Massively parallel Monte Carlo simulation with graphics processing units (GPU)

Qianqian Fang, Ph.D.
Martinos Center for Biomedical Imaging
Massachusetts General Hospital
Harvard Medical School

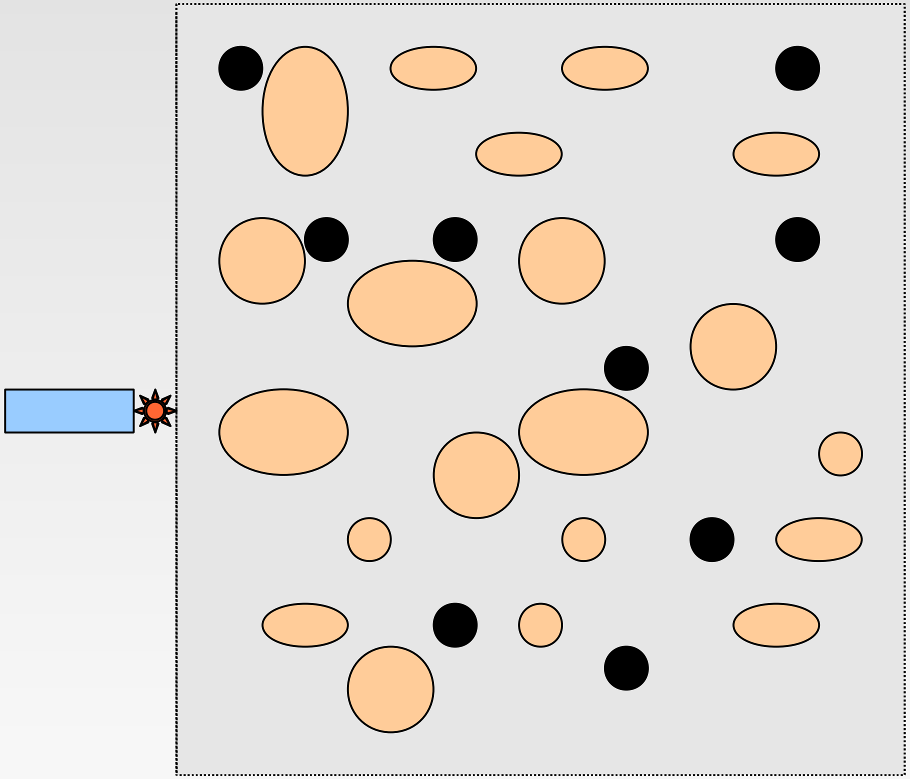
Software website: http://mcx.sf.net

Outline

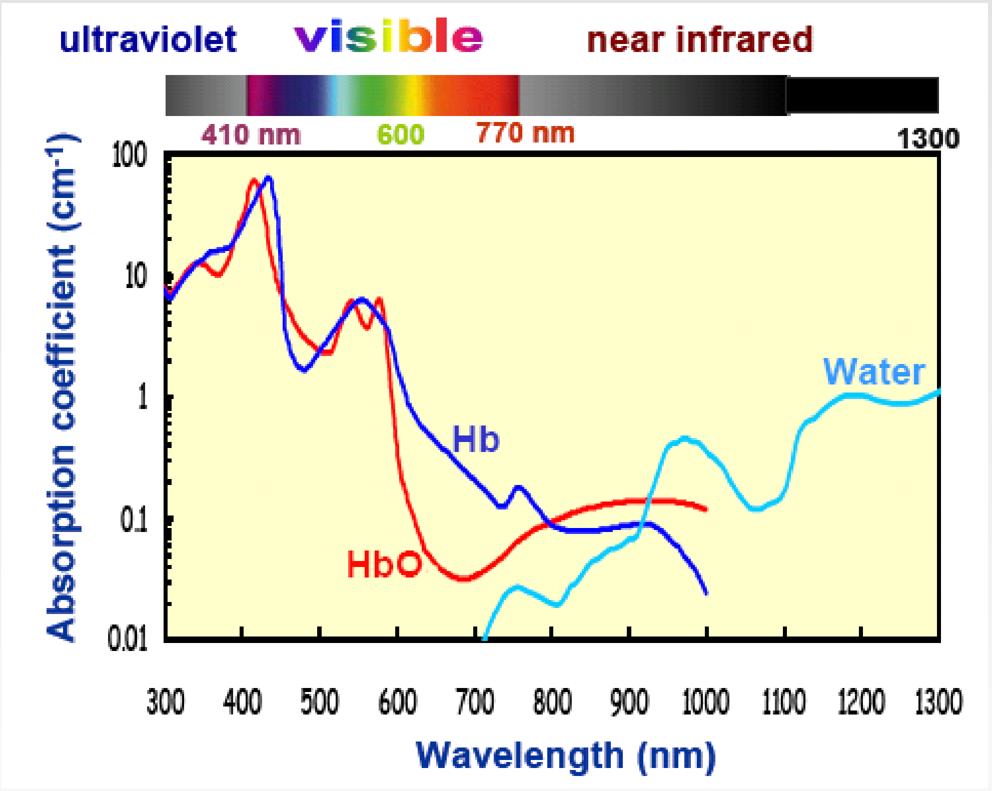
- Introduction
  - Diffuse optical imaging and applications
  - Monte Carlo simulations for photon migration
  - GPU-based parallel computing
- MCX: a GPU-based MC photon migration simulation software
  - Algorithm flowchart
  - Random number generators
  - Boundary reflections
  - Atomic vs. non-atomic operations
  - Validations and comparisons
- What's next?

Diffuse optical imaging

- Tissue chromophores: HbO, HbR, water, lipids



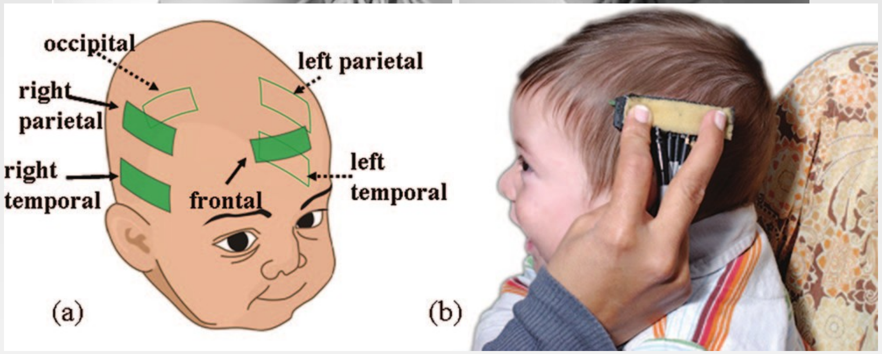
Turbid media

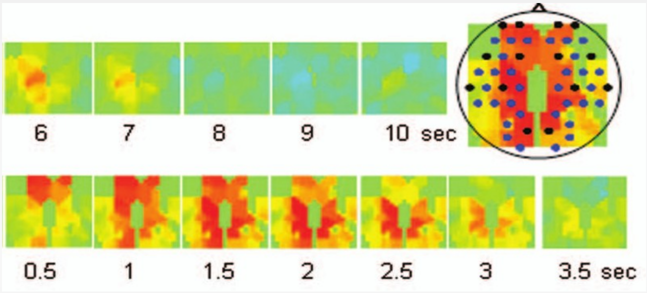


Near-infrared spectroscopy for the study of biological tissue
Angelo Sassaroli, et al. Tufts Univ.

