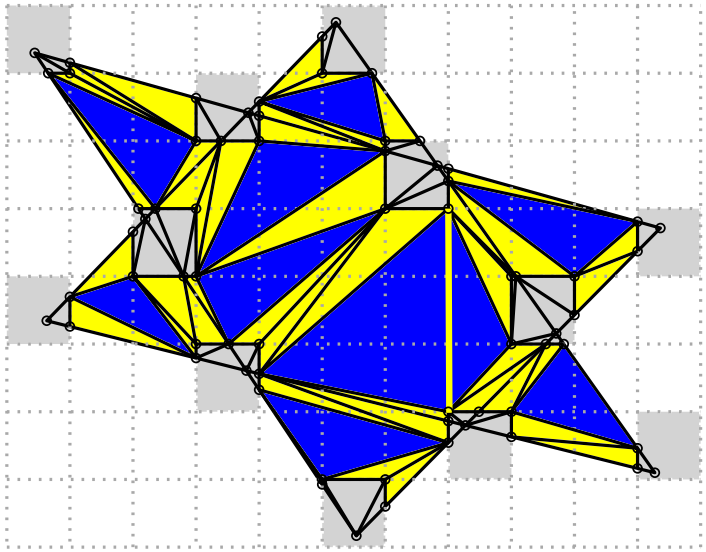
Days of department 1

Rounding polygonal meshes

Léo Valque



Float Number

$$Float = mantis * 2^{exponent}$$

Integer of d digits

Integer of k digits

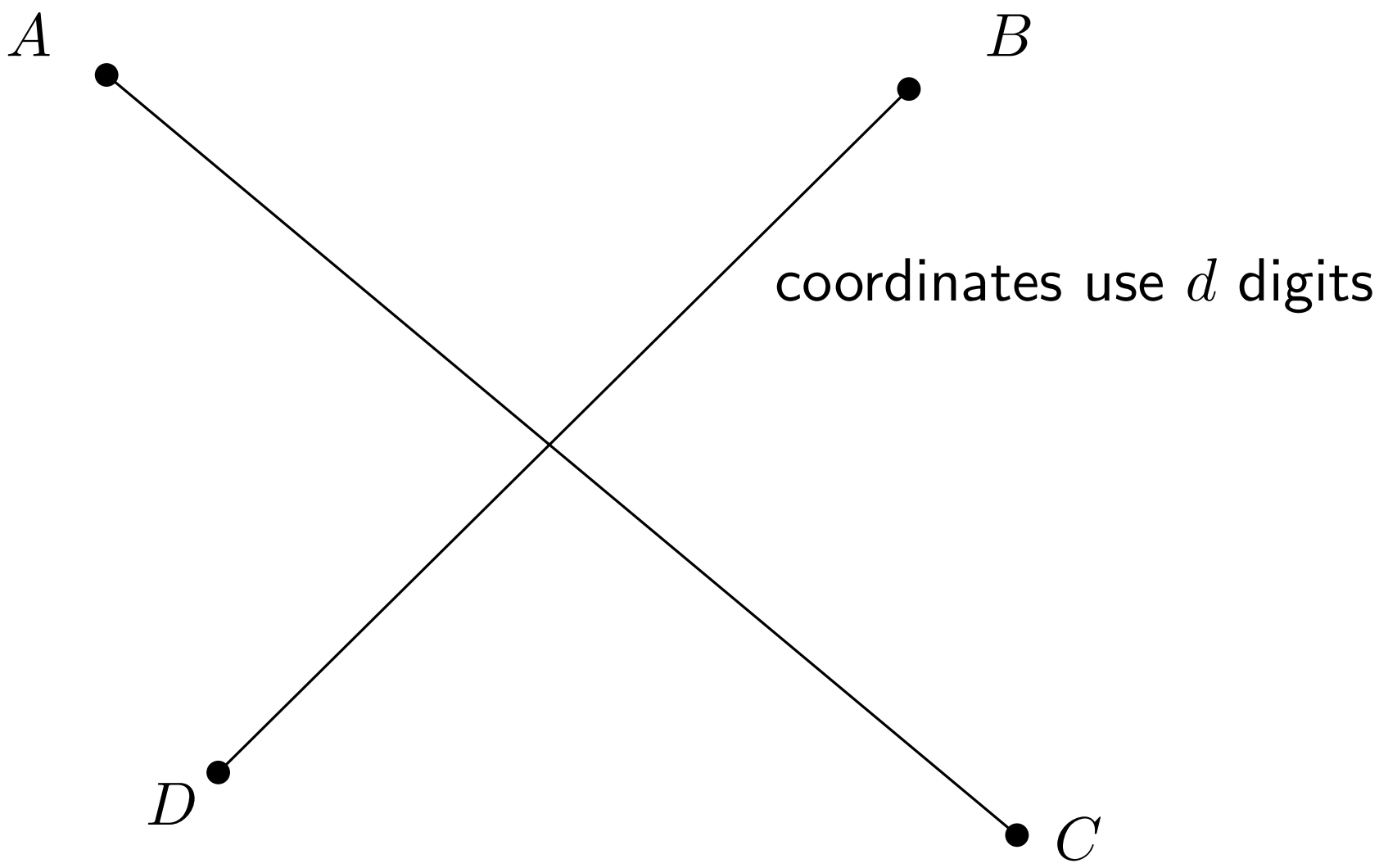
Float Number

d_1 digits d_2 digits

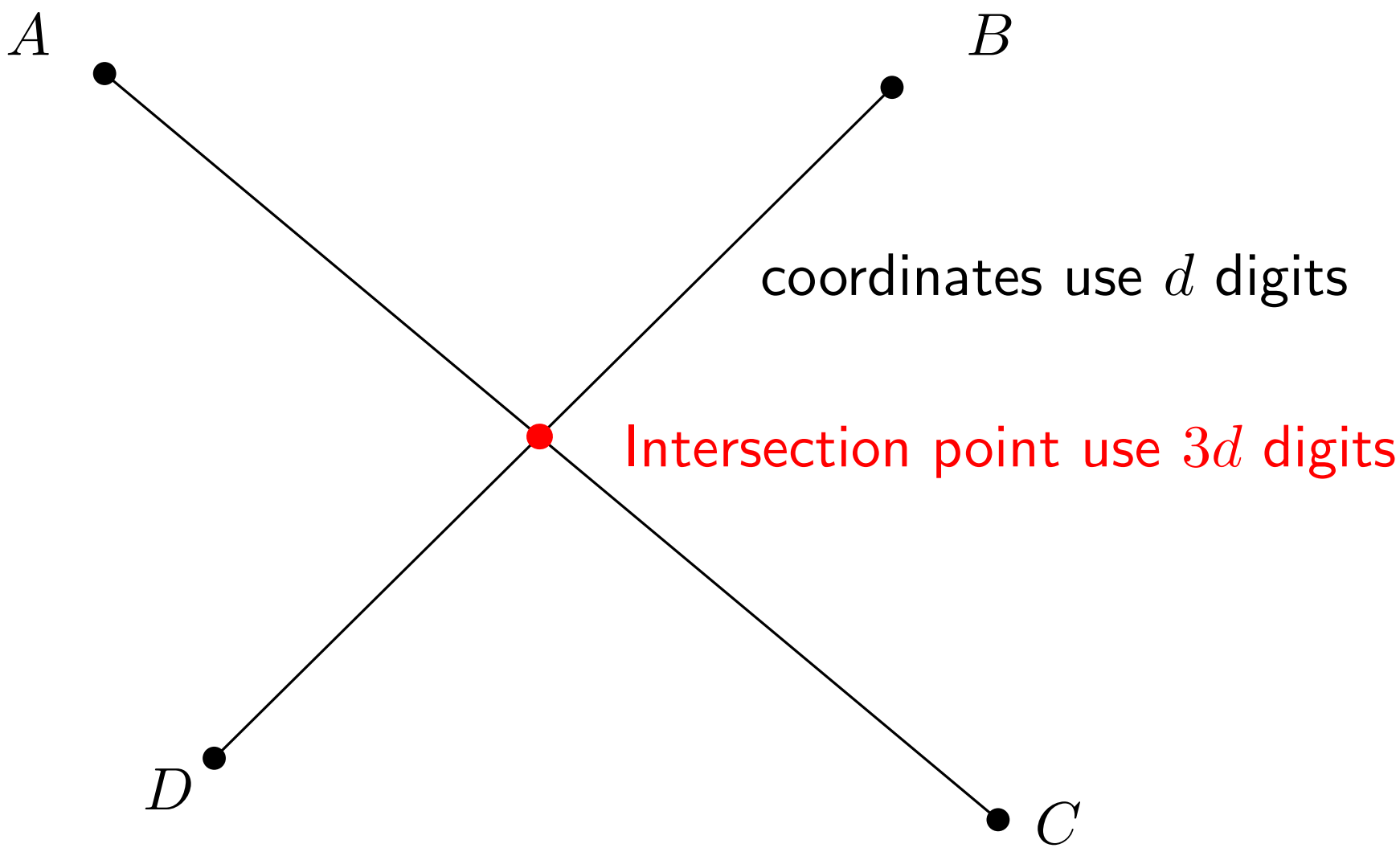
$$f_1 * f_2 = \boxed{\boxed{m_1} * \boxed{m_2}} * 2^{e_1+e_2}$$

$d_1 + d_2$ digits

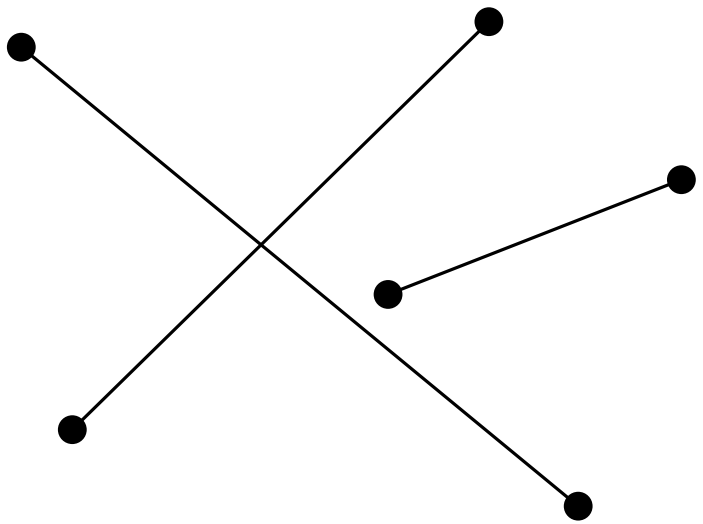
Intersection of Segments



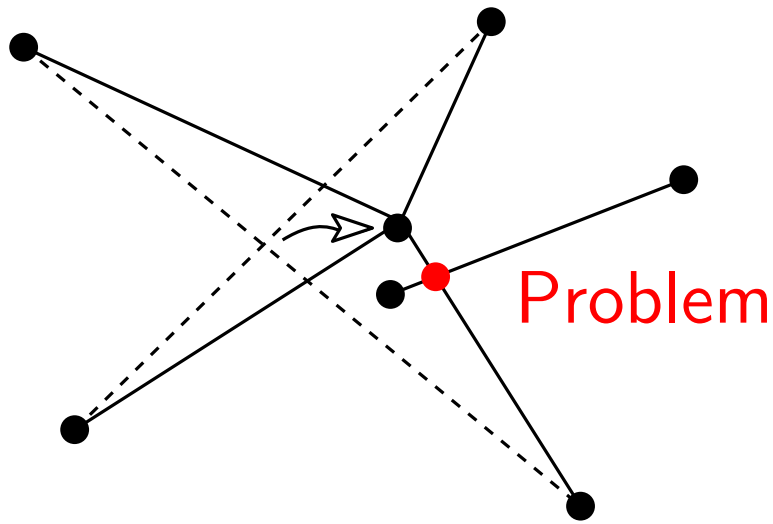
Intersection of Segments



Rounding Issues in 2D

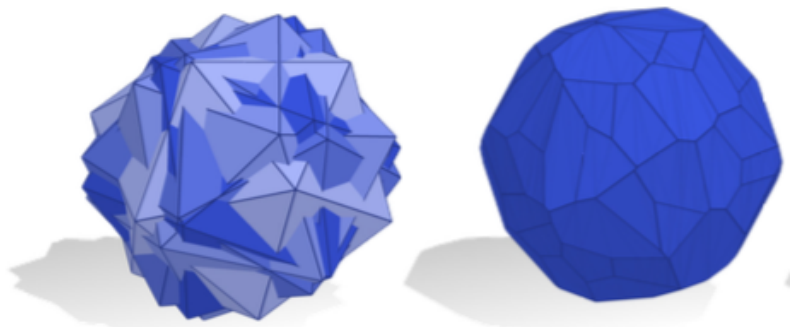


Rounding Issues in 2D



Rounding Issues in 3D

Boolean operations on solids



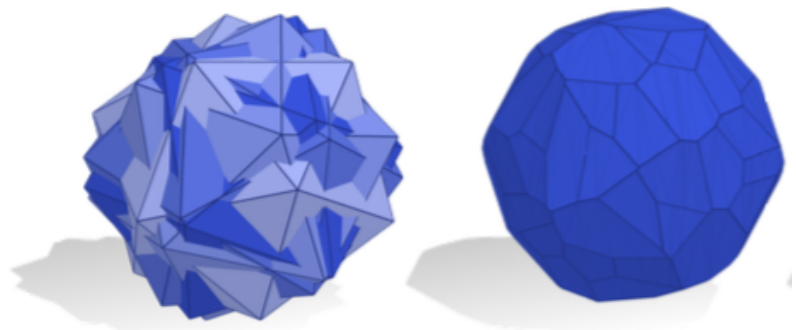
Intersection of four icosahedra

[Zhou et al. 2016]

Need to round the output coordinates

Rounding Issues in 3D

Boolean operations on solids



Intersection of four icosahedra

[Zhou et al. 2016]

