프레임 율 향상을 위한 계층적 다방향 움직임 추정 알고리즘

유송현, 박범준, 정제창
한양대학교 전자컴퓨터통신공학과
3069song@naver.com, kkbbbj1@naver.com, jjeong@hanyang.ac.kr

Hierarchical Multidirectional Motion Estimation Algorithm for Frame Rate Up-Conversion

Songhyun Yu, Bumjun Park, Jechang Jeong
Department of Electronics and Computer Engineering, Hanyang University

요 약

본 논문에서 프레임 율 향상을 위한 새로운 움직임 추정 알고리즘에 대해 제안한다. 계산량을 줄이고 다해상도의 영상을 이용하기 위하여 원본 프레임들을 계층적 구조로 형성하고, 최상위 계층에서 단방향 움직임 추정을 수행한다. 최상위 계층은 낮은 해상도 때문에 움직임 벡터의 정확도가 낮아지므로, 정확도를 향상시키기 위해 각각의 블록은 5 개의 움직임 벡터 후보들을 가진다. 이 후보들은 아래 계층들에서 수정되며, 움직임 추정이 완료되면 최하위 계층의 움직임 벡터들은 SAD (sum of absolute difference) 값을 이용해서 최종적으로 수정된다. 이렇게 구해진 단방향 움직임 벡터들은 양방향 움직임 벡터로 변환되고 양방향 보간법을 사용하여 보간 프레임을 생성한다. 결과적으로, 제안하는 알고리즘은 기존 알고리즘들에 비해 낮은 계산량을 나타내면서 PSNR (peak signal-to-noise ratio) 수치에서 최대 1.3 dB 의 향상을 나타냈고, 주관적으로도 더 선명한 결과를 보여주었다.

1. 서론

프레임 율 향상 (FRUC: frame rate up-conversion)은 원래 존재하는 프레임들 사이에 새 프레임을 삽입하여 프레임을 보간 하는 기술이다. 초기에는 프레임 반복 또는 두 프레임을 단순 평균하는 방법을 사용하여 프레임을 보간 하였다. 그러나 이로 인해 화질 저하가 발생한다. 최근에는 움직임 보상 FRUC (motion compensated FRUC) 알고리즘이 개발되었다 [1-3]. 이 알고리즘은 일반적으로 움직임 추정 (motion estimation), 움직임 벡터 수정 (motion vector smoothing) 및 움직임 보상 보간 (motion compensated interpolation)의 세 단계로 이루어진다.

움직임 추정은 원본 프레임을 이용하여 각 물체의 움직임을 찾는 과정이며 이 단계에서 움직임 벡터를 얻는다. 움직임이 예측되는 방향에 따라 움직임 추정은 단방향 움직임 추정, 양방향 움직임 추정의 두 가지로 나눌 수 있다. 단방향 움직임 추정은 원본 프레임을 기준으로 한 쪽 방향으로 움직임을 추정하며, 보간 영상에서 홀 (hole)과 중첩 영역 (overlapped region)이 발생한다. 양방향 움직임 추정은 보간 된 프레임의 시점에서 움직임을 양방향으로 예측하므로 홀과 중첩 영역이 발생하지 않는다. 그러나 현재 참조 블록에 대한 정보가 없으므로 잘못된 움직임 벡터를 찾을 가능성이 더 크다는 단점을 가지고 있다.

기본적으로 움직임 추정은 높은 연산량을 가지고 있기 때문에 고해상도 비디오의 실시간 처리를 위해서는 복잡도를 줄이는 것이 필수적이다. 낮은 복잡도를 가지는 FRUC 알고리즘들이 여러 연구들에서 제안되었다. [13]은 움직임 추정 단계에서 가우시안 (Gaussian) 피라미드 구조와 라플라시안 (Laplacian) 피라미드 구조를 사용하였고, [15]는 원본 영상을 서브샘플링 하여 움직임 추정을 수행했다. 그러나 이러한 알고리즘들은 원래 이미지의 해상도 감소로 인해 움직임 벡터의 정확도 또한 저하되는 문제점이 있다. 본 논문에서는 복잡도를 줄이고 움직임 벡터의 정확도를 유지하기위한 다방향 움직임 추정 알고리즘을 제안한다.

