論 文
38~7~9

가산 투영법을 이용한 이동물체 추적 방법

A Moving Target Tracking Algorithm Using Integral Projection

金 兌 垣*·徐 一 弘**·梁 海 元***·吳 常 綠§·任 達 鎬§§
(Tae-Won Kim · Il-Hong Suh · Hai-Won Yang · Sang-Rok Oh · Dal-Ho Im)

요 약

가산투영법을 이용하여 밝기와 모양이 변하는 움직이는 물체를 추적하는 방법을 제시하고자 한다. 계산량을 줄이며 복잡한 배경의 영향을 감소시키기 위하여 물체에 따라 크기가 자동적으로 변화하는 윈도우(window)를 사용하였다. 또한, 배경을 제거하기 위하여 전 setp에서의 물체의 밝기의 임계값과 물체의 크기를 이용하였다. 물체의 이동속도와 가속도를 이용하여 윈도우의 대략적인 위치를 추정한 후 가산투영법을 이용하여 물체의 정확한 위치를 추적하였다. 실험 결과 잡음이 들어간 실화상에서 밝기와 모양이 변하는 물체를 추적할 수 있었다.

Abstract- This paper deals with a tracking algorithm base l on integral projection which tracks moving targets with varying brightness and size. An adaptive windowing technique is employed to reduce the sensitivity of the system to the complex background image and also to reduce the computational load. The threshold value is determined by considering both the size and the threshold value of the brightness intensity of the recognized target obtained in the previous processing step. Window position is estimated by using the information of the velocity and acceleration of the target. And integral projection is applied to find the position of the target in the window accurately. Experime tal results show that moving targets with varying brightness and size can be tracked properly in noisy environments.

1. 서 론

디지탈 영상처리 기술의 발전과 더불어 시변환 영상(time varying image)에 대한 연구와 관심이 높아지고 있다. 시변환 영상 분석 문제 중에서도 특히 중요한 것은 연속 영상으로부터 움직이는 물체의 이동속도 및 이동거리 등의 정보를 얻어내는 것으로서, 차량감시나 작업자의 작업 동작 분석, 자동 영상 추적기 및 로보트 비전 등 여러 분야에

*正 會 員 : 漢陽大 大學院 電氣工學科 碩士課程
**正 會 員 : 漢陽大 工大 電子工學科 副教授·工博
***正 會 員 : 漢陽大 工大 電氣工學科 副教授·工博
§正 會 員 : 韓國科學技術研究院 電氣制御室 先任研究員·工博
§§正 會 員 : 漢陽大 工大 電氣工學科 教授·工博
接受日字 : 1989年 2月 17日
1次修正 : 1989年 6月 12日

유용하리라 기대된다.

종래의 이동정보의 검출방법은 정합법(matching method)[1, 4, 6, 7], 시공간 경사법(spatio-temporal gradient method)[8], 특징 추출법(feature extraction method)[9] 등이며 이외에도 이진영상(binary image)으로부터 이동정보를 검출하는 variation알고리즘[2, 3]과 물체의 무게중심을 이용하는 중심점 추적법[5]등이 있다.

정합법은 대상 화면에서 틀(template) 영상을 예상위치로 이동시키면서 틀영상과 대상 화면 사이의 유사성을 계산하여, 여러 예상범위 중에서 유사성이 가장 큰 값을 갖는 예상범위를 찾아내어 그 때에 두 화면이 정합되었다고 보고 이동정보를 검출하는 방법이다. 이 방법은 비교적 잡음에 강하나 배경이 복잡한 경우 성능이 떨어지며 계산량이 많은 단점이 있다. 이러한 문제를 해결하기 위하여 많은 연구가 진행되어 계산량을 상당히 줄였으나 목표물의 모양이 일정하지 않은 경우에는 적용하기 힘들다.

시공간경사법은 물체가 이동함으로써 생기는 시간상의 밝기변화와 공간상의 밝기변화 사이의 상호관계로부터 이동정보를 계산해 내는 방법이다. 이 방법은 비교적 하드웨어로 구현하기 쉬우나 속도로 빠른 경우에는 적용하기 힘들며 잡음에 약하

고자 한다. 즉, 물체가 포함되어 있는 화면의 화소(pixel)와 배경화면의 화소를 비교하여 밝기차가 일정치 이하인 경우에는 물체가 포함되어 있는 화면의 화소의 밝기를 0으로 만들어 배경을 제거하고자 한다. 그렇지만 잡음이나 그림자등 여러가지 요인 때문에 배경이 완벽하게 제거되지 않을 수가 있다. 그러므로 어떤 값 이하의 밝기를 갖는 배경화소는 제거하여 물체만이 남게 하기 위하여, histogram의 valley들에 가중치를 주어 임계값(threshold value)을 구하는 방법을 제시하고자 한다.

상기 방법을 화면 전체를 대상으로 적용한다면 계산량이 너무 많으므로 윈도우(window)를 이용하고자 한다. 특히 추적하려는 물체 주위에 물체의 크기에 따라 적절한 크기의 윈도우를 씌우고 모든 처리를 윈도우 내의 화상에 대해서만 하면 계산량이 대폭 감소한다. 또한 물체는 계속 이동하므로 이를 추적하기위해서는 윈도우가 다음 위치를 추정하여 움직이도록 해야하나 물체의 움직임이 불규칙적이기 때문에 정확한 위치 추정이 힘들다. 그렇기 때문에 추정위치에서 물체를 정확히 찾지 못한 경우에는 가산투영법을 이용하여 윈도우를 그 근처 위치로 이동시켜 물체를 찾고자 한다.

2. 가산투영법을 이용한 이동물체 추적방법

단점이 있다.

Variation알고리즘과 중심점추적법은 매우 고속화된 방법으로 실시간 처리가 가능하나, 사용되는 영상이 물체와 배경으로 분할되어있는 경우에만 적용할 수 있다. 그런데 일반적으로 원영상(raw image)을 물체와 배경으로 분할하기가 쉽지 않으며 정확히 분할하기 위해서는 복잡한 사전처리가 필요하다.

가산투영법(Integral Projection)은 화상을 임의의 방향에서 투영하여 그 투영선상에서 각 화소의 gray level값을 모두 더한 후, 그 밝기분포를 이용하여 물체를 구별하는 방법이다. 이 방법은 잡음에 강하며 계산량이 적다는 장점이 있다.

지금까지의 목표물 추적에는 모양이 변하지 않는 물체, 병진운동, 그리고 복잡하지 않은 배경에 대하여 추적하였다. [4, 5] 그러나 본 논문에서는 배경이 복잡할 뿐 아니라 모양과 밝기가 변하며 회전 및 병진운동을 하는 여러개의 물체를 가산투영법을 이용하여 추적하는 방법을 제안한다.

