압축영역에서 객체 움직임 맵에 의한 효율적인 비디오 인덱싱 방법에 관한 연구

김 소 연† · 노 용 만††

요 약

객체 움직임은 비디오 시퀀스에서 내용을 나타내는 매우 중요한 특징 정보 중 하나이다. 지금까지 비디오 데이터에서 객체 움직임과 관련된 대부분의 방법들은 정확한 특징 추출에 중점을 두어왔다[1, 2]. 그러나, 이러한 기존의 방법들을 이용하여 움직임에 관한 인덱싱을 하기 위해서는 많은 저장공간이 필요하며, 복잡한 인덱싱 변수들로 인해 움직임을 이용한 비디오 인덱싱에는 적합하지 않았다[3, 4]. 따라서, 본 논문에서는 객체 움직임을 위한 효율적인 인덱싱 방법으로 객체 움직임 맵을 제안하고자 한다. 제안된 객체 움직임 맵은 객체가 움직이는 동안, 움직이는 객체의 전체적인 정보 뿐만 아니라 부분적인 움직임 변화 정보 또한 자세히 포함하고 있으며 적은 비트 수로 인덱싱할 수 있다. 제안된 인덱싱 기술의 성능평가를 위해, MPEG-7 테스트 시퀀스에서 MPEG-1 비디오 시퀀스들을 이용하여 데이터 베이스를 구성하였다.

An Efficient Video Indexing Method using Object Motion Map in Compressed Domain

So-Yeon Kim† · Yong Man Ro††

ABSTRACT

Object motion is an important feature of content in video sequences. By now, various methods to exact feature about the object motion have been reported[1, 2]. However they are not suitable to index video using the motion, since a lot of bits and complex indexing parameters are needed for the indexing[3, 4]. In this paper, we propose object motion map which could provide efficient indexing method for object motion. The proposed object motion map has both global and local motion information during an object is moving. Furthermore, it requires small bit of memory for the indexing. To evaluate performance of proposed indexing technique, experiments are performed with video database consisting of MPEG-1 video sequences in MPEG-7 test set.

1. 서 론

우리는 현재 멀티미디어의 홍수 속에 살고 있다. 각종 멀티미디어 데이터들은 놀라울 정도로 다양하며 사용자의 요구에 맞게 갖가지 방식으로 제작되어 있다. 그러나 사용자 요구에 의해 만들어진 이 방대한 자료들 사이에서 정작 사용자가 원하는 데이터를 찾아내기란 쉽지 않은 일이 되버렸다. 따라서 이러한 멀티미디어 데이터를 효과적으로 관리하고 검색하기 위해 데이터에 대한 정확한 특징 추출 및 효율적인 인덱싱(indexing) 기술이 무엇보다도 절실히 요구되고 있다[1]. 본 논문은 이러한 연구의 일환으로 멀티미디어 데이터를 대표하는 MPEG 비디오 시퀀스내의 움직임 정보를 이용하여 객체 움직임을 효과적으로 인덱싱하는 방법에 관해 연구한다.

† 정 회 원 : 한국정보통신대학교 대학원
†† 종신회원 : 한국정보통신대학교 대학원 교수
논문접수 : 1999년 10월 21일, 심사완료 : 2000년 4월 26일

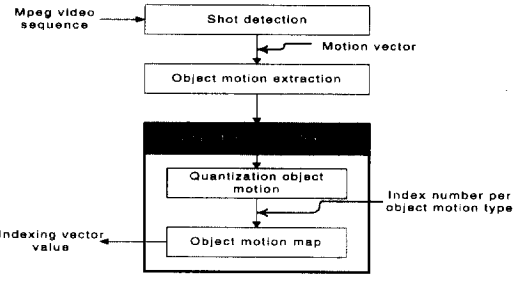
비디오 시퀀스란 여러 장의 영상들이 모여서 이루어진 의미 있는 내용의 흐름이라 할 수 있으며, 이러한 내용 흐름에서 가장 중심이 되는 요소가 바로 움직이는 객체이다. 따라서 움직이는 객체를 파악하고 이에 관한 정보를 수집하고자 하는 여러 시도들이 있어왔다[2]. 이러한 대부분의 방법들은 객체 움직임에 대한 정확한 특징 추출에 초점을 맞추어 인덱싱을 하기에는 비효율적이다. 특징 추출에만 중점을 둔 기존의 방법들은 압축된 비디오 시퀀스를 복호화한 뒤 정확한 객체 움직임 추출을 위한 여러 복잡한 알고리즘들을 적용하였다[2-4]. 따라서, 이와 같은 방법들을 인덱싱에 적용하기 위해서는 많은 비트 수를 할당해야 하며, 복잡한 인덱싱 변수들로 인해 객체 움직임을 한눈에 파악하기 힘들다. 이외에 객체 움직임 인덱싱에 중점을 둔 방법으로는 객체 움직임을 크기와 방향에 관한 개별적인 히스토그램으로 나타내는 방법이 있다[3]. 히스토그램에 의한 인덱싱 방법은 비교적 단순하고 쉬우나, 개별적인 크기와 방향 히스토그램을 조합하여 비디오 시퀀스내의 객체 움직임을 한 눈에 예측하기란 쉽지 않다. 더구나 히스토그램 자체를 저장하기 위해서는 많은 저장 공간을 요구한다. 이처럼, 기존의 방법들은 압축된 비디오 시퀀스를 복호화한다거나 많은 비트량을 차지하는 인덱싱 벡터를 이용함으로써 시간과 저장 공간의 낭비를 가져왔으며, 사용자가 이해하기 어려운 인덱싱 변수를 사용하였다.

