碩士學位論文

블록 정합을 이용한 반복적인 움직임 검출

濟州大學校 大學院

컴 퓨 터 工 學 科

高 奉 秀

2003年 12月

블록 정합을 이용한 반복적인 움직임 검출

指導敎授 金 壯 亨

高 奉 秀

이 論文을 工學 碩士學位 論文으로 提出함.

2003年 12月

高奉秀의 工學 碩士學位 論文을 認准함.

審査委員長 邊 翔 庸 印

委 員 金 壯 亨 印

委 員 李 尙 浚 印

濟州大學校 大學院

2003年 12月

The Recusive Motion Detection Using Block Matching

Bong-Soo Ko
(Supervised by professor Jang-Hyung Kim)

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENT FOR THE DEGREE OF
MASTER OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING
GRADUATE SCHOOL
CHEJU NATIONAL UNIVERSITY

2003. 12.

목 차

[그림 목차]

[표 목차]

Summary

This paper presents the motion detection algorithm that can run robustly about recusive motion. The motion detection compares and analyzes two frames each other, motion of whether happened judge. The existing motion detection algorithm that uses difference image is robustly in some degree brightness or noise, but it frequently causes false alarms to temporal clutter, at the repetitive motion within a certain area. This example of recusive motion blink and so on of action, action of electric fan, flame color of object by wind of be. Such recusive motion in schedule area of reflex motion of detect. Therefor we developed a motion detection algorithm using mean-absoulte-error which calculates the set of moving regions and performs block matching. mean-absoulte-error is small when is recursive motion, and in case invader is reflex that happen, mean-absoulte-error is big fairly. The experimental results revealed that our approach is superior to existing methodolgies to handling various temporal clutter.

Ⅰ. 서 론

최근 컴퓨터 시스템의 발달로 인해 무인 영상 감시 장치에 대한 관심이 급증하고 있는 가운데 있어, 영상 감시 시스템의 주요 목적은 입력되는 영상을 분석하여 움직임을 감시하고 저장하는 기능이다. 이러한 감시 시스템의 기능을 수행하는데 있어 입력된 영상에서 움직임의 검출되었을 경우에만 저장 할 수 있도록 하면 감시 시스템에 저장 공간을 줄이고 차후에 저장된 영상에서 침입자가 발생한 부분 영상을 보다 쉽게 검색 할 수 있어 효율적인 감시 기능이 가능하게 된다. 이러한 이유로 인해 무인 영상 감시 장치에서 움직임을 검출하는 여러 가지 방법들이 연구 되었다.[1]

영상 감시 시스템에서 움직임을 검출하는 방법에는 크게 프레임 간 차이법과 배경 차이법으로 나눌 수 있다.[2] 프레임 간 차이법은 현재 입력된 입력 영상과 바로 직전에 입력된 영상이 픽셀값들을 서로 비교 분석하여 움직임을 검출 하는 방법이고 배경 차이법은 현재 입력된 영상과 이전에 입력된 모든 영상들이 평균값을 구하여 배경 영상을 만들고 현재 영상과 비교 분석하여 움직임을 검출해 내는 방법이다. 이러한 여러 가지 움직임 검출 기법들 중 현재 가장 많이 사용되는 움직임 검출 기법에는 차영상 움직임 검출 기법이 있다. 차영상 움직임 검출 기법은 기존의 다양한 움직임 검출 기법 중 가장 처리속도가 빠르고, 알고리즘 구현이 쉽기 때문에 실제 많이 사용되는 움직임 검출 기법이다.[3] 하지만 이 기법은 입력 장치에 유입되는 잡음이나 반복적인 움직임을 갖는 물체의 동작에 상당히 민감하게 동작 함으로써 신뢰성 있는 움직임 검출이 어렵다는 문제점을 가지고 있다.[4][5]

따라서 본 논문에서는 이러한 차영상 움직임 검출 기법의 문제점을 개선하기 위해 반복적인 물체의 움직임을 검출하는 알고리즘을 제시 하고자 한다. 제시한 알고리즘은 먼저 배경이 되는 이미지와 현재 이미지의 차를 이용하여 현재 움직임의 발생 하였는지 판단하고, 움직임의 발생 하였을 경우 두 이미지간 차영상(Difference Image)을 이진화 처리하여, 움직임의 발생한 영역을 블록 영역으로 추출 하고, 배경 영상과 현재 영상에 추출된 블록 영역의 범위를 설정하여, 두 블록 영역을 정합(Matching)시킨다.[6][7] 그리고 나서 두 블록 영상 간에 평균절대오차(Mean Absolute Error :

MAE)값을 구하여, 평균절대오차값이 작으면 반복된 물체의 움직임으로 판단하고, 반대로 값이 크면 새로운 움직임으로 판단하도록 하였다.[8][9] 이때 반복된 물체의 움직임인 경우에는 배경이 되는 영상을 현재 입력된 영상으로 갱신하고, 새로운 움직임인 경우에는 갱신되지 않도록 하였다.

논문의 구성은 Ⅰ장 서론에서는 본 논문의 연구 동향에 대하여 서술하고, 논문에서 제안한 알고리즘의 필요성을 제시하였으며, Ⅱ장에서는 영상 검출 기법의 원리와 움직임 검출 기법의 개요에 대해 서술하였다. 그리고 각각 Ⅲ장에서는 본 논문에서 제안한 알고리즘의 개요와 그 구조에 대해서, Ⅳ장에서는 실제 실험을 통해 그 결과를 기존의 움직임 검출 기법과 비교하여 보여주고 있으며, 끝으로 Ⅴ장에서는 본 논문에서 제안한 알고리즘의 실용성과 향후 연구 과제 및 방향에 대해 서술 하였다.

Ⅱ. 움직임 검출 기법의 개요

1. 움직임 검출 기법의 필요성

최근 들어 컴퓨터를 이용한 영상 감시 장치 및 영상회의 시스템의 응용을 목적으로 한 움직임 추적 시스템에 대한 연구 개발의 널리 진행되고 있다. 일반적으로 무인 영상 감시 시스템에서 감시 카메라는 침입자를 감시하기 위한 수단으로 카메라를 통해 들어온 영상은 감시 시스템에 녹화, 저장된다.

하지만 침입자가 없는 반복적인 영상들을 지속적으로 모니터링 하여 녹화, 저장하는 것은 감시 시스템의 저장 공간 낭비를 초래하고, 침입자가 발생한 영상을 검색하는데 있어 많은 시간과 비용을 투자해야 하므로 인해, 감시 시스템의 성능에 있어서는 매우 비효율적이다. 따라서 감시 카메라를 통해 입력 받은 영상들을 분석하여 침입자가 발생했다고 판단되었을 때만 경고음을 들려 주거나, 그 입력 영상이 저장될 수 있도록 하면 무인 영상 감시 시스템의 성능을 더욱 더 높일 수 있다.[1]

또한 이렇게 감시 카메라를 통해 녹화된 영상 파일을 저속 네트워크망을 통해 실시간으로 전송하는데 있어서도 움직임 검출 기법이 필요하다. 즉 녹화된 영상 파일을 실시간으로 전송하기 위해서는 빠른 영상 전송 기술이 필요로 하게 되는데, 이럴 경우 고가의 전송라인 임대료를 지불해야 하므로 비용면에서 상당한 어려움의 따르게 된다. 반면에 저속 전송망을 통해 실시간 영상파일을 전송하기 위해서는 고압축의 영상 압축 기술을 필요로 하는데, 이때 이러한 고압축의 영상 압축 기술을 적용하기 위해서는 영상간의 중복성이 제거 되어야 한다. 이러한 영상간의 중복성을 제거하기 위해서는 제일 먼저 영상에서 움직임 영역을 검출하는 처리가 이루어 져야 한다. 이렇듯 움직임 검출 기법은 무인 영상 감시 시스템이나 영상 압축 기술 및 움직임 추정 기법등에 있어 우선적으로 고려되어야 할 사항이다.

2. 움직임 검출 기법의 개요

영상에서 움직임 검출은 입력되는 3차원의 실세계의 영상을 2차원 데이터로 나타내고 이 데이터에서 움직임이 있는 부분만을 영역화 하여 표현 가능하다. 즉 부분 영역화란 입력되는 정보로부터 동일한 특징을 갖는 영역으로 구분하는 과정을 의미한다. 주로 영상에서 사용되는 움직임 검출 기법은 움직임이 있는 영상 정보를 일정한 시간 간격으로 입력받아 입력된 두 영상을 서로 비교하여 영상 정보의 차를 조사, 분석하여 움직임으로 판단하는 방법을 사용한다. 즉 입력된 영상의 각 프레임들을 서로 비교 분석하여 움직임이 발생한 영역을 찾아내고, 그 영역의 특징들을 이용하여 움직임을 인식하거나 정의한다.

현재까지 다양한 움직임 검출 알고리즘들이 연구되고 개발되었는데, 이러한 움직임 검출 알고리즘들은 크게 연속적인 영상을 분석하여 그 영상의 특정 화소나 혹은 영역에 특정 벡터를 할당하여, 이전의 영상의 화소나 벡터를 비교 분석하여 가장 유사한 영역을 움직임 영역으로 할당하고, 이 영역의 특징을 분석하여 움직임을 검출하는 형태학적인 움직임 검출 기법과 두 영상간의 명암차(Difference of gray-value)에 의하여 형성된 차영상(Difference Image)을 분석하여 물체의 움직임을 검출하는 기법 등이 있다.

형태학적인 움직임 검출 기법에는 영상에서 검출된 에지(Edge)를 이용하여 움직임을 검출하는 방법과 특징이 되는 점(Point)을 이용하여 검출하는 방법, 그리고 특징선(Line) 벡터를 이용하여 움직임을 검출하는 방법 등이 있다.

명암차를 이용하여 움직임을 검출하는 방법에는 두 이미지간에 점대점 대응관계로 픽셀들을 서로 비교하여 차영상(Difference Image)를 구하고 분석하여 움직임을 검출하는 방법과 이미지를 여러개의 블록(Block)으로 나누고 두 이미지간에 블록을 서로 정합(Matching) 시켜 차영상(difference image)를 구하고, 움직임을 검츨하는 방법이 있다.[2][3]

다음은 Fig. 1은 지금까지 개발되고 연구된 다양한 물체의 움직임을 추정 및 검출하는 알고리즘을 분류한 그림을 보여주고 있다.

