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

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 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 [2:15pm – 3:15pm]

- Part 2 – Scalable Volume Visualization Architectures
  Johanna Beyer [3:15pm – 4:00pm]

- Break
  [4:00pm – 4:15pm]

COURSE OVERVIEW - SCHEDULE

- Part 3 – GPU-Based Ray-Guided Volume Rendering
  Johanna Beyer [4:15pm – 5:15pm]
- Part 4 – Display-Aware Visualization and Processing
  Markus Hadwiger [5:15pm – 5:45pm]
- Wrap-Up, Summary
  Johanna Beyer, Markus Hadwiger [5:45pm – 6:00pm]

Part 1 -
Introduction & Basics of
Scalable Volume Visualization

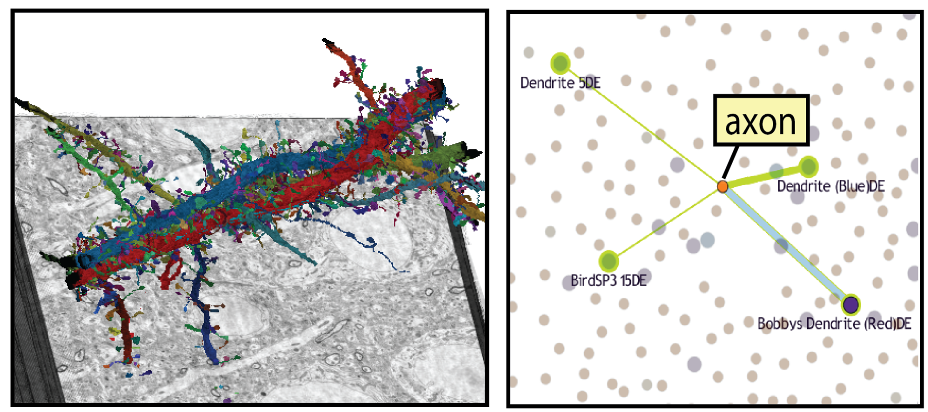
Motivation

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 main interest:
Very large 3D volume data

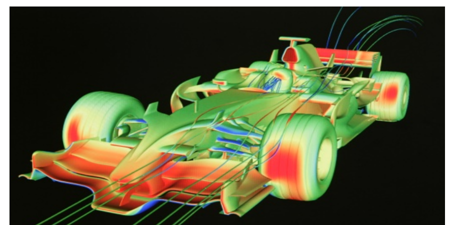


Example: Connectomics (neuroscience)

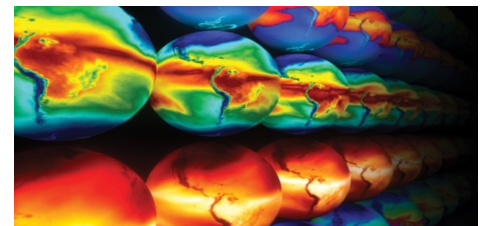
DATA-DRIVEN SCIENCE (E-SCIENCE)

MEDICINE
Digital Health Records

BIOLOGY
Connectomics



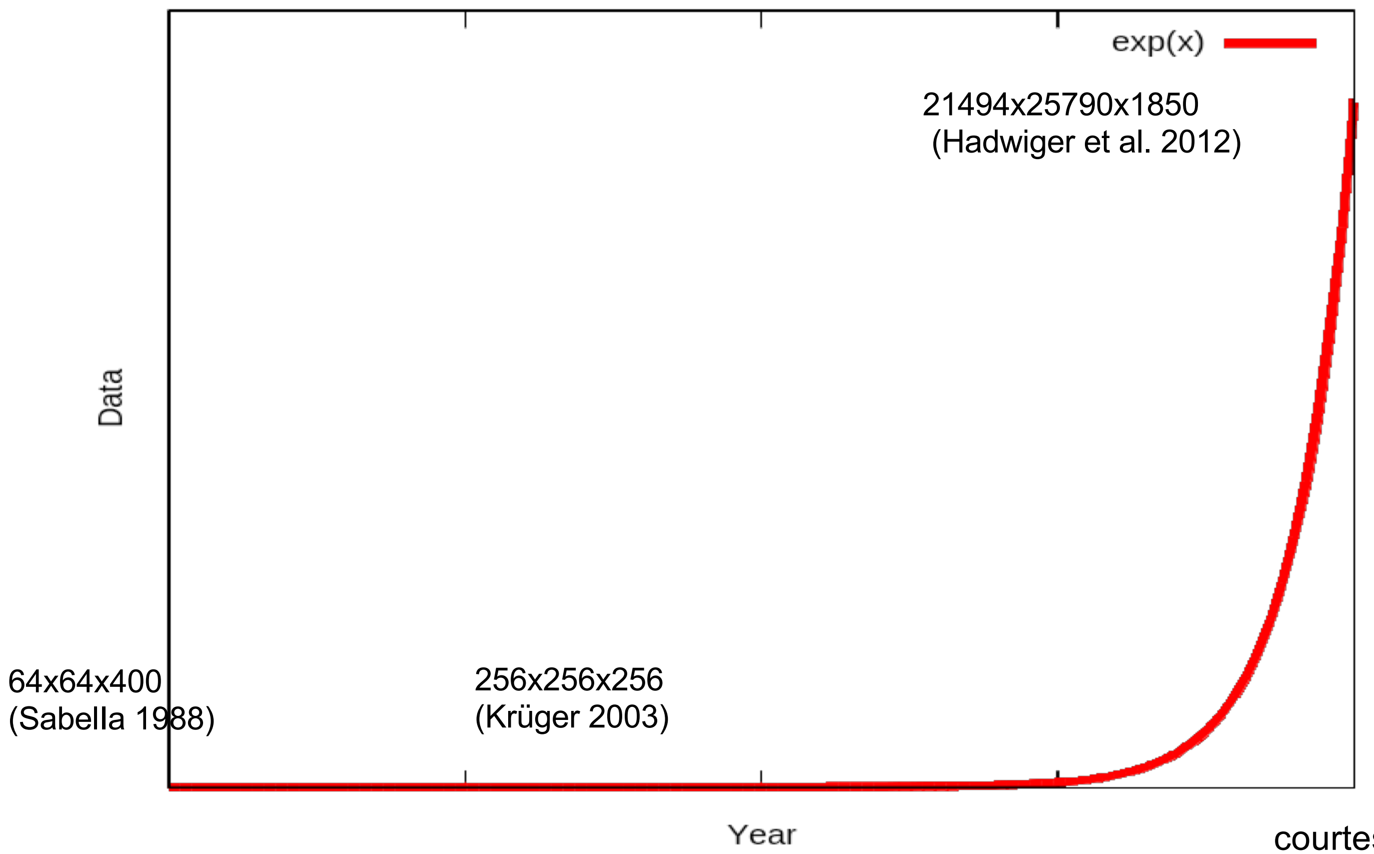
ENGINEERING
Large CFD Simulations



EARTH SCIENCES
Global Climate Models

courtesy Stefan Bruckner

VOLUME DATA GROWTH



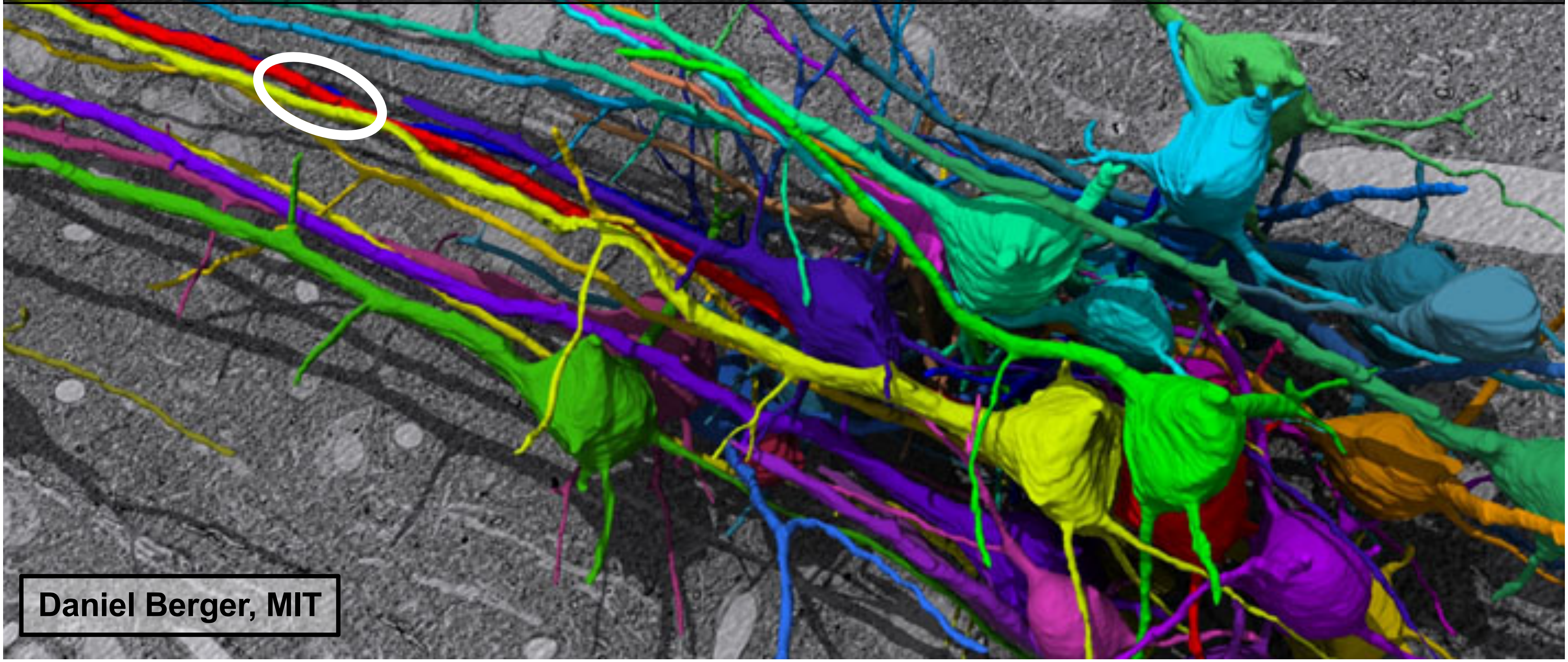
courtesy Jens Krüger

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 et al.	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

