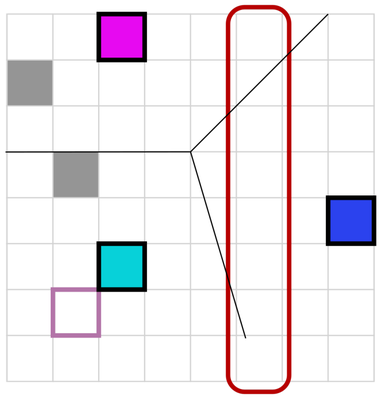


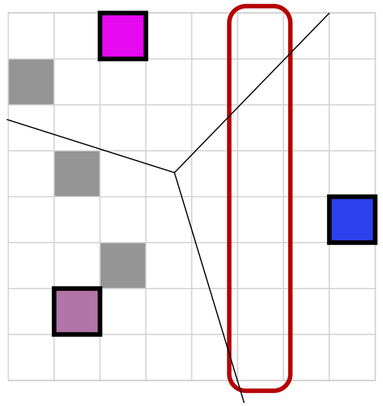
hiddenBy(u,v,w,S)

Stack based algorithm using a 3-ary *hiddenBy* predicate, *à la* sweep line ⇒ $O(n)$ per column

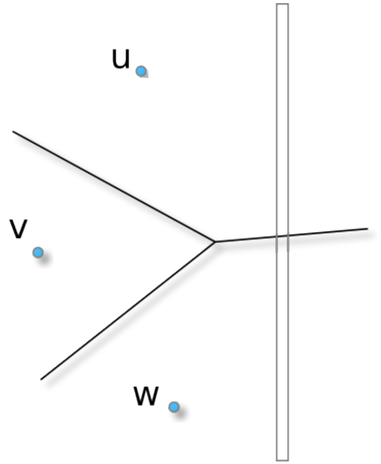






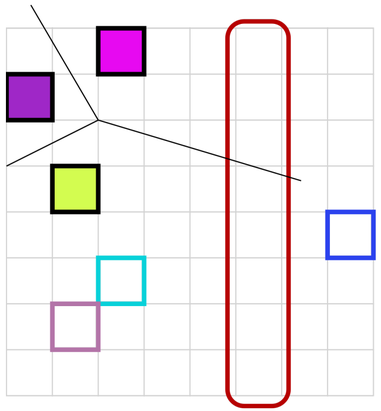


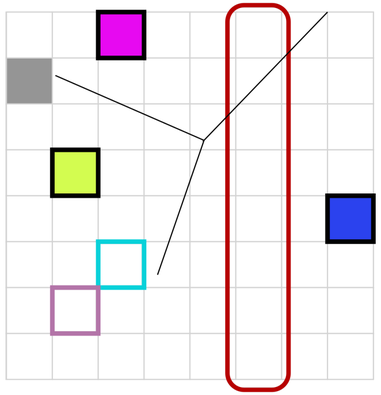
Separable Voronoi map: step 2

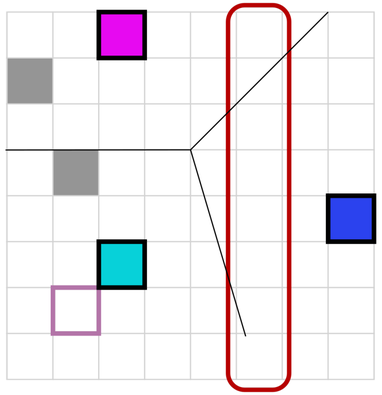


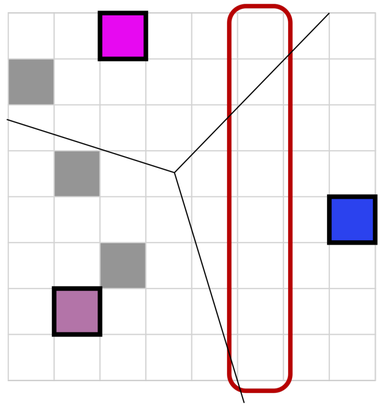
Stack based algorithm using a 3-ary *hiddenBy* predicate, *à la* sweep line ⇒ $O(n)$ per column

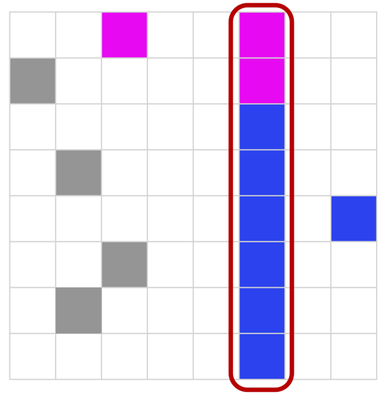
hiddenBy(u,v,w,S)



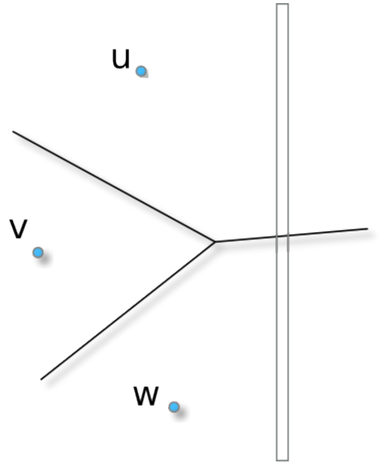






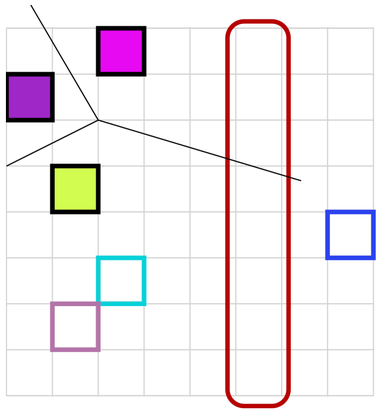


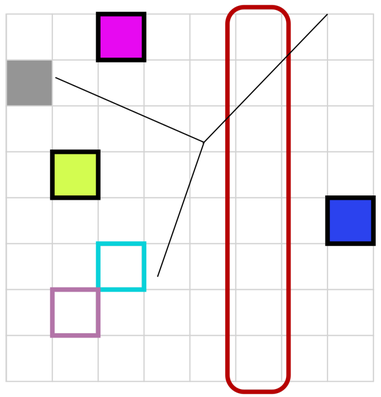
Separable Voronoi map: step 2

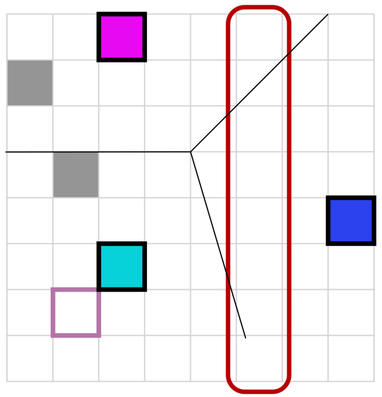


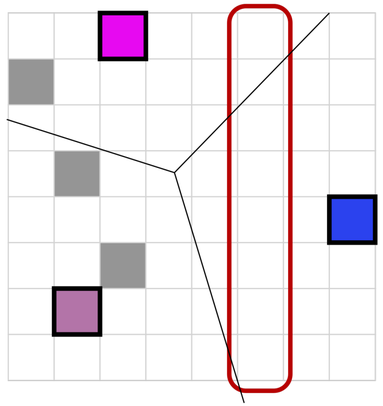
hiddenBy(u,v,w,S)

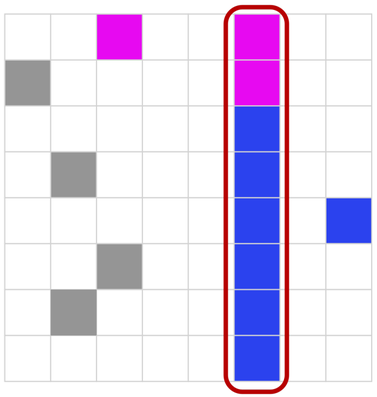
Stack based algorithm using a 3-ary *hiddenBy* predicate, *à la* sweep line $\Rightarrow O(n)$ per column

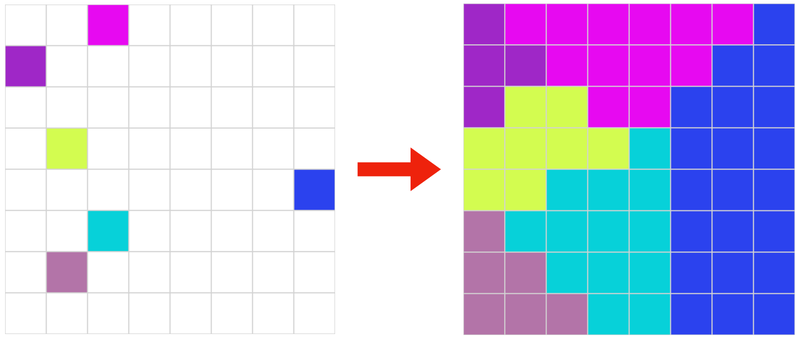




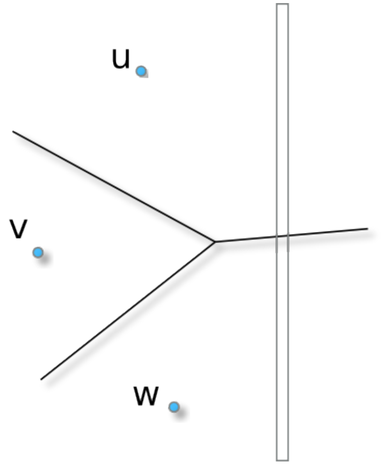






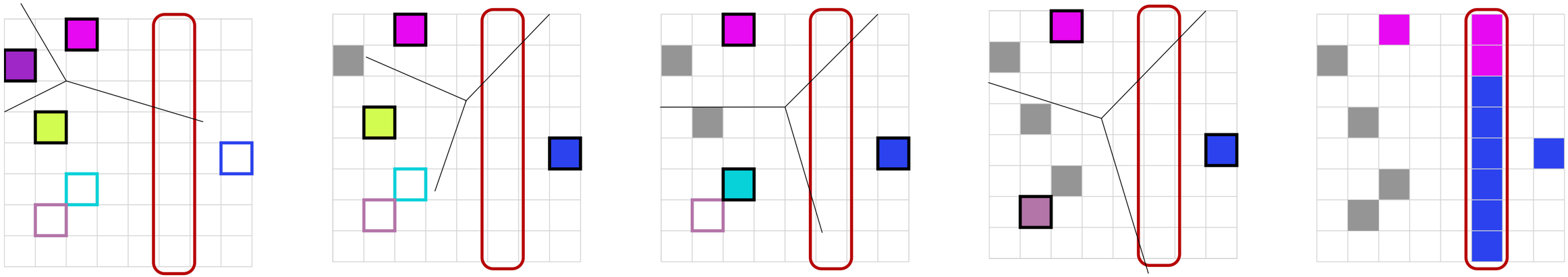


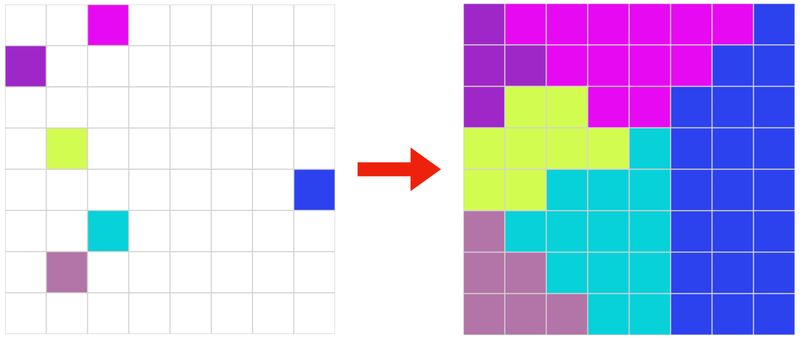
Separable Voronoi map: step 2



Stack based algorithm using a 3-ary *hiddenBy* predicate, *à la* sweep line ⇒ $O(n)$ per column

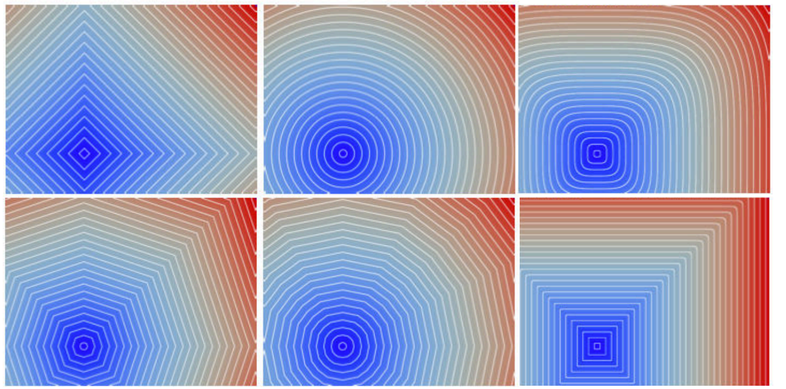
hiddenBy(u,v,w,S)

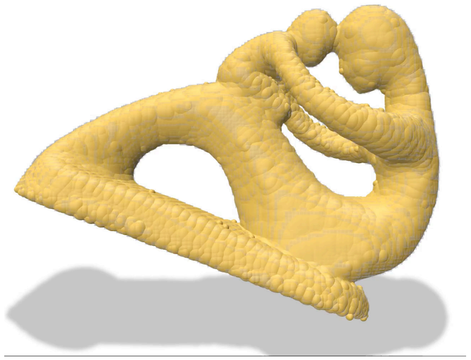




⇒ $O(n^2)$ in total

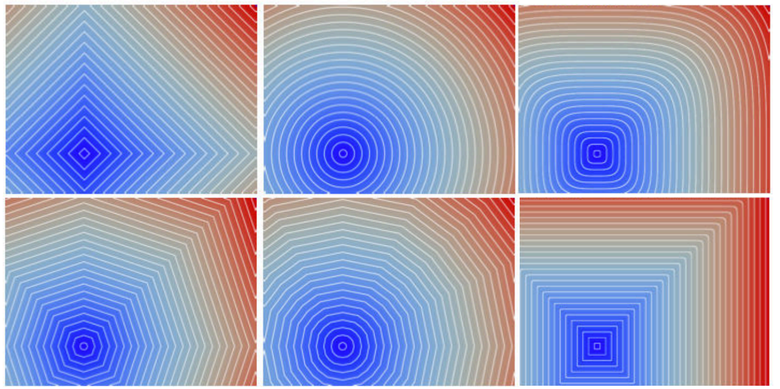
Separable approaches

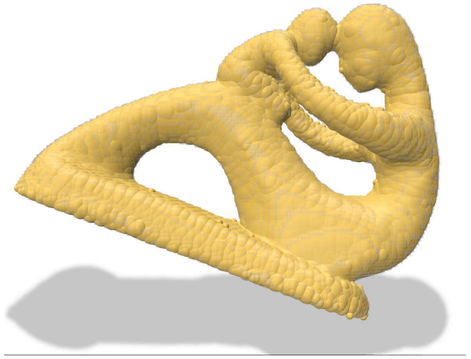




Separable approaches

The algorithm is correct:

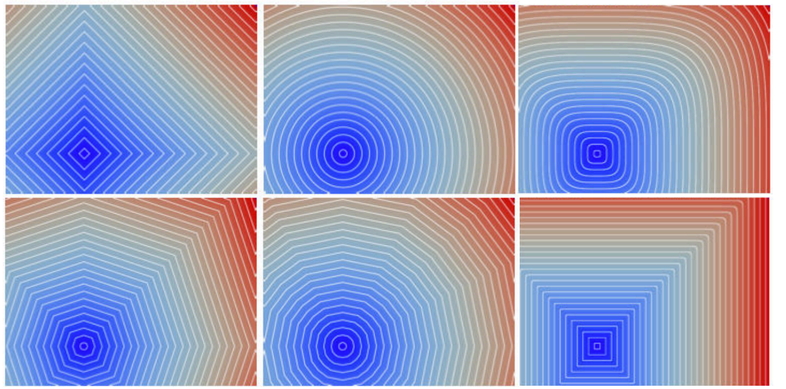


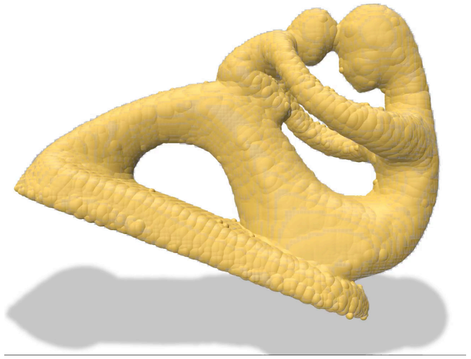


Separable approaches

The algorithm is correct:

- for any dimension

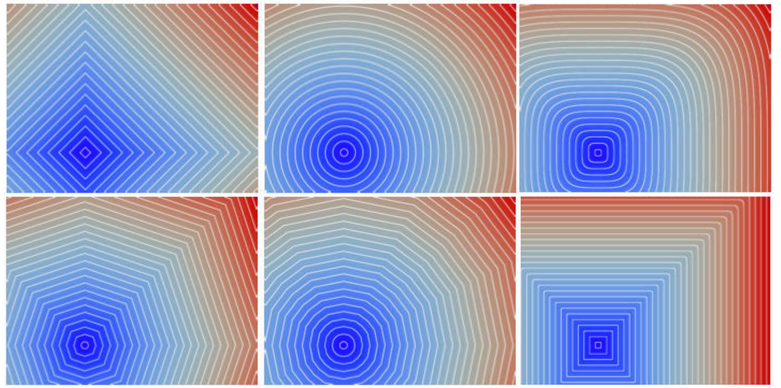


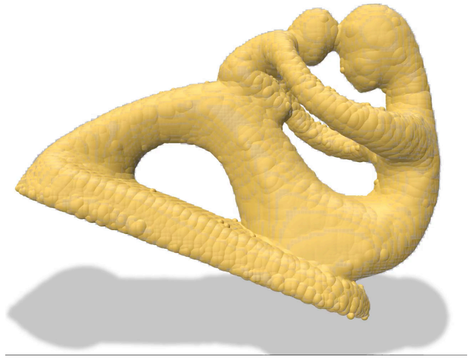


Separable approaches

The algorithm is correct:

- for any dimension
- for any metric with axis symmetric unit ball (e.g. any l_p)

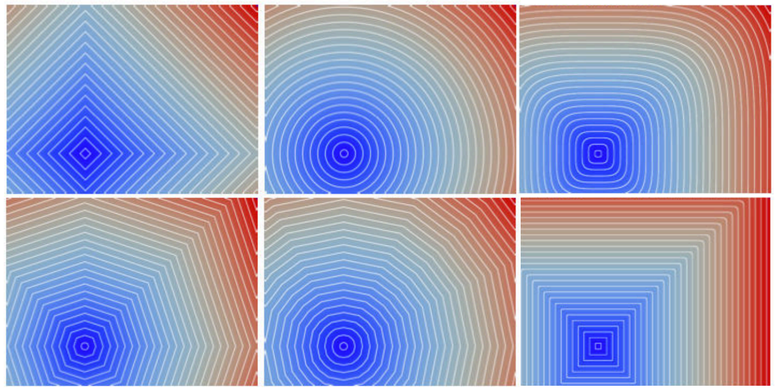


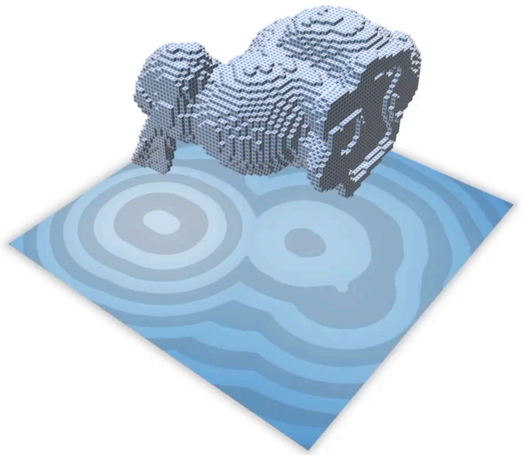


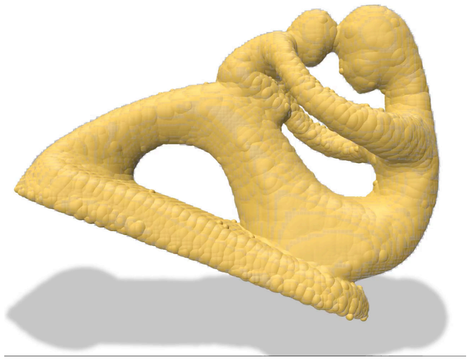
Separable approaches

The algorithm is correct:

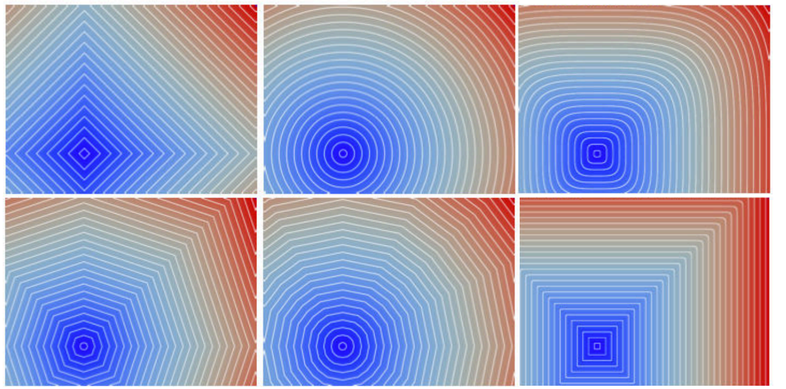
- for any dimension
- for any metric with axis symmetric unit ball (e.g. any l_p)
- on any toroidal nD domains







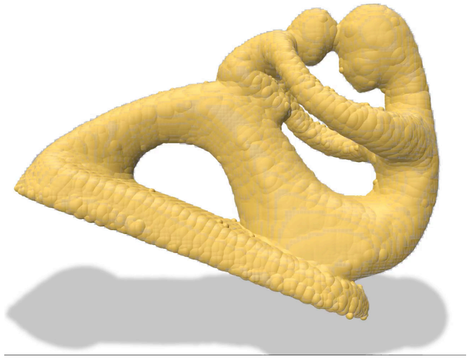
Separable approaches



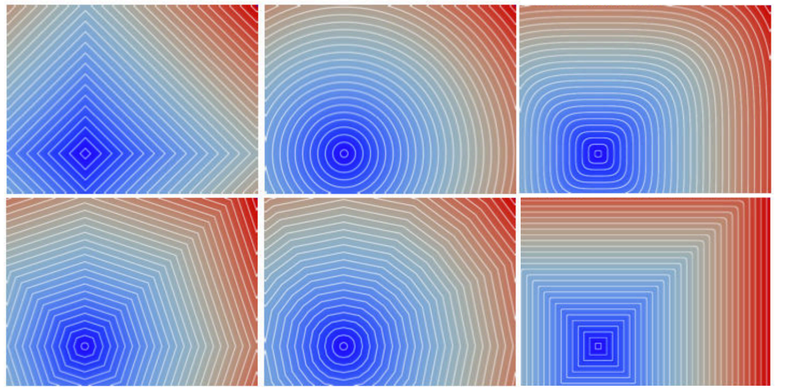
The algorithm is correct:

- for any dimension
- for any metric with axis symmetric unit ball (e.g. any l_p)
- on any toroidal nD domains

Exact and linear in time w.r.t. the number of grid points $O(d \cdot n^d)$ for l_2



Separable approaches

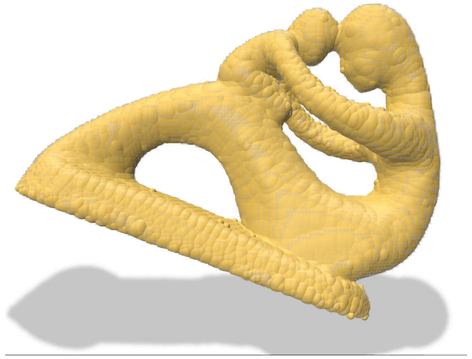


The algorithm is correct:

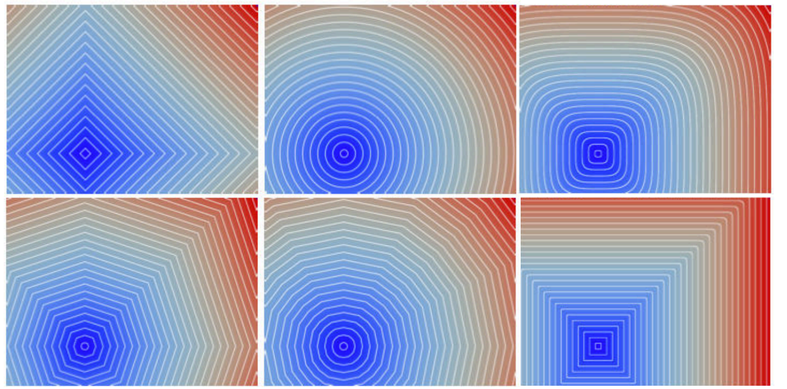
- for any dimension
- for any metric with axis symmetric unit ball (e.g. any l_p)
- on any toroidal nD domains

Exact and linear in time w.r.t. the number of grid points $O(d \cdot n^d)$ for l_2

$O(d^2 \cdot \log(p) \cdot \log(n) \cdot n^d)$ for exact l_p ($p \in \mathbb{Z}^+$), $O(d \cdot n^d)$ approx.



Separable approaches

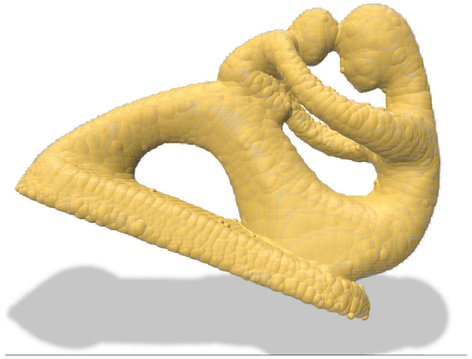


The algorithm is correct:

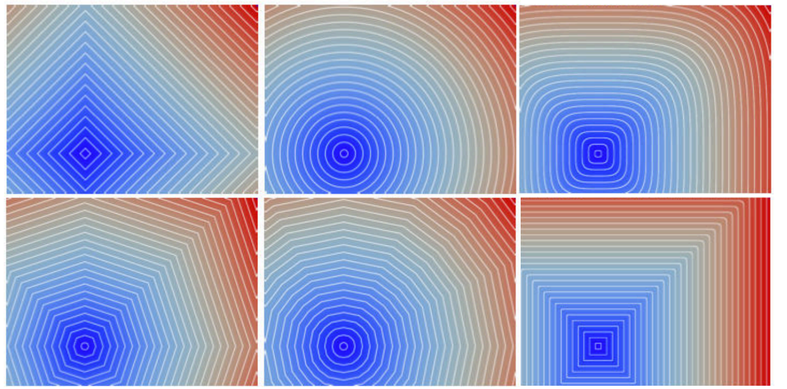
- for any dimension
- for any metric with axis symmetric unit ball (e.g. any l_p)
- on any toroidal nD domains

Exact and linear in time w.r.t. the number of grid points $O(d \cdot n^d)$ for l_2

$O(d^2 \cdot \log(p) \cdot \log(n) \cdot n^d)$ for exact l_p ($p \in \mathbb{Z}^+$), $O(d \cdot n^d)$ approx.



Separable approaches



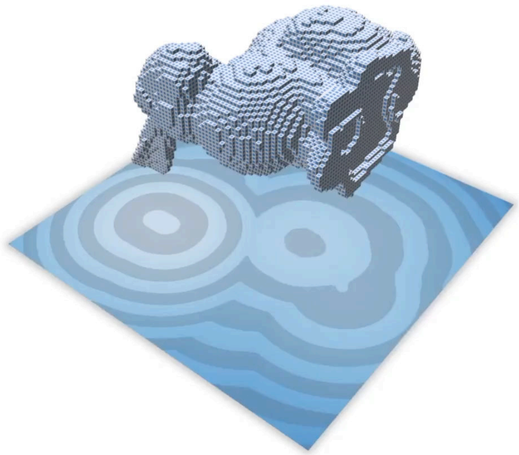
The algorithm is correct:

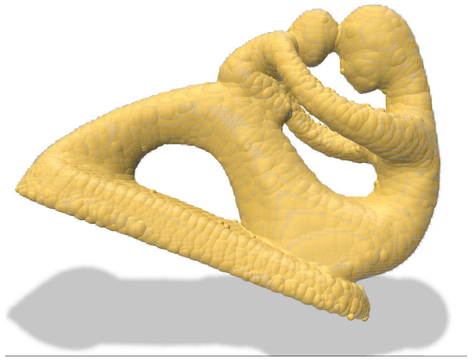
- for any dimension
- for any metric with axis symmetric unit ball (e.g. any l_p)
- on any toroidal nD domains

Exact and linear in time w.r.t. the number of grid points $O(d \cdot n^d)$ for l_2

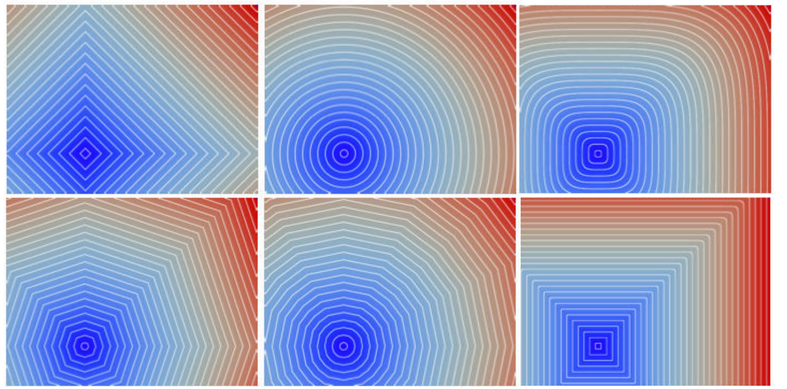
$O(d^2 \cdot \log(p) \cdot \log(n) \cdot n^d)$ for exact l_p ($p \in \mathbb{Z}^+$), $O(d \cdot n^d)$ approx.

Trivial multithread / GPU / out-of-core implementations





Separable approaches



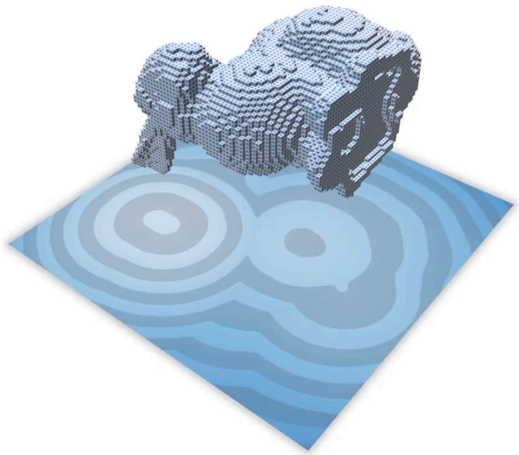
The algorithm is correct:

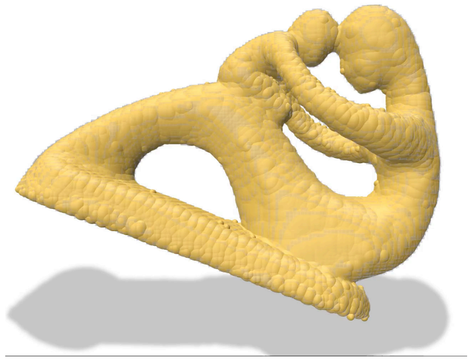
- for any dimension
- for any metric with axis symmetric unit ball (e.g. any l_p)
- on any toroidal nD domains

Exact and linear in time w.r.t. the number of grid points $O(d \cdot n^d)$ for l_2

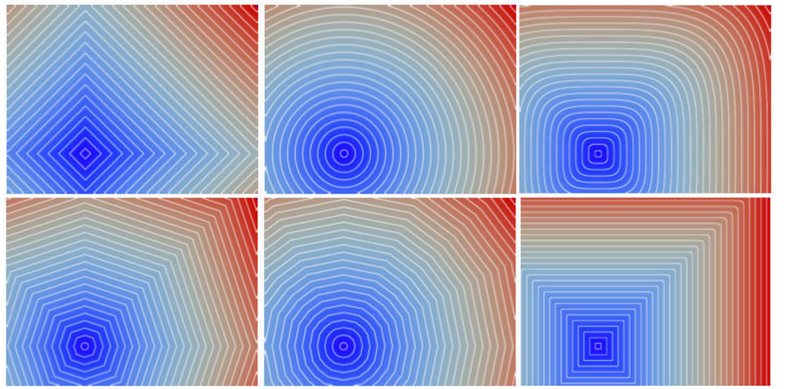
$O(d^2 \cdot \log(p) \cdot \log(n) \cdot n^d)$ for exact l_p ($p \in \mathbb{Z}^+$), $O(d \cdot n^d)$ approx.

Trivial multithread / GPU / out-of-core implementations





Separable approaches



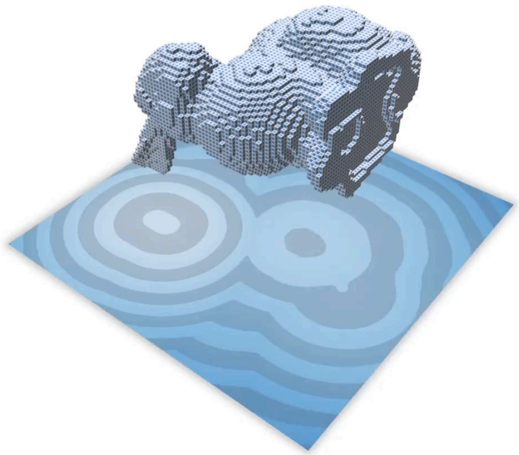
The algorithm is correct:

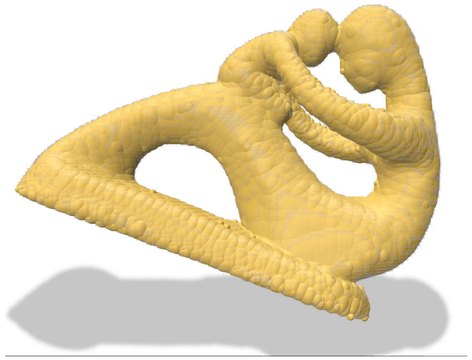
- for any dimension
- for any metric with axis symmetric unit ball (e.g. any l_p)
- on any toroidal nD domains

Exact and linear in time w.r.t. the number of grid points $O(d \cdot n^d)$ for l_2

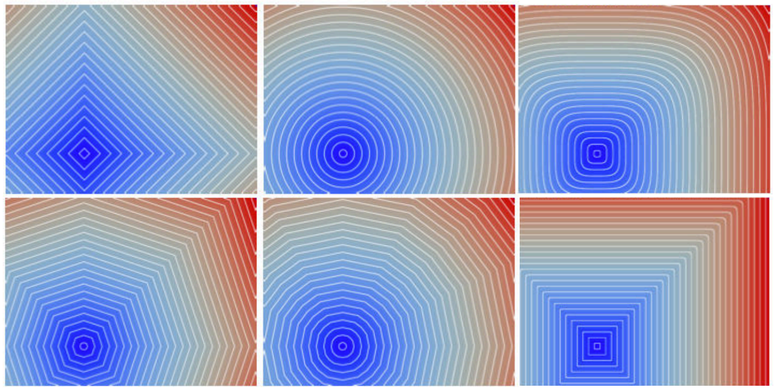
$O(d^2 \cdot \log(p) \cdot \log(n) \cdot n^d)$ for exact l_p ($p \in \mathbb{Z}^+$), $O(d \cdot n^d)$ approx.

