The trapezoidal map of non-crossing line segments

Problem: Polygon Triangulation

Given a simple polygon $P$ with $n$ edges, compute a triangulation of its interior.

Solution via Trapezoidal Map

Given a set $S$ of $n$ nonintersecting segments in the plane, compute its trapezoidal map.

Trapezoidal Map

- planar graph, vertices $V$, edges $E$, faces $F$
- $V$: endpoints, artificial vertices
- $E$: pieces of segments, vertical extensions
- $F$: set of trapezoids, each one incident to at most 4 segments (assuming no two endpoints have the same $x$-coordinate; not true in triangulation application, but can be achieved even there)
Randomized Incremental Construction

1. Compute trapezoidal map of \( \{s_1\} \mapsto T_1 \)

\[ s_i \]

2. Insert segments \( s_2, \ldots, s_n \) in random order \( \mapsto T_n \)

---

From \( T_{r-1} \) to \( T_r \) (I)

Find

---

From \( T_{r-1} \) to \( T_r \) (II)

Split

---

From \( T_{r-1} \) to \( T_r \) (III)

1. **Find**: Find the trapezoid containing the left endpoint of \( s_r \)

2. **Split**: Trace \( s_r \) through \( T_{r-1} \) and split all the trapezoids intersected by \( s_r \)

3. **Merge**: Remove parts of vertical extensions “cut off” by \( s_r \) and merge the adjacent trapezoids
RIC – Analysis (I)

Apply configuration spaces!

- \( X \): the set \( S \) of segments
- \( \Pi \): set of all trapezoids \( \Box \) defined by segments of \( S \)
- \( D(\Box) \): the (at most 4) segments incident to the trapezoid \( \Box \)
- \( K(\Box) \): the set of segments intersecting \( \Box \)

RIC – Analysis (II)

Cost of step \( T_{r-1} \mapsto T_r \):

- **Find**: we’ll care for that later...
- **Split**: constant time per traced \( \Box \); \( \Box \) is replaced by at most 4 new trapezoids.

\[ \Rightarrow O(\text{number of removed trapezoids}) = O(\text{number of created trapezoids}) \]

- **Merge**: \( O(\text{number of trapezoids created in step Split}) \)

Analysis of Update \( T_{r-1} \mapsto T_r \) (I)

**Observation**: The number of trapezoids created by Split is at most twice as large as the number of new trapezoids in \( T_r \).

**Proof**: For every Merge operation above (below) \( s_r \), one new trapezoid below (above) \( s_r \) survives. It follows that at most half of the previously created trapezoids are not in \( T_r \).

\[ \Rightarrow \text{Complexity of Split and Merge is} \]

\[ O(|\{\Box \mid \Box \in T_r \setminus T_{r-1}\}|) = O(\deg(s_r, T_r)). \]

Analysis of Update \( T_{r-1} \mapsto T_r \) (II)

Configuration Spaces \( \Rightarrow \) expected value of \( \deg(s_r, T_r) \) is \( \leq \frac{2}{r}E(|T_r|) \).

- \( |T_r| \leq 6r \) (each \( \Box \) is incident to a segment endpoint, and each endpoint is charged by at most three segments).

- **Expected update cost** \( T_{r-1} \mapsto T_r \) is \( O(1) \)

- **Overall expected update cost** is \( O(n) \)
Realization of Find

- History approach: store all the trapezoids of \( T_r, r = 1 \ldots n \). \( \square \in T_{r-1} \setminus T_r \) has pointers to all \( \square' \in T_r \setminus T_{r-1} \) with \( \square \cap \square' \neq \emptyset \).

