블록을 이용한 비디오의 fade와 zoom 영역 검출 기법

정인식, 권오진
세종대학교 전자공학과

Detection of Fade and Zoom Effects Using Blocks in Video

In-shik Jeong, Oh-Jin Kwon
Dept. of Electronics, Sejong University

요 약

비디오에서 적은 수의 대표 화면으로 그 비디오의 내용을 요약할 수 있다는 것은 효율적인 비디오 브라우징 및 비디오 검색에 있어서 매우 중요하다. 다양한 종류의 셔트(shot) 추출 방법이 제시되어 왔다. 다양한 종류의 셔트 추출 방법 중에서 칼라 히스토그램을 이용하는 방법이 가장 많이 사용되어 왔다. 그러나 칼라 히스토그램을 이용하는 방법은 fade effect, zoom effect 등과 같이 특별한 효과가 들어있는 비디오에서는 적절하지 못한 결과를 종종 초래한다. 이 논문에서는 블록을 이용한 fade와 zoom 효과가 있는 영역을 검출하는 방법을 제시한다. 대부분의 칼라 히스토그램 방법은 인접한 프레임간 또는 일정한 거리가 떨어져 있는 프레임간의 차이를 이용하였다. 이 논문에서는 차이를 구하고자 하는 프레임간의 거리를 변동시키는 방법을 이용하여 구함으로써 그 성능을 개선하였고, 또한 단순히 두 프레임만을 비교하는 것이 아니라 일정한 수의 프레임을 그룹핑 하여 하나의 블록으로 만들고, 그 블록에서 히스토그램 차이의 평균 및 중간 값을 이용하면 hard cut과 fade같은 효과가 한 블록 내에 같이 있는 경우 더욱 효과적으로 셔트를 추출할 수 있다.

1. 서론

멀티미디어 시대가 도래하면서 전자 도서관 구축 및 인터넷 동영상 서비스가 활성화되고 있다. 네트워 환경과 컴퓨터 주변기기들의 발달로 불과 몇 년 전에는 상상할 수 없는 이러한 일들이 최근에 일어나고 있다. 그러나 네트워 환경이 많이 발전했지만 멀티미디어 데이터베이스는 그 용량 자체가 너무 크기 때문에 여전히 인터넷이라는 매개로 사용하기에는 많은 어려움이 있다. 그렇기 때문에 이러한 멀티미디어 데이터베이스를 의미를 나타낼 수 있는 가장 적은 수의 정지영상으로 보여 줄 수 있다면 비디오 브라우징을 할 때 매우 효율적일 것이다.

이 논문에서 우리는 블록을 이용하여 현재 나와있는 히스토그램 방법을 더욱 개선시킨 알고리즘을 제안한다. 2장에서는 기존의 방법들을 소개하고 3장에서는 제안하는 알고리즘을 소개하고, 4장과 5장에서는 실험결과와 결론을 각각 서술한다.

2. 기존의 방법들

현재까지 비디오를 요약하여 적은 수의 셔트로 그 비디오의 내용을 나타내기 위한 방법들이 많이 제시되어 왔다. 가장 보편적으로 쓰이는 방법은 히스토그램 방법이다. 이것은 연속된 비디오 이미지에서 인접한 프레임들 또는 일정한 거리가 떨어져 있는 프레임들의 히스토그램을 구하고 그 값의 차이를 구하여 그 값이 정해놓은 임계값보다 클 경우 셔트의 경계로 보는 방법이다. 그 중에서 가장 많이 사용되는 방법이 칼라 히스토그램을 이용하는 방법이다[1]. 다른 방법으로는 흑백성분의 영상만 가지고 히스토그램을 구하는 방법[2]이 있다. 또 한 두 프레임간의 차이를 구할 때 전체 프레임에 대해서 구하지 않고, 전체의 이미지를 4x4블록으로 나눠서 다음 프레임의 블록과의 히스토그램의 차이를 구하고 그 국부적인 값들의 합으로

셔트를 추출하는 방법도 사용되고 있다[3]. 그밖에 에지의 변화율을 이용하여 셔트의 경계를 찾아내는 방법도 제시되었으며[4][5]. MPEG비디오의 셔트 추출시에는 DCT계수를 이용하는 방법과[6], 더 나아가 I-,P-,B-frame에서 추출된 DC성분들만 가지고 히스토그램의 차이를 구하는 방법도 연구되었다[7].

