MOEPE: 스테레오 정합 하드웨어를 위한 *Merged Odd-Even PE* 구조

한 필 우 , 양 영 일
경상대학교 전자재료공학과
경남 진주시 가좌동 900
E-mail : pwhan@cecc-1.gsnu.ac.kr , yyang@nongae.gsnu.ac.kr

MOEPE: *Merged Odd-Even PE* Architecture for Stereo Matching Hardware

Phil-Woo Han, Yeong-Yil Yang
Members of Research Center for Aircraft Parts Technology
Dept. of Electronic Materials Eng. GyeongSang Nat'l. Univ.
900 Gazwa Dong, Chinju, KyungNam, 660-701, Korea

Abstract

In this paper, we propose the new hardware architecture which implements the stereo matching algorithm using the dynamic programming method. The dynamic programming method is used in finding the corresponding pixels between the left image and the right image. The proposed *MOEPE(Merged Odd-Even PE)* architecture operates in the systolic manner and finds the disparities from the intensities of the pixels on the epipolar line. The number of *PEs* used in the *MOEPE* architecture is the number of the range constraint, which reduced the number of the necessary *PEs* dramatically compared to the traditional method which uses the *PEs* with the number of pixels on the epipolar line. For the normal sized images, the number of *PEs* in the *MOEPE* architecture is less than that of *PEs* in the traditional method by 25 times. The proposed architecture is modeled with the VHDL code and simulated by the SYNOPSYS tool.

Ⅰ. 서 론

스테레오 시각은 서로 다른 위치에서 두 대의 카메라로 획득한 2차원 영상들로부터 3차원 거리 정보를 추출하며, 기본적으로 카메라 모델링, 영상 획득, 대응점 정합, 삼각 측량법에 의한 3차원 거리 정보 추출 단계 등으로 이루어진다. 특정한 목적을 위한 산업 분야에 실용적인 스테레오 시각 시스템이 등장하고 있으며, 가까운 장래에는 로봇에 시각 기능을 부여하는 것도 가능할 것으로 예측되고 있다.[1] 3차원 거리정보를 실시간으로 처리하기 위하여 전용 하드웨어가 필요하다.

스테레오 정합 알고리듬의 하드웨어 구조는 Guerra & Kanade[3] 등에 의해 제안되었으나, 이 구조에서 사용된 알고리듬[2]은 스테레오 영상에서 중요한 정보인 폐색 정보가 비용함수에 포함되지 않고, 두 화소의 대응점 정합 비용은 인접한 대응점뿐만 아니라 3 또는 4단계 떨어진 곳의 대응점들에 대한 정합 비용도 고려되므로 하드웨어 구현에 적합하지 않다. Guerra & Kanade[3] 등이 제안한 스테레오 정합 구조에서는 극상선상의 화소 수만큼의 *PE*가 사용된다.

본 논문에서는 Cox *et al*[4]이 제안한 스테레오 정합 알고리듬을 구현하는 하드웨어 구조를 개발하였다. Cox 등이 제안한 알고리듬에서는 폐색 정보에 대한 비용함수를 확률적으로 계산하고, 두 화소의 대응점 정합 비용은 두 화소의 밝기 값의 차와 인접한 대응점들의 비용과의 합에 의해 결정된다. 본 논문에서는 변이제약조건 *R*을 만족하면서 최소의 *PE*를 사용하는 *MOEPE* (*Merged Odd-Even PE*) 구조를 제안하였다. 예를 들어 변이 제약조건 *RANGE*가 9인 경우, 256×256 영상에서 변이를 구하기 위해서는 기존의 구조[3]에서는 256개의 *PE*가 사용되나 본 논문에서 제안한 구조에서는 9개의 *PE*가 사용되므로 VLSI로 구현이 가능하다.

