Journal of the Korea Academia-Industrial cooperation Society
Vol. 16, No. 4 pp. 2749-2756, 2015

http://dx.doi.org/10.5762/KAIS.2015.16.4.2749
ISSN 1975-4701 / eISSN 2288-4688

CUDA 연산을 이용한 개선된 영상 매칭 방법에 관한 연구

조경래[1], 박병준[1], 윤태복[1*]
[1]서일대학교 컴퓨터소프트웨어과

A Study on Improved Image Matching Method using the CUDA Computing

Kyeongrae Cho[1], Byungjoon Park[1], Taebok Yoon[1*]
[1]Dept. of Computer Software, Seoil University

요 약 최근 데이터의 질이 높아짐에 따라 영상을 처리하는데 많은 시간이 소모되는 문제가 제기되어 영상 처리 알고리즘의 가속화가 필요하게 됨으로써, 기존의 CPU와 CUDA(Compute Unified Device Architecture) 기반의 인식 시스템에서 연산속도와 성능이득 비교를 위해 OpenMP를 가지고 측정할 수 있는 문자 인식시스템으로 학습된 문자데이터가 입력되면 매칭이 가장 잘 되는 영상의 영역을 인식하는 환경으로 구현하여 각 영문 알파벳의 글씨체가 일정하고 크기가 규격화 되어 있으므로 문자를 학습하고 문자 정합도를 계산하기 위한 영상 매칭 방법을 구현하게 되었다. GPGPU(General Purpose GPU)프로그래밍 플랫폼 기술인 CUDA연산 기법을 이용하여 알고리즘을 빠르고 효율적으로 처리하는 OpenMP에서 인텔 i5 2500의 네 개의 코어를 사용하여 인식 할 때, 기존 CPU의 성능보다 4배의 속도가 나오지 않고 데이터의 분할과 병합 연산의 지연으로 인해 약 3.2배의 속도로 향상되는 가속화 방법을 제안하고 그래픽카드에서 처리하는 병렬처리 결과, 순차적 연산을 수행하였던 CPU 기반의 처리에 비해 성능이득이 약 21X(배)로 향상됨을 확인하였다.

Abstract Recently, Depending on the quality of data increases, the problem of time-consuming to process the image is raised by being required to accelerate the image processing algorithms, in a traditional CPU and CUDA(Compute Unified Device Architecture) based recognition system for computing speed and performance gains compared to OpenMP When character recognition has been learned by the system to measure the input by the character data matching is implemented in an environment that recognizes the region of the well, so that the font of the characters image learning English alphabet are each constant and standardized in size and character an image matching method for calculating the matching has also been implemented. GPGPU (General Purpose GPU) programming platform technology when using the CUDA computing techniques to recognize and use the four cores of Intel i5 2500 with OpenMP to deal quickly and efficiently an algorithm, than the performance of existing CPU does not produce the rate of four times due to the delay of the data of the partition and merge operation proposed a method of improving the rate of speed of about 3.2 times, and the parallel processing of the video card that processes a result, the sequential operation of the process compared to CPU-based who performed the performance gain is about 21 tiems improvement in was confirmed.

Key Words : Image matching, image processing, parallel programming

1. 서론

Multi-CPU와 GPU의 다중코어 형태의 프로세서가 보급됨으로써 병렬처리 프로그래밍의 중요성은 점점 더 커지고 있다. 영상과 데이터의 질(Quality)이 높아짐에 따라 GPU는 그래픽스 연산 시 많은 화소(Pixel) 데이터

*Corresponding Author : Taebok Yoon(Seoil Univ.)
Tel: +82-2-490-7441 email: tbyoon@seoil.ac.kr
Received October 26, 2014 Revised(1st January 19, 2015, 2nd April 7, 2015)
Accepted April 9, 2015 Published April 30, 2015