Trivial multithread / GPU / out-of-core implementations





Separable approaches



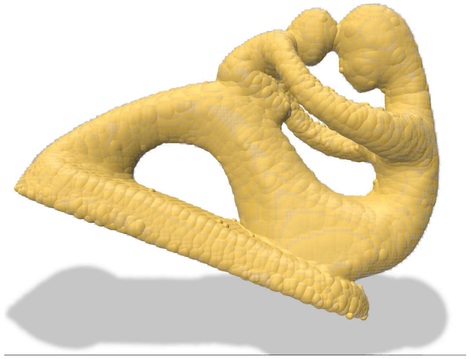
The algorithm is correct:

- for any dimension
- for any metric with axis symmetric unit ball (e.g. any l_p)
- on any toroidal nD domains

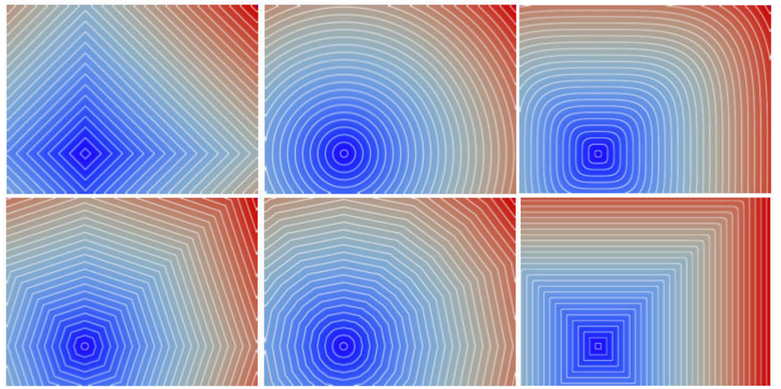
Exact and linear in time w.r.t. the number of grid points $O(d \cdot n^d)$ for l_2

$O(d^2 \cdot \log(p) \cdot \log(n) \cdot n^d)$ for exact l_p ($p \in \mathbb{Z}^+$), $O(d \cdot n^d)$ approx.

Trivial multithread / GPU / out-of-core implementations



Separable approaches



The algorithm is correct:

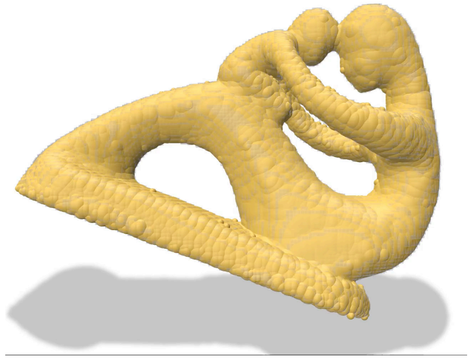
- for any dimension
- for any metric with axis symmetric unit ball (e.g. any l_p)
- on any toroidal nD domains

Exact and linear in time w.r.t. the number of grid points $O(d \cdot n^d)$ for l_2

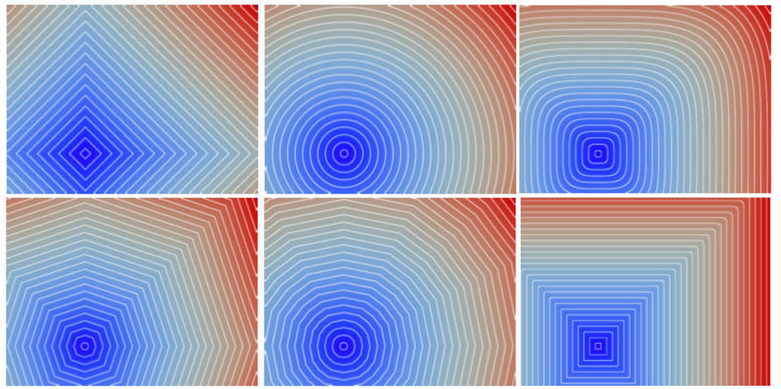
$O(d^2 \cdot \log(p) \cdot \log(n) \cdot n^d)$ for exact l_p ($p \in \mathbb{Z}^+$), $O(d \cdot n^d)$ approx.

Trivial multithread / GPU / out-of-core implementations

Same techniques and computational costs for: [C. et al 07]



Separable approaches



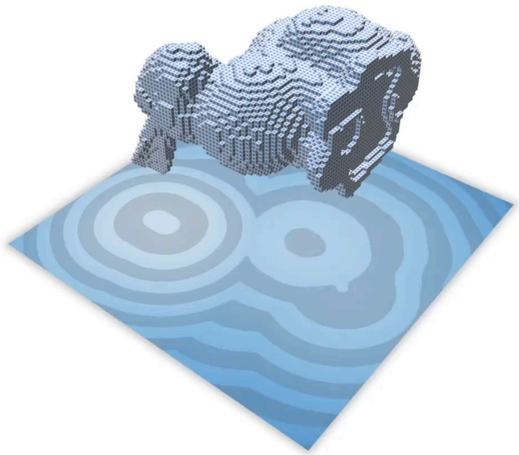
The algorithm is correct:

- for any dimension
- for any metric with axis symmetric unit ball (e.g. any l_p)
- on any toroidal nD domains

Exact and linear in time w.r.t. the number of grid points $O(d \cdot n^d)$ for l_2

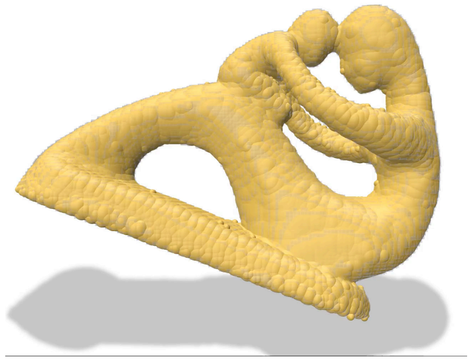
$O(d^2 \cdot \log(p) \cdot \log(n) \cdot n^d)$ for exact l_p ($p \in \mathbb{Z}^+$), $O(d \cdot n^d)$ approx.

Trivial multithread / GPU / out-of-core implementations

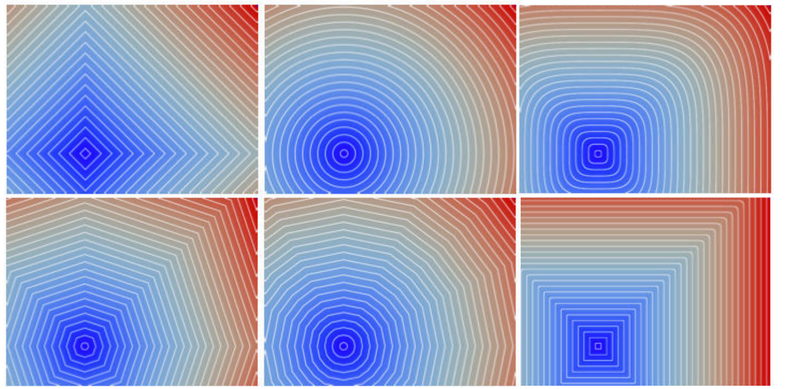


Same techniques and computational costs for: [C. et al 07]

- Power diagram / power maps construction



Separable approaches



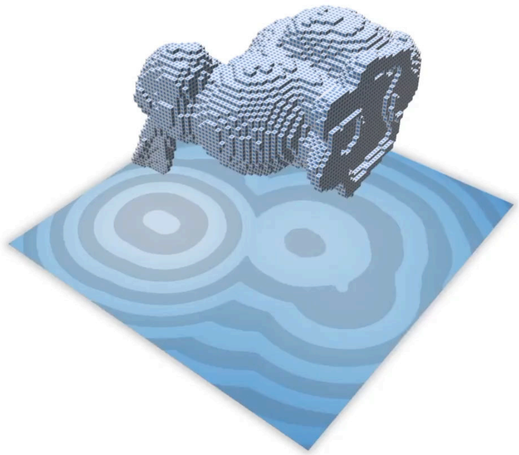
The algorithm is correct:

- for any dimension
- for any metric with axis symmetric unit ball (e.g. any l_p)
- on any toroidal nD domains

Exact and linear in time w.r.t. the number of grid points $O(d \cdot n^d)$ for l_2

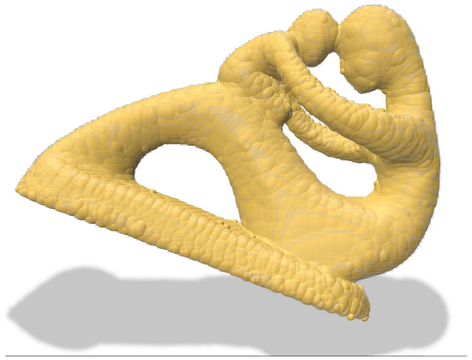
$O(d^2 \cdot \log(p) \cdot \log(n) \cdot n^d)$ for exact l_p ($p \in \mathbb{Z}^+$), $O(d \cdot n^d)$ approx.

Trivial multithread / GPU / out-of-core implementations

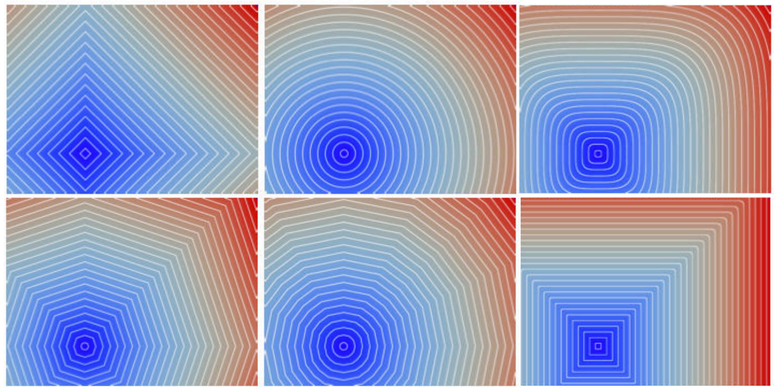


Same techniques and computational costs for: [C. et al 07]

- Power diagram / power maps construction
- Discrete Medial Axis extraction (aka non-empty inner power cells)



Separable approaches



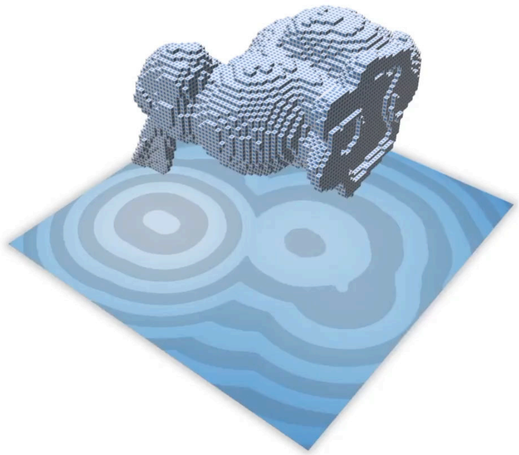
The algorithm is correct:

- for any dimension
- for any metric with axis symmetric unit ball (e.g. any l_p)
- on any toroidal nD domains

Exact and linear in time w.r.t. the number of grid points $O(d \cdot n^d)$ for l_2

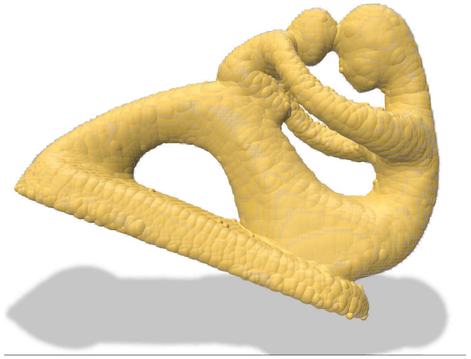
$O(d^2 \cdot \log(p) \cdot \log(n) \cdot n^d)$ for exact l_p ($p \in \mathbb{Z}^+$), $O(d \cdot n^d)$ approx.

Trivial multithread / GPU / out-of-core implementations

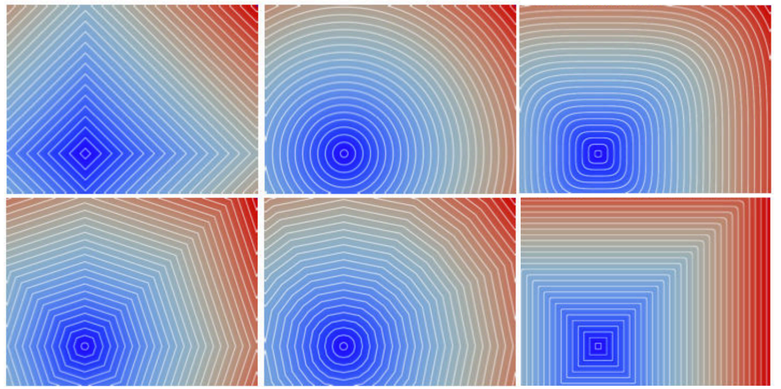


Same techniques and computational costs for: [C. et al 07]

- Power diagram / power maps construction
- Discrete Medial Axis extraction (aka non-empty inner power cells)
- Reverse reconstruction (balls→shape)



Separable approaches



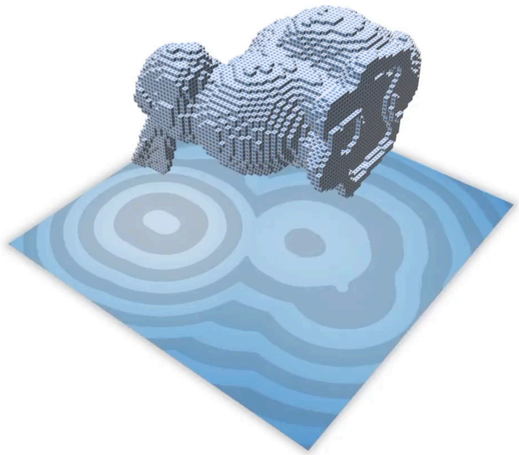
The algorithm is correct:

- for any dimension
- for any metric with axis symmetric unit ball (e.g. any l_p)
- on any toroidal nD domains

Exact and linear in time w.r.t. the number of grid points $O(d \cdot n^d)$ for l_2

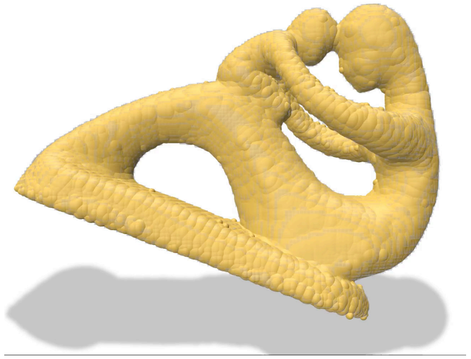
$O(d^2 \cdot \log(p) \cdot \log(n) \cdot n^d)$ for exact l_p ($p \in \mathbb{Z}^+$), $O(d \cdot n^d)$ approx.

Trivial multithread / GPU / out-of-core implementations

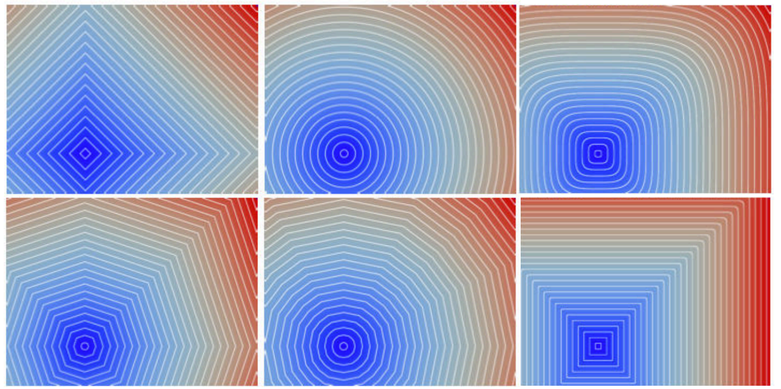


Same techniques and computational costs for: [C. et al 07]

- Power diagram / power maps construction
- Discrete Medial Axis extraction (aka non-empty inner power cells)
- Reverse reconstruction (balls→shape)



Separable approaches



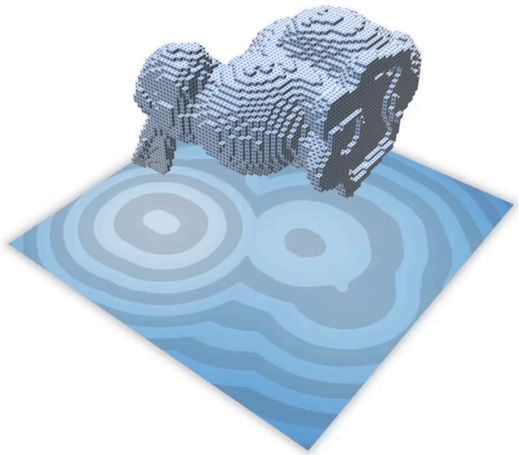
The algorithm is correct:

- for any dimension
- for any metric with axis symmetric unit ball (e.g. any l_p)
- on any toroidal nD domains

Exact and linear in time w.r.t. the number of grid points $O(d \cdot n^d)$ for l_2

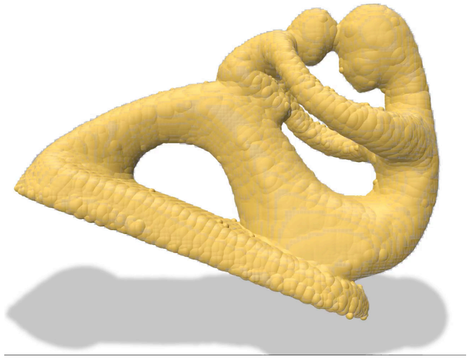
$O(d^2 \cdot \log(p) \cdot \log(n) \cdot n^d)$ for exact l_p ($p \in \mathbb{Z}^+$), $O(d \cdot n^d)$ approx.

Trivial multithread / GPU / out-of-core implementations



Same techniques and computational costs for: [C. et al 07]

- Power diagram / power maps construction
- Discrete Medial Axis extraction (aka non-empty inner power cells)
- Reverse reconstruction (balls→shape)



Separable approaches

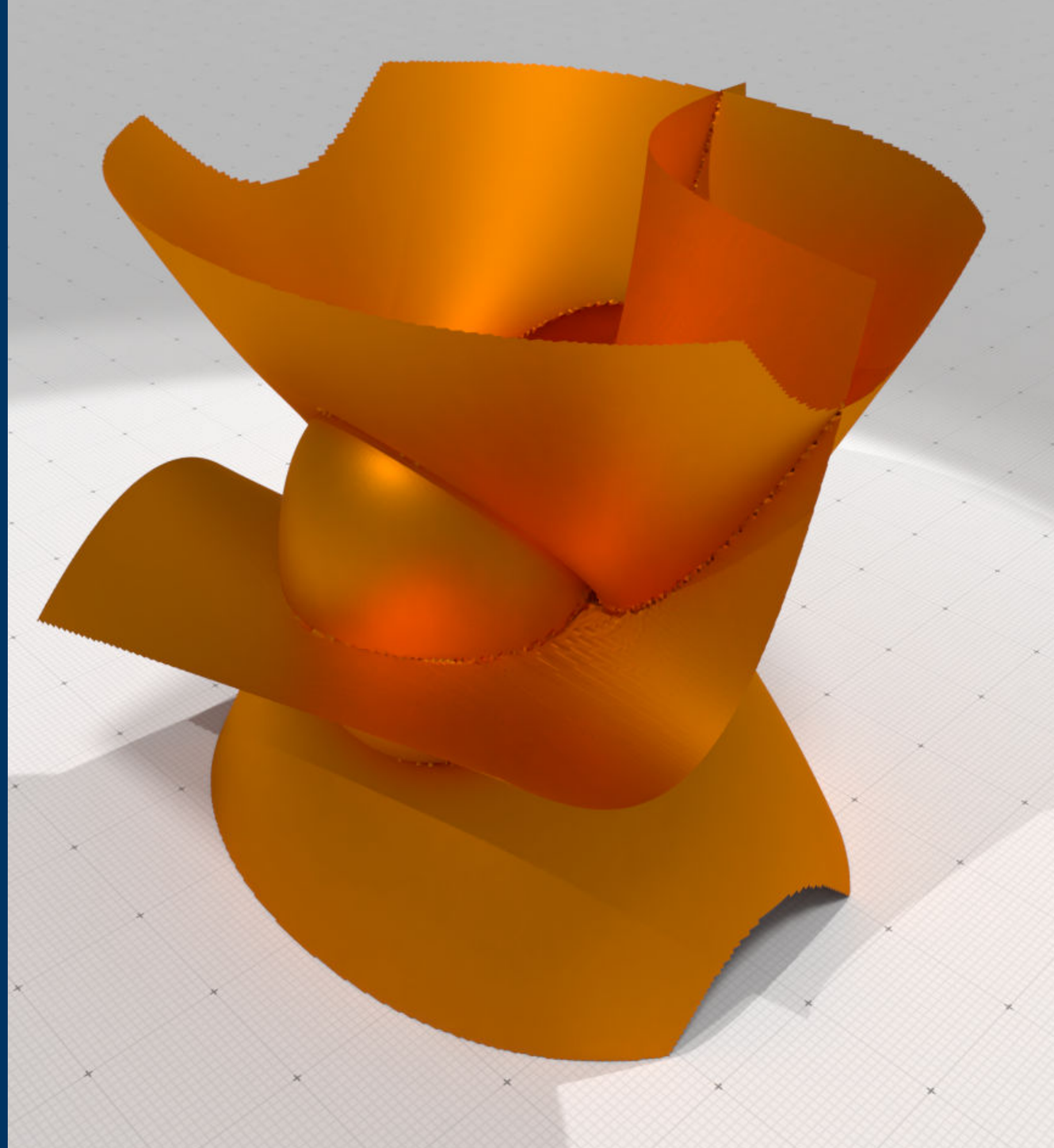
Alternatives: Jump flooding, Fast Marching Methods or distance propagation… but only approximation and/or non-linear complexity (e.g. $O(n^2 \log n)$ in 2D for FMM)

Limitations: full ambient space computation (i.e. no geodesic, use FMM or PDE based approaches instead)

But

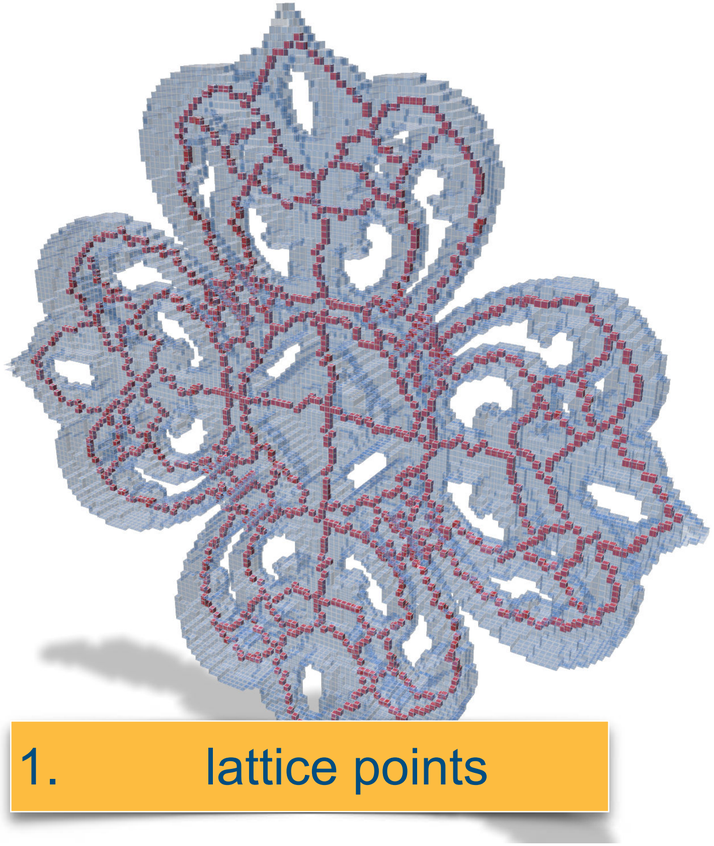
- range based / dexel based approach for faster computations [Chen et al 2020]
- narrow band approaches
- some extensions to hierarchical / adaptive grids
- can use sub-pixel information (coverage, QEM, …)
- …

topology on $\mathbb{Z}^d$

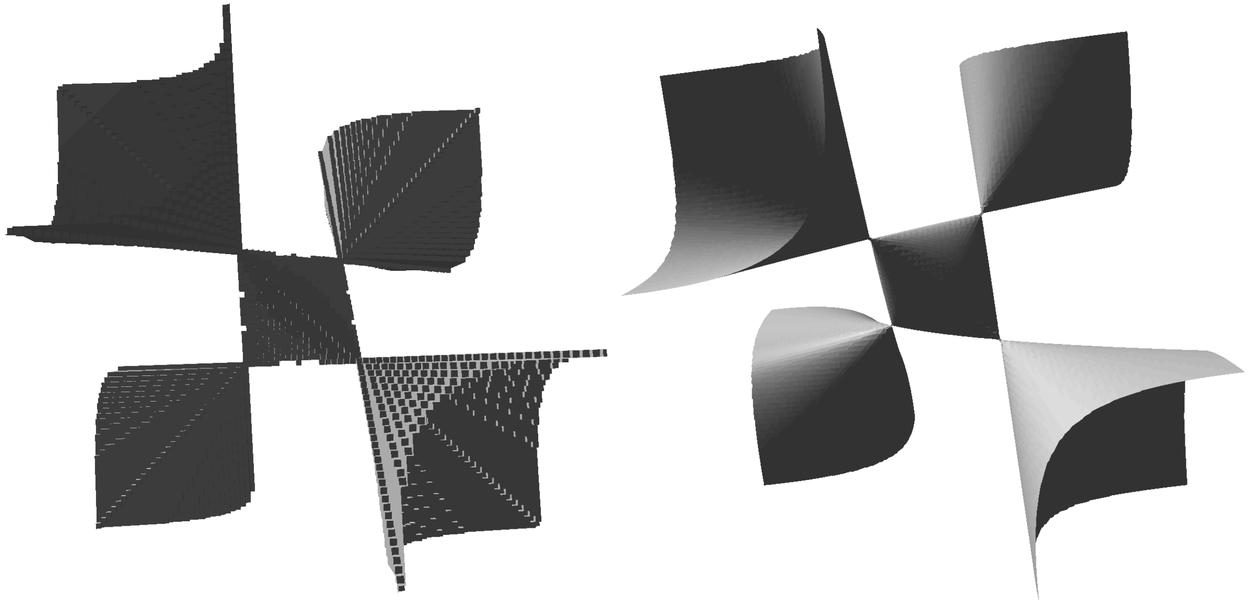


Before geometry : topological models for $\mathbb{Z}^d$

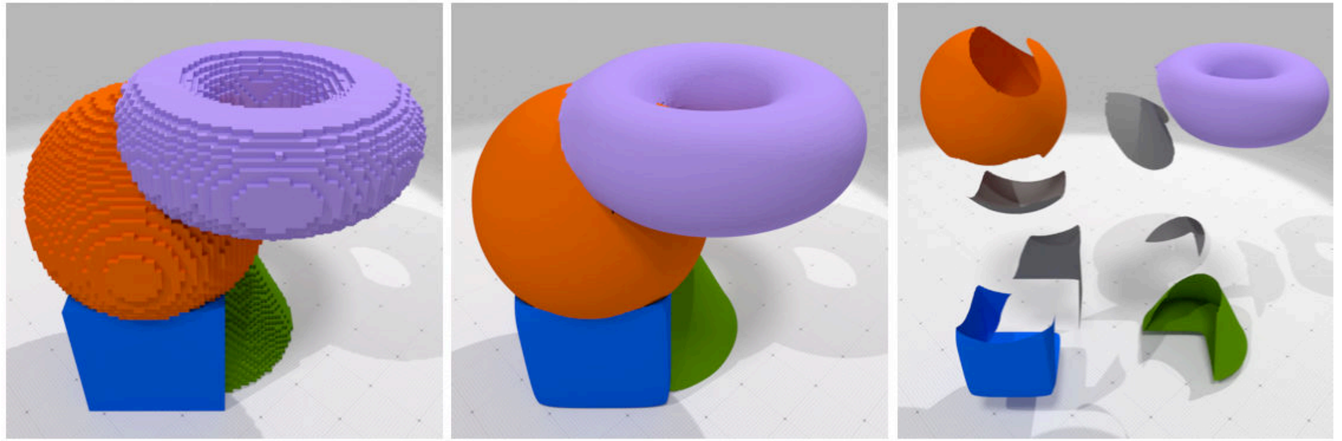
How to represent volumes, boundaries, curves, surfaces, partitions ?



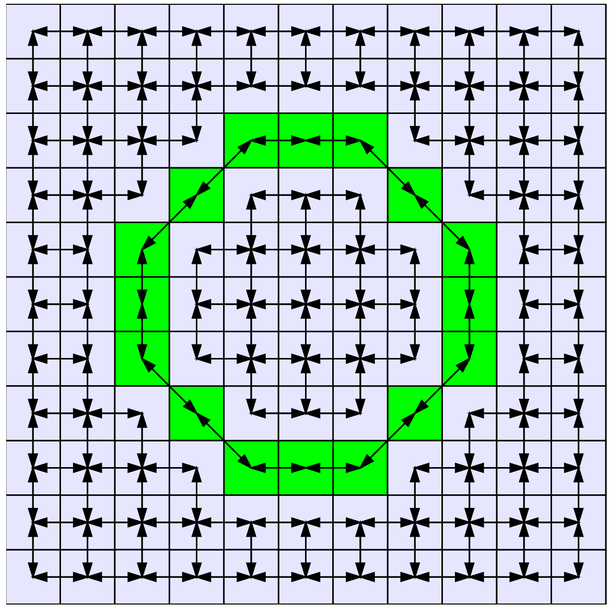
1. lattice points



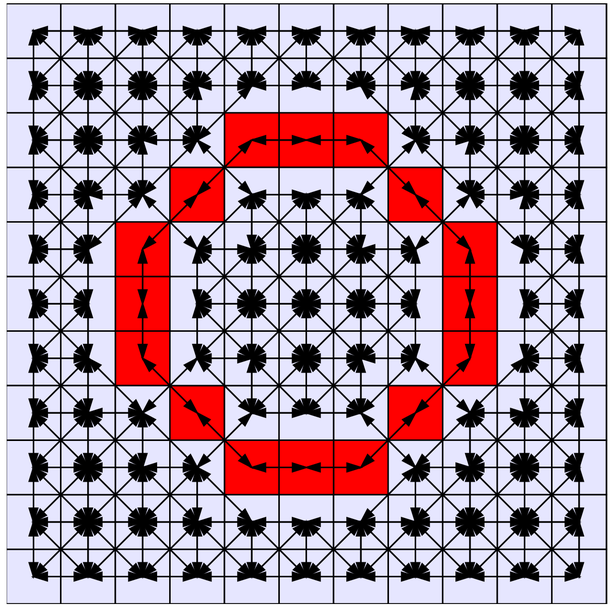
2. cubical complexes



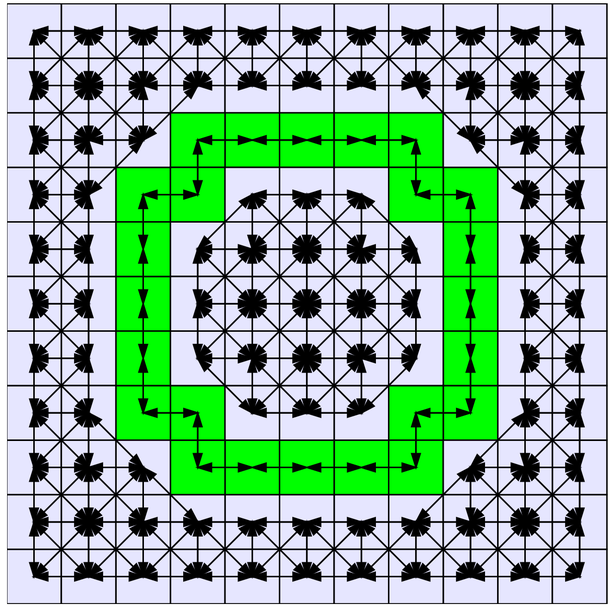
Digital topology



(8,4)-topology



(8,8)-topology

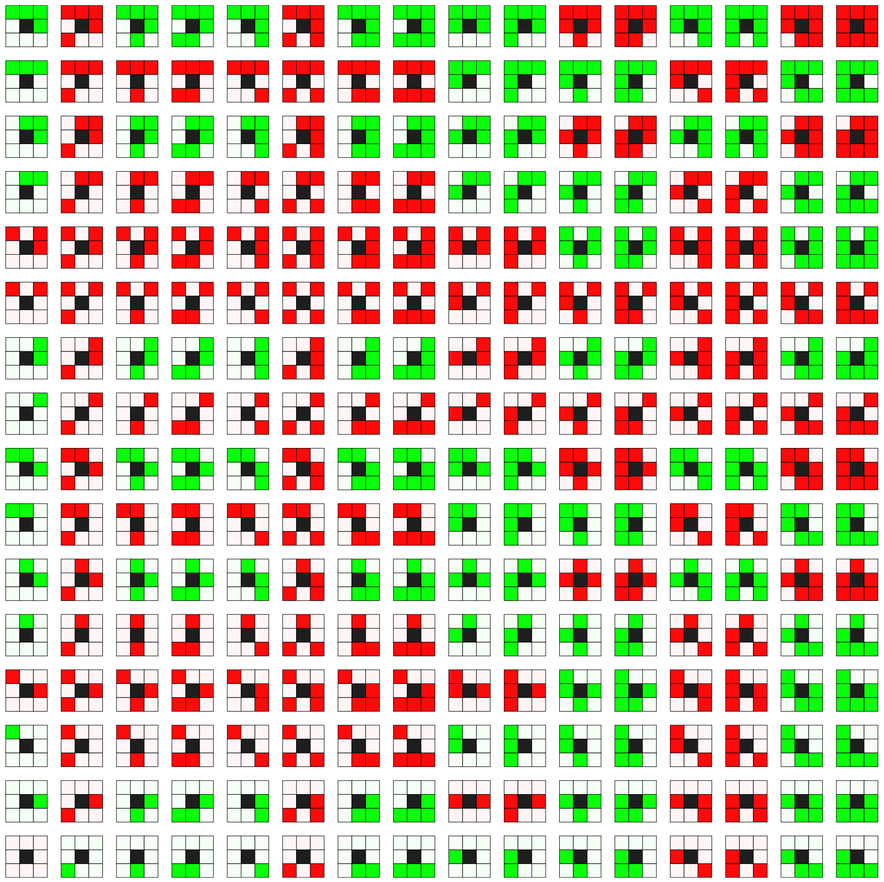


(4,8)-topology

Good adjacencies for object/background

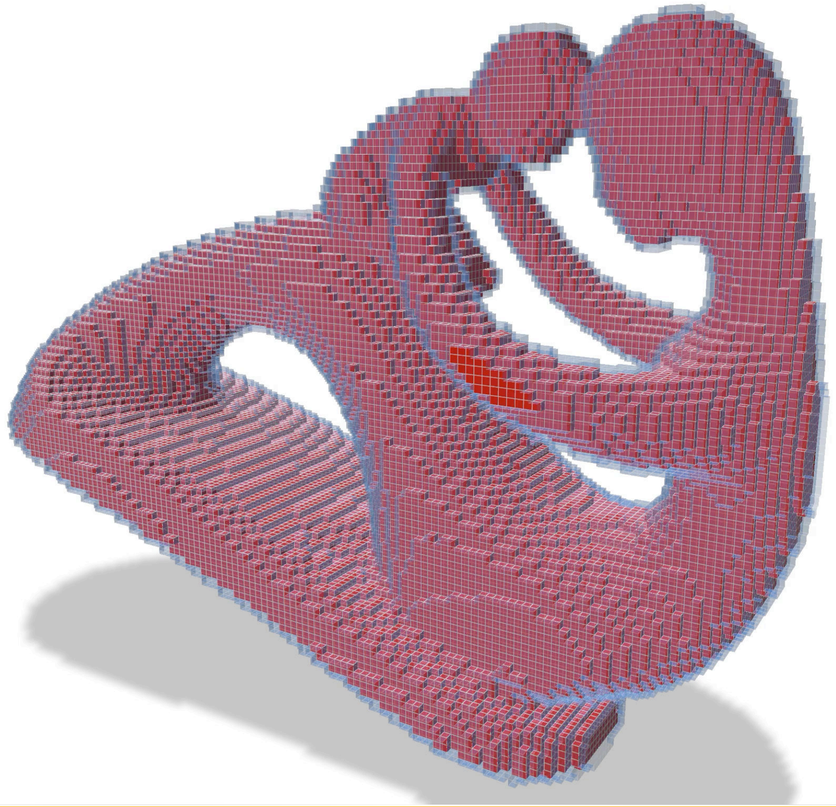
- Jordan separation theorem
- consistence borders and interior components
- definition of surfaces in $\mathbb{Z}^d$