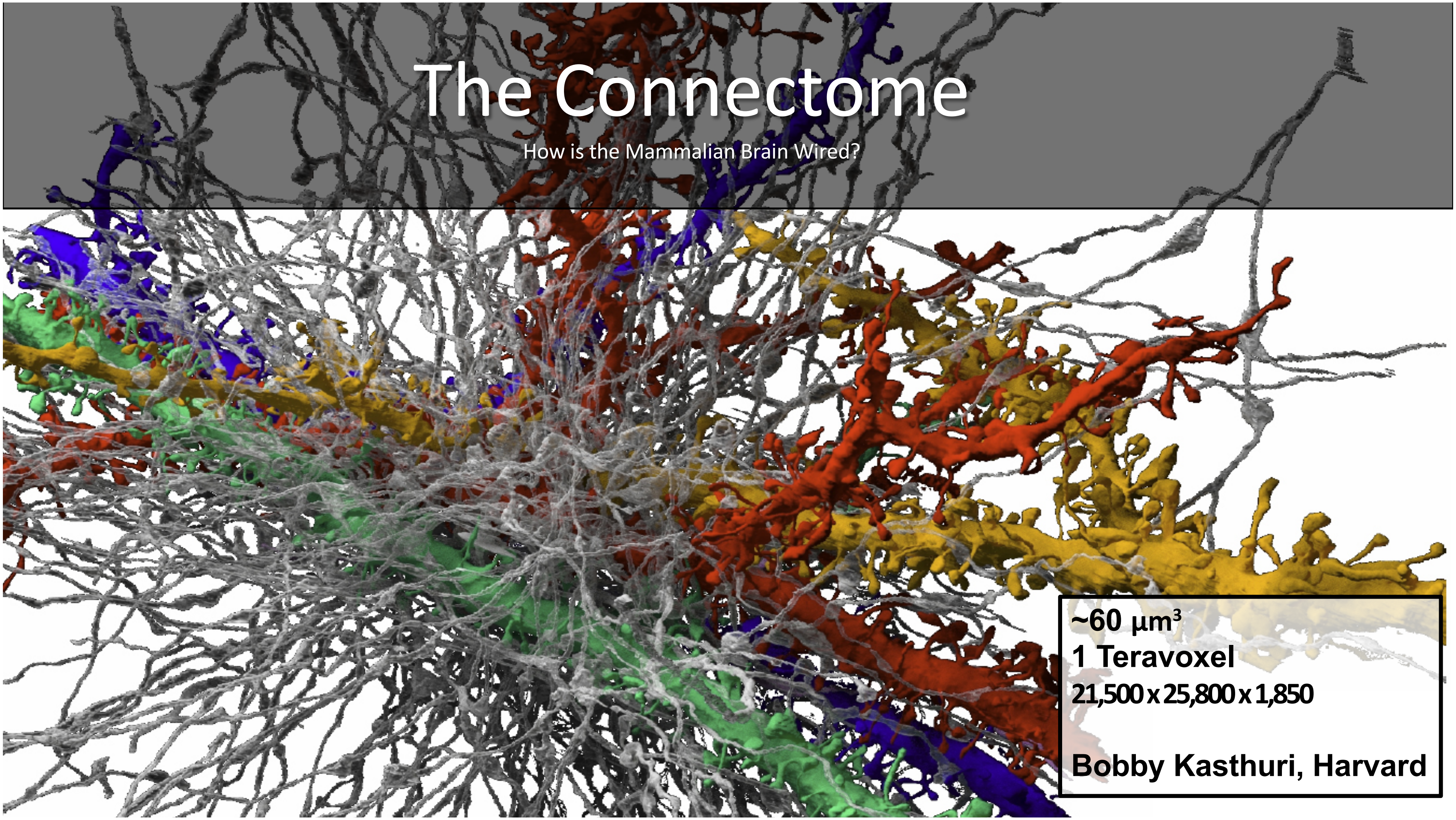
The Connectome

How is the Mammalian Brain Wired?

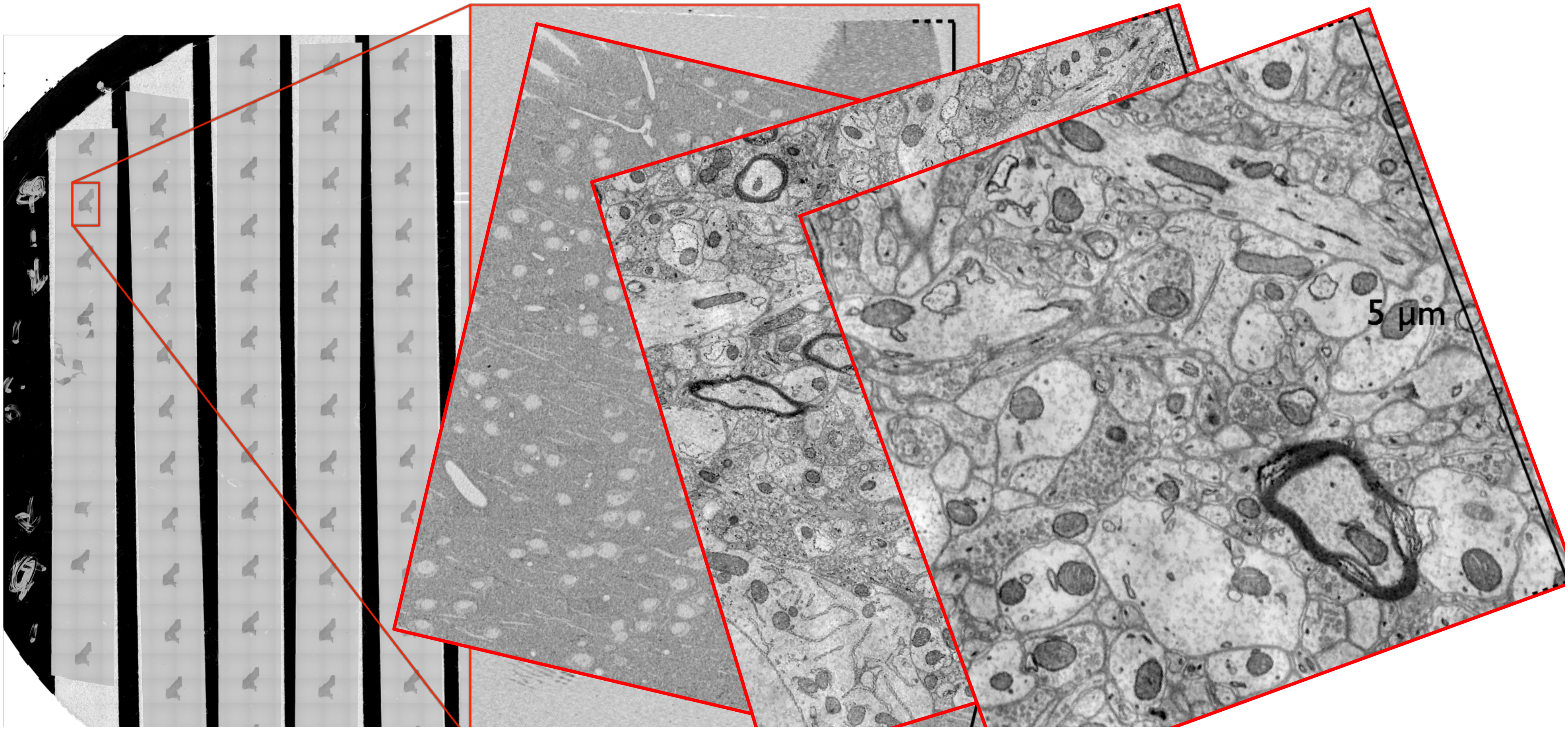


The Connectome

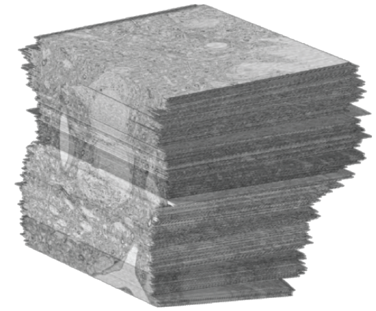
How is the Mammalian Brain Wired?



