특집논문 (Special Paper)
방송공학회논문지 제24권 제6호, 2019년 11월 (JBE Vol. 24, No. 6, November 2019)
https://doi.org/10.5909/JBE.2019.24.6.1064
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

임베디드 GPU에서의 병렬처리를 이용한 모바일 기기에서의 다중뷰 스테레오 정합

전 윤 배[a)], 박 인 규[a)‡]

Multiview Stereo Matching on Mobile Devices Using Parallel Processing on Embedded GPU

Yun Bae Jeon[a)] and In Kyu Park[a)‡]

요 약

다중뷰 스테레오 정합 알고리즘은 시점이 다른 복수의 2차원 영상으로부터 3차원 형상을 복원하기 위해 사용된다. 기존의 다중뷰 스테레오 정합 알고리즘은 단계별로 많은 계산량을 포함하는 복잡한 구조 때문에 고성능 하드웨어에서만 주로 구현되어왔다. 그러나 최근에 모바일 그래픽 프로세서가 발전하면서 충분한 부동소수점 계산 성능이 확보됨에 따라 기존의 PC 환경에서만 수행되었던 복잡한 컴퓨터 비전 알고리즘들이 모바일 GPU에서 구현되고 있다. 본 논문에서는 임베디드 보드의 모바일 GPU에서의 병렬처리를 기반으로 다중뷰 스테레오 알고리즘의 병렬처리를 구현하고 자원이 제한적인 하드웨어에서의 성능 최적화 기법을 제안한다.

Abstract

Multiview stereo matching algorithm is used to reconstruct 3D shape from a set of 2D images. Conventional multiview stereo algorithms have been implemented on high-performance hardware due to the heavy complexity that contains a large number of calculations in each step. However, as the performance of mobile graphics processors has recently increased rapidly, complex computer vision algorithms can now be implemented on mobile devices like a smartphone and an embedded board. In this paper we parallelize an multiview stereo algorithm using OpenCL on mobile GPU and provide various optimization techniques on the embedded hardware with limited resource.

Keywords : Embedded GPU, parallel processing, multiview stereo matching, OpenCL

a) 인하대학교 정보통신공학과(Inha University, Department of Information and Communication Engineering)
‡ Corresponding Author : 박인규(In Kyu Park)
E-mail: pik@inha.ac.kr
Tel: +82-32-860-9190
ORCID: https://orcid.org/0000-0003-4774-7841

※ 이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2017-0-00142, 스마트기기를 위한 온디바이스 지능형 정보처리 가속화 SW플랫폼 기술 개발).
※ This work was supported by Institute for Information & communications Technology Planning&Evaluation(IITP) grant funded by the Korea government(MSIT) (No.2017-0-00142, Development of Acceleration SW Platform Technology for On-device Intelligent Information Processing in Smart Devices)
· Manuscript received October 7, 2019; Revised November 21, 2019; Accepted November 21, 2019.

Copyright Ⓒ 2016 Korean Institute of Broadcast and Media Engineers. All rights reserved.
"This is an Open-Access article distributed under the terms of the Creative Commons BY-NC-ND (http://creativecommons.org/licenses/by-nc-nd/3.0) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited and not altered."

Ⅰ. 서 론

다중뷰 스테레오 정합 알고리즘은 정밀한 3차원 복원을 위해서 많은 입력 영상에 대해 복잡한 알고리즘을 요구하기 때문에 기존의 연구환경은 대부분 PC 기반의 고성능 하드웨어로 제한되어 있었다[1,5,10,13]. 그러나 최근 모바일 GPU의 성능이 급격하게 발전하면서 다양한 컴퓨터비전 알고리즘을 임베디드 환경에서 구현하는 연구들이 늘고 있다[8,11]. 본 논문에서는 다중뷰 스테레오 정합 전체 과정을 OpenCL 기반으로 병렬화하고 임베디드 보드의 모바일 GPU상에서 실제 구동하며 최적화하여 CPU로 처리하는 기존 기법과의 정량적 성능 비교 및 평가를 수행한다.