Topology invariance: simple points



(8,4)-topology

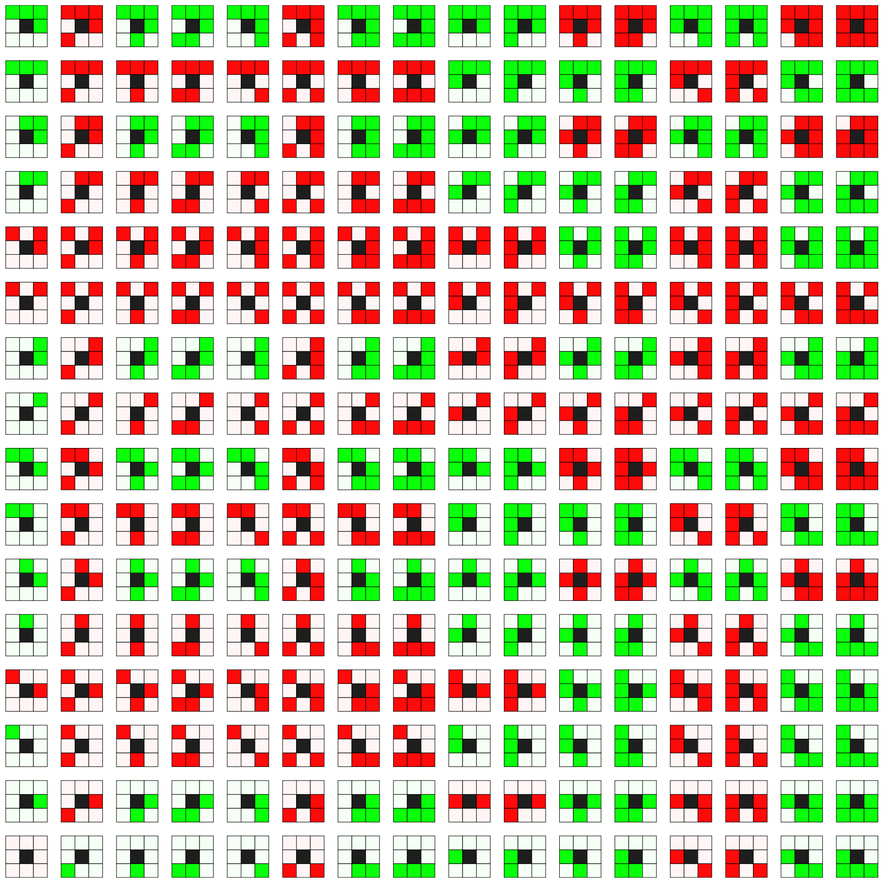
locally keep connected components



Simple points: points whose removal preserves topology

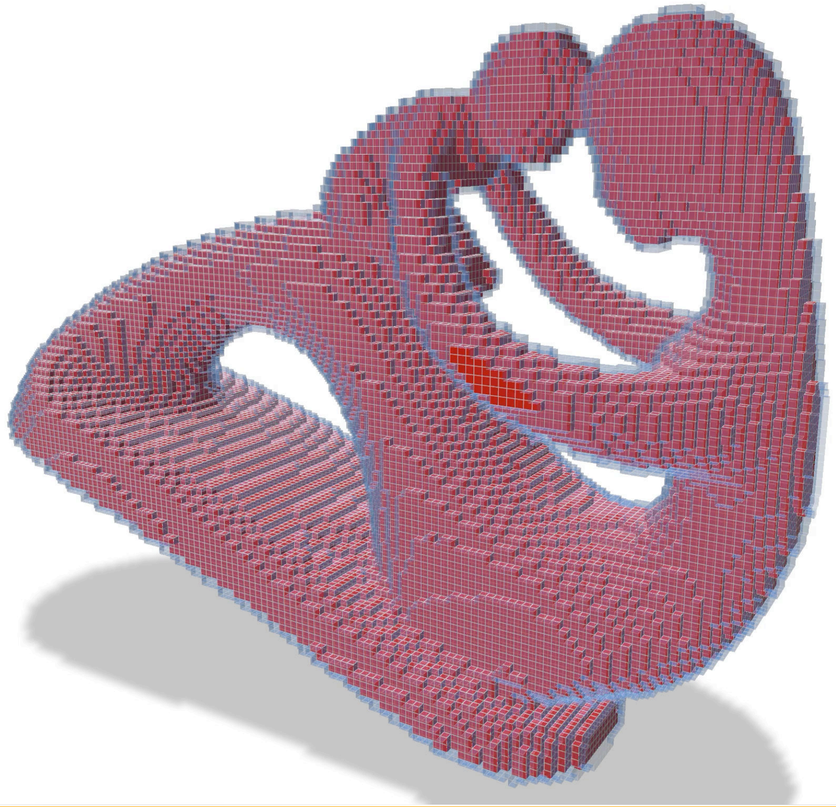
- digital topology invariance of object and background
- very fast: look-up tables in 2D and 3D
- useful for skeleton extraction / coupled with medial axis

Topology invariance: simple points



(8,4)-topology

locally keep connected components



Simple points: points whose removal preserves topology

- digital topology invariance of object and background
- very fast: look-up tables in 2D and 3D
- useful for skeleton extraction / coupled with medial axis

hands on…

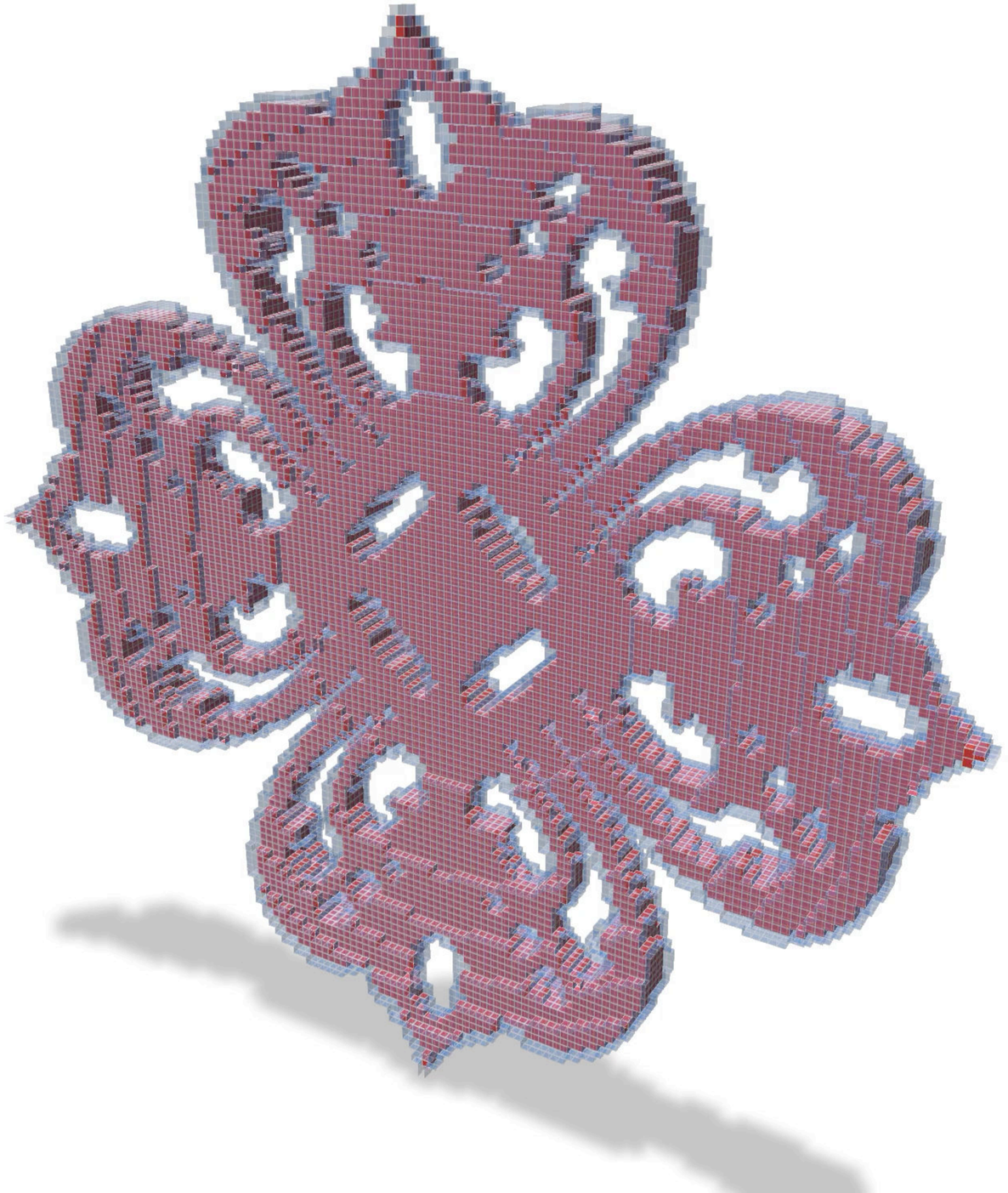
```cpp
// Build object with digital topology
const auto K = SH3::getKSpace( binary_image );
Domain domain( K.lowerBound(), K.upperBound() );
Z3i::DigitalSet voxel_set( domain );
for ( auto p : domain )
  if ( (*binary_image)( p ) ) voxel_set.insertNew( p );
the_object = CountedPtr< Z3i::Object26_6 >( new Z3i::Object26_6( dt26_6, voxel_set ) );
the_object->setTable(functions::loadTable<3>(simplicity::tableSimple26_6));
```

Create object with (26,6) topology from binary image

```cpp
// Removes a peel of simple points onto voxel object.
bool oneStep( CountedPtr< Z3i::Object26_6 > object )
{
  DigitalSet & S = object->pointSet();
  std::queue< Point > Q;
  for ( auto&& p : S )
    if ( object->isSimple( p ) )
      Q.push( p );
  int nb_simple = 0;
  while ( ! Q.empty() )
    {
      const auto p = Q.front();
      Q.pop();
      if ( object->isSimple( p ) )
        {
          S.erase( p );
          binary_image->setValue( p, false );
          ++nb_simple;
        }
    }
  trace.info() << "Removed " << nb_simple << " / " << S.size()
               << " points." << std::endl;
  registerDigitalSurface( binary_image, "Thinned object" );
  return nb_simple == 0;
}
```

Queue simple points

Remove simple points



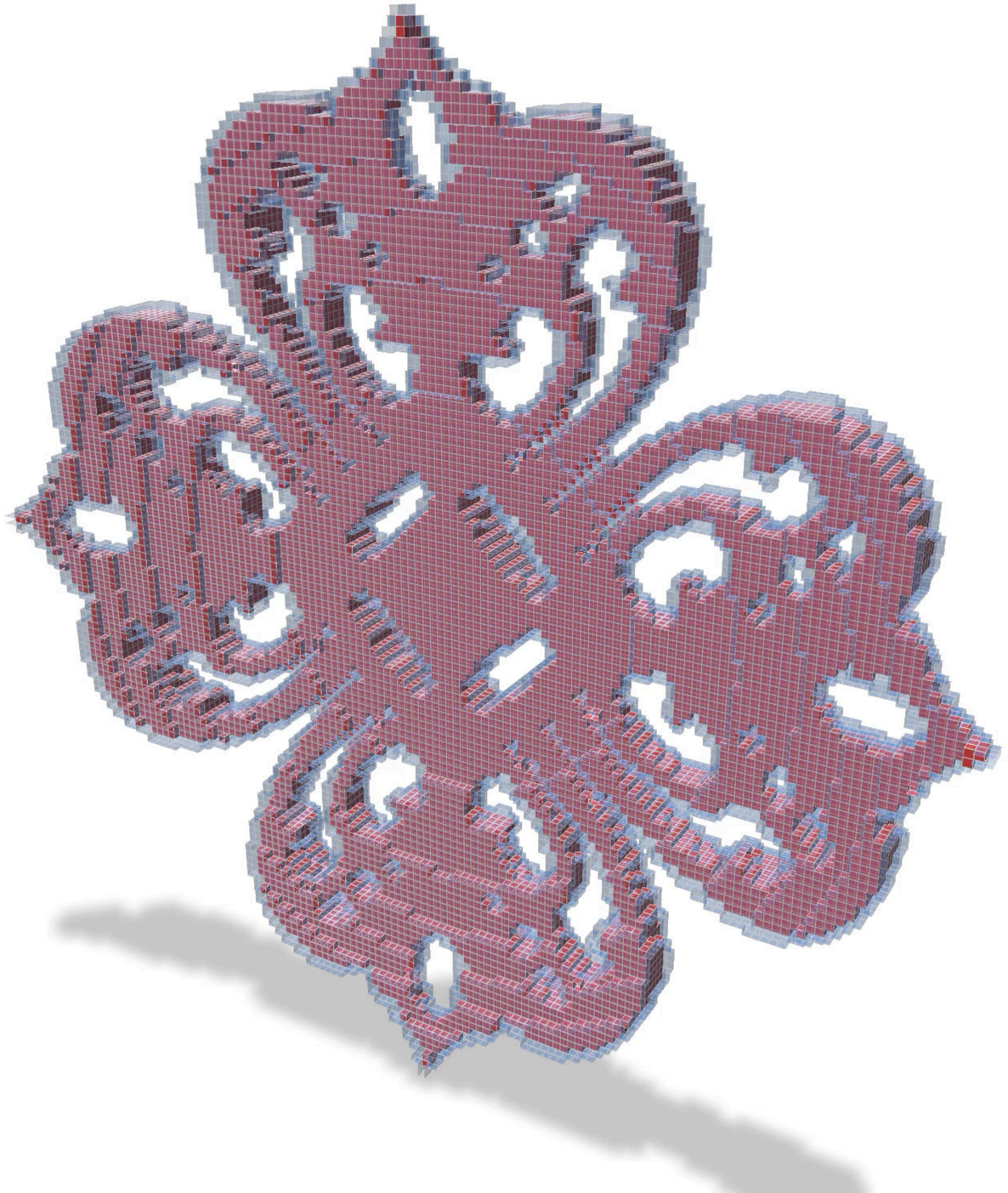
```cpp
// Build object with digital topology
const auto K = SH3::getKSpace( binary_image );
Domain domain( K.lowerBound(), K.upperBound() );
Z3i::DigitalSet voxel_set( domain );
for ( auto p : domain )
  if ( (*binary_image)( p ) ) voxel_set.insertNew( p );
the_object = CountedPtr< Z3i::Object26_6 >( new Z3i::Object26_6( dt26_6, voxel_set ) );
the_object->setTable(functions::loadTable<3>(simplicity::tableSimple26_6));
```

Create object with (26,6) topology from binary image

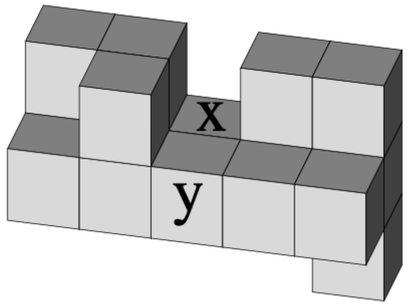
```cpp
// Removes a peel of simple points onto voxel object.
bool oneStep( CountedPtr< Z3i::Object26_6 > object )
{
  DigitalSet & S = object->pointSet();
  std::queue< Point > Q;
  for ( auto&& p : S )
    if ( object->isSimple( p ) )
      Q.push( p );
  int nb_simple = 0;
  while ( ! Q.empty() )
    {
      const auto p = Q.front();
      Q.pop();
      if ( object->isSimple( p ) )
        {
          S.erase( p );
          binary_image->setValue( p, false );
          ++nb_simple;
        }
    }
  trace.info() << "Removed " << nb_simple << " / " << S.size()
               << " points." << std::endl;
  registerDigitalSurface( binary_image, "Thinned object" );
  return nb_simple == 0;
}
```

Queue simple points

Remove simple points

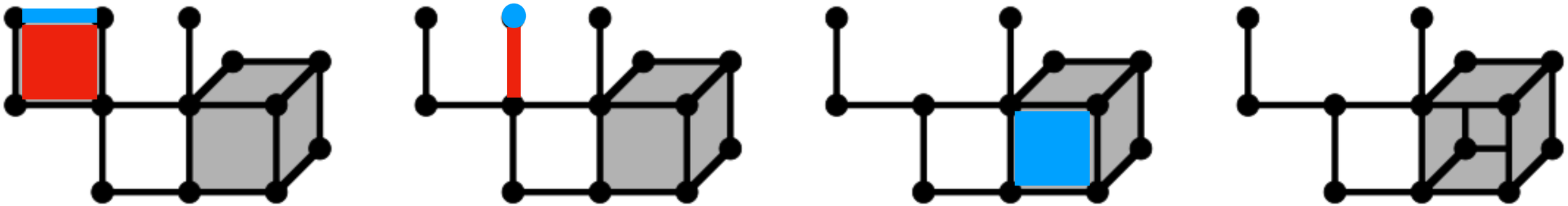


Homotopic collapses



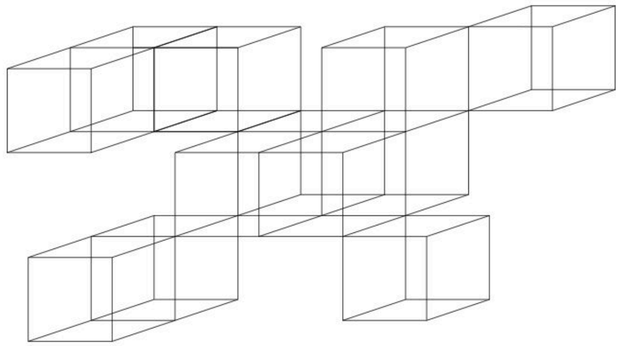
x and y are simple
but cannot be removed in parallel

Needs cubical complex representation

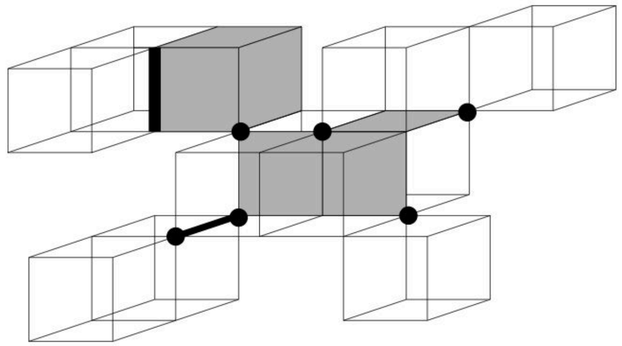


Elementary **collapse** : removing cell pairs (f,g) where g is free
preserves homotopy

Homotopic collapses and critical kernels

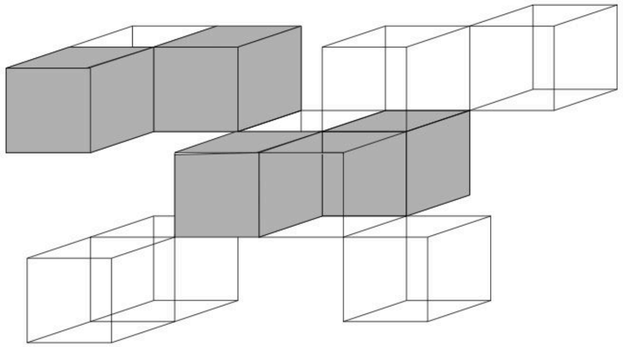


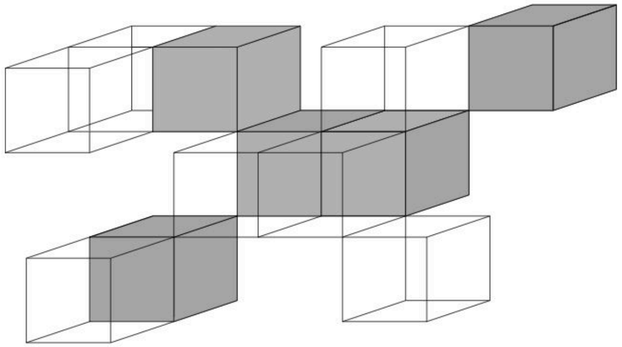
cubical complex X



Z := critical kernel of X

critical cells : cells that do not collapse onto their neighborhood



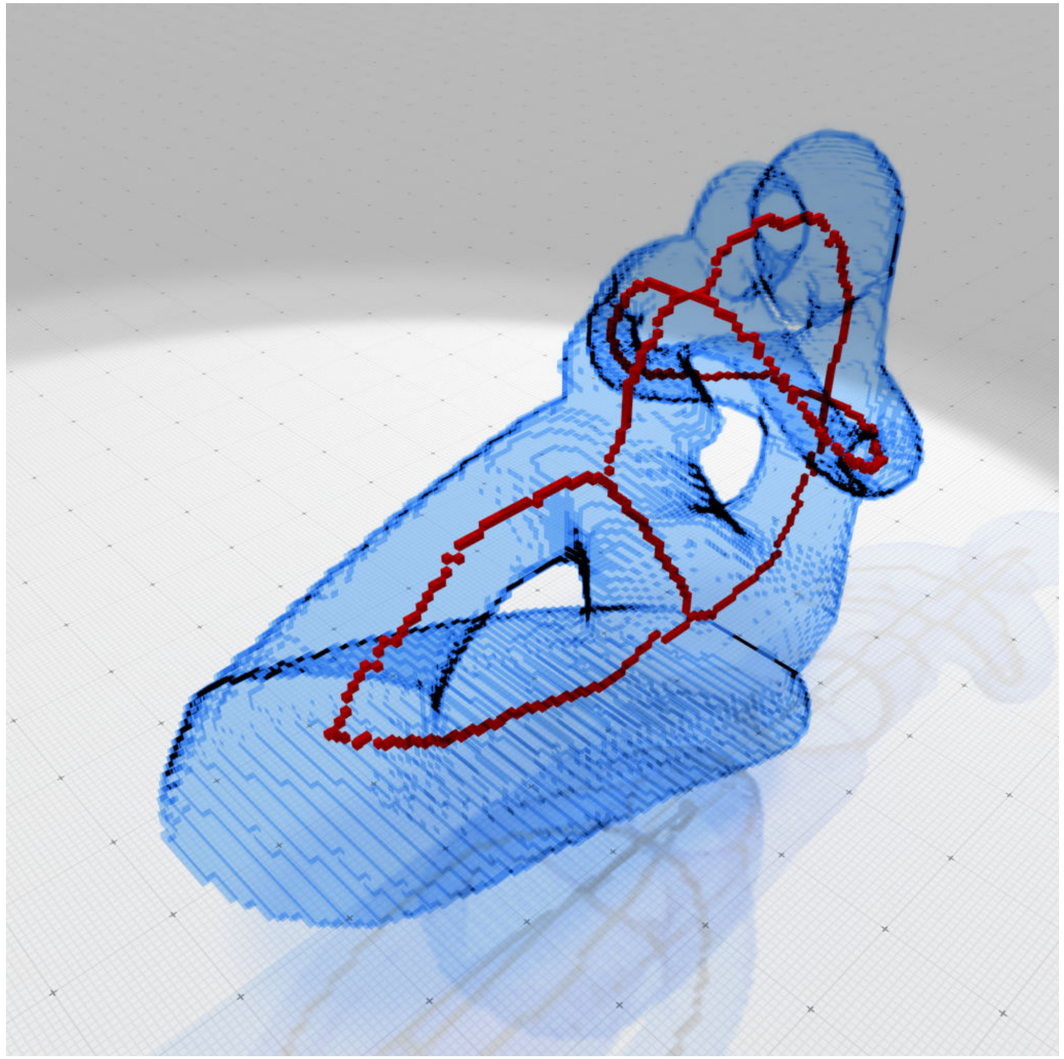


Both complexes Y_1, Y_2 are thinning, since $Z \subseteq Y_i \subseteq X$

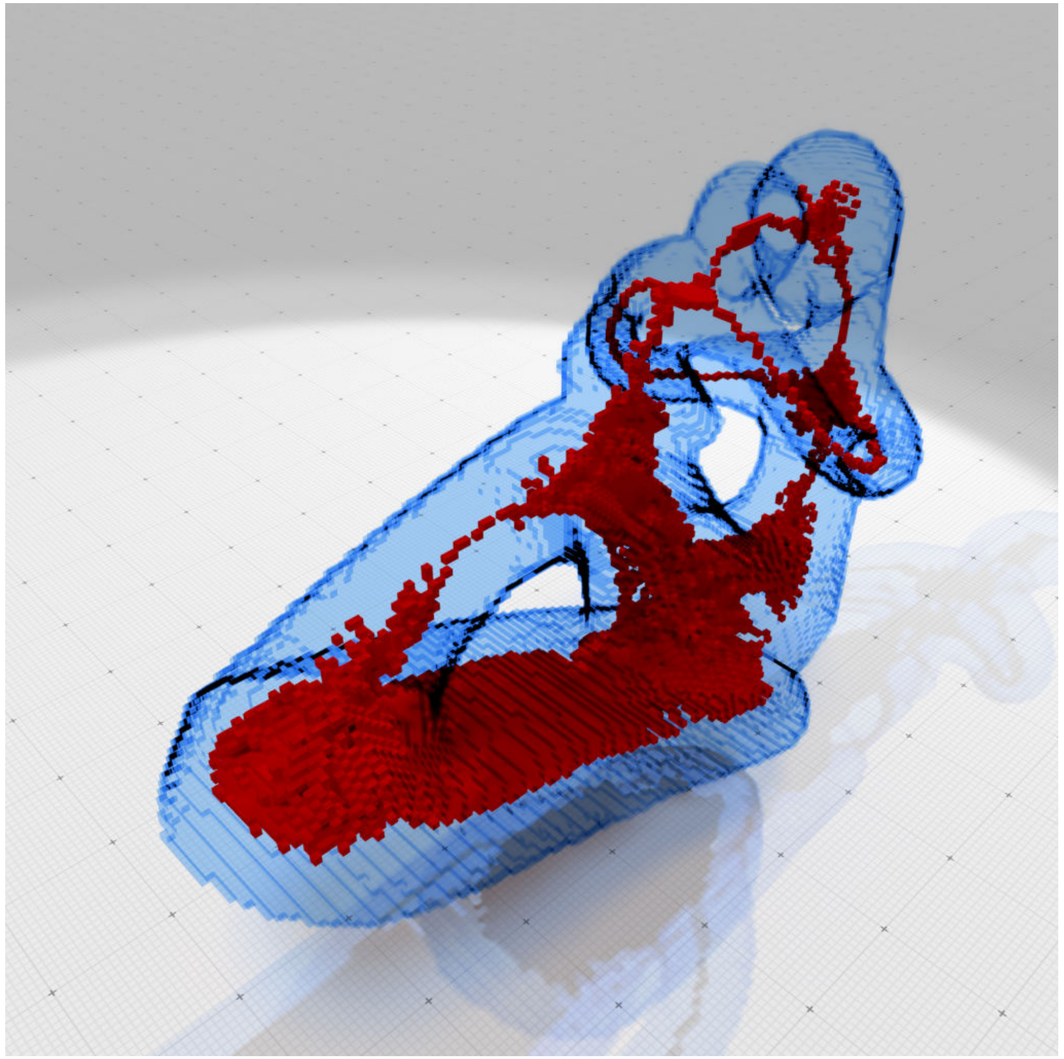
All complexes Y, such that $Z \subseteq Y \subseteq X$ are homotopic to X !

Allows parallel algorithms for extracting skeletons

Skeletons with critical kernels

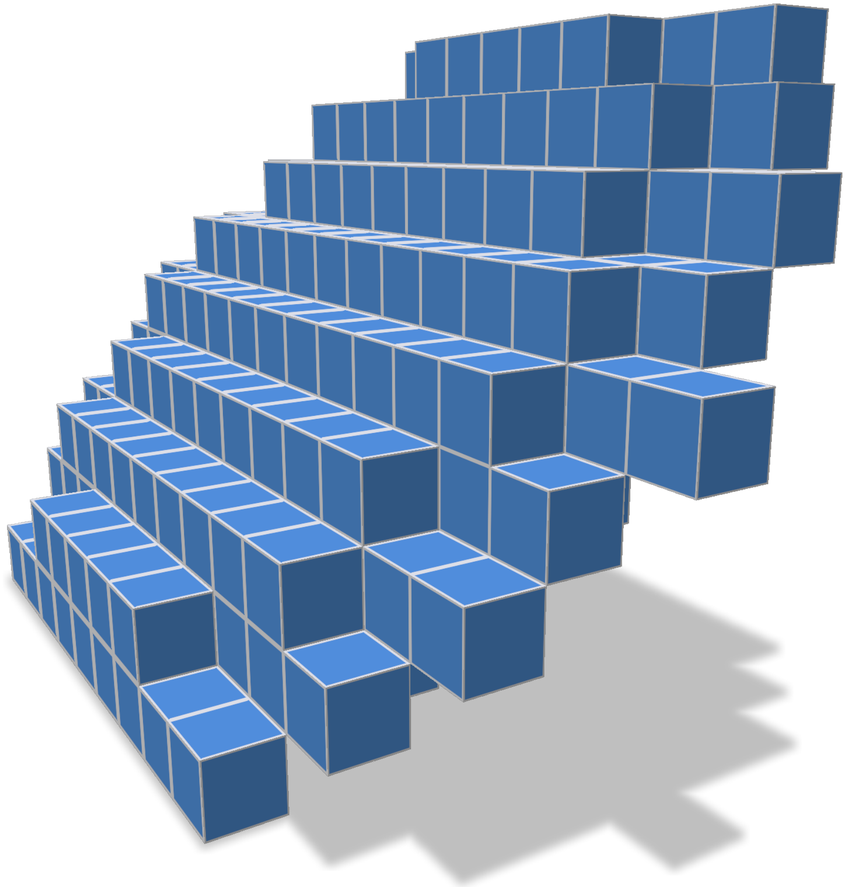


« curved » skeleton



« surface » skeleton

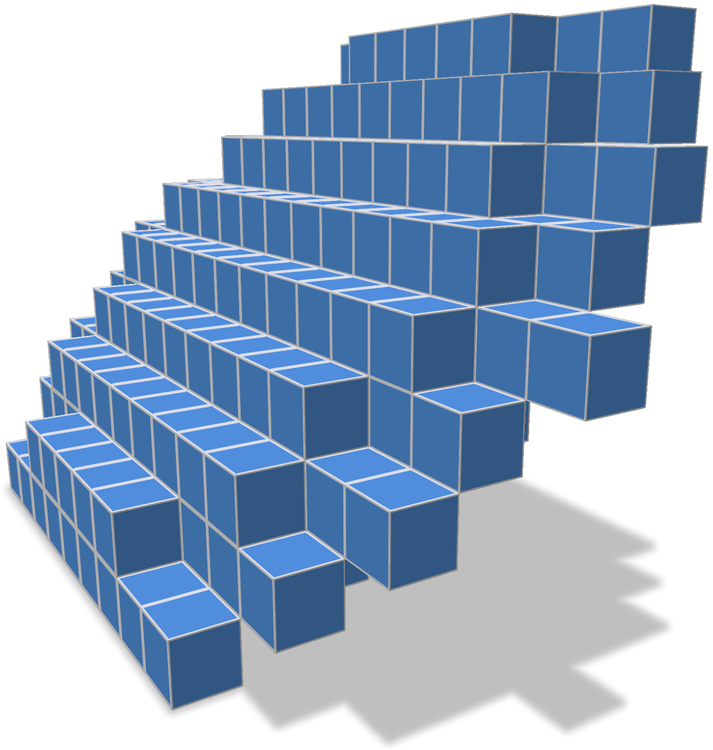
Digital surfaces



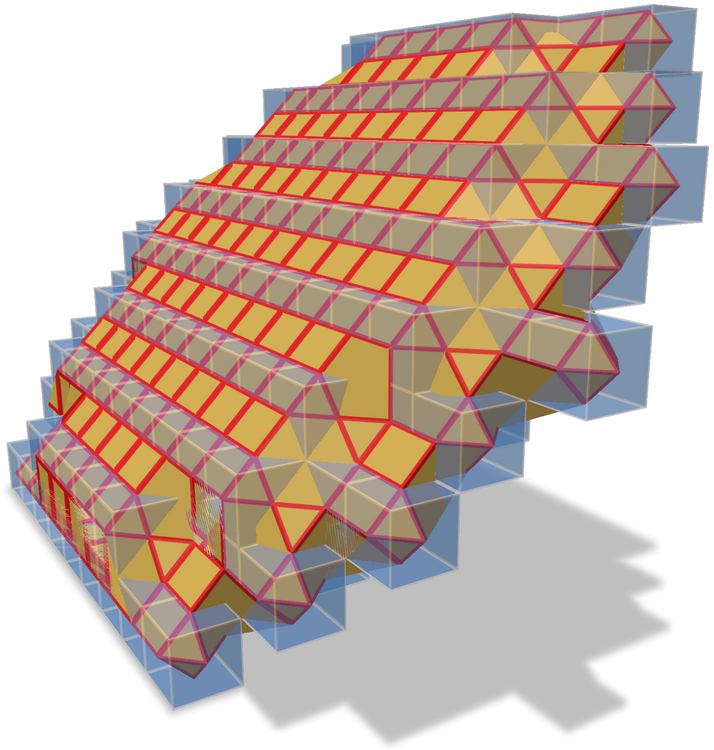
Primal surface
(here, digitization of some ellipsoid)

- digital surface ≈ set of faces of voxels
- in « ideal cases » 4-regular graph (3D)
  - vertices = surfels/faces
- generally not a manifold
  - pinched on edges and/or vertices
- not a sampling, only approximation
- only 6 different normals in 3D
  - even fine digital surface have poor normals

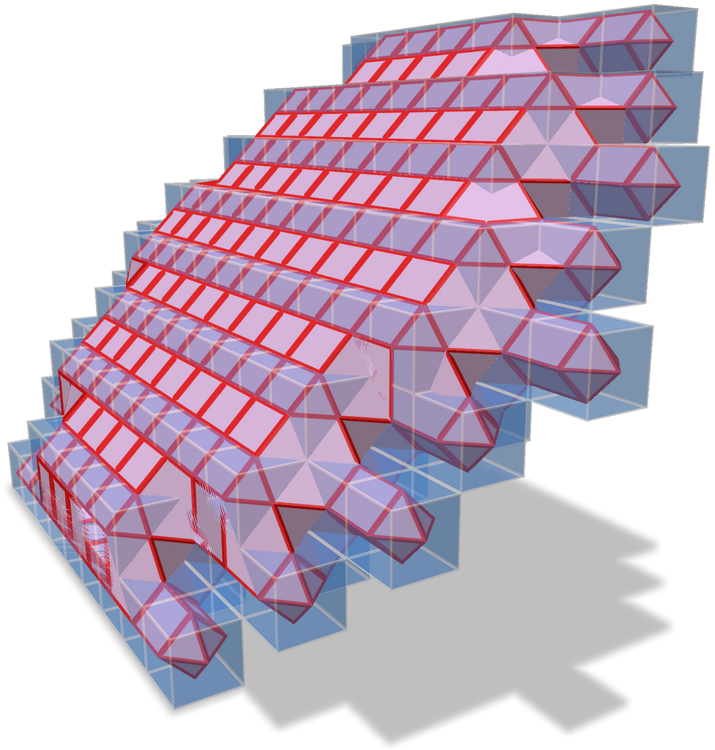
Digital surfaces + topology (primal ↔ dual)



