컷 검출을 위한 블록별 히스토그램 비교에 관한 연구

고석만* · 김형균* · 오무송*

A Study on block histogram's comparison for cut detection

Seok-Man Go* · Hyeng-Gyun Kim* · Moo-Song Oh*

요 약

동영상 검색 시스템에서는 사용자가 전체 동영상 정보를 한눈에 파악하고, 필요한 경우 동영상의 원하는 지점부터 직접 재생할 수 있도록 하기 위하여 전체 동영상의 내용을 요약해 놓은 대표 프레임 리스트를 제공하며 대표 프레임 리스트를 작성하기 위하여 장면전환을 정확하게 검출할 필요성이 발생한다. 본 논문에서는 장면전환 지점을 추출하기 위하여 프레임을 일정한 블록으로 분할하고 다음 프레임의 동일 블록에서의 히스토그램 값을 비교하여 임계값을 넘지 못하면 다음 프레임을 컷으로 추출하였다.

Abstract

Video retrieval system must offer representation frame list to do to do play from point that user wants. Representation frame list can get though cut detection point exactly. This paper dismembers frame to fixed block to cut detection point, and compare same block histogram cost of next time frame. If result that compare does not exceed thresold, detect next frame to cutting.

Ⅰ. 서 론

기존의 멀티미디어 검색 방법은 미리 입력된 키워드를 사용한 텍스트를 기반으로 하는 질의 및 접근 방법이 주류를 이루고 있으며 이러한 접근 방식은 텍스트 주석이 각 화상에 첨가되고 텍스트를 검색 질의어로 사용하기 때문에 사용자는 사전에 화상을 기술하는 키워드 범위를 알고 있어야 한다는 점과 나라마다 다양한 언어와 문화적 특성에 종속되는 문제점을 가지고 있다. 따라서 현재 텍스트를 기반으로 하는 화상 검색 시스템의 문제를 극복하기 위한 방법으로 화상의 내용(content)을 특징으로 사용하는 여러 방법들이 연구되고 있다.

일반적으로 동영상 검색 시스템에서는 사용자가 전체 동영상 정보를 한눈에 파악하고, 필요한 경우 동영상의 원하는 지점부터 직접 재생할 수 있도록 하기 위하여 전체 동영상의 내용을 요약해 놓은 대표프레임 리스트를 제공하며 대표프레임 리스트를 작성하기 위하여 장면전환을 정확하게 검출할 필요성이 발생한다. 동영상의 장면 분할을 위한 장면전환 알고리즘으로는 Nagasaka[4]가 제안한 칼라쌍(color-pair)의 경우 화상내의 경계선 사이에서 변화되는 칼라 성분의 차로 히스토그램을 형성하는 방식, Hampapur[9] 및 Shaharay[12] 등이 제안한 모델기반 장면변환기법, Grsky가 제안한 검색대상이 되는 객체의 윤곽선을 백터화하여 화상 데이터의 특징값으로 이용하는 방식, Chua가 제안한 Nagasaka가 제안한 칼라쌍에 유사도

* 조선대학교 컴퓨터공학과

접수일자: 2001. 12. 14

를 첨가한 방식등이 있다. 그러나 Nagasaka가 제안한 방법은 히스토그램의 임계값이 영상의 값의 분포에 따라 변경되는 문제점을 가지고 있고, Hampapur가 제안한 방법은 압축이 안된 원 영상에 적용할 수 없어 MPEG로 압축된 동영상 데이터에 이 방법을 적용하기 위해서는 먼저 디코딩 한 다음 이 방법을 적용함으로 데이터를 처리하는데 시간이 오래 걸리고 메모리 양이 많이 필요한 문제점을 나타내고 있다.

본 논문에서는 동영상 검색의 전처리 과정으로 장면전환점 지점을 추출하기 위하여 프레임을 일정한 블록으로 분할하고 각각의 블록당 ALH(Average of Luminance Histogram)값을 구하여 다음 프레임의 동일 블록에서의 ALH값을 비교하여 임계값을 넘지 못하면 다음 프레임을 컷으로 추출하는 방법을 이용하여 장면전환 지점을 구하는데 보다 효율적인 방법을 제시하고자 한다.