움직임 벡터 수정을 위한 여러 가지 연구들이 진행되었는데, [4]는 인접 움직임 벡터들의 평균과 분산을 사용하여 각 움직임 벡터들의 신뢰성을 판단하고, 현재 움직임 벡터가 신뢰성이 좋지 않은 경우 이들의 가중치 합으로 현재 벡터를 대체한다. [6]에서는 현재 블록의 움직임 벡터 정보를 사용해서 잘못된 움직임 벡터를 수정한다. 그러나 현재 블록이 객체의 경계에 있거나 현재 움직임 벡터가 올바르지 않은 경우 이러한 알고리즘들을 사용할 경우 오류가 발생할 수 있다. 본 논문에서는 이러한 한계를 극복하기 위해 적응적 전역 움직임 추정 알고리즘을 제안한다.

본 논문의 나머지 부분은 다음 순서로 구성되어 있다. 2 장에서는 제안한 알고리즘을 설명하고, 3 장은 다른 알고리즘과 비교한 실험 결과를 설명한다. 마지막으로 4 장에서 결론을 짓는다.

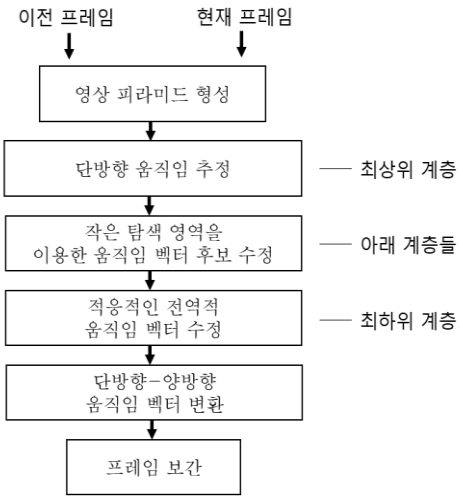

그림 1. 제안하는 알고리즘 흐름도

2. 제안하는 알고리즘

그림 1 은 제안하는 알고리즘의 전체 흐름도를 보여준다. 첫 번째로 두 개의 원본 프레임들로 영상 피라미드 구조를 형성한다[7]. 최상위 계층에서 단방향 움직임 추정을 수행하며, 여기서 각 블록마다 5 개의 후보 위치를 지정한다. 하위 계층들에서 각 후보 위치들은 주변의 작은 탐색 영역 (search range) 을 사용한 움직임 추정에 의해 수정된다. 최하위 단계에서 5 개의 후보군 중 오차의 절대값의 차 (SAD)가 가장 작은 후보가 최종적으로 선택되고, 이렇게 한 프레임의 모든 블록의 움직임 벡터들이 결정되면 이들은 SAD 값에 의해 전역적으로 수정된다. 수정된 움직임 벡터들은 움직임 벡터 변환 알고리즘[5]을 사용하여 양방향 움직임 벡터들로 변환된다. 결과적으로 양방향 보간을 사용하여 새로운 프레임이 생성된다.

2.1 다방향 움직임 추정

움직임 추정을 수행하기 전에, 그림 2 처럼 각 프레임들은 피라미드 구조를 형성한다. 최상위 계층에서 움직임 추정 시 탐색 영역을 그림 3 처럼 4 개의 영역으로 분할하고, 각 영역마다 최소의 SAD 값을 가지는 위치를 후보 위치로 지정한다. SAD 는 다음 식과 같이 표현된다.

$$E(B,v_h)=\sum_{x\in B}|f_{n-1}(x)-f_n(x+v_h)|$$
$$v_m=\arg\min_{v_h\in SR_m}\{E(B,v_h)\},\quad m=1,2,3,4 \tag{1}$$

여기서 $E(B,v_h)$ 는 블록 B 의 움직임 벡터 후보 v_h 위치에서의 단방향 SAD 값을 나타내고, x 는 화소 위치이다. SR_m 은 그림 3 에서 표현된 4 개의 탐색 영역을 나타내고, v_m 은 각 영역에서의 움직임 벡터 후보들을 의미한다.

최상위 계층에서의 움직임 추정이 끝나면, 아래 계층에서

그림 2. 영상 피라미드 구조

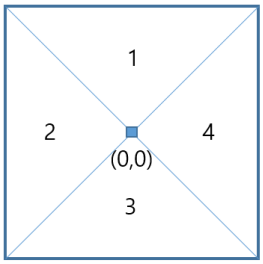

그림 3. 탐색 범위에서의 4 개의 영역

각 움직임 벡터 후보들은 작은 탐색 영역에서의 움직임 추정을 통해서 수정된다. 이 때 블록 별로 탐색 영역의 크기는 해당 블록의 SAD 값에 의해서 다음과 같이 결정된다.

$$SR_{small}(E')=\alpha(E'-255)^2+\beta \tag{2}$$

여기서 E' 은 SAD 값을 블록의 화소 개수로 나누어 준 정규화된 SAD 값을 의미한다. β 는 최대 탐색 영역의 크기로, 50 으로 설정하였고 α 는 가중치 계수로 0.0077 을 사용했다. 본 논문은 8 비트로 표현된 영상을 기준으로 하므로, 8 비트 영상의 최대 화소 값인 255 가 식(2)에서 사용되었다.

