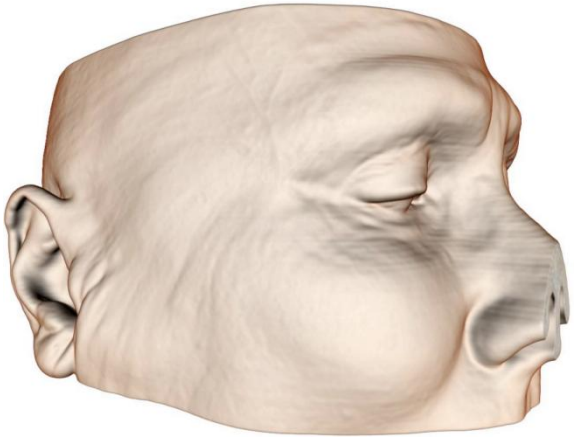
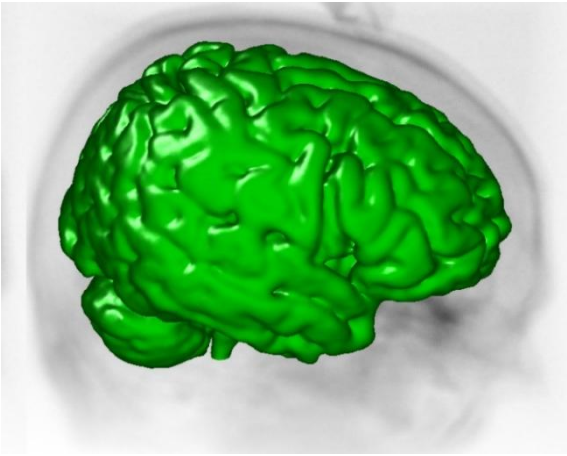
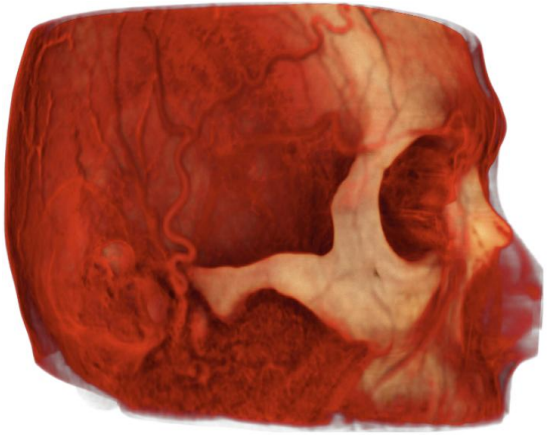
Advanced Medical Volume Rendering and Segmentation on the GPU

- Mike Roberts, Eric Penner, Jeff Packer, Mario Costa Sousa, Joseph Ross Mitchell
- Imaging Informatics Lab, University of Calgary





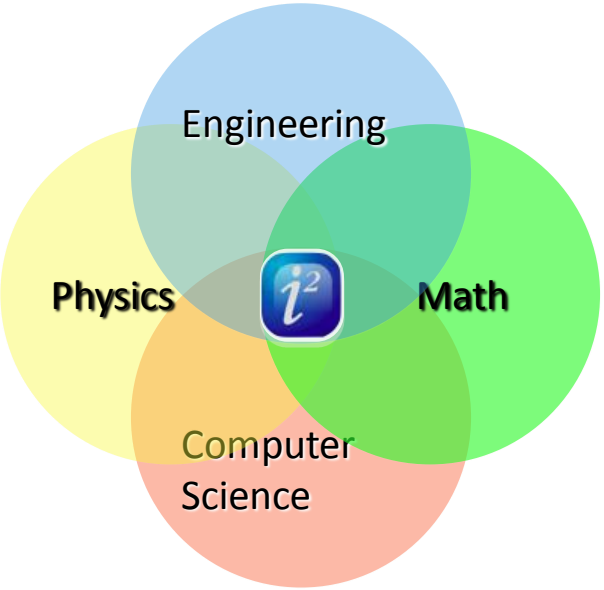


imaging informatics .ca

UNIVERSITY OF CALGARY

Medical Imaging Informatics

- *Information technologies that improve the diagnosis, treatment and monitoring of disease.*



imaging informatics .ca

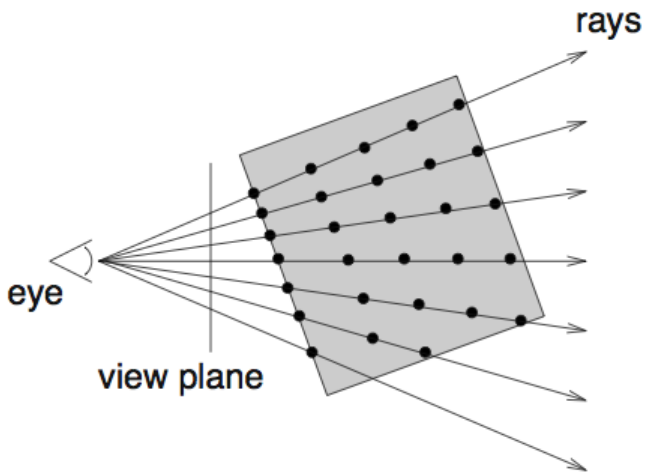
UNIVERSITY OF CALGARY

Overview

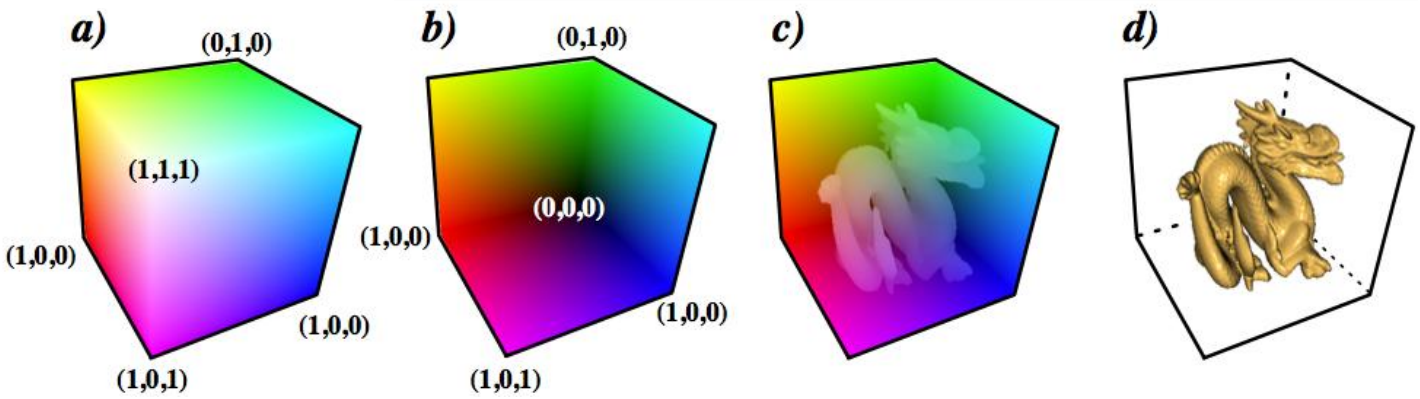
- Volume Rendering
  - Hierarchical Ray Casting Acceleration
  - Real-Time Volumetric Ambient Occlusion
- Volume Segmentation
  - Efficient Level Set Segmentation
- Conclusions / Questions

Accelerated GPU Volume Rendering

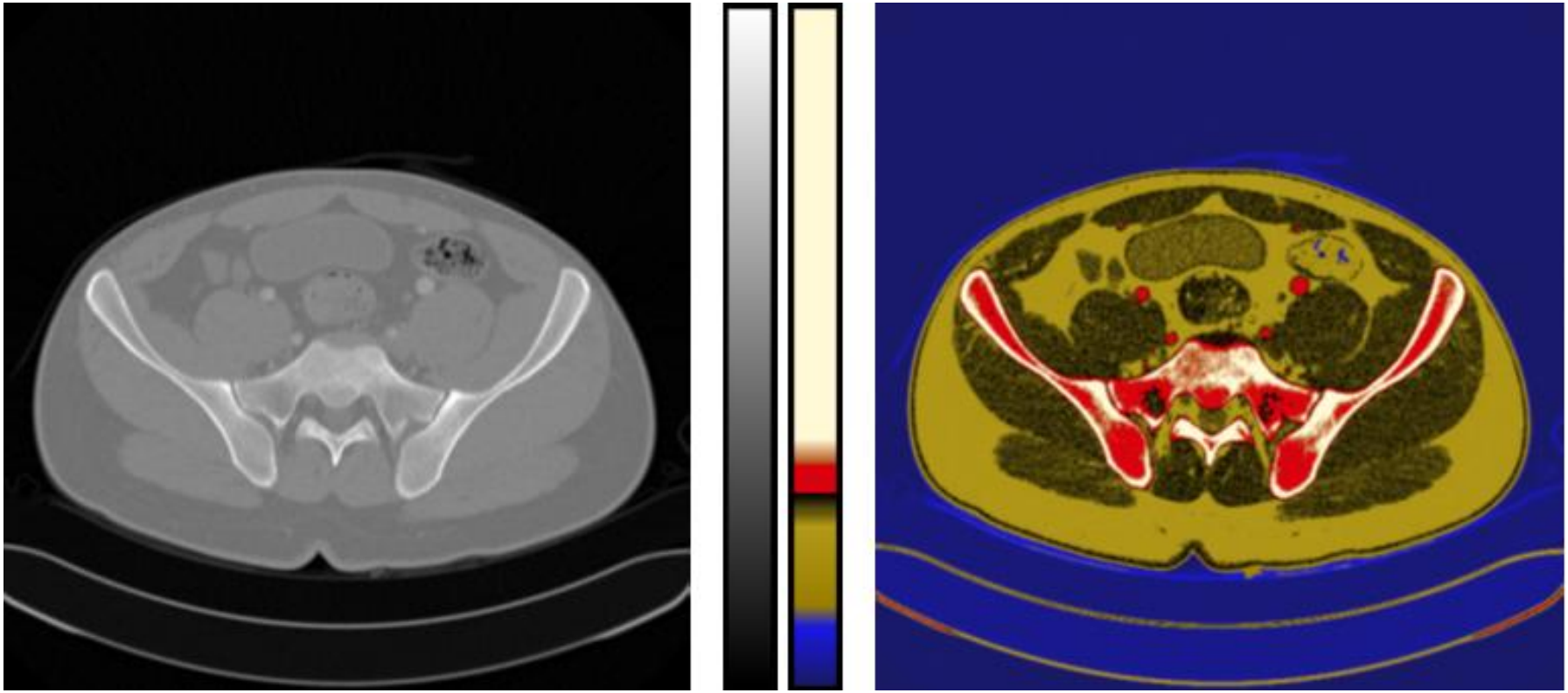
Using Overlapped Min Max Octrees


rays
eye
view plane

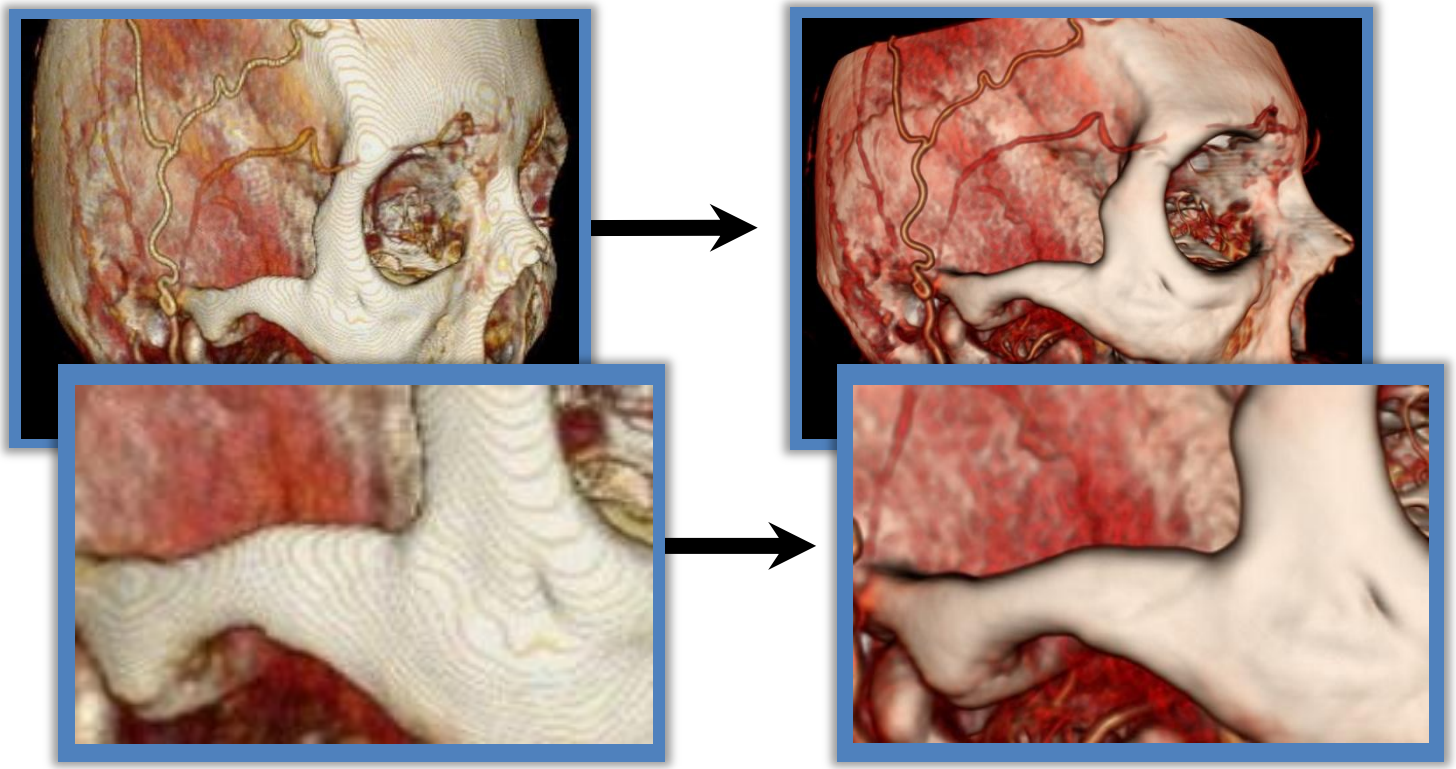
Ray Casting


a)
(0,1,0)
(1,1,1)
(1,0,0)
(1,0,0)
(1,0,1)
b)
(0,1,0)
(0,0,0)
(1,0,0)
(1,0,0)
(1,0,1)
c)
d)

Classification (Transfer Functions)



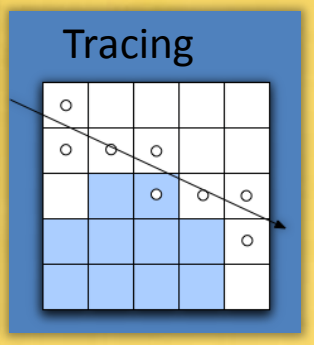
Motivation for High Image Quality

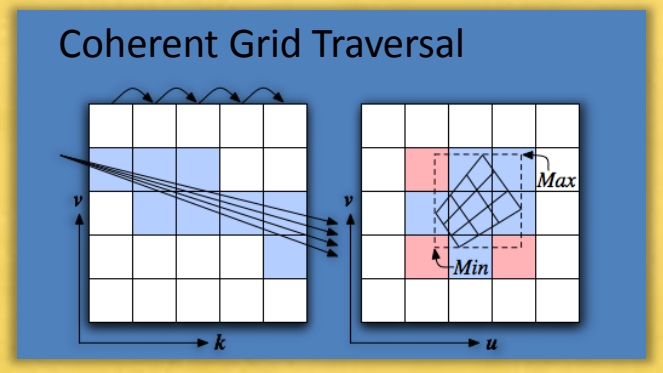


Goal:
High Quality Interactive Visualization

Approach:
Optimized acceleration structure with more time spent on sampling

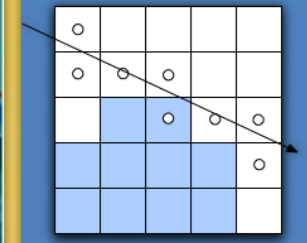
CPU Acceleration



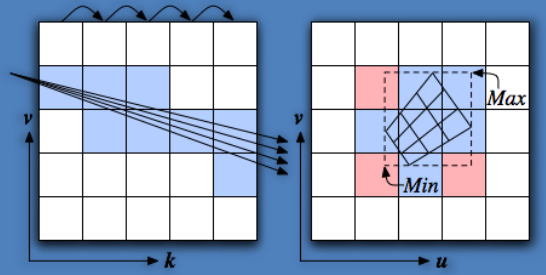


CPU Acceleration

Tracing

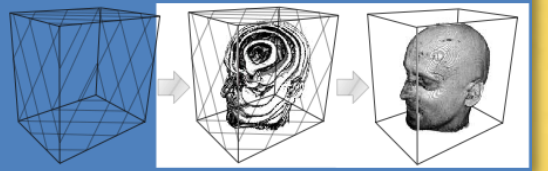


Coherent Grid Traversal

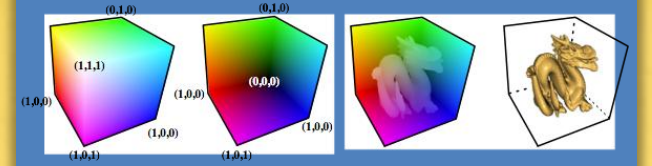


GPU Acceleration

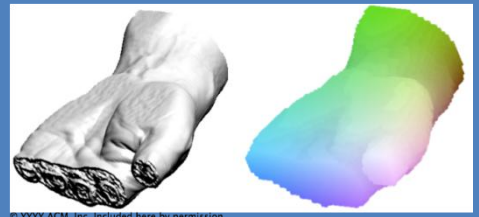
Slicing



Ray Casting

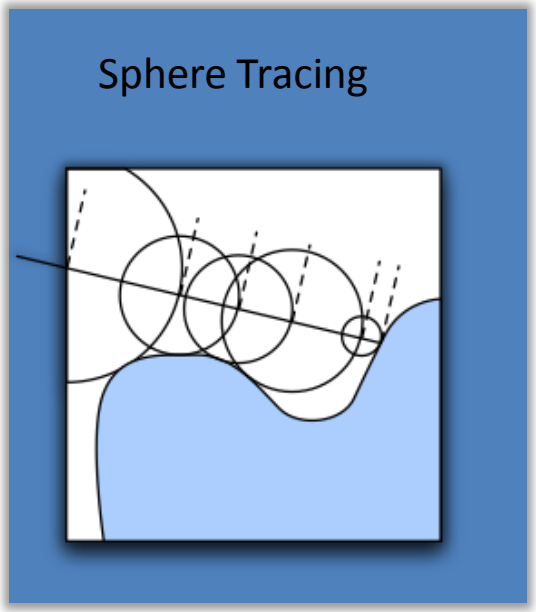


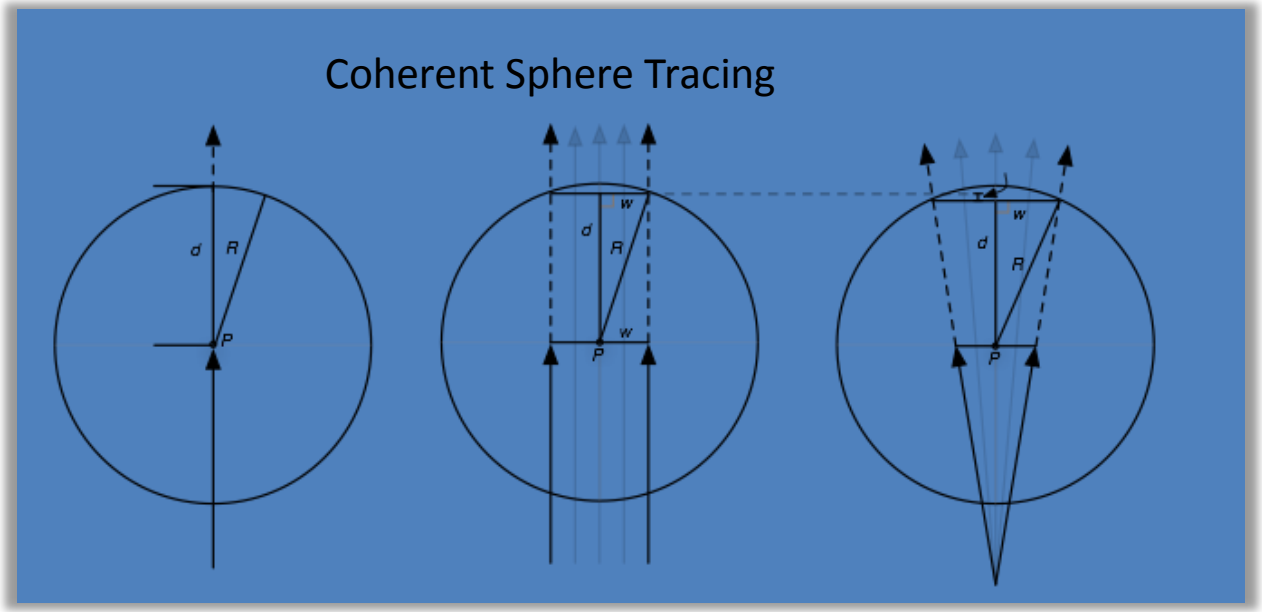
Proxy Geometry



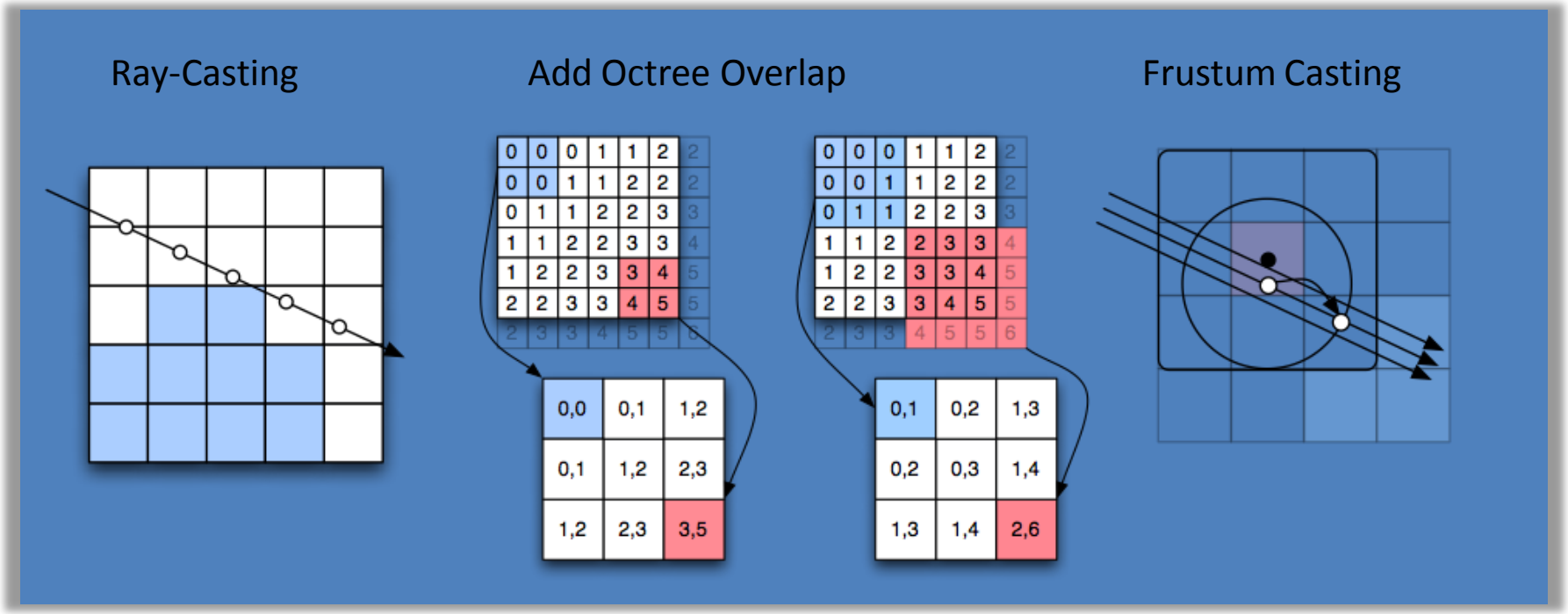
© YYYY ACM, Inc. Included here by permission.

