H.264|AVC 비트스트림에서의 움직임 객체 추적

*유원상 **호우아리 사비린 ***김문철

한국정보통신대학교

*wsyou@icu.ac.kr

Moving Object Tracking in H.264/AVC bitstream

*Wonsang You **M.S. Houari Sabirin ***Munchurl Kim

Information and Communications University

요약

T-DMB BIFS 데이터 방송은 방송 AV 콘텐츠에 부가데이터를 연결하여 방송함으로써 대화형 방송 서비스를 가능하게 하고 있다. T-DMB 비디오 콘텐츠에서 움직이는 객체 부분에 부가데이터를 연결함으로써 시청 중에 사용자 Interaction을 통한 움직임 객체 영역의 정보 소비를 위한 응용을 고려할 때에, 저작단계에서 움직임 영역을 정의하고 이를 추적하는 도구가 필요하다. 본 논문에서는 H.264|AVC 비디오에 대해 효율적인 움직임 객체 영역 추적을 완전 부호화를 수행하지 않고 비트스트림에서 부호화 정보를 이용하여 움직임 영역을 추적하는 방법을 제안한다. 제안된 방법은 움직임 정보 및 부분적으로 복원된 텍스쳐 정보를 사용하여 객체의 특성에 따라 적응적으로 객체의 특성점을 추적함으로써 빠른 처리 속도와 정확한 추적을 동시에 가능하게 한다. 실험을 통하여, 제안하는 방법이 움직임 정보만을 사용한 방법의 처리 속도와 유사하면서도 정확한 추적 성능을 보일 뿐만 아니라 다양한 유형의 객체에 대한 적응적인 추적이 가능함을 확인하였다.

Ⅰ. 서론

기존의 MPEG-4가 콘텐츠를 여러 객체 단위로 분리하는 객체 기반 부호화(Object-based coding)를 지향하는 반면, H.264|AVC는 콘텐츠를 단지 블록 단위로 분리하여 처리하며 비디오 객체를 직접적으로 다루지는 않는다. 그러나 최근 T-DMB BIFS 데이터 방송 서비스에서는 사용자가 직접 객체에 대한 정보에 접근할 수 있어야 한다. 즉, 사용자는 자신이 선택한 객체가 무엇이며 어떻게 움직이는지 등의 정보를 획득하게 된다. 이러한 부가 정보들은 부가 콘텐츠 제작 시스템에 의하여 MPEG-7 메타데이터의 형태로 패키징되어 단말에 전송될 수 있다. 이 시스템에서 핵심적인 요소 중의 하나는 객체의 위치 정보를 생성하는 객체 추적 기술이다.

일반적으로 객체 추적 기술은 대부분 픽셀 정보를 사용하는 방식이지만 H.264|AVC 동영상에서는 압축 영역에 포함된 블록 단위의 움직임 정보 또는 잔차 신호의 정수변환 정보를 이용하여 효율성을 높일 수 있다. 많은 연구자들이 이러한 정보를 이용하여 MPEG 동영상을 위한 객체 추적 기술을 연구해 왔다. Babu와 Ramakrishnan은 공간적으로 보간된 움직임 벡터로부터 기대치 최대화 알고리즘(Expectation Maximization Algorithm)에 의하여 객체를 분리하는 방법을 사용하였다[1]. Treetasanatavorn은 베이지안 예측 방법(Bayesian Method)을 사용하여 주어진 움직임 필드로부터 객체 영역을 분리하였다[2]. 그러나 이러한 방법들은 사용자가 직접 선택한 객체를 추적하기 보다는 프레임을 여러 개의 객체로 분리하기 때문에 처리시간이 길다는 단점이 있다. 한편, Aggarwal은 움직임 벡터로부터 객체 대상 영역의 위치를 예측하고 배경제거에 의하여 추천 객체를 선별한 후, 히스토그램 비교를 통하여 표적 객체를 선택하는 방식을 제안하였다[3]. 또한 Zeng 등은 블록 기반 MRF 모델을 사용하여 H.264|AVC 비트스트림에서 직접 객체를 분리하는 알고리즘을 제안하였다[8]. 이 방법들은 처리시간은 짧지만 움직임이 빠르게 변화하는 객체의 추적에는 적합하지 않다.

따라서, 사용자가 직접 선택한 객체를 빠르게 추적할 뿐 만 아니라 객체의 특성에 따라 적응적으로 추적함으로써 정확도를 높일 수 있는 새로운 기술이 필요하다. 본 논문에서는 부분 복원 및 신경망 회로(Neural Network)를 이용한 적응적 고속 객체 추적 방법을 제안한다. 제안 방법은 특징점 기반 객체 추적 방식(Feature-based Object Tracking)으로서, 사용자가 선택한 객체의 여러 특징점을 빠르고 정확하게 추적한다. 먼저 H.264|AVC 비트스트림에 포함된 움직임 벡터로부터 각 특징점의 움직임을 예측하고, 이 예측 지점으로부터 일정한 탐색 범위에서 더욱 정밀한 위치를 찾는다. 이를 위하여 특징점 주변 영역의 텍스쳐, 이웃 특징점과의 상대적인 위치, 그리고 순방향 움직임 벡터의 신뢰성에 따라 특징점 예측 위치의 정확성을 판단한다. 각 판단 요소는 비유사성 에너지에 의하여 정량화된다. 에너지의 계산을 위해서 단지 특징점 주변 영역만 부분적으로 복원하기 때문에 계산량의 증가가 적다. 한편, 최적의 특징점 위치는 동적 프로그래밍(Dynamic Programming)에 의하여 각 에너지의 총합을 최소화시키는 배열로 선택된다. 각 에너지에 대한 가중치는 신경망 회로에 의하여 움직임이나 형태가 다양하게 변화하는 객체에 따라 최적값으로 자동 갱신된다. 마지막으로 객체의 위치는 각 특징점의 중심에 의하여 결정된다.

본 논문의 구성은 다음과 같다. 2장과 3장에서는 제안된 적응적 고속 객체 추적 알고리즘을 설명하고, 4장에서는 실험 결과를 통하여 제안된 알고리즘의 성능을 검증하며, 5장에서는 연구결과에 대한 결론을 제시한다.

