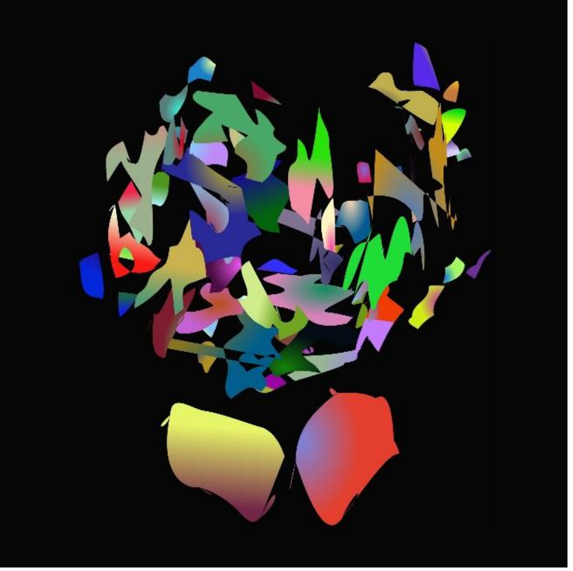


Algorithms for 3D Printing and Other Manufacturing Processes

c

Digital Model Simplification

Dan Halperin

School of Computer Science

Tel Aviv University

Spring 2017

Outline

- Background
- Minimal nested polygons
- Approximation of convex polytopes
- The Douglas-Peuker algorithm
- Progressive mesh decimation

Background

Basics

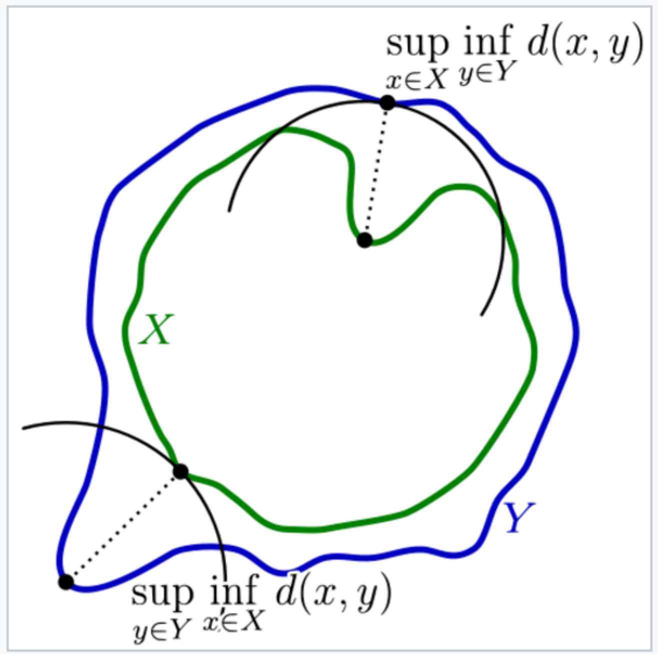
- Polyhedral (polygonal) models
- Simplification/approximation: produce a model with fewer vertices, edges, and faces
- Preserve some user-defined quality criteria
- Flavors:
  - Given a fixed number of allowable features, produce the best approximation within this bound
  - Produce the minimal model (feature-# wise) preserving the quality criteria

Distance measures of approximation quality

- Symmetric difference
- Hausdorff distance

$$d_{\mathrm{H}}(X,Y) = \max\{\sup_{x\in X}\inf_{y\in Y} d(x,y),\ \sup_{y\in Y}\inf_{x\in X} d(x,y)\}$$

- Fréchet distance, between curves (boundaries)


$\sup_{x\in X}\inf_{y\in Y} d(x,y)$
X
Y
$\sup_{y\in Y}\inf_{x\in X} d(x,y)$

Components of the calculation of the Hausdorff distance between the green line *X* and the blue line *Y*.