따라서 본 논문에서는 빠른 시간에, 적은 비트 수로 인덱싱 가능하며, 비디오 시퀀스내의 객체 움직임 변화를 한눈에 알 수 있는 효율적인 인덱싱 방법에 관해 연구하였다. 우리는 이를 '객체 움직임 맵'이라 정의한다.

본 논문의 구성은 다음과 같다. 2장에서는 제안한 방법에 관해 자세히 기술한다. 3장에서는 실험 및 결과에 관한 사항을 다루고, 마지막 4장에서는 결론을 제시한다.

2. 제안 방법

(그림 1)은 제안한 방식에 의해 압축영역에서 비디오를 인덱싱하는 전체적인 과정을 보여준다. MPEG 비디오 시퀀스가 입력되면, 인덱싱의 기본 단위가 될 샷을 검출해 낸다. 샷이 검출되면, 매크로 블록 당 움직임 벡터에 근거하여 객체 움직임을 검출해 내고 객체 움직임들을 인덱싱한다. 인덱싱 순서는 다음과 같다. 먼저, 객체 움직임을 양자화한 후, 객체 움직임 종류 별로 인덱스 번호를 부여한다. 그리고 이를 기반으로 객체 움직임 맵을 작성하게 되는데, 객체 움직임 맵은 비디오 시퀀스내의 객체 움직임 변화 추이(객체 움직임 궤적 : object motion trajectory)를 나타내는 좋은 특징 변수가 된다. 따라서, 제안된 객체 움직임 맵은 객체 움직임에 의해 효과적으로 비디오 인덱싱을 수행할 수 있게 한다.

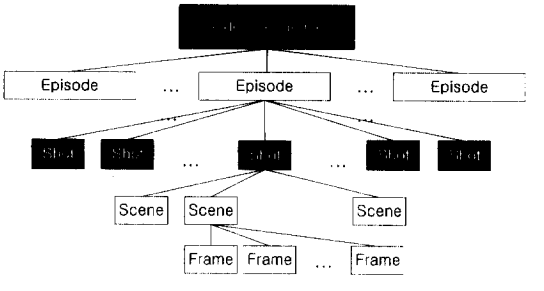


(그림 1) 제안된 방법에 의한 인덱싱 순서

다음 장은 (그림 1)의 각 세부사항에 대해 자세히 기술한다.

2.1 샷 검출

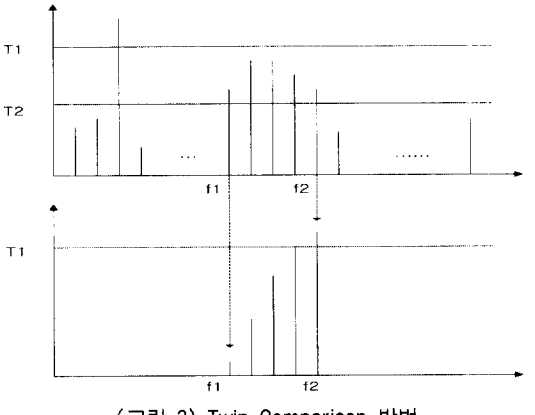
비디오 시퀀스를 의미 계층별로 분석해 보자면 (그림 2)와 같이 나타낼 수 있다.



(그림 2) 비디오 시퀀스 계층 모델

먼저, 프레임이란 정지 영상 한 장, 한 장을 일컫는 말이며, 서로 의미적으로 유사한 정지 영상들의 모임을 신(scene)이라 한다. 유사한 신들은 같은 방식으로 모여 샷(shot)을 구성하게 되는데, 영화에서는 흔히 카메라의 움직임이 동일한 장면, 혹은 유사한 신들의 모임을 샷이라 규정한다. 샷의 상위 개념인 에피소드

(episode)란, 말 그대로 샷들이 모여 간략한 토막 토막의 줄거리를 형성 한 것이며, 최종적으로 이러한 에피소드들이 모여 전체 비디오 시퀀스를 구성하게 된다. 여기서 샷과 신은 보통 구별되지 않고 사용되는 경향이 강한데, 두 단위들은 의미적으로 구별이 힘들며, 프레임 길이를 보더라도 별반 차이가 나지 않기 때문이다. 우리는 객체 움직임 검색, 출력의 기본단위로 샷을 사용한다. 이러한 샷 검출은 비디오 인덱싱에 기본이 되므로 그 자체 만으로도 많은 연구가 되고 있다[5]. 그러나 본 연구에서는 객체 움직임 인덱싱에 중점을 두므로 기존의 단순하고 쉬운 샷 검출 방법을 적용하였다. 사용된 방법은 Twin-Comparison 방법[5]으로 급격한 장면 전환뿐만 아니라 점진적인 장면 전환을 갖는 샷을 검출해 낼 수 있다.



(그림 3) Twin-Comparison 방법

이 방법에서는 점진적인 변화를 검출하기 위해서 (그림 3)과 같은 두 개의 임계값을 적용한다(T2 < T1). Twin-comparison에 의한 샷 검출 단계는 다음과 같다.