Need to round the output coordinates

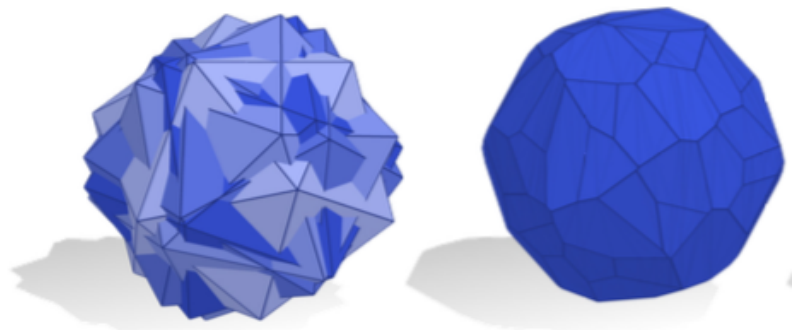
Thingiverse: 10K meshes 45% with self intersections

Mesh repair (elimination of self-intersections)

Zhou et al. [SIGGRAPH 2016] exact arithmetic

Rounding Issues in 3D

Boolean operations on solids



Intersection of four icosahedra

[Zhou et al. 2016]

Need to round the output coordinates

Thingiverse: 10K meshes 45% with self intersections

Mesh repair (elimination of self-intersections)

Zhou et al. [SIGGRAPH 2016] exact arithmetic

Naive rounding fails (self-intersections) in 2.2%

Snap Rounding Problem

Given a set of interior disjoint faces in 3D (or segments in 2D)

Round the vertices coordinates on a grid – Allow subdivisions

Preserve the geometry

Bound the distance between a face and its rounded image

Preserve the topology "up to collapses":

Snap Rounding Problem

Given a set of interior disjoint faces in 3D (or segments in 2D)

Round the vertices coordinates on a grid – Allow subdivisions

Preserve the geometry

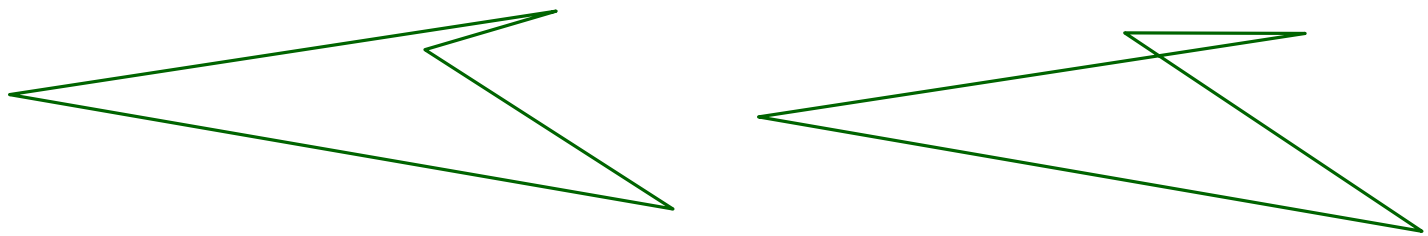
Bound the distance between a face and its rounded image

Preserve the topology "up to collapses":

Subdivisions are needed

NP-hard to round simple polygons with fixed precision, bounded Hausdorff distance, and preserving the topology

[Milenkovic, Nackman, 1990]



Snap Rounding Problem

Given a set of interior disjoint faces in 3D (or segments in 2D)

Round the vertices coordinates on a grid – Allow subdivisions

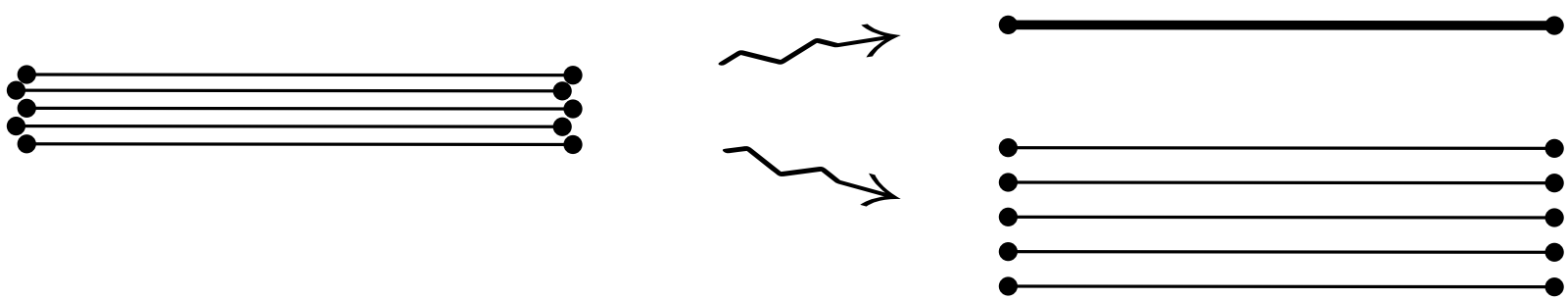
Preserve the geometry

Bound the distance between a face and its rounded image

Preserve the topology "up to collapses":

Subdivisions are needed

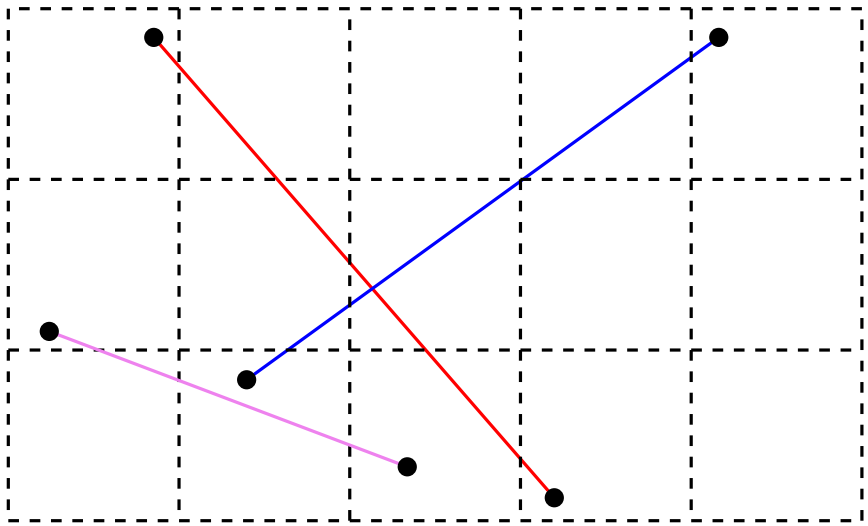
Collapses or unbounded distance may be needed



2D Snap Rounding

Well understood in 2D

1986 Greene, Yao
1997 Goodrich, Guibas, Hershberger, Tanenbaum
1998 Guibas and Marimont
1999 Hobby
2002 Halperin, Packer
2007 de Berg, Halperin, Overmars
2008 Hershberger
2013 Hershberger

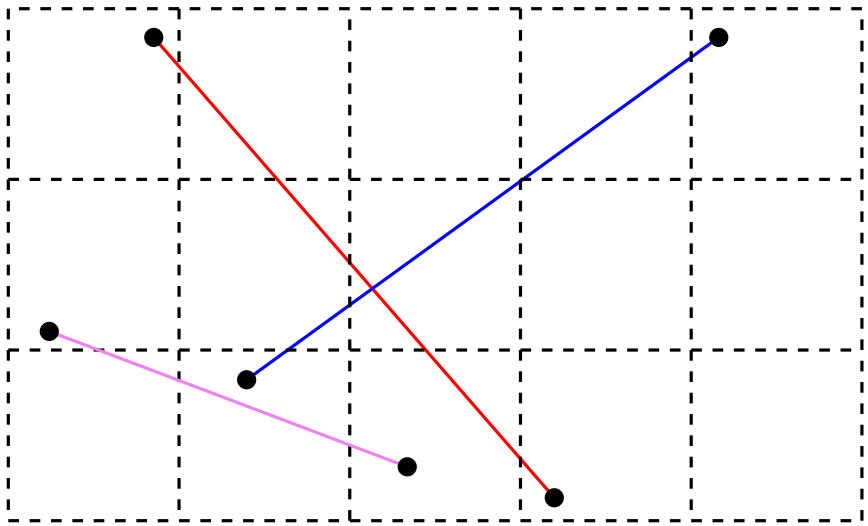


2D Snap Rounding

Well understood in 2D

We round coordinates on the integer grid [Guibas & Marimont 98]

- Pixels that contain a vertex of the arrangement are tagged *hot*
- Segments that intersect a hot pixel are subdivided in that pixel
- All vertices are rounded to their pixels centers

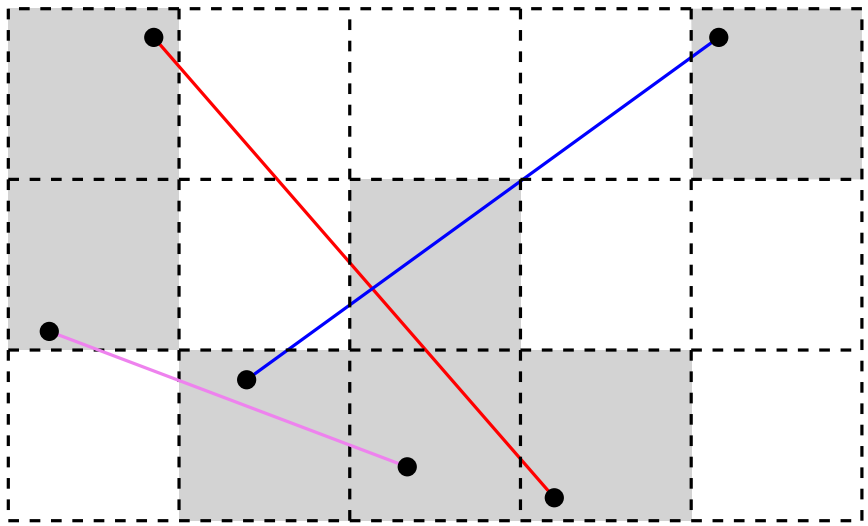


2D Snap Rounding

Well understood in 2D

We round coordinates on the integer grid [Guibas & Marimont 98]

- Pixels that contain a vertex of the arrangement are tagged *hot*
- Segments that intersect a hot pixel are subdivided in that pixel
- All vertices are rounded to to their pixels centers

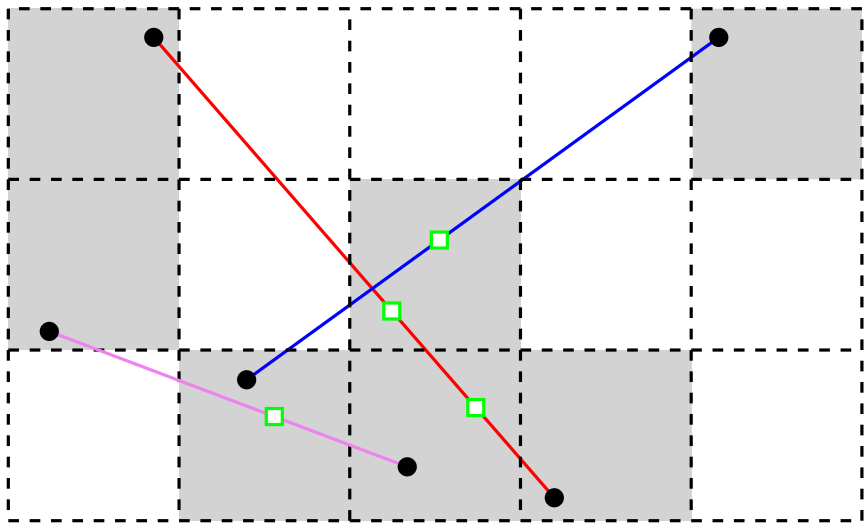


2D Snap Rounding

Well understood in 2D

We round coordinates on the integer grid [Guibas & Marimont 98]

- Pixels that contain a vertex of the arrangement are tagged *hot*
- Segments that intersect a hot pixel are subdivided in that pixel
- All vertices are rounded to to their pixels centers

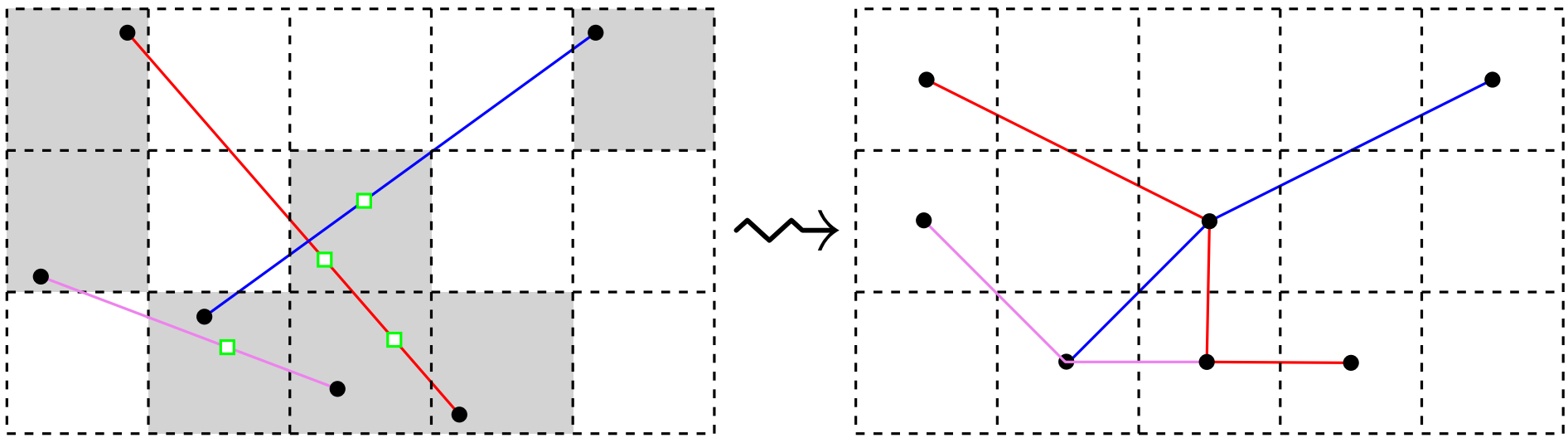


2D Snap Rounding

Well understood in 2D

We round coordinates on the integer grid [Guibas & Marimont 98]

- Pixels that contain a vertex of the arrangement are tagged *hot*
- Segments that intersect a hot pixel are subdivided in that pixel
- All vertices are rounded to their pixels centers