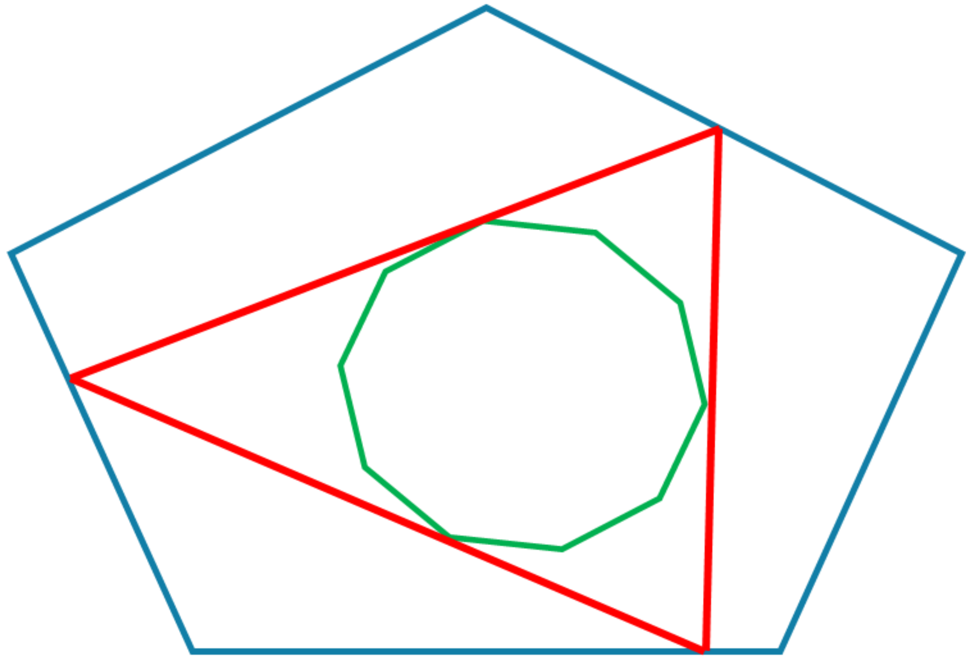
[Wikipedia]

Minimal nested polygons

The problem

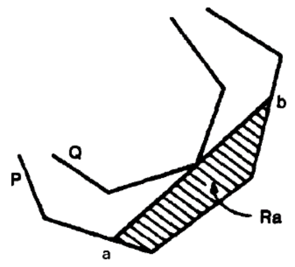
Given two convex polygons P and Q such that Q is contained in P, determine a minimum-vertex polygon K that contains Q and is contained in P

$|P|+|Q|=n$



Terminology

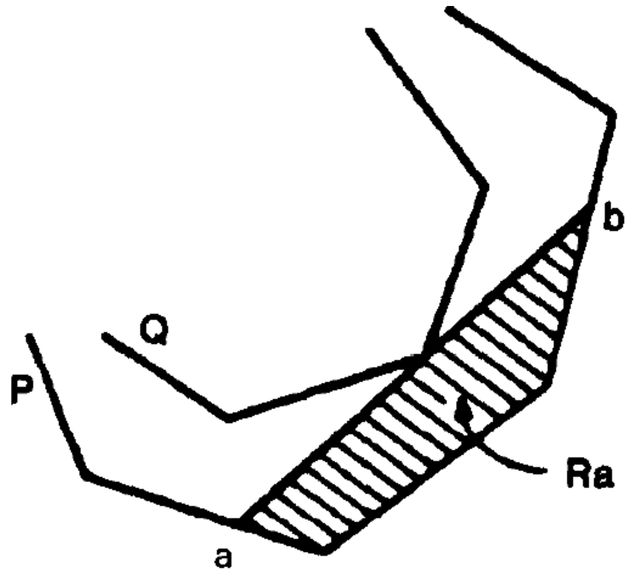
- Given P and Q, a nested polygon between P and Q is contained in P and contains Q
- A nested polygon between P and Q is minimal if it has the minimum number of vertices among all nested polygons
- We use P and Q (also) to denote the boundaries
- A: the closed region (the annulus) bounded between P and Q
- We walk along all polygons in CCW direction
- A supporting line segment: a directed segment in A that supports Q on its left and has both its endpoint in P



