Representing Polyhedra

Fumei Lam

Outline

Goal: Develop a representation that fully captures all topological properties of subdivisions (2D, 3D)

- Doubly Connected Edge List
- Winged Edge
- Quad Edge
  - Application: Voronoi diagrams and Delaunay triangulations
- Facet Edge

Application-specific data

- Geometric Information

  Vertex coordinates (edge lengths, face normals, ...)

- Attribute Information

  Color, Temperature/Pressure, Population, other statistics

- Applications

  Partitioning: nearest neighbor (Voronoi Diagram/ Delaunay Triangulation)

Doubly-Connected Edge List (Eastman, 1982)

Every edge is represented by two half-edge structures

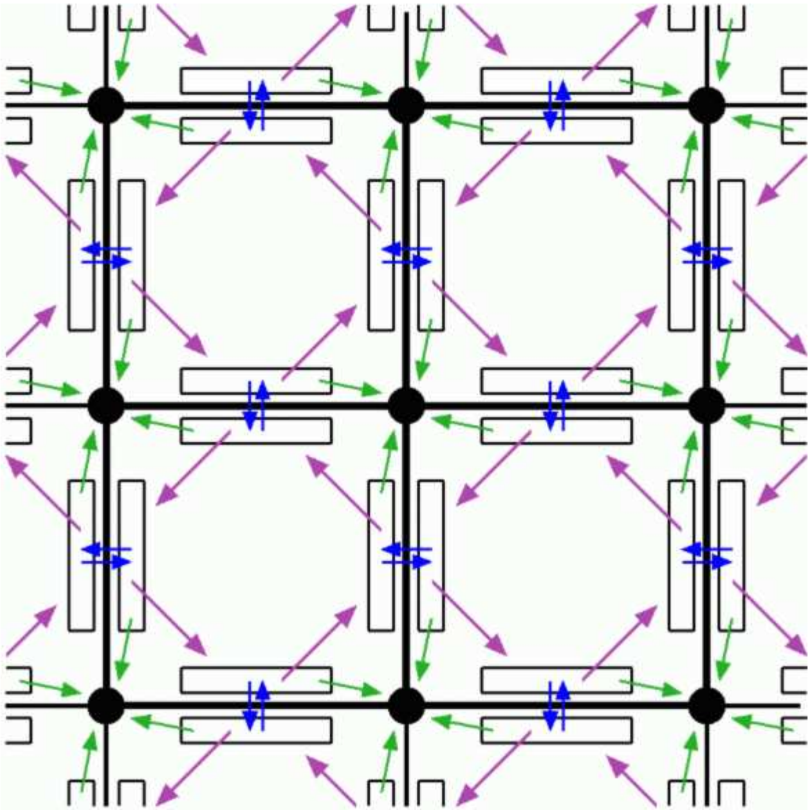
Pointers: Vertex, Sym, and Next

Sym points to symmetric half-edge

- Same Edge
- Opposite Vertex
- Opposite Face

Next points to half-edge
Counter-Clockwise around Face on left

- Same Face
- CCW Vertex around Face on left



Vertex points to an incident edge

Face points to edge on its perimeter

Data

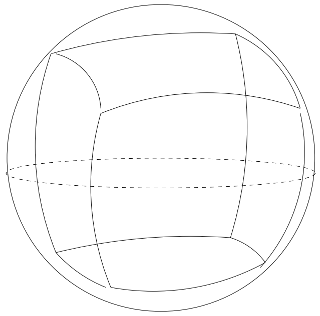
- Geometric Information in Vertices
- Topological Information in half-edges

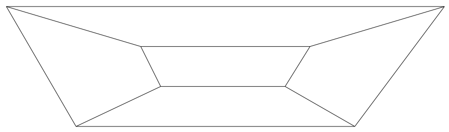
Time Complexity

- Time is linear in the amount of information gathered
- Independent of global complexity

Winged Edge Data Structure

Planar graph (subdivision of a sphere)





Components of a set of Polyhedra

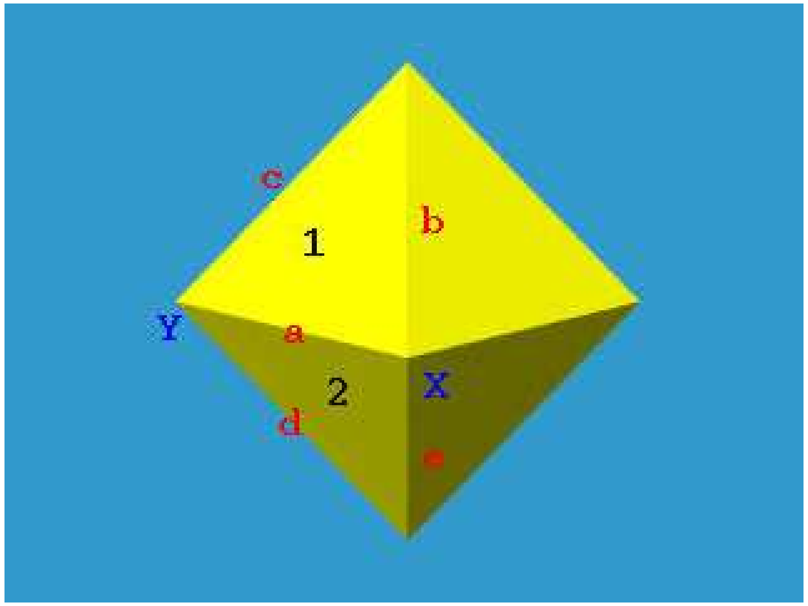
- Bodies, Faces, Edges, Vertices
- Each body contains ring of faces, ring of edges, ring of vertices

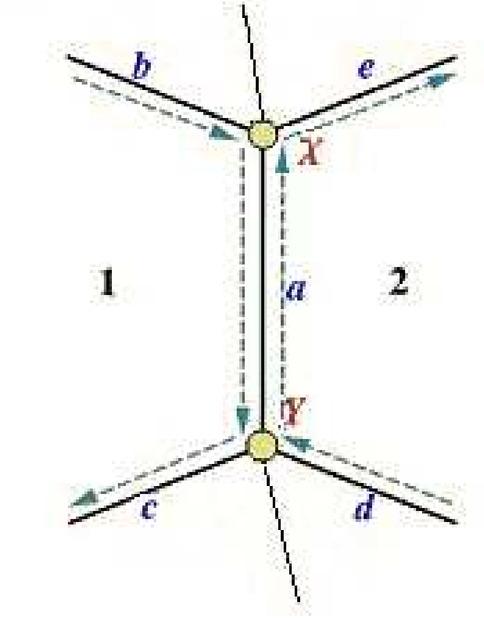
Each vertex points to one of its adjacent edges

Each face points to one of the edges on its perimeter (boundary)