이 논문에서는 흑백영상을 이용하여 히스토그램을 구하고 그것의 일정한 수의 프레임을 그룹핑 하여 블록을 만들고 그 블록 내에서 프레임간의 차이를 구하여 셔트를 추출한다.

3. 셔트의 추출

2장에서 기술한 것처럼 칼라 히스토그램이 일반적으로 사용되는 셔트 추출 방법이다. 그러나 대부분의 방법들이 인접한 프레임간의 차이를 사용하기 때문에 빠르게 움직이는 카메라 모션의 경우 일반적인 비디오물에 대해서 적용되어 있는 임계값을 사용하면 정확한 셔트를 찾아내는데 한계가 있다. 그래서 어느 정도는 가변적인 임계값이 필요하다.

또한 fade 효과와 hard cut이 인접하여 있는 경우, 현재의 방법들은 두 프레임의 단순한 비교를 사용하기 때문에 두 가지 중에서 하나는 추출하지 못하는 경우가 많았다. 특히, fade 효과 같은 경우에는 이미지 안에 물체들이 사라지거나 나타나는 현상이므로 물체의 변화에 대해서 추출하는 에지 검출 방법이나 화소들에 명도의 표준편차를 이용하는 방법에서는 제대로 추출이 된다[8]. 그러나 히스토그램 방법은 이런 효과에 대해서 비교적 정확하게 추출하지 못하는 경우가 대부분이다. 이러한 2가지 문제를 해결하기 위해서 우리의 알고리즘에서는 블록을 적용한 히스토그램 방법을 사용하였고, 칼라표현 형식으로는 Y/Cb/Cr을 사용한다. 그러나 블록을 이용하기 때문에 Cb/Cr을 모두 사용하면 계산양도 매우 많아지고 버퍼의 크기도 매우 커지게 되므로 Cb/Cr은 히스토그램에 사용하지 않고 Y성분만을 사용하였다.

프레임 M개를 그룹핑 하여 하나의 블록으로 만든다. 그 후에 블록 내에서 i-번째 프레임의 화소 중에서 Y성분이 k인 화소점의 수를 나타내는 $H^i(k)$를 정의한다. 여기에서 $i=1,2,3,.....,M-1$; $k=0,1,2,.....,N-1$이고 M은 그룹에 속하는 총 프레임 수, N은 비디오 표현 포맷에서 Y성분이 취할 수 있는 값의 총 개수이다.

그룹핑 한 후에 블록 내에서 프레임간의 히스토그램의 차이는 아래와 같이 나타낼 수 있다.

$$D(i,j)=\sum_{k=0}^{N-1}|H^i(k)-H^j(k)|\ ,\ i<j \qquad (1)$$

$D(i,j)$는 i-번째 프레임과 j-번째 프레임의 차이에 대한 척도이며 이 차이가 클수록 두 프레임간 변화가 심하고 적을수록 두 프레임은 변화가 적은 프레임으로 판별된다.

3.1 hard cut

일반적으로 히스토그램 방법에서 가장 잘 추출되는 것이 hard cut이다. hard cut은 한 블록 내에서도 넓은 영역에 걸쳐 나타나지 않고 인접한 프레임간에서만 국부적으로 발생하므로 hard cut이 발생하는 지역의 전체적인 $D(i,j)$ 값들이 반영된 임계값이 필요하다.

