제26회 한국정보처리학회 추계학술발표대회 논문집 제13권 제2호 (2006. 11)

SSE 명령어 기반 실시간 처리 가우시안 필터 연구

강필중*, 이종수*
*울산대학교 컴퓨터정보통신공학부

e-mail:bug6nq@mail.ulsan.ac.kr

A Study on Real-time Processing of The Gaussian Filter using The SSE Instruction Set.

Pil-Jung Ghang*, Jong-Soo Lee*
*School of Computer Engineering & Information Technology,
University of Ulsan

요 약

본 논문은 SIFT(Scale Invariant Feature Transform)알고리즘의 실시간처리 응용프로그램 작성기법을 기술하고 있는데, 단일 프로세서에서 병렬처리 기능을 지원하도록 설계된 SSE 명령어 집합을 사용하여 가우시안 convolution을 구현하고 있다. SIFT알고리즘의 Scale-space를 생성하는 과정에 수행되는 가우시안 Convolution은 연산시간이 과도하게 요구된다.[1] 2D의 가우시안 필터가 영상을 구성하는 모든 셀과 1:1로 연산을 수행하므로 이 연산의 소요시간은 영상의 가로, 세로 길이 그리고 필터의 크기에 비례하여 결정된다. 이 논문에서 제안하는 방법은 연산을 위해 CPU 내부로 한번 읽어 들인 픽셀자료에 대해 가능한 모든 연산을 SSE 명령어 집합을 사용하여 수행함으로써 병렬 연산에 의한 연산시간 절감과 메모리 접근 최소화를 통한 입출력시간 절감을 통해 전체 연산시간을 단축 하였다.

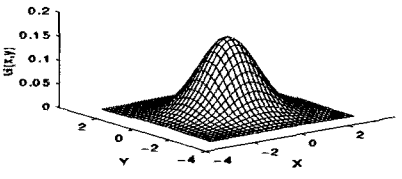
1. 서론

SIFT 알고리즘은 복수의 영상으로부터 특이점을 발견하고 각각의 특이점에 대한 기술자를 생성하여 이들 사이의 일치 여부를 가늠하는 것을 목적으로 한다. 특히 패턴인식분야에서는, 인식하고자 하는 객체를 크기에 구애받지 않고 찾아낼 수 있는 강력한 알고리즘으로 알려져 있다.[2]

Scale space는 다수의 가우시안 convolution 연산을 통해 구축할 수 있다.[3] 이때 소요되는 시간은 영상의 픽셀 개수와 마스크를 구성하는 픽셀수의 곱에 비례하게 된다. SIFT 알고리즘은 특이점 탐색 및 기술자 생성을 위해 scale space를 사용하는데, 로봇 비전분야나, 증강현실 분야와 같은 실시간처리를 요구하는 응용영역에서 SIFT알고리즘을 사용하기위해서는 알고리즘 수행시간을 최대한 단축시켜야 한다. 본 논문에서는 SIFT 알고리즘의 수행시간을 단축시키기 위해 SSE 명령어 집합을 convolution 연산에 적용하는 방법에 관해 소개한다. 본론에서 실험에 사용된 가우시안 필터에 대한 이론을 간단히 요약하고, SSE 명령어 집합을 사용하기위해 개발된 3가지 필터링 방법을 소개한다. 결론 부분에서는 본 논문에서 소개한 3가지 필터링방법의 성능을 SSE 명령어를 사용하지 않는 방법과 연산 소요 시간을 비교하는 방식으로 고찰하였다.

2. 가우시안 커널과 convolution 연산

가우시안 필터링은 기능적으로는 중수필터링과 유사하지만 그림 1과 같은 종 모양의 특이한 커널이



$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

(그림 1) 가우시안 분포곡면과 식

사용된다. 이론적으로 가우시안 분포 값은 항상 0보

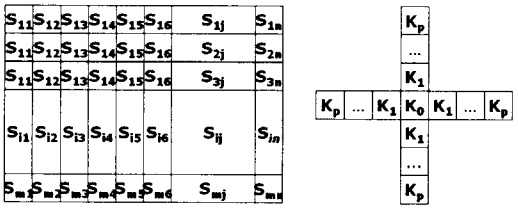
다 크다. 하지만 convolution 마스크는 무한하게 커질 수 없으므로 거의 0에 가까운 값들은 제거함으로써 가우시안 커널을 구한다. 일반적으로 평균값으로부터 3σ까지의 값을 유효한 것으로 취한다. 2차원 등방성 커널은 연산속도를 향상시키기 위해 x, y 성분의 1차원 커널로 분리할 수 있다.

2.1 Convolution

convolution은 영상처리 분야에서 다양하게 이용되는 기본적인 연산으로 수식은 아래와 같다.

$$D_{(i,j)} = \sum_{k=-p}^{p} [\{S_{(i+k,j)} + S_{(i,j+k)}\} K_k]$$

그림 2는 본 논문에서 사용하는 이미지와 커널의 모델이다.