3D Snap Rounding

1990 CCCG, Milenkovic. Face lattices in $\mathbb{R}^d$ Incorrect in 3D

1997 CAGD, Fortune High-level algo for plane-based polyhedra
Does not generalize to vertex-based polyhedra

1999 DCG, Fortune

- The algorithm is very intricate
- Coordinates are rounded to integer multiples of about $1/n$

No satisfying solution

3D Snap Rounding

1990 CCCG, Milenkovic. Face lattices in $\mathbb{R}^d$ Incorrect in 3D

1997 CAGD, Fortune High-level algo for plane-based polyhedra
Does not generalize to vertex-based polyhedra

1999 DCG, Fortune
- The algorithm is very intricate
- Coordinates are rounded to integer multiples of about $1/n$

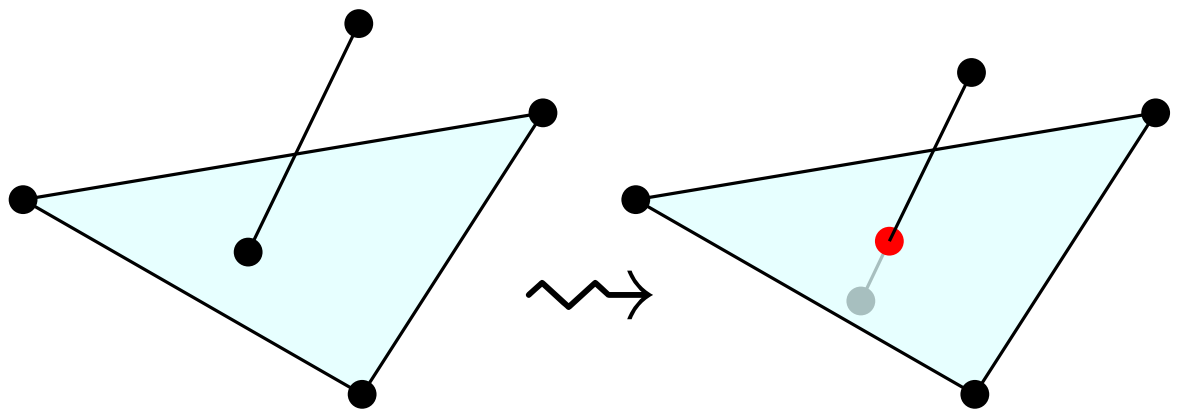
No satisfying solution

2020 DCG, S. Lazard, W. Lenhart, O. Devillers

The difficulty of 3D Snap Rounding

Assume that vertices are rounded to the center of their voxel

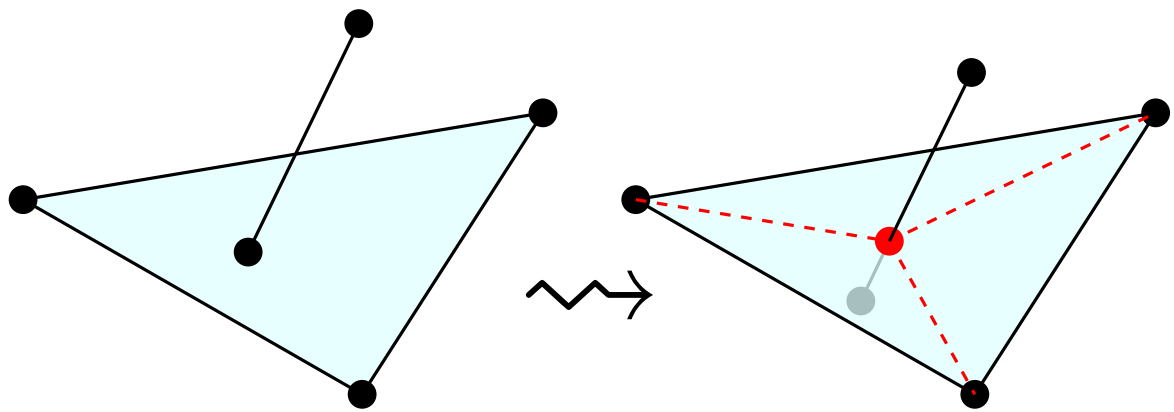
During rounding, a vertex might traverse a face



The difficulty of 3D Snap Rounding

Assume that vertices are rounded to the center of their voxel

During rounding, a vertex might traverse a face



Faces should be retriangulated

The new edges may intersect other edges when rounded

Unknown if such approaches always terminates

The difficulty of 3D Snap Rounding

Second example: Nice but flawed algorithm

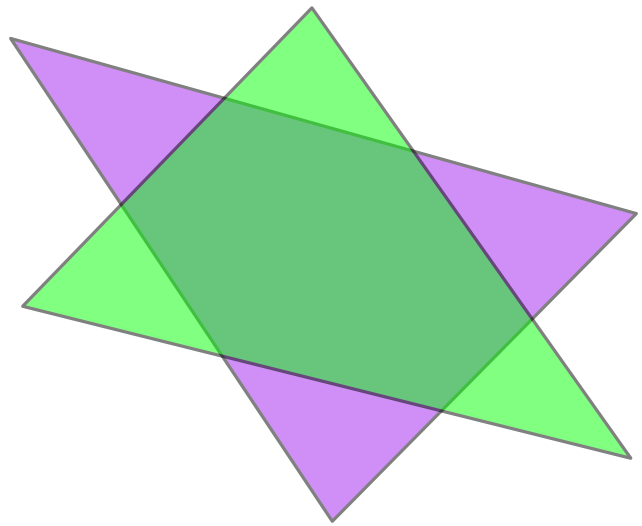
Project all edges on the xy-plane

Subdivide them as in 2D snap rounding

Add the boundaries of hot pixels (to avoid recursion) & Triangulate

Lift this triangulation vertically back on all faces

Round all vertices to the centers of their voxels



The difficulty of 3D Snap Rounding

Second example: Nice but flawed algorithm

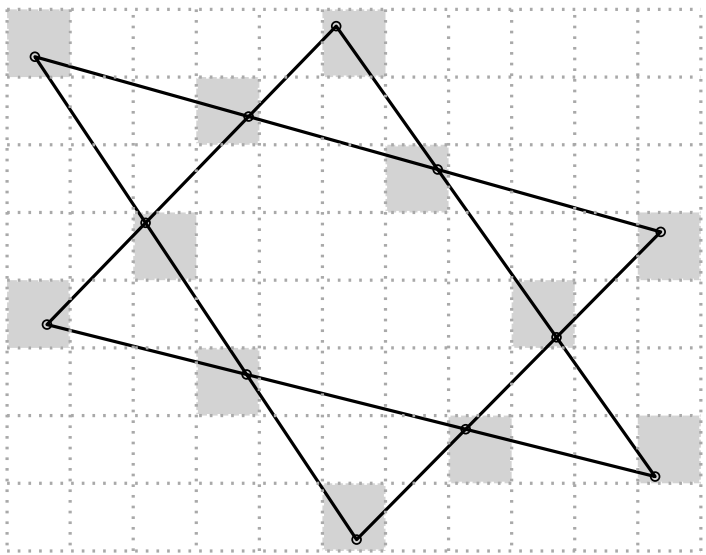
Project all edges on the xy-plane

Subdivide them as in 2D snap rounding

Add the boundaries of hot pixels (to avoid recursion) & Triangulate

Lift this triangulation vertically back on all faces

Round all vertices to the centers of their voxels



The difficulty of 3D Snap Rounding

Second example: Nice but flawed algorithm

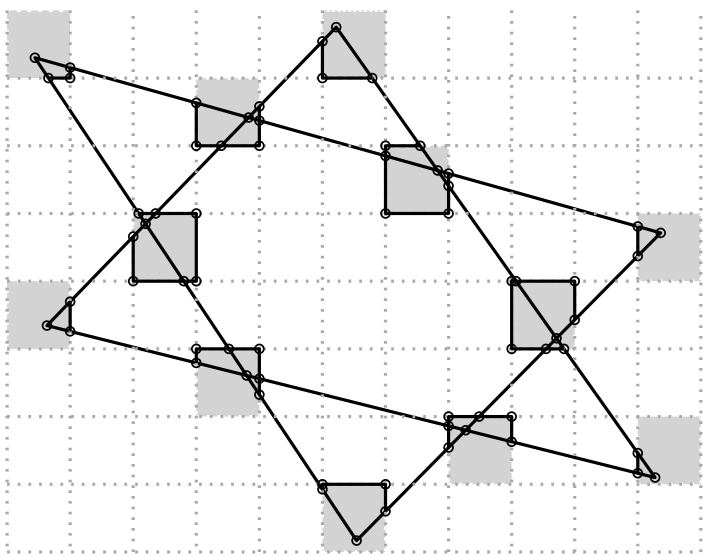
Project all edges on the xy-plane

Subdivide them as in 2D snap rounding

Add the boundaries of hot pixels (to avoid recursion) & Triangulate

Lift this triangulation vertically back on all faces

Round all vertices to the centers of their voxels



Blue: connect 3 pixels
Yellow: connect 2 pixels

Grey: in 1 pixel

The difficulty of 3D Snap Rounding

Second example: Nice but flawed algorithm

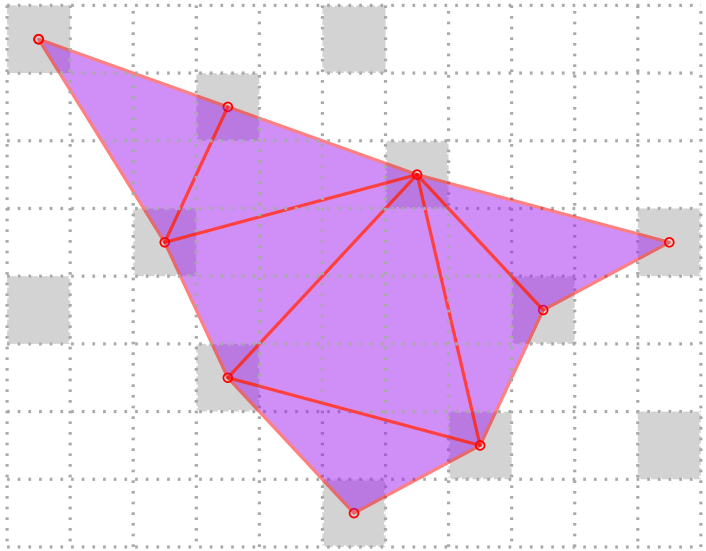
Project all edges on the xy-plane

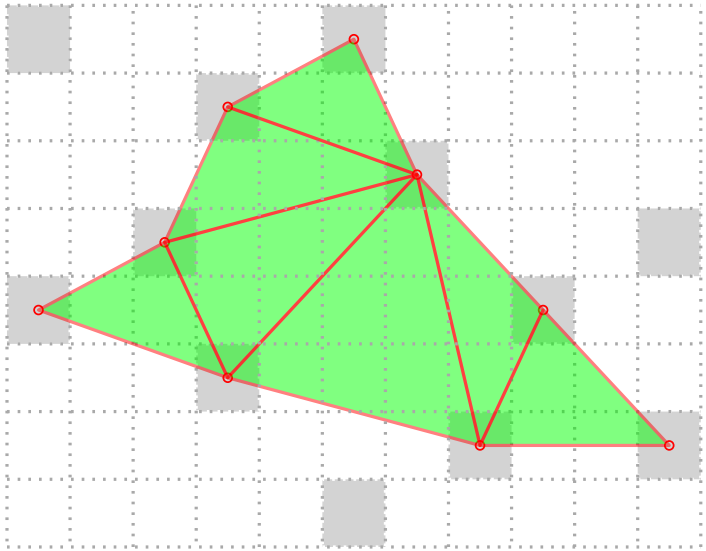
Subdivide them as in 2D snap rounding

Add the boundaries of hot pixels (to avoid recursion) & Triangulate

Lift this triangulation vertically back on all faces

Round all vertices to the centers of their voxels





The difficulty of 3D Snap Rounding

Second example: Nice but flawed algorithm

Project all edges on the xy-plane

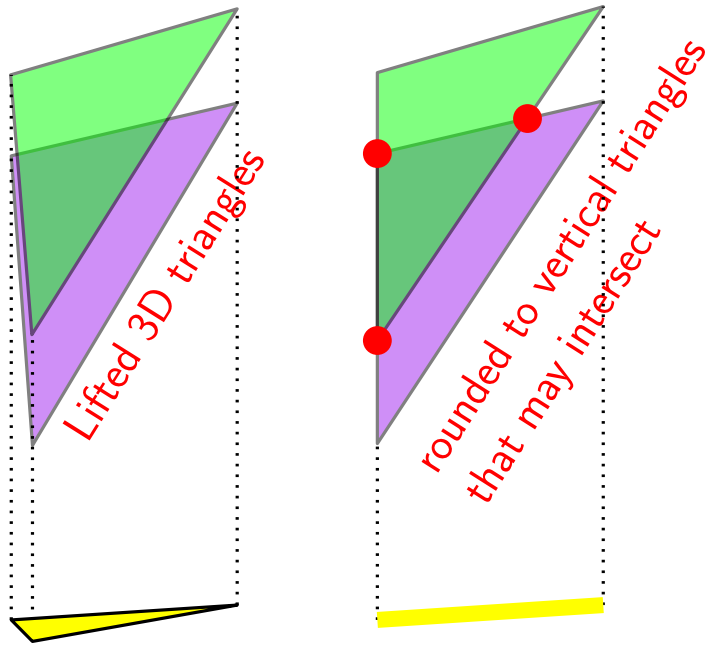
Subdivide them as in 2D snap rounding

Add the boundaries of hot pixels (to avoid recursion) & Triangulate

Lift this triangulation vertically back on all faces

Round all vertices to the centers of their voxels

Flaw:



