공 학 석 사 학 위 논 문

다수의 카메라를 이용한 이동하는 객체의 위치 추적

2015 년 2 월

부 산 대 학 교 대 학 원

차 세 대 기 판 학 과

우 혜 진

공 학 석 사 학 위 논 문

다수의 카메라를 이용한 이동하는 객체의 위치 추적

지도교수 백 광 렬

2015 년 2 월

부 산 대 학 교 대 학 원

차 세 대 기 판 학 과

우 혜 진

우혜진의 공학석사 학위 논문을 인준함

2014 년 11 월 28 일

위원장 김 형 남 (인)

위 원 백 광 렬 (인)

위 원 이 문 석 (인)

목 차

그림목차

표 목차

1. 서론

정보산업 사회로 발전할수록 세계 각국에서는 사람들이 노출될 수 있는 테러나 범죄와 같은 위험 요소에 대한 관심이 증가하고 있다. 다양한 위험 상황에 보다 효과적으로 대처하기 위해, 영상 보안 기술은 사고 대응 중심에서 사전 예방 중심으로 변화하고 있다. 사회적 안전 기반을 공고히 하기 위한, 국내 영상 감시 시스템은 2000년 이후 매년 15%이상 성장하고 있다[1].

수많은 영상자료를 모니터링 하는데 한계가 있기 때문에 최근에는 지능형 영상 감시 시스템의 관심이 증가하고 있다. 지능형 영상 감시 시스템이란 영상을 이해하고 적절한 대응을 할 수 있는 시스템이다. 사람의 눈이나 직감이 아닌 컴퓨터를 통해 관리자가 설정한 규칙에 위반되는 움직임이나 행동을 자동으로 감지할 수 있기 때문에 그 필요성이 대두 되고 있다. 국내 산업체의 영상 보안 시스템은 세계 정상급의 기술력을 보유하고 있지만, 지능형 영상 감시 기술은 해외업체들이 강세를 보이고 있어 지속적인 연구가 필요하다[2].

카메라를 이용한 감시 시스템은 목표 객체가 카메라의 화면을 벗어나면 지속적인 감시가 어렵기 때문에 FOV (field of view : 관측 시야)를 넓히기 위해 단일 카메라(PTZ(pan, tilt, zoom)카메라, 이동식 카메라)나 다중 카메라를 이용한다. 지속적으로 목표 객체를 추적하기 위해서는 영상 내에서 객체로 인식할 수 있도록 독특한 특징(색상, 크기)을 추출해야한다. 단일 카메라는 두 개 이상의 객체들이 가려지거나 겹쳐지면 이러한 특징을 추출할 수 없다. 다중 카메라를 이용하면 이러한 문제를 해결할 수 있다. 하지만 기존의 다중 카메라를 이용한 대부분의 시스템은 목표 객체에 대한 지속적인 추적을 위해 모든 카메라의 영상을 확인해야

하는 불편함이 있다. 그 이유는 시스템이 독립적으로 운용되고 있기 때문이다. 이러한 문제를 해결하기 위해 다중 카메라 환경에서 정보를 공유하여 효과적으로 감시하기 위한 연구가 활발히 진행되고 있다 [3]-[5].

본 논문에서는 객체의 겹침과 협소한 FOV로 인해 인식률이 감소하는 부분을 최소화하기 위해 카메라를 다중으로 배치한다. 또한 한 대 이상의 카메라에서 인식되는 객체를 지속적으로 감시할 수 있도록 카메라 간의 정보를 공유한다. 이를 토대로 움직임이 감지된 물체가 이동한 궤적과 높이를 구해 일정 영역을 감시할 수 있는 알고리즘과 시스템을 구현한다.

연속된 프레임이 지속적으로 입력되고 있을 때 차 영상으로 움직임 영역을 구분한 후, 전 처리 단계를 거쳐 객체를 배경으로부터 분할한다. 분할된 객체들은 특징이 되는 정보를 추출한다. 각 카메라에서 인지한 객체들은 최하위 지점이 유사한 위치이고, 이동 벡터가 유사하면 같은 객체로 인지한다. 객체의 최하위 지점들의 평균으로 바닥에 놓인 위치를 인식한다. 일정 영역에서 비대칭으로 설치된 카메라는 카메라가 없는 방향으로 가중치를 주어 정확한 위치를 찾는다. 객체의 최상위 지점으로 바닥에 투영된 영상 축과 바닥의 각도를 이용하여 객체의 높이를 추정한다. 이때 설치된 카메라가 많을수록 객체가 놓인 위치와 높이를 더욱 정확하게 인식할 수 있다. 분할된 객체에서 얻어낸 특징 중 이동 벡터를 이용하여, 이전 영상의 객체가 현재 영상의 위치로 이동한 경로를 토대로 궤적을 그린다. 그림 1은 전체 개요도이다.

본 논문은 다음의 순서로 구성된다. 2장에서는 단일 카메라에서 객체를 인식, 추적하는 알고리즘을 소개한다. 3장에서는 일정 영역 내에서 움직임을 보이는 모든 물체를 감지하기 위해 효과적으로 다중 카메라를

배치하는 방법과 개별 카메라에서 객체들을 정합하고, 이를 토대로 이동하는 객체가 이동한 궤적을 그리는 과정을 보인다. 4장에서는 실험 환경 구축과 알고리즘 성능에 대한 검증을 보이며, 마지막 5장에서는 결론을 맺는다.

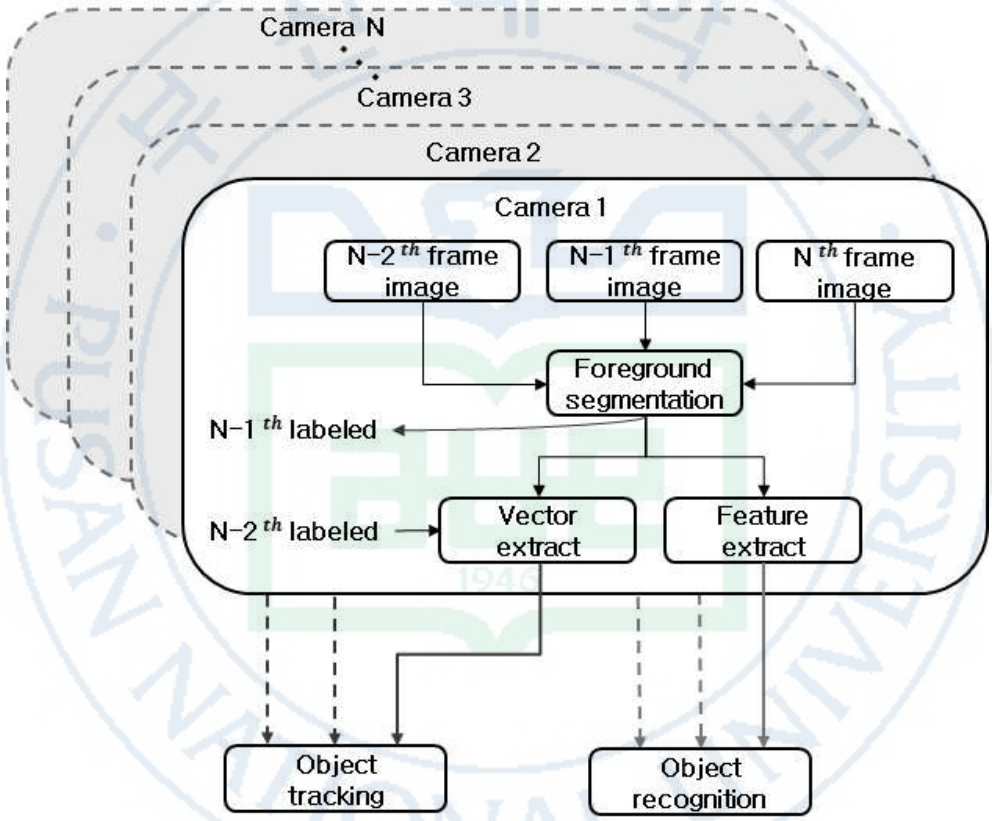


그림 1. 전체 개요도

Fig. 1. Configuration of the whole system

2. 단일 카메라에서의 객체 인식, 추적 알고리즘

2.1. 알고리즘 개요

제안하는 알고리즘은 단일 카메라에서 인식할 객체에 대해서 다음의 두 가지 가정을 기본으로 구현한다.

(1) 인식할 객체와 배경의 색상 명암비가 뚜렷하다.
(2) 객체는 FPS(frame per second : 초당 프레임 수)보다 천천히 움직인다.

단일 카메라에서 인식하고자하는 객체와 배경의 색상 명암비가 뚜렷하면 객체를 배경으로부터 보다 선명하게 분할할 수 있다. 객체의 움직임은 프레임의 변화에 따른 이동 벡터로 추정한다. 두 개 이상의 객체가 FPS에 비해 빠르게 움직였다면 이전 프레임에서 어떤 객체가 다음 프레임으로 이동했는지 예측하기 어렵다.

두 가지의 가정을 토대로 본 절에서는 단일 카메라에서 움직임이 감지된 객체를 인식하기 위한 알고리즘 개요를 설명한다. 그림 2는 단일 카메라에서의 알고리즘 개요도이다. 알고리즘은 크게 객체 분할, 추적, 인식으로 나누어진다.

객체 분할에서는 연속된 프레임으로 영상이 들어오고 있을 때, 새로운 영상이 입력되면 이전 영상에서 얻은 정보를 바탕으로 이동하고 있는 객

체들을 배경으로부터 분할한다. 분할된 객체들과 이전 영상에서 얻은 분할된 객체들은 블록 매칭을 이용하여 이동한 거리와 방향을 구분한다. 새로운 영상이 들어왔을 때 앞서 구한 이동 벡터를 통해서 이동한 객체의 이동 경로를 추적한다. 객체 인식에서는 움직임이 감지된 객체의 특징이 되는 정보를 추출한다.

