Journal of Korea Multimedia Society Vol. 19, No. 2, February 2016(pp. 122-127)
http://dx.doi.org/10.9717/kmms.2016.19.2.122

SoC 시스템에서의 깊이 영상 분할을 위한 효율적인 설계 구성 방법

성지목[†], 김봉성[††], 강봉순[†††]

Efficient Implementation Method Of Depth Image Segmentation In SoC System

Jimok Sung[†], Bongsung Kim[††], Bongsoon Kang[†††]

ABSTRACT

This paper propose implementation method of SoC system for efficient depth image segmentation. SoC systems are combined platform in the form of the Software and Hardware IP. In order to perform effectively, the user to determine the operation of the configuration of each part. In this paper, we implemented a segmentation of depth images taken by the infrared sensor at APU of SoC system. The proposed method efficiently implements high performance and low power in SoC system. Proposed method that using software parts of SoC system is capable to use at several depth image processing systems.

Key words: SoC System, IR Sensor, Depth, Segmentation, Grassfire, Bandwidth

1. 서 론

거리 영상(Depth Image) 이란 TOF(Time-of-flight range)카메라나 Kinect와 같은 거리 측정 센서를 활용하여 촬영한 거리 정보를 포함한 영상을 의미한다. 각 객체의 거리 정보에 따라 표현된 영상은 기존의 컬러 영상보다 배경과 객체의 분리가 용이하고, 이를 통하여 기존 동작인식의 어려움들을 상당부분 극복할 수 있기 때문에 3차원 자세 추정(3D pose estimation), 스켈레톤 등의 모션인식 기술 영역에서 많이 사용되고 있다[1]. 거리 측정 센서를 활용한 영상처리 분야의 증가에 따라 사용 분야 및 목적에 맞춰 고해상도 및 넓은 색상영역을 지원하는 제품들이 출시되고 있다. 또한 컬러와 깊이 영상을 함께 활용한 DIBR(Depth-Image-Based-Rendering)등의 알고리즘에 대한 연구도 증가하는 추세이다. 이에 따라 각각의 입력을 활용한 신호처리 알고리즘도 더욱 복잡해지고, 처리해야 할 연산양도 증가하고 있는 추세이다. DSP(Digital Signal Processor)를 활용해서는 이러한 복잡한 고성능 신호처리를 처리 하는데 한계가 있기 때문에 최근에 HW(Hardware)언어인 HDL (Hardware Description Language)로 동작하는 FPGA (Field Programmable Gate Array)와 SW (Software) 언어로 동작하는 APU(Accelerated Processing Unit)가 결합된 형태인 SoC(System on a chip) 시스템을 활용한 영상처리가 많이 이루어지고

※ Corresponding Author : Bongsoon Kang, Address: (604-714) 37, Nakdong-daero 550beon-gil, Saha-gu, Busan, Korea, TEL : +82-051-200-7703 , TEL : +82-, FAX : +82-, E-mail : bongsoon@dau.ac.kr
Receipt date : Aug. 10, 2015, Revision date : Nov. 27, 2015
Approval date : Dec. 28, 2015
[†] Dept. of Electronic Eng., Dong-A University
(E-mail : garrincha42@gmail.com)
[††] Dept. of Electronic Eng., Dong-A University
(E-mail : kbs1439@gmail.com)
[†††] Dept. of Electronic Eng., Dong-A University
※ This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0009777)

있다[2]. 이러한 SoC 시스템의 특성상 HW와 SW언어의 특성 및 구현 하고자 하는 알고리즘의 특성에 따른 연산의 분배를 고려하지 않은 경우 최적의 성능을 얻을 수 없다.