Sphere Tracing

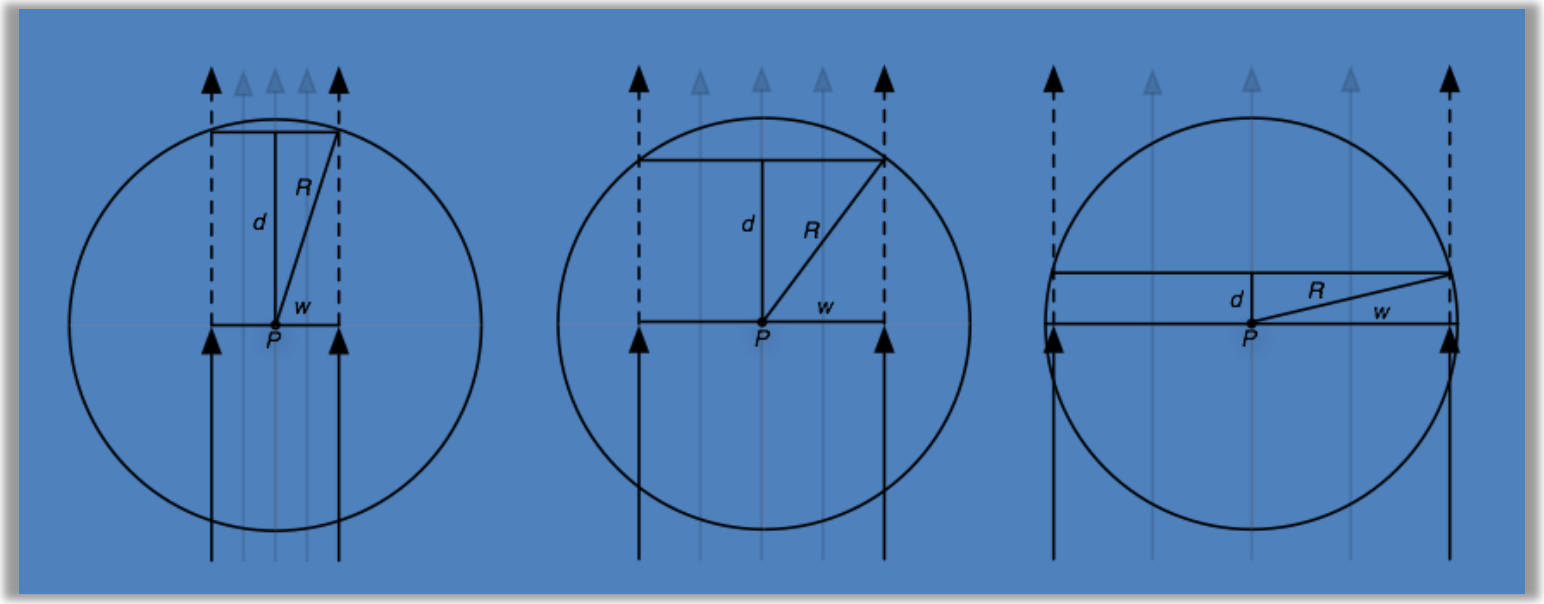




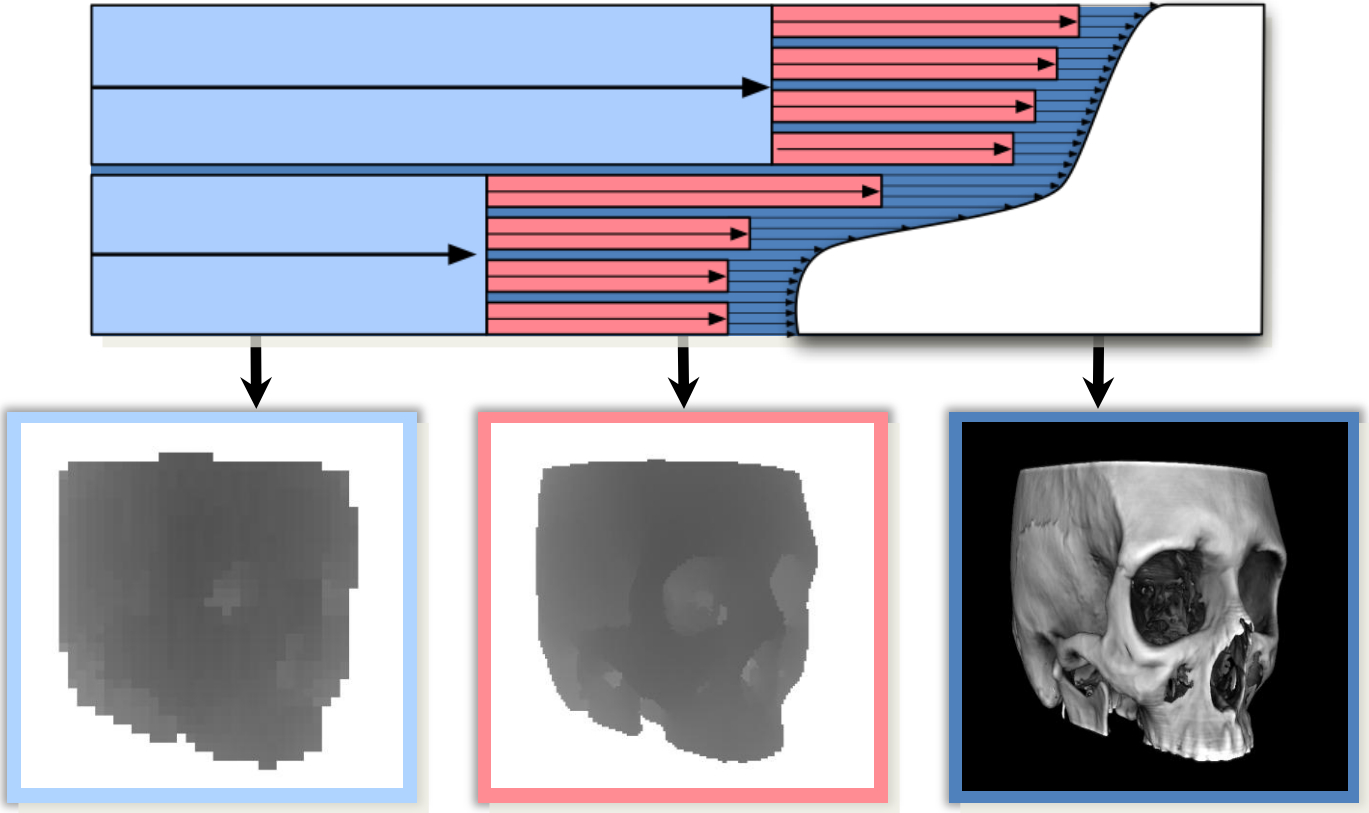
Overlapped Min-Max Octree



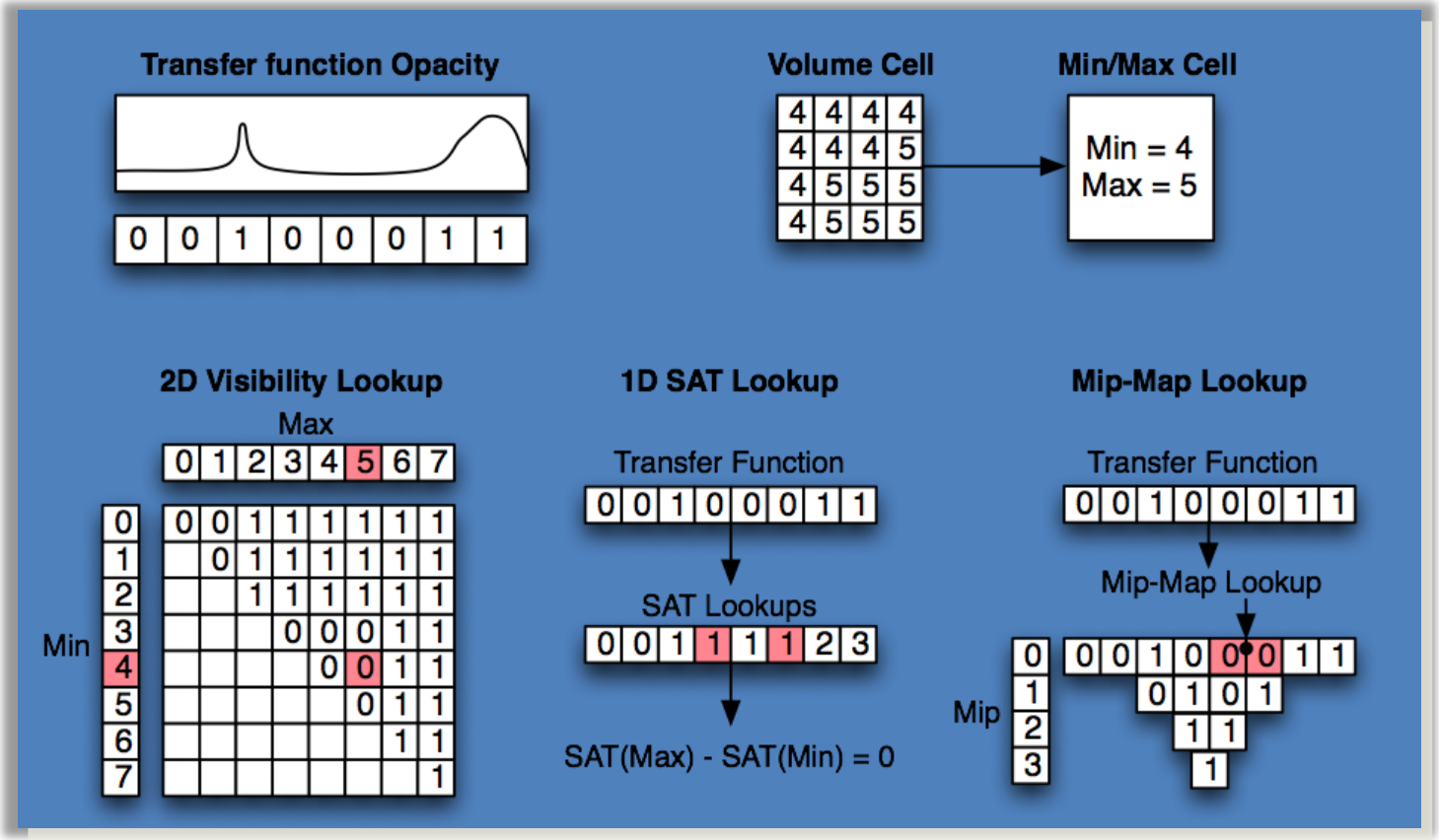
Overlapped Min-Max Octree



Frustum Casting



Min-Max Visibility Tests

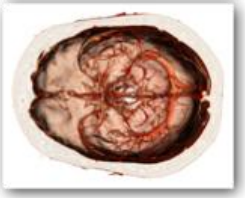


Iso Performance (ms)

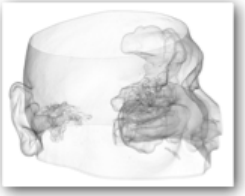
Skin

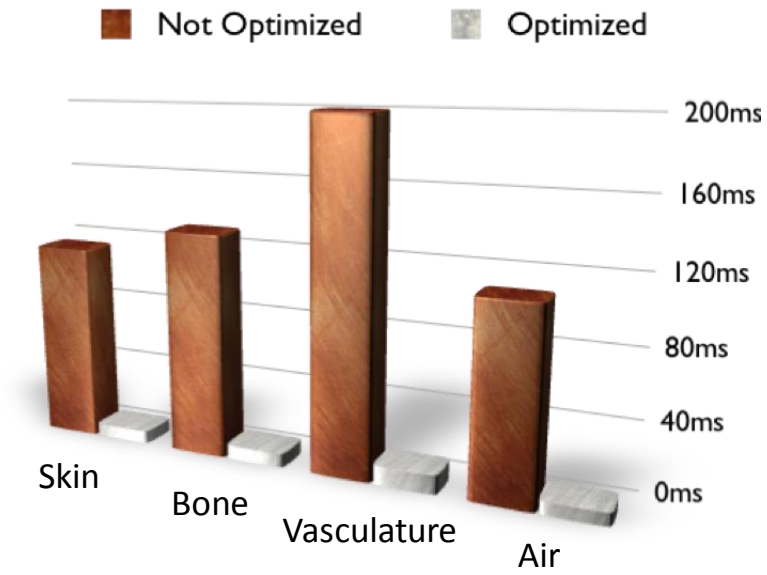
Bone

Vasculature



Air



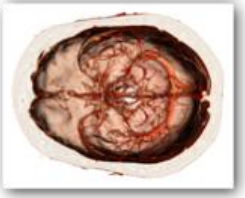


DVR Performance (ms)

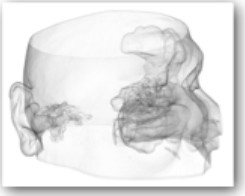
Skin

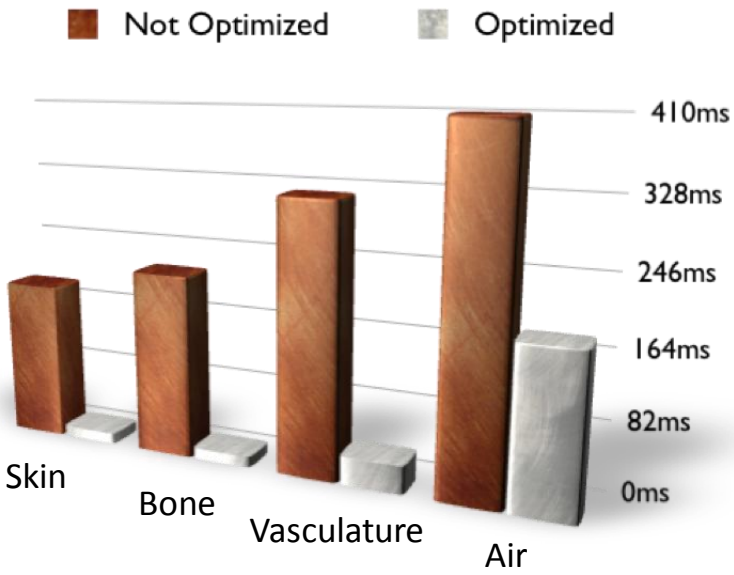
Bone

Vasculature



Air



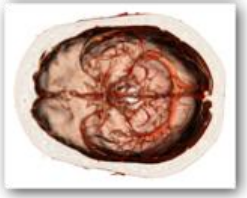


Performance Increase

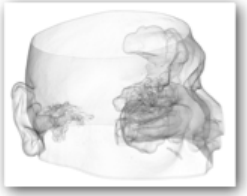
Skin

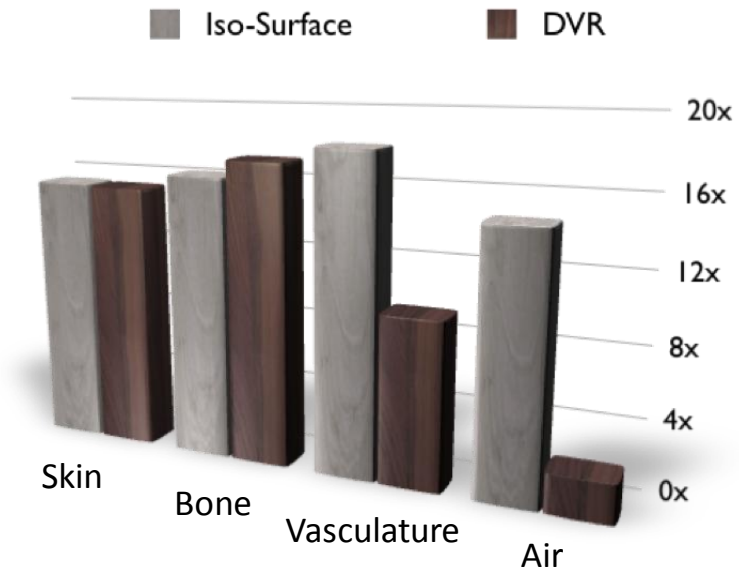
Bone

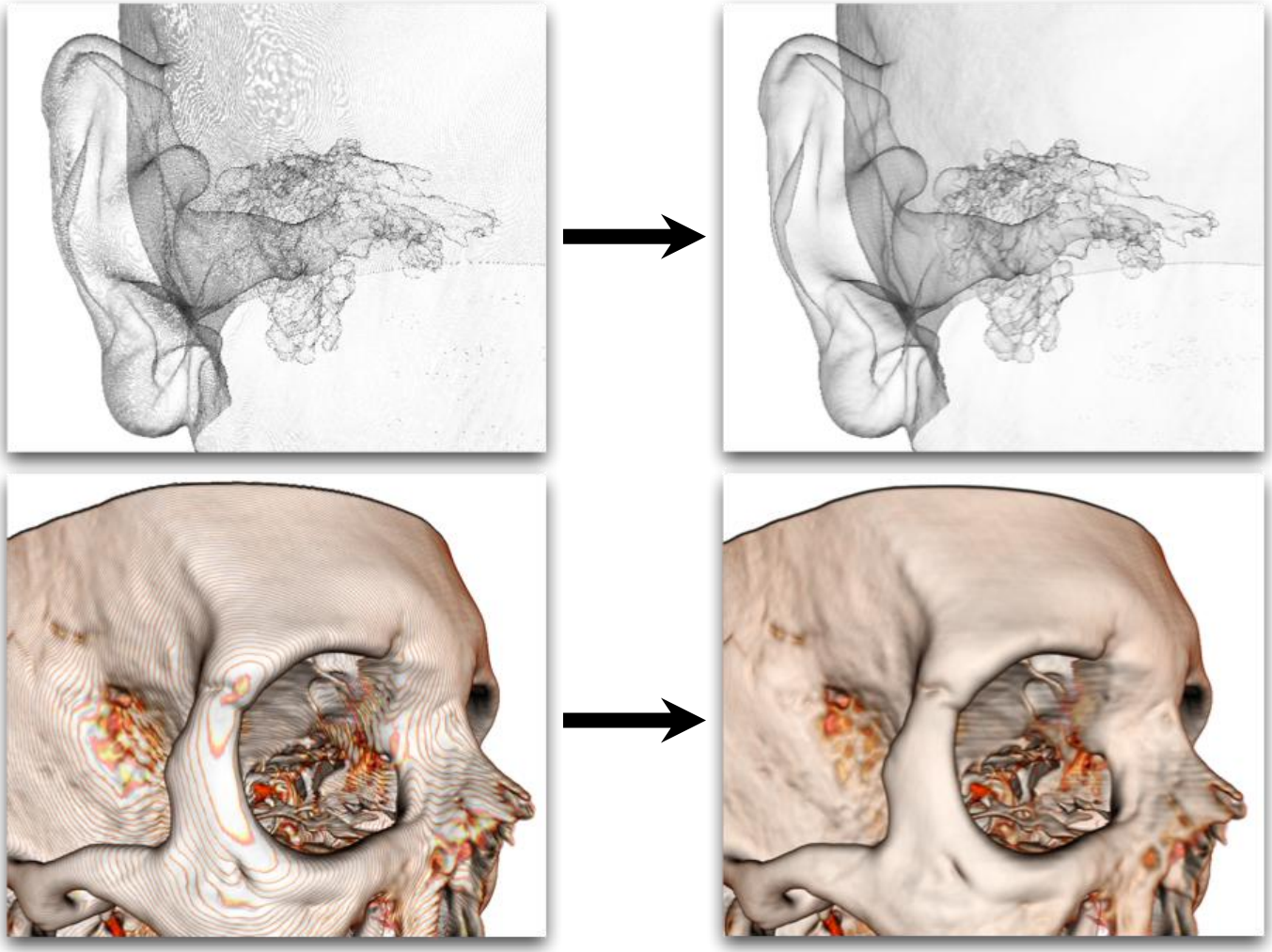
Vasculature

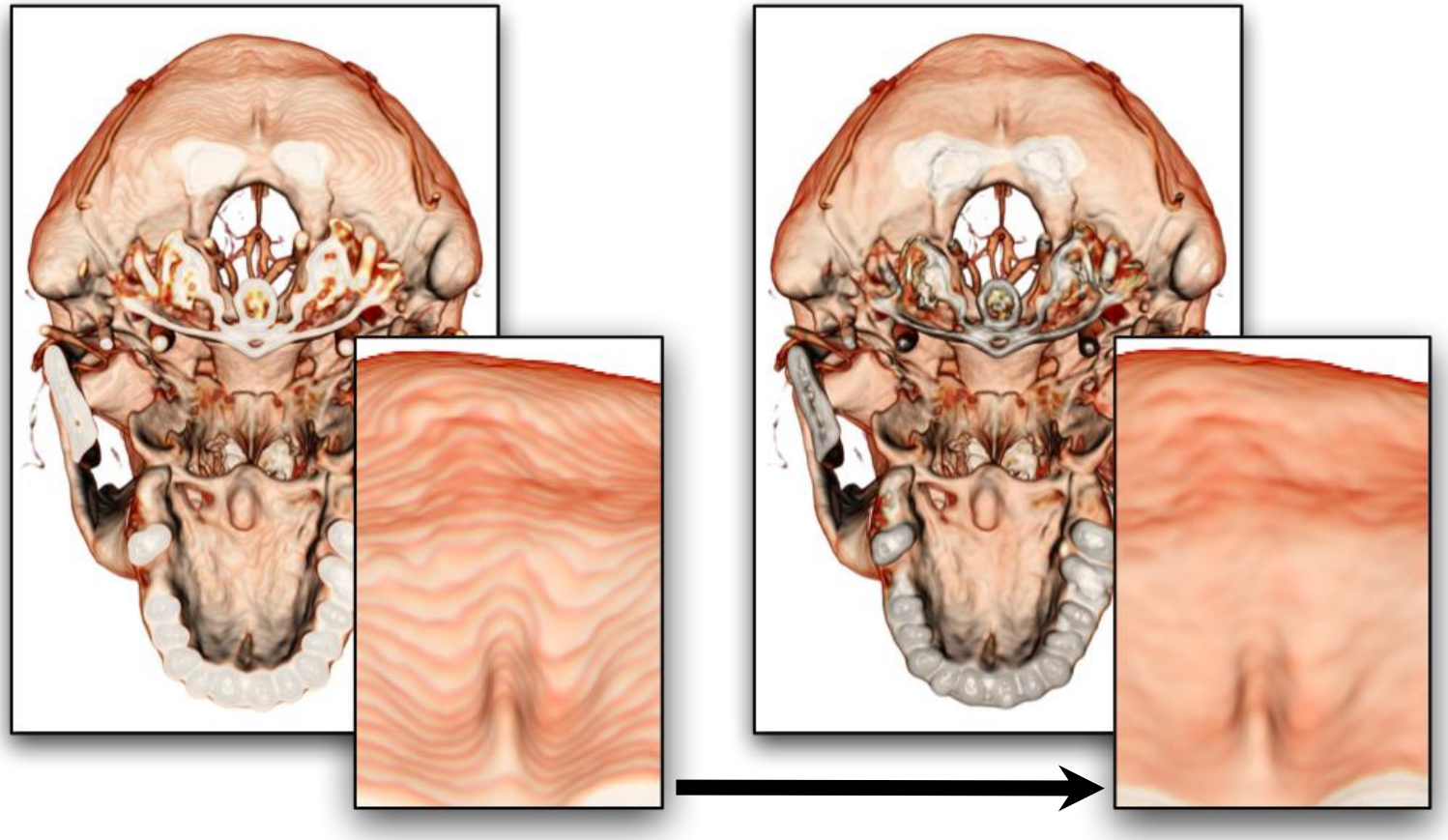


Air







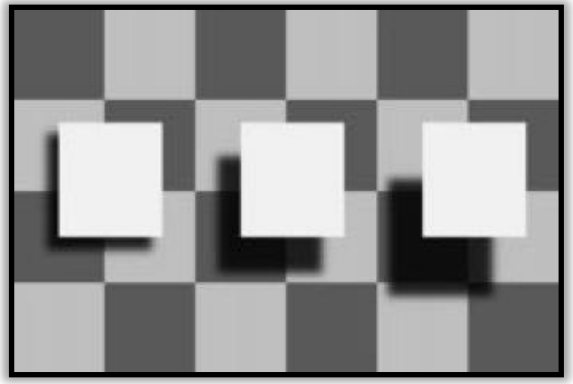


Ambient Occlusion and Soft Shadows

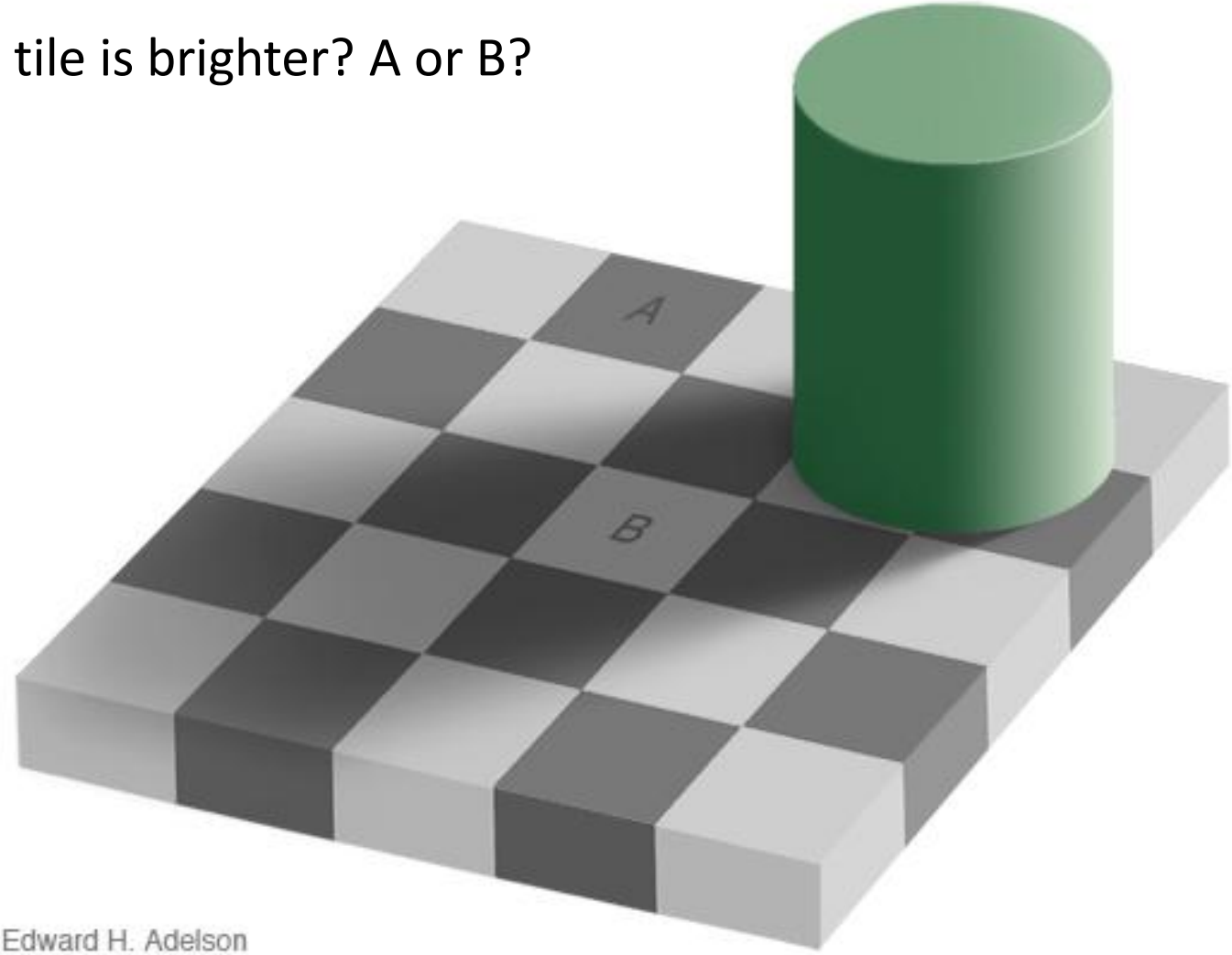
Using Filterable Occlusion Maps

Motivation for Shadows / Lighting

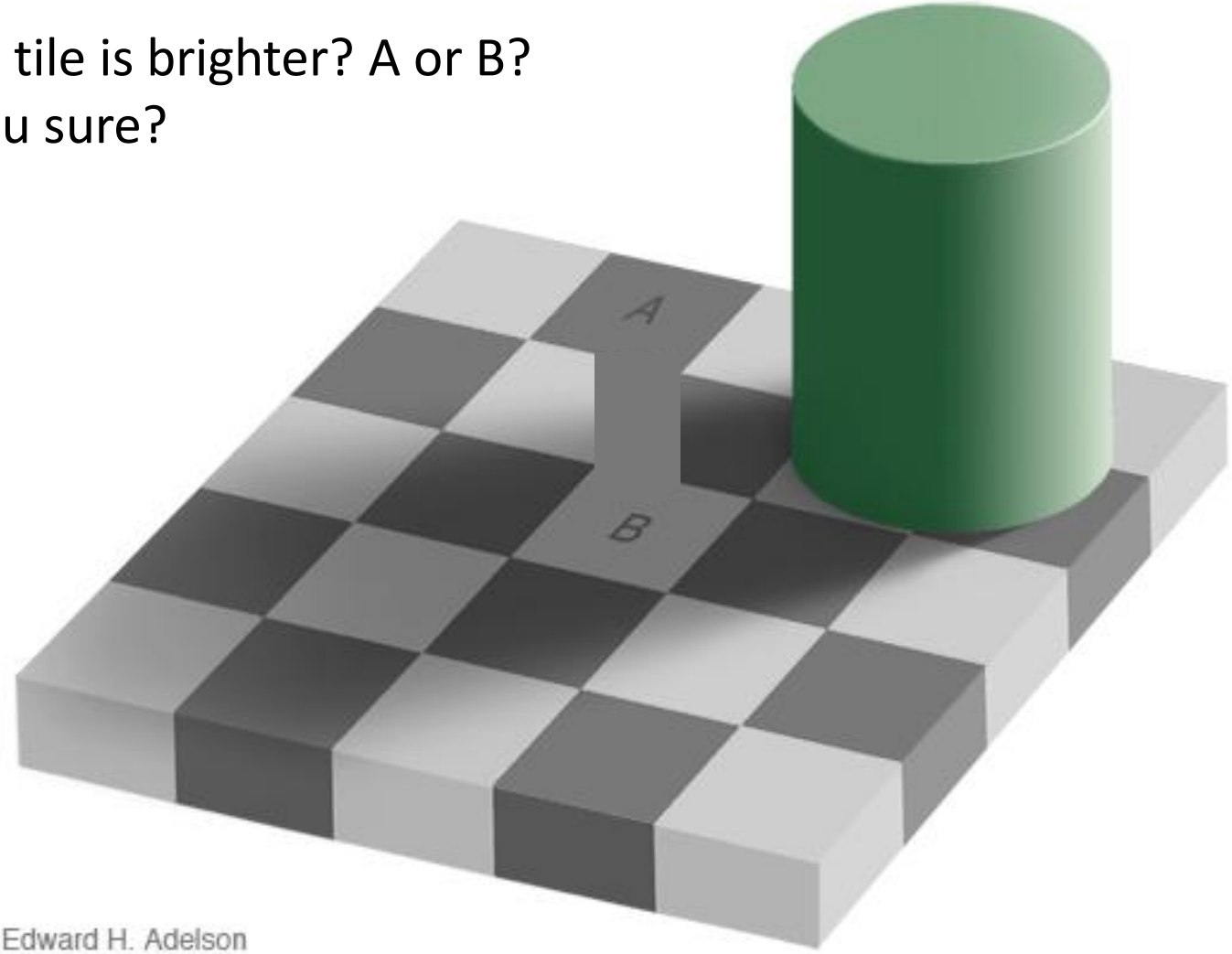
- Plays key role in perception of 3D
- Provides information about the scene
  - Position, Size and Geometry
  - Both Occluder and Receiver
- The more realistic the better



Which tile is brighter? A or B?

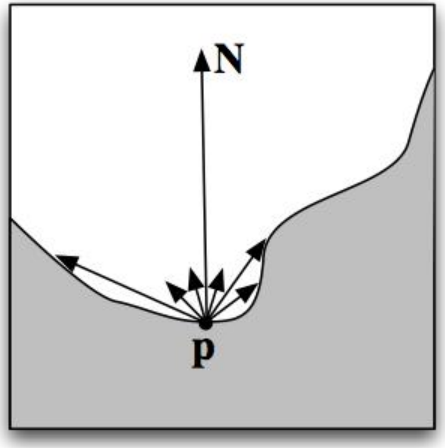


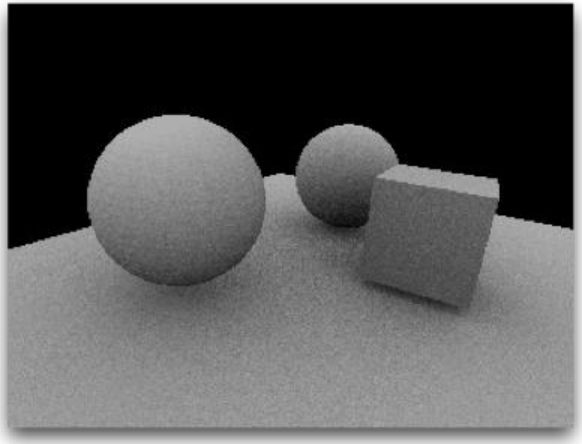
Which tile is brighter? A or B?
Are you sure?



