Local Topological Properties of Polyhedra

Wm Randolph Franklin

Rensselær Polytechnic Institute
Troy, NY, 12180, USA

wrf@ecse.rpi.edu
+1 (518) 276-6077, *fax:* -6261

?

What various data structures are possible to represent polygons and polyhedra?

Polygon: The union of a finite number of non-degenerate triangles.

This includes nested disjoint components and non-manifold cases.

Obvious Answer

Complete topology:

- vertices,
- edges, loops,
- faces, shells,
- pointers from everyone to everyone.

Nice, but complicated, and often not necessary

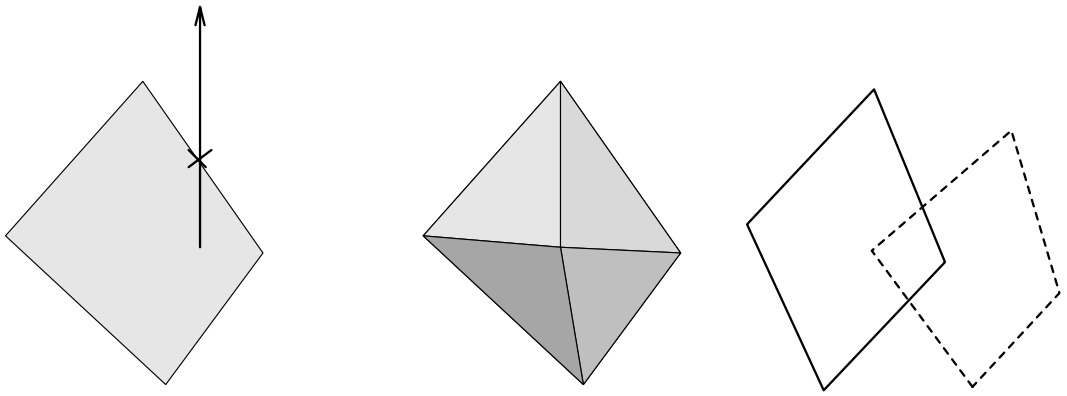
Set of Edges

A minimal rep.

Represent a polygon as { E }.

Possible ops:

- Point inclusion test,
- Area calculation,
- Intersection



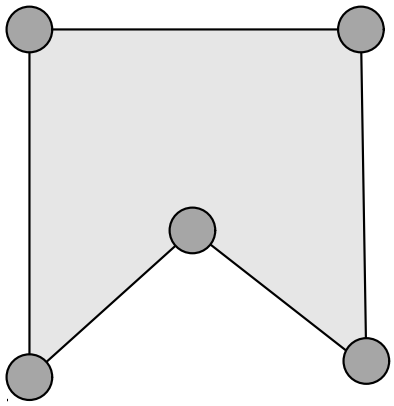
All easy

Advantages

- Simple,
- Fewer special cases,
- Faster to code,
- Faster to execute.

Set of Vertices

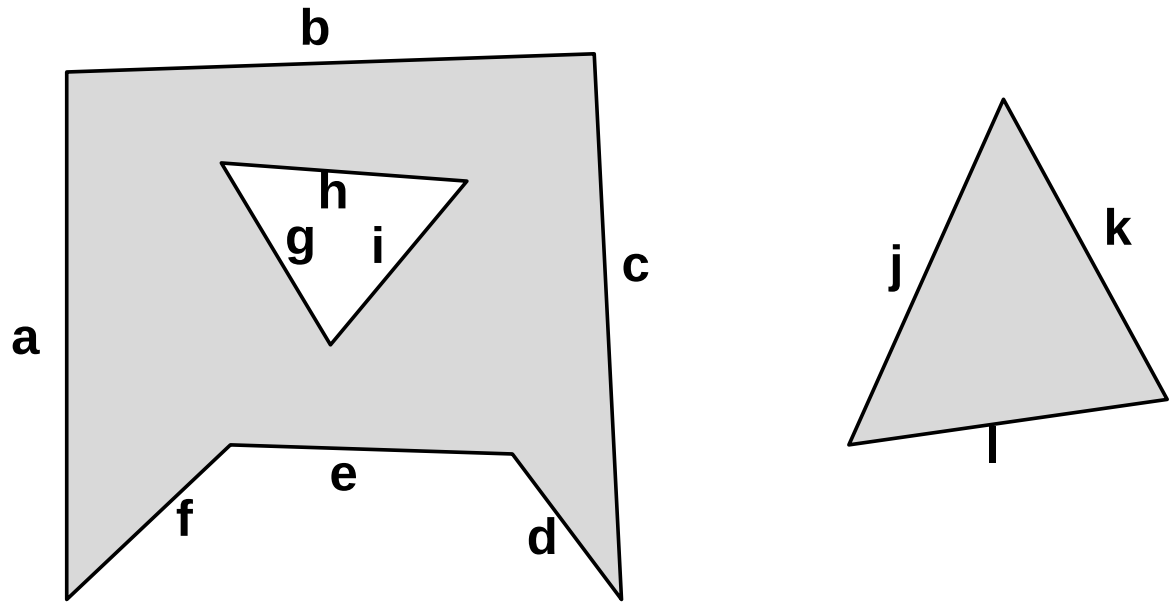
Too minimal – ambiguous



Combos of Halfplanes

Extend each edge of a polygon to a halfplane.

