H.263 압축 속도 향상과 영상 복원용 화질 보상 연구

(An Enhancement of the Encoding Speed and a Compensation of Decoded Video Quality for H.263 Codec)

윤 성 규 † 강 의 선 †† 유 환 종 ††† 임 영 환 ††††

(Sungkyu Yoon) (Euiseon Kang) (Hwanjong Yu) (Younghwan Lim)

요 약 H.263 압축 방식은 실현하는데 여러 가지 문제가 있지만 그 중에서 대표적인 것은 인코딩 과정에서의 압축 시간이 오래 걸린다는 것이고 다른 한 가지는 과도한 압축률에 의한 복원된 이미지의 화질 저하이다. 이 논문에서는 H.263에서의 압축 속도 향상과 복원 이미지의 화질 보상에 대한 두 가지 새로운 방법을 제안하였다. 압축 속도를 향상시키기 위해서 움직임 벡터를 찾는 알고리즘을 개선하여 새로운 4단계 탐색 알고리즘 제안하였다. 또한 화질을 보상하기 위해 디코더에서 블록 아티팩을 제거하고 복원 이미지를 선명하게 하는 알고리즘을 제안하였다. 여기서 화질 보상은 원본 이미지와 동일하게 만드는 것이 아니라 인간이 더 좋은 영상으로 인식하도록 하는 걸 목적으로 한다. 우리가 제한한 알고리즘에 의해서 압축 속도는 초당 2.5에서 17 프레임으로 증가하였고 블록 아티팩을 제거하고 명암 대비를 높임으로써 보기 좋은 영상을 제공하였다.

Abstract For practical software implementation of the H.263 standard, the outstanding problem is that it requires much time in the encoding process. Another problem is that the image quality of a decoded frame is excessively degraded due to the requirement of the very high compression rate. This paper describes two new methods for enhancing the encoding speed and for compensating quality of decoded data. To enhance encoding speed, we suggest a new four-step search algorithm that improves algorithm of finding motion vector. Also is suggested the algorithm that reduces block artifacts and makes a decoded image clear. Our goal of the quality compensation is not to make the decoded video frame identical to the original video frame but to make it perceived better through human eyes. As shown in the experiment result, the new four-step search algorithm improves encoding speed from 2.5 fps to 17fps. And the quality compensation provides better video quality because of reducing blocking artifacts and enhancing the contrast.

1. 서 론

H.263 압축 방식은 실현하는데 여러 가지 문제가 있지만 그 중에서 대표적인 것은 인코딩 과정에서의 압축 시간이 오래 걸린다는 것이고 다른 한 가지는 과도한 압축률에 의한 복원된 이미지의 화질 저하이다. H.263에서는 시간적 중복을 제거하기 위해 움직임 추정(motion estimation)을 사용하고 공간적 중복을 제거하기 위해 DCT를 사용한다. 표 1은 Full Search 블록 정합 알고리즘을 사용한 H.263의 함수별 실행 시간을 나타낸다.

표 1에서 볼수 있듯이 움직임 추정(_Motion Estimation)이 압축 시간의 70%을 소비한다[1]. 그래서 압축 시간을 줄이기 위해 움직임 추정 알고리즘에 대한 연구가 활발히 진행되었다. H.263에서는 블록 정합 알고리즘(Block matching algorithm)이 사용되는데 전역 탐

† 정 회 원 : (주)글로벌시스템 연구원
anakin@media.soongsil.ac.kr
†† 비 회 원 : 숭실대학교 컴퓨터학과
kanges@media.soongsil.ac.kr
††† 비 회 원 : 이지시스템부설연구소 연구원
bewell@dreamwiz.com
†††† 종신회원 : 숭실대학교 컴퓨터학과 교수
yhlim@computing.soongsil.ac.kr
논문접수 : 2000년 8월 16일
심사완료 : 2001년 4월 2일

표 1 각 함수별 실행시간과 호출 횟수

Func Time	%	Func+Child Time	%	Hit	Count Function
70378.454	67.4	40378.454	67.4	7722000	_SAD_Macroblock
5297.207	5.1	5297.207	5.1	32244	_idctref
4927.666	4.7	76685.679	73.4	9801	_MotionEstimation
4275.228	431	4275.228	4.1	8787	_FinalHalfPel
2978.313	2.9	2978.313	2.9	101	_ReadImage
2437.030	2.3	2437.030	2.3	59400	_Dct
2135.137	2.0	1000015.914	95.7	99	_CodeOneOrTwo

색 알고리즘(full search algorithm), 3 단계 고속 탐색 알고리즘(fast three-step search algorithm), 새로운 3 단계 고속 알고리즘, 4 단계 고속 탐색 알고리즘(fast four-step search algorithm)이 대표적인 방법이다. 여기서 4 단계 고속 알고리즘은 적은 계산량과 좋은 화질을 제공하여 최근에 많이 사용되는 방법이다.

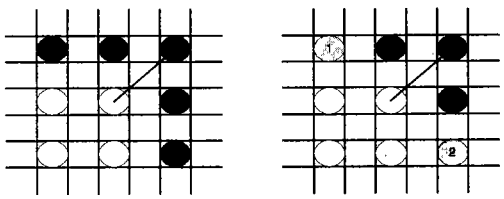
또한 H.263은 저 대역에서의 실시간 전송을 지원하기 위해 압축률을 과도하게 높여야 한다. 이 경우 원본 이미지의 손실이 많아지고 그 결과 복원시 화질 저하가 필연적으로 나타난다. 특히 근접한 블록 경계상의 데이타의 불연속으로 인해 이미지가 블록으로 나누어져 보이는 블록 아티팩 현상을 발생한다. 블록 아티팩은 DCT를 사용하는 압축 방식에서 나타나며 블록 경계상의 픽셀 데이타의 상관 관계를 고려하지 않고 DCT가 수행되기 때문에 발생한다[2]. 블록 아티팩을 개선하는 방법에는 블록의 경계값이 서로 같은 값으로 수렴하기 위해 반복적인 알고리즘을 사용하는 방법, 정규화된 이미지 복원 방법 (Constrained Least Squares), 주파수 도메인에서 형성되는 DCT를 분석하여 변형하는 방법이 연구되어 지고 있다[1][3][4][5]. 하지만 이런 방법들은 압축시에 너무 많은 복잡도를 요구하여 시간이 많이 소비되는 단점을 가지고 있다.

