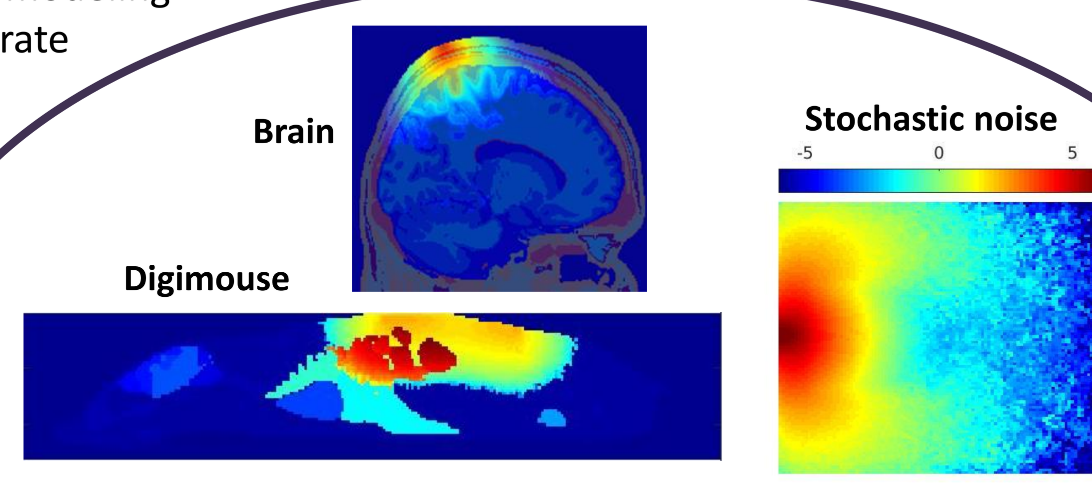
Denoising in Monte Carlo Photon Transport Simulations Using GPU-accelerated Adaptive Non-Local Mean Filter

Yaoshen Yuan[1], Leiming Yu[1] and Qianqian Fang[2]
[1] Department of Electrical and Computer Engineering, and [2] Department of Bioengineering
Northeastern University, Boston, Massachusetts, MA 02115, USA
coti Computational Optics & Translational Imaging Lab
http://fanglab.org

Motivation

1. Monte Carlo (MC) is widely recognized as a gold standard for modeling light propagation inside turbid media. MC method gives accurate solution to Radiative Transport Equation (RTE).
2. Graphic Processing Units (GPU) has been used in MC algorithm to accelerate run-time from hours to seconds.
3. Inherent stochastic noise can be reduced by launching more photons whereas this results in slower computation even with GPU acceleration.

