Surface Relevant Cells

Daniel Lederer,
Institute of Theoretical Computer Science

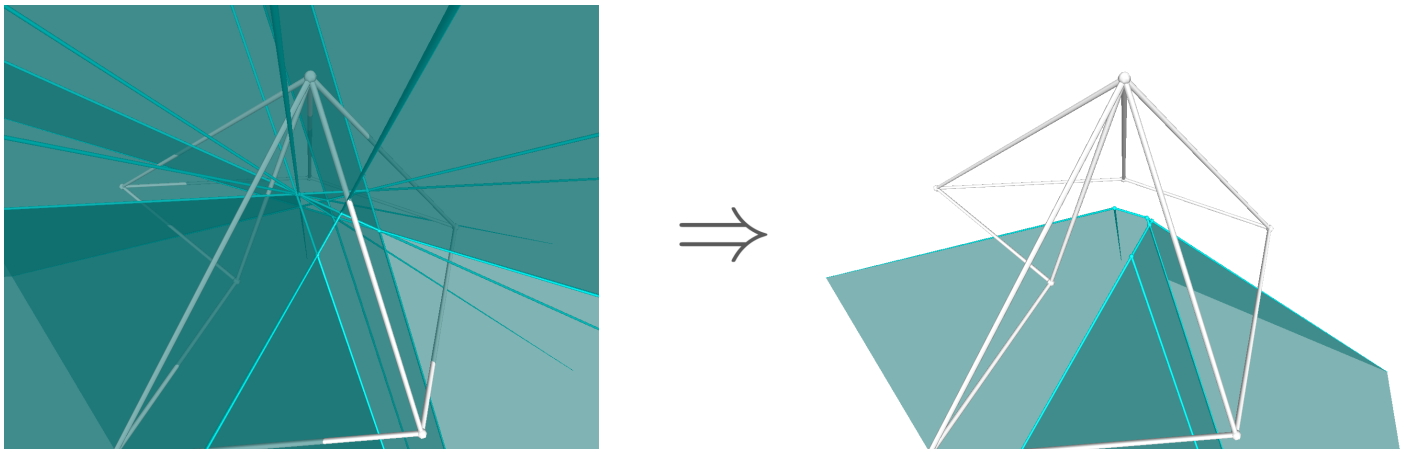
22. Mai 2022

What we have so far

- We identified a vertex of the polyhedron where an event takes place
- The event vertex v_{event} should be resolved (vertex gets split, new edges are created/merged, etc.)
- The offset supporting planes of adjacent facets were calculated
- The simple arrangement of these planes was computed

Problem definition

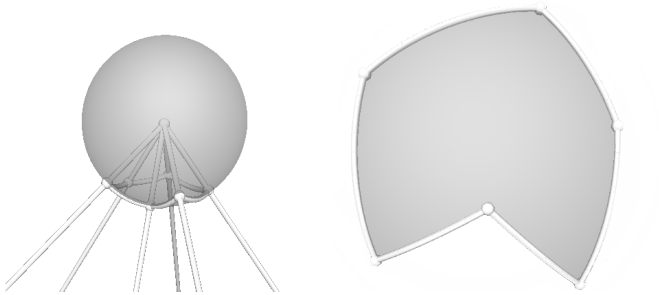
- From all arrangement cells we have to identify those that are relevant for the offset surface



