碩 士 學 位 論 文
指導敎授 姜 熙 中

중복이 제거된 합성 영상과 에지 영상의 AND 연산을 통한 다수 보행자의 검출 및 추적

Detection and Tracking of Multiple Pedestrians Using AND Operation of Redundancy-removed Composition Image and Edge Image

2006年 6月 日

漢城大學校 大學院

컴퓨터工學科

컴퓨터工學 專攻

崔 良 鎭

碩 士 學 位 論 文
指導敎授 姜 熙 中

중복이 제거된 합성 영상과 에지 영상의 AND 연산을 통한 다수 보행자의 검출 및 추적

Detection and Tracking of Multiple Pedestrians Using AND Operation of Redundancy-removed Composition Image and Edge Image

위 論文을 컴퓨터工學 碩士學位論文으로 提出함

2006年 6月 日

漢城大學校 大學院

컴퓨터工學科

컴퓨터工學 專攻

崔 良 鎭

崔良鎭의 工學 碩士學位論文을 인정함

2006年 6月 日

심사위원장 김 진 환 (인)

심사위원 이 항 찬

심사위원 강 희 중

목 차

표 목차

그림 목차

요 약

동영상 내에서 이동하는 객체를 추적하기 위해서는 우수한 객체 검출 및 추적 방법이 필요하다. 이를 위하여, 본 논문에서는 인접한 프레임들의 합성과 차를 이용하여 움직이는 보행자의 대략적인 형태를 알아내고, 대략적인 형태를 이진화 시킨 영상과 현재 프레임의 에지 영상과의 AND 연산을 통하여 보행자의 형태를 알아내는 방법을 제안한다. 그리고 이 과정에서 생성되는 노이즈를 채움 연산과 영역화 연산을 통하여 제거하였고, 얻어진 보행자의 크기 비율을 고려한 수직 투영을 통하여 다수의 보행자를 보다 정확하게 분리해 낼 수 있었다.

제안된 보행자 검출 방법에 의해서 검출된 보행자는 유사도를 평가하여 추적을 수행하게 된다. 유사도란 이전 프레임에서 추적되던 객체(보행자)와 현재 프레임에서 검출된 객체(보행자) 간의 유사한 정도로, 본 논문에서 제안하는 이를 평가하는 방법은 연결성, 인접성, 방향성, 검출된 객체의 크기의 값을 평가하는 것이며, 평가요소 점수의 합으로서 현재 프레임에서 검출된 객체의 리스트와 이전 프레임에서 검출된 객체의 리스트의 비교를 통해서 최대값을 갖는 리스트의 노드를 업데이트하여 보행자 추적을 한다. 실제 환경에서 수행된 실험에서는 제안된 보행자 검출 및 추적 방법을 기존의 이전 프레임과 현재 프레임의 차영상만을 이용할 경우와 비교하여 객체 검출이 개선됨을 보여 주었으며, 정확하게 추적됨을 보여 주었다.

제 1장 서론

1.1 연구 배경

최근 들어 컴퓨터를 이용한 영상 전송기술의 발전과 더불어 무인 감시 장치나 영상 회의 시스템 등의 응용을 목적으로 한 카메라 자동 추적 시스템의 연구개발이 널리 진행되고 있다. 이러한 무인 감시 시스템은 감시를 요하는 곳에서 영상 전송기술을 바탕으로 원격지에서 관리할 수 있고, 사람이 24 시간 감시를 하지 않아도 된다는 점 등이 상당한 장점으로 부각될 수 있다. 감시 카메라가 들어가는 감시 장치들의 경우 아무도 없는 빈 화면만을 녹화하거나 지속적으로 모니터링을 한다는 것은 매우 비효율적인 일이 되기 때문에 입력 받은 영상 중 움직임이 검출될 때 경고음을 내 주거나 그 화면만을 녹화함으로써 이러한 시스템의 효율을 높일 수 있다[1]. 이러한 시스템을 컴퓨터 비전시스템이라 하며 최근 실생활에 많이 쓰이고 있다.

컴퓨터 비전시스템의 목적은 컴퓨터를 기반으로 스캐너, 카메라 등의 영상 감지 장치를 이용해서 입력영상을 얻고, 얻은 입력 영상으로부터 주어진 문제에 대한 지식과 영상처리에 관한 일반적인 정보를 활용하여 유용한 정보를 검출하는데 있다[2]. 오늘날 컴퓨터 비전시스템은 하드웨어와 기술의 발전으로 인하여 여러 응용 분야에 적용되고 있다. 그 예로서 컴퓨터 비전시스템으로 검출할 수 있는 정보를 이용하여 범죄의 예방에 필수적인 감시 시스템 및 보안 시스템이 사용되고 있으며, 영상 회의 시스템 및 유비쿼터스 환경에서의 중요한 요소로도 활용될 수 있다. 실제로 현재 교통통제 시스템과 불법 주정차 감시 시스템을 비전시스템을 이용하여 운용하고 있다.

이러한 컴퓨터 비전 시스템에서 움직이는 객체의 검출은 필수 요소이며 중요한 문제가 될 수 있다. 이의 근거는, 움직이는 객체가 확실하지 않다면 추적을 하는데 많은 오류가 발생할 것이며, 오작동의 근본적인 문제가 되기 때문이다.

1.2 연구 목적 및 방법

이동 물체를 검출하는 방법에 대해서는 그 동안 많은 연구가 이루어져 왔다. 그 중에서 고정된 카메라를 이용하는 경우, 가장 널리 사용되고 있는 방법은 계산량이 적고 처리 시간이 빠른 차영상(Difference Image)방법을 많이 이용한다[3]. 차영상을 이용하는 방법 중 이전 프레임과 현재 프레임을 이용한 차영상 방법의 경우, 움직이는 객체의 형태를 정확히 알 수가 없는 단점이 존재하였다. 따라서 본 논문에서는 움직이는 객체를 검출하기 위한 방법으로 이전 영상과 현재 영상의 차이를 이용할 때 발생할 수 있는 객체 형태의 불확실성 문제점을 제거하기 위하여 중복이 제거된 합성 영상과 에지 영상의 AND 연산을 통한 다수의 보행자의 검출 및 추적을 하는 비교적 간단한 방법을 제안하고자 한다.

1.3 논문의 구성

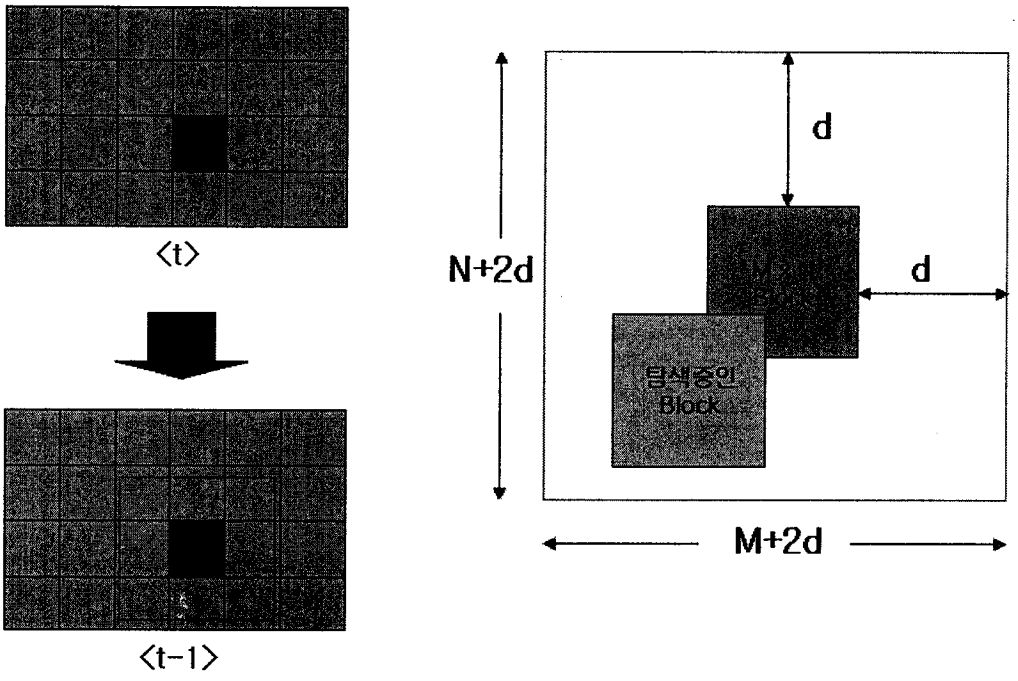
본 논문은 5 장으로 구성되어 있으며, 1장에서는 서론, 2장에서는 관련 연구, 3장에서는 제안방법을 세부적으로 기술하였고, 4장에서는 실험 환경 및 실험 장비에 대해 기술하기로 하며, 5장에서는 결론 및 향후 연구과제에 대해서 논의한 후 참고문헌을 밝히고 논문을 마치고자 한다.

제 2장 관련 연구

본 장에서는 움직이는 물체를 검출 및 추적하기 위한 대표적인 방법인 블록 정합 기법, 광류, 차영상에 대해서 알아보고자 한다. 블록 정합 기법이란 영상을 임의의 블록 단위로 나누어서 그 블록의 유사도를 평가하여 움직임 벡터를 추정하는 방법이다[4]. 광류는 직관적으로 움직임을 구분할 수 있는 특징으로서 영상에서 밝기(intensity)로 나타내는 화소들의 움직임을 나타낸다[5]. 마지막으로 차영상은 두 프레임의 시간차가 아주 작고 이동하는 물체가 서로 겹치는 부분을 제거함으로써 움직이는 객체를 검출하는 방법이다.

2.1 블록 정합 기법(Block Matching Algorithm)

블록 정합 기법은 한 영상을 몇 개의 임의의 작은 블록으로 나눈 후 블록 내의 모든 픽셀은 동일 방향의 움직임을 갖는다고 가정하여 움직임 벡터(motion vector)를 추정하는 것으로 이전 프레임에서 설정된 탐색영역에서 어떤 블록이 현재 프레임의 정해진 블록으로 블록 당 픽셀의 변화가 가장 작은 조합을 찾아내어 얼마나 이동하였는가를 판별하는 것이다. [그림 2.1]에서 처럼 t-1 번째 프레임의 검은색의 블록의 위치를 t 번째 프레임에서 N+2d, M+2d의 범위 안에서 찾게 되는 것이다[6].



[그림 2.1] 블록 정합 기법

블록 정합 기법은 영상 신호의 압축 기법인 움직임 보상 부호화 기법(Motion Compensation coding)에서 움직임 추정을 위해 사용되는 방법 중의 하나이다. 움직임 보상 부호화 기법이란 연속적인 영상에서 움직이는 물체의 이동 값을 추정하고 이로써 복원된 영상과의 예측 오차를 구하여 움직임 정보와 예측 오차만을 부호화하는 방법이다. 블록 정합 기법은 간단하며 실시간 처리가 용이하기 때문에 화상회의, 화상 전화, HDTV등에서 동영상 부호화에 사용되고 있다.

블록 정합 기법은 시간적으로 서로 이웃한 두 장의 프레임에서 각각의 프레임을 일정한 블록을 나눈 후 해당 블록의 움직임을 추정한다. 일반적으로 움직이는 물체는 회전 운동 또는 크기의 변화가 있을 수 있으나 연

속 프레임 간에는 물체의 변형이 많이 생기지 않으며, 단지 물체의 평행 이동으로 근사화 할 수 있다는 전제하에서 사용되는 움직임 추정 방식이다.

블록 정합 기법에서는 t-1 번째 프레임과 t 번째 프레임을 M×N 크기의 일정한 블록으로 나눈 후, t 번째 프레임에서 각 블록에 대하여 t-1 번째 프레임 내에서 유사도가 가장 높은 블록을 찾는다[7].

블록 정합 기법에서 서로 정합될 수 있는 두 블록 사이에서 유사도의 정도를 나타내는 척도로 MAD(Mean Absolute Difference), MSD(Mean Square Difference)가 널리 쓰이고 있고, 그 외에도 MPC(Matching Pixel Count), Cross-correlation 값이 가장 큰 블록을 찾는 방법인 최대상관계수(Maximum Cross Correlation)법 등이 있다[8].

MAD는 (식 2.1)과 같이 표현되며, MSD는 (식 2.2)에 표현 된다.

$$MAD(u,v)=\frac{1}{N_1\times N_2}\sum_{i=0}^{N_1-1}\sum_{j=0}^{N_2-1}|L^{n+1}(i+u,j+v)-L^{n}(i,j)| \quad \text{(식 2.1)}$$

$$MSD(u,v)=\frac{1}{N_1\times N_2}\sum_{i=0}^{N_1-1}\sum_{j=0}^{N_2-1}|L^{n+1}(i+u,j+v)-L^{n}(i,j)|^2 \quad \text{(식 2.2)}$$

여기서 $L^n(i,j)$는 n 번째 프레임, $L^{n+1}(i,j)$는 n+1 번째 프레임의 (i,j) 위치에서의 밝기 값을 나타내며 (u,v) 는 탐색 영역에서의 탐색점이다. 각 탐색점에 대해 MAD 혹은 MSD가 최소가 되는 (u,v)는 모션 벡터(motion vector)로 정의한다. MAD는 MSD에 비하여 계산량이 적고 하드웨어 구

현이 용이하여 널리 이용한다. 유사도 평가를 위한 탐색 방법이 연구되기도 하였으며, 모든 블록을 모두 탐색하는 full-search 알고리즘, three-step search 방식[9], direction of the minimum distortion 방식[10], 메뉴 방식[11] 등이 있다. 본 논문에서는 full-search 알고리즘과, three-step search 방식에 대해서만 기술하였다.

