칼라와 모션 정보를 이용한 비디오 구조화 시스템 구현
Implementation of video structuring system using color and motion information

송창준*· **, 고한석** , 권용무*
*한국과학기술연구원 영상미디어연구센터, **고려대학교 전자공학과

요약

본 논문에서는 기존에는 샷 경계 검출에 초점을 맞춘 것과는 달리 본 논문에서는 샷 보다 상위레벨인 비디오 씬 추출에 초점을 맞추어 디지털 비디오를 구조화하는 시스템을 제안한다. 샷간의 유사도를 측정하기 위해서 칼라와 모션 특징을 이용하였으며, 비디오 내의 동적 또는 정적 특성을 반영하기 위해서 적응적 가중치를 적용하였다. 칼라 특징을 추출하기 위해서 각 샷의 내부에서 대표 프레임을 추출하였고, 각 샷 내부의 모션 정보는 MPEG 비디오 모션 벡터를 이용해서 추출하였다. 또한, 비디오 씬 분할 시 연산 시간을 줄이기 위한 기법을 제시하였다. 마지막으로 추출된 비디오 씬에 대해서 성능평가를 하였다.

1. 서 론

최근 들어서 관련 기술의 발달로 디지털 비디오의 사용은 놀랄만한 속도로 증가되고 있다. 그러나 디지털 비디오는 데이터 양이 방대하고, 특정한 구조가 없기 때문에, 사용자가 브라우징 또는 검색을 위해 비디오에 접근하는 일은 그다지 쉬운 일이 아니다.

기존에는 사용자로 하여금 비디오에 접근할 수 있게 하기 위해서 비디오로부터 샷 레벨의 구조 추출에 초점을 맞추어 왔다[2][3]. 그러나 이러한 샷 레벨의 비디오 구조화로는 비디오내의 사건이나 내용을 충분히 반영하지 못한다는 단점과 비디오에 접근하기 위해서 사용자에게 제시되는 데이터의 양이 너무 많다는 단점이 있다. 따라서, 본 논문은 샷 보다 하나 상위 레벨인 씬 분할 방법을 제안한다. 샷간의 유사도를 측정하기 위해서, 칼라 및 모션 특징을 사용하였으며 비디오 내의 동적 또는 정적에 특성에 따른 가중치를 적용하였다. 본 논문은 다음과 같이 구성되어 있다. 2장에서는 최근에 대두가 되고 있는 비디오 구조화 연구에 대해서, 3장에서는 본 논문에서 제안하는 알고리즘을, 4장에서는 제안하는 방법에 대한 실험 결과, 마지막으로 5장에서는 결론 및 향후 과제에 대해서 논할 것이다.

2. 관련 연구

최근 들어 샷 레벨의 비디오 구조화의 문제점을 극복하는 연구가 진행되고 있다. 이들 연구들의 공통점은 비디오 내에서 사건 또는 내용을 전달할 수 있는 씬 구조를 추출한다는 점이다.

비디오 내에서 하나의 씬은 같은 장소(배경)에서 일어나거나 같은 주인공(얼굴, 의상 등)이 나오기 때문에 하나의 씬 내부에서는 샷들간의 유사한 컨텐트(칼라, 모션, 오브젝트 등)를 가지는 샷들이 연속되어 나오거나 반복되어 나온다. 따라서, 위의 컨텐트를 이용해서 샷간의 유사도를 구함으로써 비디오 씬을 분할할 수 있다.

M.M. Yeung and B.L. Yeo[5]는 시간 구속 조건(time-constrained clustering)방법에 의한 스토리 단위로의 비디오 분할을 제안하

였다. Alan Hanjalic, et al.[4]은 한 편의 영화를 분할해서 자동으로 Logical Story 단위로 묶는 방법에 대해 제안했으며, Yong Rui, et al.[6]은 비디오 시퀀스에 대해서 T.O.C(Table-of-Content)를 만드는 방법을 제안했다.

