碩士學位論文

채도 정보를 이용한 움직임 검출

濟州大學校 大學院

情 報 工 學 科

孫 周 植

2004年 12月

채도 정보를 이용한 움직임 검출

指導敎授　金　壯　亨

孫　周　植

이 論文을 工學 碩士學位 論文으로 提出함.

2004年　12月

孫周植의 工學 碩士學位 論文을 認准함.

審査委員長	安　基　中	印
委　　員	金　壯　亨	印
委　　員	郭　鎬　榮	印

濟州大學校 大學院

2004年 12月

Motion detection using information of saturation

Ju-Sik Son
(Supervised by professor Jang-Hyung Kim)

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

DEPARTMENT OF INFORMATION ENGINEERING
GRADUATE SCHOOL
CHEJU NATIONAL UNIVERSITY

2004. 12.

목 차

Summary v

Ⅰ. 서 론 1

Ⅱ. 움직임 검출 기법의 개요 3

1. 움직임 검출 기법의 필요성 3
2. 움직임 검출 기법의 개요 4
3. 동영상에서의 움직임 검출 기법 5
4. 블록 정합 알고리즘 7
   1) 전역 탐색 알고리즘 9
   2) 고속 탐색 알고리즘 9
5. 차영상 움직임 검출 기법 10
6. 히스토그램을 이용한 움직임 검출 기법 13
7. 배경 영상 갱신을 이용한 움직임 검출 기법 15

Ⅲ. 채도 정보를 이용한 움직임 검출 시스템 설계 17

1. 전처리 과정 21
2. 차영상 획득 및 배경영상 갱신 22
3. 움직임 검출 및 입력 영상 저장 23

Ⅳ. 구현 및 실험 결과 24

1. 실험 환경 24
2. 적용된 알고리즘 25
3. 시스템 구현 27
4. 실험 결과 29
1) 카메라 환경설정을 통한 밝기 값 변화 실험 29
2) 실내조명의 밝기를 서서히 변화시키며 실험 30
3) 임의의 잡음을 유입시키며 실험 31
4) 검출 속도를 비교하는 실험 32

Ⅴ. 결 론 34

참고문헌 35

[그림 목차]

Fig. 1 A Distribution Diagram of Moving Detection Algorithm ····· 5
Fig. 2 A Moving Picture Hierarchy ····· 6
Fig. 3 Block Matching Search Area ····· 8
Fig. 4 Search Algorithm ····· 10
Fig. 5 Flow Diagram that The Motion Detection Using Difference Image ····· 12
Fig. 6 Step of Histogram equalization ····· 15
Fig. 7 A bright room ····· 18
Fig. 8 Medium light ····· 18
Fig. 9 A dark room ····· 18
Fig. 10 Including noise ····· 19
Fig. 11 Motion Estimation by Block matching algorithm ····· 19
Fig. 12 The Structure of Proposed algorithm ····· 21
Fig. 13 The Flow Diagram of Proposed algorithm ····· 26
Fig. 14 Shape of the System ····· 27
Fig. 15 The Captured Images when Motion detect ····· 28
Fig. 16 The Setting of Camera ····· 29
Fig. 17 Shape of another System ····· 32

[표 목차]

Table 1. Comparison of MAD ····· 20
Table 2. Simulation Environment ····· 23
Table 3. Motion detection errors by Camera setting when Brightness increase ····· 30
Table 4. Motion detection errors by Camera setting when Brightness reduce ····· 30
Table 5. Motion detection errors by Brightness Variation of room ··· 31
Table 6. Motion detection errors by Noise ····· 32
Table 7. Number of times to Operate ····· 33

Summary

In this paper, I propose an algorithm to solve wrong recognitions caused by noise or brightness variation when I detect motion.

Among the motion detection method, method using difference image is used very generally. And it occurs many wrong operations caused by brightness variation and noise. Also, algorithm using block matching has powerful operations even if brightness variations.

However, according to the time, it has many wrong operations. Therefore in this paper, I've updated background image in order to powerful operate although noise and brightness variation.

Also I propose a powerful motion detection method when I detect motion, it used information of saturation as threshold. In this study, I have better results for noise and brightness variation according to the time than method using difference image and algorithm using block matching.

Ⅰ. 서 론

최근 인터넷의 발달과 함께 각종 보안 감시 시스템이 출시되고 있다. 이러한 보안 감시 시스템의 주요 목적은 보안 장소에서 침입자를 정확하게 탐지하여 영상을 저장하고 관리자에게 통보하는 기능이다.

이러한 주 기능은 움직임, 즉 침입자 발생시 자동으로 움직임이 검출되도록 함으로써 보안 감시 시스템의 성능을 높일 수 있는데, 이로 인해 이러한 보안 감시 시스템에는 침입자 탐지를 위한 다양한 움직임 검출 기법이 제시되었다.

보안 감시 시스템에서 움직임 검출을 위한 알고리즘에는 크게 차영상을 이용한 움직임 검출 알고리즘, 블록정합을 이용한 움직임 검출 알고리즘, 배경영상 갱신을 이용한 움직임 검출 알고리즘, 히스토그램을 이용한 움직임 검출 알고리즘 등이 있다.

먼저 차영상을 이용한 움직임 검출 기법은 처음 입력된 영상과 현재 입력된 영상들의 픽셀간의 차가 설정한 문턱치값 이상인 경우 움직임이 발생한 것으로 인식하고 검출하는 알고리즘으로써, 움직임 검출 시간이 상당히 빠르다는 장점은 있지만, 입력장치에 유입되는 잡음 및 밝기 변화에는 상당히 민감하게 움직임이 발생한 것으로 인식하는 문제점이 있다.

두 번째, 블록정합을 이용한 움직임 검출인 경우에는 침입자에 대한 움직임 경로 추적 및 검출이 가능하지만 침입자가 수시로 정지하거나, 멈춰있는 경우에는 검출하지 못하는 단점이 있다. 더욱이 밝기 변화에 대해서도 블록정합 움직임 검출 알고리즘은 상당히 민감하게 동작한다.

세 번째, 배경영상을 이용한 움직임 검출 기법인 경우에는 유입되는 잡음이나 밝기 변화에는 어느 정도 강건하게 동작하지만, 움직임 검출을 위해 필요한 연산 작업 및 처리 시간이 길고 시스템의 메모리 낭비가 심하다는 단점이 있다.

마지막으로 히스토그램을 이용한 움직임 검출 기법인 경우에는 픽셀 값의 빈도수로 움직임을 검출하기 때문에 어느 정도 밝기 변화에는 강건하게 동작하지만, 시간이 지날수록 유입되는 잡음에는 오동작하는 문제점을 발생시킨다.