Ambient Occlusion

- Traditional Approach - Monte-Carlo
  - Very Time Consuming
  - Under-sampling Artifacts





$$\int_{\Omega} N \cdot L(\omega) d\omega$$

Monte-Carlo

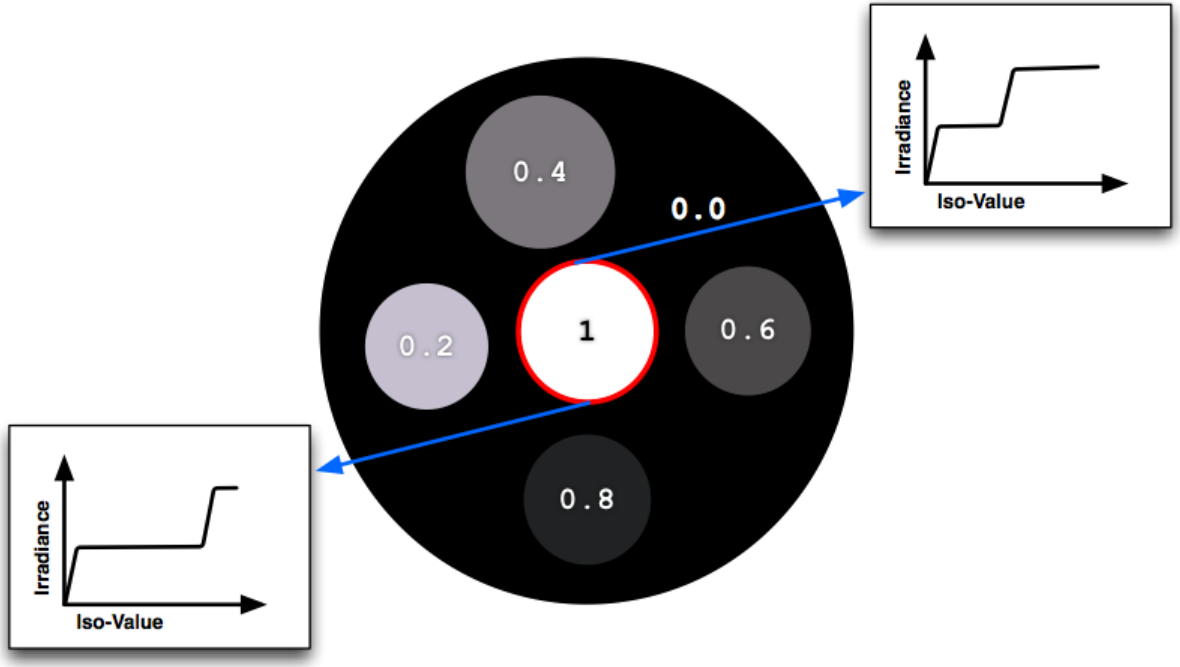
$$\sum_{i=0}^{k} N \cdot L(\omega_i) \Delta\omega$$

Pre-computed Lighting?

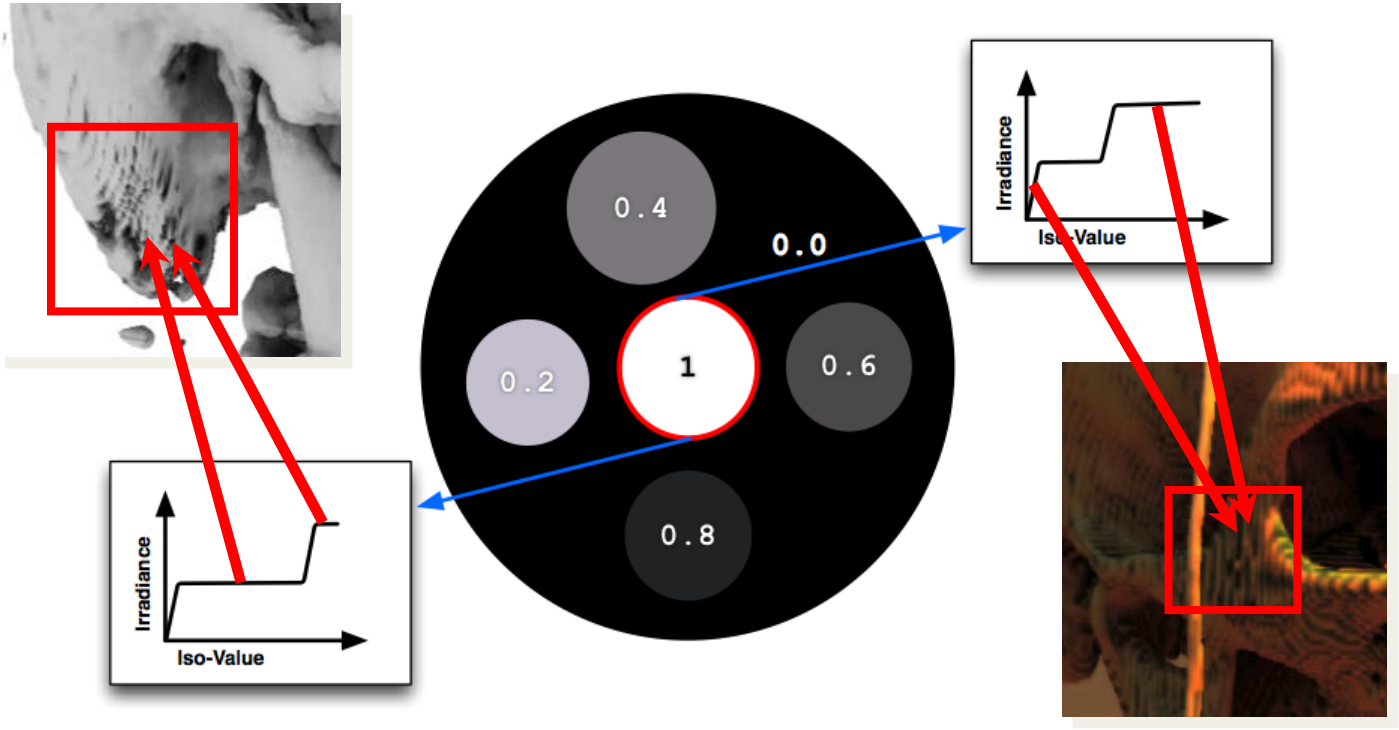
Stewart et al. 2003

Wyman et al. 2006

Pre-computed Lighting?

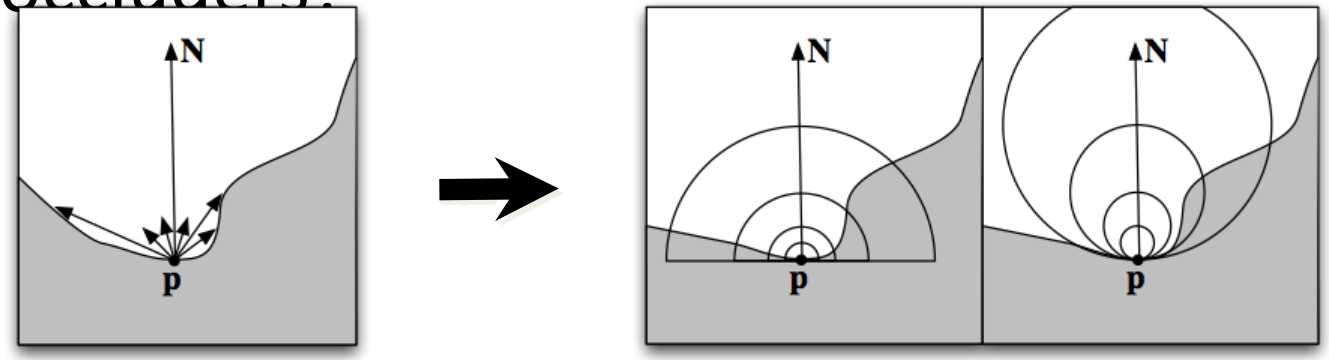


Pre-computed Lighting?



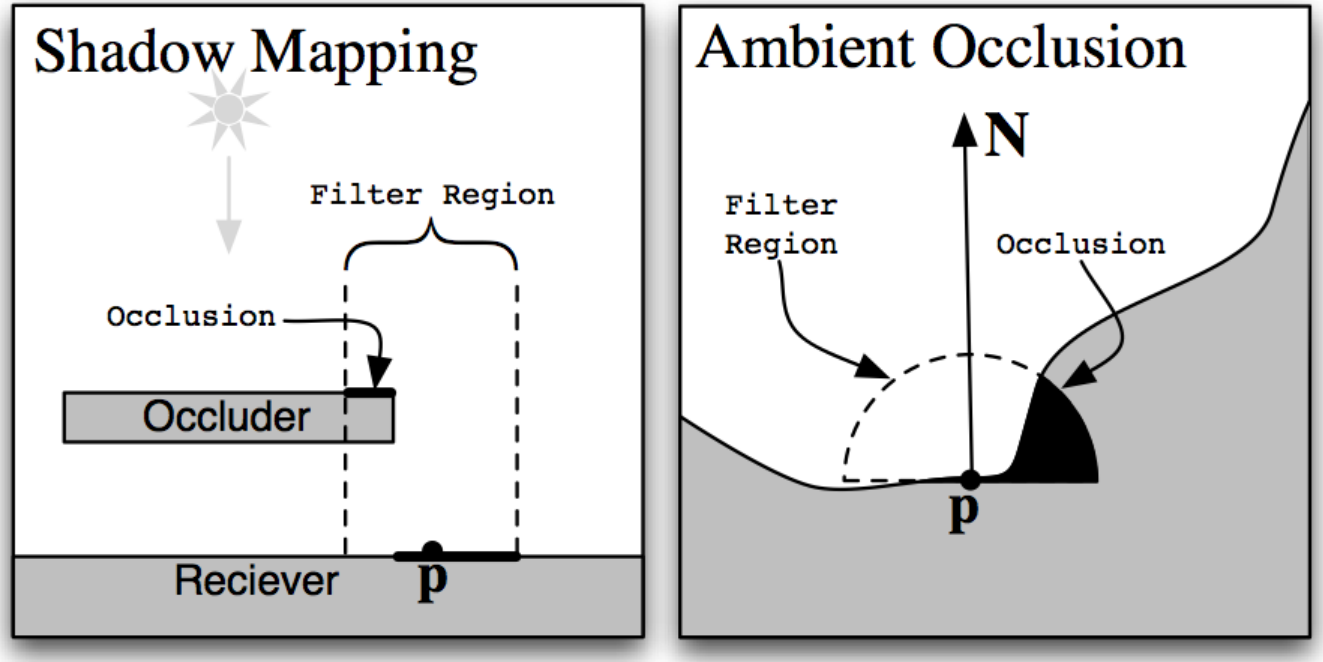
Our Approach: Dynamic AO

- Monte-Carlo ray-tracing much too slow for real-time
- Approximation: How many neighboring occluders?



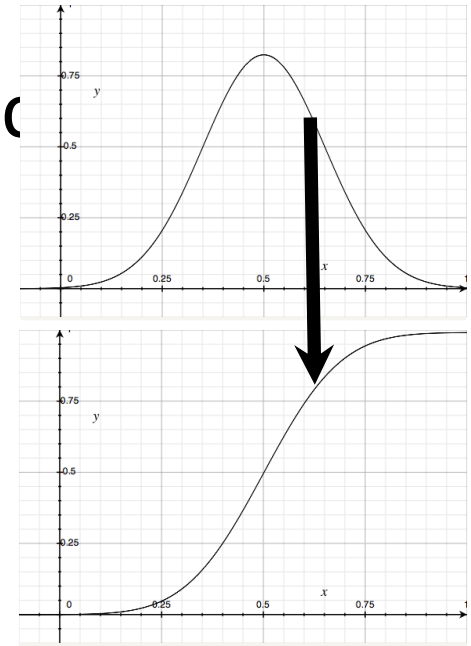
$$\sum_{i=0}^{k} N \cdot L(\omega_i)\Delta\omega \quad \rightarrow \quad \sum_{i=0}^{k} L'(v_i)\Delta v \qquad L'(x) = \begin{cases} 1 & \text{if } x \leq c\ , \\ 0 & \text{otherwise.} \end{cases}$$

Dynamic AO


Shadow Mapping
Filter Region
Occlusion
Occluder
Reciever
p
Ambient Occlusion
N
Filter
Region
Occlusion
p

Dynamic AO

- Can we query large region at onc
- Lookup into CDF



$$CDF(x) = P(X \leq x) = \sum_{x_i \leq x} P(X = x_i) = \sum_{x_i \leq x} p(x)$$

Variance Shadow Maps

$$M_1 = E(x) \quad = \int_{-\infty}^{\infty} x p(x) dx$$

$$M_2 = E(x^2) = \int_{-\infty}^{\infty} x^2 p(x) dx$$

$$P(x \leq t) \leq p_{max}(t) \equiv \begin{cases} 1 & \text{if } t \leq \mu \text{ ,} \\ \frac{\sigma^2}{\sigma^2 + (t-\mu)^2} & \text{otherwise} \end{cases}$$

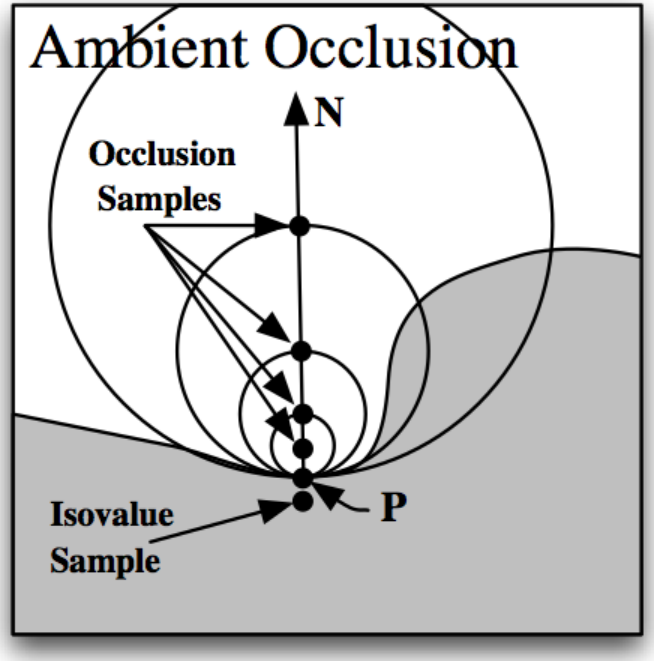
$$\mu = E(x) = M_1$$

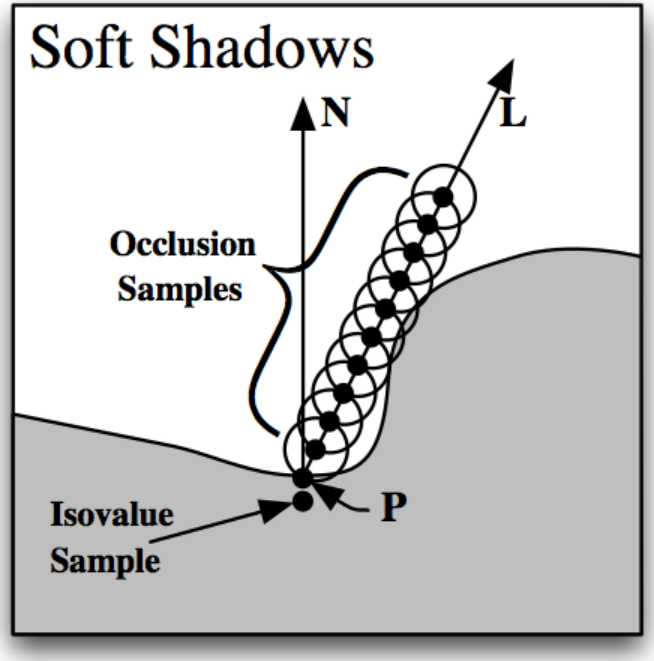
$$\sigma^2 = E(x^2) - E(x)^2 = M_2 - M_1$$

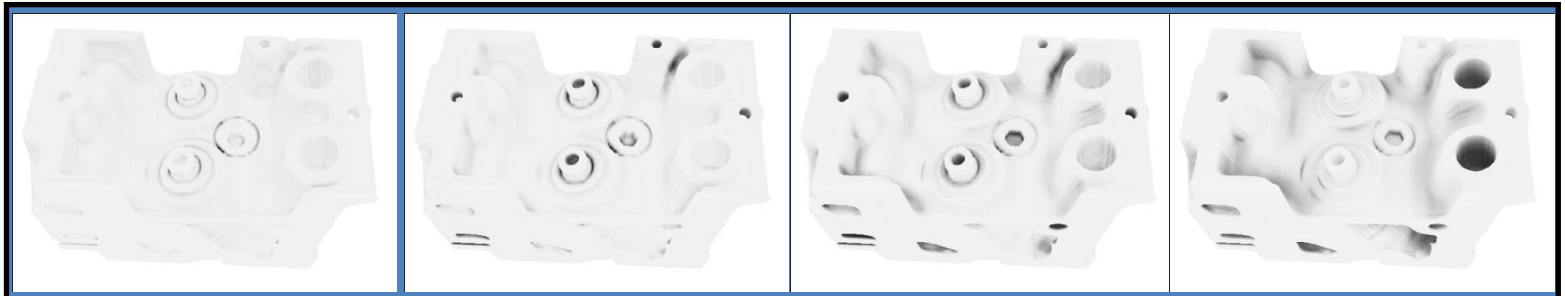
Filterable Occlusion Maps

- Store mean/variance of data
- Blur to distributions
- Generate and blur mip-map for larger distributions

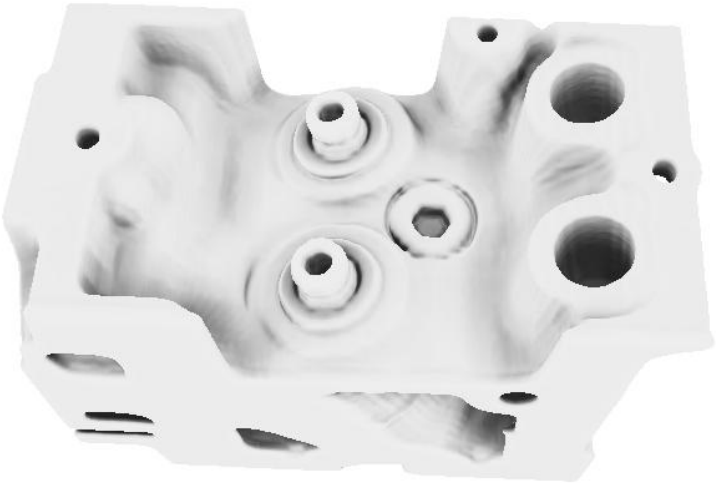
AO and SS Algorithms



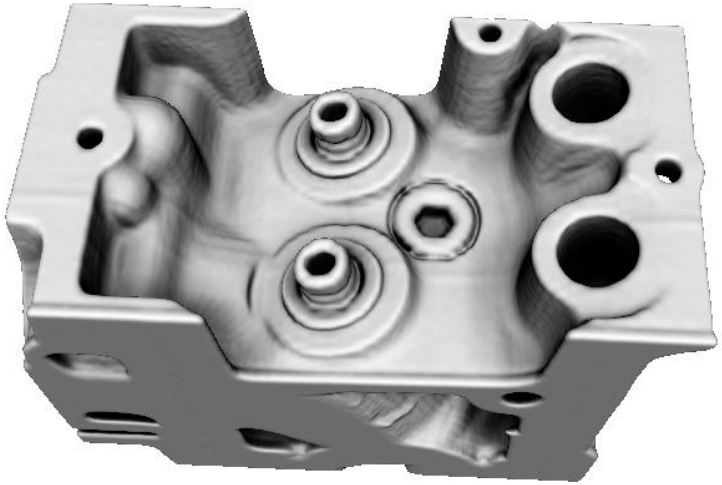




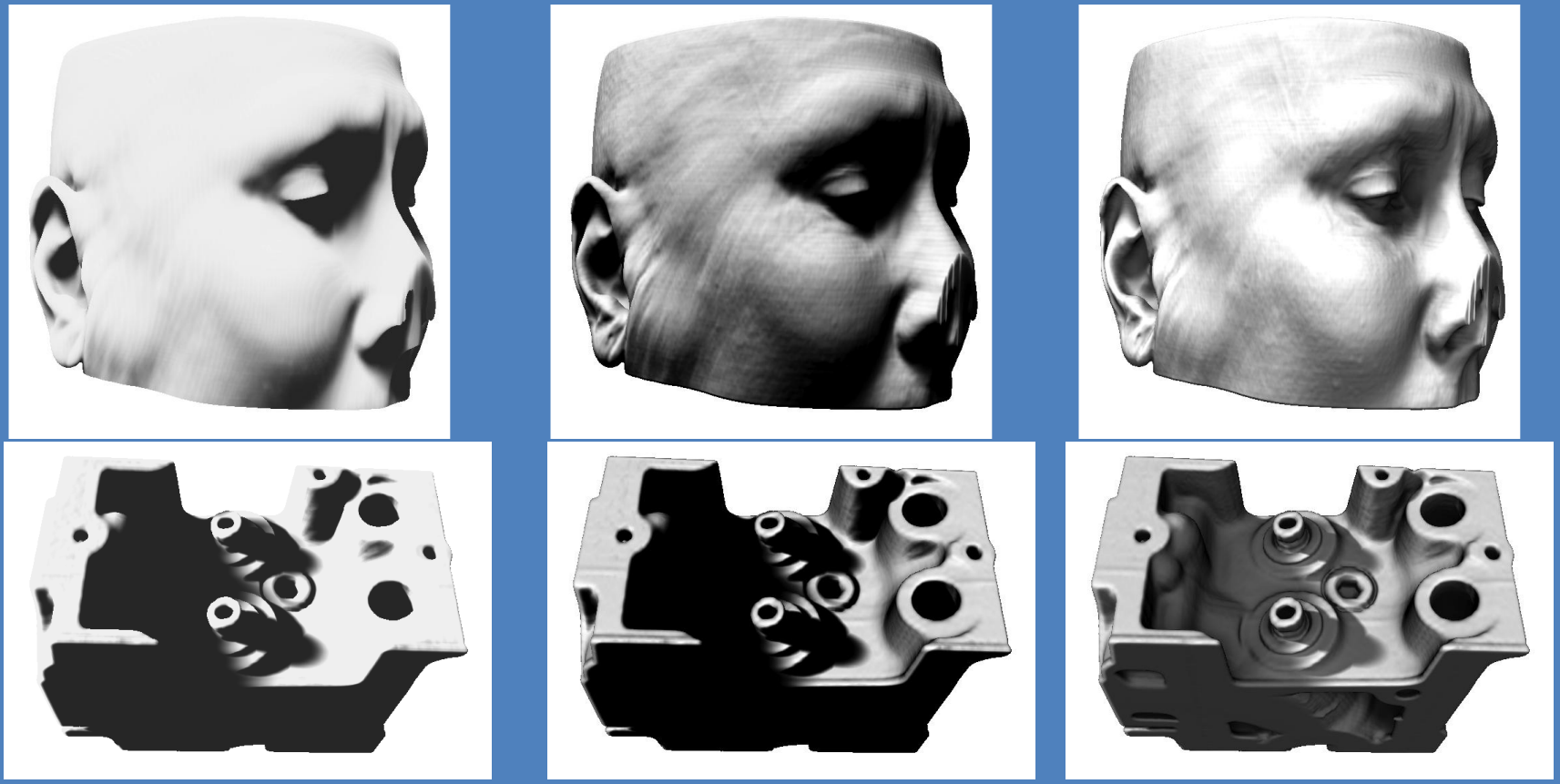
Ambient Samples



Ambient Occlusion



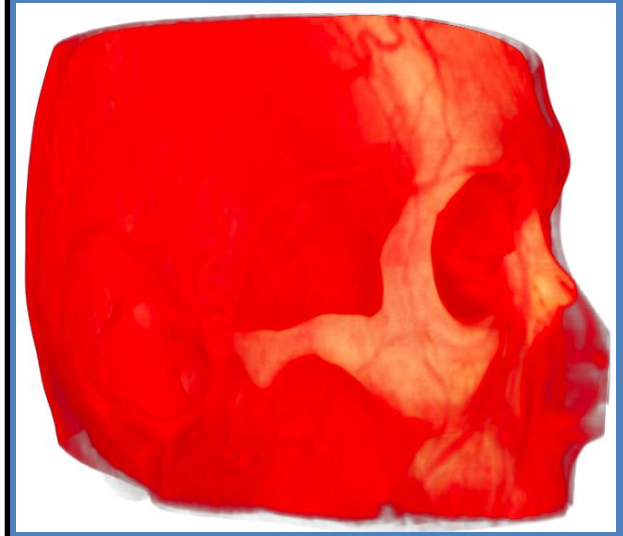
Phong + Ambient Occlusion



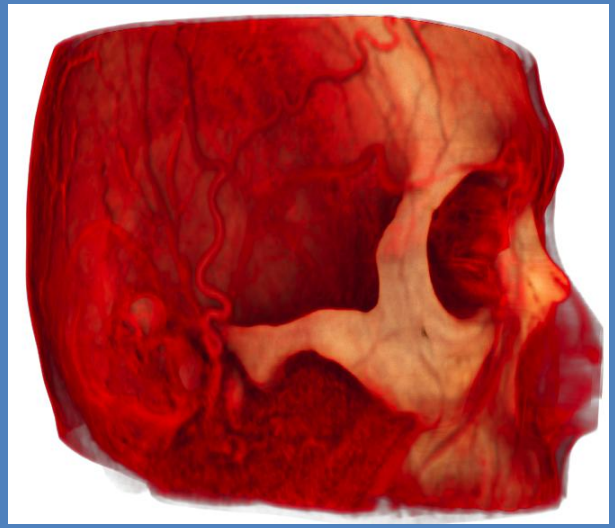
Soft Shadows

Soft Shadows + Phong

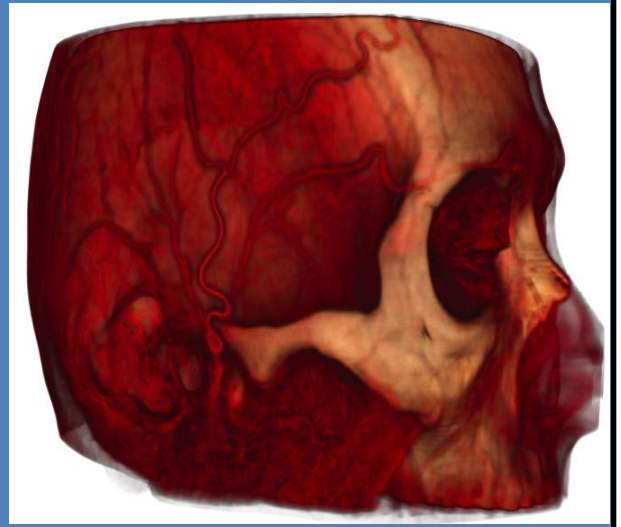
SS + Phong + AO



DVR

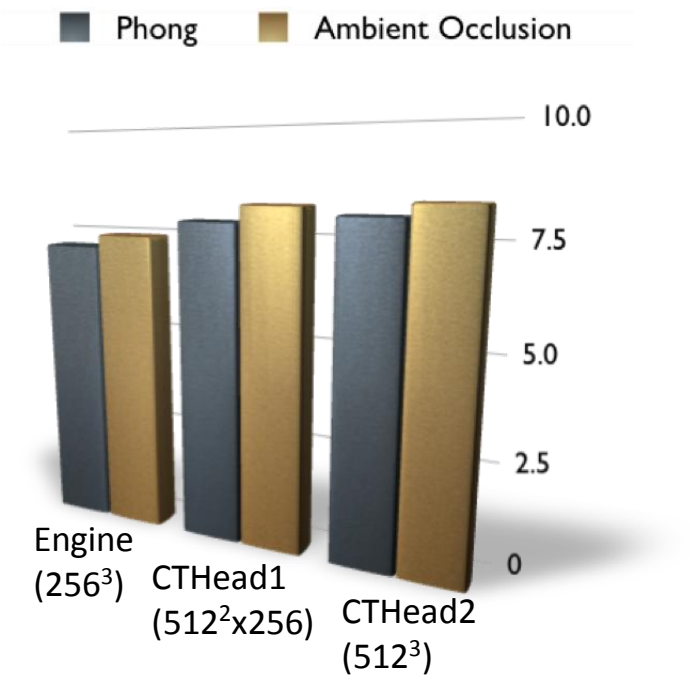


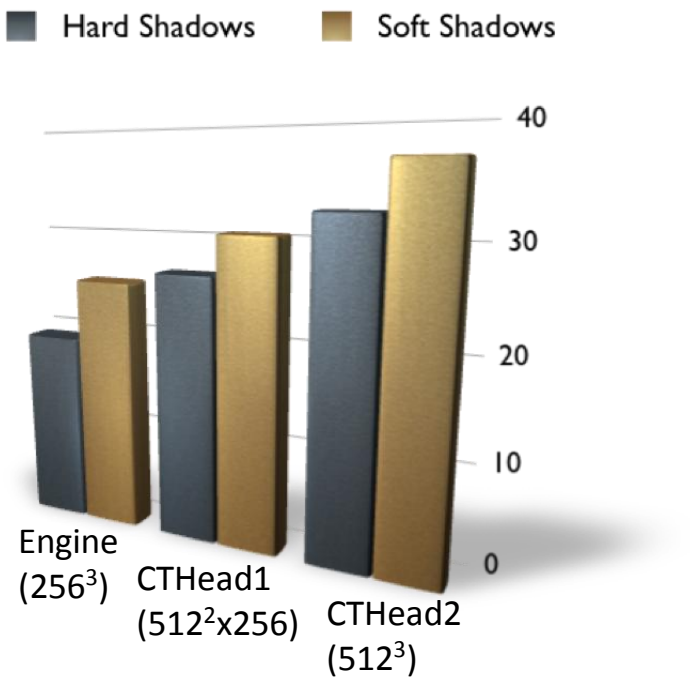
DVR + AO



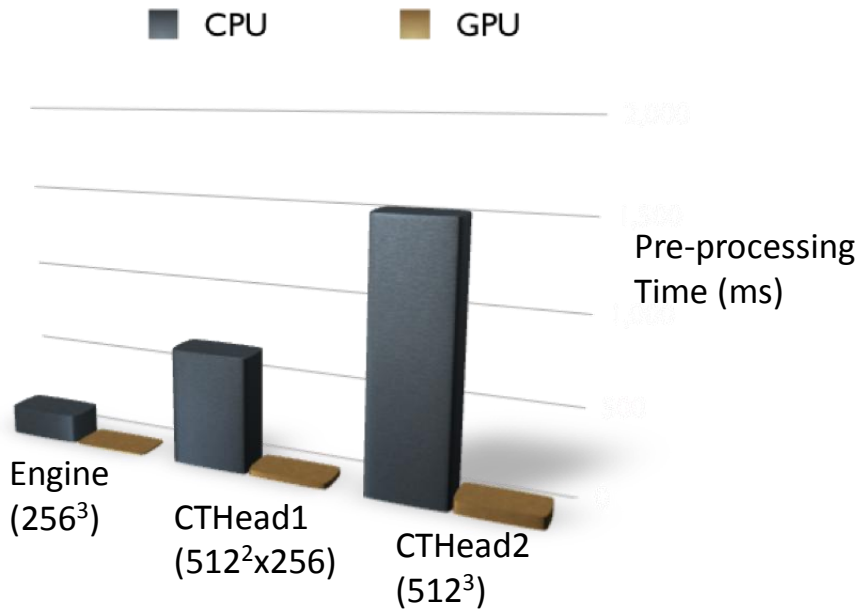
DVR + AO + Phong

Rendering Performance (ms)

