가변 탐색블록을 이용한 객체 추적에 관한 연구

민 병 묵[†] · 오 해 석[††]

요 약

카메라를 통하여 입력되는 객체의 움직임은 잡음이나 조명의 변화에 따라 정확하게 추출하고 추적하는 것이 어렵다. 실시간으로 입력되는 영상에서 객체를 추출하고 움직임을 추적하기 위해서는 고속탐색 알고리즘이 필요하다. 본 논문에서는 정확한 객체의 움직임을 추출하고 고속 추적을 위하여 배경화면의 변화에 강인한 배경영상 갱신 방법과 가변적인 탐색영역을 이용한 정확하고 빠른 알고리즘을 제안한다. 배경영상 갱신 방법은 임계값이 실험적 기준치 보다 작은 경우에는 배경영상을 갱신하고, 큰 경우에는 객체가 유입된 시점으로 판단하여 픽셀검사를 통해 객체의 윤곽점을 추출한다. 추출된 윤곽점은 객체 최소블록의 생성과 일정한 거리를 유지하는 탐색블록을 생성하여 정확하고 빠른 객체의 움직임을 추적한다. 실험결과, 제안한 방법은 95% 이상의 높은 정확도를 보였다.

키워드 : 객체추출, 객체추적, 영상보안

A Study on Object Tracking using Variable Search Block Algorithm

Min Byoung Muk[†] · Oh Hae Seok[††]

ABSTRACT

It is difficult to track and extract the movement of an object through a camera exactly because of noises and changes of the light. The fast searching algorithm is necessary to extract the object and to track the movement for realtime image. In this paper, we propose the correct and fast algorithm using the variable searching area and the background image change method to robustic for the change of background image. In case the threshold value is smaller than reference value on an experimental basis, change the background image. When it is bigger, we decide it is the point of the time of the object input and then extract boundary point of it through the pixel check. The extracted boundary points detect precise movement of the object by creating area block of it and searching block that maintaining distance. The designed and embodied system shows more than 95% accuracy in the experimental results.

Key Words : Object Extraction, Object Trace, Image Processing

1. 서 론

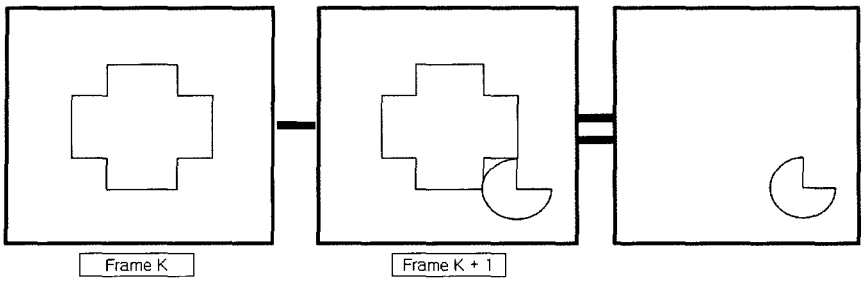
인터넷 시대에 접어들면서 웹 카메라를 이용한 보안 시스템의 개발이 활발하다. 원격지의 카메라로부터 전송된 영상을 통하여 현재의 상황을 파악할 수 있으며, 적절한 조치를 웹을 통해 취할 수 있다. 이러한 웹 멀티미디어 보안 시스템은 교통현황 파악, 건설현장이나 상가매장의 모니터링, 무인 시설물감시 등에 사용되고 있다. 그러나 활용영역이 확대되면서 영상의 해상도와 전송속도, 그리고 보안시스템의 핵심인 객체영역 인식, 영상 정보의 처리, 저장, 검색 기술, 추적 등의 연구가 추가적으로 요구 되고 있다[1].

실시간 영상에서 객체를 인식하고 추적하기가 매우 어려운 일임에도 불구하고 관련 연구가 활발히 진행되고 있다. 객체 추적은 카메라로부터 입력된 영상에서 움직임을 보이는 객체를 인식하고, 그 움직임을 추정하여 추적하는 것이다. 객체를 추적하는 방법은 보안, 의료, 군사, 교통, 제어분야 등 여러 분야에 응용될 수 있어 많은 연구와 개발이 이루어지고 있다.

하지만, 기존의 실시간 객체 추적 시스템의 구현에는 많은 어려움이 존재한다. 우선 강인한 실시간 객체 추적 시스템을 구현하기 위하여 고가의 장비를 요구한다. 움직임을 감지하기 위한 센서를 장착한 카메라나 pan/tilt로 움직이는 카메라 등의 요구이다. 또한 객체 추출이나 추적에 많은 연산을 필요로 하는 알고리즘 등을 적용하여 객체 추출에는 뛰어난 성능을 보이지만 실시간 객체 추적에는 부적합하다. 또한 픽셀 변화만을 사용하기 때문에 객체의 움직임이 없어도 조명의 변화에 따라 쉽게 객체로 오인하기 쉽다.

