논문 2014-09-41

FPGA를 이용한 실시간 영상 워핑 구현

(An Implementation of Real-time Image Warping Using FPGA)

류 정 래, 이 은 상, 도 태 용*
(Jung Rae Ryoo, Eun Sang Lee, Tae-Yong Doh)

Abstract : As a kind of 2D spatial coordinate transform, image warping is a basic image processing technique utilized in various applications. Though image warping algorithm is composed of relatively simple operations such as memory accesses and computations of weighted average, real-time implementations on embedded vision systems suffer from limited computational power because the simple operations are iterated as many times as the number of pixels. This paper presents a real-time implementation of a look-up table(LUT)-based image warping using an FPGA. In order to ensure sufficient data transfer rate from memories storing mapping LUT and image data, appropriate memory devices are selected by analyzing memory access patterns in an LUT-based image warping using backward mapping. In addition, hardware structure of a parallel and pipelined architecture is proposed for fast computation of bilinear interpolation using fixed-point operations. Accuracy of the implemented hardware is verified using a synthesized test image, and an application to real-time lens distortion correction is exemplified.

Keywords : Image warping, Look-up table, Real-time, FPGA, Backward mapping, Fixed-point operation.

Ⅰ. 서론

디지털 반도체 및 영상 센서의 발전에 힘입어 다양한 산업 분야에서 비전 시스템의 활용이 증가하고 있다. 핸드폰과 노트북 컴퓨터에는 영상 센서가 기본으로 장착되고 있고, 보안 시스템과 생산 설비, 엔터테인먼트 산업 및 자동차 산업에 이르기까지 그 응용 분야는 급격히 확대되고 있다. 응용 분야의 확장과 함께 비전 시스템에 대한 소형 경량화 요구에 따라 임베디드 비전 시스템(Embedded Vision System)의 수요가 증가하고 있다. 하지만, PC 등 일반 컴퓨터에 비하여 다소 부족한 연산 처리 능력으로 인하여 임베디드 비전 시스템의 확산에 어려움이 있다.

임베디드 비전 시스템에서의 처리 속도 문제를 해결하는 방법으로 전체 영상처리 알고리즘 중 일부를 하드웨어로 구현하는 방안이 관심을 모으고 있다. 하드웨어는 병렬 구조 및 파이프라인 구조를 활용하여 많은 양의 반복적인 연산을 빠르게 처리할 수 있는 장점이 있기 때문이다[1]. 영상 처리 분야에서는 픽셀 단위의 연산으로 구성되는 낮은 수준의 영상처리 알고리즘이 이에 해당하는데, 영상 필터링과 영상 워핑(image warping) 등이 하드웨어 구현에 적합한 대표적인 영상처리 기법이다.

영상 워핑은 입력 영상 각 픽셀의 좌표를 변환하여 2D 영상을 기하학적으로 변형하는 영상 처리 알고리즘으로서 대표적 활용 예로는 렌즈 왜곡 보정[2], 스테레오 매칭을 위한 영상 정렬(image rectification)[3], 그리고 파노라믹 영상 합성을 위한 영상 스티칭(image stitching)[4] 등을 꼽을 수 있다. 이와 관련된 기존 연구에서는 카메라 캘리브레이션 등의 방법을 활용하여 입력 영상을 출력 영상으로 변환하기 위한 변환 관계를 추출하는 내용

*Corresponding Author(dolerite@hanbat.ac.kr)
Received: 8 July 2014, Revised: 19 Aug. 2014, 13 Oct. 2014, Accepted: 14 Oct. 2014.
J.R. Ryoo: SeoulTech
E.S. Lee: WITHROBOT Co. Ltd.
T.-Y. Doh: Hanbat National University
※ 이 연구는 서울과학기술대학교 교내연구비의 지원으로 수행되었습니다(2014-0328).

 2014 Dec.: 9(6) 335-344
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2014.9.6.335

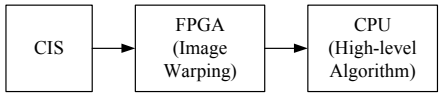


그림 1. 하드웨어 영상 워핑
Fig. 1 Hardware image warping

을 주로 다루었으며, 실시간 구현과 관련해서는 연산량 및 실행 시간을 분석하는 등의 연구가 진행되었다[5].

한편, 영상 워핑의 구현과 관련하여 FPGA 등의 하드웨어를 이용한 실시간 처리에 대한 연구도 진행되고 있다. 렌즈 왜곡 보정[6]과 영상 정렬[7]에 적용하기 위한 FPGA 기반 하드웨어 구조가 연구되었고, 하드웨어의 특징인 병렬 처리 구조를 적용한 방식[8]이 제시되었으며, 모의실험을 통하여 세부 구현 방법에 따른 성능 비교 연구가 진행되었다[9]. 이들 연구에서는 메모리 디바이스의 속도 한계를 극복하고자 영상 캐시를 적용[10]하기도 하였으나 시스템이 지나치게 복잡해져서 구현이 어려운 단점이 있다. 또한, 영상 워핑 결과의 정량적 분석이 없어 구현된 시스템의 정확성을 확인할 방법이 없다.

본 논문에서는 FPGA를 활용하여 영상 워핑을 실시간으로 처리하는 하드웨어 구조를 제시한다. 그림 1에는 하드웨어 영상 워핑을 포함한 임베디드 비전 시스템을 나타내었다. 기존에는 CPU에서 소프트웨어에 의해 처리되던 영상 워핑을 CMOS 영상 센서(CMOS Image Sensor, CIS)에서 CPU에 입력하기 전에 하드웨어에서 처리하므로 워핑에 필요한 CPU의 연산 부하를 줄여서 워핑 이후의 상위 레벨 알고리즘 처리를 용이하게 한다.

