적응적 배경영상과 그물형 픽셀 간격의 윤곽점 검출을 이용한 객체의 움직임 검출

정회원 이 창 수*, 전 문 석**

Motion Detection using Adaptive Background Image and A Net Model Pixel Space of Boundary Detection

Chang soo Lee*, Moon seog Jun** ***Regular Member***

요 약

카메라를 통한 객체의 움직임검출은 불필요한 잡음이나 조명의 변화에 따라 정확한움직임검출 하는 것은 어렵다. 또한 객체의 유입 후 일정시간 동안 움직임이 없을 경우에는 배경으로 인식될 수도 있다 본 논문에서는 초기의 배경영상을 기준으로 입력영상과의 차를 구하고 시간에 따라 변화하는배경영상을 N×M 픽셀 마스크만큼 교체하여 갱신한다. 이미지 픽셀 검사는 모든 픽셀을 연산에 참여시키지 않고 일정한 간격의 그물형으로 이미지의 픽셀을 검색하여 보다 효과적으로 움직임을 검출한다. 또한 픽셀검사를 통하여 검출된 객체의 윤곽점을 이용하여 객체의 최소영역을 설정하여 객체의 움직임을 검출하므로, 매 프레임마다 이미지 검사를 하지 않고도 빠르고 정확하게 움직임 검출이 가능하다. 설계하고 구현한 시스템은 실험을 통한 성능평가에서90% 이상의 높은 정확도를 보였다.

Key Words : Motion Detection, Background Image, Video, Pixel Space, Boudary

ABSTRACT

It is difficult to detect the accurate detection which leads the camera it moves follows in change of the noise or illumination and Also, it could be recognized with backgound if the object doesn't move during hours.In this paper, the proposed method is updating changed background image as much as N*M pixel mask as time goes on after get a difference between imput image and first background image.

And checking image pixel can efficiently detect moving by computing fixed distance pixel instead of operate all pixel. Also, set up minimum area of object to use boundary point of object abstracted through checking image pixel and motion detect of object. Therefore motion detection is available as is fast and correct without doing checking image pixel every frame. From experiment, the designed and implemented system showed high precision ratio in performance assessment more than 90 percents.

Ⅰ. 서 론

인터넷 시대에 접어들면서 웹 카메라를 이용한 보안 시스템의 개발이 활발하다 원격지의 카메라로부터 전송된 영상을 통하여 현재의 상황을 파악할 수 있으며, 적절한 조치를 웹을 통해 취할 수 있다 이러한 웹 멀티미디어 보안 시스템은 교통현황 파악, 건설현장이나 상가매장의 모니터링 무인 시설

* 숭실대학교 컴퓨터공학과 통신연구실(powerofmicro@yahoo.co.kr), ** 숭실대학교 컴퓨터공학과(mjun@computing.ssu.ac.kr)

논문번호 : KICS2004-12-340, 접수일자 : 2004년 12월 29일
※본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음

물감시 등에 사용되고 있다. 그러나 활용영역이 확대되면서 영상의 해상도와 전송속도, 그리고 보안시스템의 핵심인 객체영역 인식, 영상 정보의 처리, 저장, 검색 기술 등의 연구가 추가적으로 요구 되고 있다[1].

효율적인 영상 저장을 위해서는 전체 프레임을 저장하는 방법보다는 객체의 움직임을 검출하여 그 시점부터 저장하는 방법을 사용한다. 그런데 이 방법은 픽셀 변화만을 사용하기 때문에 객체의 움직임이 없어도 조명의 변화에 따라 쉽게 객체로 오인하기 쉽다.

실시간 영상에서 배경영상과 입력영상을 구분하여 움직인 객체를 인식하거나 검출하기 위해 차영상을 이용한 방법, 블록정합기법, 배경영상을 이용한 방법 등을 이용한 연구가 주로 이루어지고 있다[2].

차영상을 이용하는 방법은 모션추출의 가장기본적인 방법으로 인접한 프레임의 두 영상의 픽셀간 차이를 이용하는 것이다. 픽셀간의 차이로 움직임을 검출하기 때문에 조명의 변화가 생기게 되면 같은 픽셀이라도 다른 영상값을 가지게되므로 정확한 검출은 어렵다. 또한 이런한 오류를 막기 위한 후처리 과정은 연산처리시간이 길어지게 된다.

블록정합기법은 현재프레임 탐색영역 안에서 이전프레임의 지정된 블록과 가장 유사한 블록을 찾는 방법으로 블록영역에서 객체의 추적이나 객체의 영역을 지정할 수 있다. 그러나 블록영역 밖에서 유입되는 객체는 측정이 불가능하다.

배경영상을 이용한 방법은 현재 프레임과 기준이 되는 배경영상의 차이를 구하는 방법이다. 인접한 프레임을 바로 비교하는 방법이 아니라 이전 프레임들로부터 배경이 될 수 있는 영상을 추출하고 이 영상과 현재 프레임을 비교하여 움직임을 추출하게 된다. 이 방법 역시 배경이 되는 영상은 객체의 유입이 없다고 하나 조명의 변화에 따라 배경자체가 변하게 되므로 움직임이 없는 시점에서도 움직임 검출 오류를 발생할 수 있다.

