SIGGRAPH ASIA 2022 DAEGU

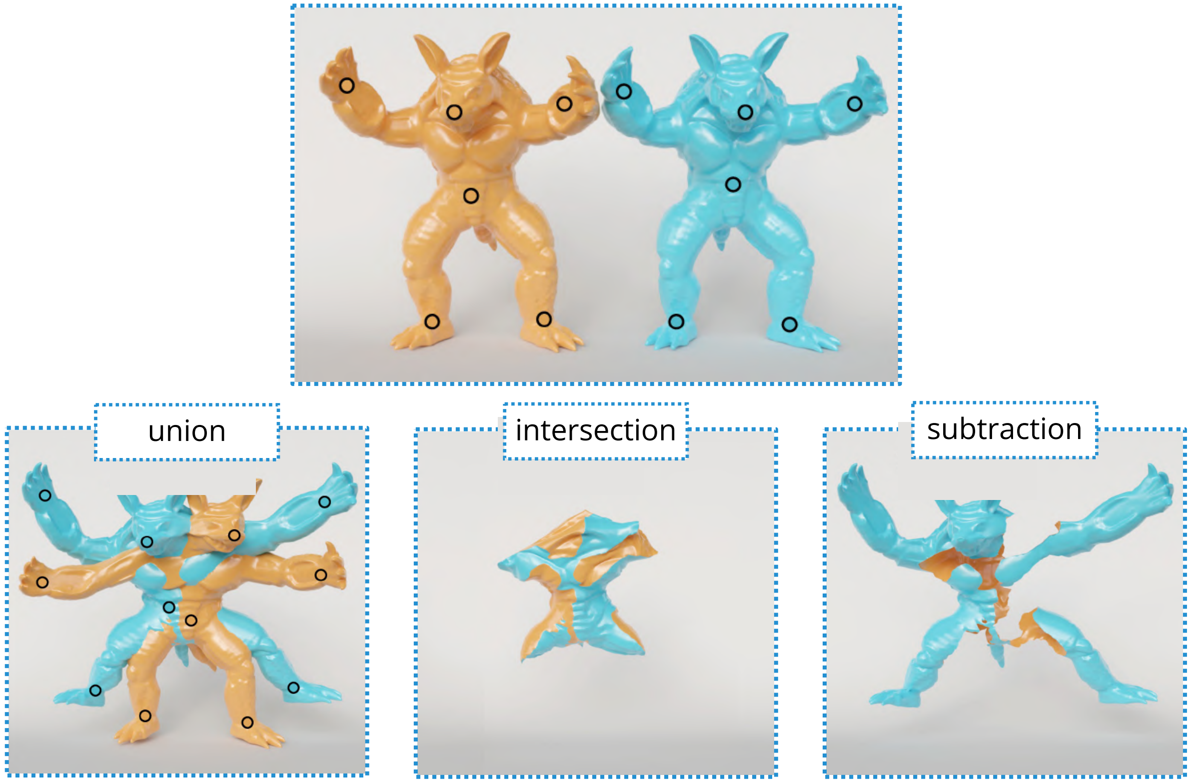
Interactive and Robust Mesh Booleans

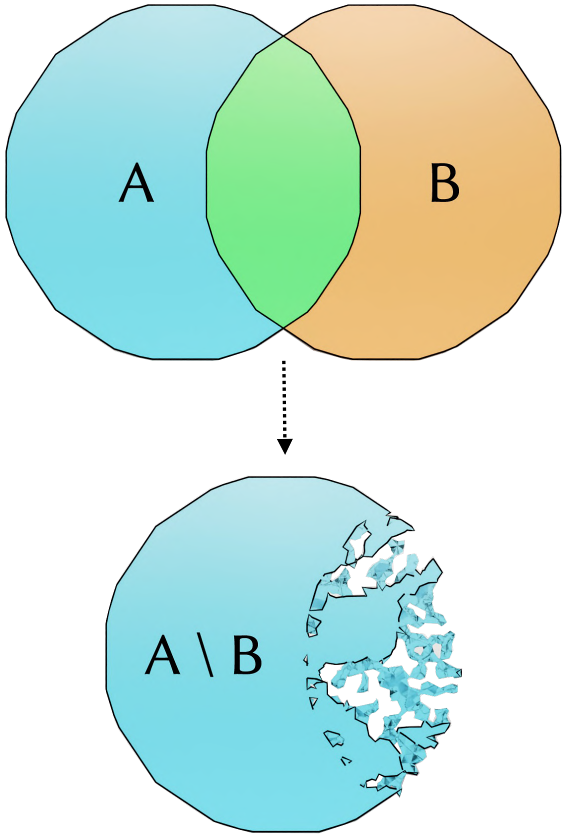
Gianmarco Cherchi (University of Cagliari, Italy)
Fabio Pellacini (Sapienza University Rome, Italy)
Marco Attene (CNR-IMATI Genoa, Italy)
Marco Livesu (CNR-IMATI, Genoa, Italy)

Mesh Booleans

conceptually simple

complex to implement correctly





Mesh Booleans algorithms

SIGGRAPH ASIA
2022 DAEGU, KOREA

floating points VS exact arithmetic

- fast
- slow
- possible errors
- exact
- geometric predicates
- snap rounding

volume-based vs surface-based

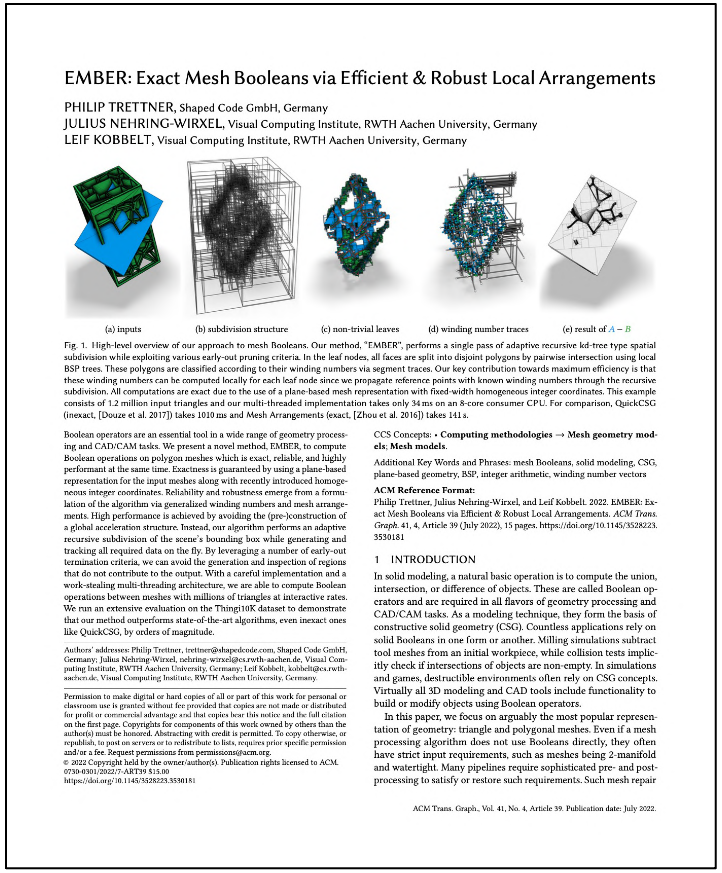
- easier I/O labeling
- more convoluted
- less efficient
- more performant
- snap rounding

exact results vs approx. results

- exact result
- needs repairing
- more convoluted
- easier implement.
- snap rounding

Robust and interactive Booleans with EMBER

SIGGRAPH ASIA
2022 DAEGU, KOREA



EMBER: Exact Mesh Booleans via Efficient & Robust Local Arrangements

PHILIP TRETTNER, Shaped Code GmbH, Germany
JULIUS NEHRING-WIRXEL, Visual Computing Institute, RWTH Aachen University, Germany
LEIF KOBBELT, Visual Computing Institute, RWTH Aachen University, Germany

(a) inputs (b) subdivision structure (c) non-trivial leaves (d) winding number traces (e) result of $A - B$

Fig. 1. High-level overview of our approach to mesh Booleans. Our method, "EMBER", performs a single pass of adaptive recursive kd-tree type spatial subdivision while exploiting various early-out pruning criteria. In the leaf nodes, all faces are split into disjoint polygons by pairwise intersection using local BSP trees. These polygons are classified according to their winding numbers via segment traces. Our key contribution towards maximum efficiency is that these winding numbers can be computed locally for each leaf node since we propagate reference points with known winding numbers through the recursive subdivision. All computations are exact due to the use of a plane-based mesh representation with fixed-width homogeneous integer coordinates. This example consists of 1.2 million input triangles and our multi-threaded implementation takes only 34 ms on an 8-core consumer CPU. For comparison, QuickCSG (inexact, [Douze et al. 2017]) takes 1010 ms and Mesh Arrangements (exact, [Zhou et al. 2016]) takes 141 s.

Boolean operators are an essential tool in a wide range of geometry processing and CAD/CAM tasks. We present a novel method, EMBER, to compute Boolean operations on polygon meshes which is exact, reliable, and highly performant at the same time. Exactness is guaranteed by using a plane-based representation for the input meshes along with recently introduced homogeneous integer coordinates. Reliability and robustness emerge from a formulation of the algorithm via generalized winding numbers and mesh arrangements. High performance is achieved by avoiding the (pre-)construction of a global acceleration structure. Instead, our algorithm performs an adaptive recursive subdivision of the scene's bounding box while generating and tracking all required data on the fly. By leveraging a number of early-out termination criteria, we can avoid the generation and inspection of regions that do not contribute to the output. With a careful implementation and a work-stealing multi-threading architecture, we are able to compute Boolean operations between meshes with millions of triangles at interactive rates. We run an extensive evaluation on the Thingi10K dataset to demonstrate that our method outperforms state-of-the-art algorithms, even inexact ones like QuickCSG, by orders of magnitude.