2.1.1 Full-search 알고리즘

Full-search 알고리즘은 탐색 범우 내에서 모든 위치를 탐색하기 때문에 최적의 모션 벡터(motion vector)를 찾을 수 있다. 그러나 이 방법은 탐색 영역의 모든 위치에서 두 블록 사이의 불일치도 값을 계산하기 때문에 정확도는 우수하지만 전체 탐색 영역을 비교하기에 처리 속도가 떨어지는 단점을 가지고 있다.

현재 프레임의 블록에 대응되는 블록을 이전 프레임에서 찾는 방법 중에서 가장 기본적인 방법이다. 이 알고리즘에서는 모션 벡터가 존재할 수 있는 모든 탐색 범위에서 픽셀을 계산하여 그 중에서 값이 최소가 되는 위치의 블록을 대응블록으로 선택한다.

2.1.2 Three-step search 알고리즘

3단계 탐색 알고리즘은 Koga[12]가 제안한 것으로 계산이 간단하고 규칙성 때문에 많이 사용되고 있는 알고리즘이다. 이 알고리즘은 단일 에러 표면 가장(Unimodal Error Surface Assumption)에 기본으로 하고 있는데, 이것은 각 탐색점의 에러가 정확한 움직임 벡터에서 멀어 질수록 커진다는 것을 나타낸다. 이는 모든 영상에 모두 정확하게 들어맞는 것은 아

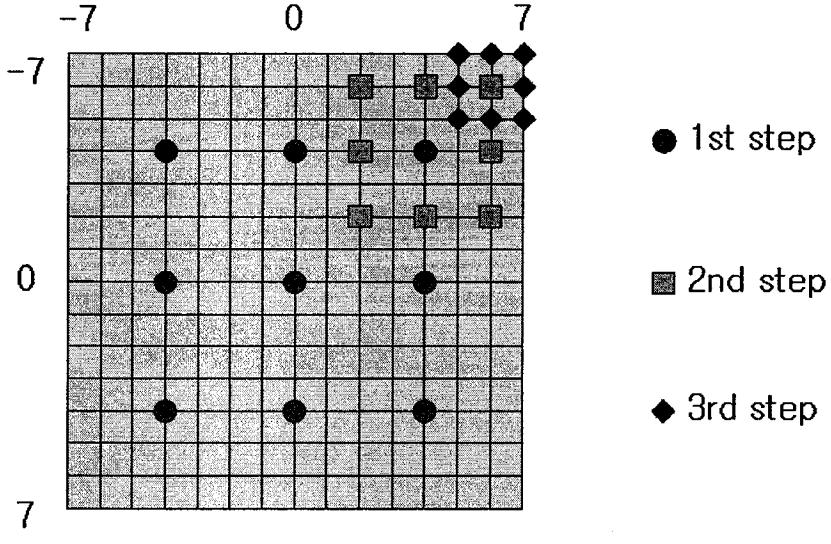
니지만 대부분의 영상에 근사하게 들어맞는다.[13]

[그림 2.2]는 3 단계 탐색 알고리즘의 과정을 보여준다. 3 단계 탐색 알고리즘은 고속 움직임 추정을 위한 여러 방법들 중에서 탐색점의 개수를 줄이는 방법이다. 탐색 구간을(W)라고 할 때 움직임 추정을 위해 필요한 탐색 단계의 수(S)는 (식 2.3)과 같다.

$$S = [\log_2(W+1)] \quad \text{(식 2.3)}$$

여기서 [X]는 x보다 크거나 같은 수에서 가장 작은 정수를 나타낸다. 위 식에서 구한 필요한 총 단계의 수에 대한 탐색점의 개수(N)는 (식 2.4)과 같이 나타낼 수 있다[8].

$$N = 8 \times S + 1 \quad \text{(식 2.4)}$$



[그림 2.2] 3 단계 탐색(three-step search) 알고리즘

[그림 2.2]에서 최대 이동 거리를 ±6 화소라고 가정하고 세 단계를 거쳐서 이동 벡터를 구하게 된다. 첫 번째, 기준점을 중심으로 상, 하, 좌, 우, 사선 방향으로 4 화소씩 떨어진 위치에 대해 MAD를 구한다. 이 때 가장 최소의 MAD의 값을 나타내는 위치를 (i+4, j-4)을 구한다. 두 번째로 이 위치를 기준으로 사선 방향으로 2 화소씩 떨어진 위치에 대해 MAD를 구하고 마지막으로 사선 방향으로 1 화소씩 떨어진 MAD를 구하여 가장 최소의 MAD값 (i+5, j-5)을 구한다.

2.2 광류(Optical Flow)를 이용한 방법

광류 방법은 영상에서의 밝기 부분의 변화와 연관된 2 차원 velocity field이다. 실제로 3 차원 공간상에서 물체의 움직임 추정을 할 경우에 motion field를 구하기가 상당히 힘들기 때문에 2 차원 image plane에서의 광류를 구하여 근사적으로 3 차원 motion field를 구한다.

일반적으로 카메라로 얻는 영상에서는 각 화소를 이루는 광도만을 측정할 수 있는데, 광류를 구하는데 광도를 어떻게 이용하느냐에 따라 미분법(gradient)을 이용한 방법, 상관관계(correlation)를 이용한 방법, 에너지를 이용한 방법 등으로 분류한다.

2.2.1 미분법(gradient)을 이용한 방법

미분법(gradient)을 이용한 방법은 영상 평면의 광도는 시간이 흘러도 일정하다고 가정하는데, 이는 결과적으로 영상 내에서 광도의 변화는 운동(motion)에 의해 발생한 것이라 할 수 있다.

Horn과 Schunck[14]는 이러한 가정을 토대로 광류를 수학적으로 정의하였다. 즉 이동 물체의 광도 $E(x,y,z)$는 시간에 대해서 일정하다고 가정하므로, 시간 t로 광도를 미분한 값은 (식 2.5)과 같이 '0'이 된다.

$$\frac{dE(x,y,t)}{dt} = \frac{E(x+dx,y+dy,t+dt)-E(x,y,t)}{dt} = 0 \quad \text{(식 2.5)}$$

$E(x+\delta x, y+\delta y, t+\delta t)$를 Tayer 급수로 전개하면,

$$E(x+\delta x, y+\delta y, t+\delta t) = E(x,y,t) + \frac{\delta E}{\delta x}dx + \frac{\delta E}{\delta y}dy + \frac{\delta E}{\delta t}dt + h.o.t \quad \text{(식 2.6)}$$

(식 2.6)에서 고차항 h.o.t를 무시하고 1 차항 만을 이용하여 (식 2.5)에 대입하여 정리하면 (식 2.7)과 같은 운동 제약 방정식(motion constraint equation)을 얻을 수 있다.

$$E_x u + E_y v + E_t = 0 \quad \text{(식 2.7)}$$

여기서, E_x, E_y, E_t는 각각 x, y, t에 대한 $E(x,y,t)$ 의 편미분을 나타내고, u, v는 $\frac{dx}{dt}$, $\frac{dy}{dt}$로 각각 x축, y축에 대한 광류 성분을 나타낸다. 그러나 (식 2.7)은 미지수가 두 개이므로 (식 2.5)로부터 광류를 구할 수가 없다. 따라서 u, v를 구하기 위한 많은 방법들이 제안되기도 하였다.

2.2.2 상관관계(Correlation)를 이용한 방법

영상을 샘플링 하는 과정에서 발생하는 잡음과 aliasing으로 인해 정확하게 미분하는 것은 불가능하다. 따라서 특징을 추출하여 이를 정합시키는 방법이 gradient 방법 보다 더 적격이라 할 수 있다. 이 방법은 일반적으로 구역의 광도 분포가 일정하다고 가정하고, 이전 영상의 각각의 화소에 대해서 다음 영상에서 가장 잘 정합되는 화소를 찾는다. 이때 가장 잘 정합되는 화소 간의 변위가 곧 광류이다[15].

이 방법에 대해 연구하는 사람으로 대표적으로 Anndan을 들 수 있는데, 그는 Laplician 피라미드로 영상을 계층화하고 해상도에 따라 소한 영상으로부터 밀한 영상으로 정합시켜 광류를 구했다[7].

이 방법에서 사용하는 특징들로는 포인트, 에지, 코너, 영역 등이 있다. 사용하는 특징에 관계없이 이 방법은 영상 사이의 특징을 대응해야 하는 문제가 있다.

2.2.3 Energy를 이용하는 방법

Energy를 이용하는 방법은 Heeger에 의해 시작이 되었고, 이동 물체에 대한 시·공간 주파수 특성에 대한 연구에 초점이 맞춰져 있다[15]. 이 방법은 연속적으로 입력되는 영상을 Fourier 변환했을 때 이동 물체의 시·공간 주파수는 (식 2.8)와 같이 이동 물체의 속도와 관련이 있음을 이용한 것이다.

$$w_t = w_x u + w_y v \quad \text{(식 2.8)}$$

여기서 w_x와 w_y는 각각 이동 물체의 공간적 주파수를 나타내고, w_t

는 시간적인 주파수를 나타낸다. (식 2.8)에서 만약 이동 물체의 시·공간적 주파수를 w_x, w_y, w_t공간에 나타내면 이것은 평면 위에 놓이게 된다. 즉, 이동 물체의 에너지는 w_x, w_y, w_t 공간의 평면에 있는 시·공간적 주파수에 포함되어 있다. 이때 광류 (u,v)는 이 평면을 결정함으로써 구할 수 있다. 이 방법은 정확도는 높지만 주파수로 변환을 해야 하므로 시간이 많이 걸리는 단점이 있다.

2.3 차영상(Difference Image)을 이용한 방법

차영상 방법은 두 프레임의 시간차가 아주 작고 이동하는 물체가 서로 겹치지 않는다고 가정하며 우리는 쉽게 영상차로부터 움직임이 없는 배경을 제거하고 움직이는 물체만을 얻을 수가 있다[3].

두 장의 이미지를 이용하여 차영상 방법을 이용할 경우 (식 2.9)와 같은 방법을 사용하여 차영상을 구한다.

$$d(x,y)=|F_{t-1}(x,y)-F_t(x,y)| \quad \text{(식 2.9)}$$

여기서 d(x, y)는 이전 프레임($F_{t-1}(x,y)$)와 현재 프레임($F_t(x,y)$)의 차의 절대값으로 이를 적절한 임계치를 이용하여 이진화 시켜 사용하기도 한다. 또 이 임계치를 구하는 방법이 다양하게 연구되기도 하였다.

기존의 이동 물체의 추출을 위한 방법을 살펴보면, 크게 배경 영상을 참조하는 방법[15,16]과 연속적인 두 개의 영상을 이용하는 방법[18,19,20]과, 연속되는 세 개의 영상을 이용하는 방법이 있다.

2.3.1 배경 영상을 참조하는 방법

배경 영상을 참조하는 방법은 움직이는 물체가 존재하지 않는 배경 영상과 현재 입력 영상의 차를 구하여 변화가 나타난 부분을 움직이는 물체로 추출하는 방법이다. 그러나 이 방법을 사용할 경우의 문제점은 영상 입력 당시의 조명 상태에 따라 배경 영상과 현재 영상의 화소값(gray value)이 다르기 때문에 이동 물체가 존재하지 않는 배경 부분에 대해서도 변화가 나타난다는 점이다. 따라서 입력되는 영상의 조명 상태와 비슷하게 배경 영상을 수정한 후 사용해야 하는 단점이 있다. 그러나 배경 영상의 수정이 이루어진다 하여도 현재 영상의 조명 상태와 동일하게 수정하기는 매우 어려운 일이다[21]. 이 문제를 해결하기 위해서 경계선과 색상 정보를 이용한 이동 물체 추출 방법이 제시되었는데[16], 이 방법에서는 배경 영상과 현재 영상의 경계선을 추출하고자 찾아진 두 경계선 영상 및 색상 정보를 이용한다.

그러나 배경 영상과 입력 영상을 이용하는 방법은 실제 야외 영상에 적용하기에는 많은 문제점이 있다. 왜냐하면, 실외 영상은 실내 영상과 달리 조명 상태의 변화가 심하기 때문이다. 예를 들어, 도로 영상에서 길가의 가로수나 건물의 그림자가 도로에 나타나 있다면 배경 영상을 현재 영상과 같은 조명 상태로 수정하기는 불가능한 일이며, [16]의 방법에서도 경계선 검출 시에 그림자에 의한 경계선으로 인하여 현재 움직이지 않고 있는 그림자도 움직이는 물체로 추출되는 문제점이 있다.

