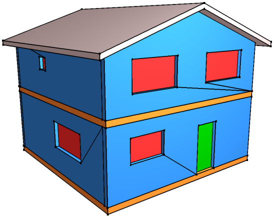
3D Rasterized Combinatorial Maps

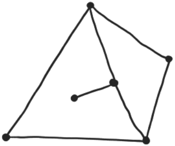
Aldo Gonzalez-Lorenzo[1] Guillaume Damiand[1]
Florent Dupont[1] Jarek Rossignac[2]

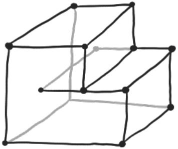
[1]Univ Lyon, CNRS, LIRIS UMR5205, F-69622 (France)
[2]School of Interative Computing, Georgia Institute of Technology, Atlanta (USA)

21 March 2018

- Volumetric object:
  3D object subdivided in cells (3D quasi-manifold)
- Large number of cells
- Complex topology:
  arbitrary number of faces and cofaces, inner edges, inner faces, ...







Representation: **3D combinatorial maps**
Drawback: huge cost in **memory space**

Contents lists available at ScienceDirect

Computer-Aided Design

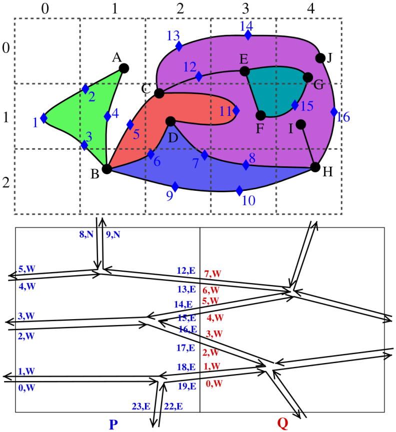
journal homepage: www.elsevier.com/locate/cad

ELSEVIER

Rasterized Planar Face Complex☆

Guillaume Damiand [a,*], Jarek Rossignac [b]

[a] Univ Lyon, CNRS, LIRIS, UMR5205, F-69622, France
[b] School of Interactive Computing, Georgia Institute of Technology, Atlanta, USA



$\longrightarrow 3D$

(using recursion)

Structure

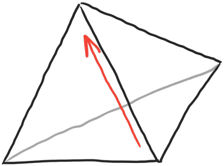
Structure

Definition (3D combinatorial map)

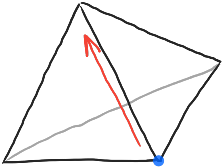
Tuple $(D, \beta_1, \beta_2, \beta_3)$ such that:

1. D is a finite set of darts
2. β_1 is a permutation on D
3. β_2 and β_3 are involutions on D
4. $\beta_1 \circ \beta_3$ is an involution

A dart $d = (v, e, f, c)$ is a sequence of incident cells.



dart d:



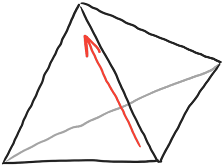
vertex v:

Definition (3D combinatorial map)

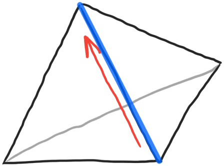
Tuple $(D, \beta_1, \beta_2, \beta_3)$ such that:

1. D is a finite set of darts
2. β_1 is a permutation on D
3. β_2 and β_3 are involutions on D
4. $\beta_1 \circ \beta_3$ is an involution

A dart $d = (v, e, f, c)$ is a sequence of incident cells.



dart d:



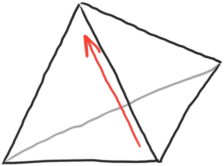
edge e:

Definition (3D combinatorial map)

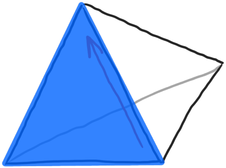
Tuple $(D, \beta_1, \beta_2, \beta_3)$ such that:

1. D is a finite set of darts
2. β_1 is a permutation on D
3. β_2 and β_3 are involutions on D
4. $\beta_1 \circ \beta_3$ is an involution

A dart $d = (v, e, f, c)$ is a sequence of incident cells.



dart d:



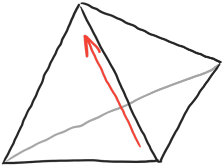
face f:

Definition (3D combinatorial map)

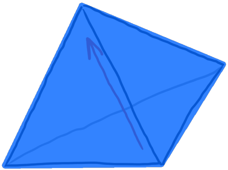
Tuple $(D, \beta_1, \beta_2, \beta_3)$ such that:

1. D is a finite set of darts
2. β_1 is a permutation on D
3. β_2 and β_3 are involutions on D
4. $\beta_1 \circ \beta_3$ is an involution

A dart $d = (v, e, f, c)$ is a sequence of incident cells.