Ⅱ. 동적 프로그래밍에 의한 스테레오 정합 알고리듬

좌우 영상의 극상선(epipolar line)의 화소 값들에 대하여 대응 쌍들을 구하기 위하여 동적 계획법(dynamic programming)이 사용된다. 동적 계획법에 의한 스테레오 정합은 극상선의 화소 값에 대하여 그림 1과 같이 탐색 영역을 만들고, 탐색 영역의 시작점에서부터 목표점에 이르는 최소비용 경로를 구함으로써 좌우 영상을 정합 시키는 것이다. 정합비용이 작다는 것은 좌우 영상의 화소 밝기의 차이가 작다는 것을 의미한다. 알고리듬 1은 Cox *et al*[4]이 제안한 좌우 영상의 극상선을 취하고 화소의 값으로부터 최소비용을 구하는 알고리듬이다.

$$\text{Occlusion} = \left\lfloor \ln\left(\frac{P_{D_S}{}^2 \phi}{(1-P_{D_S})} \frac{1}{|(2\pi)^d S_s^{-1}|^{1/2}} \right) \right\rfloor$$

```
for (i = 1; i ≤ N; i++) { C(i, 0) = i * Occlusion }
for (j = 1; j ≤ M; j++) { C(0, j) = j * Occlusion }
for (i = 1; i ≤ N; i++) {
    for (j = 1; j ≤ M; j++) {
        cost1 = C(i-1, j-1) + c(z_{1,i_1} , z_{2,i_2});
        cost2 = C(i, j-1) + Occlusion;
        cost3 = C(i-1, j) + Occlusion;
        C(i, j) = cmin = min(cost1, cost2, cost3);
        if(cost1 == cmin) P(i,j) =  0
        if(cost2 == cmin) P(i,j) =  1
        if(cost3 == cmin) P(i,j) =  2
    }
}
```

알고리듬 1. 최소 정합비용을 찾는 알고리듬의 슈도 코드

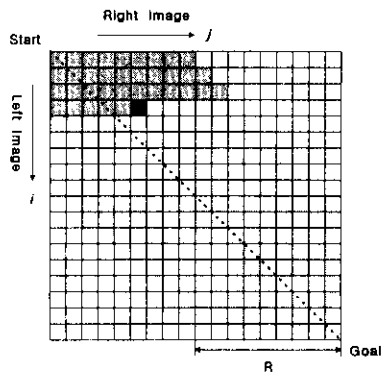

그림 1. 변이제약 조건이 R인 경우의 탐색영역

그림 1에서 컬럼은 왼쪽에서부터 순서대로 우 영상의 화소 값을 나타내고 로우는 위에서 아래로 순서대로 좌 영상의 화소 값을 나타낸다. 그림 1에서 각 요소는 좌 영상에서 극상선의 한 화소 (z_{1,i_1})와 우 영상에서 극상선상의 한 화소 (z_{2,i_2})의 쌍을 나타낸다. 검정으로 표시된 요소는 좌 영상에서 4번째 화소와 우 영상에서 6번째 화소 쌍에 해당된다. 계산 순서는 첫째 열에 대하여 좌측에서부터 우측으로 각 화소 쌍에 대하여 최소비용을 계산하고 완료되면 다음 열에 대하여 순서대로 계산하며 마지막 열까지 반복 수행한다. 그림 1과 같이 모든 화소 쌍에 대하여 최소비용을 구하지 않고 대각선에서 일정한 범위 내의 화소 쌍에 대해서만 탐색 영역으로 정의되는데, 이와 같이 변이가 일정한 값으로 한정 되어 있다고 가정하는 것을 변이제약 조건(*disparity constraint*)이라 한다.

그림 2(a)에서 회색으로 된 사각형들은 최소비용이 구하여진 화소 쌍들이고, 좌 영상의 i번째 화소와 우 영상의 j번째 화소(■로 표시된 사각형)는 최소비용 $C(i,j)$이 계산될 화소 쌍이다. 그림 2(b)에서 보는 것과 같이 화소 쌍 (i,j)에 도달하는 경로는 대각방향인 화소 쌍 $(i-1,j-1)$에서 화소 쌍 (i,j)로의 경로와 화소 쌍 $(i,j-1)$에서 화소 쌍 (i,j)로의 경로 그리고 화소 쌍 $(i-1,j)$에서 화소 쌍 (i,j)로의 경로가 존재한다. 각각의 경로에 대한 비용은 알고리듬 1에서 $cost1$, $cost2$, $cost3$으로 계산되고, 이는 정합 점에 대한 비용, 왼쪽 폐색 비용 및 오른쪽 폐색 비용을 나타낸다.