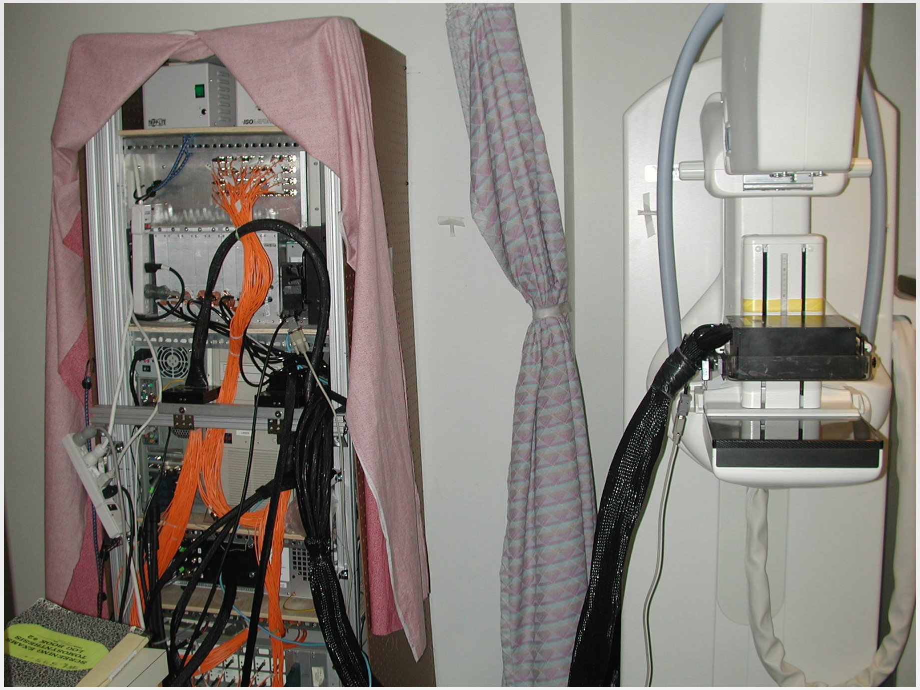
DOI related studies at PMI Lab

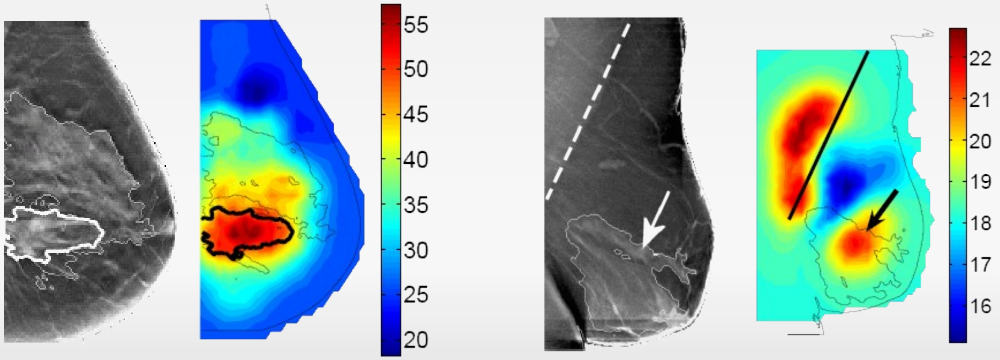
- Brain functional imaging





- Breast imaging



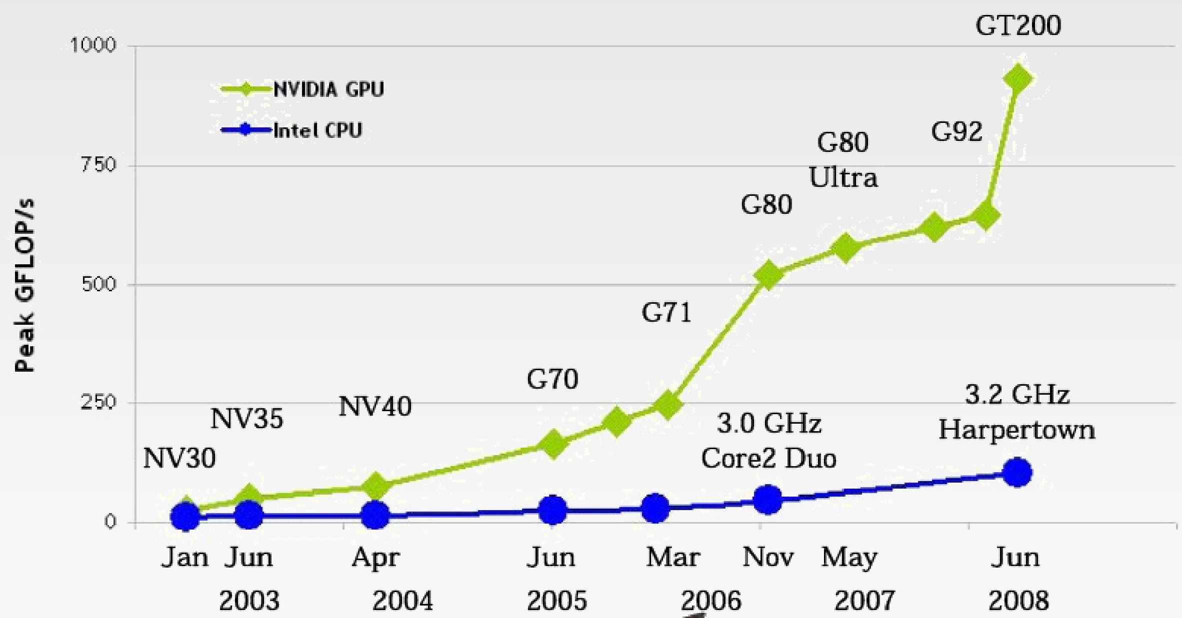


Modeling photon migration

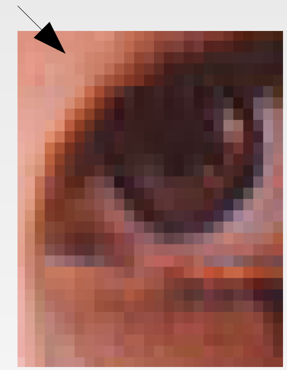
- Transport equation → Finite element or Monte Carlo
- Diffusion → Finite element, finite difference …
- Software tools:
  - Finite element: NIRFAST (Dartmouth), TOAST (UCL), Redbird II (MGH)
  - Monte Carlo method:
    - MCML: Wang & Jacques (1995), layered media
    - tMCimg: Boas (2002), 3D complex media
    - CUDAMCML: Alerstam (2009), layered media+GPU

Multi-core processors and GPU

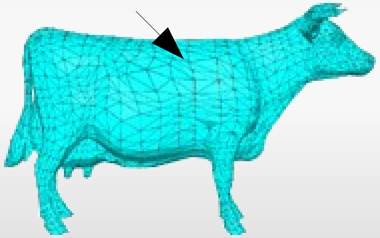
- Multi-core is becoming the mainstream
- Graphics hardware have many cores and optimized for processing data-parallel tasks