그리고 블록 아티팩 화질 저하와 더불어 복원된 이미지는 원본 이미지보다 휘도의 변화가 적다. 이런 결과는 같은 휘도의 데이타가 분포됨을 의미하고 명암이 하나의 값으로 집중되는 경향을 보인다. 명암이 넓게 분포되지 않으면 같은 이미지라도 선명도가 떨어진다[2].
하드웨어를 사용하지 않고 소프트웨어적으로 동영상 자료를 압축할 경우 실시간 전송에 만족할 만한 압축 속도를 제공하지 못하고 과도한 압축률에 의해 화질 저하 문제가 발생한다. 따라서 이 논문에서는 압축 속도를 향상시키기 위해 4단계 고속 알고리즘 보다 적은 비교 횟수를 수행하는 새로운 4단계 탐색 알고리즘과 복원 이미지의 화질 저하를 보상하기 위해 블록 아티팩을 제거하는 알고리즘을 제안하고 명암 대비 스트레칭을 적용하여 화질 저하 문제를 해결하려고 접근하였다. 여기서 화질 보상은 원본 이미지와 동일하게 만드는 것이 아니라 인간의 눈에 더 보기 좋게 인식하도록 하는 것 의미한다.

제2장에서는 새로운 4단계 고속 알고리즘을 제안하고 3장에서는 블록 아티팩을 제거하고 복원 이미지의 명암 대비를 높이는 알고리즘을 제안한다. 그리고 4장에서는 성능 평가를 보여준다.

2. 새로운 4단계 탐색 알고리즘

움직임 추정은 현재 프레임의 매크로 블록이 이전 프레임의 어디에 존재하는지 찾은 과정을 말한다. 많은 고속 블록 정합 알고리즘이 복잡도를 줄이고 탐색 시간을 줄이기 위해 제안되었다[1]. 이 중에서 4 단계 탐색 알고리즘은 좋은 화질과 적은 복잡도를 제공한다. 이 알고리즘은 3 단계 탐색 알고리즘과 비교해서 적은 초기 증감 크기를 사용하고 실세계의 비디오 시퀀스가 중앙 편중적인 특성 (center biased characteristics)을 가지고 있다는 것을 이용하였다. 4단계 탐색 알고리즘은 최악의 경우 27번의 비교를 수행한다. 여기서 4단계 탐색 알고리즘을 개선하여 더욱 적은 탐색을 사용할 수 있음을 발견하였다. 4단계 탐색 알고리즘의 경우 2,3 단계에서 5번의 탐색을 시도한다. 하지만 이 경우 움직임 벡터의 방향성을 이용한다면 대각선에 존재하는 2개의 점은 탐색할 필요가 없어진다.



(a) 단계 탐색 지점 (b) 방향을 고려한 경우의 탐색 지점

그림 1 4단계 탐색 알고리즘의 탐색 지점

그림 1에서 움직임 방향이 대각선인 경우 1번과 2번 지점이 유사한 블록으로 결정될 확률은 극히 적다. 따라서 이 경우 1번과 2번 지점을 탐색할 필요가 없어진다. 4단계 탐색 알고리즘을 개선한 새로운 4단계 탐색 알고리즘은 다음과 같다.

[알고리즘 2.1] 새로운 4단계 탐색 알고리즘

Step 1

15 * 15의 탐색 지역에서 중앙에 위치한 5 * 5구역의 9개의 점을 검사하여 최소의 BDM(Block Distortion Measure)을 찾는다. BDM이 중앙에 존재한다면 step 4로 수행한다. 그렇지 않은 경우 step 2를 수행한다.

Step 2

탐색 윈도우 크기는 5 * 5로 유지한다. 하지만 탐색 패턴은 이전 최소 BDM의 위치에 의존한다.

I. 이전 최소 BDM이 탐색 윈도의 구석(대각선 방향)에 존재한다면 3개의 점을 검사한다.

II. 이전 최소 BDM이 수평, 수직 방향에 존재한다면 그 방향의 3개의 점을 검사한다.

이 단계에서 최소의 BDM이 중앙에서 발견되면 step 4를 수행한다. 그렇지 않다면 step 3를 수행한다.

Step 3

Step2와 동일한 방법을 적용하고 최소의 BDM이 발견되지 않으면 step 4를 수행한다.

Step 4

탐색 윈도우의 크기를 3으로 줄이고 9개의 탐색 점 중에 최소의 BDM값이 존재하는 위치를 움직임 벡터로 결정한다.

4단계 탐색 알고리즘의 경우 2,3 단계에서는 3 또는 5개의 탐색 점을 필요로 한다. 탐색 단계의 최소 대조점이 가운데이면, 단계의 크기는 반으로 줄고 4단계로 뛰어 넘는다.

