논문 2008-45CI-1-6

실시간 스테레오 비젼 시스템을 위한 SAD 정합연산기 설계

(Development of a SAD Correlater for Real-time Stereo Vision)

이 정 수*, 양 승 구*, 김 준 성**

(Jongsu Yi, SeungGu Yang, and JunSeong Kim)

요 약

실시간 삼차원 영상은 충돌 방지를 위한 수동 시스템을 포함하는 다양한 응용 분야에 활용될 수 있으며, 기존 능동 시스템에 대한 훌륭한 대안으로서 잡음이 많은 복잡한 환경에서 외부의 영향을 최소화 할 수 있는 장점이 있다. 본 논문에서는 하드웨어 자원 사용량에 주목하여 실시간 삼차원 영상을 위한 스테레오 비전 시스템의 최적화에 관한 연구를 진행하였다. SAD 알고리즘은 규칙적인 구조, 선형적인 데이터 흐름과 풍부한 병렬성을 가지므로 재구성 가능한 하드웨어에서 구현하기 위한 좋은 조건을 가지고 있다. HDL을 이용하여 SAD 정합연산기를 설계하고 하드웨어 자원 사용량과 성능을 확인하기 위해서 Xilinx를 사용하여 합성하였다. 실험을 통하여, 초당 30프레임을 실시간으로 처리할 수 있는 충분한 처리 속도를 가지고 있으며, 적은 자원은 사용하면서 높은 정합율을 보이는 SAD 정합연산기를 설계하였음을 확인하였다.

Abstract

A real-time three-dimensional vision is a passive system, which would support various applications including collision avoidance, home networks. It is a good alternative of active systems, which are subject to interference in noisy environments. In this paper, we designed a SAD correlator with respect to resource usage for a real-time three-dimensional vision system. Regular structures, linear data flow and abundant parallelism make the correlation algorithm a good candidate for a reconfigurable hardware. We implemented two versions of SAD correlator in HDL and synthesized them to determine resource requirements and performance. From the experiment we show that the SAD correlator fits into reconfigurable hardware in marginal cost and can handle about 30 frames/sec with 640x480 images.

Keywords: SAD correlator, stereo vision, real time system, reconfigurable computing

I. 서 론

로봇의 비전시스템, 차량의 충돌방지시스템, 홈네트워크 시스템 등을 포함한 많은 분야에서 응용 가능한 비전 기반 시스템은 현재 많은 연구가 진행되고 있다. 이러한 비전 시스템의 제작에 있어서 필요한 요소 중 하나는 주변 환경에 대한 거리정보를 수집하는 것이다. 거리정보의 수집에 있어 전통적 방식인 초음파나 레이저 센서 등을 이용한 능동 시스템은 외부 환경에 민감하게 반응하기 때문에 잡음이나 간섭에 의해 정확한 정보를 얻기 힘들어 질 수 있다. 반면 수동 시스템은 능동 시스템에 비해 주변 환경의 변화에 대해 상대적으로 간섭이 적은 장점이 있어 잡음이 많은 환경에서 보다 효과적으로 사용될 수 있다. 스테레오 비전 시스템은 대표적인 수동 시스템 중 하나로서 주변 환경에 대한 세부화된 3차원 정보를 제공한다[1~2].

스테레오 비전 시스템의 알고리즘을 처리하는데 있어서 소프트웨어 방식을 사용하면 200x200 화소 크기를 가지는 상대적으로 작은 크기의 영상으로 가정하고 응용 분야에 따른 전-후처리 과정을 무시하더라도 초단위의 시간이 필요하다[3~4]. 이러한 소프트웨어 처리 방법은 실제 응용시스템에서 사용하기에는 처리속도가 매우

* 학생회원, ** 정회원, 중앙대학교 전자전기공학부
(School of Electrical and Electronics Engineering, Chung-Ang University)

※ 본 논문은 정보통신부 정보통신연구진흥원 지원의 HNRC(HomeNetwork Research Center)-ITRC 사업의 지원으로 수행되었음.

접수일자: 2007년10월25일, 수정완료일: 2008년1월11일

느리다. 따라서 실시간 처리를 요구하는 응용시스템을 위해서는 하드웨어를 이용한 빠른 계산이 필수적이다.

