Polygon Partitioning

Lecture03

History of Triangulation Algorithms

Year	Complexity	Reference
1911	$O(n^2)$	Lennes (1911)
1978	$O(n \log n)$	Garey et al. (1978)
1983	$O(n \log r)$, r reflex	Hertel & Mehlhorn (1983)
1984	$O(n \log s)$, s sinuosity	Chazelle & Incerpi (1984)
1988	$O(n + nt_0)$, t_0 int. triangs.	Toussaint (1990)
1986	$O(n \log \log n)$	Tarjan & Van Wyk (1988)
1989	$O(n \log^* n)$, randomized	Clarkson, Tarjan & Van Wyk (1989)
1990	$O(n \log^* n)$, bnded. ints.	Kirkpatrick, Klawe & Tarjan (1990)
1990	$O(n)$	Chazelle (1991)
1991	$O(n \log^* n)$, randomized	Seidel (1991)

Outline

- Monotone polygon
  - Triangulation of monotone polygon
- Trapezoidal decomposition
- Decomposition in monotone mountain
- Convex decomposition

Polygon Partitioning

Monotone Partitioning

Monotone polygons

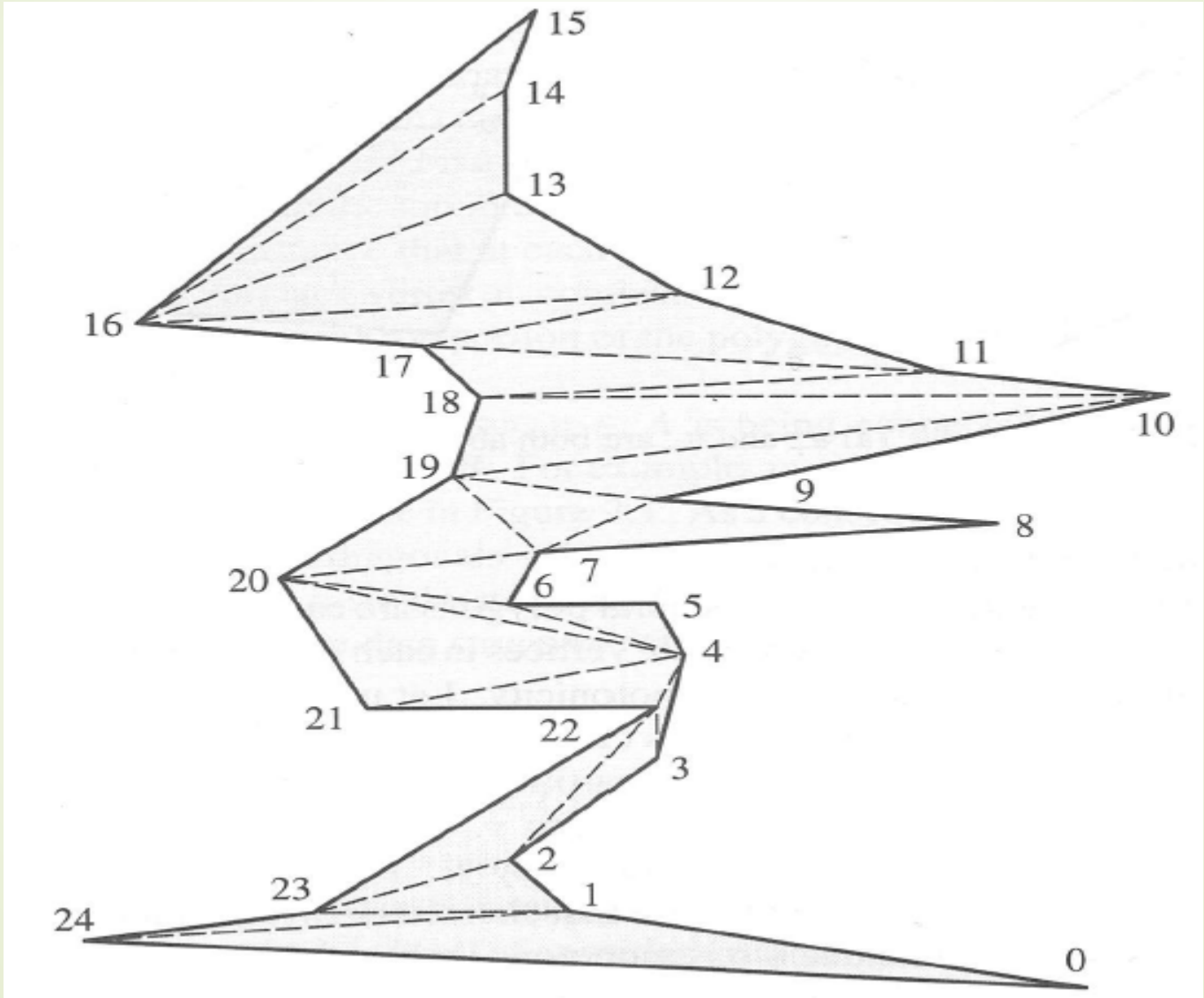
Definition

- A polygonal chain C is **strictly monotone** with respect to L' if every line L orthogonal to L' meets C in at most one point.
- A polygonal chain C is **monotone** if L has at most one connected component
  - Either empty, a single point, or a single line segment
- A polygon P is said to be **monotone** with respect to a line L if ∂P can be split into two polygonal chains A and B such that each chain in monotone with respect to L

Monotone polygons

- The two chains should share a vertex at either end.
- Figure 2.1
  - A polygon monotone with respect to the vertical line
  - Two monotone chains
    - A = (v_0 ,…, v_{15}) and B= (v_{15},…, v_{24}, v_0)
  - Neither chain is strictly monotone (two horizontal edges – v_5v_6 and $v_{21}v_{22}$)

Monotone polygons



Monotone polygons

Properties of Monotone Polygons

- The vertices in each chain are sorted with respect to the line of monotonicity (LoM)
  - Let y axis be the LoM
- Can be split into two chains
  - Left and right chains if y-axis is the LoM
  - Find a highest vertex, a lowest vertex and partition the boundary into two chains

Monotone polygons

- An **interior cusp** of a polygon is
  - a reflex vertex v
  - v's adjacent vertices v_- and v_+ are either both at or above, or both at or below, v
- Interior cusp can't have both adjacent vertices with the same y coordinate as v

Lemma

- If a polygon P has no interior cups, then it is monotone
- Lack of interior cusp implies strict monotonicity

