Three-step 알고리즘을 이용한 H.263 기반의 움직임 측정

윤성규, 유환종, 임명수, 임영환
숭실대학교 컴퓨터학과

H.263 Motion Estimation using the three-step algorithm

Sung-Kyu Yoon, Hwan Jong Yu, Myung-Su Lim,Young-Hwan Lim
Dept.of Computer Science Soongsil University

요약

영상 압축 기법에는 여러 가지 알고리즘이 적용되고 있다. 이런 알고리즘들에는 주파수 영역 중복을 제거하기 위한 DCT, 시간 중복성 제거를 위한 움직임 측정, 압축기법에 의해서 만들어진 정보를 부호화하는 VLC 들이 있다. 이런 부호화 알고리즘들은 부호화기를 구현하는데 많은 시간을 요구하며 특히 움직임 추정은 부호화기의 절반에 가까운 시간을 소비한다.움직임 측정 기술의 복잡도는 search algorithm, cost function, search range parameter 의 요인으로 나타낼 수 있다. 본 논문에서는 기존의 Full Search 알고리즘 대신에 three-step 알고리즘을 사용하여 움직임 측정 시간을 줄였다. Full Search 알고리즘은 search area 에서 모든 지역에 대해 cost function 를 사용하여 이전 블록과 얼마나 유사한지를 조사한다. 따라서 이전 블록과 가장 유사한 부분을 찾는 좋은 방법이지만 그 만큼 시간이 많이 사용한다. Three-step 알고리즘은 search area 의 일정 지역에 대해 cost function 를 사용하여 이전 블록과의 유사성을 찾는 fast 알고리즘이다. Three – step 알고리즘을 사용한 경우 기존의 full search 알고리즘을 사용할 때 보다 60% 정도의 시간이 단축되었다. 그리고 생성되는 압축 데이터의 크기는 full search 알고리즘을 사용할 때 보다 많이 차지한다. 생성되는 H.263 파일의 화질에서는 Three -step 알고리즘을 사용한 경우일지라도 full search 알고리즘을 사용한 경우와 거의 비슷한 화질을 보여준다.

1. 서론

H.263 은 저 전송률을 가지는 통신 선로(64kbps 이하)에서 영상 회의나 영상 비디오 전화 등을 위한 멀티미디어 통신 서비스의 동영상 부분에 대한 압축에 쓰일 수 있는 부호화된 표현에 대한 권고안이다. 기본적인 영상 소스 코딩 알고리즘은 H.261 에 기반하고 있다. 그러나 H.261 에 비하여 같은 화질의 영상에 대해 거의 반정도의 비트를 생성한다. 이런 lower data-rate 를 발생시키는 것은 여러 가지 향상된 알고리즘을 사용하기 때문이다.
영상 압축 기법에는 여러 가지 알고리즘이 적용되고 있다. 이런 알고리즘들에는 주파수 영역 중복을 제거하기 위한 DCT, 시간 중복성 제거를 위한 Motion Estimation, 압축기법에 의해서 만들어진 정보를 부호화하는 VLC 들이 있다. 이런 부호화 알고리즘들은 부호화기를 구현하는데 많은 시간을 요구하며 특히 Motion Estimation 은 부호화기의 절반에 가까운 시간을 소비한다.
따라서 본 연구에서는 부호화기의 수행 속도를 향상시키기 위해서 가장 수행 속도를 많이 차지하는 Motion Estimation 의 알고리즘을 살펴보고 속도를 향상시키기 위해 fast 알고리즘 중에서 three-step 알고리즘을 구현하고 비교하는 것을 목적으로 한다.

2. 연구 배경

2.1. 움직임 측정

움직임 측정 (Motion Estimation)은 인코딩 과정에서 움직임 벡터 (Motion Vector)를 찾아내는 과정이다. 즉 현재의 매크로 블록이 이전 블록의 어디에 위치에 있는지를 찾아야 하는데 찾는 과정을 움직임 측정이라 하고 값을 벡터로 표시하는 것이다. 움직임 측정은 모든 매크로 블록마다 수행이 된다.
움직임 측정 기술의 복잡도는 세가지 요인으로 나타낼 수 있다.

- Search algorithm
- Cost Function
- Search range parameter

Search 는 움직임 측정의 가장 기본이 되는 기술이다. 현재 매크로 블록이 이전의 어느 위치에 있는지를 찾기 위해서는 이전 프레임의 Search Area 에서 적당한 알고리즘을 수행시켜야 한다. 보통 search algorithm 으로 block matching 알고리즘을 사용한다. 또한 Search Area 의 범위를 결정해 주는 것이 search range parameter 이다. Search range parameter 는 매크로 블록 가장 자리 사이의 거리를 나타낸다. 일반적으로 H.263 에서 search range parameter 값은 6 이다. 이전 프레임의 search area 에서 매크로 블록 만큼의 데이터를 찾아서 매크로 블록을 비교해서 유사성이 얼마나 있는지를 측정해야 하는데 이 경우 Cost function 를 사용하여 알아볼 수 있다.
위에서 언급한 3 가지 요인이 움직임 측정을 하는 가장 중요한 부분이다. 그러므로 만약 영상 압축의 움직임 측정에서 속도를 개선하고자 한다면 위의 3 가지 요인의 복잡도를 줄이면 상당한 효과를 볼 수 있을 것이다.

