Computing the Overlay of Two Subdivisions

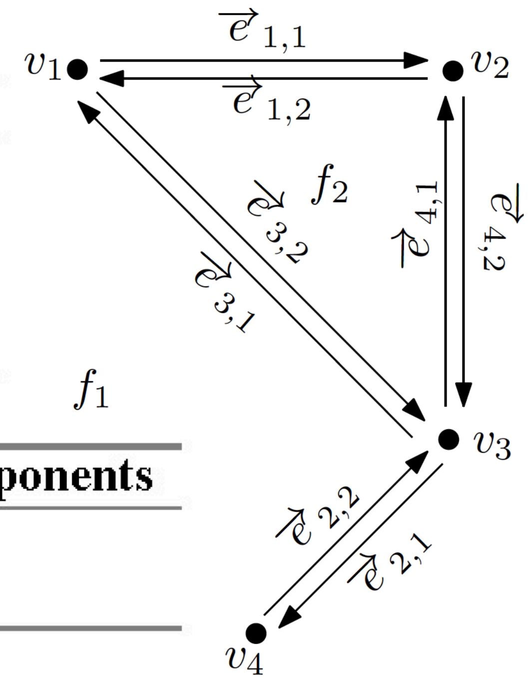
Michael Goodrich

Computational Geometry

DCEL contains:

- a record for each vertex,
  1. $Coordinates(v)$: the coordinates of v,
  2. $IncidentEdge(v)$: a pointer to an arbitrary half-edge that has v as its origin.
- a record for each face,
  1. $OuterComponent(f)$: to some half-edge on its outer boundary (nil if unbounded),
  2. $InnerComponents(f)$: a pointer to some half-edge on the boundary of the hole, for each hole.
- a record for each half-edge $\overrightarrow{e}$,
  1. $Origin(\overrightarrow{e})$: a pointer to its origin,
  2. $Twin(\overrightarrow{e})$ a pointer to its twin half-edge,
  3. $IncidentFace(\overrightarrow{e})$: a pointer to the face that it bounds.
  4. $Next(\overrightarrow{e})$ and $Prev(\overrightarrow{e})$: a pointer to the next and previous edge on the boundary of $IncidentFace(\overrightarrow{e})$.

Example DCEL



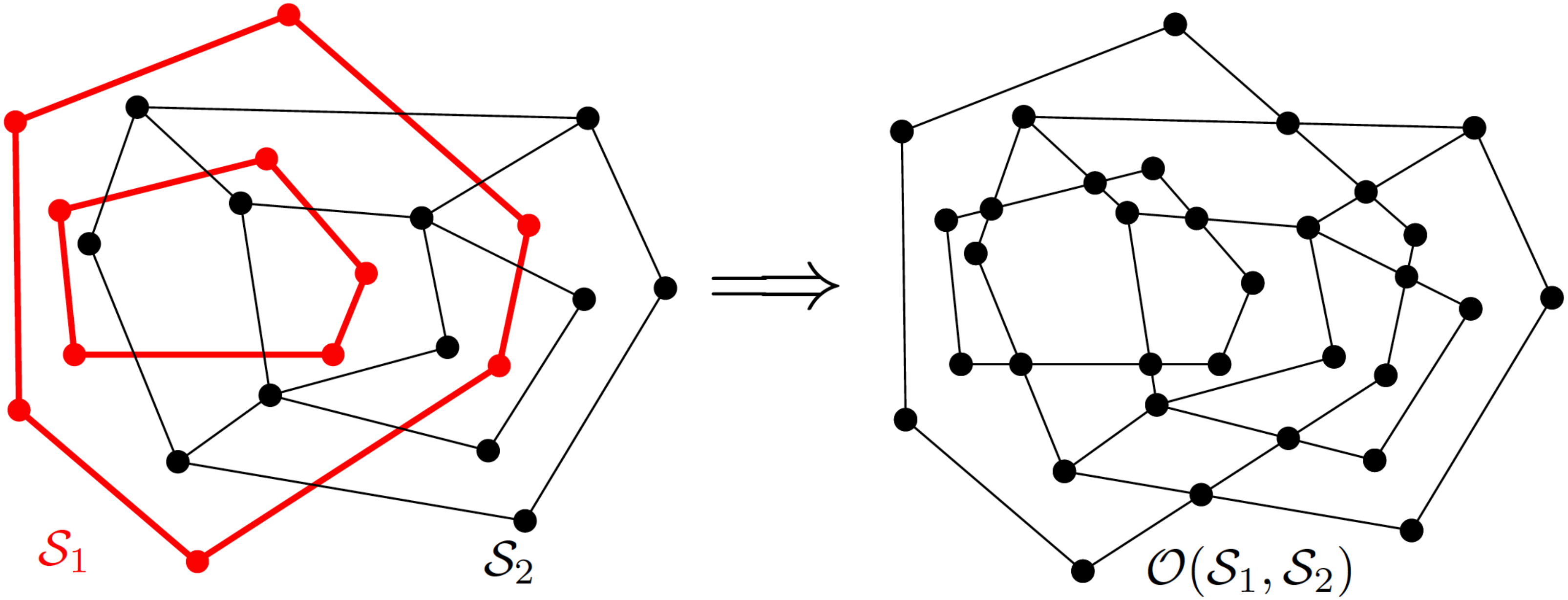
Vertex	**Coordinates**	**IncidentEdge**
v_1	$(0,4)$	$\vec{e}_{1,1}$
v_2	$(2,4)$	$\vec{e}_{4,2}$
v_3	$(2,2)$	$\vec{e}_{2,1}$
v_4	$(1,1)$	$\vec{e}_{2,2}$

