프레임율 감소를 위한 효율적인 MPEG 비디오 트랜스코딩 기법

*박경준, **양시영, **정제창
*한양대학교 정보디스플레이공학과 **한양대학교 전자통신전파공학과
joony79@ece.hanyang.ac.kr

An Efficient MPEG Video Transcoding Technique for Frame Rate Reduction

*Kyoungjoon Park , **Siyoung Yang, and **Jechang Jeong
*Information Display Engineering, Hanyang University
*Dept. of Electrical and Computer Engineering, Hanyang University

요약

다양한 처리능력을 가진 단말기들은 복잡한 네트워크 환경의 호환성을 제공하기 위해서, 전송 네트워크 채널이 허용하는 범위내로 부호화된 비디오의 비트율을 적응적으로 맞춰 주어야 한다. 트랜스코더는 특정 비트율로 부호화 되어 있는 비디오를 원하는 비트율로 다시 변환하기 위해서 복호화한 후 다시 부호화의 과정을 거쳐야 하기 때문에 이에 따른 계산량의 증가와 더불어 전송시간에 문제가 발생한다. 이를 해결하기 위한 한 가지 방법으로 제안된 것이 프레임 건너뜀 기법, 즉 시간적 해상도 변환 트랜스코딩이다. 비디오를 부호화하는 과정에서 계산량을 가장 많이 차지하는 움직임 추정과정의 계산량을 줄임으로써 트랜스코딩을 수행하는데 소모되는 시간과 노력을 크게 줄이고, 건너뛰지 않고 남아있는 프레임에 더 많은 비트를 할당하여 요구되는 화질을 유지할 수 있다. 본 논문에서 움직임 벡터의 방향성을 고려하여 제안한 기법인 E-FDVS (Efficient-Forward Dominant Vector Selection)는 움직임 벡터의 방향성을 고려하여 매크로블록과 움직임 벡터의 차이를 보상하여 움직임 벡터를 재추정한다. 실험에서는 MPEG-2 영상에 대해서 제안한 방법을 적용하여 기존의 방법들에 비해서 성능이 우수함을 보인다.

1. 서론

방송의 디지털화는 고화질 방송을 제공하여 방송 서비스의 품질을 한 단계 높이는 한편으로, 기존의 아날로그 방송과는 달리 이동 중에도 화면이 흔들리지 않는 방송을 가능하게 하였다. 디지털방송 기술에 기반을 두어 새로이 출현한 DMB (Digital Multimedia Broadcasting) 서비스는 최대 7인치 화면에서 이동 중 언제 어디서나 CD급 고품질의 라디오, TV 동영상 및 문자방송 수신이 가능한 서비스이다. 또한 디지털 방송을 보여주는 디지털 TV는 디지털 미디어 센터의 의미를 가지게 됨에 따라 다양한 포맷의 콘텐츠를 해석하고 이를 디스플레이 하거나 다른 기기들로 전송해줄 필요가 생긴다. 이에 따라 다양한 포맷의 미디어를 복호화하고 포맷간의 상호변환과 화질을 최대한 유지하면서 비트스트림 상에서 고속으로 변환하기 위한 트랜스코딩 기술이 필요하다. 트랜스코딩은 비트율 변환, 해상도 변환, 그리고 신텍스(syntax) 변환 등으로 분류 할 수 있다 [1].

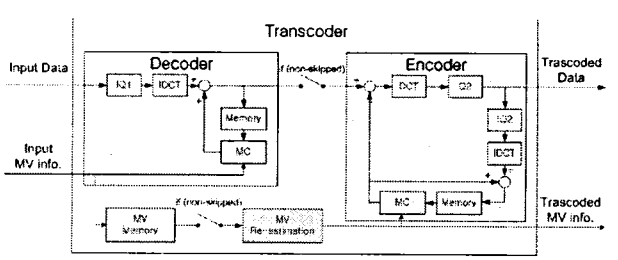

그림 1. 시간적 해상도 변환 트랜스코더 구조

※ 본 논문은 정보통신부의 출연금으로 수행한 IT SoC 핵심설계인력양성사업의 수행결과입니다.