- **단계 1** : 프레임 차분치[1]가 T1보다 크면 현재 프레임은 급격한 샷 전환의 시작 프레임으로 인식된다.
- **단계 2** : 프레임 차분치가 T1보다 작고, T2보다 크면 현재 프레임은 점진적인 샷 전환의 시작 프레임(f1)으로 잠재적으로 인식된다. 단계 3을 수행한다.
- **단계 3** : 다음 프레임 차분치를 계산 후 이 값이 T2보다 작으면 f1은 탈락되고, 단계 5를 수행한다. 이 값이 T2보다 클 경우는 <프레임 차분치-T2>를 계산하여 누적 시킨 후 단계 4를 수행한다.
- **단계 4** : 누적 값이 T1보다 작은 경우는 단계 3을 수행하고, 누적 값이 T1보다 클 경우는 단계 2의 f1은 점진적 샷 전환의 시작 프레임으로, 현재 프레임 f2는 샷 전환의 마지막 프레임으로 인식된다. 단계 5를 수행한다.
- **단계 5** : 모든 프레임에 대해 프레임 차분치가 계산되었으면, 샷 검출을 중지하고 그렇지 않으면 단계 1을 수행한다.

2.2 객체 움직임 검출

MPEG 비디오 시퀀스는 매크로 블록(16×16 픽셀) 당 움직임 벡터를 지니고 있다. 따라서 완벽한 복호화 과정을 거치지 않고 부호화된 비트 스트림내에서 움직임 정보를 얻어낼 수 있다. 프레임 당 움직임 정보가 얻어지면, 움직임 객체를 세그멘테이션 하여 객체 움직임을 검출해 낸다. 그러나, MPEG 움직임 벡터에 근거하여 움직임을 검출해 내는 방법은 카메라 움직임이나 조명의 변화와 같은 요인에 의해서도 움직임이 검출되게 된다. 본 논문에서 사용한 비디오 시퀀스는 카메라 움직임과 조명 변화를 지니지 않는 시퀀스이지만 카메라 떨림과 같은 현상에 의한 잘못된 움직임 검출을 제거하기 위해 간단한 임계값 방법을 사용하였다. 즉, 임계값(본 논문에서는 임계값을 2로 두었다.)을 두어 임계값 이하의 미세한 움직임은 제거하고 실제 움직이는 물체의 움직임만을 추출토록 하였다. 또한, 객체로써 인식할 수 없는 지나치게 작은 매크로 블록 세그먼트들을 배제하기 위해 역시, 임계치를 두어 3개 미만의 매크로 블록 세그먼트들이 객체로 인식될 경우에는 이를 객체로 인식하지 않았다. 이와 같은 임계치들은 실험 데이터들을 여러 번 실험한 결과 얻은 실험치들이다.

객체 움직임을 세그멘테이션 해내는 방법은 다음과 같다. 먼저, 프레임 내 매크로 블록에서 임계값보다 큰 움직임 벡터를 찾아 합당한 객체 움직임(valid object motion)의 초기치로 삼는다. 다음으로 객체 움직임을 세그멘테이션하기 위해, 초기 객체 움직임 벡터 4방향으로 이웃한 매크로 블록의 움직임 벡터 값을 조사하여 이 값들이 역시 임계값 보다 큰 경우는 객체 움직임들을 같은 그룹으로 연결한다. 이 과정을 모든 매크로 블록에 적용하면서 객체 움직임을 세그멘테이션 한다. 보다 구체적인 알고리즘은 다음과 같다.

1) 현재 프레임의 픽셀 값에서 연속하는 다음 프레임의 픽셀 값을 뺀 정량적인 값.

```
Connect(int x, int y){
  if( MV[x][y].check ==FALSE){
    if( MV[x][y].value > THRESHOLD ){
      if(!(pre_x==x && pre_y==y-1)) Clustering(x, y);
       Pre_x=x; Pre_y=y;
       Connect(x, y-1);
       Connect(x+1, y);
       Connect(x, y+1);
       Connect(x-1, y);
    }
    MV[x][y].check=TRUE;
  }
}
```

Connect ( ) : 매크로 블록의 움직임 벡터 값이 임계값을 넘는지를 조사하여 객체 연결성 여부를 체크하는 함수

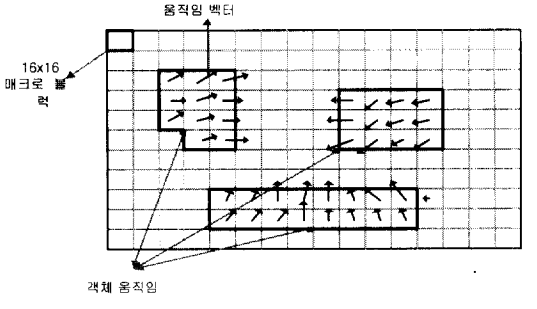
Clustering ( ) : 임계값을 넘는 매크로 블록들을 그룹화 하는 함수

THRESHOLD : 객체 움직임으로 인정할 수 있는 가장 작은 값.

x : 매크로 블록의 x 좌표, **y** : 매크로 블록의 y 좌표

MV[xl[y].value : x, y좌표에서 매크로 블록의 움직임 벡터 값

MV[x][y].check : x, y 좌표를 기준으로, 상·하·좌·우 연결성을 모두 조사 하였으면 TRUE.



(그림 4) 객체 움직임 검출

(그림 4)는 세그멘테이션을 통해 객체 움직임이 3개 검출된 예이다.