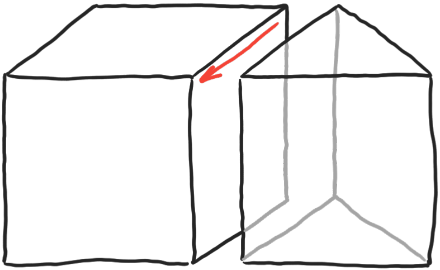
dart d:



volume c:

Operations:

1. β_1: next edge in the same face and volume ($\longleftarrow$ *next*)
2. β_2: switch face (2-cell), and vertex ($\longleftarrow$ *opposite*)
3. β_3: switch volume (3-cell), and vertex ($\longleftarrow$ *opposite* en 3D)

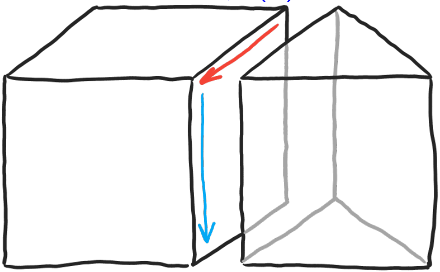


$\beta_1, \beta_2, \beta_3 \longrightarrow$ cells traversal

Operations:

1. β_1: next edge in the same face and volume ($\longleftarrow$ *next*)
2. β_2: switch face (2-cell), and vertex ($\longleftarrow$ *opposite*)
3. β_3: switch volume (3-cell), and vertex ($\longleftarrow$ *opposite* en 3D)

$d \longmapsto \beta_1(d)$

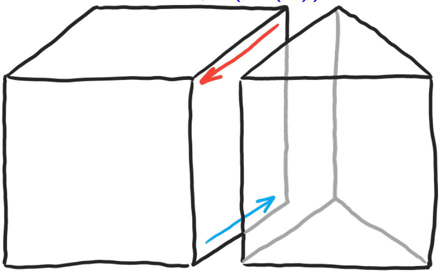


$\beta_1, \beta_2, \beta_3 \longrightarrow$ cells traversal

Operations:

1. β_1: next edge in the same face and volume ($\longleftarrow$ *next*)
2. β_2: switch face (2-cell), and vertex ($\longleftarrow$ *opposite*)
3. β_3: switch volume (3-cell), and vertex ($\longleftarrow$ *opposite* en 3D)

$d \longmapsto \beta_1(\beta_1(d))$

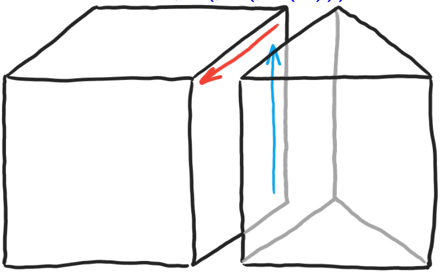


$\beta_1, \beta_2, \beta_3 \longrightarrow$ cells traversal

Operations:

1. β_1: next edge in the same face and volume ($\longleftarrow$ *next*)
2. β_2: switch face (2-cell), and vertex ($\longleftarrow$ *opposite*)
3. β_3: switch volume (3-cell), and vertex ($\longleftarrow$ *opposite* en 3D)

$d \longmapsto \beta_1(\beta_1(\beta_1(d)))$

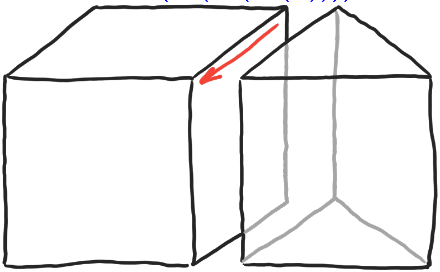


$\beta_1, \beta_2, \beta_3 \longrightarrow$ cells traversal

Operations:

1. β_1: next edge in the same face and volume ($\longleftarrow$ *next*)
2. β_2: switch face (2-cell), and vertex ($\longleftarrow$ *opposite*)
3. β_3: switch volume (3-cell), and vertex ($\longleftarrow$ *opposite* en 3D)

$$d \longmapsto \beta_1(\beta_1(\beta_1(\beta_1(d))))=d$$

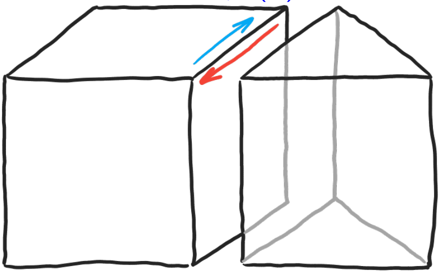


$\beta_1, \beta_2, \beta_3 \longrightarrow$ cells traversal

Operations:

1. β_1: next edge in the same face and volume ($\longleftarrow$ *next*)
2. β_2: switch face (2-cell), and vertex ($\longleftarrow$ *opposite*)
3. β_3: switch volume (3-cell), and vertex ($\longleftarrow$ *opposite* en 3D)

$d \longmapsto \beta_2(d)$

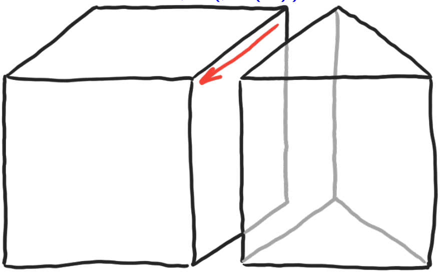


$\beta_1, \beta_2, \beta_3 \longrightarrow$ cells traversal

Operations:

1. β_1: next edge in the same face and volume ($\longleftarrow$ *next*)
2. β_2: switch face (2-cell), and vertex ($\longleftarrow$ *opposite*)
3. β_3: switch volume (3-cell), and vertex ($\longleftarrow$ *opposite* en 3D)

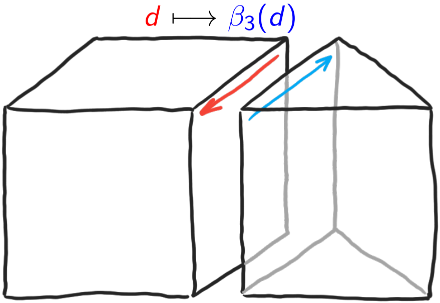
$d \longmapsto \beta_2(\beta_2(d))=d$



$\beta_1, \beta_2, \beta_3 \longrightarrow$ cells traversal

Operations:

1. β_1: next edge in the same face and volume ($\longleftarrow$ *next*)
2. β_2: switch face (2-cell), and vertex ($\longleftarrow$ *opposite*)
3. β_3: switch volume (3-cell), and vertex ($\longleftarrow$ *opposite* en 3D)

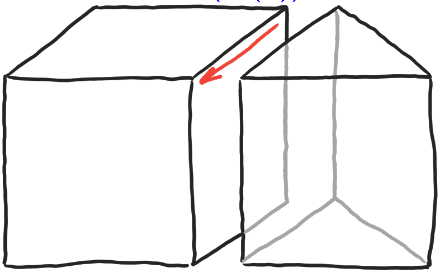


$\beta_1, \beta_2, \beta_3 \longrightarrow$ cells traversal

Operations:

1. β_1: next edge in the same face and volume ($\longleftarrow$ *next*)
2. β_2: switch face (2-cell), and vertex ($\longleftarrow$ *opposite*)
3. β_3: switch volume (3-cell), and vertex ($\longleftarrow$ *opposite* en 3D)

$d \longmapsto \beta_3(\beta_3(d)) = d$

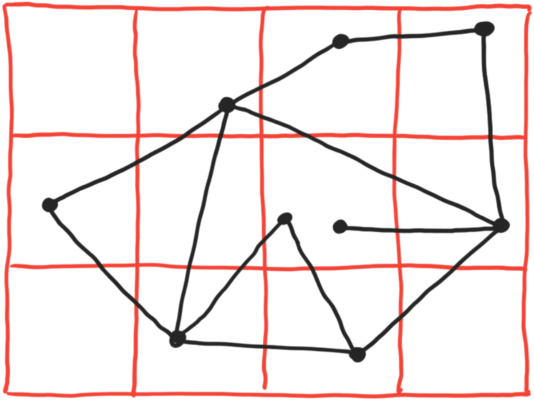


$\beta_1, \beta_2, \beta_3 \longrightarrow$ cells traversal

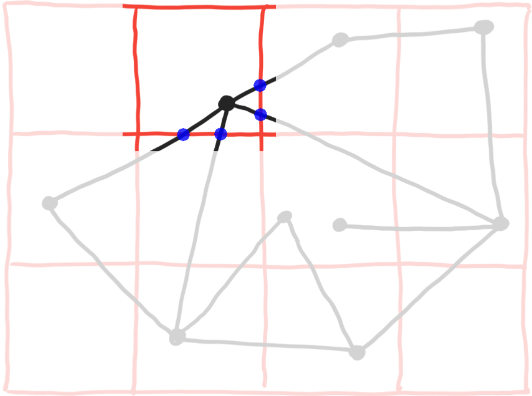
Structure

Rasterized **P**lanar **F**ace **C**omplex (rPFC)

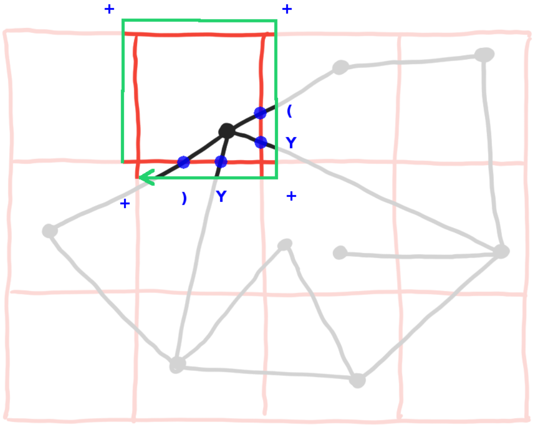
- Compact encoding of a planar graph in a grid of pixels
- Exact topology and approximated geometry
- Condition: an edge of the graph cannot fit entirely inside a pixel
- Directly apply operations **next**, **previous** and **opposite**.
- Only local decoding
- Example . . .



