Polygon triangulation and the art gallery problem

Dan Halperin, Tel Aviv University

Figures taken from:

CGAA: Chapter 3 in Computational Geometry Algorithms and Applications, de Berg et al

L4vK: Lecture 4 in Computational Geometry, Triangulating a polygon, by Marc van Kreveld

Overview

- Polygon triangulation, basics
- Art gallery
- Triangulating a monotone polygon
- Trapezoidal decomposition
- Triangulating a simple polygon
- Polyhedron tetrahedralization

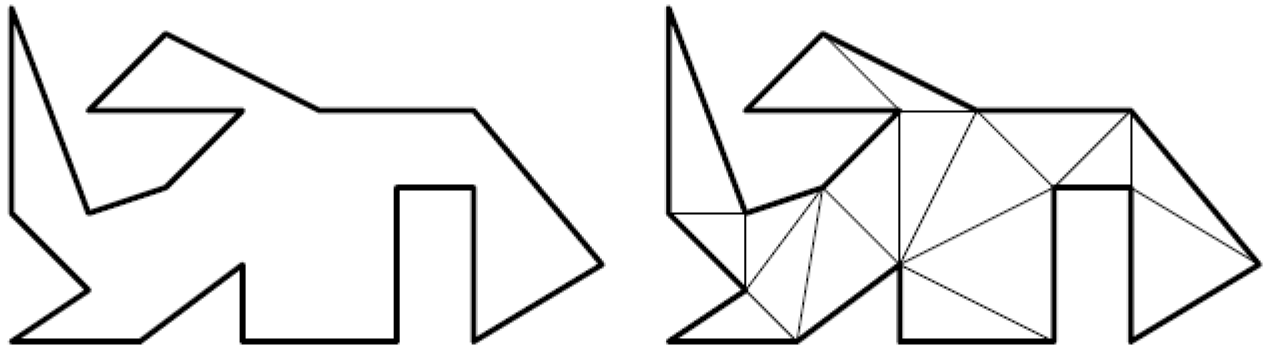
Credits

- CGAA: Chapter 3 in Computational Geometry Algorithms and Applications, Polygon triangulation, by de Berg et al
- L4vK: Lecture 4 in Computational Geometry, Triangulating a polygon, by Marc van Kreveld

Polygon triangulation, basics

Polygon triangulation, definitions

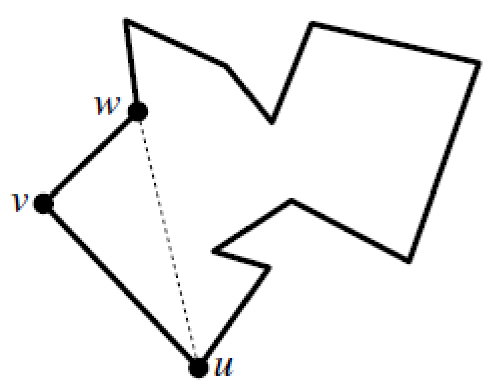
- *Diagonal* in a polygon is an open line segment that connects two vertices and lies completely inside the polygon
- *Triangulation* of a polygon is a subdivision of a polygon by a maximal set of diagonals that are pairwise disjoint

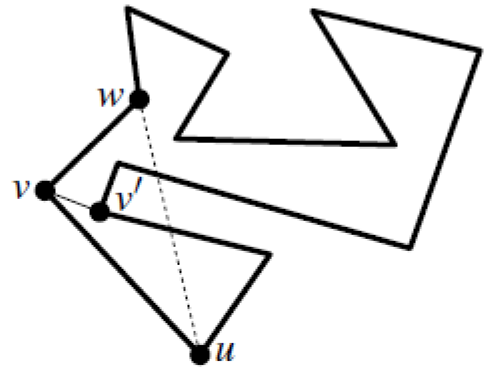


[CGAA]

Polygon triangulation, facts

- Every simple polygon admits a triangulation





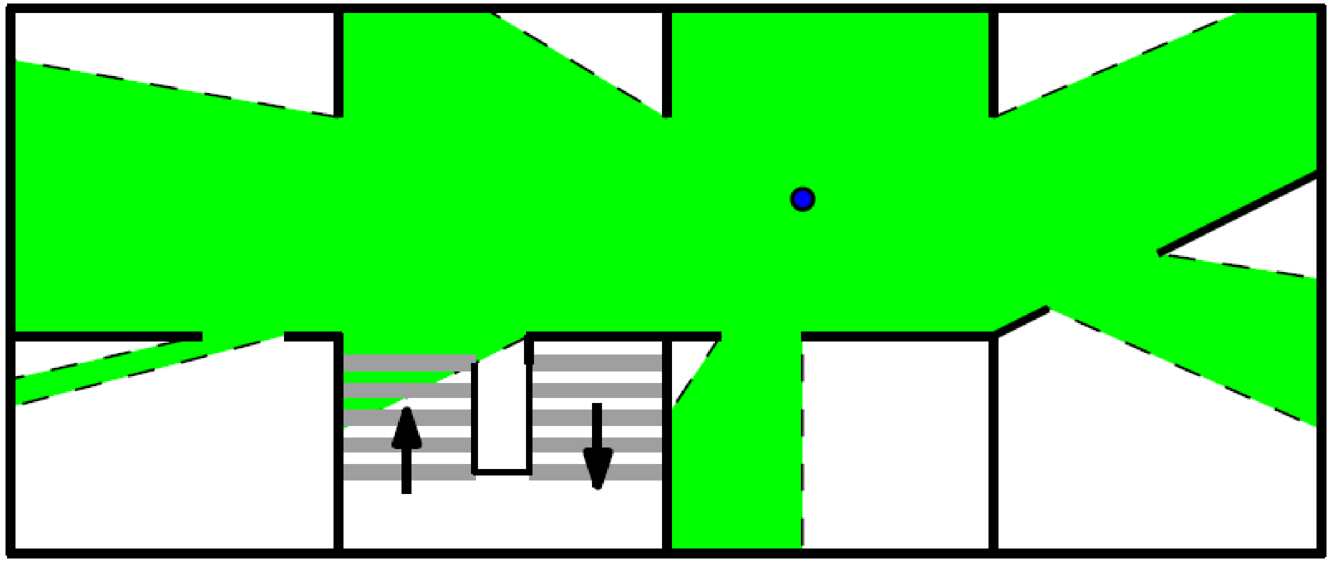
[CGAA]

- The triangulation of a simple polygon with n vertices uses $n-3$ diagonals and has $n-2$ triangles

Art gallery

Guarding an art gallery

- A point guard (or camera) placed inside the gallery sees every point in the gallery to which it can be connected with an open line segment that lies completely in the interior of the gallery (interior of a polygon or interior of a polyhedron)



[L4vK]

How many cameras do we need to guard a simple polygon with n vertices?