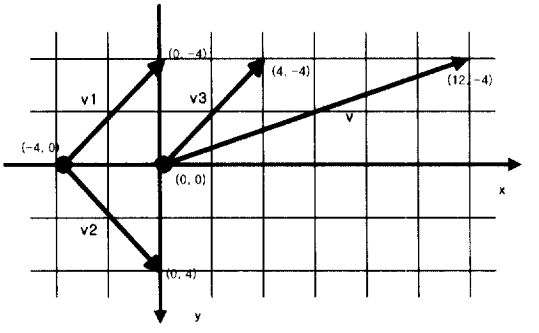
세그멘테이션 된 움직임을 대표하는 객체 움직임 벡터 값은 식 (1)과 같은 벡터 합 연산을 통해 구해진다.

$$OM(X, Y) = \sum_{i=1}^{N} MV_i(x, y) \qquad (1)$$

여기서, N은 세그멘테이션 된 매크로 블록의 개수를 나타내며 MV_i는 i번째 매크로 블록의 움직임 벡터 성분이다. OM은 객체 움직임 벡터 성분이다.(x, X는 수평 움직임 벡터 성분, y, Y는 수직 움직임 벡터 성분을 의미한다.) 따라서 객체 움직임 벡터의 크기 M과 방향 D는 식 (2), (3)과 같이 구해진다.

$$D = \angle OM(X, Y) = -\tan^{-1}(Y/X) \qquad (2)$$

$$M = |OM(X, Y)| = (\sqrt{X^2 + Y^2})/N \qquad (3)$$



* 화면의 x, y좌표는 MPEG프레임의 x, y 좌표를 기준으로 한 것이다.

(그림 5) 벡터 연산

예를 들어, (그림 5)와 같은 세 종류의 움직임 벡터가 한 객체에 있다고 생각해 보자. 각각의 움직임 벡터는 $V1 = (4, -4)$, $V2 = (4, 4)$, $V3 = (4, -4)$와 같은 벡터 좌표를 지니고 있다. 따라서 세 벡터의 움직임을 대표하는 벡터 합은 다음과 같이 구해질 수 있다.

$$V = V1+V2+V3 = (4, -4)+(4, 4)+(4, -4) = (12, -4)$$

이때 $V1$, $V2$, $V3$가 객체로 세그멘테이션된 MPEG 비트 스트림내의 매크로 블록(MV_i)을 나타내므로 V는 OM을 나타낸다고 할 수 있다. 따라서 이 예제에서 $OM(12, -4)$의 방향(D)은 식 (2)에 의해 $\tan^{-1}(-4/12) = 18.43° = 0.32\ rad$이며, $OM(12, -4)$의 크기(M)은 식 (3)에 의해 $\sqrt{12^2+(-4)^2}/3 = 4.22$이다.

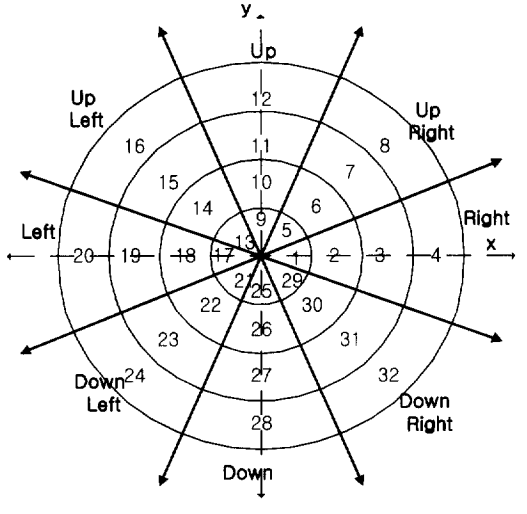
2.3 객체 움직임 인덱싱

크기와 방향으로 대표되는 객체 움직임을 효과적으로 인덱싱하기 위해서 극 좌표계를 사용하였다(그림 6 참조). 따라서, 직교 좌표상의 객체 움직임 벡터를 나타내는 식 (1)은 극 좌표계로 식 (4)와 같이 표시될 수 있다.

$$OM(m, d) = quant[OM(X, Y)] \qquad (4)$$

여기서, m은 양자화된 객체 움직임 크기(magnitude),

d는 양자화된 객체 움직임 방향(direction), 그리고 *quant* [·]은 양자화 연산자를 나타낸다. 결국, 객체 움직임은 m개의 크기와 d개의 방향으로 분할된 극 좌표계에 의해, $m \times d$개의 움직임 종류를 갖게 된다. 본 논문에서는 <표 1>과 같이 32종류로 양자화된 객체 움직임 종류를 정의 하였다. 물론, 크기를 4등분하고, 방향을 8등분하여 총 32종류의 인덱싱 번호를 부여한 것은 본 논문에 사용된 실험 데이터의 객체 움직임 속도와 방향을 염두에 둔 것이므로, 모든 데이터에 동일하게 적용될 필요는 없다. 만약, 움직임이 느리고, 방향이 단순한 영상의 경우에는 크기와 방향을 좀더 듬성, 듬성 양자화하여 하여 32보다 적은 수의 인덱싱 번호를 부여할 수 있으며, 반대의 경우에는 조밀한 양자화를 통해 32보다 더 큰 수의 인덱싱 번호를 부여 할 수 있다.



(그림 6) 객체 움직임 인덱싱을 위한 극좌표계

〈표 1〉 움직임 종류와 인덱싱 번호와의 관계

Direction \ Magnitude	Slow	Middle Slow	Middle Fast	Fast
Right	1	2	3	4
UP Right	5	6	7	8
UP	9	10	11	12
UP Left	13	14	15	16
Left	17	18	19	20
Down Left	21	22	23	24
Down	25	26	27	28
Down Right	29	30	31	32

<표 2>와 <표 3>은 <표 1>의 객체 움직임의 크기와 종류를 양자화한 방식을 보여준다. <표 1>에 의해, 각 종류별 객체 움직임들은 이에 해당하는 인덱스 번호와 대응된다. 즉, 객체 움직임 종류는 해당 번호로써 인덱스될 수 있다. (그림 6)는 <표 1>의 인덱스 번호가 부여된 객체 움직임 종류를 극 좌표계에 나타낸 것이다. 극 좌표계의 번호는 그 영역에 해당하는 움직임 벡터의 종류를 의미한다. 즉, 객체 움직임을 크기와 방향별로 양자화한 후 <표 1>, (그림 6)과 같은 인덱스 번호를 부여함으로써 객체 움직임을 인덱싱 하고자 했다.

