論文2000-37TE-3-4

액티브 블록을 이용한 단일 이동 물체 추적 시스템

(A Tracking System of Moving Object using Active Blocks)

安寅秀*, 崔太燮*, 金光勳*, 林承河**, 司空石鎭**

(In-Soo Ahn, Tae-Sup Choi, Kwang-Hun Kim, Seung-Ha Lim, and Sug-Chin Sakong)

요 약

본 논문에서는 액티브 블록을 사용하여 하나의 움직이는 물체를 효율적으로 검출하고 추적하는 방법에 대해 제안한다. 이것은 특정한 픽셀 값을 대표값으로 추출하여 8*8 픽셀 값들을 대표하는 방식을 사용하고, 인접한 명암차에 의한 차영상 정보를 비교, 분석하여 움직이는 물체를 검출한다. 이미지 데이터 획득 부분은 소프트웨어로 처리함으로써 하드웨어의 비중을 크게 감소시켰으며, 고해상도 이미지를 저해상도 이미지로 변환하여 획득해야 할 데이터 수를 대폭 줄임으로써 실시간으로 이동하는 물체를 추적할 수 있도록 하였다. 따라서 이미지에 대한 모든 픽셀 값 정보나 비디오 인터페이스 카드 제어를 위한 가상 드라이버와 같은 특정 목적의 시스템 드라이버가 없는 간단한 시스템에서도 움직이는 물체를 실시간으로 추적할 수 있도록 충분한 속도를 제공하며 개인용 컴퓨터상에서도 저렴한 비용으로 고정 영역 내의 움직이는 단일 이동 물체 추적 시스템을 구축할 수 있다.

Abstract

In this paper, we propose a way to detect a moving object efficiently and to track it using the active blocks. Instead of all pixels in 8*8 pixel value, any special pixel is extracted and we detect a moving object by comparison and by analysis the difference image information from darkness value of the same area. In the acquisition of image data by software processing, we reduce the number of data which obtain by convert high resolution image to low resolution image, and we can track a moving object in real time. So it can track a moving object in simple system without all the pixel value of the image data or additional VxD(Virtual x Driver). This system can be useful to track of a moving object in fixed block on PC(Personal Computer) and low custom.

Ⅰ. 서 론

영상을 표현하는 방식에는 픽셀 기반 표현 방식인 PWD(Pixel-Wise Description)와 블록 표현 방식인 BWD(Block-Wise Description) 두 가지가 있다[1]~[3].

PWD는 각각의 픽셀들과 주위 픽셀들 사이의 상관관계를 반영하지 못하는데 반해서 BWD는 각 블록들이 테두리 혹은 그 밖의 몇 가지 특징에 대한 정보를 포함하고 있다. 따라서 BWD는 영상 확대와 잡음 제거를 위한 영상 처리 분야에 매우 유용하다. 일반적인 영상 처리 중에서 동화상 처리 방식에는 다음과 같은 방식들이 있다.

첫째, 연속적인 영상들을 분석하여 각 픽셀 또는 영역에 속도 벡터를 할당하고 유사한 속도 벡터를 갖는

* 正會員, 國民大學校 電子工學科
(Dept. of Electronics Engineering, Kookmin University)
** 正會員, 富川大學 電子科
(Dept. of Electronics Engineering, Bucheon College)
接受日字:2000年7月18日, 수정완료일:2000年9月6日

픽셀이나 영역들의 그룹을 감지하여 이동 물체에 대한 동작 정보를 추출하는 방법이 있다[4]~[6]. 이 방법은 약간의 움직임에도 민감하고 정확한 움직임 벡터의 산출에 많은 계산량이 수반되므로 원하는 속도의 개선은 어렵다. 둘째, 인접한 영상간의 명암차에 의해 형성된 차영상을 분석하여 물체의 이동 형태에 대한 정보를 추출할 수 있다. 여기서 적용되는 기법은 명암차를 이용하여 정지 영역과 동작 영역을 분리하는 방법[7]과 차연산 결과와 에지(edge) 연산 결과를 병합하여 물체의 동작 정보를 추출하는 방법[8] 등이 있으나 이는 하드웨어의 발달로 인해 기본적으로 화상에 대한 데이터 획득에 대한 과정을 생략하고 있다고 가정한다. 이것은 전자에서 언급한 벡터 기반 방법보다는 속도의 개선이 다소 쉽다.

본 논문에서는 데이터 획득이라는 근본적인 문제를 하드웨어 구성에 의존하지 않고, 소프트웨어로 해결하기 위해 움직이는 물체 영역을 나타내는 기준 영상이 되는 액티브 블록(Active Block)을 사용한다. 액티브 블록을 사용하여 데이터 획득에 소요되는 시간을 최소화시킴으로써 얻어진 영상 데이터를 효율적으로 이용하여 기존의 동화상을 처리할 때 제안된 영상 처리 기법[7][8]에 액티브 블록만이 갖는 저해상도 이미지 처리 방식, 색상값 설정, 잡음 제거를 통한 액티브 블록의 생성, 움직임이 발생한 블록을 비교 블록으로 삼는 기준을 정하는 문제와 더불어 블록간의 상관도 계산 방식에 있어 속도 향상을 위해 나타날 수 있는 문제점들을 보완하고자 한다. 이에 본 논문에서는 어떤 한 대표 픽셀의 획득으로 발생한 액티브 블록을 가지고 실제 동화상 내의 움직이는 물체의 인식과 추적에 대한 활용성을 다음과 같은 제약 조건하에 다룬다. 첫째, 움직이는 물체의 수는 하나로 한정하고, 둘째, 움직이는 물체는 연속적으로 입력되는 세 개의 영상 내에 존재해야 한다. 셋째, 움직이는 물체의 운동 방향은 처리 과정 중간에는 변화되지 않고, 넷째, 입력되는 영상의 획득에 사용되는 카메라는 고정되어 있다. 이런 가정하에 본 시스템은 고정된 입력 영상 영역 내에서 움직이는 일정 면적을 가진 단일 이동 물체를 검출하고 추적한다.

