SIGGRAPH ASIA 2018 TOKYO

CONFERENCE 4 – 7 December 2018
EXHIBITION 5 – 7 December 2018
Tokyo International Forum, Japan
SA2018.SIGGRAPH.ORG

Sponsored by

GPU-Based Large-Scale Scientific Visualization

Johanna Beyer, Harvard University

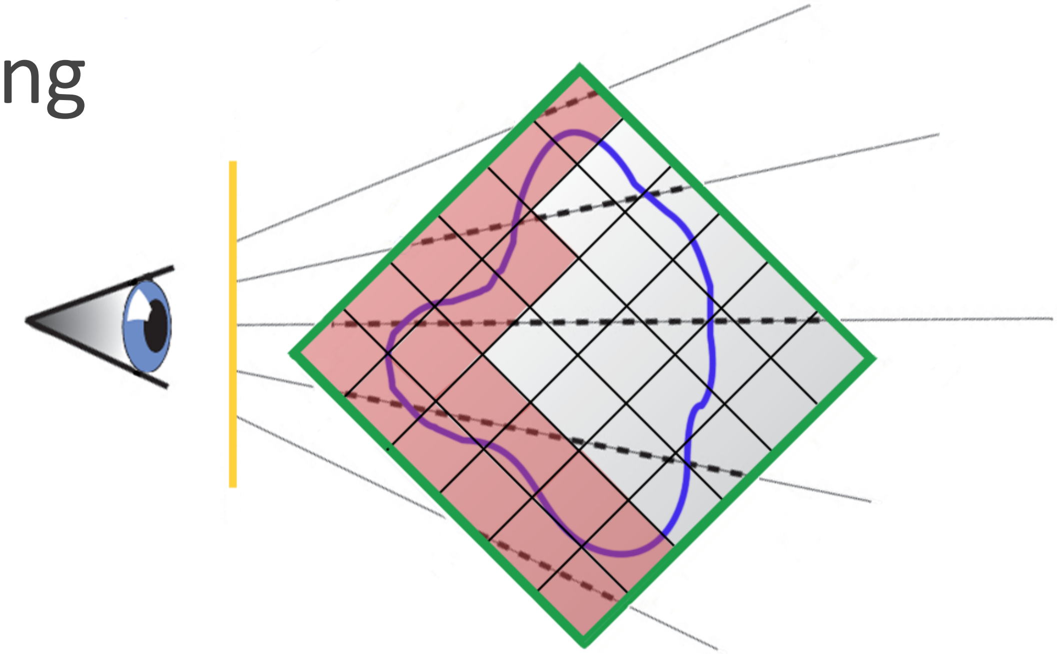
Markus Hadwiger, KAUST

Course Website:
http://johanna-b.github.io/LargeSciVis2018/index.html

Part 3 -
GPU-Based Ray-Guided Volume Rendering Algorithms & Efficient Empty Space Skipping

RAY-GUIDED VOLUME RENDERING

- Working set determination on GPU
- Single-pass rendering
- Traversal on GPU
- Virtual texturing



RAY-GUIDED VOLUME RENDERING (2)

Examples using octree traversal (kd-restart):

- Gigavoxels [Crassin et al., 2009]
  - Gigavoxel isosurface and volume rendering
- Tera-CVR [Engel, 2011]
  - Teravoxel volume rendering with dynamic transfer functions

RAY-GUIDED VOLUME RENDERING (2)

Examples using virtual texturing instead of tree traversal

- Petascale volume exploration of microscopy streams [Hadwiger et al., 2012]
  - *Visualization-driven* pipeline, including data construction
- ImageVis3D [Fogal et al., 2013]
  - Analysis of different settings (brick size, …)

Ray-guided Volume Rendering Examples

EARLY ‘RAY-GUIDED’ OCTREE RAY-CASTING (1)

[Gobbetti et al., The Visual Computer, 2008]

Volume representation	Octree
Rendering	GPU octree traversal
Working set determination	Interleaved occlusion queries

EARLY ‘RAY-GUIDED’ OCTREE RAY-CASTING (1)

Data structure: Octree with ropes

- Pointers to 8 children, 6 neighbors and volume data
- Active subtree stored in spatial index structure and texture pool on GPU

[Gobbetti et al.]

Volume representation	**Octree**
Rendering	GPU octree traversal
Working set determination	Interleaved occlusion queries

EARLY ‘RAY-GUIDED’ OCTREE RAY-CASTING (2)

Rendering:

- Stackless GPU octree traversal (rope tree)

[Gobbetti et al.]

Volume representation	Octree
Rendering	**GPU octree traversal**
Working set determination	Interleaved occlusion queries

EARLY ‘RAY-GUIDED’ OCTREE RAY-CASTING (2)

Culling: Culling on CPU

- Culling uses global transfer function, iso-value, view frustum
- Only visible nodes of previous rendering pass get refined
- Occlusion queries to check bounding box of node against depth of last sample during raycasting

[Gobbetti et al.]

Volume representation	Octree
Rendering	GPU octree traversal
Working set determination	**Interleaved occlusion queries**

RAY-GUIDED OCTREE RAY-CASTING (1)

[Crassin et al., ACM SIGGRAPH i3D, 2009]

Volume representation	Octree
Rendering	GPU octree traversal
Working set determination	Ray-guided

RAY-GUIDED OCTREE RAY-CASTING (1)

Data structure: N^3 tree + multi-resolution volume

- Subtree stored on GPU in node/brick pool
  - Node: 1 pointer to children, 1 pointer to volume brick
  - Children stored together in node pool

[Crassin et al.]

Volume representation	**Octree**
Rendering	GPU octree traversal
Working set determination	Ray-guided

RAY-GUIDED OCTREE RAY-CASTING (2)

Rendering:

- Stackless GPU octree traversal (Kd-restart)
- 3 mipmap levels for correct filtering
- Missing data substituted by lower-res data

[Crassin et al.]

Volume representation	Octree
Rendering	**GPU octree traversal**
Working set determination	Ray-guided

RAY-GUIDED OCTREE RAY-CASTING (2)

Culling:

- Multiple render targets write out data usage
- Exploits temporal and spatial coherence

[Crassin et al.]

Volume representation	Octree
Rendering	GPU octree traversal
Working set determination	**Ray-guided**

RAY-GUIDED MULTI-LEVEL PAGETABLE RAY-CASTING (1)

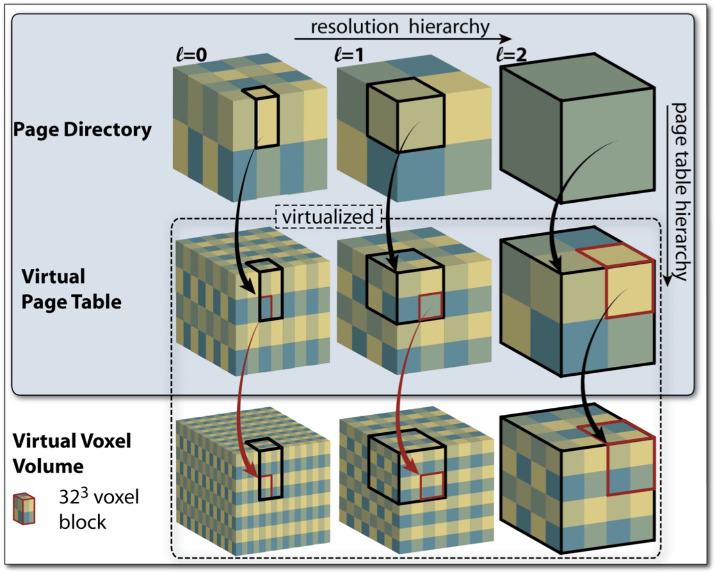
[Hadwiger et al., IEEE SciVis 2012]

Volume representation	Multi-resolution grid
Rendering	Multi-level virtual texture ray-casting
Working set determination	Ray-guided

RAY-GUIDED MULTI-LEVEL PAGETABLE RAY-CASTING (1)

Data structure: Multi-res grid

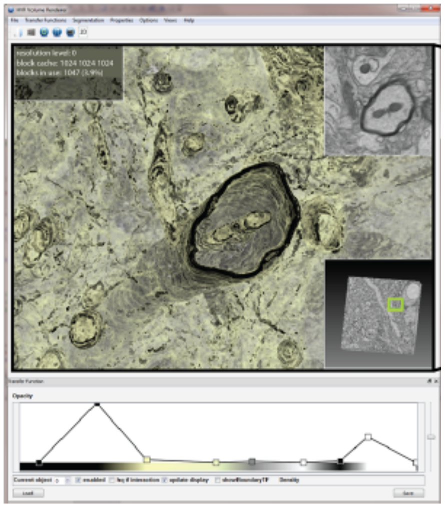
- On-the-fly reconstruction of bricks
- Stored on disk in 2D multi-resolution grid
- Multi-level multi-res. page table on GPU



[Hadwiger et al.]

Volume representation	Multi-resolution grid
Rendering	Multi-level virtual texture ray-casting
Working set determination	Ray-guided

RAY-GUIDED MULTI-LEVEL PAGETABLE RAY-CASTING (2)



resolution level: 0
block cache: 1024 1024 1024
blocks in use: 1047 (3.99%)

Rendering:

- Multi-level virtual texture ray-casting
- LOD chosen per individual sample
- Data reconstruction triggered by ray-caster

[Hadwiger et al.]

Volume representation	Multi-resolution grid
Rendering	**Multi-level virtual texture ray-casting**
Working set determination	Ray-guided

RAY-GUIDED MULTI-LEVEL PAGETABLE RAY-CASTING (2)

Culling:

- GPU hash table to report missing blocks
  - Exploits temporal and spatial coherence

[Hadwiger et al.]

Volume representation	Multi-resolution grid
Rendering	Multi-level virtual texture ray-casting
Working set determination	**Ray-guided**

RAY-GUIDED MULTI-LEVEL PAGETABLE RAY-CASTING - ANALYSIS

[Fogal et al., IEEE LDAV 2013]

Volume representation	Multi-resolution grid
Rendering	(Multi-level) virtual texture ray-casting
Working set determination	Ray-guided

RAY-GUIDED MULTI-LEVEL PAGETABLE RAY-CASTING - ANALYSIS

Implementation differences:

- Lock-free hash table, pagetable lookup only per brick
- Fallback for multi-pass rendering

[Fogal et al.]

Volume representation	Multi-resolution grid
Rendering	(Multi-level) virtual texture ray-casting
Working set determination	Ray-guided

RAY-GUIDED MULTI-LEVEL PAGETABLE RAY-CASTING - ANALYSIS

Analysis:

- Many detailed performance numbers (see paper)
- Working set size: typically lower than GPU memory
- Brick size: larger on disk (>= 64^3), smaller for rendering (16^3, 32^3)

[Fogal et al.]

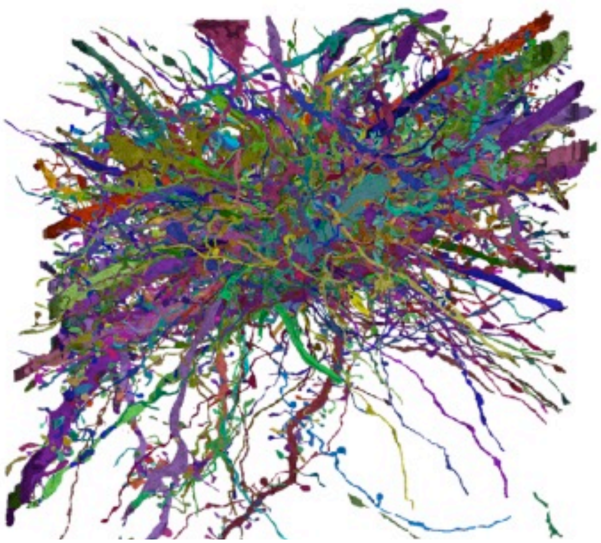
Volume representation	Multi-resolution grid
Rendering	(Multi-level) virtual texture ray-casting
Working set determination	Ray-guided

SIGGRAPH ASIA 2018 TOKYO

Scalable Empty-Space Skipping

MOTIVATION

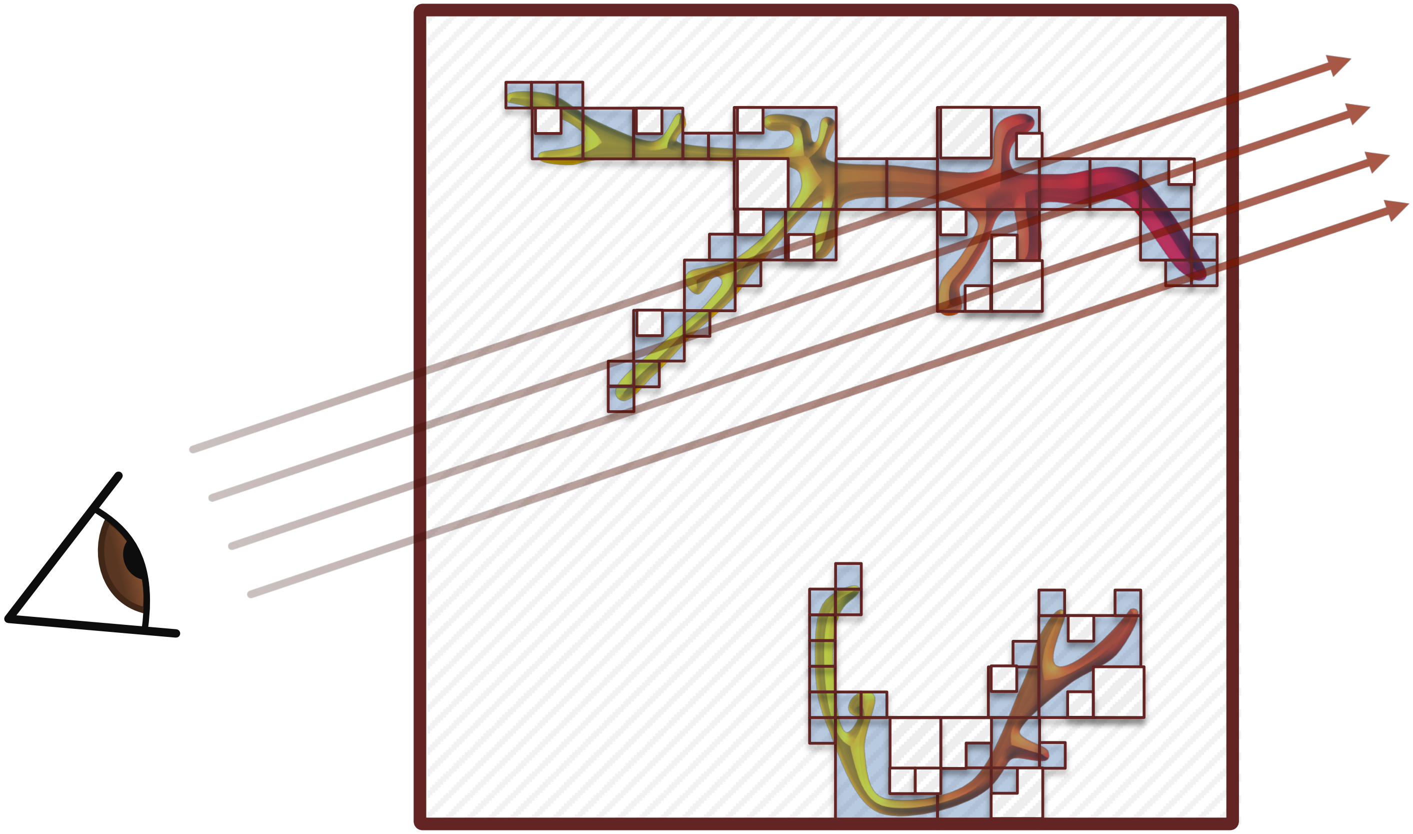
Large volumes, finely detailed structures, many segmented objects

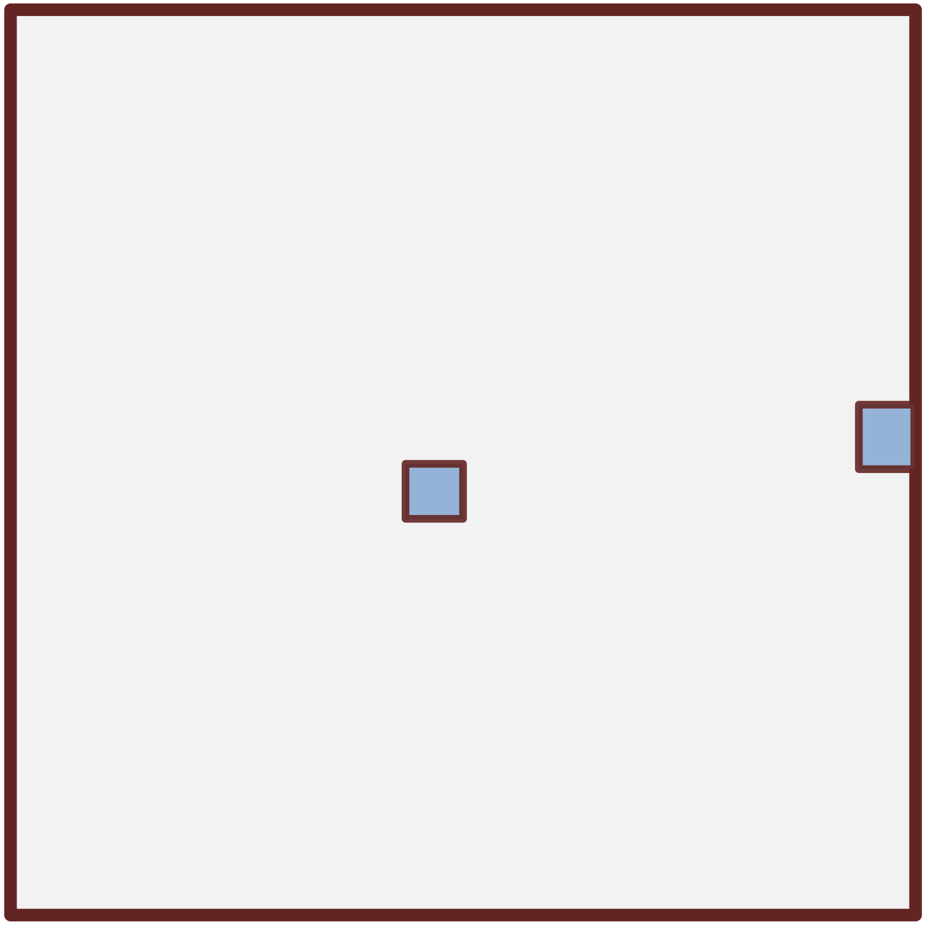


connectomics electron microscopy volume

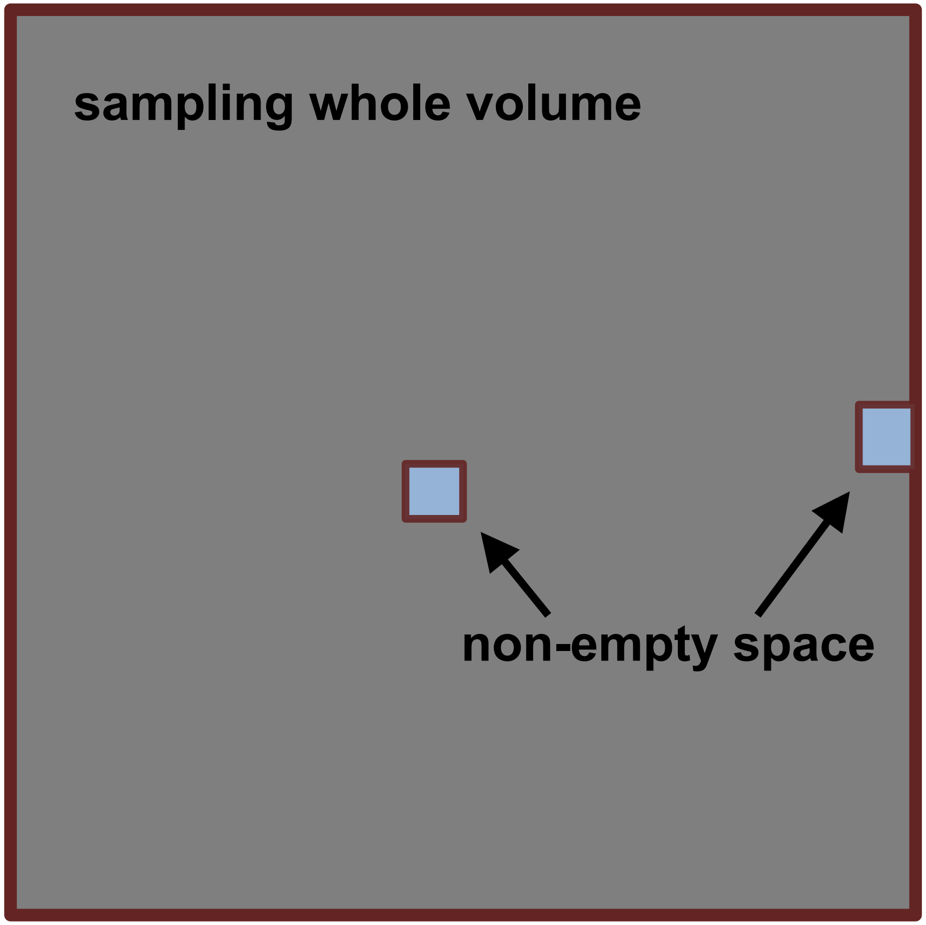
21,000 x 25,000 x 2,000 > 1 teravoxels > 4,000 objects

