논문 2007-44SP-6-7

H.264/AVC에서 변환계수의 부분집합을 사용한 인트라 16×16 예측 모드 선택 방법

(Intra 16×16 Mode Decision Using Subset of Transform Coefficients in H.264/AVC)

임 상 희*, 이 성 원**, 백 준 기***

(Sanghee Lim, Seongwon Lee, and Joonki Paik)

요 약

본 논문에서는 기존의 고속 인트라 4×4 예측 모드 선택 알고리즘을 인트라 16×16 예측에 적용하여 연산량을 감소시키고, 변환 계수의 일부분만을 사용하여 인트라 16×16 예측 모드 선택을 고속으로 수행하는 방법을 제안한다. 제안하는 방법은 기준 블록과 예측 블록간의 차분에 4×4 변환을 수행한 후 DC계수 하나와 주변의 AC계수 3개만을 사용하여 모드 선택을 수행한다. 이론적인 분석과 실험적 결과를 통해서 제안한 방법을 사용하여 인트라 16×16 예측 모드 선택을 수행할 경우 부호화 효율의 큰 감소 없이 필요한 연산량을 50%까지 줄일 수 있었다.

Abstract

In this paper, we significantly reduces the amount of computation for intra 16×16 mode decision in H.264 by applying the fast algorithm, which obtains the transformed prediction residual with fewer computations. By extending the existing intra 4×4 mode decision, we propose the new algorithm for fast intra 16×16 mode decision. The proposed algorithm uses partial transform coefficients which consist of one DC and three adjacent AC coefficients after 4×4 transform in the intra 16×16 mode decision. Theoretical analysis and experimental results show that the proposed algorithm can reduce computations up to 50% in the intra 16×16 mode decision process with unnoticeable degradation.

Keywords : H.264/AVC, intra coding, mode decision

I. 서 론

H.264는 ISO/IEC의 MPEG과 ITU-T의 VCEG 두 그룹이 공동연구기관인 JVT를 창설하여 새롭게 제안한 동영상 압축이다. H.264는 기존의 다른 압축 표준들보다 향상된 압축 성능을 제공하는데, 이는 가변 블록 크기의 움직임 예측 및 보상, 정수 변환 및 양자화, 1/4 화소 단위의 화면 간 예측, 다중 참조 프레임 기반의 움직임 예측 및 보상, 방향성을 고려한 화면 내 예측 부호화, 향상된 블로킹제거 필터, 향상된 엔트로피 부호화 등의 신기술을 도입하였기 때문이다.

* 학생회원, *** 평생회원, 중앙대학교 첨단영상대학원 영상공학과
(Dept. of Image Engineering, Graduate School of Advanced Imaging Science, Multimedia, and Film, Chung-Ang University)
** 평생회원, 광운대학교 컴퓨터공학과
(Dept. Computer Engineering, Kwangwoon University)

※ 이 연구는 2007년도 정부(과학기술부)의 재원으로 한국과학재단(NRL)의 지원과 서울시 산학연협력사업으로 구축된 서울 미래형콘텐츠컨버전스 클러스터 (SFCC)의 지원과 2007년도 중앙대학교 우수연구자연구비 지원에 의한것임.
접수일자: 2007년7월18일, 수정완료일: 2007년10월31일

반면 이러한 새로운 알고리즘과 특성들이 추가됨에 의해 연산의 복잡도가 크게 증가하였으며, 이를 줄이기 위한 연구가 한창 진행 중에 있다[1~5]. 특히, Chao-Hsuing Tseng은 인트라 4×4 모드 선택을 위해 기존에 사용하고 있던 sum of absolute difference (SAD)나 sum of absolute Hadamard-transformed difference (SATD)이외에 정수 DCT를 사용하는 sum of absolute integer-transformed difference (SAITD)를 제안하였으며, 더불어 인트라 4×4의 각 예측 모드에 대해 SATD와 SAITD의 계산량을 줄이는 알고리즘을 제안하였다[1~2]. Tseng이 제안한 인트라 4×4 모드 선택을 위한 SATD의 고속 알고리즘은 인트라 4×4뿐만 아니라 SAD나 SATD로 모드 선택을 수행하고 있는 인트라 16×16에도 그대로 적용될 수 있다. 본 논문에서는 인트라 16×16에 Tseng의 알고리즘을 적용하여 연산량을 감소시키고, 모드 결정에 사용되는 계수의 개수를 줄이는 새로운 알고리즘을 사용하여 율-왜곡 (rate-distortion; R-D) 곡선은 거의 같게 유지하면서 필요한 연산량을 추가로 감소시키는 방법을 제안한다.