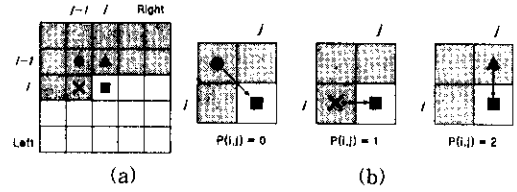

그림 2. (a) 최소비용 찾기 (b) 화소 쌍 (i,j)로의 가능한 세 가지 경로

Ⅲ. 스테레오 정합을 위한 하드웨어의 구조

3.1 시스톨릭 어레이 구조

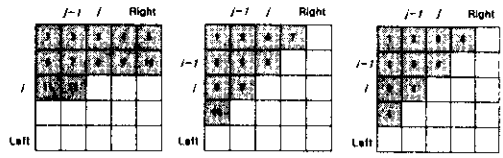

그림 3. 화소 쌍의 계산순서 (번호는 계산순서를 나타낸다.) (a) 동적 프로그래밍에서의 계산 순서 (b) 계산 순서 재배열 (c) 여러 개의 PE를 사용할 경우의 계산순서

동적 프로그램에서 계산순서는 그림 3(a)와 같고, 그림 3(b)와 같이 계산 순서를 바꿀 수 있다. 그림 3(b)를 살펴보면 2번째와 3번째 계산되는 화소 쌍들은 독립적으로 최소비용을 계산할 수 있다. 또한 4번째, 5번째, 6번째 계산되는 화소 쌍들도 서로 독립적으로 계산이 이루어질 수 있으므로, 그림 3(c)와 같이 계산순서를 변환할 수 있다.

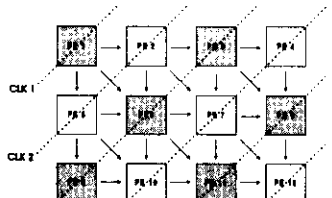

그림 4. 시스톨릭 어레이 구조

그림 3(c)와 같은 방법은 그림 4와 같은 데이터 흐름을 갖는 시스톨릭 어레이 구조로 구현될 수 있다. 화살

표는 데이터의 흐름을 나타낸다. 그림 4에서 동일한 점선 상의 *PE*들은 계산이 동시에 이루어지고, 계산 결과는 다음 점선 상의 *PE*들의 입력으로 사용된다. 짙은 회색으로 표시된 *PE*는 홀수 클럭에 활성화되는 *PE*이고, 엷은 회색으로 표시된 *PE*는 짝수 클럭에 활성화되는 *PE*이다. 홀수 클럭에 동작하는 *PE*들을 *Odd PE*, 그리고 짝수 클럭에 동작하는 *PE*들을 *Even PE*라 정의한다.