- max-min problem

$g(P)$: the minimum number of cameras to guard the simple polygon P

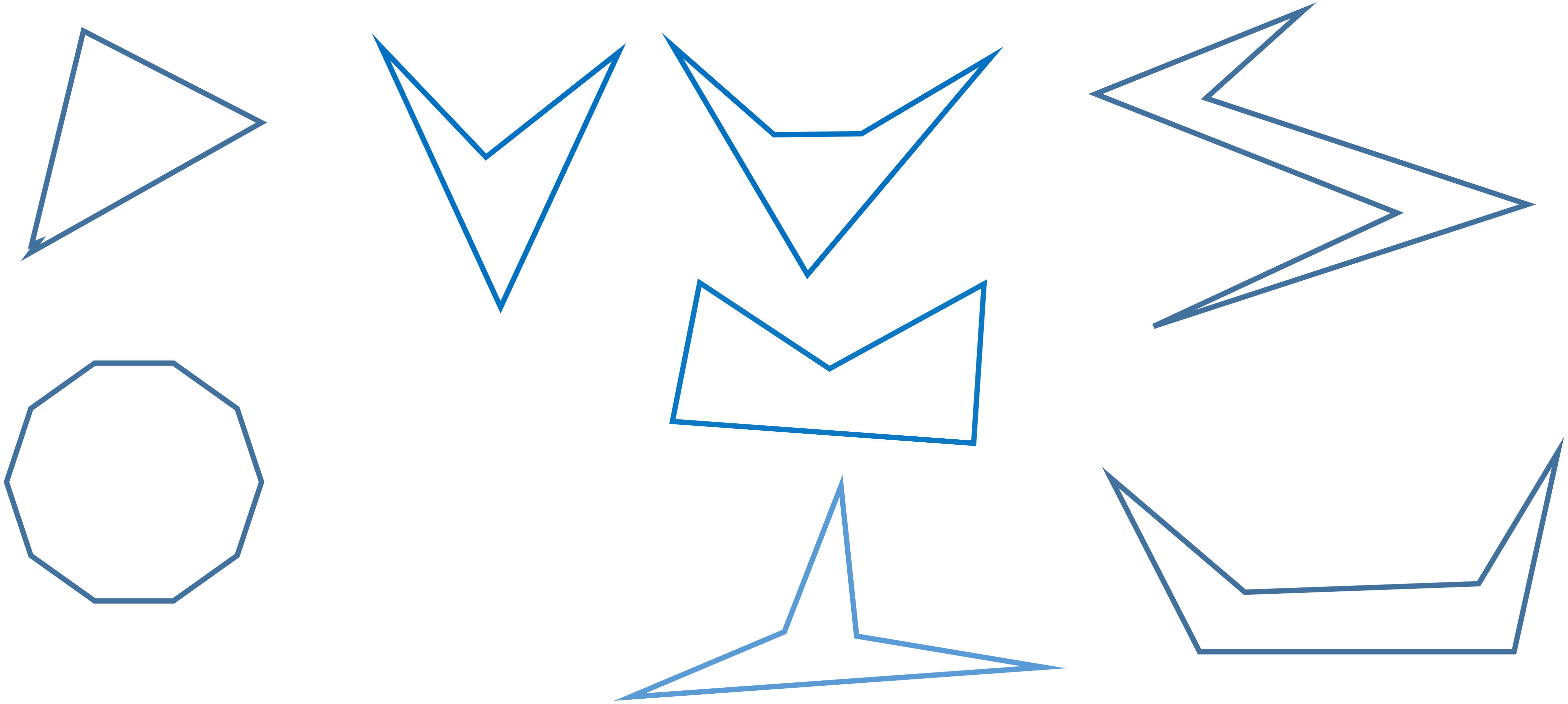
$g(n)$: the maximum of $g(P)$ over all simple polygons with n vertices

we prepare for the worst case

- Optimization: finding the minimum number of cameras for a given polygon is NP-hard

We will focus on the max-min problem

How many guards are necessary?

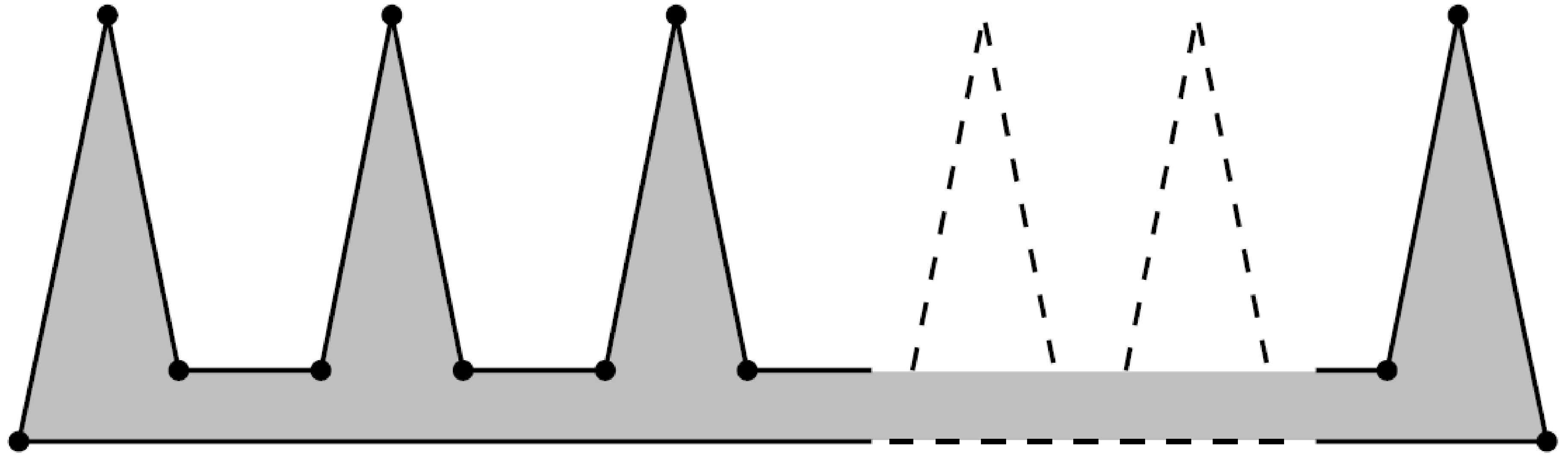


- $g(3)=1$, $g(4)=1$, $g(5)=1$, $g(6)=2$, $g(7)=?$

Art gallery theorem

$\lfloor n/3 \rfloor$ cameras are sometimes necessary and always sufficient to guard a simple polygon

Lower bound



[L4vK]

Upper bound ingredients

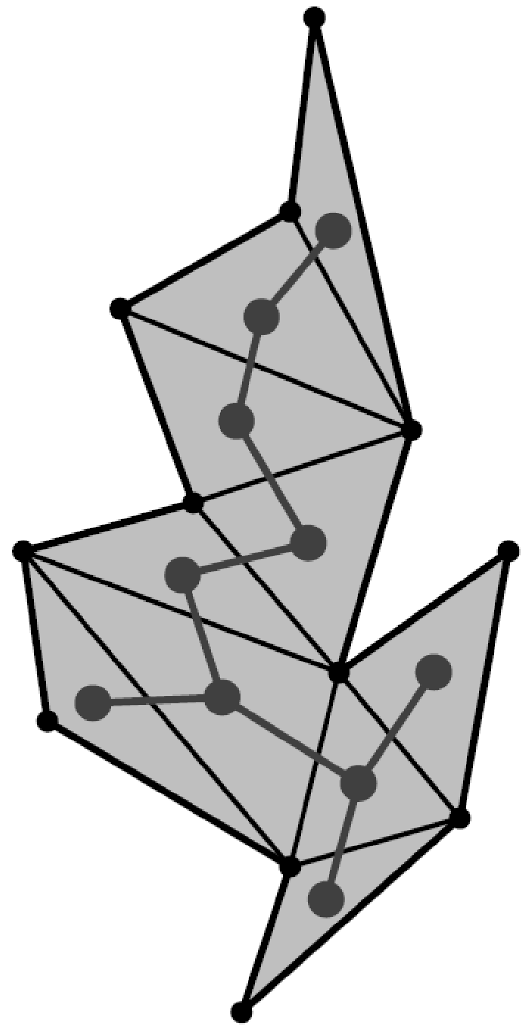
- The dual graph of a triangulation of a simple polygon
- Graph coloring

- The triangulation graph:
  - V: the vertices of the polygon
  - E: the edges of the polygon and the diagonals of the triangulation

The dual of a triangulation

- Like the dual of a plane graph, without the vertex for the unbounded face and its incident edges
- The dual graph of the triangulation of a simple polygon is a tree
- Three consecutive vertices along the boundary of a polygon u, v, w, define an *ear* iff uw is a diagonal
- A simple polygon has at least two ears

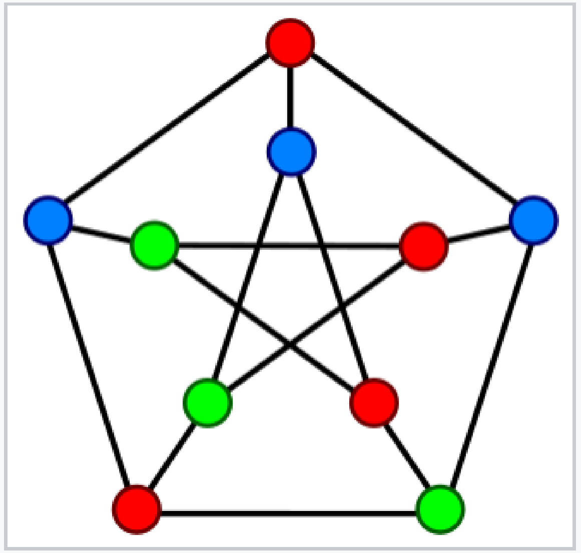
[breakout]



[L4vK]

Graph coloring

- *Graph coloring* is an assignment of labels, called colors, to the vertices of a graph such that no two adjacent vertices share the same color. The *chromatic number* $\chi(G)$ of a graph G is the minimal number of colors for which such an assignment is possible.


A proper vertex coloring of the Petersen graph with 3 colors, the minimum number possible.