Ⅱ. 액티브 블록의 생성과 특징

1. 액티브 블록의 생성

입력되는 영상은 320*240 픽셀 크기를 가진 직사각형의 동화상을 기준으로 하였으며 이를 입력 영상으로 하고, 초기 영상의 전체 픽셀 중에서 8*8 픽셀 크기를 단위 블록이라고 하면 단위 블록은 가로 40개, 세로 30개가 생성된다. 이렇게 8*8의 단위 블록이 가로 40개, 세로 30개 생성된 영상 단위 영상이라 하고, 인접한 두 단위 영상의 차연산에 의해서 새로이 생성되는 영상을 단위 차영상이라 한다. 이때, 단위 차영상의 값은 '0' 또는 '1'의 값을 가진다. 이 값들은 연속적으로 입력되는 단위 영상간의 차이를 계산함으로써 그 차이의 크기에 따라 다음의 식 (1)과 같이 정의한다.

$$S(x, y) = \begin{cases} 1 & (\text{만약, } m(x, y) > \text{기준 한계값}) \\ 0 & (\text{나머지}) \end{cases} \quad (1)$$

(S(x, y) : 입력 영상, m(x, y) : 단위 영상의 차)

본 논문에서의 입력 영상은 카메라로부터 직접 입력되는 방식을 취할 수 있도록 설계되었으며 카메라를 이용하여 AVI 파일 형식으로 준비된 동영상을 테스트한다. 미리 제안한 추적 방식의 제한 요소로 인하여 하나의 움직이는 물체의 크기가 단위 블록보다는 크다는 전제하에 본 시스템은 동작한다.

연속적으로 입력되는 세 단위 영상 중에 각각 인접한 단위 영상끼리 차연산을 실시하여 얻은 두 개의 단위 차영상에는 물체의 움직임에 따라 한 개 내지는 두 개의 '1'의 값을 가진 라벨링 블록(labeling block)이 발생한다. 생성된 두 개의 단위 차영상 중에서 하나의 단위 차영상 내에서 라벨링 블록을 포함한 최대 사각형 영역을 다른 단위 차영상의 동일 형태의 영역과 교집합을 구함으로써 액티브 블록이 발생하게 된다. 이때 생성된 액티브 블록은 움직이는 물체의 모습을 보여주는 기준 영상(reference image)이 된다.

2. 저해상도 처리

화상 이미지의 표현에 있어서 고해상도로 이루어진 영상을 저해상도로 표현하면 물체의 윤곽이 거칠어지기는 하지만 원래의 물체가 가진 특성은 보존되어 눈으로도 물체의 특징적인 윤곽의 식별이 가능하다[9].

그림 1의 256*256 해상도를 가진 영상과 32*32 해상도를 가진 영상을 비교해 보면 영상의 정밀도는 떨어지지만 바탕과 물체의 윤곽의 대비로 육안으로의 식별이 가능하다. 본 논문에서는 이 점을 이용하여 초기 영

상을 단위 영상으로 만든 후, 단위 차영상에서 액티브 블록을 생성한다.

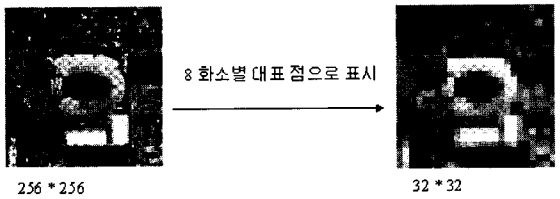
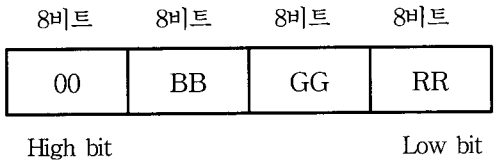


그림 1. 해상도 효과
Fig. 1. Resolution efficiency.

3. 비교 색상값 처리

컴퓨터 비전(computer vision) 분야의 기술적인 발달로 비디오 카드의 성능은 상당히 향상되어 현재는 자연 색상을 거의 완벽에 가깝게 지원하게 되었다.



(BB : 파랑색 값 / GG : 녹색 값 / RR : 빨강색 값)
그림 2. 32비트 픽셀 데이터의 구조
Fig. 2. Structure of 32 bit pixel data.

그림 2와 같이 32비트 배열에서 둘째 부분의 파랑색 값에 따라 전체 32비트 값의 차이가 발생한다. 이 중에서 특정한 부분의 색상만을 얻어서 픽셀 값으로 설정하기에는 많은 문제점이 있다. 그것은 첫째, 설정한 특정 부위의 값이 초기 영상에서 무의미한 색일 수 있기 때문이다. 따라서 색의 통일성을 유지하기 위해 256단계의 계조도로 변환하여 초기 영상을 단위 영상으로 만든다[9].