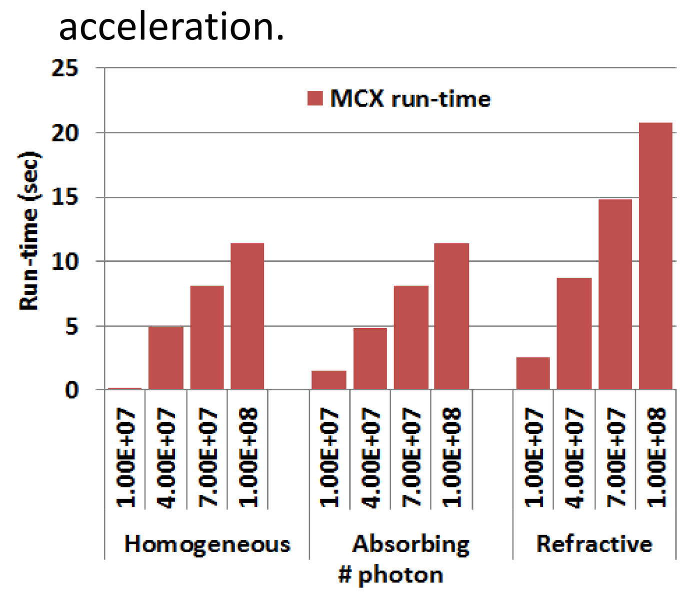


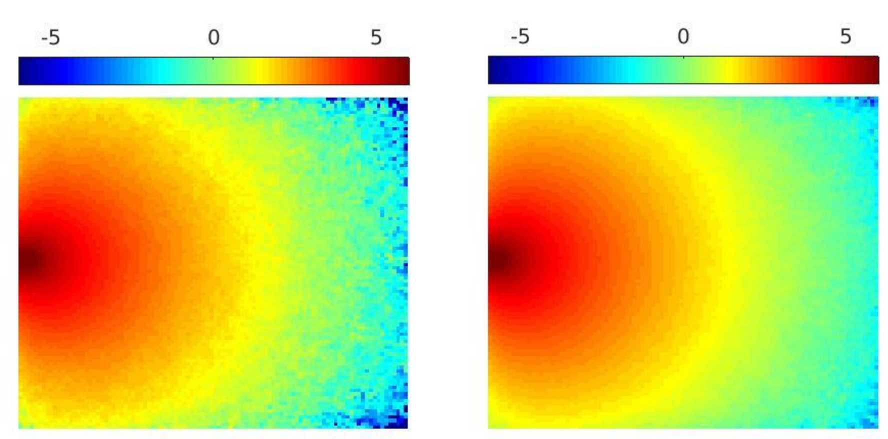
4. An adaptive non-local means (ANLM) can be applied to MC images due to its superior edge-preservation and effective noise reduction.
5. **Purpose: To improve SNR of low-photon MC images using image denoising; equivalently, to accelerate MC simulations.**
6. With our optimized GPU-accelerated ANLM filter, we are able to achieve SNRs comparable to those generated with nearly 10-fold photons. The GPU ARLM filter was also accelerated by a 3x-6x compared to published works [1].
7. Will be released as part of MCX: http://mcx.space/

Introduction

Challenges of Monte Carlo simulation

- Monte Carlo (MC) simulation is more accurate with more photons.
- Run-time becomes longer as the photon number increases even with GPU acceleration.





10^6 vs. 10^7 photons using MC simulation

Noise characteristics in MC images

- MC noise presents as shot-noise
- Decreases with more simulated photons
- Spatially changing due to fewer photons distal to the source

What is Adaptive Non-local Mean (ANLM) filter and why

Pros:
- Superior edge-preservation
- Exceptional noise suppression
- Adaptive to spatial-variant noise
- Parallelizable

Cons:
- Very high time complexity

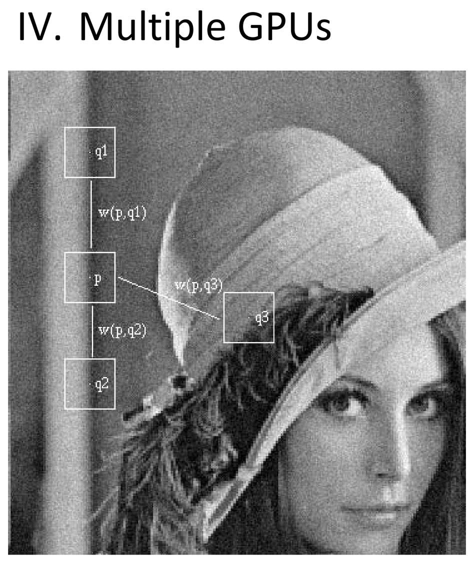
Previous work [2]
I. GPU-accelerated ANLM
II. Filtering on MR images
III. Multi-component (T1, T2 and PD)
IV. Multiple GPUs

The filtered value is [1]:

$$u_i = \sum_{x_j \in V_i} w(x_i, x_j) u_j$$

The weight is computed:

$$w(x_i, x_j) = \frac{1}{Z_i} \exp\left(\frac{\left\| u(D_i) - u(D_j) \right\|_2^2}{(\boldsymbol{h})^2} \right)$$

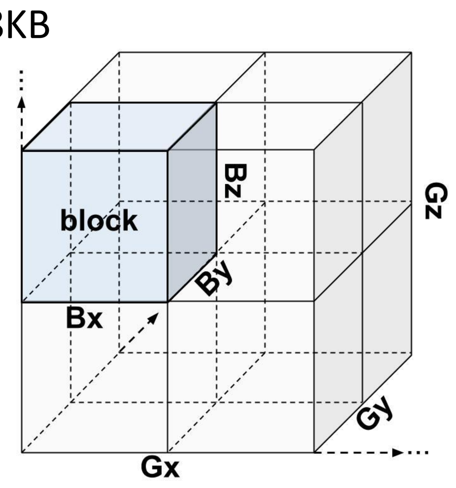


h: adaptive parameter

Our work
I. Support single precision
II. Selection rule for non-local patch
III. Optimized memory config.
IV. 3D search area for computing ***h***