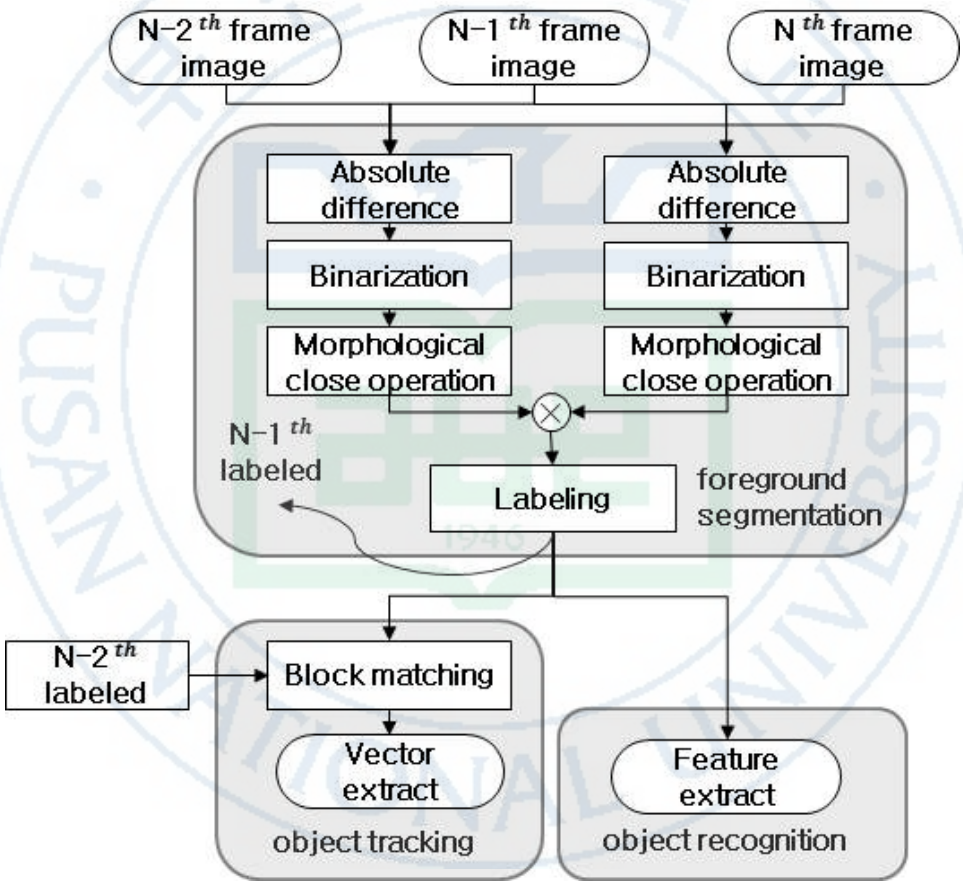


그림 2. 이동하는 객체 검출 알고리즘 블록도

Fig. 2. Block diagram of the moving object detection algorithm

2.2. 객체 분할 (foreground segmentation)

카메라를 통해 입력된 흑백영상에서 움직임이 있다고 감지된 객체의 영역을 분할하는 알고리즘을 소개한다. 객체 분할은 새로운 영상이 입력되었을 때 제일 먼저 수행한다. 먼저 연속된 세 프레임 중에서 인접한 두 프레임씩 차분 영상 기법을 이용해서 움직임이 감지된 영역을 추출한다. 움직임이 있었던 영역을 Otsu 이진화로 연산에 불필요한 부분을 제거한다. 각종 잡음을 제거하고, 객체의 움직임이 FPS에 비해 매우 느리게 될 경우 차분 영상 기법 결과에서 객체의 일부분만 검출되지 않도록 모폴로지 닫기 연산을 시행한다. 이 과정으로 얻은 두 이진 영상을 AND연산 한 후 레이블링을 통해 움직임이 있는 객체를 배경으로부터 분할한다. 그림 3은 단일 카메라에서 움직임이 있었던 영역을 배경으로부터 분할한 과정을 영상 처리한 결과이다.

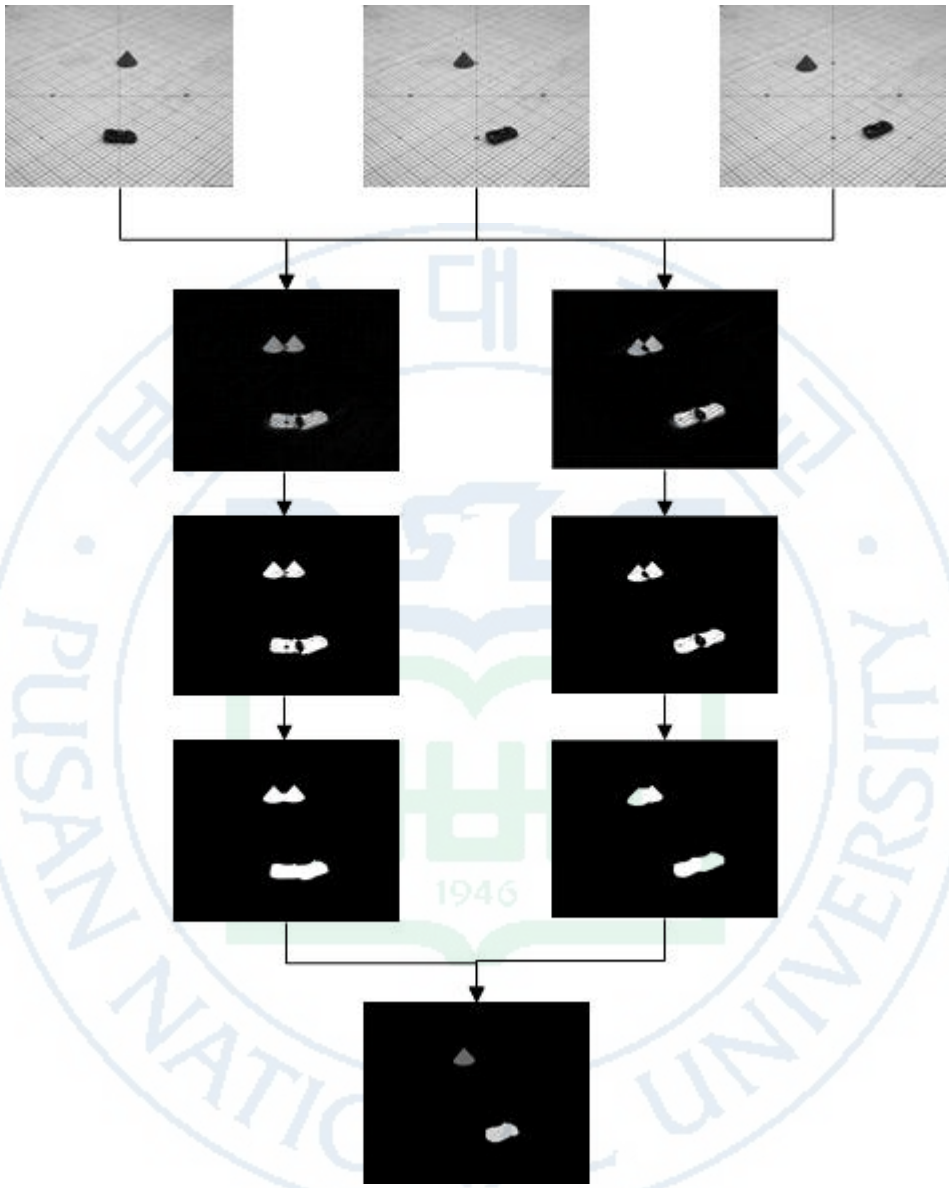


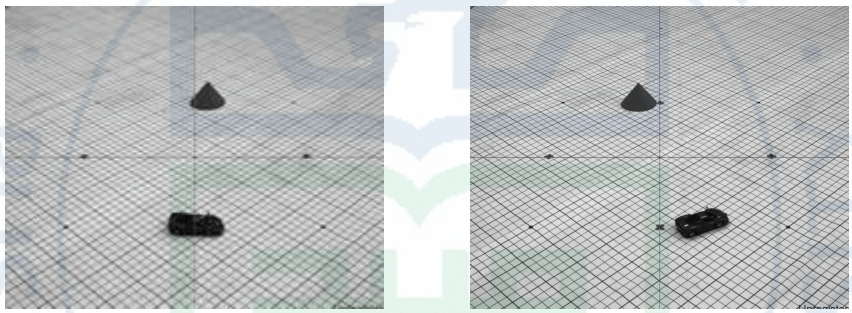
그림 3. 이동하는 객체 검출 과정

Fig. 3. Processes of the moving object detection

2.2.1. 차분 영상

인접한 두 프레임 영상에서 움직임이 있는 영역을 추출하기 위해 차분 영상 기법을 사용한다. 차분 영상은 이전 영상과 현재 영상을 픽셀단위로 차이의 절댓값으로 구한다. 식 (2.2.1.1)은 차분 영상을 구하기 위한 식이다. 그림 4는 인접한 프레임 영상의 차분 영상을 구한 결과이다.

$$u,v) = \ I(u,v) - I_{t-1}(u,v)| \qquad (2.2.1.1)$$



(a) (b)

(c)

그림 4. 차분 영상 : (a) 이전 영상 (b) 현재 영상 (c) 차분 영상

Fig. 4. Difference image : (a) Previous image (b) Current image (c) Difference image

2.2.2. 이진화

이진화는 일반적으로 색상 영상이나 흑백 영상을 2진 영상으로 변환하는 기법으로 임계값 처리가 쓰인다. 이때, 적절한 임계값을 결정하는 것이 객체와 배경을 분할하는 중요한 매개 변수가 된다. 이진화는 전역 고정 이진화(global fixed thresholding), 지역 가변 이진화(locally adaptive thresholding), 히스테리시스 이진화(hystheresis thresholding)으로 구분한다. 본 논문에서는 2.2.1.절에서 차분 영상 기법으로 배경과 움직임이 감지된 영역을 구분하였기 때문에 전역 고정 이진화 방법인 Otsu 방식을 선정하였다[6]. Otsu 알고리즘은 어떤 집합을 두 클래스로 나눌 때 통계학적인 방법으로 분류하는 방법이다. 전체 분산은 클래스 내 분산과, 클래스 간 분산의 합으로 나타낼 수 있다. 이는 식 (2.2.2.1)으로 표현한다. 최적화된 임계값( )은 클래스 내 분산이 최소가 되거나 클래스 간 분산이 최대가 되는 임계값인데, 일반적으로 클래스 내 분산에 비해 쉽게 구할 수 있는 클래스 간 분산을 구하여 임계값을 찾는다. 클래스 간 분산은 식 (2.2.2.2)로 표현이 되고 최적화된 임계값은 식(2.2.2.3)과 같다.

$$\sigma^2 + \sigma_B^2 = \sigma_T^2 \tag{2.2.2.1}$$

$$\sigma_B^2(k) = \frac{[\mu_T \omega(k) - \mu(k)]^2}{\omega(k)[1-\omega(k)]} \tag{2.2.2.2}$$

$$\sigma_B^2(k^*) = \max \sigma_B^2(k) \tag{2.2.2.3}$$

위 식에서 는 클래스 내 분산이고, σ_B^2는 클래스 간 분산이며, σ_T^2가 전체 분산이다. 그림 5는 Otsu 알고리즘으로 이진화한 결과이다.

(a) (b)

그림 5. Otsu 이진화 : (a) 차분 영상 (b) 이진 영상

Fig. 5. Otsu thresholding : (a) Difference image (b) Otsu's image

2.2.3. 모폴로지 연산

모폴로지 연산은 영상을 형태학적인 관점에서 다루는 기법으로 다양한 영상 처리 시스템에서 전, 후 처리의 형태로 널리 사용되고 있다. 이 기법의 가장 기본이 되는 연산은 침식과 팽창인데, 일반적으로 침식 연산은 영상 내에서 객체를 깎아내는 효과가 있고, 팽창 연산은 객체를 확장하는 효과가 있다. 식 (2.2.3.1)은 이진 영상에서의 침식 연산을, 식 (2.2.3.2)는 이진 영상에서의 팽창 연산을 정의한다.

$$B = \{x | (B) \subset A\} \tag{2.2.3.1}$$

$$\begin{aligned} A \oplus B &= \{x | (B) \cap A \neq \varnothing\} \\ &= \{x | [(\hat{B})_x \cap A] \subseteq A\} \end{aligned} \tag{2.2.3.2}$$

위 수식에서 집합 A는 입력 영상을 나타내고, 집합 B는 모폴로지 연산의 결과를 조정하는 구성요소(structure element)를 나타낸다. 구성요소에 따라 모폴로지 연산을 수행한 결과가 다르게 나타난다. 본 논문에서는 일반적인 형태인 정사각형의 구성요소를 사용하였다. 그림 6은 구성요소를 나타내는 그림이고, 구성요소에서 가운데 점(●)이 찍힌 픽셀이 원점에 해당하는 픽셀이다.

