



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>





COURSE OVERVIEW - TOPICS

1. Introduction to scalable volume visualization
 - Focus on volume data
 - General scalability and out-of-core techniques
2. Scalable GPU volume rendering
 - Virtual texturing
 - GPU virtual memory architectures
3. Ray-guided volume rendering
 - Visibility-driven data processing
 - Empty-space skipping
4. Display-Aware visualization and data processing



COURSE OVERVIEW - MATERIAL

Course webpage (updated material):

<http://johanna-b.github.io/LargeSciVis2018/index.html>

State-of-the-Art in GPU-Based Large-Scale Volume Visualization

[J. Beyer, M. Hadwiger, H. Pfister; Computer Graphics Forum, 2015]

<https://dl.acm.org/citation.cfm?id=3071497>



COURSE OVERVIEW - SCHEDULE

- Part 1 – Introduction & Basics of Scalable Volume Visualization
Markus Hadwiger [60 min]
- Part 2 – Scalable Volume Visualization Architectures
Johanna Beyer [45 min]
- Break



COURSE OVERVIEW - SCHEDULE

- Part 3 – GPU-Based Ray-Guided Volume Rendering & Empty Space Skipping
Johanna Beyer [60 min]
- Part 4 – Display-Aware Visualization and Processing
Markus Hadwiger [45 min]
- Part 5 – Outlook and Summary
Johanna Beyer, Markus Hadwiger [15 min]



Part 1 -

Introduction & Basics of

Scalable Volume Visualization



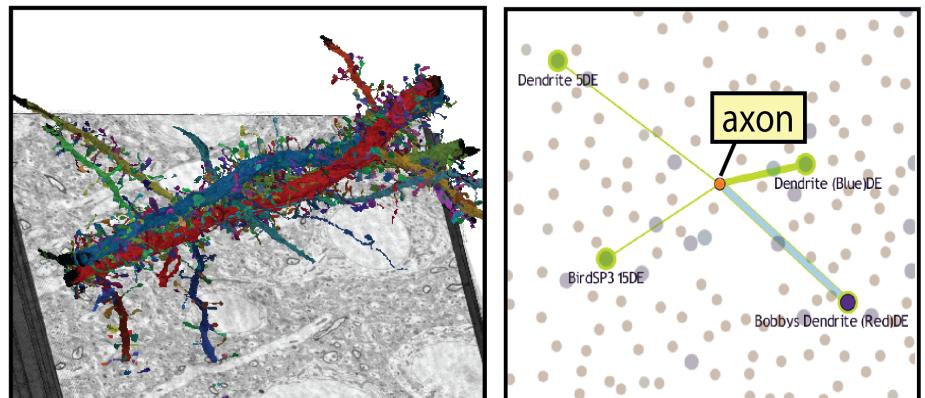
Motivation

BIG DATA

“In information technology, big data is a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications. The challenges include capture, curation, storage, search, sharing, analysis, and visualization.”

‘Big Data’ on wikipedia.org

Our interest:
Very large 3D volume data



Example: Connectomics (neuroscience)

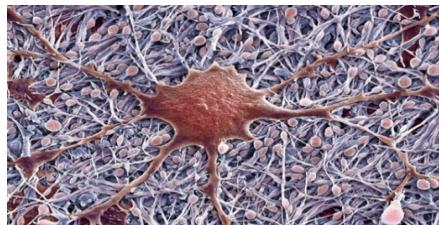


DATA-DRIVEN SCIENCE (E-SCIENCE)



MEDICINE

Digital Health Records



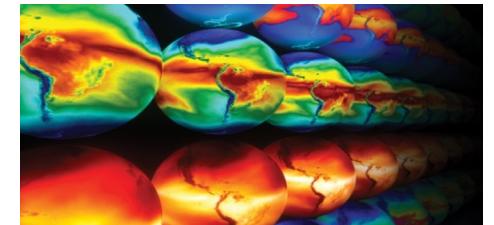
BIOLOGY

Connectomics



ENGINEERING

Large CFD Simulations



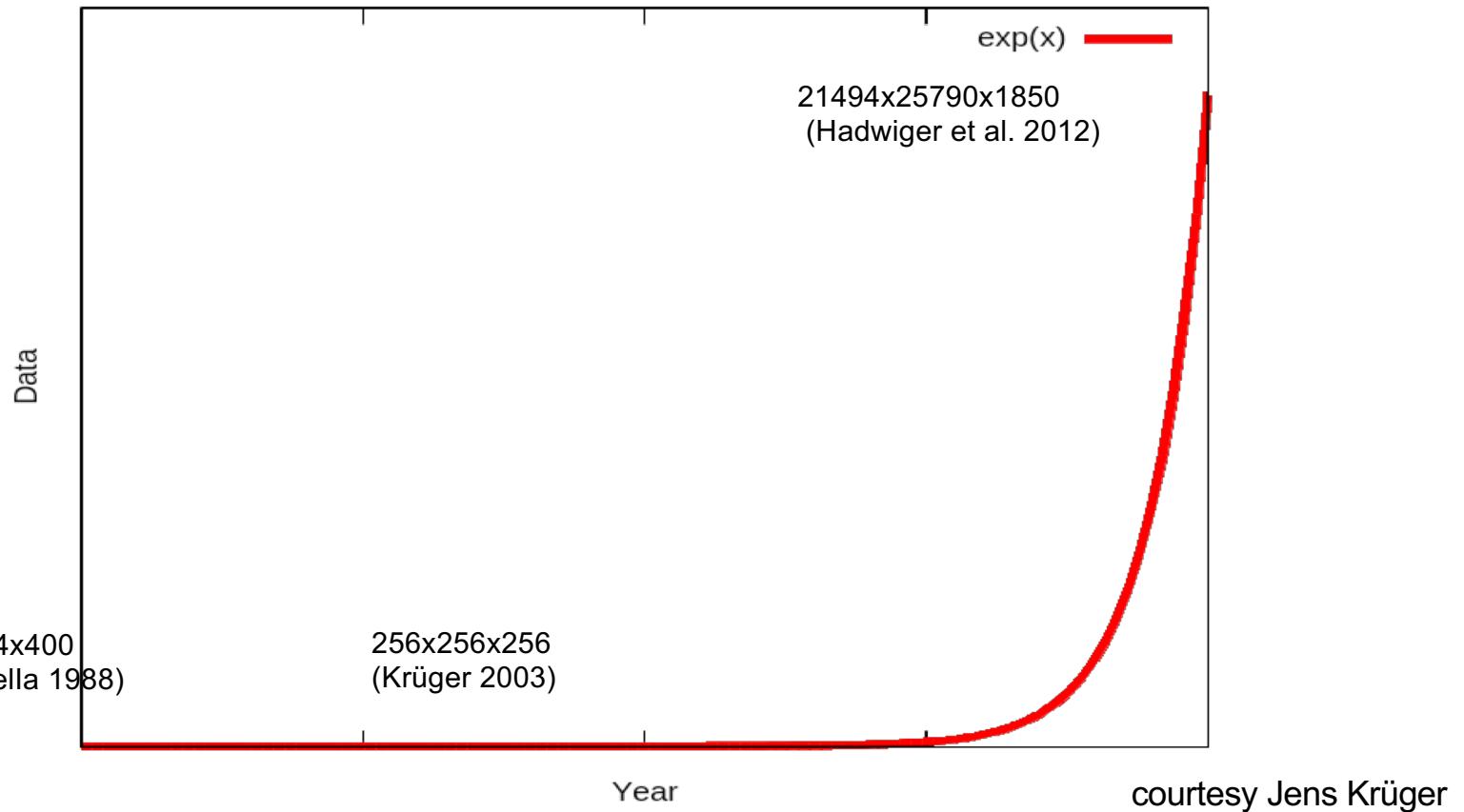
EARTH SCIENCES

Global Climate Models

courtesy Stefan Bruckner



VOLUME DATA GROWTH



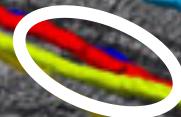


DATA SIZE EXAMPLES

year	paper	data set size	comments
2002	Guthe et al.	512 x 512 x 999 (500 MB) 2,048 x 1,216 x 1,877 (4.4 GB)	multi-pass, wavelet compression, streaming from disk
2003	Krüger & Westermann	256 x 256 x 256 (32 MB)	single-pass ray-casting
2005	Hadwiger et al.	576 x 352 x 1,536 (594 MB)	single-pass ray-casting (bricked)
2006	Ljung	512 x 512 x 628 (314 MB) 512 x 512 x 3396 (1.7 GB)	single-pass ray-casting, multi-resolution
2008	Gobbetti et al.	2,048 x 1,024 x 1,080 (4.2 GB)	'ray-guided' ray-casting with occlusion queries
2009	Crassin et al.	8,192 x 8,192 x 8,192 (512 GB)	ray-guided ray-casting
2011	Engel	8,192 x 8,192 x 16,384 (1 TB)	ray-guided ray-casting
2012	Hadwiger et al.	18,000 x 18,000 x 304 (92 GB) 21,494 x 25,790 x 1,850 (955 GB)	ray-guided ray-casting visualization-driven system
2013	Fogal et al.	1,728 x 1,008 x 1,878 (12.2 GB) 8,192 x 8,192 x 8,192 (512 GB)	ray-guided ray-casting
2018	Beyer et al.	21,494 x 25,790 x 1,850 (955 GB) images + 10,747 x 12,895 x 1,850 (489 GB) segmentation	ray-guided ray-casting, empty space skipping

The Connectome

How is the Mammalian Brain Wired?