Yellow triangle rounded to a segment

The difficulty of 3D Snap Rounding

Second example: Nice but flawed algorithm

Project all edges on the xy-plane

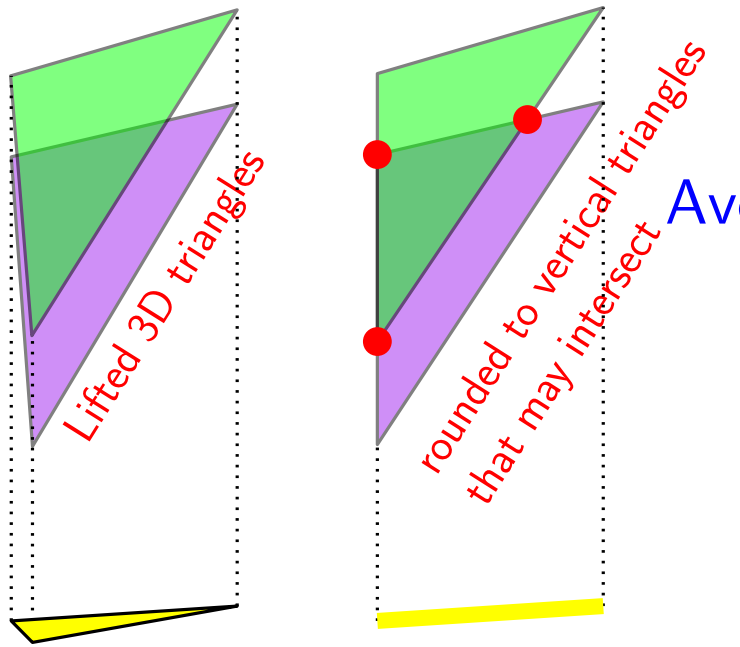
Subdivide them as in 2D snap rounding

Add the boundaries of hot pixels (to avoid recursion) & Triangulate

Lift this triangulation vertically back on all faces

Round all vertices to the centers of their voxels

Flaw:



Yellow triangle rounded to a segment

Fortune's solution [1999]:
Avoid vertical rounding of the faces
by using a finer grid
(integer multiples of $\approx 1/n$)

The difficulty of 3D Snap Rounding

Second example: Nice but flawed algorithm

Project all edges on the xy-plane

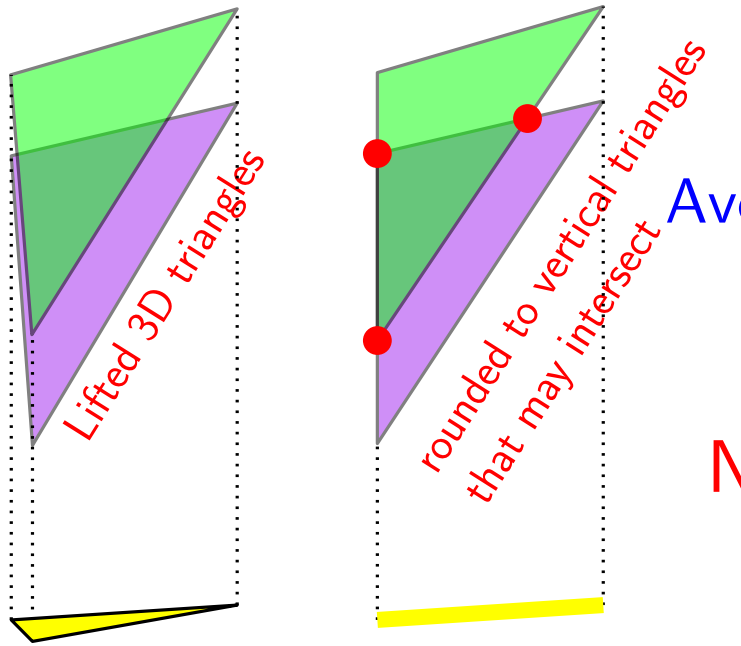
Subdivide them as in 2D snap rounding

Add the boundaries of hot pixels (to avoid recursion) & Triangulate

Lift this triangulation vertically back on all faces

Round all vertices to the centers of their voxels

Flaw:



Yellow triangle rounded to a segment

Fortune's solution [1999]:
Avoid vertical rounding of the faces
by using a finer grid
(integer multiples of $\approx 1/n$)

Nice but flawed algorithm

DCG 2020 Algorithm

Four steps:

1. Collapse the faces that are close to one another
2. Project faces on the floor, subdivide them and lift.
3. Partition the space into xy-parallel slabs
4. Triangulate and round all vertices to their voxels centers

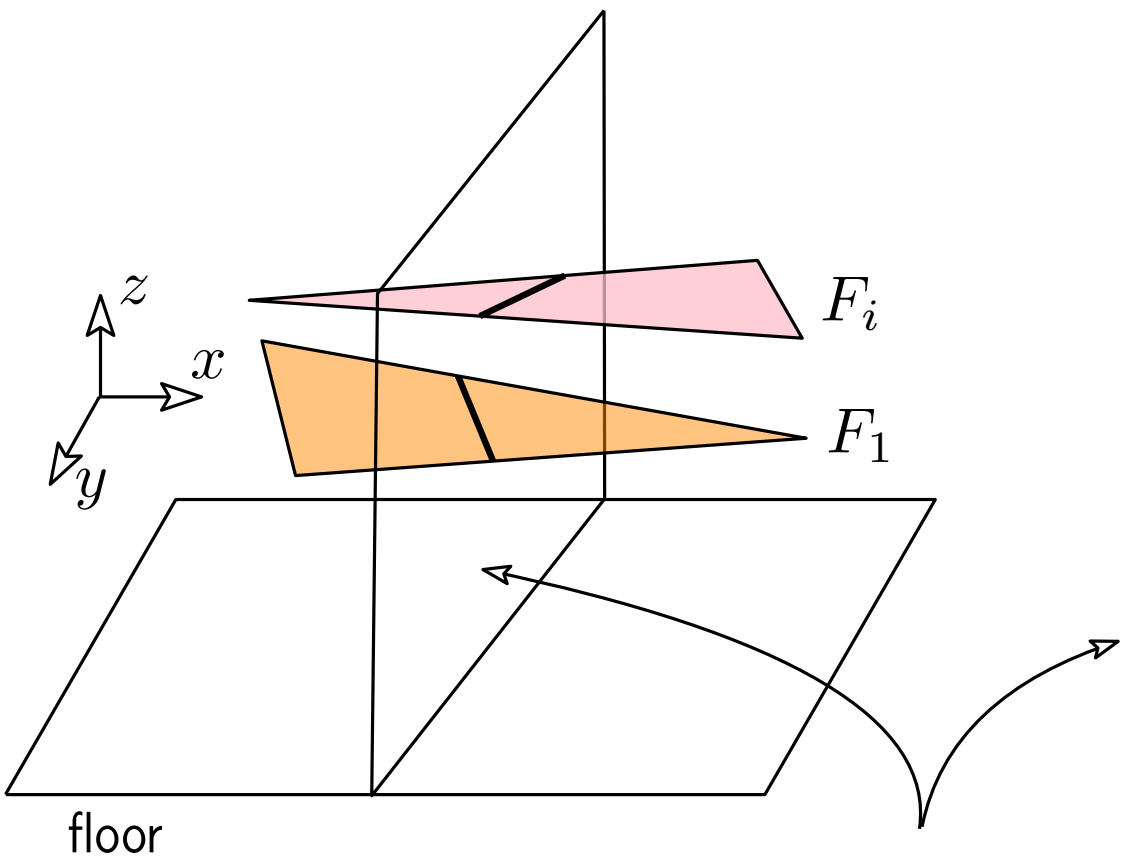
1. Collapse the faces that are close to one another

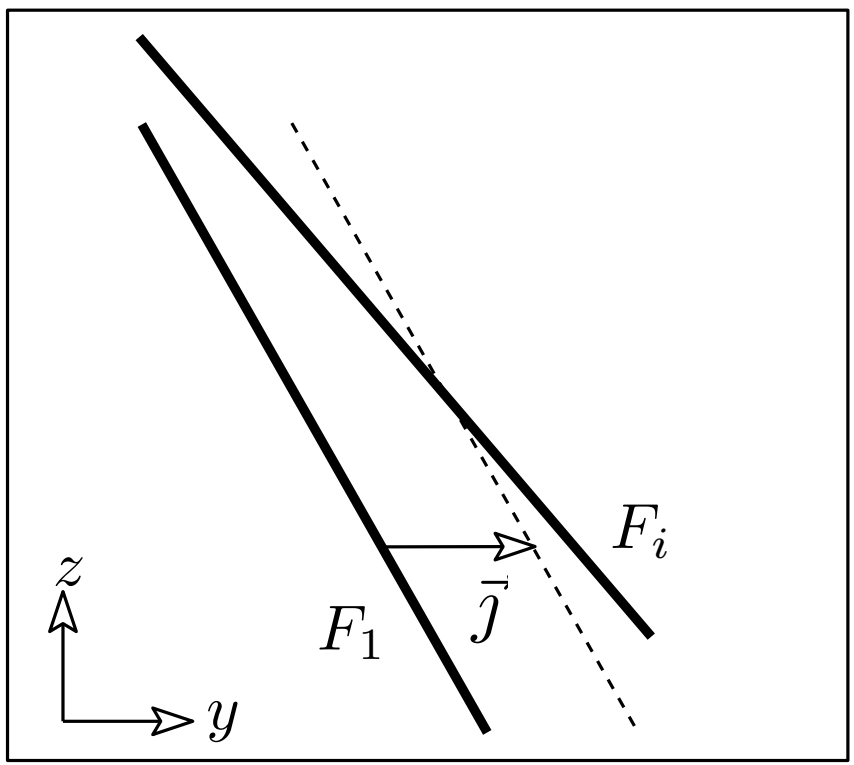
We deform iteratively the input faces ordered arbitrarily from F_1 to F_n

For i from 2 to n

Project the points of F_i along y onto $F_1, \ldots, F_{i-1}$, in order

iff the points project at distance at most 1





any plane $x =$cst

1. Collapse the faces that are close to one another

We deform iteratively the input faces ordered arbitrarily from F_1 to F_n

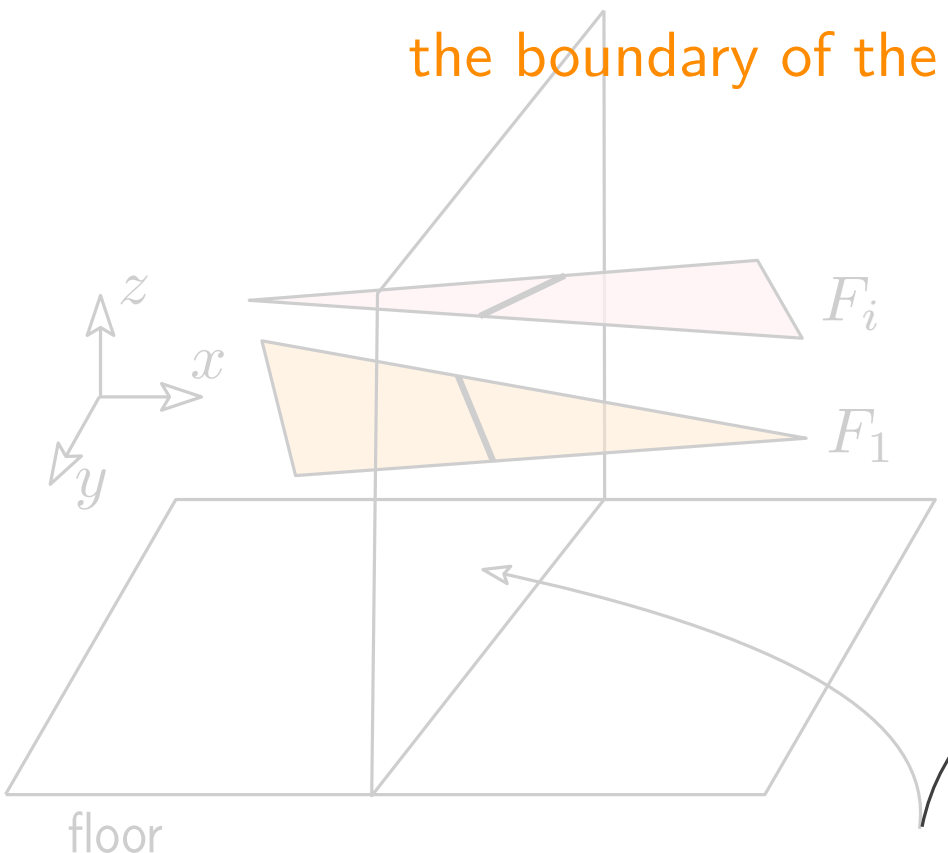
For i from 2 to n

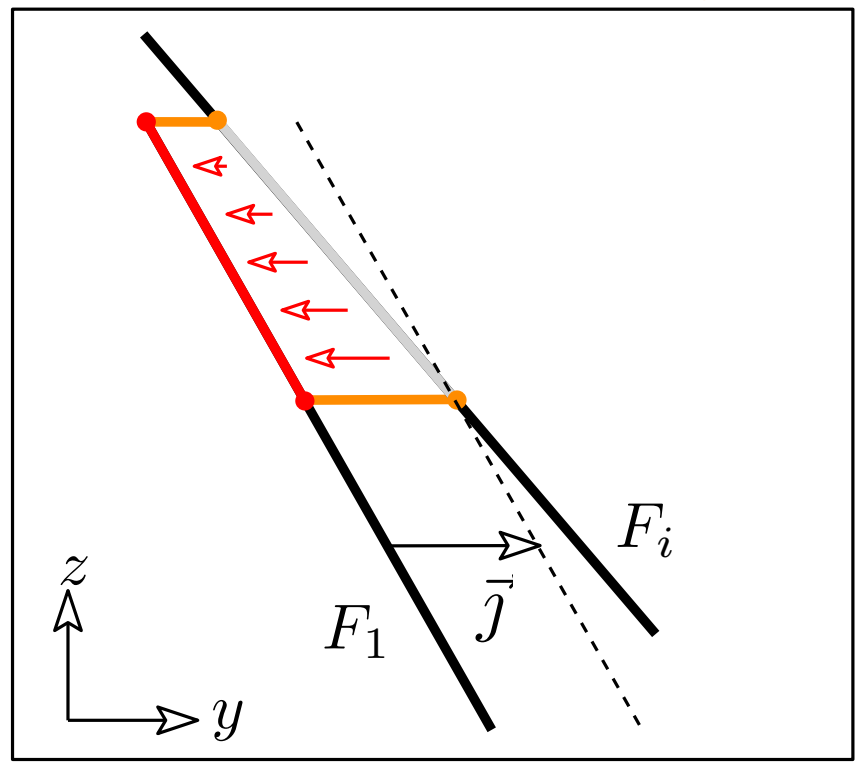
Project the points of F_i along y onto $F_1, \ldots, F_{i-1}$, in order