4. 잡음 제거를 이용한 액티브 블록 생성

단위 블록으로 단위 영상을 만들 때 제안된 시스템에서는 단위 블록이 입력 영상에서 해당 단위 블록 범위 내에 존재하는 64개의 픽셀 데이터 중에서 좌측 상단값을 대표 데이터 값으로 받아들임으로써 잡음이 발생된 픽셀 데이터를 대표 데이터로 간주할 수 있다. 이 때 단위 블록은 더 이상의 대표값을 나타낼 수 없으므로 잡음에 불과하다. 따라서, 단위 차영상에 이런 잡음이 많이 발생하면 단위 차영상 내에서는 제안 시스템에서 요구하는 비교 블록을 선정할 수 없게 된다. 그러나 라벨링 블록의 면적을 고려하면, 잡음의 특성상 시간과 위치에 대해 랜덤(random)하게 발생하는 특성이 있으므로 단위 차영상 내 라벨링 블록의 면적이 상대적으로 작은 것을 잡음으로 간주하여 이 부분의 문제점을 개선할 수 있다. 여기에서의 잡음은 거친 장애물이 실리는 것에 따른 열화로 인한 화상 잡음을 말한다[9]. 화면상에 잡음이 실리는 위치가 불규칙하거나 그 크기가 불규칙한 랜덤 잡음은 불규칙한 간격으로 발생하기 때문에 무한정의 화면이 겹치게 되면 잡음이 없는 화상을 얻을 수 있다.

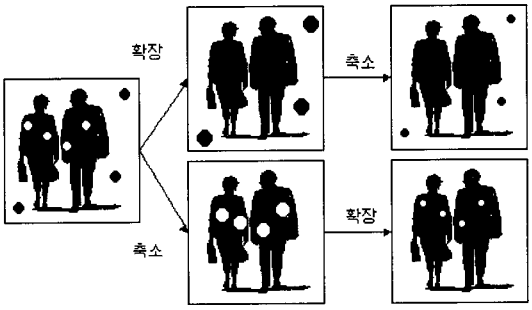


그림 3. 영상 잡음의 제거
Fig. 3. Reduction of image noise.

본 논문에서는 처리 시간을 무한정 할당할 수 없으므로 잡음이 발생한 픽셀 데이터는 주변 픽셀 값과 비교함에 있어 색상에서 명확한 차이가 있다는 성질을 이용해 잡음을 제거한다.

영상 잡음 제거에 있어서 본 논문에서는 입력 영상에서 단위 블록으로 나눈 후에 단위 영상을 가지고 모든 영상 데이터 처리하므로 외부의 잡음에 대해서는 민감하지 않지만 움직이는 물체를 직접적으로 표현하는 물체 내부의 잡음에 대해서는 움직이는 물체를 나타내는 데이터의 수가 모든 데이터를 읽는 경우에 비해 적으므로 그 중에 어느 하나라도 잡음이 발생하여 백색 잡음으로 나타나게 되면 라벨링 연산을 실시할 때, 하나의 물체가 떨어지거나 깨진 모양의 하나 내지는 두 개의 라벨링 블록을 형성한다. 따라서 이런 결과를 가지고 영상 처리를 할 수 없으므로 움직이는 물체를 이루는 데이터들 사이에 잡음을 제거하기 위해 '확장→축소' 의 방법을 사용하였다.

'확장→축소' 의 방법을 그림 4에서와 같이 사용하면 깨어진 블록이 하나로 이어지게 되고, 그림 4의 우측에

나타난 이어진 블록은 액티브 블록이 되기 위한 비교 블록 또는 후보 블록이 될 수 있다. 여기서 비교 블록, 후보 블록이라는 것은 상대적인 크기를 기준으로 구분한 것으로 라벨링 연산 결과 가장 많은 면적(블록 수)을 차지하는 블록을 비교 블록으로 간주하며, 두 개 이상의 블록이 발생하지 않았을 경우에는 위의 블록을 액티브 블록으로 한다. 이에 반해 후보 블록은 운동량이 클 경우에 발생할 수 있는 두 개 이상의 블록 중에서 비교 블록을 제외한 가장 많은 블록 수를 가진 블록을 나타낸다. 이 블록은 운동량이 클 경우에 상관도 검사를 할 수 있는 영역으로 처리한다. 실험적 결과에 의해서 비교 블록 수의 75[%] 정도 이상 되는 블록을 후보 블록으로 간주하며 위의 수치보다 작은 블록에 대해서는 잡음에 의해 발생한 블록으로 간주하여 처리 과정에서 무시한다.

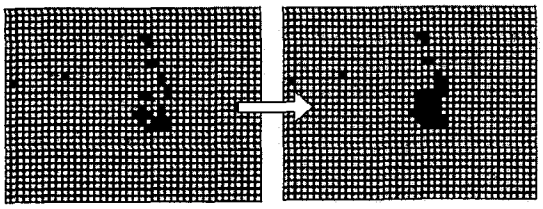


그림 4. 잡음 제거를 이용한 라벨링 블록의 생성
Fig. 4. Creation of labeling block using noise removal.

5. 움직임에 따른 블록 구분 문제

동일 색상의 물체가 운동량이 클 때와 작을 때 각각에 대해 다른 단위 차영상을 보여 주게 된다. 움직임이 작을 때는 두 개의 액티브 블록 사이에는 교집합 영역이 발생하고 이 영역은 변화하지 않은 것으로 판단되어 가운데 홀(hole)이 생성된다. 반면에, 운동량이 큰 경우에는 두 개의 액티브 블록이 발생하게 되고, 이 때는 어느 블록이 본 논문에서 제시한 액티브 블록인지 구별할 수 없다. 따라서 블록간 상관도 문제는 물체의 움직임이 클 경우에 어느 블록이 액티브 블록인지 구분하기 위해서 이전에 생성된 기준 액티브 블록의 위치와 액티브 블록을 이루는 단위 블록의 값들과의 상관 관계식 통해 구분한다.