Terminology, cont’d

- For any point a on P let L_a be the unique supporting segment ab directed from a to b
- R_a: the region $H_{La} \cap A$, where H_{La} is the closed right half-plane determined by La
- R'_a: R_a – {a}

The figures in this section are taken from the paper “Finding minimal nested polygons” by Aggarwal et al; see references.



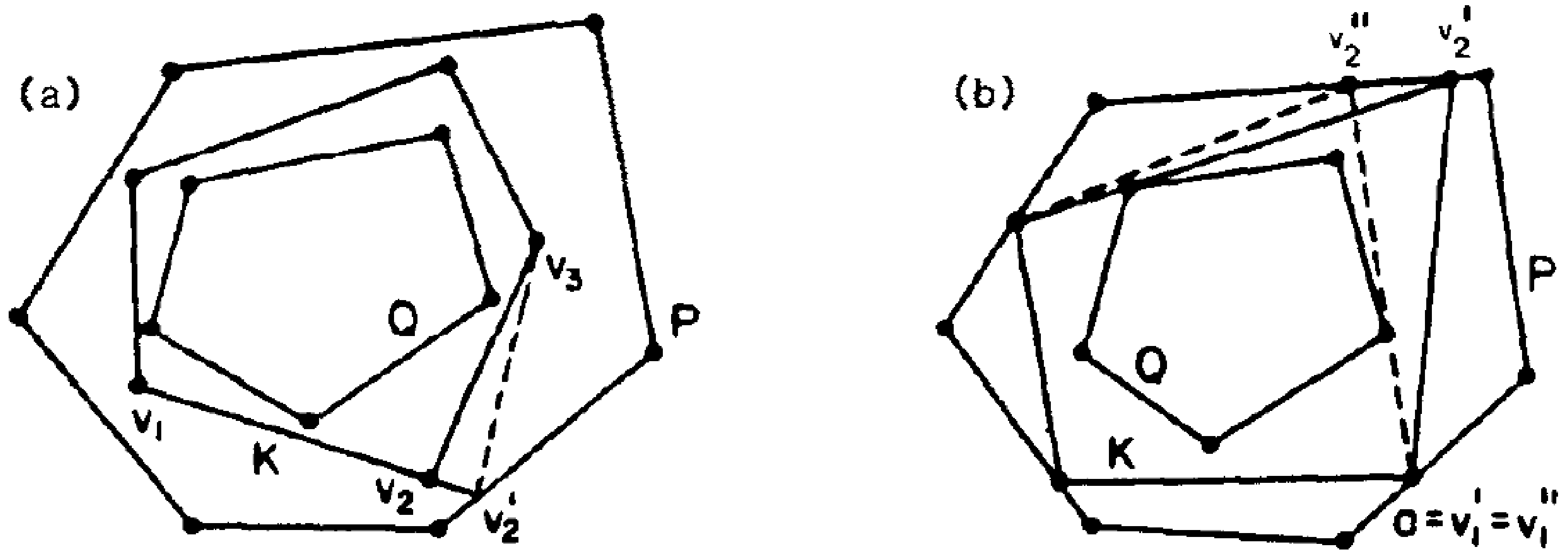
Basic lemmas

- Lemma 1: Every minimal nested polygon is convex
- Lemma 2: For any a∈P, R'_a contains at least one vertex of any minimal polygon
- Supporting polygon: all whose edges, except perhaps the last one, are supporting segments
- S_a: a supporting polygon starting at a∈P
- Lemma 3: For any a∈P, S_a has at most one vertex more than the minimum
- Lemma 4: Any minimal polygon $K=\{v_1,\ldots,v_k\}$ can be transformed into an S_a for some a∈P