본 논문에서는 스테레오 비전 시스템의 대표적인 알고리즘 중 하나인 SAD(Sum of Absolute Difference) 알고리즘을 사용하는 정합연산기를 하드웨어로 구현하고, 하드웨어 자원 소모를 최소화할 수 있도록 설계한다. Ⅱ장에서는 스테레오 비전 관련 정합 알고리즘과 본 논문에서 사용된 SAD 정합 알고리즘에 대해서 간략히 소개한다. Ⅲ장에서는 SAD 정합연산기의 구성 요소와 설계 과정을 살펴보고, Ⅳ장에서는 시뮬레이션 결과를 분석하여 하드웨어 자원 소모의 타당성과 실시간 동작에 대하여 분석한다. 마지막으로 Ⅴ장에서는 본 논문을 종합적으로 정리하면서 마무리 짓는다.

Ⅱ. 배 경

1. 영상 정합

영상 정합은 스테레오 영상을 분석하는 핵심 영역이다. 스테레오 비젼 시스템의 영상 정합에 있어 중요한 요소는 보는 관점이 서로 다른 두 개 영상간의 차이이다. 스테레오 비젼 시스템은 두 영상간의 차를 이용하여 한쌍의 스테레오 영상에서 개개의 화소들에 대한 대응점을 찾는데 그 목적을 둔다. 대응점 간의 가로축의 차이를 이용하여 해당 화소 또는 영역의 거리 정보를 구할 수 있다. 스테레오 영상 정합 방법에는 대표적으로 영역 기반(area-based) 정합 방법과 특징 기반(feature-based) 정합 방법의 2가지가 있다[5].

영역 기반 정합 방법은 윈도우(window:일정크기의 이웃 화소들의 집합 단위)의 영역을 연속적으로 비교하는 알고리즘이다. 그림 1은 영역 기반 정합 알고리즘을 이용한 영상 정합 과정을 보여주며 오른쪽 영상의 기준 윈도우 대하여 왼쪽 영상의 윈도우를 가로축으로 이동시키면서 각 윈도우 쌍간의 유사성을 계산한다. 오른쪽 영상의 기준 윈도우에 대하여 비교 계산되는 왼쪽 영상

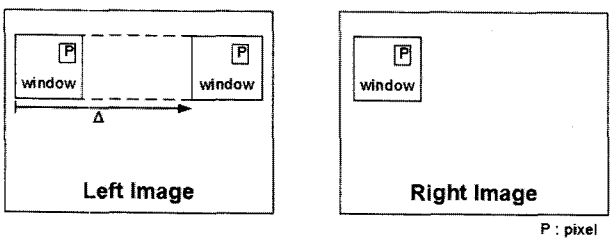
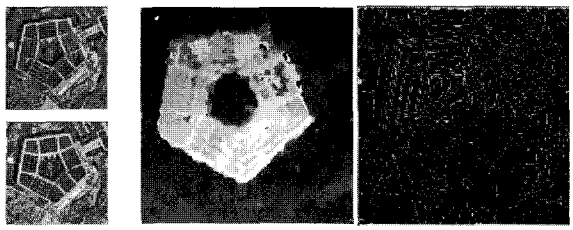


그림 1. 영역 기반 정합 알고리즘의 영상 정합 과정
Fig. 1. An area-based correlation algorithm.



a) 입력 영상쌍 b) 영역 기반 결과 c) 특징 기반 결과

그림 2. 영상 정합의 예
Fig. 2. Examples of image correlations.

의 윈도우의 영역은 최대시차(maximum disparity, Δ)에 의해 한정되며 Δ 범위 내의 윈도우 중 유사성이 가장 높은 왼쪽 영상의 윈도우로 그 대응점을 결정한다. 그림 2(b)는 영역 기반 정합 방법을 이용한 스테레오 정합의 예를 보여주며, 좌우 스테레오 영상의 비교를 통해서 가까운 부분은 밝게, 먼 부분은 어둡게 표현되어 있다.

이에 반해 특징 기반 정합 방법에서는 경계선이나 윤곽선 등과 같은 영상의 특징을 추출하여 그 특징 공간에서 정합시키는 것으로 처리속도를 향상시킬 수 있으나 영상의 전체 영역에 대한 조밀한 변이 추정은 불가능하며 보간(interpolation)과정을 거쳐야 하는 단점이 있다.[6] 그림 2(c)는 특징 기반 정합 방법을 이용한 특징점 추출의 예를 보여주며, 영상의 경계면에 해당하는 부분은 밝게, 경계가 아닌 부분은 어둡게 표현되어 있다.