본 논문에서는 실시간 영상에서 초기의 배경영상을 기준으로 입력영상과의 차를 구하고 시간에 따라 변화하는 배경영상을 N×M 픽셀 마스크만큼 교체하여 갱신 한다. 이미지 픽셀 검사는 모든 픽셀을 연산에 참여시키지 않고 일정한 그물형 픽셀 간격을 두고 이미지의 픽셀을 검색하여 효과적으로 객체의 윤곽점을 검출하다. 객체의 윤곽점의 좌표는 최대 가로방향, 세로방향을 측정하여 객체의 최소영역을 생성하고 움직임을 검출 한다. 최소영역의 생성은 매 프레임 마다 객체의 움직임을 검출하기 위해 이미지를 스캔하지 않고 객체의 유입시점부터는 최소영역의 방향성을 예측하고 검사하여 보다 빠른 움직임 검출을 정확하게 측정할 수 있다.

제안하는 방식인 적응적 배경영상 생성과 그물형 픽셀간격의 임계값은 실험적 데이터를 기준으로 가장 적절한 임계값을 선택하여 테스트 하였다. 제안하는 방식을 사용하여 움직임 검출을 하였을 경우 기존의 움직임 검출방법을 사용할 때보다 조명이나 잡음에 강점을 가니고, 연산처리속도가 상대적으로 빠르기 때문에 즉각적인 반응을 보였다. 또한 동영상 캡쳐 및 정지영상으로 저장이 가능하기 때문에 영상 보안시스템의 저장 공간 절약 및 객체 추적과 같은 분야에 적용이 가능하다.

본 논문에서 2장은 기존 연구방법을 분석하고, 3장은 제안하는 방법인 적응적 배경영상을 갱신하는 방법과 그물형 픽셀 간격을 이용하여 객체의 윤곽점 추출방법 및 객체의 최소영역 생성방법을 이용한 움직임 검출방법을 제안한다. 4장에서는 제안한 방법으로 실험한 결과를 기술하고, 마지막으로 5장에서는 결론과 향후 연구 방향을 기술한다.

Ⅱ. 움직임정보 검출 방법

움직임 정보는 프레임 단위의 움직임과 각 화소에서의 움직임으로 구분된다. 프레임 움직임은 그 움직임을 추정하여 움직임 벡터를 전송한 다음 프레임의 움직임 보상을 행한다. 그러나 각 화소에서의 움직임 정보는 프레임 움직임만큼 보상된 이전 프레임과 현재 프레임과의 차이가 큰지 작은지를 나타내는 움직임 검출 정보로서 표현된다.

움직임을 검출하기 위한 방법에는 차영상을 이용한 방법, 블록정합기법, 배경영상방법 등이 있다.

2.1 차영상을 이용한 방법

움직임이 있는 후보영역을 검출하기 위해 연속된 두 프레임간의 차영상 분석 방법을 사용한다. 그림 1의 차영상 분석 방법은 연속된 두 프레임간의 밝기 차이를 구한 후, 임계값을 사용하여 임계값 보다 낮은 밝기 차이를 가진 부분은 움직임이 없는 배경으로 구별하고 임계값 보다 큰 밝기 차이를 가진 부분은 움직임이 있는 물체로써 구별한다[3].

이러한 임계값의 선택은 획득한 영상 안의 잡음뿐만 아니라 시간에 따라 변하는 조명에 상당히 의존적이다. 따라서 움직임이 있는 후보영역들을 검출

하기 위한 임계값은 유동적으로 선택되어야 한다. 또한 물체가 정지해 있는 상황에서는 배경으로 인식될 수 있다.

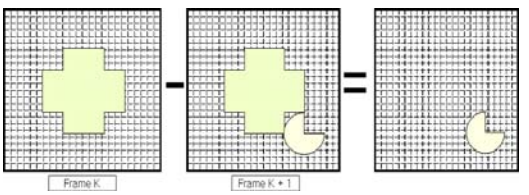


그림 1. 차영상 기법

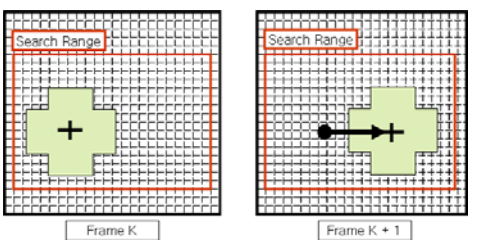


그림 2. 블록 정합 기법을 이용한 움직임 검출

2.2 블록정합 기법

블록정합 기법은 현재프레임 탐색영역 안에서 이전프레임의 지정된 블록과 가장 유사한 블록을 찾는 방법이다. 그림 2와 같이 객체가 움직이지 않다가 다시 움직이는 경우에도 추적이 가능 하고 블록의 크기와 추적할 객체를 지정할 수 있다[4].

블록정합기법에는 전역탐색 알고리즘과 계층적 블록알고리즘이 흔히 사용된다. 전역탐색 알고리즘은 영상의 밝기값 분포가 비교적 균일한 영역이 없는 곳에서 사용된다. 밝기값 분포가 균일한 영역에서는 부정확한 정합가능성이 발생될 수 있다. 계층적 블록알고리즘은 모든 레이어에 동작백터를 적용함으로써 정확한 움직임 검출을 할 수 있다. 그러나 부정합오류가 상층 레이어로부터 전파되어 정합오류가 증가 될수 있고, 각 레이어로 진행될수록 시간복잡도는 계속 증가하게 된다.