Ⅱ. 역방향 움직임 벡터의 순방향 매핑

사용자에 의하여 선택된 객체의 특징점들이 연속되는 프레임에서 어느 지점으로 이동할 것인지 예측하기 위하여, H.264|AVC 비트스트림에 포함된 움직임 벡터를 사용할 수 있다. H.264|AVC 동영상에서 각 프레임은 다양한 크기의 블록으로 구분되어 있으며 움직임 벡터는 각 블록마다 할당되어 있다. H.264|AVC 동영상에서 P 프레임의 움직임 벡터는 역방향으로 되어 있는데, 객체 추적을 위해서는 이 벡터를 순방향으로 전환하여야 한다. Porikli와 Sun에 따르면, 순방향 움직임 벡터 필드는 다음과 같은 방법에 의하여 구성할 수 있다[4]. 먼저, 다양한 크기의 블록마다 할당되어 있는 움직임 벡터를 4×4 블록 단위로 균일하게 할당한다. 다음에는 그림 1과 같이 현재 프레임의 각 블록을 움직임 벡터에 따라 이전 프레임에 매핑한 후, 겹쳐지는 블록의 집합을 추출한다.

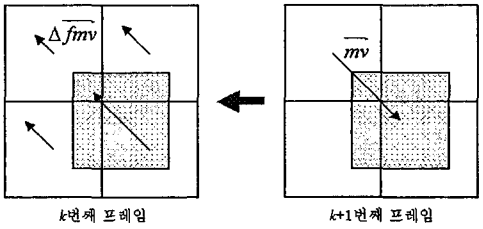

그림 1. 순방향 움직임 벡터의 할당

움직임 벡터는 겹쳐지는 면적의 비율에 따라 겹쳐지는 각 블록에 역방향으로 분산되어 할당된다. k번째 프레임의 j번째 4×4 블록 $b_{k,j}$이 $k-1$번째 프레임의 i번째 4×4 블록 $b_{k-1,i}$와 겹친다고 가정하자. 이때, 겹쳐지는 면적을 $S_{k-1}(i,j)$이라 하면, $i,j=1,2,...,N$일 때 블록 $b_{k,j}$의 역방향 움직임 벡터 $\overrightarrow{mv_k}(b_{k,j})$에 대하여 블록 $b_{k-1,i}$의 순방향 움직임 벡터 $\overrightarrow{fmv_{k-1}}(b_{k-1,i})$는 다음 식 (1)과 같다.

$$\overrightarrow{fmv_{k-1}}(b_{k-1,i}) = -\sum_{j=1}^{N}\left(\frac{S_{k-1}(i,j)}{16} \cdot \overrightarrow{mv_k}(b_{k,j})\right) \quad (1)$$

본 논문에서 고려하는 동영상은 I 또는 P 프레임만을 포함하는 베이스라인 프로파일로 부호화되어 있다고 가정한다. 즉, 하나의 GOP는 하나의 I 프레임과 여러 P 프레임으로 구성된다. 마지막 P 프레임을 제외한 나머지 프레임에서는 위 방법에 의하여 순방향 움직임 벡터 필드를 구성할 수 있다. $k-1$번째 프레임이 마지막 P 프레임인 경우 다음 I 프레임이 움직임 벡터를 가지고 있지 않기 때문에 다음 식 (2)와 같이 순방향 움직임 벡터를 계산한다. 즉, 움직임의 속도가 일정하다는 가정 하에 움직임 벡터의 방향을 반대로 하여 순방향 움직임 벡터로 할당한다.

$$\overrightarrow{fmv_{k-1}}(b_{k-1,i}) = -\overrightarrow{mv_{k-1}}(b_{k-1,i}) \quad (2)$$

순방향 움직임 벡터 필드를 구성한 후, 특징점을 포함하는 블록의 순방향 움직임 벡터에 의하여 다음 프레임에서 각 특징점의 위치를 예측할 수 있다. 즉, $k-1$번째 프레임에서 특징점 n의 위치 벡터가 $\overrightarrow{f_{k-1,n}}=(fx_{k-1,n},fy_{k-1,n})$이고 이 특징점이 i번째 블록 $b_{k-1,i}$에 포함되어 있다면, k번째 프레임에서 해당 특징점의 예측 위치 벡터 $\overrightarrow{p_{k,n}}=(px_{k,n},py_{k,n})$는 다음 식 (3)과 같이 계산한다.

$$\overrightarrow{p_{k,n}} = \overrightarrow{f_{k-1,n}} + \overrightarrow{fmv_{k-1}}(b_{k-1,i}) \quad (3)$$

Ⅲ. H.264|AVC 비트스트림에서의 움직임 객체 추적

순방향 움직임 벡터는 4×4 블록마다 할당되어 있기 때문에 이 벡터에 의하여 예측된 특징점의 위치는 정확하지 않다. 따라서 예측 위치 주변의 탐색 지점에서 더욱 정확한 위치를 찾는 과정이 필요하다. k번째 프레임의 특징점 n의 경우, 예측된 위치 $\overrightarrow{p_{k,n}}=(px_{k,n},py_{k,n})$를 중심으로 $(2M+1)\times(2M+1)$ 정사각형의 탐색 영역을 설정하고 탐색 영역 내의 각 후보 위치가 최적인지 아닌지 여부를 결정한다. 이러한 결정을 위하여 텍스쳐와 형태의 유사성 그리고 순방향 움직임 벡터의 신뢰성이 중요한 척도가 된다. 텍스쳐의 유사성이란 이전 프레임의 특징점 인근 텍스쳐 영역에 대한 후보 위치 인근 텍스쳐 영역의 유사성을 의미한다. 형태의 유사성이란 이전 프레임의 특징점 연결망에 대한 후보 위치 연결망의 유사성을 의미한다. 순방향 움직임 벡터의 신뢰성이란 순방향 움직임 벡터에 의하여 예측된 위치에 대한 신뢰성을 나타낸다. 이를 정량적으로 측정하기 위하여 각 특징점마다 텍스쳐 비유사성 에너지, 형태 비유사성 에너지 및 움직임 비유사성 에너지를 계산한다. 특징점 위치의 최적 배열은 위 에너지의 합을 최소화하는 배열로 선택된다.

