SIMD기법에 의한 H.263 코덱의 PC기반 실시간 구현

하교동, 남수영, 김남철
경북대학교 대학원 전자공학과

PC-Based Realtime Implementation of H.263 CODEC Using SIMD Method

Gyo Dong Ha, Soo Young Nam, and Nam Chul Kim
Department of Electronic Engineering, Kyungpook National University
E-mail: hgd76@vcl.knu.ac.kr

Abstract

This paper implements H.263 codec using SIMD(single instruction multiple data) method in real time based on PC. This system uses INS algorithm previously proposed by the authors as motion estimation module. SIMD method is used in DCT, IDCT, quantization, motion estimation, and display module. The developed algorithms are implemented using TMN5. Using the above algorithm, H.263 Codec can communicate more than 15 frames/sec in CIF resolution on a Pentium-IV 1.7GHz computer.

I. 서 론

디지털 비디오 적용분야가 점차 넓어짐에 따라 ITU-T H.261, H.263, H.263+와 ISO/IEC MPEG-1, MPEG-2같은 성공적인 표준안들이 발표되었다. 이중 H.263은 64Kbps이하의 저 전송율 비디오 압축 표준으로 채택되었다. 최근 들어 이러한 표준들을 실시간 부/복호화를 위해 ASICs 이나 DSP 플랫폼들이 소개되고 있다. 그러나 하드웨어적인 구조들은 호환성과 추가적인 비용증가라는 문제를 가지고 있다. 이러한 하드웨어적인 제약과 멀티미디어 처리의 증가들로 인해 많은 프로세서들이 멀티미디어 처리를 위한 구조를 가지게 되었다. 예를 들어 인텔 Pentium의 MMX, 선의 Ultra Sparc, 휴렛 패커드의 PA-RISC들이 SIMD(single instruction multiple data) 구조를 사용하여 멀티미디어 데이터들을 처리하고 있다.[4] 본 논문에서는 PC를 기반으로 한 H.263 코덱의 속도를 높이기 위해 두 가지 방법을 제안한다. 첫 번째는 저자들이 제안한 고속 움직임 추정 알고리즘을 적용함으로써 FBMA(full search block matching Algorithm)비해 0.1dB 이하의 화질저하만을 가지면서, 처리 속도면에서는 3배의 향상을 이루었다. 두 번째는 코덱에 SIMD 기법을 적용함으로써, 고속 움직임 추정 알고리즘을 적용한 코데의 처리속도에 비해 1.68배의 향상되었다.

이러한 결과를 바탕으로 비디오 입력단을 제작하고, UDP를 이용한 네트웍 통신을 사용함으로써 실시간 H.263 시스템을 구현하였다. 결과적으로 인텔 펜티엄 IV 1.7GHz에서 CIF 해상도의 실 영상을 초당 15프레임 속도로 양방향 통신을 한다.

II. 시스템 구조

1. 전체 시스템 구조

1.1 시스템 블록도

이 시스템은 양방향 실시간 비디오 통신 시스템을 구현하기 위해 64Kbps이하의 저 전송률 가지는 H.263 코덱을 채택했다. 여기에 카메라로부터 영상 입력을 받기 위한 프레임 그래버를 제작하여 비디오 입력단을 구성했다. 처리된 결과는 UDP를 사용하여 데이터를 전송하고, 결과를 모니터에 디스플레이 한다.

그림 1은 시스템 블록도를 나타낸다. 프레임 그래버는 NTSC와 PAL방식의 비디오 신호를 디지털 신호로 변환하는 목적으로 제작된 PCI 영상 A/D 컨버팅 보드이다. 이 보드는 영상 A/D 컨버터와 PCI Controller기능을 동시에 내장하고 있는 컨넥선트사의 퓨전 878A 칩을 사용하여 초당 30 프레임의 영상을 PC에 전송 할 수 있다. 특히 이 보드는 DMA기능을 지원함으로써 영상 입력단에서의 PC 프로세싱 시간을 최소화하였다. 네크웍통신 방식은 실시간 영상전송에 적합한 UDP를 사용하였다. 예측 부호화와 VLC의 특성상 에러가 전파되는 것을 막기 위하여 패킷의 사이즈를 조절하였으며,