동영상 압축 방식에서 비디오 신호의 시간상 중복성을 제거하고 동영상을 압축하기 위해 움직임 보상방식이 요구되고

† 종신회원 : 숭실대학교 멀티미디어연구실 박사과정
†† 종신회원 : 경원대학교 소프트웨어대학 교수
논문접수 : 2006년 4월 18일, 심사완료 : 2006년 7월 5일



(그림 1) 차영상기법

있다. 움직임 보상방식(Motion Compensation)은 비디오 영상에서 배경과 객체를 구분하여 전송하기 위한 방법으로 움직임 벡터(Motion Vector)를 이용하여 객체의 움직임이나 추적을 예측한다. 움직임 벡터는 실시간으로 처리해야 하기 때문에 계산량을 현저히 줄일 수 있는 알고리즘을 필요로 한다[2].

실시간 영상에서 배경영상과 입력영상을 구분하여 움직인 객체를 추출하거나 추적하기 위해 차영상을 이용한 방법, 블록정합기법, 배경영상을 이용한 방법 등을 이용한 연구가 이루어지고 있다[3].

차영상을 이용하는 방법은 인접한 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 과정으로 인해 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 객체의 유입이 없는 시점에서도 객체로 오인할 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 간격을 두고 이미지의 픽셀을 검색하여 효과적으로 객체의 윤곽점을 검출한다. 객체의 윤곽점 좌표는 최대 가로방향, 세로방향을 측정하여 객체의 최소블록을 생성하고 일정한 거리를 유지하는 가변적인 탐색블록을 생성하여 정확하고 빠르게 객체를 추출 하고 추적 할 수 있도록 하였다.

배경영상 생성과 그물형 픽셀간격의 임계값은 실험적 데이터를 기준으로 가장 적절한 임계값을 선택하였고 객체의 최소블록과 탐색블록사이의 임계값은 값을 변화시켜 최적화된 임계값을 선택하였다.

본 논문의구성은 2장에서 기존의 움직임 검출방법을 분석하고, 3장에서는 배경영상 갱신과 객체의 윤곽점 추출방법 및 객체의 최소블록 및 탐색블록 이용한 객체의 움직임 추적방법을 제안한다. 4장에서는 제안한 방법으로 실험한 결과를 기술하고, 마지막으로 5장에서는 결론을 기술한다.

2. 객체 추출 및 추적 방법

객체의 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.

2.1 차영상 기법

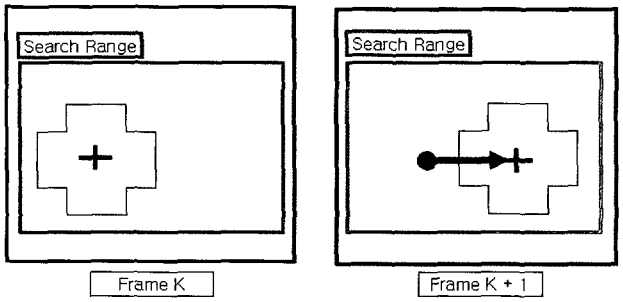
객체 추출을 위한 후보영역을 검출하기 위해 연속된 두 프레임간의 차영상 분석 기법을 사용한다. (그림 1)의 차영상 분석 기법은 연속된 두 프레임간의 밝기 차이를 구한 후, 임계값을 사용하여 임계값 보다 낮은 밝기 차이를 가진 부분은 객체가 없는 배경으로 구별하고 임계값 보다 큰 밝기 차이를 가진 부분은 객체의 유입이 있다고 판단한다[4].

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

2.2 블록정합 기법

블록정합 기법은 현재 프레임 탐색영역 안에서 이전 프레임의 지정된 블록과 가장 유사한 블록을 찾는 방법으로 (그림 2)와 같이 객체가 움직이지 않다가 다시 움직이는 경우에도 추적이 가능 하고 블록의 크기와 추적할 객체를 지정할 수 있다[5, 6].

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



(그림 2) 블록정합기법

동작벡터를 적용함으로써 정확한 움직임 검출을 할 수 있다. 그러나 부정합오류가 상층 레이어로부터 전파되어 정합오류가 증가 될 수 있고, 각 레이어로 진행될수록 시간복잡도는 계속 증가하게 된다.

2.3 배경영상 교체기법

배경영상 교체기법은 현재 프레임과 기준이 되는 배경 영상의 차이를 구하는 방법이다. 차영상기법과 같이 인접한 두 프레임을 비교하는 것이 아니라 (그림 3)과 같이 이전 프레임들로부터 배경이 되는 영상을 추출하고 이 영상과 현재 프레임을 비교하여 모션을 추출하게 된다[6,7,8]. 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 n개의 프레임들에 대한 화소값이 필요하며, 이들 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 n개 프레임들에 대한 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 하는 문제점을 갖고 있다.

