무손실 압축을 위한 H.264/AVC 정수 변환의 변형

유 훈*

Modification of the integer transform in H.264/AVC for lossless compression

Hoon Yoo*

이 논문은 2009년도 상명대학교 연구비를 지원받았음

요 약

본 논문에서는 H.264/AVC 표준에서 사용되는 정수 변환을 변형하여 무손실 압축에 효율적인 변환 구현 방법을 제안한다. 기존의 가역(reversible) 변환용으로 제시된 정수 변환은 변환 계수의 값의 범위가 상당히 커서 무손실 압축에 효율적이지 못한 면이 있다. 이런 문제점을 해결하기 위해서 효율적인 정수 변환을 제시한다. 기존 정수 변환의 변형은 고속 연산 수행을 위해서 리프팅을 기반으로 설계되고 효율적인 구조를 도출한다. 본 논문의 결과로서 고속 연산 수행을 위한 신호 흐름도를 제시하고, 이에 관련된 실험 결과를 제공한다. 실험 결과는 제안된 변형 정수 변환이 기존 구현 방법에 비하여 무손실 압축 성능에서 우수하단 것을 보여준다.

ABSTRACT

This papers describes modification of the integer transform used in H.264/AVC in order to efficiently apply to lossless compression. The previous reversible integer transform is not efficient for lossless compression due to large dynamic range of the transform coefficients. To reduce the problem, efficient and reversible integer transforms are proposed. The modified transforms are designed based on the lifting scheme for fast transforms. This paper introduces signal flow graphs for the proposed fast transforms and provides corresponding experimental results. The results indicate that the proposed modified reversible integer transform are superior to the previous transform in terms of lossless compression efficiency.

키워드

가역 정수 변환, H.264/AVC, 무손실 압축, 리프팅 구조

Key word

Reversible integer transform, H.264/AVC, lossless compression, lifting scheme

* 상명대학교 디지털미디어학부 (교신저자, hunie@smu.ac.kr)

접수일자 : 2010. 09. 02
심사완료일자 : 2010. 10. 01

Ⅰ. 서 론

컴퓨터 성능의 향상에 따라서 보다 더 복잡한 알고리즘과 기술로 영상 압축하는 기법들이 소개되었다. 대표적인 예로서 H.264/AVC 압축 방법을 들 수 있다 [1]. 그러나 최근 저 복잡도가 중요한 문제로 대두 되고 있다. 이유는 컴퓨터 환경이 점차로 이동형 단말기 환경으로 변경되고 있어서 컴퓨터 환경이 상당히 제약적인 경우가 많아졌기 때문이다. 이에 따라서 이전 동영상 압축 표준에서 사용된 8점 DCT는 제외되었고 H.264/AVC에서는 4점 정수 변환(integer transform)이 사용되었다 [1, 2]. 중요한 이유로 4x4 블록 단위로의 처리 기준 변경과 저 복잡도 변환의 필요성을 들 수 있다. 4점 단위의 변환에서도 DCT가 영상 압축으로 활용하기에 성능면에서 우수한 것으로 알려져 있지만 실수 연산을 해야 하는 부담감과 역변환에서의 불일치와 같은 기존 DCT 변환이 안고 있던 문제점들을 해결하기 위한 방안으로 정수 변환이 검토되었다.

정수 변환은 곱셈기를 사용하지 않을 수 있다는 점 등, 실수 연산에 비하여 획기적으로 비용을 절감할 수 있을 뿐만 아니라 변환에 있어서 무손실 압축이 가능한 가역(reversible) 변환을 제공한다는 장점이 있다. 따라서 H.264/AVC에서도 가역(reversible) 정수 변환으로 그림 1에서 제시된 것과 같은 정수 변환이 다소 변경되어 그림 2와 같은 방식이 제안되었다 [3]. 하지만 제시된 가역(reversible) 정수 변환은 무손실 압축 성능을 충분히 고려하지 않은 방법으로 성능면에서 문제를 안고 있다.
본 논문에서는 이점을 해결하고자 H.264에서 사용되고 있는 가역(reversible) 정수 변환을 무손실 영상 압축에 적합한 형태로 변형한다. 고속 연산을 유지하기 위해서 변형된 정수 변환은 리프팅을 기반으로 개선하는 과정을 유도하고 그 결과를 고속 구현을 위한 신호 흐름도로써 제시한다. 또한 제안된 변환과 기존의 변형된 정수 변환에 대해서 무손실 압축 기법에 적합한 방법으로 1차 엔트로피 분석 실험을 통해서 평가한다.

