능동적 블록정합기법을 이용한 객체의 움직임 검출에 관한 연구

정회원 이 창 수*, 박 미 옥**, 이 경 석***

A Study on Motion Detection of Object Using Active Block Matching Algorithm

Chang-soo Lee*, Mi-og Park**, Kyung-seok Lee*** ***Regular Members***

요 약

카메라를 통한 객체의 움직임 검출은 불필요한 잡음이나 조명의 변화에 따라 정확한 움직임을 검출하는 것은 어렵다. 또한 객체의 유입 후 일정시간 동안 움직임이 없을 경우에는 배경으로 인식될 수도 있다. 따라서 실시간으로 입력되는 영상에서 정확한 객체를 추출하고 움직임을 검출할 수 있는 알고리즘이 필요하다. 본 논문에서는 객체의 정확한 움직임을 검출하기 위한 방법은 시간에 따라 변화하는 배경영상의 일부를 교체하고, 객체가 유입된 시점에서 객체의 영역을 추출하기 위하여 그물형 픽셀검사를 통하여 객체의 윤곽점을 추출한다. 추출된 윤곽점은 객체의 사각영역인 최소블록의 생성과 객체의 움직임을 빠르게 검출하기 위한 가변 탐색블록을 생성하여 정확한 객체의 움직임을 검출한다. 설계하고 구현한 시스템은 실험을 통한 성능평가에서 95% 이상의 높은 정확도를 보였다.

Key Word : Motion Detection, Background Image, Block Matching, Searching Block, Object Boundary Point

ABSTRACT

It is difficult for the movement detection of an object through a camera to detect exact movement because of unnecessary noises and changes of the light. It can be recognized as a background, when there is no movement after the inflow of an object. Therefore, It is necessary to fast search algorithm for tracking and extract of object that is realtime image.

In this thesis, we evaluate the difference of the input vision based on initial image and replace some pixels in process of time. When there is a big difference between background image and input image, we decide it is the point of the time of the object input and then extract boundary point of it.

The extracted boundary point detects precise movement of the object by creating minimum block of it and searching block that maintaining distance. The designed and embodied system shows more than 95% accuracy in the performance test.

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), ** 성결대학교 컴퓨터공학부 (mopark777@hanmail.net)
*** 산업연구원 연구위원 (kslee@kiet.re.kr)
논문번호 : KICS2005-12-503, 접수일자 : 2005년 12월 26일, 최종논문접수일자 : 2006년 3월 24일

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. 또한 동영상 압축 방식에서 비디오 신호의 시간상 중복성을 제거하고 동영상을 압축하기 위해 움직임 보상방식이 요구되고 있다. 움직임 보상방식(Motion Compensation)은 비디오 영상에서 배경과 객체를 구분하여 전송하기 위한 방법으로 움직임 벡터(Motion Vector)를 이용하여 객체의 움직임을 예측한다. 움직임 벡터는 실시간으로 처리해야 하기 때문에 계산량을 현저히 줄일 수 있는 알고리즘을 필요로 한다[2, 3].

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

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 가지게 되므로 정확한 검출은 어렵다. 또한 이러한 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 최소영역인 최소블록을 생성한다. 또한 최소블록과 일정한 거리를 유지하는 탐색블록을 생성하여 매 프레임마다 객체의 움직임을 검출하기 위해 이미지를 전체 스캔하지 않고 탐색블록 내의 최소블록의 방향성을 예측하고 매칭 시켜 보다 빠른 움직임 검출과 정확성을 높인다.

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 가지고, 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 기존의 움직임 검출방법을 분석하고, 3장은 배경영상 갱신과 객체의 윤곽점 추출방법 및 객체의 최소블록 및 탐색블록 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. 그리고 움직임을 검출하기 위한 방법은 다음과 같이 차영상 기법, 블록정합기법, 배경영상 교체기법 등이 있다.

2.1 차영상 기법

움직임이 있는 후보영역을 검출하기 위해 연속된 두 프레임간의 차영상 분석 기법을 사용한다. 그림

1의 차영상 분석 기법은 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[4-6].

이러한 임계값의 선택은 획득한 영상 안의 잡음뿐만 아니라 시간에 따라 변하는 조명에 상당히 의존적이다. 따라서 움직임이 있는 후보영역들을 검출하기 위한 임계값은 유동적으로 선택되어야 하고 물체가 정지해 있는 상황에서는 배경으로 인식될 수 있다.

