Nucl Med Mol Imaging
Vol. 43, No. 5, Oct 2009

Original Article

방출단층촬영 시스템을 위한 GPU 기반 반복적 기댓값 최대화 재구성 알고리즘 연구

서울대학교 공과대학 전기공학부[1], 서울대학교 의과대학 핵의학교실 및 방사선응용생명과학 협동과정[2], 서울대학교 의과대학 핵의학교실 및 의공학 협동과정[3]
하우석[1] · 김수미[2] · 박민재[3] · 이동수[2] · 이재성[2]

A Study on GPU-based Iterative ML-EM Reconstruction Algorithm for Emission Computed Tomographic Imaging Systems

Woo Seok Ha[1], Soo Mee Kim, M.S.[2], Min Jae Park, M.S.[3], Dong Soo Lee, M.D.[2], and Jae Sung Lee, Ph.D.[2]
[1]Department of Electrical Engineering, Seoul National University College of Engineering, Seoul, Korea; [2]Department of Nuclear Medicine and Interdisciplinary Program in Radiation Applied Life Science Major, [3]Department of Nuclear Medicine and Interdisciplinary Program in Biomedical Engineering Major, Seoul National University College of Medicine, Seoul, Korea

Purpose: The maximum likelihood-expectation maximization (ML-EM) is the statistical reconstruction algorithm derived from probabilistic model of the emission and detection processes. Although the ML-EM has many advantages in accuracy and utility, the use of the ML-EM is limited due to the computational burden of iterating processing on a CPU (central processing unit). In this study, we developed a parallel computing technique on GPU (graphic processing unit) for ML-EM algorithm. **Materials and Methods:** Using Geforce 9800 GTX+ graphic card and CUDA (compute unified device architecture) the projection and backprojection in ML-EM algorithm were parallelized by NVIDIA's technology. The time delay on computations for projection, errors between measured and estimated data and backprojection in an iteration were measured. Total time included the latency in data transmission between RAM and GPU memory. **Results:** The total computation time of the CPU- and GPU-based ML-EM with 32 iterations were 3.83 and 0.26 sec, respectively. In this case, the computing speed was improved about 15 times on GPU. When the number of iterations increased into 1024, the CPU- and GPU-based computing took totally 18 min and 8 sec, respectively. The improvement was about 135 times and was caused by delay on CPU-based computing after certain iterations. On the other hand, the GPU-based computation provided very small variation on time delay per iteration due to use of shared memory. **Conclusion:** The GPU-based parallel computation for ML-EM improved significantly the computing speed and stability. The developed GPU-based ML-EM algorithm could be easily modified for some other imaging geometries. (Nucl Med Mol Imaging 2009;43(5):459-467)

Key Words: SPECT, PET, image reconstruction, GPU, CUDA

서 론

핵의학 분야의 대표적 영상 시스템인 단일광자단층촬영(single photon emission computed tomography, SPECT)과 양전자단층촬영(positron emission tomography, PET)에서 널리 사용되는 영상 재구성 방법은 여과 역투사(filtered backprojection, FBP)와 최대우도 기댓값 최대화(maximum likelihood-expectation maximization, ML-EM) 알고리즘으로 알려져 있다. FBP 영상 재구성 알고리즘은 램프 필터(ramp filter)와 저주파 통과 필터(low pass filter)를 취한 검출데이터의 역투사(backward projection) 과정을 통하여 보다 쉽고 빠르게 영상을 재구성하는 반면 데이터의 검출과정에서 발생하는 통계적(statistical) 및 물리적 잡음(physical noise)으로 인한 영향을 제어하기 어렵다는 단점이 있다.[1-6] 그러나 방출 및 검출과정을 포아송(Poisson) 확률분포로 표현되는 통계학적 모델을 기반으로 유도된 ML-EM 알고리즘[7,8]은 FBP 방법에 비

• **Received**: 2009. 7. 31. • **Revised**: 2009. 9. 2.
• **Accepted**: 2009. 9. 3.
• Address for reprints: Jae Sung Lee, PhD., Department of Nuclear Medicine, Seoul National University College of Medicine, 28 Yungun-dong, Chongno-gu, Seoul 110-799, Korea
Tel: 82-2-2072-2938, Fax: 82-2-745-2938
E-mail: jaes@snu.ac.kr
※본 연구는 과학기술부 및 과학재단의 지원을 받아 기초공동연구소 과제 (2009-0078724)와 고유강점기술육성 과제 (2008-03852)를 통하여 수행되었음.

하여 월등히 향상된 질의 영상을 얻을 수 있다.

ML-EM 알고리즘은 투사(forward projection) 및 역투사 과정을 정해진 횟수만큼 반복함으로써 영상을 재구성하는데 일반적으로 수십 번의 반복연산을 수행해야 한다. 또한 재구성된 영상의 화소 수 M과 검출 데이터를 저장하는 bin 개수 N이 주어졌을 때 투사 및 역투사의 반복연산 당 $M \times N$ 에 비례하는 연산량이 요구되므로 영상이나 데이터를 표현하는 차원(dimension)이 커질수록 급격히 연산량이 증가하게 된다. ML-EM의 재구성 속도를 가속화하기 위하여 OSEM(ordered subset EM)[9]과 같이 검출데이터에 대한 부분집합을 사용하여 반복횟수를 현격히 줄이는 알고리즘이 제안되었지만 여전히 여러 번의 반복연산을 수행하여야 한다.