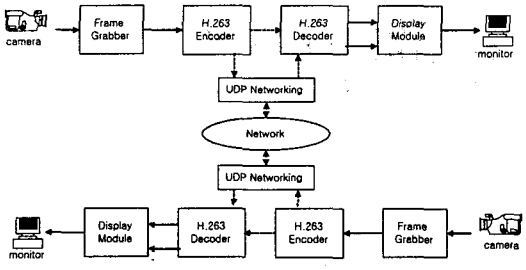

그림 1. H.263 시스템의 블록도

수신측과의 동기화를 맞추기 위하여 개별적인 제어 신호를 두었다.

1.2 동작 및 블록별 계산량

이 시스템은 비디오 입력을 받아서 부호화를 거친 후, 네트웍으로 전송, 수신, 복호화블록을 거쳐 처리된 결과를 디스플레이 하는 과정을 반복적으로 수행하게 된다. 순차적인 데이터 처리를 통하여 송수신 측의 통신속도를 동일하게 유지하여, 연속적인 영상을 일정한 속도로 처리 할 수 있게 구성하였다. 그림 1에서와 같이 이 시스템은 비디오 입력, H.263 코덱, 네트웍 통신, 디스플레이 블록으로 구성되어 있고, 이들 각각의 블록별 계산량은 그림 2에서 확인할 수 있다.

H.263 코덱이 시스템 계산량의 90%이상을 차지하고 있다. 시스템을 고속화하기 위해서 H.263 코덱의 구조를 파악하고, 처리속도를 향상시킬 수 있는 방법을 제안하기로 한다.

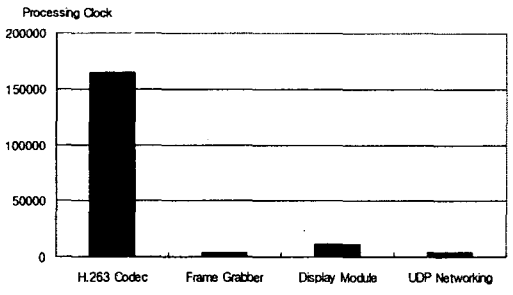

그림 2. 블록별 계산량 비교

2 H.263구조

H.263은 64Kbps이하의 저 전송 동영상 부호화를 위한 표준으로 현재 저 전송률뿐만 아니라 다양한 비트율도 수용할 수 있어 여러 분야에 응용되고 있다. H.263 코덱의 구조는 그림 3와 그림 4와 같다. H.263 부호화에서 부호화를 위한 핵심 알고리즘은 H.261과 MPEG 등 다른 표준 영상 부호화 방식에서와 마찬가지로 움직임 추정, 예측 부호화, 변환 부호화 그리고 양자화 처리로 이루어져 있다. 입력 동영상이 부호화기에입력되면 화면내 부호화(intra frame coding)에서는 움직임 추정 없이 바로 DCT 과정과 양자화과정을 거쳐 VLC를 하여 공간적 압축만 하여 전송한다. 화면간 부