일반적인 시간적 해상도 변환 트랜스코더 구조는 그림 1과 같다. 특정 비트율로 부호화 되어 있는 비디오를 원하는 비트율로 다시 변환하기 위해서는 복호화한 후 다시 부호화의 과정을 거쳐야 하기 때문에 이에 따른 계산량의 증가와 더불어 전송시간에 문제가 발생하게 된다. 이를 해결하기 위한 방법으로 프레임율을 변환시키는 프레임 건너뜀 (frame-skipping) 기법, 즉, 시간적 트랜스코딩이 있다. 이는 비디오를 부호화하는 과정에서 계산량을 가장 많이 차지하는 움직임 추정과정의 계산량을 줄임으로써 트랜스코딩을 수행하는 시간과 노력을 크게 줄이고, 건너뛰지 않고 남아있는 프레임에 더 많은 비트를 할당하여 요구되는 화질을 유지할 수 있다 [2]. 시간적 트랜스코딩에서 많은 시간을 할애해야 하는 곳은 움직임 벡터를 재추정 (re-estimation)하는 과정으로, 계산량을 줄이기 위해서 입력 비트스트림으로부터 추출해 낸 기저 움직임 벡터를 재사용한다.

트랜스코더는 복호화부터 부호화 순으로 수행되며, 복호기는 전탐색 움직임 추정을 거친 움직임 벡터가 비트스트림으로부터 입력되며, 부호기는 입력 비트스트림을 원하는 비트율의 출력 비트스트림으로 재부호화 한다. 트랜스코더

에서의 전역탐색 움직임 추정은 높은 계산 복잡도를 갖는다. 이 복잡도를 감소시키기 위해서 움직임 벡터를 재사용하는 움직임 벡터 추정 방법을 사용한다 [3].

트랜스코딩에서 입력 프레임들의 프레임율이 감소될 때 출력 비트스트림들은 새로운 움직임 벡터로 구성된다. 대표적인 기법인 쌍선형 보간법(Bilinear Interpolation) [4]과 FDVS (Forward Dominant Vector Selection) 기법 [5]은 네 개의 인접한 움직임 벡터들로부터 한 개의 움직임 벡터를 구성한다. 그러나 이 기법들은 움직임 벡터의 오추정 또는 전이가 발생하게 된다. 이를 피하기 위해 본 논문에서는 움직임 벡터의 방향성을 고려한 움직임 벡터 합성 기법을 제안한다. 또한 움직임 합성 기법을 통해 얻은 움직임 벡터는 최적의 움직임 벡터가 아니므로, 보다 더 양호한 화질을 얻으면서 비트율을 줄이기 위해 움직임 벡터 정제(refinement) 과정을 수행하게 된다 [6].

본 논문의 2장에서는 시간적 트랜스코딩 기법들에 대해서 살펴보고, 3장에서는 움직임 재추정을 위한 움직임 합성 기법을 제안한다. 4장에서는 실험결과를 제시하여 기존의 기법들과의 차이를 비교하고, 5장에서 결론을 맺는다.

2. 시간적 트랜스코딩

가. 프레임율 변환

기존의 현존하는 비디오 압축 표준들은 대부분 매크로블록 단위의 BMA (Block Matching Algorithm)을 기반으로 움직임 추정을 수행한다. 그림 2는 두 프레임 건너뛰었을 때의 역방향 움직임 벡터 추정 기법이 있다.

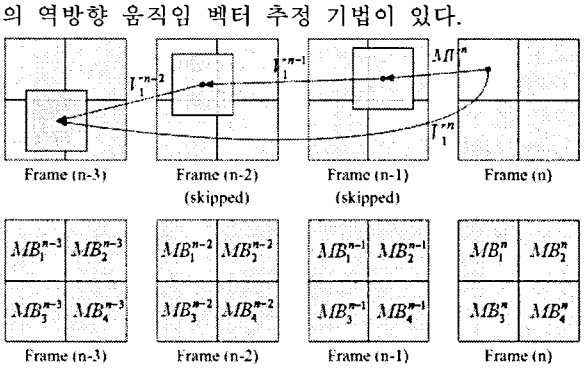

그림 2. 역방향 움직임 벡터 추정 기법

그림 2에서 MV_1^n은 프레임 (n)의 첫 번째 매크로블록 MB_1^n의 실제 움직임 벡터를 말하고, V_1^{n-1}은 프레임 (n-1)에서의 가상의 움직임 벡터이고, V_1^{n-2}는 프레임 (n-2)에서의 가상의 움직임 벡터를 말한다. MB_1^n의 움직임 벡터 V_1^n은 식 (1)과 같이 구할 수 있다.