위 논문 모두 씬 분할을 위해서, 하나의 씬 내에는 유사한 칼라 분포를 가지는 샷들로 이루어져 있다는 가정과 유사 샷을 찾는 과정에 있어서 시간 제약성을 이용한다. 시간 제약성을 사용하는 이유는, 현재의 씬에 있는 샷과 다른 씬의 내부에 있는 샷간에서도 유사한 칼라 분포를 가질 수 있기 때문에 두 샷간의 유사도 측정 시 두 샷간의 시간 간격을 고려하는 것이다. 마지막으로 유사한 샷들이 검출되면 두 샷의 내부에 있는 샷들은 하나의 씬에 포함된다.

위 방법들은 샷간의 유사도를 측정하기 위해서 주로 칼라 또는 칼라와 칼라 액티버티 정보를 사용해 왔다. 그러나 기존의 칼라 정보만을 사용해서는 실제 하나의 씬 임에도 불구하고, 유사한 칼라 분포를 지니고 있지 않는 샷들 때문에 좋은 씬 분할 결과를 보여 주지 못한다. 이런 경우는 특히 화면 전환이 빠르게 일어나는 액션 영화에서 흔히 나타난다. 화면 전환이 빠르게 일어나는 구간 내에서는 사건이 일어나고 있는 배경이 빠른 속도로 바뀌기 때문에 유사한 컨텐트를 지니는 샷들이 나타나지 않을 수 있다. 따라서 실제 하나의 씬이 여러 씬으로 분할되는 씬의 과다 분할 현상과 이런 현상에 의해서 검출된 씬이 사건을 충분히 반영하지 못한다는 문제점을 지니고 있다.

3. 제안하는 비디오 구조화 방법

아래 그림 1은 본 논문에서 제안한 비디오 구조 추출에 관한 알고리즘을 나타내고 있다.

3-1. 샷 경계 검출

비디오 분석을 위해서 가장 중요한 초기 작업은 비디오 시퀀스를 샷 단위로 나누는 일이다. 여기서 검출된 샷을 기반으로 대표 프레임 추출, 샷 클러스터링 그리고 씬 분할 등에 이용되기 때문이다.

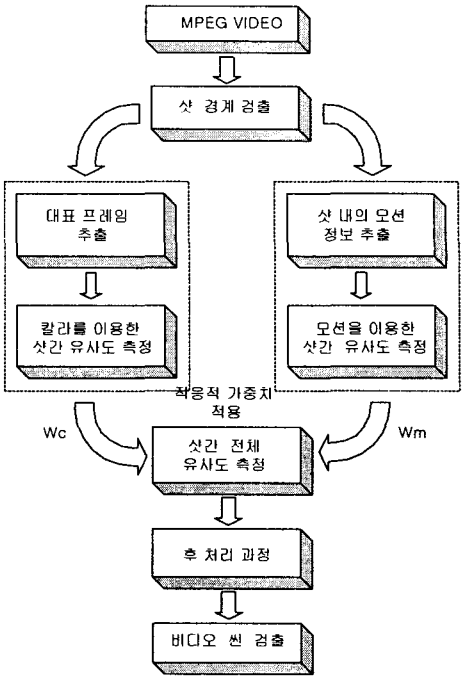

그림 1. 제안하는 비디오 구조 추출 알고리즘

여기서 검출된 각각의 한 샷은 가장 작은 검색 또는 찾는 단위가 된다. 샷 경계 검출은 앞에서도 언급했지만, 기존에 많은 연구가 되어 왔기 때문에 본 논문에서의 샷 경계 검출은 Boon-Lock Yeo[1]가 제시한 방법을 이용한다.

3-2. 샷간의 칼라 유사도 측정을 위한 대표 프레임 추출

샷간의 칼라 유사도를 측정하기 위해서, 각 샷 내의 모든 프레임에 대해서 유사도를 측정할 수 없기 때문에 각 샷을 대표할 수 있는 대표 프레임을 추출하여 이들간의 유사도를 측정함으로써 샷간의 유사도를 대신할 수 있다. 기존의 씬 분할 기법에서는 샷간의 유사도 측정 시, 추가적인 연산량을 줄이기 위해서 처음과 마지막 프레임을 대표 프레임으로 추출하고 이들간의 유사도로 샷간의 유사도를 대신하였다. 그러나 샷 내의 칼라 분포의 변화가 심한 경우는 위와 같은 처음과 마지막 프레임을 가지고는 샷 내의 칼라 분포의 변화를

충분히 반영할 수 없다. 따라서 본 논문에서는 샷간의 유사도 측정 시 샷 내의 칼라 분포의 변화를 충분히 반영하기 위해서, Sub-shot을 추출하고 각 Sub-shot의 중간 프레임을 대표 프레임을 추출하는 방법을 이용한다. 아래 그림 2는 Sub-shot 추출에 대한 방법을 나타낸다.