GPU memory configuration and optimization

- **Global memory**: large but slow
- **Texture memory**: read only and beneficial when fetching data from adjacent memory address
- **Shared memory**: faster but only 48KB



Algorithm 1: GPU memory configuration for ANLM

Input: noisy image Ω_i, search radius v, patch radius f
Output: Denoised image Ω_o
1: Load Ω_i and compute $mean$, var and $R = \Omega_i - mean$ for filtering computation.
2: Load Ω_i, $mean$, var and R into texture memory
3: Load Ω_i and R into shared memory. In each block, the size of shared memory is $2 \times (B_x + 2(f+v))(B_y + 2(f+v))(B_z + 2(f+v))$.
4: The denoised image Ω_o is save in the global memory.

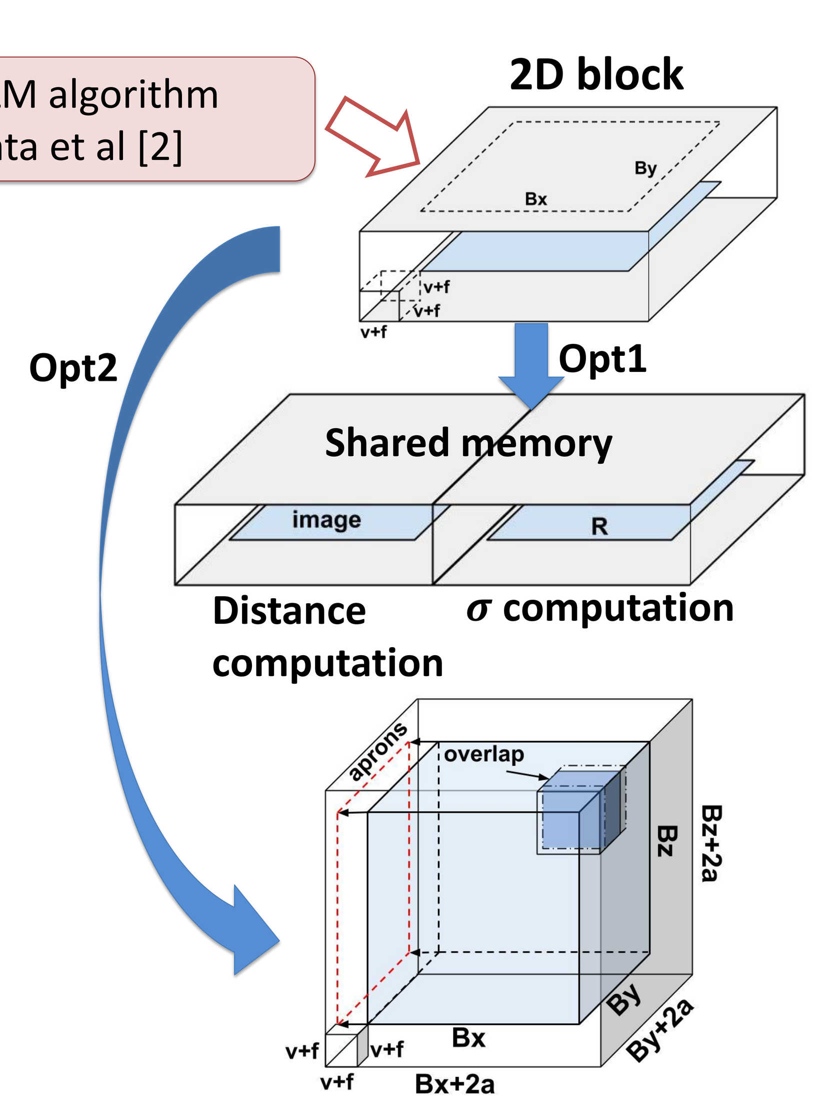
Baseline: GPU ANLM algorithm proposed by Granata et al [2]

Optimization 1: Load both image Ω_i and $R = \Omega_i - mean$ into shared memory.