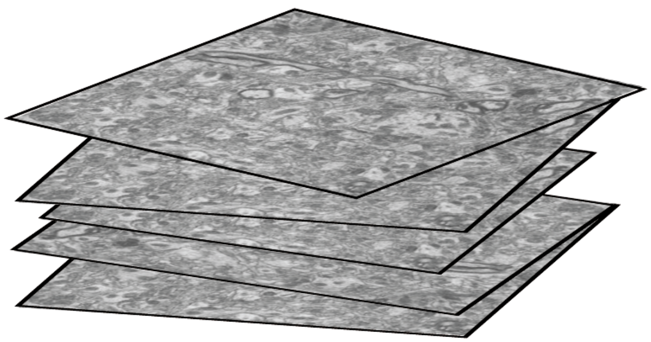
ELECTRON MICROSCOPY (EM) IMAGES

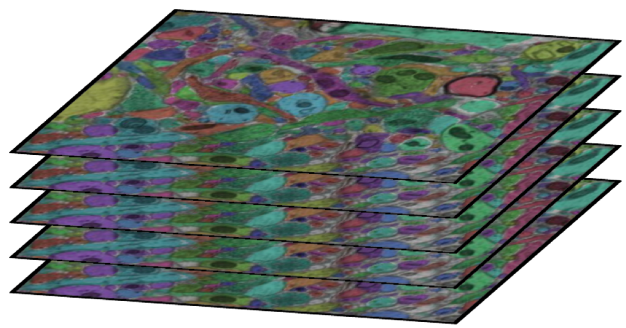


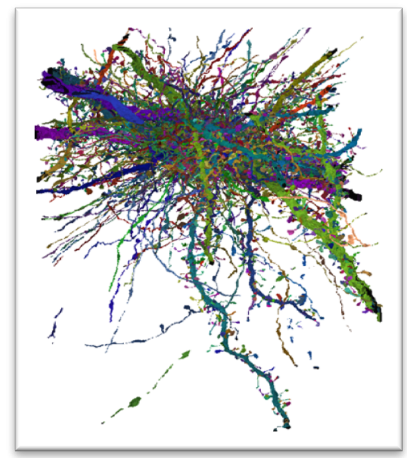
PETAVOXEL MICROSCOPY VOLUMES

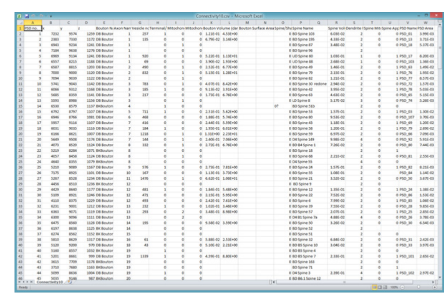


- Huge amount of raw data (terabytes to petabytes)
- Takes months to years to scan, align, segment
- How to visualize and analyze this?









COURSE SCOPE

Course focus

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

Techniques orthogonal to this course (will not cover details)

- Parallel and distributed rendering, clusters, supercomputers, …
- Compression (encoding, decoding, ...)

RELATED BOOKS AND SURVEYS

Books

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

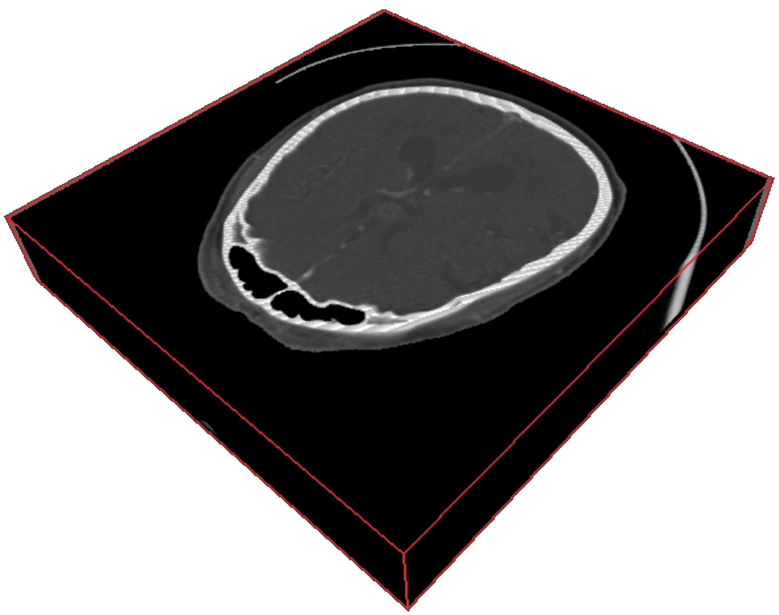
Surveys

- 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 Performace Comp.: Bauer et al. ’16
- State of the art in transfer functions for direct volume rendering: Ljung et al. ‘16

Fundamentals

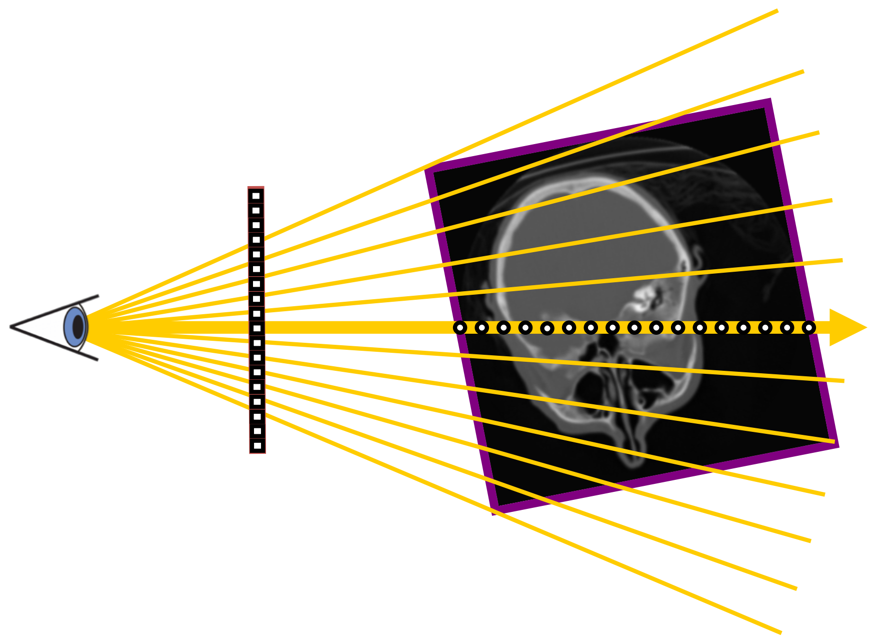
VOLUME RENDERING (1)