3.2 *Odd-Even PE* 구조

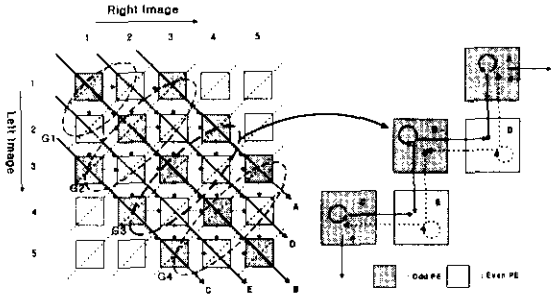

그림 5. *Odd-Even PE* 구조 및 데이터 흐름도

그림 5의 왼쪽 그림은 변이제약 조건이 3인 경우의 시스톨릭 구조를 보여준다. 동시에 계산이 이루어지는 *PE* 수는 홀수 클럭에는 3개이고, 짝수 클럭에는 2개이다. 이 절에서는 데이터의 흐름을 간단히 하기 위하여 그림 5의 오른쪽에는 홀수 클럭에 해당하는 요소 쌍을 계산하는 *Odd PE*와 그리고 짝수 클럭에 해당하는 요소 쌍을 계산하는 *Even PE*로 이루어진 *Odd-Even PE* 구조를 보여준다. 그림 5의 왼쪽 그림에서 *G1, G2, …*, 등은 연속된 홀수 클럭과 짝수 클럭에서 동작하는 *Odd PE*와 *Even PE*로 이루어진 그룹을 나타낸다. 그림 5의 왼쪽 그림에서 화살표로 표시된 같은 대각선 상의 *PE*들은 *Odd-Even PE* 구조에서 같은 *PE*에 의하여 계산이 이루어진다. *Odd-Even PE* 구조에서는 *G1*의 계산을 수행하고, *G2, G3, …* 에서 이루어지는 계산을 순차적으로 수행한다. *Odd-Even PE* 구조는 *Odd PE*가 먼저 계산을 수행한 후 *Even PE*가 계산을 수행하는 연산과정을 반복한다. *PE*들은 계산을 수행한 후 자신의 결과를 *PE*에 저장하여 인접한 *PE*에 보내준다. *G3*에서의 *Odd-Even PE*의 계산 과정을 살펴보자. *Odd PE*가 계산되기 위해서는 *G2*의 *Even PE*의 계산 결과와 *G2*의 *Odd PE*의 계산 결과가 요구된다. 이 데이터는 *Odd-Even PE* 구조에서 각각 *Even PE*에서의 데이터와 *Odd PE* 자신이 저장하고 있는 데이터를 사용하면 된다. *Even PE*가 계산되기 위해서는 *G3*의 *Odd PE*의 계산 결과와 *G2*의 *Even PE*의 데이터가 요구된다. 이 데이터는 *Odd-Even PE* 구조에서 각각 *Odd PE*의 계산 결과와 *Even PE* 자신이 저장하고 있는 계산 결과를 사용하면 된다.

3.3 *MOEPE(Merged Odd-Even PE)* 구조

그림 5의 *Odd-Even PE* 구조에서 보면 변이제약 조건인 *R*개의 *PE*가 동시에 계산이 이루어진다. 그림 5에 설명한 *Odd-Even PE* 구조는 홀수 클럭에는 *Even PE*가 계산을 하지 않고 짝수 클럭에는 *Odd PE*가 계산을 하지 않으므로 이를 병합하여 최소의 *PE* 수로 구성되어지는 *MOEPE* 구조를 제안하였다.

