# 멀티프로세서 기반의 고속 영상처리 기술에 대한 벤치마킹

최학남, 박은수, 김준철, 김학일
인하대학교 컴퓨터 비젼 연구실

# Benchmarking on High-speed Image Processing Techniques based on Multi-processor

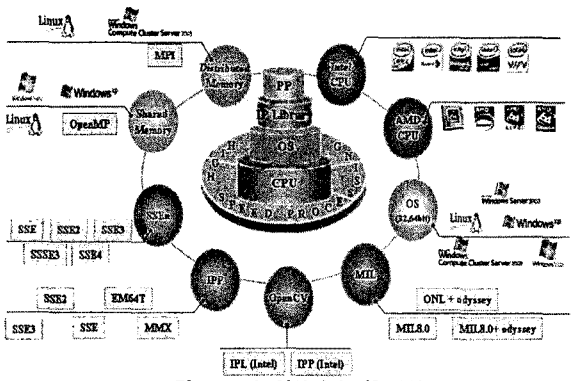
Xue-Nan Cui, Eun-Soo Park, Jun-Chul Kim, Hak-Il Kim
Computer Vision Laboratory, Inha University

**Abstract** -본 논문에서는 멀티프로세서 기반의 고속 영상처리 알고리즘 개발방법에 대해 소개한다. 영상획득 방식의 발전과 더불어 고해상도 영상의 획득이 가능해지고 영상이 컬러화가 되면서 많은 영상처리 응용분야에서 알고리즘 고속화를 필요로 하고 있다. 이러한 수요를 만족시키기 위해서는 최근에 출시되고 있는 멀티프로세서를 최대한 활용할 수 있는 알고리즘 개발이 최우선이다. 본 논문에서는 OpenMP, MIL(Matrox Image Library), OpenCV, IPP(Integrated Performance Primitives), SSE (Streaming SIMD (Single Instruction Multiple Data) Extensions)등 병렬처리와 고속 영상처리 라이브러리를 이용한 알고리즘 개발방법에 대해 소개하고, 각 개발방법에 따른 알고리즘 성능을 분석 및 평가하였다. 실험결과로부터 SSE와 IPP, MIL(Thread)을 이용하여 Mean, Dilation, Erosion, Open, Closing, Sobel등 알고리즘을 구현하여 4057×4048크기의 영상에 적용하였을 때 7~35msec의 좋은 성능을 나타내어 기타 방식보다 우수함을 알 수 있었다.

## 1. 서 론

최근 들어 인텔이나 AMD에서 CPU의 전력소모를 줄이고 처리속도를 향상시키기 위하여 멀티코어 프로세서를 잇달아 출시하고 있다. 현재 인텔에서는 Quad-core까지 발표하였고, 2008년에는 Octa-Core가 출시할 예정이다[7]. 이렇듯 CPU가 단일프로세서에서 멀티프로세서로 변화하면서 컴퓨터 성능이 한 단계 향상되었다. 이는 기존의 하나의 프로세서에서 하던 일을 두 개의 프로세서에서 동시에 진행하기에 가능하다. 하지만 기존의 프로그램을 그대로 멀티프로세서에서 실행시켜 성능향상을 얻을 수 있는 것은 아니다. 반드시 병렬처리 프로그램을 작성하여 실행시켜야 만이 성능향상을 가져온다. 영상처리 알고리즘 최적화는 이미 성숙기에 들어서서 많은 노력을 들이더라고 큰 효과를 얻지 못하고 있다. 더욱이 영상의 해상도는 점점 더 높아지고 컬러 영상화 되면서 영상처리 알고리즘 고속화는 더욱 절실히 필요한 실정이다. 이러한 고속화 수요를 만족시키기 위해서는 Multi-core 프로세서의 추세에 맞춰 각 프로세서에 최적화된 명령어들의 사용과 Multi-core를 최대한 활용할 수 있는 스레드 프로그램은 필수적이다. 영상처리 고속화 관련 연구동향을 살펴보면 조상현[1] 등은 인텔 MMX(MultiMedia eXtension)기술을 이용한 영상처리 고속화에 대한 연구를 진행하였고, Kanysun Choi[2]등은 SSE2를 이용하여 실시간 PCB(Printed Circuit Boards) 결함검출 방법을 제안하였으며, David A.Bader[3]등은 병렬처리의 전반적인 소개와 필요성 및 공유메모리 기반의 병렬처리 방법에 대해 소개하였다.