CCS Concepts: • **Computing methodologies → Mesh geometry models**; **Mesh models**.

Additional Key Words and Phrases: mesh Booleans, solid modeling, CSG, plane-based geometry, BSP, integer arithmetic, winding number vectors

ACM Reference Format:
Philip Trettner, Julius Nehring-Wirxel, and Leif Kobbelt. 2022. EMBER: Exact Mesh Booleans via Efficient & Robust Local Arrangements. *ACM Trans. Graph.* 41, 4, Article 39 (July 2022), 15 pages. https://doi.org/10.1145/3528223.3530181

1 INTRODUCTION

In solid modeling, a natural basic operation is to compute the union, intersection, or difference of objects. These are called Boolean operators and are required in all flavors of geometry processing and CAD/CAM tasks. As a modeling technique, they form the basis of constructive solid geometry (CSG). Countless applications rely on solid Booleans in one form or another. Milling simulations subtract tool meshes from an initial workpiece, while collision tests implicitly check if intersections of objects are non-empty. In simulations and games, destructible environments often rely on CSG concepts. Virtually all 3D modeling and CAD tools include functionality to build or modify objects using Boolean operators.

In this paper, we focus on arguably the most popular representation of geometry: triangle and polygonal meshes. Even if a mesh processing algorithm does not use Booleans directly, they often have strict input requirements, such as meshes being 2-manifold and watertight. Many pipelines require sophisticated pre- and post-processing to satisfy or restore such requirements. Such mesh repair

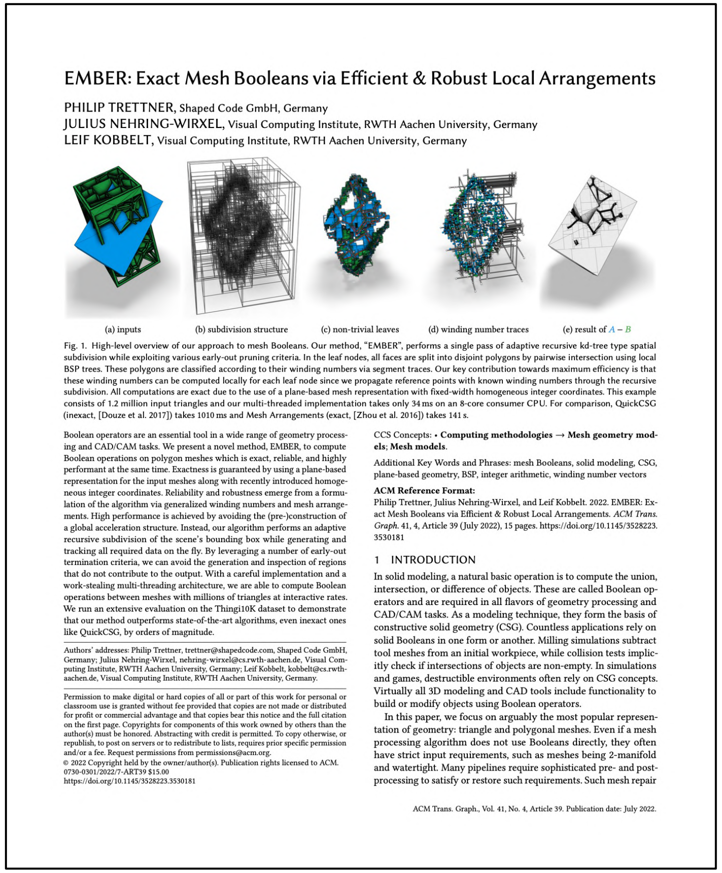
Authors' addresses: Philip Trettner, trettner@shapedcode.com, Shaped Code GmbH, Germany; Julius Nehring-Wirxel, nehring-wirxel@cs.rwth-aachen.de, Visual Computing Institute, RWTH Aachen University, Germany; Leif Kobbelt, kobbelt@cs.rwth-aachen.de, Visual Computing Institute, RWTH Aachen University, Germany.

0730-0301/2022/7-ART39 $15.00
https://doi.org/10.1145/3528223.3530181

fastest Boolean pipeline

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

Robust and interactive Booleans with EMBER



EMBER: Exact Mesh Booleans via Efficient & Robust Local Arrangements

PHILIP TRETTNER, Shaped Code GmbH, Germany
JULIUS NEHRING-WIRXEL, Visual Computing Institute, RWTH Aachen University, Germany
LEIF KOBBELT, Visual Computing Institute, RWTH Aachen University, Germany

(a) inputs (b) subdivision structure (c) non-trivial leaves (d) winding number traces (e) result of A – B

Fig. 1. High-level overview of our approach to mesh Booleans. Our method, "EMBER", performs a single pass of adaptive recursive kd-tree type spatial subdivision while exploiting various early-out pruning criteria. In the leaf nodes, all faces are split into disjoint polygons by pairwise intersection using local BSP trees. These polygons are classified according to their winding numbers via segment traces. Our key contribution towards maximum efficiency is that these winding numbers can be computed locally for each leaf node since we propagate reference points with known winding numbers through the recursive subdivision. All computations are exact due to the use of a plane-based mesh representation with fixed-width homogeneous integer coordinates. This example consists of 1.2 million input triangles and our multi-threaded implementation takes only 34 ms on an 8-core consumer CPU. For comparison, QuickCSG (inexact, [Douze et al. 2017]) takes 1010 ms and Mesh Arrangements (exact, [Zhou et al. 2016]) takes 141 s.

Boolean operators are an essential tool in a wide range of geometry processing and CAD/CAM tasks. We present a novel method, EMBER, to compute Boolean operations on polygon meshes which is exact, reliable, and highly performant at the same time. Exactness is guaranteed by using a plane-based representation for the input meshes along with recently introduced homogeneous integer coordinates. Reliability and robustness emerge from a formulation of the algorithm via generalized winding numbers and mesh arrangements. High performance is achieved by avoiding the (pre-)construction of a global acceleration structure. Instead, our algorithm performs an adaptive recursive subdivision of the scene's bounding box while generating and tracking all required data on the fly. By leveraging a number of early-out termination criteria, we can avoid the generation and inspection of regions that do not contribute to the output. With a careful implementation and a work-stealing multi-threading architecture, we are able to compute Boolean operations between meshes with millions of triangles at interactive rates. We run an extensive evaluation on the Thingi10K dataset to demonstrate that our method outperforms state-of-the-art algorithms, even inexact ones like QuickCSG, by orders of magnitude.

Authors' addresses: Philip Trettner, trettner@shapedcode.com, Shaped Code GmbH, Germany; Julius Nehring-Wirxel, nehring-wirxel@cs.rwth-aachen.de, Visual Computing Institute, RWTH Aachen University, Germany; Leif Kobbelt, kobbelt@cs.rwth-aachen.de, Visual Computing Institute, RWTH Aachen University, Germany.

0730-0301/2022/7-ART39 $15.00
https://doi.org/10.1145/3528223.3530181

CCS Concepts: • **Computing methodologies → Mesh geometry models**; **Mesh models**.

Additional Key Words and Phrases: mesh Booleans, solid modeling, CSG, plane-based geometry, BSP, integer arithmetic, winding number vectors

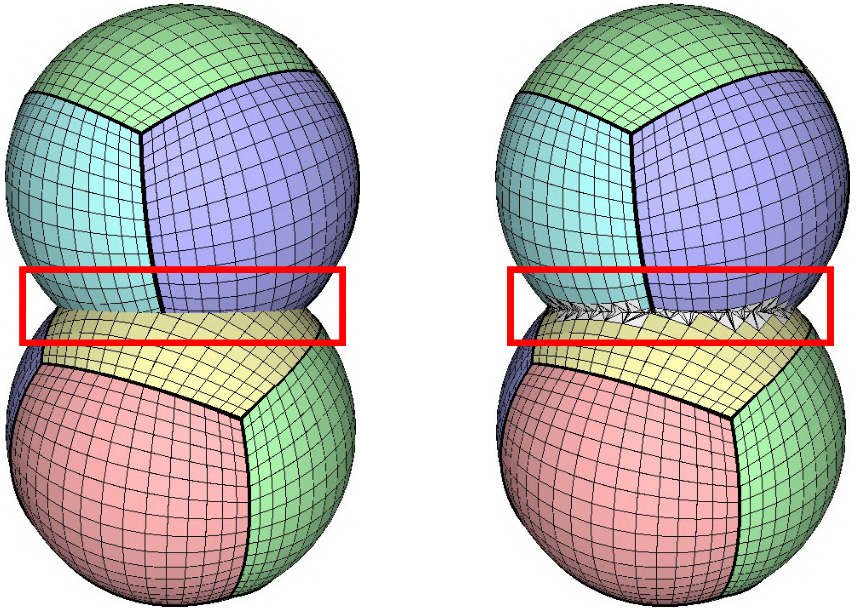
ACM Reference Format:
Philip Trettner, Julius Nehring-Wirxel, and Leif Kobbelt. 2022. EMBER: Exact Mesh Booleans via Efficient & Robust Local Arrangements. *ACM Trans. Graph.* 41, 4, Article 39 (July 2022), 15 pages. https://doi.org/10.1145/3528223.3530181

1 INTRODUCTION

In solid modeling, a natural basic operation is to compute the union, intersection, or difference of objects. These are called Boolean operators and are required in all flavors of geometry processing and CAD/CAM tasks. As a modeling technique, they form the basis of constructive solid geometry (CSG). Countless applications rely on solid Booleans in one form or another. Milling simulations subtract tool meshes from an initial workpiece, while collision tests implicitly check if intersections of objects are non-empty. In simulations and games, destructible environments often rely on CSG concepts. Virtually all 3D modeling and CAD tools include functionality to build or modify objects using Boolean operators.

