2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

교통 정보 시스템을 위한 차량 검지기 설계

강경훈*, 정성태*, 이상설*, 금기정**, 남궁문***
*원광대학교 전기전자 및 정보공학부
**명지대학교 SOC 공학부
***원광대학교 건축·도시·토목환경공학부

Design of a Vehicle Detector for Transport Information System

Kyung-Hoon Kang*, Sung-Tae Jung*, Sang-Seol Lee*, Ki-Jung Kum**, Kung-Moon Nam***
*Division of Electric Electronic and Information , Wonkwang University
**Division of Civil and Transportation Engineering, MyongJi University
***Division of Architecture, Urban, Civil & Environmental Engineering Wonkwang University

요약

본 논문에서는 교통 정보 제공 시스템에서 기본적으로 필요로 하는 신뢰성 있는 교통데이터 획득을 위한 교통 영상검지기를 ASIC을 이용하여 효과적으로 구현할 수 있는 구조를 제안한다. 본 논문의 교통 영상검지기에서는 먼저 저가의 CMOS 이미지 센서를 이용하여 영상을 획득한다. 그 다음에 영상을 여러 개의 블록으로 분할하고 블록 매칭 기법을 이용하여 각 블록의 모션 벡터, 즉 각 블록이 다음 프레임에서 어느 방향으로 얼마만큼의 거리를 이동했는지를 추적한다. 그 다음에는 블록들의 모션 벡터로부터 자동차의 속도와 크기를 추출한다. 본 논문의 교통 검지기는 실시간으로 시내 도로나 고속도로에서 실시간으로 교통 정보를 검지할 수 있을 뿐만 아니라 보정이 필요 없어 설치가 매우 간편하다.

1. 서론 [1)]

자동차화의 급격한 진전과 이용률의 증가에 따라 자동차 이용의 효율성에 관한 사회적 관심은 매우 급격히 증가하고 있다. 특히, 도시 내 도로의 용량 부족이나 자동차 교통량 증가에 적절히 대응할 수 있는 도로정비 등이 원활히 추진되지 못하고 있는 상황을 고려할 때 도로교통 문제를 개선할 수 있는 방안의 하나로 저비용으로 교통정보를 통한 도로교통 기능의 효율성 도모는 중요한 기능을 담당할 것으로 크게 주목되고 있다.

이러한 배경에서 교통분야에서는 최근에 ITS(Intelligent Transport System)와 같이 교통분야의 전반에 걸쳐 지능화된 교통체계 구축을 추진하고 있다. ITS 시스템을 구축하기 위해서는 운전자에게 자동차 주행 중 필요한 교통정체 정보, 우회정보 등과 같이 유익한 정보를 제공할 수 있는 시스템이 필수적이다. 교통정보 제공기술에 있어 중요한 부분적 영역으로는 신뢰성과 실시간 성이 확보된 교통정보의 획득문제, 교통정보 이용자의 제공 정보에 대한 인지반응과 모형화 문제, 운전자의 Human Factor에 의거한 교통정보 제공시설 설계, 운영 및 관리문제 등을 들 수 있다.

본 논문에서는 교통 정보 제공 시스템에서 기본적으로 필요로 하는 신뢰성 있는 교통데이터 획득을 위한 교통 검지기를 설계한다. 교통 검지에 가장 많이 사용되는 것이 Loop 검지기이다. 이 Loop 검지기는 매설의 용이성과 검지가 단순하다는 장점을 가지고 있어 많이 활용되고 있지만 도로 파손에 따른 손실, 대형차량의 통행에 의한 수명단축, 검지의 부적확성 등과 같은 단점을 가지고 있다. 이러한 문제점을 극복하기 위한 대안 중의 하나가 카메라를 이용한 영상검지 기술이다. 본 논문에서는 정확한 보정이 필요 없으며 실시간으로 자동차의 속도와 크기를 측정할 수 있는 교통 검지기를 설계한다. 본 논문의 교통 검지기에서는 먼저 저가의 CMOS 이미지 센서를 이용하여 영상을 획득한다. 그 다음에 영상을 여러개의 블록으로 분할하고 블록 매칭 기법을 이용하여 각 블록의 모션 벡터, 즉 각 블록이 다음 프레임에서 어느 방향으로 얼마만큼의 거리를 이동했는지를 추적한다. 그 다음에는 블록들의 모션 벡

본 연구는 한국과학재단 목적기초연구(과제번호: 2000-2-31300-001-3) 지원으로 수행되었음

터로부터 자동차의 속도와 크기를 추출한다. 본 논문에서는 각 블록의 모션 벡터를 계산하는 시스템을 중점적으로 설명한다.