Primal surface



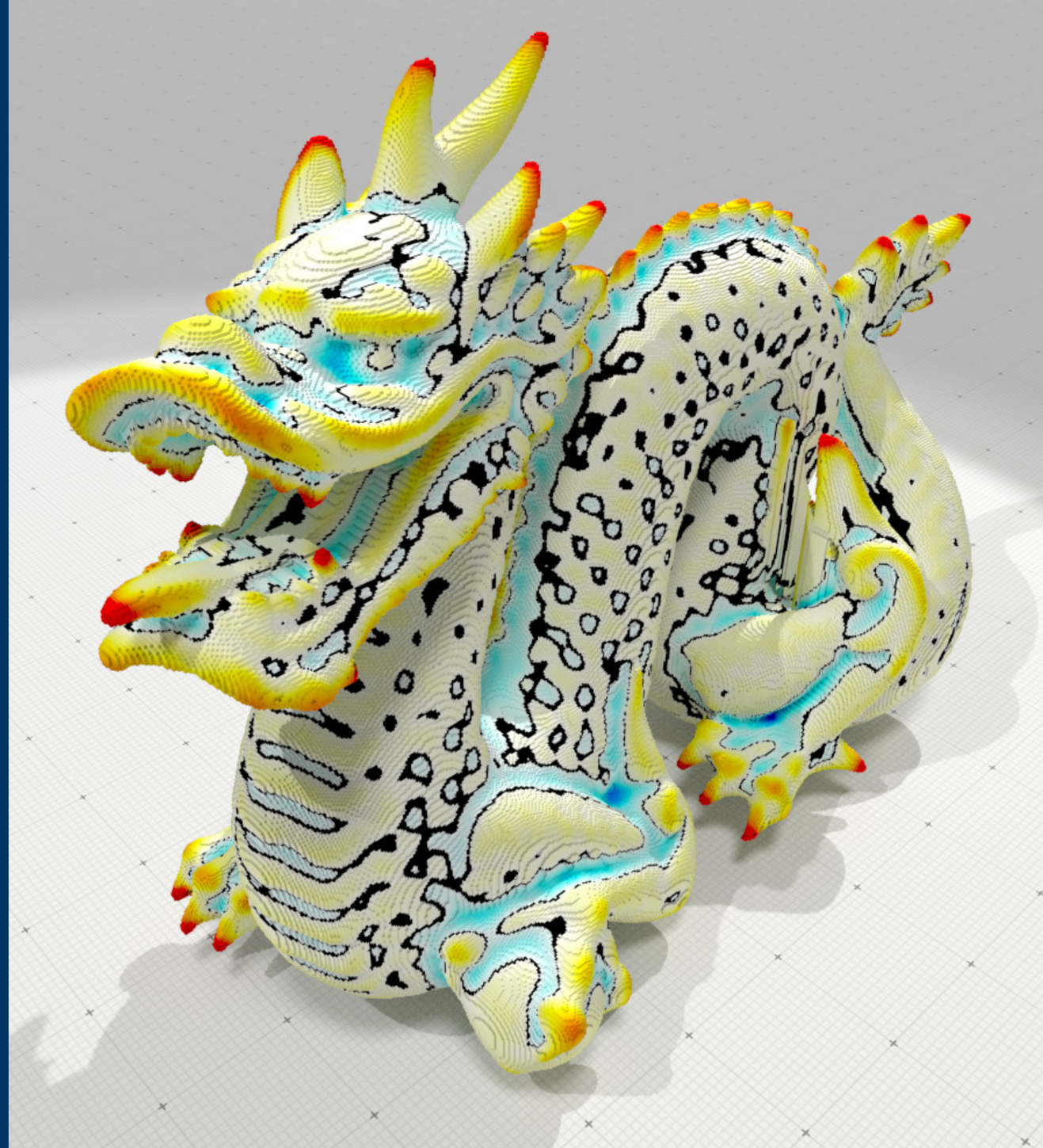
Dual surface
(26,6) topology



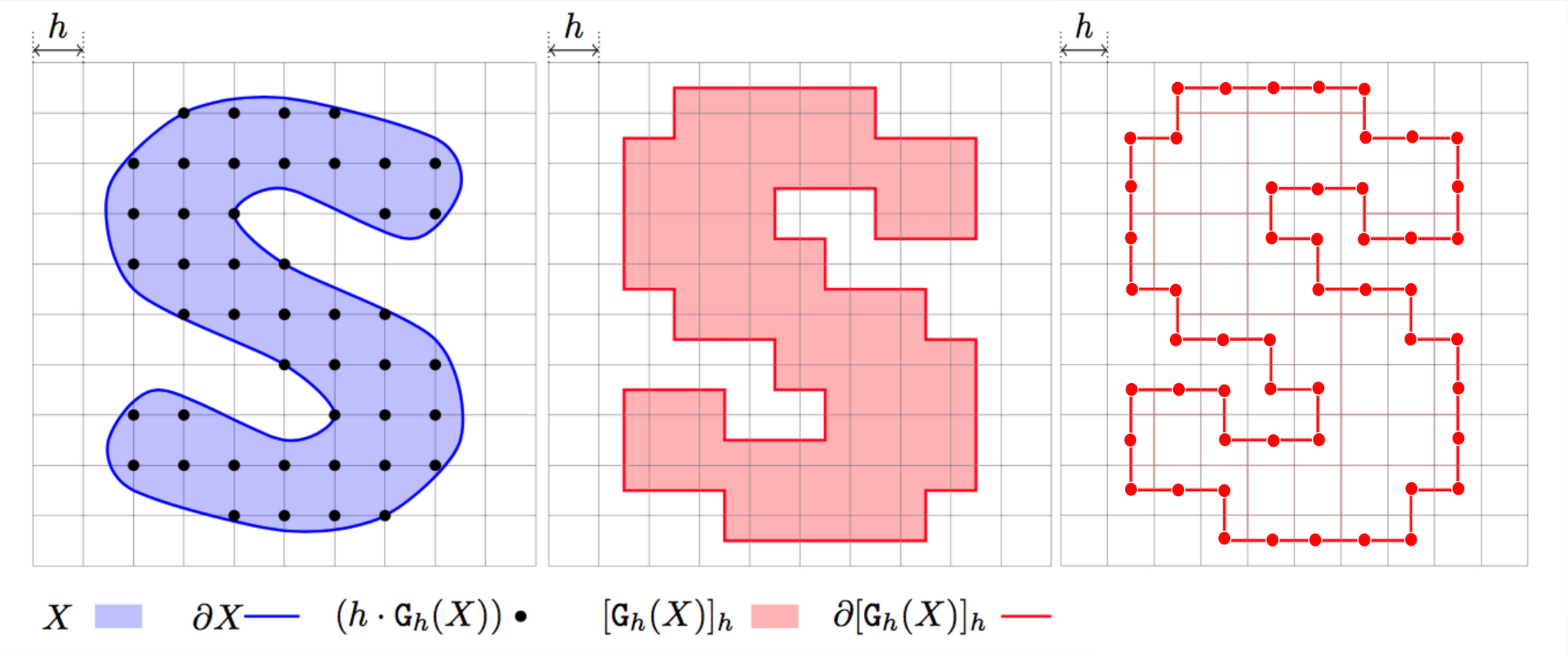
Dual surface
(6,26) topology

Adding object/background topology allows manifoldness in arbitrary dimensions - exactly d-1 paths crossing at each point

digital surface geometry



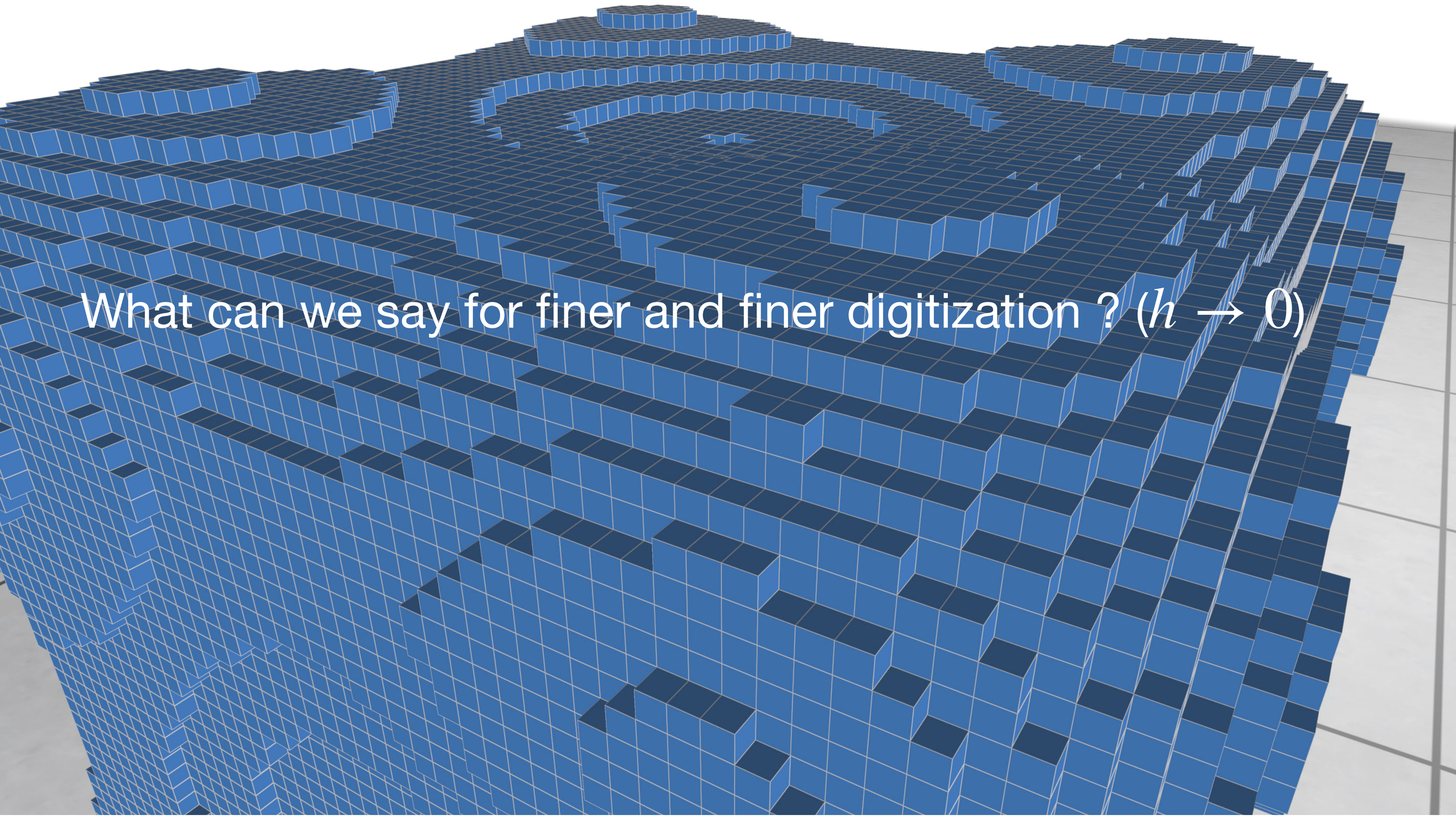
Linking continuous and digital geometry : Gauss digitization with gridstep h



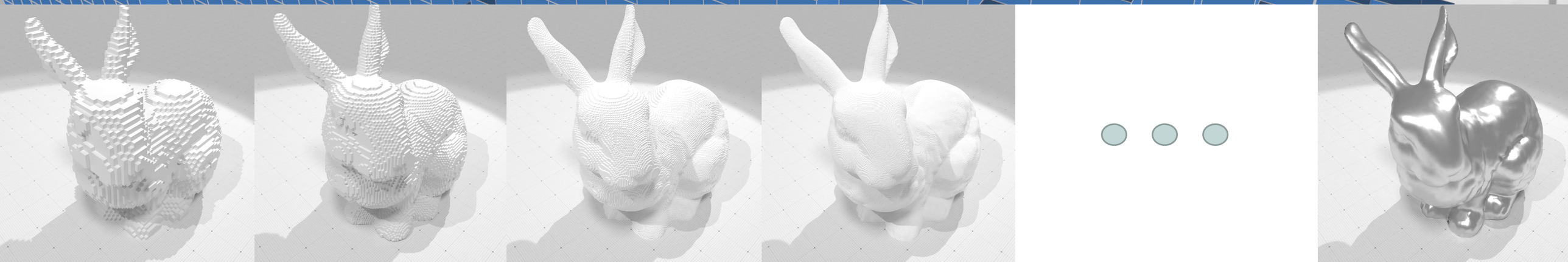
« digitization »

« voxelization »

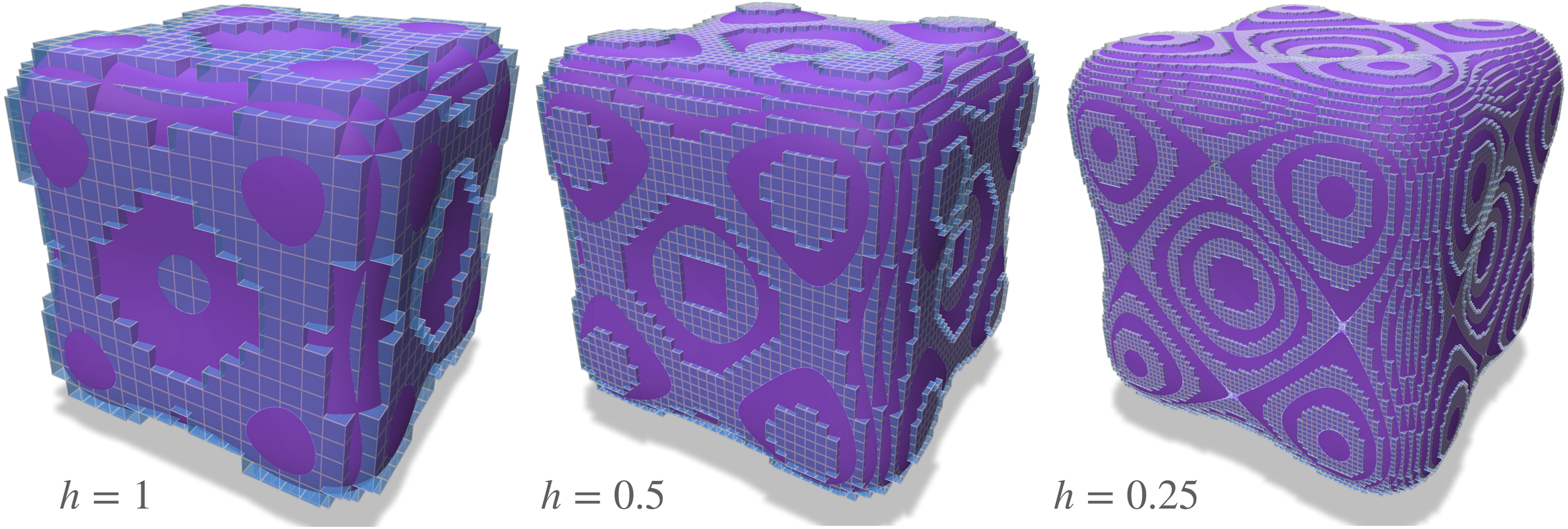
« digitized surface »


What can we say for finer and finer digitization ? ($h \to 0$)

What can we say for finer and finer digitization ? ($h \to 0$)



Hausdorff closeness of digitized shapes

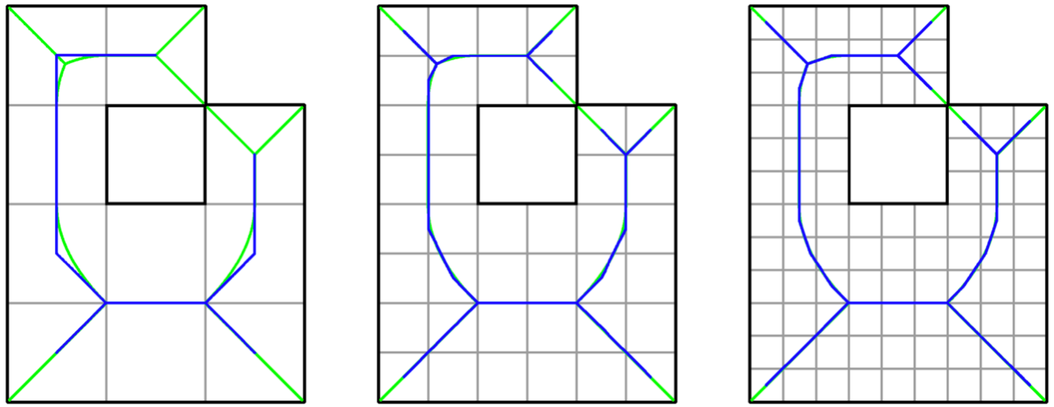


For any compact domain $X \in \mathbb{R}^d$ *such that* ∂X *has positive reach, and its digitization* $X_h := [G_h(X)]_h$ *on a grid with grid-step h, then* $d_H(\partial X, \partial X_h) \leq \sqrt{d}/2h$ *for small enough* h

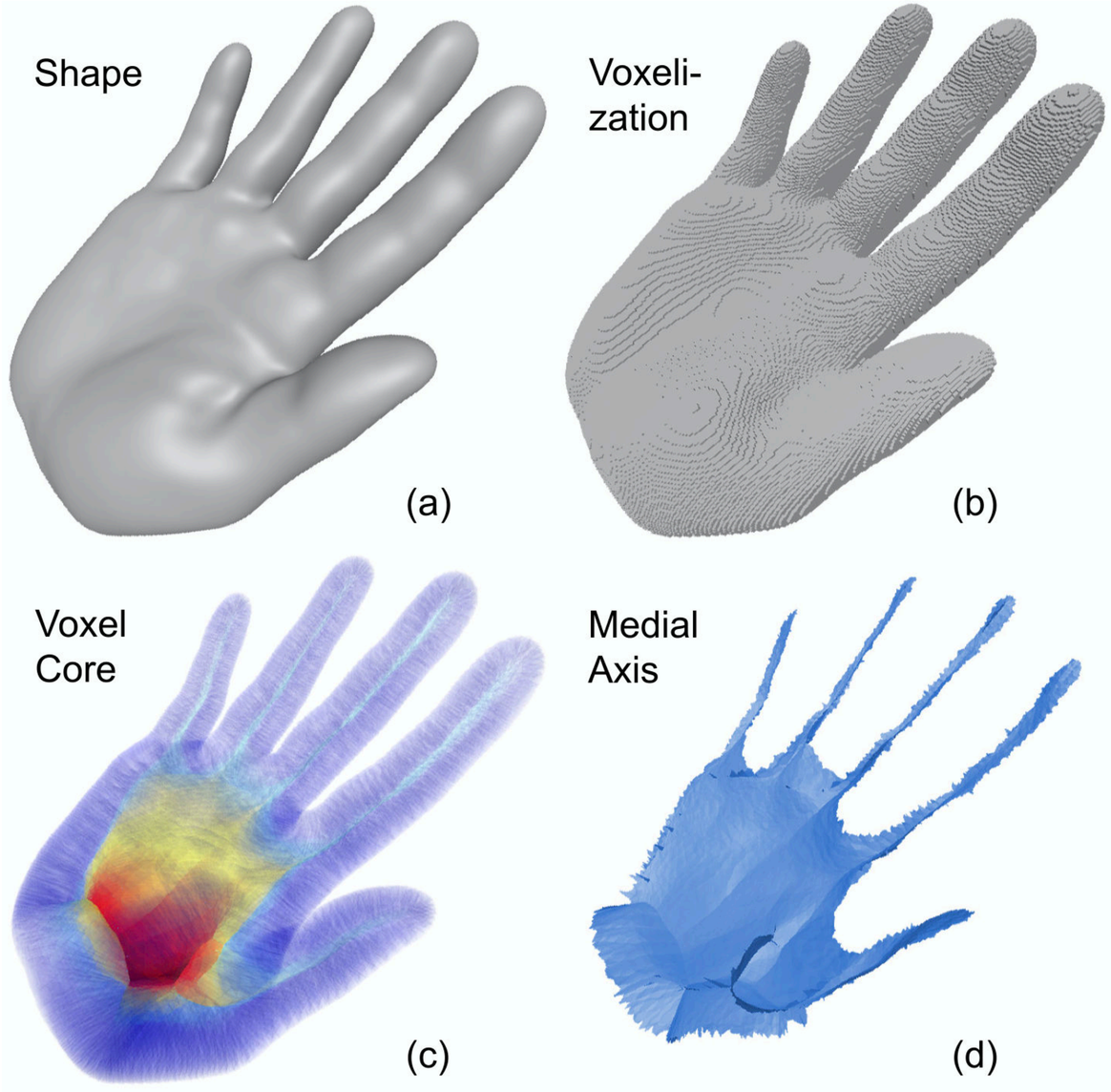
[LT16]

Homotopy equivalence

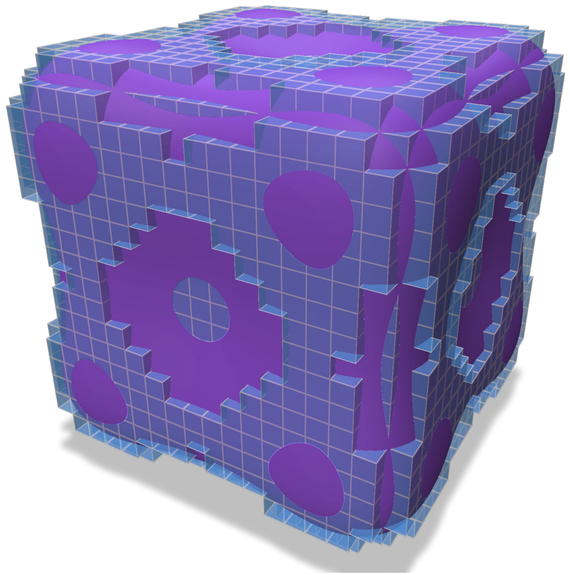
For a compact shape X with positive reach ρ, for $h < \frac{2\sqrt{3}}{3}\rho$, the set X and its voxelization $[G_h(X)]_h$ are homotopy equivalent.
Its voxel core is also homotopy equivalent.



[YLJ2018]

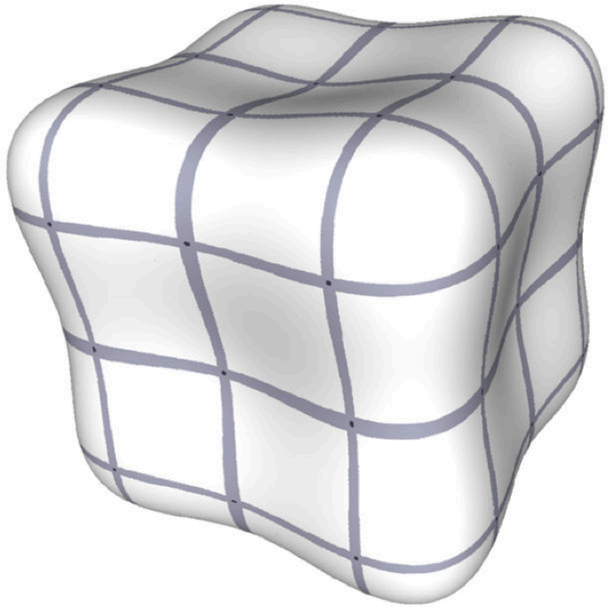


Bijectivity of projection and manifoldness



$h = 0.1$

$h = 0.05$



$h = 0.025$

If X has positive reach,
the size of the non-injective part of projection
$\pi_X : \partial X_h \to \partial X$ tends to zero as $h \to 0$.
(light gray + dark gray zones $\approx O(h)$)
[LT16]

If X has positive reach,
the size of the non-manifoldness part of ∂X_h
tends quickly to zero as $h \to 0$.
(dark gray zones $\approx O(h^2)$)
[LT16]

Multigrid convergence

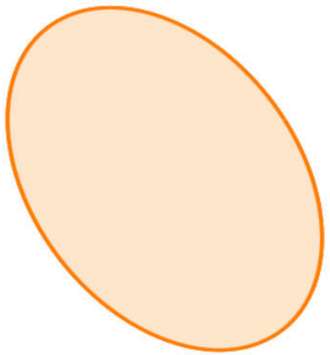
For digitization process G, the discrete geometric estimator $\hat{E}$ is multigrid convergent to the geometric quantity E for the family of shapes $\mathbb{X}$, iff, for any $X \in \mathbb{X}$, there exists a grid step $h_X > 0$, such that :

$\hat{E}(G_h(X), h)$ is defined for any $0 < h < h_X$,

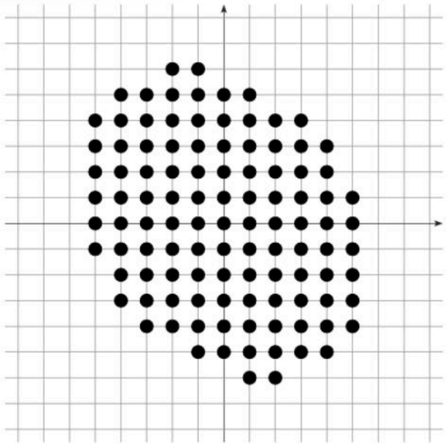
$$|\hat{E}(G_h(X), h) - E(X)| < \tau_X(h)$$

where the speed of convergence $\tau_X(h)$ has null limit when $h \to 0$.

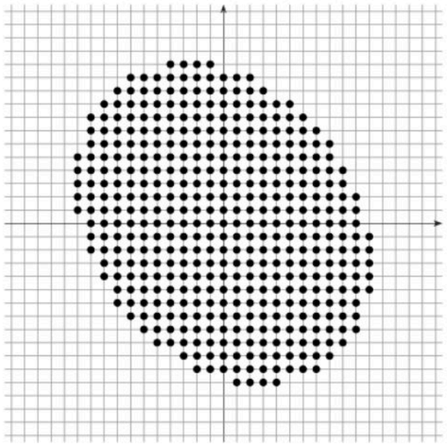
(Typically area, perimeter, integrals)



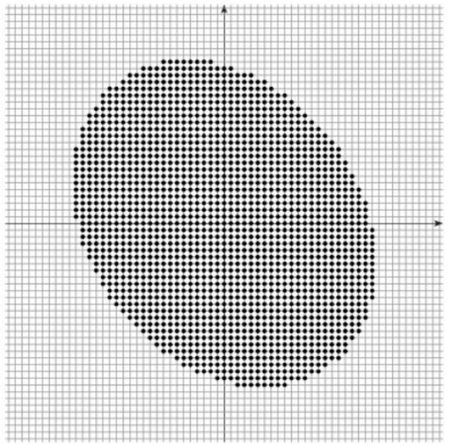
$M \in \mathbb{X}$



$\mathbb{G}_1(M)$



$\mathbb{G}_{0.5}(M)$



$\mathbb{G}_{0.25}(M)$

$\widehat{\text{Area}}\,(G_h(X), h) := h^2 \#(G_h(X))$ tends toward $\text{Area}(M)$ as $h \to 0$

Convergence speed is $O(h)$ and even $O(h^{\frac{22}{15}})$ for smooth enough M

Multigrid convergence (local version)

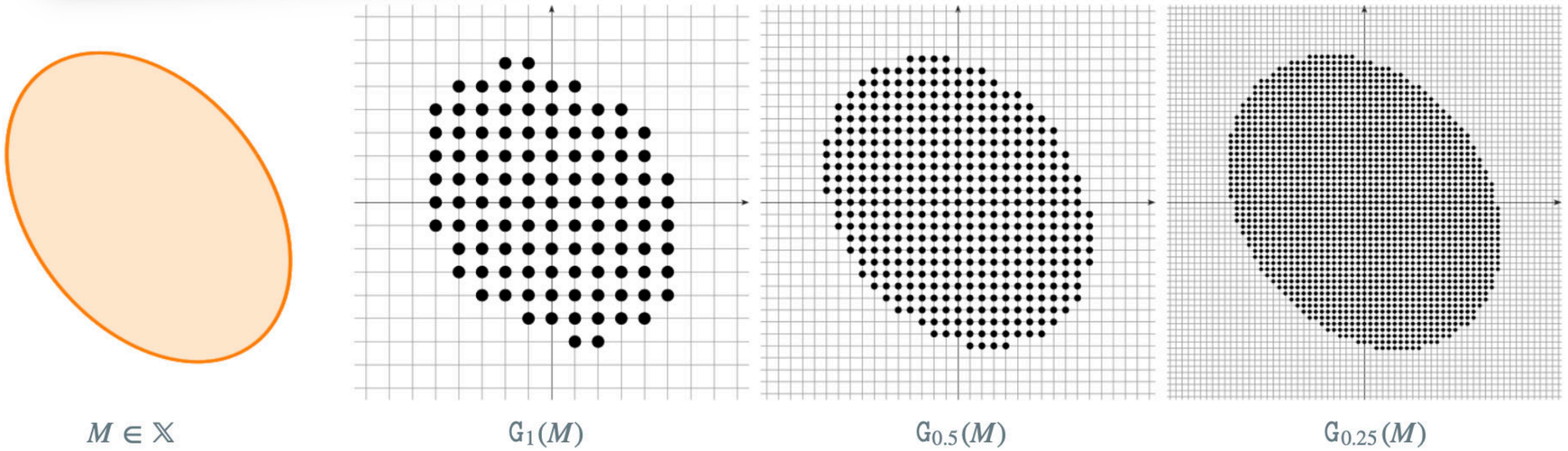
For digitization process G, the local discrete geometric estimator $\hat{E}$ is multigrid convergent to the geometric quantity E for the family of shapes $\mathbb{X}$, iff, for any $X \in \mathbb{X}$, there exists a grid step $h_X > 0$, such that :

$\hat{E}(G_h(X), \hat{x}, h)$ is defined for any $\hat{x} \in \partial[G_h(X)]_h$ with $0 < h < h_X$,

for any $x \in \partial X$, for any $\hat{x} \in \partial[G_h(X)]_h$ with $\|x - \hat{x}\|_\infty \leq h$, $\quad |\hat{E}(G_h(X), \hat{x}, h) - E(X, x)| < \tau_X(h)$

where the speed of convergence $\tau_X(h)$ has null limit when $h \to 0$.

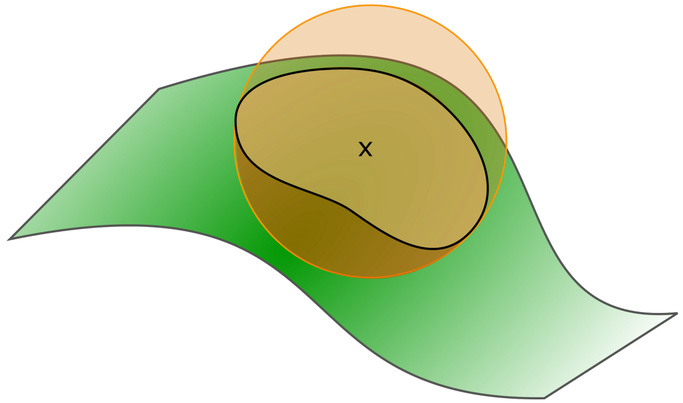
(Typically normal direction, curvatures, ...)

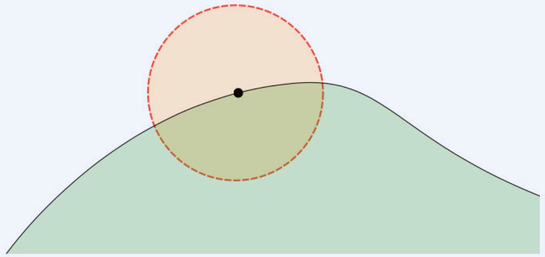


$M \in \mathbb{X}$ $\quad \mathbb{G}_1(M)$ $\quad \mathbb{G}_{0.5}(M)$ $\quad \mathbb{G}_{0.25}(M)$

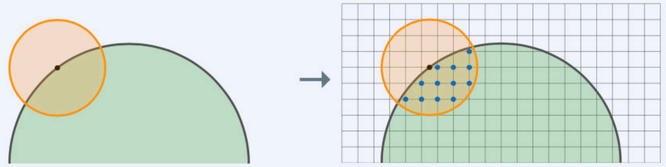
Normal vector and curvatures estimation

- Integral Invariants : analyzing set $B_R(x) \cap X$ gives normal vector, principal directions and curvatures [Pottmann et al. 2007]

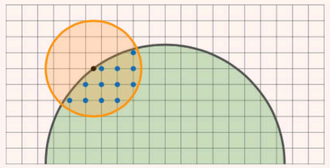




$$\kappa(M,\mathbf{x}) := \underbrace{\frac{3\pi}{2R} - \frac{3 \cdot A_R(M,\mathbf{x})}{R^3}}_{\kappa^R(M,\mathbf{x})} + O(R)$$ [Pottmann et al. 2007]

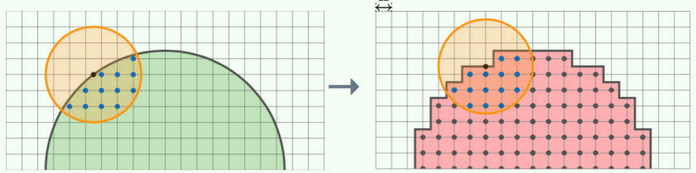


$$A_R(M,\mathbf{x}) \rightarrow \widehat{\text{Area}}\,(B_{R/h}(\mathbf{x}/h) \cap \mathbb{G}_h(M))$$



+ [Pottmann et al. 2007]

$$\kappa^R(\mathbb{G}_h(M),\mathbf{x},h)$$



$$\kappa^R(\mathbb{G}_h(M),\mathbf{x},\hat{h}) \rightarrow \kappa(M,\mathbf{x})$$

[C., Levallois, Lachaud]

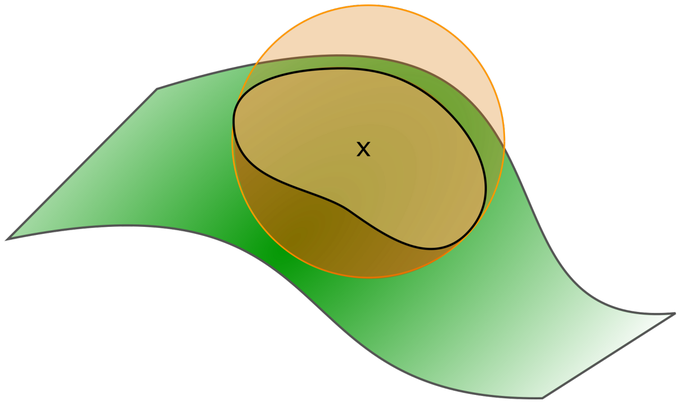
Let M be a convex shape in $\mathbb{R}^2$ with a C^3 bounded positive curvature boundary.

