특집논문 (Special Paper)
방송공학회논문지 제24권 제6호, 2019년 11월 (JBE Vol. 24, No. 6, November 2019)
https://doi.org/10.5909/JBE.2019.24.6.974
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

실시간 360 VR 스테레오 게임 영상 획득 성능 개선을 위한 다중 GPU 스케줄링에 관한 연구

이 준 석[a], 백 준 기[b)‡]

Multiple GPU Scheduling for Improved Acquisition of Real-Time 360 VR Game Video

Junsuk Lee[a] and Joonki Paik[b)‡]

요 약

게임 엔진을 기반으로 하는 실시간 360 VR(Virtual Reality) 스테레오 영상 획득 기술이 제안되었으나, 병목 현상이 발생하여 GPU(Graphics Processing Unit)의 성능을 충분히 활용하지 못 하고 있다. 본 논문에서는 기존에 제안된 실시간 360 VR 스테레오 영상 획득 기술의 병목 현상을 해결할 수 있도록 새로운 GPU 스케줄링 기법을 제안하고, 게임 엔진의 샘플 게임을 이용하여 제안하는 기법의 성능을 측정하였다. 측정 결과 기존에 제안된 기법보다 최대 약 70%의 성능 향상을 보였으며, GPU 자원이 좀더 균등하게 사용됨을 보였다.

Abstract

Real-time 360 VR (Virtual Reality) stereo image acquisition technique based on game engine was proposed. However, GPU (Graphics Processing Unit) resource is not fully utilized due to bottlenecks. In this paper, we propose an improved GPU scheduling technique to solve the bottleneck of the existing technique and measure the performance of the proposed technique using the sample games of the commercial game engine. As a result, proposed technique showed an improvement of performance up to 70% and usage of GPU resources more evenly compared existing technique.

Keyword : eSports, Broadcasting, Virtual Reality, Game Engine, Real Time Video Acquisition

a) 전자부품연구원 홀로그램연구센터(Hologram Research Center, Korea Electronics Technology Institute)
b) 중앙대학교 첨단영상대학원 영상학과(Dept. of Image Engineering, Graduate School of Advanced Imaging Science, Multimedia, and Film, Chung-Ang University)
‡ Corresponding Author : 백준기(Joonki Paik)
E-mail: paikj@cau.ac.kr
Tel: +82-2-820-5407
ORCID: https://orcid.org/0000-0002-8593-7155
※ This research is supported by Ministry of Culture, Sports and Tourism(MCST) and Korea Creative Content Agency(KOCCA) in the Culture Technology(CT) Research & Development Program 2017 (R2017030018).
· Manuscript received September 5, 2019; Revised November 4, 2019; Accepted November 12, 2019.

Ⅰ. 서 론

최근 해외에서는 e스포츠와 VR(Virtual Reality)를 접목한 중계서비스 시장이 형성되어 활발하게 확장되고 있다. 대표적으로 미국의 e스포츠 VR 중계 플랫폼인 SLIVER TV는 게임 내에 가상 카메라 어레이(Virtual Camera Array)를 설치하여 게임을 360도로 저장할 수 있는 특허를 보유하고 있으며, VR 헤드셋으로 e스포츠 경기를 실시간으로 관람할 수 있는 LiveVRCast 기술 등을 발표하고 상용화에 성공한 바 있다. 세계 최대 e스포츠 리그인 ESL(Elect- ronic Sports League)과 파트너쉽을 체결하고 2016년 10월 세계 최초로 Counter-Strike 토너먼트를 실시간으로 중계했다[1]. 최근 국내에서도 상용 게임엔진을 기반으로 하여 360 VR 스테레오 고품질 영상을 획득하는 기술들이 연구되고 있다[2,3]. 하지만 기존 연구들의 경우 앨리어싱(Aliasing), GPU 연산 부하와 같은 문제들이 내제되고 있으며, 실제 상용화를 위해서는 위와 같은 문제점들을 해결하고 시스템에 대한 고도화 및 안정화에 대한 연구가 반드시 필요하다. 이에 본 논문에서는 기존에 발표되었던 기술들과 연구에 대한 문제점을 분석하고, 해당 문제점을 개선하여 안정적으로 실시간 360 VR 스테레오 게임 영상을 획득하기 위한 새로운 기법을 제안하고 구현하였다. 그리고 실험을 통해 기존 Kim[3]이 제안한 방식과의 GPU(Graphic Processing Unit) 사용에 대한 효율성을 비교하였고, 앨리어싱과 같은 문제 해결과 생성된 영상에 대한 비트레이트(Bitrate) 개선 등에 대한 검증을 수행하였다.