Each edge points to

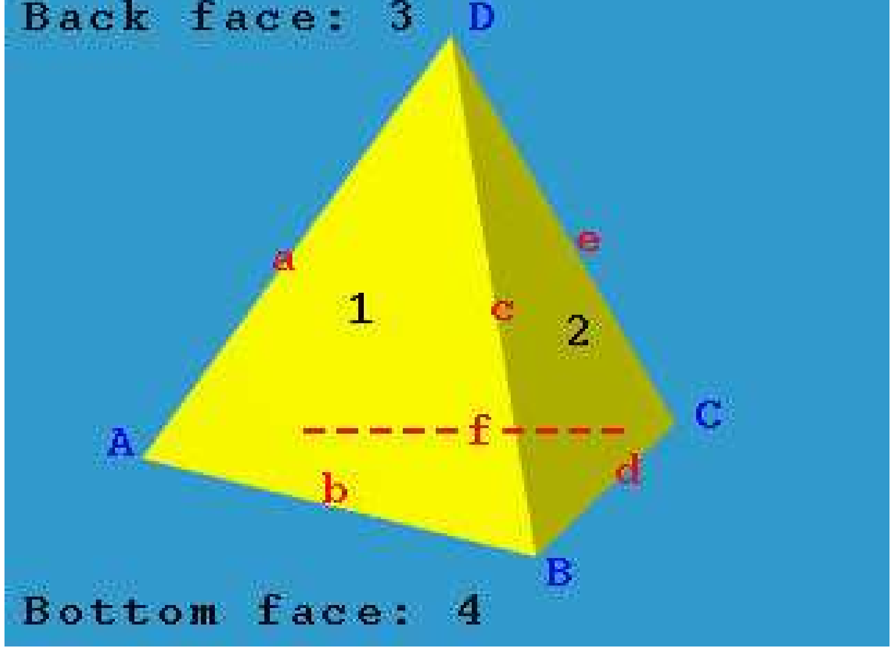
- two neighboring faces and defining vertices
- four immediate neighboring edges about its face perimeter ("wings" of the edge)





Edge	Vertices		Faces		Left Traverse		Right Traverse	
Name	Start	End	Left	Right	Pred	Succ	Pred	Succ
a	Y	X	1	2	b	c	d	e

Topology stored in edge nodes, geometry stored in vertex nodes



Operations

- Enumerate vertices, edges, faces
- Sequential Access

Edge	Vertices		Faces		Left Traverse		Right Traverse	
Name	Start	End	Left	Right	Pred	Succ	Pred	Succ
a	A	D	3	1	e	f	b	c
b	A	B	1	4	c	a	f	d
c	B	D	1	2	a	b	d	e
d	B	C	2	4	e	c	b	f
e	C	D	2	3	c	d	f	a
f	A	C	4	3	d	b	a	e

Euler Operators

- Modify polyhedron by adding or deleting vertices, edges and faces.
- Euler formula

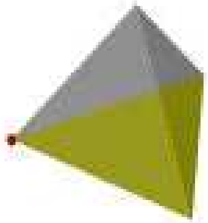
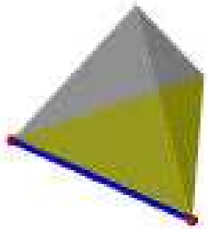
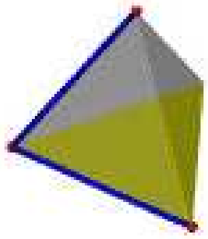
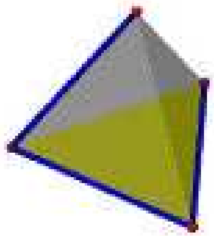
$$V - E + F = 2$$

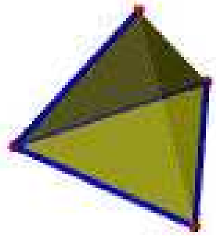
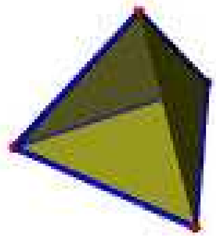
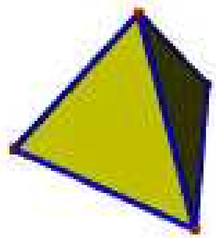
V vertices $\quad$ E edges $\quad$ F faces

- Operators edit polyhedron so that Euler formula is always satisfied
- Operators:
  Make (Mxy) group
  Kill (Kxy) group
  x, y edare elements of the model (e.g., a vertex, edge, face)

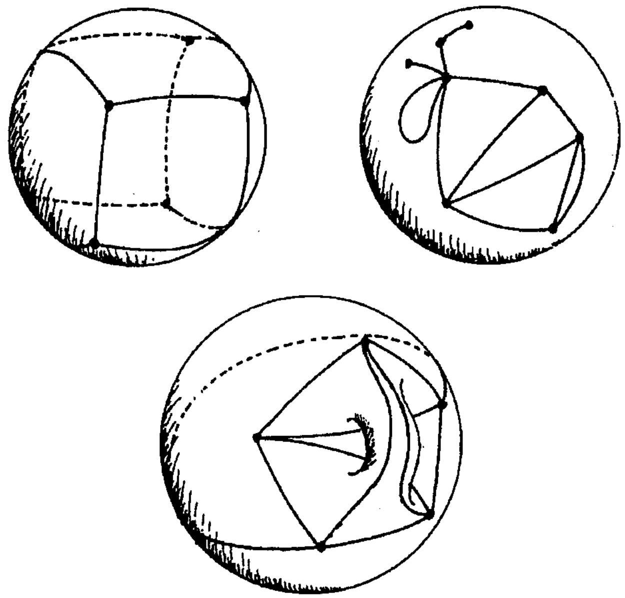
- (Mantyla, 1984) Every topologically valid polyhedron can be constructed from an initial polyhedron by a finite sequence of Euler operators

Operator Name	Meaning	V	E	F
MEV	Make edge and vertex	+1	+1	
MFE	Make face and edge		+1	+1
KEV	Kill edge and vertex	-1	-1	
KFE	Kill face and edge		-1	-1

Operator Name	Meaning	V	E	F	*Result*
MFV	Make a face and a vertex	+1		+1	
MEV	Make an edge and a vertex	+1	+1		
MEV	Make an edge and a vertex	+1	+1		
MEV	Make an edge and a vertex	+1	+1		

Operator Name	Meaning	V	E	F	*Result*
MFE	Make a face and an edge		+1	+1	
MFE	Make a face and an edge		+1	+1	
MFE	Make a face and an edge		+1	+1	

Examples of subdivisions



QuadEdge Data Structure (Guibas and Stolfi, 1985)

Orientation: two ways of defining local clockwise rotation

- Oriented element of subdivision is an element x together with an orientation of a disk containing x

Direction:

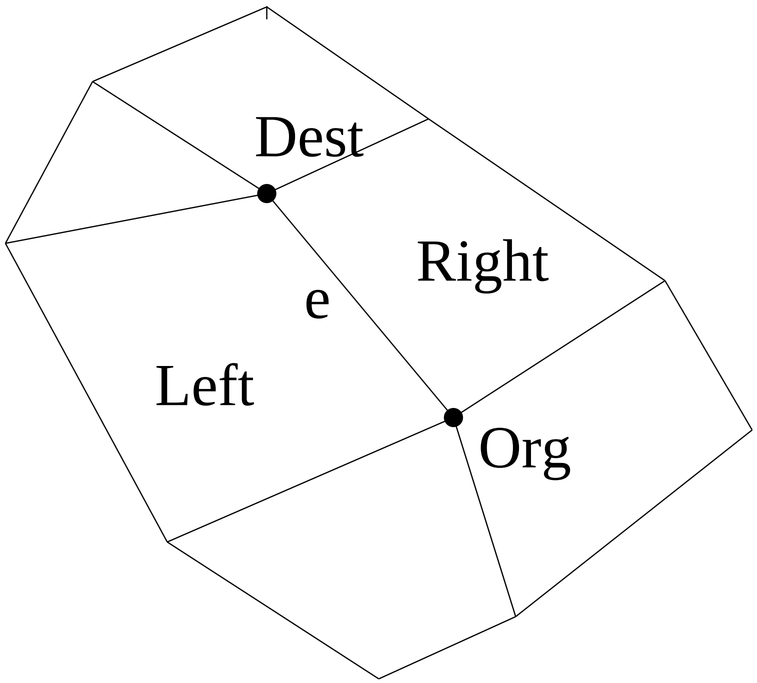
- Directed edge is an edge together with a direction along it

Four ways to choose orientation and direction

- Bug Demo

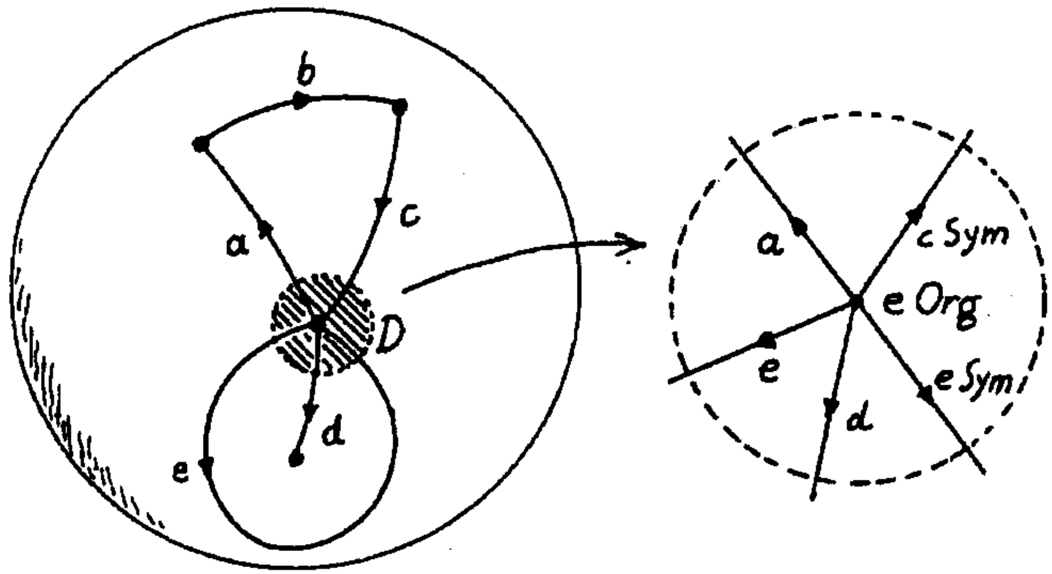
Given an oriented and directed edge, define

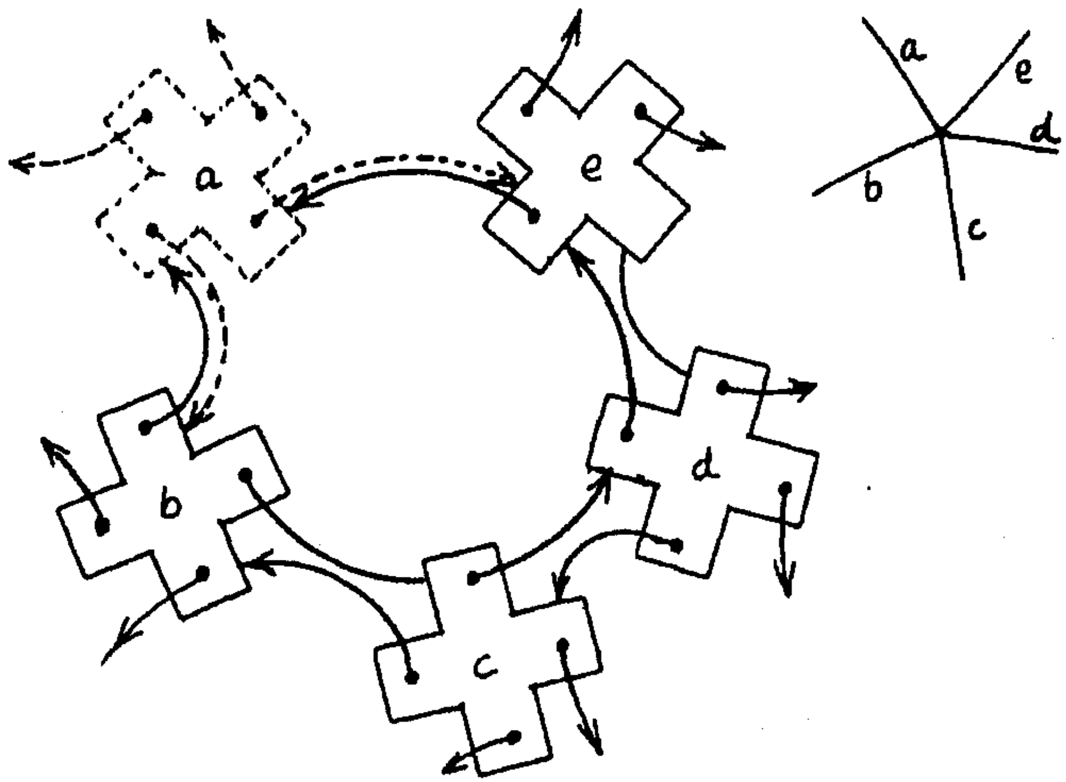
- *e Org*, *e Dest* vertex of origin, vertex of destination
- *e Left*, *e Right* left face, right face
- Elements *e Org*, *e Left*, *e Right*, *e Dest* given orientation that agrees locally with orientation of *e*



Each vertex has ring of edges (perimeter)

