내용 기반 코딩을 위한 강력한 에지 연결에 의한 움직임 객체 자동 분할

(Automatic Moving Object Segmentation using Robust Edge Linking for Content-based Coding)

김 준 기 † 이 호 석 ††
(Jun Ki Kim) (Ho Suk Lee)

요 약 움직임 객체 분할은 내용 기반 응용을 위하여 핵심적인 것이다. 다중 프레임 차이 누적은 프레임 차이 정보를 누적하여 움직임 에지를 검출한다. 검출된 움직임 에지와 분할될 현재 프레임의 에지를 비교하여 움직임 객체 에지를 생성한다. 그러나 실시간 카메라로 입력되는 연속 동영상의 움직임 객체 에지에는 객체 색과 배경 색의 일치 혹은 객체의 움직임 감소로 말미암아 에지 단락이 발생한다. 에지 단락은 매우 심각한 문제로서 움직임 객체의 영상 품질을 심하게 손상시키는 경우도 있다. 본 연구에서는 강건하고 포괄적인 에지 연결 알고리즘을 개발하여 이 문제를 해결하였다. 또한 본 연구에서는 자동 움직임 객체 분할 알고리즘을 개발하여 분명하고 깨끗한 모양의 움직임 객체를 자동으로 분할하였다. 개발한 알고리즘은 CIF 영상을 초당 30 프레임 이상 처리할 수 있다. 본 논문에서 개발한 알고리즘은 MPEG-4 내용 기반 코딩 시스템에 적용할 수 있다.

키워드 : 강력한 에지 연결, 자동 객체 분할, 내용 기반 코딩

Abstract Moving object segmentation is a fundamental function for content-based application. Moving object edges are produced by matching the detected moving edges with the current frame edges. But we can often experience the object edge disconnectedness due to coincidence of similarity between the object and background colors or the decrease of movement of moving object. The edge disconnectedness is a serious problem because it degrades the object visual quality so conspicuously. That it sometimes makes it inadequate to perform content-based coding. We have solved this problem by developing a robust and comprehensive edge linking algorithm. And we also developed an automatic moving object segmentation algorithm. These algorithms can produce the completely linked moving object edge boundary and the accurate moving object segmentation. These algorithms can process CIF 30 frames/sec in a PC. These algorithms can be used for the MPEG-4 content-based coding.

Key words : robust edge linking, automatic object segmentation, content-based coding

1. 서 론

움직임 객체 분할은 입력된 비디오 동영상의 연속으로부터 움직임이 있는 객체의 형상을 추출하는 것이다. 움직임 객체 분할은 내용 기반 응용을 위하여 핵심적인 것이며 새로 개발되는 내용 기반코딩 표준에도 없어서는 안될 핵심적인 도구이다. 또한, 대부분의 내용 기반 응용은 실시간 요구사항을 가지고 있기 때문에 강력하고 효율적인 비디오 움직임 객체 분할 알고리즘의 개발은 매우 중요한 문제로 대두되었다.

MPEG(Moving Picture Experts Group) 그룹에서는 멀티미디어 환경에서 효율적인 동영상 코딩을 위한 새로운 국제표준인 MPEG-4를 개발하였다. MPEG-4는 기존 영상 코딩의 핵심인 프레임기반 비디오 코딩뿐만 아니라 고효율의 압축기능을 제공하는 내용기반 비디오 코딩을 요구한다[1,2].

MPEG-4의 내용기반 비디오 코딩은 VOP(Video Object Plane)라 불리는 독립된 객체를 이용한다. VOP는 동영상 안의 의미있는 요소로서 단위별 처리와 영상 합성 등 새로운 기능을 성취할 수 있도록 한다. 즉, VOP

† 비 회 원 : 호서대학교 컴퓨터공학부
kjk73@hanmail.net
†† 정 회 원 : 호서대학교 컴퓨터공학부 교수
hslee@office.hoseo.ac.kr
논문접수 : 2003년 11월 10일
심사완료 : 2004년 2월 12일

의 사용은 각 객체에 대한 비트스트림 접근, 비트스트림 처리, 객체와의 상호작용, 그리고 장면 구성에 의한 내용정보의 재사용성 등 내용 기반 응용에 매우 적합하다[1].

전통적인 비디오 움직임 객체 분할 알고리즘은 분할 기준에 따라 대략 두 가지 범주로 나누어진다. 참고문헌 [3][4]는 분할 기준으로서 공간적 분할과 시간적 분할, 마지막으로 시간-공간 분할의 조합을 이용하여 객체를 추출하였다. 이 알고리즘의 중요 단계는 다음과 같이 요약할 수 있다. 처음, 수리형태학(morphology) 필터는 영상을 단순하게 만드는데 이용하고, 분수계(watershed) 알고리즘은 지역 경계 결정을 위해 적용된다. 이후 각 지역의 움직임 벡터는 비슷한 움직임을 가진 지역과 움직임 추정에 의하여 계산되어 최종적인 객체 지역을 형성하기 위해 합병된다. 이 알고리즘의 분할 결과는 분수계 알고리즘 때문에 다른 알고리즘보다 정확하게 객체의 경계가 추출되는 경향이 있다. 그러나 분수계 알고리즘과 움직임 측정 알고리즘은 계산의 복잡도가 높고 매우 집약된 계산과정의 수행을 필요로 한다.

다른 접근 방법인 [5][6][7]은 주요한 분할 기준으로 변환 검출(change detection) 알고리즘을 사용하였다. 변환 검출 알고리즘은 연속되는 두 프레임의 차이로부터 움직이는 객체의 모양과 위치를 검출한다. 이것은 새로운 형상의 자동 검출을 가능하게 하며 수월한 구현 때문에 매우 효율적인 알고리즘이다. 또한 이것은 두 영상에 대한 움직임 추정과 보상을 수행하는 것이 아니라 배경으로부터 움직임 객체를 분리하는 빠른 알고리즘이기 때문에 분수계 알고리즘보다 더욱 효율적인 접근방법으로 인식된다.

그러나, 만일 움직임 정보를 알지 못한 상황에서 우선적으로 공간적 분할을 다루는 알고리즘을 사용하면 배경으로부터 객체를 분할하는데 많은 계산량을 필요로 할 것이다.

또한 변환 검출 알고리즘의 사용은 몇몇 단점을 가지고 있다. 첫째, 변환 검출 마스크는 노이즈, 객체의 그림자, 그리고 빛 변화에 민감하다. 둘째, 종종 전경 물체의 운동 때문에 겹치지 않는 배경 영역을 객체 영역으로 간주할 수 있다. 따라서 이러한 상황에서 배경지역을 제거하기 위하여 열린 배경 지역에 움직임 측정 알고리즘을 적용하는 방법이 있다[8]. 그러나 움직임 측정 알고리즘은 매우 많은 처리 시간을 소비한다. 셋째, 변환 검출의 주요한 분할 기준은 프레임 차이이다. 그러나 프레임 차이 값은 객체의 움직임 속도에 의존적이다. 따라서 만일 객체의 움직임 변화 속도가 매우 유동적이라면 분할 결과에 따른 영상의 품질이 떨어지는 단점이 있다.

본 논문에서는 객체 색과 배경 색의 일치 혹은 객체의 움직임 감소로 인하여 나타나는 에지의 단락을 완벽하게 해결하여 분명하고 정확한 움직임 객체를 분할하는 빠르고 강건한 자동 비디오 움직임 객체 분할 알고리즘을 제안한다. 제안한 알고리즘의 가장 큰 특징은 강력한 에지 연결이다. 움직임 객체의 단락된 에지에 시작 픽셀과 마지막 픽셀을 설정하여 강력하고 포괄적인 에지 연결 알고리즘을 수행한다. 에지 연결 알고리즘은 객체의 에지 단락을 제거하여 더욱 분명하고 정확한 비디오 움직임 객체의 추출을 가능하게 한다.

본 논문의 구조는 다음과 같다. 2장에서는 개발된 알고리즘을 자세히 소개하였고, 3장에서는 개발된 알고리즘의 구현과 실험 결과를 소개하였고, 마지막으로 4장에서는 결론을 맺었다.

2. 비디오 움직임 객체 분할 알고리즘

비디오 움직임 객체 분할 알고리즘은 사용자 개입이 없이 완전 자동이어야 한다. 또한 분명하고 정확한 객체의 경계가 추출되어 내용 기반 코딩 응용에 적용할 수 있어야 한다.

본 논문에서 제안하는 객체 분할 알고리즘의 기본 아이디어는 변환 검출(change detection)과 강력한 에지 연결(edge linking)이다. 움직임 객체는 움직임 정보를 기반으로 장면에서 배경과 분리 되어야 한다. 그러나 우리의 분할 기준은 다른 변환 검출 알고리즘[6][7]과 다르다. 우리는 연속적인 두 프레임 사이에서 움직임을 찾지 않는다. 대신에 우리는 각 프레임의 차이 픽셀 정보를 비교하고 누적하여 연속 동영상으로부터 가장 최신의 입력 프레임 정보를 유지한다. 왜냐하면 두 프레임 사이의 차이 픽셀 정보는 잡영(노이즈), 객체의 그림자, 그리고 빛 변화 등에 의하여 정확한 객체의 위치가 나타나지 않을 수가 있기 때문이다.