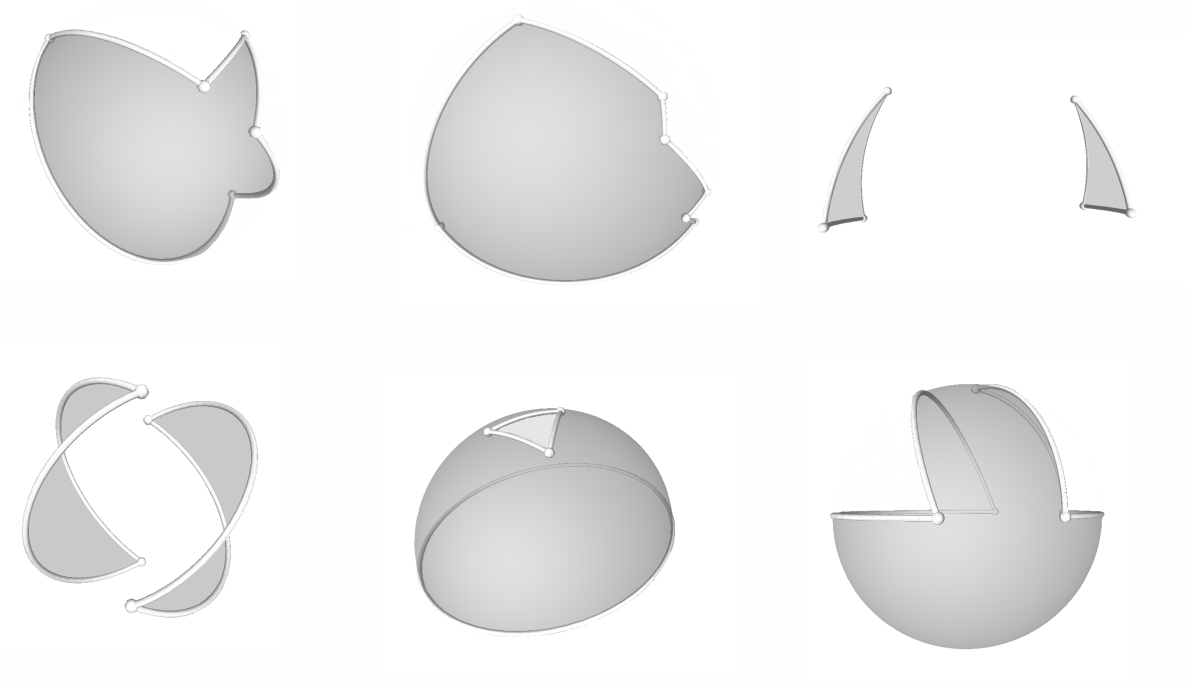
An arrangement consisting of 42 cells, where only 3 are relevant

Solution (1)

- A cell is relevant if its boundary is fully visible from v_{event} (inner side of polyhedron)
- Determine visibility by using a spherical polygon S
- S is created by intersecting our polyhedron with a sphere, centered at v_{event}
- A cell is visible iff its radial projection is entirely contained in S



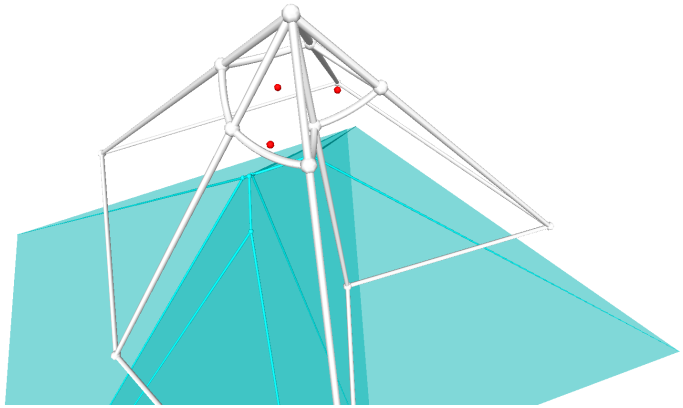
Examples of spherical polygons



A spherical polygon can be arbitrarily complex: disconnected, containing holes, etc.

Solution (2)

- For each cell *C*, consider the boundary: Create sum of directions of unbounded edges of *C*
- Project the new direction as a point onto the sphere
- **Point is inside spherical polygon S $\Leftrightarrow$ C lies entirely inside S (and thus is surface relevant)**