Represent a polygon('s interior) as a CNF formula of them.



$$P = abc(d+e+f)(g+h+i) + jkl$$

Extend this to Polyhedra?

No.

Guibas proved extra planes may be needed.

Aside — 2D and 3D Differences

- $\exists\infty$ number of regular polygons *(not polyhedra).*
- Given two equal-area polygons, one can be dissected into a finite number of pieces, then reassembled into the other. *($\not\forall$ polyhedra).*
- $\exists$ a square decomposable into smaller, different, squares. *($\not\exists$ cube).*
- All polygons are decomposable into triangles by adding only edges. *(Not always for polyhedra into tetrahedra by adding only faces).*

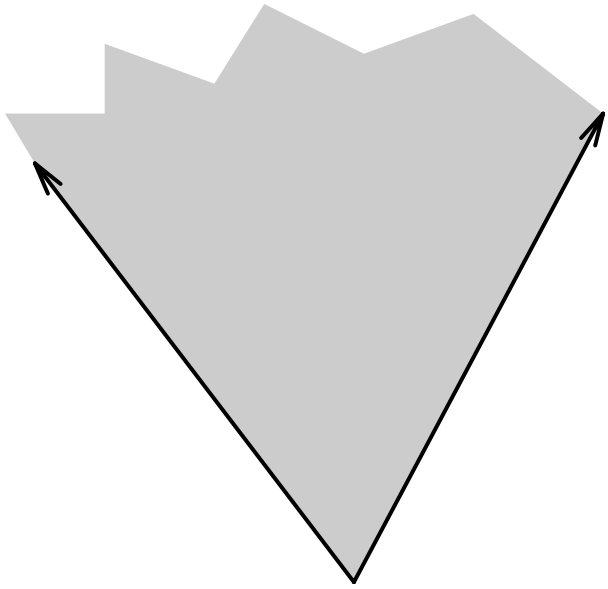
2d vs 3d – ctd

- For polygons, all such decomps have the same number of triangles. *(Not always for polyhedra*.
- Every polygon has every interior point visible from some vertex. *(∄ polyhedra.)*
- A 2-D Voronoi diagram's complexity is linear. *(3D: quadratic)*
- Given two convex polygons, ∃ an edge of one that separates them. *(Not always for polyhedra's faces)*

Augment the Vertex Info

Store the local neighborhood

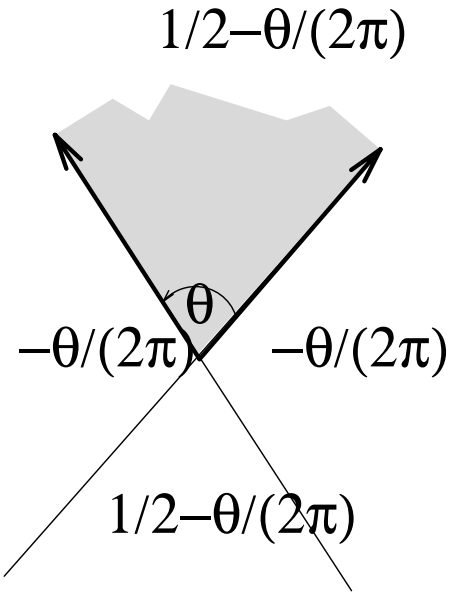
- The directions that the edges leave,
- Which sector is interior.



How to use this?