최악의 경우 탐색 점의 수는 $(18\lceil \log_2 \frac{d+1}{4} \rceil + 9)$을 필요로 해서 최대 27회의 탐색 점을 필요로 한다. 하지만 새로운 4 단계 탐색 알고리즘을 적용할 경우 최악의 경우 $(14\lceil \log_2 \frac{d+1}{4} \rceil + 9)$의 탐색 점을 필요로

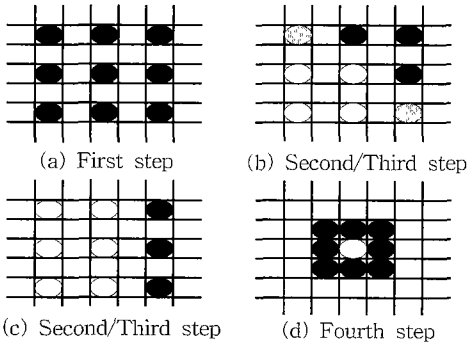


그림 2 새로운 4단계 탐색 알고리즘의 탐색 패턴

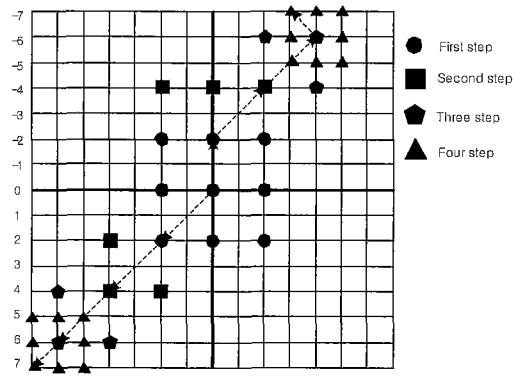


그림 3 새로운 4단계 탐색 패스

최대 23회의 점을 필요로 해서 매 블록마다 최대 4개의 탐색 점이 줄어들게 된다(d는 참조 윈도우의 거리, ±7).

4단계 탐색 알고리즘과 우리가 제안한 새로운 4단계 탐색 알고리즘을 비교하기 위해 QCIF 크기의 프레임 200개를 테스트했다. 이미지 데이타는 움직임이 적고 많음에 따라 3종류로 나누어서 비교하였다.

우선 속도 비교를 위해 4단계 탐색 알고리즘과 새로운 4단계 탐색 알고리즘의 탐색 비교 횟수를 알아보았다. 표 1은 4단계 탐색 알고리즘과 새로운 4단계 탐색 알고리즘의 비교 횟수를 보여준다(표 1 안의 값은 한 프레임 당 비교 횟수를 나타낸다).

표 2 탐색 비교 횟수

	많음	보통	적음
FS	77340	77340	77340
4SS	2052.27	1417.15	1353.78
N4SS	1864.28	1409.71	1353.26

*. FS (Full Search)

*. 4SS (4 Step Search)

*. N4SS (New 4 Step Search)

표 1에서 나타난 것처럼 N4SS이 가장 적은 비교를 하였고 특히 움직임 많은 경우에 다른 알고리즘 보다 더 비교 횟수가 적어 움직임 추정 시간이 적게 걸림을 알 수 있다.

BDM 오차에 의한 성능 비교는 표 2에서 보여준다. 매크로 블록의 오차는 SAD(Sum of Absolute Difference)를 사용해서 계산하였다. 표 2는 움직임 추정 시 4단계 탐색 알고리즘과 새로운 4단계 탐색 알고리즘의 매크로

블록 오차 합의 평균을 나타낸다(표 2 안의 값은 한 프레임 당 발생하는 오차를 나타낸다).

표 3 매크로 블록 오차

	많음	보통	적음
FS	359904	184490	135499
4SS	669361	185905	135407
N4SS	671552	185874	135407

표 2에서 보듯이 N4SS 알고리즘을 적용했을 경우 오차가 4SS 보다 약간 많음을 알 수 있다. 움직임이 적은 경우는 N4SS과 4SS의 차이가 없으며 움직임이 많은 경우 오차가 약간 많았다.

알고리즘에 따른 압축 크기 비교는 표 3에 제시하였다(표 안의 값은 압축 크기를 나타낸다. 단위는 Kb이다).

표 4 압축 크기 비교

	많음	보통	적음
FS	1378	334	45
4SS	1614	388	41
N4SS	1615	390	41

새로운 4단계 탐색 알고리즘이 4단계 탐색 알고리즘보다 압축 크기가 거의 같거나 약간 크다는 것을 알 수 있다.

움직임 방향에 대한 비교 횟수 분석은 표 4에 나타난다. 새로운 4단계 탐색 알고리즘과 4단계 탐색 알고리즘의 차이는 대각선으로 이동할 경우 새로운 4단계 탐색 알고리즘이 더 적은 비교 횟수를 수행한다는 것이다. 따라서 물체의 이동 방향이 수직, 수평, 대각일 경우 알고리즘에 따른 비교 횟수를 알아보았다.

표 5 물체 이동 방향에 대한 탐색 비교 횟수

	수직	수평	대각
FS	15468000	15468000	15468000
4SS	298765	298009	301610
N4SS	291509	290494	292572

표 4의 내용을 보면 물체의 이동 방향에 대한 탐색 비교 횟수는 이동 방향에 상관하지 않고 새로운 4단계 알고리즘이 전반적으로 비교 횟수가 적음을 알고 있다. 하지만 대각의 경우에는 수직과 수평으로 이동한 경우보다 비교 횟수의 차이가 많음을 알 수 있다. 즉 수직과 수평으로 이동하는 물체인 경우 4단계 탐색 알고리즘과 새로운 4단계 탐색 알고리즘의 비교 횟수의 차이가 각각 7256, 7515인 반면에 대각의 경우에는 9038로 더 차이가 남을 알 수 있다. 따라서 새로운 4단계 탐색 알고리즘은 대각으로 이동할 때 더 효율적임을 알 수 있다. 표 2, 3, 4를 볼 때 제안한 N4SS이 4SS보다 매크로 블록의 오차나 압축 크기가 많음을 알 수 있다. 하지만 그 차이는 근소하며 움직임이 없을 경우에는 그 차이가 거의 없음을 알 수 있다. 그런 반면에 비교 횟수를 보았을 경우 제안한 N4SS이 4SS보다 작음을 알 수 있고 움직임이 많은 경우 비교 횟수가 더 적음을 알 수 있다. 따라서 움직임 추정 시간이 더 적게 소비됨을 알 수 있다.