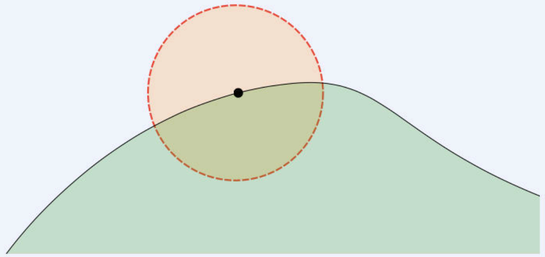
$$\forall \mathbf{x} \in \partial M, \forall \hat{\mathbf{x}} \in \partial[\mathbb{G}_h(M)]_h, \|\hat{x} - x\|_\infty \leq h \Rightarrow$$

$$\begin{aligned} |\kappa^R(\mathbb{G}_h(M),\mathbf{x},\hat{h}) - \kappa(M,\mathbf{x})| = \; & O(R) \\ + \; & O\left(\frac{h^\beta}{R^{1+\beta}}\right) \\ + \; & O\left(\frac{h^{\alpha'}}{R^2}\right) + O\left(h^{\alpha'}\right) + O\left(\frac{h^{2\alpha'}}{R^2}\right) \end{aligned}$$

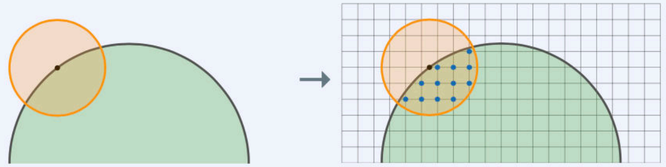
Normal vector and curvatures estimation

- Integral Invariants : analyzing set $B_R(x) \cap X$ gives normal vector, principal directions and curvatures [Pottmann et al. 2007]

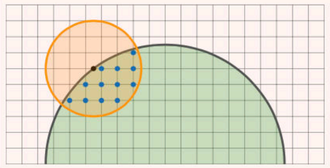




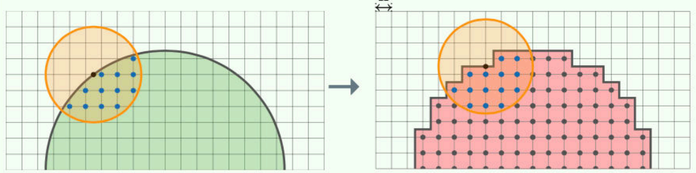
$$\kappa(M,\mathbf{x}) := \underbrace{\frac{3\pi}{2R} - \frac{3 \cdot A_R(M,\mathbf{x})}{R^3}}_{\kappa^R(M,\mathbf{x})} + O(R)$$ [Pottmann et al. 2007]



$$A_R(M,\mathbf{x}) \rightarrow \widehat{\text{Area}}\,(B_{R/h}(\mathbf{x}/h) \cap \mathbb{G}_h(M))$$



+ [Pottmann et al. 2007] $\quad \kappa^R(\mathbb{G}_h(M),\mathbf{x},h)$



$$\kappa^R(\mathbb{G}_h(M),\hat{\mathbf{x}},h) \rightarrow \kappa(M,\mathbf{x})$$

[C., Levallois, Lachaud]

Let M be a convex shape in $\mathbb{R}^2$ with a C^3 bounded positive curvature boundary.

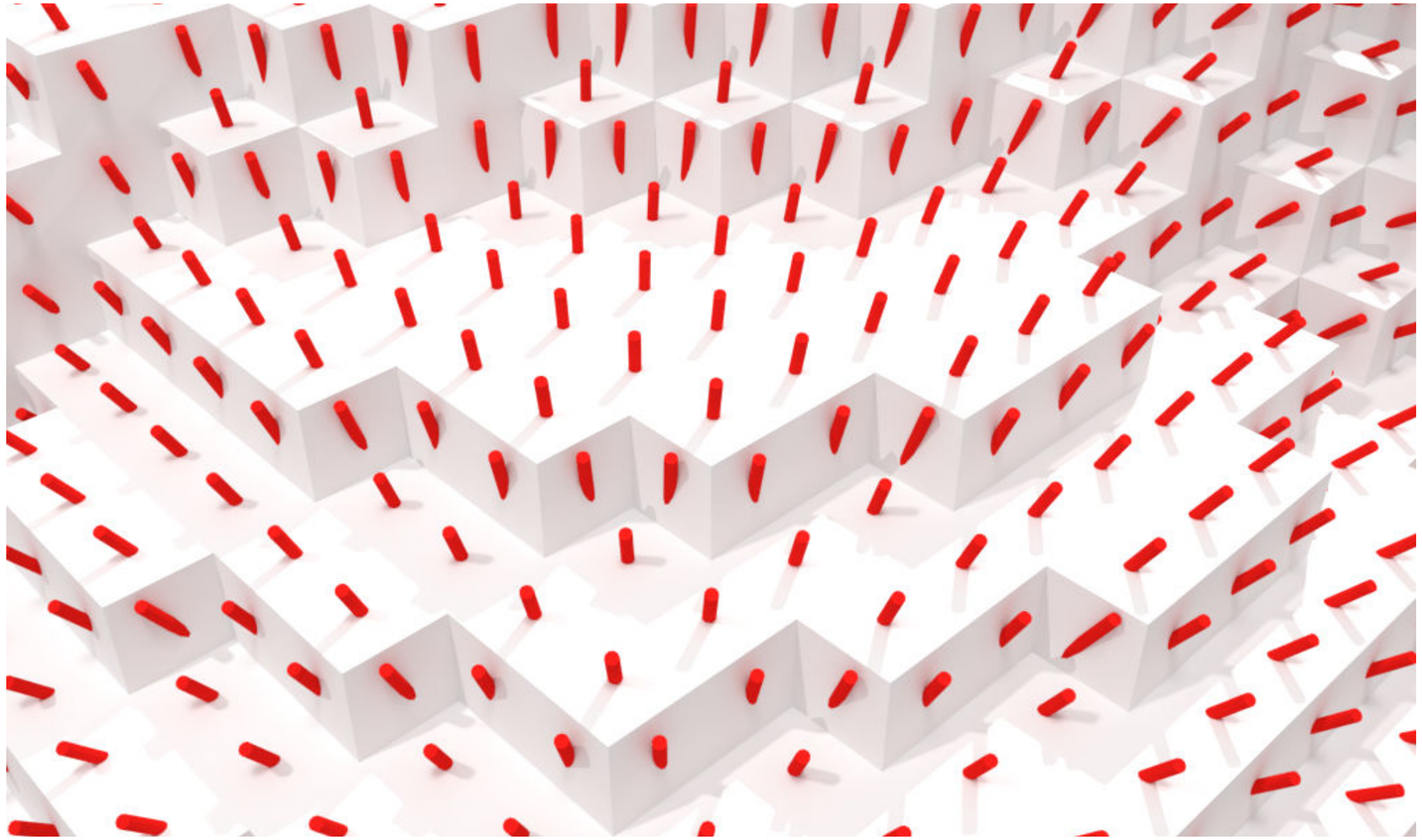
$$\forall \mathbf{x} \in \partial M, \forall \hat{\mathbf{x}} \in \partial[\mathbb{G}_h(M)]_h, \|\hat{x} - x\|_\infty \leq h \Rightarrow$$

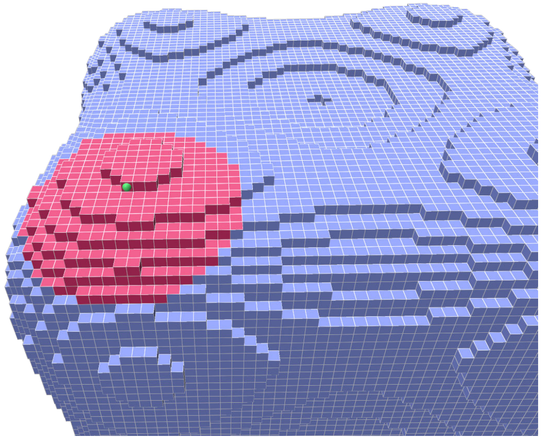
$$\begin{aligned} |\kappa^R(\mathbb{G}_h(M),\hat{\mathbf{x}},h) - \kappa(M,\mathbf{x})| = \; & O(R) \\ & + O\left(\frac{h^\beta}{R^{1+\beta}}\right) \\ & + O\left(\frac{h^{\alpha'}}{R^2}\right) + O\left(h^{\alpha'}\right) + O\left(\frac{h^{2\alpha'}}{R^2}\right) \end{aligned}$$

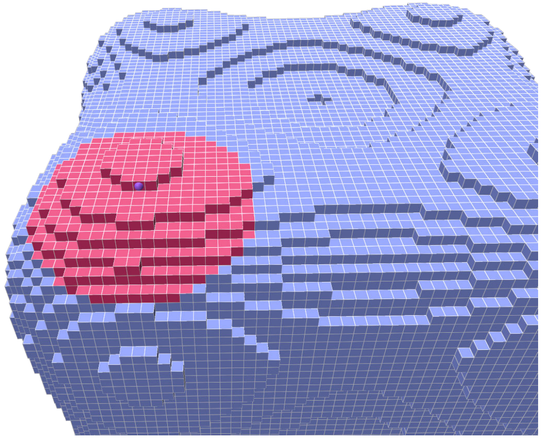
With optimal radius $R = O(h^{\frac{1}{3}})$, then :

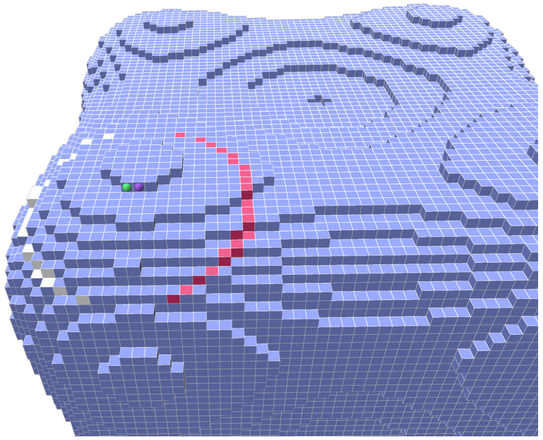
- normals $\left\| \hat{\mathbf{n}}(G_h(M),\xi(x),h)) - \mathbf{n}(M,x) \right\| \leq C \cdot h^{\frac{2}{3}}$
- mean curvature $\left\| \hat{\kappa}(M_h,\xi(x))) - \kappa(M,x) \right\|_2 \leq C \cdot h^{\frac{1}{3}}$
- … [CLL2014], [LCL2017]

Normal vector field estimation

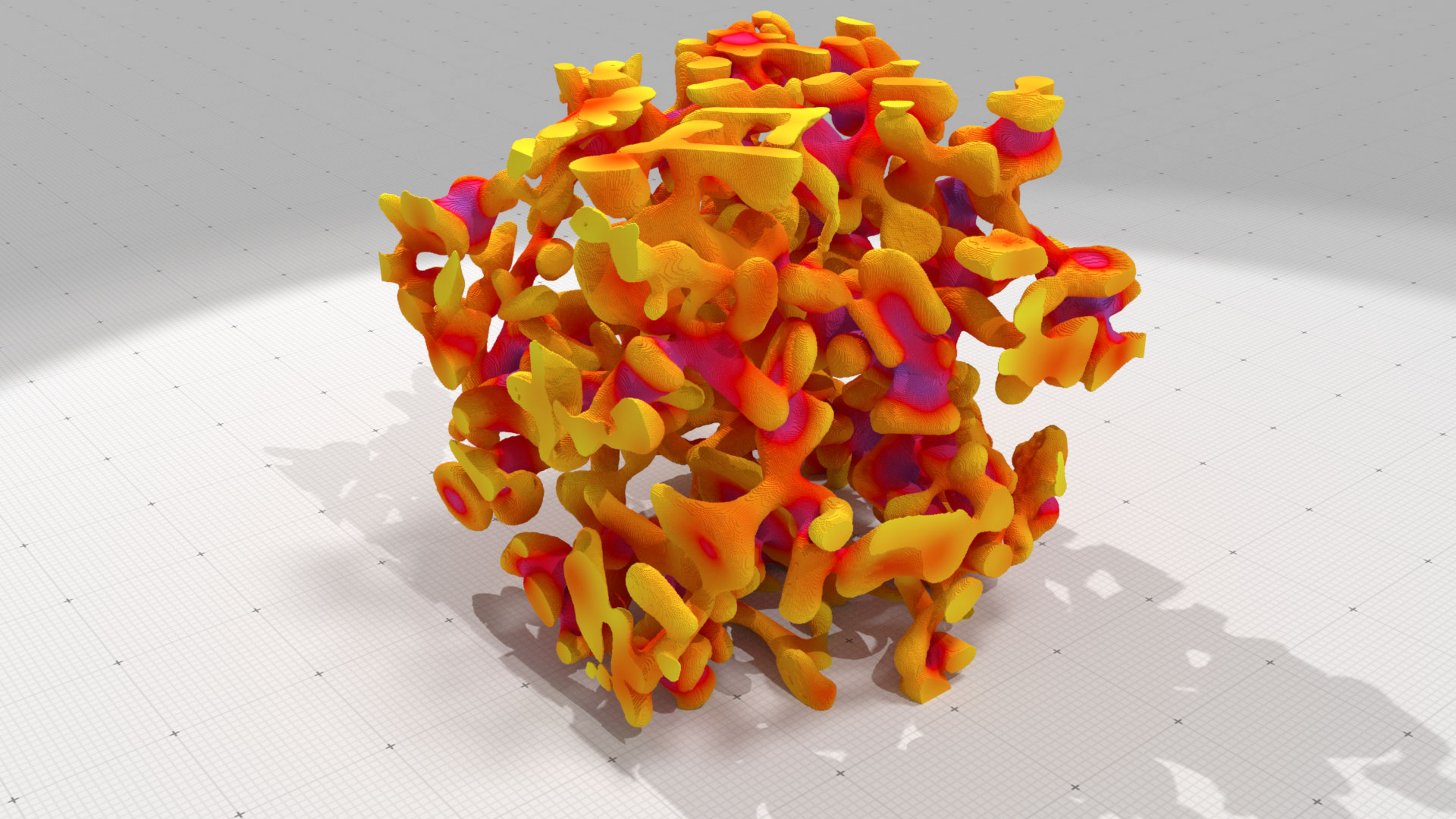


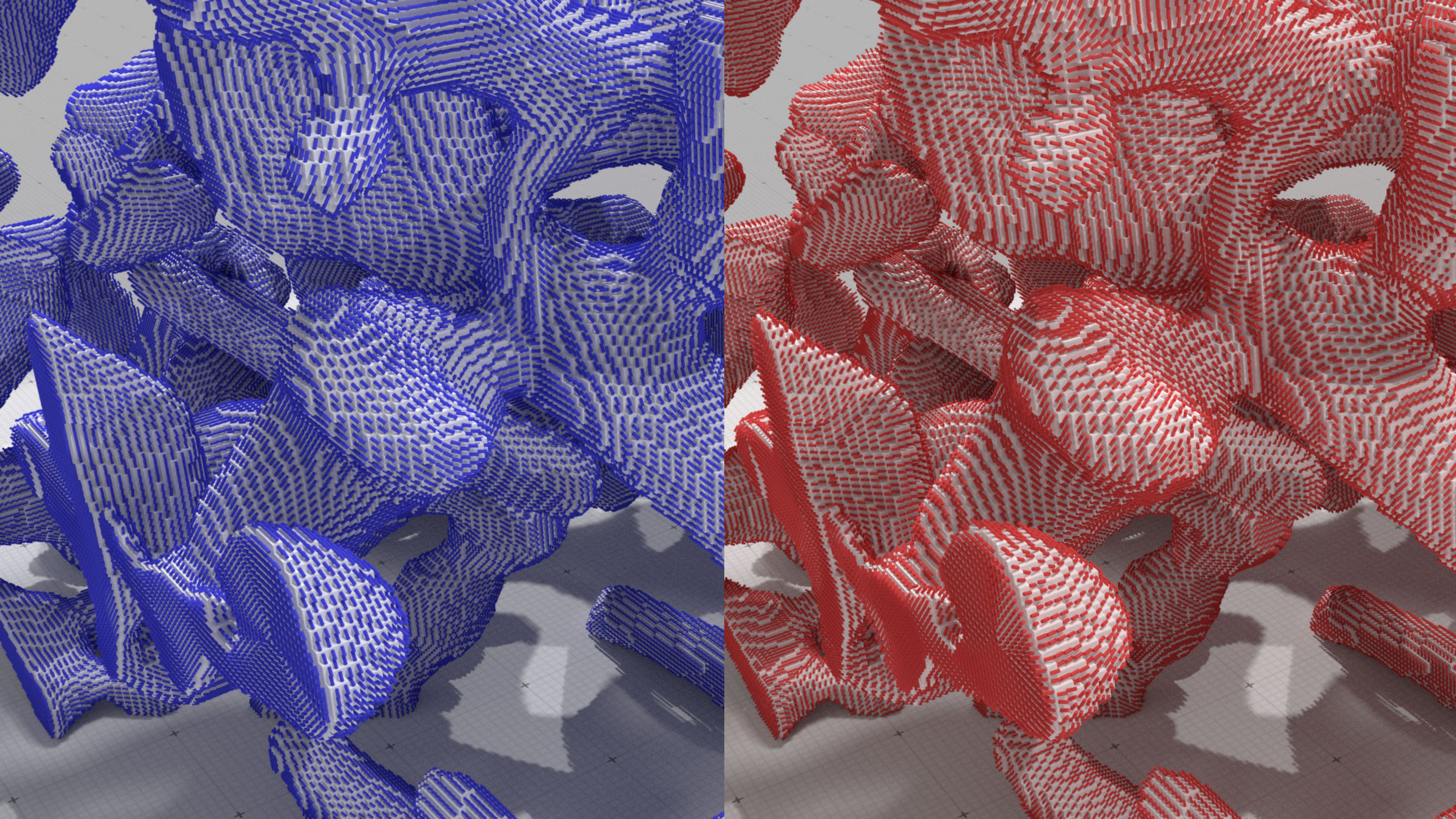






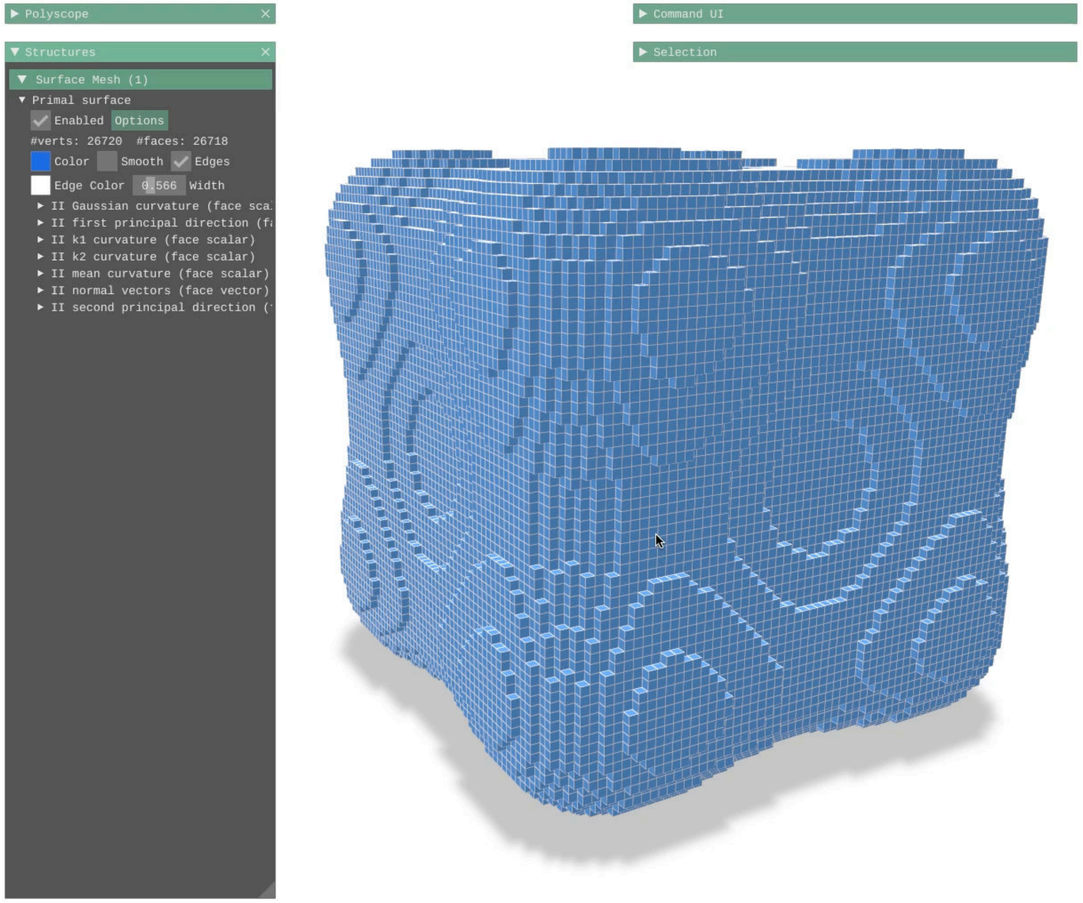
Incremental computation : estimate at y nearby x only requires preceding result + looking at points within $B_R(y) \ominus B_R(x)$



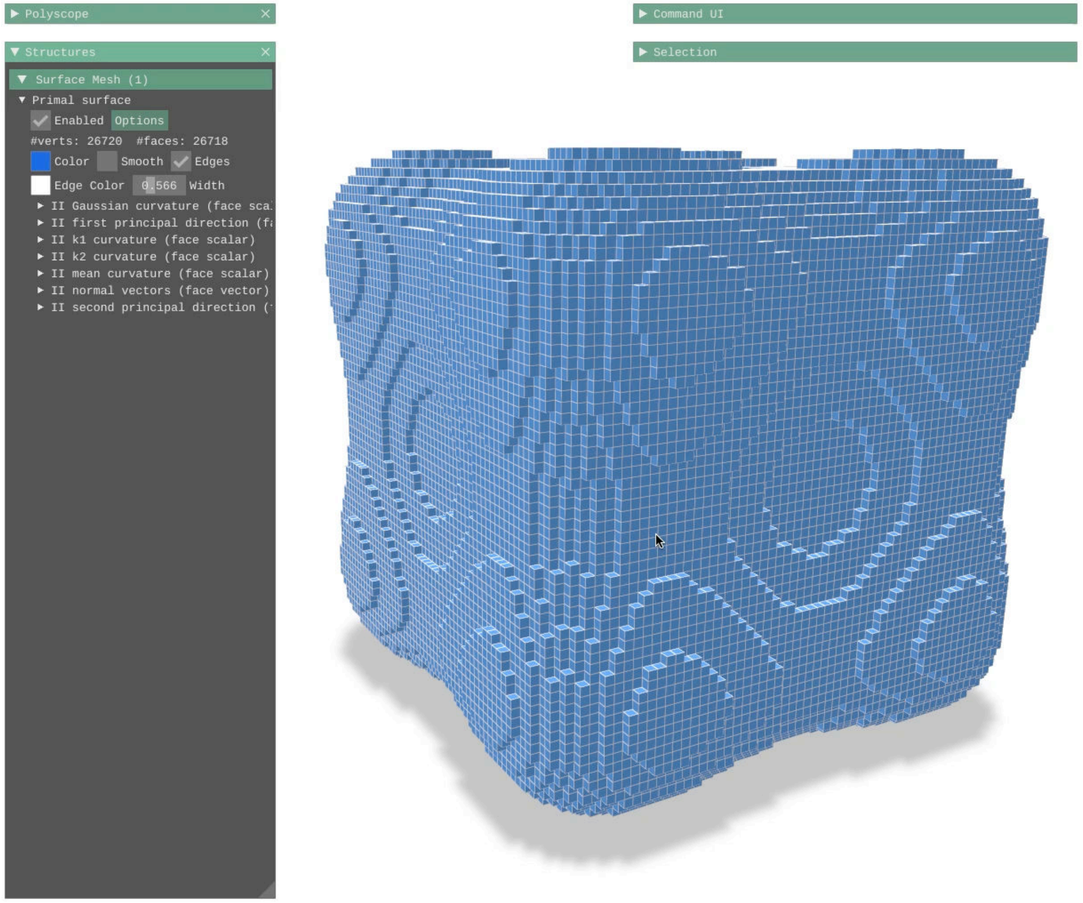


hands on…

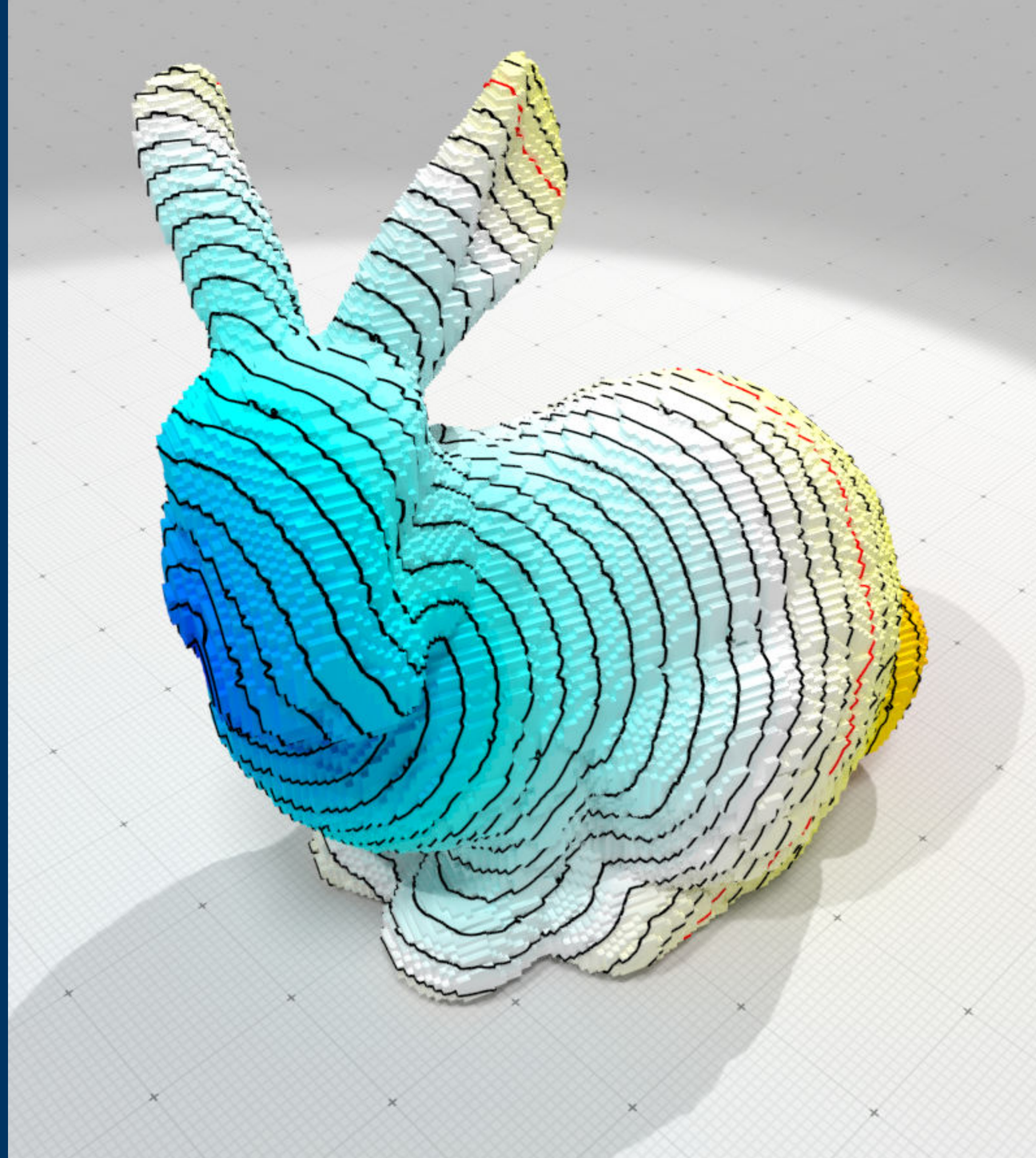
```cpp
void oneStepAll(double h)
{
  auto params = SH3::defaultParameters() | SHG3::defaultParameters() |  SHG3::parametersGeometryEstimation();
  params( "polynomial", "goursat" )( "gridstep", h );
  auto implicit_shape  = SH3::makeImplicitShape3D  ( params );
  auto digitized_shape = SH3::makeDigitizedImplicitShape3D( implicit_shape, params );
  auto K               = SH3::getKSpace( params );
  auto binary_image    = SH3::makeBinaryImage( digitized_shape, params );
  auto surface         = SH3::makeDigitalSurface( binary_image, K, params );
  auto embedder        = SH3::getCellEmbedder( K );
  SH3::Cell2Index c2i;
  auto surfels         = SH3::getSurfelRange( surface, params );
  auto primalSurface   = SH3::makePrimalPolygonalSurface(c2i, surface);

  //Need to convert the faces
  std::vector<std::vector<std::size_t>> faces;
  for(auto &face: primalSurface->allFaces())
    faces.push_back(primalSurface->verticesAroundFace( face ));
  auto digsurf = polyscope::registerSurfaceMesh("Primal surface", primalSurface->positions(), faces);
  digsurf->rescaleToUnit(); digsurf->setEdgeWidth(h*h);  digsurf->setEdgeColor({1.,1.,1.});

  //Computing some differential quantities
  params("r-radius", 5*std::pow(h,-2.0/3.0));
  auto Mcurv     = SHG3::getIIMeanCurvatures(binary_image, surfels, params);
  auto normalsII = SHG3::getIINormalVectors(binary_image, surfels, params);
  auto KTensor   = SHG3::getIIPrincipalCurvaturesAndDirections(binary_image, surfels, params);  //Recomputing...

  std::vector<double> Gcurv(surfels.size()),k1(surfels.size()),k2(surfels.size());
  std::vector<RealVector> d1(surfels.size()),d2(surfels.size());
  auto i=0;
  for(auto &t: KTensor) //AOS->SOA
  {
    k1[i]    = std::get<0>(t);
    k2[i]    = std::get<1>(t);
    d1[i]    = std::get<2>(t);
    d2[i]    = std::get<3>(t);
    Gcurv[i] = k1[i]*k2[i];
    ++i;
  }

  //Attaching quantities
  digsurf->addFaceVectorQuantity("II normal vectors", normalsII, polyscope::VectorType::AMBIENT);
  digsurf->addFaceScalarQuantity("II mean curvature", Mcurv);
  digsurf->addFaceScalarQuantity("II Gaussian curvature", Gcurv);
  digsurf->addFaceScalarQuantity("II k1 curvature", k1);
  digsurf->addFaceScalarQuantity("II k2 curvature", k2);
  digsurf->addFaceVectorQuantity("II first principal direction", d1, polyscope::VectorType::AMBIENT);
  digsurf->addFaceVectorQuantity("II second principal direction", d2, polyscope::VectorType::AMBIENT);
}
```



```cpp
void oneStepAll(double h)
{
  auto params = SH3::defaultParameters() | SHG3::defaultParameters() |  SHG3::parametersGeometryEstimation();
  params( "polynomial", "goursat" )( "gridstep", h );
  auto implicit_shape  = SH3::makeImplicitShape3D  ( params );
  auto digitized_shape = SH3::makeDigitizedImplicitShape3D( implicit_shape, params );
  auto K               = SH3::getKSpace( params );
  auto binary_image    = SH3::makeBinaryImage( digitized_shape, params );
  auto surface         = SH3::makeDigitalSurface( binary_image, K, params );
  auto embedder        = SH3::getCellEmbedder( K );
  SH3::Cell2Index c2i;
  auto surfels         = SH3::getSurfelRange( surface, params );
  auto primalSurface   = SH3::makePrimalPolygonalSurface(c2i, surface);

  //Need to convert the faces
  std::vector<std::vector<std::size_t>> faces;
  for(auto &face: primalSurface->allFaces())
    faces.push_back(primalSurface->verticesAroundFace( face ));
  auto digsurf = polyscope::registerSurfaceMesh("Primal surface", primalSurface->positions(), faces);
  digsurf->rescaleToUnit(); digsurf->setEdgeWidth(h*h);  digsurf->setEdgeColor({1.,1.,1.});

  //Computing some differential quantities
  params("r-radius", 5*std::pow(h,-2.0/3.0));
  auto Mcurv     = SHG3::getIIMeanCurvatures(binary_image, surfels, params);
  auto normalsII = SHG3::getIINormalVectors(binary_image, surfels, params);
  auto KTensor   = SHG3::getIIPrincipalCurvaturesAndDirections(binary_image, surfels, params);  //Recomputing...

  std::vector<double> Gcurv(surfels.size()),k1(surfels.size()),k2(surfels.size());
  std::vector<RealVector> d1(surfels.size()),d2(surfels.size());
  auto i=0;
  for(auto &t: KTensor) //AOS->SOA
  {
    k1[i]    = std::get<0>(t);
    k2[i]    = std::get<1>(t);
    d1[i]    = std::get<2>(t);
    d2[i]    = std::get<3>(t);
    Gcurv[i] = k1[i]*k2[i];
    ++i;
  }

  //Attaching quantities
  digsurf->addFaceVectorQuantity("II normal vectors", normalsII, polyscope::VectorType::AMBIENT);
  digsurf->addFaceScalarQuantity("II mean curvature", Mcurv);
  digsurf->addFaceScalarQuantity("II Gaussian curvature", Gcurv);
  digsurf->addFaceScalarQuantity("II k1 curvature", k1);
  digsurf->addFaceScalarQuantity("II k2 curvature", k2);
  digsurf->addFaceVectorQuantity("II first principal direction", d1, polyscope::VectorType::AMBIENT);
  digsurf->addFaceVectorQuantity("II second principal direction", d2, polyscope::VectorType::AMBIENT);
}
```



advanced digital surface geometry processing



Laplace-Beltrami on digital surfaces

$$\Delta u = \nabla \cdot \nabla u$$

Many discretization scheme for triangular/polygonal meshes

	SYM	LOC	LIN	POS	PSD	C^2-CON
Mean Value	x	✓	✓	✓	x	x
Intrinsic Del	✓	x	✓	✓	✓	x
Combinatorial	✓	✓	x	✓	✓	x
Cotan	x	✓	✓	x	✓	x
Polygonal Lap.	x	✓	✓	x	✓	x
Convolutional	x	x	?	✓	?	✓
r– local	✓	x	?	✓	?	✓

(update of "Discrete Laplace operators: No free lunch" [Wardetzky et al., 2007])

Laplace-Beltrami on digital surfaces

$$\Delta u = \nabla \cdot \nabla u$$

Many discretization scheme for triangular/polygonal meshes

	SYM	LOC	LIN	POS	PSD	C^2-CON
Mean Value	x	✓	✓	✓	x	x
Intrinsic Del	✓	x	✓	✓	✓	x
Combinatorial	✓	✓	x	✓	✓	x
Cotan	x	✓	✓	x	✓	x
Polygonal Lap.	x	✓	✓	x	✓	x
Convolutional	x	x	?	✓	?	✓
r– local	✓	x	?	✓	?	✓

(update of "Discrete Laplace operators: No free lunch" [Wardetzky et al., 2007])

Question: can we design a Laplace-Beltrami on digital surface with strong consistency?