〈표 2〉 양자화된 움직임 크기

Quantized magnitude=m	Magnitude range
Slow	$0 < M \le 8\sqrt{2}$
Middle slow	$8\sqrt{2} < M \le 16\sqrt{2}$
Middle fast	$16\sqrt{2} < M \le 24\sqrt{2}$
Fast	$24\sqrt{2} < M \le 32\sqrt{2}$

〈표 3〉 양자화된 움직임 방향

Quantized direction=d	Direction range
Right	$-\pi/8 \le D < \pi/8$
Up right	$\pi/8 \le D < 3\pi/8$
Up	$3\pi/8 \le D < 5\pi/8$
Up left	$5\pi/8 \le D < 7\pi/8$
Left	$7\pi/8 \le D < -7\pi/8$
Down left	$7\pi/8 \le D < -7\pi/8$
Down right	$-5\pi/8 \le D < -3\pi/8$
Right	$-3\pi/8 \le D < -\pi/8$

<표 2>는 식 (3)을 통해 얻은 객체 움직임 크기를 크게 4종류로 양자화한 표이다. 움직임의 최대 크기를 $32\sqrt{2}$로 둔 이유는 동영상 압축 표본인 MPEG-1,2에서 움직임 추정이 32픽셀의 탐색 영역을 지니므로 이를 통해 얻어낼 수 있는 최대 움직임 크기가 $32\sqrt{2}$가 될 수 있기 때문이다. 객체 움직임의 크기는 객체의 움직이는 빠르기를 나타내는 변수로써 움직임 크기가 클수록 빠른 객체 움직임을, 움직임 크기가 작을수록 느린 객체 움직임을 의미한다.

<표 3>은 식 (2)를 통해 얻은 객체 움직임 방향을 45도 범위로, 총 8종류로 구분하여 양자화한 것이다.

이제, 2.2장에서 일례로 든 *OM(12, -4)*를($D = 0.32$, $M = 4.22$) 객체 움직임으로 인덱싱해 보자. 먼저, 양자화를 위한 <표 2>과 <표 3>을 참조하면 *OM*의 방향 d = *'Right'*($\because -\pi/8 \le 0.32 < \pi/8$), 크기 m = *'Slow'*($\because 0 <$

$4.22 < 8\sqrt{2}$)로 양자화됨을 알 수 있다. 따라서, *OM(12, -4)*를 식 (4)로 양자화하여 표현하면, *OM(Slow, Right)*가 된다. 결국, *OM(Slow, Right)*의 움직임 종류에 대한 인덱싱 번호는 <표 1>에 의해 1번이 되며, 이는 (그림 6)의 극 좌표계상에서 1번 위치에 해당된다. 즉, 이 예제에서 한 프레임 내의 객체 움직임은 인덱싱 번호 1로 인덱싱 될 수 있다.

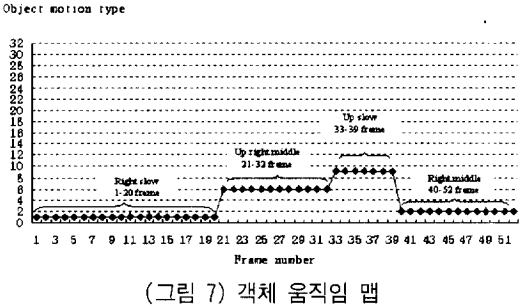
2.4 객체 움직임 맵

이전 장에서 우리는 객체 움직임을 종류별로 인덱싱하는 방법에 대해 설명하였다. 이 장에서는 한 샷 내의 객체 움직임의 변화를 효율적으로 인덱싱하기 위한 '객체 움직임 맵'을 제안하고자 한다.

객체 움직임 맵은 객체가 움직이는 일정 시간동안 극좌표계에 나타난 객체 움직임 종류들을 기록함으로써 얻을 수 있다. 이러한 객체 움직임 맵을 수식으로 표시하면 식 (5)와 같다.

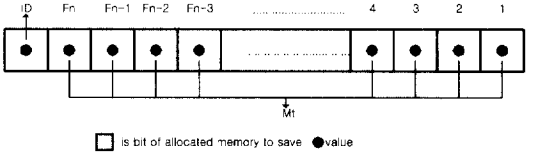
$$\{MAP(ID, t, i);\ 0 \le ID \le m,\ 1 \le t \le Fn,\ 1 \le i \le Mt\} \quad (5)$$

여기서 *MAP(.)*는 객체 움직임 맵을 뜻하며, *ID*는 객체를 구별 짓는 번호, t는 프레임 번호, i는 움직임 종류를 의미하는 정수 값이다. m은 샷 내의 움직이는 객체의 총 개수, *Mt*(motion type)은 양자화된 객체 움직임의 크기와 방향으로 만들 수 있는 객체 움직임 종류에 해당하며, *Fn*(frame number)는 샷을 이루는 프레임 총 개수이다.