3. 복원 비디오의 화질 보상

3.1 블록 아티팩 제거 알고리즘

블록 아티팩은 블록 경계상의 픽셀들의 큰 차이로 인한 화질 저하를 말한다. 직관적으로 블록 아티팩을 제거하기 위해서는 블록 경계선상의 픽셀들의 차이가 커지지 않도록 하면 된다. 블록 아티팩을 줄이기 위해서 블록 경계간 수평보상과 수직 보상을 한다. 블록의 크기는 8 * 8로 정한다. 보상 방법은 경계 값을 중심으로 최대, 최소 값을 결정하고 픽셀 위치에 따른 가중치를 적용하여 보상 값을 구한다. 먼저 두 개의 경계 값을 비슷한 값으로 보상하고 주위의 값들을 원래의 값과 허용범위를 넘지 않은 수준에서 점진적으로 보상해 나간다. 수직, 수평 보상 시에 적용되는 픽셀들의 위치를 나타낸다.

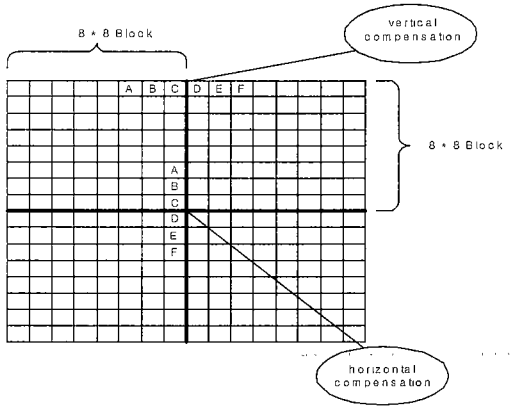


그림 4 블록 경계의 보상 위치

블록 경계에 위치한 픽셀들에 대한 보상 값은 두 픽셀 값의 차이에 기반을 둔다. 보상 간격은 다음과 같은 수식으로 결정된다.

$$factor = (\max - \min)/7$$

여기서 max는 C와 D에 위치한 값 중에서 큰 값을 말하며 min은 작은 값을 말한다. 실제 C와 D에 위치한 값들을 보상하기 위해서 보상 간격 factor에 픽셀 위치에 해당하는 가중치를 곱함으로써 계산된다. 그런 다음 최대, 최소 값의 위치에 따라 보상 값의 가감을 결정하여 최종적으로 보상한다. 블록 경계에 위치한 픽셀일수록 그 차이가 커지기 때문에 가중치가 높아진다. 픽셀 위치에 따른 가중치 비율은 다음과 같다.

$$C : B : A = 4 : 2 : 1$$

$$D : E : F = 4 : 2 : 1$$

보상 간격과 위치에 따른 가중치를 적용하여 C와 D에 위치한 값의 보상은 다음과 같다.

$$Compensate(C, \max_{pos}, \min_{pos}) = Clip(C + Sign(\max_{pos}, \min_{pos}) \times 4 \times factor)$$

$$Compensate(D, \max_{pos}, \min_{pos}) = Clip(D + Sign(\max_{pos}, \min_{pos}) \times 4 \times factor)$$

Sign 함수는 max, min의 위치에 따라 보상치가 더해질지, 뺄지를 결정한다. Clip 함수는 픽셀 값의 범위가 0~255까지로 제한되어 있는데 보상 시에 허용 범위를 벗어날 수 있으므로 범위를 벗어나는 값을 0~255으로 제한한다.

위의 수식을 이용하면 C와 D에 위치한 픽셀을 유사한 값으로 보상할 수 있다. 하지만 경계선 주위의 픽셀 값(A,B,E,F)을 위의 수식대로 적용한다면 픽셀 사이의 차이가 줄어들지 않는 경우도 발생한다.

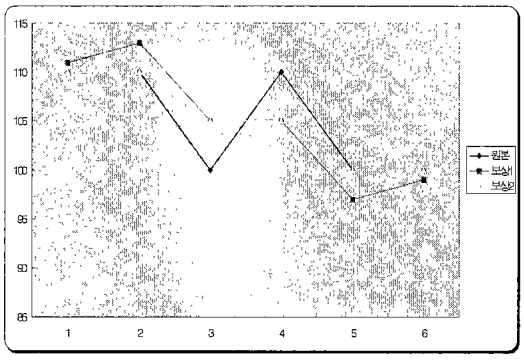


그림 5 보상 값의 결정(픽셀간이 차이가 적게 나는 compensation2를 결정)

그림은 A,B,E,F에 위치한 픽셀의 보상을 나타낸다. 위의 수식에 의한 보상은 compensation1에 나타난다. 예를 들어 픽셀 위치가 5인 경우를 보자. 보상 후의 4번째 5번째의 픽셀 차이는 보상 하기 전보다 줄어든 것을 확인할 수 있다. 하지만 이 경우 5번째 픽셀 값을 보상하지 않고 보상 전의 픽셀을 그대로 유지한다면 4번째와 5번째의 차이가 더 줄어드는 것을 확인할 수 있다. 즉 4번째 픽셀은 위의 수식대로 보상을 하고 5번째 픽셀은 보상하지 않고 원래의 값을 그대로 유지한다면 픽셀 간의 차이가 줄어든다. 따라서 A,B,E,F에 위치한 픽셀을 보상하기 위해서는 보상 전후의 값을 비교하여 차이가 적은 것을 결정해야 한다. 이 방법을 적용한 알고리즘은 다음과 같다.