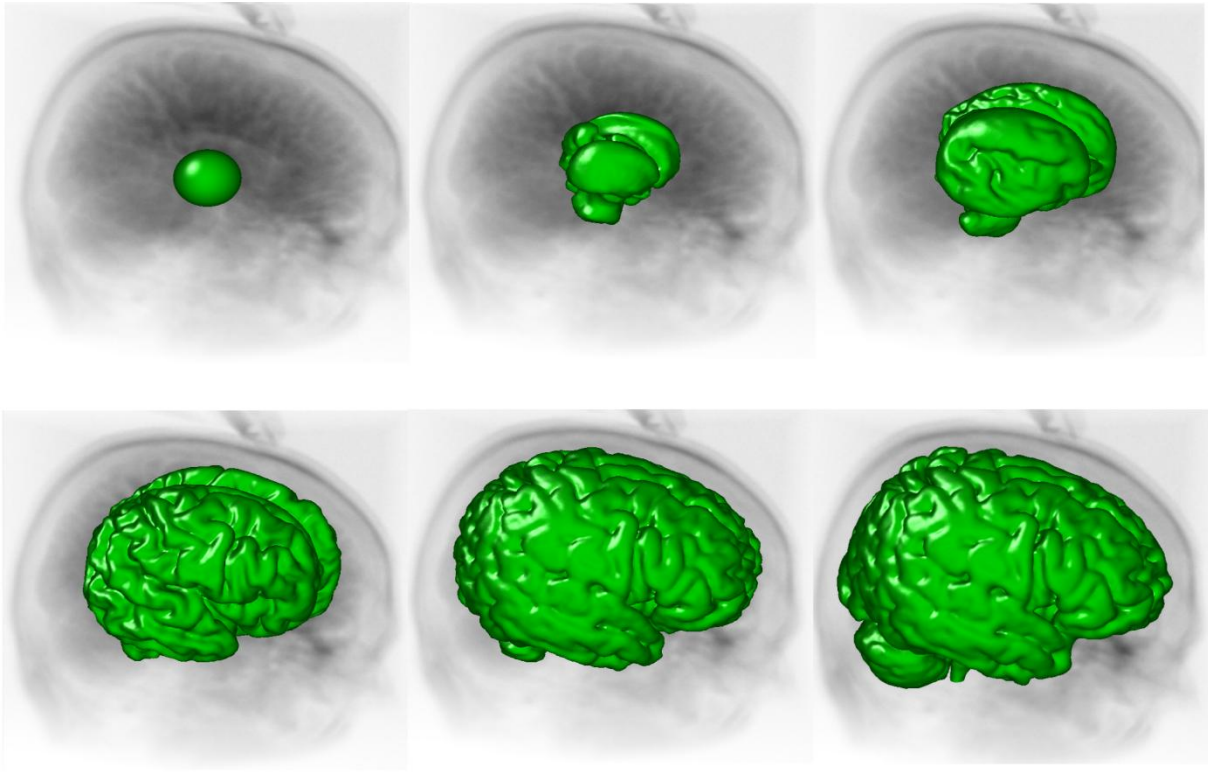




Pre-Processing (ms)

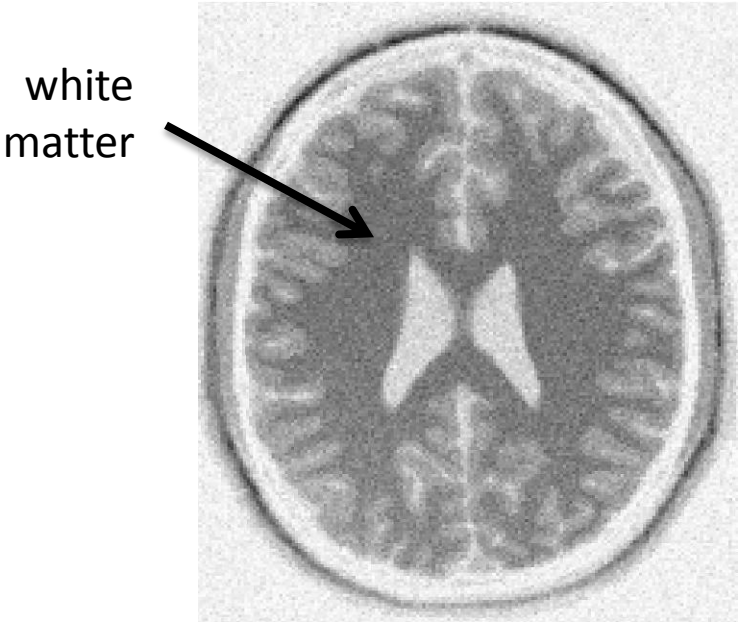


Efficient Level Set Segmentation on the GPU

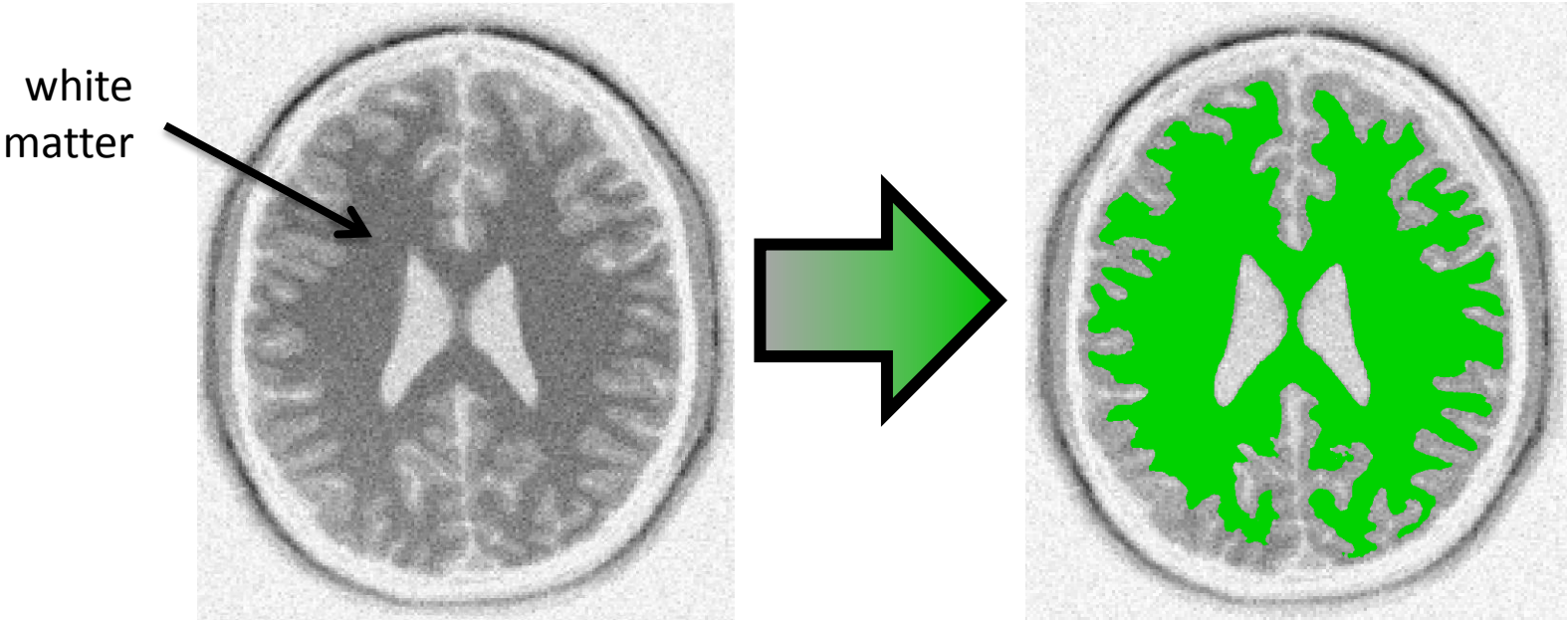


What do I mean by segmentation?

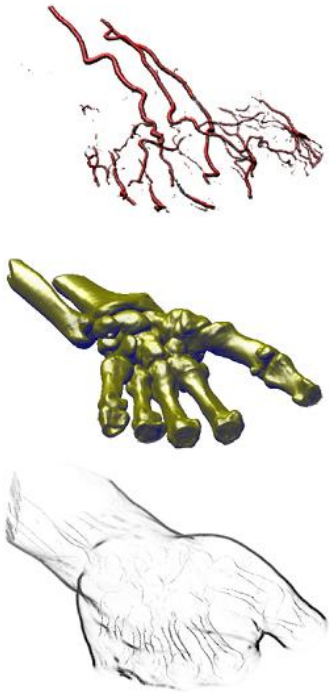
What do I mean by segmentation?

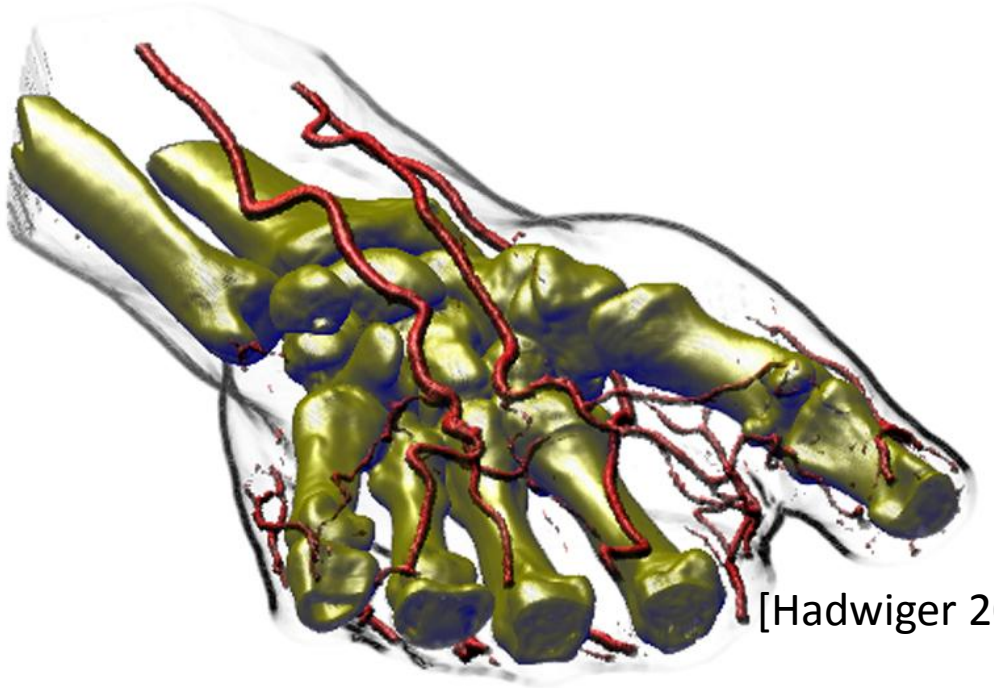


What do I mean by segmentation?



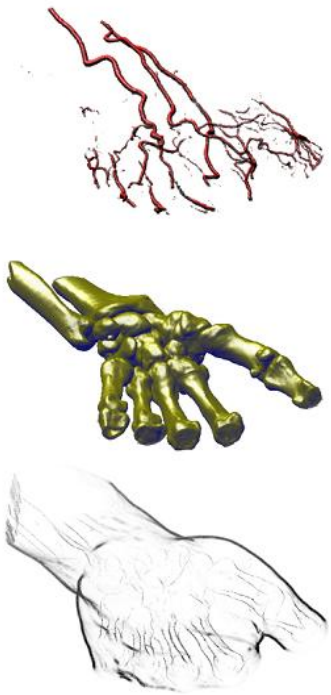
Synergy Between Segmentation and Visualization

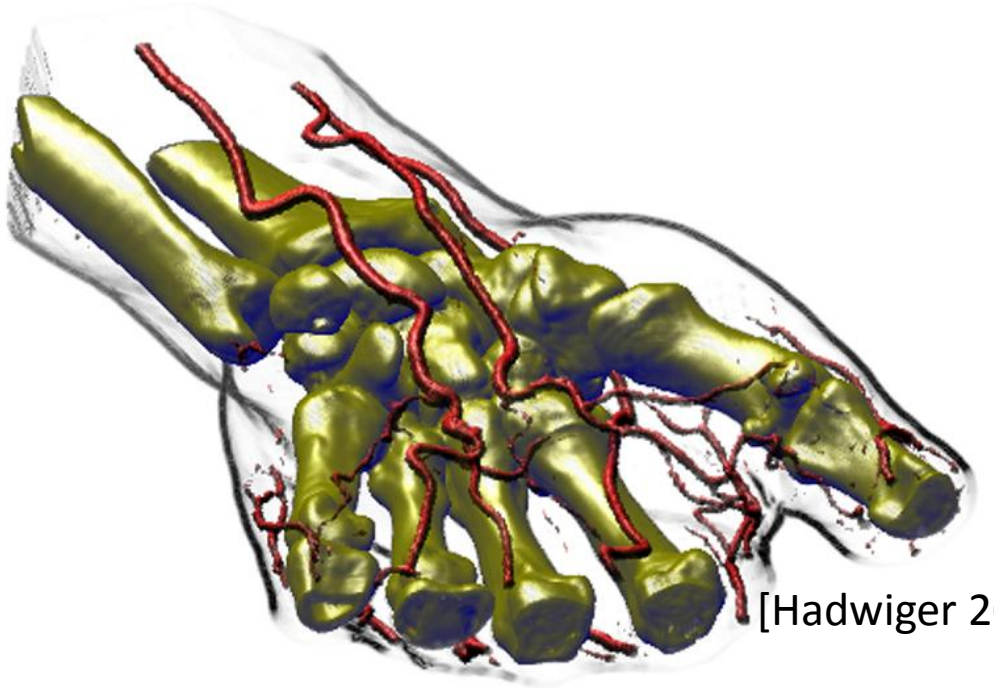




[Hadwiger 2004]

Synergy Between Segmentation and Visualization





[Hadwiger 2004]

Goals: fast, interactive, accurate

Why Level Sets?

Good: interactive [Lefohn et al. 2004]

accurate [Cates et al. 2004]

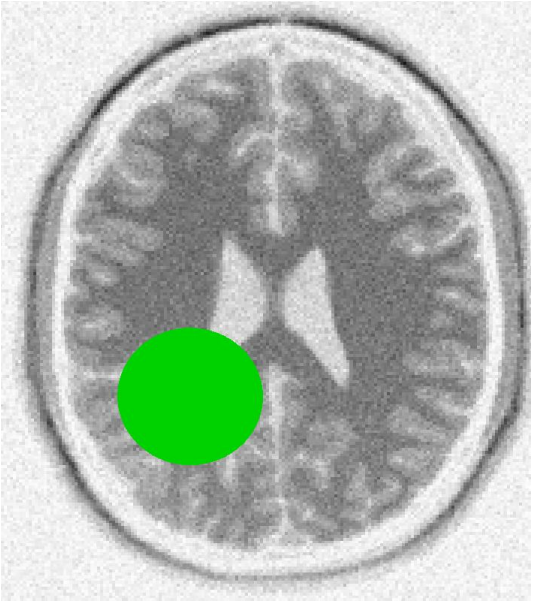
Why Level Sets?

Good: interactive [Lefohn et al. 2004]

accurate [Cates et al. 2004]

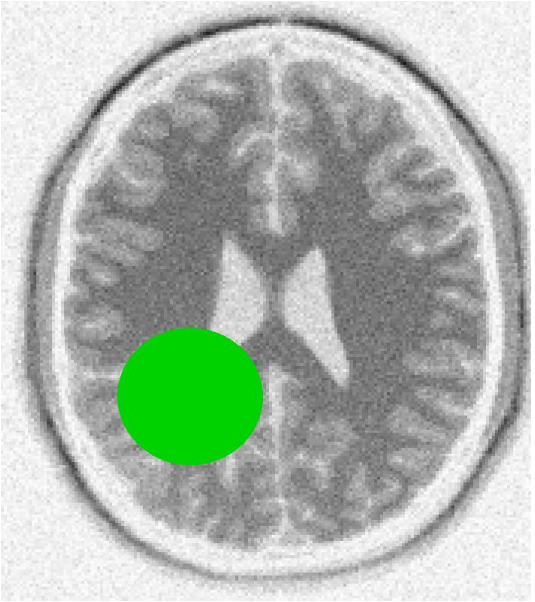
Bad: slow (even on the GPU) [Roberts et al. 2010]

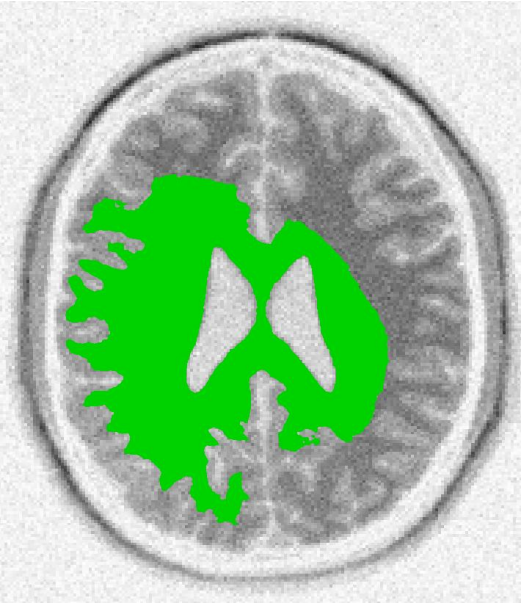
Segmentation with Level Sets



iterations

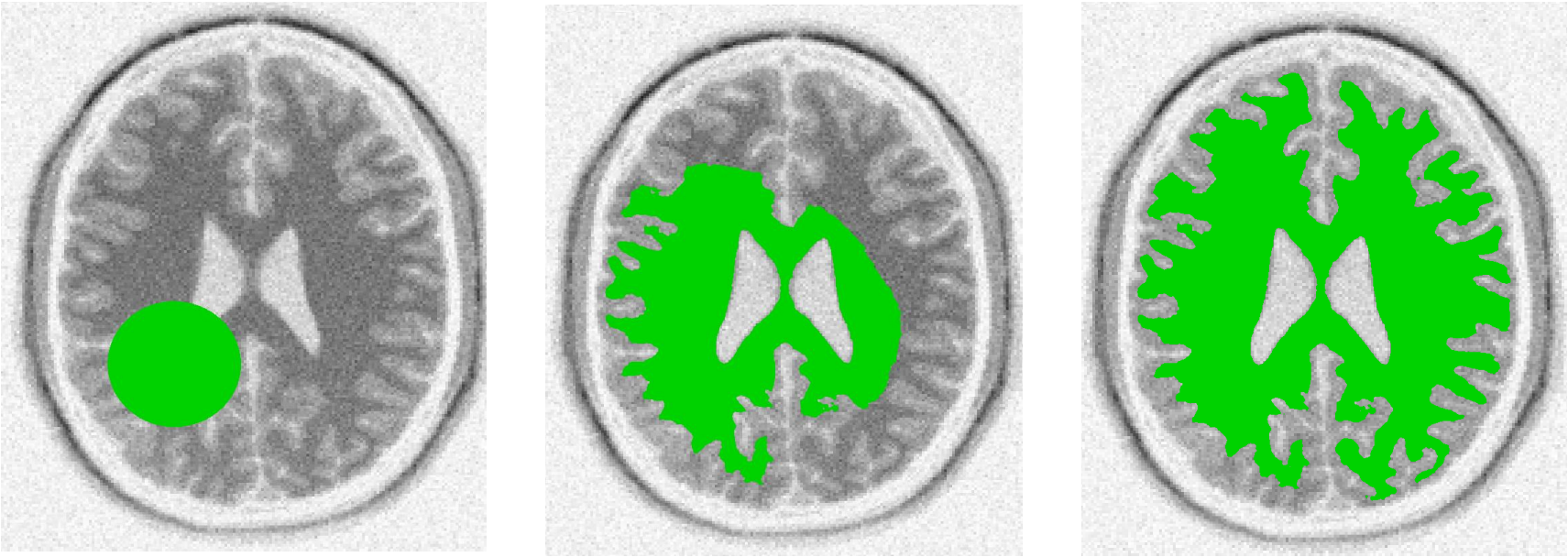
Segmentation with Level Sets





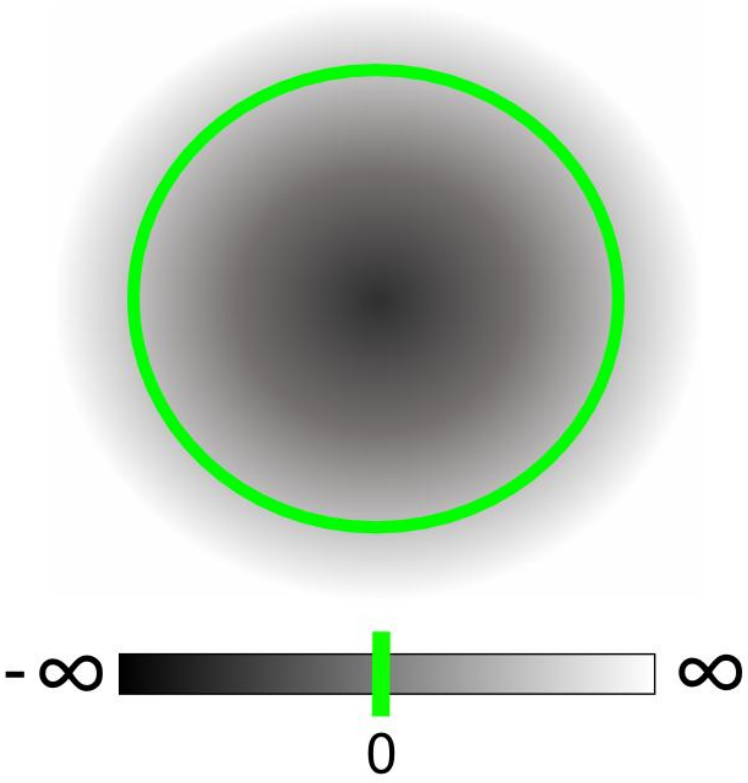
iterations

Segmentation with Level Sets



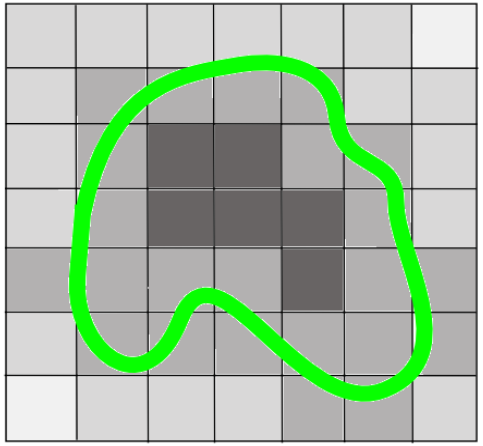
iterations

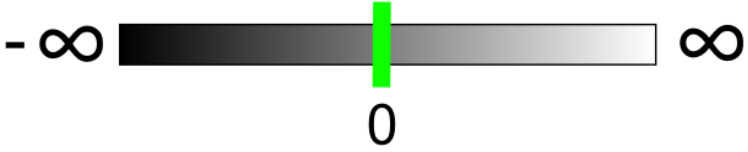
Segmentation with Level Sets



Segmentation with Level Sets

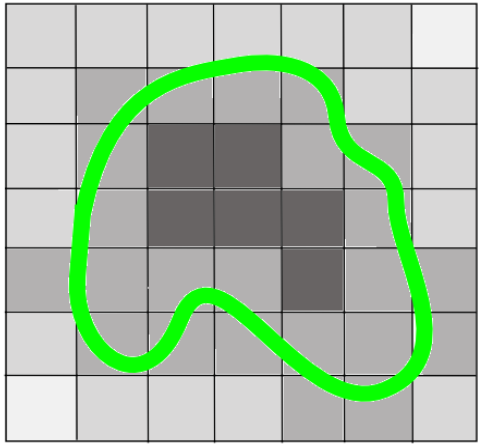
Before:



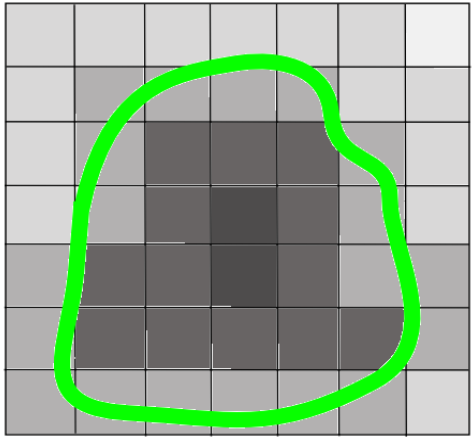


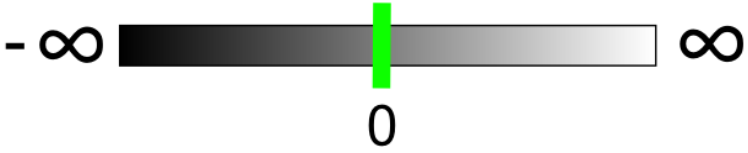
Segmentation with Level Sets

Before:



After:





Previous Work

- CPU
  - Narrow Band [Adalsteinson and Sethian 1995]
  - Sparse Field [Whitaker 1998; Peng et al. 1999]
  - Sparse Block Grid [Bridson 2003]
  - Dynamic Tubular Grid [Nielson and Museth 2006]
  - Heirarchical Run-Length-Encoded [Houston et al. 2006]