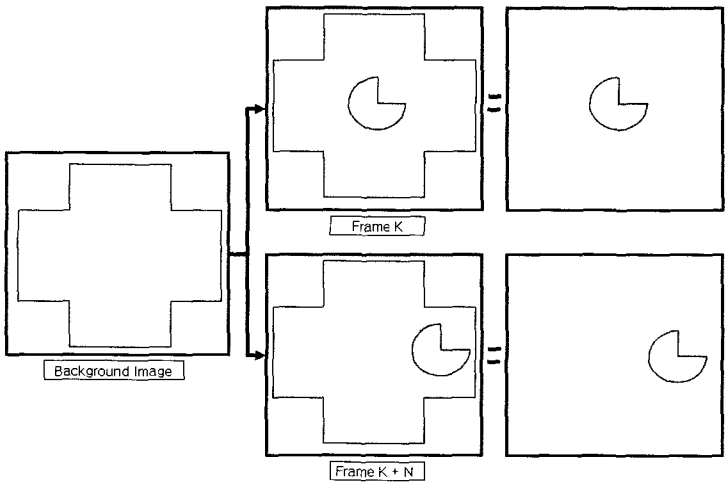
3. 실시간 객체 추출 및 추적

3.1 제안 방법

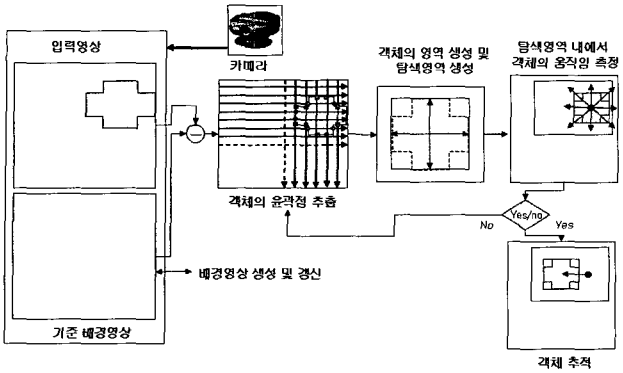
제안하는 객체 추출 및 추적 방법은 카메라로부터 획득되는 초기 영상 이미지를 배경영상 이미지로 선택하며 이 이미지는 객체의 유입이 없는 상태에서 획득된 순수한 배경이미지이다. 배경영상과 카메라로부터 입력되는 영상과의 차를 이용하여, 임계값이 존재하면 이미지 검사를 통하여 움직인 객체의 윤곽선에 해당하는 부분을 찾고 움직임 검출여부를 판단하여 추적하게 된다.

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 되므로 정확한 객체 추적을 위해서 배경영상을 갱신한다. 빠른 객체 추적을 위해서 객체의 영역을 추출하고 움직임을 측정하기 위한 탐색영역을 생성하여 매번 전체 이미지의 픽셀을 스캔하지 않고도 정확한 추적을 할 수 있도록 하였다.

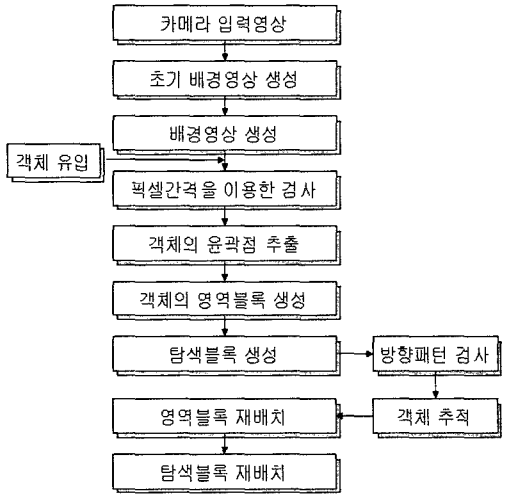
(그림 4)는 제안하는 객체추출 및 추적시스템을 나타내고 있으며, (그림 5)에서는 본 논문에서 제안하는 움직임 검출 과정을 나타내고 있다.



(그림 3) 배경영상 교체기법



(그림 4) 제안하는 객체 추출 및 추적 시스템



(그림 5) 객체 추출 및 추적 과정

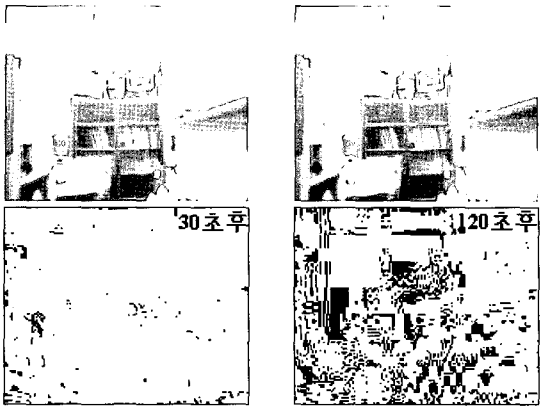
3.2 배경영상 획득

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 않은 상황이더라도 배경영상 자체가 변하게 된다.