2.3.2 연속되는 두 개의 영상을 이용하는 방법

동영상에서 연속되는 두 개의 영상을 이용하는 방법에서는 인접한 두 영상의 차이를 구해서 차이가 0이 아닌 부분에 움직이는 물체의 윤곽선이 나타남을 이용한다. 이 방법은 연속적인 영상을 이용하기 때문에 조명의 변화를 무시할 수 있으며, 실제 움직임이 있는 부분만을 추출할 수 있는 장점이 있다. 그러나 이 방법은 움직이는 물체의 속도에 따라 그 경계선의 굵기가 일정하지 않게 나타나는 단점이 있다. 그래서 퍼지 추론(fuzzy inference)과 프레임 통합(frame integration)을 통해서 윤곽선의 굵기를 일정하게 추출하는 방법이 제안되었다[20]. 그러나 이 방법은 영상 안에 하나의 물체만이 존재함을 가정하고 있으며, 이 방법에서도 물체의 이동 방향과 평행한 윤곽선은 정확하게 추출하지 못하는 단점이 있다.

본 논문에서는 이동 물체의 정확한 경계선 추출을 위한 방법으로서, 중복이 제거된 영상의 합성과 에지 영상의 AND 연산을 통하여 다수의 보행자를 검출 및 추적하는 방법을 제안한다.

2.3.3 연속되는 세 개의 영상을 이용하는 방법

연속되는 세 프레임의 영상을 $F_{t-1}(x,y)$, $F_t(x,y)$, $F_{t+1}(x,y)$ 이라 할 때, 연속되는 두 개의 프레임의 차를 (식 2.10), (식 2.11)과 같이 구한다.

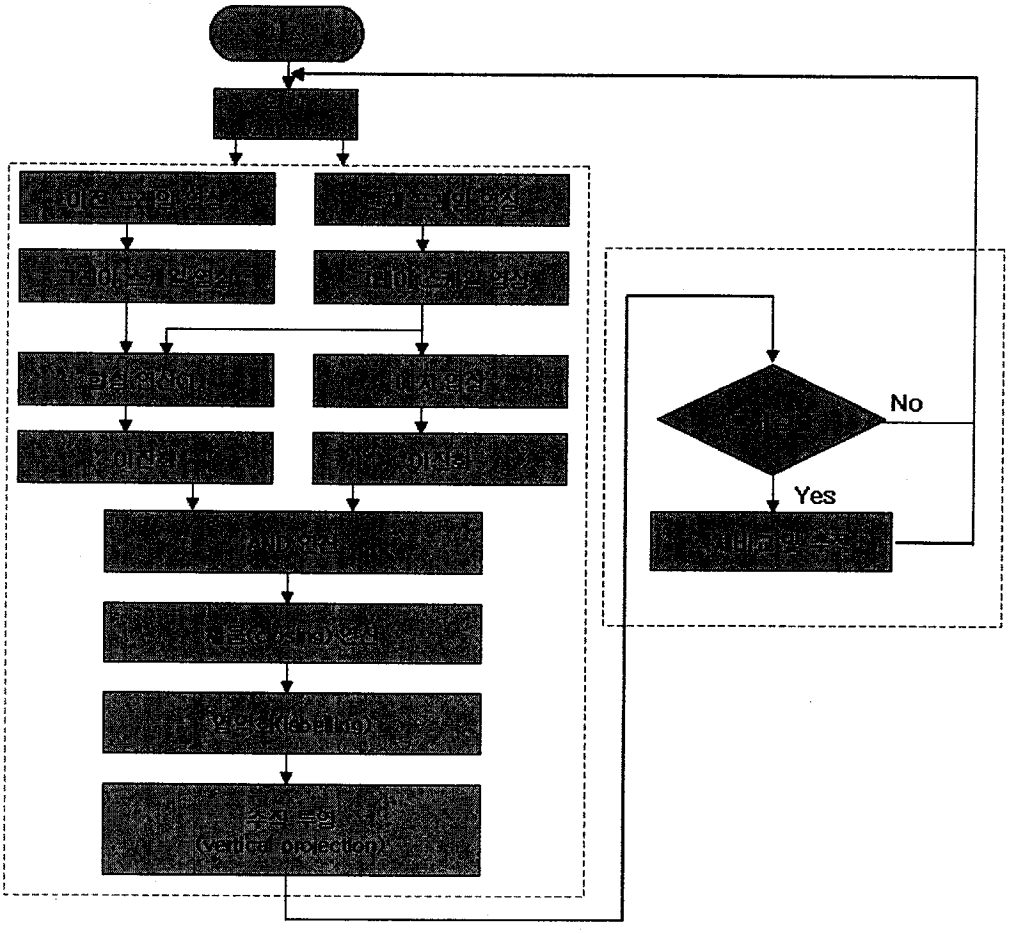
$$d_{t,t-1}(x,y) = |F_t(x,y) - F_{t-1}(x,y)| \quad \text{(식 2.10)}$$

$$d_{t+1,t}(x,y) = |F_{t+1}(x,y) - F_t(x,y)| \quad \text{(식 2.11)}$$

그리고, 임계치 과정을 두 개의 차 영상 $d_{t,t-1}(x,y)$, $d_{t+1,t}(x,y)$에 적용하게 되는데 이는 움직이는 부분을 확실히 하기 위해서 이다. 여기에 임계치를 적용하여 두 개의 이진 영상을 만들 수가 있는데, 두 개의 이진 영상을 AND 연산을 통하여 프레임의 이동 부분을 구할 수 있다.

제 3장 보행자 검출 및 추적 방법의 제안

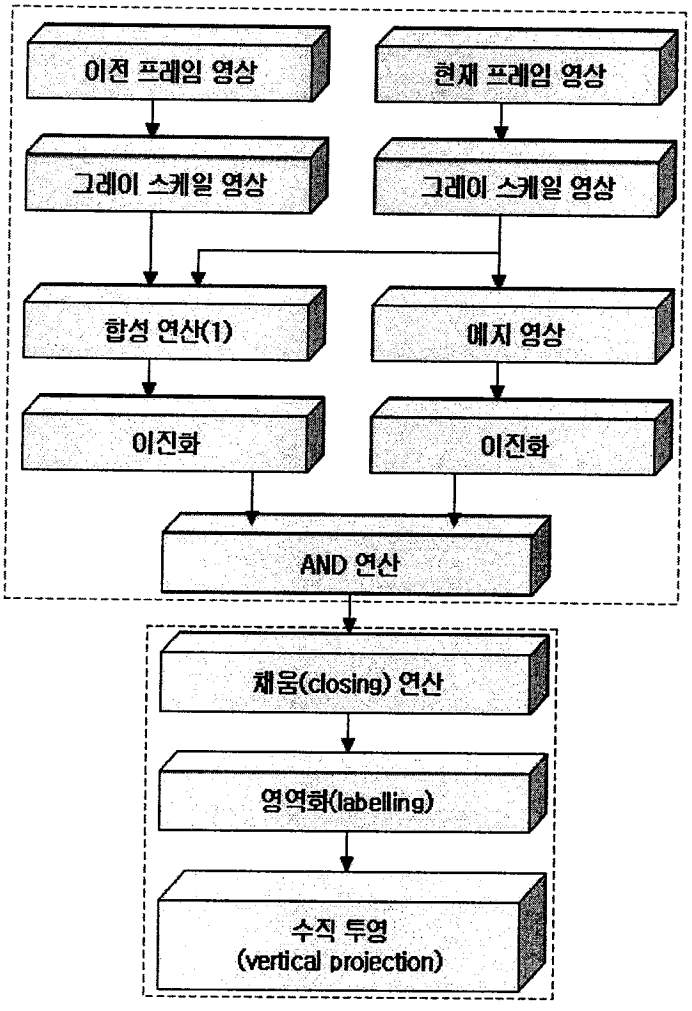
본 논문에서 제안하고자 하는 방법은 차영상 방법을 이용하며, 차영상 방법 중에서도 연속된 프레임에서의 차영상을 이용하고자 한다. 본 논문에서 제안하는 방법은 크게 보행자를 검출하기 위한 부분과 보행자를 추적하기 위한 부분으로 나눌 수 있다. [그림 3.1]은 제안하는 방법의 전체적인 순서도 이다.



[그림 3.1] 제안 방법의 순서도

3.1 보행자 검출 방법의 제안

본 논문에서 제안하는 보행자 검출방법은 크게 두 부분으로 나눌 수 있다. 첫 번째는 대략적인 움직이는 객체의 형태를 검출하고 이를 두 번째 부분에서 좀 더 확실하게 형태를 추출하여 보행자를 검출하게 된다. [그림 3.2]는 이 두 부분을 보여주고 있다.



[그림 3.2] 움직임 검출 단계의 순서도

[그림 3.2]를 개략적으로 살펴보면, 먼저 [그림 3.2]에서 보여주는 것처럼, 인접한 두 컬러 영상을 그레이 스케일로 변환 후 이를 다시 제안하는 합성 방법을 이용하여 변환하고 현재 프레임의 그레이 스케일 영상의 에지 영상을 AND 연산을 시킴으로써 객체의 대략적인 형태를 알아낼 수 있다.

움직이는 객체의 대략적인 형태를 채움 연산(closing), 영역화(labelling), 수직 투영(vertical projection)을 통해서 객체의 형태를 좀 더 정확하게 추출해 냄과 동시에 노이즈 제거 및 다중 객체의 분할을 하게 된다. 이를 좀 더 세부적으로 알아보도록 하겠다.

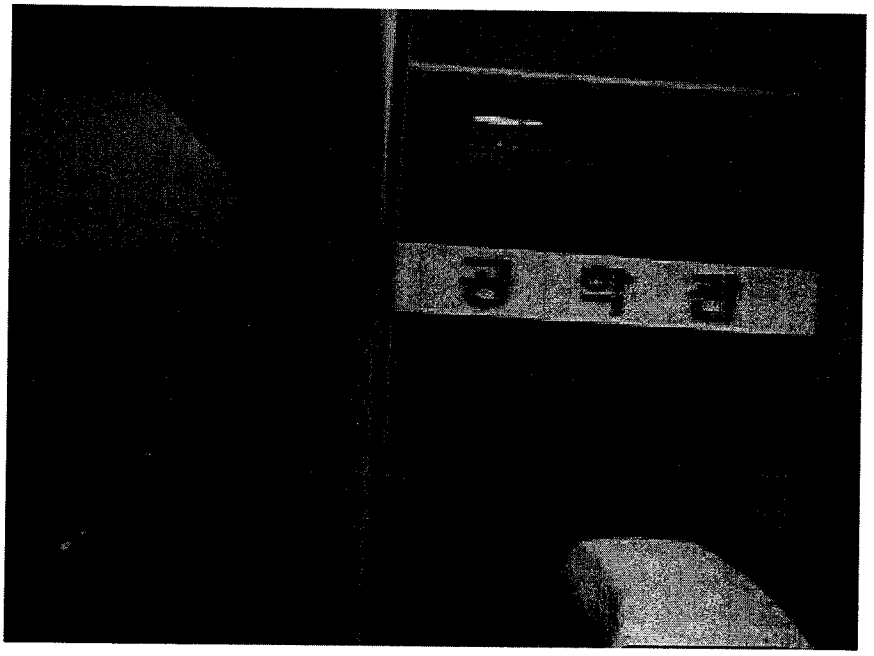
3.1.1 이전 프레임과 현재 프레임의 그레이 스케일 영상 변환

전체 영상을 카메라로부터 얻을 경우 아날로그 CCD카메라를 사용하려면 아날로그 신호를 디지털 신호로 변환하기 위한 프레임 그래버(Frame Grabber)를 사용해야 한다. 보통의 경우 이 프레임 그래버는 비트맵 형식의 컬러 이미지를 전송하지만 설정을 달리함으로써 그레이 이미지를 얻을 수 있다. 만약 사용하는 프레임 그래버가 그레이 이미지를 지원하고, 구현하는 시스템에서 컬러 이미지를 필요로 하지 않을 경우 이런 하드웨어 수준의 그레이 이미지를 사용하면 메모리 공간과 처리 속도를 더욱 높일 수 있다[22].

본 논문에서 사용하는 환경은 프레임 그래버를 사용하지 않고, CCD 카메라와 네트워크 기능이 있는 비디오 서버를 사용한다. 비디오 서버는 카메라로부터 얻은 영상을 네트워크에 연결되어 있는 영상 처리용 컴퓨터에 보내게 된다. 네트워크에 연결 되어 있는 컴퓨터 에서는 이를 입력으로

받아들이게 되는데, 입력 형식은 32bit의 RGB 칼라 영상이게 된다.

입력으로 받아들여진 32bit RGB 칼라 영상을 그레이 스케일로 변환하게 되는데, 이는 4 byte 비트맵 이미지를 1 byte로 줄이게 되므로 이는 메모리와 처리 속도의 향상을 얻을 수 있다. [그림 3.3]은 이전 프레임과 현재 프레임의 그레이 스케일 영상을 보여주는 그림이다.