본 연구는 복잡한 배경에서 움직이는 물체를 추적하는 것이기 때문에 배경과 물체를 정확히 분리하기가 힘들다. 따라서 물체가 포함되어 있지 않은 배경화면을 미리 저장하여 물체 추적에 이용하

2.1 목표물체의 위치추정 및 윈도우 크기 결정

컴퓨터 기술의 발달과 더불어 움직이는 물체의 추적에 응용되는 기법 중 칼만 필터(Kalman Filter)가 가장 널리 사용되고 있다. 그러나 실제 시스템에 적용할 경우, 움직이는 물체가 가지는 비선형성을 어떻게 선형화시켜 최적성을 부여할 것인가 하는 문제와 이에 따른 많은 계산량, 디지탈 처리때 생기는 오차의 누적으로 인한 필터의 발산 및 물체의 형상에 따른 잡음 상호 분산 크기의 적절한 선택 등의 많은 문제점이 있다[11]. 이러한 문제점때문에 본 연구에서는 물체의 이동 속도와 가속도를 이용하여 물체의 다음 위치를 계산한 후 그 부분의 화면을 분석하여 물체의 정확한 위치를 찾는다.

i step에서의 물체의 중심 위치를 (x_{ci}, y_{ci})라 하면 화면에서의 물체의 평균이동 속도 및 가속도는 다음과 같이 정의할 수 있다.

$$v_{xi} = x_{ci} - x_{ci-1} \tag{1}$$

$$v_{yi} = y_{ci} - y_{ci-1} \tag{2}$$

$$a_{xi} = v_{xi} - v_{xi-1}$$

$$= (x_{ci} - x_{ci-1}) - (x_{ci-1} - x_{ci-2})$$
$$= x_{ci} - 2 * x_{ci-1} + x_{ci-2} \quad (3)$$

$$a_{yi} = v_{yi} - v_{yi-1}$$
$$= (y_{ci} - y_{ci-1}) - (y_{ci-1} - y_{ci-2})$$
$$= y_{ci} - 2 * y_{ci-1} + y_{ci-2} \quad (4)$$

여기에서 v_{xi}는 i step에서의 물체의 이동 속도의 x성분, v_{yi}는 y성분, a_{xi}는 물체의 이동 가속도의 x성분, a_{yi}는 y성분을 뜻한다.

속도와 가속도를 이용하여 i step에서 i+1 step의 위치를 추정하는 식은 다음과 같이 사용하기로 한다. (1)식에서 forward shift operator q를 이용하면

$$q v_{xi} = q(x_{ci} - x_{ci-1})$$
$$v_{xi+1} = x_{ci+1} - x_{ci} \quad (5)$$

가 된다. 이 식을 x_{ci+1}에 대해 정리하면

$$x_{ci+1} = v_{xi+1} + x_{ci} \quad (5')$$

가 된다. i step에서 v_{xi+1}을 알 수 없으므로 (3)식에 forward shift operator q를 곱하면

$$a_{xi+1} = v_{xi+1} - v_{xi} \quad (6)$$

$$v_{xi+1} = v_{xi} + a_{xi+1} \quad (6')$$

이 된다. (6')식에서도 a_{xi+1}을 알 수 없으나 화면상에서 물체의 이동 가속도는 이동 거리나 이동 속도에 비해 변화량이 적으므로 전 step에서의 값을 이용하기로 한다. 즉 (6')식을 다음과 같이 근사화하여 이용한다.

$$v_{xi+1} = v_{xi} + a_{xi} \quad (6'')$$

(6'')식을 (5')식에 대입하면 다음과 같은 근사식을 얻을 수 있다.

$$x_{ci+1} = x_{ci} + v_{xi} + a_{xi}$$
$$= x_{ci} + (x_{ci} - x_{ci-1})$$
$$+ (x_{ci} - 2 * x_{ci-1} + x_{ci-2})$$
$$= 3 * x_{ci} - 3 * x_{ci-1} + x_{ci-2} \quad (7)$$

같은 방법으로 y_{ci+1}을 구하면 다음과 같다.

$$y_{ci+1} = 3 * y_{ci} - 3 * y_{ci-1} + y_{ci-2} \quad (8)$$

한편 윈도우의 크기도 물체의 크기에 따라 가변되어야 하므로 전 step의 물체의 크기를 고려하여 변하게 해야한다. i−1 step에서의 물체의 x, y길이를 각각 O_{xli-1}, O_{yli-1}라 할 때 윈도우의 네 변의 좌표는 다음과 같이 구할 수 있다. 전 step의 길이에 0.5대신 0.6을 곱함으로써 i step에서는 윈도우가 물체 길이의 1.2배가 되어 윈도우 위치가 정확하지 않더라도 물체를 놓치지 않게 된다.

$$W_{xsi} = x_{ci} - O_{xli-1} * 0.6$$
$$W_{xli} = x_{ci} + O_{xli-1} * 0.6$$
$$W_{ysi} = y_{ci} - O_{yli-1} * 0.6$$
$$W_{yli} = y_{ci} + O_{yli-1} * 0.6$$

여기에서 W_{xsi}는 i step에서의 윈도우의 왼쪽 수직 line의 x좌표, W_{xli}는 i step에서의 윈도우의 오른쪽 수직 line의 x좌표, W_{ysi}는 i step에서의 윈도우의 위쪽 수평 line의 y좌표, W_{yli}는 i step에서의 윈도우의 아래쪽 수평 line의 y좌표이다.

W_{ysi}와 W_{yli}에서의 부호(−, +)는 screen 좌표에서의 y좌표와 world 좌표에서의 y좌표가 서로 반대이기 때문에 부호가 반대로 붙은 것이다.

실영상은 잡음이 많아 물체 크기가 실제보다 크게 인식될 수 있으므로 윈도우가 너무 크면 그 크기를 제한시켜야 한다. 이때 윈도우의 크고 작음을 판단하는 기준으로 첫번째 윈도우 설정 때 쓰인 윈도우의 크기를 원래 물체의 크기로 보고 윈도우가 이 크기보다 너무 크면 (본 연구에서는 1.5배 이상) 그 크기를 처음 설정된 크기로 제한시킨다.

```
if ((W_xli − W_xsi) > O_xl0 * 1.5){
    W_xsi = x_ci − O_xl0 * 0.5 ;
    W_xli = x_ci + O_xl0 * 0.5 ;
    }

if ((W_yli − W_ysi) > O_yl0 * 1.5){
    W_ysi = y_ci − O_yl0 * 0.5 ;
    W_yli = y_ci + O_yl0 * 0.5 ;
    }
```

여기에서 O_{xl0}는 물체의 초기 수평 길이, O_{yl0}는 초기 수직 길이이다.

이러한 단순한 추정 방법을 쓴 이유는 계산이 간단할 뿐 아니라 물체의 이동이 불규칙적으로 나타나므로 정확하게 추정하는 것이 불가능하기 때문이다. 이러한 이유 때문에 윈도우가 물체를 모두 포함하고 있는지 판단하여 위치를 조정해 줄 필요가 있다. 이를 위해서는 윈도우 내의 배경을 제거한 후 가산투영법을 이용하여 물체의 정확한 위치를 찾아야 한다.