본 논문에서 적용한 다중뷰 스테레오 정합 알고리즘은 카메라 시점을 알고 있는 2차원 영상으로부터 표면요소(surfel) 기반의 3차원을 복원하는 알고리즘을 기반으로 하며[2], 이는 복잡한 부동소수점(floating-point) 연산과 많은 메모리를 요구한다. 다중뷰 스테레오 알고리즘은 각 단계별로 화소, 3차원 점구름(point cloud), 그리고 구축된 표면요소의 개수만큼 반복문을 포함하기 때문에 GPU의 다중 코어로 병렬처리 하기 적합하다. 본 논문에서는 타겟 임베디드 보드의 하드웨어적 특성에 맞게 적용하는 병렬처리 과정과 병렬처리 알고리즘의 최적화 기법을 제안한다. 제안하는 기법의 기존의 기법에 비해 가지는 기여점을 다음에 요약하였다.

- 기존의 임베디드 GPU에서 병렬화 한 알고리즘에 비해 훨씬 대규모의 컴퓨터비전 알고리즘에 대해 병렬화 및 최적화 수행
- 저사양의 임베디드 GPU에서의 병렬화 관점에서의 효과적인 병렬화 방식과 최적화 방식에 대한 가이드라인 제공
- 실제 다중뷰 데이터에 대한 알고리즘 수행 결과 및 성능을 확인함으로써 병렬화 구현의 정확성과 효율성을 증명

본 논문의 구성은 다음과 같다. 2장에서는 표면요소 기반의 다중뷰 스테레오 정합 알고리즘의 과정을 소개하고, 각 단계별로 적용될 수 있는 병렬화 요소를 제시한다. 또한 GPU 병렬처리 알고리즘의 OpenCL 기반 최적화 기법에 대해 소개한다. 3장에서는 제안하는 기법의 성능을 정량적, 정성적으로 평가한다. 마지막으로 4장에서 본 논문의 결론을 맺는다.

Ⅱ. 제안하는 기법

1. 표면요소 기반 다중뷰 스테레오 정합 알고리즘

본 논문에서는 기존의 표면요소 기반 다중뷰 스테레오 알고리즘[2]를 기반 알고리즘으로 채택하여 전체 알고리즘을 병렬화하고 임베디드 보드에서 구동한다. 전체적인 알고리즘의 구조는 그림 1에 도시하였다.

1.1 Plane Sweeping Stereo 알고리즘을 통한 다중뷰 깊이영상 복원

표면요소는 3차원 좌표와 법선 벡터를 갖는 3차원 형상 복원의 기본 단위이며 3차원 점으로 표현된다. 우선 3차원 복원을 위해 plane sweeping stereo[3,5]기법으로 다중뷰 깊이 영상을 얻는다. 입력 다중뷰의 인접한 영상들로부터 깊이 영상을 얻기 위해 모든 인접 카메라에 대해 호모그래피(homography) 행렬로 시점이 보정된 윈도우 패치(patch)와 현재 영상의 윈도우 패치 간 비용을 계산한다. Plane sweeping stereo 알고리즘은 참조 영상에 대한 인접한 영상

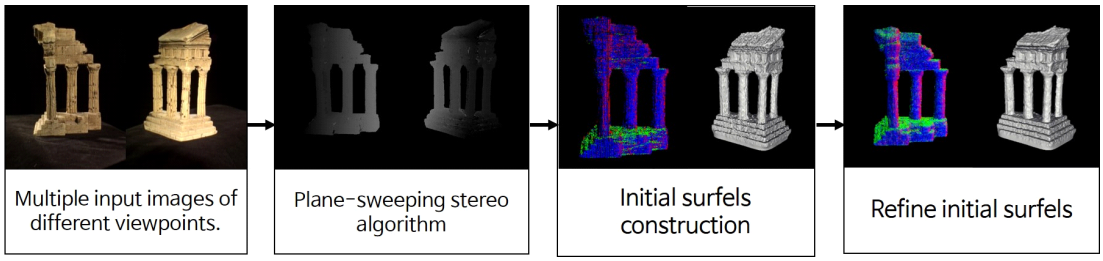


그림 1. 병렬화 대상인 다중뷰 스테레오 알고리즘[2]의 전체적인 구조
Fig. 1. Overall structure of the multiview stereo algorithm employed in this paper

들을 알고 있는 상황에서 화소 별로 독립적으로 비용을 계산하기 때문에 병렬처리가 용이하다.

1.2 초기 표면요소 모델 구축

Plane sweeping stereo 알고리즘으로 구한 다중뷰 깊이 영상은 초기 표면요소를 구축하는 데 사용된다[2]. 초기 표면요소는 다중뷰 깊이 영상에 3차원 볼륨 공간의 모든 3차원 복셀(voxel)을 투영하여 2개 이상의 카메라로부터 보여지는 표면요소에 대해서만 구축된다. 각 참조 영상에서 모든 3차원 점에 대해 독립적으로 구축 과정이 실행되기 때문에 초기 평면 구축 또한 병렬처리가 가능하다.

