A Subquadratic Algorithm for the Straight Skeleton

David Eppstein

Dept. Information and Computer Science
Univ. of California, Irvine

http://www.ics.uci.edu/∼eppstein/

(Joint work with Jeff Erickson, Duke Univ.)

Generalized Voronoi Diagrams

We know lots of Voronoi diagram algorithms

Which ones work when we want to construct something "similar to" a Voronoi diagram?

E.g. "abstract Voronoi diagram" – system of pseudolines behaving similarly to bisectors in VD – has $O(n \log n)$ randomized incremental construction [Klein et al., CGTA 1993]

But "straight skeleton" [Bookstein, CGIP 1979; Aichholzer et al., J. Univ. Comp. Sci. 1995] – relative of medial axis of a polygon, is not an abstract Voronoi diagram. How quickly can we construct it?

Outline

I. Roof design and straight skeletons

(i) Roof design problem

(ii) Straight skeleton definition

(iii) Connections to medial axis

II. Straight skeleton algorithm

III. Collision detection data structure

(i) Data structure definition

(ii) Update operations

(iii) Nearest neighbor searching

IV. Generalizations and conclusions

Roof design problem:

Given floor plan of a building

And given style of roof

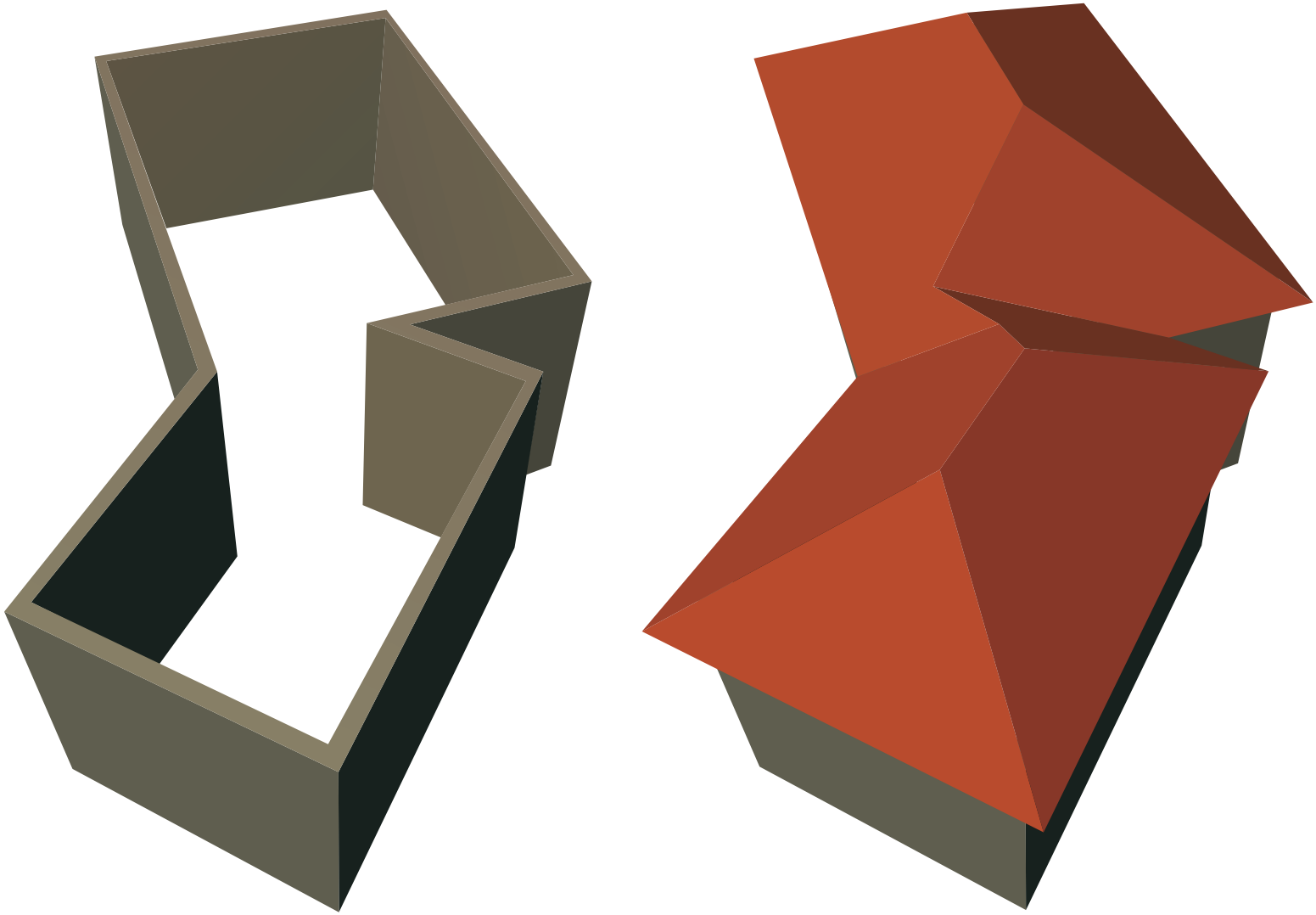
E.g. hip: all walls support a roof plane, all roof planes have the same slope