한편 최근 컴퓨터 하드웨어 산업분야에서 그래픽처리장치(graphic processing unit, GPU)의 성능 발전이 눈부시다. 그래픽 산업의 폭발적인 성장에 따른 수요증가와 GPU 특유의 병렬적인 구조로 인한 다른 처리장치에 비하여 클록(clock)수 향상이 손쉽다는 점이 성능 발전의 요인이 되고 있다. 이런 관점에서 그동안 일반적으로 중앙처리장치(central processing unit, CPU)가 도맡아 왔던 컴퓨터 작업들을 GPU가 보다 빠른 속도로 처리하는 응용 예가 늘어나고 있는 추세이다. GPU는 여러 작은 멀티프로세서들이 일을 분담해 병렬적으로 해당하는 연산을 수행하기 때문에 두 개 혹은 네 개의 멀티프로세서로 구성된 CPU보다 효율적인 계산을 수행할 수 있다.

본 연구에서는 ML-EM 알고리즘의 투사 및 역투사 과정에 대한 병렬화 전략을 개발하고 이를 GPU 기반으로 구현하여 연산속도의 향상을 꾀하였다. 보다 빠른 영상재구성을 위한 기존의 가속화된 접근방법인 OSEM 알고리즘 및 고성능 다중 CPU[10,11]가 장착된 컴퓨터의 사용에 비하여 GPU 기반 영상 재구성법은 가속화된 연산속도를 얻을 뿐 아니라 가격대비 클록 수 측면에서 보다 저렴한 시스템을 구현할 수 있다.[12] 또한 GPU 기반 시스템은 다중 GPU를 서로 물리적으로 연결하면 별도의 호환 프로그램 없이 바로 동작할 수 있는 다중 GPU 솔루션이 이미 존재하므로 시스템의 업그레이드 작업이 간편하다는 이점도 기대할 수 있다.[13,14] GPU 기반 영상 재구성 알고리즘은 NVIDIA사에서 발표한 쿠다(computed unified device architecture, CUDA) 기술[15,16]을 사용하여 구현하였으며 이를 통한 성능 향상 여부를 평가하였다. CUDA를 이용한 GPU 기반 알고리즘의 성능은 GPU를 구성하는 각 프로세서에 배분되는 작업의 구성에 따라 영향을 받으므로 본 연구에서는 ML-EM 알고리즘의 병렬화 과정에 대한 최적화를 수행하였으며, 연산시간 및 재구성된 영상의 정확도 측면에서 단일 CPU(single CPU) 기반으로 구현된 알고리즘의 결과와 비교 평가하였다.

대상 및 방법

1. 기댓값 최대화 (maximum likelihood-expectation maximization, ML-EM) 영상 재구성 알고리즘

여과역투사 방법에 비하여 보다 좋은 질의 영상을 재구성하는 ML-EM 알고리즘은 투사 및 역투사를 여러 번 반복하여 참값(true solution)에 근접한 결과를 얻는 반복적 영상재구성법(iterative reconstruction)의 대표적인 기법이다. 일반적으로 반복적 영상재구성법은 다음과 같은 순서를 따라 수행된다.

① 재구성될 영상의 각 화소를 임의의 양수로 초기화하여 추정한다.

② 추정된 영상을 실제 영상 획득 과정을 묘사한 투사과정을 통하여 투사데이터의 추정치를 계산한다.

③ 실제로 시스템을 통하여 측정한 투사데이터와 ②번에서 추정한 투사데이터를 비교한 후 오차에 해당하는 수치들을 반영하여 재구성될 영상을 새로이 추정한다.

④ 정해 놓은 회수만큼 ②~③ 과정을 반복하여 최종 결과를 얻는다.

위 알고리즘에서 ②번은 투사 과정으로, ③번은 투사데이터의 실측치와 추정치의 오차를 재구성될 영상에 반영하는 역투사 과정으로 간주되어진다.

$$p_{r\theta} = \sum_{ij} H_{ij,r\theta} f_{ij} \qquad (1)$$

$$f_{ij} = \sum_{r\theta} H_{ij,r\theta} p_{r\theta} \qquad (2)$$

투사 및 역투사 과정은 각각 식 (1)과 (2)처럼 수식으로 표현할 수 있다. 수식 (1)과 (2)에서 f_{ij}는 재구성될 영상의 (i,j)번째 화소 값을, $p_{r\theta}$는 회전각도 θ에 놓인 검출기의 r번째 검출격자에 검출된 투사데이터를 나타낸다. 시스템 행렬 $H_{ij,r\theta}$는 (i,j)번째 화소에서 θ 방향으로 방출된 하나의 광자가 r번째 검출격자에서 검출될 확률을 나타낸다.

투사 및 역투사의 반복연산을 통하여 영상을 재구성하는 ML-EM의 최종 식은 식 (3)과 같이 수식적으로 표현할 수 있다. 식 (3)에서 n은 반복횟수를 나타낸다.

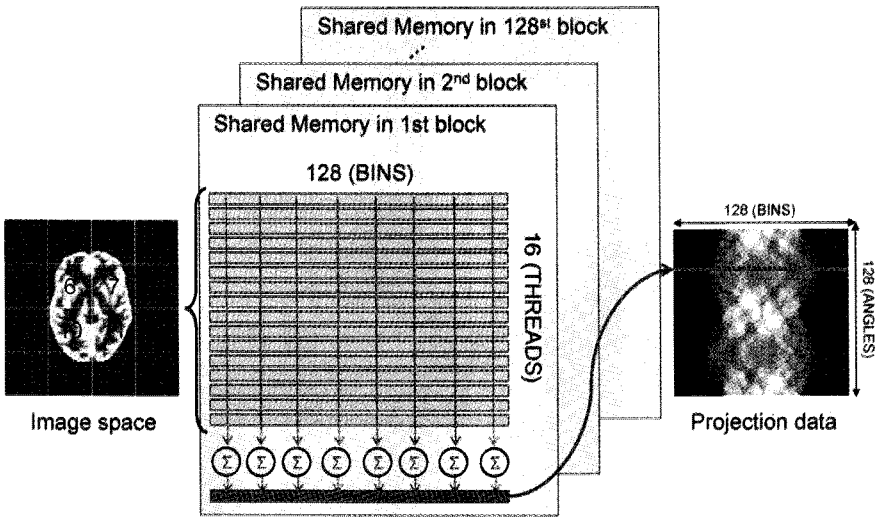

Figure 1. It represents a strategy for GPU-based parallel processing for ML-EM reconstruction algorithm.