MOTIVATION





no skipping

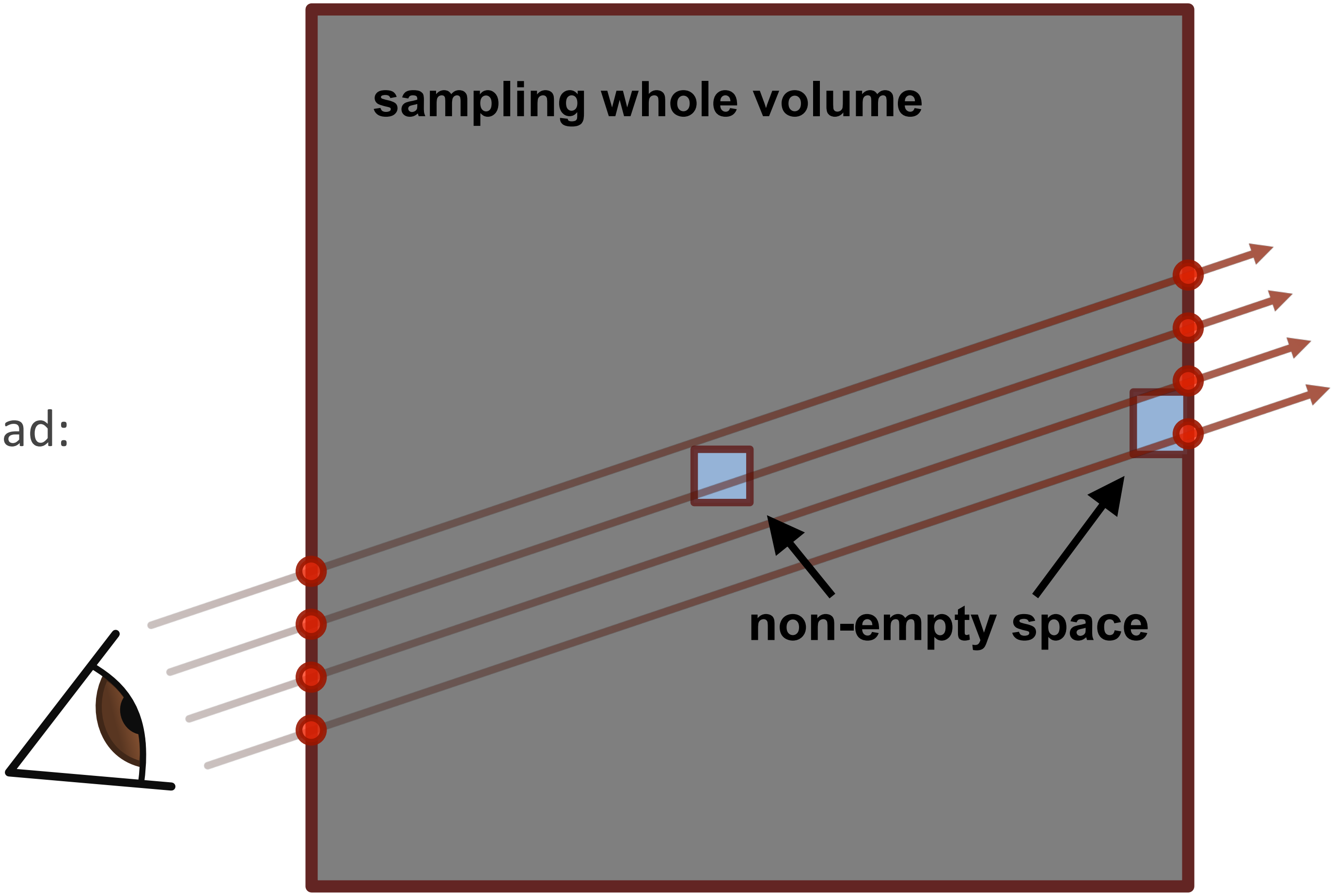


no skipping

look-up overhead:

none

● look-ups



sampling whole volume

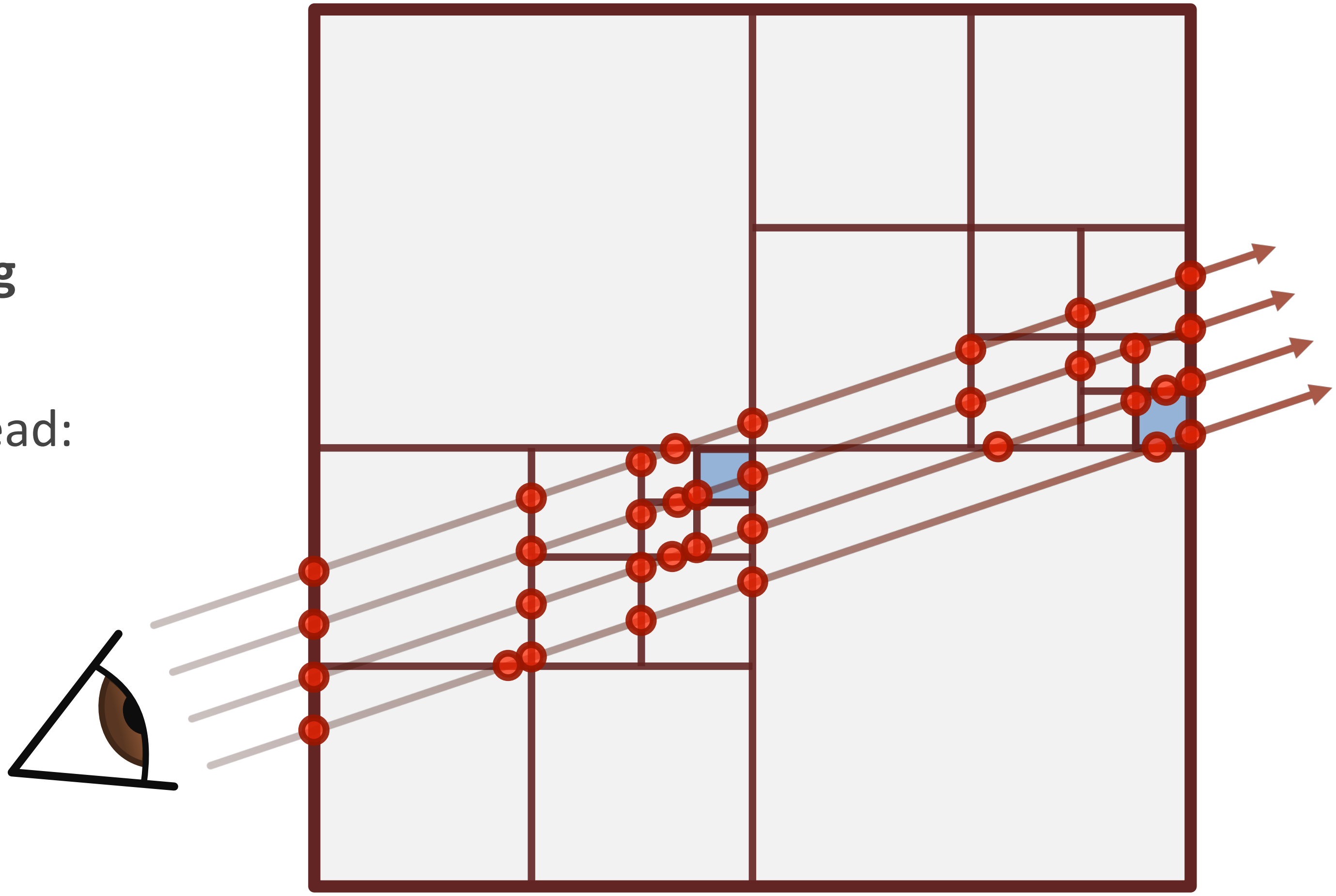
non-empty space

SIGGRAPH ASIA 2018 TOKYO

octree skipping

look-up overhead:

high

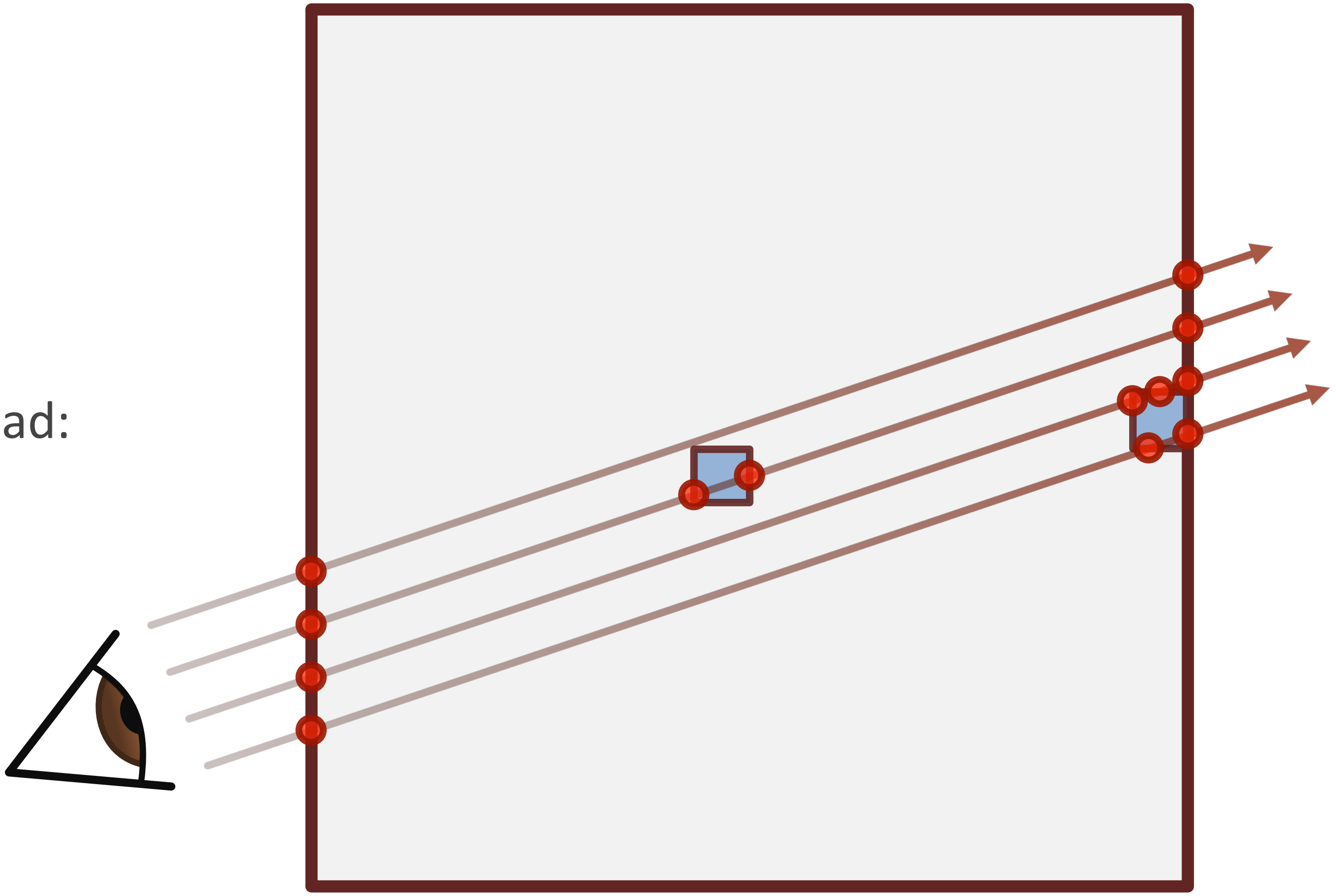


● look-ups

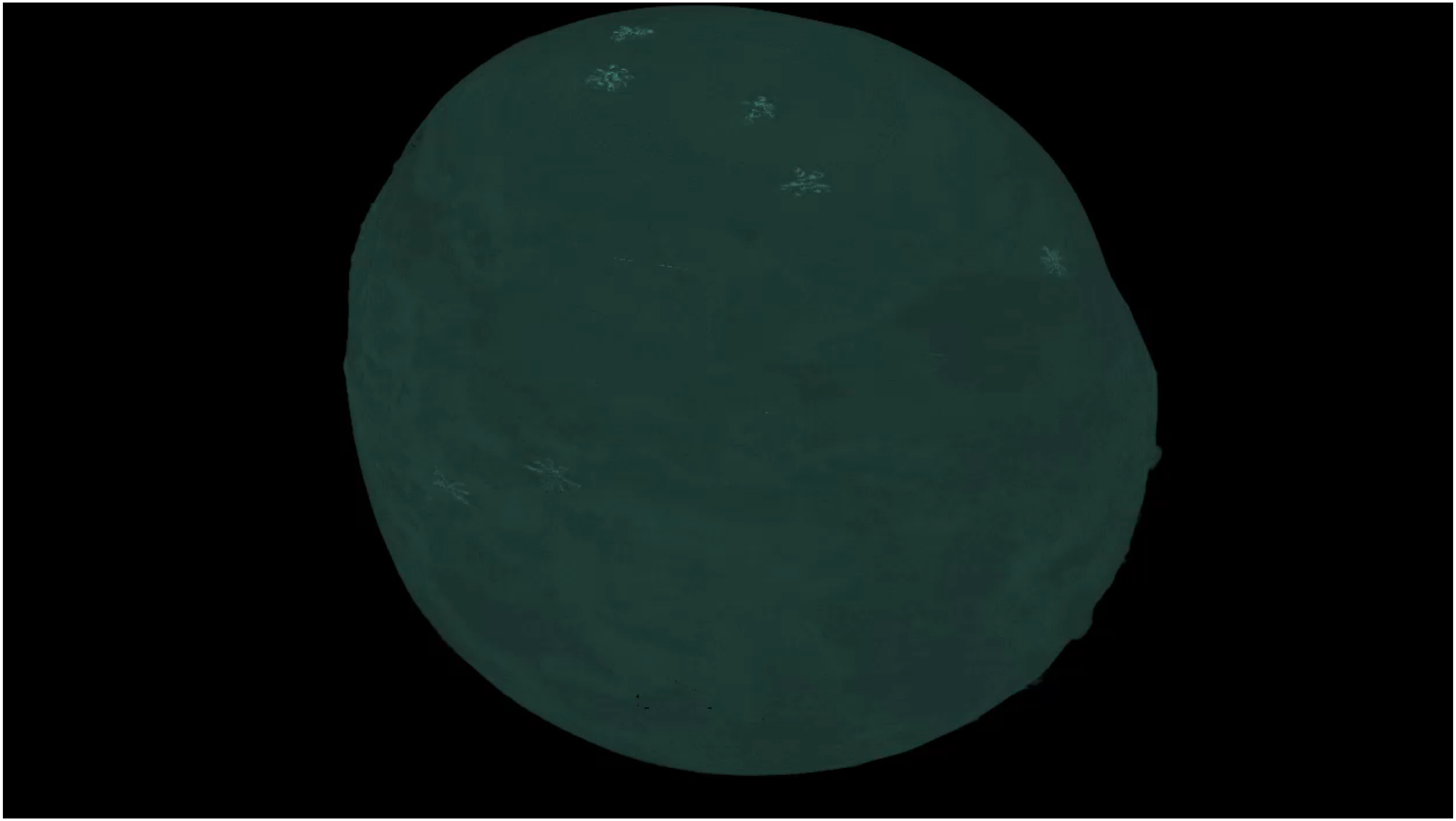
SparseLeap

look-up overhead:

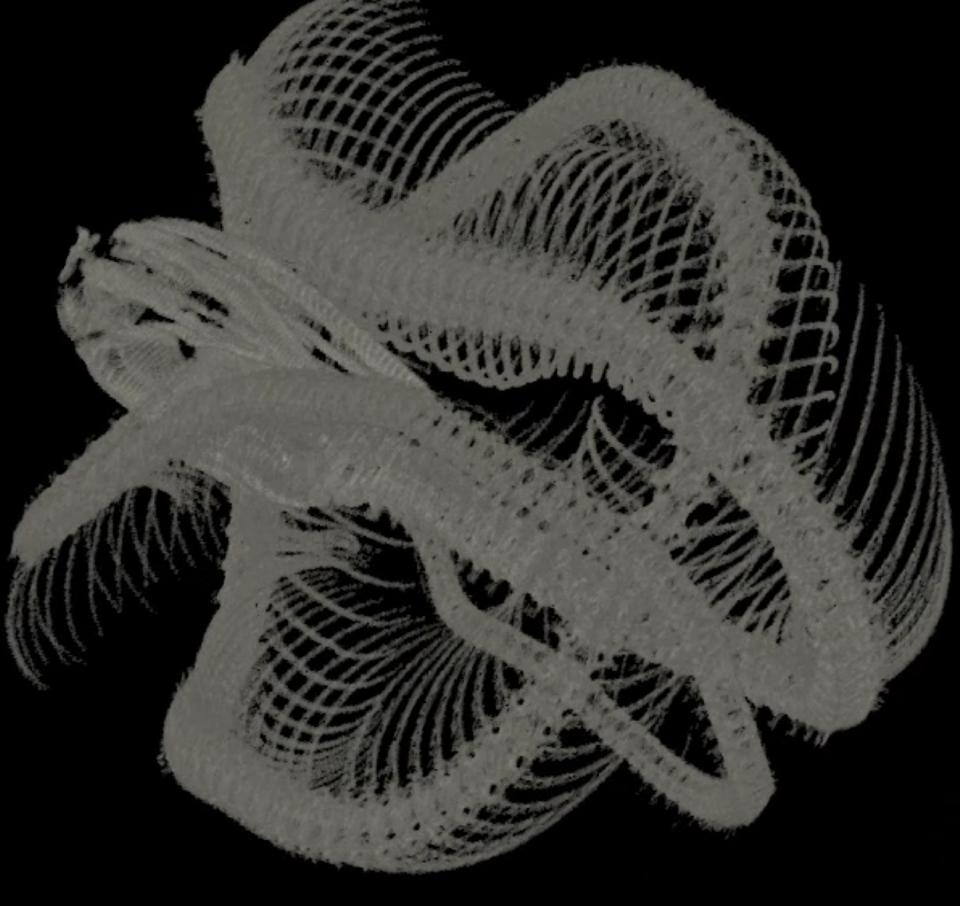
small



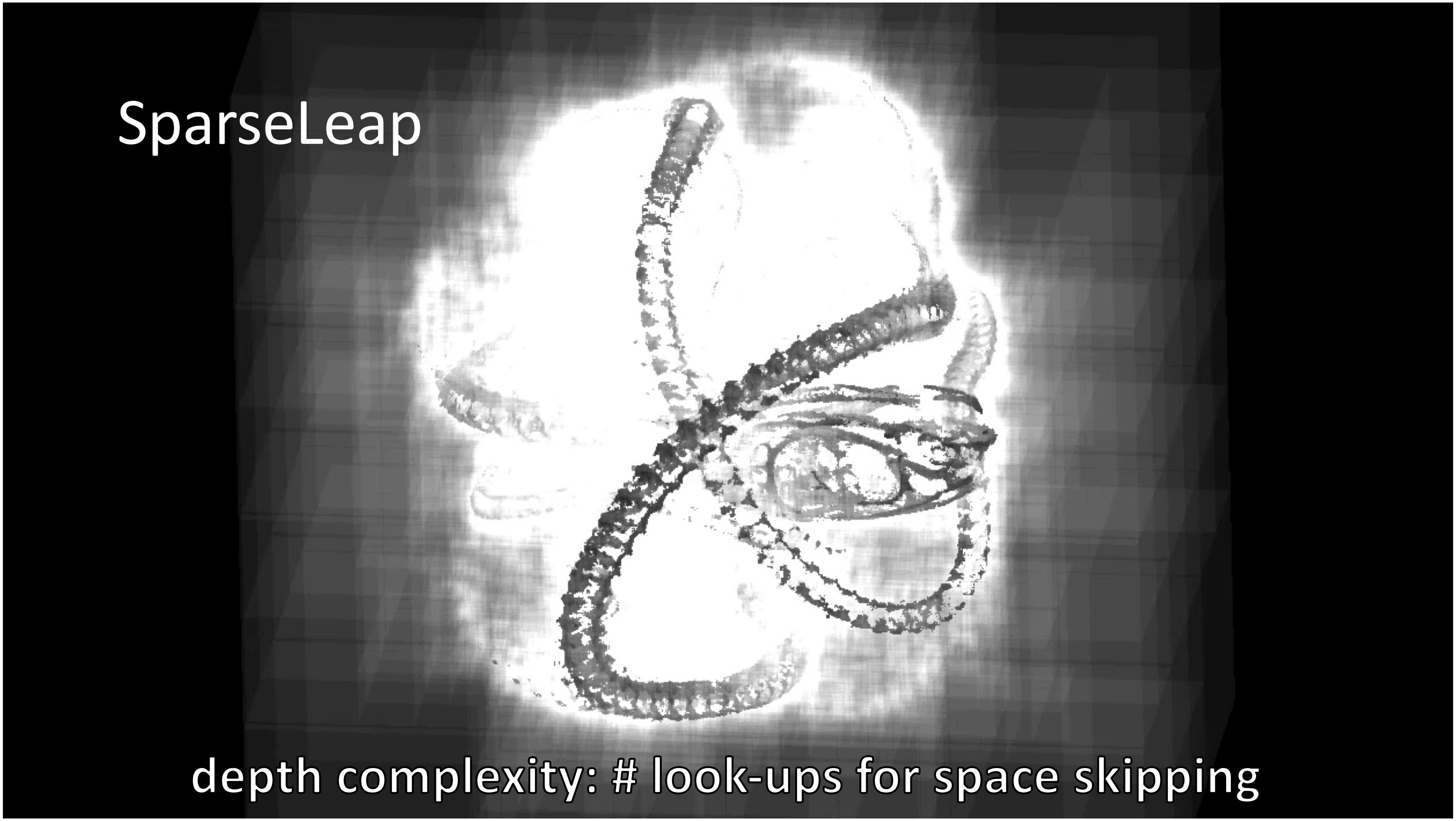
● look-ups



Octree



depth complexity: # look-ups for space skipping



SparseLeap

depth complexity: # look-ups for space skipping

SPARSELEAP PIPELINE

Track volume occupancy

- Occupancy histogram tree

Extract nested occupancy

- Occupancy geometry

Rasterize occupancy

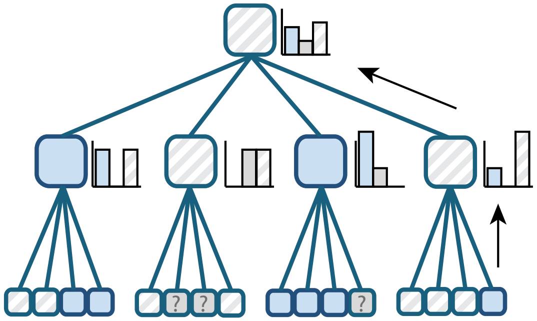
- Ray segment lists

Empty space skipping: Linear list traversal

SPARSELEAP PIPELINE

Track volume occupancy

- **Occupancy histogram tree**



Extract nested occupancy

- Occupancy geometry

Rasterize occupancy

- Ray segment lists

Empty space skipping: Linear list traversal

SPARSELEAP PIPELINE

Track volume occupancy

- Occupancy histogram tree

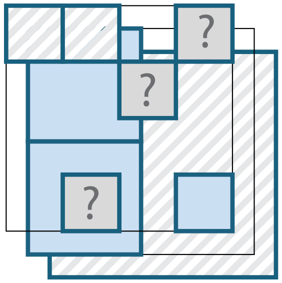
Extract nested occupancy

- **Occupancy geometry**

Rasterize occupancy

- Ray segment lists

Empty space skipping: Linear list traversal



SPARSELEAP PIPELINE

Track volume occupancy

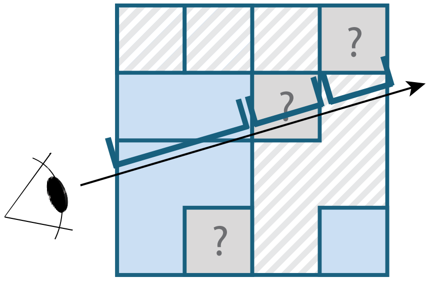
- Occupancy histogram tree

Extract nested occupancy

- Occupancy geometry

Rasterize occupancy

- **Ray segment lists**



Empty space skipping: Linear list traversal

SPARSELEAP PIPELINE

Track volume occupancy

- Occupancy histogram tree

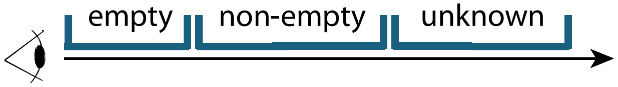
Extract nested occupancy

- Occupancy geometry

Rasterize occupancy

- Ray segment lists

Empty space skipping: Linear list traversal

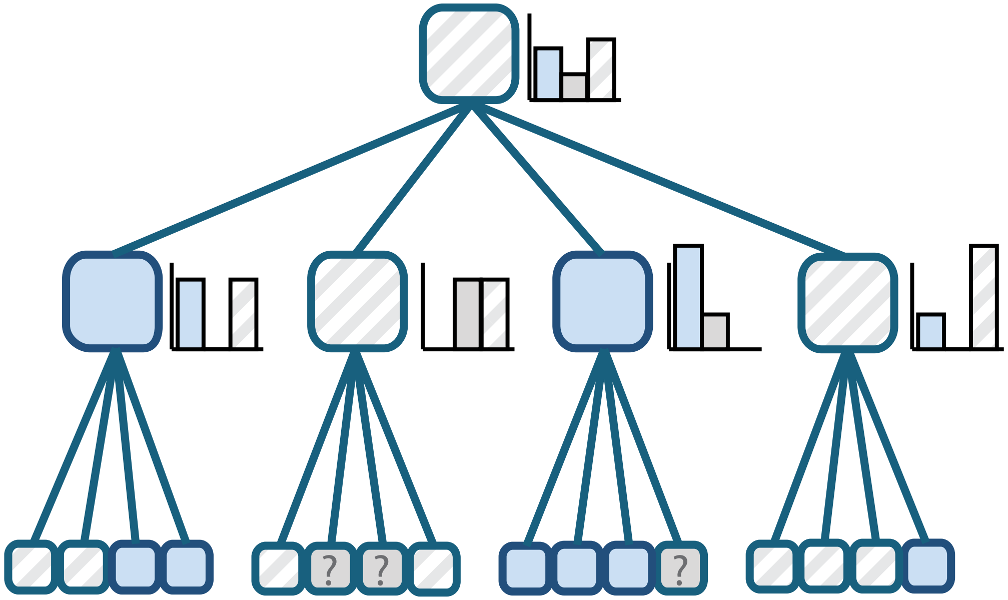


OCCUPANCY HISTOGRAM TREE

Occupancy classes

- 🟦 non-empty
- ▨ empty
- ? unknown

Node count in each class over whole subtree

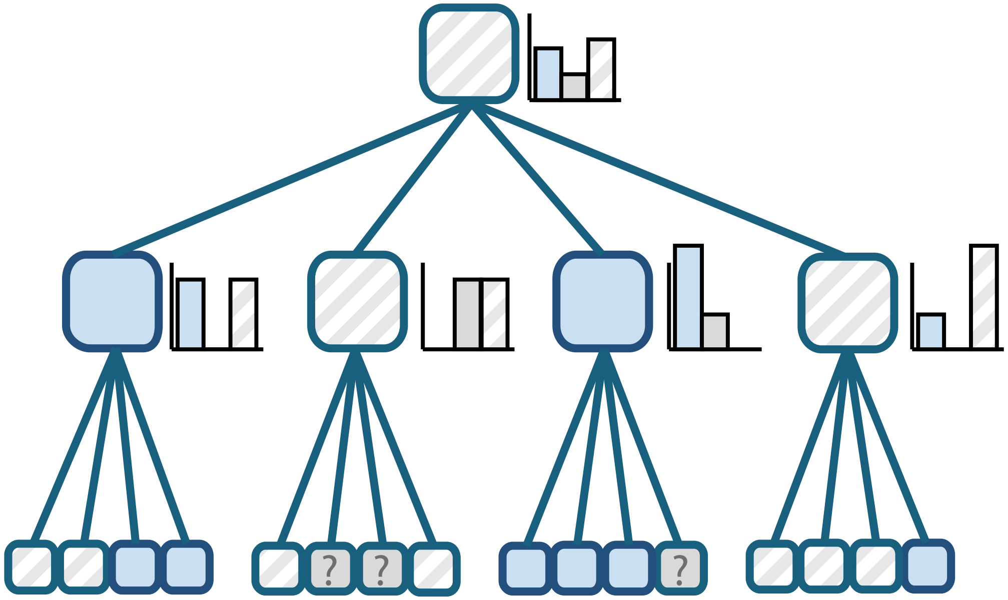


OCCUPANCY HISTOGRAM TREE

Occupancy classes

- ■ non-empty
- ▨ empty
- ? unknown *

Node count in each class over whole subtree



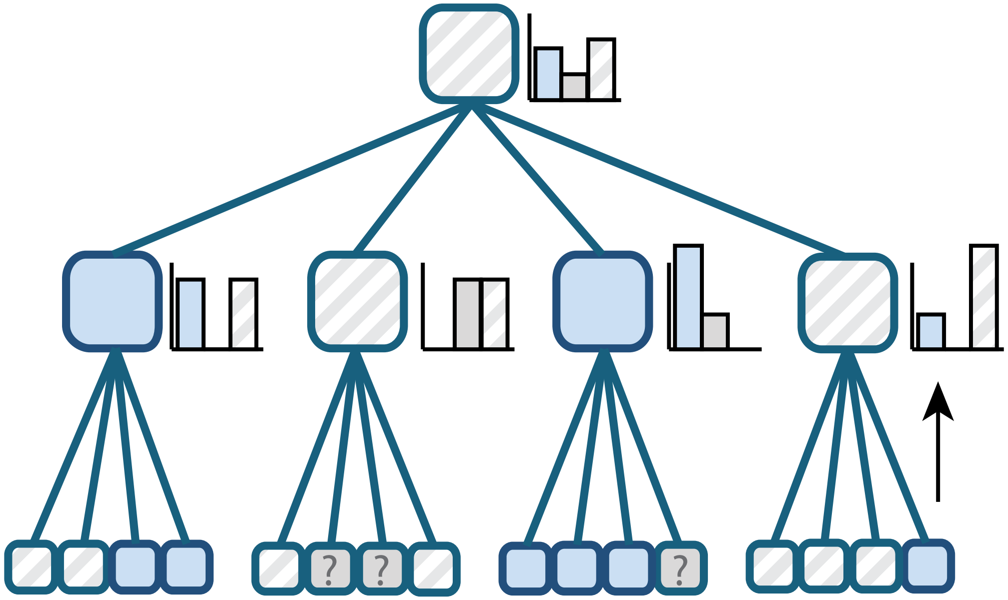
* enables deferred culling

OCCUPANCY HISTOGRAM TREE

Occupancy classes

- ■ non-empty
- ▨ empty
- ? unknown *

Node count in each class over whole subtree



build bottom-up

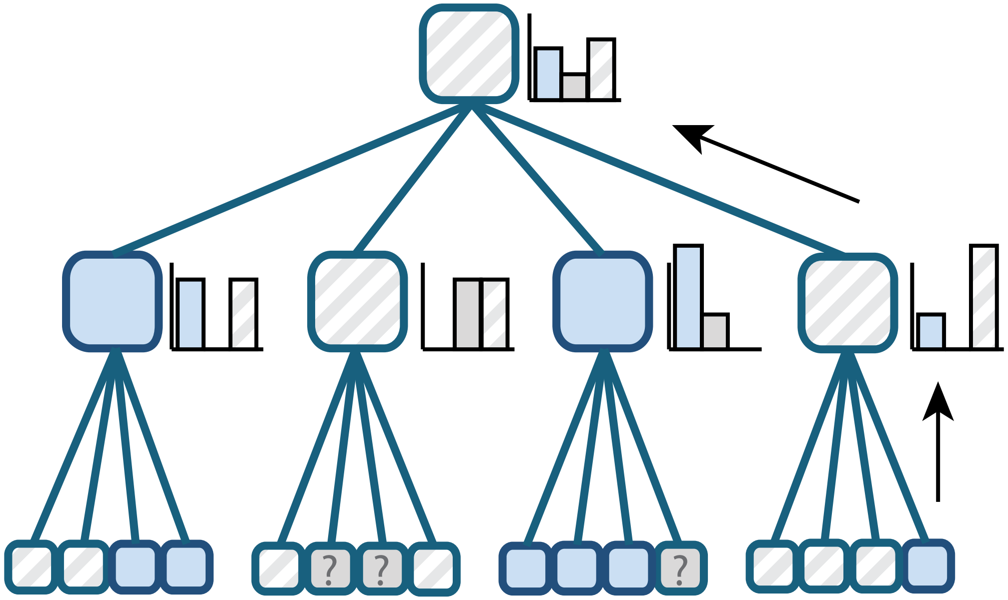
* enables deferred culling

OCCUPANCY HISTOGRAM TREE

Occupancy classes

- ■ non-empty
- ▨ empty
- ? unknown *

Node count in each class over whole subtree