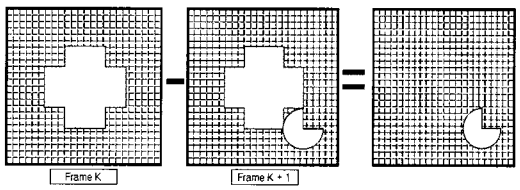


그림 1. 차영상 기법
Fig. 1. Different image algorithm

2.2 블록정합 기법

블록정합 기법은 현재프레임 탐색영역 안에서 이전프레임의 지정된 블록과 가장 유사한 블록을 찾는 방법으로 그림 2와 같이 객체가 움직이지 않다가 다시 움직이는 경우에도 추적이 가능 하고 블록의 크기와 추적할 객체를 지정할 수 있다[7].

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. 그리고 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.

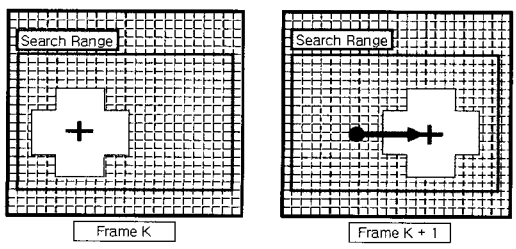


그림 2. 블록 정합 기법
Fig. 2. Block matching algorithm

2.3 배경영상 교체기법

배경영상 교체기법은 현재 프레임과 기준이 되는 배경 영상의 차이를 구하는 방법이다. 그림 3과 같이 차영상 방법과 같이 인접한 두 프레임을 비교하는 것이 아니라 이전 프레임들로부터 배경이되는 영상을 추출하고 이 영상과 현재 프레임을 비교하여 모션을 추출하게 된다[8]. 매번 프레임을 검사하면서 기준이 되는 배경 영상은 오래 전 프레임의 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. 그리고 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 n개의 프레임들의 화소값이 필요하고, 이 화소값을 크기순으로 정렬하여 $\frac{2}{n}$번째 크기의 화소값을 배경영상으로 사용하는 방법이다[9]. 그러나 시간적 중간치법은 정확한 미디언 값을 추출하기 위해서는 많은 이전 프레임을 저장해야 해야 하는 문제점이 있다.

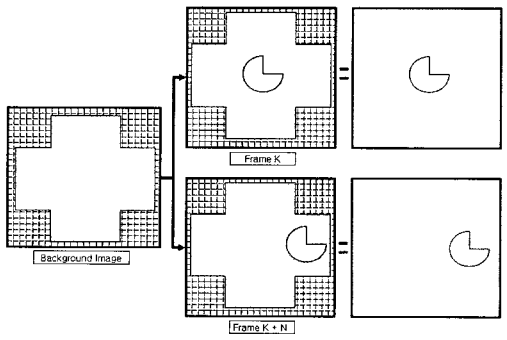


그림 3. 배경영상 교체기법
Fig. 3. Background image change algorithm

Ⅲ. 실시간 움직임 검출

3.1 제안 방법

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

배경영상과 카메라로부터 입력되는 영상과의 차를 이용하여 임계값이 존재하면 이미지 검사를 통하여 움직인 객체의 윤곽선에 해당하는 부분을 찾고 윤곽점의 최대·최소점을 이용하여 객체의 최소영역인 최소블록을 생성하고, 최소블록의 움직임에 따라 항상 일정한 거리를 유지하는 가변적 탐색블록을 생성하여 빠르고 정확한 객체의 움직임 검출을 한다.

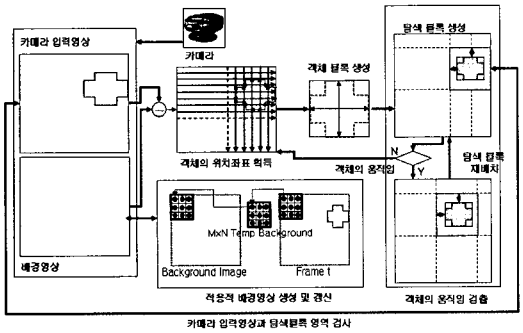


그림 4. 움직임 검출 시스템 구조
Fig. 4. Motion detection system construction

본 논문에서 제안하는 움직임 검출 방법은 그림 4와 같으며, 움직임 검출 과정은 그림 5와 같이 초기 배경영상부터 가변적인 탐색영역을 이용하여 빠르고 정확한 객체의 움직임 검출을 위한 과정이다. 각 과정에 대한 설명은 각 절에서 설명하도록 한다.