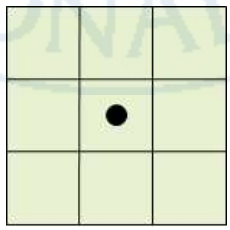


그림 6. 구성요소

Fig. 6. Structure element

본 논문에서는 팽창 연산을 수행한 후 침식 연산을 수행하여 가늘게 패인 부분을 채워주는 닫기 연산을 활용하였다. 식 (2.2.3.3)은 닫기 연산을 나타내고, 그림 7은 영상에서의 모폴로지 연산을 나타낸다.

$$B=(A\oplus B)\ominus B \qquad (2.2.3.3)$$

(a) (b)

그림 7. 모폴로지 연산 : (a) 이진 영상 (b) 모폴로지 영상(닫기)

Fig. 7. Morphological operation :

(a) Binary image (b) Morphological image(close)

2.2.4. 레이블링

레이블링은 이진 영상에서 픽셀들의 집합을 구분하기 위해 각 집합에 고유의 번호를 매기는 방법으로 객체를 인식하기 위한 대표적인 전 처리 방법이다. 본 논문에서는 인접한 모든 픽셀을 하나의 객체로 인식하기 위하여 8-이웃 연결성 레이블링을 사용하였다. 그림 8은 객체를 분할하기 위해 전 처리를 거친 이진 영상에 레이블링을 적용한 결과이다.

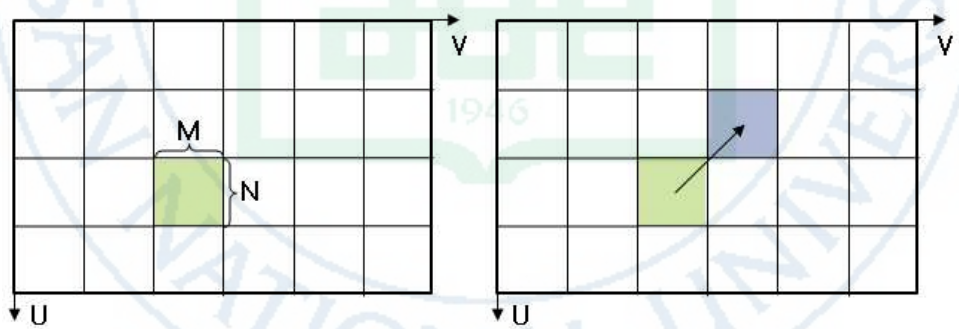
(a) (b)

그림 8. 레이블링 : (a) 이진 영상 (b) 레이블 영상

Fig. 8. Labeling : (a) binary image (b) label image

2.3. 객체 추적 (object tracking)

영상에서 움직임을 추적하는 방법에는 대표적으로 옵티컬 플로우(optical flow)와 블록 매칭(block matching)으로 나뉜다. 옵티컬 플로우는 영상 내부의 모든 픽셀에서 속도장을 계산하는 혼-셩크 방법과, 코너와 같이 두드러진 특징점으로 움직임을 계산하는 루카스-카나데 방법으로 나뉜다[7][8]. 블록 매칭은 영상을 블록으로 나누어 블록의 움직임을 계산한다[9]. 2.2절에서 객체와 배경을 분할하는 알고리즘을 구현하였기 때문에 별도의 특징점을 구할 필요 없이 객체의 움직임만 계산하면 된다. 그러므로 본 논문에서는 블록 매칭을 이용하여 영상에서 객체의 움직임을 추적하였다. 그림 9는 인접한 두 프레임의 영상에서 블록 매칭을 이용하여 이동 벡터를 구하는 과정을 그림으로 나타내었다.



(a) (b)

그림 9. 블록 매칭 : (a) 이전 영상의 특정 블록
(b) 인접한 두 영상에서의 움직임 벡터

Fig. 9. Block matching : (a) Particular block of the previous image
(b) Motion vectors of the two adjacent image

블록 매칭으로 움직임을 계산하는 방법은 다음과 같다. 우선 레이블링을 수행한 영상과 연속된 두 프레임의 흑백 영상을 32 32 화소 크기의 블록으로 나누어 해상도를 낮춘다. 한 블록 내에 객체의 영역이 30%이상 포함되는 블록들의 위치 좌표로 움직임 벡터를 구한다. 이전 영상에서 블록들의 화소 값과 가장 유사한 블록을 다음 영상에서 찾기 위해 해당 블록을 기준으로 일정한 범위 내에서 유사도가 가장 높은 블록으로 매칭한다. 유사도를 비교하기 위해서 식 (2.3.1)과 같이 블록간의 MAD(mean absolute difference : 중간 절댓값 차)를 이용한다. 두 영상의 MAD를 가장 작게 만드는 변위 (x,y)의 값을 해당 블록의 움직임 벡터로 간주하고 이를 식으로 표현하면 식 (2.3.2)과 같다.

$$AD_{k,l)}(x,y) = \frac{1}{MN}\sum_{i=0}^{-1N-1}\sum_{j=0}|I_t(k+i,l+j)-I_{t-1}(k+x+i,l+y+j)| \quad (2.3.1)$$

$$v(k,l) = \arg\min MAD_{(k,l)}(x,y) \quad (2.3.2)$$

위 수식에서 I_t는 t번째 프레임의 영상을, I_{t-1}는 $t-1$번째 프레임의 영상을 나타내고, M 과 N은 블록의 크기를 나타낸다. v는 움직임 벡터를 나타낸다. 일정한 범위 내의 모든 블록에서 MAD를 비교하는 경우 윈도우 크기에 따라 연산 시간이 좌지우지된다. 본 논문에서는 이러한 연산량을 최소화하기 위해 다이아몬드 검색 방법(diamond search)을 이용하여 움직임 벡터를 찾는다[10]. 다이아몬드 검색 방법은 큰 다이아몬드 패턴과 작은 다이아몬드 패턴으로 구성되어있다. 큰 다이아몬드 패턴으로 최소 MAD의 근방까지 이동한 후, 작은 다이아몬드 패턴으로 블록이 이동한 정확한 위치를 찾는다. 그림 10은 다이아몬드 검색 패턴과 이

를 이용한 움직임 추적 결과이다. 그림 11은 다이아몬드 검색 알고리즘으로 움직임 벡터를 구한 결과를 영상처리로 나타낸다.

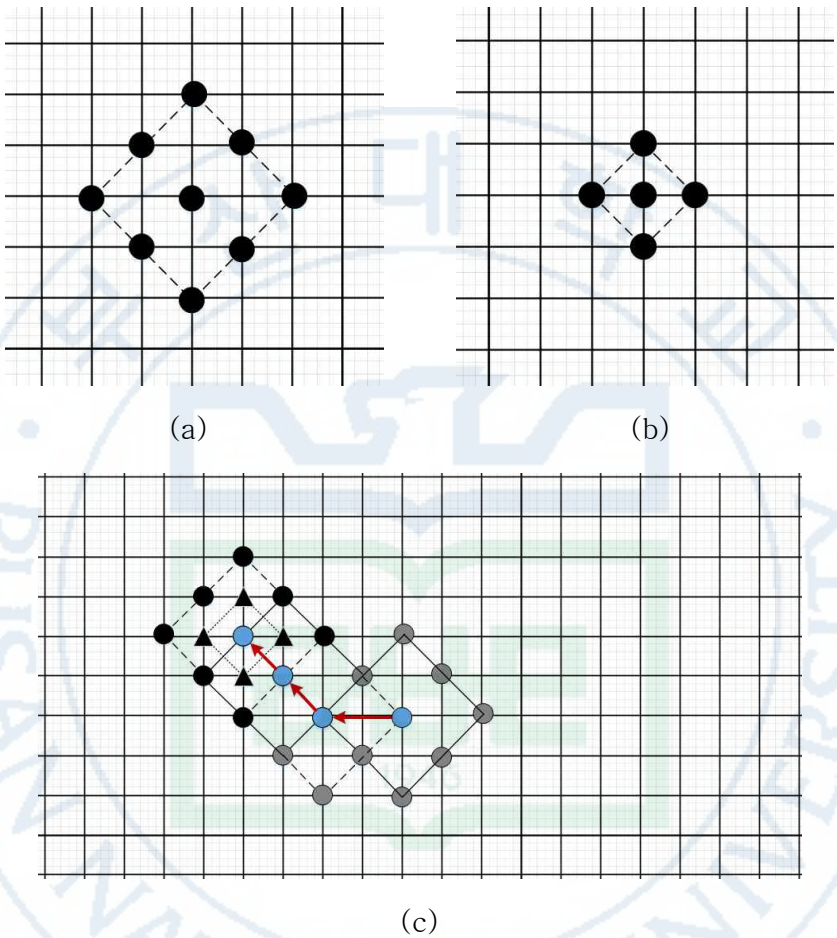


그림 10. 다이아몬드 검색 패턴을 이용한 움직임 추적
(a)큰 다이아몬드 패턴 (b)작은 다이아몬드 패턴
(c) 움직임 추적

Fig. 10. Motion tracking with diamond search method
: (a) Large diamond pattern
(b) Small diamond pattern
(c) Motion tracking

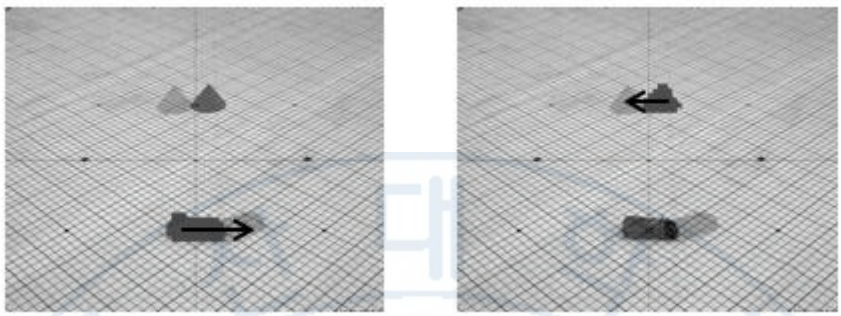


그림 11. 이동 벡터 검출 결과 영상

Fig. 11. The movement vector detected resulting image

2.4. 객체 인식 (object recognition)

2.4절에서는 입력 영상에서 분할된 객체로부터 특징이 되는 정보를 토대로 객체를 인식한다. 이것은 같은 레이블을 할당받은 영역의 연결 성분을 이용하여 판별되는데 기하학적 특징, 경계선 대조 특징, 형태 규칙성 특징, 공간적 일관성 특징, 외곽선 특징 등을 이용하여 구분할 수 있다[11]. 본 논문에서는 관측 시야 내에서 움직이는 모든 객체를 인지하는 것이 목표다. 그러므로 특정 물체를 인식하기 위한 조건을 거는 것이 아니라 객체로부터 얻을 수 있는 기본적인 특징들을 토대로 객체가 놓인 위치를 인식한다. 기본적인 특징을 추출하기 위해 영상에 투영된 객체의 최하위 지점과 최상위 지점을 이용한다.