In this paper, we focus on arguably the most popular representation of geometry: triangle and polygonal meshes. Even if a mesh processing algorithm does not use Booleans directly, they often have strict input requirements, such as meshes being 2-manifold and watertight. Many pipelines require sophisticated pre- and post-processing to satisfy or restore such requirements. Such mesh repair

no compatibility with existing geometry processing tasks

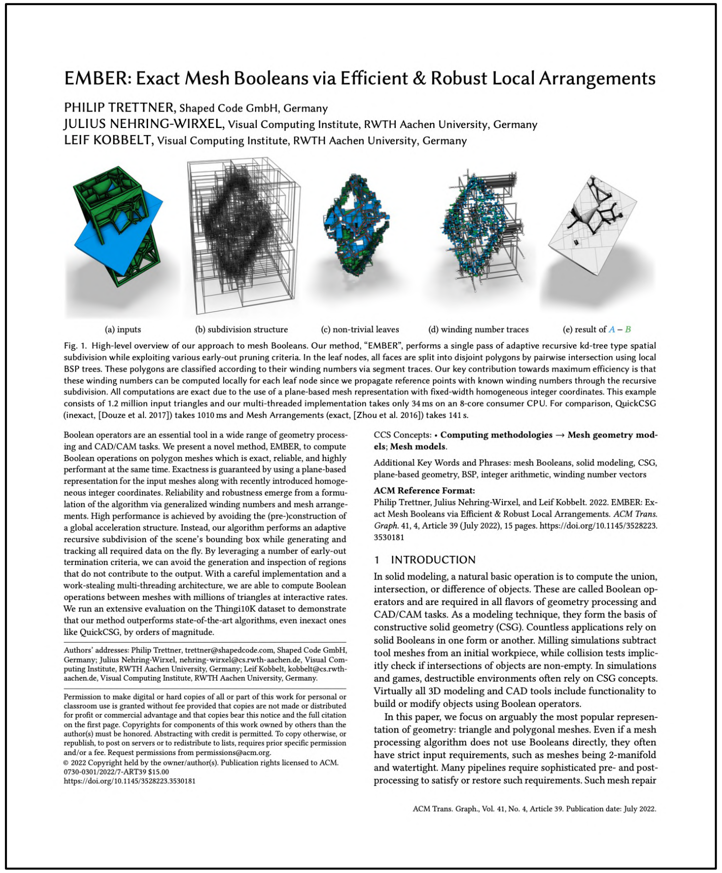


example:
QuadMixer
[Nuvoli et al. 2019]

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

Robust and interactive Booleans with EMBER

SIGGRAPH ASIA
2022 DAEGU, KOREA



EMBER: Exact Mesh Booleans via Efficient & Robust Local Arrangements

PHILIP TRETTNER, Shaped Code GmbH, Germany
JULIUS NEHRING-WIRXEL, Visual Computing Institute, RWTH Aachen University, Germany
LEIF KOBBELT, Visual Computing Institute, RWTH Aachen University, Germany

(a) inputs (b) subdivision structure (c) non-trivial leaves (d) winding number traces (e) result of A − B

Fig. 1. High-level overview of our approach to mesh Booleans. Our method, "EMBER", performs a single pass of adaptive recursive kd-tree type spatial subdivision while exploiting various early-out pruning criteria. In the leaf nodes, all faces are split into disjoint polygons by pairwise intersection using local BSP trees. These polygons are classified according to their winding numbers via segment traces. Our key contribution towards maximum efficiency is that these winding numbers can be computed locally for each leaf node since we propagate reference points with known winding numbers through the recursive subdivision. All computations are exact due to the use of a plane-based mesh representation with fixed-width homogeneous integer coordinates. This example consists of 1.2 million input triangles and our multi-threaded implementation takes only 34 ms on an 8-core consumer CPU. For comparison, QuickCSG (inexact, [Douze et al. 2017]) takes 1010 ms and Mesh Arrangements (exact, [Zhou et al. 2016]) takes 141 s.

Boolean operators are an essential tool in a wide range of geometry processing and CAD/CAM tasks. We present a novel method, EMBER, to compute Boolean operations on polygon meshes which is exact, reliable, and highly performant at the same time. Exactness is guaranteed by using a plane-based representation for the input meshes along with recently introduced homogeneous integer coordinates. Reliability and robustness emerge from a formulation of the algorithm via generalized winding numbers and mesh arrangements. High performance is achieved by avoiding the (pre-)construction of a global acceleration structure. Instead, our algorithm performs an adaptive recursive subdivision of the scene's bounding box while generating and tracking all required data on the fly. By leveraging a number of early-out termination criteria, we can avoid the generation and inspection of regions that do not contribute to the output. With a careful implementation and a work-stealing multi-threading architecture, we are able to compute Boolean operations between meshes with millions of triangles at interactive rates. We run an extensive evaluation on the Thingi10K dataset to demonstrate that our method outperforms state-of-the-art algorithms, even inexact ones like QuickCSG, by orders of magnitude.

Authors' addresses: Philip Trettner, trettner@shapedcode.com, Shaped Code GmbH, Germany; Julius Nehring-Wirxel, nehring-wirxel@cs.rwth-aachen.de, Visual Computing Institute, RWTH Aachen University, Germany; Leif Kobbelt, kobbelt@cs.rwth-aachen.de, Visual Computing Institute, RWTH Aachen University, Germany.

0730-0301/2022/7-ART39 $15.00
https://doi.org/10.1145/3528223.3530181

CCS Concepts: • **Computing methodologies → Mesh geometry models**; **Mesh models**.

Additional Key Words and Phrases: mesh Booleans, solid modeling, CSG, plane-based geometry, BSP, integer arithmetic, winding number vectors

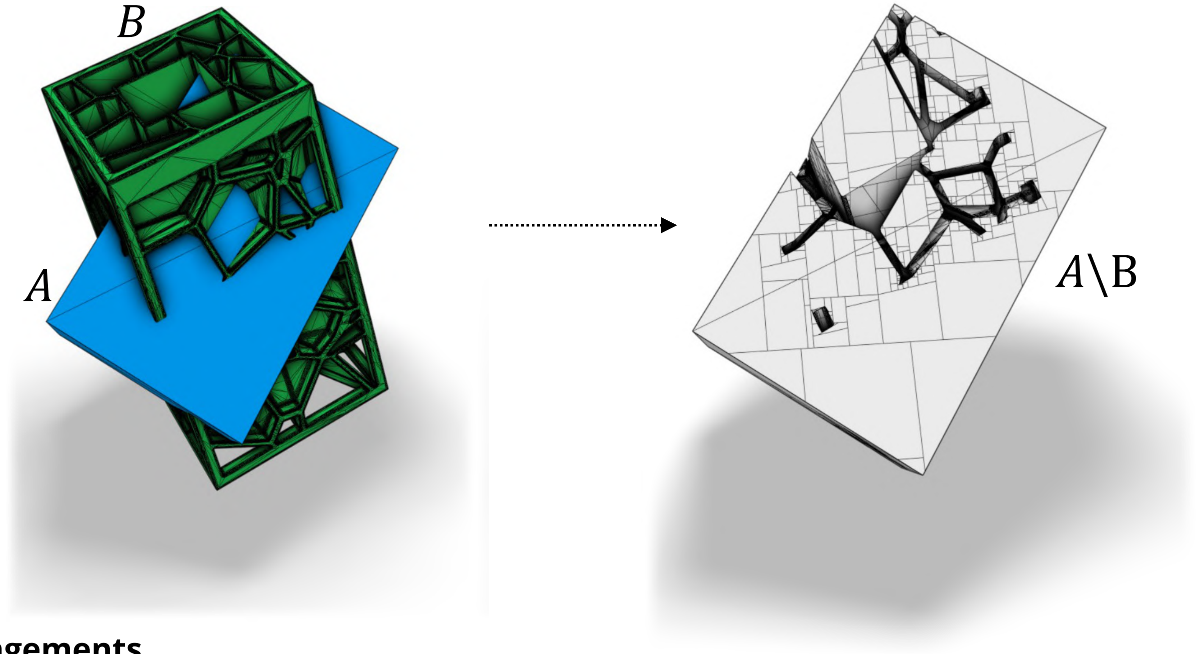
ACM Reference Format:
Philip Trettner, Julius Nehring-Wirxel, and Leif Kobbelt. 2022. EMBER: Exact Mesh Booleans via Efficient & Robust Local Arrangements. *ACM Trans. Graph.* 41, 4, Article 39 (July 2022), 15 pages. https://doi.org/10.1145/3528223.3530181

1 INTRODUCTION

In solid modeling, a natural basic operation is to compute the union, intersection, or difference of objects. These are called Boolean operators and are required in all flavors of geometry processing and CAD/CAM tasks. As a modeling technique, they form the basis of constructive solid geometry (CSG). Countless applications rely on solid Booleans in one form or another. Milling simulations subtract tool meshes from an initial workpiece, while collision tests implicitly check if intersections of objects are non-empty. In simulations and games, destructible environments often rely on CSG concepts. Virtually all 3D modeling and CAD tools include functionality to build or modify objects using Boolean operators.