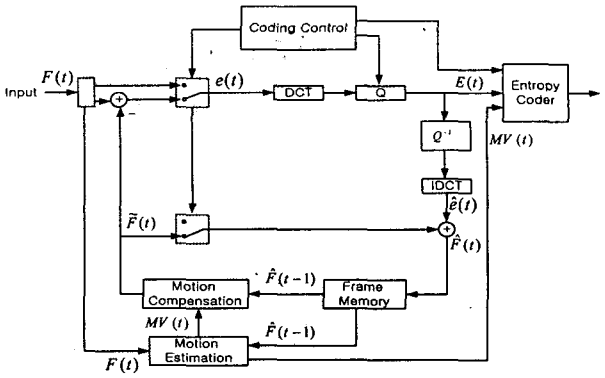

그림 3. H.263의 부호화기

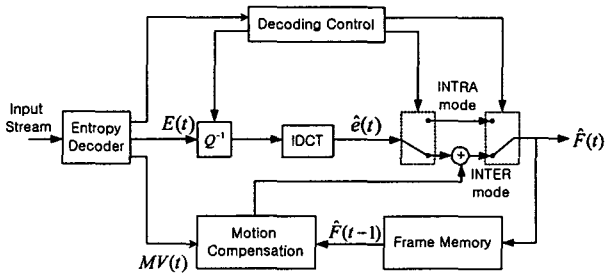

그림 4. H.263의 복호화기

호화(inter frame coding)에는 역양자화와 IDCT를 거친 복원된 영상을 메모리에 저장하고, 메모리에 저장된 영상값과 현재 영상의 값을 이용하여 움직임 벡터를 추정하고, 다시 추정된 움직임 벡터를 이용하여 복원영상을 만들어 낸다. 복원된 영상과 현재영상의 차를 DCT와 양자화과정을 거쳐 결과와 움직임 벡터를 함께 VLC를 하여 전송함으로써, 시공간 압축을 하게 된다. 복호화 과정은 부호화 과정과 마찬가지로 움직임 추정 벡터를 이용하여 이전영상으로부터 복원된 영상과 차영상을 합하여 역양자화, IDCT를 거쳐 영상을 복원한다.

H.263 통신 시스템 전체 계산량 중 ME(motion estimation) 함수가 차지하는 비율이 전체의 시스템 처리시간의 58%를 차지한다. 실시간 통신을 구현하기 위해서는 고속 움직임 추정 알고리즘의 적용이 필요하다. 따라서 H.263내에서 움직임 알고리즘, FBMA를 본 저자들이 제안하는 INS(initial point and neighbor search)[1] 알고리즘으로 대체하기로 한다.

3. 고속 움직임 추정 알고리즘

본 저자들이 제안한 INS 알고리즘은 초기 탐색위치를 FBMA에 의한 움직임 벡터에 근사하게 추정함으로써 고속 움직임 추정을 수행한다. 본 알고리즘에서는 먼저 초기 탐색위치를 선택하기 위하여 세 개의 후보 움직임 벡터를 구한다. 첫 번째 후보 움직임 벡터는 영

상을 2:1 부표본화여 얻은 부표본화 블록에 대하여 FBMA를 사용하여 얻고, 두 번째와 세 번째 후보 움직임 벡터는 이전 영상의 같은 위치 블록의 움직임 벡터와 현재 참조 블록의 주위 블록들의 움직임 벡터 조합으로 얻는다. 이렇게 찾아낸 각 후보 움직임 벡터의 SAD를 구하여, 그중 최소 SAD를 갖는 후보 움직임 벡터를 초기 탐색 위치로 선택한다. 그리고 초기 탐색 위치를 중심으로 8개의 주위 탐색 위치에 대하여 SAD를 구하고, 그 중 최소 SAD를 가지는 탐색 위치를 움직임 벡터로 추정한다. 그림 5는 INS알고리즘의 움직임 벡터 추정 과정을 나타낸 것이다.