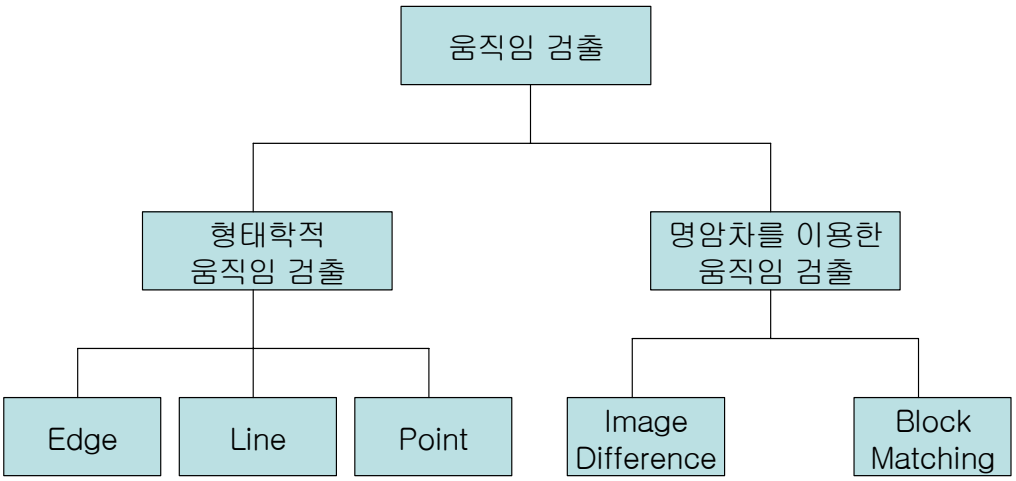


Fig. 1 A Distribution Diagram of Moving Detection Algorithm

3. 동영상에서 움직임 검출 기법

현재 연구되고 있는 대부분의 움직임 검출 기법(Motion Detection)은 입력되는 동영상을 분석하여 움직임을 검출한다. 동영상은 매초 동안 여러 개의 이미지가 시간에 순차적으로 배열된 것으로써 이는 전체 동영상을 시퀀스(Sequence)라는 것으로 나누고, 이 시퀀스(Sequence)는 다시 장면(Scene)으로 나뉘며, 장면은 샷(Shot) 으로 분할되고, 각각의 샷은 프레임(Frame)으로 나뉘게 된다.

이러한 동영상의 구조에서 움직임을 검출하기 위해서는 먼저 입력된 방대한 동영상들을 분석하여 움직임 검출 비교 대상으로 사용될 프레임(Frame)들을 추출해야 한다. 일반적으로 무인 영상 감시 시스템에서는 카메라를 통해 들어오는 입력 동영상은 중요한 객체 및 시점의 변화가 나타나고 사라지는 시점을 기준으로 작은 단위인 장면(Scene)로 나눌 수 있으며, 이러한 장면(Scence)들은 다시 여러 개의 프레임들로 나뉘어 진다. 이때 이렇게 장면의 어느 시점을 기준으로 변화하는 부분을 동영상에서는 장면 전환 지점이라고 한다.

다음 Fig.2는 다수개의 프레임들로 구성된 동영상의 구조를 보여주고 있다.

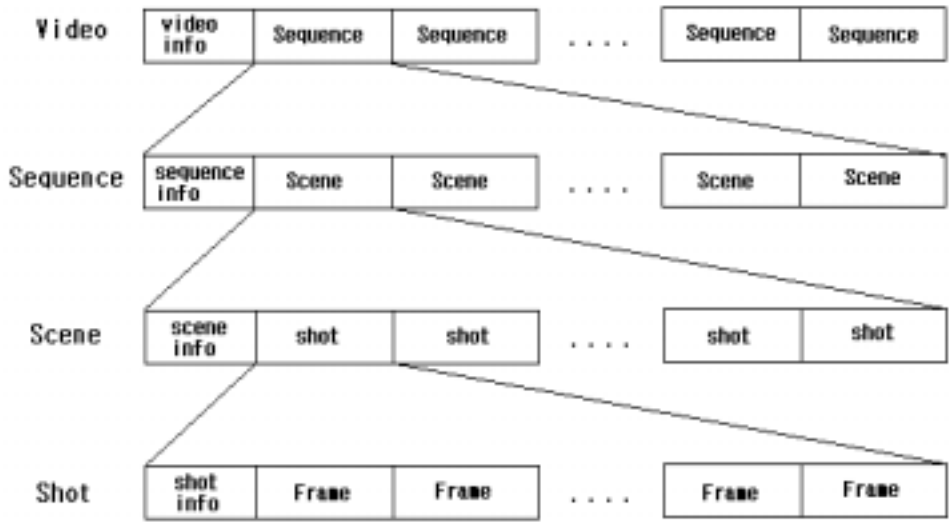


Fig.2 A Moving Picture Hierarchy

이렇게 입력된 동영상을 가지고 움직임을 검출 할 때는 시간에 순차적으로 입력되는 영상의 프레임(Frame)들 중 비교 대상으로 사용할 키 프레임(Key Frame)을 추출하여 이 프레임들을 서로 비교, 분석하여 움직임을 검출한다.

동영상의 계층 구조에 있어서 시퀀스(Sequence)나 장면(Scence)은 고도의 내용 정보(Sementic Infomation)에 의존하는 개념으로 현재의 기술로는 자동화하기가 매우 곤란한 반면, 샷 개념은 하나의 동작이 끝나고 다른 동작으로 넘어가기까지의 프레임들의 집합이므로 일반적으로 그 경계 부근에서 커다란 변화를 보여주기 때문에 영상처리 기술을 이용한 자동 검출이 가능한 부분이라 할 수 있다.

따라서 키 프레임(Key frame)을 추출하기 위해서는 카메라로부터 입력되는 영상의 한 장면(Scene)에서 다른 장면(Scene)으로 전환되는, 장면 전화 지점(Scence Change)을 찾고, 이 장면 전화 지점의 전, 후 장면(Scence)을 구성하는 샷(Shot)과 이 샷(Shot)을 구성하는 여러개의 프레임(Frame)들을 찾는다.

이때 이 프레임(Frame)들은 전체적인 구조나 색상 분포등의 동일한 특성들을 지니

게 되는데. 이러한 여러개의 프레임(Frame)들 중에서 하나를 키 프레임(Key frame)으로 지정하여 움직임을 검출하는데 사용하면 된다.

하지만 이때 비교 분석되는 키 프레임(Key Frame)을 추출 할 때 동영상은 입력 중간에 동영상의 정보를 변화시키거나 제거시키는 이미지 처리(Image Processing)작업이나 동영상 압축과 같은 변환 처리는 없어야 된다. 즉 입력되는 동영상에서 특정한 처리 과정을 거치지 않은 입력된 고유의 영상 데이터를 가지고 움직임 검출을 해야 한다. 만약 화질 개선을 위한 이미지 처리(Image Processing) 작업이나 혹은 저전송 네트워크 망을 통해 입력된 동영상을 전송하기 위한 영상 압축 과정을 거친 후에는 입력된 동영상의 프레임들의 손실을 입게 됨으로 인해 정확한 움직임 검출 작업이 어렵게 된다.

4. 블록 정합 알고리즘

블록 정합 알고리즘 (Bolck Matching Algorithm)은 원래 디지털 영상 신호전송에서 프레임 간 중복성을 줄이기 위한 움직임 추정 방법으로 현재 영상 프레임과 이전 영상 프레임의 제한된 후보 영역(Candidate Area)들 중에서 최적의 정합점(Matching Point)을 찾아내는 방법이다.[6][7]

일반적으로 각 블록은 중첩되지 않은 상태에서 대상을 인식하여 대상 블록이 새로운 위치를 나타내는 변위 벡터(Displacement Vector)를 구하게 된다. 이때 블록이 크기가 커지면 이전 프레임에서 일치하는 블록을 찾는 탐색 시간은 줄어들고 영상의 화질은 떨어지게 된다.

블록 정합이 주 원리는 현재 프레임의 블록과 이전 프레임 중, 후보 블록을 선정하고 정합 함수(Matching Function : MF)를 계산하는 것이다. 이 과정을 후보 블록 전체에 대해 반복한 후 가장 정합이 잘된 블록을 결정하고 여기서 결정된 블록의 위치와 이동전 위치까지의 거리와 방향의 추정된 변위 벡터이다. 만약 화상의 블록크기가 N × M이고, 최대 수평, 수직 방향이 변위량을 각 d_{mzx-x} , d_{max-y} 라고 하면 모든

탐색영역(Search Region)의 크기는 Fig.3 처럼 (N + $2d_{mzx-x}$) × (M + $2d_{max-y}$)가 된다. 이때 가능한 탐색점이 수는 총 ($2d_{mzx-x}$ + 1) × ($2d_{max-y}$ + 1)이 된다.

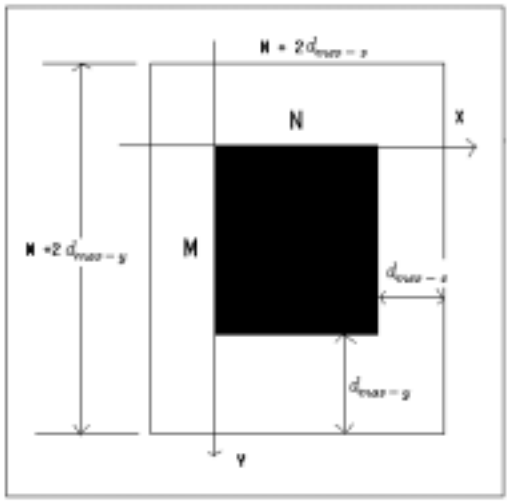


Fig.3 Block Matching Search Area

블록 정합을 위해 많이 사용되는 정합함수(Matching Function : MF)로는 평균절대오차(Mean Absolute Error : MAE), 상호 상관함수(Cross Correlation Function : CCF), 평균자승오차(Mean Squared Error : MSE), 최소화된 최대오차(Minimized Maximum Error Function : MME)등의 있다. 이들 중 계산량이 적어, 알고리즘 구현의 용이한 평균절대오차(MAE)가 가장 많이 사용되고 있다.