2.3 배경영상을 이용한 방법

배경영상을 이용한 방법은 현재 프레임과 기준이 되는 배경 영상의 차이를 구하는 방법이다. 그림 3과 같이 차영상 방법과 같이 인접한 두 프레임을 비교하는 것이 아니라 이전 프레임들로부터 배경이 되는 영상을 추출하고 이 영상과 현재 프레임을 비교하여 모션을 추출하게 된다[5]. 매번 프레임을 검사하면서 기준이 되는 배경 영상은 오래 전 프레임

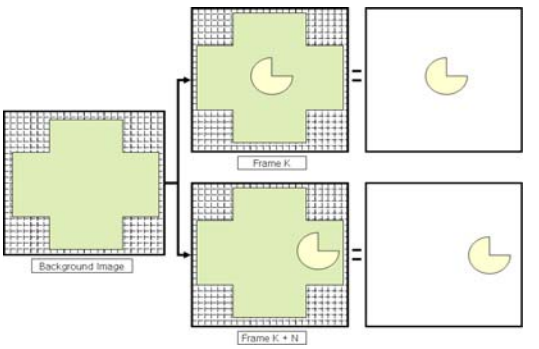


그림 3. 배경영상 기법을 이용한 객체추출

의 영향을 줄이고 현재 프레임의 영향을 추가시키는 방법으로 특정한 방법에 따라 계속 수정된다. 이 방법에서 많이 사용되는 것으로 시간적 평활법과 시간적 중간치법으로 들 수 있다.

시간적 평활법은 기본적으로 배경 영상을 만들 때 이전 프레임들의 화소값을 평균하는 방법을 사용하는 것이다. 현재 프레임에서 임의의 화소의 배경 영상을 만들기 위해서는 현재 프레임 이전의 n 개의 프레임들의 화소값이 필요하며, 이들을 화소값을 모두 더해서 n으로 나누면 $n-Frame$으로 평활화 된 현재 프레임의 배경 영상이 만들어지게 된다. 그러나 이 방법은 이전 프레임의 정보를 기억하기 위한 메모리 낭비가 많고, 또한 배경 영상을 만들 때 최근 프레임과 이전의 프레임의 화소값이 같이 나타나는 문제점이 있다.

시간적 중간치법은 임의의 화소에서 이전 프레임에 나타난 값들 중에서 빈도가 높은 값을 배경영상으로 사용한다. 현재 프레임에서 임의의 화소의 배경 영상을 만들기 위해서는 현재 프레임 이전의 n 개의 프레임들의 화소값이 필요하고, 이 화소값을 크기순으로 정렬하여 $\frac{2}{n}$ 번째 크기의 화소값을 배경영상으로 사용하는 방법이다. 그러나 시간적 중간치법은 정확한 미디언 값을 추출하기 위해서는 많은 이전 프레임을 저장해야 해야 하는 문제점이 있다.

Ⅲ. 실시간 움직임 검출

3.1 제안 방법

제안하는 움직임 검출 방법은 카메라로부터 획득되는 초기 영상 이미지를 배경영상 이미지로 선택한다. 배경 영상 이미지는 움직임이 없는 상태에서 획득된 순수한 배경이미지이다.

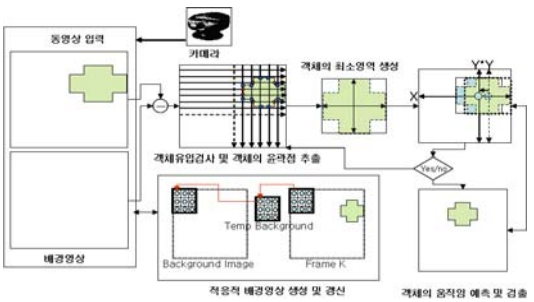


그림 4. 제안하는 움직임 검출 방법

배경영상과 카메라로부터 입력되는 영상과의 차를 이용하여 임계값이 존재하면 이미지 검사를 통하여 움직인 객체의 윤곽선에 해당하는 부분을 찾고 움직임 검출여부를 판단하게 된다.

초기의 배경영상은 시간이 지남에 따라 조명효과나 잡음에 의해서 이미지 자체가 변화하게 된다. 따라서 초기의 배경영상만으로 입력영상과의 차를 구하게 되면 객체의 유입이나 움직임이 없어도 움직임이 있는 것처럼 인식하게 된다.

따라서 정확한 움직임 검출을 위해서 배경영상을 갱신한다. 그림 4는 제안하는 방법을 이용한 움직임 검출 과정을 나타내고 있다.

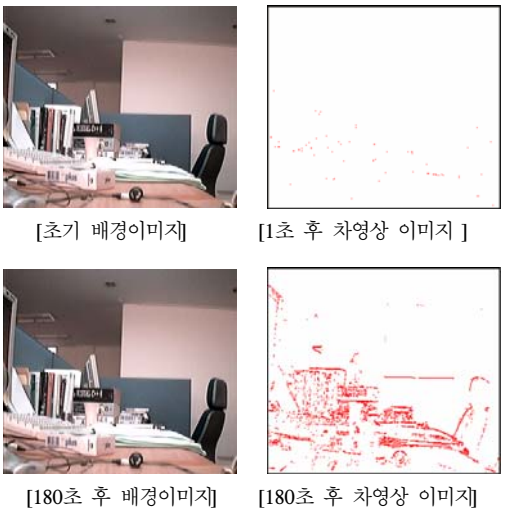
본 논문에서 제안하는 움직임 검출 과정은 다음과 같다.