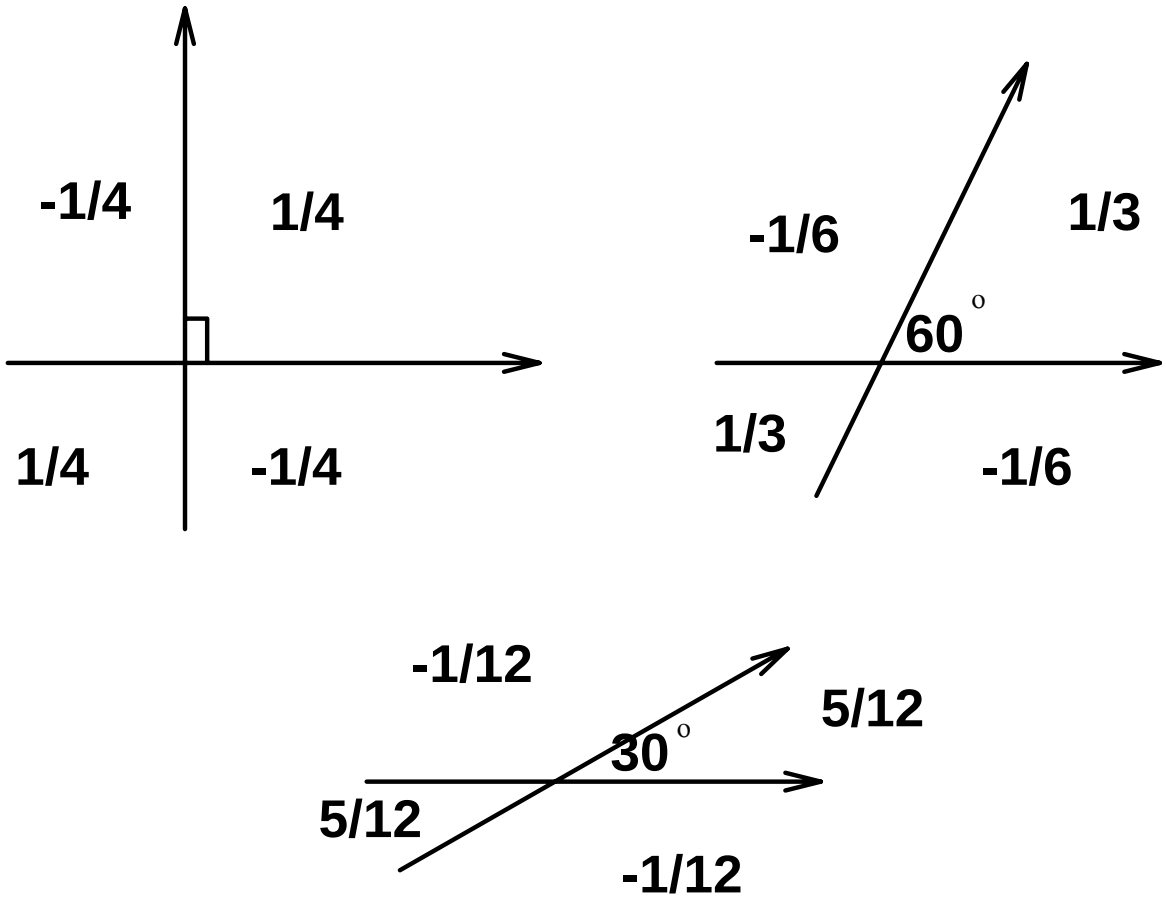
Define a χ Function

on all points in the plane depending on how they relate to the vertex neighborhood.