1. 텍스쳐 비유사성 에너지

텍스쳐의 유사성은 텍스쳐 비유사성 에너지에 의하여 정량화된다. k번째 프레임의 (x,y) 좌표에 대한 휘도 성분값을 $s_k(x,y)$라 하자. k번째 프레임의 특징점 n에 대하여 $(2M+1)\times(2M+1)$ 탐색 영역 내의 후보 위치의 집합을 $C_{k,n}=\{\overrightarrow{c_{k,n}}(1),\overrightarrow{c_{k,n}}(2),...,\overrightarrow{c_{k,n}}(L)\}$ $(L=(2M+1)\times(2M+1))$이라 하면, 특징점 n의 i번째 후보 위치 $\overrightarrow{c_{k,n}}(i)=(cx_{k,n}(i),cy_{k,n}(i))$에 대하여 다음 식 (4)와 같이 텍스쳐 비유사성 에너지 E_C를 정의할 수 있다. 여기서, W는 후보 위치에 대한 이웃 픽셀의 최대 간격을 나타낸다.

$$E_C(k;n,i) = \frac{1}{(2W+1)^2}\sum_{x=-W}^{W}\sum_{y=-W}^{W}\left|s_k(x+cx_{k,n}(i),y+cy_{k,n}(i)) - s_{k-1}(x+fx_{k-1,n},y+fy_{k-1,n})\right| \quad (4)$$

텍스쳐 비유사성 에너지가 작을수록 후보 위치의 인근 텍스쳐 영역은 이전 프레임의 특징점 인근 영역과 유사하다고 생각할 수 있다. 이와 같이 텍스쳐 비유사성 에너지를 적용함으로써 가능한 한 인근 영역의 텍스쳐가 유사한 위치로 특징점의 위치를 결정할 수 있도록 한다. 그림 2는 탐색 영역의 설정과 텍스쳐 비유사성 에너지를 계산하는데 적용되는 특징점 인근 영역을 보여준다.

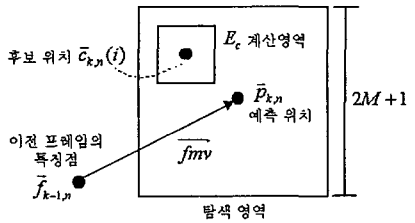

그림 2. 텍스쳐 비유사성 에너지의 계산

한편, 처리 시간을 줄이기 위하여 텍스쳐 비유사성 에너지를 계산하는데 필요한 영역의 휘도 성분만을 부호화된 비트스트림으로부터 부분적으로 복원한다. H.264|AVC에서는 매크로블록 단위로 부호화가 이루어지기 때문에 P 또는 B 프레임에서는 원하는 특정 블록만 복원

할 수 있다. 그러나 I 프레임에서는 각 블록이 주위 블록을 이용하여 화면내 예측 부호화되기 때문에 부분 복원이 불가능하다.

P 프레임에서 특정 블록을 복원하기 위해서는 참조 프레임에 있는 여러 개의 참조 블록을 복원하여야 하며, 이 참조 프레임이 P 프레임인 경우 이 참조 블록 역시 이전 프레임에서 관련되는 참조 블록들을 복원하여야 한다. 이러한 역방향 복원은 디코더 상에서 많은 처리시간을 요하기 때문에 효율적이지 못하다.

따라서, 각 프레임에서 복원하여야 할 영역을 미리 예측함으로써 부분 복원의 속도를 향상시킬 수 있다. k번째 프레임이 P 프레임인 경우 이 프레임의 부분 복원 영역을 예측하기 위해서, k번째 프레임으로부터 해당 GOP의 마지막 프레임까지 특징점의 움직임 속도가 k-2번째 프레임의 순방향 움직임 벡터와 동일하게 일정하다고 가정한다. k번째 프레임부터 GOP의 마지막 프레임 사이에 존재하는 i번째 프레임 (i=k,k+1,...,K)에 대하여, 특징점 n의 가능한 모든 후보 위치의 텍스쳐 비유사성 에너지를 계산하는데 필요한 픽셀의 집합을 예측 탐색 영역 $P_{k,n}(i)$이라 정의하자. 탐색 영역의 예측 오차를 Y라고 하면 예측 탐색 영역의 최대 간격 $T_{k,i}$는 $T_{k,i}$=(i-k+1)×M+W+Y가 된다. 이때, 예측 탐색 영역 $P_{k,n}$ (i)는 k번째 프레임에서의 모든 가능한 후보 위치를 고려하여 다음 식 (5)와 같이 얻어진다. 여기서 $b(\overrightarrow{f_{k-2,n}})$은 n번째 특징점 $\overrightarrow{f_{k-2,n}}$를 포함하는 블록을 나타내며, $\overrightarrow{m}$은 탐색 영역의 중심에 대한 각 상대적인 위치를 나타낸다.

$$P_{k,n}(i)=\{\vec{p}\,|\,\vec{p}=(i-k+1)\overrightarrow{fmv_{k-2}}(b(\overrightarrow{f_{k-2,n}}))+\vec{m} + \overrightarrow{f_{k-1,n}}, \vec{m}=(x_m,y_m); x_m,y_m=-T_{k,i},...,T_{k,i}\} \quad (5)$$

또한 i번째 프레임의 예측 탐색 영역 $P_{k,n}(i)$를 복원하기 위하여 k번째 프레임에서 복원되어야 할 텍스쳐 블록의 집합 $D_{k,n}(i)$는 k-1번째 프레임의 움직임 벡터로부터 다음 식 (6)과 같이 얻을 수 있다.

$$D_{k,n}(i)=\{b(\vec{d})\,|\,\vec{d}=(i-k)\overrightarrow{mv_{k-1}}(b(\overrightarrow{f_{k-1,n}}))+\vec{p} ,\vec{p}\in P_{k,n}(i)\} \quad (6)$$

특징점이 F개 존재한다고 가정하면, 최종적으로 k번째 프레임에서 복원되어야 할 전체 텍스쳐 블록의 집합 D_k는 다음 식 (7)과 같다.

$$D_k=\bigcup_{n=1}^{F}\bigcup_{i=k}^{K} D_{k,n}(i) \quad (7)$$

그림 3은 GOP가 'IPPP'인 구조의 첫 번째 P 프레임에서 부분 복원이 수행되는 구조를 보여준다.