본 논문의 구성은 2장에서 관련 연구로 기존의 360 VR 영상 획득 기술을 분석하고 개선해야 하는 문제점을 연구하였다. 3장에서 개선된 360 VR 영상 획득 기술을 제안하였고, 4장에서 실험을 통해 그 기능과 성능을 기존 연구의 결과와 비교 검증하였다. 마지막으로 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

1. 360 VR영상 생성 기술

사람은 두 눈으로 보는 각각의 두 영상을 통해 입체감을 인지할 수 있다. 가까이 있는 물체는 멀리 있는 물체보다 더 큰 시야의 차이를 가져오며, 이에 따라 스테레오 영상을 획득하려면 IPD(Interpupillary Distance)에 따라 두 대의 카메라에서 동기화된 영상을 획득하여야 한다. 이렇게 구성된 두 대의 카메라를 모든 각도로 회전시켜 360 스테레오 영상을 획득할 수 있으며, 가상 공간에서는 렌더링을 통해 이를 간단하게 획득할 수 있다[4].

큐브맵(Cubemap) 이미지를 변환하여 ERP(Equirectangular Projection) 이미지를 획득할 수 있다. 이러한 ERP 변환은 필요한 연산이 적기 때문에 더 낮은 사양의 환경에서도 높은 속도로 변환을 수행할 수 있으며, GPU을 통하여 쉽게 가속화할 수 있다는 장점이 있다.

2. 게임 엔진 기반의 360 VR 영상 획득 기술

최근 발표된 게임 엔진 기반의 360 VR 영상 획득 기술의 대표적인 기술로는 Facebook의 360 Capture SDK, Nvidia의 Ansel이 있다.

Facebook에서 오픈소스로 공개한 360 Capture SDK는 실시간으로 4K 30FPS(Frames per Second)의 360 VR 영상을 획득할 수 있다. 또한 Facebook과 연동하여 실시간으로 360 VR 영상의 스트리밍이 가능하다[5]. 하지만 실제 VR 기기에서 요구되는 60FPS 이상의 영상을 실시간으로 획득할 수 없다는 단점이 있다.

Nvidia에서 발표한 Ansel은 고품질 360 게임 내 이미지 획득 기술이다[6]. 다만 게임 진행을 멈추고 자유 시점으로 이동하여 16K 이상의 초 고화질 360 이미지를 획득할 수 있지만, 실시간으로 영상을 생성할 수 없는 단점이 있다.

국내에서 발표된 Kim의 연구[3]에서는 가상 공간 내 가상 카메라를 배치하여 스테레오 큐브맵 이미지를 획득하여 픽셀 맵핑을 통해 ERP이미지 형식의 360 VR 스테레오 영상을 획득하는 기법을 제안하였으며, 이를 통해 실시간으로 4K(4096x4096) 360 VR 스테레오 60FPS 영상 획득이 가능함을 확인하였다.

하지만 Kim이 제안한 기법에서는 다음과 같은 몇 가지 개선해야 할 문제점이 있다. 첫째, ERP 변환을 하기 위한 GPU 연산을 줄이기 위하여 사전에 역 매핑 테이블을 생성하는데, 이를 생성하는 시간이 상당히 소요되기 때문에 초

기화 과정을 위한 긴 시간이 요구된다. 둘째, 연산량 감소를 위해 역 매핑 테이블을 사용하기 때문에 ERP 변환 시 영상에 앨리어싱(Aliasing)이 발생하게 된다. 셋째, 게임 렌더링과 ERP 변환이 모두 하나의 GPU에서 수행되기 때문에 렌더링 과정에서의 병목 현상이 발생한다. 넷째 많은 프레임을 한 번에 인코딩하기 때문에 인코딩 시 GPU에 순간적인 부하가 주기적으로 발생함으로 안정성이 떨어진다.

