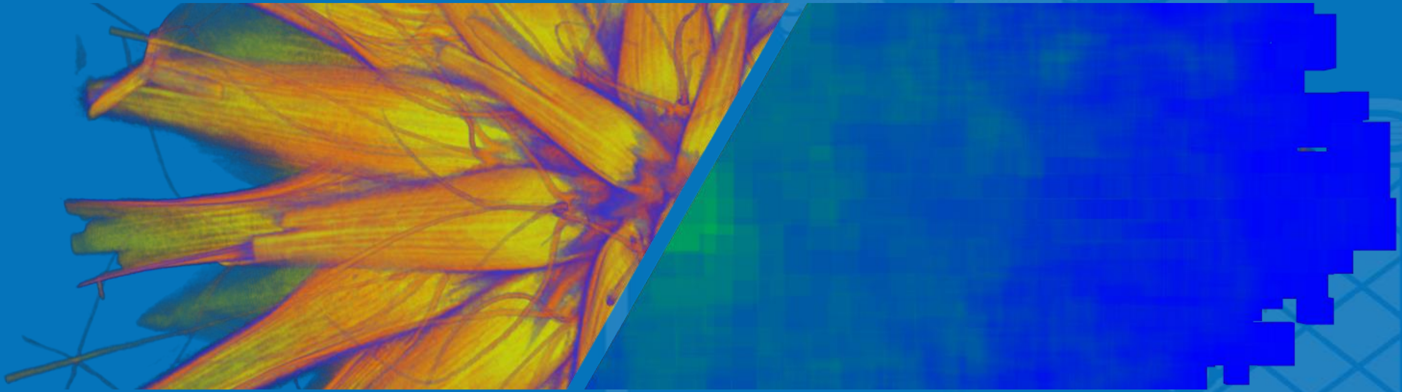
Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin



An Analysis of Region Clustered BVH Volume Rendering on GPU

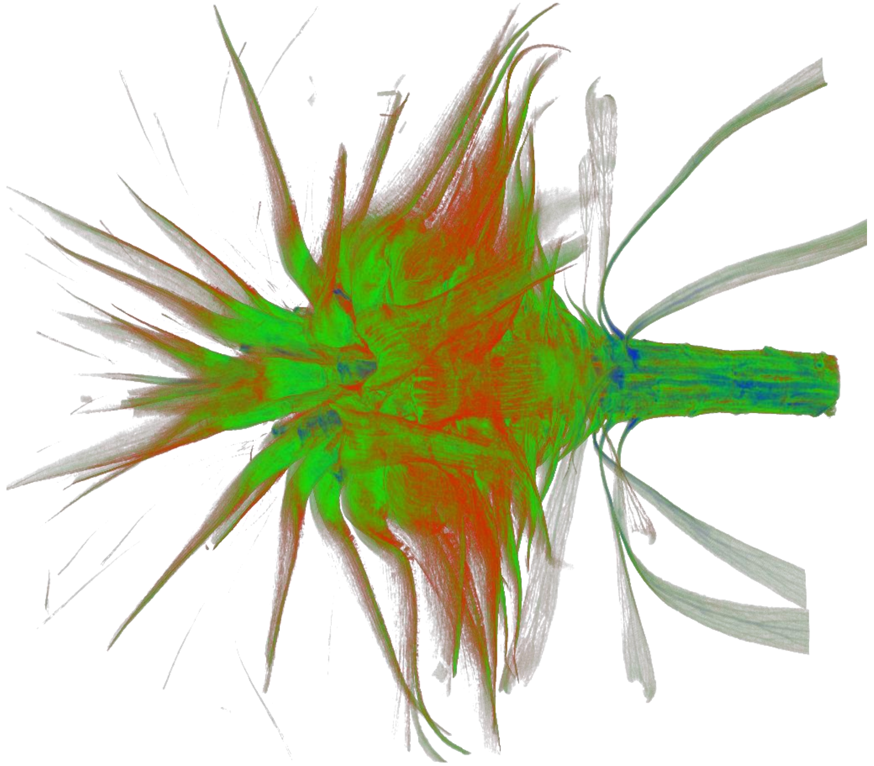
David Ganter[1] and Michael Manzke[2]
School of Computer Science and Statistics
Trinity College Dublin
Ireland

[1]ganterd@scss.tcd.ie
[2]manzkem@scss.tcd.ie

Direct Volume Rendering

Applications

- **Medical**
  - 3D MRI Scans
- **Scientific**
  - Acquired Data
  - Simulations