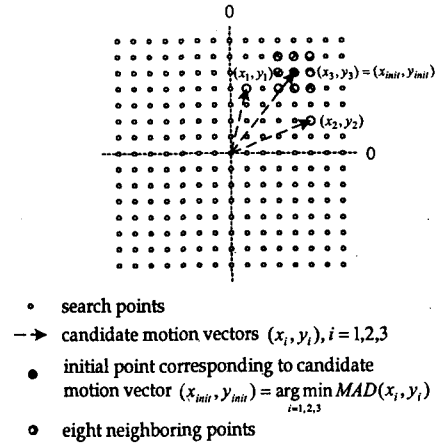

그림 5. 초기 위치와 주위 8개 탐색점에 대한 탐색

실험 결과로부터 INS 알고리즘이 FBMA와 거의 동일한 PSNR을 유지하면서, 계산량을 대폭 줄임으로서 실시간 통신에 적합한 고속 움직임 추정 알고리즘임을 확인할 수 있었다. 표 1는 INS 알고리즘의 보상된 영상의 PSNR과 FBMA에 대한 계산량의 비를 나타낸다. INS 알고리즘은 각 참조 블록에 대하여 일정한 계산량으로 움직임 벡터를 탐색하므로, 특히 하드웨어적인 구조에 적합하다. 이러한 결과를 바탕으로 실시간 통신을 위한 고속 움직임 알고리즘으로 INS 알고리즘을 이용하였다. 표1에서 보듯이 INS 알고리즘은 FBMA에 비해 PSNR의 저하가 0.1dB이하임을 알 수 있다. 그에 비해 한 픽셀의 움직임 벡터를 추정하기 위한 평균 계산 횟수를 나타내는 ENPS의 개선량은 매우 큰 것을 볼 수 있다.

표 1. FBMA와 INS 알고리즘의 PSNR과 계산량 비교 (ENPS : Equivalent Number of Search Points for each reference block)

	Carphone		Foreman		Stefan	
	Average PSNR	Average ENSP	Average PSNR	Average ENSP	Average PSNR	Average ENSP
FBMA	31.53	782	29.64	782	22.40	782
INS with PDE-SEA	31.44	10	29.50	11	22.33	18

표 2는 고속 움직임 추정 알고리즘을 적용한 H.263 시스템의 주요 함수별 계산량을 비교하였다. 전체 시스템의 58%의 계산량을 차지하던 움직임 추정 블록이 3.7%로 급격한 감소를 보였다. 그 결과 H.263시스템의 속도는 FBMA을 사용하였을 경우보다 3배의 속도 향상이 되었다. 고속 움직임 추정 함수의 적용으로 CIF 영상을 초당 9~10 프레임의 속도로 처리가 가능하게 되었지만, 실시간 구현을 위해서는 추가적인 고속화가 필요하다. 움직임추정 블록의 계산량이 급격히 감소함에 따라 상대적으로 디스플레이 블록이 시스템 전체 계산량의 42.6%를 차지하였다. 또한 DCT, 양자화, 디스플레이, IDCT, 역양자화 비중이 커졌음을 알 수 있다. 이 블록들은 영상 데이터들의 반복적인 계산이 적용되므로, 인텔의 MMX 와 스트리밍 SIMD구조에 SIMD기법을 적용함으로써 계산량을 줄였다.

표 2. H.263시스템 주요 함수별 계산량 비교 (네트웍 통신과 프레임 그래버는 제외)

Block	Display	DCT	ME	Quanti-zation	IDCT
Clock	12756	3540	1099	3363	2901
Ratio [%]	42.6	11.8%	3.7	11.2%	9.7
Block	Dequant-zation	**Subtotal**	Other Function	**Total**	
Clock	2341	**26000**	3958	**29958**	
Ratio [%]	7.8	**86.8**	13.2	**100**	

III. 고속화

1. SIMD

멀티미디어 데이터는 바이트나 워드같이 작은 데이터 타입을 가지고 반복적인 연산을 주로 처리한다. 이러한 데이터 처리를 위하여 하나의 명령으로 여러 개의 데이터를 한번에 처리하는 SIMD방식이 많이 사용되고 있다. 현재는 멀티미디어 처리를 위해 만들어진 DSP뿐만 아니라 많은 범용 프로세서도 SIMD를 위한 구조를 가지고 있다. 인텔은 멀티미디어 데이터 처리를 위하여 MMX 와 스트리밍 SIMD 구조를 만들고, 새로운 데이터 타입과 명령어들을 제안하였다. 본 논문에서는 이러한 구조와 명령어들을 이용하여 각 함수들을 고속화하였다.

2. SIMD를 이용한 함수의 고속화

움직임 추정블록의 감소로 인해 디스플레이, DCT[5], IDCT, 양자화, 역양자화부분 상대적으로 비중이 커졌다. 이러한 함수들을 MMX[6]와 스트리밍 SIMD구조를 이용하여 한번에 4 ~ 8 픽셀씩 처리함으로써 속도를 향상 시켰고, 또한 새로운 명령어들을 활용함으로써 처리시간을 단축시켰다.