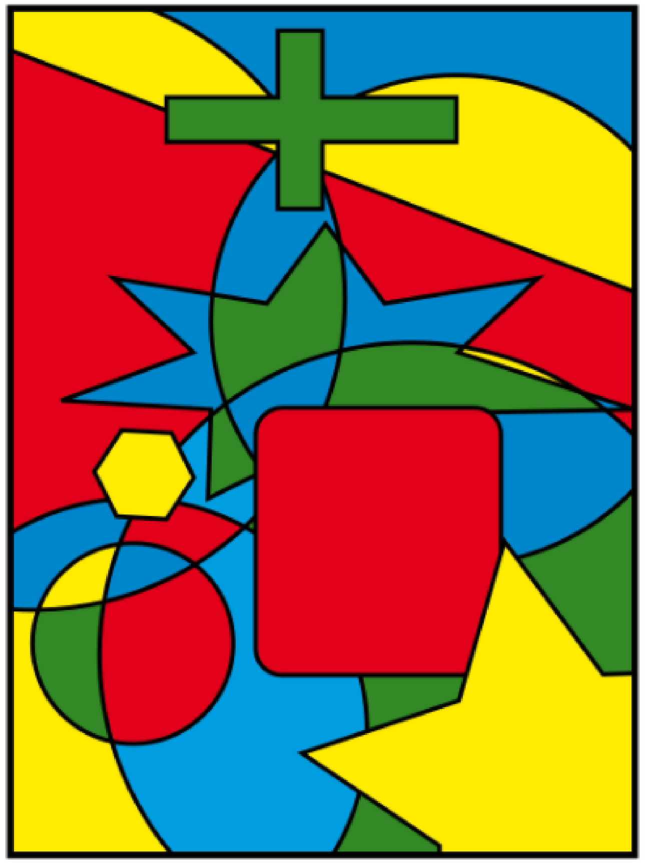
[wikipedia]

Side note

The four color theorem

Given any separation of a plane into contiguous regions, producing a figure called a map, no more than four colors are required to color the regions of the map so that no two adjacent regions have the same color.

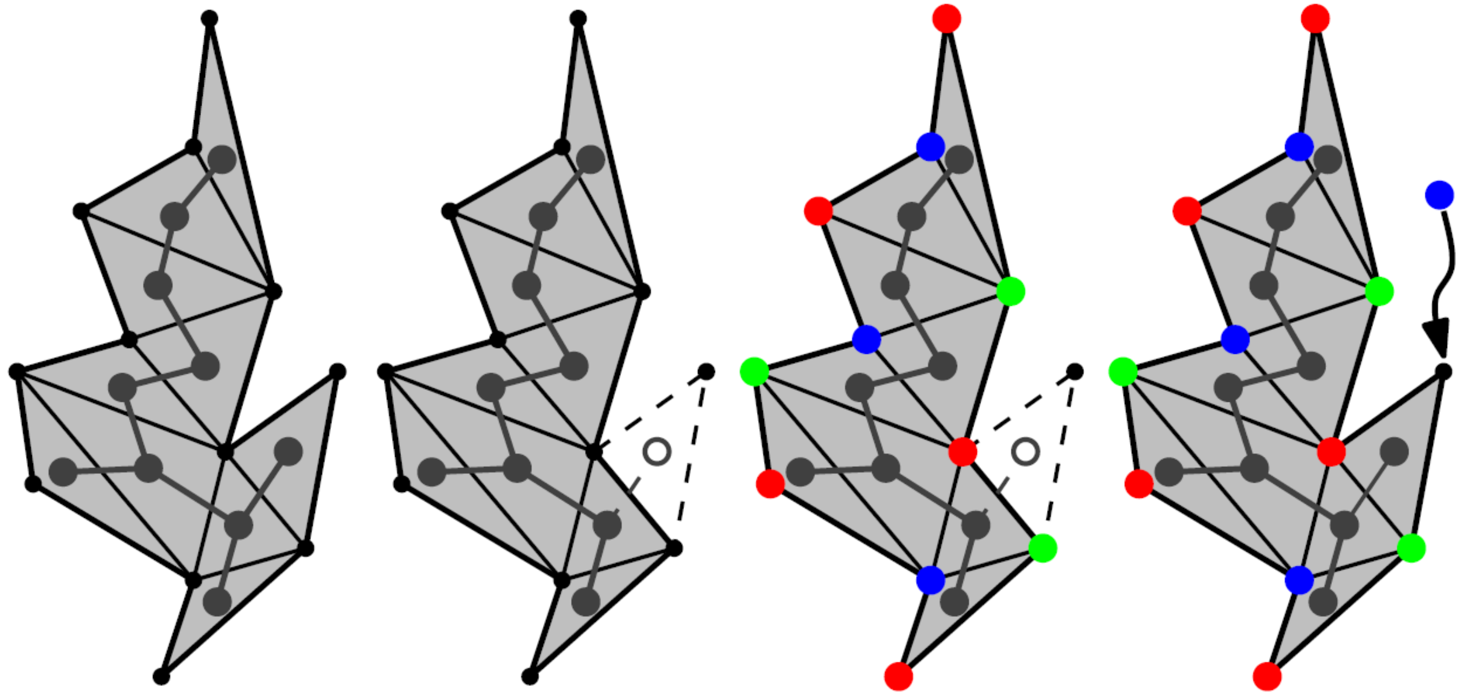
The chromatic number of a planar graph ≤ 4


Example of a four-colored map

[wikipedia]

The triangulation graph is 3-colorable

- Recall, the triangulation graph:
  - V: the vertices of the polygon
  - E: the edges of the polygon and the diagonals of the triangulation



[L4vK]

Upper bound

- 3 color the triangulation graph
- Take the color used by the least number of vertices, and place cameras at these vertices
- All the triangles are guarded and there are at most $\lfloor n/3 \rfloor$ such vertices

QED

[Fisk 1978, Chvátal 1975], [Meisters 1975]

Finding the cameras’ placement

- By traversing the dual graph of the triangulation, say in depth first. When we reach a new node (a new triangle) all the vertices of the already visited triangles have been properly colored. We reach the new triangle via an edge and hence two colors have already been used for this triangle, and we color the remaining vertex v with the third color. Since the graph is a tree, the other triangles having the vertex v have not been traversed yet.
- $O(n)$ time for a triangulated polygon with n vertices.

Art galley problems keep attracting research

- Fixed-parameter results
- Exact solutions (some CGAL based)
- Sensor placement

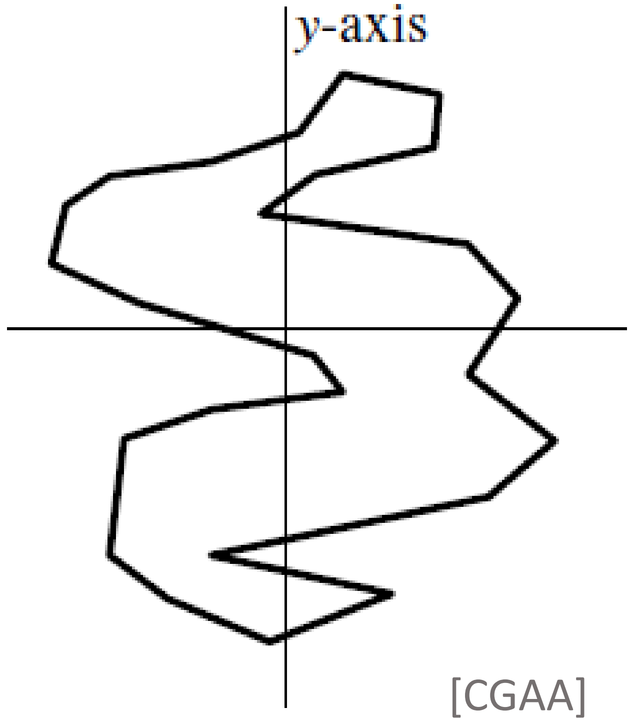
Video

http://www.computational-geometry.org/SoCG-videos/socg13video/

Triangulating a monotone polygon

Monotone polygons

- A polygon is *monotone* with respect to a line ℓ iff every line perpendicular to ℓ intersects the polygon in a connected set (in particular the empty set)
- A y-monotone polygon has a top vertex, a bottom vertex and two chains connecting these vertices, which together form the boundary of the polygon



[CGAA]