다시 설명하면, 우리는 두 프레임 사이의 변화하는 부분의 특징은 매우 예측 불가능하기 때문에 단지 두 프레임 사이의 장면 변화로부터 객체의 형상 정보를 얻으려고 하지 않았다. 따라서 우리의 초점은 다중 프레임 차이 누적에 의한 차이 프레임 정보이다. 즉, 프레임 정보는 두 프레임 사이의 차이 픽셀을 계산하여 발생하는 차이 픽셀의 크기를 기준으로 단일 차이 프레임과 다중 누적 차이 프레임으로 나누었다. 다중 누적 차이 프레임 픽셀 정보는 단일 차이 프레임 픽셀 정보와 달리 속도가 빠른 움직임 객체나 혹은 속도가 느린 움직임 객체와 같이 어떠한 움직임 속도를 갖는 객체에 대하여서도 신뢰할 수 있는 움직임 정보를 제공하여 완전 자동 움직임 객체 분할을 가능하게 한다.

그러나 움직임 에지에는 일정하지 않은 조명에 의한 에지 검출 실패, 객체의 색과 배경 색의 일치, 혹은 객

체의 움직임 속도의 현저한 감소로 인하여 에지의 단락이 발생하여 완전한 경계를 나타내지 못하는 경우가 많다. 그러나 기존에 개발된 에지 연결 알고리즘은 에지 기울기의 크기값과 이웃하는 픽셀의 방향성에 의존하여 에지 연결을 수행하였다. 따라서, 이러한 두 가지 속성을 만족하지 못하는 경우에는 에지 연결이 불가능해진다. 이 문제를 해결하고자 어떤 경우[8]에는 영상의 기울기에 초점을 두지 않고 색 정보를 이용하여 객체의 경계를 구분하는 알고리즘을 개발하기도 하였다. 그러나 이 알고리즘은 계산 비용이 많이 들며, 배경과 객체의 색이 동일하게 되면 배경 또한 객체로 분리하는 단점을 나타내었다.

본 논문에서는 새로운 강력하고 포괄적인 에지 연결 알고리즘을 개발하여 경계가 분명하지 않은 단락된 에지 픽셀들을 하나의 의미 있는 객체의 경계로 결속하였다.

참고문헌 [9][10]에서는 카메라 움직임에 따른 배경 변화를 유사(affine) 움직임 모델과 같은 전역 움직임 측정과 보상에 의해 객체를 분할하는 과정을 보여주었다. 그러나 이 방법은 완전하지 않다. 우리는 고정된 카메라와 움직이지 않는 배경을 가정하고 알고리즘을 개발하였다. 이러한 가정은 영상 회의, 원격 교육, 그리고 원격 감시 시스템과 같은 응용 분야에 적합하다. 또한 우리는 실시간 영상 카메라로부터 연속 동영상을 입력하여 실험하였으며 영상 내부의 객체는 오직 하나의 객체만을 처리하였다.

2.1 알고리즘 개요

개발한 알고리즘의 전체 구조는 그림 1과 같다. 각 단계의 주요 처리 내용은 다음의 7단계로 요약할 수 있다.

1. 영상 노이즈 제거
2. 단일 차이 프레임과 다중 차이 프레임 누적
3. 공간적 분할인 Canny 에지 검출
4. 시간-공간 모델 매칭에 의한 비디오 움직임 객체 에지 추출
5. 단락된 움직임 객체 에지의 시작 픽셀과 마지막 픽셀 설정
6. 5번에서 설정한 시작 픽셀과 마지막 픽셀을 이용한 방향, 거리, 각도에 의한 에지 연결
7. 움직임 객체 추출과 후처리

처음 단계에서는 프레임 차이를 계산하기 전에 영상의 잡영을 제거한다.

두 번째 단계에서는 연속되는 두 입력 프레임 사이의 차이 픽셀에 절대값을 적용하여 차이 프레임을 추출한다. 이때 차이 프레임은 3에서 5개의 이전 프레임을 누적하여 오랫동안 움직임이 없는 픽셀은 배경으로 간주하고 누적된 차이 픽셀의 범위의 신뢰도에 따라 객체의 움직임을 인지한다. 이 단계에서는 누적 버퍼에 가장 최

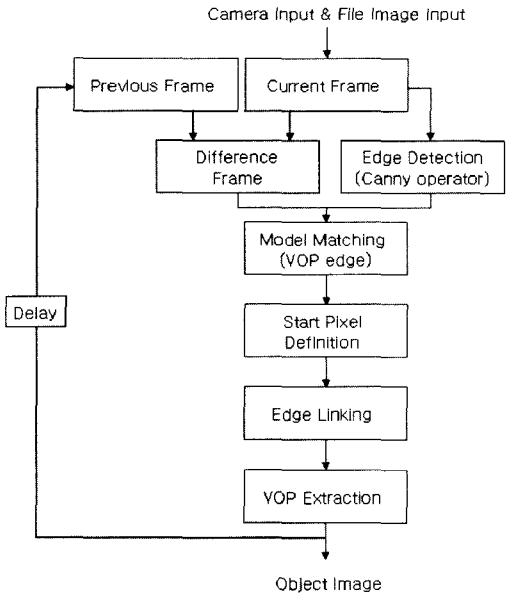


그림 1 움직임 객체 분할 알고리즘

신의 움직임 객체 픽셀을 유지한다.

세 번째 단계는 현재 분할될 객체의 에지 정보를 추출하는 과정이다. 에지 검출은 Canny[11] 검출 알고리즘을 사용하였다. 현재 프레임에 대한 에지 검출은 정확한 객체의 에지를 추출하여 움직임 객체 에지를 생성하기 위한 단계이다.

네 번째 단계는 누적된 프레임 버퍼와 분할될 현재 프레임의 에지 정보를 매칭하는 단계로 이 단계는 움직임 객체 에지 생성을 위한 중요한 단계이다. 그러나 매칭에 의하여 추출된 움직임 객체 에지는 여러 가지 요인으로 인하여 에지 단락 현상을 나타낸다. 이 문제는 강력한 에지 연결 알고리즘을 적용하여 해결한다.

다섯 번째 단계는 시작 픽셀을 설정하는 단계이다. 에지 연결 알고리즘을 적용하기 위하여 시작 픽셀을 설정한다. 에지 연결은 객체의 단락된 에지를 서로 연결하여 움직임 객체의 완전한 경계를 얻도록 한다. 연결된 에지는 분명하고 정확한 움직임 객체를 추출하는데 결정적으로 기여한다.

여섯 번째 단계는 에지 연결 단계로 이전 단계에서 설정한 시작 픽셀을 출발점으로 거리, 방향, 각도를 이용하여 강력한 에지 연결을 수행한다. 움직임 객체는 에지 연결에 의하여 단락이 없는 완전한 에지를 가지게 된다.

일곱 번째 단계에서는 움직임 객체를 추출한다. 추출된 움직임 객체는 에지 연결 알고리즘에 의하여 연결된 완전한 에지에 의하여 침식 현상이 없는 분명하고 정확한 객체이다.

2.2 차이 프레임과 프레임 누적

움직임은 연속 동영상에서 객체 분할을 위한 매우 유용한 특징이다. 이것은 색, 명암도, 에지와 함께 동영상의 분할에 사용된다.

본 논문에서는 움직임 검출을 위하여 변환 검출 알고리즘을 이용하였다. 변환 검출은 움직임 추정이나 보상 또는 공간 분석과 같은 계산 증가량이 없기 때문에 빠르고 효율적으로 실시간 응용에 적용할 수 있다. 그러나 변환 검출은 잡영, 객체와 배경 색의 일치, 그리고 객체의 움직임 속도에 따라 분할 결과의 정확성이 달라지는 단점이 있다. 이러한 단점을 극복하기 위하여 본 논문에서는 다음과 같은 해결책을 사용하였다.

잡영은 차이 프레임을 계산하기 전에 가우시안 필터를 적용하여 제거하였다. 객체와 배경 색의 일치에 의한 에지 단락 지역에는 개발된 에지 연결 알고리즘을 적용하여 해결하였다.

객체의 움직임 속도에 따른 문제의 해결은 두 프레임 사이의 차이 픽셀을 계산하여 임계값 이하의 차이 픽셀이 나올 경우에는 움직임이 미세 하다고 판단하여 누적 차이 프레임과 블록 기반 변환 검출 알고리즘을 적용하였고, 차이 픽셀 값이 클 경우에는 움직임이 많다고 판단하여 차이 프레임과 픽셀 기반 변환 검출 알고리즘을 적용하였다.

(1) 누적 차이 프레임과 블록기반 변환 검출: 개발한 변환 검출 알고리즘은 이전의 방법과는 다른 방법을 이용하였다. 움직임이 미세한 부분에서는 오직 두 프레임 사이의 차이 정보만을 얻기 위하여 노력하지 않았다. 즉 움직임이 미세한 부분은 에지의 단락 지역이 넓기 때문에 다수의 프레임으로부터 차이를 누적하여 움직임 범위를 검출하였다. 또한 두 프레임에 대한 차이 픽셀 결과에 3×3 블록 혹은 5×5 블록을 적용하여 움직임 객체의 에지 단락을 최소화 하였다.

