비디오 코딩을 위한 빠른 블록 모션 추정 방법

이연철*, 김은이, 김항준
경북대학교 컴퓨터공학과 인공지능연구실

A Fast Block Motion Estimation Algorithm for Video Coding

Yeon Chul Lee*, Eun Yi Kim, Hang Joon Kim
Dept. of Computer Engineering, Kyungpook National University
yclee@ailab.knu.ac.kr

요 약

본 논문에서는 비디오 코딩을 위한 빠른 움직임 추정(motion estimation) 방법을 제안한다. 계산량을 줄이기 위해서, 제안된 움직임 추정 알고리즘은 블록 당 탐색 점(searching point)의 수를 줄이는 대신에 프레임 당 탐색 블록(searching block)의 수를 줄임으로써 실행되어진다. 이를 위해서, 연속된 두 프레임간의 시간적인 상관관계(temporal correlation)를 통해 현재 프레임에 있는 모든 블록들을 움직임 블록(moving block)과 배경 블록(background block)으로 분류되어진다. 잘 알려진 비디오 영상들에게서 실험한 결과들을 통해 제안된 방법이 상당히 정확한 움직임 벡터(motion vector)들 뿐 만 아니라 계산적인 효율성을 향상할 수 있음을 볼 수 있다.

Abstract

This paper presents a new fast motion estimation algorithm for video coding. This method classifies blocks in a frame into moving blocks and background blocks, and then searches the best-matched blocks for only moving blocks. Experimental results show the effectiveness of the proposed method.

I. 서 론

움직임 추정은 연속된 두 프레임 사이에서 움직이는 물체의 변위(displacement)를 획득하는 것으로 정의되어진다. 이는 두 프레임 사이의 시간적인 중복성을 제거하는 능력 때문에 비디오 코딩에 중요한 역할을 담당한다 [1,2].

지금까지 연구된 많은 움직임 추정 알고리즘 가운데서 블록 정합 알고리즘(block matching algorithm: BMA)은 알고리즘이 간단하고, 정형화된 데이터 구조를 가지며, 빠른 움직임 추정을 위해 하드웨어 구현이 쉽다는 장점 때문에 국제 표준안 등에서 채택되어 가장 많이 사용되고 있다. 블록 정합 알고리즘에서 움직임 추정은 정형화된 크기의 블록 단위로 하며, 기존 프레임(reference frame) 내에 설정된 탐색 윈도우(search window)에서 현재 프레임(current frame)의 블록에 대해 정합 차이(matching difference)가 최소를 갖는 블록을 찾아 움직임 벡터를 정하는 알고리즘이다. 블록 정합 알고리즘 중에 가장 쉽고, 늘리 사용되는 전역 탐색 알고리즘(full search: FS)은 탐색 윈도우 내의 모든 후보 탐색 점에 대해 매칭 차이를 계산하기 때문에 많은 계산량이 요구되어 실시간 비디오 코딩에 적용되기 어려운 문제점이 있다. 이런 문제점을 개선하기 위해, 3 단계 탐색 알고리즘(three step search algorithm: TSS), 새 3단계 탐색 알고리즘(new three step search algorithm), 2차원 로그 탐색 알고리즘(2D-LOG search algorithm), 크로스 탐색 알고리즘(cross search algorithm)와 같은 고속 움직임 추정 알고리즘들이 제안되었다 [3-7]. 그러나, 이런 빠른 알고리즘들 또한 움직임이 없는 블록에 대해서도 모든 후보 탐색 점들을 탐색한다. 이는 불필요한 계산량을 요구하는 문제점을 가지고 있다.

이런 문제점을 해결하기 위해서, 본 논문에서는 빠른 블록 움직임 추정 방법을 제안한다. 일반적인 비디오 영상들은 시간적인 중복성이 상당히 높다. 이런 성질을 이

용하여, 현재 프레임에 있는 모든 블록들을 움직임 블록과 배경 블록의 두 그룹으로 나눈다. 이런 블록 그룹이 움직임 벡터를 구할 것인지 아닌지를 결정하게 된다. 즉, 배경 블록에 대해서 제로의 움직임 벡터를 할당하고, 움직임 블록에 대해서는 기존에 연구된 BMA을 사용해 움직임 벡터를 구한다. 이는 움직임이 없는 배경에 대한 움직임 벡터를 구하지 않기 때문에 불필요한 계산량을 제거함으로써 속도를 향상시킬 수 있다.