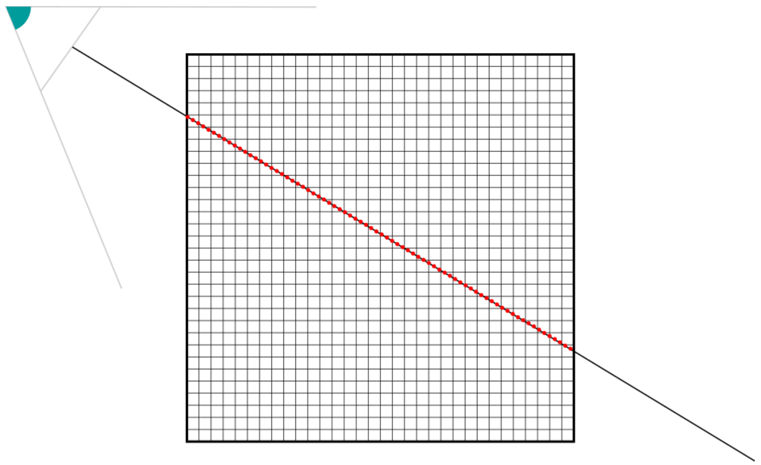
Direct Volume Rendering

Background

- In this work we consider 'volumes' to be regular 3D grids of discrete scalar data volume elements (voxel)
- Volume is resampled by ray at regular intervals
- Scalar value is translated to colour and opacity by a transfer function
    - In our case this is a 1D Look-up-table
- Ray can be traversed front-to-back or back-to-front using either under or over compositing operator

$$I(D_n) = I(D_{n-1})T(D_{n-1}, D_n) + \int_{D_{n-1}}^{D_n} C(s)T(s, D_n)\, ds$$

$$T(p_1, p_2) = e^{-\int_{p1}^{p2} \alpha(p)\, dp}$$



Direct Volume Rendering

Optimisations

- **Early Ray Termination (ERT)**
  - Once an opacity threshold is reached, stop sampling ray
- **Empty Space Skipping (ESS)**
  - Regions of the volume that don’t translate to any opacity don’t need to be sampled

Direct Volume Rendering
Empty Space Skipping (ESS)

- **In Essence**
  - Divide volume into regions of the same size
  - If any voxels in region have opacity greater than zero, region is considered **active**
  - If region is **inactive**, the ray can skip over the empty space to avoid sampling non-contributing data

Background
GPU Based ESS

- **Octrees are popular**
  - Regions – or bricks – make up octree leaves
  - Inner nodes marked as active/inactive based on leaves
  - Ray traverses from top down
  - Inner nodes can be skipped
  - *But:*
    - Fine-grained regions = deeper octrees = potentially more expensive traversal
    - Especially with sparse or thin strands of voxels

Background
Sparseleap (Hadwiger et al 2018)

- Uses octree to generate “occupancy geometry” on CPU when TF updates
- Geometry is rasterized on GPU in front-to-back order
- Gives a list of per-ray entry-exit events
- Events can be merged on the fly if criteria are met
- Ray traversal now just uses entry-exit event list
- ***But:***
  - Occupancy geometry is still tied to octree subdivision bounds
  - Occupancy geometry needs to be rasterised when camera moves

Background

Bounding Volume Hierarchies

- Can represent sparse or thin strands of data with less nodes
- Less nodes can equate to less traversal for ESS
- Traditionally suited to continuous space data like polys (i.e. not on a regular grid like voxels)
- Have not been traditionally used in GPU DVR, partly due to build times, partly to traversal logic

Background
BVH in Direct Volume Rendering

CPU

Knoll et al. 2011 (and subsequent work)

- Used BVH on CPU for full-resolution direct volume rendering

GPU

- ?

Background
NVidia OptiX & RTX

OptiX

- Ray-tracing API

RTX RTCore

- New hardware for Ray-BVH logic

Why not re-evaluate BVHs as a standard for GPU-based ESS for Direct Volume Rendering?