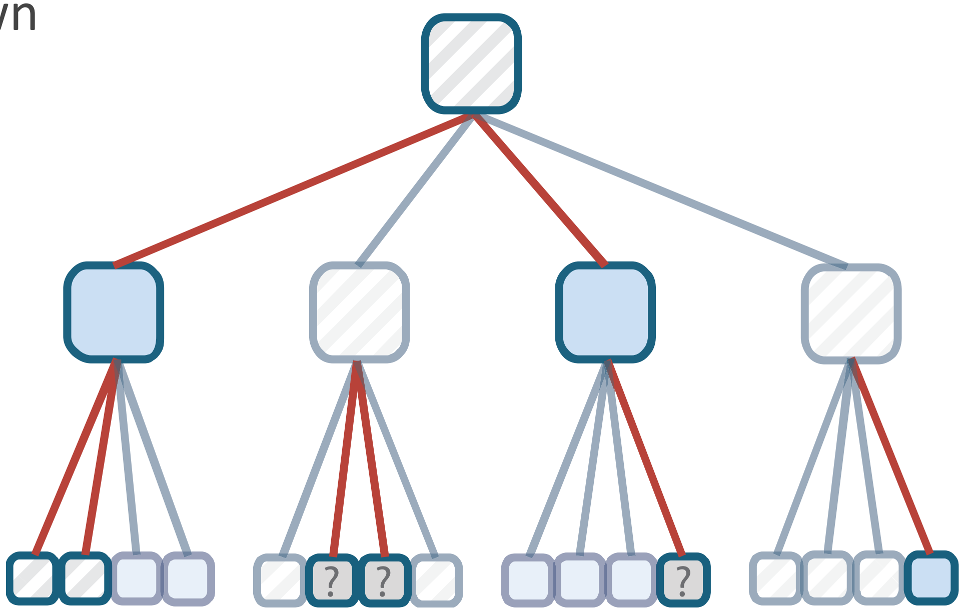
build bottom-up

* enables deferred culling

OCCUPANCY GEOMETRY

Traverse histogram tree top-down

Pick majority class in each node

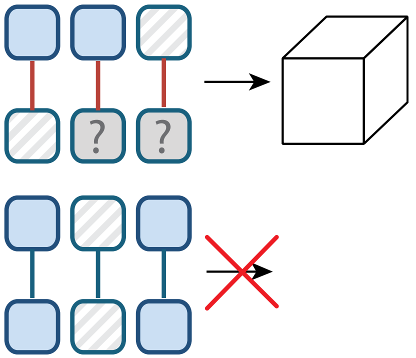


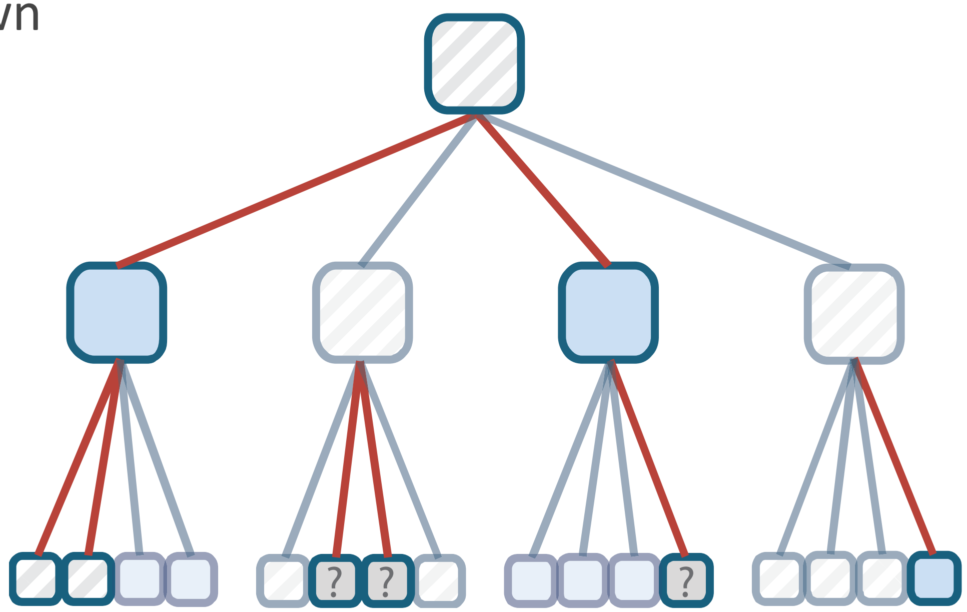
OCCUPANCY GEOMETRY

Traverse histogram tree top-down

Pick majority class in each node

Emit box on class change



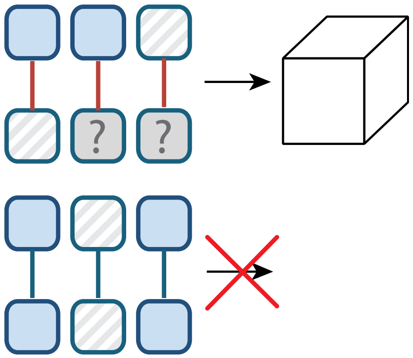


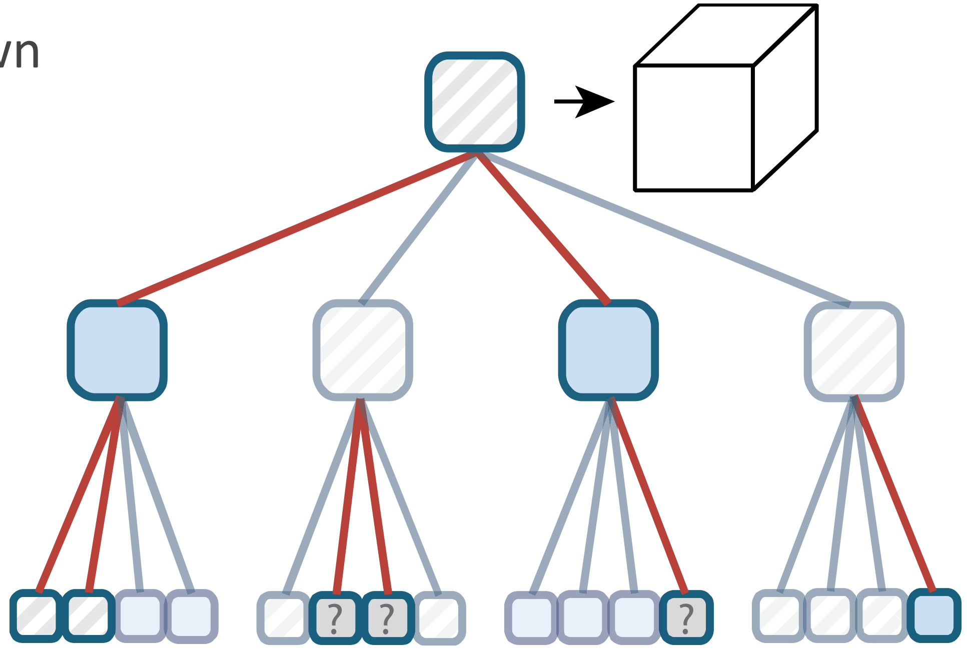
OCCUPANCY GEOMETRY

Traverse histogram tree top-down

Pick majority class in each node

Emit box on class change



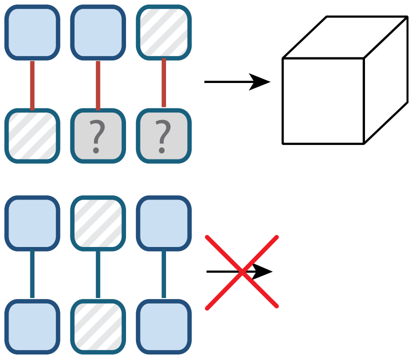


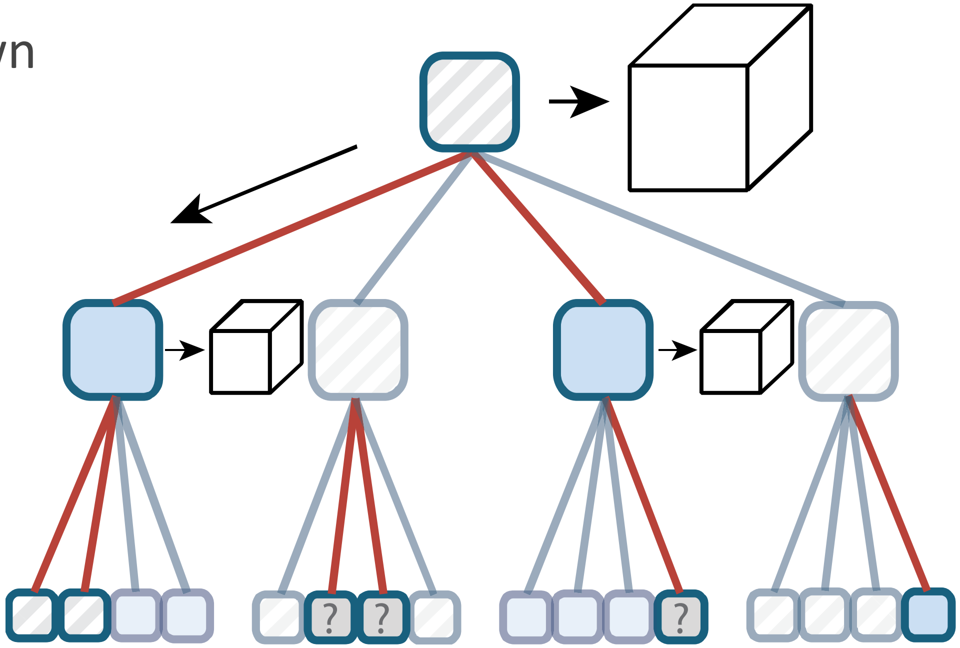
OCCUPANCY GEOMETRY

Traverse histogram tree top-down

Pick majority class in each node

Emit box on class change



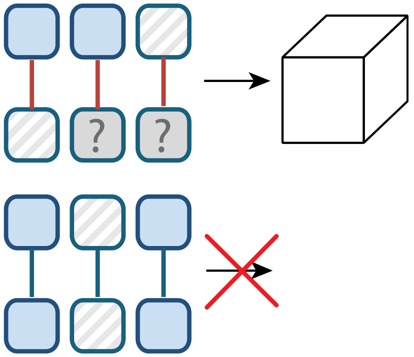


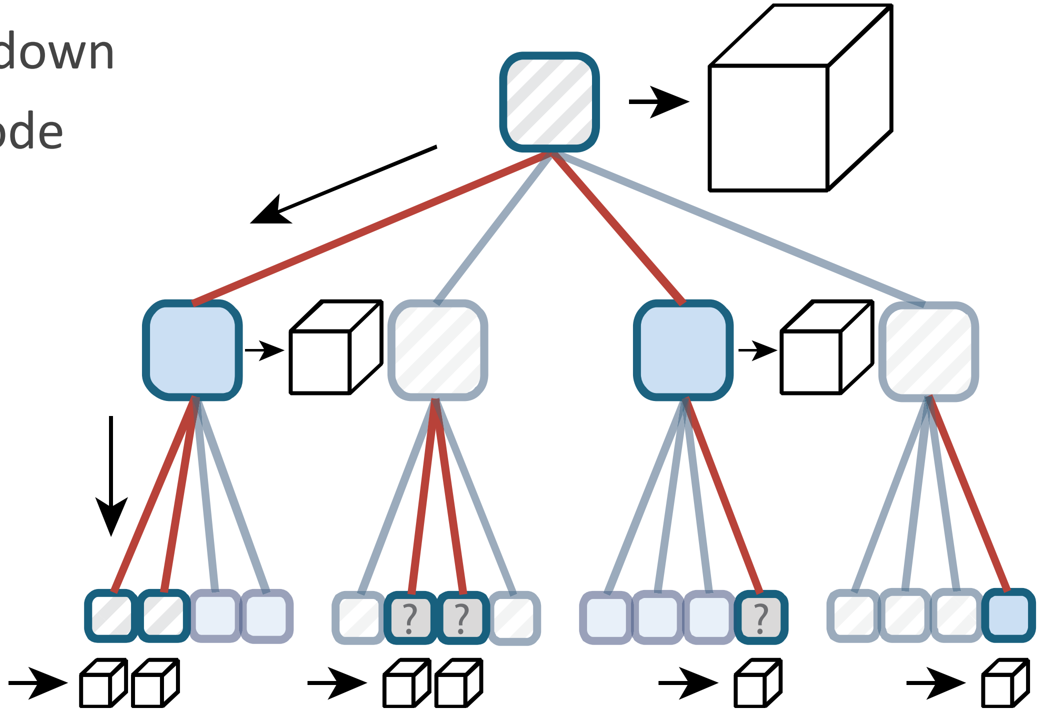
OCCUPANCY GEOMETRY

Traverse histogram tree top-down

Pick majority class in each node

Emit box on class change



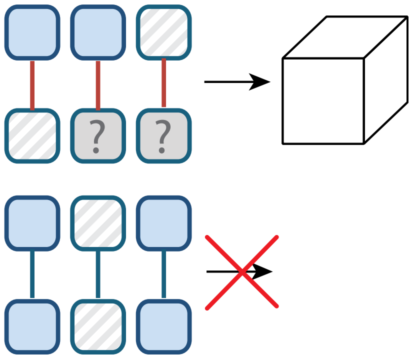


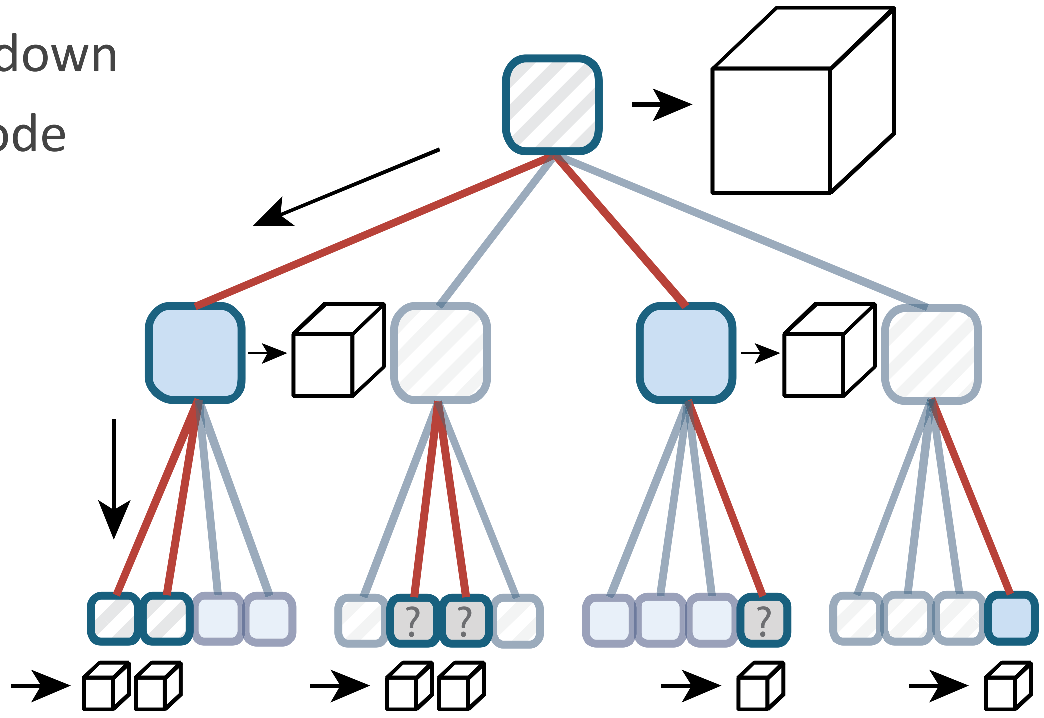
OCCUPANCY GEOMETRY

Traverse histogram tree top-down

Pick majority class in each node

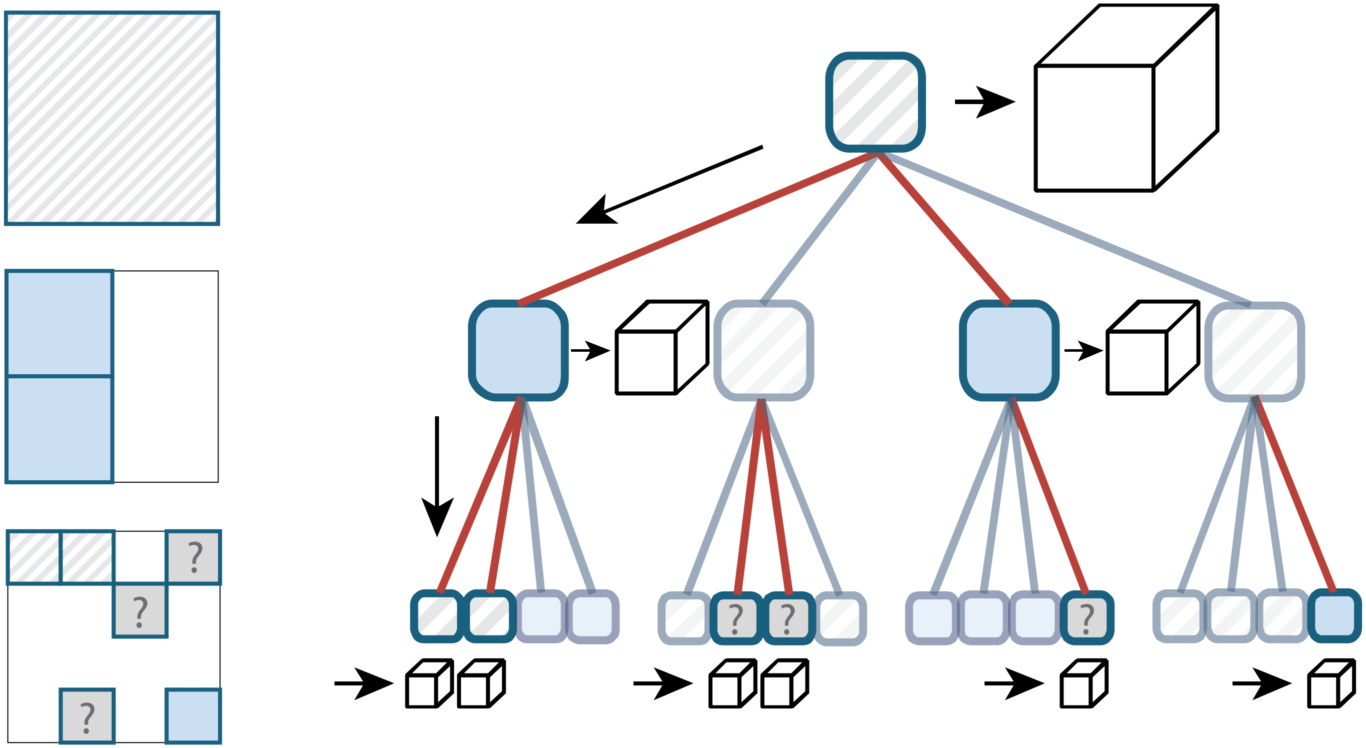
Emit box on class change





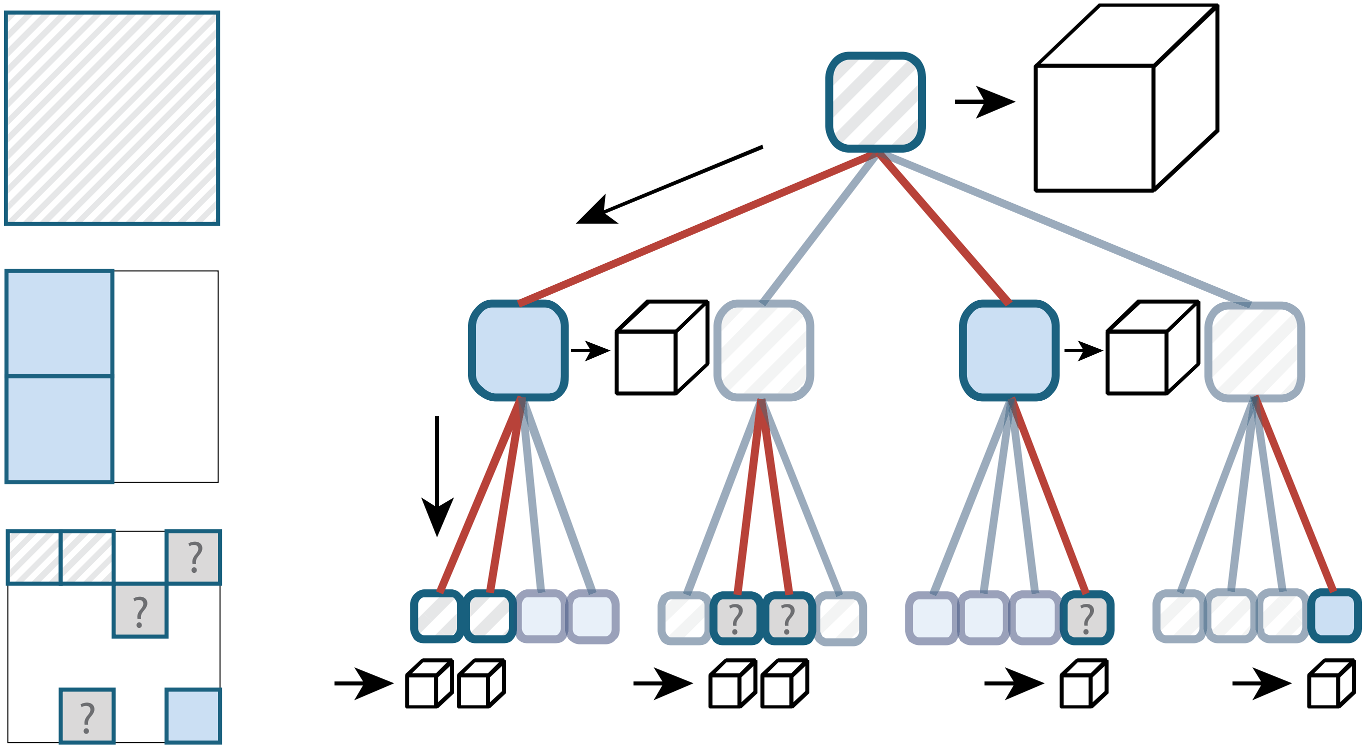
OCCUPANCY GEOMETRY

extracted geometry

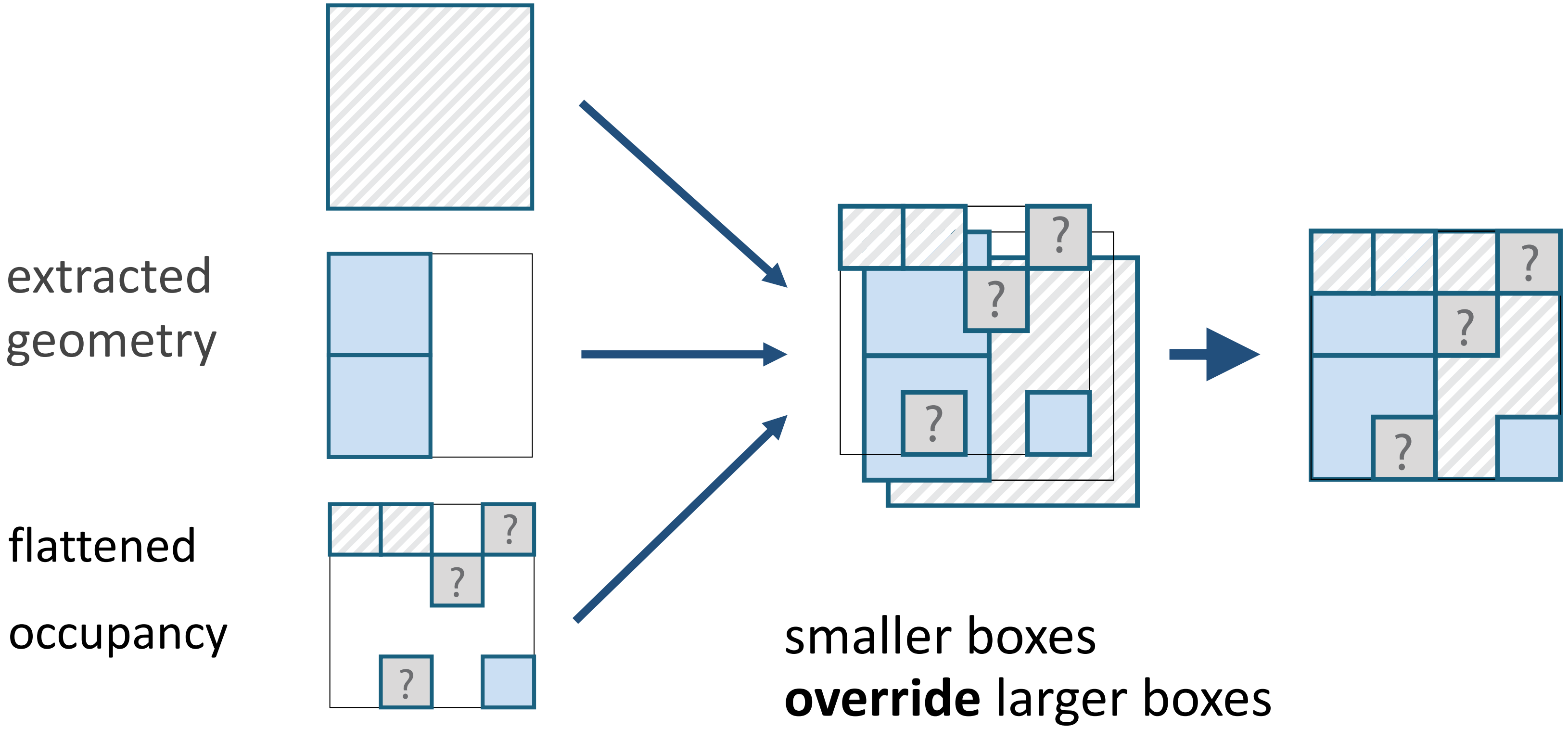


OCCUPANCY GEOMETRY

extracted geometry



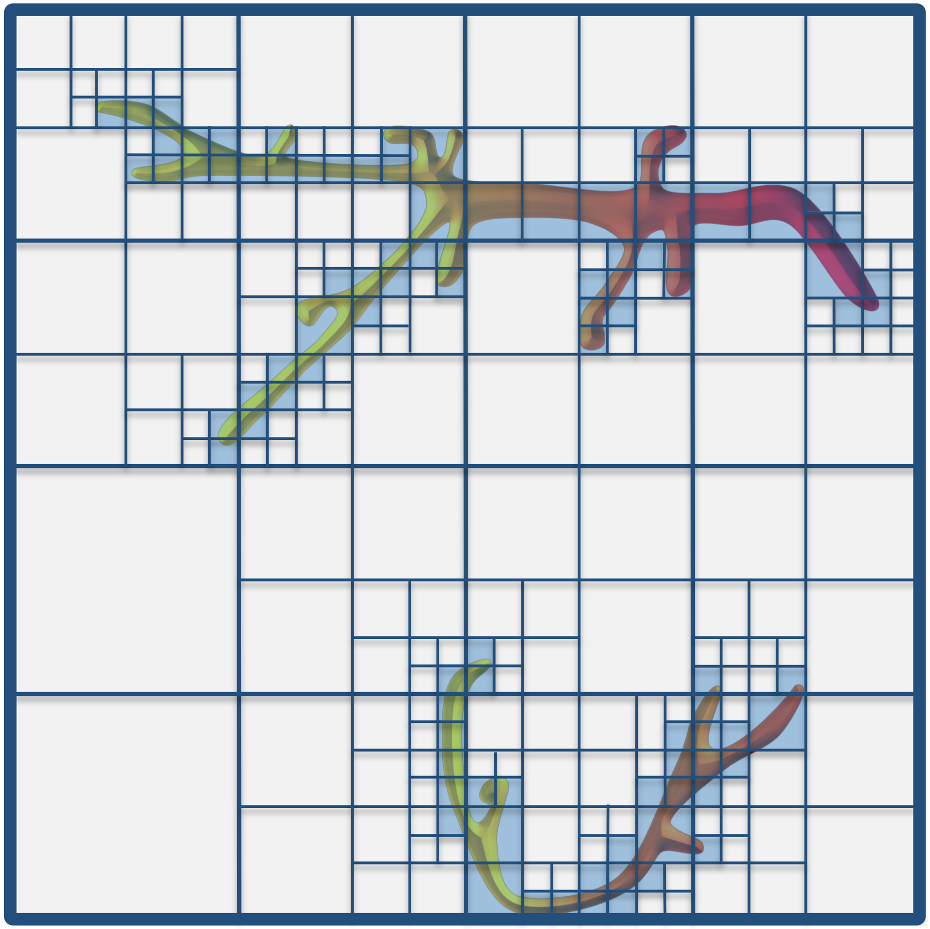
OCCUPANCY GEOMETRY



smaller boxes
override larger boxes

COMPARISON

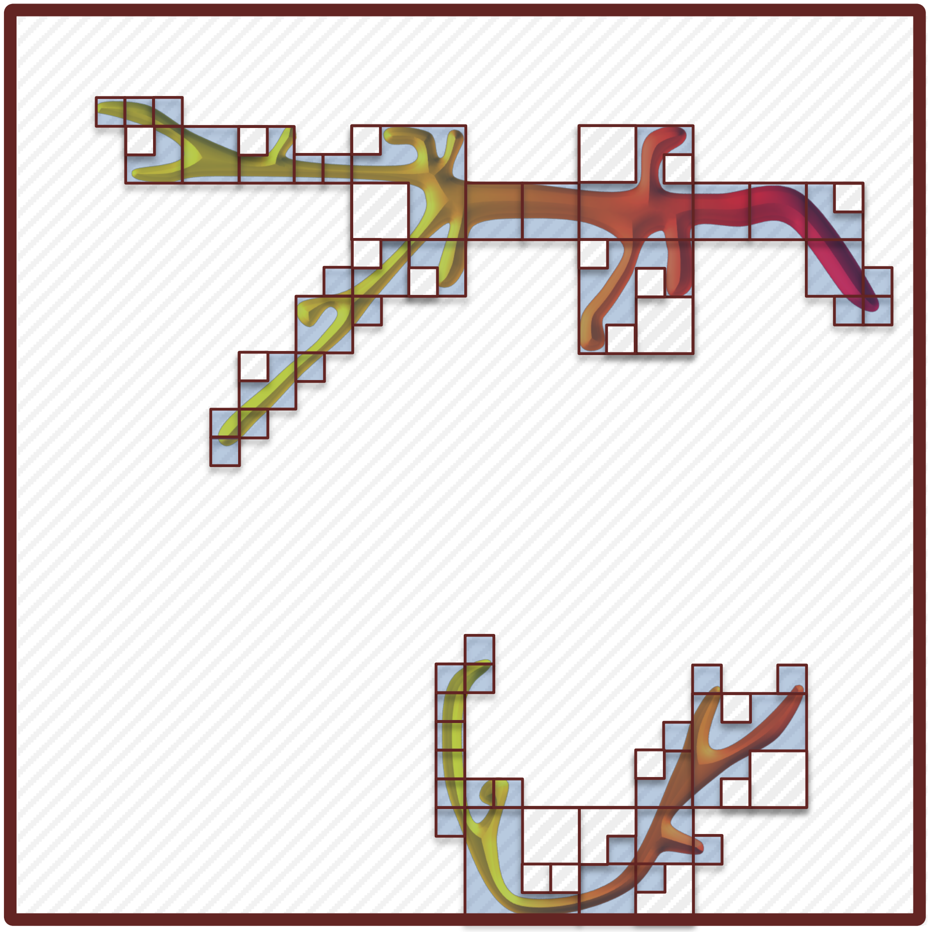
octree
subdivision



COMPARISON

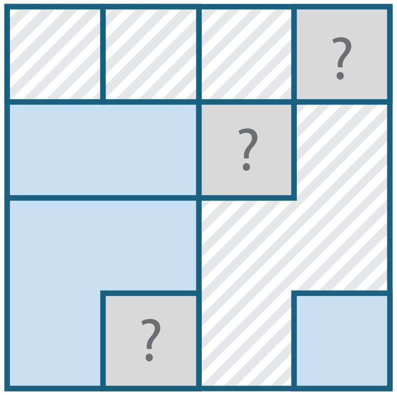
occupancy

geometry



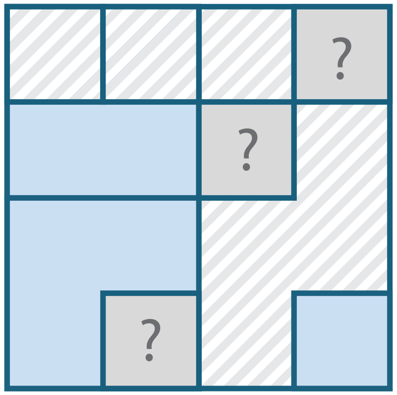
RASTERIZATION: OVERVIEW

occupancy geometry

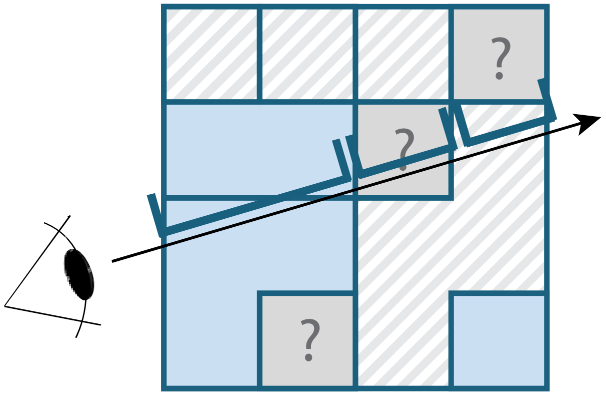


RASTERIZATION: OVERVIEW

occupancy geometry



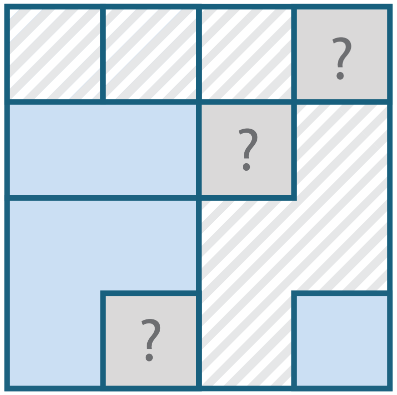
rasterize front-to-back



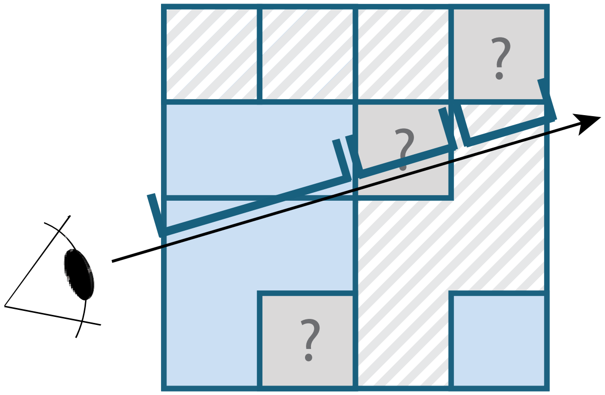
merge consecutive segments
of same occupancy class

RASTERIZATION: OVERVIEW

occupancy geometry

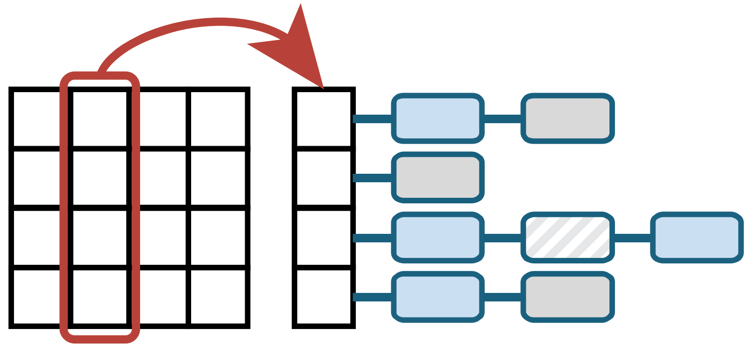


rasterize front-to-back



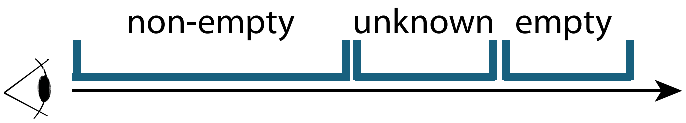
merge consecutive segments of same occupancy class