[그림 3.3] 그레이 스케일 영상

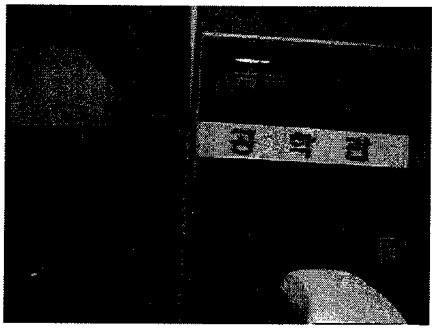
3.1.2 이전 프레임과 현재 프레임의 합성

그레이 스케일로 변환된 이전 프레임과 현재 프레임의 영상을 (식 3.1)과 같이 제안하는 합성 방법으로 배경을 제거하게 된다.

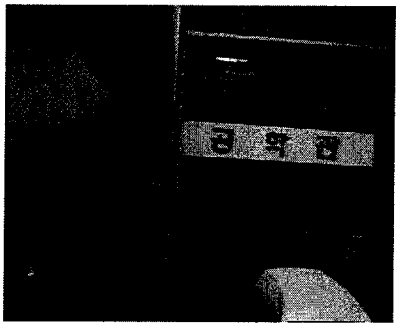
$$F(x,y) = (F_{t-1}(x,y) + F_t(x,y))/2 - F_{(t-1)}(x,y) \quad \text{(식 3.1)}$$

여기서 $F_{t-1}(x,y)$, $F_t(x,y)$은 (t-1) 과 t 시간에서의 프레임이고 (x,y)는 프레임 안에서의 픽셀의 위치 정보이다. 두 개의 이미지를 합성했으므로 2 로 나누어 주고 이를 다시 현재 프레임으로 빼 주게 된다. (식 3.1)의 결과로 나온 영상을 다시 이진화를 시킴으로써 배경과 움직임이 있는 객체의 대략적인 형태를 알아 낼 수 있다. 이때 이진화의 방법은 (식 3.2)와 같으며, 사용되는 임계값은 실험에 의하여 결정하였다.

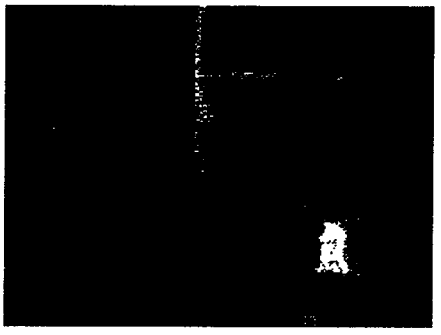
$$dx(x,y)\begin{cases}0 & \text{if}(\,F(x,y) \leqq threshold) \\ 1 & otherwise\end{cases} \qquad \text{(식 3.2)}$$



(a) 이전 영상



(b) 현재 영상



(c) 제안한 식의 의한 합성 영상

[그림 3.4] 이전 영상과 현재 영상의 제안한 식에 의한 합성

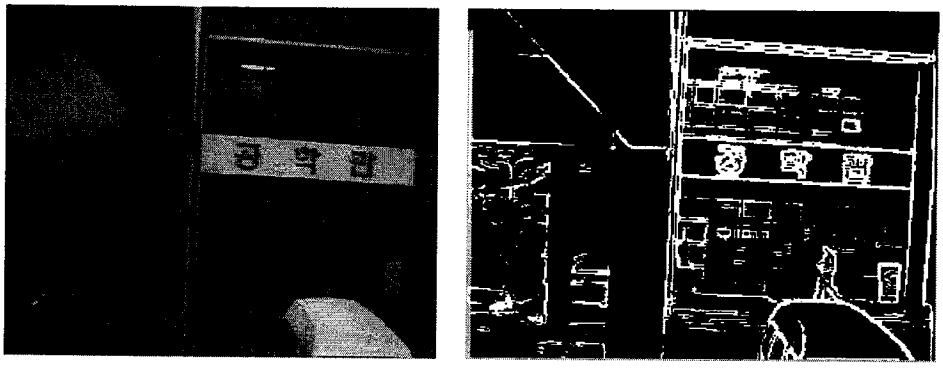
[그림 3.4]는 이전 프레임(a) 과 현재 프레임(b)의 영상을 보여 주고 본 논문에서 제안한 (식 3.2)의 결과에 이진화를 시킨 영상(c)의 그림이다.

[그림 3.4]에서 알 수 있듯이, 이전 프레임과 현재 프레임을 이용하여 제안한 방법으로 합성을 했을 경우 배경이 분리되고 움직임이 있는 부분만 검출하게 된다. 하지만 이것은 완전한 객체의 형태로 보기 어렵기 때문에 다음 단계로 현재 프레임의 에지 영상과 AND 연산을 하게 된다.

3.1.3 현재 프레임의 에지 영상

현재 프레임의 그레이 스케일 영상을 구하고 이를 이용한 에지영상은 Canny 마스크를 이용해서 에지 영상을 구하게 된다. 여기서 Canny 마스크의 특징은 먼저 가우시안 스무딩 필터를 사용하기 때문에 잡음을 제거하고 Sobel 마스크를 사용하여 에지를 구하는 것이다. Canny 마스크를 사용하는 이유는 대부분의 검출 마스크는 잡음에 대하여 매우 민감한 특성을 가지고 있어서 작은 잡음도 윤곽선으로 검출할 경우가 많지만 Canny 마스크는 잡음에 민감하지 않은 에지 검출 방법이기 때문이다. Canny 마스크를 적용한 영상을 다시 이진화 시킴으로써 물체의 형태를 좀 더 강하게 표현할 수 있다.

[그림 3.5]는 현재 프레임에 Canny 마스크를 적용하고 이진화를 시킨 영상의 예를 보인 그림이다.



(a) 현재 프레임 (b) Canny 방법에 의한 에지 검출

[그림 3.5] 현재 프레임과 Canny 방법에 의한 에지 검출

3.1.4 합성 영상과 에지 영상의 AND 연산

(식 3.1)을 간단히 하면 (식 3.3)의 결과를 보인다.

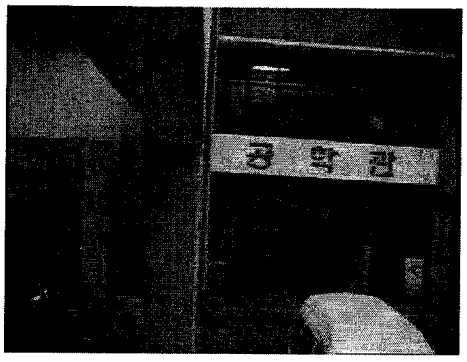
$$\frac{F_{t-1}(x,y) - F_t(x,y)}{2} \quad \text{(식 3.3)}$$

(식 3.3)은 이론적으로 (식 3.4)와 비슷한 결과를 보일 수 있다.

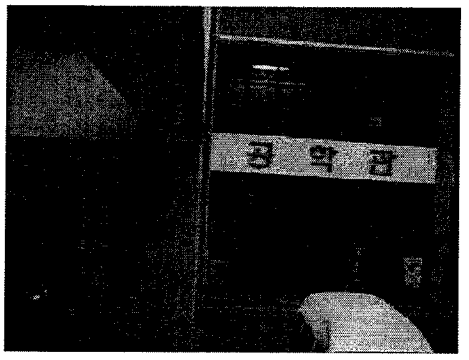
$$F_{t-1}(x,y) - F_t(x,y) \quad \text{(식3.4)}$$

하지만 실질적으로 (식 3.1)과 (식 3.4)의 결과를 나온 영상과 이진화된 에지 영상을 AND 연산을 하게 되면 [그림 3.6]과 같이 다르게 나타난다. 이는 현재 프레임에서 변화된 부분이 많을수록 많은 부분을 배경에서 제외시킬 수 있으므로 배경만 제거하게 되면 좀 더 강건한 객체의 형태를 추출할 수 있게 되기 때문이며 이를 다시 에지 이미지와 AND 연산을 함으로써 물체의 형태를 추출할 수가 있다. 또한 (식 3.4)를 적용할 경우 하

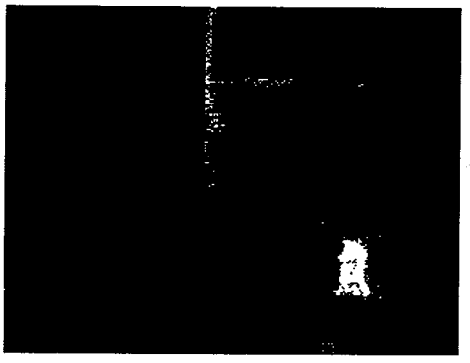
나의 객체임에도 불구하고 두 개로 분리되는 경우가 발생하지만 (식 3.1)을 사용하는 제안한 방법을 적용할 경우 이런 문제를 어느 정도 해결할 수 있음을 알 수 있다.



(a) 이전 프레임

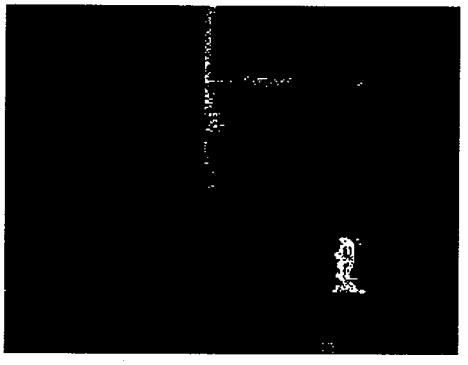


(b) 현재 프레임

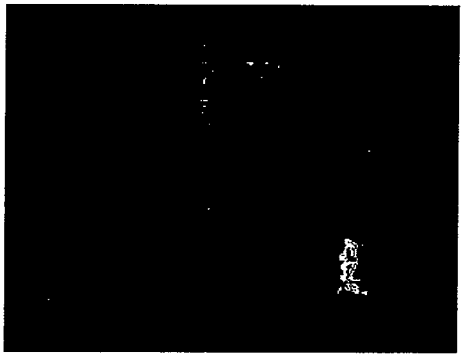


(c) (식 3.1)을 적용

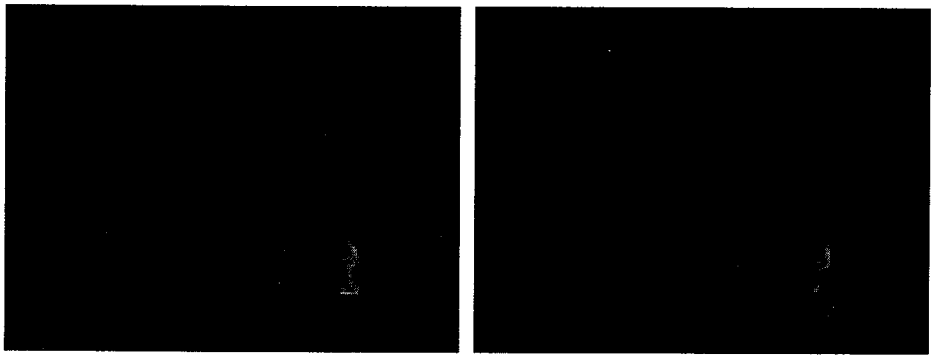
(d) (식 3.4)를 적용



(e) (식 3.1) 후 AND 연산 결과



(f) (식 3.4) 후 AND 연산 결과



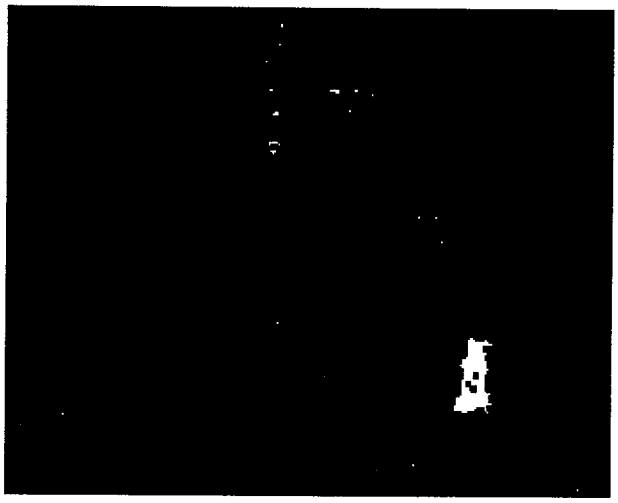
(g) (식 3.1) 후 최종 결과　　(h) (식 3.4) 후 최종 결과

[그림 3.6] (식 3.1)과 (식 3.4)의 비교

3.1.5 채움 연산(closing)

지금까지의 방법으로 움직이는 객체의 대략적인 형태를 추출할 수 있었다. 하지만 이대로 보행자의 영역을 검출하기에는 많은 문제가 있다. 잡음이 이에 속하며 이를 해결하는 필터들이 많이 있다. 하지만 본 논문에서는 채움 연산(closing)을 사용하였다.

채움 연산은 팽창 연산과 축소 연산을 수행하는 것으로 팽창 연산으로 인한 확장이 먼저 일어나므로 작은 구멍들은 메워지고, 축소 작용 때에 원 영상의 크기로 복원될 수 있다. 채움 연산은 작은 구멍들이 메워지므로 노이즈 제거와 더불어 하나의 객체임에도 불구하고 명암의 차이에 의해서 두 개로 분리된 객체를 하나로 연결할 수 있게 한다. [그림 3.7]은 채움 연산을 적용한 예의 그림이다. 그림에서 알 수 있듯이 작은 구멍들이 매워졌음을 알 수 있다.

