Computational Geometry: Intersection Search

Joseph S. B. Mitchell
Stony Brook University
Intersection Search

- **Input**: A set \( S \) of geometric objects (segments, polygons, disks, solid models, etc)
Intersection Search

- **Versions of the problem:**
  - **DETECT:** Answer yes/no: Are there any intersections among the objects $S$? (if “yes”, then we may insist on returning a witness)
  - **REPORT:** Output all pairs of objects that intersect
  - **COMPUTE:** Compute the common intersection of all objects
  - **COUNT:** How many pairs intersect?
  - **QUERY:** Preprocess $S$ to support fast queries of the form “Does object $Q$ intersect any member of $S$?” (or “Report all intersections with $Q$”)

May also want to insert/delete in $S$, or allow objects of $S$ to move.
Warm Up: 1D Problem

- Given n segments (intervals) on a line
- DETECT: $O(n \log n)$, based on sorting
- REPORT: $O(k+n \log n)$, based on sorting, then marching through, left to right
- Lower bound to DETECT: $\Omega(n \log n)$, from ELEMENT UNIQUENESS

---

Element Uniqueness
Input: \{x_1, x_2, ..., x_n\}
Are they distinct? (yes/no)
$\Omega(n \log n)$
Intersection Search: Segments

- Segment intersection: Given a set $S$ of $n$ line segments in the plane, determine:
  - Does some pair intersect? (DETECT)
  - Compute all points of intersection (REPORT)

Naïve: $O(n^2)$

CG: $O(n \log n)$ DETECT, $O(k+n \log n)$ REPORT

Lower Bound to DETECT: $\Omega(n \log n)$, from ELEMENT UNIQUENESS

Element Uniqueness
Input: $\{x_1, x_2, \ldots, x_n\}$
Are they distinct? (yes/no)
$\Omega(n \log n)$
**Primitive Computation**

- Does segment $ab$ intersect segment $cd$?
  - Types of “intersect”
  - Test using “Left” tests (sign of a cross product (determinant), which determines the orientation of 3 points)
    
    \[
    \text{Left}(a, b, c) = \text{TRUE} \iff ab \times ac > 0
    \]
  - Time $O(1)$ (“constant”)

\[
\begin{vmatrix}
\hat{i} & \hat{j} & \hat{k} \\
A_0 & A_1 & A_2 \\
B_0 & B_1 & B_2 \\
\end{vmatrix} = (A_1 B_2 - A_2 B_1)\hat{i} + (A_2 B_0 - A_0 B_2)\hat{j} + (A_0 B_1 - A_1 B_0)\hat{k}.
\]
Bentley-Ottmann Sweep

- **Main idea:** Process events in order of discovery as a horizontal line, L, "sweeps" over the scene, from top to bottom.

- **Two data structures:**
  - SLS: Sweep Line Status: left-to-right ordering of the segments intersecting L
  - EQ: Event Queue
Two data structures:
- **SLS**: Sweep Line Status: left-to-right ordering of the segments intersecting \( L \)
- **EQ**: Event Queue

Initialize:
- \( SLS = \emptyset \)
- \( EQ = \text{sorted list of segment endpoints} \) \( O(n \log n) \)

How to store:
- \( SLS \): balanced binary search tree
  (dictionary, support \( O(\log n) \) insert, delete, search)
- \( EQ \): priority queue (e.g., heap)
Event Handling

- **Hit top endpt of e**
  
  \[ O(\log n) \]
  
  Find segs a and b left/right of e in SLS. **Insert** e into SLS. Test(a,e), Test(b,e) and insert crossings (if any) in EQ

- **Hit bottom endpt of e**
  
  \[ O(\log n) \]
  
  Find segs a and b left/right of e in SLS. **Delete** e from SLS. Test(a,b), and insert crossing (if any) in EQ

- **Hit crossing point \( e \cap f \)** (only needed in REPORT)
  
  \[ O(\log n) \]
  
  Exchange e, f in SLS. Test(a,f), Test(b,e) and insert crossings (if any) in EQ
Algorithm Analysis

**Invariants** of algorithm:
- SLS is correctly ordered.
- Test(a,b) for intersection is done whenever segments a and b become adjacent in the SLS order

- Discovered crossings are inserted into EQ
  
  (for REPORT; for DETECT, stop at first detected crossing)