2. SAD 알고리즘

본 논문에서는 영역 기반 정합 알고리즘의 한 종류인 SAD 알고리즘을 구현하였다. 이는 최적의 정합을 각 윈도우의 대응되는 화소간 차이에 대한 절대값의 합(SAD값)이 최소가 되는 화소의 변이값으로 정의한다[7]. 식(1)은 개별 윈도우의 SAD 값을 구하는 과정을 나타낸다.

$$C(x,y,\delta)=\sum_{y=0}^{wh-1}\sum_{x=0}^{ww-1}|I_R(x,y)-I_L(x+\delta,y)| \quad (1)$$

wh : 윈도우 높이
ww : 윈도우 넓이
δ : 시차값
$I_R(x,y)$: 왼쪽 영상의 좌표에 대응하는 화소값
$I_L(x,y)$: 오른쪽영상의 좌표에 대응하는 화소값

두 영상 사이의 최적의 정합점을 계산해내기 위하여 시차값 δ를 0에서 최대시차 Δ까지 변화시키면서 계산

되는 Δ+1개의 $C(x,y,\delta)$ 값 중 최소값을 갖는 δ를 최적 정합점으로 선택한다. SAD 정합 알고리즘은 Δ값에 따라 계산할 수 있는 거리 범위가 결정되며, Δ값이 커질수록 카메라에서 가까운 물체에 대한 인식이 가능하게 된다. 그러나 Δ를 지나치게 크게 설정할 경우 계산해야 하는 $C(x,y,\delta)$ 값의 수가 증가하므로 계산량이 증가하게 되고, 비슷하게 생긴 다른 물체가 최대시차 범위내에 존재할 경우 최적 정합점 계산에 있어 오류가 발생할 가능성이 증가한다. 반면, Δ를 지나치게 작게 설정하는 경우 기준 시야 이내에 접근하는 물체를 인식하지 못하는 오류가 발생하므로 스테레오 비젼 시스템에서 필요로 하는 적절한 거리 정보를 얻어낼 수 없다. 따라서 최대시차 Δ는 계산량을 고려하여 설정하되 지나치게 작아져서 영상 정합 과정에 오류가 생기지는 않도록 해야 한다.

Ⅲ. SAD 정합 연산기 설계

SAD 정합 알고리즘은 규칙적인 구조를 가지며, 재구성 가능한 하드웨어에서 구현함에 있어 풍부한 병렬성을 포함하는 데이터 흐름을 가지므로 병렬처리가 가능하다. 더불어, SAD 정합 알고리즘의 연산은 식(1)에서 보이는 바와 같이 FPGA에서 충분한 자원을 제공하는 덧셈기와 비교기만을 이용하여 구현할 수 있다.

본 논문에서는 실시간 스테레오 비전 시스템에서 사용가능한 SAD 정합연산기를 구현하였다. SAD 정합연산기 v1.0에서는 식(1)의 SAD 알고리즘을 그대로 구현하였으며, v1.1에서는 윈도우를 행과 열로 분리하여 계산함으로써 v1.0과 동일한 성능을 가지면서 자원 사용량을 현격히 줄였다.

1. SAD 정합 연산기 v1.0

그림 3은 SAD 정합연산기 v1.0의 블록다이어그램으로 크게 세 개의 모듈(Pixel Shift Register, Disparity Calculator, Minimum Calculator)로 구성되어 있다. 그림에서 볼 수 있는 것처럼, 각각의 시차값 δ에 대한 SAD값을 계산하기 위한 블록들이 독립적인 하드웨어 자원을 가지므로 계산 과정은 병렬적으로 수행된다.

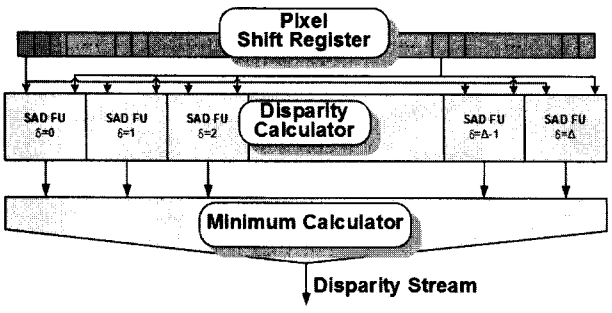


그림 3. SAD 정합연산기 v1.0의 블록다이어그램
Fig. 3. A block diagram of the SAD correlator v1.0.

► Pixel Shift Register 모듈

그림 4는 Pixel Shift Register 모듈에 대한 블록다이어그램이다. Pixel Shift Register 모듈은 SAD 정합 윈도우에 대한 모든 화소들이 동시에 SAD 값을 계산하기 위하여 충분한 수의 화소를 제공할 수 있도록 "scan line length × (wh - 1) + Δ"만큼의 길이를 갖는 1차원 배열 형태로 구성된다. 그림 4에서는 좀 더 쉽게 알아볼 수 있도록 하기 위해서 scan line의 길이만큼으로 잘려져 있는 것으로 묘사하였다. SAD 정합연산에서 병렬적으로 계산을 수행하기 위하여 Left Pixel Shift Register에서 기준으로 삼는 하나의 윈도우와 Right Pixel Shift Register에서 비교를 위한 Δ+1개의 윈도우들이 동시에 선택되어 Disparity Calculator 모듈로 전송된다.