2.4.1. 최하위 지점

객체가 바닥에 놓인 위치를 찾기 위해 단일 카메라에서 얻은 최하위 지점을 이용한다. 최하위 지점을 구하기 위해 너비의 중점에서 가장 아래에 있는 지점을 이용한다. 이때 바닥에 닿는 면이 원형에 가까울수록, 설치된 카메라가 많을수록 추정하는 위치의 정확도는 높아진다. 그림 12은 최하위 지점으로 객체가 놓인 위치를 찾는 과정을 나타낸다.

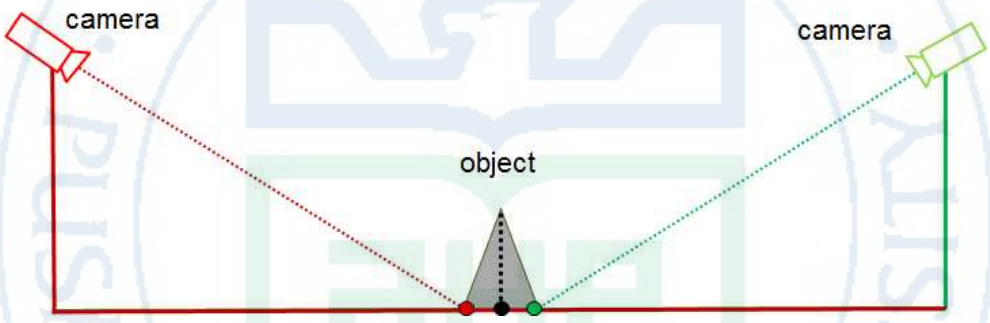


그림 12. 개별 카메라에서의 최하위 지점으로 객체의 위치 인식

Fig. 12. Positioning of the object to the lowest point in each camera

2.4.2. 최상위 지점

영상에 투영된 객체의 최상위 지점은 실제로 높이가 있는 물체의 최상위 지점이 아니라 바닥에 투영된 지점이다. 카메라가 최상위 지점을 바라본 영상 축과 바닥의 각도를 이용하면 객체의 높이를 추정할 수 있다. 이는 식 (2.4.1)으로 표현할 수 있다.

$$_{am}\quad L_{cam,\,t\,point} = H_{obj} : L_{obj,\,t\,point} \tag{2.4.1}$$

위 식에서 H_{cam}는 카메라가 설치 된 높이, $L_{cam,\,t\,point}$는 카메라가 설치된 위치에서 객체의 최상위 지점이 바닥에 투영된 지점 사이의 거리, H_{obj}는 객체의 높이, $L_{obj,\,t\,point}$는 객체가 놓인 위치에서 객체의 최상위 지점이 바닥에 투영된 지점 사이의 거리이다. 그림 13는 객체의 최상위 지점이 바닥에 투영된 지점을 나타낸다.

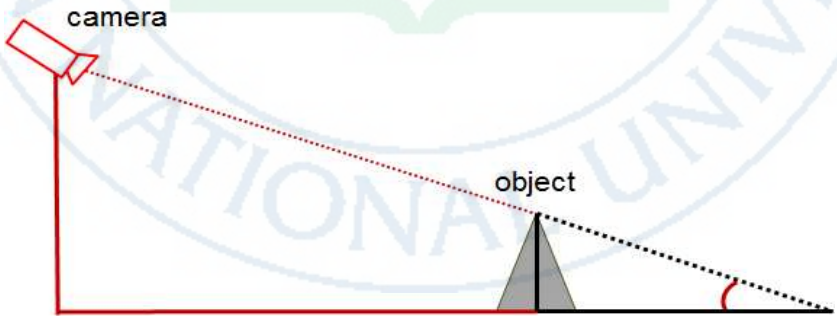


그림 13. 바닥에 투영된 객체의 최상위 지점

Fig. 13. The highest point of the projected object on the floor

3. 다중 카메라 배열

3.1. 카메라 배치

다중 카메라를 이용한 감시 시스템은 단일 카메라의 좁은 FOV를 보완한다. 또한 단일 카메라에서 두 개 이상의 객체들이 장애물에 가려지거나 서로 겹쳐지는 현상에서도 지속적인 인식이 가능하다. 일정 영역을 감시하는 시스템을 구축하기 위해 다양한 방법으로 카메라 배치를 할 수 있다.

본 논문에서는 다중 카메라 시스템이 단일 카메라 시스템 대비 뛰어난 성능을 검증하기 위해 모든 카메라를 동일한 조건으로 설치하였다. 또한, 이동하는 객체를 감시하기 위해서는 비교적 높은 위치에서 고정된 형식으로 카메라를 설치해야 한다. 동등한 구동조건으로 고정된 카메라들은 바닥을 향해 수직으로 바라보거나 사선으로 바라보도록 배치할 수 있다. 본 절에서는 두 가지의 카메라 배열 방법을 비교하고, 이를 토대로 성능을 검증하기 위한 적합한 방식을 선정한다.

① 바닥을 향해 수직으로 내려다볼 수 있도록 설치

설치된 카메라가 많을수록 관측시야가 넓어지게 되어 감시할 수 있는 영역이 넓어지게 된다. 객체가 이동하면서 형체가 변하지 않는다면, 영상을 통해 인식되는 객체의 모양이나 크기는 관측된 위치에 상관없이 동일하다. 또한, 입력되는 영상이 바닥을 향해 수직으로 투영되기 때문에 두 개 이상의 객체가 붙어 있더라도 충분히 분할 할 수 있다. 하지만 영상에서 볼 수 있는 대부분의 면이 객체의 윗면이라는 특징이 있다. 그렇기 때문에 객체의 위치와 윗면의 모양은 알 수 있지만, 객체의 높이를 알기 힘들다. 다중 카메라가 설치된 경우 스테레오-비전 카메라의 원리를 이용하여 높이를 구할 수 있다. 하지만 카메라 간의 FOV가 중첩되어야만 적용할 수 있으므로 많은 카메라가 필요하다는 단점이 있다. 그림 14은 다수의 카메라가 바닥을 향해 수직으로 내려다 볼 때의 카메라 배치도이다[12].

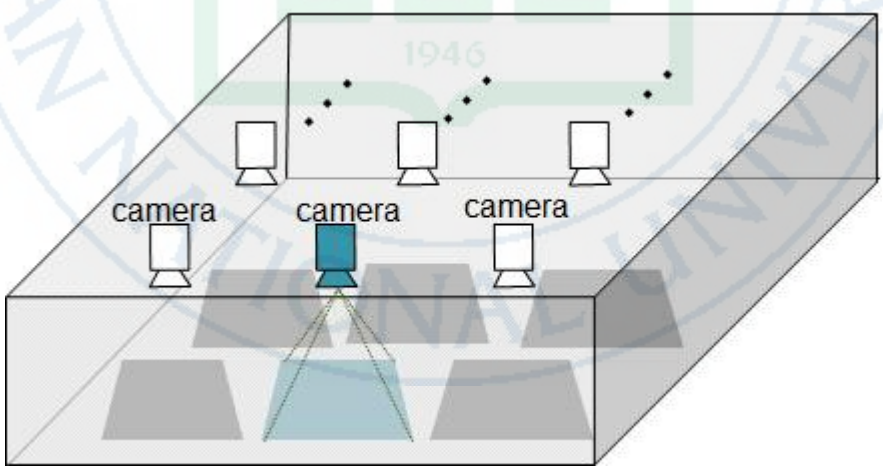


그림 14. 카메라 배치도 (바닥을 향해 수직으로 설치)

Fig. 14. Camera layout (mounted vertically towards the floor)

② 바닥을 향해 사선으로 내려다볼 수 있도록 설치

카메라가 바닥을 향해서 사선으로 내려다보는 경우에는 설치된 카메라 개수가 많을수록 FOV가 중첩된 영역이 증가한다. 단일 카메라로 설치해도 에 따라 충분히 넓은 영역을 감시할 수 있기 때문에 설치된 카메라 개수에 따라 FOV가 크게 영향을 미치지는 않는다. 다중 카메라 환경에서는 설치된 카메라 개수가 많을수록 다양한 위치에서 객체를 바라볼 수 있다. 여기서 2.4절에서 설명한 위치 인식 알고리즘을 적용하면 객체가 놓인 위치와 높이에 대한 정보를 추정할 수 있다.

이와 같이 카메라 배치를 하였을 때, 단일 카메라에서 두 개 이상의 객체가 서로 겹치거나 가리는 경우 설치된 위치에 따라 인식하지 못한다. 그렇지만, 다중 카메라 환경에서 다른 각도에서 바라본 영상이 있다면 충분히 해결할 수 있다. 그림 15은 카메라가 바닥을 향해 사선으로 내려달 볼 때의 카메라 배치도이다.

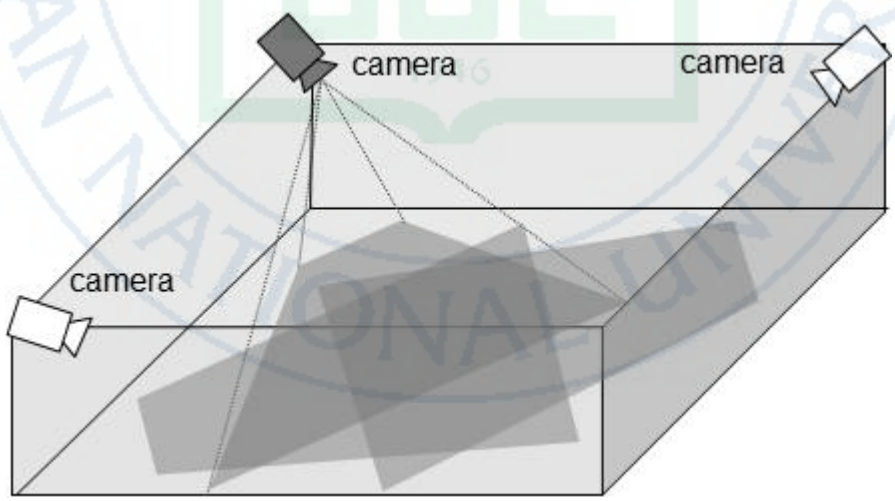


그림 15. 카메라 배치도 (바닥을 향해 사선으로 설치)

Fig. 15. Camera layout (mounted at an angle towards the floor)

3.2. 영상 좌표와 실제 좌표

본 논문에서는 영상에서 얻은 객체로부터 실제 좌표에서의 위치를 인식하기 위해 Bertozzi와 Broggi가 제안한 삼각함수를 활용한다. Bertozzi와 Broggi는 렌즈에서의 각도와 픽셀의 위치와는 균일한 분포를 가지고 있다고 가정하고 선형함수를 통해 영상 좌표와 실제 좌표의 관계를 정의한다[13]. 그림 16은 영상 좌표와 실제 좌표간의 관계를 나타낸다.