ray segment lists



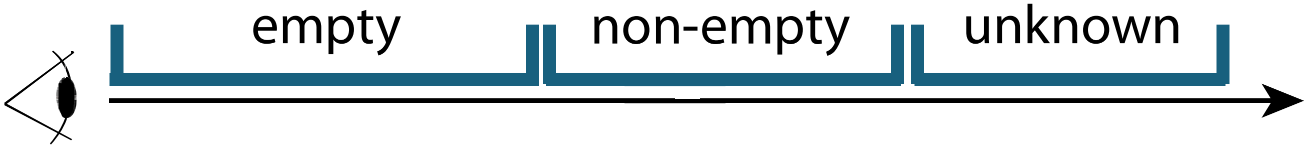
screen pixels

per-pixel linked list



RAY-CASTING

Linear traversal of ray segment list



Deferred culling for large volumes:
Occupancy class *unknown*

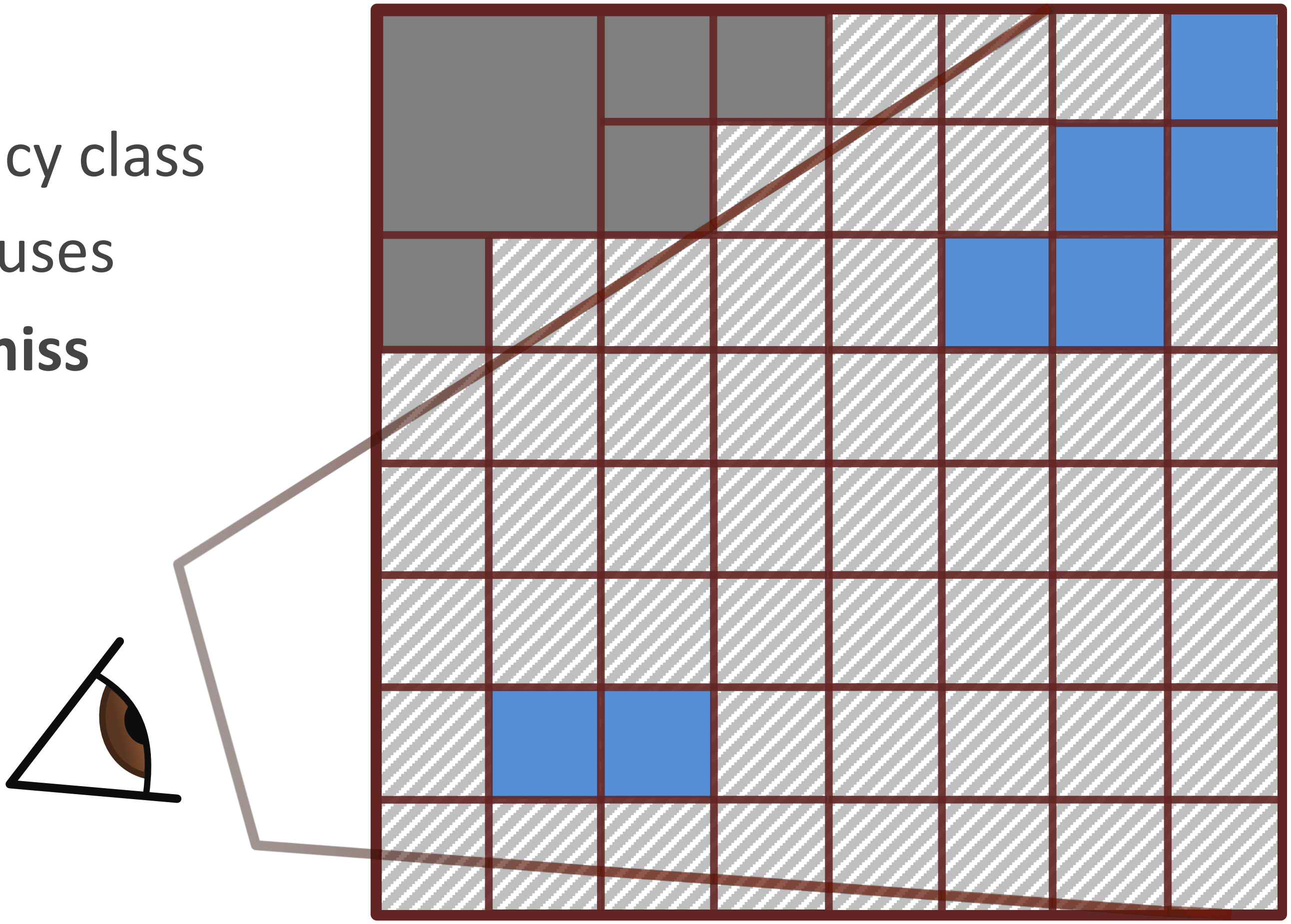
DEFERRED CULLING

The occupancy class **unknown** causes **occupancy miss**

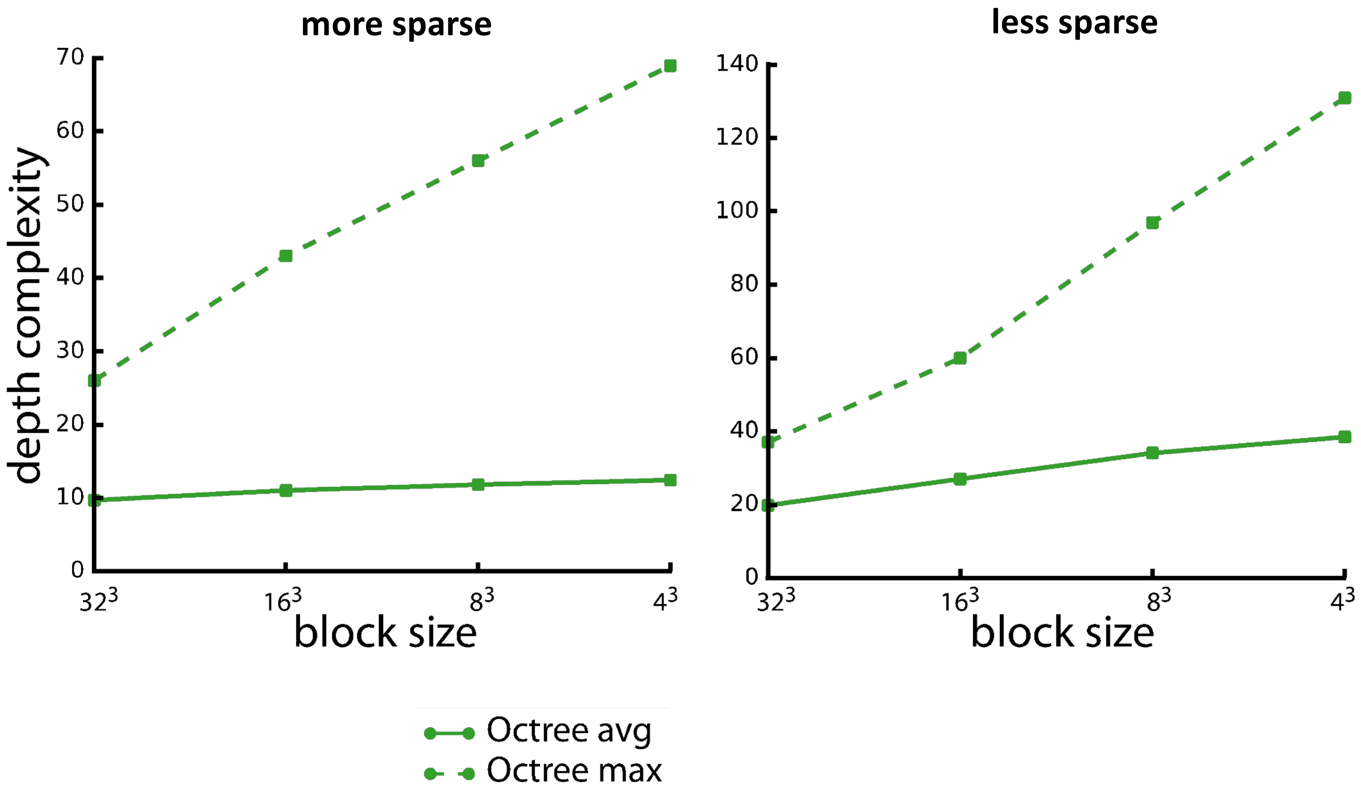
■ unknown

▨ empty

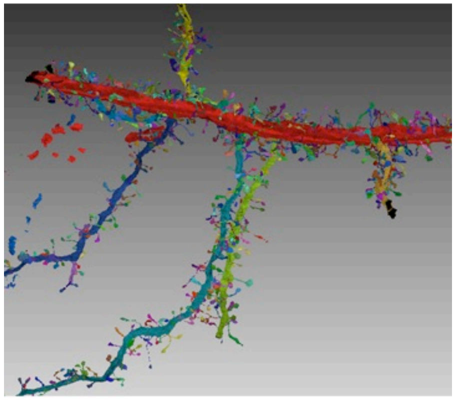
🟦 non-empty



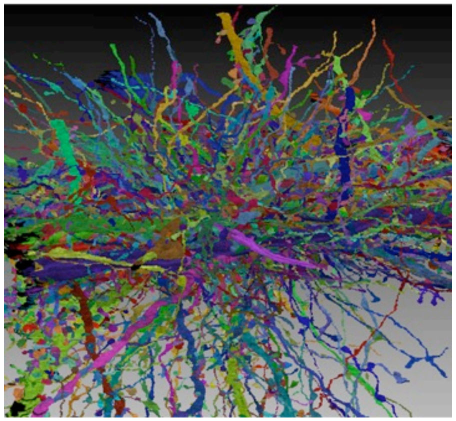
RESULTS: DEPTH COMPLEXITY



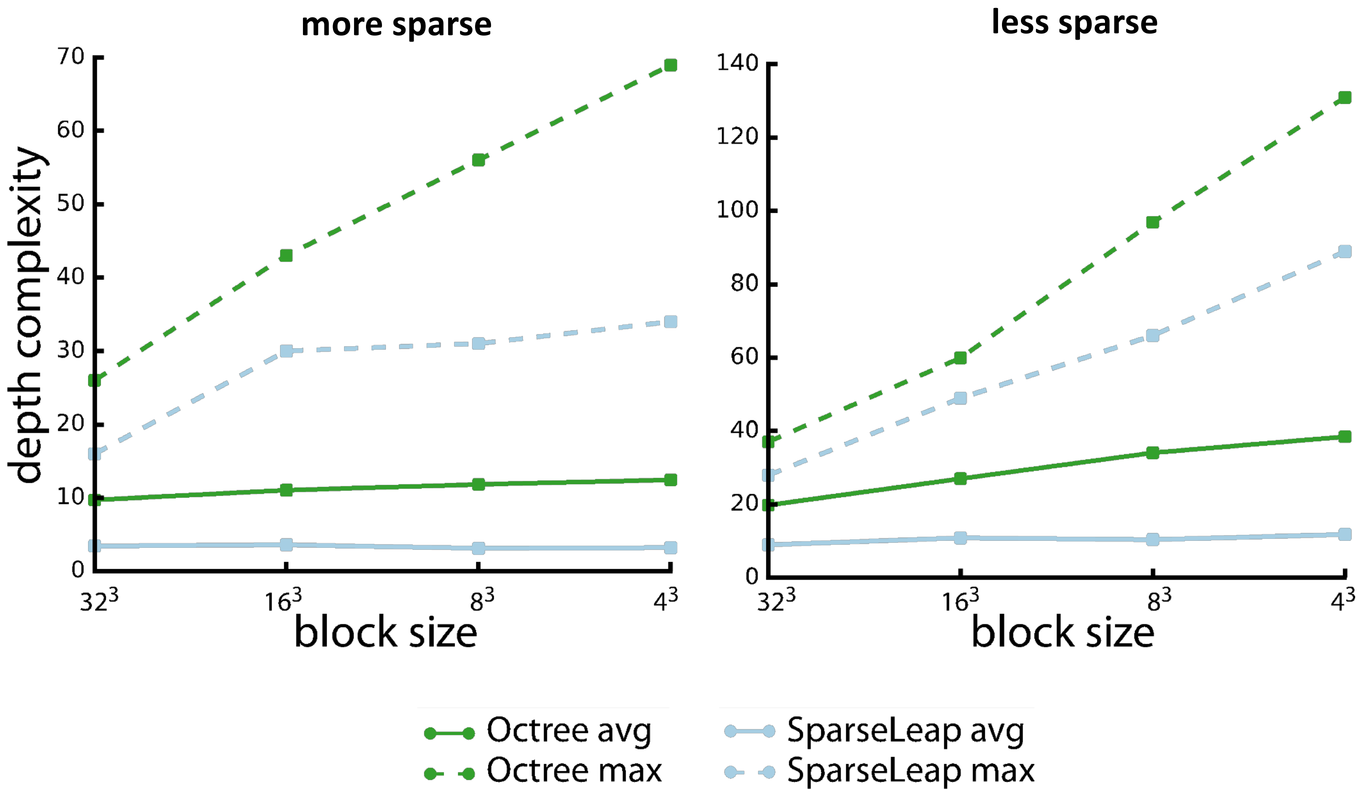
more sparse



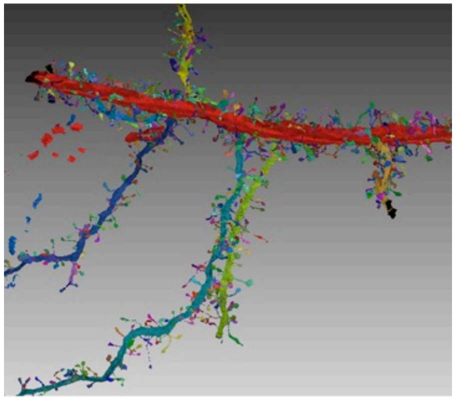
less sparse



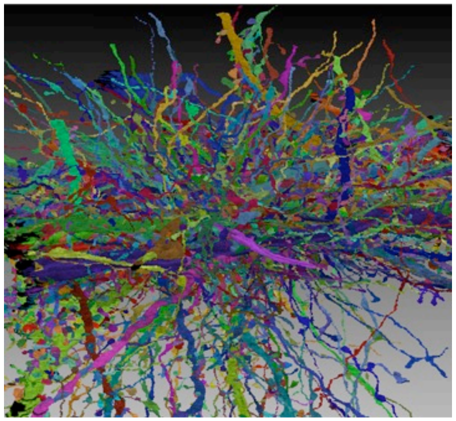
RESULTS: DEPTH COMPLEXITY



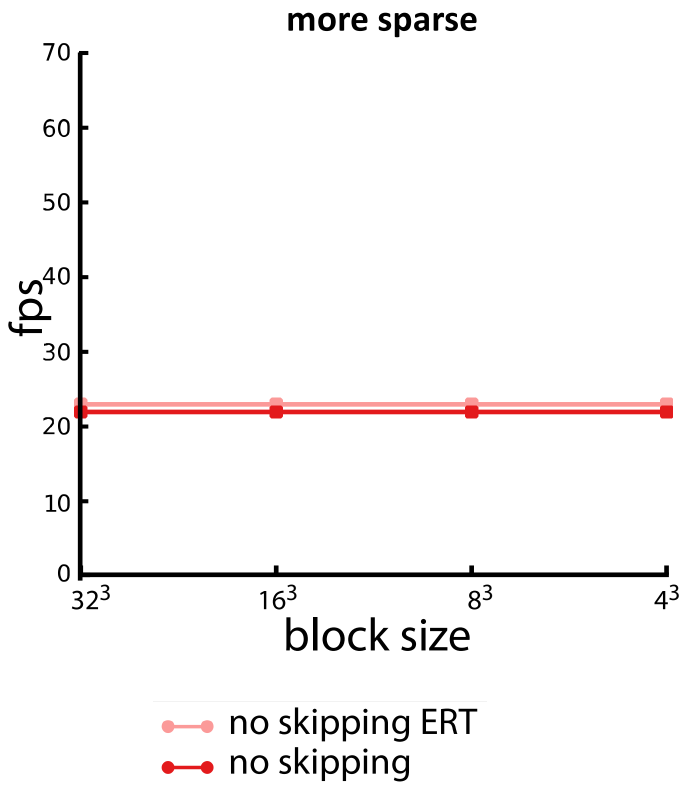
more sparse



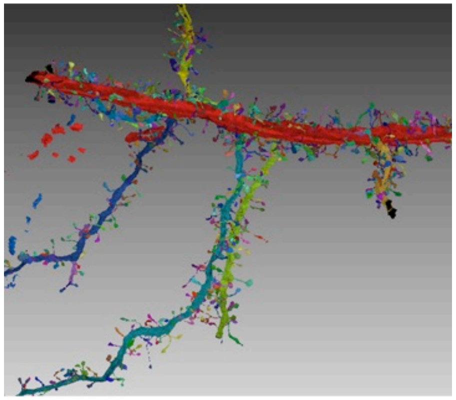
less sparse



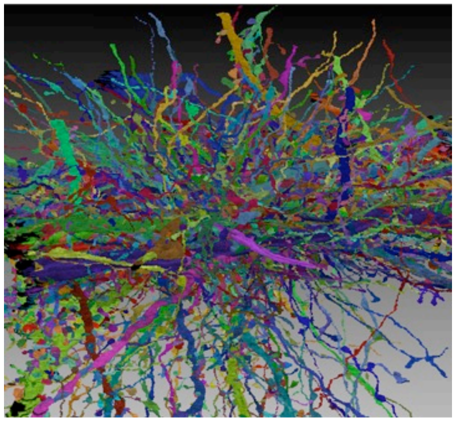
RESULTS: PERFORMANCE



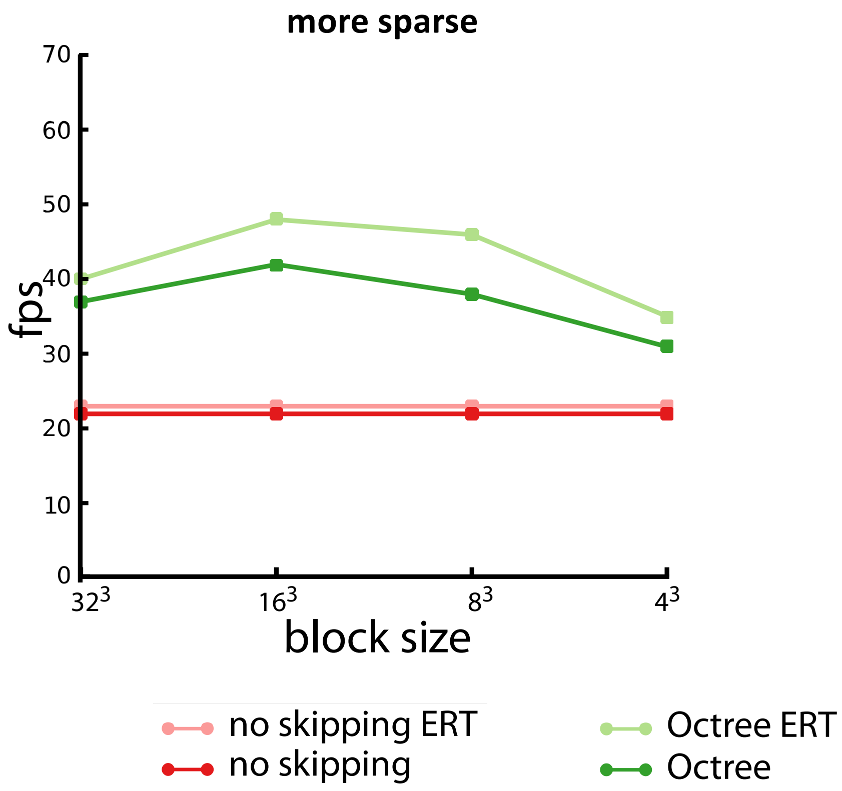
more sparse



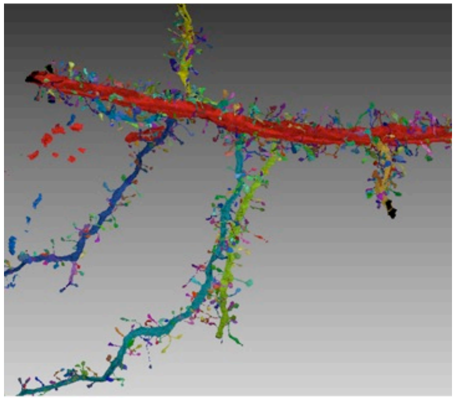
less sparse



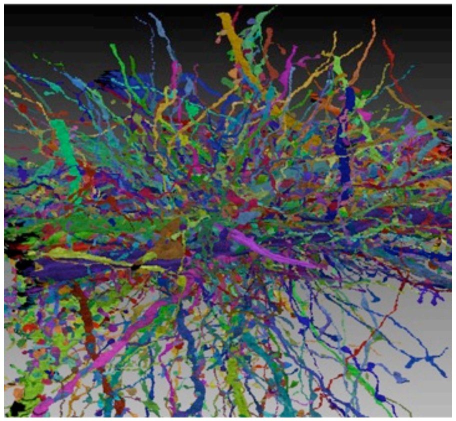
RESULTS: PERFORMANCE



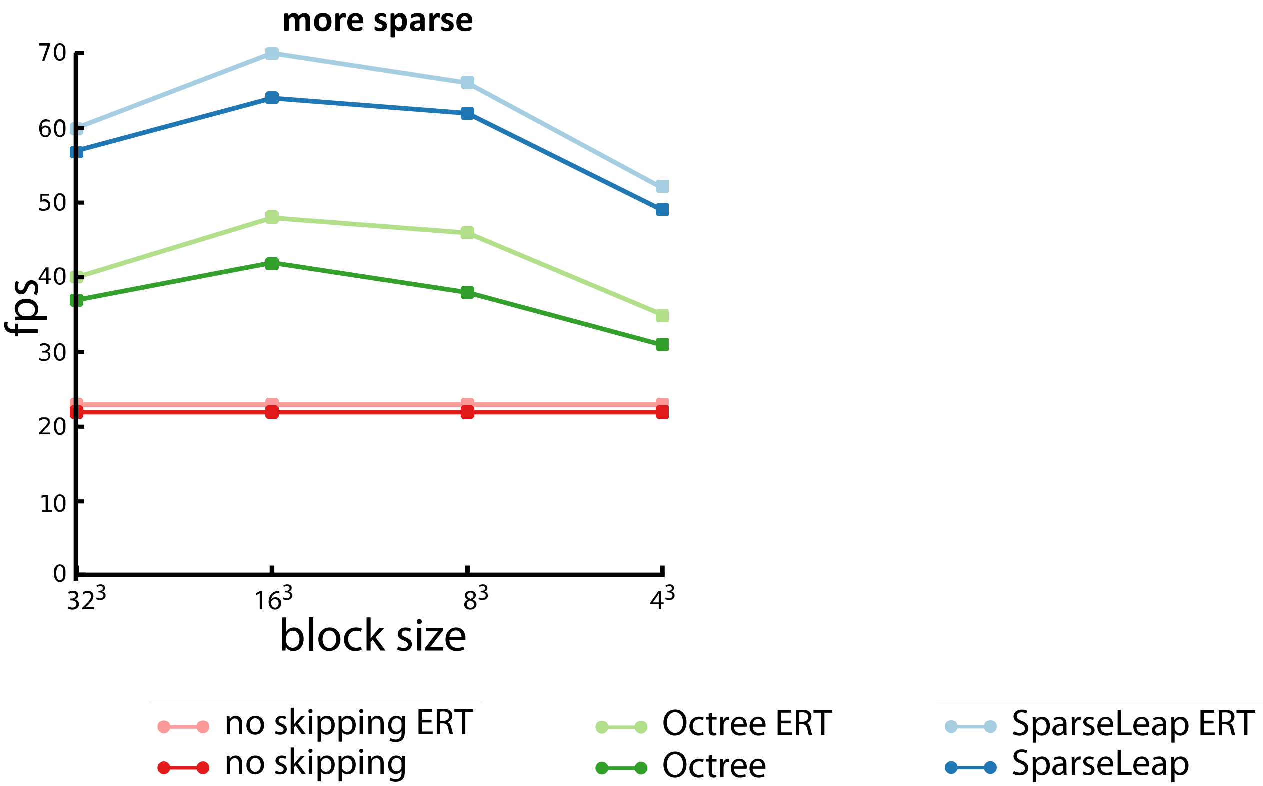
more sparse



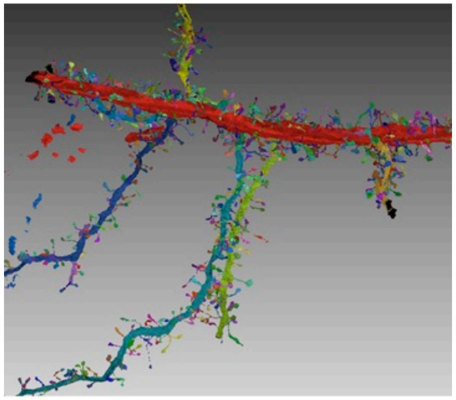
less sparse



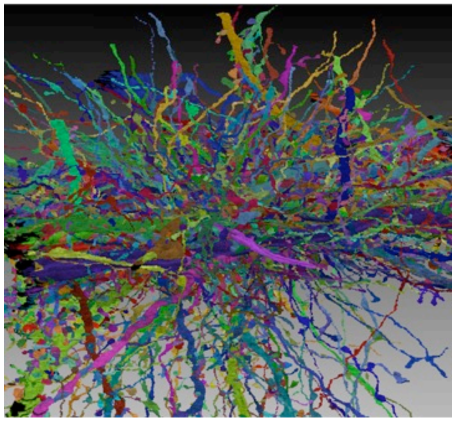
RESULTS: PERFORMANCE



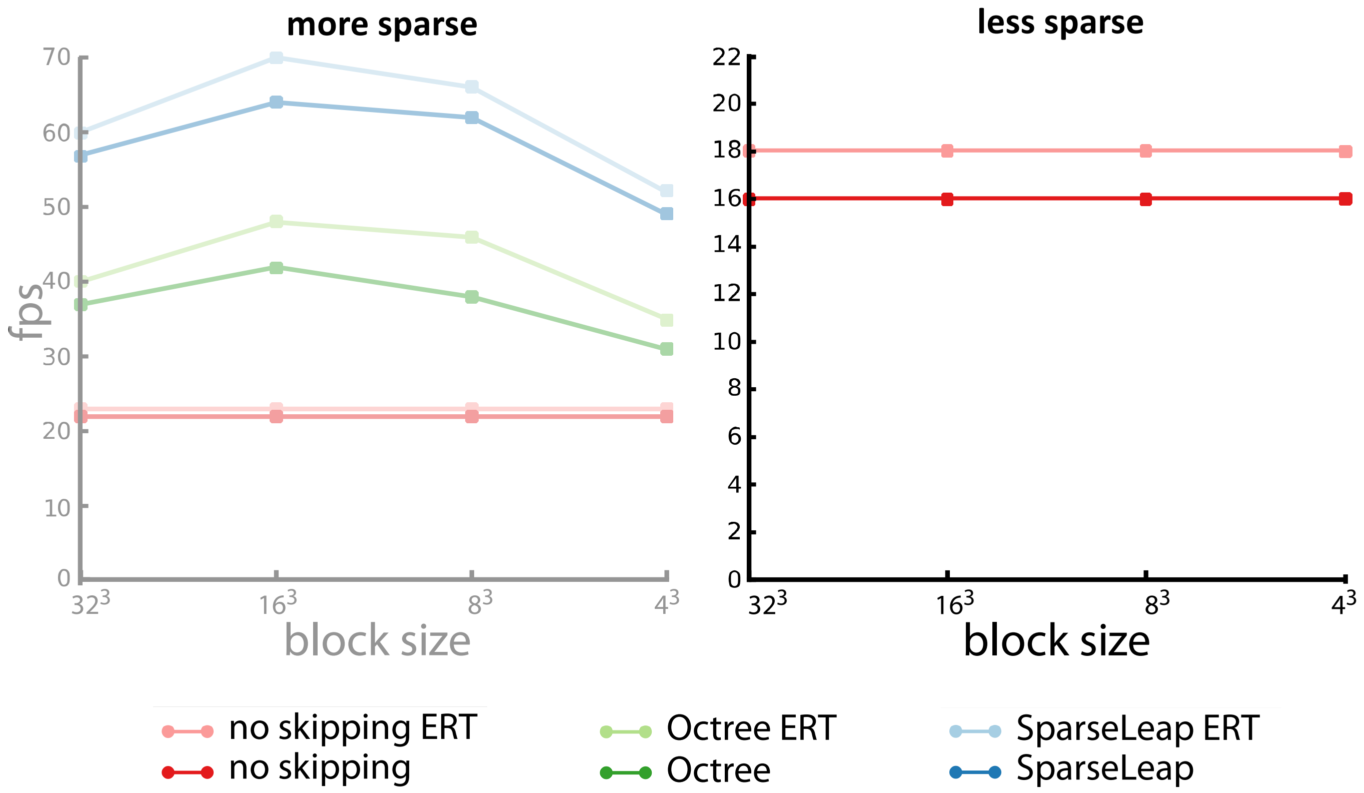
more sparse



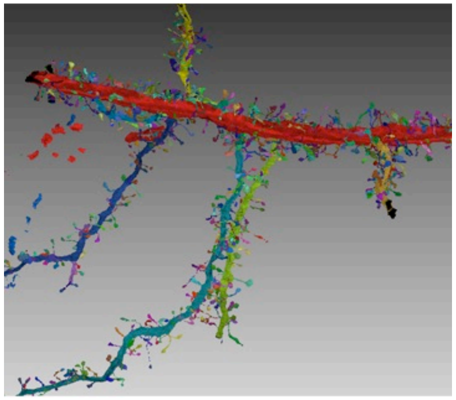
less sparse



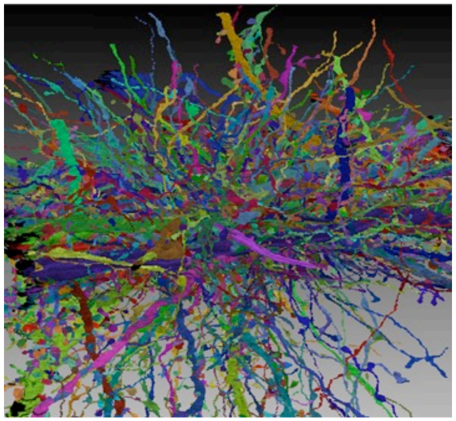
RESULTS: PERFORMANCE



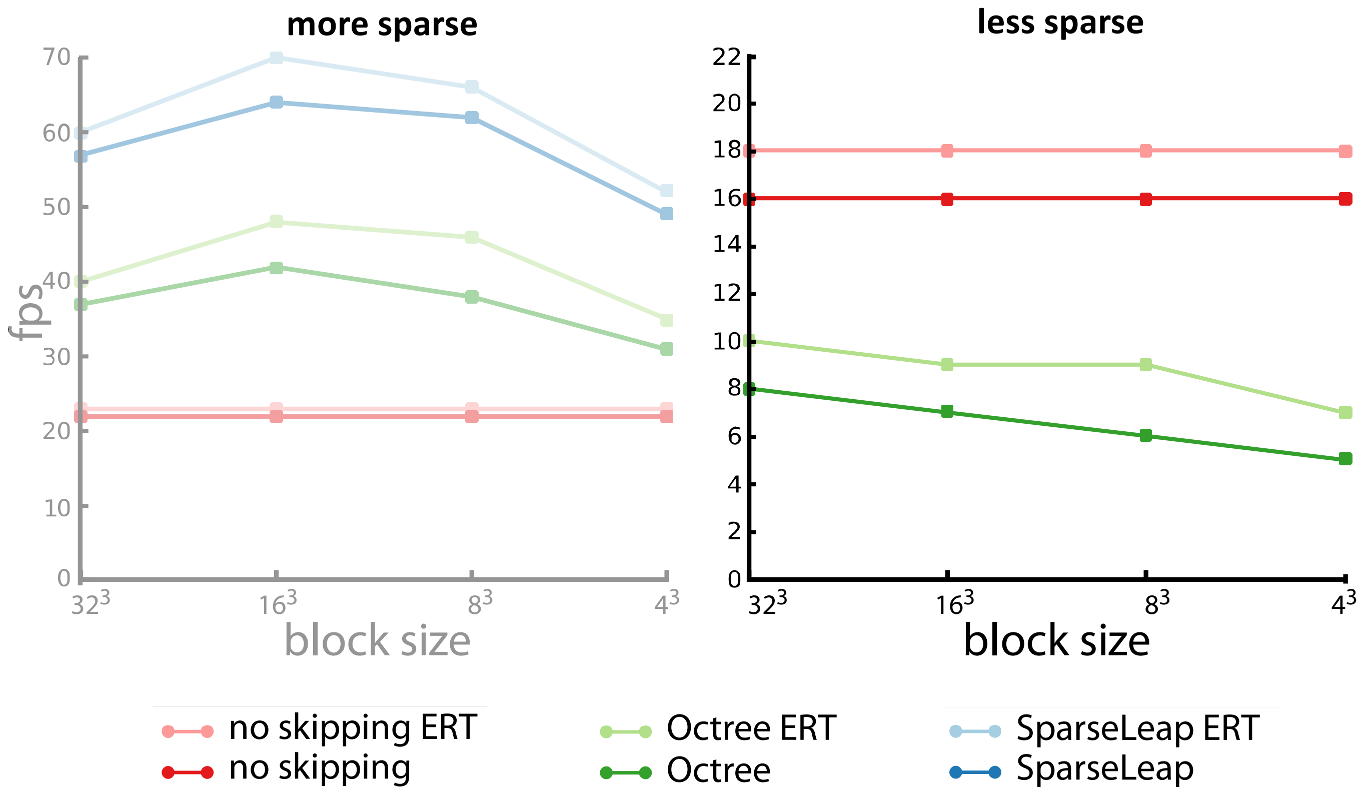
more sparse



less sparse

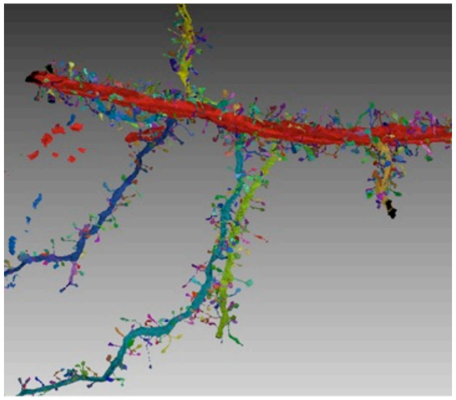


RESULTS: PERFORMANCE

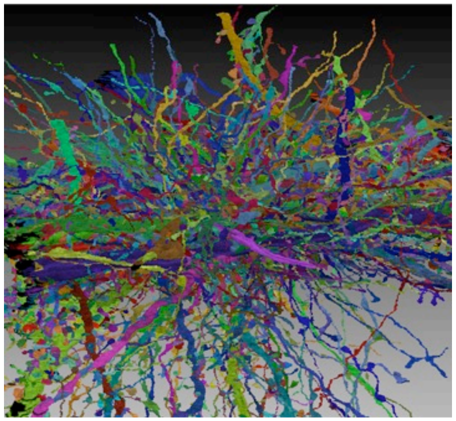

more sparse

less sparse

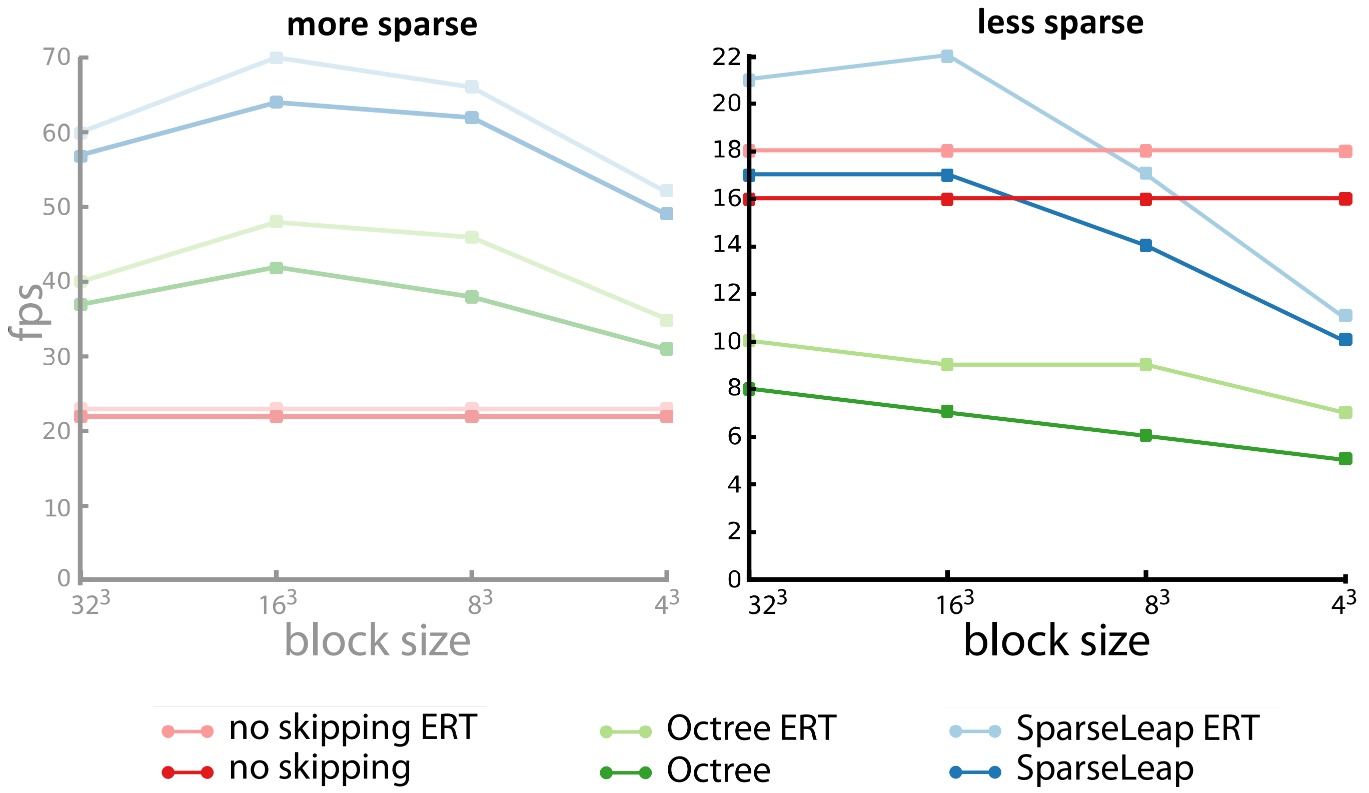