Ⅱ.H.264/AVC에서의 정수 변환

DCT를 대치하기위한 정수 변환 매트릭스로 검토된 것은 아래와 같다.

$$H=\begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix}$$

초기 H.264 디자인에서는 파라미터로써 {a=13, b=17, c=7}로 선택하여 제시되었다. 이 선택은 DCT와 상당히 근사된다는 장점으로 선정되었다. 그런데 이 정수 변환은 변환 계수의 범위(dynamic range)문제가 있었다. 즉, 파라미터 계수들 값이 커서 중간 저장 및 연산시에 32비트 산술연산이 필요한 점이다. 이는 하드웨어 구현에서 있어서 상당한 부담을 줄 수 있는 부분이다. 따라서 H.264 디자인에서 최종적으로 선택된 파라미터는 {a=1, b=2, c=1}이다. 즉 해당하는 정수 변환 매트릭스는 아래와 같다.

$$H=\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

이 매트릭스에 의한 변환으로 중간 계수의 범위는 원 입력 데이터 비트 범위에서 추가되는 범위가 최대 6bit 이내에서 구현이 가능하다. 예를 들면, 8비트 입력 영상에 대해서 중간 계수 값은 14비트면 충분하다. 따라서 16비트 산술 연산으로 구현이 가능하다. 또한 이 변환 매트릭스는 그림 1과 같은 고속 알고리즘이 존재하여 연산량 면에서 매우 효율적인 방법이다.

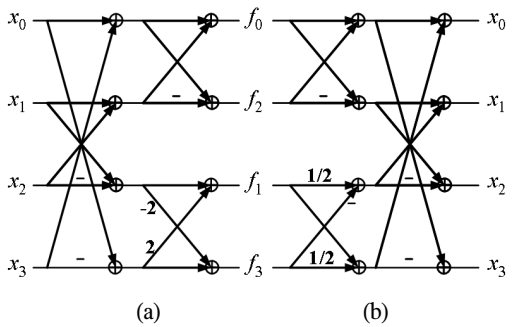

그림 1. H.264 정수 변환의 고속 구현
(a) 변환 (b) 역변환
Fig. 1. Fast implementation of the H.264 integer transform (a) transform (b) inverse transform

그럼에도 불구하고 다소 거친 근사로 보이는 이 정수 변환은 DCT와 비교하여 변환이득(transform gain)의 손실이 0.01dB에 불과해서 그 유용성이 입증되었다 [2].

그림 1에 제시된 정수 변환에서 무손실 압축을 위해서는 가역(reversible) 정수변환으로의 변형이 요구된다. 무손실 압축에서는 변환 계수의 범위가 압축 성능에 매우 민감하게 작용하는 관계로 계수 값의 범위를 최대한 줄일 필요가 있다. 이에 따라서 H.264의 정수 변환에 대한 변형 정수 변환이 그림 2와 같이 제시되었다 [3].

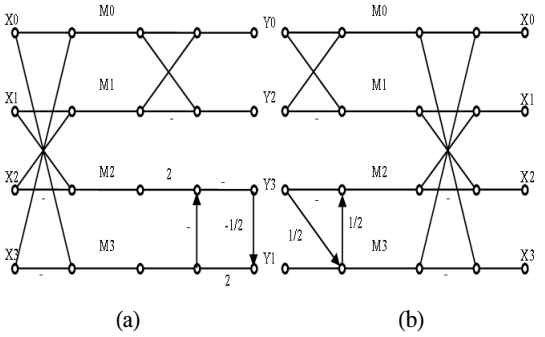

그림 2. 변형 H.264 정수 변환 (a) 변환 (b) 역변환
Fig. 2. Modified H.264 integer transform (a) transform (b) inverse transform

변형된 정수 변환은 무손실을 보장하는 가역(reversible) 변환이 되고 또한 그림 1에서 제시된 변환보다 변환 계수의 범위가 줄어드는 것을 신호 흐름도에서 확인할 수 있다. 그러나 제시된 방법이 무손실을 보장하지만 여전히 변환 계수의 범위가 여전히 큰 것으로 보아 효율적인 측면은 충분히 고려하지 않은 방법인 것을 알 수 있고, 이에 대한 성능 개선이 필요하다는 것을 알 수 있다.