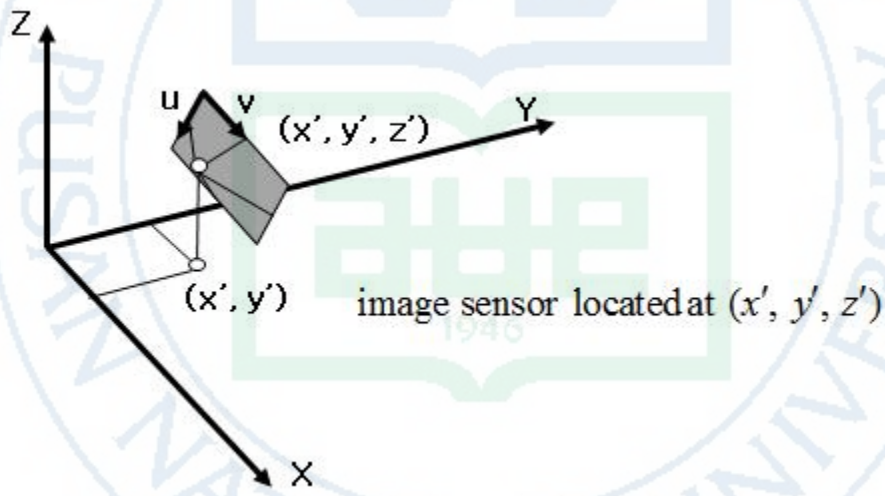


그림 16. 두 좌표계의 연관 관계

Fig. 16. Relationship between the two coordinate systems

$$
\begin{aligned}
(u,v) &= \begin{matrix} h\sin[(\bar{\gamma}-a_{\gamma})+v\dfrac{2\alpha_{\gamma}}{n-1}]\Big) \\ \tan[(\theta-\alpha)+u\dfrac{2\alpha_{\theta}}{m-1}\end{matrix} + d \\
y(u,v) &= \frac{h\left(\cos[(\bar{\gamma}-a_{\gamma})+v\dfrac{2\alpha_{\gamma}}{n-1}]\right)}{\tan[(\bar{\theta}-\alpha_{\theta})+u\dfrac{2\alpha_{\theta}}{m-1}}+l \\
z &= 0
\end{aligned}
\qquad (3.2.1)
$$

위 수식에서 γ는 광축을 $z=0$인 평명으로 사영시켜 x축과 이루는 각도이고, θ는 광축을 $y=0$인 평면으로 사영시켜 z축과 이루는 각도이다. 입력 영상의 해상도는 $m\times n$이고 $u=0,1,2,...,m-1$ (m=height of image), $v=0,1,2,...,n-1$ (m=width of image) 그리고 카메라의 상하 시야각은 $2\alpha_{\theta}$로 좌우 시야각은 $2\alpha_{\gamma}$로 주어진다.

식 (3.2.1)는 카메라의 내, 외부 매개변수를 이용하여 구한 식으로 변환 공식을 거친 모든 좌표는 z=0인 영역에 대응한다고 가정한다. 렌즈의 비선형성으로 인한 왜곡은 고려하지 않았기 때문에 그로 인한 오차가 발생할 수 있다. 그림 17은 수식에 사용된 주요 매개변수이다.

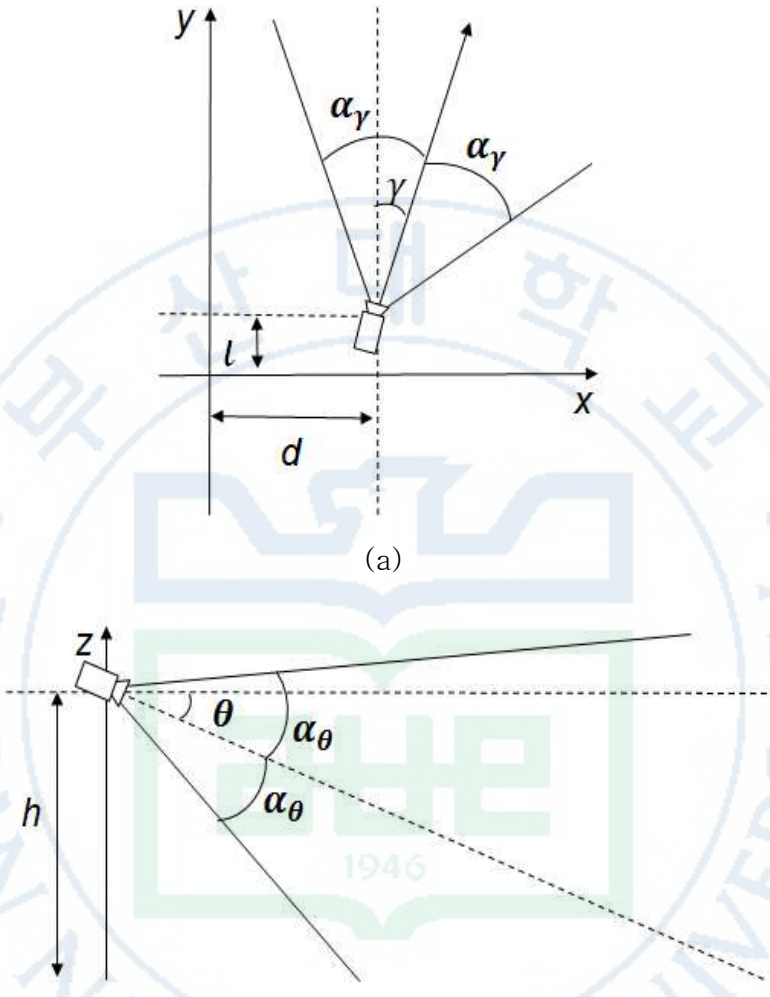


그림 17. 카메라 주변 매개변수 :
(a) y좌표, (b) xz좌표

Fig. 17. Around the camera parameters :
(a) The xy plane in the real world space
(b) The xz plane in the real world space

3.3. 개별 카메라 객체 정합

2.2절에서는 단일 카메라에서 움직임이 있는 영역에 레이블링을 이용하여 객체들을 분할하였다. 본 절에서는 다중 카메라로 확장하여 고유의 레이블을 가진 객체들이 다른 카메라에서도 같은 객체로 인식할 수 있도록 정합한다. 한 카메라를 기준으로 나머지 카메라들에서 추출한 객체 최하위 지점의 실제 좌표와 2.3절에서 추출한 이동 벡터의 유사도가 가장 높은 객체들을 같은 객체로 정합하였다.

① 최하위 지점의 유사도

기준 카메라에서 얻은 최하위 지점과 나머지 카메라에서 얻은 최하위 지점을 실제 좌표로 변환한 후, RMS(root mean square)가 가장 낮은 점들을 같은 객체의 점으로 인식한다. RMS는 식 (3.3.1)를 이용하여 구한다.

$$MS_{bj} \qquad (Y_{ref} - Y_{cam\_obj})^2 + (X_{ref} - X_{cam\_obj})^2 \qquad (3.3.1)$$

위 수식에서 (Y_{ref}, X_{ref})는 기준 카메라에서 최하위 지점의 좌표, $(Y_{cam_obj}, X_{cam_obj})$는 다른 카메라에서 최하위 지점의 좌표이다. 그림 18은 기준 카메라에서 객체의 최하위 지점과 RMS가 가장 낮은 점으로 객체를 정합하여 같은 객체로 구분한 결과이다.

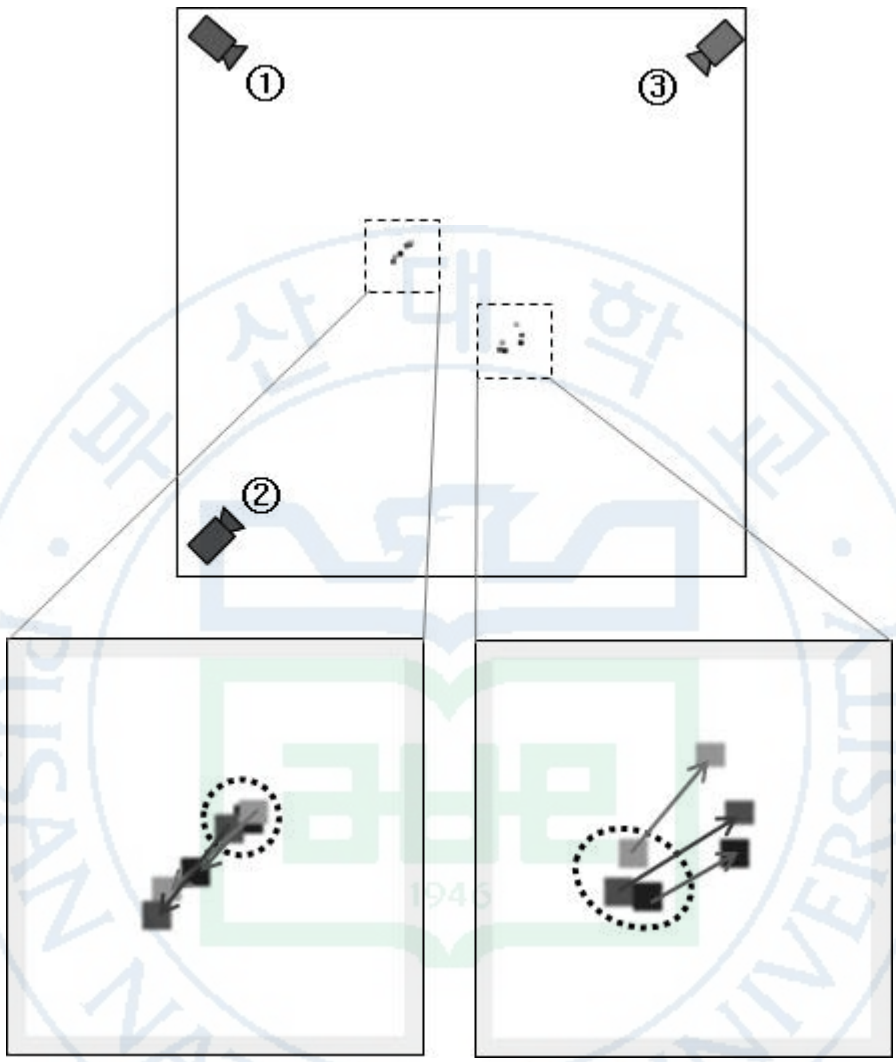
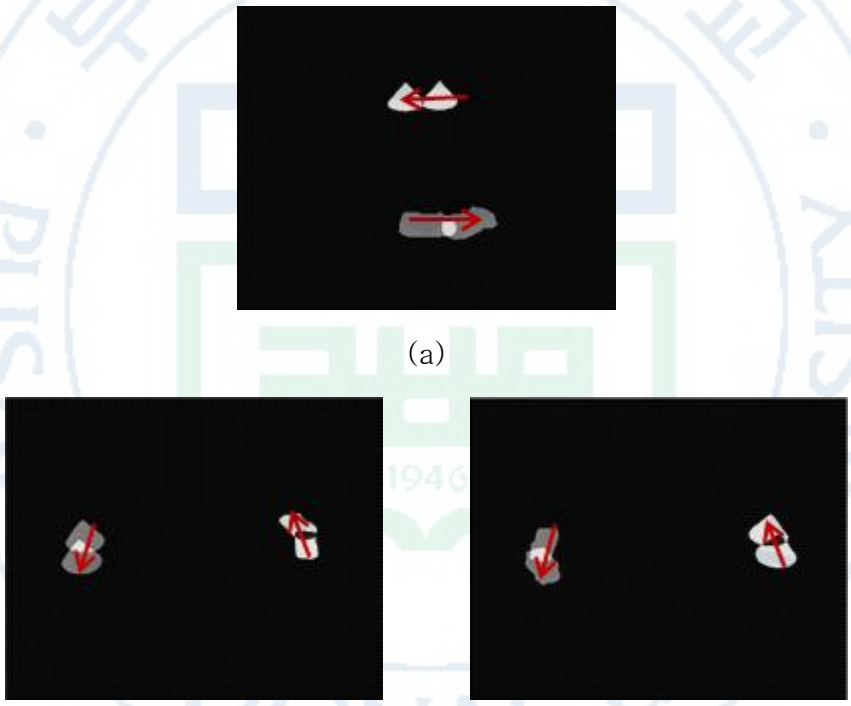