영상 워핑을 위한 하드웨어 구현의 편의성을 고려하여 역방향 매핑(backward mapping) 방식과 룩업테이블(Look-Up Table, LUT) 방식을 적용한다. 영상 워핑의 주요 연산은 입력 영상 데이터를 참조하기 위한 메모리 읽기/쓰기와 출력 영상 픽셀의 리샘플링을 위한 보간 연산이 주를 이룬다. 실시간 처리 성능의 만족을 위하여 LUT와 영상 버퍼 메모리의 액세스 특성을 분석하여 각각의 용도에 적합한 메모리 디바이스를 선정한다. 출력 영상의 픽셀 생성에 필요한 보간법으로는 4개의 이웃하는 입력 픽셀을 이용한 쌍일차 보간법(bilinear interpolation)을 사용하는데, FPGA 구현에 적합하도록 고정 소수점 연산(fixed-point operation)으로 이를 구현하는 방법을 기술한다. 또한, 보간 연산 블록에서의 타이밍 문제를 해결하기위하여 파이프라인 구조를

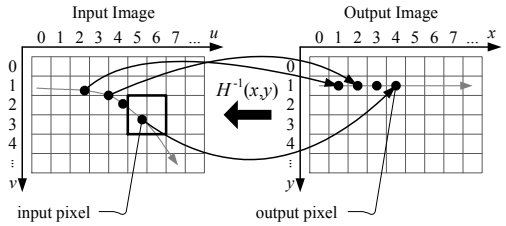


그림 2. 역방향 매핑을 활용한 영상 워핑
Fig. 2 Image warping using backward mapping

적용하고, 곱셈기의 수를 줄인 보간 연산 구조를 제시한다. 마지막으로 구현된 FPGA 기반 하드웨어에서의 영상 워핑 실험을 수행하고 그 결과를 제시한다. 워핑 결과의 정확성을 확인하기 위하여 테스트용 합성 영상을 PC 소프트웨어와 FPGA에서 각각 워핑한 후 결과 영상을 픽셀 단위로 비교한다. 또한, 실제 CIS의 영상을 워핑한 결과와 렌즈 왜곡 보정에 활용한 결과를 제시하여 구현된 영상 워핑 하드웨어의 효용성을 확인한다.

II. LUT 기반 영상 워핑

1. 픽셀 좌표 변환 및 보간

영상 워핑은 영상의 픽셀 좌표를 변환하여 영상을 기하학적으로 변형하는 영상처리 기법으로 (1)과 같은 입력 영상과 출력 영상의 픽셀 좌표 변환이 정의되어야 한다. 여기서 H는 입력 영상 좌표 (u, v)를 출력 영상 좌표 (x, y)로 변환하는 좌표변환 함수이다.

$$(x, y) = H(u, v) \tag{1}$$

(1)을 활용하여 영상 워핑을 구현하는 순방향 매핑(forward mapping) 방식과 달리 (2)의 역변환 관계를 활용하는 역방향 매핑(backward mapping) 방식이 보다 널리 활용된다[11].

$$(u, v) = H^{-1}(x, y) \tag{2}$$

그림 2에는 역방향 매핑 방식의 영상 워핑을 나타내었다. 출력 영상 픽셀을 순차적으로 생성하기 위한 입력 영상의 픽셀 좌표를 (2)를 활용하여 구하는데, 그 결과 입력 영상의 픽셀 정보에 대한 참조는 비순차적으로 발생한다.

(2)를 활용하여 (u, v)를 구하는 방법으로는 (2)

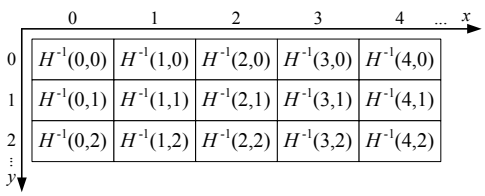


그림 3. 역방향 매핑 LUT
Fig. 3 LUT for backward mapping

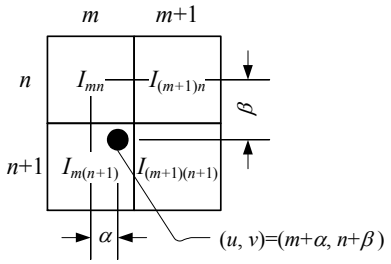


그림 4. 쌍일차 보간
Fig. 4 Bilinear interpolation

를 온라인에서 연산하는 방식과 오프라인에서 미리 연산하여 LUT 형태로 적용하는 방식이 있다. LUT 방식은 테이블 저장을 위한 메모리 공간이 필요한 대신 온라인에서의 연산량이 적고, 수식 형태의 함수로 표현이 어려운 좌표변환도 처리가 가능하며 연산량이 H^{-1}의 특성과 무관하게 일정하므로 실시간 구현에 적합하다. 그림 3에는 출력 영상의 픽셀에 대응하는 입력 영상의 좌표로 구성된 역방향 매핑 LUT의 내용을 나타내었다. 본 논문에서는 LUT 기반 영상 워핑의 실시간 구현을 다루며 역방향 매핑 LUT의 결정은 선행 연구에서 많이 다루어졌으므로 미리 결정되어 주어짐을 가정한다.