$$f_{ij}^{n+1} = \frac{f_{ij}^{n}}{\sum_{r\theta} H_{ij,r\theta}} \sum_{r\theta} \frac{H_{ij,r\theta} p_{r\theta}}{\sum_{mn} H_{mn,r\theta} f_{mn}^{n}} \qquad (3)$$

시스템 행렬 $H_{ij,r\theta}$는 검출시스템에 관련된 확률값으로 시스템이 주어지면 ML-EM 알고리즘의 반복 연산과정에서 불변하는 항목이므로 일반적으로 미리 계산하여 메모리에 저장한 후 반복 연산 시 단순히 불러들여 사용할 수 있다. 식 (3)에서 우변의 두 번째 항은 분모의 추정 투사데이터를 계산하기 위한 투사과정과 분자의 실측 투사데이터 사이의 오차에 대한 역투사 과정으로 구성되어진다.

2. 쿠다(computed unified device architecture, CUDA)

쿠다는 NVIDIA사에서 2007년 발표한 C 프로그래밍 언어 기반 범용 그래픽처리장치(general-purpose computing on graphics processing units, GPGPU) 기술로, 기존에 CPU로부터 그래픽 관련 연산 작업만 넘겨받아 수행하던 GPU의 기능을 넘어 프로그래머로 하여금 직접적 접근을 허용함으로써 일반적인 목적의 프로그램들을 GPU에서 수행할 수 있게 해준다. 다중 코어(multi-core)의 커다란 연산장치들이 많은 양의 작업을 재빠르게 차례대로 처리해 나가는 다중 CPU 기반 병렬작업과 달리, GPU는 여러 개의 작은 연산장치들이 작업의 양을 잘게 나누어 처리하는 병렬처리(parallel processing) 구조를 가지고 있다. 작업의 특성에 따라 다중 CPU와 GPU의 병렬연산 수행 효율이 다를 수 있다. 기존의 C 소스파일 대신 쿠다 프로그래밍 언어를 적용한 CU 소스파일을 사용하며 NVIDIA사에서 쿠다 드라이버와 함께 제공한 컴파일러 NVCC를 이용하여 컴파일하는 것이 쿠다의 특징이다. NVIDIA사의 Geforce와 Tesla 시리즈 GPU에만 쿠다가 적용될 수 있다는 한계가 있다.

3. ML-EM의 투사 및 역투사에 대한 병렬화 전략

쿠다를 이용하여 기존의 영상 재구성 알고리즘을 GPU 기반 알고리즘으로 바꾸기 위하여 ML-EM 알고리즘에 대한 병렬화 전략을 구상하여야 한다. 이 과정에서 기존의 작업들을 병렬적으로 세분화하여 나눈 각각의 개체를 스레드(thread)라고 말한다. 세분화된 모든 스레드의 작업량이 균등하게 분배되어야 전체적인 GPU의 병렬처리 능력을 최대화할 수 있다. 또한 세분화된 스레드는 동시에 수행되기 때문에 각 스레드의 연산결과에 서로 영향을 주지 않는 구조로 병렬화 되어야 한다. 특히 병렬화 과정에서 효율적인 계산 작업을 위하여 무척 중요하게 고려되어야 할 점은 스레드가 GPU의 전역 메모리(global memory)에 접근하는 빈도를 최소화하여야 한다는 것이다. 이는 CPU가 램(RAM)에서 레지스터로 하나의 데이터 블록을 읽어올 때 CPU 기본 연산의 수만 배에 해당하는 시간이 걸리는 것과 같은 맥락으로 이해할 수 있다.

쿠다 기반 GPU의 경우, 블록(block)이라는 개념을 통하여 일련의 스레드 작업들을 하나로 묶어 접근시간이 무척 빠른 공유 메모리(shared memory)를 함께 제공하고 있다. 그러므로 병렬연산을 수행하는 기본 단위는 블록과 스레드를 이용하여 결정되어진다. 모든 스레드가 계산과정에서 나

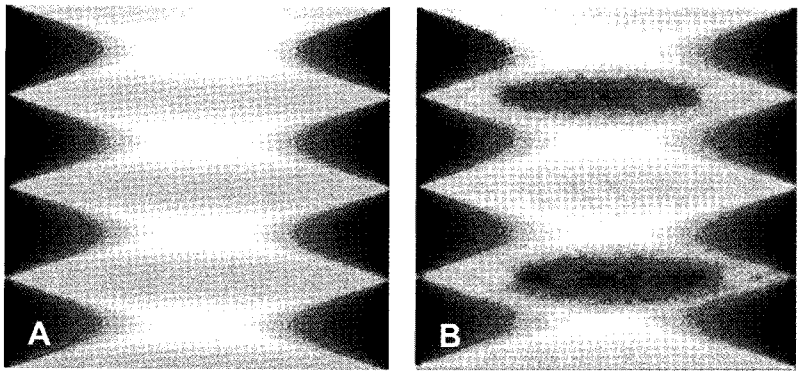

Figure 2. Due to bank conflict during forward projection, an artifact was produced: (A) expected image, (B) image with artifact because of bank conflict problem.