Ⅱ. 관련 연구

비디오 데이터의 효율적인 데이터 베이스화 뿐만 아니라 정확한 분석, 효과적인 검색의 기본이 되는 과정이 비디오 분할(video segmentation)이다. 즉 , 비디오의 내용이 바뀔때마다 해당 내용을 분류하고 인덱싱(indexing)해야 비디오 정보를 체계적으로 데이터베이스화하고 효율적으로 검색할 수 있다. 이 분류의 기본 단위는 주로 카메라 샷(camera shot)에 의해 구분되며, 이를 컷 검출(cut detection)이라 한다. 컷은 대개 두 가지 종류로 분류된다. 첫 번째는 영상의 급격한 변위(abrupt transition), 즉 카메라 브레이크이며, 두 번째는 페이드-인(fade-in), 페이드-아웃(fade-out), 디졸브(dissolve)와 같은 여러 가지 편집 효과에 의한 점진적 변위(gradual transition)이다[4]. 이외에 컷은 아니지만 주밍(zooming), 패닝(panning)과 같은 카메라의 움직임도 고려되어야 할 부분이다. 또한, 비디오 분할은 대상 영상에 따라 크게 비압축 비디오 스트림에 대한 분할과 압축 비디오 스트림에 대한 분할로 구분될 수 있다. 기존의 비압축 비디오 스트림에 대한 장면전환 검출 기법으로는 다음과 같은 방법이 있다. 우선, 가장 간단한 방법은 단순히 두 프레임(frame)간의 대응 픽셀(pixel)을 비교하여 밝기 값이 변화한 픽셀을 양으로 표현하여 이것이 일정한 임계치(threshold)를 초과할 경우 장면 전환으로 판단하는 화소 비교(pixel comparison) 방법이 있다[4]. 이 방법은 영상내의 부분적인 변화에 약한 단점을 가지고 있다. 이 문제를 해결하기 위하여 프레임간의 히스토그램(histogram)의 변화가 일정한 임계치를 초과할 경우 장면 전환으로 판단하는 히스토그램 비교(histogram comparison) 방법이 제안되었다[7][10]. 히스토그램을 이용하는 또 한가지 방법으로 장면 전환이 이루어 질 경우 화소값의 변화가 랜덤하게 분포하므로 프레임간 같은 위치의 픽셀의 밝기값의 차이에 대한 히스토그램을 구하여 이것의 분산이 일정한 임계치 이하일 경우 장면 전환으로 판단하는 화소간 차이의 히스토그램 비교 방법이 있다. 밝기값을 이용한 또 다른 방법으로는 현재 프레임의 밝기 평균과 이전 프레임의 밝기 평균의 차이로 나눈 값이 임계치를 초과할 경우 장면 전환으로 판단하는 평균 밝기 차이를 이용하는 방법이 있다[9]. 지금까지 소개한 방법들은 주로 영상내 밝기값의 분포에만 의존하므로 모양(shape), 색상(color), 질감(texture)과 같은 영상의 내용정보를 반영하지는 못한다. 영상의 내용정보를 장면전환 검출에 이용하기 위하여 윤곽 추출 필터(edge edtector)를 이용하여 에지를 추출한 후 프레임간의 에지의 개수를 비교하여 개수의 차이가 일정한 임계치를 초과할 경우 장면 전환으로 판단하는 윤곽 화소 비교 방법이 있다[13]. 또한, 각 프레임을 블록으로 분할하여 각각 움직임 벡터(motion vector)를 구한 후 현재 프레임에서 큰 값을 갖는 움직임 벡터의 개수와 현재 프래임과 다음 프레임과의 차이값이 큰 움직임 벡터의 개수의 비를 이용하는 움직임 연속성(motion continuity)을 이용한 방법이 있다[14]. 압축 비디오 스트림에 대한 장면 전환 검출 방법에는 DCT DC 계수를 이용한 장면 전환 검출[5], 발생 데이터율에 의한 장면 전환 검출[6], 움직임 정보를 이용한 장면 전환 검출[7]등이 있다.