Ⅱ. 이론적인 배경

1. 인트라 16×16 모드 선택

H.264/AVC는 인트라 코딩에서 예측 블록을 사용하여 기준 블록과 예측 블록의 차분을 코딩하는 방법을 사용하고 있다. 이 때, 예측 블록은 인코딩되었다 다시 디코딩된 주변 블록의 화소로부터 얻어진다. H.264/AVC는 인트라 4×4 예측을 위해 9가지 모드를 제공하고, 인트라 16×16 예측을 위해 4가지 모드를 제공한다[6]. 각각의 모드 중에서 압축 성능이 가장 좋은 것을 선택하는 일은 전적으로 인코더에서 처리하는 작업이며 기준 소프트웨어 (JM)에서는 High-complexity (HC) 모드와 Low-complexity (LC) 모드 두 가지 방법을 사용하여 모드 선택을 수행하고 있다[7]. HC 모드에서는 각 예측 모드의 압축 효율을 측정하기 위해 다음과 같은 R-D 비용함수를 사용한다.

$$C_{RD} = D + \lambda(Q_P)R, \tag{1}$$

여기서 D는 sum of squared difference (SSD)로 계산되며, 이것은 원 영상에서의 매크로 블록과 압축했다가 다시 복원한 매크로 블록과의 차분을 제곱한 다음 모두 더한 것을 말한다. 또한, $\lambda(Q_P)$는 양자화 매개변수 Q_P 에 대한 지수함수이며, R은 부호화에 사용된 비트수로써 매크로 블록의 헤더에 들어가는 비트수와 양자화된 변환 계수를 엔트로피 코딩 한 결과 얻어지는 비트수를 합친 것이다. HC 모드 선택을 수행하면 실질적으로 압축 성능이 가장 좋은 예측 모드를 선택할 수 있지만 많은 연산량을 필요로 하는 부작용이 있다.

LC 모드에서는 인트라 4×4 예측 모드 9가지 중 하나를 선택할 때와 선택된 16개의 인트라 4×4 예측 모드들과 선택된 인트라 16×16 예측 모드 중 하나를 선택할 때 R-D 비용함수를 사용한다. LC 모드에서 사용하는 R-D 비용함수도 식 (1)처럼 표현되지만 이때의 D는 SAD 혹은 SATD로 계산되고, R은 매크로 블록의 헤더비트부분만을 고려하여 계산된다. SAD는 16×16 예측 차분의 절대치를 모두 더하는 것이고, SATD는 16×16 예측 차분에 대해 4×4 하다마드 (Hadamard) 변환을 16번 수행한 다음 각각의 변환 계수에서 DC값 16개를 다시 하다마드 변환하여 모든 계수의 절대치를 더하는 것이다. LC 모드에서 16×16 예측 모드 4가지 중 압축 성능이 좋은 하나를 선택하기 위해서는 식 (1)에서 D (SAD 혹은 SATD)만을 사용한다. 일반적으로 LC 모드에서 SAD보다 SATD의 성능이 우수한 것으로 알려져 있다.

2. 고속 인트라 4×4 모드 선택을 위한 Tseng의 알고리즘

H.264/AVC의 모드 선택에서 사용하고 있는 SATD의 하다마드 변환은 다음과 같이 표현된다.

$$T_H(A-B)T_H^T, \tag{2}$$

여기서 A는 기준 4×4 블록이고, B는 예측된 4×4 블록이며, T_H는 다음과 같이 정의된다.

$$T_H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{3}$$

H.264/AVC에서 사용하고 있는 4×4 하다마드 변환과 4×4 정수 DCT는 분리가능 (separable)한 성질을 가지고 있어서, 2차원 변환은 1차원 변환을 독립적으로 두 번 수행함으로 구현할 수 있다. 임의의 4개의 정수 데이터에 버터플라이 (butterfly) 알고리즘을 적용하여 하다마드 변환을 수행할 경우 덧셈연산이 8번 필요하

표 1. 인트라 4×4 예측 블록의 하다마드 변환과 정수 DCT에 Tseng의 고속 알고리즘을 적용하였을 때 필요한 연산량 비교

Table 1. The amount of computation for the transform of intra 4×4 prediction block by using Tseng's fast algorithm.

변환 / 모드	하다마드		정수 DCT	
	덧셈	쉬프트	덧셈	쉬프트
0	8	4	8	6
1	8	4	8	6
2	0	1	0	1
3	28	5	34	13
4	32	5	38	13
5	36	6	58	16
6	36	6	58	16
7	36	6	58	16
8	36	6	58	16

고, 정수 DCT를 수행할 경우 덧셈 연산 8번과 쉬프트 연산 2번이 필요하다. 따라서, 임의의 4×4 정수 데이터에 하다마드 변환을 수행할 경우 덧셈 연산이 64번 필요하고, 정수 DCT를 수행할 경우 덧셈 연산 64번과 쉬프트 연산 16번이 필요하다.

Tseng의 고속 알고리즘은 하다마드 변환을 다음 식과 같이 표현하여 기준 블록과 예측 블록의 하다마드 변환을 먼저 구한 후 두 변환 계수의 차분을 구하는 것이다[2].

$$T_H A T_H^T - T_H B T_H^T. \quad (4)$$

식 (4)를 사용해서 인트라 4×4에 주어진 각 모드에 대해 SATD의 하다마드 변환 계수를 구할 때 $T_H A T_H^T$는 한 번만 수행되며, 각 모드에 대해 $T_H B T_H^T$만 구하여 빼기만 하면 된다. 각 모드에 대해 예측 블록의 하다마드 변환인 $T_H B T_H^T$를 구할 때 필요한 연산은 임의의 4×4 데이터에 하다마드 변환을 수행하는 것보다 더 적은데, 그 이유는 각 모드의 예측 블록은 특정한 패턴을 가지기 때문이다. 인트라 4×4 예측 모드들에 대해 하다마드 변환을 수행할 때 필요한 연산량이 표 1에 정리되었다.

