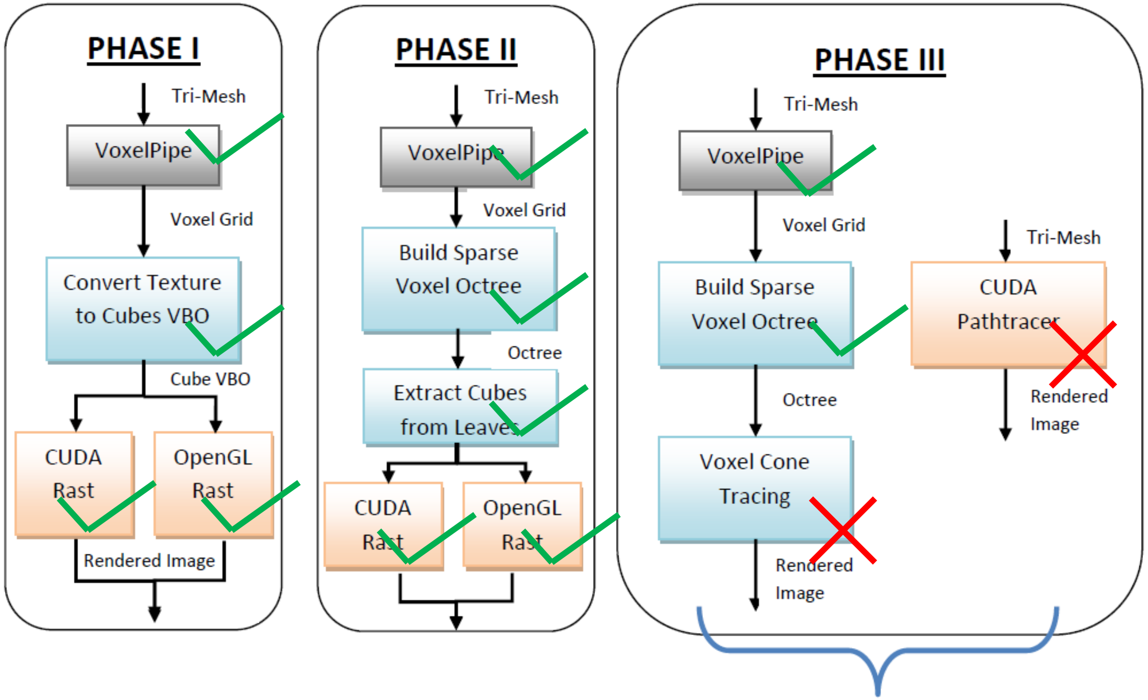
A Voxel Rendering Pipeline in CUDA for Real-time Indirect Illumination

Dave Kotfis

Jiawei Wang

What have we done?

- CUDA/OpenGL equivalent rendering pipelines.
- Voxelization with texture mapping.
- Sparse Voxel Octree Construction (Nodes, but not Bricks)
- Cube extraction and rendering.