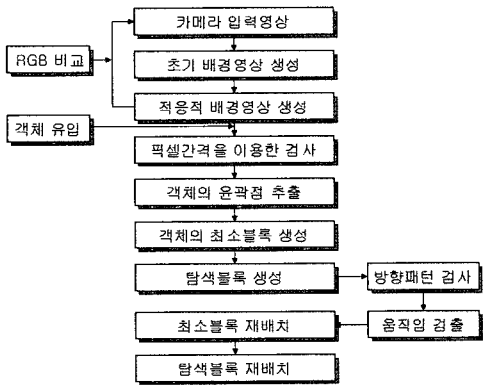


그림 5. 움직임 검출 과정
Fig. 5. Motion detection process

3.2 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. 따라서 객체의 움직임이 없는 배경영상 임에도 불구하고 객체가 움직이는 것과 같은 오류를 발생 한다.

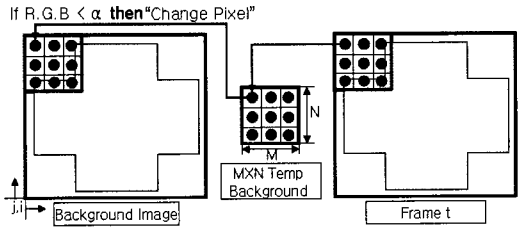


그림 6. 배경영상 갱신
Fig. 6. Background image renewal

본 논문에서는 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 소요되므로 N × M 마스크를 사용하여 계속적으로 배경영상을 갱신하면서 객체의 움직임을 정확하게 판별할 수 있도록 한다.

그림 6은 입력영상 "Frame t"의 N × M 마스크 내에서 RGB 채널의 값과 배경영상 "Background Image"와의 차이값이 임계값 α보다 작으면 배경영상이 변경된 것으로 해당 픽셀을 입력영상의 픽셀로 변경한다.

식(1)에서 RGB 채널 각각의 차이값이 α 보다 작으면, 객체가 유입된 상황이 아니라고 판단하고 입력영상에서 픽셀을 선택하여 배경영상을 갱신하여 준다. α 보다 클 경우는 새로운 객체가 영상 안에 유입된 경우이므로 배경영상의 픽셀을 그대로 사용한다.

$$R_{channel} = \sum_{i=0}^{M}\sum_{j=0}^{N} abs(R_{Value}(Background_{Imgmask}[i,j] -$$

$$R_{Value}(Input_{Imgmask}[i,j]))$$

$$G_{channel}=\sum_{i=0}^{M}\sum_{j=0}^{N}abs(G_{Value}(Background_{Imgmask}[i,j]-$$

$$G_{Value}(Input_{Imgmask}[i,j])))$$

$$B_{channel}=\sum_{i=0}^{M}\sum_{j=0}^{N}abs(B_{Value}(Background_{Imgmask}[i,j]-$$

$$B_{Value}(Input_{Imgmask}[i,j])))$$

$$T_{RGB}=\frac{R_{channel}+G_{channel}+B_{channel}}{3}$$

$$Object_{detect}=\begin{pmatrix}1,\ \text{if}(T_{RGB}>\alpha)(20\le\alpha\le 30)\\ 0,\ otherwise\end{pmatrix} \quad (1)$$

3.3 객체의 유입검사 및 윤곽점 추출

카메라를 통하여 입력되는 영상이미지는 배경영상의 RGB 채널과 비교하여 수시로 배경영상을 갱신하거나 객체의 유입을 검사한다. 객체가 유입이 되는 시점은 배경영상 갱신 중 임계값 α 보다 큰 경우에는 객체가 유입된 시점이다.