- GPU
  - GPU Narrow Band [Lefohn et al. 2003, 2004; Jeong et al. 2009]

Previous Work

- CPU
  - Narrow Band [Adalsteinson and Sethian 1995]
  - Sparse Field [Whitaker 1998; Peng et al. 1999]
  - Sparse Block Grid [Bridson 2003]
  - Dynamic Tubular Grid [Nielson and Museth 2006]
  - Heirarchical Run-Length-Encoded [Houston et al. 2006]
- GPU
  - GPU Narrow Band [Lefohn et al. 2003, 2004; Jeong et al. 2009]

leverage spatial coherence to reduce workload

Our Approach

Our Approach

*also leverage **temporal coherence** to further reduce computational workload*

Our Approach

also leverage ***temporal coherence*** *to further reduce computational workload*

…by only updating the voxels that will actually change their scalar value

Necessary Conditions for a Voxel to Change Value

Necessary Conditions for a Voxel to Change Value

spatial coherence: *close to the level set surface*

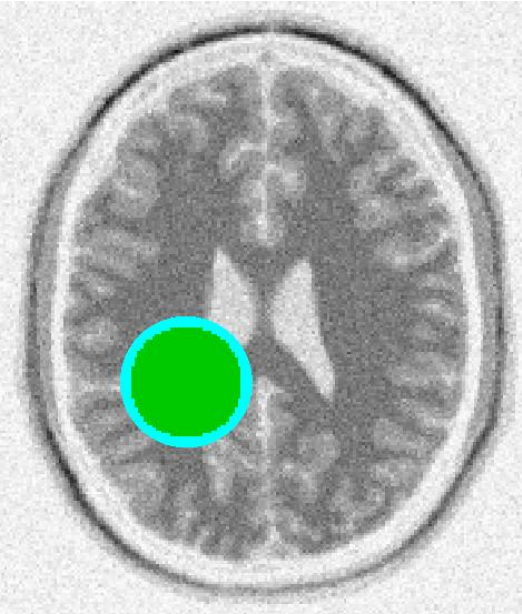
Necessary Conditions for a Voxel to Change Value

spatial coherence: *close to the level set surface*

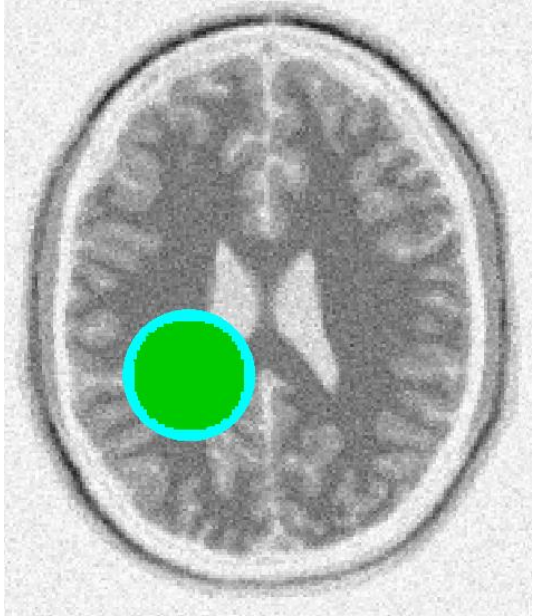
temporal coherence: *local neighborhood changing over time*

[Roberts et al. 2010]

Leveraging Temporal Coherence



spatial coherence only

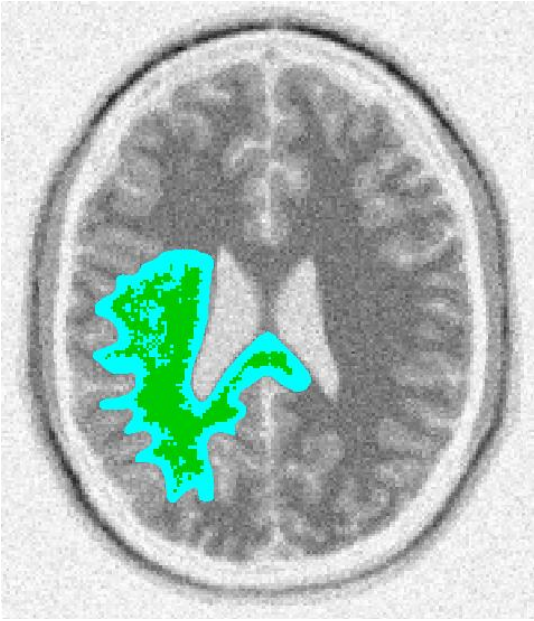


spatial and temporal coherence

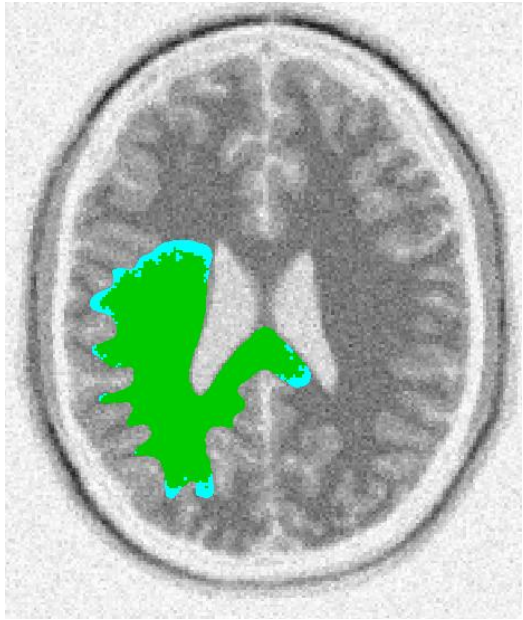
active computation

segmented region

Leveraging Temporal Coherence



spatial coherence only

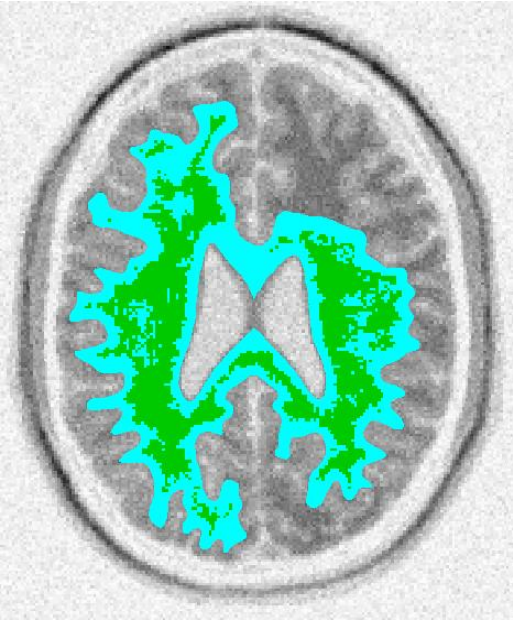


spatial and temporal coherence

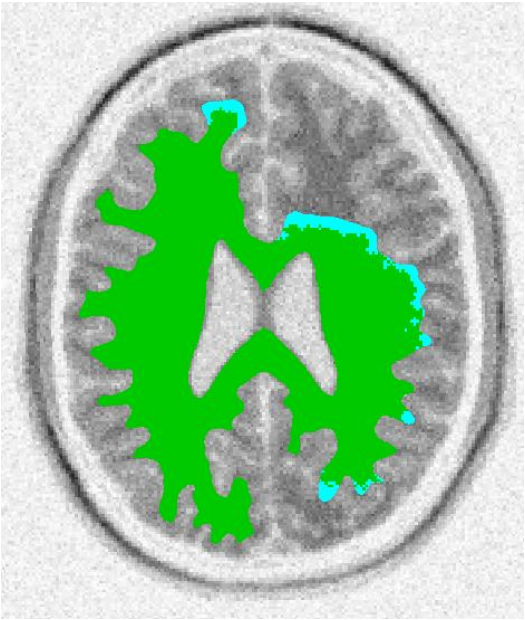
active computation

segmented region

Leveraging Temporal Coherence



spatial coherence only

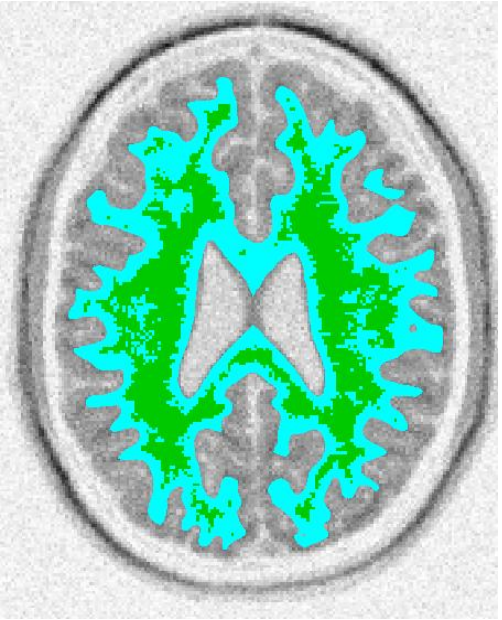


spatial and temporal coherence

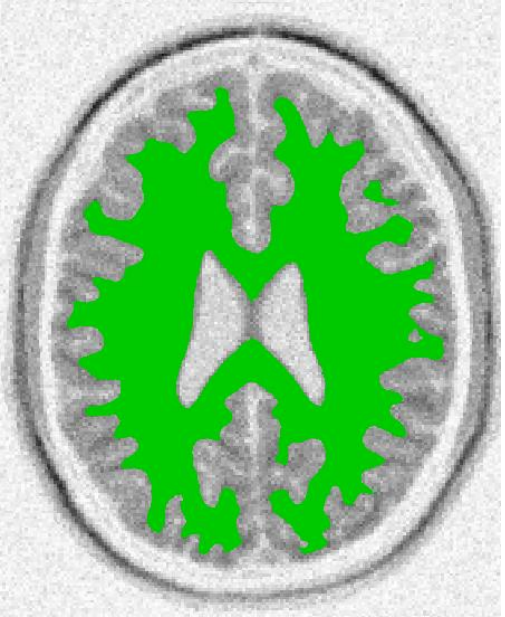
active computation

segmented region

Leveraging Temporal Coherence



spatial coherence only



spatial and temporal coherence

active computation

segmented region

Demo

GPU Algorithm

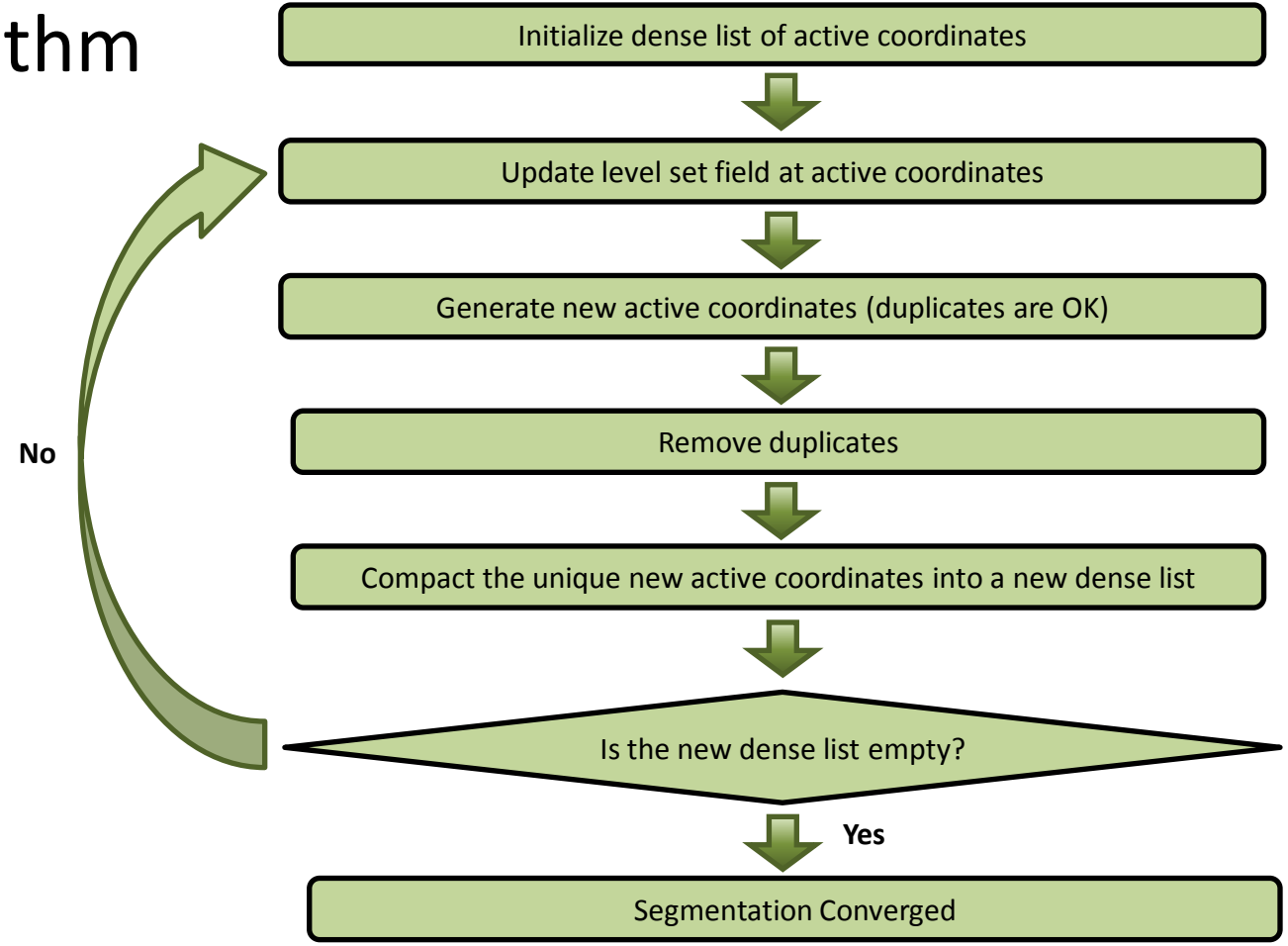
maintain a dense list of active coordinates

GPU Algorithm

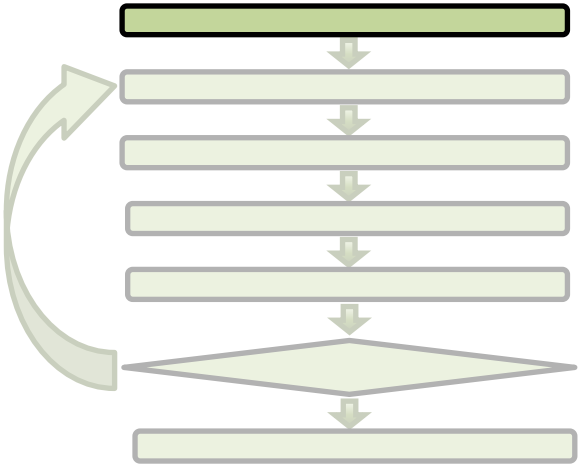
maintain a dense list of active coordinates

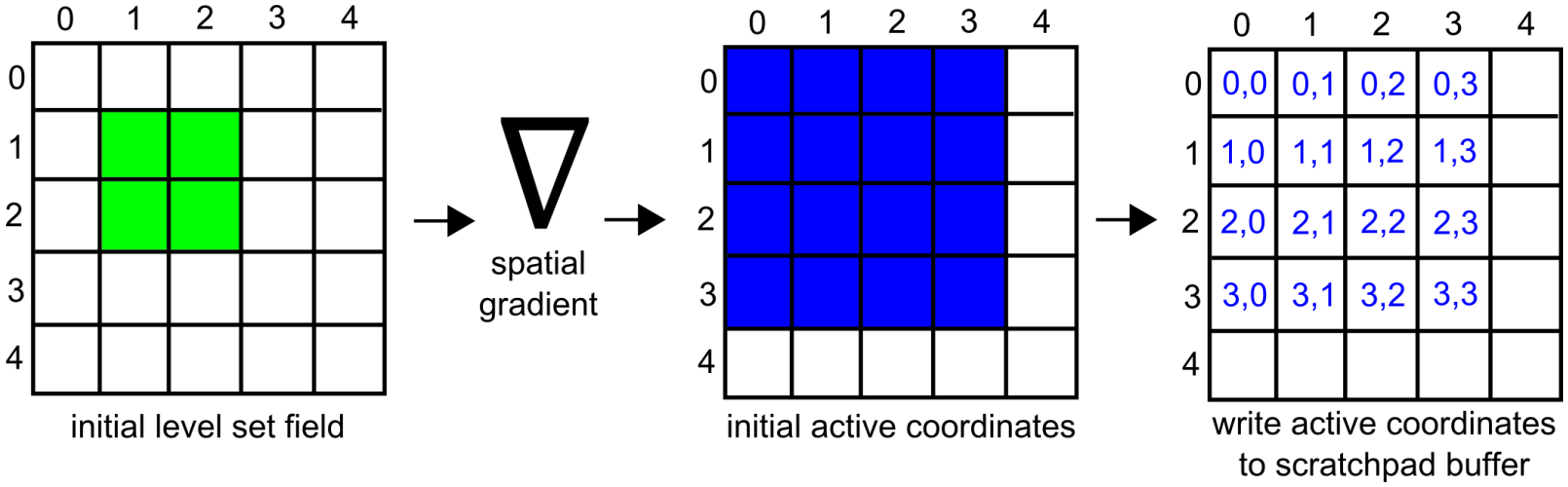
use it to sparsely update the level set field

GPU Algorithm

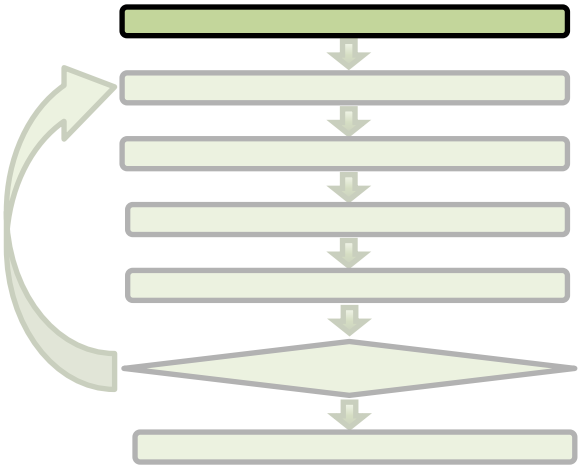


Initialize a scratchpad buffer with active coordinates





Interpret scratchpad as 1D

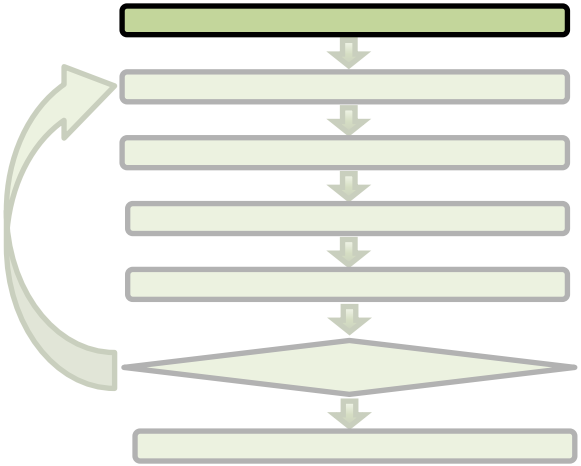


	0	1	2	3	4
0	0,0	0,1	0,2	0,3	
1	1,0	1,1	1,2	1,3	
2	2,0	2,1	2,2	2,3	
3	3,0	3,1	3,2	3,3	
4					

=

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0,0	0,1	0,2	0,3		1,0	1,1	1,2	1,3		2,0	2,1	2,2	2,3		3,0	3,1	3,2	3,3						

Compact the scratchpad buffer

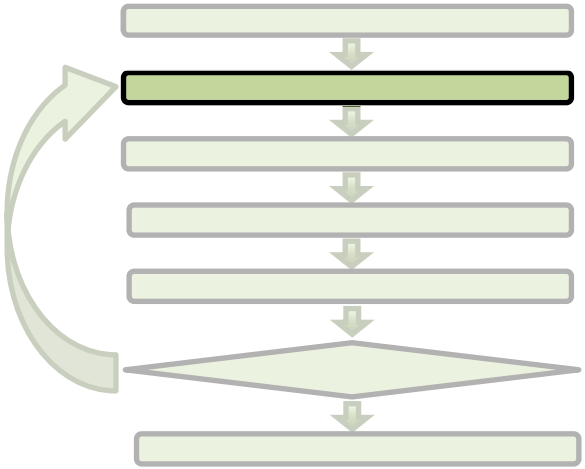


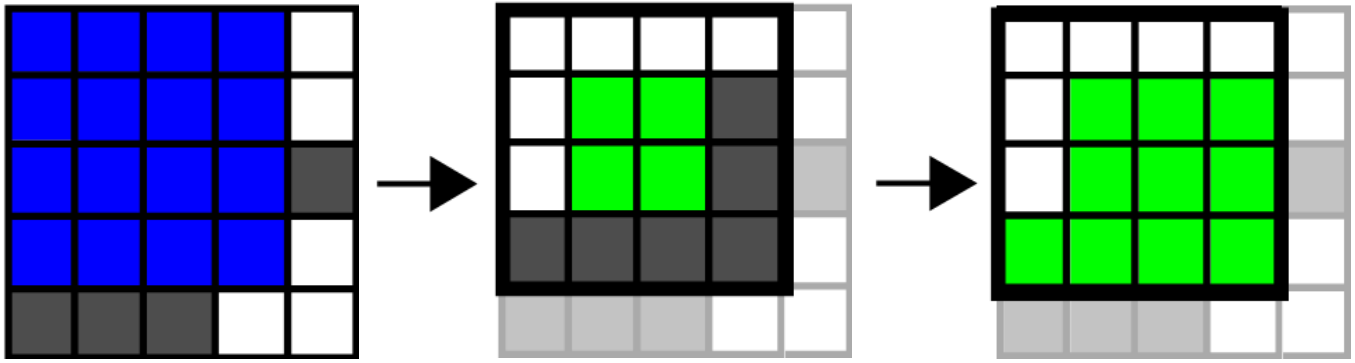
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
scratchpad buffer	0,0	0,1	0,2	0,3		1,0	1,1	1,2	1,3		2,0	2,1	2,2	2,3		3,0	3,1	3,2	3,3						

Stream Compaction

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
coordinate buffer	0,0	0,1	0,2	0,3	1,0	1,1	1,2	1,3	2,0	2,1	2,2	2,3	3,0	3,1	3,2	3,3									

Update the level set field at active coordinates



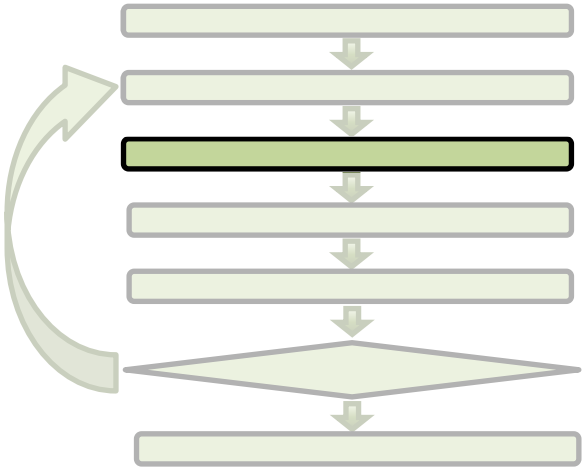


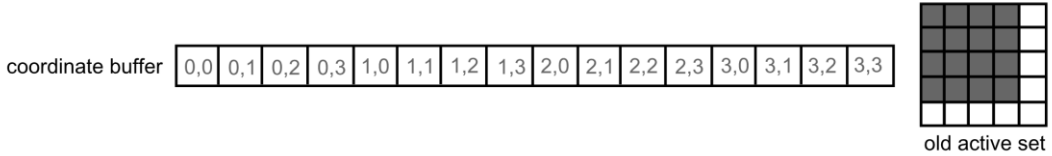
active coordinates

old level set field

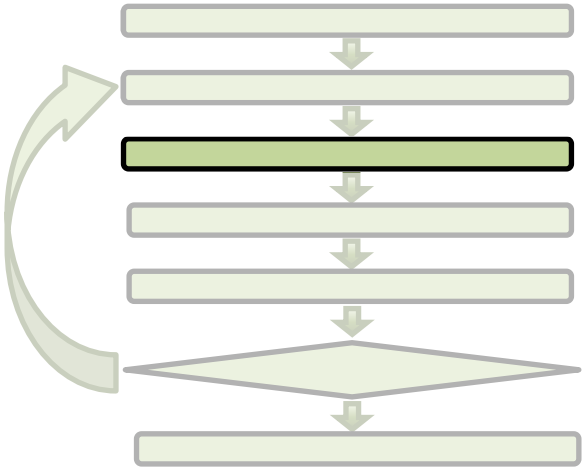
new level set field

Generate new active coordinates (duplicates are OK)



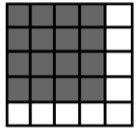


Generate new active coordinates (duplicates are OK)

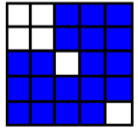


coordinate buffer

0,0	0,1	0,2	0,3	1,0	1,1	1,2	1,3	2,0	2,1	2,2	2,3	3,0	3,1	3,2	3,3

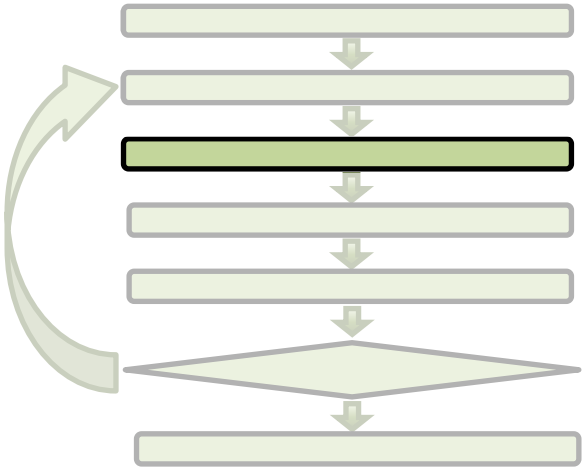


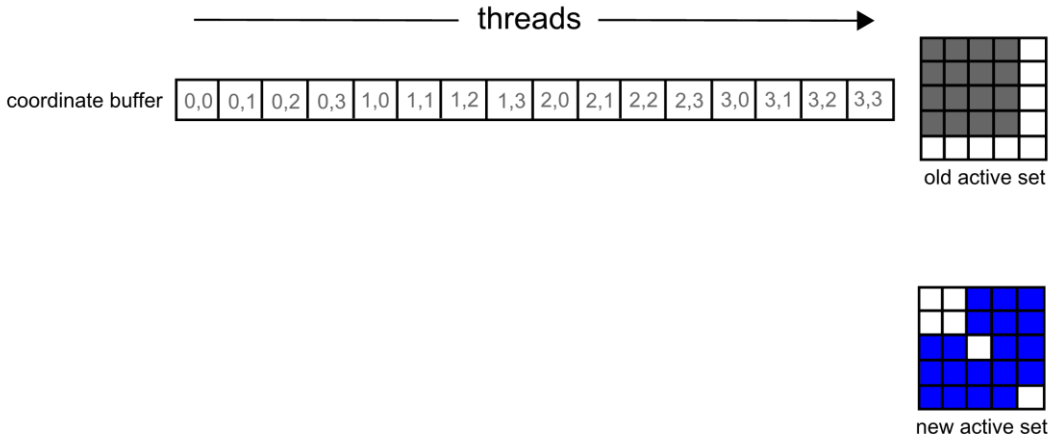
old active set



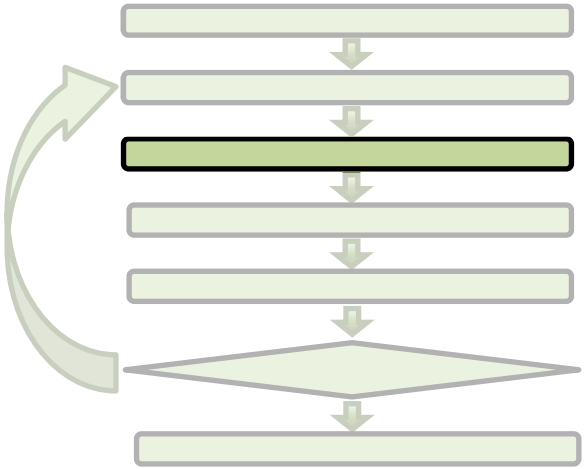
new active set

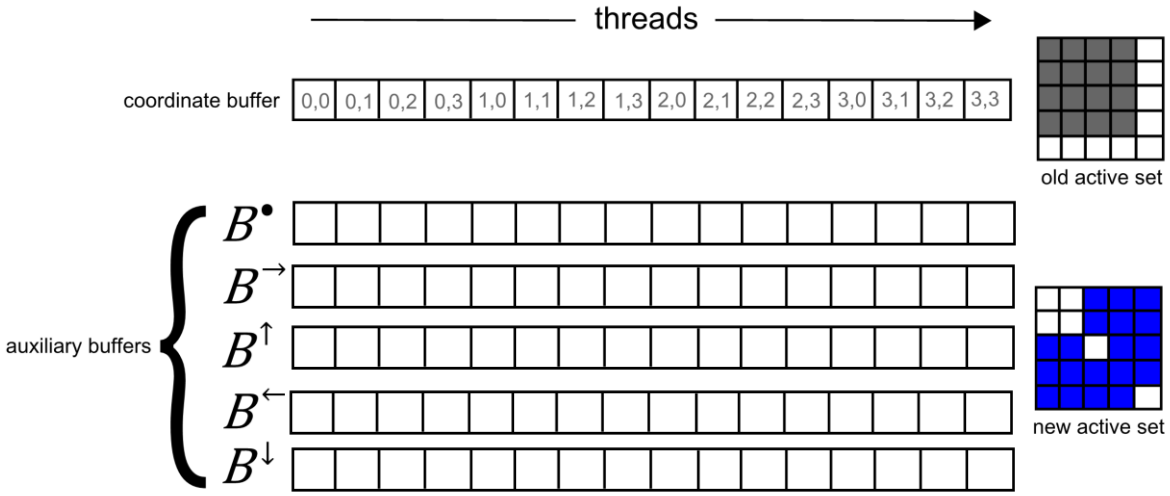
Generate new active coordinates (duplicates are OK)