Approach
Assumptions

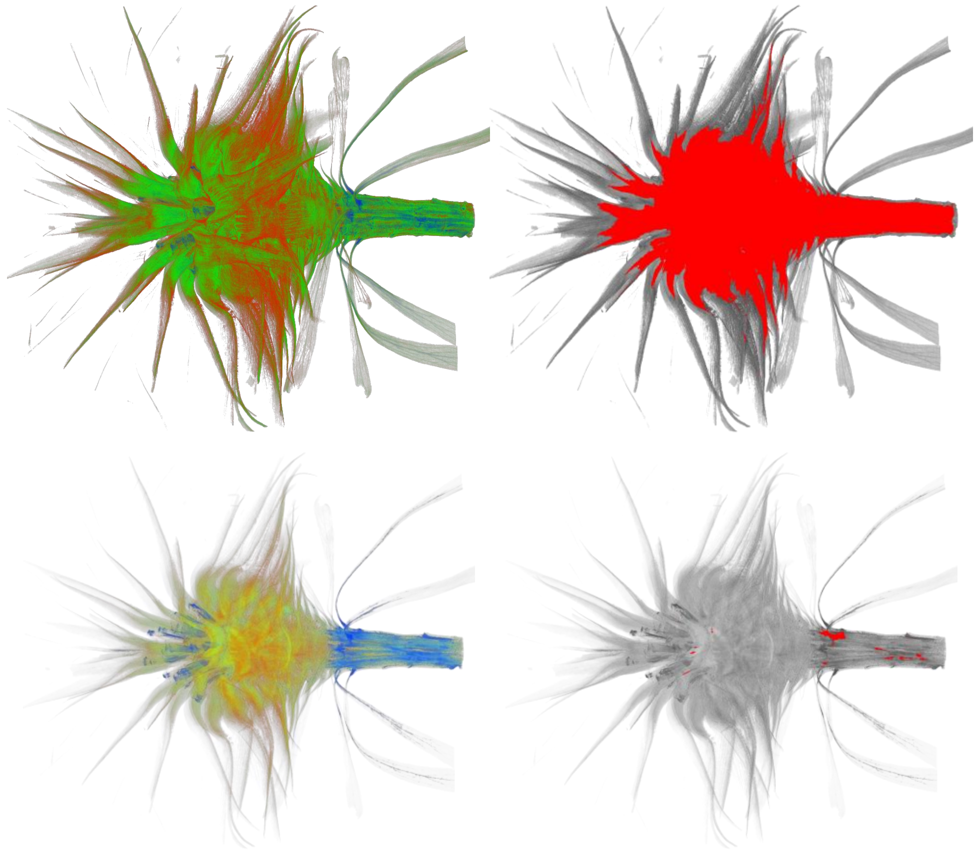
Like Sparseleap:

- Just focussing on ESS portion of DVR
- Underlying sampling is abstracted
  - Uses paged region/brick pool
  - Sampling brick size is not necessarily same as ESS region size
  - Might be optimised for disk IO or cache

Approach

1. Divide volume into regions, storing min/max voxel values
2. When TF updates, regions are quickly tested in parallel.
3. Now we have an array of active/inactive regions (can be stored as bit-string)
4. Spatial bounds of active regions are given to OptiX as AABBs
5. Tell OptiX to ray-trace