Monotone Partitioning

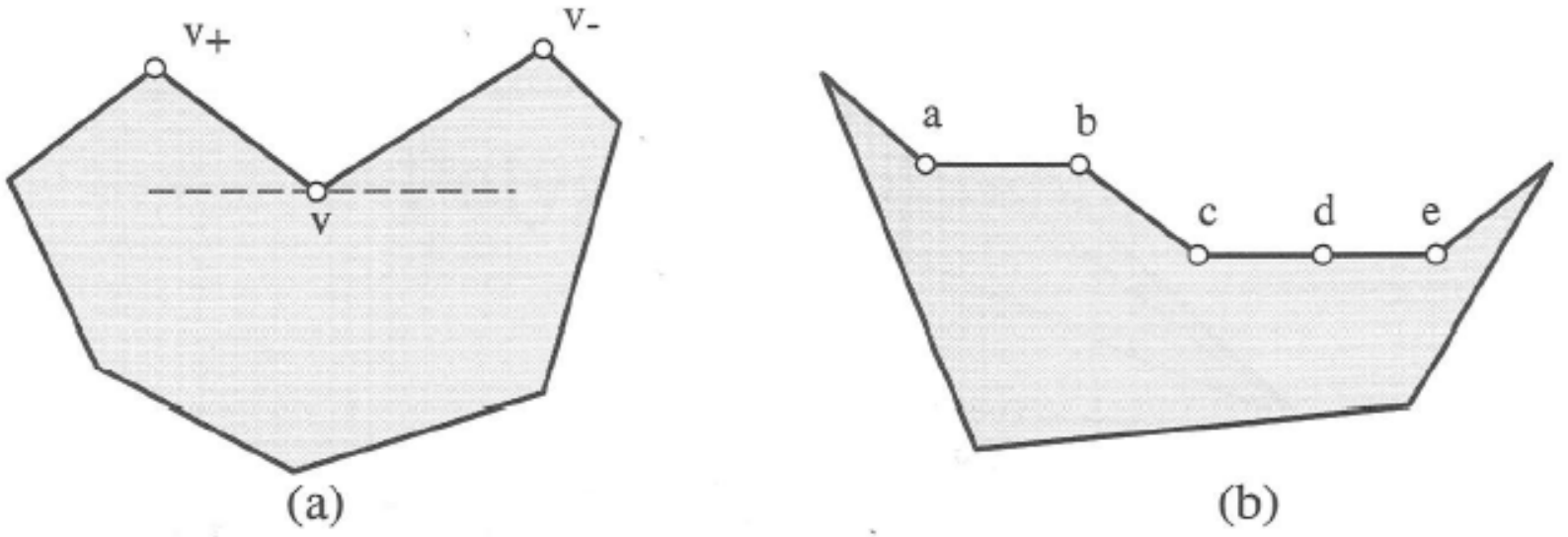


FIGURE 2.2 Interior cusps: (a) v_+ and v_- are both above v; (b) a, c, and e are interior cusps; b and d are not.

Triangulating a Monotone Polygon

- Monotone polygons are so special that they are easy to triangulate

- Hint of algorithm
  - Sort the vertices from top to bottom
  - Cut off triangles from the top in a “greedy” fashion

Triangulating a Monotone Polygon

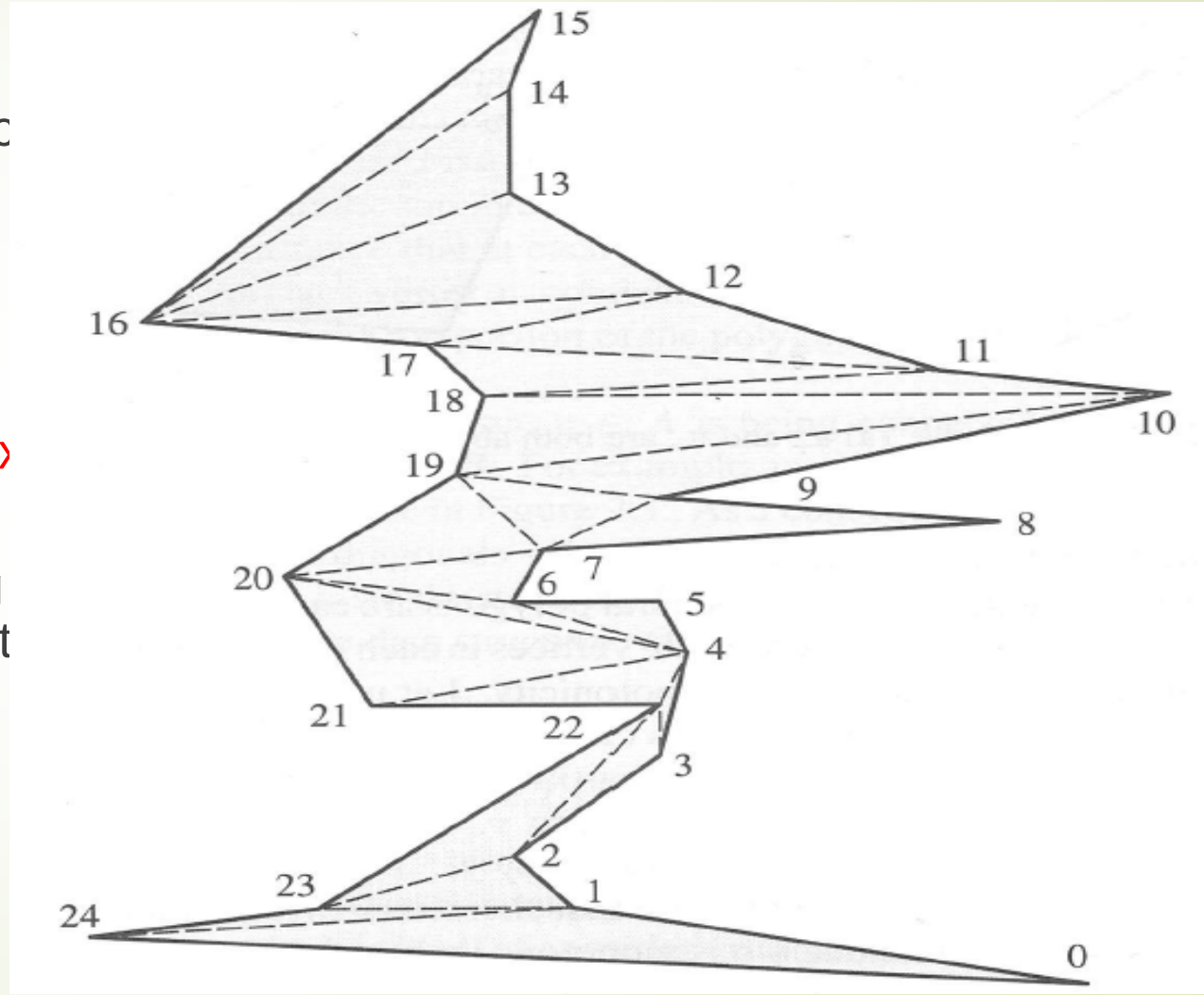
Algorithm

- Sorting all vertices on decreasing y-coordinate
- Push v_1 and v_2 onto the stack S
- for j ← 3 to n ← 1
  - if v_j and vertex on top of S are on different chains
    - Add diagonals from u_j to all vertices in S
  - if v_j and vertex on top of S are on same chains
    - Add diagonals from u_j to vertices in S until you cannot do so

- Linear time triangulation

Example

- For example, v_{16} is connec to (v_{14}, v_{13}, v_{12}) in the first iteration
- As a consequence, no visibility check is required
- The stack S holds the reflex chain above
- Between the linear sorting and this simple data struct $O(n)$ is achieved



Polygon Partitioning

Trapezoidalization

Trapezoidalization

- A **horizontal trapezoidalization** of a polygon is obtained by drawing a horizontal line through every vertex of the polygon.

- Pass through each vertex v the maximal horizontal segment s such that $s \subset P$ and $s \cap \partial P = v$

- Assumption: We only consider polygons whose vertices have unique y coordinates.