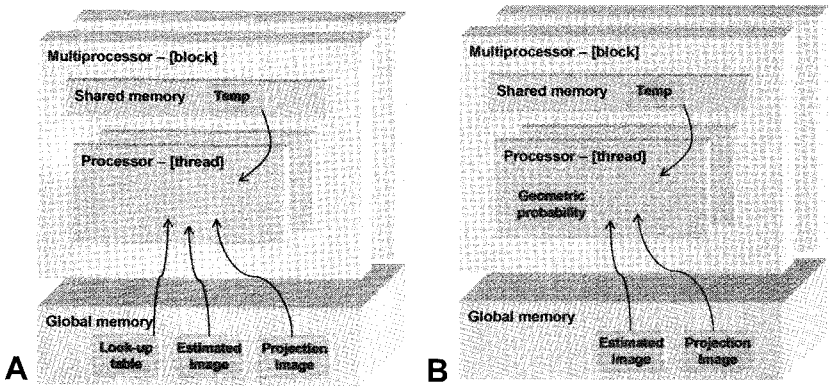

Figure 3. It represents a process and a memory structure for GPU's parallel processing strategy: (A) using LUT in global memory, (B) repeating computation of system matrix in each iteration.

오는 부산결과를 일일이 전역 메모리에 접근하여 저장하기엔 많은 시간이 소요되므로, 블록 내 공유 메모리에 결과를 저장한 후 같은 블록 내 모든 스레드의 작업이 끝나면 일괄적으로 전역 메모리로 데이터를 복사하게 된다. 각 스레드에 균등한 작업량을 할당하기 위한 병렬연산 전략으로 영상공간을 여러 부분영역으로 분할하여 스레드에 할당하였다. 본 연구에서는 Fig. 1에 나타난 바와 같이 영상공간을 4×4, 즉 총 16개 부분영역으로 서로 겹치지 않게 구분하였다. 그러므로 각 스레드는 영상공간의 분할된 부분영역 중 하나를 배정받아 투사 및 역투사 과정을 수행하게 된다. 16개의 스레드는 하나의 블록을 구성하며 총 블록 수는 피사체 주변에서 촬영하는 시스템의 이산회전각도 개수에 해당하도록 설계되어 각 회전각도에 대하여 병렬적으로 투사 및 역투사 과정이 수행되었다.

병렬연산의 특성상 같은 블록 내 모든 스레드가 공유 메모리에 동시에 접근하여 의도한 대로 정확한 값이 메모리에 저장되지 못하는 뱅크 충돌(bank conflict) 문제가 발생할 수 있다. Fig. 2에 나타나는 바와 같이 투사과정 중 뱅크 충돌에 의하여 인공산물(artifact)이 발생할 수 있다. Fig. 2(A)와 (B)는 전체적인 개형은 비슷하지만 투사과정에 대한 병렬연산의 결과인 (B)는 기대치 (A)에 비하여 정확하게 연산되지 못하고 뱅크 충돌에 의하여 예기치 않은 값이 발생함을 확인할 수 있다. 이러한 뱅크 충돌을 피하기 위한 전략으로 Fig. 1에 나타난 바와 같이 공유 메모리 내 각 스레드가 접근하는 부분을 영상공간에 대한 부분영역 개수만큼 나누어 모든 스레드 작업이 끝난 후 모든 결과의 합계를 계산하는 과정을 추가하였다. 현재 사용하는 GPU의 공유 메모리 용량이 16 Kbytes로 제한되어 있어 스레드 수를 16개 이상으로 늘릴 수 없다는 제한점이 있다.

4. 시스템 행렬 계산

탐색표는 연산과정 중 불변하는 정보를 미리 연산하고 기억장치에 저장한 후 실제 연산 시 언제든지 쉽게 참조할 수 있도록 만든 자료집합을 의미한다. 반복적 연산과정을

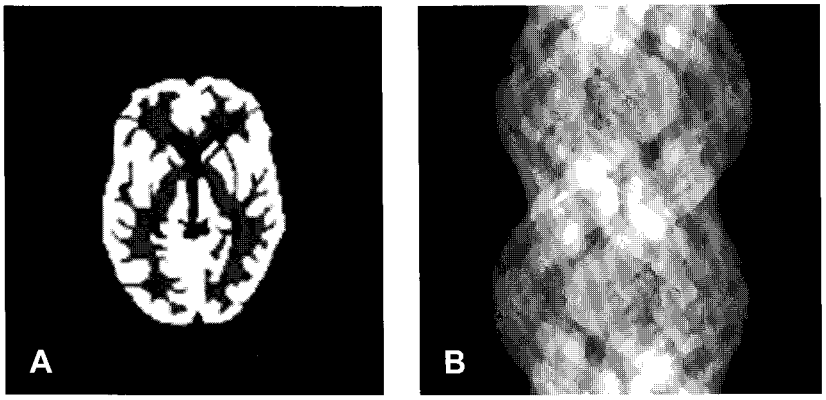

Figure 4. For simulation, it is used an (A) Hoffman brain software phantom and generated a (B) projection data for the phantom from projector modelled with pixel-driven method.

Table 1. Computation Specification

Device	Model
CPU[a)]	Intel(R) Core(TM) DUO 6300 @ 1.86GHz x 2
RAM[b)]	Samsung 1G DDR2 6400 SAM x 2
GPU[c)]	NVIDIA GeForce 9800 GTX+

[a)]Centeral Processing Unit, [b)]Random Access Memory, [c)]Graphic Processing Unit.