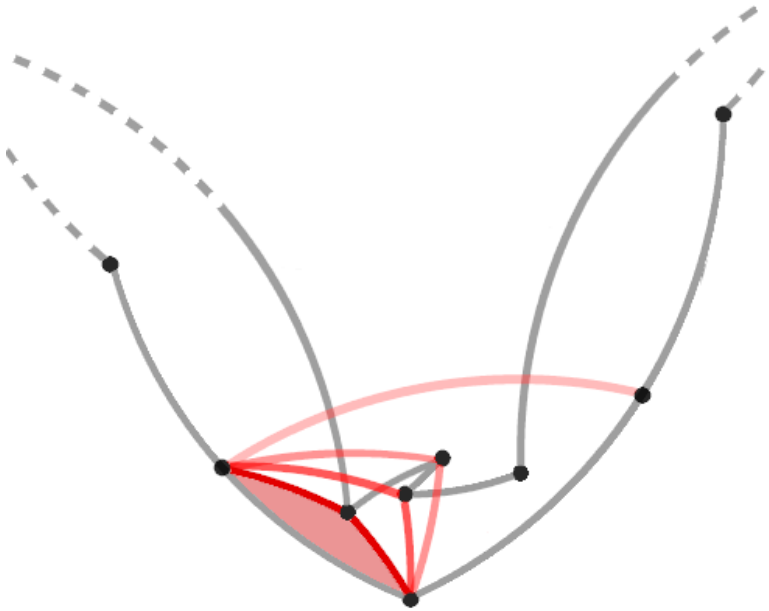
Spherical polygon

- As we have seen, the spherical polygon can be of any shape
- Therefore, there must be a clear distinction between what is inside/outside
- Regardless of the cells, we specify some points that are certainly inside
  - *We will see at the end that computing many points (instead of a single one) will increase the numerical stability of our solution*

Computing enclosed points

- Within the spherical polygon consider a small enclosed triangular region
- If there is any convex polygon vertex, consider this vertex and the two midpoints of the adjacent arcs as triangular region
  - We have to make sure that the region is fully enclosed $\Rightarrow$ maybe make region smaller

Example of enclosed region (convex case)

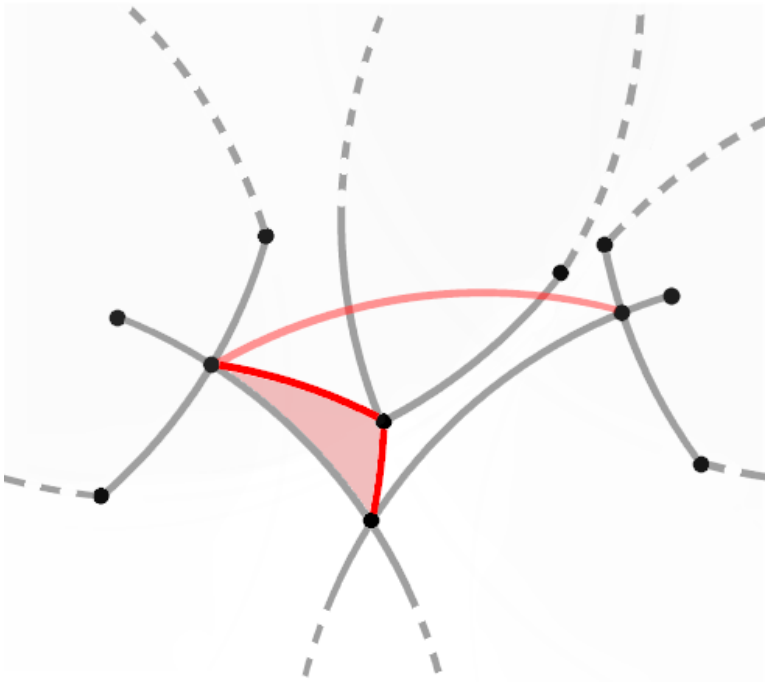


The region is gradually reduced until it is completely enclosed from the spherical polygon

Computing enclosed points

- If all vertices of the spherical polygon are reflex, consider any vertex
- From this vertex create two new arcs opposite to the original adjacent arcs, but with half length
- Consider the three vertices as triangular region
  - We again have to make sure that the region is fully enclosed $\Rightarrow$ maybe make region smaller
- Finally, compute “random” points in this region $\Rightarrow$ they are certainly enclosed as well

Example of enclosed region (reflex case)