본 논문에서는 Kim의 연구[4]에 존재하는 위와 같은 문제점을 해결하기 위해 시스템 구조를 재 설계하고, 새로운 GPU 스케줄링 기법을 제안하고 설계한다.

Ⅲ. 360 VR 스테레오 게임 영상 교차 획득 시스템

1. 360 VR 스테레오 게임 영상 획득 시스템 설계

제안하는 시스템은 게임 렌더링을 위한 GPU(GPU#0)와 인코딩을 위한 GPU 두 개(GPU#1, GPU#2)를 사용한다. 게임 렌더링 및 360 VR 가상 카메라는 GPU#0에서 동작하며 60FPS 속도로 렌더링 되어 총 12개의 텍스처가 획득된다. 이 때 획득된 텍스처는 GPU#0에서 ERP 변환을 수행하지 않고 곧바로 GPU#1 또는 GPU#2로 전송된다. 전송 작업 외 추가적인 작업을 수행하지 않기 때문에 GPU#0은 전송이 끝나고 곧바로 다음 렌더링 작업을 시작할 수 있다. 인코딩 작업 시 하나의 GPU가 GOP(Group of Picture) 만큼의 프레임을 연속적으로 인코딩해야 하기 때문에 GOP 크기만큼의 프레임이 연속적으로 인코딩 GPU로 전송되어야 한다. 따라서 12개의 텍스처를 GOP크기와 동일한 횟수만큼 GPU#1과 GPU#2에 번갈아 가며 전송한다. 표 1은 ERP 변환 전/후 이미지를 다른 GPU로 전송할 때 필요한 전송 대역폭을 설명하고 있다.

표 1. GPU간 전송할 텍스처 종류에 따른 필요 대역폭
Table 1. Required bandwidth depending on the type of texture to transfer between GPUs

Texture Type	Bandwidth
4K VR Stereo Image (4096x4096) 8bit RGBA 60FPS	3.84 GB/s
1K Image (1024x1024) 8bit RGBA 12EA 60FPS	2.88 GB/s

표 1에서 확인할 수 있듯이, ERP 변환 후의 이미지를 전송하는 것 보다 2D 텍스처를 12개 전송하는 것이 필요 대역폭이 적으며 이를 통해 GPU간 전송 작업 시 더 빠른 시간에 전송을 완료하여 이후 렌더링 작업의 지연을 더욱 줄일 수 있음을 알 수 있다.

그림 1은 본 논문에서 제안하는 시스템의 전체적인 구조를 나타내고 있다.

인코딩 GPU는 12개의 텍스처가 모두 전송되면 곧바로 전송된 12개의 텍스처에 대해 ERP 변환을 수행한다. 이 시점의 인코딩 GPU는 ERP 변환 외 다른 작업을 수행하지 않기 때문에 ERP변환 연산을 줄이기 위하여 역 매핑 테이블을 별도로 생성할 필요가 없다. 또한 인코딩 GPU의 남는

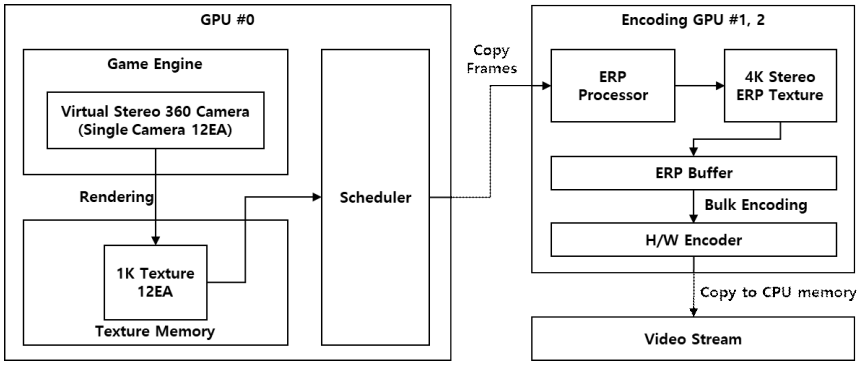

그림 1. 제안 시스템 구조도
Fig. 1. Structure of proposal system

연산 성능을 이용하여 ERP 변환 시 보간(Interpolation)을 적용할 수 있다 이를 이용하여 역 매핑 테이블을 사용하였을 때 발생하는 앨리어싱 현상을 제거하는 안티 앨리어싱(Anti-Aliasing) 효과를 얻을 수 있으며 이는 그림 2에서와 같은 결과로 나타나게 된다.