수행하는 ML-EM 알고리즘의 경우, (i,j)번째 화소에서 θ 방향으로 방출된 하나의 광자가 r번째 검출격자에서 검출될 확률 $H_{ij,r\theta}$는 반복연산 시 불변하는 항목이며 상당한 연산시간이 소요되므로 반복횟수만큼 매번 계산하는 것보다 미리 계산 후 저장한 값을 불러들여 사용하는 것이 효율적이다. 그러나 GPU 기반 병렬연산의 경우 탐색표를 전역 메모리에 저장한 후 사용하게 되면 메모리 전송속도가 너무 느려 때론 매번 동일한 연산을 직접 계산하는 것보다 메모리 내 탐색표로부터 값을 읽어 오는 시간이 더 오래 걸리는 경우가 발생할 수 있다. 그러므로 본 연구에서는 다소 비효율적이지만 탐색표를 사용하지 않고 각 스레드 내에서 매번 직접 계산하여 전역메모리에 대한 접근으로 인한 시간 지연을 줄이고자 하였다. Fig. 3은 GPU 내 스레드가 담당하는 연산과 메모리에 저장되는 데이터에 대한 개념도를 나타내고 있다. Fig. 3A는 탐색표를 사용하는 경우를, B는 탐색표 없이 스레드 안에서 직접 매번 계산하는 경우를 나타내고 있다. Fig. 3A에 나타난 바와 같이 시스템 행렬을 저장할 탐색표, 재구성될 영상과 투사과정을 통하여 얻은 투사데이터는 GPU 내 전역 메모리에 존재하므로 EM 알고리즘을 수행하기 위하여 매 반복연산마다 4번의 스레드와 전역 메모리 사이의 전송이 이루어져야 한다. 2번의 전역 메모리 접근은 투사 및 역투사 과정 중 탐색표의 참조로 인하여 필요하며 나머지 2번의 접근은 투사데이터와 재구성 영상을 참조할 때 필요하다. 탐색표를 제거한 Fig. 3B의 경우, 시스템 행렬은 매 반복연산마다 계산되며 전역 메모리에 대한 접근은 2번으로 줄어들게 된다.

5. 실험 조건

실험에 사용한 컴퓨터 하드웨어 사양은 Table 1과 같다. GeForce 9800 GTX+는 512개의 thread와 512×512×64개의 block으로 설정하는 것이 가능하여 본 연구에서 사용하기에 적합하였다. 전역 메모리는 512 Mbytes, 공유 메모리는 16 Kbytes의 용량을 제공한다.

실험을 위하여 Fig. 4A와 같은 128×128 크기의 Hoffman 뇌 소프트웨어 모형을 사용하였으며 화소구동법(pixel-driven method)으로 구현된 투사기(projector)를 이용하여 피사체 주변을 3도마다 회전하여 Hoffman 뇌 모형에 대한 투사데이터 (B, 120×128)를 얻을 수 있었다. ML-EM 알고리즘을 이용하여 Fig. 4B와 같은 투사데이터를 재구성하였다. 단일 CPU와 GPU 기반 ML-EM 알고리즘을 32, 64, 128, 256, 1024번의 반복횟수만큼 수행하였다. 최종 재구성된 결과 영상에 대하여 식 (4)와 같이 계산된 퍼센트 오차 (percent error, PE) 및 연산시간을 비교하였다. 또한 GPU 기반 ML-EM의 구현 시 탐색표 존재여부에 따른 연산시간의 변화를 관찰하였다.

$$PE = \sqrt{\frac{\sum(f_{original} - f_{estimated})^2}{\sum f_{original}^2}} \times 100(\%) \quad (4)$$

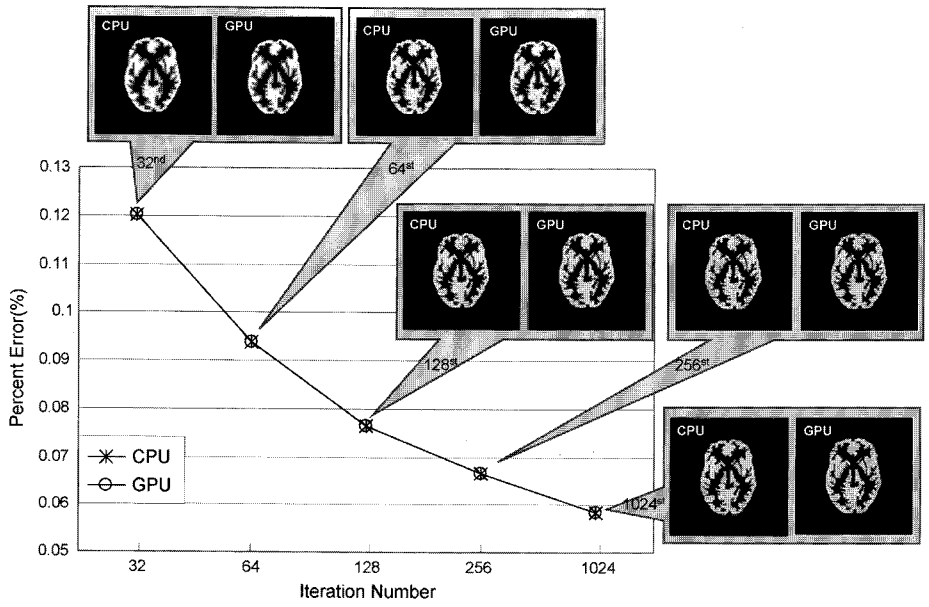

Figure 5. A percent error graph and reconstructed images from CPU- and GPU-based ML-EM algorithms were obtained using various iterations of 32, 64, 128, 256 and 1024.

Table 2. Computation Times for CPU and GPU based ML-EM Reconstructions (Using LUT Stored in Global Memory)

Iteration number	Latency (msec)		Ratio
	CPU[a)]	GPU[b)]	
32	3832	1224	x3.1
64	7641	2433	x3.1
128	28852	4848	x5.9
256	142676	9695	x14.7
1024	1066182	38699	x27.6

[a)]Centeral Processing Unit, [b)]Graphic Processing Unit.