II. 제안된 알고리즘

제안된 움직임 추정 방법은 블록 당 탐색 점들의 수를 줄이는 대신에 프레임 당 탐색 블록들의 수를 줄임으로써 실행되어 진다. 제안된 방법에서 현재 프레임에 있는 모든 블록들은 두 개의 블록 타입으로 분류되어진다. 즉, 움직임 블록과 배경 블록이다. 이런 분류는 변화 발견 마스크(change detection mask: CDM)를 통해 이루어진다. 그 후에, 움직임 벡터들을 추정한다. 즉, 움직임 블록들에 대해서는 기존에 연구된 움직임 추정 방법을 사용하여 움직임 벡터를 추정하고, 배경 블록에 대해서는 제로의 움직임 벡터를 할당한다. 제안된 방법은 블록 분류(block classification) 모듈과 움직임 벡터 추정(motion vector estimation) 모듈로 구성되어 있다. 제안된 알고리즘의 흐름도는 그림 1에서 보여진다.

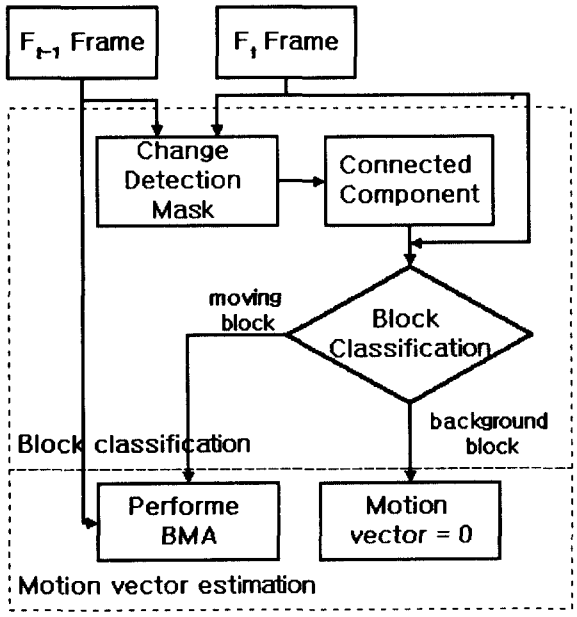


그림 1 제안된 알고리즘의 블록 다이어그램

2.1 블록 분류

블록들을 분류하기 위해서, 우린 CDM을 사용한다. CDM은 움직인 영역(changed area)과 움직이지 않은 배경 영역(unchanged area)으로 나누어진 이진화 영상이다 [8]. CDM을 얻는 가장 단순한 방법은 연속된 두 프레임의 밝기 차이를 얻고 이를 일정한 임계값(thresholding value)으로 임계치(threshold)하는 것이다. CDM은 식 (1)과 같이 정의된다.

$$CDM(i,j) = \begin{cases} 1, & if\ |D(i,j)| > t \\ 0, & otherwise \end{cases} \quad (1)$$

여기서, $D(i,j)$는 현재 프레임과 기준 프레임에서 픽셀 (i,j)의 밝기 차이 값이며, t는 임계값이다. 그림 2는 CDM을 생성하는 과정을 보여주고 있다. (가)와 (나)는 "*Hall-monitor*" 영상의 30번째와 31번째 프레임이고, (다)는 (가)와 (나)의 밝기 차 이진화 이미지이다. (다)를 통해선 움직임 영역만을 나타내고 있지는 않다. 그래서, (라)에서 보이는 것처럼 CDM은 움직임 영역과 배경 영역을 구별시키는데 사용되어 진다.