Ⅲ. 리프팅을 이용한 효율적인 정수 변환의 변형

본 논문에서는 그림 1에서 제시된 H.264의 정수 변환을 리프팅을 활용하여 효율적인 가역(reversible) 정수 변환으로 변형한다. 그림 1에서 보이듯이 4점 정수 변환의 고속 알고리즘은 4점 하다마드(Hadamard) 변환의 고속 알고리즘과 상당히 유사하다. 2점 DCT와 2점 하다마드 변환이 동일한 것을 고려한다면 4점 변환에서도 유사성이 보이는 것은 당연하다고 할 수 있다. 이와 같은 하다마드 변환의 고속 알고리즘과 DCT간의 유사성으로 하다마드 변환을 기반으로 DCT를 근사하여 intDCT로 알려져 있다 [4, 6]. intDCT와 다소 다른 접근법으로 DCT의 고속 알고리즘에서 리프팅을 이용하여 정수 변환을 생성하는 방법이 제안되었다 [6]. 이 방법은 binDCT로 알려져 있고 현재까지 나와 있는 DCT에 대한 정수 변환으로의 근사에서는 가장 우수한 것으로 알려져 있다. intDCT와 binDCT는 현재 사용되는 DCT를 곱셈기를 사용하지 않고 정수 연산만으로 구현 가능한 변환으로의 근사하는데 기본적인 초점이 맞추어져 있다. 한편으로 하다마드 변환의 계수의 범위를 줄이는 방법으로 변형 하다마드 변환이 제안되었다 [7, 8, 9]. 이 방법은 무손실 압축에 응용하기 위한 방안으로 가역(reversible) 하다마드 변환으로 변형한 것이다. 앞서 언급한 것처럼 그림 1에서도 알 수 있듯이 H.264의 정수 변환은 구조면에서 하다마드 변환과 상당히 유사하다. 따라서 H.264 정수 변환을 무손실 압축용으로 변형하기에 좋은 방법론이 된다.

그림 1에서 알 수 있듯이 H.264 정수 변환의 고속 알고리즘에서 나비(butterfly) 연산으로 구성되어있다. 특히 4점 변환에서는 그림 3에서 소개한 바와 같이 2종류의 나비 연산으로만 구성된다. 따라서 이 두 종류에 대한 나비 연산을 리프팅을 활용하여 변형하는 것이 하나의 방법이 된다 [5].

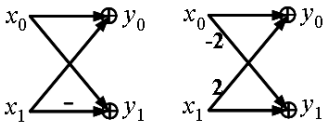

그림 3. 나비 연산들
Fig. 3. Butterfly operations

그림 3의 좌측에 보이는 나비 연산은 2점 하다마드 변환과 동일하다. 또한 이 나비 연산을 무손실용으로 변형하는 것은 알려져 있다 [5, 7, 8]. 좌측 나비 연산에서 $y_0=x_0+x_1$이 되어서 y_0의 계수 값의 범위가 2배로 증가한다. 또한 $y_1=x_0-x_1$도 계수 값의 범위를 증가 시킨다. 여기서 재미있는 것은 y_0, y_1의 LSB(least significant bit)는 항상 같다는 것이다. 즉, y_0가 홀수면 y_1도 항상 홀수이고 반대

로 y_0가 짝수면 y_1도 항상 짝수가 된다 [7, 8]. 이 사실로부터 y_0, y_1의 둘 중 하나는 쉬프트 연산으로 LSB를 없애 버려도 역변환에서 문제없이 복원해 낼 수 있다. 이 변환은 S변환으로 알려져 있고 보통 식 (1), (2)와 같이 리프팅에 의거 변환과 역변환이 정의된다.

$$y_1 = x_0 - x_1 \qquad (1)$$
$$y_0 = x_0 - \lfloor y_1/2 \rfloor$$

$$x_0 = y_0 + \lfloor y_1/2 \rfloor \qquad (2)$$
$$x_1 = x_0 - y_1$$

여기서, 식 (1)과 (2)를 리프팅에 의거 고속 알고리즘을 구현하면 그림 4와 같은 흐름도가 얻어진다. 참고로 변환 계수에 해당하는 y_0, y_1의 계수의 범위를 보면 y_1 계수는 입력 x_0나 x_1보다 2배 더 크다. 즉 1비트 더 많은 범위를 갖게 된다. 하지만 y_0 계수는 입력 변수의 범위와 동일한 범위를 갖는다. 따라서 y_0의 계수의 범위가 기존의 2점 하다마드 변환보다 작아진 사실을 확인할 수 있다.