Ⅲ. 시스템 구현

1. 제안 시스템의 비교 및 적용된 기본 알고리즘

현재 구현된 시스템은 입력 부분이 비디오, CCD 카메라, AVI와 같은 동화상 파일을 통해서 입력되는데 대부분 카메라를 통해서 영상 신호가 입력된다. 카메라를 통해서 입력받은 데이터는 메모리에 저장되고, 저장된 데이터는 V×D를 프로그램에서 제어하여 원하는 데이터를 얻어낼 수 있으며 코드를 보내어 비디오 카드를 제어할 수 있다. 여기서 비디오 카드는 특정한 목적에 사용하는 그레버(grabber)를 통칭하는 말로써, 일반적으로 컴퓨터에 부착되어 사용되는 비디오 카드와는 다른 하드웨어이다. 그러므로, 프로그램에서는 데이터를 읽어오는 속도가 빠르므로 얻어진 데이터를 입력받아 실시간으로 복잡한 이미지 처리 과정을 실시한다.

이와는 달리, 제안 시스템은 영상을 입력받은 부분은 기존 시스템과 유사하다. 단지 차이가 있다면 사용되는 비디오 카드는 특정 목적을 수행하는 하드웨어가 아니라 일반적으로 사용되는 그래픽 카드를 대신할 수 있다. 또한 데이터를 읽어오는 방식이 물리적인 메모리에서 읽어 오는 것이 아니라 화면에 표출되어 있는 픽셀에서 직접 데이터를 읽어온다. 단점으로는 제안 시스템을 통해서는 비디오 카드를 제어할 수 없다는 것이다. 즉, 본 논문에서 구현한 시스템은 폼이나 캔버스와 같은 윈도우 프로그램의 표면에 동화상이 재현되면 데이터를 읽어올 수 있는 시스템이다.

다음은 특정 상황에서 적용된 알고리즘에 대한 내용으로 우선, 움직임을 가진 이동 물체는 연속적으로 들어오는 단위 영상을 가지고 차연산을 실시하여 얻은 단위 차영상 내에 존재하는 '1'의 값을 가진 라벨링 블록을 말한다. 이것은 단위 차영상 내에서 육안으로 확인될 수 있는 형태를 가진 물체를 말하며, 움직임이 없는 상태로 라벨링 블록이 생성되지 않았으면 제안 시스템에서는 이동 물체라고 보지 않는다. 특정 상황에서 적용된 알고리즘은 다음과 같다.

첫째, 추적하고자 하는 물체는 연속적으로 입력된 세 단위 영상 내에 존재하면서 이전 영상 내의 위치에서 다른 위치로 이동하여 시스템이 판별 가능한 움직임을 가져야 한다. 그림 5에서와 같이 세 영상 내에 움직이는 물체가 모두 존재할 경우에는 기준 블록의 인식 과정상 문제가 발생하지 않는다. 이와 비교해서 두 개의 단위 영상에만 움직이는 물체가 존재할 경우도 차영상 2에 라벨링 블록이 생성된다고 전제하면 기준 블록을 인식할 수 있지만 한 영상에만 움직이는 물체가 존재할 경우와 단위 영상 1, 2에서 물체가 이동하지 않으므

로 인해 단위 차영상 2에 라벨링 블록이 생성되지 않을 경우에는 기준 블록의 위치와 데이터 값은 정확한 의미를 가지지 못한다. 이런 경우에 대비하여 본 시스템에서는 프로그램 동작을 일시 중지하고 모든 데이터를 초기화시킨 후에 처음부터 다시 프로그램을 실행시키고 이러한 과정에는 다소의 시간적 지연이 필연적으로 발생하지만 지연 과정을 거친 후 새로운 기준 블록을 가지고 프로그램은 다시 실행된다.

둘째, 초기 프로그램의 수행에 있어서 약간의 지연 과정을 주어 프로그램 초기에 발생할 수 있는 문제를 최소화하였다. 프로그램이 처음 실행될 경우에 액티브 블록은 발생하지 않는다. 프로그램이 수행되기 위해서는 액티브 블록이 발생해야 하므로 초기 프로그램 수행에 있어서는 일정 지연 시간을 둔다. 셋째, 상관 관계 계산에 있어서 유사 정도의 기준을 비교적 작은 값인 50[%]의 수치로 한정함으로써 액티브 블록 특성상 잡음의 형성으로 인한 단위 차영상 내의 움직이는 물체를 표현하고 있는 라벨링 블록의 외형에 다소의 변형이 발생하더라도 상관 관계를 계산하여 동일 물체임을 확인할 수 있게 설계하였다. 넷째, 물체의 움직임은 인접한 두 개의 입력 영상에서는 굴절없이 한 방향으로 이동한다고 가정하여 움직이는 물체의 이동 속도와 방향을 제한하였다. 움직임이 연속적인 두 영상 내에서 방향 전환을 실시한다면 추적 시스템에서는 실제 움직임을 감지할 수 없어 제안 시스템은 올바른 위치 추적을 할 수 없으므로 이런 경우는 고려하지 않았다.

그림 5. 연속적인 세 단위 영상에서 움직이는 물체
Fig. 5. Moving object on continuous three unit images.

2. 프로그램 순서도

아래 그림 6의 흐름도에 따른 제안 시스템을 소프트웨어로 제작한 테스트 프로그램은 다음 그림 7과 같다. 제작 환경은 운영 체제로는 윈도우즈를 사용했으며, 개발 툴(tool)로는 인프라이즈社(구 볼랜드社)의 C++ Builder 3.0 C/S를 사용하였다.