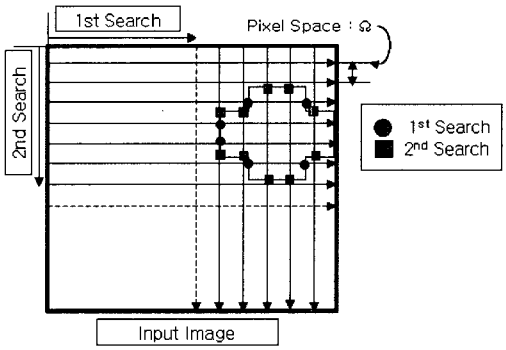


그림 7. 객체의 윤곽점 추출을 위한 그물형 탐색
Fig. 7. A net type search for contours point extraction of object

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.

본 논문에서는 영상이미지의 전체 픽셀을 연산에 참여 시키지 않고 효과적으로 객체의 위치를 검출하기 위하여 일정한 픽셀간격인 Ω를 설정하고 그물형 탐색을 수행한다.

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

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

$$R_{object}=\sum_{i=0}^{Height}\sum_{j=0}^{Width}abs[R_{Value}(Background_{Img}[i,j]_{\Omega}$$

$$-R_{Value}(Input_{Img}[i,j]_{\Omega}))]$$

$$G_{object}=\sum_{i=0}^{Height}\sum_{j=0}^{Width}abs[G_{Value}(Background_{Img}[i,j]_{\Omega}$$

$$-G_{Value}(Input_{Img}[i,j]_{\Omega}))]$$

$$B_{object}=\sum_{i=0}^{Height}\sum_{j=0}^{Width}abs[B_{Value}(Background_{Img}[i,j]_{\Omega}$$

$$-B_{Value}(Input_{Img}[i,j]_{\Omega}))]$$

$$T_{RGB}=\frac{R_{object}+G_{object}+B_{object}}{3}$$

$$Object(i,j)=\begin{pmatrix}1,\ \text{if}(T_{RGB}>\beta)(40\le\beta\le 50)\\ 0,\ otherwise\end{pmatrix} \quad (2)$$

3.4 최소블록 생성

픽셀간격을 통하여 추출한 객체의 위치 좌표를 이용하여 객체의 최소블록을 설정한다. 객체의 최소블록은 움직임을 효율적으로 검출하기 위해서 생성한다. 그리고 그림 8은 객체의 최소영역을 설정하는 방법이다. 1차 검사 후 추출된 객체의 윤곽점 $x_1,x_2,\ldots,x_n$의 각각의 좌표를 저장하고, 동일한 스캔라인에 위치한 좌표와의 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 최소블록을 생성한다.

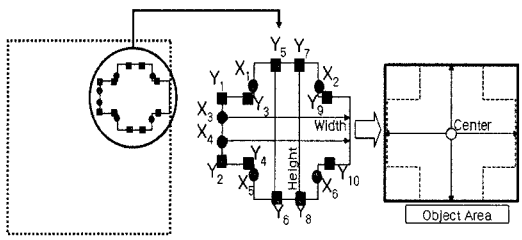


그림 8. 객체의 최소영역 설정
Fig. 8. The minimum area of object

식(3)는 객체의 최소영역인 사각형 블록을 설정된다.

$$Object_{X_{Large}} = Max(x_1, x_2, \cdots, x_n), Object_{X_{Small}} = Min(x_1, x_2, \cdots, x_n)$$
$$Object_{Y_{Large}} = Max(y_1, y_2, \cdots, y_n), Object_{Y_{Small}} = Min(y_1, y_2, \cdots, y_k)$$
$$Object_{Rect} = RECT(Object_{X_{Small}}, Object_{Y_{Small}}, Object_{X_{Large}}, Object_{Y_{Large}}) \quad (3)$$

최소영역인 사각형 블록이 설정되면 중심좌표를 계산하며 중심좌표는 객체가 다음 프레임의 탐색블록에서 움직일 수 있는 방향성을 예측하고 최소블록의 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 그물형 검사를 사용하여 윤곽점을 추출할 필요 없이 탐색블록 내에서 움직임을 검출할 수 있다.

최소영역인 사각형 블록의 중심좌표는 식(4)와 같으며 $Object_{Width}$는 가로방향의 최대 길이 값이고, $Object_{Height}$는 세로방향의 최대길이 값이다.

$$Object_{Center}(x,y) = [Object_{Width}/2, Object_{Height}/2] \quad (4)$$

3.5 탐색블록 생성 및 움직임 검출

객체의 윤곽점 추출을 통하여 생성된 최소블록을 기본으로 생성하며, 카메라 입력영상 이미지와 현재 설정된 최소블록 내의 객체의 움직임을 검출하기 위해서는 일정한 탐색영역을 설정해야 한다.