2.2 윈도우 내의 배경 및 잡음 제거

실 영상에서 움직이는 물체를 추적하는 경우, 배경이 복잡하고 그림자와 햇볕에 의한 반사때문에 물체의 밝기와 모양이 수시로 변한다. 이 화상에서 일정한 임계값으로 배경을 제거하기는 거의 불가능하다. 배경을 효과적으로 제거하기 위하여 임계값을 주어진 환경에 따라 변화시켜야 하므로 전 step의 물체 크기와 임계값을 이용한다.

두개의 화면을 저장할 수 있는 digitizer를 이용하여 한쪽 memory(MEM_B)에는 물체가 포함되어 있지 않은 배경화면을 저장시키고, 다른 한쪽 memory(MEM_A)에는 물체가 포함된 화면을 연속적으로 받아 들인다.

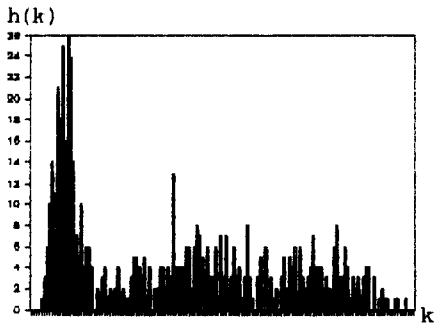
MEM_A와 MEM_B의 각 화소값을 비교하여 그 차이가 일정치 (본 연구에서는 15) 이하이면 두 화소가 같은 밝기인 것으로 간주하여 MEM_A의 해당 화소의 밝기값을 0으로 만든다. (즉, 검정색으로 만들어 화면에서 제거시킨다.) 이와같이 하면 MEM_A는 배경 화면과 밝기가 비슷한 곳은 모두 지워지고 거의 물체만이 남게 된다. 화면 전체를 대상으로 이러한 계산을 하게 되면 계산량이 너무 많으므로 물체 주위에 윈도우를 씌워서 계산량을 줄인다.

그렇지만 조명변화와 그림자 등의 이유때문에 배경이 완벽하게 제거되지 않으므로 물체 주의에 씌운 윈도우내의 밝기 정보를 이용하여 histogram을 구하고, histogram상에서 임계값이 될 수 있는 후보점을 찾아 그 중 가장 적절한 임계값을 찾고자 한다. 원영상(raw image)에는 잡음이 섞여 있을 수가 있기 때문에 peak와 valley가 많이 나타나므로 원하는 후보점을 잘못 찾을 수도 있고 후보점이 많으면 계산량도 많으므로 후보점 갯수를 줄일 필요가 있다. 따라서 histogram의 유효정보는 유지하면서 잡음이 섞인 histogram의 peak와 valley를 부분적으로 제거해 주기 위하여 다음과 같은 가중치를 고려한 평균을 이용한다.

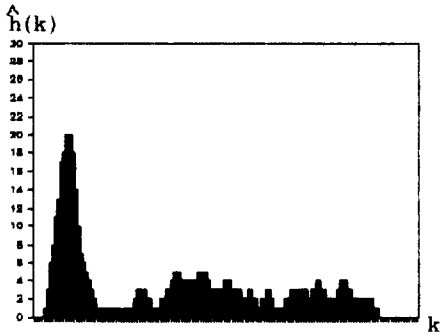
$$\hat{h}(k)=\{h(k-2)+2*h(k-1)+5*h(k)+2*h(k+1)+h(k+2)\}/11 \quad (9)$$

여기에서 k는 gray level, $h(k)$는 gray level이 k인 화소의 갯수, $\hat{h}(k)$는 평균치이다.

위와 같은 방법으로 평활화(smoothing)된 histogram(그림 1(b) 참조)에서 $\hat{h}(k)$가 0인 곳과 valley의 오른쪽 끝부분을 후보점으로 한다. 이때 valley가 긴 경우는 valley 중간과 오른쪽 끝부분을 모두 후보점으로 한다. 이 후보점들과 전 step의 임계값을 비교하여 전 step의 임계값과의 거리



(a) 원래 histogram
(a) Original histogram



(b) 평활화된 histogram
(b) Smoothed histogram

그림 1 histogram 평활화
Fig. 1 Histogram smoothing

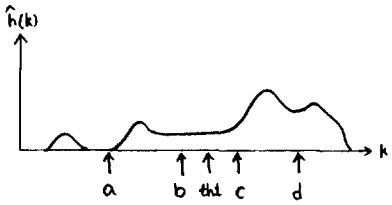


그림 2 임계값 후보점
Fig. 2 Candidates of threshold value

가 가까울수록 큰 가중치를 두며, 거리가 일정치 보다 멀면 후보점에서 제외시킨다.

Th_{old}=전 step의 임계값
C_i=i번째 임계값 후보점의 값
$All\ W_i=1$ ($_i$=1, 2, ……, N)
N은 후보점 갯수
if ($|TH_{old}-C_i|<7$) $w_i=w_i*2.0$
else if ($|TH_{old}-C_i|<15$) $w_i=w_i*1.5$
else $w_i=0$

또한 후보점 이상의 면적을 전 step의 물체 크기와 비교하여 그 크기가 전 step의 물체 크기에 가까울수록 큰 가중치를 두고, 크기가 너무 크거나 너무 작으면 후보점에서 제외시킨다.

A_{old}=전 step의 물체의 크기
A_i=i번째 임계값 후보점의 값보다 큰 값을 갖는 화소의 갯수의 합
if ($|A_{old}-A_i|<A_{old}*0.2$) $w_i=w_i*2.0$
else if ($|A_{old}-A_i|<A_{old}*0.4$) $w_i=w_i*1.5$
else $w_i=0$

이상과 같은 과정을 거친후 가중치가 가장 큰 후보점을 이용하여 thresholding하면 윈도우내의 배경은 제거된다.

그림2에서 임계값 후보점은 높이가 0인 a와 valley인 c, d만이 될 수 있으나 c가 위치한 valley는 기준치(본 연구에서는 15)보다 길기 때문에 중간점인 b도 후보점이 되었다.

전 step에서의 임계값이 b와 c사이인 th1이었다면, b와 c는 th1과 사이가 가까우므로 가중치가 크게되며 a와 d는 거리가 멀므로 가중치가 작게된다. 즉, 전 step의 임계값만을 고려한 경우 b와 c가 새로운 임계값이 될 가능성이 높게 된다. 두

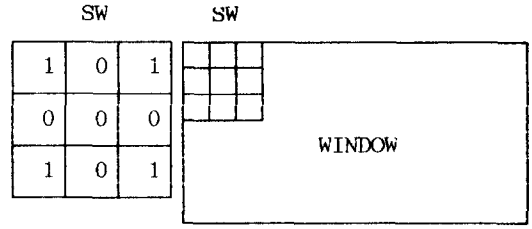


그림 3 잡음 제거에 사용되는 sub-window
Fig. 3 Sub-window for removing the noise

값중 어느것이 적절한 임계값인지 결정하기 위하여 전 step에서의 물체 면적을 고려해야 한다. a보다 큰 gray level을 갖는 pixel의 합을 A, b보다 큰 경우는 B, ……라 할때, 전 step에서의 물체 면적이 C와 D사이였다면 후보점 c와 d는 큰 가중치를 갖게 되고, a는 b는 작은 가중치를 갖게 된다. 전 step의 임계값과 면적을 고려한 경우, 가장 큰 가중치를 갖는 c가 가장 적절한 임계값이 된다.