차이 픽셀과 누적 버퍼를 구하는 방법은 (1)과 같다.

$$D\_Buffer = | G*F - G*Fn |$$
$$D\_Pixel \mathrel{+}= D\_Buffer$$

$$\begin{aligned} &if(D\_Pixel < Threshold) \\ &\quad Accumulate\ base \\ &else \\ &\quad Pixel\ base \end{aligned} \tag{1}$$

G : Gaussian convolution

F : Previous Frame, Fn : Current Frame

실시간 카메라를 통한 입력 동영상은 주변 환경에 의하여 많은 잡영을 동반한다. 우선 입력 영상의 잡영 제거를 위해 식 (1)에서와 같이 입력된 영상에 가우시안 필터를 적용하였다.

*D_Buffer*는 잡영이 제거된 영상의 픽셀 차이값을 저장하는 버퍼이다. *D_Pixel*은 저장된 차이값을 누적하여 영상에서 움직임의 크기를 측정하는 단위이다. 실험에 의하여 임계값은 500,000 정도가 적절한 값으로 결정되었다. 임계값은 누적 차이 프레임과 블록 기반 변환 검출을 적용할 것인지 아니면 픽셀 기반 변환 검출을 적용할 것인지를 결정하는 중요한 변수가 된다. 다음 그림 2는 (a) CIF(352*288)입력 영상, (b) 3×3 블록기반 차이 영상, (c) 3개의 프레임을 누적한 차이 영상을 나타내었다.

(2) 차이 프레임과 픽셀 기반 변환검출: 픽셀기반 변환 검출은 움직임이 빠른 영상에 적용하였다. 차이 프레임의 임계값은 식 (1)의 Threshold 값 설정에 따라 움직임이 발생한 부분과 정지된 배경을 구분하였다. 두 연속된 프레임 사이 픽셀 값의 절대값 차이를 기준으로 움직임 부분을 측정하였다. 경험적 결과로서 절대값 차이를 7로 설정했을 경우 잡영이 없는 좋은 움직임 에지를 얻었다. 다음 그림 3은 잡영이 없는 픽셀기반 변환 검출 영상을 보여준다.

그러나 변환 검출에 기반한 움직임 검출의 최대 단점은 그림 2의 (b),(c) 그리고 그림 3의 (b)에서와 같이 움직임에 따라 객체와 배경 사이에 겹치는 지역과 겹치지 않는 지역이 나타난다는 것이다. 즉 그림 2, 그림 3을 보면 배경은 검은색이고 객체는 흰색이지만 흰색 객체 내부에 검은색이 나타난다. 따라서 이러한 부분은 배

(a) (b) (c)

그림 2 (a) 입력 영상, (b) 3×3 블록 기반 차이 영상, (c) 3개의 프레임을 누적한 영상

(a)

(b)

그림 3 (a) 입력 영상, (b) 픽셀기반 변환 검출 영상

경과 객체를 식별할 수 없게 된다는 점이다. 그러나 본 연구에서는 이러한 단점은 문제되지 않는다. 왜냐하면 개발한 알고리즘은 시간적으로 검출된 움직임 객체의 경계만을 사용하기 때문이다. 또한 혹시 에지의 경계 부분에 단락이 발생하더라도 강력한 에지 연결 알고리즘에 의하여 에지의 단락 부분이 연결되기 때문이다.

2.3 공간적 분할의 객체 에지 검출

Canny 에지는 객체의 윤곽을 검출하기 위하여 사용된다. *Canny* 에지 검출 방법의 특징은 다른 에지 검출 방법보다 노이즈에 강하며 컨벌루션하는 가우시안 함수의 조절을 통해 적절한 에지 결과 영상을 얻을 수 있다는 점이다. Canny에지 검출을 위한 계산식은 다음과 같다[10].

(1) 가로 방향을 x축, 세로방향을 y축이라 가정하고 [g(x) : 가우시안 함수, f(x,y) : x 행, y열의 그레이스케일 값] 세로 방향에 대한 에지 값을 얻기 위해

$$\frac{r}{r(x)}(g(x)*f(x,y)) \quad (2)$$

가로 방향에 대한 에지 값을 얻기 위해

$$\frac{r}{r(y)}(g(y)*f(x,y)) \quad (3)$$

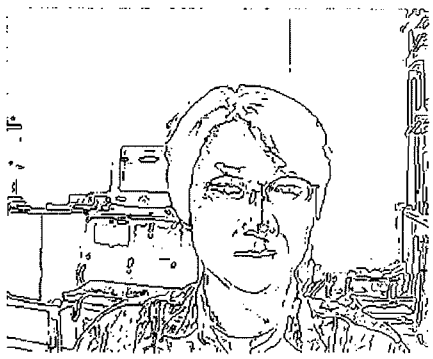
(2) 가로와 세로 방향의 에지 성분의 조합을 위해

$$\sqrt{[\frac{r}{r(x)}(g(x)*f(x,y))]^2+[\frac{r}{r(y)}(g(y)*f(x,y))]^2} \quad (4)$$

로 합쳐준다

다음 그림 4의 (a)와 (b)는 에지 검출의 결과이다. 가우시안 함수의 시그마(σ) 값은 1.2를 사용하였다.

(a)



(b)

그림 4 (a) Canny 에지 검출 결과 영상-1,
(b) Canny 에지 검출 결과 영상-2

2.4 시간-공간 매칭과 움직임 객체 에지

처음에는 움직임 객체의 위치는 알 수 없으며 어떤 모델도 존재하지 않는다. 움직임 객체는 배경으로부터 구분되는 움직임 차이를 보이는 물리적 객체이다. 앞 절에서 시간적 공간적 검출 알고리즘을 소개하였으나 이전에 검출된 객체는 배경과 분리된 정확한 움직임 객체로 인식할 수 없다. 따라서 시간-공간 매칭을 사용하여 정확한 움직임 객체 모델을 검출하여야 한다. 초기 검출 객체는 누적 또는 차이 프레임 결과와 Canny 에지 검출 결과로부터 생성된다. 매칭에 사용하는 식은 다음과 같다.

$$VE(VOP\_Edge) = DIF \ \& \ CE \quad (5)$$

DF(Difference Frame), CE(Canny Edge)

(5)는 시간적 분할에 이은 차이 프레임과 공간적 분할에 이은 Canny 에지 검출을 픽셀기반 논리곱에 의하여 두 프레임을 매칭시키는 수식이다. 다음 그림 5의 (a)는 입력영상, (b)는 시간적 분할 영상, (c)는 공간적 분할 영상, (d)는 움직임 에지 영상으로 (5)식에 의해 매칭된 결과이다.

2.5 움직임 에지와 시작 픽셀

우리는 시간-공간 매칭 결과를 움직임 에지라 정의하였다. 움직임 에지는 시간적 분할에 의한 프레임 차이 결과와 공간적 분할에 의한 결과를 매칭하여 나타난 최종적인 에지 영상을 말한다. 즉, 움직임 에지 영상은 그림 5의 (d)이다.

그러나 움직임 에지에는 크게 두 가지 형태의 문제점이 있다.

첫 번째 문제점은 입력 동영상에서 객체의 색이 배경의 색과 일치한 부분에서는 움직임 에지에 단락이 발생하여 완벽한 에지 검출 결과를 얻을 수가 없다는 점이다.

두 번째 문제점은 실시간으로 카메라에서 동영상을 입력 받기 때문에 객체가 움직임을 멈췄을 경우에는 움직임 에지가 검출되지 않아서 객체와 배경 지역을 분리할 수 없다는 것이다.

이 두 가지 문제를 해결하기 위해서 본 논문에서는 다음과 같은 해결책을 제시한다.

첫 번째 해결책은 에지 연결이다. 움직임 에지에 시작 픽셀과 마지막 픽셀을 설정하고 에지 연결을 수행하여 객체의 에지 연결성에 완전함을 이룩하였다. 두 번째 해결책은 프레임 차이를 설정하여 시간적으로 객체의 움직임이 없더라도 이전 프레임을 계속 갱신하여 멈춤이 없이 객체를 연속적으로 분할할 수 있도록 하였다.

우리는 에지 연결을 효율적으로 수행하기 위하여 전체 움직임 에지 영상에 에지 연결이 시작될 위치를 정하는 시작 픽셀(start pixel)과 마지막 위치를 나타내는 마지막 픽셀(end pixel)을 정의하였다. 시작 픽셀이란 다음 그림 6에서와 같이 추출된 각 라인의 첫 번째 위치를 말하며 마지막 픽셀은 라인 연결의 마지막 위치를 말한다. 그림 6에서와 같이 시작 픽셀은 한 라인에 중복될 수 없다.

시작 픽셀과 마지막 픽셀 설정 알고리즘의 블록 다이어그램은 다음 그림 7과 같다.