Pixel shaders



Vertex shaders



Nvidia and ATI GPU parameters

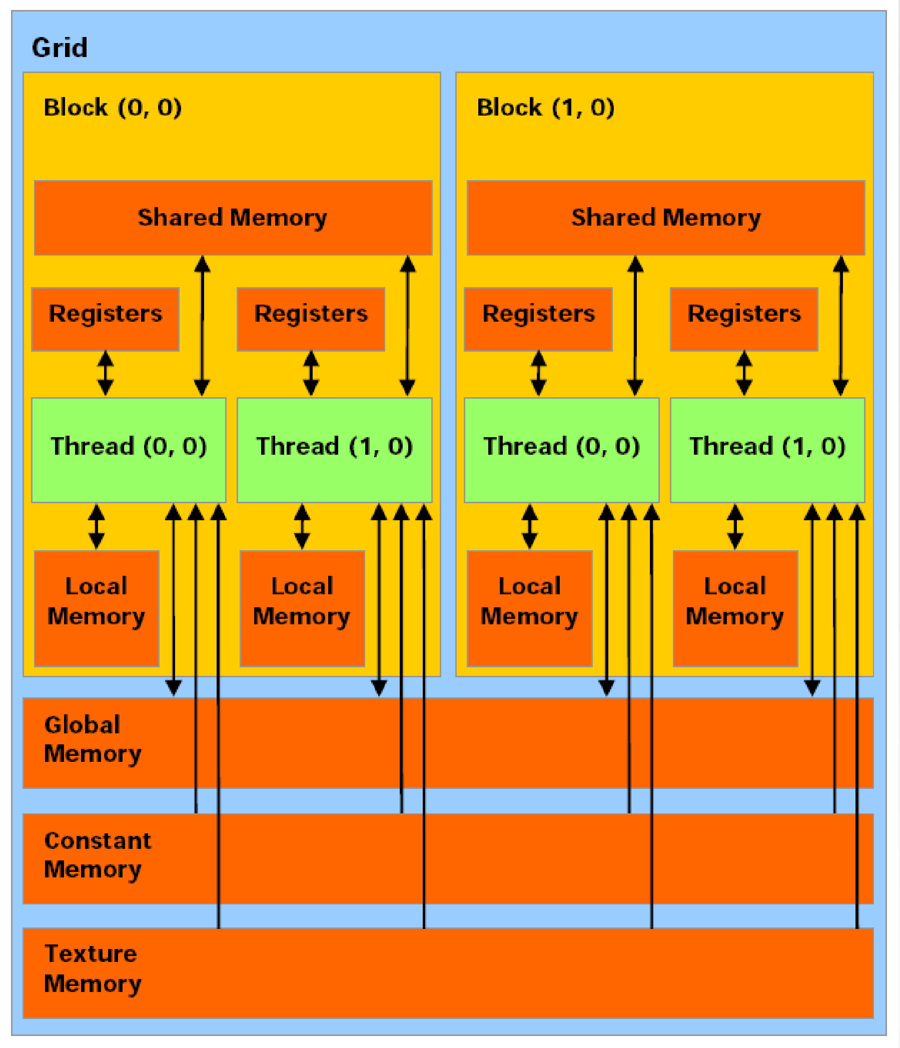
Nvidia GPU	Cores	Core clock	Memory clock	Support
6800GTU (2004)	16	400	1100	dx9c
7800GTX (2005)	24	650	1600	dx9c
8800GTX (2006)	128	575	1800	dx10
9800GTX (2008)	128	675	2200	dx10
GTS150 (2009)	128	740	1000	dx10
GTX260 (2008)	192	576	1998	dx10
GTX280 (2008)	240	602	2214	dx10
Tesla 1075 (4 GTX280)	240x4	602	1600	dx10

ATI GPU	Cores	Core clock	Memory clock	Support
R X800 XT (2004)	16	500	500	dx9b
R X1950 Pro (2006)	48	575	690	dx9c
R 2900 XT (2006)	320	743	800	dx10
R 3870X2 (2007)	320x2	668	828	dx10
R 4870x2 (2008)	800x2	750	900	dx10.1
R 4890 (2009)	800	850	975	dx10.1

http://en.wikipedia.org/wiki/Nvidia_gpu | Ati_gpu

How is a GPU different from a CPU?

- **Pros:**
  - Many-core processor
  - High speed internal memory
  - Parallel pipeline
  - Highly optimized
- **Cons:**
  - No recursion,no function pointers, no host functions
  - GPU-CPU memory transfer is slow
  - Portability across hardware



NVIDIA CUDA Programming Guide

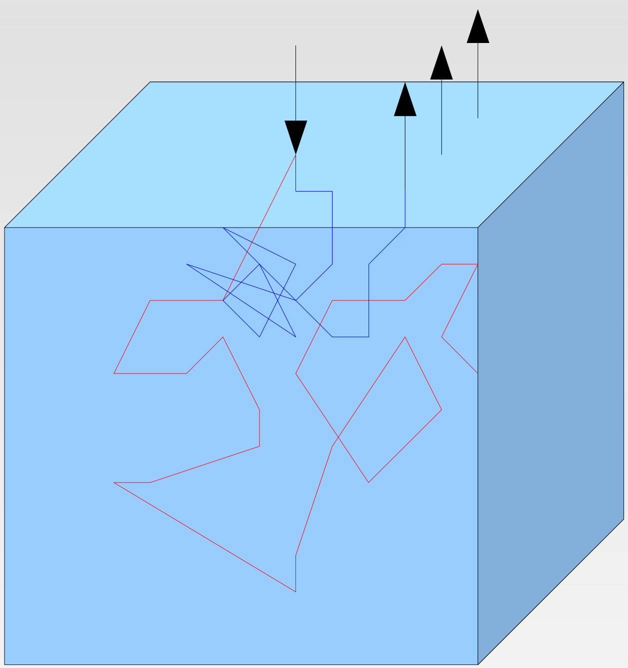
GP-GPU Programming models

- Shading language (OpenGL and DirectX)
  - Cg (nVidia)
  - GLSL (OpenGL)
  - HLSL (Microsoft)
  - CTM (ATI)
- High level meta-language/libraries
  - BrookGPU
  - CUDA
  - Brook+
  - OpenCL (Apple)
  - Lib-sh
  - Rapid mind
  - Jacket (for matlab)