Daniel Berger, MIT

The Connectome

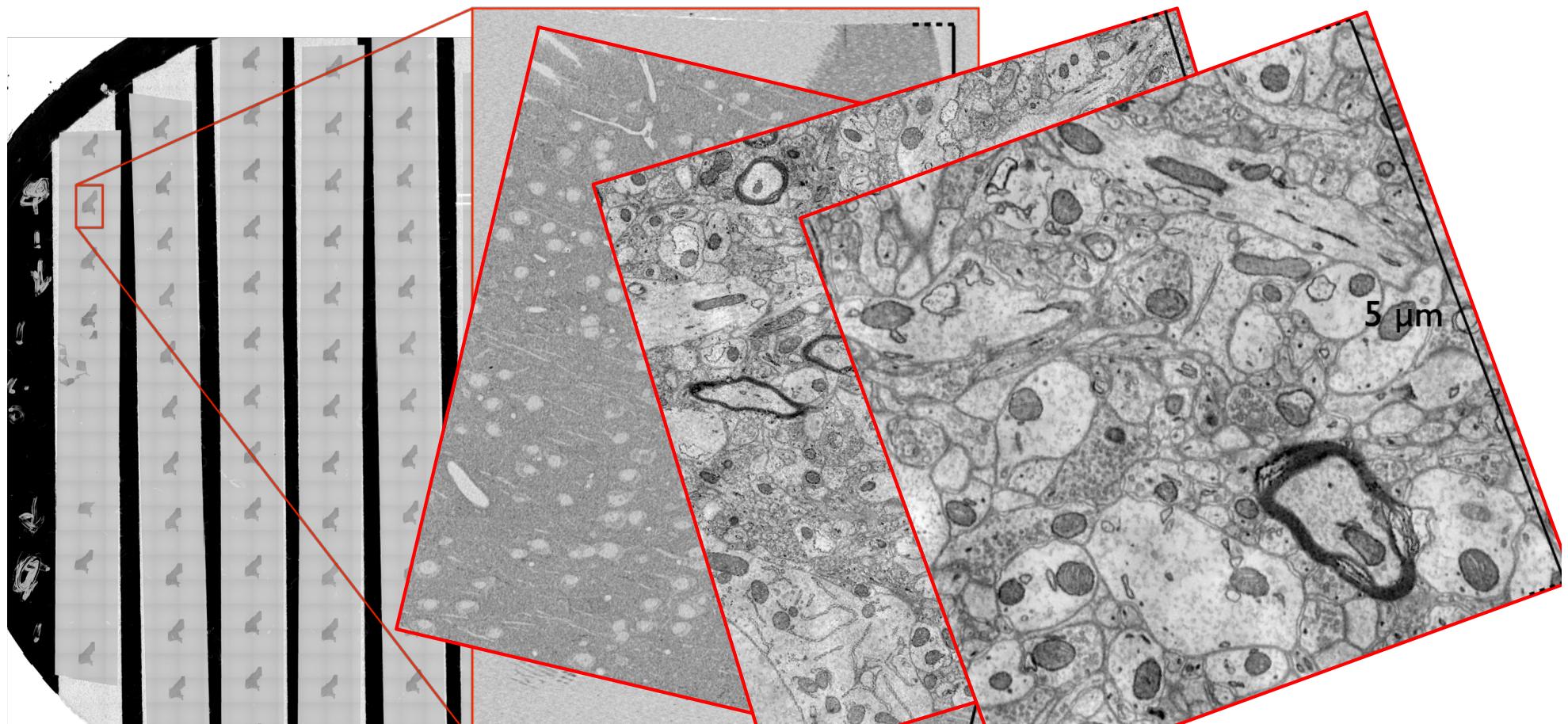
How is the Mammalian Brain Wired?

~60 μm^3
1 Teravoxel
 $21,500 \times 25,800 \times 1,850$

Bobby Kasthuri, Harvard



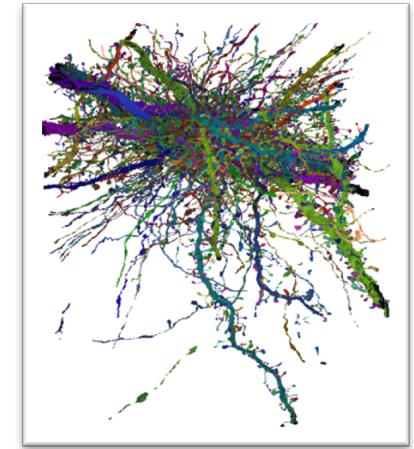
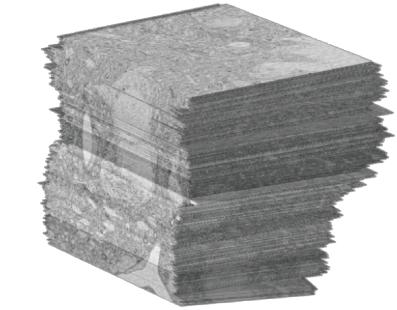
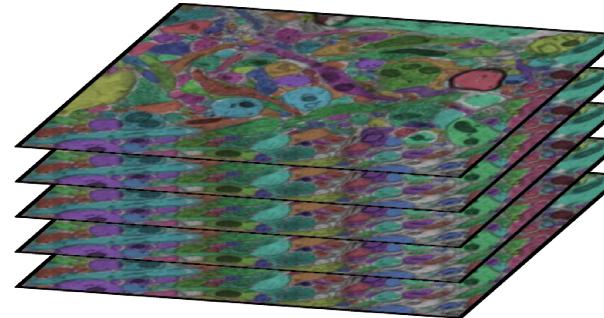
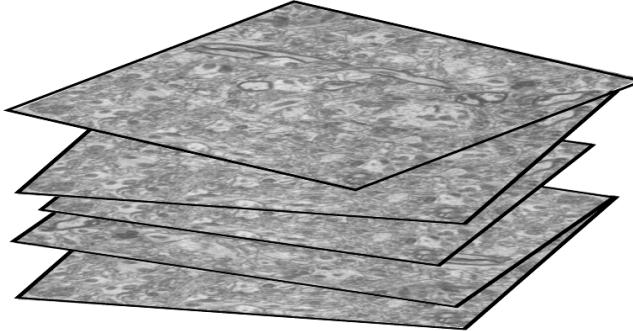
ELECTRON MICROSCOPY (EM) IMAGES





PETAVOXEL MICROSCOPY VOLUMES

- Huge amount of raw data (terabytes to petabytes)
- Takes months to years to scan, align, segment
- How to visualize and analyze this?



Index	Time Step	Cell Type	Location	Properties
1	1	Pyramidal	Layer 1	High density, low variance
2	2	Interneuron	Layer 2	Medium density, moderate variance
3	3	Pyramidal	Layer 3	Low density, high variance
4	4	Interneuron	Layer 4	Very low density, very high variance
5	5	Pyramidal	Layer 5	Medium density, moderate variance
6	6	Interneuron	Layer 6	High density, low variance
7	7	Pyramidal	Layer 7	Low density, high variance
8	8	Interneuron	Layer 8	Very low density, very high variance
9	9	Pyramidal	Layer 9	Medium density, moderate variance
10	10	Interneuron	Layer 10	High density, low variance
11	11	Pyramidal	Layer 11	Low density, high variance
12	12	Interneuron	Layer 12	Very low density, very high variance
13	13	Pyramidal	Layer 13	Medium density, moderate variance
14	14	Interneuron	Layer 14	High density, low variance
15	15	Pyramidal	Layer 15	Low density, high variance
16	16	Interneuron	Layer 16	Very low density, very high variance
17	17	Pyramidal	Layer 17	Medium density, moderate variance
18	18	Interneuron	Layer 18	High density, low variance
19	19	Pyramidal	Layer 19	Low density, high variance
20	20	Interneuron	Layer 20	Very low density, very high variance
21	21	Pyramidal	Layer 21	Medium density, moderate variance
22	22	Interneuron	Layer 22	High density, low variance
23	23	Pyramidal	Layer 23	Low density, high variance
24	24	Interneuron	Layer 24	Very low density, very high variance
25	25	Pyramidal	Layer 25	Medium density, moderate variance
26	26	Interneuron	Layer 26	High density, low variance
27	27	Pyramidal	Layer 27	Low density, high variance
28	28	Interneuron	Layer 28	Very low density, very high variance
29	29	Pyramidal	Layer 29	Medium density, moderate variance
30	30	Interneuron	Layer 30	High density, low variance
31	31	Pyramidal	Layer 31	Low density, high variance
32	32	Interneuron	Layer 32	Very low density, very high variance
33	33	Pyramidal	Layer 33	Medium density, moderate variance
34	34	Interneuron	Layer 34	High density, low variance
35	35	Pyramidal	Layer 35	Low density, high variance
36	36	Interneuron	Layer 36	Very low density, very high variance
37	37	Pyramidal	Layer 37	Medium density, moderate variance
38	38	Interneuron	Layer 38	High density, low variance
39	39	Pyramidal	Layer 39	Low density, high variance
40	40	Interneuron	Layer 40	Very low density, very high variance
41	41	Pyramidal	Layer 41	Medium density, moderate variance
42	42	Interneuron	Layer 42	High density, low variance
43	43	Pyramidal	Layer 43	Low density, high variance
44	44	Interneuron	Layer 44	Very low density, very high variance
45	45	Pyramidal	Layer 45	Medium density, moderate variance
46	46	Interneuron	Layer 46	High density, low variance
47	47	Pyramidal	Layer 47	Low density, high variance
48	48	Interneuron	Layer 48	Very low density, very high variance
49	49	Pyramidal	Layer 49	Medium density, moderate variance
50	50	Interneuron	Layer 50	High density, low variance
51	51	Pyramidal	Layer 51	Low density, high variance
52	52	Interneuron	Layer 52	Very low density, very high variance
53	53	Pyramidal	Layer 53	Medium density, moderate variance
54	54	Interneuron	Layer 54	High density, low variance
55	55	Pyramidal	Layer 55	Low density, high variance
56	56	Interneuron	Layer 56	Very low density, very high variance
57	57	Pyramidal	Layer 57	Medium density, moderate variance
58	58	Interneuron	Layer 58	High density, low variance
59	59	Pyramidal	Layer 59	Low density, high variance
60	60	Interneuron	Layer 60	Very low density, very high variance
61	61	Pyramidal	Layer 61	Medium density, moderate variance
62	62	Interneuron	Layer 62	High density, low variance
63	63	Pyramidal	Layer 63	Low density, high variance
64	64	Interneuron	Layer 64	Very low density, very high variance
65	65	Pyramidal	Layer 65	Medium density, moderate variance
66	66	Interneuron	Layer 66	High density, low variance
67	67	Pyramidal	Layer 67	Low density, high variance
68	68	Interneuron	Layer 68	Very low density, very high variance
69	69	Pyramidal	Layer 69	Medium density, moderate variance
70	70	Interneuron	Layer 70	High density, low variance
71	71	Pyramidal	Layer 71	Low density, high variance
72	72	Interneuron	Layer 72	Very low density, very high variance
73	73	Pyramidal	Layer 73	Medium density, moderate variance
74	74	Interneuron	Layer 74	High density, low variance
75	75	Pyramidal	Layer 75	Low density, high variance
76	76	Interneuron	Layer 76	Very low density, very high variance
77	77	Pyramidal	Layer 77	Medium density, moderate variance
78	78	Interneuron	Layer 78	High density, low variance
79	79	Pyramidal	Layer 79	Low density, high variance
80	80	Interneuron	Layer 80	Very low density, very high variance
81	81	Pyramidal	Layer 81	Medium density, moderate variance
82	82	Interneuron	Layer 82	High density, low variance
83	83	Pyramidal	Layer 83	Low density, high variance
84	84	Interneuron	Layer 84	Very low density, very high variance
85	85	Pyramidal	Layer 85	Medium density, moderate variance
86	86	Interneuron	Layer 86	High density, low variance
87	87	Pyramidal	Layer 87	Low density, high variance
88	88	Interneuron	Layer 88	Very low density, very high variance
89	89	Pyramidal	Layer 89	Medium density, moderate variance
90	90	Interneuron	Layer 90	High density, low variance
91	91	Pyramidal	Layer 91	Low density, high variance
92	92	Interneuron	Layer 92	Very low density, very high variance
93	93	Pyramidal	Layer 93	Medium density, moderate variance
94	94	Interneuron	Layer 94	High density, low variance
95	95	Pyramidal	Layer 95	Low density, high variance
96	96	Interneuron	Layer 96	Very low density, very high variance
97	97	Pyramidal	Layer 97	Medium density, moderate variance
98	98	Interneuron	Layer 98	High density, low variance
99	99	Pyramidal	Layer 99	Low density, high variance
100	100	Interneuron	Layer 100	Very low density, very high variance