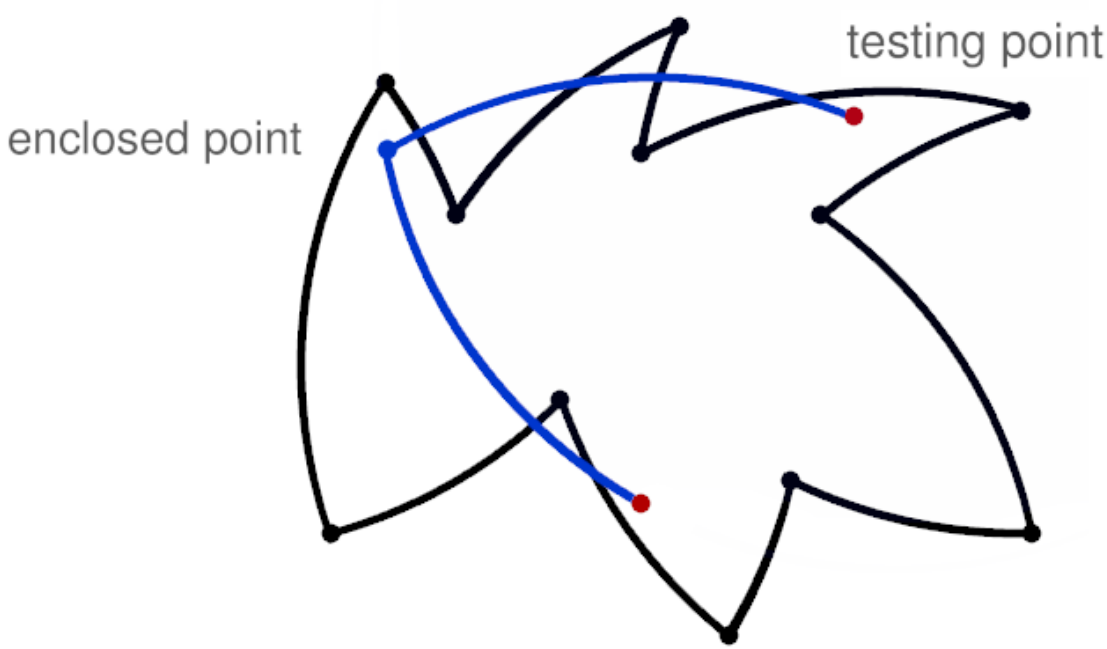
First, the opposite arcs are shortened and then the region is further reduced

Inclusion test

- Considering the enclosed points, inside/outside is sufficiently defined
- Now we are able to decide if a new point is also contained by the spherical polygon:

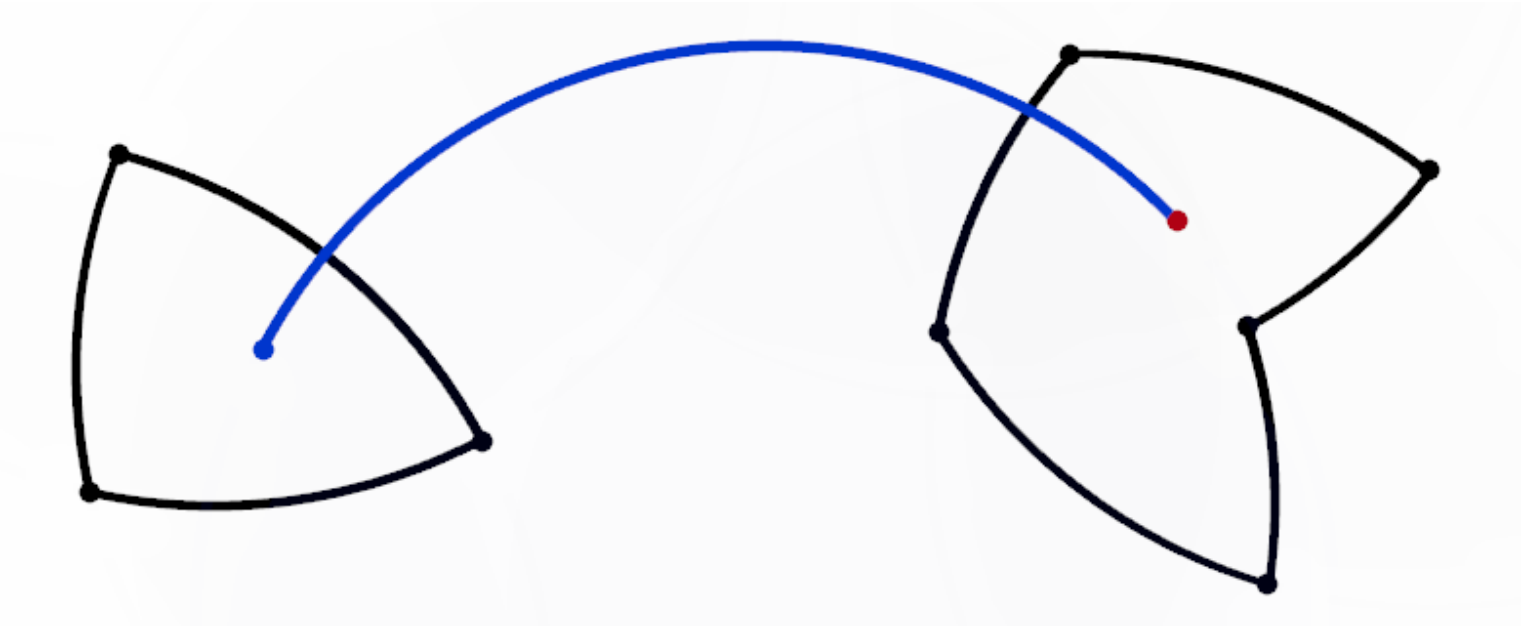
1. Create an (shortest-distance) arc from each enclosed point to the testing point
2. For each of those arcs count the intersections with other polygon's arcs
3. If number of intersections is even, testing point is enclosed as well