다음 식(1)과 식(2)는 각각 평균절대오차(MAE)와 평균자승오차(MSE)를 구하는 식을 나타내고 있다. 여기서 NM은 후보블록의 크기를 나타내고, f_t와 f_{t-1}은 각각 현재 프레임의 후보 블록과 이전 프레임의 블록을 나타낸다.

$$MAE(d_{1,}d_2) = \frac{1}{NM}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}| f_t(n,m) - f_{t-1}(n-d_{1,}m-d_2) \quad (1)$$

$$MSD(d_{1,}d_2) = \frac{1}{NM}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}[f_t(n,m) - f_{t-1}(n-d_{1,}m-d_2)]^2 \quad (2)$$

1) 전역탐색 알고리즘

블록 정합(Block Matching)에서 이동 가능한 최대 거리가 각각 d_{max-x}및 d_{max-y}이므로 현재 프레임의 블록과 같은 좌표를 가지는 이전 프레임의 블록을 중심으로 탐색 영역은 수평, 수직 방향으로 N과 M만큼씩 이동시키면서 탐색 범위내의 가능한 모든 블록을 조사하여 평균 절대 오차(MAE)를 계산하는 방법이다.
이러한 전역 탐색 알고리즘은 탐색범위 내에서 가장 적합한 블록들을 찾을 수 있지만 계산량이 그만큼 많아지므로 실제 실시간 비디오 코딩이나 소프트웨어 구현에 많은 어려움을 가지고 있어 이를 해결하기 위해 여러 가지 고속 블록 정합 알고리즘(Fast Block Matching Algorithm : FBMA)들을 사용한다.[8]

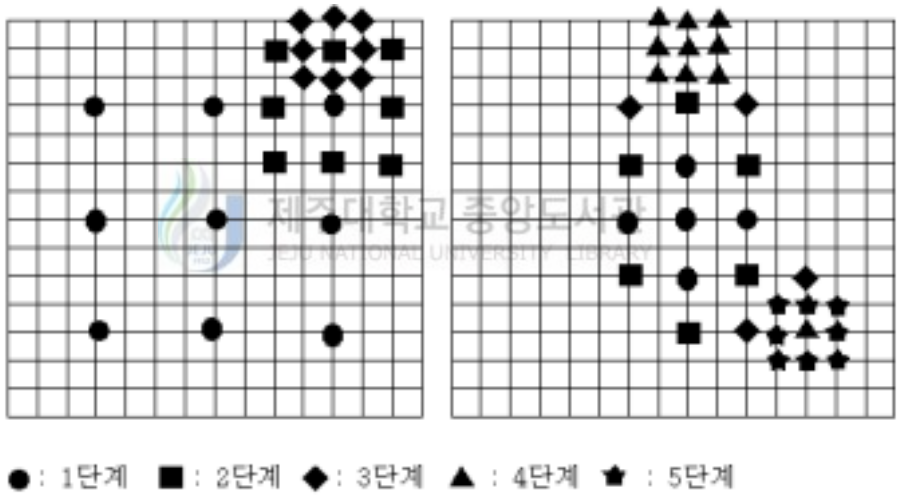
2) 고속 탐색 알고리즘

전역 탐색 알고리즘 자체가 계산량이 많아 실제 구현에 사용 할 수 없는 문제점을 해결하기 위해 나온 블록 정합 알고리즘(Block Matching Algorithm)으로 대표적인 고속 탐색 알고리즘(FSA)으로는 3단계 탐색 알고리즘(Three Step Search Algorithm : TSS), 2D-log 탐색 알고리즘(2 Dimension LOGarithmic Search Algorithm : 2D-LOG), 4단계 탐색 알고리즘(Four Step Search Algorithm : 4SS), 2단계 탐색 알고리즘(2 Step Search Algorithm : 2SS)[9], 다이아몬드 탐색 알고리즘(Diamond Search Algorithm : DS)[10]등이 있다.

Koga에 의해 제안된 3단계 탐색 알고리즘은 가장 고속 정합이 가능한 알고리즘으로서, 탐색 영역의 반의 크기로 탐색 영역의 중심에서 시작하여 각 단계마다 앞 단계 탐색 범위의 반으로 줄여 가면서 반복 계산하여 이동 벡터를 구하는 방법이다.[11]
하지만 이 알고리즘은 정합 속도는 빠르지만, 정밀한 정합을 얻을 수 없다는 단점이 있다.

A. K. Jain이 제안한 2D-log 탐색범은 이진 탐색(Binary Search)를 확장한 형태로서 블록의 좌상의 화소를 기준으로 하여 움직임 예측의 변위화소 길이만큼 떨어진 4개의 점에 대한 평균 절대 오차(MAE) 값를 구하고 이를 5개의 점에서 평균 절대 오차(MAE)값의 가장 작은 값의 위치를 중심으로 하여 1단계 과정을 반복한다. 이때 탐색 영역의 경계선에 도달하거나 최소점이 중심에서 나타나면 변위화소 길이를 반으로 줄여서 1단계 과정을 반복한다. 이러한 과정을 탐색간격이 1이 될 때까지 반복한 후 이의 최종 길이를 이동 벡터로 간주하는 방법이다.[12]

다음 그림 Fig.4는 각각 3단계 탐색 알고리즘(3SS), 2D-log 탐색 알고리즘(2D-LOG)의 움직임 추정 과정을 보여주고 있다.



(가) 3SS (나) 2D-LOG

Fig.4 Search Algorithm

5. 차영상 움직임 검출 기법

차영상 움직임 검출 기법은 현재 입력된 동영상의 키 프레임(Key Frame)과 현재

바로 직전에 입력된 키 프레임(Key Frame)간에 특정 영역이나 혹은 전체 프레임 영역의 모든 픽셀 값들을 서로 비교하여 특정 임계값(Threshold) 이상 차이가 있으면 움직임의 발생한 것으로 인식하여 움직임을 검출하는 방법이다. 이때 특정 임계값은 현재 입력되는 영상의 주위 환경 및 입력 장치의 특징들을 고려하여 보통 사용자가 임의로 변경 설정 할 수 있도록 하는 방법이 많이 사용되고 있다. 다음 식(3)은 픽셀 값을 이용하여 차영상(Difference Image)을 구하는 식을 보여주고 있다.

$$DI(x,y) = | Ft(x,y) - F_{t-1}(x,y) | \quad (3)$$

여기서 $F_t(x,y)$는 시간 t일때 입력된 영상의 키 프레임(Key Frame)을 나타내고, $F_{t-1}(x,y)$는 t-1일때 입력된 영상에서 추출된 키 프레임(Key Frame)을 각각 나타내고 있다.

이때 $DI(x,y)$는 시간 t일때 입력된 영상의 키 프레임(Key Frame)에서 시간 t-1일때 입력된 영상의 키 프레임(Key Frame)에 서로 대칭되는 두 좌표의 픽셀값들을 뺀 차영상(Difference Image)으로 차영상 $DI(x,y)$ 픽셀값들의 합이 사용자가 지정한 특정 임계값 이상인 경우에만 침입자가 발생한 것으로 간주된다.

하지만 이러한 차영상(Difference Image) 움직임 검출 기법은 침입자를 검출하는데 있어서는 빠른 처리 속도를 가지지만, 조명이나 빛에 의한 영상의 밝기 변화나 잡음이 입력된 영상, 혹은 물체의 반복적인 움직임이 발생한 입력 영상들에 대해서는 비교 대상이 되는 두개의 키 프레임의 픽셀값들의 조그만 차이가 있더라도 침입자가 발생한 것으로 인식하는 잘못된 움직임 검출이 이루어진다.

따라서 차영상 움직임 검출 기법은, 실제 침입자를 감시하는데 있어 신뢰성 있는 움직임 검출이 어렵다.

또한 이 움직임 검출 기법에서는 차영상(Difference Image)의 픽셀값들의 합과 서로 비교할 임계값(Threshold)를 설정하는데 있어서도 현재 입력 장치나 감시하는 배경이 되는 주변 환경들을 두루 고려하여 가장 적당한 임계값을 찾아 설정해야 하는데 이러한 임계값을 자동으로 설정하는 부분 역시 실제로는 매우 어렵다.

Fig.3은 차영상을 이용한 움직임 검출 기법(The Motion Detection Using Difference)의 전체 흐름도를 보여주고 있다.

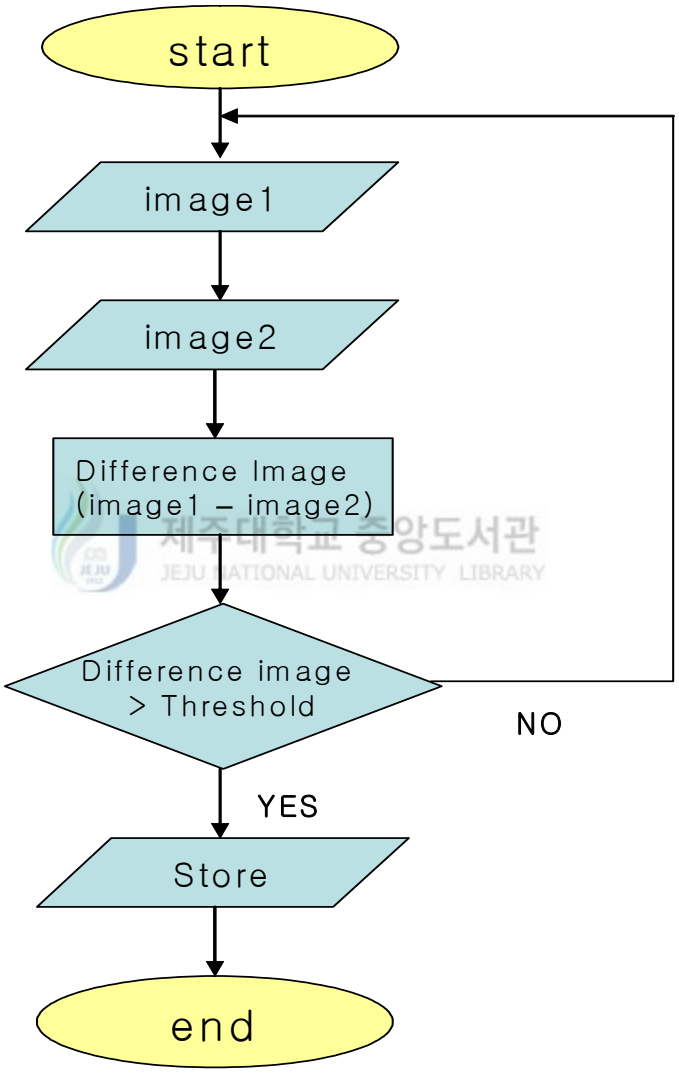


Fig.5 Flow Diagram that The Motion Detection Using Difference Image

6 히스토그램을 이용한 움직임 검출 기법

히스토그램을 이용한 움직임 검출 기법은 입력 영상의 전체적인 밝기(Brightness) 및 명암 대조(Contrast)에 대한 정보를 쉽게 분석 할 수 있는 히스토그램의 특징을 이용하여 움직임을 검출하는 방법으로서, 이전 키 프레임(Key frame)의 히스토그램과 현재 입력된 영상의 키 프레임의 히스토그램을 서로 차 연산 하여 특정 임계값 이상 차이가 발생 하면 침입자가 발생한 것으로 인식하는 움직임 검출 방법(Motion Detection) 이다. 다음 식(4)는 히스토그램(Histogram)을 이용하여 차영상을 구하는 식을 보여주고 있다.

$$DI(k) = |\,(I_K/N) - (B_K/N)\,|\ (k = 0, 1, 2 \ldots, 255) \tag{4}$$

여기서 I_k/N 은 현재 입력 영상에서 들어온 키 프레임의 히스토그램(Histogram)을 나타내고, B_k/N은 배경이 되는 영상의 히스토그램을 정의하는 식으로써, 각각 I_k와 B_k는 현재 입력 영상의 키 프레임과 배경의 되는 영상의 키 프레임에 gray값 k를 갖는 픽셀의 계수를 표현하며, N은 키 프레임의 총 픽셀 계수를 나타낸다. 하지만 이 방법 역시 입력 영상에 유입되는 잡음이나, 영상의 밝기 변화 및 반복적인 움직임의 발생하는 영상에 대해서는, 침입자가 입력영상에 없더라도, 침입자가 발생한 것으로 오 인식하는 문제점이 여전히 발생한다.