최하위 계층에서는 각 블록들이 가진 5 개의 후보 움직임 벡터들 중에서 SAD 가 가장 작은 후보를 해당 블록의 최종 움직임 벡터로 결정한다.

2.2 적응적인 전역적 움직임 벡터 수정

한 프레임에서의 움직임 추정 과정이 종료되면 움직임 추정 과정에서 발생한 오류를 줄이기 위해 움직임 벡터 수정을 실행한다. 제안하는 알고리즘은 블록의 SAD 값을 이용해서 해당 블록의 신뢰성을 판단한다. 이 과정은 다음 식과 같이 나타난다.

$$R(B)=1 \qquad if\ SAD(B)>Th$$
$$R(B)=0 \qquad otherwise \tag{3}$$

여기서 $R(B)$ 는 블록 B 의 신뢰성을 나타내며 현재 블록의 움직임 벡터에 오류가 있을 경우 1 의 값을 가지고, 그렇지 않을 경우 0 의 값을 가진다. Th 는 임계값으로 반복 알고리즘[8]을 사용하여 결정되며 그 과정은 그림 4 와 같다. 현재 블록의 움직임 벡터가

1. 초기 임계값 설정: T
2. SAD의 히스토그램을 구하고, T를 기준으로 두 개의 구간으로 나눔 $(Z_1 > T, Z_2 \leq T)$
3. 각 영역에서의 평균값을 구함: w_1, w_2 (각각 Z_1, Z_2 영역에서)
4. 새로운 임계값을 구함: $T_{new} = \frac{(w_1+w_2)}{2}$
5. 조건($\Delta T = |T_{new} - T| < \gamma$)이 만족할 때 까지 2-4단계를 반복

그림 4. 반복적 임계치 결정 알고리즘

오류가 있다고 판별되면, 주변 움직임 벡터들의 가중 합으로 움직임 벡터를 수정한다[16].

3. 실험 결과

실험 결과를 비교하기 위해 객관적인 평가와 주관적인 평가를 모두 시행하였다. 실험 영상으로는 HEVC (high efficiency video coding) 표준 실험 영상들을 사용하여 고해상도 영상까지 실험에 포함하였다. 블록 크기는 16 × 16 화소를 사용하였고, 최상위 계층에서의 탐색 영역은 ±12 로 설정하였다.

객관적인 평가를 위해서 PSNR 과 계산 복잡도를 기존의 알고리즘들과 비교하였다. 표 1 은 기존 알고리즘들 [15, 4, 10]과 비교한 PSNR 결과를 보여준다. 모든 실험 영상에서 제안하는 알고리즘이 가장 높은 PSNR 값을 나타냈으며 각 알고리즘과 비교하면 [15]에 비해 최대 1.17 dB, [4]에 비해 최대 1.34 dB, [10]에 비해 최대 1.05 dB 의 향상을 보여주었다. 평균 값에서도 제안하는 알고리즘이 최대 수치를 기록했다.

표 2 는 제안하는 알고리즘과 기존 알고리즘들의 계산 복잡도를 나타낸다. 여기서 N 은 블록의 크기(16), R 은 탐색 영역의 크기(25), 그리고 R_s 는 작은 탐색 영역의 크기(2)를 나타낸다. 제안하는 알고리즘에서 R_s 는 블록의 SAD 값에 따라 적응적으로 결정되지만, 실제 알고리즘 실행 시 평균적인 값인 2 로 추정하여 계산하였다. 표 2 에서 상대 복잡도는 제안하는 알고리즘의 복잡도를 1 로 가정했을 때, 상대적인 복잡도를 의미한다. 제안하는 알고리즘이 기존의 알고리즘들에 비해서 적은 계산 복잡도를 가짐을 알 수 있다. [15]와 비교해서는 복잡도 측면에서 큰 차이를 보이지 않지만, 제안하는 알고리즘은 아래 계층들에서 움직임 벡터의 후보군을 수정할 때, 겹쳐지는 영역이 많이 발생하기 때문에 실제로 수행할 때의 복잡도는 계산 복잡도 1 보다 더 줄어들게 된다.

그림 5 는 *BasketballDrive* 영상에서의 실험 결과를 보여준다. 빨간색 네모로 표시된 영역을 보면, 제안하는 알고리즘이 기존 알고리즘들에 비해 주관적으로도 선명한 영상을 나타내는 것을 확인할 수 있다.