따라서 본 논문에서는 움직임 검출시 가장 오인식하는 빈도수가 많은 유입되는 잡

음 및 밝기 변화에 강건하게 동작할 수 있는 움직임 검출 알고리즘을 제안하고자 한다. 본 논문에서 제안한 움직임 검출 알고리즘은 영상영역으로 유입되는 잡음이 아닌 침입자의 움직임을 보다 정확하게 검출해낼 수 있기 때문에 보안 감시 시스템의 저장 공간 절약 및 침입자 위치 추적과 같은 분야에 적용 가능하다.

또한 입력되는 영상에서 채도 정보만을 추출하여 배경영상과 비교, 분석함으로써 밝기 변화에 강건하고 시간에 따라 변화하는 영상의 밝기나, 내부 조명에 의한 영상 변화에도 강건하게 동작하는 알고리즘을 제안한다.

논문의 구성은 Ⅰ장 서론에서는 본 논문의 연구 동향에 대하여 서술하고, 논문에서 제안한 알고리즘의 필요성을 제시하였으며, Ⅱ장에서는 움직임 검출 기법의 개요와 종류, 원리에 대해 서술하였다. 그리고 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 서술하였다.

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 들려주거나, 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 어려움이 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 이루어져야 한다. 이렇듯 움직임 검출 기법은 무인 영상 감시 시스템이나 영상 압축 기술 및 움직임 추정기법 등에 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 영역화하여 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 특징 선(Line) 벡터를 이용하여 움직임을 검출하는 방법 등이 있다.

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

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

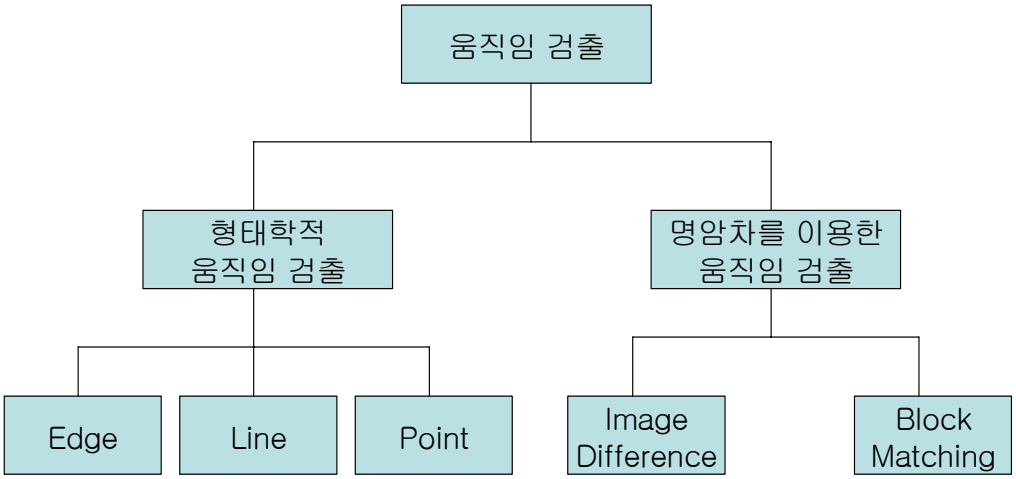


Fig.1 A Distribution Diagram of Moving Detection Algorithm

3. 동영상에서의 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(Scene)들은 다시 여러 개의 프레임들로 나뉘어 진다. 이때 이렇게 장면의 어느 시점을 기준으로 변화하는 부분을 동영

상에서는 장면 전환 지점이라고 한다.

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

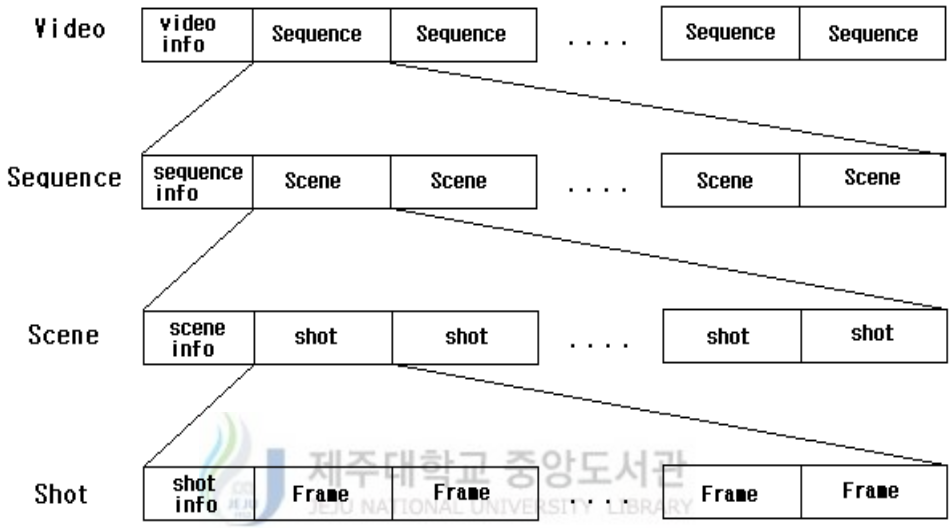


Fig.2 A Moving Picture Hierarchy

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(Scene)은 고도의 내용 정보(Sementic Information)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 전환 지점(Scene Change

)을 찾고, 이 장면 전화 지점의 전, 후 장면(Scene)을 구성하는 샷(Shot)과 이 샷(Shot)을 구성하는 여러 개의 프레임(Frame)들을 찾는다.

이때 이 프레임(Frame)들은 전체적인 구조나 색상 분포 등의 동일한 특성들을 지니게 되는데. 이러한 여러 개의 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 저 전송 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 (Block 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 정합의 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 블록 크기가 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 + 2 d_{max-y})가 된다. 이때 가능한 탐색점의 수는 총 ($2d_{mzx-x}$ + 1) × ($2d_{max-y}$ + 1) 이 된다.

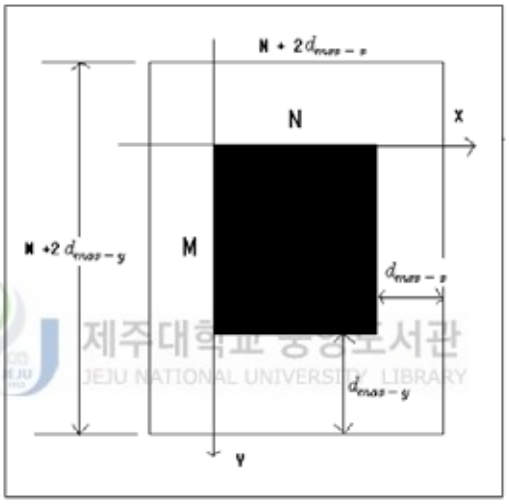


Fig.3 Block Matching Search Area

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 구현이 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.

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

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

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.