[알고리즘 3.1] 위치에 따른 픽셀 보상 알고리즘

Step 1

블록 경계 주위의 픽셀을 보상한다.

Step 2

근접한 픽셀과 보상한 값의 차이를 계산한다.

Step 3

근접한 픽셀과 원래의 값의 차이를 계산한다.

Step 4

만약 step 2에서 구한 차이가 step 3에서 계산된 값보다 작다면 픽셀 값을 보상한 값으로 바꾼다. 그렇지 않다면 픽셀의 값을 원래의 값으로 유지한다.

만약 블록 경계에 실제 경계선이 존재할 경우 위 알고리즘을 사용하여 보상을 하면 경계선이 보상되어 원 영상과 차이를 보일 수 있다. 하지만 본 논문은 인간이 자연스럽게 영상을 인식하는데 초점을 두고 있고 블록 경계에 실제 경계선이 존재할 확률은 희박하다고 할 수 있어 실제 경계선들의 차이에 대한 화질 열화가 나타날 경우는 드물다고 가정하였다.

3.2 스트레칭을 이용한 명암 대비

하나의 이미지를 비교할 경우 명암 대비가 높은 이미지일수록 어두운 부분과 밝은 부분의 차이가 선명하게 보인다. 압축된 동영상 이미지를 복원시킨 경우 명암 대비가 적게 나타나는 현상을 보인다. 따라서 복원 이미지에 대해 명암 대비를 높게 한다면 이미지가 선명하게 보인다. 명암 대비는 영상 히스토그램(histogram)을 사용해서 바꿀 수 있다.

영상 히스토그램은 영상의 모든 픽셀들에 대한 밝기 값을 출현빈도로 나타낸 것이다[6]. 일반적으로 어두운 영상은 픽셀 값 분포가 왼쪽으로 편중된 히스토그램을 가지며 밝은 영상은 픽셀 값 분포가 오른쪽으로 편중된 히스토그램을 갖는다. 또한 명암의 대비가 낮으면 픽셀

이 히스토그램의 왼쪽, 오른쪽 또는 중앙의 오른쪽으로 집중된다. 그리고 히스토그램의 막대가 촘촘하게 밀집되며 픽셀 값의 범위가 일부분에만 분포한다. 반대로 명상의 대비가 높으면 넓은 범위의 픽셀 값을 포함한다.
영상의 명암 대비를 높이는 방법은 히스토그램을 고르게 분포시키면 된다. 따라서 히스토그램이 0부터 255까지 고르게 분포하도록 스트레칭을 시키면 된다.

명암 스트레칭을 하기 위해서 일정 양의 픽셀을 흰색, 검은색을 갖도록 지정한다. 그리고 낮은 범위와 높은 범위의 임계값을 발견하기 위해서 히스토그램을 조사한다. 그 다음 임계 값 범위 내의 픽셀 빈도에 해당하는 룩업 테이블(Lookup Table)을 구성한다. 룩업 테이블의 데이타의 값은 다음의 수식으로 결정한다.

$$value(x) = 0 \qquad \text{for } x \le low$$

$$value(x) = 255 \times \frac{x - low}{high - low} \qquad \text{for } low \le x \le high$$

$$value(x) = 255 \qquad \text{for } x \le high$$

위의 수식은 일정한 양만큼 낮은 값 (0)과 높은 값(255)을 유지하면서 임계 값 안의 값들은 전체 명암 값을 포함하도록 확장된다. 따라서 명암이 넓게 분포되기 때문에 명암 대비가 뚜렷해지므로 선명한 화질을 보여준다.

4. 성능 평가

본 논문의 결과는 펜티엄 350MHz, 256M RAM를 가진 PC에서 수행되었다. 압축 속도를 측정하기 위해 QCIF 크기의 이미지 200개의 프레임이 사용되었고 움직임이 적고 많음에 따라 3 종류의 데이타를 사용하였다.

4.1 새로운 4 단계 탐색 알고리즘을 적용한 속도 성능 향상

그림 6는 N4SS과 복잡도를 많이 요하는 함수에 MMX을 적용한 후의 성능을 그림으로 보여주고 있다. Origianl 함수는 H.263에서 제공하는 기존의 방식을 그대로 적용했을 때 생성되는 초당 프레임 수를 나타내고 N4SS는 기존의 방식 중 Full Search 알고리즘을 N4SS으로 구현하였을 때 나타나는 초당 프레임 수를 나타낸다. MMX_SAD는 현재 프레임이 이전 프레임의 어디에 있는지 찾아내기 위하여 각 벡터의 값을 계산하는 Cost-Function의 일종인 SAD 함수를 MMX로 구현하였을 때 생성되는 프레임 수를 나타낸다.

그리고 MMX_DCT는 공간 영역을 주파수 영역으로 변환하는 DCT함수를 MMX로 구현했을 때 나타나는 프레임 수를 나타낸다. High 는 움직임이 많은 데이타를 나타내며 Mid 는 중간 정도인 데이타 그리고 Low는 움직임이 적은 데이타를 나타낸다.

위 그림에서 알 수 있듯이 H.263에서 제공하는 함수를 사용한 경우에는 초당 2.5 프레임의 압축속도를 나타냈지만 N4SS를 적용시켰을 때는 초당 6 프레임이 향상되었고 MMX_SAD와 N4SS를 적용시켰을 때는 초당 12 프레임이 향상되었다. 그리고 MMX_SAD, N4SS와MMX_DCT를 모두 적용시키면 초당 17 프레임의 속도로 압축됨을 알 수 있다.