Table 3. Computation Times for CPU and GPU based ML-EM Reconstructions (Without LUT and Repeating Computation in Each Iteration)

Iteration number	Latency (msec)		Ratio
	CPU[a)]	GPU[b)]	
32	3832	258	x14.8
64	7641	506	x15.1
128	28852	1003	x28.8
256	142676	1996	x71.5
1024	1066182	7932	x134.4

[a)]Centeral Processing Unit, [b)]Graphic Processing Unit.

결 과

Fig. 5에서 단일 CPU와 GPU 기반 ML-EM 영상 재구성 알고리즘의 결과 영상과 퍼센트 오차를 비교하였다. Fig. 5의 그래프에 나타난 바와 같이 정량적인 비교를 위하여 계산한 퍼센트 오차를 비교해보면 단일 CPU와 GPU 기반 ML-EM 알고리즘에서 얻은 결과 사이에 차이를 발견할 수 없었다. 또한 지정된 반복 횟수(32, 64, 128, 256, 1024번)마다 얻은 결과 영상을 정성적으로 비교해 보아도 거의 유사함을 확인할 수 있다.

GPU 기반 ML-EM 알고리즘 구현 시 탐색표의 존재여부에 따른 연산시간을 비교하였다. 미리 계산하여 전역 메

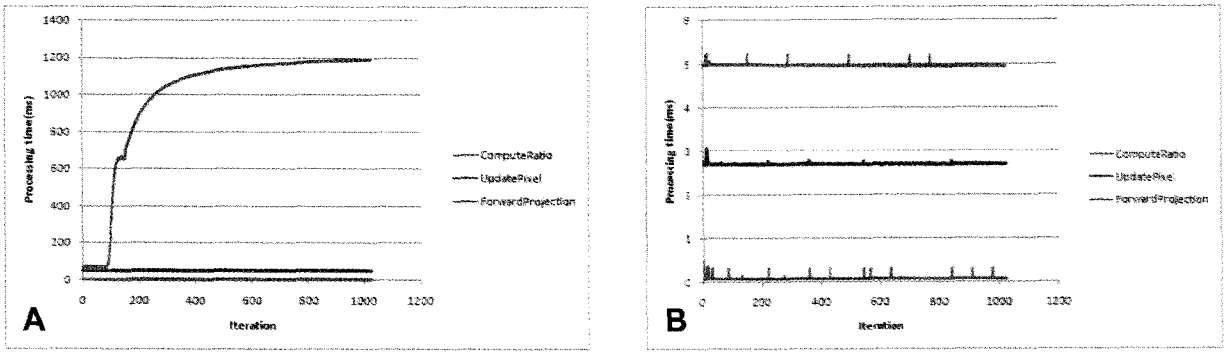

Figure 6. Processing time of the sub-functions including computations on forward projection, backprojection of error between measured and estimated projection data, and updating error into estimated image, were compared for (A) CPU and (B) GPU-based ML-EM.

모리에 저장한 시스템 행렬을 사용하여 구현한 GPU 기반 ML-EM 알고리즘은 Table 2에 나타난 바와 같이 단일 CPU 기반 재구성의 결과보다 연산속도가 최소 3배에서 최대 27배까지 더 빨라지며 동일한 질의 영상을 얻는 것을 확인할 수 있었다. 그러나 탐색표를 제거하고 각 블록 내 스레드에서 직접 매번 계산하는 방식으로 GPU 기반 ML-EM을 구현하였을 때 Table 3에 나타난 바와 같이 연산속도가 상당히 개선되었음을 확인할 수가 있다. 또한 단일 CPU 기반의 경우보다 최소 15배에서 최대 134배까지 개선된 연산속도를 보였다. 즉 GPU 구조상 전역 메모리 접근으로 인한 지연된 연산시간을 5배 정도 빠르게 최적화할 수 있었다.

ML-EM 알고리즘은 투사데이터의 추정치를 계산하는 투사과정, 투사데이터의 실측치와 추정치의 오차에 대한 역투사 과정 및 재구성될 영상에 오차를 갱신하는 작업으로 세분화 할 수 있다. 반복횟수가 증가할수록 단일 CPU와 GPU 기반 연산속도의 비가 더 커지는 경향의 원인을 규명하기 위하여 반복횟수에 따른 각 세분화된 작업에 대한 지연시간을 측정하였다. Fig. 6은 ML-EM의 각 세분화된 작업에 대한 연산 시간을 비교한 그래프이다. Fig. 6A와 B는 각각 단일 CPU와 GPU 기반 연산에 해당한다. Fig. 6에서 확인할 수 있는 것처럼 반복횟수가 진행될수록 단일 CPU와 GPU 기반 연산 사이의 속도비가 커지는 이유는 단일 CPU 기반 ML-EM에서 투사과정의 지연시간이 반복횟수가 증가할수록 무척 커지기 때문이며 이는 과도한 반복 작업으로 인한 CPU 점유율의 저하에 기인한 것으로 보인다. 그러므로 단일 CPU와 GPU 기반 ML-EM 알고리즘의 실질적인 연산속도 비는 15배라고 볼 수 있다. 즉 GPU를 이용한 병렬연산을 통하여 15배 정도의 개선된 성능을 구축할 수 있었다. 기대한 바와 같이 GPU의 병렬처리 기능을 이용한 영상 재구성 방법이 기존의 단일 CPU 기반 방법에 비하여 훨씬 빠른 것으로 관찰되지만, 그 정도의 차이는 병렬화 전략에 따라 크게 좌우될 수 있다.

고 찰

본 연구에서는 ML-EM 영상재구성 알고리즘의 병렬연산을 GPU 기반으로 개발하였다. ML-EM에서 가장 많은 연산시간이 소요되는 투사 및 역투사 과정을 병렬화 하였으며 재구성된 영상을 표현하는 영상공간과 검출기의 회전각도에 따라 병렬화 전략을 구축하였다. 영상공간을 부분영역으로 분할하여 스레드를, 회전각도에 따라 스레드의 집합인 블록을 구성하였다.