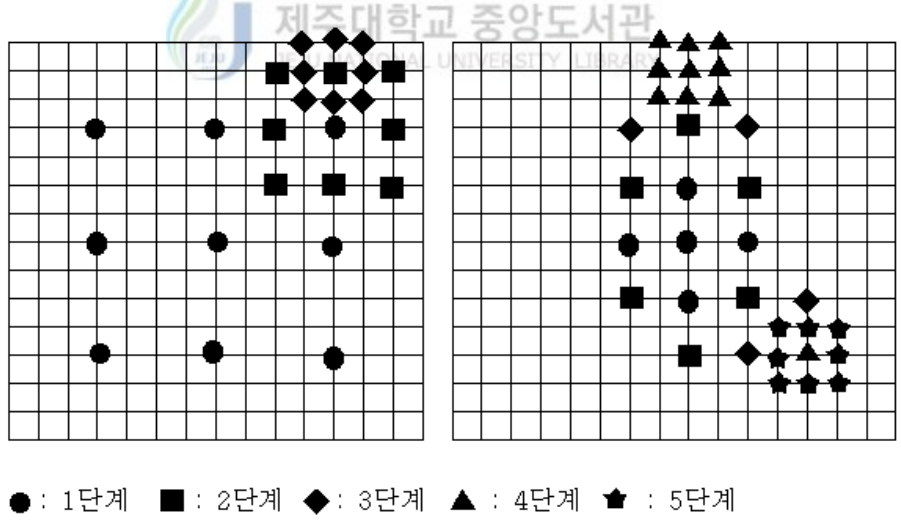
Koga에 의해 제안된 3단계 탐색 알고리즘은 가장 고속 정합이 가능한 알고리즘으로서, 탐색 영역의 반의 크기로 탐색 영역의 중심에서 시작하여 각 단계마다 앞

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 탐색법은 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 픽셀값을 이용하여 차영상(Difference Image)을 구하는 식을 보여주고 있다.

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

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 서

로 비교할 임계값(Threshold)을 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.

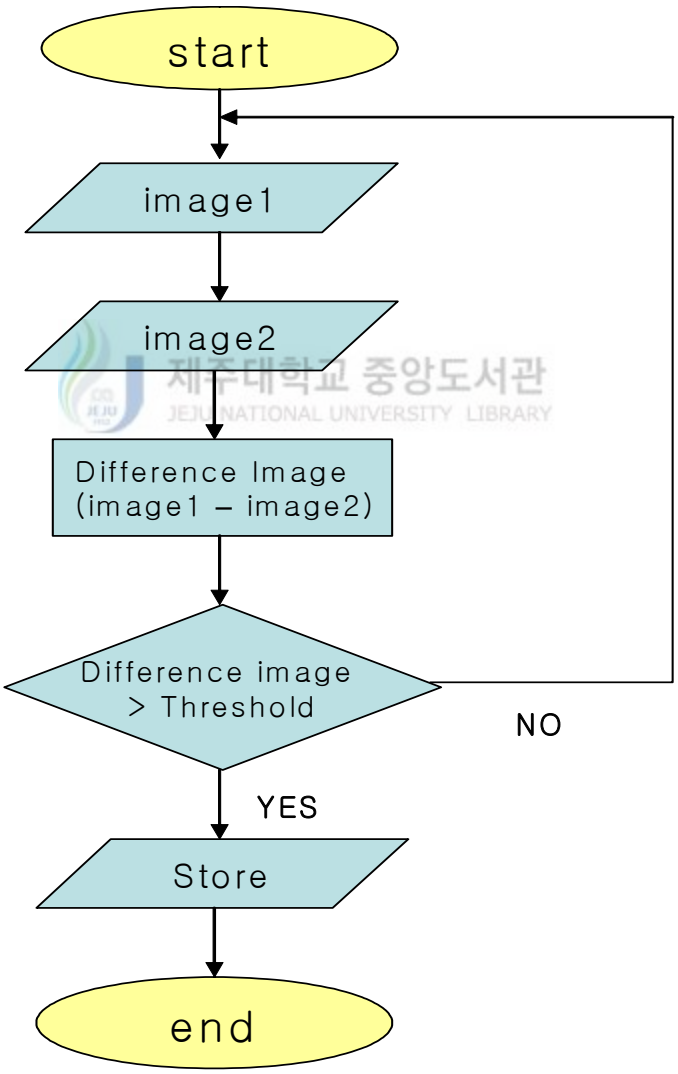


Fig. 5 Flow Diagram that The Motion Detection Using Difference Image

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 발생하면 침입자가 발생한 것으로 인식하는 움직임 검출 방법(Motion Detection) 이다. 다음 식(4)는 히스토그램(Histogram)을 이용하여 차영상을 구하는 식을 보여주고 있다.

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

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 움직임이 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 방법이 제시되고 있으나, 이러한 방법 역시 밝기나 잡음에는 어느 정도 강건하게 움직임이 검출되지만 반복적인 움직임이 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.

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

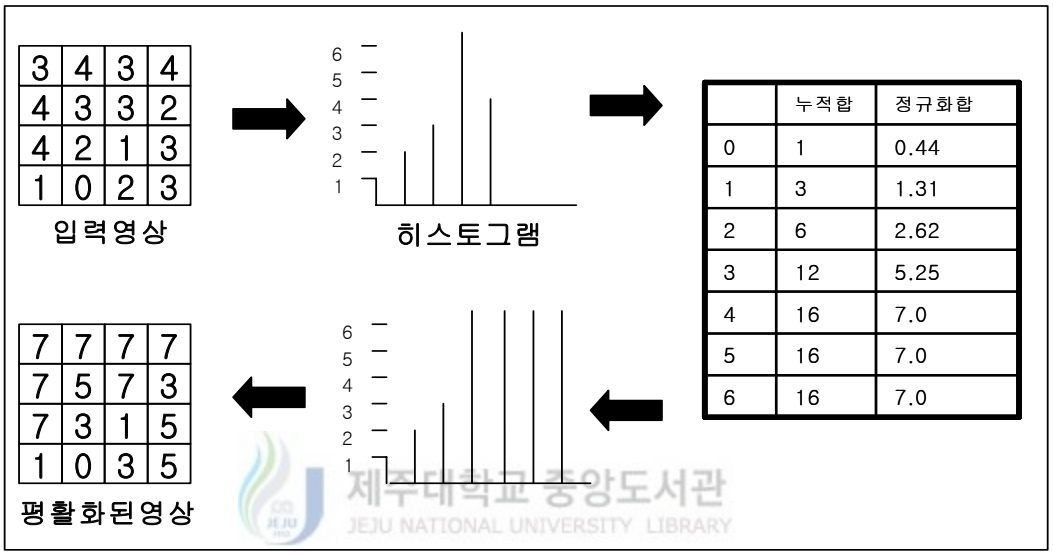


Fig. 6 Step of Histogram equalization

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 = 20,000이 된다. 물론 i값의 범위는 0 ~ 255이다. 이 $h(i)$ 값을 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 움직임 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. 채도 정보를 이용한 움직임 검출 시스템의 설계

본 논문에서는 입력장치로부터 들어오는 첫 번째 입력영상을 배경영상으로 지정한다. 이 때 들어오는 영상은 컬러영상이므로 배경영상으로 지정하기 위한 전처리 단계는 입력되는 RGB 컬러영상을 YUV 컬러영상으로 변환하는 것이다. 일반적으로 밝기 변화가 있는 영상에서는 명암 정보가 가장 많이 변화하며, 잡음이 유입된 영상에서는 색상 정보의 변화가 가장 많이 발생한다. YUV영상으로 변환하는 이유는 밝기 변화와 잡음 유입에 강건하게 동작하게 하기 위해 명암도(luminance)와 색상(hue), 채도(saturation) 정보로 표현되는 색상 모델인 YUV로 RGB 색상을 변환하고, 이 때 명암도 값인 Y와 색상 정보 U를 제외하여 채도 정보만 배경영상으로 지정한다.