시작 픽셀과 마지막 픽셀 식별 알고리즘은 움직임 에지 영상에서 각 에지 라인을 추적하여 시작 위치와 마지막 위치를 식별하는 것이다. 처음으로 그림 8에서와

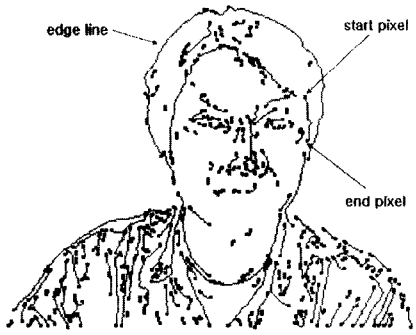


그림 6 시작 픽셀 영상

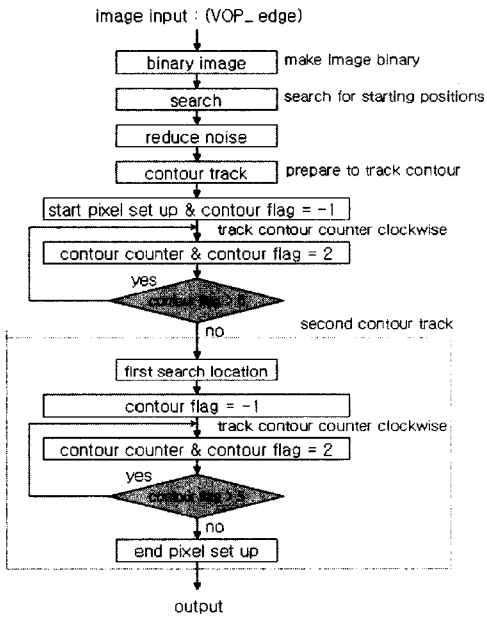


그림 7 시작 픽셀과 마지막 픽셀 설정 알고리즘 블록 다이어그램

같이 최종적인 움직임 에지 영상을 0과 255 이진 형태로 변환하였다. 이후 전체 영상에 대하여 좌에서 우로 한 라인씩 탐색을 수행한다. 처음에 시작 픽셀을 발견하

(a) (b) (c) (d)

그림 5 (a) 입력 영상, (b) 시간적 분할 영상, (c) 공간적 분할 영상, (d) 움직임 에지 영상

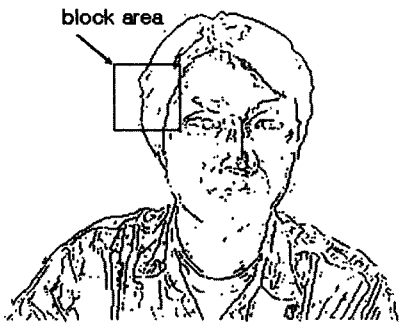


그림 8 움직임 에지 영상

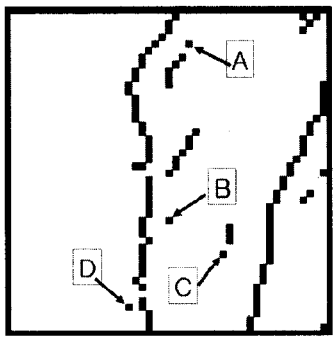


그림 9 그림 8의 확대 영상

는 과정은 흰색 배경지역에 검은색 픽셀이 발견되는 그림 10의 S(start pixel) 위치를 시작 픽셀로 설정하는 것이다. 다음에 마지막 픽셀을 발견하는 과정은 S를 출발점으로 연결된 픽셀을 8-이웃 방식으로 추적하여 더 이상 연결이 없는 마지막 위치를 마지막 픽셀 E(end Pixel)로 설정하는 것이다.

그림 8의 블록 지역을 확대하면 다음 그림 9와 같다. 그런데 움직임 에지 영상인 그림 9를 보면 A, B, C, D와 같은 시작 픽셀 설정에 불필요한 부분이 존재한다. 우리는 이러한 부분을 잡영으로 판단하여 제거한다.

따라서 시작 픽셀 설정이 불필요한 에지 픽셀은 각각의 A, B, C, D 픽셀을 기준으로 3x3 블록을 설정하여 A, B, C, D 픽셀과 3x3 블록을 매스킹 하였다. 매스킹 작업은 A 픽셀 주변의 이웃 픽셀을 검사하여 이웃하는 곳에 계속적으로 라인 연결이 없다면 시작 픽셀 설정단계를 스킵하는 과정이다. 그림 9의 영상에서 잡영을 제거한 결과 그리고 시작 픽셀과 마지막 픽셀을 설정한 영상은 그림 10과 같다.

움직임 에지 영상에 시작 픽셀과 마지막 픽셀을 설정하는 단계는 그림 7의 블록 다이어그램과 같이 두 번의 반복 루틴이 있다. 이렇게 두 번의 라인 추적을 수행하는 이유는 확대 영상 그림 10에서와 같이 S와 E의 조

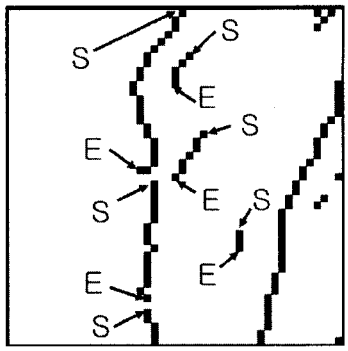


그림 10 그림 9에서 잡영이 제거되고 시작 픽셀과 마지막 픽셀이 설정된 영상

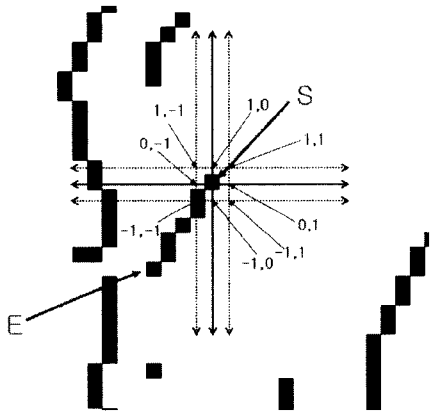


그림 11 추적 방법

건을 찾기 위하여 두 번의 비교가 필요하다는 것이다. 다르게 말하면 영상 어느 위치에서나 좌에서 우로 흰색 배경과 검은 색 픽셀을 찾으면 그 곳을 시작 픽셀로 설정하고 다음 그림 11과 같은 추적 방법을 통하여 라인의 위쪽과 아래쪽을 추적 함으로서 시작 픽셀과 마지막 픽셀을 찾을 수가 있다.

그림 12는 전체 영상을 좌에서 우로 스캔하면서 검은색 픽셀과 흰색 픽셀이 만나는 처음 부분을 시작 픽셀로 설정하는 부분이다. 이후 시작 픽셀의 주변을 그림 11과 같이 (0,-1), (-1,-1), (-1,0), (-1,1), (0,1), (1,1), (1,0), (1,-1), 순으로 탐색을 수행한다. 이렇게 탐색되는 과정에서 (0,-1)에서 시작해서 다시 (0,-1)로 탐색이 반복된다면 이러한 위치는 주변에 더 이상 추적할 픽셀이 없으므로 이곳을 시작 픽셀이나 마지막 픽셀로 설정하는 것이다. 다음 그림 12는 시작과 마지막 픽셀의 탐색 과정과 탐색 결과를 보여준다.

요약하자면, 처음 탐색을 통하여 그림 12의 (a)를 시작 픽셀로 설정한다. 이후 그림 11의 라인 추적 알고리

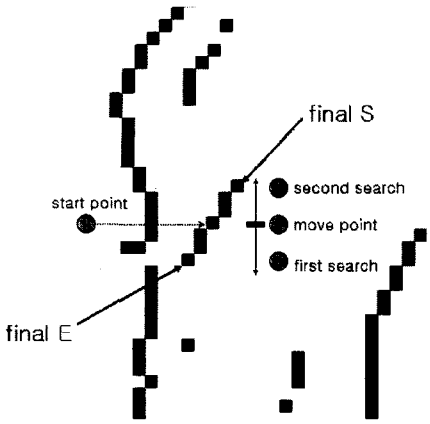
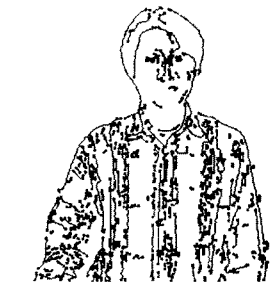


그림 12 탐색 과정과 탐색 결과



(a)

(b)

그림 13 (a) 시작 픽셀 설정 영상-1
(b) 시작 픽셀 설정 영상-2

즘에 의하여 그림 12의 (b) 방향으로 추적을 수행하다가 더 이상 연결이 되어있지 않은 라인을 만나면 최종적인 마지막 픽셀(E)를 설정한다. 다음으로 그림 12의 (d) 위치로 시작 점을 이동하고 그림 12의 (c) 방향으로 라인 추적을 수행하여 마지막 시작 픽셀 (S)를 설정한다. 최종적으로 양쪽 라인의 위치를 조사하여 각각 시작 픽셀과 마지막 픽셀을 255로 설정한다. 수행 결과는 그림 13의 (a), (b)와 같다.

2.6 에지 연결

대부분의 에지 연결 알고리즘은 에지 픽셀의 기울기

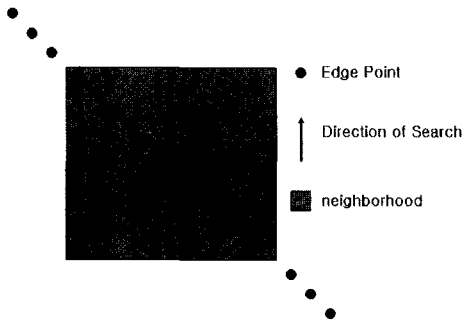


그림 14 일반적인 에지 연결

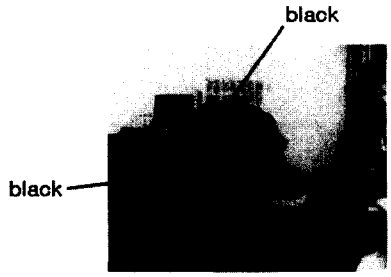
값과 에지 픽셀의 방향 정보를 이용한다. 즉 비슷한 방향을 가지고 있는 이웃 픽셀은 계속하여 연결이 된다고 가정하는 것이다. 에지 연결 방법은 보통 몇몇의 가장자리 픽셀에서 출발하고 방향의 유사성을 통하여 그림 14와 같이 에지 방향을 고려하여 연결을 수행한다.