제안하는 시스템의 탐색영역은 객체의 최소블록을 기반으로 생성한다. 만약 최소블록의 위치가 이미지의 하단부분이면 하단부분부터 검색을 할 수 있도록 설정하고 좌측상단 부분이면 좌측상단부분부터 움직임을 검출할 수 있도록 하며, 제안하는 시스템의 움직임 탐색영역인 탐색블록을 설정하는 과정은 그림 9와 같다.

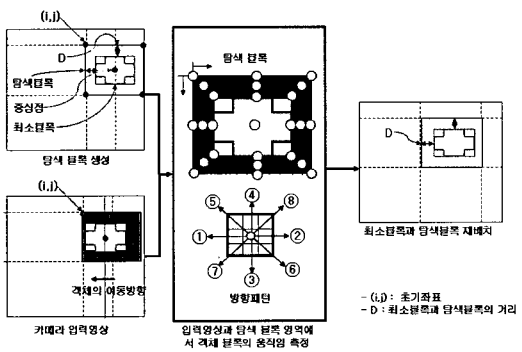


그림 9. 탐색블록을 이용한 움직임 검출
Fig. 9. Motion detection using search block

카메라를 통하여 입력되는 입력 영상이미지와 탐색블록의 탐색영역과 매칭하는 과정과 방향 패턴을 사용하여 움직임을 검사하는 과정은 그림 9와 같으며 탐색영역인 탐색블록의 초기 좌표 (i,j)는 카메라 입력영상의 (i,j)의 위치가 탐색영역의 초기 좌표가 된다. 설정된 최소블록과 입력영상의 탐색영역 내에서 객체의 이동성을 검사하고 이 검사는 객체 블록의 네 변의 중심점을 중심으로 그림 9의 방향 패턴을 사용하여 우선순위를 정하여 검사한다.

설정된 입력영상의 탐색영역 내의 초기좌표인 (i,j)부터 픽셀의 RGB 값과 이전 프레임의 탐색영역의 RGB 값의 차이 값을 비교하여 최소블록이 이동된 경우이면, 최소블록과 탐색블록을 이동된 위치만큼 갱신한다.

탐색블록인 $RGB_{Object}(i,j)$과 입력영상의 탐색블록인 $RGB_{Input}(i,j)$의 RGB값을 비교하여 식(5)와 같이 이동한 좌표인 $Move_{position}(i,j)$에 저장한다. 그리고 입력영상의 탐색블록과 비교하여 움직임이 검출된 경우에는 탐색블록과 객체블록과의 거리가 D만큼을 유지 하지 않기 때문에, 탐색블록과 객체블록과의 거리가 D가 될 수 있도록 재배치하는 과정은 식(6)과 같다.

$$Move_{position}(i,j) = \sum_{i=y_{Rect}}^{Search_y} \sum_{j=x_{Rect}}^{Search_x} (RGB_{Object}(i,j) - RGB_{Input}(i,j)) \quad (5)$$

$$\begin{cases} Search_{RECT}(Object_{x_{\min}} + x_1 - D, Object_{x_{\max}} + x_2 + D, \\ \qquad Object_{y_{\min}} + y_1 - D, Object_{y_{\max}} + y_2 + D) \\ Search_{RECT}(Object_{x_{\min}} - D, Object_{x_{\max}} + D, \\ \qquad Object_{y_{\min}} - D, Object_{y_{\max}} + D) \end{cases} \quad (6)$$

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 6)을 이용하여 구현하였으며, 배경영상과 입력영상은

전송을 고려하여 320×240의 RGB 24bit 컬러 영상을 이용하였다.

4.2 실험 결과

제안하는 움직임검출 방법을 사용하여 구현한 시스템은 그림 10과 같으며, 초기 카메라에서 입력되는 영상은 초당 15프레임으로 설정하여 사용한다. 그림의 좌측 하단의 그래프는 움직임을 검출하였을 경우 생성되고 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 있었다.

그러나 제안한 방법을 이용하여 움직임을 검출한 결과 움직임 객체가 있을 경우만을 검색하고, 픽셀을 모든 연산에 포함시키지 않고 그물형 픽셀간격을 이용하여 객체의 유입을 인식한다. 그리고 윤곽점을 추출하여 객체의 최소블록과 탐색블록을 생성하여 객체의 방향성과 움직임을 예측할 수 있기 때문에 객체의 정확한 움직임과 빠른 검출이 가능하였다.