앞에서 구한 값을 이용하여 thresholding을 하더라도 잡음의 영향 때문에 실제 화상에서는 작은 점이나 blob 등이 남아있을 수 있기 때문에 이러한 것을 제거할 필요가 있다. 이를 위하여 그림3과 같은 3×3sub-window를 이용하여 윈도우 내를 이동시켜가며 잡음을 제거한다. sub-window와 window 내의 각 화소들을 논리연산시켰을 때, 네 귀퉁이가 모두 0인 경우에는 가운데 십자 부분의 화소들을 모두 0으로 만든다. 이 sub-window를 이용하면 1 pixel폭의 선(line) 잡음이나 점은 제거된다. 윈도우가 큰 경우에는 6×6나 9×9 sub-window로 만들면 작은 blob도 제거된다.

if ((SW · WINDOW)=all zero)
then WINDOW[i][j]=0
· 는 logical AND

배경 및 잡음 제거 방법을 간략히 표현하면 다음과 같다.

*Algorithm : REMOVE BACKGROUND AND NOISE
Subtract Background Image from Current Image
Make Histogram
Histogram Smoothing
Find Candidate Points on the Histogram
Calculate Weight Value of the Candidate Points

using last Threshold Value
Calculate Weight Value of the Candidate Points
using last Area
Find Maximum Weight Value
Thresholding
Remove Isolate Points and Lines

2.3 가산투영법을 이용한 윈도우 위치보정

가산 투영법은 화상을 임의의 방향에서 투영하여 그 투영선상에서 각 화소가 갖고있는 gray level 값을 모두 합하는 것이다. 각 방향의 가산투영은 그 방향의 화상의 밝기분포를 나타내므로 가산 투영법은 물체들을 구별하는데 사용할 수 있다.[4]

가산투영법의 장점은 잡음에 강하다는 점이다. 모든 자연계의 신호는 잡음을 고려하지 않을 수 없으며, 더군다나 실 영상과 같이 배경이 복잡한 경우에는 그 영향을 무시할 수 없다. 이러한 화상에서 물체를 추적하기 위해서는 잡음을 완벽히 제거하는 알고리즘이나 잡음에 강한 추적방법이 필요한데 가산투영법은 적분의 개념을 이용한 것이므로 잡음에 강하다.

또한, 가산투영법은 2차원적인 계산을 1차원적인 계산으로 줄일 수가 있어서 계산상의 많은 이점이 있다. 중심점 추적 방법[5]도 계산량이 적으면서도 비교적 우수한 추적 성능을 보이나 배경과 물체가 명확히 분할되어 있어야 하며, 물체의 일부만을 가지고 계산할 경우 잘못된 중심점을 찾게 된다.

본 연구에 쓰이는 화상은 운동물체의 움직임이 불규칙적이기 때문에 정확한 이동 위치 추정이 힘들며, 배경이 복잡하므로 추적 윈도우를 너무 크게 하면 물체 이외의 부분이 남게되어 중심점 추적 방법으로는 엉뚱한 곳을 찾게될 가능성이 높아 가산 투영법을 적용하였다.

또한 가산투영법을 이용하면 탐지된 물체의 크기와 위치를 알 수 있기 때문에 윈도우가 물체를 모두 포함하고 있는 지의 여부를 판단하여 필요한 방향으로 윈도우를 이동시킬 수 있다.

배경 및 잡음을 제거한 화상에 대하여 수직과 수평 방향으로 가산투영시킨다. 일반적인 경우에는 가산투영시킨 값이 어느 일정치보다 큰 부분을 물체로 인식하여 그 가운데 점을 중심점으로 간주해도 되나 실제 영상은 잡음이 심할뿐아니라 윈도우의 위치가 정확하지 않기 때문에 중심점이 잘못 인식될 수 있다.

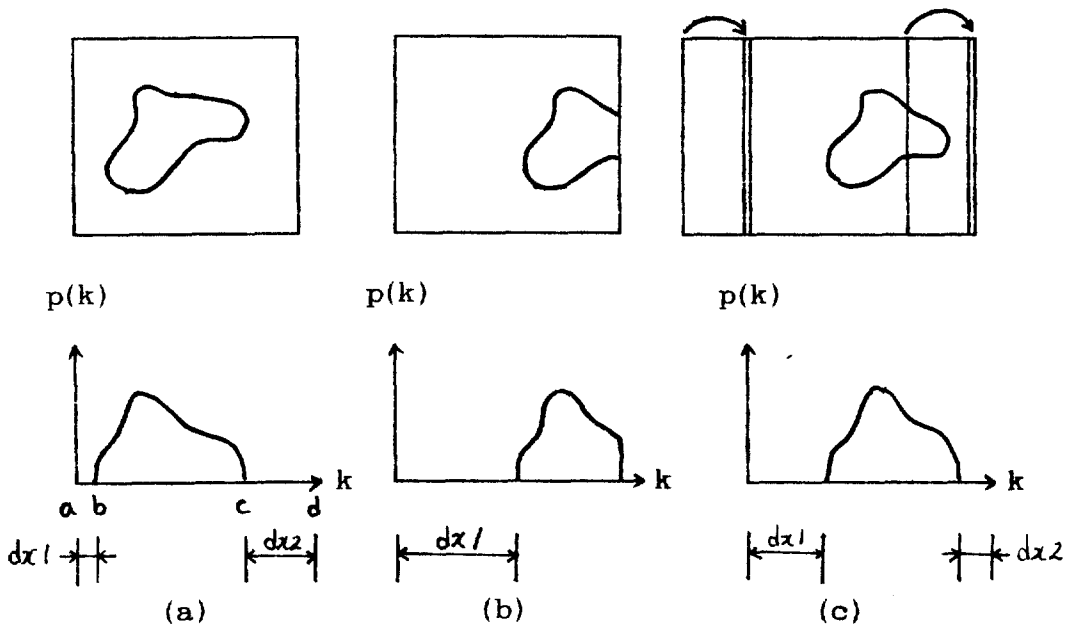


그림 4 윈도우 위치 보정 (a) 위치 추정이 잘 된 경우
(b) 위치 추정이 잘못된 경우 (c) 위치 보정 후 윈도우 이동

Fig. 4 Reset the window position (a) Correct estimation
(b) Incorrect estimation (c) Window shifting

배경이 단순한 경우라면 윈도우를 크게하여 목표물이 윈도우 밖으로 벗어나는 것을 방지할 수 있으나, 배경이 복잡한 경우에는 불필요한 부분이 물체와 함께 남아 있게 된다. 이런 경우를 방지하기 위해서 본 연구에서는 윈도우를 필요이상 크게 하지 않았으며, 검출된 물체가 윈도우 외곽에 위치할 경우 윈도우를 물체쪽으로 이동시켜서 물체를 검출하게 한다. 이와 같이 하면 추정된 윈도우 위치가 정확하지 않은 것을 보정해 줄 수 있다.