그림 14와 같이 만일 픽셀이 방향성을 만족한다면 픽셀을 추가하여 에지 연결을 이룬다. 가장자리 픽셀 주의에 놓여진 이웃 픽셀의 조건을 탐색하여 에지 연결이 진행되며 만일 이웃 픽셀이 조건을 만족하지 않는다면 에지 연결 과정을 중단한다. 이러한 에지 연결 알고리즘은 에지의 기울기에 의존하여 연결을 수행한다. 그러나 다음 그림 15(a) 블록 지역은 사람의 머리 색깔과 뒤 배경의 색깔이 같다. 이러한 경우는 기울기 값이 존재하지 않기 때문에 객체의 경계를 명확하게 구분할 수 없다. 실제로 그림 15(a) 영상에 Canny 에지 검출을 적용하여 그림 15(b) 영상을 구할 수 있다. 그림 15(b)를 보면 지정된 블록 지역에는 에지가 단락되어 있다. 더욱더 이러한 지역은 색을 이용한 분할을 수행하더라도 같은 색을 가지고 있기 때문에 객체의 경계를 검출하는 것은 불가능하다.

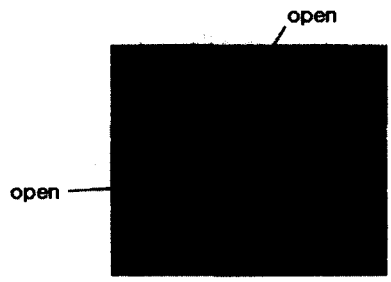
그림 16의 (b)와 (c)에 위 그림 15에서 설명한 에지 단락을 해결한 에지 연결 그림이 제시되어 있다.

즉, 우리의 에지 연결 알고리즘은 이러한 문제를 에지 연결에 의하여 해결할 수 있는 큰 장점이 있다. 즉, 순간적으로 에지가 검출되지 않은 에지 단락 지역을 에지 연결에 의하여 완전한 에지를 형성할 수가 있다. 그러나 이런 문제와는 달리, 움직임 객체와 영상 프레임의 가장자리가 만나는 부분에는 에지가 연결되지 못하는 경우가 있다. 이러한 경우는 후처리 부분에서 직선 연결 알고리즘을 이용하여 에지를 연결하였다.

본 논문에서 제안된 에지 연결 알고리즘은 시작 픽셀과 마지막 픽셀 사이의 거리, 방향, 그리고 각도 3가지 중요한 변수를 사용한다. 에지 연결 알고리즘은 이전 단계에서 설정한 중복되지 않는 시작 픽셀과 마지막 픽셀



(a)



(b)

그림 15 (a) 객체와 배경이 같은 색을 가지고 있는 영상, (b) (a)영상에 Canny 에지 검출 연산을 적용하여 검출된 에지 결과 영상

을 입력으로 수행된다.

본 연구에서 개발한 에지 연결 알고리즘의 전체 구조는 다음 그림 16과 같다.

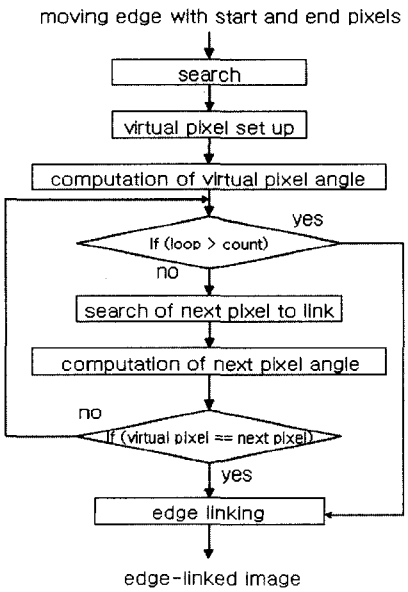


그림 16 에지 연결 전체 구조

그림 17은 시작 픽셀이 정의된 전체 움직임 에지 영상이다.

다음에 설정된 시작 픽셀과 마지막 픽셀은 가상 픽셀을 설정하는 단계를 수행한다. 가상 픽셀이란 설정된 시작 픽셀과 마지막 픽셀의 주변을 조사하여 그 픽셀들의 방향과 각도를 가상으로 설정한 픽셀을 말한다. 가상 픽셀을 사용하는 이유는 그림 18과 같이 시작 픽셀과 마지막 픽셀의 초기 방향을 고려하여 앞으로 에지가 연결될 지역을 미리 결정하기 위해서이다.

그림 18에서 (a)와 (b)에 각각 시작 픽셀(S)와 마지막 픽셀 (E)가 설정되어 있다. 처음에, (a)픽셀 주변을 탐색하면 아래 방향에 회색 픽셀 (c)가 있다. 그러므로 (a)

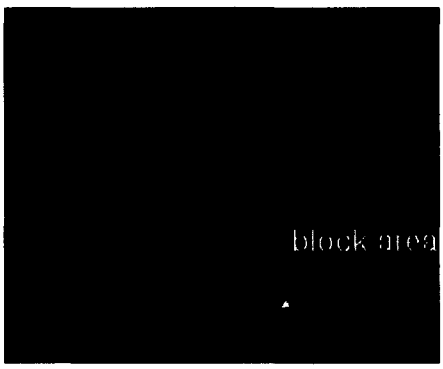


그림 17 시작 픽셀이 정의된 전체 움직임 에지 영상

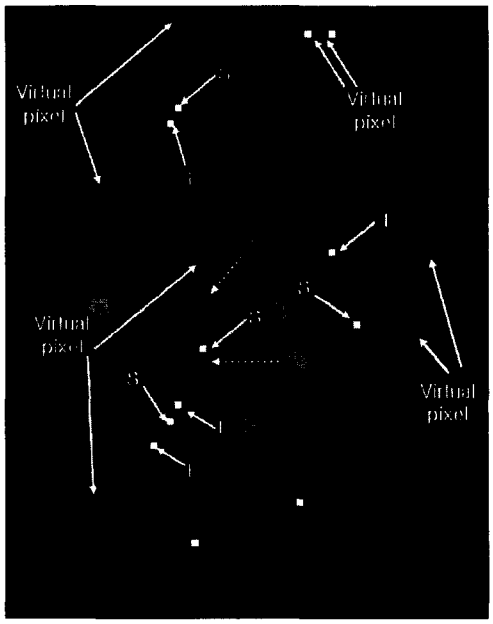


그림 18 그림 17의 블록 지역을 확대한 영상에서의 가상 픽셀 설정 과정

시작픽셀은 (c)를 기준으로 반대 방향으로 계속해서 이동할 방향성이 있다고 가정한다. 즉 그림 18의 짙은 회색 라인인 (d)는 시작 픽셀 (a)가 계속해서 이동할 방향성을 나타낸다. 그러므로 가상 픽셀 설정 단계에서는 시작 픽셀을 기준으로 가상 픽셀의 각도와 방향의 정보를 얻을 수 있다. 하지만 모든 시작 픽셀이 설정된 가상 픽셀의 방향으로 연결이 되어야 한다는 보장은 없다. 왜냐하면 실시간 입력 영상뿐만아니라 모든 영상의 픽셀 분포도는 움직임 객체 영상에 따라 달라지기 때문이다. 물론 사각형이나 다각형과 같은 컴퓨터 그래픽 영상의 경우에는 가상 픽셀의 설정만으로 에지 연결이 가능할 것이다. 그러나 그 이외 실제 영상에서는 가상 픽셀의 정보만 가지고는 완벽한 에지 연결이 불가능한 경우가 있을 수가 있다. 때문에 우리의 알고리즘에서는 가상 픽셀의 정보와 추가적으로 방향성을 고려한 그림 19와 같은 에지 방향 연결 범위를 정의 하였다.

그림 19는 각 사분면에 걸친 각도 범위를 표시하였다. 만일 (S)가 시작 픽셀이며 시작 픽셀 주변을 탐색한 결과 (V)방향의 가상픽셀 정보를 얻었다면 시작 픽셀의 각은 1.570796이 된다. 따라서 (S)는 (V) 방향 쪽으로 에지가 연결될 가능성이 있다. 하지만 에지 연결을 위하여 주변의 또 다른 시작 픽셀을 탐색할 경우 거리와 방향을 고려하기 때문에 (V) 방향에 나타날 확률은 매우 낮아 진다. 예로 그림 19에서는 주변의 또 다른 시작 픽셀을 (s1)과 (s2)로 표시하였다. 즉 (S)의 시작 픽셀을 기준으로 만일 인접한 (s2)의 시작 픽셀을 발견하였다면 (S)와 (s2)는 에지 연결을 하지 않는다. 왜냐하면 (s2)는 (S)와 (V)의 각도 범위인 0.785398에서 2.356194안의 범위가 아니기 때문이다. 그러므로 에지 연결은 또 다른 (s1)의 시작 픽셀을 찾아 각도 범위 조건에 만족하게 되는 (S)와 (s1)을 연결하게 된다. 이렇게 조건을 만족하는 루프는 그림 16의 블록 다이어그램에 조건식으로 나타내었다. 경험적 실험결과 시작 픽셀을 기준으로 5번 정도의 반복을 통하여 주변 시작 픽셀을 탐색했을 경우에 각도 범위 안에서 거리, 방향, 각도 조건을 모두 만족하였다. 또한 더욱 정확한 에지 연결을 위해 5번 이상의 반복을 수행하게 되면 계산량이 많아지는 단점이 발생한다. 때문에 본 논문에서는 5번 이상의 반복일 경우는 시작 픽셀과 가장 가까운 픽셀을 무조건 연결하도록 처리하였다.