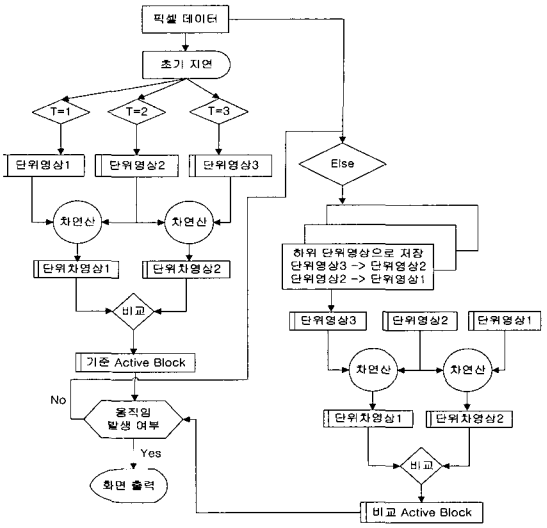


그림 6. 흐름도
Fig. 6. Flow chart.

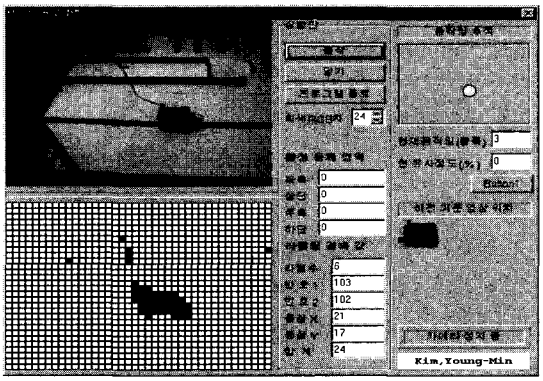


그림 7. 프로그램
Fig. 7. Program.

그림 7에서와 같이 구성은 좌측 상단에 입력 영상을 나타내는 입력 부분이 있다. 이 부분은 입력 장치로부터 받은 데이터를 폼 위에 나타낸다. 좌측 하단에는 입력 부분의 입력 영상을 단위 블록 단위로 만들어 단위 영상을 나타내며 라벨링 연산까지 거친 단위 차영상이 나타나게 된다. 상황판 안에는 명령 버튼들과 프로그램 수행 결과값을 나타내는데 이 수치들은 단지 프로그램의 수행 정도를 분석하는 수치로 각 콤포넌트(component) 사이에 동기화 되지 않는 경우도 있다. 위에서 회색 RGB차 라고 표시된 콤보 박스(combo box) 부분은 라벨링 블록의 수를 3~5까지 제한하기 위하여 자동적으로 한계값을 조정한다. 한계값을 위와 같은 수치로 정한 이유는 움직이는 물체를 본 제안 시스템에서는 한

개로 한정하였으므로 라벨링 블록이 두 개 이상은 원칙적으론 발생할 수 없다. 하지만 잡음의 영향과 그림 7과 같이 이동 물체와 연결된 선이 있을 경우에는 두 개 이상의 라벨링 블록이 발생할 수 있다. 이 점을 고려하여 라벨링 블록이 일정 수의 범위를 벗어나면 한계값을 자동 조정하여 입력 영상의 품질에 상관없이 일정한 라벨링 수를 유지할 수 있도록 하였다.

액티브 블록은 액티브 블록의 위치를 표시하며 아직 생성되지 않았을 경우나 적당치 않은 블록을 검색했을 경우에는 '0'값을 표시한다. 라벨링 결과값 부분에는 라벨 수에 현재 단위 차영상 내에 존재하는 라벨링 블록의 총 수를 표시하며 번호 1은 비교 블록을 번호 2는 후보 블록을 표시한다. 면적이 비교 블록의 75[%]가 안되면 후보 블록이라고 보지 않고 잡음에 의해 생성된 잡음 블록으로 보고 무시한다. 중심 X, Y는 액티브 블록의 단위 차영상 내의 무게 중심을 표시하며 합계는 비교 블록 내의 '1'의 값을 가진 단위의 블록 총 개수를 표시한다.

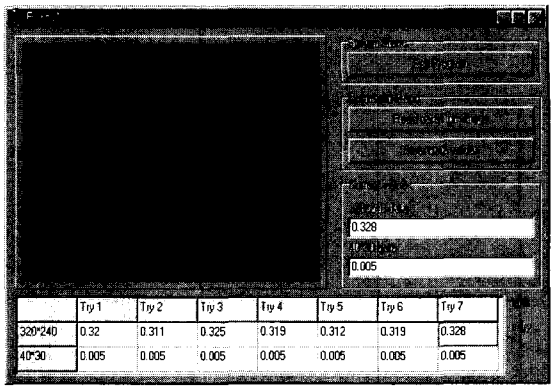
움직임 추적 부분은 동그라미를 표시해서 현재 액티브 블록의 위치를 표시하며, 현재 움직임(블록) 부분은 거리를 블록 단위로 계산하여 이전 위치에 대해서 얼마만큼의 이동이 있었는지를 표시한다. 현 유사 정도는 상관 계산을 실행하게 되면 그 정도를 백분율 [%]단위로 표시하게 된다. 이전 기준 영상 위치 부분에는 현재 액티브 블록의 위치 정보를 추출해 입력 영상에서 액티브 블록 범위 만큼의 영상을 복사하여 표시하게 되므로 현재 시스템이 올바른 액티브 블록을 가지고 있는지를 알 수 있다.

