Adaptive Constructive Solid Geometry with constant evaluation complexity for modeling implicitly defined complex objects

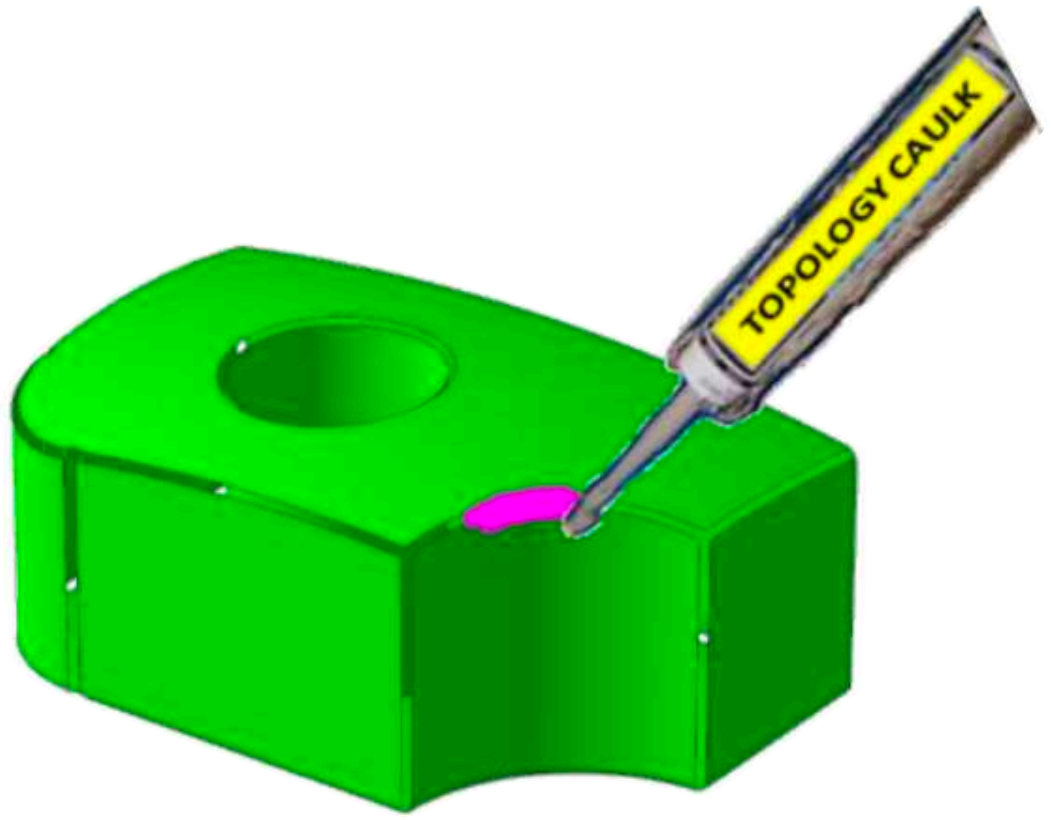
Student: Egor Chulkov
Research Advisor: Oleg V. Vasilyev

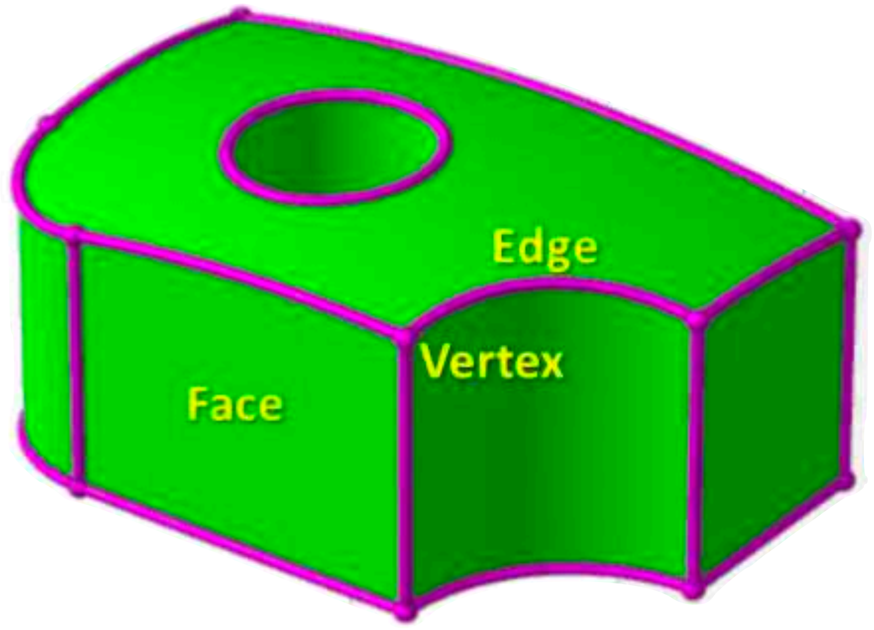
MSc Program: Applied Computational Mechanics

Skoltech

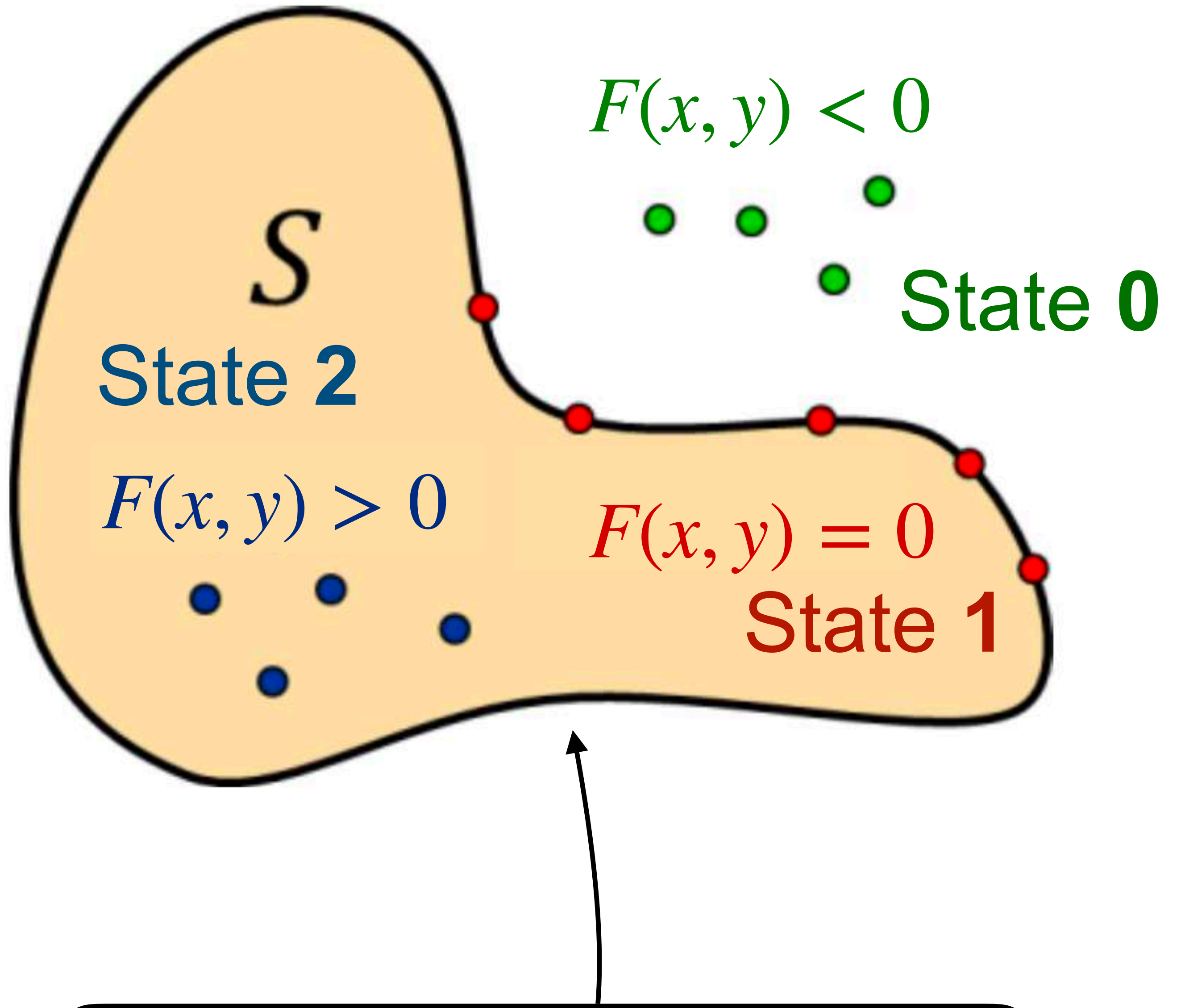
Background

Object representation in computer graphics and design





Boundary Representation (**B-rep**)	Function Representation (**F-rep**) or implicit modeling
external skin using faces, edges, and vertices	**implicit functions** F(x,y,z)
faces are "**glued**" together by topology information describing connectivity	**no explicit** topology - objects can have any topology and change freely
+ straightforward and intuitive approach	- complicated representation even for simple objects
+ efficient local changes	- local changes affect entire function
- high memory requirements	+ low memory footprint
- homogeneous objects	+ heterogenous objects
- possible ill-defined behavior in geometric operations	+ well-defined geometric operations



Only sign is important!

Skoltech

Background

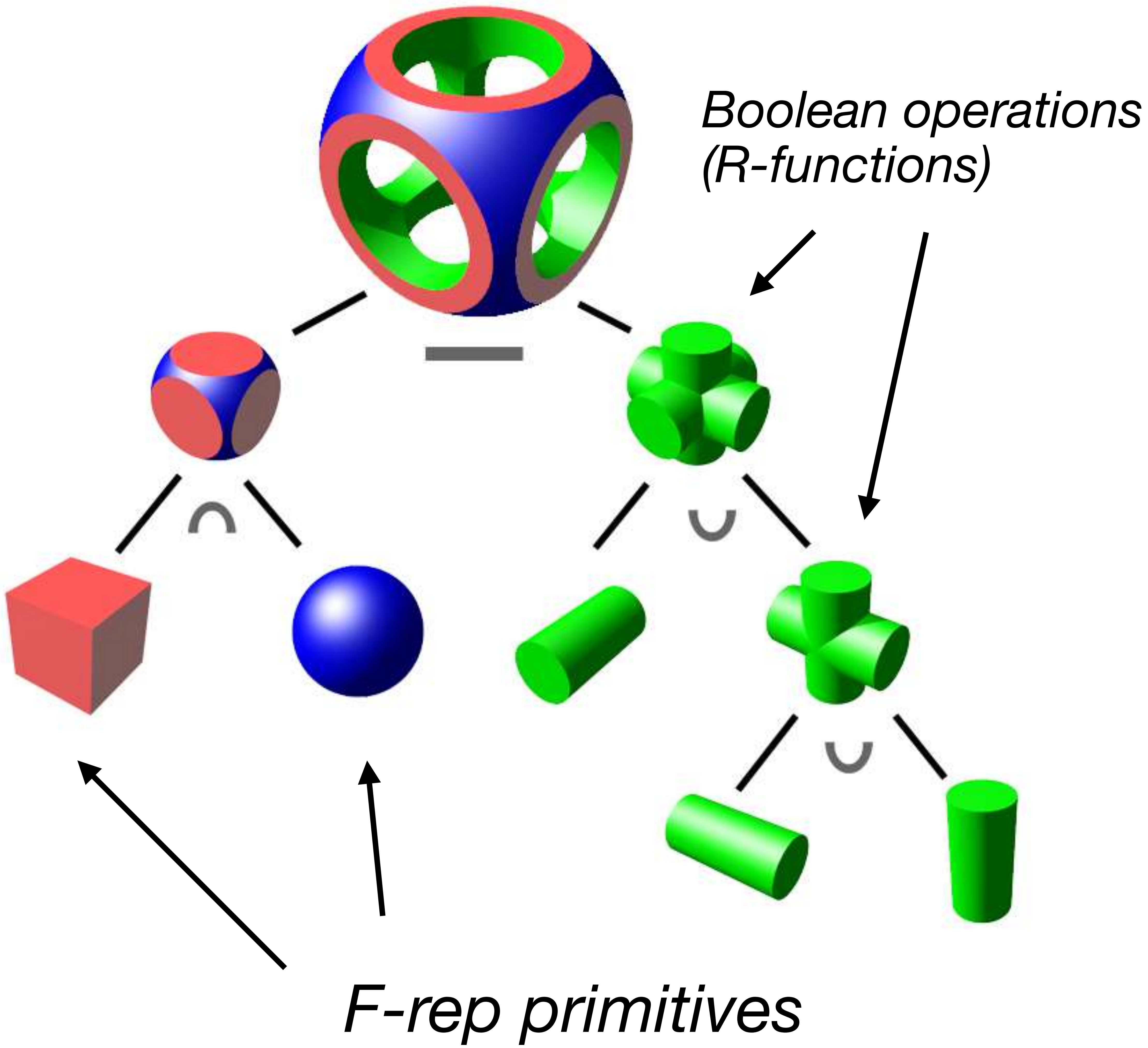
Constructive Solid Geometry (CSG)

Allows a modeler to create a complex object by using *Boolean operators* to combine simpler objects (*primitives*)

Easy to classify arbitrary points as being either inside or outside the shape (*Point Membership Classification*)

Exact Boolean operations on geometric objects

Evaluation complexity: $\mathcal{O}(N)$, where N is the number of nodes in CSG tree

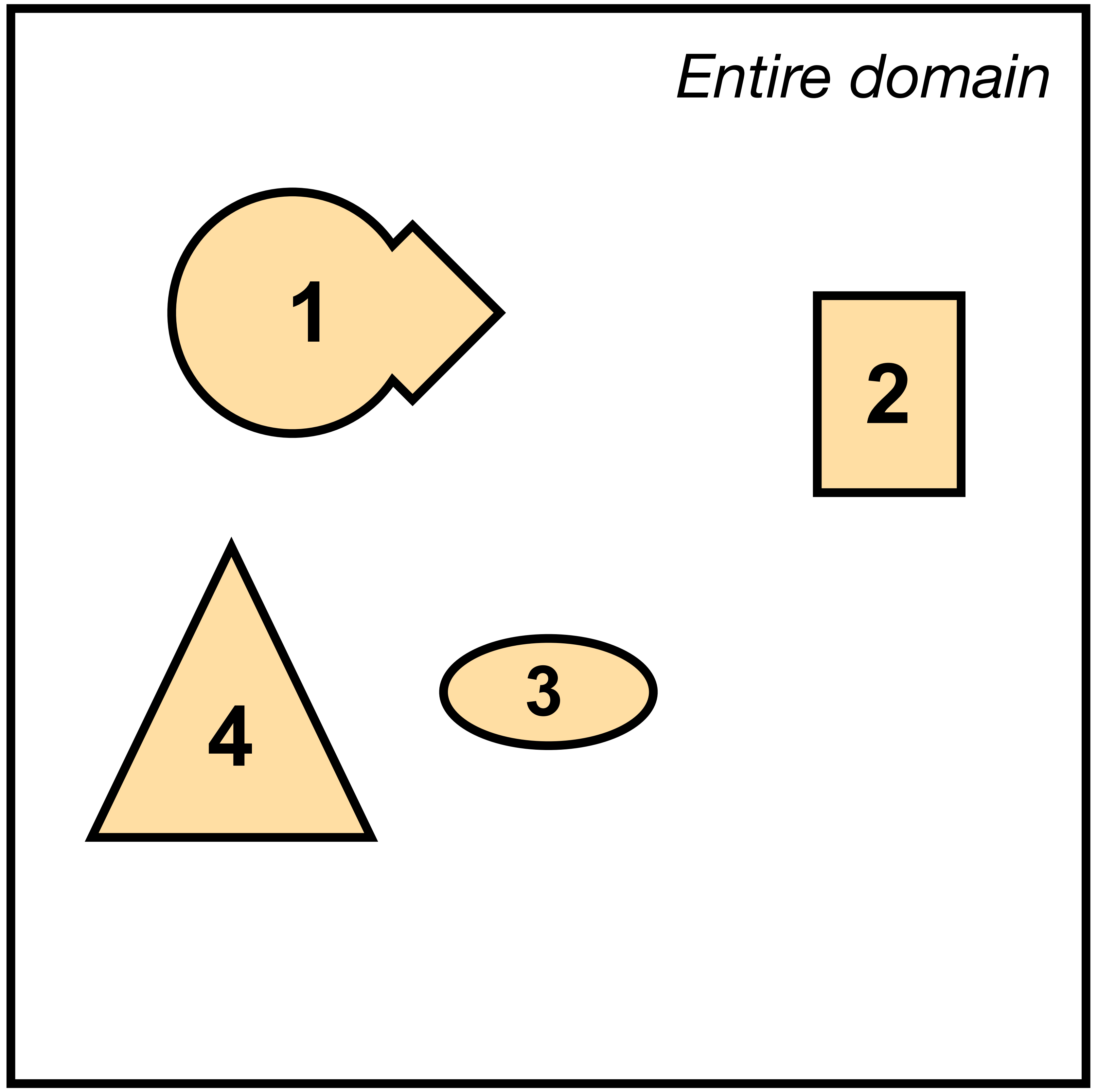


Overall Aim

To develop a comprehensive mathematical framework based on adaptive Constructive Solid Geometry for modeling complex implicitly defined objects