단계 1. 카메라로부터 입력영상과 배경영상 획득
단계 2. 배경영상과 입력영상에서 밝기의 변화가 있는지 검사하고 변화가 있으면 배경영상을 픽셀단위로 입력영상의 배경에 해당하는 부분과 교체
단계 3. 배경영상과 입력영상의 차영상 획득 및 객체의 유입 및 윤곽점 검출
단계 4. 객체의 최소영역 생성
단계 5. 객체의 움직임 예측 및 검출

3.2 적응적 배경영상 획득

카메라의 입력 영상에 의해 얻어지는 이미지는 초기에 배경영상을 얻기 위해 물체의 변화나 이미지의 변화가 없는 것으로 간주한다. 배경영상은 카메라로부터 입력받은 후부터 시간이 지남에 따라 객체가 검출되지 않는 상황이더라도 배경영상 자체가 변하게 된다.

변화가 없는 공간에서 조명의 변화 등과 같은 미세한 움직임에도 차 영상을 통하여 초기의 배경화면과 현재의 배경화면을 비교해보면 이미지 내에 잡음이 발생함을 할 수 있다. 이것은 시간이 지남에 따라 더 많은 잡음이 발생한다.



[초기 배경이미지] [1초 후 차영상 이미지]

[180초 후 배경이미지] [180초 후 차영상 이미지]

그림 5. 배경영상의 변화

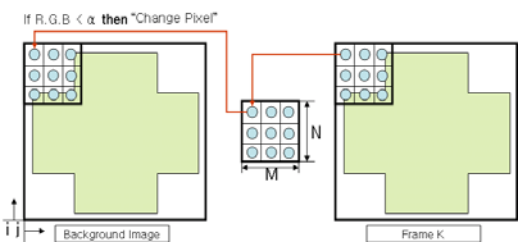


그림 6. 배경영상 갱신

그림 5는 배경영상의 시간에 따른 변화를 보여준다.

따라서 객체의 움직임이 없는 배경영상 임에도 불구하고 객체가 움직이는 것과 같은 오류를 발생한다.

제안하는 방법은 이러한 잡음을 제거하여 보다 정확한 움직임 검출을 위해서 배경영상을 갱신하는 방법을 사용한다. 전체적인 배경영상의 갱신은 많은 연산량을 필요로 하기 때문에 객체의 움직임을 검출하는데 시간이 많이 소요된다. 따라서 N × M 마스크를 사용하여 계속적으로 배경영상을 갱신하면서 객체의 움직임을 정확하게 판별할 수 있도록 한다.

그림 6은 입력영상 "Frame K"의 N × M 마스크 내에서 RGB 채널의 값과 배경영상 "Background Image" 와의 차이값이 α 보다 크면 배경영상이 변경된 것으로 해당 픽셀을 입력영상의 픽셀로 변경한다.

식(1)에서 RGB 채널 각각의 차이값이 임계값 α 보다 작으면, 객체가 유입된 상황이 아니라고 판단

하고 입력영상에서 픽셀을 선택하여 배경영상을 갱신하여 준다. 임계값 α 보다 클 경우는 새로운 객체가 영상 안에 유입된 경우이므로 배경영상의 픽셀을 그대로 사용한다.

$$R_{Channel} = abs(RValue(BackGround_{RImageMask}[i,j] - RValue(InputImage_{RImageMask}[i,j])))$$

$$G_{Channel} = abs(GValue(BackGround_{GImageMask}[i,j] - GValue(InputImage_{GImageMask}[i,j])))$$

$$B_{Channel} = abs(BValue(BackGround_{BImageMask}[i,j] - BValue(InputImage_{BImageMask}[i,j])))$$

$$\text{if}((R_{Channel} < \alpha)\ or\ (G_{Channel} < \alpha)\ or\ (B_{Channel} < \alpha))$$
$$Select\ InputImage[i,j]$$
$$else$$
$$Select\ BackGroundImage[i,j] \quad (1)$$

3.3 객체의 유입검사 및 윤곽점 추출

카메라를 통하여 입력되는 영상이미지는 배경영상의 RGB 체널과 비교하여 수시로 배경영상을 갱신하거나 객체의 유입을 검사한다. 객체가 유입이 되는 시점은 Ⅲ-2절에서 임계값 α 에 따라 객체가 유입되었음을 알 수 있다.

객체가 유입된 영상이미지는 픽셀검사를 통하여 객체의 위치를 검출하고 위치 변화에 따라 움직임을 검출하게 된다. 영상이미지 전체를 픽셀단위로 검색을 한다면 실시간으로 추출되는 이미지를 모두 검사하기에는 많은 연산량을 요구하게 된다.

따라서 영상이미지의 전체 픽셀을 연산에 참여시키지 않고 효과적으로 객체의 위치를 검출하기 위하여 픽셀간격(Pixel Space)을 설정하고 1차, 2차 검사를 수행한다.

그림 7에서 객체의 위치 검출은 실험적을 통하여 추출된 값인 일정한 간격인 Ω만큼의 픽셀간격을 두고 검사를 진행한다. 1차 검사를 통해 객체의 가로 위치값을 최소, 최대형태로 저장하고, 2차 검사를 통해 객체의 세로 위치값을 최소, 최대형태로 저장한다. 이때 객체의 윤곽선을 결정하는 과정은 배경영상의 차이값으로 확인한다.