2.2. 문제점 분석

H.263 움직임 측정 시에 위에서 언급한 3 가지 요인이 인코딩 속도에

얼마나 영향을 많이 주는지 TMN 에서 제공하는 자료를 가지고 속도 분석을 해 보았다. TMN 에서 제공하는 실행 파일을 사용하여 300 프레임을 압축하였다. TMN 의 Source file 의 Function 별로 분석을 하였다.

Func Time	%	Hit Count	Function
136448.978	66.7	123231700	_SAD_Macroblock (block_functions.obj)
12014.309	5.9	109998	_idctref (dct.obj)
8165.945	4.0	29700	_MotionEstimation (mot_est.obj)
7331.081	3.6	24469	_FindHalfPel (mot_est.obj)
4916.204	2.4	178794	_Dct (dct.obj)

표 1 Full Search 알고리즘 Function time & Hit Count

위의 데이터는 Visual C++의 profile 에서 제공하는 데이터이다. 위에서 볼 수 있는 것처럼 인코더의 전체 수행의 65% 이상을 차지하는 것이 SAD_Marcoblock 이라는 함수이다. SAD 란 Sum of Absolute Difference 의 약자로, 이 함수는 앞에서 말한 cost function 에 해당하는 함수이다.

$$SAD = \sum_{k=1}^{16}\sum_{l=1}^{16}\left|B_{i,j}(k,l) - B_{i-u,j-v}(k,l)\right|$$

그런데 더 주의 깊게 보아야 할 것은 이 함수의 호출 횟수이다. 전체 수행 동안에 23231700 번 호출이 되었다. 이것으로 미루어 보아 SAD_Macroblock 이라는 Cost function 자체 수행 시간이 많은 것이라기 보다는 Cost function 이 많이 호출이 되었기 때문에 수행 시간이 오래 걸렸다고 볼 수 있다. Search 를 수행 할 때마다 데이터를 비교하기 위해서 Cost function 이 호출이 되므로 그 만큼 Search 횟수가 많다는 것이다. 실제로 TMN Encoder 는 Block matching 알고리즘 중에서 Exhaustive search 알고리즘을 수행한다. Exhaustive search 알고리즘 방법은 현재 매크로 블록이 이전에 어디에 있는 알 수 있는 가장 좋은 방법이다. 하지만 모든 위치에서 일일이 매크로 블록을 비교하여 cost function 를 실행하기 때문에 시간이 많이 걸린 것이다. Exhaustive search 알고리즘을 사용할 경우 복잡도는 $O(2d+1)^2 \approx O(d)^2$ 이다. 이것은 현재 프레임의 한 매크로 블록에서 움직임 측정을 할 경우이다. 움직임 측정은 하나의 매크로 블록에서만 하는 것이 아니라 이미지 프레임의 모든 매크로 블록에서 수행되는 것이 때문에 전체 인코더 측면에서 볼 경우에 수행 시간이 많이 소비된다.

2.3. 연구 방향

위의 문제점에서 본 것처럼 기존의 H.263 인코더 실행 시 가장 문제점이 되는 부분은 Search algorithm 에 있다. 움직임 벡터를 찾기 위한 알고리즘에는 Exhaustive search 알고리즘처럼 Search area 의 모든 부분을 Search 하는 Full search 알고리즘뿐만 아니라 일정 지역만 search 하는 Fast search 알고리즘이 존재한다. 그러므로 인코더의 수행 속도를 저하시키는 Full search 알고리즘을 대치하여 Fast search 알고리즘을 사용하여 인코더의 수행 속도를 향상 시킬 수가 있을 것이다.
따라서 이번 연구의 목표는 fast 알고리즘 중에서 three- step 알고리즘을 사용하여 움직임 측정 수행 속도를 향상시키는 것이다.

3. 구현

3.1. three step 알고리즘

three step 알고리즘은 세 단계로 나누어진다.

➢ Step 1

- Search area 의 중심에서 weight 를 구한다. (Cost function 를 실행)
- Step size 를 결정 , search range parameter 를 2 로 나눈다.
- 센터를 중심으로 8 개의 위치를 센터로부터 step size 만큼 떨어진 위치로 한다.
- 각각의 위치에서 cost function 를 구하고 그 중에서 가장 좋은 걸 선택한다.

