Search and Intersection

O’Rourke, Chapter 7
Anouncements

- Assignment 2 has been graded
- Assignment 3 has been posted
Outline

• Polygon Intersection
  ◦ Segment Intersection

• Convex Polygon Intersection
Polygon Intersection

Given polygons $P$ and $Q$, in the worst case they can intersect in $O(|P| \cdot |Q|)$ positions.

Can we compute the intersection in an output sensitive manner?
Segment Intersection

Given a set of line segments, find crossings.

**Approach:**
Assume general position.
Use sweep line algorithm.
Segment Intersection

Sweep line algorithm:
• Initialize queue with end-points sorted by height.
• Initialize list of events.
• Advance:
  ◦ Add/remove segments
  ◦ Adjust event list
  ◦ Test for neighboring intersections
  ◦ Adjust queue

\[ Q = (a_1, c_1, b_1, d_1, c_2, a_2, d_2, b_2) \]
\[ L = \emptyset \]
Segment Intersection

Sweep line algorithm:

- Initialize queue with end-points sorted by height.
- Initialize list of events.
- Advance:
  - Add/remove segments
  - Adjust event list
  - Test for neighboring intersections
  - Adjust queue

\[
Q = (c_1, b_1, d_1, c_2, a_2, d_2, b_2)
\]
\[
L = (a)
\]
Segment Intersection

Sweep line algorithm:
- Initialize queue with end-points sorted by height.
- Initialize list of events.
- Advance:
  - Add/remove segments
  - Adjust event list
  - Test for neighboring intersections
  - Adjust queue

\[ Q = (b_1, d_1, ac, c_2, a_2, d_2, b_2) \]
\[ L = (a, c) \]
Segment Intersection

Sweep line algorithm:
• Initialize queue with end-points sorted by height.
• Initialize list of events.
• Advance:
  ○ Add/remove segments
  ○ Adjust event list
  ○ Test for neighboring intersections
  ○ Adjust queue

\[ Q = (ab, d_1, ac, c_2, a_2, d_2, b_2) \]
\[ L = (a, b, c) \]
Segment Intersection

Sweep line algorithm:
• Initialize queue with end-points sorted by height.
• Initialize list of events.
• Advance:
  ◦ Add/remove segments
  ◦ Adjust event list
  ◦ Test for neighboring intersections
  ◦ Adjust queue

\[ Q = (d_1, ac, c_2, a_2, d_2, b_2) \]
\[ L = (b, a, c) \]
Segment Intersection

Sweep line algorithm:
- Initialize queue with end-points sorted by height.
- Initialize list of events.
- Advance:
  - Add/remove segments
  - Adjust event list
  - Test for neighboring intersections
  - Adjust queue

\[ Q = (ac, c_2, a_2, d_2, b_2) \]
\[ L = (b, a, c, d) \]
**Segment Intersection**

**Sweep line algorithm:**
- Initialize queue with end-points sorted by height.
- Initialize list of events.
- **Advance:**
  - Add/remove segments
  - Adjust event list
  - Test for neighboring intersections
  - Adjust queue

\[
Q = (c_2, ad, a_2, d_2, b_2) \\
L = (b, c, a, d)
\]
Segment Intersection

Sweep line algorithm:

- Initialize queue with end-points sorted by height.
- Initialize list of events.
- Advance:
  - Add/remove segments
  - Adjust event list
  - Test for neighboring intersections
  - Adjust queue

\[ Q = (ad, a_2, d_2, b_2) \]
\[ L = (b, a, d) \]
Segment Intersection

Sweep line algorithm:
- Initialize queue with end-points sorted by height.
- Initialize list of events.
- Advance:
  - Add/remove segments
  - Adjust event list
  - Test for neighboring intersections
  - Adjust queue

\[ Q = (bd, a_2, d_2, b_2) \]
\[ L = (b, d, a) \]
Segment Intersection

Sweep line algorithm:

• Initialize queue with end-points sorted by height.
• Initialize list of events.
• Advance:
  ◦ Add/remove segments
  ◦ Adjust event list
  ◦ Test for neighboring intersections
  ◦ Adjust queue

\[ Q = (a_2, d_2, b_2) \]
\[ L = (d, b, a) \]
Segment Intersection

Sweep line algorithm:

- Initialize queue with end-points sorted by height.
- Initialize list of events.
- Advance:
  - Add/remove segments
  - Adjust event list
  - Test for neighboring intersections
  - Adjust queue

\[ Q = (d_2, b_2) \]
\[ L = (d, b) \]
Segment Intersection

Sweep line algorithm:
• Initialize queue with end-points sorted by height.
• Initialize list of events.
• Advance:
  ◦ Add/remove segments
  ◦ Adjust event list
  ◦ Test for neighboring intersections
  ◦ Adjust queue

\[ Q = (b_2) \]
\[ L = (b) \]
Segment Intersection

Sweep line algorithm:

- Initialize queue with end-points sorted by height.
- Initialize list of events.
- Advance:
  - Add/remove segments
  - Adjust event list
  - Test for neighboring intersections
  - Adjust queue

\[
Q = \emptyset \\
L = \emptyset
\]
**Segment Intersection**

**Sweep line algorithm:**
- Initialize queue with end-points sorted by height.
- Initialize list of events.
- Advance:
  - Add/remove segments
  - Adjust event list
  - Test for neighboring intersections
  - Adjust queue

With the right data-structures, this has complexity $O((n + k) \log n)$, with $k$ the number of intersections.
Sweep line algorithm:
A similar approach gives polygon intersection.
Need to track:
- Span labels
- Polygon chains

\[ Q = (q_1, p_1, p_4, p_7, p_5, q_4, q_2, p_6, p_2, q_3, p_3) \]
\[ L = \emptyset \]
Polygon Intersection

Sweep line algorithm:
A similar approach gives polygon intersection.
Need to track:
- Span labels
- Polygon chains

\[ Q = (p_1, p_4, p_7, p_5, q_4, q_2, p_6, p_2, q_3, p_3) \]
\[ L = (\emptyset - q_{12} - Q - q_{41} - \emptyset) \]
Polygon Intersection

Sweep line algorithm:
A similar approach gives polygon intersection.
Need to track:
- Span labels
- Polygon chains

\[ Q = (p_{71} q_{12}, p_4, p_7, p_5, q_4, q_2, p_6, p_2, q_3, p_3) \]
\[ L = (\emptyset - p_{12} - P - p_{71} - \emptyset - q_{12} - Q - q_{41} - \emptyset) \]
Polygon Intersection

Sweep line algorithm:
A similar approach gives polygon intersection.

Need to track:
- Span labels
- Polygon chains

\[ Q = (p_4, p_7, q_4, p_5, q_2, p_6, q_4, p_3, p_3) \]
\[ L = (\emptyset - p_{12} - P - q_{12} - PQ - p_{71} - Q - q_{41} - \emptyset) \]
Polygon Intersection

Sweep line algorithm:
A similar approach gives polygon intersection.
Need to track:
- Span labels
- Polygon chains

\[ Q = (p_{71}q_{41}, p_7, p_{45}q_{12}, p_5, q_4, q_2, p_6, p_2, q_3, p_3) \]
\[ L = (\emptyset - p_{12} - P - p_{34} - \emptyset - p_{45} - P - q_{12} - PQ - p_{71} - Q - q_{41} - \emptyset) \]
Sweep line algorithm:
A similar approach gives polygon intersection.

Need to track:

- Span labels
- Polygon chains

\[ Q = (p_7, p_{45}q_{12}, p_5, q_4, q_2, p_6, p_2, q_3, p_3) \]
\[ L = (\emptyset - p_{12} - P - p_{34} - \emptyset - p_{45} - P - q_{12} - PQ - q_{41} - P - p_{71} - \emptyset) \]
Sweep line algorithm:

A similar approach gives polygon intersection.

