碩士學位論文

ET에 기반한 실시간 영상복원을 위한 병렬 및 네트워크 컴퓨팅 기법의 구현

濟州大學校 大學院

에너지工學科

朴 淑 熙

2001年 12月

ET에 기반한 실시간 영상복원을 위한 병렬 및 네트워크 컴퓨팅 기법의 구현

指導敎授 李 潤 俊

朴 淑 熙

이 論文을 工學 碩士學位 論文으로 提出함

2001年 12月

朴淑熙의 工學 碩士學位 論文을 認准함

審査委員長 李 政 勳 印

委 員 李 憲 周 印

委 員 朴 在 雨 印

濟州大學校 大學院

2001年 12月

An Implementation of Parallel and Networked Computing Schemes for the Real-Time Image Reconstruction Based on Electrical Tomography

Sook-Hee Park

(Supervised by Professor Yoon-Joon Lee)

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF NUCLEAR AND ENERGY ENGINEERING
GRADUATE SCHOOL
CHEJU NATIONAL UNIVERSITY

2001. 12.

목 차

LIST OF FIGURES

LIST OF TABLES

SUMMARY

This thesis implements and analyzes the parallel and networked computing libraries based on the multiprocessor computer architecture as well as networked computers, aiming at improving the computation speed of ET(Electrical Tomography) system which requires enormous CPU time in reconstructing the unknown internal state of the target object. As an instance of the typical tomography technology, ET partitions the cross-section of the target object into the tiny elements and calculates the resistivity of them with signal values measured at the boundary electrodes surrounding the surface of the object after injecting the predetermined current pattern through the object. The number of elements is determined considering the trade-off between the accuracy of the reconstructed image and the computation time. As the elements become more finer, the number of element increases, and the system can get the better image. However, the reconstruction time increases polynomially with the number of partitioned elements since the procedure consists of a number of time consuming matrix operations such as multiplication, inverse, pseudo inverse, Jacobian and so on. Consequently, the demand for improving computation speed via multiple processor grows indispensably. Moreover, currently released PCs can be stuffed with up to 4 CPUs interconnected to the shared memory while some operating systems enable the application process to benefit from such computer by allocating the threaded job to each CPU, resulting in concurrent processing. In addition, a networked computing or cluster computing environment is commonly available to almost every computer which contains communication protocol and is connected to local or global network. After partitioning the given job(numerical operation), each CPU or computer calculates the partial result independently, and the results are merged via common memory to produce the final result. It is desirable to adopt the commonly used library such as Matlab to boost the

reliability and high computation speed of basic primitive matrix operations. The DLL(Dynamic Link Library) is a good candidate for a Matlab programmer to conveniently call the new library, since the original Matlab code does not need to be changed. The DLL library receives Matlab array represented as mxArray, and converts it into the appropriate C language structure after partitioning the array for the parallel operation. Then the DLL calls Matlab's efficient C language library, which is enabled by creating the definition files as well as including the Matlab library into the Visual C 6.0 project file. Finally, the partial results are merged at the shared memory, so the DLL integrates them to pass the final result to the caller residing in Matlab code. In this procedure, the elimination of the complex Matlab interpreting step, in addition to the parallel programming. According to the implementation described as above, matrix multiplication, inverse, pseudo inverse, and Jacobian are implemented. The first two DLLs speed up the computation by the effect of pure parallel processing. Pseudo inverse can enhance the performance based on the previous parallel procedures if and only if the given matrix is full-rank one, as data dependancy hinders the parallel computing otherwise. The enhancement of Jacobian code owes to eliminating the unnecessary code rather than parallel processing, as the operation contains so much overhead. Also implemented are the network version libraries. However, the speed is not so good as the original code because there is network speed limitation. With the better network interface, the speed up can be expected. The performance of the implemented parallel libraries has been assessed by directly measuring the execution time comparing with the original Matlab code. And the calculating times of matrix multiplications, inverse, and pseudo inverse have been reduced to 59.4 %, 34.8 % and 52 %, respectively. The execution time of Jacobian is dramatically decreased from about 25 second to 100 millisecond. Finally, the library has been applied to the ET procedure. Without any change in the Matlab program, ET can reconstruct the image of a frame with 48 % of the original code. The better performance is expected when the number of CPU increases.

Ⅰ. 서론

ET(Electrical Tomography)는 서로 다른 전기적 특성으로 이루어진 물체 주변에 특수하게 제작된 전극을 여러 개 배치하고, 적절하게 설계된 전류 또는 전압을 주입 또는 인가하여 이에 따른 전압 또는 전류를 물체 경계에서 측정한 후, 이들 측정치를 바탕으로 한 영상복원 알고리즘을 이용하여 물체 내부의 미지의 전기적 특성 분포를 재구성하는 단층 촬영 기법으로서 측정 대상물에 대한 비파괴적, 비침입적인 특성을 갖는다.

영상복원 알고리즘은 대상체 내부에서의 전기적 특성 분포 해석을 위해 대상체 단면적을 작은 계산 격자로 분할하고 각각의 계산 격자에서의 전기적 물성치를 찾아내는 과정을 거친다. 이 때 요소 수를 증가시켜 정밀한 격자를 사용하면 영상복원의 질은 좋아지지만 계산 시간이 증가한다. 영상복원 방법은 정적(static) 복원 방법과 동적(dynamic) 복원 방법으로 분류될 수 있다. 정적 복원 방법은 측정치를 컴퓨터의 메모리에 저장하여 오프라인으로 전기적 특성 분포를 추정하는 것이고 동적 복원 방법은 온라인으로 전기적 특성 분포를 추정하는 과정을 순차적으로 모든 전류패턴에 대해 반복 수행하는 것으로서 시간에 따라 변하는 표적의 저항률 분포를 동적으로 추적할 수 있고 실시간 복원이 가능하다(Kim, K. Y. et. al, 2001). 확장 칼만 필터(Extended Kalman filter)는 동적 복원 방법의 한 예로서 급격히 변하는 동적 영상을 효과적으로 복원할 수 있다.

확장 칼만 필터와 같이 복잡하고 많은 행렬 연산을 수반하는 영상복원 알고리즘을 개발하고 시험하는 상용 소프트웨어 도구로서 Matlab이 가장 일반적으로 사용되고 있다. Matlab은 다양한 행렬 연산과 더불어 그래픽 표현 도구를 지원한다(Mathworks, 1999). 인터프리터(interpreter)에 기반한 명령처리 방식과 복잡한 데이터 표현방식 때문에 전반적인 수행속도가 늦어지는 단점을 갖고 있지만 더하기, 빼기, 곱하기, 역행렬 계산 등 행렬의 기본 연산에 대해서는 상용 소프트웨어 중에서 가장 수행속도가 빠르다. 또한 프로그램의 상당 부분을 행렬 함수로서 처리하고 다양한 수치 계산 응용을 제작하는데 유용한 많은 행렬 함수들을 제공하여 수식처리에 있어 뛰어난 기능

을 갖는다. 따라서 ET 시스템과 같이 많은 행렬 연산을 포함하는 프로그램들은 Matlab으로 개발되어 있으며 그 속도가 상당히 우수하다. 이와 아울러 효율적인 행렬 계산을 수행하는 C 라이브러리를 제공하여 C 언어 프로그램에서 Matlab 함수들을 호출하도록 할 뿐 아니라 새로운 함수들을 작성하여 Matlab의 라이브러리에 추가할 수 있도록 한다.

ET 시스템에서 수행하는 영상복원 알고리즘은 많은 계산시간이 필요하다. 특히 이상적인 실시간 처리를 위해서는 데이터의 수집 주기 이내에 한 프레임(frame)에 대한 영상복원이 완료되어야 하는데 이와 같은 막대한 계산 시간과 기억공간을 요구하는 문제를 풀어야 하는 경우 제한된 메모리의 컴퓨터나 단일 프로세서에서 처리하는 것은 많은 수행시간을 요구하기 때문에 실시간 처리는 불가능하게 된다. 더욱이 Matlab 코드로 구현된 확장 칼만 필터 알고리즘은 각각의 계산 루프마다 주로 행렬에 대한 연산으로 구성되어 있는 FEM, 자코비언(Jacobian) 계산, 측정 갱신, 시간 갱신 등의 단계를 거친다. 특히 자코비언 계산과 측정 갱신 계산 수행시간이 대부분을 차지하며 이를 개선하는 것이 필수적이다. 결국 많은 양의 데이터에 대해 산술 계산을 수행하는 응용의 수행속도를 개선하려면 복원 알고리즘의 효율성을 극대화시키거나 병렬 컴퓨팅 혹은 네트워크 컴퓨팅 등을 이용할 수 있다.

본 논문에서는 ET와 같이 많은 양의 데이터에 대해 산술 계산을 수행하는 응용의 수행속도를 개선하기 위하여 이중(dual) CPU PC 상에서 Matlab의 기본연산, 즉 행렬 곱하기, 역행렬 계산, 의사 역행렬(pseudo inverse) 계산 등을 병렬로 수행하는 라이브러리 프로그램을 구현하고 그 성능을 측정한다. 구현된 라이브러리는 행렬의 곱하기, 역행렬 계산, 의사 역행렬 계산 등 기본적인 행렬 연산에 대해 각 CPU에서 수행될 쓰레드(thread)를 생성하고 이 쓰레드에 분할 행렬을 인자로 넘겨줌으로써 병렬계산을 실행하도록 한 후 부분 결과를 합성하여 최종적인 결과를 산출하게 된다. 이와 아울러 분산된 컴퓨팅 환경에서 병렬 처리 기법을 도입하여 네트워크로 연결된 두 컴퓨터에 대하여 각 컴퓨터에서 분할 행렬에 대한 연산을 한 후 서버 컴퓨터에서 부분 결과를 합성하여 최종적인 결과를 산출하는 프로그램을 구현하고 그 성능을 측정한다. 확장 칼만 필터와 같이 막대한 계산량을 갖는 영상복원 응용이 PC 상의 병렬 처리에 의해 수행속도가 현저히 개선이 됨을 보이고 이에 따른 실시간 복원의 가능성을 제시한다.