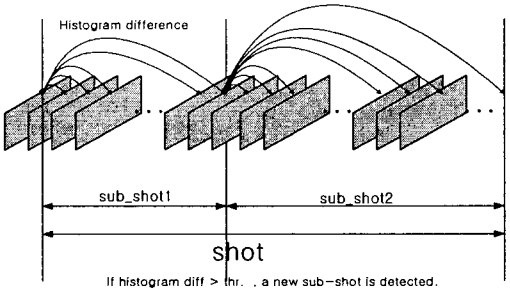

그림 2. Sub-shot을 이용한 대표 프레임 추출

본 알고리즘은 적은 연산량을 가지고 적절히 샷 내의 칼라 분포를 충분히 반영할 수 있다는 장점이 있다. 또한 각 Sub-shot이 가질 수 있는 최소시간 T를 고려함으로써, 하나의 샷 내에서 검출될 수 있는 전체 Sub-shot의 최대 개수를 제한할 수 있다.

3-3. 칼라, 모션 정보를 이용한 샷간의 유사도 측정

샷간의 유사도를 측정하기 위해서, 본 논문에서는 두 개의 특징인 칼라와 모션 정보를 이용한다. 칼라를 이용한 샷간의 유사도는 위에서 추출한 각 샷의 대표 프레임간의 유사도를 구한 후, 가장 큰 유사도 값을 채택한다. 아래 식 (1), (2)는 샷간의 칼라 유사도를 측정하는 방법을 나타낸다.

$$ShotColorSim_{ij} = \max(FrameColorSim_{ij}) \quad (1)$$

$$FrameColorSim_{ij} = \sum_{k=1}^{Bin} \min(Hist_i(k), Hist_j(k)) \quad (2)$$

여기서 i, j 는 각 샷의 대표 프레임의 인덱스를 나타낸다.

각 샷 내의 모션 정보는 샷 내에 있는 P-프레임에 대해서 모션 정보를 추출한다. P-프레임내의 16 by 16 매크로블록 중에서 순방향 매크로 블록 내의 모션 벡터만을 추출하여 x, y 방향에 대해서 절대 평균 변위를 구한 후, 각 방향에 대해서 16으로 나누어 0~1 값으로 정규화한다. 마지막으로 식 (3)을 이용해 평균 모션 크기인 $AvgMotion$을 구한다.

$$AvgMotion = \left((normalized)motion_x^2 + (normalized)motion_y^2 \right)^{1/2} \quad (3)$$

i, j번째의 두 샷간의 모션 유사도는 아래 식 (4)와 같다.

$$ShotMotionSim_{ij} = 1 - |AvgMotion_i - AvgMotion_j| \quad (4)$$

위에서 구한 두 샷간의 칼라와 모션 유사도를 이용하여, 전체적인 샷간의 유사도를 아래 식 (5)와 같이 구할 수 있다.

$$totalShotSim_{ij} = ShotColorSim * W_C + ShotMotionSim * W_m \quad (5)$$

단, $W_C + W_M = 1$ 이다.

만약 샷 i와 j번째의 전체 유사도가 임계값 δ보다 크다면, 두 샷은 같은 컨텐트를 가지고 있다고 볼 수 있기 때문에 하나의 씬에 포함시킬 수 있다.

3-4. 비디오 씬 분할 과정

비디오 씬 분할은 시간 제약 조건을 가지고 유사 샷을 찾는 과정을 통해 가능하다. 유사 샷이 발견되면, 두 샷 내부의 다른 샷들은 하나의 씬에 포함된다. 따라서 유사한 두 개의 샷 내에 있는 샷들은 이미 하나의 씬에 포함되었기 때문에, 추가적인 유사 샷을 찾는 과정은 불필요하다.