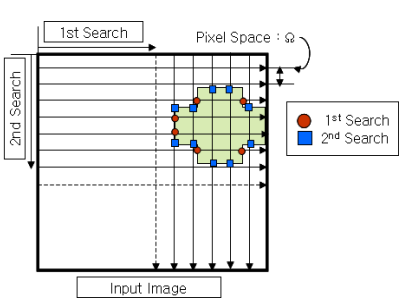


그림 7. 픽셀간격을 이용한 객체의 유입검사 및 윤곽선 검출

픽셀 검사에서 배경영상과 입력영상의 R,G,B 픽셀간 차이가 임계값 β 값을 넘거나 같은 값을 가지면, 식(2)에 따라 객체가 있는 것으로 간주한다.

$$R_{Object} = abs(RValue(BackGround_{RImage}[i,j] - RValue(InputImage_{RImage}[i,j])))$$
$$G_{Objectl} = abs(GValue(BackGround_{GImage}[i,j] - GValue(InputImage_{GImage}[i,j])))$$
$$B_{Object} = abs(BValue(BackGround_{BImage}[i,j] - BValue(InputImage_{BImage}[i,j])))$$

$$\text{if}((R_{Object} \geq \beta)\ or\ (G_{Object} \geq \beta)\ or\ (B_{Object} \geq \beta))$$
$$Object[i,j] = ture$$
$$else$$
$$Object[i,j] = false \quad (2)$$

그리고 R,G,B 값만으로는 명암도의 변화에도 민감한 반응을 보이기 때문에 식(3)에 의해 H,S,I 로의 변환과정을 통해 2차 검사를 하게 된다[6].

$$I = 0.3R + 0.59G + 0.11$$
$$V_1 = R - I = 0.7R - 0.59G - 0.11B$$
$$V_2 = B - I = -0.3R - 0.59G + 0.89B$$
$$H = \tan^{-1}\left(\frac{V_1}{V_2}\right),\ S = \sqrt{V^2_{+V^2_2 1}} \quad (3)$$

색상 변환식을 사용하여 각 채널에 대한 색조, 명도, 채도에 대한 값을 구한다[7,8].

H는 0~180° 의 값이 나오는데, B>G 일 경우 H = 360° - H 한다. S는 0~1 사이의 값을 얻게 되는데, H와 S는 다시 0~255 사이의 값을 갖도록 정규화 한다.

R,G,B는 약 1,600만 가지 색상으로 표현된다 H,S,I로 변환 과정은 대표색상 테이블을 이용하여 색조, 채도, 명도의 범위에 해당하면 하나의 대표색상으로 사용한다[9].

대표색상은 표 1과 같이 한국 공업규격에서 정하고 있는 유채색 10가지와 무채색 5가지를 합한 15가지의 색으로 표현하였다. 여기에서 15가지 대표색상으로 표현한 것은 하나의 방법이며 다른 표현 방법을 채택할 수도 있다. 예를 들면 명도 값으로 표현하여 그 명도값이 임계값 범위 일 때 각 픽셀의

명도 평균값을 사용하여 그레이 색상으로 표현할 수도 있다.

식(3)을 적용하여 H,S,I에 대한 픽셀간 차이값을 구하여 만족하면 객체가 유입된 것으로 최종 판단하게 된다.

표 1. 대표색상 매핑 테이블

대표색상	색조	채도	명도
빨강	$0 \le H \le 18$ $342 \le H \le 360$	$S \neq 0$	$0.3 \le I \le 0.9$
주황	$18 \le H \le 54$	$S \neq 0$	$0.3 \le I \le 0.9$
노랑	$54 \le H \le 90$	$S \neq 0$	$0.3 \le I \le 0.9$
연두	$90 \le H \le 126$	$S \neq 0$	$0.3 \le I \le 0.9$
녹색	$126 \le H \le 162$	$S \neq 0$	$0.3 \le I \le 0.9$
청록	$162 \le H \le 198$	$S \neq 0$	$0.3 \le I \le 0.9$
파랑	$198 \le H \le 234$	$S \neq 0$	$0.3 \le I \le 0.9$
남색	$234 \le H \le 270$	$S \neq 0$	$0.3 \le I \le 0.9$
보라	$270 \le H \le 306$	$S \neq 0$	$0.3 \le I \le 0.9$
자주	$306 \le H \le 342$	$S \neq 0$	$0.3 \le I \le 0.9$
흰색	~	$S \simeq 0$	$0.85 \le V \le 1$
밝은 회색	~	$S \simeq 0$	$0.65 \le I \le 0.85$
회색	~	$S \simeq 0$	$0.45 \le I \le 0.65$
어두운 회색	~	$S \simeq 0$	$0.25 \le I \le 0.45$
검정색	~	~	$0 \le I \le 0.25$

3.4 최소영역을 통한 객체의 움직임예측

픽셀간격을 통하여 추출한 객체의 위치 좌표를 이용하여 객체의 최소 영역을 설정한다. 객체의 최소영역은 움직임을 미리 예측하여 효율적인 움직임 검출을 위하여 생성한다.

그림 8은 객체의 최소영역을 설정하는 방법이다. 1차 검사 후 추출된 객체의 윤곽점 $x_1, x_2, \ldots, x_n$ 의 각각의 좌표값을 저장하고, 동일한 스캔라인에 위치한 좌표값들과의 거리를 측정하여 객체의 최소영역의 최대 가로길이 값을 추출한다. 세로길이 값은 2차 검사 후 추출된 윤곽점인 $y_1, y_2, \ldots, y_n$의 좌표점을 계산하여 최대 세로길이 값을 추출하여 객체의 최소영역을 생성한다.