Trapezoidalization

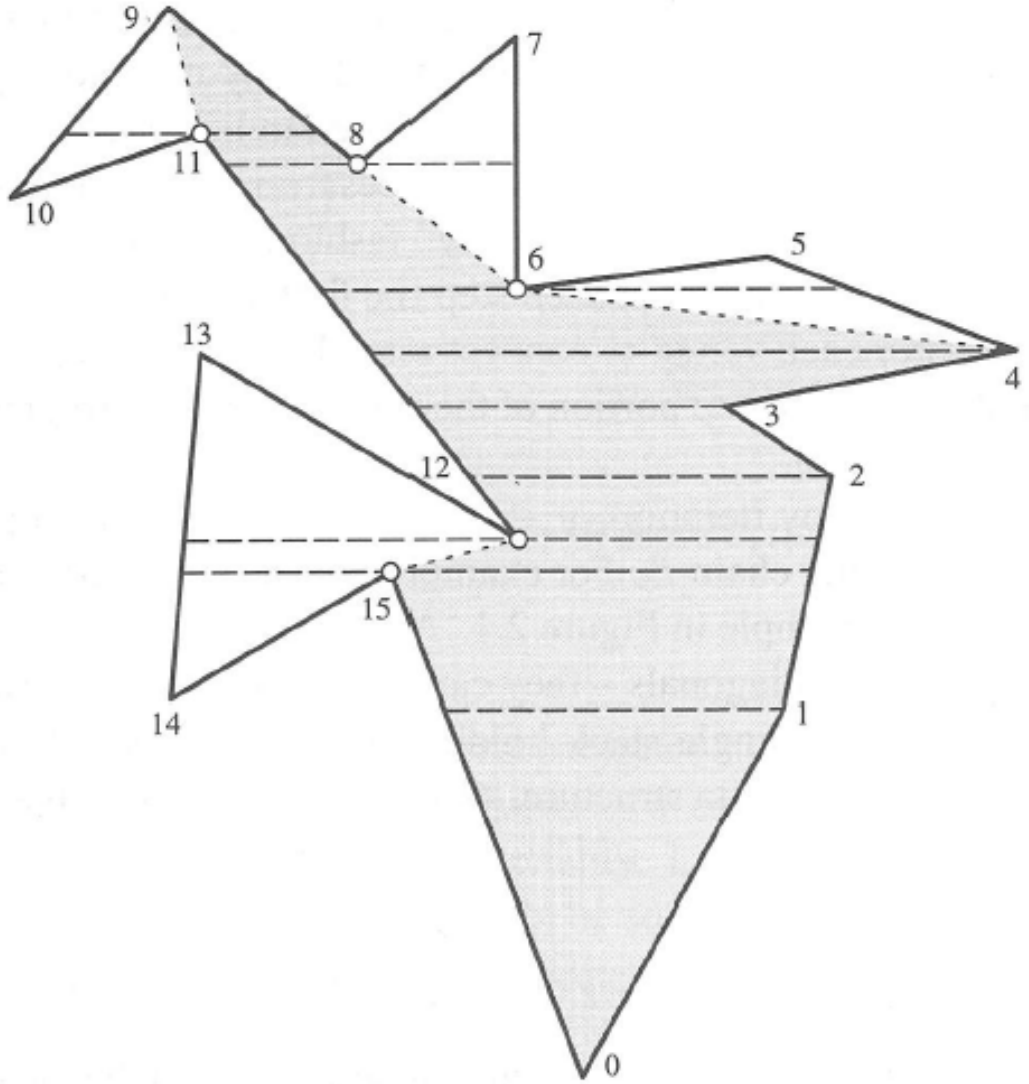
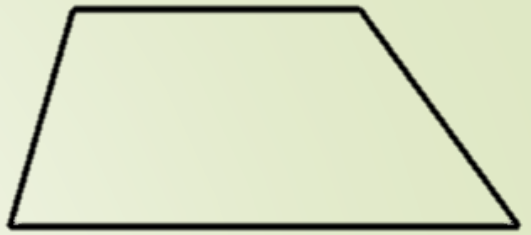


FIGURE 2.3 Trapezoidalization. Dashed lines show trapezoid partition lines; dotted diagonals resolve interior cusps (circled). The shaded polygon is one of the resulting monotone pieces.

Trapezoidalization



- A **trapezoid** is a quadrilateral with two parallel edges
- Vertices through which the horizontal lines are drawn **supporting vertices**
- Every trapezoid has exactly two supporting vertices (and at most 4 neighbors)
- The connection to monotone polygons
  - If a supporting vertex is on the interior of an upper or lower trapezoid edge, then it is an interior cusp

Trapezoidalization

- Monotone partition
  - Connect every interior vertex v to the opposing supporting vertex of the trapezoid v supports
  - Then, these diagonals partition P into monotonic pieces
  - For example, diagonal v_6v_4, $v_{15}v_{12}$, and so on in Figure 2.3

Plane Sweep

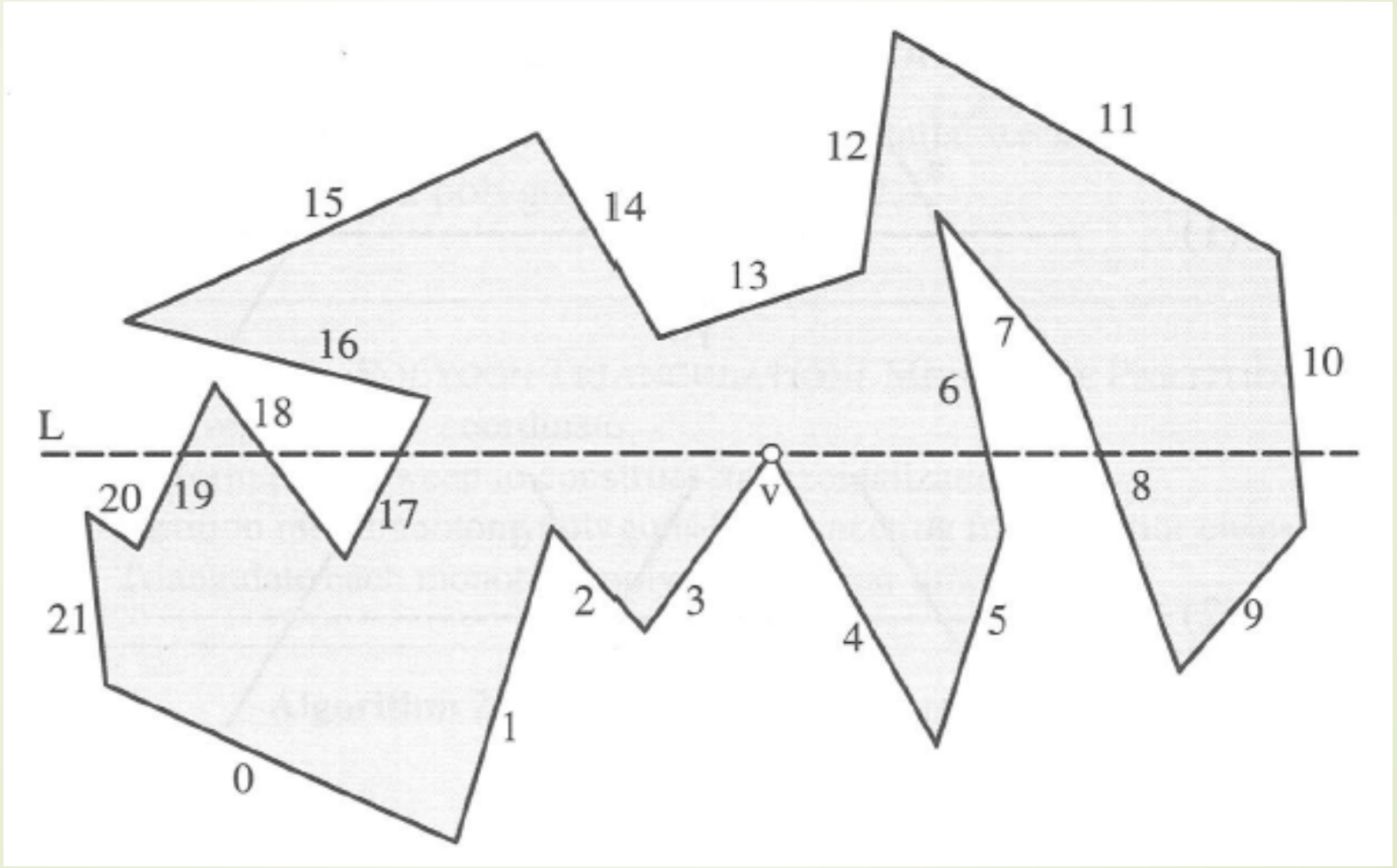
- Useful in many geometric algorithms
- Main idea is to “sweep” a line over the plane maintaining some type of data structure along the line
  - Sweep a horizontal line L over the polygon, stopping at each vertex
  - Sorting the vertices by y coordinate
- $O(n \log n)$ time