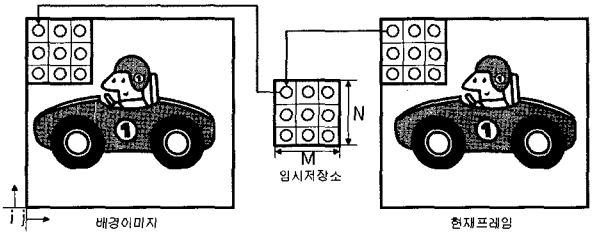
변화가 없는 공간에서 조명의 변화 등과 같은 미세한 움직임에도 차 영상을 통하여 초기의 배경영상과 현재의 배경영상을 비교해보면 이미지 내에 잡음이 발생함을 할 수 있다. (그림 6)에서와 같이 이러한 잡음은 시간이 지남에 따라 더 많이 발생한다. 따라서 객체의 움직임이 없는 배경영상 임에도 불구하고 객체의 움직임을 검출하는 오류를 발생시킨다.

본 논문에서는 이러한 잡음을 제거하여, 보다 정확한 움직임 검출을 위해서 배경영상을 갱신하는 방법을 사용한다. 전체적인 배경영상의 갱신은 많은 연산량을 필요로 하고 객체의 움직임을 검출하는데 시간이 많이 소요되므로 N × M 마스크를 사용하여 계속적으로 배경영상을 갱신하면서 객체의 유입을 검사한다.

(그림 7)은 입력영상 "Frame t"의 N×M 마스크 내에서 RGB 채널의 값과 배경영상 "Background Image"와의 차이값이 임계값α 보다 작으면 배경영상이 변경된 것으로 간주하여 해당 픽셀을 입력영상의 픽셀로 변경한다.



(a) 30초 후 배경영상 (b) 120초 후 배경영상
(그림 6) 시간에 따른 배경영상의 변화



(그림 7) 배경이미지 갱신

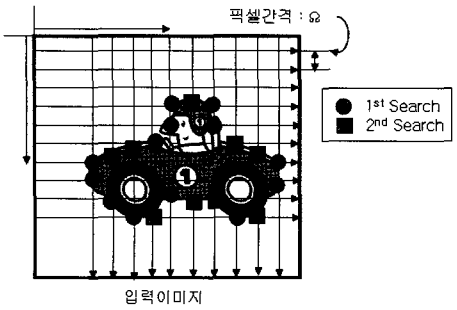
식(1)에서는 RGB 채널 각각의 차이값이 α보다 작으면, 객체가 유입된 상황이 아니라고 판단하고 입력영상에서 픽셀을 선택하여 배경영상을 갱신하여 준다. α보다 클 경우는 새로운 객체가 영상 안에 유입된 경우이므로 배경영상의 픽셀을 그대로 사용한다.

$$
\begin{aligned}
R_{channel} &= \sum_{i=0}^{M}\sum_{j=0}^{N} abs(R_{Value}(Background_{Imgmask}[i,j] - \\
&\quad R_{Value}(Input_{Imgmask}[i,j]))) \\
G_{channel} &= \sum_{i=0}^{M}\sum_{j=0}^{N} abs(G_{Value}(Background_{Imgmask}[i,j] - \\
&\quad G_{Value}(Input_{Imgmask}[i,j]))) \\
B_{channel} &= \sum_{i=0}^{M}\sum_{j=0}^{N} abs(B_{Value}(Background_{Imgmask}[i,j] - \\
&\quad B_{Value}(Input_{Imgmask}[i,j]))) \\
T_{RGB} &= \frac{R_{channel} + G_{channel} + B_{channel}}{3} \\
Object_{detect} &= \begin{cases} 1, & \text{if}(T_{RGB} > \alpha) \\ 0, & otherwise \end{cases}
\end{aligned} \tag{1}
$$

3.3 객체의 윤곽점 추출

카메라를 통하여 입력되는 영상이미지는 배경영상의 RGB 채널과 비교하여 수시로 배경영상을 갱신하거나 객체의 유입을 검사한다. 즉, 배경영상 갱신 중 임계값 α보다 큰 경우가 객체가 유입된 시점이다.

객체가 유입된 영상이미지는 픽셀검사를 통하여 객체의 위치를 검출하고 윤곽점을 추출하게 된다. 영상이미지 전체를



(그림 8) 객체의 윤곽점 추출

픽셀단위로 검색을 한다면 실시간으로 추출되는 이미지를 모두 검사하게 되므로 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.