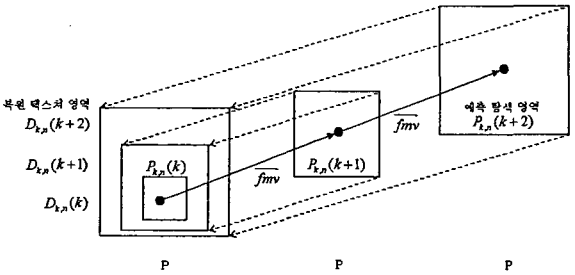

그림 3. 부분 복호화 개념도

부분 복원 과정에서 주의할 점은 부분 복원 영역을 예측하는데 걸리는 시간이 GOP 구조의 길이에 비례하여 길어진다는 점이다. 더구나 GOP 구조의 길이가 길어질 경우, 하나의 GOP 내에서 특징점의 움직임 속도가 일정하다고 가정하였기 때문에 부분 복원 영역의 예측은 더욱 정확도가 떨어진다. 따라서, 위와 같은 방법은 GOP가 5개 이하의 프레임으로 구성된 동영상에 적합하다.

3. 형태 비유사성 에너지

형태의 유사성은 형태 비유사성 에너지에 의하여 정량화된다. 이를 위하여 각 특징점은 그림 4와 같이 단일 선으로 연결된다. 최초의 특징점을 선택한 후 아직 연결되지 않은 특징점 중에서 가장 가까운 특징점을 연결한다. 이와 같은 방법으로 객체 추적을 시작하는 첫 번째 프레임에서 모든 특징점을 순차적으로 연결하여 특징점의 1차원 연결망을 구성한다.

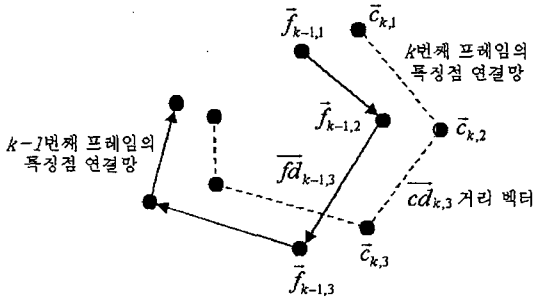

그림 4. 형태 비유사성 에너지의 계산

k번째 프레임에서 각 후보 위치의 형태 비유사성 에너지를 계산하는 방법은 다음과 같다. 먼저, k-1번째 프레임에서 각 특징점 $\overrightarrow{f_{k-1,n}}$ (n=1,2,... ,F)이 첫 번째 프레임에서 연결한 순서대로 정렬되어 있다고 가정하자. 즉, n번째 특징점 $\overrightarrow{f_{k-1,n}}$에 연결된 특징점은 n-1번째 특징점 $\overrightarrow{f_{k-1,n-1}}$이 된다. 이때, 특징점 $\overrightarrow{f_{k-1,n}}$은 이웃 특징점 $\overrightarrow{f_{k-1,n-1}}$에 대해서 거리 벡터 $\overrightarrow{fd_{k-1,n}}$ = $\overrightarrow{f_{k-1,n}}$ - $\overrightarrow{f_{k-1,n-1}}$를 가진다. 마찬가지로 k번째 프레임에서 n-1번째 특징점의 후보 위치 인덱스가 j인 경우, k번째 프레임에서 n번째 특징점의 i번째 후보 위치는 거리 벡터 $\overrightarrow{cd_{k,n}}(i)$ = $\overrightarrow{c_{k,n}}(i)$ - $\overrightarrow{c_{k,n-1}}(j)$를 가진다. 그림 4는 이전 프레임과 현재 프레임의 특징점 연결망에 따라 거리 벡터가 어떻게 설정되는지 보여준다. 이 거리 벡터로부터 k번째 프레임에서 특징점 n (n>0)의 i번째 후보 위치에 대한 형태 비유사성 에너지 E_F를 다음 식 (8)과 같이 정의할 수 있다. 첫 번째 특징점 (n=0)의 경우, $E_F(k;0,i)$=0이 된다.

$$E_F(k;n,i)=\| \overrightarrow{cd_{k,n}}(i)-\overrightarrow{fd_{k-1,n}} \|^{1/2} \quad (8)$$

형태 비유사성 에너지가 작을수록 이웃 특징점에 대한 후보 위치의 형태적 변화가 적다고 생각할 수 있다. 이와 같이 형태 비유사성 에너지를 적용함으로써 가능한 한 형태의 변화가 적은 위치로 특징점의 위치를 결정할 수 있도록 한다.

4. 움직임 비유사성 에너지

특성점에 대한 여러 후보 위치를 비교할 때 텍스쳐와 형태가 모두 유사하여 최적 위치를 찾기 어려운 경우가 발생할 수 있다. 이러한 경

우, 순방향 움직임 벡터에 의하여 예측된 위치의 신뢰성에 따라 최적 위치를 판별할 수 있다. 즉, 순방향 움직임 벡터에 의하여 예측된 위치가 신뢰할 만하다면 후보 위치가 예측 위치에서 멀수록 비유사성이 크다고 판단한다. 이러한 과정은 움직임 비유사성 에너지에 의해서 정량화된다.

Fu 등에 따르면, 순방향 움직임 벡터의 신뢰성은 다음과 같이 순방향 움직임 벡터와 역방향 움직임 벡터를 비교함으로써 측정할 수 있다[5]. k번째 프레임에서 특징점 n의 예측 위치 $\overline{p_{k,n}}=(px_{k,n},py_{k,n})$는 이전 프레임의 i번째 블록 $b_{k-1,i}$에 포함되어 있는 특징점 $\overline{f_{k-1,n}}=(fx_{k-1,n},fy_{k-1,n})$의 순방향 움직임 벡터 $\overline{fmv_{k-1}}(b_{k-1,i})$에 의하여 식 (3)으로부터 계산된다. 한편, 이 예측 위치 $\overline{p_{k,n}}$이 k번째 프레임의 j번째 블록 $b_{k,j}$에 포함되어 있다면, 이 위치로부터 역방향 움직임 벡터 $\overline{mv_k}(b_{k,j})$를 얻을 수 있다. 두 벡터를 비교하여 다음 식 (9)와 같이 순방향 움직임 벡터의 신뢰성을 정의할 수 있다. 여기서 σ는 두 벡터의 차분에 대한 신뢰도 값의 편차를 조정하는 상수이다.

$$R(\overrightarrow{p_{k,n}})=\exp\left(-\frac{\| \overrightarrow{fmv_{k-1}}(b_{k-1,i})+\overrightarrow{mv_k}(b_{k,j}) \|^2}{2\sigma^2}\right) \quad (9)$$

즉, 순방향 움직임 벡터에 의해서 이동한 지점이 역방향 움직임 벡터에 의해서 원래 위치로 되돌아온다면, 순방향 움직임 벡터의 신뢰성이 높다고 판단할 수 있다. 그림 5는 신뢰성이 높은 순방향 움직임 벡터와 신뢰성이 낮은 순방향 움직임 벡터를 보여준다.