COURSE SCOPE

Course focus

- (Single) GPUs in standard workstations
- Scalar volume data; single time step
- But a lot applies to more general settings...

Orthogonal techniques (will not cover details)

- Parallel and distributed rendering, clusters, supercomputers, ...
- Compression



RELATED BOOKS AND SURVEYS

Books

- Real-Time Volume Graphics, Engel et al., 2006
- High-Performance Visualization, Bethel et al., 2012

Surveys

- GPU-Based Large-Scale Volume Visualization: Beyer et al. '15
- Parallel Visualization: Wittenbrink '98, Bartz et al. '00, Zhang et al. '05
- Real Time Interactive Massive Model Visualization: Kasik et al. '06
- Vis and Visual Analysis of Multifaceted Scientific Data: Kehrer and Hauser '13
- Compressed GPU-Based Volume Rendering: Rodriguez et al. '14
- Web-based Visualization: Mwalongo et al. '16
- In-Situ Methods, Infrastructures, and Applications in High Performance Computing: Bauer et al. '16

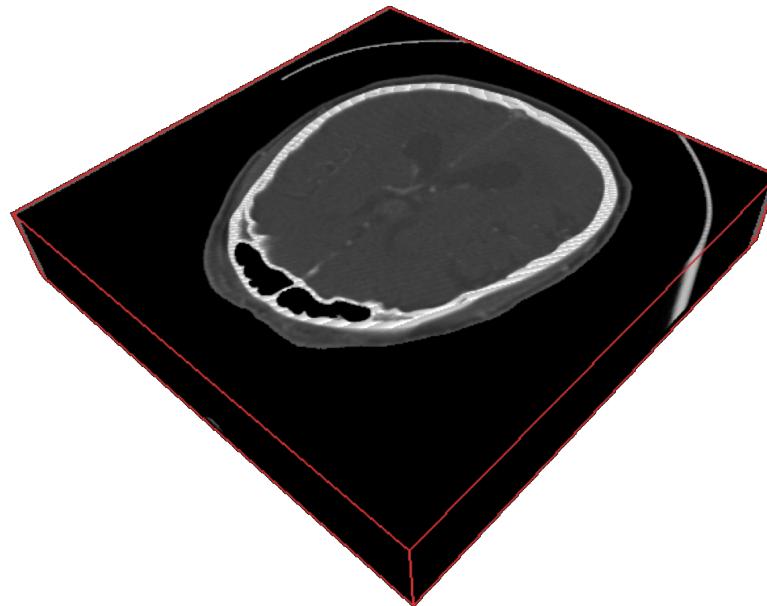


Fundamentals



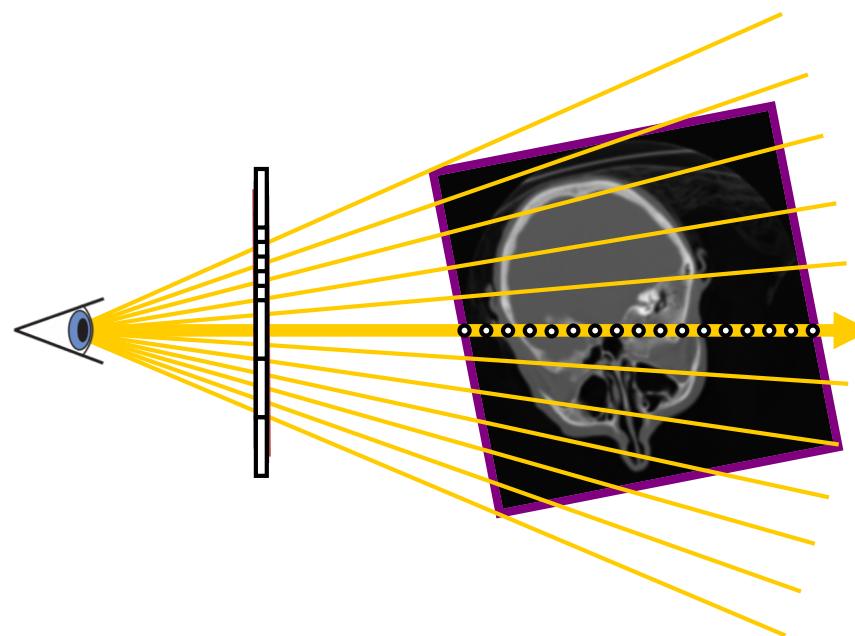
VOLUME RENDERING (1)

Assign optical properties (color, opacity) via *transfer function*



VOLUME RENDERING (2)

Ray-casting





SCALABILITY

Traditional HPC, parallel rendering definitions

- Strong scaling (“more nodes are faster for same data”)
- Weak scaling (“more nodes allow larger data”)

Our interest/definition: output sensitivity

- Running time/storage proportional to size of output instead of input
 - Computational effort scales with visible data and screen resolution
 - Working set independent of original data size



SOME TERMINOLOGY

Output-sensitive algorithms

- Standard term in occlusion culling (of geometry)

Ray-guided volume rendering

- Determine working set via ray-casting
- Actual visibility; not approximate as in traditional occlusion culling

Visualization-driven pipeline

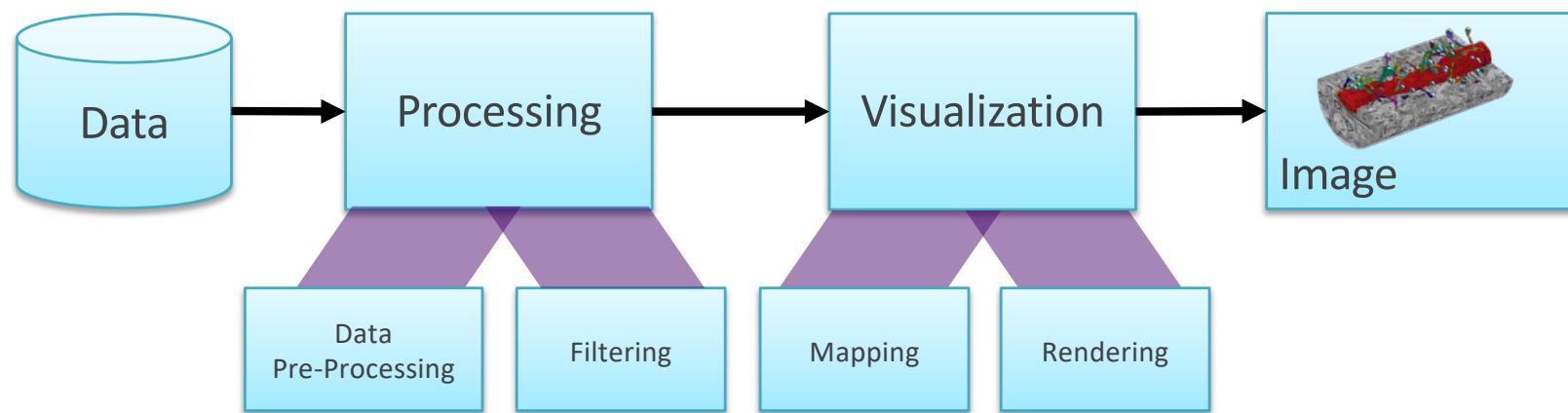
- Drive entire visualization pipeline (including processing) by actual on-screen visibility

