

움직임 검출의 캠 제어에 의한 웹기반 이동 객체 추적

Web-based Moving Object Tracking by Controlling Pan-Tilt Camera using Motion Detection

박천주
한국기술교육대학교 정보기술공학부 기술연구원

Cheon-Joo Park
Tech. Researcher, School of Information Technology Eng.,
Korea University of Technology and Education

박희정
안동과학대학 정보처리과 교수

Hee-Jung Park
Professor, Dept. of Information Processing,
Andong Science College

이재협
한국기술교육대학교 정보기술공학부 교수

Jae-Hyub Lee
Professor, School of Information Technology Engineering,
Korea University of Technology and Education

진병민
충북대학교 전기전자및컴퓨터공학부 교수

Byung-Min Jun
Professor, School of Electrical and Computer Engineering,
Chungbuk National University

중심어 : 이동 객체 추적, 움직임 검출, 차분 영상, RCM

요 약

본 연구에서는 연속된 비디오 프레임에 대상으로 움직임을 검출하고, 이에 따른 움직임 벡터를 이용하여 카메라를 좌우 상하로 제어함으로써 이동 객체 중심의 비디오 프레임을 지속적으로 확보할 수 있는 웹 기반의 이동 객체 추적 방법을 제안한다. 비디오 스트림에서 두 프레임간 일정 간격의 격자점에서의 차분을 계산하고, 임계치 이상의 차분을 갖는 격자점들에 대한 수평 및 수직 투영 히스토그램을 이용하여 움직임의 방향과 거리를 결정한다. 이 방향과 거리 정보를 이용하여 RCM을 제어함으로써 이동 객체의 움직임 영역이 비디오 스트림의 중앙에 위치하도록 한다. 실험 결과, 움직임이 있는 부분이 비디오 프레임의 중앙에 위치하는 것을 확인할 수 있었으며, 웹상의 서버/클라이언트 방식으로 구현함으로써 인터넷을 이용하여 원격의 클라이언트가 비디오 스트림을 모니터링할 수 있었다.

Abstract

In this paper, we suggest a method to acquire the moving object centered video by panning and tilting a camera automatically according to motion vectors calculated by detecting the motion of a moving object on video stream. We create a difference image by estimating the intensity difference at the grid points of neighboring frames. And we detect the motion using both horizontal projection histogram and vertical projection histogram and decide the center of motion part. Then we calculate a new direction and degree of the motion by comparing coordinates at the center of current motion and the center of previous motion. By controlling the RCM using these motion vectors, we can get video stream positioned moving object on the center of video frame. Through the experiments, we could get a moving object centered video stream continuously and monitor remotely by implementing sever/client architecture based on the web.

I. 서론

최근 보안 감시 분야의 시장이 증가함에 따라 이동 객체 검

출 및 추적 기술은 보안 감시 분야에서 매우 중요한 기술로 대두되고 있다. 기존의 보안 감시 시스템에서는 감시 요원이 각 카메라로 들어오는 영상들을 육안으로 일일이 지켜봐야 했

었지만, 움직임 객체 및 추적 기술로 보안 감시의 자동화가 가능해지고 있다. 이러한 보안 시스템의 자동화는 크게 센서에 의한 방법과 획득된 영상을 이용한 방법으로 개선이 진행되어 왔다. 센서 기반 방법은 센서를 이용하여 움직임에 대한 신호를 센서에 의해 검출함으로써 비디오 스트림의 녹화를 개시하거나 경보음을 발생하는 방법이다. 이 방법은 움직임이 검출되는 동안에만 녹화를 행하는 것이 가능함으로써 녹화를 위한 비디오 테이프의 비용 절감 등의 잇점을 가진다. 그러나 부가적인 센서 장치를 요구하며 이들 센서 장치의 설치에 있어 제약을 받는다. 반면, 영상 기반 방법은 카메라로부터 획득된 영상에 대한 영상 처리에 의해 움직임을 검출하여 녹화를 개시하거나 경보음을 발생하는 방식이다. 영상 기반 방식은 센서와 같은 부가적인 장치가 불필요하지만, 획득된 영상에 대한 영상 처리를 위한 처리기와 많은 계산량을 요구하므로 실시간 감시에 어려움이 있을 수 있다. 그러나 오늘날과 같이 인터넷과 컴퓨터가 대중화되고 있는 상황에서 영상 기반 방법은 처리 기법의 적용에 따라 많은 유연성을 가지며, 원격 감시 및 원격 카메라 제어 등의 응용이 가능하다는 장점을 가진다[1]. 또한 인터넷의 대중적 보급과 이동 통신 시장의 급격한 성장에 따라 보안감시의 현장뿐만 아니라 현장에서 떨어진 원격에서의 감시활동의 요구가 증대되었다. 따라서 오늘날에는 인터넷 혹은 이동 통신 기술과 접목된 보안 감시 시스템의 효용성이 커지게 되었다.

이동 객체 추적 기술과 관련된 기존 연구들을 살펴보면 크게 차분 영상을 이용하는 방법과 광류(optical flow) 벡터를 이용하는 방법으로 나누어 볼 수 있다. 화소간 차분 영상에 의한 방법[2],[3],[4],[5],[6],[7]은 일반적으로 카메라가 고정되어 있어 배경 장면의 변화가 없는 상황에서 이동객체를 추적하는데 주로 사용될 수 있는 방법으로 배경영상과 획득된 프레임 영상간의 차분을 이용함으로써 객체의 위치를 파악하게 된다. 이러한 방법은 주위 환경의 변화에 많은 제약을 받으며, 카메라가 이동하는 경우 배경영상이 변경되므로 이동 객체의 움직임을 검출하기 위한 차분 영상의 획득에 부적절하다. 반면 광류 벡터를 이용한 방법[8],[9]은 일반적으로 카메라와 객체 중 어느 하나가 고정되어 있고 다른 하나가 이동하는 경우에 이용된다. 이 방법은 이전 프레임에서 구해진 객체가 시간 및 공간의 변경을 통해 새로운 프레임의 어느 위치로 이동되었을 때, 시공간 벡터를 이용하여 객체의 이동 방향과 크기를 파악하는 방법이다. 이 방법은 처리에 요구되는 계산량이 많기 때문에 처리 속도가 느려서 실시간 환경에서의 적용이 부적합하다는 단점을 가진다.