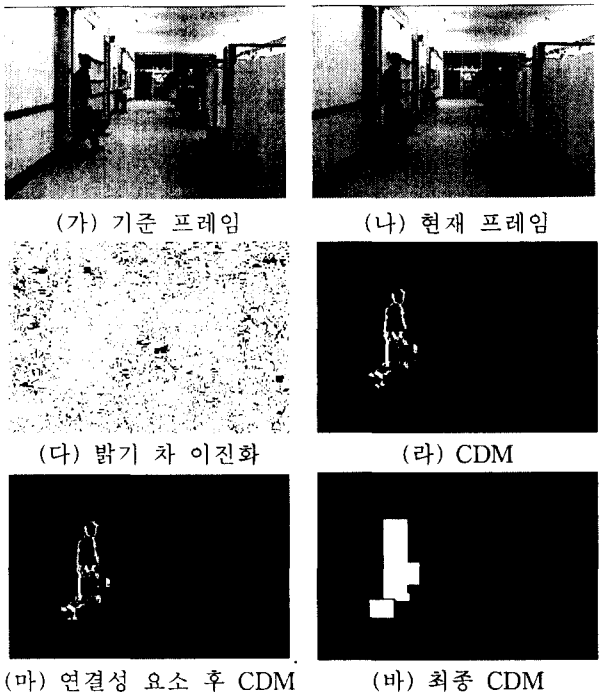
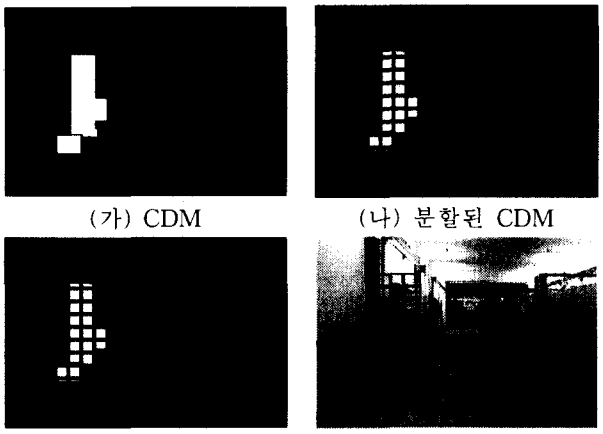


그림 2. CDM 발생 과정

그러나, 그림 2의 (라)에서처럼, CDM은 잡음을 포함하고 있으며, 이는 후처리 과정을 통해 해결할 수 있다 [9]. CDM은 물체의 움직임이 있는 부분의 내부가 아닌 외부의 경계만이 변한 것으로 나타난다. 이런 문제점들은 인접 픽셀의 연결성 성분 조사에 의해 쉽게 해결 될 수 있다. 그림 2의 (마)는 연결성 성분 조사에 의해 잡음을 제거한 CDM이며, 움직인 물체를 사각형으로 구성 CDM은 그림 2의 (바)에서 보여주고 있다.

CDM이 얻어진 후, 현재 프레임에 있는 모든 블록들

은 분류되어진다. 블록들을 분류하기 위해, 우린 *BC(i)*를 사용한다. *BC(i)*는 현재 프레임의 *i*번째 블록에 대한 블록 정보를 나타낸다. 우선, CDM은 고정된 크기의 블록들로 나누어진다. CDM내의 *i*번째 블록의 픽셀들이 움직인 물체에 속한다면, 현재 프레임의 *BC(i)*는 움직임 블록으로 분류되어진다. 그렇지 않으면, *BC(i)*는 배경 블록으로 분류되어진다. 블록 분류 과정은 그림 3에서 보여진다. (가)는 CDM이며, (나)는 블록 크기로 나누어진 CDM이다. (다)는 CDM에서 블록을 분류한 영상이다. 여기서 사각형은 움직임 블록들을 나타낸다. (라)는 현재 프레임에서 블록을 분류한 영상이다.



(가) CDM (나) 분할된 CDM

(다) 분류된 CDM (라) 분류된 현재 프레임

그림 3. 블록 분류 과정

2.2 움직임 벡터 추정

블록들이 분류되어진 후, 현재 프레임에 있는 모든 블록들에 대한 움직임 벡터는 *BC(i)*를 이용해 추정되어진다. *BC(i)*가 움직임 블록이면, 기존의 연구된 FS와 TSS와 같은 BMA이 움직임 벡터를 추정하기 위해 실행되어진다. 그렇지 않으면(*BC(i)*가 배경 블록이면), 제로의 움직임 벡터를 할당한다. 제로의 움직임 벡터를 할당하는 것은 배경 영역에는 움직임이 존재하지 않는다는 것을 말한다. 비디오 영상에서 배경 블록의 구성 비율이 매우 높기 때문에, 움직임 벡터를 추정하는데 소유되는 상당한 시간을 절약 할 수 있다. 배경 블록의 구성 비율에 따른 프레임 당 탐색점 수를 기존의 연구된 FS와 TSS와 비교한 것은 표 1에서 보인다.