Display-aware techniques

- Image processing, ... for current on-screen resolution

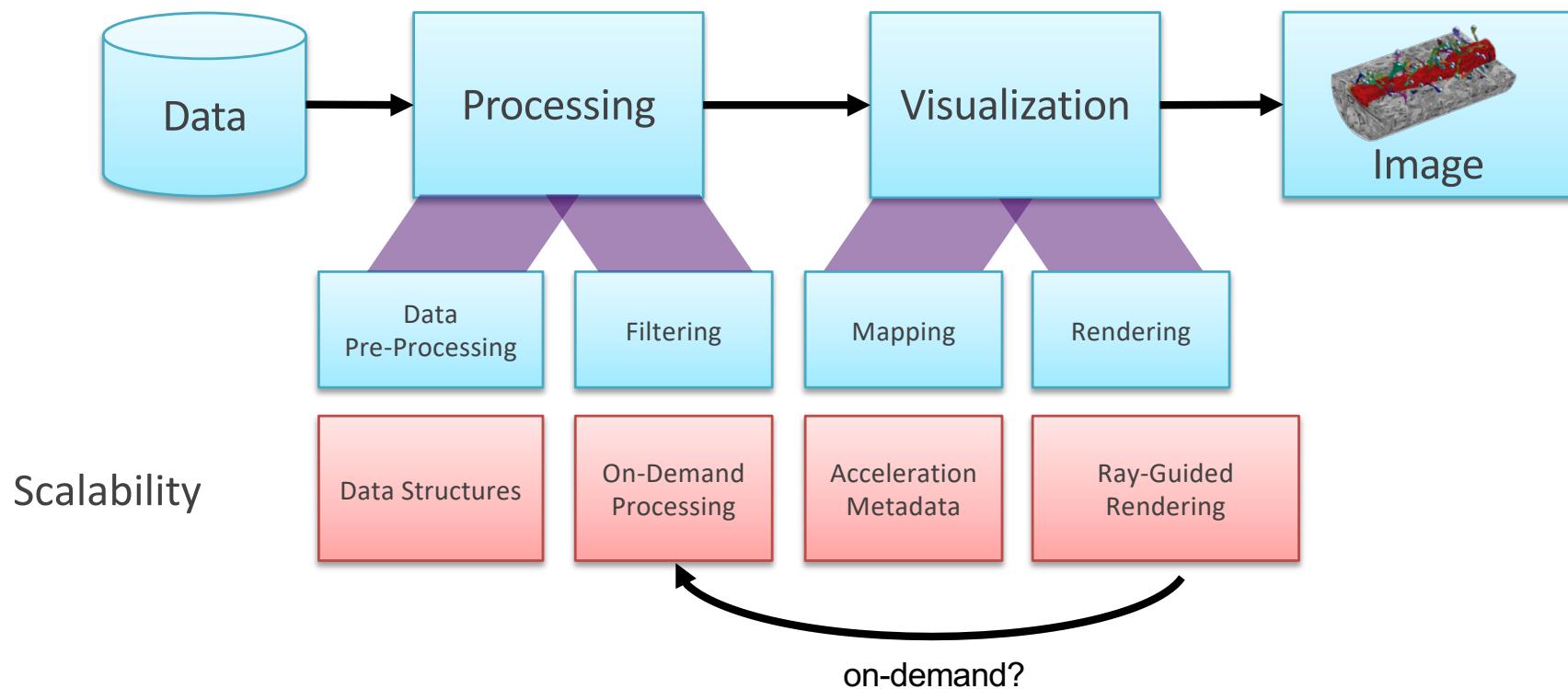


LARGE-SCALE VISUALIZATION PIPELINE





LARGE-SCALE VISUALIZATION PIPELINE





Basic Scalability Issues



SCALABILITY ISSUES

Scalability issues	Scalable method
Data representation and storage	Multi-resolution data structures
	Data layout, compression
Work/data partitioning	In-core/out-of-core
	Parallel, distributed
Work/data reduction	Pre-processing
	On-demand processing
	Streaming
	In-situ visualization
	Query-based visualization



SCALABILITY ISSUES

Scalability issues	Scalable method
Data representation and storage	Multi-resolution data structures Data layout, compression
Work/data partitioning	In-core/out-of-core Parallel, distributed
Work/data reduction	Pre-processing On-demand processing Streaming In-situ visualization Query-based visualization



DATA REPRESENTATIONS

Data structure	Acceleration	Out-of-Core	Multi-Resolution
Mipmaps	-	Clipmaps	Yes
Uniform bricking	Cull bricks (linear)	Working set (bricks)	No
Hierarch. bricking	Cull bricks (hierarch.)	Working set (bricks)	Bricked mipmap
Octrees	Hierarchical traversal	Working set (subtree)	Yes (interior nodes)

Additional issues

- Data layout (linear order, Z/Morton order, ...)
- Compression

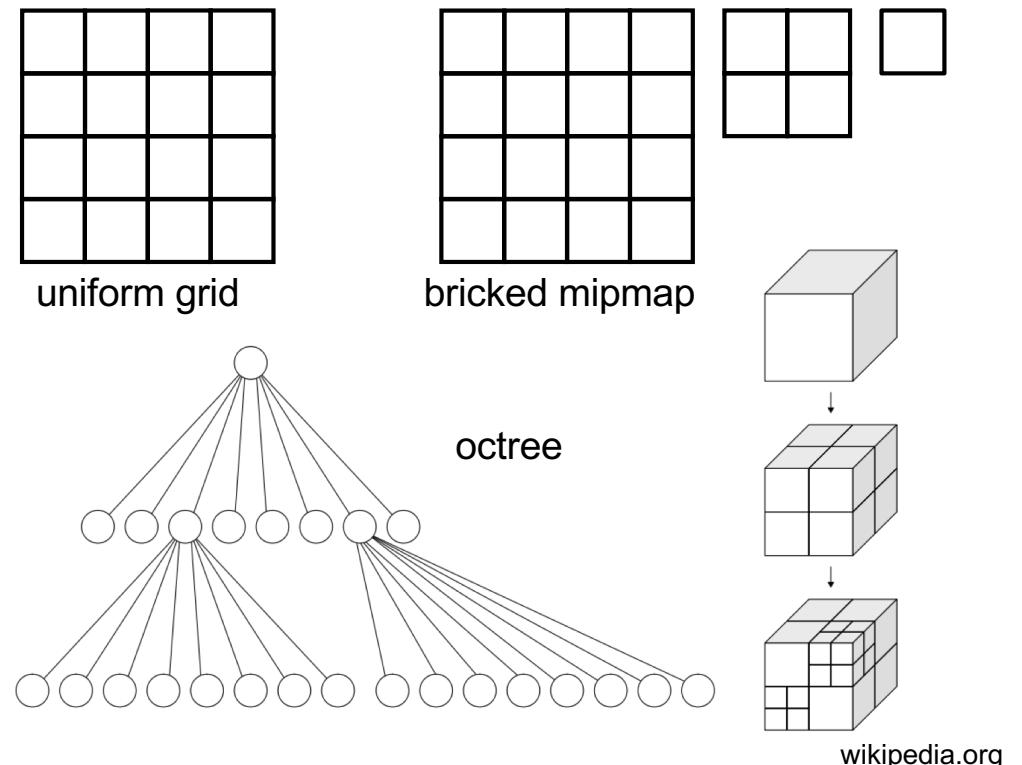
UNIFORM VS. HIERARCHICAL DECOMPOSITION

Grids

- Uniform or non-uniform

Hierarchical data structures

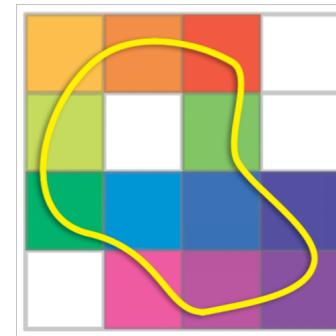
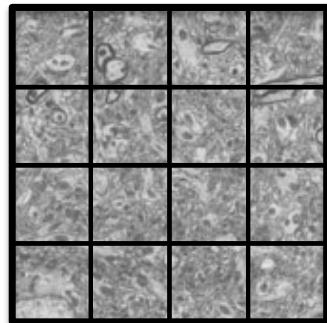
- Pyramid of uniform grids
 - Bricked 2D/3D mipmaps
- Tree structures
 - Quadtree, octree, kd-tree



BRICKING (1)

Object space (data) decomposition

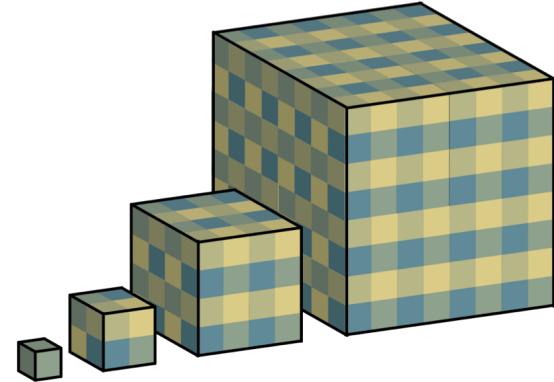
- Subdivide data domain into small bricks
- Re-orders data for spatial locality
- Each brick is now one unit (culling, paging, loading, ...)