이러한 기존 연구의 대부분은 이미 획득된 비디오 스트림 혹은 고정된 카메라로부터 획득된 비디오 스트림 상에 존재하는 객체의 움직임을 추적하는데 주안점을 두고 있다. 객체가 이동하고 이를 추적하기 위해 카메라가 이동하는 경우는 다른 경우와는 달리 카메라가 이동되면서 배경도 바뀐다는 문제점을 가진다[1],[10]. 따라서 단순한 차분 영상만을 이용할 수는 없다.

본 연구에서는 이동 객체를 따라 카메라의 방향을 이동할 수 있는 환경에서 응용할 수 있도록 비디오 프레임에서의 이동 객체의 움직임을 검출하고 이동 객체를 따라 카메라의 방향을 이동함으로써, 이동 객체 중심의 비디오 프레임을 지속적으로 획득하는 방법에 관한 것이다. 이를 위해 변경된 배경을 가지는 두 프레임간 차분 위치의 조절에 의한 차분 영상의 획득과 수직 수평 투영 히스토그램을 이용하여 움직임 부분의 중심을 구한다. 그리고, 이 정보를 이용하여 카메라를 좌우 상하로 조절함으로써, 다음 프레임의 중심에 객체가 위치할 수 있도록 한다. 또한 클라이언트에 의한 수동 제어 모드와 제안한 방법에 의한 자동 제어모드의 기능을 포함하며 원격 보안감시가 가능하도록 웹기반의 서버/클라이언트 시스템으로 구현한다.

II. 움직임 추적

움직임 추적은 객체의 움직임에 따라 이전 프레임과 현재 프레임간의 차분 영상을 구하고, 이 차 영상에서 큰 차를 가지는 위치를 카메라 비디오 스트림의 중앙에 위치하도록 RCM(Robotic Camera Mount)을 제어하는 과정으로 구성된다. 결국, 움직임 추적은 크게 움직임 검출과 RCM 제어로 나누어 볼 수 있다.

1. TransitRCM

우선 실험에 사용된 Surveyer사의 TransitRCM[11] 모듈의 여러 특성들을 소개하고자 한다. 이는 Pan과 Tilt의 제어단위와 검출된 움직임 부분의 중점간 관계를 파악하는데 도움이 되기 때문이다. 일단 움직임 검출 과정을 통하여 결정된 새로운 위치 정보는 RCM 모듈을 제어할 수 있도록 제어 단위로 환산된 후, RCM 제어 명령으로 변환되어야 한다. 이 명령은 직렬 포트를 통하여 RCM 모듈로 전송되어 카메라의 위치를 변경하게 된다.

본 연구에서 이용하는 카메라 제어 모듈인 TransitRCM은 초당 240° 이상의 최대 제어 속도를 제공하며 고속의 위치

변경을 제공한다. 또한 그림 1과 같이 좌우 이동(Pan) 범위는 180°, 상하 이동(Tilt) 범위는 120에서 180°를 제공하는데, 이 범위를 128개 단계로 나누어 이동할 수 있다. 따라서 각 이동 제어 단위의 이동각은 $180/128 = 1.41^\circ$ 가 된다. 움직임 검출에 의해 결정된 위치는 이 제어 단위에 적합하도록 환산되어야 하며, 이는 움직이는 객체와 카메라 간의 거리, 각 주사선의 개수, 카메라의 줌(zoom) 정도에 의존하여 결정되어야 한다.

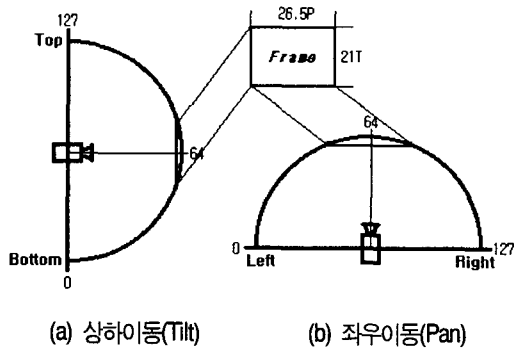


그림 1. RCM 제어 단위

카메라를 좌우와 상하로 적절히 회전시키기 위해서는 RCM이 이해할 수 있는 형식의 명령으로 지시를 해주어야 하는데, RCM 제어를 위해 사용되는 명령은 표 1과 같은 형식을 따라야 한다. 좌우 위치 이동(Pan)을 위해서는 'A' 문자 뒤에 0부터 127까지의 값을 적고, <CR> 문자가 덧붙는 형식의 명령을 사용한다. 상하 위치 이동(Tilt)을 위해서는 'B' 문자 뒤에 0부터 127까지의 값을 적고, <CR> 문자가 오는 형식의 명령을 사용한다. 이들 명령에서 숫자의 값은 위치 이동에 있어 절대 위치를 표현하는 것으로, 그 값이 0일 때, Pan의 절대 위치는 카메라를 움직일 수 있는 최좌측을 의미하며, Tilt의 절대 위치는 최하단의 위치를 가리킨다. 이 형식에 따른 명령을 직렬 포트를 통해 RCM 모듈로 전달하게 되면, 비로소 카메라의 방향을 변경할 수 있게 된다.