본 논문은 다음과 같이 구성된다. 1장에서 논문에서 해결하고자 하는 문제에 대해 소개한 후 2장에서는 수치 계산, 병렬 계산, 네트워크 계산, 실시간 계산 등 영상복원의 속도를 개선하기 위한 다양한 계산 환경에 대한 관련 연구를 소개한다. 3장에서는 영상복원의 속도에 직접적으로 영향을 주는 자코비언 계산, 행렬의 곱하기, 역행렬, 의사 역행렬 등의 행렬 계산에 대해 병렬 계산 방법을 제시하고 이를 이중 PC에서 Matlab 함수와 도구를 이용하여 DLL(Dynamic Link Library)을 구현한다. 4장에서는 구현된 DLL에 대해 Matlab 명령으로 호출하여 병렬 처리에 의한 속도 향상도를 측정한 결과와 이 성능의 향상이 확장 칼만 필터의 영상복원 속도 개선에 미치는 영향을 보인다. 5장에서는 논문을 요약하고 결론을 도출하며 향후의 과제를 제시한다.

Ⅱ. 관련 연구 및 배경

1. ET 영상복원

ET는 대상이 되는 물체에 인위적인 전기신호를 주입한 후 그 물체에서 나오는 신호를 조작하여 물체의 내부를 영상화하는 단층 촬영 기법이다. ET가 갖고 있는 가장 큰 매력은 다른 단층 촬영 기법과는 달리 시스템이 간단하고 저렴하며 또한 그 활용 분야가 매우 다양하다는 점이다. 특히 시스템의 구성을 통해 3차원으로도 확장시킬 수 있으며 효율적인 알고리즘의 구현에 의해 동적 영상도 얻을 수 있게 된다.

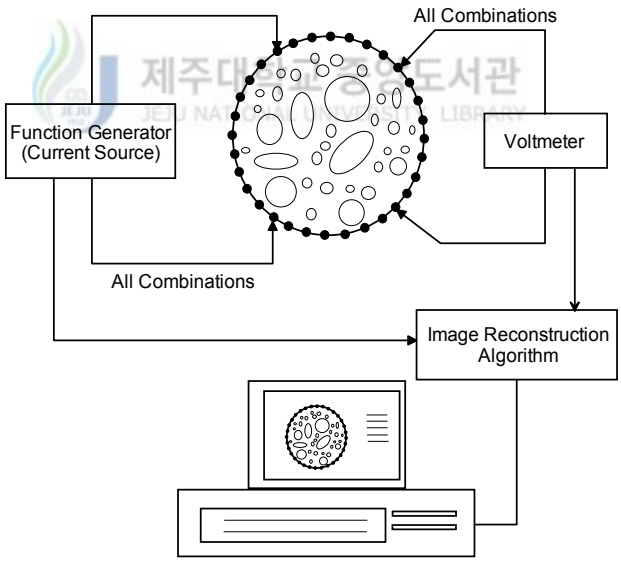

Fig. 1. Schematic diagram of ET system

ET의 기본적 원리는 Fig. 1에서 보는 바와 같이 미지의 전기적 특성(저항률 혹은 유전율) 분포를 갖는 물체 주위에 특수하게 제작된 전극을 여러 개 배치하여, 적절하게 설계된 전류 혹은 전압신호를 주입한 후 주입된 신호에 따라 물체 내부의 전기적 특

성에 의해 형성된 전장의 형태 혹은 유전율 차이에 근거한 분극의 정도에 따라 상이한 출력신호가 전극에 나타나게 되는데 이 출력신호(전압 혹은 전류)를 물체 표면 혹은 경계에서 측정하고 이 측정값을 사용하여 맥스웰 방정식에 기초한 영상복원 알고리즘을 통해 물체 내부의 미지의 전기적 특성 분포를 재구성하는 것이다. 전기적 특성 분포는 이상유동장에서 각 상의 분포로 대체하여 생각할 수 있다.

영상복원 알고리즘은 대상체 내부에서의 전기적 특성 분포 해석을 위해 대상체 단면적을 작은 계산 격자로 분할하고 각각의 계산 격자에서의 전기적 물성치를 찾아내는 과정을 거친다. 이 때 요소 수를 증가시켜 정밀한 격자를 사용하면 영상복원의 질은 좋아지지만 계산 시간이 증가한다. ET의 영상복원 방법은 정적 복원 방법과 동적 복원 방법으로 분류될 수 있다. 정적 복원 방법은 경계면에 주입한 전류패턴에 의해 유기된 전압 값을 측정한 후 컴퓨터의 메모리에 저장하여 오프라인으로 저항률의 분포를 추정하는 것으로서 추정 정확도는 높지만 영상복원이 느리다. 반면 동적 복원 방법은 경계면에 한 패턴의 전류를 주입하고 유기되는 전압 값을 이용하여 온라인으로 저항률의 분포를 추정하는 과정을 순차적으로 모든 전류패턴에 대해 반복 수행하는 것으로서 시간에 따라 변하는 표적의 저항률 분포를 동적으로 추적할 수 있고 실시간 복원이 가능하다. 확장 칼만 필터는 동적 복원 방법의 한 예로서 각 전류 패턴에 의해 추정되는 저항률 분포에 대해 자코비언 행렬을 계산하여 갱신시키는 과정의 반복 연산으로 구성되기 때문에 급격히 변하는 동적 영상을 효과적으로 복원할 수 있다.

ET는 시스템 구현 시에 하드웨어 비용이 비교적 저렴하고, 측정 대상물에 대한 비파괴적, 비침입적인 특성을 가지고 있으며, 특히 측정 속도가 매우 빠른 특성으로 인하여, 화공학, 지질학 및 재료공학 등에서 모니터링 도구로 주목받고 있으며, X-ray 및 MRI 단층촬영법에 비해 아직 복원된 영상의 공간해상도는 떨어지지만, 시간해상도가 뛰어나고 인체에 대한 안정성이 보장되므로 의공학 분야의 보조장비로도 사용되고 있다. 특히, 이상유동장 또는 다상유동장에서 공정 내부의 동적변화를 실시간 또는 준 실시간으로 모니터링하기 위한 도구로 주목을 받고 있으며, 앞으로 그 이용범위가 확대될 것으로 판단된다.

2. 수치 계산(Numerical Computing)

수치 계산이라 함은 막대한 양의 데이터를 처리하기 위하여 데이터를 컴퓨터에 효율적으로 저장하고 저장된 데이터에 대해 컴퓨터로 다양한 연산을 수행하도록 함을 의미한다. 저장되는 형태는 주로 행렬에 기반하고 있으며 이에 대한 연산에는 더하기를 포함한 사칙연산은 물론 LU분할, 의사 역행렬, 정렬 등 다양한 연산들이 포함된다. 에너지공학과 같이 막대한 양의 데이터를 처리하는 분야의 응용들을 지원하기 위하여 많은 소프트웨어 도구들이 학계 혹은 업체에서 연구용 상업용으로 개발된 바 있으며 이들은 각기 나름대로의 프로그래밍 언어로 구현되어 있다. 더욱이 개인용 컴퓨터의 성능이 개선됨에 따라 일반 사용자들도 많은 수치 응용을 필요로 하고 있다.

수치 계산 소프트웨어는 많은 데이터의 양 때문에 그 수행속도와 신뢰성이 중요한데 이는 계산 알고리즘과 상용 실험에 의해 결정된다. 최근에는 웹 기술의 발달과 더불어 Java에서도 고속의 수치 계산을 지원하려는 연구가 진행되고 있다(Boisvert et. al. 2001).

다양한 수치 계산 소프트웨어에 대해 장단점을 판단하는 것도 상당히 중요한 연구분야로서 math.nist.gov에는 다양한 소프트웨어 도구를 평가하여 그 결과를 나열하고 있다. Matlab은 상용 소프트웨어 도구로서 MS-DOS, MS-Windows 운영체제에 기반한 개인용 PC나 UNIX에 기반한 워크스테이션 등에서 수행되며 가장 효율적이고 일반적인 수치 계산 소프트웨어 도구로 알려져 있다. 본 논문에서 대상으로 하는 ET 시스템의 영상복원 기법들도 Matlab을 이용하여 구현되어 있으며 Matlab은 다음과 같은 특징을 갖고 있다.

- 간단한 프로그래밍 인터페이스
- 정수, 실수, 복소수 값들의 일관적인 처리
- 다양한 수학연산 라이브러리
- GUI(Graphic User Interface) 함수를 포함하는 풍부한 그래픽 도구
- 기존의 프로그래밍 언어와의 결합 가능성
- Matlab 프로그램의 이식성

Matlab은 600 개이상의 수학적, 통계적, 공학적 함수를 보유하여 사용자에게 빠르고 정확하면서도 신뢰할 수 있는 고성능의 수학적 계산을 지원함으로서 수학적 작업과 데이터 해석을 가능하게 한다. 그 예로 수치 해석, 통계학, 선형 대수, 행렬 연산, 미적분학, 행렬 인수분해, 다항식의 근 및 평가, 희소행렬, 무한 급수, 라플라스 및 푸리에 변환 등을 쉽게 구하여 사용할 수 있게 하고 알고리즘 개발, 과학 및 공학의 모델링, 신호 처리, 제어 계측, 인공지능의 활용 및 다양한 그래픽 기능을 제공한다.