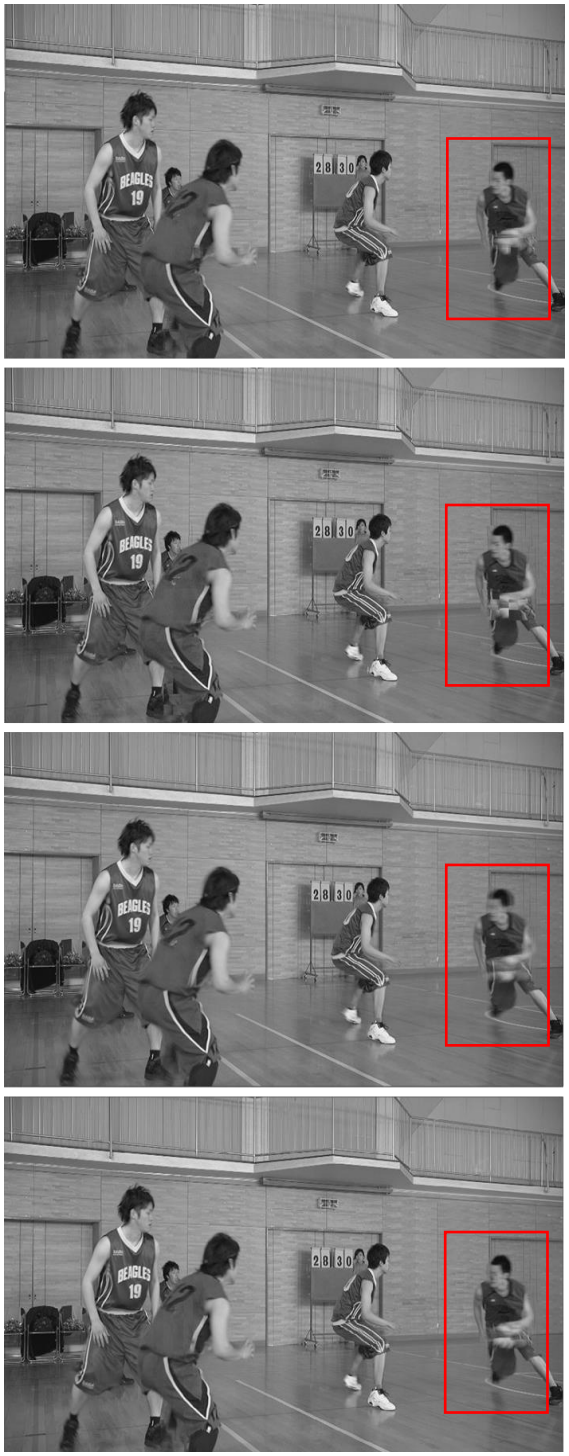

그림 5. 실험 결과 영상. (a) [15], (b) [4], (c) [10], (d) 제안하는 알고리즘

4. 결론

본 논문에서, 프레임 율 향상을 위한 움직임 추정과 움직임 벡터 수정 알고리즘을 제안하였다. 움직임 추정 과정에서, 낮은 복잡도를 가지면서 정확한 움직임 벡터를 찾기 위해 다방향 움직임 추정을 제안하였고, 움직임 벡터 수정 과정에서는 프레임의 성질에 따라 적응적으로 임계치를 결정하여 움직임 벡터의 신뢰성을 판단하여 수정하는 알고리즘을 제안하였다. 결과적으로, 제안하는 알고리즘은 객관적 평가에서 기존 알고리즘들에 비해 가장 낮은 복잡도를 가짐과 동시에 가장 높은 PSNR 값을 나타냈고, 주관적 평가에서 역시 선명한 결과 영상을 보여주었다.

표 1. PSNR 비교

	[15]	[4]	[10]	제안하는 알고리즘
PeopleOnStreet (2460×1600)	26.86	26.69	27.91	28.03
BasketballDrive (1920×1080)	30.30	30.39	30.46	30.84
ParkScene (1920×1080)	35.10	34.95	34.30	35.35
Johnny (1280×720)	41.39	41.40	41.52	41.62
BQMall (832×480)	33.01	32.53	32.94	33.01
평균	33.33	33.19	33.43	33.77