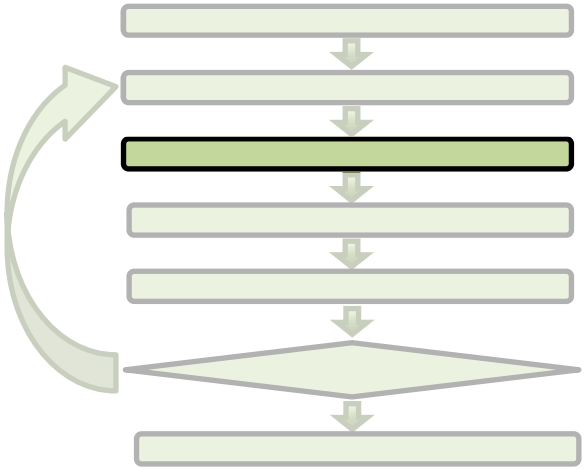


Generate new active coordinates (duplicates are OK)



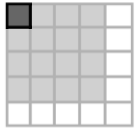


Generate new active coordinates (duplicates are OK)



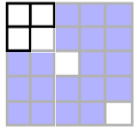
threads →

coordinate buffer: 0,0 | 0,1 | 0,2 | 0,3 | 1,0 | 1,1 | 1,2 | 1,3 | 2,0 | 2,1 | 2,2 | 2,3 | 3,0 | 3,1 | 3,2 | 3,3



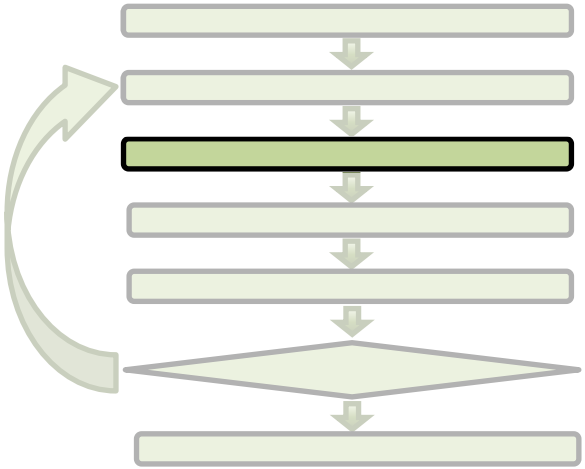
old active set

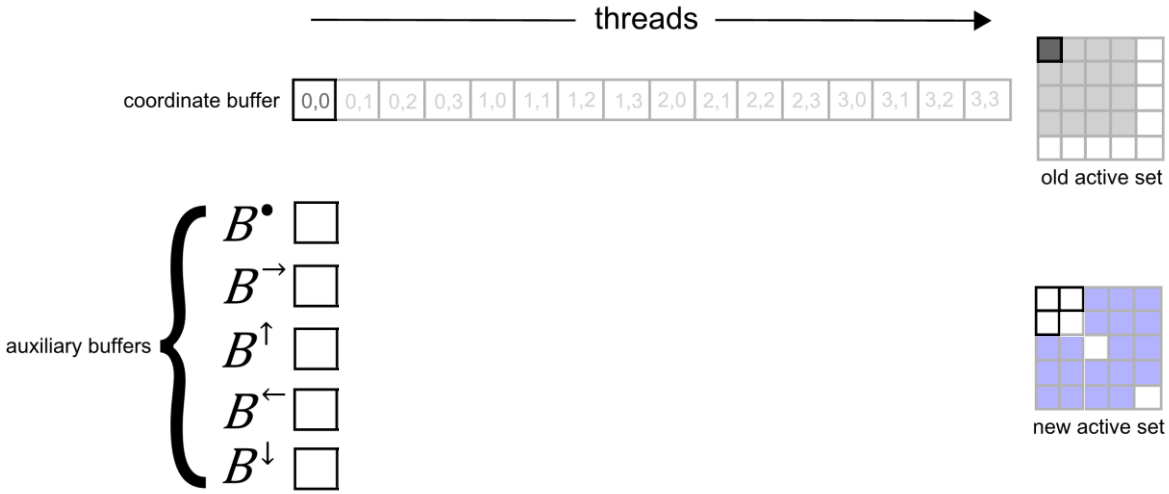
auxiliary buffers $\left\{ B^{\bullet}, B^{\rightarrow}, B^{\uparrow}, B^{\leftarrow}, B^{\downarrow} \right.$



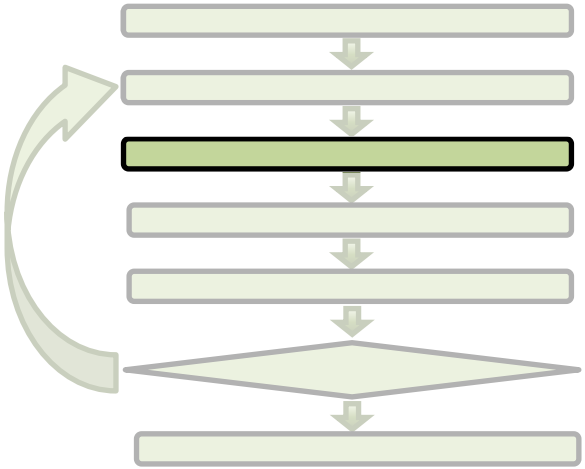
new active set

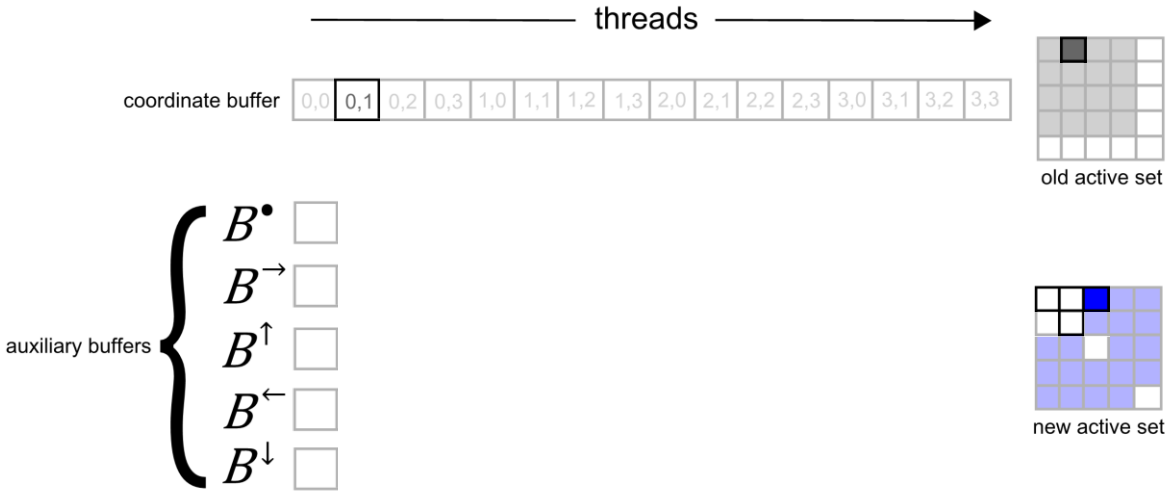
Generate new active coordinates (duplicates are OK)



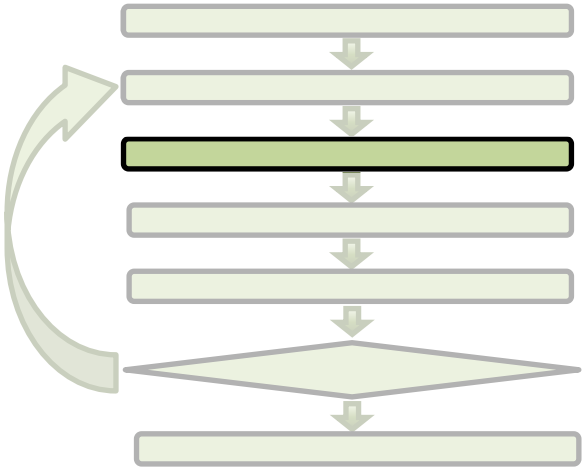


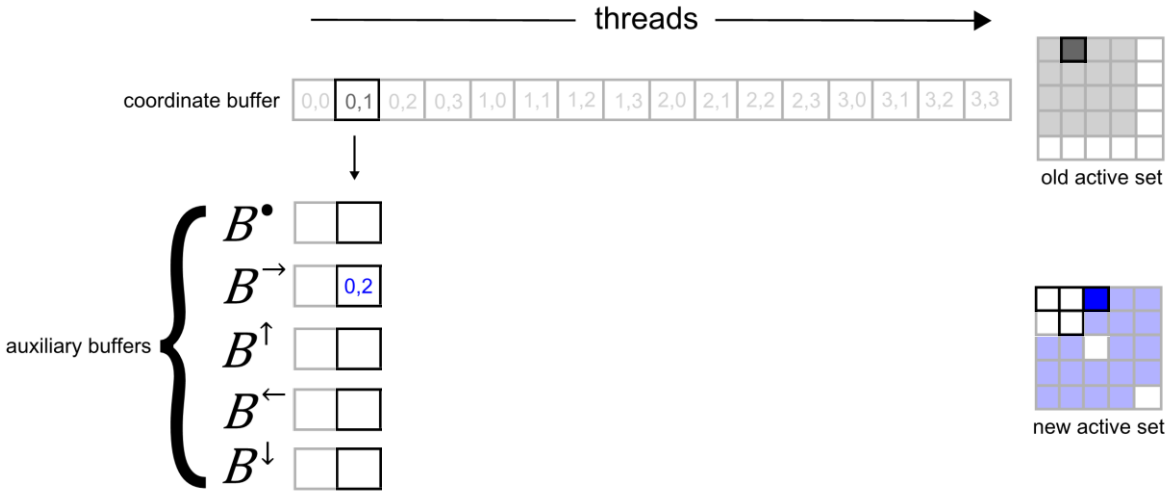
Generate new active coordinates (duplicates are OK)



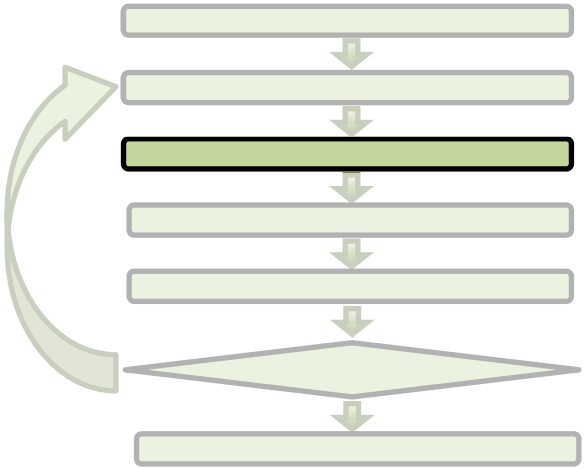


Generate new active coordinates (duplicates are OK)



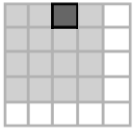


Generate new active coordinates (duplicates are OK)



threads →

coordinate buffer: 0,0 | 0,1 | 0,2 | 0,3 | 1,0 | 1,1 | 1,2 | 1,3 | 2,0 | 2,1 | 2,2 | 2,3 | 3,0 | 3,1 | 3,2 | 3,3



old active set

auxiliary buffers

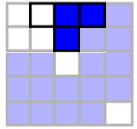
$B^{\bullet}$

$B^{\rightarrow}$ 0,2

$B^{\uparrow}$

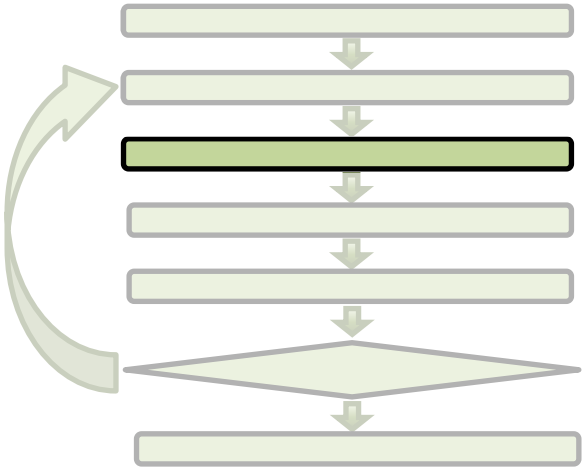
$B^{\leftarrow}$

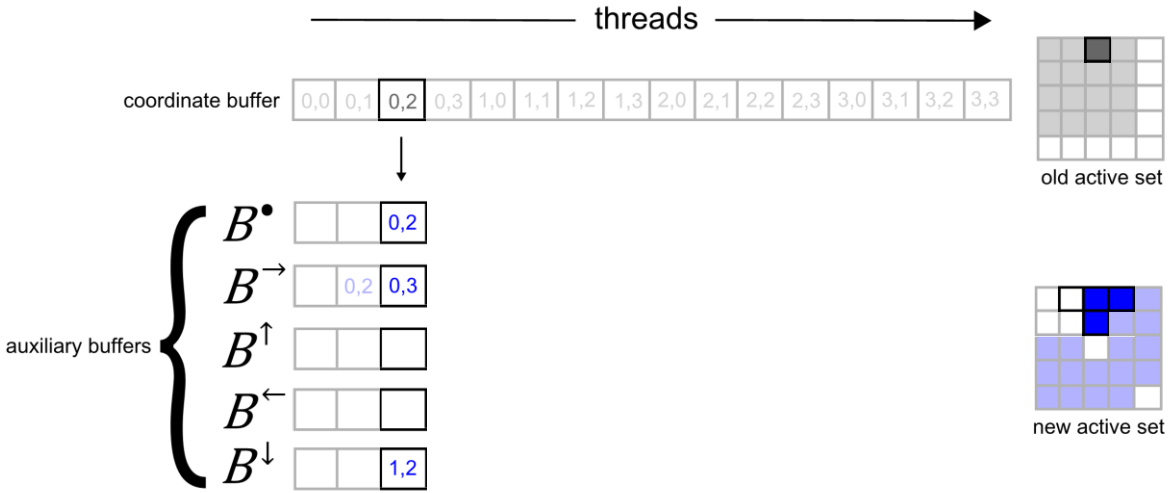
$B^{\downarrow}$



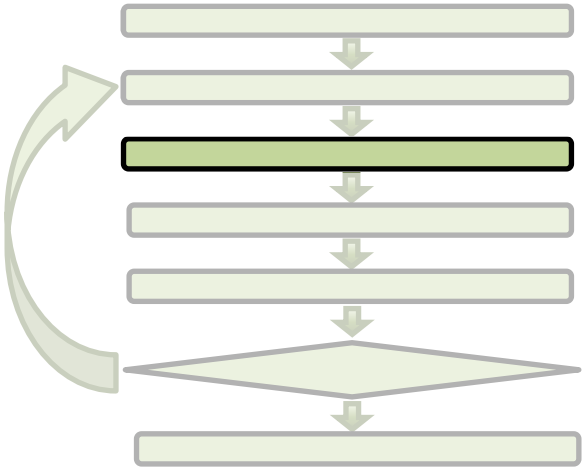
new active set

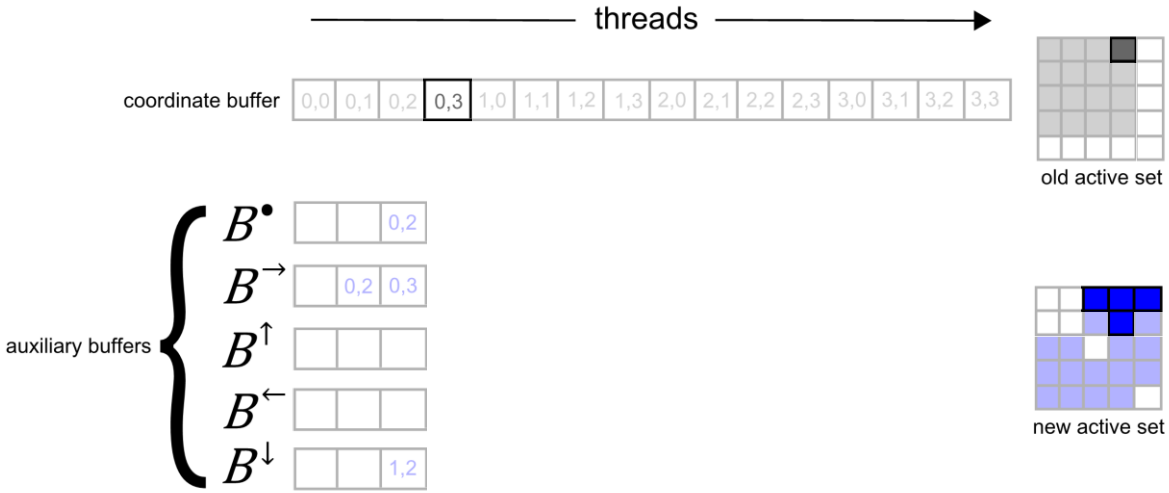
Generate new active coordinates (duplicates are OK)



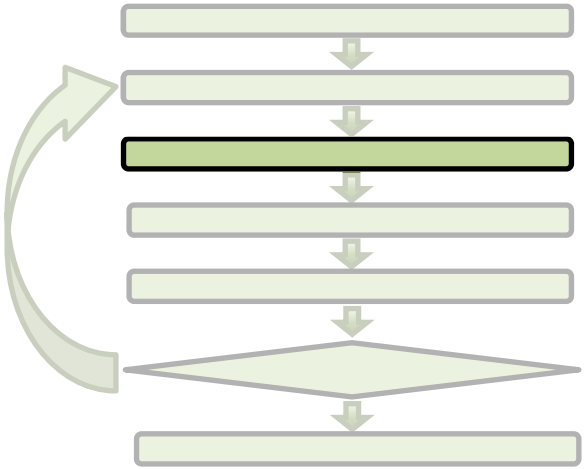


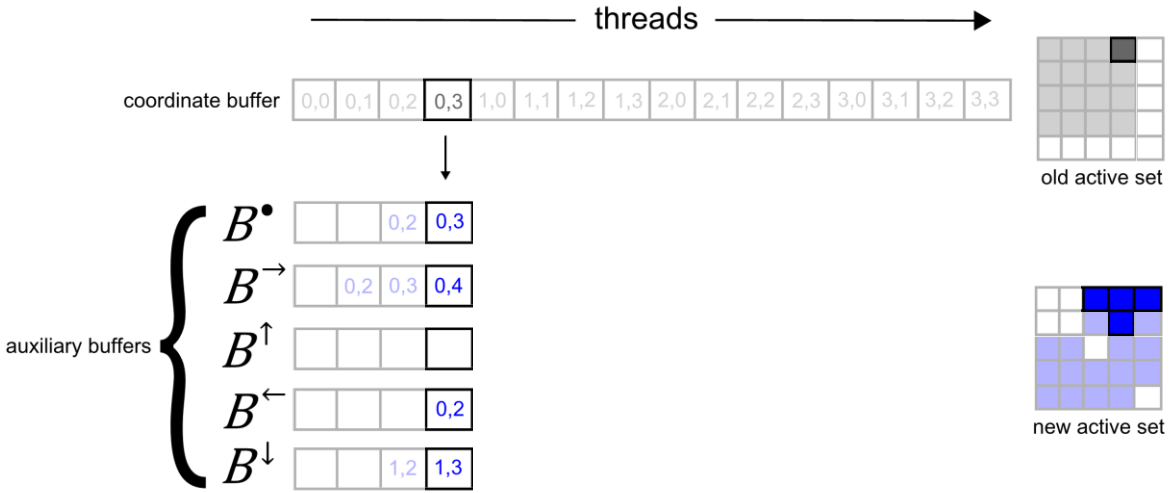
Generate new active coordinates (duplicates are OK)



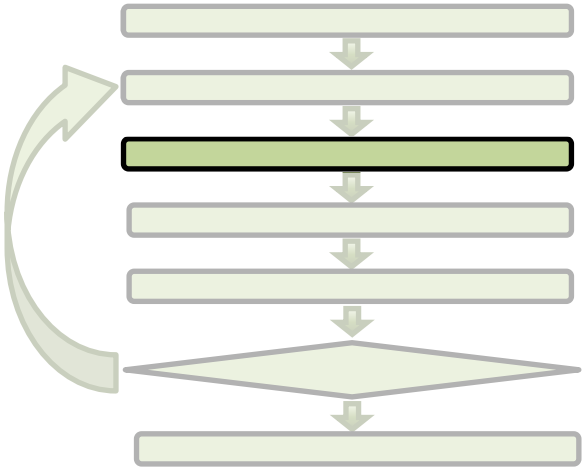


Generate new active coordinates (duplicates are OK)





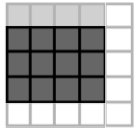
Generate new active coordinates (duplicates are OK)



threads →

coordinate buffer

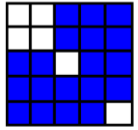
0,0	0,1	0,2	0,3	1,0	1,1	1,2	1,3	2,0	2,1	2,2	2,3	3,0	3,1	3,2	3,3



old active set

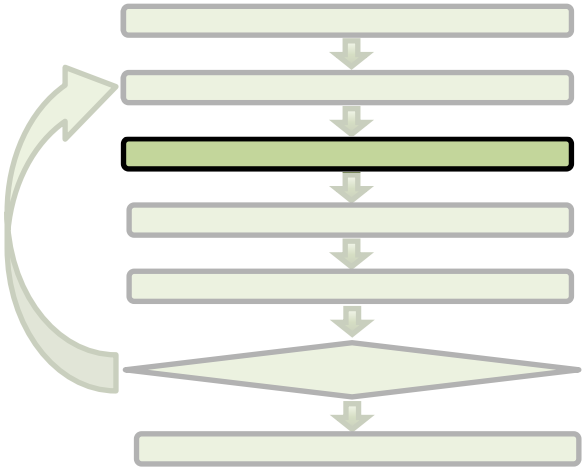
auxiliary buffers

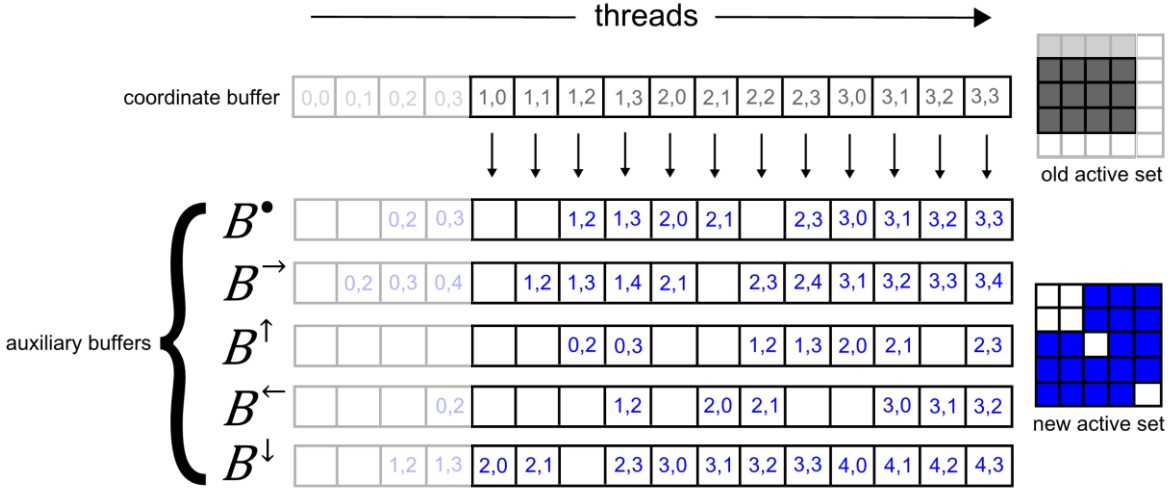
$B^{\bullet}$			0,2	0,3
$B^{\rightarrow}$		0,2	0,3	0,4
$B^{\uparrow}$				
$B^{\leftarrow}$				0,2
$B^{\downarrow}$			1,2	1,3



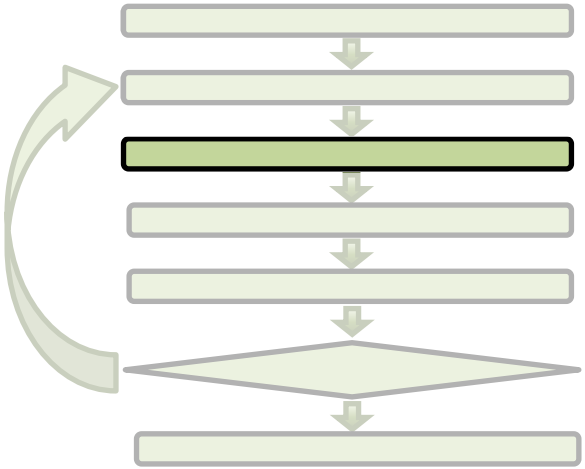
new active set

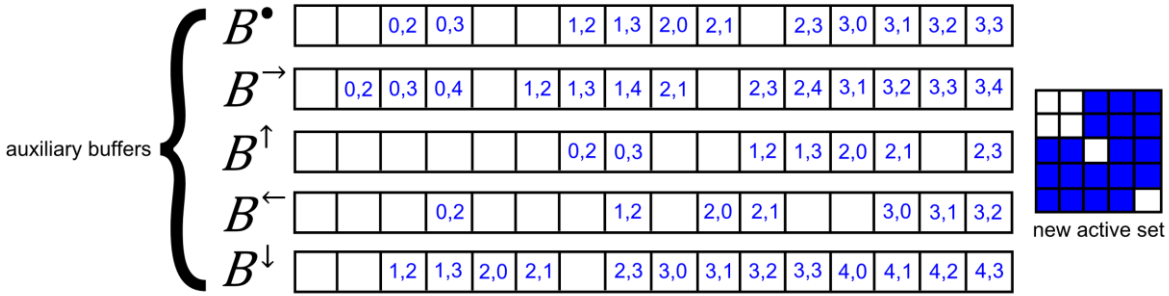
Generate new active coordinates (duplicates are OK)



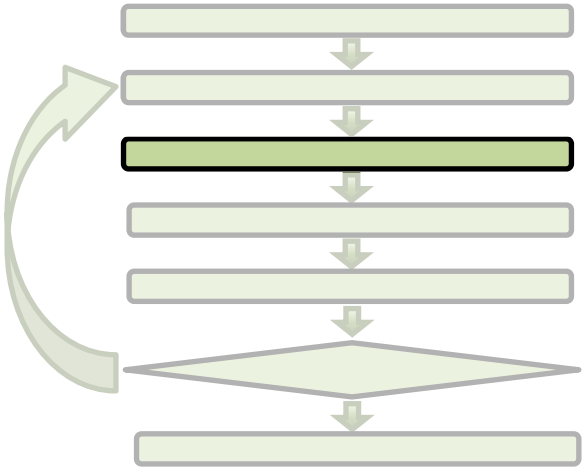


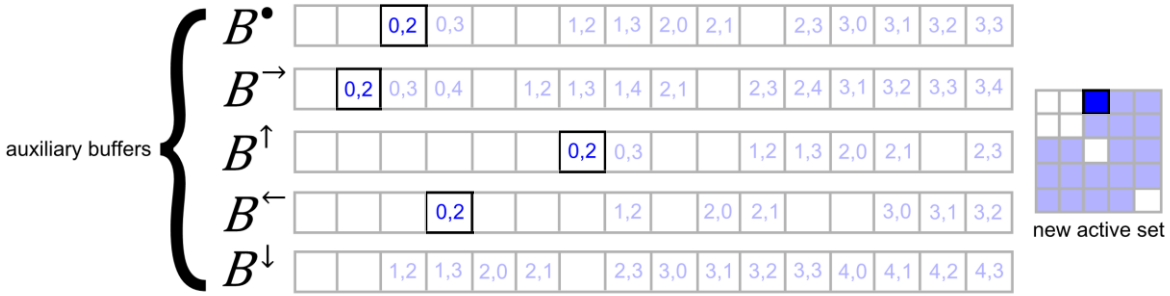
Generate new active coordinates (duplicates are OK)