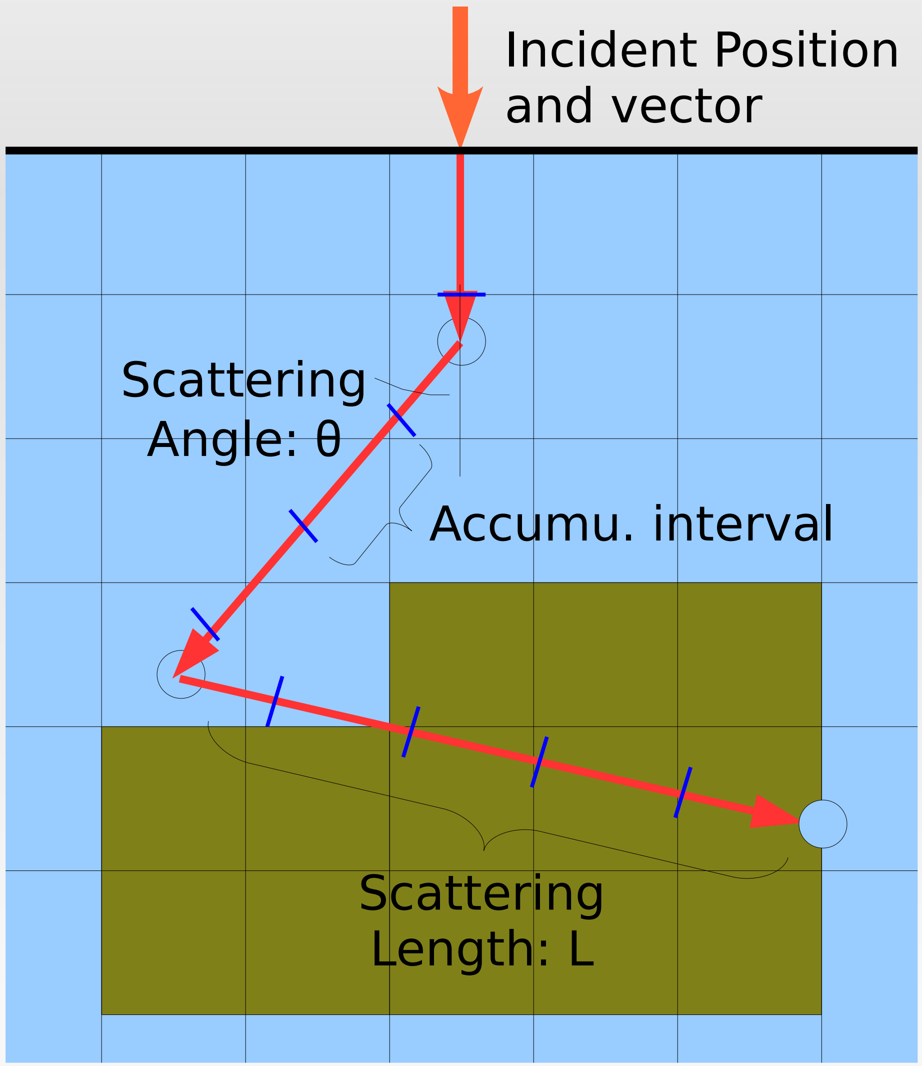
SIMD!