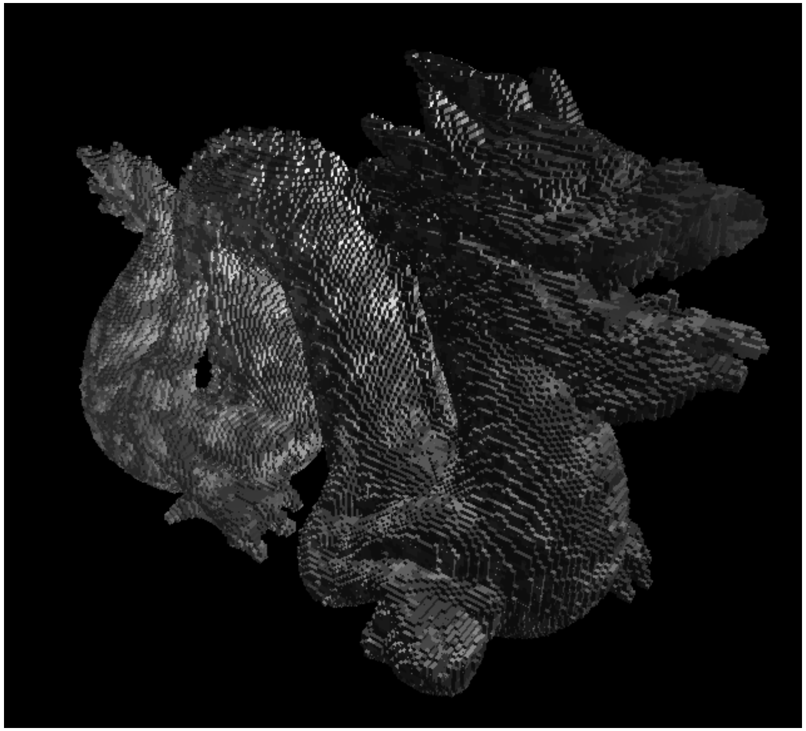


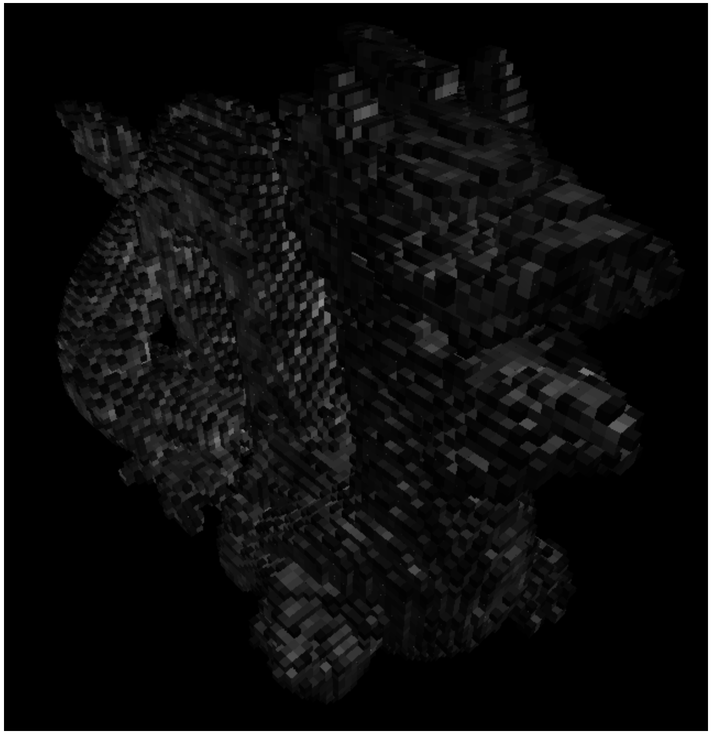
Optional CUDA-Only Pipelines in All Phases

Performance comparison between a voxel-based global illumination scheme, and a path-tracer that runs with meshes without acceleration structures.

Results

- Texture Mapping in VoxelPipe -> Colored Voxels
- Extraction of Cubes from Octree at Arbitrary Level of Detail





Performance Analysis

Data Structure Sizes

Model	Res.	# Voxels	# Octree Nodes
Bunny	128	3,234	9,656
	256	11,603	38,408
	512	43,126	151,552
Dragon	128	6,327	14,696
	256	20,604	58,596
	512	69,865	225,360

Colored voxels are 4 bytes, binary voxels are 1 bit.

Each node is 8 bytes

Performance Analysis (cont)

Voxel Data Structure Timing

Model	Res.	Voxelization	Vox ->Cubes	SVO from Vox	SVO->Cubes
Bunny	128	16.2 ms	8.46 ms	1.85 ms	7.78 ms
	256	54.2 ms	19.7 ms	2.07 ms	22.6 ms
	512	203.9 ms	48.6 ms	2.3 ms	39.9 ms
Dragon	128	26.8 ms	12.1 ms	1.06 ms	12.06 ms
	256	49 ms	30.5 ms	2.79 ms	33.6 ms
	512	221 ms	90.9 ms	2.7 ms	71.8 ms

- The Voxel Grid and SVO structures have comparable runtime to extract cubes for rendering.
- SVO Construction is relatively fast, and scales well at ~ log(Res)
- Dense voxelization cannot be run at real-time rates. Need a static voxelized background.