BRICKING (2)

What brick size to use?

- Small bricks
 - + Good granularity:
Better culling efficiency, tighter working set, ...
 - More bricks to cull, more overhead for ghost voxels,
one rendering pass per brick is infeasible
- Traditional out-of-core volume rendering: **large** bricks (e.g., 256^3)
- Modern out-of-core volume rendering: **small** bricks (e.g., 32^3)
 - Task-dependent brick sizes
(small for rendering, large for disk/network storage)



Analysis of different brick sizes: [Fogal et al. 2013]

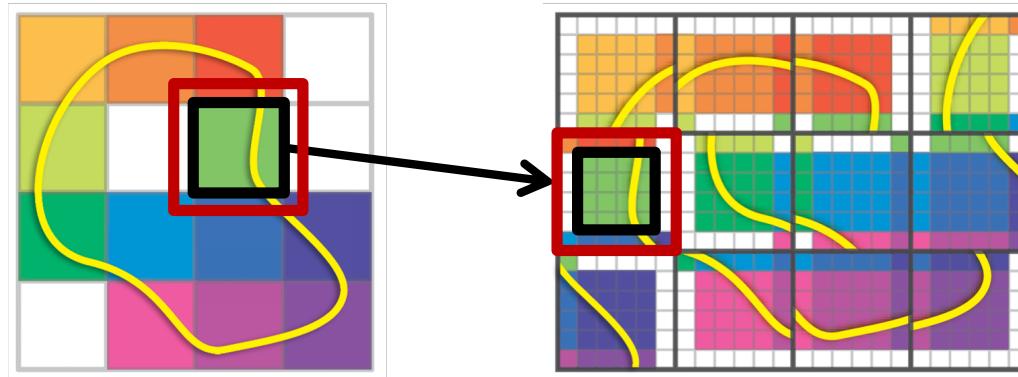
FILTERING AT BRICK BOUNDARIES

Duplicate voxels at border (“ghost” voxels)

- Need at least one voxel overlap
- Large overhead for small bricks

Otherwise costly filtering at brick boundary

- Except with hardware support: *sparse textures*





PRE-COMPUTE ALL BRICKS?

Pre-computation might take very long

- Brick on demand? Brick in streaming fashion (e.g., during scanning)?

Different brick sizes for different tasks (storage, rendering)?

- Re-brick to different size on demand?
- Dynamically fix up ghost voxels?

Can also mix 2D and 3D

- E.g., 2D tiling pre-computed, but compute 3D bricks on demand

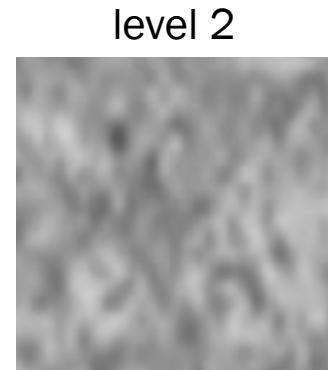
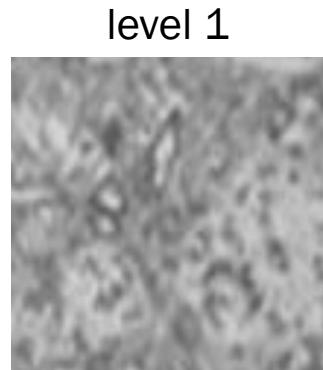
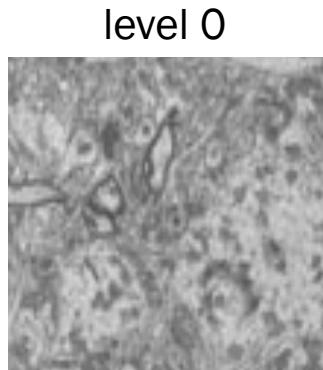
MULTI-RESOLUTION PYRAMIDS (1)

Collection of different resolution levels

- Standard: dyadic pyramids (2:1 resolution reduction)
- Can manually implement arbitrary reduction ratios

Mipmaps

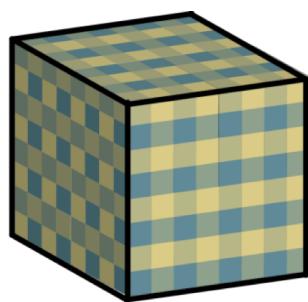
- Isotropic



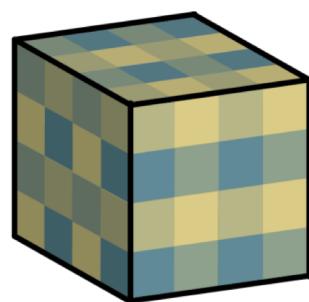
MULTI-RESOLUTION PYRAMIDS (2)

3D mipmaps

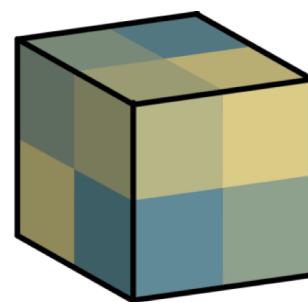
- Isotropic



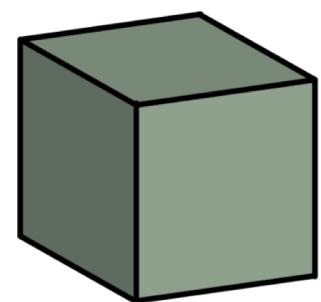
level 0
(8x8x8)



level 1
(4x4x4)



level 2
(2x2x2)

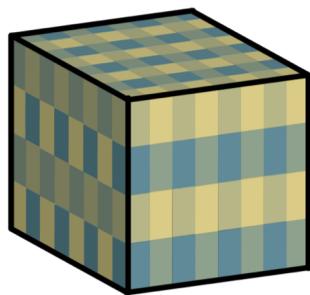


level 3
(1x1x1)

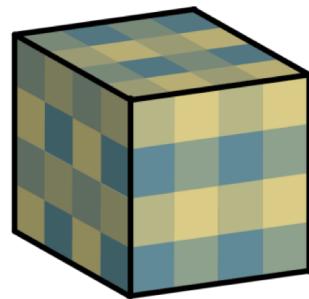
MULTI-RESOLUTION PYRAMIDS (3)

Scanned volume data are often anisotropic

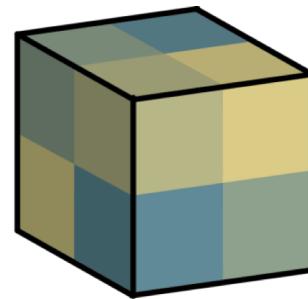
- Reduce resolution anisotropically until isotropy reached



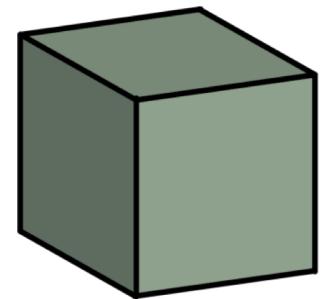
level 0
(8x8x4)



level 1
(4x4x4)



level 2
(2x2x2)

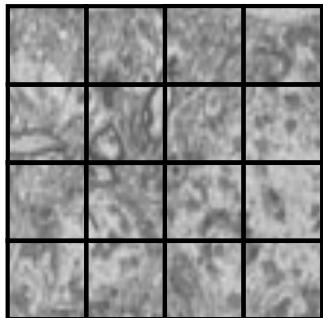


level 3
(1x1x1)

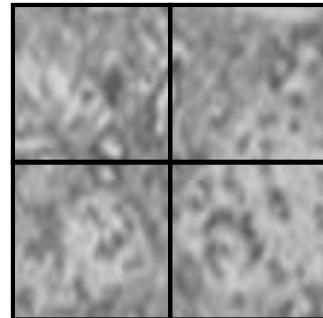
BRICKING MULTI-RESOLUTION PYRAMIDS (1)

Each level is bricked individually

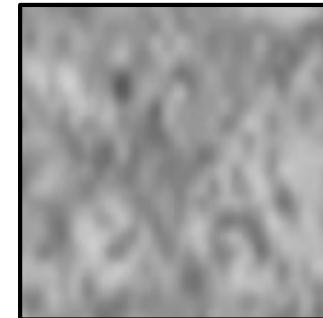
- Use same brick resolution (# voxels) in each level



level 0



level 1



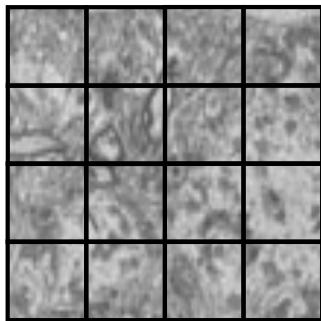
level 2

spatial
extent

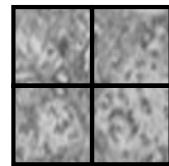
BRICKING MULTI-RESOLUTION PYRAMIDS (2)

Virtual memory: Each brick will be a “page”