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

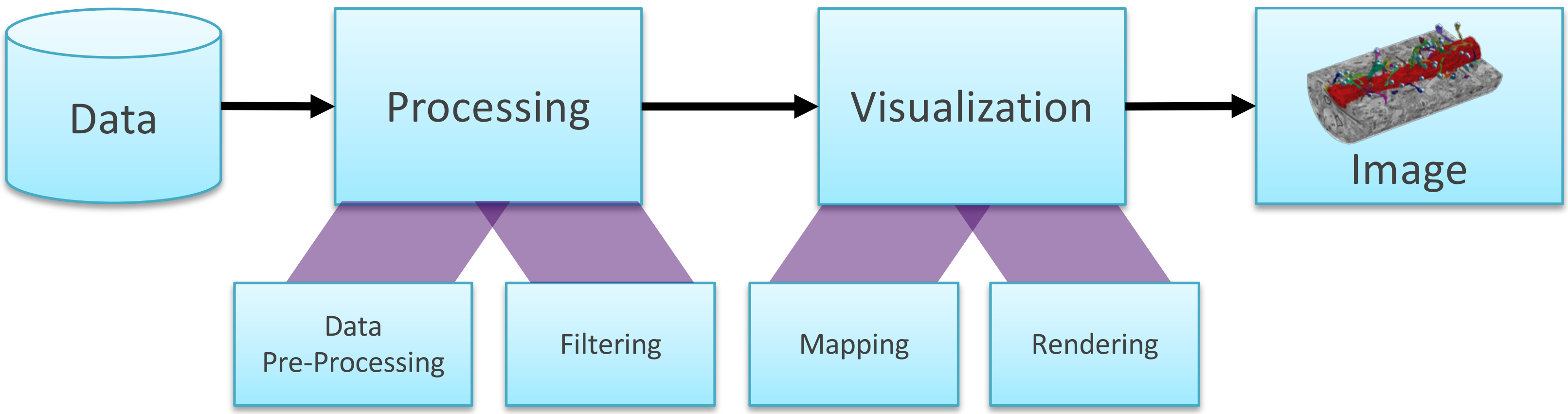
Visualization-driven pipeline

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

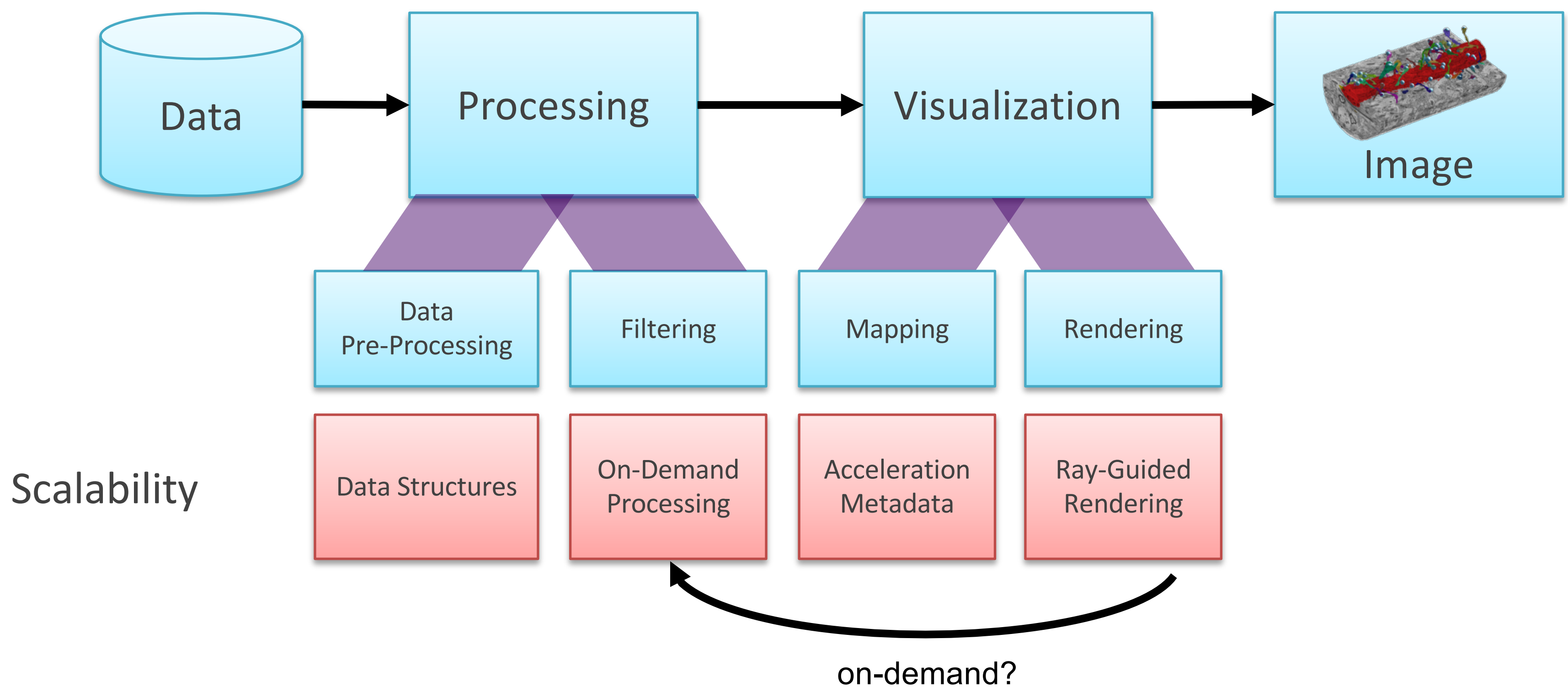
Display-aware techniques

- 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

SIGGRAPH ASIA 2018 TOKYO

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

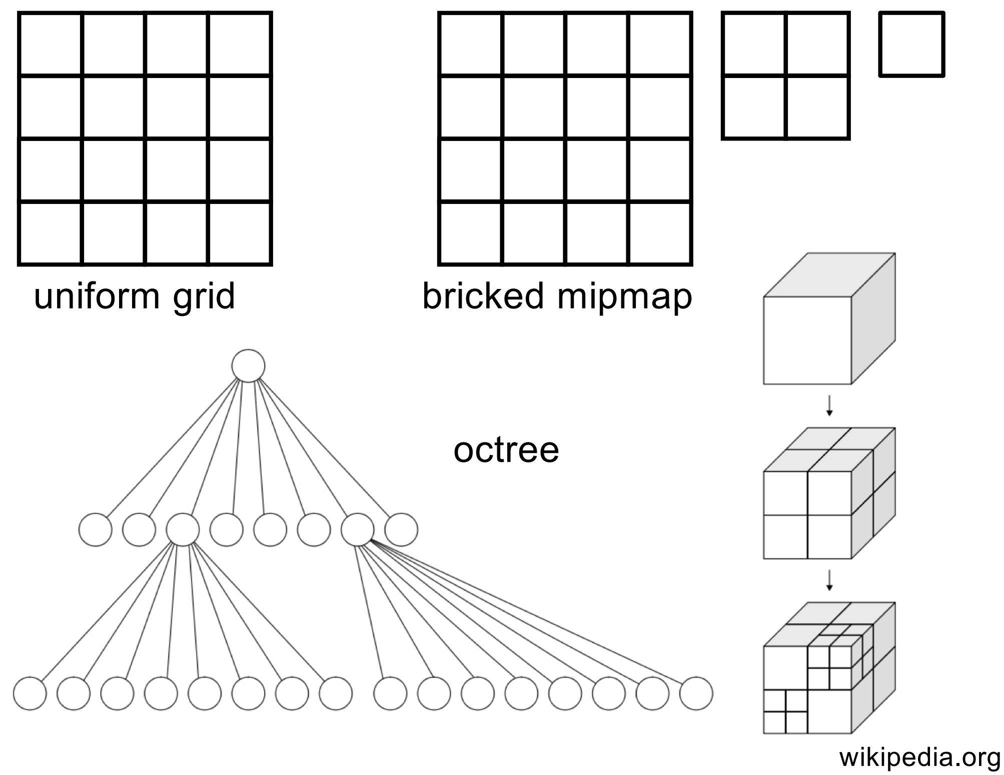
UNIFORM VS. HIERARCHICAL DATA DECOMPOSITION

Grids

- Uniform or non-uniform

Hierarchical data structures

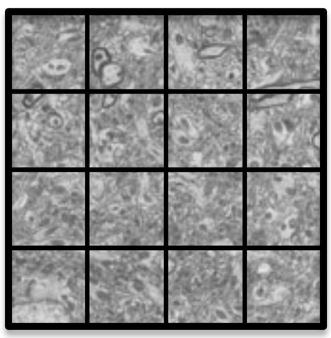
- Pyramid of uniform grids
  - Bricked 2D/3D mipmaps
- Tree structures
  - Quadtree, octree, kd-tree

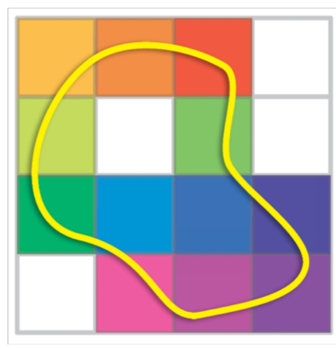


BRICKING (1)

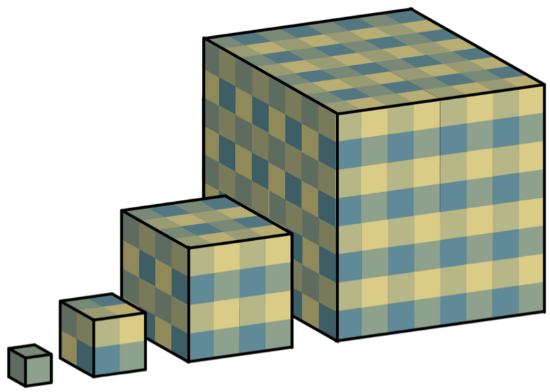
Object space (data) decomposition

- 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]

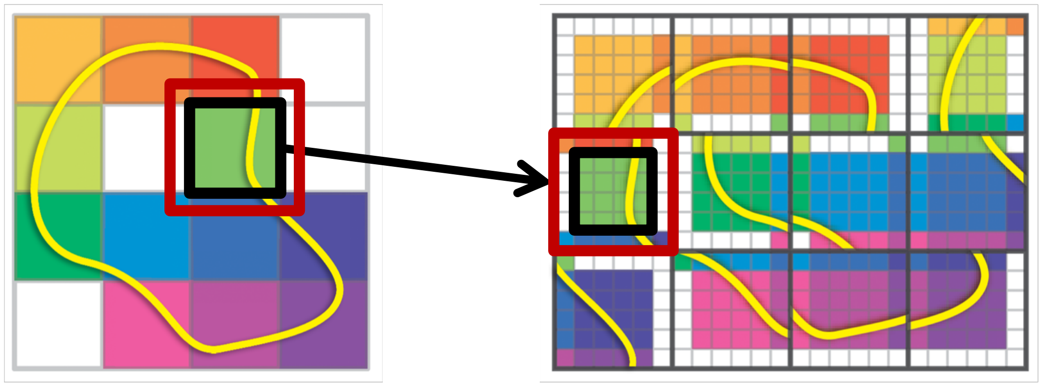
FILTERING AT BRICK BOUNDARIES

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: OpenGL sparse textures / Vulkan sparse images