카메라를 통하여 보여지는 영상과 객체의 유입과 움직임은 그림 11과 같으며, 쉽게 알 수 있도록 그래프를 사용하여 표현하였다. 조명의 변화나 잡음을 알 수 있도록 그래프에서 모두 표현하도록 하였고, 객체의 움직임이 검출되었을 경우에는 파형이 큰 형태를 표현 할 수 있도록 하였다.

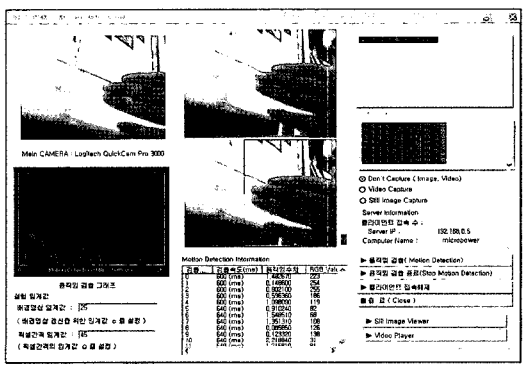


그림 10. 제안한 능동적 블록정합기법을 이용한 움직임 검출 시스템
Fig. 10. Motion detection extraction system using active block matching algorithm

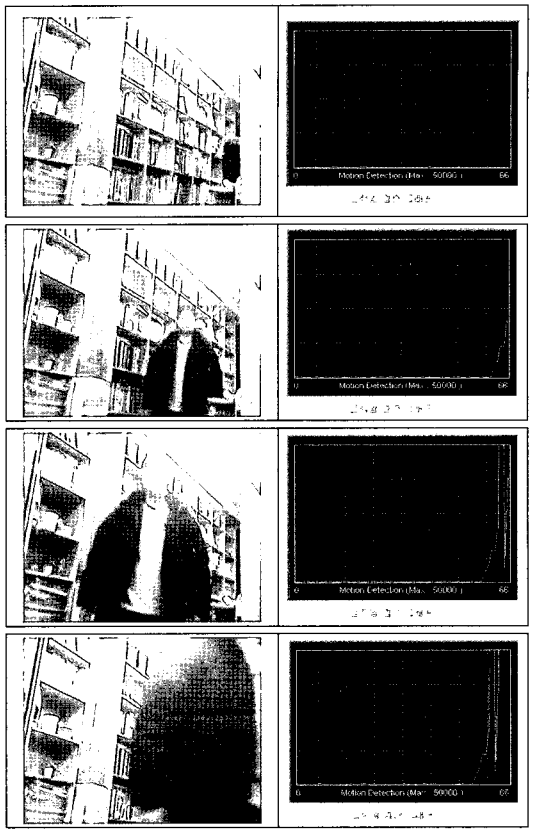


그림 11. 객체의 유입과 움직임 그래프
Fig. 11. Motion graph and incoming of object

제안한 방법을 위한 임계값들과 변수에 대한 최적화 실험 결과는 표 1과 같다. 적응적 배경영상 생성 임계값은 실험값이 5인 0.210 일 때, 배경영상 이미지내의 잡음을 제거 할 수 있었다. 그리고 픽셀 간격 임계값은 조밀할수록 움직임 추출시 객체 인식이 잘 이루어 졌지만, 많은 연산량을 요구하기 때문에 보다 빠른 탐색을 위해 실험값 "5"를 선택하여 실험하였다.

표 1. 임계값과 변수에 대한 최적화
Table 1. Threshold and variable for optimization

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
…	…	…

탐색블록 내에서 객체의 움직임에 따라 이동하는 객체블록과의 거리를 유지하기 위하여 설정하였던 최소블록과 탐색블록의 거리 D는 표 2와 같이 픽셀 간격의 거리는 검출속도 및 오류검출이 가장 적절한 "15픽셀"을 사용하였다.

표 2. 최소블록과 탐색블록의 거리 (D)
Table 2. Minimum block and search block of distance(D)

실험값(pixel)	검출속도(ms)	오류검출
5	210	0.358
...	...	...
10	251	0.310
...	...	...
15	273	0.215
...	...	...
20	350	0.198
...	...	...
25	399	0.181
...	...	...