Example of inclusion test



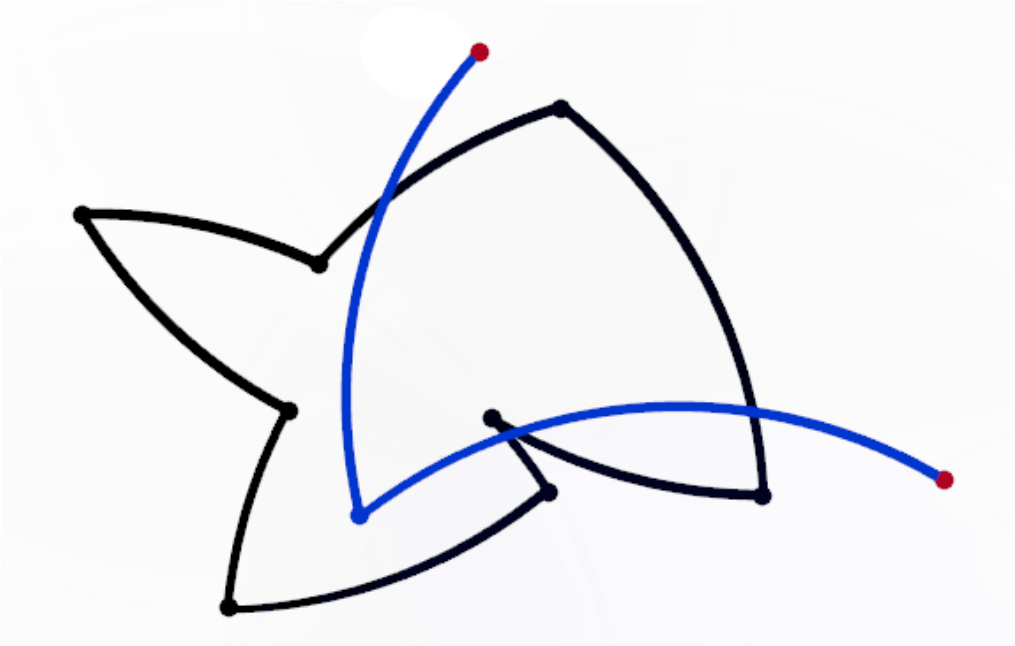
There are 4 and 2 intersections (even), so both testing points lie inside the spherical polygon

Example of inclusion test



The strategy also works wonderfully for disconnected spherical polygons and for those with holes