를 병렬적으로 처리하기 위해 코어의 수를 지속적으로 확장시키면서 쓰레드(Thread)를 사용한 병렬처리로 계산 능력을 획기적으로 향상시켜 왔다. 이렇게 다수코어 기반 스트리밍 계산에 특화된 GPU의 연산 능력은 현재 CPU 대비 상당한 정도의 성능 우위를 점유하게 되었다[1]. 동일한 반복적인 패턴을 요구하는 처리에서 GPU 사용이 늘어나고 있으며, 특히 많은 계산을 요구하는 과학계산 분야나 그래픽 분야에서 CPU를 대신하여 빠른 연산 속도를 보여주고 있다. 하지만 효율적인 GPU 프로그래밍을 하기 위해서는 파이프라인을 이해하여야 하는 문제가 발생한다. 이러한 문제를 해결하기 위하여 NVIDIA사에서는 그래픽 카드에 GPGPU (General Purpose GPU) 아키텍처인 CUDA (Compute Unified Device Architecture)를 제공하여 파이프라인을 거치지 않고 GPU를 이용할 수 있는 프로그래밍 기법을 제공하여 GPU 구조에 대한 지식이 없이도 MFC(Microsoft foundation class) 환경에서 그래픽 카드를 이용한 병렬처리 개발 환경을 제공하였다[2].

GPU는 그래픽 처리를 위해 제작되어 데이터를 병렬적으로 수행하는데 최적화 되었으며 GTS450은 192개의 스트림 프로세서를 가지고 있다. CUDA는 C언어의 확장으로 설계된 쓰레드와 GPU 메모리를 위한 API를 제공하며 GPU에서 실행 되도록 작성된 커널 (Kernel)은 호스트 PC의 호출에 의해서 실행된다. 영상 매칭은 비교적 단순한 알고리즘이지만 연산의 양이 많아 응답 시간이 길어지는 문제점을 가지고 있다. CPU만으로 영상 매칭을 구현할 경우 데이터가 커질수록 많은 행렬 연산에 의해 연산 속도가 급격히 떨어지게 된다. 반면 CUDA를 이용하여 연산을 할 경우 병렬연산이 가능하기 때문에 각 픽셀마다 순차적으로 수행되는 연산을 병렬적으로 처리하여 매우 빠르게 연산 수행속도를 높일 수 있다. 영상 매칭 과정에서 소벨연산, 이진화, 라벨링, 영상 정합도(adjustment) 계산에 CUDA를 적용하였다.

본 논문에서 사용한 NVIDIA 그래픽 카드(GTS 450)는 192개의 스트림 프로세서(SP)를 가지고 있으며 32개의 SP를 가지는 스트림 멀티프로세서(SM) 4개를 가지고 있다[3].

제안 방법의 실험환경으로 운영체제 (OS)는 Windows XP Service Pack3, CPU는 AMD FX6300 Six-Core), 주메모리 4.00GB, 프로그래밍 툴 Visual Studio10, 그리고 NVIDIA CUDA연산의 하드웨어인 그래픽 카드는 GeForce GTS 450으로 하였다[3]. 본 논문에서는 많은 연산이 필요한 영상 매칭 알고리즘에 병렬 처리 기법을 적용함으로써 알고리즘의 효율성과 영향을 분석한다. 논문의 구성은 2장에서는 영상 매칭 방법과 CUDA 연산의 구현방법에 대하여 기술하였다. 3장 에서는 실험 결과에 대하여 서술하였고 최종 4장에서는 실험 결과 및 향후 연구과제에 대하여 서술하였다.

2. 영상 매칭

2.1 영상 인식

영상 매칭 기술 관련하여 E. 마사카즈는 고급 이미지 처리에 대한 영상 비전 실용 기술에서 컴퓨터 비전은 대상의 크기, 회전, 조명과 같은 다양한 변화에 따라 같은 영상이라도 서로 다른 상황이 될 수 있다고 설명하고 있다[4]. 특히 크기가 서로 다른 부분을 검사할 경우 동일한 객체 일 경우라도 다른 객체로 인식할 문제를 가지고 있다. 본 논문 에서는 객체를 -2배, 0배, 2배, 4배, 6배, 8배로 늘리거나 줄이면서 Fig. 1과 같이 객체와 맞는 값을 자동으로 찾아서 적용한다.