Face	**OuterComponent**	**InnerComponents**
f_1	**nil**	$\vec{e}_{1,1}$
f_2	$\vec{e}_{4,1}$	**nil**

Half-edge	**Origin**	**Twin**	**IncidentFace**	**Next**	**Prev**
$\vec{e}_{1,1}$	v_1	$\vec{e}_{1,2}$	f_1	$\vec{e}_{4,2}$	$\vec{e}_{3,1}$
$\vec{e}_{1,2}$	v_2	$\vec{e}_{1,1}$	f_2	$\vec{e}_{3,2}$	$\vec{e}_{4,1}$
$\vec{e}_{2,1}$	v_3	$\vec{e}_{2,2}$	f_1	$\vec{e}_{2,2}$	$\vec{e}_{4,2}$
$\vec{e}_{2,2}$	v_4	$\vec{e}_{2,1}$	f_1	$\vec{e}_{3,1}$	$\vec{e}_{2,1}$
$\vec{e}_{3,1}$	v_3	$\vec{e}_{3,2}$	f_1	$\vec{e}_{1,1}$	$\vec{e}_{2,2}$
$\vec{e}_{3,2}$	v_1	$\vec{e}_{3,1}$	f_2	$\vec{e}_{4,1}$	$\vec{e}_{1,2}$
$\vec{e}_{4,1}$	v_3	$\vec{e}_{4,2}$	f_2	$\vec{e}_{1,2}$	$\vec{e}_{3,2}$
$\vec{e}_{4,2}$	v_2	$\vec{e}_{4,1}$	f_1	$\vec{e}_{2,1}$	$\vec{e}_{1,1}$