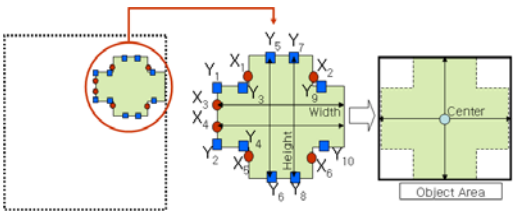


그림 8. 객체의 최소영역 설정

식(4)는 객체의 최소영역인 사각형 블록을 설정한다.

$$\begin{aligned} Object_{X_{Large}} &= Max(x_1, x_2, \ldots, x_n), Object_{X_{Small}} \\ &= Min(x_1, x_2, \ldots, x_n) \\ Object_{Y_{Large}} &= Max(y_1, y_2, \ldots, y_n), Object_{Y_{Small}} \\ &= Min(y_1, y_2, \ldots, y_k) \\ Object_{Rect} &= RECT(Object_{X_{Small}}, \\ & Object_{Y_{Small}}, Object_{X_{Large}}, Object_{Y_{Large}}) \end{aligned} \quad (4)$$

최소영역인 사각형 블록이 설정되면 중심좌표를 계산한다. 중심좌표는 객체가 다음 프레임에서 움직일 수 있는 방향성을 예측하고 사각형 블록의 위치를 갱신하기 위하여 사용된다. 이러한 방향성 예측은 객체의 움직임을 빠르게 검출할 수 있고 다음 프레임인 입력영상 이미지의 객체유입 시점을 다시 검사하지 않기 위함이다. 그러므로 객체가 존재하는 시간동안에는 매 프레임 마다 픽셀간격을 사용하여 검사하지 않고도 정확하게 움직임을 검출 할 수 있다.

식(5)는 최소영역인 사각형 블록의 중심좌표를 계산한다. $Object_{Width}$ 는 가로방향의 최대 길이 값이고, $Object_{Height}$ 는 세로방향의 최대길이 값이다.

$$Object_{Center}(x, y) = [Object_{Width}/2, Object_{Height}/2] \quad (5)$$

객체의 최소영역인 사각형 블록과 중심좌표를 이용하여 객체의 이동방향 예측하는 과정은 그림9와 같이 X축 좌표의 연장선 방향으로 1픽셀단위로 검사하고 객체가 이동하였다면 사각형 블록의 위치좌표를 수정해 준다. Y축 좌표의 연장선 방향은 X축 연장선 방향으로 이동하지 않았을 경우 객체의 이동방향을 예측하게 된다.

객체의 이동방향의 예측 순서는 객체가 유입되는 X축을 중심으로 진행방향의 우선순위가 주워진다. 그림 9에서와 같이 우측에서 좌측으로 유입되는 객

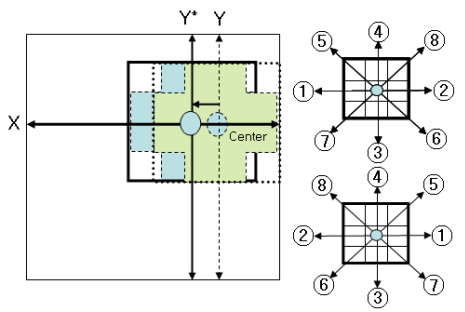


그림 9. 객체의 이동방향 예측

체의 우선순위는 좌측방향에 우선순위가 주워지고 좌측에서 유입되면 우측방향에 우선순위가 주워진다. Y축을 중심으로 움직이는 객체는 X축과 같은 방법으로 진행방향에 우선순위를 주어지게 된다.

Ⅳ. 실험 결과 및 분석

4.1 실험 환경

본 실험은 명암도의 변화가 너무 크지 않은 오후 시간대에 웹 카메라를 사용하여 배경영상과 입력영상을 실시간으로 처리하여 실험하였다.

시스템은 Intel Pentium 4 CPU 1.60GHz, 256M RAM의 PC에서 Visual C++ 6.0(Service Pack 5)을 이용하여 구현하였다. 배경영상과 입력영상은 전송을 고려하여 320×240의 RGB 24bit 컬러 영상을 이용하였다.

4.2 실험 결과

그림 10은 제안하는 움직임검출 방법을 사용하여 구현한 시스템이다. 초기 카메라에서 입력되는 영상은 초당 15프레임으로 설정하여 사용한다. 그림의 좌측 하단의 그래프는 움직임을 검출하였을 경우 생성된다. 움직임 검출시 각 실험임계값과 객체의 움직임 값은 리스트컨트롤을 사용하여 중단부에 표시하도록 하였다.

기존의 블록정합 방법은 객체를 블록 안에 있다는 가정 하에 움직임을 검출하기 때문에 블록 밖으로 객체가 유입되면 초기의 블록과의 유사성을 찾지 못하기 때문에 객체를 인식하지 못하는 오류가 있고, 차 영상만을 이용할 때는 잡음에 민감한 반응을 보이기 때문에 움직임 객체가 입력영상에 없다고 하더라도 움직임으로 검출하는 경우가 있었으나, 본 논문에서 제안한 적응적 배경영상을 이용하여 움직임을 검출한 결과 움직임 객체가 있을 경우만을 검색하고, 픽셀을 모든 연산에 포함시키지 않고 그물형 픽셀간격을 이용하여 객체의 유입을 검색하고 윤곽점을 추출하고, 객체의 최소영역을 생성하여 방향성과 움직임을 예측할 수 있기 때문에 픽셀 스캔을 다시 하지 않고도 객체의 정확한 움직임과 빠른 검출이 가능하다.