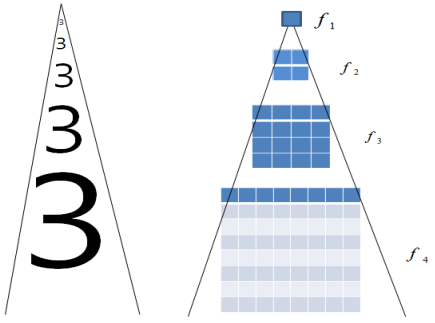

Fig. 1. The object recognition size conversion

학습된 영상이 f_2일 경우 f_8정도 크기의 객체를 인식하기 위해서는 4 × 4(f_4), 8 × 8(f_8)로 확대 샘플링(Sampling)을 하였고 반대일 경우는 2, 1, -2, -4, -8로 다운 샘플링을 하며 인식을 한다. 학습 영상이 1단계 일 때, 1024 × 1024 영상을 인식을 하려면 총 8단계의 샘플링 학습을 거쳐 식 1과 같은 공식을 얻을 수 있으며 k값은 현재 샘플링 단계이며 q 값은 최대 확대나 축소할 단계의 값이다.

$$f_k(x,y) = f_{(k-1)}\left(\frac{x}{r}, \frac{y}{r}\right) \quad (1)$$
$$r = 2 \text{ or } \frac{1}{2}$$
$$1 \le k \le q$$

2.2 영상 정렬

두 영상 사이의 상대적 위치를 결정하는 영상의 정렬은 자동 시각 검사 분야에서 매우 중요한 부분이다. 정확한 검사를 하기 위하여 영상 매칭 알고리즘을 이용하여 두 영상의 좌표축을 정확히 일치시켜야 한다[4, 5].

본 논문에서의 영상 정렬은 제품의 모서리나 절단면, 화살표 등의 윤곽선이 선명한 부분들을 기준으로 선택하여 지정된 정보를 가지고 정렬한다. 가급적이면 검사할 영상은 정렬 영역에 포함시키지 않는 것이 좋다. 검사 할 영상을 정렬 영역에 포함시켜 검사할 경우 정렬 영역이 불량이면 영상정렬에 영향을 미치게 되어 정렬이 잘못될 수도 있기 때문이다.

문서의 영상의 전처리는 영상 획득 과정에서 기울어짐이 발생할 수 있다. 영상의 왜곡(Distortion)이 심해지면 문서 인식이 불가능하기 때문에 기울어진 영상의 교정이 필요하다. 영상의 기울어진 정도를 Fig. 2와 같이 위쪽(Top Angle)과 왼쪽(Left Angle)의 기울기를 계산하여 각도를 구한다.

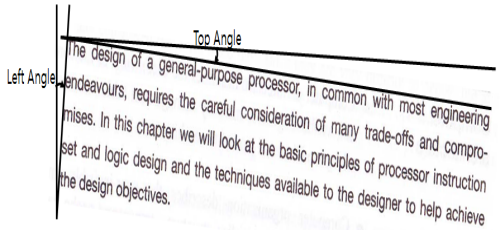

Fig. 2. Tilted image correction and image

2.3 문자 영역 추출

기울어진 영상이 교정되면 검사 할 영상이 있는 문자를 따로 추출하여 영역 설정을 한다. 검사 영역을 설정하는 이유는 영상을 검사할 때 다른 영역이 잘못 선택되어 잘못된 검사를 방지 할 수 있고 작은 영역을 검사하여 처리 속도를 빠르게 할 수 있기 때문이다.

이미지에서 문자를 인식하기 위해서는 먼저 문자의 꼭대기(Top)의 가로축과 밑 부분(Bottom)의 가로축, 첫 문자의 세로축(Left)과 문자간의 간격(Spacing) 값을 찾아야 한다.

추출된 텍스트 영역은 인식의 대상인 문자 단위로 분할을 하며 연결요소 분석(Connected Component Analysis) 방법 등을 적용한다[6].