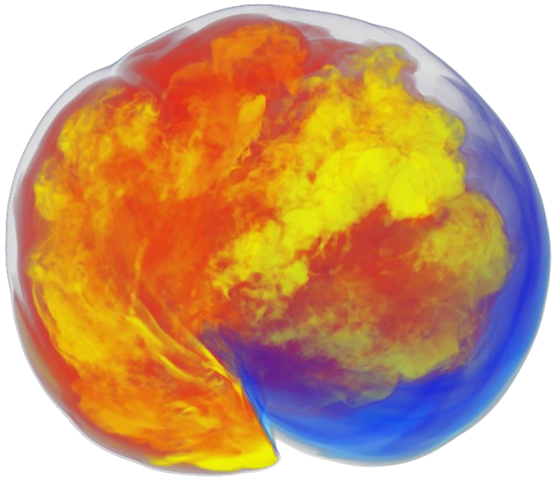
Experiment Data

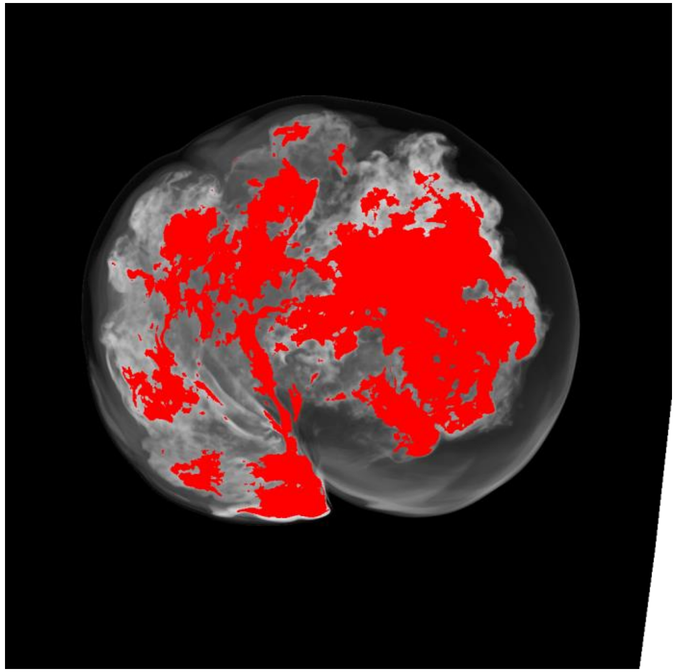


ARTIVVIS

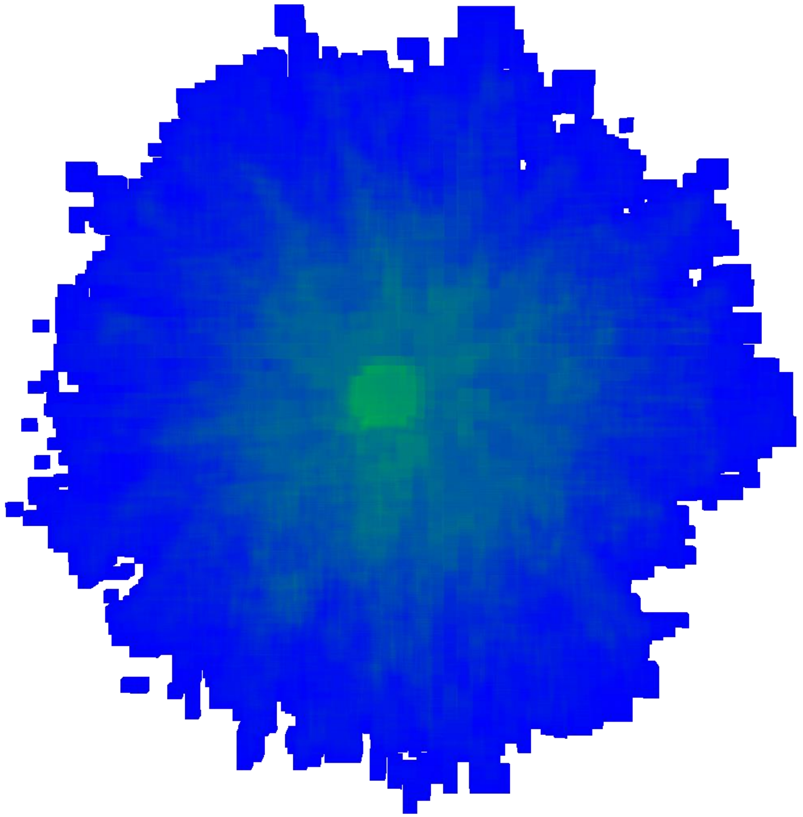
Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Experiment Data