- e *Onext* next counterclockwise edge with same origin

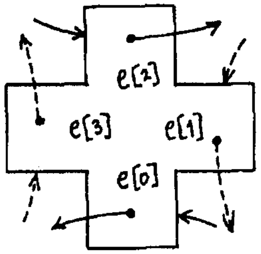


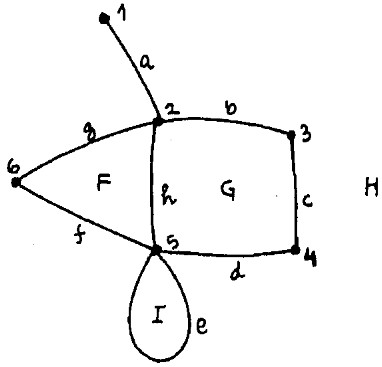


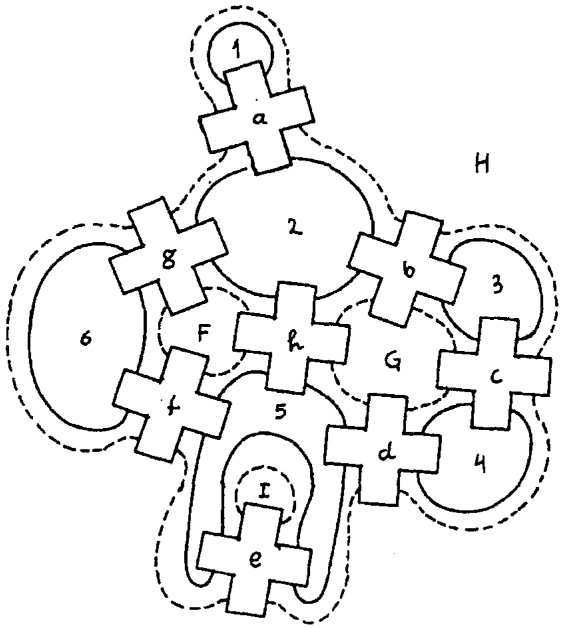
a
b
c
d
e
a
e
d
b
c

Each face has ring of edges

- *e Lnext* next counterclockwise edge with same left face





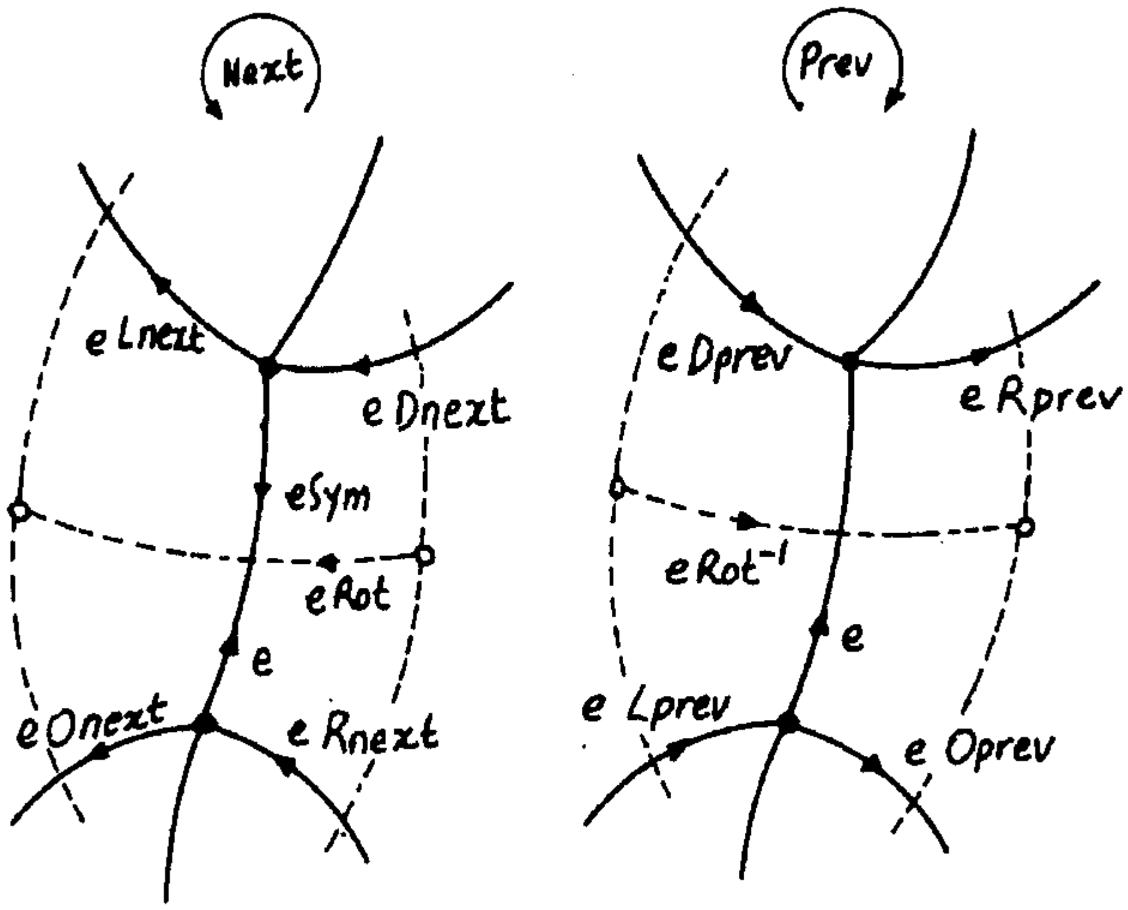


Edge Functions:

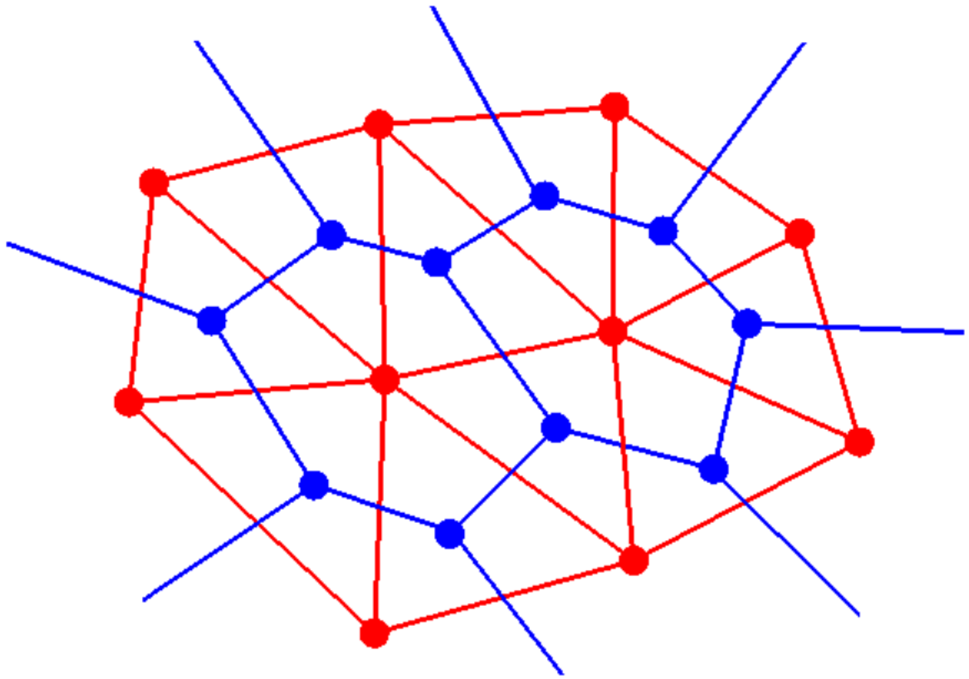
Rot: Bug rotates 90 degrees

Sym: Bug rotates back to front

Flip: Bug flips up-side down



Dual subdivisions



1. $(e\ Dual)\ Dual = e$
2. $(e\ Sym)\ Dual = (e\ Dual)\ Sym$
3. $(e\ Flip)\ Dual = (e\ Dual)\ Flip\ Sym$
4. $(e\ Lnext)\ Dual = (e\ Dual)\ Onext^{-1}$