Fu가 이러한 신뢰성 척도를 사용하여 객체의 테두리 추적을 위한 움직임 에너지를 정의한 것과 유사한 방법으로[5], k번째 프레임에서 특징점 n의 i번째 후보 위치 $\overline{c_{k,n}}(i)$에 대한 움직임 비유사성 에너지 E_M을 다음 식 (10)과 같이 정의할 수 있다.

$$E_M(k;n,i)=R(\overrightarrow{p_{k,n}}) \| \overrightarrow{c_{k,n}}(i)-\overrightarrow{p_{k,n}} \| \quad (10)$$

특징점 예측 위치의 신뢰성이 낮을수록 움직임 비유사성 에너지는 최적 위치를 찾는데 큰 영향을 주지 못한다. 반면, 특징점 예측 위치의 신뢰성이 높다면 예측 위치에 대한 후보 위치의 거리에 따라 움직임 비유사성 에너지 값이 크게 변화하게 된다. 즉, 특징점 예측 위치의 신뢰성이 높은 경우, 움직임 비유사성 에너지가 텍스쳐 또는 형태 비유사성 에너지에 비하여 최적 위치를 찾는데 더욱 큰 영향을 준다.

5. 비유사성 에너지의 최소화

각 특징점의 후보 위치는 앞서 언급한 세 가지 에너지- 즉, 텍스쳐 비유사성 에너지, 형태 비유사성 에너지, 그리고 움직임 비유사성 에너지를 지니고 있다. k번째 프레임에서 특징점 n의 i번째 후보 위치가 지닌 비유사성 에너지 $E_{k,n}(i)$는 위 세 가지 에너지로부터 다음 식 (11)과 같이 정의된다. 여기서, $\omega_T(k)$, $\omega_F(k)$, 그리고 $\omega_M(k)$는 각각 k번째 프레임에서 텍스쳐 비유사성 에너지, 형태 비유사성 에너지, 그리고 움직임 비유사성 에너지의 가중치를 나타낸다.

$$E_{k,n}(i)=\omega_T(k)E_T(k;n,i)+\omega_F(k)E_F(k;n,i)+\omega_M(k)E_M(k;n,i) \quad (11)$$

특징점 후보 위치의 배열을 $I=\{\overline{c_{k,1}}(i_1),\overline{c_{k,2}}(i_2),...,\overline{c_{k,F}}(i_F)\}$라 할 때, k

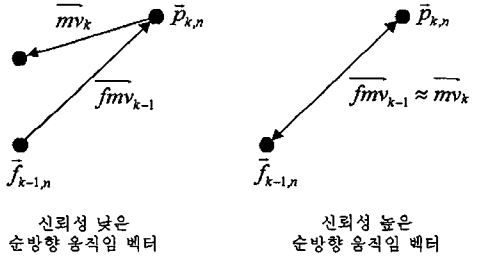

그림 5. 순방향 움직임 벡터의 신뢰성

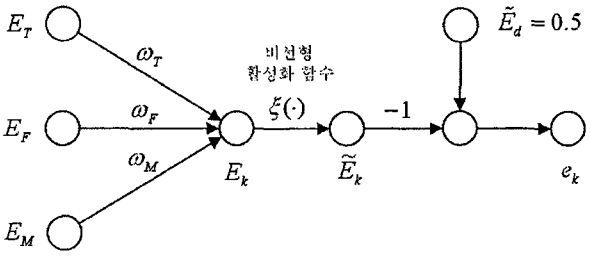

그림 6. 신경망 회로에 의한 가중치 갱신

번째 프레임에서 특징점 후보 위치의 최적 배열 $I_{opt}(k)$는 다음 식 (12)와 같이 전체 비유사성 에너지의 합 $E_k(I)$를 최소화하는 배열로 선택된다.

$$E_k(I)=\sum_{n=1}^{F}E_{k,n}(i_n) \quad (12)$$

특징점 후보 위치 배열의 모든 경우에 대하여 전체 비유사성 에너지의 합 $E_k(I)$를 계산할 경우 $\Theta((2M+1)^{2F})$만큼의 시간이 소요된다. 즉, 탐색 영역의 크기 또는 특징점의 개수가 증가할 경우 특징점 후보 위치의 최적 배열을 찾는데 많은 시간이 필요하다. 반면, 이산 다단계 결정 과정- 즉, 동적 프로그래밍을 사용할 경우 처리 시간을 $\Theta(F)$로 최소화할 수 있다[6]. 동적 프로그래밍을 이용한 비유사성 에너지 최소화 과정은 다음과 같다.

1) 각 후보 위치의 누적 비유사성 에너지를 첫 번째 특징점부터 순차적으로 계산한다. 특징점 n의 i번째 후보 위치의 누적 비유사성 에너지 $E_{local}(n,i)$는 다음 식 (13)과 같이 계산한다. 첫 번째 특징점의 누적 비유사성 에너지는 $E_{local}(0,i) = E_{k,0}(i)$가 된다. 또한, 특징점 n-1의 후보 위치 중에서 누적 비유사성 에너지를 최소화하는 연결점의 인덱스를 다음 식 (14)와 같이 $s(n,i)$로 저장한다.

$$E_{local}(n,i)=\min_j\left(E_{k,n}(i,j)+E_{local}(n-1,j)\right) \quad (13)$$

$$s(n,i)=\arg\min_j\left(E_{k,n}(i,j)+E_{local}(n-1,j)\right) \quad (14)$$

2) 마지막 특징점에서 누적 비유사성 에너지가 가장 작은 후보 위치를 최적 위치로 선택하고 다음 식 (15)와 같이 귀납적으로 각 특징점 n의 최적 위치를 결정한다. 여기서, o_n은 n번째 특징점의 최적 위치에 대한 인덱스를 나타내며 최적 위치는 $\overline{f_{k,n}} = \overline{c_{k,n}}(o_n)$이 된다.