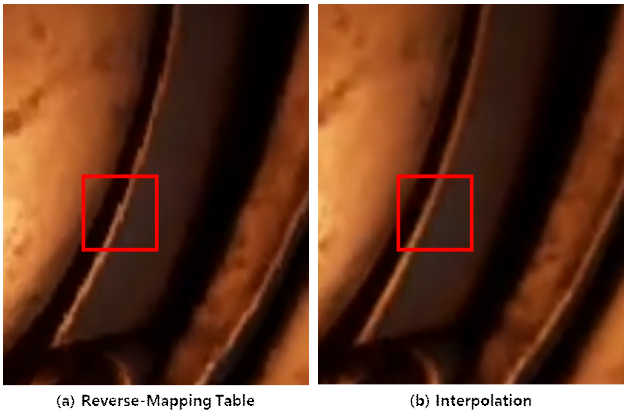

그림 2. 역 매핑 테이블과 보간 방식의 결과 비교
Fig. 2. Comparation of reverse-mapping table and interpolation

1K 텍스처 12개를 ERP 변환하여 획득된 4K 360 VR 스테레오 이미지의 경우, 크기가 너무 크기 때문에 최신의 GPU 기반 인코더를 사용하더라도 60FPS 속도로 인코딩이 불가능하다. 때문에 ERP 변환 후 곧바로 인코딩을 시작하게 되면 이후 GPU#0으로부터 전송되는 텍스처 수신이 지연되어 병목 현상이 발생하게 된다. 이 문제를 해결하기 위해 ERP 변환 이후 곧바로 인코딩을 시작하지 않고 GOP 크기만큼 변환이 완료될 때까지 GPU 메모리에 저장하도록 하였다. GOP 크기만큼의 ERP 변환이 완료되면 저장된 모든 360 VR 스테레오 이미지를 이용하여 인코딩을 시작한다.

본 논문에서 제안하는 기법과 기존 Kim이 제안한 기법의 GPU 작업 순서는 그림 3과 같다.

Kim이 제안한 기법은 GPU#0이 매 프레임마다 Rendering, ERP, Copy 작업을 수행한다. ERP와 Copy 작업이 완료될 때까지 다음 프레임의 렌더링을 시작할 수 없기 때문에 전체적인 게임 렌더링 성능이 저하되는 병목 현상이 발생한다. 본 논문에서 제안하는 기법은 GPU#0은 Rendering 및 Copy 작업만을 수행하고 다른 GPU가 ERP 작업을 수행하기 때문에 Rendering 및 ERP 작업이 동시에 진행될 수 있어 더 빠른 주기로 Rendering 작업을 수행할 수 있고, 각 GPU에 비교적 균등하게 작업이 할당됨을 확인할 수 있다.