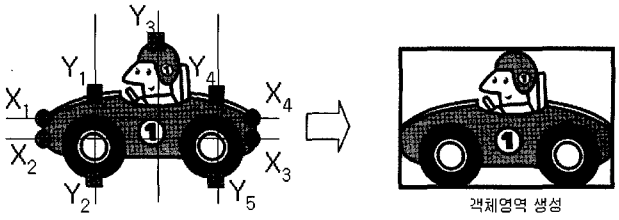
(그림 8)에서 객체의 위치 검출은 실험을 통하여 최적값으로 추출된 일정한 간격값인 Ω만큼의 픽셀간격을 두고 검사를 진행하는 것을 나타내었다. 1차 검사를 통해 객체의 가로 위치값을 최소, 최대 형태로 저장하고, 2차 검사를 통해 객체의 세로 위치값을 최소, 최대 형태로 저장한다. 이때 객체의 윤곽선을 결정하는 과정은 배경영상의 차이값으로 확인하게 된다.

픽셀 검사에서 배경영상과 입력영상의 RGB 픽셀간 차이가 임계값 β를 넘거나 같은 값을 가지면, 식(2)에 따라 객체가 있는 것으로 간주한다.

$$
\begin{aligned}
R_{object} &= \sum_{i=0}^{Height}\sum_{j=0}^{Width} abs[R_{Value}(Background_{Img}[i,j]_{\Omega} \\
&\quad - R_{Value}(Input_{Img}[i,j]_{\Omega}))] \\
G_{object} &= \sum_{i=0}^{Height}\sum_{j=0}^{Width} abs[G_{Value}(Background_{Img}[i,j]_{\Omega} \\
&\quad - G_{Value}(Input_{Img}[i,j]_{\Omega}))] \\
B_{object} &= \sum_{i=0}^{Height}\sum_{j=0}^{Width} abs[B_{Value}(Background_{Img}[i,j]_{\Omega} \\
&\quad - B_{Value}(Input_{Img}[i,j]_{\Omega}))] \\
T_{RGB} &= \frac{R_{object}+G_{object}+B_{object}}{3} \\
Object(i,j) &= \begin{pmatrix} 1, \text{ if}(T_{RGB} > \beta) \\ 0, otherwise \end{pmatrix} \quad . \qquad (2)
\end{aligned}
$$

3.4 객체의 최소블록 생성

픽셀간격을 통하여 추출한 객체의 위치 좌표를 이용하여 객체의 최소블록을 설정한다. (그림 9)는 객체의 영역을 설정하는 방법이다. 1차 검사 후 추출된 객체의 윤곽점 $x_1, x_2, \ldots, x_n$ 각각의 좌표값을 저장하고, 동일한 스캔라인에 위치한 좌표점과의 거리를 측정하여 객체의 최소영역에 대한 최대 가로길이 값을 추출한다. 그리고 2차 검사 후 추출된 윤곽점인 $y_1, y_2, \ldots, y_n$의 좌표점을 계산하여 최대 세로길이 값을 추출한 다음 객체의 최소블록을 생성한다.



(그림 9) 객체의 최소블록 생성

식(3)은 객체의 최소블록인 사각형 블록을 설정한다.

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

사각형 블록이 설정되면 중심좌표를 계산하며, 중심좌표는 객체가 다음 프레임의 탐색블록에서 움직일 수 있는 방향성을 예측하고 최소블록의 위치를 갱신하기 위하여 사용되어진다. 이러한 방향성 예측은 객체 추적을 빠르게 검출하고 다음 프레임인 입력영상 이미지의 객체유입 시점을 다시 검사하지 않기 위함이다. 그러므로 객체가 존재하는 시간동안에는 매 프레임마다 그물형 검사를 사용하여 윤곽점을 추출할 필요 없이 탐색블록 내에서 움직임을 검출할 수 있다.

객체의 영역인 사각형 블록의 중심좌표는 식(4)로 나타내는데, 여기서 $Object_{Width}$는 가로방향의 최대 길이 값이고, $Object_{Height}$는 세로방향의 최대길이 값이다.

