

하드웨어 구조에 적합한 2차원 회선처리 기법

정윤혜[○] 박용진 박진홍 변혜란 한탁돈

연세대학교 컴퓨터과학과

yhjung@msl.yonsei.ac.kr, jini99@cs.yonsei.ac.kr, jhpark@msl.yonsei.ac.kr, hrbyun@yonsei.ac.kr,
hantack@msl.yonsei.ac.kr

2D Convolution Method Suitable for Hardware Architecture

Yunhye Jung[○], Yongjin Park, Jinhong Park, Hyeran Byun, Tackdon Han

Dept. of Computer Science, Yonsei University

요약

다양한 응용프로그램에서 효과적으로 2차원 영상을 처리하기 위해서는 여러 가지 기법들이 이용되는데 그 중 2차원 필터링은 가장 많이 사용되는 방법 중 하나이다. 2차원 필터링에서 회선처리는 수평과 수직 방향의 1차원 선형 필터를 이용하는 방법이다. 2차원 회선처리는 커다란 이미지 위를 커널이 움직이며 연산을 해야 하므로 연산량이 매우 많으며 메모리 접근을 많이 필요로 한다. 하지만 회선처리는 입력화소 뿐 아니라 주변 화소 값까지 고려하는 지역적인 동작으로 인해 병렬화된 처리가 가능하다. 이에 본 논문에서는 메모리 접근을 줄이고 연산을 병렬적으로 처리함으로서 회선처리의 수행 시간을 개선하는 하드웨어 기반의 회선처리 방법을 제안한다.

1. 서 론

2차원 필터링 기법은 영상의 화질개선이나 영상 내의 객체 인식을 위하여 많이 사용된다. 2차원 필터링 기법 중 하나인 회선처리(convolution)기법은 블러링, 샤프닝, 특징점 추출을 등의 영상 처리 알고리즘에 있어서 많이 사용된다[1, 2].

회선처리 기법은 해당 입력 화소 뿐만 아니라 그 주위의 화소 값도 함께 고려해 공간 영역을 계산한다. 일반적으로 2차원 회선처리는 1차원 선형 필터를 사용해 수평과 수직 방향으로 적용해 구현이 가능하다[3, 4]. 입력 이미지 위에 커널을 이동하며 행렬의 승산을 반복해 그 계산량이 방대하기 때문에 회선처리를 이용하는 시스템에서는 많은 계산 시간이 소요된다.

$N \times N$ 영상에 대해 $M \times M$ 회선처리 커널이 주어졌을 때 출력되는 행렬은 다음과 같은 식(1)으로 연산된다[5].

$$b(x, y) = \frac{1}{M \times M} \sum_{i=x}^{x+M-1} \sum_{j=y}^{y+M-1} f(i, j) h(x-i, y-j) \quad \text{식(1)}$$

f 는 입력 이미지, h 는 입력 커널, b 는 회선 처리 된 이미지이다. 이러한 수행을 입력 이미지의 모든 화소에 대해서 수행을 하기 때문에 $O(N^2)$ 정도의 복잡도를 가지게 된다. 기존의 방법은 하나의 화소의 회선처리 결과를 얻

기 위해 반드시 9개의 화소를 가져와서 연산을 수행해야 하나 제시된 방법은 이전에 가져온 화소 값들을 재사용함으로써 메모리 접근을 줄이는 효과를 얻을 수 있다. 따라서 본 논문에서 제시하는 회선처리 방법은 메모리에 대한 접근을 줄이고 연산을 병렬화하여 수행시간을 감소시키는 하드웨어 구조를 제안한다.

2. 관련 연구

회선처리에 관한 알고리즘은 다양한 환경에서 제시되었다. 본 장에서는 기존에 여러 환경에서 적용된 회선처리 알고리즘의 특징에 대해서 기술하였다.

2.1 CUDA를 이용한 회선처리 알고리즘[6]

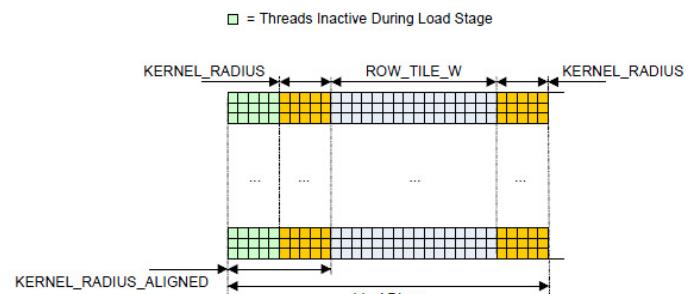


그림 1. CUDA에서 쓰레드 블록 padding

회선처리 값을 얻는 데는 병렬적 하드웨어로 구성된 GPU(Graphics Processing Unit) 등을 이용한 방법에

이 논문은 2009년도 정부(교육과학기술부)의 지원으로 한국연구재단의 지원을 받아 수행된 기초연구사업 연구임(No. 2009-0088516)

매우 적합하다.