2. 블록 매칭

본 논문에서는 차량의 이동을 검지하기 위해 블록 매칭 기법을 사용한다. 블록 매칭 방법은 동영상 압축에 널리 사용되는 모션 계측 방법이다. 블록 매칭은 블록의 픽셀들이 프레임간에 서로 같은 방향으로 움직인다는 특성을 이용한다. 현재 프레임에 일정한 크기의 블록을 지정하고 다음 프레임(참조 프레임) 안에서 현재 프레임에서의 블록과 가장 일치하는 블록의 좌표를 찾음으로써 두 프레임 사이에서 블록의 모션벡터를 구할 수 있다. 일치하는 블록을 찾기 위해서는 (식1)과 같이 정의되는 블록간의 픽셀에 대한 불일치도를 이용한다. 여기에서 L'(i,j)는 현재 프레임 블록 안의 픽셀들에 대한 luminance값을 나타내고, L''(i,j)는 다음 프레임 블록 안의 픽셀들에 대한 luminance값을 나타낸다. M, N은 블록의 크기를 나타낸다.

$$D(u,v)=\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}|L''(i+u,j+v)-L'(i,j)| \quad (식\ 1)$$

현재 프레임의 블록과 일치하는 블록을 다음 프레임에서 찾는 방법 중에서 가장 기본적인 방법은 full-search 방법이다. 모션 벡터의 최대 변위가 x축 방향으로 $\pm d_x$이고 y축 방향으로 $\pm d_y$라 할 때에 full-search 방법에서는 모션벡터가 존재할 수 있는 모든 탐색공간 $(-d_x \le u \le d_x, -d_y \le v \le d_y)$에서 $D(u,v)$을 계산하여 그 중에서 $D(u,v)$값이 최소가 되는 위치의 블록을 선택한다. 이 방법은 탐색 영역의 모든 위치에서 두 블록 사이의 불일치도 $D(u,v)$값을 계산하기 때문에 정확도는 높지만 $(2d_x+1)\times(2d_y+1)$번의 블록 비교를 필요로 하기 때문에 처리 시간이 오래 걸려서 실시간으로 동작할 수 있도록 구현하기에는 문제가 있다.

탐색 공간의 일부에서만 $D(u,v)$를 계산하면서도 탐색 결과의 정확도는 많이 떨어지지 않는 여러 가지 방법들[1,2]이 제안되었는데 그 중의 하나가 3-step 알고리즘[1]이다. 본 논문에서는 기존의 3-step 알고리즘을 수정하여 사용한다. 그림 1은 $d_x=5, d_y=10$인 경우의 탐색 공간을 나타내고 있다. 사각형, 원, 삼각형은 각각 첫 번째, 두 번째, 세 번째 단계에서 $D(u,v)$를 계산하는 위치를 나타낸다. 사각형 점들은 x,y 방향으로 각각 3 픽셀 간격으로 설정되었다. 이들 중에서 $D(u,v)$ 값이 최소인 점이 선택되고 그 점을 중심으로 x,y 방향으로 거리가 2인 점들에서 $D(u,v)$ 값을 계산한다. 그 중에서 다시 $D(u,v)$ 값이 최소인 점이 선택되고 이번에는 거리가 1인 점들에서 $D(u,v)$ 값을 계산한다. 그림 1에서는 첫 번째 단계에서는 $D(-6,3)$ 값이 최소인 경우에 그 다음 비교 위치가 원으로 표시된 상태를 나타낸다. 그리고 원으로 표시된 점들 중에서는 $D(-8,3)$ 값이 최소인 경우에 다음의 비교 위치가 삼각형으로 표시된 상태를 나타낸다.

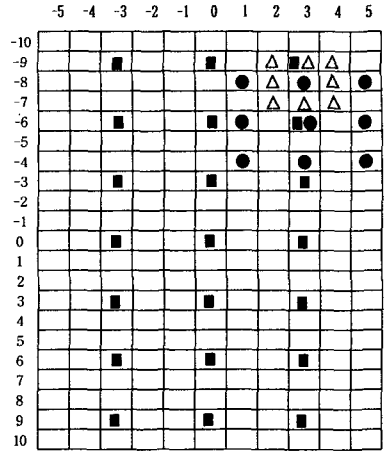

그림 1 수정된 3-step 기법의 탐색 위치 예

3. 블록 매칭 회로 설계

차량 검지를 위해서는 그림 2에 나타나 있는 바와 같이 카메라 이미지의 중앙 부분에 다수의 비교 블록들을 설정하고 각각의 블록에 대하여 블록 매칭을 수행한 다음 각 블록의 모션 벡터를 종합하는 방법을 사용할 수 있다.