Spatially Adaptive F-rep

Challenges to be addressed: linear complexity of CSG tree



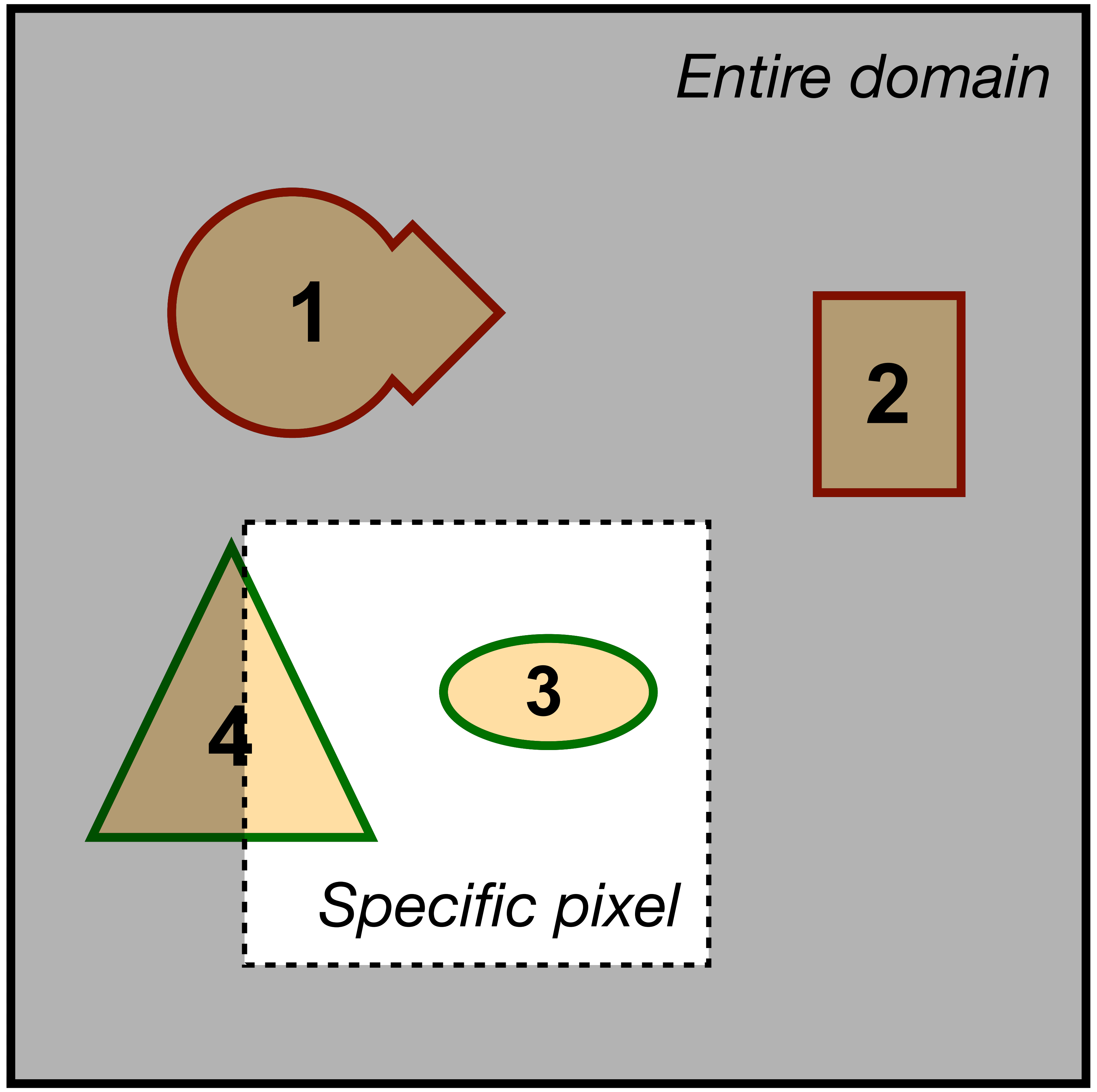
1. **Storage of CSG tree of F-rep primitives**

   Infix and reverse Polish notations (RPN)

Skoltech

Spatially Adaptive F-rep

Challenges to be addressed: linear complexity of CSG tree



1. **Storage of CSG tree of F-rep primitives**

   Infix and reverse Polish notations (RPN)

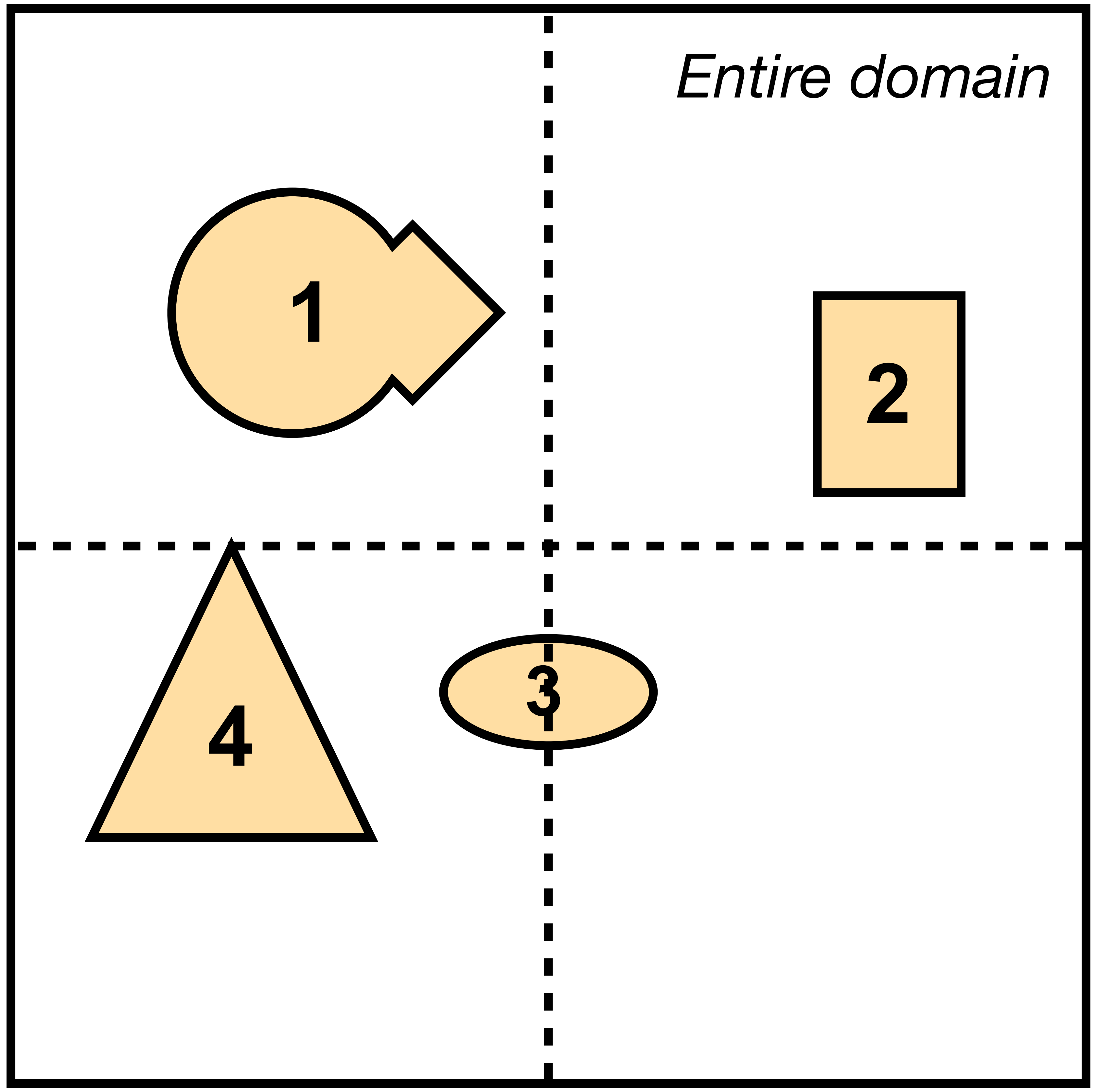
2. **Pruning** (*simplification*) **CSG tree** for specific **voxel / pixel** (*3D / 2D space volume*).

   Range evaluation using interval analysis

Skoltech

Spatially Adaptive F-rep

Challenges to be addressed: linear complexity of CSG tree



1. **Storage of CSG tree of F-rep primitives**

   Infix and reverse Polish notations (RPN)

2. **Pruning** (*simplification*) **CSG tree** for specific **voxel / pixel** (*3D / 2D space volume*).

   Range evaluation using interval analysis

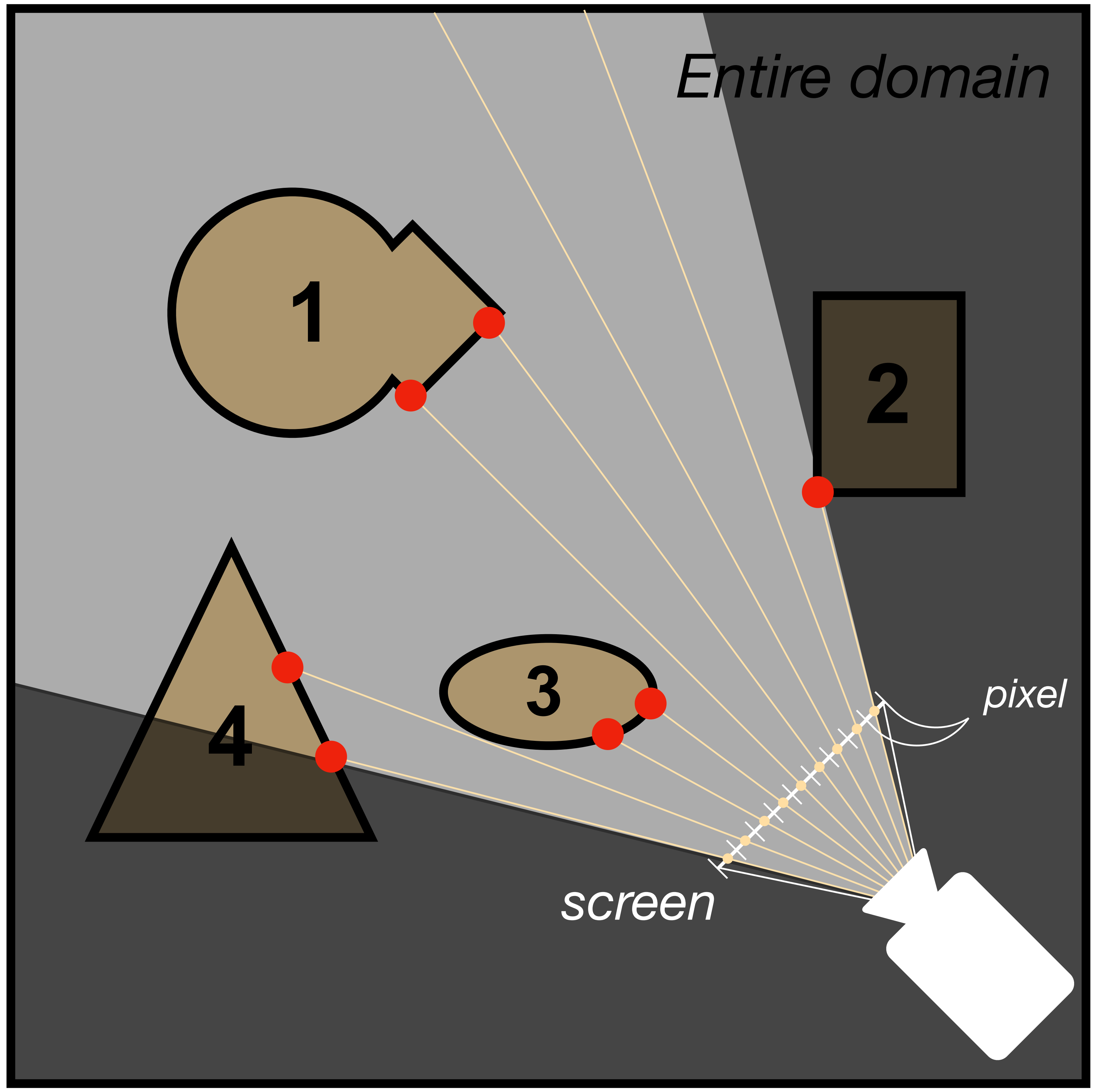
3. **Spatial sampling algorithm**

   Sparse Voxel Octree (SVO)

Skoltech

Spatially Adaptive F-rep

Challenges to be addressed: linear complexity of CSG tree



1. **Storage of CSG tree of F-rep primitives**

   Infix and reverse Polish notations (RPN)

2. **Pruning** (*simplification*) **CSG tree** for specific **voxel / pixel** (*3D / 2D space volume*).

   Range evaluation using interval analysis

3. **Spatial sampling algorithm**

   Sparse Voxel Octree (SVO)

4. **Rendering algorithm for visualization**

   Ray-casting through SVO

Specific Objectives

1. *To design* and *implement* efficient algorithm for pruning complex F-rep scenes within localized spatial regions

2. *To develop* hierarchical data structure and algorithms for optimal storage and evaluation of compressed F-rep scene

3. *To design* and *implement* adaptive spatial sampling algorithm enabling constant evaluation complexity using Sparse Voxel Octree

4. *To develop* a robust ray-casting algorithm for rendering F-rep objects

5. *To conduct* performance evaluation of the proposed methodology

Composite F-rep as CSG tree

Data structures to store objects in F-rep

Infix notation (expression tree)