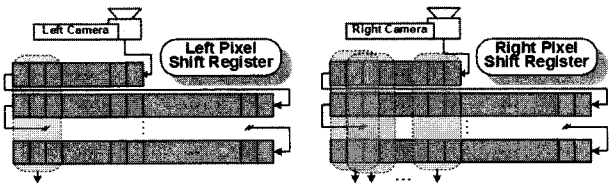


그림 4. Pixel Shift Register 모듈의 내부 구조
Fig. 4. A block diagram of the Pixel Shift Register,

► Disparity Calculator 모듈

그림 5는 Disparity Calculator 모듈의 내부 구조를 상세하게 나타낸 것이다. Disparity Calculator 모듈은 Left Pixel Shift Register로부터 입력받은 한 개의 윈도

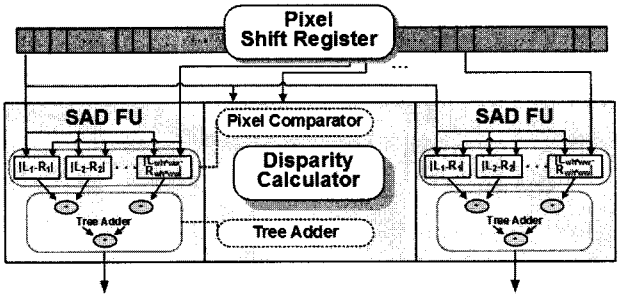


그림 5. Disparity Calculator 모듈의 내부 구조
Fig. 5. A block diagram of the Disparity Calculator module.

우를 Right Pixel Shift Register로부터 입력받은 Δ+1개의 윈도우들과 비교하는 모듈로서 Δ+1개의 SAD FU(Functional Unit)으로 구성된다. SAD FU은 Pixel Shift Register 모듈에서 입력받은 한 쌍의 윈도우를 Pixel Comparator 블록을 통해 화소 단위로 비교하여 차이를 구한 뒤, 그 절대값을 Tree Adder 블록의 덧셈 연산을 통해 총합을 구하여 Minimum Calculator 모듈로 결과값을 전달한다.

SAD 정합연산기에서 지연시간을 유발하는 가장 중요한 요인은 SAD FU의 덧셈 연산으로서 성능 개선을 위하여 트리 형태의 덧셈기 집합을 사용하였다. HDL 모델로 구현하기 평이한 간단한 순차 덧셈기를 사용하는 경우 $\Theta(wh+ww-2)$의 지연 시간이 필요한 반면에 Tree Adder는 성능이 $\Theta(\log(wh+ww-2))$로 개선될 수 있으며 또한 회로를 매우 규칙적으로 생성할 수도 있다[8~9].

► Minimum Calculator 모듈

그림 6은 Minimum Calculator 모듈의 블록다이어그램이며 Disparity Calculator와 같이 트리 형태의 덧셈기 집합으로 구성하였다. Minimum Calculator 모듈에서는 Disparity Calculator 모듈의 출력인 Δ+1개의 SAD값에 대하여 가장 작은 값을 갖는 δ를 찾는다. 이 정보가 각 클럭 단위로 출력되어 우리가 원하는 depth map을 포함하여 물체와의 거리정보를 계산할 수 있다.

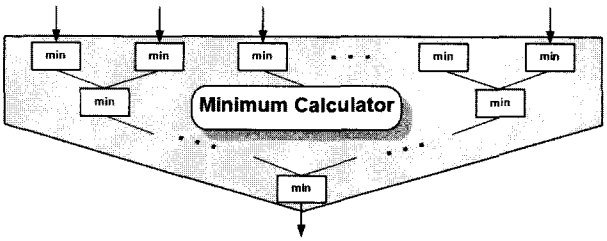


그림 6. Minimum Calculator 모듈의 내부 구조
Fig. 6. A block diagram of the Minimum Calculator.

2. SAD 정합 연산기 v1.1

SAD 정합연산기 v1.0는 매 클럭마다 윈도우가 한 화소씩 이동하여 연산되므로 하나의 윈도우를 기준으로 $(ww-1)$개의 열이 중복되어 연산된다. SAD 정합연산기 v1.1은 윈도우 단위의 연산을 열 단위 연산으로 변경하여 중복되는 연산을 최소화 하였다. 즉 SAD 정합연산기 v1.0에서는 Pixel Shift Register 모듈에서 윈도우 단위로 데이터를 전달하여 연산하는 것에 비하여 v1.1에서는 윈도우의 열 단위로 데이터를 전달한다. 매 클럭마다 화소의 이동으로 인한 새로운 열에 대해서만 연산하고 그 값을 버퍼에 저장하며, 나머지 $(ww-1)$개의 열에 대해서는 버퍼에 저장된 값을 이용하여 전체 윈도우에 대한 연산을 수행함으로써 덧셈기의 수를 줄이면서 동일한 성능을 유지하도록 하였다.