iff the points project at distance at most 1

Create, if needed, new faces that connect

the boundary of the projected points to their pre-image





any plane $x =$cst

1. Collapse the faces that are close to one another

We deform iteratively the input faces ordered arbitrarily from F_1 to F_n

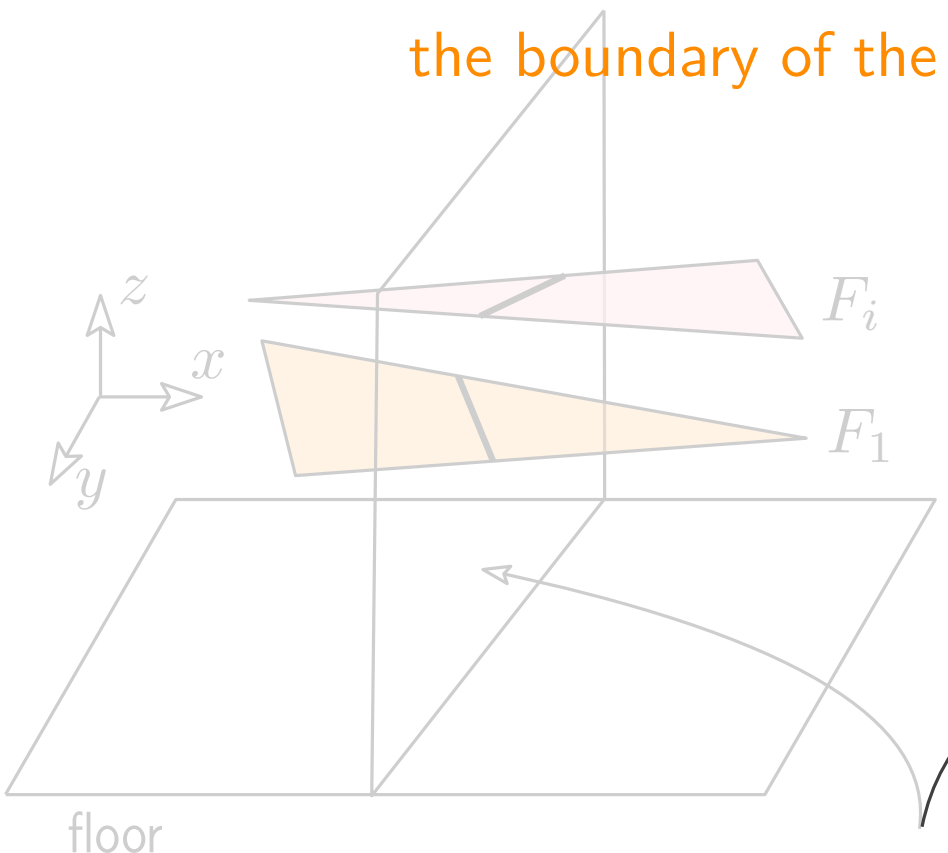
For i from 2 to n

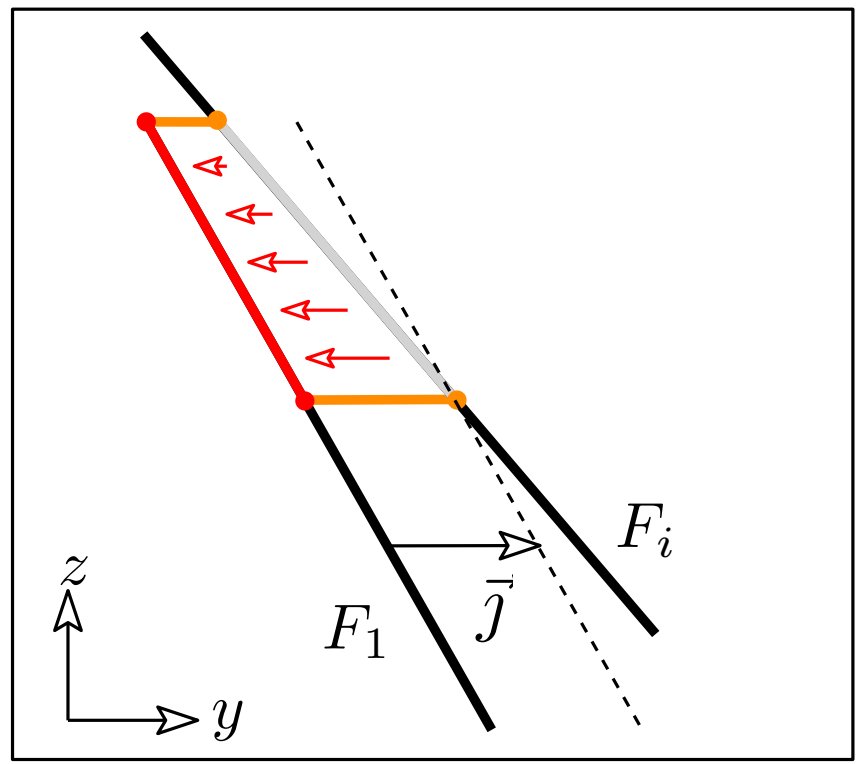
Project the points of F_i along y onto $F_1, \ldots, F_{i-1}$, in order

iff the points project at distance at most 1

Create, if needed, new faces that connect

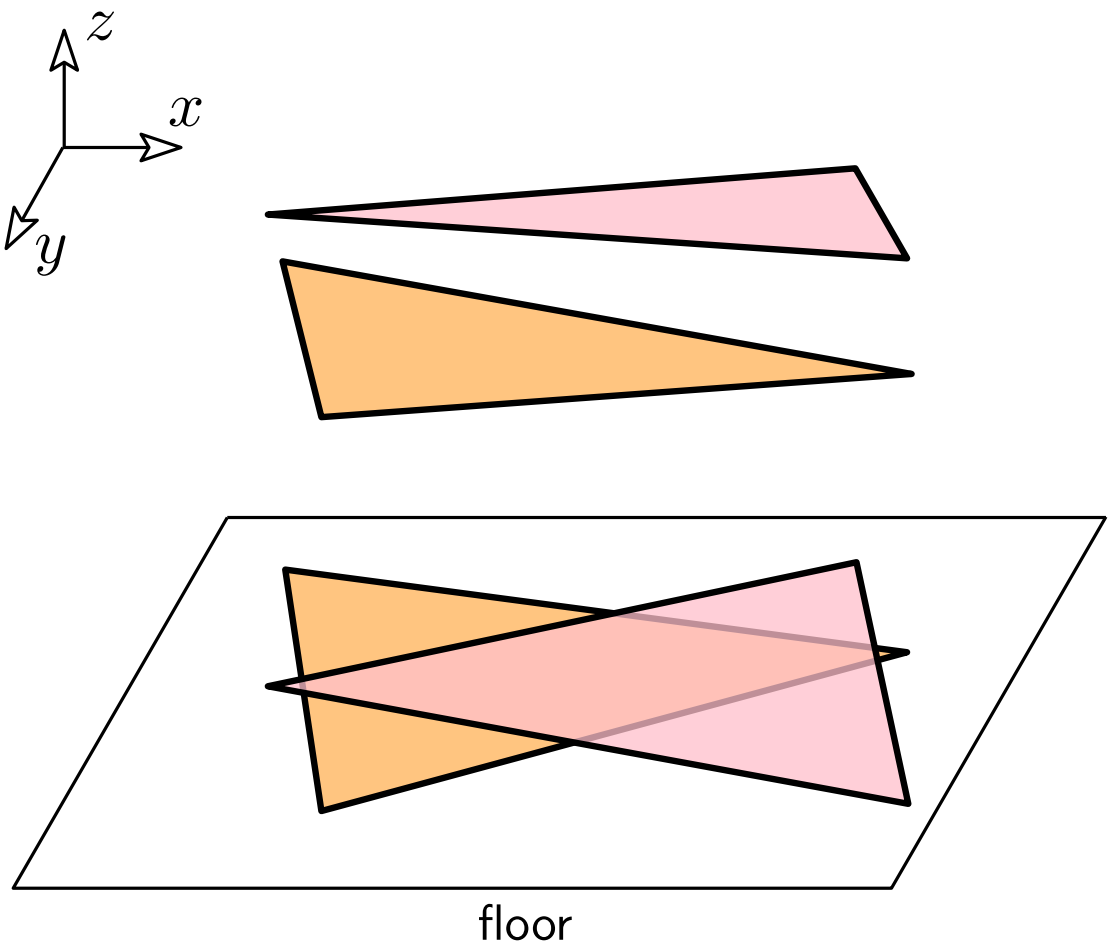
the boundary of the projected points to their pre-image