CUDA(Compute Unified Device Architecture)에서 사용되는 가장 간단한 구현 방법은 shared memory에 이미지를 블록 단위로 가져오는 것이다. 따라서 각 쓰레드는 블록안의 화소 값을 공유할 수 있고, 이미지를 블록 단위로 처리하게 된다.

CUDA를 통한 회선처리 시, global 메모리에서 화소 값을 가져와 수평, 수직 필터를 적용해 계산한다. GPU 구현의 효율성은 이용할 수 있는 shared memory에 의존하기 때문에 그림 1과 같이 블록 구성이 중요하다.

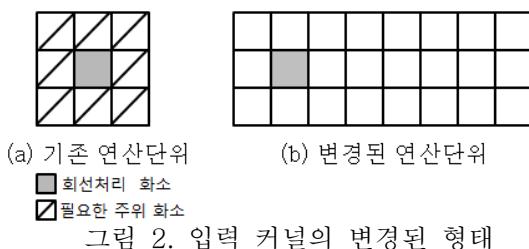
2.2 VLIW Mediaprocessor에서 회선처리 알고리즘[5]

실시간 응용 프로그램을 지원하기 위해 ASIC과 같은 하드웨어적 해결책으로 연산 수행의 능력을 개선할 수 있다. [5]에서 제시한 회선처리 기법은 programmable 한 접근으로 VLIW processor에 적용하였다. 이러한 programmable 회선처리 알고리즘은 다양한 이미지 응용 프로그램에 대체가 가능하므로 유용하다. 또한 여러 개의 기능적 유닛, programmable DMA controller의 특징을 가지고 있다.

3. 본 론

3.1 제안하는 회선처리 과정

회선처리는 전체 이미지에서 입력 커널 단위로 수행한다. 입력 커널에서 회선처리는 행렬의 승산 부분과 가산 부분으로 나누어진다. 그림 2 (a)의 기준의 회선처리 방법 연산 과정에서는 하나의 화소의 결과 값을 얻기 위해 추가로 8개 화소를 읽어야 한다. 따라서 그림 2 (b)에서와 같이 연산단위를 변경하여 추가적인 화소 값을 매번 읽는 것을 방지하고 재사용하도록 한다. 2차원 영상 데이터의 경우 주위의 화소들 사이에 서로 연관성을 가지며 유사하다는 것을 이용해 입력 커널의 경계면에 있는 화소 값을 인접한 다음 라인에 다시 사용하게 되기 때문이다.



입력커널이 8x8인 경우 그림2 (b)와 같은 연산단위를 사용하여 스캔라인 버퍼에 3x8 크기의 화소 값을 읽어온다. 화소의 회선처리 결과를 얻기 위해서는 9화소에 대해 3x3 마스크를 곱하는 연산을 해야 한다. 따라서 승산 연산을 병렬로 처리하기 위해서는 그림 3과 같이 마스크를 1x3으로 나누어 3x8 버퍼의 각 라인에 곱한다.

그림 3의 버퍼에서 한 라인만 살펴보면 이 중 3개의 화소가 하나의 회선처리 결과 값에 영향을 미치기 때문에 마스크와 곱해진 3개의 화소 값을 더해 레지스터에 저장한다. 최종 결과 값을 얻기 위해 가산 연산을 진행한다. 가산 연산을 통해 도출된 값은 SRAM에 저장되고 그 후 DRAM에 전체 이미지에 대한 2차원 필터 결과를 저장하게 된다. 연산 결과는 다음과 같다.

$$\begin{aligned} A*11 + B*21 + C*31 &= A^*(\text{REG}) \\ I*12 + J*22 + K*32 &= B^*(\text{REG}) \\ Q*13 + R*23 + S*33 &= C^*(\text{REG}) \\ A^* + B^* + C^* &= \text{SRAM1} \quad (\text{화소 J의 최종 결과 값}) \\ &\dots \\ F*11 + G*21 + H*31 &= A^*(\text{REG}) \\ N*12 + O*22 + P*32 &= B^*(\text{REG}) \\ V*13 + W*23 + X*33 &= C^*(\text{REG}) \\ A^* + B^* + C^* &= \text{SRAM6} \quad (\text{화소 O의 최종 결과 값}) \end{aligned}$$