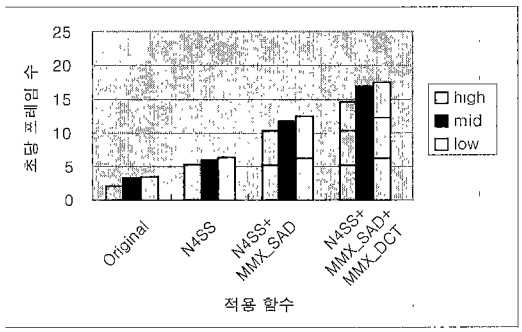


그림 6 제안한 각 함수를 적용하였을 때 향상되는 프레임 수

* Original : Full Search 블록 정합 알고리즘을 사용한 경우
* N4SS : 새로운 4단계 탐색 알고리즘
* MMX_SAD : SAD 함수를 MMX로 구현한 함수
* MMX_DCT : DCT 함수를 MMX로 구현한 함수

4.2 화질 평가

복원된 이미지의 화질을 평가하기 위해서 주관적 또는 분석적 방법을 사용한다. 하지만 주관적 평가는 평가하는데 시간이 많이 걸리며 주관적인 요소가 너무 많이 들어가 객관적으로 증명할 수가 없다. 따라서 객관적인 자료를 바탕으로 이미지의 화질을 평가하는 방법이 필요한데 가장 많이 사용되는 방법이 PSNR(Peak Signal to Noise Ratio)이다. PSNR은 원본 이미지와 상대적으로 얼마나 손상되었는지를 측정하는 방식이다. PSNR은 다음과 같이 정의된다[7].

$$PSNR(dB) = 10 \log_{10} \frac{(2^n - 1)^2}{MSE}$$

여기서 n은 이미지의 픽셀 수를 나타내며 MSE는 복원된 이미지와 원본 이미지와의 오차의 제곱의 평균을 나타낸다.

표 6은 QCIF 크기의 프레임을 180개 수행하였을 때 속도 향상 전후의 fps와 PSNR의 관계를 보여주고 있다.

표 6 속도향상 전과 속도 향상 후의 PSNR

	속도향상 전 PSNR	속도향상 후 PSNR
1	40.90	40.90
18	25.28	25.36
33	30.24	30.37
48	28.65	28.76
63	27.68	27.70
78	29.19	29.23
31	31.22	31.02
108	26.08	26.12
123	26.10	26.11
138	25.68	25.70
153	26.96	25.70
168	25.26	25.21
180	26.41	26.42

위 표에서 보면 속도 향상 전이나 속도 향상 후의 PSNR 값은 그다지 차이가 나지 않는 것을 볼 수 있다. 따라서 속도향상을 위하여 제안한 알고리즘이나 다시 구현한 함수를 사용하더라도 화질에는 그다지 큰 영향을 주지 않는 것을 알 수 있다.

그리고 PSNR로 화질 개선 필터를 사용해 보상한 이미지의 화질은 다음과 같다.

그림 7은 블록 아티팩을 가지고 있는 복원된 이미지와 필터를 사용하여 화질 보상을 한 이미지의 PSNR를 나타낸다. 데이타는 연속된 180개의 이미지를 사용하였고 원본 이미지는 카메라에서 받은 이미지를 사용하였다. 블록 아티팩을 가진 복원된 이미지의 평균 PSNR은 34.96dB이며 필터를 사용하여 화질 보상을 한 후의 이

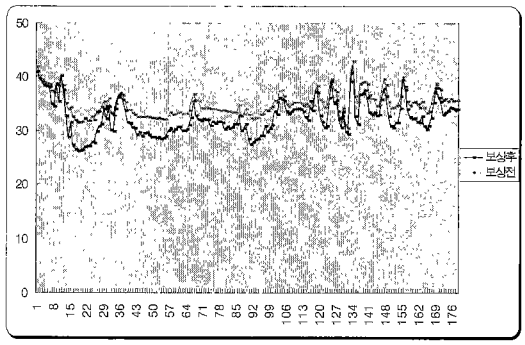


그림 7 보상 전후의 PSNR

미지의 평균 PSNR은 32.46dB이다. PSNR의 결과를 보면 화질 보상을 한 후의 이미지의 화질이 보상 전보다 수치상으로 낮게 나와 화질이 나쁜 것으로 분석되었다. 화질 보상 후 PSNR이 낮게 나온 결과는 두 가지 이유로 설명할 수 있다.

첫번째로 PSNR에서 사용한 원본 이미지는 카메라에서 캡쳐한 이미지이다. 하지만 보상 시에 원본 이미지를 고려하지 않고 오직 복원 이미지만 사용하였다. 화질 보상이 원본 이미지와 동일하게 만드는 것이 아니었다. 우리는 인간이 더 좋은 화질로 인식하는데 초점을 맞추었다.

따라서 화질 보상이 원본 이미지를 고려하지 않았기 때문에 원본 이미지와의 손상 여부를 판단하는 PSNR은 우리가 제안한 방법에 대한 평가 기준으로 적합하지 않았다.

그림에서 보듯 화질 보상한 후에 원본 이미지와 비슷한 경우 차이가 나지 않으며 원본 이미지와 다른 경우는 PSNR의 결과가 낮아졌다. 두 번째 이유는 대부분의