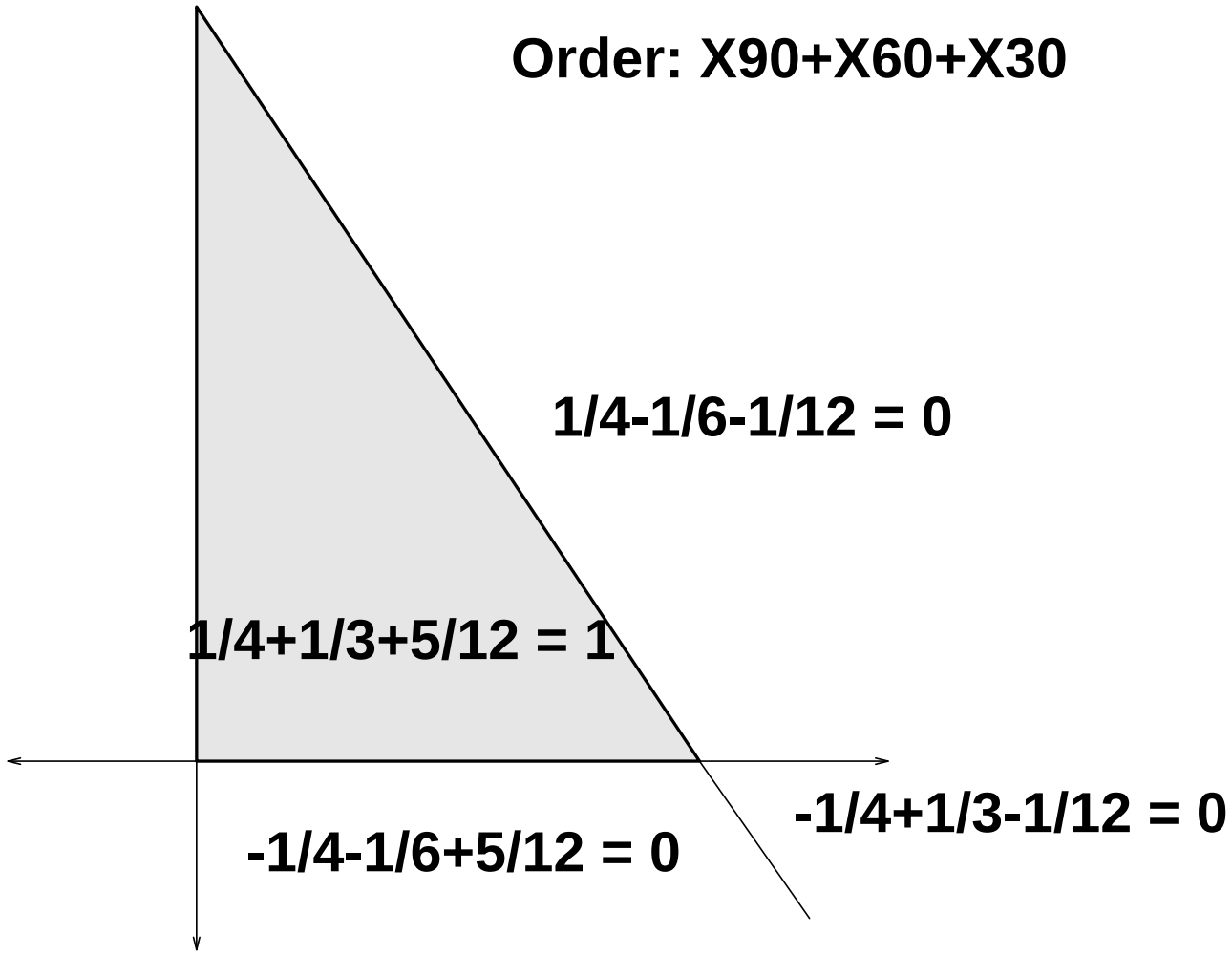
χ: Point $\Rightarrow$ Weight

Examples



Now add up the χ -functions for all the vertices of a polygon.

χs Summed Over Vertices of a Triangle

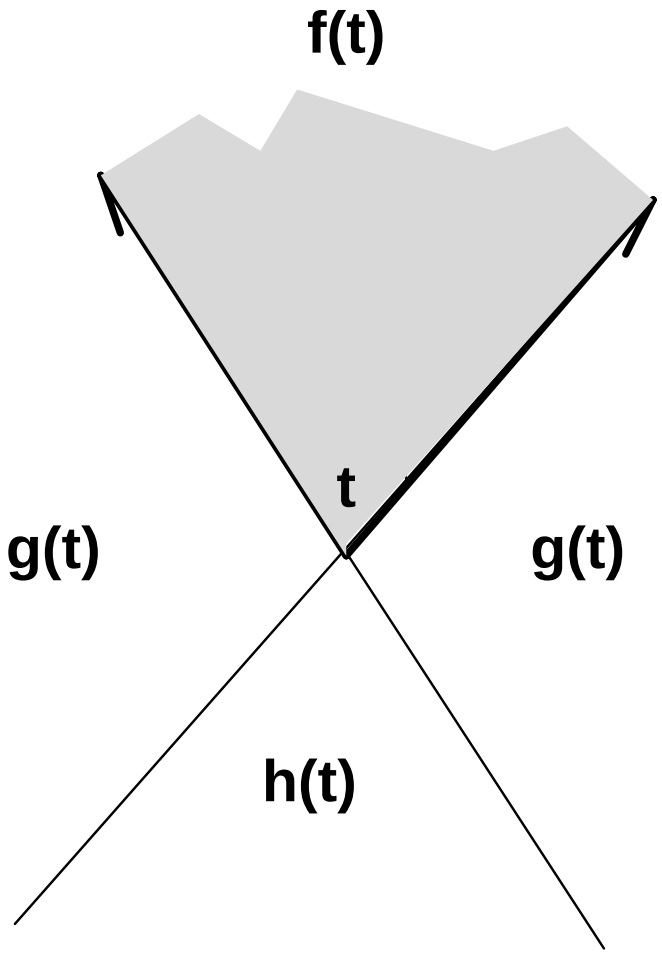


We have a characteristic function for the triangle. (0: out, 1:in).

This can be extended to all polygons, including concave with multiple components and holes.