➢ Step 2

- 선택한 위치를 center 로 설정한다.
- Step size 를 조절한다. (현재 step size / 2)
- 센터를 중심으로 다시 8 개의 위치를 결정하고 cost function 를 구한다.

➢ Step 3

- Step size 가 최소가 될 때까지 step2 를 실행한다.

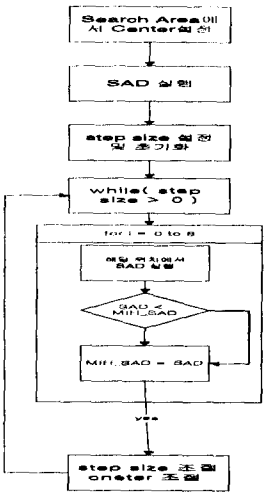

그림 1 Three - Step 알고리즘 순서도

3.2. 테스트 환경

이 연구의 구현 환경은 tmn 3.2 인코더 소스를 기반으로 구성하였다. 사용한 머신은 pentinum 350MHz, 128M, Windows NT 4.0 에서 테스트를 실행해 보았다. YUV 데이터는 4 : 1 : 1 planner 데이터를 사용하였으며 움직임에 따라 움직임이 적고, 많고 , 중간인 데이터를 생성하여 사용하였다. 모든 같은 옵션을 사용하였으며 IPP.. 프레임이 생성되게 하였다. YUV 300 개의 프레임을 압축시켜 소비되는 시간을 측정하였다.

4. 결과

full search 알고리즘과 구현한 three - step 알고리즘을 서로 비교하였다. 비교 대상은 압축 시간과 압축된 크기를 가지고 비교를 하였다. 이 경우 압축된 시간은 디버그 모드에서 실행한 시간을 비교한 것이므로 실제 압축 시간이 아니다. 압축 시간은 상대적인 비율로 비교를 하면 될 것이다.

Function time	%	Hit Count	Function
139851.831	69.5	25787187	SAD_Macroblock
10784.816	5.4	32959	FindHalfPel
7747.355	3.9	334	ReadImage
5600.790	2.8	198396	Dct

표 2 Function time & Hit Count (Full Search algorithm)

Function time	%	Hit Count	Function
12416.695	20.8	956043	SAD_Macroblock
10025.426	16.8	31162	FindHalfPel
5306.390	7.1	198396	Dct
4224.106	5.6	198396	Quant_blk

표 3 Function time & Hit Count (Three - Step algorithm)

표 3 은 움직임이 적은 데이터를 가지고 값을 산출한 것이다. 표에 나타난 것처럼 전체 수행 시간에서 SAD_Marcoblock 이 수행된 시간이 20% 으로 줄어 들었다. 이유는 호출된 횟수가 2587187 에서 956043 번으로 줄어 들었다.

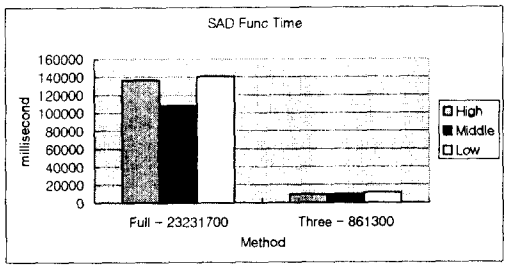

그림 2 SAD Function time Compare

그림 2 는 SAD_Macroblock 의 전체 수행 시간을 비교한 것 이다. Three - step 알고리즘은 영상의 움직임이 적고,많고, 중간인 것에 상관없이 실행된 횟수가 같다. 그러므로 상대적으로 full search 알고리즘 보다 수행 시간이 월등히 줄어든 것을 알 수 있다.

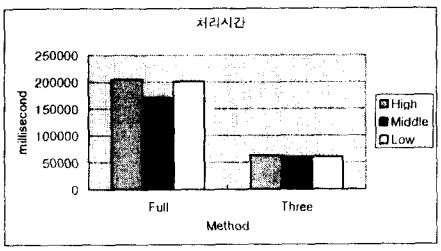

그림 3 Processing time

전체적인 수행 시간을 비교해서 three -step 알고리즘이 full search 알고리즘 보다 수행시간이 적은 걸 알 수 있다. 또한 움직임의 대소에 관계없이 three - step 알고리즘의 수행 시간은 거의 비슷한 것을 알 수 있다.