Ⅳ. 결과 및 고찰

1. 실행 시간의 비교

데이터 획득 시간에서 320*240개의 데이터와 8*8 단위 블록 가로 40개, 세로 30개의 데이터를 읽는 시간을 비교함으로써 실제 본 시스템의 수행 능력을 실험하였다. 영상 데이터를 처리하는데 소요되는 시간이 동일하다고 보면, 실제 시스템의 계산 시간에 큰 영향을 미치는 것은 입력 영상에서 영상 데이터를 읽어오는 시간이다.

그림 8에서와 같이 비교 대상은 320*240의 데이터 획득에 걸리는 시간과 단위 영상을 기준으로 한 데이터 획득 시간을 비교한 것으로 위의 그림 8의 하단에 7번의 시도에 따른 값의 차이를 보여 주고 있다.



	Try 1	Try 2	Try 3	Try 4	Try 5	Try 6	Try 7
320*240	0.32	0.311	0.325	0.319	0.312	0.319	0.328
40*30	0.005	0.005	0.005	0.005	0.005	0.005	0.005

그림 8. 처리 시간 (1)의 비교
Fig. 8. Comparison of processing time(1).

320*240 데이터 획득을 경우 1이라 하고, 단위 영상의 데이터 획득을 경우 2라 할 때, 경우 1에서는 초당 3프레임을 읽어 올 수 있는데 반해 경우 2에서는 초당 200 프레임을 읽어 올 수 있다. 경우 2는 본 논문에서 제안하는 방식으로도 일반적인 동화상 프레임을 처리할 수 있다는 가능성을 보여 준다.

그림 8은 C++ Builder의 Canvas Pixels[x][y]를 가지고 테스트한 결과값이다. 이에 반해 SDK에서 제공되는 Get Pixel(x,y)를 사용하여 Visual C++ 5.0을 사용한 테스트 프로그램의 데이터 획득 시간은 그림 9와 같은 결과를 나타낸다.

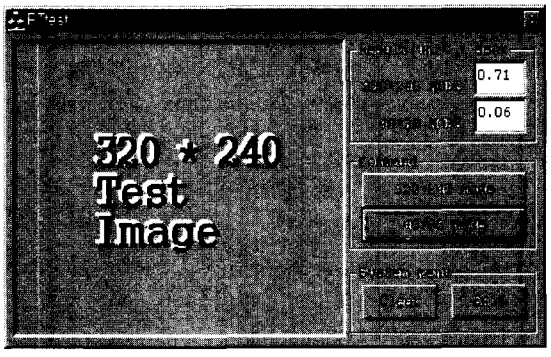


그림 9. 처리 시간 (2)의 비교
Fig. 9. Comparison of processing time(2).

그림 9에서와 같이 영상 데이터를 화면에서 읽는 방식을 바꿈으로 인해 속도의 차이는 320*240 데이터 획득의 경우엔 그림 8과 비교하여 2배 정도, 단위 영상의 데이터 획득의 경우에는 약 10배 정도로 많은 차이를 발생하는 것을 그림 9의 우측 상단값을 보면 알 수 있다.

여기서 실제 프로그램이 수행된 후의 실행 시간을 다시 비교해 보면 다음 그림 10과 같다.

그림 10의 테스트는 SDK의 Get Pixel(x,y)을 사용했을 경우의 실제 실행 결과이다.

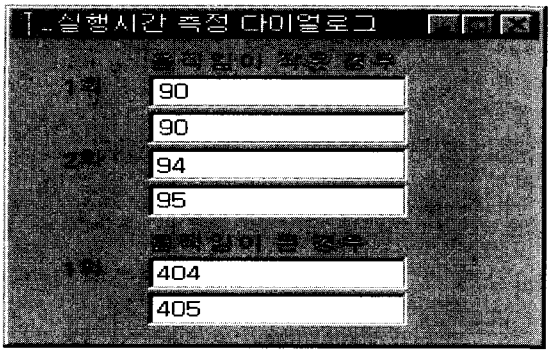


그림 10. 처리 시간
Fig. 10. Processing time.

그림 10에서와 같이 데이터를 읽어온 후, 영상 처리 과정을 모두 거친 후의 실행 시간이 표시되어 있다. 여기서 표시된 결과값은 임의의 시간에 검사하여 움직임이 작을 때와 클 때로 나누어 그 당시 프레임 번호를 표시한 것이다. 분석해 보면, 움직임이 작을 때는 영상 처리가 시작하고 끝나는 시간이 한 프레임이 끝나기 전에 이루어지므로 동화상 재현 속도와 시스템 처리 속도는 일치하게 된다. 움직임이 클 때는 영상 처리가 시작하고 끝나는 시간이 한 프레임에 걸쳐 있으므로 동화상의 재현 속도와 시스템의 처리 속도가 일치할 경우도 있고 더 느린 경우도 발생한다는 결과를 보여주고 있다. 그림 10의 테스트를 Canvas Pixels[x][y]를 사용하여 얻은 결과는 그림 10의 데이터 값과는 달리 대부분 같은 값들만이 나타난다. 이것은 움직임이 클 때나 작을 때에도 동화상의 재현 속도보다 시스템의 처리 속도는 빠르다는 것을 보여 준다.

2. 오차의 발생

단위 블록내의 대표 픽셀에 에지(edge)의 존재 여부에 따라 움직임에 최대 10픽셀의 떨림이 발생한다.

그림 11은 단위 차영상 내에 나타나는 라벨링 블록의 테두리 부위의 값들이 유사한 형태임에도 불구하고 3가지 모습으로 보일 수 있다는 것을 나타낸다.