표 1. RCM 제어 명령 형식

구분	제어 명령 형식
좌우이동(Pan)	A[P] < CR >
상하이동(Tilt)	B[T] < CR >

실험에 사용된 카메라에서 획득되는 비디오 프레임은

352×288의 해상도를 가지는데, RCM 모듈의 제어 단위와 비디오 프레임과의 관계를 살펴보면 다음과 같다. 비디오 프레임에서 352화소의 가로축(F_w)은 RCM모듈의 Pan을 26.5회 이동한 분량에 해당하며, 288화소의 세로축(F_h)은 Tilt의 단위 변위를 21회 수행한 것과 동일하게 나타났다. 결국 식 (1)과 같이 P와 T의 이동 제어 단위를 화소 개수로 환산하면 13.28화소에 해당함을 알 수 있다.

$$1P = 1T = \frac{F_w}{26.5} \approx \frac{F_h}{21} \approx 13.28 \text{ pixels} \quad (1)$$

이는 수직 투영 히스토그램과 수평 투영 히스토그램을 통해 결정된 움직임 검출 부분의 중점이 다음 프레임의 정중앙에 위치하도록 RCM 모듈을 제어하는데 있어 서로간의 환산 관계를 파악하는데 중요한 정보를 제공한다.

2. 실시간 처리를 위한 고려 사항

각 프레임을 구성하는 모든 화소간의 차분을 계산하는 것은 실시간 제어를 하기에는 너무 많은 시간을 요구하므로, 본 연구에서는 다음과 같은 세가지 방법을 이용하여 계산량을 줄인다.

우선 움직임을 검출하기 위해 서로 다른 시간에 발생한 두 개의 비디오 프레임이 필요하다. 본 연구에서는 생성되는 비디오 프레임 중에서 식(2)와 같이 k 의 배수번째의 프레임을 취하고 나머지는 버리는 방법을 통하여 움직임 검출에 이용되는 프레임의 개수를 줄인다.

$$F = \{ Ft \mid t \bmod k = 0 \} \quad (2)$$

또한 카메라를 통해 생성되는 비디오 프레임은 색상 영상인데, 이러한 색상 영상에 대한 차분 계산량을 줄이기 위해 계조 영상으로 변환하여 처리에 이용한다.

끝으로 추출된 두 프레임에 대해 모든 화소들간의 연산보다는 프레임 전역에 분포되어 있는 일부 화소들에 대해서만 연산하는 것이 바람직하다. 이를 위해 각 프레임에 대해 일정한 간격(Δg)을 가지도록 수직방향과 수평방향으로 주사 라인을 정하고, 이들 격자 형태의 주사선 구조에서 교차하는 격자점에 대해서만 연산을 하도록 한다. 인접한 주사선들간의 간격은 움직임 위치의 정확도 및 연산 시간과 밀접한 관련이 있다. 즉, 주사선의 간격을 크게 하면 계산량의 감소되나 움직

입의 유무를 판단하는데 어려움이 있을 수 있다. 일반적으로 Δg 는 RCM 모듈의 제어 단위인 1P 혹은 1T와 동일한 화소수로 하는 것이 카메라의 이동을 제어하는데 바람직하다. 격자 구조의 해상도에 따른 격자간 간격을 화소로 표현하면 식(3)과 같다. n 은 정수배로 하는 것이 카메라를 제어하기에 편리하며, 이 값이 커질수록 격자 구조의 해상도는 커진다.

$$\Delta g = \frac{F_w}{n \times 26.5} \approx \frac{F_h}{n \times 21} \quad (3)$$

3. 움직임 검출

움직임을 판단하는 근거는 이전 프레임과 현재 프레임에서 해당 격자점들의 화소값을 비교하여 차이가 존재한다면, 어떤 움직임이 있다고 판단할 수 있다. 따라서 두 프레임간 동일 위치의 대응 화소간 차분 연산을 수행한다. 이러한 차분 연산은 식(4)와 같이 표현될 수 있다. x, y 는 각 프레임의 격자구조 상에서의 교차점의 위치를 나타내며, F_t 는 현재 프레임을, F_{t-1} 은 이전 프레임을 나타내고, $G()$ 함수는 각 화소값을 의미한다.

$$D(x, y) = |G(F_t(x, y)) - G(F_{t-1}(x, y))| \quad (4)$$

그러나, 움직임의 추적에 따라 카메라의 방향을 바꾸는 상황에서는 동일한 좌표에 해당하는 차분 영상을 구하는 것은 큰 의미가 없다. 이러한 상황은 그림 2에서 보듯이, 인접한 두 프레임의 배경 화면이 카메라의 이동으로 인하여 서로 다르다. 즉, 이전 프레임에서의 특정 위치의 화소는 새로운 프레임에서는 카메라의 이동 변량 ΔP , ΔT 만큼 이동된 위치에 있게 된다. 이러한 이유로 화소값의 차분을 구하기 위한 대상 영역은 식(5)와 같이 두 프레임 영역간의 교차되는 영역에 국한하여 처리한다. 이는 서로 다른 시각에 취해진 인접한 프레임간의 교차되지 않는 영역은 비디오 프레임의 전체 영역에 비해 상대적으로 좁기 때문에 교차되는 영역에서의 화소 차분만으로도 움직임의 검출이 가능하다. 따라서 식(4)는 식(6)과 같이 두 프레임에서 동일 화소에 대해 비교가 이루어질 수 있도록 카메라의 이동 변량인 ΔP , ΔT 에 따른 Δi , Δj 를 고려하여야 한다. ΔP , ΔT , Δi , Δj 에 대해서는 아래에서 살펴본다.