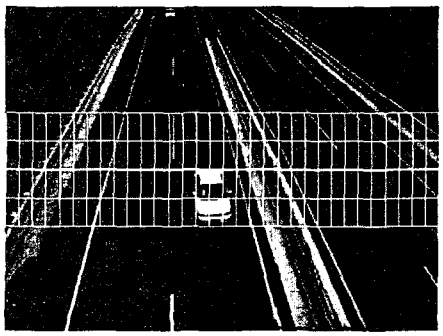

그림 2 차량검지를 위해 매칭되는 블록들

그림 2와 같이 많은 블록들에 대한 블록 매칭을 하나의 처리기에 의해 순차적으로 수행할 수 있는지 또는 여러 개의 처리기들에 의해 병렬 처리해야 하는지를 먼저 검토해야 한다. 본 논문의 차량검지기에서는 초당 60 프레임의 이미지를 출력하는 CMOS 이미지 센서를 사용한다. 따라서 한 프레임 당 $\frac{1}{60}=16666.6u$초의 처리 시간이 주어지는데 이중 $600u$초는 다른 동작을 위한 여유 시간으로 할당하면 한 프레임 당 블록 매칭에 사용될 수 있는 시간은 $T_{frame}=16000u$초가 된다. 블록 매칭에 소요되

는 대부분의 시간은 메모리로부터 픽셀 값을 읽어들이는데 사용되므로 한번의 블록 매칭에 소요되는 시간은 다음과 같이 T_{PE}로 계산될 수 있다. 여기에서 d는 수정된 3-step에서 첫 번째 단계의 비교 회수로서 그림 1에서 사각형의 수와 같다. m과n은 블록의 크기로서 11과 21을 사용하였고 c는 한 픽셀을 읽어서 처리하는데 소요되는 시간으로 $40n$초로 정하였다. 두 번째 항의 8은 3-step의 두 번째와 세 번째 단계에서의 최대 비교 회수 8을 나타낸다.

$$T_{PE} = d\times m\times n\times c + 2\times 8\times m\times n\times c$$
$$= 21\times 11\times 21\times 40 + 2\times 8\times 11\times 21\times 40$$
$$= 342u\text{초}$$

그런데, 16000/342 = 47 이므로 한 처리기로는 47개의 블록에 대한 비교만 수행할 수 있다. 따라서 그림 2와 같은 구성에 대해서는 여러 개의 처리기에 의한 병렬 처리가 필요하다. 여러 개의 처리기를 사용할 때에 고려해야 할 한가지 사항은 이웃한 블록들에 대한 블록 매칭을 수행할 때에 참조 프레임 이미지에서 접근되는 영역이 서로 중복된다는 것이다. 즉, 현재 프레임의 블록이 그림 3의 안쪽 사각형과 같이 $m\times n$이고 모션 벡터의 최대 변위가 x축 방향으로 $\pm d_x$이고 y축 방향으로 $\pm d_y$라 할 때에 참조 프레임에서 접근될 수 있는 영역은 그림 3의 바깥 사각형과 같이 $(2d_x+m)\times(2d_y+n)$이다. 본 논문에서는 $m=2dx+1, n=2dy+1$이 되도록 m,n의 값을 설정하였다. 따라서 이웃 블록 사이에는 항상 참조 프레임 이미지의 접근에 있어서 중복되는 영역이 발생하게 된다. 또한 블록 매칭을 위한 메모리 접근은 물론 이미지 센서를 통한 영상저장, 처리 결과에 대한 접근 등 추가의 메모리 접근이 필요하게 된다. 이를 해결하기 위해서는 멀티포트 메모리와 같은 값비싼 메모리 소자를 사용해야 하는 문제점을 지니고 있다.