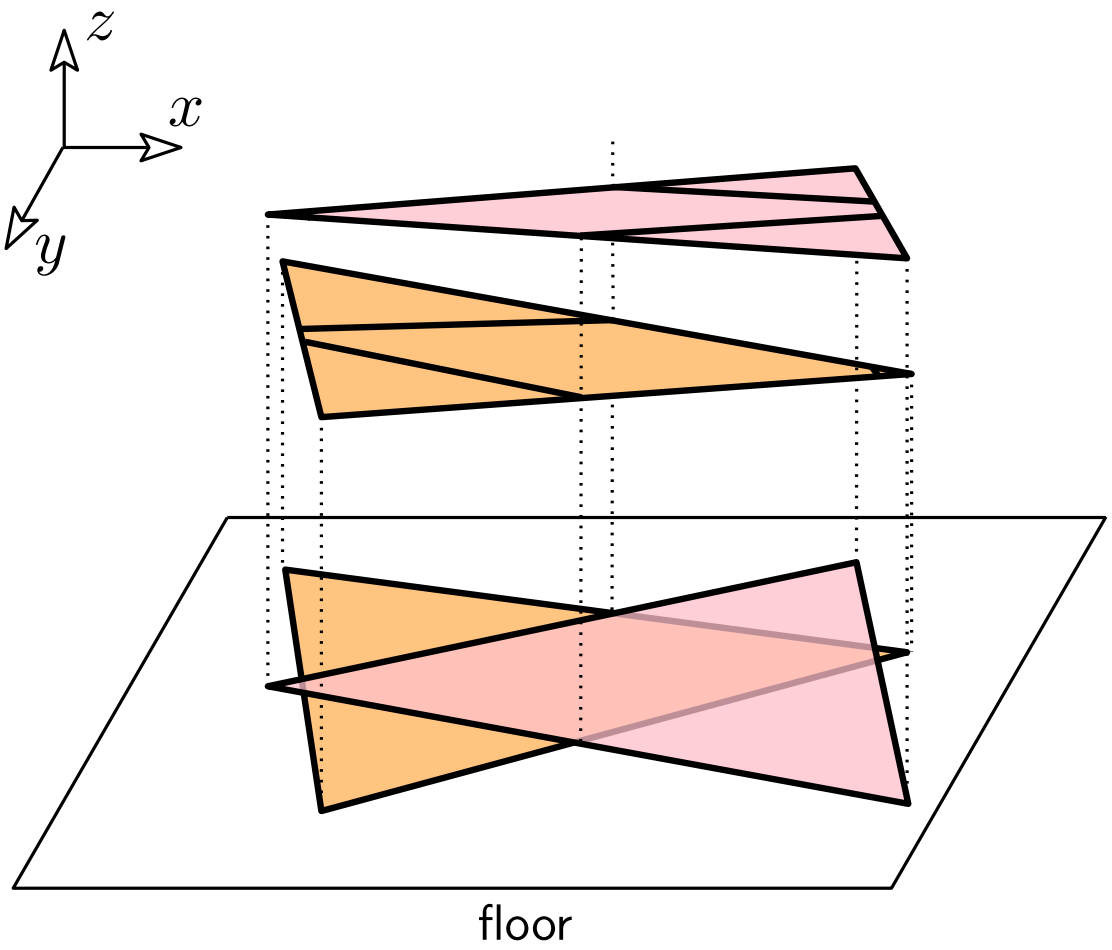
any plane $x =$cst

2. Project faces on the floor, subdivide them and lift.



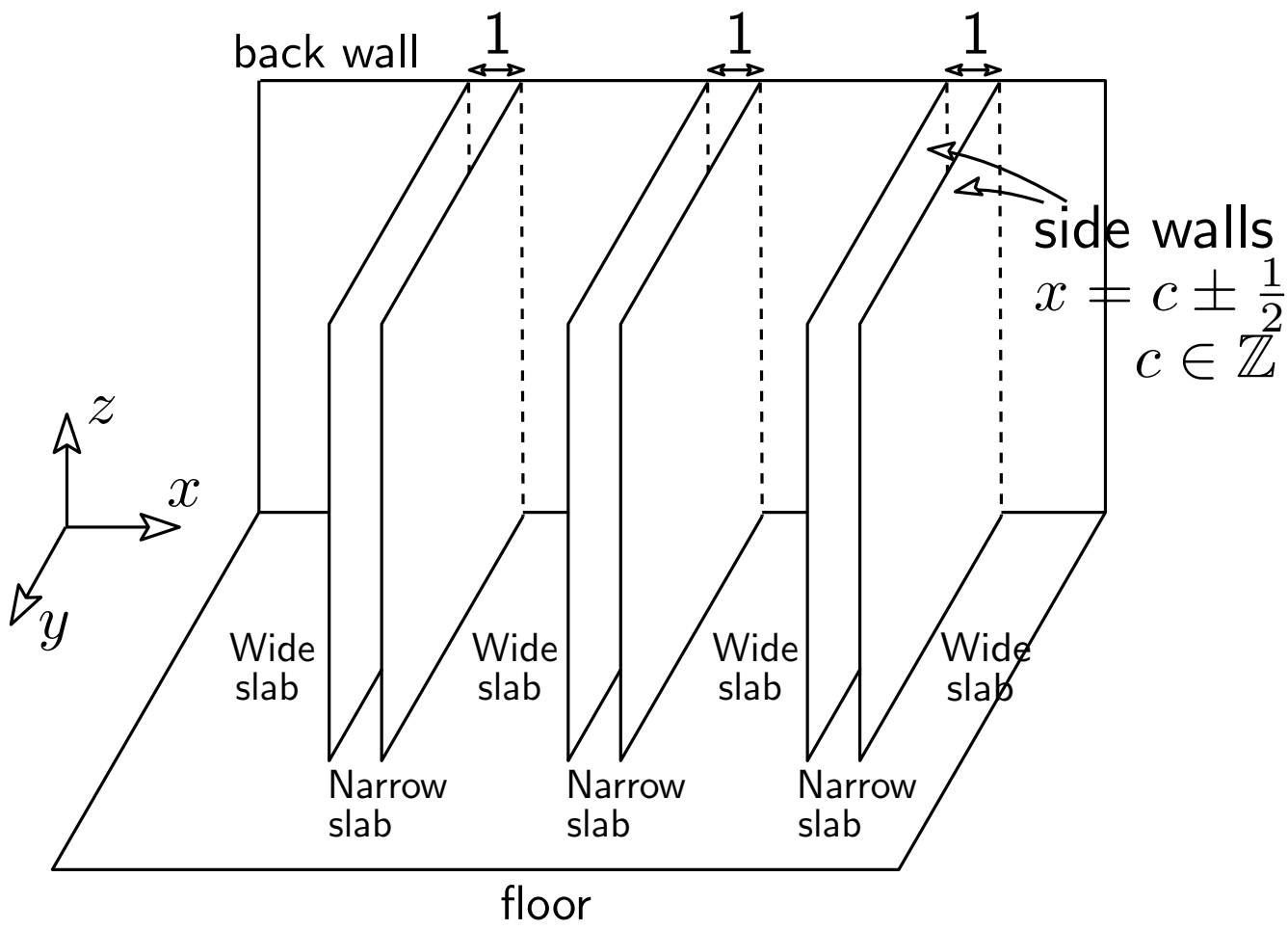
- Project on the floor and compute Arrangement $\mathcal{A}_F$

2. Project faces on the floor, subdivide them and lift.



- Project on the floor and compute Arrangement $\mathcal{A}_F$
- Lift it back on all faces and subdivide them

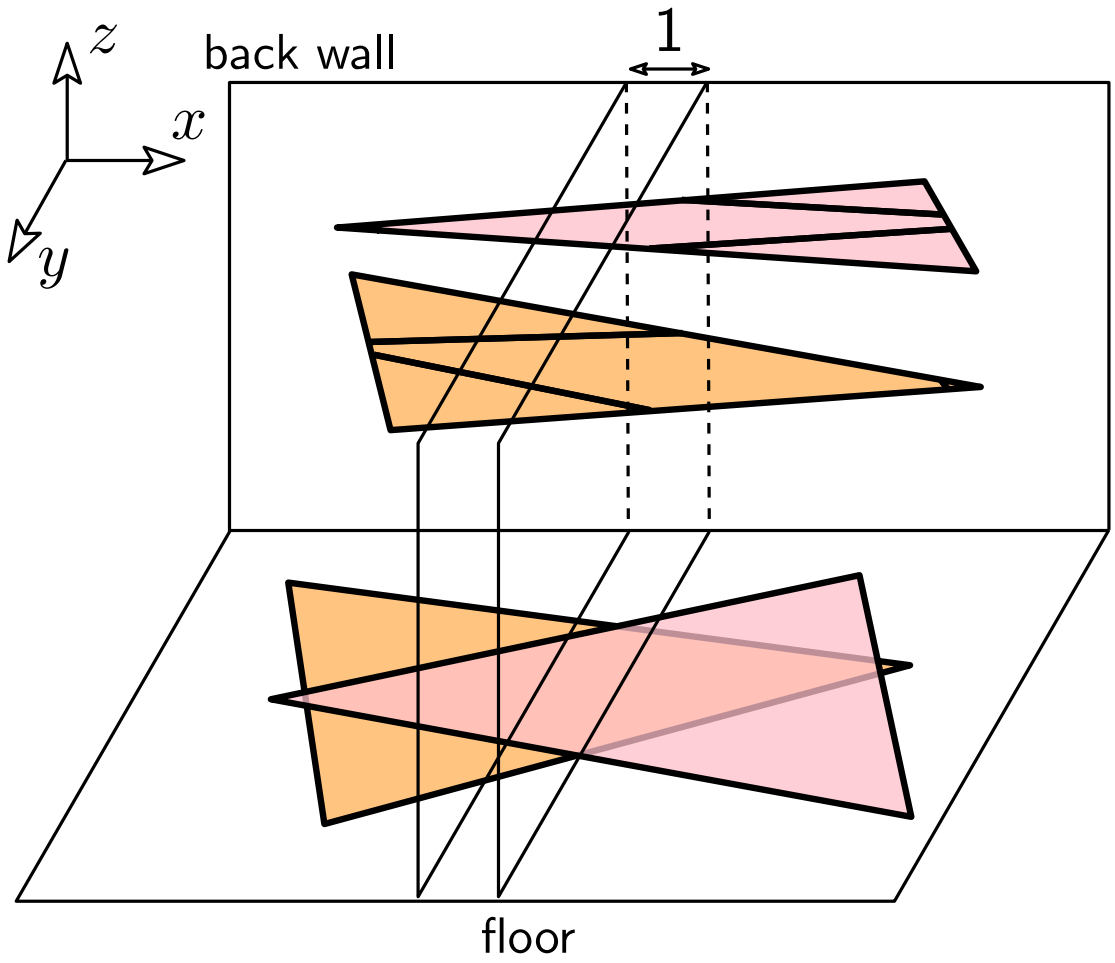
3. Partition the space into xy-parallel slabs



Narrow slab: Region in between two side walls $x = c \pm \frac{1}{2},\ c \in \mathbb{Z}$

Wide slabs: Region bounded by two consecutive thin slabs

3. Partition the space into xy-parallel slabs

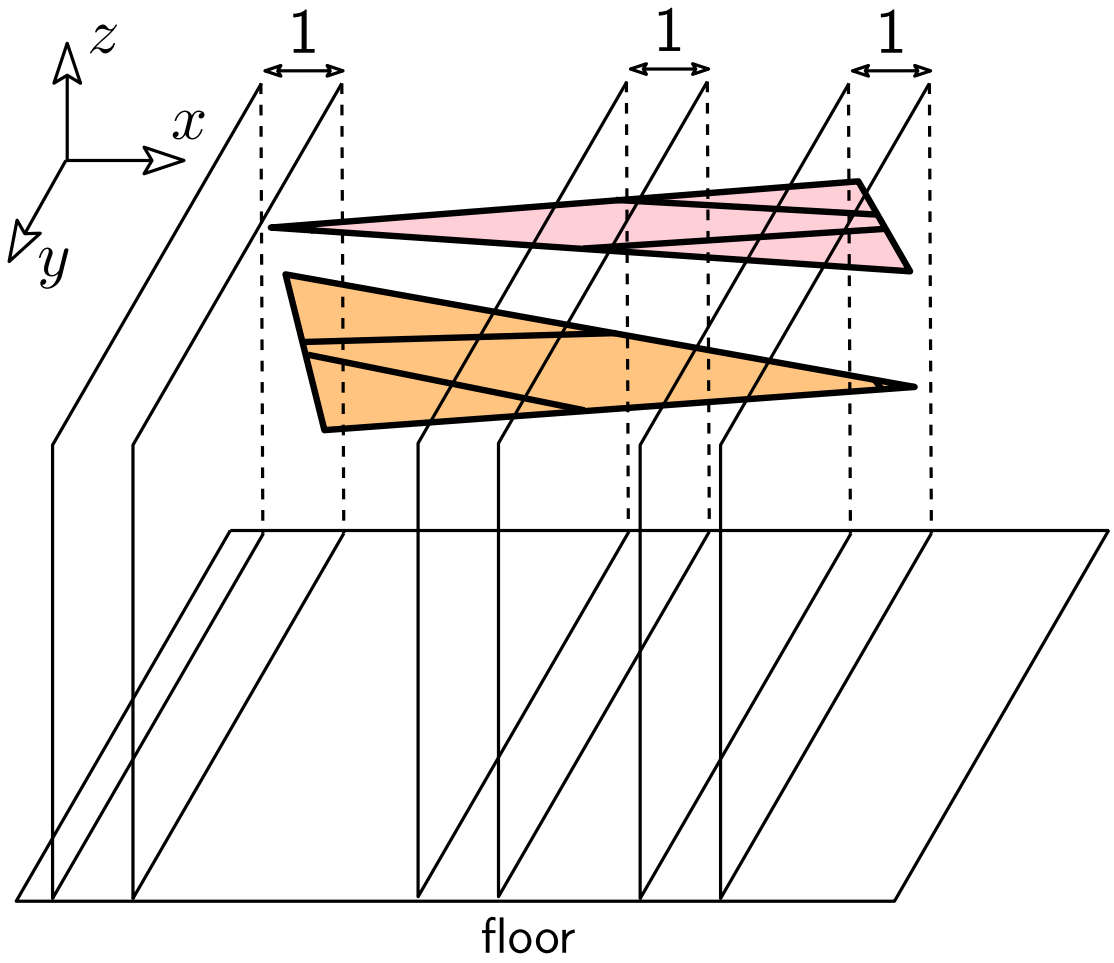


- All vertices define narrow slabs

Narrow slab: Region in between two side walls $x = c \pm \frac{1}{2},\ c \in \mathbb{Z}$

Wide slabs: Region bounded by two consecutive thin slabs

3. Partition the space into xy-parallel slabs

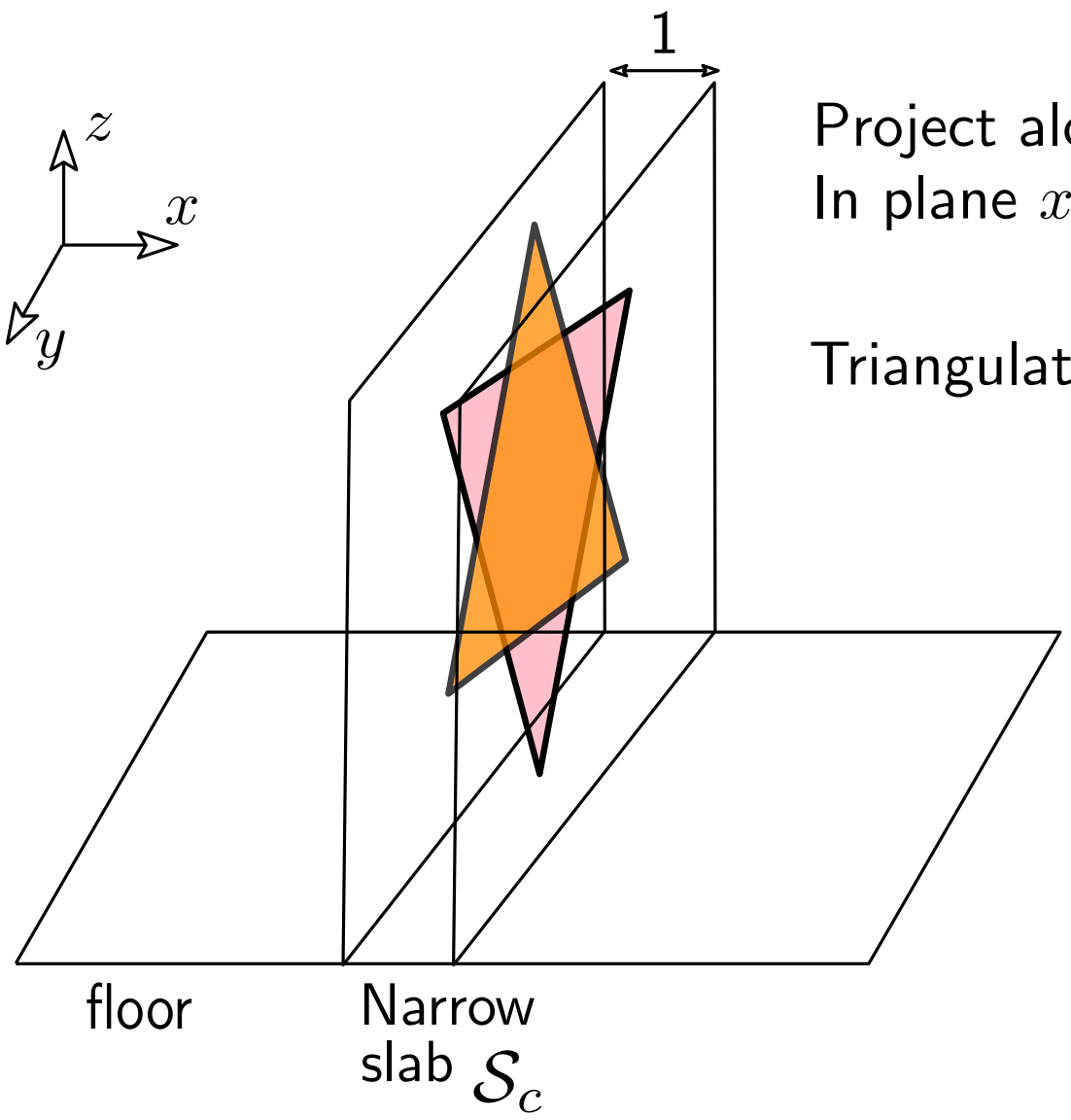


- All vertices define narrow slabs
- Subdivide 3D faces by the side walls of all slabs

Narrow slab: Region in between two side walls $x = c \pm \frac{1}{2},\ c \in \mathbb{Z}$