Design the roof

Determine topology of ridge and valley segments

Find coordinates for corners

How to fit a roof to these walls?



Offset curves and straight skeletons

Consider cutting roof by horizontal planes

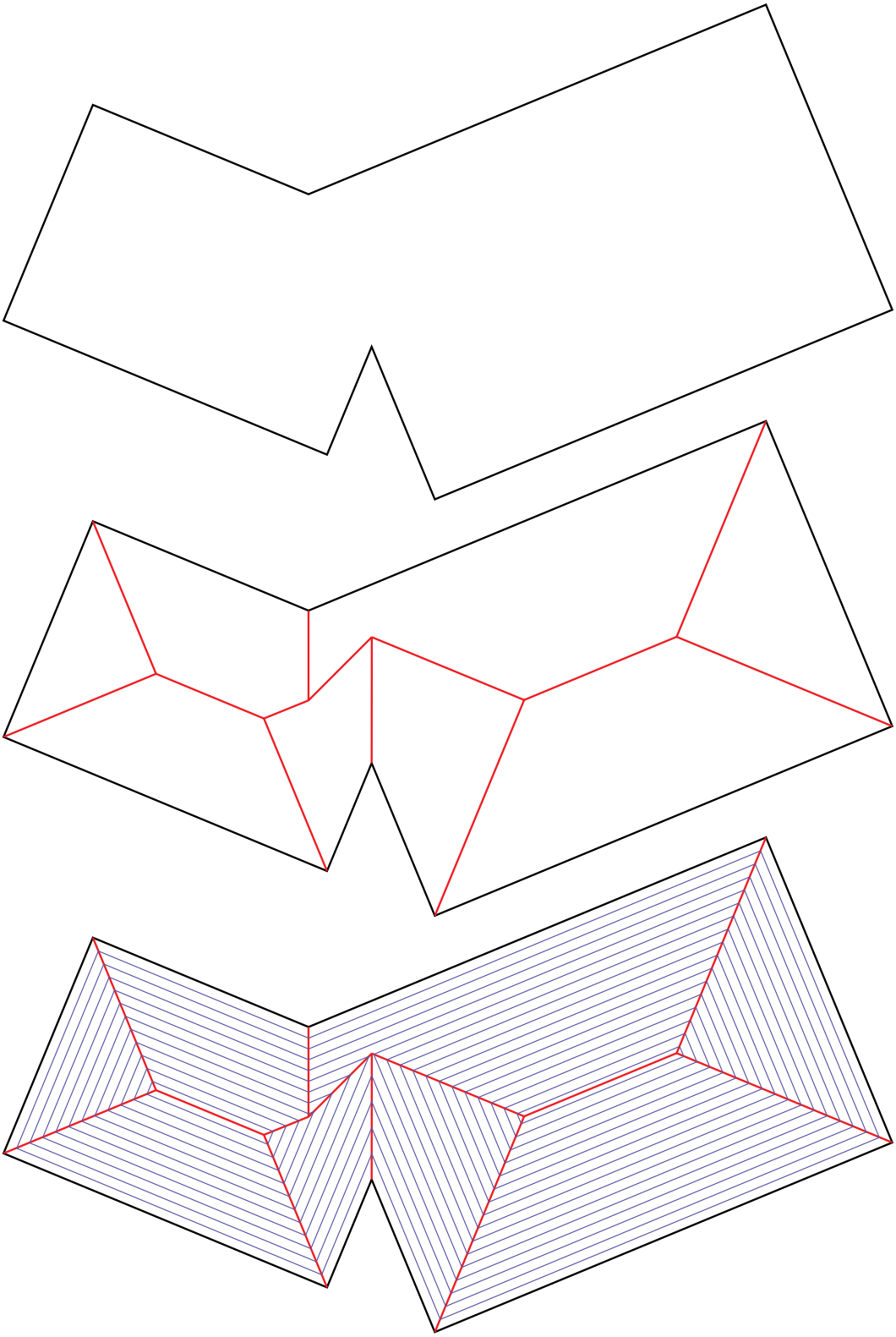
Forms offset curves:

- Polygons with edges parallel to original edges, shifted some fixed distance
- Corners are mitered (sharp)

Vertices in offset curves trace ridges and valleys of roof, forming a tree inside the polygon

This tree is known as the straight skeleton

Straight Skeleton and Offset Curves



Medial axis

Medial axis has multiple definitions

- Voronoi diagram of polygon edges
- Lower envelope of wedges (above edges) and cones (above vertices)
- Locus of centers of circles tangent to two points of polygon
- Paths traced by vertices of offset curves with rounded corners

Many applications

Problem: has curved edges

Relation to medial axis

Both are one-dimensional retractions of polygon

SS has no curved edges

SS and MA both generated by offset curves

(So both trees coincide for convex polygons)

SS can also be defined as a certain lower envelope

Wedges above edges
and tilted wedges above roof valleys

But where do those valleys end?

Straight skeleton algorithm

Continuously shrink offset polygon

At every discrete change in topology, form a straight skeleton feature

Possible topology changes:

- Whole polygon shrinks to nothing
- Single edge shrinks to nothing
- Reflex vertex collides into opposite edge

First two types easy to maintain in a priority queue; what about third?

Collision detection for offset poly features

Given two sets S and T, undergoing additions, removals, and modifications of objects

and given a binary function $f(s,t)$

maintain the pair $(s \in S, t \in T)$
that minimizes the value $f(s,t)$

For straight skeleton, $S =$ offset polygon vertices, $T =$ edges, $f(s,t) =$ time when vertex hits edge

Previous approaches I: Brute force

After each update, compare all pairs (s, t) to find the pair minimizing $f(s, t)$

Advantage: no extra storage

Disadvantage: slow ($O(n^2)$ per update)

Previous approaches II: Discrete event simulation

Maintain priority queue of all pairs

After update, change n queue entries

Advantage: relatively fast ($O(n \log n)$ per update)

Disadvantage: uses too much memory ($O(n^2)$)

New results

Relatively simple data structure

Reduces collision detection to range searching

(well-studied problem with nearly optimal solutions)

Sublinear update times

Can trade off time and space:

$O(n^{6/11+\epsilon})$ per update with $O(n^{17/11+\epsilon})$ space

$O(n^{3/4+\epsilon})$ per update with $O(n^{1+\epsilon})$ space

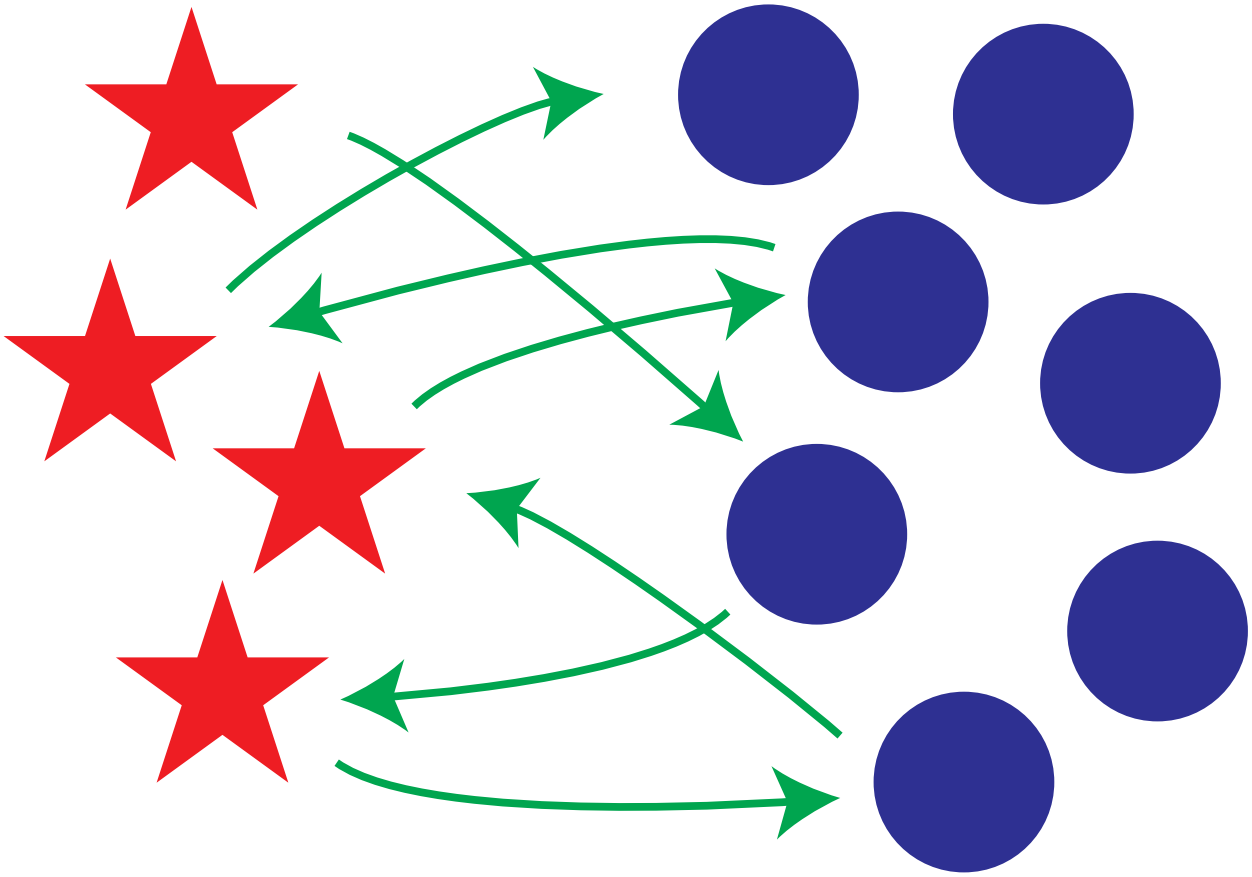
So can compute straight skeleton in $O(n^{17/11+\epsilon})$ time, or $O(n^{7/4+\epsilon})$ with nearly linear space.

Simplified version with naive range searching still improves previous approaches and may be more practical.