Data-parallel
Task-parallel

Photon migration in a voxelated volume

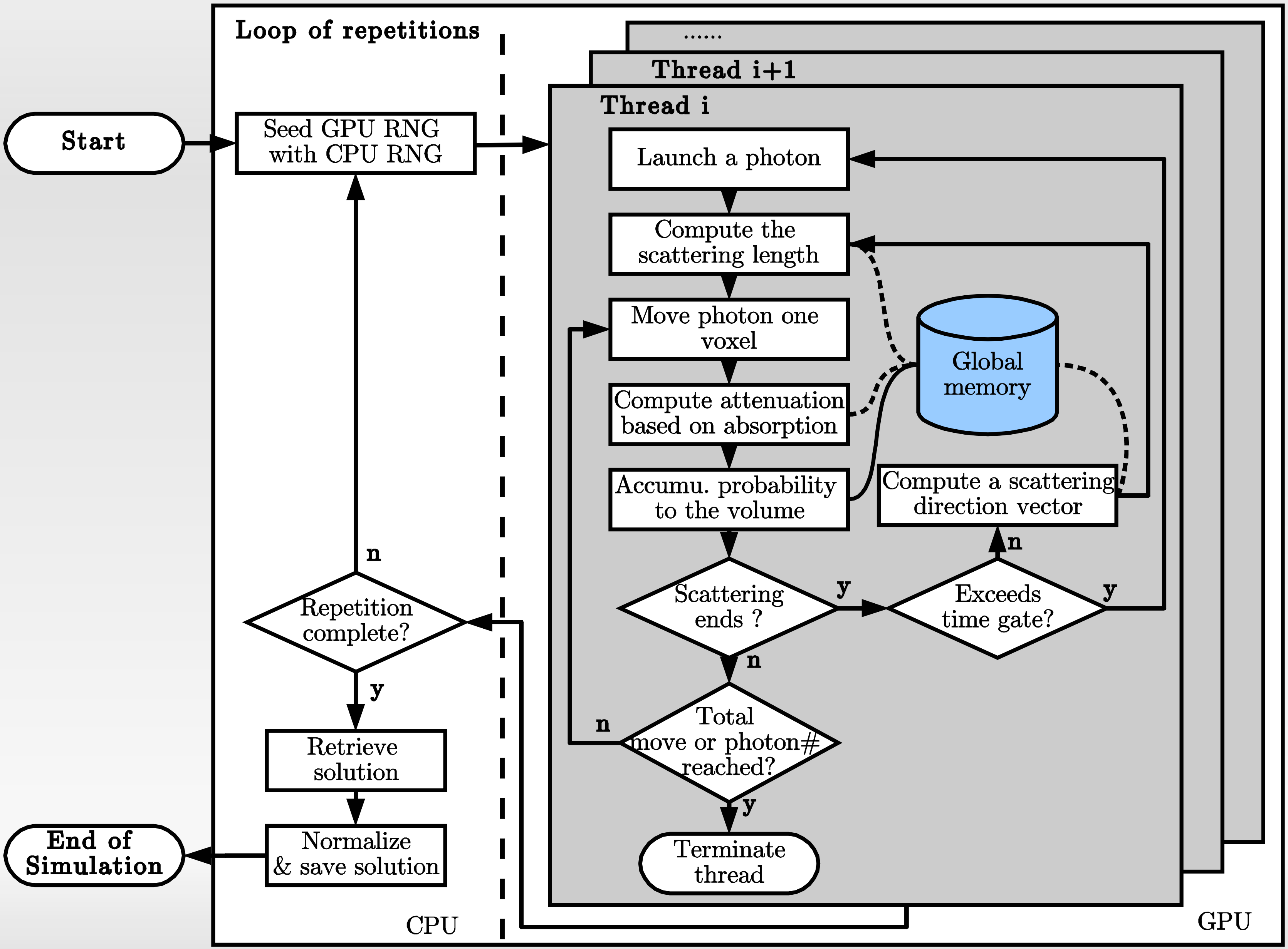


θ = RNG(media[i,j].g) Henyey-Greenstein
L = RNG(media[i,j].μs) Exp. distribution



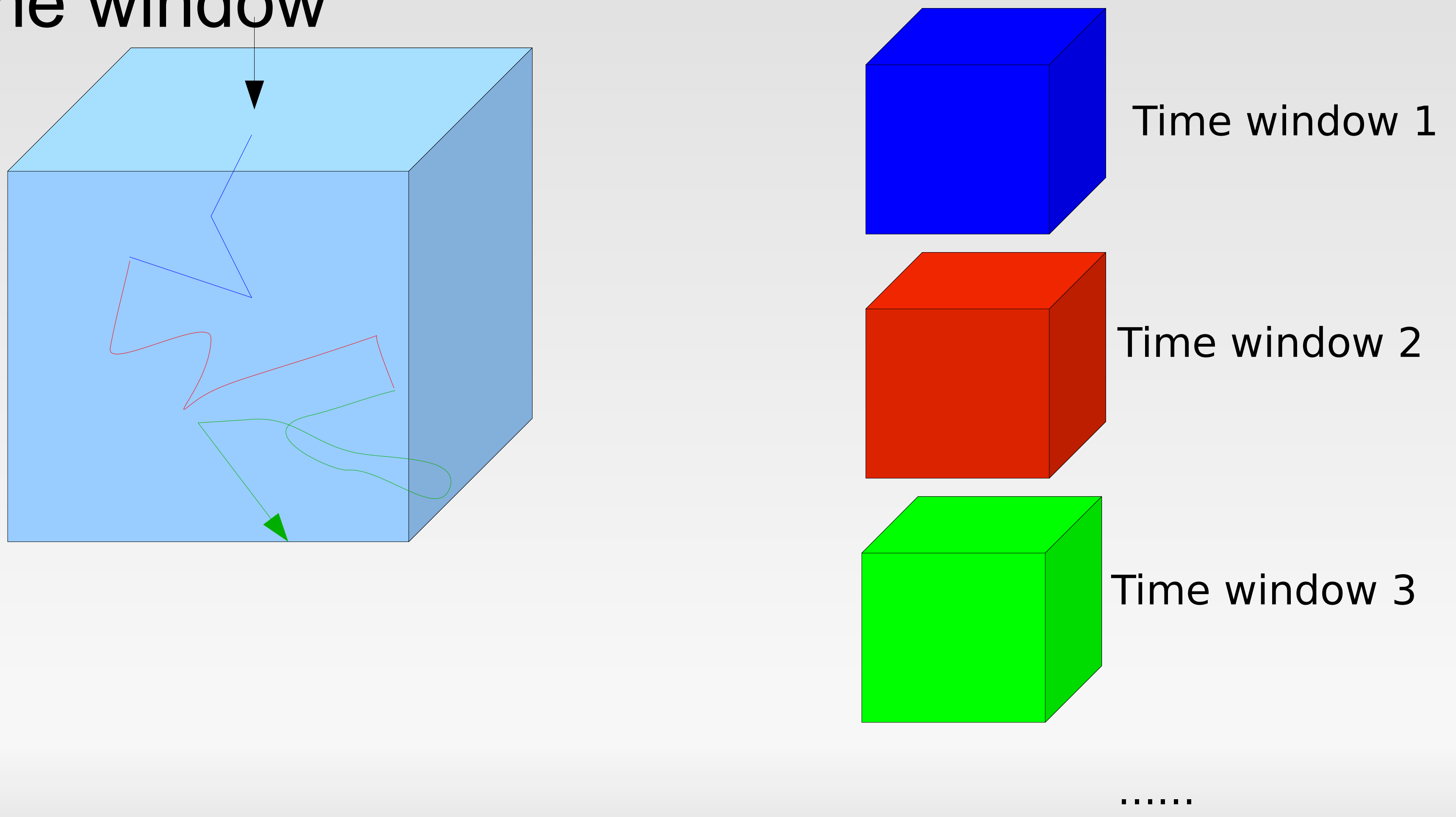
φ = RNG(0~2Pi)
θ = RNG(media[i,j].g)
L = RNG(media[i,j].μs)

GPU-based MC Algorithm Diagram



Recording time-resolved solutions

- Accumulate photon packet to the corresponding time window



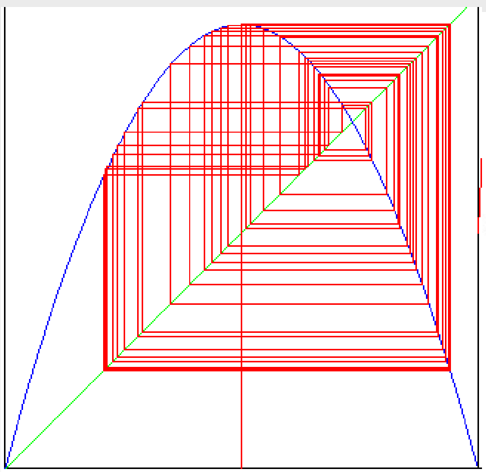
Random number generators

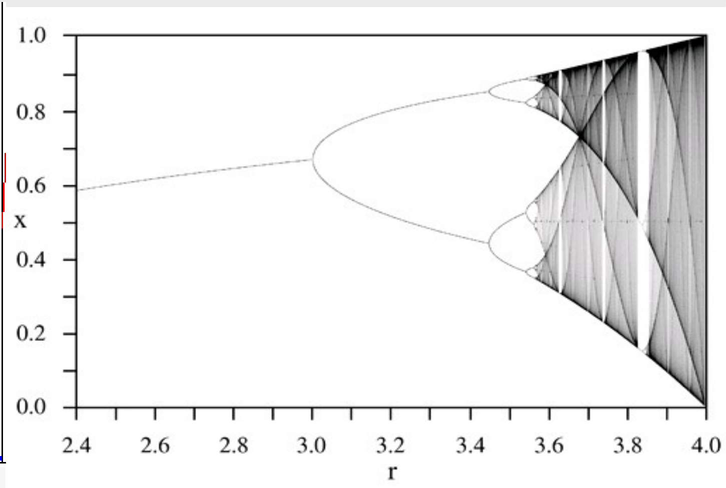
- Mersenne Twister: MT19937 (shared memory)
- Logistic map and lattices: a coupled chaotic system, floating-point only RNG

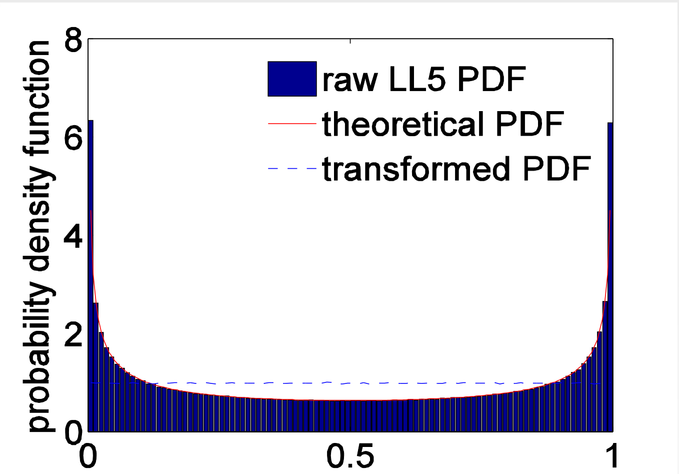
$$p(x) = \frac{1}{\pi\sqrt{x(1-x)}}$$

$$x_{n+1} = r x_n (1 - x_n)$$

When r=4, x is a random-variable with PDF:







Coupled Lattice:

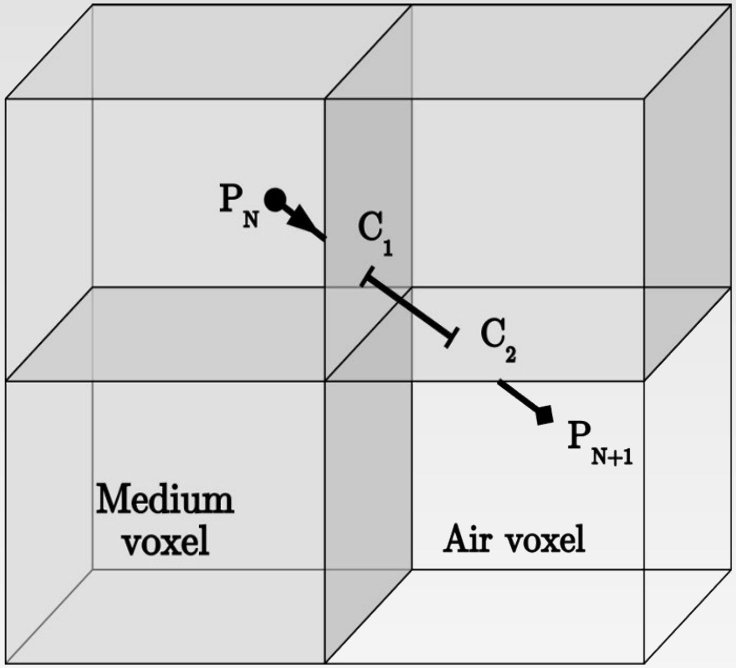
$$x_{n+1}^{i+p \bmod N} = f(x_n^{i \bmod N}) + \nu\left(f(x_n^{i-1 \bmod N}) - 2f(x_n^{i \bmod N}) + f(x_n^{i+1 \bmod N})\right)$$

Wagner, 1995

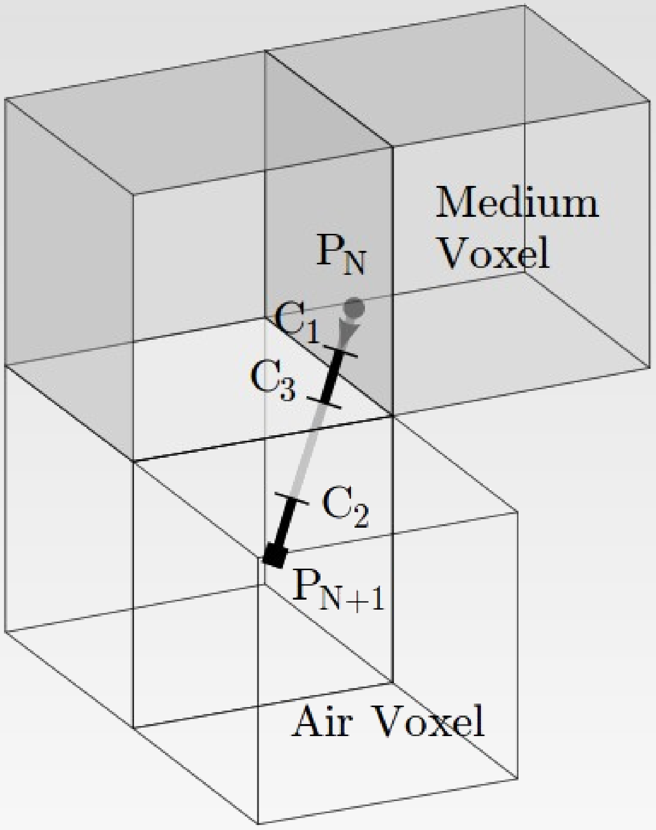
Eric Mills, parallel MT19937 random number generator

Handling of Boundary Reflection

- Case 1



- Case 2



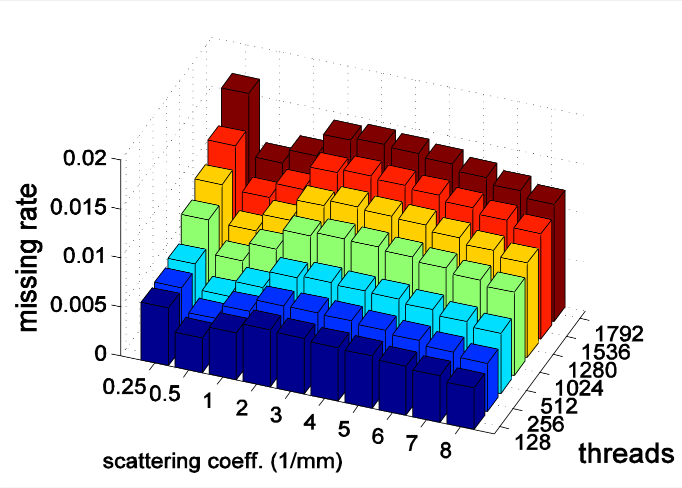
Detecting the exit point:

1. one intersection: find C1 from Pn and done
2. two intersections: find C1, if not, find C2 from Pn+1, done
3. three intersections: if not C1 and C2, orientation of C3 is uniquely decided

Atomic vs. non-atomic operations

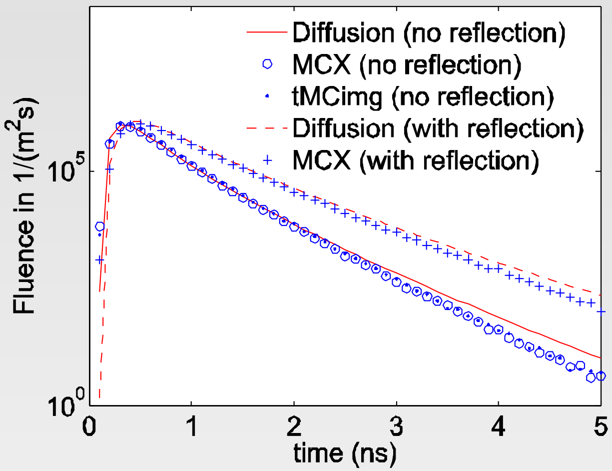
- Non-atomic operations:
  - global val;
  - newval=val+x;
  - val=newvalue;
- Race conditions: when multiple threads read/write a single address, value may become non-consistent.
- How bad is the race condition?