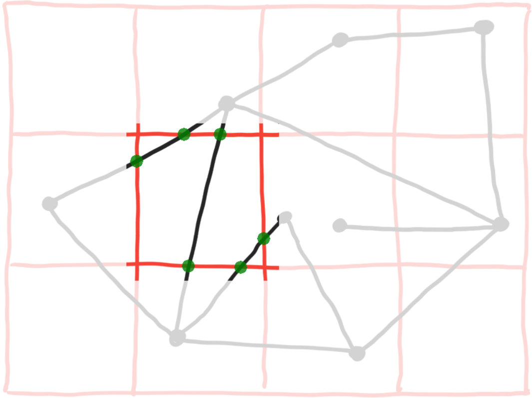
++++	++(Y+Y)+	[(++)+]+	(+++)+
++(+)+	[+][+[+]]+	(+[+]=+)+	(Y+Y++)+
+[+]++	(+YY+)++	(+Y+)++	[+]+++



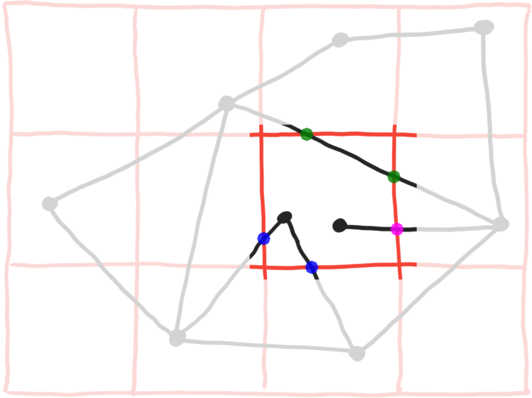
+++++	++(Y+Y)+	[(++)+]+	(+++)+
++(+)+	[+][+[+]]+	(+[+]=+)+	(Y+Y++)+
+[+]++	(+YY+)++	(+Y+)++	[+]+++



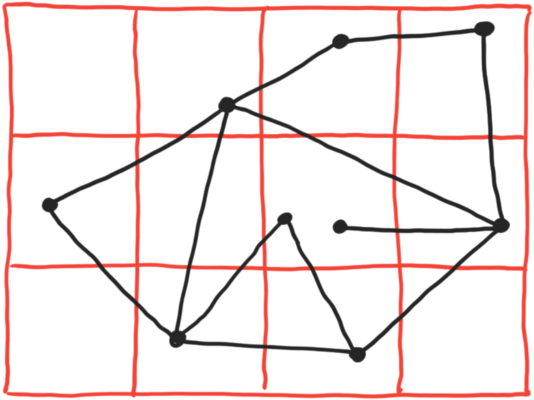
++++	++(Y+Y)+	[(++)+]+	(+++)+
++(+)+	[+][+[+]]+	(+[+]=+)+	(Y+Y++)+
+[+]++	(+YY+)++	(+Y+)++	[+]+++



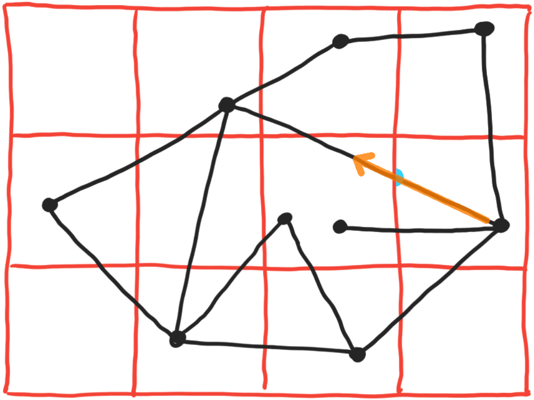
++++	++(Y+Y)+	[(++)+]+	(+++)+
++(+)+	[+][+[+]]+	(+[+]=+)+	(Y+Y++)+
+[+]++	(+YY+)++	(+Y+)++	[+]+++



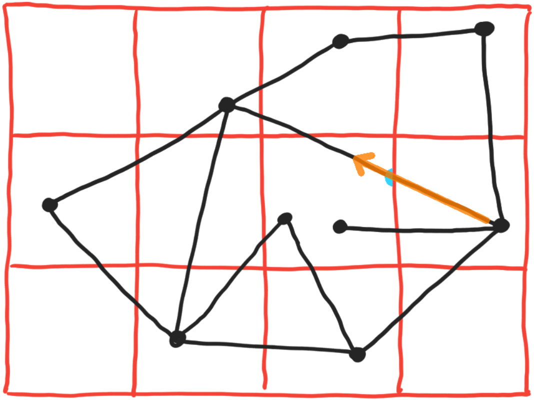
+++++	++(Y+Y)+	[(++)+]+	(+++)+
++(+)+	[+][+[+]]+	(+[+]=+)+	(Y+Y++)+
+[+]++	(+YY+)++	(+Y+)++	[+]+++



