JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 18, No. 10 : 2443~2450 Oct. 2014

3차원 점군데이터의 깊이 영상 변환 방법 및 하드웨어 구현

장경훈 · 조기쁨 · 김근준 · 강봉순*

Conversion Method of 3D Point Cloud to Depth Image and Its Hardware Implementation

Kyounghoon Jang · Gippeum Jo · Geun-Jun Kim · Bongsoon Kang*

Department of Electronic Engineering, Dong-A University, Busan 604-714, Korea

요 약

깊이 영상을 이용한 동작 인식 시스템에서는 효율적인 알고리즘 적용을 위하여 깊이 영상을 3차원 점군 데이터로 구성되는 실제 공간으로 변환하여 알고리즘을 적용한 후 투영공간으로 변환하여 출력한다. 하지만 변환 과정 중 반올림 오차와 적용되는 알고리즘에 의한 데이터 손실이 발생하게 된다. 본 논문에서는 3차원 점군 데이터에서 깊이 영상으로의 변환 시 반올림 오차와 영상의 크기 변화에 따른 데이터 손실이 발생하지 않는 효율적인 방법과 이를 하드웨어로 구현 하는 방법을 제안 하였다. 최종적으로 제안된 알고리즘은 OpenCV와 Window 프로그램을 사용하여 소프트웨어적으로 알고리즘을 검증하였고, Kinect를 사용하여 실시간으로 성능을 테스트하였다. 또한, Verilog-HDL을 사용하여 하드웨어 시스템을 설계하고, Xilinx Zynq-7000 FPGA 보드에 탑재하여 검증하였다.

ABSTRACT

In the motion recognition system using depth image, the depth image is converted to the real world formed 3D point cloud data for efficient algorithm apply. And then, output depth image is converted by the projective world after algorithm apply. However, when coordinate conversion, rounding error and data loss by applied algorithm are occurred. In this paper, when convert 3D point cloud data to depth image, we proposed efficient conversion method and its hardware implementation without rounding error and data loss according image size change. The proposed system make progress using the OpenCV and the window program, and we test a system using the Kinect in real time. In addition, designed using Verilog-HDL and verified through the Zynq-7000 FPGA Board of Xilinx.

키워드 : 동작인식, 깊이 영상, 3차원 점군 데이터, 좌표 변환, 홀 제거

Key word : Motion Recognition, Depth Image, 3D Point Cloud, Coordinate Conversion, Hole Filling

접수일자 : 2014. 07. 18 심사완료일자 : 2014. 08. 18 게재확정일자 : 2014. 09. 01

* **Corresponding Author** Bongsoon Kang(E-mail:bongsoon@dau.ac.kr, Tel:+82-51-200-7703)
Department of Electronic Engineering, Dong-A University, Busan 604-714, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2014.18.10.2443 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

사람의 동작인식에 관한 연구는 컴퓨터비전 (Computer vision)의 초기부터 수행되어 온 매우 중요한 분야이다. 특히 최근 동작인식 기술은 3D TV와 스마트 TV 등을 비롯한 엔터테인먼트 및 게임 시장에 폭넓게 채택되고 있다. 이에 따라 비접촉식 인터랙션으로 각종 전자기기를 제어하는 기술이 차세대 핵심 기술로 주목 받고 있다. 기존의 동작인식 기술은 2차원 컬러 영상을 기반으로 연구되어 왔다. 하지만 사람이 취할 수 있는 동작은 수학적으로 고차원이고, 사람이 움직이는 영역은 기본적으로 3차원 공간에 속하므로 2차원 영상으로의 투사(Projection) 과정에서 정보의 손실이 발생한다. 이로 인하여 배경과 전경의 분리, 객체 외양 정보의 왜곡 등으로 복잡하고 다양한 환경에서 뛰어난 인식률과 성능을 기대하기 어렵다. 따라서 최근에는 거리 측정 센서 기술의 등장으로 3차원으로 구성된 깊이 영상 기반의 동작인식 기술들이 실용화되었고 활발하게 연구되고 있다[1, 2].