이러한 문제점들을 해결하기 위해 히스토그램(Histogram)을 이용한 움직임 검출 기법에서는 히스토그램 평활화(Histogram Equalization) 및 특정 임계값(Threshold)을 설정하는 방법의 제시되고 있으나, 이러한 방법 역시 밝기나 잡음에는 어느 정도 강건하게 움직임의 검출되지만 반복적인 움직임의 발생하는 영상에 대해서는 앞선 차영상 움직임 검출 기법(The Motion Detection Using Difference Image)과 마찬가지로, 침입자가 발생한 것으로 오 인식하여 움직임을 검출하는 문제점이 여전히 발생한다.[15] 다음 식(5)는 히스토그램 평활화 처리를 위한 수식이고 그림 Fig.6은 히스토그램의 평활화 단계를 보여주고 있다.

$$h(i) = \frac{G_{\max}}{N_t} H(i) \tag{5}$$

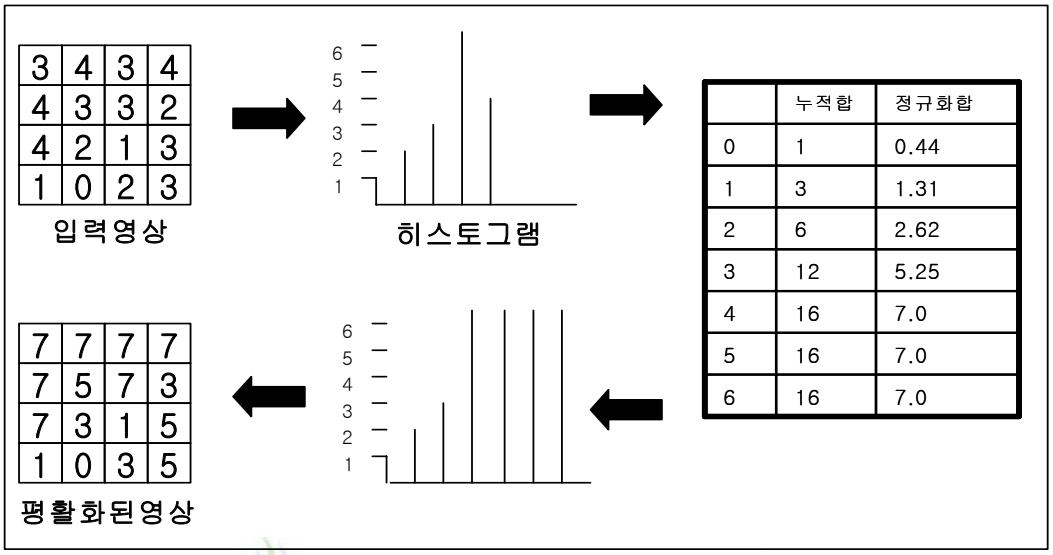


Fig.6 Step of Histogram equalization

식(5)에서 $H(i)$는 원본 입력영상의 누적 히스토그램이고 $h(i)$는 정규 화합 히스토그램 이다. $G_{\max}$는 영상의 최대 밝기값이므로 일반적인 흑백영상에서 255이고 N_t는 입력영상 내부에 존재하는 픽셀의 개수이다. 만일 입력영상이 가로 크기가 200이고 세로가 100이라면 200 × 100 = 20이 된다. 물로 i값이 범위는 0 ~ 255이다. 이 $h(i)$ 값 을 이용하여 입력영상을 변형하면 히스토그램 평활화된 영상이 출력되게 된다.

7 배경 영상 갱신을 이용한 움직임 검출 기법

동영상에서 움직임을 검출하는 방법으로는 프레임간의 차이법과 배경 차이법으로

나눌 수 있다. 앞서 서술한 차영상 움직이 검출 기법이나 혹은 히스토그램을 이용한 움직임 검출 기법은 인접한 두 프레임 간 화소값의 차이를 이용하여 움직임을 검출하는 프레임 차이법으로 볼 수 있다.

이러한 프레임 차이법과는 달리 배경 차이법은 현재 입력 영상과 배경이 되는 영상의 차를 구하여 움직임을 검출하는 방법으로서 인접한 두 프레임을 서로 비교하고 분석하여 움직임을 검출하는 것이 아니라 현재 입력 영상에서 이전 프레임 영상들로부터 배경이 되는 영상을 추정하여 현재 영상의 키 프레임과 배경이 되는 영상의 키 프레임을 비교 분석하여 움직임을 검출하는 방법이다.

이때 비교 대상이 되는 배경 영상 키 프레임은 오래된 이전 영상의 영향을 줄이고, 최근에 입력된 영상의 영향을 추가하여 계속적으로 갱신되고 수정 된다. 이처럼 배경 영상 키 프레임을 계속적으로 갱신하여 움직임을 검출하는 방법으로는 크게 시간적 평활법(Temporal Mean)과 시간적 중간치법(Temporal Median)으로 나눌 수 있다.
시간적 평활법(Temporal Mean)은 배경 영상을 만들때 이전 프레임들의 화소값들을 평균하여 배경의 되는 키 프레임을 만드는 방법이다. 즉 배경 영상의 키 프레임을 만들기 위해서, 현재 입력된 영상의 이전 프레임들의 수가 N개라고 가정하면, N개의 이전 프레임들의 픽셀 화소값들을 모두 더하여 N으로 나누어 배경 영상 키 프레임(Background Key Frame)을 만드는 방법이다. 이 방법은 이전 프레임들의 정보를 기억하기 위해 메모리의 낭비가 심하고, 배경 영상을 만들 때 최근 프레임의 영향과 오래전 프레임의 영향 사이에 비중이 같게 나타나는 문제점이 발생한다.[13]

그래서 시간적 평활화 기법(Temporal Mean)은 다음과 같은 수식(6)을 이용해 화소값이 시간적인 평활화(Temporal Mean) 값을 근사하는 수식을 사용하여 배경 영상의 키 프레임을 만든다.[14][15]

$$m_n = ax_n + (1-a\ )m_{n-1}\,(0 < a < 1) \tag{6}$$

여기서 m_a는 배경 영상의 키 프레임을 나타내고 m_{n-1}은 이전 배경 영상의 키 프레임, a는 근사 영향 비중, x_n은 현재 입력 영상의 키 프레임을 각각 나타낸다.

배경 차이법의 다른 한 방법인 시간적 중간치법(Temporal Median)[5][8]은 임의의

화소에서 이전 프레임에 나타난 값들 중에서 빈도가 높은 값을 배경 영상의 키 프레임으로 사용하는 방법이다.[16]

이러한 방법은 일반적으로 감시 시스템에서 움직임의 발생한 경우, 각 화소에 배경이 나타나는 빈도가 움직임의 발생시 나타나는 빈도수 보다 훨씬 크기 때문에 움직임 검출에 효율적인 방법이 될지 모르지만 이 역시 현재 입력 영상에서부터 이전 영상들을 메모리에 저장해 두었다가 계속적으로 배경이 되는 키 프레임(Background Key Frame)을 갱신해야 하므로 많은 메모리 낭비가 심하고 이전 영상이 많으면 많을수록 처리속도가 느려진다.

Ⅲ. 반복적인 움직임 검출 시스템 설계

본 논문에서는 입력 영상에서 들어온 첫 번째 장면(Scene)에서 키 프레임(Key Frame)을 추출하고 추출된 키 프레임 컬러 영상을 256gray 영상으로 바꾸어 다른 키 프레임과 비교할 배경 키 프레임으로 지정하고 두 번째부터 들어온 프레임을 키 프레임을 설정하여 이를 gray 영상으로 변화시켜 지정된 배경 키 프레임(Background Key Frame)과의 차영상(Difference Image)을 구한다.

이렇게 차영상을 구하고 난 후 블록정합(Block Matching)을 위해 먼저 차영상(Differ ence Image)을 이진화 처리 하고 이진화 된 차영상을 가지고 블록 영역(Block Area)좌표를 추출한다. 그 다음 블록 영역의 (x, y) 픽셀 좌표를 지정된 배경과 현재 키 프레임에 각각 적용시켜 블록 영역을 설정한다.

다음 설정된 두 블록 영역은 각 픽셀 좌표 간 블록정합(Block Matching)시켜 평균절대오차(Mean Absolute Error : MAE)를 계산한다. 이때 계산된 평균절대오차(MAE)가 임계값(Threshold) 보다 작으면 반복적인 물체의 움직임으로 간주하여 침입자가 발생하지 않은 것으로 판단하고 만약에 크면 침입자가 발생한 것으로 간주되어 현재 입력되고 있는 영상은 저장되도록 하였다.

또한 평균절대오차값이 설정한 임계값 보다 크면, 배경 키 프레임을 현재 입력된 영상의 키 프레임으로 갱신하여 메모리에 저장해두고, 만약 작으면 새로운 움직임의 발생한 것으로 판단하여 배경이 되는 키 프레임의 갱신되지 않고 이전 키 프레임의 계속 유지 되도록 하였다.