한편, 역방향 매핑에 활용되는 (2)에서 (x, y)는 정수인 반면 (u, v)는 정수가 아니므로 주변의 픽셀들을 활용한 쌍일차 보간을 일반적으로 적용한다[11]. 역방향 매핑에 의한 입력 영상 좌표 (u, v)는 다음과 같이 정수부 m과 n, 그리고 소수부 α와 β로 표현된다.

$$(u,\ v) = H^{-1}(x,\ y) = (m+\alpha,\ n+\beta) \tag{3}$$

여기서 m과 n은 정수이고 $0 \leqq \alpha < 1$과 $0 \leqq \beta < 1$을 만족한다. 그림 4에는 쌍일차 보간에 반영되는 입력 영상 픽셀을 나타내었으며, I_{mn}은 (m, n) 좌표의 픽셀 데이터를 의미한다. 4개의 픽셀 데이터로부터 출력 픽셀의 데이터 I_{out}은 다음과 같이 결정된다.

$$\begin{aligned} I_{out} &= I_{mn}(1-\alpha)(1-\beta) + I_{(m+1)n}\alpha(1-\beta) \\ &\quad + I_{m(n+1)}(1-\alpha)\beta + I_{(m+1)(n+1)}\alpha\beta \\ &= [1-\beta \ \ \beta]\begin{bmatrix} I_{mn} & I_{(m+1)n} \\ I_{m(n+1)} & I_{(m+1)(n+1)} \end{bmatrix}\begin{bmatrix} 1-\alpha \\ \alpha \end{bmatrix} \end{aligned} \tag{4}$$

2. 실시간 구현을 위한 주요 연산 분석

LUT 기반 영상 워핑은 LUT와 영상 데이터를 메모리에서 읽는 동작과 보간 연산을 위한 덧셈 및 곱셈 연산 등으로 구성된다. 따라서, 실시간 성능의 만족을 위해서는 필요한 데이터를 제한된 시간 내에 메모리에서 읽어야 한다. 이러한 제한 조건의 만족을 위해서 LUT와 영상 정보를 저장하는 메모리 디바이스는 영상 워핑에서의 메모리 액세스 특성을 분석하여 그 특성을 만족하는 소자로 선정하여야 한다. FPGA 내부의 램 블록은 충분한 성능을 나타낼 수는 있으나 통상적으로 충분한 용량을 확보하기에 어려움이 있으므로 타이밍 문제를 해결하기 위한 FIFO(First-In First-Out) 버퍼 용도로만 활용하고 LUT와 영상 데이터 버퍼는 외부 메모리를 활용한다.

현실적으로 가장 널리 활용되는 메모리 디바이스로는 SDRAM(Synchronous Dynamic RAM)과 SSRAM(Synchronous Static RAM)이 대표적이며, 각 메모리 디바이스의 입출력 특성은 다음과 같다.

- SDRAM: 순차적 접근 속도는 매우 빠르지만 비순차적 접근에서는 시간 지연(latency)이 발생하여 속도가 느림.
- SSRAM: 저장 용량과 순차적 접근 속도는 SDRAM보다 다소 부족하지만 순차적 접근과 비순차적 접근에서 속도 차이가 크지 않음.

역방향 매핑 방식에서는 래스터(raster) 영상에서의 출력 픽셀[12]을 순차적으로 생성하기 위하여 입력 픽셀을 비순차적으로 참조한다. LUT 각 항목은 출력 필셀과 일대일 대응되므로 결국 LUT는 순차적으로 참조하는 반면 입력 영상 데이터는 비순차적으로 참조한다. 특히, (4)의 쌍일차 보간에서는 4개의 입력 영상 픽셀을 참조하므로 영상 데이터의 비순차적 접근을 피할 수 없다. 따라서, 순차적으로 읽는 LUT는 순차적 접근 특성이 우수한 SDRAM에 저장하고, 비순차적으로 참조하는 실시간 영상 데이터는 비순차적 접근에서도 속도 저하가 심하지 않은 SSRAM에 저장한다.

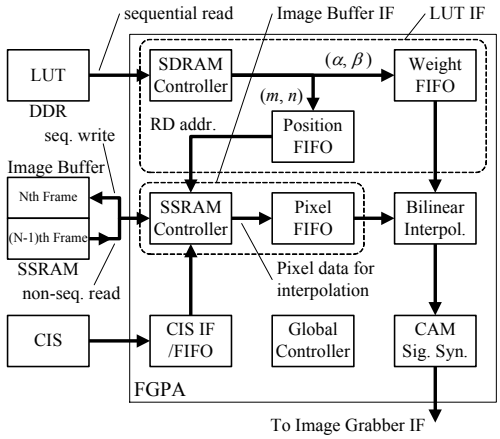


그림 5. LUT 기반 영상 워핑 FPGA 구조
Fig. 5 FPGA for LUT-based image warping

메모리 액세스 동작 외에 출력 영상의 픽셀 데이터를 생성하기 위한 보간 연산도 실시간 특성을 만족하여야 한다. 쌍일차 보간 연산 블록의 크기 및 사용 가능한 리소스를 고려하였을 때 고정 소수점 연산을 적용하며, 그럼에도 보간 연산의 결과에는 제한된 워드 길이(finite word length)로 인한 연산 오차가 없어야 한다. 쌍일차 보간 연산은 (4)에 나타내었듯이 4개 픽셀 데이터 각각에 가중치를 곱하여 더하는 연산인데, 사용되는 4개의 가중치 합은 항상 1이므로 가중치 평균 연산으로 볼 수 있으며, 그 결과 덧셈에 의한 오버플로우가 발생하지 않는다. 또한, 보간 연산에 사용되는 입력 픽셀 데이터는 양의 정수이고, 4개의 가중치도 모두 0보다 크거나 같으므로 쌍일차 보간에 사용되는 연산은 모두 unsigned 연산으로 구현이 가능하다. 본 논문에서는 앞에서 서술한 메모리 액세스 패턴과 사용 가능한 메모리 디바이스의 특성, 그리고 쌍일차 보간 연산의 특성을 바탕으로 실시간 영상 워핑을 위한 FPGA 기반 하드웨어 구조를 제안한다.