3. 병렬 컴퓨팅

병렬 컴퓨팅이란 단일 CPU 용량 한계를 극복하기 위해 다수의 CPU를 동시에 이용하여 보다 효율적으로 문제를 풀고자 하는 방법을 말한다(Akl, 1989). CPU들이 협력하는 방식은 작업의 목적에 따라 병렬 계산에 의한 계산 속도 향상, 일부 요소의 고장에도 정상 동작할 수 있도록 하는 결함 허용(fault tolerant) 등 다양하게 적용될 수 있다. 본 논문에서의 병렬 컴퓨팅은 협력 계산에 의한 영상복원 속도의 향상을 목적으로 한다.

다수의 CPU들이 협력하려면 이들이 서로 통신할 수 있는 채널이 필요하다. 즉 각 CPU가 자신에게 할당된 작업을 수행하고 최종적으로 이를 수합하여야 하는데 할당과 수합 과정은 CPU들간 통신이 수반된다. 이 채널은 상호연결망(interconnection network)이라 불리우며 상호연결망의 형태에 따라 공유 메모리, 파이프라인, 벡터 프로세서 및 어레이 프로세서 등의 구조가 결정된다. 또 각 CPU는 자신의 국지 메모리를 갖고 있어서 명령어와 데이터를 이 메모리에서 습득하며 경우에 따라 상호연결망을 통해 다른 프로세서의 메모리를 읽거나 쓸 수 있다. 상호연결망의 성능에 따라 협력 계산의 필수적인 요소인 데이터 이동의 수행속도가 결정되므로 효율적인 상호연결망의 구성이 필요하다. 이 상호연결망은 일반적으로 고가이며 네트워크보다 고속으로 데이터 교환이 이루어진다.

기존의 단일 CPU 상에서의 알고리즘들은 컴퓨터의 병렬 구조를 효율적으로 이용하

지 못하고 있기 때문에 행렬 곱 등 다양한 연산에 대해 각 병렬 컴퓨터 구조에 적합한 병렬 계산 알고리즘들이 개발되어 왔다(Gallivan, 1990). 이러한 알고리즘들은 작업과 데이터를 컴퓨터 구조에 맞게 분할하고 각 CPU에게 계산을 시킨 다음 결과를 산출하는데 한 시스템에 여러 CPU가 독립적으로 수행되고 이들이 변수들을 공유할 수 있기 때문에 복잡한 동기화(synchronization) 문제가 야기된다. 동기화란 다수의 프로세서들이 어떤 자원에 동시에 접근할 때 자원의 성격에 따라 순서를 결정한다든지 하나의 CPU가 자원을 갖고 있을 때는 다른 CPU들이 접근할 수 없도록 하는 방식을 말하며 운영체제에서 임계 영역(critical section)이나 Lock과 같은 기법을 제공하고 있다.

현재 출시되는 PC들은 최대 4 개까지의 CPU를 연결할 수 있으며 각 CPU가 메모리를 공유한다. 즉, 한 CPU가 공유 변수의 내용을 갱신하면 바로 다른 CPU가 이를 읽을 수 있다. 또 시스템 버스가 상호연결망으로 동작하여 두 CPU로부터의 공유 메모리 접근을 중재한다. 작업의 분할이나 동기화는 운영체제가 지원하는데 작업을 분할하는 과정은 응용 프로그램을 쓰레드로 나누면 NT 이상의 운영체제는 쓰레드를 각 CPU에 할당하여 독립적으로 수행하도록 한다.

쓰레드란 프로그램의 흐름을 말하는데 보통 프로그램들은 오직 하나의 흐름을 갖는 단일 쓰레드 구조이다. 다중 쓰레드 구조는 한 프로세스 내에서 여러 개의 쓰레드가 메모리 영역, 네트워크 자원, 디스크 자원 등의 대부분의 자원을 공유하며 각각의 독립적인 작업을 수행할 수 있다. 다른 쓰레드와 자원을 공유함으로써 새로운 자원의 생성이 필요가 없으며 공유되는 자원의 효과적인 관리가 가능해 진다. 다중 쓰레드는 프로그램 상에서 여러 개의 부함수(subroutine)가 병렬로 수행하는 것과 같은 효과를 나타내며 한 프로그램에서 정의된 전역 변수들은 모든 쓰레드에 의해 공유된다. 쓰레드는 프로세스에 비해 그 생성 시간이 짧으며 문맥 교환 시간이 빠르다. 결국 쓰레드는 병렬 프로그램의 실행 단위로서 하나의 프로그램에서 여러 개의 쓰레드를 생성하여 독립적으로 실행시킬 수 있다. 또 운영체제는 임계영역과 같은 함수들을 제공하여 공유 변수에 대한 접근을 순서화하는 동기화 기능도 제공한다. 다중 쓰레드 프로그램은 일반적으로 수행시간, 사용자 응답성, 프로그램 구조 중 적어도 하나 이상이 단일 쓰레드 프로그램보다 우수하다.

4. 네트워크 컴퓨팅

네트워크 컴퓨팅은 네트워크 지역망으로 연결된 컴퓨터들이 서로 협력하여 주어진 작업을 수행한다. 병렬 컴퓨팅과는 달리 데이터 교환이 네트워크를 통해 수행되므로 그 속도가 떨어진다. 그러나 네트워크는 상호연결망에 비해 더 많은 컴퓨터들을 연결할 수 있으며 현재 ATM과 같은 네트워크 기술의 발달은 고속의 네트워크 컴퓨팅을 가능하게 한다. 네트워크 컴퓨팅 구조는 지역망으로 연결된 다수의 컴퓨터들이 독립적으로 수행되는 클러스터 구조를 갖는다.

클러스터 시스템은 고성능 계산에 있어 넓은 응용에 사용될 수 있도록 하드웨어 및 소프트웨어 기술이 개발되었고 동시대의 슈퍼컴퓨터에 비해 가격면에서의 이점을 갖고 출시되었다. 시스템구조의 모든 매개변수 즉 노드의 수와 노드당 메모리 용량 및 프로세스의 수 그리고 서로 연결된 위상은 기본 시스템에서 추가적인 비용을 초래하지 않고 쉽게 수정시킬 수 있다. 클러스터는 또한 기술향상에 빠르게 대응하여 프로세스, 메모리, 디스크 그리고 네트워크를 포함한 새로운 장치들의 사용을 가능하게 한다.

클러스터를 구성하는 가장 중요한 요소는 계산을 수행하는 노드와 노드간에 데이터 통신을 지원하는 네트워크이다. 클러스터의 노드는 독립적인 수행이 가능한 것으로 시스템 컴퓨팅과 데이터 저장 능력을 제공하며 프로세서, 메모리, 보조기억장치, 외부 인터페이스와 같은 여러 가지 부분 시스템을 통합한 형태이다. 클러스터의 네트워크는 개별적인 노드들 사이에서 데이터 패킷이 전송될 수 있도록 한다. 네트워크의 성능을 측정하는 중요한 두 요소는 대역폭(bandwidth)과 지연시간(latency)이다. 네트워크의 대역폭은 하나의 채널을 통해 단위 시간 동안 전송될 수 있는 비트 수에 의해 결정된다. 예를 들어, 네트워크의 대역폭이 10Mbps라면 매초 1천만 비트가 전달될 수 있다. 지연시간은 매체를 타고 한 비트가 네트워크의 한 쪽에서 다른 쪽으로 전송되는데 걸리는 시간을 말한다. 그러나 실제 측정된 값은 패킷 사이즈, 충돌 그리고 소프트웨어 낭비시간과 같은 부수적 효과에 의존하여 단일 시스템 내에서도 크게 변화할 것이다. 클러스터 환경에서 데이터를 교환하기 위한 도구는 OSI 7계층 중 트랜

스포트 상에서 동작하는 소켓(Socket) 라이브러리와 아울러 MPI(Message Passing Interface)등이 사용되고있다.

5. 실시간 처리

실시간 시스템(real-time system)은 시스템의 정확성이 계산 결과의 정확성뿐만 아니라 결과가 산출되는 시간에 의해 결정되는 시스템을 의미한다(Liu, 2000). 실시간 시스템에서의 작업들은 종료시한(deadline) 이내에 완료되어야 한다는 시간 제약조건(time constraint)을 가지며 종료시한 이후에 산출된 결과는 그 의미가 없거나 시스템에 치명적인 영향을 초래할 수 있다.

실시간 시스템은 그 결과의 중요도에 의해 경성(hard) 실시간 시스템과 연성(soft) 실시간 시스템으로 구분되는데 경성 실시간 시스템은 모든 작업이 종료시한 이내에 완료되어야 하는 시스템으로서 일부 작업의 종료시한 초과는 전체 시스템의 동작에 치명적인 영향을 주게 된다. 경성 실시간 시스템의 예로서는 공장자동화, 원자력 발전소 제어, 우주선 항법 시스템 등이 속한다. 반면 연성 실시간 시스템은 일부 작업의 종료시한 초과를 허용하는 시스템으로 은행에서의 트랜잭션 시스템이나 멀티미디어 재생 시스템이 이에 속한다.

경성 실시간 시스템에서는 주어진 작업 집합에 대한 사전 분석이 오프라인 시에 수행되는데 각 작업은 일반적으로 주기와 수행시간과 같은 시간 특성을 갖는다. 이를 바탕으로 각 작업에 대해 분석하여 시스템에서 주어진 작업 집합이 종료시한 내에 완료될 수 있는지 판단하고 그 순서를 결정하는 스케쥴링이 완료되고 온라인 시에는 이 스케쥴에 따라 작업을 수행시킨다. 이러한 과정은 rate monotonic과 같은 스케쥴 및 보장 기법을 따른다. 반면 연성 시스템은 동적으로 생성되는 실시간 작업에 대해 가능한 많은 작업들이 종료시한을 만족시킬 수 있도록 스케쥴을 동적으로 수행한다.