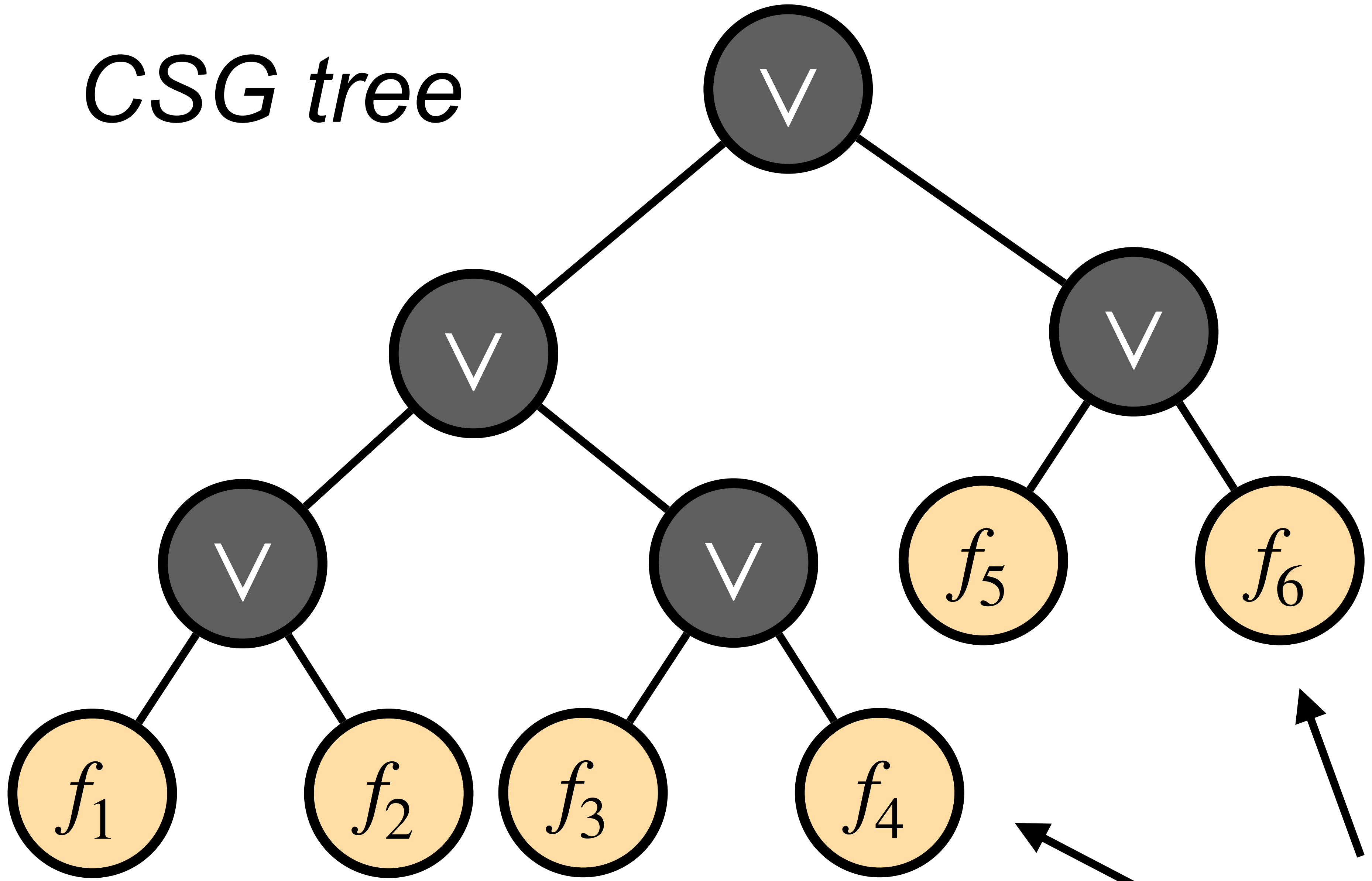
$$((f_1 \vee f_2) \vee (f_3 \vee f_4)) \vee (f_5 \vee f_6)$$

Shunting-yard algorithm

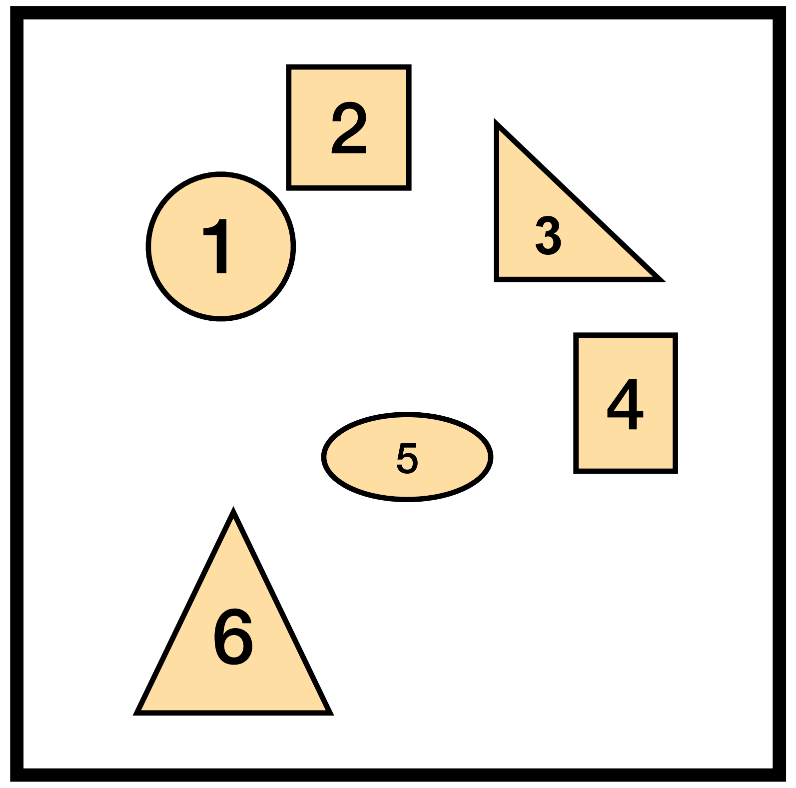
Reverse Polish notation (RPN)

$$f_1 f_2 \vee f_3 f_4 \vee \vee f_5 f_6 \vee \vee$$

CSG tree



F-rep primitives



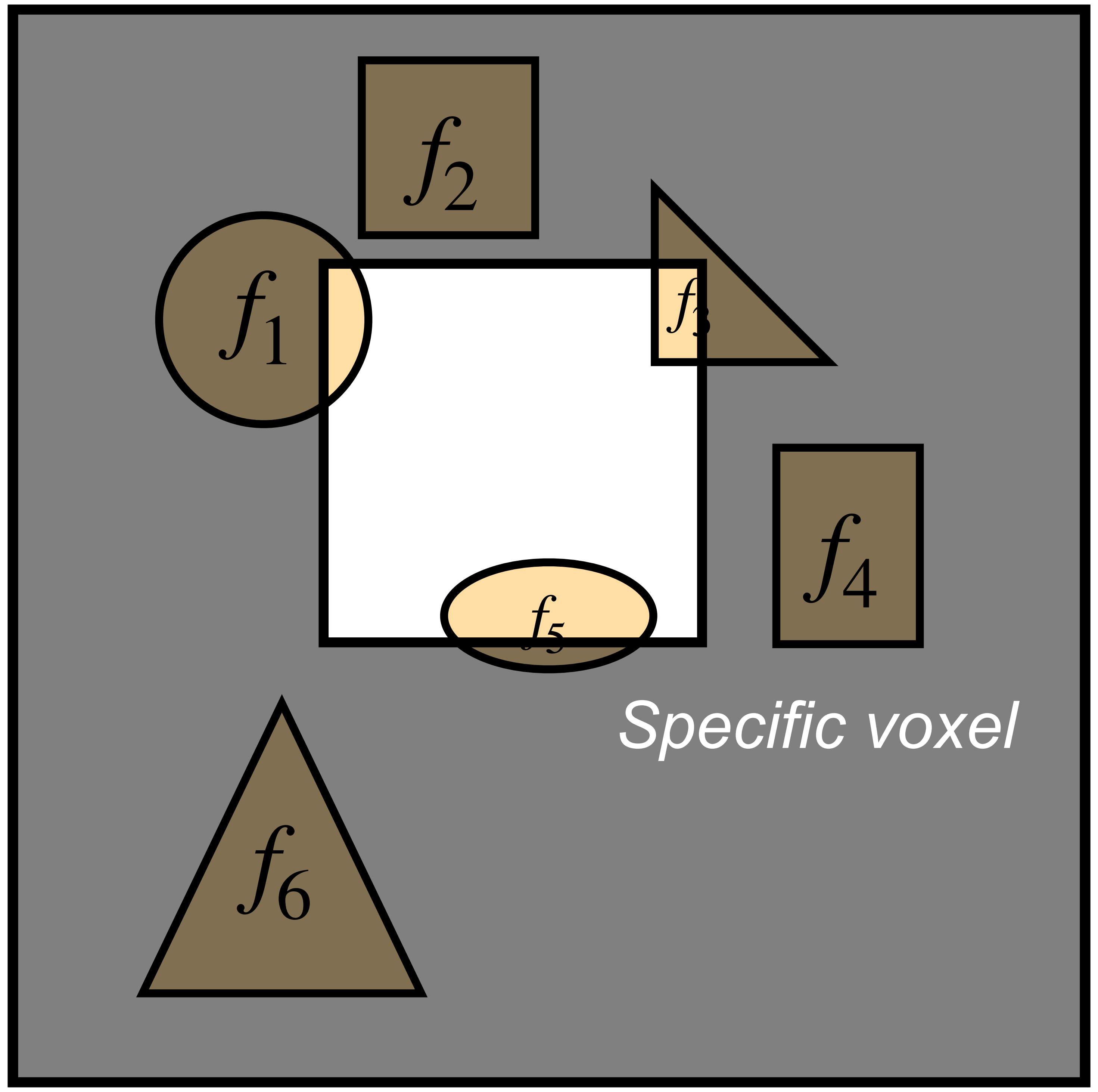
Linearized CSG tree

Array with $O(1)$ access to elements

Pruning linearized CSG tree

Pruning of specific voxel domain

f_l	f_r	$f_l \vee f_r$	$f_l \wedge f_r$
+	+, -, ±	f_l	f_r
+, -, ±	+	f_r	f_l
-	+, -, ±	f_r	f_l
+, -, ±	-	f_l	f_r



For each node i in the linearized CSG tree:

1. **If node i is an F-rep primitive**

   Compute its F-rep interval using interval-analysis (IA) techniques (Pasko)

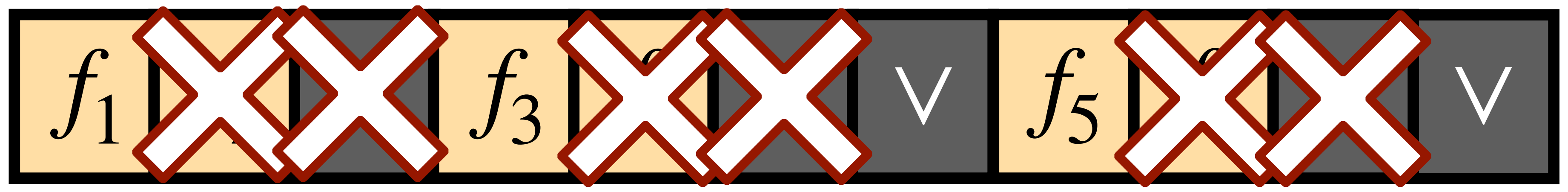
   Store the sign of resulting interval: $+$, $-$, or $\pm$

2. **If node i is a Boolean operator**

   Check whether the node can be pruned (see *Table*).

   If it *can* be pruned, recursively prune its subtree.

   If it *cannot* be pruned, compute its interval with IA and store its sign.

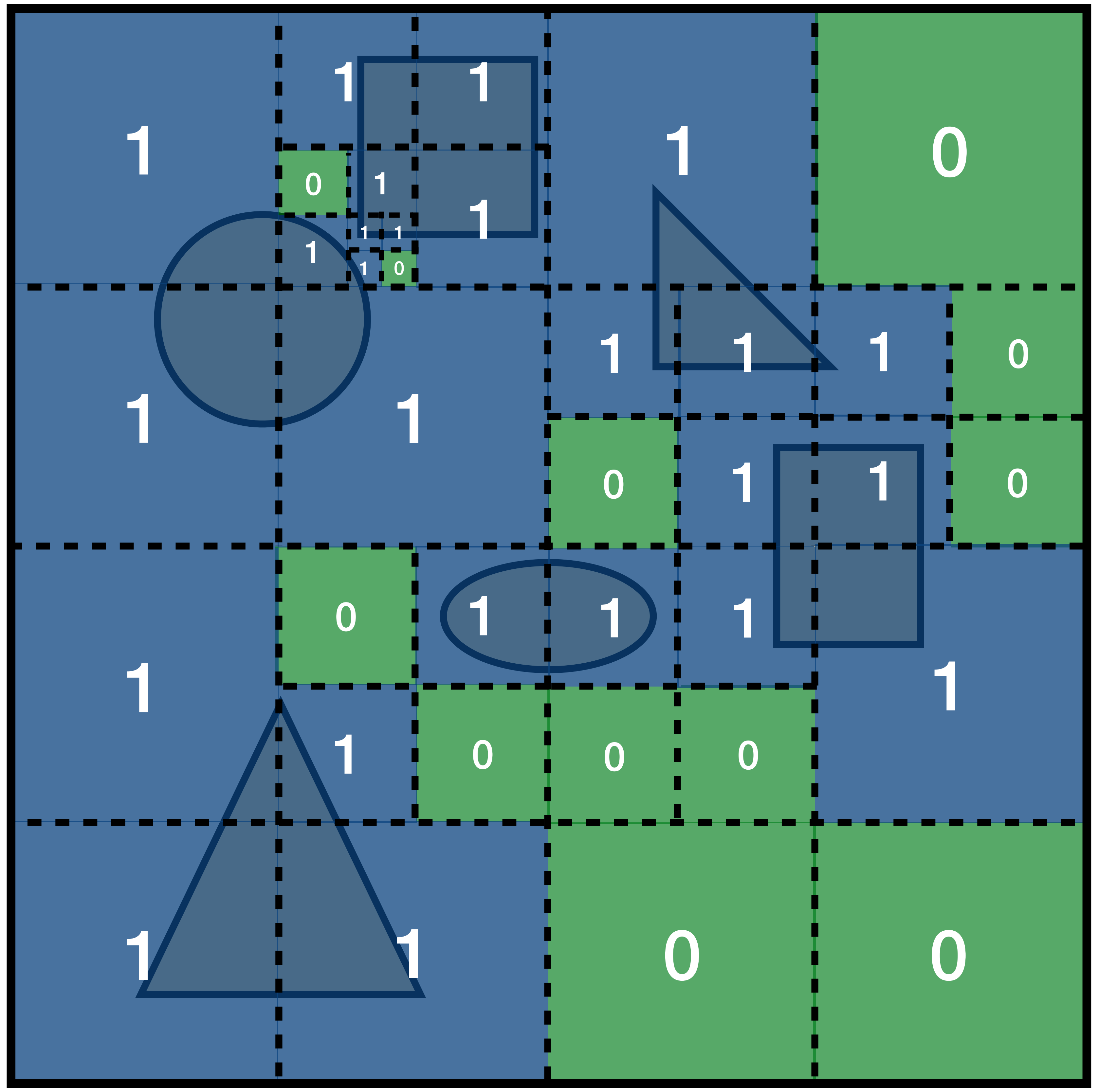


Result: A pruned F-rep represented as a bit-mask, with **1** for *active* nodes and **0** for *pruned* nodes.

bit-mask	1	0	0	1	0	0	1	1	0	0	1

Adaptive Spatial Decomposition

Sparse Voxel Octree (SVO)



Entire domain

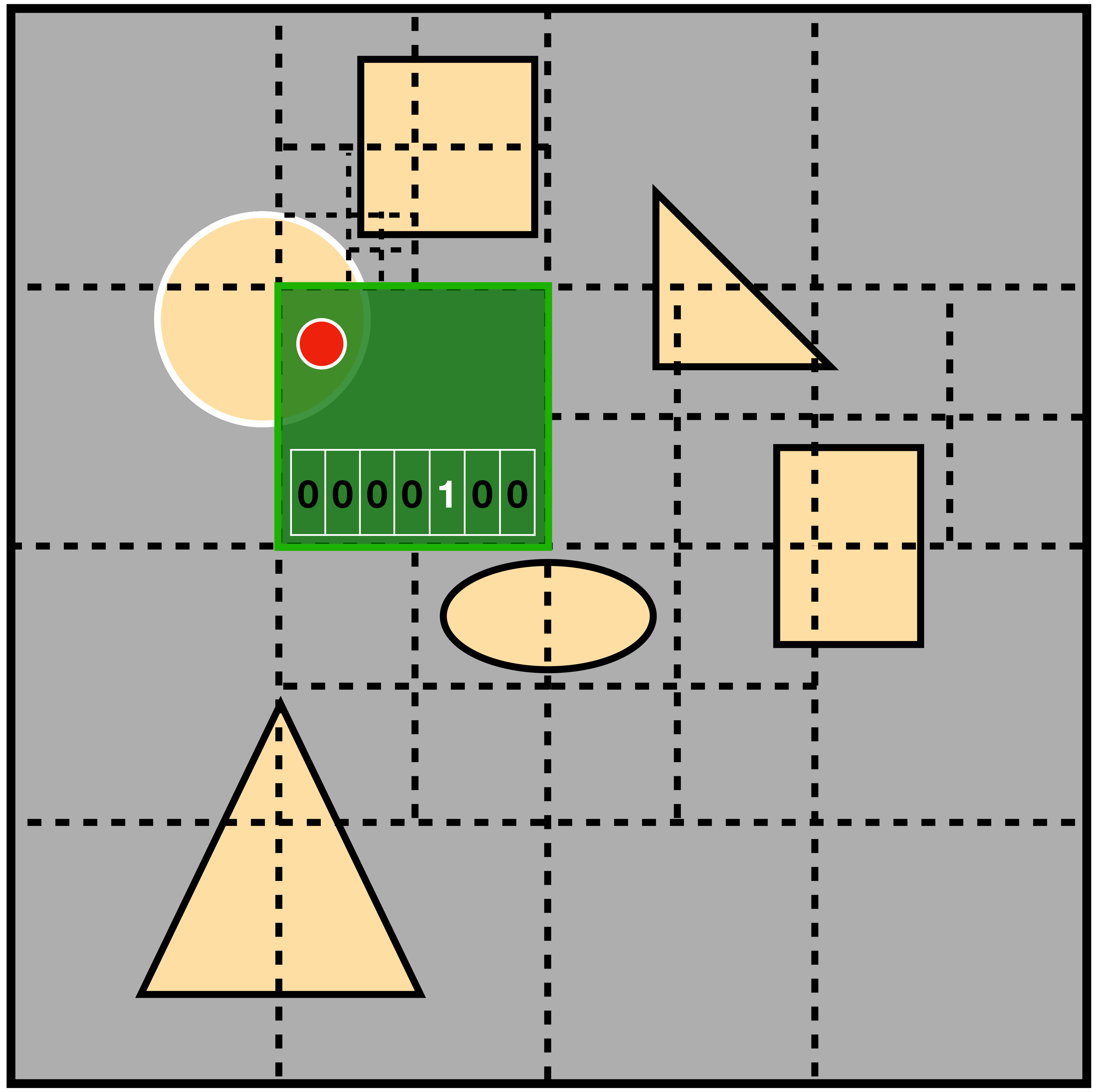
Implementation: octree structure

Subdivision process:

1. Prune composite F-rep (CSG tree) for region
2. Count active CSG tree nodes
3. Compare with threshold (e.g., 2)
4. Subdivide if needed ($n = 8$ for octree, $n = 4$ for quadtree)
5. Recursively prune for subregions

Result: voxel octree with pruned F-rep in each node (*empty* voxels - **green**, *filled* voxels - **blue**)

Complexity of spatially adaptive F-rep



Entire domain

Evaluation process:

1. Choose a point in the geometry domain
2. Determine SVO leaf node that the point belongs to

   Complexity: $\mathcal{O}(K(L))$, where K is the level of SVO leaf node, and L is the threshold value

3. Get bit mask (compressed F-rep) from this node

   Complexity: $\mathcal{O}(1)$

4. Evaluate pruned F-rep expression

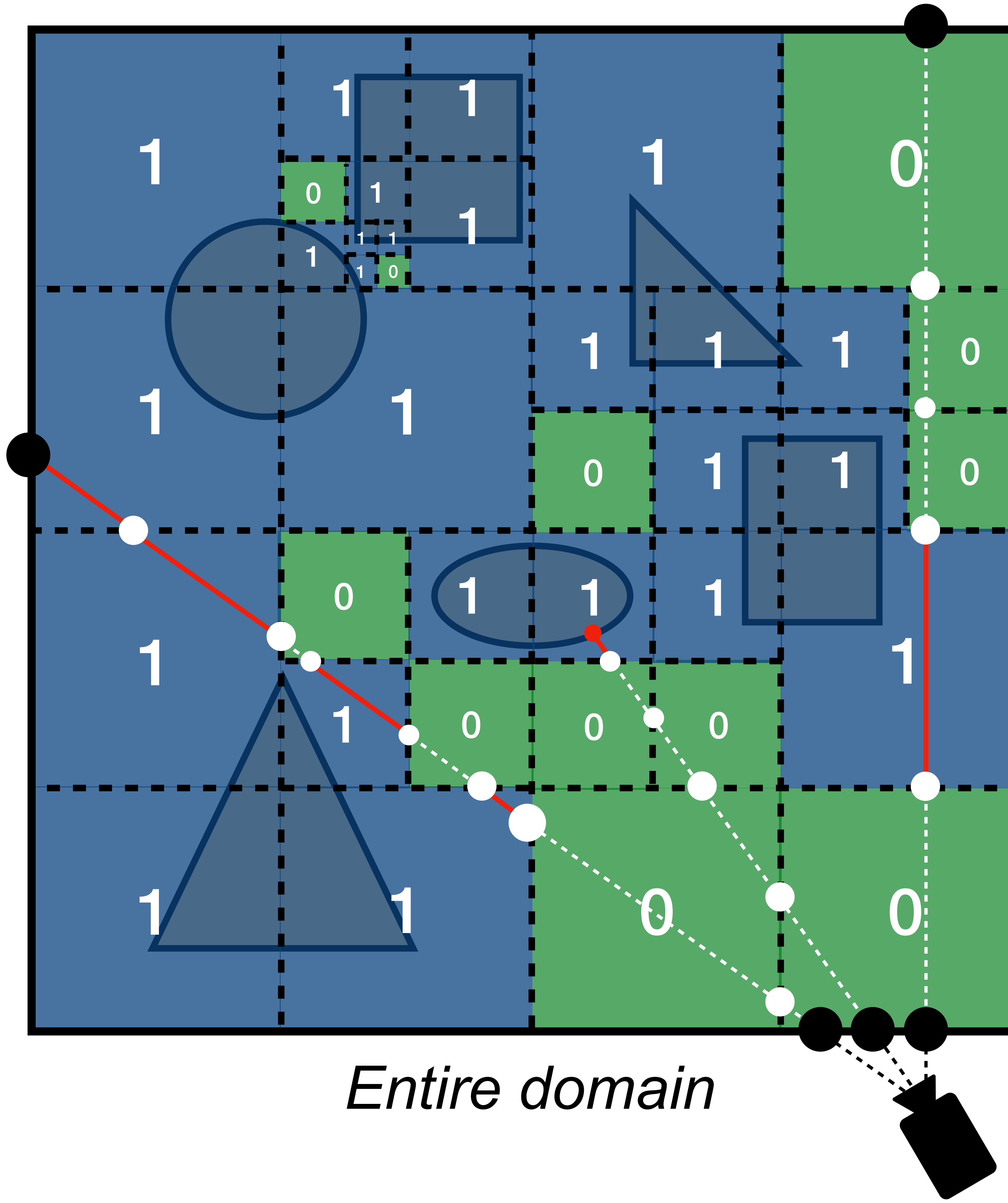
   Complexity: $\mathcal{O}(L)$, where L is the threshold value

Total complexity: $C_1 \cdot K(L) + C_2 \cdot L + C_3$

In practice, for large L, $C_2 \cdot L \gg C_1 \cdot K(L) \implies$ total complexity is **constant,** controlled by threshold value

Rendering spatially adaptive F-rep

Ray-traversal through SVO



Entire domain

Implementation: ray-casting through voxel octree

Ray-traversal process:

1. Shoot a ray, specifying its origin and direction
2. Determine the voxels intersected by the ray
3. If a voxel is **empty**, skip it (*dotted line section*)
4. If a voxel is **filled**, attempt to find an intersection with the pruned F-rep (*red line section*)
5. If no intersection is found, proceed to the next voxel

Skoltech

Results

Developed core mathematical framework for an adaptive CSG kernel in C++. Specifically:

1. *Developed* a memory-efficient storage of CSG tree in reverse Polish notation, employing bitmasks.

2. *Designed* an algorithm that recursively prunes the CSG tree and builds a Sparse Voxel Octree (SVO) with a compressed F-rep in every node.

3. *Developed* robust ray-casting through SVO, based on an interval bisection method.

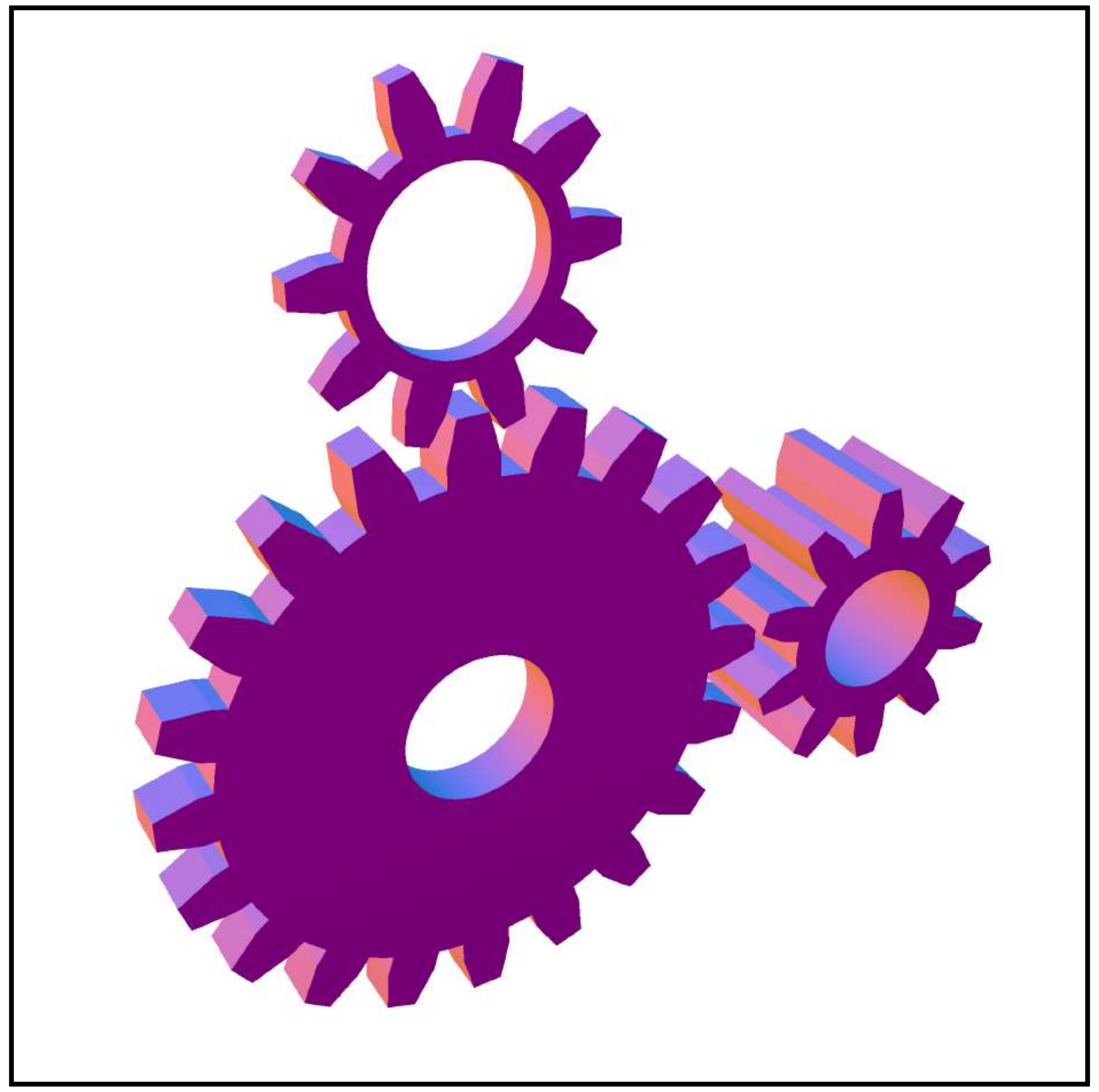
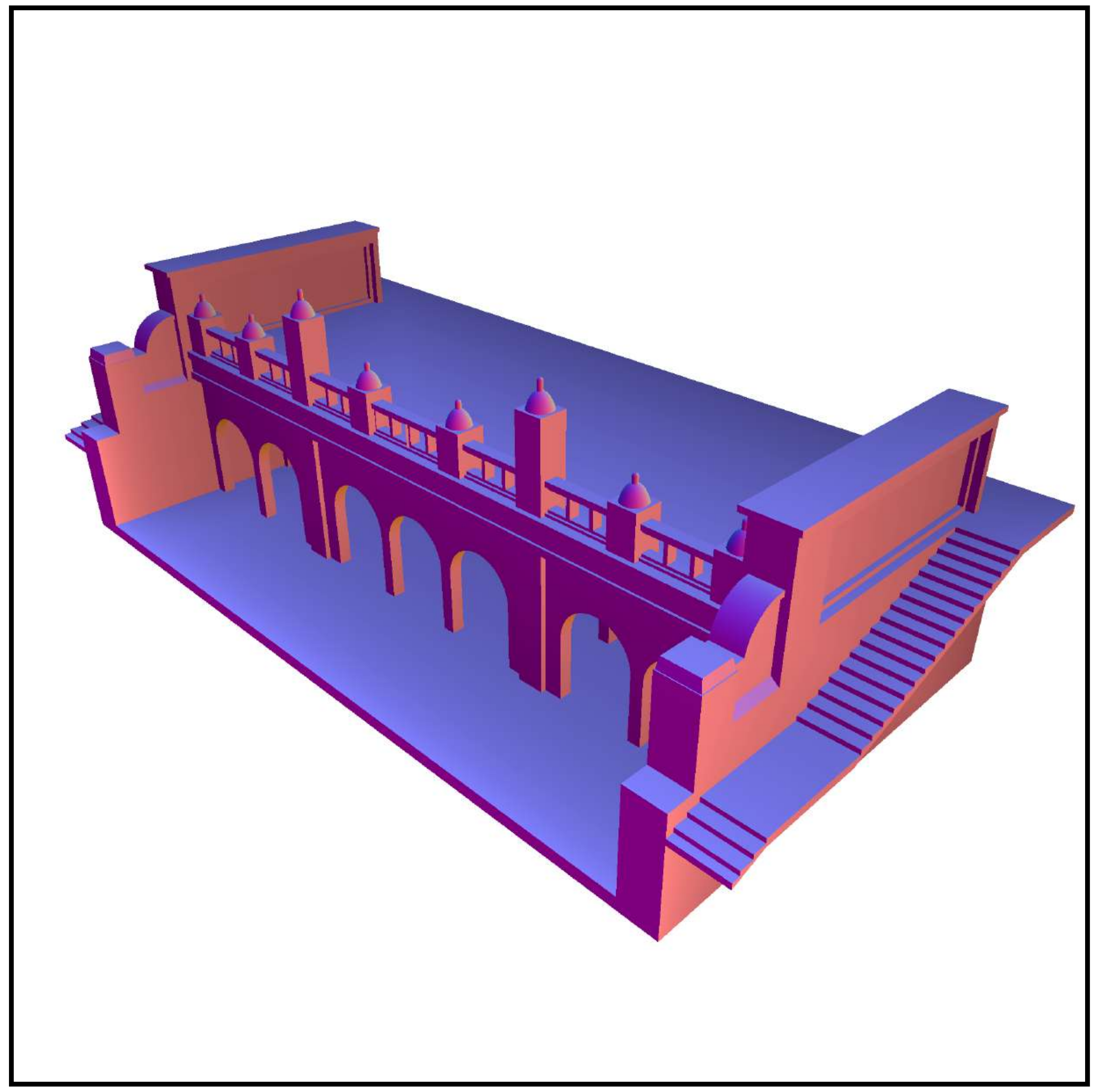
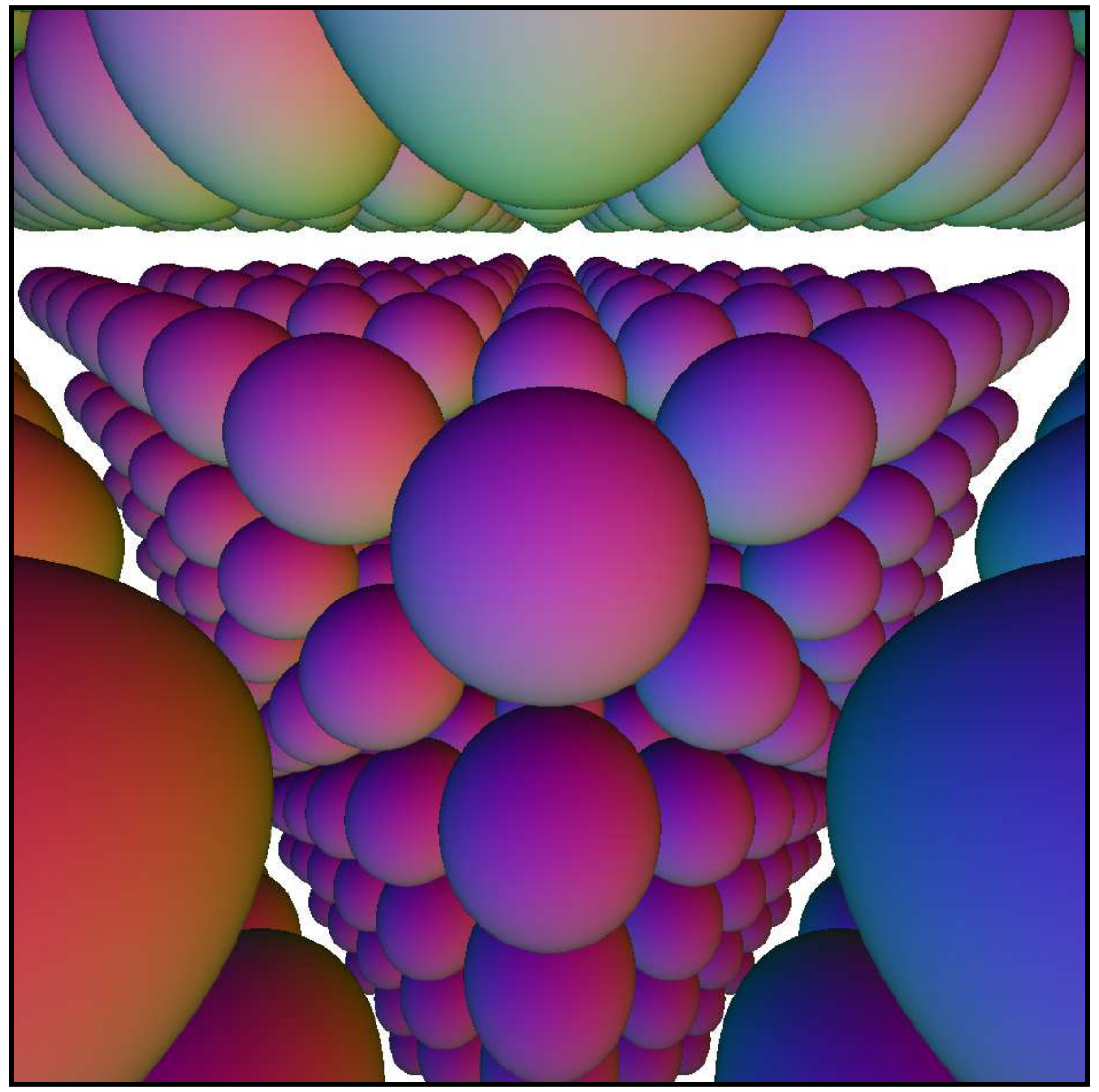
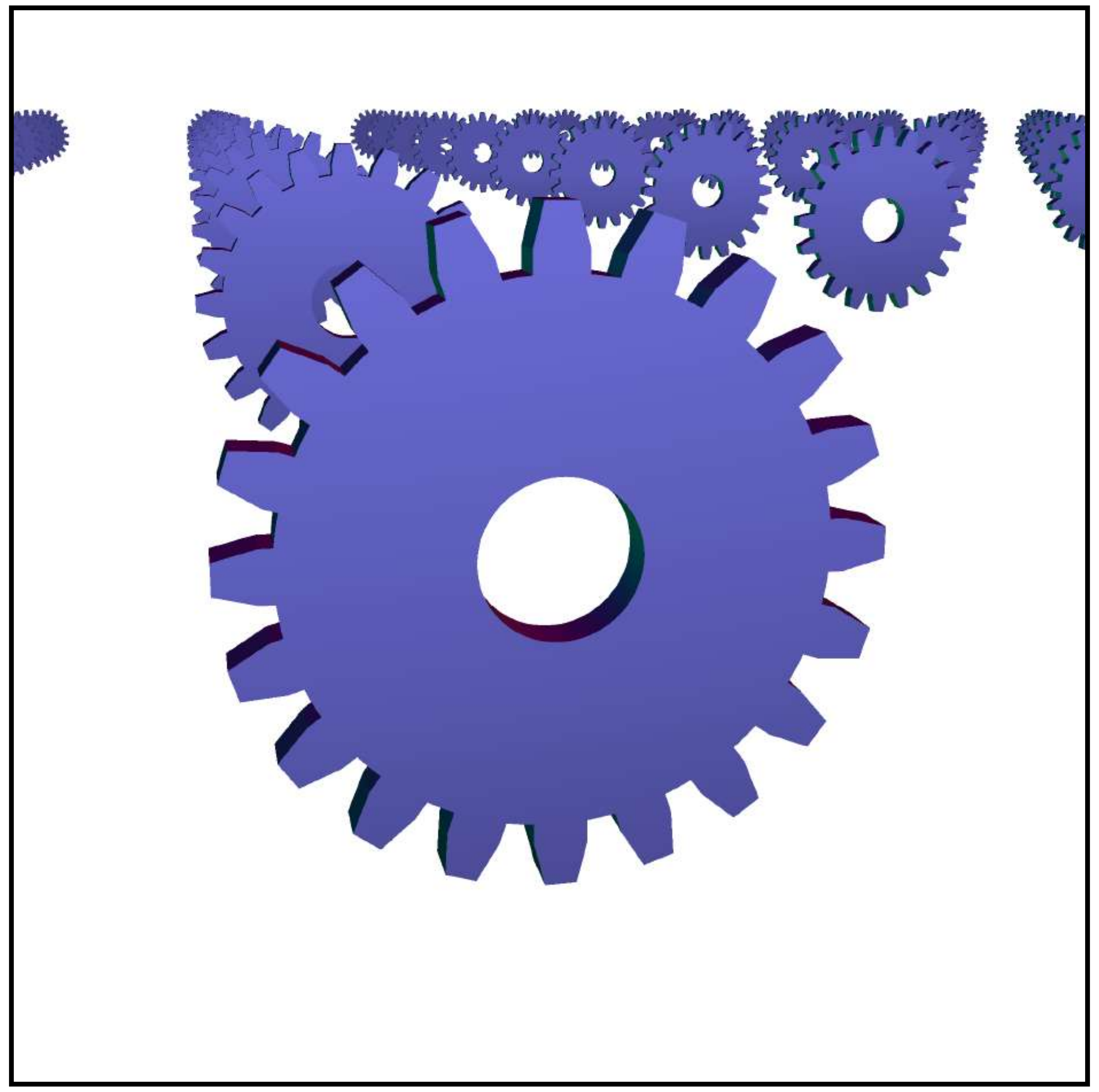
4. *Implemented* all algorithms in a prototype geometry kernel written entirely in C++.