$$R(F_{t-1}) \cap R(F_t) \quad (5)$$

$$D(x, y) = |G(F_t(x, y)) - G(F_{t-1}(x - \Delta i, y - \Delta j))| \quad (6)$$

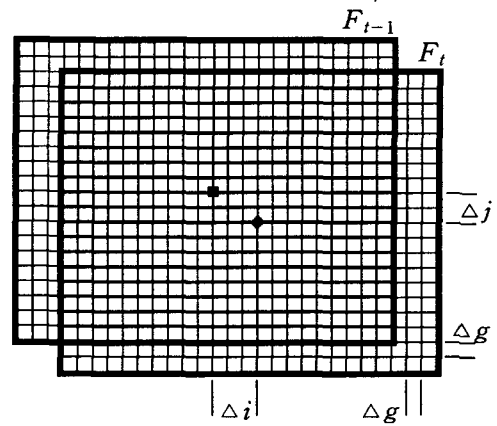


그림 2. 카메라 이동에 따른 프레임간 관계($n = 1$)

이와 같이 화소간 비교를 통해 차를 구할 때의 문제는 각 프레임의 영상들은 조도, 초점 등과 같은 주변환경에 영향을 받기 때문에 동일 장면임에도 불구하고 인접한 시간에서 취해진 두 프레임간에 대응하는 위치의 화소값이 동일하지 않다는 것이다. 따라서 이러한 환경에 따른 가변성을 수용할 수 있도록 임계치를 두어, 해당 위치의 차 연산 결과가 임계치 이하인 경우는 동일한 화소값으로 간주하고, 그 보다 큰 차이를 가지면 이동 객체가 위치한 것으로 간주한다. 계조 영상의 두 프레임에 대한 차 연산 결과의 임계치를 T 라 하자. 각 주사선의 교차점의 차이값은 $M \times N$ 의 배열 $D[M][N]$ 을 이용하여 표현하고, 식 (7)과 같이 차분값이 임계치 T 보다 크면 1로, 그렇지 않으면 0으로 표현한다. 여기서 1로 표현된 격자점의 위치에는 이동 물체의 일부분이 진입하거나, 지나간 위치일 것이다. 결국, 이 배열에서 1로 표현된 부분은 움직임이 발생된 부분으로 간주할 수 있다.

$$D[y][x] = \begin{cases} 1, & \text{for } D(x, y) > T \\ 0, & \text{for otherwise} \end{cases} \quad (7)$$

이제, 이들 격자점들에 대한 차분을 나타내는 배열을 이용하여 움직임이 많이 발생된 부분을 결정할 수 있다. 이를 위

해 식(8)에 의해 수직방향의 주사선들 상에 존재하는 격자점에 대한 누적 히스토그램인 수직투영 히스토그램(*vHisto*)과 수평 방향의 주사선 상에 존재하는 격자점에 대한 누적 히스토그램인 수평투영 히스토그램(*hHisto*)을 생성한다. 여기서 수직 방향으로의 주사선 수는 *M*개, 수평 방향으로의 주사선 수는 *N*개라 가정한다.

$$\begin{aligned} vHisto[x] &= \sum_{x=1}^N D[y][x] \\ hHisto[y] &= \sum_{y=1}^M D[y][x] \end{aligned} \quad (8)$$

다음으로, 식(9)와 같이 각 방향의 투영 히스토그램에서 누적값이 가장 큰 위치를 결정한다. 이렇게 하여 결정된 *i*, *j*로 표현되는 격자점의 화소는 다음 프레임의 중앙 좌표가 되어야 한다.

$$\begin{aligned} i &= \{x \mid \max(vHisto[x]), \text{ for } x=1, \dots, N\} \\ j &= \{y \mid \max(hHisto[y]), \text{ for } y=1, \dots, M\} \end{aligned} \quad (9)$$

움직임이 검출되었을 때, 새로운 위치로의 RCM 제어는 현재의 위치로부터 상하 혹은 좌우 방향으로의 변위만큼 더함으로써 절대 위치로 표현되는 제어 명령을 생성할 수 있다. 따라서 현재 프레임의 중심으로 요구되는 각 방향으로의 변위를 구할 필요가 있다. 각 방향으로의 변위를 각각 Δi , Δj 라 할 때, 식(10)에 의해 각 변위를 계산할 수 있다. 이 제어 변위는 비디오 프레임의 정 중앙의 위치로부터 떨어진 정도를 나타내는 것으로, 이전 프레임에서 중앙의 위치는 (*N*/2, *M*/2)로 표현될 수 있다.

$$\begin{aligned} \Delta i &= i - \frac{N}{2} \\ \Delta j &= j - \frac{M}{2} \end{aligned} \quad (10)$$

투영 히스토그램으로부터 결정된 각 방향으로의 변위인 Δi , Δj 는 격자 구조의 해상도에 따라 RCM 모듈을 제어하기 위한 명령을 생성하기 위해, 좌우 및 상하의 제어 단위로 적절히 환산되어야 한다. 이러한 환산은 식(11)에 의해 결정된다.

$$\begin{aligned} \Delta P &= \frac{\Delta i}{n} \\ \Delta T &= \frac{\Delta j}{n} \end{aligned} \quad (11)$$

이제 이전 비디오 프레임의 중앙 위치로부터의 변위가 얻어졌으므로, 이전 프레임의 중앙 위치인 P_{t-1} , T_{t-1} 위치에 변위를 더함으로써 식(12)와 같이 새로운 상하 이동, 좌우 이동의 절대 위치값 P_t , T_t 를 결정하게 된다.

$$\begin{aligned} P_t &= P_{t-1} + \Delta P \\ T_t &= T_{t-1} + \Delta T \end{aligned} \quad (12)$$

이와 같은 과정을 거쳐서 얻어진 P_t , T_t 값을 표 1의 형식으로 구성하여 직렬 포트를 통해 RCM 모듈로 전송하면, 카메라의 위치를 조절할 수 있다.