Relations Between Edges: Edge Algebra

Some Properties of Flip, Rot, and Onext:

$$e\ Rot^4 = e$$

$$e\ Rot^2 \neq e$$

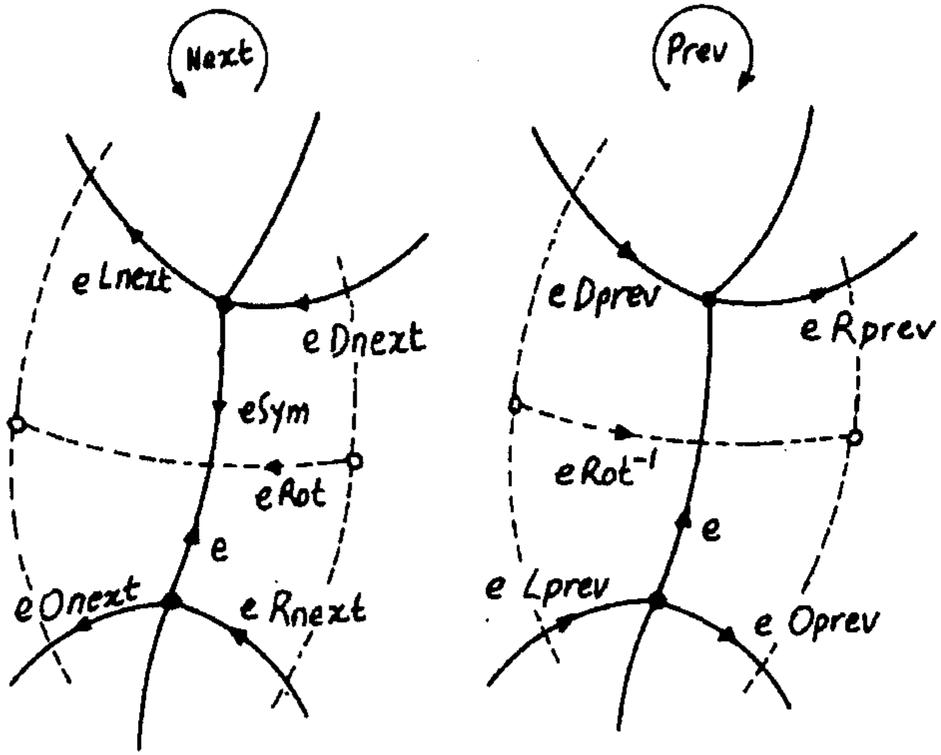
$$e\ Flip^2 = e$$

$$e\ Flip\ Rot\ Flip\ Rot = e$$

$$e\ Rot\ Flip\ Rot\ Flip = e$$

$$e\ Rot\ Onext\ Rot\ Onext = e$$

$$e\ Flip\ Onext\ Flip\ Onext = e$$



Properties of Edge Algebra deduced from those above:

$e\ Flip^{-1} = e\ Flip$

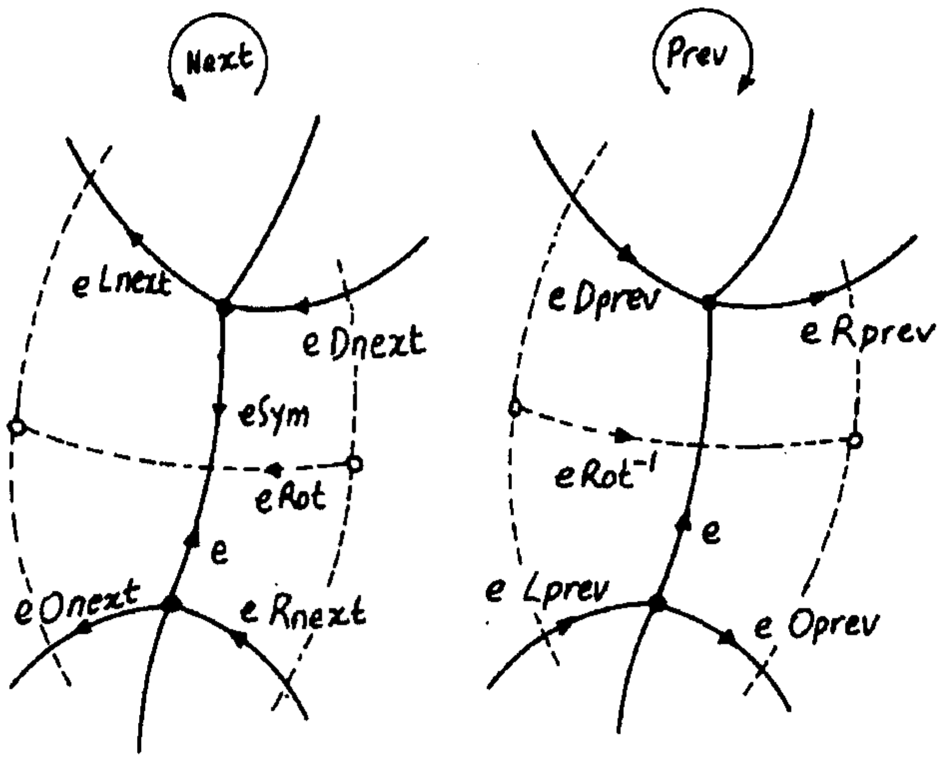
$e\ Sym = e\ Rot^2$

$e\ Rot^{-1} = e\ Rot^3$

$e\ Rot^{-1} = e\ Flip\ Rot\ Flip$

$e\ Onext^{-1} = e\ Rot\ Onext\ Rot$

$e\ Onext^{-1} = e\ Flip\ Onext\ Flip$



NOTE: Every function defined so far can be expressed as a constant number of Rot, Flip, and Onext operations, independently of the local topology and the global size and complexity of the subdivision.