1.3 컬러 일관성을 이용한 초기 표면요소 모델의 정제 (refinement)

위에서 구축한 초기 표면요소는 지지 카메라의 각 지지 영상에 대하여 컬러 일관성(photo-consistency)이 유지되지 않는 아웃라이어(outlier)를 포함하고 있다. 구축된 초기 표면요소들을 모든 지지 영상에 투영하여 참조 영상에 의한 표면요소의 화소 값과의 컬러 일관성을 SAD 비용 함수로 계산한다. 이러한 초기 표면요소를 구축 후 아웃라이어를 제거하는 과정은 대부분 멀티뷰 스테레오 알고리즘에서 채택하고 있으며 이와 같은 과정을 거쳐 초기 결과물에 대한 표면 정제 과정을 진행한다[9]. 이 과정은 지역적 최적화 과정으로 해석할 수 있으며, 참조하는 영상의 각 표면요소 인덱스에 따라 독립적으로 실행되기 때문에 병렬처리로 수행할 수 있다.

2. OpenCL 기반의 알고리즘 병렬화 기법

본 논문에서는 임베디드 보드에서 다중뷰 스테레오 병렬처리 알고리즘의 구현을 위해 OpenCL[4,7,8]을 사용하였다. OpenCL은 플랫폼에 제한없이 GPU가 존재하는 디바이스에 대해서 GPGPU를 사용할 수 있도록 도와주는 병렬 컴퓨팅 프레임워크이다. OpenCL에서 제공하는 디바이스 메모리 구조는 그림 2와 같으며 커널 함수 실행 시 병렬화 할 인자, 그 크기 그리고 반복 횟수를 정확히 명시해야 한다. OpenCL을 사용하여 순차적 알고리즘(serial code)을 병렬화 하는 과정은 아래와 같다.

① 반복 횟수가 많은 반복문의 어떤 인덱스(index) 변수를 워크아이템(work-item)으로 지정할 지 결정한다.
② 디바이스에서 접근하기 위한 모든 OpenCL 버퍼 메모리를 호스트(CPU) 측에서 할당하고 초기화한다.
③ 커널 소스 코드는 host의 순차적 알고리즘에서 인덱스 변수가 랜덤하게 접근되는 워크아이템으로 바뀌는 것을 고려하여 작성한다. OpenCL에서는 3차원(3-dimensional) 구조의 워크아이템까지 지원한다.
④ 커널 함수는 호스트 측에서 커널 실행 명령 함수를 호출함에 따라 실행되며, 이 때 명령 함수에서 그림

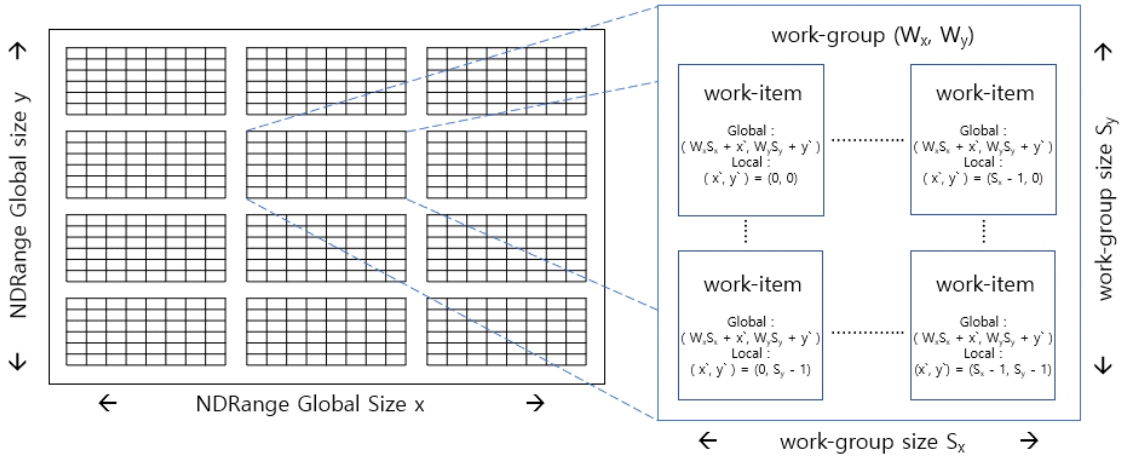


그림 2. OpenCL 디바이스 메모리 구조
Fig. 2. Device memory structure of OpenCL

2의 Global NDRange 크기와 그 시작점(offset) 그리고 워크그룹(work-group) 크기를 결정한다.