Need to track:
- Span labels
- Polygon chains

\[ Q = (p_{45}q_{12}, p_5, q_4, q_2, p_6, p_2, q_3, p_3) \]
\[ L = (\emptyset - p_{12} - P - p_{34} - \emptyset - p_{45} - P - q_{12} - PQ - q_{41} - \emptyset - p_{67} - \emptyset) \]
Polygon Intersection

Sweep line algorithm:
A similar approach gives polygon intersection.
Need to track:
- Span labels
- Polygon chains

\[ Q = (p_5, p_{34}, q_{12}, q_4, q_2, p_6, p_2, q_3, p_3) \]
\[ L = (\phi - p_{12} - P - p_{34} - \phi - q_{12} - Q - p_{45} - PQ - q_{41} - P - p_{67} - \phi) \]
Polygon Intersection

Sweep line algorithm:
A similar approach gives polygon intersection.
Need to track:
- Span labels
- Polygon chains

\[ Q = (p_{34}q_{12}, q_4, q_2, p_6, p_2, q_3, p_3) \]
\[ L = (\emptyset - p_{12} - P - p_{34} - \emptyset - q_{12} - Q - p_{56} - PQ - q_{41} - P - p_{67} - \emptyset) \]
Polygon Intersection

Sweep line algorithm:
A similar approach gives polygon intersection.
Need to track:
- Span labels
- Polygon chains

\[ Q = (q_4, q_2, p_6, p_2, q_3, p_3) \]
\[ L = (\emptyset - p_{12} - P - q_{12} - PQ - p_{34} - Q - p_{56} - PQ - q_{41} - P - p_{67} - \emptyset) \]
Polygon Intersection

**Sweep line algorithm:**
A similar approach gives polygon intersection.

Need to track:
- Span labels
- Polygon chains

\[ Q = (p_{67}q_{34}, q_2, p_6, p_2, q_3, p_3) \]
\[ L = (\emptyset - p_{12} - P - q_{12} - PQ - p_{34} - Q - p_{56} - PQ - q_{34} - P - p_{67} - \emptyset) \]
Sweep line algorithm:
A similar approach gives polygon intersection.

Need to track:
- Span labels
- Polygon chains

\[ Q = (q_2, p_6, p_2, q_3, p_3) \]
\[ L = (\emptyset - p_{12} - P - q_{12} - PQ - p_{34} - Q - p_{56} - PQ - p_{67} - Q - q_{34} - \emptyset) \]
Sweep line algorithm:
A similar approach gives polygon intersection.

Need to track:
- Span labels
- Polygon chains

\[ Q = (p_{34}q_{23}, p_6, p_2, q_3, p_3) \]
\[ L = (\emptyset - p_{12} - P - q_{23} - PQ - p_{34} - Q - p_{56} - PQ - p_{67} - Q - q_{34} - \emptyset) \]
Polygon Intersection

**Sweep line algorithm:**
A similar approach gives polygon intersection.

Need to track:

- Span labels
- Polygon chains

\[ Q = (p_6, p_2, q_3, p_3) \]
\[ L = (\emptyset - p_{12} - P - p_{34} - \emptyset - q_{23} - Q - p_{56} - PQ - p_{67} - Q - q_{34} - \emptyset) \]
Polygon Intersection

Sweep line algorithm:
A similar approach gives polygon intersection.

Need to track:
- Span labels
- Polygon chains

\[ Q = (p_2, q_3, p_3) \]
\[ L = (\emptyset - p_{12} - P - p_{34} - \emptyset - q_{23} - Q - q_{34} - \emptyset) \]
Sweep line algorithm:
A similar approach gives polygon intersection.
Need to track:

- Span labels
- Polygon chains

\[ L = (\emptyset - p_{23} - P - p_{34} - \emptyset - q_{23} - Q - q_{34} - \emptyset) \]
\[ Q = (q_3, p_3) \]
Polygon Intersection

Sweep line algorithm:
A similar approach gives polygon intersection.
Need to track:

- Span labels
- Polygon chains

\[ Q = (p_3) \]
\[ L = (\emptyset - p_{23} - P - p_{34} - \emptyset) \]
Polygon Intersection

Sweep line algorithm:
A similar approach gives polygon intersection.
Need to track:
- Span labels
- Polygon chains

\[ Q = \emptyset \]
\[ L = \emptyset \]
Polygon Intersection

Sweep line algorithm:
A similar approach gives polygon intersection.