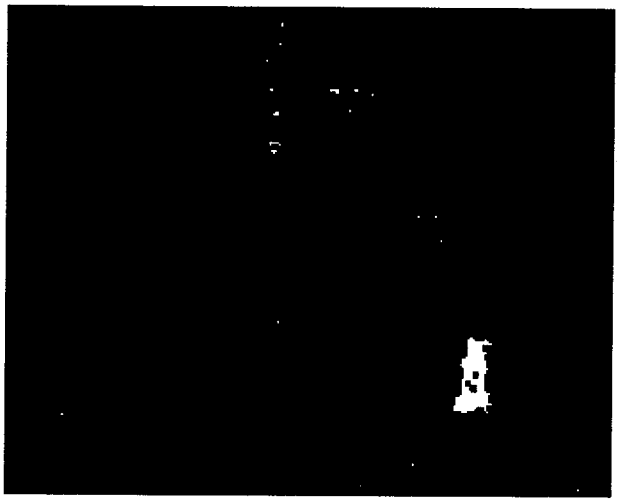


[그림 3.7] 채움 연산(closing)

3.1.6 영역화(labelling)

채움 연산을 통해 작은 구멍이 매워지고 분리되었던 객체가 합쳐지면 객체의 영역화를 하게 된다. 이동 물체의 검출은 되었지만 아직 객체로 보기 힘들기 때문에 이를 영역화 방법을 통해서 연결되어 있는 픽셀들을 하나의 객체로 묶어 주게 된다. 영역화는 fire grass 라는 방법으로 수행되어 지는데, 이는 주변에 연결되어 있는 픽셀들을 조사하여 같은 값으로 번호를 지정하게 된다. [그림 3.8]은 영역화를 진행한 후의 결과 영상이다. 영역화 결과는 채움 연산을 적용한 결과 영상과 큰 차이는 없다.

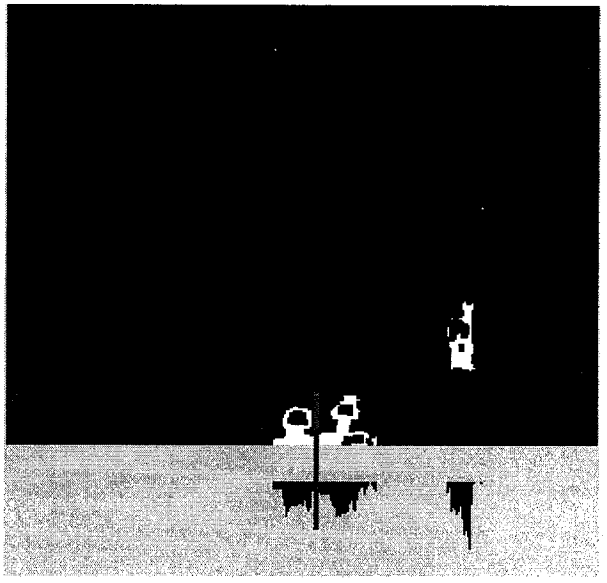
하지만 영역화를 통해서 연결되어 있는 화소를 하나로 묶어 주게 되어서 하나의 객체로 인식할 수 있으며, 그로 인해서 보행자의 경계 및 보행자의 중심 값을 구해낼 수 있다.



[그림 3.8] 영역화의 결과

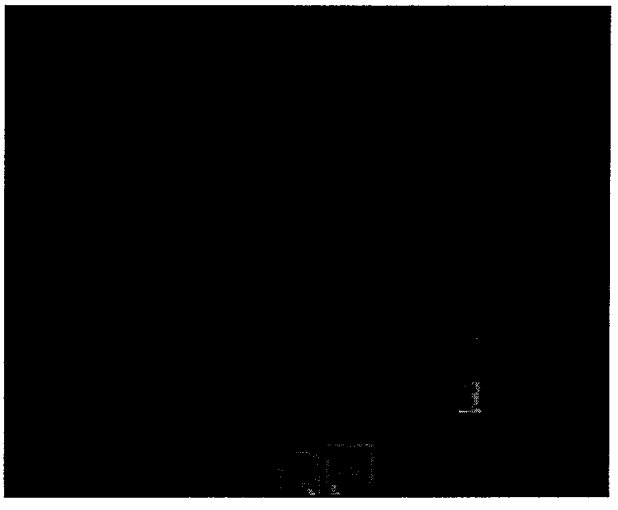
3.1.7 수직 투영(vertical projection)

영역화(labelling) 후의 묶여진 픽셀들의 집합을 본 논문에서는 하나의 객체로 보고 있으며, 또한 한 명의 보행자로 보고 있다. 하지만 하나의 객체가 가로, 세로의 비를 이용해서 하나의 객체가 한 명의 보행자로 생각되지 않을 때에는 [그림 3.9]와 같이 수직 투영을 하게 된다. 수직 투영은 수직 히스토그램을 이용하여 실험을 통하여 정한 임계값 보다 작을 경우 객체를 분리하게 된다.



[그림 3.9] 수직 투영의 결과

[그림 3.10]은 이렇게 수직 투영을 통해서 검출된 객체의 결과를 보인 그림이다.

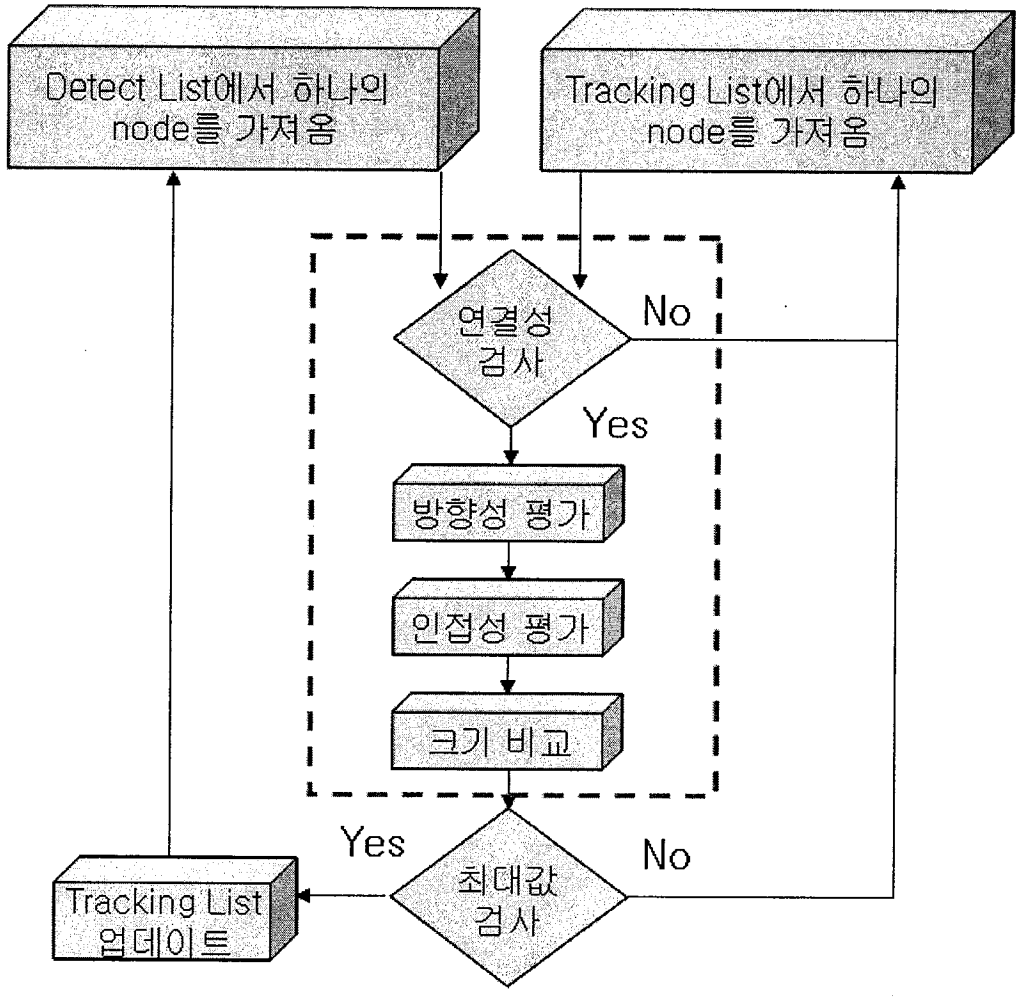


[그림 3.10] 최종 결과 영상

3.2 보행자 추적 방법의 제안

움직임 검출 단계에서 검출되어진 객체는 현재 프레임에서 검출되어진 리스트에 삽입하게 된다. 그리고 이전 프레임에서 추적을 하고 있던 리스트와 유사도 비교를 통해서 리스트의 업데이트를 하여 객체의 추적을 수행하게 된다.

[그림 3.11]은 제안하는 방법을 그림으로 도식화 한 것이다.



[그림 3.11] 제안 추적 방법

[그림 3.11]에서 보듯이 추적되고 있던 객체의 리스트에서 하나의 노드와 현재 프레임에서 검출된 객체의 리스트의 노드의 유사도 비교하여 유사도가 최대값이 되는 노드를 업데이트 시켜 객체를 추적할 수 있게 한다.

추적되고 있던 객체의 리스트란 이전 프레임까지 계속해서 추적되고 있던 객체 노드를 말한다. 초기 노드가 하나도 존재하지 않을 경우는 현재 프레임에서 검출되는 모든 객체들이 모두 리스트에 추가되게 된다. 그리고 현재 프레임에서 검출된 객체의 리스트는 현재 프레임에서 제안하는 방법에 의해 움직임이 검출된 모든 객체들의 리스트를 말하고, 이를 이전 프레임의 객체들의 리스트와 유사도를 비교하게 된다.

3.2.1 유사도 평가

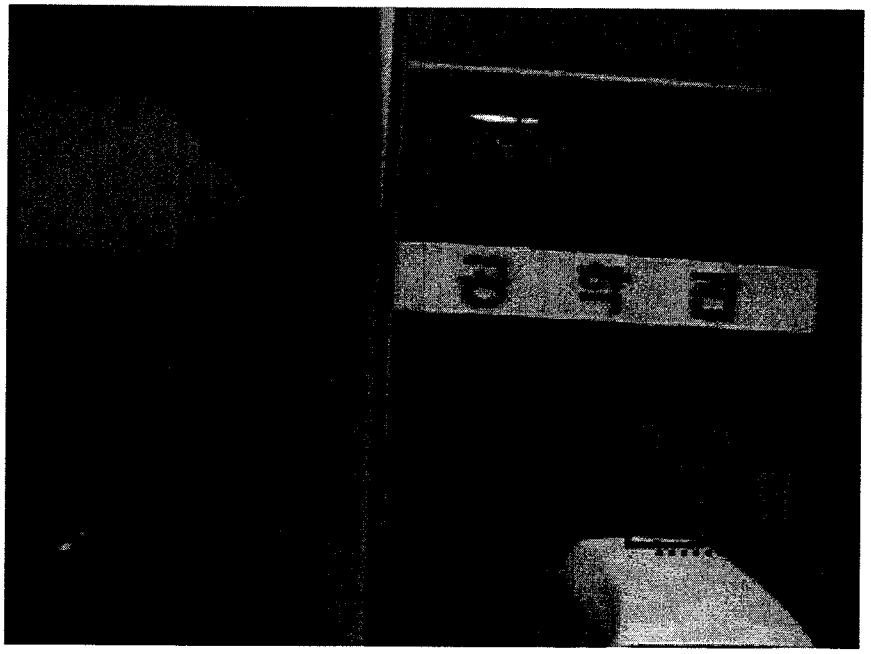
유사도란 이전프레임에서 추적되던 객체와 현재 프레임에서 검출된 객체간의 유사한 정도를 평가하는 것으로 본 논문에서 제안하는 유사도를 평가하는 방법은 [그림 3.11]의 점선 부분과 같이 연결성, 인접성, 방향성, 검출된 객체의 크기를 점수로 평가하며, 이들 평가요소의 합으로서 각각의 리스트의 노드와 비교를 통해서 최대값을 갖는 리스트의 노드와 업데이트를 하게 된다.

3.2.1.1 연결성

연결성은 객체의 연속성을 고려한다. 즉, 이동하는 보행자는 갑자기 순간이동을 하지 않기 때문에 이전 프레임에서 검출된 객체는 현재 프레임에서 똑같이 검출이 된다면 그 이동 거리가 크지 않다. 그리고 검출된

객체는 경계를 가지게 되는데 언제나 검출된 경계 안에 포함되게 된다. 만약 포함하지 않는다면 이동속도가 빠른 다른 객체로 인식을 하게 되므로 본 논문에서 대상으로 하는 보행자가 아니기 때문에 이는 추적 대상에서 제외하게 된다.

[그림 3.12]는 연결성을 보여주는 그림으로서 점선은 이전 프레임에서 검출된 객체의 경계를 나타내며, 실선은 현재 프레임에서 검출된 객체의 경계를 나타낸다.