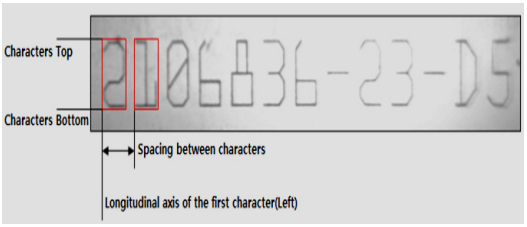

Fig. 3. Extraction of character spacing

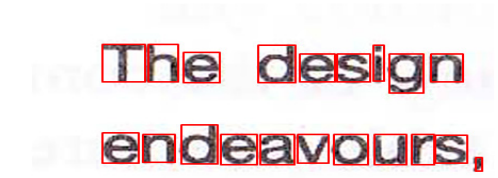

Fig. 4. Case of character-based segmentation

Fig. 3은 문자의 추출 간격이고 Fig. 4는 문자단위 분할 결과이다[7]. 연결요소 분석방법은 영역을 분할할 때 인식기가 인식할 수 있도록 문자분할 과정을 거쳐야 한다. 비정상적인 문자분할로 인해 인식결과가 전혀 다르게 나올 수 있기에 정확한 분할 과정이 요구된다. 문자분할의 경우 분할된 사각형(Rectangle)은 폭과 높이의 비율이 비슷하면 좋지만 Fig. 5와 같이 문자들 간격이 서로 다르고 문자들이 공간상 서로 겹치는 문제가 발생한다. 두 영상 사이의 상대적 위치를 결정하는 영상의 정렬은 자동 시각 검사 분야에서 매우 중요한 부분이다[8].

Fig. 5. Case of character segmentation incorrect

문자가 정상적으로 분할 됐는지를 평가하기 위해서는 분류에서 텍스트로 설정된 영역들에 대해 학습된 골든 이미지 문자와 비교하여 비정상적으로 분할된 문자들을 파악하여 Fig. 6과 같이 분할 영역을 재수정 한다.

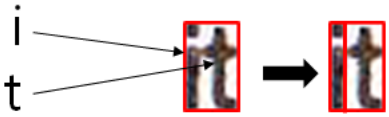

Fig. 6. Area of the modified text

2.3 문자 인식

각 문자의 정보는 벡터 폰트 정보를 이용하여 문자를 인식한다.

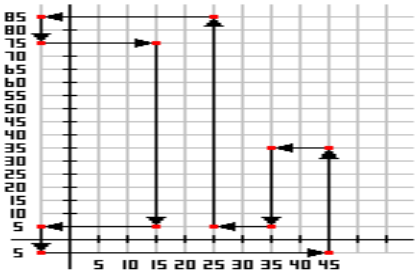

Fig. 7. Vector information for the number “1”

Fig. 7은 숫자 “1”에 대한 벡터 정보이며 화살표가 끝난 지점의 빨간 점(vertex)의 위치와 vertex의 방향을 저장한 벡터 폰트 정보를 가지고 있다. 인식된 문자 및 숫자를 벡터 폰트로 학습을 한다. Fig. 8은 벡터 폰트 정보를 이미지의 문자 크기에 맞게 pixel(화소)화 시킨 것으로 인식할 숫자 “1”의 일부분이다.

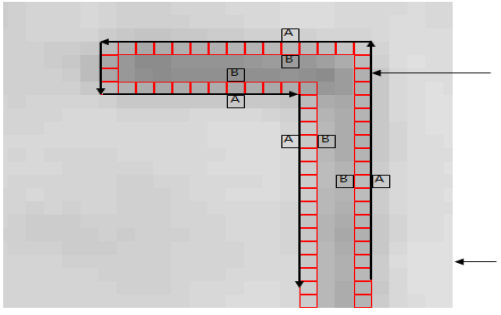

Fig. 8. Vector font information for the number “1”

하나의 벡터 폰트 정보의 모든 pixel들에 대해 (A - B) 연산을 한 값들을 더한 후, 비교했던 pixel들의 개수로 나누면 평균값이 계산된다. 비교한 벡터 폰트 정보들 중 이 평균값이 가장 큰 것을 문자의 학습 데이터로 선택한다[9].

2.4 정합도 계산

문자를 서로 구분하기 위하여 측정을 하였다. 측정 방법은 문자의 중점에서의 원주율과 라벨링 값으로 측정하였다. (식 2)와 같이 문자의 중점을 구하고 (식 3)과 같이 원주율을 구하였다. Fig. 9은 원형도와 라벨링을 한 영상이고 Table 1은 계산된 수치 값이다[10].

$$x = \frac{1}{n}\sum_{i=0}^{N-1} x_i \tag{2}$$

$$y = \frac{1}{n}\sum_{i=0}^{N-1} y_i$$

$$\ell = \frac{4\pi A}{\ell^2} \tag{3}$$

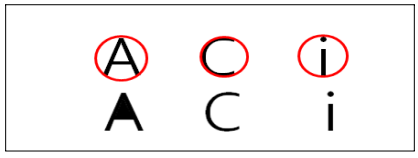

Fig. 9. Image of circularity and labeling

Table 1. Round and labeling calculated

	A	C	i
radius (mm)	0.750	0.675	0.710
circle density (mm^2)	2.250	1.822	2.016
circumference (mm)	4.71	4.239	4.459
labeling (pixel)	363.0	187.0	103.0

CPU환경에서 생성하는 스레드(Thread)는 생성 즉시 실행코드가 프로세서에서 실행된다. 생성한 스레드의 개수와 CPU의 코어 수가 동일하면 최적의 효율을 나타내지만, 코어의 수보다 많아지면 시분할 스케줄로 스레드가 동작하게 되어 효율이 떨어지게 된다.