임계값을 찾기 위해 우선 M개의 프레임으로 구성된 블록에서 $D(i,j)$들의 평균값과 중간 값을 구한다.

$$D(AVG)=\frac{1}{M}\left(\sum_{i=1}^{M-1}\left(\sum_{k=0}^{N-1}|H^i(k)-H^j(k)|\right)\right) \qquad (2)$$

$$D(MID)=Median\{D(i,j)\} \qquad (3)$$

(3)식에서 $D(i,j)$값들의 중간 값을 구하기 위해서는 크기에 의해서 정렬을 해준 뒤 위에 방법을 적용한다.

우리가 찾고자 하는 값은 $D(i,j)$값들의 중간 값이다. 그런데 인위적인 편집이 되어있는 비디오 물에서는 인접한 프레임간의 $D(i,j)$값이 0 이 나오는 경우가 다수 발생하였다. 이런 경우는 $D(MID)$값이 매우 작은 값을 갖기 때문에 이 것을 보상해주기 위해서 $D(AVG)$가 필요하다.

(2),(3)식을 적용하여 hard cut이 발생하는 프레임을 다음과 같이 찾을 수 있다.

$$\begin{cases} D(MID)*h_1<D(i,j)\ ,\ D(MID)>\kappa 2 \\ D(MID)*h_2<D(i,j)\ ,\ \kappa 1<D(MID)\le \kappa 2 \\ D(AVG)*h_3<D(i,j)\ ,\ \text{otherwise} \end{cases} \qquad (4)$$

위와 같이 3가지 조건을 사용하여 셔트를 추출하면 $D(i,j)$값들에 의해 $D(MID)$와 $D(AVG)$값이 변하므로 다른 블록들과 비교해 봤을 때 임계값 h_1,h_2,h_3이 변하는 효과를 줄 수 있다.

3.2 fade 효과

fade 효과는 그림 1 과 같이 블록 내에서 $D(i,j)$가

어떠한 효과도 없는 $D(i,j)$보다는 크지만 hard cut이 발생하는 $D(i,j)$보다는 작다.

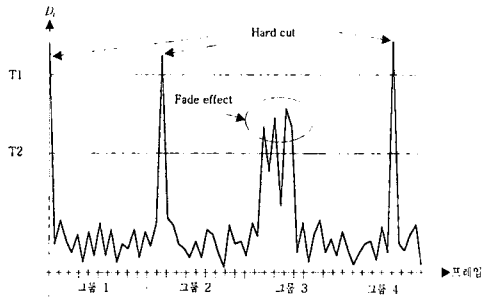

그림1. 프레임들간 차이에서 fade 효과를 찾는 방법

그림1에서 $T1$보다 큰 $D(i,j)$는 hard cut으로 보고, $T1$과 $T2$ 사이의 값을 찾는다.

NO.(large D) = [$D(i,j) > T2$인 프레임의 개수] - [$D(i,j) > T1$인 프레임의 개수]

위에 NO.(large D)를 한 블록에 대해서 적용시키면 그 블록이 fade 효과가 있는지를 알 수 있다.

$$\begin{cases} \text{NO.(large D)} \geq F_1 \ , \ D(AVG) > \kappa_3 \\ \text{NO.(large D)} \geq F_2 \ , \ \text{otherwise} \end{cases} \quad (5)$$

fade 효과가 한 블록에 2개 이상 들어있는 경우는 2개가 정확하게 추출이 안 될 가능성이 매우 높기 때문에 적당한 M값을 찾는 것이 매우 중요하다. (5)식과 같은 방법을 사용하면 같은 블록 안에 hard cut과 fade 효과가 존재할 때 NO.(large D)에 의해서 hard cut을 보상해주기 때문에 hard cut과 fade 효과를 각각 추출할 수 있다.

3.3 zoom 효과

zoom 효과의 특징을 보면 인접한 프레임간의 $D(i,j)$의 값은 아무런 효과도 없는 프레임간의 $D(i,j)$와 거의 비슷하게 나온다. 그러나 i,j의 거리가 길어질수록 $D(i,j)$값이 매우 커짐을 알 수 있다. 이러한 특성을 이용하면 zoom 효과를 효과적으로 추출 할 수 있다.