[그림 3.12] 이전 프레임과의 연결성

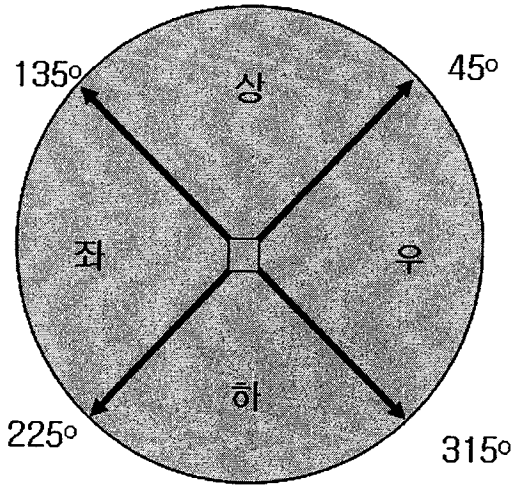
3.2.1.2 방향성

보행자는 언제나 가려던 방향으로 가는 성질이 있다. 이는 본 논문에서 제안하는 추적 방법에서 사용할 수 있는 근거가 된다. 본 논문에서는

크게 상, 하, 좌, 우 이렇게 4 방향으로 나누어서 방향성을 검사하게 된다.

하지만 보행자가 갑자기 방향을 바꿀 때도 존재하므로 이를 절대적인 추적평가 방법으로 사용하지 않고 이를 점수로서 반영을 하게 된다. [그림 3.13]은 방향성 검사에 사용되는 상태를 표현하는 그림이다.

[그림 3.13]에서와 같이 상, 하, 좌, 우의 방향으로 진행방향을 정하고 이전 프레임에서 검출되었던 보행자가 계속 같은 방향으로 진행할 경우 점수를 증가시키게 된다. 이때의 점수 증가식은 (식 3.5)와 같다.



[그림 3.13] 방향성의 상태

$$score = score + n \quad \text{(식 3.5)}$$

(식 3.5)에서 n은 방향성 점수 증가를 위한 상수로서 본 논문에서는 실험을 통하여 5 로 고정하였다.

3.2.1.3 인접성

인접성은 이전 프레임에서 추적되고 있던 보행자와 현재 프레임에서 검출된 보행자 사이의 거리를 평가하는 것이다. 보행자는 거의 등속도로 움직인다. 따라서 (t-2), (t-1), t 시간에서의 프레임에서 같은 객체가 검출되었고, 검출된 객체가 보행자라면 (t-2), (t-1) 시간의 프레임 사이에서 검출된 보행자의 중심 값의 거리가 (t-1), t 시간의 프레임에서 검출된 보행자의 중심 값의 거리가 비슷할 것이기 때문에 이 또한 객체를 추적하기 위한 하나의 요소로서 사용할 수 있다. 따라서 본 논문에서 이를 활용하여 점수로 평가를 하게 된다. 이때의 점수 증가식은 (식 3.5)와 같고, 상수 n은 실험을 통하여 방향성보다는 좀 더 가중치를 두어서 7 로 고정시켰다.

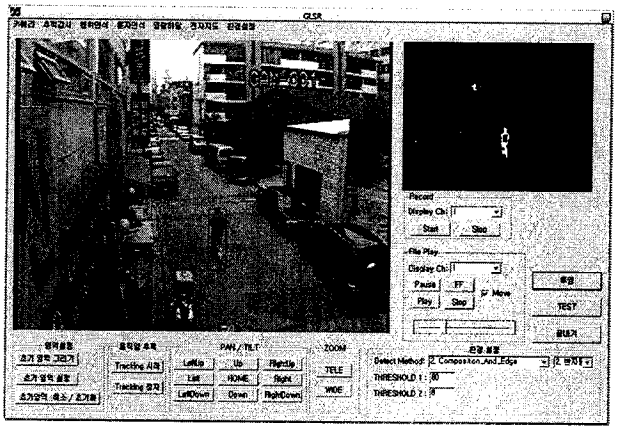
3.2.1.4 크기 비교

이전 프레임에서 검출된 객체의 가로, 세로 비율은 현재 프레임에서도 거의 비슷한 비율로 나타날 것이다. 하지만 검출된 보행자는 가로, 세로의 비율이 비슷할 확률이 높기 때문에 크기 비교에는 점수평가에서 가중치를 좀 더 낮게 책정하게 된다. 이때의 증가식은 (식 3.5)와 같고, 상수 값 n은 가장 가중치를 낮게 책정하여 2 로 고정 하였다.

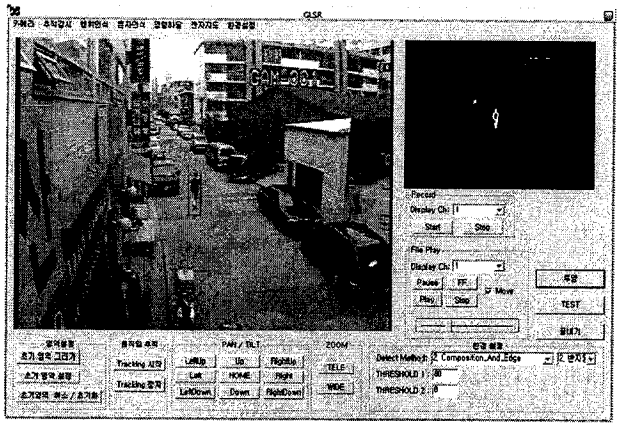
연결성, 방향성 인접성, 크기비율 비교 등의 평가 요소로 점수를 매기고 이를 모두 합산하여 유사도로서 평가하고, 종합적인 판단으로 추적을 하게 된다. 유사도 평가는 이전프레임에서 추적되고 있던 보행자의 노드 하나와 현재 프레임에서 검출된 보행자의 노드 모두 유사도를 평가 하여서 유사도가 최대값이 되는 노드를 이전 프레임까지 추적되고 있던 보

행자라고 판단을 하고, 추적 리스트에 업데이트를 시켜주게 된다. 이렇게 모든 추적 리스트의 노드들을 현재 프레임의 검출된 객체의 리스트와 유사도를 평가하여 이전 프레임까지 추적이 되고 있던 객체들이 업데이트 되면서 보행자의 추적이 이루어지게 된다.

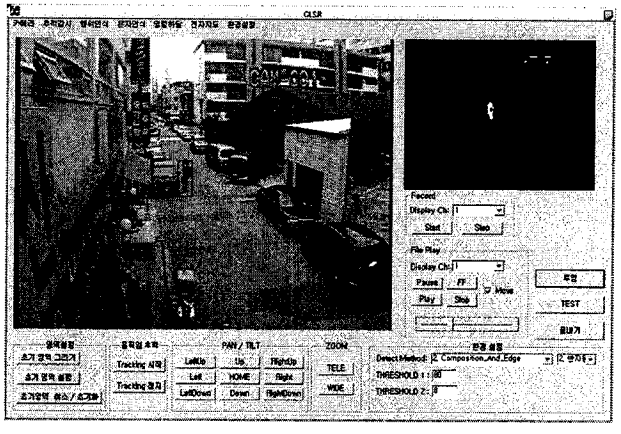
[그림 3.14] 보행자가 검출되어 추적되는 결과를 보인 그림이다. [그림 3.14] (a)와 [그림 3.14] (b)에서 보행자가 추적되는 모습을 보인다. 하지만 [그림 3.14] (c)에서는 두 명의 보행자이지만 너무 밀접하게 연결되어 있어 한 명의 보행자로 검출되었다. 제안된 검출방법의 수직 투영만으로는 이를 해결 하지 못하였다. 하지만 [그림 3.14] (d)와 같이 이후 두 명의 보행자가 분리되었을 때 다시 원래의 추적되고 있던 보행자를 따라 가는 모습을 볼 수 있다.



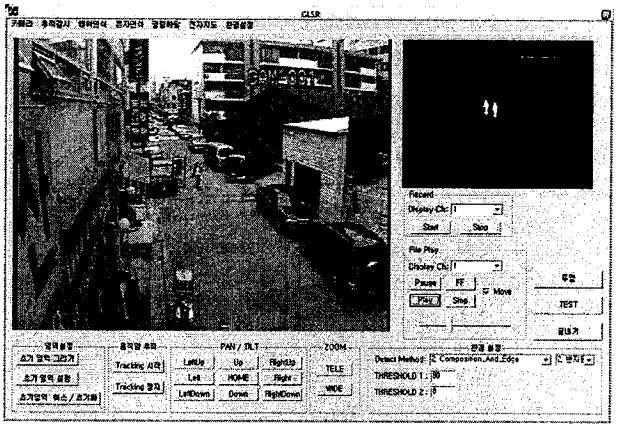
(a) 보행자 추적 1



(b) 보행자 추적 2



(c) 보행자 추적 3



(d) 보행자 추적 4

[그림 3.14] 보행자 추적의 결과

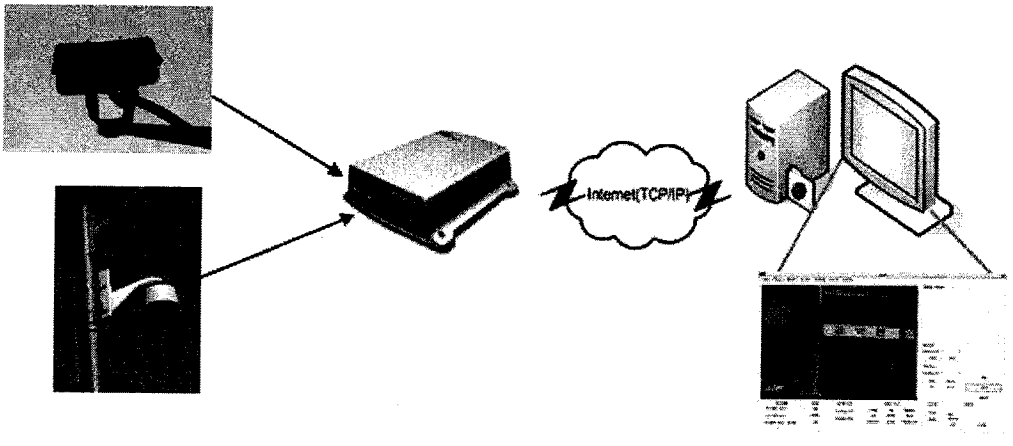
제 4장 실험 및 결과 분석

4.1 실험 방법

본 논문에서 보행자를 검출하고 추적하기 위한 실험 환경과 실험 방법에 대해서 설명하겠다. 실험 환경 및 방법은 다음과 같으며, 이를 통하여 제안한 방법들의 성능을 평가한다.

4.1.1 실험 환경

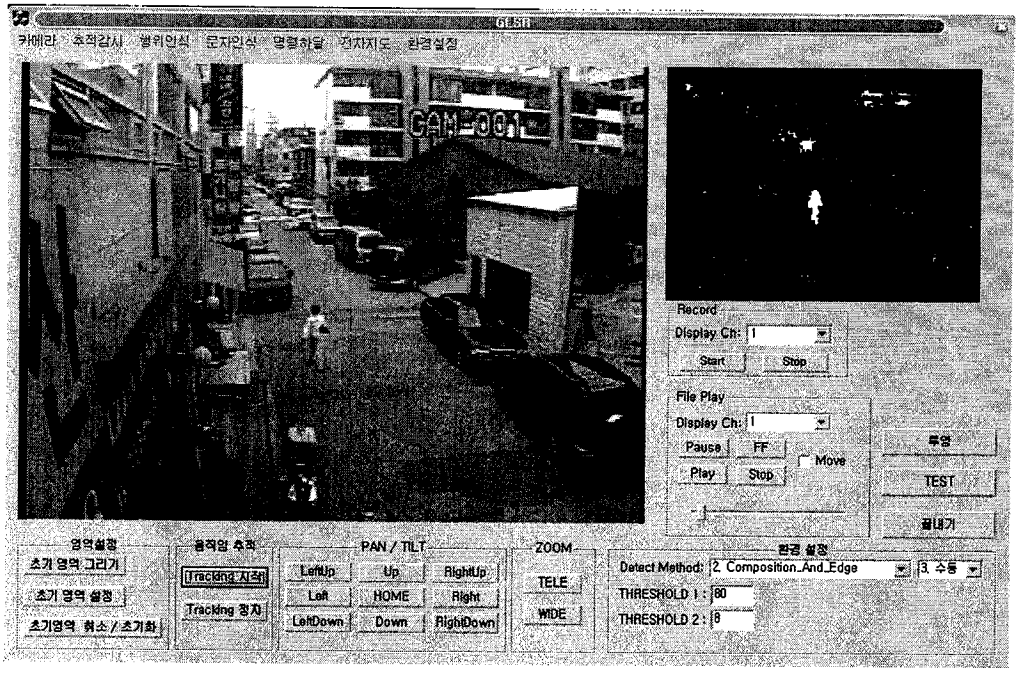
실험 환경은 [그림 4.1]과 같은 구조로 한성대학교 교내 건물 사이의 골목과 운동장, 그리고 시내의 일반 보도(서울, 구리)에서 이루어 졌다.



[그림 4.1] 실험 환경의 구조도

[그림 4.1]은 움직이는 객체의 검출 및 추적을 위한 실험 및 성능 평가를 위해 사용된 시스템 구성도이다. 사용된 컴퓨터의 사양은 [표 4.1]과 같으며, 네트워크 서버는 [표 4.2]의 사양 및 특징을 갖는다. 그리고 실

험에서 사용된 CCD카메라의 사양 및 특징은 [표 4.3]에 나타내었고, 실험 환경은 [표 4.4]에 나타내었다. 또한 [그림 4.2]는 실험을 위하여 작성된 프로그램의 사용자 인터페이스이다.