4. 서버/클라이언트 시스템

시스템의 구성은 그림 3과 같이 카메라가 장착된 RCM 모듈이 직렬 케이블에 연결되어 있는 서버 시스템과 이를 원격에서 모니터링하고 제어할 수 있는 웹브라우저가 탑재된 클라이언트 시스템으로 구성된다. 좀더 구체적으로 살펴보면 서버 시스템에는 웹 서비스를 위한 웹 서버와 원격의 클라이언트에 적재되어 비디오를 모니터링하는데 이용될 자바 애플릿, 카메라로부터 USB 포트를 통하여 영상을 획득하고 이를 특정 디렉토리에 동영상 파일로 저장하는 WebCam32 응용 프로그램, 그리고 비디오 프레임으로부터 움직임을 검출하고 이에 따른 정보를 이용하여 RCM 모듈을 제어하기 위한 RCMServer로 구성된다. 이러한 두 시스템은 인터넷을 통해 연결된다.

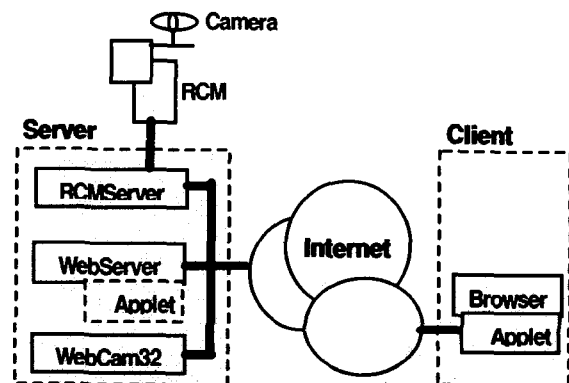


그림 3. 시스템 구성요소

수행 과정을 간략히 살펴보면, 우선 서버 시스템에서는 웹 서버와 RCMServer 그리고 WebCam320이 수행되고, 클라이언트의 접속 요청을 기다린다. 클라이언트 시스템에서 웹 브라우저를 통해 서버로의 접속을 요청하게 되면, 서버에 있던 클라이언트 모니터용 자바 애플릿이 다운로드되어 수행을 개시한다. 자바 애플릿은 서버의 RCMServer에 접속을 요청하고, 이 요청에 대해 RCMServer는 비디오 스트림을 클라이언트로 보내는 쓰레드와 RCM 제어 요청을 기다리는 다른 쓰레드를 수행하게 된다. 만일 클라이언트로부터의 자동 제어 모드로의 전환 요청이 있게 되면, RCMServer는 제안한 방법에 따라 움직임을 검출하고 RCM 모듈 제어 정보를 얻어 RCM 제어 명령을 생성하여 직렬 포트를 통해 RCM 모듈에 전송하게 된다. 그러면 제어 명령에 따라 RCM 모듈은 카메라의 방향을 바꾸게 된다. 반면 수동 제어 모드의 요청이 있게 되면, 이후의 클라이언트측 방향 제어 패널에서의 조작에 따른 명령을 RCM 모듈에 전송함으로써 카메라의 방향을 바꾸게 된다.

비디오 스트림의 전송 및 제어 모드에 따른 RCM 제어 신호의 흐름은 그림 4와 같이 표현할 수 있다. 직선의 화살표는 지속적으로 발생하는 흐름이며, 점선의 화살표는 클라이언트의 간헐적인 요청을 나타낸다. 우선, 클라이언트의 연결 접속 요청과 비디오 스트림의 전달에서의 제어 흐름을 (a)에서 보여준다. 일단 서버 사이트에 접속하면, 원격 카메라 제어 애플릿을 자동으로 다운로드하여 실행한다. 이 클라이언트측의 애플릿은 RCMServer 애플리케이션에 연결 요청을 하고, RCMServer는 접속을 수락한다. RCMServer는 WebCam320이 생성하는 비디오 스트림을 애플릿의 비디오 창에 계속하여 전송한다. 만일 클라이언트가 웹 브라우저를 종료하게 되면, 이는 접속중단 요청 신호로 간주되고 RCMServer는 대기상태에 있게 된다.

수동 제어 모드는 (b)에서 보여주듯이 애플릿 상의 제어패널의 방향 버튼을 사용자가 수동으로 조작하게 되면, 해당 방향으로 RCM을 제어할 수 있는 메시지를 RCMServer에 전달한다. RCMServer는 이 메시지를 해독하여 RCM 모듈 제어 명령으로 변환하고, 직렬 포트를 통하여 RCM 모듈로 전송하면, 명령에 따라 카메라의 위치를 변경시킨다.

자동 제어 모드에서는 (c)와 같이 자동 제어 모드로의 요청 메시지를 RCMServer가 수신하면, 제안한 방법에 따라 움직임 추적을 할 수 있도록 또 다른 쓰레드(thread)를 시작하게 한다. 자동 모드에서는 WebCAM320이 획득하는 비디오 스트림에서 매 다섯 번째마다 취한 두 프레임에 대하여 제안한 방법을 적용하여 움직임을 검출한다. 검출된 움직임의 위치 정보

를 이용하여 RCM 제어 명령을 생성하고, 이 명령을 직렬 포트를 통하여 RCM 모듈로 전송하면, 명령에 따라 카메라의 위치를 변경시킨다.