Example of inclusion test

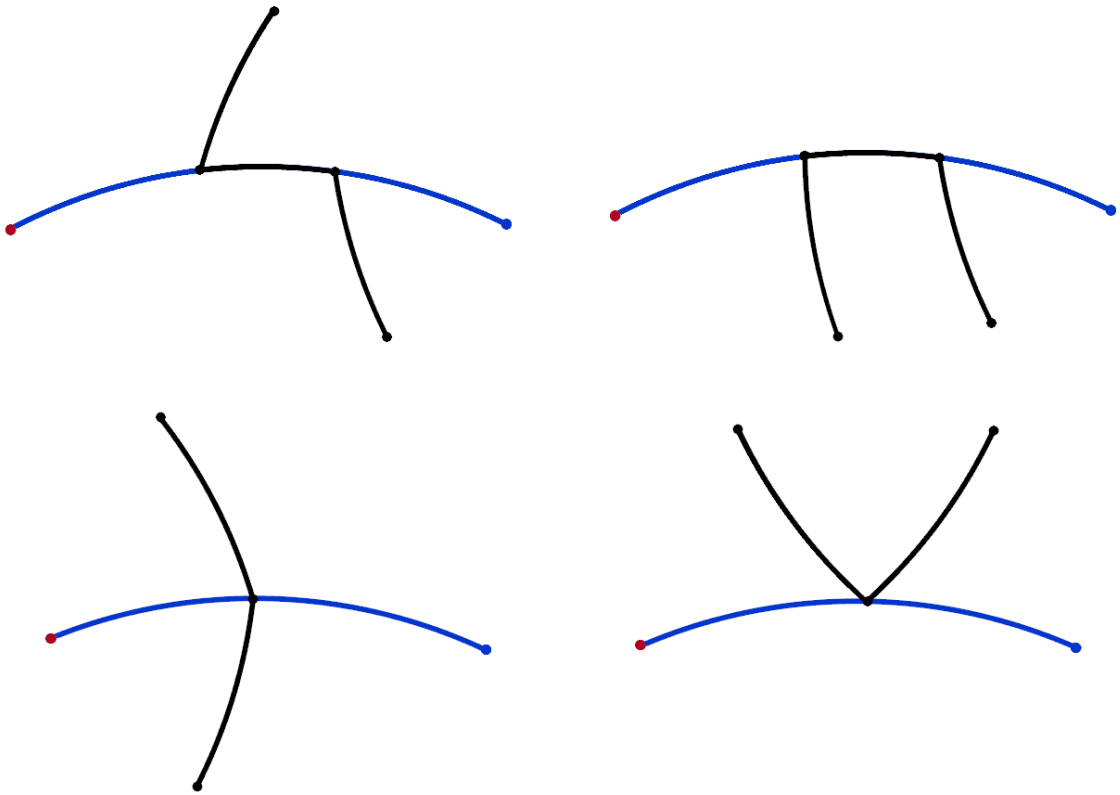


There are 3 and 1 intersections (odd), so both testing points do not lie inside the spherical polygon

What can go wrong?

- We can be confident that the enclosed points lie strictly inside the spherical polygon
- We can be confident that two arcs intersect in max. 1 point (created arc lies on one hemisphere)
- Testing point may be close to the polygon border
- It might happen that the created arc goes right through a vertex
  - How many intersections should then be counted?
  - One for each arc or one altogether?

Problematic intersection scenarios



Increasing numerical robustness

- We cannot completely avoid the problematic intersection scenarios
- But we can try to overcome the negative impacts using many enclosed points instead of a single one
- For each different (!) enclosed point, count the intersections and decide independently
- In the end we trust the majority
- In practical terms, this works fine, since such problematic scenarios are rare

Complexity

- Let d be the facet-degree of v_{event}, then the spherical polygon has complexity $\Theta(d)$
- Let n be the number of planes in the arrangement with $n \leq d \Rightarrow \Theta(n^3)$ cells
- Initially, we say, only unbounded cells can be relevant, nevertheless there are $\mathcal{O}(n^3)$ of them $\Rightarrow \mathcal{O}(n^3)$ projected points need to be tested
- For each of the k enclosed points we have to decide for each projected point whether it is inside the spherical polygon $\Rightarrow \mathcal{O}(k \cdot d \cdot n^3)$ time in total

Summary

- Using the spherical polygon we can decide whether an unbounded cell is fully visible, and consequently is relevant for the offset surface
- For each cell, project a point onto the sphere and make an inclusion test
- We need enclosed points to define the inside
- Several different enclosed points increase the numerical robustness of the algorithm
- All surface relevant cells can be found in $\mathcal{O}(k \cdot d \cdot n^3)$ time