M.M. Yeung, et al.[5]와 Yong Rui, et

al.[6]는 비디오 씬 분할을 위해서 모든 샷에 대해서 유사도를 비교하기 때문에 연산 시간 면에서 비효율적이다.

Alan Hanjalic, et al.[4]은 시간 구속 조건을 이용한 유사한 샷끼리의 overlapping links를 이용해 유사한 샷들의 내부에 있는 샷들에 대해서는 별도의 유사도를 측정하지 않는 방법을 제안하였다. 전체 샷 시퀀스가 아닌 유사한 샷끼리 계속 묶어서 연결시켜 나감으로써, 씬을 분할하는 방법이다. 유사한 샷이 더 이상 나오지 않을 때는 그 이전 샷에서부터 뒤로 이동하면서 다시 검색을 해나감으로써, 씬의 경계를 확장시키는 방법이다. 하지만 유사 샷을 찾는데 있어서 정해진 시간 내에서, 순차적으로 찾아 나가기 때문에 불필요한 연산을 하게된다. 예를 들면, 만약 유사 샷의 검색 범위가 10이라 할 때, 현재의 샷과 유사한 샷이 마지막 샷이라면 10번을 검색해야 한다. 하지만 역으로 검색을 해나간다면 한번의 검색을 통해서 유사 샷을 검출할 수 있다. 본 논문에서는 이런 검색 과정을 역으로 해나감으로써 연산량을 줄이려고 노력하였다.

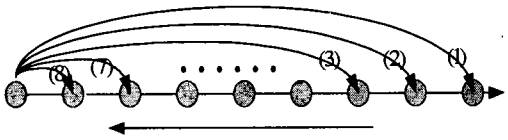

그림 3. 샷 검색 과정

위 그림 3은 검색 범위가 8일 때의 유사 샷 검색 순서를 나타내고 있다. 이런 방법을 통해서 추출된 샷들의 집합을 SceneUnit이라고 명명할 것이다.

3-5. 샷간의 전체 유사도를 구하기 위한 적응적 가중치 적용

위 식 (5)와 같이 전체적인 샷간의 유사도를 구하기 위해서는 칼라와 모션을 통해 구한 샷간의 유사도에 각각 일정한 가중치를 주어야 한다. 비디오 내의 특성을 충분히 반영한 샷 간 유사도를 구하기 위해서는, 비디오 내의 특성에 따라 가중치를 적응적으로 조정해야 한다. 따라서, 현재의 샷과 유사 샷을 찾기 위해서 검색하는 탐색 범위 이내에 있는 샷들에 대해서 식 (6)을 이용해 평균 모션을 구하고, 이를 탐색 범위 구간 내의 비디오 특성으로 사용한다.

만약 평균 모션이 크다면, 이 구간은 액션 씬과 같이 화면이 빠르게 진행되는 동적 구간이기 때문에 모션에 더 큰 가중치를 주게 된다. 반대로 탐색 범위내의 평균 모션이 작다면 이 구간은 정적인 구간이기 때문에 모션보다는 칼라에 더 큰 가중치를 적용하게 된다.

$$W_M = \frac{1}{ShotSearchRange+1} \times \sum_{i=CurrentShot}^{CurrentShot+ShotSearchRange} Motion(i) \quad (6)$$

실제로 비디오를 위의 씬 분할 알고리즘을 적용해 보면, 1~3개의 샷들로 이루어진 SceneUnit이 상당수 검출된다. 그러나 이렇게 소수의 샷들로 이루어진 씬들은 실제로는 사용자에게 제대로 된 사건(내용)을 전달 할 수 없다. 이렇게 과다 분할된 SceneUnit들은 더 이상 칼라 정보를 이용해서는 묶이지 않는다.

따라서 하나의 씬 내에 있는 샷은 유사한 모션 특성을 가진다는 점과 대부분의 비디오에서는 액션이 일어나는 씬과 같이 빠른 사건의 진행을 반영하기 위해서 모션이 큰 샷들이 연속적으로 나열된다는 점을 이용한다. 두 SceneUnit의 경계에 있는 샷들의 모션 유사도가 주어진 임계값보다 크거나, 두 씬의 경계에 있는 샷의 모션이 둘 다 주어진 임계값보다 크다면, 두 SceneUnit을 서로 합친다.