- At most 4 pointers per \( \square \)

- Location of segment endpoint \( p_r \) of \( s_r \): trace \( p_r \) through the history graph

Analysis of Find (I)

Assume \( p_r \) runs through a trapezoid \( \square \). Then there is \( j \leq r \) such that

- \( \square \in T_j \setminus T_{j-1} \)
- \( s_r \) intersects \( \square \)

\( \Rightarrow \) length of history path to \( p_r \)

\[ \leq \sum_{j=0}^{n-1} \sum_{\square \in T_j \setminus T_{j-1}} [s_r \in K(\square)] \]

\( \Rightarrow \) expected time for history searches is proportional to the expected number \( \sum_{r=0}^{n-1} K(r) \) of conflicts that appear during the algorithm.

Analysis of Find (II)

Configuration spaces \( \Rightarrow \)

\[
\sum_{r=1}^{n-1} K(r) \leq \sum_{r=1}^{n-1} (k_1(r) - k_2(r) + k_3(r)) \\
\leq d(n - 1)t_1 + d(d - 1)n \sum_{r=1}^{n-1} \frac{t_{r+1}}{r(r + 1)} - d^2 \sum_{r=1}^{n-1} \frac{t_{r+1}}{r + 1} \\
= O(n \log n),
\]

because

\( t_{r+1} = E(|T_r|) = O(r + 1) \).

Trapezoidal Map – Conclusion

Given a set \( S \) of \( n \) nonintersecting segments in the plane, its trapezoidal map \( T(S) \) can be computed in time

\[ O(n \log n). \]

(The assumption that segment endpoints have different \( x \)-coordinates can be achieved by comparing them lexicographically.)
Special Case: \( S \) forms simple polygon \( P \)

Trapezoidal Map \( \rightarrow \) Triangulation (I)

**Step 1:** Within each trapezoid, connect the two polygon vertices

Trapezoidal Map \( \rightarrow \) Triangulation (II)

**Step 2:** Triangulate the resulting \( x \)-monotone polygons separately, in total time \( O(n) \) (Exercise)

A fast method for the special case (I)

Runtime will be \( O(n \log^* n) \).

- \( \log^{(h)} n := \log \log \ldots \log n \), \( h \) times
- \( \log^* n := \max \{ h \mid \log^{(h)} n \geq 1 \} \)
- Example: \( \log^*(2^{65536}) = 5 \Rightarrow \log^* n < 5 \) “for all” \( n \).

**Definition:**

\[
N(h) := \left\lceil \frac{n}{\log^{(h)} n} \right\rceil, \quad 0 \leq h \leq \log^* n.
\]
A fast method for the special case (II)

Generalized history management: keep several histories and for each \( p \in P \) a pointer to the ‘history in charge’.

compute \( T_1 \) and initialize one history, in charge of all points

\[
\text{FOR } h = 1 \text{ TO } \log^* n \text{ DO}
\]

\[
\text{FOR } r = N(h-1) + 1 \text{ TO } N(h) \text{ DO}
\]

compute \( T_r \) from \( T_{r-1} \) (* as usual *)

\[
\text{END}
\]

(* Renew histories by tracing \( S \) through \( T_r \ *))

\[
\text{FOR ALL } \Box \in T_r \text{ containing an endpoint DO}
\]

make \( \Box \) the root of a history in charge of all the points it contains

\[
\text{END}
\]

\[
\text{END}
\]

\[
\text{FOR } r = N(\log^* n) + 1 \text{ TO } n \text{ DO}
\]

compute \( T_r \) from \( T_{r-1} \) (* as usual *)

\[
\text{END}
\]

Analysis of the fast method (I)

- **Split** and **Merge** proceed as before in expected time \( O(n) \)

- **Find** will be faster on average, but we have \( \log^* n \) additional **Trace** steps

Analysis of **Find** (I)

In phase \( h \), every trapezoid traced during the history search corresponds to a trapezoid that

- has been present in the beginning of phase \( h \) (root of a hierarchy) or was created during phase \( h \)

- is in conflict with a segment inserted in phase \( h \)