고속 영상처리를 실현하기 위해서는 CPU, OS(Operation System), 영상처리 라이브러리 및 구현방식, 병렬처리 방법 등 네 가지 요소를 고려해야 한다. [그림 1]은 고속 영상처리의 다양한 접근방식을 나타내고 있다.




<그림 1> 고속 영상처리 접근방식

본 논문은 서론, 본론, 결론 세 부분으로 나누어 졌다. 서론에서는 고속 영상처리 연구동향에 대해 서술하였고, 본론에서는 OpenMP, MIL, OpenCV, IPP, SSEn에 대한 소개와 실험결과에 대해 서술하였으며, 결론에서는 실험결과에 대한 간단한 분석과 향후 추가적으로 수행해야 할 과제에 대하여 서술하였다.

## 2. 본 론

### 2.1 OpenMP

OpenMP는 공유메모리 병렬 프로그램 표준 프로토콜로 대부분 Unix, Windows NT에 OpenMP 컴파일러가 존재하며 Fortran과 c/c++을 지원한다. OpenMP는 컴파일러 지시어, 실행시간 라이브러리, 환경변수로 구성되었으며, 공유메모리 병렬성을 표현하기 위한 컴파일러 지시어들의 집합이라고 할 수 있다. 컴파일러 지시어 집합은 병렬성 표현을 위한 제어 구조, 스레드들 간의 통신을 위한 데이터 환경 구문, 다중스레드의 실행을 조율하는 동기화 구문으로 구성되고, 실행시간 라이브러리는 병렬 실행에 참여하는 스레드 개수, 스레드 번호 등과 같은 병렬 인수들의 설정과 조회를 가능하게 한다. 환경변수는 병렬실행에 참여하는 스레드 개수의 지정과 같은 실행 시스템의 병렬 인수들의 설정과 조회를 가능하게 한다.

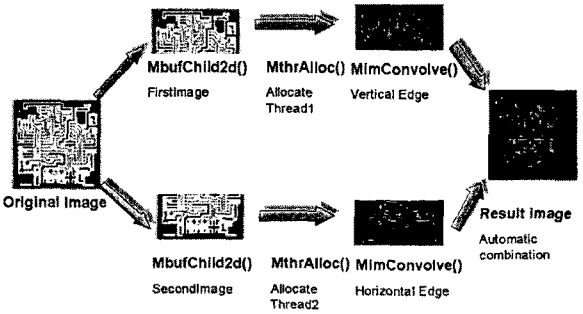
OpenMP를 이용하여 병렬 프로그램을 작성하면 상대적으로 쉽게 구현할 수 있지만 동기화, 데이터 의존성 등을 고려하지 않고 OpenMP를 적용하면 사용자가 원하지 않은 엉뚱한 결과를 출력한다. 따라서 프로그래머는 프로그램 작성시 동기화, 데이터 의존성 등 문제를 고려하여 작성하여야 만이 정확하고 빠른 병렬처리 알고리즘을 개발할 수 있다.

OpenMP 프로그램은 일반적으로 루프문에 적용하기 때문에 우선 프로파일링을 통해 시간이 가장 많이 걸리는 루프를 찾고, 루프내에 존재하는 데이터 의존성, 데이터 유효범위 등을 조사하여 데이터 의존성을 제거한 다음 OpenMP 명령어를 사용하여 병렬화 한다[4].

### 2.2 MIL

MIL(Matrox Image Library)은 Matrox 사에서 제공하는 고성능 영상처리 라이브러리로써 영상 캡쳐, 프로세싱, 분석, 디스플레이, 아카이빙(Archiving)을 쉽게 사용할 수 있게 알고리즘을 제공해준다. 또한 Intel의 MMX, SSE, SSE2를 이용하여 개발되어 고속처리를 보장하고, Odyssey와 같은 영상처리 전용보드, Odyssey에 최적화된 ONL(Odyssey Native Library)을 제공하여 성능향상을 가져오게 한다. 하지만 추가적인 옵션 사용에 따른 비용은 사용자가 부담해야 한다.