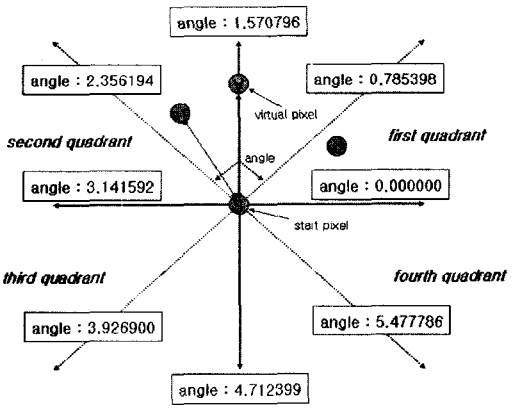


그림 19 에지 방향 연결 범위

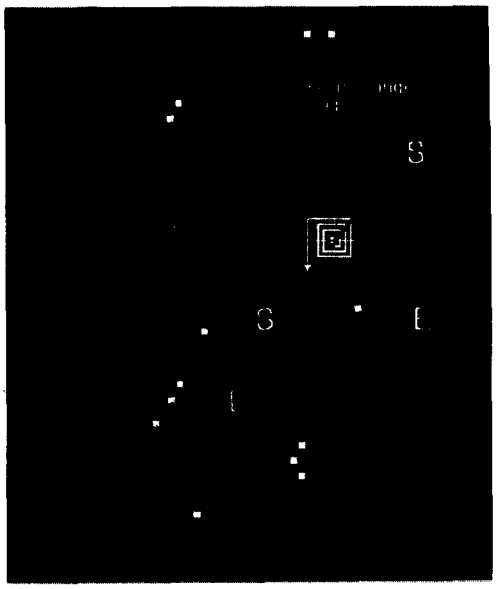


그림 20 에지 탐색 방법

그림 19에서 같이 (S)를 기준으로 (s1)과 (s2)를 탐색하는 알고리즘은 그림 20과 같다.

에지 탐색 방법은 그림 20에서와 같이 (S)를 기준으로 주변 24 픽셀의 범위를 사각형 방향으로 탐색으로 수행한다. 이전 단락에서 언급하였듯이 만일 (S)에서 탐색한 결과가 그림 19의 범위 안에 만족하지 않는다면 다시 (S)를 기준으로 주변의 또 다른 (S)나 (E) 픽셀을 탐색하게 되는 것이다.

요약하자면 거리, 방향, 각도의 변수를 통하여 조건에 만족하는 픽셀이 발견되면 에지 라인을 연결하는 것이다. 연결된 에지 영상의 결과는 다음 그림 21과 같다.

우리의 알고리즘이 이렇게 에지 연결을 수행하는 중요한 이유는 분명하고 또렷한 움직임 객체를 추출하기 위해서이다. 즉, 에지가 단락된 부분이 없으면 더욱 분명하고 정확하게 움직임 객체를 분할할 수가 있기 때문이다.

2.7 움직임 객체 추출과 후처리

그림 21은 에지 연결 단계의 출력 결과이다. 최종적으로 남아 있는 단계는 입력된 동영상에서 에지 영상과

그림 21 연결된 에지 영상

일치하는 실제 움직임 객체를 추출하는 과정이다. 즉 에지 영상을 실제 영상과 매칭하는 과정이 남아있다. 우리는 이러한 과정을 움직임 객체 추출 과정이라 정의하였다. 움직임 객체 추출과정에서의 가장 큰 어려운 점은 이진 영상의 단락이다. 추출된 에지 내부에 색을 채우는 과정에서 에지의 단락이 발생하게 되면 객체와 배경을 구분하지 못한다. 하지만 우리의 알고리즘은 에지 연결을 수행하여 에지 단락을 제거하였기 때문에 더욱 분명하고 또렷한 움직임 객체의 추출이 가능하다.

우리는 움직임 객체의 경계를 결정하는 효율적인 알고리즘을 이곳에 소개한다. 움직임 객체 경계를 결정하는 기본 아이디어는 지역 채우기 알고리즘이다. 채우기 알고리즘은 에지로 지정된 닫혀진 영역 내부를 객체의 원래 색으로 빈틈없이 채우는 방법을 말한다. 기존의 채우기 알고리즘은 사용자 정의에 의한 씨앗(seed)픽셀의 설정이 필요하며, 씨앗 픽셀을 기준으로 내부 영역을 채우기 위해 상하 좌우의 점들에 대해 반복적인 알고리즘이 수행된다. 따라서 이러한 알고리즘은 계산량이 많아 실용성이 없다. 만일 그림 21과 같이 영역 내부에 픽셀들이 매우 많이 뭉쳐 있다면 영역 내부를 채우기 위해 많은 픽셀들을 탐색하여야 한다. 이러한 과정은 매우 비효율적이며, 실제로 실행 속도도 매우 느리다. 우리는 비효율적인 채우기 알고리즘을 개선하여 빠르고 효율적인 움직임 객체 추출 알고리즘을 개발하였다.

움직임 객체 추출 알고리즘의 전체 구조는 그림 22와 같다.

처음에 움직임 객체 추출 알고리즘에 입력되는 영상은 에지 연결 영상이다. 하지만 그림 21, 그림 23과 같이 추출된 객체 에지의 좌측, 우측 그리고 하단을 보면 에지와 프레임 가장자리의 경계 부분에 에지가 단락된 부분이 있다. 이러한 지역은 객체의 내부에 영역을 채우는 알고리즘에 있어 매우 치명적이다. 즉 그림 23의 (a)

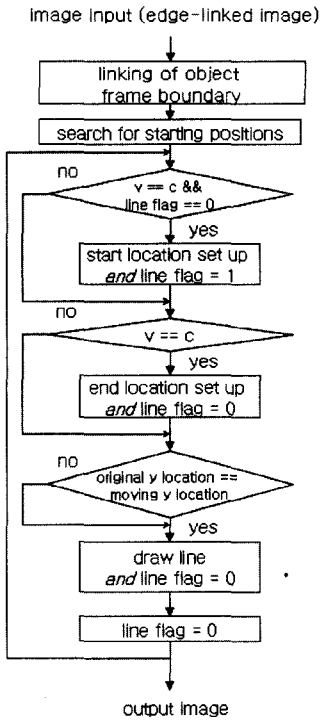


그림 22 움직임 객체 추출 블록 다이어그램

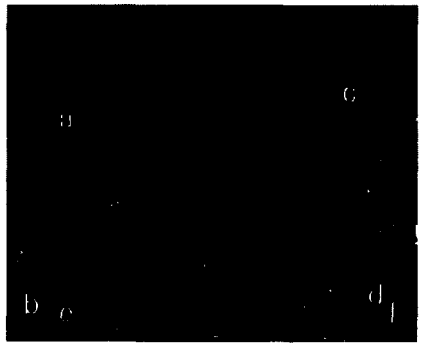


그림 23 움직임 객체에서 단락된 에지 경계

와 (b), (c)와 (d) 그리고 (e)와 (f)는 단락된 객체의 에지 지역을 보여준다. 이러한 에지 단락 지역은 최종적으로 추출될 움직임 객체의 침식 현상을 증가시키게 된다. 때문에 본 논문에서는 전체 영상의 사각 경계 지역을 좌에서 우로 그리고 위에서 아래로 탐색하여 (a)와 (b), (c)와 (d), 마지막으로 (e)좌표와 (f)좌표를 연결하였다. 각 좌표를 연결한 영상은 그림 24와 같다. 이러한 경계 지역의 에지 연결은 에지 영상의 단락을 최소화 하여 더욱 정확한 객체의 에지 경계를 얻도록 한다.

다음으로, 경계지역의 에지 연결 후에는 그림 22 블록

그림 24 움직임 객체에서 사각 경계를 연결한 영상

다이어그램의 두 번째 박스와 같이 전체 영상을 좌에서 우로 탐색한다. 탐색하는 초기 조건은 그림 22의 첫 번째 조건식에 의해 탐색 픽셀 (v)가 경계색 (c)와 동일한지를 반복적으로 비교한다. 만일 (v)와 (c)가 같다면 이 픽셀은 영상 내부에 채워지는 첫 번째 좌표가 된다. 이후 같은 라인에서 (v)와 (c)가 같아지는 마지막 좌표를 탐색한다. 즉 첫 번째 좌표를 원 좌표 (x, y)로 설정하고 마지막 좌표를 이동 좌표 (x', y')로 설정하여 객체 영역의 내부를 수평선으로 채운다. 그러므로 이런 과정은 기존 채우기 알고리즘처럼 씨앗 픽셀을 기준으로 상하 좌우의 반복적인 탐색과 영상 내부 픽셀의 탐색을 배제하여 반복된 비교 없이 영상 내부를 빠르게 채울 수 있게 한다.

본 연구에서 개발한 움직임 객체 분할 알고리즘의 출력 결과는 그림 25와 그림 26이다.