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

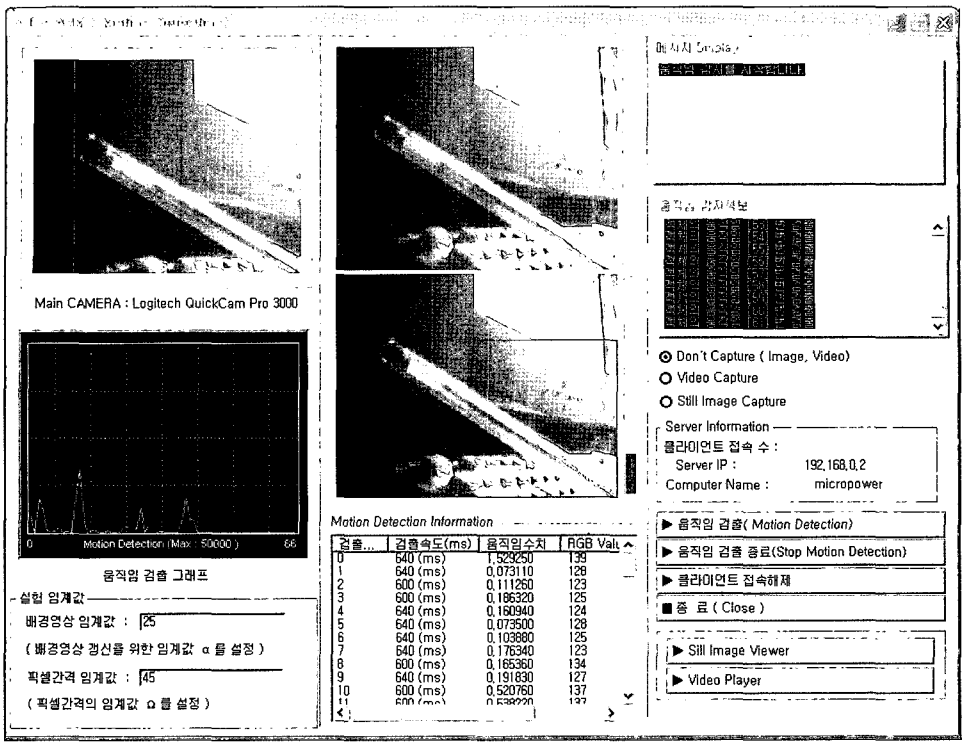
3.5 탐색블록 생성 및 객체 추적

객체의 윤곽점 추출을 통하여 생성된 최소블록은 카메라 입력영상 이미지와 현재 설정된 최소블록 내의 객체 움직임을 빠르게 추적하기 위해서 탐색블록을 설정한다.

제안하는 시스템의 탐색영역은 객체의 최소블록을 기반으로 생성한다. 만약 최소블록의 위치가 이미지의 하단부분이면 하단부분부터 검색을 할 수 있도록 설정하고, 좌측상단부분이면 좌측상단부분부터 추적할 수 있도록 한다. 제안하는 시스템의 움직임 탐색영역인 탐색블록을 설정하는 과정이 (그림 10)에 나타나 있다.

카메라를 통하여 입력되는 입력 영상이미지와 탐색블록의 탐색영역과 매칭하는 과정, 방향 패턴을 사용하여 움직임을 검사하는 과정은 (그림 10)과 같다. (그림 10)에서 탐색영역인 탐색블록의 초기 좌표 (i,j)는 카메라 입력영상의 (i,j)의 위치로 탐색영역의 초기 좌표가 된다. 설정된 최소블록과 입력영상의 탐색영역 내에서 객체의 이동성을 검사하고 이 검사는 객체블록의 네 변의 중심점을 중심으로 (그림 10)의 방향패턴을 사용하여 우선순위를 정해 검사한다.

설정된 입력영상의 탐색영역 내의 초기좌표인 (i,j)부터 픽셀의 RGB 값과 이전 프레임의 탐색영역 RGB 차이 값을 비교하여 최소블록이 이동된 경우이면, 최소블록과 탐색블



(그림 11) 제안 시스템의 메인화면

록을 이동된 위치만큼 갱신한다.

탐색블록인 $RGB_{Object}(i,j)$와 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 최소블록과의 거리가 D만큼을 유지하지 않기 때문에, 탐색블록과 객체블록과의 거리가 D가 될 수 있도록 재배치하는 것은 식(6)에 의해 이루어진다.

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

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

4. 실험 결과 및 분석

4.1 실험 환경

본 실험에서는 사람이라는 움직임 객체를 추적하기 위해 실내 환경인 실험실내부에서 실험하였으며 명암도의 변화가 너무 크지 않은 오후시간대에 PC카메라를 사용하여 배경영상과 입력영상을 실시간으로 처리하였다.

시스템은 Intel Pentium 4 CPU 2.00GHz, 512M RAM의 PC에서 Visual C++ 6.0(Service Pack 6)을 이용하여 구현하였으며, 배경영상과 입력영상은 전송을 고려하여 320×240의 RGB 24bit 컬러 영상을 이용하였다.