CUDA에서의 스레드는 CPU의 분할과 작업을 분할하는 태스크의 개념이 병행되어 있다. CUDA는 그래픽 카드가 발휘할 수 있는 성능 이내의 개수만큼 스레드가 생성되어 코어에서 동작을 한다. 설정된 스레드이상의 데이터가 있을 경우 효율성이 떨어지지 않을 정도의 개

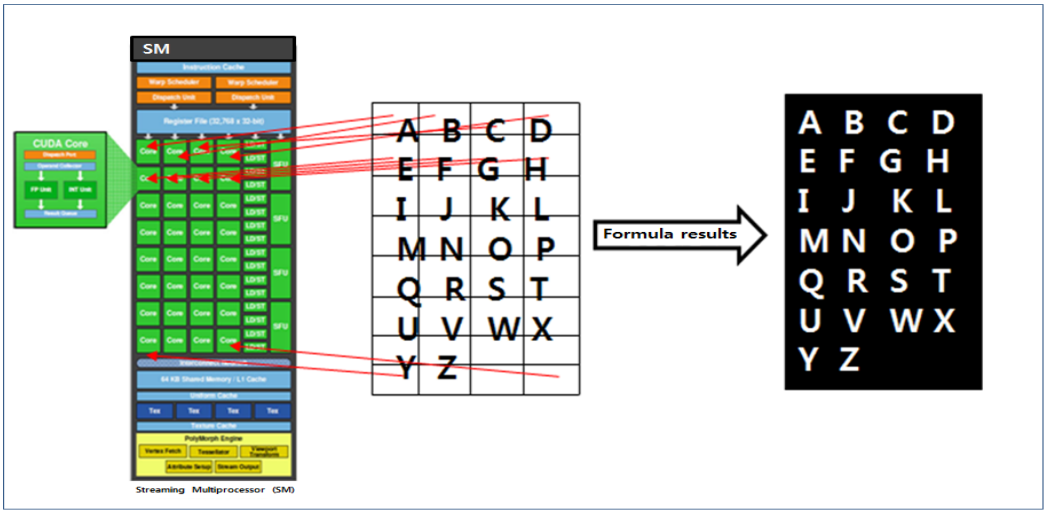

Fig. 10. One image per thread block allocation block operation

수만 실행을 하고 나머지는 대기를 하며 유휴코어가 발생하면 GPU는 나머지 대기하는 스레드를 처리하여 유휴 발생률을 줄일 수 있다[11]. GPU를 이용한 CUDA 연산은 구조적인 문제로 CPU와 GPU간에 메모리를 공유하여 사할 수 없다. GPU 연산은 메인 메모리로부터 데이터를 전송 받아 연산을 수행하여야 한다.

데이터 전송에 따른 지연(Delay)을 줄이고 연산의 효율을 높이기 위해선 최소한의 데이터 교환이 일어나야 하며 CPU는 GPU에서 처리할 수 있는 최대 용량의 데이터를 생성해야 한다. 하지만 하나의 공유 메모리를 병렬 프로그램에서 동일한 작업을 반복 연산을 할 경우 Fig. 11과 같은 경쟁상대(Race Condition)문제가 발생하여 CPU연산으로 계산된 값과의 차이가 발생할 수 있다.

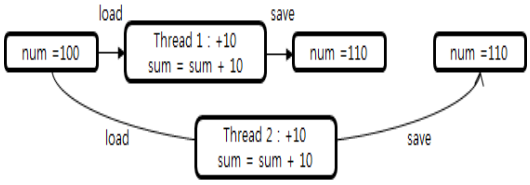

Fig 11. Competition process occurs

이러한 경쟁 상태를 해결하기 위해서는 아토믹(원자계산)이나 크리티컬 섹션을 이용하여 Fig. 12와 같이 동기화(Synchronization)를 맞춰야 한다.