Convolution based Laplace-Beltrami operator on Digital Surfaces

à-la [Belkin et al 08]

$$(L_h \tilde{u})(\mathbf{s}) := \frac{1}{t_h (4\pi t_h)^{\frac{d}{2}}} \sum_{\mathbf{r} \in S} e^{-\frac{||\mathbf{r}-\mathbf{s}||^2}{4t_h}} [\tilde{u}(\mathbf{r}) - \tilde{u}(\mathbf{s})] \mu(\mathbf{r})$$

Convolution based Laplace-Beltrami operator on Digital Surfaces

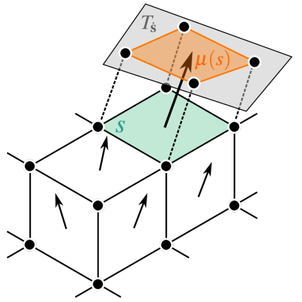
à-la [Belkin et al 08]

$$(L_h\,\tilde{u})(\mathbf{s}) := \frac{1}{t_h(4\pi t_h)^{\frac{d}{2}}} \sum_{\mathbf{r}\in S} e^{-\frac{||\mathbf{r}-\mathbf{s}||^2}{4t_h}} [\tilde{u}(\mathbf{r}) - \tilde{u}(\mathbf{s})]\mu(\mathbf{r})$$

Convolution based Laplace-Beltrami operator on Digital Surfaces

à-la [Belkin et al 08]

$$(L_h \tilde{u})(\mathbf{s}) := \frac{1}{t_h (4\pi t_h)^{\frac{d}{2}}} \sum_{\mathbf{r} \in S} e^{-\frac{||\mathbf{r}-\mathbf{s}||^2}{4t_h}} [\tilde{u}(\mathbf{r}) - \tilde{u}(\mathbf{s})] \mu(\mathbf{r})$$

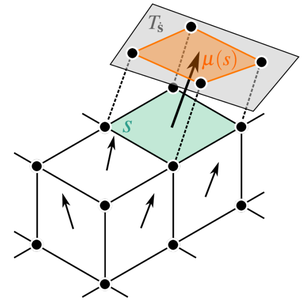


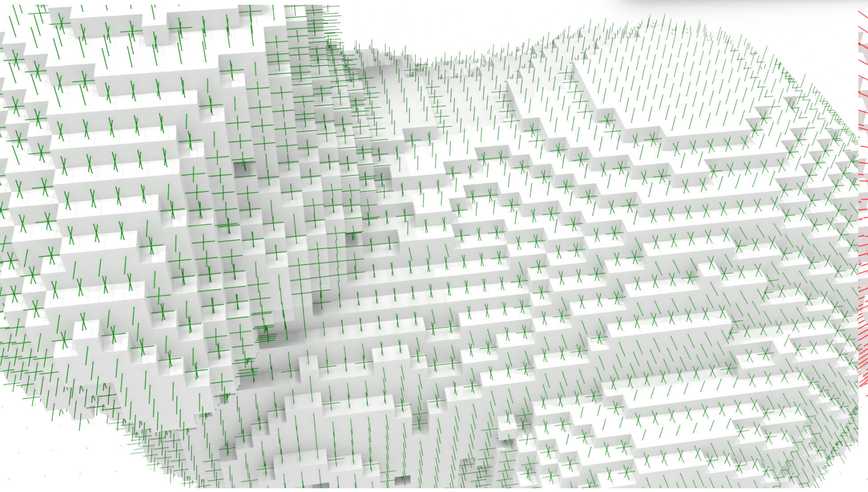
$$\mu(r) := \overrightarrow{n}_{elem} \cdot \overrightarrow{n}_{estim}$$

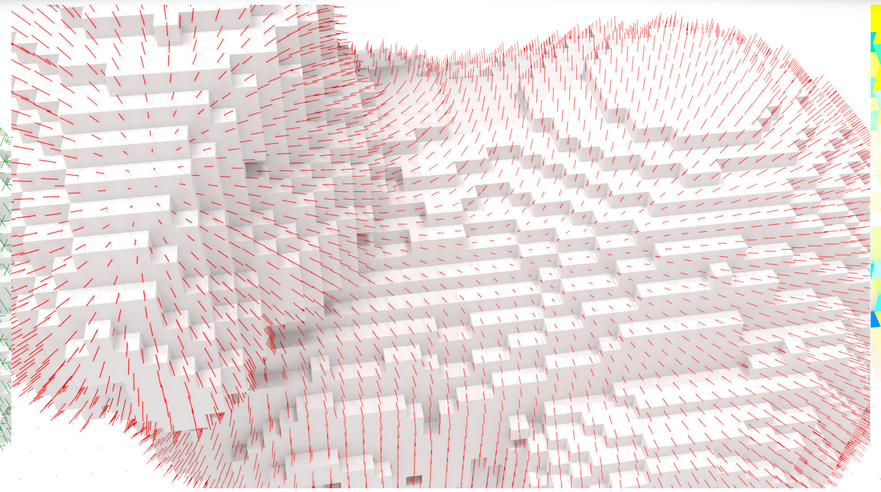
Convolution based Laplace-Beltrami operator on Digital Surfaces

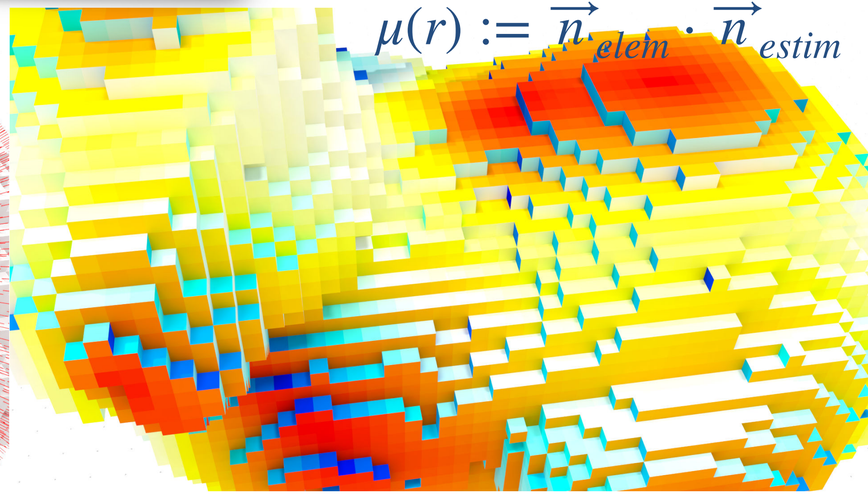
à-la [Belkin et al 08]

$$(L_h \tilde{u})(\mathbf{s}) := \frac{1}{t_h (4\pi t_h)^{\frac{d}{2}}} \sum_{\mathbf{r} \in S} e^{-\frac{||\mathbf{r}-\mathbf{s}||^2}{4t_h}} [\tilde{u}(\mathbf{r}) - \tilde{u}(\mathbf{s})] \mu(\mathbf{r})$$







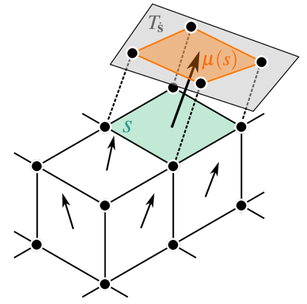


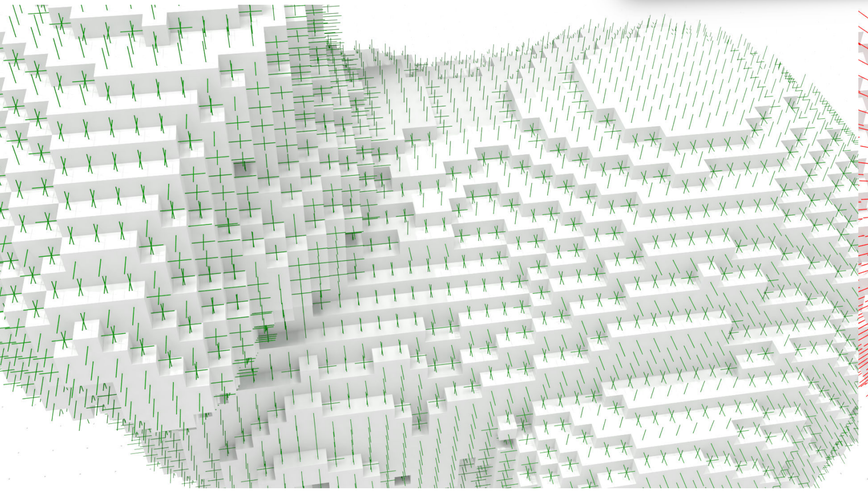
$$\mu(r) := \vec{n}_{elem} \cdot \vec{n}_{estim}$$

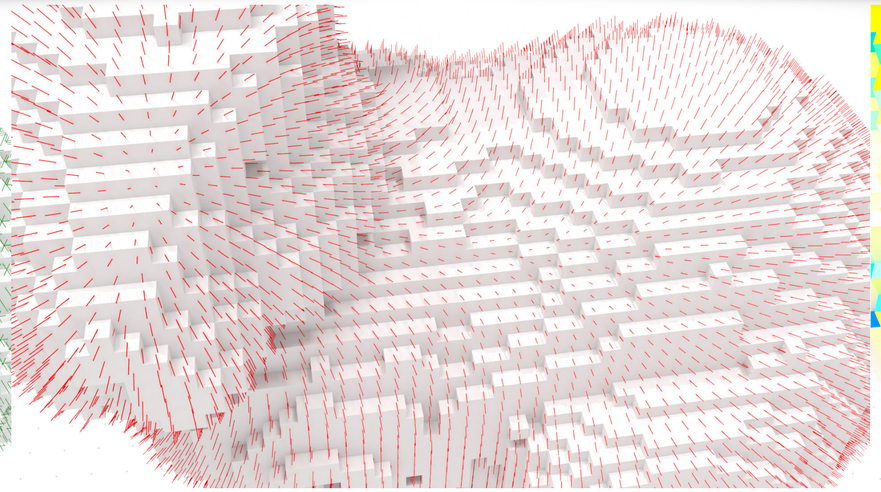
Convolution based Laplace-Beltrami operator on Digital Surfaces

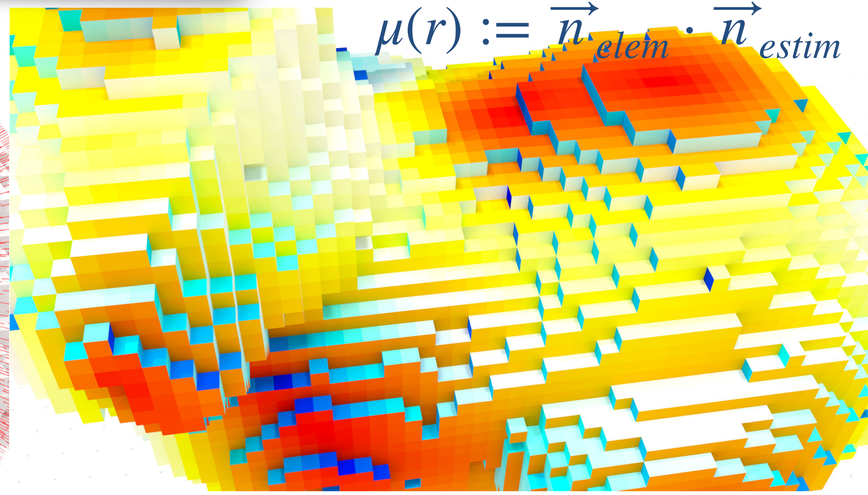
à-la [Belkin et al 08]

$$(L_h \tilde{u})(\mathbf{s}) := \frac{1}{t_h (4\pi t_h)^{\frac{d}{2}}} \sum_{\mathbf{r} \in S} e^{-\frac{||\mathbf{r}-\mathbf{s}||^2}{4t_h}} [\tilde{u}(\mathbf{r}) - \tilde{u}(\mathbf{s})] \mu(\mathbf{r})$$









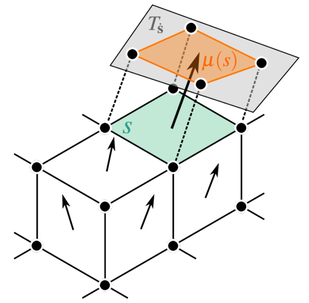
$$\mu(r) := \vec{n}_{elem} \cdot \vec{n}_{estim}$$

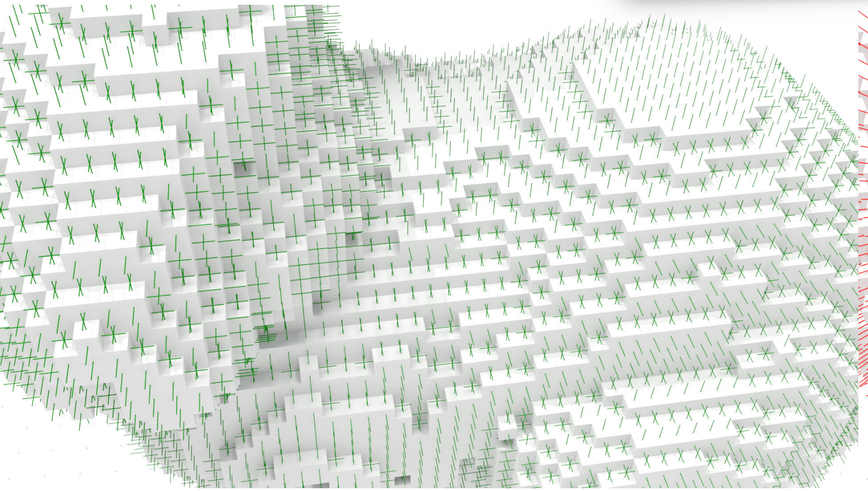
$$\left|(\Delta u)(\xi(s)) - (L_h \tilde{u})(s)\right| \leq \underbrace{\left|(\Delta u)(\xi(s)) - (\mathcal{L}_t u)(\xi(s))\right|}_{\text{[Belkin et al]}} + \underbrace{\left|(\mathcal{L}_t u)(\xi(s)) - (\mathcal{L}\tilde{u})(s)\right|}_{\text{Projection error}} + \underbrace{\left|(\mathcal{L}_t \tilde{u})(s) - (L_h \tilde{u})(s)\right|}_{\text{Digital integration error}}$$

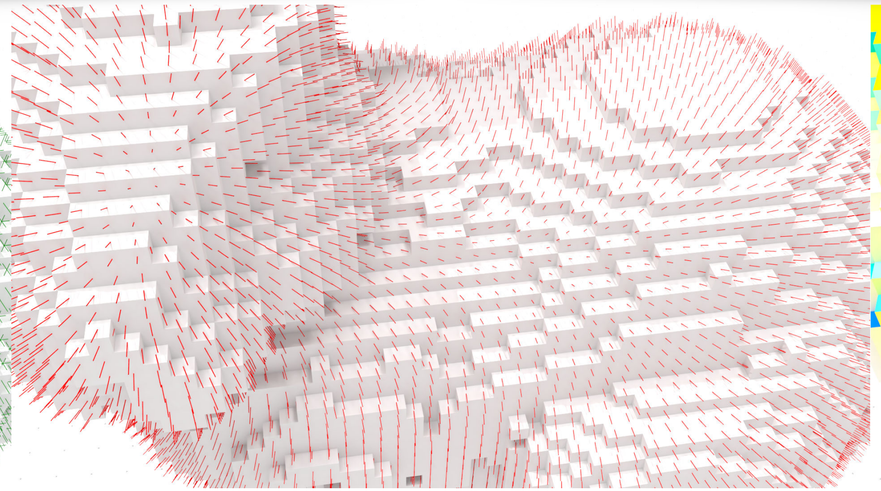
Convolution based Laplace-Beltrami operator on Digital Surfaces

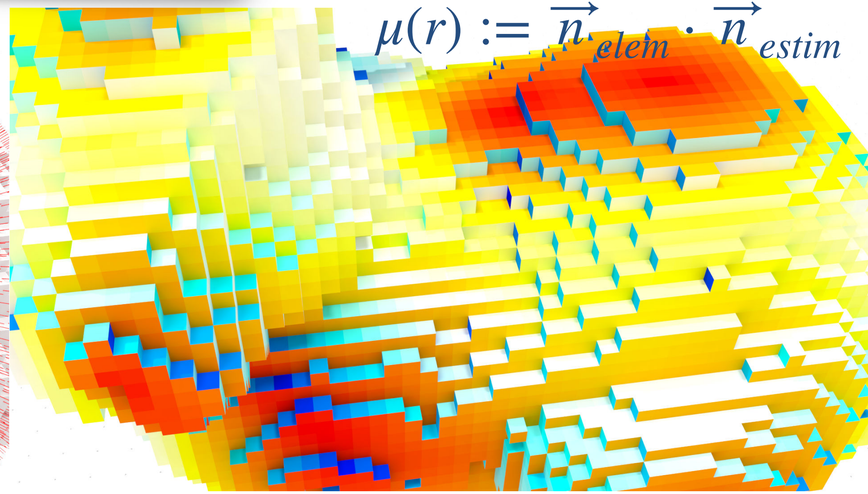
à-la [Belkin et al 08]

$$(L_h \tilde{u})(\mathbf{s}) := \frac{1}{t_h (4\pi t_h)^{\frac{d}{2}}} \sum_{\mathbf{r} \in S} e^{-\frac{||\mathbf{r}-\mathbf{s}||^2}{4t_h}} [\tilde{u}(\mathbf{r}) - \tilde{u}(\mathbf{s})] \mu(\mathbf{r})$$







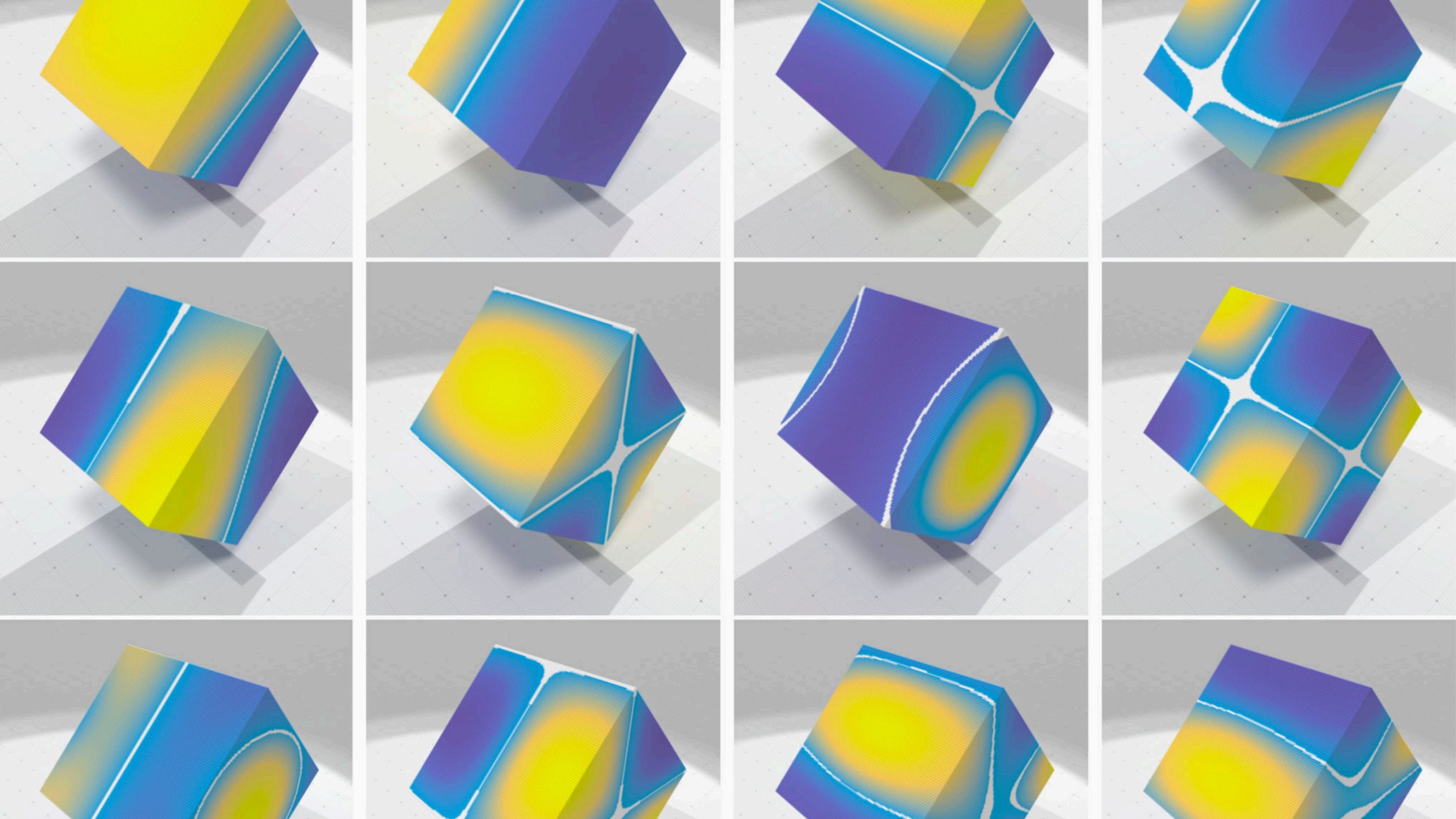


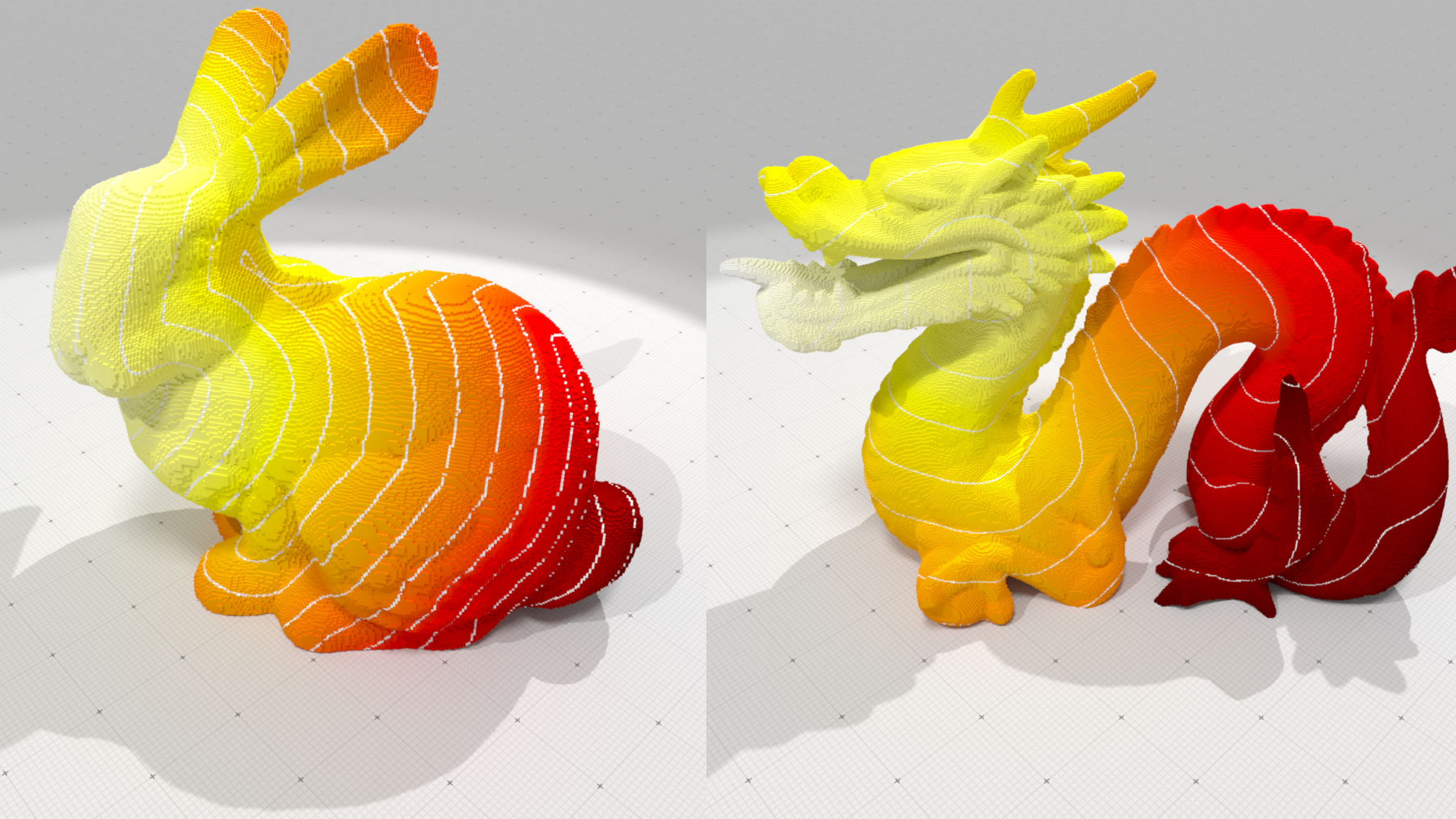
$$\mu(r) := \vec{n}_{elem} \cdot \vec{n}_{estim}$$

$$\left|(\Delta u)(\xi(s)) - (L_h \tilde{u})(s)\right| \leq \underbrace{\left|(\Delta u)(\xi(s)) - (\mathcal{L}_t u)(\xi(s))\right|}_{\text{[Belkin et al]}} + \underbrace{\left|(\mathcal{L}_t u)(\xi(s)) - (\mathcal{L}\tilde{u})(s)\right|}_{\text{Projection error}} + \underbrace{\left|(\mathcal{L}_t \tilde{u})(s) - (L_h \tilde{u})(s)\right|}_{\text{Digital integration error}}$$

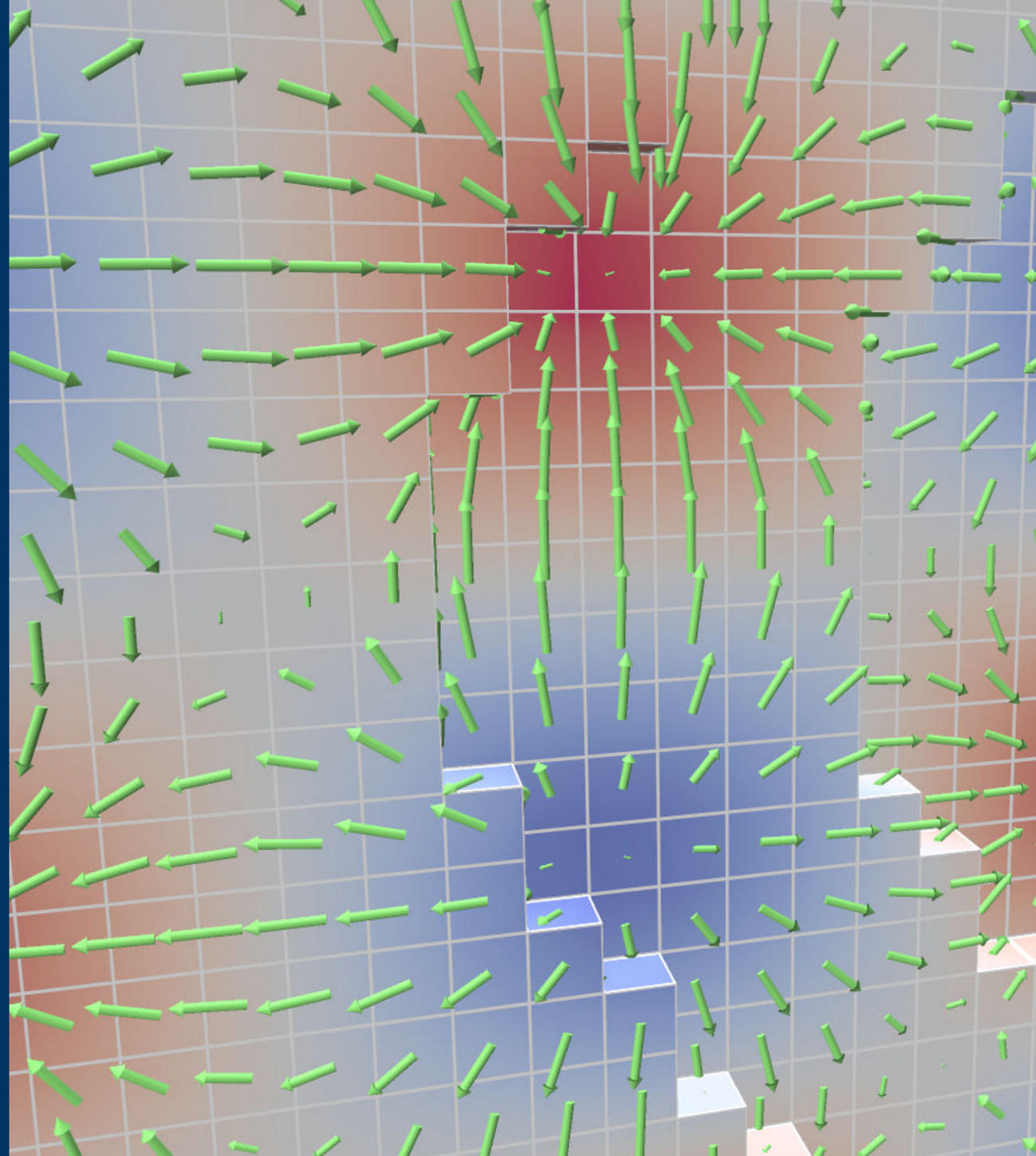
[Caissard et al 19]

- $(L_h \tilde{u})$ is *strongly consistent* when $h \to 0$
- but not local…

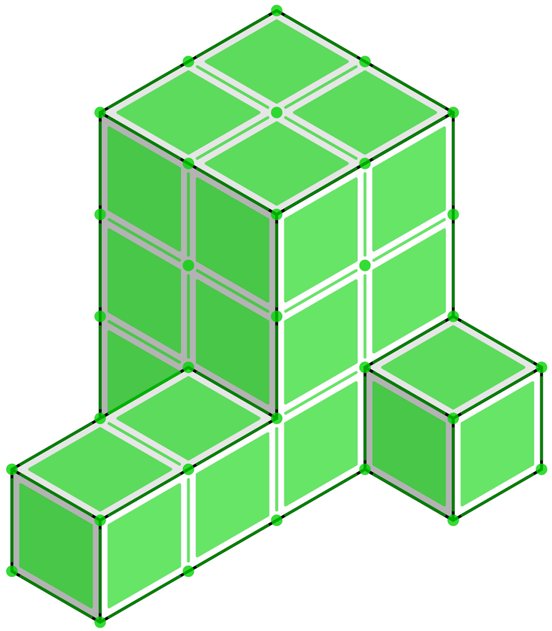


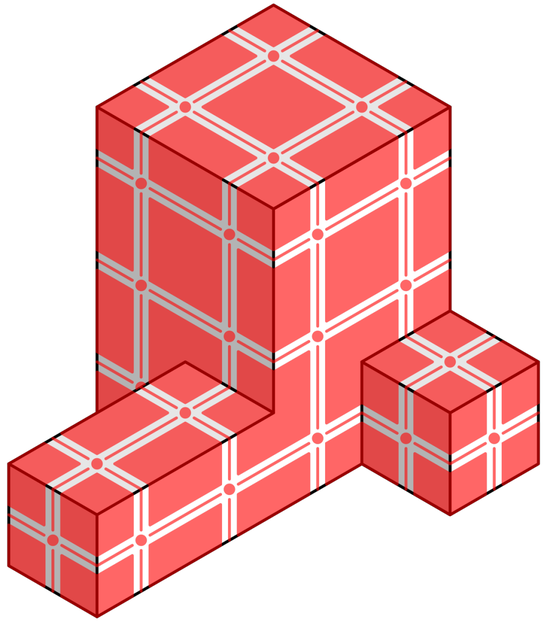


corrected digital calculus



Discrete Calculus *à la DEC*





$\mathrm{d} \quad \wedge \quad \star \quad \sharp \quad \flat$

$$\operatorname{div} F = \star\, \mathrm{d}(\star F^{\flat}) \qquad \operatorname{curl} F = (\star\, (\mathrm{d} F^{\flat}))^{\sharp}$$

$$\Delta \phi = (\star \mathrm{d} \star \mathrm{d}) \phi$$

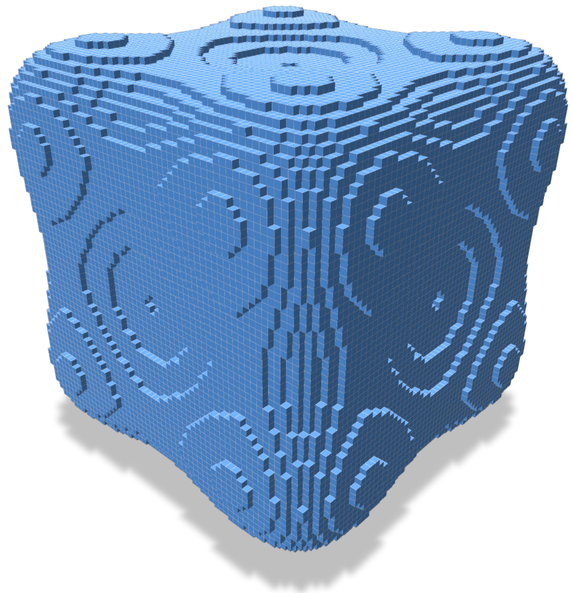
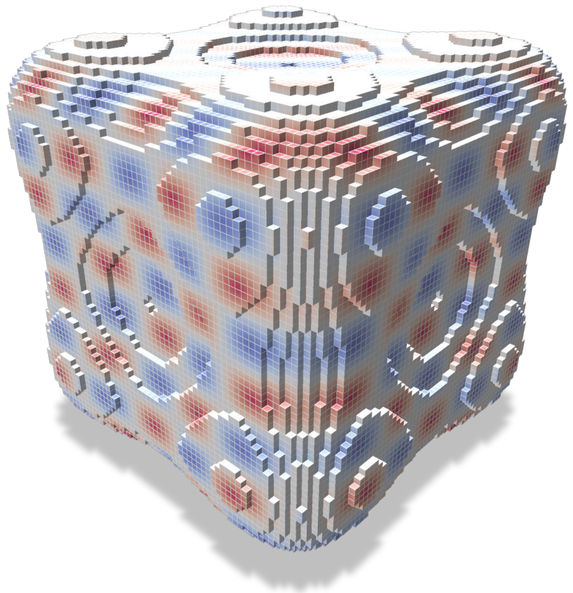
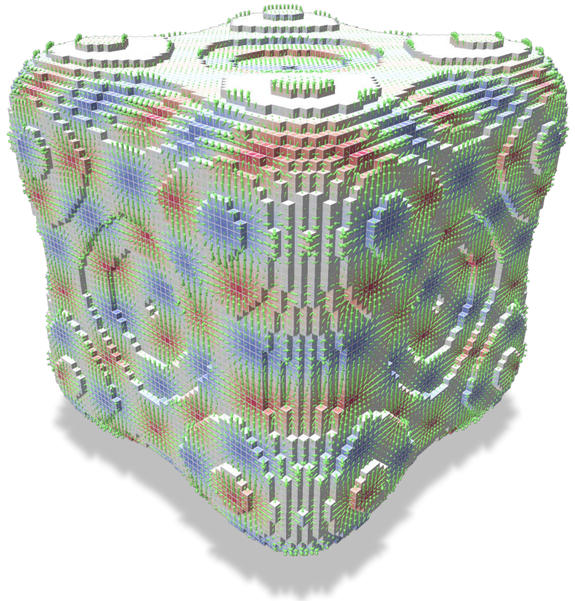
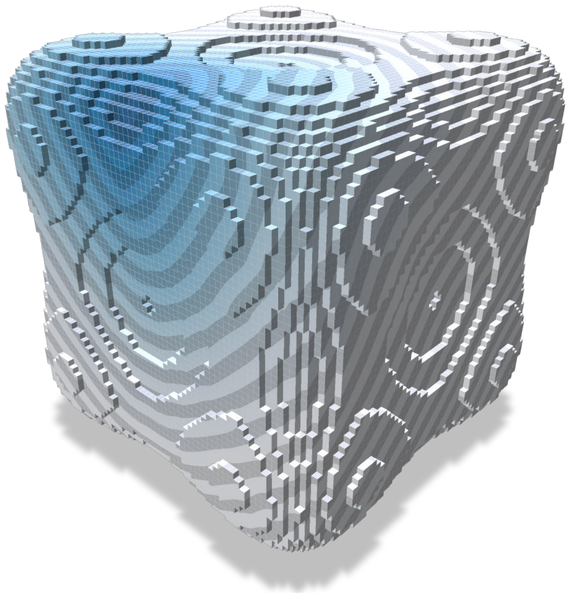
But:

- subtle combinatorial/topological construction
- non-trivial *correction* of the embedding ($T_p M$)

Discrete Differential Operators on Polygonal Meshes

[de Goes et al 20]