Conga Lines

Choose any object to start the line

End of line chooses its favorite object among unchosen objects in other set



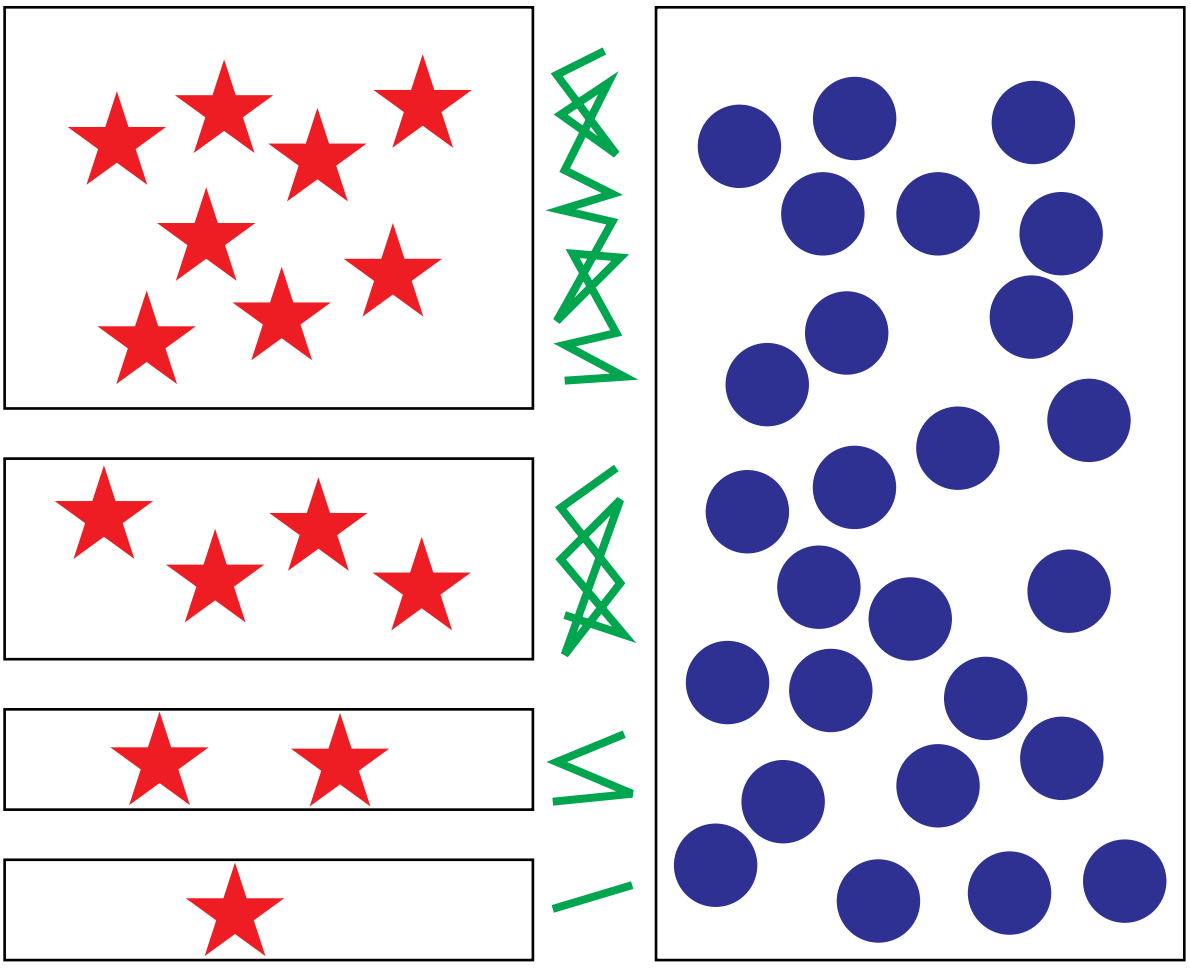
Lemma: if *f(s,t)* is minimized, then either *s* chooses *t* or *t* chooses *s*

Overall Data Structure

Divide *S* into powers of two

For each subset, form conga line with *T*

(similarly, divide *T* and form conga lines)



Data structure insertions

To insert an object:

- Make new singleton subset
- Regroup subsets into distinct powers of two
- Recompute conga lines

Analysis:

- Each time object is involved in a recomputation, subset size doubles
- So at most $\log n$ recomputations
- Total per insertion:
  $O(\log n)$ nearest neighbor queries

Data structure deletions

To remove an object:

Remove it from $O(\log n)$ conga lines (breaking each line in two)

Treat neighbors at broken ends of lines as if they were newly inserted objects

Analysis:

Each deletion causes $O(\log n)$ insertions

Total time per deletion:
$O(\log^2 n)$ nearest neighbor queries

Nearest Neighbor Queries

Return to 3d view: $S =$ valleys, $T =$ roof planes

Nearest neighbor to valley:

> Which plane does this valley hit first?
> (Ray shooting)

Nearest neighbor to plane:

> Which valley hits this plane first?
> (Inverse ray shooting)

In both cases, need to answer queries while inserting and deleting objects from sets.

Both can be solved efficiently using standard ϵ-cutting based algorithms

Summary Times for nearest neighbor queries (and therefore for collision detection):

$O(n^{6/11+\epsilon})$ per update with $O(n^{17/11+\epsilon})$ space

$O(n^{3/4+\epsilon})$ per update with $O(n^{1+\epsilon})$ space

So can compute straight skeleton in $O(n^{17/11+\epsilon})$ time, or $O(n^{7/4+\epsilon})$ with nearly linear space.

Some improvement when few reflex vertices: $O(n^{8/11+\epsilon}\, r^{9/11+\epsilon})$

Generalizations

Other kinds of roof (mansard, half-hip, gambrel, gull-wing, gable, etc) give generalized skeletons

No longer a lower envelope, but still generated by offset curves (with different speeds for different edges, or change of speed for some edges)

Same algorithm applies, collision detection somewhat slower ($O(n^{3/5})$ per collision)

Collision detection data structure
has many other applications

Other generalized Voronoi diagrams?
Higher dimensions?