APPLIED aspects of COMPUTATIONAL GEOMETRY

Arrangements, 3D

Dan Halperin
School of Computer Science
Tel Aviv University

Overview

- Collins-style decomposition
- output-sensitive construction of the vertical decomposition of arrgs of triangles
- extension to surface patches
- arrgs of planes

Arrg complexity, reminder

the maximum combinatorial complexity of an arrangement of n well-behaved surfaces in R^3 is $O(n^3)$; there are such arrangements whose complexity is $\Omega(n^3)$

Arrgs of triangles, Collins decomposition

- complexity K, $\Omega(n) \leq K \leq \theta(n^3)$
- simple representation: Collins-style decomposition
- complexity of the decomposition and construction
- implementation

Vertical decomposition

- reminder, planar arrgs
- arrgs of triangles?

Vertical decomposition, arrgs of triangles

- input triangles $T = \{t_1, t_2, \ldots, t_n\}$
- we assume here general position
- A(T), the arrg
- boundary edge (1 triangle), intersection edge (2 triangles)
- vertical visibility of points
- vertical wall W(e)
- $V_1(T)$ = A(T) + W(e)'s for triangle boundary edges

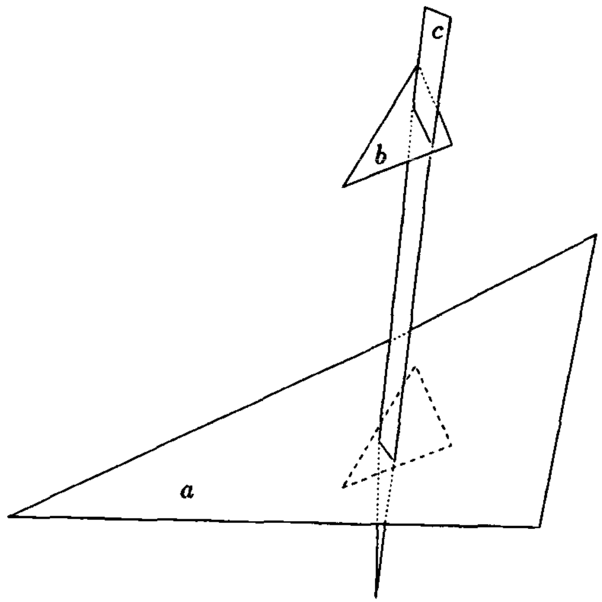
General position for triangles in space

- no two edges of distinct triangles intersect
- no vertex of a triangle is contained in another triangle
- no triangle is vertical (parallel to the z-axis)
- etc

<back>

The shape of cells in $V_1(T)$

- Q: Are they convex?
- A: No! they are not even simply connected



- so we need further refinement

VD of triangles, Step 2

- $V_2(T) = V_1(T) + W(e)$'s for intersection edges
- in $V_2(T)$ the cell are cylindrical but can still have complex shapes

VD of triangles, the final refinement

- $V_3(T) = V_2(T)$ + trapezoidal decomposition of each face of $V_2(T)$ on each triangle, extended to 3D cells (details later)

- in $V_3(T)$ each cell is a convex prism and bounded by at most six facets

The complexity of $V_i(T)$

- let K denote the complexity of A(T), namely $|A(T)| = K$
- $|W(e)|$ of a boundary edge is $O(n\alpha(n))$, for a total of $O(n^2\alpha(n))$ (there are 3n boundary edges), so $|V_1(T)| = O(n^2\alpha(n)+K)$
- similarly, $|V_2(T)| = O(n^3\alpha(n)+K) = O(n^3\alpha(n))$
- step 3 does not increase the asymptotic complexity, so $|V_3(T)| = O(n^3\alpha(n))$

The complexity of $V_i(T)$, cont'd

- we know better $|V_3(T)| = O(n^{2+\varepsilon}+K)$
- in particular $|V_3(T)| = \theta(n^3)$

  such tight bound known only for planes (trivial) and triangles

- there are arrgs of triangles with $K=O(n)$ and $|V_2(T)| = \theta(n^2\alpha^2(n))$

  this is an unfortunate property of VD in 3-space

Output sensitive algorithm

- input: n triangles in R^3 in general position
  $T = \{t_1, t_2, \ldots, t_n\}$
- output: $V_3(T)$ represented as a graph
  $G=G(C,E)$
  - the nodes C: the cells (vertical prisms) of the decomposition
  - the edges E: connect neighboring cells
- goal: keep it simple!