Optimization 2: Use 3D block instead of 2D block to reduce shared memory usage.

Optimization 3: Pre-compute Ω_i, R and var matrices using GPU.

Optimization 4: Combine two filtering without calling kernel twice.



Experiments

	$\mu_a\,(mm^{-1})$	$\mu_s\,(mm^{-1})$	g	n
homogeneous	0.005	1	0	1.37
absorbing	0.025	1	0	1.37
refractive	0.005	1	0	6.85

Experiment setup
- $100 \times 100 \times 100$ homogeneous cube (1 mm/voxel)
- Pencil beam source is placed on coordinate (50,50,50) with orientation (0,0,1)
- Absorbing and refractive inclusion are added (see table)
- Patch radius for 1st filtering: $f_1 = 1$, Patch radius for 2nd filtering $f_2 = 2$, search radius: $v = 3$.

Hardware
- CPU: Intel® Core™ i7-6700K CPU @ 4.00GHz
- GPU: NVIDIA GeForce GTX 980 Ti

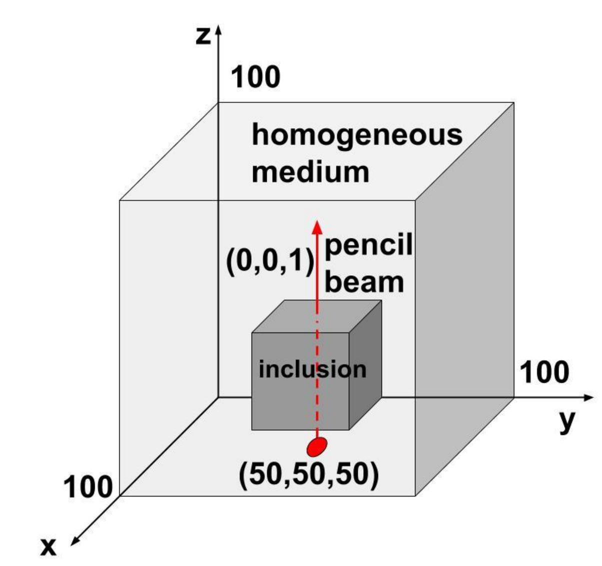


Fig. 1 a $40 \times 40 \times 40$ inclusion in cube

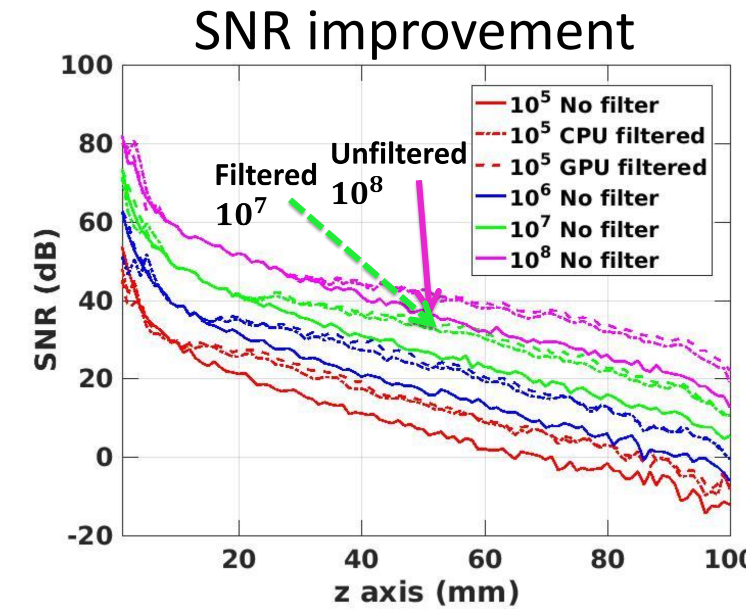
Metric

$SNR(dB) = 20\log_{10}\frac{\mu}{\sigma}$

where μ and σ^2 are mean and variance.