그림 4(a)는 윈도우의 추정 위치가 정확한 경우의 예이다. 그림 4(b)는 윈도우의 위치 추정이 정확하지 않아 윈도우내에는 물체의 일부만이 탐지된 예이다. 그림 4(a)에서 윈도우의 왼쪽 변의 x 좌표를 a, 오른쪽 변의 x좌표를 d, 가산투영시킨 값이 시작되는 점을 b, 끝나는 점을 c라 할때,

$$dx1 = b - a ;$$
$$dx2 = d - c ;$$
$$O_{xli} = c - b ;$$

라 정의할 수 있다. 그림 4(b)의 경우, $dx1$은 충분히 큰데 비하여 $dx2$는 0이고 물체의 크기(O_{xli})도 초기 크기(O_{xlo})보다 작으므로 물체가 윈도우의 오른쪽에 비껴 있음을 알 수 있다. 윈도우의 위치를 전체적으로 $1/2 * dx1$만큼 오른쪽으로 이동시킨 것이 그림 4(c)이다. 이와같이 가산투영시킨 값을 이용하면 윈도우 위치가 정확하게 추정되지 않은 것을 보완해 줄 수 있다. 위치보정이 끝난 후 다시 한번 더 투영하여 물체의 위치를 찾고 그 중앙을 물체의 중심으로 간주하여 윈도우의 다음 위치 추정에 이용하였다.

이상의 물체 탐지 알고리즘은 다음과 같다.

```
*Algorithm : DETECT OBJECT
 do {
   Integral Projection
   Find Length of Object
   Find Center of Object
   L1←Current Length of Object
   L2←Last Length of Object
   D1←Start point of Object−Start Point of
       Window
   D2←Last Point of Window−Last Point of
       Object
   if (D1<2 AND L1< L2)  Shift Window to
       Left Direction
   if (D2<2 AND L1<L2) Shift Window to
       Right Direction
 }while (D1<2 OR D2<2)
```

2.4 목표물체의 확인

이상의 처리과정을 거치더라도 윈도우내에 물체 이외의 다른 부분이 남아있는 경우 그 부분이 물체로 잘못 인식될 수 있으므로 실제 찾은 부분이 원하는 물체인가를 확인하는 과정이 필요하다. 물체의 중심점 주위에 7×7 윈도우를 씌운 후 그 윈도우 내의 평균밝기를 구하여 그 전 단계의 평균값과의 밝기 차이가 일정치보다 작으면 원하는 물체로 인식하고 일정치보다 크면 물체를 잘못 찾은 것으로 인식하여 원래 윈도우 내에서 현재 물체로 인식되어 있는 부분을 제거한다. 윈도위 내에서 다시 가산투영법을 적용하여 물체를 찾게되면 처음 잘못 인식되었던 곳은 제거되었으므로 실제 물체를 찾을 수 있다. 이때에도 원하는 물체인가 확인하게 된다. 이와같이 하여도 물체를 찾을 수 없는 경우는 배경이 제거되지 않는 원래의 화면으로 환원시킨 후, computer의 keyboard(arrow key)를 이용하여 화면에 나타난 십자모양(+)의 표시기를 물체의 중심에 일치하게끔 조정하여 사용자가 직접 물체의 중심 위치를 정해주게 한다.

$$AVE_t = \{ \sum_{i=X_c-3}^{X_c+3} \sum_{j=Y_c-3}^{Y_c+3} L(i, j) \} / 49$$
$$|AVE_t - AVE_{t-1}| < Th. \rightarrow True$$
$$|AVE_t - AVE_{t-1}| \geq Th. \rightarrow False$$

전 단계와 비교하여 물체의 움직임이 너무 많거나 밝기 변화가 심한 경우, 또는 중심점의 위치가 일정 한계 밖으로 벗어날 때에는 사용자로 하여금 확인하게 한다. 엉뚱한 물체를 추적한 것이되면 사용자가 직접 물체의 중심을 표시하여 주게된다.

이상의 물체 확인 알고리즘은 다음과 같다.

```
*Algorithm : VERIFY OBJECT
   Calculate Average Brightness of 7×7Window
   on the Center Point
   if(abs(Current_AVE− Last_AVE)< Th.)
     Result←True
     else {
          Result ←False
          Remove Current Object Area
          Iterate All Process
          }
   if(Can't find correct object)
     User Mode
```

지금까지의 전 과정을 요약하면 다음과 같은

algorithm으로 기술할 수 있다. 여기에서 'Initial Window Setting'때는 크기 및 위치를 가변시킬 수 있는 표시기를 이용하여 화면에서의 물체의 초기 위치와 크기를 지정해 준다(그림 6(b) 참조). 또한 이상의 과정을 여러 물체에 대하여 각각 적용시켜 독립적인 추적이 가능하게 한다.

```
*Algorithm : MOVING TARGET TRACKING
    System Initialization
    Take Background Image
    Take Current Image
    Initial Window Setting
    while(step<max_frames) {
        Take Current Image
        for i : =1 to No_of_objects {
          Estimate Displacement Using Last
          Data
          Move Window to Estimated Displace-
          ment
          Fing Threshold Value Using Last Data
          and Histogram
          Thresholding
          if (Brightness of Background Image>
          Brightness of Current Image)
          Recover Object
      do {
        Integral Projection
        Remove Isolate Points or Lines
        Find Object
        if (Window position is not correct)
            Adjust Window Position
        Result←Verify Object
        if (Result=Uncorrect Object)
            Delete current Object
        } while(Result=Correct Object)
      Store Date
    }
}
```

3. 실험 및 고찰

이상의 물체 추적 알고리즘을 체육관에서 달리는 선수를 찍은 VTR을 이용하여 그 선수의 관절주위에 부착한 밴드를 추적하는데 적용하였다. 본 실험은 Panasonic AG-6300 VTR을 이용하였으며, PC Vision plus를 장치한 IBM-PC/XT를 이용하여 알고리즘을 수행하였다.

연구에 사용된 digitizer인 PC Vision plus는 화면을 저장할 수 있는 memory가 둘이기 때문에 한쪽 memory(MEM_B)에는 물체가 포함되어 있지 않는 배경화면을 저장시키고, 다른 한쪽 memory (MEM_A)에는 물체가 포함된 화면을 연속적으로 받아 들인다.[12]

본 연구는 물체가 주변 배경화면이나 선수의 다른 신체부위보다 밝다는 가정하에 실험되었다. 물론 예외적인 경우도 발생하지만 대부분 이 가정을 만족시킬 수 있었고 그 결과는 그림6~12에 나타내었다. 그림에서의 기준궤적은 십자모양의 표시기를 이용하여 매 step마다 밴드의 중심점을 표시한 것이다.

그림 6에서 볼 수 있듯이 물체(밴드)가 배경화면의 밝은 부분을 지나는 경우가 생기는데 이는 앞에서한 가정에 어긋나므로 정상적인 추적이 힘들다. 즉, 배경중에 있는 흰 물체와 흰 밴드가 교차하게 될 경우, 배경화면을 이용하여 불필요한 배경을 제거시키며 배경과 밴드가 유사한 밝기이므로 밴드도 함께 제거되어 원하는 밴드를 탐지할 수 없다. 이 화면에서 사라진 배경은 그대로 두고 밴드만을 복원시켜야 한다.