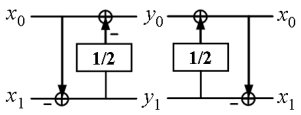

그림 4. S변환에 대한 리프팅 구현.
Fig. 4. Lifting implementation of S-transform.

H.264 정수 변환의 무손실 압축용으로의 변형을 완성하기 위해서는 그림 3의 우측 나비 연산에 대한 변환 계수의 범위를 무손실 압축용으로 최소화하는 것이 기본적인 목표가 되고 이는 리프팅을 활용한 방법이 가장 체계적이고 효율적이다. 먼저 그림 3의 우측 나비 연산 식은 아래와 같다.

$$\begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \qquad (3)$$

리프팅의 과정은 식 (3)의 매트릭스를 인수분해 (factorization)해서 간단한 매트릭스들의 곱의 꼴로 바꾸는 작업이다. 특히 리프팅은 주로 2x2 매트릭스에 대해서 이루어진다. 일반적인 2x2 매트릭스 A는 식 (4)와 같은 인수분해가 가능하다.

$$\begin{aligned} \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} &= \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \\ &= \begin{bmatrix} k_0 & 0 \\ 0 & k_1 \end{bmatrix} \begin{bmatrix} 1 & u \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ p & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \\ &= \begin{bmatrix} D/d & 0 \\ 0 & d \end{bmatrix} \begin{bmatrix} 1 & bd/D \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ c/d & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \end{aligned} \qquad (4)$$

여기서, $D=ad-bc$이다. 식(4)와 같은 인수분해는 유일하지는 않다. 즉, 다양한 인수분해가 가능한데, 식 (4)의 형태가 무손실용 변환으로의 변형이 가장 유용한 형태이다. 왜냐하면 k_0, k_1변수는 무손실용 변환에서는 필요하지 않기 때문이다. 즉 실제 무손실용 정수 변환에서는 p와 u변수에 대한 연산만 하면 되고, 이는 추가적인 연산을 필요하지 않다는 의미이다.

참고로, 식(4)에서 리프팅 변수인 k_0, k_1, p, u를 얻는 방법에는 여러 방법이 존재한다. 식(4)에서 단순히 전개를 해서 원 매트릭스 A와 변수 비교를 통한 방법으로 얻어 낼 수도 있고 좀 더 전략적인 방법으로는 매트릭스 A의 역 매트릭스에 대한 가우스 소거법을 적용하되 LDU 분해를 변형하여 LUD 분해를 해서 구하는 방법도 존재한다. 즉 A^{-1}=LUD 분해를 통해서 $A=D^{-1}U^{-1}L^{-1}$ 분해를 얻는 것이다. 여기서, 매트릭스 L과 U는 각각 lower와 upper 삼각 매트릭스이고 그 역원도 동일한 삼각 매트릭스이다.

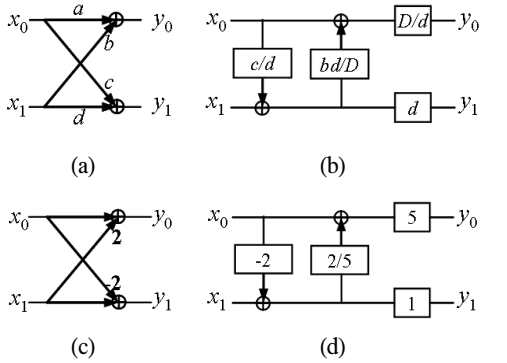

그림 5. 2x2 매트릭스 연산에 대한 리프팅 구현.
(a)나비연산 (b)리프팅 구조 (c)H.264 나비연산
(d) (c)의 리프팅 구조.
Fig. 5. Lifting implementation of 2x2 matrix operation
(a) butterfly (b) lifting structure (c) H.264 butterfly
(d) lifting structure of (c).