그림 8 원본 이미지와 확대한 이미지

그림 9 보상 전 이미지와 확대한 이미지

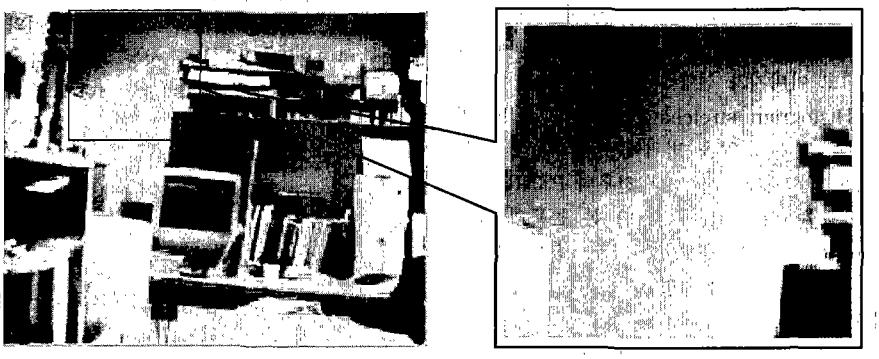


그림 10 보상 후 이미지와 확대한 이미지

영상 처리 기법은 이미지의 픽셀 값을 변환시키는 경우가 많다. 화질 보상 시 명암 대비를 높게 하기 위해서 명암 스트레칭을 사용하는데 이미지의 픽셀 값을 변환하는 히스토그램을 적용하였다. 이것은 원본 이미지와의 차이가 나기 때문에 원본과 비교한 PSNR에서 화질이 낮게 분석되는 것은 당연하다.

본 논문의 화질 보상 목적은 원 영상과 동일하게 만드는 것이 아니라 인간이 자연스럽게 영상을 인식하도록 하는 것이다.

따라서 관찰자들의 주관적인 관점을 통해 화질 평가를 수행하였고 실제 보상전의 이미지와 보상후의 이미지를 비교한 결과 88%의 관찰자가 보상 후의 이미지에서 블록화 현상이 제거되었으며 이로 인해 화질이 더 좋게 보인다고 평가하였다.

그림 10은 보상 후의 이미지이다. 그림 9에서 보이는 블록 아티팩 현상이 보이지 않는 것을 확인할 수 있다. 그림 8, 9, 10의 우측이미지는 좌측 이미지의 우측상단에 네모부분을 확대한 이미지이다.

그림 11은 블록 경계의 두 픽셀의 차이를 나타낸다. 보상 후의 블록 경계의 두 픽셀 값의 차이가 3이상이 되지 않는 것을 확인 할 수 있다. 물론 블록 경계의 값의 주위의 값들도 급격히 변화되지 않고 완만하게 변화가 된다.

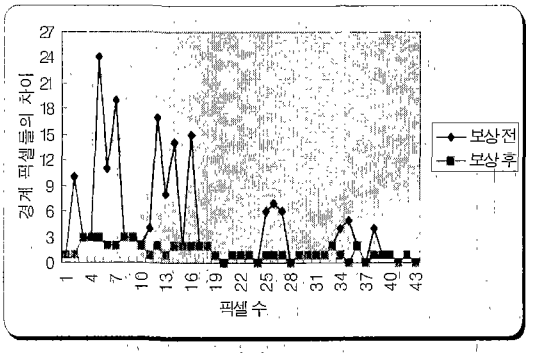


그림 11 보상 후의 두 픽셀 차이

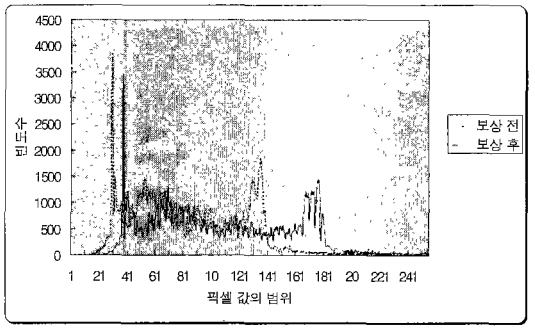


그림 12 보상 전후의 히스토그램

그림 12는 보상 전후의 히스토그램을 나타내고 있다. 보상 후의 히스토그램이 오른쪽으로 치우치고 넓게 분포되어 있는 것을 확인할 수 있다. 따라서 명암 대비가 넓게 분포가 되고 이미지가 선명하다.

본 논문에 Histogram stretching 적용시 원 영상의 상황 정보를 왜곡시키지 않을 범위에서 임계값을 적용하였다. 본 논문의 그림 7에서 보상 전후의 PSNR을 비교해 보면 보상 후의 PSNR이 낮지만 본 논문의 목적인 인간이 자연스럽게 영상을 인식한다는 점을 고려할 때 PSNR 차이는 객관성을 유지할 수 있다.

4.3 복잡도

화질 보상의 성능 평가를 위해 실행 시간과 실행 시간을 계산하였다. 표 5는 보상 전과 보상 후의 실행 시간을 보여준다.

표 7 보상 전후의 실행 시간

	보상 전		보상 후	
	실행 시간	초당 프레임 수	실행 시간	초당 프레임 수
많음	2865	69	4270	46
보통	1979	101	3229	61
적음	972	205	2326	85

표 7에서 나타난 것처럼 화질 보상 후에도 초당 30 프레임 이상을 실행시킬 수 있으므로 실시간 환경에서 사용할 수 있다.

화질 보상은 명암 스트레칭, 수직 보상, 수평 보상으로 복원된 이미지를 보상한다. 명암 스트레칭의 경우 시간 복잡도를 계산하면 다음과 같다.

$$O(c \cdot r) + O(low) + O(high) + O((c \cdot r) - (low + high)) + O(c \cdot r) = O(c \cdot r)$$

C는 이미지의 높이, r은 이미지의 너비, low는 흰색 비율 값을 나타내는 상수, high는 검은색 비율 값을 나타내는 상수이다.

수직 보상인 경우의 시간 복잡도는 다음과 같다.

$$O(c/16 \cdot r) + O(6) = O(c \cdot r)$$

수평 보상인 경우는 다음과 같다.

$$O(r/16 \cdot c) + O(6) = O(r \cdot c)$$

따라서 전체 시간 복잡도는 $O(c \cdot r) + O(c \cdot r) + O(r \cdot c) = O(c \cdot r)$이다.