Ⅲ. LUT 기반 영상 워핑 하드웨어

1. 전체 영상 워핑 하드웨어 구조

그림 5에는 LUT 기반 실시간 영상 워핑 시스템의 전체 하드웨어 구조를 나타내었다. 앞에서 설명한대로 역방향 매핑 LUT는 순차적 액세스에 유리한 DDR SDRAM에 저장하고, 순차적으로 저장된 영상 데이터를 비순차적으로 참조하기 위한 영상 버퍼로는 SSRAM을 사용한다. 워핑된 출력 영상은

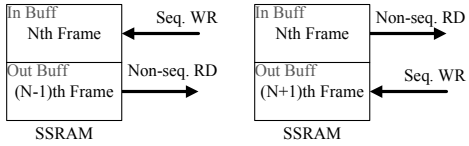


그림 6. 영상 버퍼의 더블 버퍼 구조
Fig. 6 Image buffer of a double buffer structure

CIS와 동일한 속도의 신호 포맷으로 출력하므로 일반적인 CIS 인터페이스 회로를 활용하여 출력 영상을 확인할 수 있다. 그림 5에 포함된 각 블록의 역할을 정리하면 다음과 같다.

- CIS IF: 센서 데이터를 SSRAM에 저장하기 위하여 CIS 출력 신호를 인터페이스.
- LUT IF: LUT를 읽기 위한 DDR SDRAM 컨트롤러와 타이밍 조절을 위한 FIFO 버퍼.
- Image Buffer IF: 영상 데이터 저장 및 읽기를 위한 SSRAM 컨트롤러와 FIFO 버퍼.
- Bilinear Interpolation: 쌍일차 보간 연산 블록 및 출력 타이밍 조절용 FIFO 버퍼.
- CAM Sig. Syn.(Camera Signal Synthesizer): 워핑된 영상의 출력을 위한 카메라 신호 합성기.
- Global Controller: 전체 시스템의 제어 및 타이밍 조절을 위한 전역 제어기.

그림 5의 구조에서 데이터의 주요 이동 경로를 설명하면 다음과 같다.

- 영상 센서 데이터를 SSRAM의 영상 버퍼에 순차적으로 저장.
- SDRAM에 저장된 LUT 항목을 읽어서 입력 영상 픽셀 좌표와 쌍일차 보간에 필요한 가중치로 분리.
- 쌍일차 보간에 필요한 4개의 입력 영상 픽셀 데이터를 SSRAM에서 비순차적으로 읽기.

영상 버퍼 메모리는 그림 6과 같은 더블 버퍼(double buffer) 구조를 적용하여 CIS에서 출력되는 N번째 프레임을 순차적으로 저장하는 입력 버퍼 영역과 순차적으로 저장된 직전의 (N-1)번째 프레임의 픽셀 데이터를 비순차적으로 참조하는 출력 버퍼 영역으로 나누어 사용한다. CIS에서의 N번째 프레임이 종료되면 입력 버퍼와 출력 버퍼가 반전된 후 입력 버퍼에 (N+1)번째 프레임이 순차적으로 저장되는 동시에 직전에 저장된 N번째 프레임 데이터는 비순차적으로 읽어서 보간에 사용된다. 여

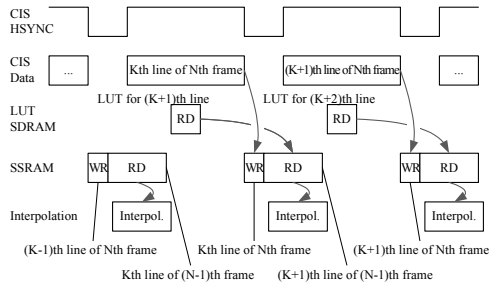


그림 7. 메모리 엑세스 타이밍도
Fig. 7 Timing diagram of memory access

기서 영상 버퍼인 SSRAM은 읽기와 쓰기가 하나의 메모리 포트로 진행되어야 하므로 SSRAM의 동작을 시분할하여 영상의 수평 라인 단위로 입력 또는 출력하도록 스케줄링한다. 그림 7에는 CIS의 출력 신호, SDRAM과 SSRAM의 메모리 액세스, 그리고 워핑 연산의 타이밍을 나타내었다. CIS의 HSYNC가 1에서 0으로 바뀐 직후 해당 수평라인의 픽셀 데이터를 SSRAM에 순차적으로 저장한다. SSRAM 저장 직후에 SDRAM에서 읽어둔 LUT를 이용하여 SSRAM에서 이전 프레임의 픽셀 데이터를 읽어서 보간 연산을 실행한다. 한 개의 수평라인 픽셀에 대한 보간 연산 후 그 다음 수평라인을 위한 LUT를 SDRAM에서 읽어온다.

2. 역방향 매핑 LUT 항목 포맷

그림 3에 나타낸 역방향 매핑 방식의 LUT 각 항목은 출력 픽셀 좌표 (x, y)에 대응하는 입력 픽셀 좌표 (u, v)를 의미한다. (u, v)는 (3)과 같이 쌍일차 보간에 적용될 입력 필셀을 지칭하기 위한 정수부 m과 n, 소수부 α와 β로 구분되는데, 그림 8에 나타낸 바와 같이 m과 n은 입력 영상이 저장된 SSRAM의 주소를 생성하는데 사용되고 α와 β는 (4)의 쌍일차 보간에 적용될 가중치를 의미한다. 여기서 $m, n \geq 0$과 $0 \leq \alpha, \beta < 1$의 조건을 만족하므로 m과 n은 16bit unsigned int 형태로 저장하고, α와 β는 16bit UQ0.16의 고정 소수점 포맷으로 저장된다. 여기서 UQ[i].[f]은 (i+f) bit의 unsigned Q 포맷 바이너리 수(binary number)로서 정수부와 소수부가 각각 i bit와 f bit의 길이를 가진다[13]. 출력 영상의 모든 픽셀에 대하여 그림 8의 LUT 항목을 미리 결정한 후 SDRAM에 저장하여 사용하는데, LUT는 시스템 초기화 과정에서 한번만 저장하면 그 이후의 영상 워핑 동작 중에는 읽기 동작만 수행한다.