(그림 7) 객체 움직임 맵

(그림 7)은 식 (5)를 통해 얻은 객체 움직임 맵과 이를 해석하는 방법이다. (그림 7)은 총 52프레임으로 이루어진 샷에서 한 개의 객체 움직임 변화를 <표 3>에 기반하여 기록하고 있다. (그림 7)의 객체 움직임 맵을 해석하면, 객체는 비디오 시퀀스에서 'Right slow'(1프레임~20프레임), 'Up right middle'(21프레임~32프레임), 'Up slow'(33프레임~39프레임), 'Right middle'(40프레임~52프레임)의 움직임을 가지고 변화하고 있다. 이처럼, 객체 움직임 맵은 시간 변화에 따른 객체 움직임 변화를 한 눈에 알 수 있다. 따라서 특정 프레임 구간, 즉 어떤 샷 동안의 객체 움직임 맵은 비디오 시퀀스에서 객체 움직임을 인덱싱하기 위한 좋은 인덱싱 변수가 된다.



(그림 8) 객체 움직임 맵의 메모리 저장 구조

객체 움직임 맵이 메모리에 저장되는 구조는 (그림 8)과 같다. 식 (5)와 (그림 8)에 의해, 객체 *ID*를 위해 할당되는 메모리 비트 a은 식 (6)과 같이 구할 수 있다.

$$a = \left\lceil \frac{m-1}{2} \right\rceil \quad (6)$$

여기서, $\lceil\ \rceil$는 내부의 연산된 값을 정수 값으로 올림한다는 의미이다. 또한, 움직임 종류를 나타내기 위한 인덱싱 번호에 할당되는 메모리 비트 b는 식 (7)과 같이 구해진다.

$$2^b \ge Mt \quad (7)$$

예를 들어 <표 1>을 보면 본 논문에서 설정한 움직임 종류는 총 32종류의 인덱싱 번호를 지닌다. 즉, 인덱싱 번호를 나타내기 위해 필요한 최소 비트 수 b는 최소한 5비트가 된다. 따라서, 객체 움직임 맵을 인덱싱 벡터로 사용하고자 할 때 요구되는 최대 비트(bit) 크기 N은 식 (8)과 같이 계산된다.

$$N = a + b \times Fn \quad (8)$$

식 (8)에 의해 (그림 7)의 한 샷 전체에 대한 객체 움직임 맵을 저장하기 위해 필요한 비트 수 N은 $\left\lceil \frac{1-1}{2} \right\rceil + 5 \times 52$에 의해 260비트(약 33바이트)로 계산된다. 이는 한 프레임(혹은 인덱스 대상 프레임들)당 이에 해당하는 히스토그램 전체를 저장하기 위해 몇 K바이트씩의 저장 공간을 할당했던 기존의 인덱싱

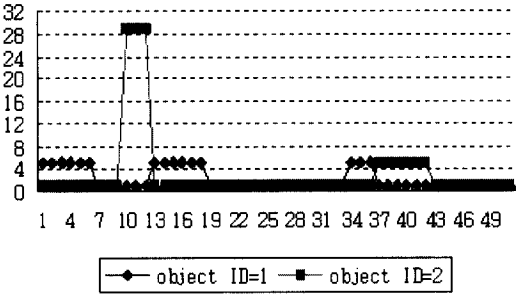
방식에 비해 매우 적은 양의 저장 공간을 요구한다.

다음 장에서는 제안한 객체 움직임 맵을 이용하여 비디오 시퀀스내의 다양한 객체 움직임을 인덱싱한 실험 및 결과를 제시한다

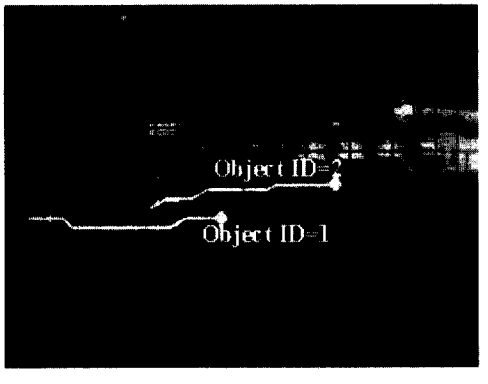
3. 실험 및 결과

실험을 위해 사용한 비디오 시퀀스는 MPEG-7 테스트 시퀀스 중 352×240 크기의 15분 분량의 MPEG-1 비디오 시퀀스이다. 이 시퀀스는 카메라 움직임이 고정된 상태에서 촬영되었으며, 하나 또는 두 종류의 움직이는 객체를 관찰할 수 있다.

실험 순서는 (그림 1)과 같으며 다음과 같이 요약된다. 2장에서 제안된 알고리즘에 의해 샷이 구분된 후, 객체 움직임이 검출되고 극 좌표계에 기반한 객체 움직임 인덱싱 과정을 거쳐 객체 움직임 맵이 생성된다. 이렇게 만들어진 객체 움직임 맵을 기반으로 객체 움직임 궤적을 추정한다.