Need to track:
- Span labels
- Polygon chains

This has complexity $O((|P| + |Q| + k) \log(|P| + |Q|))$, with $k$ the number of intersections.
Outline

• Polygon Intersection

• Convex Polygon Intersection
Convex Polygon Intersection

Notation:

Given a (directed) edge \( f = (a, b) \) we refer to \( b \) as the head of \( f \).

Given edges \( e \) and \( f \) we say that \( e \) is interior / exterior to \( f \) if the head of \( e \) is left / right of \( f \).

Given edges \( e \) and \( f \) we say that \( e \) aims at \( f \) if:

- \((e,f)\) is CW and \( e \) is exterior to \( f \), or
- \((e,f)\) is CCW and \( e \) is interior to \( f \).
Convex Polygon Intersection

Given convex polygons $P$ and $Q$, find the (convex) intersection $P \cap Q$.

Approach:

Find intersections between $P$ and $Q$ and track which polygon is interior between successive crossings.
Convex Polygon Intersection

Given convex polygons $P$ and $Q$, find the (convex) intersection $P \cap Q$.

Greedy Algorithm:

Advance an edge $e$ if it aims at the line through the other edge $f$. 
Convex Polygon Intersection

- Choose edges \( e \in P \) and \( f \in Q \).
- While not done:
  - If neither/both edge aim at each other:
    - If \( f \) interior to \( e \): \( e ++ \)
    - Else if \( e \) interior to \( f \): \( f ++ \)
    - Else: exit( "Can't happen" )
  - Else if \( e \) aims at \( f \): \( e ++ \)
  - Else if \( f \) aims at \( e \): \( f ++ \)
Convex Polygon Intersection

- Choose edges $e \in P$ and $f \in Q$.
- While not done:
  - If neither/both edge aim at each other:
    » If $f$ interior to $e$: $e++$
    » Else if $e$ interior to $f$: $f++$
    » Else: exit(“Can’t happen”)
  - Else if $e$ aims at $f$: $e++$
  - Else if $f$ aims at $e$: $f++$
Claim:

This algorithm outputs the correct solution and iterates at most $2(|P| + |Q|)$ times.
**Convex Polygon**

**Sub-Claim 1:**

The algorithm finds at least one intersection point.*

<table>
<thead>
<tr>
<th></th>
<th>$\text{aim}(e, f)$</th>
<th>$\neg\text{aim}(e, f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{aim}(f, e)$</td>
<td>exterior</td>
<td>$f$</td>
</tr>
<tr>
<td>$\neg\text{aim}(f, e)$</td>
<td>$e$</td>
<td>exterior</td>
</tr>
</tbody>
</table>

*Assume $P$ and $Q$ intersect non-degenerately. (i.e. At most one point of intersection in the interior of an edge.)
Proof (Sub-Claim 1):

Assume to the contrary.

After $|P| + |Q|$ iterations we will have completed a cycle of either $P$ or $Q$, w.l.o.g. assume $Q$.

$\Rightarrow$ At some edge $f \in Q$ the polygon $P$ passes from outside $Q$ to inside.

<table>
<thead>
<tr>
<th></th>
<th>aim$(e, f)$</th>
<th>$\neg$aim$(e, f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>aim$(f, e)$</td>
<td>exterior</td>
<td>$f$</td>
</tr>
<tr>
<td>$\neg$aim$(f, e)$</td>
<td>$e$</td>
<td>exterior</td>
</tr>
</tbody>
</table>
$P$ passes from outside $Q$ to inside at $f$.