실험에 사용하는 움직임 검출 방법은 차영상 기법, 블록정합 기법, 배경영상 교체기법과 비교하였으며, 정지영상 분석은 일정한 시간과 장소를 설정하고 움직임 검출 시 저장되는 BMP 이미지 파일을 분석한다. 저장된 BMP파일 중 배경만을 포함하는 이미지는 오류검출 하였다고 판단하고, 배경이외에 다른 객체가 포함되어 있으면 움직임 검출로 판단하였다.

표 3는 기존방법과 제안방법의 움직임 검출비교표로서 실험시간을 오전 10시부터 오후 9시까지 정해서 오전 10시부터 2시, 오후 6시부터 9시까지는 객체의 유입이 많지 않은 시간대이고, 3시부터 5시까지는 객체의 유입이 활발한 시간대로 움직임을 검출한 빈도수이다.

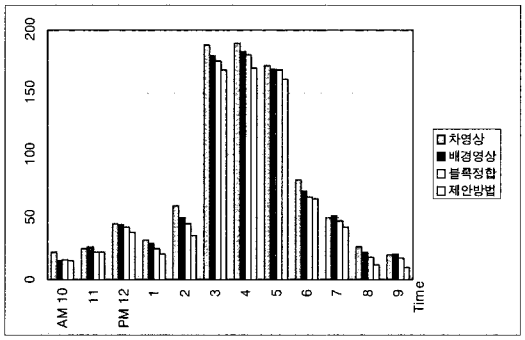
실험한 움직임 검출 방법인 차영상 기법, 블록정합 기법, 배경영상 교체기법과 제안하는 방법의 특징은 다음과 같고 움직임 검출 빈도수와 검출오류 빈도수는 그림 12와 같다.

- 차영상 기법 : 객체의 유입이나 움직임이 거의 없는 오전 10시부터 오후 2시, 오후 7시부터 9시까지의 시간대에 미세한 환경의 변화에도 민감하여 움직임이 있는 것으로 판단하고 움직임을 검출하고 있다.
- 배경영상 교체기법 : 잡음이나 기타 환경적 변화에 차영상 기법보다 강인성을 보이나 객체의 유입이 활발한 오후 3시부터 5시까지 검출한 결과를 볼 경우 객체의 유입이외에도 불필요한 정지영상을 포함하고 있어 오류빈도수가 증가하고 있다.
- 블록정합기법 : 차영상 기법이나 배경영상 교체기법에 비하여 오류빈도수가 많지는 않지만 오후 3시부터 5시까지의 검출 결과를 보면 오류빈도수가 급격히 증가 하고 있음을 알 수 있다.
- 제안한 방법 : 기존 방법은 조명의 미세한 변화나 영상내의 잡음에 의한 오류가 오전 10시부터 오후 2시까지 객체의 유입이 많지 않은 시간대에도 오류빈도수가 나타나고 있으나, 제안방법은 거의 오류가 없음을 알 수 있다. 그리고 오후 3시부터

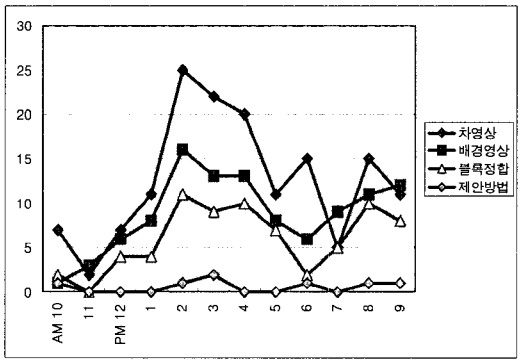
표 3. 움직임 검출비교
Table 3. Motion detection comparison

시간	차영상 기법		배경영상 교체기법		블록정합 기법		제안한 방법	
	빈도수	오류	빈도수	오류	빈도수	오류	빈도수	오류
10	22	7	15	1	16	2	15	1
11	25	2	26	3	22	0	22	0
12	45	7	44	6	42	4	38	0
1	32	11	29	8	25	4	21	0
2	59	25	50	16	45	11	35	1
3	188	22	179	13	175	9	168	2
4	190	20	183	13	180	10	170	0
5	172	11	169	8	168	7	161	0
6	80	15	71	6	66	2	65	1
7	50	5	51	9	47	5	42	0
8	26	15	22	11	18	10	12	1
9	20	11	21	12	17	8	10	1

5시까지 객체의 유입이 활발했던 시간대에도 오류빈도수가 적기 때문에 정확한 객체의 움직임을 검출하였다.