⑤ 커널 실행 이후 최종 결과 값이 저장되어 있는 디바이스 버퍼 메모리를 호스트 단의 메모리로 읽어온다.

3. 임베디드 GPU에서의 최적화 기법

모바일 GPU는 PC GPU와 구조상으로 다르기 때문에 타겟 디바이스가 바뀔 때마다 반드시 하드웨어 사양을 확인해야 한다. 우선 타겟 보드 하드웨어에서 지원하는 OpenCL의 버전과 사용 가능한 자료구조 및 확장 기능을 확인해야 한다[8]. 커널 소스 코드 및 호스트 측에서 작성하는 명령 실행문은 적용할 디바이스의 OpenCL 명세를 따라야 올바르게 병렬처리가 가능하다. 임베디드 보드에서 GPU 최적화 기법은 디바이스 메모리 관점의 최적화와 host 단의 커널 함수 실행 방법에 따른 최적화로 크게 2가지로 분류할 수 있다[6].

3.1 메모리 구조에 따른 OpenCL 최적화

OpenCL의 명령어는 디바이스의 메모리 정보와 하드웨어 정보를 상세히 알려준다. 해당 디바이스에서 사용할 수 있는 최대 메모리 크기와 읽고 쓰기에 최적화된 메모리 크기 그리고 디바이스 구조 상 가장 적합한 벡터 크기 등을 제공한다[4,6,7,8]. 이 중 MAX_WORK_GROUP_SIZE 값은 커널을 실행할 때 최대로 설정할 수 있는 워크그룹(work-group) 크기를 나타내며 처리하는 디바이스 메모리 크기가 그 값의 배수일 때 성능은 극대화된다. OpenCL에서 SIMD[4,7]연산은 벡터연산으로 구현 가능하며 각종 자료구조(char, int, half, float)에 대해 N(1, 2, 4, 8, 16)개의 데이터를 동시에 계산할 수 있다. 보통 PC환경의 고성능 GPU는 레지스터가 256bit로 8개의 32bit(int, float) 데이터를 한꺼번에 처리하는데 최적화되어 있고 모바일 GPU같은 경우 128bit로 4개의 데이터를 한꺼번에 처리하는데 최적화되어 있다. 따라서 커널 함수 실행 시 처리할 데이터의 전역적 크기가 워크그룹 사이즈의 배수가 되는 것이 중요하고, 커널 소스 코드 작성시 해당 디바이스에서 최적화된 크기의 벡터를 적극적으로 활용하는 것이 가장 기본적인 최적화 기법에 해당된다[6,7]. 마지막으로 CPU의 사양이 좋지 않은 모바일 환경에서는 디바이스 메모리 버퍼를 의미없이 반복적으로 초기화하지 않는 것이 바람직하다.

3.2 커널 함수 실행 법에 따른 OpenCL 최적화

모바일 환경에서는 커널 함수 호출하는 방식에 따라 실행 시간을 줄일 수 있다. 첫째로 작업 우선순위가 없는 커널 함수의 경우 OpenCL에서 제공하는 이벤트 처리기(event- handler)를 이용하여 호출하면 순차적으로 호출하는 것에 비해 매우 큰 이득을 얻을 수 있다[4,7]. 작업 A와 B가 우선순위가 없을 때, 이벤트로 등록하여 A와 B 커널 함수를 동시에 호출하는 것이 A → B 혹은 B → A 순차적으로 호출하는 것에 비해 더 빠르게 처리된다. OpenCL 커널 소스 코드 내에서 조건문은 CPU에서 사용하는 조건문에 비해 많은 부하를 일으키게 된다[6]. 디바이스 메모리의 전역 오프셋(offset)과 그 크기를 조절하면 경계선 문제(boundary-problem)와 같은 워크아이템(work-item)에 관한 조건문을 제거할 수 있다.