특히 영상 분할(Segmentation)은 동작 및 제스처 인식[3-4], 깊이 맵 생성(Depth map generation)[5] 등의 알고리즘에서 필수적인 전처리 연산 과정이며 이를 연산하기 위해서는 메모리의 할당 및 접근 과정을 복잡하게 수행해야 하기에 HDL(Hardware Description Language)로 구현하는 데에 어려움이 따른다. 따라서 이를 효과적으로 수행하기 위한 연구들이 지속적으로 이루어져왔다. 그 중 한 가지 방식은 PC에서 깊이 영상을 활용한 영상 분할 연산 후 해당 데이터를 활용하여 SoC시스템에서 영상처리를 하는 방식이다[6]. 해당 방식은 PC에서 영상 분할을 수행하기 때문에 알고리즘 구현이 쉽고 효율적이지만, PC와 SoC시스템간의 데이터 교환을 담당하는 USB의 Data path를 통과하는 데이터가 증가하는 문제점이 있다. 일반적으로 데이터 전송에 많이 사용되는 USB 2.0의 경우 이론상 480Mbps의 Bandwidth를 가지고, Data path를 통과하는 데이터 수의 증가는 처리 속도 및 기능의 저하를 야기하고, 특히 영상 촬영 장치의 발달로 인해 영상처리를 위해 사용되는 입력 영상의 크기가 고해상도나 넓은 색상 영역을 요구할수록 속도 및 기능저하의 문제점은 더욱 부각된다.

본 논문에서는 SoC 시스템을 활용한 깊이영상의 영상처리 과정에서 영상 분할을 효율적으로 수행하면서도 Data path를 통과하는 데이터양의 증가에 의해 발생하는 문제를 해결할 수 있는 구성 방식에 대하여 제안한다. 제안된 구성은 기존 PC에서 수행하던 영상 분할 연산을 SoC 시스템에서 GPIO(General Purpose Input Output)및 주변장치 제어 용도로 사용되던 APU에서 수행함으로써 PC, SoC 시스템 간의 데이터 교환을 위해 사용되는 USB에서의 Data path를 통과하는 데이터의 양을 감소시키는 방안을 제안한다. 이를 Xilinx사의 SoC 시스템인 Zynq-7000 board를 사용하여 구현하고 기존의 방식과의 USB 2.0에서 전송에 사용되는 Data path를 통과하는 데이터양의 비교를 통하여 구성의 효율성을 비교하였다.

2. 영상분할 알고리즘 특성에 따른 기존의 처리 방법

2.1 Grassfire방식의 영상 분할 방법

영상 분할이란 영상내의 각 픽셀이 가지고 있는 밝기, 색상 등의 특성을 활용하여 유사한 특성을 가지는 영역을 분할하는 작업으로 검사 및 인증, 모션 인식 등에서 널리 사용되어지고 있는 기본적인 영상처리 기법이다[7]. 영상 분할 방식으로는 Grassfire, Two-pass, ARTS(Advanced Real Time Segmentation), TLL(Twin Line Labeling)방식 등이 있는데 가장 많이 사용되는 방식은 Grassfire 방식이다.

Fig. 1은 Grassfire 방식의 알고리즘을 활용한 영상 분할과정의 순서도이다. 일반적인 영상 분할 과정은 객체 레이블링(Labeling) 과정에서 해당 픽셀의 연산 처리 여부를 확인하기 위한 재탐색 과정을 필요로 한다. 반면 Grassfire 방식의 알고리즘은 스택메모리(Stack Memory)를 활용하여 메모리에 저장된 처리해야 할 좌표에 자유롭게 접근이 가능하다. 이를 통하여 재탐색 과정을 필요로 하지 않는 효율적인 알고리즘 구성이 가능하다는 장점이 있다[8]. 하지만 호출을 대기하고 있는 메모리는 제한적일 수밖에 없고, 이로 인해 메모리 부족 등의 문제가 발생할 수 있기 때문에 이에 주의하여야 한다는 단점이 있다.

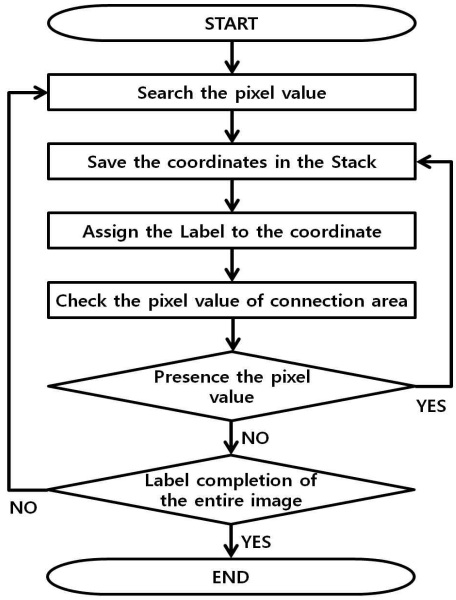


Fig. 1. Flowchart of grassfire segmentation algorithm.