먼저 우리의 방법에서는 현재의 블록을 길이가 M/n인 n개의 서브블록으로 나눈다. 여기서 $n \leq M/5$다. 그런 후에 각각의 서브블록에 대하여 서브블록의 첫 프레임과 다른 프레임들간의 $D(1,j)$를 구한 후 아래의 연산을 수행한다.

$$Si = D(1, i+3) - D(1, i+1) \quad (6)$$

여기서 i는 $(i = 1,2,3,4.....) < (\frac{M}{n} - 1)$이다. Si는 zoom 효과에서 프레임간 거리에 비례해서 $D(i,j)$가 커지기 때문에 첫 프레임과 각각의 거리의 프레임들 차이들을 구하여 비교하는 함수이다.

만약 i번째 Si값이 (-)인 경우에는 $D(1,i+3)$ 와 $D(1,i+1)$의 값을 바꾸어준다. 이것은 다음의 i+2번째 연산에서 $D(1,(i+2)+1)$의 값에 $D(1,i+3)$가 아닌 $D(1,i+1)$넣어주는 작용과 같다. 이것은 가능한 잘못된 zoom 효과를 방지하기 위한 방법이다.

현재의 블록이 zoom 효과가 있는지를 알기 위해서 각각의 서브블록에서 (-)값이 나오는 Si의 개수가 Z_1개 이하로 나오는 서브블록이 존재하고, 그러한 서브블록의 개수가 전체블록에서 Z_2개 이상이면 zoom effect가 존재하는 블록이다.

그러나 (6)식과 같은 방법은 zoom과 camera motion이 같이 잡히기 때문에 두 가지를 구별해 주어야 한다. 한 프레임의 이미지를 LxL개의 서브이미지로 분할을 한 후 아래의 연산을 수행해준다.

$$M_{bmin}(1,j) = \min_l \left(\sum_{k=0}^{N-1} |H_p^1(k) - H_l^j(k)| \right)$$
$$, 1<j<M, \ 0<l,p<LxL-1 \quad (7)$$

위의 함수는 첫 번째 프레임의 p번째 서브이미지와 j번째 프레임에서 가장 잘 매칭되는 l번째 서브이미지를 찾는 역할을 한다. 그 후 $H_p^1(k)$의 위치와 $H_l^j(k)$들의 위치의 벡터 차를 구하면 블록에서 시간의 흐름에 따라 수평 및 수직 방향으로 $H_p^1(k)$이 어느 정도 이동했는 지를 알 수 있다. 여기서 첫 프레임의 서브이미지 p중에서 같은 방향으로 모이거나 또는 원점을 기준으로 대각선으로 발산하는 p의 개수가 Zp개 이상의 그 블록에 zoom 효과가 존재하고, 방향이 다른 경우는 camera motion이다.

4. 실험 결과

우리는 변화가 적은 뉴스나 드라마보다는 변화가 다양한 광고물에 대해서 실험을 하였다. 또한 비디오의 크기는 352x240, 320x240등으로 하였고, 여기서 M값은 사람이 비디오 물에서 변화를 감지하기에 적당한 시간을 3초로 간주하여 MPEG1을 기준으로 45프레임

으로 하였다. 그리고 Y성분이 취할 수 있는 값의 총 개수는 64로 양자화 시켜서 사용하였다.