more sparse



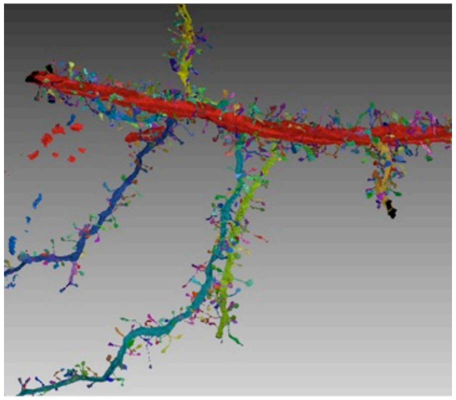
less sparse



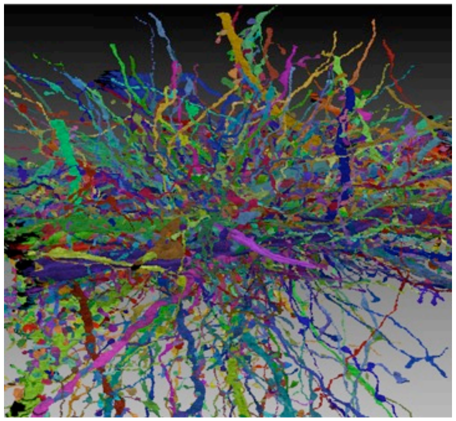
RESULTS: PERFORMANCE



more sparse

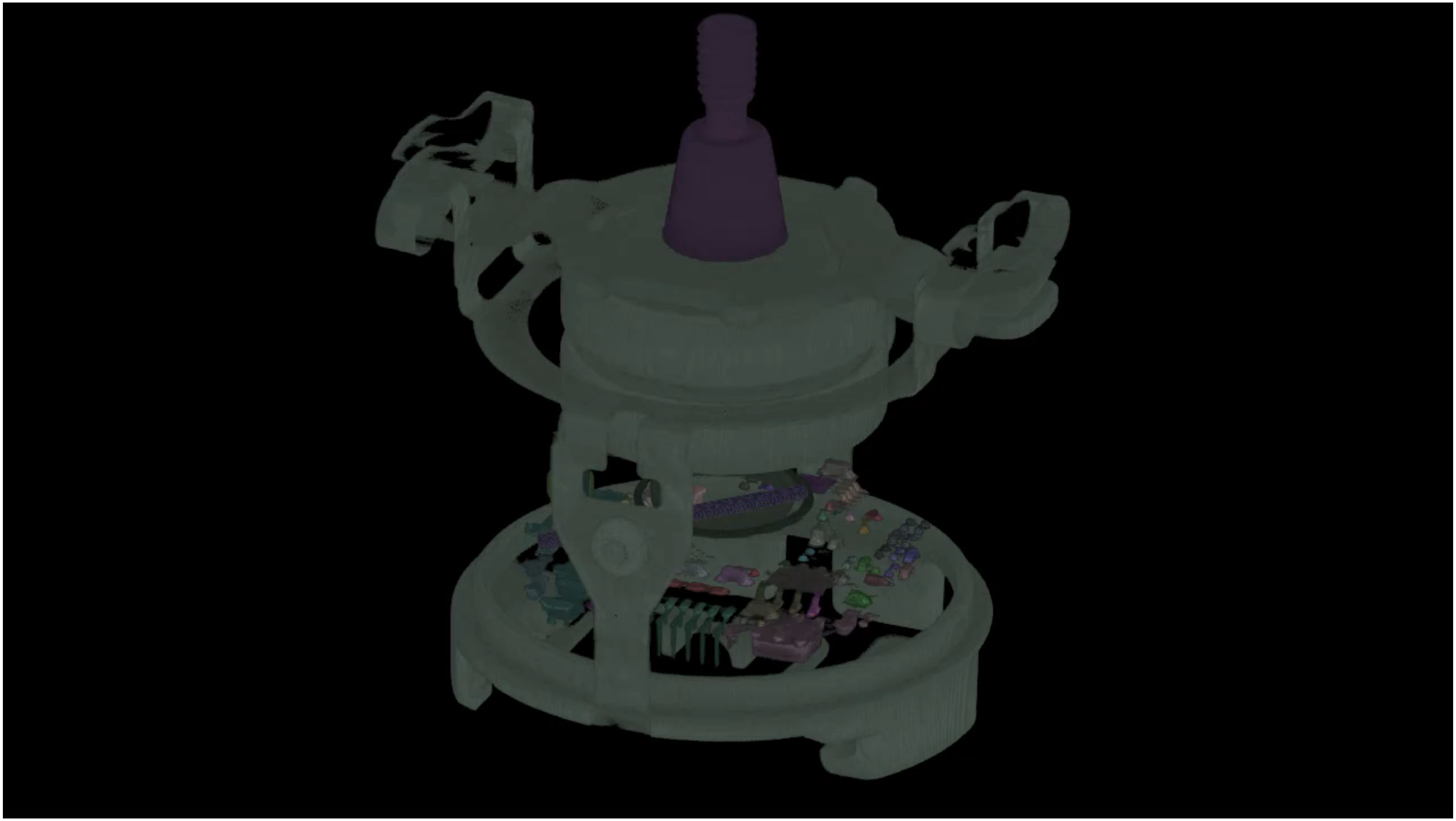


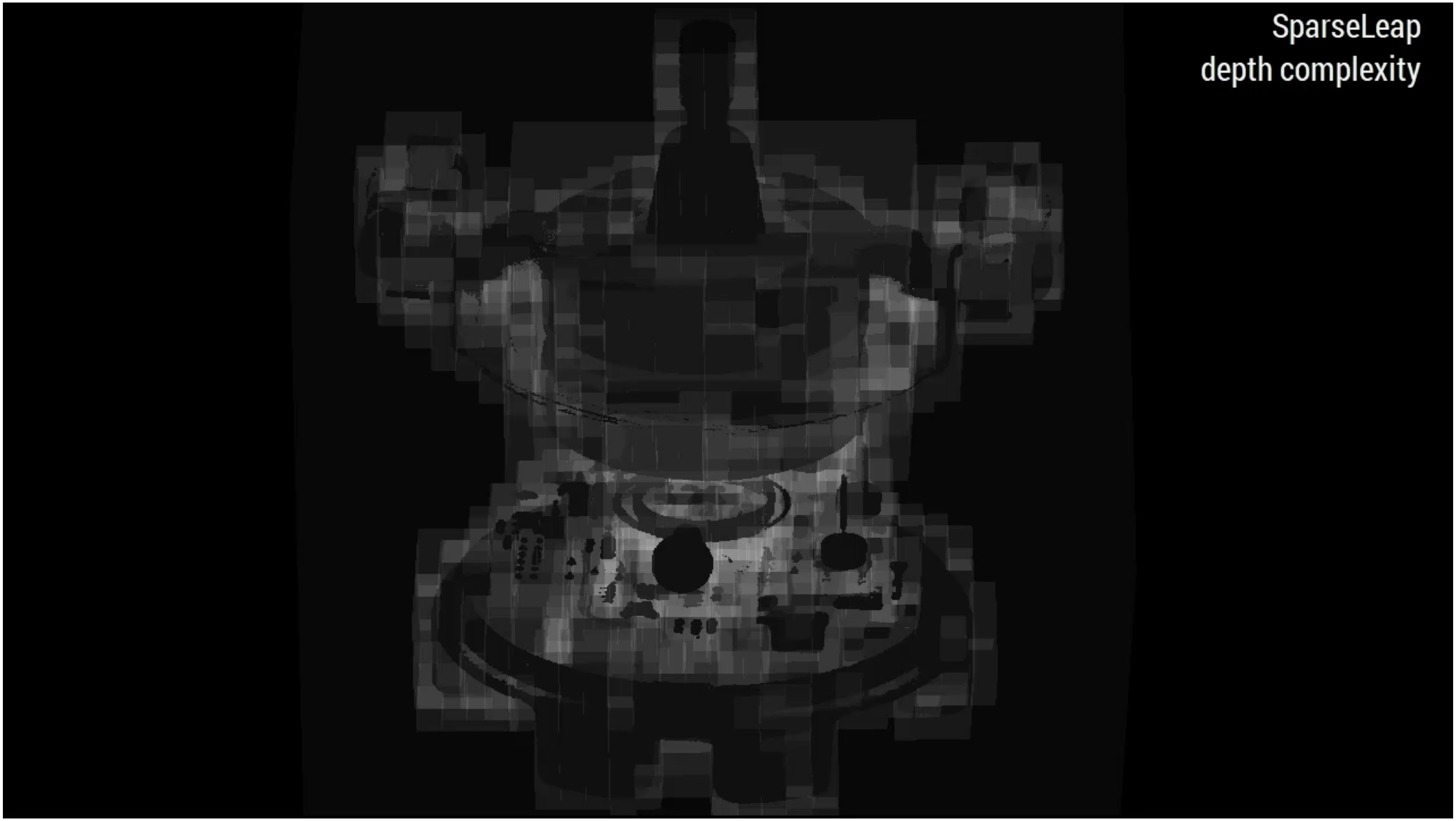
less sparse



Dreh Sensor data set: 2,048 × 2,048 × 2,048

85 segmented objects




SparseLeap
depth complexity

SUMMARY

Cost of empty space skipping moved out of ray-casting loop

Attractive alternative for complex volumes

Memory consumption (GPU)

- Occupancy geometry: very low; much lower than octree storage
- Lists: depends on screen resolution and average depth complexity

SIGGRAPH
ASIA 2018
TOKYO

Scalable Culling for Large Segmentation Volumes

LARGE SEGMENTATION VOLUMES

Raw image volume

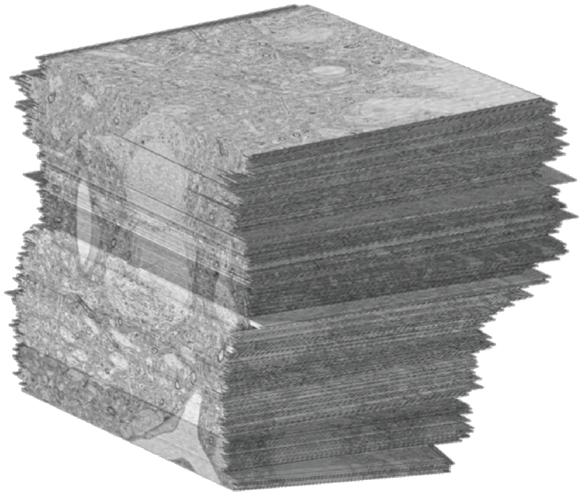
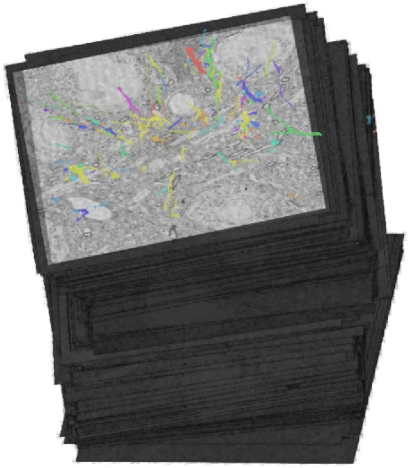
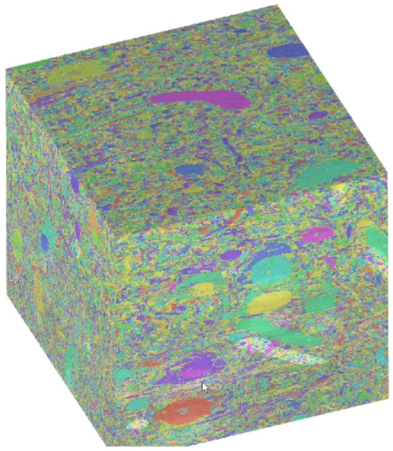


Image + Label volumes



SEM Mouse Cortex
21,494 x 25,790 x 1,850
4,125 labels

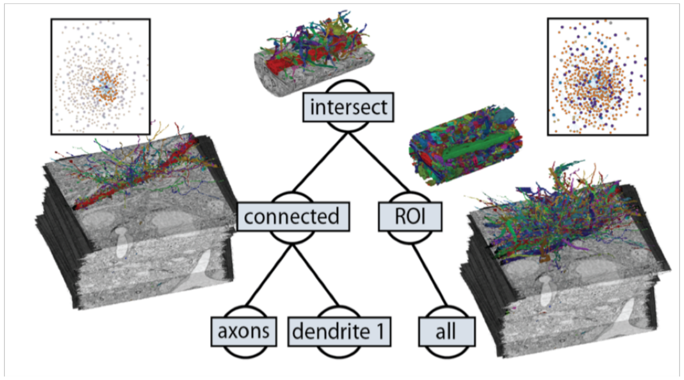


Mouse Cortex 2
4,096 x 4,096 x 4,096
16,77M labels

MOTIVATION – INTERACTIVE VIS APPLICATIONS

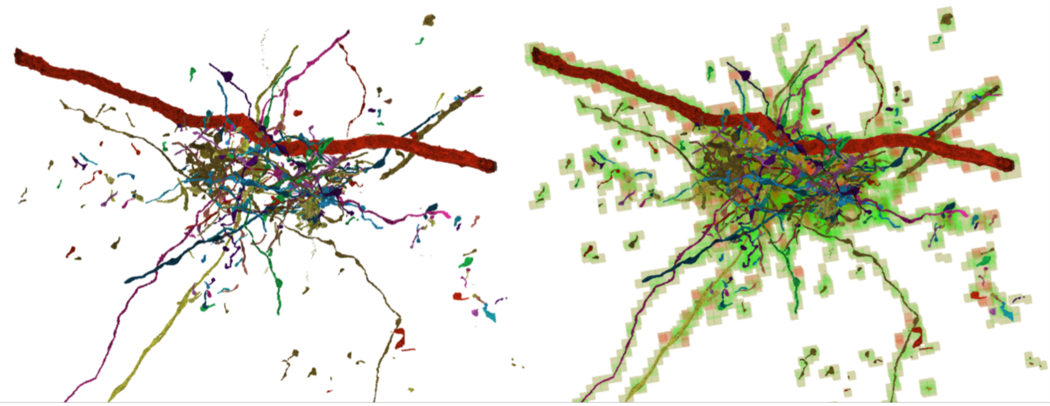
Visual Queries

[ConnectomeExplorer.
Beyer et al., SciVis 2013]

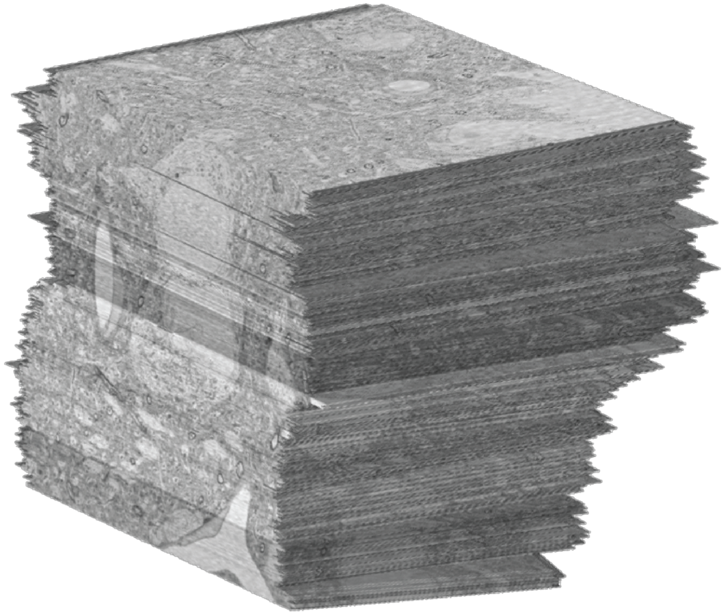


Fast Volume Rendering

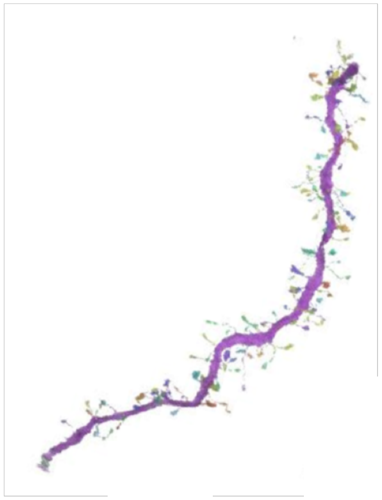
[SparseLeap.
Hadwiger et al., SciVis 2018]



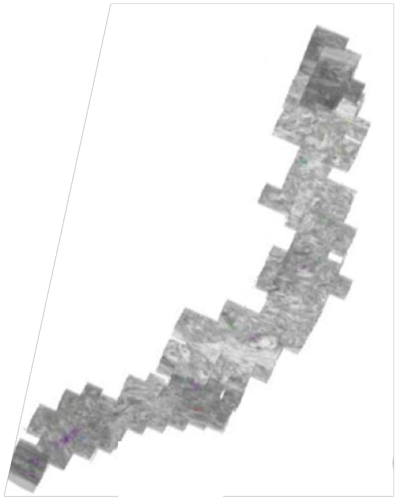
EXAMPLE: CULLING FOR EMPTY SPACE SKIPPING



Raw image volume



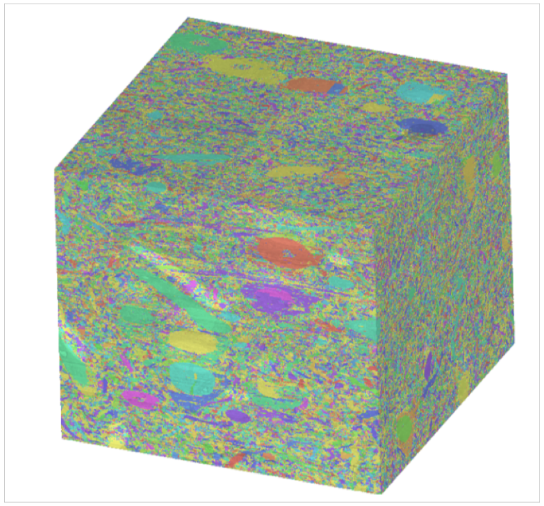
Single label within volume



Volume blocks after culling (<0.1% of volume blocks)

CHALLENGES

Large label volumes stored as up to 64-bit integer data.