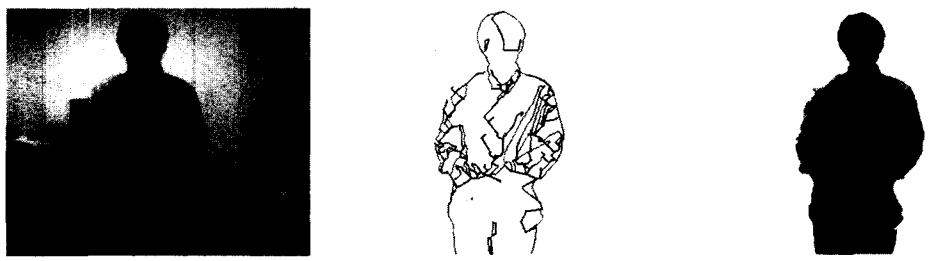
2.8 자동 움직임 객체 분할 시스템

우리의 최종 목적은 MPEG-4 내용기반 코딩을 위한 효율적인 완전 자동 움직임 객체 분할 시스템을 개발하는 것이다. 이러한 목적을 달성하기 위해서, 우리는 움직임 객체 추출 알고리즘에서 집약적인 계산의 사용을 피하였다. 우리는 분할 결과의 영상 품질을 손상시키지 않고 더욱 빨리 처리될 수 있도록 최적화에 초점을 맞추었다. 또한 사용자의 개입이 없이 완전 자동으로 움직임 객체가 분할될 수 있도록 하였다.

다음 그림 27은 완전 자동 움직임 객체 분할 알고리즘의 구조이다.

그림 27의 세부적인 알고리즘은 이전 단락에서 자세히 소개하였다. 추가적으로 그림 27에서의 dpc는 현재 프레임과 이전 프레임의 차이 픽셀 수이다. 즉 dpc의 값이 250,000 이상이 되면 시각적으로 차이를 구분할 수 있으나 그 이하의 값이 되면 시각적으로 차이를 구분할 수 없다. 다르게 말하면 dpc가 250,000이하의 값이면 거의 움직임이 존재하지 않는다는 결론이다. 때문에 이러한 경우는 움직임 객체가 분할되지 않는다. 하지만 본 논문에서는 이러한 부분을 움직임 객체 갱신 모듈을 통하여 처리하였다. 그림 28은 움직임 객체 갱신 모듈이 적용된 결과를 보여준다.

그림 28의 dpc 값은 분명 25000이하의 값을 갖는다.



(a) (b) (c)

그림 25 (a) CIF(352x288)의 입력 영상, (b)에지 연결 영상, (c) 움직임 객체 출력 영상

(a) (b) (c)

그림 26 (a) CIF(352x288)의 입력 영상, (b)에지 연결 영상, (c) 움직임 객체 출력 영상

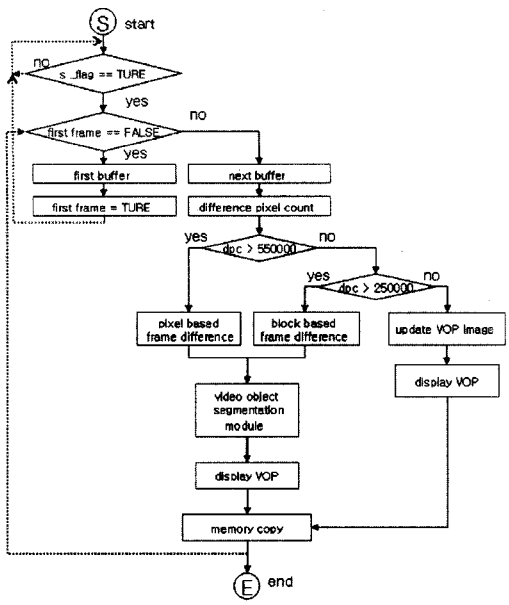


그림 27 완전 자동 움직임 객체 분할 알고리즘 구조

왜냐하면 그림 28의 (a),(b),(c),(d)를 자세히 보면 객체의 외부는 모두 같은 형태를 취하고 있다. 그렇지만 사람 객체 내부의 눈을 보면 (a)영상에서부터 (d)영상까지 눈을 감았다 뜬 형태를 보여준다. 즉 우리의 알고리즘은 움직임 객체가 중간에 정지했더라도 움직임 객체 갱신 모듈을 통하여 객체를 분할할 수 있는 특징을 가지고 있다.

표 1은 제안된 알고리즘의 실행 시간 분석 결과이다. 우리의 테스트 플랫폼은 펜티엄-4 2.0의 개인용 컴퓨터이다.

표 1은 전체 움직임 객체 추출 단계의 평균 처리 시간이다. 영상의 입력은 실시간 영상 카메라에서 움직임이 많은 영상과 움직임이 적은 영상으로 구분하였다. 입력 영상의 형태는 CIF(352 x 288)로 초당 30 프레임의 형태로 1000프레임을 입력 받았다. 실험 결과 1000프레임의 평균 처리 시간은 0.063초와 0.047초로 나타내었다. 즉 하나의 영상을 분할하는데 처리되는 시간은 0.1초 미만으로 계산량을 최소화하여 높은 효율을 달성하였다.

3. 실험 결과

본 논문에서는 입력 영상을 실시간 영상 카메라로부터 입력 받았다. 따라서 실험은 MPEG-4 표준 테스트 영상인 "*Akiyo*"영상과 같이 머리와 어깨만 움직이는 움직임이 적은 영상 그리고 "*Hall Monitor*"와 같은 복잡한 배경과 많은 움직임을 나타내는 영상으로 구분하여 실시간으로 입력하여 실험하였다. 다음은 실험을 수행하여 얻은 객관적 그리고 주관적 평가 결과이다.

3.1 객관적 평가

제안된 객체 분할 알고리즘의 객관적 평가는, 참고문헌[12]에서 제안된 오차 비율을 사용하였다. 오차 비율의 개념은 영상 경계의 왜곡(distortion) 부분을 누적하여 측정하는 것이다. 오차 비율은 다음 식 (6)에 의해 정의된다.

$$ErrorPercentage = \frac{\sum_{W}\sum_{H}(a(x,y) \oplus OM(x,y))}{W \times H} \times 100\% \quad (6)$$

(a) CIF(352x288) 45 프레임 (b) 46 프레임 (c) 47 프레임 (d) 48 프레임

그림 28 움직임 객체 갱신 모듈이 적용된 결과

표 1 평균 움직임 객체 분할 처리 시간

Algorithm	Test environment	Sequece	Frame	Average elapsed time(sec)
proposed algorithm	pentium-4 2.0	CIF(352×288) "Active moving image"	1000	0.06300000
		CIF(352×288) "Small moving image"	1000	0.04700000

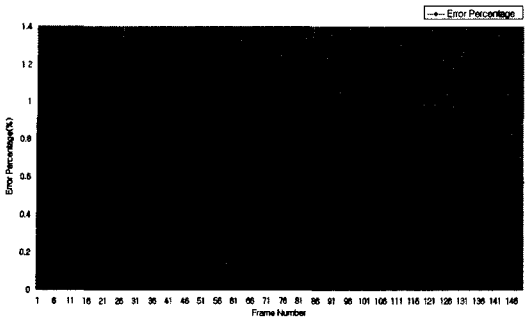


그림 29 각 프레임에 대한 오차 비율 실험 결과

여기서 $a(x, y)$는 참조로 사용되는 원래 영상의 움직임 객체에 대한 알파맵, (+)는 Exclusive-OR, $OM(x, y)$는 알고리즘에 의해 생성된 움직임 객체 알파맵이다. W, H는 입력 영상의 가로와 세로이다.

그림 29는 각 프레임에 대한 오차 비율 실험 결과이다.

그림 29의 실험은 그림 30의 실시간 입력 영상을 사용하였다. 실험한 영상 시퀀스는 초당 25 프레임의 352 x 288 크기 영상이다. 실험 방법은 정확한 결과를 얻기 위하여 실시간 영상 카메라로 입력 받은 원본 영상을 포토샵 영상 도구를 사용하여 사용자에 의해 수동으로 움직임 객체 알파맵을 $[a(x, y)]$ 추출하였다. 또한 같은 원래 영상에 개발한 움직임 객체 분할 알고리즘을 적용하여 움직임 객체 알파맵을 $[OM(x, y)]$ 추출하였다. 이후 추출된 움직임 객체 알파맵에 식 (6)에 적용하여 오차 비율을 측정하였다.