다음 그림 Fig.7은 본 논문에서 제안한 시스템의 전체 구성도이다.

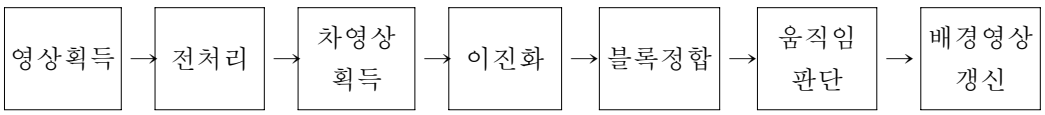


Fig.7 The Proposed flow diagram

1. 전처리 과정

본 논문에서 전처리 과정(Pre-processing)은 입력된 컬러 영상의 키 프레임(Key Frame)을 256 gray레벨의 흑백 영상으로 바꿔주는 작업만 수행하게 된다.
일반적으로 입력 장치에 의해 획득된 영상은 컬러 영상으로써 본 논문에서는 PC에 연결된 화상 카메라에서 RGB 24bit 컬러영상을 입력받는다. 이렇게 입력된 컬러 영상을 256 gray 흑백 영상으로 변환하기 위해서는 다음과 같은 식(7)와 식(8)가 가장 많이 사용된다.

$$g(x,y) = 0.299R(x,y) + 0.587G(x,y) + 0.114B(x,y) \quad (7)$$

$$g(x,y) = 0.333R(x,y) + 0.333G(x,y) + 0.333B(x,y) \quad (8)$$

식(7)와 (8)에서 각각 $g(x,y)$는 256 gray 흑백 영상을 나타내고, $R(x,y)$,$G(x,y)$, $B(x,y)$는 각각 입력받은 RGB 컬러영상의 R, G, B 픽셀값을 나타낸다.

식(7)은 TV 방송국에서 사용하는 컬러 영상을 흑백 영상으로 변환하는 YIQ 컬러 모델로써 명암도를 나타내는 Y값을 RGB 컬러로 표현한 식이다. 즉 만약 RGB 컬러로 TV영상 전파를 쏘아 보낸다면 가정에서 흑백 TV로 시청한다고 할 경우 다시 밝기를 나타내는 명암도 Y값을 계산하여야 한다. 이는 상당히 귀찮은 일이 되고, 또한 소프트웨어나 하드웨어적인 추가 계산이 필요하다. 따라서 방송국에서 미리 RGB 컬러 영상을 명암도와 색상으로 표현되는 YIQ 컬러로 변환시켜 수신기가 흑백 TV인지 혹은 컬러 TV인지에 상관없이 YIQ신호만 쏘아 보낸다면, 가정에서는 수신기가 흑백이라면 식(7)과 같은 Y신호만 취하면 방송을 시청 할 수 있다.[17]

반면에 식(8)은 일반적으로 컬러영상을 256 gray 흑백 영상으로 변환하는 식으로써, 각각의 R, G, B픽셀값을 다 더하여 3으로 나눈 수식을 보여주고 있다. 이 식은 일반적으로 히스토그램 처리, 마스크를 이용한 에지 검출, 영상의 기하학적 변환 처리, 이진 영상 처리,등, 컬러값에 상관없이 처리되는 다양한 이미지 작업에 사용되는 수식이다.

2. 차영상 획득과 이진화 처리

입력 장치를 통해 입력된 컬러 영상을 256 gray 흑백 영상으로 변환 하는 전처리 과정(Pre-processing)을 거친 후에는 배경 키 프레임(Key Frame)과 현재 입력된 영상에서 추출한 키 프레임을 이용해 두 프레임 간에 차영상(Difference Image)을 구해야 한다. 이때 배경 키 프레임(Key Frame)은 영상 입력 장치를 통해 가장 먼저 입력된 영상에서의 키 프레임을 배경 키 프레임으로 설정한다.

차영상(Difference Image)을 획득하는 식(9)와 같다.

$$DI(x,y) = |\, g(x,y) - g_b(x,y) \,|$$

$$DI(x,y) = \begin{cases} g(x,y) & (\ DI(x,y) > t\ ) \\ 0 & (\ DI(x,y) \leq t\ ) \end{cases} \qquad (9)$$

식(9)에서 $g(x,y)$는 현재 입력된 컬러 영상의 256 gray 흑백 영상으로 변환된 후, 키 프레임(Key Frame)을 표현하며 $g_b(x,y)$는 배경의 되는 키 프레임으로 $DI(x,y)$는 현재 입력 영상 키 프레임과 배경 키 프레임간 픽셀들의 차를 통해 획득된 차영상(Difference Image)을 각각 나타낸다. 이때 차영상(Difference Image)의 픽셀값이 지정된 임계값 t 보다 크면, 현재 입력된 영상의 키 프레임의 픽셀값을 할당하고 만약 작거나 같으면 0값을 할당한다. 다음 Fig.8은 두 키 프레임간에 차영상을 구한 결과 그림을 보여 주고 있다.

Fig.8 Background Image, Current Image, Difference Image

이때 검출된 차영상(Difference Image)은 블록 정합(Block Matching)처리를 하기위한 전처리 과정으로 정합 시킬 블록 영역(Block Area)을 설정하기 위해 먼저 영상의 픽셀값을 0과 255로 지정하는 이진화(Binary) 처리를 수행한다. 다음 식(10)과(11)은 각각 차영상을 이진화 처리하기 위해 임계값(Threshold)을 설정하고 처리 하는 수식을 보여주고 있다.

$$T = \sum_{x=0}^{x=N} \sum_{y=0}^{y=M} DI(x,y)/(NM) \tag{10}$$

$$DI(x,y) = \begin{cases} 255 & (\ DI(x,y) > \mathrm{T} \ ) \\ 0 & (\ DI(x,y) \leq \mathrm{T} \ ) \end{cases} \tag{11}$$

위 식(10)과 (11)에서 $DI(x,y)$는 차영상을 나타내고 있고 T는 임계값(Threshold), NM은 차영상의 크기를 각각 나타내고 있다. 이때 임계값 T는 차영상 $DI(x,y)$의 모든 픽 셀값들의 평균을 T로 설정하여 이진화 처리를 하게 된다.

3. 블록 정합

본 논문에서는 먼저 차영상(Difference Image)을 이진화 처리하여 블록 정합(Block Matching)을 적용시킬 블록 영역의 범위(Block Area Regions)를 설정한다. 즉 블록 매칭 알고리즘(BMA)를 적용하여 평균절대오차값(MAE)를 검출할 블록영역의 범위는 움직임의 발생한 영역만을 범위로 설정한다.

블록 영역은 이진화 처리된 차영상들을 행과 열로 한줄씩 차례대로 검색하여 255의 픽셀값을 갖는 픽셀들의 개수를 먼저 구한다. 일반적으로 이진화 처리된 차영상에서 많은 움직임의 발생한 영역은, 255픽셀값의 빈도수가 비교적 많다. 이를 이용하여 255값을 갖는 픽셀들의 빈도수를 행과 열로 한줄씩 검색하여 임계값(Threshold)보다

큰 x, y 좌표값 들을 먼저 구한다. 이렇게 구한 x, y 좌표값들 중 최소, 최대값을 가지는 x, y값들을 각각 $x_{min,}x_{max,}y_{min,}y_{\max}$값으로 지정하고, 이 4개의 값을 이용하여 $(x_{min,}y_{\min}),(x_{min,}y_{\max}),(x_{max,}y_{\min}),(x_{max,}y_{\max})$를 좌표로 하는 사각형 영역을 블록 영역(Block Area)로 설정한다. 이진화 처리된 차영상 에서 $x_{min,}x_{max,}y_{min,}y_{\max}$ 좌표값을 구하는 알고리즘은 다음과 같다.

Step 1 : x축으로 차영상의 픽셀값들의 수를 한줄씩 차례대로 검색한다.

Step 2 : 한줄 검색의 끝나면 255값을 가지는 픽셀의 개수를 구한다.

Step 3 : 255값을 가지는 픽셀들의 개수가 임계값 이상이면 이때 y좌표의 값을 저장한다.

Step 4 : x축 검색이 끝나면 저장된 y좌표값들 중에서 최소값을 $y_{\min}$으로, 최대값을 $y_{\max}$값으로 각각 지정한다.

Step 5 : $y_{\min}$값과 $y_{\max}$값을 구한후 step1부터 step4까지 y축으로 똑같은 방법을 적용하여 $x_{\min}, x_{\max}$ 값을 각각 구한다.

다음 Fig.9는 $x_{min,}x_{max,}y_{min,}y_{\min}$ 좌표를 가지고 블록 영역을 설정하는 모습을 보여주고 있다.

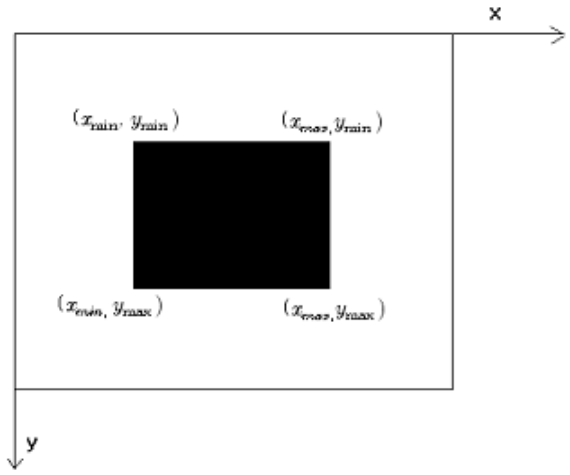


Fig.9 Block Area

이처럼 블록 영역을 설정한 후에는 실제 설정된 블록 영역의 좌표를 실제 블록 정합(Block Matching)시킬 프레임에 대입해야 한다.

즉 $(x_{min,}y_{\min}),(x_{max,}y_{\min}),(x_{min,}y_{\max}),(x_{max,}y_{\max})$ 좌표를 메모리에 저장해둔 배경 키 프레임(Background key frame)과 현재 영상 키 프레임(Current key frame)에 각각 설정된 4개의 좌표값을 대입시켜 실제 블록 정합(Block Matching)시킬 두개의 블록을 얻는다.

Fig.10은 이진화 처리된 차영상에서 $(x_{min,}y_{\min}),(x_{min,}y_{\max}),(x_{max,}y_{\min}),(x_{max,}y_{\max})$ 좌표를 구하여 이를 기준으로 하는 최대 사각형 영역을 설정하여 블록 영역을 검출한 결과 영상을 보여주고 있다.