3.3 병렬 알고리즘 최적화 기법의 예시

본 논문에서 plane sweeping stereo 알고리즘과 정제 작업의 병렬 알고리즘 구현 시 입력 영상의 너비와 높이를 전역 NDRange 크기로 정하고, 각 화소를 work-item으로 지정하여 5x5 크기의 윈도우에 대해 SAD 비용을 병렬적으로 계산하였다. 이때 특정 화소에서 윈도우의 범위가 전역 크기를 넘어가는 경계 문제가 발생할 수 있는데, 본 논문에서는 전역 오프셋과 전역 NDRange 크기를 조절하여 경계 조건의 if 문을 제거함으로 branching diversity로 인한 병렬 알고리즘의 성능 저하를 막았다[6]. 또한 전역 NDRange 크기 Nx, y가 지정한 work-group 크기 Wx, y로 나누어떨어지지 않을 때 OpenCL 이벤트를 활용하여 명령어 큐에 대한 전역 크기를 NDRange(Nx – Nx % Wx, Ny – Ny % Wy)와 NDRange(Nx % Wx, Ny % Wy)로 나누어 Task-parallel 모델을 구축하여 실행하였다[4].

Ⅲ. 실험 결과

본 논문에서는 모바일 GPU에서 구현한 병렬화된 다중뷰 스테레오 알고리즘의 결과를 정량적, 정성적으로 측정하였다. 실험에서 사용한 임베디드 보드는 Rockchip사의 RK

그림 3. 실험에서 사용한 RK3399 임베디드 보드
Fig. 3. RK3399 Embedded board used in the experiment

3399 모델로 600Mhz의 쿼드코어 GPU Mali T860를 탑재하고 있고, 외관은 그림 3과 같다. 제안하는 병렬화 기법 및 최적화 기법의 정량적 결과로서 표 1의 실험환경에서 알고리즘의 각 단계별 CPU 실행 시간과 GPU 실행 시간을 초 단위로 측정하였다. 정성적 결과는 임베디드 보드에서 복원한 점구름 형태와 삼각 메쉬 형태의 3차원 렌더링 결과를 데이터 셋에 따라 캡처한 결과로 제시하였다. 실험에서 사용한 다중뷰 입력 영상 데이터 셋은 Middlebury 다중뷰 데이터 셋[1,12]의 TempleRing과 DinoRing을 사용하였다.

1. 디바이스에 따른 CPU와 GPU 실행 비교 결과

아래 표 2, 표 3의 결과는 각 디바이스의 CPU, GPU 실행 시간을 소수점 첫째 자리에서 반올림하여 초(sec)단위로 나타낸 것이다. 표 2, 표 3에서 공통적으로 Plane sweeping stereo와 초기 표면요서 정제 단계에서 시간이 많이 소요되는 것을 확인할 수 있다. Plane Sweeping Stereo 단계에서 PC의 경우 CPU에 비해 GPU가 56배 더 빠르게 처리한 것을 확인할 수 있었고 임베디드 보드의 경우 94배 더 빠르게 처리한 것을 확인할 수 있었다. 반면 가속 배율이 낮은 단계도 있는 것을 확인할 수 있다. 초기 평면 구축단계 같은 경우 PC에서 CPU로 실행한 결과와 GPU로 실행한 결과가 같았고, 임베디드 보드에서는 3배 정도 밖에 차이가 나지 않는 것을 확인할 수 있다. 또한 표 4에서 OpenCL에서 각

표 1. 실험 환경
Table 1. Experimental environment

Device	CPU	GPU	RAM
PC	Intel I7-7700 (4C8T @3.6GHz)	NVIDIA GTX1080TI (11GB)	16GB
Embedded board RK3399 (Custom)	Dual Coretex-A72 + Quad Cortex-A53 @1.8Ghz	Mali T860 MP4 (4C @600MHz)	4GB

표 2. PC에서 CPU, GPU간 단계별 알고리즘 실행 시간 비교
Table 2. Comparison of step-by-step execution time between CPU and GPU on PC environment

Dataset : templeRing 47 images @VGA	CPU	GPU	Speedup Ratio
Plane sweeping stereo SAD, 5x5	1,572	28	56.14
Construct initial surfels	1	1	1
Refine initial surfels	194	6	32.33
Sum	1,767	35	50.49

표 3. 임베디드 보드에서 CPU, GPU간 단계별 알고리즘 실행 시간 비교
Table 3. Comparison of step-by-step execution time between CPU and GPU on the embedded board

Dataset : templeRing 47 images @VGA	CPU	GPU	Speedup Ratio
Plane sweeping stereo SAD, 5x5	35,442	377	94
Construct initial surfels	19	6	3.16
Refine initial surfels (bundle adjustment)	2,267	22	103.05
Sum	37,728	405	93.16

표 4. 임베디드 보드 GPU와 PC GPU간 초당 부동 소수점 연산성능 비교
Table 4. Comparison of FLoating point Operations Per Second (FLOPS) between PC GPU and embedded GPU

	PC	RK3399
OpenCL (float)	12829.5 GFLOPS	25.4 GFLOPS
OpenCL (float4)	13119.3 GFLOPS	50.43 GFLOPS

디바이스 별로 초당 부동 소수점 연산량 비교를 통해 입력 데이터의 벡터화에 따른 이론적 가속 비율이 임베디드 보드가 PC보다 2배 더 빠르므로 표 2, 표 3에서 특정 단계의 가속 배율이 PC보다 높은 결과와 부합된다.