거리 측정 센서로부터 입력되는 거리 정보는 투영 공간(Projective world)으로 표현되어 2차원 배열에 각 픽셀의 거리 정보로 구성 된다[3]. 투영 공간으로 표현된 깊이 영상 (Depth image)은 거리 측정 센서의 화각(Field of View, FoV)에 의한 왜곡으로 인하여 실제 거리 값으로 표현이 불가능 하다. 따라서 투영 공간으로 표현된 깊이 영상을 이용하여 배경과 전경의 분리 또는 인체 모델링을 위한 스켈레톤 추출 등의 동작인식을 수행할 경우 시스템의 성능 저하가 불가피 하다. 따라서 거리 정보를 이용한 시스템에서는 입력되는 깊이 영상을 3차원 점군 데이터로 구성되는 실제 공간 (Real world) 으로 변환하여 동작인식을 위한 알고리즘을 수행한 후 투영공간으로 역변환 하여 결과 영상을 출력하는 것이 일반적이다. 깊이 영상의 3차원 점군 데이터로의 변환과 역변환은 거리 측정센서의 화각을 고려하여 수행 된다[2, 3].

깊이 영상에서 3차원 점군 데이터로의 변환과정은 입력 영상의 크기보다 좌표축이 큰 좌표계로 변환되어 데이터가 분포되기 때문에 데이터의 손실이 발생하지 않는다. 하지만, 3차원 점군 데이터에서 깊이 영상으로 역변환을 수행하는 경우에는 좌표축이 축소됨에 따라 픽셀 위치 연산 과정의 반올림 오차에 의한 데이터 손실이 발생하게 된다. 뿐만 아니라, 동작인식을 위한 알고리즘 처리 후 데이터가 변형 된 경우 역변환 과정에서의 데이터 손실이 불가피 하다. 특히, 센서의 위치와 각도에 의한 영상왜곡 보정 과정에서 평행이동(Translation), 척도변환(Scaling), 회전(Rotation), 층밀리기(Shearing) 등이 적용 되는 경우 영상의 크기가 변하게 되고 픽셀의 손실이 발생하여 홀(Hole)이 발생하게 된다[4]. 따라서 반올림 오차와 영상의 크기 변화에 따른 데이터 손실의 보정은 역변환 과정에서 필수적이다. 기존에는 영상 크기 변화에 따른 데이터 손실 방지를 위하여 센서의 위치와 각도에 의한 영상왜곡 보정 시 다대일 매칭을 이용하여 홀 생성을 사전에 방지하는 방법을 사용하였다. 이는 보정된 깊이 영상의 모든 픽셀 위치를 순차적으로 이동시켜가며 해당 픽셀의 깊이 정보들을 입력 깊이 정보들로 역으로 추적하여 채워나가는 방법이다[5]. 이에 따라 홀이 발생하지 않는다는 장점이 있는 반면, 하드웨어 구현 시 Memory Random Access가 이루어 져야 하므로 비효율적이다. 또한 3차원 점군 데이터로 재구성하여 실제 공간에서의 처리가 아닌 투영공간에서만 수행되기 때문에 동작인식을 위한 다양한 알고리즘의 적용이 어렵다.

본 논문에서는 3차원 점군 데이터에서 깊이 영상으로의 영상 변환 과정에서 반올림 오차와 영상의 크기 변화에 따른 데이터 손실이 발생하지 않는 효율적인 방법과 이를 하드웨어로 구현하는 방법을 제안한다. 입력되는 3차원 점군 데이터는 좌표변환 수식이 적용된 후 데이터 손실에 의한 홀 제거를 수행한 후 깊이 영상을 출력한다. 제안한 방법은 Microsoft Visual Studio 2010과 OpenCV를 이용한 윈도우 검증 프로그램을 통하여 성능을 검증하였다. 또한 Verilog-HDL을 이용하여 하드웨어로 구현하고 Field Programmable Gate Array (FPGA)와 Central Processing Unit (CPU)을 연동하여 검증 가능한 Zynq-7000 FPGA Board를 이용하여 실시간 처리를 확인 하였다.

본 논문의 Ⅱ장에서는 제안하는 3차원 점군 데이터의 깊이 영상 변환 방법 대하여 논하고 Ⅲ장에서는 제안한 알고리즘의 시뮬레이션 결과에 대하여 논하고 성능을 평가한다. Ⅳ장에서는 제안된 시스템의 하드웨어 설계에 관하여 서술하고 Zynq-7000FPGA Board를 이용한 합성 결과와 성능에 대하여 논한다. Ⅴ장에서는 결론을 맺는다.