이상적인 ET 시스템은 동적으로 물체 내부의 상태를 추적하여야 한다. 이를 위해서는 한 프레임의 전류신호가 주입되고 다음 신호가 주입되기 이전에 한 프레임에 대

한 영상복원이 완료되어야 한다. 그러나 이 프레임 주입 시간 간격은 수 msec인데 반해 현재 PC에 기반한 영상복원 알고리즘의 수행은 수십 sec 이상 소요된다. 따라서 현재의 ET 시스템은 일단 데이터를 수집한 후 오프라인으로 영상복원을 수행하는 단계에 머물러 있다. 실시간으로 영상을 복원하려면 복원 시간을 대폭 단축하여야 하는데 이를 위해 병렬 계산, DSP(Digital Signal Processor), 효율적인 알고리즘을 사용할 수 있다. 또 물체 내부의 상태가 자주 변경되지 않는 응용에 대해서는 주입 시간 간격을 늘려 복원을 하는데 필요한 여유시간을 늘릴 수 있다. 본 논문에서는 실시간 영상복원을 위한 전 단계로서 병렬 계산이나 네트워크 계산에 의해 프로그램의 수행속도를 대폭 감소시키고자 한다.

Ⅲ. 속도 향상 기법

1. 병렬 컴퓨팅 환경

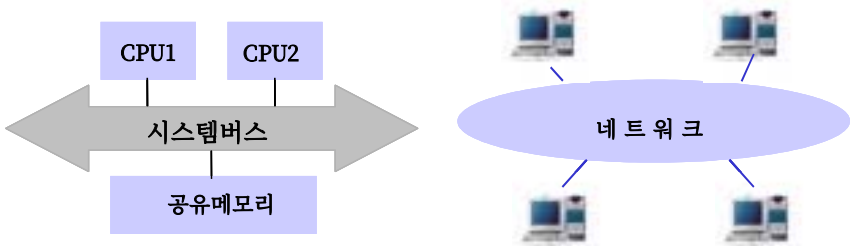

Fig. 2. Architecture of parallel and networked computing

Fig. 2에서 보는 바와 같이 병렬 컴퓨팅은 한 시스템에 다수의 CPU들이 연결된 구조에서 공유된 메모리에 자료를 저장하고 각 CPU들이 협력하는 구조로서 통신 오버헤드는 제거되지만 시스템의 가격이 비싸다. 그러나 Windows 운영체제나 IBM 계열의 컴퓨터에서 이중 혹은 다중 CPU를 탑재한 PC들이 출시되고 있고 최근의 PC들은 응용의 계산 속도 향상 요구를 만족시키기 위하여 최대 4개까지의 CPU를 탑재하여 병렬 계산을 지원할 뿐 아니라 Windows NT 이상의 운영체제에서는 프로그래머가 지정한 각 쓰레드를 각 CPU에게 할당하여 계산 속도의 향상을 기하는 등 병렬 컴퓨팅 환경은 이제 일반 사용자들에게도 널리 확대되고 있다(Hinsberg, 1999). 반면 네트워크 컴퓨팅은 네트워크로 연결된 여러 컴퓨터들로 하여금 통신에 의해 협력 계산을 지원하는 것으로서 데이터 교환시 통신에 따르는 오버헤드를 극복하기 위해 네트워크의 실시간 메시지 전송 기능이 필수적일 뿐 아니라 통신 오류를 최소화하기 위해 링크 수준의 오류제어 기능을 제공하는 것이 바람직하다.

본 절에서는 Fig. 2의 구조에 기반한 행렬 계산 라이브러리를 구현하여 프로그램의 수행속도를 개선하고자 한다. 대상 연산은 행렬 곱하기, 역행렬, 의사 역행렬, 자코비언 등이며 각각 주어진 연산을 두 부분으로 분할하여 협력계산을 수행한다.

2. 구현

2.1 자료구조

수행속도를 개선하기 위해서 Matlab에서 제공하는 Matlab compiler를 사용하여 C 언어로 작성한 프로그램으로 Matlab에서 사용할 수 있는 DLL을 생성할 수 있다. 이러한 Matlab compiler로 인하여 C 언어의 수행속도의 장점을 이용할 수도 있을 뿐 아니라 Matlab 프로그래머로 하여금 쉽게 새로운 함수를 이용할 수 있게 한다. 더욱이 행렬 연산에 있어서 다중 CPU 구조를 이용하여 코드를 작성한다면 병렬 계산에 의해 빠른 수행속도를 제공할 수 있다.

C 언어로 작성한 프로그램을 Matlab에서 사용할 수 있게 하려면, C 언어로 작성한 프로그램을 mex 파일로 바꾸어 주어야 한다. mex 파일이란 Matlab 인터프리터가 자동적으로 적재하고 Matlab 환경에서 실행할 수 있는 동적 링크 라이브러리로서 Windows 환경에서는 DLL 파일로 바뀐다. Matlab 워크스페이스에서 다음과 같이 수행하면 DLL 파일이 생성되고 다른 Matlab 함수처럼 사용할 수 있게 된다.

> mex filename.c

Matlab에서 사용하는 여러 가지 데이터 타입을 C 언어로 작성한 DLL 파일이 받아들일 수 있도록 C 파일에 mexFunction을 Fig. 3과 같은 형태로 프로그램을 작성하여야 한다. 이 때 mex.h 헤더 파일을 반드시 포함시켜주어야 한다.

```
// mex-file로 바꾸고자 하는 C subroutine
    void CSubRoutine(double *, …)
    {
            처리부분
    }

//Gateway function part
    void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
    {
            변수 선언
            Matlab에서 전달된 입력 매개변수에 설정된 데이터의 크기 분석
            : mxGetM, mxGetN
            Matlab에서 전달된 입력 매개변수의 데이터에 대한 포인터 설정
            : mxGetPr, mxGetPi
            Matlab으로 보낼 줄 mxArray output 포인터의 생성과 설정
            : mxCreateDoubleMatrix

            C subroutine의 호출
    }
```

Fig. 3. Architecture of mex-file

Matlab에서 제공하는 입력 매개변수와 출력 매개변수에 대한 모든 정보는 mxArray 구조체에 저장되고 입력 매개변수는 C 언어에서 mxArray 구조체 형태로 선언된 *prhs[] pointer에 의해서 읽어 들일 수 있고 출력 매개변수는 *plhs[] pointer에 의해서 읽어 들일 수 있다. mxArray는 Matlab에서 제공되는 매개변수에 대한 데이터 타입의 종류, 차원, 실질적인 데이터, 실수인지 허수인지에 대한 정보를 저장하는 구조

체로서 Fig. 4에서 보는 바와 같이 일반적인 크기의 행렬을 저장하기 위하여 행의 수, 열의 수 및 데이터에 대한 포인터 등을 갖고 있고 데이터 영역에는 행렬의 각 원소들이 열 우선(column major) 방식으로 저장되어 있다.

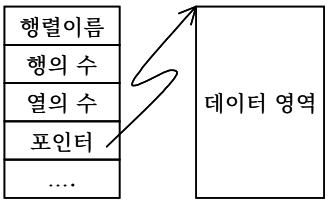

Fig. 4. Matlab mxArray structures

결국 C 언어에서 선언한 mxArray pointer 변수가 Matlab에서 사용하는 입·출력 매개 변수를 지정한다. int nrhs와 int nlhs는 각각 입력 매개변수의 개수와 출력 매개변수의 개수를 의미한다. 이와 같이 void mexFunction()을 만들면 Matlab 인터프리터와 C 언어로 작성된 DLL이 서로 데이터를 교환할 수 있다. 이 과정에서 주의할 것은 Matlab은 데이터 영역에 행렬의 원소들을 열 우선으로 저장하는데 반해 C 언어에서는 행 우선으로 처리하는 점인데 결국 C 프로그램 작성자가 이를 유의하여야한다.

2.2 쓰레드 구조

병렬 컴퓨팅을 하려면 작업을 분할하고 각각 CPU로 하여금 연산을 수행시키고 최종적으로 결과를 수합한다. Fig. 5에서 보는 바와 같이 작업을 분할하는 과정은 응용프로그램에서 쓰레드로 나누면 운영체제가 각 CPU에 할당한다. 쓰레드 프로그램에서 쓰레드는 병렬 프로그램의 실행 단위로서 프로그램은 다수의 프로그램 흐름을 갖게 된다. 분할 과정은 mxArray의 포인터를 변경하여 낭비시간이 거의 없이 수행될 수 있다. 이중 CPU PC는 각 CPU가 버스를 통해 공유 메모리에 접근할 수 있는 다중처리기(multi-processor)이므로 쓰레드들은 전역 변수들을 공유할 수 있어서 쓰레드간 통신에 필요한 오버헤드는 최소화된다. 그러나 쓰레드 생성과 동기화에 따르는 낭비시간은 불가피하게 발생하므로 계산이 단순한 행렬의 더하기와 빼기 등은 이 낭비시

간 때문에 병렬 수행을 하더라도 속도가 개선되지 않는다.