5. 결 론

이 논문에서는 새로운 4단계 탐색 알고리즘(N4SS)을 사용하여 압축 속도를 향상시키고 블록 아티팩을 제거하고 명암 대비를 확장하여 복원 이미지의 선명도를 높이는 방법을 제안하였다. 성능 평가에서 보듯이 새로운 4단계 탐색 알고리즘은 기존의 4단계 탐색 알고리즘 보다 움직임이 많은 경우와 물체가 대각선으로 자주 이동할 때 비교 횟수가 더 적어짐을 알 수 있다. 복원 영상의 화질 보상에서는 블록 경계에 위치한 픽셀들의 차이를 완만하게 줄이고 히스토그램을 사용하여 명암 대비를 확장함으로써 더 보기 좋은 복원 영상을 보여 주었다. 또한 초당 30 프레임 이상으로 복원하는데 문제가 없기 때문에 소프트웨어 복원으로 실시간 환경에서 이용이 가능하다.

참 고 문 헌

[1] Borko Furht, Joshua Greenberg and Raymond Westwater, Kluwer Academic Publishers, Motion Estimation Algorithms for Video Compression, 1997.

[2] Mei-Yin Shen, JongWon Kim, C.C. Jay Kuo, "FAST COMPRESSION ARTIFACT REDUCTION TECHNIQUE BASED ON NONLINEAR FILTERING," IEEE International Symposium on Circuits and Systems, vol.4. pp.179-182, May 1999Berrou, C., M. Alard, and B. Le Floch, Coded Orthogonal Frequency-Division Multiplexing, Proc. ACM User Services, 81(6), 892-996 (June, 1995).

[3] Bong Gyun Roh, Nam Ik Cho and Sang Uk Lee, "BLOCKING ARTIFACTS REDUCTION USING CONSTRAINED LEAST SQUARES METHOD WITH DIRECTIONAL INFORMA- TION," Proceedings of the Picture Coding Symposium, pp.199-202, April 1999.

[4] Seyfullah H. Oguz, Truong Q. Nguyen, Yu Hen Hu, "CRITICAL QUANTIZATION DECI- SIONS

IN TRANSFORM CODING AND BLOCKING ARTIFACTS," IEEE International Symposium on Circuits and Systems, Vol.4. pp.143-146, May 1999.

[5] Wing-Kuen Ling, Bing Zeng, "A NOVEL METHOD FOR BLOCKING EFFECT REDUCTION IN DCT-CODED IMAGES," IEEE International Symposium on Circuits and Systems, Vol.4, pp.46-49, May 1999.

[6] Randy Crane, A Simplified Approach to Image Processing, Prentice-Hall, 1997.

[7] Martyn J.Riley and lain E.G.Richardson, *Digi- tal Video Communiations*, Artech House, 1997.

[8] S K. Lee, Y.-S. Ho, T. K. Kim, J. K. Paik , "FAST IMAGE RESTORATION FOR REDUCING BLOCKING ARTIFACTS," Proceedings of the Visual Communications and Image Processing '99-Part 1, pp.677-686, January. 1997.

[9] Guy Cote, Berna Erol, Michael Gallant, and Faouzi Kossentini, "H.263+: VIDEO CODING AT LOW BIT RATES," IEEE Transactions on Circuits and Systems for Video Techno- logy, Vol. 8, No. 7, pp.848-866, November 1998.

[10] Alessandro Bellini, Alberto Del Lungo, "A Fast H.261 SOFTWARE CODEC FOR HIGH QUALITY VIDEOCONFERENCING ON PCS," Proceedings of the IEEE Multimedia Systems '99-Volume 2, Vol.2. pp.1007-1008, June 1999.

[11] ITU-T Standardization Sector of ITU, Video Coding Test Model Near-Term, Version 8 (TMN8), Release 0, H.263 Ad Hoc Group, June 1997.

[12] K.R.Rao, J.J. Hwang, Techniques & Standards for Image. Video & Audio Coding, Prentice Hall, 1996.

[13] Cote, G., Gallant, M. and Kossentini, F. "Efficient Motion Vector Estimation and Coding for H.263-Based Very Low Bit Rate Video Compression." Online document available at URL http://www.ece.ubc.ca/spmg/, 1997.

[14] CCITT H.263-Image Compression-Online document available at URL http://www.stud.ee.ethz.ch/~rmprince/h263.html
Performance Analysis of Intel MMX Technology for an H.263 Video Encoder-Ville Lappala inen

[15] James Abel, Kumar Balasubramanian, Mike Bargeron, Tom Craver, Mike Phlipot, Applications Tuning for Streaming SIMD Extensions, Intel Technology Journal, Q2, 1999.

윤 성 규
1998년 숭실대학교 컴퓨터 학부 학사. 2000년 숭실대학교 컴퓨터 학과 석사. 2000년 ~ 현재 (주)글로벌데이타시스템 연구원.

강 의 선
1999년 탐라대학교 산업정보학과 졸업. 1999년 9월 ~ 현재 숭실대학교 컴퓨터학과 재학 중.

유 환 종
1998년 2월 숭실대학교 소프트웨어공학과 졸업(학사). 1998년 3월 ~ 2000년 3월 숭실대학교 컴퓨터학과 졸업(석사 : 전공-멀티미디어). 1999년 12월 ~ 현재 이지시스템 부설 연구소

임 영 환
1977년 경북대학교 수학과 졸업(이학사). 1979년 한국과학기술원 전산학과 졸업(이학석사). 1985년 Northwestern Universty 졸업(이학박사). 1979년 1월 ~ 1996년 2월 한국전자통신연구소 책임연구원. 1996년 3월 ~ 현재 숭실대학교 정보과학대학 부교수. 관심분야는 멀티미디어, 초고속정보통신 시스템 소프트웨어, 에이젼트 등.