Ⅱ. 3차원 점군데이터의 깊이 영상 변환

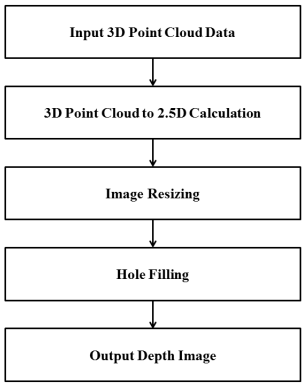


그림 1. 3차원 점군 데이터의 깊이 영상 변환 흐름도
Fig. 1 Flow Chart of 3D Point Cloud to Depth Image Conversion

그림 1은 제안한 방법의 흐름도이다. 입력되는 3차원 점군 데이터로 구성되는 실제 공간을 깊이 영상으로 표현 되는 투영 공간으로 변환하는 Inverse Coordinate Conversion, 영상 크기 조정을 위한 Resizing Image, 좌표 변환과 영상처리 알고리즘 적용 시 발생되는 홀을 제거하는 Hole Filling으로 구성된다.

$$p_x{}'(x,y)=\left(\frac{p_x(x,y)}{p_z(x,y)\times\tan\frac{f_h}{2}}+0.5\right)\times(w-1) \quad (1)$$

$$p_y{}'(x,y)=\left(\frac{p_y(x,y)}{p_z(x,y)\times\tan\frac{f_v}{2}}+0.5\right)\times(h-1) \quad (2)$$

$$p_z{}'(x,y)=p_z(x,y) \quad (3)$$

수식 (1)~(3)은 3차원 점군 데이터의 깊이 영상 변환 수식 이다. 수식 (1)은 깊이 영상의 x축 좌표 $p_x{}'(x,y)$, 수식 (2)는 y축 좌표 $p_y{}'(x,y)$, 수식 (3)은 픽셀 값 $p_z{}'(x,y)$이다. 수식 (1)~(3)의 $p_x(x,y)$는 3차원 점군 데이터의 x축 좌표, $p_y(x,y)$는 y축 좌표, $p_z(x,y)$는 z축 좌표를 나타낸다. w와 h는 각 각 입력 영상의 가로크기와 세로크기를 나타낸다. x, y는 입력 영상의 픽셀이고, f_h는 Horizontal FoV, f_v는 Vertical FoV 이다[3, 6].

Image Resizing 에서는 영상의 왜곡보정 과정과 좌표변환을 수행 하면서 발생되는 영상 크기 변화를 조정하기 위하여 영상 크기 조정을 수행한다. 깊이정보가 존재하는 픽셀 위치의 최대, 최소값을 계산하여 깊이영상의 중심을 찾고, 중심에서 입력 깊이 영상의 크기만큼 크기 조정을 수행 한다[6]. Hole Filling은 좌표 변환에 의한 반올림 오차와 영상 크기 조정에 의하여 생성된 홀을 주변 픽셀의 깊이 값을 활용하여 효과적으로 제거하는 기능이다. 투영 공간을 실제 공간으로 변환하여 3차원 점군 데이터로 재구성된 정보는 늘어난 스케일의 3차원 공간상에 형성되며, 모든 좌표 상에 존재하지 않고 드문드문 떨어져서 존재 한다[7]. 다음 과정인 왜곡보정을 수행하면서 좌표들이 모이거나 퍼지는 경우가 발생한다. 따라서 좌표 변환을 수행했을 때, 좌표들이 모인 경우에는 하나의 픽셀에 여러 개의 깊이정보가 채워지고, 퍼진 경우에는 퍼진 채로 픽셀에 깊이정보가 채워지므로 홀이 형성된다.

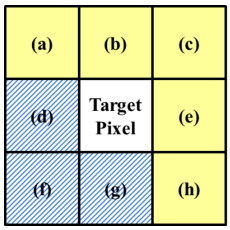


그림 2. 홀 제거를 위한 3x3 마스크
Fig. 2 3x3 Mask for Hole Filling

그림 2는 홀 제거를 위하여 적용한 3×3 마스크이다. 그림 2의 (a), (b), (c), (e), (h) 픽셀들을 유효 픽셀이라고 했을 때 다음과 같이 유효한 픽셀들에 대해서만 계산한 평균값과 가장 유사한 픽셀의 Depth값을 $TargetPixel$에 채워준다. 수식 (4)는 3×3 마스크내의 유효 픽셀들에 대한 평균값 $MaskValue$ 연산하기 위한 수식이다. 유효 픽셀들의 합 $VaildPixelSum$을 유효 픽셀의 개수 $VaildPixelCount$로 나누어 연산한다. 수식 (5)는 생성된 Hole을 제거하기 위한 $TargetPixel$을 연산하기 위한 수식이다. 3×3 마스크내의 유효 픽셀 $VaildPixel$과 $MaskValue$의 차이가 가장 작은 깊이 값으로 홀을 대체한다.