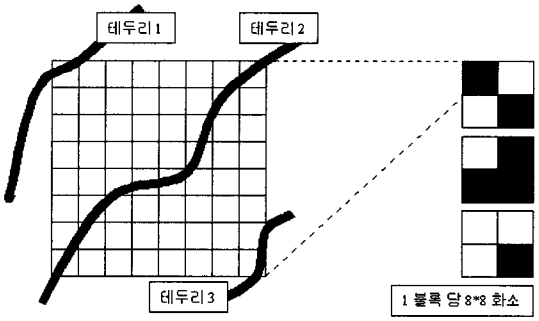


그림 11. 키 값 에러
Fig. 11. Key value error.

각각의 테두리는 단위 영상을 만들 때 이동 물체의 테두리만을 나타낸 그림으로 각 테두리를 기준으로 우측 하단 부는 물체가 있는 것으로 본다. 테두리1 과 같은 상황에서는 대표 데이터 픽셀은 배경이 아닌 값을 가지므로 단위 블록은 테두리가 존재한다고 판단하여 그림 11의 우측 상단의 첫 번째 그림과 같은 블록 형태를 가지게 된다. 테두리 2의 경우는 인접 블록은 이동 물체가 있는 것으로 인식하지만 단위 블록 자신은 테두리가 없는 것으로 판단하며 블록의 형태는 위 그림의 우측 중간 블록과 같다. 테두리 3과 같은 경우에는 주변 블록은 물론 단위 블록 자신도 이동 물체가 존재하지 않는 것으로 인식하게 된다. 이런 것이 발행할 수 있는 경우는 고정된 카메라도 다소의 떨림이 발생하여 움직이지 않는 물체라도 같은 물체이지만 전체적으로 이동된 것과 같은 형태를 가지게 되므로 최대 10픽셀의 차이를 발생시킬 수 있다.

V. 추적 실험

그림 12는 실제 프로그램 작동 예이다.

그림 13에서와 같이 총 6회의 테스트 점에 대해서 실제 픽셀의 위치와 본 논문에서 구현한 시스템에서 추적한 위치의 차이를 위 그림의 좌측 하단의 결과로 보여 주고 있다. 빈 사각형은 실제로 이동 물체가 움직인 궤적이며, 색이 채워진 작은 사각형은 추적 시스템의 이동 물체 추적 위치를 나타내고 있다. 결과를 보면 최대 13픽셀의 차이가 나며 다소 3픽셀의 차이가 발생

했다.

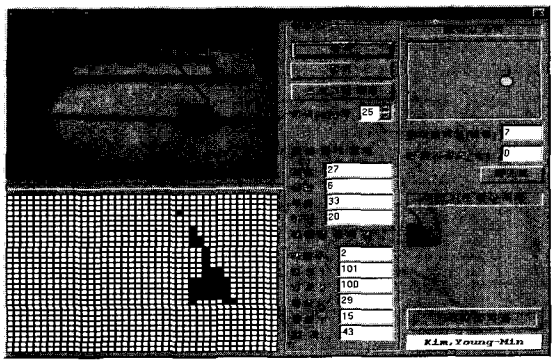


그림 12. 실제 프로그램 작동의 예
Fig. 12. Example real program operation.

이론적으로 최대 10픽셀보다 실제 최대 오차가 3 픽셀 정도 크게 나온 것은 테스트 과정에 있어 사용자 조작에 의한 오차라고 가정할 때, 그림 11에서 언급한 오차에 매우 근접한 오차를 발생시켰으며 실제로 그림 13에서와 같이 본 추적 시스템은 이동 물체의 위치에 근접하게 추적하고 있다.

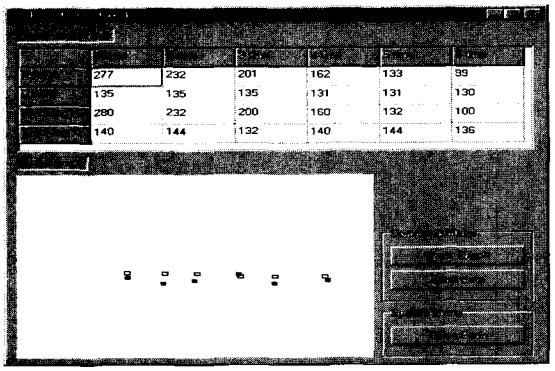


그림 13. 실제 위치와 추적 위치
Fig. 13. Real locus and tracking locus.

302*240개의 데이터를 읽어오는데는 초당 3번까지 가능한데 이 속도로는 실시간으로 추적하기 위한 알고리즘을 적용시킬 수 없다. 단위 영상의 데이터를 읽어오는데는 초당 200번까지 가능하므로 실시간 추적에 필요한 충분한 속도를 제공하고 있다. 실제 위치와 추적 위치의 오차 결과값을 보면 평균 4픽셀의 차이를 보인다. 이 점은 테스트 당시에 눈으로 확인 후에 마우스를 가지고 위치를 선택함으로써 발생하는 오차, 그림 11에서 설명된 이유로 발생할 수 있는 오차와 운동량에 따른 추적시스템의 인식 위치와의 오차 영향으로 발생한 것으로 볼 수 있다. 이론적인 계산에 의하면 운동량이 큰 경우에도 22프레임으로 입력되는 영상까지 처리할 수 있으나 실제 테스트에 사용한 영상이 초당 15프레임을 기준으로 하기 때문에 제안 시스템은 최고 15프레임의 처리 속도를 보인다.

표 1. 결과값의 비교
Table 1. Comparisons of result value.