그림 18. 실제 좌표로 변환 한 최하위 지점들의 유사도

Fig. 18. Similarity of the lowest point of conversion to real world space

② 이동 벡터의 유사도

최하위 지점의 유사도를 판별하여 각 카메라에서 같은 객체로 간주하였다. 각 카메라에서 이동 벡터 또한 유사한 추세를 보이면 최종적으로 동일한 객체로 인식한다. 최하위 지점의 유사도와 이동 벡터의 유사도를 동시에 활용하면 객체가 겹치거나 움직임 벡터가 유사한 경우에도 목표 객체를 인지할 수 있다. 그림 19는 개별 영상에서의 이동 벡터를 나타낸 결과이고, 그림 20은 개별 카메라에서 객체들을 정합한 결과이다.



(a)

(b) (c)

그림 19. 개별 영상에서의 이동 벡터 : (a) 기준(1번) 카메라
(b) 2번 카메라 (c) 3번 카메라

Fig. 19. Movement vector at each camera :
(a) reference(1st) camera
(b) 2nd camera (c) 3rd camera

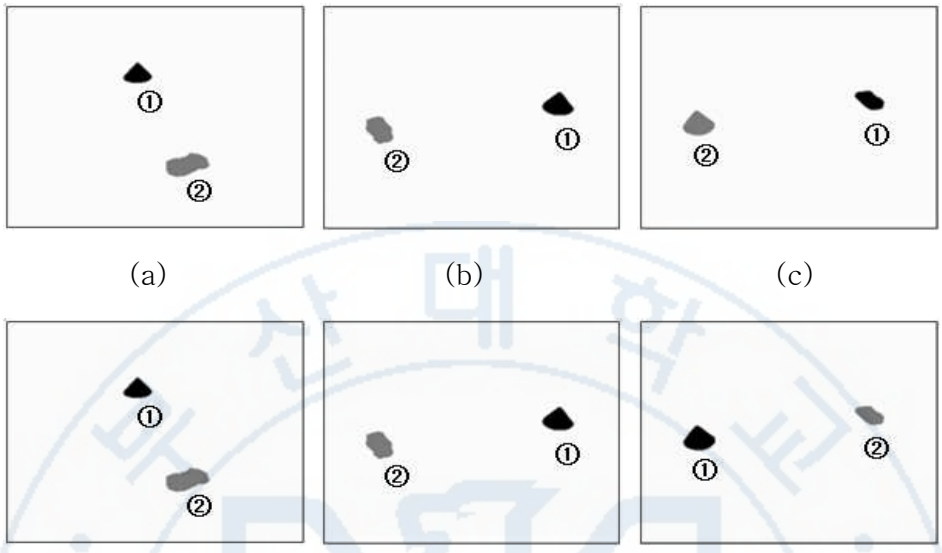
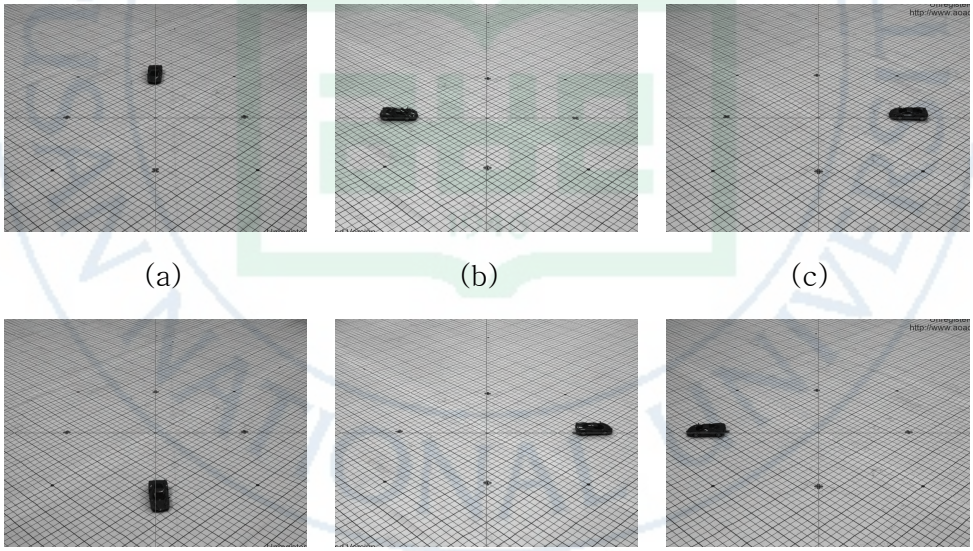


그림 20. 개별 카메라 객체 정합 : (a) (b) (c) 전, (d) (e) (f) 후

Fig. 20. Individual camera object matching :
(a) (b) (c) before (d) (e) (f) after

3.4. 이동 경로 표시

본 절에서는 객체가 놓인 위치 인식을 기반으로 프레임 변화에 따른 객체들의 이동 벡터를 이용하여 이동 경로를 추적한다. 초기에 객체의 정보가 없는 상태에서는 2.4절 객체 인식 알고리즘으로 실제 좌표에서 객체가 놓인 위치를 구한다. 다음 영상이 입력되면, 블록 매칭으로 구한 이동 벡터만큼 이전 영상에서의 좌표에서 더하여 위치를 추정한다. 연속된 프레임에서 이와 같은 과정을 반복하여 이동한 객체의 경로를 추적한다. 그림 21은 개별 카메라에서 객체가 이동하기 전과 후의 영상이고, 그림 22는 초기에 위치 인식을 한 후 이동 벡터를 이용하여 궤적을 그린 결과이다.



(a) (b) (c)

(d) (e) (f)

그림 21. 개별 카메라 이동 : (a) (b) (c) 전, (d) (e) (f) 후

Fig. 21. Individual camera movement :
(a) (b) (c) before, (d) (e) (f) after

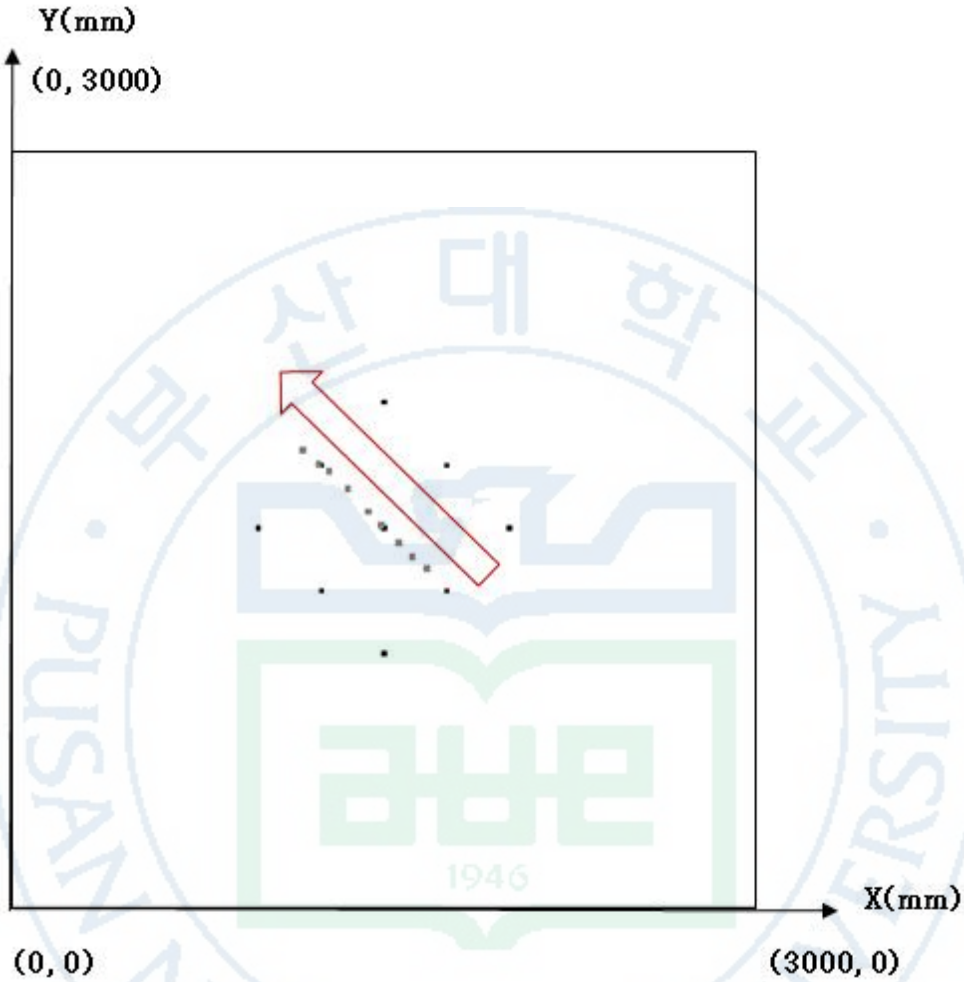


그림 22. 이동 경로 표시

Fig. 22. Trajectory results

4. 실험 및 결과

4.1. 실험 환경 구축

영상 입력을 위해 사용된 카메라는 Basler社의 acA1300-60gc를 사용하였다. /1.8′ CMOS카메라인 acA1300-60gc의 interface는 GigE방식이며 C-mount 렌즈를 사용할 수 있고 2α (상하 화각)는 25.6°, $2\alpha_\gamma$ (좌우 화각)은 37.0°이다. 렌즈는 VS Technology社의 SV-1214H를 사용 하였고, 렌즈의 초점거리는 $12mm$이다. 카메라에서 영상을 획득하여, 객체 인식과 추적 알고리즘을 구현하기 위해 PC를 사용하였다. PC의 사양은 CPU가 Intel Core i7 2.8GHz, RAM은 8GB이다. 영상의 획득을 위해 Visual Studio 2008을 사용하였고, 알고리즘 구현을 위해 Matlab 2014a를 사용하였다. 바닥에는 좌표확인용 grid를 $3000\times3000(mm)$ 크기로 설치하였고, 눈금 간격은 $25mm$이다. 카메라들은 삼각대를 이용하여 θ 는 33°로 바닥을 향해 사선으로 설치를 하였다. 바닥으로부터 설치된 카메라의 높이는 1220mm이다. 설치된 모든 카메라들은 grid의 중앙이 영상의 중앙을 볼 수 있도록 γ는 각각 45°,135°,225°로 세팅하였다. 그림 23은 실험 환경 전경이고, 그림 24, 25은 사용한 장비이다. 그림 26은 실제 좌표에서 카메라 배치도이고, 그림 27은 전체적인 실험 환경 배치도이다.