$$MaskValue=\frac{ValidPixelSum}{ValidPixelCount} \quad (4)$$

$$TargetPixel=\min(ValidPixel(i,j)-MaskValue) \quad (5)$$

Ⅲ. 시뮬레이션 결과

본 논문에서 제안한 시스템은 거리 측정 센서로부터 3차원 점군 데이터를 입력으로 하여 Microsoft Visual Studio 2010과 OpenCV를 이용한 검증 프로그램을 통하여 성능을 검증하였다.

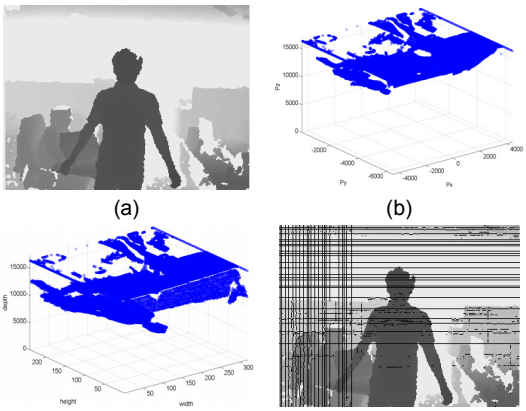


그림 3. 3차원 점군 데이터의 깊이 영상 변환 결과 영상 (a) 깊이 영상, (b) 입력 실제 공간 3차원 점군 데이터, (c) (b)를 변환한 투영 공간 좌표, (d) (c)의 깊이 영상
Fig. 3 Result Image of 3D Point Cloud to Depth Image (a) Depth Image, (b) Input Real World 3D Point Cloud, (c) Projective World Obtained by (b), (d) Depth Image of (c)

그림 3은 실제 공간인 3차원 점군 데이터를 투영 공간인 깊이영상으로 변환하는 성능을 평가하기 위한 영상이다. 그림 3의 (a)는 640×480 크기의 깊이영상이다. (b)는 (a)를 3차원 점군 데이터로 나타낸 입력 데이터로 p_x, p_y는 각각 x축 좌표, y축 좌표를, p_z는 깊이 값을 나타낸다. (c)는 (b) 에 좌표 변환 수식을 적용한 투영 공간 좌표를 나타낸 것이다. (b)의 p_x, p_y는 가로 크기와 세로 크기이고, p_z는 픽셀 값으로 거리 정보다. (d)는 (c)의 투영 공간 좌표를 깊이영상으로 나타낸 것으로 홀이 발생한 것을 알 수 있다.

그림 4는 Resizing Image의 성능을 평가하기 위한 영상이다. 그림 4의 (a)는 영상 왜곡 보정 알고리즘 적용 후 실제 공간 에서 투영 공간으로 변환된 결과 영상이다. 영상의 크기가 변화하여 732×543 크기로 생성된 영상이다. (b)는 본 논문에서 제안한 Resizing Image를 적용한 결과로 640×480 크기로 영상의 크기가 재조정된 것을 확인 가능하다.

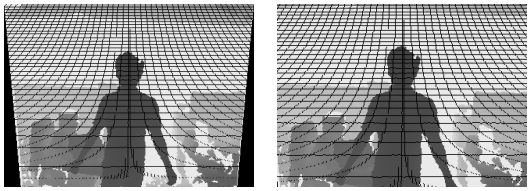


그림 4. Resizing Image 시뮬레이션 결과 이미지 (a) 원본이미지의 깊이영상, (b) (a)의 Resizing Image 결과
Fig. 4 Resizing Image Simulation Result Image (a) 2.5D data of Original Image, (b) Result Image using Resizing Image of (a)

그림 5는 마스크의 특성에 따른 Hole Filling 을 적용한 결과이다. 그림 5의 (a)와 (b)는 마스크 내의 픽셀들의 평균값으로 적용한 Hole Filling 영상이다. 그림 5의 (c)와 (d)는 마스크 내의 픽셀들의 평균값과 가장 가까운 픽셀의 값으로 적용한 Hole Filling 영상이다. 확대한 그림 (b)와 (d)를 비교해보면 평균값으로 Hole Filling을 적용했을 때는 객체의 경계 부분이 명확하지 않지만, 평균값과 가장 인접한 값으로 Hole Filling을 적용했을 때는 객체와 배경 사이의 경계가 뚜렷하게 구분된다.