Discovery

- Assume that such a thing might exist, with unknown weights.



- Examine limiting cases. From an a-b-c triangle:

  1. f(a)+f(b)+f(c)+1,

  2. f(a)+g(b)+g(c)=0,

  3. h(a)+g(b)+g(c)=0

- Therefore:

  1. f(a)=h(a) *(unexpected!)*

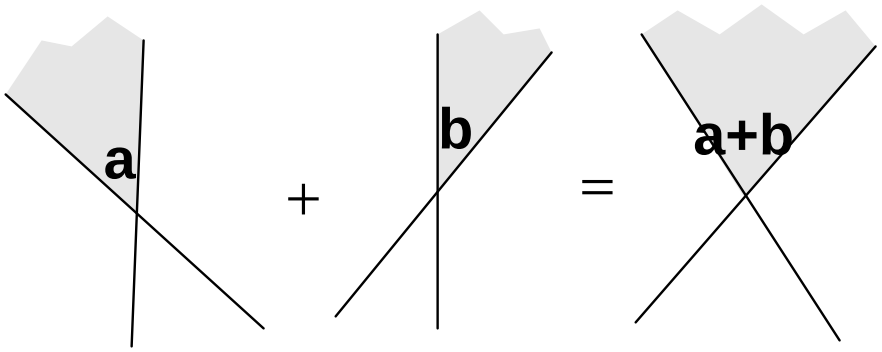
  2. f(a)=1/2+g(a)

- etc...

This discovers the only possible definition of χ.

Proof

Proof that this definition works by induction on the number of vertices.



E.g. f(a)+g(b) = g(a)+f(b) =f(a+b)

Note: There is a different set of weights for angles $> \pi$.

Point Inclusion Test

Given a point and a polygon, compare the point to each vertex, calculate χ, and sum. $0 \Rightarrow$ out. $1 \Rightarrow$ in.

Area Calculation

$$K = \int_{p \epsilon E^2} \sum_{v \epsilon V} \chi_v(p) dp$$

.

$$K \stackrel{?}{=} \sum \int \chi_v(p) dp$$

Unfortunately, no.

Use a limiting weight function

$$K_v{}' \triangleq \lim_{R \to \infty} \int_{p \epsilon E^2} w_R(p) \chi_v(p) dp$$

E.g., W_R is a box of size R.

$$K = \sum_v K_v{}'$$

Details

$$A_1 \triangleq (R^2 - v_x^2)(\tan\alpha_1 - \tan\alpha_2)$$

χ_1 is the weight of the point $(0,\infty)$, χ_2 the weight of $(\infty,0)$.

$$K_v{}' = A_1\chi_1 + (4\rho R^2 - A_1)\chi_2 = A_1(\chi_1 - \chi_2) + 4\rho R^2\chi_2$$