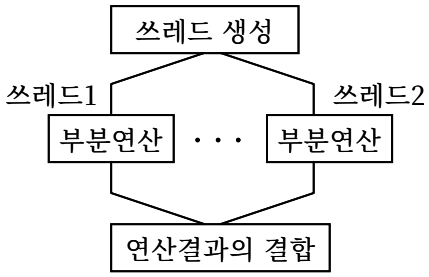

Fig. 5. Theaded computation

응용의 수행속도를 개선하려면 행렬의 곱하기와 역행렬 계산 시간을 감소시켜야 하는데 희소행렬인 경우에는 Matlab에서의 수행시간과 큰 차이가 없지만 일반 행렬인 경우에는 병렬 수행의 장점을 이용하여야 한다. 따라서 DLL 파일에서는 희소행렬에 대한 연산은 Matlab C Math 라이브러리에서 제공하는 함수들을 호출하고 일반 행렬에 대한 연산은 병렬적으로 수행되도록 함으로써 수행시간을 개선한다. Matlab C Math 라이브러리 함수로서 mlfMtimes, mlfInv 등을 이용하는데 메모리 관리에 있어서 오버헤드를 최소화하기 위해 자동 문맥 저장 기능을 사용하지 않고 프로그램에서 직접 mxDestroyArray를 호출하여 명시적으로 메모리를 할당 및 해제한다. 또 이중 CPU를 이용하기 위해서는 쓰레드 유틸리티를 사용하여야 하는데 Windows NT 이상의 운영체제에서는 쓰레드 단위로 CPU에게 할당하여 자동으로 병렬 수행되도록 한다. 이중 CPU를 탑재한 PC라도 쓰레드 프로그램을 작성하지 않으면 하나의 CPU에 의해 수행된다. 쓰레드 프로그램을 작성하는데 있어서 windows.h 헤더 파일에 정의되어 있는 CreateThread 등의 함수를 사용한다(Cohen, 1998). 전형적인 쓰레드 프로그램의 구조는 Fig. 6에 나타난 바와 같이 메인 부분에서 두 개의 쓰레드를 생성한 후 각 쓰레드가 종료할 때까지 기다린다. 쓰레드는 서로 자료 구조를 공유하므로 상호 통신의 오버헤드가 없으며 메인 부분은 쓰레드가 종료한 후 처리 결과를 읽어야 한다.

```
void thread_proc1( )                    void thread_proc2( )
{                                       {
    // 실제 계산 루틴                          // 실제 계산 루틴
}                                       }

void ThreadMain(Parameters) {
mulParam par[2];  HANDLE hThread[2];  DWORD    dwThreadId[2];

//  쓰레드 함수에 대한 인자의 설정
hThread[0] = CreateThread(0, 500, (LPTHREAD_START_ROUTINE)
                    thread_proc1, &par[0], 0, &dwThreadId[0]);
hThread[1] = CreateThread(0, 500, (LPTHREAD_START_ROUTINE)
                    thread_proc2, &par[1], 0, &dwThreadId[1]);
WaitForSingleObject(hThread[0], -1);          // 쓰레드의 종료를 기다림
WaitForSingleObject(hThread[1], -1);          // 쓰레드의 종료를 기다림
return;
 }
```


Fig. 6. Skeleton of thread programs

한 프로세스 내의 쓰레드들은 하나의 주소 공간내에서 동작하기 때문에 데이터의 공유는 쉬운 편이다. 하지만 여러 쓰레드를 병렬적으로 진행할 경우의 동기화 문제는 매우 복잡할 수 있는 단점도 있다. WaitForSingleObject 함수는 쓰레드간의 동기화에 사용되는데 이것이 호출되면 어떤 사건이 일어날 때까지 리턴하지 않고 기다린다. 여기서는 쓰레드가 종료되기를 기다리기 위해 쓰레드 핸들을 인자로 WaitForSingleObject 함수를 호출한다.

2.3 곱하기 계산

A*B와 같은 행렬의 곱하기를 병렬로 수행하려면 Fig. 7에서 보는 바와 같이 먼저 한 행렬을 둘로 분할하여야 한다. B 행렬을 분할하는 이유는 Matlab에서 행렬을 표현할 때 열우선으로 각 항목을 저장하기 때문이며 연속적인 항목을 나누는 것이 효율적이다. 분할 후 각 쓰레드는 부분곱을 수행하며 각 쓰레드는 mlfMtimes라는 Matlab C Math 라이브러리 함수를 사용한다.

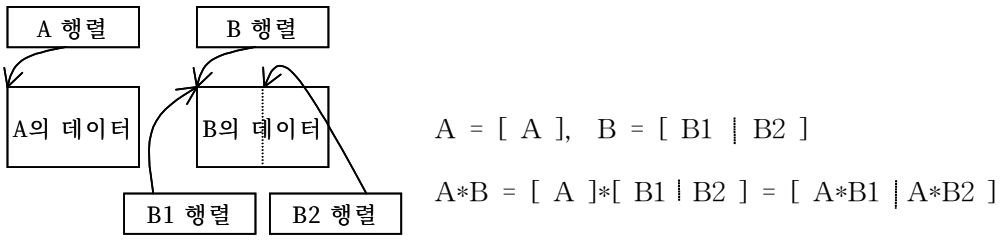

Fig. 7. Submatrix and thread for parallel multiplication

이 함수의 결과는 mxArray 타입의 임시 구조체에 저장되는데 쓰레드들이 종료되면 이를 결합하여 새로운 구조체를 생성하여야 한다. 쓰레드의 종료를 기다리기 위해 윈도우에서 제공하는 WaitforSingleObject 동기화 함수를 이용하여 쓰레드의 종료를 기다린다. 결과를 합성하는 과정에서는 Matlab 라이브러리의 오버헤드를 최소화하기 위하여 memcpy와 같은 C 언어 함수를 사용하여 고속의 메모리 복사를 수행한다. 물론 Matlab 라이브러리를 사용하지 않고 C 함수를 이용하여 부분 곱을 수행한다면 메모리를 프로그래머가 원하는 대로 할당할 수 있지만 곱하기의 수행속도가 현저히 저하된다.

2.4 역행렬 계산

행렬의 곱하기와 더불어 역행렬을 계산하는 프로그램도 이중 CPU 구조를 이용할 수 있는데 역시 행렬을 분할하여 역행렬을 계산한다. 분할함에 있어서 병렬 수행의 장점을 극대화하려면 연속된 공간의 분할 즉, 사각형 형태의 분할이 타당하며 대각선 분할은 수행속도의 저하를 초래할 수 있다. 더욱이 두 CPU가 메모리를 공유하므로 두 쓰레드가 메모리를 공유하기 때문에 데이터의 의존성을 효율적으로 처리할 수 있다. 따라서 Fig. 8과 같은 분할 방식을 사용하여 위에서 구현된 병렬 곱하기 코드를 사용한다면 병렬 역행렬 계산을 구현할 수 있다. ET 시스템에서 주로 사용되는 808×808 차원의 행렬에 대해 역행렬을 한번 계산하는데 그림은 404×404 차원의 행렬 4 개를 생성하고 이들에 대한 역행렬 계산과 곱하기로 전체 역행렬 계산을 대치한다.

$$A = \left[\begin{array}{c|c} A11 & A12 \\ \hline A21 & A22 \end{array}\right] \quad A^{-1} = \left[\begin{array}{c|c} G & -G*A12*A22^{-1} \\ \hline -A22^{-1}*A21*G & A22^{-1}+A22^{-1}*A21*G*A12*A22^{-1} \end{array}\right]$$

$$G = (A11-A12*A22^{-1}*A21)^{-1}$$

Fig. 8. Matrix decomposition for parallel inverse

Fig. 8의 분할 방식은 하나의 역행렬 계산을 행렬의 차원이 반으로 감소한 행렬에 대해 7 번의 역행렬, 10 번의 곱하기 및 2 번의 더하기 혹은 빼기 연산으로 구성되는데 중복된 계산이 많으므로 연산의 수는 줄어든다. A22에 대한 역행렬을 계산한다면 임시적으로 생성된 행렬을 각 쓰레드가 공유하므로 중복 계산을 감소시킬 수 있다. 즉, $A22^{-1}$를 한번 계산해 놓으면 이후 계속 사용할 수 있으며 $A22^{-1}*A21*G$의 결과도 추후 사용할 수 있는 장점이 있다. 따라서 두 번의 역행렬 계산과 5 번의 곱하기로 줄어든다. 행렬 연산의 수행속도는 행렬의 원소 수에 크게 영향을 받는데 행렬의 차원이 반감되었기 때문에 분할의 효과만으로도 속도의 향상을 기할 수 있다. 부분 행렬의 역행렬 계산은 Matlab의 mlfInv 함수를 사용하는 반면 곱하기는 앞서 2.3에서 구현된 병렬 곱하기를 이용하여 계산 속도를 향상시킨다.

2.5 의사 역행렬 계산

의사 역행렬은 행렬 A가 정방행렬이 아니어서 역행렬을 갖지 않을 때, A의 일반적인 의미의 역행렬로서 선형방정식 Ax=y에 대한 해 $x=A^{+}y$를 구하는데 중요한 역할을 한다. 행렬 A에 대해 다음을 만족하는 유일한 행렬 M이 존재한다는 것이 무어(Moore)에 의해 주장되고, 펜로즈(Penrose)에 의해 존재성과 유일성이 증명되었다. 여기서 M은 무어-펜로즈(Moore-Penrose) 역행렬이라고 불려진다.