Plane Sweep

- Fine the edge immediately to the left and immediately to the right of v along L
  - A sorted list L of polygon edges pierced by L is maintained at all times
  - How to determine that v lies between e_{17} and e_6?

Plane Sweep

For the sweep line in the position L = (e_{19}, e_{18}, e_{17}, e_6, e_8, e_{10})



Plane Sweep

- Assume that e_i is a pointer to an edge and the vertical coordinate of v is y.
- Suppose we know the endpoints of e_i
- Then, we can compute the x coordinate of the intersection between L and e_i
- We can determine v's position by computing x coordinates of each intersection
- Time proportional to the length of L ($O(n)$)by a naive search from left to right
- With an efficient data structure, a height-balanced tree, the search require $O(\log n)$ time

Plane Sweep

- Updates at each event
- There are three types of event (Figure 2.5)
- Let v fall between edges a and b and v be shared by edges c and d
  - c is above L and d is below. Then delete c from L and insert d :
    - $(\ldots, a, c, b, \ldots) \Rightarrow (\ldots, a, d, b, \ldots)$
  - Both c and d are above L. Then delete both c and d from L :
    - $(\ldots, a, c, d, b, \ldots) \Rightarrow (\ldots, a, b, \ldots)$
  - Both c and d are below L. Then insert both c and d into L :
    - $(\ldots, a, b, \ldots) \Rightarrow (\ldots, a, c, d, b, \ldots)$

Plane Sweep

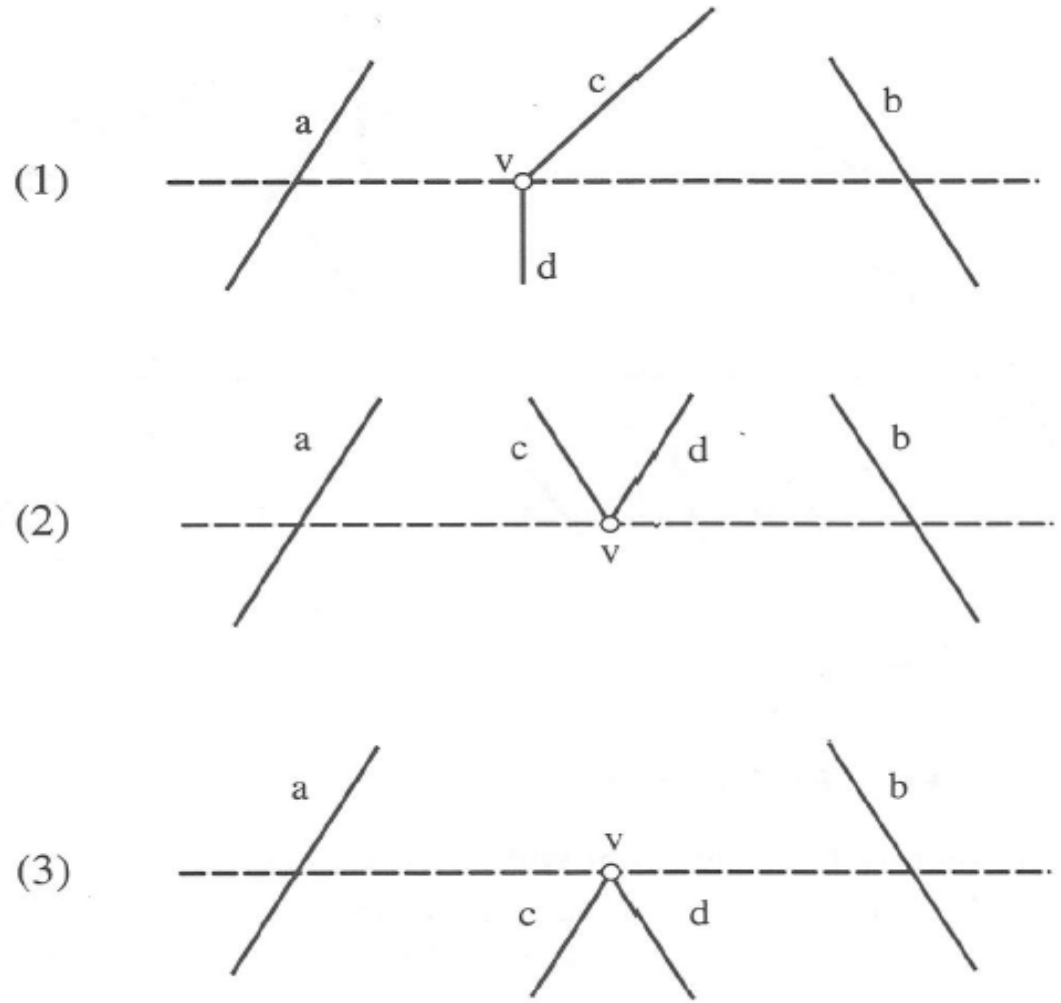


FIGURE 2.5 Sweep line events: (1) replace c by d; (2) delete c and d; (3) insert c and d.