(a) 객체 움직임 맵



(b) (a)에 의한 객체 움직임 궤적

(그림 9)

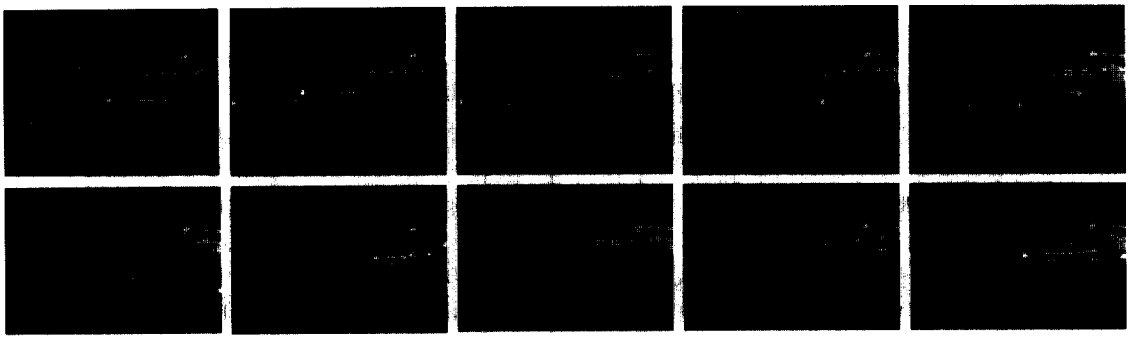
(그림 9)는 2개의 객체 움직임을 포함하는 52프레임으로 구성된 샷에 대해 제안된 알고리즘을 사용한 결과이다. (그림 9(a))는 한 샷 내부(52프레임)에서 객체들이 어떻게 움직이고 있는지를 인덱싱 번호로써 도시하는 객체 움직임 맵이며, (그림 9(b))는 (그림 9(a))의 객체 움직임 맵에 나타난 인덱싱 번호를 (그림 6)의 극 좌표계 방향에 맞추어 시간순서(프레임 번호)에 따라 샷의 첫 프레임에 도시한 객체 움직임 궤적이다.

제안된 인덱싱 변수의 성능평가는 객체 움직임 맵으로 추정한 객체 움직임 궤적이 실제 움직임 궤적과 얼마나 유사한지를 비교함으로써 수행되었다. 실제 객체 움직임 궤적은 비디오 시퀀스를 한 프레임씩 출력시켜 가면서 객체의 움직임 중심점을 연결하여 수동적으로 얻어냈다.

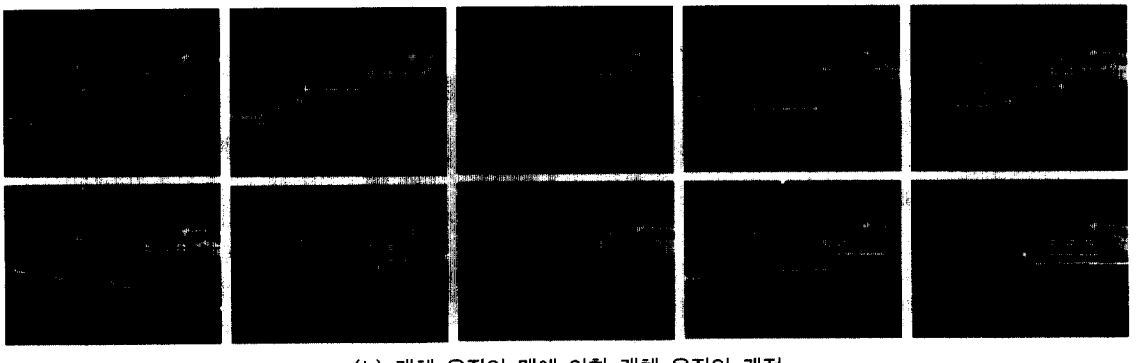
〈표 4〉 객체 움직임 맵의 성능 평가

Shot number	Object ID	Similarity ratio(%)	Memory requirement(bit)
0068~0172	1	99.3	526
0173~0286	1	99.6	571
	2	98.9	
0287~0556	1	92.4	1351
0557~0643	1	84.9	436
0644~1276	1	96.5	3166
	2	94.8	
1277~1471	1	73.8	976
1472~1528	1	99.7	286
	2	98.0	
1529~1819	1	98.6	1456
1820~2164	1	98.6	1726
2165~2320	1	93.9	780
2321~2392	1	99.8	361
	2	98.4	
2393~2782	1	97.3	1951
2783~2791	1	79.1	45
2792~2983	1	97.9	961
	2	91.7	
2984~3000	1	95.6	86
Total average similarity ratio(%)		Total memory requirement(bit)	
94.44		14678	

<표 4>는 객체 움직임 맵으로 추정한 객체 움직임 궤적과 수동적으로 검출한 객체 움직임 궤적과의 유사도 및 객체 움직임 맵을 저장하기 위해 필요한 메모리 요구량을 보여준다. 유사도는 평균 94.44%였다. 평균 유사도 C 계산은 식 (9)와 같이 구해진다. 식 (9)에서, C_d는 방향 유사도, C_m은 크기 유사도, i는 프레임 번



(a) 수동적인 방법에 의한 객체 움직임 궤적



(b) 객체 움직임 맵에 의한 객체 움직임 궤적

(그림 10)

호이다. C_d, C_m는 식 (10), (11)에 의해 얻어진다.