$$V_x^n = \sum_{d=1}^{N} V_x^{n-d} + MV_x^n$$
$$\text{where, } N: \text{number of skipped frame} \quad (1)$$
$$n: \text{current frame number}$$
$$x: \text{macroblock number in a frame}$$

움직임 추정을 거치지 않고 건너뛴 프레임의 움직임 벡터를 찾는 방법은 벡터 MVx^n와 Vx^{n-d}를 합하는 방법을 생각할 수 있으나, 실제로 벡터 Vx^{n-d}는 입력 비트스트림에 존재하지 않으므로 구할 수 없다.

나. 쌍선형 보간법

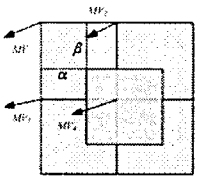

그림 3. 쌍선형 보간법

입력 비트스트림에 있는 움직임 벡터를 활용하기 위해서 제안된 쌍선형 보간법은 그림 3과 같이 움직임 벡터가 가리키는 매크로블록과 겹치는 참조 프레임의 이웃 매크로블록의 움직임 벡터에 가중치를 두어 식 (2)와 같이 계산하는 방법이다.

$$MV_T = (1-\alpha)(1-\beta)MV_1 + (\alpha)(1-\beta)MV_2 + (1-\alpha)(\beta)MV_3 + (\alpha)(\beta)MV_4 \quad (2)$$

그러나 쌍선형 보간법은 건너뛴 프레임의 매크로블록만큼의 계산량을 필요로 하며, 하나의 움직임 벡터가 실제 움직임 벡터와 다른 방향으로 발산하면, 나머지 움직임 벡터가 정확한 움직임 벡터와 유사하더라도 잘못된 값을 갖는다는 단점이 있다.

다. FDVS 기법

쌍선형 보간법은 움직임 벡터를 가중치를 부여해 합하게 되어서 잘못된 값을 가질 확률이 있으므로 매크로블록의 참조블록이 겹치는 블록들 중에서 가장 대표되는 블록을 선택하는 기법이 소개 되었다. 그림 4는 두 프레임이 건너뛰어졌을 때의 FDVS 기법을 나타낸다.

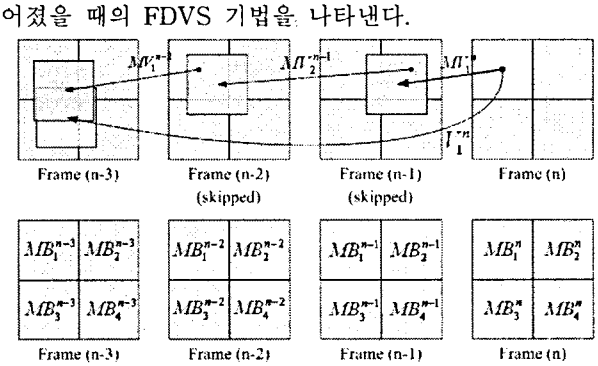

그림 4. 기존의 FDVS 기법

FDVS 기법은 그림 4와 같이 매크로블록의 참조 블록이 가장 많이 겹쳐지는 매크로블록의 움직임 벡터를 그 이전 프레임으로의 움직임 벡터로 사용한다. 지배적인 매크로블록은 참조블록이 가장 많이 겹쳐지는 블록이고, 지배적인 움직임 벡터는 지배적인 매크로블록의 움직임 벡터이다. 그림 4의 경우 프레임 (n-1)에서의 지배적인 매크로블록은 MB_2^{n-1}이며, 지배적인 움직임 벡터는 MV_2^{n-1}이다. 그림 4와 같이 두 프레임 건너뛰었을 때의 움직임 벡터는 식 (3)과 같이 구할 수 있다.