Plane Sweep

- In Figure 2.4, the list L of edges pierced by L is initially empty, when L is above the polygon
- Then follows this sequence as it passes each event vertex

(e_{12}, e_{11})

$(e_{15}, e_{14}, e_{12}, e_{11})$

$(e_{15}, e_{14}, e_{12}, e_6, e_7, e_{11})$

$(e_{15}, e_{14}, e_{13}, e_6, e_7, e_{10})$

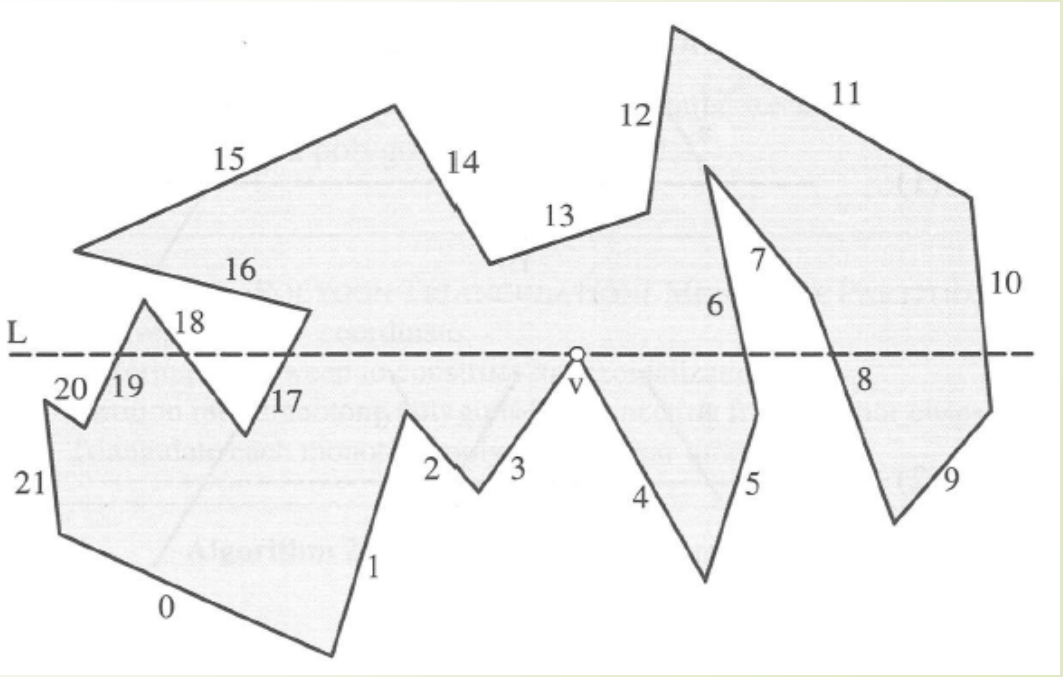
$(e_{16}, e_{14}, e_{13}, e_6, e_7, e_{10})$

$(e_{16}, e_6, e_7, e_{10})$

$(e_{16}, e_6, e_8, e_{10})$

$(e_{19}, e_{18}, e_{16}, e_6, e_8, e_{10})$

$(e_{19}, e_{18}, e_{17}, e_6, e_8, e_{10})$.



Triangulation in $O(n \log n)$

Algorithm: POLYGON TRIANGULATION: MONOTONE PARTITION
Sort vertices by y coordinate.
Perform plane sweep to construct trapezoidalization.
Partition into monotone polygons by connecting from interior cusps.
Triangulate each monotone polygon in linear time.

Algorithm 2.1 $O(n \log n)$ polygon triangulation.

Polygon Partitioning

Partition into Monotone Mountains

Monotone Mountains

- A **monotone mountain** is a monotone polygon with one of its two monotone chains a single segment, the **base**.
- Note that both end points of the base must be convex

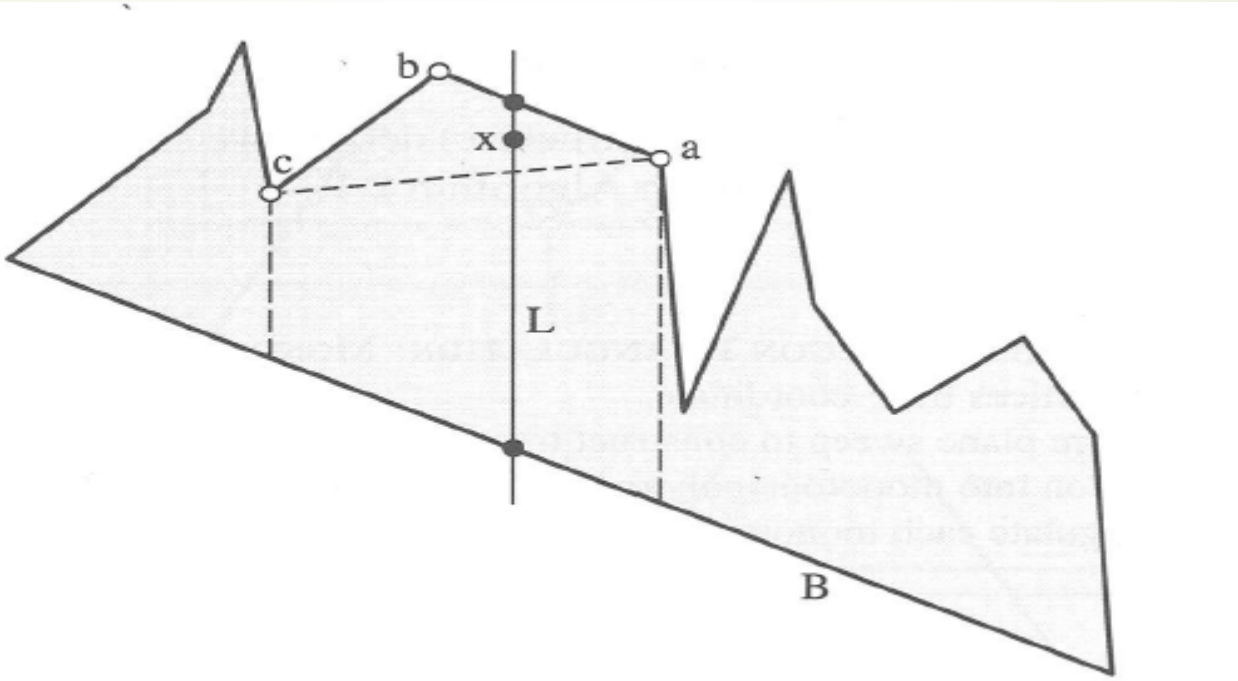


FIGURE 2.6 A monotone mountain with base B; b is an ear tip.

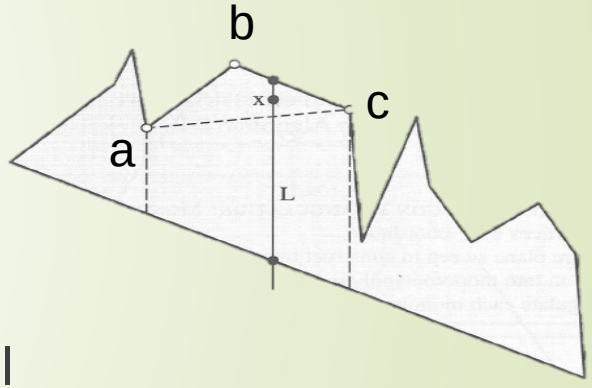
Triangulating a Monotone Mountain

Algorithm: TRIANGULATION OF MONOTONE MOUNTAIN
Identify the base edge.
Initialize internal angles at each nonbase vertex.
Link nonbase strictly convex vertices into a list.
while list nonempty do
    For convex vertex b, remove $\triangle abc$.
    Output diagonal ac.
    Update angles and list.

Algorithm 2.2 Linear-time triangulation of a monotone mountain.