식 (4)의 구조를 도식화하면 그림 5와 같이 얻어진다. 그림 5의 (c)와 (d)에서는 H.264에서 사용되는 나비 연산에 대한 구현을 예로써 보여주고 있다. 그림 4와 5에서 얻어진 나비 연산을 그림 1에 적용하면 무손실용 정수 변환으로 이루어진다. 그러나 여전히 문제점이 남아 있음을 알 수 있다. 먼저, 그림 5(d)에서도 알 수 있듯이 y_1의 계수 범위는 여전히 크다. 사실 그림 1의 구조에서 변한 것이 없다. 또한 5로 나누는 연산이 필요하게 됨을 알 수 있다. 따라서 그림 5의 (d)구조는 기존 방법인 그림 2와 비교하여 성능면에서 나아진 것이 없으며 오히려 나눗셈 연산으로 인해서 더더욱 복잡해짐을 알 수 있다.

리프팅 구조에서 변환 계수의 범위를 줄이는 방법으로 p, u계수가 1보다 작게 설정하는 것이다. 가급적이면 p, u계수가 0.5보다 작다면 변환 계수의 범위가 획기적으로 줄어들 수 있다. 따라서, 다음과 같은 조건식 (5)이 효율적인 무손실용 정수 변환의 관계식이 될 수 있다.

$$c/d < 1,\ bd/D < 1 \tag{5}$$

식 (5) 조건을 만족할 수 있게끔 나비 연산을 바꾸기 위해서는 순열(permutation)이 효과적이다. 즉, 입력 벡터의 원소의 순서를 바꾸던가 아니면 출력 벡터 원소의 순서를 바꾸면 나비 연산은 식(5)를 만족하는 식을 변경이 가능하다. 그림 6에서 두 종류의 변형된 나비 연산과 해당하는 리프팅 구조를 보여준다.

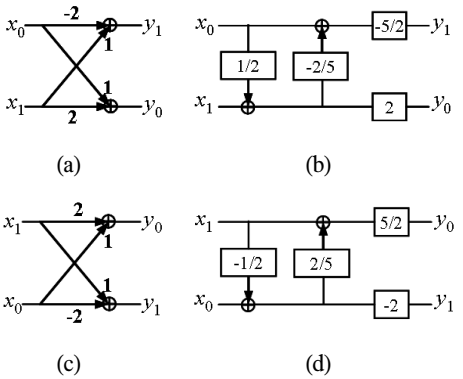

그림 6. 2x2 매트릭스 연산에 대한 permutation (a)와 (c)나비 연산 (b)와(d) 리프팅 구조.
Fig. 6. Permutation of 2x2 matrix operation (a) and (c) butterfly (b) and (d) lifting structure of (a) and (c).

그림 6에서 알 수 있듯이 변환 계수의 범위가 앞서 제시된 구조보다 획기적으로 감소한다는 것을 알 수 있다. 감소되는 범위는 k_0, k_1 파라미터에서 추정할 수 있다. 즉, $k_0=5/2$, $k_1=-2$는 무손실용 변환에서는 삭제되기 때문에 상대적으로 원 변환보다 k0, k1으로 나누는 효과로 인하여 계수의 범위가 줄어드는 효과가 난다. 결론적으로 무손실용 정수변환은 k_0, k_1값이 크게 선택하는 것이 효과적인 방법임을 알 수 있다.

그림 6과 그림4를 통해서 그림 1을 무손실용으로 변형이 가능하다. 변환 계수의 범위도 그림 2에서 소개된 기존 방법보다 상당히 감소함을 확인할 수 있다. 그러나 그림 6(d)에서 문제점이 하나가 더 남아있다. u 파라미터의 계수가 2/5로 5로 나누는 요소가 생긴 것이다. 이 부분은 복잡도에 많은 영향을 미치므로 그대로 사용하기에 하드웨어 구현 등에서 문제점이 있다. 또한 무손실용 정수 변환으로의 변형에서 중요한 점이 무손실 압축 성능이라고 볼 때, 2/5라는 수치를 적당한 값으로 근사해도 압축 성능이 큰 차이를 보이지 않을 가능성이 있다. 따라서 본 논문에서는 2/5를 적당한 근사치로 바꾸어 예를 들면 2/5를 1/2, 3/8, 13/32 등등으로 근사할 수 있다. 1/2로 근사하면 간단한 쉬프트 연산으로 구현되고 3/8로 근사하면 덧셈 연산 하나가 추가된다. 13/32로 근사하면 2개의 덧셈과 3개의 쉬프트 연산이 필요하게 된다.