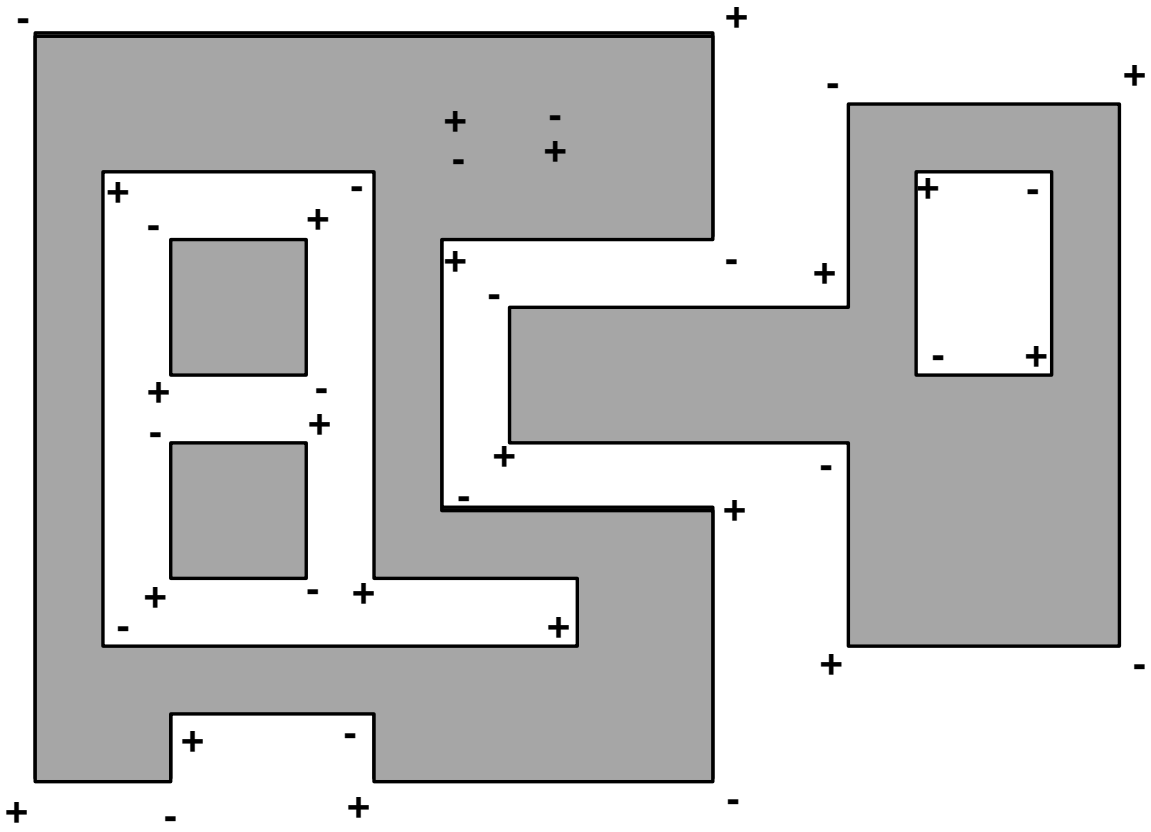
$$= \begin{cases} (R^2 + {v_x}^2)(\tan\alpha_1 - \tan\alpha_2) - \frac{2\phi}{\pi}\rho R^2 & \text{if } \phi < \pi \text{ \& the point } (0,\infty) \text{ has positive weight, i.e. } \alpha_1 < \pi < \alpha_2 \text{ or } \alpha_1 < -\pi < \alpha_2 \\ -(R^2 + {v_x}^2)(\tan\alpha_1 - \tan\alpha_2) - 2(1 - \frac{\phi}{\pi})\rho R^2 & \text{if } \phi < \pi \text{ \& the point } (0,\infty) \text{ has negative weight} \\ -\frac{1}{2}(R^2 + {v_x}^2)(\tan\alpha_1 - \tan\alpha_2) - 4(1 - \frac{\phi}{2\pi})\rho R^2 & \text{if } \phi > \pi \text{ \& the point } (0,\infty) \text{ has positive weight} \\ \frac{1}{2}(R^2 + {v_x}^2)(\tan\alpha_1 - \tan\alpha_2) - 4(\frac{1}{2} - \frac{\phi}{2\pi})\rho R^2 & \text{if } \phi > \pi \text{ \& the point } (0,\infty) \text{ has negative weight} \end{cases}$$

Ignore terms tending to ∞ since we know that for well-formed polygons they will cancel.

$$K_v' = \begin{cases} {v_x}^2(\tan\alpha_1 - \tan\alpha_2) & \text{if } \phi < \pi \text{ \& the point } (0,\infty) \text{ has positive weight} \\ -{v_x}^2(\tan\alpha_1 - \tan\alpha_2) & \text{if } \phi < \pi \text{ \& the point } (0,\infty) \text{ has negative weight} \\ -\frac{1}{2}{v_x}^2(\tan\alpha_1 - \tan\alpha_2) & \text{if } \phi > \pi \text{ \& the point } (0,\infty) \text{ has positive weight} \\ \frac{1}{2}{v_x}^2(\tan\alpha_1 - \tan\alpha_2) & \text{if } \phi > \pi \text{ \& the point } (0,\infty) \text{ has negative weight} \end{cases}$$