Fig.10 Block Area Detect

블록 영역(Block Area)의 검출되면, 블록 영역의 x, y 좌표들를 배경 키 프레임(Key Frame)과 현재 영상 키 프레임에 적용하여 실제 블록 정합(Block Matching)시킬 배경 블록과 현재 블록을 얻어온다. Fig.11은 실제 블록 영역 좌표를 배경과 현재 영상키 프레임에 각각 대입하여 배경 블록과 현재 블록을 얻어온 모습을 보여주고 있다.

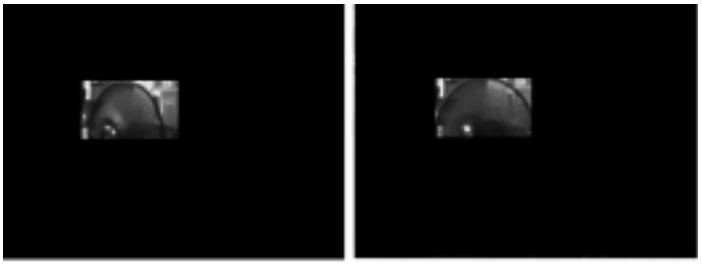


Fig.11 The Result of Block Area Coordinate

이때 지정된 두개의 블록을 식(12)를 이용해 블록정합(Block Matching)시켜 평균절대오차(mean absolute error : MAE)를 계산하고, 계산된 값을 가지고 물체의 반복적인 움직임인지 아니면 침입자가 발생한 움직임인지를 구분하게 된다.

다음 식(12)에서 NM은 블록 영역의 크기를 나타내고 $f_t(n,m)$은 배경 블록을 $f_{t-1}(n-d_{1,}m-d_2)$은 현재 블록을 각각 나타내고 있다.

$$MAE(d_{1,}d_2) = \frac{1}{NM}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}| f_t(n,m) - f_{t-1}(n-d_{1,}m-d_2) \quad (12)$$

실제 물체의 반복적인 움직임인 경우에는 배경 키 프레임과 현재 영상 키 프레임간, 픽셀값의 분포도가 비슷함으로 인해 계산되는 평균절대오차(MAE)값은 작고, 반대로 침입자가 발생한 영상에서는 배경 키 프레임과 현재 영상 키 프레임간, 픽셀값의 분포도가 크게 차이가 나기 때문에 계산되는 평균절대오차(MAE)값은 커지게 된다. 따라서 계산된 평균절대오차(MAE)값이 특정 임계값(Threshold)보다 적으면, 이때 입력된 영상은 침입자가 아닌 물체의 반복적인 움직임으로 검출하고, 반대로 크면 이는 침입자가 발생한 영상으로 판단하고 입력되는 영상들은 감시 시스템에 저장되기 시작한다.

4 배경 키 프레임 갱신

앞서 배경 블록과 현재 블록 간 블록정합(Block Matching)을 시키고, 그때 계산된 평균절대오차(MAE)값을 분석하여 물체의 반복적인 움직임의 발생한 영상과 침입자가 발생한 영상을 구분 할 수 있도록 하였다. 이때 물체의 반복적인 움직임의 발생한 영상인 경우에는 배경 키 프레임(Background Key Frame)을 현재 입력 영상 키 프레임(Key Frame)으로 갱신한다. 다음 식(13)은 물체의 반복적인 움직임인지 판단하여 배경 키 프레임(Background Key Frame)을 갱신하는 수식을 보여주고 있다. 여기

서 $g(x,y)$는 현재 입력 영상 키 프레임이고, $g_b(x,y)$는 배경 키 프레임을 각각 나타내고 있으며, t값은 평균절대오차(MAE)값과 비교할 임계값을 나타내고 있다.

$$g_b(x,y) = \begin{cases} g\ (x,y) & (\ \mathrm{MAE} < \mathrm{t}\ ) \\ g_b(x,y) & (\ \mathrm{MAE} \geq \mathrm{t}) \end{cases} \tag{13}$$

즉 계산된 평균절대오차(MAE)값의 시스템에 설정한 임계값(Threshold : t)보다 작은 경우에만 최근에 입력 영상의 키 프레임으로 배경 키 프레임의 갱신된다.

Ⅳ. 구현 및 실험 결과

본 논문에서 제안한 반복적인 움직임 검출 기법은 침입자가 발생한 영상과 어떤 물 체가 주변 환경에 인해 자연적으로 움직임의 발생한 영상(반복적인 움직임)을 구분 하여 침입자가 발생한 영상만 감시 시스템에 저장 될 수 있도록 하는데 목적이 있다.

이 장에서는 실제 3장에서 제안한 반복적인 움직임 검출 기법의 효용성을 보이기 위 해 제안한 알고리즘을 구현하여 실제 가장 많이 구현되어 사용되고 있는 차영상 움 직임 검출 기법과 비교하여 실험을 하고 그 결과를 비교 분석 하였다.

1. 실험 환경

본 논문에서 구현한 반복적인 움직임 검출 시스템은 크게, 영상획득, 전처리, 움직임 블록정합 및 평균절대오차(Mean-absolute-error)계산, 움직임 검출 및 배경 키 프레임 갱신 부분으로 구성 된다.

이때 영상획득은 USB(Universal Serial Bus : 범용 직렬버스)포트에 연결 가능한 PC 화상 카메라를 시스템에 연결시켜, 화상 카메라를 고정시키고 입력된 24bit RGB 컬러 영상을 사용하여 처리하였다.

본 논문에서의 실험 환경은 CPU 1.7Ghz, 메모리 256MB를 가지는 펜티엄 4 PC와 USB 화상 카메라를 가지고 Windows XP OS 환경에서 실험을 하였다. 실험을 위한 시스템 구현 프로그래밍 언어로는 Visual C++ 6.0과 Direct X 8.0 SDK를 사용하여 구현 하였다.

Table. 1은 반복적인 움직임 검출 시스템의 실험에 사용된 실험 환경과 영상에 대하여 요약한 것이다.

Table. 1 Simulation Environments

시스템 사양	Pentium Ⅳ processor, 256MB RAM
운영체제	Windows XP
프로그램 언어	Visual C++ 6.0, Direct X 8.0 S아
입력영상 파일 포멧	RGB 24bit 컬러 영상
입력영상의 해상도	320 × 240

2. 영상 획득 및 전처리

본 논문에서는 움직임의 있는 영상에서의 움직임 검출을 실험하기 위해 카메라를 통해 입력된 320×240 크기의 RGB 컬러 영상을 식(14)을 이용해 256gray 레벨의 흑백 영상으로 변환하여 ,다음과 같은 3가지 종류의 그룹으로 나눈 입력 영상들을 가지고 10번에 걸쳐 실험 하였다.

$$g(x,y) = 0.333R(x,y) + 0.333G(x,y) + 0.333B(x,y) \quad (14)$$

① 입력 영상 I

- 침입자가 발생한 영상

② 입력 영상Ⅱ

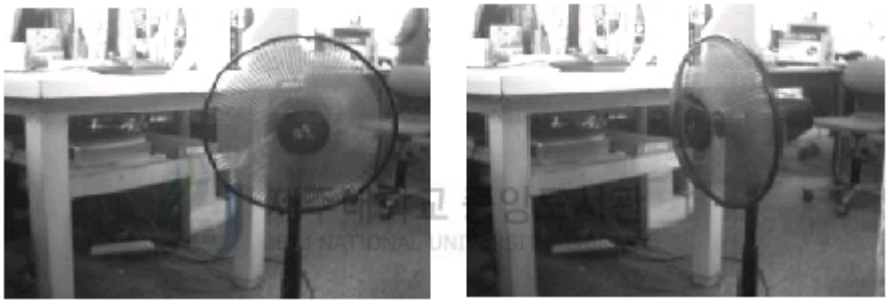
- 상하, 좌우로 반복 움직임 영상

③ 입력 영상Ⅲ

- 불규칙적인 반복 움직임 영상

(a) Background Key Frame (b) Current Key Frame

Fig.12 Input Image Ⅰ



(a) Background Key Frame (b) Current Key Frame

Fig.13 Input Image Ⅱ

(a) Background Key Frame (b) Current Key Frame

Fig.14 Input Image Ⅲ

그림에서 보면 Fig.12는 침입자가 실제 발생하여 감시 시스템에 저장되어야 할 영상의 배경 프레임과 현재 프레임을 각각 보여주고 있고, Fig.13은 침입자가 발생하지 않고, 선풍기가 좌우로 회전하는 움직임만 발생한 영상이다. 마지막으로 Fig.14는 바람에 의해 종이가 날리면서 움직임의 발생한 영상으로 Fig.13처럼 실제 침입자가 발생하지 않은 경우의 예를 보여주고 있다. 즉 실제 침입자가 발생한 경우는 Fig.12인 경우만 해당되고, 나머지 경우에는 침입자가 발생하지 않고 단지 물체가 특정 영역에서 반복적으로 움직이는 반복적인 영상의 예를 보여주고 있다. 따라서 실제 검출되고 저장되어야 할 영상은 Fig.12와 같이 침입자가 발생한 영상만이 실제 움직임 검출이 이루어지고 저장되어야 한다.

3. 이진화 처리

다음 Fig.15는 각각 앞에서 제시한 실험 입력영상Ⅰ, 입력영상 Ⅱ, 입력영상Ⅲ에 대한 차영상(Difference Image)을 구한 결과 영상과, Fig.16은 구한 차영상을 이진화 처리한 결과 영상을 각각 보여주고 있다.

Fig.15 The Result of Difference Image

Fig.16 The Result of Binary Image

여기서 차영상 움직임 검출 기법인 경우에는 Fig.15처럼 차영상만을 구하고 이 차영상을 시스템에서 설정한 임계값과 비교하여 움직임을 검출 하게 된다. 하지만 본 논문에서는 블록정합(Block Matching)을 시켜 평균절대오차(MAE)를 계산하기 위한 전처리 과정으로 차영상을 구한 후 다시 한번 영상의 픽셀값을 0과 255로 설정하는 이진화 처리를 하게 된다.

4. 블록 영역 검출

이진화 처리된 영상은 실제 블록 정합을 시키기 위해 현재 입력 영상 키 프레임과 배경 키 프레임에 대입할 블록 영역을 설정해야 한다. 이때 블록 영역은 이진화 처리된 영상에서 행과 열로 한줄씩 검색하여 255값을 가지는 픽셀들의 계수가 최대, 최소인 x, y값들을 구하고, $(x_{min,}y_{\min}),(x_{max,}y_{\min}),(x_{min,}y_{\max}),(x_{max,}y_{\max})$ 4개의 좌표값을 블록영역의 좌표값으로 설정하여 이를 메모리에 저장해둔 배경 키 프레임(Back ground key frame)과 현재 영상 키 프레임(Current key frame)에 각각 대입시켜 실제 블록 정합(Block Matching)시킬 두개의 블록을 얻는다.