  work as much as possible in 2D spaces

The bounding simplex

- we assume that the triangles are bounded inside a simplex whose faces are special triangles in T:
  - they violate the general position assumption
  - we are only interested in one side of each: the side that faces the interior of the simplex

  this way we do not have to handle unbounded features

- in VD we view each triangle $t_i \in T$ as two-sided: it has a top side t_i^+ and a bottom side t_i^-

The grand scheme

- algorithm's steps follow the definitions of $V_1(T)$, $V_2(T)$, $V_3(T)$
- we first compute the features of $V_1(T)$
- then add the features of $V_2(T)$
- then add the features of $V_3(T)$
- and only then construct the output representation

Computing the features of $V_1(T)$

- constructing the envelopes of W(e) for a triangle boundary edge
- the curves $\Gamma(t_i^+)$ defining the top arrangement of the triangle side t_i^+
- similarly for t_i^-
- their arrangements A_i^* (either top or bottom)
- running time $O(n^2 \log n)$ for the envelopes, and
- $O(|V_1(T)| \log n)$ in total

Computing the features of $V_1(T)$, remarks

- can be computed slightly faster but will be subsumed by other parts of the algorithm
- in the process we add cross pointers:
  - every intersection edge that appears in one arrangement A_i^* of triangle t_i, points to the other triangle t_j involved in the intersection
  - every envelope edge points to the boundary edge on whose envelope it appears
- at the end of step one the entire $V_1(T)$ is connected, the boundary of each 3D cell is connected

  with still rather convoluted cell shapes

Computing the features of $V_2(T)$

- sweeping a plane P_x parallel to the yz-plane over $V_1(T)$
- $A_x := P_x \cap V_1(T)$
- claim: A_x is a convex subdivision
- major difficulty in computing $V_2(T)$: identifying pairwise visibilities of intersection edges inside a 3D cell
- why not compute W(e) for all intersection edges?

Computing the features of $V_2(T)$, cont'd

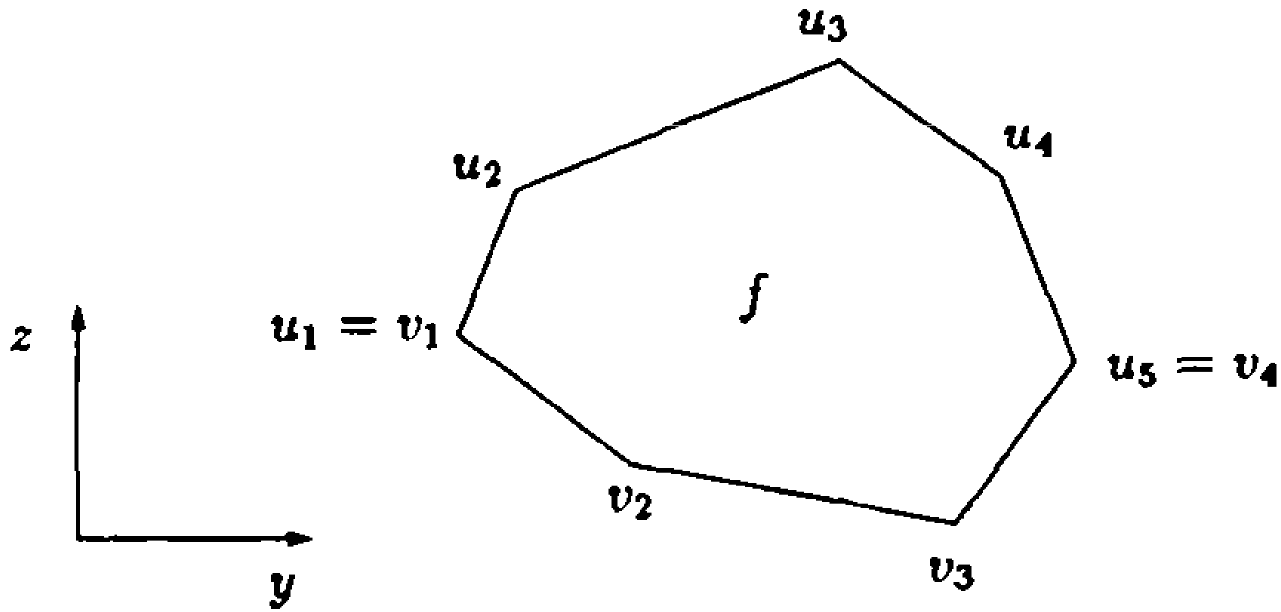
- goal of this step: mark on A_i^+ intersection edges visible from t_i^+ when looking upwards and similarly mark on A_i^- intersection edges visible from t_i^- when looking downwards