덧셈과 뺄셈 같은 간단한 연산은 아토믹 함수를 이용하고 복잡한 연산이 요구되는 연산은 크리티컬 섹션을 이용한다. 한 장의 각도를 가지는 영상 이미지를 하나의 블록 단위로 생성하여 개별적인 데이터를 사용하도록 아토믹 함수를 이용하여 설계하였다[12].

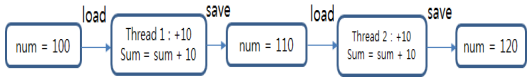

Fig 12. Resolve competition problems with synchronization

각각의 블록단위 마지막 스레드 값 연산이 끝난 후 동기화 하도록 설계하여 한번에 4개의 이미지를 Fig. 10과 같이 4개의 SM가 동시에 연산 한다. CUDA연산의 결과와 CPU기반의 연산 결과가 100% 일치하여 경쟁상대(Race Condition)문제를 해결하였다[13].

3. 실험 및 결과

학습된 문자와 읽어 들인 문자를 비교분석 하여 가장 정합도가 높은 최적의 값을 찾는다. 정합도를 결정하기까지는 많은 계산 양으로 인하여 응답 까지 오랜 시간이 걸리는 문제가 있다. 빠른 연산을 하기 위하여 본 논문에서는 GPU를 이용하여 계산을 하였다.

일반적인 도서의 한 개 장(chapter)에는 평균 2,000개의 알파벳이 있고 CPU를 이용하여 정합도를 계산할 경우 약 3초의 시간이 소요 되었다. 본 논문의 실험환경은 10 개의 장을 대상으로 약 20,000 자의 알파벳을 가지고 실험하였다. 실험 방법으로는 그래픽 카드의 병렬 연산

과 CPU의 병렬 연산을 비교하기 위하여 FX6300과 GeForce GTS 450 그래픽카드 그리고 OpenMP를 구동을 위한 FX6300과 비슷한 클록을 가지고 있는 인텔 i5 2500 CPU로 실험하였다. OpenMP는 공유 메모리 다중 프로세서에 병렬처리를 수행할 수 있도록 하는 기술로 최적화 되어 있는 인텔 컴파일러를 사용하였다. OpenMP를 이용한 문자 처리는 서로의 데이터가 겹치지 않도록 전체 문자를 동등하게 프로세서의 코어의 개수만큼 분할하고 각 코어는 할당 된 데이터만 처리하도록 분배를 하였다[14].

본 논문에서는 실험방법에 따른 알고리즘의 성을 성능을 평가하기 위하여, 동일한 영상에 대하여 인식 결과를 도출하였고, Table 2를 보면 세 가지 방식 모두 같은 인식률이 측정되어 문자 인식 성능은 세 가지 방법 모두 같음을 확인 하였다.

Table 2. chapter char recognition rate

Item	Alpa.	CPU	GPU	Open MP
1 chapter	2,000	92%	92%	92%
10 chapter	20,000	88%	88%	88%

$$\text{인식률}(\%) = \frac{\text{인식된 글자 수}}{chapter} \times 100 \quad (4)$$

문자만 있는 1 chapter 는 90%가 넘는 인식률을 보였지만 2 chapter 부터는 기호와 문자가 붙어있는 것을 제대로 분리를 하지 못하여 전체 인식률이 떨어졌다. 차 후 분리 알고리즘의 향상을 통하여 개선할 것이다.

Table 3. Comparing the execution time of the CPU and GPU computing

Item	Alpa.	Time (msec)		
		CPU	GPU	Open MP
1 chapter	2,000	2,800 msec	147 msec	900 msec
10 chapter	20,000	29,300 msec	1,395 msec	9,630 msec

OpenMP에서 인텔 i5 2500의 네 개의 코어를 사용하여 인식 할 때, CPU의 성능보다 4배의 속도가 나오지 않는 것은 데이터의 분할과 병합 연산의 지연으로 인하여 약 3.2배의 속도 향상을 볼 수 있다. 향후 CPU의 발달로 다중코어와 CPU 병렬 연산의 개선으로 더욱 빠른 속도 향상을 할 것이다.