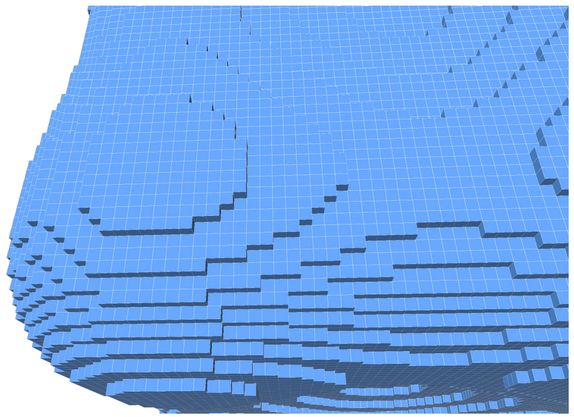
per face $\nabla, \nabla\cdot, \nabla\times, \sharp, \flat, \Delta \ldots$ Levi-Civita…

But still flat embedding hypothesis**…**

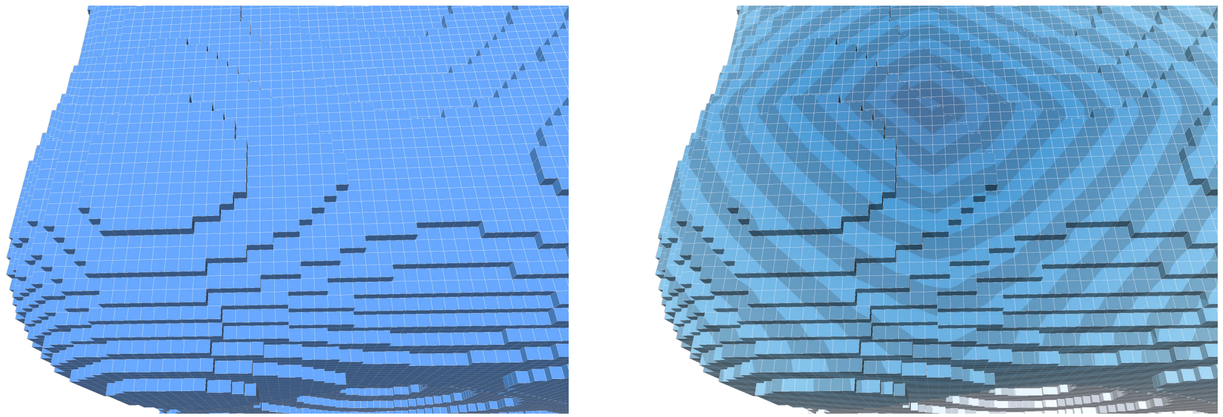
Discrete Differential Operators on Polygonal Meshes

[de Goes et al 20]



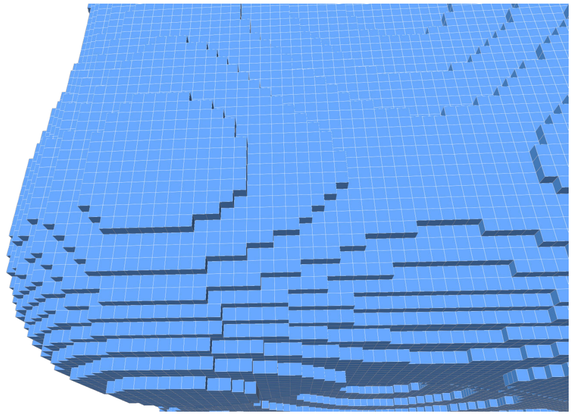
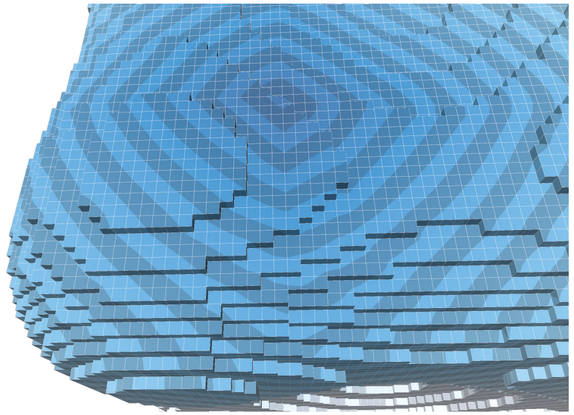
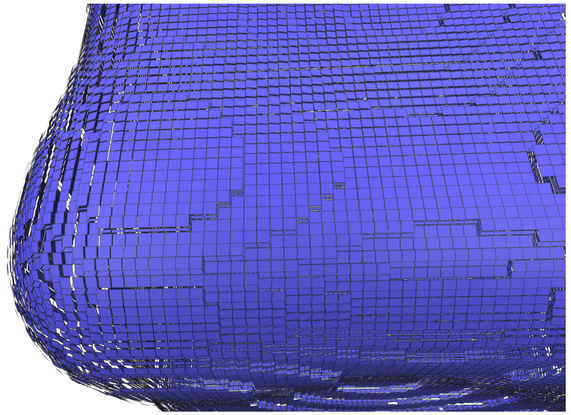
Discrete Differential Operators on Polygonal Meshes

[de Goes et al 20]



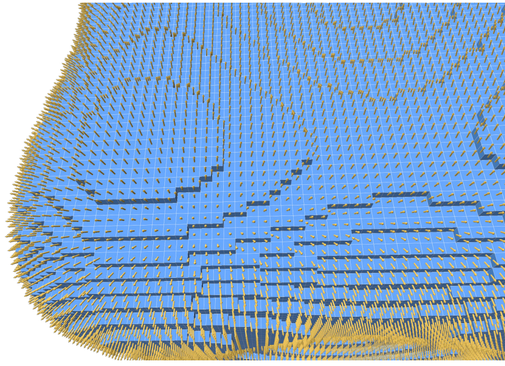
Discrete Differential Operators on Polygonal Meshes

[de Goes et al 20]

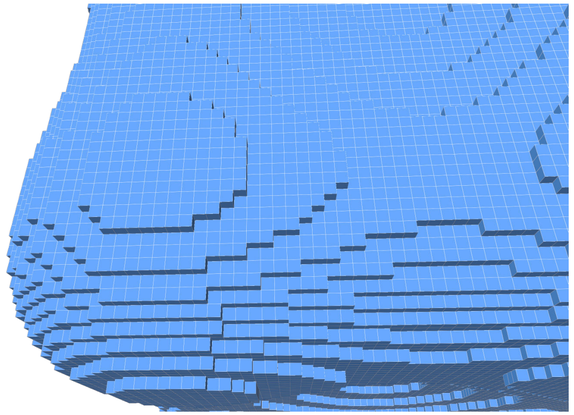
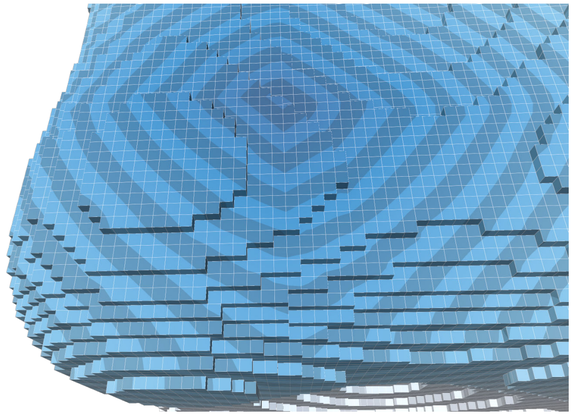
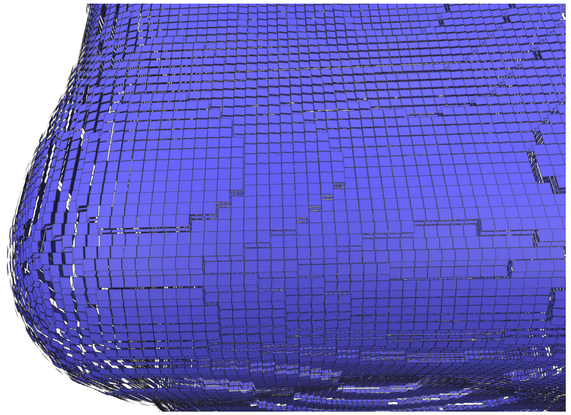
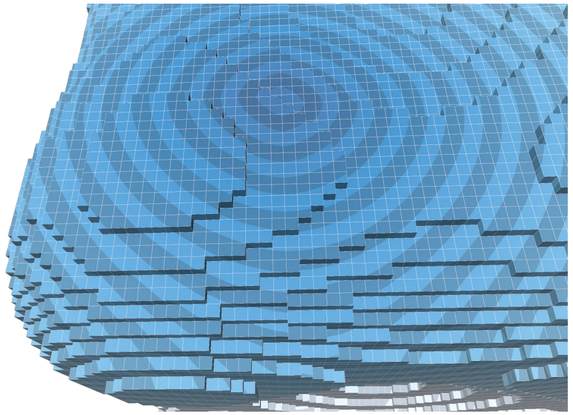
We can *correct* the face embedding using asymptotic convergence normal vector field

Challenges: advance corrections (e.g. on the Grassmanian, higher order schemes…) for asymptotic properties



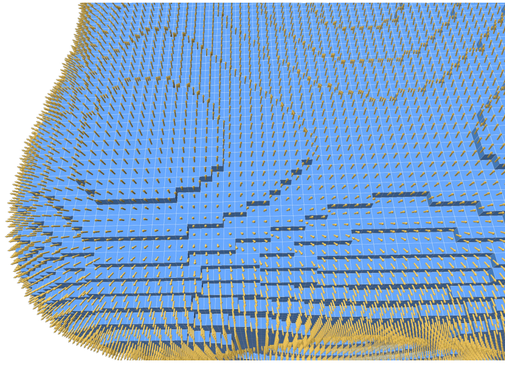
Discrete Differential Operators on Polygonal Meshes

[de Goes et al 20]

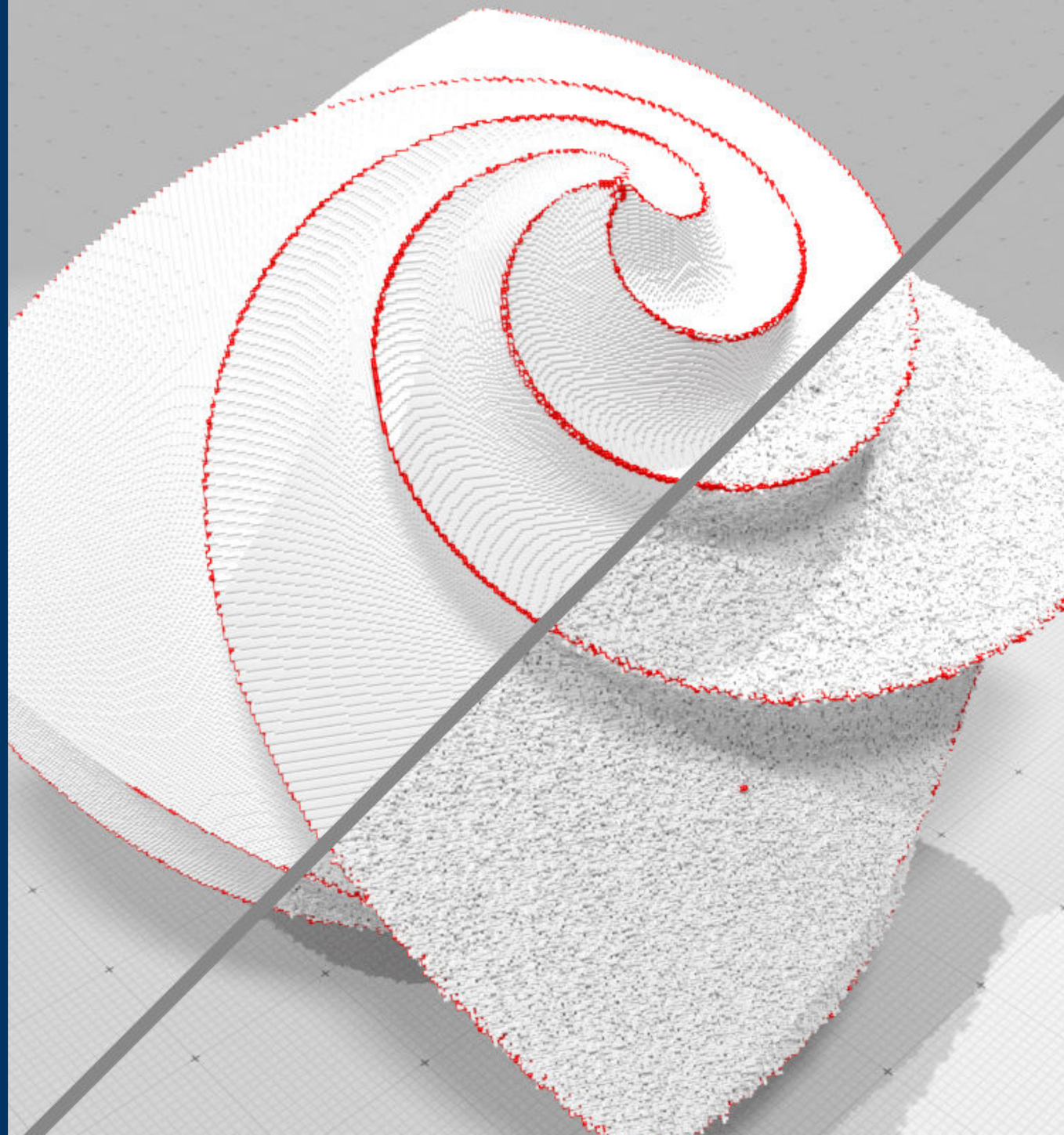





We can *correct* the face embedding using asymptotic convergence normal vector field

Challenges: advance corrections (e.g. on the Grassmanian, higher order schemes…) for asymptotic properties



quick wrap-up example

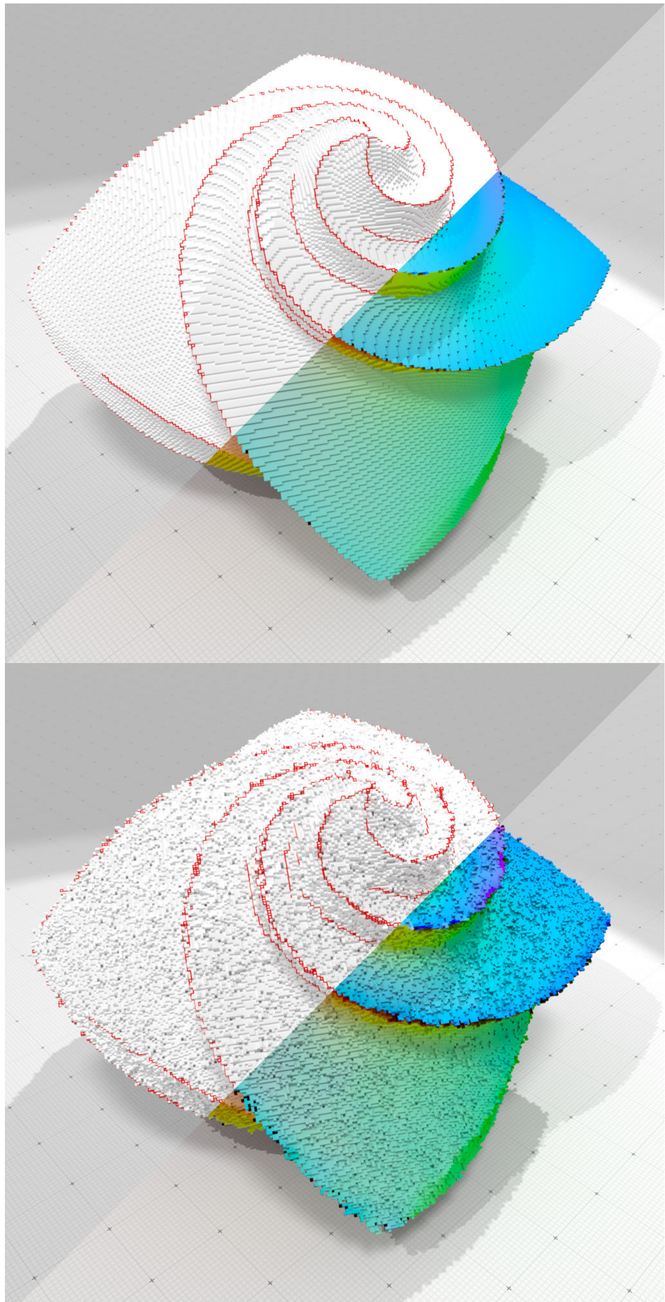


Piecewise smooth reconstruction

***Step1**: normal vector field reconstruction*

Ambrosio-Tortorelli functional: solve u,v s.t.

$$AT_\epsilon(u,v) = \alpha \int_M |u-g|^2 dx + \int_M |v\nabla u|^2 + \lambda\epsilon |\nabla v|^2 + \frac{1}{4\epsilon}|1-v|^2 dx$$



[C. et al 16]

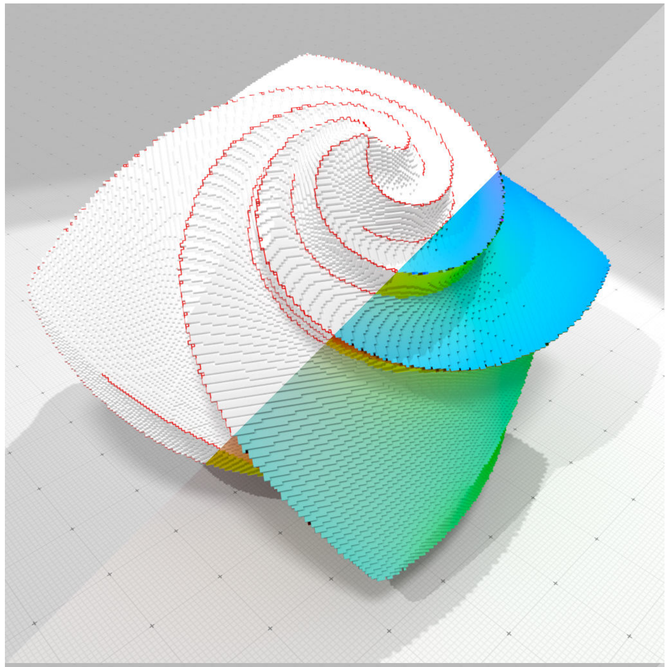
Piecewise smooth reconstruction

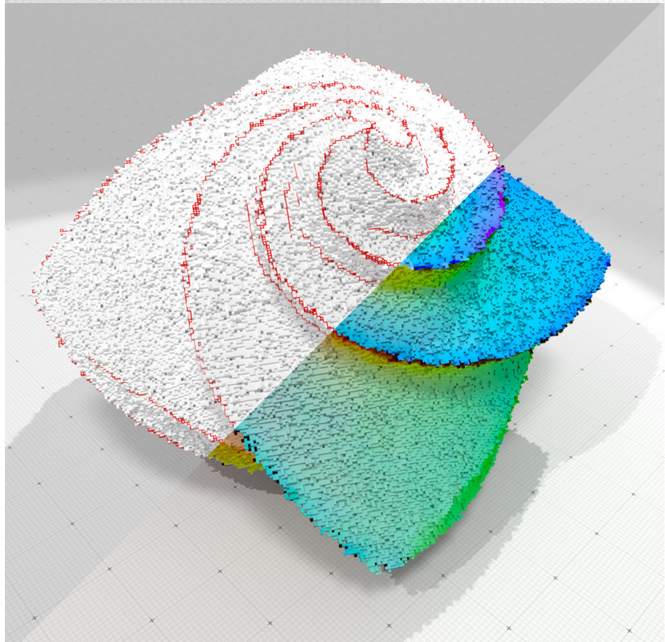
***Step1**: normal vector field reconstruction*

Ambrosio-Tortorelli functional: solve u,v s.t.

$$AT_\epsilon(u,v) = \alpha \int_M |u-g|^2 dx + \int_M |v\nabla u|^2 + \lambda\epsilon |\nabla v|^2 + \frac{1}{4\epsilon}|1-v|^2 dx$$

Reconstructed normals are close to the input ones





[C. et al 16]

Piecewise smooth reconstruction

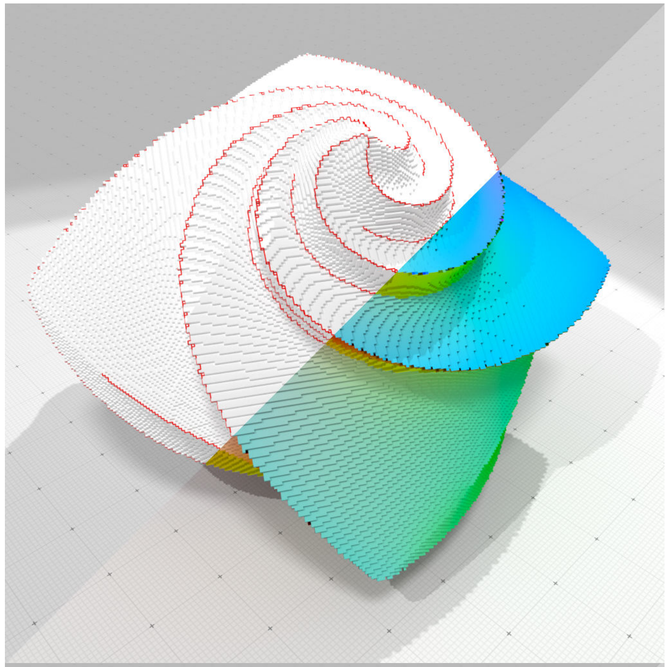
***Step1**: normal vector field reconstruction*

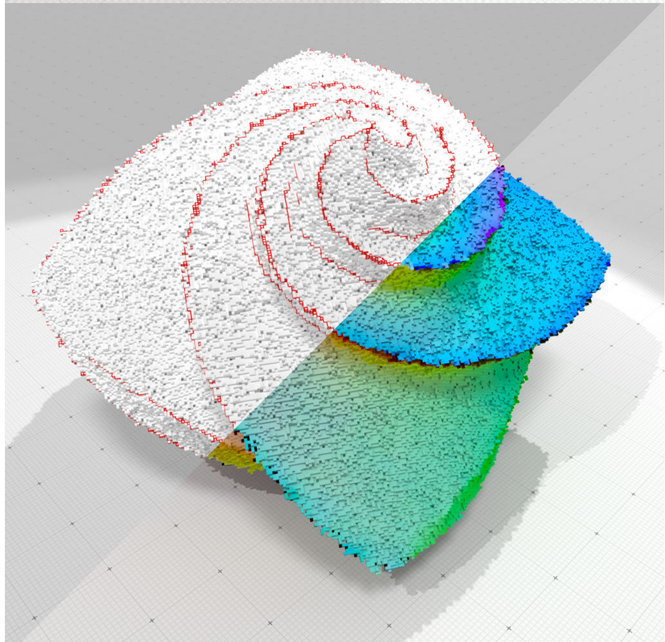
Ambrosio-Tortorelli functional: solve u,v s.t.

$$AT_\epsilon(u,v) = \alpha \int_M |u-g|^2 dx + \int_M |v\nabla u|^2 + \lambda\epsilon |\nabla v|^2 + \frac{1}{4\epsilon}|1-v|^2 dx$$

Reconstructed normals are close to the input ones

Normal field must be smooth except at singularities v





[C. et al 16]

Piecewise smooth reconstruction

***Step1**: normal vector field reconstruction*

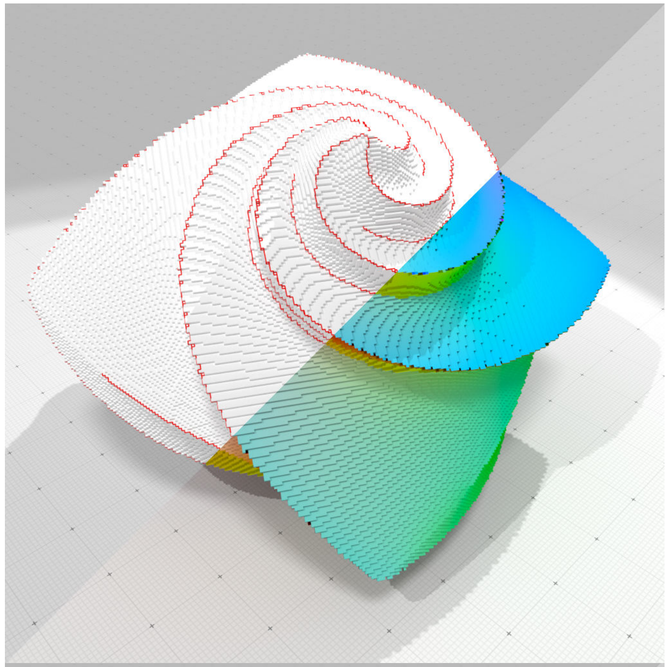
Ambrosio-Tortorelli functional: solve u,v s.t.

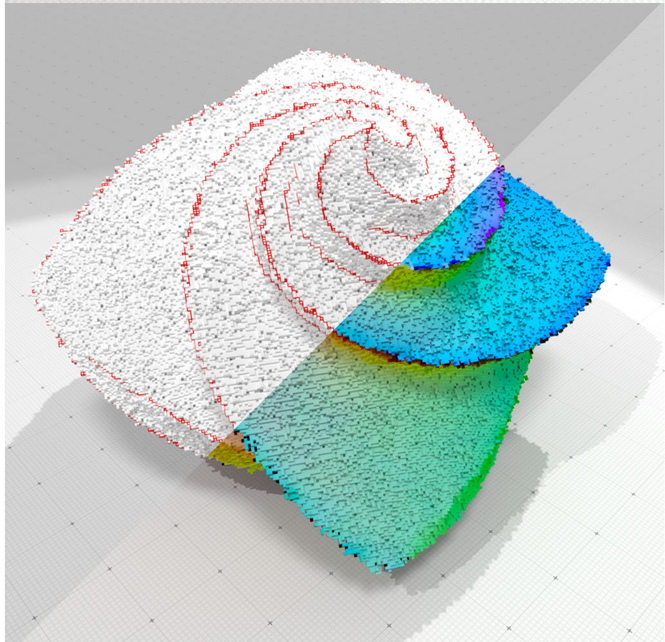
$$AT_\epsilon(u,v) = \alpha \int_M |u-g|^2 dx + \int_M |v\nabla u|^2 + \lambda\epsilon |\nabla v|^2 + \frac{1}{4\epsilon}|1-v|^2 dx$$

Reconstructed normals are close to the input ones

Normal field must be smooth except at singularities v

Penalizes the *length* of singularities





[C. et al 16]

Piecewise smooth reconstruction

***Step1**: normal vector field reconstruction*

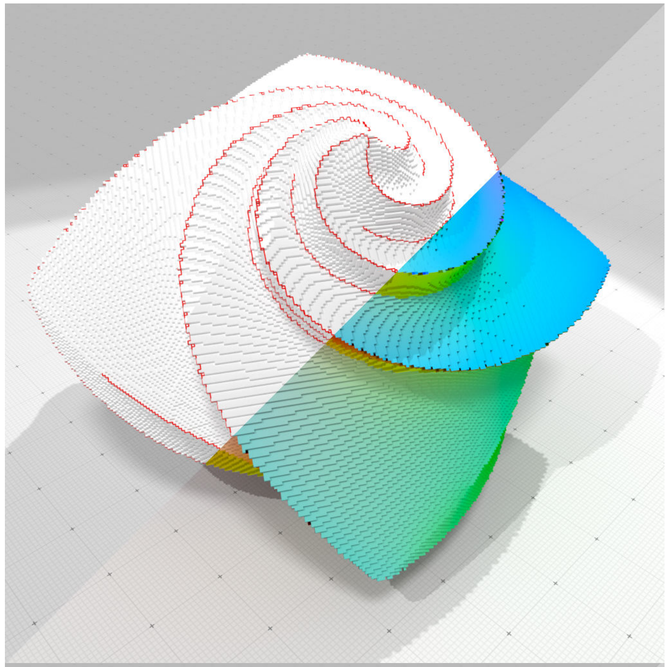
Ambrosio-Tortorelli functional: solve u,v s.t.

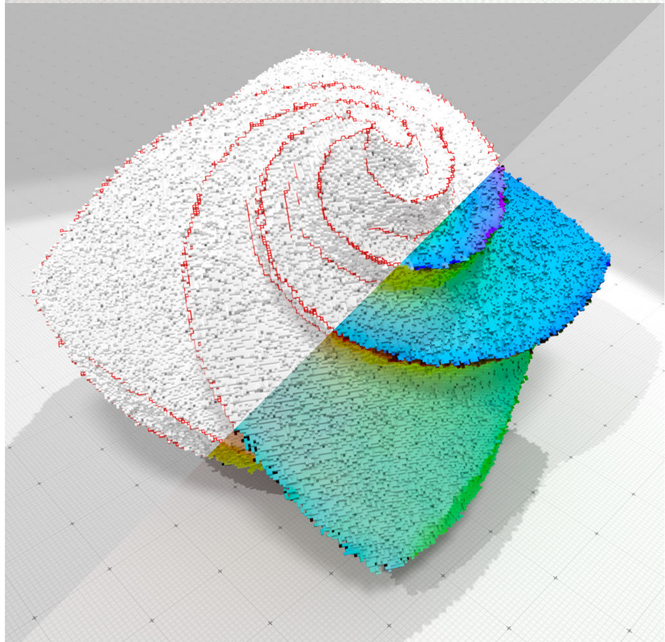
$$AT_\epsilon(u,v) = \alpha \int_M |u-g|^2 dx + \int_M |v\nabla u|^2 + \lambda\epsilon |\nabla v|^2 + \frac{1}{4\epsilon}|1-v|^2 dx$$

Reconstructed normals are close to the input ones

Normal field must be smooth except at singularities v

Penalizes the *length* of singularities



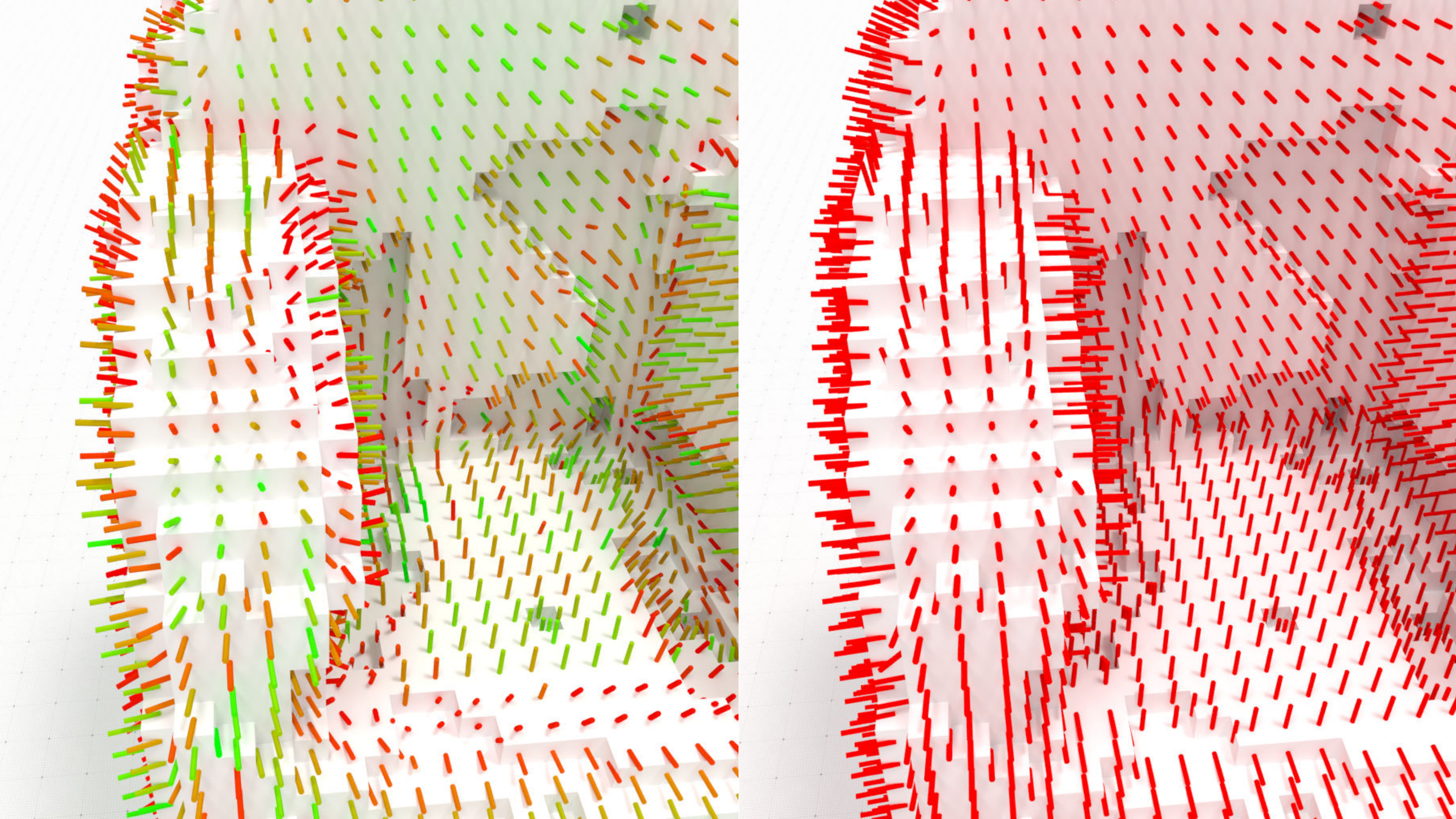


digital DEC:

$$AT_\epsilon(u,v) = \alpha \sum_{i=1}^{3} \langle u_i - g_i, u_i - g_i \rangle_{\overline{0}} + \sum_{i=1}^{3} \langle v \wedge d_{\overline{0}} u_i, v \wedge d_{\overline{0}} u_{i\overline{1}} \rangle_{\overline{1}} + \lambda\epsilon \langle d_0 v, d_0 v \rangle_1 + \frac{\lambda}{4\epsilon} \langle 1-v, 1-v \rangle_0$$

\+ energy is convex for fixed u or v $\Rightarrow$ alternate minimization

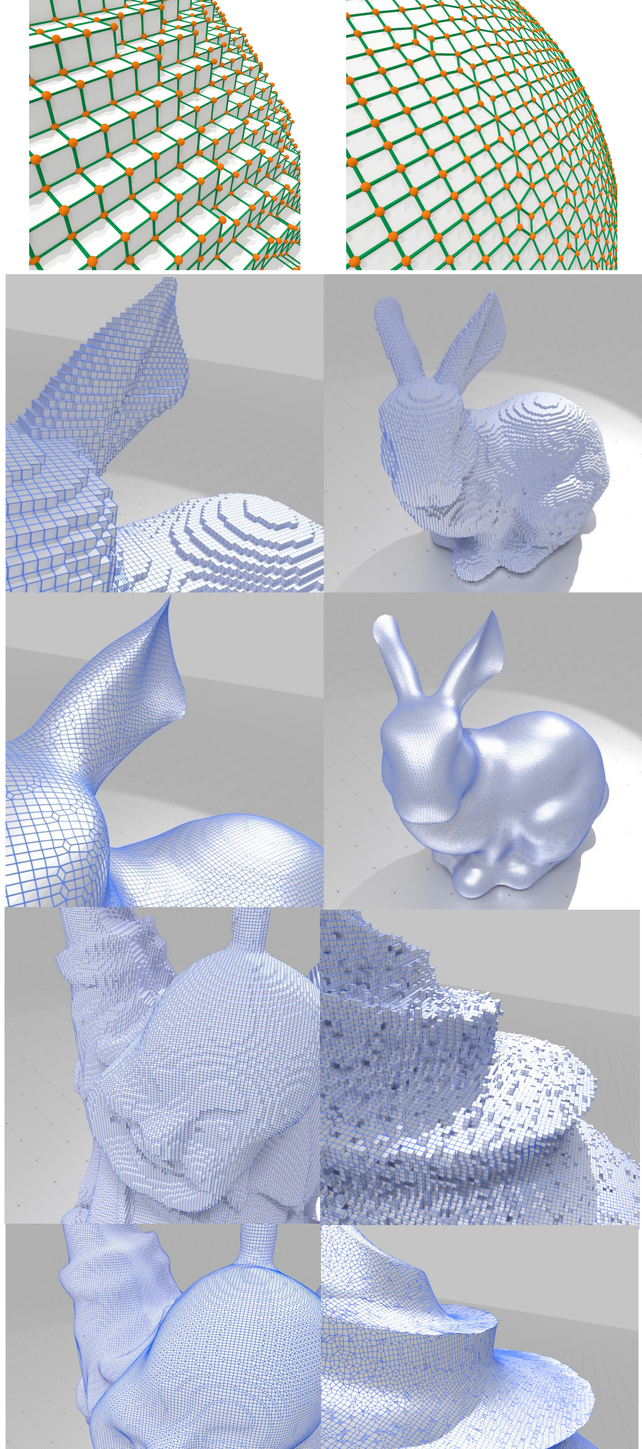
[C. et al 16]