5. *Compared* the proposed methodology with state-of-the-art approaches.

Skoltech

Results

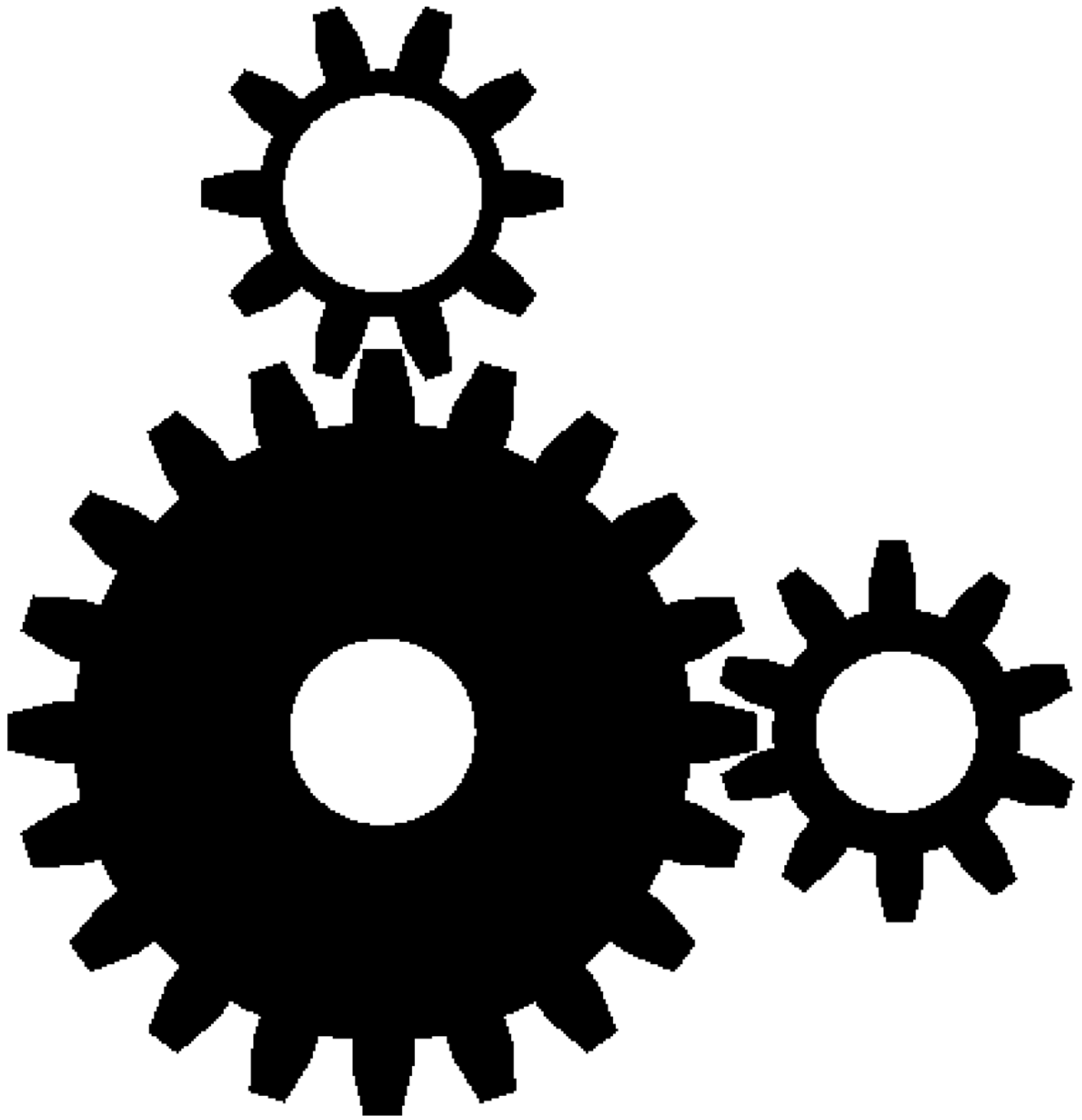
Complex F-rep 3D scenes for benchmarking framework

Gears	Architecture	Many Spheres	Many Gears
			
RPN length = **506**	RPN length = **1431**	RPN length = **5489**	RPN length = **30377**