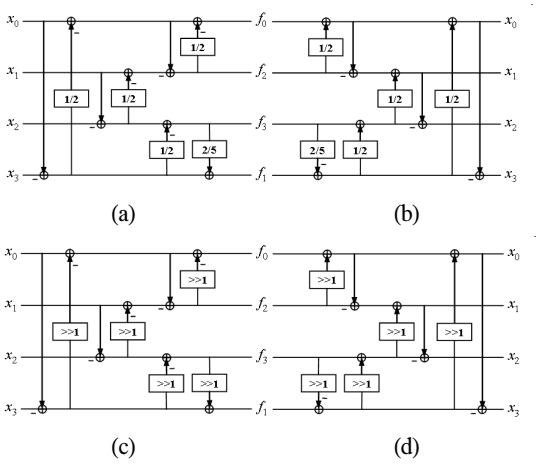

그림 7. 제안된 정수 변환 (a) 변환 (b) 역변환 (c) 근사변환 (d) 역변환.
Fig. 7. Proposed integer transform (a) transform (b) inverse transform (c) approximated version (d) inverse transform.

그림 7에서는 본 논문에서 최종적으로 제안된 정수 연산에 대한 고속 구조를 제시하고 있다. 여기서 2/5에 대한 근사 값은 1/2로 구현되었다. 이 근사된 정수 변환은 덧셈 연산 수면에서는 가장 간단한 형태인 하다마드 변환과 동일함으로 연산량 면에서는 상당히 효율적임을 알 수 있다.

Ⅳ. 실험 및 결과

본 논문에서 제안된 방법은 2/5의 값을 어떻게 근사하느냐에 따라서 몇 개의 근사 버전이 존재한다. H.264 정수 변환자체가 DCT를 최소한의 연산량으로 근사한 것임으로 무손실용으로 재차 근사화한 것이 성능에 좋은 영향일 지 아닐지는 실험을 통해서 검증을 해야 한다. 따라서 이 부분을 확인하고 제안된 방법이 기존 방법보다 성능면에서 우월함을 보이는지에 대한 실험을 한다.

일반적으로 변환의 성능 평가를 위해서 많이 사용되는 방안이 변환 이득을 비교하는 것이다. 이는 SNR 관점에서는 맞는 방법이지만, 무손실용 변환에서는 다소 문제가 있다. 즉, 무손실 변환은 손실이 없기 때문에 SNR 비교 차체가 의미가 없다. 따라서 변환 이득 비교보다는 1차 엔트로피를 비교하는 것이 더욱 합리적이기 때문에 제안된 방법을 평가하기 위해서 본 논문에서는 1차 엔트로피를 비교평가 한다. 이에 대한 입력 데이터로는 AR(1) 소스와 실제 영상을 이용해서 실험을 진행한다.

표 1. AR(1) 소스에 대한 실험 결과
(ρ=0.95, 분산=16, 엔트로피=5.673bpp)
Table 1. Experimental results for AR(1) source
(ρ=0.95, variance=16, entropy=5.673bpp)

		f_0	f_1	f_2	f_3	Ave
H.264		7.567	5.737	4.606	4.866	5.970
ours	0	5.613	4.829	3.619	3.870	4.687
	1	5.613	4.766	3.619	3.870	4.666
	1/2	5.613	4.748	3.619	3.870	4.660
	3/8	5.613	4.763	3.619	3.870	4.665
	2/5	5.613	4.761	3.619	3.870	4.664

표 2. AR(1) 소스에 대한 실험 결과
(ρ=0.5, 분산=16, 엔트로피=4.245bpp)
Table 2. Experimental results for AR(1) source
(ρ=0.5, variance=16, entropy=4.245bpp)

		f_0	f_1	f_2	f_3	Ave
H.264		5.756	5.540	4.823	5.206	5.373
ours	0	3.780	4.611	3.838	4.214	4.076
	1	3.780	4.697	3.838	4.214	4.105
	1/2	3.780	4.550	3.838	4.214	4.056
	3/8	3.780	4.546	3.838	4.214	4.055
	2/5	3.780	4.546	3.838	4.214	4.055

표 1, 2는 AR 소스에 대한 실험결과를 보여준다. 상관도가 높은 소스(ρ=0.95)와 상대적으로 낮은 소스(ρ=0.5)에 대한 결과를 보여준다. 실험 결과에서도 알 수 있듯이 2/5의 근사는 주로 f_1 데이터에 영향을 미친다. 이는 해당하는 수치의 변화가 f_1 커널의 모양새에 영향을 주기 때문이다. 표 1, 2에서 명확히 알 수 있는 것은 제안된 방법이 그림 2의 H.264 무손실 정수 변환보다 엔트로피 면에서 우수하다는 것이다. 또한 제안된 방법들 중에서 근사치가 1/2, 3/8를 적용한 결과와 원래의 값인 2/5를 적용한 것과는 대동소이한 것으로 보인다. 굳이 따지자면 오히려 1/2를 적용한 것이 원래의 값을 그대로 적용한 것보다 약간 더 우수한 결과를 보이고 있다. 이는 앞서 기술한 바와 같이 H.264 정수 변환 자체가 DCT의 근사 변환이기 때문에 1/2로 근사가 더더욱 효율이 좋을 수도 있다는 것을 실험적으로 보여준다.