2. 데이터 셋에 따른 3차원 점구름 복원 결과

본 논문에서 제안한 기법의 정성적인 결과를 Middlebury

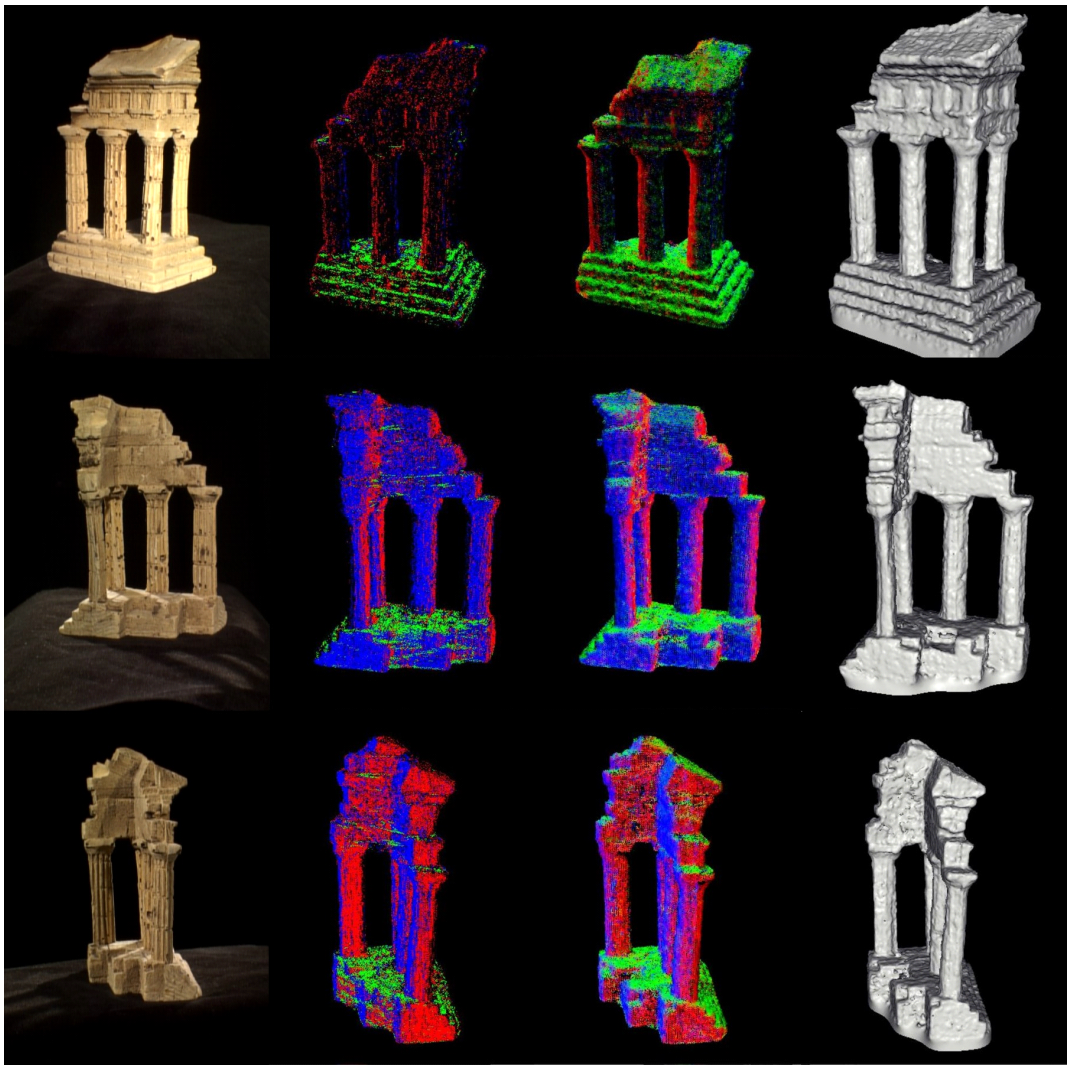
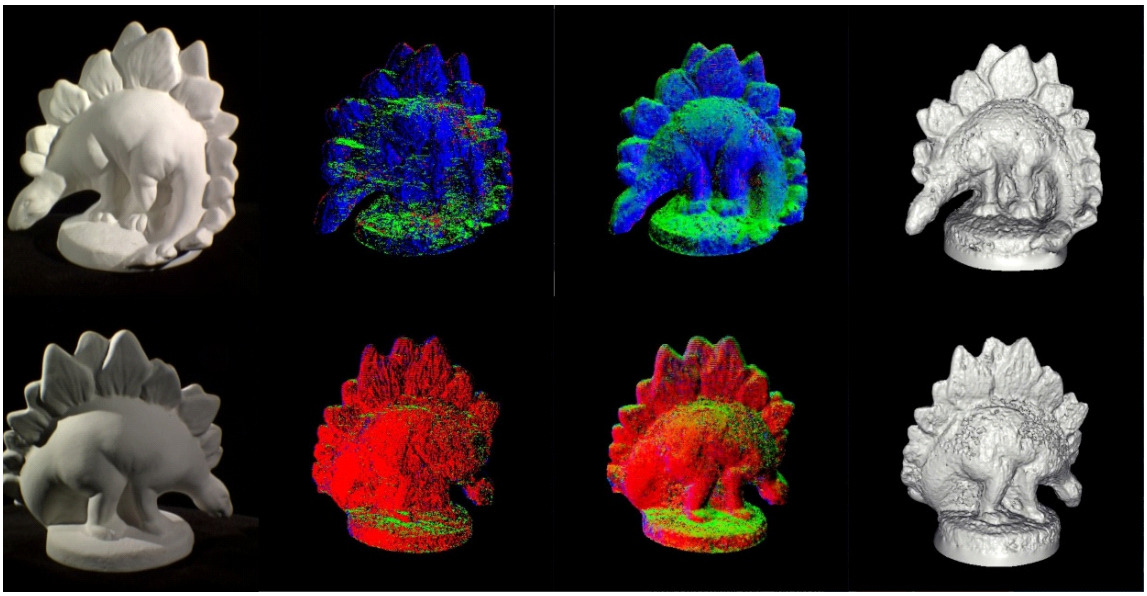