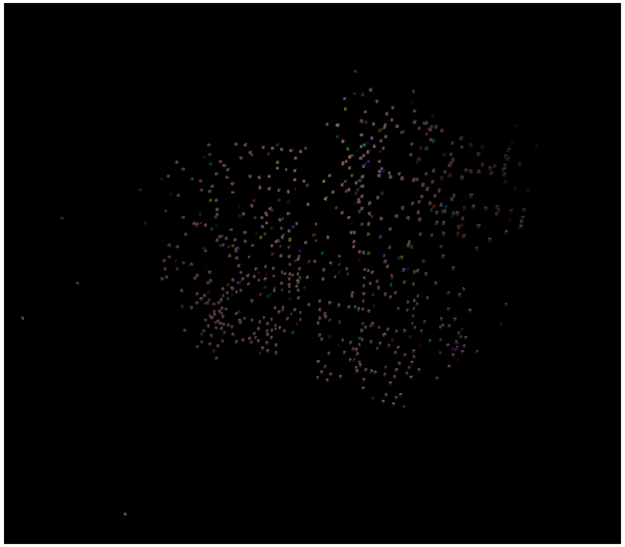
Performance Analysis (cont)

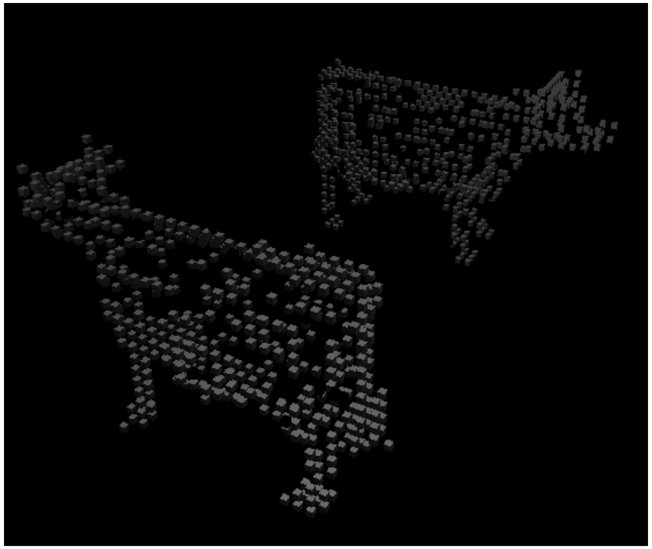
Voxelized?	Model	CUDA Render	CUDA FPS	OpenGL Render	OpenGL FPS
No	Dragon	47 ms	18	15 ms	60
	Two Cows	16 ms	35	0-5 ms	60
	Three*	55 ms	17	15 ms	58
	Bunny	31 ms	29	~0 ms	60
Yes	Dragon	156 ms	5	31 ms	30
	Two Cows	23 ms	27	15 ms	60
	Three*	141 ms	7	16 ms	34
	Bunny	109 ms	10	15 ms	47

*Three is a model scene contains dragon, bunny and buddha objects.

Lessons Learned

- Voxel Data Structures on GPU
  - Memory Hog – Need efficient packing to scale well
  - SVO lacks convenience functions of a conventional CPU octree
- Dynamic Memory Allocation in CUDA Kernels?
  - We pre-allocate memory -> must be conservative!
  - Dynamic allocation would conserve memory.





References

“Interactive Indirect Illumination Using Voxel Cone Tracing” – Cyril Crassin

“Octree-Based Sparse Voxelization Using the GPU Hardware Rasterizer.” – Cyril Crassin. OpenGL Insights, Chapter 22.

“GigaVoxels: A Voxel-Based Rendering Pipeline For Efficient Exploration Of Large And Detailed Scenes” – Cyril Crassin

“VoxelPipe: A Programmable Pipeline for 3D Voxelization” - Jacopo Pantaleoni, NVIDIA Research

Thank You!