Case 1 ($e$ is exterior):

- If $e$ does not aim at $f$:
  
  ⇒ $f$ is interior and $(f, e)$ is CW
  
  ⇒ $f$ cannot aim at $e$
  
  ⇒ Advance $e$
Convex Polygon

<table>
<thead>
<tr>
<th>aim(e, f)</th>
<th>−aim(e, f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>aim(f, e)</td>
<td>exterior</td>
</tr>
<tr>
<td>−aim(f, e)</td>
<td>e</td>
</tr>
</tbody>
</table>

$P$ passes from outside $Q$ to inside at $f$.

Case 1 ($e$ is exterior):

- If $e$ does not aim at $f$:
  - $f$ is interior and $(f, e)$ is CW
  - $f$ cannot aim at $e$
  - Advance $e$
**Convex Polygon Intersection**

<table>
<thead>
<tr>
<th></th>
<th>( \text{aim}(e, f) )</th>
<th>( \neg \text{aim}(e, f) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{aim}(f, e) )</td>
<td>exterior</td>
<td>( f )</td>
</tr>
<tr>
<td>( \neg \text{aim}(f, e) )</td>
<td>( e )</td>
<td>exterior</td>
</tr>
</tbody>
</table>

\( P \) passes from outside \( Q \) to inside at \( f \).

**Case 1 (\( e \) is exterior):**

- If \( e \) does not aim at \( f \):
  - \( f \) is interior and \((f, e)\) is CW
  - \( f \) cannot aim at \( e \)
  - Advance \( e \)
Convex Polygon

<table>
<thead>
<tr>
<th></th>
<th>aim(e, f)</th>
<th>¬aim(e, f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>aim(f, e)</td>
<td>exterior</td>
<td>f</td>
</tr>
<tr>
<td>¬aim(f, e)</td>
<td>e</td>
<td>exterior</td>
</tr>
</tbody>
</table>

$P$ passes from outside $Q$ to inside at $f$.

Case 1 ($e$ is exterior):

- If $e$ does not aim at $f$: advance $e$
- If $e$ aims at $f$: advance $e$
$P$ passes from outside $Q$ to inside at $f$.

Case 1 ($e$ is exterior):

- If $e$ does not aim at $f$: advance $e$
- If $e$ aims at $f$: advance $e$

$\Rightarrow$ Until $e$ crosses $f$ we advance $e$. 
$P$ passes from outside $Q$ to inside at $f$.

**Case 2 ($e$ is interior):**

- If $e$ aims at $f$:
  
  $\Rightarrow f$ is interior and $(f, e)$ is CCW
  
  $\Rightarrow f$ cannot aim at $e$
  
  $\Rightarrow$ Advance $e$

  $\Rightarrow$ Until $e$ is exterior, advance $e$. 
$P$ passes from outside $Q$ to inside at $f$.

Case 2 ($e$ is interior):

- If $e$ aims at $f$:
  - $f$ is interior and $(f, e)$ is CCW
  - $f$ cannot aim at $e$
  - Advance $e$

  $\Rightarrow$ Until $e$ is exterior, advance $e$.

  $\Rightarrow$ Back to case 1.
$P$ passes from outside $Q$ to inside at $f$.

Case 2 ($e$ is interior):

• If $e$ does not aim at $f$:

Claim:
In this case the edges meet at the next intersection.

Whichever edge gets to the next intersection first waits for the other.
Convex Polygon

<table>
<thead>
<tr>
<th></th>
<th>$\text{aim}(e, f)$</th>
<th>$\neg\text{aim}(e, f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{aim}(f, e)$</td>
<td>exterior</td>
<td>$f$</td>
</tr>
<tr>
<td>$\neg\text{aim}(f, e)$</td>
<td>$e$</td>
<td>exterior</td>
</tr>
</tbody>
</table>

$P$ passes from outside $Q$ to inside at $f$.