Computing the features of $V_2(T)$, the sweep

- A_x changes continuously between events
- the events: the vertices of $V_1(T)$ plus one new type of events: vertical visibility of intersection edges
- Data structures:
  - events queue Q, sorted by x-coordinate
    operations: insert, delete, fetch minimum
  - 2D status structure

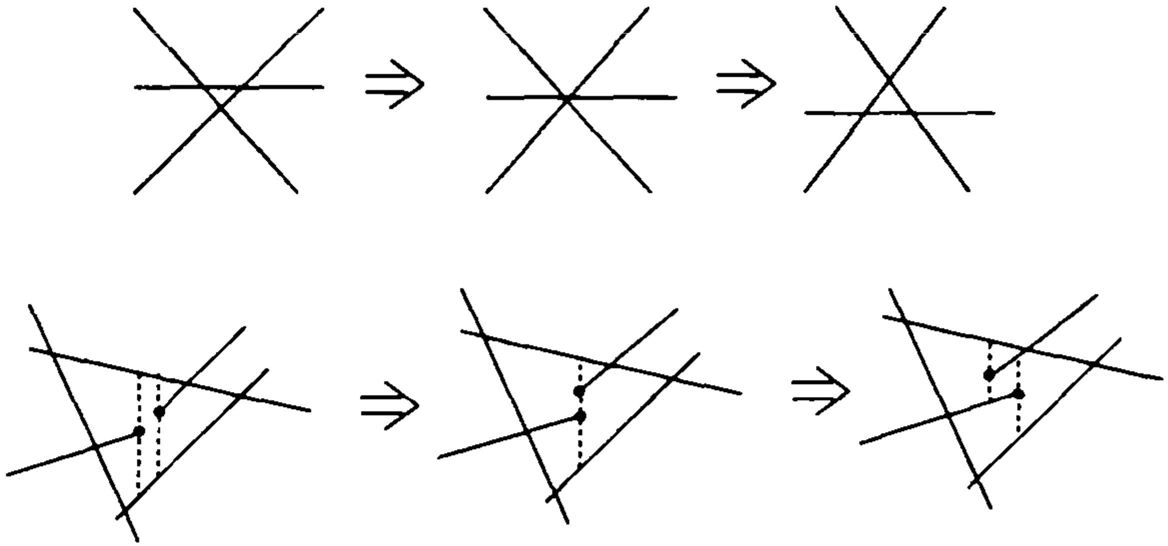
The status structure

- a collection of convex faces represented as two list of vertices for the upper and lower chains respectively

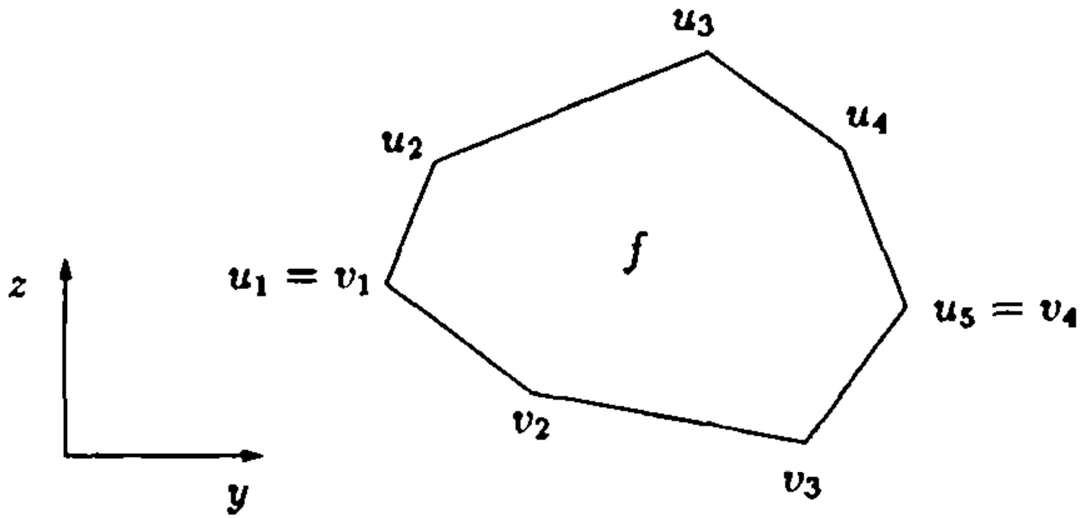


The events

- a vertex may (dis)appear
- an edge may (dis)appear
- a face may (dis)appear



Vertical visibility events



- how to detect them
- how to handle them
- actual vs. false events

Computing the features of $V_2(T)$, wrap up

- $V=|V_2(T)|$ events, each handled at O(log n) time
- at the end of this step, each cell has cylindrial shape, with one top and one bottom triangle
- the cells can still have complex shape
- as in the previous steps we have not yet built a 3D structure but rather collected features of dimension <3
- denote the arrg on t_i^* after Step 2 by B_i^*