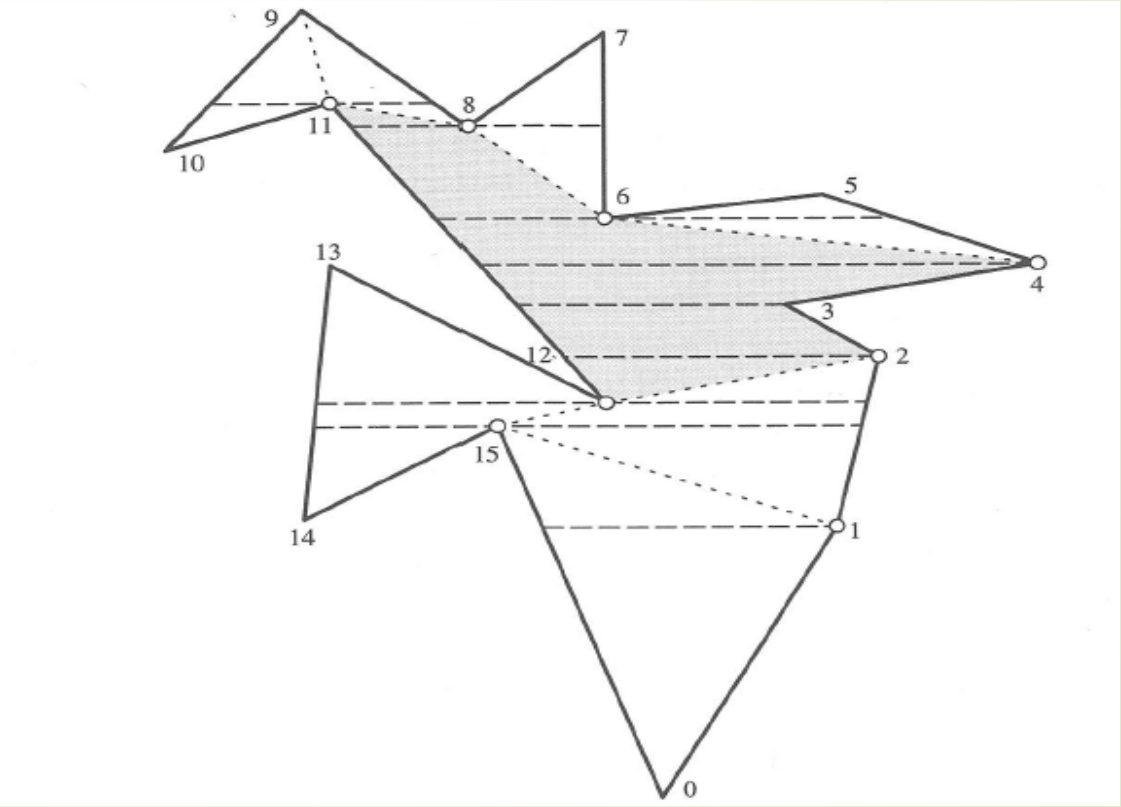
Triangulating a Monotone Mountain

- Linear time algorithm
- The base is identified in linear time
  - The base endpoints are extreme along the direction of monotonicity
  - Leftmost and rightmost vertices
- The "next" convex vertex is found without a search in constant time
  - Update the convexity status using stored internal angles of vertices
    - By subtracting from a and c's angles in $\triangle abc$
  - Update the list with each ear clip



Trapezoidalization of Monotone Mountains

- Build a monotone mountain from trapezoids abutting on a particular base edge, for example $v_{11}v_{12}$



Convex Partitioning

- Two goals
  - Partition a polygon into as few convex pieces as possible
  - Do so as fast as possible
- Compromise on the number of pieces
- Find a quick algorithm whose output size is bounded with respect to the optimum
- Two types of partition may be distinguished
  - By diagonals
  - By segments

Bounding Size of Convex Partition

Lemma (Chazelle)

- Let Φ be the fewest number of convex pieces into which a polygon may be partitioned. For a polygon of r reflex vertices, $\lceil r/2 \rceil + 1 \leq \Phi \leq r + 1$
- Proof
  - Drawing a segment that bisects each reflex angle results in a convex partition
  - The number of pieces is $r + 1$ (Figure 2.10)
  - At most two reflex angles can be resolved by a single partition segment (Figure 2.11)
  - This results in $\lceil r/2 \rceil + 1$ convex pieces

Bounding Size of Convex Partition

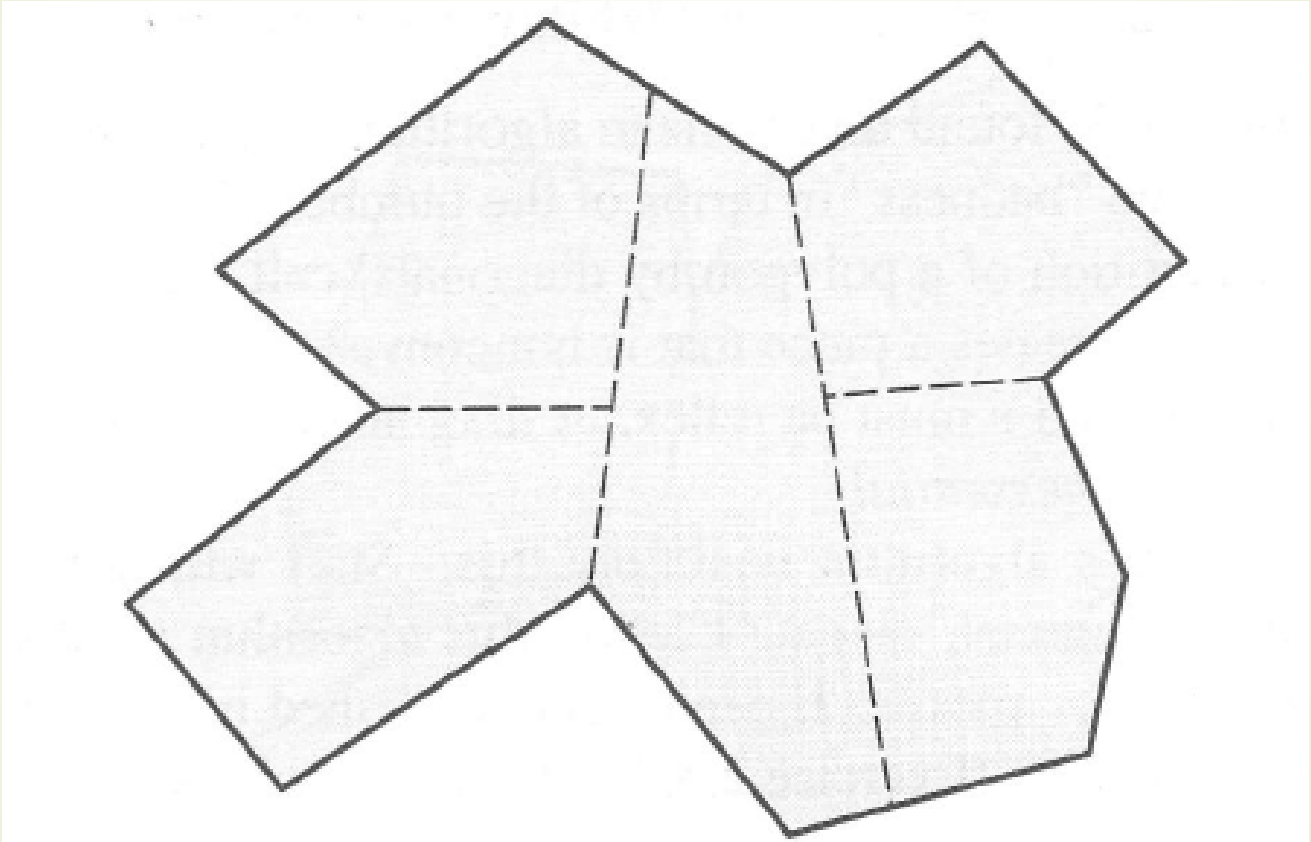


FIGURE 2.10 $r + 1$ convex pieces: $r = 4$; 5 pieces.