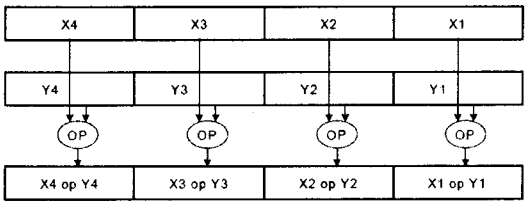
(그림 2) convolution

3. SSE 명령어를 이용한 필터링

가우시안 convolution연산에서 수행되는 곱셈의 횟수는 프로그램 속도와 가장 밀접한 관계를 갖고 있다. 다시 말해, 곱셈의 횟수를 줄일 수 있다면 프로그램의 속도를 높일 수 있다. SSE 명령어를 사용하는 방법은 직접적으로 횟수를 줄이는 것은 아니다. 하지만, 네 번의 곱셈을 한 번의 곱셈 연산시간에 수행함으로써 추가로 수행된 곱셈을 감추는 것은 곱셈의 횟수를 줄이는 것과 유사한 효과를 가져 온다.

3.1 SSE 명령어

인텔에서는 펜티엄 프로세서의 MMX(Multimedia Extension) 기술로 SIMD(Single Instruction Multiple Data)를 처음 소개하였다. 그 후에 펜티엄3 프로세서 계열에서 SSE, 펜티엄4 프로세서에서 SSE2 그리고 SSE3로 진화하면서 더 많은 명령어 집합으로 보다 다양한 데이터 타입을 지원하려고 하였다.[5] 그림 4에서 보는 것처럼 SSE 명령들은 4개의 단정도 부동소수점 데이터들 간의 연산을 동시에 수행함으로써 이론적으로 4배의 효율을 얻는다. SSE2에는 2개의 배정도 부동소수점 데이터들 간의 연산을 지원하는SIMD 명령이 포함되었다.



(그림 3) 전형적인 SIMD 동작

3.2 필터링

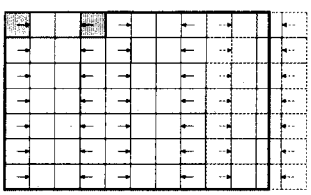
SSE 명령어들은 연산을 수행하기위해 기본적으로 xmm레지스터에 데이터를 로드한다. 본 논문에서는 로드하는 단위 데이터에 따라서 점, 선, 면 필터링 방법으로 구분하고 그 방법을 설명한다.

3.2.1 점 필터링

convolution 연산은 K_0가 놓이는 $S_{(i,j)}$가 중심이 되고 상·하·좌·우 방향에 위치한 픽셀을 로드하면서 각 셀에 오버레이 된 가우시안 커널의 각 요소와 곱의 합을 $D_{(i,j)}$로 저장하는 방식이다. 점 필터링은 이 방법을 역으로 수행한다. 우선, 한쪽 레지스터에는 MOVSS 명령과 Shuffling을 이용하여 동일한 값을 4개 채운다. 그리고 다른 하나의 xmm 레지스터에는 커널 값을 K_0부터 4개씩 로드한다. 그 다음, 곱셈 결과를 $D_{(i,j)}$ 주변 셀에 누적시킨다. 이 방법은 연산 수행시간동안 $S_{(i,j)}$에 오직 한번 접근하면서 SSE 명령어를 사용한다는 특징이 있다.

3.2.2 선 필터링

SSE 명령어 중에는 aligned 16바이트를 한 번에 읽어 들이는 MOVAPS 명령과 alignement를 필요로



(그림 4) 선형 필터링

하지 않는 MOVUPS 명령이 있다. 둘 다 선형 데이

터에 대한 접근만을 지원하는 명령어로써 가우시안 convolution의 수직방향연산이 문제가 된다. 이 문제를 해결하기 위해 선 필터링 방법은 원래영상에 위상 변환을 추가적으로 수행한다. 16바이트씩 데이터를 로드하고 convolution을 정 방향과 역 방향으로 수행하여 그 결과를 그림 4의 →, ← 이 있는 위치에 저장한다. 그림 2에 표현된 수식은 아래와 같이

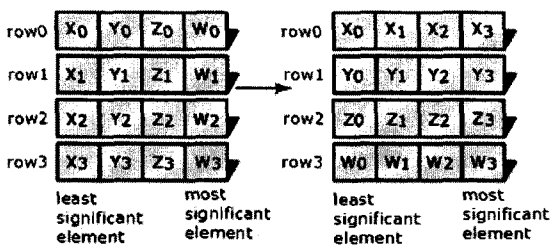
$$D_{(i,j)} = \sum_{k=0}^{h}[\{S_{(i+k,j)} + S_{(i-k,j)} + S_{(i,j-k)} + S_{(i,j+k)}\}K'_k]$$