Ⅲ. Tseng의 고속 알고리즘을 적용한 인트라 16×16 모드 선택

H.264/AVC에서는 인트라 모드를 위해 인트라 4×4 예측 모드와 인트라 16×16 예측 모드를 제공한다. 인

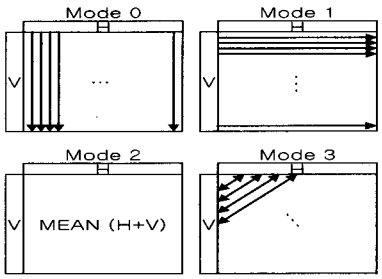


그림 1. 인트라 16×16 예측 모드

Fig. 1. Intra 16×16 prediction mode.

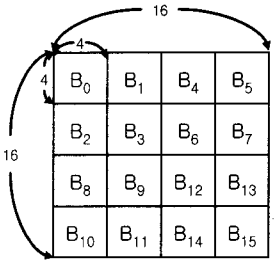


그림 2. 16개의 4×4 블록으로 분할된 인트라 16×16 예측 블록

Fig. 2. An intra 16×16 prediction block which is divided into sixteen 4×4 blocks.

트라 16×16 예측에는 그림 1과 같이 4가지 모드가 제공되는데, 이 중 모드 0, 1, 2의 경우 예측 블록 패턴이 정해져 있어서 식 (4)를 사용한 Tseng의 고속 알고리즘이 동일하게 적용될 수 있다. 또한 인트라 16×16 예측 블록은 그림 2와 같이 4×4 블록이 16개 있어서 4×4 변환이 16번 수행되어야 하지만 모드 0, 1, 2의 경우 4×4 예측 블록이 특정한 패턴을 가지므로 각각 16번씩 모두 수행할 필요는 없다. 예를 들어 모드 0의 인트라 16×16 예측 블록의 경우 서로 다른 패턴을 가지는 블록은 B_0, B_1, B_4, B_5이므로 총 4번의 4×4변환만이 필요하다. 물론 모드 3의 경우 예측 블록내에 반복되는 패턴이 없으므로 4×4 변환이 16번 수행되어야 한다.

본 장에서는 인트라 16×16 모드 선택을 SATD로 수행할 때 Tseng의 고속 알고리즘을 적용할 경우 필요한 연산량을 확인하기로 한다. 또한 Tseng이 인트라 4×4 모드 선택에서 제안한 SAITD를 인트라 16×16 모드 선택에 적용할 경우 Tseng의 고속 알고리즘에 필요한 연산량을 확인한다. 인트라 16×16 모드 선택을 위한 SAITD는 예측 차분에 16회 적용되는 4×4 변환이 하다마드 변환이 아닌 정수 DCT를 사용한다는 것 이외의 모든 과정은 SATD와 동일하다.

예측 차분에 16회 적용되는 4×4 변환이 하다마드 변환과 정수 DCT중 어떤 것이 되건 식 (2)대신 식 (4)를

사용하면 연산량을 줄일 수 있는데, 그 이유는 식 (4)에서 TAT^T는 한번만 계산하면 되고, 특별한 패턴을 가지는 각 모드의 TBT^T만 구하여 빼주기만 하면 되기 때문이다. 여기서 A는 원 16×16 매크로 블록을, B는 16×16 예측 블록을, 그리고 TAT^T는 4×4 변환을 16회 적용한 것을 의미한다. TAT^T에 필요한 연산량은 임의의 4×4 정수 데이터에 필요한 연산량에 16배를 취하면 되므로 하다마드 변환을 사용하는 SATD의 경우 덧셈 연산 1024번이 필요하고, 정수 DCT를 사용하는 SAITD의 경우 덧셈 연산 1024번과 쉬프트 연산 256번이 필요하다. 각각의 인트라 16×16 예측 블록의 변환 (TBT^T)을 위해 필요한 연산량은 다음과 같이 결정된다.

모드 0일 때는 그림 2의 인트라 16×16 예측 블록 중 B_0, B_1, B_4, B_5만이 서로 다른 값으로 구성되므로 총 4번의 4×4변환만이 필요하다. B_0, B_1, B_4, B_5는 모두 표 1의 인트라 4×4 예측 모드 0과 같은 패턴을 가지므로 TBT^T를 위해 필요한 연산은 하다마드 변환일 경우 덧셈 연산 32번과 쉬프트 연산 16번, 정수 DCT일 경우 덧셈 연산 32번과 쉬프트 연산 24번이 필요하다.

모드 1일 때는 그림 2에서 B_0, B_2, B_8, B_{10}만이 서로 다른 값으로 구성되며, 이 4×4 블록들은 모두 인트라 4×4 예측 모드 1과 같은 패턴을 가진다. 따라서 TBT^T를 위해 필요한 연산은 하다마드 변환일 경우 덧셈 연산 32번과 쉬프트 연산 16번, 정수 DCT일 경우 덧셈 연산 32번과 쉬프트 연산 24번이 필요하다.