Piecewise smooth reconstruction

***Step 2**: surface reconstruction*

$$\mathscr{E}(\hat{P}) := \alpha \sum_{i=1}^{n} \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|^2 + \beta \sum_{f \in F} \sum_{\hat{\mathbf{e}}_j \in \partial f} (\hat{\mathbf{e}}_j \cdot \mathbf{n}_f)^2 + \gamma \sum_{i=1}^{n} \|\hat{\mathbf{p}}_i - \hat{\mathbf{b}}_i\|^2$$

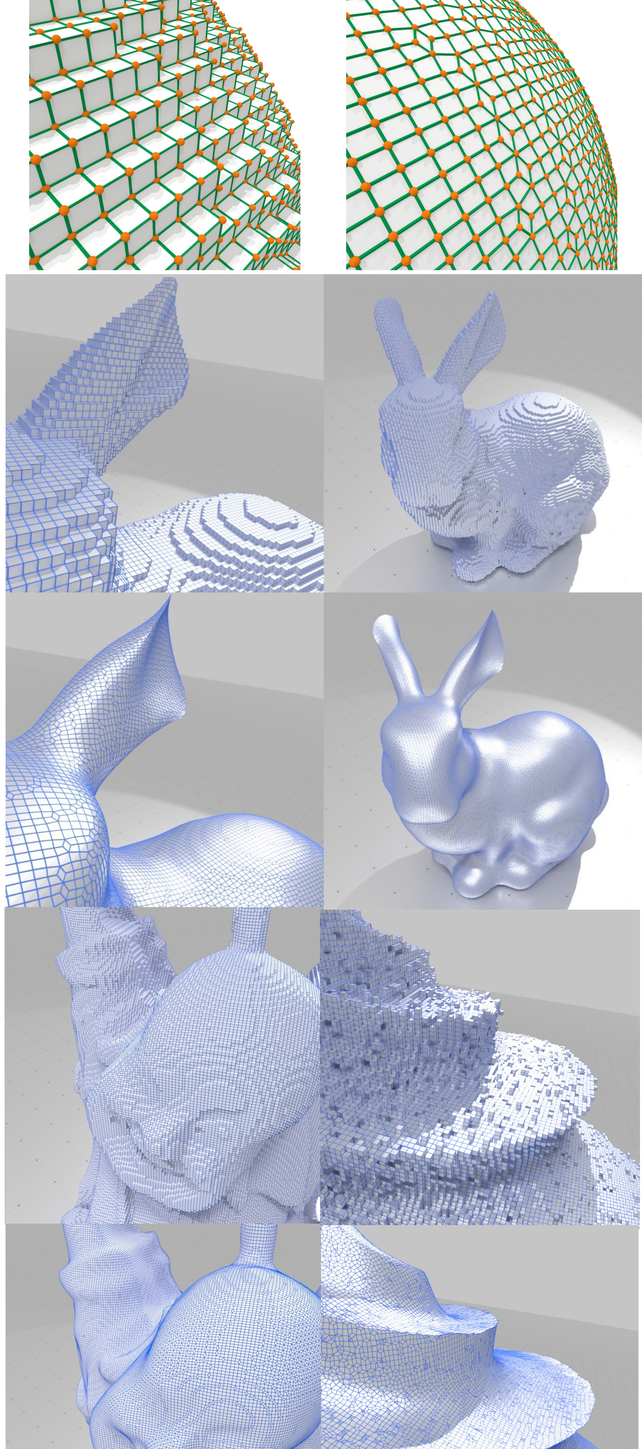


Piecewise smooth reconstruction

Step 2: *surface reconstruction*

$$\mathscr{E}(\hat{P}) := \alpha \sum_{i=1}^{n} \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|^2 + \beta \sum_{f \in F} \sum_{\hat{\mathbf{e}}_j \in \partial f} (\hat{\mathbf{e}}_j \cdot \mathbf{n}_f)^2 + \gamma \sum_{i=1}^{n} \|\hat{\mathbf{p}}_i - \hat{\mathbf{b}}_i\|^2$$

optimized vertices are not too far from original ones



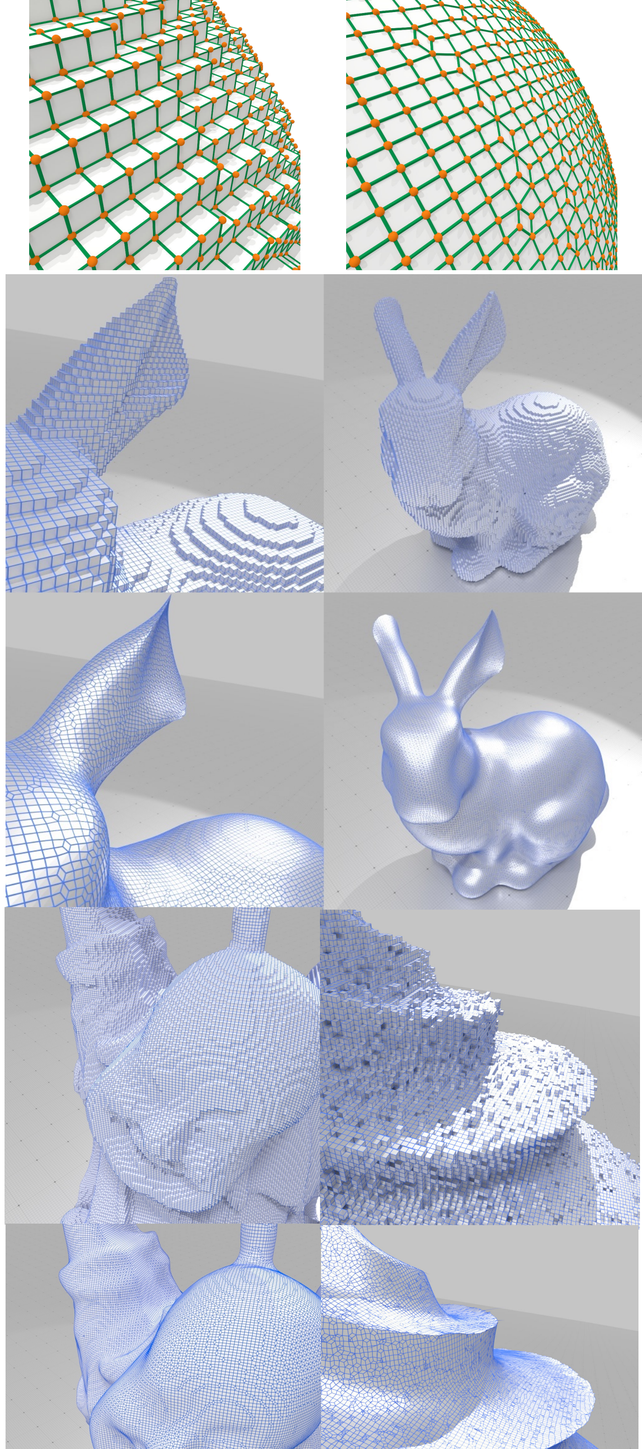
Piecewise smooth reconstruction

Step 2: *surface reconstruction*

$$\mathscr{E}(\hat{P}) := \alpha \sum_{i=1}^{n} \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|^2 + \beta \sum_{f \in F} \sum_{\hat{\mathbf{e}}_j \in \partial f} (\hat{\mathbf{e}}_j \cdot \mathbf{n}_f)^2 + \gamma \sum_{i=1}^{n} \|\hat{\mathbf{p}}_i - \hat{\mathbf{b}}_i\|^2$$

optimized vertices are not too far from original ones

Edges must be as orthogonal as possible to the given normal vectors



Piecewise smooth reconstruction

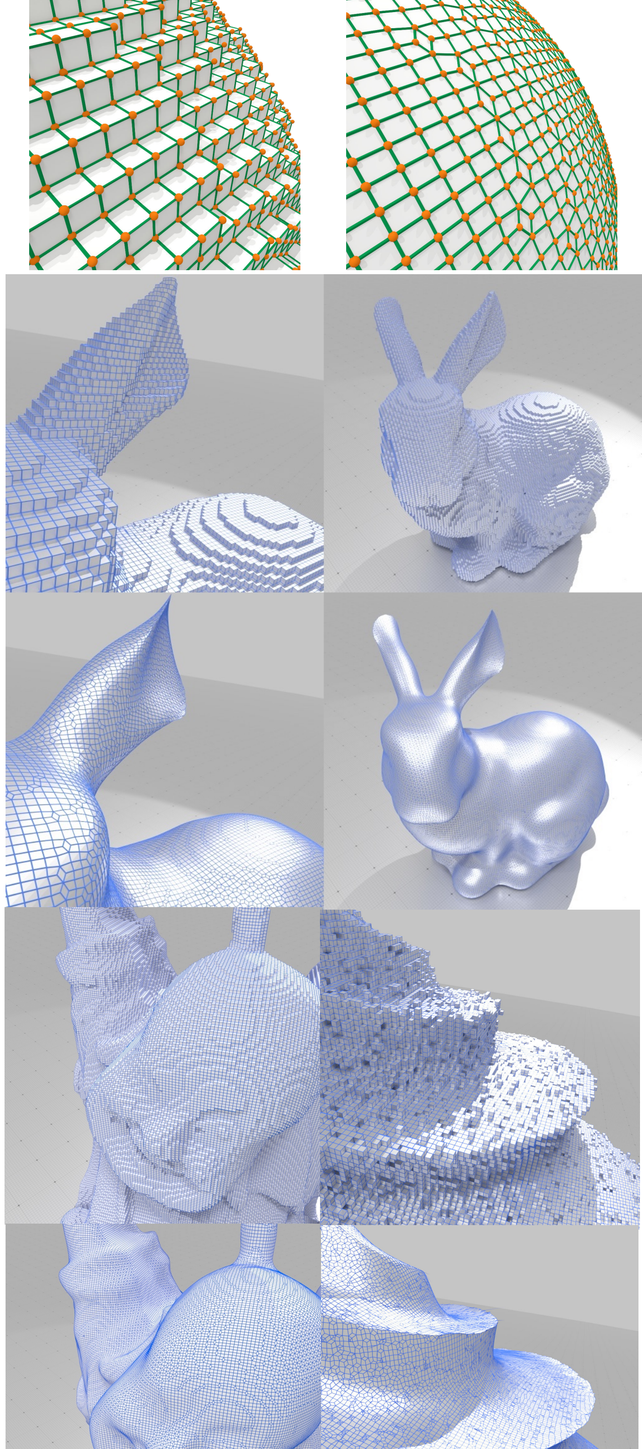
Step 2: *surface reconstruction*

$$\mathscr{E}(\hat{P}) := \alpha \sum_{i=1}^{n} \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|^2 + \beta \sum_{f \in F} \sum_{\hat{\mathbf{e}}_j \in \partial f} (\hat{\mathbf{e}}_j \cdot \mathbf{n}_f)^2 + \gamma \sum_{i=1}^{n} \|\hat{\mathbf{p}}_i - \hat{\mathbf{b}}_i\|^2$$

optimized vertices are not too far from original ones

Edges must be as orthogonal as possible to the given normal vectors

Fairness term



Piecewise smooth reconstruction

***Step 2**: surface reconstruction*

$$\mathscr{E}(\hat{P}) := \alpha \sum_{i=1}^{n} \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|^2 + \beta \sum_{f \in F} \sum_{\hat{\mathbf{e}}_j \in \partial f} (\hat{\mathbf{e}}_j \cdot \mathbf{n}_f)^2 + \gamma \sum_{i=1}^{n} \|\hat{\mathbf{p}}_i - \hat{\mathbf{b}}_i\|^2$$

optimized vertices are not too far from original ones

Edges must be as orthogonal as possible to the given normal vectors

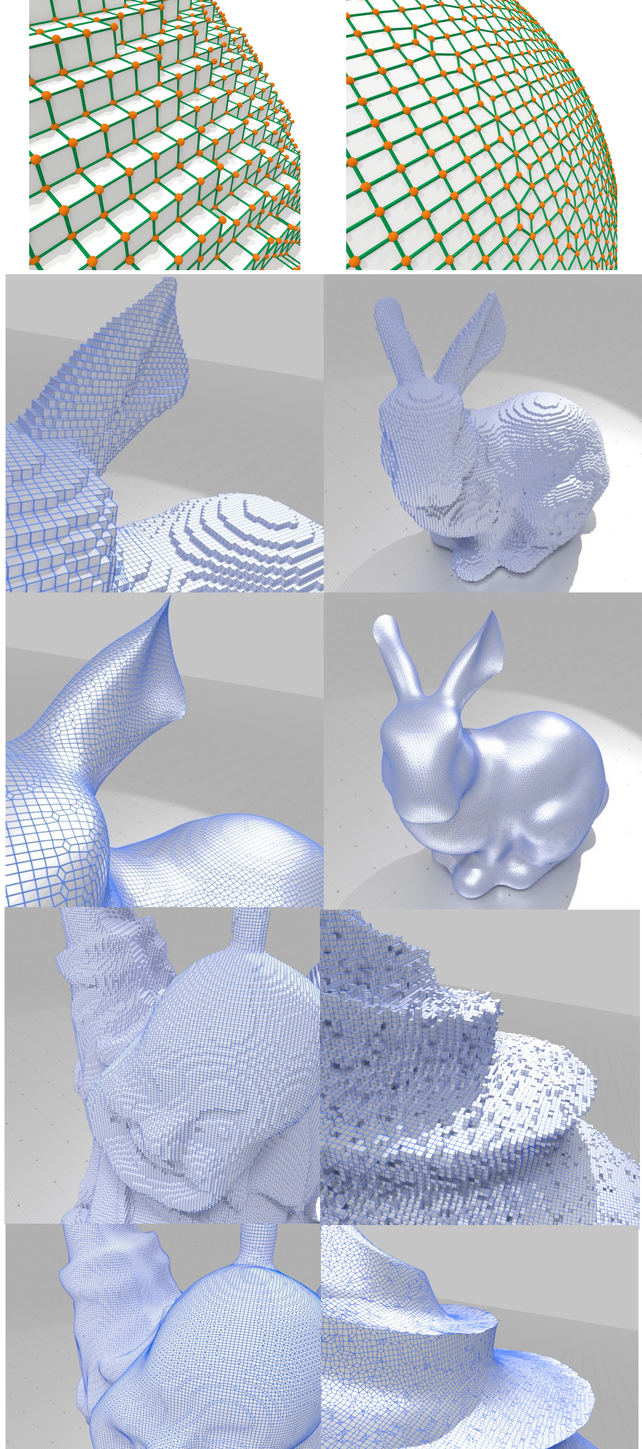
Fairness term

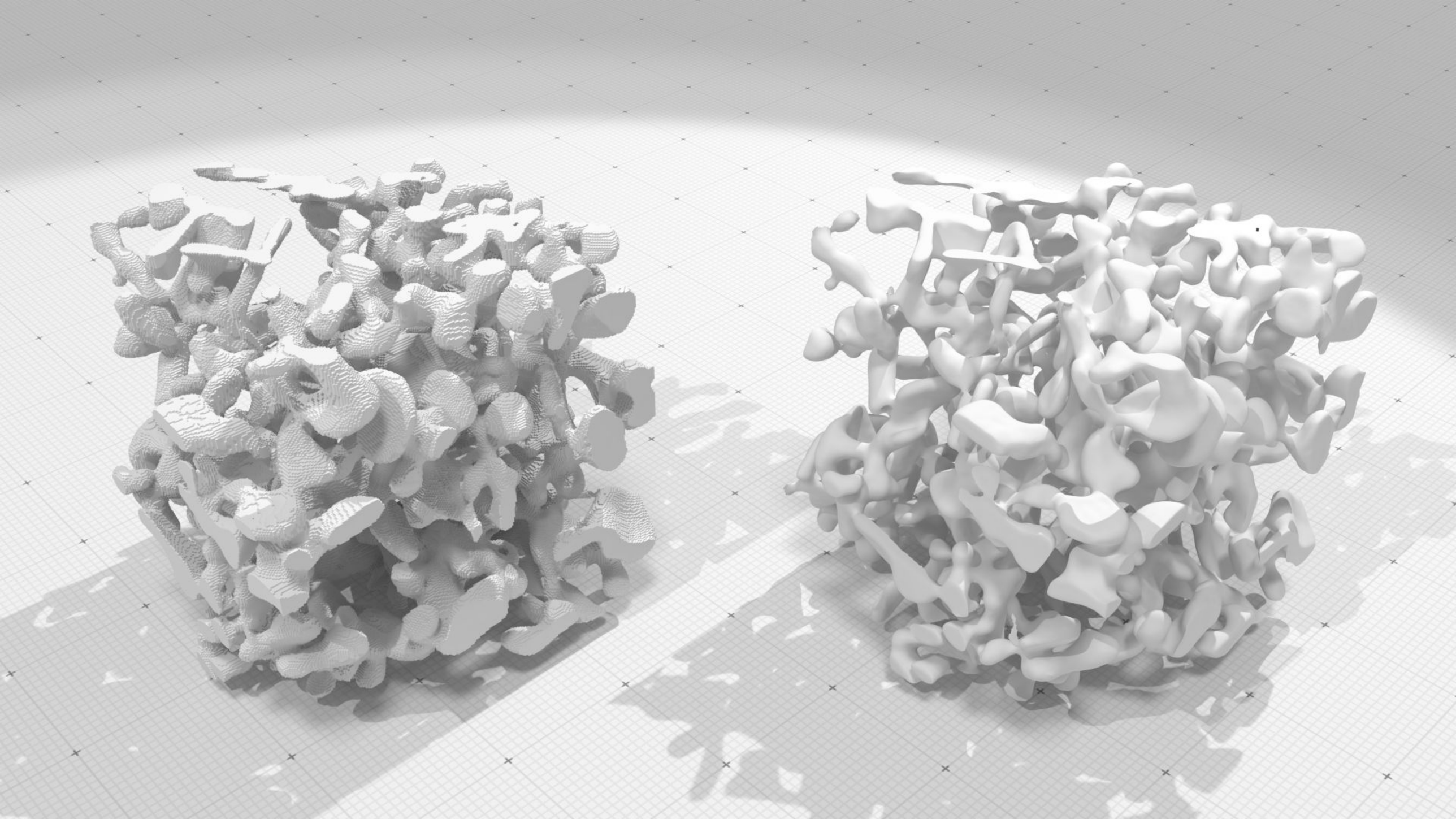
Using multigrid convergent normal vector field or its piecewise smooth regularization:

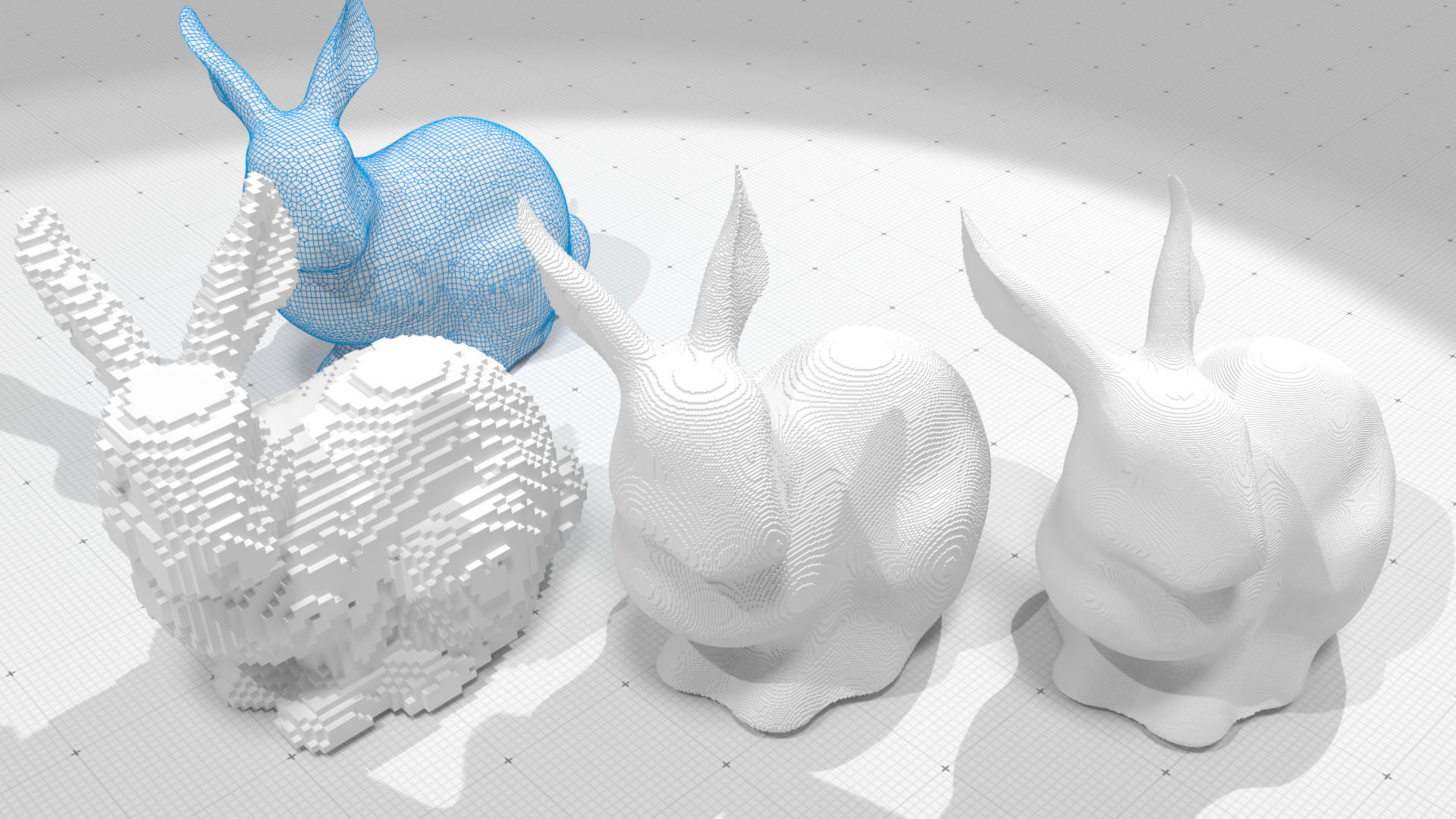
$$\frac{1}{n} \sum_{i=1}^{n} \|\mathbf{p}_i^* - \mathbf{p}_i\| \leq C \cdot h$$

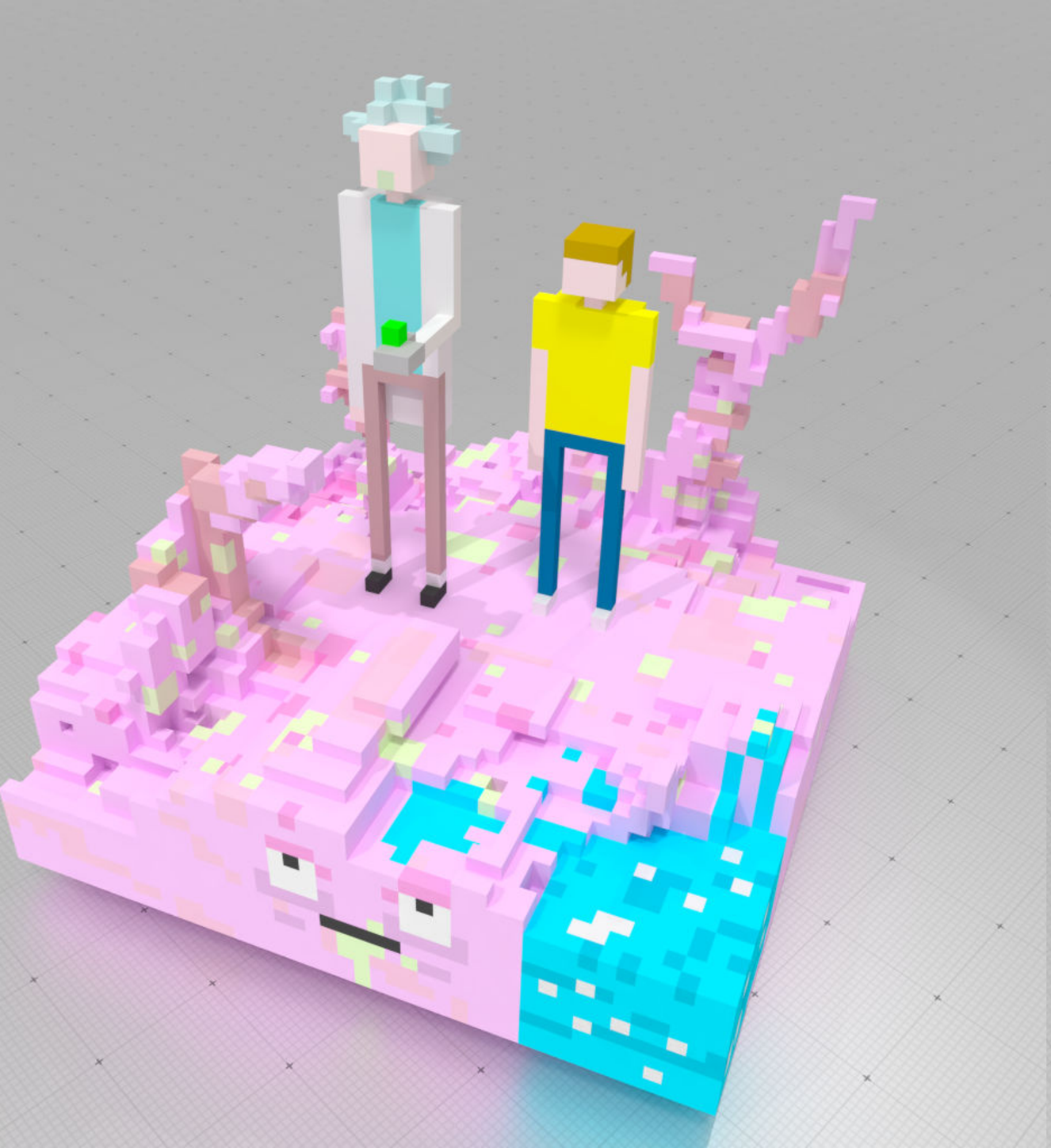
$$\frac{1}{n} \sum_{i=1}^{n} d(\mathbf{p}_i^*, \partial M) \leq C' \cdot h$$

+topological guarantee
+multi-label case
+fast GPU based minimization
+…
[C. et al 21]

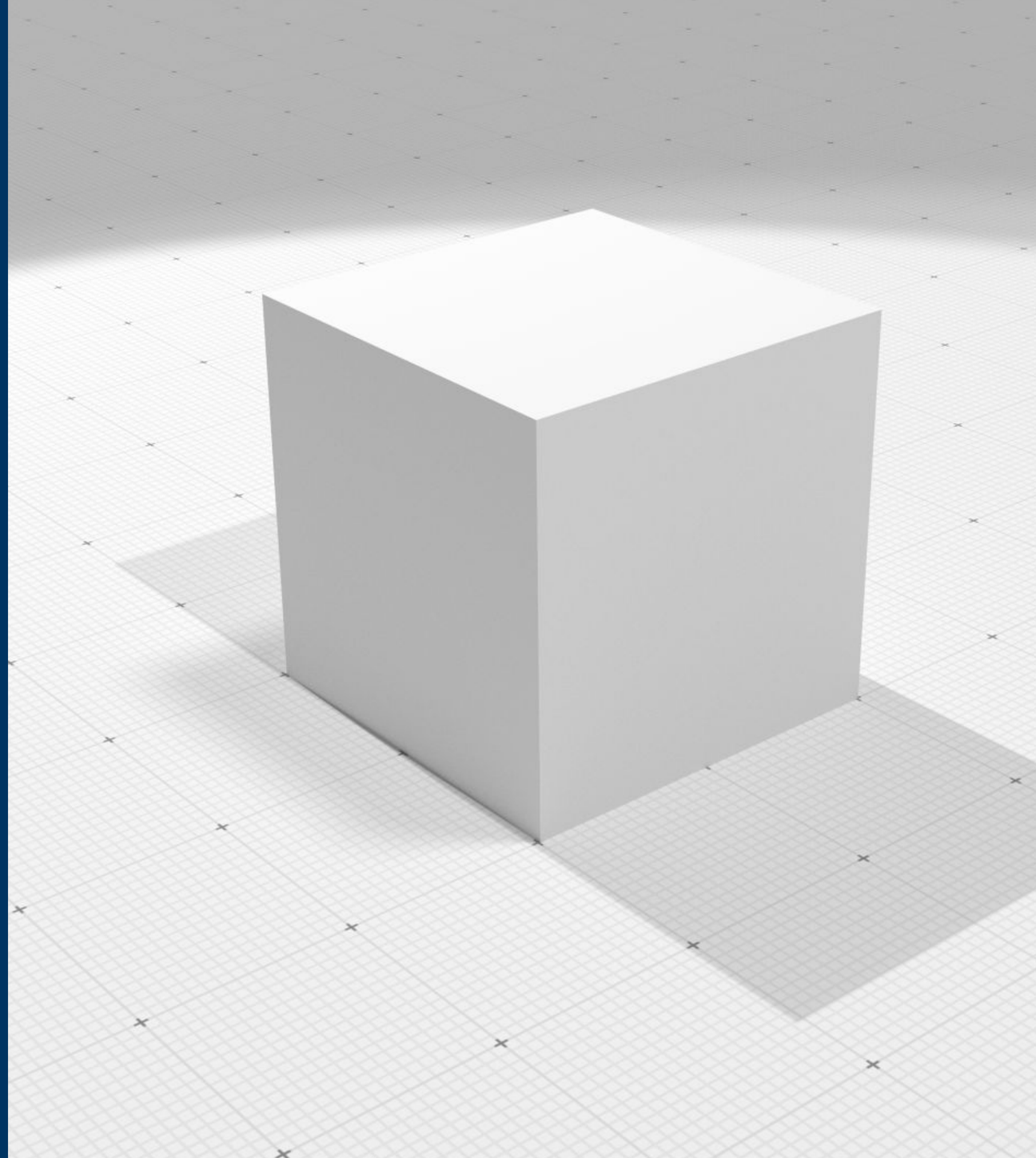








conclusion



Conclusion

Topology and geometry processing on regular data:

- fast algorithms thanks to the regularity of the data
- simple topological structure
- integer based computations
- advanced surface based geometry processing

… in $\mathbb{Z}^d$

DGtal

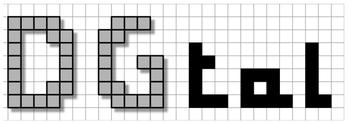
dgtal.org

https://github.com/dcoeurjo/SGP-GraduateSchool-digitalgeometry

(slides + code)

Challenges

- Corrected digital calculus, what kind of guarantee can we get?
- DEC operators targeting the limit surface (à-la *Subdivision Exterior Calculus*)
- Localized geometry processing operators on DAG Sparse Voxel Octrees



dgtal.org

https://github.com/dcoeurjo/SGP-GraduateSchool-digitalgeometry

(slides + code)

References

[Villanueva et al 17] Alberto Jaspe Villanueva, Fabio Marton, and Enrico Gobbetti, Symmetry-aware Sparse Voxel DAGs (SSVDAGs) for compression-domain tracing of high-resolution geometric scenes, Journal of Computer Graphics Techniques (JCGT), vol. 6, no. 2, 1-30, 2017

[Chen et al 2020] Half-Space Power Diagrams and Discrete Surface Offsets, Zhen Chen, Daniele Panozzo, Jérémie Dumas. In TVCG, 2019.

[C. et al 07] Optimal Separable Algorithms to Compute the Reverse Euclidean Distance Transformation and Discrete Medial Axis in Arbitrary Dimension, David Coeurjolly Annick Montanvert, IEEE Transactions on Pattern Analysis and Machine Intelligence, March 2007

[Martinez et al 20] Orthotropic k-nearest Foams for Additive Manufacturing, Jonàs Martínez, Haichuan Song, Jérémie Dumas, Sylvain Lefebvre, ACM TOG 2017

[Liu et al 18] Narrow-band topology optimization on a sparsely populated grid. Liu, H., Hu, Y., Zhu, B., Matusik, W., & Sifakis, E. (2018). ACM Transactions on Graphics (TOG), 37(6), 1-14.

[de Goes et al 20] Discrete Differential Operators on Polygonal Meshes, de Goes, Butts, Desbrun SIGGRAPH / ACM Transactions on Graphics (2020)

[C. et al 21] Digital surface regularization with guarantees, David Coeurjolly, Jacques-Olivier Lachaud, Pierre Gueth, IEEE Transactions on Visualization and Computer Graphics, January 2021

[C. et al 16] Piecewise smooth reconstruction of normal vector field on digital data, David Coeurjolly, Marion Foare, Pierre Gueth, Computer Graphics Forum (Proceeding Pacific Graphics), September 2016

[Caissard et al 19] Laplace–Beltrami Operator on Digital Surfaces, Thomas Caissard, David Coeurjolly, Jacques-Olivier Lachaud, Tristan Roussillon, Journal of Mathema Imaging and Vision, January 2019

[Delanoy et al 19] Combining voxel and normal predictions for multi-view 3D sketching, Johanna Delanoy, David Coeurjolly, Jacques-Olivier Lachaud, Adrien Bousseau Computers and Graphics, June 2019

[Belkin et al 08] Belkin, M., Sun, J., Wang, Y.: Discrete laplace operator on meshed surfaces. In: M. Teillaud (ed.) Proceedings of the 24th ACM Symposium on Computational Geometry, College Park, MD, USA, June 9-11, 2008, pp. 278–287. ACM (2008)

References

[Bertrand94] Bertrand, Gilles. "Simple points, topological numbers and geodesic neighborhoods in cubic grids." *Pattern recognition letters* 15.10 (1994): 1003-1011.

[BC94] Bertrand, Gilles, and Michel Couprie. "On parallel thinning algorithms: minimal non-simple sets, P-simple points and critical kernels." *Journal of Mathematical Imaging and Vision* 35.1 (2009): 23-35.

[YLJ18] Yan, Yajie, David Letscher, and Tao Ju. "Voxel cores: Efficient, robust, and provably good approximation of 3d medial axes." *ACM Transactions on Graphics (TO* 37.4 (2018): 1-13.

[LT16] Lachaud, Jacques-Olivier, and Boris Thibert. "Properties of gauss digitized shapes and digital surface integration." *Journal of Mathematical Imaging and Vision* 54 (2016): 162-180.

[LTC17] Lachaud, Jacques-Olivier, David Coeurjolly, and Jérémy Levallois. "Robust and convergent curvature and normal estimators with digital integral invariants." *Mod Approaches to Discrete Curvature*. Springer, Cham, 2017. 293-348.

[LRTC20] Lachaud, Jacques-Olivier, Pascal Romon, Boris Thibert, and David Coeurjolly. "Interpolated corrected curvature measures for polygonal surfaces." *Computer Graphics Forum*. Vol. 39. No. 5. 2020.