Proof of Lemma 4

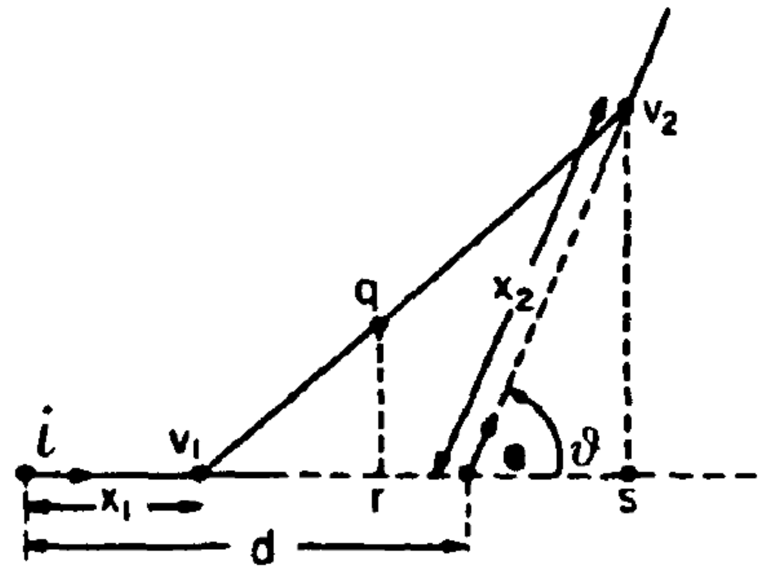
(a) move vertices to P

(b) make segments supporting (except perhaps the last one)



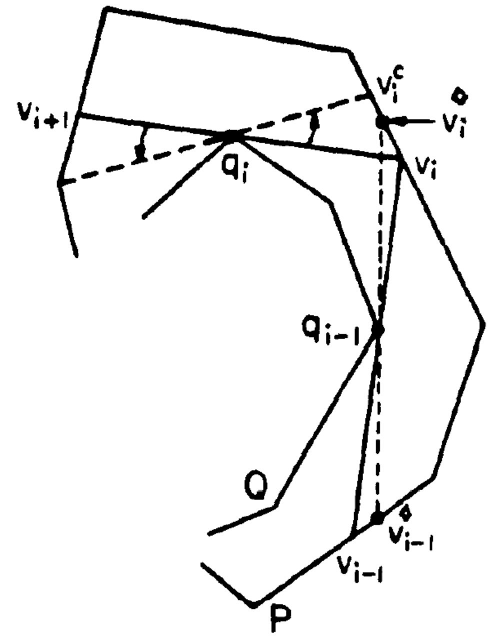
Algorithm

- Construct S_a for some $a \in P$, $S_x = S_a = S_{v1} = = \{v_1, \dots, v_k\}$
- Compute the projection functions
- q: the contact point of $L_a = L_{v1}$ with Q
- $q_y/(q_x - x_1) = x_2 \sin \vartheta/(d - x_1 + x_2 + \cos \vartheta)$
- $\rightarrow x_2 = q_y(d - x_1)/\sin \vartheta(q_x - q_y \cot \vartheta - x_1)$
- $\rightarrow x_2 = (c_1 + c_2 x_1)/(c_3 + c_4 x_1)$, where c_1, c_2, c_3, c_4 depend only on the features of P and Q involved in the contact with the edge $v_1 v_2$ of S_a



Projection functions cont'd

- Similarly we can write $x_3=(d_1+d_2x_2)/(d_3+d_4x_2)$, where d_i are constants as before
- We substitute x_2 in the equation above using $x_2=(c_1+c_2x_1)/(c_3+c_4x_1)$, and simplify, to get $x_3=(e_1+e_2x_1)/(e_3+e_4x_1)$
- We go on to obtain the functions $x_i=f_i(x_1)$, i=2,…,k
- These functions are valid unless a change in contact occurs for some edge v_i,v_{i+1}in S_x