모드 2일 때는 B_i ($0 \le i \le 15$)가 모두 같은 값을 가지며, 이 4×4 블록은 인트라 4×4 예측 모드 2와 같은 패턴을 가진다. 이 경우 TBT^T를 위해 필요한 연산은 하다마드와 정수 DCT 중 어느 것을 사용하든 쉬프트 연산 1번만이 필요하다.

모드 3일 때는 16×16 예측 블록을 구성하는 모든 4×4 블록 (B_i, $0 \le i \le 15$)들이 특별한 패턴을 가지지 않으므로 4×4변환을 16회 수행하여야 하며, TBT^T를 위해 필요한 연산은 TAT^T를 위해 필요한 연산과 같다.

지금 까지는 식 (4)의 TAT^T과 TBT^T부분을 계산하는데 필요한 연산만을 고려하였는데, 인트라 16×16 모드 선택 과정과 선택된 모드에 대해 수행되는 정수 DCT까지 고려하였을 때 필요한 총 연산량을 표 2에 정리하였다. 모드 선택 후 수행되는 정수 DCT에 필요한 연산량은 인트라 4×4 모드까지 고려한 최종적인 모드 선택에서 인트라 16×16 모드가 선택되었음을 가정한 것이다. 표 2에서 굵은체로 표시된 TAT^T와 TBT^T, $TAT^T - TBT^T$ 부분을 식 (2)의 $T(A-B)T^T$에 필요한 연산량으로 대체하고, 대괄호 속에 있는 연산량을 제외하면 식 (2)를 사용한 원래의 방식으로 모드 선택을 수행하고 모드 선택 후 수행되는 정수 DCT에 필요한 총 연산량을 얻을 수 있다. 식 (2)를 사용하

표 2. Tseng의 고속 알고리즘을 사용한 인트라 16×16 모드 선택과 모드 선택 후 정수 DCT에 필요한 연산량 ([ ]안의 내용은 원래의 방식에서 필요한 연산을 수행할 때 제외되어야 하는 부분)

Table 2. The amount of computation for the intra 16×16 mode decision and integer DCT of selected modes by using Tseng's fast algorithm (computation in [ ] should be excluded when we consider the original algorithm).

		SATD와 정수 DCT	SAITD와 정수 DCT
TAT^T		덧셈: 1024, 쉬프트: 0	덧셈: 1024, 쉬프트: 256
TBT^T	모드 0	덧셈: 32, 쉬프트: 16	덧셈: 32, 쉬프트: 24
	모드 1	덧셈: 32, 쉬프트: 16	덧셈: 32, 쉬프트: 24
	모드 2	덧셈: 0, 쉬프트: 1	덧셈: 0, 쉬프트: 1
	모드 3	덧셈: 1024, 쉬프트: 0	덧셈: 1024, 쉬프트: 256
$TAT^T - TBT^T$		덧셈: 256×4=1024	
변환 계수의 절대치 모두 더함 (DC 계수 제외)		덧셈: 240×4=960	
4×4 DC계수 하다마드 변환		덧셈: 64×4=256	
DC 하다마드 변환 계수의 절대치 모두 더함		덧셈: 16×4=64	
모드 선택 후 선택된 모드에 대한 정수 DCT		[덧셈: 256 (A-B)] 덧셈: 1024, 쉬프트: 256	추가 연산 필요 없음
정수 DCT 후 4×4 DC계수 하다마드 변환		덧셈: 64, 쉬프트: 0	추가 연산 필요 없음
총 연산		덧셈: 5760, 쉬프트: 289	덧셈: 4416, 쉬프트: 561

표 3. 원래 방식과 Tseng의 고속 알고리즘을 적용하였을 때 필요한 총 연산량

Table 3. Total amount of computation for mode decision with and without integer DCT for selected modes by using the original and Tseng's fast algorithm.

	원래 방식	Tseng의 고속 알고리즘
SATD	덧셈: 6400 쉬프트: 0	덧셈: 4416 쉬프트: 33
SAITD	덧셈: 6400 쉬프트: 1024	덧셈: 4416 쉬프트: 561
SATD와 정수DCT	덧셈: 7488 쉬프트: 256	덧셈: 5760 쉬프트: 289
SAITD와 정수DCT	덧셈: 6400 쉬프트: 1024	덧셈: 4416 쉬프트: 561

면 하나의 모드에 대해 $(A-B)$를 수행하는데 덧셈 연산 256번, $T(A-B)T^T$를 수행하는데 하다마드 변환일 경우 덧셈 연산 1024번, 정수 DCT일 경우 덧셈과 쉬프트 연산이 차례로 1024번과 256번이 필요하다.

식 (2)를 사용하여 모드 선택을 수행하고 모드 선택 후 수행되는 정수 DCT까지 고려하였을 때 필요한 총 연산량을 표 3에 정리하였다. 표 3을 보면 각각의 경우 원래의 방식보다 Tseng의 고속 알고리즘이 더 적은 연산량을 요구함을 알 수 있고, 모드 선택 후 정수 DCT까지 고려하였을 경우 SAITD와 정수 DCT를 사용한 것이 SATD와 정수 DCT를 사용한 것보다 더 적은 연산량이 필요함을 볼 수 있다.