[그림 4.2] 실험 프로그램 사용자 인터페이스

실험에 사용된 시스템은 [표 4.1]과 같으며 시스템의 움직임 검출 및 추적의 속도 비교를 위해서 상대적으로 고사양인 컴퓨터와 저사양인 컴퓨터, 이렇게 총 두 대의 컴퓨터가 사용되었다.

[표 4.1] 실험에 사용된 컴퓨터 사양

	PC 1	PC 2
CPU	Intel Pentium 4 (2.8GHz)	Intel Pentium 4 (1.8GHz)
RAM	1G (DDR Ram)	512 (DDR Ram)
OS	Windows XP	Windows XP

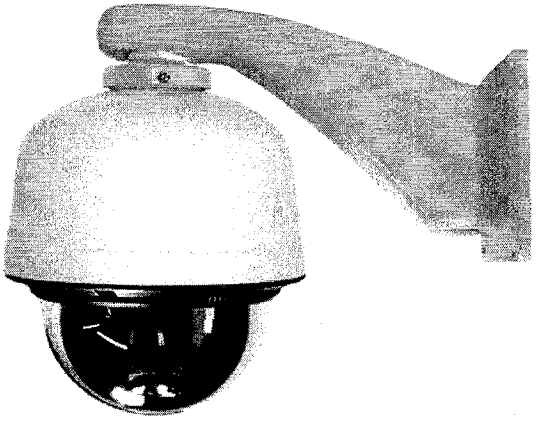
[그림 4.3]은 실험에 사용된 네트워크 비디오 서버의 사진이다. 네트워크 비디오 서버는 BRANS사의 BNT100 제품을 사용하였고 [표 4.2]에 제품의 특징을 나타내었다.

[그림 4.3] 실험에 사용된 네트워크 비디오 서버

[표 4.2] 네트워크 비디오 서버의 사양 및 특징

사양 및 특징	BNT100
	Embedded Linux
	30(25)FPS @ 720X480(720X576)
	Hardware MPEG4 Video Codec
	ADPCM Audio Codec (Full duplex & Two-way communication)
	Support Fixed & Dynamic IP
	Remote Relay & PTZ Control
	Cooperation with Local Storage Device : USB external HDD,
	CF Memory, 2.5" internal HDD
	Various Recording Mode : Continuous, Motion, Sensor, Video loss, Network-Fail
	Dual search mode : Time-map, file list
	Others : Dual Streaming, 1CH TV Out

실험에 사용된 CCD 카메라는 [그림 4.4]와 [표 4.3]과 같은 사양 및 특징을 나타낸다.



[그림 4.4] 실험에 사용된 CCD 카메라

[표 4.3] 실험에 사용된 CCD 카메라의 사양 및 특징

사양 및 특징	1/4 " SONY Super HAD CCD
	최대 220배 줌렌즈 (22x Optical,10x Digital)일체형
	360° Continuous, 92° Vertical
	Day&Night 기능
	128개 Preset point 지정

실험은 네트워크 서버의 기능 중에 하나인 파일 저장 기능을 이용하여 한성대학교의 교내 건물 및 운동장과 시내 일반 보도(서울, 구리)에서 영상 파일을 저장하여 실험 컴퓨터에서 동작시켜 실험을 하였다. 이는 네트워크 성능으로 인한 영향을 줄이기 위해서이다. 영상의 크기는 320 × 240으로 제한하였으며, 또한 보행자는 어느 일정한 한계 내의 속도로 이동

한다는 가정 하에 영상은 초당 4 프레임으로 처리하게 되며, 실험을 위해 작성된 프로그램은 Microsoft 사의 Visual C++ 6.0으로 작성되었다.

[표 4.4] 실험 환경

실험 장소	한성대학교 교내 건물 및 운동장 시내 일반 보도(서울, 구리)
영상 크기	320 × 240
Frame Rate	초당 4 frames (4 frames / second)
Tool	Visual C++로 작성

4.1.2 실험 방법

실험은 크게 세 부분으로 나누어서 평가하였다. 실험 방법은 상대적으로 고사양인 컴퓨터와 저사양인 컴퓨터에서의 1 프레임의 처리시간과 움직임 검출에서 나타날 수 있는 형태의 불확실성으로 인한 오류율을 두 가지로 나누어서 실험하고, 움직임 추적에서 사용되는 제안 방법의 추적 오류율을 실험 요소로 본다. 여기서 지칭하는 기존 방법이란 이전 프레임과 현재 프레임의 차영상만을 이용한 방법을 말한다.

실험 1 은 컴퓨터의 성능에 의한 제안 방법의 처리 속도 비교를 통해서 제안 방법의 성능의 우수성을 검증하기로 하며, 실험은 상대적으로 고사양인 컴퓨터와 저사양인 컴퓨터에서의 1 프레임의 처리 시간을 총 3 개의 실험 데이터를 이용하여 실험을 하였다.

실험 2 는 제안 방법의 보행자 검출 성능을 비교하기 위하여 제안한 방법과 기존 방법인 연속된 동영상에서 이전 프레임과 현재 프레임의 차만을 이용한 방법에서의 보행자의 검출 수를 비교한다. 실험은 실험 데이

터 파일을 3 개를 가지고 30 초간 이동 객체의 검출을 실험하였다.

이 실험에서는 검출된 객체의 형태의 불확실성으로 인하여 하나의 객체임에도 불구하고 두 개로 객체로 인식하는 문제를 오류로 보게 된다. 이 오류를 제안된 방법과 기존 방법의 비교 요소로 보고 오류를 비교하여 제안 방법의 우수성을 검증한다.

실험 3 은 제안 방법의 움직임 추적의 성능을 비교한다. 이는 추적되는 객체를 잃어버리거나, 다른 객체를 추적하는 것을 오류로 보게 된다.

4.2 실험 결과 및 분석

4.2.1 실험 1 의 결과 및 분석

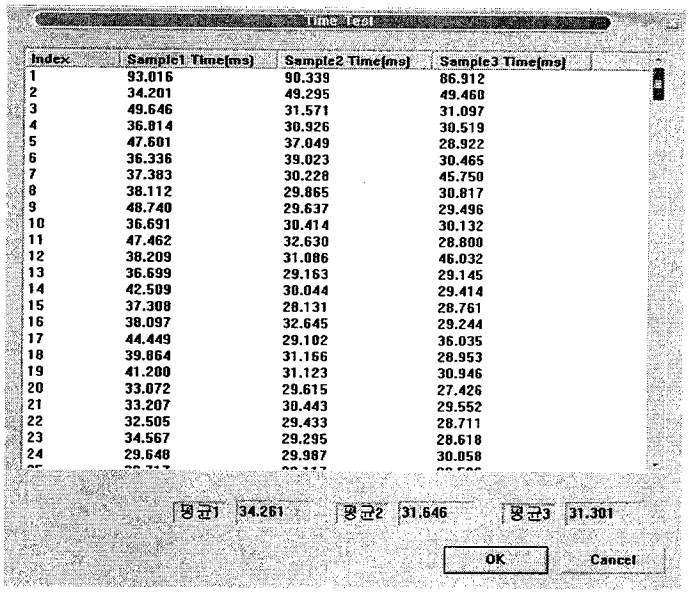
실험 1 에 대한 실험 데이터에 대한 설명을 [표 4.5]에 나타내었다. 실험은 총 3 개의 실험 데이터를 사용하였다. 실험 데이터는 1 분 동안에 움직이는 보행자의 검출 개수로 구분하였다.

[표 4.5] 수행 시간 실험의 데이터

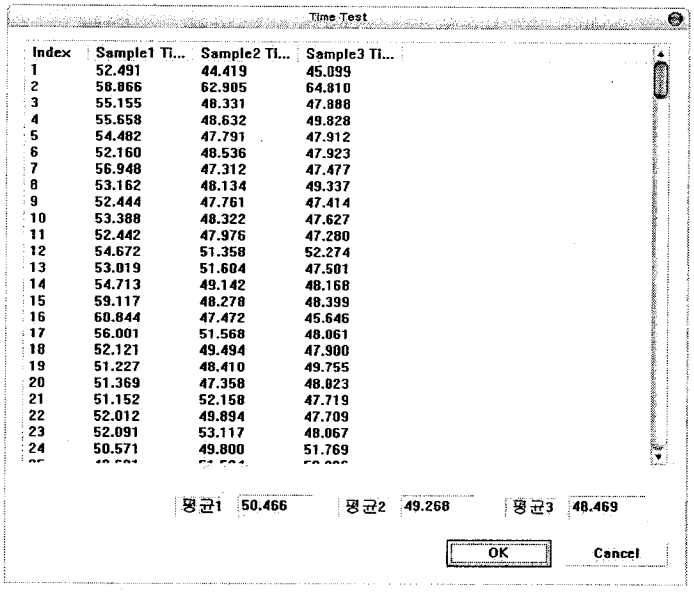
Data 이름	보행자 수	총 Frame 수 (초 × 초당 Frame 수)
Sample Data 1	6명 이상	240 Frame (60 × 4)
Sample Data 2	3 ~ 5 명	240 Frame (60 × 4)
Sample Data 3	1 ~ 2 명	240 Frame (60 × 4)

[그림 4.5]는 고사양인 컴퓨터의 실험 결과 화면으로서 첫 번째 칼럼은 Sample Data 1, 두 번째 칼럼은 Sample Data 2, 세 번째 칼럼은

Sample Data 3이다. [그림 4.5]의 하단의 평균값이 검출 및 추적 시간의 평균값이다. 보행자 수가 많을수록 근소하게나마 시간이 증가함을 알 수 있다. 하지만 그 차이가 경미하기 때문에 다수의 보행자 검출에서도 좋은 성능을 발휘함을 알 수 있다. [그림 4.6]은 저사양인 컴퓨터의 실험 결과 화면으로서 [그림 4.5]와 평균 시간을 비교했을 때 약간의 차이가 나는 것을 알 수 있었다. 이유는 검출 방법에서 사용되는 영역화(labelling) 알고리즘이 재귀적으로 연결된 모든 픽셀을 찾기 때문인데, 이 방법을 다른 방법으로 대체할 경우 성능의 향상이 가능할 것으로 생각된다.



[그림 4.5] 고사양인 컴퓨터에서의 수행 시간 결과



[그림 4.6] 저사양인 컴퓨터에서의 수행 시간 결과

4.2.2 실험 2 의 결과 및 분석

실험 2 에 대한 실험 데이터에 대한 설명을 [표 4.6]에 나타내었다. 실험은 총 3 개의 실험 데이터를 사용하였다. 실험 데이터는 30 초 동안의 보행자의 검출 개수로 구분하였다. 이때 검출되는 객체의 형태 불확실성으로 인하여 발생하는 문제, 즉 객체가 하나임에도 불구하고 두 개로 인식되는 문제를 오류로 본다.

비교 대상은 이전 프레임과 현재 프레임의 차영상만을 이용하여 객체 검출을 하는 기존 방법으로 하며, Object Num 은 실험데이터를 통틀어 나타나는 객체의 수이며, Frame Num 은 실험에 사용된 프레임의 수이다. 제안방법과 기존방법의 object 검출 수는 제안된 방법과 기존의 방법으로 실험하였을 때, 실험 데이터에서 검출된 객체의 수를 나타낸다.

[표 4.6] 실험 2의 실험 데이터

Data 이름	Object Num	Frame Num	제안 방법의 object 검출 수	기존 방법의 object 검출 수	제안 방법의 오류율	기존 방법의 오류율
Sample Data 4	768	120	775	796	0.1(%)	3.6(%)
Sample Data 5	432	120	437	455	0.1(%)	5.3(%)
Sample Data 6	128	120	128	132	0(%)	3.1(%)

$$\text{객체검출 오류율} = \frac{|\text{검출된 } object \text{ 수} - object \text{ 수}|}{object \text{ 수}} \times 100 \quad \text{(식 4.1)}$$

[표 4.6]과 같이 제안된 방법의 오류율이 기존 방법의 오류율보다 매우 낮음을 볼 수 있다. 따라서 본 제안 방법이 기존 방법보다 객체 검출의 성능은 우수하다고 할 수 있겠다.

4.2.3 실험 3의 결과 및 분석

단일 보행자일 경우 검출이 잘 되었다면 추적 또한 잘 될 수 있다. 하지만 다수의 보행자일 경우 단일 보행자 보다는 많은 처리가 필요하며 본 논문에서 제안하는 방법으로 추적을 했을 때, 객체의 추적을 놓치거나 다른 객체를 추적하게 된다면 오류로 보고 오류율을 조사하였다.

실험은 3 개의 실험 데이터 파일로 실험을 진행하였으며, 3 개의 데이터 파일은 각각 30 초의 영상데이터가 저장되어 있다. [표 4.7]은 각각의 파일에 대한 기술하였다. Sample Data 4 은 3 - 4 명의 보행자가 일정한 패턴이 없이 걸어갈 때, Sample Data 5 는 1 - 2 명의 보행자가 맞은편에서 걸어오면서 교차할 때, Sample Data 6 은 여러 명의 보행자가 일정한

패턴이 없이 움직일 경우의 3 개의 파일이 저장되어 있으며, 이 중 한 사람의 보행자만을 추적하게 되며 이때 발생하는 추적을 놓치거나, 잘못 추적하는 오류를 계산하게 된다.