In this paper, we focus on arguably the most popular representation of geometry: triangle and polygonal meshes. Even if a mesh processing algorithm does not use Booleans directly, they often have strict input requirements, such as meshes being 2-manifold and watertight. Many pipelines require sophisticated pre- and post-processing to satisfy or restore such requirements. Such mesh repair

unnecessary stitching in the result mesh



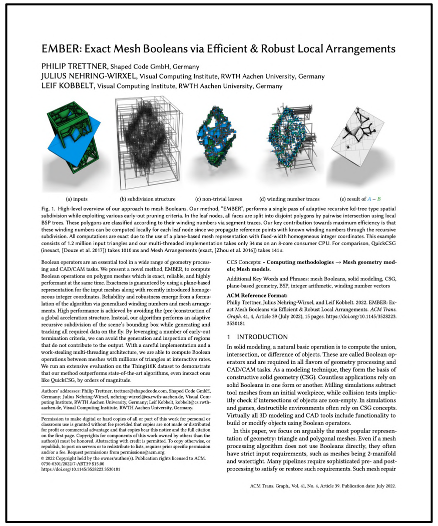
EMBER: Exact Mesh Booleans via Efficient & Robust Local Arrangements
Trettner P., Nehring-Wirxel J., Kobbelt L.
ACM TOG 2022

Robust and interactive Booleans with EMBER

SIGGRAPH ASIA
2022 DAEGU, KOREA

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


EMBER: Exact Mesh Booleans via Efficient & Robust Local Arrangements
PHILIP TRETTNER, Shaped Code GmbH, Germany
JULIUS NEHRING-WIRXEL, Visual Computing Institute, RWTH Aachen University, Germany
LEIF KOBBELT, Visual Computing Institute, RWTH Aachen University, Germany

speed

compatibility with existing geometry processing tasks

VS

possible slowdown

compatibility with existing geometry processing tasks

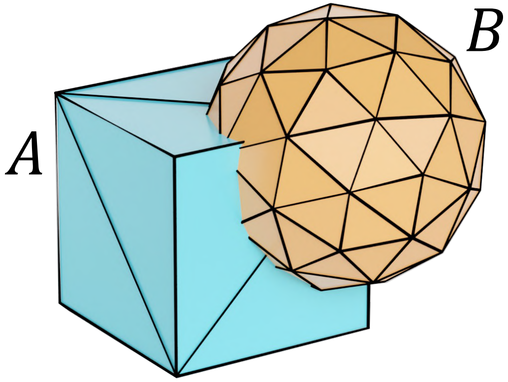

Interactive and Robust Mesh Booleans
GIANMARCO CHERCHI, University of Cagliari, Italy
FABIO PELLACINI, Sapienza University of Rome, Italy
MARCO ATTENE, CNR IMATI, Italy
MARCO LIVESU, CNR IMATI, Italy

[OURS]

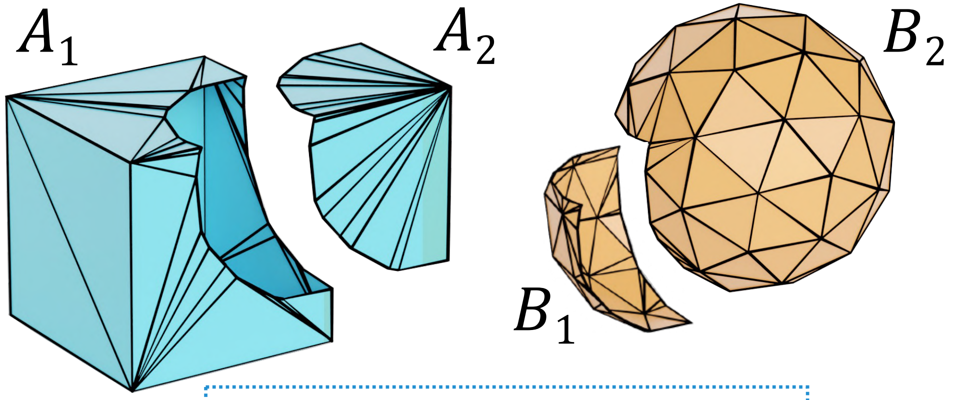
different goals

sa2022.siggraph.org

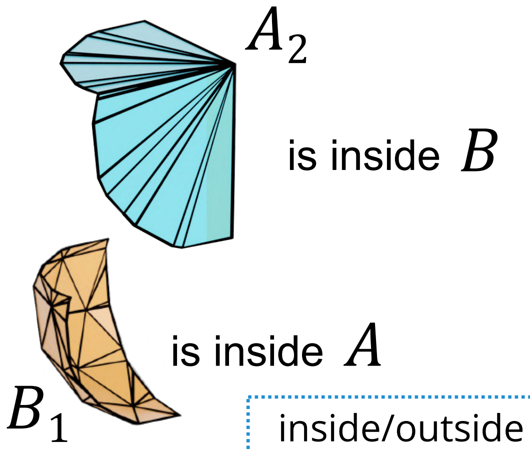
The general pipeline



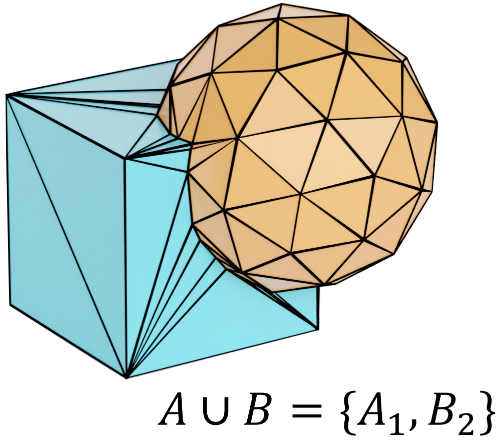
input meshes

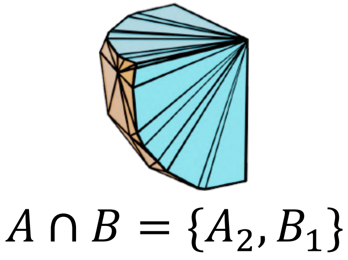


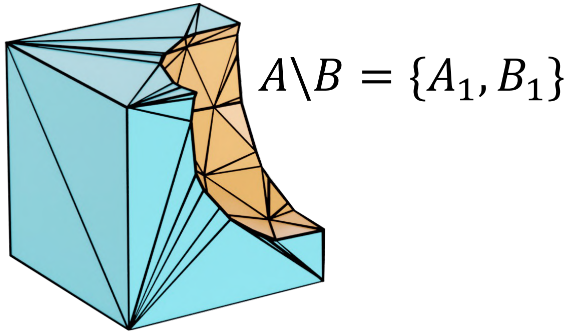
resolve mesh intersections and create conforming patches

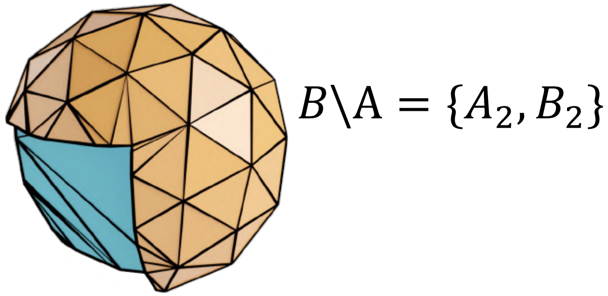


inside/outside patch labelling





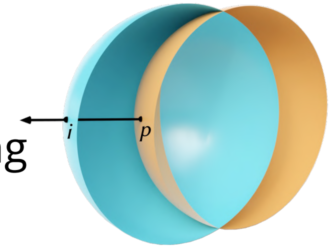




filter and merge patches to form the output mesh

Our main contributions

exact
ray tracing



robustness of exact floating point methods

+

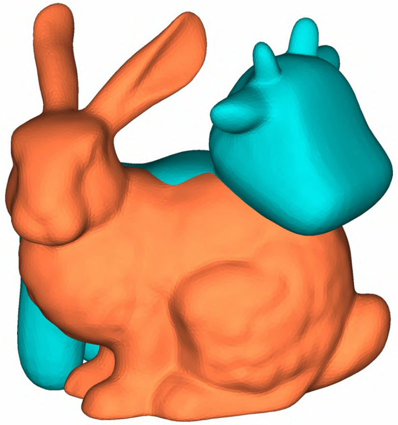
fast inside/outside triangle classification