Wide slabs: Region bounded by two consecutive thin slabs

4.a Triangulate the faces in a narrow slab



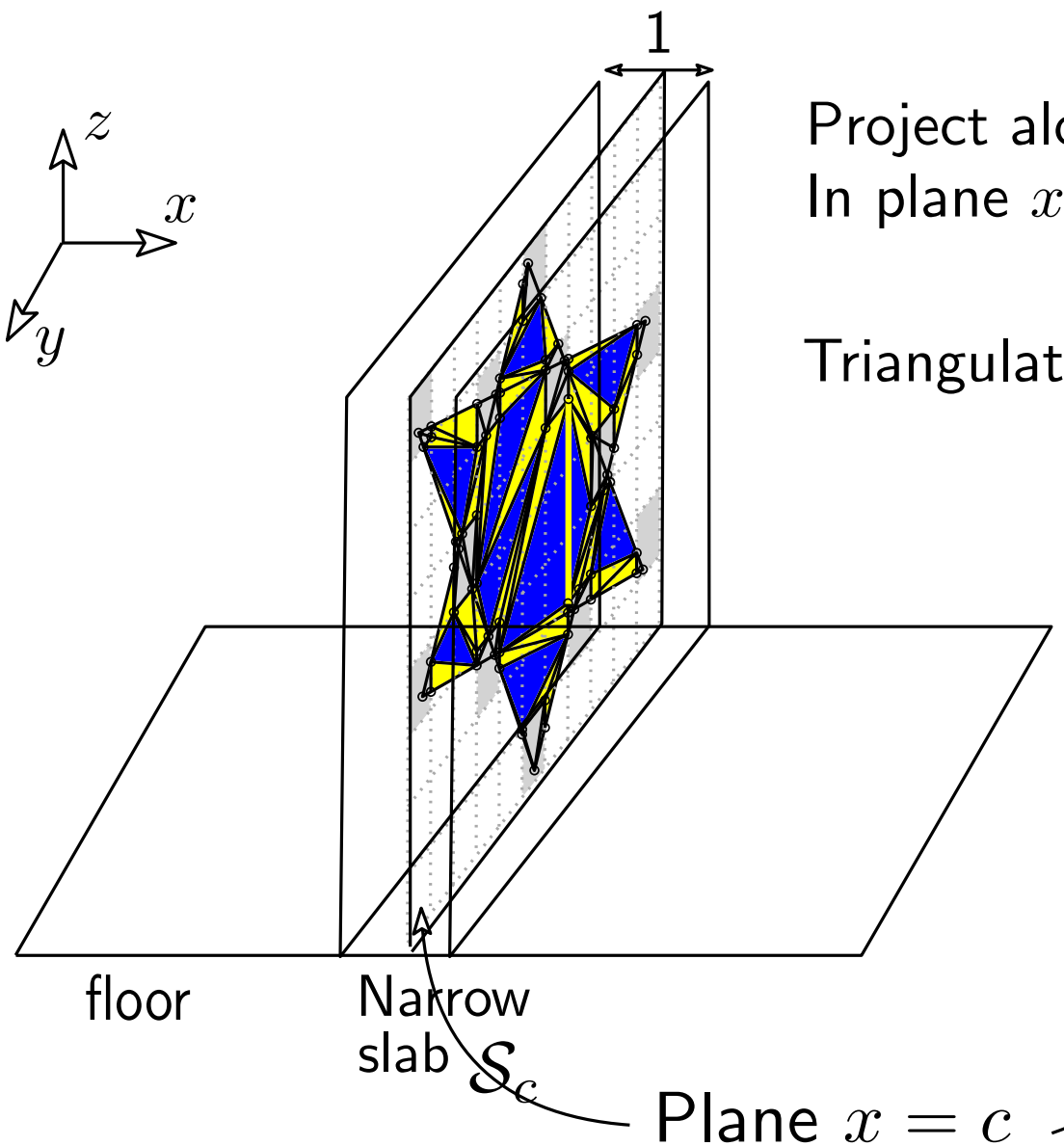
Project along x all faces on $x = c$

In plane $x = c$, subdivide all edges

as in 2D snap rounding

Triangulate and lift back on the faces in $\mathcal{S}_c$

4.a Triangulate the faces in a narrow slab

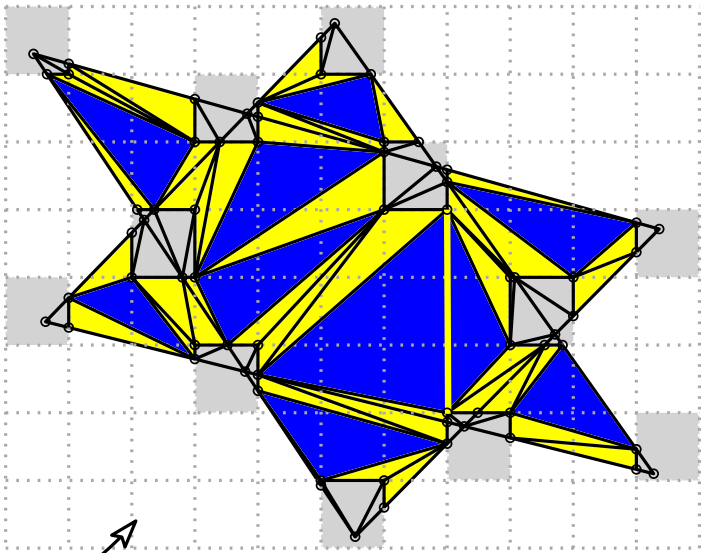


Project along x all faces on $x = c$

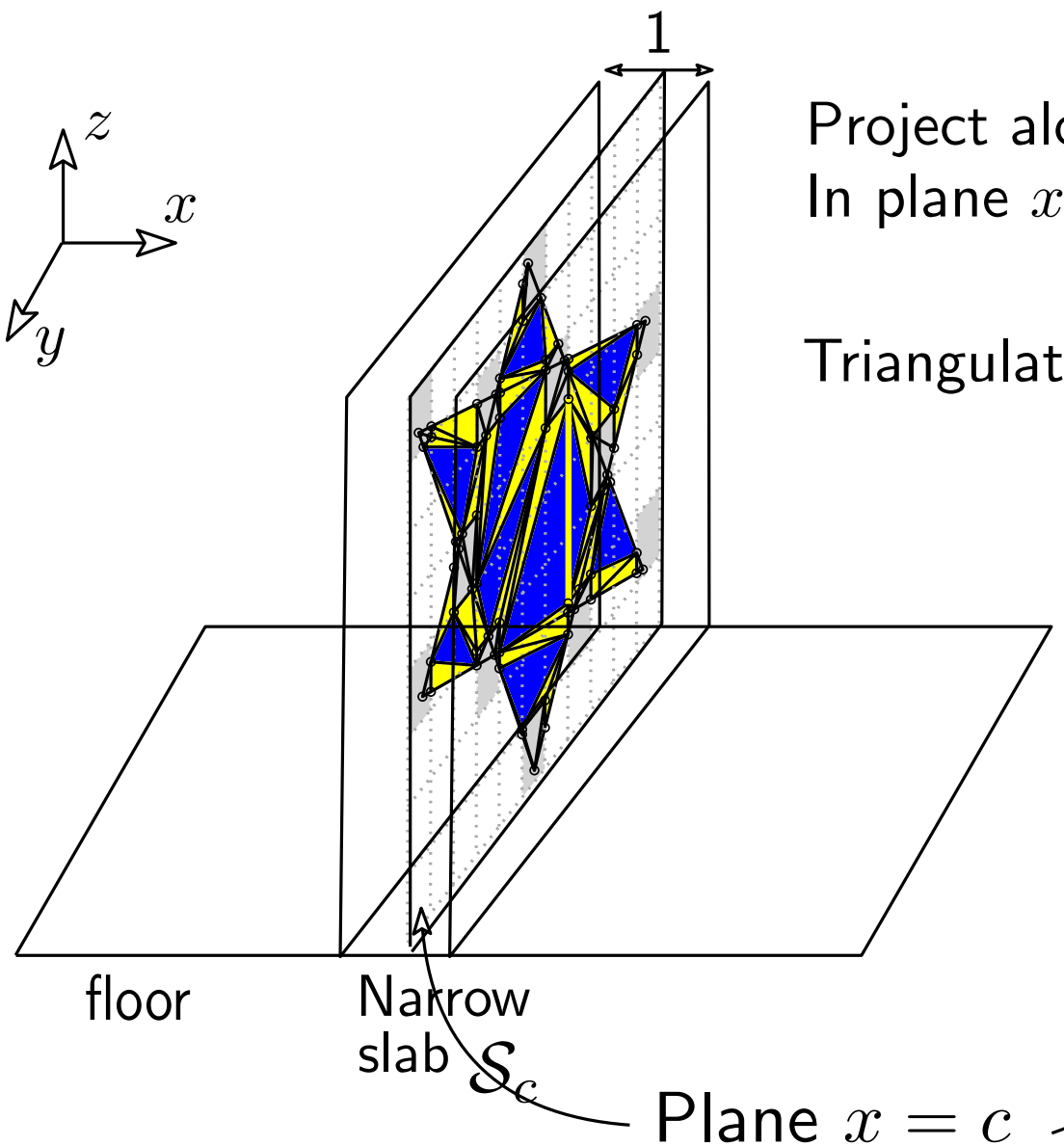
In plane $x = c$, subdivide all edges

as in 2D snap rounding

Triangulate and lift back on the faces in $\mathcal{S}_c$



4.a Triangulate the faces in a narrow slab

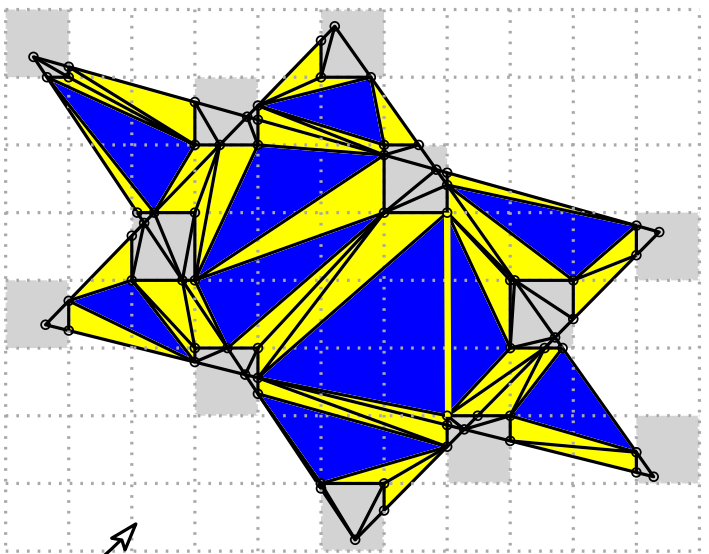


Project along x all faces on $x = c$

In plane $x = c$, subdivide all edges

as in 2D snap rounding

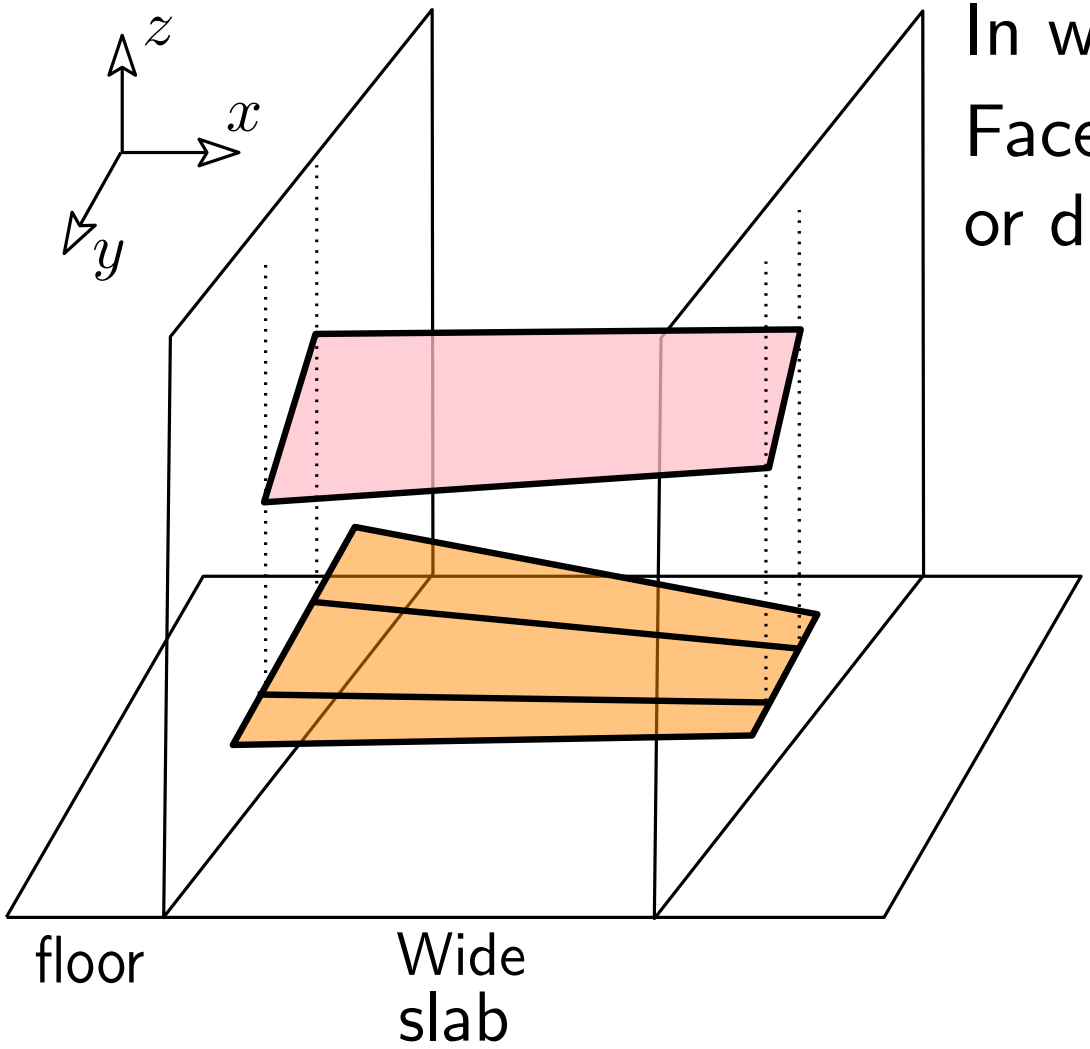
Triangulate and lift back on the faces in $\mathcal{S}_c$