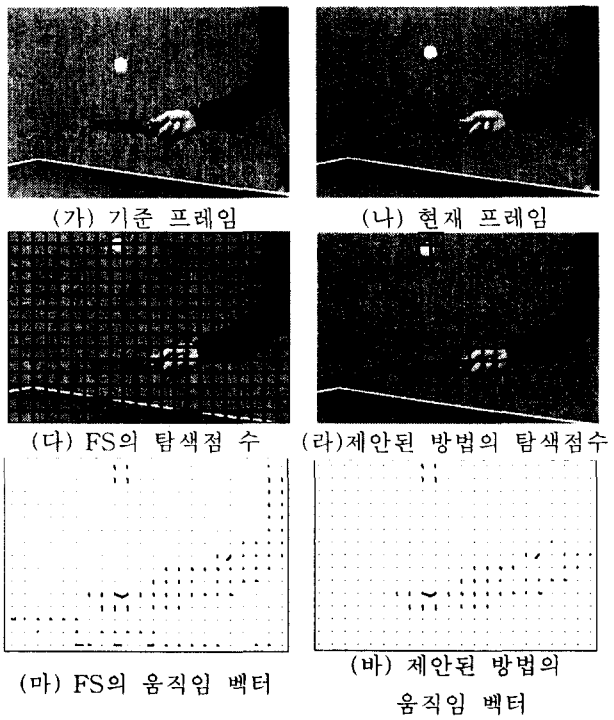
표 1. 배경 블록 구성 비율에 따른 탐색점의 수 비교

배경 블록 비율	50%	70%	90%
FS	86,700	86,700	86,700
TSS	8,100	8,100	8,100
제안된 FS	43350	26010	8670
제안된 TSS	4050	2430	810

표 1에서, 각각의 비율은 프레임 당 배경 블록의 구성 비율이며, 이미지 크기는 320×240이며, 블록 크기는 16×16이며, 탐색 범위(search range)는 ±16이다.

III. 실험 결과

본 논문에서 제안한 방법 평가하기 위해서 잘 알려진 비디오 영상들에서 실험 해 보았다. 우린 움직임 벡터를 추정하기 위해 FS와 TSS를 실행한 제안된 방법을 FS와 TSS의 질적인 면과 계산량의 관점에서 비교해 보았다. 전자는 PSNR(peak signal to noise ratio)에 의해 비교하였고, 후자는 탐색점의 수와 펜티엄 400 개인용 컴퓨터에서 실행한 프로그램 실행 시간에 의해 비교하였다. 실험에서 사용된 비디오 영상은 *"Hall-monitor(352×240)", "Claire(240×208)", "Table tennis(351×239)"를 사용하였다.* 블록 크기는 16×16이고, 탐색 범위는 수직과 수평 방향으로 ±16이고, 정합 기준(matching criterion)은 MSE(mean square error)를 사용하였다. 또한, CDM을 얻기 위한 임계값과 연결성 요소의 크기는 실험을 통해 얻어진 값을 사용하였다.



(가) 기준 프레임 (나) 현재 프레임

(다) FS의 탐색점 수 (라)제안된 방법의 탐색점수

(마) FS의 움직임 벡터 (바) 제안된 방법의 움직임 벡터

그림 4. 제안된 방법과 FS의 움직임 벡터와 탐색 블록의 수 비교

그림 4는 제안된 방법과 FS의 움직임 벡터와 탐색 블록에 대한 비교를 보여주고 있다. (가)와 (나)는 "*Table tennis*"의 30번째와 31번째 프레임이고, (다)와 (라)는 FS와 제안된 방법에서의 탐색 블록들을 보여 주고 있다.

표 2. 제안된 알고리즘의 평균 PSNR, 탐색점 수, 속도 비교

알고리즘	FS			TSS			제안된 FS			제안된 TSS		
실험 영상	Claire	Hall-monitor	Table tennis	Claire	Hall-monitor	Table tennis	Claire	Hall-monitor	Table tennis	Claire	Hall-monitor	Table tennis
PSNR	41.18	35.05	31.60	41.06	35.01	31.26	40.42	34.84	31.48	40.37	34.83	31.12
탐색점 수	56355	95370	84996	5263	8910	7938	5321	6902	37437	561	727	3665
속도	0.71	1.21	0.72	0.07	0.12	0.07	0.1	0.11	0.11	0.02	0.04	0.04

제안된 방법에서의 탐색 블록의 수가 현저히 적음을 볼 수 있다. (마)와 (바)는 FS와 제안된 방법을 통해 얻어진 움직임 벡터를 보여주고 있다. 그림에서 보듯이, FS는 배경 영역에 대한 잘못된 움직임 벡터를 구하는 반면, 제안된 방법에선 이런 배경 영역에 대한 움직임 벡터를 구하지 않음으로써 더 정확한 움직임 추정을 할 수 있다.