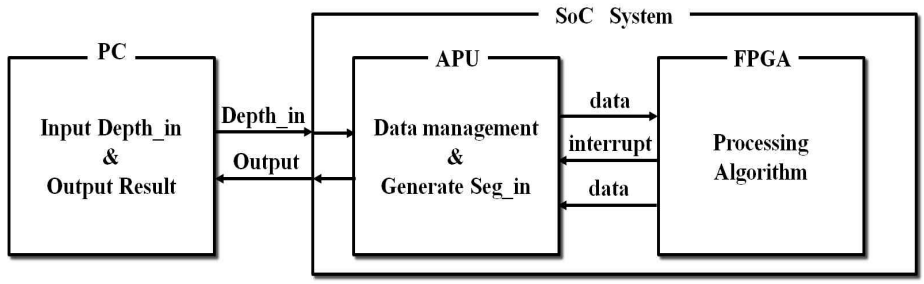


Fig. 2. Data flow of proposed implementation of SoC system.

2.2 SoC를 활용한 기존의 영상 분할 방법

SoC를 활용한 기존의 영상 분할 방법은 PC에서 분할된 영상을 SoC 시스템으로 전송하는 방식이다. 이러한 형식의 영상 분할은 SoC 시스템 내에서 HDL로 작성된 영상처리 알고리즘을 수행하기 위한 전처리 과정으로서, HDL로 구현하기 힘든 영상 분할을 PC에서 우선 수행하는 방식이다.

PC에서 영상 분할 된 데이터를 SoC 시스템으로 전송하고 SoC의 APU가 GPIO 및 주변장치 제어 기능을 수행하며 외부에서 깊이 영상 및 분할 영상을 입력 받고, 이를 FPGA에서 처리할 수 있도록 각 데이터를 알맞게 처리하고 전송해주는 역할을 수행한다. FPGA에서는 영상처리가 끝난 데이터는 다시 APU로 전송되고, APU에서는 출력 결과를 PC로 전송하도록 구성되어있다[6].

이러한 구성을 활용할 경우 PC에서 영상 분할을 수행하기 때문에 SoC 시스템으로 데이터를 주고받는 과정에서 깊이 영상과 함께 분할영상도 전송해야 하고, 이로 인해 USB전송에 요구되는 Data path를 통과하는 데이터양이 증가하게 된다. USB의 Bandwidth는 고정되어 있으므로 입력 영상이 고해상도나 넓은 색상 영역을 사용할 경우 데이터 경로를 지나가는 데이터의 수의 증가로 인해 속도 및 기능이 저하되는 문제점이 발생할 수 있다.

3. 제안한 방법

본 논문에서는 SoC 시스템을 활용하여 영상처리 과정에서 영상 분할을 효율적으로 처리하기 위한 구성 방법을 제안한다.

$$Total\,Transfer\,Data = Number_{Buffer} \times Size_{InputImage} \times Bits_{Buffer} \quad (1)$$

수식 (1)은 Segmentation을 위해 PC와 SoC 시스템간 전송되어야 하는 전체 데이터의 양을 나타 낸 식이다. $Number_{Buffer}$는 전송되는 Buffer의 개수, $Size_{InputImage}$는 입력 영상의 크기, $Bits_{Buffer}$는 한 픽셀당 Bit의 크기를 의미한다. 기존의 방법은 깊이 영상, 객체 분할 영상, 출력영상의 총 3개의 Buffer에 대한 데이터 전송이 이루어 져야 한다. 제안한 방식은 기존 PC에서 처리하던 영상 분할 과정을 SoC 시스템의 APU에서 처리함으로써 전송되는 버퍼를 줄임으로써 PC와 SoC 시스템간 전송되는 데이터를 줄일 수 있다.

Fig. 2는 제안한 방식의 데이터의 흐름을 나타 낸 것이다. 기존의 방식과 달리 제안된 방식의 경우 SoC 시스템 내부에서 깊이 영상을 활용하여 분할 영상 생성을 수행하므로 PC와 SoC 시스템 사이에는 깊이 영상 및 출력 결과에 대한 입출력 데이터 전송만을 수행한다.