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

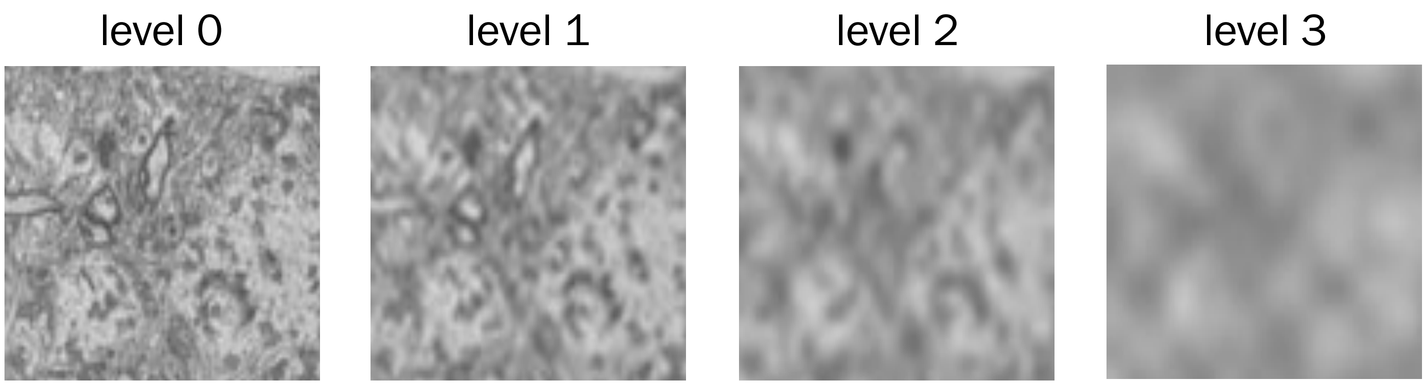
MULTI-RESOLUTION PYRAMIDS (1)

Collection of different resolution levels

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

Mipmaps

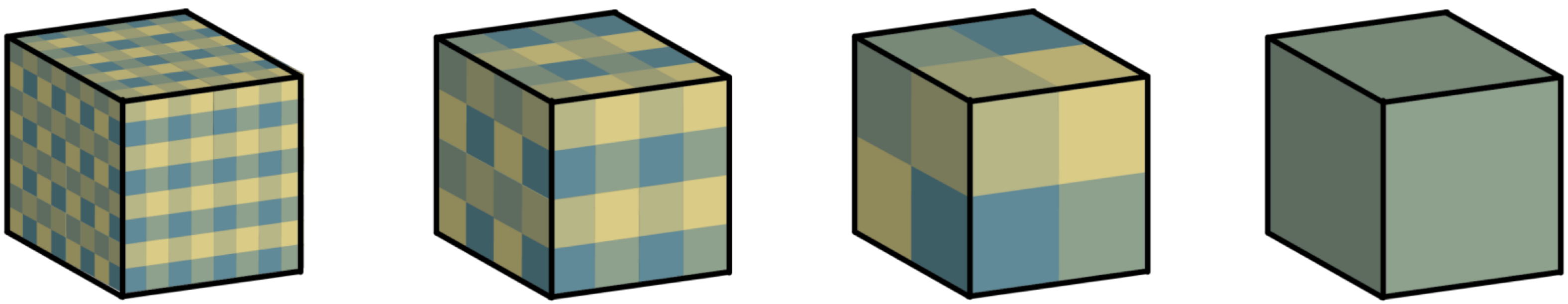
- Isotropic



MULTI-RESOLUTION PYRAMIDS (2)

3D mipmaps

- Isotropic



level 0
(8x8x8)

level 1
(4x4x4)

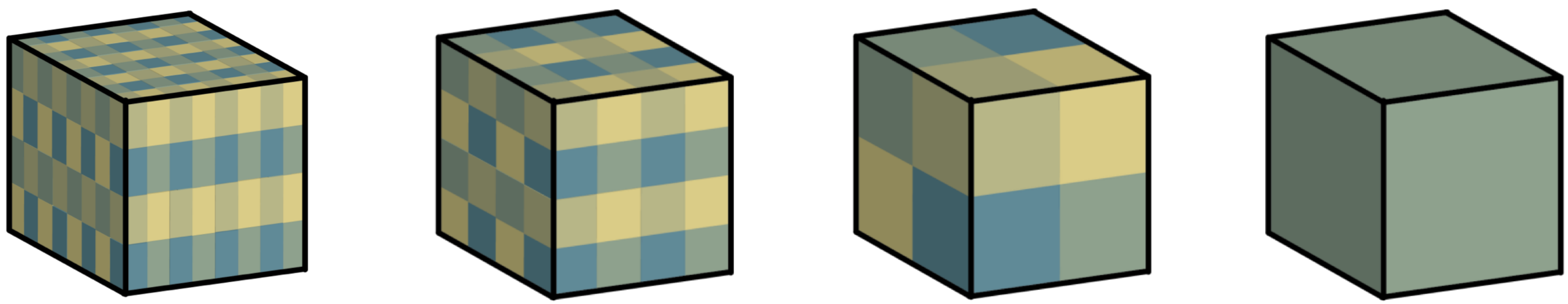
level 2
(2x2x2)

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)

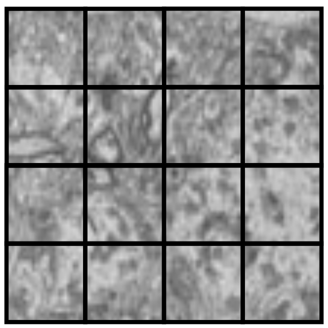
level 2
(2x2x2)

level 3
(1x1x1)

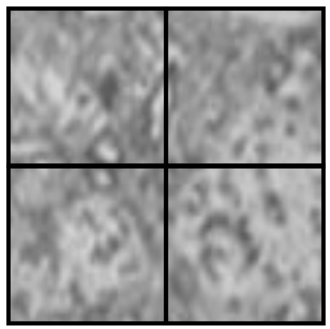
BRICKING MULTI-RESOLUTION PYRAMIDS (1)

Each level is bricked individually

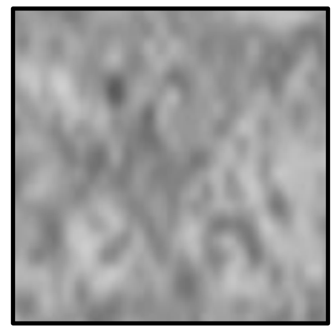
- Use same brick resolution (# voxels) in each level



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

memory extent

4x4 pages

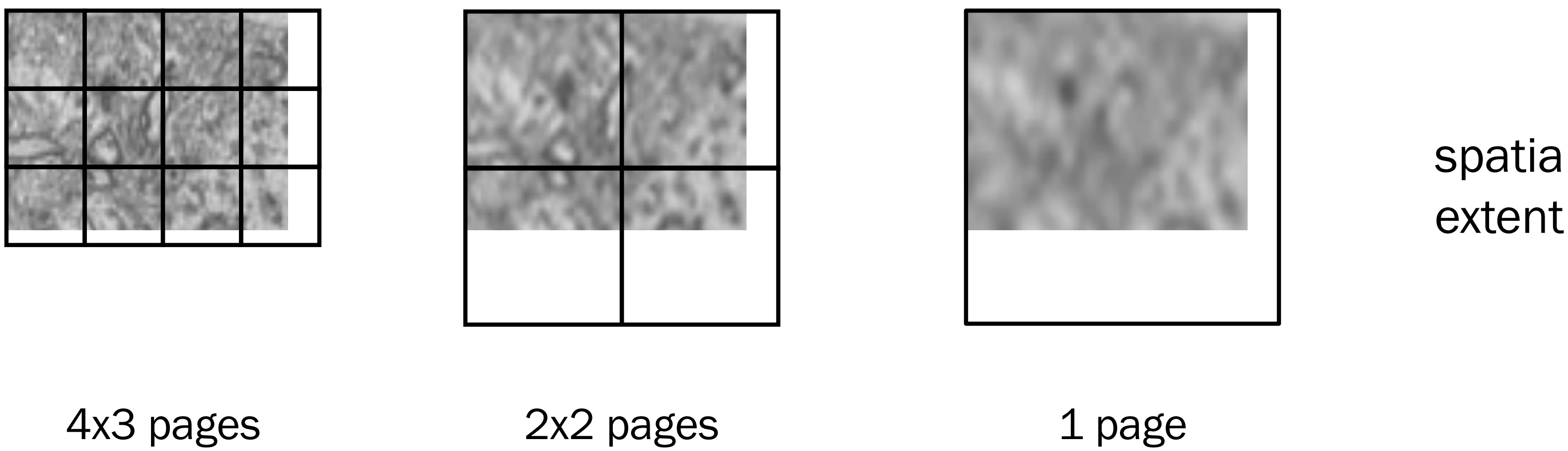
2x2 pages

1 page

BRICKING MULTI-RESOLUTION PYRAMIDS (3)