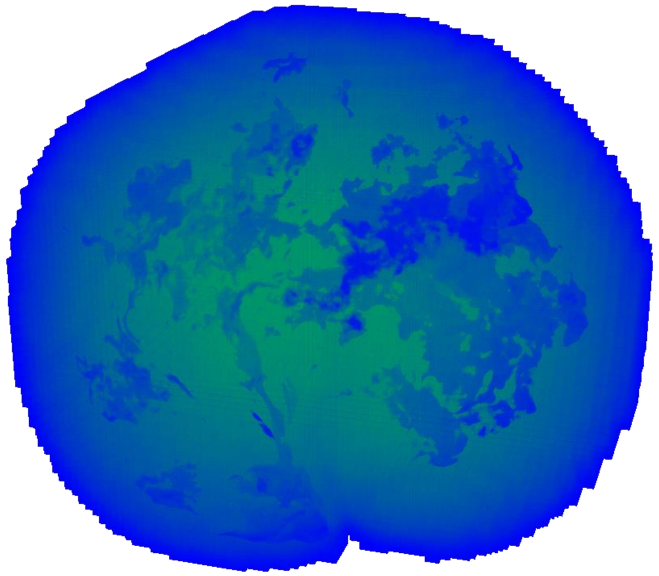


Observation
Depth Complexity



Observation

Depth Complexity



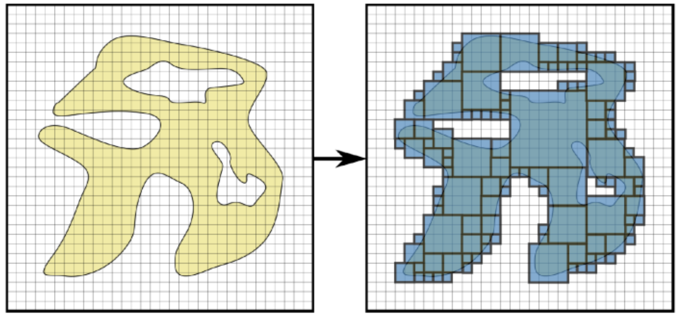
Clustering Approach

Many contiguous groups of active regions

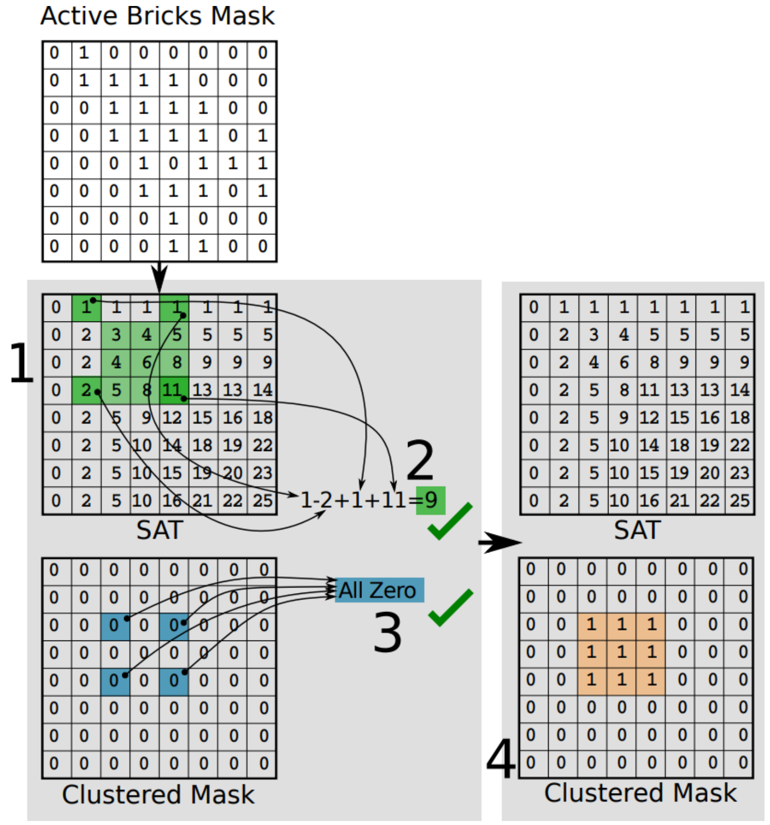
- Many regions share borders, needlessly subdividing the space in the BVH
- We cluster cube-shabed groups of active regions (2x2x2, 3x3x3, 4x4x4 regions, etc)
- Prefer to cluster the largest regions first
- How can we do this efficiently on CPU before giving AABBs to OptiX?

Clustering Approach
3D Summed Area Table (3DSAT)

- Sweep across the active regions array searching for groups of completely active regions
- If considered as a 3D array of active/inactive flags (0/1) we create a 3D summed area table, adding 1 to the sum for every active region.
    - Example: a cluster of 3x3x3 active regions will have a summed-area of 27
- 3DSAT allows us to query how many active regions in an area with 8 lookups.
- Keep another bit-string that represents currently clustered regions (regions that have already been added to a cluster)
- Sweep in descending order of cluster size (64^3, 63^3, 62^3, etc)



Clustering Approach
3D Summed Area Table (3DSAT)



Active Bricks Mask

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

0	1	1	1	1	1	1	1
0	2	3	4	5	5	5	5
0	2	4	6	8	9	9	9
0	2	5	8	11	13	13	14
0	2	5	9	12	15	16	18
0	2	5	10	14	18	19	22
0	2	5	10	15	19	20	23
0	2	5	10	16	21	22	25

SAT

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Clustered Mask

0	1	1	1	1	1	1	1
0	2	3	4	5	5	5	5
0	2	4	6	8	9	9	9
0	2	5	8	11	13	13	14
0	2	5	9	12	15	16	18
0	2	5	10	14	18	19	22
0	2	5	10	15	19	20	23
0	2	5	10	16	21	22	25