MIL을 이용하면 스레드를 이용한 병렬처리도 가능하다. 영상을 2등분으로 나누고, 나누어진 영상에 영상처리 알고리즘을 동시에 적용하는 방식으로 구현할 수 있는데, 주의할 것은 두 영상으로 나눌 때 정확하게 2등분으로 나누면 두 영상의 경계면에서 약간의 오차가 발생하게 되므로 2등분보다 좀 더 큰 사이즈로 영상을 나누어야 한다[5].




<그림 2> MIL에서의 스레드 프로그램

### 2.3 OpenCV

OpenCV는 Intel의 주도하에 만들어진 무료 영상처리 전용 라이브러리이다. OpenCV는 Linux처럼 세계 각지의 개발자들에 의해 만들어지고 있으며 OpenCV 코딩 규칙을 지킨다면 자신이 만든 알고리즘도 OpenCV에 등록할 수 있다. 또한 고속처리를 위하여 Intel에서 제공하는IPL(Image Processing Library), IPP(Integrated Performance Primitives)를 사용하여 만들어졌으며 Intel CPU에 최적화 되어있다.

OpenCV는 영상을 읽고 쓰는 등 간단한 기능에서 부터 Optical Flow, Camera Calibration 등 고급 알고리즘까지 350여 가지의 영상처리 관련 알고리즘을 지원하고 있다. 이러한 알고리즘들을 이용하여 OpenCV로 구현할

수 있는 Application들은 Human-computer interaction(HCI), Object identification, Segmentation, Recognition, Face recognition, Gesture recognition, Motion Tracking, Ego motion, Motion understanding, Structure from motion(SFM), Mobile robotics 등이 있다[6].

### 2.4 IPP

IPP(Integrated Performance Promitives)는 Intel에서 제공하는 low-level 라이브러리로써 MMX, SSE, SSE2를 이용하여 개발된 최적화된 알고리즘들을 제공해준다. Intel CPU에 최적화되었기 때문에 다른 종류의 CPU에서는 좋은 성능을 나타내지 못하는 단점이 있기는 하지만 Inte Processor에서는 최고의 성능을 보장한다. IPP는 영상처리뿐 만아니라 신호처리(Signal Processing), 행렬처리(Matrix Processing), 3D Data Processing 등을 위한 1000여개의 최적화된 함수들을 제공한다[7].

IPP는 별도로 인터페이스를 제공하지 않기 때문에 OpenCV의 인터페이스를 이용하면 쉽게 알고리즘을 구현하여 테스트 해볼 수 있다. OpenCV와 IPP는 데이터 타입이 틀리기 때문에 OpenCV의 데이터 타입으로부터 IPP 데이터 타입으로 변환시켜 주고 IPP 함수를 적용시켜야 한다. 다음은 OpenCV와 IPP의 혼합사용방법에 대한 예이다.

```
#include "ipp.h"
// Mean filter
IplImage* image = cvLoadImage("Filenames.bmp",0);
IplImage* dst = cvCreateImage(cvGetSize(image),IPL_DEPTH_8U,1);
IppiSize size;
size.width = image->width;
size.height= image->height;
S_img = (Ipp8u *)ippsMalloc_8u(size.width*size.height);
D_img = (Ipp8u *)ippsMalloc_8u(size.width*size.height);
T_img = (Ipp8u *)ippsMalloc_8u(size.width*size.height);
ippiCopy_8u_C1R((const Ipp8u*)image->imageData,size.width,S_img,size.width,size);

ippiFilterLowpassBorder_8u_C1R(S_img, size.width, D_img, size.width,
                               size, ippMskSize3x3, ippBorderConst, 0, T_img);

cvSetImageData(dst, D_img, size.width);
cvSaveImage( "ippmean.bmp", dst);
cvNamedWindow("IPPmean",0);
cvShowImage("IPPmean",dst);
cvReleaseImage(&image);
ippiFree(img);
```