- “Multi-resolution virtual memory”: every page lives in some resolution level



4x4 pages



2x2 pages



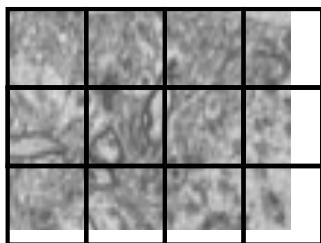
memory extent

1 page

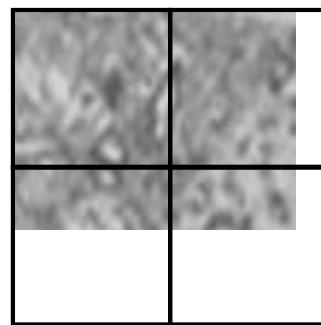
BRICKING MULTI-RESOLUTION PYRAMIDS (3)

Beware of aspect ratio and partially-filled pages

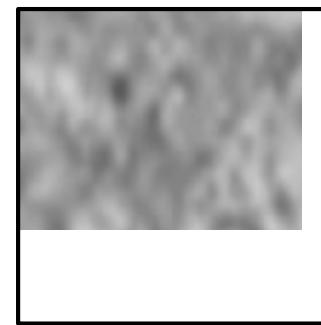
- Reduce total resolution in voxels; compute number of pages (ceil); iterate



4x3 pages



2x2 pages



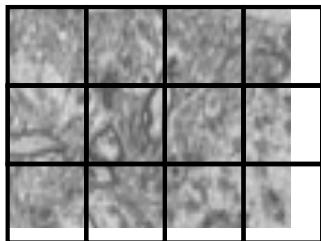
1 page

spatial
extent

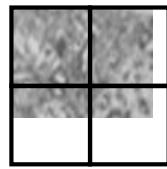
BRICKING MULTI-RESOLUTION PYRAMIDS (3)

Beware of aspect ratio and partially filled pages

- Reduce total resolution in voxels; compute number of pages (ceil); iterate



4x3 pages



2x2 pages



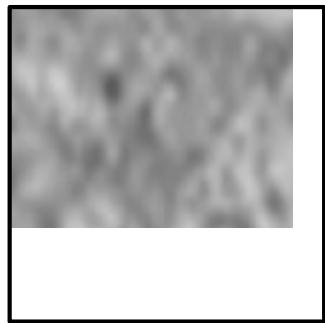
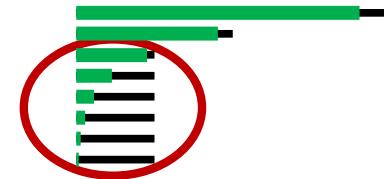
1 page

memory
extent

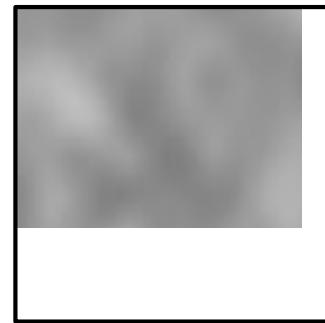
BRICKING MULTI-RESOLUTION PYRAMIDS (4)

Tail of pyramid

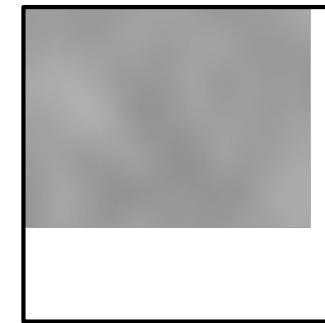
- Below size of single page; can cut off early



1 page



1 page



spatial
extent

1 page

BRICKING MULTI-RESOLUTION PYRAMIDS (4)

Tail of pyramid

- Below size of single page; can cut off early



memory
extent

- **GL_ARB_sparse_texture** treats tail as single unit of residency (implementation-dependent definition of tail !)

1 page

1 page

1 page

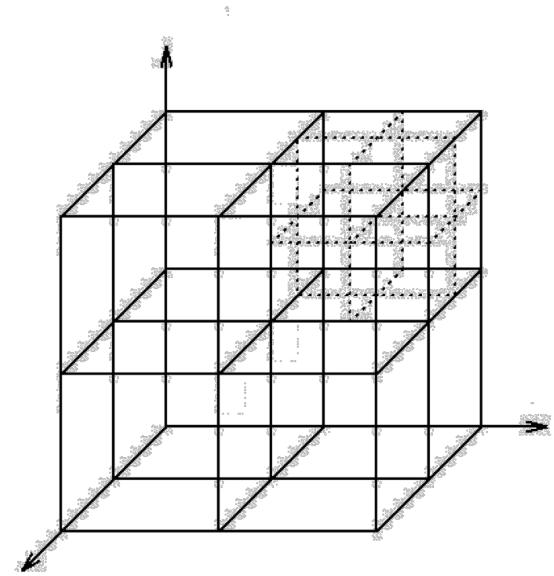
OCTREES FOR VOLUME RENDERING (1)

Multi-resolution

- Adapt resolution of data to screen resolution
 - Reduce aliasing
 - Limit amount of data needed

Acceleration

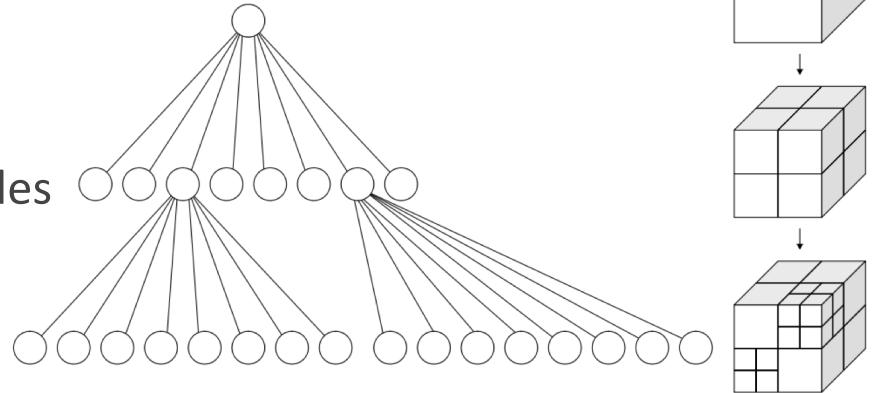
- Hierarchical empty space skipping
- Start traversal at root
 - (but different optimized traversal algorithms:
kd-restart, kd-shortstack, etc.)



OCTREES FOR VOLUME RENDERING (2)

Representation

- Full octree
 - Every octant in every resolution level
- Sparse octree
 - Do not store voxel data of empty nodes



wikipedia.org

Data structure

- Pointer-based
 - Parent node stores pointer(s) to children
- Pointerless
 - Array to index full octree directly



SCALABILITY ISSUES

Scalability issues	Scalable method
Data representation and storage	Multi-resolution data structures Data layout, compression
Work/data partitioning	In-core/out-of-core Parallel, distributed
Work/data reduction	Pre-processing On-demand processing Streaming In-situ visualization Query-based visualization



WORK/DATA PARTITIONING

- Out-of-core techniques
- Domain decomposition
- Parallel and distributed rendering

OUT-OF-CORE TECHNIQUES (1)

Data too large for GPU memory

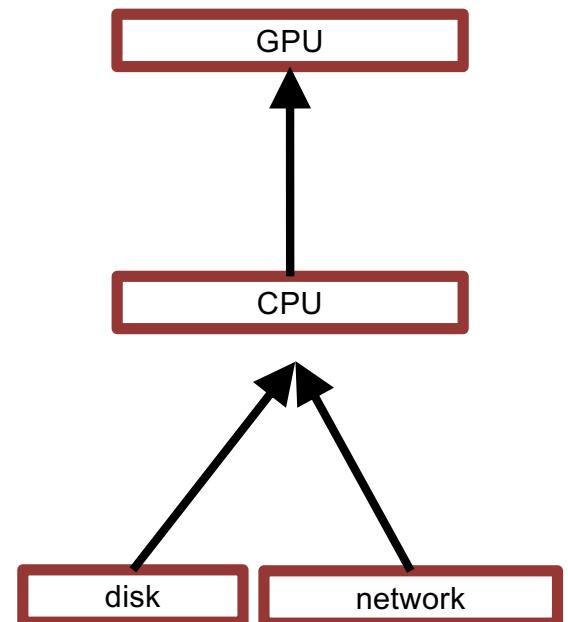
- Stream volume bricks from CPU to GPU on demand

Data too large for CPU memory

- Stream volume bricks from disk on demand

Data too large for local disk storage

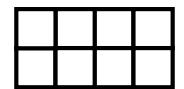
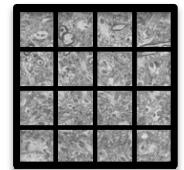
- Stream volume bricks from network storage



OUT-OF-CORE TECHNIQUES (2)

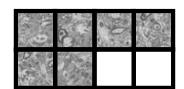
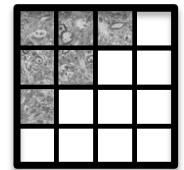
Preparation

- Subdivide spatial domain
 - May also be done “virtually”, i.e., data re-ordering may be delayed
- Allocate cache memory (e.g., large 3D cache texture)