Fig. 3는 PC와 SoC 시스템의 데이터 흐름에 따른 전체적인 구성을 나타 낸 것이다. Fig. 4는 Fig. 3의

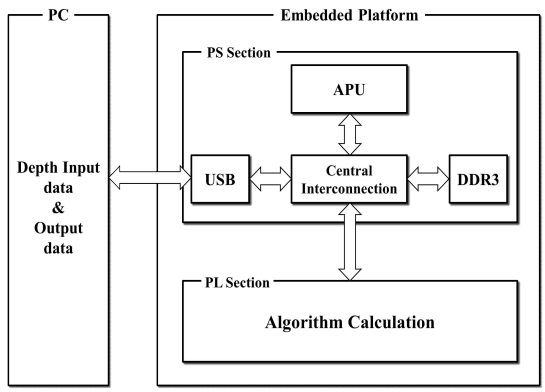


Fig. 3. Structure of proposed implementation of SoC system.

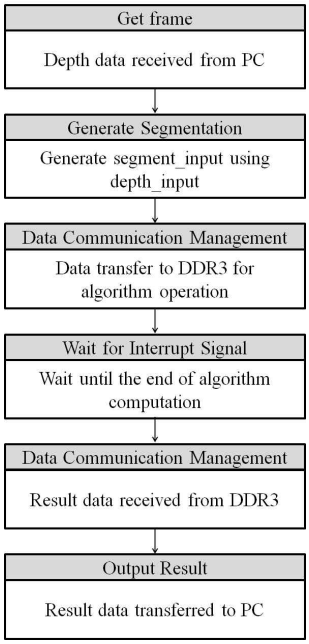


Fig. 4. Flowchart of APU operation.

APU(Accelerated Processing Unit)에서 Segmentation을 수행하는 과정을 나타낸 순서도이다. PC를 통해 입력된 Depth Input data는 USB 를 통하여 SoC system에 입력되고 Central Interconnection을 통하여 DDR3 메모리에 저장된다. PS의 APU는 DDR3의 Depth data를 이용하여 영상 분할연산을 수행하고 그 결과를 다시 DDR3에 저장한다. FPGA는 Central Interconnection을 통하여 DDR3에 접근하여 해당 Segmentation 결과를 이용하여 추가적인 영상처리 알고리즘 연산을 수행하고 연산된 결과를 다시 DDR3에 저장한다. 알고리즘이 완료된 결과는 다시 PC로 전송되어 결과 영상을 확인 가능하다.

해당 구성을 이용하면 기존 PC에서 생성하여 SoC 시스템으로 전송하던 객체 분할 영상을 SoC 시스템 내부에서 생성함으로써 PC와 SoC 시스템 간 데이터 전송을 위해 요구되는 USB에서의 Data path를 통과하는 데이터를 줄일 수 있다는 장점이 있다. 또한 Grassfire 방식의 객체 분할 방식의 특성을 고려하여 이를 HW에서 HDL이 아닌 SW의 APU에서 수행함으로써 연산의 효율성을 높일 수 있다.

4. 실험 결과

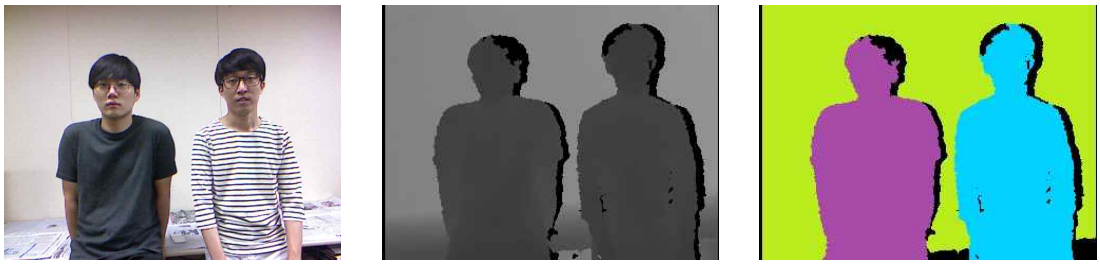
설계된 SoC 시스템의 Xilinx사의 Zynq-7000 board에서 설계하고 동작을 검증하였다. 시뮬레이션에 사용한 영상은 Infrared Sensor인 Kinect를 이용하여 촬영한 깊이 영상이다.