Table 3과 같이 처리 속도가 “CPU < OpenMP < GPU”의 순으로 빠른 것을 확인할 수 있다. 이처럼 영상의 크기가 커질수록 GPU 의 처리 성능이 점점 향상되는 것을 확인할 수 있다. 최근의 CPU 와 GPU 연산간의 실행속도 비교에서 1 chapter는 약 19배의 처리 속도의 향상을 나타내었고, 10 chapter에서는 21배 까지 데이터가 늘어날수록 성능 이득이 21배 까지 측정되었으며, 메모리간의 통신에 의한 오버헤드를 포함한다면 시스템 상에서의 성능 이득은 약 30배로 볼 수 있다[15].

4. 결론

본 논문에서는 GPGPU(General Purpose GPU)프로그래밍 플랫폼 기술인 CUDA연산 기법을 이용하여 알고리즘을 빠르고 효율적으로 처리하는 OpenMP에서 인텔 i5 2500의 네 개의 코어를 사용하여 인식 할 때, 기존 CPU의 성능보다 4배의 속도가 나오지 않고 데이터의 분할과 병합 연산의 지연으로 인해 약 3.2배의 속도로 향상되는 가속화 방법을 이용하여 그래픽카드에서 처리하는 병렬처리 결과 순차적 연산을 수행하였던 CPU 기반의 처리에 비해 성이 약 21X(배)로 향상됨을 확인할 수 있었다. CPU와 CUDA 기반의 인식시스템에서 CPU와 CUDA간의 연산속도와 비교하기 위해 OpenMP를 가지고 성능 이득을 측정할 수 있는 문자인식시스템에서 학습된 문자의 데이터가 입력되면 가장 매칭이 잘 되는 영상의 영역을 인식하는 환경으로 구현하였다. 각 영문 알파벳의 글씨체가 일정하고 크기가 규격화 되어 있으므로 문자를 학습하고, 문자의 정합도를 계산하기 위한 영상 매칭 기법을 적용하였다.

계산 시간이 많이 요구되는 정합도 계산이지만, GPU를 이용한 연산으로 약 20 X(배) 이상의 계산 시간을 줄일 수 있었다. 실생활에서의 응용은 다양하지 않겠지만 상호작용(Interaction)이 중요한 가상현실 (VR), 게임 인공지능(G-AI), 로봇의 인지 과학 분야 (cognitive science field)에서 효율적으로 응용될 수 있을 것으로 기대된다. 하지만 GPU를 활용하더라도 데이터의 수가 증가할수록 연산의 시간은 상대적(relative) 이고 비례적(proportional)으로 증가하게 됨으로써 제안시스템을 실시간으로 구현하는데 어려움이 있었지만, 테이블 3과 같이 GPU를 이용한 CUDA 연산으로 효율적인 연산 결과를 얻을 수 있

었다. 하지만, 다양한 비디오 카드의 종류와 버전에 따른 실험 환경의 한계로 API의 성능 실험을 하지 못하여 좀 더 정확한 데이터를 얻을 수 없었다. 향후 연구에서는 고성능 그래픽 카드를 이용할 경우 더욱 빠른 연산 결과를 기대할 수 있을 것이며 GPU성능이 지속적으로 발전해 감에 따라 비교 데이터의 종류의 범위를 영문 알파벳을 포함한 한글이나 숫자형식으로 확대하여 문자 매칭과, 지원 되는 API에 따른 코드 최적화, 알고리즘 개선 등으로 더욱 발전된 영상 매칭 시스템을 개발할 것이다.

References

[1] Jiangang Kong,Yangdong Deng, "GPU Accelerated Face Detection", International Conference on Intelligent Control and Information Processing, August13-15, 2010.

[2] J.D.Owens, M.Houston, D.Luebke, S.Green, J.E.Stone and J.C.Phillips, "GPU Computing", Proceedings of the IEEE, vol.96, no.5, pp.879-899, May.2008.
DOI: http://dx.doi.org/10.1109/JPROC.2008.917757

[3] NVIDIA CUDA Programming Guide v2.1 8 Dec. 2008. <http://www.nvidia.com/ object/ cuda_home.html>

[4] E. Masakazu, “Machine Vision A Practical Technology for Advanced Image Processing,” Japanese Technology Reviews, Computers and Communications, Grodon and Breach

[5] R.C. Gonzalez, and R.E. Woods, Digital Image Processing, 2nd Ed, Prentice-Hall, 2002.