Ⅲ. 블록별 히스토그램 비교를 이용한 컷 검출

동영상 검색의 전처리 과정으로서 컷 검출은 방대한 양의 데이터에서 찾고자 하는 정보를 쉽고, 빠르게 찾고자 하는데 이용된다. 컷을 검출하는 알고리즘은

지금까지 상당히 많은 방법이 제안되고 있다. 이중에서도 히스토그램 비교법은 전체 히스토그램의 픽셀 누적 포인트를 이용하기 때문에 부분 부분의 변화값들을 적용하기 힘들었다.

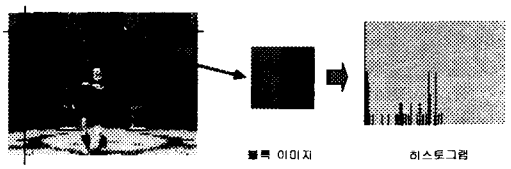

그림 1. 블록별 히스토그램 생성
Fig. 1. Block histogram's creation

따라서 본 연구에서는 프레임을 일정한 블록으로 분할하고 각각의 블록당 ALH(Average of Luminance Histogram)값을 구하여 다음 프레임의 동일 블록에서의 ALH값을 비교하였다. ALH 값은 이미지의 부분 부분을 일정 블록별로 계산을 하기 때문에 점진적인 장면 변화에도 쉽게 적용할 수 있다.

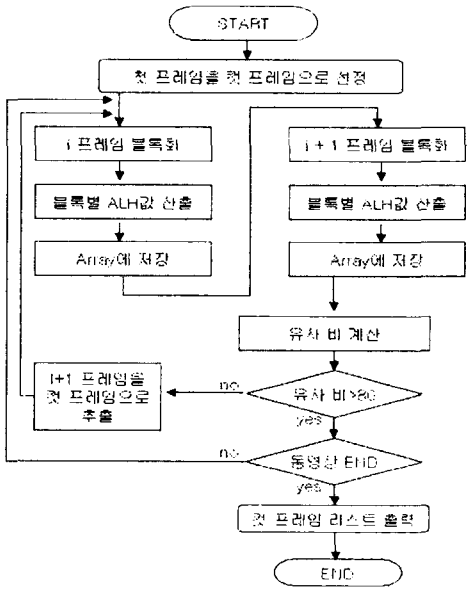

그림 2. 컷 검출 시스템 플로차트
Fig. 2. System flowchart for cut detection

1. 블록화

입력된 영상에서 컷 프레임을 검출하기 위한 첫 단계로서, 프레임을 각각 8*8 블록으로 분할을 하게 된다. 실제적으로 물리적으로 분할을 하는 것이 아니고, 논리적으로 분할을 하게된다. 먼저 입력된 영상의 가로, 세로 크기를 알아내서 블록화를 시킨다.

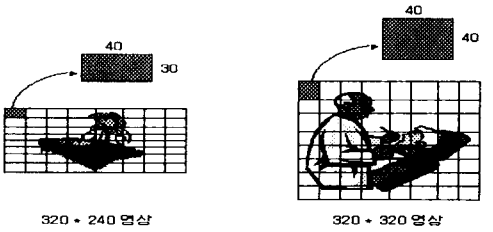

그림 3. 프레임에서 블록 분할
Fig. 3. Division of block in frame

본 논문에서는 정규화된 영상을 이용하지 않고, 입력된 영상의 크기를 알아내어 실제 픽셀을 8*8로 분할한다. 이로서 여러 크기의 영상을 검색에 이용할 수 있다는 이점이 있게된다.

2. ALH 추출

ALH는 프레임을 블록으로 나누어 각각의 블록에 대한 명도 히스토그램 평균값을 나타낸다.

$$ALH = \frac{\sum_{i=1}^{k} M_i}{k}$$

이렇게 구해진 블록의 ALH 값은 배열에 저장된 후, i 프레임의 ALH값과 i+1 프레임의 ALH 값을 비교하게 된다. 비교한 결과를 임계치와 비교하여 일정의 비율을 넘게 되면 다음 프레임을 검사하고, 일정 비율을 넘지 못할 경우 i+1 프레임을 컷 프레임으로 검출한다.