Fig. 5는 입력된 RGB영상과 해당 영상의 깊이 영상, APU를 통하여 분할한 결과를 나타낸 것이다. 시뮬레이션에 사용한 영상은 Fig. 5의 (a)는 입력 영상의 RGB 영상, Fig. 5의 (b)는 각 객체를 거리정보에 따라 Gray Scale로 표현한 깊이 영상이다. Fig. 5의 (C)는 APU에서 Fig. 3의 (b)를 이용하여 영상 분할을 실행하고 각각의 객체에 대하여 서로 다른 색으로 표현한 결과 영상이다.

제안된 방법과 기존 방법으로 영상분할을 수행하였을 경우 PC와 SoC 시스템 간 전송에 요구되는 데이터 전송량의 차이를 비교해보았다.

$$TransferData(bps) = Size_{InputImage} \times Bits_{Buffer} \quad (2)$$

식 (2)은 데이터를 저장 하는 각 버퍼별 전송되는 데이터의 양을 의미한다. $Size_{InputImage}$는 입력 영상 한 Frame의 크기를 의미하고 $Bits_{Buffer}$는 버퍼의 한 픽셀 당 데이터의 크기를 의미한다. $Size_{InputImage}$는



(a) (b) (c)

Fig. 5. Input and output result image. (a) Color image, (b) Depth image, and (c) Segmentation image.

Table 1. Compare of Total Data Transmission Rate of Original with Proposed Method

		Buffer Set	Bytes per Frame	Transfer Data per Frame (ms)
Original Method	Input	depth	153600	8.653
		segment	153600	
	Output	output	153600	
Proposed Method	Input	depth	153600	5.769
	Output	output	153600	

*본 연구는 IDEC의 EDA Tool을 지원하여 수행하였음.

입력 영상 한 Frame의 크기를 의미한다. *TransferData*는 한 Frame에 전송되는 개별 버퍼의 데이터의 양을 의미한다. PC와 Zynq보드 간 데이터 전송에 사용되는 USB 2.0의 경우 High Speed에서 125us동안 1개의 Micro Frame에 최대 6656Byte의 전송이 가능하다. 따라서 데이터의 전송 시간은 다음 식 (3)과 같이 정의할 수 있다.

$$Time_{TransferData} = \frac{TransferData \times 125us}{6656}(Bytes/us) \quad (3)$$

식 (3)에서 $Time_{TransferData}$는 한 Frame에 전송되는 데이터인 *TransferData*의 양에 따라 전송하는데 걸리는 시간을 의미한다. 식(2)과 식(3)를 활용하여 한 Frame의 입출력에 소모되는 시간을 비교해 보았다.

Table 1은 기존의 방법과 본 논문에서 제안한 방식을 활용했을 경우 한 Frame의 입출력 데이터를 전송하는데 걸리는 시간을 비교한 것이다. 입출력에 사용된 Frame의 크기는 320×240이다. 사용한 샘플의 Depth와 Segment의 Buffer Set은 각각 14bit, 9bit이고 출력 Output의 Buffer Set은 14bit이지만 Zynq에서 구성된 Interface 구조에 따라 각각 16bit로 정의하였다. 계산 결과 기존의 방법은 한 Frame을 전송하는데 8.654ms가 걸리지만 제안된 방식은 5.769 ms로 시간을 단축시킬 수 있다는 것을 확인하였다. 이를 통하여 USB의 고정된 Bandwidth내에서 데이터양을 감소시킴으로써 효율적인 데이터 전송을 수행할 수 있다는 것을 확인할 수 있다.

해당 수식 및 수치는 USB 2.0을 활용하여 입출력할 수 있는 최대 속력 및 속도의 이론적인 수치이므로 실제 동작 결과는 계산 결과보다 저하될 수 있지만, 기존의 방식과 비교하였을 때 제안된 방식을 활용할 경우 전송 속도가 증가함을 확인할 수 있다.