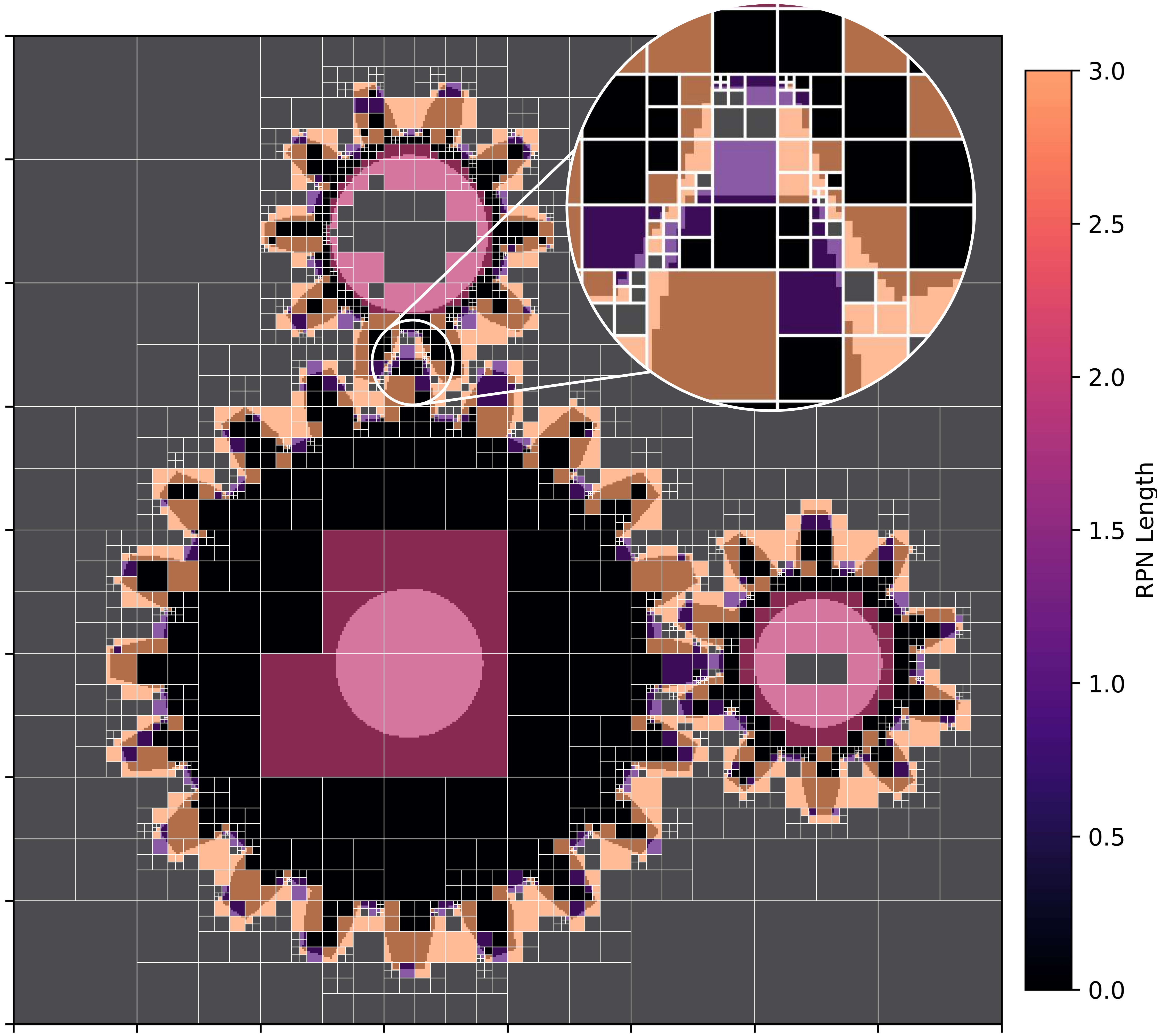
Skoltech

Results

SVO construction for 2D case



Original RPN length = **502**



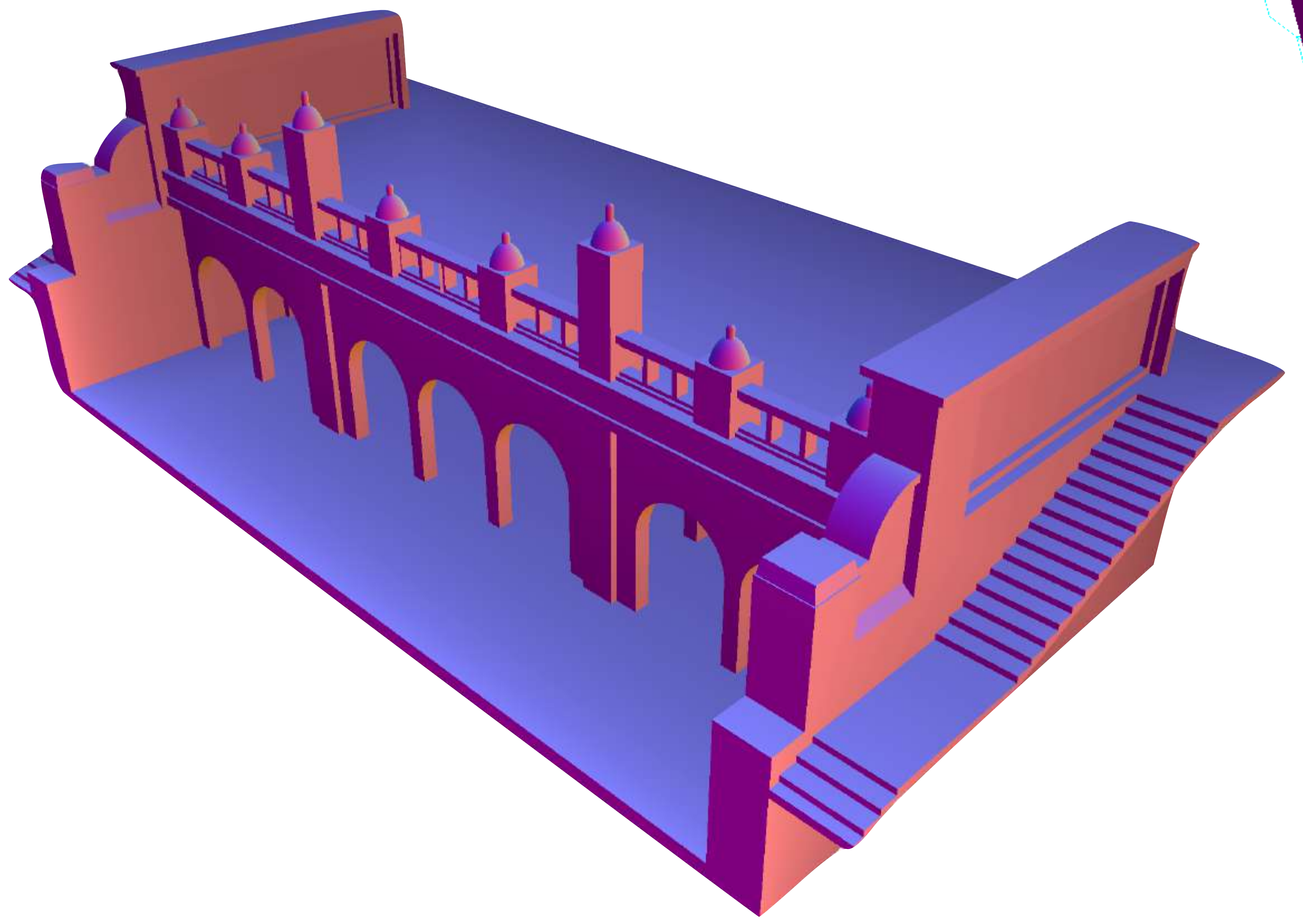
Max octree level = **10**

Threshold (max RPN length) = **3**

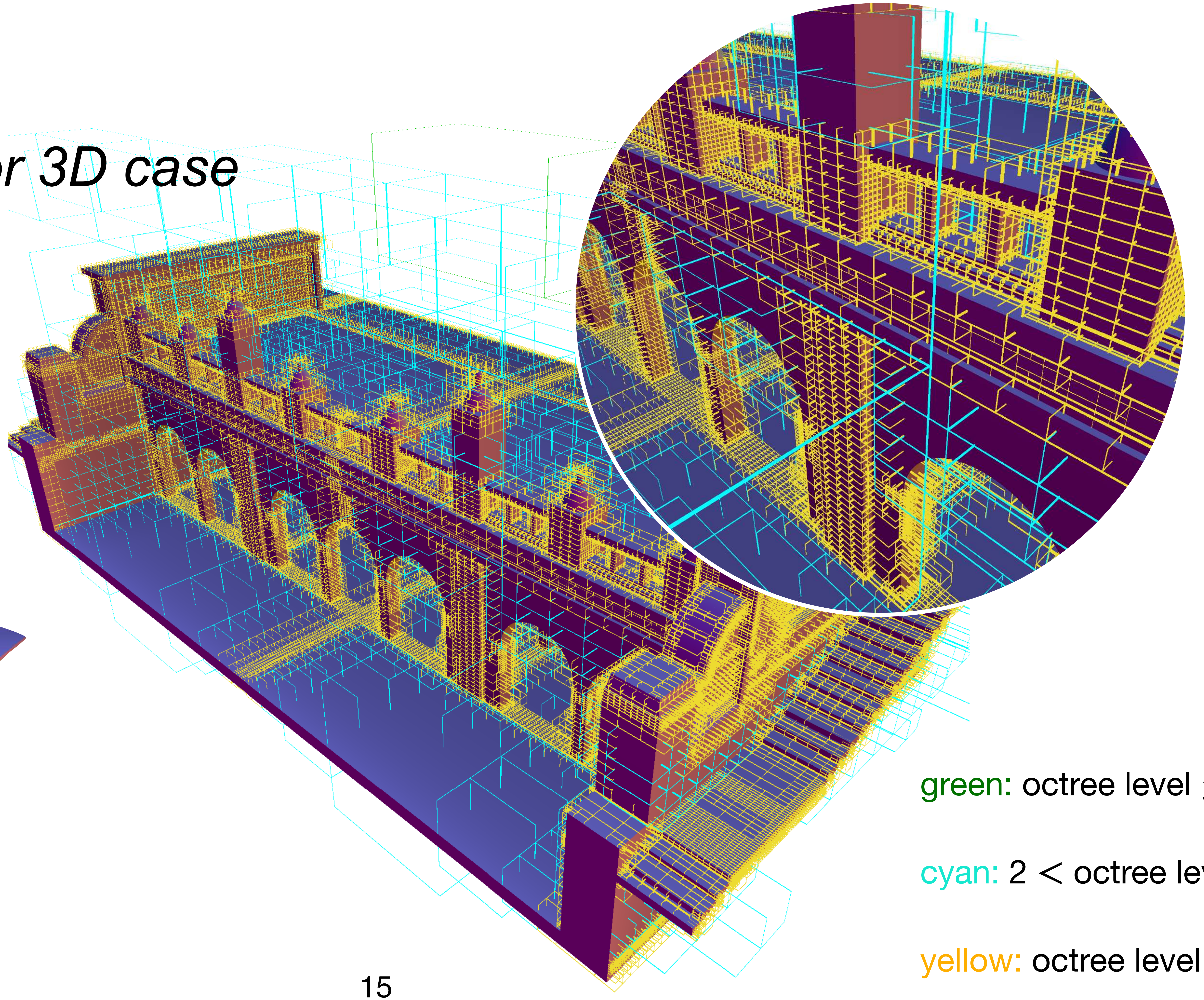
Skoltech

Results

SVO construction for 3D case



Original RPN length = **1431**



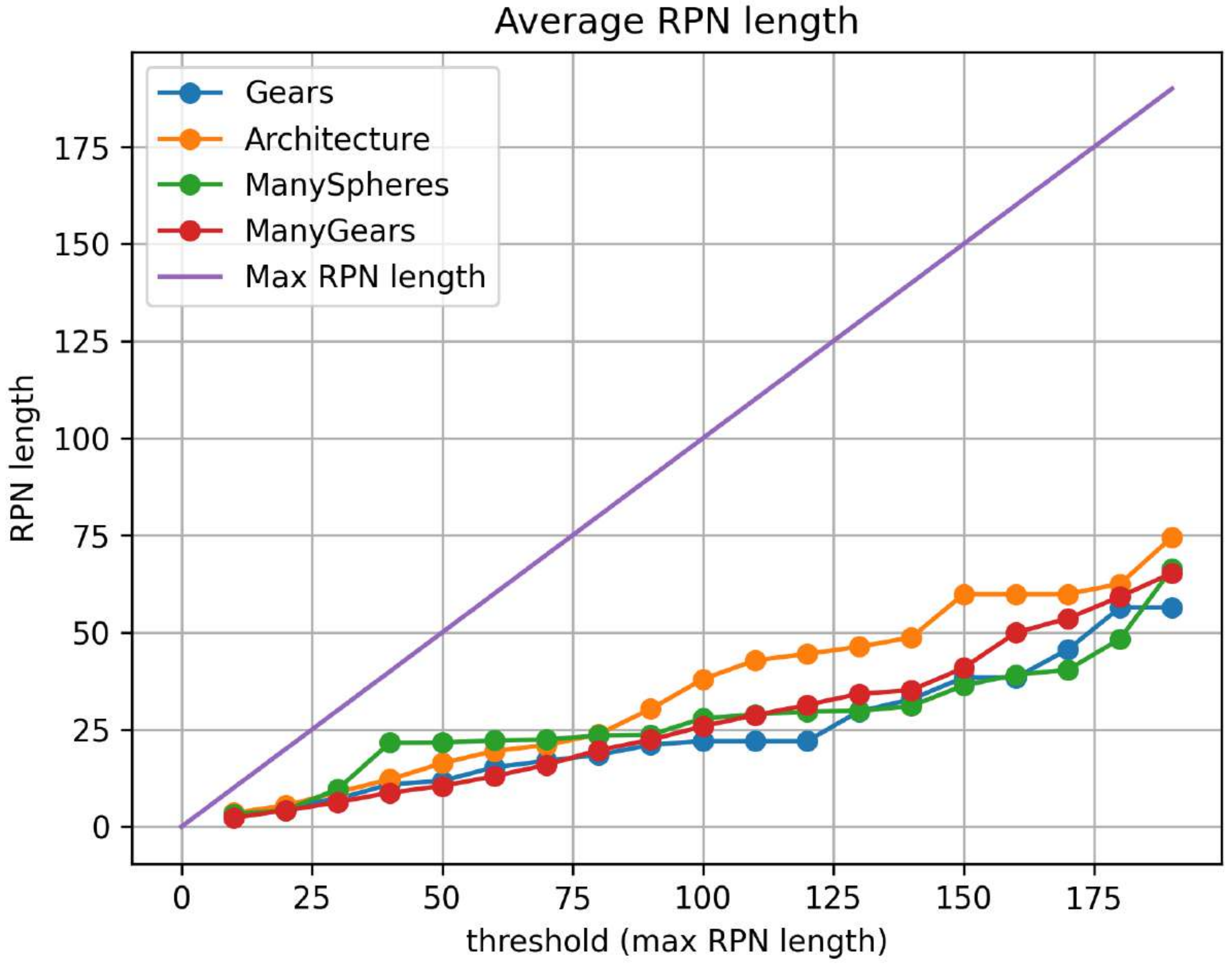
green: octree level ≤ 2

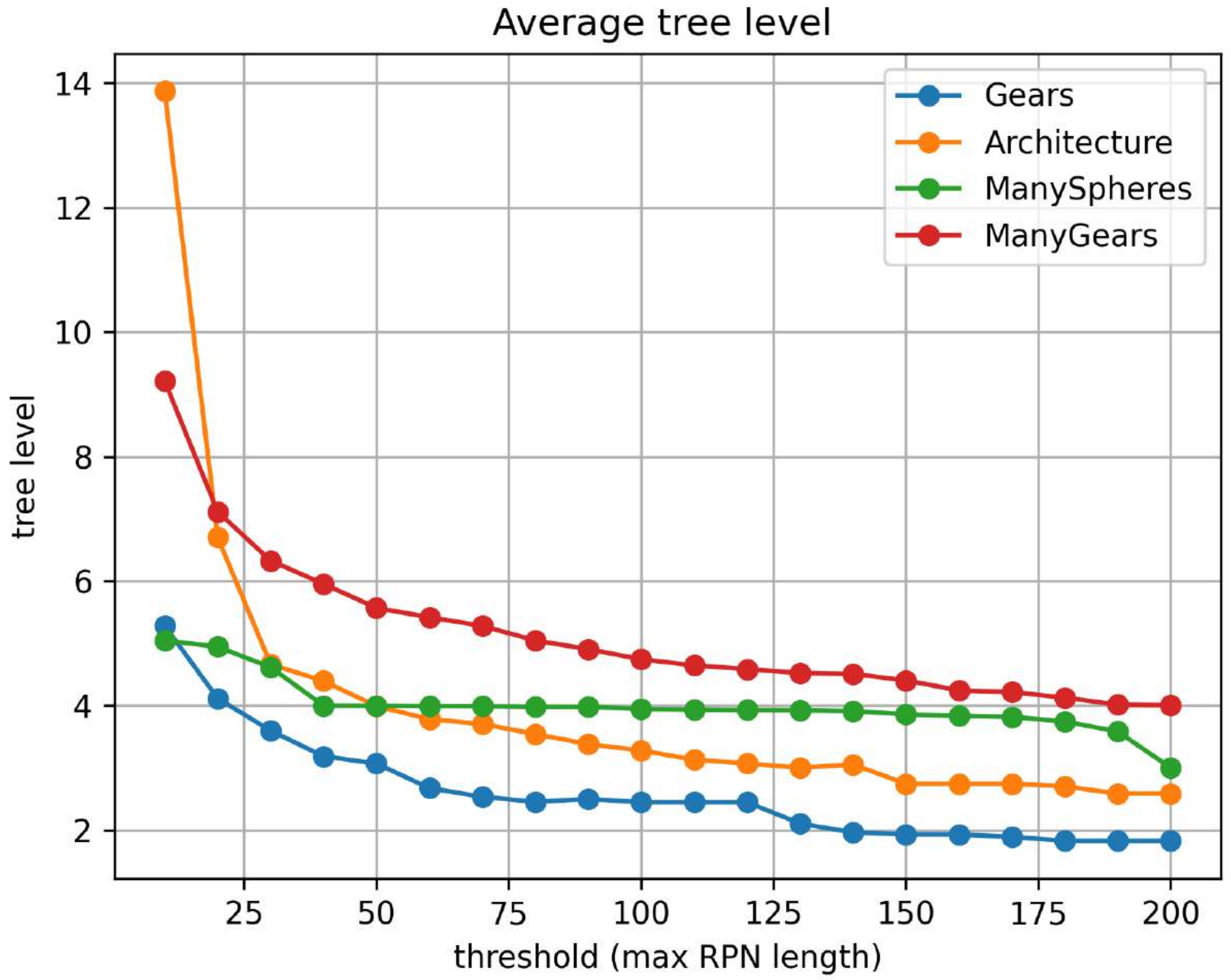
cyan: $2 <$ octree level ≤ 5

yellow: octree level > 5

Results

Pruning results

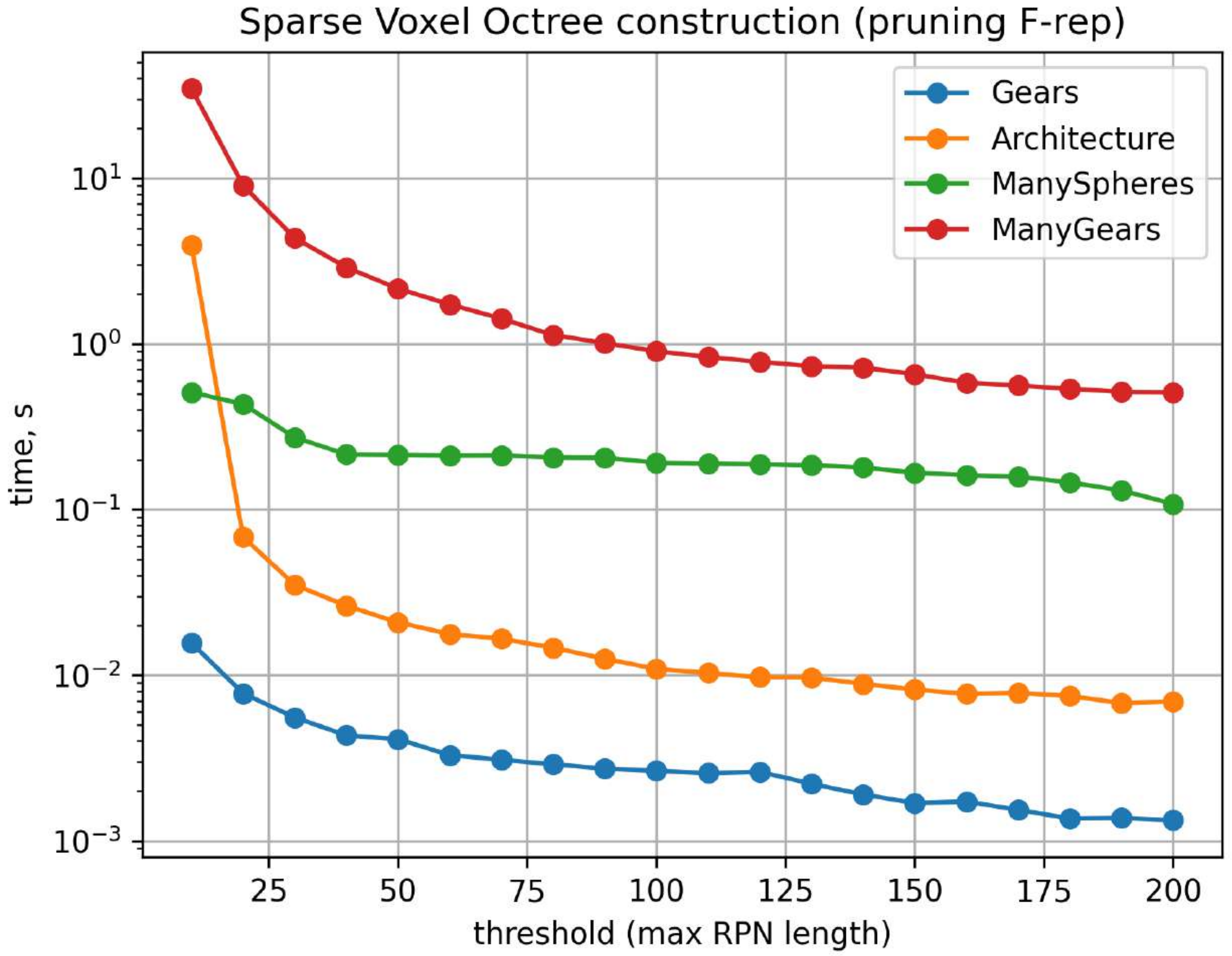




Average RPN length is far below the threshold value because many SVO nodes are empty

For **small thresholds**, the average tree level rises sharply.