본 연구에서의 초기 임계치는 80을 이용한다. 이것은 다른 여러 컷 프레임 검출 방법에서 실험에 의한 최적의 값을 추출한 것이다. 점진적인 장면 변환점을 검출하기 위해 일정 비율의 유사비가 계속될 경우, 임계치에서 일정 수준의 값을 낮춰 줌으로서 점진적인 장면 변환점에 대응하고 있다.

```
ALH[ ]; // ALH 유사비 배열

While ( != 프레임의 끝) {
  ALHi[ ]; // i 프레임의 ALH값 배열
  ALHj[ ]; // i+1 프레임의 ALH값 배열
  for( k=0; k< 블록의 개수; k++ ) {
      ALHi[k] = 0;
      ALHj[k] = 0;
  } // 배열의 초기화

  for(k =0; k< 블록의 개수 ; k++) {
      if(ALHi[k] > ALHj[k] )
          ALH[k] = ALHj[k] / ALHi[k] ×100;
      else
          ALH[k] = ALHi[k] / ALHj[k] ×100;
          sum = sum + ALH[k];
  }
  유사비 = sum / 블록의 개수;

  if ( 유사비 > 임계값(80) ) {
      임계값 = 임계값 - 0.2;
      if (임계값 <= 77) {
            임계값 = 초기 임계값;
            i+1프레임을 cut 프레임으로 추출;
      }
  }
  else  i+1 프레임을 cut 프레임으로 추출;
}
```

컷 프레임 검색 알고리즘

Ⅳ. 실험 및 결과

사용자가 입력한 질의화상에 대해서 유사한 여러 후보화상을 검색하는 내용기반 화상검색시스템의 성패여부는 화상의 특징을 어떻게 빠르고, 효과적으로 정확하게 추출해내는가 하는 특징추출 부분이다. 따라서, 사용자 인터페이스를 통해 입력된 원화상에 대해 시스템은 특징을 추출하는 모든 과정을 자동으로 수행해야 하는데, 이는 통상적으로 일정시간이 소요되는 화상처리시간의 단축을 위해서도 사용자의 간섭 없이 특징 추출 전과정이 빠르게 진행되어야 하기 때문이다.

다음 표는 본 연구에서 사용한 시스템 사양표이다.

표 1. 시스템 사양표
Table 1. Specification of system

CPU	Pentium III 450Mhz
RAM	128Mb
VGA	Voodoo3 2000
OS	Windows 2000 Professional
SOFTWARE	Visual C++ 6.0

우리가 이용할 수 있는 동영상의 분류는 무한히 많다. 이렇게 많은 동영상 중에서 쉽게 접할 수 있는 애니메이션, 뉴스, 영화를 본 논문에서는 이용할 것이다. 애니메이션은 특수효과를 많이 사용하기 때문에 검색의 효율을 측정해 볼 수 있고, 뉴스는 빠른 장면 변화에 민감하게 반응을 할 수 있는지 알아볼 수 있다. 영화는 사물에 대해 정확히 반응할 수 있는지 알아볼 수 있게 한다. 본 논문에서 애니메이션, 뉴스, 영화 세가지 각각 5종류의 데이터를 가지고 실험을 하였다. 동일한 환경에서의 성능 측정을 하기 위하여 AVI 파일 포맷을 이용하였다.

다음 Table은 본 연구에서 사용한 동영상 재료이다.

표 2. 컷 검출에 이용한 동영상 데이터
Table 2. Animation data that use in cut detection

	프레임의 크기	프레임수	가시 컷 프레임수	파일 포맷
애니메이션	320*240	496	130	AVI
		465	87	
		413	95	
		517	107	
		417	93	
뉴스	320*320	615	254	
		617	217	
		624	196	
		587	237	
		607	274	
영화	320*320	875	139	
		914	157	
		927	243	
		892	195	
		936	217	

그림 4는 컷 검출을 위한 프로그램 창이다.