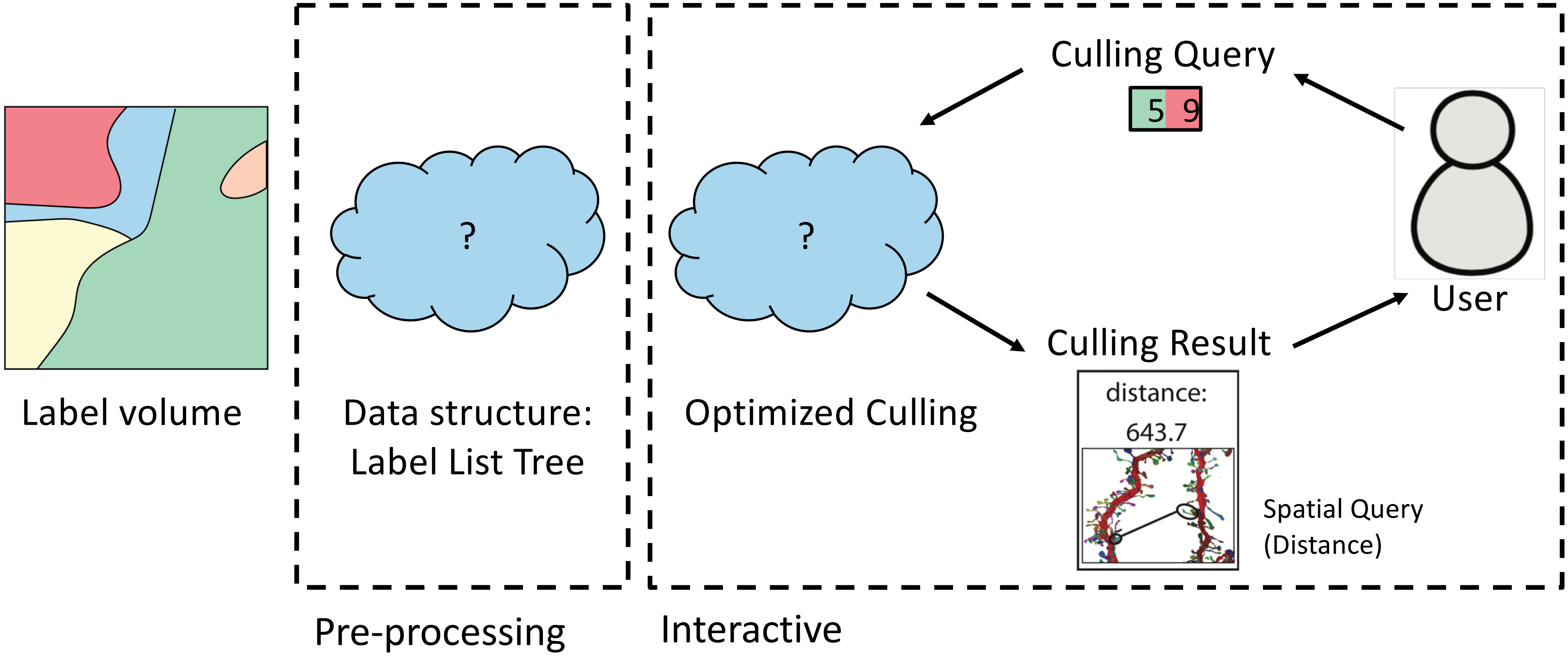
> 250 GB

discrete labels

> 13 million labels

(24 bit data)

OUR APPROACH FOR SCALABLE CULLING



Label volume

Data structure:
Label List Tree

Optimized Culling

Culling Query

User

Culling Result

distance:
643.7

Spatial Query
(Distance)

Pre-processing

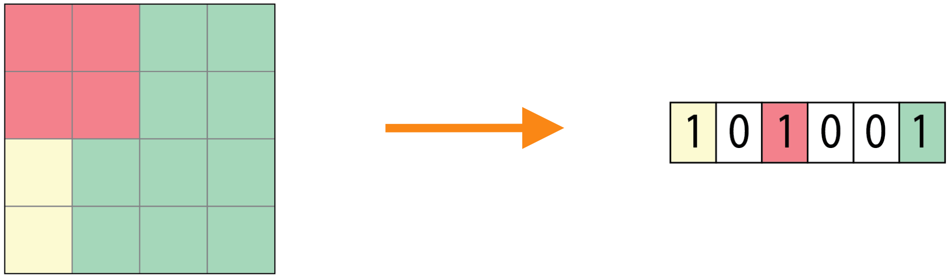
Interactive

SIGGRAPH ASIA 2018 TOKYO

Data Structure: Label List Tree

LABEL LISTS

- Which labels are present in a volume block?
- Store a list (or set) of labels per volume block



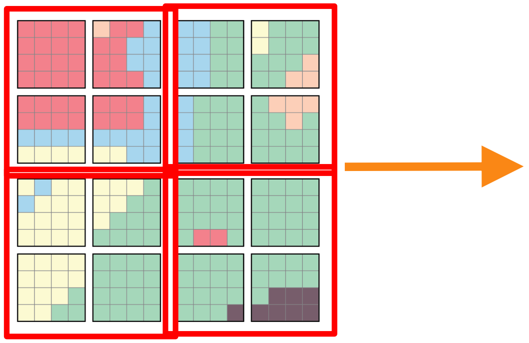
1	0	1	0	0	1

Volume Block

Label List

LABEL LISTS

Label Volume



0 0 1 0 0 0

0 1 1 1 0 0

1 0 1 1 0 0

1 0 1 1 0 0

1 0 0 1 0 0

1 0 0 0 0 1

1 0 0 0 0 1

0 0 0 0 0 1

0 0 0 1 0 1

1 1 0 0 0 1

0 0 0 1 0 1

0 1 0 0 0 1

0 0 1 0 0 1

0 0 0 0 0 1

0 0 0 0 1 1

0 0 0 0 1 1

HYBRID LABEL LIST ENCODING

	Data Structure	Data Access Time	Culling
Deterministic	Roaring Bitmap [1]	Logarithmic	Exact
Probabilistic	Bloom Filter [2]	Constant	Conservative

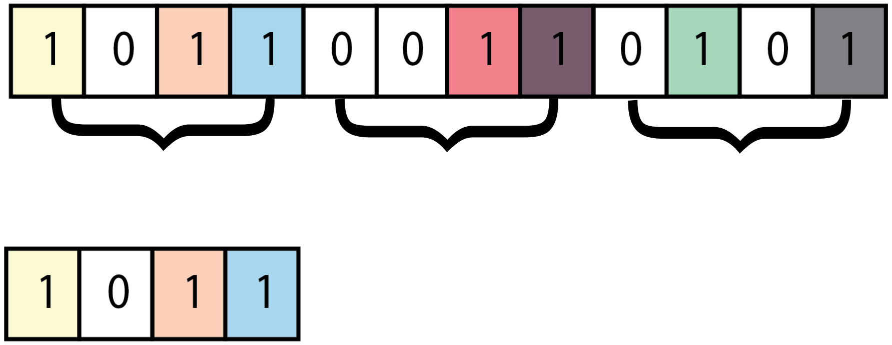
Best representation chosen based on:

- Memory size
- Expected run time query performance
- User preferences

[1] Better bitmap performance with roaring bitmaps. Chambi et al., 2016.

[2] Space/time trade-offs in hash coding with allowable errors. Bloom, 1970.

LABEL LIST ENCODING - DETERMINISTIC



Bit string

1	0	1	1	0	0	1	1	0	1	0	1

buckets

Roaring bitmaps

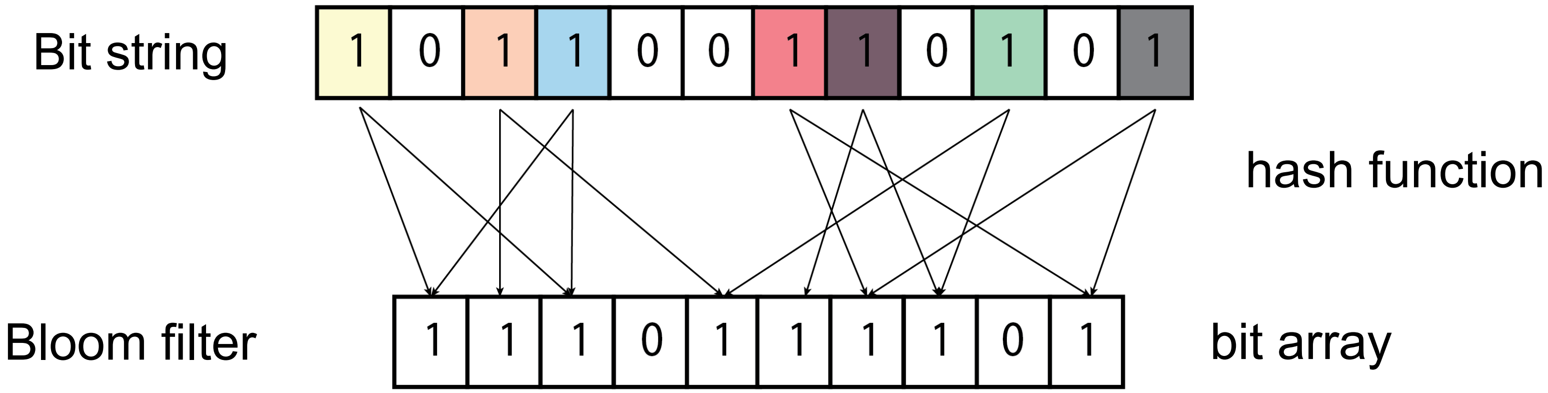
1	0	1	1

Bitmap
(dense)

RLE
(runs)

Sorted list
(sparse)

Label List Encoding - Probabilistic



Bit string

1	0	1	1	0	0	1	1	0	1	0	1

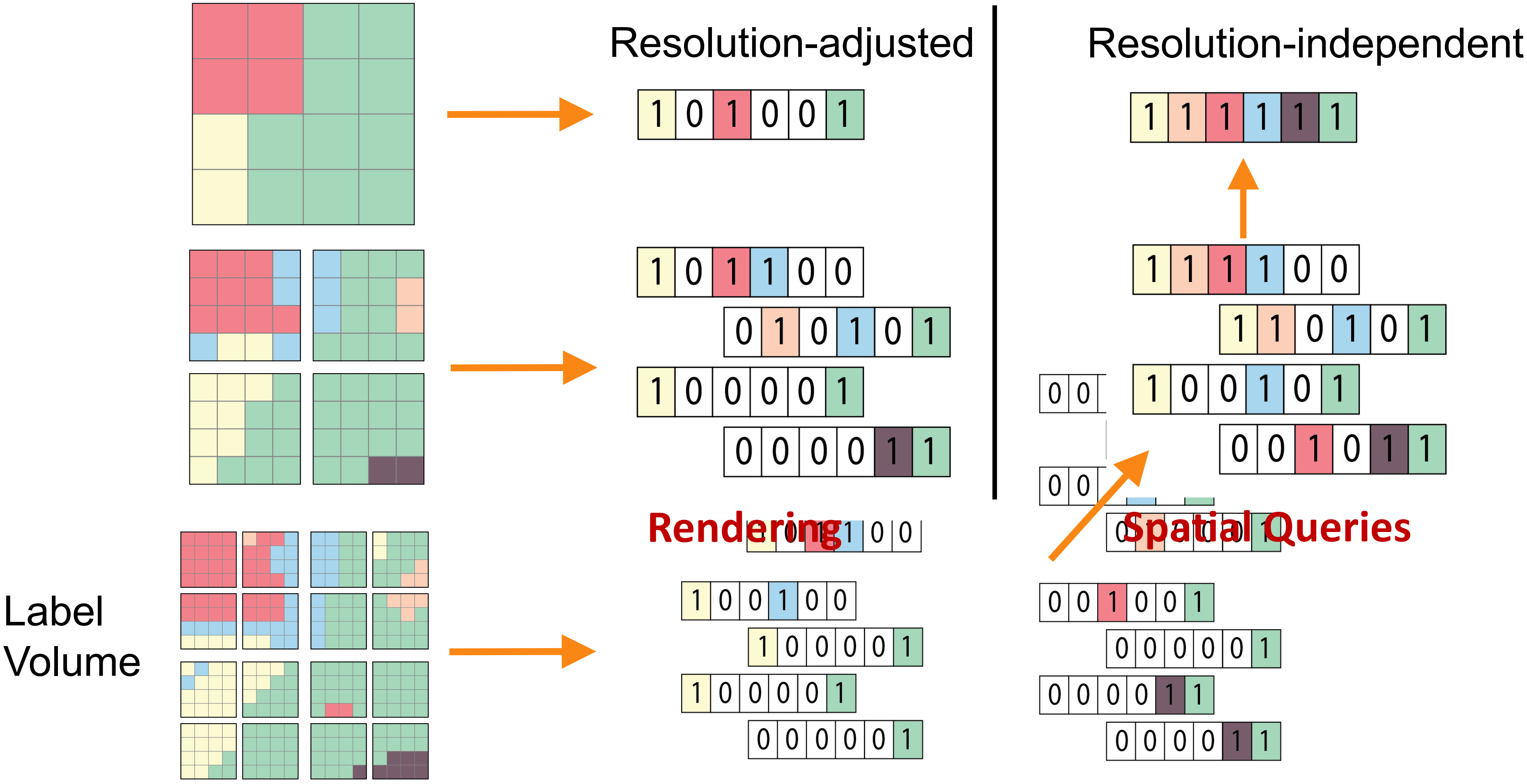
hash function

Bloom filter

1	1	1	0	1	1	1	1	0	1

bit array

MULTI-RESOLUTION LABEL LIST TREE



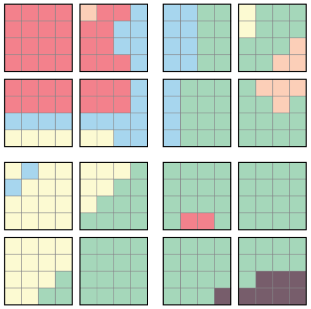
Optimized Culling

CULLING

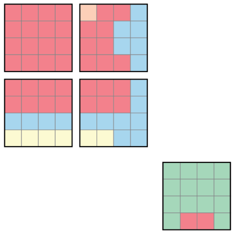
- Culling input: Culling Query, set of labels we are interested in
- Culling output: List of volume blocks that contain labels from query