이렇게 지정된 초기 배경영상은 명암 정보를 배제하였기 때문에 배경의 밝기 변화에는 상당히 강건하게 동작하지만, 유입되는 잡음에는 민감하게 동작하므로 시간이 지나면서 유입되는 잡음이 많아지면 움직임이 발생하지 않더라도 발생한 것으로 검출하는 오동작이 발생하게 된다.

따라서 이러한 이유로 시간이 지나면서 지속적으로 배경영상이 갱신되어야 하는데, 일반적으로 배경영상을 갱신하기 위해 입력되는 현재 영상과 배경영상의 M×N 블록 내에서 채도 정보를 가지고 차영상을 구하고 구한 차영상을 이진화시킨 후 255픽셀값의 계수가 임계값보다 작으면 움직임 객체가 영상에 들어온 것이 아니라고 판단하여 배경영상을 차영상과 더하여 갱신하고 만약 임계값보다 크면 움직임 객체가 입력영상에 유입된 것으로 판단하여 배경영상을 갱신하지 않는다.

Fig. 7, 8, 9는 각각 실내조명이 밝음, 중간, 어두울 때의 환경에서 각각 입력된 영상의 원본 영상(좌측) 과 원본 영상에서 채도 정보만 추출한 영상(우측)을 각각 보여주고 있고, Fig. 10은 실내조명이 밝을 때의 원본 영상에 Adobe Photoshop5.5를 이용하여 Noise값 30의 잡음을 유입시킨 영상(좌측)과 이 영상에서 채도 정보만 추출해낸 영상(우측)을 각각 보여주고 있다.

Fig. 7 A bright room

Fig. 8 Medium light

Fig. 9 A dark room

Fig. 10 Including noise

여기서 움직임의 검출과정에서 채도 정보만 추출한 영상을 사용하면 RGB값을 이용한 그레이 영상을 사용했을 때에 비해 밝기 변화에 강하다는 사실을 이론적으로 증명하기 위해 탐색시간보다는 정확한 계산량을 주목적으로 하는 전역 탐색 블록정합 알고리즘에서 정의되는 MAD(Mean Absolute Difference)값을 서로 비교하였다.

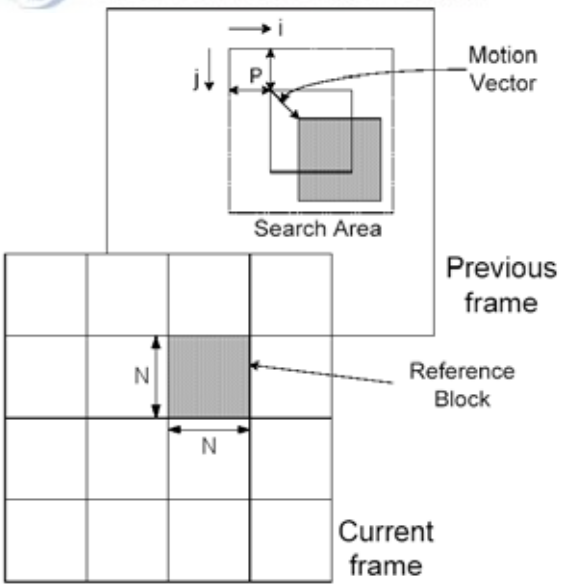


Fig. 11 Motion estimation by block matching algorithm

Fig. 11은 블록정합 움직임 예측과정을 나타내고 있는데 비교되는 두 블록간의 차이를 나타내는 MAD(Mean Absolute Difference)는 다음과 같이 정의된다.

$$MAD(u,v) = \sum_{i=1}^{N}\sum_{j=1}^{N} | S(i+u, j+v) - R(i,j) |, \; -P \le u,v \le P \tag{7}$$

여기서 N은 기준 블록의 크기, P는 탐색영역의 크기를 나타낸다.

MAD값을 비교하기 위해 실내조명이 밝을 때, 중간일 때, 어두울 때와 잡음을 유입시킨 원영상을 각각 그레이 영상으로 변화시킨 영상과 각각의 원영상에서 채도 정보만 추출한 영상을 실내조명이 중간일 때를 기준으로 하여 각각의 MAD값을 계산하고 그 결과를 Table 1.에 기록하였다.

Table 1. Comparison of MAD

영상의 종류 / MAD값	RGB영상 비교	채도정보만 추출한 영상 비교
밝을 때와 중간일 때의 MAD값	40.251877	1.322448
어두울 때와 중간일 때의 MAD값	91.455276	0.295859
잡음 유입시와 중간일 때의 MAD값	13.432942	3.828411

Table 1.에서 그 결과를 살펴보면 실내조명이 중간일 때를 기준으로 한 MAD값은 밝을 때, 어두울 때, 잡음이 유입되었을 때 등 모든 경우에 RGB영상을 이용했을 때가 채도정보만 추출한 영상을 이용했을 때보다 상당히 크다는 것을 알 수 있다. 따라서 채도정보만 추출한 영상을 이용하여 움직임 검출을 시도했을 때 밝기 변화나 잡음 유입에 의해 잘못된 검출을 할 가능성이 일반적인 움직임 검출 기법인 RGB영상을 이용한 차영상 기법보다 매우 적다는 사실을 추론할 수 있다.

앞의 검증에서 얻은 결과를 토대로 시스템을 설계하는 과정 중에서 시간이 지나면서 유입되는 잡음이나 침입자가 수시로 정지하거나, 멈춰있는 경우에는 검출하지 못하는 단점을 해결하기 위해 복잡한 연산을 필요로 하지 않는 범위 내에서 배경영상 갱신

을 이용하기로 하였다. Fig.12는 앞에서 언급한 모든 조건을 만족시키기 위한 본 논문에서 제안한 알고리즘의 전체 구성도를 보여주고 있다.