- **Claim**: All crossings are discovered

- **Time**: $O(n \log n)$ to DETECT  ($O(n)$ events @ $O(\log n)$ )
- **Time**: $O((n+k) \log n)$ to REPORT  ($O(n+k)$ events @ $O(\log n)$ )

- **Example Applet**

  $k = \# \text{ crossings} = \text{output size}$
Basic (original) version of B-O sweep:
Naively, the EQ has size at most $O(n+k)$, since we store the SLS ($O(n)$) and the EQ (size $2n+k$)

Better bounds on size of EQ: $O(n \log^2 n)$
[Pach and Sharir]
Modified B-O Sweep:

- Any time 2 segments STOP being adjacent in SLS, remove from EQ the corresponding crossing point (if any); we will re-insert the crossing point again (at least once) before the actual crossing event.

- Note: Now the EQ has at most n-1 crossing events (and at most 2n endpoint events)
<table>
<thead>
<tr>
<th>Event</th>
<th>Event Queue, Q</th>
<th>Sweep Status, $\mathcal{L}$</th>
<th>Intersection Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-$</td>
<td></td>
<td>$(\emptyset)$</td>
<td>$s_1 \cap s_0 = \emptyset$</td>
</tr>
<tr>
<td>$a_0$</td>
<td>$(a_0 a_1 a_2 a_3 a_4 a_5 a_6 b_1 b_2 b_3 b_5 b_4 b_0 b_6)$</td>
<td>$(s_0)$</td>
<td>$s_2 \cap s_1 = x_{12}$</td>
</tr>
<tr>
<td>$a_1$</td>
<td>$(a_1 a_2 a_3 a_4 a_5 a_6 b_1 b_2 b_3 b_5 b_4 b_0 b_6)$</td>
<td>$(s_1, s_0)$</td>
<td>$s_1 \cap s_3 = \emptyset$, $s_3 \cap s_0 = x_{03}$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$(a_2 a_3 a_4 a_5 a_6 b_1 b_2 b_3 b_5 b_4 b_0 b_6)$</td>
<td>$(s_2, s_1, s_0)$</td>
<td>$s_2 \cap s_3 = x_{23}$</td>
</tr>
<tr>
<td>$a_3$</td>
<td>$(a_3 x_{12} a_4 a_5 a_6 b_1 b_2 b_3 b_5 b_4 b_0 b_6)$</td>
<td>$(s_2, s_1, s_3, s_0)$</td>
<td>$s_3 \cap s_4 = x_{34}$, $s_4 \cap s_0 = \emptyset$</td>
</tr>
<tr>
<td>$a_4$</td>
<td>$(a_4 a_5 a_6 x_{23} b_1 b_2 b_3 b_5 b_4 b_0 b_6)$</td>
<td>$(s_1, s_2, s_3, s_0)$</td>
<td>$s_5 \cap s_1 = \emptyset$</td>
</tr>
<tr>
<td>$a_5$</td>
<td>$(a_5 x_{34} a_6 x_{23} b_1 [x_{03}] b_2 b_3 b_5 b_4 b_0 b_6)$</td>
<td>$(s_5, s_1, s_2, s_3, s_4, s_0)$</td>
<td>$s_2 \cap s_4 = x_{24}$, $s_3 \cap s_0 = x_{03}$, $s_6 \cap s_5 = \emptyset$</td>
</tr>
<tr>
<td>$a_6$</td>
<td>$(a_6 x_{24} [x_{23}] b_1 x_{03} b_2 b_3 b_5 b_4 b_0 b_6)$</td>
<td>$(s_5, s_1, s_2, s_4, s_3, s_0)$</td>
<td>$s_1 \cap s_4 = \emptyset$, $s_2 \cap s_3 = x_{23}$</td>
</tr>
<tr>
<td>$x_{24}$</td>
<td>$(a_6 x_{24} [x_{23}] b_1 x_{03} b_2 b_3 b_5 b_4 b_0 b_6)$</td>
<td>$(s_6, s_5, s_1, s_2, s_4, s_3, s_0)$</td>
<td>$s_4 \cap s_3 = \emptyset$, $s_2 \cap s_0 = x_{02}$</td>
</tr>
<tr>
<td>$x_{23}$</td>
<td>$(b_1 x_{02} [x_{03}] b_2 b_3 b_5 b_4 b_0 b_6)$</td>
<td>$(s_6, s_5, s_1, s_4, s_3, s_2, s_0)$</td>
<td></td>
</tr>
</tbody>
</table>