Results

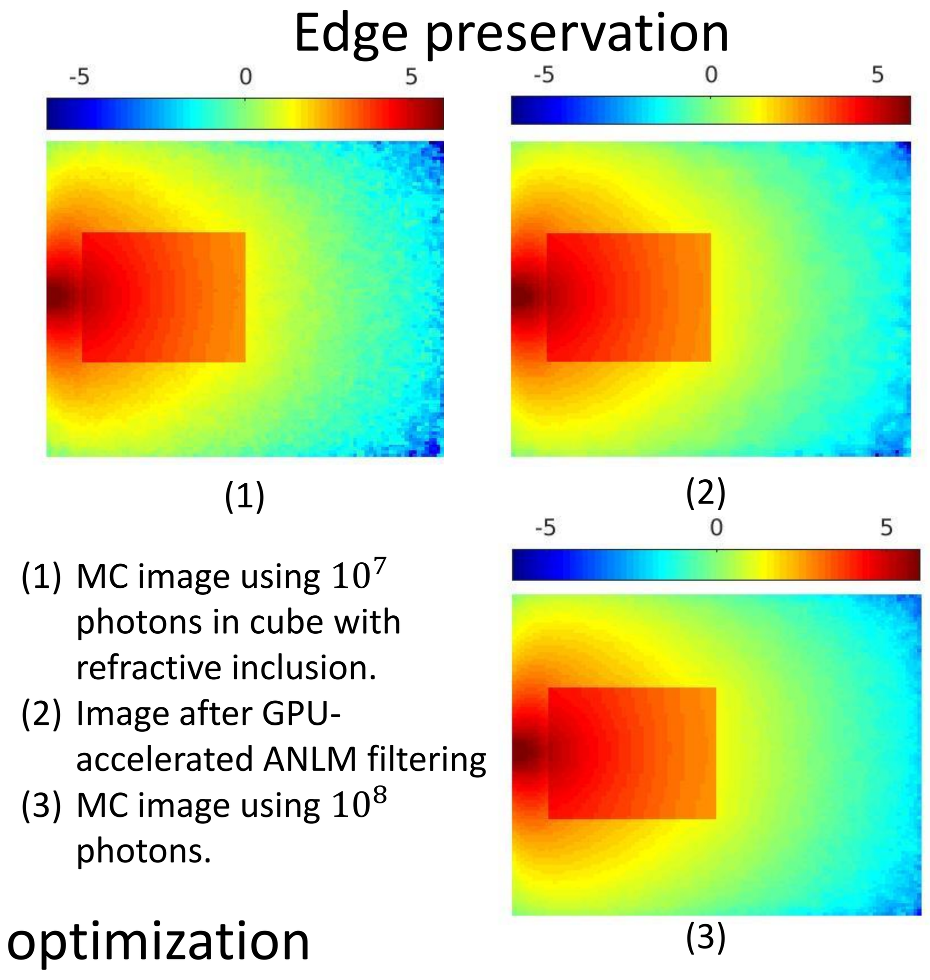
SNR improvement



SNR trend along red arrow in Fig. 1 in homogeneous cube

- The improvement is around 10 dB
- Independent of the photon count
- Comparable to increasing photon count by 10-fold.