<그림 3> OpenCV 인터페이스를 이용한 IPP

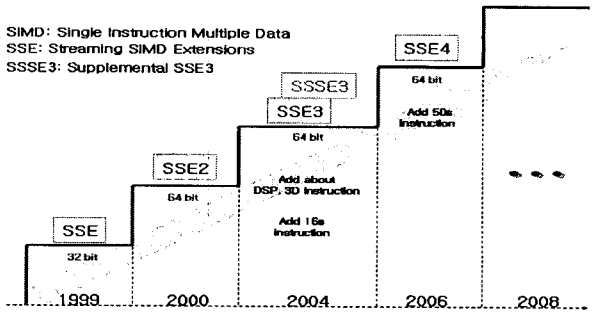
<표 1> 영상처리 구현 방식간의 장단점

| | 장점 | 단점 |
|---|---|---|
| OpenMP | -병렬처리 구현이 쉬움<br>-병렬처리에 의한 속도 향상 | -OpenMP를 지원하는 컴파일러 필요<br>-데이터 의존성, 동기화를 고려 |
| MIL | -구현이 쉬움<br>-다양한 알고리즘 제공<br>-고속 영상처리 가능 | -상대적으로 고가<br>-추가적인 옵션에 따른 비용<br>-32bit만 지원(64bit 출시예정) |
| OpenCV | -무료<br>-구현이 쉬움<br>-인터페이스 제공<br>-IPP를 설치하면 IPP와 비슷한 성능을 냄<br>-고속 영상처리 가능 | -Intel Processor에만 최적화<br>-MIL, IPP 등에 비해 성능이 떨어짐 |
| IPP | -구현이 쉬움<br>-고속 영상처리 가능<br>-이식성이 좋음<br>-라이브러리를 이용하여 만든 제품에 대한 로얄티 지불 없음 | -Intel Processor에만 최적화<br>-인터페이스 제공 안됨 |
| SSEn | -고속 영상처리 가능<br>-복합 함수에 대한 고속처리 가능 | -Intel Processor에만 최적화<br>-MIL, IPP에 비해 상대적으로 구현이 어려움<br>-기존의 알고리즘 변경<br>-개발시간이 오래 걸림<br>-유연성이 떨어짐 |

### 2.5 SSEn

SSEn은 SSE(Streaming SIMD(Single Instruction Multiple Data) Extensions), SSE2, SSE3, SSSE3, SSE4등을 포함한다. SSE는 Pentium3에서 부터 지원하였고, 70개의 명령어를 포함하고 있으며, 8개의 128bit 레지스터를 사용한다. SSE2는 SSE의 확장으로써 double-precision floating과 8bit, 16bit, 32bit을 위한 새로운 명령어들이 추가 되었고, Pentium4에서부터 지원되었다. SSE3은 DSP 연산을 쉽게 할 수 있는 명령어와 캐쉬 명령과 같은 Process management 명령어가 추가되었으며, SSSE3는 SSE3의 확장으로써 16개의 새로운 최적화된 명령어들이 추가되었다. SSE4는 50개의 소수점 연산을 포함한 연산들이 추가되었으며 2007년 하반기에 인텔의 제품군에 포함될 예정이다. [그림 4]는 SSE의 발전과정을 나타낸 그림이다.

SSEn 명령어를 이용하여 프로그램을 작성하는 방법은 주로 Assembly, Instrinsics, Classes, Automatic Vectorization을 사용하는 4가지 방식이 있다. Assembly로 프로그램을 작성하면 하드웨어에 근접한 언어이기 때문에 좋은 성능을 얻을 수 있지만 프로그램 작성이 어렵고 이식성이 떨어지는 단점이 있다. Intrinsics은 Assembly보다 상대적으로 코딩이 쉽고 성능도 비슷하여 사용자들이 많이 사용한다. Classes 방식은 클래스로 선언되어 있는 명령들을 직접 사용기 때문에 C++코딩 스타일에 가까운 코딩방법을 제공한다. Automatic Vectorization은 -QAX, -QRESTRICT 스위치를 사용하여 Intel 컴파일러에서 자동적으로 SSE 프로그램으로 변환하여 수행하는 방식이다. 이 방식은 데이터 의존성에 크게 의존하기 때문에 프로그램에 변수들 사이의 의존성이 존재하면 컴파일러는 SSE 코드로 변화하여 수행하지 않는다[7].