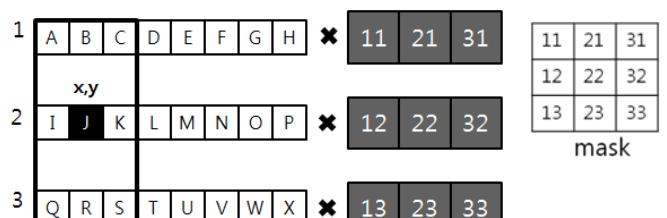
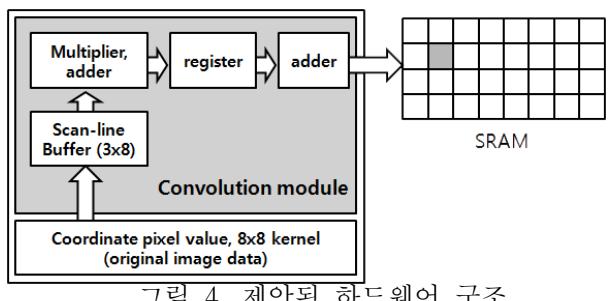


그림 3. 라인 별 병렬화 된 회선처리의 승산계산

3.2 제안한 하드웨어 구조

그림 4는 본 논문에서 제안하는 하드웨어 구조이다.



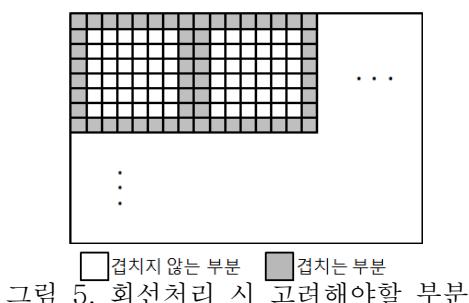
Coordinate pixel value는 원본 이미지에서 좌표 값에 해당하는 화소 값을 입력 커널의 크기만큼 가져온다. 그리고 scan-line 버퍼는 입력 커널로부터 연산단위(3x8) 크기의 화소 값을 가져와 저장한다. Scan-line에 저장 후 multiplier, adder를 통해 승산연산과 부분적인 가산 연산을 적용하고 결과 값을 register에 저장한다. 그리고 adder에서 가산연산을 마치면 최종 회선 처리 되어진 값이 SRAM에 저장되게 된다. 이 SRAM에 저장된 값들은 입력 커널의 한 라인을 기준으로 DRAM으로 이동된다.

- 1) 입력커널에서 8화소 가져오는 시간 : 19 cycle
- 2) 마스크 적용 위해 9개 곱셈연산 시간 : 1 cycle
- 3) 한 라인에서 3개 화소 덧셈연산 시간 : 1 cycle
- 4) Adder에서의 덧셈연산 시간 : 1 cycle
- 5) SRAM으로 8 화소 저장 시간 : 1 cycle
- 6) SRAM에서 DRAM으로 8화소 전달 시간 : 19cycle

4. 성능 측정

본 논문에서 제안하는 회선처리 기법과 하드웨어 구조 제안 방법에 대한 성능 평가는 입력 이미지에 대해 8x8 입력 커널을 가지고 적용하는데 필요한 사이클 횟수를 가지고 평가하였다.

회선처리 시 입력 커널의 마지막 행과 열은 다음 회선 처리 결과에 영향을 준다(그림 5). 따라서 입력 커널에서 열의 경우 8화소 단위로 이동하고 맨 앞과 마지막에 필요한 화소 정보에 대해서 버퍼를 둔다. 추가적인 버퍼는 하드웨어 구조에서 제시한 scan-line에 맞춰서 두게 된다. 행의 경우 역시 열과 같은 현상이 발견되므로 가장 위와 아래 행에 버퍼를 두어 이를 해결한다.



이 때 추가되어지는 버퍼는 입력 커널에 대해 그림 6과 같은 크기만큼 필요하다. 회선 처리를 위한 알고리즘의 psuedo code는 그림 7과 같다.