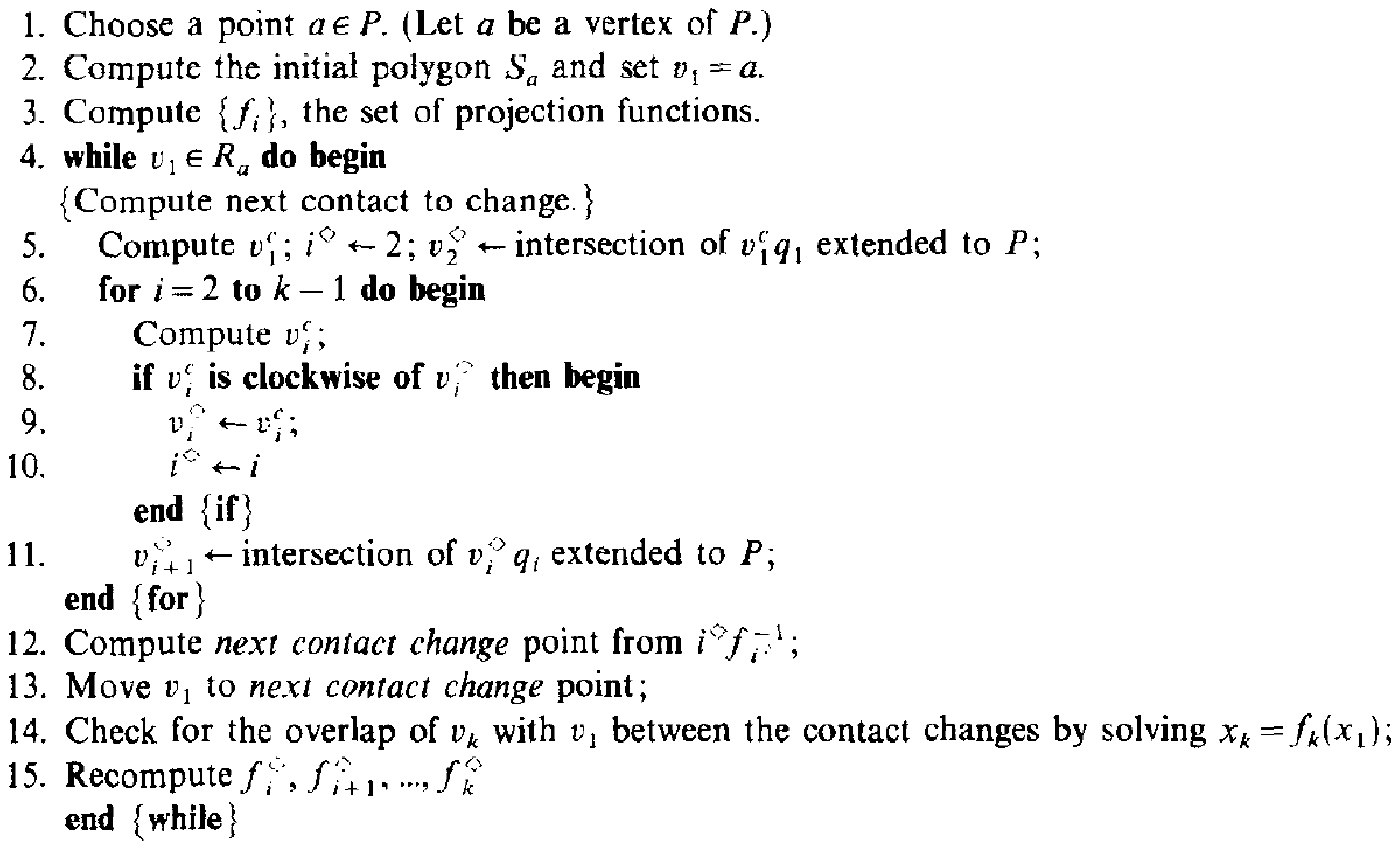
Computing the next contact change point

- A point $z \in P$ is called a contact-change point if S_z has at least one contact different from S_y where $y \in P$ is a point in a neighborhood of z and immediately preceding it in clockwise order
- A change in contact occurs for some edge v_i, v_{i+1}in S_x , while it rotates around q_i (its contact with Q), if one of the following happnes:
  - v_i reaches a vertex edge of P
  - v_{i+1} reaches a vertex edge of P
  - v_i, v_{i+1} aligns with the next edge of Q
- We denote the change point by v_i^c
- While sliding from v_i to v_i^c at least one edge of $\{v_1, \ldots, v_i\}$ changes contact; let v^*_i be the first such location

Change points cont’d

- Our goal is to compute v^*_{k-1}
- Assume we have already computed v^*_{i-1} then v^*_i is the more clockwise of
  - v_i^c
  - The intersection point of the extension of $v^*_{i-1}q$ with P
- In the next slide, in the algorithm summary, * is replaced by a diamond ◊

The algorithm



```
1. Choose a point $a \in P$. (Let $a$ be a vertex of $P$.)
2. Compute the initial polygon $S_a$ and set $v_1 = a$.
3. Compute $\{f_i\}$, the set of projection functions.
4. while $v_1 \in R_a$ do begin
   {Compute next contact to change.}