5. 결 론

본 논문에서는 SoC 시스템기반의 플랫폼에서 깊이 영상의 영상 분할 처리 과정을 효율적으로 처리하기 위한 방법을 제안 하고 이를 구현하여 Data path를 통과하는 데이터양의 변화의 확인을 통해 효율성을 검증하였다. 제안된 방법은 기존 SoC 시스템에서 연산 효율 증가를 위하여 HDL이 아닌 PC에서 수행되던 영상 분할 처리를 SoC 시스템의 SW인 APU에서 수행하는 방식이다. 이를 통하여 PC와 SoC 시스템간의 데이터 전송에 요구되는 데이터양을 감소시키고, SoC 시스템의 특성을 고려한 APU에서의 영상 분할을 통하여 SoC 시스템에서의 연산 효율을 높일 수 있는 방식을 제안하였다. 제안한 시스템 구성을 Xilinx사의 Zynq-7000 Board에서 검증하여 USB 2.0에서 전송되는 Data path에서의 데이터 감소를 확인하였다. 이를 통하여 SoC 시스템에서 영상 분할을 SW에서 수행함으로써 연산 및 구성의 효율성을 높일 수 있다는 것을 확인하였다.

향후 추가적인 영상 처리 알고리즘의 구현에 본 논문에서 제안한 영상 분할 처리 구성 방식을 적용함으로써 효율적인 저 전력, 고효율 시스템 구성에 도움이 될 것으로 예상한다.

REFERENCE

[1] J.Y. Chang, M.W. Ryu, and S.C. Park, “Technology Trends of Range Imge Based Gesture Recognition,” *Electronics and Telecommunications Trends*, Vol. 29, No. 1, 2014.

[2] B.G. Lim and M.H Kang, “HW/SW Co-design For an Ultrasonic Signal Processing System Using Zynq SoC,” *Journal of The Institute of*

Electronics and Information Engineers, Vol. 51, No. 8, pp. 148-155, 2014.

[3] H.J. You and T.Y. Kim, "A Background Segmentation and Feature Point Extraction Method of Human Motion Recognition," *Journal of Korea Game Society*, Vol. 11, No. 2, pp. 161-166, 2011.

[4] E.J. Han, H. Kang, and K.C. Jung, "Recognition-Based Gesture Spotting for Video Game Interface," *Journal of Korea Multimedia Society*, Vol. 8, No. 9, pp. 1177-1186, 2005.

[5] S.D. Kim, J.W. Ahn, Y.H. Seo, D.W. Kim, and J.S. Kim, "Depth Map Generation Method Using Segmentation and Motion Information," *Proceeding of 2010 Korean Society of Broadcast Engineers Summer Conference*, pp. 116-118, 2010.

[6] K.H. Jang, G.J. Kim, H.S. Cho, and B.S. Kang, "Foreground Segmentation Using Morphological Operator and Histogram Analysis for Indoor Applications," *IEICE TRANSACTION on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E98-A, No. 9, pp. 1998-2003, 2015.

[7] M.Y. Choong, W.Y. Kow, Y.K. Choong, L. Angeline, and K.T.K. Teo, "Image Segment via Normalized Cuts and Clustering Algorithm," *Proceeding of International Conference on Control System, Computing and Engineering, IEEE*, pp. 430-435, 2012.

[8] BLOB Analysis (Introduction to Video and Image Processing) Part 1, http://what-when-how.com/introduction-to-video-and-image-processing/blob-analysis-introduction-to-video-and-image-processing-part-1/ (accessed May., 17, 2015).

성 지 목

2014년 2월 동아대학교 전자공학과(공학사)
2014년 3월~현재 동아대학교 전자공학과 석사과정
관심분야 : 영상처리, 상황인식 , 물체추적

김 봉 성

2014년 2월 동아대학교 전자공학과(공학사)
2014년 3월~현재 동아대학교 전자공학과 석사과정
관심분야 : 영상 신호처리, 디지털 회로 설계

강 봉 순

1985년 연세대학교 전자공학과 공학사
1987년 미국 University of Pennsylvania 전기공학과 공학석사
1990년 미국 Drexel University 전기 및 컴퓨터 공학과 공학박사
1989년~1999년 삼성전자 반도체 수석연구원
1999년~현재 동아대학교 전자공학과 교수
2006년~2011년 멀티미디어 연구센터 소장
2006년~2013년 2단계 BK21 사업팀장
2013년~현재 BK21 Plus 사업팀장
관심분야 : 영상신호처리, SoC설계 및 무선통신