**Case 2 ($e$ is interior):**

- If $e$ does not aim at $f$:
  - If $f$ is interior:
    - $\Rightarrow f$ aims at $e$
    - $\Rightarrow$ Advance $f$
Convex Polygon

<table>
<thead>
<tr>
<th>aim(e, f)</th>
<th>¬aim(e, f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>aim(f, e)</td>
<td>exterior</td>
</tr>
<tr>
<td>¬aim(f, e)</td>
<td>e</td>
</tr>
</tbody>
</table>

$P$ passes from outside $Q$ to inside at $f$.

Case 2 ($e$ is interior):
- If $e$ does not aim at $f$:
  - If $f$ is interior:
    - $f$ aims at $e$
    - Advance $f$
  - Until $f$ is exterior, advance $f$.
- At that point, $e$ and $f$ aim away from each other
Convex Polygon

<table>
<thead>
<tr>
<th></th>
<th>aim(e, f)</th>
<th>¬aim(e, f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>aim(f, e)</td>
<td>exterior</td>
<td>f</td>
</tr>
<tr>
<td>¬aim(f, e)</td>
<td>e</td>
<td>exterior</td>
</tr>
</tbody>
</table>

$P$ passes from outside $Q$ to inside at $f$.

Case 2 ($e$ is interior):

- Until the next intersection:
  
  ⇒ $e$ only advances if $f$ is interior

Note:

If $e$ advances and $f$ is exterior then $e$ aims at $f$ and $f$ does not aim at $e$.

But this cannot be.
Convex Polygon

<table>
<thead>
<tr>
<th></th>
<th><code>aim(e, f)</code></th>
<th><code>¬aim(e, f)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>aim(f, e)</code></td>
<td>exterior</td>
<td><code>f</code></td>
</tr>
<tr>
<td><code>¬aim(f, e)</code></td>
<td><code>e</code></td>
<td>exterior</td>
</tr>
</tbody>
</table>

$P$ passes from outside $Q$ to inside at $f$.

Case 2 ($e$ is interior):

- Until the next intersection:
  - $e$ only advances if $f$ is interior
  - If $e$ advances to the intersection, $f$ must have been interior before.
  - $f$ is exterior after.
  - $e$ waits until $f$ arrives.
Convex Polygon

\[ P \text{ passes from outside } Q \text{ to inside at } f. \]

Case 2 (\( e \) is interior):

- Until the next intersection:
  - \( e \) only advances if \( f \) is interior
  - If \( f \) advances to the intersection, \( f \) must be interior.
  - \( f \) waits until \( e \) arrives.
Convex Polygon

<table>
<thead>
<tr>
<th></th>
<th>$\text{aim}(e,f)$</th>
<th>$\neg\text{aim}(e,f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{aim}(f,e)$</td>
<td>exterior</td>
<td>$f$</td>
</tr>
<tr>
<td>$\neg\text{aim}(f,e)$</td>
<td>$e$</td>
<td>exterior</td>
</tr>
</tbody>
</table>

$P$ passes from outside $Q$ to inside at $f$.

Case 2 ($e$ is interior to $f$):

- If $e$ does not aim at $f$:  
  $\Rightarrow e$ and $f$ advance to the next intersection point.
Sub-Claim 2:

Once a point of intersection has been found, the next intersection will be found (without skipping).
Proof (Sub-Claim 2):

W.l.o.g., assume that \( e \) is interior.

\[ \Rightarrow e \text{ does not aim at } f. \]

\[ \Rightarrow \text{As above, } e \text{ and } f \text{ advance to the next intersection.} \]
Proof (Sub-Claim 2):

W.l.o.g, assume that $e$ is interior to $f$.

⇒ $e$ does not aim at $f$.

⇒ As above, $e$ and $f$ advance to the next intersection.

Thus, we find an intersection within the first $|P| + |Q|$ iterations and find the rest within the next $|P| + |Q|$ iterations.