<그림 4> SSE의 발전과정

### 2.5 실험결과

본 논문에서는 각 구현방식들 간의 장단점에 대해 [표 1]과 같이 분석하였고, 각 구현방식에 따른 알고리즘 수행시간을 비교하기 위하여 Intel® Core 2 CPU 6600 @ 2.4GHz를 사용하였으며, Windows XP환경에서 [표 2]의 알고리즘들을 구현하여 수행속도를 비교하였다. 실험에서 사용된 영상의 크기는 4057×4048이다.

<표 2> 구현방식에 따른 영상처리 알고리즘 수행시간 비교

| Function | Filter size | OpenMP (msec) | MIL (msec) | MIL Thread (msec) | OpenCV (msec) | IPP (msec) | SSE (msec) |
|---|---|---|---|---|---|---|---|
| Mean | 3x3 | 78.07 | 37.39 | 34.67 | 65.93 | 7.64 | 30.28 |
| Dilation | 3x3 | 36.40 | 15.05 | 11.76 | 101.56 | 11.6 | 10.10 |
| Erosion | 3x3 | 36.16 | 13.85 | 11.12 | 104.33 | 11.49 | 10.47 |
| Closing | 3x3 | 75.87 | 34.39 | 29.43 | 205.8 | 23.58 | 20.21 |
| Opening | 3x3 | 73.56 | 34.45 | 28.50 | 206.11 | 24.44 | 20.41 |
| Sobel(H) | 3x3 | 152.84 | 30.12 | 26.76 | 192.06 | 14.81 | 14.32 |

## 3. 결 론

본 논문에서는 고속 영상처리를 위한 여러 가지 접근 방법들에 대해 소개하였고, 영상처리에서 흔히 사용하는 알고리즘들을 구현하여 수행속도를 비교 분석하였으며, 영상처리 구현 방식간의 장단점에 대해 분석하였다. MIL, OpenCV, IPP등은 영상처리 라이브러리 형태로 제공하기 때문에 구현이 쉽지만 OpenMP와 SSE는 사용자가 직접 알고리즘을 구현해야 한다. OpenMP는 데이터 의존성, 동기화 등 문제를 고려하여 프로그램을 작성해야만 정확한 결과를 얻을 수 있고. SSEn를 이용한 방식은 128Bit 전용 레지스터를 사용하기 때문에 데이터 형식이 바뀌므로 알고리즘을 SSE에 적용 가능하도록 변화시켜야 한다. 본 논문에서는 SSSE3까지 이용하여 알고리즘을 구현하였다. 실험결과로부터 보면 전반적으로 우수한 처리속도를 가지지만 그중에서도 SSE와 IPP, Thread를 이용한 MIL에서 좋은 성능을 보였고, 또한 OpenCV는 IPP를 설치하면 IPP와 비슷한 성능을 냄을 알 수 있었다. 향후 각 영상처리 접근방법에 대한 성능을 보다 정확하게 분석하기 위하여 영상처리에 필요한 여러 가지 알고리즘들을 추가적으로 구현하여 수행 시간을 비교할 필요가 있고, SSEn에 OpenMP나 MPI(Message Passing Interface)와 같은 공유 및 분산 병렬처리가 가능한지, 또한 성능향상을 가져오는지에 대해 추가적으로 연구할 필요가 있다.

## [참 고 문 헌]


[1] 조상현, 박창준, 최홍문등, "인텔 MMX 기술을 이용한 영상처리 루틴의 고속화", 전자기술연구지, 경북대 공대, vol. 22, pp.154-161, 2001

[2] Kany-Sun Choi, Jae-Young Pyun, Nam-Hyeong Kim, "Real-Time Ins pection System for Printed Circuit Boards", DAGM 2003, LNCS 2781, pp. 458-465, 2003

[3] David A. Bader, Bernard M. E., Peter Sanders, "Algorithm Engineerin g for Parallel Computation", Experimental Algorithmics, LNCS 2547, pp.1 -23, 2002

[4] Rohit Chandra, Leonardo Dfum, Dave Kohr, Dror Maydan, Jeff McDonald, Ramesh Menon, "Parallel Programming in OpenMP", Morgan Kaufmann Publishers, 2000

[5] www.matrox.com

[6] http://www.opencv.co.kr/

[7] www.intel.com