Beware of aspect ratio and partially-filled pages

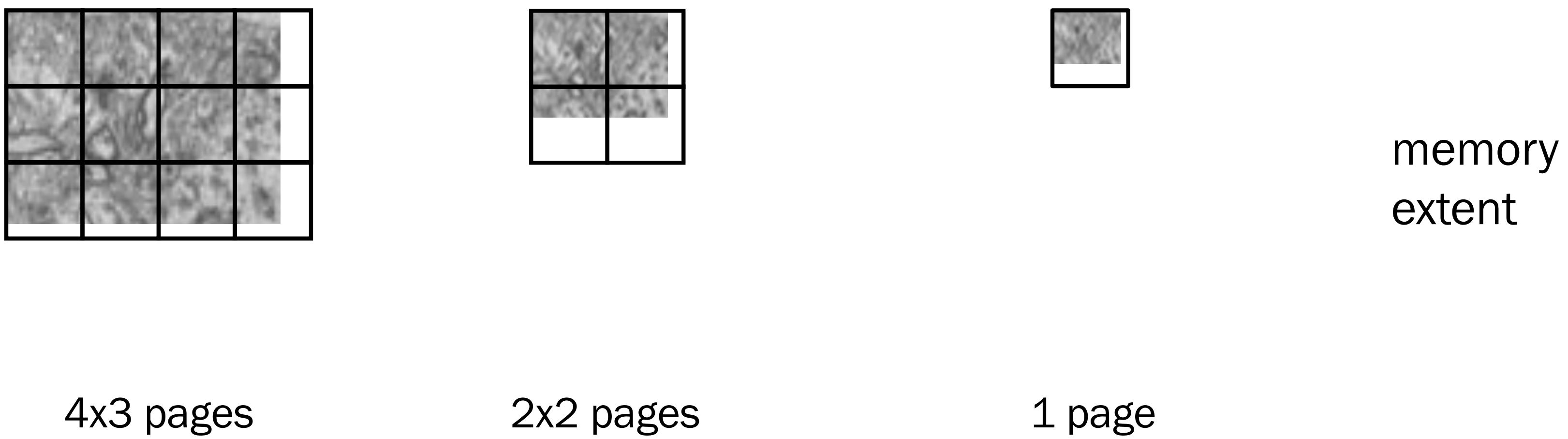
- Reduce total resolution in voxels; compute number of pages (ceil); iterate



BRICKING MULTI-RESOLUTION PYRAMIDS (3)

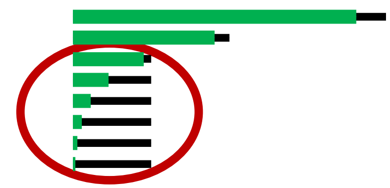
Beware of aspect ratio and partially-filled pages

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



SIGGRAPH ASIA 2018 TOKYO

BRICKING MULTI-RESOLUTION PYRAMIDS (4)



Tail of pyramid

- Below size of single page; can cut off early

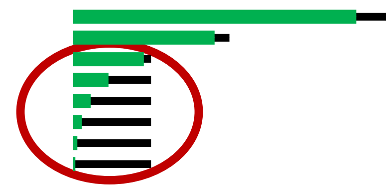
spatial extent

1 page

1 page

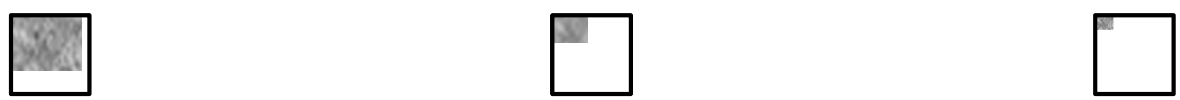
1 page

BRICKING MULTI-RESOLUTION PYRAMIDS (4)



Tail of pyramid

- Below size of single page; can cut off early



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

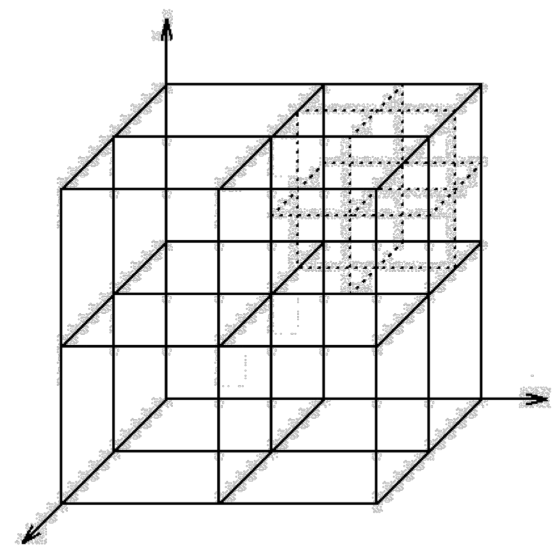
OCTREES FOR VOLUME RENDERING (1)

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.)



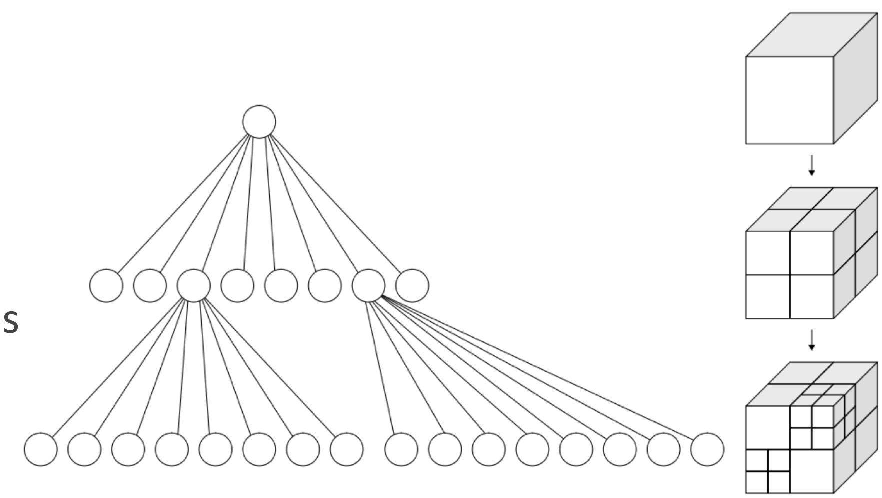
OCTREES FOR VOLUME RENDERING (2)

Representation

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

Data structure

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



wikipedia.org

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

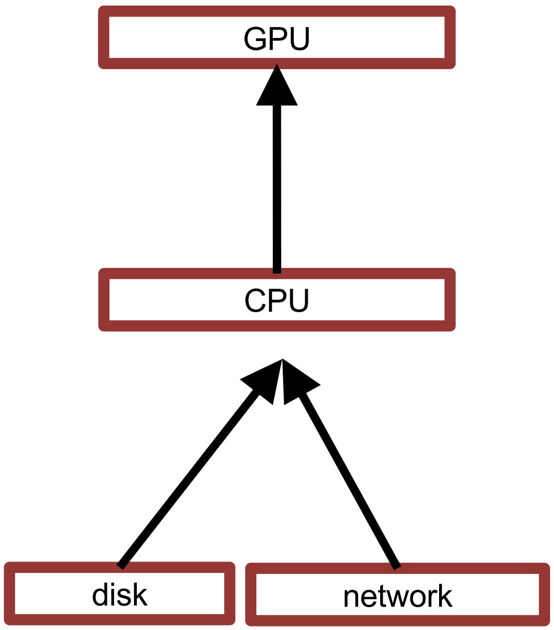
- Stream volume bricks from CPU to GPU on demand

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



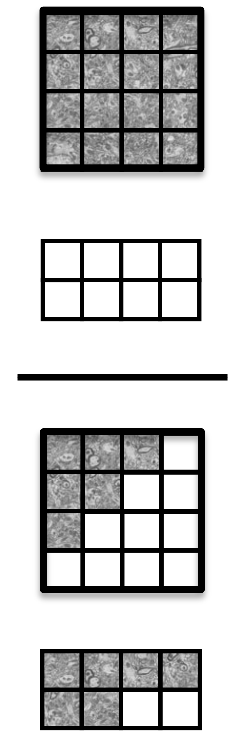
OUT-OF-CORE TECHNIQUES (2)

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

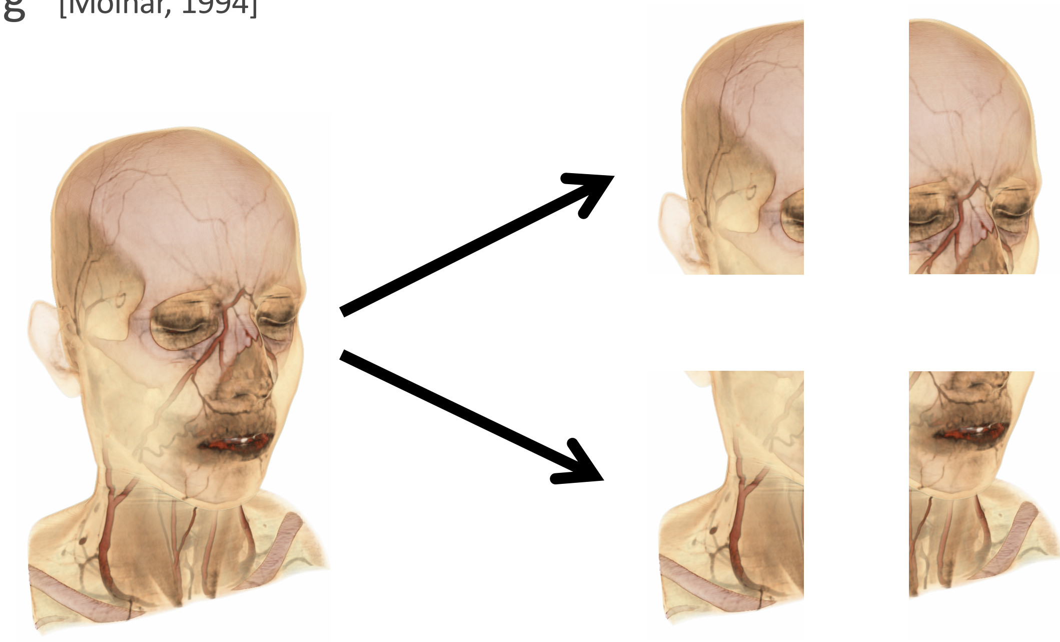
- Determine **working set**
- Page working set into cache memory
- Render from cache memory