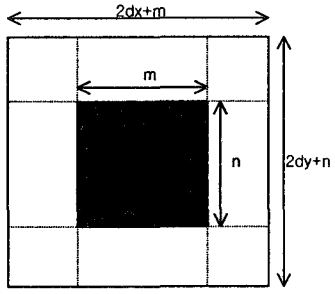

그림 3 참조 프레임에서의 접근 영역

따라서, 저렴한 단일포트 메모리를 이용한 효율적인 처리를 위해서는 두 가지를 문제를 고려해야 한다. 하나는, 참조 프레임 이미지를 어떤 구조로 메모리에 저장해야 하는 가이다. 만약에 모두 한 메모리 모듈에 저장하면 처리기들이 메모리 접근 시에 항상 충돌이 일어나기 때문에 여러 개의 처리기를 사용한 효과를 전혀 거두지 못한다. 또 하나의 문제는 블록들에 처리기를 어떻게 분배하느냐 이다. 예를 들어 한 줄에 걸쳐있는 블록들을 같은 처리기에 할당하면 그 줄 내의 블록에 대한 처리 시에는 메모리 충돌이 일어나지 않게 된다. 본 논문에서는 그림 4와 같은 형태로 참조 프레임 이미지를 분할하여 4개의 메모리 모듈에 분산 저장한다. 여기에서, 작은 사각형들은 블록을 나타내고 가로로 긴 사각형은 메모리 모듈을 나타낸다. 각 메모리 모듈에는 이웃한 두 줄의 블록이 반절씩 저장되는 것에 주의할 필요가 있다. 그림에서 화살표의 중심이 되는 블록은 처리기가 매칭 시키고자 하는 원시 블록을 나타내고 화살표는 처리기에서 그 블록에 대한 블록 매칭 시 접근하는 참조 프레임의 영역을 가리킨다. 여기에서 각 처리기는 이웃한 두 개의 메모리 모듈을 접근함을 알 수 있다. 만약에 한 줄의 블록 전체를 한 메모리 모듈에 저장한다면 처리기가 접근해야할 메모리 모듈이 3 개가 되어 처리 시간이 길어지게 된다.

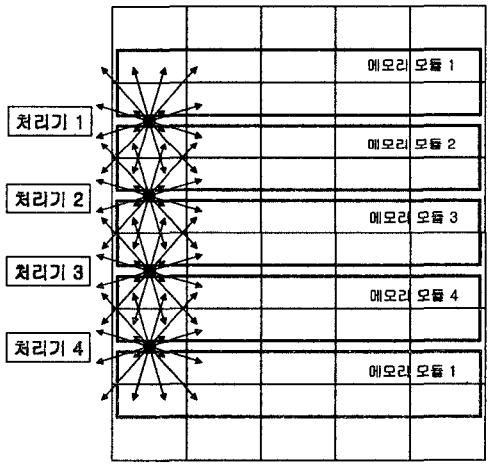

그림 4 메모리 모듈 구성

따라서, 한번의 블록 매칭에 소요되는 시간은 다음과 같이 계산된다. 여기에서 주의할 것은 수정된 3-step의 첫 번째 단계에서는 각 처리기가 동일한 순서로 비교 위치를 설정하면 처리기들은 항상 서로 다른 메모리 모듈을 접근하게 된다는 것이다. 따라서 이때에는 메모리 인터리빙이 필요 없다. 두 번째와 세 번째 단계에서는 이웃한 두 메모리 모듈간에 인터리빙이 필요하고 다음 식에서 $i=2$가 이 때문에 추가되었다.

$$T_{PE} = d\times m\times n\times c + i\times 2\times 8\times m\times n\times c$$
$$= 21\times 11\times 21\times 40 + 2\times 2\times 8\times 11\times 21\times 40$$
$$= 490u\text{초}$$

한 처리기가 수행할 수 있는 블록 매칭의 수는 16000/490 = 33이 된다. 본 논문에서 사용하는 CMOS 이미지 센서는 해상도가 352×288이고 $32\times 11 = 352$이므로 한 줄에 32개의 블록을 설정할

수 있고 한 처리기가 한 줄의 블록들을 담당할 수 있게 된다.

또 한가지 고려해야 할 사항은 프레임 이미지를 저장할 메모리를 처리기만 접근하는 것이 아니라 이미지 센서가 이미지를 저장하기 위해서도 사용한다는 것이다. 이미지 센서와 처리기가 동시에 같은 메모리 모듈을 사용하면 또 다시 메모리 인터리빙이 필요하게 되므로 본 논문의 시스템에서는 더블 버퍼링 기법을 사용하여 이 문제를 해결한다. 즉, 메모리 모듈을 4개가 아니라 8개 사용하여 이미지 센서가 메모리 모듈 4개에 이미지를 저장하는 동안에 처리기는 또 다른 4개의 메모리 모듈에서 이미지를 읽어들인다. 본 논문의 시스템에서는 그림 5와 같이 128KB 크기의 메모리 8개를 사용하여 메모리 모듈을 구성한다. 한 프레임 이미지의 저장에는 $352\times288 = 99\,KB$의 메모리가 필요하므로 8개의 메모리 블록에는 8개의 프레임 이미지를 저장할 수 있지만 그림 5와 같이 기본적으로 4개의 프레임 이미지만을 사용하면 된다.