Ⅳ. 제안하는 인트라 16×16모드 선택 방법

인트라 4×4 예측 모드는 복잡도가 높은 부분에서 선택될 확률이 높은 반면 인트라 16×16 예측 모드는 비교적 평탄한 영역에서 선택될 확률이 높다. 즉, 인트라 16×16 예측 모드가 선택되는 부분은 고주파 성분이 많이 포함되지 않은 부분임을 뜻하므로 본 논문에서 제안하는 알고리즘은 이러한 성질을 이용하여 SATD 혹은 SAITD로 모드 선택을 수행할 때 4×4 변환 계수를 모두 사용하지 않고 그림 3에 보이는 것처럼 DC계수 하나와 DC 주변의 AC 성분 3개만을 사용한다. 즉, 제안하는 인트라 16×16 모드 선택 알고리즘은 4×4변환을 16회 수행한 후 DC계수 주위의 AC계수 48 (3×16)개와 하다마드 변환을 한 번 더 거친 DC계수 16개의 절대치를 모두 합하여 적은 값을 가지는 모드를 선택한다. 변

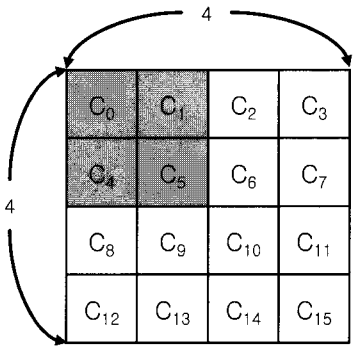


그림 3. 인트라 16×16 모드 선택에 사용되는 4개의 4×4 변환 계수

Fig. 3. Four low-frequency coefficients of 4×4 transform used in 16×16 mode decision.

환 계수의 일부분만을 사용하여 모드 선택을 수행하면 4×4변환을 수행할 때 모든 계수를 다 구할 필요 없이 DC계수 하나와 AC계수 3개만 구하면 되므로 필요한 연산량을 현저하게 줄일 수 있으며, 화질을 유지하면서 모드 선택을 수행할 수 있다.

SATD로 모드 선택을 수행할 경우 4×4 변환의 16개 계수를 모두 구할 필요 없이 그림 3의 C_0, C_1, C_4, C_5 의 4개만 구하면 된다. 즉, 2차원 하다마드 변환을 수행할 때 열 방향으로는 1차원 하다마드 변환을 모두 수행하고, 행 방향으로는 1행과 2행에 대해서만 1차원 하다마드 변환을 수행하면 된다. 행 방향으로 1차원 하다마드 변환을 수행할 때도 한 행에 대해 변환 계수 4개를 모두 구할 필요 없이 처음 2개의 변환 계수만 구하면 된다. SAITD로 모드 선택을 수행할 경우도 SATD와 마찬가지로 저 주파수의 4개 계수만 구하면 된다. 변환 계수의 일부분만을 사용하는 인트라 16×16 모드 선택에서도 Tseng의 고속 알고리즘을 적용하며, 이 때 필요한 연산의 자세한 내용은 아래와 같다.

Ⅲ장에서와 같이 Tseng의 고속 알고리즘을 적용하면 각 모드의 예측 차분의 변환 계수를 구할 때 TAT^T는 한 번만 계산하면 되고, 각 모드의 TBT^T를 구하여 TAT^T에서 빼면 된다. 이 때 TAT^T는 4개의 변환 계수만을 구하는 4×4변환을 16회 적용한 것을 의미 한다. TAT^T 위해 필요한 연산은 하다마드 변환일 경우 덧셈 연산 704번, 정수 DCT일 경우 덧셈 연산 704번과 쉬프트 연산 160번이 필요하다.

모드 0에서는 TBT^T를 구할 때 그림 2의 B_0, B_1, B_4, B_5의 4×4블록에 대해서만 변환 계수의 일부분을 얻으면 된다. 따라서 하다마드 변환일 경우 덧셈 연산 24번

과 쉬프트 연산 8번, 정수 DCT일 경우 덧셈 연산 24번과 쉬프트 연산 12번이 필요하다.

모드 1에서는 TBT^T를 구할 때 그림 2의 B_0, B_2, B_8, B_{10}의 4×4블록에 대해서만 변환 계수의 일부분을 얻으면 된다. 따라서 하다마드 변환일 경우 덧셈 연산 24번과 쉬프트 연산 8번, 정수 DCT일 경우 덧셈 연산 24번과 쉬프트 연산 12번이 필요하다.

모드 2에서는 TBT^T를 구할 때 그림 2의 B_0의 4×4 블록에 대해서만 변환 계수의 일부분을 얻으면 된다. 따라서 하다마드 변환일 경우와 정수 DCT일 경우 모두 쉬프트 연산 1번이 필요하다.

모드 3에서는 TBT^T를 구할 때 그림 2의 모든 4×4 블록에 대해서 변환 계수의 일부분을 얻어야 한다. 따라서 하다마드 변환일 경우 덧셈 연산 704번, 정수 DCT일 경우 덧셈 연산 704번과 쉬프트 연산 160번이 필요하다.