배경과 밴드가 제거된 화면에 대하여 기존의 윤곽선 검출 방법[10]을 이용하여 관절의 윤곽선을 찾는다. 이 윤곽선의 왼쪽 끝에서 수직으로 선을 긋고, 오른쪽끝에서 수직으로 선을 그어 생기는 폐곡면을 사라진 밴드로 보고 그 부분의 밝기를 배경의 평균 밝기로 만든다. 이때 사라진 밴드의 밝기는 배경의 평균 밝기와 거의 같으므로 원래의 밴드로 환원시킨 것과 같은 효과를 가진다. 이 윈도우 내에 가산 투영법을 적용시키면 원하는 밴드를 찾을 수 있다.

이러한 방법은 밴드가 관절주위에 부착되어 있고 폐곡면이 생길만큼 충분히 큰부분은 무릎관절이나 허리부분밖에 없다는 사실을 이용하여 고안된 것이다. 그렇기 때문에 수직의 선을 그어 폐곡면을 만들 수 있었으나 밴드를 감은 관절부분이 수평으로 되며 밴드와 같은 밝기의 배경과 교차하는 경우에는 수평의 선을 그으면 같은 식의 폐곡면을 만들 수 있다. 물론 이런 식으로 구한 중심점이 실제 물체의 중심점과 정확하게 일치한다고 볼 수는 없지만 거의 비슷한 위치는 찾을 수 있으며 추적이 불가능한 상태가 되는 것을 방지할 수 있다.

이때 배경이 제거된 윈도우의 평균 밝기와 배경의 평균 밝기를 비교하면 밴드와 배경의 밝기가 거의 같다는 사실을 알 수 있다. 즉, 배경의 평균 밝기가 윈도우의 평균 밝기보다 밝다는 것은 밴드

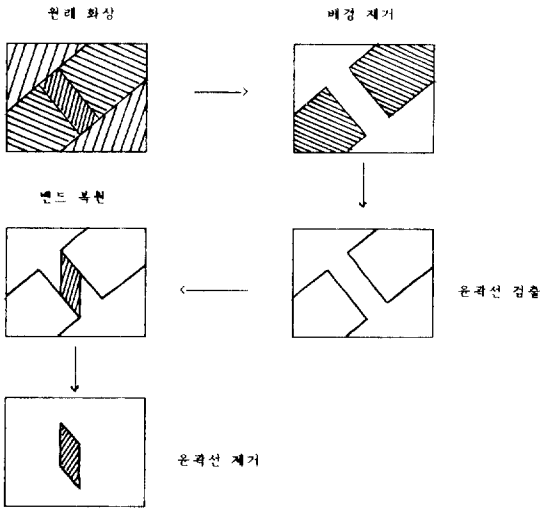


그림 5 밴드 복원 개요도
Fig. 5 Schematic diagram of band recovery

와 배경의 밝기가 거의 같다는 것이므로 위의 과정이 필요하다.

그림 6(a)는 실험에 쓰인 배경화면, (b)는 초기 윈도우 설정 화면, (c)는 밴드를 추적하는 과정이며 (d)는 추적한 결과를 화면에 나타낸 것이다.

이 궤적들을 기준값들과 비교하여 나타낸 것이 그림 7, 8, 9이다. 그림에서 보면 추적한 궤적과 기준점 사이에 약간의 오차가 나타났는데 실제의 경우 480×512의 화상에서 2, 3pixel 차이는 거의 구별하기 힘들므로 정확히 추적한 것으로 간주하였다. 그림 7은 운동선수의 팔꿈치에 부착한 밴드를 추적한 결과이다. 12step의 시행중 4회는 computer가 정확히 추적하지 못하였고, 그 중 2회는 오차가 너무 심해 인위적으로 위치를 조정해 주었다. 그림에서 보듯이 2 step, 3 step 수행때 정확한 위치를 찾지 못했는데, 이는 팔꿈치 밴드가 허리의 밝은 부분 근처를 지나갈 때 허리부분을 추적하는 밴드로 잘못 탐지했기 때문이다. 또한 6 step과 11 step도 잘못 찾았는데, 배경중에 팔꿈치 밴드와 밝기가 거의 같은 부분이 가로 질러 있어서 그 높이를 지날 때는 항상 잘못 탐지한 것이다. 11 step 수행때는 밴드가 그림자에 가려 인간의 눈으로도 거의 식별하기 힘들었다. 이 궤적을 xy좌표상에 나타낸 것이 그림9(a)이다. 그림에서 보듯이 팔꿈치가 밑으로 내려갔을 때 허리부분을 탐지했음을 알 수 있다.

그림 8은 손목에 부착한 밴드를 추적한 결과이다. 이 경우에도 손목이 허리부분을 지날 때 잘못 탐지하고 있음을 알 수 있다. 손목밴드의 경우 팔꿈치밴드와는 달리 밝기가 좀더 밝아서 일정 높이에서 항상 잘못 탐지하는 일은 일어나지 않았다. 이와같이 추적하는 물체와 유사한 밝기를 가진 물체가 주변에 있을 때 어느 것이 추적하려는 진짜