($k=0$일때 $K'_0 = \frac{K_0}{4}$, 그 외에는 $K'_k = K_k$)

바꿔 쓸 수 있다. 그림 4를 적용시켜서 생각해보면 가로, 세로 길이가 7인 커널의 중심 K_0가 $S_{(1,1)}$에 놓인 경우에 대한 $\sum_{k=0}^{3} S_{(1+k,1)}K_k$ 와 $S_{(1,4)}$에 놓인 경우에 대한 $\sum_{k=0}^{3} S_{(4-k,1)}K_k$을 동시에 구할 수 있다. 결과적으로 메모리의 절반만 접근하면서 완전한 수평방향 convolution 을 수행한 결과를 얻을 수 있다.

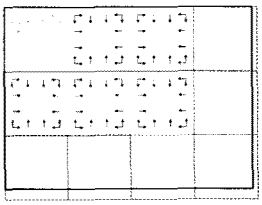
3.2.3 면 필터링

선 필터링 방법은 원본영상의 위상을 변환하는 전·후처리연산이 문제를 안고 있었다. 면 필터링 방법은 이 문제를 해결한다. 이 방법은 4개의 단정도 부동소수를 저장할 수 있는 xmm 레지스터 4개를 이용해 4x4 데이터 블록을 하나의 연산단위로 한다. 이 레지스터 블록이 구성되면 수평, 순·역 방향의 convolution 연산은 물론 수직, 순·역방향 연산도 전·후처리 없이 수행할 수 있다. 그림 5는 전·후처리 대신 레지스터 내부에서 간단하게 수행되는 위상변환 동작을 보여준다.

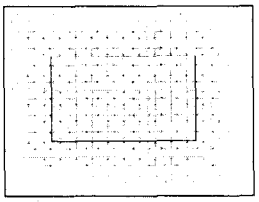


(그림 6) _MM_TRANSPOSE4_PS 매크로 함수를 이용한 매트릭스 변환

그림 6은 첫 번째 블록루프 연산을 마친 상태를 보여주는 간단한 예시이다. 이 경우 1회 블록루프 연산은 6개의 레지스터 블록 연산으로 구성된다. 이미지의 우측과 하단에 점선으로 표시된 블록은 이미지의 경계에 걸쳐있는 블록으로 연산하지 않는다. 블록루프 연산은 처음 $S_{(1,1)}$을 시작점으로 하고 대각선을 따라 $S_{(2,2)}$, $S_{(3,3)}$, $S_{(4,4)}$를 새로운 시작점으로 다음 블록루프 연산을 시작한다. 블록 한 변의 길이만큼 블록 루프가 수행되고 나면 그림 7과 같이 영상 내부에 convolution이 완성된 영역과 그렇지 않은 영역이 형성된다. 미완성 상태의 영역은 convolution 의 정의에 의해 완벽하게 정의될 수 없는 영역에 국한되므로 무시할 수 있다.



(그림 6) 첫 루프연산 후



(그림 7) 최종 결과

4. 실험 및 결과

<표 1> 컴퓨터 환경

프로세서	Intel Pentium4 3.20GHz MMX, SSE, SSE2, SSE3 지원
캐시 메모리	8-way set associative 64byte line size L1 16KByte, L2 1024KByte
메인 메모리	PC3200(200MHz) 1024 MBytes
운영체제	Window server 2003
프로그래밍언어	C++ (Microsoft .net 2005)

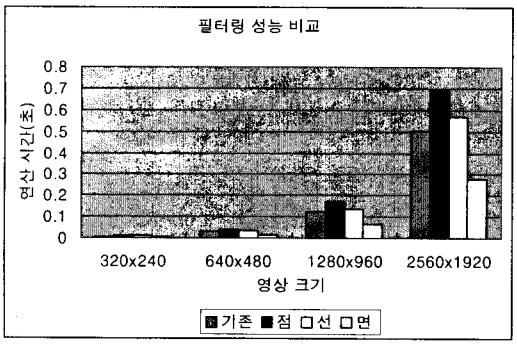
실험에 사용된 컴퓨터 환경은 아래의 표 1에 정리된 내용과 같다. 본 논문에서 소개한 필터링 방법의 성능실험을 위해 해상도가 각각 320x240, 640x480, 1280x960, 2560x1920인 4개의 영상을 준비하였다. 원본 컬러 영상의 각 픽셀은 1바이트 gray-level값으로 변환한 다음 convolution 연산을 위해 단정도 부동소수 형 데이터로 거듭 변환하는 전처리 과정과 convolution 결과 데이터를 gray-level값으로 복원하는 후처리 과정이 필요하다. 성능 비교를 목적으로 SSE를 이용하지 않는 기존의 방법과 본 논문에서 제안하고 있는 SSE를 이용한 점, 선, 면 필터방법의 전처리, convolution 연산, 후처리과정에 소요된 시간을 각각 측정하여 표 2로 정리하였다. 실험 결과 면 필터방식만이 기존의 방법보다 평균 36% 연산시

간 절감효과를 얻을 수 있었고 점, 선 방식의 경우 각각 34%, 63% 연산시간이 추가 소요됨을 확인하였다.