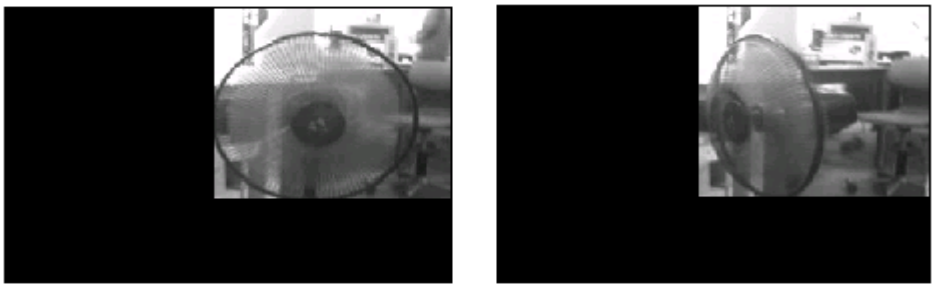
다음은 실험 입력영상Ⅰ, 입력영상Ⅱ, 입력영상Ⅲ에 대해 산출한 블록 영역 좌표(Table.2)와 이 좌표들을 실제 배경 키 프레임(Background Key Frame)과 현재 입력영상 키 프레임(Current Key Frame)에 적용시켜 블록 정합(block matching) 시킬 배경 블록과 현재 블록을 얻어온 화면을 각각 보여주고 있다.(Fig.17,Fig18,Fig19)

Table.2 Block Area Position

	x_{min}	x_{max}	y_{min}	y_{max}
입력영상 I	158	210	97	239
입력영상 II	157	294	22	166
입력영상 III	110	185	27	185

(a) Background Block Area (b) Current Block Area

Fig.17 Input Image I



(a) Background Block Area (b) Current Block Area

Fig.18 Input Image II

(a) Background Block Area (b) Current Block Area

Fig.19 Input ImageⅢ

5. 평균 절대 오차 계산

일반적으로 현재 영상 키 프레임(Current Key Frame)과 배경 키 프레임(Back ground Key Frame)에서 움직임 물체를 검출하는 방법은 현재 영상 키 프레임에서 움직임 물체의 블록과 가장 유사한 블록을 배경 키 프레임에서 검색 하는 방법이다. 본 논문에서는 블록 정합시 두 블록 영역 간 (배경 블록과 현재 블록) 매칭의 척도가 되는 함수들 중 평균절대오차(Mean Absolute Error : MAE)을 사용하여 실험하였다. 다음 식 (15)는 평균절대오차(MAE)을 구하는 식을 보여주고 있다.

$$MAE(d_{1,}d_2) = \frac{1}{NM}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} | f_t(n,m) - f_{t-1}(n-d_{1,}m-d_2) \qquad (15)$$

위 식에서 f_t는 현재 영상 키 프레임(Current Key Frame)이고 f_{t-1}은 배경 키 프레임(Background Key Frame)이며 NM은 블록영역(block area)의 크기를 각각 나타내고 있다. 다음 Table 3은 각 입력 영상들의 현재 입력 영상 키 프레임과 배경 영상 키 프레임들을 전처리 과정(Pre-processing)만을 처리한 후 평균절대오차(MAE)값를

계산한 결과와 차영상을 구하고 이진화 처리후 블록 영역(Block Area)을 구하고, 이 블록 영역 좌표를 현재 영상 키 프레임과 배경 키 프레임에 대입하여 추출된 현재 블록과 배경 블록을 가지고 평균절대오차(MAE)값를 계산하여 실험한 결과이다.

Table.3 Mean Absolute Error

	전처리 과정 후		블록 영역 설정 후	
	이미지 크기(N×M)	MAE	이미지 크기(N×M)	MAE
입력영상Ⅰ	320 × 240	6.1	52 × 142	62.1
입력영상Ⅱ	320 × 240	6.2	137 × 144	22.0
입력영상Ⅲ	320 × 240	8.5	75 × 128	21.8

Table 3을 분석해 보면 블록 영역을 설정하지 않고 256gray영상으로만 변환하는 전처리 과정만을 거친 후 평균절대오차(MAE)값를 계산해본 결과, 침입자가 발생한 실험 입력 영상Ⅰ과 반복적인 움직임을 갖는 실험 입력영상Ⅱ와 입력 영상Ⅲ는 그 값이 유사하게 나타남을 알 수 있다. 반면에 블록 영역(Block Area)을 설정한 후 계산된 평균절대오차(MAE)값은 침입자가 발생한 입력 영상Ⅰ인 경우 반복적인 움직임을 가지는 입력 영상 Ⅱ나 입력영상 Ⅲ보다 상당히 커짐을 확인 할 수 있다. 따라서 이렇게 블록 영역을 설정한 후 계산된 평균절대오차(MAE)값을 시스템에서 설정한 임계값 30과 비교하여 임계값 30보다 크면 침입자가 발생한 영상이고, 그렇지 않으면 어떤 물체가 주변 환경에 의해 자연적으로 움직인, 반복적인 움직임의 발생한 영상으로 구분하여 인식 할 수 있었다.

6. 배경 키 프레임 갱신

배경 키 프레임(Backgroud Key Frame)은 현재 입력 영상 키 프레임(Current Key Frame) 과 함께 평균절대오차(MAE)값를 구하는데 기준이 되는 중요한 요소이다. 만약 물체의 반복적인 움직임의 발생한 영상인 경우 배경 키 프레임을 갱신하지 않으면, 계산되는 평균절대오차(MAE)값은 점점 더 커지게 된다. 따라서 입력영상Ⅱ, Ⅲ과 같은 물체의 반복적인 움직임이 발생한 영상인 경우에는 배경 키 프레임을 갱신해야 할 필요가 있다. Fig.20과 Fig.21은 앞선 입력영상Ⅱ의 선풍기의 회전처럼 바람에 의해 문이 열리는 반복적인 움직임이 발생한 영상을 가지고 배경 키 프레임을 갱신 할 때와 그렇지 않을 때를 각각 보여주고 있다.

	1번째 비교 영상	2번째 비교 영상	3번째 비교 영상
배경 키 프레임			
현재 입력 영상 키 프레임			

Fig.20 When fix background key frame

	1번째 비교 영상	2번째 비교 영상	3번째 비교 영상
배경 키 프레임			
현재 입력 영상 키 프레임			

Fig.21 When update backbround key frame

실제 다음 Table.4는 앞선 영상들(Fig.20, Fig.21)에 대한 평균절대오차(Mean-absol ute -error)값를 계산한 결과이다.

Table.4 Mean-absolute-error change by state of background key frame

	1번째	2번째	3번째
배경 키 프레임의 고정될때 MAE : Fig.20	5.0	9.0	13.0
배경 키 프레임의 갱신될때 MAE : Fig.21	5.0	6.0	4.5

Table.4를 살펴보면 배경 키 프레임(Bsckground Key Frame)의 고정된 경우(Fig.20)에는 시간이 지날수록 입력되는 영상과의 평균절대오차(MAE)값이 변화가 커지는 반면에 배경 키 프레임의 현재 입력 영상의 직전 프레임으로 갱신되는 경우(Fig.21)에는 평균절대오차(MAE)값이 변화가 적음을 실험을 통해 확인 할 수 있었다. 배경 키 프레임(Background Key Frame)의 갱신은 다음 식(15)를 이용하여 계산된 평균절대오차(MAE)값이 시스템에서 설정한 임계값 30보다 작으면, 현재 입력 영

상으로 갱신한다.

$$g_{b+1}(x,y) = \begin{cases} g\ (x,y) & (\ \mathrm{MAE} < \mathrm{t}\ ) \\ g_{b+0}(x,y) & (\ \mathrm{MAE} \geq \mathrm{t}\ ) \end{cases} \quad (16)$$

식 (16)에서 $g_{b+1}(x,y)$은 현재 갱신될 배경 키 프레임(key frame)이고, $g_{b+0}(x,y)$는 이전의 배경 키 프레임(key frame), 그리고 $g\ (x,y)$는 현재 입력된 영상의 키 프레임(key frame), t는 임계값을 각각 나타내고 있다.

7. 실험 결과 분석

다음 Fig.22은 본 논문에서 구현한 전체 시스템의 모습을 보여주고 있으며 Table.5는 본 논문에서 제시한 알고리즘을 앞서 설명한 3가지의 서로 다른 움직임의 발생한 입력 영상들을 가지고 각각 모의 실험을 해본 결과를 차영상 움직임 검출 기법과 비교하여 보여주고 있다.

Fig.221 The Captured Screen of the System

Table.5 The Result

분 류 / 영상 환경	차 영상 기법				제안 알고리즘			
	평균 차 픽셀 계수	평균 MAE	검출 오류 ((F/A)*100%)	평균처리 시간(sec)	평균 차 픽셀 계수	평균 MAE	검출 오류 ((F/A)*100%)	평균처리 시간(sec)
입력 영상 Ⅰ	12002.6	3.8	6.2%	0.194	4282	67.6	17.3%	0.465
입력 영상 Ⅱ	20030	3.7	77.66%		2122	13.6	10.0%	
입력 영상 Ⅲ	12380	9.8	91.2%		8524	16.5	13.3%	

위 Table 5에서 평균 차 픽셀 계수는 배경 키 프레임(Background Key Frame)과 현재 영상 키 프레임(Current Key Frame)간 차영상(Difference Image)에 의해 검출된 픽셀들의 계수이고, 평균절대오차(MAE)값은 배경 키 프레임과 현재 영상 키 프레임에 각각 블록 영역좌표를 구하여 설정된 배경 영역과 현재 영역간 블록정합(Block Matching)시켜 계산된 평균절대오차(MAE)값를 나타낸다. 또한 검출오류는 현재 입력된 움직임의 발생한 총 영상 프레임의 계수(A)를 움직임의 검출된 영상의 프레임 계수(F)로 나누어 백분율로 환산하여 계산된 값이다.