실험결과에 나타난 바와 같이 GPU 기반 ML-EM 영상재구성이 단일 CPU 기반의 경우에 비하여 15배 정도 개선된 연산능력을 보였다. 또한 단일 CPU와 GPU 기반 ML-EM 영상 재구성법이 모두 동일한 퍼센트 오차를 보였으며 두 결과 영상의 차이를 육안으로도 볼 수 없었다. 이는 GPU 기반 ML-EM이 연산에 대한 병렬처리 기능만 추가되었을 뿐 재구성 알고리즘의 정확도를 변화시킨 것은 아니라는 점과 부합한다. 특히 퍼센트 오차 측면에서 GPU 기반 ML-EM의 결과 영상이 단일 CPU 기반의 경우에 비하여 퍼센트 오차 측면에서 굉장히 작은 차이 (<0.000001)를 보였는데 이는 쿠다에서 사용하는 삼각함수와 일반 C 언어에서 제공하는 삼각함수의 차이로 무시할 수 있을 정도로 작았다.

GPU 기반 ML-EM 영상 재구성의 병렬연산의 속도는 최적화 여부에 따라 크게 좌우되는 경향을 보였다. 작업에 대한 병렬화 전략과 관련하여 스레드 수가 프로세서 수에 비하여 작거나 스레드 간 작업량이 불균등할 경우 한 프로

세서가 남은 작업량을 처리하는 동안 이미 연산을 마친 다른 프로세서들은 하는 일없이 가만히 있게 되는 비효율성이 발생할 수 있다. 본 연구에서 사용하는 GPU의 스레드 수는 16개에 불과하여 GPU 기반 병렬연산의 성능을 최대로 끌어올리는데 제한이 있었다. 알고리즘 개선을 통하여 더 많은 스레드 분할이 가능하여 진다면 병렬연산 속도는 두 세배 더 빨라질 것으로 기대된다. 또한 상당한 시간이 소요되는 GPU의 전역 메모리 접근 시간을 고려하여 메모리 접근 횟수를 줄이는 방식으로 병렬화를 구현하였으며 탐색표 참조를 삭제하고 각 스레드에서 ML-EM의 반복연산 시 불변하는 항목인 시스템 행렬을 매번 직접 계산하는 방법을 택하였다. 그러나 검출 시스템의 구조에 따라 시스템 행렬의 연산량이 결정되므로 GPU 기반 병렬연산 시 탐색표 참조여부에 따른 병렬연산 속도의 변화를 확인하고 최적화된 방식을 선택하여야 한다.

본 실험에 사용된 ML-EM 알고리즘은 적절한 병렬화 전략을 통하여 오차 없이 연산속도를 향상시키는 GPU 기반 프로그램으로 구현되었다. 최적화된 병렬화 전략을 구축하지 못하면 서로 다른 프로세서에서 동일한 메모리에 접근하여 올바른 신호 전달이 이루어지지 않는 뱅크충돌 현상이 발생할 수 있으며 이로 인하여 GPU 기반으로 병렬화된 프로그램이 오히려 기존의 경우보다 훨씬 느린 처리속도를 보일 수도 있다. 그러므로 다른 영상 재구성 알고리즘에 대하여 병렬화를 구현하려면 병렬화 가능 여부를 확인하고 최적화된 방법을 구축해야 한다. 본 연구에서 구축한 GPU 기반 병렬연산 전략은 다른 영상 시스템에 대한 병렬 재구성 기법 개발을 위하여 확장될 수 있다. 부채살 조준기를 장착한 SPECT 시스템[17]과 같이 특정한 검출 경로를 모델링하여 영상을 재구성 하거나 고해상도 영상을 지원하기 위하여 반응 깊이(depth of interaction, DOI)를 측정하는 DOI-PET 시스템[18-19]과 같이 방대한 양의 데이터를 처리해야 하는 경우에 GPU 기반 병렬 연산은 유용할 것으로 기대된다. 또한 선 적분(line integration)을 기반으로 모델링하는 일반적인 SPECT나 PET 시스템과 달리 타원추 표면 적분(conical surface integration)을 기반으로 3차원 데이터를 직접 다루어야 하는 컴프턴 카메라(Compton camera) 시스템[20,21]은 영상 재구성 시 상당한 연산시간을 소요하므로 GPU 기반 병렬연산의 적용이 필요한 연구 분야이다.

GPU기반으로 수행한 ML-EM 영상재구성법의 병렬연산으로 인한 빠른 속도를 이용하여 기존의 영상 재구성 시스템을 GPU 기반 시스템으로 대체할 수 있을 것으로 기대된다. 또한 여러 개의 GPU를 물리적으로 연결한 후 이미 존재하는 다중 GPU 솔루션을 이용하여 별도의 호환 프로그램 없이 바로 다중 GPU 시스템을 구축할 수 있어 시스템 성능 업그레이드도 한층 간편할 것이다.

본 연구에서는 NVIDIA사의 쿠다 기술을 이용하여 같은 회사의 GPU 제품에만 적용할 수 있는데 반해 향후 연구에서는 최근 새로이 개발된 개방형 범용 병렬 컴퓨팅 프레임워크인 OpenCL(Open Computing Language) 기술을 사용하여 다른 회사의 GPU뿐 아니라 CPU도 함께 병렬 작업에 사용하는 방법에 대한 연구가 필요할 것이다.