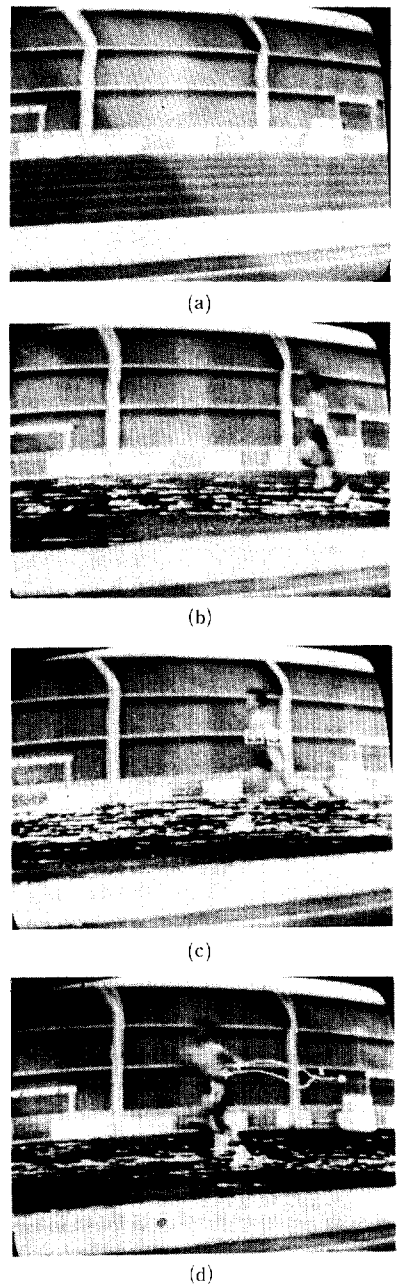
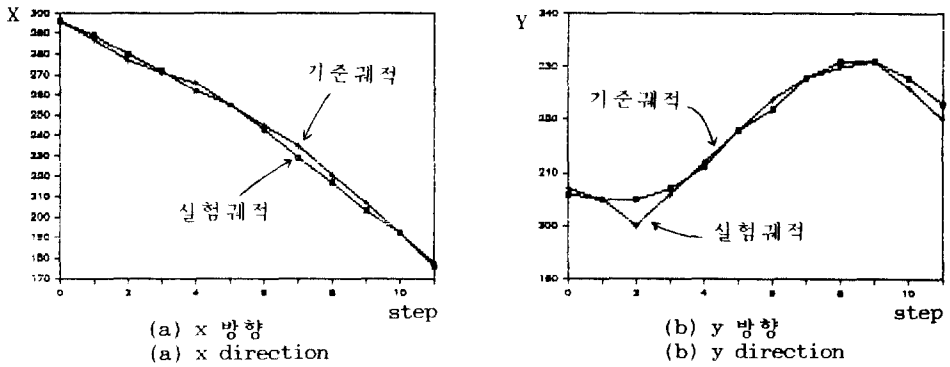
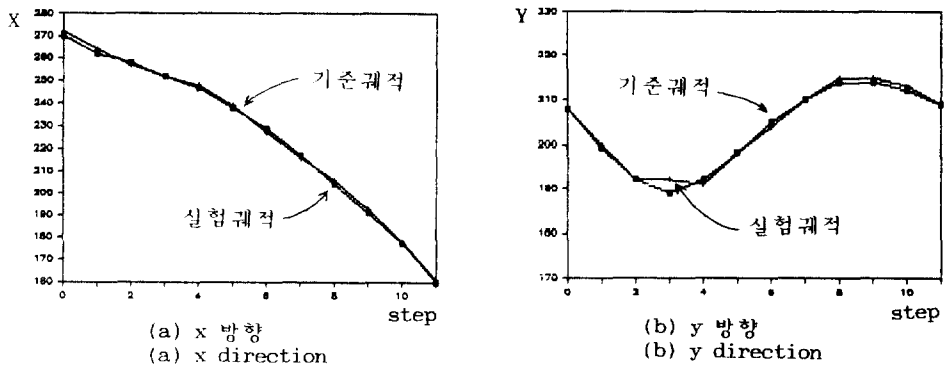


그림 6 실험에 사용된 화상
Fig. 6 Image for the experiment



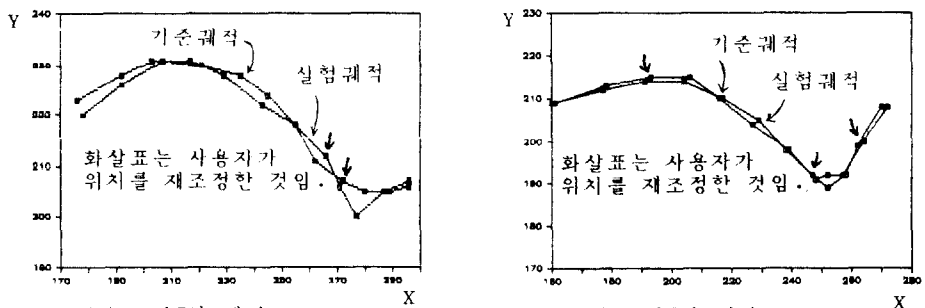
(a) x 방향 (a) x direction　　(b) y 방향 (b) y direction

그림 7 팔꿈치 밴드의 추적 성능(가산투영법)
Fig. 7 Tracking performance of elbow band(Using Integral Projection)



(a) x 방향 (a) x direction　　(b) y 방향 (b) y direction

그림 8 손목 밴드의 추적 성능(가산투영법)
Fig. 8 Tracking performance of wrist band(Using Integral Projection)



(a) 그림7의 궤적 (a) Trajectory of Fig. 7.　　(b) 그림8의 궤적 (b) Trajectory of Fig. 8.

그림 9 xy좌표에 대한 추적 성능(가산투영법)
Fig. 9 Tracking performancd on XY coordinate

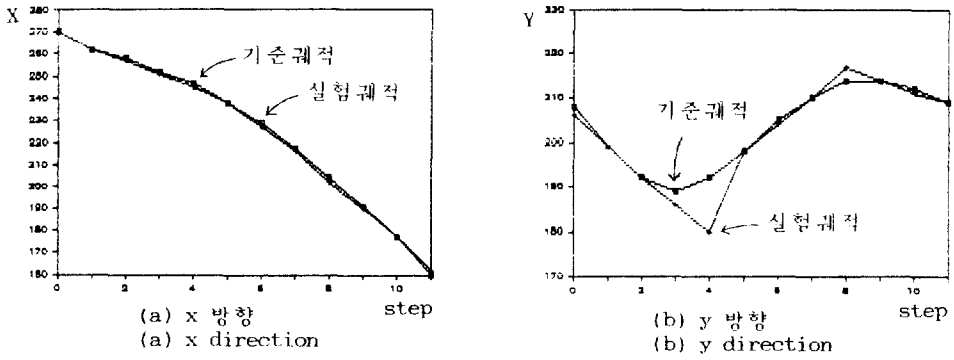


그림 10 팔꿈치 밴드의 추적 성능(중심적 추적법)
Fig. 10 Tracking performance of elbow band(Using Centroid Tracking)

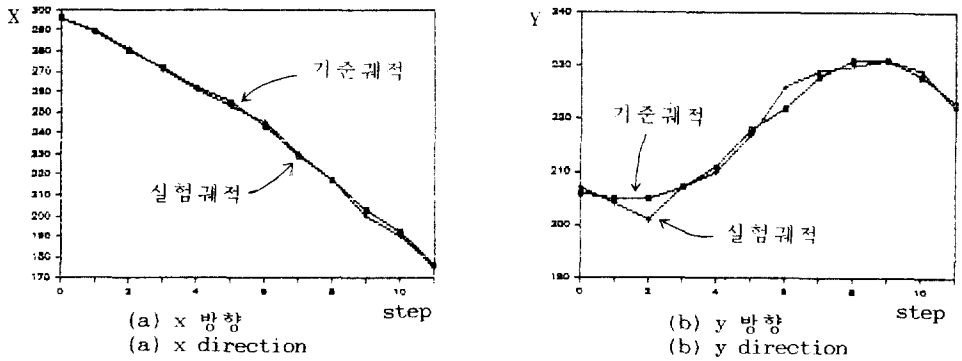


그림 11 손목 밴드의 추적 성능(중심적 추적법)
Fig. 11 Tracking performancd of wrist band(Using Centroid Tracking)

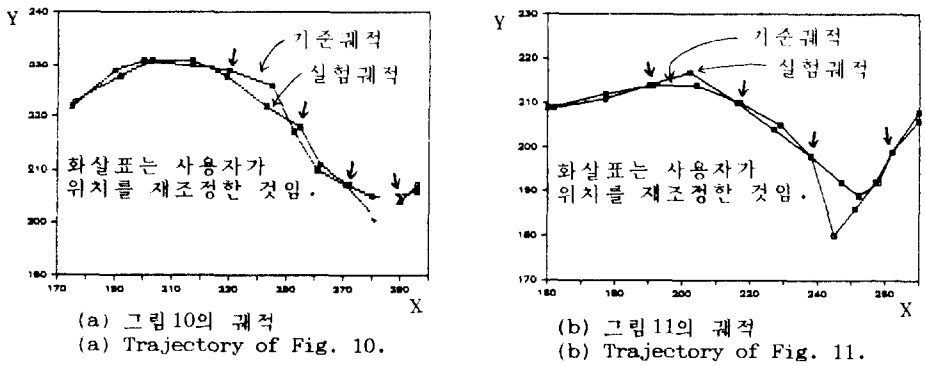


그림 12 xy좌표에 대한 추적 성능(중심적 추적법)
Fig. 12 Tracking performance on XY coordinate(Using Centroid Tracking)

물체인가 확인하는 방법이 요망된다.