제안된 방법의 PSNR, 탐색점 수, 속도 면에서의 실험 결과를 표 2에서 보여주고 있다. 제안된 알고리즘의 질적인 평가를 위한 평균 PSNR은 약간의 차이를 보이고 있다. 또한, 제안된 알고리즘의 속도는 프로그램 실행 시간과 프레임 당 탐색점 수의 관점에서 평가하였다. 제안된 알고리즘의 평균 실행 시간은 블록 분류 시간과 움직임 벡터 추정 시간으로 구성되어 있다. 제안된 알고리즘의 실행 시간은 FS와 TSS의 실행 시간보다 빠른 것을 볼 수 있다. 또한, 제안된 알고리즘의 평균 탐색점 수는 FS와 TSS의 평균 탐색점 수보다 10배 이상 적은 것을 볼 수 있다. 이런 결과로부터, 제안된 방법이 계산량을 상당히 줄일 수 있으며, 이는 질적인 면을 유지하면서 실시간의 빠른 비디오 코딩에 적용될 수 있음을 알 수 있다.

IV. 결 론

본 논문에서는 계산량을 향상시킬 수 있는 빠른 움직임 추정 알고리즘을 제안하였다. 제안된 방법에서, 현재 프레임에 있는 모든 블록들은 움직임 블록과 배경 블록으로 분류하였다. 우리는 블록 분류를 위해 CDM을 사용하였다. 움직임 블록에 대해서만 기존의 BMA을 사용하여 움직임 벡터를 추정하였다. 움직임 블록에서만 움직임 벡터를 추정하였기 때문에, 움직임 추정을 위한 상당한 시간을 절약할 수 있었다.

그러나, 제안된 알고리즘은 움직임이 없는 배경을 가진 비디오 영상에서만 좋은 결과를 얻을 수 있다는 문제점을 갖고 있다. 이러한 문제점을 해결하기 위해서 계속 실험 중에 있다.

참고 문헌

[1] D. L. Gall, "MPEG: A video compression standard for multimedia application", *Communication of the ACM*, Vol. 34, pp. 46-58, April 1991.

[2] M. Liou, "Overview of the Px64 kbit /s videocoding standard", *Communication of the ACM*, Vol. 34, pp.59-63, April 1991.

[3] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and Ishiguro, Motion compensated interframe coding for video conferencing , in *Proc. NTC 81*, New Orleans, LA, pp. 9.6.1-9.6.5, November/December 1981.

[4] R. Li, B. Zeng and M. L. Liou, "A New Three-Step Search Algorithm for Block Motion Estimation", *IEEE Trans. Circuits and Systems for Video Tech.*, Vol. 4, pp 438-442, August. 1994.

[5] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding", *IEEE Trans. Commun.*, Vol. COM-29, pp. 1799-1808, December. 1981.

[6] M. Ghanbari, "The cross-search algorithm for motion estimation", *IEEE Trans. Commun.*, Vol COM-38, pp. 950-953, July 1990.

[7] H. S. Oh, C. H. Lee, H. K. Lee and J. H. Jeon, "A New Block-Matching Based on an Adaptive Search Area Adjustment Using Spatio-Temporal Correlation", *IEEE Trans. Consumer Electronics*, Vol. 45, No. 3, pp. 745-752, August 1999.

[8] N. Habili, A. Moini and N. Burgess, "Automatic Thresholding for Change Detection in Digital Video", *Proc. Visual Communications and Image Processing 2000*, Perth, Australia, Vol. 4067, pp. 133-142, June 2000.

[9] Y. C. Lee, Y. E. Kim, H. J. Kim, "Motion Detection Method for Nonstationary Video Sequences", *Proc. KISPS autumn conference 2000*, vol. 1, no. 2, pp. 137-140, December 2000.