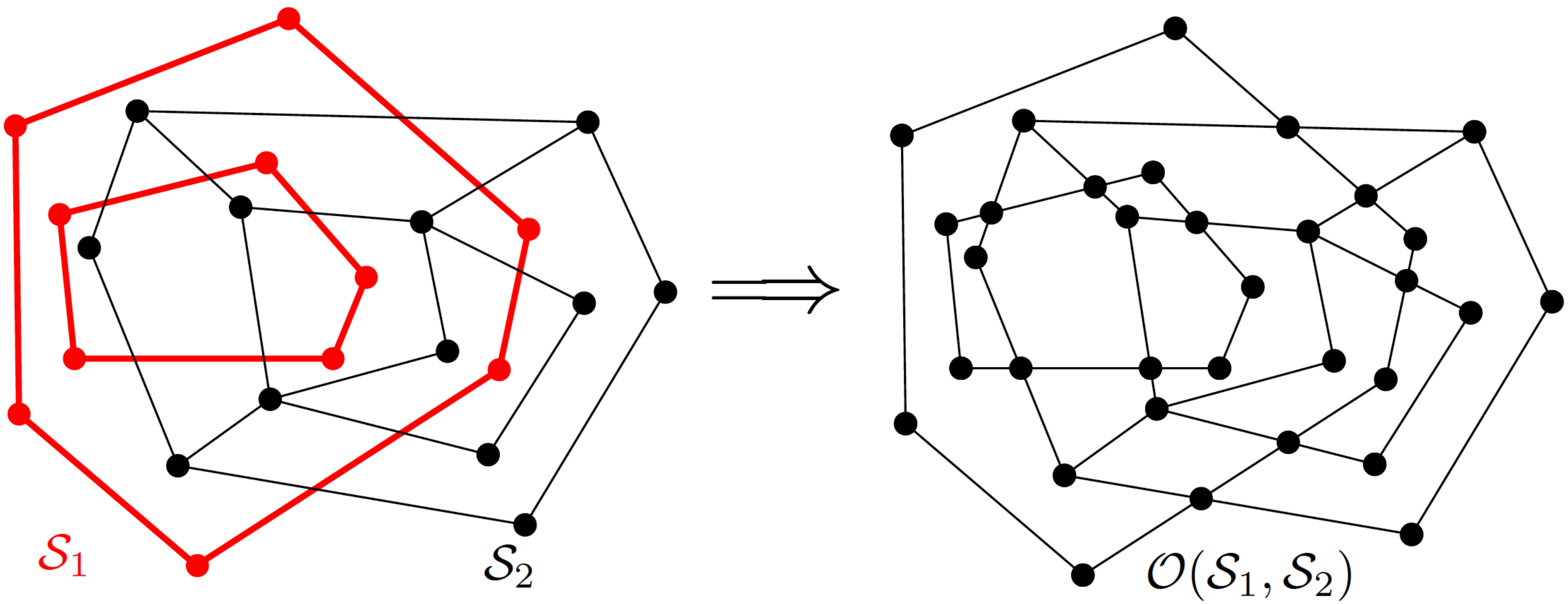
Computing the Overlay

- Input: DCEL for S_1 and DCEL for S_2
- Output: DCEL for the overlay of S_1 and S_2



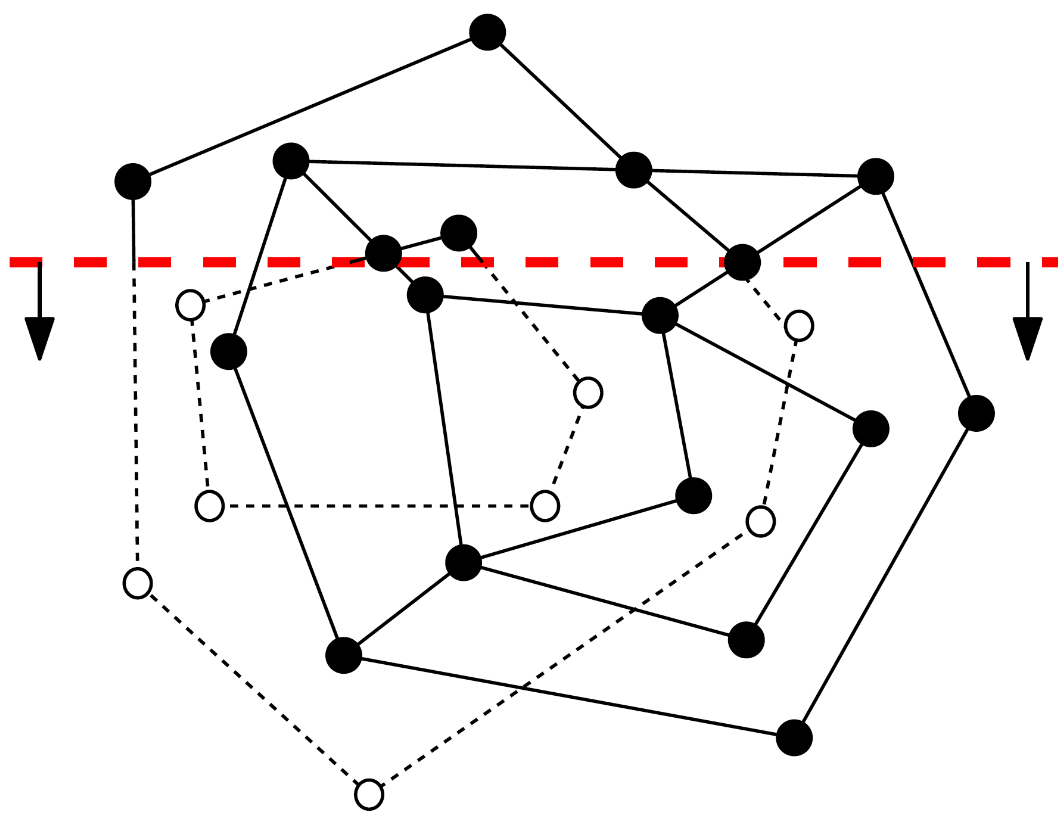
Computing the Overlay

- Initialization: copy the DCEL for S_1 and S_2
- These are then "merged" into one