=

one order of magnitude **faster** than state of art

compatible with existing floating point algorithms

interactive up to **150K** triangles on commodity laptop



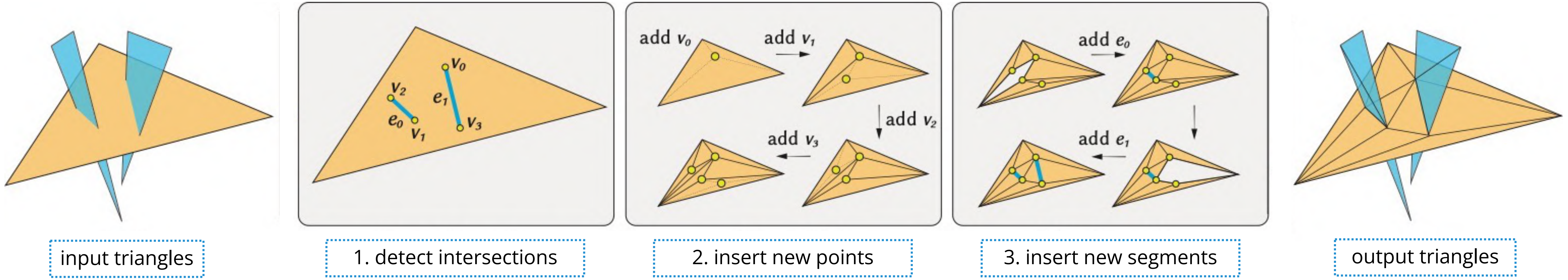
speedup
≥ 5×

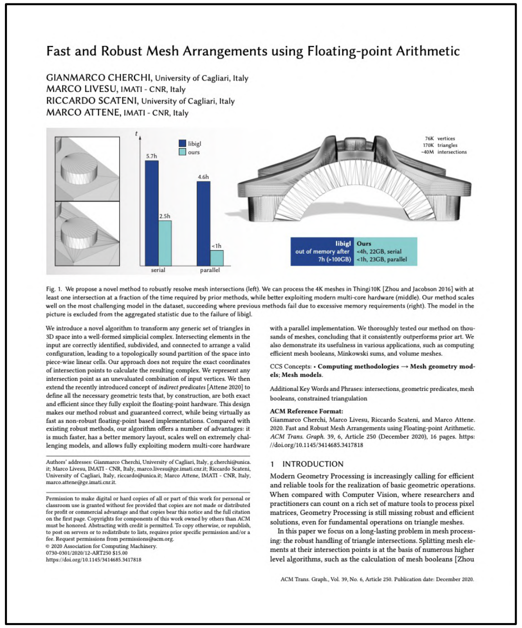
[Cherchi et al. 2020]

SIGGRAPH ASIA
2022 DAEGU, KOREA

Our Booleans pipeline

Intersection resolution




Fast and Robust Mesh Arrangements using Floating-point Arithmetic

GIANMARCO CHERCHI, University of Cagliari, Italy
MARCO LIVESU, IMATI - CNR, Italy
RICCARDO SCATENI, University of Cagliari, Italy
MARCO ATTENE, IMATI - CNR, Italy

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

+

cached predicates

segment insertion

implementation improvements

sa2022.siggraph.org

Cached predicates

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

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

[Shewchuk 1997]

exact and robust

4×4 determinant for each *orient3D* call

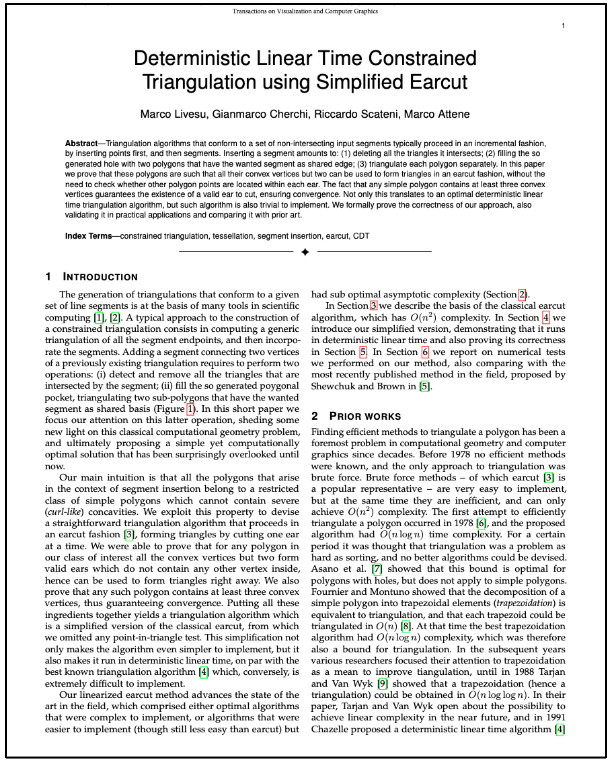
pre-computed and cached version

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

exact and robust

single scalar product in 4D for each *orient3D* call

Segment insertion and low-level implementation



Transactions on Visualization and Computer Graphics

1

Deterministic Linear Time Constrained Triangulation using Simplified Earcut

Marco Livesu, Gianmarco Cherchi, Riccardo Scateni, Marco Attene

Abstract—Triangulation algorithms that conform to a set of non-intersecting input segments typically proceed in an incremental fashion, by inserting points first, and then segments. Inserting a segment amounts to: (1) deleting all the triangles it intersects; (2) filling the so generated hole with two polygons that have the wanted segment as shared edge; (3) triangulate each polygon separately. In this paper we prove that these polygons are such that all their convex vertices but two can be used to form triangles in an earcut fashion, without the need to check whether other polygon points are located within each ear. The fact that any simple polygon contains at least three convex vertices guarantees the existence of a valid ear to cut, ensuring convergence. Not only this translates to an optimal deterministic linear time triangulation algorithm, but such algorithm is also trivial to implement. We formally prove the correctness of our approach, also validating it in practical applications and comparing it with prior art.

Index Terms—constrained triangulation, tessellation, segment insertion, earcut, CDT

1 INTRODUCTION

The generation of triangulations that conform to a given set of line segments is at the basis of many tools in scientific computing [1], [2]. A typical approach to the construction of a constrained triangulation consists in computing a generic triangulation of all the segment endpoints, and then incorporate the segments. Adding a segment connecting two vertices of a previously existing triangulation requires to perform two operations: (i) detect and remove all the triangles that are intersected by the segment; (ii) fill the so generated poygonal pocket, triangulating two sub-polygons that have the wanted segment as shared basis (Figure 1). In this short paper we focus our attention on this latter operation, sheding some new light on this classical computational geometry problem, and ultimately proposing a simple yet computationally optimal solution that has been surprisingly overlooked until now.

Our main intuition is that all the polygons that arise in the context of segment insertion belong to a restricted class of simple polygons which cannot contain severe (*curl-like*) concavities. We exploit this property to devise a straightforward triangulation algorithm that proceeds in an earcut fashion [3], forming triangles by cutting one ear at a time. We were able to prove that for any polygon in our class of interest all the convex vertices but two form valid ears which do not contain any other vertex inside, hence can be used to form triangles right away. We also prove that any such polygon contains at least three convex vertices, thus guaranteeing convergence. Putting all these ingredients together yields a triangulation algorithm which is a simplified version of the classical earcut, from which we omitted any point-in-triangle test. This simplification not only makes the algorithm even simpler to implement, but it also makes it run in deterministic linear time, on par with the best known triangulation algorithm [4] which, conversely, is extremely difficult to implement.

Our linearized earcut method advances the state of the art in the field, which comprised either optimal algorithms that were complex to implement, or algorithms that were easier to implement (though still less easy than earcut) but had sub optimal asymptotic complexity (Section 2).

In Section 3 we describe the basis of the classical earcut algorithm, which has $O(n^2)$ complexity. In Section 4 we introduce our simplified version, demonstrating that it runs in deterministic linear time and also proving its correctness in Section 5. In Section 6 we report on numerical tests we performed on our method, also comparing with the most recently published method in the field, proposed by Shewchuk and Brown in [5].

2 PRIOR WORKS

Finding efficient methods to triangulate a polygon has been a foremost problem in computational geometry and computer graphics since decades. Before 1978 no efficient methods were known, and the only approach to triangulation was brute force. Brute force methods – of which earcut [3] is a popular representative – are very easy to implement, but at the same time they are inefficient, and can only achieve $O(n^2)$ complexity. The first attempt to efficiently triangulate a polygon occurred in 1978 [6], and the proposed algorithm had $O(n \log n)$ time complexity. For a certain period it was thought that triangulation was a problem as hard as sorting, and no better algorithms could be devised. Asano et al. [7] showed that this bound is optimal for polygons with holes, but does not apply to simple polygons. Fournier and Montuno showed that the decomposition of a simple polygon into trapezoidal elements (*trapezoidation*) is equivalent to triangulation, and that each trapezoid could be triangulated in $O(n)$ [8]. At that time the best trapezoidation algorithm had $O(n \log n)$ complexity, which was therefore also a bound for triangulation. In the subsequent years various researchers focused their attention to trapezoidation as a mean to improve tiangulation, until in 1988 Tarjan and Van Wyk [9] showed that a trapezoidation (hence a triangulation) could be obtained in $O(n \log \log n)$. In their paper, Tarjan and Van Wyk open about the possibility to achieve linear complexity in the near future, and in 1991 Chazelle proposed a deterministic linear time algorithm [4]