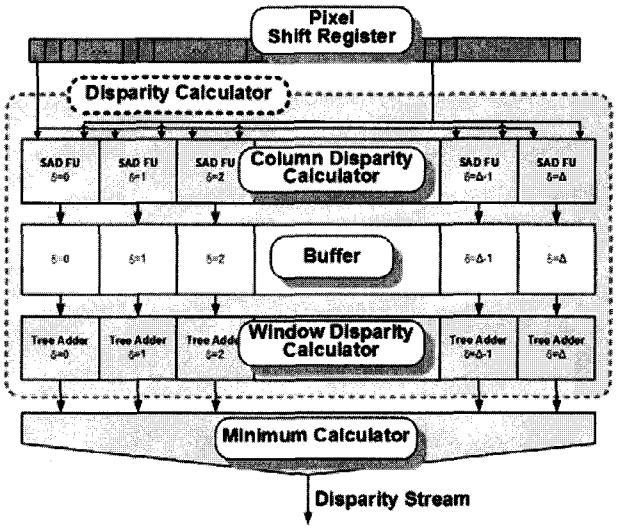


그림 7. SAD 정합연산기 v 1.1 블록다이어그램
Fig. 7. A block diagram of the SAD correlator v 1.1.

그림 7은 SAD 정합연산기 v1.1의 블록 다이어그램으로써 Pixel Shift Register, Minimum Calculator 모듈은 v1.0과 동일한 구조의 모듈을 사용한다. 그림 8은 SAD 정합연산기 v1.1의 Disparity Calculator 모듈의 의 세부 블록 다이어그램을 보여준다. 앞서 설명한 바와 같이 v1.1에서는 열단위로 연산을 수행함에 따라

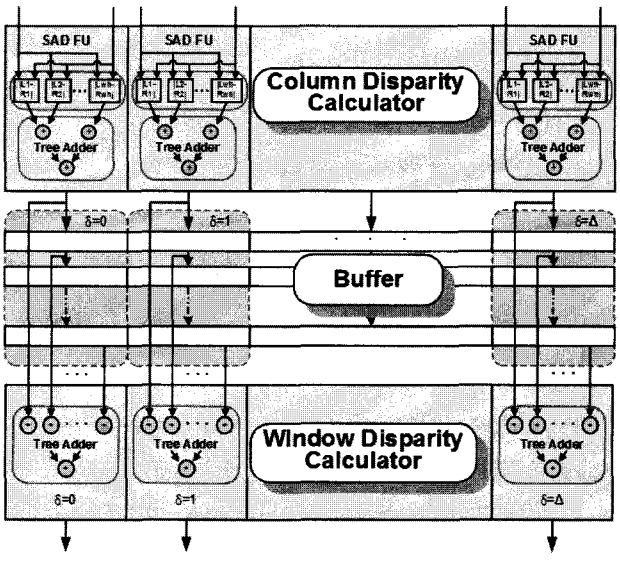


그림 8. Disparity Calculator 모듈 v1.1의 블록다이어그램
Fig. 8. A block diagram of the Disparity Claculator v1.1.

Column Disparity Calculator, Buffer, Window Disparity Calculator 서브 모듈로 분할하여 설계하였다.

Column Disparity Calculator에서는 Left Pixel Shift Register로부터 입력받은 한 개의 열을 Right Pixel Shift Register로부터 입력받은 Δ+1개의 열들과 비교하는 모듈로서 Δ+1개의 SAD FU으로 구성된다. SAD FU에서 연산된 결과값은 윈도우의 열에 대한 시차값이 되며 Buffer 모듈에 저장된다. Buffer 모듈은 현클럭에서 새로이 연산된 값을 포함하여 하나의 윈도우를 구성하는 ww열에 대한 계산값들을 쉬프트하여 저장할 수 있도록 구성되며 $(\Delta+1)\times ww$의 크기를 가진다. 한 쌍의 윈도우에 대한 SAD값의 계산을 위하여 해당 δ값의 Buffer에 저장되어 있는 ww개의 결과값이 Window Disparity Calculator의 대응되는 Tree Adder 블럭으로 전달된다. Window Disparity Calculator는 각 Tree Adder 블럭을 통하여 각 윈도우쌍에 대한 SAD 연산을 완료하며 그 결과 값인 시차값을 Minimum Calculator 모듈로 전달한다.