$$V_1^n = MV_1^{n-2} + MV_2^{n-1} + MV_1^n \quad (3)$$

3. 제안하는 기법

FDVS 기법은 많은 계산량과 메모리를 요구하는 선형 보간법보다 우수한 결과를 얻을 수 있으나 프레임 건너뜀이 많아질수록 움직임 벡터의 오추정이 전파된다. 현재의 프레임 (n)에서 움직임 벡터가 매크로블록 경계에 위치해 있으면, 더 이상의 움직임 합성 과정이 필요 없게 되나 대부분의 움직임 벡터는 이전 참조 프레임의 주변 2~4개의 매크로블록에 겹쳐지게 된다. 주변 매크로블록 중에서 가장 지배적인 매크로블록의 현재 프레임의 움직임 벡터로 결정되면, 새로운 움직임 벡터와 원래의 움직임 벡터간의 차이가 발생하게 되고, 프레임 건너뜀 수가 많아질수록 움직임 벡터가 오추정될 확률이 높아진다. 오추정이 누적되면 재추정된 움직임 벡터는 실제의 원영상과 전혀 다른 매크로블록을 현재 매크로블록의 움직임 벡터로 오인할 수 있다.

따라서 본 연구에서는 원래 움직임 벡터와 새로 합성한 움직임 벡터의 차이를 이용하여 지배적인 매크로블록의 움직임 벡터를 찾고, 이 움직임 벡터를 합성하는 E-FDVS (Efficient-FDVS) 기법을 제안한다. E-FDVS 기법은 그림 5와 같고 단계별 진행은 다음과 같다 [7].

초기화) 프레임 (n)에서의 cv^n와 최종 움직임 벡터 lmv,는 식 (4)와 같이 초기화 하고, k는 n으로 설정한다. 식 (4)에서 n 은 현재 프레임 번호이다.

$$cv^n = (cv^n[0], cv^n[1]) = (0,0)$$
$$lmv = (lmv[0], lmv[1]) = (0,0) \quad (4)$$

단계 1) mv^k 는 프레임 (k)의 지배적인(dominant) 매크로 블록의 움직임 벡터이다. lmv는 식 (5)와 같이 mv^k에 더해져서 갱신된다. 식 (5)에서 k는 프레임 번호이고, x는 프레임 내에서의 매크로블록 번호이다.

$$lmv = lmv + mv_x^k \quad (5)$$

단계 2) 프레임 (k)에서 지배적인 매크로블록의 움직임 벡터 fv^k는 식 (6)과 같이 구해진다. 그리고 나서 프레임 (k-1)에서의 다음 지배적인 매크로블록은 fv^k로 선택된다.

$$fv^k = cv^k + mv_x^k \quad (6)$$

단계 3) 움직임 보상 벡터는 프레임 (k)에서의 지배적인 움직임 벡터와 프레임 (k-1)에서의 지배적인 매크로블록과의 차이로 식 (7)과 같이 구해진다.

$$cv^{k-1}[i] = mv_x^k[i]$$
$$while(|cv^{k-1}[i]| > 8)\{$$
$$cv^{k-1}[i] = \begin{cases} cv^{k-1}[i]-16 & if\ cv^{k-1}[i] > 8 \\ cv^{k-1}[i]+16 & if\ cv^{k-1}[i] < -8 \end{cases} \quad (7)$$
$$\}$$
$$where,\ k: frame\ number$$
$$i: vector\ index,\ i = 0, 1$$
$$x: macroblock\ number\ in\ a\ frame$$

단계 4) 더 이상 건너뛰는 프레임이 없을 때까지, 단계 2와 단계 3을 반복한다.

단계 5) E-FDVS는 종료되고, 합성 벡터는 최종 움직임 벡터가 된다.

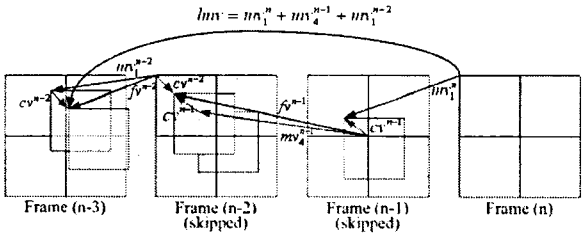

그림 5. 제안하는 E-FDVS 기법

4. 실험결과

제안한 알고리즘을 검증하기 위하여 실험은 MPEG-2 TM5 환경에서 IPPP 구조를 사용하였다. 실험은 4:2:0 형식을 갖는 CIF 영상 Bus, Coastguard, Football, Foreman, Mother&Daughter, Stefan에 대해서 50 프레임씩 수행하였다. 움직임 벡터 합성 기법의 우수성을 보이기 위해서 전탐색(full search)뿐만 아니라, 고속 탐색 알고리듬인 TSS (Three Step Search), NTSS (New TSS), FSS (Four Step Search), CDS (Cross Diamond Search)와 비교하였고, 탐색범위는 ±15로 정수 픽셀 탐색을 하였다.