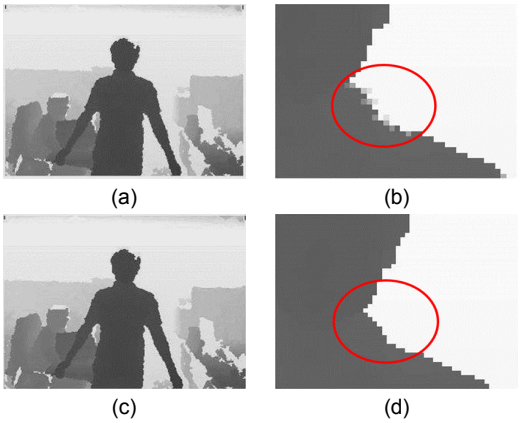
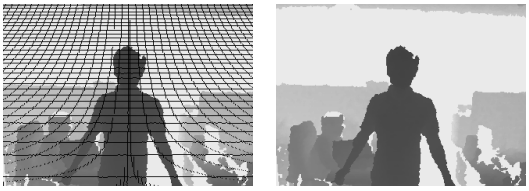


그림 5. 마스크 특성에 따른 홀 제거 적용 결과 비교 (a) 홀 제거 영상 #1, (b) (a)의 확대 영상, (c) 홀 제거 영상 #2, (d) (c)의 확대 영상
Fig. 5 Comparison of Hole Filling according mask characteristic (a) Hole Filling Image #1, (b) Magnification Image of(a), (c) Hole Filling Image #2, (d) Magnification Image of(c)



(a) (b)

그림 6. 홀 제거 시뮬레이션 결과 이미지 (a) Resizing Image, (b) (a)의 홀 제거 결과
Fig. 6 Hole Filling Simulation Result Image (a) Resizing Image, (b) Result Image using Hole Filling of(a)

그림 6은 Hole Filling 하는 방법의 성능을 평가하기 위한 영상이다. 실제 공간에서 투영 공간으로 변환하면서 영상의 홀이 발생하게 된다. 그림 6의 (a)는 Resizing Image까지 적용한 결과이고, (b)는 발생한 홀을 본 논문에서 제안한 3×3 마스크 연산을 이용하여 채워준 최종 출력 이미지이다.

Ⅳ. 하드웨어 구현

제안된 시스템의 하드웨어 구조는 그림 7과 같이 크게 inverse_conversion_calc, resizing_img, hole_filling의 3개의 블록으로 나누어진다. inverse_conversion_calc은 Inverse Coordinate Conversion 연산을 수행하는 블록이다.

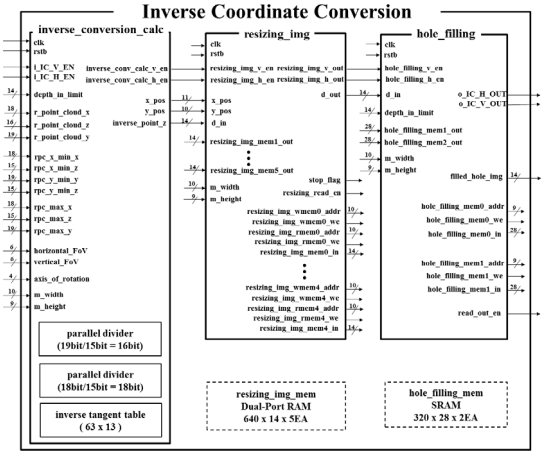


그림 7. Inverse Coordinate Conversion 블록 다이어그램
Fig. 7 Inverse Coordinate Conversion Block Diagram

resizing_img inverse_conversion_calc 적용한 후 스케일이 변한 깊이영상을 입력되는 크기와 동일한 깊이영상으로 만들어주는 블록이다. hole_filling 블록은 inverse_conversion_calc을 수행하면서 발생한 Hole을 채워주는 기능을 한다.

4.1. inverse_conversion_calc 블록 하드웨어 구현

3차원 점군 데이터 정보들을 입력받아 Inverse Coordinate Conversion 수식을 적용한 깊이영상을 출력한다. 많은 입력이 연속적으로 들어오기 때문에 Parallel Divider 2개와 inverse tangent table을 사용하여 실시간 처리를 가능하게 한다.