Run-Time

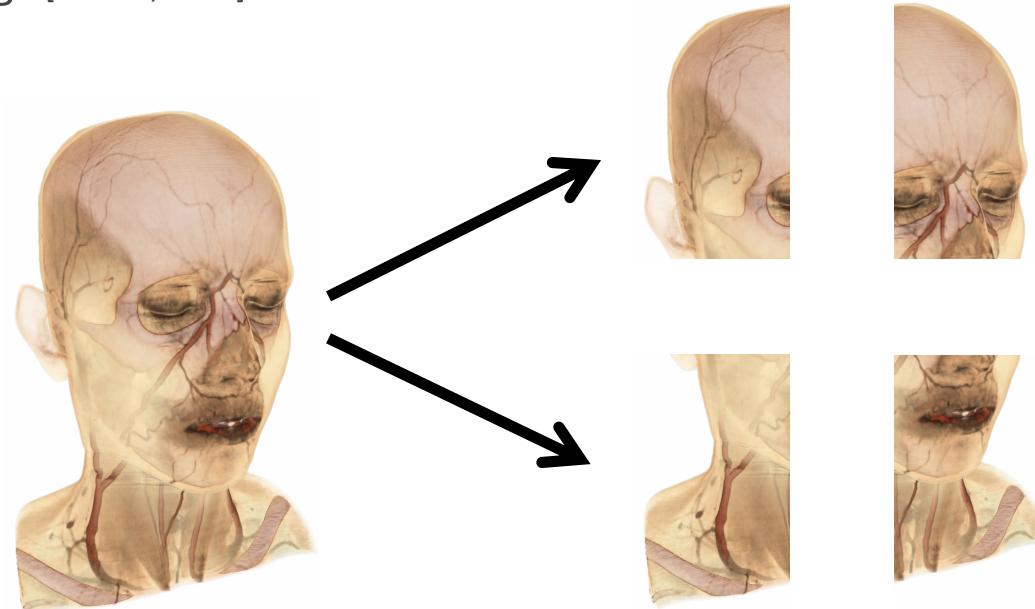
- Determine **working set**
- Page working set into cache memory
- Render from cache memory



DOMAIN DECOMPOSITION (1)

Subdivide image domain (image space)

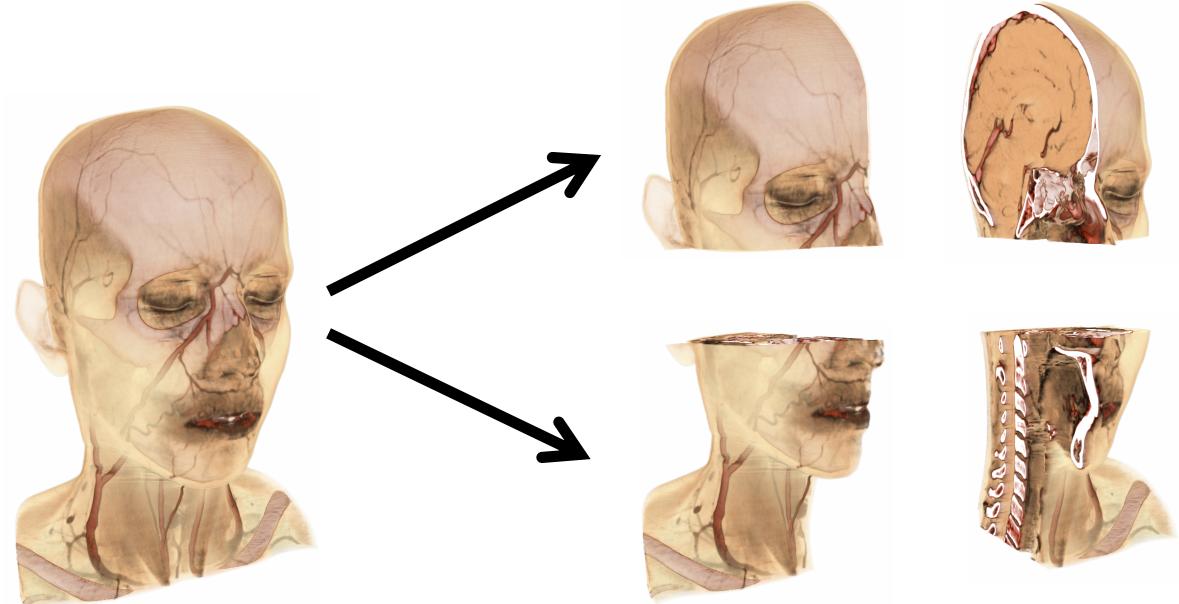
- “Sort-first rendering” [Molnar, 1994]
- View-dependent



DOMAIN DECOMPOSITION (2)

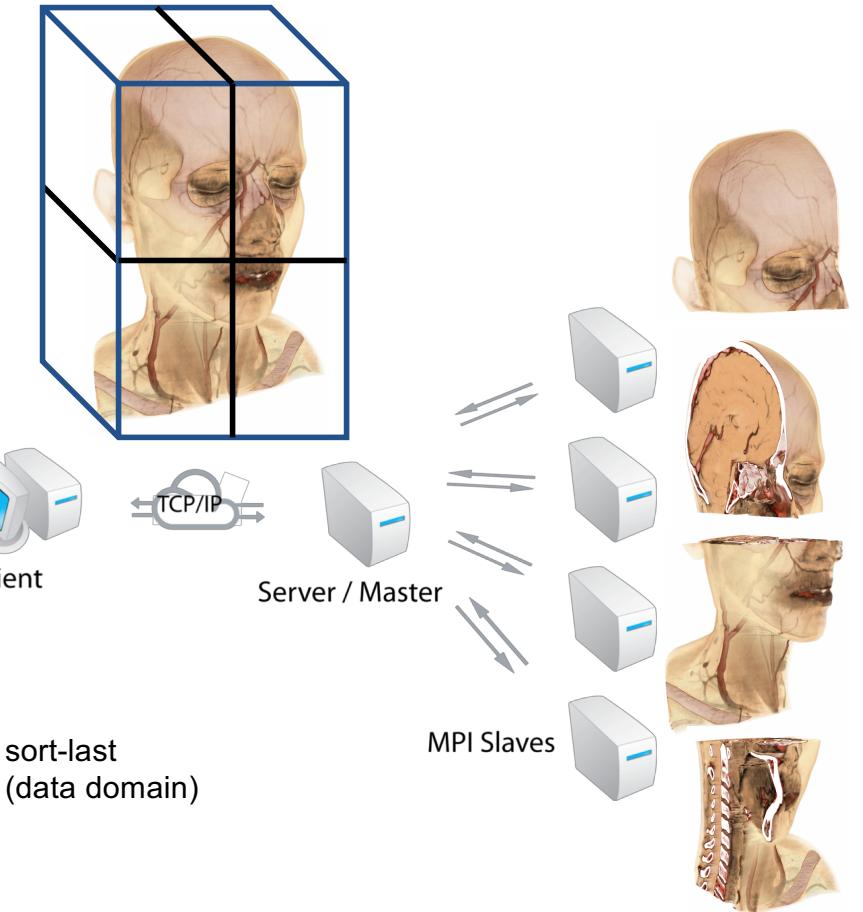
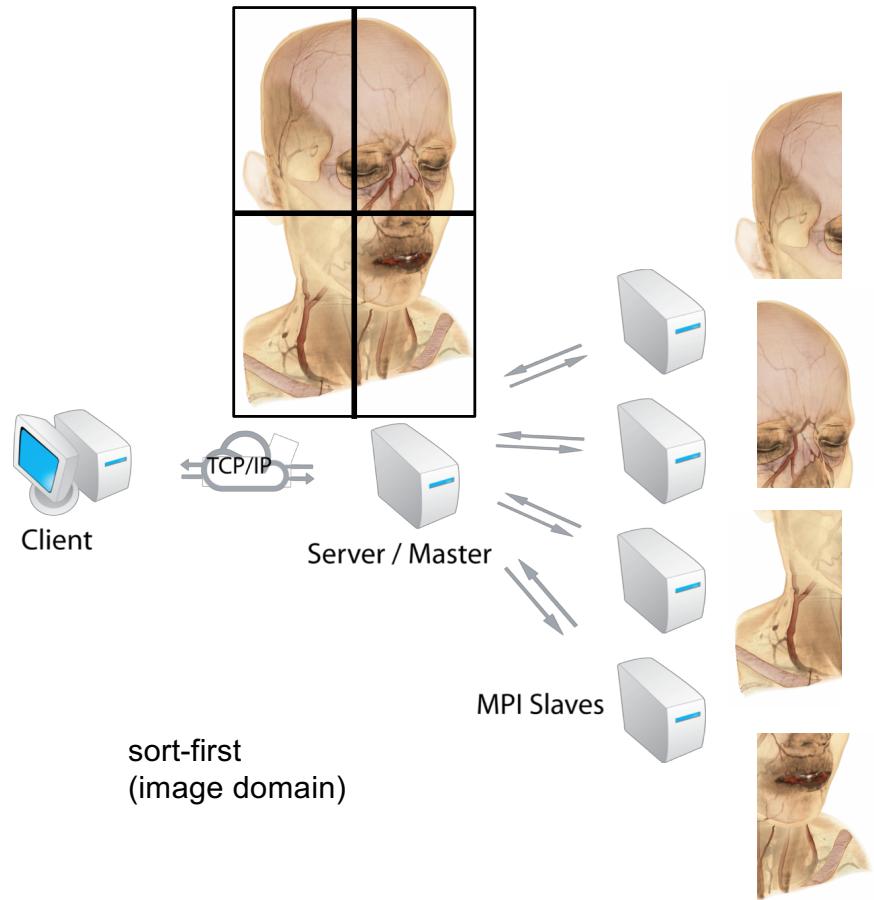
Subdivide data domain (object space)

- “Sort-last rendering” [Molnar, 1994]
- View-independent





SORT-FIRST VS. SORT-LAST





SCALABILITY ISSUES

Scalability issues	Scalable method
Data representation and storage	Multi-resolution data structures Data layout, compression
Work/data partitioning	In-core/out-of-core Parallel, distributed
Work/data reduction	Pre-processing On-demand processing Streaming In-situ visualization Query-based visualization