표 1은 프레임율을 30 fps에서 10 fps로 프레임율을 감소시면서 비트율을 1Mbps에서 300kbps로 낮추었을 때, 다양한 움직임 합성 기법 비교의 성능 비교하였다. PSNR은 영상의 객관적 화질을 나타내고, SP (Search Point)는 BMA 수행시 블록 매칭하는 탐색점의 개수를 나타낸다. 움직임 벡터 합성 기법은 입력 비트스트림으로부터 움직임 벡터를 합성하여 구하므로 블록매칭을 필요로 하지 않는다. 탐색점의 개수가 0임에도 불구하고 쌍선형 보간법, FDVS 기법, E-FDVS 기법은 기존의 고속 탐색 알고리듬보다 화질이 우수하며, 움직임 벡터 합성기법에서는 E-FDVS 기법의 성능이 가장 우수하다.

움직임 합성 과정을 거쳐 얻어진 움직임 벡터들은 원래의 움직임 벡터를 근사화한 값이어서 최적의 움직임 벡터라 할 수 없으므로, 더 나은 성능을 얻기 위해 움직임 벡터 정제 과정이 필요하다. 표 2는 프레임율을 30 fps에서 10 fps로 프레임율을 감소시면서 비트율을 1Mbps에서 300kbps로 낮추었을 때, 다양한 움직임 합성 기법과 움직임 정제 기법의 조합에 따른 성능 비교하였다. 움직임 정제 기법으로는 탐색범위 ±2만큼 갖는 FSS (Full Scale Search), HAVS (Horizontal and vertical search) [6], VSS (Variable Step-size Search) [7], DOS (Direction Oriented Search) [8]을 사용하였고, E-FDVS와 DOS 조합이 성능이 가장 우수함을 볼 수 있다.

5. 결론

유무선 네트워크 환경에서 제공되는 대부분의 비디오 서비스와 멀티미디어 응용서비스를 사용자가 이용할 때 다양한 채널의 대역폭에 따라 적절한 비트율로 변환해 주어야만 정해진 채널상황 속에서 최대한의 비디오 품질을 보장할 수 있다. 본 논문에서는 MPEG에서 프레임율을 감소해야 할 때, 움직임 벡터의 방향성을 이용한 움직임 재추정 기법을 제안하였다.

표 1. 프레임율을 30 fps에서 10 fps로 프레임율을 감소시면서 비트율을 1Mbps에서 300kbps로 낮추었을 때, 다양한 움직임 합성 기법 비교의 성능 비교 [PSNR : dB, SP : Search Point]

Test Sequence	Measure	Full Search	Fast Motion Estimation				Motion Composition		
			TSS	NTSS	FSS	CDS	Bilinear	FDVS	E-FDVS
Bus	PSNR	19.184	17.9319	17.7446	17.7269	17.1608	17.2841	18.5988	**19.0466**
	SP	380556	13068	12589.9	12958.4	9320.63	0	0	**0**
Coast guard	PSNR	25.3165	24.938	24.4927	24.9769	24.11	21.4752	24.9169	**24.9162**
	SP	380556	13068	11305.5	11509.4	7897	0	0	**0**
Football	PSNR	19.7362	19.1896	19.072	18.9733	18.4948	16.7896	18.6199	**18.7787**
	SP	380556	13068	10844.1	12075.1	8558	0	0	**0**
Foreman	PSNR	28.8727	27.6926	27.6369	28.0122	28.0484	21.1123	28.1661	**28.2824**
	SP	380556	13068	10637.9	12363.3	8651	0	0	**0**
Mother & daughter	PSNR	37.4807	37.1009	37.1317	37.2688	37.2361	23.1432	36.3513	**36.3518**
	SP	380556	13068	7650	10343.7	4410.19	0	0	**0**
Stefan	PSNR	21.1615	20.2557	20.2381	20.3393	20.0194	18.4631	20.8984	**21.0451**
	SP	380556	13068	10326.4	11792.7	7488.06	0	0	**0**