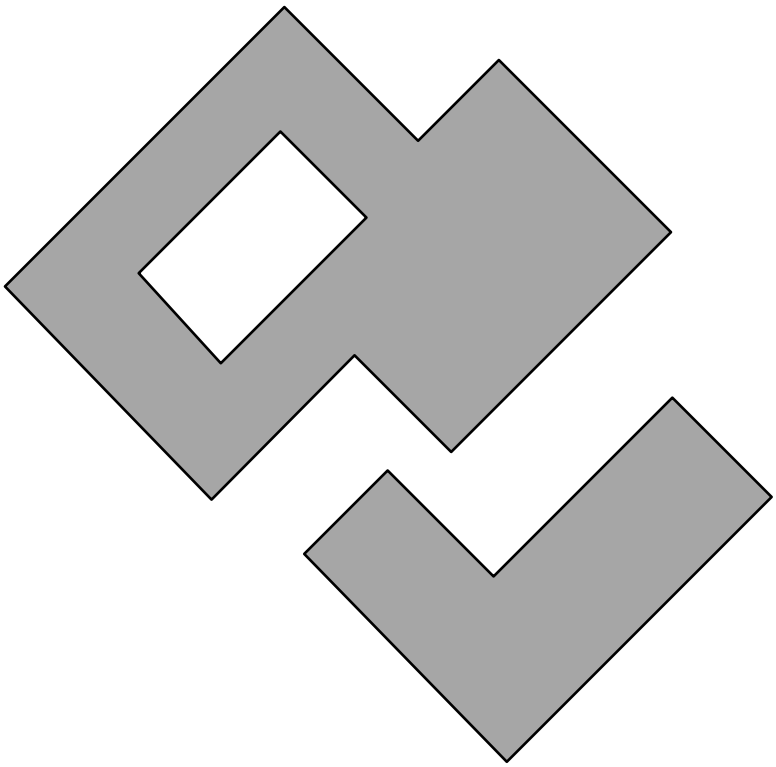
Then $K = \sum_v K_v'$

Example — Rectilinear Polygons



Area$= \sum x_i y_i s_i$

Nonuniqueness



Area= $\sum x_i^2 s_i = \sum y_i^2 s_i$

and linear combos thereof.

Error Checking

This assumes the input is topologically correct, or else garbage.

Use this to check the correctness: Apply a random rigid transformation and recalculate the area.

Other Mass Properties

Moments of inertia etc. can be calculated equally easily.

3D

Three-dimensional vertices have more complicated topologies in general.

However, this extends easily to trihedral vertices.

Cusp-Based Data Structure

Subdivide the vertex neighborhood into vertex-edge adjacencies.

Data element is now (V, T, N), one per adjacent vertex & edge.

V = a vertex

T = the unit tangent along an edge

N = the unit normal to edge, pointing into polygon.

Polygon = { (V,N,T) }.

The various triples referring to one vertex, or to one edge, are not associated in any way in the data structure.

Mass Properties Are Easy

- total edge length, $L = -\sum V \cdot T$
- polygon area, $A = 1/2 \sum V \cdot T \; V \cdot N$
- moments of inertia, etc, similarly.
- Point inclusion test also possible.

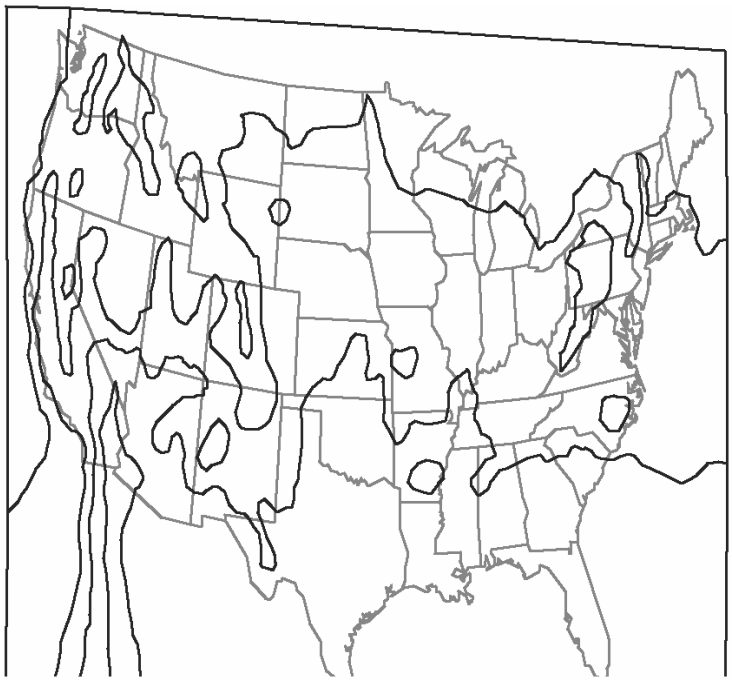
Application — Properties of Intersected Polygons

Wanted: the area of the union of 25,000 rectangles (extracted from a VLSI database).

Use a 750×750 uniform grid to find the 262,058 edge intersections, then apply these formulae.

Time = 34 CPU seconds on a Sun 4/280.

Wanted: the areas of all the output polygons from overlaying two maps.