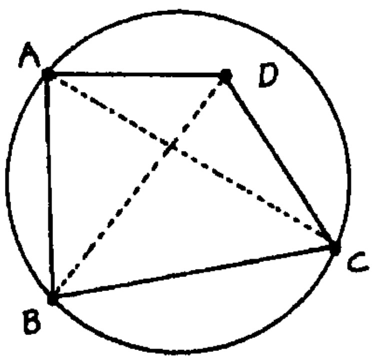
Represent Subdivision and its Dual Simultaneously

- Vertices $\longrightarrow$ Faces
- Edges $\longrightarrow$ Edges

Application: Voronoi Diagrams and Delaunay Triangulations

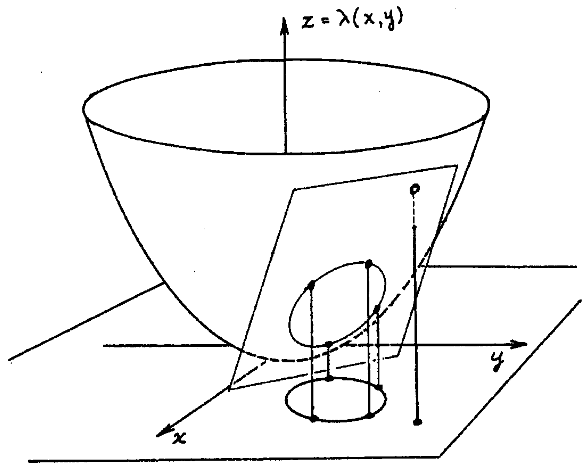
The InCircle Test

Define $InCircle(A, B, C, D)$ to be true if D lies to the left of the oriented circle ABC, false otherwise.



The test $InCircle(A, B, C, D)$ is equivalent to the test

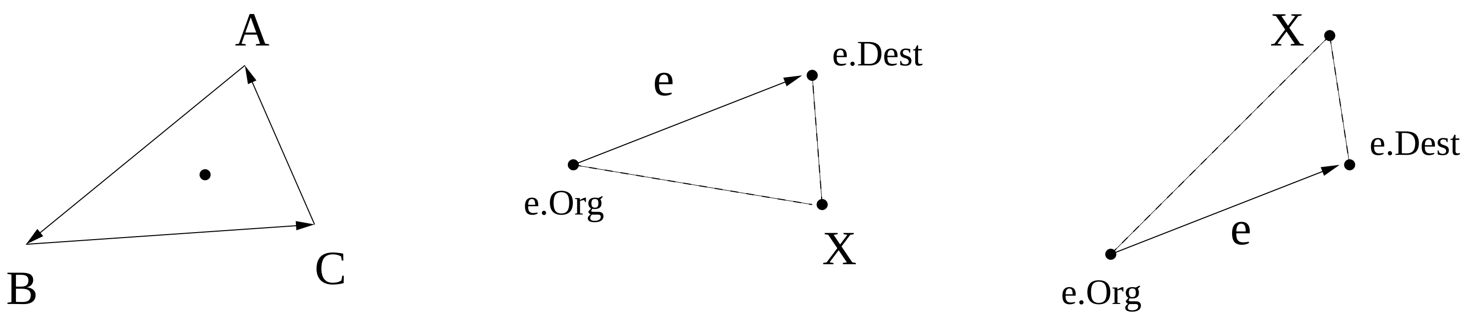
$$det \begin{bmatrix} x_A & y_A & x_A^2 + y_A^2 & 1 \\ x_B & y_B & x_B^2 + y_B^2 & 1 \\ x_C & y_C & x_C^2 + y_C^2 & 1 \\ x_D & y_D & x_D^2 + y_D^2 & 1 \end{bmatrix} > 0$$



$CCW(A, B, C) =$ True if points A, B, C form counterclockwise oriented triangle

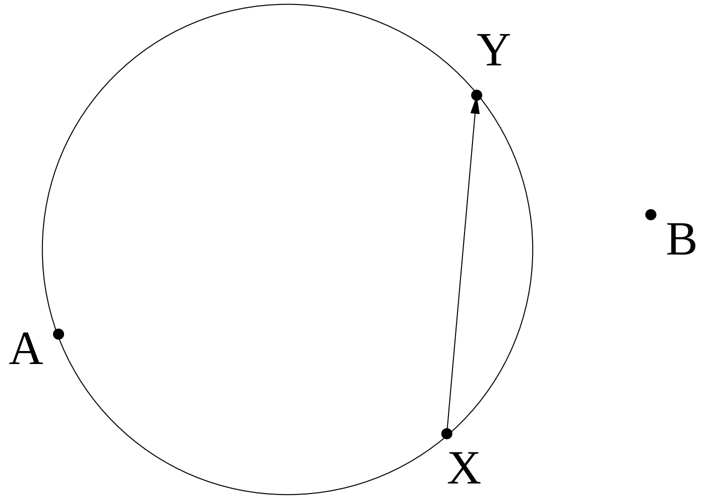
$$RightOf[X, e] = CCW[X, e.Dest, e.Org]$$

$$LeftOf[x, e] = CCW[X, e.Org, e.Dest]$$



1. Let L and R be two sets of points. Any edge of the Delaunay triangulation of $L \cup R$ whose endpoints are both in L is in the Delaunay triangulation of L.

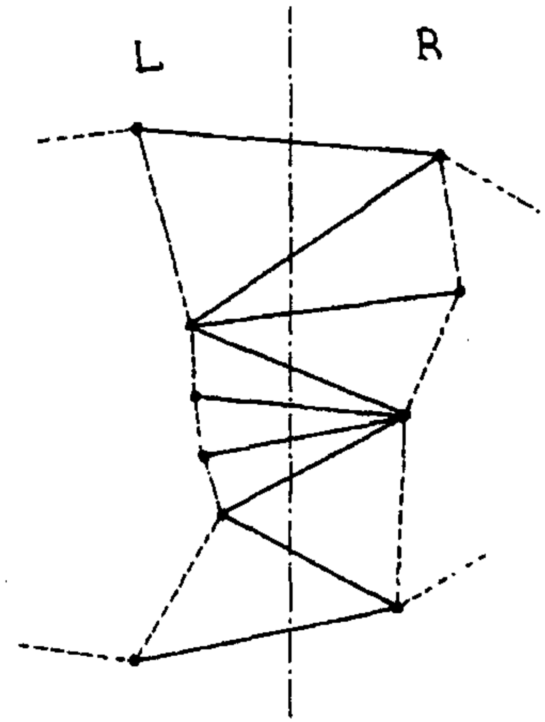
2. An edge XY is Delaunay if $InCircle(A, X, Y, B)$ is false for every pair of sites A and B to the left and right, respectively, of line XY.



A triangulation T is Delaunay if and only if all its edges pass the circle test.