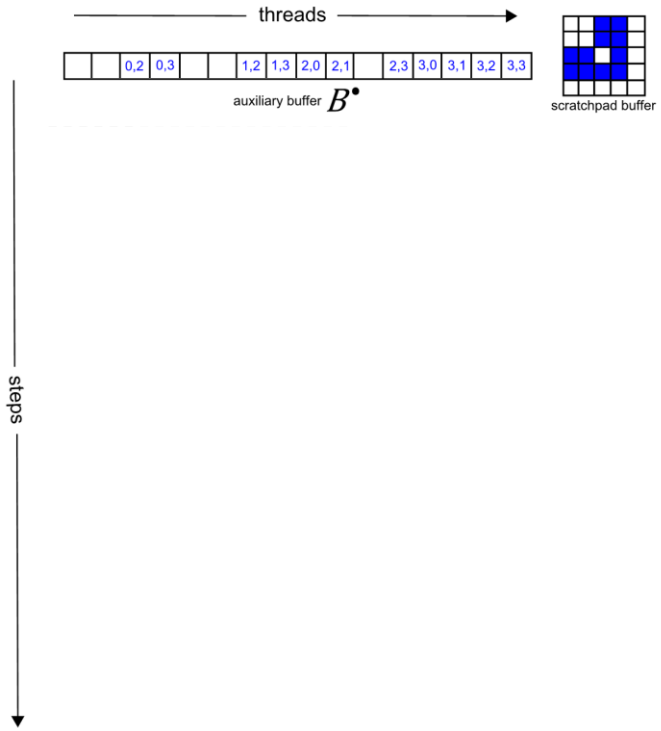


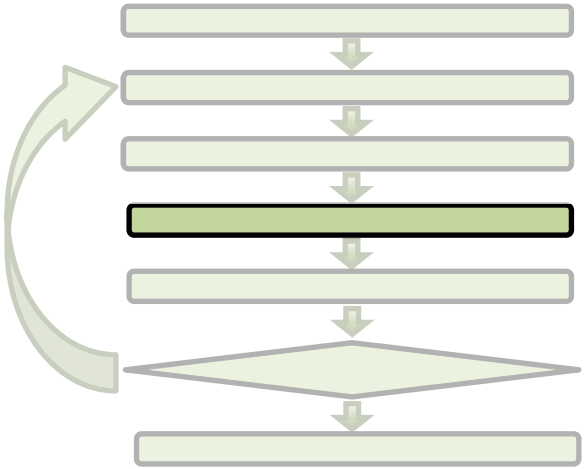
Generate new active coordinates (duplicates are OK)





Remove duplicates



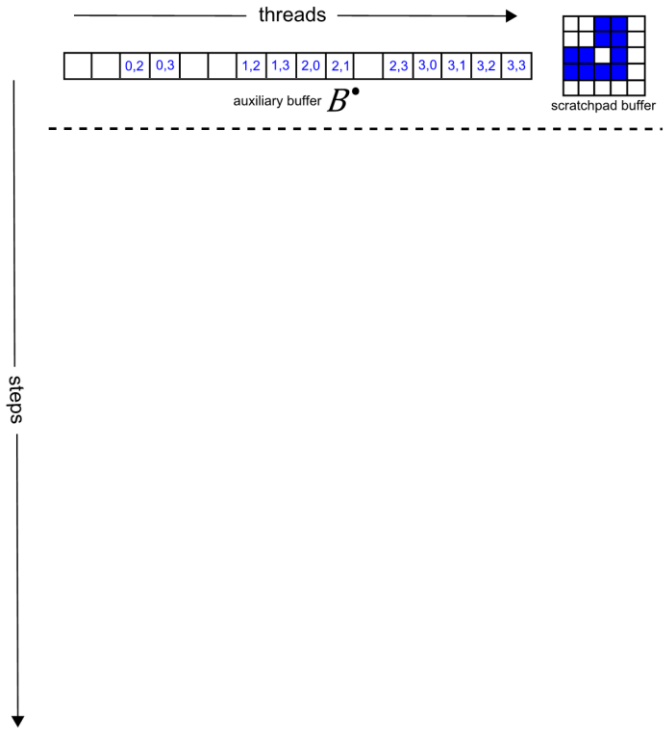


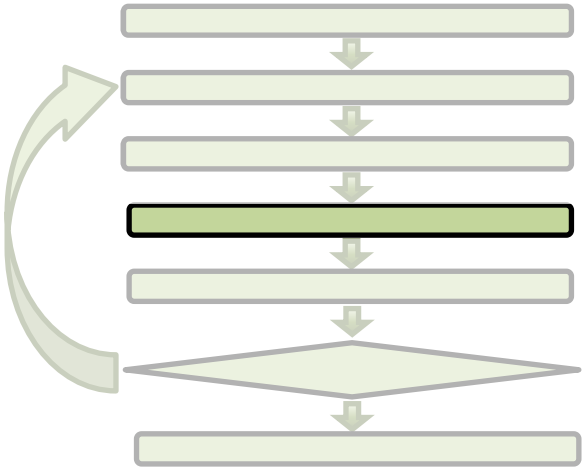
global synchronization

not previously tagged → tag in scratchpad

previously tagged → remove from auxiliary

Remove duplicates



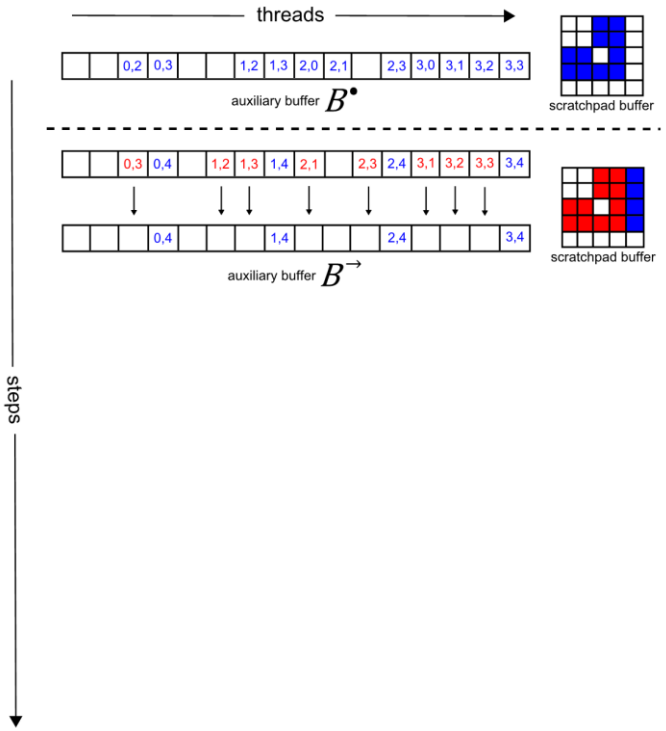


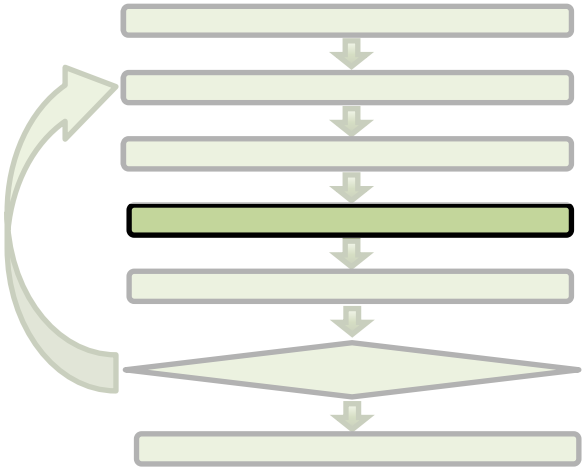
global synchronization

not previously tagged → tag in scratchpad

previously tagged → remove from auxiliary

Remove duplicates



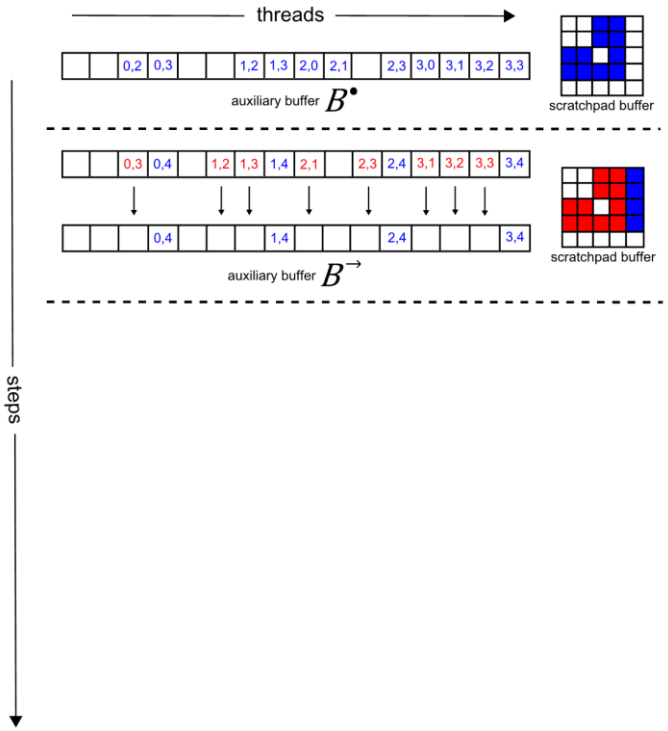


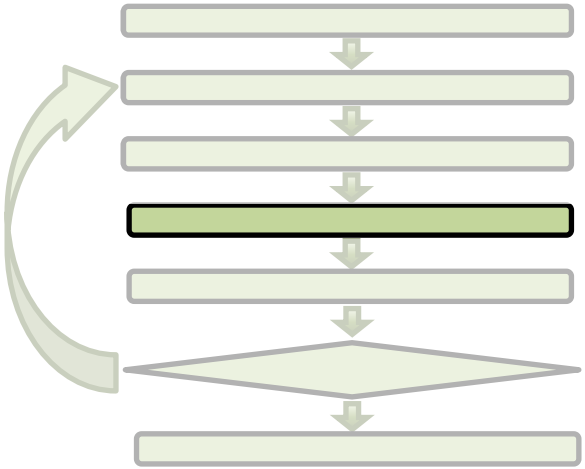
global synchronization

not previously tagged → tag in scratchpad

previously tagged → remove from auxiliary

Remove duplicates



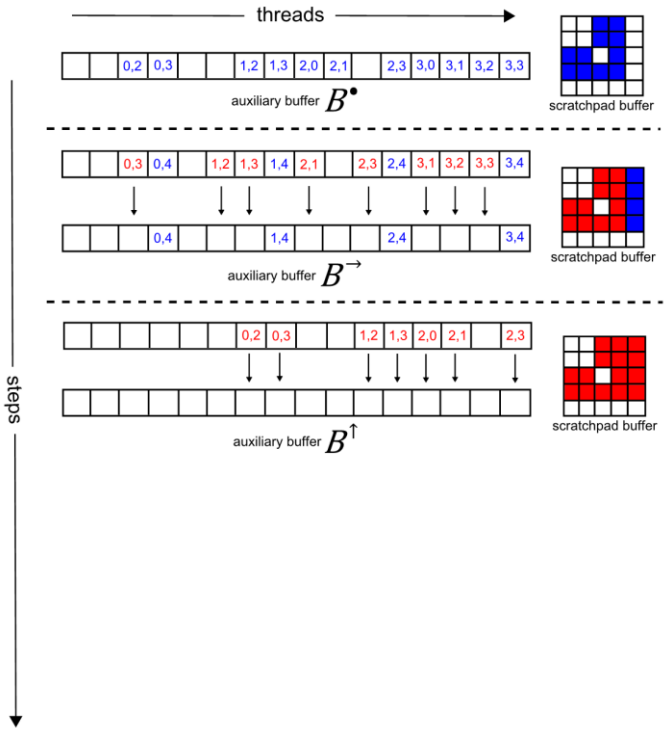


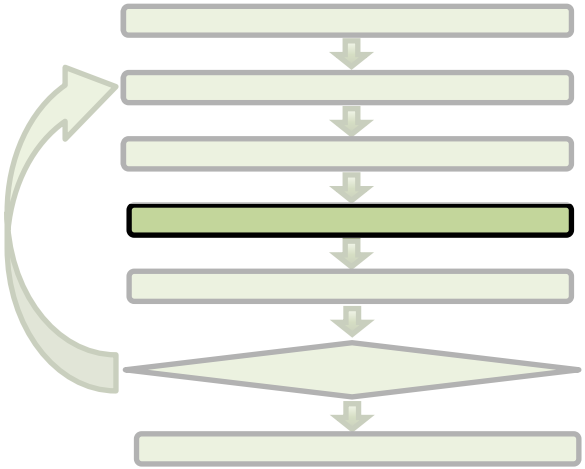
global synchronization

not previously tagged → tag in scratchpad

previously tagged → remove from auxiliary

Remove duplicates



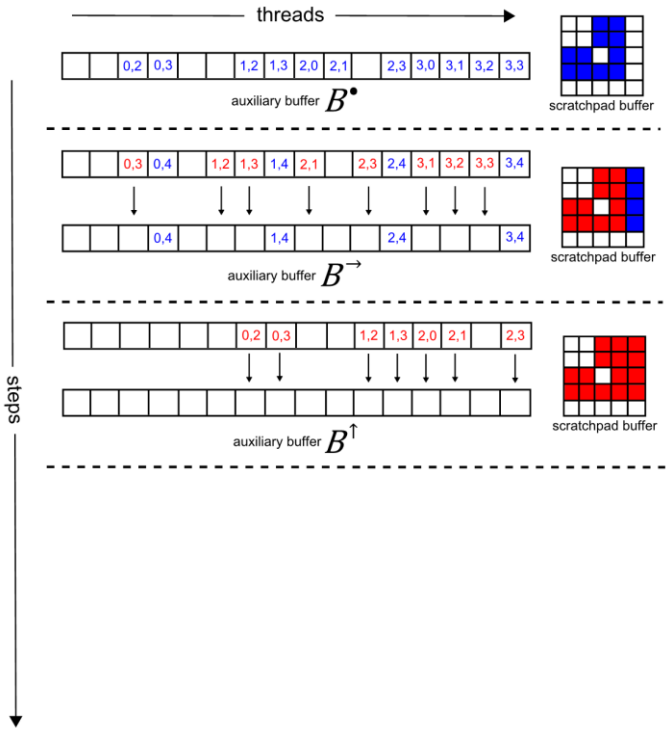


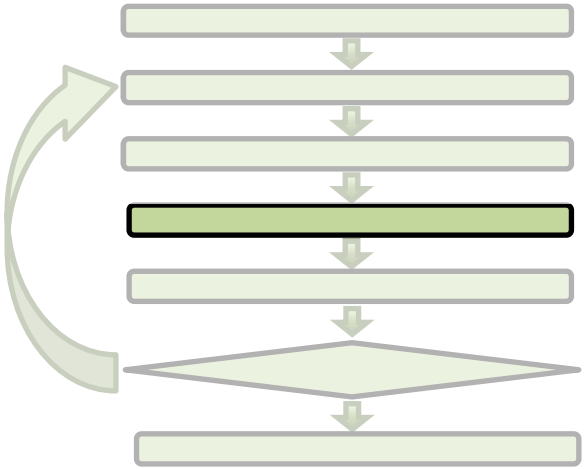
global synchronization

not previously tagged → tag in scratchpad

previously tagged → remove from auxiliary

Remove duplicates



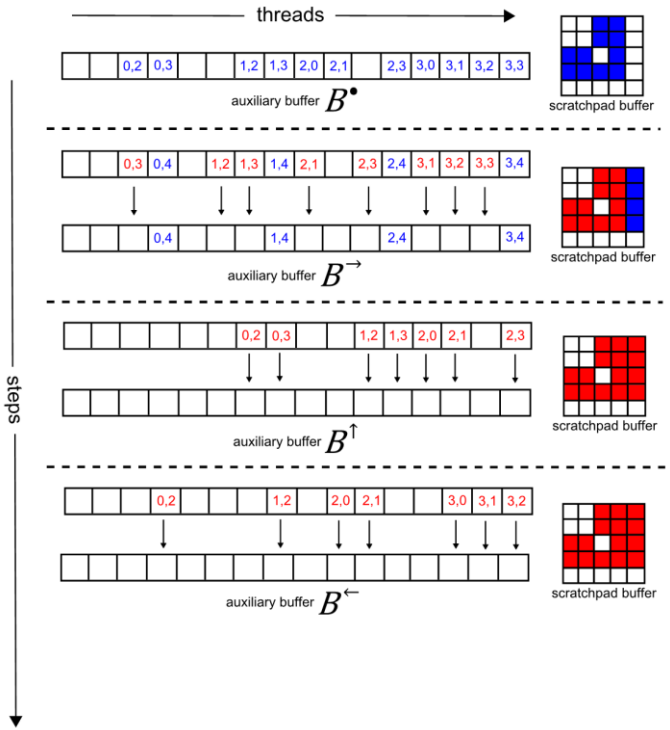


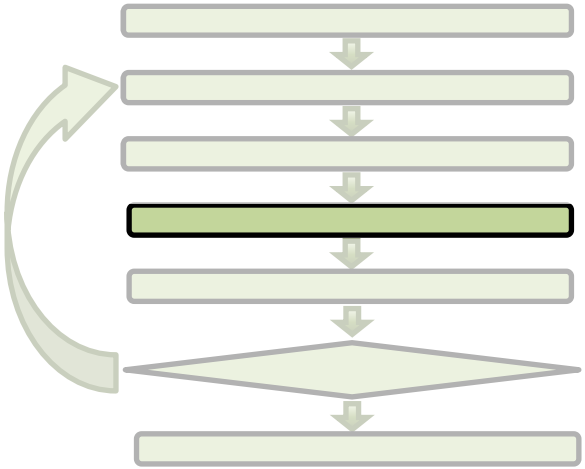
global synchronization

not previously tagged → tag in scratchpad

previously tagged → remove from auxiliary

Remove duplicates



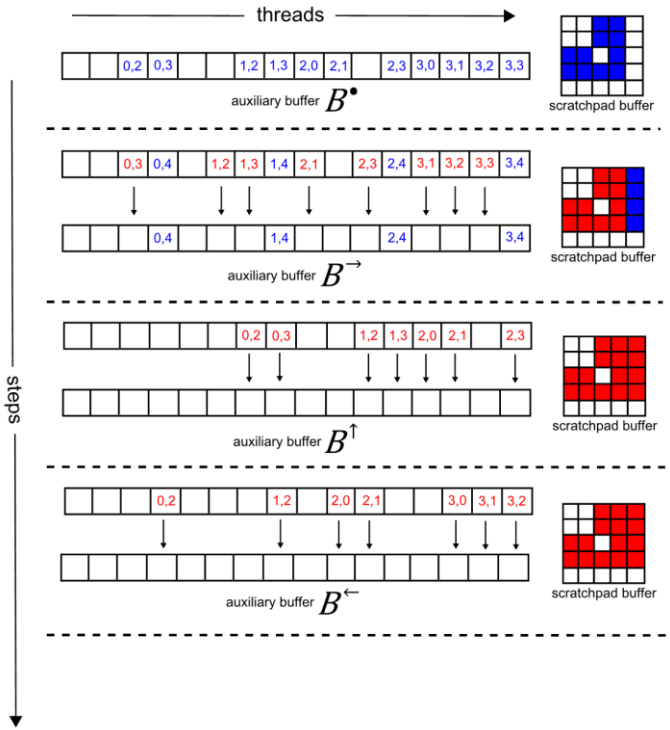


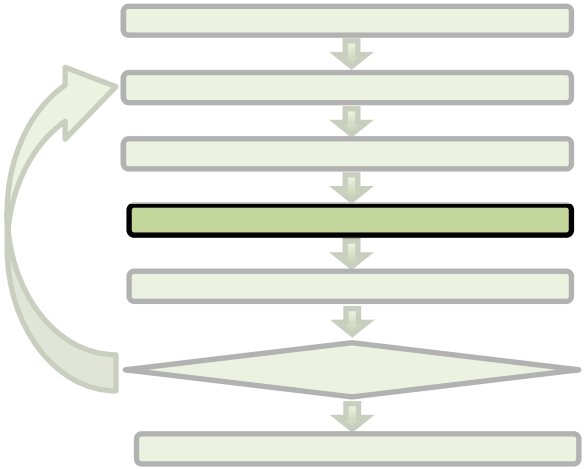
global synchronization

not previously tagged → tag in scratchpad

previously tagged → remove from auxiliary

Remove duplicates



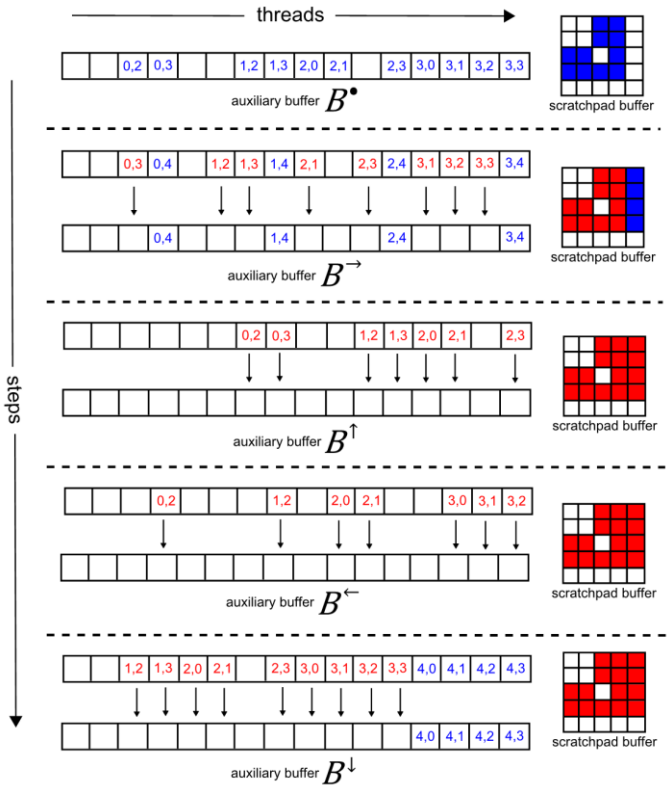


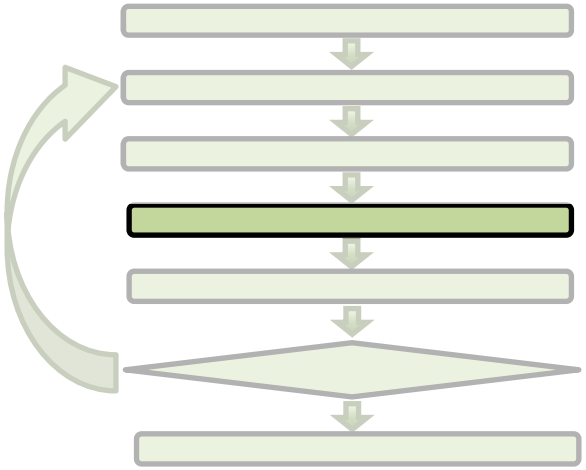
global synchronization

not previously tagged → tag in scratchpad

previously tagged → remove from auxiliary

Remove duplicates



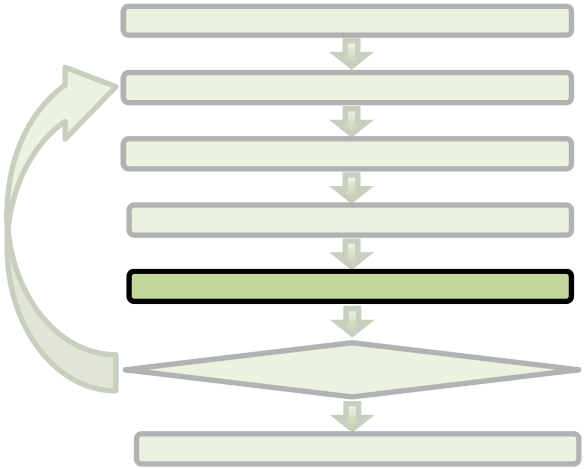


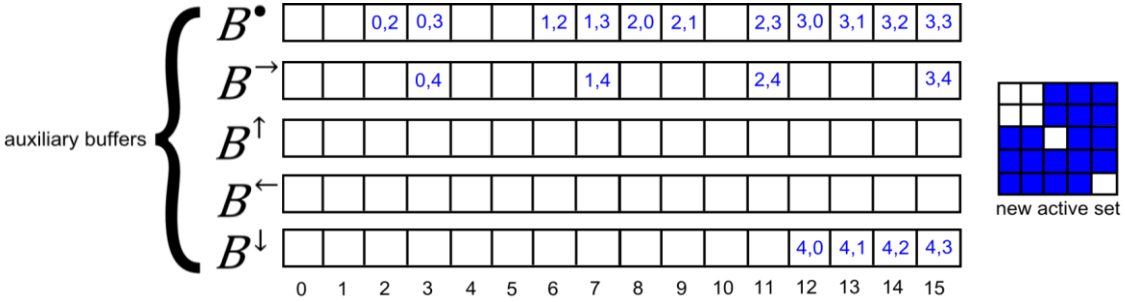
global synchronization

not previously tagged → tag in scratchpad

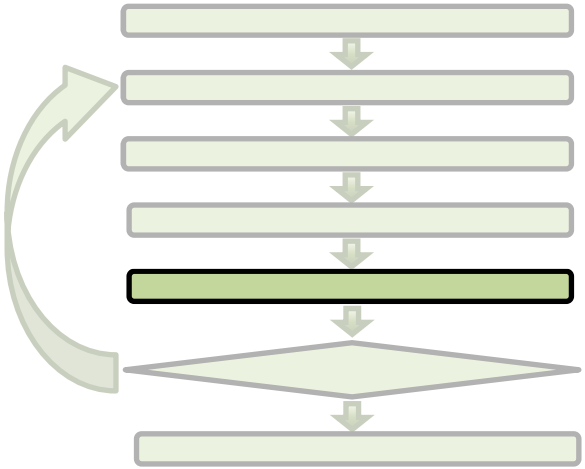
previously tagged → remove from auxiliary