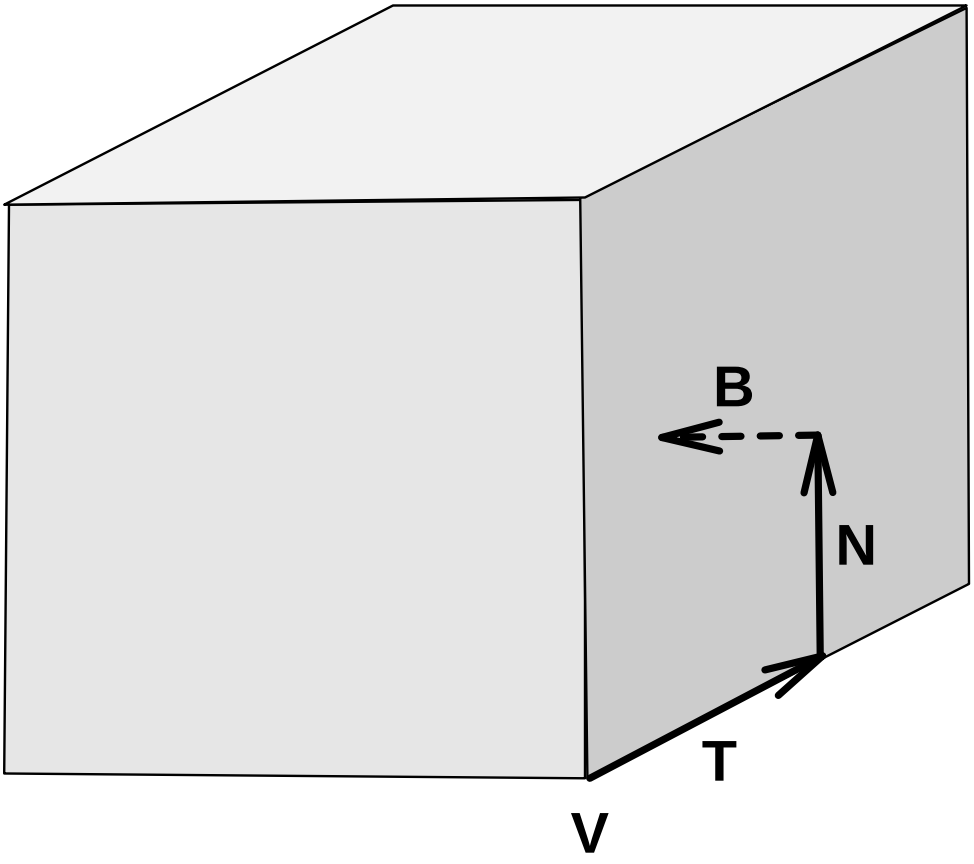


	Verts	Edges	Faces
Coterminous US States	1081	915	49
Isotherms	920	884	6

Finding the output edges is twice as difficult as finding only the vertices, so these local-topological formulae are quite useful.

Polyhedra

- Polyhedron = { (V,N,T,B) }
- One tuple per vertex-edge-face adjacency. (i.e. 6 per trihedral vertex).
- V = Cartesian coordinates of vertex.
- N = unit tangent vector along edge.
- T = unit normal vector to edge, in plane of face.
- B = unit vector normal to N and T, pointing into solid.


B
N
T
V

Properties

- Total edge length, counted once per face $= \sum V \cdot T$

  w/o knowing the number of edges, or the length of any one edge

- Total face area $= 1/2 \sum V \cdot T \; V \cdot N$

  w/o knowing the number of faces

- Volume $= \sum V \cdot N \; V \cdot T \; V \cdot B$

  w/o knowing the number of connected components

Implementation

Map overlay area calculation on a Sun 4 IPC (25 MHz, 10 MIPS) using a 32×32 uniform grid. US coterminous states against 10F isotherms.

Operation	CPU time (secs)
Read map data	1.83
Scaling reformatting	0.08
Calc input areas	0.12
Inserting into grid	0.07
Intersecting edges	0.08
Locating vertices	0.18
Accumulating output areas	0.15
Printing stats	0.22
Total time	2.73
Total excluding I/O	<0.68

Humongous Example

	Verts	Edges	Faces	Chains
US counties	55068	46116	2985	8941
Hydrography	76215	69835	2075	6380

Operation	CPU time (secs)
Read map:	99.32
Scale vertices:	1.03
Extract edges:	2.38
Calculate areas:	3.65
Make grid:	1.28
Add map to grid:	8.60
Intersect edges:	6.50
Locate map 0 points in map 1:	5.83
Locate map 1 points in map 0:	8.17
Accumulate areas:	14.35
Print areas:	17.23
TOTAL TIME=	168.35

- I/O (116 sec) is slower than everything else combined (52 sec)!
- Using a 500×500 uniform grid.
- 168,781 edge pairs tested to find 11,207 intersections.

Weaknesses

Some operations are difficult with local topological data structures:

- shading,
- boolean combinations.

Nevertheless, they are possible, since the global topology is implicit, and can be determined.

Conclusion

We need much less info than expected to represent geometric objects for most operations. Local topological data structures allow

- faster *and*
- simpler,
- parallelizable,
- useful for large databases

data structures and programs.