그림 11은 카메라를 통하여 보여지는 영상과 객체의 유입과 움직임을 쉽게 알 수 있도록 그래프를

그림 10. 제안하는 움직임 검출방법을 사용한 시스템

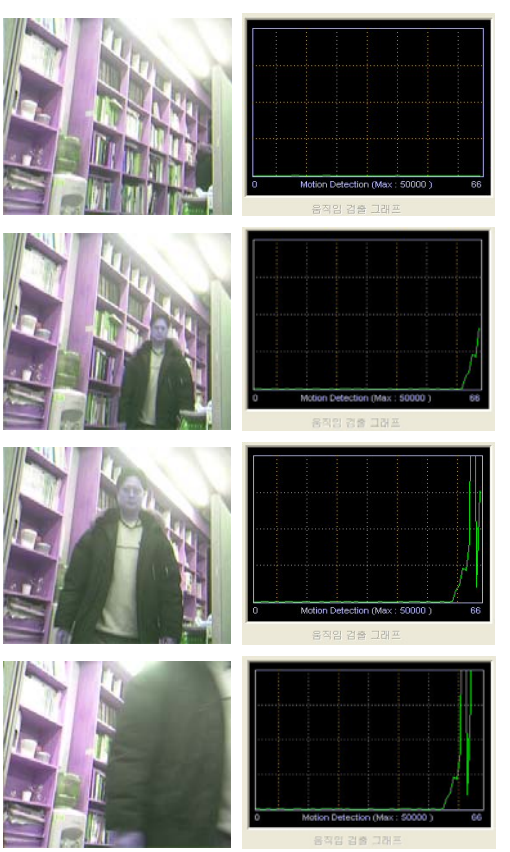


그림 11. 객체의 유입과 움직임 그래프

사용하여 표현하였다. 배경영상의 미세한 변화율까지 표현할 수 있도록 이미지 스캔과정에서 추출되는 객체의 윤곽선을 이용한 최소영역은 빠르게 움직임 검출을 할 수 있다.

표 2는 본 논문에서 제안한 방법을 위한 임계값들과 변수에 대한 최적화 실험 결과이다. 적응적 배경영상 생성 임계값은 실험값이 5인 0.210 일 때, 배경영상 이미지내의 잡음을 제거 할 수 있었다.

픽셀 간격 임계값은 조밀할수록 움직임 추출시 객체 인식이 잘 이루어 졌지만, 많은 연산량을 요구하기 때문에 보다 빠른 탐색을 위해 실험 값 5를 선택하여 실험하였다.

표 2. 임계값과 변수에 대한 최적화

실험값	적응적 배경영상 생성 임계값(α)	움직임 검출을 위한 픽셀 간격 임계값(β)
1	0.205	0.951
2	0.151	0.903
…	…	…
5	0.210	0.851
…	…	…
25	0.915	0.595
…	…	…
30	0.887	0.496
…	…	…

표 3은 실험시간을 오후 1시부터 10시까지 정해서 1시부터 3시, 7시부터 10시까지는 객체의 유입이 없는 시간대로 설정하고, 4시부터 6시까지는 객체를 유입하여 움직임을 검출하는 빈도수로 실험한 결과이다.

실험은 차영상 만을 이용하여 움직임을 검출하는 방법과 배경영상을 이용하여 움직임을 검출하는 방법, 그리고 제안하는 방법인 적응적 배경영상과 그물형 픽셀간격을 이용한 방법을 각각 적용하여 실시간으로 측정하였다.

실험결과 차영상 만을 이용하여 움직임을 검출하는 방법 A는 객체의 유입이나 움직임이 없는 1시부터 3시까지의 시간대에 미세한 환경의 변화에도 민감하여 움직임이 있는 것으로 판단하고 움직임을 검출하고 있다.

배경영상을 이용하는 방법 B는 초기 배경영상을 획득하고 N 프레임의 영상의 차를 구하여 객체의 유무를 판단하게 된다. 표 3에서 1시와 3시 사이는 차영상 방법보다 오류검출이 적지만 시간이 지남에 따라 오류 검출횟수가 증가 하고 있다.

- 방법 A : 차영상만을 이용한 움직임 검출
- 방법 B : 배경영상을 이용한 움직임 검출
- 방법 C : 적응적 배경영상과 그물형 픽셀간격을 이용한 움직임 검출

표 3. 움직임 검출 빈도수 및 검출 오류

시간 (오후)	방법 A		방법 B		방법 C	
	빈도수	오류	빈도수	오류	빈도수	오류
1	5	5	3	3	0	0
2	7	7	4	4	1	1
3	4	4	6	6	1	1
4	25	4	28	7	22	1
5	24	6	26	6	20	0
6	26	5	28	7	21	0
7	10	10	19	19	2	2
8	6	6	9	9	1	1
9	7	7	10	10	2	2
10	4	4	11	11	1	1

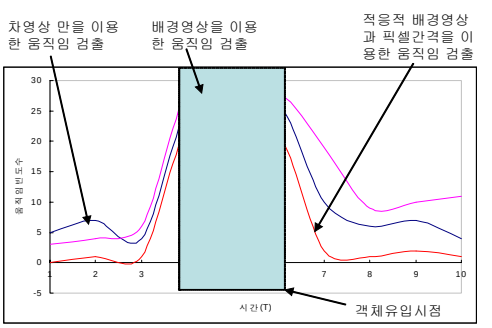


그림 12. 움직임 검출 빈도수 비교

반면에 적응적 배경영상과 그물형 픽셀간격을 이용한 움직임 검출에서는 객체가 유입된 시점인 4시부터 6시까지에 움직임을 검출하고 다른 시간대에는 조명의 변화와는 상관없이 움직임을 거의 검출하지 않음을 알 수 있다.