\( \Rightarrow \) expected cost of history search is at most proportional to \( n + K_h \)

\[
K_h := \sum_{r=N(h-1)+1}^{N(h)} \sum_{\Box \in T_r \setminus T_{r-1}} |K(\Box) \cap S_{N(h)}|
\]

Analysis of **Find** (II)

For fixed \( X := S_{N(h)} \), \( E(K_h) \) is the expected number of conflicts appearing in steps \( N(h-1) + 1 \) to \( N(h) \) when \( T(X) \) is computed.

\[
i := N(h - 1) + 1, \quad j := N(h) - 1.
\]

\( \text{Configuration spaces analysis } \Rightarrow \)

\[
E(K_h) \leq \sum_{r=i}^{j} (k_1 - k_2 + k_3)
\]

\[
\leq \frac{d(j + 1 - i)}{i} t_i + 
\]

\[
d(d-1)(j+1) \sum_{r=i}^{j} \frac{t_{r+1}}{r(r+1)} - 
\]

\[
d^2 \sum_{r=i}^{j} \frac{t_{r+1}}{r+1}.
\]
Analysis of \textbf{Find} (III)

Recall:
\[ t_{r+1} = O(r + 1). \]

Then
\[
E(K_h) = O(N(h) - N(h-1)) + O\left(\sum_{r=N(h-1)+1}^{N(h)-1} \frac{1}{r} \right) = O(N(h) + N(h) \log\frac{N(h)}{N(h-1)}) = O(N(h) + N(h) \log(h)n) = O(n).
\]

(This also holds for a random set \(S_{\mathcal{N}(h)}\) and for the last insertion phase \((i = N(\log^* n)+1, j = n-1)\).) The total cost for \textbf{Find} over all \(h\) is then \(O(n \log^* n)\).

Analysis of \textbf{Trace} (I)

The expected cost \(T_h\) of tracing \(S\) through \(T_{\mathcal{N}(h)}\) is at most proportional to the expected number of conflicts between trapezoids in \(T_{\mathcal{N}(h)}\) and segments in \(S\), which is
\[
\frac{1}{N(h)} \sum_{R \subseteq S, |R| = N(h)} \sum_{y \in S \setminus R} |\{\square \in T(R) \mid y \in K(\square)\}|.
\]

Up to a missing factor of \(d/N(h)\) this is exactly the bound for the expected number \(K_{\mathcal{N}(h)}\) of new conflicts when \(s_{\mathcal{N}(h)}\) is inserted that we derived from the configuration spaces.

Analysis of \textbf{Trace} (II)

<table>
<thead>
<tr>
<th>configuration spaces</th>
<th>here</th>
</tr>
</thead>
<tbody>
<tr>
<td>(k_1)</td>
<td>(\frac{d}{r(r+1)}(n-r)t_r)</td>
</tr>
<tr>
<td>(k_2)</td>
<td>(\frac{d}{r(r+1)}(n-r)t_{r+1})</td>
</tr>
<tr>
<td>(k_3)</td>
<td>(\frac{d^2}{r(r+1)}(n-r)t_{r+1})</td>
</tr>
</tbody>
</table>

Setting \(r = N(h)\), we obtain \(T_h = k_1 - k_2 + k_3\) as
\[
T_h \leq (n - N(h))t_{N(h)} - (n - N(h))t_{N(h)+1} + \frac{d}{N(h)+1}(n - N(h))t_{N(h)+1} = O(n(t_{N(h)} - t_{N(h)+1}) + n) = O(n),
\]

because \(t_{N(h)} \leq t_{N(h)+1}\).

The total cost for \textbf{Trace} over all \(h\) is then \(O(n \log^* n)\).

Fast Trapezoidal Map – Conclusion

Given a simple polygon \(P\) with \(n\) vertices in the plane, its trapezoidal map \(T(P)\) can be computed in time
\[ O(n \log^* n). \]

(This is not optimal, because Chazelle has given a (rather complicated) \(O(n)\) algorithm for the problem.)