SATD로 모드 선택을 수행하고 선택된 모드에 대해 정수 DCT를 수행할 때는 새롭게 예측 차분을 구하여 정수 DCT를 수행하여야 한다. SAITD로 모드 선택을 수행하였을 때는 선택된 모드에 대해 정수 DCT를 수행할 때 새롭게 수행하지 않고 모드 선택에서 구하지 않은 나머지 변환 계수만을 구하면 되는데 이 때 필요한 연산량은 어떤 모드가 선택되느냐에 따라 달라진다. 먼저 TAT^T부분의 모드 선택에서 구하지 않은 나머지 변환 계수들을 구해야 하는데 이 때 필요한 연산은 덧셈 320번과 쉬프트 96번이다. 만약 선택된 모드가 모드 0나 모드 1이라면 추가적으로 필요한 연산은 덧셈 연산 8번과 쉬프트 연산 12번이고, 모드 3이면 덧셈 연산 320번과 쉬프트 연산 96번이 추가적으로 필요하다. 모드 2일 때는 모드 선택을 수행할 때 필요한 변환 계수를 모두 구해놓은 상태이므로 정수 DCT를 수행할 때 추가적인 연산이 필요하지 않다.

제안하는 알고리즘으로 인트라 16×16 모드 선택을 수행하고 인트라 코딩에서 모든 모드를 선택한 후 최종적으로 인트라 16×16 모드가 선택되어 정수 DCT까지 수행되었을 때 필요한 연산량을 표 4에 정리하였다. 표 4에서 [ ]안의 연산은 SAITD로 모드 선택을 수행하고 모드 선택 후 정수DCT를 수행할 때 추가로 필요한 연산을 나타낸다. SAITD로 모드 선택 후 정수 DCT를 수행할 때 추가되는 연산은 어떤 모드가 선택되느냐에 따라 필요한 추가적인 연산이 다르므로 총 연산 또한 어떤 모드가 선택되느냐에 따라 달라진다.

표 4. 변환 계수의 일부분만을 사용한 인트라 16×16 모드 선택과 모드 선택 후 정수 DCT에 필요한 총 연산량 ([ ]안의 내용은 SAITD로 모드 선택 후 정수 DCT를 수행할 때 추가로 필요한 연산)

Table 4. The amount of computation for the intra 16×16 mode decision by using partial transform coefficients and integer DCT of selected modes (computation in [ ] should be included when we consider the integer DCT of selected mode).

		SATD와 정수 DCT	SAITD와 정수 DCT
TAT^T		덧셈: 704, 쉬프트: 0	덧셈: 704, 쉬프트: 160 [덧셈: 320, 쉬프트: 96]
TBT^T	모드 0	덧셈: 24, 쉬프트: 8	덧셈: 24, 쉬프트: 12
	모드 1	덧셈: 24, 쉬프트: 8	덧셈: 24, ,쉬프트: 12
	모드 2	덧셈: 0, 쉬프트: 1	덧셈: 0, 쉬프트: 1
	모드 3	덧셈: 704, 쉬프트: 0	덧셈: 704, 쉬프트: 160
$TAT^T - TBT^T$		덧셈: 64×4=256 [덧셈: 192]	
변환 계수의 절대치 모두 더함 (DC 계수 제외)		덧셈: 48×4=192	
4×4 DC계수 하다마드 변환		덧셈: 64×4=256	
DC 하다마드 변환 계수의 절대치 모두 더함		덧셈: 16×4=64	
모드 선택까지 필요한 총 연산량		덧셈: 2224, 쉬프트: 17	덧셈: 2224, 쉬프트: 345
모드 선택 후 선택된 모드에 대한 정수 DCT		덧셈: 256 (A-B) 덧셈: 1024, 쉬프트: 256	[모드 0 ➡ 덧셈: 8,쉬프트: 12] [모드 1 ➡ 덧셈: 8, 쉬프트: 12] [모드 2 ➡ 덧셈: 0, 쉬프트: 0] [모드 3 ➡ 덧셈: 320, 쉬프트: 96]
정수 DCT 후 4×4 DC계수 하다마드 변환		덧셈: 64, 쉬프트: 0	추가연산 필요 없음
모드 선택 후 정수 DCT까지 고려하였을 때 필요한 총 연산		덧셈: 3568, 쉬프트: 273	모드 0 ➡ 덧셈: 2744, 쉬프트: 453 모드 1 ➡ 덧셈: 2744, 쉬프트: 453 모드 2 ➡ 덧셈: 2736, 쉬프트: 441 모드 3 ➡ 덧셈: 3056, 쉬프트: 537

표 4를 보면 SAITD와 정수 DCT의 총 연산이 최대 덧셈 연산 3056번, 쉬프트 연산 537번이고 (모드 3), 이 연산량은 SATD와 정수 DCT의 총 연산량(덧셈 연산 3568번, 쉬프트 연산 273번) 보다 적음을 확인할 수 있다. 추가적으로 표 3의 모든 변환 계수를 사용한 경우의 총 연산량과 비교했을 때 표 4의 변환 계수의 일부분을 사용한 경우의 총 연산량이 더 적음을 확인할 수 있다.