요 약

목적: ML-EM (The maximum likelihood-expectation maximization) 기법은 방출과 검출 과정에 대한 통계학적 모델에 기반한 재구성 알고리즘이다. ML-EM은 결과 영상의 정확성과 유용성에 있어 많은 이점이 있는 반면 반복적인 계산과 방대한 작업량 때문에 CPU(central processing unit)로 처리할 때 상당한 연산시간이 소요되었다. 본 연구에서는 GPU(graphic processing unit)의 병렬 처리 기술을 ML-EM 알고리즘에 적용하여 영상을 재구성하였다. **대상 및 방법:** 엔비디아社의 CUDA 기술을 이용하여 ML-EM 알고리즘의 투사 및 역투사 과정을 병렬화 전략을 구상하였으며 Geforce 9800 GTX+ 그래픽 카드를 이용하여 병렬화 연산을 수행하여 기존의 단일 CPU 기반 연산법과 비교하였다. 각 반복횟수마다 투사 및 역투사 과정에 걸리는 총 지연 시간과 퍼센트 오차(percent error)를 측정하였다. 총 지연 시간에는 RAM과 GPU 메모리 간의 데이터 전송 지연 시간도 포함하였다. **결과:** 모든 반복횟수에 대해 CPU 기반 ML-EM 알고리즘보다 GPU 기반 알고리즘이 더 빠른 성능을 나타내는 것을 확인하였다. 단일 CPU 및 GPU 기반 ML-EM의 32번 반복연산에 있어 각각 3.83초와 0.26초가 걸렸으며 GPU의 병렬연산의 경우 15배 정도의 개선된 성능을 보였다. 반복횟수가 1024까지 증가하였을 경우, CPU와 GPU 기반 알고리즘은 각각 18분과 8초의 연산시간이 걸렸다. GPU 기반 알고리즘이 약 135배 빠른 처리속도를 보였는데 이는 단일 CPU 계산이 특정 반복횟수 이후 나타나는 시간 지연에 따른 것이다. 결과적으로, GPU 기반 계산이 더 작은 편차와 빠른 속도를 보였다. **결론:** ML-EM 알고리즘에 기초한 GPU기반 병렬 계산이 처리속도와 안정성을 더 증진시킴을 확인하였으며 이를 활용해 다른 영상 재구성 알고리즘에도 적용시킬 수 있을 것으로 기대한다.

References

1. Ollinger J, Fessler J. Positron-emission tomography. ***IEEE Signal Processing Magazine*** 1997;14:43-55.
2. Kak A, Slaney M. Principles of computerized tomography imaging. ***PA SIAM***; 2001.
3. Macovski A. Medical imaging. ***Prentice-Hall*** 1983.
4. Lewitt R, Matej S. Overview of methods for image reconstruction from projections in emission computed tomography. ***Proceedings of the IEEE*** 2003;91:1588-611.
5. Lee JS. Nuclear medicine physics. In: Chung J-K, Lee MC, eds. Nuclear Medicine. ***Korea Medical Book Publisher*** 2008. p. 59-64.
6. Lee SJ. Principles of image formation and reconstruction in emission computed tomography. ***J Kor Soc Precision Eng*** 2008; 25:51-62.
7. Shepp L, Vardi Y. Maximum likelihood reconstruction for emission tomography. ***IEEE Trans Med Imag*** 1982;1:113-22.
8. Lange K, Carson R. EM reconstruction algorithms for emission and transmission tomography. ***J Comput Assist Tomogr*** 1984;8: 306-16.
9. Hudson HM, Larkin RS. Accelerated image reconstruction using ordered subsets of projection data. ***IEEE Trans Med Imag*** 1994; 13:601-9.
10. Miller MI, Butler CS. 3-D maximum a posteriori estimation for single photon emission computed tomography on massively-parallel computers. ***IEEE Trans Med Imag*** 1993;12:560-5.
11. Jones MD, Yao R, Bhole CP. Hybrid MPI-OpenMP programming for parallel OSEM PET reconstruction. ***IEEE Trans Nucl Sci*** 2006;53:2752-8.
12. Park SI, Ponce SP, Huang J, Cao Y, Quek F. Low-cost, high-speed computer vision using NVIDIA's CUDA architecture. ***37th IEEE Applied Imagery Pattern Recognition Workshop*** 2008; 1-7.
13. ATI CrossFireX^{TM}. http://ati.amd.com/technology/crossfire
14. NVIDIA SLI. http://www.slizone.com
15. Micikevicius P. Optimizing CUDA. ***NVIDIA Developer Newsletter***; 2008.
16. NVIDIA corp. NVIDIA CUDA Programming Guide ver 2.1. 2008.
17. Kim SM, Lee JS, Lee S-J, Kim KM, Lee DS. Development of regularized expectation maximization algorithms for fan-beam SPECT data. ***Nucl Med Mol Imaging*** 2005;39:464-72.
18. Lee JS, Kim SM, Lee KS, Sim K-S, Rhee JT, Park KS, et al. Compensation methods for non-uniform and imcomplete data sampling in high resolution PET with multiple scintillation crystal layers. ***Nucl Med Mol Imaging*** 2008;42:52-60.
19. Hong SJ, Kwon SI, Ito M, Lee GS, Sim KS, Park KS, et al. Concept verification of three-layer DOI detectors for small animal PET. ***IEEE Trans Nucl Sci*** 2008;55:912-7.
20. Kim SM, Lee JS, Lee MN, Lee JH, Lee CS, Kim CH, et al. Two approaches to implementing projector-backprojector pairs for 3D reconstruction from Compton scattered data. ***Nucl Instr and Meth A*** 2007;571:255-8.
21. Kim SM, Lee JS, Lee MN, Lee JH, Kim JH, Kim CH, et al. A comparative study of subset construction methods in OSEM algorithms using simulated projection data of compton camera. ***Nucl Med Mol Imaging*** 2007;41:234-40.