IV. SAD 정합연산기 구현 결과

SAD 정합연산기의 프로토타입을 구현하기 위하여 Xilinx Virtex-II XC2V8000를 사용하였다. XC2V8000은 800만 게이트의 크기를 가지며 공개된 메모리 블록은 다수의 단일 포트 혹은 이중 포트로 배치되도록 프로그래밍 될 수 있으며, 1.8V 의 공급 전압으로 400 MHz 이상의 내부 클럭을 사용할 수 있다[10].

1. SAD 정합연산기의 성능 비교

그림 9는 앞에서 구현한 SAD 정합연산기에서 최대 시차 Δ값을 32로 설정하여 Virtex-II를 기반으로 합성하였을 때, 윈도우 크기의 변화에 따른 하드웨어 자원 사용량을 보여준다. SAD 정합연산기 v1.1가 v1.0에 비하여 60% 이하의 적은 하드웨어 자원을 사용하는 것을 알 수 있다. SAD 정합연산기 v1.0의 경우 16×16 윈도우 크기를 가질 경우 XC2V8000의 최대 허용량을 초과하여 합성이 불가능하였으나, v1.1에서는 25,000 슬라이스 가량에서 합성이 가능하였다.

그림 10은 SAD 정합연산기의 성능평가를 위하여 사용한 여러 종류의 테스트 영상 중 tsukuba 영상[11~12]에 대하여 8x8 윈도우 크기를 가지는 SAD 정합연산기의 시뮬레이션 결과 영상을 보여준다. v1.1의 경우 중복된 덧셈기를 대폭 줄여 하드웨어 자원 소모량은 획기적으로 감소하였으나 시뮬레이션 결과가 v1.0과 일치한다. 따라서 SAD 정합연산기 v1.1은 성능상의 차이는 없으면서 적은 하드웨어 자원을 사용하는 설계임을 확인할 수 있다.

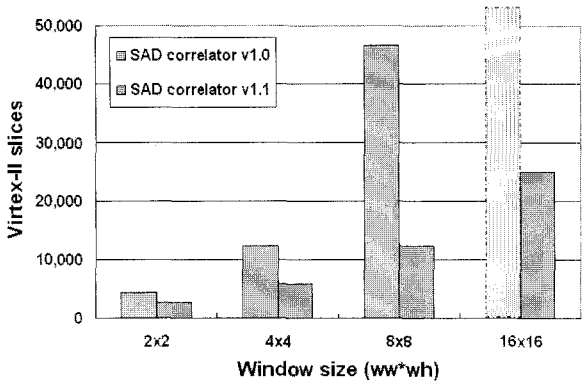
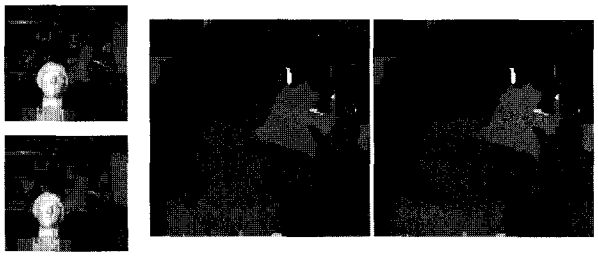


그림 9. 윈도우 크기의 변화에 따른 SAD 정합연산기의 하드웨어 자원 사용량

Fig. 9. Resource usage of the SAD correlators in varying window sizes.



a) 입력 영상쌍 c) v1.0 결과 d) v1.1 결과

그림 10. tsukuba 영상에 대한 시뮬레이션 결과 영상

Fig. 10. Sample simulation results from tsukuba image.

2. SAD 정합연산기의 실시간 동작

그림 11은 SAD 정합연산기에서 윈도우 크기는 16x16, 최대시차 Δ는 64, 시스템 클럭속도는 10MHz로 설정하고 320x240 크기를 가지는 tsukuba 영상에 대한 시뮬레이션 수행 결과 파형을 보여준다. 하나의 영상을 처리하는데 걸리는 총 수행 시간은 약 8,070μs로 이는 시스템이 초당 약 123장의 영상을 처리할 수 있음을 보여준다. 영상을 처리하는데 걸리는 시간은 스테레오 비전 시스템에서 처리하고자 하는 영상의 크기, 윈도우의 크기, SAD 정합연산기 클럭수와 시스템의 클럭 속도(cct)에 영향을 받는다. 640x480 크기의 영상을 사용하는 경우 처리할 화소수가 4배 증가하므로 초당 약 30프레임을 처리할 수 있음을 알 수 있다. 이는 전형적인 카메라가 제공하는 초당 30프레임의 영상에 대해서도 실시간으로 처리가 가능함을 의미한다.