ON-DEMAND PROCESSING

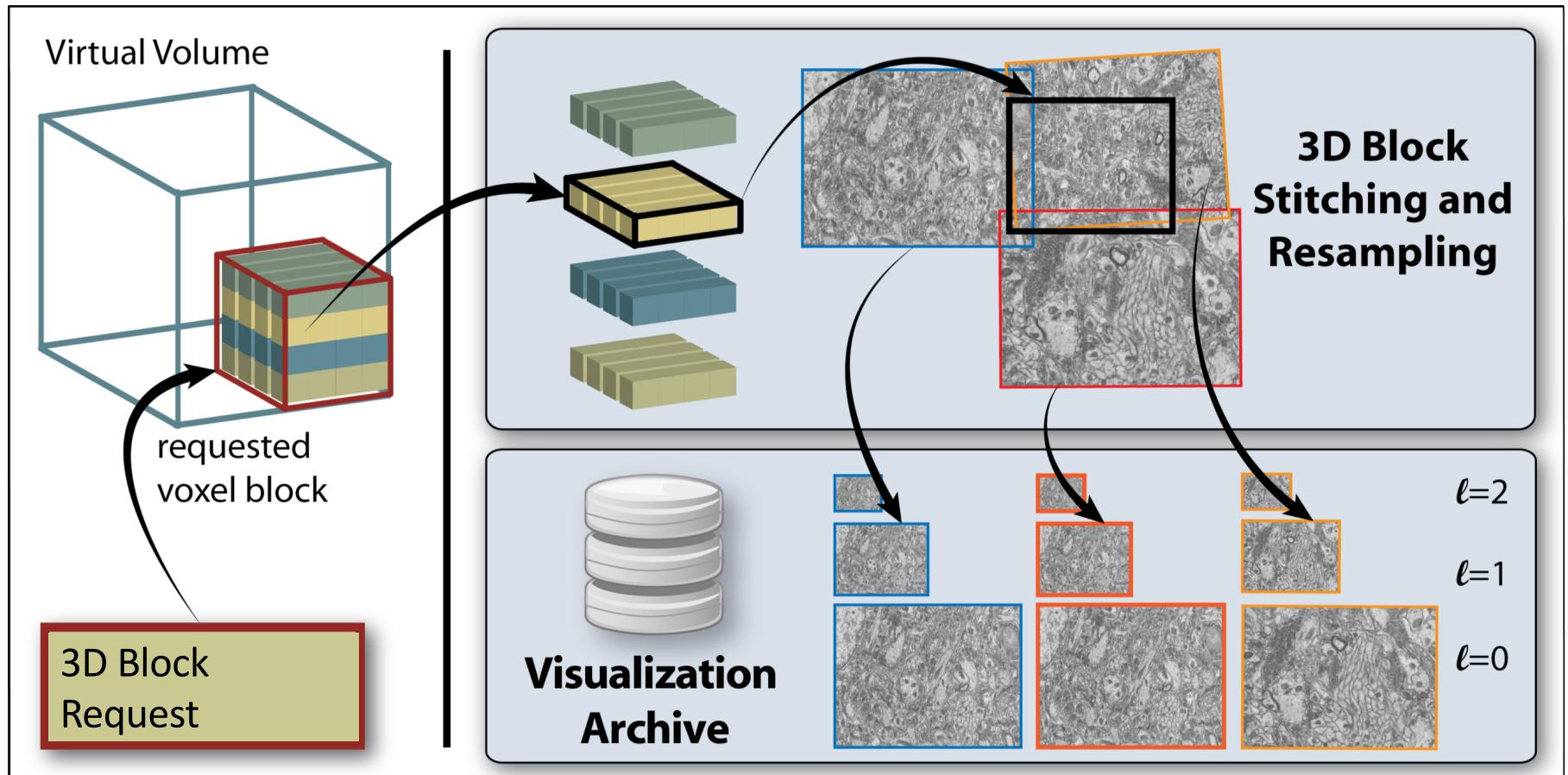
First determine what is visible / needed: working set

Then process only this working set

- Basic processing
 - Noise removal and edge detection
 - Registration and alignment
 - Segmentation, ...
- Basic data structure building
 - Construct pages/bricks/octree nodes only on demand?

EXAMPLE: 3D BRICK CONSTRUCTION FROM 2D EM STREAMS

[Hadwiger et al., IEEE Vis 2012]



EXAMPLE: DENOISING & EDGE ENHANCEMENT

Edge enhancement for EM data

Caching scheme

- Process only currently visible bricks
- Cache result for re-use

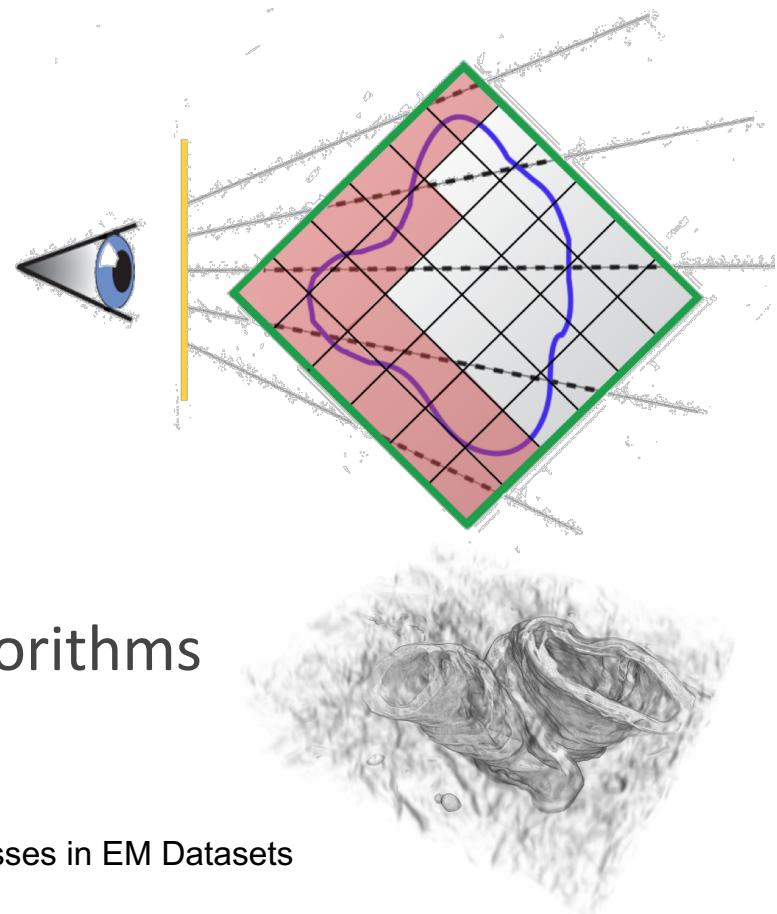
GPU Implementation

- CUDA and shared memory for fast computation

Different noise removal and filtering algorithms

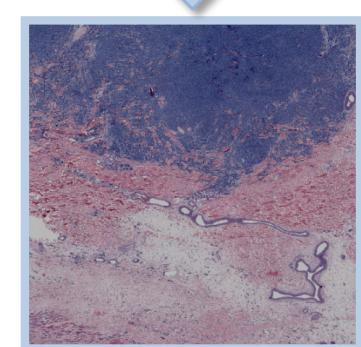
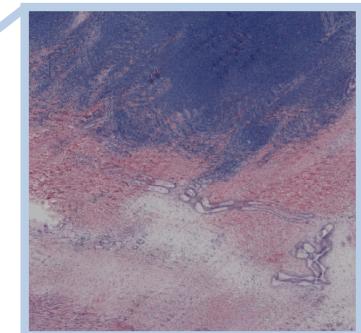
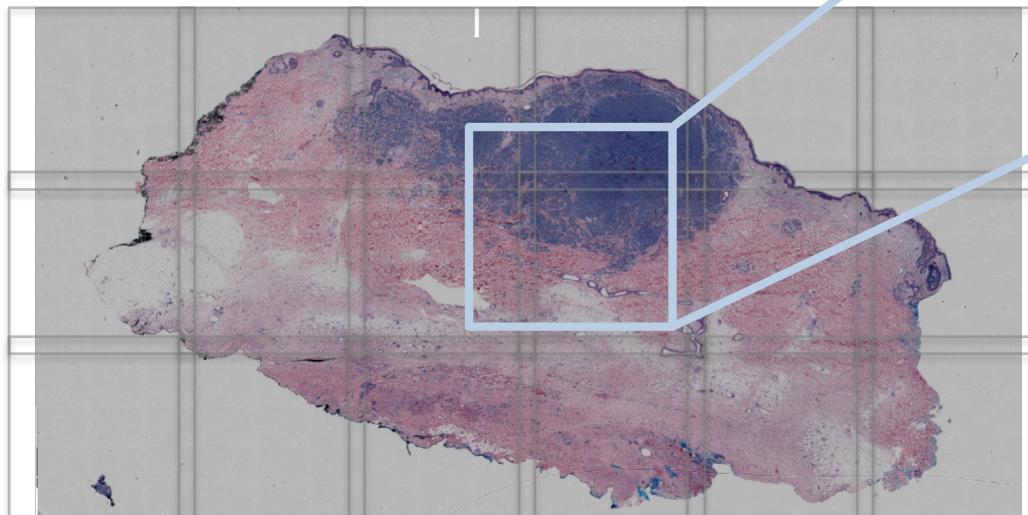
[Jeong et al., IEEE Vis 2009]

Scalable and Interactive Segmentation and Visualization of Neural Processes in EM Datasets



EXAMPLE: REGISTRATION & ALIGNMENT

Registration at screen/brick resolution



[Beyer et al., CG&A 2013]

Exploring the Connectome – Petascale Volume Visualization of Microscopy Data Streams



Questions?