그림 6. 추가 버퍼의 형태

```

1 : 입력 : 원본 이미지
2 : 출력 : 회선 처리된 이미지
3 : for 전체 이미지의 행
4 :   for 전체 이미지의 열
5 :     입력 커널 전 필요한 추가적 buffer 정보 저장
6 :     for 입력 커널의 행
7 :       행 별로 데이터 가져와 scan-line 버퍼에 저장
8 :       for 입력 커널의 열
9 :         곱셈과 덧셈의 병렬적 연산 후
10:        레지스터에 저장
11:      end of for loop
12:    행 별로 덧셈 연산 후 SRAM 저장
13:  end of for loop
14: end of for loop
15: end of for loop

```

그림 7. 회선처리를 위한 알고리즘 psuedo code

표 1은 본 논문에서 제안한 회선처리 기법을 사용하여 이미지 크기에 따른 각 단계 별 연산횟수를 나타낸다.

표 1. 회선처리 시 이미지 크기에 따른 연산 횟수

이미지크기	곱셈	덧셈	adder 덧셈	SRAM	DRAM
320x240	804,720	804,720	262,720	11,200	11,200
640x480	3,176,208	3,176,208	1,051,520	44,160	44,160
800x600	4,949,472	4,949,472	1,643,200	68,800	68,800

각 이미지 크기에서 320x240은 1400, 640x480은 5520, 800x600은 8600번의 입력 커널을 기준으로 연산을 수행하게 된다. 표 1의 단계별 연산 횟수를 기반으로 총 사이클 수를 도출해 낼 수 있다. 총 사이클 수는 구하기 위해서는 입력 커널로부터 scan-line으로 8화소씩 가져오는 것, 곱셈연산, 덧셈연산, 레지스터 저장 후 adder에서의 덧셈 연산, SRAM으로 이동, DRAM으로 이동하는 매커니즘을 기반으로 한다. 기존 방법의 경우 총 사이클 계산 시 연산의 병렬화 부분이 없기 때문에 회선처리에서 필요한 곱셈과 덧셈에 요구되는 시간을 모두 소비하게 된다. 제안된 방법에서 총 사이클을 구하는

식은 다음 식(2)와 같다. 여기서 연산 횟수에 대한 나눗셈은 연산을 병렬로 처리함으로서 속도를 개선한 부분이다.

$$\begin{aligned} \text{총 사이클} = & (\text{커널연산횟수}*8)*19 + \\ & (\text{곱셈 횟수}/9) + \\ & (\text{덧셈 횟수}/3) + \\ & (\text{adder}/3) + \\ & \text{SRAM접근횟수} + \\ & \text{DRAM접근횟수}*19 \end{aligned} \quad \text{식(2)}$$

표 2는 식(2)를 이용해 이미지 크기에 따라 제안한 하드웨어 구조에서의 총 사이클 수를 나타낸다.

표 2. 이미지 크기에 따른 HW 사이클 수

이미지크기	320x240	640x480	800x600
총사이클수	882,028	3,484,394	5,430,698

5. 결 론

본 논문에서는 2차원 영상 처리 분야에서 병목현상을 유발하는 회선처리 기법을 하드웨어에 적용하였다. 회선 처리의 주변 화소 값을 고려하는 지역성을 이용하여 읽어온 데이터를 재사용할 수 있도록 입력의 커널의 형태를 바꾸었다. 그리고 연산을 병렬적으로 수행함으로써 빠른 회선 처리 결과를 얻을 수 있었다.

이렇게 성능이 개선된 회선처리 기법은 연산량이 많은 다양한 영상 처리 알고리즘에 적용이 가능하다. 추후에는 여러 가지 필터에 대한 성능분석과 커널 크기에 따른 성능 개선에 대한 연구가 요구된다.

6. 참고문헌

- [1] C Lin, M Sheu, H Chiang, C Liaw, Z Wu, "An efficient convolution interpolation kernel for digital image scaling", IEICE Electronics Express, 2008.
- [2] 이지원, 한탁돈, "이차원 콘볼루션을 위한 배열처리 기의 설계", 대한전자공학회 학술대회 논문집, 1990.
- [3] Bracewell,R,"Convolution and Two-Dimensional Convolution." Ch. 3 in The Fourier Transform and Its Applications. New York: McGraw-Hill, pp. 25-50 and 243-244, 1965.
- [4] J. W. Woods and S. D. O'Neil, "Sub-banded coding of imgae," IEEE Trans. Acoustics,

Speech, and Signal Processing, Vol. ASSP-34, no.5, pp.1278-1288, Oct. 1986.

- [5] R Managuli, G York, Y Kim, "An Efficient Convolution Algorithm for VLIW Media processors", Proceedings of SPIE, 1988.
- [6] V Podlozhnyuk, "Image convolution with CUDA", NVIDIA Corporation white paper, 2007.