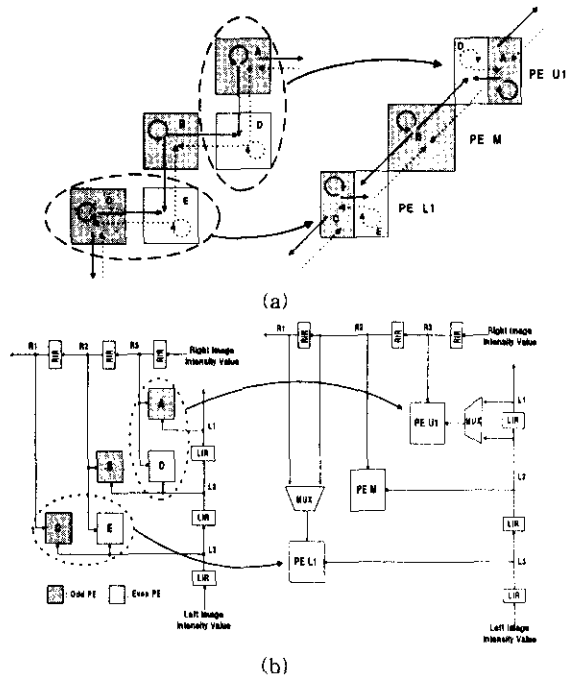

그림 6. (a) *MOEPE(Merged Odd-Even PE)* 구조 및 데이터 흐름도 (b) 화소 값을 고려한 *MOEPE* 구조

그림 6(a)의 왼쪽 그림은 *Odd-Even PE* 구조를 나타내고 오른쪽은 이를 개선한 *MOEPE(Merged Odd-Even PE)* 구조를 나타낸다. *MOEPE* 구조에는 계산에 필요한 최소한의 *PE*, 즉 변이 제약조건 *R*개의 *PE*들로 구성된다. *Odd-Even PE* 구조에서 2개의 *PE* (*Even PE*와 *Odd PE*)가 *MOEPE* 구조에서는 그림 6(a)와 같이 하나의 *PE*로 병합된다. 가운데 프로세싱 엘리먼트인 *PE B*는 병합이 이루어지지 않고, 가운데 *PE B*를 중심으로 오른쪽의 *PE*는 수직방향으로 *Odd PE*와 *Even PE*가 병합이 이루어지고, 아래의 *PE*는 수평방향으로 *Odd PE*와 *Even PE*가 병합이 이루어진다. 그림 6(a)의 *Odd-Even PE* 구조에서 *PE A*와 *PE D*의 연산은 *MOEPE* 구조에서 *PE U1*에서, *Odd-Even PE* 구조에서 *PE C*와 *PE E*의 연산은 *MOEPE* 구조에서 *PE L1*에서 이루어진다. 즉, *PE U1*은 홀수 클럭에는 *PE A*에서 이루어지는 연산을, 짝수 클럭에는 *PE D*에서 이루어지는 연산을 수행하고, *PE L1*은 홀수 클럭에는 *PE C*에서 이루어지는 연산을, 짝수 클럭에는 *PE E*에서 이루어지는 연산을 수행한다. 그리고 *PE M*은 *PE B*의 연산을 홀수 클럭에만 수행하고 짝수 클럭에는 연산을 수행하지 않는다.

그림 6(a)의 *MOEPE* 구조에서 데이터의 흐름을 살펴보자. *PE U1*은 각각 *PE A*와 *PE D*의 연산을 수행하는데, 홀수 클럭에 연산되어진 계산 결과(그림에서 실선의 데이터)는 다음 짝수 클럭에 *PE U1*에서(*PE U1*이 *PE D*의 연산을 수행할 때) 사용되어질 뿐만 아니라 다음 홀수 클럭에 *PE U1*에서도(*PE U1*이 *PE A*의 연산을 수행할 때) 사용이 된다. *PE L1*에서도 *PE U1*에서와 동일한 동작 특성을 갖는다. *PE M*은 홀수 클럭에만 연산이 이루어지고, 계산 결과를 인접한 *PE*로 보내고 자신의 *PE M*에 저장한 후 다음 홀수 클럭에 저장된 계산 결과를 사용한다.

각 *PE*에서 최소비용을 계산하기 위해서는 정합비용을 계산하기 위한 화소 쌍의 밝기 값이 주어져야 한다. 그림 6(b)는 화소 값이 고려된 *Odd-Even PE* 구조와 *MOEPE* 구조의 데이터 흐름을 보여준다.

Ⅳ. 스테레오 정합 하드웨어의 구현

그림 7은 제안된 스테레오 정합 하드웨어의 전체적인 구조이다. 제안된 스테레오 정합 하드웨어는 최소비용을 계산하기 위하여 *MOEPE* 구조를 구현하는 최소비용 모듈(*Minimum Cost Module*)과, 최소비용 모듈에서 발생한 최소비용 경로 정보로부터 최소비용 경로를 찾는 경로 생성기(*Path Generator*)와, 최소비용 경로로부터 변이를 추출하는 변이 추출기(*Disparity Extractor*)로 구성된다.