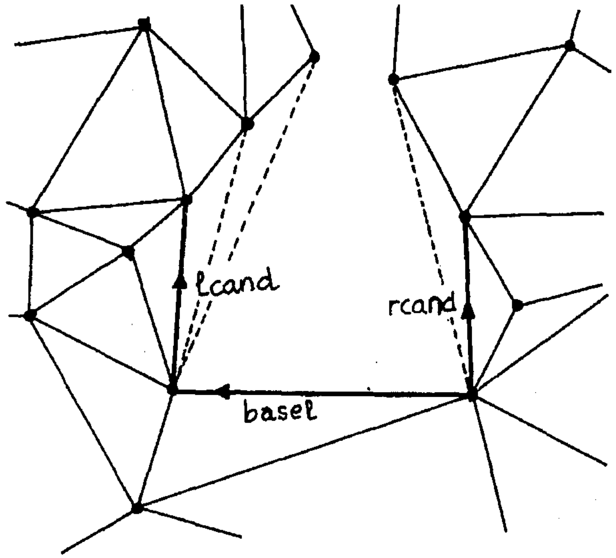
Divide-and-Conquer Algorithm

- Partition the points into two halves by x-coordinate, *Left* and *Right*
- Find Delaunay triangulation of each half

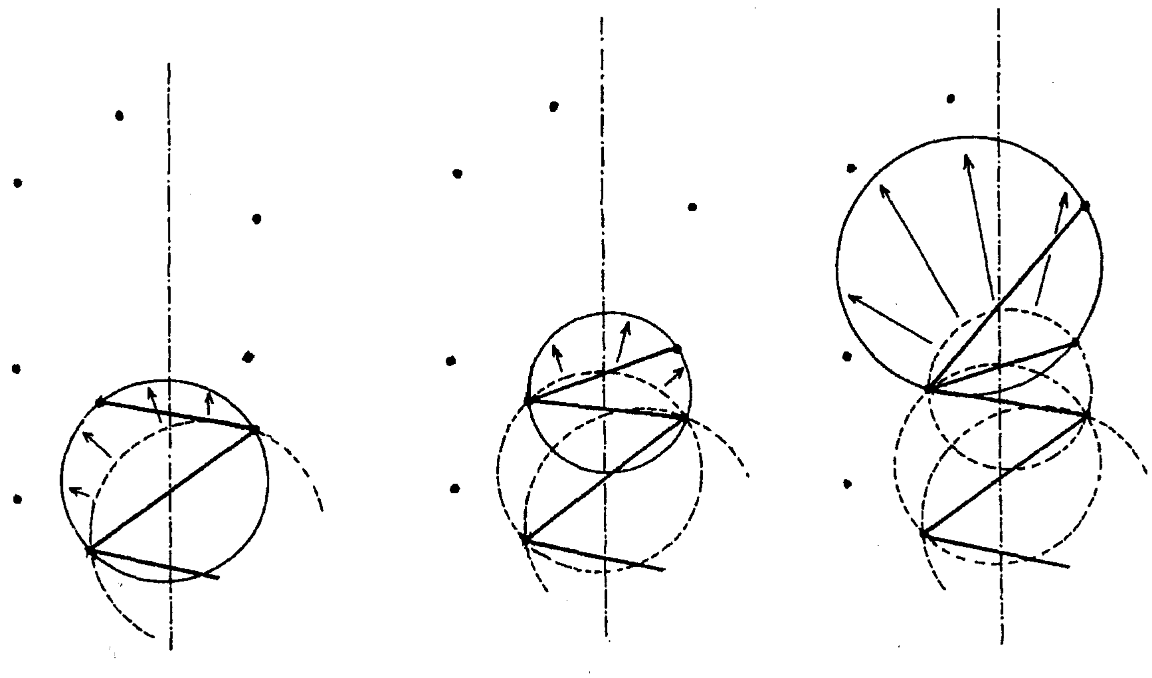


- Marry two half triangulations

  Create a first cross edge basel



Lemma. Any two cross edges adjacent in the y-ordering share a common vertex. The third side of the triangle they define is either an $L-L$ or a $R-R$ edge.



- Locate the first L point to be encountered by rising bubble
- Delete L edges out of basel.Dest that fail the circle test
- Locate first R point to be encountered and delete R edges out of basel.Org that fail the circle test
- Next cross edge is to be connected to either lcand.Dest or rcand.Dest
- If both are valid, then choose the appropriate one using the InCircle test

- Uses geometric primitives InCircle and CCW(A,B,C)
- Overall cost of merge pass is linear in size of L and R
- Running time for entire algorithm $O(n \log n)$

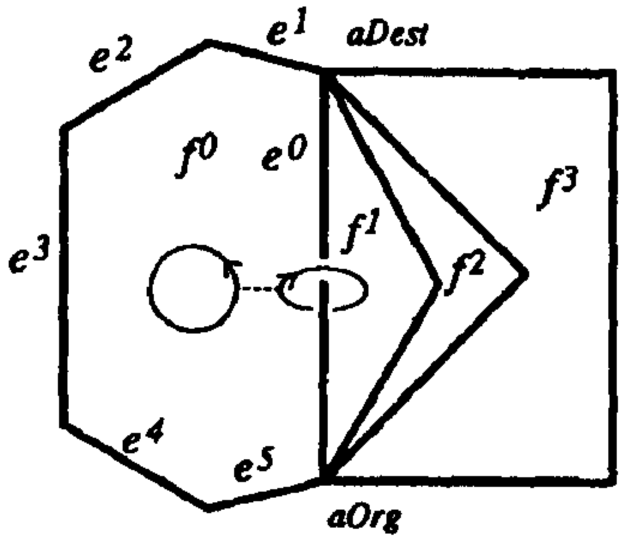
Facet-Edge (Dobkin, Laszlo, 1987)

Models polyhedral complexes in $\mathbb{R}^3$, surfaces of 4-polyhedra

Each facet f has an edge-ring $\mathcal{E}_f = (e^0, e^1, \ldots e^{n-1})$

Each edge e has a facet-ring $\mathcal{F}_e = (f^0, f^1, \ldots f^{m-1})$

Facet-edge pair $a = (f_a, e_a)$: facet f_a and edge e_a adjacent to f_a



Traversal Functions

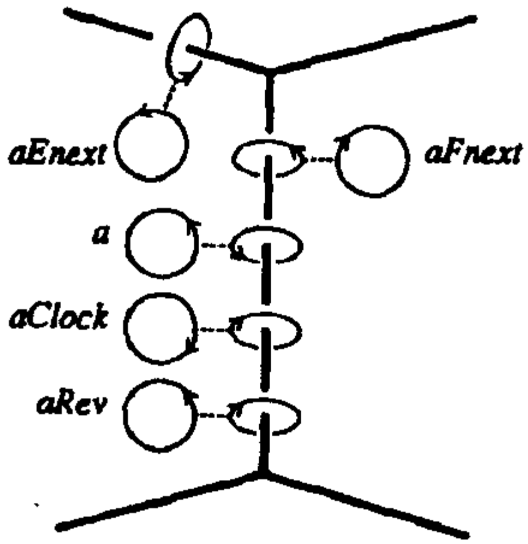
Each function is applied to facet-edge pair