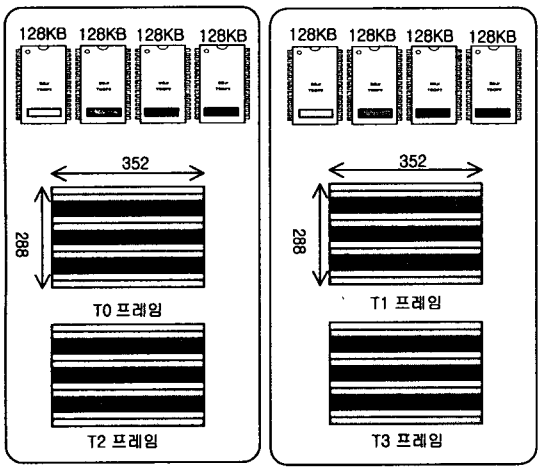

그림 5 메모리 구성

이미지 센서가 T0 프레임과 T1 프레임을 저장하면 처리기는 T0 프레임을 현재 프레임으로 하고 T1 프레임을 참조 프레임으로 하여 블록 매칭을 수행한다. 이와 병렬로 이미지 센서는 T2 프레임을 저장한다. 현재 프레임에서 한 블록을 읽어들이는데 소요되는 시간은 $11\times21\times40 = 9.24u$초이고 4개의 처리기가 병렬로 서로 다른 메모리 모듈에서 동시에 수행된다. 그리고 이미지 센서가 한 프레임을 메모리에 저장하는 시간은 $352\times288\times40 = 4055u$초 인 반면에 한 프레임의 처리에 사용될 수 있는 시간은 $16000u$초이므로 이미지 센서와 현재 프레임의 블록 접근은 인터리빙에 의해 처리 될 수 있다.

T0 프레임과 T1 프레임 사이의 블록 매칭이 끝나면 이번에는 T1이 현재 프레임이 되고 T2가 참조 프레임이 되어 블록 매칭이 수행된다. 그리고 이미지 센서는 T3 프레임 이미지를 저장한다. 이와 같이 계속해서 현재 프레임과 참조 프레임을 번갈아 가면서 메모리 접근을 하게 함으로써 메모리 접근에 대한 충돌 문제를 해결하였다.

그림 6에는 본 논문에서 설계한 블록 매칭 회로의 데이터 경로가 나타나 있다.

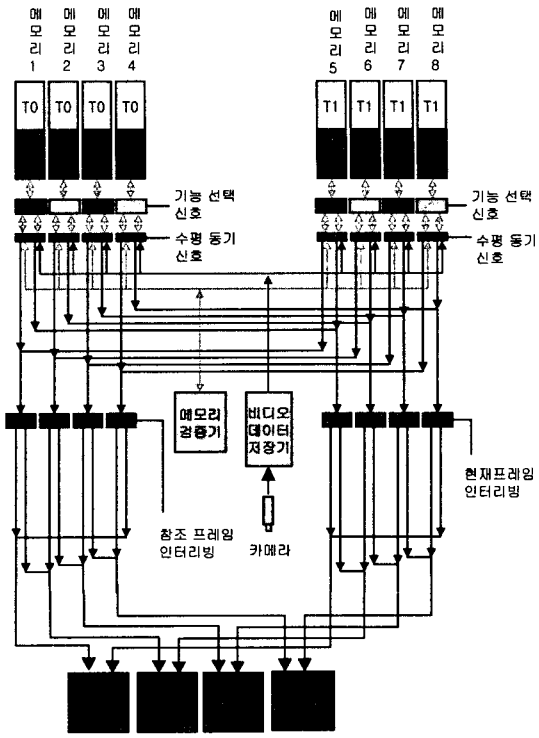

그림 6 블록 매칭 회로의 데이터 경로

4. 결론

본 논문에서는 블록 매칭을 이용한 실시간 차량 검지기 중에서 블록 매칭 부분을 설계하였다. 수정된 3-step 기법을 사용하여 처리 속도를 향상시켰다. 또한 한 프레임 이미지를 4개의 메모리 모듈에 분할하여 저장하고 한 줄의 블록들에 대해서는 같은 처리기를 할당함으로써 메모리 충돌을 줄일 수 있도록 하였다. 또한 이미지 센서가 프레임 이미지를 저장할 때에 더블 버퍼링 방식을 사용함으로써 실시간으로 블록 매칭을 수행할 수 있도록 하였다.

참고문헌

1. T. Koga, L. Iinuma, A.Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing", in Proc. NTC 81, pp. C9.6.1-9.6.5 Nov. 1981

2. R. Li, B. Zeng, M.L. Liou, "A New Three-Step Search Algorithm for Block Motion Estimation", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 4, No. 4, pp. 438-442, Aug. 1994