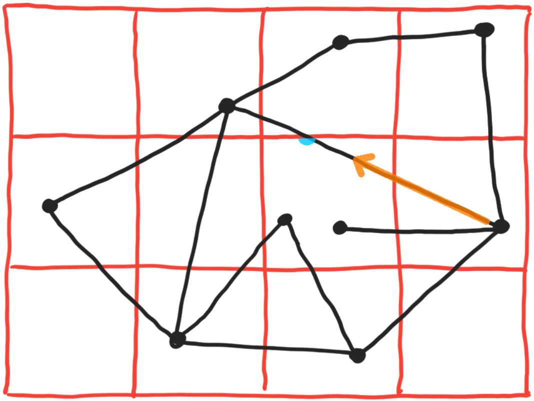
++++	++(Y+Y)+	[(++)+]+	(+++)+
++(+)+	[+][+[+]]+	(+[+]=+)+	(Y+Y++)+
+[+]++	(+YY+)++	(+Y+)++	[+]+++



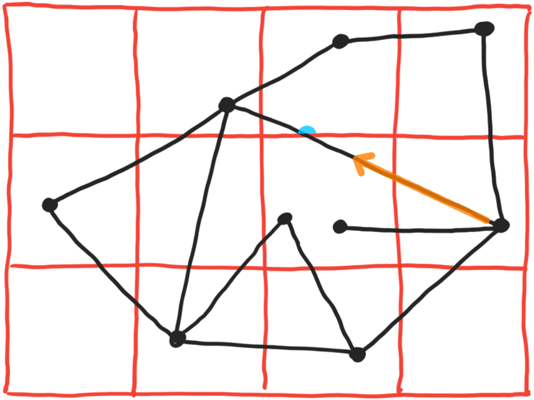
++++	++(Y+Y)+	[(++)+]+	(+++)+
++(+)+	[+][+[+]]+	(+[+]=+)+	(Y+Y++)+
+[+]++	(+YY+)++	(+Y+)++	[+]+++



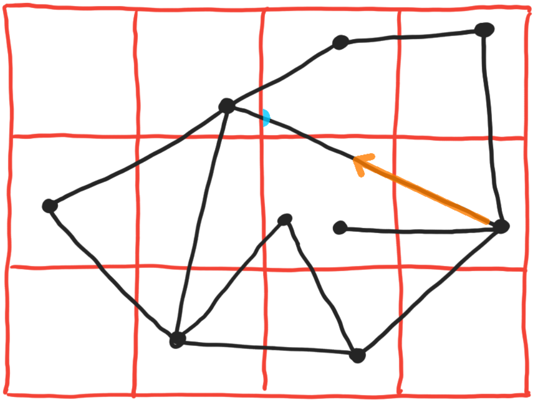
++++	++(Y+Y)+	[(++)+]+	(+++)+
++(+)+	[+][+[+]]+	(+[+]=+)+	(Y+Y++)+
+[+]++	(+YY+)++	(+Y+)++	[+]+++



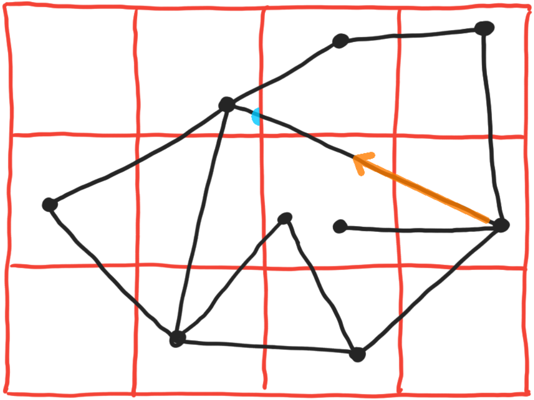
++++	++(Y+Y)+	[(++)+]+	(+++)+
++(+)+	[+][+[+]]+	(+[+]=+)+	(Y+Y++)+
+[+]++	(+YY+)++	(+Y+)++	[+]+++



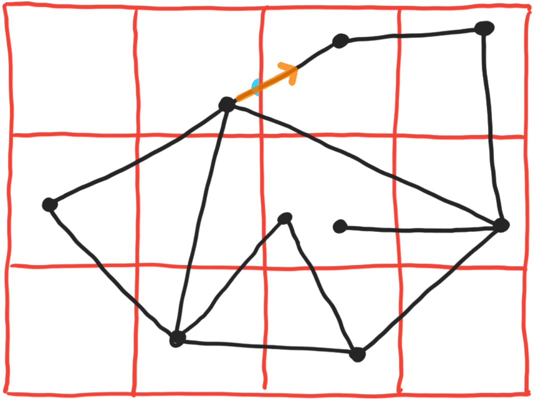
++++	++(Y+Y)+	[(++)+]+	(+++)+
++(+)+	[+][+[+]]+	(+[+]=+)+	(Y+Y++)+
+[+]++	(+YY+)++	(+Y+)++	[+]+++