표 2. 계산 복잡도 비교

알고리즘	계산 복잡도	상대 복잡도
[15]	$(\frac{N}{2})^2(R+1)^2$	1.01
[4]	$2(1,5N)^2(2R+1)^2$ $-((1.5N)^2-1.5N)^2$	63.03
[10]	$2N^2(2R+1)^2$ $+2N^2(2R_S+1)^2$	31.48
제안하는 알고리즘	$(\frac{N}{4})^2(\frac{R}{2}+1)^2$ $+5(\frac{N}{2})^2(2R_S+1)^2$ $+5(N)^2(2R_S+1)^2$	1

감사의 글

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신 · 방송 연구개발사업의 일환으로 수행하였음.[R0601-16-1063, ICT 장비용 SW 플랫폼 구축]

참고 문헌

[1] C. Cafforio, F. Rocca, and S. Tubaro, "Motion compensated image interpolation," *IEEE Trans. Communication*, vol. 38, no. 2, pp. 215 -222, Feb. 1990.

[2] S. H. Lee, Y. C. Shin, S. Yang, H. H. Moon, and R.H. Park,"Adaptive motion-compensated interpolation rate-up-conversion," *IEEE Trans. Consumer Electronics*, vol. 48, no. 3, pp. 444-450, Aug. 2002.

[3] S. H. Lee, O. Kwon, and R. H. Park, "Weighted-adaptive motion compensated frame rate up-conversion," *IEEE Trans. Consumer Electronics*, vol. 49, no. 3, pp. 485-492, Aug. 2003.

[4] D. G. Yoo, S. J. Kang, and Y. H. Kim, "Direction-select motion estimation for motion-compensated frame rate up-conversion," *J. Display Technol.*, vol. 9, no. 10, pp. 840- 850, Oct. 2013.

[5] Y. Guo, L. Chen, Z. Gao, and X. Zhang, "Frame rate up-conversion using linear quadratic motion estimation and trilateral filtering motion smoothing," J. Display Technol., vol. 12, no. 1, pp. 89-98, Jan. 2016.

[6] D. Wang, L. Zhang, and A. Vincent, "Motion-compensated frame rate up-conversion—Part I: Fast multi frame motion estimation," *IEEE Trans. Broadcast*, vol. 56, no. 2, pp. 133- 141, Jun. 2010.

[7] P. J. Burt, and E. H. Adelson, "The laplacian pyramid as a compact image code," *IEEE Trans. Commun.*, vol. 31, no. 4, pp. 532-540, Apr. 1983.

[8] R. C. Gonzalez, and R. E. Woods, *Digital Image Processing*, Pearson Prentice Hall. 2010.

[9] C. Wang, L. Zhang, Y. He, and Y. P. Tan, "Frame rate up-conversion using trilateral filtering," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 6, pp. 886-893, Jun. 2010.

[10] S. J. Kang, S. Yoo, and Y. H. Kim, "Dual motion estimation for frame rate up-conversion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 12, pp. 1909- 1914, Dec. 2010.

[11] D. Wang, A. Vincent, P. Blanchfield, and R. Klepko, "Motion-compensated frame rate up-conversion — Part II: New algorithms for frame interpolation," *IEEE Trans. Broadcast.*, vol. 56, no. 2, pp. 142- 149, Jun. 2010.

[12] S. J. Kang, K. R. Cho, and Y. H. Kim, "Motion compensated frame rate up-conversion using extended bilateral motion estimation," *IEEE Trans. Consumer Electron.*, vol. 53, no. 4, pp. 1759- 1767, Nov. 2007.

[13] B. W. Jeon, G. I. Lee, S. H. Lee, and R. H. Park, "Coarse-to-fine frame interpolation for frame rate up-conversion using pyramid structure," *IEEE Trans. Consum. Electron.*, vol. 49, no. 3, pp. 499-508, 2003.

[14] M. T. Orchard and G. J. Sullivan, "Overlapped block motion compensation: An estimation-theoretic approach," *IEEE Trans. Image Process.*, vol. 3, no. 5, pp. 693-699, 1994.

[15] B. Choi, S. Lee, and S. Ko, "New frame rate up-conversion using bi-directional motion estimation," *IEEE Trans. Consumer Electronics*, vol. 46, no. 3, pp. 603-609, 2000.

[16] S. Yu, J. Ryu, and J. Jeong, "Frame rate up-conversion using hierarchical motion estimation and variance-based motion vector smoothing," *International journal of electrical, electronics and data communication*, vol. 4, no. 8, pp. 12-15, Aug. 2016.

[17] J. Ryu, S. Yu, and J. Jeong, "Feature-based robust 3-DRS motion estimation for high definition frame rate up-conversion," *International journal of electrical, electronics and data communication*, vol. 4, no. 8, pp. 7-12, Aug. 2016.