Edge preservation



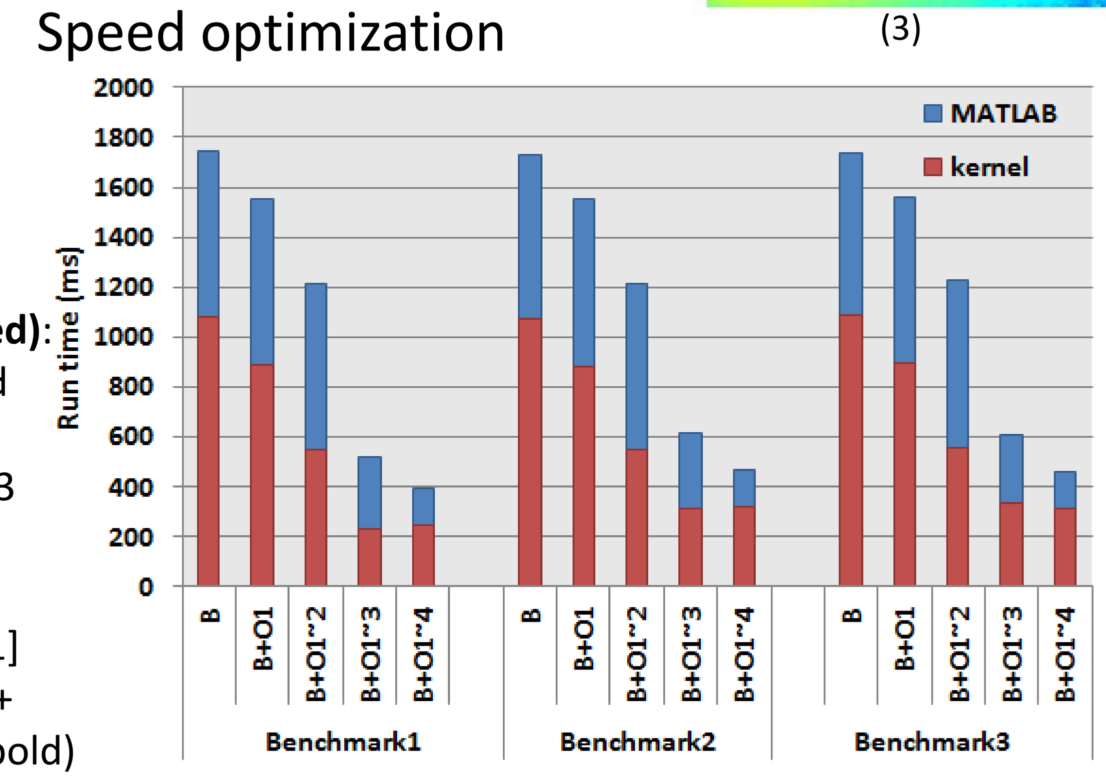
(1) MC image using 10^7 photons in cube with refractive inclusion.
(2) Image after GPU-accelerated ANLM filtering
(3) MC image using 10^8 photons.

Speed optimization

- **Benchmark 1~3:** 3 types of cube (see Experiments)
- **B:** Baseline from Granata et al [2]
- **O1~O4:** Optimization 1 to 4

Bar chart (sub-band mixing excluded):
1. Kernel time (red bar) is improved with O1 to O4
2. CPU overhead is reduced with O3 (blue)

Table (sub-band mixing included):
1. 5x~6x faster than CPU filter in [1]
2. Overall run-time of (GPU ANLM+ MC)<MC when # photon>10^7 (bold)



	Homogeneous (sec)			Absorbing inclusion (sec)			Refractive inclusion (sec)		
Photon#	MCX	GPU-ANLM	CPU-ANLM	MCX	GPU-ANLM	CPU-ANLM	MCX	GPU-ANLM	CPU-ANLM
10^5	0.1837	1.0084	5.8466	0.1824	0.9192	4.2923	0.1720	1.2762	3.9410
10^6	0.3359	1.1332	6.5333	0.3313	1.0307	5.4646	0.4390	0.9563	5.7687
10^7	**1.1746**	**0.8224**	6.6445	**1.1819**	**1.2076**	5.9047	**2.1839**	**0.9379**	6.5319
10^8	**8.3531**	0.9441	6.1433	**8.3537**	1.1286	5.6366	**17.7453**	0.8432	6.3297

Conclusions

We optimized the GPU-ANLM filter to achieve 5x~6x speedup (2 filtering+sub-band mixing process) compared to multi-thread CPU version [1] and 3x~4x speedup (2 filtering process) compared to the architecture proposed in [2]. To achieve shorter overall run-time, the application of GPU ANLM filter on MC simulation will have a speed benefit for photon number above 10^7.

References

[1] Manjón J V, Coupé P, Martí-Bonmatí L, et al. Adaptive non-local means denoising of MR images with spatially varying noise levels[J]. Journal of Magnetic Resonance Imaging, 2010, 31(1): 192-203.
[2] Granata D, Amato U, Alfano B. MRI denoising by nonlocal means on multi-GPU[J]. Journal of Real-Time Image Processing, 2016: 1-11.

Acknowledgement

This research is supported by National Institutes of Health (NIH) grants # R01-GM114365 and R01-CA204443