<표 2> 단계별 소요시간 비교 단위:초

가로 세로		기존	점	선	면
320 240	전처리	0.00096	0.00116	0.00216	0.001158
	필터링	0.00711	0.00961	0.00817	0.003308
	후처리	0.00072	0.00076	0.00172	0.000593
	종합	0.00379	0.00153	0.01265	0.005059
640 480	전처리	0.00389	0.00484	0.00962	0.004996
	필터링	0.02962	0.04102	0.03447	0.017511
	후처리	0.00289	0.00307	0.00817	0.002409
	종합	0.0364	0.00893	0.05226	0.024916
1280 960	전처리	0.0155	0.01918	0.04208	0.019362
	필터링	0.12225	0.17188	0.1352	0.0668925
	후처리	0.01162	0.01244	0.0352	0.0097209
	종합	0.14937	0.2035	0.21248	0.095975
2560 1920	전처리	0.06198	0.07568	0.49523	0.077172
	필터링	0.50348	0.69524	0.56777	0.276459
	후처리	0.0465	0.04878	0.34298	0.039276
	종합	0.61196	0.8197	1.40598	0.392907

영상의 크기를 4배수로 증가시켜가며 실험한 결과 선 필터의 경우 영상의 크기가 커짐에 따라 전·후 처리에 소요되는 시간이 급격히 증가하였고, 그 외의 방법들의 연산 소요시간은 영상의 크기와 선형 관계에 있음을 알 수 있었다.



(그림 8) 순수 필터링 성능 비교 그래프

전·후 처리를 제외한 필터링의 성능 측면에서는 면 필터방식이 평균 46% 연산시간 절감효과를 얻은 것으로 확인되었고 점, 선 방식의 경우 각각 38%, 13% 연산시간이 추가 소요되는 것으로 나타나 종합한 결과와 차이가 있음을 확인 할 수 있었다.

5. 결론

점 필터링 방법의 경우 SSE 명령어 활용의 관점에서는 효율적이지만 성능은 그렇지 못하다. 그 원인은 결과 이미지를 만드는 과정에 포함된 누적 연산이 메모리 읽기와 쓰기 동작을 반복하므로 메모리 접근횟수가 기존 방법의 2배가 되기 때문이다. 결과적으로 잦은 메모리 접근은 SSE명령어를 사용함으로써 얻는 연산시간의 절감효과를 상쇄시킴을 알 수 있다. 선 필터링 방법은 점 필터링 방법에 비해 필터링 성능이 개선되었지만 전·후처리 연산에 오버헤드가 가중되어 전체 성능은 점 필터링 방법보다도 떨어진다. 본 논문에서 제시하는 면 필터링 방법은 선 필터링 방법의 문제점을 보완한 방법으로 SSE 명령어를 사용하지 않는 것보다 대략 36%가량 연산시간을 절감할 수 있다. 이 방법은 4x4 블록 단위 필터링 연산을 수행하는 방법으로 가우시안 커널과 같은 가변 크기의 2차원 등방성 커널에 적용 할 수 있다.

6. 감사의 글

본 연구는 DMITRC(Digital Manufacturing and Information Technology Research Center)와 울산대학교 교내 연구비 지원을 받아 수행되었다.

참고문헌

[1] J. Lee "A Study on the SIFT Algorithm Implementation for Real Time Processing" vol 18 Image Processing and Image Understanding 2006, p 315-320

[2] D. G. Lowe "Distinctive image features from scale-invariant keypoints" IJCV, 2004

[3] T. Lindeberg "Scale-space theory: A basic tool for analysing structures at different scales." Journal of Applied Statistics, 21(2):224-270

[4] R. Gonzalez and R. Woods "Digital Image Processing" Addison-Wesley Publishing Company, 1992, p 191.

[5] E. Davies "Machine Vision: Theory, Algorithms and Practicalities" Academic Press, 1990, p 42-44

[6] L. Shapiro "Computer and Robot Vision" Addison-Wesley Publishing, 1992, Vol 1, Chap 7.

[7] Horn "Robot Vision" MIT Press, 1986, Chap 8.

[8] D. Vernon "Machine Vision"Prentice-Hall, 1991, p 59-61

[9] R. Gerber "High-Performance Recipes for IA-32 Platforms" 2nd Edition Intel press, Chap 12.