[6] K. Hendengren, “Methodology for Automatic image-based inspection of industrial objects,” in Advances in Machine Vision, Sanz J. ed, Springer-Verlag, 1988.
DOI: http://dx.doi.org/1007/978-1-4612-4532-2_4

[7] R.T. Chin, and C.A. Harlow, “Automated Visual Inspection: A Survey,” IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.PAMI-4, No.6, pp. 557-573, 1982.
DOI: http://dx.doi.org/10.1109/TPAMI.1982.4767309

[8] K.W. Tobin, “Inspection in Semiconductor Manufacturing,” Webster’s Encyclopedia of Electrical and Electronic Engineering, Vol.10, pp. 242-263, 1999.

[9] D. J., Kang, J. E. Ha, Digital Image Processing Using Visual C ++, SciTech media, chap 11, 2003.

[10] M. Sonka, V. Hlavac, and R. Boyle, “Image Processing Analysis and Machine Vision,” Cengage Learning, 2007.

[11] "CUDA 2.1 Programing Guide", http://developer.download.nvidia.com/compute/cuda/2_1/toolkit/docs/nVidia_CUDA_Programming_Guide_2.1.pdf

[12] Tom.R.halfhill, "Parallel Processing with CUDA", Reed Electronics Group, http://www.nvidia.com/docs/IO/55972/220401_Reprint.pdf

[13] G. Poli, J. H. Saito, J. F. Mari, M. R. Zorzan, "Processing Neocognitron of Face Recognition on High Performance Environment Based on GPU with CUDA Architecture," International Symposium on Computer Architecture and High Performance Computing, pp.81-88, 2008.
DOI: http://dx.doi.org/10.1109/SBAC-PAD.2008.25

[14] Intel Architecture Software Developer’s Manual vol.1, 2, 3.

[15] N. Ashraf, Sibi. A, "CUDA-Accelerated Face Recognition," poster presentation, GPU Technology Conference, 2010.

[16] K.-W. Lee, "Implementation of Video Surveillance System with Motion Detection based on Network Camera Facilities", The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 14, No. 1, pp.169-177, Feb. 28, 2014.
DOI: http://dx.doi.org/10.7236/JIIBC.2014.14.1.169

[17] W. Lee, D. Nam, "Volume Rendering Architecture of Mobile Medical Image using Cloud Computing", The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 14, No. 4, pp.101-106, Aug. 31, 2014.
DOI: http://dx.doi.org/10.7236/JIIBC.2014.14.4.101

[18] I.-H. Jee, "Effective Compression Technique of Multi-view Image expressed by Layered Depth Image", The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 14, No. 4, pp.29-37, Aug. 31, 2014.
DOI: http://dx.doi.org/10.7236/JIIBC.2014.14.4.29

조 경 래(Kyeongrae Cho) **[정회원]**

- 1999년 2월 : 대진대학교 컴퓨터공학과 (공학사)
- 2000년 2월 : 성균관대학교 정보통신공학과 (연구과정)
- 2002년 8월 : 성균관대학교 정보통신공학과 (공학석사)
- 2009년 2월 : 성균관대학교 컴퓨터공학과 (공학박사)
- 2012년 3월 ~ 현재 : 서일대학교 컴퓨터소프트웨어과 조교수

<관심분야>
인간과컴퓨터 상호작용, 컴퓨터프로그래밍, 디지털콘텐츠, 사물인터넷, 센서 네트워크, 클라우드 및 빅데이터 분석, 컴퓨팅 보안과 개인 정보보호

박 병 준(Byungjoon Park) **[정회원]**

- 2000년 2월 : 고려대학교 전자정보공학부 (공학사)
- 2002년 8월 : 고려대학교 대학원 의료정보기기학과 (공학석사)
- 2010년 2월 : 국민대학교 대학원 전산과학과 (이학박사)
- 2011년 3월 ~ 현재 : 서일대학교 컴퓨터소프트웨어과 조교수

<관심분야>
영상처리, 패턴인식

윤 태 복(Taebok Yoon) **[종신회원]**

- 2001년 8월 : 공주대학교 전자계산학과 (이학사)
- 2005년 2월 : 성균관대학교 컴퓨터공학과 (공학석사)
- 2010년 8월 : 성균관대학교 컴퓨터공학과 (공학박사)
- 2011년 3월 ~ 현재 : 서일대학교 컴퓨터소프트웨어과 조교수

<관심분야>
데이터 마이닝, 사용자 모델링, 게임인공지능