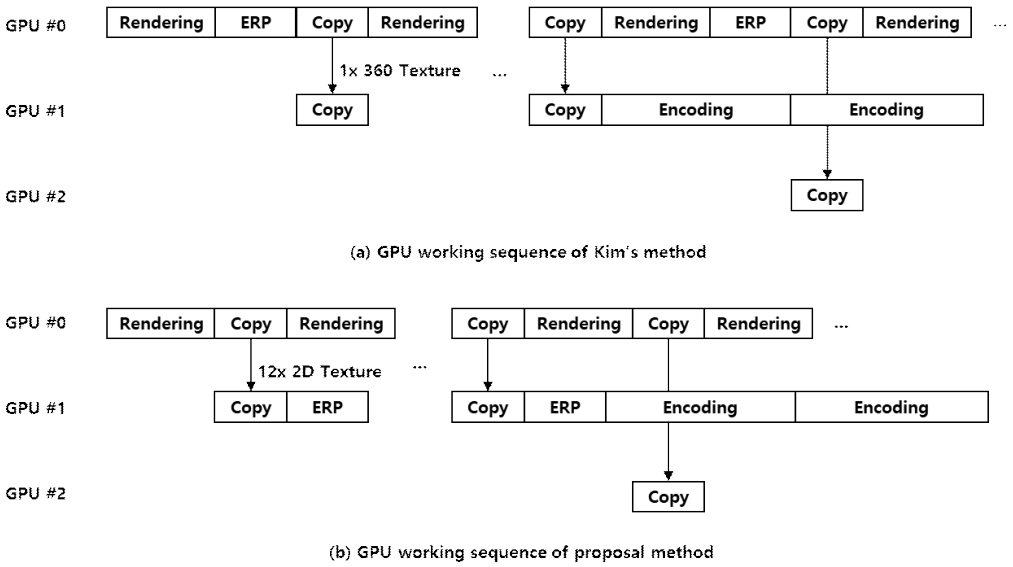

그림 3. GPU 작업 순서 비교
Fig. 3. Comparation of GPU working sequences

2. 360 VR 스테레오 게임 영상 획득 시스템 구현

앞서 설계한 360 VR 스테레오 영상 교차 획득 시스템을 구현하기 위하여 Unreal Engine 4.17 버전을 기반으로 하여 플러그인을 구현하였다. 가상 카메라는 1024x1024 크기의 90도 FOV 카메라를 12개 이용하였다. Unreal Engine에서 획득 가능한 텍스처 형식과 CUDA에서 지원 가능한 텍스처 형식의 호환성 이슈로 인하여 8bit RGBA 형식의 텍스처가 아닌 16bit RGBA 형식의 텍스처를 사용하였다. 이 경우 GPU간 전송 대역폭의 한계 때문에 60FPS로 텍스처 12개를 전송할 수 없게 된다. 따라서 획득된 16bit 텍스처를 우선 8bit 텍스처로 다운 샘플링(Down Sampling) 하는 과정을 추가하여 GPU간 전송 대역폭 내에서 충분히 전송 가능하도록 하였다. GPU간 전송 대역폭 및 텍스처 형식에 따른 필요 대역폭은 표 2와 같다.

표 2. 텍스처 전송에 필요한 대역폭과 GPU간 데이터 전송 대역폭
Table 2. Bandwidth of texture transfer and data transfer between GPUs

Type	Bandwidth
Copy GPU to GPU (without SLI)	5.5 GB/s
16bit 1024x1024 RGBA 12ea 60FPS	5.76 GB/s
8bit 1024x1024 RGBA 12ea 60FPS	2.88 GB/s

텍스처 다운 샘플링 및 ERP 변환은 CUDA 커널(Kernel)을 이용하여 구현하였고, ERP 변환의 출력으로 4K(4096x4096) 8bit RGBA 형식의 텍스처가 생성되도록 하였다. Nvidia 하드웨어 인코더를 이용하여 h.264 60FPS 비디오 스트림으로 인코딩하였다.

Ⅳ. 실험 환경 및 방법

1. 실험 환경 및 방법

게임엔진은 Kim이 제안한 기법과의 성능 비교를 위하여 동일한 Unreal Engine 4.17 버전을 사용하였다. 또한 Unreal Engine 버전에 따른 성능 비교를 위하여 추가적으로 4.22 버전을 이용하여 실험하였다. 실험을 위한 게임은 Unreal Engine에서 기본적으로 제공하는 샘플 게임을 이용하였다. 다양한 그래픽 효과 및 환경에서 실험할 수 있도록 각각 다른 특징을 가지는 샘플 게임을 4종 선정하였으며, 샘플 게임의 이름과 이미지는 아래 표 3과 같다. Sun Temple은 고사양 모바일 기능을 선보이도록 설계되었으며, 다양한 그래픽 기능 및 기술을 구현함에 있어 유용하게 사용될 수 있는 물리 기반 렌더링 샘플이 포함되어 있다[7]. Stylized Rendering은 게임에 스타일이 적용되어 만화 같은 형태의 환경을 렌더링하는 방법을 보여주는 샘플이며, 포스트 프로세싱(Post Processing)등의 기법을 사용하여 구현되었다[8]. Realistic Rendering은 Unreal Engine의 실사 렌더링 기능을 구현한 샘플이며, 빛과 관련된 각종 엔진 기능을 사용하여 현실과 같은