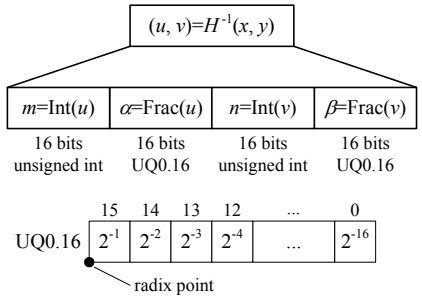


그림 8. 역방향 매핑 LUT 항목 포맷
Fig. 8 LUT element for backward mapping

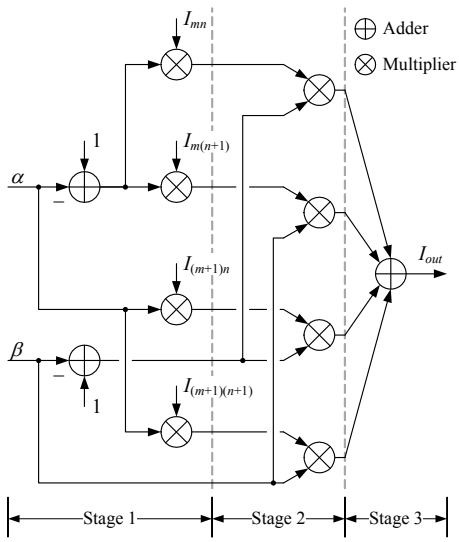


그림 9. 쌍일차 보간 연산
Fig. 9 Computation of bilinear interpolation

3. 쌍일차 보간을 위한 고정 소수점 연산

출력 영상의 픽셀 데이터 I_{out}은 4개의 입력 픽셀 데이터 I_{mn}, $I_{(m+1)n}$, $I_{m(n+1)}$, $I_{(m+1)(n+1)}$에 (4)의 쌍일차 보간을 적용하여 구한다. 그림 9에는 (4)의 쌍일차 보간 연산 블록의 구조를 나타내었는데, 고속 연산을 위하여 다수의 덧셈기(adder)와 곱셈기(multiplier)를 활용한 병렬 구조를 채택하였다. 그림 9에서 $(1-\alpha)$와 $(1-\beta)$의 가중치를 연산하는 덧셈기는 UQ0.16 포맷의 연산을 수행하므로 실질적으로는 다음과 같이 각 비트를 NOT 게이트한 결과와 같다.

$$1-\alpha \leftrightarrow 16'hFFFF-\alpha[15:0] = \sim \alpha[15:0] \quad (5)$$

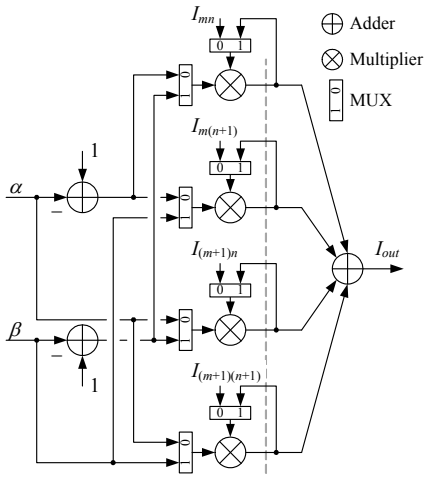


그림 10. 보간 연산의 곱셈기 수 감소
Fig. 10 Interpolation using reduced number of multipliers

8개의 곱셈기는 UQ0.16 포맷의 가중치와 UQ8.8 포맷의 픽셀 데이터를 곱하여 UQ8.24 포맷의 결과를 생성한 후 하위 16 비트를 제외한 UQ8.8 포맷의 결과로 출력한다. 입력 영상의 픽셀 데이터 I_{mn} 등은 0x00~0xFF의 8 비트 데이터로 입력되는 입력 영상의 픽셀 데이터를 상위 바이트에, 8 비트의 0을 하위 바이트에 채워서 UQ8.8 포맷을 형성한다. 최종 출력을 위하여 가중치가 적용된 4개의 값을 더하는 덧셈기의 경우 4개의 UQ8.8 포맷의 수를 더하여 UQ8.8 포맷으로 출력한다. 여기서 더해지는 4개의 수는 (6)과 같이 가중치의 합이 1인 가중치 평균(weighted average)의 의미를 가지므로 오버플로우가 발생하지 않는다.

$$(1-\alpha)(1-\beta)+\alpha(1-\beta)+(1-\alpha)\beta+\alpha\beta=1 \quad (6)$$

최종 결과인 UQ8.8 포맷에서 정수부인 상위 8 비트를 출력 영상의 픽셀 강도로 사용한다.

한편, 쌍일차 보간의 연산에서 발생하는 게이트 시간 지연의 문제는 3단의 파이프라인 구조를 적용하여 해결한다. 실제로 100MHz의 클럭을 사용하는 보간 연산에 파이프라인 구조를 적용하지 않았을 때는 타이밍 분석에서 setup time slack이 -6.6ns로 나타나지만 3단의 파이프라인을 적용하였을 때는 3.0ns로 분석되어 타이밍 조건을 만족한다. 최대 동작 가능 클럭 주파수는 파이프라인 미적용시 60.4MHz이고 적용시 141.9MHz로 나타나서 파이

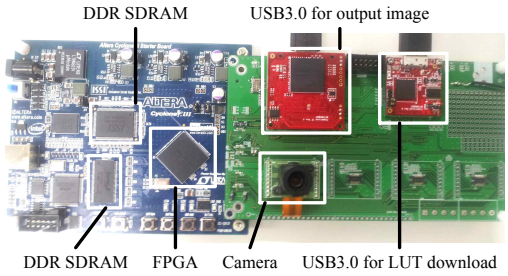


그림 11. FPGA 기반 영상 워핑 실험 환경
Fig. 11 Experimental setup for image warping

표 1. FPGA 활용률
Table 1. Resource usage

Block	Logic Cells (total 24,624)	Multipliers (total 132)	Block Mem (total 66)
CIS IF	320(1.3%)	0(0.0%)	4(6.1%)
LUT IF	6,085(24.7%)	0(0.0%)	15(22.7%)
Img. Buff. IF	346(1.4%)	0(0.0%)	4(6.1%)
Interpolation	688(2.8%)	8(6.1%)	8(12.1%)
Cam Sig.Syn.	194(0.8%)	0(0.0%)	0(0.0%)
Global Ctrl.	500(2.0%)	0(0.0%)	0(0.0%)
ETC	325(1.3%)	0(0.0%)	8(12.1%)
Total	8,458(34.3%)	8(6.1%)	39(59.1%)

프라인의 적용이 게이트 시간 지연 문제를 해소함을 확인할 수 있다. 파이프라인 적용에 따라 2 클럭 시간 지연이 발생하지만 전체 시스템 차원에서 문제되지 않는다.