그림 23. 실험 환경 전경

Fig. 23. Experimental environment views

그림 24. 카메라 acA1300-60gc)

Fig. 24. Camera (acA1300-60gc)

그림 25. 삼각대 (Kepha7500B)

Fig. 25. tripod (Kepha7500B)

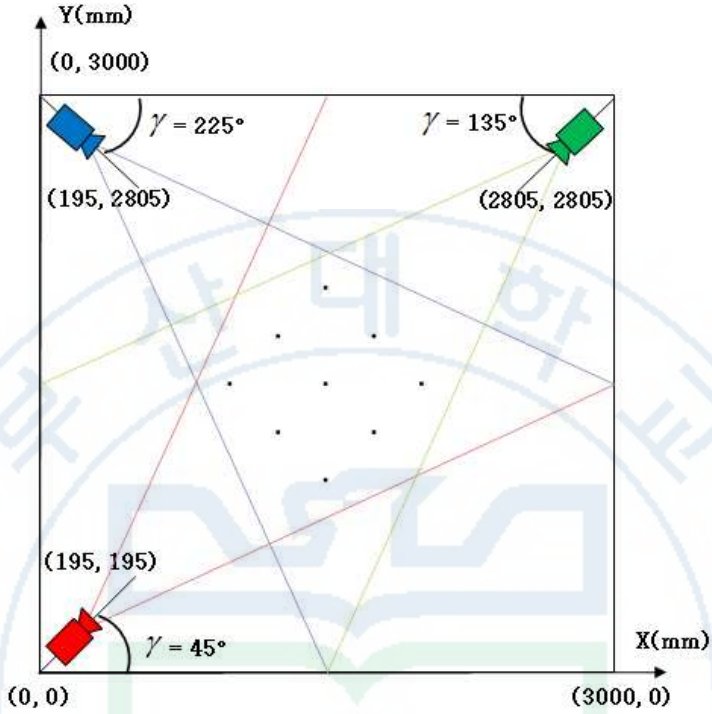


그림 26. 실제 좌표에서의 배치도

Fig. 26. Layout of real world space

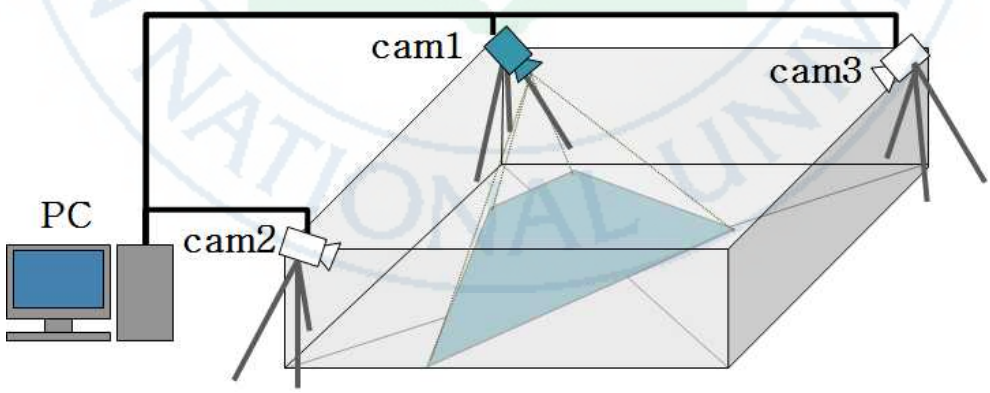


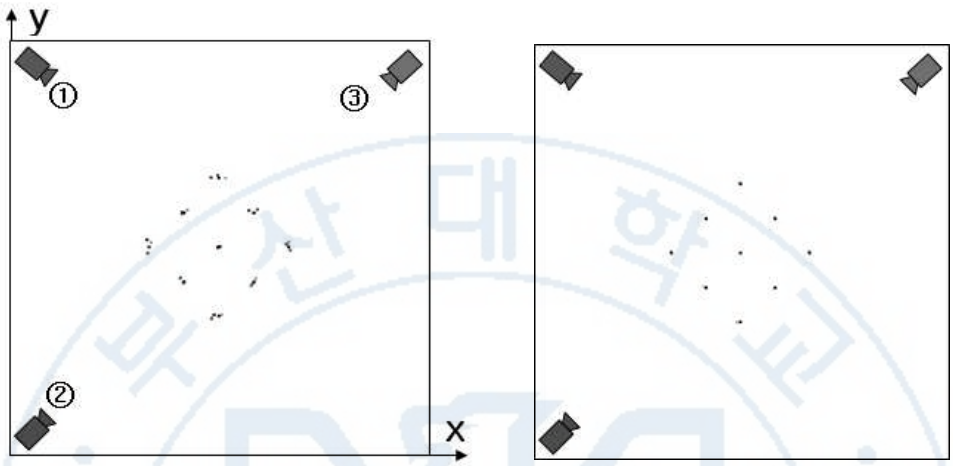
그림 27. 실험 환경 배치도

Fig. 27. Experimental environment layout

4.2. 성능 검증

알고리즘의 성능을 검증하기 위해 다음과 같은 경우에 대하여 실험을 진행하였다. 첫 번째는 카메라 개수에 따른 영상 좌표를 실제 좌표로 변환했을 때의 실험이다. 카메라의 개수가 많을수록 왜곡에 의한 영향이 줄어드는 것을 확인하여 다중 카메라 환경이 유리한 것을 검증하였다. 두 번째는 본 논문에서 제안한 알고리즘을 이용하여 단일 카메라와 다중 카메라 환경에서 이동 경로 추적을 하였고, 본 논문과 같이 비대칭으로 설치된 카메라 환경에서의 추적 결과에 대해 검증하였다. 세 번째는 바닥의 면적에 따른 객체의 위치 추적 결과이다. 객체의 바닥 모양이 다른 경우에도 일정한 성능을 보이는지 검증하였다. 네 번째는 객체가 다중 카메라 환경에서 지속적으로 인식할 수 있는 상황에서의 실험이다. 다중 카메라 환경에서 가려지지 않고, FOV내에서 이동할 경우의 성능을 검증하였다. 다섯 번째는 객체가 다중 카메라 환경에서 일부 카메라가 지속적으로 인식할 수 없는 상황에서의 실험이다. 객체가 이동하면서 FOV를 벗어나는 상황에 대해서 추적 여부를 검증하였다. 마지막 여섯 번째는 객체를 인식하면서 추정한 실제 높이에 대한 결과이다.

① 카메라 개수에 따른 성능



(a) (b)

그림 28. 9 기준 점에 대한 좌표 변환 :
(a) 단일 카메라 (b) 다중 카메라 평균

Fig. 28. Average coordinates of each camera of the 9 reference points : (a) simgle camera (b) average of multiple cameras

표 1. 9 기준점에 대한 오차 분포

Table 1. Error distributions for the 9 reference points

error (mm)	① 카메라		②카메라		③ 카메라		카메라 3대	
	x1	y1	x2	y2	x3	y3	x	y
min	-7.0	-4.0	26.0	24.0	-7.0	-2.0	-2.0	-3.0
max	-59.0	54.0	-53.0	-44.0	47.0	38.0	-25.0	14.0
mean	-26.8	17.0	-7.4	-19.0	11.4	13.0	-8.0	6.8

② 카메라 개수에 따른 성능

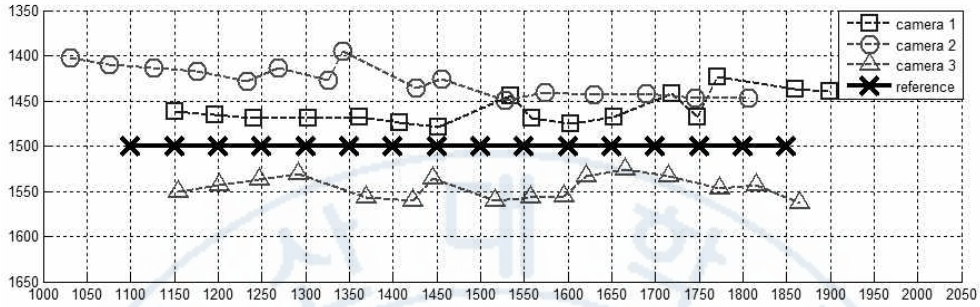


그림 29. 단일 카메라에서의 성능

Fig. 29. Performance of a single camera

표 2. 단일 카메라에서의 오차 분포

Table 2. Error distribution of a single camera

error (mm)	① 카메라		②카메라		③ 카메라	
	x1	y1	x2	y2	x3	y3
min	-21.0	21.0	-51.0	-44.0	26.0	15.0
max	-76.0	85.0	-105.0	-107.0	63.0	73.0
variance	270.5	181.1	296.5	203.1	151.2	429.2
mean	-40.4	53.6	-72.6	-72.8	45.9	41.3

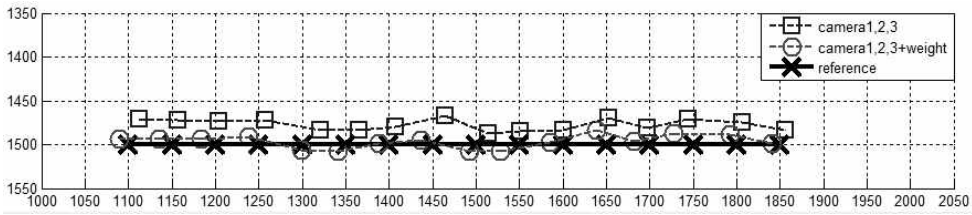


그림 30. 다중 카메라에서의 성능

Fig. 30. Performance of a multiple camera

표 3. 다중 카메라에서의 오차 분포

Table 3. Error distribution of a multiple camera

error (mm)	카메라 3대		카메라 3대 + 가중치	
	x	y	x	y
min	-13.0	0.0	-0.8	-1.0
max	-33.0	21.0	-16.2	-21.9
variance	43.4	53.0	56.9	29.3
mean	-22.7	7.0	-3.2	-12.4