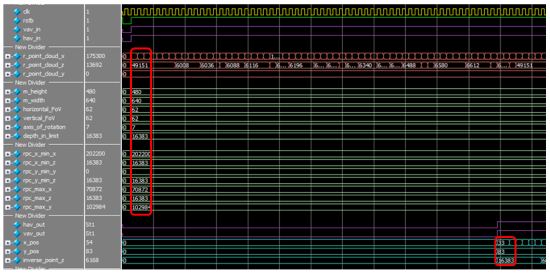


그림 8. inverse_conversion_calc 블록 시뮬레이션 파형
Fig. 8 Simulation Waveform of inverse_conversion_calc block

그림 8은 inverse_conversion_calc 블록에 입력이 인가되고 연산을 수행한 후 깊이영상 정보로 출력 될 때까지 58 clk Delay가 발생하는 것을 나타낸 것이다.

4.2. resizing_img 블록 하드웨어 구현

resizing img 과정을 하드웨어로 구현하기 위해서는 전처리 과정에서 생긴 홀 로 인하여 한 라인내의 데이터가 여러 라인 데이터를 포함하고 있을 수 있으므로 영상의 픽셀 값을 프레임 메모리에 저장 해 두어야 한다. 하지만 프레임 메모리를 사용할 시 하드웨어 자원의 소모가 증가한다. 따라서 본 논문에서는 픽셀 정보 저장을 위한 메모리 접근 시 5개의 SRAM을 스위칭 하는 형태로 사용한다. 알고리즘에서 사용되는 라인 메모리는 5개이다. 알고리즘이 처리할 수 있는 최대 영상의 크기가 640×480이기 때문에 640개의 어드레스를 가지는 라인 메모리를 사용하였다. 본 알고리즘에서는 SRAM을 스위칭하여 픽셀의 깊이 값을 메모리에 읽기/

쓰기를 위하여 메모리 선택신호가 필요하다. 또한 메모리에 읽기와 쓰기가 가능함을 알려주는 Read enable flag와 Write enable flag가 필요하다. Read enable flag, Write enable flag를 판단하기위해 사용한 메모리개수가 필요하다.

$$M_{sel}=\begin{Bmatrix} y_p-y_{pf}+3 & \text{if, } L=1 \\ (M_{pres}+D_y)=5 & otherwise \end{Bmatrix} \quad (6)$$

쓰기를 수행 할 메모리 선택신호 M_{sel}는 수식 (6)과 같이 계산된다. 여기서 y_p는 y축 좌표 중 0과 영상의 세로사이즈 사이의 값이다. y_{pf}는 한 프레임에서 첫 번째 y_p이다. M_{pres}는 이전 라인의 최대 M_{sel}값이다. D_y는 현재 입력되는 y_p값에서 이전 라인에서 y_p의 최대값을 뺀 값이다. L은 영상의 라인을 나타낸다. 한 라인내의 y_p 값은 아래위로 최대 2 라인씩 점프하는 값을 가질 수 있으므로 아래위로 2개의 라인 메모리를 확보해야한다. 따라서 첫 라인의 경우 y_{pf}값이 M_{sel}가 3이 되도록 하고, y_{pf}와 y_p값들의 차이를 이용하여 M_{sel}를 결정하도록 한다. 나머지 라인들은 M_{pres}에 D_y를 더한 후 라인 메모리의 개수인 5로 나눈 나머지를 M_{sel}로 하여 깊이 값을 쓰기를 한다. 한 라인 내에서 아래위로 최대 2개의 라인 메모리를 확보하여 프레임 단위의 데이터를 읽기/쓰기를 수행 하기에는 5개의 라인 메모리로는 부족하다. 5개의 라인 메모리를 스위칭 하면서 경우에 따라 쓰기 중지 또는 읽기를 중지하는 기능이 필요하다. 따라서 사용된 메모리의 개수 U_{mem}를 이용하여 쓰기를 중지하는 flag s_{write}, 읽기를 중지하는 flag s_{read}를 구한다.

$$U_{mem}=\begin{Bmatrix} y_{\max}-y_{\min}+1 & \text{if, } L=1 \\ y_{\max}-y_{premax} & otherwise \end{Bmatrix} \quad (7)$$