V. 실험 결과

기존 알고리즘과 제안하는 알고리즘의 비교 실험을 위해 QCIF 영상 3개 (News, Mother and Daughter, Container)와 CIF 영상 3개 (Tempete, Paris, Mobile)를 사용하였으며, 각 시퀀스당 150 프레임을 모두 인트라 코딩하였다. 실험에 사용된 인트라 16×16 모드 선택 방식은 4×4 변환 계수를 모두 사용한 SATD ($SATD_{all}$), 4×4 변환 계수 중 저주파 4개만 사용한 SATD ($SATD_4$), 4×4 변환 계수를 모두 사용한 SAITD ($SAITD_{all}$), 4×4 변환 계수 중 저주파 4개만 사용한 SAITD ($SAITD_4$)으로 총 4가지이다. QP값은 20, 24, 28, 32, 36, 40을 사용하여 각 비트율에 대한 휘도(Y)성분의 PSNR을 비교하였으며, 참조 소프트웨어 JM10.2을 사용하여 실험을 수행하였다.

표 5는 News 영상의 각 16×16 모드 선택 방법에 대한 QP값에 따른 비트율과 Y성분의 PSNR을 나타내고, 그림 4는 이에 대한 R-D곡선을 나타낸다. 다른 영상을 사용한 실험에서도 이와 비슷한 결과를 얻었다. 다만 Container영상에서는 $SAITD_{all}$의 압축 성능이 $SATD_{all}$ 보다 좀 더 떨어지고, $SATD_{all}$을 사용했을 때 보다 $SAITD_{all}$를 사용했을 때 상대적으로 16×16 모드보다 4×4 모드가 더 많이 선택되는 결과를 보여주었다. 이것은 인트라 코딩의 4×4 모드와 16×16 모드의 코딩 효율 비교에서 16×16 모드가 더 좋음을 판단하는 것은 $SAITD_{all}$보다 $SATD_{all}$이 더 우수함을 알려준다. 그러나 Container 영상에서도 $SAITD_{all}$의 압축 성능이 $SATD_{all}$ 보다 좀 더 떨어지는 결과를 보여준 것을 제외하고는 다른 영상들과 비슷한 결과를 보여주었다.

그림 4를 보면 $SATD_{all}$와 $SAITD_{all}$의 압축 성능은 거의 비슷하면서 가장 우수한 것을 확인할 수 있다. 또한, $SAITD_4$는 $SATD_4$보다 압축 성능이 더 우수하며 4×4 변환 계수를 모두 사용한 $SATD_{all}$ 혹은 $SAITD_{all}$

표 5. news영상의 각각의 인트라 16×16 모드 선택 방법에 대한 비트율과 Y성분의 PSNR

Table 5. The bit-rates and PSNRs of Y component about individual intra 16×16 mode decision methods in news sequence.

	$SATD_{all}$		$SATD_4$	
QP	비트율 (kbits/s)	PSNR_Y (dB)	비트율 (kbits/s)	PSNR_Y (dB)
20	1485.73	43.685	1610.24	43.658
24	1105.44	40.561	1213.16	40.534
28	812.81	37.591	906.64	37.521
32	586.38	34.384	660.14	34.342
36	410.01	31.42	471.14	31.26
40	281.41	28.648	320.78	28.46
	$SAITD_{all}$		$SAITD_4$	
QP	비트율 (kbits/s)	PSNR_Y (dB)	비트율 (kbits/s)	PSNR_Y (dB)
20	1489.68	43.709	1509.43	43.684
24	1110.35	40.554	1127.88	40.574
28	817.18	37.589	832.3	37.579
32	588.85	34.373	603.55	34.395
36	412	31.395	425.56	31.356
40	284.15	28.642	294.09	28.563

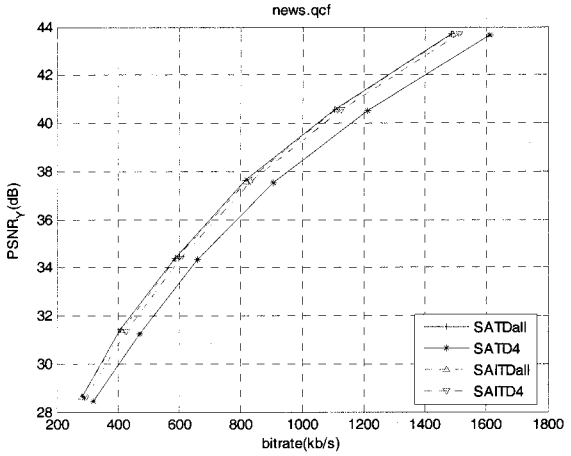


그림 4. 다양한 16×16 모드 선택 방법에 대한 R-D 곡선

Fig. 4. R-D curves obtained by various intra 16×16 mode decision algorithms.

보다 압축 성능이 다소 낮은 것을 확인할 수 있다. 표 4에 정리된 연산량과 함께 비교하면 제안하는 알고리즘 중 $SAITD_4$가 $SATD_4$보다 모드 선택까지는 연산량이 약간 더 많지만 모드 선택 후 정수 DCT까지는 $SAITD_4$의 연산량이 더 적으며 그림 4의 R-D 곡선도 $SAITD_4$가 훨씬 좋기 때문에 $SATD_4$보다 $SAITD_4$를 사용하는 것이 타당하다. $SAITD_4$의 연산량과 표 3에 제

시되어 있는 $SATD_{all}$와 $SAITD_{all}$의 연산량 (Tseng의 고속 알고리즘을 적용했을 때의 연산량)을 비교해 보면 모드 선택까지 뿐만 아니라 모드 선택 후 정수 DCT까지 모두 $SAITD_4$의 연산량이 더 적으며, 모드 선택까지는 $SAITD_4$의 연산량이 거의 절반가까이 떨어진다. 비록 그림 4에서 $SATD_{all}$의 R-D 곡선보다 $SAITD_4$의 R-D 곡선이 좋지 않게 나오지만 전체 비트율에 걸쳐 동일한 비트율에 대한 PSNR의 차가 평균적으로 0.5dB 이하이기 때문에 연산량이 훨씬 적게 드는 $SAITD_4$를 인트라 16×16 모드 선택 방법으로 사용하기에 적당하다고 할 수 있다.