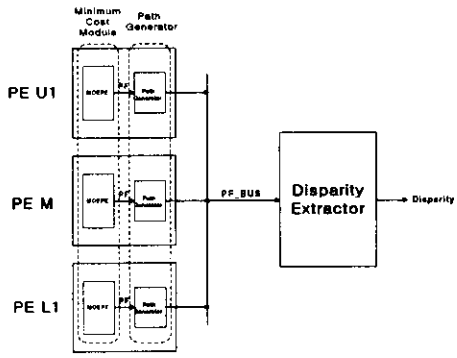

그림 7. 제안된 스테레오 정합 하드웨어의 전체 구조

그림 8은 *Merged Odd-Even PE* 구조의 프로세싱 엘리먼트(*PE*)의 회로도이다. *PE*는 외부로부터 다음 데이터를 받는다. i) *CVER* : 해당 *PE*의 위쪽에 있는 *PE*로부터의 최소비용, ii) *CHOR* : 해당 *PE*의 왼쪽에 있는 *PE*로부터의 최소비용, iii) I_L 과 I_R : 좌 영상과 우 영상에서 해당되는 화소의 밝기 값, iv) *OCC* : 폐색 비용. *PE*는 이 데이터로부터 최소비용을 구하고 최소비용 경로 정보를 저장한다. *cost1*이 *CMIN*(최소비용)일 때는 *PF="00"*이 되고, *cost2*가 *CMIN*일 때는 *PF="01"*이 되고, *cost3*가 *CMIN*일 때는 *PF="10"*이 된다. *PF*는 최소비용 경로 정보를 포함하고 있고, *PRF(Path Register File)*에 순차적으로 저장된다.

본 논문에서 제안한 스테레오 정합 하드웨어의 각 모듈들의 각 기능회로들을 VHDL로 모델링 한 후 SYNOPSYS tool을 사용하여 논리 합성한 후 동작 특성을 모의실험 하였다. *PE*를 구현하는데 소요되는 게이트 수는 6000개이고, *Path Generator*와 *Disparity Extractor*를 구현하는데는 각 60개와 120개의 게이트가 소요된다. *R*개의 *PE*로 이루어진 제안된 스테레오 정합 하드웨어는 $R\times$(하나의 *PE*의 게이트 수) + (*Path Generator*의 게이트 수) + (*Disparity Extractor*의 게이트 수)로 구현이 가능하므로, $R\times6000 + 60 + 120$ 개의 게이트로 구현된다.

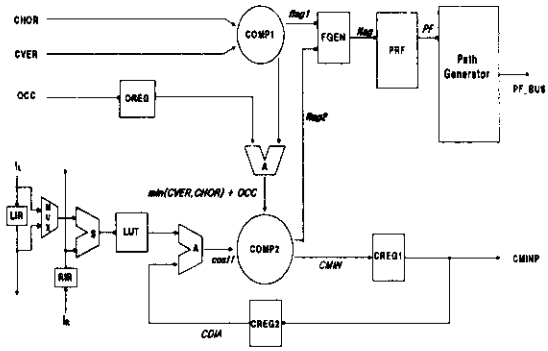

그림 8. 프로세싱 엘리먼트(*PE*)의 구조

Ⅴ. 결론

본 논문에서는 동적 프로그래밍에 기반 한 스테레오 정합 하드웨어를 설계하였다. *Minimum Cost* 모듈에서는 최소비용을 계산하고, *Path Generator*에서는 최소비용 경로를 계산하고, *Disparity Extractor*에서는 *Path Generator*에서 계산된 최소비용 경로를 이용하여 변이를 추출한다.

제안된 스테레오 정합 하드에어는 계산에 필요한 *PE*의 수를 변이제약 조건으로 제한함으로 256×256 영상에서 변이제약 조건이 9이라 가정하면 기존의 제안된 구조에 비해 *PE*의 수를 약 25배정도로 줄일 수 있다.

참 고 문 헌

[1] R. C. Jain and A. K. Jain, "Review of the NSF Range Sensing Workshop, *"Computer Vision and Pattern Recognition Conf.*, Ann Arbor, MI, May 1988.

[2] Y. Ohta and T. Kanade, "Stereo by intra- and inter-scanline search using dynamic programming," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 7, No. 2, pp. 139-154, 1985.

[3] C. Guerra and T. Kanade, "A Systolic Algorithm for Stereo Matching," *Proc. of International Workshop on Parallel Computing and VLSI*, Amalfi, Italy, May 1984.

[4] I. J. Cox, S. L. Hingorani and S. B. Rao, "A Maximum likelihood stereo algorithm," *Computer Vision and Image Understanding*, Vol. 63, No. 3, pp. 542-567, May 1996.

[5] N. Weste and D. J. Burr, "Dynamic Time Warp Pattern Matching Using an Integrated Multiprocessing Array," *IEEE Trans. on Computers*, Vol. c-32, No. 8, 1983.