$$o_F=\arg\min_i\left(E_{local}(F,i)\right) \text{ and } o_n=s(n+1,o_{n+1}) \quad (15)$$

6. 적응적인 가중치 결정

사용자가 추적하고자 하는 객체는 다양한 특성을 지닐 수 있기 때문에, 식 (11)에 나타난 비유사성 에너지의 가중치를 객체의 특성에 따라 적응적으로 결정할 필요가 있다. 가령, 동영상 내에서 형태의 변화가 적은 객체의 경우 형태 비유사성 에너지의 가중치를 상대적으로 크게 설정할 필요가 있다. 마찬가지로, 텍스쳐의 변화가 적은 객체라면 컬러 비유사성 에너지의 가중치를 상대적으로 크게 설정하여야 한다. 반면에 형태나 텍스쳐의 변화가 매우 큰 경우, 움직임 비유사성 에너지의 가중치를 크게 설정하여 순방향 움직임 벡터의 신뢰성에 따라 특징점의 이동 위치를 결정할 수 있도록 하여야 한다.

그러나 수동적인 가중치 할당은 추적상의 오류를 야기할 수 있기 때문에, 본 논문에서는 신경망 회로를 사용하여 객체의 특성에 따라 매 프레임마다 자동적으로 가중치를 갱신하는 방법을 사용한다. 그림 6은 비유사성 에너지의 가중치를 갱신하는 신경망 회로의 구조도이다. 가중치에 의해서 합산된 k번째 프레임에서의 비유사성 에너지 E_k는 비선형 활성화 함수 ξ에 의하여 출력값 $\tilde{E}_k$를 갖게 된다. 이상적인 출력값 $\tilde{E}_d$는 0.5이므로, 역전파 알고리즘(Backpropagation Algorithm)에 의하여 다음 식 (16)과 같은 출력값 에러의 제곱 ε_k를 최소화하는 가중치로 갱신한다.

$$\epsilon_k = \frac{1}{2}\left(\tilde{E}_d - \tilde{E}_k\right)^2 \tag{16}$$

단극성 S형 함수(Unipolar Sigmoidal Function)를 비선형 활성화 함수로 사용할 경우($\xi(x)=1/(1+e^{-x})$), k번째 프레임에서 갱신되는 가중치의 변화량은 다음 식 (17)과 같다[7]. 여기서, ω_x는 ω_T, ω_F, 또는 ω_M을 나타내고, E_x는 E_T, E_F, 또는 E_M을 나타낸다. η는 업데이트 상수(Learning Constant)이다.

$$\Delta\omega_x(k) = \eta\left(0.5 - \tilde{E}_k\right)\tilde{E}_k\left(1 - \tilde{E}_k\right)E_x(k) \tag{17}$$

Ⅳ. 실험 및 고찰

제안한 알고리즘의 성능을 평가하기 위하여 다양한 객체를 추적한 위치 데이터를 추출하였다. 실험에 사용된 동영상은 CIF 크기의 'Stefan'과 'Coastguard'를 사용하였다. 각 동영상은 H.264|AVC 베이스라인 프로파일을 기반으로 'IPPP'의 GOP 구조로 부호화되었으며, P 프레임은 직전 프레임만 참조 프레임으로 사용하도록 되어 있다.

먼저 객체가 느리게 움직이는 동영상인 'Coastguard'의 객체 추적 결과가 그림 7에 나타나 있다. 이 동영상에서 초기에 4개의 특징점이 배의 몸체에 선택되었으며, 배가 움직이는 방향에 따라 정확하게 추적하고 있음을 볼 수 있다. 배의 전체적인 형상이 크게 변화하지 않기 때문에 특징점의 연결망도 거의 일정한 형태를 유지하고 있다. 다음으로, 객체가 빠르게 움직이는 동영상인 'Stefan'의 실험 결과가 그림 8에 나타나 있다. 이 동영상에서는 테니스 선수가 매우 빠르게 움직이기 때문에 초기에 선택한 3개 특징점 연결망의 형태가 크게 변화하지만 초기에 선택한 위치를 정확하게 추적하고 있음을 확인할 수 있다. 특히 그림 8(d)처럼 팔 부위에 있는 특징점이 관중들의 색과 유사한 경우, 텍스쳐의 유사성보다는 이웃 특징점에 대한 상대적 위치의 유사성이 객체의 최적 위치를 결정하는 더욱 중요한 척도가 된다.

그림 9는 두 동영상의 객체 추적에 대한 수치적인 데이터를 나타낸다. 'Coastguard'의 경우, 텍스쳐, 형태 및 움직임 비유사성 에너지가 'Stefan'에 비해서 상대적으로 적다. 이것은 이 동영상이 상대적으로 텍스쳐, 형태 또는 움직임의 변화가 적다는 사실을 의미하며, 객체 추적이 'Stefan'에 비해서 더욱 원활하게 이루어지고 있음을 보여주고 있다. 그림 9(d)는 'Coastguard' 동영상에서 순방향 움직임 벡터의 신뢰성을 나타낸다. 이 동영상에서 순방향 움직임 벡터 신뢰성의 평균은 93.9%로서 'Stefan'에 비해서 12.2% 더 높았다. 실제로 'Coastguard' 동영상에서 객체의 움직임은 거의 일정하며 중간에 화면 이동이 일어나는 정도라는 사실로 볼 때, 움직임 비유사성 에너지는 객체의 움직임 특성을 반영하는 효과적인 척도임을 알 수 있다.