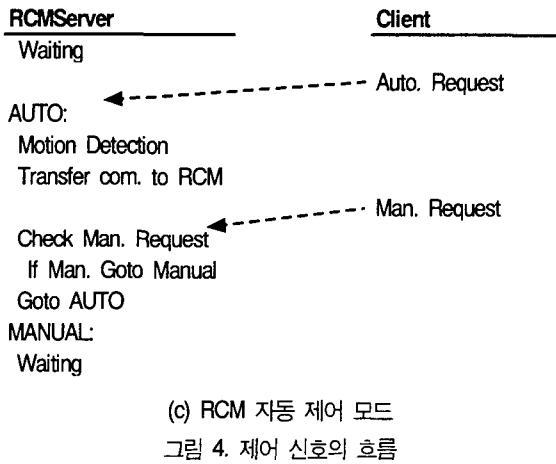
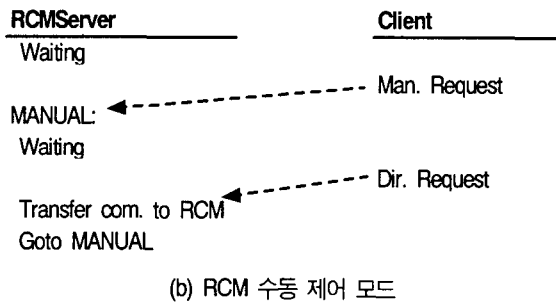
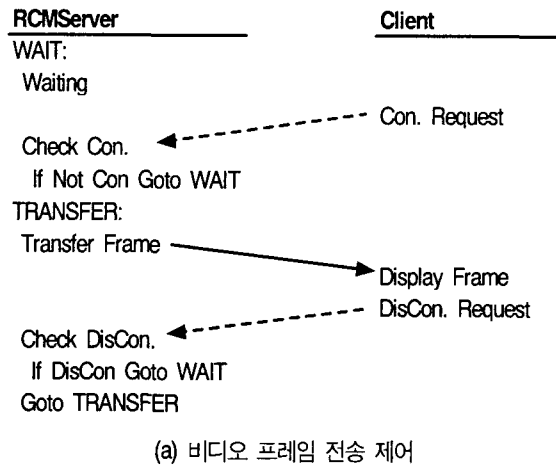


그림 4. 제어 신호의 흐름

여기에서 비디오 스트림의 전송과 수동 혹은 자동의 제어는

별도의 쓰레드로 동작함으로써 병행성을 가진다. 즉, RCM의 제어와 비디오 스트림의 전송은 다중 쓰레딩으로 처리되므로 처리 시간에서의 효율을 가져오게 된다.

III. 실험 및 고찰

인터넷을 이용하여 원격에서 카메라를 제어하면서 보안 감시 활동을 하기 위한 추적 시스템의 실험 환경 및 결과에 대해서 살펴본다.

1. 실험 환경

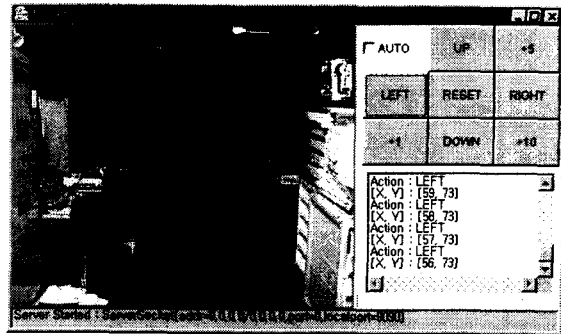
우선 실험을 위해 구현할 프로그램은 RCMServer와 클라이언트용의 Java Applet이다. 개발 도구로 Sun Microsystems사의 Java 1.3 SDK와 직렬 및 병렬 포트와 통신 기능을 제공하기 위한 확장 API인 JavaComm API를 이용한다. 이 도구를 이용하여 비디오 프레임에 취하여 움직임을 검출하고 RCM 모듈을 제어하기 위한 서버 애플리케이션인 RCMServer를 구현하고, 카메라의 수동 원격 제어 및 비디오 스트림 수신을 위해 클라이언트 시스템의 웹브라우저 상에서 실행될 자바 애플릿을 구현한다. RCMServer는 서버 시스템에서 실행되어야 하며, 클라이언트용 애플릿은 클라이언트의 요청에 의해 클라이언트 시스템으로 다운로드될 수 있도록 웹서버의 서비스 디렉토리에 보관한다.

서버 시스템은 네트워킹이 가능한 Intel Pentium II 개인용 컴퓨터를 사용한다. 카메라의 위치를 상하 및 좌우로 변경하는데 사용할 Suveyor사의 TransitRCM 장치는 제어 신호를 위해 9600bps 속도의 직렬 포트를 이용하여 서버 시스템에 연결한다. 카메라로는 삼성사의 ScanEye SNC-320 디지털 웹 카메라를 사용하며, TransitRCM 모듈에 탑재한다. 이 카메라는 352×288의 공간 해상도를 가지는 색상 영상을 하나의 프레임으로 생성하여 USB 포트를 통해 서버 시스템에 저장하게 된다. 이러한 서버 시스템에 Apache 웹 서버를 설치함으로써 웹 서비스가 가능하도록 구성하며, 구현된 클라이언트용 애플릿을 적절한 곳에 위치시킨다. 카메라를 통하여 비디오 스트림을 획득하여 연속된 비디오 스트림을 생성하여 파일로 저장하는 응용 프로그램 WebCam32과 구현된 RCMServer를 실행한다.