4. 실험 결과

비디오 씬 분할을 위해서 모든 실험은 터미네이터 II MPEG 1 비디오의 후반부(약 1시간)에 대해 이루어 졌다. 초당 30프레임으로

이루어져 있고, 영상 사이즈는 352 ×240이다. 샷 경계 검출 결과 1348개의 샷이 검출이 되었으며, 여기에 제안된 대표 프레임 추출 방법을 적용하였다. 대표 프레임 추출 시 Sub-shot의 최소 지속 시간을 각 샷의 지속시간(duration)의 1/5로 정해 놓음으로써, 추출될 수 있는 대표 프레임의 수를 5개로 제한하였다. 유사 샷 탐색 범위는 10이고, 샷간의 유사도가 0.66이상이면 두 샷은 유사하다 보며, 두 샷 내에 있는 샷들은 하나의 SceneUnit에 포함된다. RGB 각 성분에 대해 8로 양자화하여 칼라 히스토그램을 측정하고, 두 샷간의 유사도를 측정하였다. 후처리 과정에서는 1~3개의 샷들로 이루어진 SceneUnit에 대해서, 씬 경계에 있는 두 샷간의 유사도 0.7이상이거나 두 샷내의 모션이 모두 0.5이상이 되면 두 SceneUnit을 합쳤다.

기존의 순차적인 유사 샷 검색과정과, 제안하는 역과정의 검색과정을 통한 씬 분할 연산시간을 비교해 보았다. 순차적으로 했을 경우는 462.421초의 시간이 걸렸으나, 역으로 검색했을 경우는 321.625초가 걸렸다.

본 알고리즘에 의해 검출된 씬의 성능을 알아보기 위해서 기존에 주로 사용했던 칼라만을 사용했을 경우, 적응적 가중치를 적용했을 경우, 적응적 가중치를 적용한 결과에 후처리 과정을 한 경우에 대한 실험 결과를 비교하였다. 표 1을 통해서 칼라만을 이용해서 SceneUnit을 검출했을 경우는 하나의 사건(내용)을 충분히 반영할 수 없는 1~3개의 샷으로 이루어진 SceneUnit이 대다수 검출되었다. 반면에 비디오 내의 특성을 고려한 적응적 가중치를 적용한 경우에 있어서는 칼라만을 사용했을 경우에 비해 1~3개의 샷으로 이루어진 SceneUnit의 개수가 상당히 줄어듬을 알 수 있다. 또한 적응적 가중치에 후처리 과정을 통해서 상당량의 과다 분할을 줄이는 효과를 가지고 왔다. 따라서, 본 논문에서 제안한 방법 적응적 가중치 적용과 후처리 과정을 적용했을 때 씬의 과다 분할 면에서 가장 우수한 성능을 나타냄을 알 수 있다.

검출된 씬의 객관적 평가를 위해 전체 비디오를 장소별로 나누고 검출된 SceneUnit이 얼마나 장소 내에서 일어나는 사건 또는 내용을 잘 반영하는지 평가하였다. 표 2는 각 장소에 대하여 여러 알고리즘에 의해 검출된 SceneUnit과 실제 사건 반영 관계를 나타낸다. 칼라만을 사용해서 씬 분할을 한 경우는 씬의 과다 분할로 검출된 각 SceneUnit은 제대로 사건을 반영할 수 없음을 알 수 있다. 적응적 가중치를 적용한 경우와 적응적 가중치에 후처리 과정을 한 경우로 갈수록 훨씬 실제 장소에서 일어나는 사건과 검출된 SceneUnit이 더욱더 유사해짐을 알 수 있었다. 가장 씬 분할이 잘되지 않는 11번째 장소를 표 2에서 보면, 이 사실을 잘 알 수 있다.

5. 결론 및 향후 과제

본 논문에서는 비디오 내의 사건이나 내용을 반영할 수 있는 씬 구조를 추출할 수 있는 새로운 방법을 제안하였다. 샷간의 정확한 칼라 유사도를 측정하기 위해서, 각 샷 내의 칼라 변화를 충분히 반영할 수 있는 대표 프레임을 추출하여 이용하였다.