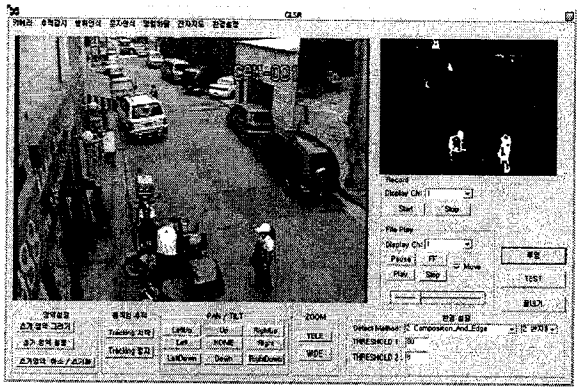
[표 4.7] 실험 4의 데이터 파일

Data 이름	총 frame 수	각 프레임 마다 나타나는 보행자 수
Sample Data 4	120 frame(30 × 4)	6 - 7
Sample Data 5	120 frame(30 × 4)	3 - 4
Sample Data 6	120 frame(30 × 4)	1 - 2

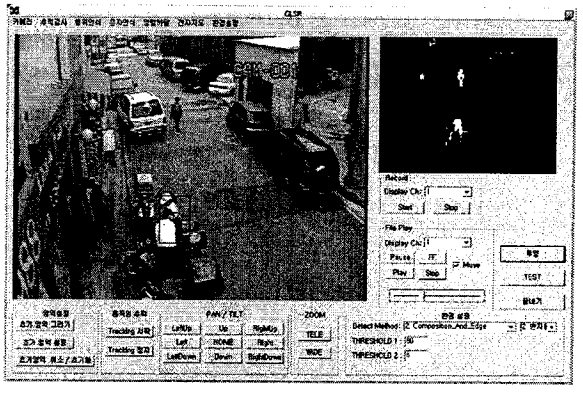
[표 4.8] 검출이 잘된 실험 파일의 추적에 관한 결과

Data 이름	보행자 수	추적을 놓치는 경우	다른 보행자를 추적하는 경우
Sample Data 4	6 - 7	0	0
Sample Data 5	3 - 4	0	0
Sample Data 6	1 - 2	0	0

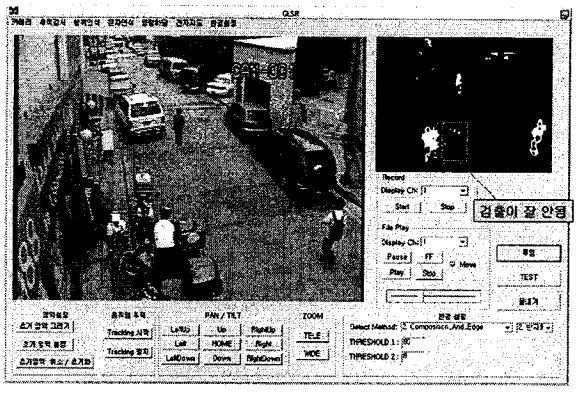
[표 4.8]과 같이 제안한 방법으로 추적을 하였을 경우, 추적을 놓치거나 다른 보행자를 추적하지 않음을 알 수 있었다. 하지만 이 실험 파일은 움직임 검출이 잘 되었을 때의 실험 데이터를 사용하였다. 만약 움직임 검출이 잘 되지 않는다면, 추적에서도 [그림 4.7]과 같이 상당한 오차가 발생하였다. 이 결과로 알 수 있듯이 움직임 추적에 있어서 선행되어야 하면서도 중요한 문제는 움직임 검출임을 알 수 있었다.



(a) 검출의 실패로 인한 추적 오류 1



(b) 검출의 실패로 인한 추적 오류 2



(c)검출의 실패로 인한 추적 오류 3

[그림 4.7] 검출의 실패로 인한 추적 오류의 결과

이상적으로는 모든 검출이 잘되는 것이 좋겠지만, 제안된 방법에도 문제점이 존재한다. 모든 환경에 적용되어지는 것이 아니라 [표 4.9]와 같은 경우에는 검출의 실패로 인하여서 추적이 안 되는 문제점도 존재한다.

[표 4.9] 제안한 방법으로 검출 및 추적이 잘 안 되는 경우

제안한 방법으로도 검출이 잘 되지 않는 상황
1. 여러 명이 이동할 경우 수직 투영만으로도 분리해 낼 수 없을 정도로 붙어 있을 경우
2. 배경색과 비슷한 색의 옷을 입은 보행자

제 5장 결론 및 향후 연구

5.1 결론

본 논문에서는 기존의 연속된 동영상에서 이전 프레임과 현재 프레임과의 차영상만을 이용할 경우 발생할 수 있는 형태의 불확실성으로 인한 문제를 해결하고자, 입력으로 받은 칼라 영상을 그레이 스케일로 변환하고 제안하는 합성 식을 이용하여 배경을 제거하였으며, 이를 다시 에지 영상과의 AND 연산을 통하여 이동 객체의 검출을 하였다. 또한 보행자 추적을 위하여 검출된 객체를 연결성, 인접성, 방향성, 크기 비율의 비교 등을 통해서 보행자 추적에 대한 방법을 제안하였다.

본 논문에서 제안된 보행자 검출 방법과 추적 방법은 세 부분으로 나누어 실험하였다. 첫 번째 실험은 상대적으로 고사양인 컴퓨터와 저성능인 컴퓨터 사이의 프레임당 처리시간을 비교하였고, 두 번째 실험은 보행자 검출수를 기존의 차영상만을 이용했을 때와 비교하였으며, 세 번째 실험은 제안된 추적 방법이 추적을 보행자를 추적하는 오차율을 실험하였다.

실험에서는 다수의 보행자를 검출하여도 처리시간의 증가는 있었지만 그 증가가 극히 작음을 볼 수 있었고, 기존 방법과 비교하여 보행자 검출 역시 개선된 결과를 볼 수 있었으며, 추적 또한 잘 되었음을 볼 수 있었다. 하지만 제안하는 방법은 [표 4.9]와 같은 상황에서는 완전하게 형태를 검출하지 못할 수 있고, 제안한 식에서 AND 연산 후의 결과 영상은 기존 방법보다 노이즈가 많은 결과를 알 수 있었다. 이는 제안한 식의 경우 합성된 영상에 나누기를 하게 됨으로써 화소값 변화에 민감한 결과가 나오기 때문이다.

5.2 향후 연구 과제

[표 4.11]에서 보인 바와 같이 본 논문에서 제안하는 방법으로도 검출이 잘 안 되는 상황이 있었다. 두 명이상의 보행자가 밀접하게 붙어있을 경우 보행자의 검출을 수직 투영만으로 검출하기 힘들어지므로, 이를 해결하는 한 방법으로 수직 투영과 동시에 보행자의 색상 정보를 이용해서 보행자의 분리가 필요할 것으로 생각된다.

또한, 움직임이 없거나 배경색과 비슷한 옷을 입었을 경우 검출이 잘 안 되는 결과를 보였다. 이는 (식 3.1)의 현재 영상의 차와 에지 영상에서의 배경 영상과 명도의 차이가 거의 없기 때문인데, 이를 보완하는 방법이 필요하다.

제안한 방법 중 AND 연산 후의 결과 영상을 보면 기존 방법보다 노이즈가 많은 것을 알 수 있다. 이는 이미지의 합성을 하고 나누기를 하게 됨으로써 화소값 변화에 민감한 결과가 나오기 때문인데, 노이즈를 없애기 위한 한 방법으로 미디언 필터링 같은 노이즈 제거 필터를 쓰면 결과가 더욱 좋아질 것이라고 생각된다.

참고문헌

[1] 이주용, "무인 감시 장치를 위한 이동물체의 추적 알고리즘", 강원대학교 대학원 석사 학위 논문, 2001년 2월

[2] 김상훈, "다중생상정규화와 움직임 생상정보를 이용한 물체 검출", 한국정보처리학회 논문지, Vol. 12, No. 7 pp.721-728, 2005.12

[3] 임종석, 김욱헌, "움직임 정보를 이용한 다수 보행자 추적", 한국정보처리학회 춘계학술대회논문집 제1권, 제1호, pp. 755-758

[4] A. Gyaourova, C. Kamath, S.-C. Cheung, "Block Matching for Object Tracking", Lawrence Livermore National Laboratory, 2003, 10. 14

[5] Bernd Neumann, "Optical Flow", Computer Graphics, 1984. 1

[6] S. Lopez, G. M. Callco, J.F. Lopez and R. Sarmiento "A High Quality/Low Computational Cost Technique for Block Matching Motion Estimation", IEEE Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, 2005. 1

[7] 이형석, "웨이블릿 기만 투영을 이용한 움직임 추적", 한신대학교 대학원, 석사 학위 논문, 2001년

[8] 김상주, "다중 대표블록 모델기법을 통한 이동물체의 효율적 추적", 부산대학교 대학원, 박사 학위 논문, 2005. 2

[9] T. Koga et al. "Motion-compensated Interframe Coding for Video Conferencing", Proc. of Int. conf. Telecommunication. No. 29, pp 531-535, Dec. 1981

[10] J. R. Jain and A. K. Jain, "Displacement Measurement and its Application in Interframe Image Coding", IEEE Trans. Communication, Vol. Com-29, pp.1799-1808, Dec. 1981

[11] Y. Ninomiya and Y.Ohtsuka, "A Motion-Compensated Interframe coding scheme for Television Pictures", IEEE trans. Communication, Vol. Com-30, pp. 201-211, Jan. 1982

[12] Dong-Ho Lee, and Sung-Ho Cho, "An Efficient VLSI Architecture for Motion Estimation With 4-Step Search Algorithm", Proceedings of ITC-CSC'97 Okinawa, Japan, pp. 153-155, 1997

[13] Junavit Chalidabhongse and C.-C. jay Kuo. "Fast Motion Vector Estimation Using Multiresolution-Spatio-temporal Correlations", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 7, No. 3, pp. 477-488, June 1997

[14] B. K. P. Horn and B.G Schunck, "Determining Optical Flow", Artificial Intelligence Vol. 17, pp 185-204, 1981

[15] A. Singh, "Optical Flow Computation", IEEE Computer Society Press, 1991

[16] Shuichi Nishio, Hiroshi Tomiyasu, "Vehicle Detection using Edge And Color Information", Proc. First Korea-Japan Joint conf. Computer Vision, pp. 500-504, Seoul, Korea, Oct. 1991.

[17] Byoung Tae Chun, Min Wang, Jung Soh, "Moving Object Extraction Using Background Images", proc. 20th KISS Fall conf. pp. 341-344, Seoul, Korea, Oct. 1993.

[18] Masahiko Yachida, Minoru Asada, Saburo Tsuji, "Automatic Analysis of moving Images", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-3, No, 1, pp.12-19, January, 1981.

[19] N. Hoose, L. G. Willumsen, "Automatically Extracting Traffic Data from Video-Tape using the CLIP4 Parallel Image Processing",

Pattern Rcognition Letters, Vol. 6, pp. 199-213, August, 1987.

[20] Toshinari Nonaka, Shigero Kimura, Shinji, Ozawa, "A Moving Object Extraction Using Adapted Frame Integraion", Proc, First Korea-Japan Joint conf. Computer Vision, pp. 186-191, Seoul, Korea, Oct, 1991.

[21] Rafael C. Gonzalez and Richard E. Woods, "Digital Image Processing", Addison-Wesley Publishing Company, 1992.

[22] 홍석용, "PLinda 시스템과 임베디드 시스템을 이용한 움직임 감지시스템의 구현 및 성능평가", 한성대학교 대학원 석사 학위논문, 2005년

ABSTRACT

Detection and Tracking of Multiple Pedestrians Using AND Operation of Redundancy-removed Composition Image And Edge Image

Choi, Yang-Jin
Major in Computer Engineering
Dept. of Computer Engineering
Graduate School
Hansung University

In order to track moving objects in a video stream, an excellent detection and tracking is needed. In this paper, it is proposed that the approximate shape of a pedestrian is calculated by the composition and subtraction of successive frames, and then the shape is decided by using AND operation of the binary image of the approximate shape and the edge image of a current frame. And, noises generated through the shape identification process are removed by using the closing and labelling operations. Multiple pedestrians are separated precisely through the vertical projection making use of the size ratio of a pedestrian.

Pedestrians detected by the proposed method are tracked through the evaluation of similarity between objects. The similarity is to measure how much similar an object tracked in the previous frame is to an object detected in the current frame. The evaluation of such similarity is to calculate the scores of their connectivity, adjacency, direction and size ratios, and to sum them up. The tracking is performed by comparing the scores of objects tracked in the list in the previous frame with those of objects detected in the list in the current frame and then by updating the information of objects which have maximum similarity values in each list. By experiments in a real world, it is shown that the proposed detection and tracking methods are better than the previous methods using the difference of a previous frame and a current frame.