DOMAIN DECOMPOSITION (1)

Subdivide image domain (image space)

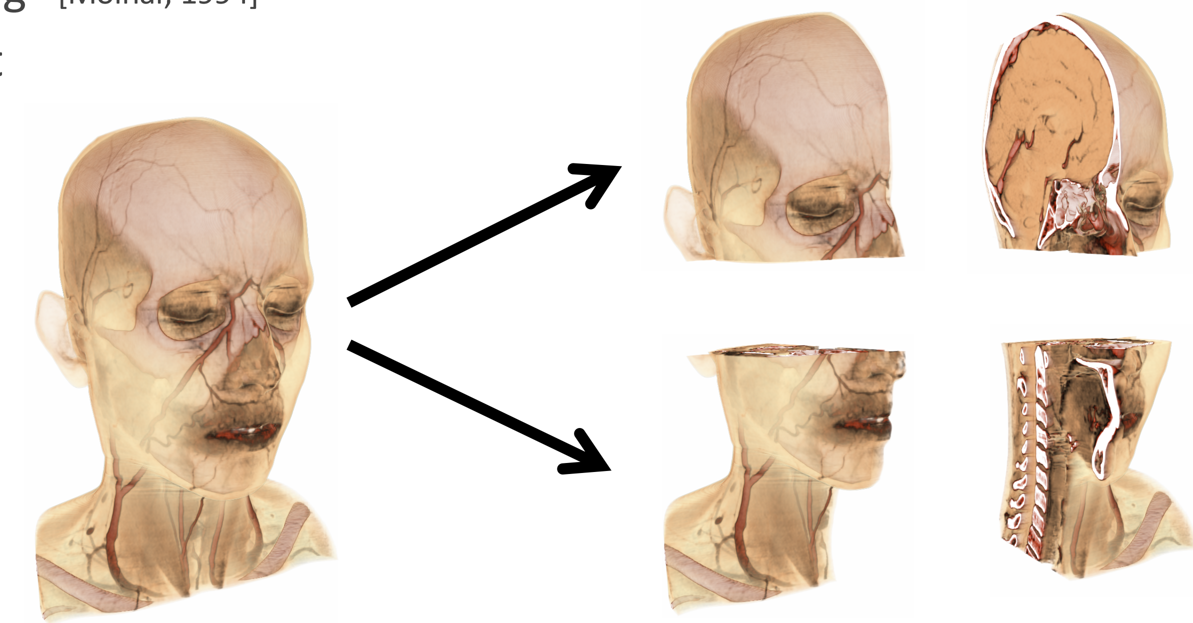
- “Sort-first rendering” [Molnar, 1994]
- View-dependent



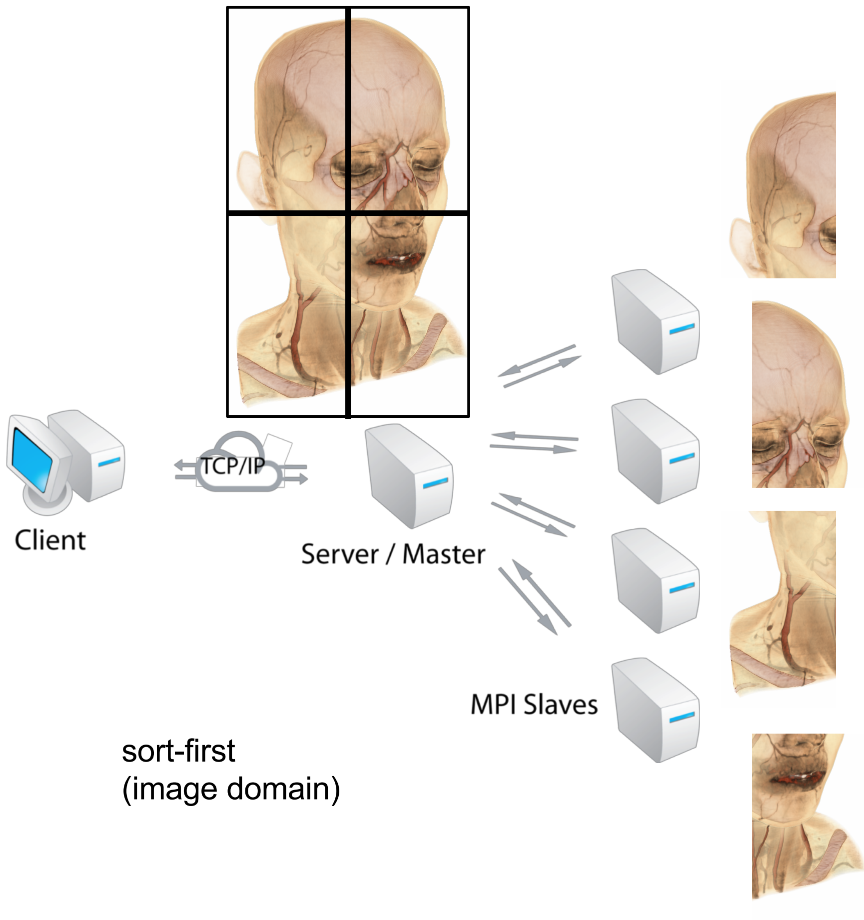
DOMAIN DECOMPOSITION (2)

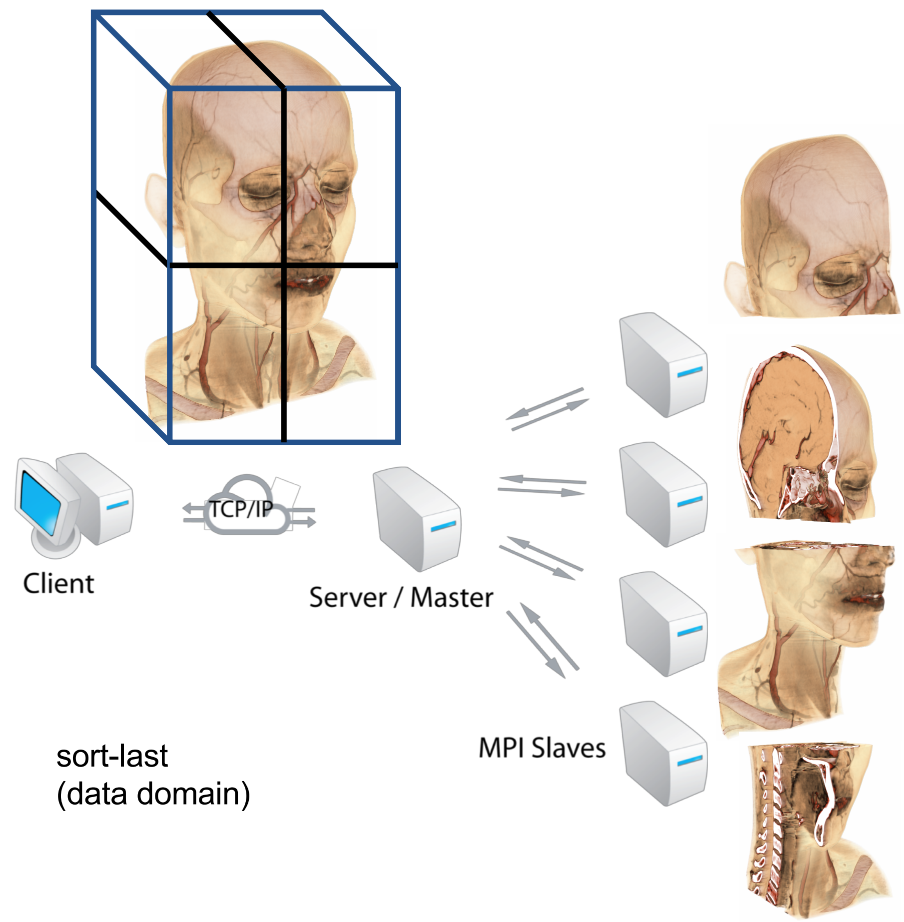
Subdivide data domain (object space)