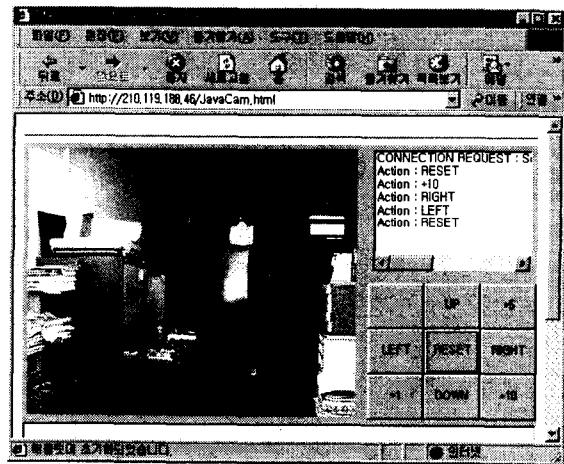
또한 클라이언트 시스템으로는 원격에서 비디오 스트림을 관찰할 수 있도록 네트워킹이 가능하고 웹브라우저가 수행중인 개인용 컴퓨터가 사용된다.

그림 5의 (a)는 서버 시스템에서 수행 중인 RCMServer의

실행 화면을 보여주며, (b)는 원격의 감시요원이 웹브라우저를 통해 클라이언트 애플릿을 수행하는 장면을 보여주는 것이다.



(a) RCMServer



(b) 클라이언트의 웹브라우저

그림 5. 시스템의 실행 화면

2. 결과 및 고찰

그림 6에서는 자동 제어모드에서 실험자의 손을 움직임으로써 움직임을 검출하여 추적하는 결과를 보여준다. 실험에 사용된 각 인자들의 값은 $k=5$, $n=1$, $T=20$ 을 사용하였다. 여기에서 움직임 검출에 이용되는 F_{t-1} 프레임을 (a)에, F_t 프레임은 (b)에 보여준다. (c)는 F_{t-1} 과 F_t 프레임을 대상으로 움직임 검출을 하여 격자점에 그 여부를 보여준다. 움직임이 대략 비디오 프레임의 우하단에 있음을 원으로 표시된 부분에서 확인할 수 있다. (d)는 움직임 추적에 의해 카메라가 이동된 후에 획득된 F_{t+1} 프레임을 보인다. 프레임 내에서 움직임 검출로 판단된 손끝 부분이 프레임의 중앙 부위에 오도록 카메라가 이동되었음을 확인할 수 있다.

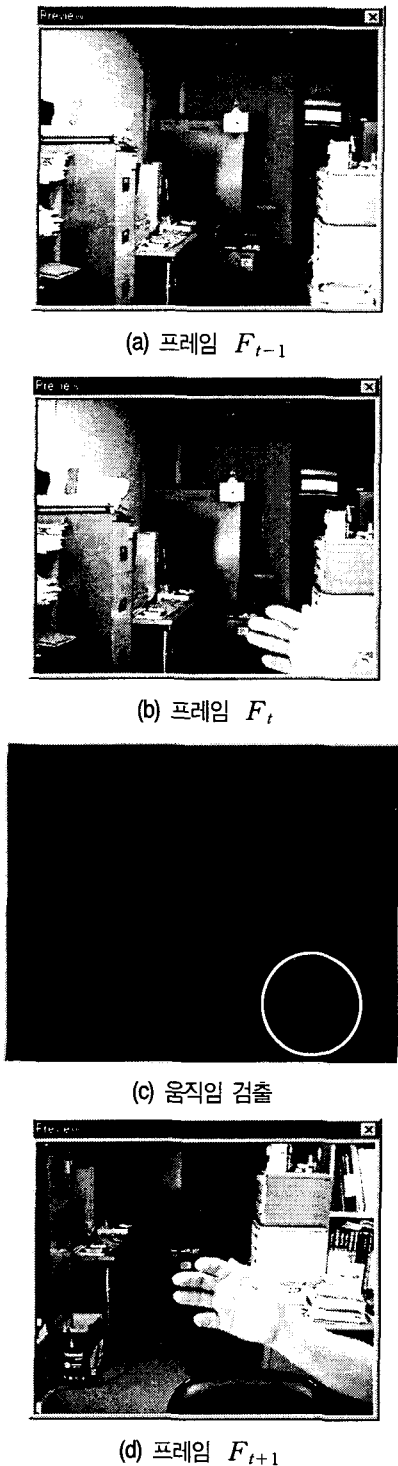


그림 6. 움직임 추적 결과($k=5, n=1$)

(c)의 결과를 살펴보면, 움직임 검출에서 조도의 가변성으로

인하여 차분 영상의 곳곳에 오류성 움직임 검출이 있음을 볼 수 있다. 또한 제안한 방법에 의해 움직임이 검출되었다고 해서 그 부분이 관찰하고자 하는 대상 객체의 중심이 아니므로, 관찰 객체를 추적하는 목적의 응용을 위해서는 객체 인식을 통한 추적 연구가 필요할 것이다.

객체가 화면의 중앙에 안정되게 수렴된 상태에서 객체가 이동하여 정지해 있을 때, 다시 안정된 상태로 이동하는데 필요한 시간을 수렴 시간이라고 가정하면, 실험을 통하여 5.13 프레임 처리시간 만큼 후에는 수렴하는 것을 확인하였다. 즉, 평균 5 혹은 6 프레임이 경과된 후에는 목표하는 객체를 비디오 스트림의 중앙에 위치시킬 수 있음을 의미한다. 또한 격자 해상도에 따른 처리 시간과 프레임 처리율(FPS, Frame/Sec)은 표 2와 같이 나타났다. 이는 적절한 격자 간격을 줌으로써 차분 영상을 생성하기 위한 계산량을 줄일 수 있음을 보여준다.