IV. 실험결과

이 실험은 인텔 펜티엄 IV 1.7GHz에서 Foreman, Stefan, Carphone CIF(320 X 240) 영상을 대상으로 하였다. 각 영상의 길이는 280 프레임이며, 각 5회씩 수행하여 평균치를 사용하였다. 동일한 영상과 일정한 실험결과를 가지기 위하여 프레임 그래버는 파일 입력으로 대체하고, 네트웍 통신은 실험에서 제외하였다. 표 3는 시스템에서 계산량을 많이 차지하는 아래 다섯 개의 블록을 SIMD를 이용하여 고속화 한 결과이다.

표 3. SIMD를 이용한 고속화 전/후의 성능 비교

Block	Display	DCT	ME	Quantization
고속화 전 (Clock 수)	12756	3540	1099	3363
고속화 후 (Clock 수)	5848	2558	447	871
γ	2.19	1.39	2.46	3.86
Block	**IDCT**	**Subtotal**	**Total**	
고속화 전 (Clock 수)	2901	23659	29958	
고속화 후 (Clock 수)	1851	11575	17874	
γ	1.57	2.04	1.68	

($\gamma = \frac{\text{고속화 전 } Clock \text{ 수}}{\text{고속화 후 } Clock \text{ 수}}$, ME : INS 알고리즘)

위 다섯 개의 함수들의 속도 향상 비율은 200%이 이른다. 하지만 고속화를 하지 않은 블록들의 영향으로 인해 전체적인 속도 향상율은 168%의 향상율을 가졌다. FBMA를 이용한 H.263 코덱은 초당 3~3.5 프레임을 처리할 수 있다. 이 시스템에 고속 움직임 추정 알고리즘을 적용하고, SIMD기법을 이용하여 코드를 고속화 한 결과, 표 4에서와 같은 처리속도를 나타냈다.

표 4. 고속화 전/후의 처리속도와 성능향상 비율

	Forman CIF(320X320) 280 frames			Stefan CIF(320X240) 280 frames		
Coding	Fuop	Fuop	γ	Fuop	Fuop	γ
Frame/ sec	9.8	16.51	1.68	9.23	15.15	1.64

Fuop : Un-optimization Frame Rate,
Fop : Optimization Frame Rate
γ : Ratio of Speed Improvement
ME : INS Algorithm

V. 결론

본 논문에서는 H.263 양방향 비디오 코딩 시스템을 구현하고, 속도 향상을 위한 알고리즘을 제안하였다. 시스템은 표준 H.263 소스, 프레임 그래버, UDP 네트웍 통신을 이용하여 전체시스템을 구현했다. 시스템의 속도를 향상시키기 위하여 고속 움직임 추정 알고리즘(INS)을 채택하였고, 추가적인 속도 향상을 위하여 인텔의 MMX and 스트리밍 SIMD구조를 이용하여 SIMD기법을 적용하였다. 결과적으로 이 시스템은 인텔 펜티엄 IV 1.7GHz에서 CIF영상을 초당 15 프레임의 속도로 양방향 통신을 한다.

참고문헌

[1] 남수영, 김석규, 임채환, 김남철, "초기 탐색 위치의 효율적 선택에 의한 고속 움직임 추정," 한국통신학회 논문지 8월 게재 예정.

[2] *Intel's MMX Technology Programmers Reference Manual, 1999.*

[3] *Intel Architecture Software Developer's Manual, Vol. 2. Instruction Set Reference.*

[4] Berna Erol, Faouzi Kossentini, Hussein Alnuweiri," Efficient coding and mapping algorithm for software-only real-time video coding at low bit rates," in proc. *IEEE Trans. on Circuit and System for Video Technology* , Vol.10, No.6 Sep. 2000.

[5] Intel's MMX application notes, "Using streaming SIMD extension in ad fast DCT algorithm for MPEG encoding," Version 1.2 Jan. 1999.

[6] Intel's MMX application notes. [Online] Available : http://developer.intel.com/drg/mmx/appnotes.