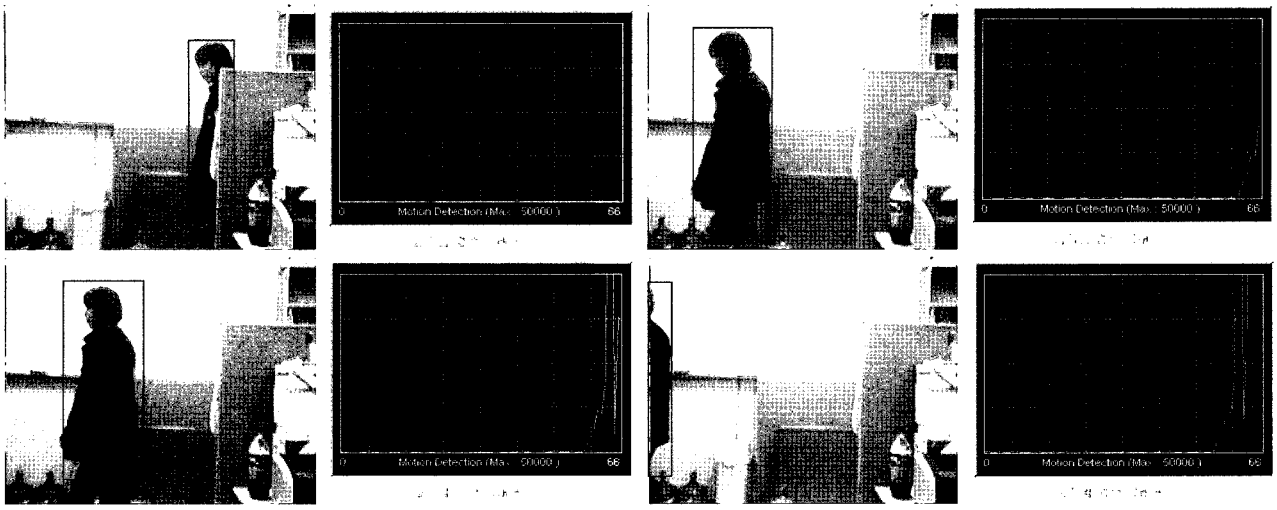
4.2 실험 결과

제안하는 객체 추적 방법을 사용하여 구현한 시스템은 (그림 11)과 같으며, 초기 카메라에서 입력되는 영상은 초당 15프레임으로 설정하여 사용하였다. (그림 11)의 움직임 그래프는 시간 변화에 따라 객체의 움직임을 검출하였을 경우, 아날로그 형태의 그래프를 생성하고 각 실험임계값과 객체 추적에 대한 이미지 정보는 리스트컨트롤을 사용하여 중하단부에 표시하도록 하였다.

기존의 블록정합 방법은 객체가 블록 안에 있다는 가정 하에 객체 추적을 한다. 그러므로 블록 밖으로 객체가 유입되면 초기 블록과의 유사성을 찾지 못하기 때문에 객체를 인식하지 못하는 오류가 발생하고, 차영상만을 이용할 때는 잡음에 민감한 반응을 보이기 때문에 움직인 객체가 입력영상에 없다고 하더라도 움직임으로 검출하고 추적을 하게 된다. 그러나 제안한 방법을 이용하여 객체를 추출하고 추적한 결과, 객체의 유입이 있고 이동성을 가지고 있을 경우만 추적을 하게 되었다. 그리고 윤곽점을 추출하여 객체의 최소블록과 탐색블록을 생성하여 객체의 방향성과 이동성을 예측할 수 있기 때문에 객체의 정확한 추출과 빠른 추적이 가능하였다.

카메라를 통하여 보여지는 영상과 객체의 유입에 따른 추적은 (그림 12)와 같으며, 조명의 변화나 잡음을 알 수 있도록 그래프에서 모두 표현하도록 하였고, 객체 추적은 적색의 사각형을 생성하여 표현하였다.

<표 1>은 제안한 방법에 의한 임계값들과 변수에 대한 최적화 실험 결과이다. 배경영상 생성 임계값은 실험값이 5인



(그림 12) 객체 영역 추출 및 추적

〈표 1〉 임계값과 변수에 대한 최적화

임계값 \ 실험값	1	2	…	5	…	25	…	30	…
α	0.023	0.151	…	0.210	…	0.915	…	0.887	…
β	0.101	0.256	…	0.494	…	0.856	…	0.934	…
Ω	0.951	0.903	…	0.851	…	0.595	…	0.496	…

〈표 2〉 탐색블록의 위치 거리(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
…	…	…

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.

탐색블록과 객체의 최소블록을 설정하기 위한 임계값 D의 실험적 최적값은 <표 2>에서와 같이 "15 pixel"의 중간치 값을 사용하여 실험 하였다.

실험 비교 방법은 차영상 기법, 블록정합 기법, 배경영상 교체기법과 비교하였으며, 비교대상은 정지영상 분석을 통하여 평가하였다. 정지영상 분석은 일정한 시간과 장소를 설정하고 객체가 추출되어 추적하는 과정에서 객체의 유입이 없는 시점까지 저장되는 BMP 이미지 파일을 분석한다. 저장된 BMP파일 중 배경만을 포함하는 이미지는 오류로 검출 되었다고 판단하고, 배경이외에 다른 객체가 포함되어 있으면 정확히 추출한 것으로 판단하였다.