③ 객체의 바닥 모양에 따른 성능

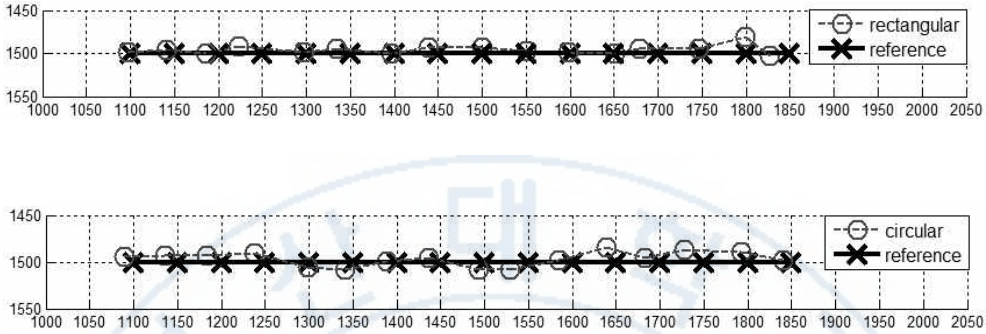


그림 31. 객체의 바닥 모양에 따른 성능

Fig. 31. Performance of the bottom of the object

표 4. 객체의 바닥 모양에 따른 오차 분포

Table 4. Error distribution according to the bottom shape of the object

error (mm)	직사각형		원형	
	x	y	x	y
min	0.0	-8.8	-0.8	-1.0
max	-18.6	-36.7	-16.2	-21.9
variance	23.8	82.8	56.9	29.3
mean	-4.3	-18.4	-3.2	-12.4

④ 모든 카메라에서 지속적으로 인식되는 경우

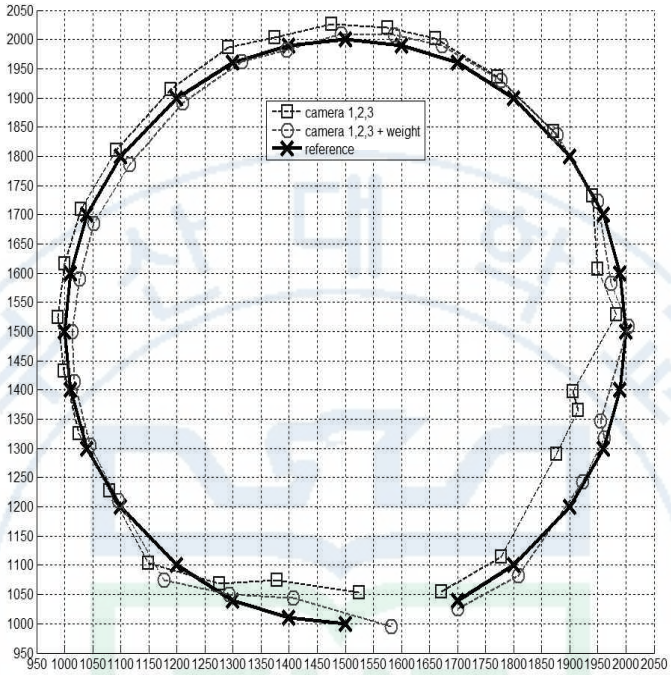


그림 32. 모든 카메라에서 지속적으로 인식될 때의 성능

Fig. 32. performance of consistently recognized when all the cameras

표 5. 모든 카메라에서 인식될 때의 오차 분포

Table 5. Error distribution when it is recognized by all cameras

error (mm)	카메라 3대		카메라 3대 + 가중치	
	x	y	x	y
min	8.0	3.0	0.6	0.0
max	84.0	90.0	82.0	52.2
variance	353.4	433.0	494.9	474.1
mean	-23.5	28.6	1.0	4.0

⑤ 모든 카메라에서 지속적으로 인식되지 않는 경우

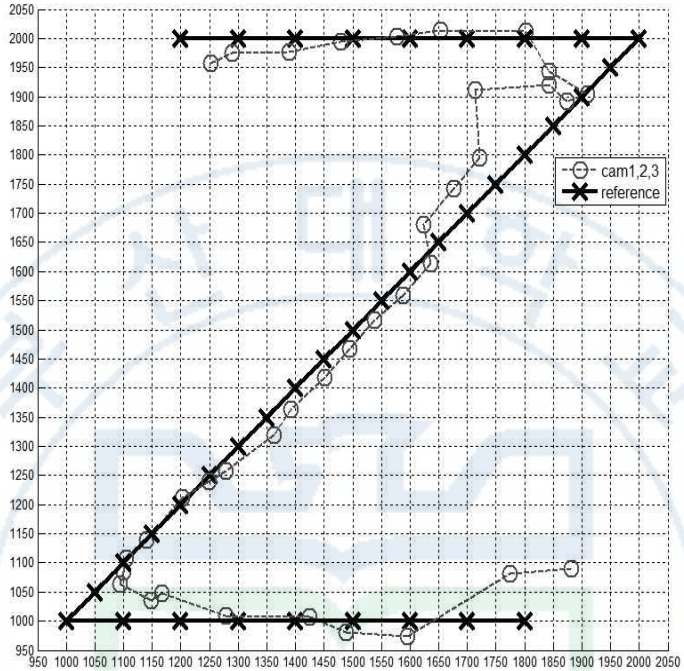


그림 33. 모든 카메라에서 지속적으로 인식되지 않을 때의 성능

Fig. 33. Performance when not consistently recognized by all of the cameras

표 6. 모든 카메라에서 인식되지 않을 때의 오차 분포

Table 6. Error distribution when it is not recognized by all cameras

error (mm)	x	y
min	0.5	2.4
max	-137.0	-95.0
variance	2387.1	1607.2
mean	-12.6	-5.0

⑥ 다양한 물체의 높이 추정

표 7. 다양한 물체의 높이 추정 결과

Table 7. Height estimation result of various objects

error (mm)	원뿔 99mm	원뿔 203mm	원기둥 98mm	원기둥 205mm
min	-10.0	-10.6	9.9	28.8
max	-27.6	-28.6	27.8	43.3
variance	23.9	25.8	25.7	21.9
mean	-16.4	-18.3	20.2	36.0

5. 결론

본 논문에서는 다수의 카메라를 이용하여 각 카메라로부터 얻은 영상에서 이동하는 객체들을 인식하고 추적할 수 있는 영상 처리 알고리즘을 제안하고 시스템을 구현하였다. 제안한 영상 처리 알고리즘은 우선 차분 영상 기법을 이용하여 이동한 객체를 배경으로부터 분할한다. 배경으로부터 분할한 객체들은 연속된 프레임에서 지속적으로 객체의 이동 벡터를 구한다. 단일 카메라에서 얻은 이동 벡터와 객체의 최하위 지점을 이용하여 다중 카메라 환경에서 객체들을 정합하여 정보를 취합한다.

구현한 시스템의 성능을 검증하기 위해 다중 카메라 환경과 단일 카메라 환경에서의 알고리즘 성능을 비교하였다. 비교 결과 단일 카메라 환경에서는 평균적으로 80mm내외의 좌표 오차가 발생하였지만, 다중 카메라 환경에서는 평균 오차가 25mm내외로 나타났다. 설치된 카메라 환경이 비대칭이기 때문에 생기는 오차를 최소화하기 위해 카메라가 없는 방향으로 가중치를 주었고, 그 결과 평균 오차가 13mm내외로 나타났다. 다중 카메라 환경이 단일 카메라 환경에 비해 성능을 보이는 것을 실험으로 확인하였다. 또한, 다양한 움직임과 물체가 단일 카메라에서 지속적으로 검출되지 않는 상황에서도 지속적으로 감시할 수 있는 알고리즘을 구현하였다.

본 논문에서는 다중 카메라 환경과 단일 카메라 환경에서의 성능을 비교하기위해 모든 카메라를 동일한 구동 조건으로 배치하였다. 추후에 각 카메라에 서로 다른 구동 조건을 적용하면, 보다 정확한 객체의 정보를 추출할 수 있을 것이다. 또한, 제안한 알고리즘을 응용하여 실시간으로 구현하면, 기존의 영상 감시 장치를 지능적으로 감시하는 장치로서 적용하는 것이 향후 과제가 될 것이다.

참고문헌

[1] 한종욱, 조현숙, “영상보안시스템 기술 동향”, 정보보호학회지, 19(5). pp.29-37. 2009년 10월.

[2] 남윤영, “지능형 영상감시 시스템의 원리 및 응용”, Jinhan M&B, p.3-4, Jan. 2011.

[3] H. Iwaki , G. Srivastava , A. Kosaka , J. Park and A. Kak "A novel evidence accumulation framework for robust multi-camera person detection", Proc. ACM/IEEE Int. Conf. Distributed Smart Cameras, pp.1 -10 2008

[4] R. Eshel and Y. Moses. Tracking in a dense crowd using multiple cameras. IJCV, 88(1):129-143, 2010. 2

[5] 장인태, et al. "다중 카메라를 이용한 실시간 객체 추적 방법." 한국산업정보학회논문지 17.4 (2012): 51-59.

[6] N. Otsu, "A Threshold Selection Method from gray-level Histogram" IEEE Trans. Systems, Man, and Cybernetics, vol. 8, pp. 62-66, 1978.

[7] B.K.P. Horn and B. Schunk, "Determining Optical Flow," Artificial Intelligence, vol. 17, pp. 185-203, 1981.

[8] S. Baker and I. Matthews, "Lucas-Kanade 20 Years on: A Unifying Framework: Part 1," Int',l J. Computer Vision, vol. 56, no. 3, pp. 221-255, 2004.

[9] J. R. Jain and A. K. Jain "Displacement measurement and its application in interframe image coding", IEEE Trans. Commun., vol. COM-29, pp.1799 -1806 1981.

[10] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation", IEEE Trans. Image Processing, vol. 9, pp.287-290 2000.

[11] ZHU Kai-hua, QI Fei-hu, “Automatic character detection and segmentation in natural scene images”, Journal of Zhejiang University SCIENCE A, Vol. 8, No. 1, pp. 63-71, 2007

[12] 김선일, “다수의 광각 카메라를 이용한 넓은 공간상의 물체 추적”, 공학석사학위논문, 부산대학교 차세대 전자기판 회로학과, 2013.

[13] M. Bertozzi and A. Broggi, "Gold: A Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection," IEEE Trans. Image Processing, vol. 7, pp. 62-81, 1998.

Position tracking of moving object using multiple cameras

Woo Hye-Jin

Department of Education Program for Samsung Advanced Integrated Circuit,
Graduate School
Pusan National University

Abstract

In this thesis, an algorithm for detecting and tracking multiple moving objects in a certain area is proposed. Multiple cameras are used in order to expand the total observable area. By strategically overlapping the sight of two or more cameras, it is possible to continuously monitor a moving object even if it leaves the sight of a single camera. The aim of this thesis, is to

implement a system to continuously monitor the moving object in specific environment. First, when a new image frame is obtained, the moving object is extracted using the difference between the new frame and previous frame. Then, pre-processing is used to subtract the background. Once the object is extracted, the lowest point and the highest point are found to determine the features of the object. The lowest point is used for estimating the position of the object whereas the highest point is used to estimate the height of the object. The following two features are used to distinguish different objects in multiple cameras environment: similarity in position of the object and similarity in motion vector of the object. Once an object is determined as an identical object by two or more cameras, the position is estimated by calculating the average of the lowest points.

In this experiment, cameras were installed in asymmetric orientation. In order to calculate the position in such environment, weights were given to each camera when computing the average in order to compensate for any bias that might occur. Also, the more overlapped area there are, the more advantageous it becomes for calculating accurate position. Based on this technique, the motion vector is used to trace the movement of an object. Implementation of the proposed algorithm for intelligent monitoring system is our further work.