Deterministic Linear Time Constrained Triangulation using Simple Earcut
Livesu M., Cherchi G., Scateni R., Attene M.
IEEE TVGC 2021

classic Earcut
$O(n^2)$

linear Earcut
$O(n)$

n = number of polygon segments

+

specialized data structures
and
massive parallelization

sa2022.siggraph.org

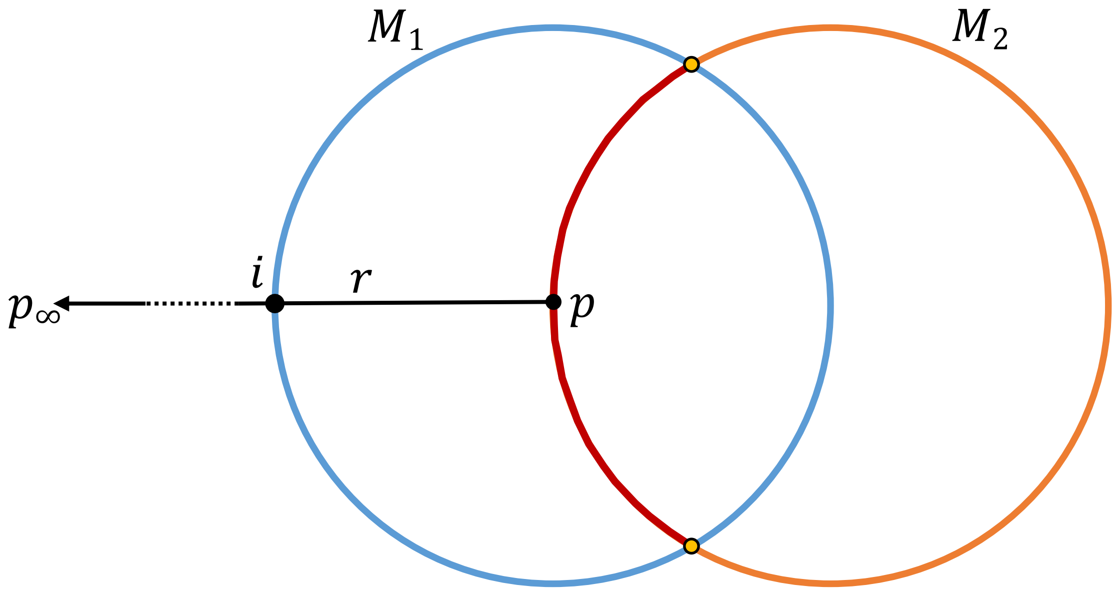
Inside/outside classification via ray casting

For each patch of the simplicial complex we determine its position w.r.t. the input meshes M_1 ... Mn

efficient ray casting

scales on patches (not on triangles)

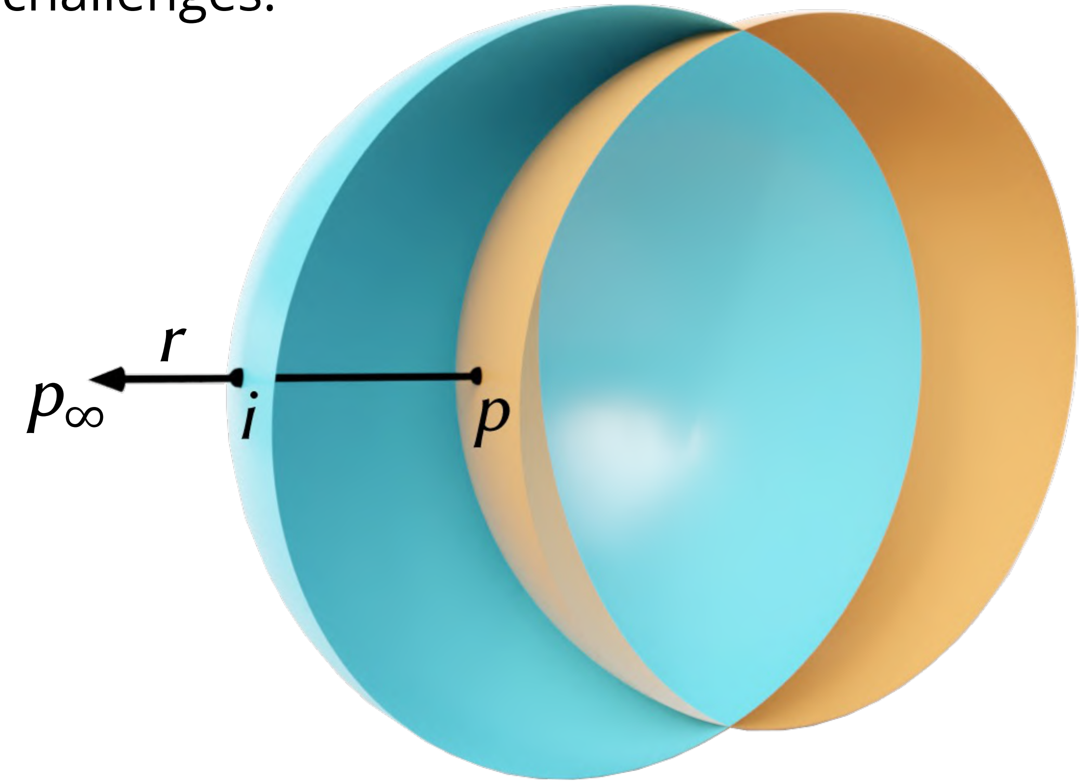
negligible comp. time



Inside/outside classification via ray casting

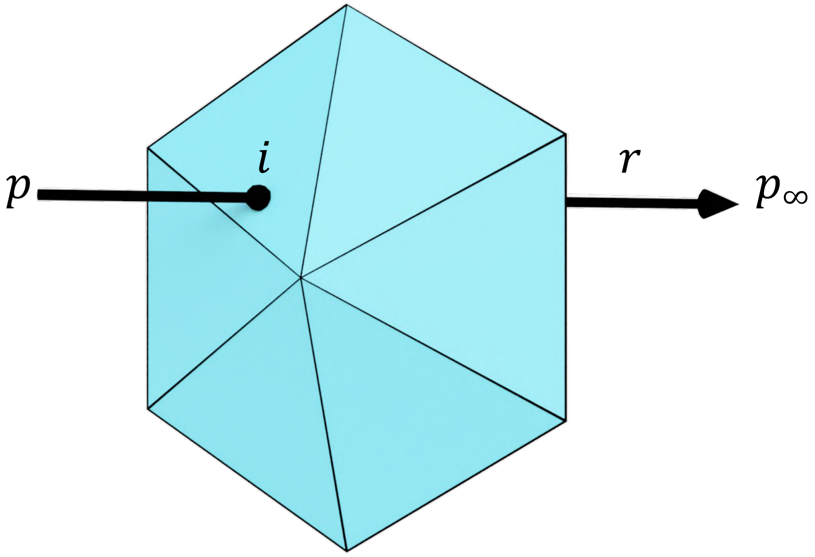
This ray casting approach poses several technical challenges:

- exact arrangements required
- exact intersections detection required
- manage implicit points
- manage ambiguous ray intersections