그림 12는 제안하는 움직임 검출방법과 기존의 방법의 움직임 검출 빈도수를 비교한 그래프이다.

V. 결 론

블록정합기법은 블록내의 객체의 움직임 측정이

나, 정지하였다가 다시 움직이는 객체의 움직임 검출을 정확히 추출할 수 있다. 그러나 탐색영역 밖으로 객체가 유입되었을 경우에는 초기의 블록과 유사한 블록으로 인식하지 못하는 단점을 가지고 있다.

배경 영상방법은 프레임마다 현재 프레임에서 배경 영상을 빼는 방법이다. 움직이는 물체에 대한 정확한 위치 정보와 형태 정보를 얻을 수 있다. 그러나 배경이미지 자체는 시간에 따라 계속적으로 변하기 때문에 정확한 움직임을 검출하는데 어려움이 있다.

본 논문에서는 초기의 배경영상을 기준으로 입력영상과의 차를 구하고 시간에 따라 변화하는 배경영상을 N×M 픽셀 마스크만큼 교체하여 배경영상을 갱신해서 기존의 배경영상기법에서 배경영상 자체의 변화 때문에 생기는 문제점을 해결하였다.

이미지 픽셀 검사는 모든 픽셀을 연산에 참여시키지 않고 일정한 간격을 두고 이미지의 픽셀을 검색하여 실시간으로 처리되는 실제적인 연산량을 줄이고 정확도를 높일 수 있었다.

제안하는 적응적 배경영상 기법과 그물형 픽셀간격을 이용한 실시간 움직임 검출방법은 기존의 방법보다 정확한 움직임을 검출할 수 있었다. 설계하고 구현한 시스템은 실험을 통한 성능평가에서 90% 이상의 높은 정확도를 보였다.

이 방법은 영상보안시스템에서 움직이는 객체를 추적하거나, 저장 방법 개선 등에 응용 할 수 있다.

참 고 문 헌

[1] H.Zhang, A.Kankanhalli, S.W.Smoliar, "Automatic Partitioning of Full-motion Video",Multimedia System, Vol. 1, No. 1, pp.10-28,1993.

[2] Andreas Koschan, Sangkyu Kang, Joonki Paik, Besma Abidi, Mongi Abidi,"Color active shape models for tracking non-rigid objects", Pattern Recognition Letters 24, pp.1751-1765, 2003.

[3] J.L. Starck,F. Murtagh, E.J. Candes, D.L.Donoho, "Gray and color image contrast enhancement by the curvelet transform", IEEE Transactions on Image Processing, VOL. 12 NO.06 pp.0706~0717, JUNE 2003.

[4] 김기주,방경구,문정미,김재호,"효율적인 화상회의 동영상 압축을 위한 블록기반 얼굴 검출방식" , *한국통신학회논문지*, VOL 29, No 9C pp.1258~1268, SEPTEMBER 2004.

[5] A.Hanjalic, R.L.Lagendijk, J.Biemond. "A New Key-Frame Allocation Method for Representing Stored Video-Streams", Proc of the First International Workshop on Image Databases and Multimedia Search, Armsterdam of The Netherlands, pp.67-74, 1996.

[6] Y. Wu, D.Suter, "A Comparison of Methods for Scene Change Detection in Noisy Image Sequence.", Proc of the Fist International conference on Visual Information Systems, Melbourne, Australia, pp.459-468, 1996.

[7] N. Ikonomakis, K.NPlataniotis, M.Zervakis, A.N. Venetsanopulos, "Region Growing and Region Merging Image Segmentation", IEEE DSP 97. p299-302, 1997.

[8] S.Y.Wan, W.E.Higgins, "Symmetric region growing", International Conference on Image Processing 2000, VOL.12 NO.09 pp.1007~1015, SEPTEMBER 2003.

[9] 이동근 "색상 및 영역특징 기반 이미지검색 시스템", 숭실대학교 석사학위논문,서울, 1999.

이 창 수(Chang soo Lee) 정회원

1999년 2월 한서대학교 컴퓨터 과학과 졸업

2002년 2월 숭실대학교 컴퓨터 공학과 석사

2004년 2월 숭실대학교 컴퓨터 공학과 박사수료

<관심분야> 영상처리, 멀티미디어 보안, 네트워크 보안

전 문 석(Moon seog Jun) 정회원

1980년 2월 숭실대학교 컴퓨터공학과(학사)
1986년 2월 University of Maryland Computer Science 석사
1989년 2월 University of Maryland Computer Science 박사
1989년 2월 Morgan State University 전산수학과 조교수
1989년~1991년 New Mexico State University 부설 Physical Science Lab. 책임연구원
1991년~현재 숭실대학교 정보과학대학 정교수
<관심분야> 네트워크보안, 암호학, 컴퓨터알고리즘, PKI