5.     Compute $v_1^c$; $i^{\diamond} \leftarrow 2$; $v_2^{\diamond} \leftarrow$ intersection of $v_1^c q_1$ extended to $P$;
6.     for $i = 2$ to $k - 1$ do begin
7.         Compute $v_i^c$;
8.         if $v_i^c$ is clockwise of $v_i^{\diamond}$ then begin
9.             $v_i^{\diamond} \leftarrow v_i^c$;
10.            $i^{\diamond} \leftarrow i$
           end {if}
11.        $v_{i+1}^{\diamond} \leftarrow$ intersection of $v_i^{\diamond} q_i$ extended to $P$;
   end {for}
12. Compute *next contact change* point from $i^{\diamond} f_{i^{\diamond}}^{-1}$;
13. Move $v_1$ to *next contact change* point;
14. Check for the overlap of $v_k$ with $v_1$ between the contact changes by solving $x_k = f_k(x_1)$;
15. Recompute $f_i^{\diamond}, f_{i+1}^{\diamond}, \ldots, f_k^{\diamond}$
   end {while}
```

Remarks

- We only need to look at supporting polygons which start at the portion of P’s boundary in R_{v1}
- We track v_k to check for its coinciding with v_1
- This may happen between contact-change points

Complexity

- Computing the initial supporting polygon S_a takes O(n) time
- Computing the initial projection functions takes O(k) time
- The total number of contact changes is O(n)
- Finding the next contact-change point takes O(k) time
- Updating the projection functions after a contact change takes O(k) time
- For a total of O(nk) time