The diagram shows a graph with points labeled 0 to 6 connected by lines.
<table>
<thead>
<tr>
<th>Event</th>
<th>Event Queue, Q</th>
<th>Sweep Status, L</th>
<th>Intersection Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>((a_0a_1a_2a_3a_4a_5a_6b_1b_2b_3b_5b_4b_0b_6))</td>
<td>()</td>
<td>(s_1 \cap s_0 = \emptyset)</td>
</tr>
<tr>
<td>(a_0)</td>
<td>((a_1a_2a_3a_4a_5a_6b_1b_2b_3b_5b_4b_0b_6))</td>
<td>((s_0))</td>
<td>(s_2 \cap s_1 = x_{12})</td>
</tr>
<tr>
<td>(a_1)</td>
<td>((a_2a_3a_4a_5a_6b_1b_2b_3b_5b_4b_0b_6))</td>
<td>((s_1, s_0))</td>
<td>(s_1 \cap s_3 = \emptyset, s_3 \cap s_0 = x_{03})</td>
</tr>
<tr>
<td>(a_2)</td>
<td>((a_3x_{12}a_4a_5a_6b_1b_2b_3b_5b_4b_0b_6))</td>
<td>((s_2, s_1, s_0))</td>
<td>(s_2 \cap s_3 = x_{23})</td>
</tr>
<tr>
<td>(a_3)</td>
<td>((x_{12}a_4a_5a_6b_1x_{03}b_2b_3b_5b_4b_0b_6))</td>
<td>((s_2, s_1, s_3, s_0))</td>
<td>(s_3 \cap s_4 = x_{34}, s_4 \cap s_0 = \emptyset)</td>
</tr>
<tr>
<td>(x_{12})</td>
<td>((a_4a_5a_6x_{23}b_1x_{03}b_2b_3b_5b_4b_0b_6))</td>
<td>((s_1, s_2, s_3, s_0))</td>
<td>(s_3 \cap s_1 = \emptyset)</td>
</tr>
<tr>
<td>(a_4)</td>
<td>((a_5x_{34}a_6x_{23}b_1[x_{03}]b_2b_3b_5b_4b_0b_6))</td>
<td>((s_5, s_1, s_2, s_3, s_4, s_0))</td>
<td>(s_2 \cap s_4 = x_{24}, s_3 \cap s_0 = x_{03})</td>
</tr>
<tr>
<td>(a_5)</td>
<td>((x_{34}a_6x_{23}b_1[x_{03}]b_2b_3b_5b_4b_0b_6))</td>
<td>((s_5, s_1, s_2, s_4, s_3, s_0))</td>
<td>(s_6 \cap s_5 = \emptyset)</td>
</tr>
<tr>
<td>(x_{34})</td>
<td>((a_6x_{24}[x_{23}]b_1x_{03}b_2b_3b_5b_4b_0b_6))</td>
<td>((s_6, s_5, s_1, s_2, s_4, s_3, s_0))</td>
<td>(s_1 \cap s_4 = \emptyset, s_2 \cap s_3 = x_{23})</td>
</tr>
<tr>
<td>(a_6)</td>
<td>((x_{24}[x_{23}]b_1x_{03}b_2b_3b_5b_4b_0b_6))</td>
<td>((s_6, s_5, s_1, s_4, s_2, s_3, s_0))</td>
<td>(s_4 \cap s_3 = \emptyset, s_2 \cap s_0 = x_{02})</td>
</tr>
<tr>
<td>(x_{24})</td>
<td>((x_{23}b_1x_{03}b_2b_3b_5b_4b_0b_6))</td>
<td>((s_6, s_5, s_4, s_3, s_2, s_0))</td>
<td>(s_5 \cap s_4 = x_{45})</td>
</tr>
<tr>
<td>(x_{23})</td>
<td>((b_1x_{02}[x_{03}]b_2b_3b_5b_4b_0b_6))</td>
<td>((s_6, s_5, s_4, s_3, s_0, s_2))</td>
<td>(s_3 \cap s_0 = x_{03})</td>
</tr>
<tr>
<td>(b_1)</td>
<td>((x_{02}[x_{03}]x_{45}b_2b_3b_5b_4b_0b_6))</td>
<td>((s_6, s_5, s_4, s_0, s_3, s_2))</td>
<td>(s_4 \cap s_0 = \emptyset, s_3 \cap s_2 = x_{23})</td>
</tr>
<tr>
<td>(x_{02})</td>
<td>((x_{03}x_{45}b_2b_3b_5b_4b_0b_6))</td>
<td>((s_6, s_5, s_4, s_0, s_2))</td>
<td>(s_6 \cap s_4 = x_{46}, s_5 \cap s_0 = x_{05})</td>
</tr>
<tr>
<td>(x_{03})</td>
<td>((x_{45}b_2b_3b_5b_4b_0b_6))</td>
<td>((s_6, s_4, s_5, s_0))</td>
<td>(s_4 \cap s_0 = \emptyset)</td>
</tr>
<tr>
<td>(x_{45})</td>
<td>((b_2b_3x_{05}b_5x_{46}b_4b_0b_6))</td>
<td>((s_6, s_4, s_5, s_0, s_3))</td>
<td>(s_6 \cap s_0 = \emptyset)</td>
</tr>
<tr>
<td>(b_2)</td>
<td>((b_3x_{05}b_5x_{46}b_4b_0b_6))</td>
<td>((s_6, s_4, s_5, s_0, s_2))</td>
<td>(s_6 \cap s_0 = \emptyset)</td>
</tr>
<tr>
<td>(b_3)</td>
<td>((x_{05}b_5x_{46}b_4b_0b_6))</td>
<td>((s_6, s_4, s_5, s_0, s_2))</td>
<td>(s_6 \cap s_0 = \emptyset)</td>
</tr>
<tr>
<td>(x_{05})</td>
<td>((b_5x_{46}b_4b_0b_6))</td>
<td>((s_6, s_4, s_5, s_0))</td>
<td>(s_6 \cap s_0 = \emptyset)</td>
</tr>
<tr>
<td>(b_5)</td>
<td>((x_{46}b_4b_0b_6))</td>
<td>((s_6, s_0))</td>
<td>(s_6 \cap s_0 = \emptyset)</td>
</tr>
<tr>
<td>(x_{46})</td>
<td>((b_4b_0b_6))</td>
<td>((s_6))</td>
<td>(s_6 \cap s_0 = \emptyset)</td>
</tr>
<tr>
<td>(b_4)</td>
<td>((b_0b_6))</td>
<td>((s_6))</td>
<td>(s_6 \cap s_0 = \emptyset)</td>
</tr>
<tr>
<td>(b_0)</td>
<td>((b_6))</td>
<td>((s_6))</td>
<td>(s_6 \cap s_0 = \emptyset)</td>
</tr>
</tbody>
</table>