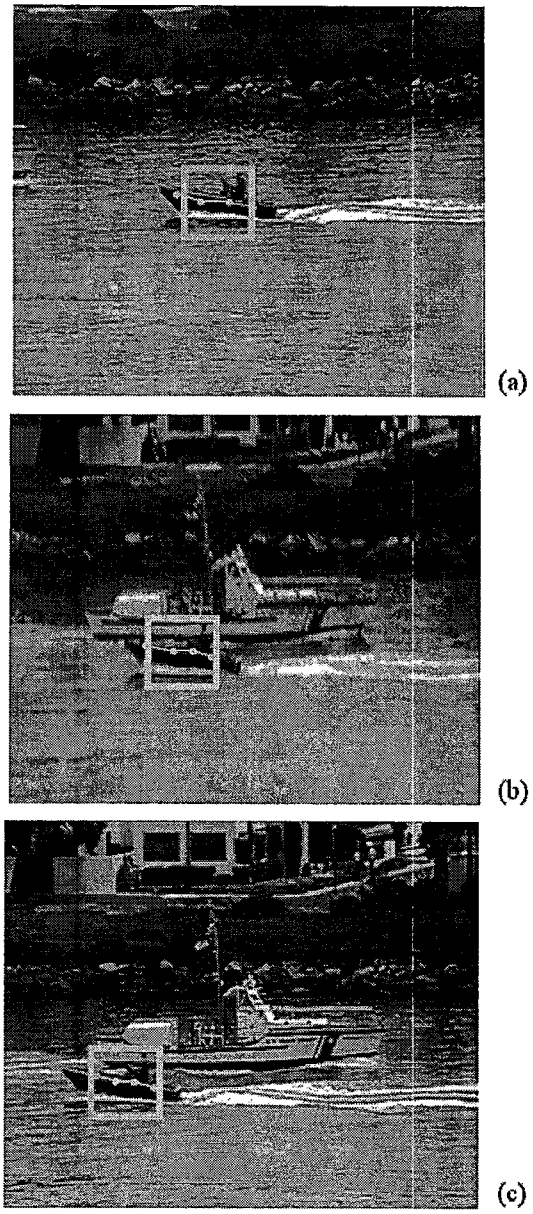

그림 7. Coastguard 영상에서의 객체 추적 (a) 초기 객체의 선택 (b) 70번째 프레임 (c) 85번째 프레임

신경망 회로가 수행되면서 프레임에 따라 비유사성 에너지 에러의 제곱은 최소화된다. 업데이트 상수를 5로 하였을 때, 실험 결과 두 동영상 모두 15번째 프레임 이후부터 에러는 거의 0에 가까운 값을 가지게 되며 비유사성 에너지의 가중치는 최적값을 가지게 된다. 그림 9(c)는 'Coastguard' 동영상의 비유사성 에너지 가중치가 매 프레임마다 자동 갱신되면서 최적값으로 수렴하게 되는 과정을 보여주고 있다.