Pruning time grows exponentially as the threshold decreases: deeper trees mean more work, so run-time balloons for low thresholds.



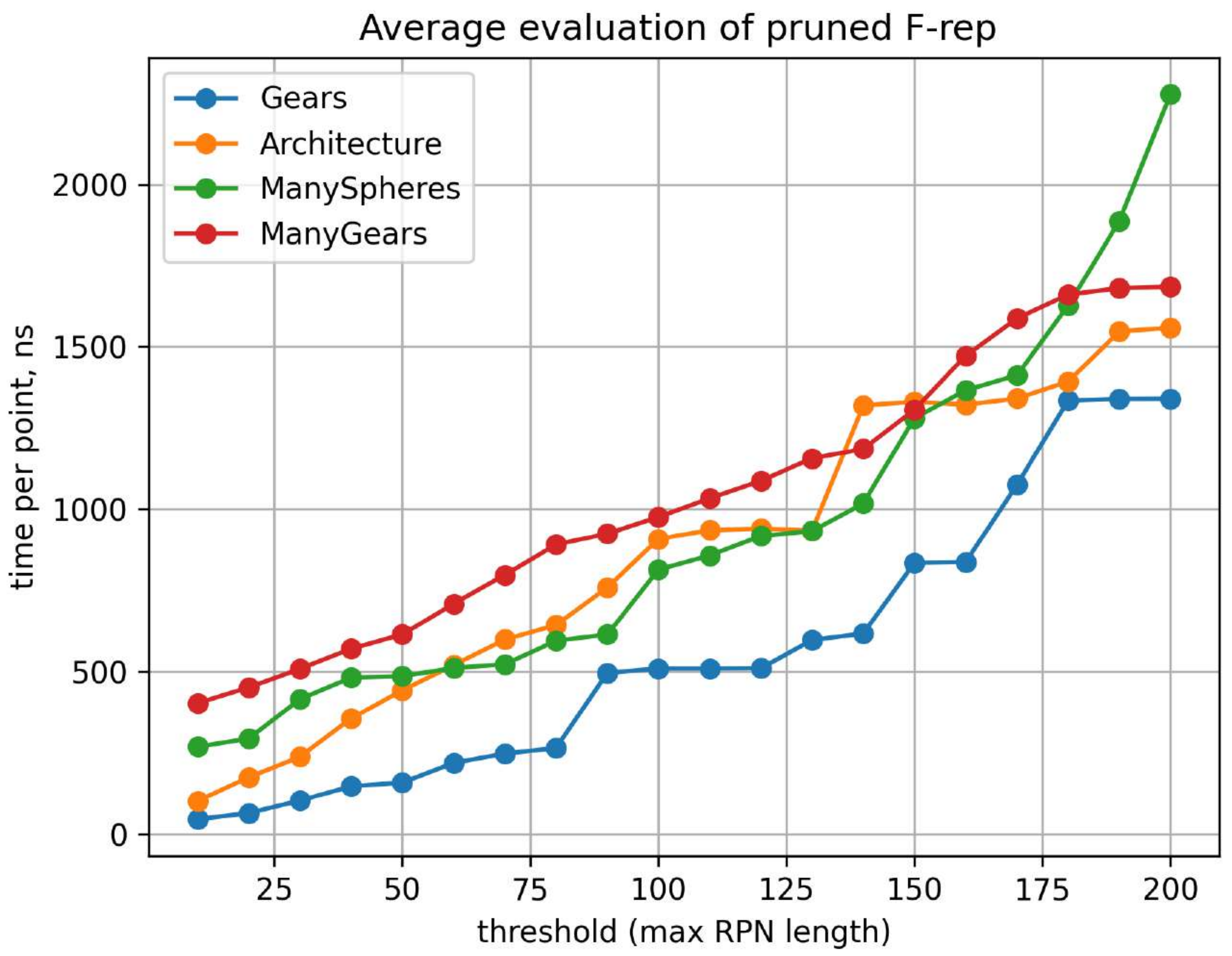
Skoltech

Results

Evaluation speedup

Scene	Time per point, ns
Gears	6'000
Architecture	20'000
ManySpheres	75'000
ManyGears	409'000

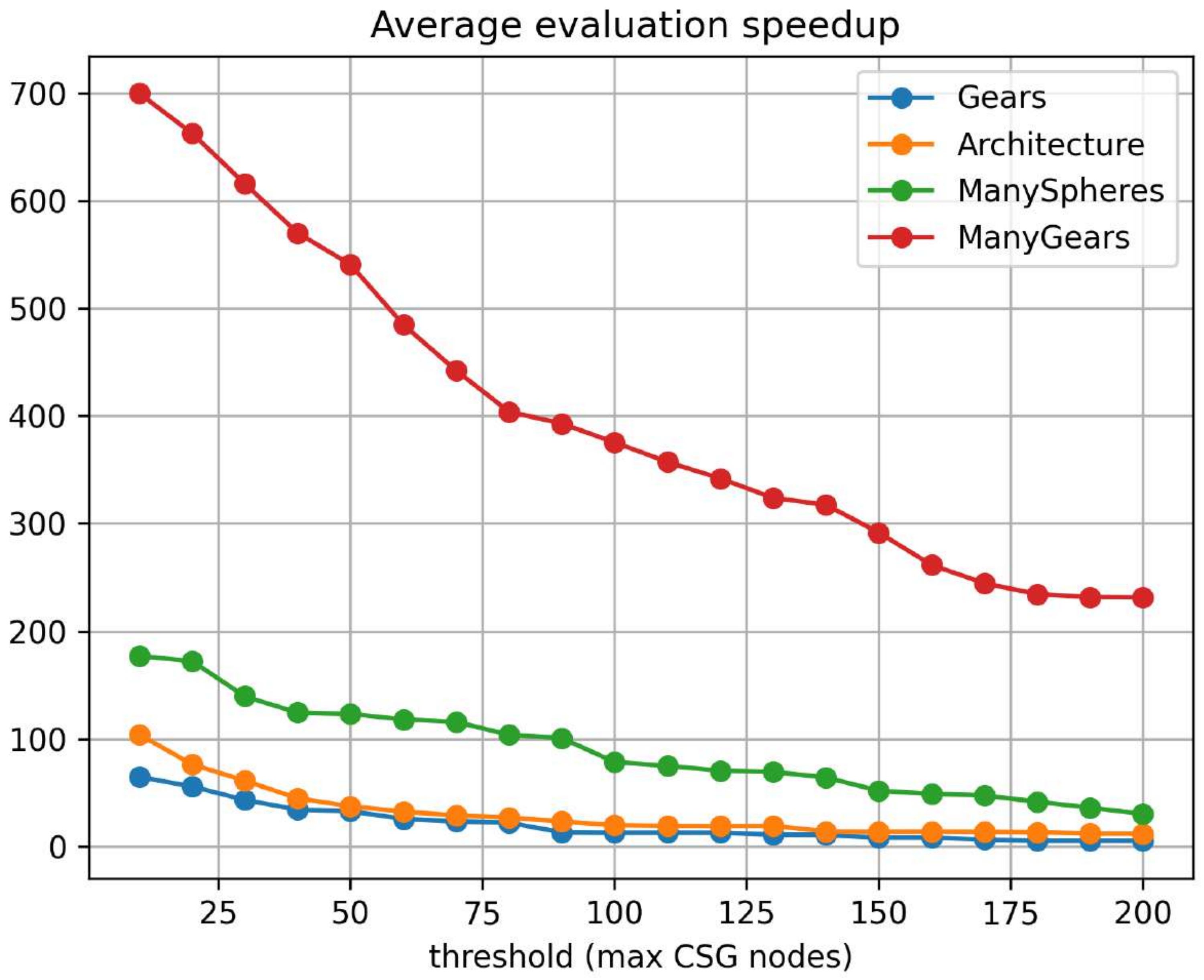
Average evaluation of original (unpruned) F-rep

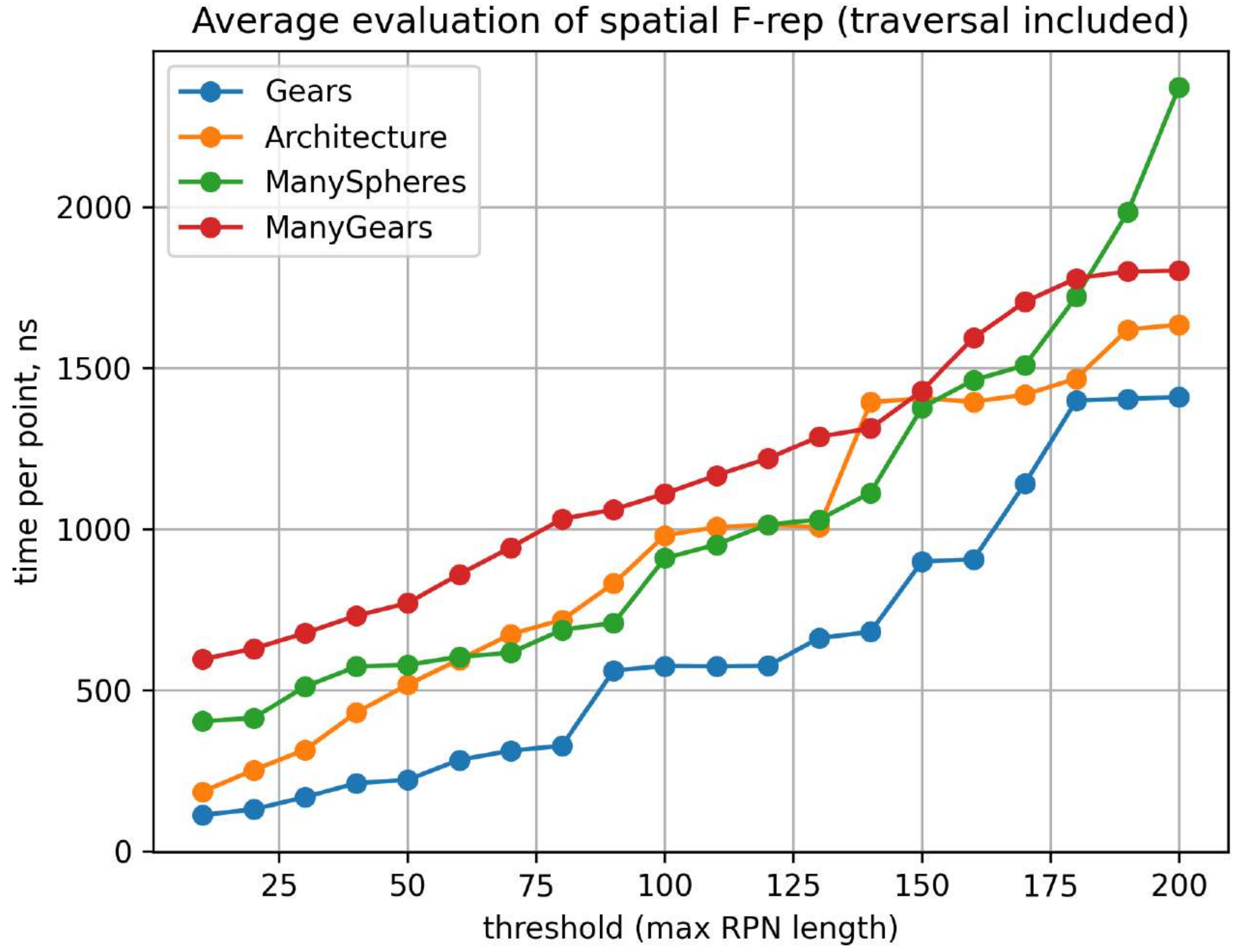


Evaluation of all benchmarking 3D scenes (SVO traversal is excluded) has **identical** complexity, despite the initial difference in model complexity.

The *ManyGears* scene achieves the greatest speedup, ranging from **200x** to **700x**

Even the simplest test case, *Gears*, attains a performance improvement of **4x** to **50x**.



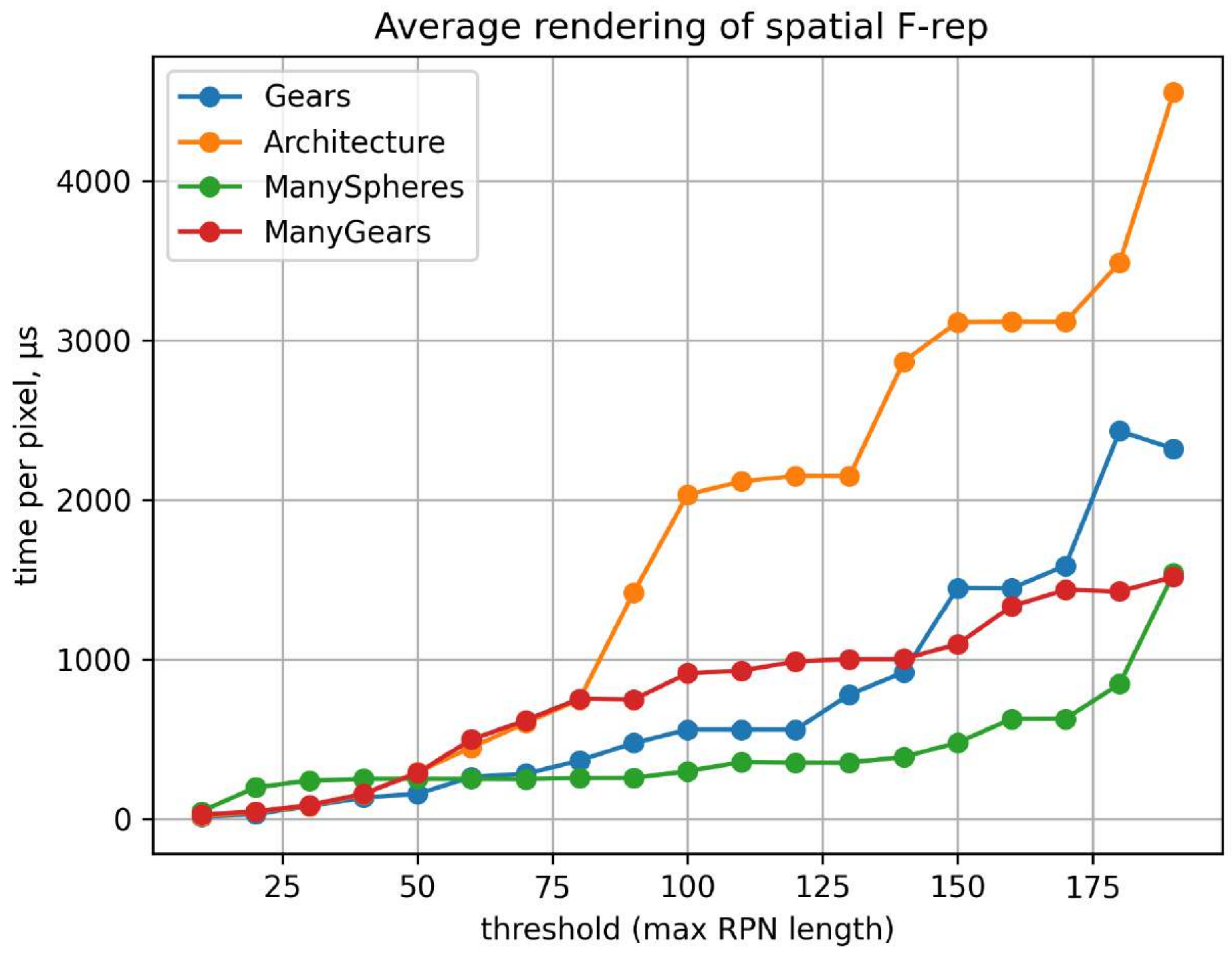


Results

Rendering speedup

Scene	Time per pixel, µs
Gears	20’000
Architecture	3’990’000
ManySpheres	16’000’000
ManyGears	840’885’000