(a) 움직임 검출
(a) Motion Detection



(b) 움직임 검출 오류
(b) Motion detection of error

그림 12. 제안방법과 기존방법의 움직임 검출 비교표
Fig. 12. Motion detection comparison diagram

V. 결 론

블록정합기법은 블록내의 객체의 움직임 측정이나, 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 방법으로 움직이는 물체에 대한 정확한 위치 정보와 형태 정보를 얻을 수 있다. 그러나 배경이미지 자체는 시간에 따라 계속적으로 변하기 때문에 정확한 움직임을 검출하는데 어려움이 있다.

실험 결과 및 평가절에서 제안한 움직임 검출 방법과 기존의 차영상 기법, 배경영상 교체기법, 블록정합기법을 비교한 결과 조명의 변화나 영상자체의 잡음에 의하여 객체로 판단하고 움직임을 검출하는 경우가 많았으나, 제안한 방법은 이러한 단점을 보완하였고, 기존 방법이 전체 이미지를 스캔하거나 정해진 영역만을 스캔하여 객체의 움직임을 검출하는 반면 제안한 방법은 가변적인 탐색영역을 생성하여 움직임을 검출하기 때문에 빠르게 검출할 수 있는 장점이 있다.

따라서 제안된 방법은 기존방법보다 오류빈도가 적은 정확률과 기존방법보다 빠른 속도 보상을 받을 수 있으며, 실시간 움직임 검출 시스템으로 적합함을 알 수 있다. 그리고 단일 객체의 움직임을 중심으로 검출하기 때문에 동시에 다른 방향에서 유입되는 객체에 대해서는 성능 실험에서는 배제하였다.

본 논문에서 설계하고 구현한 시스템은 실험을 통하여 다른 시스템보다 효과적으로 움직임 검출을 할 수 있었으며, 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] 이완범, "저 전송률 무선 시스템에 적합한 새로운 움직임 추정 알고리즘 및 VLSI 설계에 관한 연구," 원광대학교 박사학위 논문, 2003.

[3] 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.

[4] 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.6 pp.706-717, JUNE 2003.

[5] 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.

[6] N. Ikonomakis, K.NPlataniotis, M.Zervakis, A.N. Venetsanopulos, "Region Growing and

Region Merging Image Segmentation", IEEE DSP 97. pp.299-302, 1997.

[7] 김기주, 방경구, 문정미, 김재호, "효율적인 화상회의 동영상 압축을 위한 블록기반 얼굴 검출방식", 한국통신학회논문지, Vol.29, No.9C pp.1258-1268, 2004.

[8] 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.

[9] S.Y.Wan, W.E.Higgins, "Symmetric region growing", International Conference on Image Processing 2000, Vol.12 No.9 pp.1007-1015, SEPTEMBER 2003.

이 창 수(Chang-soo Lee) 정회원

1999년 2월 한서대학교 컴퓨터과학과 졸업

2002년 2월 숭실대학교 컴퓨터공학과 석사

2005년 7월 숭실대학교 컴퓨터공학과 박사

<관심분야> 영상처리, 멀티미디어 보안, 네트워크 보안

박 미 옥(Mi-og Park) 정회원

1991년 2월 조선대학교 전산통계학과 졸업

1993년 2월 숭실대학교 컴퓨터학과(석사)

2001년 2월 매직캐슬 선임연구원

2004년 1월 숭실대학교 컴퓨터학과 졸업(박사)

2005년 3월 성결대학교 컴퓨터공학부 전임강사

<관심분야> 이동통신 보안, 정보보호, 암호알고리즘

이 경 석(Kyung-seok Lee) 정회원

1978년 숭실대학교, 학사

1981년 성균관대학교, 석사

1983년~1986년 Univ. Paris 7 연구소(ITODYS) 연구원

1986년 University Paris 7, 박사

1987년~현재 산업연구원 연구위원

2001년~현재 건국대 정보통신대학원 겸임교수

<관심분야> 데이터베이스, 네트워크보안, 정보보안표준, 정보보안알고리즘