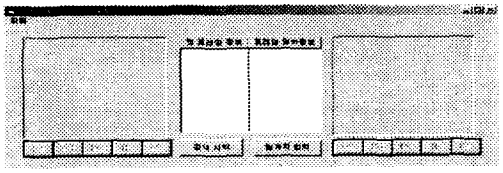

그림 4. 컷 검출 프로그램 화면
Fig. 4. Program screen for cut detection

화면의 좌측에 검색을 위한 동영상이 로드되고, 임계값을 입력한 후 검색 시작 버튼을 누르게 되면 컷 프레임 검출이 시작된다. 이렇게 구해진 컷 프레임은 화면의 중간에 유사비를 포함하여 리스트로 나타나게 된다.

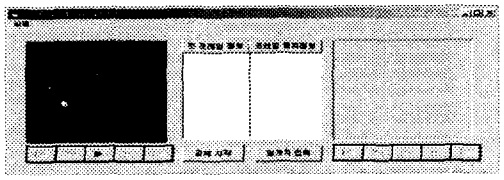

그림 5. 동영상 로드 화면
Fig. 5. Screen that load animation

그림 5는 동영상이 입력된 화면이다. 동영상이 로드되면, 화면의 좌측 창에 영상이 나타나고, 엑티브 무비 컨트롤을 이용하여 컷 프레임 검출을 위한 동영상을 관람할 수 있다.

그림 6은 임계값 입력 화면이다. 이곳에서 사용자는 검색에 이용할 임계값을 입력하게 된다. 임계값을 너무 높게 설정하게 되면 실제 컷 프레임보다 많은 프레임이 검색되게 될 것이며, 너무 낮게 설정하게 되면 추출이 어렵게 된다. 이처럼 임계값의 입력은 컷 프레임 추출에 중요한 요소이다. 본 연구에서 컷 프레임 검출을 위한 임계값으로 80을 이용하였으며, 이것은 매칭 비율을 나타낸다. 이것은 여러번의 실험에 의한 최적의 임계값이다.

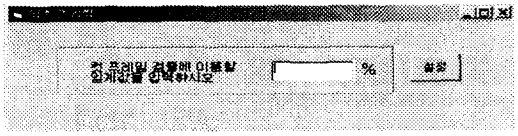

그림 6. 임계값 입력 화면
Fig. 6. Screen that input threshold

그림 7은 컷 검출이 완료된 화면이다. 컷 검출 작업이 끝나게 되면 프레임 리스트가 화면의 중간에 표시된다. 그리고 추출된 프레임의 유사비도 표현된다. 추출된 프레임을 클릭하게 되면 화면의 우측창에 프레임이 로드된다. 그리고, 컨트롤을 이용하여 그 프레임부터 검색할 수 있게 하였다.

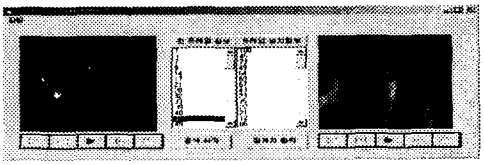

그림 7. 컷 검출 완료 화면
Fig. 7. Screen that complete cut detection

본 연구에서는 세가지 유형의 동영상 데이터에 대하여 각각 5가지의 데이터를 가지고 실험을 하였다. 애니메이션, 뉴스, 영화 각기 화면 구성에 특성을 지니고 있기 때문에 본 연구가 모든 유형의 동영상에 효율적인지 판단하기 위하여 다양한 데이터를 이용하였다. 먼저 각각의 동영상에 대한 컷 프레임 추출률을 비교해 본 후, 정확도까지 비교한다.

다음 표는 각각의 동영상에 대한 컷 프레임 추출률을 나타낸다.