내 용	단 위	평 균	비 고
320*240	sec	0.32	초당 3프레임
단위 영상	sec	0.005	초당 200프레임
실제 위치	X position/Y position	184/133	추적 위치와 평균 4픽셀 차이 발생
추적 위치	X position/Y position	184/139	초당 45프레임 처리 가능
큰 움직임	시작 프레임/끝 프레임	88.2~88.5	초당 22프레임 처리 가능
작은 움직임	시작 프레임/끝 프레임	337.3~338	
현재 속도가 15[frame/sec]인 테스트 AVI 파일 처리 가능			

VI. 결 론

윈도우 95이상의 환경에서 특정 목적으로 사용되는 비디오 카드를 제어하기 위해서는 VxD를 사용해야 하는데 이것만으로는 추적 시스템 구축에 소프트웨어적으로 많은 비용이 든다. 근래 소개되는 Windows NT 5.0이나 Windows 98은 WDM(Win32 Driver Model)이라는 새로운 개념을 이용하여 시스템 드라이버의 제작이 전보다는 훨씬 용이해졌다고는 하지만 본 논문에서는 WDM과 같은 새로운 개념을 적용시키지 않고 본 시스템을 구성하였으며, VxD 제작에 따른 비용 문제에 대해 액티브 블록을 도입하여 소프트웨어적으로 간단한 시스템에서도 구현 가능한 추적 알고리즘에 대해 제안하였다.

제안 시스템은 첫째, 움직이는 물체의 수를 한 개로 한정하였으며, 둘째, 움직임의 속도도 시스템에서 처리하는 과정에서 방향의 굴절이 없다고 가정하였다. 셋째, 입력 장치로 카메라를 사용할 경우에 카메라는 고정되어 있다고 전제하였으며, 넷째, 이동 물체는 연속으로

입력된 세 단위 영상 내에서 움직임을 가져야 한다는 실행 조건하에서 올바른 실행을 할 수 있었다. 본 논문에서 제시한 내용은 실행에 있어서 위와 같이 많은 부분에서 제한을 가지고 있으나 일반적으로 저렴하게 개인용 컴퓨터를 이용한 단일 이동 물체의 움직임만을 관측하는 감시 시스템의 구성시 유용하게 활용될 수 있다. 향후 입력 영상을 단위 영상으로 만드는 과정에서 대표 픽셀 설정시, 발생할 수 있는 실제 움직이는 물체의 위치와 본 시스템에서 적용한 액티브 블록의 위치 오차를 최소로 줄이는 방법에 대해서 고찰할 필요성이 있다. 이 문제는 단위 블록 내의 총 픽셀 수가 적을수록 오차의 범위도 줄지만 상대적으로 속도의 저하를 초래하게 된다는 점을 고려하여 개선하면 최소 비용으로 최적의 추적 시스템을 구성할 수 있을 것으로 기대된다.

참 고 문 헌

[1] D.Chen and A.C.Bovik, "Visual pattern image coding," IEEE Trans. Commun. Vol. COM-38, pp.2134-2146, Dec. 1990.

[2] N.M.Nasrabadi and R.A.King, Image coding using vector quantization: A review, IEEE Trans. Commun. Vol. COM-36, pp.857-871, Aug. 1988.

[3] N.C.Griswold, D.R.Halverson, and G.L.Wise, A note on adaptive block truncation coding algorithm for image compression, IEEE Trans. Acoust. Speech Signal Processing, Vol. ASSP-35, pp.1201-1203, Aug. 1987.

[4] William B. Thompson, Pamela Lechleider, and Elizabeth R.Stuck, "Detecting moving objects using the rigidity constraint," IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI-15, No. 2, pp.162- 166, 1993.

[5] W.K.Chow and J.K.Aggarwal, Computer analysis of planar curvilinear moving images, IEEE Trans. Computer, C-26, pp.179-185, 1977.

[6] His Jian, Lung Fa Huang, and Z. Chen, Multi-frame ship detection and tracking in a infrared image sequence, Pattern Recognition, Vol. 23, No. 7, pp.785-798, 1990.

[7] R. Jain, D. Milter, and H. H. Nagel, Separating non-stationary from stationary scene components in a sequence of real world TV-images, Proc. 5th Int, Joint Conf. Artificial Intelligence, pp.612-618, 1977.

[8] Ramesh Jain, "Extraction of motion information from peripheral processes," IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI-3, No. 5, pp.489-5032, Sep. 1981.

[9] NHK 방송기술연구소 화상연구부, C언어에 의한 화상 처리 실무, 국제 테크노 정보연구소, pp.101-124, pp.149-168, 1994

저 자 소 개

金 光 勳(正會員)

1966년 12월 생. 1989년 2월 국민대학교 전자공학과 졸업(공학사). 1991년 2월 국민대학교 전자공학과 졸업(공학 석사). 1991년 1월~1996년 9월 (주)나우정밀 전임연구원. 1996년 10월~1997년 9월 (주)해태전자 전임연구원. 1997년 10월~1998년 7월 (주)건인텔레콤 선임연구원. 1998년 8월~1999년 9월 (주)넥스컴 선임연구원. 1999년 10월~현재 (주)퓨처텔 책임연구원. 1997년 9월~현재 국민대학교 전자공학과 박사 과정

安 寅 秀(正會員) 第 37卷 SD編 第 6號 參照

崔 太 燮(正會員) 第 37卷 SD編 第 6號 參照

林 承 河(正會員) 第 36卷 T編 第 3號 參照
현재 부천대학 전자과 교수

司空石鎭(正會員) 第 37卷 SD編 第 6號 參照
현재 국민대학교 전자공학부 교수