Computing the full decomposition $V_3(T)$

- project each B_i^* onto the xy-plane and apply y-vertical decomposition
- lift the added walls back to B_i^*
- extend each of the added segments into a z-vertical wall inside its 3D cell: this is $V_3(T)$
- each cell has at most six facets, with a single triangle at the top and at the bottom
- this step takes $O(V \log n)$ time
- overall time of the algo $O(n^2 \log n + V \log n)$

Representing the full decomposition

- Reminder, $V_3(T)$ represented as a graph G=G(C,E)
  - the nodes C: the prisms of the decomposition
  - the edges E connect neighboring cells that share an edge either on the floor or on the ceiling
- work the prisms from say the B_i^+ 's, need to record top triangle info for each vertex

Completing same-triangle ceiling/floor connections, altenatives

- for each ceiling vertex record the face on the floor above it, and for each floor vertex record the face on the ceiling below it
- construct a point-location structure on each Bi* (within the algorithm's asymptotic running time); one crossing will cost O(log n)
- propagate floor/ceiling info thru cells (details only known in 2D) such that each cell has a constant number of neighbors across triangle

Output-sensitive algorithm, take II (lighter overhead)

- same input, same output
- as before, sweep a plane parallel to the yz-plane over the triangles T keeping
  - an event queue Q, oredered by x-coordinates
  - the set of convex faces of $V_1(T)$ on A_x (but without first computing $V_1(T)$)
- as before, the sweep produces the features of $V_2(T)$; from that point on we resort to the previous algorithm

Algorithm, take II, the difference

- do not compute (almost) anything in advance
- the only events inserted into Q before the sweep: triangles corners
- everything else is detected on the fly
- Problem: how will we know where to insert a new triangle when it appears
- Solution: maintain a dynamic point location (PL) structure for A_x: $O(\log n)$ update time, $O(\log^2 n)$ query time

Algorithm, take II, analysis

- every operation during the sweep, including the update of the PL structure, but without PL queries, can be carried out in O(log n) time
- every operation above can be charged to a feature of $|V_2(T)|$, and no feature gets charged more than a constant number of time, as we assumed general position
- the n PL queries take $O(\log^2 n)$ time each
- the total running time is $O(n \log^2 n + V \log n)$

Extension: well-behaved surface patches

- a patch is an xy-monotone portion of an algebraic surface of constant maximum degree
- when projected onto the xy-plane it is bounded by a constant number of algebraic curves of constant maximum degree
- the patches are in general position

Extension to arrgs of surface patches, cont'd

- instead of convex, the faces of A_x are now y-monotone
- need to add extra vertices on boundary curves at points where their projection onto the xy-plane has y-vertical tangency
- the total running time is the same: $O(n \log^2 n + V \log n)$
- the first version runs in $O(n\lambda_q(n)\log n + V \log n)$ for an appropriate q

Alternative with fewer cells: Partial vertical decomposition

- $V_1(T)$ as before

  cells still rather unwieldy
- extra flood faces on A_x when it meets
  - the first corner of a triangle
  - the last corner of a triangle
  - a middle corner (face on the side that does not contain the triangle)
  - intersection of a boundary edge with another triangle
- all cells are convex
- partial VD for triangles has $\theta(n^3)$ complexity

Arrgs of planes

- single cell, envelope
- vertical decomposition
- our output-sensitive algo runs in near-optimal time
- optimal-time alternative: incidence-graph representation and incremental construction
- alternative decomposition: bottom-vertex

References

- for general references see the end of the presentation on 2D arrangements
- output-sensitive algorithm for VD of 3D arrgs: [de Berg-Guibas-H '96] Vertical decompositions for triangles in 3-space, DCG
- output-sensitive algorithm with lighter overhead, partial 3D decomposition: [Shaul-H ‘02] Improved construction of vertical decompositions of 3D arrangements, SoCG

References, cont'd

- arrangements of planes, the incidence graph:

  [Edelsbrunner '87]

  Chapter 7 of *Algorithms in Combinatorial Geometry*, Springer

- requirements from well-behaved surface patches

  [Agarwal-Sharir '00]

  Section 2 of Chapter 2 "Arrangements and Their Applications" in *NH Handbook of Computational Geometry*

THE END