(ⅰ) AMA = A 이다
(ⅱ) MAM = M 이다
(ⅲ) AM은 대칭행렬이다
(ⅳ) MA는 대칭행렬이다

A의 의사 역행렬 M을 구하기 위해 Fig. 9에서와 같이 A를 분할한다. 차원이 p×q인 행렬 A는 계수 r개의 행과 열을 갖는 행렬의 선형결합으로 즉, $A_{p\times q} = K_{p\times r}L_{r\times q}$ 와 같이 표현할 수 있다. 그림에서 X는 r×r 차원의 정칙행렬로서 X^{-1}가 존재하고 $F = ZX^{-1}$, $H = X^{-1}Y$이다

$$A_{p\times q} = \left[\begin{array}{c|c} X_{r\times r} & Y_{r\times(q-r)} \\ \hline Z_{(p-r)\times r} & W_{(p-r)\times(q-r)} \end{array}\right] = \left[\begin{array}{c|c} X & X \\ \hline FX & FXH \end{array}\right] = \left[\begin{array}{c} I \\ \hline F \end{array}\right] X \left[\begin{array}{c|c} I & H \end{array}\right] = \left[\begin{array}{c} X \\ \hline FX \end{array}\right] \left[\begin{array}{c|c} I & H \end{array}\right] = \left[\begin{array}{c} I \\ \hline F \end{array}\right] \left[\begin{array}{c|c} X & XH \end{array}\right]$$

Fig. 9. Matrix decomposition for pseudo inverse

A가 full-rank이면 행렬의 분할은 더욱 단순화되어 p<q인 경우 p×p 차원의 행렬 X와 p×(q-p) 차원의 행렬 Y로서 $A = [\, X \mid Y \,]$, $K=[\, I \,]$, $L = X*[\, I \mid X^{-1}*Y \,]$ 으로 표현할 수 있고, p>q 인 경우에는 q×q 차원의 행렬 X와 (p-q)×q 차원의 행렬 Z로서 A, K, L을 다음과 같이 표현할 수 있다.

$$A = \left[\frac{X}{Z} \right], \quad K = \left[\frac{I}{ZX^{-1}} \right] * X, \quad L = \left[I \right]$$

이렇게 A를 인수분해한 후 의사 역행렬 M을 $M=L'(K'AL')^{-1}K'$으로 구할 수 있다. 이 과정에서 2.3과 2.4에서 구현된 곱하기와 역행렬 계산으로 병렬계산을 수행하고 병합연산은 Matlab 라이브러리 함수를 사용한다.

2.6 자코비언의 개선

ET를 이용한 영상복원을 위해서는 반복계산이 필요하며, 반복계산의 각 단계에서 그 전 단계에서 또는 초기에 예측된 저항률 또는 유전율을 이용하여 입력 전류 또는 전압에 의해 생성된 전기장을 해석하는 과정이 필요하다. 전기장에 대한 해석해는 일반적으로 주어지지 않으므로 다양한 수치해법들이 사용된다. 이 중 불규칙한 경계면을 자연스럽게 처리할 수 있고, 내부 물성치들의 급격한 변화에도 수렴성이 좋은 유한요소법(FEM : Finite Element Method)은 각각의 전극에 대해 모든 전류 주입 패턴 즉, 모든 FEM 요소를 대상으로 저항률에 대한 경계면 전압의 변화율인 자코비언을 계산한다. 자코비언 J를 계산하는 Matlab 코드는 극심한 희소행렬에 대해 reshape 함수를 호출하고 이에 대해 측정치 U0와 계산치 U 행렬을 곱하는 부분으로 구성된다.


```
for ii=1:size(Agrad,2);
        JJ=U0.'*1/rho(ii)^2*reshape(Agrad(:,ii),NNode,NNode)*U;
        JJ=JJ(:);
        J(:,ii)=JJ;
end
```

이 코드는 Matlab에서 사용하는 mxArray 구조체 U0, Agrad, U 등의 자료구조에 대한 곱하기 코드로 구성되어 있으며 1이나 rho(ii)와 같은 스칼라 값도 역시 mxArray로 되어 있어서 행렬간 곱하기를 수행할 때 많은 오버헤드를 초래한다. 따라서 스칼라 양은 C 언어의 double 타입의 변수로 사전에 변환하여 계산을 하면 그 속도가 개

선될 수 있다. 다음으로 reshape 함수는 행렬의 차원을 변경하는 함수로서 수행 결과는 NNode의 차원을 갖는 희소행렬이다. 이 호출은 C 언어를 사용하여 Agrad 행렬의 일부분을 추출하는 함수를 구현함으로써 오버헤드를 제거할 수 있다. 또 reshape의 결과가 희소행렬인데 앞의 곱하기를 먼저 수행하면 희소성이 없어져 자동으로 일반 행렬로 변환되기 때문에 뒤의 곱하기는 일반 대 일반 행렬의 곱하기가 수행된다. 따라서 뒤의 곱하기, 즉 reshape 결과와 U를 먼저 곱하면 희소성이 유지되고 일반 행렬과 희소행렬의 곱이 되므로 수행시간의 개선을 기할 수 있다. 결국 자코비언을 C 함수로 변환한 결과 Matlab에서 20초 이상 소요되던 연산이 0.03초 이내로 단축되었다. 이것은 Matlab의 단점을 C로 극복하여서 속도 개선을 한 것이므로 다시 분할계산을 하게되면 0.015초 이내로 단축될 것으로 기대된다.

Matlab 코드에서 계산된 결과를 비교하면 최대 10^{-9} 정도의 오차를 보이는데 자코비언의 유효숫자는 10^{-1}에서 나타나므로 계산 결과의 정확성에는 영향을 주지 않는다.

3. ET 문제에의 적용 방안

동적 영상복원 알고리즘인 Matlab 코드로 구현된 확장 칼만 필터는 각각의 계산 루프마다 FEM, 자코비언 계산, 측정 갱신, 시간 갱신 등의 단계를 거치는데 주로 행렬에 대한 연산이 대부분을 차지한다. Table 1은 각 루프에 있어서 단계별 수행시간을 측정한 결과로서 첫 번째 루프에서는 처리할 행렬들이 모두 희소행렬이기 때문에 다른 루프보다 계산시간이 빠른 반면 그 이후는 루프 진행에 따라 희소 행렬이 제거되기 때문에 계산 시간이 증가한다. 결국 각 루프는 자코비언 계산과 측정 갱신 계산이 대부분을 차지하며 이를 개선하는 것이 필수적이다.

Table 1. The execution time analysis of dynamic Kalman filter algorithm

Iteration	1	2	3	4	5	6	7	8	9	10
FEM	1.04	0.99	1.38	1.31	1.32	0.99	0.99	0.99	1.05	1.04
Forward	0.28	0.28	0.44	0.39	0.33	0.27	0.27	0.33	0.27	0.27
Jaccobian	20.10	19.99	27.02	26.91	27.36	20.16	19.89	21.15	21.31	21.31
공간조정	0	0	0.05	0	0	0	0	0	0	0
측정갱신	7.80	58.06	58.39	59.82	55.85	44.11	45.09	45.04	45.04	44.11
시간갱신	0.27	0.38	0.33	0.32	0.28	0.27	0.27	0.27	0.28	0
계	29.49	79.70	88.75	88.75	85.14	65.80	66.51	67.78	67.95	66.73

확장 칼만 필터 프로그램에서 가장 많은 수행시간을 필요로 하는 측정 갱신 단계는 아래와 같이 구성되어 있다.

$$G_k = C_{k|k-1} \cdot H_k^T \cdot [H_k \cdot C_{k|k-1} \cdot H_k^T + \Gamma_k]^{-1} \quad (1)$$

$$C_{k|k} = (I - G_k \cdot H_k) \cdot C_{k|k-1} \quad (2)$$

$$\rho_{k|k} = \rho_{k|k-1} + G_k \cdot (\overline{y_k} - H_k \cdot \rho_{k|k-1}) \quad (3)$$

위 식에서 초기조건, $\rho_{1|0}$ 및 $C_{1|0}$을 설정하여 $k=1$ 부터 rK까지 반복 수행한다. 식 (1)은 Kalman gain G_k을 구하는 과정이며, 식 (2)은 측정 갱신된 오차공분산 행렬 $C_{k|k}$을, 식 (3)은 측정 갱신된 저항률의 추정값 $\rho_{k|k}$을 구하는 과정이다.

C 언어 구현에 의해 측정 갱신 계산에 있어서 행렬의 곱하기와 역행렬 계산을 이중 CPU 구조에 기반한 쓰레드 프로그램에 의해 수행속도를 개선하였다.

4. 네트워크 컴퓨팅

Matlab의 DLL 작성 도구는 네트워크 연산을 지원하지 않는다. 따라서 네트워크 컴퓨팅 프로그램을 작성하려면 Matlab의 함수들을 Visual C에서 사용하고 Winsock 라이브러리를 결합하여야 한다. 먼저 Matlab 함수를 Visual C에서 사용하려면 Matlab에서 제공하는 라이브러리를 Visual C 환경으로 이식하여야 하는데 각 라이브러리 파일들에 대해 다음과 같은 변환을 DOS 창에서 수행한다(Kruglinski, 1997).

C:> lib /DEF:xxx.def /MACHINE:IX86

이후 Visual C의 프로젝트에 위의 명령어에 의해 생성된 lib 파일들을 포함하면 C 프로그램에서 mxDestroyArray, mlfPrintMatrix 등과 같은 Matlab 함수들을 그대로 사용할 수 있다. 또 프로젝트에 wsock32.lib를 추가하면 네트워크를 통한 데이터의 전송이 가능하다.

데이터 전송은 네트워크 컴퓨팅에서 가장 필수적인 요소로서 부분 행렬의 전달이나 부분 결과의 결합 등의 측면에서 사용된다. 네트워크 계산을 위한 데이터 교환을 지원하는 라이브러리로서 MPI, RMI, DCOM, CORBA 등이 상용화되어 이용되고 있는데 이들은 복잡한 데이터 전달을 지원하기 위해 통신 오버헤드를 초래한다. 그러나 데이터의 교환 패턴이 알려져 있다면 이러한 통신 라이브러리 대신에 트랜스포트 계층의 바로 위에서 동작하는 소켓 라이브러리를 사용할 경우 그 오버헤드를 최소화할 수 있다. 소켓 라이브러리는 연결형 및 비연결형 데이터 전달을 지원하며 각 컴퓨터의 프로세스는 서버 혹은 클라이언트로 동작한다. 소켓을 사용하는 경우 서버와 클라이언트에서 사용하는 함수들의 절차는 Fig. 10과 같다.

