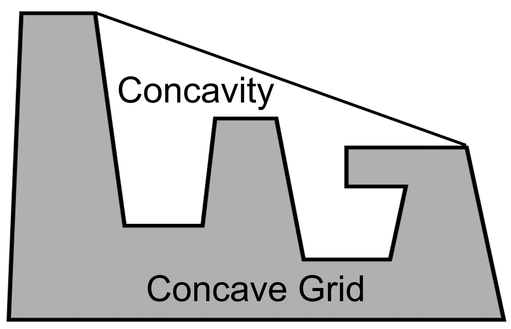
9th International Fall Workshop
VISION, MODELING, AND VISUALIZATION 2004
November 16 - 18, 2004
Stanford (California), USA

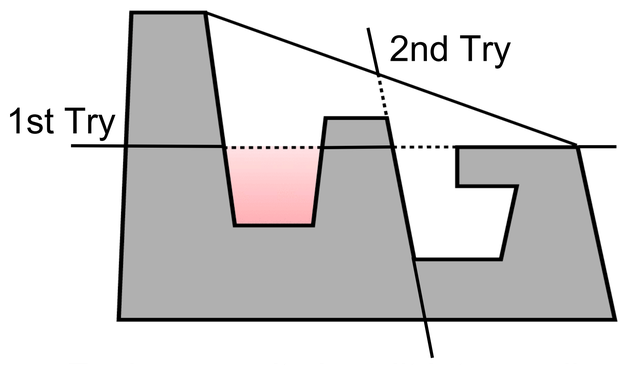
Convexification of Unstructured Grids

Stefan Roettger, Stefan Guthe

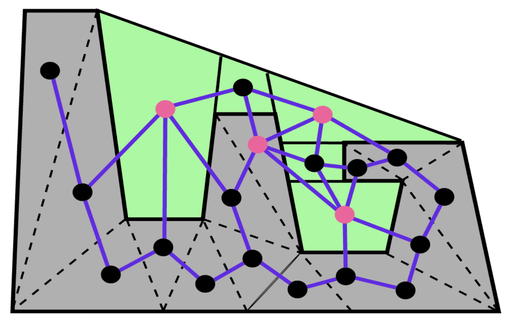
- Unstructured tetrahedral grids need to be depth-sorted before rendering with the PT Algorithm
- Sorting time is linear for convex grids using the MPVO algorithm
- We use convexification algorithm to transform concave into convex grids



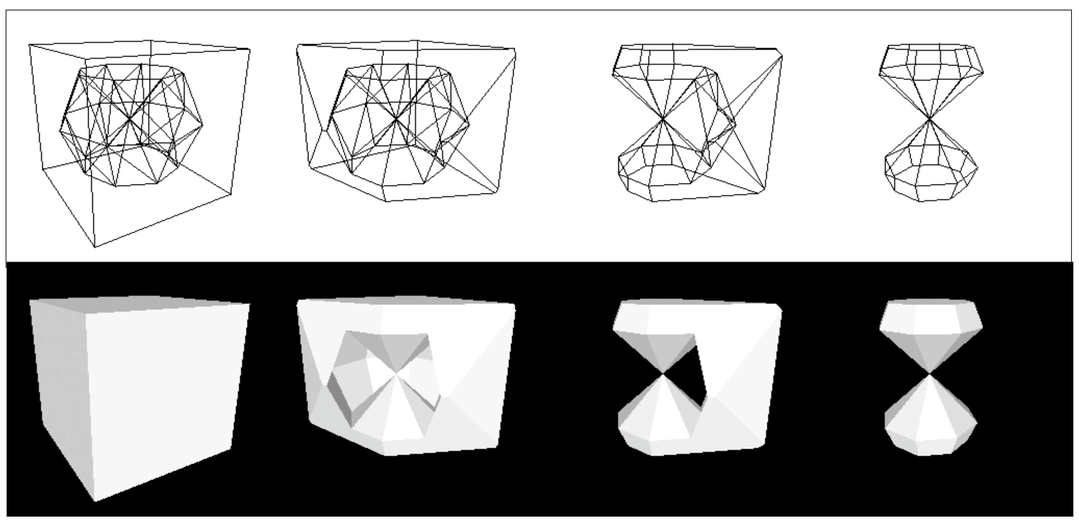
The convexfication process starts with the concavities which need to be split into convex polyhedra. Then these polyhedra are added to the original grid to yield a convex mesh. Splitting is performed along the “concave” faces of a concavity.



Each concavity is split into smaller auxiliar polyhedra until all remaining polyhedra are convex. Above, the first try of a split is bad because it produces the redundant red cell. The second cut is a better choice, since it has fewer intersections with the boundary.

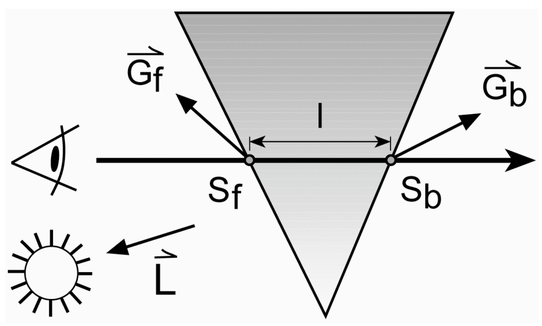


From the resulting green convex polyhedra a DAG is constructed that can be sorted with the MPVO in linear time. Note that the DAG has multiple dependencies per face for the red nodes.



A 3D example is depicted on the left. It shows a rotational concave solid enclosed in a cube. The cube is split subsequently until two convex polyhedra remain which augment the rotational solid forming a convex grid. Note that the auxiliar cells outside the convex hull of the solid are discarded. In 3D the split operation is much more complex than in 2D. However, we managed to break it down into a series of simple operations on triangle meshes which result in a robust convexification algorithm.

- Once the convexified mesh is sorted, rendering is straightforward
- Drop all auxiliary polyhedra and project the remainig original tetrahedra with the PT algorithm
- Per-pixel exact evaluation of the ray integral is performed using pre-integrated lighting



The ray integral of each ray segment inside a tetrahedron depends only on the scalar values (S) and gradients (G) at the entry and exit points and the ray segment length (l). Therefore, the ray intregral can be pre-integrated for each ray segment and stored in a lookup table (dependent 3D texture map).