- “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

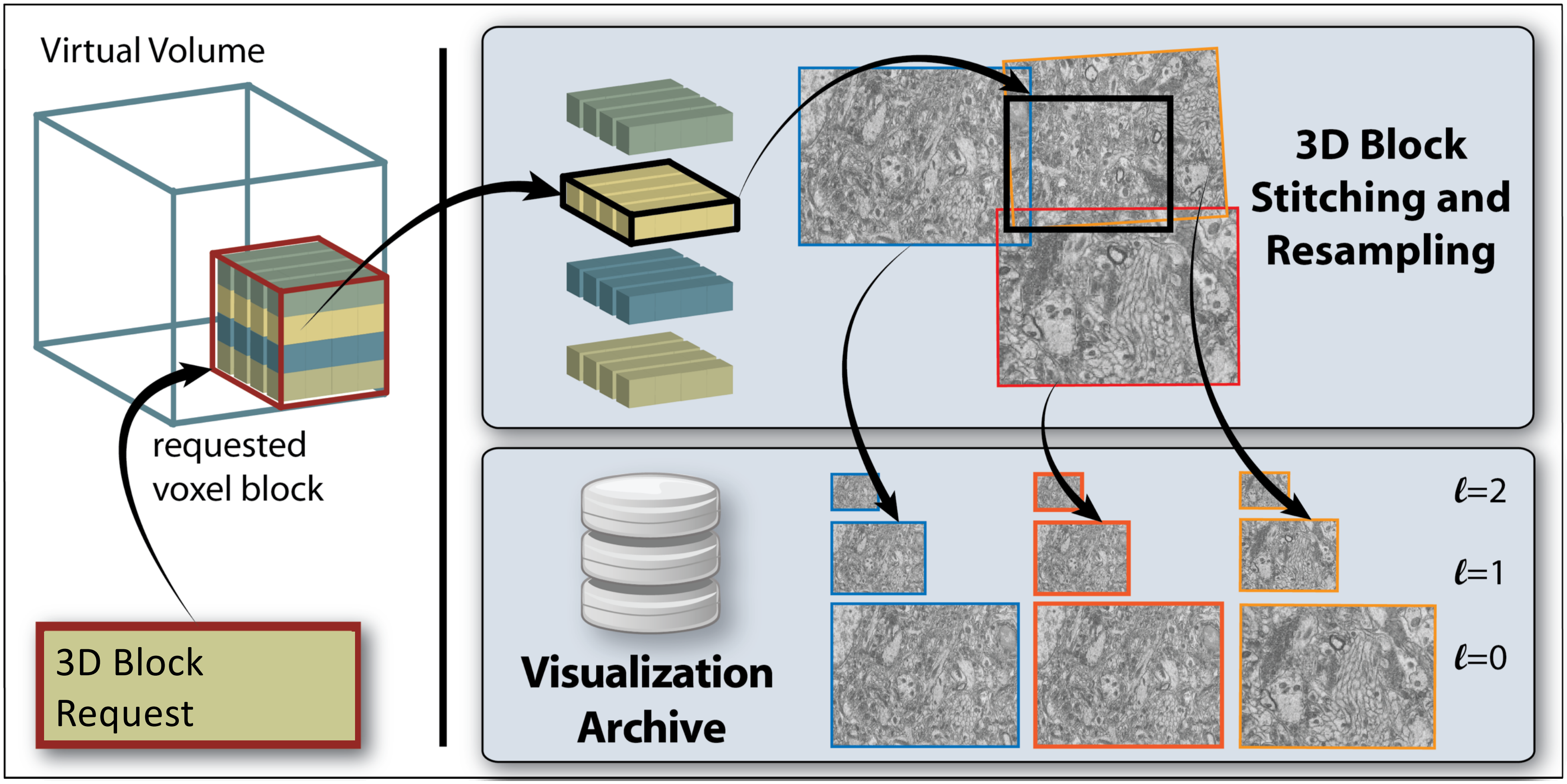
First determine what is visible / needed: working set

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

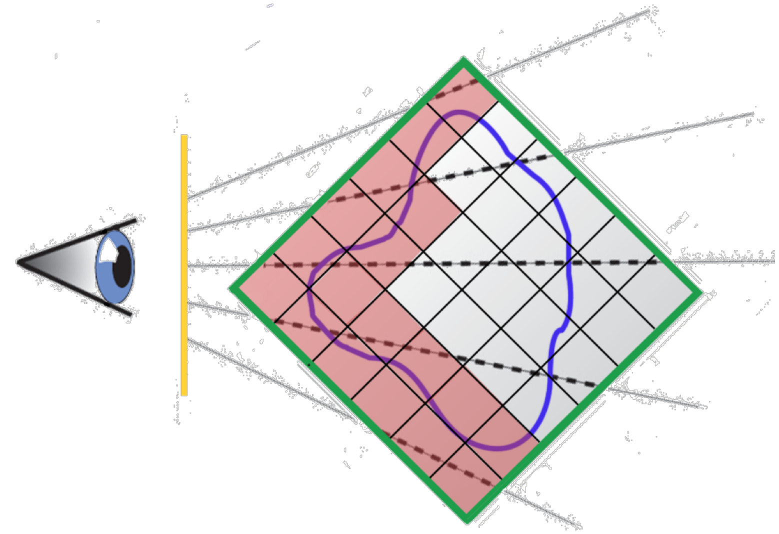
Caching scheme

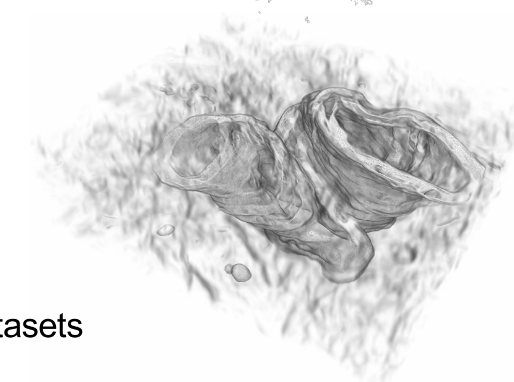
- Process only currently visible bricks
- Cache result for re-use

GPU Implementation

- CUDA and shared memory for fast computation

Different noise removal and filtering algorithms

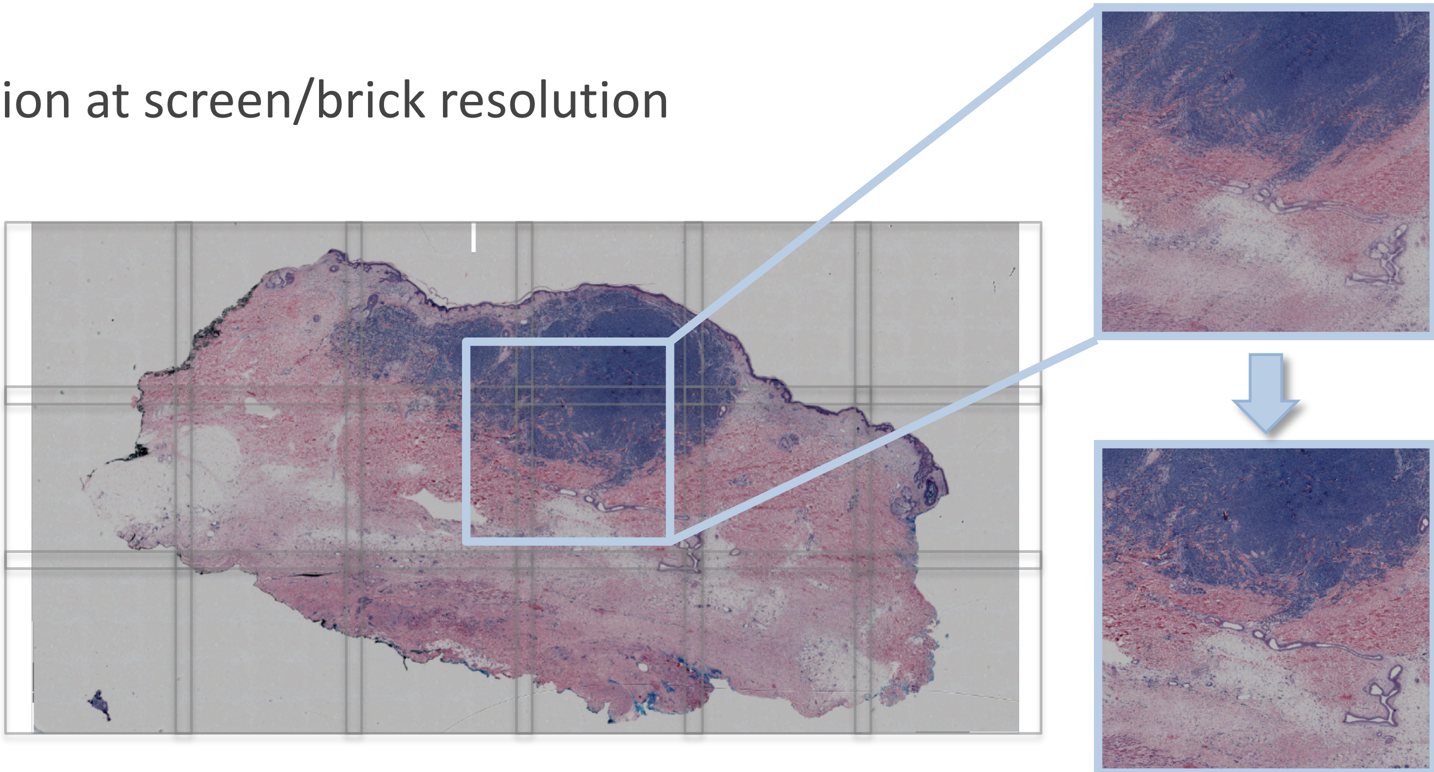




[Jeong et al., IEEE Vis 2009]
Scalable and Interactive Segmentation and Visualization of Neural Processes in EM Datasets

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?

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