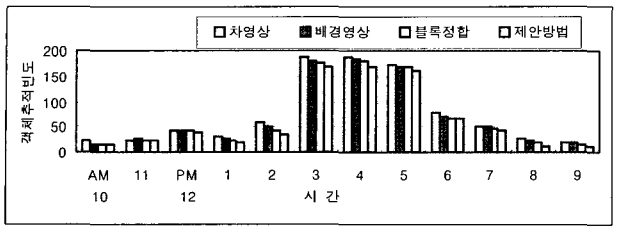
〈표 3〉 기존방법과 제안한방법의 객체추출 오류횟수 비교

방법 \ 시간	차영상 기법		배경영상교체기법		블록정합 기법		제안한 방법	
	빈도수	오류	빈도수	오류	빈도수	오류	빈도수	오류
AM 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
PM 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

실험시간은 객체의 유입이 많지 않은 오전 10시부터 오후2시, 오후6시부터 오후9시까지와 객체의 유입이 활발한 오후 3시부터 오후5시로 하여 실험한 결과 <표 3>과 같은 결과를 얻었고 이때의 객체 추적 빈도수 측정 그래프는(그림 13)과 같다.

실험한 움직임 검출 방법인 차영상 기법은 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 교체기법은 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.



(그림 13) 객체 추적 빈도수 측정결과

블록정합기법은 차영상 기법이나 배경영상 교체기법에 비하여 오류빈도수가 많지는 않지만 오후 3시부터 5시까지의 검출결과를 보면 오류빈도수가 급격히 증가하고 있음을 알 수 있다.

기존 방법은 조명의 미세한 변화나 영상내의 잡음에 의한 오류가 오전 10시부터 오후 2시까지 객체의 유입이 많지 않은 시간대에도 오류빈도수가 나타나고 있다. 그러나 제안방법은 거의 오류가 발생하지 않고, 오후 3시부터 5시까지 객체의 유입이 활발했던 시간대에도 오류빈도수가 적기 때문에 정확한 객체의 움직임을 검출하였다.

5. 결 론

본 논문에서는 초기의 배경영상을 기준으로 입력영상과의 차를 구하고 시간에 따라 변화하는 배경영상을 N×M 픽셀 마스크만큼 교체하여 배경영상을 갱신하였다. 실험결과, 기존의 배경영상기법에서 배경영상 자체의 변화 때문에 생기는 문제점을 해결하였고, 픽셀 검사는 모든 픽셀을 연산에 참여시키지 않고 일정한 간격을 두고 이미지의 픽셀을 검색하여 실시간으로 처리되는 실제적인 연산량을 줄여 객체의 윤곽점을 효율적으로 추출할 수 있도록 하였다. 추출된 윤곽점을 이용하여 객체 영역을 추출하고 탐색블록을 생성하여 객체 추적을 정확하고 빠르게 측정할 수 있도록 하였다.

제안한 방법은 기존방법보다 오류빈도가 적은 정확률과 빠른 속도 보상을 받을 수 있으며, 실시간 객체 추적 시스템으로 적합함을 알 수 있다. 그러나 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 실시간 객체 추적에 적합하고, 이 방법은 영상보안시스템에서 움직이는 객체를 추적하거나 저장 방법 개선 등에 응용 되어질 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] R. Feraud, O.J. Bernier, J.E. Viallet, and M. Collobert, "A Fast and Accurate Face Detection Based on Neural Network." IEEE Trans. Pattern Analysis and Machine Intelligence, pp.42-53, 2001.

[6] 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.

[7] 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.

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

민 병 묵

e-mail : ceo@nanoware.co.kr
1985년~1987년 서울산업대학교 전자계산학과(학사)
1987년~1989년 연세대학교 산업대학원 전자계산학과(석사)
2001년~현재 숭실대학교 대학원 컴퓨터공학과(박사과정수료)
2004년~현재 나노웨어(주) 대표이사
2004년~현재 Nanoware Do BRAZIL 대표이사
2006년~현재 Nanoware Of CANADA 대표이사
관심분야 : Multimedia, Security 분야, 이동통신, 영상처리

오 해 석

e-mail : oh@kyungwon.ac.kr
1975년 서울대학교 응용수학과(학사)
1981년 서울대학교 대학원 계산통계학과 (석사)
1989년 서울대학교 대학원 계산통계학과 (박사)
1990년~1991년 일본 동경대학 객원교수
2000년~2001년 미국 스탠퍼드대학교 객원교수
2003년 한국정보처리학회 회장(역임)
1982년~2003년 숭실대학교 컴퓨터학부 교수/부총장(역임)
2003년~현재 경원대학교 소프트웨어대학 교수/부총장
관심분야 : Multimedia, Database, 지식경영, 영상처리