$$C = \frac{\frac{\sum_{i=1}^{N} C_d}{N} + \frac{\sum_{i=1}^{N} C_m}{N}}{2} \times 100 \tag{9}$$

$$C_d \begin{cases} 1.00 , & 0 \le D_d < \pi/4 \\ 0.75 , & \pi/4 \le D_d < \pi/2 \\ 0.50 , & \pi/2 \le D_d < 3\pi/4 \\ 0.25 , & 3\pi/4 \le D_d < \pi \\ 0.00 , & D_d \ge \pi \end{cases} \tag{10}$$

$$C_m \begin{cases} 1.00 , & 0 \le D_m < 8\sqrt{2} \\ 0.75 , & 8\sqrt{2} \le D_m < 16\sqrt{2} \\ 0.50 , & 16\sqrt{2} \le D_m < 24\sqrt{2} \\ 0.25 , & 24\sqrt{2} \le D_m < 32\sqrt{2} \\ 0.00 , & D_m \ge 32\sqrt{2} \end{cases} \tag{11}$$

$$D_d = |M_d - P_d| \tag{12}$$

$$D_m = |M_m - P_m| \tag{13}$$

D_d와 D_m은 방향과 크기에 대한 차분치를 의미하며, M_d와 M_m은 한 프레임씩 영상을 플레이 시켜가며 구한 수동적인 객체 움직임의 방향과 크기에 해당한다. P_d와 P_m은 제안된 객체 움직임 맵으로 구한 방향과 크기이다. 이와 같은 인덱싱 변수들을 저장하기 위해 필요한 저장 공간은 <표 4>에 제시된 바와 같이 총 14678 비트(약 2K 바이트)이다.

(그림 10)은 <표 4>의 결과를 보다 이해하기 쉽게 샷의 객체 움직임 궤적을 샷의 첫 프레임에 표시한 예제들이다. (그림 10(a))는 수동적인 방법에 의한 객체 움직임 궤적을 도시한 것이고, (그림 10(b))는 같은 샷에서 객체 움직임 맵을 이용하여 추정한 객체 움직임 궤적을 보여주고 있다. 보는 바와 같이 (그림 10(a))와 (그림 10(b))가 매우 유사함을 알 수 있다. 즉, 객체 움직임 맵은 비디오 시퀀스에서 실제적인 객체 움직임 궤적을 잘 검출해 내고 있다.

4. 결 론

본 논문은 압축 비디오 시퀀스에서 효율적인 객체 움직임 인덱싱을 위해 극 좌표계의 움직임 종류에 기반한 '객체 움직임 맵'을 제안하였다.

제안된 객체 움직임 맵은 정확한 객체 특징 추출에만 중점을 두어 시간과 저장 공간의 낭비를 초래하며, 사용자가 이해하기 어려운 인덱싱 변수들을 지닌 기존의 방법들과는 달리, 비디오 시퀀스의 압축 영역에서 빠른 특징 추출을 할 수 있으며, 적은 비트 수를 가지고 이를 인덱싱 할 수 있다. 또한, 시간 변화에 따른 객체 움직임의 변화를 한 눈에 알 수 있다. 따라서, 객체 움직임 맵은 압축 영역의 비디오 시퀀스에서 객체 움직임을 효율적으로 인덱싱할 수 있는 좋은 인덱싱 변수라 할 수 있다. 컴퓨터 시뮬레이션과 실험 결과는 이를 잘 뒷받침 해주고 있다.

앞으로는 제안한 객체 움직임 맵의 객체 움직임에 관한 인덱싱 정보를 토대로 하여 객체 추적, 비디오 내용 요약 시스템, 감시 시스템 등의 응용 시스템을 개발하는 것에 관한 추가 연구를 계속 수행하고자 한다.

참 고 문 헌

[1] F. Idris and S. Panchanathan, "Review of Image and Video Indexing Techniques," *Journal of Visual Communication and Image Representation* , Vol.8, No.2, pp.146-166, June 1997.

[2] Christoph Stiller and Janusz Konrad, "Estimating Motion in Image Sequences," *IEEE signal processing magazine*, pp.70-91, JULY 1999.

[3] Zaher AGHBARI, Kunihiko KANEKO, and Akifumi MAKNOUCHI, "A Motion-Location Based Indexing Method for Retrieving MPEG Videos," *IEEE*, pp.102-107, 1998.

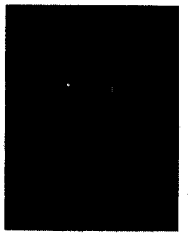
[4] Suchendra M. Bhandarkar, Aparna A. Khombhadia, "Motion-based Parsing of Compressed Video," *IEEE*, pp.80-87, 1998.

[5] Ahme K. Elmagrmid, *Video Data Base System*, pp. 16-32, 1997, Kluner Academic Publishers.

[6] Yong Man Ro and So-Yeon Kim, "Texture extraction for Contents-based Medical Image Indexing," *APCMBE99*, 1999.

[7] Edorado Ardizzone, "Video Indexing using Optical Field," IEEE, pp.831-833, 1996.

[8] M. J. Swain and D. H. Ballard, "Color indexing," *Int. J. Comput. Vision7(1)*, 1991.



김 소 연

e-mail : b613@icu.ac.kr

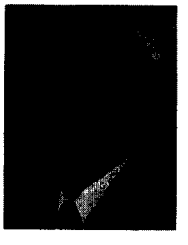
1998년 전남대학교 컴퓨터공학과 졸업(공학학사)

1999년 한국정보통신대학교대학원 졸업(공학석사)

1999년 한국전자통신연구소 (ETRI) 위촉연구원

1999년~현재 한국정보통신대학교대학원 박사과정

관심분야 : 실시간 영상처리, 영상압축과 복원, 멀티미디어 처리 및 MPEG-7, 컴퓨터비전



노 용 만

e-mail : yro@icu.ac.kr

1992년 KAIST 졸업(공학박사)

1995년 UC Irvine 연구원

1996년 UC Berkeley 연구원

1998년~현재 한국정보통신대학교 대학원 부교수

관심분야 : 이미지/비디오 신호처리, MPEG-7, 영상 특징추출, 멀티미디어 시스템, Data hiding