Fnext: next face in facet-ring

Enext: next edge in edge-edge ring

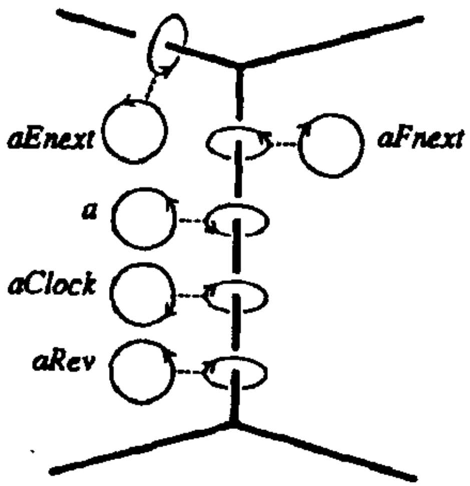
Rev: $e_{a'} = e_a, f_{a'} = f_a$, directions of $\mathcal{E}_{a'}, \mathcal{E}_a$ are the same, direction of $\mathcal{F}_{a'}$ is opposite that of $\mathcal{F}_a$

Clock: $e_{a'} = e_a, f_{a'} = f_a$, directions of $\mathcal{E}_{a'}, \mathcal{F}_{a'}$ are opposite those of $\mathcal{E}_a, \mathcal{F}_a$



Traversal function relations

1. $aRev^2 = a$
2. $aClock^2 = a$
3. $aRevClock = aClockRev$
4. $aFnext^{-1} = aClockFnextClock$
5. $aEnext^{-1} = aClockEnextClock$

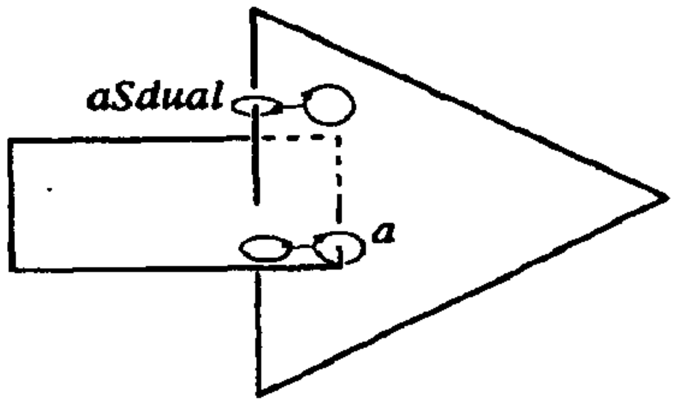


Space-Duality

Space Dual

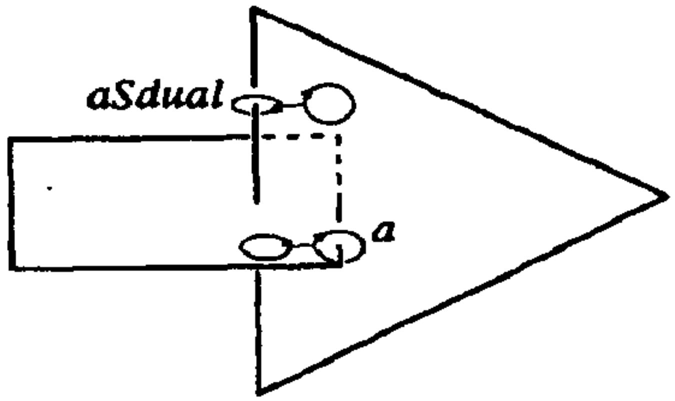
- Cells $\rightarrow$ Vertices
- Facets $\rightarrow$ Edges
- Edges $\rightarrow$ Facets
- Vertices $\rightarrow$ Cells

Sdual applies to a facet-edge pair a and returns a second facet-edge pair $aSdual$ belonging to space dual



Relations

- $aSdual^2 = a$
- $aClockSdual = aSdualClock$
- $aFnext = aSdualEnextSdual$
- $aEnext = aSdualFnextSdual$

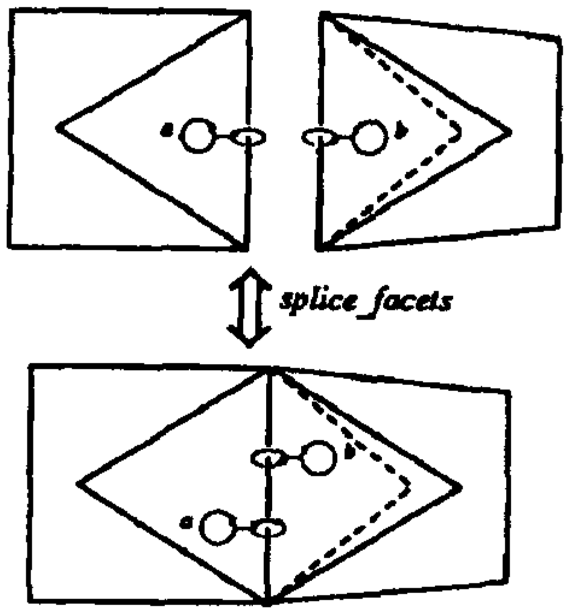


Operators

Splice_facets

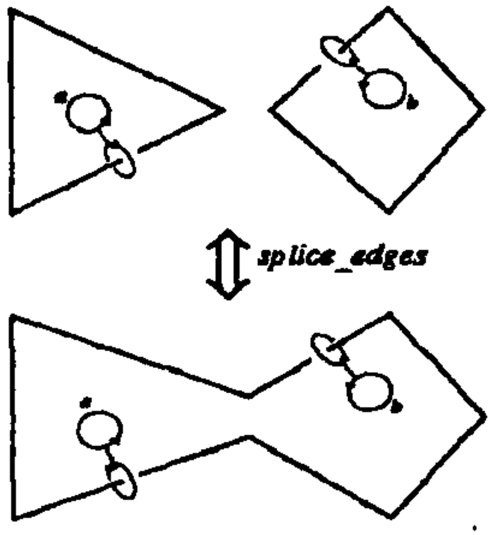
- Input: two facet edge pairs (a, b)
- Operation modifies facet-rings $\mathcal{F}_a, \mathcal{F}_b$

1. if rings are distinct, combines them into one ring
2. if rings are identical, breaks the ring into two distinct rings



Splice_edges

- Input: two facet edge pairs (a, b)
- Operation modifies edge-rings $\mathcal{E}_a, \mathcal{E}_b$
  1. if rings are distinct, combines them into one ring
  2. if rings are identical, breaks the ring into two distinct rings



Applications

Decomposing a polyhedron

- partitioning polyhedron into simplier constituents
- non-convex, handles

Incremental Construction of 3-Dimensional Delaunay triangulation

References

[B] B. Baumgart, *Winged-Edge Polyhedron Representation for Computer Vision*, National Computer Conference, May 1975

[DL] D. Dobkin, M. Laszlo, *Primitives for the Manipulation of Three-Dimensional Subdivisions*, Algorithmica, vol. 4, pp. 3–32, 1989.

[GS] L. Guibas, J. Stolfi, *Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams*, ACM Transactions on Graphics, Vol. 4, No. 2, April 1985 74-123

[L] Legakis, J., 6.838 Spring 1998