++++	++(Y+Y)+	[(++)+]+	(+++)+
++(+)+	[+][+[+]]+	(+[+]=+)+	(Y+Y++)+
+[+]++	(+YY+)++	(+Y+)++	[+]+++



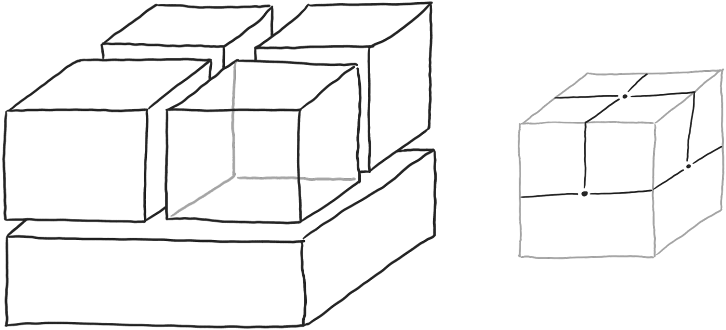
++++	++(Y+Y)+	[(++)+]+	(+++)+
++(+)+	[+][+[+]]+	(+[+]=+)+	(Y+Y++)+
+[+]++	(+YY+)++	(+Y+)++	[+]+++



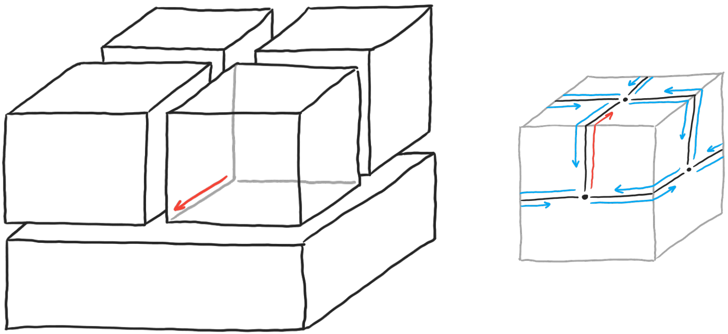
++++	++(Y+Y)+	[(++)+]+	(+++)+
++(+)+	[+][+[+]]+	(+[+]=+)+	(Y+Y++)+
+[+]++	(+YY+)++	(+Y+)++	[+]+++

Structure

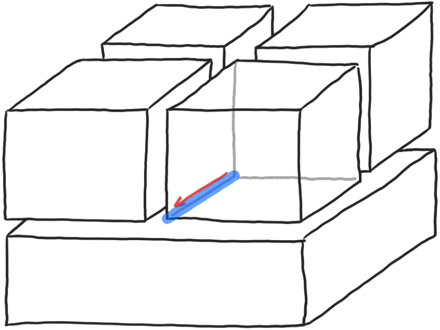
Main idea: intersect a 3-map with the boundary of a voxel

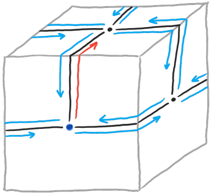


Main idea: intersect a 3-map with the boundary of a voxel

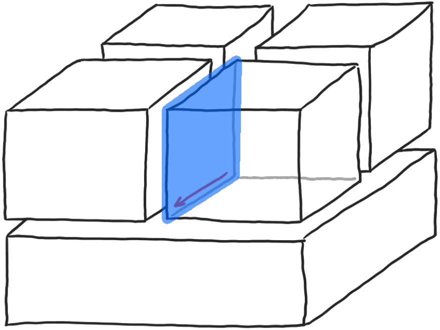


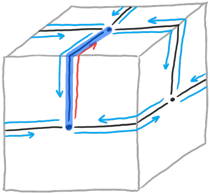
Main idea: intersect a 3-map with the boundary of a voxel



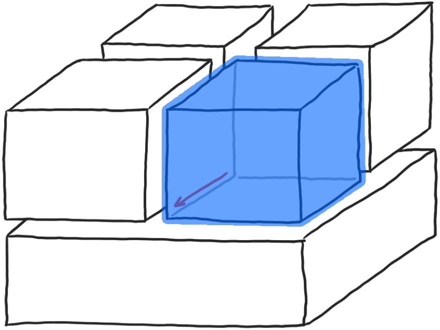


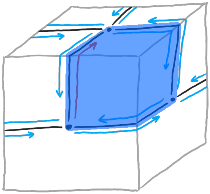
Main idea: intersect a 3-map with the boundary of a voxel



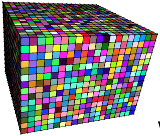


Main idea: intersect a 3-map with the boundary of a voxel

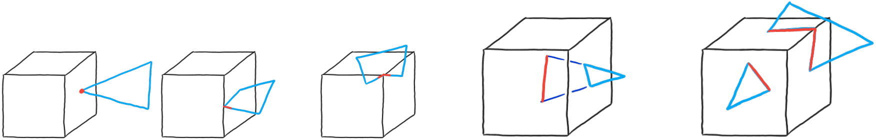




Voxel grid

Regular grid of voxels  with the following conditions:

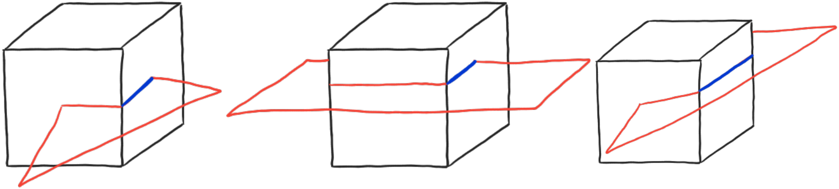
- C1 A 0-/1-/2-cell does not intersect a face/edge/vertex of the voxel grid
- C2 A voxel does not contain any 1-cell in its interior
- C3 The intersection of a 2-cell with the boundary of a voxel is empty or connected



Crossings

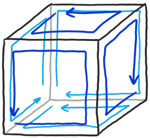
A **crossing** is the intersection of
– a face f of the 3-map with
– an edge e' of a pixel f' of a voxel c'.
Its type is:

Type 0 $\exists e, e < f, e \cap f' \neq \emptyset$ and $\exists v, v < e, v \cap c' \neq \emptyset$
Type 1 $\exists e, e < f, e \cap f' \neq \emptyset$ and $\nexists v, v < e, v \cap c' \neq \emptyset$
Type 2 $\nexists e, e < f, e \cap f' \neq \emptyset$ and $\nexists v, v < e, v \cap c' \neq \emptyset$



Type 0 Type 1 Type 2

Words

Fix an orientation for each face of the voxel: 

For each f', c' $(f' < c')$,

Type 0	$\longrightarrow$	`( Y ··· Y )` or `=`
Type 1	$\longrightarrow$	`{ Y ··· Y }` or `#`
Type 2	$\longrightarrow$	`[ ]`
End of edge	$\longrightarrow$	`+`

Edge-word: word of crossings in an edge (e', f', c').
Crossing: dart in the voxel grid + index in the edge-word

Rasterized darts

Dart $d \in D$, $d = (v, e, f, c)$
Rasterized dart $\rho(d) = (d', i) = (e', f', c'; i)$, where

- c': voxel that contains the vertex v
- f': pixel of c' that contains the intersection of e with the boundary of c'
- e': edge of f' that contains the intersection of f with the boundary of f'
- i is the position of the crossing of f in the edge-word of d'

Operations

$$\rho \circ \beta_3 = \beta_3[3] \circ \rho = (conv[3], type[0], opp) \circ \rho$$
$$\rho \circ \beta_2 = \beta_2[3] \circ \rho = next \circ opp \circ \beta_3[3] \circ \rho$$
$$\rho \circ \beta_1 = \beta_1[3] \circ \rho = opp \circ \beta_3[3] \circ \rho$$

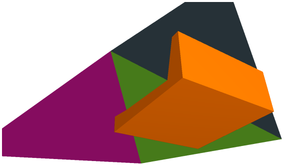
where

$$conv[3](d', i) = (\beta'_3(d'), |w(d')| - i)$$
$$conv[2](d', i) = (\beta'_2(d'), |w(d')| - i)$$
$$\vdots$$

(See details in the paper)

Hierarchical voxel grid

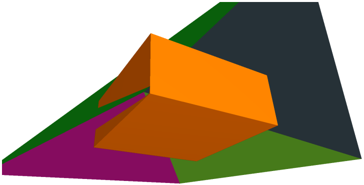
Condition **C3** is hard to satisfy: refining the grid is not sufficient

4×4 grid: 

Solution: Octree of voxels $\rightarrow$ quadtree of pixel, bintree of edges.

Hierarchical voxel grid

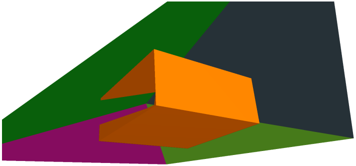
Condition **C3** is hard to satisfy: refining the grid is not sufficient

8×8 grid: 

Solution: Octree of voxels $\rightarrow$ quadtree of pixel, bintree of edges.

Hierarchical voxel grid

Condition **C3** is hard to satisfy: refining the grid is not sufficient

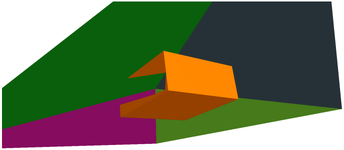
16×16 grid: 

Solution: Octree of voxels $\rightarrow$ quadtree of pixel, bintree of edges.

Hierarchical voxel grid

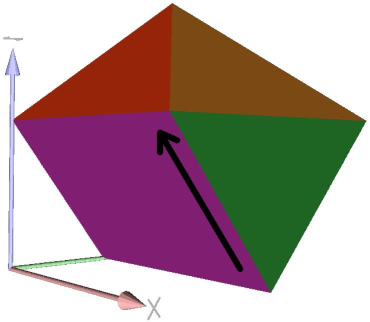
Condition **C3** is hard to satisfy: refining the grid is not sufficient

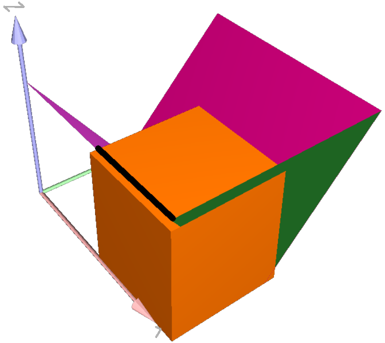
32×32 grid:



Solution: Octree of voxels $\rightarrow$ quadtree of pixel, bintree of edges.