단 입력 영상Ⅰ인 경우에는 움직임의 발생하지 않는 영상을 F값으로 설정하여 검출오류를 계산하였다. 그리고 계산된 평균처리시간은 계속적인 움직임의 발생한 60초 입력 영상에서 움직임의 검출된 프레임의 개수를 가지고 평균 처리 시간을 얻었다. 차영상 움직임 검출 기법인 경우에는 60초 동안 평균 309개의 움직임 영상을 얻을 수 있었고 본 논문에서 제안한 기법인 경우에는 평균 129개의 움직임 영상을 얻을 수 있었다.

위 Table.5 로부터 움직임 검출 오류를 측정한 결과 실험 입력 영상Ⅰ인 경우에대

해, 차영상 움직임 검출 기법에서는 6.2%, 제안한 알고리즘에서는 17.3%이며, 입력 영상Ⅱ에서는 각각 77.66%, 10.0% 이 고 입력 영상Ⅲ에서는 91.2%, 13.3%로 나타났으며, 처리속도는 차영상 움직임 검출 기법이 제안한 알고리즘 보다 약 2.4배 빠르게 검출됨을 확인 할 수 있었다.

또한 실험 입력 영상Ⅱ나 입력 영상Ⅲ의 환경처럼 물체의 반복적인 움직임이 있는 영상에서는 제안한 움직임 검출 기법이 차영상 움직임 검출 기법보다 상당히 좋은 결과를 얻을 수 있었으나 입력영상Ⅰ처럼 침입자가 발생한 영상인 경우에는 좋지 않은 결과를 보였다.

Ⅴ. 결 론

감시 시스템에서 움직임 검출 기법은 상당히 중요한 비중을 차지하는 부분으로, 얼마나 정확하게 움직임의 발생하였는지 검출하여 저장하는 것은 감시 시스템의 성능을 좌우하는 중요한 요소들 중 하나이다.

이러한 감시 시스템에서 움직임을 검출하는 방법에는 가장 빠른 처리 속도를 가지는 차영상 움직임 검출 기법(The Motion Detection Using Difference Image)이 가장 많이 사용되고 있으나, 이 움직임 검출 기법은 배경의 명암도의 변화나 잡음 및 반복적인 물체의 움직임에 대해서는 많은 문제점이 발생한다.

본 논문에서는 이렇게 차영상 움직임 검출 기법에서 문제점으로 대두되고 있는, 반복적인 움직임의 발생하였을 경우에 대해, 그 해결 방법을 제안하고 있다.
차영상(Difference Image)을 통해 움직임의 검출 되었을 때, 이 움직임의 물체의 반복적인 움직임 영상인지, 아니면 침입자가 발생한 영상인지를 구분하기 위해 본 논문에서는 물체의 반복적인 움직임인 경우에 현재 입력 영상 키 프레임(Current Key Frame)과 배경 영상 키 프레임(Background Key Frame)과의 움직임의 검출된 영역간 픽셀값의 분포도가 매우 유사한 반면에 침입자가 발생한 영상인 경우에는 두 블록간 픽셀값의 분포도가 매우 차이가 크다는 특징을 이용하였다.

이러한 특징을 적용하기 위해 먼저 현재 입력 영상 키 프레임과 배경 영상 키 프레임과의 차영상(Difference Image)구하고, 블록 영역의 좌표를 구하기 위해 차영상(Difference Image)을 이진화 처리(Binary) 하여 움직임의 검출된 영역만을 블록 영역(Block Area)의 좌표로 지정 하였다.

이렇게 지정된 블록 영역(Block Area)좌표을 현재 입력 영상 키 프레임과 배경 영상 키 프레임에 대입하여 현재 블록과 배경 블록을 얻고, 이 두 블록을 정합시켜 평균절대오차(Mean-absolute-error : MAE)값를 계산하고, 계산된 평균절대오차값의 시스템에서 설정된 임계값 보다 크면 침입자가 발생한 것으로 반대로 작으면 물체가 주변 환경에 의해 자동으로 움직인 반복적인 움직임 영상으로 인식 되도록 하였다.
실험 결과 본 논문에서 제시한 방법으로 움직임을 검출 하였을 때 물체의 반복적인

움직임의 발생한 영상에서는 차영상 움직임 검출 기법(The Motion Detection Using Difference Image)과 비교하여 약 77%의 성능 개선을 보였으나, 침입자가 발생한 영상인 경우에는 오히려 차영상 움직임 검출 기법(The Motion Detection Using Difference Image)보다는 약 11%의 성능의 떨어짐을 확인 할 수 있었다.

향후 좀더 본 논문에서 제안한 시스템을 개선하여 침입자가 발생한 영상인 경우에 있어서 움직임 검출 처리 속도와 성능을 향상시킨다면 실제 무인 감시 시스템에 적용하여 감시 시스템의 성능을 향상 시킬 수 있는 결과를 얻을 수 있을 것으로 본다.

[참고문헌]

1. 이규원, 김영호, 이재규, 박규태 “무인 감시 장치 구현을 위한 단일 이동물체 추적 알고리즘”, 전자 공학회 논문지, 제31권 B편, 제11호, Startpage 11, 1995
2. 김혜경, 남시병 “2D FFT를 이용한 움직임 검출에 관한 연구” 삼척대학교 논문지, Vol.35, No.1, Startpage 159, 2002
3. 하영현, 채옥삼, “차영상의 차 히스토그램을 이용한 자동 임계값 결정”, 신호 처리 종합 학술 대회 논문집 제 11권 1호, 1998
4. 김용훈, 이태홍, 백지흠, 조영창, 김성욱 “시간적 상관성을 이용한 움직임 추정” 한국멀티미디어 추계학술발표 논문집, Vol.2, No.2, Startpage 799, 1999
5. 조태훈,최영규 “다중 배경 분포를 이용한 움직임 검출” 한국정보처리학회 논문집 ,Vol.8, No.4, Startpage 381, 2001
6. 조영식, 이주신 “부분 외곽선 정보를 이용한 이동물체의 추적 알고리즘“, 정보처리학회 논문지, Vol.8, No.5, Startpage 539, 2001
7. 조영창, 이태홍 “움직임 영역간 보상오차의 최소편차를 이용한 최적 블록정합 움직임 추정”, 정보처리학회논문지, Vol.8, No.5, Startpage 557, 2001
8. 전춘, 김태식, 이면길, 이주신 “동적 배경에서 Hausdorff 거리를 이용한 이동물체의 추적”, 한국통신학회 하계종합학술대회논문집, Vol .19, No.1, Startpage 537, 1999
9. N.McFarlance, "Segmentation and Trackig of Piglets in Images,"Machine Vision Application, Vol.8,PP. 187-193,1995
10. C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, “Pfinder : Real-Time Tracking of the Human Body." IEEE Trans. Pattern Analysis and Machine Intelligence. Vol.9, No. 7, 1997
11. Y. Ivanov, A. Bovick, an J.Liu, "Fast Lighting Independent Background Subtraction," Technical Report No.437, MIT Media Lab. 1997
12. Willian B. Thompson, Pamela Lechleider, and Elizabeth R. Stuck, "Detecting moving objects using the rigidity constranint." IEEE Transaction on pattern Analysis and Machine Intelligence, PAMI-15, NO.2, pp.162-169, 1993.
13. Vincent S. S Hwang, "Tracking feature points in time-varying images using

an opportunitic selection approach, "Pattern Recgnition Vol.22, No.3, pp.247-256, 1989

14. W. K. Chow and J. K. Aggarwal, "Computer analysis of planarcurvilinear moving images," IEEE Transaction Computer, C-26, pp.179-185, 1977.
15. Hsi Jian, Lung Fa Huang, and Z. Chen, "Multifrane ship dectectin and tracking in a infrared image sequence," Pattern Recognition, Vol.23, No.7, pp.785-798, 1990..
16. A. Rognone M. Campani, and A. Veri,"Detecting moving objects from optical Flow," Pattern Recognition and Image Analysis, vol.2, No.1, pp.109-111, 1991
17. R. Jain, D. Militer, and H. H. Nagel, "Separating non-stationary from stationary scene components in a sequence of real world TV-image," Proc 5th Int, Joint Conf Artificial Intelligence, pp.612-618, 1977

감사의 글

새로운 마음으로 힘차게 시작했던 2년간의 대학원 생활이 벌써 하루 하루 흘러 어는덧 아쉬운 끝으로 다가 왔습니다. 2년 전만 해도 대학원이라는 생활이 참으로 힘들거라는 두려움으로 시작했지만, 생활해 가면서 교수님들과 선배님들 그리고 같이 공부하고 연구하던 동기들이 있어 참으로 힘든 줄 모르고, 열심히 대학원 생활을 할 수 있었습니다.

이제 대학원 생활의 마지막 자리에 서면서, 언제나 관심을 가지고 끝가지 성심 성의껏 지도해 주신 김장형 지도 교수님과, 이렇게 좋은 결과를 맺을 수 있도록 조언해 주신 안기중 교수님, 변상용 교수님, 곽호영 교수님, 이상준 교수님, 송왕철 교수님, 변영철 교수님께 먼저 감사의 마음을 전합니다.

또한 연구실에서 공부하는 동안 언제나 대학원 생활과 연구에 대해 조언을 아끼시지 않으시고, 관심을 가져주신 박사과정 강영도 선생님, 강진석 선생님, 박창희 선생님, 김정효 선생님, 강길봉 선생님, 변태보 선생님, 강명화 선생님께도 진심으로 감사 드립니다.

그리고 같이 연구하고 생활하던 석사과정 홍유기 선생님, 문일남 선생님, 강진영 선생님, 김남식 선생님과 멀티미디어 연구실이 여러 식구들과 대학원에 같이 들어와서 함께 열심히 공부했던 저의 동기들 및 학과 사무실에서 저희들을 위해 애써주신 이정하 조교 선생님께도 감사의 마음을 전하는 바입니다.

2년이란 짧은 대학원 생활동안 이처럼 감사의 마음을 전할 수 있는 멋진 교수님들과 좋은 선배님들 그리고 함께 했던 동기들이 있어 대학원 생활이 저희 기억속에서는 언제나 좋은 기억으로 남을 수 있을 것 같습니다.

끝으로 언제나 가슴 졸이면서 아들의 대학원 생활을 뒷바라지 하고, 뒤에서 항상 따뜻하게 맞아주고, 격려의 말씀을 아끼시지 않으시던 집에 계신 아버님, 어머님에게 마지막으로 진심 어린 감사의 말씀을 드립니다.

마지막으로 언제나 이 모든 분들께 살아가면서 언제나 좋은 일만 있으시길 빌겠습니다. 감사합니다.