0	0	1	0	0	0

Culling query

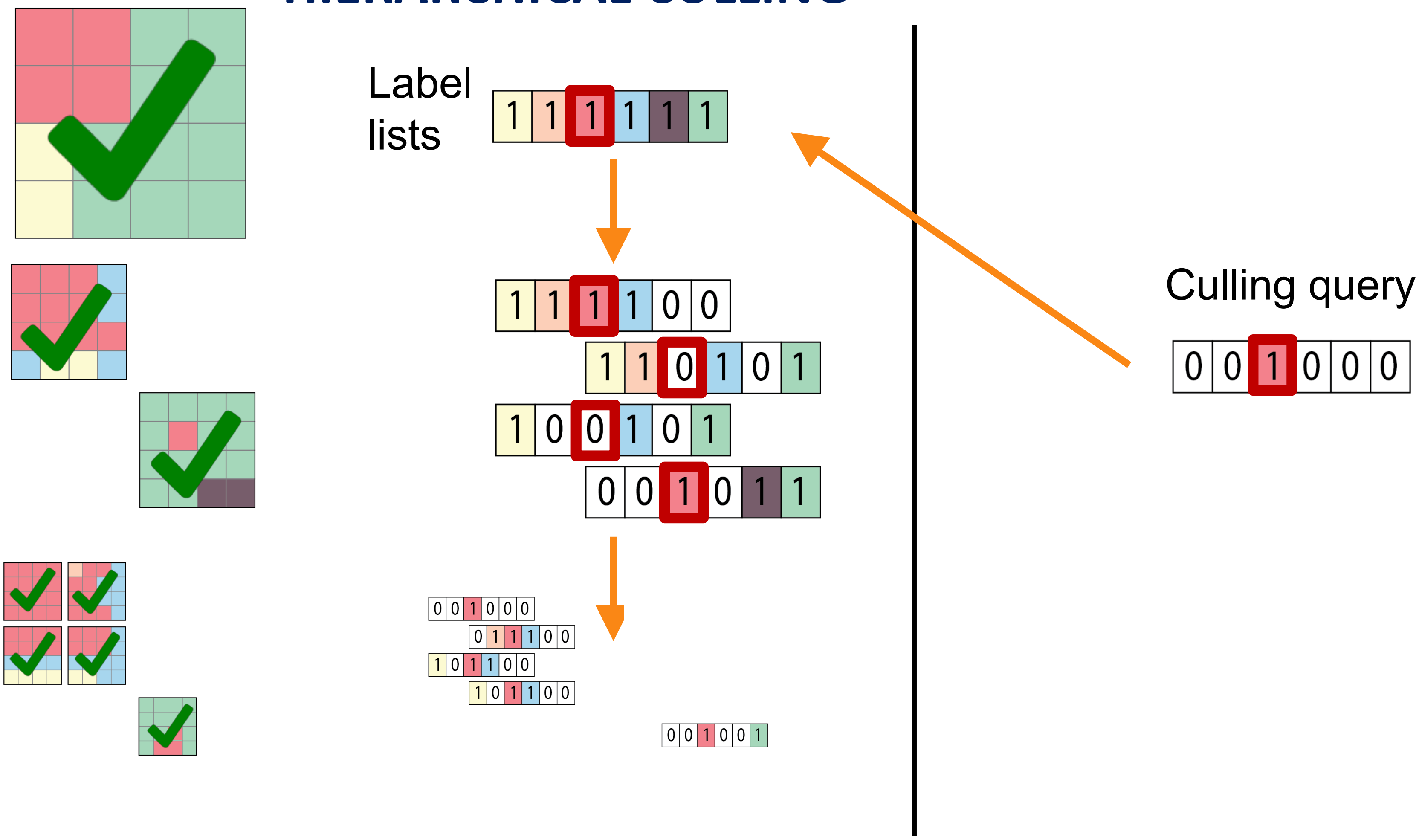


Volume blocks



Culling result

HIERARCHICAL CULLING



Label lists

1 1 1 1 1 1

1 1 1 1 0 0

1 1 0 1 0 1

1 0 0 1 0 1

0 0 1 0 1 1

0 0 1 0 0 0

0 1 1 1 0 0

1 0 1 1 0 0

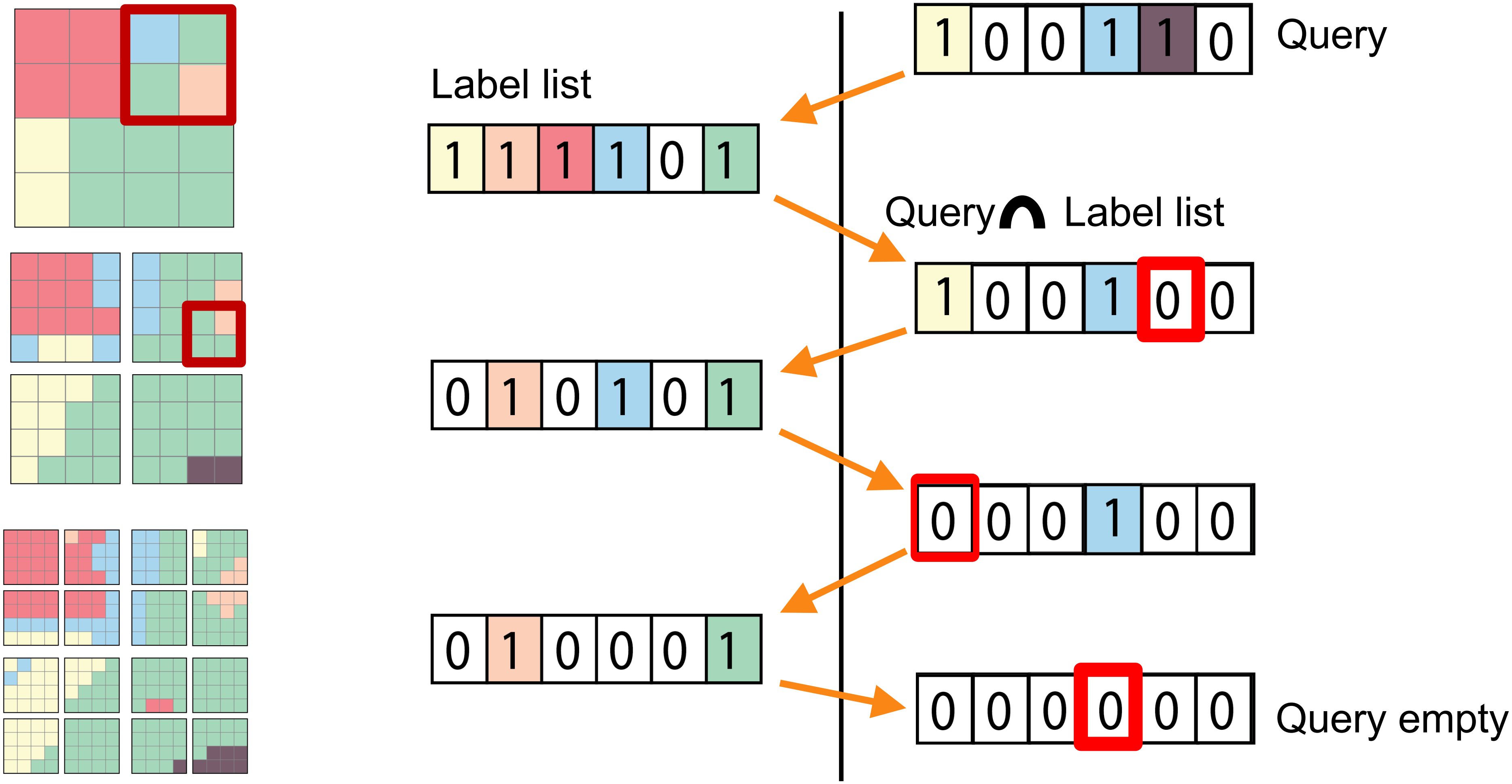
1 0 1 1 0 0

0 0 1 0 0 1

Culling query

0 0 1 0 0 0

HIERARCHICAL QUERY PRUNING



Query

1 0 0 1 1 0

Label list

1 1 1 1 0 1

Query ∩ Label list

1 0 0 1 0 0

0 1 0 1 0 1

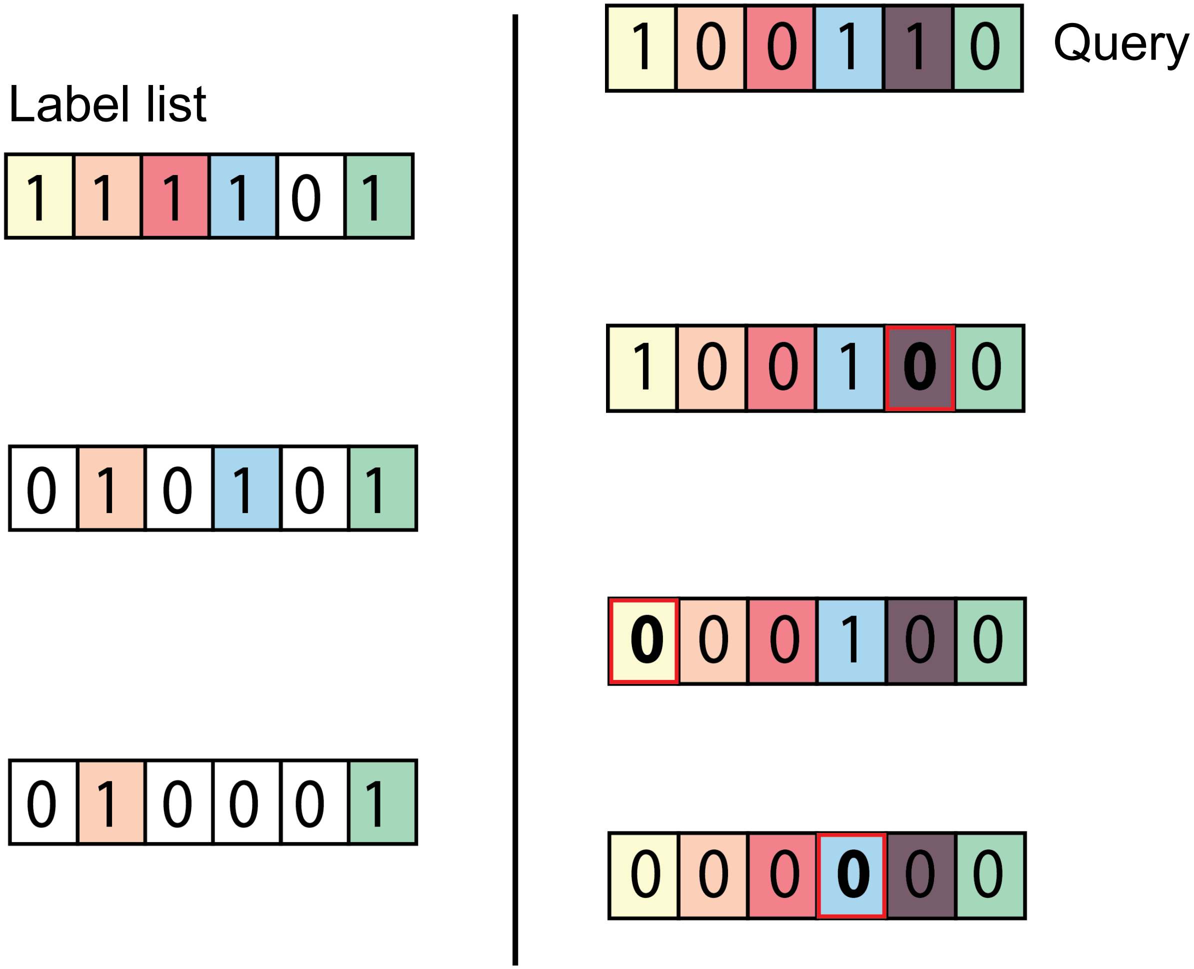
0 0 0 1 0 0

0 1 0 0 0 1

0 0 0 0 0 0

Query empty

HIERARCHICAL QUERY PRUNING



Label list

1 1 1 1 0 1

0 1 0 1 0 1

0 1 0 0 0 1

1 0 0 1 1 0 Query

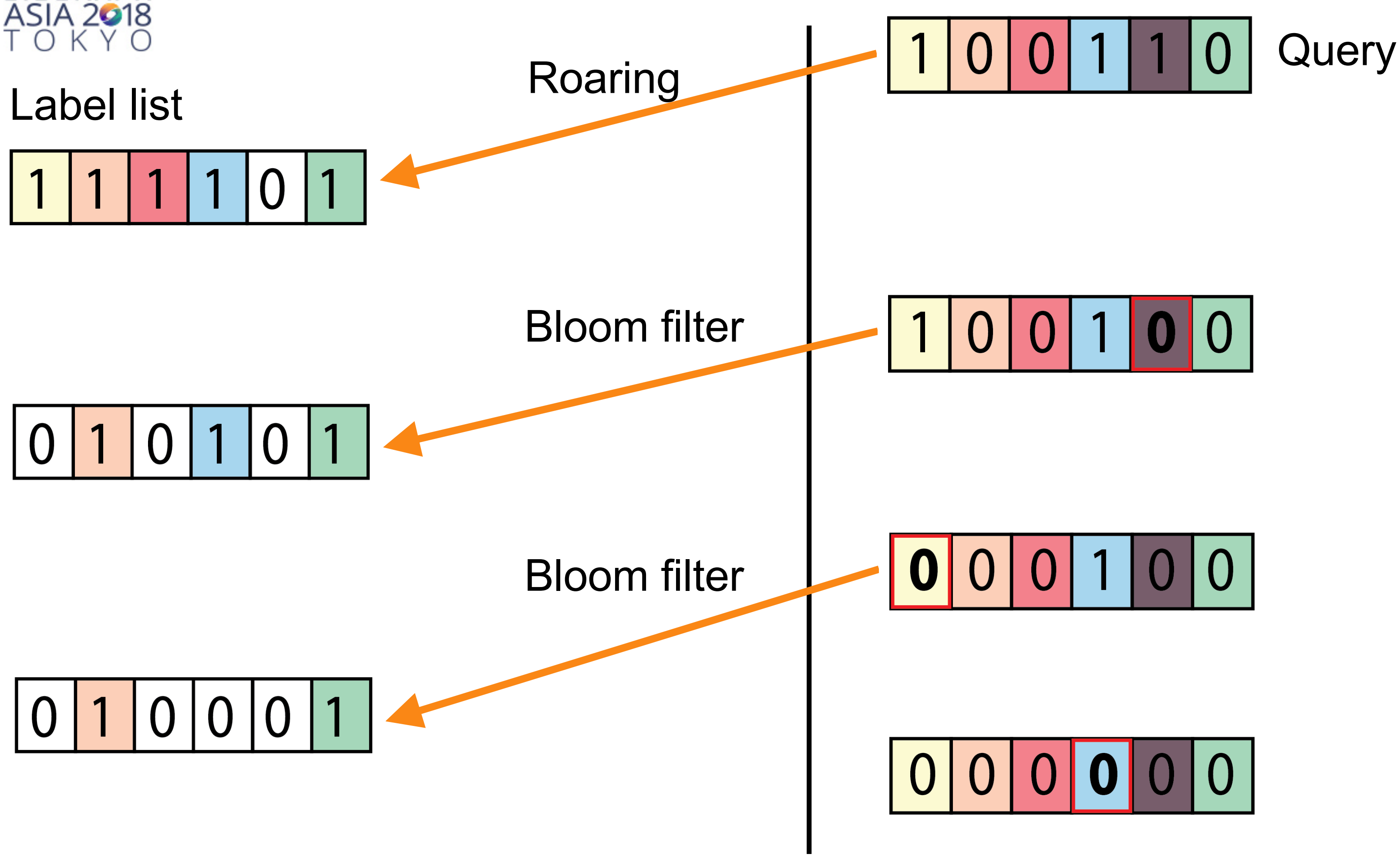
1 0 0 1 **0** 0

0 0 0 1 0 0

0 0 0 **0** 0 0

SIGGRAPH ASIA 2018 TOKYO

QUERY-ADAPTIVE LABEL LIST REQUESTS



Label list

1 1 1 1 0 1

0 1 0 1 0 1

0 1 0 0 0 1

Roaring

Bloom filter

Bloom filter

Query

1 0 0 1 1 0

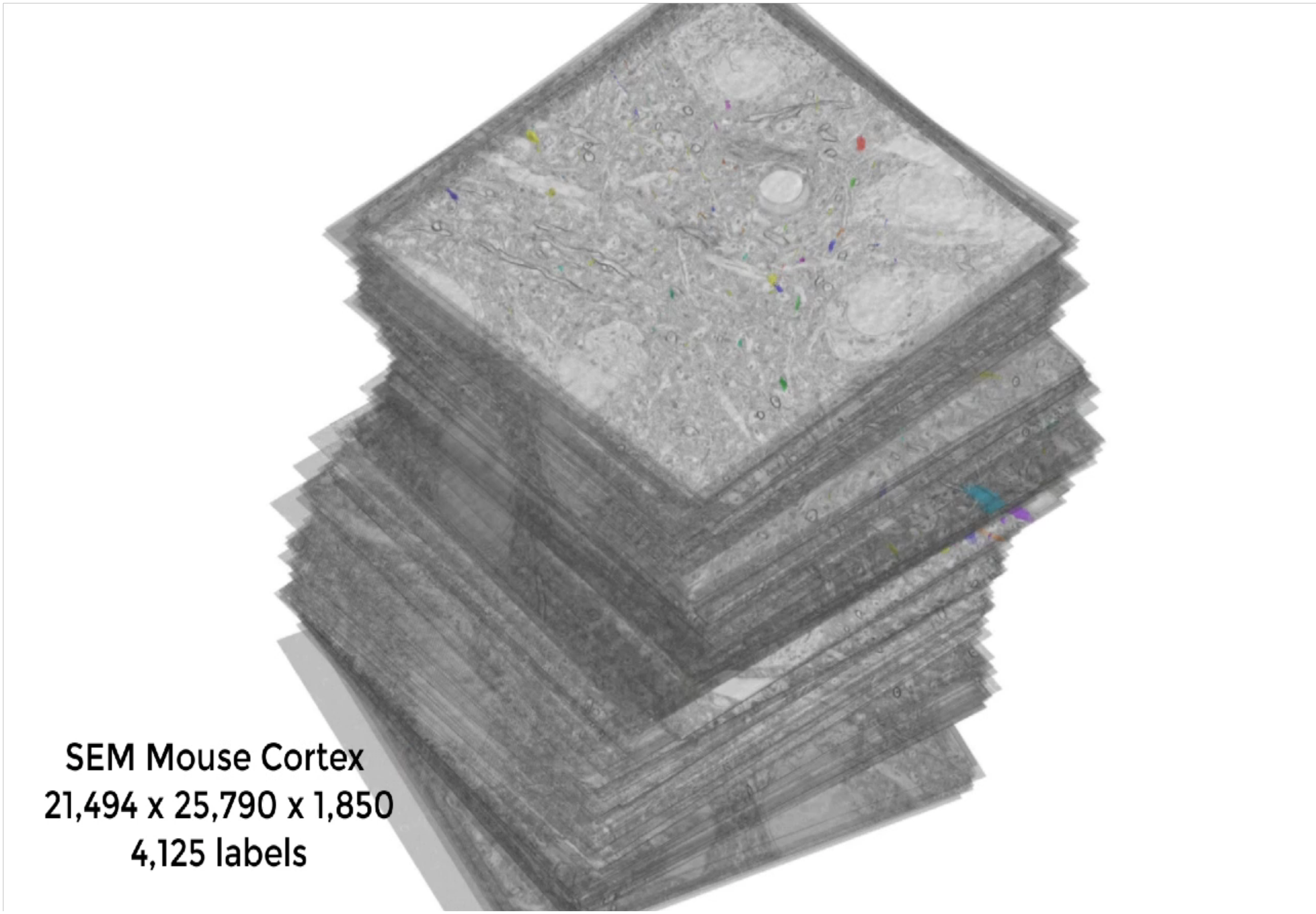
1 0 0 1 **0** 0

0 0 0 1 0 0

0 0 0 **0** 0 0

SIGGRAPH ASIA 2018 TOKYO

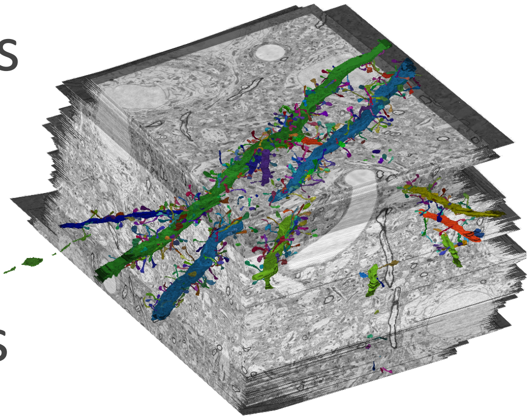
Results

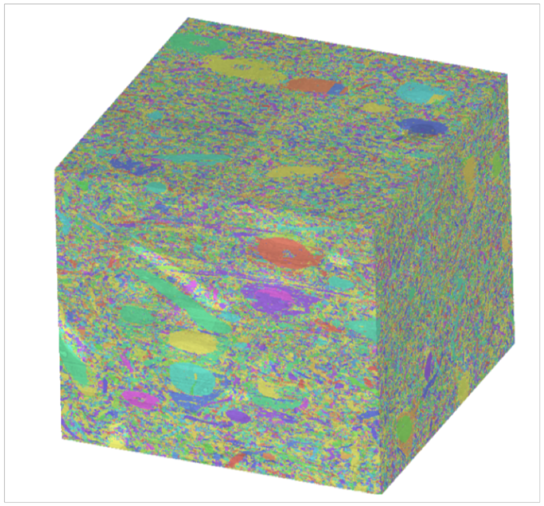


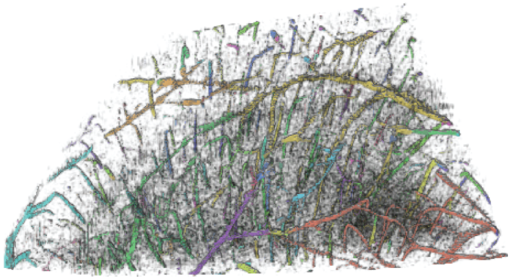
SEM Mouse Cortex
21,494 x 25,790 x 1,850
4,125 labels

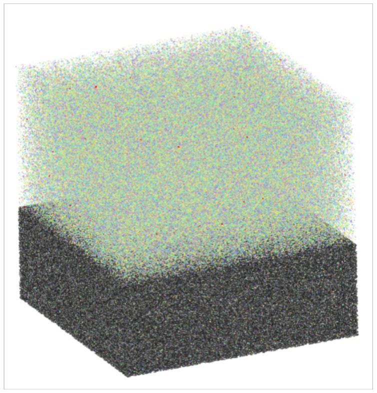
RESULTS - DATASETS

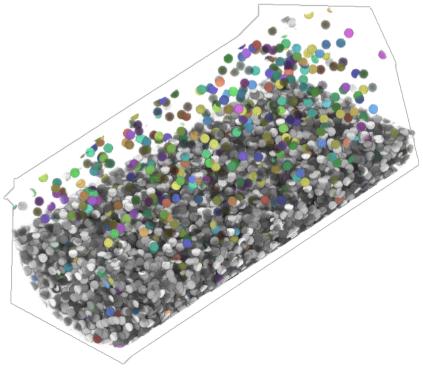
- 3 neuroscience volumes
- 2 phantom datasets
- 16 - 24 bit label data
- 4,000 - 13 million labels
- 4 GB - 1.5 TB data size



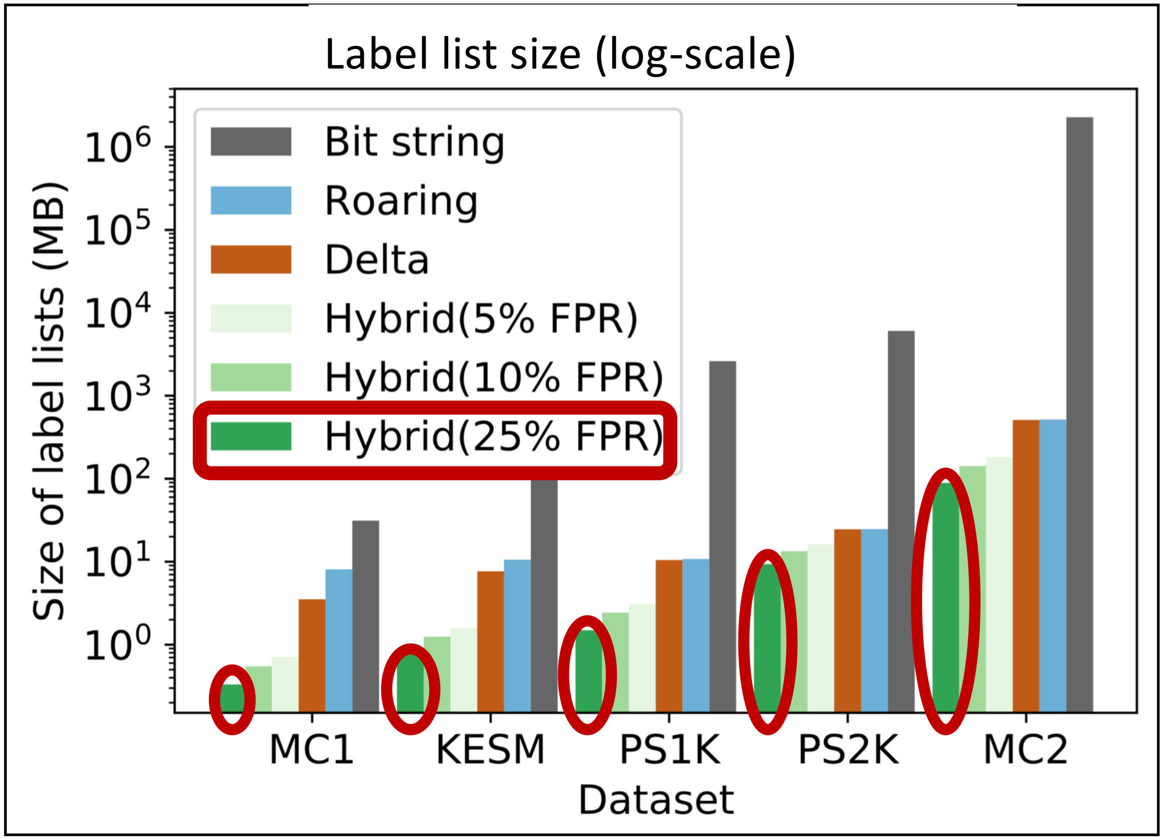




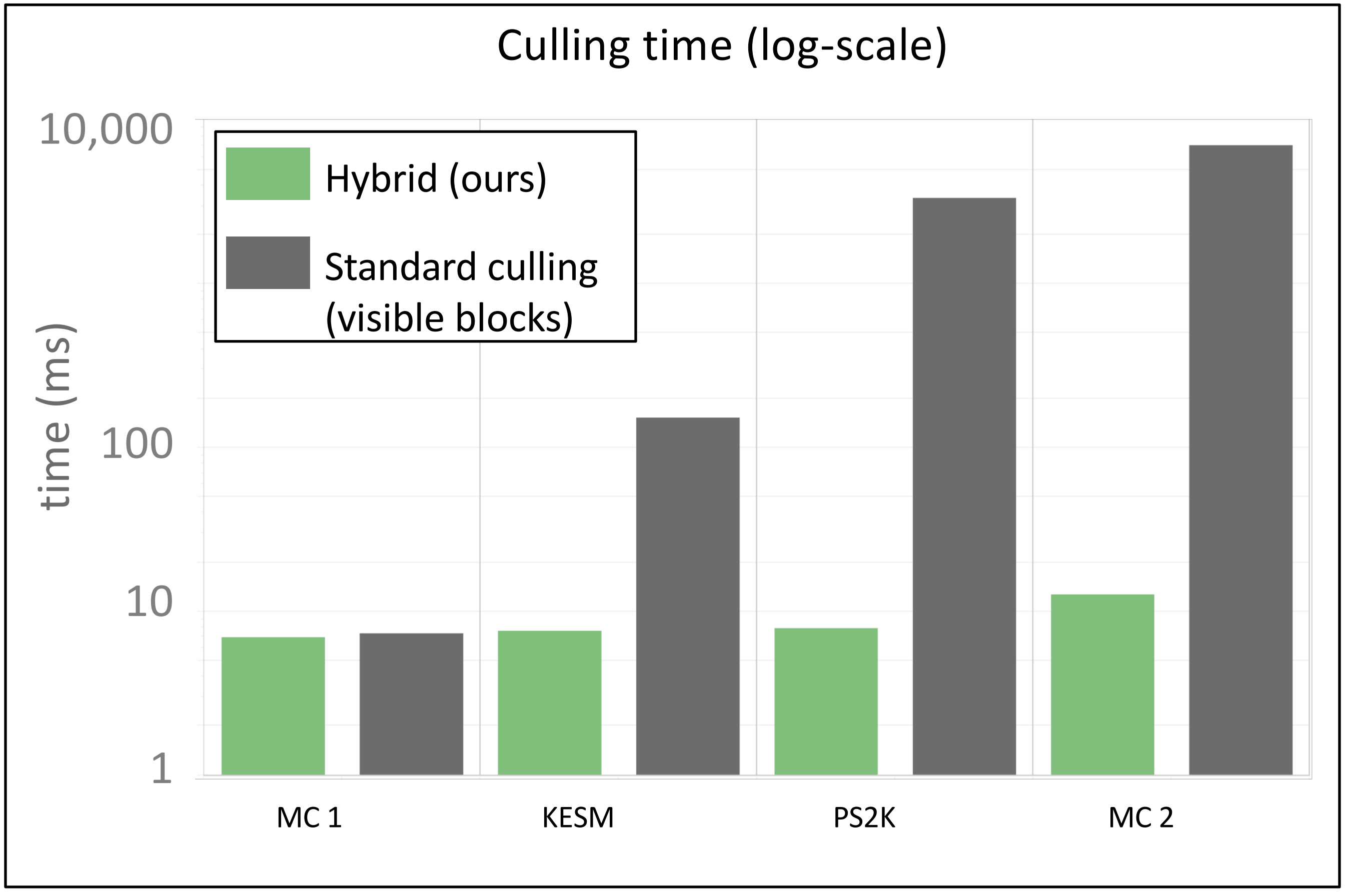




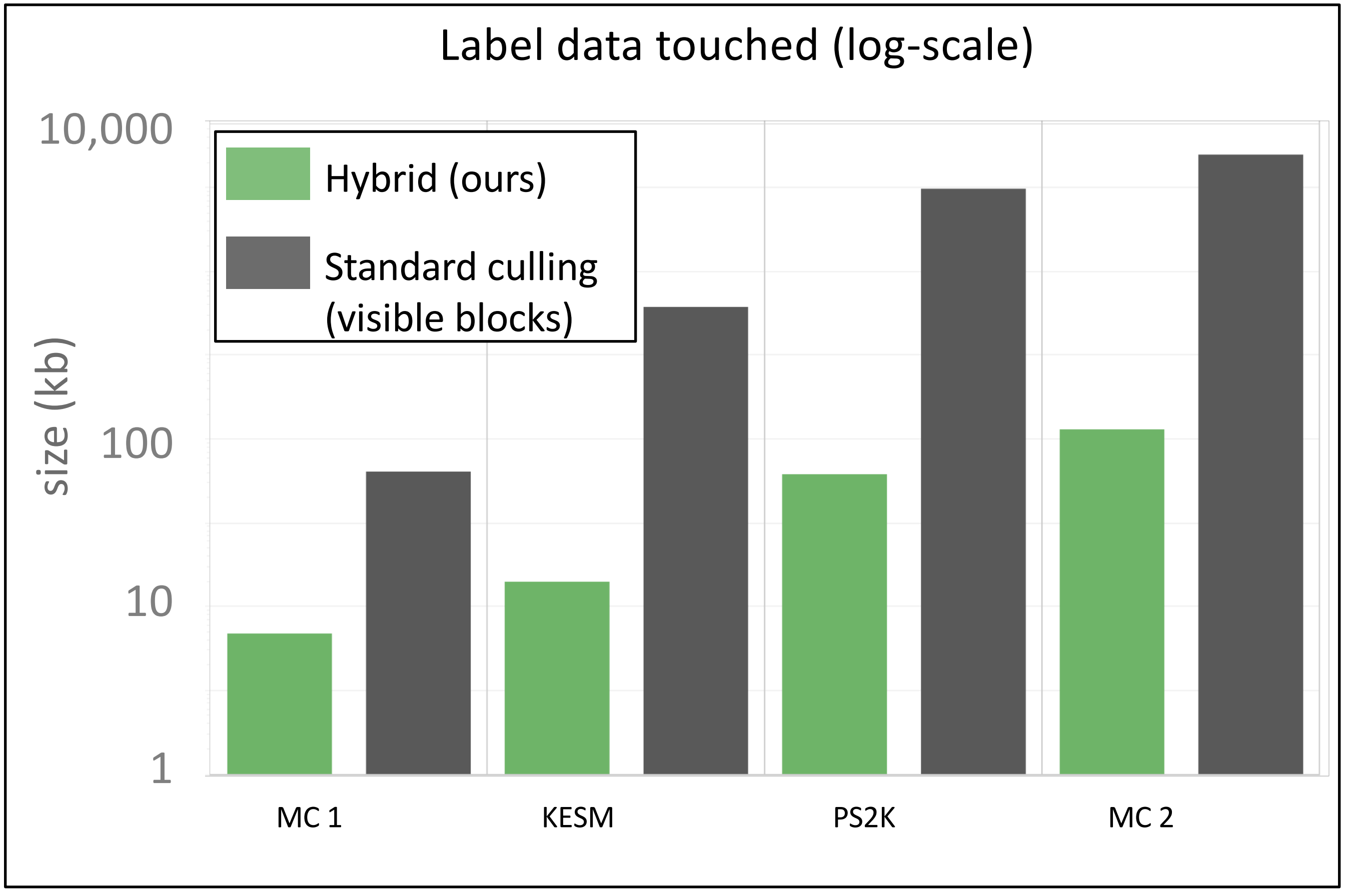
RESULTS – MEMORY CONSUMPTION OF LABEL LISTS



RESULTS – CULLING PERFORMANCE



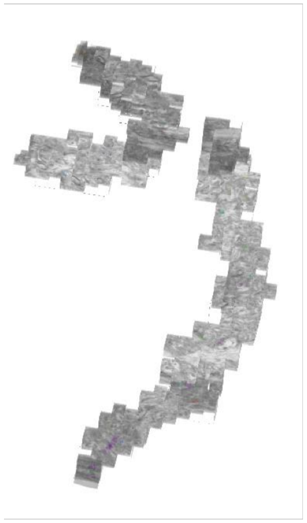
RESULTS – CULLING PERFORMANCE

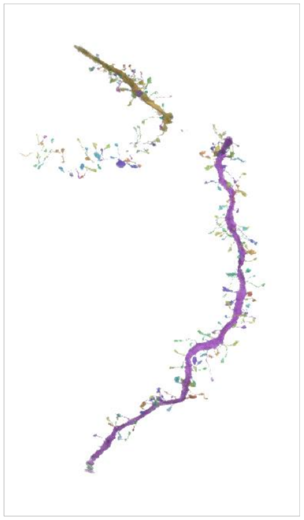


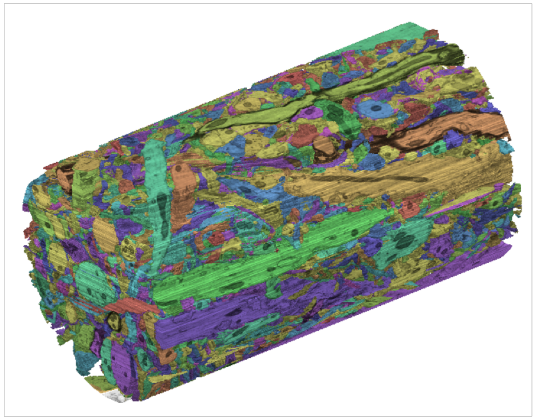
SUMMARY

Our method

1. Novel hybrid data structure → compact storage of integer label lists
2. Hierarchical culling algorithm → fast, memory efficient culling







SIGGRAPH ASIA 2018 TOKYO

Questions?

SIGGRAPH ASIA 2018 TOKYO

CONFERENCE 4 – 7 December 2018
EXHIBITION 5 – 7 December 2018
Tokyo International Forum, Japan
SA2018.SIGGRAPH.ORG

Sponsored by

GPU-Based Large-Scale Scientific Visualization

Johanna Beyer, Harvard University

Markus Hadwiger, KAUST

Course Website:
http://johanna-b.github.io/LargeSciVis2018/index.html