사용된 메모리의 개수 U_{mem}는 수식 (7)과 같이 계산된다. 여기서 $y_{\max}$와 $y_{\min}$은 각각 현재의 입력된 라인에서 가장 큰 y_p값, 가장 작은 y_p의 값을 나타낸다. y_{premax}는 이전 라인이 입력됐을 때 y_p의 최대 값이다.

$$S_{write}=\begin{Bmatrix} 1 & \text{if, } R_{mem}<N_{mem} \\ 0 & otherwise \end{Bmatrix} \quad (8)$$

수식 (8)은 쓰기를 중지하는 flag인 s_{write}의 조건을 나타낸 것이다. R_{mem}은 사용하지 않은 남은 메모리이고, N_{mem}는 다음 라인에서 확보해야하는 필요한 메모리이다. R_{mem}가 N_{mem}보다 작을 경우 쓰기 동작을 중지하고 읽기만을 동작시켜 R_{mem}를 N_{mem}만큼 확보한 후에 다시 읽기/쓰기를 동시에 수행한다.

$$S_{read}=\begin{Bmatrix} 1 & \text{if, } \left|y_{\max}-y_{premax}\right|\le 1 \\ 0 & otherwise \end{Bmatrix} \quad (9)$$

수식 (9)는 읽기를 중지하는 flag인 s_{read}의 조건을 나타낸 것이다. $y_{\max}$와 y_{premax}의 값의 차가 1 이하일 경우 읽기가 되어야 할 메모리에 쓰기가 완료 되지 않은 시점에 읽기가 동작하는 경우가 발생하기 때문에 읽기를 중지하고 쓰기만을 동작시켜 쓰기가 완료 된 후에 다시 읽기/쓰기를 동시에 수행한다.

4.3. hole_filling 블록 하드웨어 구현

resizing img 블록의 출력을 입력받아 홀을 제거하기 위하여 3×3 마스크를 이용한 hole filling 연산을 수행한다. 3×3 마스크연산을 수행하는 과정에서 2개의 SRAM이 필요하고, 한 라인 지연이 발생한다. 그림 9는 hole filling Block에 입력이 인가되고 연산을 수행한 후 홀이 없는 최종 출력이 될 때까지 한 라인 지연이 발생하는 것을 시뮬레이션 파형으로 나타낸 것이다.

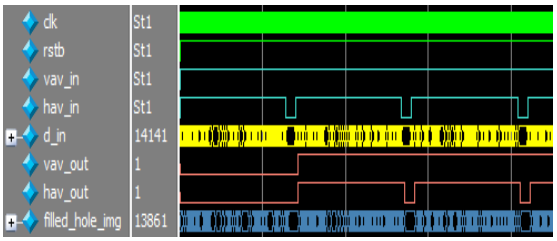


그림 9. hole_filling 블록 시뮬레이션 파형
Fig. 9 hole_filling Block Wave form

Verilog-HDL로 구현된 제안한 시스템은 FPGA와 CPU를 연동하여 확인 가능한 Zynq-7000 FPGA Board를 이용하여 실시간 처리를 검증 하였다. 본 논문에서는 Zynq-7000 All Programmable SoC Series 중 xc7z045ffg900-1 모델을 사용하여 하드웨어 IP를 검증

하였다[8].

표 1은 Xilinx PlanAhead을 이용하여 Device를 Zynq-7000으로 설정하여 합성한 결과이다. Registers는 12,478개를 사용하여 전체 면적의 2%를 차지하였고, LUTs는 12,264개를 사용하여 5%가 사용되었다. Slice는 5,010개를 사용하여 전체의 9%를 차지하였다. 사용된 RAMB36E1은 2개, RAMB18E1은 6개를 사용하였으며, DSP48E1은 5개, BUFG는 2개를 사용하였다. 전체 시스템의 최대 동작주파수가 103.22MHz까지 확보되는 것을 확인 하였다. 따라서 매우 작은 하드웨어 자원을 사용하면서 빠른 속도의 시스템 구현이 가능한 것을 알 수 있다.

표 1. Xilinx Design Analyzer 합성 결과
Table. 1 Xilinx Design Analyzer Result

Xilinx PlanAhead			
Device	Zynq-7000 (xc7z045ffg900-1)		
Slice Logic Utilization	Available	Used	Utilization
Registers (#)	437,200	12,478	2 %
LUTs (#)	218,600	12,263	5 %
Slice (#)	54,650	5,010	9 %
RAMB36E1/IFO36E1s	1,090	2	1 %
RAMB18E1/IFO18E1s	1,090	6	1 %
DSP48E1	900	5	1 %
BUFG	32	2	6 %
Minimum period (ns)	9,688		
Maximum Freq. (MHz)	103.22		

*IDEC의 EDA Tool을 제공받아 수행하였음.