Bounding Size of Convex Partition

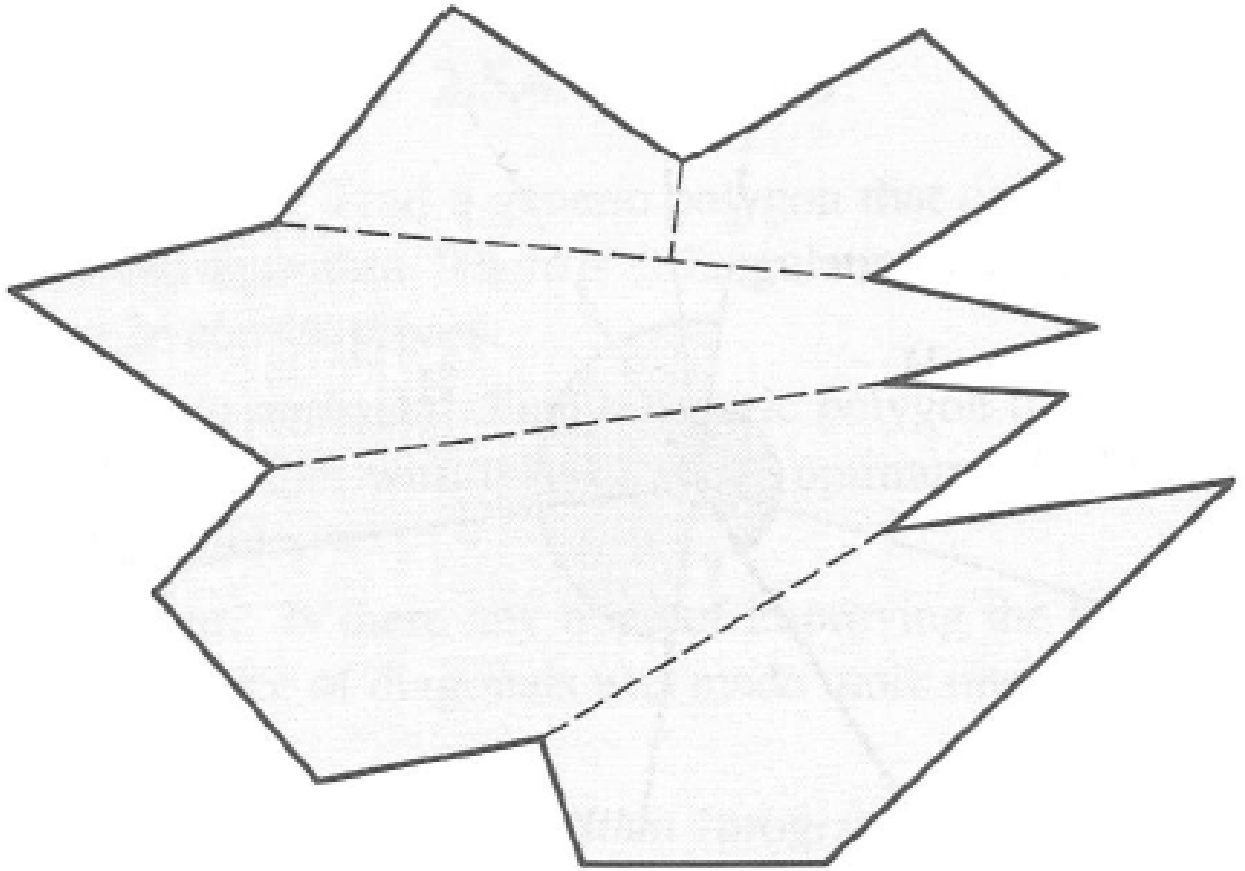


FIGURE 2.11 $\lceil r/2 \rceil + 1$ convex pieces: $r = 7$; 5 pieces.

Hertel and Mehlhorn Algorithm

- A very clean algorithm that partitions with diagonals quickly
  - has bounded "badness" in terms of the number of convex pieces
- A diagonal d *is* **essential** for vertex v if removal of d makes v non-convex
- The algorithm
  - start with a triangulation of P
  - remove an inessential diagonal
  - repeat

Optimal Convex Partitions

- Finding a convex partition optimal in the number of pieces is much more time consuming than finding a suboptimal one
  - Greene's algorithm runs in $O(r^2n^2) = O(n^4)$ time
  - Keil's algorithm improved it to $O(r^2n\log n) = O(n^3\log n)$ time
  - Both employ **dynamic programming**

- The problem is even more difficult if the partition **may be formed with arbitrary segments**
  - Chazelle solve this problem in his thesis with an intricate $O(n + r^2) = O(n^3)$ algorithm