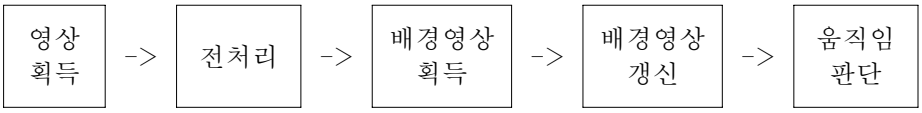


Fig. 12 The Structure of Proposed algorithm

1. 전처리 과정

본 논문에서 영상획득 단계는 영상입력장치로부터 24bit RGB 컬러영상을 입력받아, YUV색상모델로 변경하여 V정보, 즉 채도(saturation) 정보만 획득하는 전처리 단계를 거치게 된다.

이 때 RGB 영상을 YUV 영상으로 변환하는 식(8)을 사용하여 입력받은 RGB 컬러영상을 채도 정보만을 가지는 흑백영상으로 변환한다.

$$
\begin{aligned}
Y &= 0.229R * 0.587G * 0.114B \\
U &= 0.596R * (-0.275G) * (-0.321B) \\
V &= 0.212R * (-0.528G) * 0.311B
\end{aligned} \tag{8}
$$

식(8)에서 Y, U, V 는 각각 YUV색상 모델에서 명암도, 색상, 채도정보를 나타내고 R, G, B는 RGB영상에서 각각의 픽셀정보를 나타낸다. 이때 이렇게 얻어진 채도 정보 V가 초기 배경영상이 된다.

2. 차영상 획득 및 배경영상 갱신

다음 단계에서는 획득된 배경영상과 입력된 영상간의 차영상을 구하여 배경영상을 갱신해야 한다.

배경영상 갱신 단계를 하기 위한 차영상은 다음 식(9)를 이용하여 구하게 된다.

$$DI(x,y) = |g(x,y) - g_b(x,y)| \tag{9}$$

여기서 $g(x,y)$는 현재 입력된 영상의 채도 정보를 가지고 있는 영상이고, $g_b(x,y)$는 초기 배경영상을 나타내고 있으며, $DI(x,y)$는 이 두 영상의 픽셀간 차를 구하여 만든 차영상이다.

이렇게 획득된 차영상은 실제 초기 배경영상과 M × N 영역간 블록 정합(block Matching)시켜 임계값 T_1과 비교하여 0 또는 255값을 부여함으로써 이진화 영상을 획득하고 또 다른 임계값 T_2보다 255값을 가진 픽셀의 개수가 많으면 움직임 객체가 영상에 유입된 것으로 인식하여 움직임 검출이 되었다고 판단하고, 만약 T_2보다 갯수가 적으면 초기 배경영상에 검출된 차영상을 더하여 초기 배경영상을 갱신한다.

$$T_1 = \sum_{x=0}^{x=N} \sum_{y=0}^{y=M} P(x,y)/NM \tag{10}$$

식(10)에서 $P(x,y)$는 x,y 좌표 픽셀의 값을 나타내고 NM은 블록정합시킬 영상의 크기를 나타낸다. 이 때 블록정합을 하기 위해서는 입력영상에서 앞선 식(8)을 적용하여 채도 정보를 얻고 이를 초기 배경영상과 블록정합(Block Matching)시킨다.

$$BI(x,y) = \begin{cases} 255 & (DI(x,y) > T_1) \\ 0 & (DI(x,y) = < T_1) \end{cases} \tag{11}$$

식(11)은 실제 블록정합을 시키면서 임계값 T_1보다 큰 경우에는 픽셀값을 255로 설정하고, 작거나 같은 경우에는 픽셀값을 0으로 설정하여 이진화된 영상 $BI(x,y)$를 획득하는 식을 보여주고 있다.

여기서 획득된 이진화된 영상 $BI(x,y)$의 255픽셀값을 가지는 픽셀의 갯수를 파악하여 지정한 임계값 T_2보다 같거나 작은 경우에는 식(12)처럼 초기 배경영상을 갱신시킨다.

이때 배경영상 갱신은 초기 배경영상과 차영상을 더하여 새로운 배경영상을 만들게 된다.

$$g_u(x,y) = \begin{cases} g_b(x,y)\ (N > T_2) \\ g_b(x,y) + DI(x,y)\ (N =< T_2) \end{cases} \quad (12)$$

식(5)에서 $g_u(x,y)$는 갱신될 배경영상을 나타내고, $g_b(x,y)$는 초기 배경영상, $DI(x,y)$는 입력영상과 초기 배경영상 $g_b(x,y)$간의 차영상을 나타내며, N은 식(11)에 의해 확보된 이진영상에서 255픽셀값을 가지는 픽셀의 갯수를 나타내고 T_2는 움직임 검출에 관한 임계값을 각각 나타낸다.

3. 움직임 검출 및 입력 영상 저장

식(12)에서처럼 이진화된 영상에서 255의 픽셀값을 가지는 픽셀의 갯수가 T_2임계값보다 작거나 같으면 초기 배경 영상은 갱신되고 그렇지 않으면 움직임 객체가 영상에 유입된 것으로 간주하고 움직임 검출로 판단하여 입력영상은 저장되고, 배경영상은 갱신되지 않고 초기 배경영상 그대로 유지된다.

Ⅳ. 구현 및 실험 결과

1. 실험 환경

본 논문에서 구현한 움직임 검출 시스템은 크게, 영상획득, 전처리, 배경영상 획득, 배경영상 갱신, 움직임 판단 부분으로 구성된다.

이 때 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 펜티엄Ⅳ 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 구현하였다.

Table 2는 채도 정보와 배경영상 갱신을 이용한 움직임 검출 시스템의 실험에 사용된 실험 환경과 영상에 대하여 요약한 것이다.

Table 2. Simulation Environments

시스템 사양	Pentium Ⅳ processor 1.7GHz, 256MB RAM
운영체제	Windows XP
프로그램 언어	Visual C++ 6.0, Direct X 8.0 SDK
입력영상 파일 포맷	RGB 24bit 컬러 영상
입력영상의 해상도	320 × 240

본 논문에서 제안한 알고리즘이 밝기 변화에 강건하게 동작한다는 것을 알아보기

위해 다음과 같은 환경에서 PC카메라를 이용하여 입력된 RGB 컬러 영상을 가지고 실험하였다. 각각의 조건하에서도 채도정보만 이용했을 때와 배경영상 갱신을 동시에 이용했을 때, 일반적인 차영상 기법과 각각 비교하여 검출 오류를 체크하였다.

1) 카메라 환경설정을 통해 밝기 값을 단계적으로 변화시키며 실험
2) 실내조명의 밝기를 서서히 변화시키며 실험
3) 임의의 잡음을 유입시키며 실험
4) 검출 속도를 비교하는 실험

2. 적용된 알고리즘

본 논문에서 적용된 알고리즘을 이루는 주요 부분을 살펴보면 다음과 같다. 입력 영상을 받아들여 채도 정보를 추출하는 단계 및 배경영상을 획득하는 단계, 그리고 배경영상을 갱신하거나 움직임을 검출하고 이미지를 저장하는 단계, 즉 세 개의 단계로 구분될 수 있다. 이러한 단계를 각각의 순서대로 처리하여 시스템을 구현하였으며 적용하기 쉽게 플로차트로 표현하였고 Fig. 13에 나타낸 바와 같다.