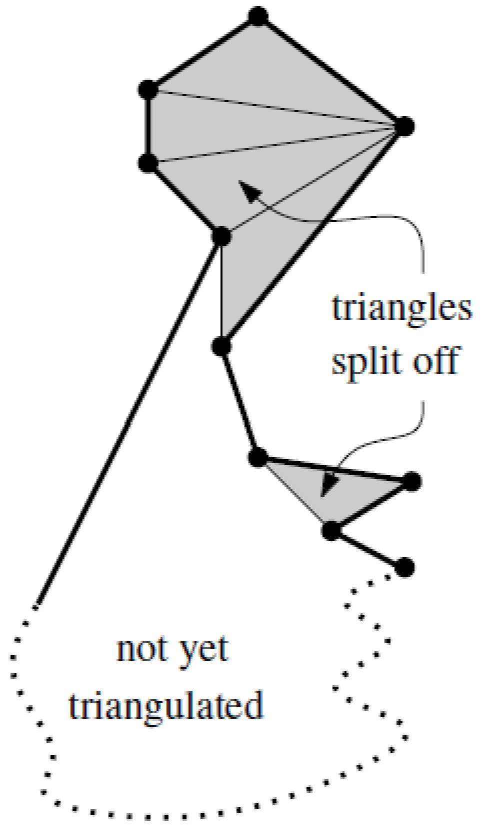
Triangulating a y-monotone polygon

High-level view

- Sort the vertices top-down by merging the vertices along the two chains
- Initialize a stack and push the first two vertices into the stack
- With the next vertex v add all possible diagonals to vertices in the stack, while producing triangles and popping these vertices
- Push v into the stack

- Assume for now that there are no horizontal edges; if vertices in different chains have the same y-coordinate, we consider the left vertex to be higher

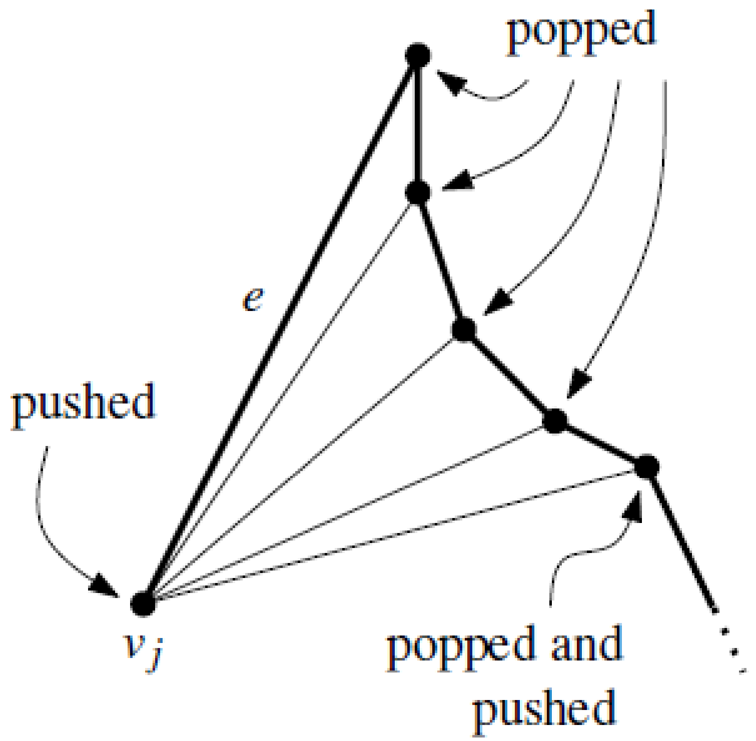
Example



[CGAA]

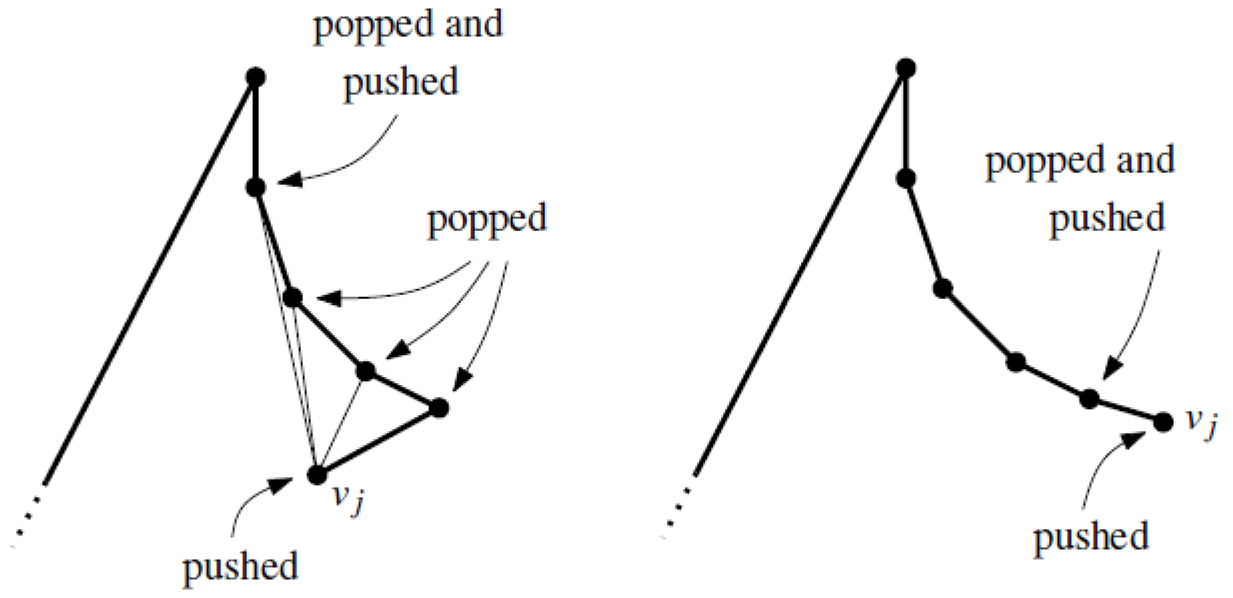
- *Invariant* of the algorithm: At the beginning of each step, the part of the polygon that has not yet been triangulated and lies above the last seen vertex has the shape of a funnel with one (incomplete) edge on one side and a reflex chain of vertices on the other side