- Atomic operations:
  - atomicAdd(val,x);
- read/add/write in one instruction
- Block other threads to avoid racing

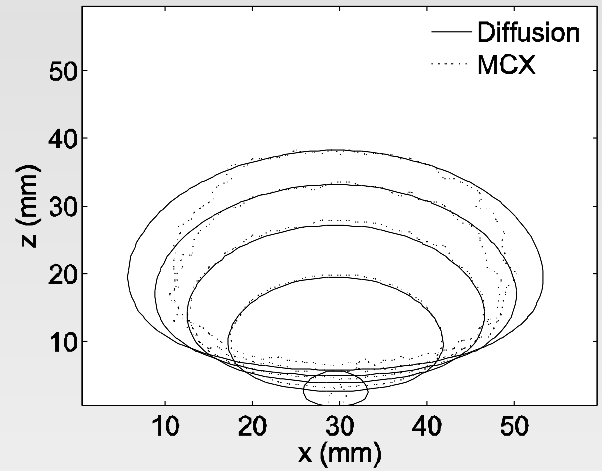


Validation and comparisons

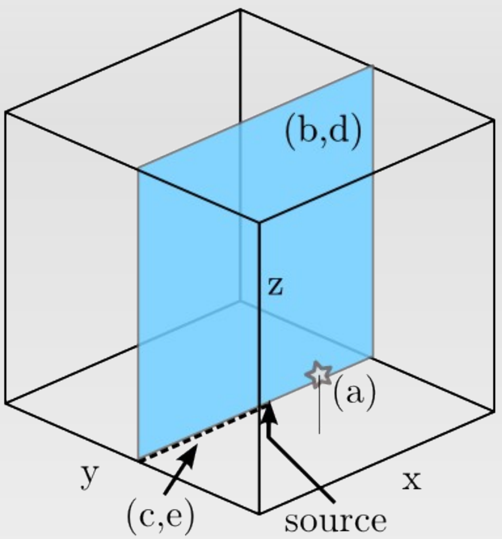
- Semi-infinite medium

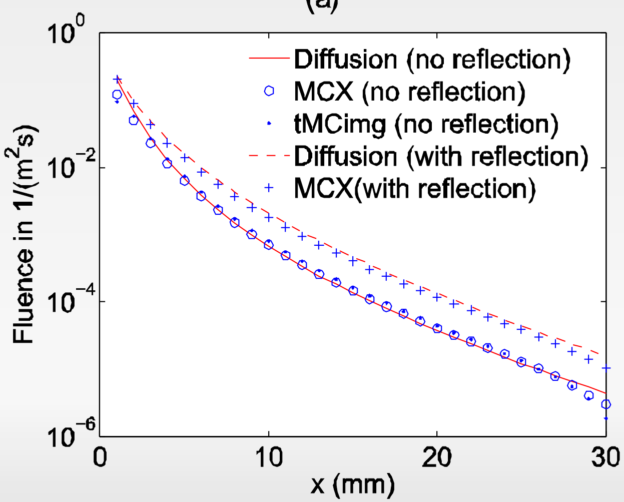


(a)

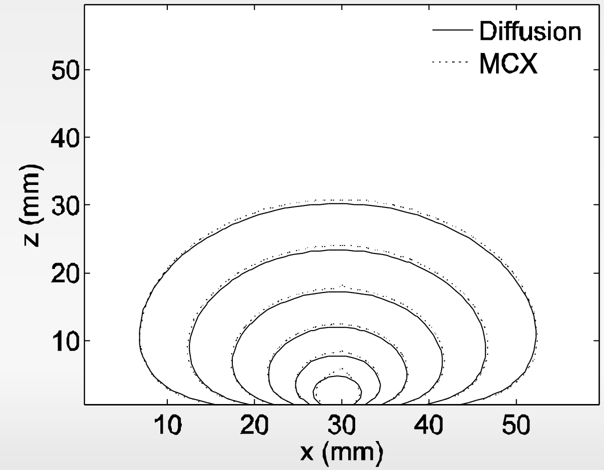


(b)

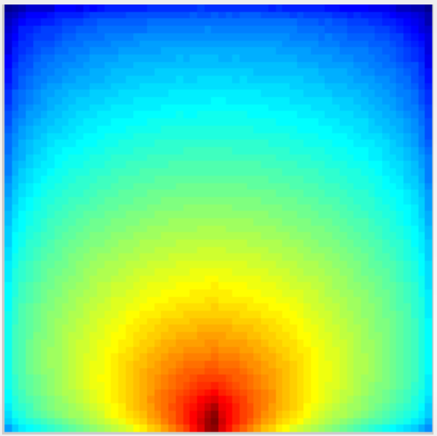




(c)

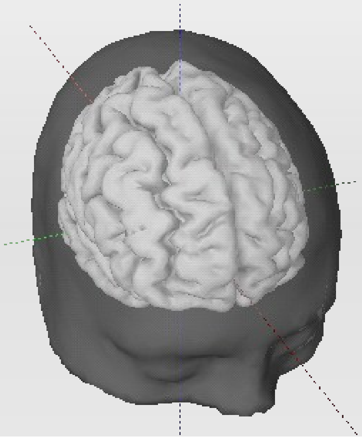


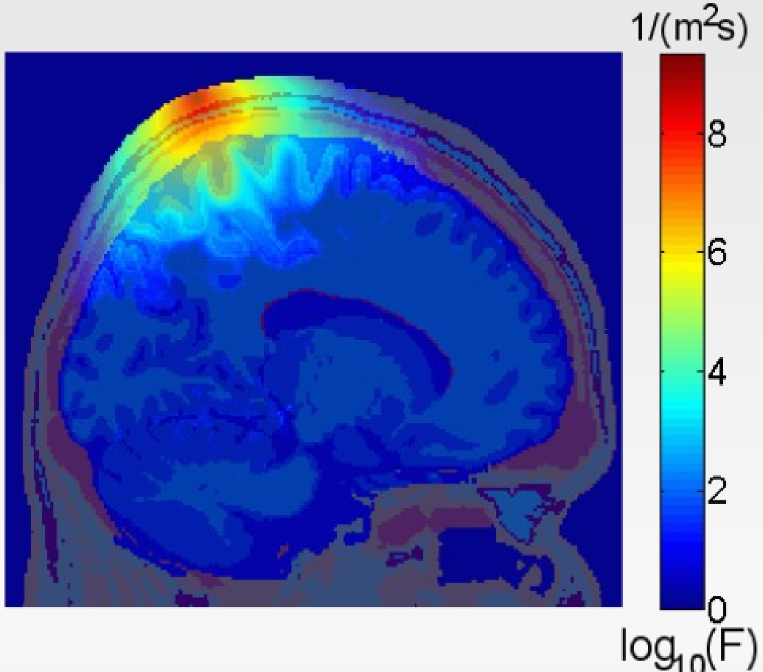
(d)