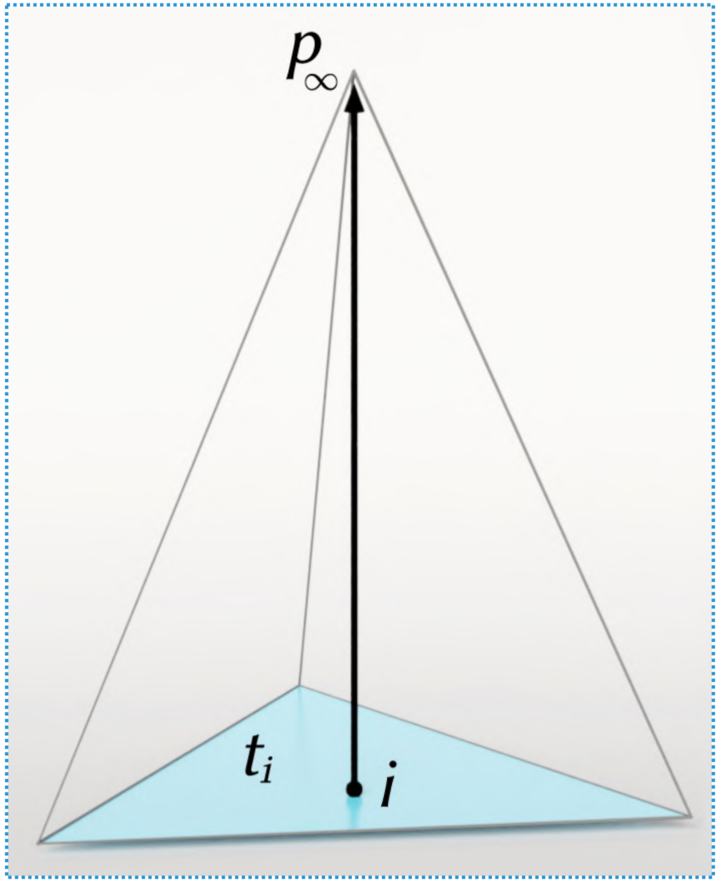


Intersection classification

across triangle



general case:
check triangle orientation

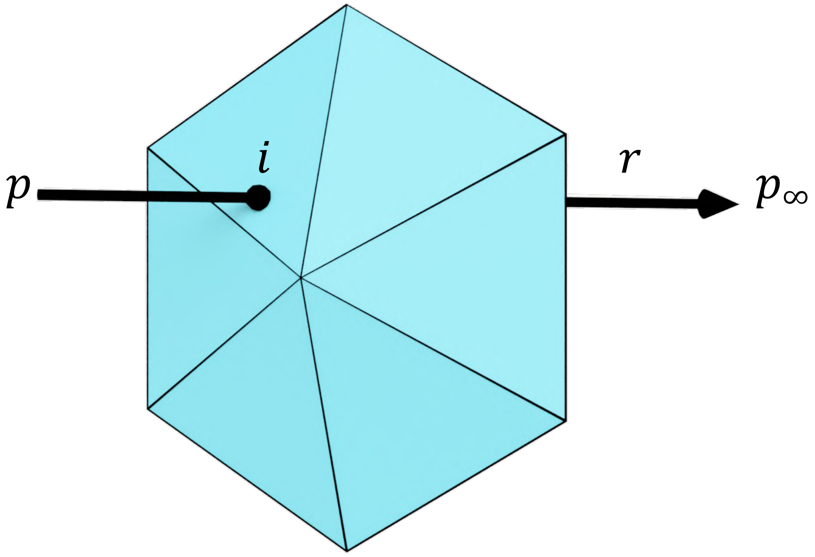


vol neg. : inside → outside

vol pos. : outside → inside

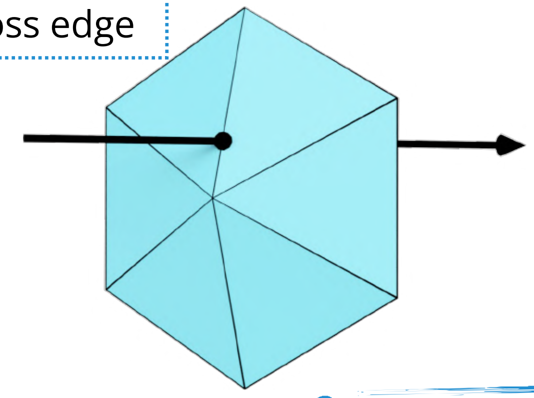
Intersection classification

across triangle

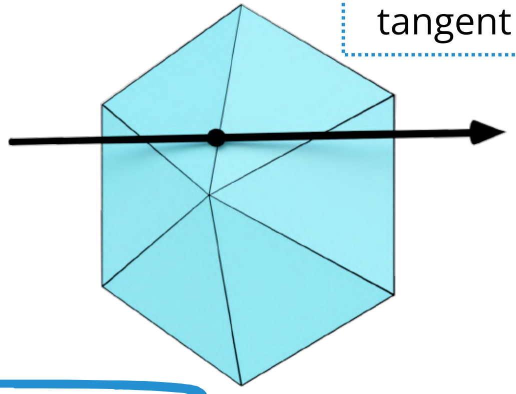


general case:
check triangle orientation

across edge

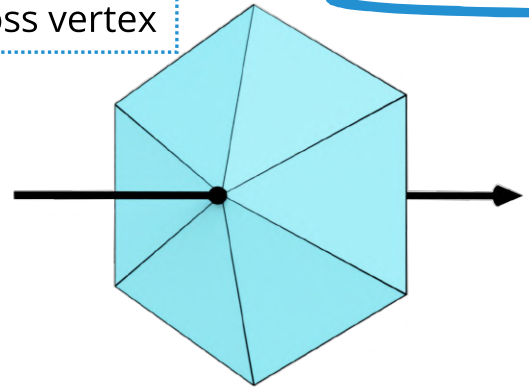


tangent edge

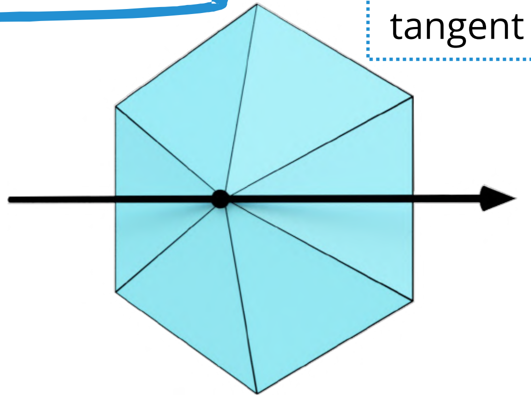


particular case:
perturb ray by ε and go to the general case

across vertex



tangent vertex



SIGGRAPH ASIA
2022 DAEGU, KOREA

Discussion and results

Implementation and comparisons

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

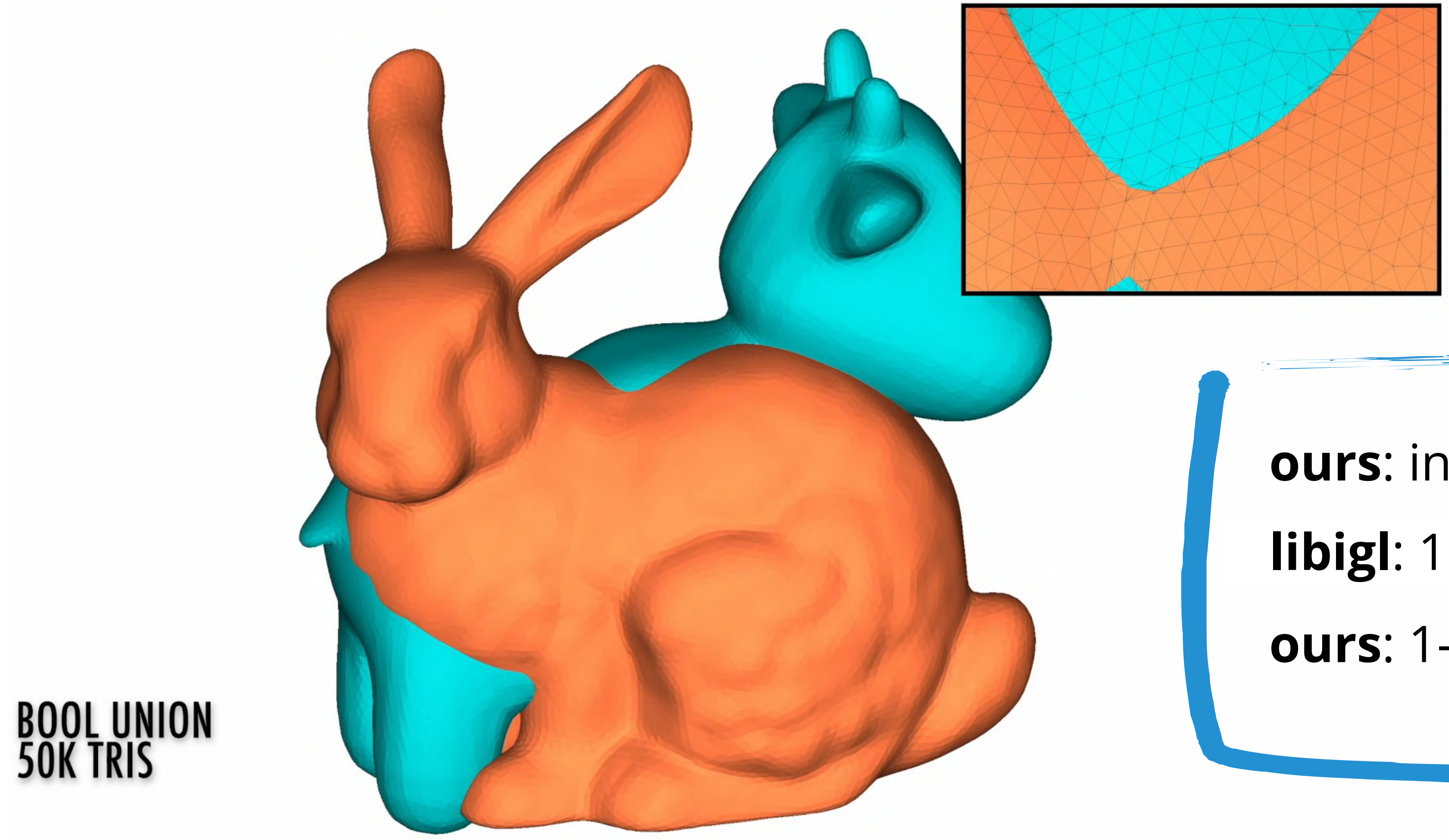
We compare to

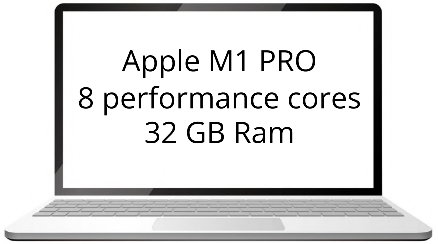

Mesh Arrangements for Solid Geometry
Qingnan Zhou[1] Eitan Grinspun[2] Denis Zorin[1] Alec Jacobson[3]
[1]New York University [2]Columbia University

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



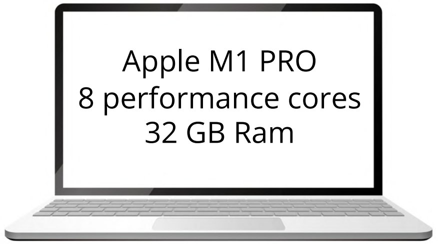


ours: interactive up to **150K** tris

libigl: 1-2 fps already at **50K** tris

ours: 1-2 fps for **1M** tris

Interactive applications: ARAP deformation



ARAP [Sorkine and Alexa 2007]