같은 화상에 대하여 중심점 추적법을 적용하여 추적한 결과가 그림 10, 11, 12이다. 가산투영법을 이용하여 추적한 경우에는 물체의 크기에 따라 윈도우의 크기를 변화시켰으나, 중심점 추적법을 이용하는 경우에는 윈도우 크기를 변화시킬 수 없으므로 처음 설정된 길이의 1.5배로 고정시켰다. 윈도우의 크기가 너무 작으면 위치 추정이 약간만 잘못되어도 윈도우 내에는 물체의 일부분만이 남게되어 물체의 중심을 정확히 찾지 못하게 되거나 물체를 제대로 추적하지 못하게 되며, 윈도우가 너무 크면 필요이상의 부분이 물체와 함께 윈도우 내에 남게되어 중심점 계산이 정확하지 않게된다.

그림 10의 1, 2, 3, 5, 6, 7 step과 그림 11의 1, 4, 5, 7, 8, 9 step은 중심점을 잘못 찾은 경우이다. 중심점을 잘못 찾았음에도 불구하고 기준좌표값과 일치하는 것은 computer에서 잘못 찾은 것을 검출하여 (2.4절 참조) 사용자가 정확한 위치를 재설정하게 하였기 때문에 그림상에서는 거의 오류가 없게 보인다. 1 step은 이전 자료가 없기 때문에 다음 위치를 추정하지 못한데서 생긴 오류이다. 중심점 추적법에서 검출된 오류의 특이한 점은 물체의 이동이 많거나 움직임 변화가 심한 경우에 윈도우의 위치 추정이 정확하지 않아 연속적으로 오류가 생긴다는 점이다. 가산투영법을 이용하였을 때는 약간의 오류는 스스로 보정하여 그 오차를 줄임으로서 다음 step에서의 추적에 무리가 없게 하였으나 중심점 추적법은 정해진 윈도우 내에서만 계산하므로 오차가 누적되어 결국 사용자가 조정해 주어야만 한다. 또한, 두 경우 모두 비슷한 위치에서 오류가 생기는 것으로 보아 중심점 추적법에서도 물체와 비슷한 밝기의 주변 배경에 영향을 받아 중심점이 부정확하게 계산되었음을 알 수 있다. 그러나 가산투영법은 유사한 밝기의 물체가 주위에 있을 경우에는 스스로 위치를 옮겨 목표물체를 잘못 추적할 가능성이 있으나 중심점 추적법은 계산된 중심점이 항상 윈도우 내에 있으므로 약간의 오차는 생길지라도 중심점이 옮겨가는 경우는 안생긴다.

배경화면이 변하게 되는 경우에는 물체 추적에 어려움이 따르나 배경화면에서 물체보다 더 밝은 부분이 물체와 교차하여 움직이지만 않는다면 큰 문제가 되지 않는다. 참고로 그림에서의 y좌표는 화면의 왼쪽 아래 끝점을 원점으로한 world좌표로 변환된 것이다.

4. 결 론

일반적인 기존의 목표물 추적 방법은 배경이 단순하고 모양이 일정한 물체만 추적이 가능하다. 그러나 본 연구에서는 가산투영법을 이용하여 복잡한 배경하에서 모양과 밝기가 변하며 움직이는 물체를 추적하였다.

배경제거를 위하여 전 step에서의 물체의 임계값과 면적을 이용하여 다음 step에서의 임계값을 정한 것은 연속 영상에 있어 상당한 효과가 있었다. 또한 가산투영법을 이용함으로써 잡음의 영향을 줄이며 추적 윈도우의 위치를 보정해 줄 수 있었고, 추적 윈도우 크기가 필요 이상 크지 않으므로 계산량이 적게 되었다.

그러나 추적하려는 물체가 배경이나 다른 물체에 가린 경우에 대한 대책이 없으며 유사한 밝기의 물체가 주위에 있을 경우 잘못 추적할 가능성이 높다. 이에 대한 계속적인 연구가 필요하다.

이 논문은 1986년도 문교부 대학 부설 연구소 지원 학술연구조성비에 의하여 연구되었음.

참 고 문 헌

[1] G.J. Vanderbrug & A. Rosenfeld, "Two-Stage Template Matching," IEEE Trans. Comp., vol. C-26, no. 4, pp. 384-393, April 1977.

[2] 김경수, 이상욱, 송유섭, "Fast 2-D moving target tracking algorithm," 전자공학회 논문지. 제22권 1호, pp. 75-85, 1985. 1.

[3] G.R. Legters JR., T.Y. Young, "A mathematical model for computer image tracking," IEEE Trans. PAMI, vol. PAMI-4, pp. 583-594, 1982.

[4] 김근형, 박래홍, "가산 투영법을 사용한 Two-Stage Template Matching," 대한 전자공학회 논문지, 제24권 2호, pp. 142-148, 1987.

[5] 황신환, 이상욱, "자동 영상 추적기 개발 연구", 대한 전자 공학회 논문지, 제24권 4호, pp. 82-91, 1988.

[6] R. Srinivasan & K.R. Rao, "Motion compensated interframe image prediciton" IEEE Trans. Commun., vol. COM-33, no. 9, pp. 1011-1044, Sept. 1985.

[7] A. Rosenfeld & G.J. Vanderbrug, "Coarse-fine template matching," IEEE Trans. Syst., Man, Cybern., vol. SMC-7, pp. 104-107, Feb. 1977.

[8] J.O. Limb & L.A. Murphy, "Estimating the velocity of moving images in television signal," Comput. Graphics Image Processing, pp. 311-321, 1975.

[9] A.L. Gilbert, M.K. Giles, G.M. Flachs et al., "A Real-time video tracking system," IEEE Trans. PAMI, vol. PAMI-2, pp. 47-56, 1980.

[10] William K. Pratt, "Digital Image Processing," John Willey & Sons, pp. 478-499, 1978.

[11] 김종화, 이만형, "칼만필터링을 사용한 목표물 추적 시스템의 설계," 전기학회 논문지, 제37권 9호, pp. 636-645, 1988. 9.

[12] "PC Vision plus Frams Grabber USER'S MANUAL," Imaging Technology Inc., 1986.