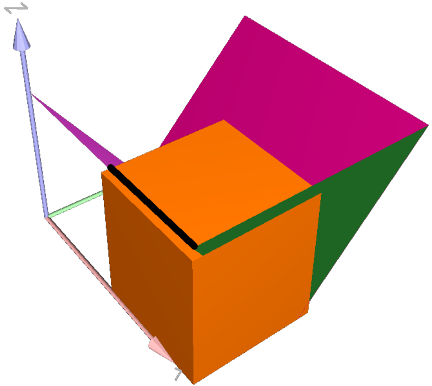
Example: rasterized dart





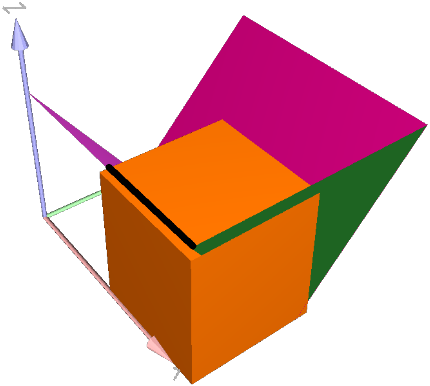
[1 0 0; ; 5;] ++(+)+

Example: β_1



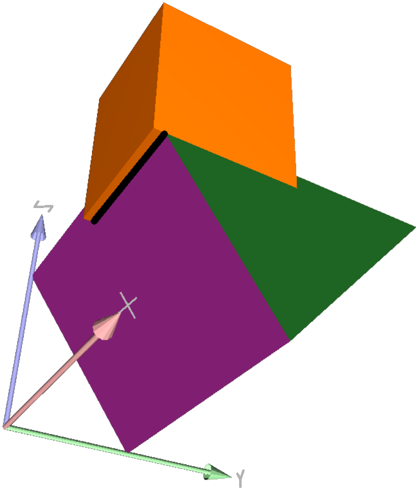
$\beta_1[3] = opp \circ \beta_3[3]$

Example: β_1



$$\beta_1[3] = opp \circ \beta_3[3]$$
$$\beta_3[3] = (conv[3], type[0], opp)$$

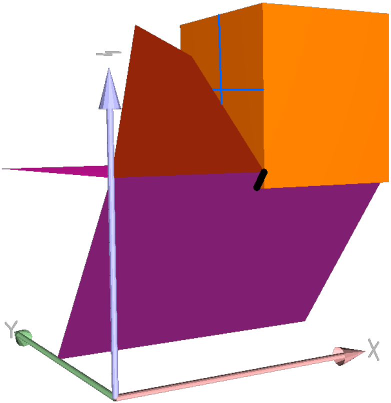
Example: β_1



$$\beta_1[3] = opp \circ \beta_3[3]$$
$$\beta_3[3] = (conv[3], type[0], opp)$$

[1 0 1; ; 2;] (+)+++

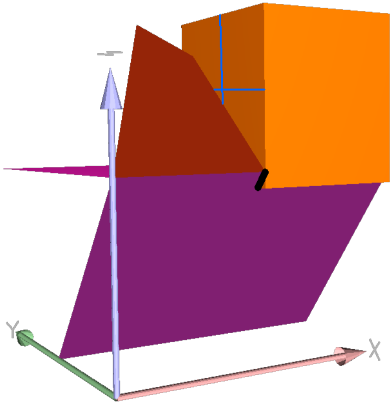
Example: β_1



$$\beta_1[3] = \textcolor{blue}{opp} \circ \beta_3[3]$$

[1 0 1; ; 0; 0] +(+Y+)+

Example: β_1



$$\beta_1[3] = opp \circ \beta_3[3]$$

[1 0 1; ; 0; 0] +(+Y+)+

Structure

3-rmap: **3D-array** of **octrees** of **arrays** of **quadtrees** of **pixel-words**

We want:

- Compact data structure
- Fast access to the symbol of any crossing

We propose four implementations, among many others.

- V1 Explicit structure (3D-array of octrees...)
- V2 Octree and dictionary of voxel-words
- V3 Octree and dictionary of voxel-words with encoded pixel-words
- V4 Only one word

Structure

- Rasterized 3D combinatorial map: encoding of a 3D combinatorial map using voxels and words
- Same properties as the rPFC: exact topology, approximated geometry, local decoding
- 11 symbols (two more than the rPFC)
- 3 conditions (one more than the rPFC)
- Hierarchical (not regular) voxel grid

- Rasterized 3D combinatorial map: encoding of a 3D combinatorial map using voxels and words
- Same properties as the rPFC: exact topology, approximated geometry, local decoding
- 11 symbols (two more than the rPFC)
- 3 conditions (one more than the rPFC)
- Hierarchical (not regular) voxel grid

Download this presentation:

`http://aldo.gonzalez-lorenzo.perso.luminy.univ-amu.fr/downloads.html`

Thank you for your attention