표 3. 컷 검출률
Table 3. The detection rate of cut

추출률	픽셀에 의한 검출	히스토그램에 의한 검출	ALH를 이용한 검출
애니메이션	68.46(89/130)	51.54(67/130)	86.92(113/130)
	66.67(58/87)	58.62(51/87)	86.21(75/87)
	63.16(60/95)	61.05(58/95)	84.21(80/95)
	64.49(69/107)	50.47(57/107)	88.79(95/107)
	68.82(64/93)	55.91(52/93)	91.40(85/93)
뉴스	84.65(215/254)	86.61(220/254)	87.80(223/254)
	82.95(180/217)	85.71(186/217)	88.94(193/217)
	80.61(158/196)	88.78(174/196)	91.84(180/196)
	84.81(201/237)	86.92(206/237)	90.72(215/237)
	82.85(227/274)	85.77(235/274)	84.67(232/274)
영화	82.73(115/139)	90.65(126/139)	89.93(125/139)
	79.62(125/157)	90.45(142/157)	91.72(144/157)
	84.77(206/243)	93.83(228/243)	90.95(221/243)
	81.54(159/195)	88.72(173/195)	86.67(169/195)
	82.95(180/217)	88.94(193/217)	90.78(197/217)

본 연구에서 비교의 대상으로 선정한 픽셀에 의한 검출, 히스토그램에 의한 검출은 현재 컷 프레임 검출을 위한 보편적으로 이용하는 방법이기 때문에, 본 연구에서도 비교의 대상으로 설정하였다. 결과에서 확인할 수 있듯이, 애니메이션 영상에서는 다른 방법에 비해 10%정도 우수한 검출율을 보여주고 있다. 단지, 영화 화면에서의 검출율이 다른 방법과 비슷한 결과를 보이고 있다. 검출 비율은 전체 가시 컷 프레임을 검출된 프레임 수로 나누어서 구하였다.

V. 결 론

컴퓨터의 성능이 발전함에 따라 정보 체계가 문자나 숫자위주의 정보에서 멀티미디어 정보로 변화되었다. 비디오는 멀티미디어 정보의 한 예인데, 현재의 비디오는 특정한 부분을 찾고자 할 때 반복적인 검색과정을 해야 한다. 하지만 디지털화된 비디오는 특정 부분으로 한번에 이동을 할 수 있으므로, 장면이 바뀌는 프레임들을 추출하여 멀티미디어 데이터베이스에 대한 색인으로 구성할 필요가 있다.

본 논문에서는 한 프레임 전체에 대한 화소를 이용하지 않고, 8*8 블록으로 분할하여 각각의 블록에 대해 명도 히스토그램 평균값(ALH)을 구해, 이전 프레임과 다음 프레임의 동일 블록에 대해 ALH 값을 비교하여 유사비를 구하였다. 이렇게 구해진 유사비를 동영상 검색시 사용자가 입력한 임계값과 비교하여 컷 프레임을 검출하였다. 본 연구에서는 세가지 종류의 동영상 데이터를 이용했는데, 동영상도 각각의 내용에 따라 표현하는 방법이 틀리기 때문에 비교의 대상으로 설정하였다. 실제 이전의 방법(픽셀값에 의한 검출법, 히스토그램에 의한 검출법)과 비교 해보았을 때, 검출율에서 많게는 30%정도의 효율적인 결과를 보였으며, 정확도면에서 10%정도의 향상됨을 확인할 수 있었다. 이전의 방법들은 점진적인 장면의 변화에 민감하지 못하였으나, 본 실험에서는 유사비가 일정비율 계속되게 되면, 일정의 상수값을 임계값에서 빼주는 방법을 이용하여, Panning과 Zooming과 같은 점진적인 장면 변화에도 민감하게 반응할 수 있도록 하였다.

비디오 인덱싱을 하기 위해서는 임계치의 설정이 중요하다. 본 논문에서는 실험에 의한 임계치를 구하였기에 임계치의 설정이 실험 결과에 많은 영향을 주었다. 그러므로 이론적인 방법에 의해 임계치를 자동으로 설정할 수 있는 통계적 모델 구축이 필요하다. 그리고 효과적인 비디오 인덱싱을 하기 위해서 단순한 장면전환뿐만 아니라 점진적인 장면전환이나, 카메라 특수 효과 등과 같은 복잡한 장면 전환도 완벽하게 검출하는 방안에 대한 연구가 계속되어야 할 것이다.