Case I: the next vertex is on the non-chain side

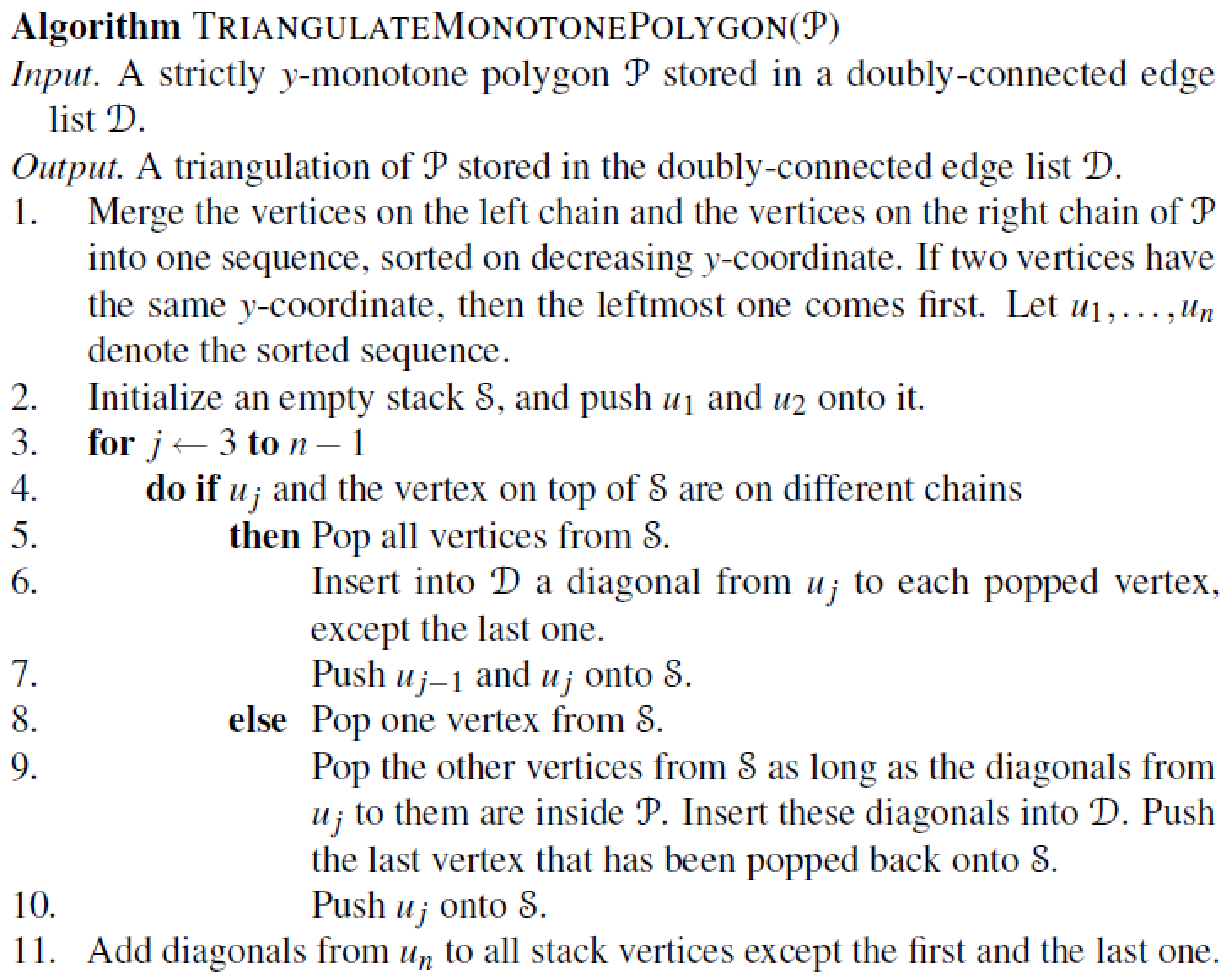


[CGAA]

Case II: the next vertex is on the chain side



[CGAA]



Algorithm TRIANGULATEMONOTONEPOLYGON($\mathcal{P}$)

Input. A strictly y-monotone polygon $\mathcal{P}$ stored in a doubly-connected edge list $\mathcal{D}$.

Output. A triangulation of $\mathcal{P}$ stored in the doubly-connected edge list $\mathcal{D}$.

```
1.  Merge the vertices on the left chain and the vertices on the right chain of $\mathcal{P}$
    into one sequence, sorted on decreasing $y$-coordinate. If two vertices have
    the same $y$-coordinate, then the leftmost one comes first. Let $u_1, \dots, u_n$
    denote the sorted sequence.
2.  Initialize an empty stack $\mathcal{S}$, and push $u_1$ and $u_2$ onto it.
3.  for $j \leftarrow 3$ to $n-1$
4.      do if $u_j$ and the vertex on top of $\mathcal{S}$ are on different chains
5.            then Pop all vertices from $\mathcal{S}$.
6.                 Insert into $\mathcal{D}$ a diagonal from $u_j$ to each popped vertex,
                   except the last one.
7.                 Push $u_{j-1}$ and $u_j$ onto $\mathcal{S}$.
8.            else Pop one vertex from $\mathcal{S}$.
9.                 Pop the other vertices from $\mathcal{S}$ as long as the diagonals from
                   $u_j$ to them are inside $\mathcal{P}$. Insert these diagonals into $\mathcal{D}$. Push
                   the last vertex that has been popped back onto $\mathcal{S}$.
10.                Push $u_j$ onto $\mathcal{S}$.
11. Add diagonals from $u_n$ to all stack vertices except the first and the last one.
```

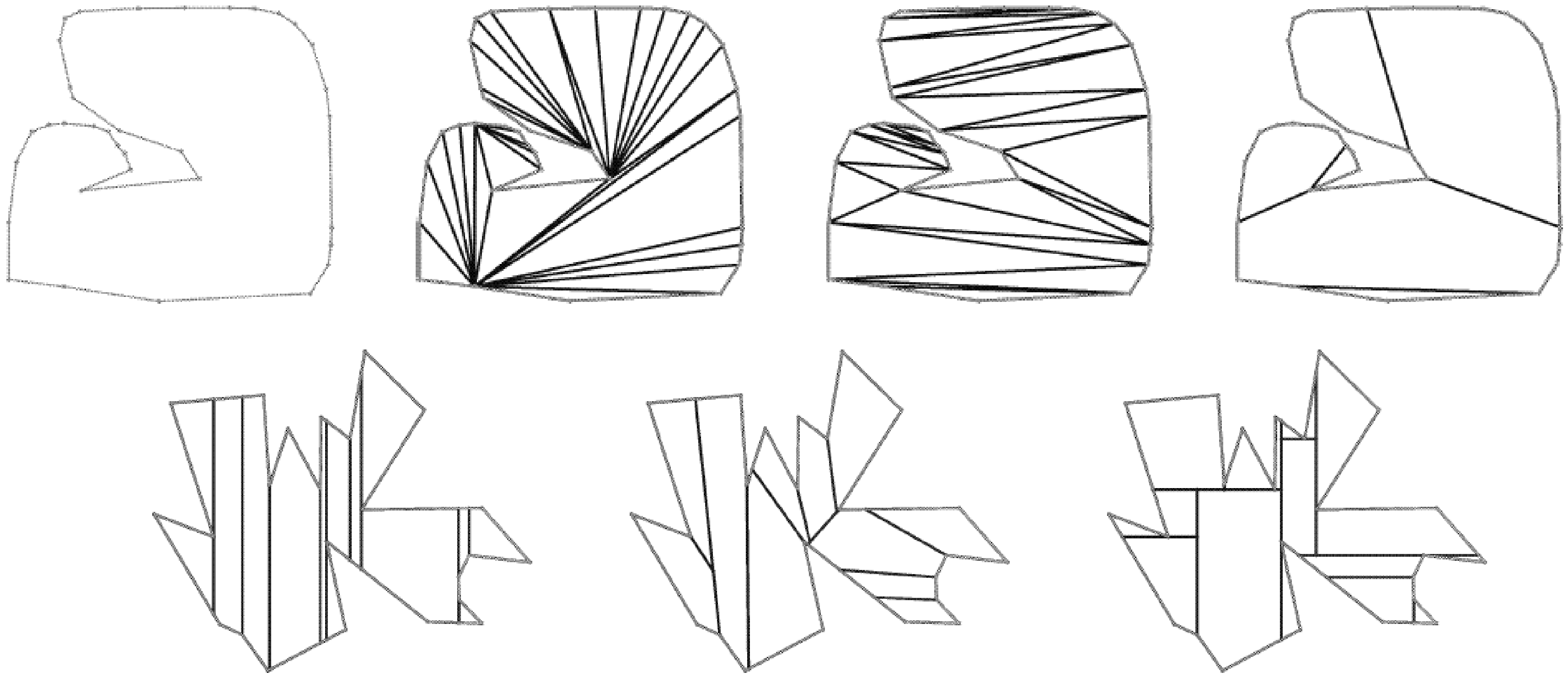
[CGAA]

Takes $O(n)$ time

Trapezoidal decomposition

of a polygon

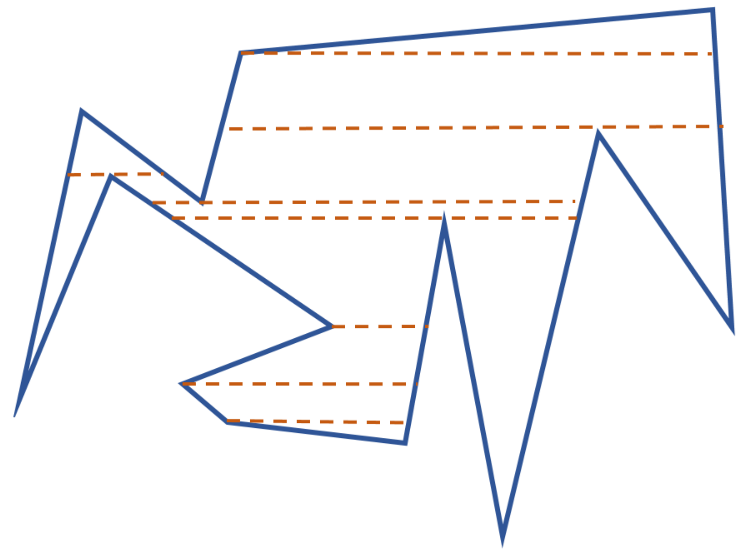
- There are many useful decompositions of a polygon, most of them into convex pieces

