Visibility Graphs and Cell Decompositions

CIS 390
Kostas Daniilidis
With material from Chapter 5 - Roadmaps
Principles of Robot Motion: Theory, Algorithms, and Implementation
by Howie Choset & al.
The MIT Press Â© 2005
For polygonal obstacles in 2D

- Assume robot is a point
- Any shortest path from start to goal among a set of disjoint polygonal obstacles is a polygonal path whose inner vertices are convex vertices of the obstacles (imagine a tight rope from start to end)
Visibility graph

- **Vertices**: all vertices of the polygonal obstacles plus the start and the goal
- **Edges**: all line segments between vertices that do not intersect obstacles
- It is not a planar graph: edges are crossing (not at vertices)
Visibility graph construction

• Brute force $O(n^3)$ algorithm visits
  – all $n$ vertices
  – applies a rotational plane sweep that connects with all other $n$ vertices
  – and determines whether each segment intersects any of the $O(n)$ edges

• We can do better by sorting the vertices.
**Sweep Line Algorithm**

- **Input:** A set of vertices \( \{v_i\} \) (whose edges do not intersect) and a vertex \( v \)
- **Output:** A subset of vertices from \( \{v_i\} \) that are within line of sight of \( v \)
- 1: For each vertex \( v_i \), calculate \( \alpha_i \), the angle from the horizontal axis to the line segment \( vv_i \).
- 2: Create the vertex list \( E \), containing the \( \alpha_i \) ’s sorted in increasing order.
- 3: Create the active list \( S \), containing the sorted list of edges that intersect the horizontal half-line emanating from \( v \).
- 4: for all \( \alpha_i \) do
  - 5: if \( v_i \) is visible to \( v \) then
    - 6: Add the edge \( (v, v_i) \) to the visibility graph.
  - 7: end if
  - 8: if \( v_i \) is the beginning of an edge, \( E \), not in then
  - 9: Insert the \( E \) into \( S \)
  - 10: end if
  - 11: if \( v_i \) is the end of an edge in then
  - 12: Delete the edge from \( S \).
  - 13: end if
  - 14: end for

If \( S \) is in a balanced binary search tree it is \( n \log n \), and including step 1: \( O(n^2 \log n) \)
Sort the angles $\alpha_i$ between edges $(v, v_i)$ (sweep line) and the horizontal

$$\mathcal{E} = \{\alpha_3, \alpha_7, \alpha_4, \alpha_8, \alpha_1, \alpha_5, \alpha_2, \alpha_6, \},$$
$$S = \{E_4, E_2, E_8, E_6\}.$$  

Sorted angles counterclockwise
Sorted edges intersecting sweep line from left to right
Main idea for graph construction

- Maintain set of edges $S$ that intersect sweep line **sorted** in increasing distance from the “center” vertex $v$.
- Visibility test: $v_i$ is visible from $v$ if segment $(v,v_i)$ does not intersect the closest edge in $S$ and if the sweep line is not inside the obstacle (between two edges incident in $v$).
<table>
<thead>
<tr>
<th>Vertex</th>
<th>New $S$</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization</td>
<td>${E_4, E_2, E_8, E_6}$</td>
<td>Sort edges intersecting horizontal half-line</td>
</tr>
<tr>
<td>$\alpha_3$</td>
<td>${E_4, E_3, E_8, E_6}$</td>
<td>Delete $E_2$ from $S$. Add $E_3$ to $S$.</td>
</tr>
<tr>
<td>$\alpha_7$</td>
<td>${E_4, E_3, E_8, E_7}$</td>
<td>Delete $E_6$ from $S$. Add $E_7$ to $S$.</td>
</tr>
<tr>
<td>$\alpha_4$</td>
<td>${E_8, E_7}$</td>
<td>Delete $E_3$ from $S$. Delete $E_4$ from $S$. ADD $(v, v_4)$to visibility graph</td>
</tr>
<tr>
<td>$\alpha_8$</td>
<td>$\emptyset$</td>
<td>Delete $E_7$ from $S$. Delete $E_8$ from $S$. ADD $(v, v_8)$to visibility graph</td>
</tr>
<tr>
<td>$\alpha_1$</td>
<td>${E_1, E_4}$</td>
<td>Add $E_4$ to $S$. Add $E_1$ to $S$. ADD $(v, v_1)$to visibility graph</td>
</tr>
<tr>
<td>$\alpha_5$</td>
<td>${E_4, E_1, E_8, E_5}$</td>
<td>Add $E_8$ to $S$. Add $E_5$ to $S$.</td>
</tr>
<tr>
<td>$\alpha_2$</td>
<td>${E_4, E_2, E_8, E_5}$</td>
<td>Delete $E_1$ from $S$. Add $E_2$ to $S$.</td>
</tr>
<tr>
<td>$\alpha_6$</td>
<td>${E_4, E_2, E_8, E_6}$</td>
<td>Delete $E_5$ from $S$. Add $E_6$ to $S$.</td>
</tr>
<tr>
<td>Termination</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Trapezoidal Decomposition is a Cell Decomposition (as opposed to vertex)
Trapezoidal Decomposition

- Defined for any set of line segments (not necessarily obstacle edges)
- Consists of trapezoids and triangles
- For each vertex draw an upper vertical and a lower vertical extension
- Main challenge is to find intersections of vertical extensions with line segments.
Find intersections faster with a sweep line

- Sweep a vertical line stopping at vertices from left to right: these stops are called events
- Maintain list $L$ of edges adjacent to vertex events which sorted based on $y$-coordinate of intersection (we can do insertion and deletion in $\log n$)

\begin{align*}
\text{delete } e_{\text{lower}} \text{ and } e_{\text{upper}} \text{ from the list} & \quad \text{delete } e_{\text{lower}} \text{ from the list and insert } e_{\text{upper}} \\
\text{insert } e_{\text{lower}} \text{ and } e_{\text{upper}} \text{ into the list} & \quad \text{delete } e_{\text{upper}} \text{ from the list and insert } e_{\text{lower}}
\end{align*}
First three events

The graph of trapezoids/triangles can be computed from the list of edge events by storing the intersections.
Figure 6.6: There are 14 events in this example.

<table>
<thead>
<tr>
<th>Event</th>
<th>Sorted Edges in L</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{a, b}</td>
</tr>
<tr>
<td>1</td>
<td>{d, b}</td>
</tr>
<tr>
<td>2</td>
<td>{d, f, e, b}</td>
</tr>
<tr>
<td>3</td>
<td>{d, f, i, b}</td>
</tr>
<tr>
<td>4</td>
<td>{d, f, g, h, i, b}</td>
</tr>
<tr>
<td>5</td>
<td>{d, f, g, j, n, h, i, b}</td>
</tr>
<tr>
<td>6</td>
<td>{d, f, g, j, n, b}</td>
</tr>
<tr>
<td>7</td>
<td>{d, j, n, b}</td>
</tr>
<tr>
<td>8</td>
<td>{d, j, n, m, l, b}</td>
</tr>
<tr>
<td>9</td>
<td>{d, j, l, b}</td>
</tr>
<tr>
<td>10</td>
<td>{d, k, l, b}</td>
</tr>
<tr>
<td>11</td>
<td>{d, b}</td>
</tr>
<tr>
<td>12</td>
<td>{d, c}</td>
</tr>
<tr>
<td>13</td>
<td>{}</td>
</tr>
</tbody>
</table>
Running time

- Sorting vertices according to x-coordinate is $O(n \log n)$
- Inserting and deleting edges according to y-coordinate is $O(\log n)$
- This happens at each event so total time is $O(n \log n)$