그림 9의 보간 연산 구조에는 병렬 연산을 바탕으로 연산 속도를 높이기 위하여 8개의 곱셈기를 활용하였다. 보간 연산에는 픽셀당 8회의 곱셈 연산이 필요한데 출력 픽셀의 생성 속도를 충분히 높이기 위해서 각 곱셈기가 1회의 연산만 수행하는 구조이다. 하지만, 곱셈기가 2회의 곱셈 연산을 처리하도록 하면 4개의 곱셈기로 보간 연산을 처리할 수 있다. 그림 10에는 4개의 곱셈기를 활용한 쌍일차 보간 연산 구조를 나타내었다. 곱셈기는 100MHz로 동작하지만 매 클럭마다 MUX를 0과 1로 선택하여 각각 그림 9의 1단계와 2단계 파이프라인을 연산한다. 따라서, 출력 I_{out}의 갱신은 50MHz로 이루어지며, 가중치 α와 β, 입력 픽셀 데이터 I_{mn}, $I_{(m+1)n}$, $I_{m(n+1)}$, $I_{(m+1)(n+1)}$도 50MHz로 입력된다. 그림 10의 연산 구조에서의 setup time slack은 MUX의 사용으로 인하여 2.6ns로 그

림 9의 3.0ns에 비하여 조금 낮아지지만 타이밍 조건을 위배하지 않는 특성을 나타낸다.

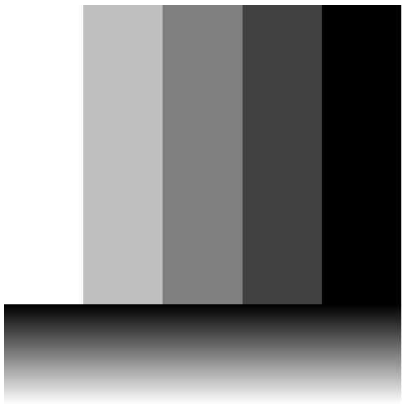
Ⅳ. 실험 및 결과

1. FPGA 구현 및 실험 환경

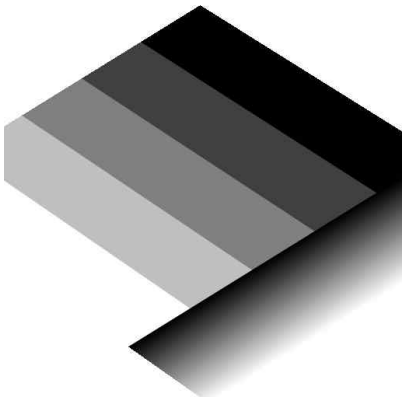
그림 11에는 구현된 FPGA 기반 영상 워핑 시스템의 실험 환경을 나타내었다. 영상 센서는 Micron 사의 M111 센서를 사용하는데, 센서의 출력은 640x512 해상도의 컬러 영상이지만 문제를 단순화하는 차원에서 512x512 해상도의 30 fps(frames per second) 흑백 영상으로 활용한다. FPGA는 Altera 사의 Cyclone III 디바이스를 탑재한 EVM 보드에 영상 센서 및 USB3.0 인터페이스를 결합하여 사용한다. 영상 워핑의 RTL 코딩은 Verilog HDL을 활용하였으며[14], DDR SDRAM 컨트롤러와 내부 RAM 블록을 이용한 FIFO 버퍼 등의 구현은 Altera에서 제공하는 IP를 활용하였다. 표 1에는 곱셈기를 4개 사용한 그림 10의 보간 연산 구조를 적용한 경우의 FPGA 리소스 사용 정보를 요약하였다. 8개의 곱셈기를 사용한 그림 9의 경우는 보간 연산 블록의 로직셀(logic cell)이 595로 감소하는 대신에 곱셈기 블록을 16개 사용한다.

LUT 저장용 메모리는 150MHz 16bit 데이터 버스의 DDR SDRAM을 사용하는데, LUT는 각 픽셀당 8바이트이므로 총 2MB를 사용한다. 영상 데이터용 버퍼 메모리는 100MHz 32bit 데이터 버스의 SSRAM을 더블 버퍼 형태로 사용하며, 영상 한 프레임은 0.25MB이므로 총 0.5MB를 사용한다. USB3.0 인터페이스는 각각 출력 영상 확인 및 LUT 다운로드의 목적으로 활용하는데, LUT는 영상 워핑 중에는 읽기 동작만 반복하므로 시스템 초기화 과정 중에 한번만 다운로드한다.