그림 4. 임베디드 보드에서 47장의 VGA 해상도의 TempleRing 데이터에 대한 3차원 복원 결과. (a) 입력영상의 예. (b) 초기 표면요소 구축 결과. (c) 표면 정제 과정 실행 결과. (d) (c)의 결과물을 삼각 메쉬 형태로 복원한 결과
Fig. 4. 3D reconstruction results for TempleRing data (47 input images with VGA resolution) on the embedded board. (a) Input sample. (b) Result of initial surfel construction. (c) Result of surface refinement. (d) Result of the triangular mesh reconstruction of (c)



(a) (b) (c) (d)

그림 5. 임베디드 보드에서 47장의 VGA 해상도의 DinoRing 데이터에 대한 3차원 복원 결과. (a) 입력영상의 예. (b) 초기 표면요소 구축 결과. (c) 표면 정제 과정 실행 결과. (d) (c)의 결과물을 삼각 메쉬 형태로 복원한 결과

Fig. 5. 3D reconstruction results for DinoRing data (48 input images with VGA resolution) on the embedded board. (a) Input sample. (b) Result of initial surfel construction. (c) Result of surface refinement. (d) Result of the triangular mesh reconstruction of (c)

Multiview 데이터셋의 templeRing 및 dinoRing 데이터에 대한 3차원 복원 결과로서 그림 4, 그림 5에 나타내었다. 그림 4, 그림 5에서 원본 영상과 각 표면요소의 법선 벡터 방향에 따른 색으로 구분하여 렌더링한 결과, 그리고 포아송 재구축 알고리즘에 따라 삼각 메쉬(mesh) 형태로 렌더링한 결과를 확인할 수 있다. 그림 4, 그림 5를 통해 초기 표면요소는 많은 아웃라이어를 갖고 있음을 확인할 수 있다. 그러나 표면 정제 단계를 거쳐 표면요소의 개수가 적어지고 결과가 선명해지는 것을 확인할 수 있다.

