**Contribution**

- **Flip-Flop**: a novel flip algorithm to transform any star-shaped polyhedron to its convex hull.
- **ffHull**: an algorithm to construct 3D convex hull using flips that works well on both CPUs and GPUs.
- **ffRT**: the first known algorithm to construct the 2D regular triangulation on the GPU.

**Flip-Flop**: a novel flip algorithm

- **V-criterion**: increases the volume of the polyhedron by flipping the reflex edges, used in the traditional flip algorithm.
- **D-criterion**: decreases the degree of the non-extreme points by flipping the edges incident to them, a new criterion to remove non-extreme points.

Finding non-extreme points:

- A reflex edge that is unflippable implies a non-extreme point.

Flip-Flop combines V- and D-criterion to compute convex hull.

**ffHull**: 3D convex hull algorithm

- Find 4 extreme points for the initial tetrahedron.
- Compute the center as the kernel point $s$.
- For each $p$, associate it with a $t$ such that $p \in C_s(t)$.
- For each $t$, find $p$ that is furthest to $t$ and $p \in C_s(t)$.
- For each new $t'$, find $p$ that is furthest to $t'$ and $p \in C_s(t')$.

Implementation Tips:

- Maintain the orientation of each triangle so that the kernel point $s$ is always beneath it.
- Use orientation determinants when finding the furthest point.
- In GPU implementation, separate the kernels for fast and exact predicates.

**ffRT**: 2D regular triangulation algorithm

- A set of weighted points
- Lift to 3D space by $z = x^2 + y^2 - w$
- Compute the lower hull using ffHull
- RT

**Experiments**

- Convex Hull
- Regular Triangulation

Notes:

- Fix the orientation of each triangle so that the kernel point $s$ is always beneath it.
- Use orientation determinants when finding the furthest point.
- In GPU implementation, separate the kernels for fast and exact predicates.