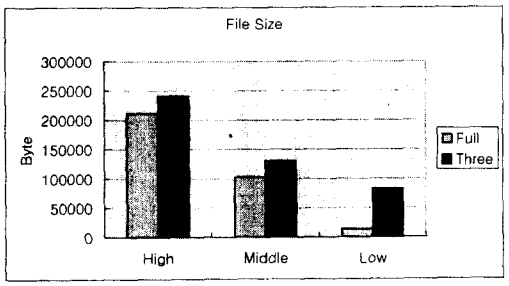

그림 4 Compare compressed file size

위의 그림은 압축된 데이터들의 크기를 나타낸 것이다. Full search 알고리즘인 경우 발생되는 압축 데이터의 크기가 적은데 이것은 exhaustive 알고리즘 방법을 사용한 경우 가장 최적의 벡터를 찾기 때문에 해당되는 매크로 블록이 intra 코딩이 될 확률이 적기 때문이다.움직임이 많을수록 압축 데이터의 크기가 큰 것을 알 수 있다. Full search 알고리즘이나 three-step 알고리즘이나 모두 마찬가지이다. 하지만 유심히 살펴 보아야 할 것은 움직임이 많아질수록 full search 알고리즘과 three-step 알고리즘의 압축 데이터의 크기의 차이가 적어지는 것을 알 수 있을 것이다. three - step 알고리즘 사용할 경우 움직임이 많은 경우에는 full search 알고리즘을 사용할 때와 압축 크기에 대해서는 많은 차이를 두지 않는다. 하지만 수행 속도는 three-step 알고리즘을 사용할 경우 월등히 향상됨을 알 수 있다. 즉 움직임이 많아지는 데이터를 처리할수록 three-step 알고리즘이 효율적인 것을 알 수 있다.

5. 결론

three - step 알고리즘을 사용하여 움직임 측정을 실행한 경우 full search 알고리즘보다 실행 속도면에서 많이 향상되는 것을 알 수 있다. 이는 full search 알고리즘이 $(2p+1)^2 \approx O(p^2)$ 복잡도를 나타내는 반면에 three-step 알고리즘은 $O(8\log P) \approx O(\log p)$ 복잡도를 가진다. 그러므로 three-step 알고리즘의 시간 복잡도가 더 적기 때문에 수행 시간에 많은 비중을 차지하는 움직임 측정에서 많은 효과를 볼 수 있었다.
압축 데이터의 크기에 대해서는 three - step 알고리즘이 full search 알고리즘을 사용한 경우보다 데이터 양이 많이 발생된다는 것을 알 수 있다. 이는 three - step 알고리즘이 가장 최적의 위치의 벡터를 구하는 것이 아니라 근접한 위치를 구하여 해당 매크로 블록의 cost function 를 수행하기 때문이다. cost function 의 weight 가 임계값을 벗어나는 경우에는 압축을 하지 않는 intra 코딩을 수행한다. 따라서 이런 intra 코딩을 하는 매크로 블록이 많아지므로 인코딩후의 압축 데이터 크기가 증가되는 것이다. 하지만 이 경우 움직임이 많은 데이터를 처리할 경우는 압축 데이터의 크기가 full search 알고리즘을 수행한 결과와 많은 차이가 나지 않는다는 것을 알 수 있다.
움직임 측정시에 고려해야 할 것이 화질과 효율성이다. Three - step 알고리즘의 경우 full search 알고리즘을 사용할 때보다 수행 시간이 좋아지기 때문에 효율성이 좋아진다고 할 수 있다. Full search 알고리즘과 three - step 알고리즘의 압축 데이터의 화질에 대해 비교한 결과는 육안으로 확인한 경우 거의 차이가 나지 않는다는 것을 확인할 수 있다. Three - step 알고리즘을 사용한 경우에도 현재와 이전의 매크로 블록이 차이가 많이 나는 경우에는 압축을 하지 않고 현재 매크로 블록의 데이터를 그대로 사용하기 때문이다. 이 경우 화질에는 차이가 나지 않고 단지 압축되는 데이터의 크기에만 영향을 준다.

6. 참고문헌

[1] Motion Estimation Algorithms for Video Compression
[2] Digital Image Compression Algorithms and Standards
[3] ITU Telecom. Standardization Sector of ITU, "Video Coding for Low Bitrate Communication", ITU-T Recommendation H.263, March 1996.
[4] ITU Telecom. Standardization Sector of ITU, "Video Coding for Low Bitrate Communication", Draft ITU-T Recommendation H.263 Version 2, September 1997.
[5] ITU Telecom. Standardization Sector of ITU, "Video Codec Test Model Near-Term, Version 8 (TMN8), Release 0", H.263 Ad Hoc Group, June 1997.
[6] G. Cote, B. Erol, M. Gallant and F. Kossentini, "H.263+: Video Coding at Low Bitrates", Submitted to IEEE Transactions on Circuits and Systems for Video Technology, October 1997.
[7] G. Cote, M. Gallant and F. Kossentini, "Efficient motion vector estimation and coding for H.263-based very low bit rate video compression", Submitted to IEEE Transactions on Image Processing, June 1997.