This may create vertices in the side-wall boundaries $x = c \pm \frac{1}{2}$

shared with the adjacent thick slabs

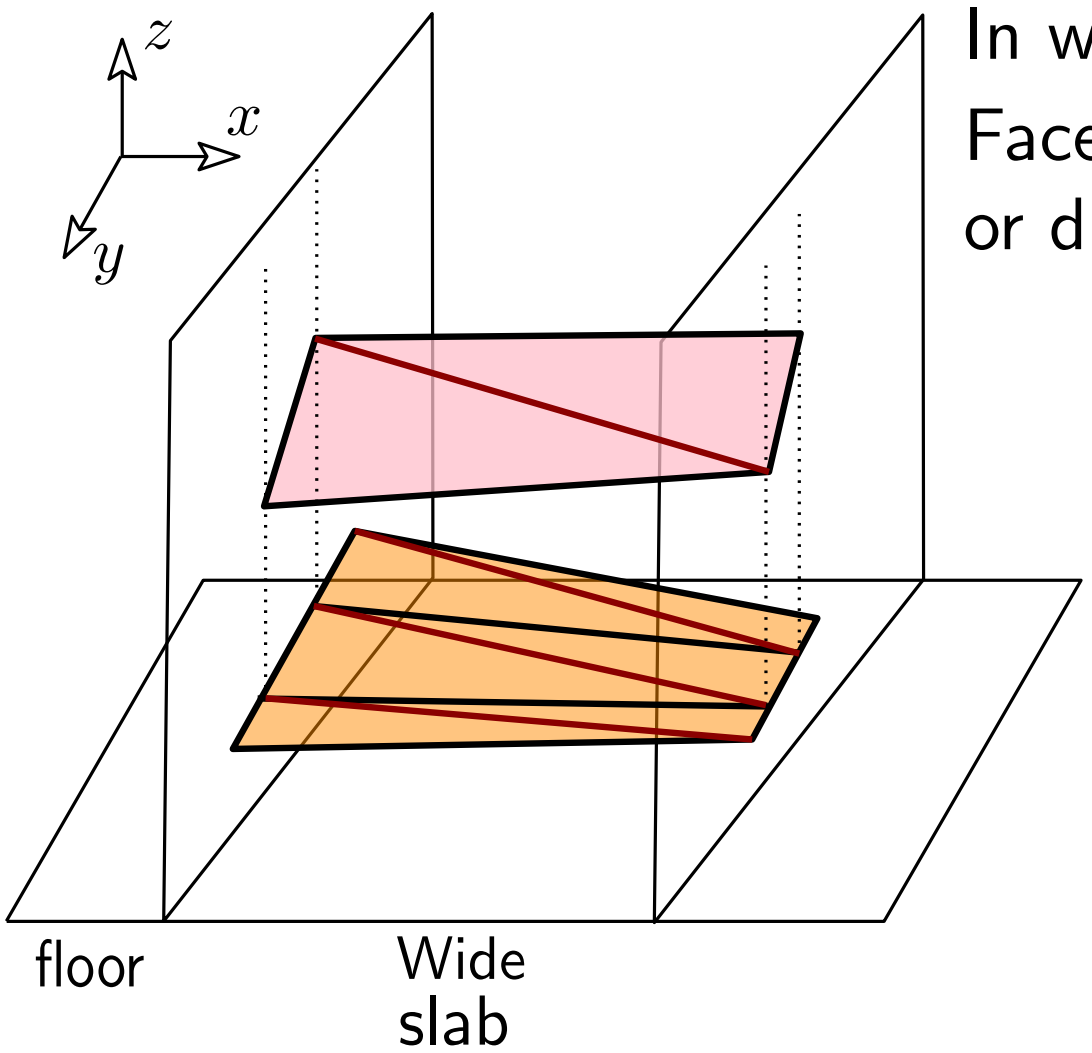
4.b Triangulate the faces in a wide slab



In wide slabs

Faces are trapezoids with identical or disjoint floor projections

4.b Triangulate the faces in a wide slab

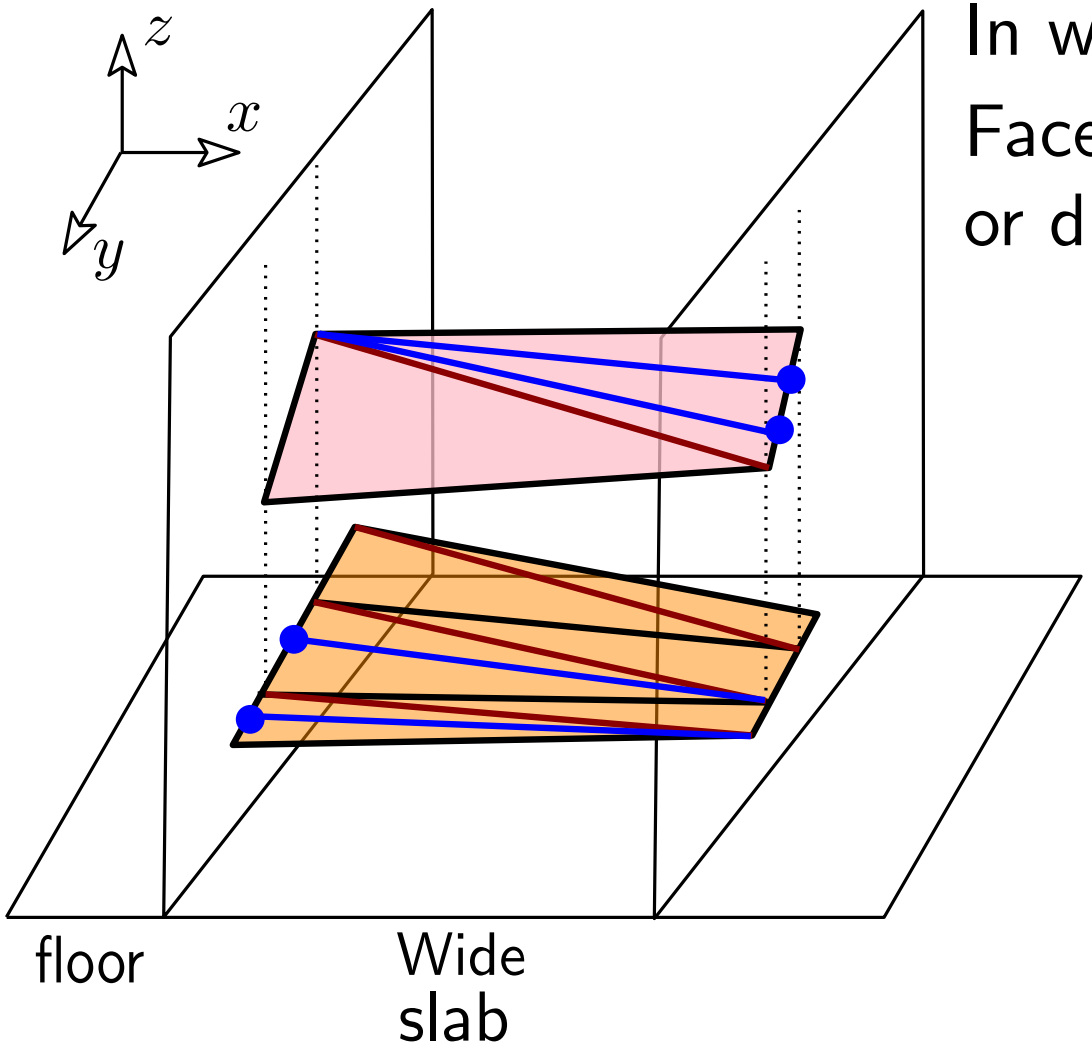


In wide slabs

Faces are trapezoids with identical or disjoint floor projections

- Triangulate with diagonals whose floor projections are consistent

4.b Triangulate the faces in a wide slab

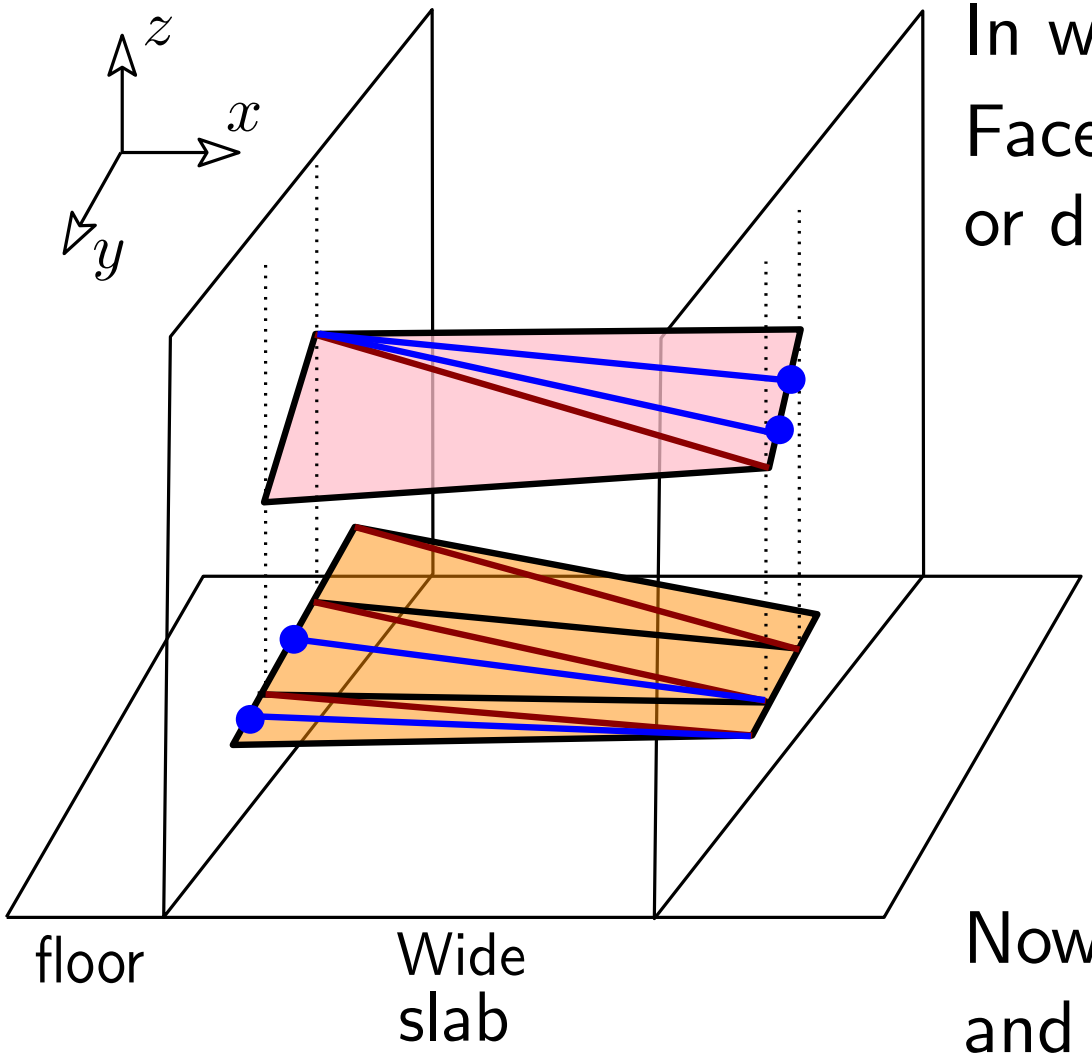


In wide slabs

Faces are trapezoids with identical or disjoint floor projections

- Triangulate with diagonals whose floor projections are consistent
- Further triangulate with the vertices just created in the adjacent narrow slabs

4.b Triangulate the faces in a wide slab



In wide slabs

Faces are trapezoids with identical or disjoint floor projections

- Triangulate with diagonals whose floor projections are consistent
- Further triangulate with the vertices just created in the adjacent narrow slabs

Now, we can round all vertices and no intersection occur

DCG 2020 Algorithm

Four steps:

1. Collapse the faces that are close to one another
2. Project faces on the floor, subdivide them and lift.
3. Partition the space into xy-parallel slabs
4. Triangulate and round all vertices to their voxels centers

Rather simple algorithm Delicate proof of correctness

Preserve the geometry L_∞ Hausdorff distance $\leqslant 3/2$

Preserve the topology "up to collapses"

Bad worst-case complexity: $O(n^{15})$ space and $O(n^{19})$ time

$O(n^5)$ space and $O(n^{6.5})$ time under some hypotheses

Conclusion and current works

First algorithm for snap rounding sets of polygons in 3D on a fixed-size grid

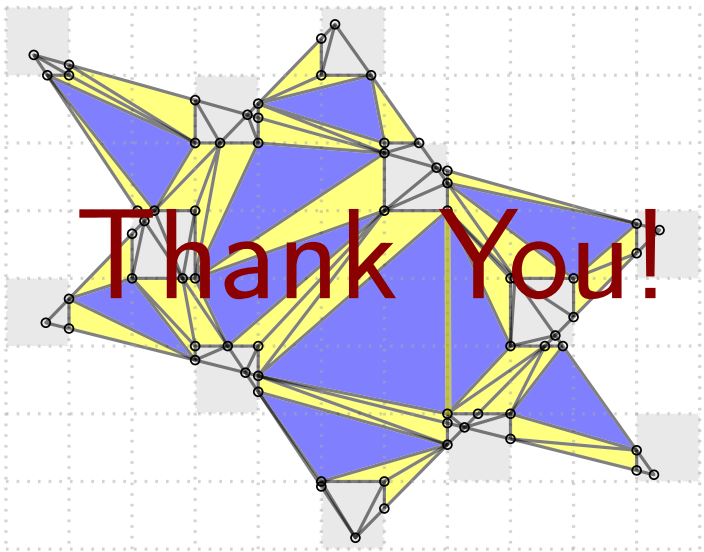
A first implementation of this algorithm give us a practical complexity $O(n\sqrt{n})$ with a big prefactor.

An input mesh of 70.000 faces out with billions of faces.

We work on a lazy version of this algorithm.

– Define a slab, project and lift only if its needed

We guess a practical complexity $O(n)$ and $O(\sqrt{n})$ subdivision with this lazy version


Thank You!