본 논문에서 사용한 알고리즘은 기본적으로는 차영상 기법을 적용한 움직임 검출 기법이지만 RGB영상을 그레이 영상으로 변환하여 움직임을 검출하는 일반적인 차영상 기법을 배제하고 입력된 영상에서 채도 정보만을 추출하여 배경영상으로 설정한 상태에서 계속적으로 입력되는 영상도 마찬가지로 채도 정보만을 추출하여 배경영상과 비교하는 기법을 사용하는 것이 본 논문에서 가장 핵심적인 알고리즘이다. 이후에 검출 오류를 개선하기 위하여 배경영상을 갱신하는 방법을 추가한 것은 시스템의 완성도를 높이기 위한 보조 역할을 하는 알고리즘이라고 할 수 있겠다.

제안 알고리즘은 차영상 기법을 기본으로 하였고 채도 정보를 추출하는 연산은 비교적 간단한 연산이며 배경영상을 갱신하는 단계에서만 약간의 처리 사간이 더 필요하기 때문에 처리 속도에서는 차영상 기법과 비교했을 때 약간 느린 처리 속도를 보여준다.

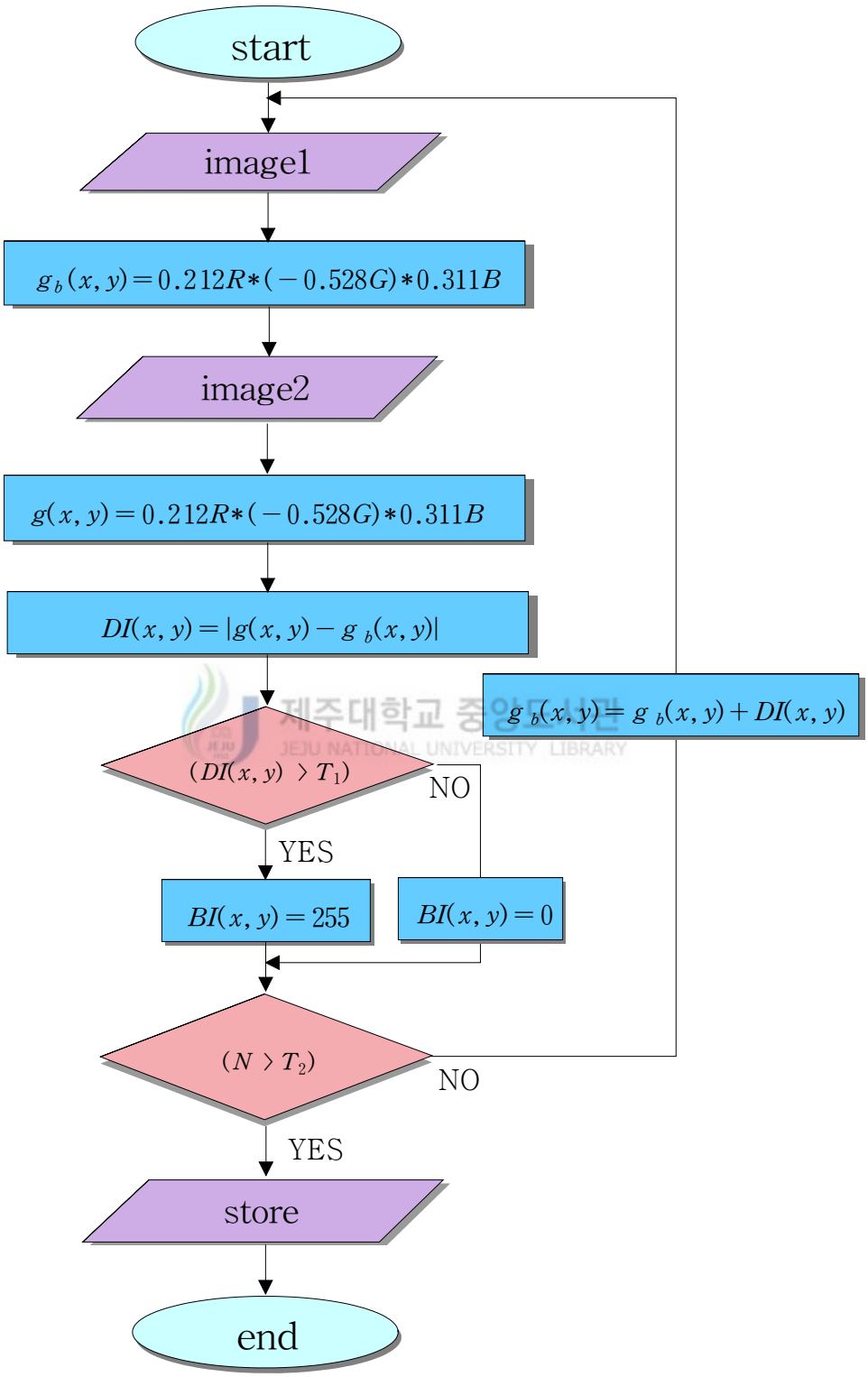


Fig. 13 The Flow Diagram of Proposed algorithm

3. 시스템 구현

구현된 시스템은 차영상 기법과 채도정보만 이용한 기법, 그리고 채도정보 및 배경영상 갱신을 이용한 움직임 검출 과정을 각각 보여주고 있으며 각 기법의 움직임 검출 횟수를 아울러 나타내주고 있어서 각 기법들의 움직임 검출 상태의 비교가 매우 용이하다.

Fig. 14는 구현된 시스템을 캡춰한 화면이며 Fig. 15는 움직임이 검출될 때마다 자동적으로 저장된 이미지들을 보여주고 있으며 밑의 숫자는 임의적으로 붙여지는 파일명이다. 움직임이 있을 때 이미지는 bmp형식으로 저장되며 옵션으로 선택할 수 있도록 구현하였다.

Fig. 14 Shape of the System

10-27-19

10-27-20

10-27-21

10-27-22

10-27-23

10-27-24

21-57-54

21-57-55

21-57-56

21-57-57

21-57-58

21-57-59

Fig. 15 The Captured Images when Motion detect

Fig. 16은 카메라의 환경 설정 화면이며 구현된 시스템에서 바로 실행시킬 수 있도록 작성되었다. 이 대화 상자를 통해서 밝기를 손쉽게 조절할 수 있어 실험을 용이하게 해 주는 역할을 담당하고 있다.