(Note that \(s_4\) and \(s_5\) intersect at a point with y-coordinate just above 8 (namely, 1025/128).)
- **Optimal REPORT algorithms exist:** (Complex)
  - $O(k + n \log n)$, $O(k+n)$ space (working memory)  
    - [Chazelle-Edelsbrunner]
  - $O(k + n \log n)$, $O(n)$ space  
    - [Balaban]

- **Special Case: REPORT for horiz/vert segs**
  - Bentley-Ottmann sweep: $O(k + n \log n)$ (optimal!)
    
    All crossings happen when $L$ hits a horizontal segment, $s_i$; just locate ($O(\log n)$) the endpoint and walk along horizontal segment in SLS to report all $k_i$ vertical segments crossed along $s_i$.

- **Special case: Simplicity testing**
  - $O(n)$, from Chazelle triangulation
Sweeping applies also to

- Unions
- Intersections
- Arrangements
Practical Methods, 3D

- **Uniform grid**
  
  With each pixel, store a list of objects intersecting it. Do brute force on pixel-by-pixel basis.

- **Quadtrees**

- **Bounding volume hierarchies**

See Samet books, SAND website
Bounding Volume Hierarchies

Input: Set $S$ of objects.

Bounding volume
QuickCD: Collision Detection
The 2-Box Cover Problem

• Find “smallest” (tightest fitting) pair of bounding boxes

• \textbf{Motivation:}
  - Best outer approximation
  - Bounding volume hierarchies
Bounding Volume Hierarchy

BV-tree: Level 0

k-dops
BV-tree: Level 1

6-dops

18-dops

14-dops

26-dops
BV-tree: Level 2
BV-tree: Level 5
BV-tree: Level 8