표 3. Lena 영상에 대한 실험 결과
Table 3. Experimental results for the Lena image

		f_0	f_1	f_2	f_3	Ave
H.264		9.412	6.346	5.173	5.319	6.977
ours	0	7.420	5.432	4.183	4.323	5.678
	1	7.420	5.395	4.183	4.323	5.666
	1/2	7.420	5.352	4.183	4.323	5.652
	3/8	7.420	5.361	4.183	4.323	5.655
	2/5	7.420	5.359	4.183	4.323	5.654

표 4. Baboon 영상에 대한 실험 결과
Table 4. Experimental results for the Baboon image

		f_0	f_1	f_2	f_3	Ave
H.264		9.187	8.073	7.362	7.585	8.207
ours	0	7.194	7.123	6.366	6.589	6.894
	1	7.194	7.203	6.366	6.589	6.921
	1/2	7.194	7.077	6.366	6.589	6.879
	3/8	7.194	7.070	6.366	6.589	6.877
	2/5	7.194	7.070	6.366	6.589	6.877

표 3, 4는 Lena와 Baboon영상에 대한 실험결과를 보여준다. 상관도가 높은 영상인 Lena와 상대적으로 낮은 영상인 Baboon에 대한 결과를 보여준다. 이 결과도 앞서 제시한 표 1, 2의 결과와 유사한 경향을 보임을 알 수 있다. 결론적으로, 성능과 연산량을 모두 고려하면 제안된 방법들 중에서 그림 7(c)에 제시된 것과 같은 1/2로 근사한 방법이 가장 추천된다.

Ⅴ. 결 론

본 논문에서는 H.264/AVC에서 이용되는 정수 변환에 대한 무손실용 정수 변환으로의 변형 방법에 대해서 소개했고 그 결과를 고속 구현 형태로 제시하였다. 또한 그 제시된 방법들을 기존 방법과 비교하여 무손실 압축 성능에서 우수함을 보였다. 실험 결과로 본 논문에서 근사된 방법 중 1/2로 근사된 방법이 가격 대비 성능이 좋은 것으로 제시되었다.

참고문헌

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE Trans. Circ. Syst. Video Tech., Vol. 13, No. 7, July 2003.

[2] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization in H.264/AVC," IEEE Trans. Circ. Syst. Video Tech., Vol 13, No. 7, July 2003.

[3] A. Hallapuro, M. Karczewicz, and H. Malvar, "Low complexity transform and quantization - Part II:Extentions" ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6 JVT B-039, Jan., 2002.

[4] Y.-J Chen, S. Oraintara, and T.Q. Nguyen, "Video compression using integer DCT," in Proc. ICIP, Sept. 2000.

[5] J. Liang and T.D. Tran, "Fast multiplierless approximations of the DCT with the lifting scheme," IEEE Trans. Signal Processing,Vol 49, No. 12, Dec. 2001.

[6] Y.-J Chen, S. Oraintara, T.D. Tran, K. Amaratunga, and T.Q. Nguyen, "Multiplierless approximation of transforms with adder constraint," IEEE Signal Processing Letters, Vol. 9, No. 11, Nov. 2002.

[7] T. Yng, B. Lee, and H. Yoo, "A low complexity and lossless frame memory compression for display devices," IEEE Trans. Consumer Electronics, Vol. 54, No. 3, Aug. 2008.

[8] P. Lux, "A novel set of closed orthogonal functions for picture coding," Arch. Elek. Ubertragung, Vol. 31, pp. 267-274, 1977.

[9] K. Ire and R. Kishmoto, "A study on perfect reconstructive subband coding," IEEE Trans. Circuits and Syst. Video Tech., pp.42-48, Mar. 1991

저자소개

한국해양정보통신학회 논문지
제12권 12호 참조