Optimal Convex Partitions

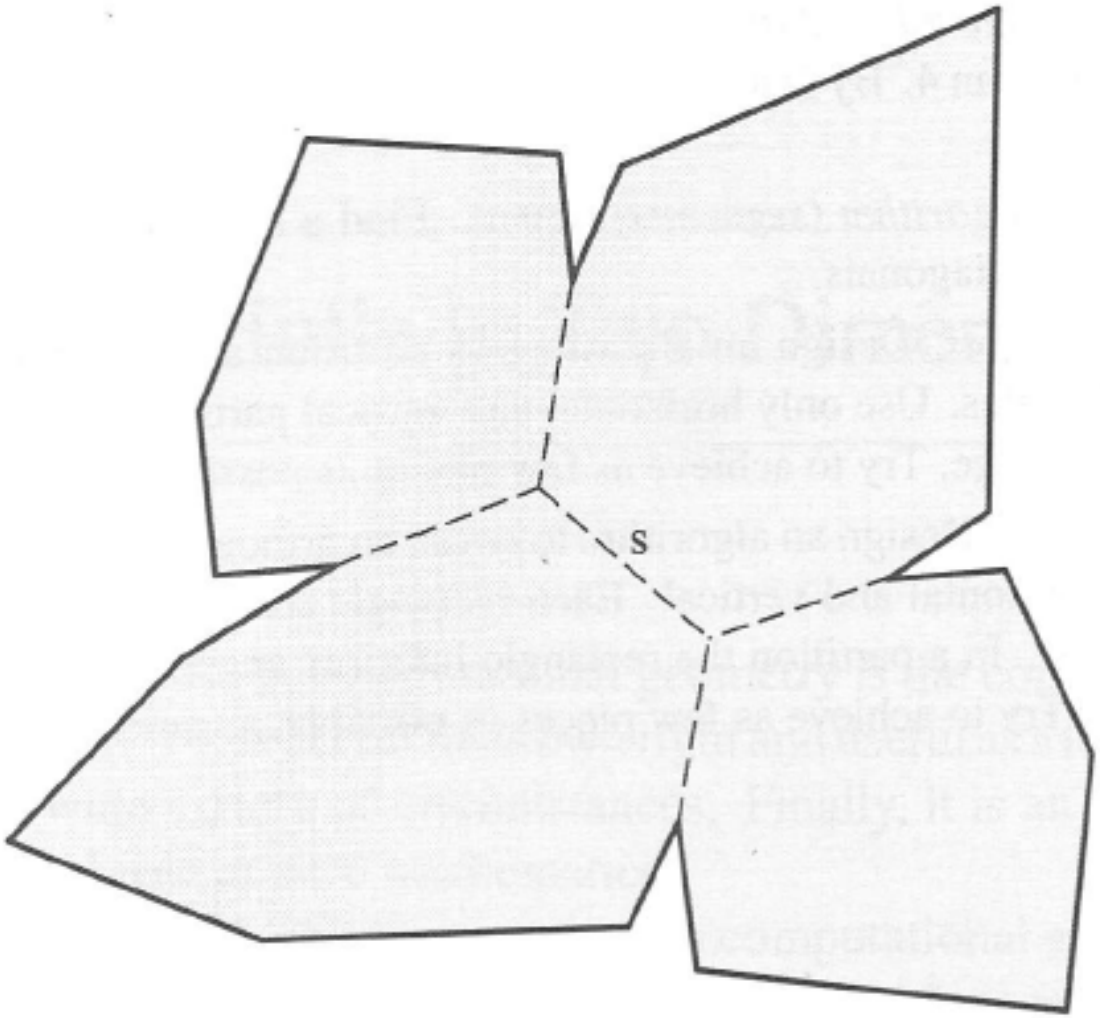
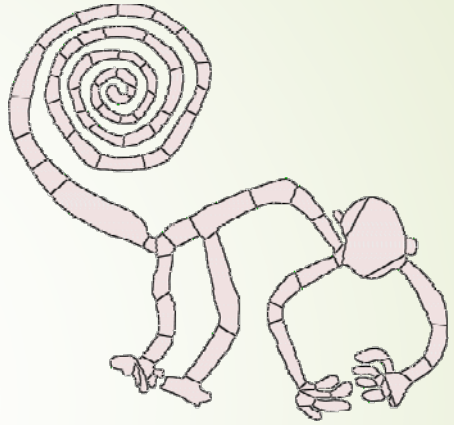


FIGURE 2.13 An optimal convex partition. Segment s does not touch ∂P.

Approximate Convex Decomposition (ACD)

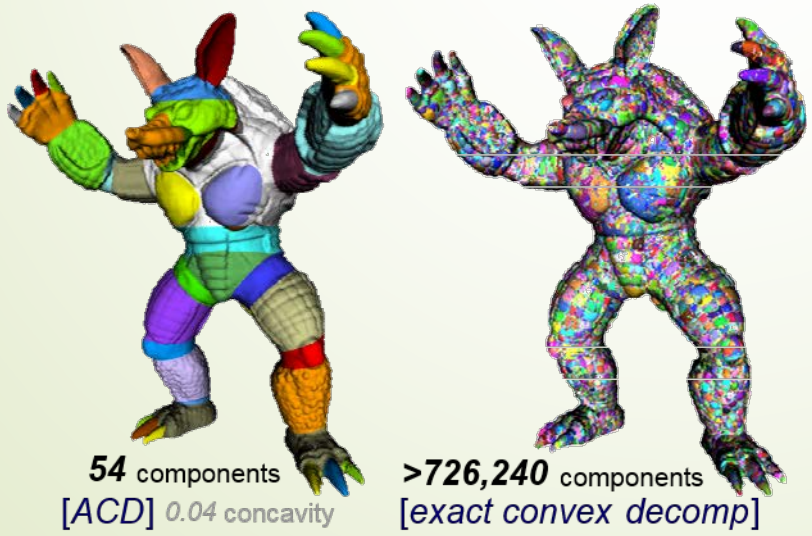
ACD

- All sub-models will have tolerable *concavity*
- Convex decomposition is useful but
- can be costly to construct
- may result in unmanageable number of components



Benefits of ACD

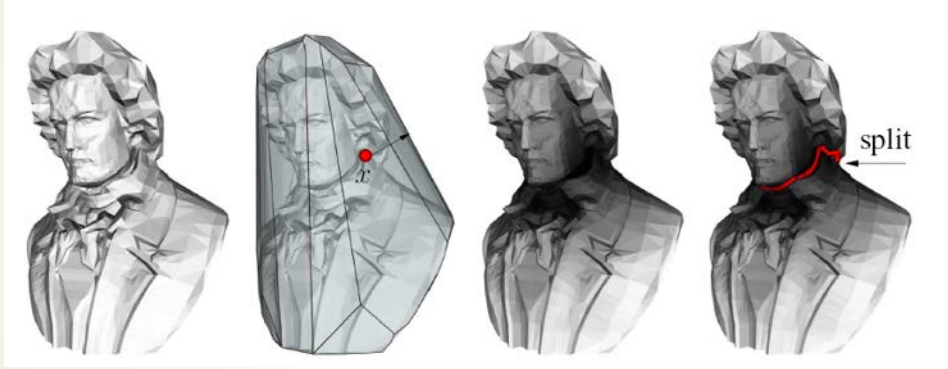
1. Number of sub-models is significantly less



2. Reveal structural features

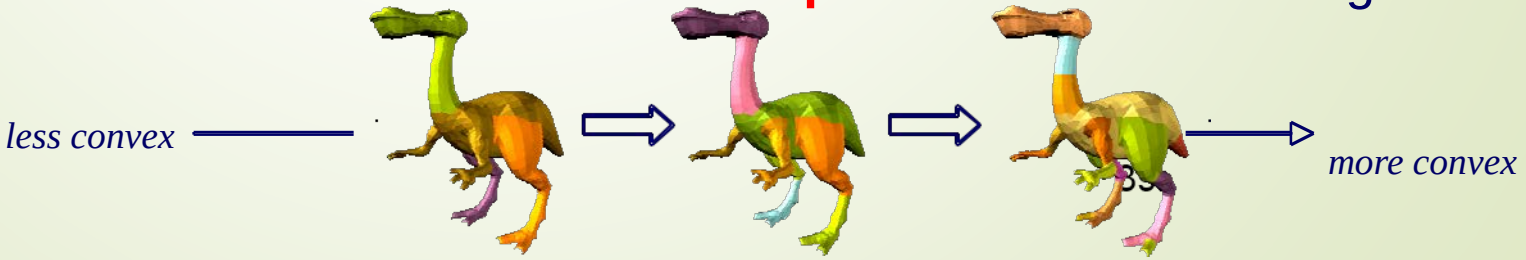
Approximate Convex Decomposition (ACD)

- How does it work?



Input model | Measure concavity | Shaded using its concavity | Decompose at areas with high concavity

Sub-models form a nice **hierarchical representation** of the original model



Conclusion

- Polygon decomposition
  - decompose to monotonic polygon/mountain
  - trapezoidal decomposition
  - convex decomposition