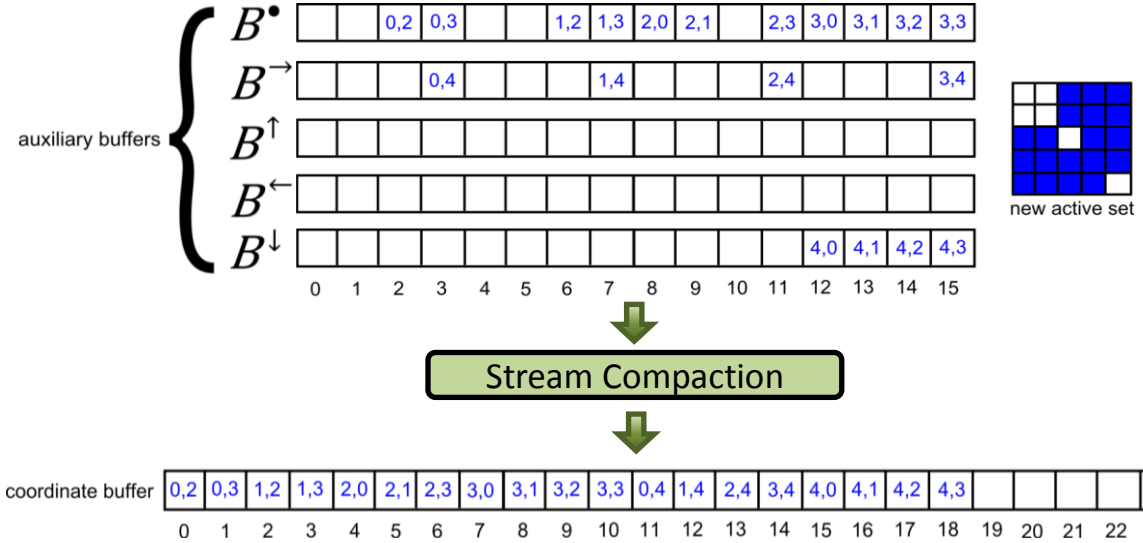
Compact the auxiliary buffers



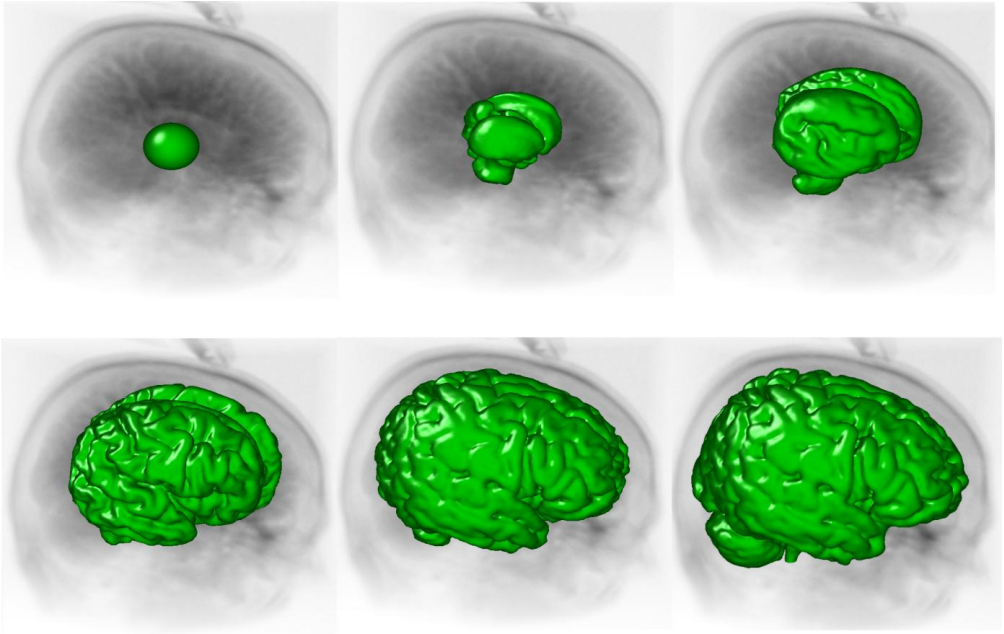


Compact the auxiliary buffers





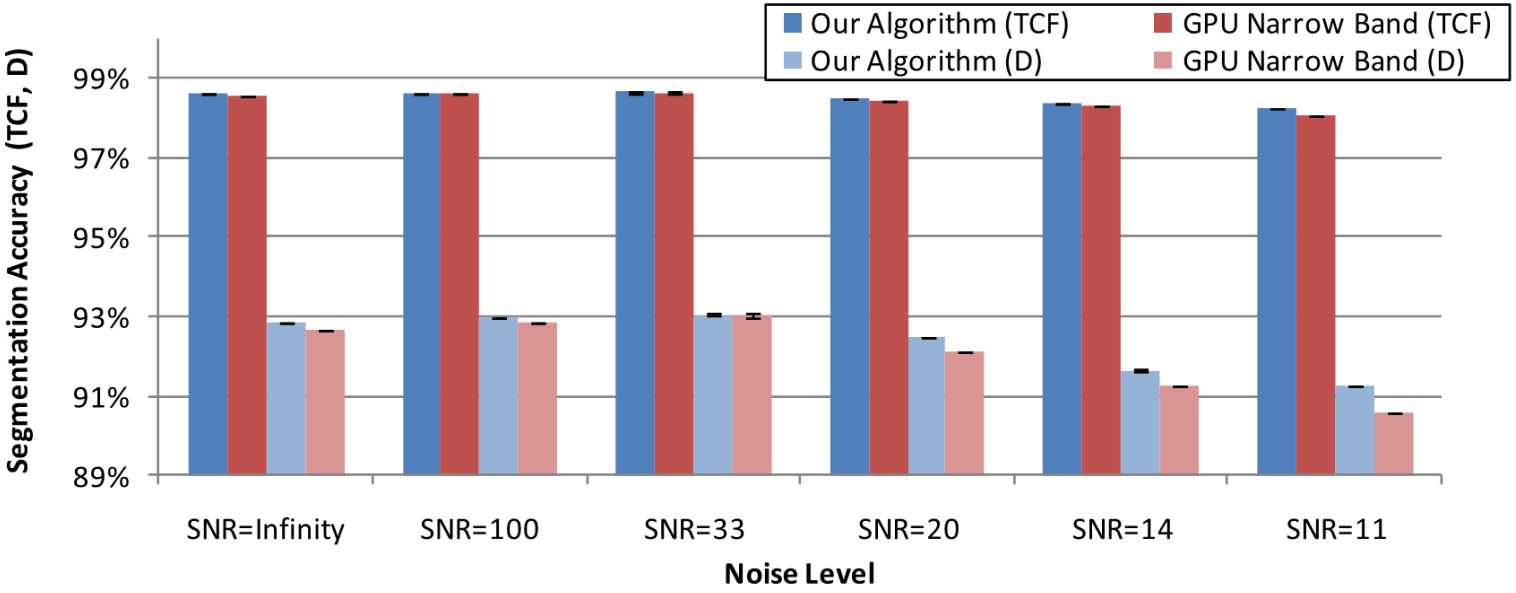
Experimental Methodology



256^3 head MRI

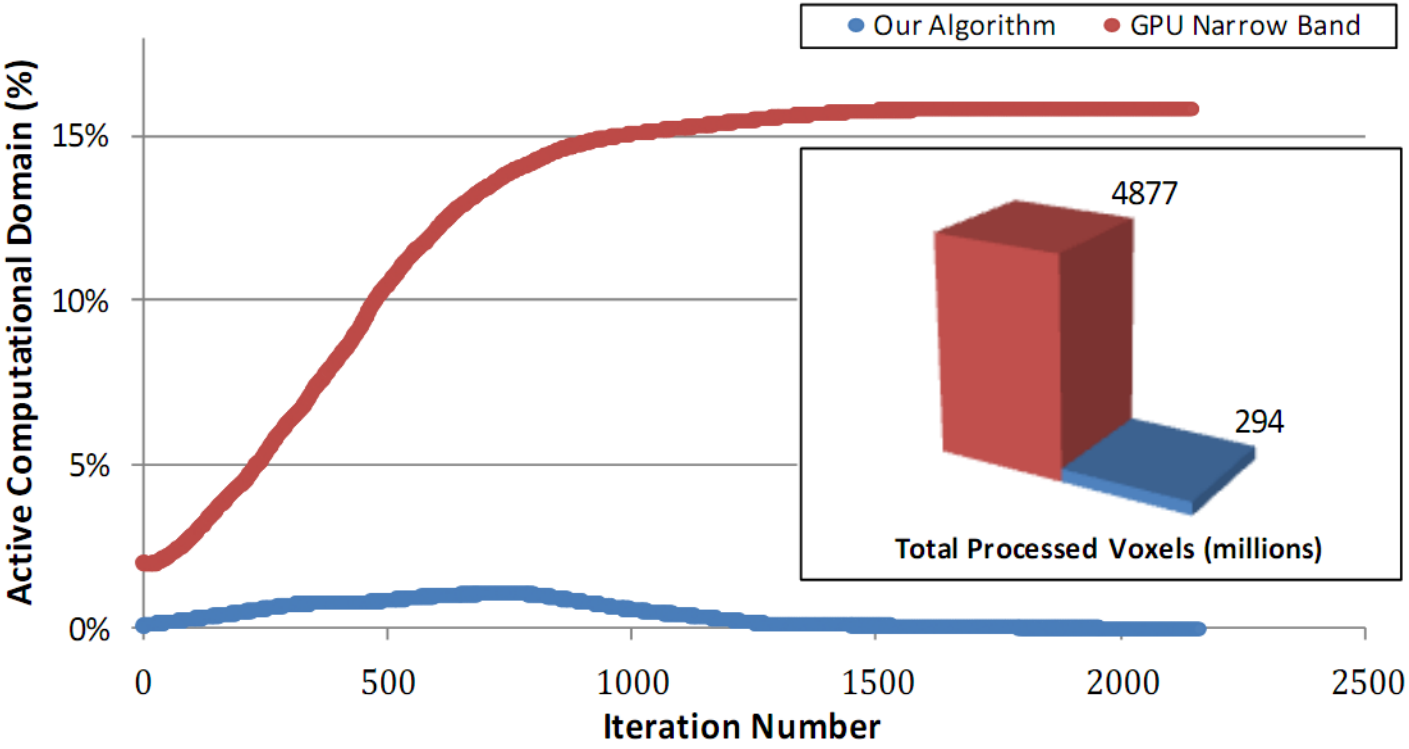
we repeated this segmentation with our algorithm and the GPU Narrow Band algorithm

Accuracy

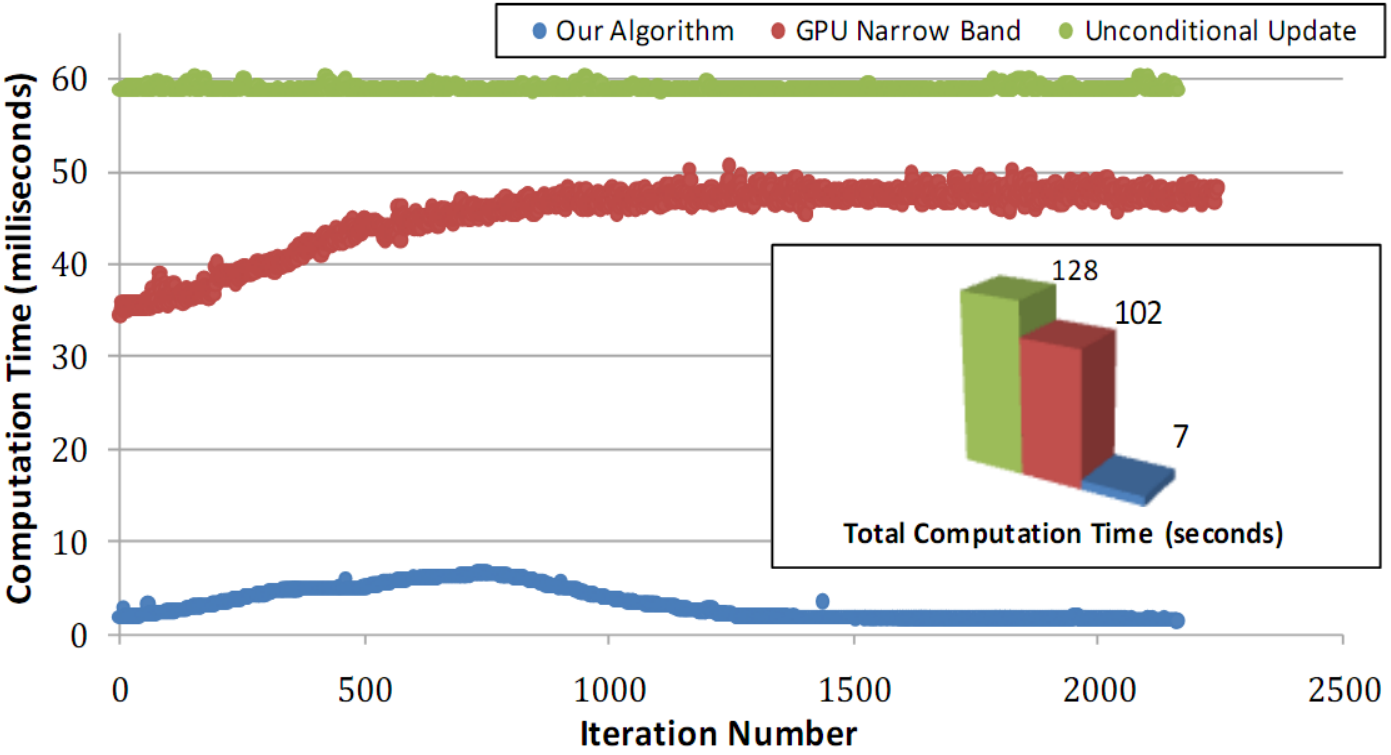


SNR = Signal-to-noise Ratio
D = Dice Coefficient
TCF = Total Correct Fraction of Labeled Voxels

GPU Work


Our Algorithm
GPU Narrow Band
Active Computational Domain (%)
15%
10%
5%
0%
0
500
1000
1500
2000
2500
Iteration Number
4877
294
Total Processed Voxels (millions)

Computation Time



Goals: fast, interactive, accurate

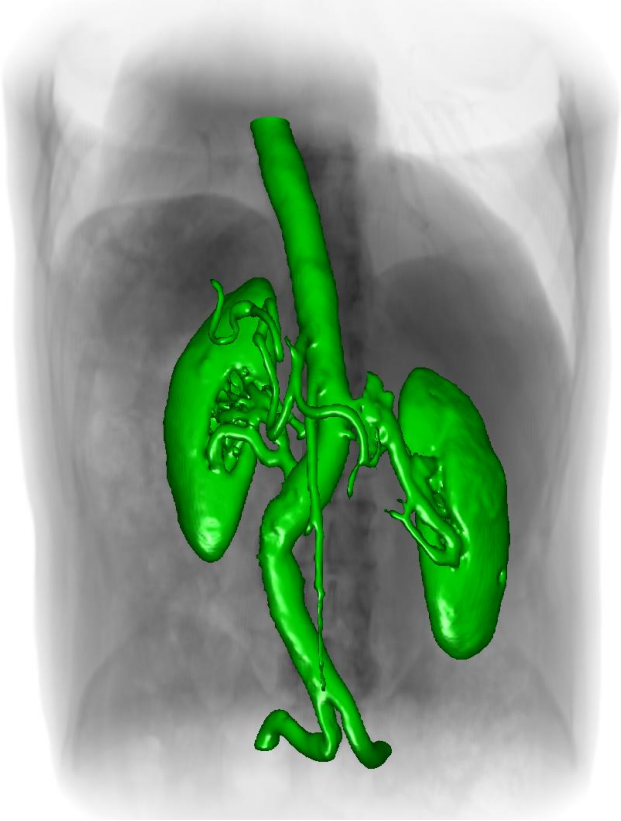
✓

Goals: fast, interactive, accurate

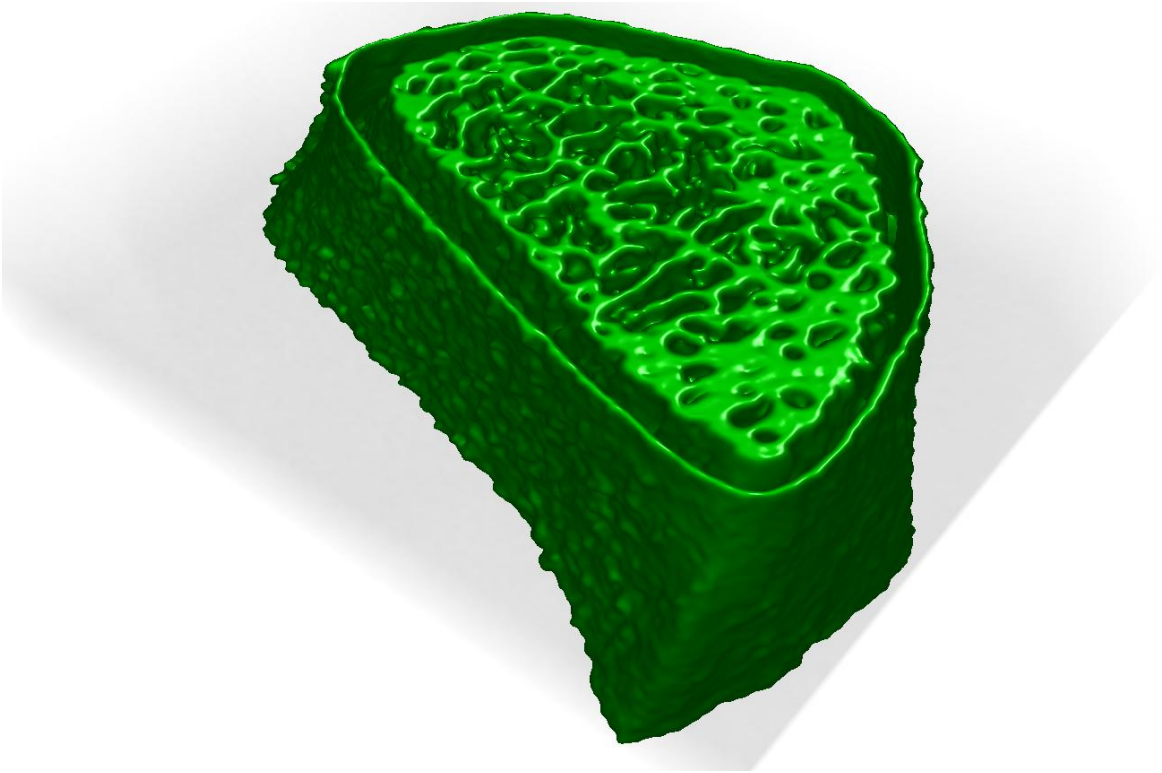
Goals: fast, interactive, accurate

Goals: fast, interactive, accurate

Segmentation Results



Segmentation Results



More Information

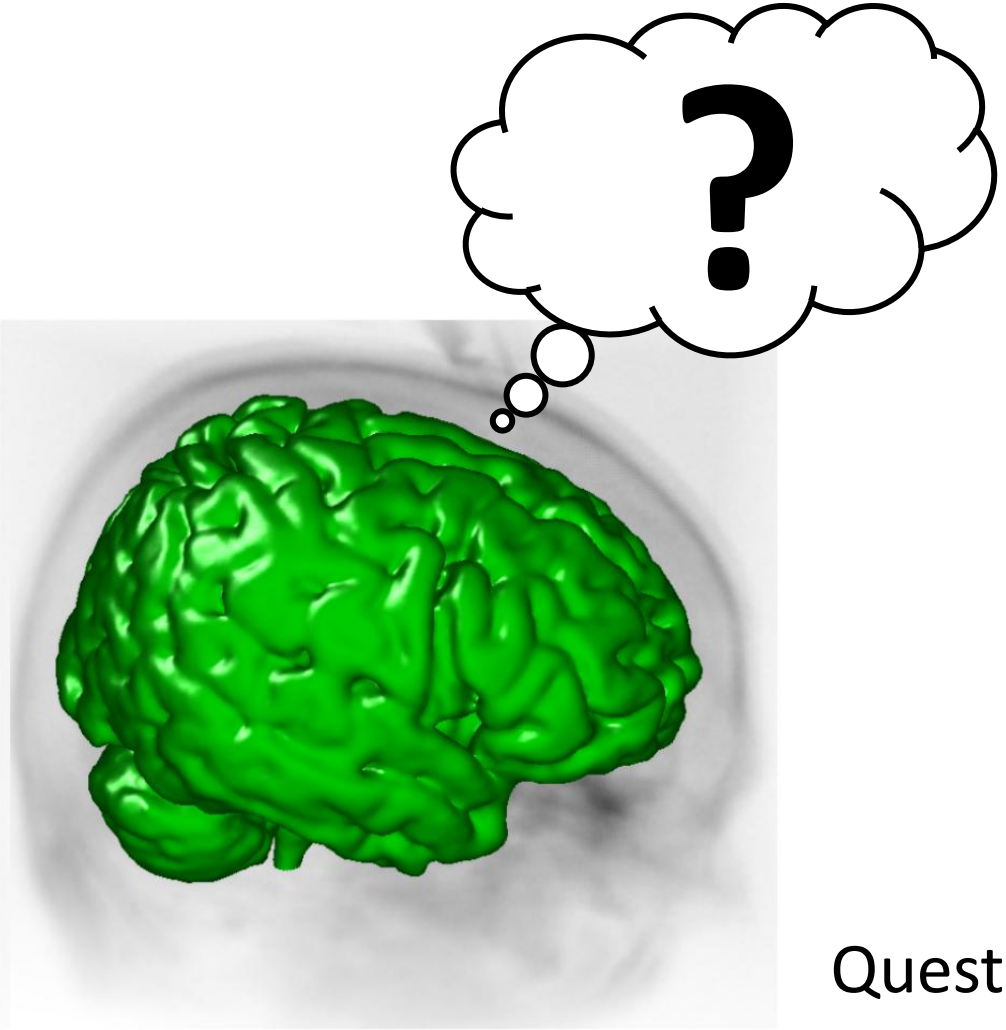
“A Work-Efficient GPU Algorithm for Level Set Segmentation”

Thursday 9am, Room B

“Three-Dimensional Medical Image Visualization Techniques on Modern Graphics Processors”
Eric Penner
University of Calgary, M. Sc. Thesis, 2009

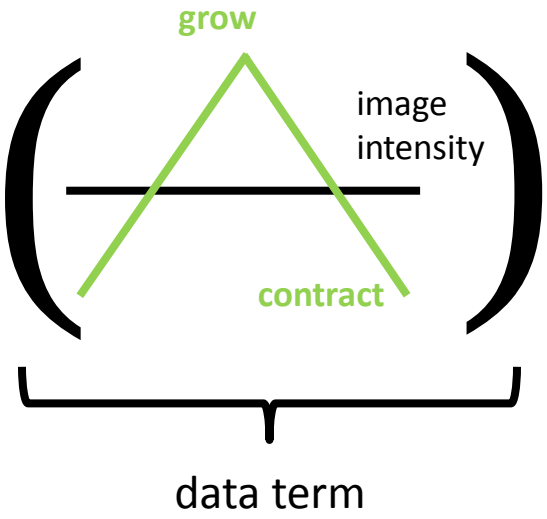
“Isosurface Ambient Occlusion and Soft Shadows with Filterable Occlusion Maps”
Eric Penner, Joseph Ross Mitchell
Proceedings of the IEEE/Eurographics Symposium on Volume and Point-Based Graphics 2008

“A Work-Efficient GPU Algorithm for Level Set Segmentation”
Mike Roberts, Jeff Packer, Mario Costa Sousa, Joseph Ross Mitchell
Proceedings of the ACM SIGGRAPH/Eurographics Conference on High Performance Graphics 2010

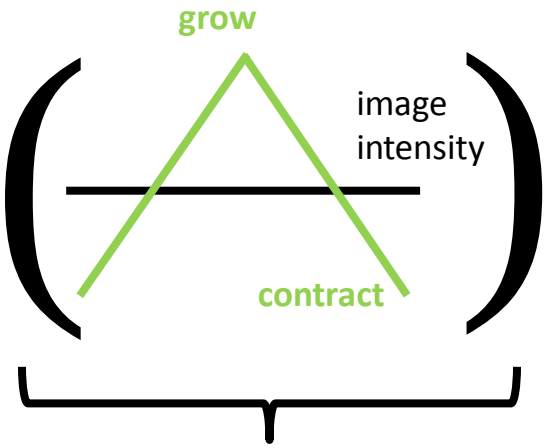


Questions?

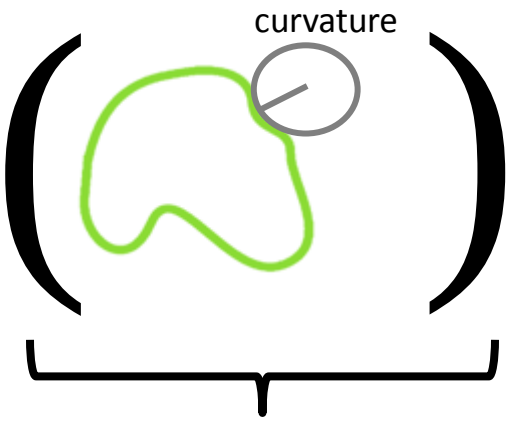
Speed Function [Lefohn et al. 2003]



Speed Function [Lefohn et al. 2003]

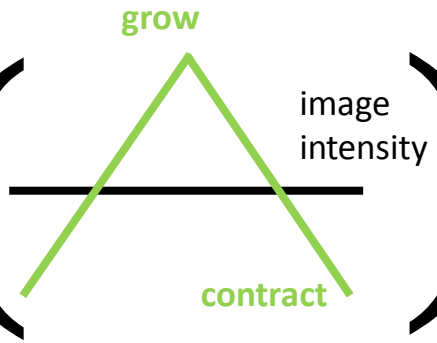


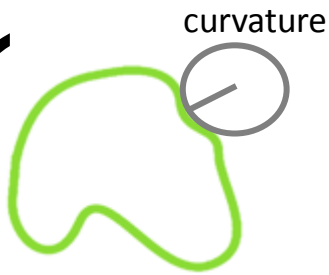
data term



curvature term

Speed Function [Lefohn et al. 2003]

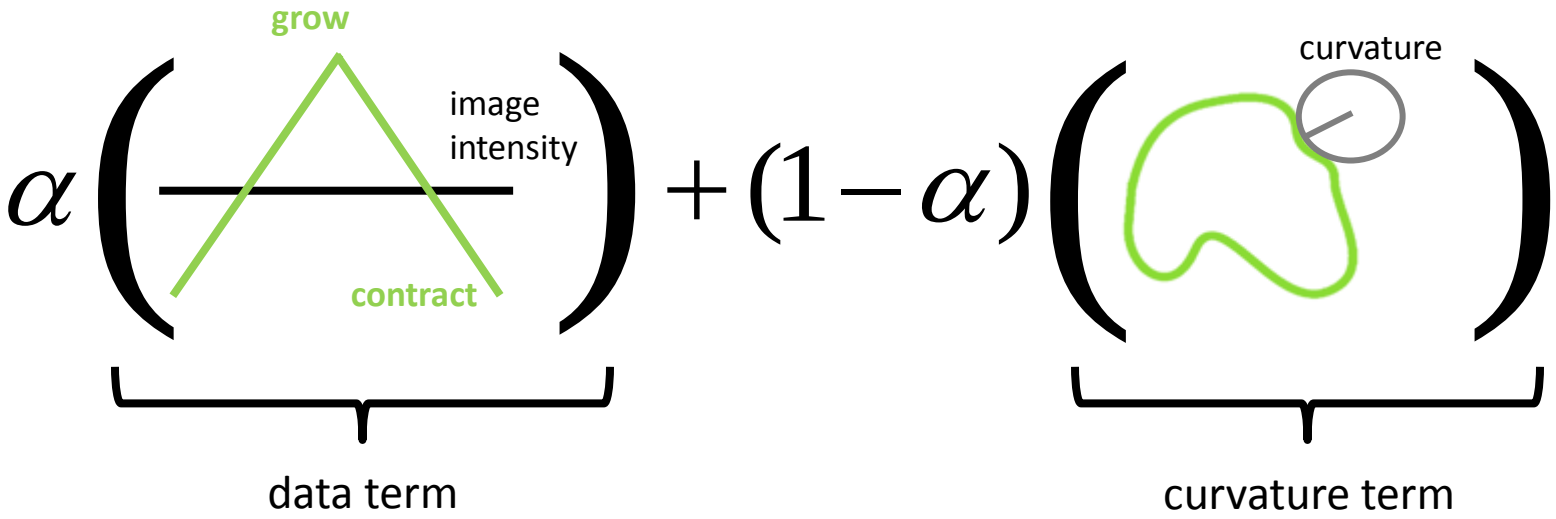
α 

$+(1-\alpha)$ 

data term

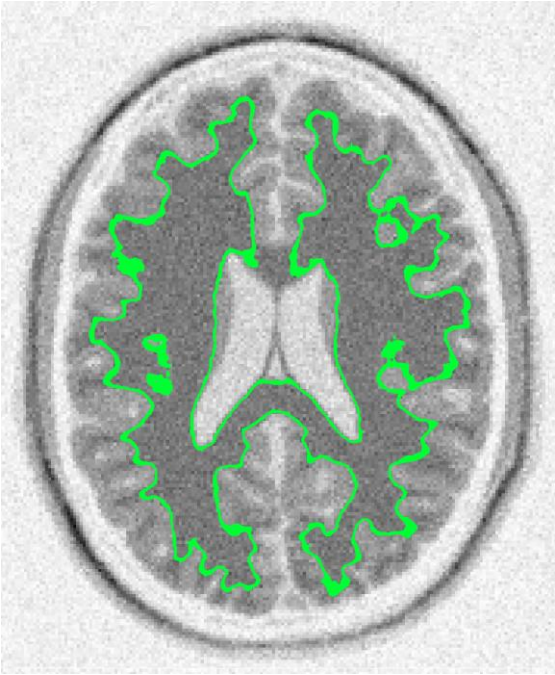
curvature term

Speed Function [Lefohn et al. 2003]



$$\alpha \approx \text{smoothness}$$

Speed Function [Lefohn et al. 2003]



$\alpha = 0.4$

$\alpha = 0.0$