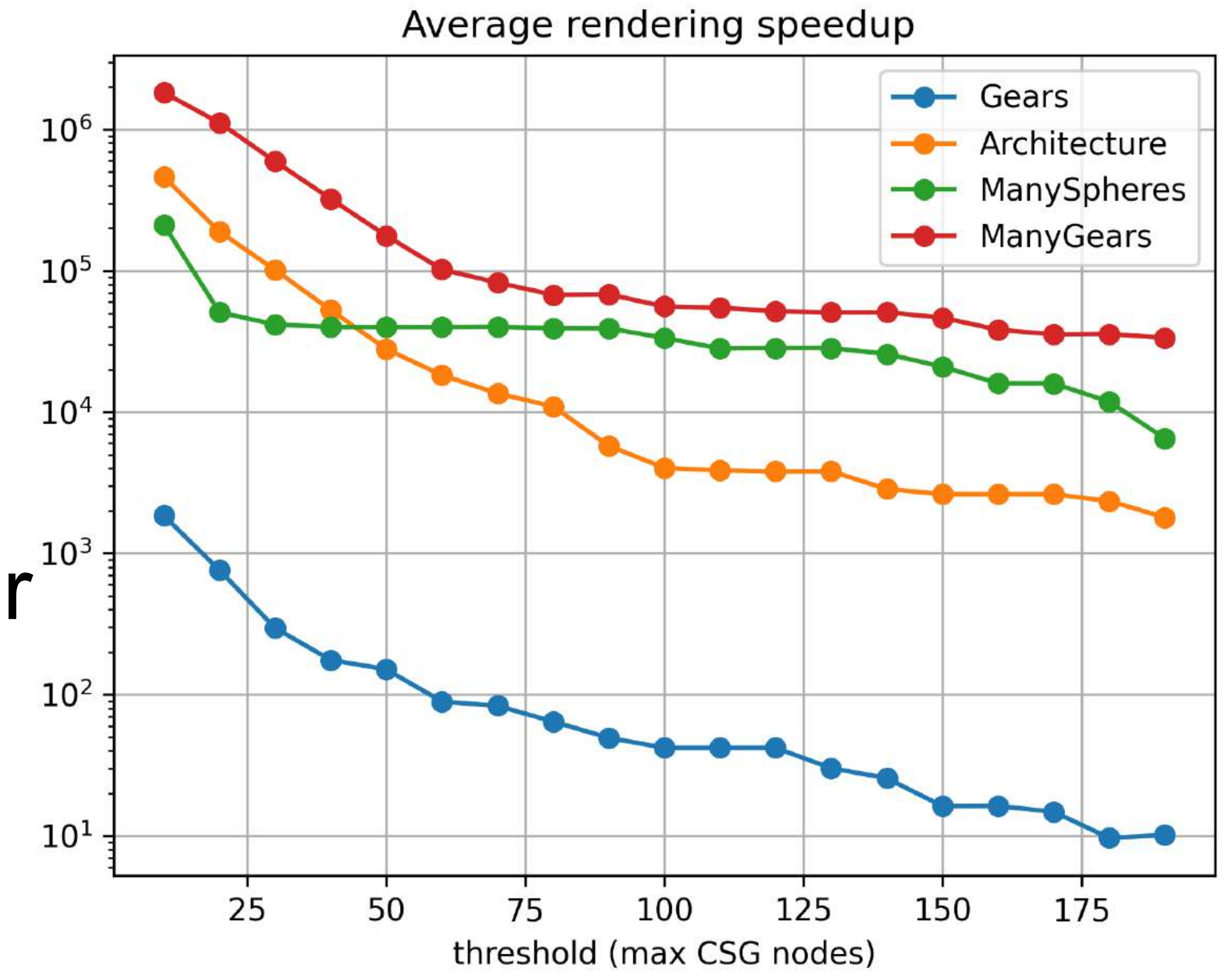
Average time of rendering original (unpruned) F-rep

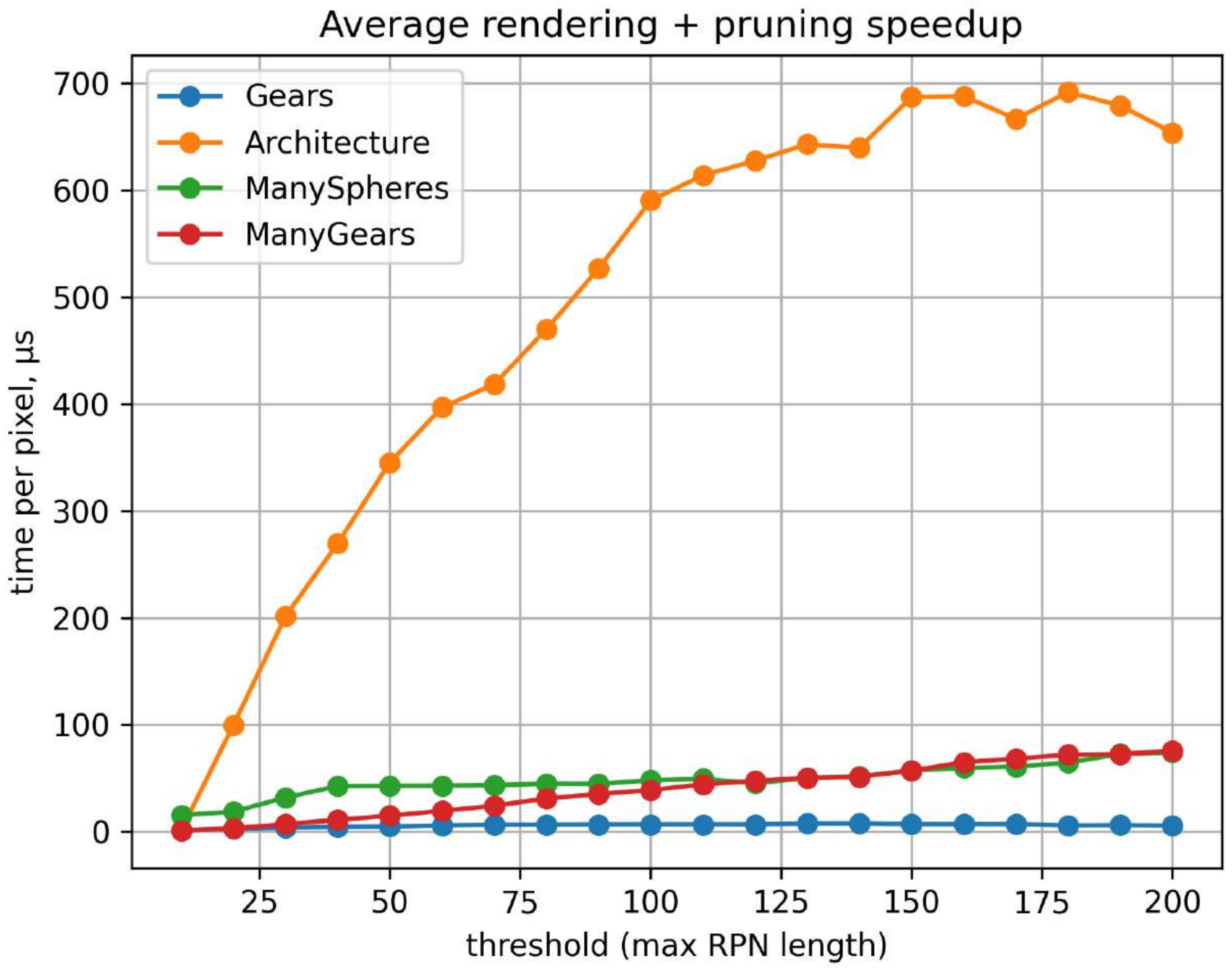


Rendering times are improved by one to six orders of magnitude (≈ **10x** to 10^6**x**)

This **significant** boost can be explained by the very poor performance of the interval bisection method for the original F-rep.

Additionally, computing pruning for every ray can boost ray-casting performance up to **700** times.





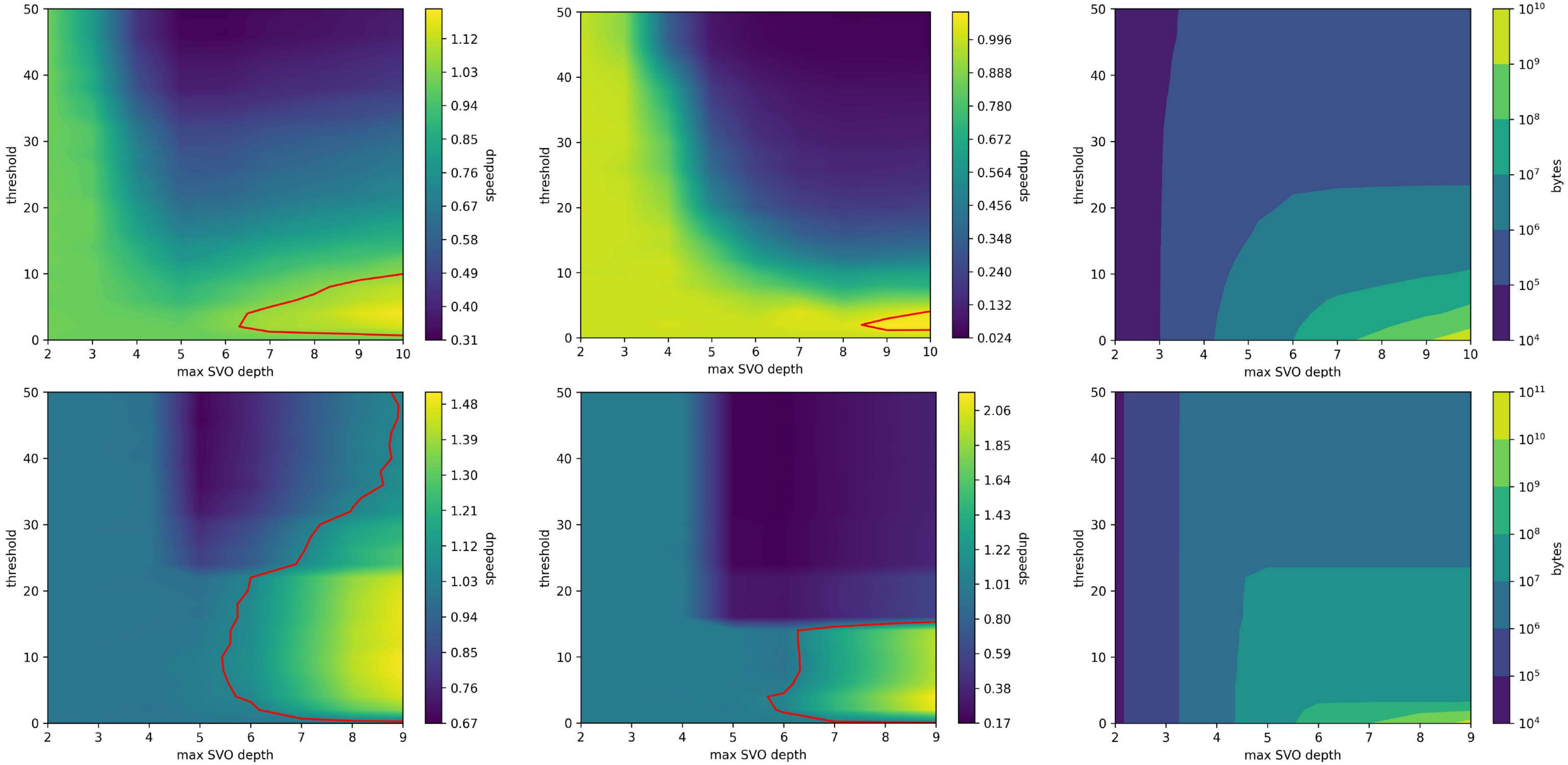
Results

Comparison with state-of-the-art approaches

Evaluation speedup | **Rendering speedup** | **Memory consumption**

Architecture

Many Spheres



Conclusion

Core mathematical framework for adaptive CSG modeling with constant evaluation complexity was developed and tested on 3D scenes with varying levels of complexity.

Developed pruning algorithm showed a significant speedup in F-rep evaluation. The worst case speedup was **4x** for the *Gears* model, and the best case speedup was **700x** for the *Many Gears* model.

The developed rendering algorithm also showed a significant speedup in F-rep ray-casting. Even computing pruning for every ray can boost ray-casting performance up to **700x** times.

All these algorithms were implemented in a prototype geometry kernel and ray tracer in C++.

Future Applications

Adaptive CSG framework

3D Printing

✓ Arbitrary resolution
✓ Heterogeneous materials

CAD Modeling

✓ Low-memory footprint
✓ Well-defined geometric operations

Mesh Generation

Procedural Content Generation in Virtual Environments

Thank you!

External Thesis Review

Questions

Can the author elaborate on how this threshold is chosen and whether it impacts the accuracy or generality of the results across varying model complexities? Can you suggest any preliminary recommendations for its value?

The threshold is chosen in range 5-200.

If at most N primitives have contact area, the threshold should be more than 2N-1.

The preliminary recommendations are threshold in range 5-20.

Skoltech

External Thesis Review

Questions

Which real-world application domains or industries could benefit most from your constant-complexity CSG approach, and what would be required to integrate it into existing modeling or manufacturing workflows?

Real-world application domains: CAD/CAM and 3D design software (e.g., Adobe Project Neo, Womp).

For CSG modeling workflows the integration is straightforward.

Otherwise, integration depends on the implementation of existing workflows.

Skoltech

External Thesis Review

Questions

What are the main assumptions and limitations of your adaptive CSG method? For instance, are there specific cases (such as extremely complex geometries or degenerate configurations) where the constant-complexity guarantee might not hold or the algorithm could struggle?

Constant evaluation complexity is guaranteed in case when SVO time traversal is much less than CSG tree evaluation.

It might not hold in areas of contact between large number of primitives (see Architecture scene).

Skoltech

Bibliography

F-rep concepts:

A. Pasko, V Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, Aug 1995.

V. L. Rvachev. Method of r-functions in boundary-value problems. *Soviet Applied Mechanics*, 11(4):345–354, April 1975.

Pruning F-rep:

Oleg Fryazinov, Alexander Pasko, and Peter Comninos. Fast reliable interrogation of procedurally defined implicit surfaces using extended revised affine arithmetic. *Computers & Graphics*, 34(6):708–718, 2010.

Matthew J. Keeter. Massively parallel rendering of complex closed-form implicit surfaces. *ACM Trans. Graph.*, 39(4), August 2020.

Tom Duff. Interval arithmetic recursive subdivision for implicit functions and constructive solid geometry. SIGGRAPH Comput. Graph., 26(2):131–138, July 1992.

Christopher Uchytil and Duane Storti. A function-based approach to interactive high-precision volumetric design and fabrication. *ACM Trans. Graph.*, 43(1), September 2023.

Evgenii Maltsev, Dmitry Popov, Svyatoslav Chugunov, Alexander Pasko, and Iskander Akhatov. An accelerated slicing algorithm for frep models. *Applied Sciences*, 11(15), 2021.

Sparse volume data structures:

Ken Museth. Vdb: High-resolution sparse volumes with dynamic topology. 32(3), July 2013.

Rama Karl Hoetzlein. GVDB: Raytracing Sparse Voxel Database Structures on the GPU. In Ulf Assarsson and Warren Hunt, editors, *Eurographics/ ACM SIGGRAPH Symposium on High Performance Graphics*. The Eurographics Association, 2016.

Doyub Kim, Minjae Lee, and Ken Museth. Neuralvdb: High-resolution sparse volume representation using hierarchical neural networks. *ACM Trans. Graph.*, 43(2), February 2024.

Johanna Beyer, Markus Hadwiger, and Hanspeter Pfister. State-of-the-art in gpu-based large-scale volume visualization. *Computer Graphics Forum*, 34(8):13–37, 2015.

Skoltech