참 고 문 헌

[1] C.L. Fennema, and W.B. Thompson, "Velocity Determination in Scene Containing Several Moving Objects", Computer Graphics and Image Processing, Vol. 9, No. 4, 1979, pp. 310-315.

[2] R.C. Jain, "Segmentation of Frame Sequence Obtained by a Moving Observer", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 6, No. 5, 1984, pp. 624-629.

[3] R. Kasturi and R. Jain, "Dynamic Vision", Computer Vision: Principles, Eds. R. Kasturi, R. Jain, IEEE Computer Society Press, Washington, 1991, pp. 469-480.

[4] A. Nagasaka, and Y. Tanaka, "Automatic Video Indexing and Full-Video Search for Object Appearances", Visual Database Systems, II, Eds. E. Knuth, and L.M. Wegner, Elsevier Science Publishers B.V., 1992 IFIP, pp. 113-127.

[5] F. Arman, A. Hsu, and M-.Y. Chiu, "Image Processing on Compressed Data for Large Video Databases", Proc. 1st ACM Intl. Conf. on Multimedia, Anaheim CA, August 1993, pp. 267-272.

[6] S.Y. Lee and H.M. Kao, "Video Indexing - An Object Based on Moving Object and Track", Proc. IS&T/SPIE, Vol. 1908, 1993, pp. 25-36.

[7] H.J. Zhang, A. Kankanhalli, and S.W. Smoliar, "Automatic Partitioning of Full-Motion Video", ACM/Springer Multimedia Systems, Vol. 1, No. 1, 1993, pp. 10-28.

[8] H.J. Zhang, C.Y. Low, Y. Gong, and S.W.

Smoliar, "Video Parsing Using Compressed Data", Proc. IS&T/SPIE, Image and Video Processing II, February 1994, pp. 142-149.

[9] A. Hampapur, R. Jain, and T. Weymouth, "Digital video segmentation", Proceedings of Second ACM Conference on Multimedia, San Francisco, California, 1994, pp.357~364.

[10] Hyeon-Soo Ahn, Il-Seok Oh, "Fast Shot Detection from Video Images using Large and Adaptable Skip Factors", ACCV'95 Second Asian Conference on Computer Vision, December 5-8, Singapore pp. 489~493, 1995.

[11] J. Meng, Y. Juan, S. F. Chang, "Scene Change Detection in a MPEG Compressed Video Sequence", Proc . IS&T/SPIE, Vol. 2419, February 1995.

[12] B. Shahraray, "Scene Change Detection and Content-Based Sampling of Video Sequences", Proc. IS&T/SPIE, Vol. 2419, CA, February 1995.

[13] A.Akutsu et al, "Video indexing using motion vectors", Visual Communications and Image Processing '92, pp.1522-1530, Boston, MA, November 1992, SPIE.

[14] BOON-LOCK YEO and Bede Liu, "Rapid Scene Analysis On Compressed Video", IEEE Trans. on CIRCUITS and SYSTEM for VIDEO TECH, Vol.5, No.6, December 1995.

저 자 소 개

고석만(Seok-Man Go)
1998년 광주대학교 경영학과(경영학사)
1992년 동국대학교 전자계산학과(이학석사)
2001년 조선대학교 컴퓨터공학 박사과정수료
2001년 현재 제주산업정보대학 조교수
관심분야 : 멀티미디어, 영상처리

김형균(Hyeng-Gyun Kim)
1999년 조선대학교 산업대학원 전자계산학과(이학석사)
2001년 현재 조선대학 컴퓨터공학과 박사과정
※관심분야 : 멀티미디어, 영상처리

오무송(Moo-Song Oh)
1965년 조선대학교 전기공학과(공학사)
1968년 조선대학교 전기공학부(공학석사)
1999년 4월~1999년 11월 조선대학교 산업대학원 원장
2001년 현재 조선대학교 컴퓨터공학부 교수
※관심분야 : 멀티미디어, 영상처리