Ⅳ. 결 론

모바일 AP칩이 개인용 컴퓨터보다 훨씬 많은 시대인 현재, 본 논문에서 구현한 온디바스 개념은 관련 분야에서 연구하는 모든 사람들에게 중요한 과제가 되었다. 본 논문에서는 다중뷰 스테레오 정합 알고리즘의 전체 과정을 모바일 GPU를 활용한 온디바이스 시스템에서 구현하였다. 정량적 측면으로는 제안한 방법으로 병렬처리 하였을 때 TempleRing 47장의 VGA 데이터 셋에 대하여 시리얼 코드(C/C++)보다 93배 빠르게 처리하는 것을 확인하였다. 모바일 환경에서 처리속도는 전체적인 소모 전력을 줄이는 것을 의미하기 때문에 본 논문에서 제안한 병렬처리 최적화 기법은 에너지 측면에서도 도움이 될 수 있다.

참 고 문 헌 (References)

[1] S. M. Seitz, et al., “A comparison and evaluation of multiview stereo reconstruction algorithms,” In Proc. of IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 519-528, 2006.

[2] J. Y. Chang, et al., “GPU-friendly multiview stereo reconstruction using surfel representation and graph cuts,” Computer Vision and Image Understanding, vol. 115, no. 5, pp. 620-634, 2011.

[3] D. Gallup, et al., “Real-time plane-sweeping stereo with multiple sweeping directions,” In Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-8, 2007.

[4] A. Munshi, et al., OpenCL programming guide, Pearson Education, 2011.

[5] J. L. Schönberger, E. Zheng, J. M. Frahm, and M. Pollefeys, “Pixelwise view selection for unstructured multiview stereo,” In Proc. of European Conference on Computer Vision, 2016.

[6] I. K. Park, et al., “Design and performance evaluation of image proc-

essing algorithms on GPUs," IEEE Trans. on Parallel and Distributed Systems, vol. 22, no. 1, pp. 91-104, 2010.

[7] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A parallel programming standard for heterogeneous computing systems," Computing in Science & Engineering, vol. 12, no. 3, pp. 66, 2010.

[8] G. Wang, et al., "Accelerating computer vision algorithms using OpenCL framework on the mobile GPU-a case study," In Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 2629-2633, 2013.

[9] Y. Furukawa and C. Hernández, "Multiview stereo: A tutorial," Foundations and Trends® in Computer Graphics and Vision, vol. 9, no. 1-2, pp. 1-148, 2015.

[10] A. Ladikos, S. Benhimane, and N. Navab, "Efficient visual hull computation for real-time 3D reconstruction using CUDA," In Proc. of IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 1-8, 2008.

[11] N. Singhal, I. K. Park, and S. Cho, "Implementation and optimization of image processing algorithms on hand-held GPU," In Proc. of IEEE International Conference on Image Processing, pp. 4481–4484, 2010.

[12] The Middlebury Computer Vision Pages, http://vision.middlebury.edu/mview/.

[13] C. H. Esteban and F. Schmitt, "Silhouette and stereo fusion for 3D object modeling," Computer Vision and Image Understanding, vol. 96, no. 3, pp. 367-392, 2004.

저 자 소 개

전 윤 배

- 2019년 2월 : 인하대학교 정보통신공학과 학사
- 2019년 3월 ~ 현재 : 인하대학교 정보통신공학과 석사과정
- ORCID : https://orcid.org/0000-0001-6771-5797
- 주관심분야 : 컴퓨터비전 및 그래픽스 (영상기반 3차원 형상 복원, 증강현실), deep learning, GPGPU

박 인 규

- 1995년 2월 : 서울대학교 제어계측공학과 학사
- 1997년 2월 : 서울대학교 제어계측공학과 석사
- 2001년 8월 : 서울대학교 전기컴퓨터공학부 박사
- 2001년 9월 ~ 2004년 2월 : 삼성종합기술원 멀티미디어랩 전문연구원
- 2007년 1월 ~ 2008년 2월 : Mitsubishi Electric Research Laboratories (MERL) 방문연구원
- 2014년 9월 ~ 2015년 8월 : MIT Media Lab 방문부교수
- 2018년 7월 ~ 2019년 6월 : University of California, San Diego (UCSD) 방문학자
- 2004년 3월 ~ 현재 : 인하대학교 정보통신공학과 교수
- ORCID : https://orcid.org/0000-0003-4774-7841
- 주관심분야 : 컴퓨터비전 및 그래픽스 (영상기반 3차원 형상 복원, 증강현실, computational photography), GPGPU