오차 비율은 참조 움직임 객체와 분할된 움직임 객체 사이의 영상 픽셀 차이의 누적을 나타낸다. 만약, 픽셀 차이가 없다면 오차 비율 값은 0%이다. 실험에 의하여 오차 비율 값이 1.5%이상의 값을 나타내면 분할된 움직

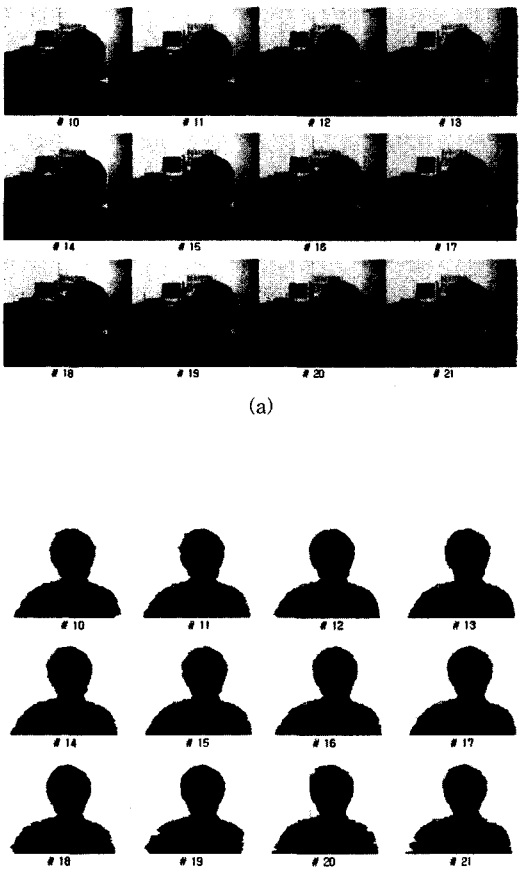


그림 30 (a) 움직임이 적은 동영상
(b) 움직임 객체 분할 결과

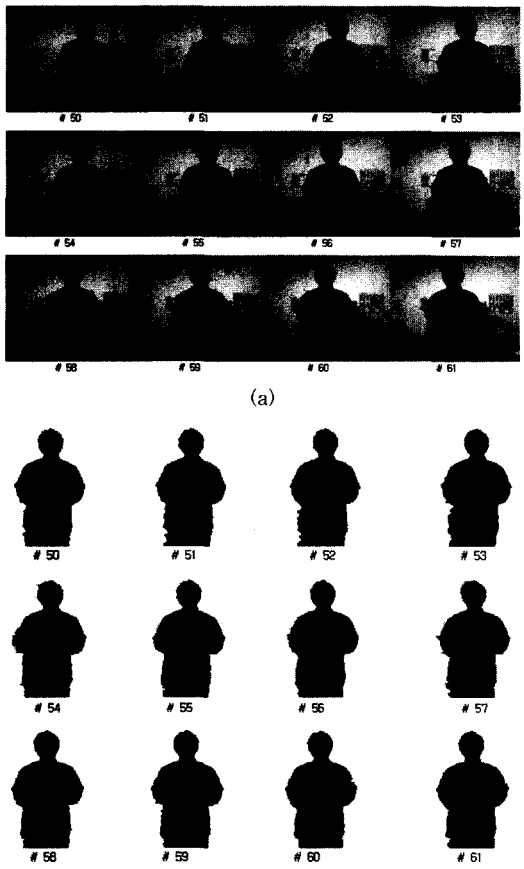


그림 31 (a) 움직임이 많은 동영상
(b) 움직임 객체 분할 결과

임 객체는 참조 움직임 객체와 시각적으로 인지할 수 있는 차이를 나타내는 것을 알 수 있었다.

본 실험에서 측정된 오차 비율 값은 대부분의 프레임에서 0.78% 보다 낮았다. 이 값은 분할되어 생성된 움직임 객체가 원래 영상의 움직임 객체와 상당히 일치함을 보여주는 것이다. 즉, 이 측정 값은 개발한 움직임 객체 분할 알고리즘의 성능이객관적으로 매우 우수함을 입증하는 것이다. 그림 29에서 30프레임과 70프레임은 다른 프레임에 비하여 오차 비율보다 비교적 크다. 왜냐하면 이 부분은 영상이 멈추었다가 다시 움직이기 시작하는 부분이기 때문이다. 또한 111 프레임부터 140 프레임에서도 오차비율 크다. 이것은 이 프레임에서 객체가 갑자기 움직임을 빠르게 하였기 때문이다.

3.2 주관적 평가

제안된 알고리즘은 복잡한 배경을 포함하는 움직임이 적은 동영상과 움직임이 많은 동영상으로 구분하여 실험하였다. 실험 결과는 그림 30과 그림 31이다.

실험 결과 우리의 알고리즘은 머리와 어깨만 움직이는 움직임이 적은 영상 그리고 전체 이미지가 움직이는 움직임이 큰 영상등 실시간으로 입력되는 동영상에 대하여 움직임 객체의 완전하고 정확한 에지와 완전 자동 객체 분할 결과를 매우 효율적으로 나타내었다. 즉, 분할되어 생성된 움직임 객체는 매우 분명하고 정확한 모양을 나타내어 시각적으로 전혀 불편함을 느낄 수가 없었다.

4. 결론 및 향후 계획

본 논문에서는 강건하고 포괄적인 에지 연결 알고리즘과 효과적인 완전 자동 움직임 객체 분할 알고리즘을 제안하였다. 다중 프레임 차이 누적 알고리즘은 프레임 차이 정보를 누적하여 신뢰할 수 있는 움직임 변화를 검출할 수 있도록 한다. 검출된 움직임 픽셀은 분할될 현재 프레임의 에지와 비교하여 움직임 에지를 생성하였다. 여기에 강건하고 포괄적인 에지 연결 알고리즘을 적용하여 에지 단락을 제거하여 분명하고 정확한 움직임 객체를 추출하였다.

개발된 알고리즘들은 객체의 색과 배경의 색이 일치하여 에지 단락이 발생하는 경우와 움직임 객체의 속도가 감소하여 에지 단락이 발생하는 경우를 해결하였다.

더욱 개발한 알고리즘은 개인용 컴퓨터에서 CIF(352 x 288)영상을 30 프레임 이상 처리가 가능하였고, 처리 시간 또한 0.1초 미만의 빠른 결과를 보여주었다. 때문에 개발된 알고리즘은 실시간 고압축 MPEG-4를 위한 전처리 시스템에 더욱 효율적이다. 개발된 자동 움직임 객체 분할 알고리즘은 MPEG-4와 결합하여 실용적인 소프트웨어를 개발할 것이며 고효율의 압축을 위해 MPEG-4 이진 형상 부호화에 적용할 것이다.

차후 본 연구실에서는 개발된 객체 분할 알고리즘을 개선하여 움직이는 배경에서도 객체를 분할할 수 있으며 또한 다중 객체에 대해서도 정확한 움직임 객체 분할을 수행할 수 있도록 할 것이다.

참 고 문 헌

[1] T.Sikora, "The MPEG-4 video standard verification model," IEEE Trans. Circuits and Systems for Video Technology, Vol.7, pp.19-31, Feb. 1998.

[2] ISO/IEC/JTC1/SC29/WG11, MPEG/N3908, "MPEG-4 Video Verification Model version 18.0," Jan. 2001, Pisa.

[3] D.Wang, "Unsupervised video segmentation based on watersheds and temporal tracking," IEEE Trans. Circuits and Systems for Video Technology, Vol. 8, pp.539-546, Sept. 1998.

[4] J.C. Choi, S.W. Lee, and S.D. Kim, "Spatio-temporal video segmentation using a joint similarity measure," EEE Trans. Circuits and Systems for Video Technology, Vol. 7, pp. 279-286, April 1997.

[5] T.Meier and K.N.Ngan, "Automatic segmentation of moving objects for video object plane generation," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 15, pp.525-538, Sept. 1998.

[6] C.I. Kim and J-N.Hwang, "Fast and Automatic Video Object Segmentation and Tracking for Content-Based Applications," IEEE Trans. Circuits and Systems for Video Technology, Vol 122-129, Feb. 2002.

[7] T. Aach, A. Kaup, and R. Mester, "Statistical model-based change detection in moving video," Signal Processing, Vol. 31, pp. 165-180, March 1993.

[8] Y. Cheng, "Mean shift, mode seeking, and clustering," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.17, pp. 790-799, 1995.

[9] R. Mech and M. Wollborn, "A noise robust method for 2D shape estimation of moving objects in video sequences considering a moving camera," Signal Processing, Vol. 66, pp. 203-217, April 1998.

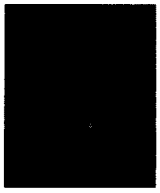
[10] H. Nicolas and C. Labit, "Global motion identification for image sequence analysis and coding," Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, pp. 2825-2828, 1991.

[11] J. Canny, "A computational approach to edge detection," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 8, pp. 679-698, Nov. 1986.

[12] Guido M. Schuster, Gerry Melnikov, and Aggelos K. Katsaggelos, "Operationally Optimal Vertex-Based Shape Coding," IEEE Signal Processing Magazine, November 1998.

김 준 기

1992년 3월~1998년 2월, 호서대학교 전자계산학과 이학사 졸업. 1998년 3월~2000년 2월, 호서대학교 컴퓨터공학부 대학원 공학석사 졸업. 2000년 3월~2001년 2월, 호서대학교 컴퓨터공학부 대학원 공학박사 수료. 2002년 3월~현재, 호서대학교 컴퓨터공학부 강사, 멀티미디어연구실 연구원. 관심분야는 영상처리, 영상분할, MPEG-4, MPEG-7



이 호 석

1979년 3월~1983년 2월, 서울대학교 전자계산기공학과 공학사 졸업. 1983년 3월~1985년 2월, 서울대학교 컴퓨터공학과 대학원 공학석사 졸업. 1989년 3월~1993년 8월, 서울대학교 컴퓨터공학부 대학원 공학박사 졸업. 1994년 3월~현재, 호서대학교 컴퓨터공학부 교수. 관심분야는 영상처리, 영상분할, MPEG-4, MPEG-7, JPEG-2000