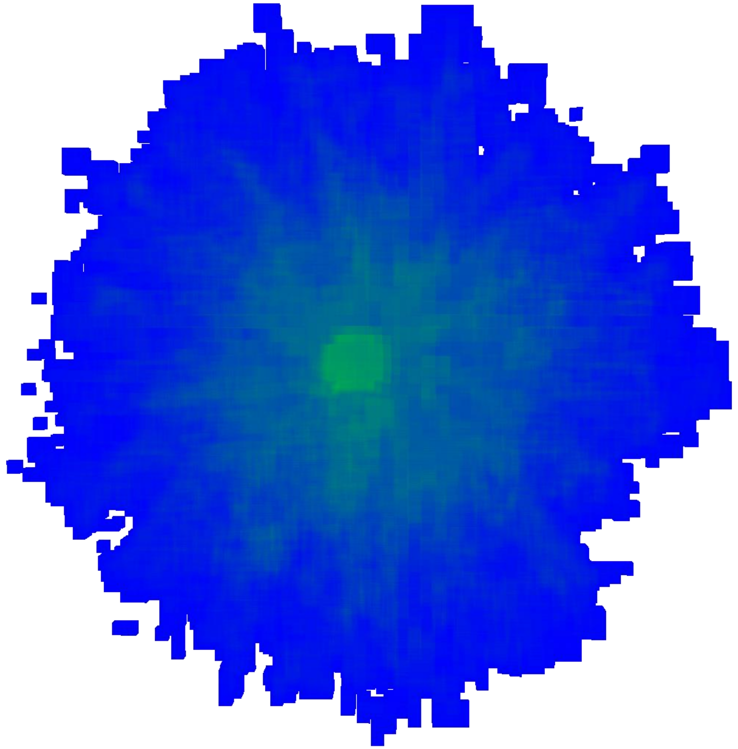
SAT

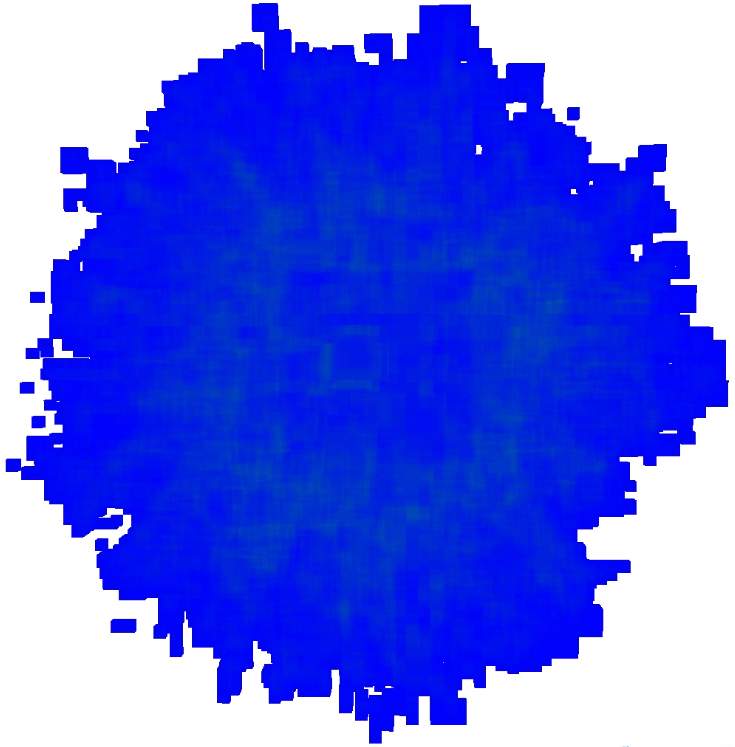
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0
0	0	1	1	1	0	0	0
0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Clustered Mask

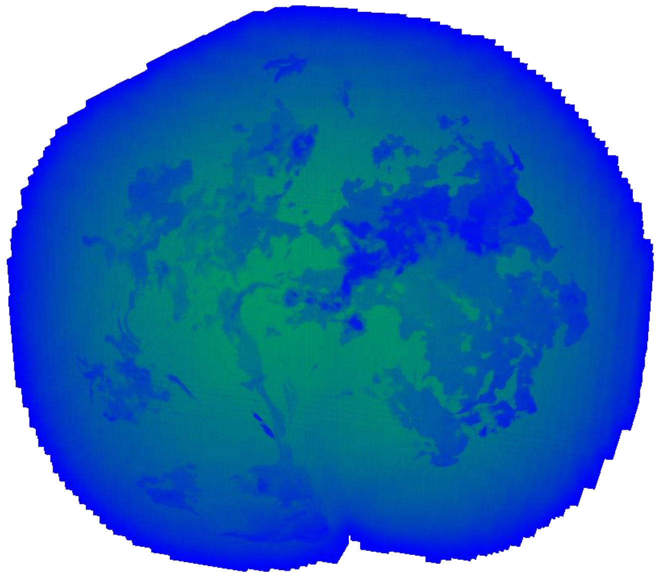
-

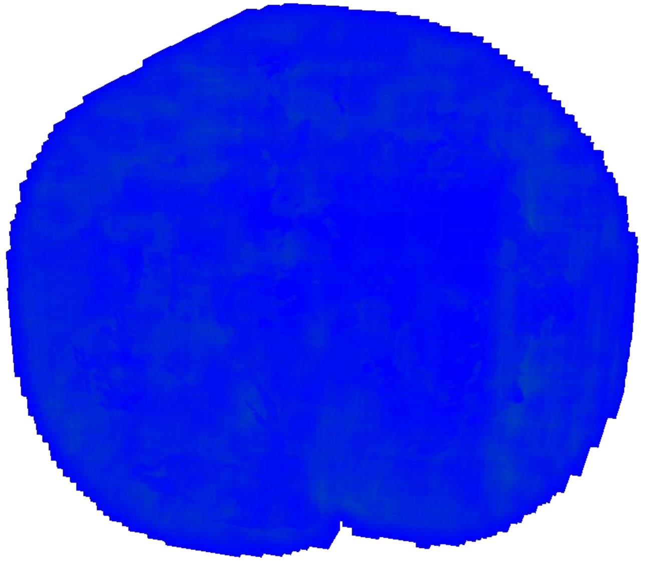
Clustering Approach
Depth Complexity





Clustering Approach
Depth Complexity





Approach

1. Divide volume into regions, storing min/max voxel values
2. When TF updates, regions are quickly tested in parallel.
3. Now we have an array of active/inactive regions (can be stored as bit-string)
4. Clustering
5. Spatial bounds of active regions are given to OptiX as AABBs
6. Tell OptiX to ray-trace

