Interactive and Robust Mesh Booleans

Gianmarco Cherchi (University of Cagliari, Italy)
Fabio Pellacini (Sapienza University Rome, Italy)
Marco Attene (CNR-IMATI Genoa, Italy)
Marco Livesu (CNR-IMATI, Genoa, Italy)
Mesh Booleans

- conceptually simple
- complex to implement correctly

union
intersection
subtraction

A ∪ B
A \ B

A
B

sa2022.siggraph.org
## Mesh Booleans algorithms

<table>
<thead>
<tr>
<th>floating points VS exact arithmetic</th>
<th>volume-based vs surface-based</th>
<th>exact results vs approx. results</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>easier I/O labeling</td>
<td>exact result</td>
</tr>
<tr>
<td>slow</td>
<td>more convoluted</td>
<td>needs repairing</td>
</tr>
<tr>
<td>possible errors</td>
<td>less efficient</td>
<td>more convoluted</td>
</tr>
<tr>
<td>exact</td>
<td>more performant</td>
<td>easier implement.</td>
</tr>
<tr>
<td>geometric predicates</td>
<td>snap rounding</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>snap rounding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>snap rounding</td>
</tr>
</tbody>
</table>

- **Floating Points vs Exact Arithmetic**
  - Fast
  - Slow
  - Possible errors
  - Exact
  - Geometric predicates
  - Snap rounding

- **Volume-Based vs Surface-Based**
  - Easier I/O labeling
  - More convoluted
  - Less efficient
  - More performant
  - Snap rounding

- **Exact Results vs Approximate Results**
  - Exact result
  - Needs repairing
  - More convoluted
  - Easier implement.
Robust and interactive Booleans with EMBER

EMBER: Exact Mesh Booleans via Efficient & Robust Local Arrangements

Trettner P., Nehring-Wirxel J., Kobbelt L.

ACM TOG 2022

fastest Boolean pipeline

EMBER: Exact Mesh Booleans via Efficient & Robust Local Arrangements

Trettner P., Nehring-Wirxel J., Kobbelt L.

ACM TOG 2022

sa2022.siggraph.org
Robust and interactive Booleans with EMBER

EMBER: Exact Mesh Booleans via Efficient & Robust Local Arrangements

PHILIP TRETTNER, Shaped Coder GmbH, Germany
JULIUS NEHRING-WIRXEL, Visual Computing Institute, RWTH Aachen University, Germany
LOD KOBELT, Visual Computing Institute, RWTH Aachen University, Germany

no compatibility with existing geometry processing tasks

EMBER: Exact Mesh Booleans via Efficient & Robust Local Arrangements

Trettner P., Nehring-Wirxel J., Kobbelt L.

ACM TOG 2022

example: QuadMixer
[Nuvoli et al. 2019]
Robust and interactive Booleans with EMBER

EMBER: Exact Mesh Booleans via Efficient & Robust Local Arrangements

**Trettner P., Nehring-Wirxel J., Kobbelt L.**

ACM TOG 2022

unnecessary stitching in the result mesh
Robust and interactive Booleans with EMBER

EMBER: Exact Mesh Booleans via Efficient & Robust Local Arrangements
Trettner P., Nehring-Wirxel J., Kobbelt L.
ACM TOG 2022

Interactive and Robust Mesh Booleans
Cherchi G., Pellacini F., Attene M., Livesu M.
ACM TOG 2022

- speed
- possible slowdown

- compatibility with existing geometry processing tasks
- compatibility with existing geometry processing tasks

different goals

sa2022.siggraph.org
The general pipeline

1. **Input meshes**
   - $A$ and $B$

2. **Resolve mesh intersections and create conforming patches**
   - $A_1$ and $B_1$
   - $A_2$ and $B_2$

3. **Inside/Outside patch labelling**
   - $A_2$ is inside $B$
   - $B_1$ is inside $A$

4. **Filter and merge patches to form the output mesh**
   - $A \cup B = \{A_1, B_2\}$
   - $A \cap B = \{A_2, B_1\}$
   - $A \setminus B = \{A_1, B_1\}$
   - $B \setminus A = \{A_2, B_2\}$
Our main contributions

- Robustness of exact floating point methods
- Fast inside/outside triangle classification

Result:
- One order of magnitude faster than state of art
- Compatible with existing floating point algorithms
- Interactive up to 150K triangles on commodity laptop

Speedup $\geq 5\times$

[Cherchi et al. 2020]
Our Booleans pipeline
Intersection resolution

Fast and Robust Mesh Arrangements using Floating-point Arithmetic
Cherchi G., Livesu M., Scateni R., Attene M.
ACM TOG 2020

1. detect intersections
2. insert new points
3. insert new segments

cached predicates
segment insertion
implementation improvements

sa2022.siggraph.org
Cached predicates

Given a point \( p \) and a plane defined by 3 points \((a, b, c)\) the orientation of \( p \) w.r.t. the plane is given by the sign of:

\[
\text{orient3D}(a, b, c, p) = \begin{vmatrix}
    a_x & a_y & a_z & 1 \\
    b_x & b_y & b_z & 1 \\
    c_x & c_y & c_z & 1 \\
    p_x & p_y & p_z & 1 \\
\end{vmatrix}
\]