- Can be improved to O(n log k) time

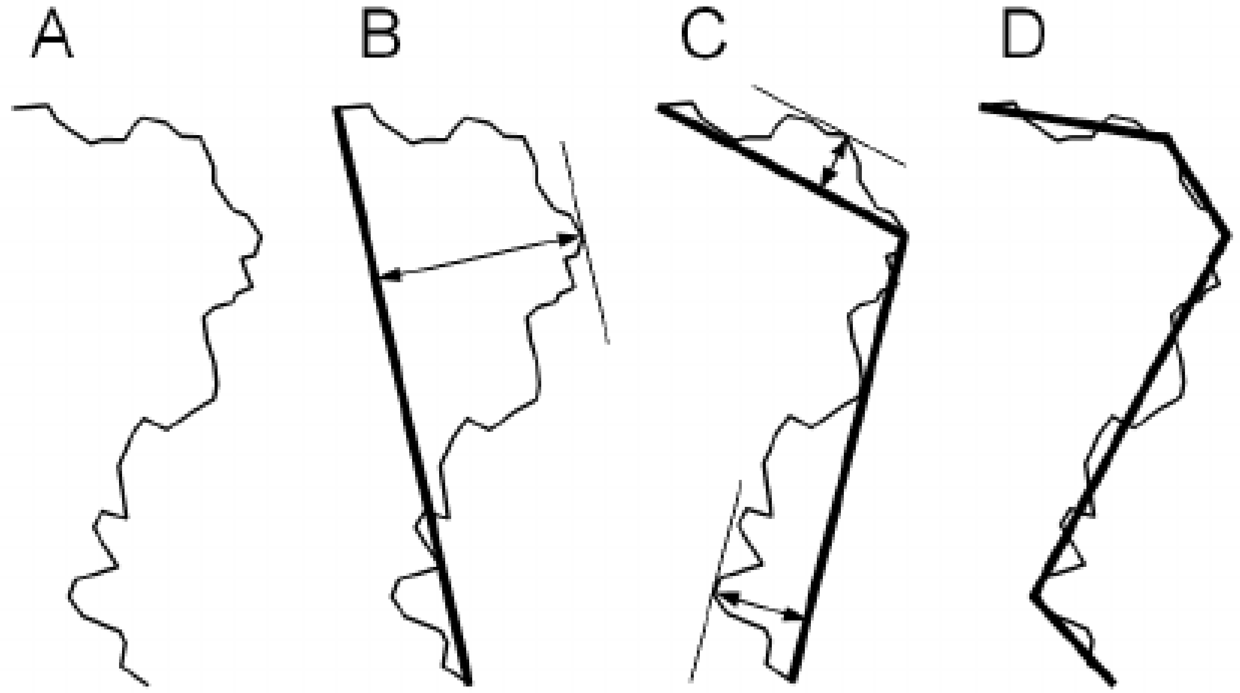
Approximating convex polytopes

- For a parameter $\varepsilon>0$, a polytope P is an ε-approximating polytope to a convex body K if the Hausdorff distance between K and P is at most ε
- Assume that the dimension d is fixed
- For $\varepsilon \leq 1$, any convex body K of unit diameter can be ε-approximated by a convex polytope P with $O((1/\varepsilon)^{(d-1)/2})$ facets [Dudley, ‘74]
- Similar result for vertices by Bronshteyn and Ivanov
- Many variants and extensions; see, e.g., a recent paper by Arya et al (bibliography below)

Simplifying a polygonal line

Douglas-Peuker

- Input: a polygonal line and a threshold ε>0



[Legland et al ‘14]

Progressive mesh decimation

Based primarily on Polygon Mesh Processing, Botsch et al, Chapter 7

Triangular meshes

- A collection of triangles
- A triangle mesh consists of
  - a geometric component: each vertex is embedded in R^3
  - and a topological component, represented, for example, by a graph structure like a DCEL, or a simpler graph connecting between vertices and faces
- A triangle mesh is a 2-manifold if it does not contain: non-manifold edges, non-manifold vertices, self-intersections

- For our purpose we assume that the mesh bounds a volume (post mesh repair)

Surface approximation is hard

The problem: Given a set S of n points sampled from a bivariate function f(x,y) and an input parameter ε>0, compute a piecewise linear function T(x,y) of minimum complexity (that is, an xy-monotone polyhedral surface, with a minimum number of vertices, edges, or faces) such that

$$|T(x_p,y_p) - z_p| \leq \varepsilon \text{ for all } (x_p,y_p,z_p) \in S$$

It is NP-hard to decide if a surface can be ε-approximated using k vertices (or facets) [Agarwal and Suri]

Possible decimation operations

- Vertex removal
- Edge collapse
- Halfedge collapse

Figure 7.3 [Botsch et al]