V. 결 론

본 논문에서는 3차원 점군 데이터에서 깊이 영상으로의 영상 변환에서 반올림 오차와 영상의 크기 변화에 따른 데이터 손실이 발생하지 않는 효율적인 방법과 이를 하드웨어로 구현 하는 방법을 제안 하였다. 제안된 방법은 입력되는 3차원 점군 데이터에 좌표변환 수식을 적용한 후 왜곡 보정 알고리즘의 의하여 변화된 영상의 크기를 재조정 한다. 영상의 크기 재조정 후 반올림 오차와 데이터 손실에 의한 홀을 제거하여 깊이 영상을 출력한다. 본 논문에서 제안된 방법은 실시간 처리 가능하도록 하드웨어 구현을 하였고 시뮬레이션을 통하여 데이터의 손실 없이 효율적으로 3차원 점군 데이터에서 깊이 영상으로의 변환 가능하다는 것을 검증하였다. 또한 하드웨어 구현 측면에서 영상의 크기 조정 시 라인 메모리를 스위칭 하여 하드웨어를 효율적으로 구축하였다. Verilog-HDL을 사용하여 하드웨어로 설계를 하였으며, Xilinx PlanAhead을 이용하여 Device를 Zynq-7000으로 합성시 최대 동작주파수가 103.22MHz까지 확보되는 것을 확인 하였고, 또한 FPGA보드를 이용하여 확인하였다.

감사의 글

본 논문은 동아대학교 학술연구비 지원에 의하여 연구되었음

REFERENCES

[1] I. Benbasat, “HCI Research: Future Challenges and Directions,” *AIS Transactions on Human-Computer Interaction,* vol. 2, no. 2, pp. 16-21, Jun. 2010.

[2] M.A. Garcia and A. Solanas, “3D Simultaneous Localization and Modeling from Stereo Vision,” *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 847-853, May. 2004.

[3] K.H Jang, H.S. Cho, G.J. Kim, and B.S. Kang, “Depth Image Distortion Correction Method according to the Position and Angle of Depth Sensor and Its hardware implementation,” *Journal of the Korea Institute of Information and Communication Engineering*, vol. 18, no. 5, pp.1103-1109, May, 2014.

[4] R. Zhou, J. Wu, Q. He, C. Hu, and Z. Yu, “Approach of Human Face Recognition Based on SIFT Feature Extraction and 3D Rotation Model,” *IEEE International Conference on Information and Automation*, pp. 476-479, Jun. 2011.

[5] I.S. Seo, G.P. Jo, K.H. Jang, and B.S. Kang, “Angle Correction Method of Depth Camera form Prevention of Hole Generation,” *The Korea Institute of Signal processing and Systems Summer Conference Proceedings* , pp. 70-72, Jul. 2013.

[6] J. Biswas and M. Veloso, “Depth Camera Based Indoor Mobile Robot Localization and Navigation,” *IEEE International Conference on Robotics and Automation*, pp.1697-1702, May. 2012.
[7] H.W. Kim, G.P. Jo, K.H. Jang, and B.S. Kang, “Resizing Image Algorithm for Real time processing and Hardware design,” *The Korea Institute of Signal processing and Systems Winter Conference Proceedings*, pp. 212-214, Dec. 2013.
[8] Xilinx, Zynq-7000 All Programmable SoC Technical Reference Manual [Online] Available: http://www.xilinx.com, Apr, 2013.

장경훈(Kyounghoon Jang)

2010년 2월 동아대학교 전자공학과 (공학사)
2010년 3월 ~ 현재 동아대학교 전자공학과 석 · 박사 통합 과정
※관심분야 : 영상 신호처리, VLSI architecture design

조기쁨(Gippeum Jo)

2013년 2월 동아대학교 전자공학과 (공학사)
2013년 3월 ~ 현재 동아대학교 전자공학과 석사 과정
※관심분야 : 영상 신호처리, VLSI architecture design

김근준(Geun-Jun Kim)

2013년 2월 동아대학교 전자공학과 (공학사)
2013년 3월 ~ 현재 동아대학교 전자공학과 석 · 박사 통합 과정
※관심분야 : 영상 신호처리, VLSI architecture design

강봉순(Bongsoon Kang)

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