VI. 결 론

본 논문에서는 인트라 16×16 예측 모드는 주로 평탄한 영역에서 선택된다는 성질을 이용하여 변환 계수의 일부분만을 사용한 인트라 16×16 모드 선택 방법을 제안하고, 변환 계수를 모두 이용한 $SATD_{all}$와 $SAITD_{all}$ 그리고 변환 계수의 일부분만을 이용한 $SATD_4$와 $SAITD_4$로 인트라 16×16 모드 선택을 수행하였을 때 필요한 연산량과 R-D 성능을 비교하였다. 변환 계수를 모두 이용한 $SATD_{all}$와 $SAITD_{all}$의 필요한 연산량은 예측 차분을 먼저 구한 후 변환을 수행하는 기존의 방법보다 기준 블록과 예측 블록을 먼저 변환한 후 차분을 구하는 Tseng의 알고리즘을 사용하였을 경우가 더 적었으며 변환 계수의 일부분만을 사용하였을 경우 연산량을 추가로 줄일 수 있었다. 특히 모드 선택까지만 고려하여 변환 계수를 모두 사용한 경우와 비교했을 때 제안한 알고리즘의 연산량은 거의 50%까지 감소했다. 물론 R-D 성능은 변환 계수의 일부분만을 사용한 경우가 변환 계수 모두를 사용한 경우보다 못하지만 $SAITD_4$의 경우 $SATD_{all}$에 비해 동일한 비트율에서 PSNR의 차가 0.5dB 이하로 그 차이는 크지 않았다.

참 고 문 헌

[1] C. Tseng, H. Wang, and J. Yang, "Enhanced intra-4×4 mode decision for H.264/AVC coders," IEEE Trans. Circuits Syst. Video Technol., vol. 16, no. 8, pp. 1027-1032, Aug. 2006.

[2] C. Tseng, H. Wang, and J. Yang, "Improved and fast algorithms for intra 4×4 mode decision in H.264/AVC," Proc. IEEE Int. Symp. Circuits Syst., Kobe, Japan, pp. 2128-2131, May 2005.

[3] Y. Huang, B. Hsieh, T. Chen, and L. Chen, "Analysis, fast algorithm, and VLSI architecture design for H.264/AVC intra frame coder," IEEE Trans. Circuits Syst. Video Technol., vol. 15, no. 3, pp. 378-401, Mar. 2005.

[4] F. Pan, X. Lin, S. Rahardja, K. Lim, Z. Li, D. Wu, and S. Wu, "Fast mode decision algorithm for intraprediction in H.264/AVC video coding," IEEE Trans. Circuits Syst. Video Technol., vol. 15, no. 7, pp. 813-822, July 2005.

[5] Y. Kim, J. Yoo, S. Lee, J. Shin, J. Paik, and H. Jung, "Adaptive mode decision for H.264 encoder," Electronics Letters, vol. 40, no. 19, pp. 1172-1173, Sep. 2004.

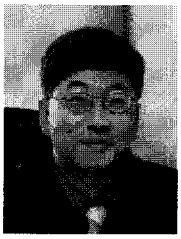
[6] Draft ITU-T Rec. and FDIS of Joint Video Spec. (H.264 | ISO/IEC 14496-10 AVC) JVT of MPEG and VCEG, Doc. JVT-G050r1, May 2003.

[7] Joint Video Team (JVT) Reference Software. ver.10.2, http://iphome.hhi.de/suehring/tml/download/

저 자 소 개

임 상 희(학생회원)

2006년 중앙대학교 전자전기 공학부 학사졸업
2007년 현재 중앙대학교 첨단영상 대학원 영상공학과 석사과정
<주관심분야: 영상압축, 신호처리>



백 준 기(평생회원)

1984년 서울대학교 제어계측 공학과 학사졸업
1987년 노스웨스턴대학교 전기 및 컴퓨터 공학과 석사졸업
1990년 노스웨스턴대학교 전기 및 컴퓨터 공학과 박사졸업
2007년 현재 중앙대학교 첨단영상대학원 영상공학과 교수
<주관심분야: 영상복원, 신호처리, 반도체>

이 성 원(평생회원)

1988년 서울대학교 제어계측 공학과 학사졸업
1990년 서울대학교 제어계측 공학과 석사졸업
2003년 University of Southern California 전기공학과 박사졸업
2007년 현재 광운대학교 전자정보공과대학 컴퓨터공학과 교수
<주관심분야: 미디어프로세서 및 SOC설계, 멀티미디어 신호처리, Power-Aware Computing>