Progressive mesh decimation by edge contraction

1. Compute a local penalty for the contraction of each edge and store the candidate edges in a heap by penalty
2. Fetch the globally lowest penalty edge e_{min}
3. Contract e_{min} to a point (see below)
4. Re-compute the penalty for edges affected by the change and update the heap accordingly
5. If heap is not empty, go to 2

Topology preservation rules

- Contracting an edge (p,q) is valid
  - If both p and q are boundary vertices, then the edge (p,q) has to be a boundary edge
  - For all vertices r incident to both p and q there has to be a triangle (p,q,r)

Figure 7.4 [Botsch et al]

- Alternatively, the link condition lemma [Dey et al ’99; Sec 4.2 in Edelsbrunner’s book]

Distance measures

- In general: each triangle t_i in the decimated mesh is associated with a surface patch S_i in the original mesh
- We wish the decimated mesh to stay within a prescribed ε>0 from the original mesh
- We typically only estimate the distance

- Error accumulation (scalar or vector)
- Error quadrics – see next slide
- Hausdorff distance – good error estimation, expensive

Error quadrics

- Recall that each triangle t_i in the decimated mesh is associated with a surface patch S_i in the original mesh
- Error quadrics: estimating the sum of squared distances of p_j from all the supporting planes of triangles in the patches S_i that are associated with the triangles t_i surrounding p_j
- Used to find the optimal location of the contraction point
- 4x4 matrix
- Over counting (up to three times per original triangle)
- Approximating the distance to a triangle by the distance to its supporting plane can incur major underestimation
- Edge contraction ≈ addition of matrices: economical in storage and computation

[Garland and Heckbert]

Fairness criteria

- We can use the edge penalty, which estimates the deviation from the original mesh, to decide if we do the contraction otr not (binary decision) or we could use it also for ordering the contraction operations
- Alternatively, if we use the penalty only for the binary decision, we can add other quality criteria (these are called fairness criteria), for example:
  - Produce near-equilateral triangles
  - Minimize normal “jump” between adjacent triangles

Memory-less simplification

- No history, only local cost estimation per edge
- Less faithful approximation but more memory efficient
- That’s the CGAL methodology, based on [Lindstrom and Turk]
- Otherwise, the CGAL framework for simplification is generic

Simplification: Bibliography

- Alok Aggarwal, Heather Booth, Joseph O'Rourke, Subhash Suri, Chee-Keng Yap: Finding Minimal Convex Nested Polygons. Inf. Comput. 83(1): 98-110 (1989)
- R. M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. J. Approx. Theory, 10(3):227–236, 1974
- Sunil Arya, Guilherme Dias da Fonseca, David M. Mount: On the Combinatorial Complexity of Approximating Polytopes. Symposium on Computational Geometry 2016: 11:1-11:15
- David Douglas, Thomas Peucker: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. The Canadian Cartographer 10(2), 112–122 (1973)
- Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, Bruno Levy: Polygon Mesh Processing. A K Peters 2010
- Michael Garland, Paul S. Heckbert: Surface simplification using quadric error metrics. SIGGRAPH 1997: 209-216
- Hugues Hoppe: Progressive Meshes. SIGGRAPH 1996: 99-108
- Herbert Edelsbrunner: Geometry and Topology for Mesh Generation, Cambridge 2001

Simplification: Bibliography, cont’d

- Pankaj K. Agarwal, Subhash Suri: Surface Approximation and Geometric Partitions. SODA 1994: 24-33
- Peter Lindstrom, Greg Turk: Fast and memory efficient polygonal simplification. IEEE Visualization 1998: 279-286
- Peter Lindstrom, Greg Turk: Evaluation of Memoryless Simplification. IEEE Trans. Vis. Comput. Graph. 5(2): 98-115 (1999)

THE END

[Gaither, ArtByAI, CGAL arrgs]