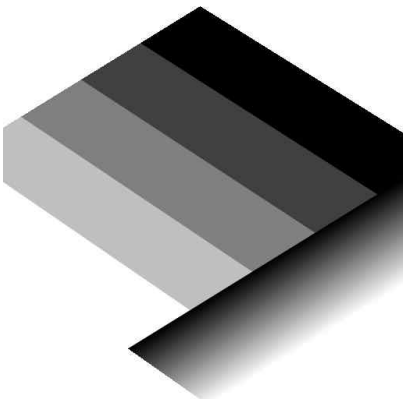
구현된 영상 워핑 시스템에서는 메모리 접근과 보간 연산이 동시에 실행되는데, 상대적으로 많은 시간이 소요되는 메모리 접근을 기준으로 유휴 시간을 제외하면 영상 워핑에는 프레임당 13ms가 소요된다. 동일한 영상 워핑을 PC와 Cortex-M4 계열 임베디드 비전 시스템에 소프트웨어로 적용하였을 때 각각 프레임당 5ms와 110ms가 소요되었다. 비교 실험에 사용된 임베디드 비전 시스템에서는 실시간 동작이 불가능하지만, FPGA를 활용하면 실시간 성능을 확보할 수 있다. PC에서의 결과와 비교하면 PC 소프트웨어가 빠른 성능을 보이지만 이는 고성능 CPU와 메모리 디바이스에 의한 것이며,



(a) 입력 테스트 영상
(a) Input test image



(b) PC에서의 워핑 결과
(b) Image warped at PC



(c) FPGA에서의 워핑 결과
(c) Image warped at FPGA

그림 12. 테스트 영상 및 워핑 결과 영상
Fig. 12 Test image and warped images

FPGA에서 영상 워핑을 처리하면 CPU 연산 부하의 약 15% 정도를 줄이는 효과로 나타날 수 있다.

2. 테스트 영상을 활용한 하드웨어 검증

본 논문에서 구현한 영상 워핑 하드웨어의 정확성을 검증하기 위하여 영상 센서 대신 테스트 영상을 생성한 후 워핑한 결과를 분석한다. 우선 테스트 영상은 영상 센서와 동일하게 512x512 해상도의 흑백 영상으로 생성한다. 생성된 테스트 영상을 PC 소프트웨어와 구현된 FPGA에서 동일한 LUT로 각각 워핑하고 그 결과를 비교한다. 이 때, PC 소프트웨어는 보간 연산에 고정 소수점 연산과 부동 소수점 연산을 각각 적용하여 그로 인한 차이도 함께 비교한다.

그림 12에는 테스트 영상과 함께 임의의 LUT를 사용하여 고정 소수점 연산의 PC 소프트웨어와 FPGA에서 각각 워핑한 결과를 나타내었다. 그림 12(b)와 (c)의 두 워핑 영상을 비교하면 영상에서의 차이가 확인되지 않으며, 픽셀 단위로 모든 출력 데이터를 비교해보아도 정확히 일치하는 결과를 나타내므로 구현된 워핑 시스템의 정확성을 확인할 수 있다. 한편, 부동 소수점 연산의 PC 소프트웨어와 FPGA에서의 워핑 결과를 비교해보면 영상의 512x512 픽셀 중 16개의 픽셀에서 오차가 발생하였으나, 그 오차는 8bit 픽셀 데이터의 최하위 비트(LSB)에서만 확인되었다. 따라서, 오차의 원인은 고정 소수점 연산에서의 제한된 워드 길이 효과로 볼 수 있다.

CMOS 영상 센서에서는 픽셀의 감도 편차, 암전류(dark current), 아날로그 신호 처리 회로의 오차로 인한 고정 패턴 노이즈(fixed pattern noise), ADC의 변환 오차 등 다양한 오차 원인에 의하여 영상 센서에서 출력되는 픽셀 데이터의 하위 비트 정확성은 보장되지 않는다[15]. 이러한 센서 특성을 감안하여 통상적인 영상 처리에서는 픽셀 데이터의 하위 2~3 비트 오차는 허용 가능한 오차로 간주되며, 따라서 FPGA 영상 워핑에서의 고정 소수점 연산으로 인한 1 LSB 오차는 무시할 수 있는 수준이다.

3. CMOS 영상 센서의 영상을 활용한 워핑

실제 CMOS 영상 센서의 영상을 구현된 FPGA에서 워핑하여 그 결과를 제시한다. 출력 영상은 입력 영상과 동일한 512x512 해상도의 흑백 영상으로 초당 30장을 실시간으로 워핑한다. 그림 13에는

(a) 입력 영상
(a) Input image

(b) FPGA에서의 워핑 결과
(b) Image warped at FPGA

그림 13. CMOS 영상 센서의 영상 워핑
Fig. 13 Image warping of CMOS image sensor

워핑 전과 후의 영상을 나타내었는데, 여기서 사용된 LUT는 그림 12에 적용한 것과 동일한 변환 관계를 적용하였다.

실질적인 하드웨어 영상 워핑의 활용을 예시하기 위하여 실험에 사용된 카메라 모듈의 렌즈 왜곡을 보정하는 실험을 수행하였다. 그림 13(a)의 입력 영상을 보면 렌즈 왜곡으로 인하여 영상이 휘어진 것을 확인할 수 있다. 이를 보정하기 위하여 우선 카메라 캘리브레이션을 통하여 카메라 내부 파라미터를 추출하고[16], 그 결과로부터 왜곡 보정용 LUT를 생성한다. 이를 구현된 하드웨어에 적용하여 워핑한 결과를 그림 14에 나타내었다. 그림 13(a)의 영상에는 렌즈 왜곡으로 인한 직선의 휘어짐 현

그림 14. 렌즈 왜곡 보정에 적용 결과
Fig. 14 Application to lens distortion correction

상이 화면 외곽으로 갈수록 심하지만, 그림 14의 보정 후 영상에서는 렌즈 왜곡 현상이 보정되었다. 렌즈에 의한 왜곡은 카메라의 내부 파라미터로 한 번 결정된 왜곡 보정용 워핑은 카메라의 자세와 무관하게 항상 유효하다.