표 2. 격자 해상도에 따른 처리시간 및 프레임처리율

n	$\Delta g(\text{pixels})$	처리시간(ms)	FPS
1/2	6.6	353	2.83
1	13.28	386	2.59
13	1	424	2.36

IV. 결론

본 연구는 인터넷을 이용한 웹기반의 클라이언트 시스템이 원격에 있는 서버 시스템에 연결되어 있는 카메라의 방향을 수동으로 제어함으로써 원하는 장면을 감시할 수 있을 뿐만 아니라, 움직임 검출을 이용하여 이동 객체를 추적하는 자동 제어 모드를 통해 움직이는 객체를 비디오 프레임의 중심에 지속적으로 확보하면서 원격에서 감시할 수 있는 방법에 관한 것이다.

특히, 이동하는 객체에 대한 움직임을 검출하기 위해 수평 및 수직 주사선에 의한 격자 구조의 각 격자점에서의 화소값 차분에 대한 투영 히스토그램에 의해 카메라의 새로운 위치 정보를 결정하고, 이 정보를 이용하여 비디오 스트림의 중앙에 움직임 부분이 위치하도록 RCM을 제어함으로써 비디오 프레임내에 움직이는 객체를 자동 추적하여 감시할 수 있게 하였다. 본 연구를 통하여 이동하는 객체를 추적하도록 카메라를 이동시킴으로써 원격지에서 지속적인 이동 객체의 감시가 가능함을 알 수 있었다.

향후 연구 방향으로는 예기치 않은 상황이 발생하였을 때 이에 대한 경고를 이동 단말기에 알리거나 관련 비디오 스트림을 전송할 수 있도록 이동 통신 기술을 접목할 필요가 있을 것으로 생각된다.

참 고 문 헌

[1] 박천주, 박희정, 이재협, 전병민, "인터넷 기반 이동 객체 자동 추적 시스템에 관한 연구", 한국통신학회, 하계 종합학술발표회논문집(하), pp.1402~1405, 2001.

[2] A. Bakowski, G. A. Jones, "Video Surveillance Tracking using Color Region Adjacency Graphs", Image Proceeding and its Application, pp.794~798, 1999.

[3] R. Jain, D. Militzer, H. Nagel, "Separating Non-stationary from Stationary Scene Components in A Sequence of Real World TV Images", Proceedings in Fifth International Joint Conference Artificial Intelligence, pp.425~428, 1977.

[4] F. Bremond, M. Thonnat. "Tracking multiple non-rigid objects in a cluttered scene", In Proceedings of the Scandinavian Conference on Image Analysis, pp.643~650, 1997.

[5] M. Hotter, R. Mester, M. Meyer, "Detection of Moving Objects using Robust Displacement Estimation including Statistical Error Analysis", In International Conference Pattern Recognition, pp.249~255, 1996.

[6] P. L. Posin, T. Ellis, "Image Difference threshold strategies and shadow detection", In Proceedings of British Machine Vision Conference, pp.347~356, 1995.

[7] J. Segen, S. Pingali, " A Camera-based System for Tracking People in Real Time", In Proceedings of IEEE International Conference Pattern Recognition, pp.63~67, 1996.

[8] B. K. Horn, B. G. Schunck, "Determining Optical Flow", Artificial Intelligence, Vol.17, pp.185~203, 1981.

[9] A. M. Earnshaw, S. D. Blostein, "The Performance of Camera Translation Direction Estimators from Optical Flow : Analysis, Comparison and Theoretical Limits",

IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.18, No.9, 1996.

[10] J. Stepan, M. Jirina, "Moving Object Tracking in the Sequence of Images Acquired from Non-stationary Camera", IEEE, pp.353~356, 2001.

[11] <http://www.surveyor.com/>

박천주(Cheon-Joo Park)

정회원



1991년 2월 : 충북대학교 전산기공학과 졸업(공학사)

1993년 2월 : 충북대학교 컴퓨터공학과 졸업(공학석사)

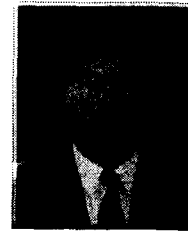
2001년 2월 : 충북대학교 컴퓨터공학과 수료(공학박사과정)

1993년 5월 ~ 현재 : 한국기술교육대학교 정보기술공학부 기술연구원

<관심분야> : 패턴인식, 영상처리, 컴퓨터비전

박희정(Hee-Jung Park)

정회원



1990년 2월 : 충북대학교 전산기공학과 졸업(공학사)

1997년 2월 : 원광대학교 컴퓨터공학과 졸업(공학석사)

2001년 2월 : 충북대학교 컴퓨터공학과 수료(공학박사과정)

1997년 9월 ~ 현재 : 안동과학대학 정보처리과 교수

<관심분야> : 영상처리, 컴퓨터그래픽, 컴퓨터비전

이재협(Jae-Hyub Lee)

정회원



1984년 2월 : 홍익대학교 화학공학과 졸업(공학사)

1987년 5월 : 일리노이공대 전산학과 졸업(공학석사)

1992년 5월 : 일리노이공대 전산학과 졸업(공학박사)

1993년 3월 ~ 현재 : 한국기술교육대학교 정보기술공학부 교수

<관심분야> : 컴퓨터그래픽스, 멀티미디어, 화상처리

전병민(Byung-Min Jun)

종신회원



1976년 2월 : 한국항공대학교 전자공학과 졸업(공학사)

1978년 2월 : 연세대학교 전자공학과 졸업(공학석사)

1988년 8월 : 연세대학교 전자공학과 졸업(공학박사)

1986년 2월 ~ 현재 : 충북대학교 전기전자및컴퓨터공학부 교수

<관심분야> : 디지털 신호처리, 디지털 영상처리