표 2. 프레임율을 30 fps에서 10 fps로 프레임율을 감소시면서 비트율을 1Mbps에서 300kbps로 낮추었을 때, 다양한 움직임 합성 기법과 움직임 정제 기법의 조합에 따른 성능 비교 [PSNR : dB, SP : Search Point]

Test Sequence	Measure	Full Search	FDVS					E-FDVS				
			None	FSS	HAVS	VSS	DOS	None	FSS	HAVS	VSS	DOS
Bus	PSNR	19.184	18.5988	19.1691	19.1043	19.2329	19.2234	19.0466	19.6023	19.5339	19.5821	**19.6104**
	SP	380556	0	5770.75	1681.81	2307.38	1882.75	0	5734.25	1658.06	2279.38	**1861.88**
Coast guard	PSNR	25.3165	24.9169	25.1834	25.156	25.049	25.1408	24.9162	25.183	25.1557	25.0485	**25.1409**
	SP	380556	0	9134.56	2315.13	3319.75	2603.94	0	9134.88	2315.13	3319.81	**2604**
Football	PSNR	19.7362	18.6199	18.9609	18.872	19.1312	19.0515	18.7787	19.1191	19.0263	19.2814	**19.1883**
	SP	380556	0	7495.81	2009.19	2810.63	2275.75	0	7517.75	2014.88	2809.63	**2272.31**
Foreman	PSNR	28.8727	28.1661	28.6727	28.5373	28.5655	28.6077	28.2824	28.7867	28.658	28.6185	**28.6742**
	SP	380556	0	8894.63	2275.38	3235.38	2564.44	0	8893.19	2273	3231.5	**2560.81**
Mother & daughter	PSNR	37.4807	36.3513	37.4351	37.312	37.0608	37.3644	36.3518	37.4351	37.312	37.0614	**37.3644**
	SP	380556	0	9139.75	2332.19	3335	2621.38	0	9139.75	2332.19	3335	**2621.38**
Stefan	PSNR	21.1615	20.8984	21.4931	21.4264	21.3706	21.4749	21.0451	21.6303	21.5638	21.4988	**21.5968**
	SP	380556	0	7458	1980.56	2833.44	2262.25	0	7442.88	1978.69	2822.5	**2253.38**

참고문헌

[1] A. Vetro, C. Christopoulos, and Sun Huifang, "Video transcoding architectures and techniques: an overview," IEEE Signal Processing Magazine, Vol.20, Issue 2, pp.18-29, March 2003.

[2] Kai-Tat Fung, Wan-Chi Siu, and Yui-Lam Chan, "A dynamic frame-skipping video combiner for multipoint video conferencing," IEEE International Symposium on Circuits and Systems 2002, Vol.3, pp.389-392, 26-29 May 2002.

[3] Wei Jen Lee and Wen Jen Ho, "Adaptive frame-skipping for video transcoding," International Conference on Image Processing 2003, Vol.1, pp.165-168, 14-17 Sep. 2003.

[4] Jenq-Neng Hwang, Tzong-Der Wu, and Chia-Wen Lin, "Dynamic frame-skipping in video transcoding," 1998 IEEE Second Workshop on Multimedia Signal Processing, pp.616-621, 7-9 Dec. 1998.

[5] Jeongnam Youn and Ming-Ting Sun, "fast motion vector composition method for temporal transcoding," IEEE International Symposium on Circuits and Systems 1999, Vol.4, pp.243-246, 30 May-2 June 1999.

[6] Jeongnam Youn, Ming-Ting Sun, and Chia-Wen Lin, "Motion vector refinement for high-performance transcoding," IEEE Transactions on Multimedia, Vol.1, Issue1, pp.30-40, March 1999.

[7] Siyoung Yang, Donghyung Kim, Yeonggyun Jeon, and Jechang Jeong, "An Efficient Motion Re-estimation Algorithm for Frame-skipping Video Transcoding", IEEE International Conference on Image Processing (ICIP 2005) Proceeding, vol.3 pp.668-671, Genova, Italy, Sept. 2005.

[8] Mei-Juan Chen, Ming-Chung Chu, and Chih-Wei Pan, "Efficient motion-estimation algorithm for reduced frame-rate video transcoder," IEEE Transactions on Circuits and Systems for Video Technology, Vol.12, Issue 4, pp.269-275, April 2002.