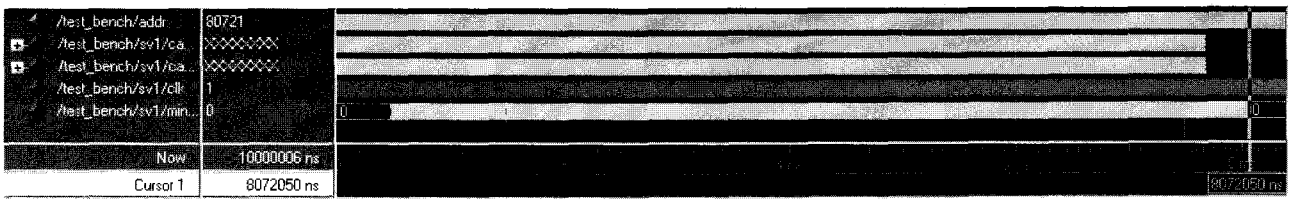


그림 11. tsukuba 영상을 사용한 시뮬레이션 결과 파형
Fig. 11. A simulation wave form using tsukuba image.

표 1. SAD 정합연산기 v1.1의 구성 모듈별 하드웨어 자원 사용량 및 지연시간
Table 1. Hardware usages and delay estimation of the SAD correlator v1.1.

모듈	자원사용량	지연시간
Shift Register	$2(sl\times(wh-1)+\Delta+1)\times c_{reg}$	t_{reg}
Column Disparity Calculator	$(\Delta+1)\times(wh+wh-1)\times c_{add}$	$(1+\log_2 wh)\times t_{add}$
Buffer	$ww\times(\Delta+1)\times c_{reg}$	t_{reg}
Window Disparity Calculator	$(\Delta+1)\times(ww-1)\times c_{add}$	$\log_2 ww\times t_{add}$
Minimum Calculator	$\Delta\times c_{add}$	$\log_2(\Delta+1)\times t_{add}$

c_{reg} = 레지스터 자원량 c_{add} = 덧셈기 자원량 t_{reg} : 레지스터 지연시간 t_{add} : 덧셈기 지연시간

표 1은 SAD 정합연산기 v1.1의 구성 모듈별 자원사용량과 지연시간을 보여준다. 버퍼에서의 지연시간이 덧셈기의 지연시간에 비하여 작으므로 $t_{buf}\approx t_{add}$로 가정하고 SAD 정합연산기의 시스템 클럭 속도값을 각 모듈에서의 지연시간의 총합으로 계산하면

$$T_{delay}\approx(\log_2 wh+\log_2 ww+\log_2(\Delta+1)+3)t_{add} \quad (2)$$

윈도우 사이즈 16x16, Δ값 64인 경우 T_{delay}값은 약 20tadd이다. Xilinx Virtex-II XC2V8000의 경우 덧셈기의 지연시간은 speed grade -5에서 3.726ns, speed grade -4에서 4.284ns의 값을 가진다. 덧셈기의 지연시간을 5ns로 가정하는 경우 총 T_{delay}는 약 100ns이 되며 앞서 예로 사용한 320x240 크기의 영상을 사용한다고 하면 총 수행시간은 약 7,680μs로 Pixel Shift Register 모듈에 화소값을 채우기 위한 지연시간을 고려하면 시뮬레이션 결과 값인 8,070μs와 근사한 값을 보인다. 따라서 식(2)를 사용하면 설정에 따른 SAD 정합연산기의 수행시간을 예측할 수 있다.

SAD 정합연산기 v1.1의 경우 구조적으로 5가지 기능 모듈에 대한 순차적 구성을 가지므로 쉽게 파이프라이닝 기법이 적용될 수 있다. 각 기능모듈의 수행시간이 동일하다는 가정을 할 경우 산술적으로 $T_{delay}'=T_{delay}/5$까지 성능을 향상시킬 수 있다. 표 1에 기술된 각 기능모듈의 수행시간을 고려하면 최대 지연시간을 가지는 Minimum Calculator를 기준으로 하는 경우에도 $T_{delay}'=\log_2\Delta\times t_{add}$까지 성능을 향상시킬 수 있다. 더 나아가 이론적으로는 덧셈기 단위로 파이프라인 단계를 더욱 증가시킨 구성이 가능하며 이를 통하여 시스템의 처리 속도가 충분히 빨라진다고 가정하면 현재 연산 가능한 영상이나 윈도우 크기보다 더욱 큰 값을 가지는 시스템이나 더욱 정확한 결과를 위한 전후처리 과정을 포함하는 비전 시스템을 구현할 수 있을 것으로 기대된다.