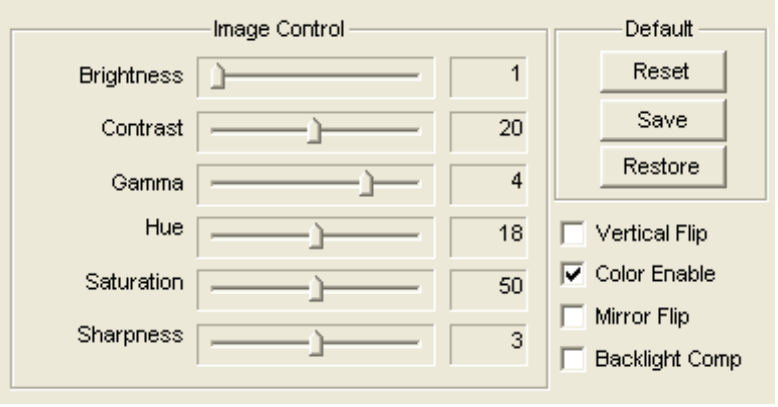


Fig. 16 The Setting of Camera

4. 실험 결과

1) 카메라 환경설정을 통한 밝기 값 변화 실험

실내에서 일정한 밝기 환경의 움직임이 전혀 없는 상태에서 카메라 환경설정을 통하여 단계적으로 밝기를 변화시키면서 움직임 검출 오류를 체크하였다. 이 때 각 단계의 밝기의 증가 또는 감소시키는 값은 7이며 최소값 1로부터 최대값 40까지 변화시켰다. 각 검출 기법들의 비교 결과는 Table 3, 4와 같다. 여기서 검출 오류로 기록된 숫자는 오인식한 횟수를 나타낸다. 실험 결과, 기계 장치에 의해 밝기 변화가 조정되므로 밝기 값의 변화에 따른 검출 오류는 거의 일정한 값을 보여주었다.

Table 3. Motion detection errors by Camera setting when Brightness increase

밝기 변화 단계 / 움직임 검출 기법	1->8	8->15	15->22	22->29	29->36	36->40
채도정보+배경영상 갱신	0	0	0	0	0	0
채도정보만 이용	0	0	0	0	0	0
차영상 기법	7	4	7	0	0	0

Table 4. Motion detection errors by Camera setting when Brightness reduce

밝기 변화 단계 / 움직임 검출 기법	40->33	33->26	26->19	19->12	12->5	5->1
채도정보+배경영상 갱신	0	0	0	1	0	0
채도정보만 이용	0	0	0	2	0	0
차영상 기법	0	0	0	7	4	4

Table 3, 4에서 보는 것처럼 가장 검출 오류가 적은 것은 바로 채도정보와 배경영상 갱신을 이용한 기법이며 다음으로 채도정보만 이용한 기법, 그리고 차영상 기법의 순으로 결과가 도출되었다. 따라서 채도 정보와 배경영상을 이용한 기법이 가장 밝기 변화에 강하다는 것을 이 실험을 통해 알 수 있었다. 그러나 검출 오류가 다소 어두운 환경에서 주로 나타남을 볼 수 있는데 일반적인 환경에서 디지털 입력 장치는 어두울수록 잡음이 많이 발생하는 것이 알려져 있고 이것이 그 원인으로 짐작된다. 그리고 채도정보를 이용한 기법에 배경영상 갱신 기법을 가미한 이유는 시간이 흐르면서 유입되는 잡음에 의한 검출오류를 방지하기 위함이었지만 밝기 변화의 상황에서도 검출 오류를 다소 감소시킴을 확인할 수 있었다.

2) 실내조명의 밝기를 서서히 변화시키며 실험

기본 실내조명이 있는 상태(30W+40W 써크라인 형광등)에서 27W 스탠드를 켠 채

로 카메라의 직상부 30cm 지점에서 전후로 1m 정도의 제한 지점을 설정하고 1초에 한번 정도의 속도로 5회를 왕복으로 움직인 후 검출오류를 체크하였고 이 실험은 총 10회를 반복 실시하여 결과를 기록하였다. 한편, 본 실험의 조건은 창문을 통해 들어오는 광량의 변화나 자연스러운 실내조명의 변화 상태를 연출하는데 초점을 맞추었음을 밝혀둔다.

Table 5. Motion detection errors by Brightness variation of room

실험 횟수 / 움직임 검출 기법	1	2	3	4	5	6	7	8	9	10	평균값
채도정보+배경영상 갱신	1	0	2	0	2	3	1	1	0	2	1.20
채도정보만 이용	5	0	9	0	7	15	3	5	0	9	5.30
차영상 기법	23	19	33	14	25	50	17	20	16	34	25.10

Table 5에 기록된 실험 결과는 카메라의 환경설정을 통한 밝기 값 변화와 거의 유사한 형태를 보여주었다. 역시 채도정보 및 배경영상 갱신을 이용한 방법이 가장 좋은 결과를 보이고 다음으로 채도정보만 이용한 기법, 차영상 기법 순으로 결과가 나타났다.

3) 임의의 잡음을 유입시키며 실험

임의의 잡음을 유입시키는 조건을 주기가 어려워 가습기를 이용해서 잡음이 유입된 것과 유사한 환경을 제공하여 실험을 하였다. 잡음의 강도는 가습량으로 조절하였으며 10초간 잡음에 노출시킨 후 결과를 기록하였다. Table 6은 실험 결과를 보여주고 있는데 본 논문에서 제안하는 알고리즘이 대체적으로 잡음에 강하다는 것을 알 수 있으며 다만 강도 높은 잡음이 유입되었을 때는 제안 알고리즘도 검출 오류가 다소 나타남을 볼 수 있다. 이는 화재 발생의 경우 등 특정한 환경에서는 오히려 더 좋은 결과를 가져올 수 있을 것으로 기대된다.

Table 6. Motion detection errors by Noise

잡음의 강도 / 움직임 검출 기법	1	2	3	4	5	평균값
채도정보+배경영상 갱신	0	0	0	2	10	2.40
채도정보만 이용	0	0	1	7	35	8.60
차영상 기법	0	14	33	65	140	50.40

4) 검출 속도를 비교하는 실험

이 실험을 위하여 채도정보를 이용한 기법, 블록 정합을 이용한 기법, 차영상 기법을 각각 선택할 수 있고 10초간만 동작한 후 연산 횟수를 나타내도록 하는 시스템을 추가적으로 구현하였다. Fig. 17은 추가적으로 구현된 시스템의 모습을 나타내고 있으며 앞의 실험과 마찬가지로 10번의 실험 후에 결과를 기록하였다.

Fig. 17 Shape of another System

Table 7. Number of times to Operate

실험 횟수 / 움직임 검출 기법	1	2	3	4	5	6	7	8	9	10	평균값
채도정보 이용	60	61	63	61	62	64	63	61	60	63	61.80
블록정합 이용	24	23	22	23	20	21	25	30	27	26	24.10
차영상 기법	66	67	69	70	62	64	67	68	71	68	67.20

이 실험에서는 세 가지의 움직임 검출 기법을 동일한 조건하에서 검출 속도를 비교하여 Table 7에 10초간의 처리 횟수를 기록하였다. 가장 우수한 결과를 보인 것은 역시 움직임 검출 기법 중 일반적으로 많이 이용되는 차영상 기법이며 두 번째는 제안 알고리즘이고 블록 정합을 이용한 움직임 검출 기법이 가장 느린 결과를 보여주었다. 실험 결과를 살펴보았을 때 제안 알고리즘의 움직임 검출 속도가 어느 정도 만족할 만한 수준임을 확인할 수 있었다.