Experiment System

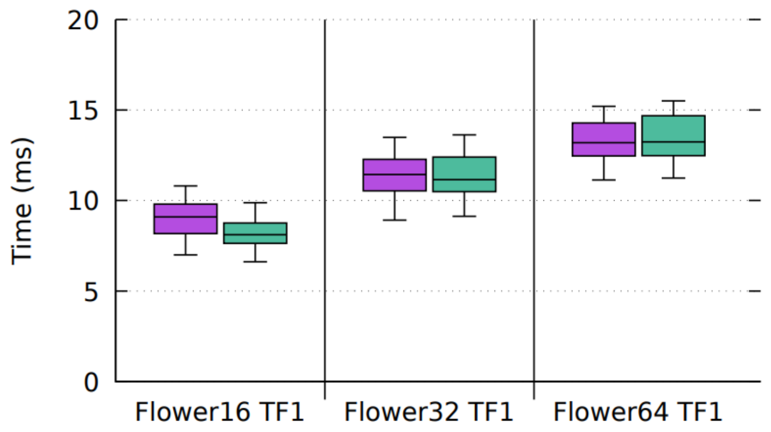
CPU

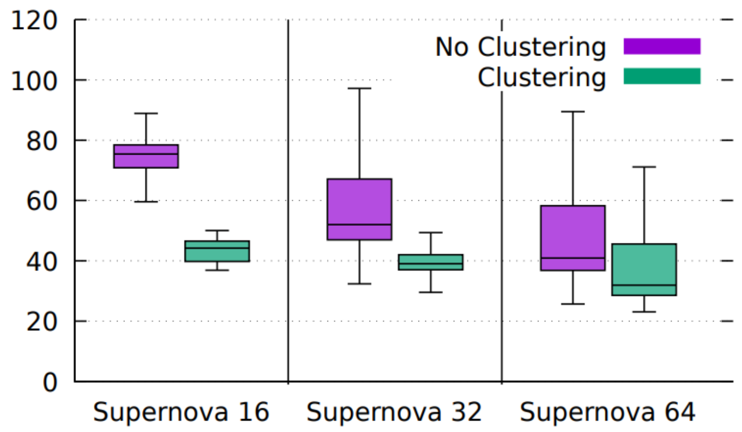
- Intel Xeon E5-1620 v2

GPU

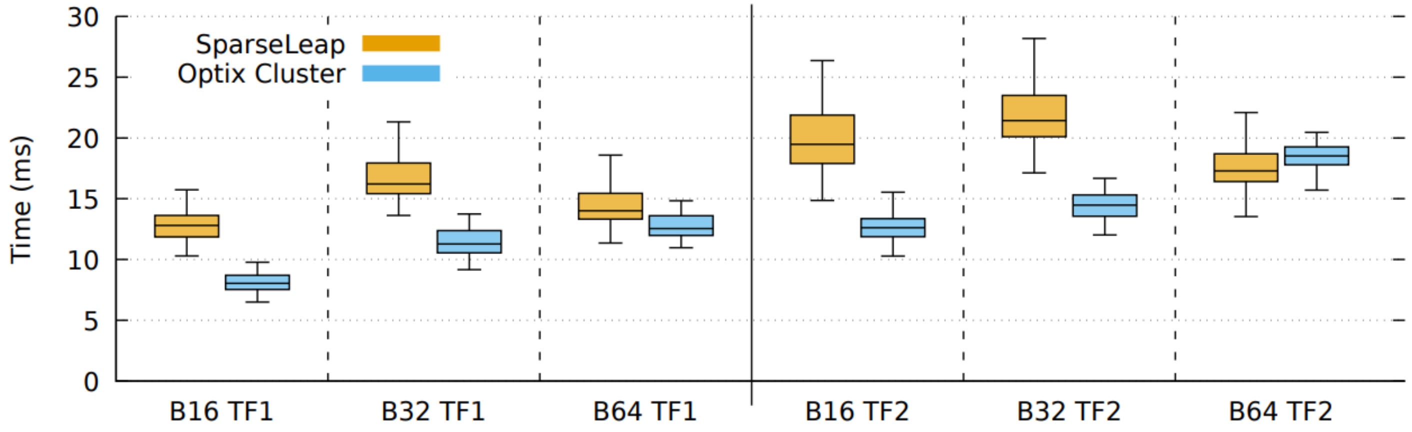
- Nvidia RTX2080

Results
Clustering vs No Clustering



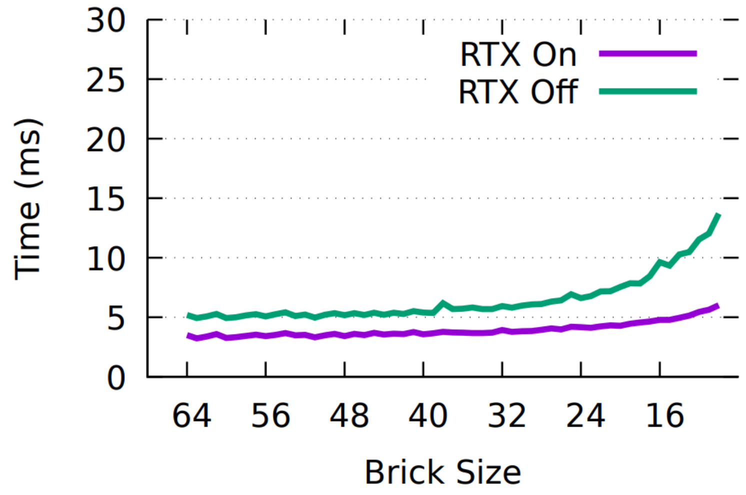


Results
Sparseleap vs OptiX Cluster

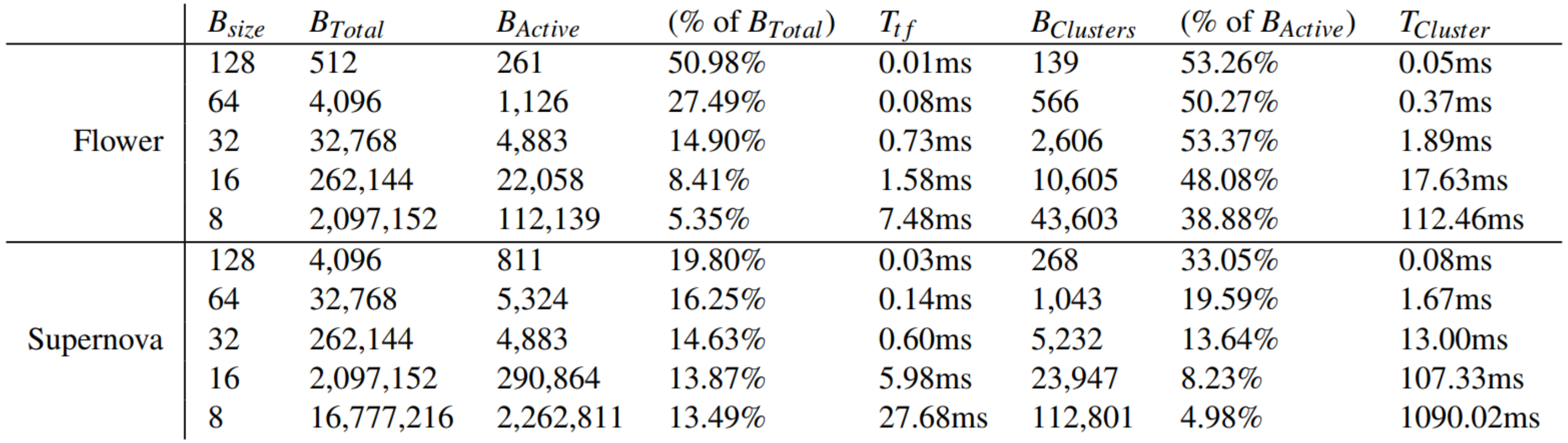


Results

RTX On vs RTX Off (Stubbed Sampling)



Result
Data



	B_{size}	B_{Total}	B_{Active}	(% of B_{Total})	T_{tf}	$B_{Clusters}$	(% of B_{Active})	$T_{Cluster}$
Flower	128	512	261	50.98%	0.01ms	139	53.26%	0.05ms
	64	4,096	1,126	27.49%	0.08ms	566	50.27%	0.37ms
	32	32,768	4,883	14.90%	0.73ms	2,606	53.37%	1.89ms
	16	262,144	22,058	8.41%	1.58ms	10,605	48.08%	17.63ms
	8	2,097,152	112,139	5.35%	7.48ms	43,603	38.88%	112.46ms
Supernova	128	4,096	811	19.80%	0.03ms	268	33.05%	0.08ms
	64	32,768	5,324	16.25%	0.14ms	1,043	19.59%	1.67ms
	32	262,144	4,883	14.63%	0.60ms	5,232	13.64%	13.00ms
	16	2,097,152	290,864	13.87%	5.98ms	23,947	8.23%	107.33ms
	8	16,777,216	2,262,811	13.49%	27.68ms	112,801	4.98%	1090.02ms

Conclusion

- We have shown that BVHs are a viable candidate for GPU direct volume rendering
- We observe that new ray-tracing hardware can benefit GPU direct volume rendering performance
- We show that BVH build times should not be considered a hindering factor
- We give one approach to reduce BVH complexity

Possible Future Work

- Use a clustering heuristic that allows <100% active groups of regions to be clustered
- Cluster non-cubed shapes (2x2x8, 1x4x16, etc)
- Investigate with more volumes
- Translate work to Time-Varying or Steaming Volume Data

Any Questions?

github.com/ganterd/optixdvr