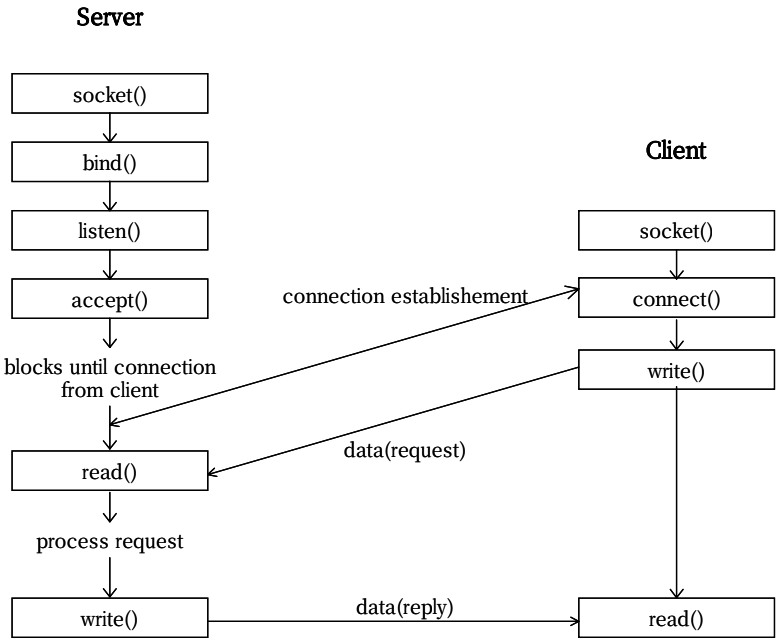

Fig. 10. Socket call sequence

C 파일은 병렬 컴퓨팅에서와 같은 방법으로 행렬을 분할한 후 Winsock을 이용하여 부분 계산을 위한 인자와 부분 계산 결과를 통신을 통해 전달하였고 Matlab C Math 라이브러리 함수 호출에 의한 부분 계산을 하였다.

Ⅳ. 성능평가 및 분석

본 절에서는 구현된 병렬 라이브러리를 중심으로 행렬의 곱하기, 역행렬 및 의사 역행렬에 대한 성능평가 및 분석을 한다. 프로그램이 수행된 컴퓨터는 2 개의 Pentium III CPU와 128 M의 메모리를 갖고 있다.

본 논문에서 구현된 병렬 곱하기 방식은 Matlab과 동일한 계산 결과를 수행하므로 정확성에 오차가 없다. 또 역행렬과 의사 역행렬인 경우에는 A22 부분행렬이 역행렬을 가질 수 있어야 해가 존재하며 피보팅이 부분 행렬 내에서만 이루어지기 때문에 오차를 포함할 수 있다. 그러나 주어진 행렬과 ET 시스템에서 계산하는 행렬에 대해 기존의 Matlab 코드와 정확성 분석 결과는 10^{-16}으로 계산 결과의 정확성에 영향을 주지 않는다.

1. 병렬 곱하기

Fig. 11은 Matlab의 곱하기와 병렬 라이브러리의 곱하기에 대한 수행시간을 보여주고 있다. 구현된 곱하기 라이브러리는 n을 행렬의 차원이라 할 때 두 개의 n×n 행렬에 대하여 수행하였고 n은 100부터 1500까지 변화시켰다. 각 행렬은 난수를 이용하여 생성시켰다. 행렬을 분할하고 합성하는 과정에서 데이터 복사에 따르는 낭비시간이 발생하므로 50% 이하로 감소되지는 않고 n=400일 때 59.4%로 수행시간을 최대로 감소시켰다. ET 시스템에서 주로 사용되는 800×800 행렬에 대해서는 63.2 %로 단축된다.

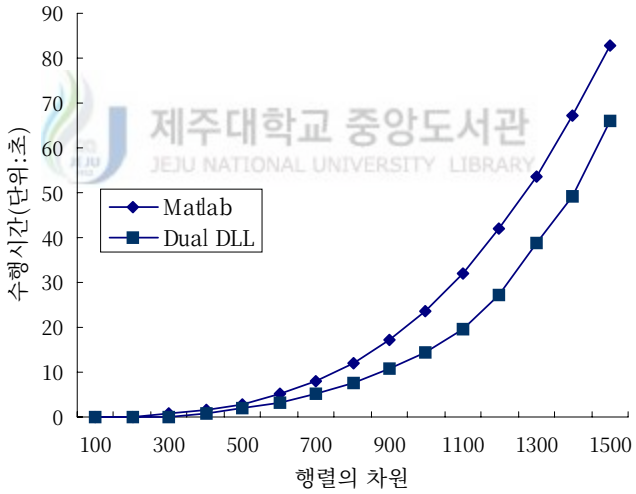

Fig. 11. Performance enhancement of matrix multiplication

2. 병렬 역행렬

Fig. 12는 Matlab의 역행렬과 병렬 라이브러리의 역행렬에 대한 수행시간을 보여주고 있다. 난수를 이용하여 n×n 차원의 정방행렬을 생성시켜 n을 100부터 1500까지 변화시켜 측정했다. 구현된 역행렬 라이브러리 역시 n=400 에서 34.8 %로 수행시간을 최대로 감소시켰다. 또 ET 시스템에서 주로 사용되는 800×800 행렬에 대해서는 역행렬의 수행시간이 54.4 %로 단축된다. 역행렬인 경우에는 병렬 계산과 아울러 분할 계산의 효과도 성능향상에 기여한다.

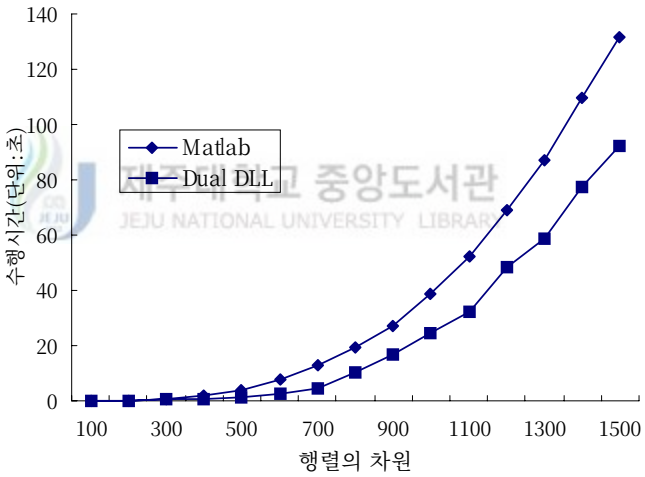

Fig. 12. Performance enhancement of matrix inverse

3. 병렬 의사 역행렬

Fig. 13은 Matlab의 의사 역행렬과 병렬 라이브러리의 의사 역행렬에 대한 수행시간을 보여주고 있다. 난수를 이용하여 (n-1)×n 차원의 행렬을 생성시켜 n을 100부터 1500까지 변화시켜 측정했다. 구현된 의사 역행렬 라이브러리는 n=500일 때 즉, 499×500 차원의 행렬에서 52 %로 수행시간을 최대로 감소시켰다. 의사 역행렬인 경우에는 병렬 계산과 아울러 분할 계산의 효과도 성능향상에 기여한다.

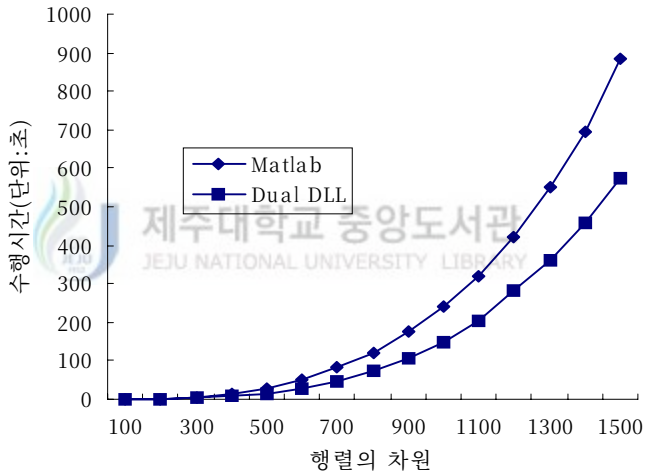

Fig. 13. Performance enhancement of matrix pseudo inverse

4. ET 확장 칼만 필터에의 적용

확장 칼만 필터 알고리즘에서 가장 많은 수행시간을 필요로 하는 측정 갱신 부분은 아래에서 보는 바와 같이 8번의 행렬 곱하기, 1번의 역행렬 계산과 기타 더하기, 빼기, 전치 등의 연산으로 구성되어 있으며 곱하기나 역행렬 계산에 입력되는 행렬의 열과 행의 개수는 원소 수에 의해 결정되는데 본 시스템에서 사용하고있는 원소의 수는 808 이다. 경우에 따라 각 행렬들이 희소행렬인 경우도 있으며 Matlab에서는 결과 행렬의 희소성에 따라 자동으로 희소 행렬 혹은 일반 행렬로 변환을 한다.

```
K = CP*JAT*inv(JA*CP*JAT+Cv);
CF = (eye(NElement1)-K*JA)*CP;
rhoF(:,t) = rhoP(:,t)+K*(Y-JA*rhoP(:,t));
```


결론적으로 확장 칼만 필터 알고리즘에 대해 자코비언 계산은 C 언어 구현에 의해, 그리고 측정 갱신 계산에 있어서 행렬의 곱하기와 역행렬 계산은 이중 CPU 구조에 기반한 쓰레드 프로그램에 의해 수행속도를 개선하였으며 개선된 시간은 Table 2와 같다. 단일 CPU 열은 Matlab 코드를 호출한 것으로 병렬 프로그램을 수행하지 않은 것이며 이중 CPU 열은 병렬 프로그램에 기반하여 작성된 DLL을 호출하는 코드로 작성된 Matlab 코드의 수행시간이다. 수행시간이 46 % 이내로 단축됨을 볼 수 있다.