V. 결 론

보안 감시 시스템에서 움직임 검출 알고리즘은 침입자를 탐지하고 영상을 저장하는 데 있어서 상당히 중요한 비중을 차지하는 부분이다. 따라서 다양한 환경에서 얼마나 정확하게 침입자가 발생하였는지 판단하고 저장할 수 있도록 하는 기능은 보안 감시 시스템의 성능을 좌우하는 중요한 요소이다.

이러한 이유로 현재 다양한 움직임 검출 알고리즘들이 나오고 있으나, 밝기 변화나 영상에 유입된 잡음, 자연적인 물체의 움직임 등의 환경에서는 오동작하는 문제점들이 발생하고 있다.

따라서 본 논문에서는 이러한 여러 가지 움직임 검출 오류를 발생시키는 환경들 중 많은 비중을 차지하고 있는 밝기 변화에 따른 움직임 검출 오류와 영상에 유입되는 잡음에 의한 오류에 대해 해결할 수 있는 방법을 제안하고 있다.

본 논문에서는 밝기 변화와 잡음에 의한 오류를 해결하기 위해 입력되는 RGB 컬러 영상에서 색상 정보와 명암 정보를 제외한 채도 정보만을 가지고 움직임 검출을 하였다.

일반적으로 밝기 변화가 있는 영상에서는 명암 정보가 가장 많이 변화하였으며, 잡음이 유입된 영상에서는 색상 정보의 변화가 가장 많이 발생하였다. 따라서 본 논문에서 제안한 알고리즘에서는 명암과 색상 정보를 제외시켜 채도 정보만을 가지고 움직임 검출을 하게 되었다.

이러한 특징을 본 제안 알고리즘에 적용하기 위해서 처음 입력된 RGB 컬러 영상을 색상과 명암 정보를 제외시키고자 YUV 영상으로 변화시키고 이 중에서 채도 정보만을 가지고 있는 V영상을 배경영상으로 지정하였다. 그 후 두 번째 입력된 영상에서 움직임이 발생하였는지 판단하기 위해 역시 채도 정보만을 지니고 있는 V영상을 검출하고 처음 입력된 V영상과 블록정합(Block Matching)시켜 특정 임계값 이상인 경우에만 움직임이 발생한 것으로 인식되도록 하였다. 또한 시간이 흐르면서 유입되는 잡음에 의한 움직임 검출 오류를 방지하기 위해 배경영상 갱신의 방법을 추가하였다.

이러한 실험 결과 제안한 알고리즘은 기존의 움직임 검출 알고리즘들에 비해 밝기

변화가 있는 환경과 입력장치에 의해 유입되는 잡음이 있는 경우에 대해서는 좋은 인식률을 보였으나, 처리속도 면에서는 차영상 기법에 다소 뒤지는 결과를 보였다. 따라서 차후 처리속도 개선을 위한 연구와 더불어 입력영상에 유입되는 여러 가지 유형의 잡음들에 대해 분석하여 잡음 유입시 움직임 검출 오류를 좀 더 개선시킨다면 실제 보안 감시 시스템에 적용시켜 적절하게 활용할 수 있을 것으로 보인다.

[참고문헌]

1. 이규원, 김영호, 이재규, 박규태 “무인 감시 장치 구현을 위한 단일 이동물체 추적알고리즘”, 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. 김계영, 이은주, 최형일, “차영상 분석에 의한 동작 정보의 추출”, 한국정보과학회 논문지, 제 21권, 제 8호, 1994
5. 이회영, 최재영, 강동구, 김홍수, 차의영 “배경영상을 이용한 목표물 추적에 관한 연구” 한국멀티미디어학회 춘계학술발표논문집, pp.386-390, 1999
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. 김용균, 이광형, 최내원, 오해석, 지정규, “동영상에서 배경영상을 이용한 실시간 객체 추적”, 한국정보과학회 02 가을 학술발표논문집(2), 2002권, pp.532-534, 1992
9. N.McFarlance, "Segmentation and Tracking 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 Recognition 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, "Multiframe ship detection 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

의욕적으로 대학원 공부를 시작했지만 학기 중간에 아내의 병고로 인하여 학업의 지속이 어려운 형편이었습니다. 하지만 지도교수님이신 김장형 교수님을 비롯한 학과의 교수님들과 대학원 선배 및 동료들의 아낌없는 도움으로 어렵사리 대학원을 수료했습니다.

그러나 중간에 임지를 전북으로 옮기고 사랑하는 아내를 세월에 앞서 저 천국으로 먼저 보내는 힘겨운 과정 중에 논문은 쓸 엄두도 내지 못했고 대학원을 수료하고서도 계속적으로 지도교수님과 연락은 되었지만 지역적인 거리감과 가정의 어려움으로 인하여 마음은 움츠러들고 논문 작성은 한없이 뒤로 미루어졌습니다. 어린 자녀 둘을 혼자서 감당하기 어려워 부모님께서도 서두른 일이었지만 나름대로 핑계를 대어 새 삶을 함께 하게 된 지금의 아내가 적극적으로 나서지 않았더라면 결국 대학원 졸업은 어려운 숙제로만 남고 말았을 것입니다.

이제 대학원 학업을 마무리하는 자리에 있어서 우선 저를 힘껏 내조한 아내의 도움에 가장 먼저 감사를 드리고 싶고 언제나 관심을 가지고 끝까지 성심 성의껏 지도해 주신 김장형 지도 교수님과, 이렇게 좋은 결과를 맺을 수 있도록 학업 생활을 이끌어 주시고 조언을 아끼지 않으신 변상용 교수님, 안기중 교수님, 곽호영 교수님, 이상준 교수님, 송왕철 교수님께 감사의 마음을 전합니다.

또한 논문을 다듬고 모양을 갖추는 데 많은 도움을 준 고봉수 선생님에게 깊은 감사를 드리며 논문 발표와 인쇄에 성심껏 협력해 준 양동호, 강경희 선생님, 그리고 함께 연구하고 생활하던 강진석, 강명화, 강영도, 양영수 선생님을 비롯한 멀티미디어 연구실의 여러 식구들, 학과 사무실에서 저희들을 위해 애써주신 정은경, 이정하 조교 선생님께도 감사의 마음을 전하는 바입니다.

끝으로 격려의 말씀을 아끼지 않으시고 물심양면으로 저를 도와주시고 가슴 졸이며 지켜봐 주신 부모님의 은혜에 진심어린 감사를 드립니다.

하나님의 넓으신 은혜가 이 모든 분들께 항상 함께 하시길 바라며 감사의 글을 마칩니다.