V. 결 론

스테레오 비젼 시스템은 능동 시스템에 비해 간섭이 적은 장점이 있으며 주변 환경에 대한 세부화된 3차원 정보를 제공한다. 스테레오 비전 시스템의 SAD(Sum of Absolute Difference) 알고리즘을 처리하는데 있어 실시간 처리를 요구하는 응용시스템을 위해서는 하드웨어를 이용한 빠른 계산이 필수적이다. 본 논문에서는 SAD 정합연산기를 하드웨어로 구현하고, 높은 정합율을 유지하면서 하드웨어 자원 소모를 최소화할 수 있도록 설계하였다.

구현된 SAD 정합연산기는 640x480 크기의 영상에 대하여 전형적인 카메라가 제공하는 초당 30프레임의 영상에 대해서도 실시간으로 처리가 가능함을 확인하였다. 차후 연구를 통하여 파이프라이닝 기법을 적용하여 구성할 경우 현재의 스펙보다 더 큰 값을 가지거나 구

현하고자 하는 응용분야의 목적에 따른 전후처리 과정을 포함하는 실시간 비전 기반 시스템을 구현할 수 있을 것으로 기대된다.

참 고 문 헌

[1] C. F. Olson, "Maximum-likelihood image matching," Pattern Analysis and Machine Intelligence, IEEE Transactions on , Volume: 24, Issue: 6 , pp. 853 - 857, June 2002

[2] N. Sebe, M. S. Lew, "Maximum Likelihood Stereo Matching," Pattern Recognition, 2000. Proceedings. 15th International Conference on, Volume: 1, pp. 900 - 903 vol.1, September 2000.

[3] J. S. Yi, J. S. Kim, LiPing, J. Morris, G. Lee, P. Leclercq, "Real-time three dimensional vision," Advances in Computer Systems Architecture: 9th Asia-Pacific Conference, ACSAC 2004, pp. 309-320, Beijing, China, September 2004.

[4] S Wong, S. Vassiliadis, S. Cotofana, "A Sum of absolute differences implementation in FPGA Hardware," Euromicro Conference, 2002. Proceedings. 28th, pp. 183-188, September 2002.

[5] S. T. Barnard, M. A. Fischler, "Computational Stereo," ACM Computing Surveys archive Volume 14, Issue 4, pp. 553 - 572, December 1982.

[6] W. Eric, L. Grimson, "Computatoinal Experiments with a Feature Based Stereo Algorithm," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-7, No. 1, pp. 17-33, 1985.

[7] P. Leclercq, J. Morris, "Robustness to Noise of Stereo Matching," Image Analysis and Processing, 2003. Proceedings. 12th International Conference on , pp. 606 - 611, September 2003.

[8] P. J. Ashenden, "Recursive and repetitive hardware models in VHDL," Tech. Rep. TR160/12/93/ECE, Electrical and Computer Engineering, University of Cincinatti, Ohio 45221-0030, 1993.

[9] J. Morris, "Reconfigurable computing," Computer Engineering Handbook, Vojin G Oklobdzija, Ed. CRC Press, pp. 37-1 - 37-16, 2001.

[10] Xilinx Inc. : Virtex-II Platform FPGAs : Complete Data Sheet, "http://www.xilinx.com," 2002.

[11] D. Scharstein, R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," International Journal of Computer Vision, vol. 47, pp. 7-42, 2002

[12] Middlebury Stereo Vision Page, http://www.middlebury.edu/stereo

저 자 소 개

이 정 수(학생회원)
2002년 중앙대학교 전자전기 공학부 학사 졸업.
2004년 중앙대학교 전자전기 공학부 석사 졸업.
2004년~현재 중앙대학교 전자 전기공학부 박사 과정.
<주관심분야 : 컴퓨터구조, SoC 설계, 영상처리, 홈네트워크>

양 승 구(학생회원)
2006년 중앙대학교 전자전기 공학부 학사 졸업.
2006년~현재 중앙대학교 전자 전기공학부 석사 과정.
<주관심분야 : 컴퓨터구조, 영상처리, 임베디드시스템 설계>

김 준 성(정회원)
1991년 중앙대학교 전자공학과 학사 졸업.
1993년 중앙대학교 대학원 전자공학과 석사 졸업.
1998년 미국 미네소타대학교 전기공학과 박사 졸업.
2002년~현재 중앙대학교 전자전기공학부 부교수.
<주관심분야 : 컴퓨터구조, 병렬처리 시스템, 홈네트워크, SoC 구조, 시스템 성능 분석>