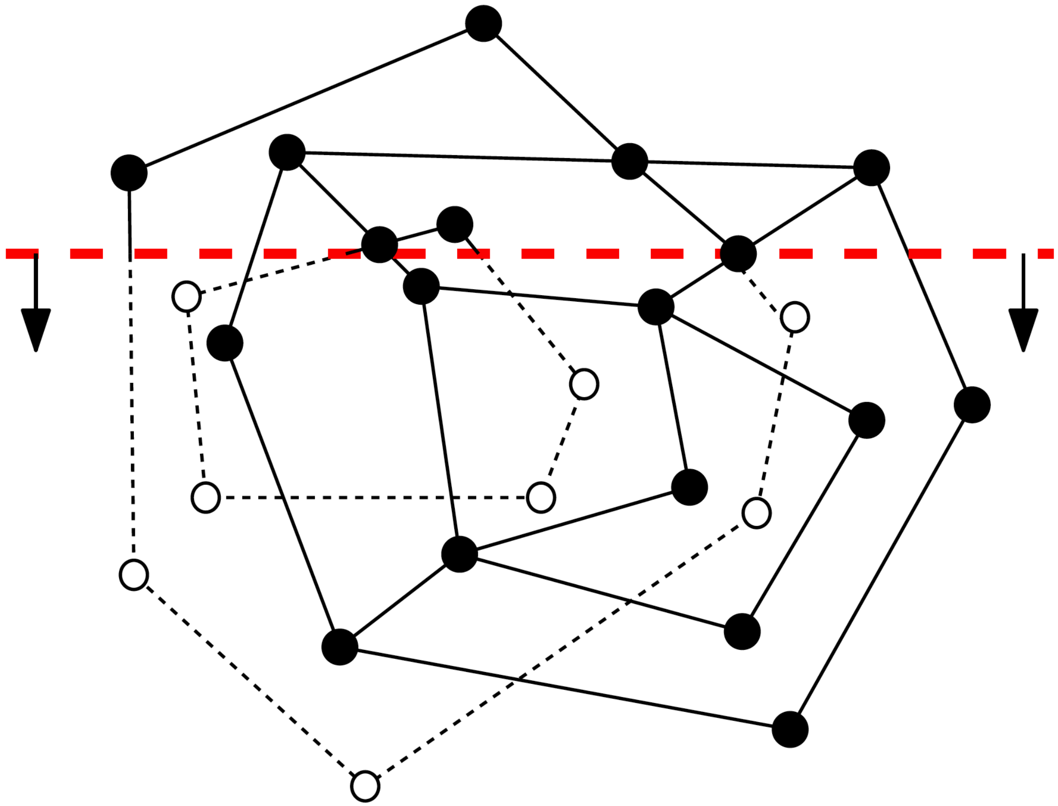
Use Our Plane-Sweep Algorithm

- Plane-sweep as in our line segment intersection algorithm



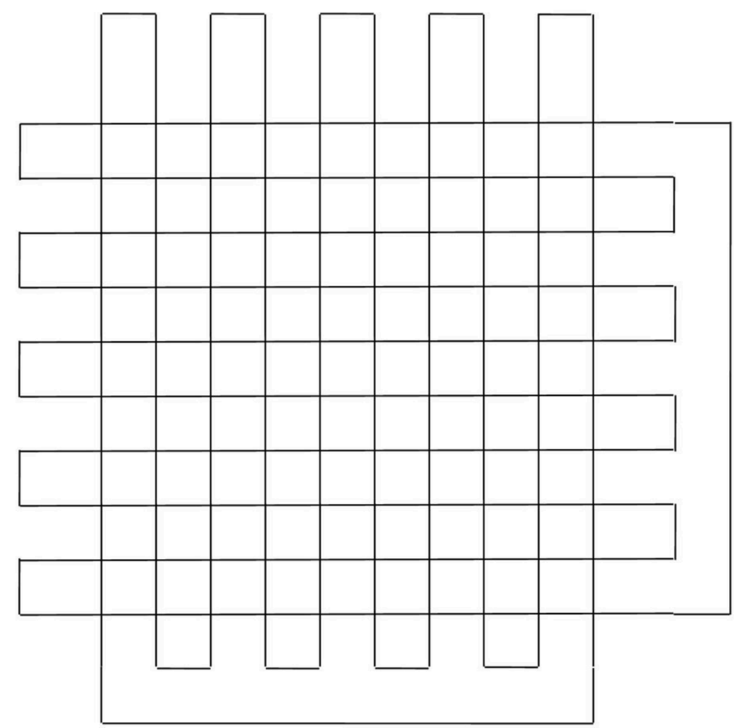
A New Step

- For each intersection event, add a new vertex to the merged DCEL



Time Complexity

- The running time is $O((n+k)\log n)$, where n is the total size of the two input subdivisions and k is the number of intersections
- And k can be $O(n^2)$:



Boolean Operations

- Essentially the same algorithm can be used for geometric Boolean operations