각각의 임계값들은 h_1,h_2,h_3은 7,8,8을 주었고, $\kappa 1, \kappa 2, \kappa 3$는 3000과 10000,10000을 주었다. 또 한 F_1,F_2는 3,4이고, zoom 효과를 추출하기 위한 임계값으로는 n은 3개의 서브블록으로 나눴고, Z_1,Z_2는 1,2 , Zp는 한 프레임에서 전체p의 개수의 절반이상, 그리고 L은 16을 사용하였다. 이러한 임계값을 가지고 셔트를 추출한 결과를 보면 다음과 같다.

sample video	hard cut	fade	zoom
cf1	8/9	2/3	4/6
cf2	5/5	2/2	0/0

표1. 셔트 추출 결과

(6)식과 같은 방법을 사용하면 zoom 효과와 hard cut이 한 블록에 존재하는 경우에도 2가지 효과를 각각 추출할 수 있고, 한 서브 블록 내에 잡음이 존재하여도 나머지 서브 블록들에 의하여 zoom 효과를 추출해 낼 수 있다. 예로 cf1의 경우 386프레임에 hard cut이 발생하는데 이것은 zoom 효과가 있는 동안에 발생하였다. 우리의 실험결과에서는 2가지 모두 추출되었다. (7)번식의 블록의 중간부터 zoom 효과가 나타나는 경우 이동한 서브이미지의 정확한 위치를 찾기가 힘들다. 이럴 때는 (7)번식을 블록의 마지막 프레임에서부터 거꾸로 시간의 흐름에 따라 매칭되는 서브이미지를 찾으면 zoom 효과가 중간부터 일어나는 경우도 보상해 줄 수 있다.

5. 결론

본 논문에서는 블록을 이용하여 셔트를 추출하는 방법을 제시하였다. 일반적으로 히스토그램 알고리즘은 대부분 fade 와 zoom 효과에 대해서 만족할 만하게 셔트를 추출하지 못했다. 그러나 본 논문에서 제시한 알고리즘을 사용하면 기존의 히스토그램 알고리즘에서 추출하지 못했던 효과들을 추출할 수 있게 되었다.

또한 hard cut의 경우 기존의 방법은 고정된 임계값을 사용하기 때문에 여러 종류의 비디오에 대해서 모두 만족할 만하게 추출되지 않는 반면 우리의 방법은 임계값이 가변하는 효과를 주기 때문에 다수의 비디오에 대해서 평균적으로 좋은 성능을 낸다. 그러나 본 논문에서 제안한 방법은 가변의 hard cut 임계값을 가지고 있기 때문에 특정 비디오에 맞추어진 특정한 임계값을 사용하는 기존의 방법보다 낮은 검출율을 보이는 경우가 발생하는 단점을 보이기도 한다. 향후에 이에 대한 추가 연구가 요구된다.

[참고문헌]

[1] H. Ueda, T. Miyatake, and S. Yoshizawa, "IMPACT: An Interactive Natural-motion-picture Dedicated Multimedia Authoring system," Proceedings of CHI, 1991, New Orleans, Louisiana, Apr.-May, 1991, ACM, NewYork, pp.343-350

[2] A. Nagasaka and Y. Tanaka "Automatic video indexing and full motion search for object appearances," Proceedings of IFIP TC2/WG2.6 Second Working Conf. on Visual Database Syst, pp.113-127, 1991.

[3] M. A. Smith and T. Kanade, "Video skimming and characterization through the combination of image and language understanding techniques," Proceedings of CVPR'97, (Puerto Pico), pp.775-781, 1997.

[4] J. Canny. "A computational Approach to Edge Detection" IEEE Transaction of Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, pp.34-43, 1986.

[5] R. Zabih, J. Miller, and K. Mai. "A Feature-Based Algorithm for Detecting and Classifying Scene Breaks" Proceedings of ACM Multimedia 95, San Francisco, CA pp.189-200, 1995.

[6] F. Arman, A. Hsu, and M.-Y. Chiu, "Image processing on encoded video sequences," Multimedia Systems, Vol. 1, no. 5, pp. 211-219, 1994.

[7] B.-L. Yeo and B. Liu, "Rapid scene analysis on compressed video" IEEE Transaction of Circuits, System, Video Technology, Vol. 5, pp.533-544, 1995.

[8] R. Lienhart "Comparison of Automatic Shot Boundary Detection Algorithms" Storage and Retrieval for Image and Video DatabasesVII, Vol SPIE-3656, pp.290-301, 1999.