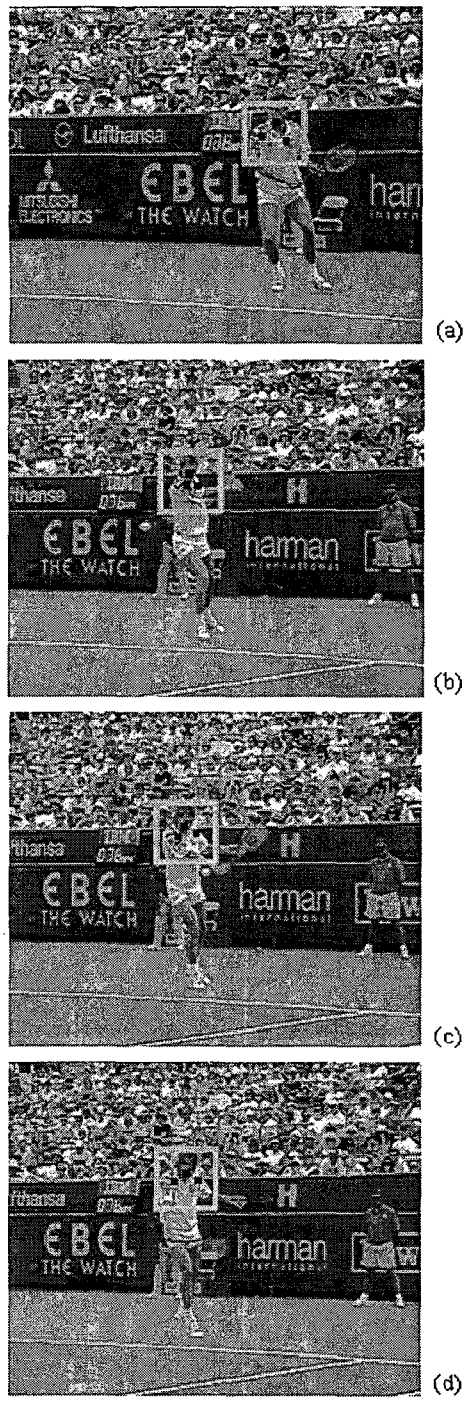

그림 8. Stefan 동영상에서의 객체 추적 (a) 초기 객체의 선택 (b) 16번째 프레임 (c) 18번째 프레임 (d) 20번째 프레임

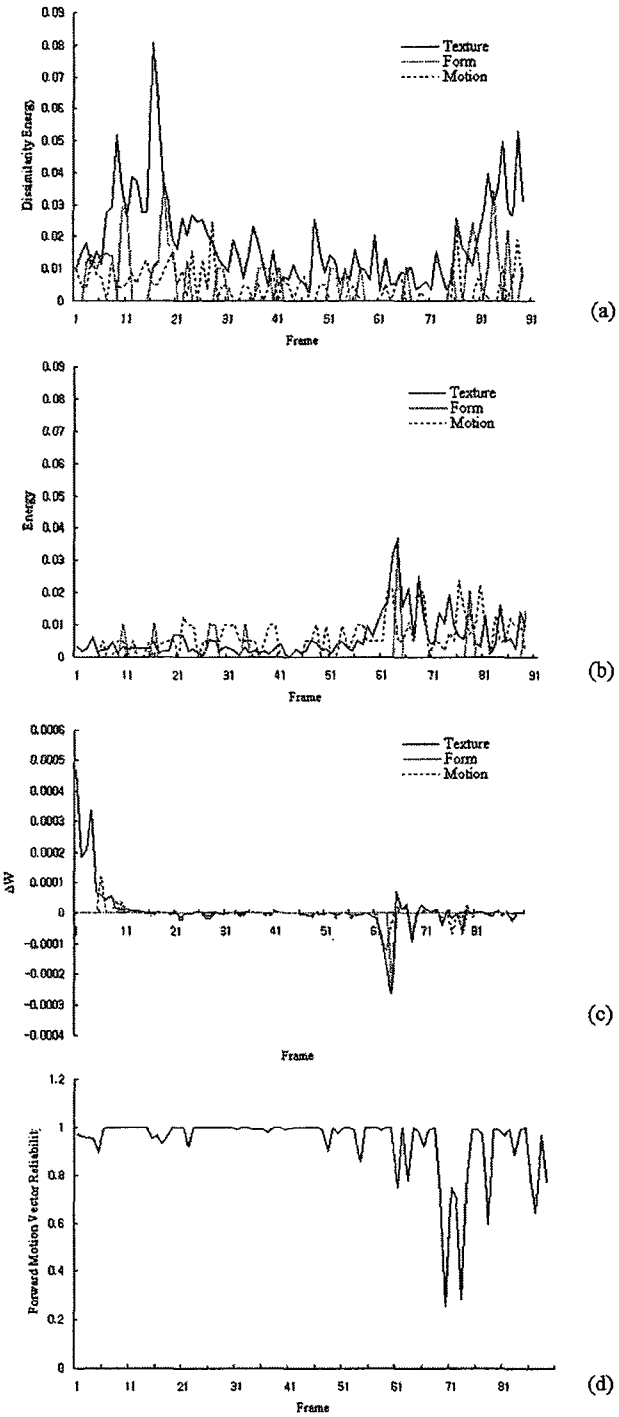

그림 9. (a) 'Stefan'의 비유사성 에너지 (b) 'Coastguard'의 비유사성 에너지 (c) 비유사성 에너지 가중치의 변화량 (d) 순방향 움직임 벡터의 평균 신뢰성

여기서, 'Coastguard' 동영상의 61번째 프레임에서 66번째 프레임 사이에 가중치의 절대값이 갑자기 커지는 현상에 주목할 필요가 있다. 이것은 그림 7(b)와 같이 하얀 색의 배가 접근하기 때문에 생기는 현상으로, 배가 접근함에 따라 가중치가 적응적으로 조절됨을 확인할 수 있다. 그림 9(b)에서 이 부분의 비유사성 에너지가 급증하는 현상도 이를 증명한다.

탐색 범위의 간격 M을 10으로 하고, 텍스쳐 비유사성 에너지 계산 영역의 간격 W를 5로 하였을 때, 제안한 알고리즘의 계산 시간은 H.264|AVC의 동영상 비트스트림을 읽어 들이기 위하여 JM 참조 소프트웨어를 사용하였을 때 대략 프레임당 430ms가 소요된다(Intel Pentium 4 CPU 3.2GHz 1GB RAM 기준). 그러나 성능이 좋은 다른 디코더를 사용할 경우, 최대 230ms까지 소요시간을 줄일 수 있다. 한편, H.264|AVC 동영상에서 움직임 벡터 정보만을 사용하여 객체 추적을 수행하는 Zeng의 알고리즘은 동영상의 특성에 따라 프레임당 30ms에서 최대 700ms까지의 시간이 소요된다[8]. 이 알고리즘이 복잡한 동영상에서 평균 450ms 정도의 시간이 소요된다는 실험결과에 비추어 볼 때, 움직임 벡터만을 사용한 방법과 유사한 속도를 보이고 있음을 알 수 있다.

결과적으로, 제안한 알고리즘은 움직임 벡터만을 사용한 기존의 알고리즘에 비하여 처리 속도는 유사하면서도 텍스쳐 또는 형태가 변하거나 움직임이 빠른 객체도 더욱 정확하게 추적한다.

V. 결론

본 논문에서는 H.264/AVC 비트스트림에서 움직임 객체의 특징점을 빠르고 정확하게 추적하는 새로운 방법을 제안하였다. 이 방법은 텍스쳐, 형태 및 움직임 특성을 복합적으로 분석함으로써 유사성이 가장 높은 특징점의 이동 위치를 찾는다. 텍스쳐 정보의 부분적 복원과 동적 프로그래밍을 통한 위치 예측의 최적화를 통하여 추적 성능을 향상시키면서도 복원에 소요되는 시간을 최소화한다. 또한 신경망 회로를 통하여 알고리즘의 주요 파라미터들이 동영상 또는 객체의 특성에 따라 적응적으로 최적화된다. 본 논문에 제안된 알고리즘은 움직임 벡터 정보만을 사용한 알고리즘과 유사한 처리 시간을 소요하면서도 형태가 변하고 움직임이 빠른 객체도 정확하게 추적하는 성능을 가지고 있음을 실험결과를 통하여 확인하였다. 이 제안 방법은 실시간으로 객체의 위치 정보를 생성하는 부가 콘텐츠 저작 도구에 직접적으로 응용 가능할 것으로 판단된다. 향후 과제로는 H.264/AVC 움직임 벡터 정보 및 부분 복원된 텍스쳐 정보를 이용하여 객체의 주요 특징점을 자동으로 추출하는 방법에 관하여 연구할 예정이다.

참 고 문 헌

[1] R. V. Babu and K. R. Ramakrishnan, "Video Object Segmentation: A Compressed Domain Approach," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, pp. 462-474, Apr. 2004.

[2] S. Treetasanatavorn, U. Rauschenbach, J. Heuer, and A. Kaup, "Bayesian Method for Motion Segmentation and Tracking in Compressed Videos," DAGM 2005, LNCS 3663, pp. 277-284, 2005.

[3] A. Aggarwal, S. Biswas, S. Singh, S. Sural, and A. K. Majumdar, "Object Tracking Using Background Subtraction and Motion Estimation in MPEG Videos," ACCV 2006, LNCS 3852, pp. 121-130, 2006.

[4] Fatih Porikli and Huifang Sun, "Compressed Domain Video Object Segmentation," Technical Report TR2005-040 of Mitsubishi Electric Research Lab, 2005.

[5] Y. Fu, T. Erdem, and A. M. Tekalp, "Tracking Visible Boundary of Objects Using Occlusion Adaptive Motion Snake," *IEEE Trans. Image Processing*, vol. 9, pp. 2051-2060, Dec. 2000.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.

[7] R. O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification*. New York: John Wiley & Sons, 2001.

[8] W. Zeng, J. Du, W. Gao, Q. Huang, "Robust moving object segmentation on H.264/AVC compressed video using the block-based MRF model," *Real-Time Imaging*, vol.11, issue 4, pp. 290-299, 2005.