interactive up to **100K** tris

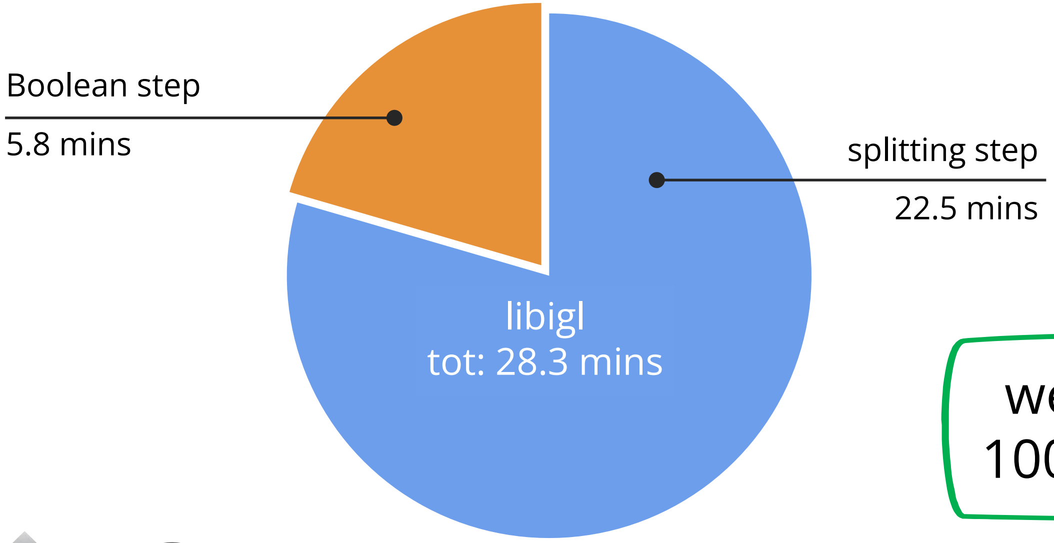
Large scale benchmark

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

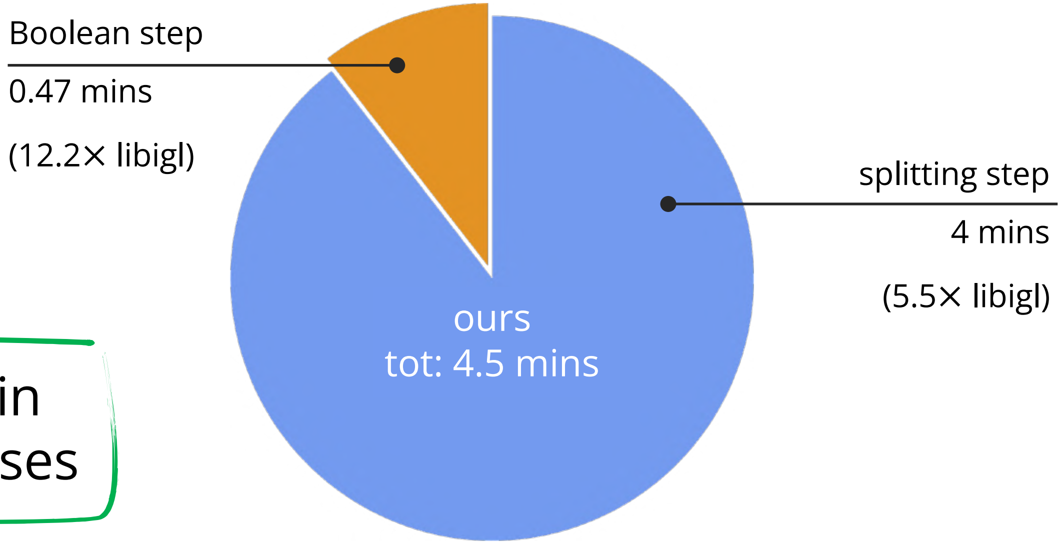
Intel i9 1.2 GHz
12 cores
128 GB Ram

ours: 3814 Booleans in **4.5 minutes**

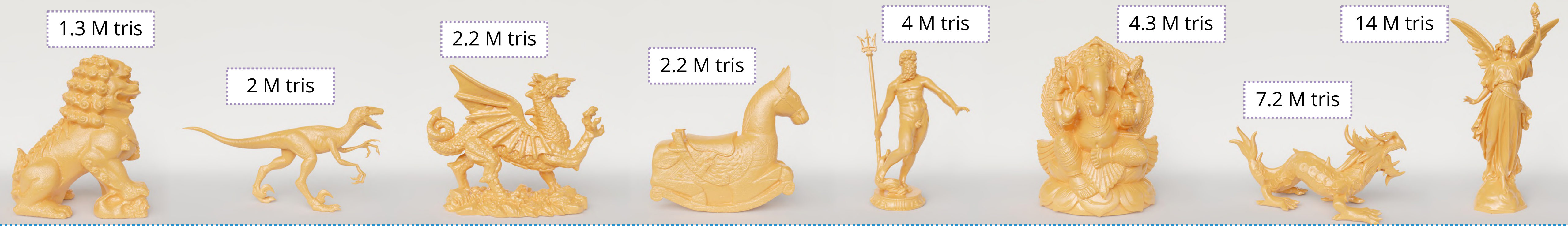
libigl: 3814 Booleans in **28.3 minutes**



we are faster in
100% of the cases

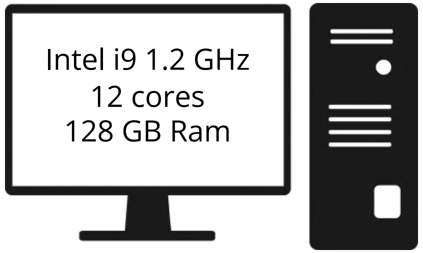


Processing of huge meshes



ours: from **2.29** to **19.5** seconds

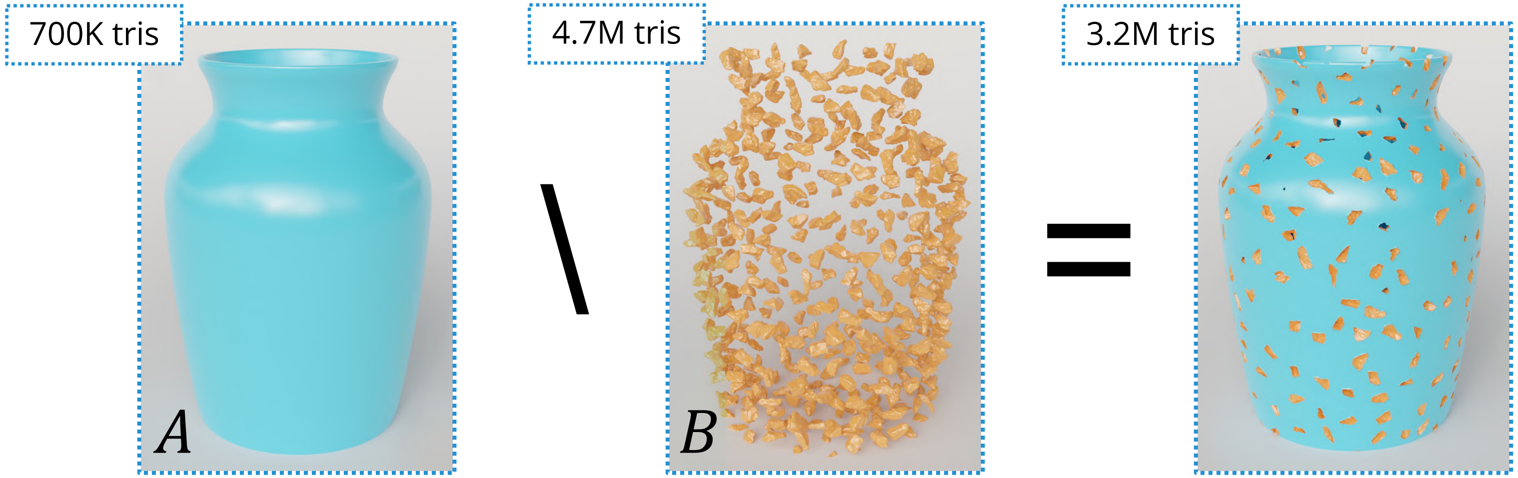
libigl: from **27.33** to **545.2** seconds

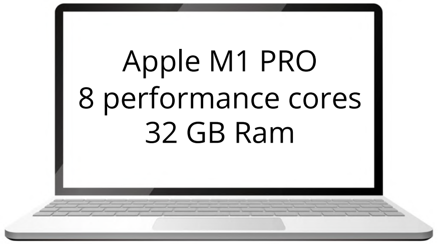


we are up to **80**×
faster than libigl
in the Boolean part

we are on average **25**×
faster than libigl
in the whole pipeline

Variadic Booleans





$$A \setminus B$$

ours: 7.49 seconds

libigl: 61.01 seconds

B is a single mesh
composed of 500 conn. components

$$A \setminus \{B_1 \cup \cdots \cup Bn\}$$

ours: 7.59 seconds

libigl: 170.93 seconds

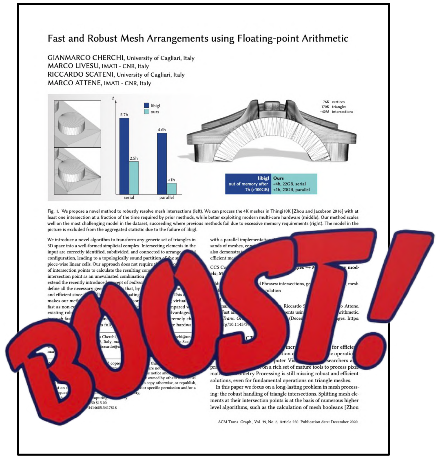
$B = \{B_1 \cup B_2 \cup \cdots \cup B_{500}\}$
B is composed of 500 meshes

SIGGRAPH ASIA
2022 DAEGU, KOREA

Final remarks

Conclusion

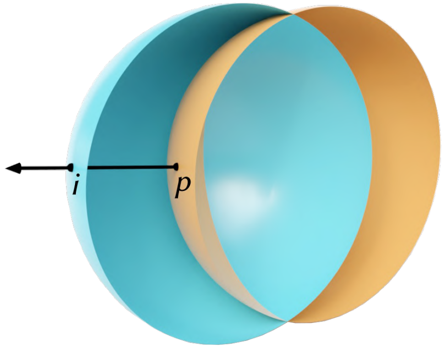
improved mesh arrangements



[Cherchi et al. 2020]

+

fast and exact ray tracing



=

- one order of magnitude faster than state of art
- interactive mesh Booleans
- basic geometric algorithms + real-time Booleans

Limitations and future works

Our system is current limited in two aspects:

Limitation	Future work
inability to achieve interactive frame rates on very high resolution meshes	smarter update of the data structures for intersections, ray casting and adjacencies
inability to robustly perform cascaded Booleans operations	cascaded version of the Indirect Predicates of [Attene 2020]

Thanks!

CODE