[Shewchuk 1997]

exact and robust

4×4 determinant for each orient3D call

pre-computed and cached version

orient3D\((a, b, c, p) = -p_x\)\begin{vmatrix}
    a_y & a_z & 1 \\
    b_y & b_z & 1 \\
    c_y & c_z & 1 \\
\end{vmatrix} + p_y\begin{vmatrix}
    a_x & a_z & 1 \\
    b_x & b_z & 1 \\
    c_x & c_z & 1 \\
\end{vmatrix} - p_z\begin{vmatrix}
    a_x & a_y & 1 \\
    b_x & b_y & 1 \\
    c_x & c_y & 1 \\
\end{vmatrix} + \begin{vmatrix}
    a_x & a_y & a_z \\
    b_x & b_y & b_z \\
    c_x & c_y & c_z \\
\end{vmatrix}

exact and robust

single scalar product in 4D for each orient3D call

sa2022.siggraph.org
Deterministic Linear Time Constrained Triangulation using Simple Earcut

Mara Livesu, Giacomo Cherchi, Roberto Scateni, Marco Attene
IEEE TVGC 2021

1. Introduction

The generation of triangulations that conform to a given set of the segments in the lower triangle involves in addition to a constrained triangulation criteria a computing plane segments intersection. We propose a new approach to this problem, which is the combination of the classic Earcut algorithm and a new data structure that allows fast and efficient intersection detection.

2. Proposed Algorithm

The proposed algorithm is divided into two main steps: plane intersection detection and triangulation. The plane intersection detection is performed using the classic Earcut algorithm, which has a time complexity of $O(n^2)$, where $n$ is the number of polygon segments. The triangulation is performed using a linear time algorithm, which has a time complexity of $O(n)$.

3. Results

The proposed algorithm has been evaluated on a set of test cases. The results show that the algorithm is able to generate high-quality triangulations in a reasonable amount of time. The average triangulation time is $O(n)$, which is significantly faster than the classic Earcut algorithm.

4. Conclusion

The proposed algorithm provides a fast and efficient solution for triangulating plane segments with constrained edges. The algorithm is simple to implement and has been shown to produce high-quality triangulations in a reasonable amount of time. The algorithm can be used in a variety of applications, including computer graphics and computational geometry.
Inside/outside classification via ray casting

For each patch of the simplicial complex we determine its position w.r.t. the input meshes $M_1 \ldots M_n$

- efficient ray casting
- scales on patches (not on triangles)
- negligible comp. time

$M_1 \quad M_2$

$p_\infty \quad i \quad r \quad p$
Inside/outside classification via ray casting

This ray casting approach poses several technical challenges:

- **exact arrangements required**
- **exact intersections detection required**
- **manage implicit points**
- **manage ambiguous ray intersections**
Intersection classification

General case: check triangle orientation

Across triangle

\[ p \rightarrow p_\infty \]

Vol neg.: inside \rightarrow outside
Vol pos.: outside \rightarrow inside

\[ \text{vol neg.: inside \rightarrow outside} \]
\[ \text{vol pos.: outside \rightarrow inside} \]
Intersection classification

general case: check triangle orientation

across vertex

tangent vertex

across edge
tangent edge

across triangle

particular case: perturb ray by $\varepsilon$ and go to the general case

$p \rightarrow p_\infty \rightarrow i \rightarrow r$
Discussion and results
Implementation and comparisons

- Exact ray casting and exact intersection check: *Indirect Predicates* [Attene 2020]
- Efficiency and parallelism: *Google Abseil* + *Intel TBB*

We compare to

**Mesh Arrangements for Solid Geometry**
Zhou Q., Grinspun E., Zorin D., Jacobson A.
ACM TOG 2016

Most recent version in *libigl*
[Jacobson et al. 2018]
Interactive applications: rotation demo

ours: interactive up to 150K tris
libigl: 1-2 fps already at 50K tris
ours: 1-2 fps for 1M tris
Interactive applications: ARAP deformation

Apple M1 PRO
8 performance cores
32 GB Ram

ARAP [Sorkine and Alexa 2007]

interactive up to 100K tris
Large scale benchmark

Thingi10K
[Zhou and Jacobson 2016] → cleaning → 7628 clean meshes
2 × 3814 meshes

**Ours:** 3814 Booleans in 4.5 minutes

**Libigl:** 3814 Booleans in 28.3 minutes

Boolean step
5.8 mins

Splitting step
22.5 mins

Total (Libigl): 28.3 mins

Boolean step
0.47 mins
(12.2 × libigl)

Splitting step
4 mins
(5.5 × libigl)

Total (Ours): 4.5 mins

We are faster in 100% of the cases.

sa2022.siggraph.org
Processing of huge meshes

ours: from 2.29 to 19.5 seconds
libigl: from 27.33 to 545.2 seconds

we are up to $80\times$ faster than libigl in the Boolean part
we are on average $25\times$ faster than libigl in the whole pipeline
Variadic Booleans

\[
A \setminus B
\]

**ours**: 7.49 seconds

**libigl**: 61.01 seconds

\[
A \setminus \{B_1 \cup \cdots \cup B_n\}
\]

**ours**: 7.59 seconds

**libigl**: 170.93 seconds

\(B\) is a single mesh composed of 500 connected components

\(B = \{B_1 \cup B_2 \cup \cdots \cup B_{500}\}\)

\(B\) is composed of 500 meshes

sa2022.siggraph.org
Final remarks
Conclusion

- Improved mesh arrangements
- Fast and exact ray tracing

$\Rightarrow$

- One order of magnitude faster than state of art
- Interactive mesh Booleans
- Basic geometric algorithms + real-time Booleans

[Cherchi et al. 2020]
Limitations and future works

Our system is currently limited in two aspects:

- Inability to achieve interactive frame rates on very high resolution meshes
- Inability to robustly perform cascaded Booleans operations

- Smarter update of the data structures for intersections, ray casting and adjacencies
- Cascaded version of the Indirect Predicates of [Attene 2020]
Thanks!