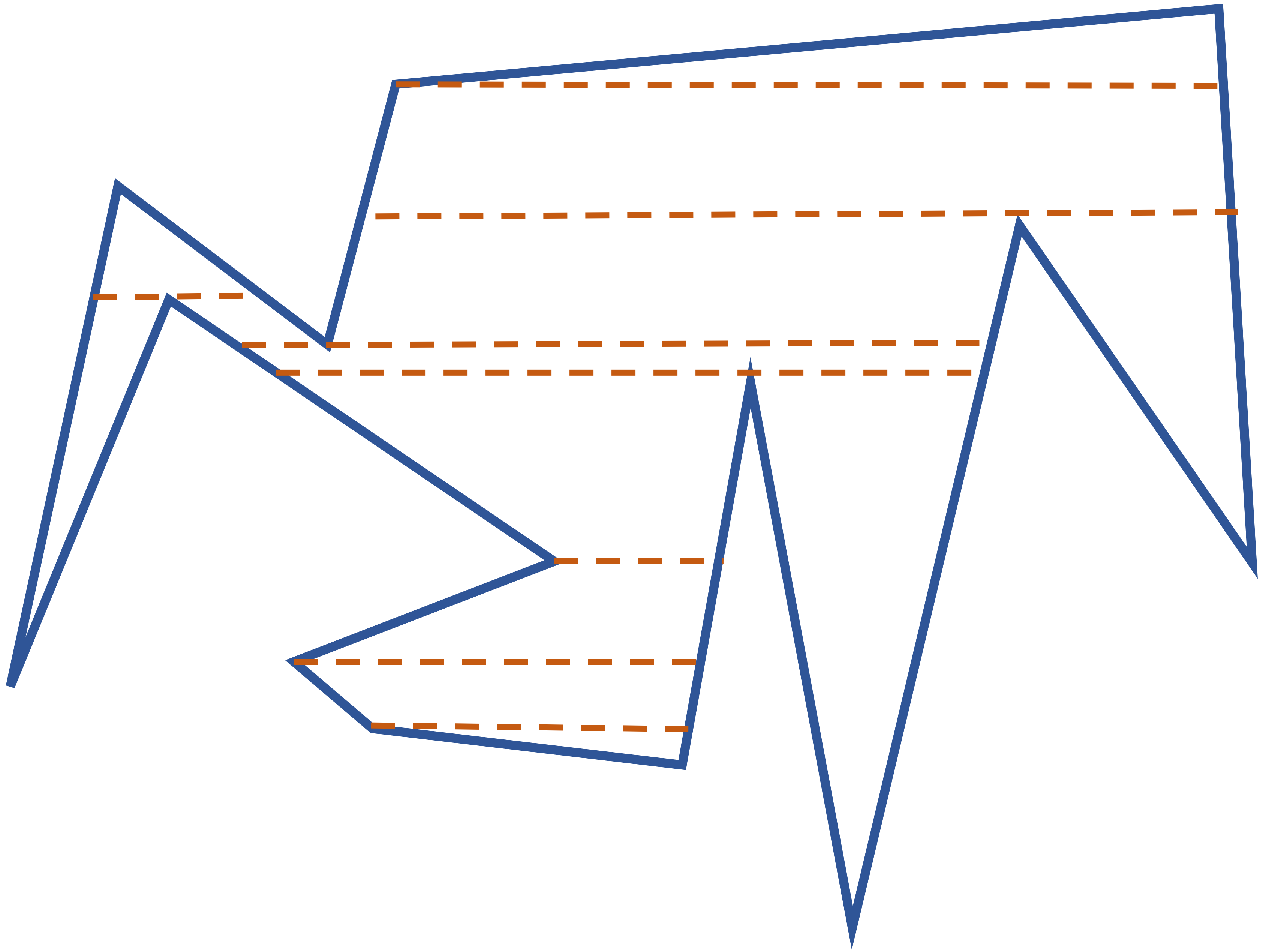


[Agawal-Flato-H]

Trapezoidal decomposition of polygon

- Special case of a procedure for planar maps (which we'll see in a few weeks) and for general subdivisions in any dimension
- Extend horizontal rays leftwards and rightwards from each vertex, *inside* the polygon, until they hit the polygon boundary





Algorithm

- How to represent the trapezoidal decomposition (TD)?

DCEL

- What is the complexity of the TD?

O(n)

- How can we compute the TD?

Sweep-line algorithm

- How much time does it take for a polygon with n vertices?

$O(n \log n)$

Triangulating a simple polygon

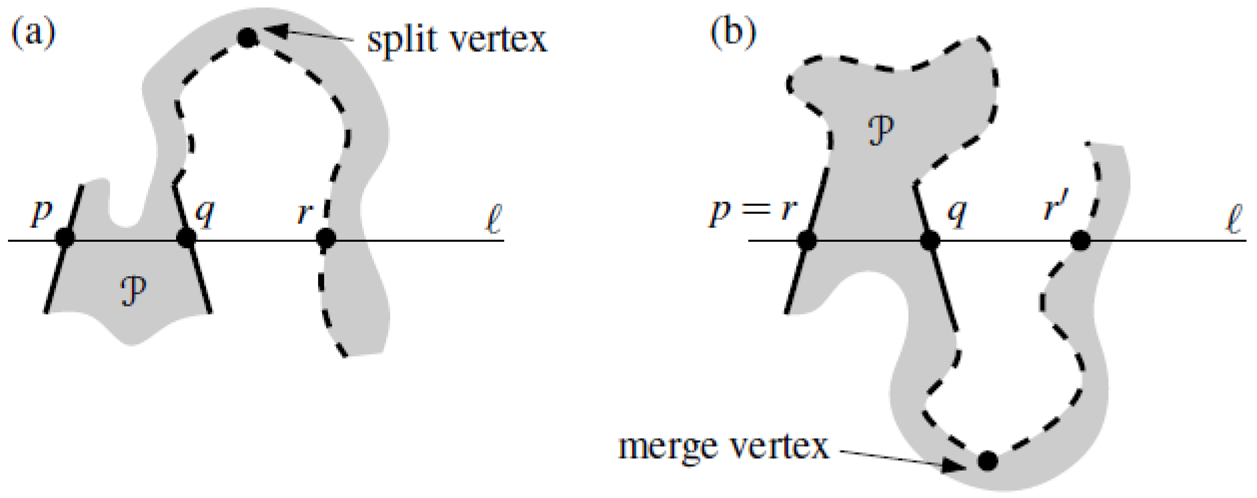
Approach

- Decompose the polygon into y-monotone polygons using diagonals
- Triangulate each of the y-monotone polygons using the linear-time algorithm that we saw earlier
- Stitch together all the sub-triangulations into the desired result

- We will use DCEL to represent the intermediate results as well as the final result
- We rely on a key observation connecting *inner cusps* and y-monotonicity

Inner cusps

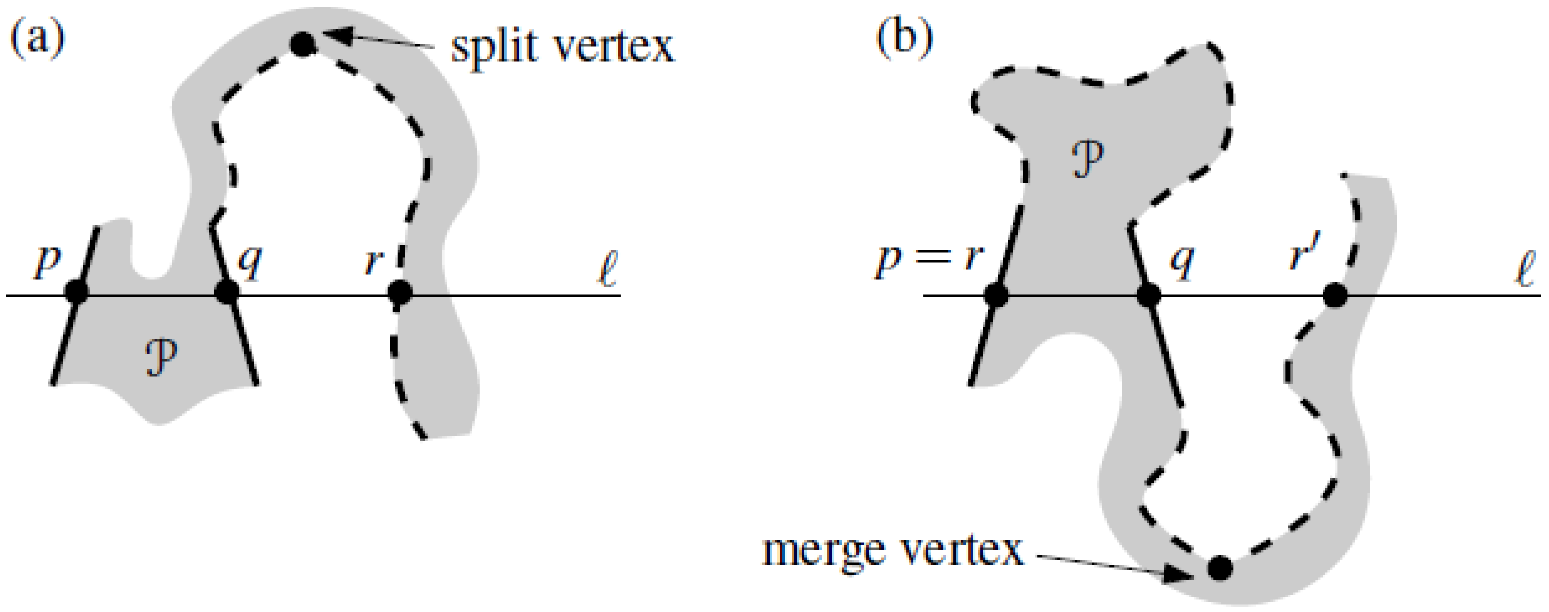
- An inner cusp of a polygon is a reflex vertex (whose interior angle is greater than π), and both its neighboring vertices are either above it (merge vertex) or below it (split vertex)
- We assume, for now and as before, that there are no horizontal edges