Table 2. Performance enhancement via dual CPU architecture

루프	단일 CPU	이중 CPU	수행시간비
1	25.140	8.719	34.68%
2	70.313	29.750	42.31%
3	65.719	29.578	45.01%
4	65.453	29.578	45.19%
5	65.422	29.563	45.19%
6	65.359	28.991	44.36%
7	65.531	30.025	45.82%
8	65.391	29.687	45.40%
9	65.422	29.625	45.28%
10	65.187	29.266	44.90%

5. 네트워크 컴퓨팅

네크워크 컴퓨팅은 Pentium Ⅲ CPU와 128 M의 메모리를 갖은 두 대의 컴퓨터를 10Mbps 네트워크로 연결하여 다른 소프트웨어의 오버헤드 없이 Visual C로 구현하였다. 분할된 행렬은 네트워크를 통하여 노드에 전송되고 각 노드에서 지역적으로 알고리즘을 수행한다. 수행된 결과는 다시 네트워크를 통하여 전송된 후 최종적으로 병합된다. 이때 네트워크 성능 및 데이터의 양이 네트워크의 전송시간을 결정한다.

Table 3은 네트워크 컴퓨팅을 이용한 행렬의 곱하기를 행렬 차원을 증가시켜가며 수행한 결과를 단일 CPU로 수행한 결과와 비교하였다. 행렬 차원이 1000×1000 이하인 경우는 오히려 단일 CPU를 사용한 경우가 수행속도가 빠르고 1000×1000 이상인 경우는 거의 비슷하다. 그러나 여러 대의 컴퓨터를 연결한 100Mbps 네트워크에서 수행한다면 수행속도를 더욱 개선할 수 있다.

Table 3. Comparison of single CPU and networked computing

행렬차원	단일 CPU	네트워크
100	0.050	0.600
200	0.220	0.880
300	1.000	1.870
400	1.650	3.130
500	2.690	5.440
600	4.340	7.850
700	6.480	11.780
800	9.390	15.430
900	13.080	21.040
1000	25.370	26.910

Ⅴ. 요약 및 결론

ET는 영상을 얻고자 하는 물체 경계면에 여러 개의 전극을 설치하고, 이들 전극을 통해 전류를 흘려주면서 동시에 전압 또는 정전용량을 측정하여 물체 내부의 저항률 또는 유전율 분포를 예측함으로써 물체 내부의 단층 영상을 얻는 기술이다. ET 시스템은 신호를 입력하고 측정하는 하드웨어부와 측정된 전압 또는 정전용량을 이용하여 표적의 저항률 또는 유전율 분포를 계산하는 영상복원 알고리즘으로 구성된다. ET의 영상복원 알고리즘은 복잡하고 많은 행렬 연산을 수반하고 있어 막대한 계산 시간이 필요하다. 특히 이상적인 실시간 처리를 위해서는 데이터의 수집 주기 이내에 영상복원이 완료되어야 하므로 수행속도의 개선을 위해 병렬 컴퓨팅 혹은 네트워크 컴퓨팅이 절실히 요구된다.

병렬 컴퓨팅은 단일 CPU 용량 한계를 극복하기 위해 다수의 CPU를 동시에 이용하여 보다 효율적으로 문제를 풀고자 하는 방법이다. 현재 출시되는 PC들은 최대 4개까지의 CPU를 탑재하여 병렬 계산을 지원한다. 병렬 계산을 위해 쓰레드 프로그램으로 작업을 분할하면 Windows NT 이상의 운영체제는 쓰레드를 각 CPU에게 할당한다. 할당된 작업을 각 CPU가 독립적으로 수행한 후 공유메모리에 각각의 결과값을 수합하여 최종 결과를 산출한다. 만일 CPU의 개수가 증가한다면 그 개수에 맞도록 쓰레드가 확장되어야 한다.

네트워크 컴퓨팅은 네트워크 지역망으로 연결된 컴퓨터들이 서로 협력하여 주어진 작업을 수행한다. 병렬 컴퓨팅과는 달리 데이터 교환이 네트워크를 통해 수행되므로 그 속도가 떨어진다. 그러나 네트워크 컴퓨팅은 병렬 컴퓨팅에 비해 더 많은 컴퓨터들을 연결할 수 있고 네트워크의 성능이 향상되면 데이터 교환 시간이 단축되어 수행속도가 더욱 개선된다.

Matlab으로 작성된 ET 영상복원 알고리즘의 수행속도를 개선하기 위해서 Matlab에서 제공하는 Matlab Compiler를 사용하여 C 언어로 작성된 병렬 프로그램을 Matlab에서 직접 호출 가능한 DLL을 생성할 수 있다. 이러한 Matlab Compiler로 인하여 C 언어의 수행속도의 장점을 이용할 수 있고 병렬 계산에 의해 수행속도를 향상시킬

수 있다. 생성된 DLL에 의해 Matlab 사용자는 병렬 계산을 하는 새로운 함수를 사용할 수 있다.

네트워크 컴퓨팅은 Matlab이 네트워크 연산을 지원하지 않기 때문에 Visual C에서 구현하였다. 네트워크로 연결된 각 컴퓨터는 분할된 작업에 대한 연산을 독립적으로 수행한 후 서버 컴퓨터에서 부분 결과를 합성하여 최종적인 결과를 산출한다. Matlab에서 제공하는 라이브러리를 Visual C 프로젝트에 포함시켜 Visual C에서 Matlab 함수들을 그대로 사용할 수 있게 하여 Matlab 함수로 연산을 수행하였고 분할된 작업과 수행된 결과값의 네트워크를 통한 전송을 위해 소켓 라이브러리를 사용하였다.

본 논문에서 구현된 병렬 컴퓨팅 코드를 수행시켜 속도를 측정한 결과 행렬의 곱하기인 경우 400×400 차원의 두 행렬의 곱에서 최대 59.4 %로 수행시간을 단축시켰다. 역행렬인 경우 역시 400×400 차원의 행렬에서 34.8 %로, 의사 역행렬인 경우 499×500 차원의 행렬에서 52 % 까지 수행시간을 단축시켰다. 확장 칼만 필터 알고리즘에 대해 자코비언 계산은 C 언어 구현에 의해 불필요한 낭비시간을 완전히 제거하여 수행시간을 25초에서 0.1초로 단축시켰으며 측정 갱신 계산에 있어서의 행렬의 곱하기와 역행렬 계산은 이중 CPU 구조에 기반한 쓰레드 프로그램에 의해 수행속도를 개선한 결과 수행시간을 46 % 이내로 단축시켰다. 반면 네트워크 컴퓨팅은 10Mbps 네트워크에서 구현한 결과 데이터 전송에 의한 지연시간이 커서 Matlab의 함수보다 더 느리거나 비슷하게 나타났다.

향후 과제로는 더 많은 CPU를 탑재한 PC에서 병렬 컴퓨팅을 수행하여 수행속도를 향상시키는 문제와, 여러 대의 컴퓨터를 연결한 100Mbps 네트워크에서 네트워크 컴퓨팅을 수행하여 수행속도를 더욱 개선하는 문제를 들 수 있다. 또 기본적인 연산에 대해서 작업을 분리하기보다는 전체적인 영상복원 루프를 변경하여 데이터의 전송량을 줄여 성능 향상을 기할 수도 있을 것으로 판단된다.

참고문헌

Akl, S. G., 1989, *The Design and Analysis of Parallel Algorithms*, Prentice hall.

Boisvert, R., Moreira, J., Philippsen, M., Pozo, R., 2001, "Java and numerical computing," *Computing in Science & Engineering*, vol. 3, no. 2, pp. 18~24.

Calvert, K. L., Donahoo, M. J., 2000, *The Pocket Guide to TCP/IP Sockets - C Version*, Morgan Kaufmann

Cohen, A. M., Woodring M., 1998, *Win32 Multithreaded Programming*, O'Reilly

Datta, B. N., 1995, *Numerical linear algebra and applications*, Brooks/Cole

Douglas, E. Comer, 1995, *Internetworking with TCP/IP*, Prentice hall.

Gallivan, K. A., Heath, M. T., Ng, E., Ortega, J. M., Peyton, B. W., Plemmons, R. J., Rommie, C. H., Sameh, A. H., Voigt, R. G., 1990, *Parallel Algorithms for Matrix Computations*, SIAM.

Hinsberg, P., 1999, *Windows NT Applications: Measuring and Optimizing Performance*, MTP.

Hwang, K., 1998, *Parallel Processing for Supercomputers*, McGraw-Hill.

Jones, A., Ohlund, J., 1999, *Network programming for microsoft windows*, Microsoft Press.

Kim, K. Y., Kim, B. S., Kim, M. C., Lee, Y. J., and Vauhkonen, M., 2001, "Image reconstruction in time-varying electrical impedance tomography based on the extended Kalman filter," *Measurement Science and Technology*, vol. 12, no. 8, pp. 1~8.

Kruglinski, D., 1997, *Inside Visual C++ 5.0*, Microsoft Press.

Liu, J. W., 2000, *Real-Time Systems*, Prentice Hall.

Mathworks, 1999a, *Matlab Compiler User's Guide*, Mathworks Inc.

Mathworks, 1999b, *Matlab C Math Library User's Guide*, Mathworks Inc.

Matthew. S., 2001, *Windows 2000 Server*, Sybex.

Richter, J, 1996, *Advanced Windows*, Microsoft Press.

Sayood, K., 2000, *Introduction to Data Compression*, Morgan Kaufmann.

Searle, S. R., 1982, *Matrix Algebra Useful for Statistics*, John Wiley & Sons.

Silberschatz, A., Gagne, G., Galvin, P., 1999, *Applied Operating Systems Concepts*, John Wiley & Sons.