V. 결 론

본 논문에서는 FPGA를 이용한 실시간 영상 워핑 하드웨어 구현을 제시하였다. 하드웨어 영상 워핑에 유리한 LUT 방식과 역방향 매핑 방식을 활용하고, LUT와 영상 버퍼 메모리 액세스 특성을 고려하여 각각을 저장하는 메모리 디바이스를 선정하였다. 또한, 하드웨어 워핑 구현을 위하여 고정 소수점 연산을 활용한 쌍일차 보간을 수행하는 연산 구조를 제시하였다. 구현된 하드웨어 워핑 시스템의 동작을 검증하기 위하여 테스트용 합성 영상에 임의의 워핑 LUT를 적용하고, 그 결과를 PC에서 소프트웨어로 워핑한 결과와 비교하여 워핑 결과의 정확성을 검증하였다. 또한, 카메라 렌즈 왜곡을 보정하는 LUT를 적용하여 렌즈 왜곡을 실시간으로 보정하는 용도로 활용할 수 있음을 확인하였다.

본 논문에서는 문제의 단순화를 위하여 해상도를 구현에 편리하도록 조정하고, 컬러 정보를 없앤 흑백 영상을 사용하였다. 또한, FPGA 구현의 편의성을 위하여 LUT와 영상 버퍼 메모리를 별도로 사용하였다. 실용성을 확보하기 위해서는 영상의 해상도를 높이고 컬러 영상을 처리하며, 구현에 용이한 단순한 영상 데이터 캐시 구조를 개발하여 하나의 외부 메모리를 사용하는 구조로의 추후 연구가 필요하다.

References

[1] S.-M. Lee, “Fast laser triangular measurement system using ARM and FPGA,” IEMEK J. Embed. Sys. Appl., Vol. 8, No. 1, pp. 25-29, 2013 (in Korean).

[2] J. Park, S.-C. Byun, B.-U. Lee, “Lens distortion correction using ideal image coordinates,” IEEE Transactions on Consumer Electronics, Vol. 55, No. 3, pp. 987-991, 2009.

[3] Z. Chen, C. Wu, H.T. Tsui, “A new image rectification algorithm,” Pattern Recognition Letters, Vol. 24, No. 1-3, pp. 251-260, 2003.

[4] M. Brown, D.G. Lowe, “Automatic panoramic image stitching using invariant features,” International Journal of Computer Vision, Vol. 74, No. 1, pp. 59-73, 2007.

[5] W. Yu, “An embedded camera lens distortion correction method for mobile computing applications,” IEEE Transactions on Consumer Electronics, Vol. 49, No. 4, pp. 894-901, 2003.

[6] K.T. Gribbon, C.T. Johnston, D.G. Bailey, “A real-time FPGA implementation of a barrel distortion correction algorithm with bilinear interpolation,” Proc. of the Image and Vision Computing New Zealand 2003, pp. 408-413, 2003.

[7] C. Vancea, S. Nedevschi, “LUT-based image rectification module implemented in FPGA,” Proceedings of IEEE International Conference on Intelligent Computer Communication and Processing, pp. 147-154, 2007.

[8] S. Oh, G. Kim, “FPGA-based fast image warping with data-parallelization schemes,” IEEE Transactions on Consumer Electronics, Vol. 54, No. 4, pp. 2053-2059, 2008.

[9] A. Serguienko, *Evaluation of image warping algorithms for implementation in FPGA,* Master thesis, Linköpings universitet, Sweden, 2008.

[10] P. Greisen, S. Heinzle, M. Gross, A. P Burg, “An FPGA-based processing pipeline for high-definition stereo video,” Journal of Image and Video Processing, Vol. 2011, No. 18, pp. 1-13, 2011.

[11] P. Giacon, S. Saggin, G. Tommasi, M. Busti, “Implementing DSP Algorithms Using Spartan-3 FPGAs,” DPS Magazine, Issue 1, pp. 16-19, 2005.
[12] D.-J. Kim, Y.-S. Park, “An implementation of FPGA embedded system for real-time SONAR signal display using the triple buffering method,” IEMEK J. Embed. Sys. Appl., Vol. 9, No. 3, pp. 173-182, 2014 (in Korean).
[13] E.L. Oberstar, “Fixed-point representation & fractional math,” Tech. Report, Oberstar Consulting, 2007.
[14] M.D. Ciletti, *Advanced Digital Design with the Verilog HDL,* Prentice Hall, 2010.
[15] Y. Hwang, M. Song, “Design of a CMOS image sensor based on a 10-bit two-step single-slope ADC,” Journal of Semiconductor Technology and Science, Vol. 14, No. 2, pp. 246-251, 2014.
[16] J. Weng, P. Cohen, M. Herniou, “Camera calibration with distortion models and accuracy evaluation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 10, pp. 965-980, 1992.

저 자 소 개

류 정 래

1996년 한국과학기술원 전기및전자공학과 공학사.
1998년 한국과학기술원 전기및전자공학과 공학석사.
2004년 한국과학기술원 전기및전자공학과 공학박사.
2004년-2005년 삼성전자 책임연구원.
2005년-현재, 서울과학기술대학교 전기정보공학과 부교수.
관심분야: 서보 제어, 디지털 제어 시스템, 하드웨어 기반 영상 처리.
Email: jrryoo@seoultech.ac.kr

이 은 상

2011년 서울과학기술대학교 제어계측공학과 공학사.
2013년 서울과학기술대학교 제어계측공학과 공학석사.
2013년-현재, 위드로봇(주) 주임연구원.
관심분야: 영상 신호 인터페이스, FPGA 기반 디지털 제어.
Email: les4454@withrobot.com

도 태 용

1992년 경북대학교 전자공학과 공학사.
1994년 한국과학기술원 전기및전자공학과 공학석사.
1999년 한국과학기술원 전기및전자공학과 공학박사.
1997년-2001년 삼성전자 책임연구원.
2002년-현재, 국립한밭대학교 전자제어공학과 교수.
관심분야: 강인제어, 학습제어, 반복제어, 디지털 제어 시스템.
Email: dolerite@hanbat.ac.kr