또한 씬 분할 시 적응적 가중치 적용을 통한 모션 정보를 추가 사용함으로써, 칼라만으로 씬 분할을 했을 경우보다 씬 과다 분할을 줄일 수 있었으며, 이를 통해서 비디오내의 내용을 충실히 반영할 수 있었다. 또한 후처리 과정을 통해 1~3개로 이루어진 실제로 하나의 사건이나 내용을 반영할 수 없는 SceneUnit을 이웃 SceneUnit과 서로 합치는 과정을 통해

씬의 과다 분할을 상당히 줄일 수 있었다.

또한 유사 샷 검색 시 역으로 검색함으로써, 연산시간을 줄일 수 있었다.

지금까지 추출된 비디오 샷, 씬 구조는 사용자로 하여금 비디오를 브라우징하거나 검색하기 위해서 기존의 샷 단위가 아닌 사건 또는 내용을 포함하고 있는 씬 단위로의 접근을 가능하게 하는 장점이 있다.

향후 과제는 추출된 비디오 구조를 이용하여, 비디오 브라우징 시스템을 구현하는 일이다.

참고 문헌

[1] Boon-Lock Yeo. Efficient processing of compressed images and video. PhD thesis, Princeton University, 1996.

[2] H. J. Zhang, A. Kankanhalli, and S. W. Smoliar, "Automatic partioning of full-motion video," *Multimedia Systems*, vol.1, pp.10-28, July. 1993.

[3] H. Zhang, et. al., "An Integrated system for content-based video retrieval and video browsing," pp.643-658, Sep. 1995.

[4] Alan Hanjalic, Reginald L. Lagendijk, "Automated Hight-Level Movie Segmentation for advanced Video-Retrieval Systems," *IEEE Transactions on Circuits & Systems for Video Technology* , Vol.9 No.4 , pp 580-588 , June. 1999.

[5] M.M. Yeung and B.L. Yeo,"Time-constrained Clustering for Segmentation of Video into Story Units," *International Conference on Pattern Recognition*, Vol. C, pp. 375-380, August. 1996.

[6] Yong Rui, Thomas S. Huang, and Sharad Mehrotra, "Constructing Table-of-Content for Videos," *ACM Multimedia Systems Journal, Special Issue Multimedia Systems on Video Libraries*, Sept, 1999.

표 1 . 각 알고리즘에 의한 segmentation 결과 비교

	칼라만 사용	제안하는 방법 #1 적응적 가중치 적용	제안하는 방법 #2 적응적 가중치+후처리
전체 SceneUnit의 개수	236	51	24
1~3개의 샷으로 이루어진 SceneUnit	204	31	4

표 2 . 장소별 SceneUnit의 사건 반영

Location Number	Manually Scene Segmentation	칼라만 사용 (C)	제안하는 방법 #1 (A)	제안하는 방법 #2 (AP)
1	1-83	C1-C6 (1-83)	A1-A2 (1-76)	AP1-AP2 (1-76)
2	84-105	C7-C12 (84-106)	A3-A5 (77-105)	AP3-AP4 (77-105)
3	106-122	C13-C15 (107-118)	A6-A7 (106-117)	AP5 (106-117)
4	123-128	C16 (119-182)	A8 (118-285)	AP6 (118-285)
5	129-285	C17-C22 (183-285)	A8 (118-285)	AP6 (118-285)
6	286-290	C23 (286-290)	A9 (286-290)	AP7 (286-290)
7	291-351	C24-C38 (291-351)	A10-A12 (291-342)	AP8 (291-341)
8	352-357	C39-C44 (352-357)	A13 (343-398)	AP9 (342-398)
9	358-638	C45-C86 (358-625)	A14-A24 (399-560)	AP10-AP15 (399-560)
10	639-943	C87-C106 (626-943)	A25-A29 (561-943)	AP16 (561-943)
11	944-1348	C107-C236 (944-1348)	A30-A51 (944-1348)	AP17-AP24 (944-1348)