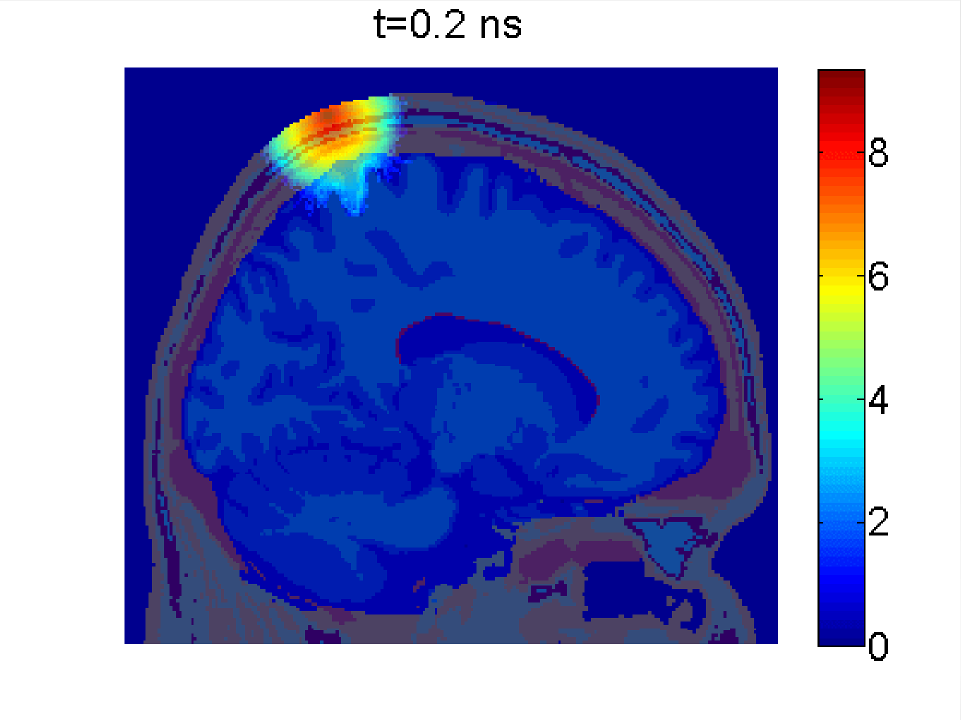
Modeling complex media

- Collins adult brain atlas
- Segmented by FreeSurfer



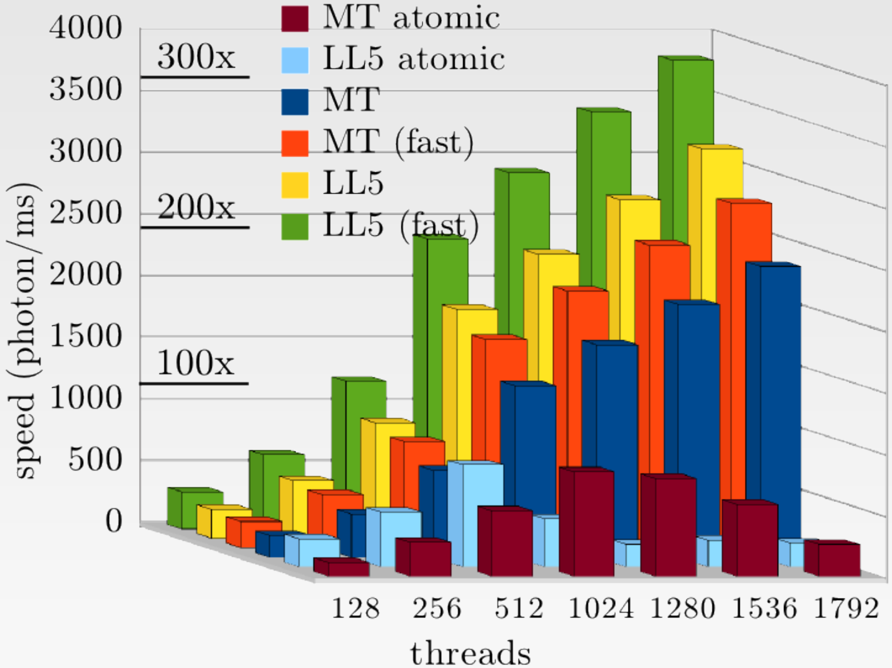


(a)



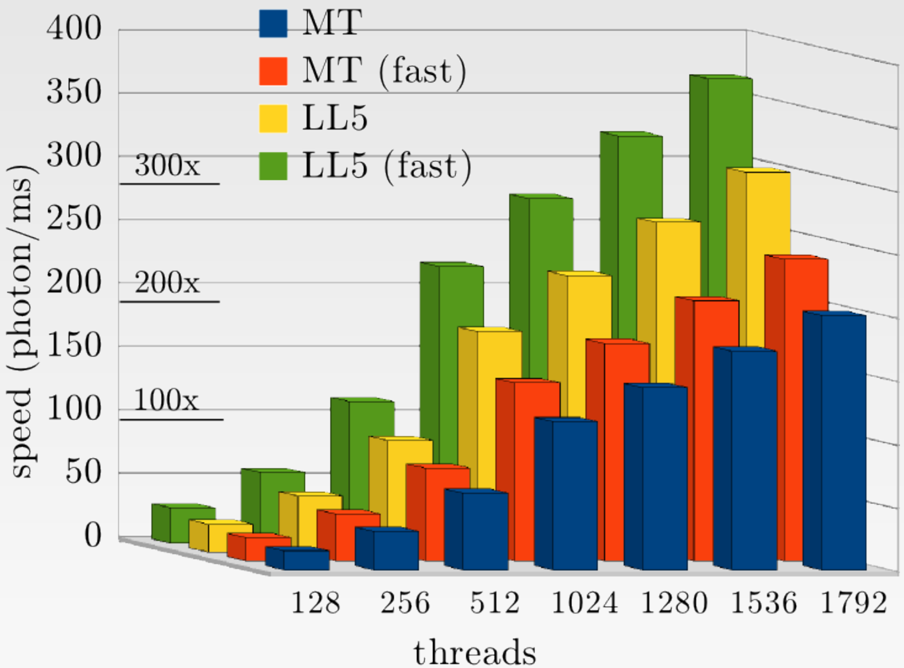
Speed benchmark

- Non-atomic: more threads, more acceleration
- Atomic: peaks around 512~1024 threads, a lot slower



(a)

semi-infinite



(b)

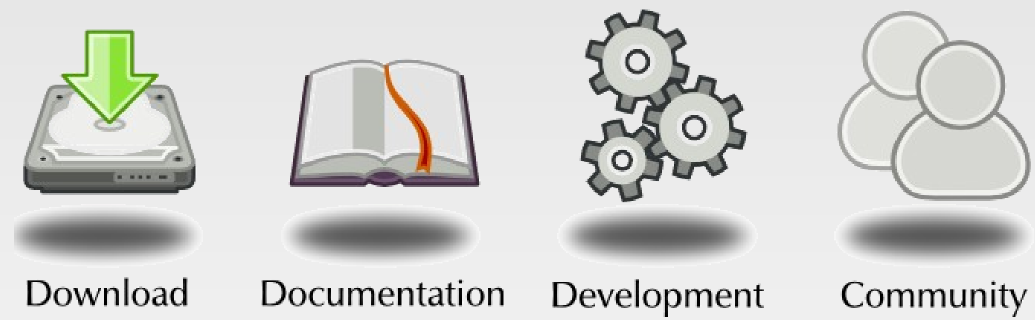
Colin atlas

What did we learn?

- Simulations with non-atomic operations give the best performance, which is scalable with better hardware
- Voxels closed to the source may be slightly (~1%) effected by race conditions; this may become more pronounced with more threads and courser grid
- Atomic operations reach the peak speed around 512~1024 threads, and decrease afterward
- When to use atomic operations?
  Source is located inside a low-scattering medium

Website and resources

- MCX is an open-source software
- Homepage: http://mcx.sf.net
- Binary (windows/linux/mac) and source code for download
- Paper published online from Optics Express



What's next?

- Cross-hardware support: OpenCL and comparisons
- More accurate boundary representation
- Better schemes to avoid non-atomic race condition
- Memory optimization

Photon Migration Lab

Thank you for your attention!

Questions ?