[CGAA]

Key observation

- A polygon is y-monotone iff it has no inner cusps
- $\Longrightarrow$ trivial
- $\Longleftarrow$



[CGAA]

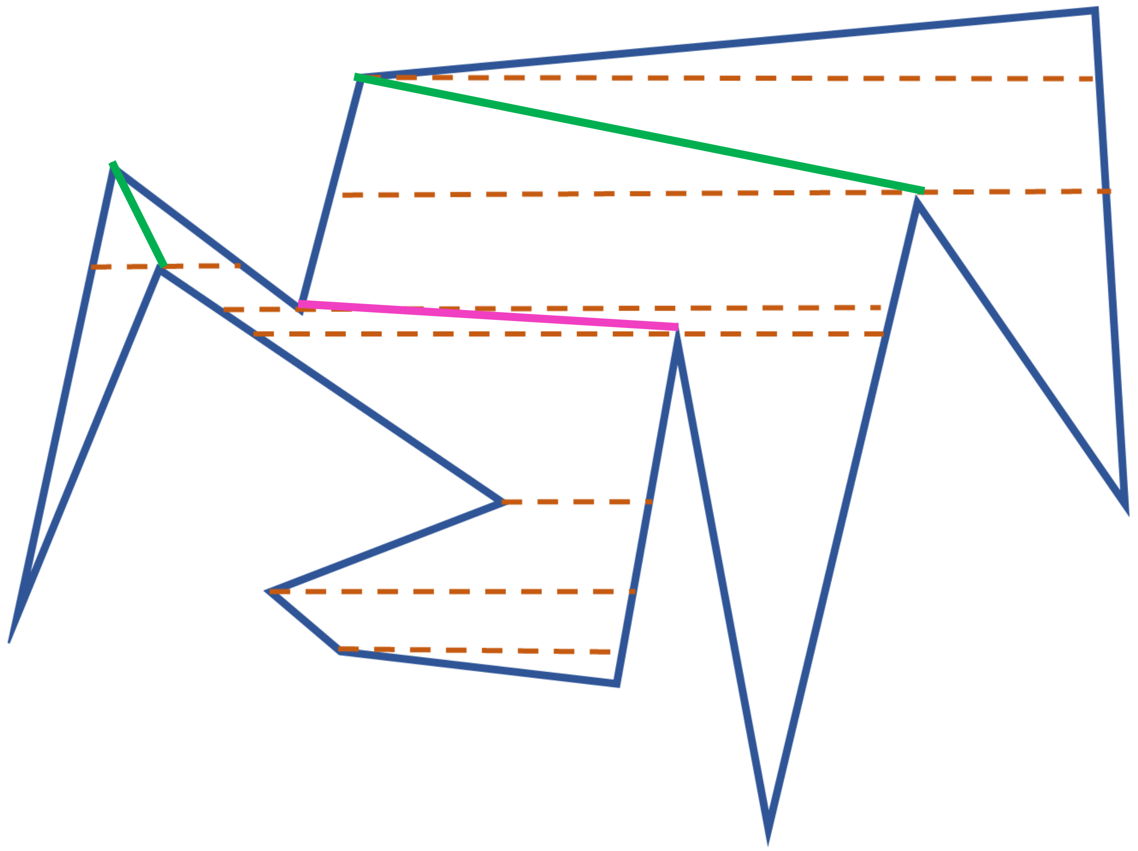
Removing inner cusps

- We will kill all the inner cusps by extending a diagonal upward from each split vertex and downward from each merge vertex
- These diagonals will partition the input polygon into y-monotone polygons
- We find the desired diagonals using the trapezoidal decomposition
- Each (horizontal) base of each trapezoid is induced by a single vertex of the polygon (assuming no two vertices with the same y-coordinate)

Adding diagonals

- If the upper base of a trapezoid t is induced by a merge vertex v, we connect v to the vertex inducing the lower base of t

- If the lower base of a trapezoid t is induced by a split vertex v, we connect v to the vertex inducing the upper base of t



Overall algorithm

- Using sweep line we compute the trapezoidal decomposition and the extra diagonals, using a DCEL to split the polygon into y-monotone polygons, in $O(n \log n)$ time
- We triangulate each of the y-monotone polygons
- The number of edges in all the y-monotone polygons together is at most $3n$, therefore triangulating them takes $O(n)$ time altogether

- We obtained an $O(n \log n)$ time algorithm to triangulate a simple polygon, using $O(n)$ space

Handling degeneracies

- Our concern are vertices with the same y-coordinate
- We rotate the polygon infinitesimally
- This induces a complete top-to-bottom ordering of events in the sweep-line algorithm as well as in triangulating each monotone polygon

Can one do better?

- Yes
- Bernard Chazelle:
  Triangulating a Simple Polygon in Linear Time. Discret. Comput. Geom. 6: 485-524 (1991)

Polyhedron tetrahedralization

The problem

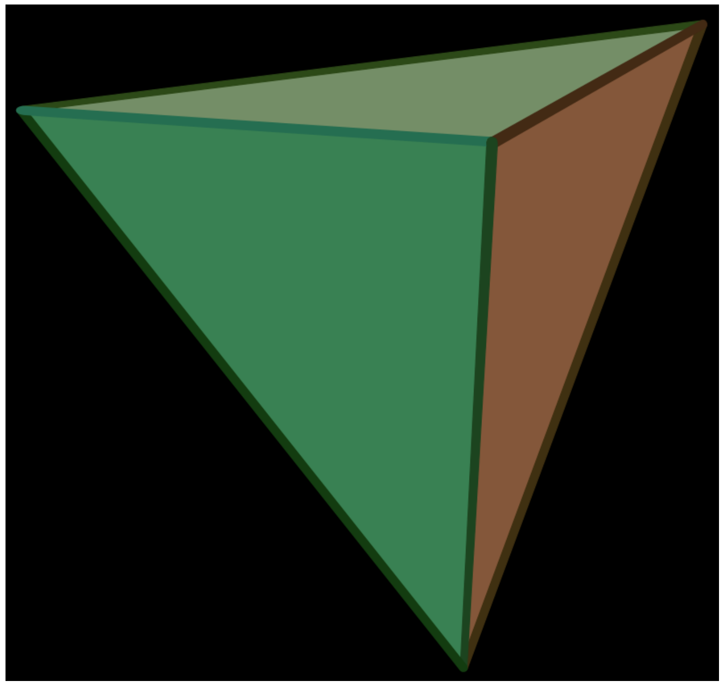
- Given a simple polyhedron with n vertices, how many cameras are needed to guard it?

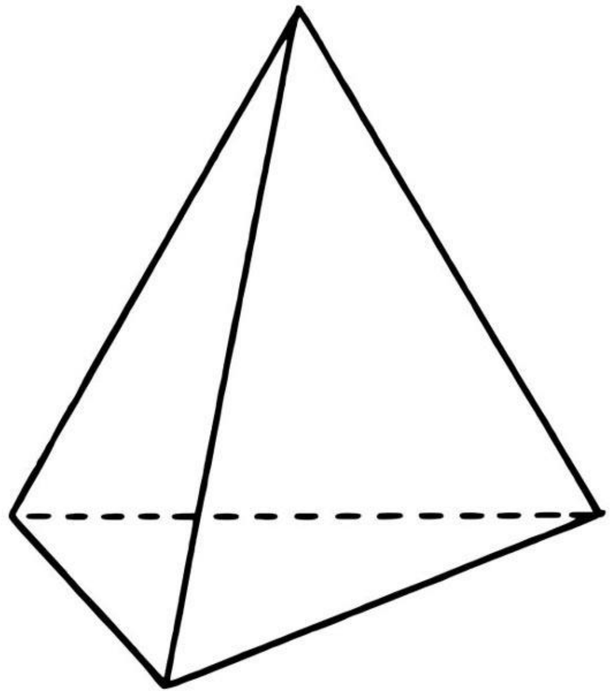
- What is a simple polyhedron?
- Euler’s formula applies and the number of edges and facets is $O(n)$
- The cameras are omni-directional

- How about cameras at the vertices?

Tetrahedron

- 3-simplex


[wikipedia]

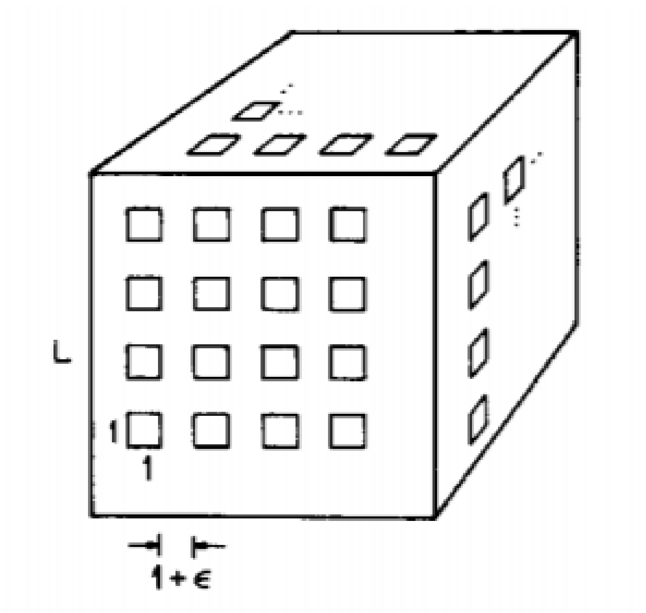

[quora]

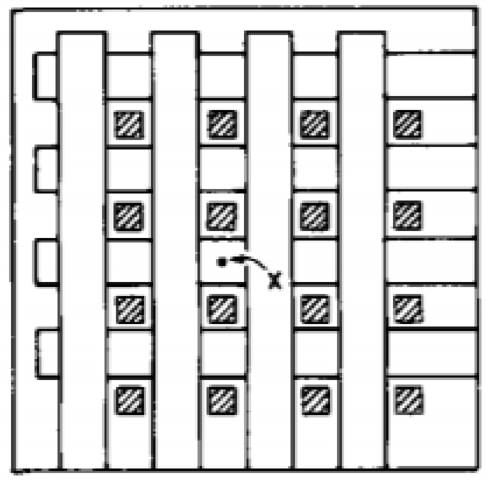
Generalizing triangulation to 3D

- Tetrahedralization: partitioning the interior of a polyhedron into tetrahedra, whose vertices are selected from vertices of the polyhedron
- Even simple polyhedra of genus zero (without holes) are not necessarily tetrahedralizable (see Ex 2.3)

Lower bound

- $\Omega(n^{3/2})$ cameras are necessary to cover this polyhedron

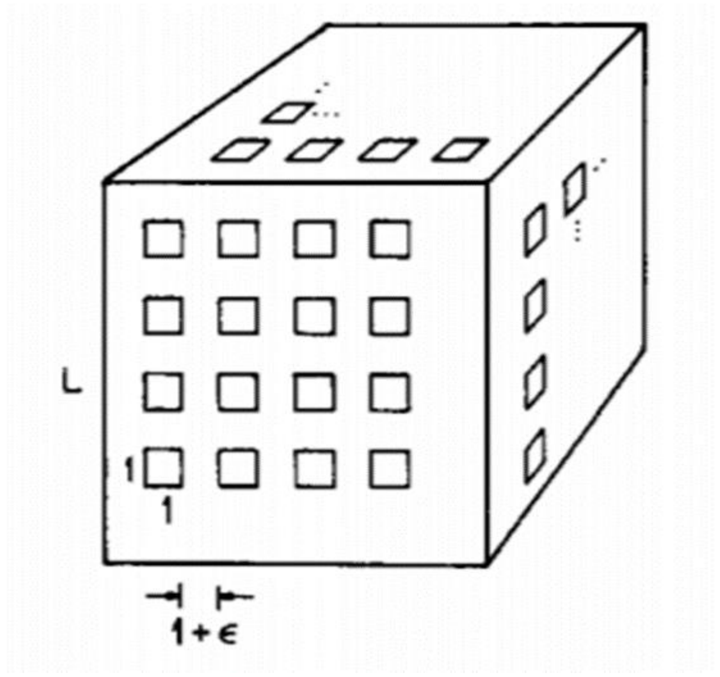




- Seidel's construction from *Art Gallery Theorems and Algorithms* by O'Rourke

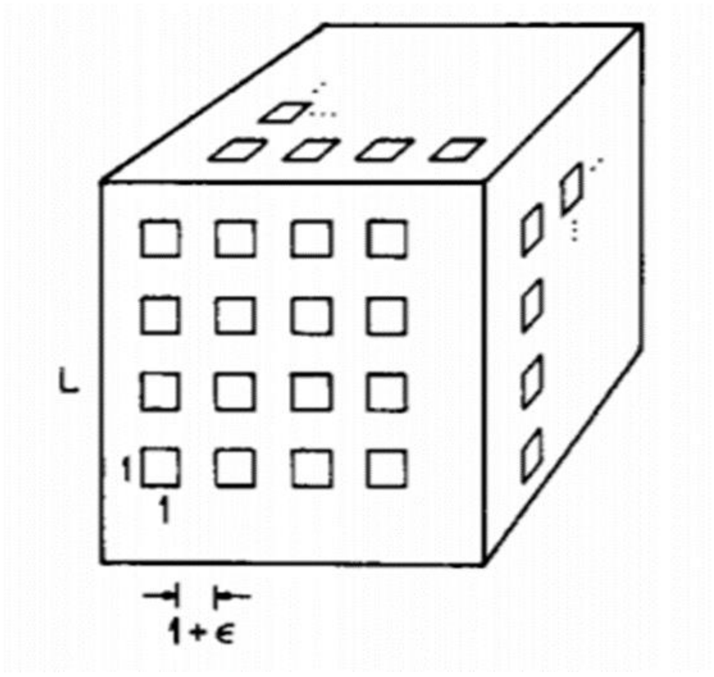
Lower bound, cont'd

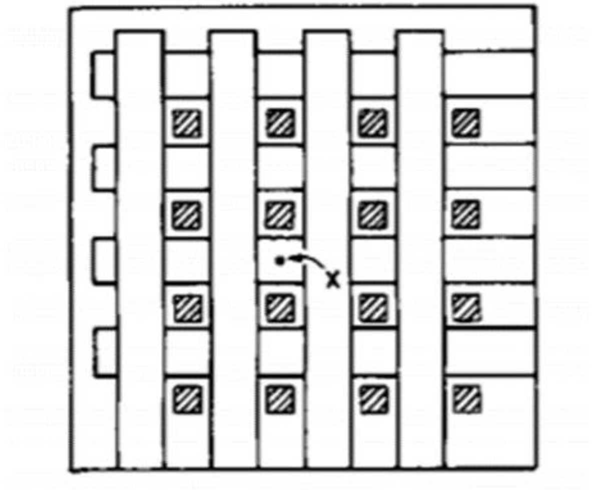
- On the front face mark squares of side length 1 in a $k * k$ regular array with distance $1 + \varepsilon$ between rows and columns, where $\varepsilon \ll 1$
- Attach a $1 \times 1 \times (L - \varepsilon)$ box behind each square and remove the square to cerate a deep dent reaching almost till the back face
- Repeat the same procedure for the right face and the top face such that none of the box dents intersect
- The polyhedron has $n = 8(3k^2 + 1)$ vertices



Lower bound, cont'd

- Point x at the bottom figure is at the center of a cube of size $(1+\varepsilon) \times (1+\varepsilon) \times (1+\varepsilon)$ bounded by six box dents and has ${}^{\varepsilon}/_{2}$ cracks along all 12 edge
- x cannot see any vertex of the polyhedron
- $(k-1)^3$ cameras are necessary
- $(k-1)^3 \approx {}^{n^{3/2}}/_{118} \ = \Omega(n^{3/2})$





THE END