표 3. 실험 시료용 샘플 게임
Table 3. Sample games for experimental samples

Sample	Image	Sample	Image
Sun Temple		Stylized Rendering	
Realistic Rendering		Particle Effects	

환경을 구현하였다[9]. Particle Effects는 다양한 환경 및 입자 시스템(Particle System)을 구현한 샘플이며, 눈보라, 물의 흐름 및 불 등의 여러 자연 환경적인 효과를 구현하였다[10].

기존 Kim이 제안한 기법과 제안하는 GPU 스케줄링 기법의 성능 차이를 비교하기 위해 FPS와 각 GPU의 사용률을 측정하였다. 샘플 게임을 실행한 후 1분간 성능을 측정하였으며, 화면 전환 스크립트가 포함되지 않은 Realistic Rendering을 제외한 나머지 샘플 게임은 성능 측정을 하는 동안 자동으로 화면이 전환되도록 하였다.

실험에 사용된 시스템 환경은 표 4과 같다.

표 4. 실험용 시스템 환경
Table 4. Experimental system environment

Device Type	Device Name
CPU	Intel Core i9-7920X 2.9GHz
GPU	Nvidia Quadro RTX5000 3EA
Memory	Samsung DDR4 2400MHz 128GB
Storage	Samsung SSD 960 Pro 1TB
OS	Windows 10 Pro 64bit

2. 실험 결과

FPS 측정에 대한 실험 결과는 표 5와 같다.

모든 샘플 게임에서 Kim이 제안한 기법보다 본 논문에서 제안하는 기법이 더 좋은 성능을 보여주었으며, 샘플 게임에 따라 30%에서 70%정도의 성능 차이가 있음을 알 수 있었다. 특히 Particle Effect 샘플의 경우 Kim이 제안한 기법은 평균 20FPS 이하로, 영상으로 사용되기 어려운 정도의 성능을 보여주었으나, 제안하는 기법은 25FPS 이상, 특히 4.22 버전의 Unreal Engine을 사용하는 경우 36FPS 이상의 성능을 달성하여 충분히 영상으로 사용될 수 있을 정도의 성능을 보여주었다.

그림 4는 측정된 평균 FPS를 그래프로 표현한 것이다.

기본적으로 모든 샘플 게임에서 본 논문에서 제안하는 기법이 좋은 성능을 나타냄을 알 수 있다. 하지만 Sun Temple, Stylized Rendering, Realistic Rendering 세 가지 샘플에서 Kim이 제안한 기법은 거의 비슷한 성능을 나타내지만 본 논문에서 제안하는 기법은 각 샘플 별로 상당한 성능 차이가 발생함을 확인할 수 있었다.

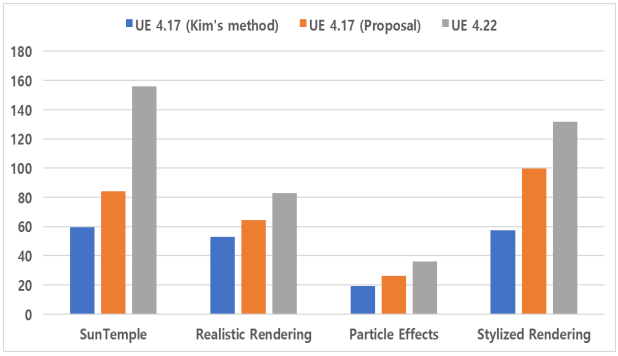

그림 4. 성능 측정 결과 그래프
Fig. 4. Graph representation of performance measurement result

그림 5는 Kim이 제안한 기법과 본 논문에서 제안하는 기법의 같은 샘플에서의 GPU#0에서의 GPU 사용률을 비교한 것이다.

표 5. 성능 측정 결과
Table 5. Performance measurement result

Sample	Version	Average FPS	Over 30FPS (%)	Over 60FPS (%)
Sun Temple	4.17 (Kim's method)	59.42	99.54	49.02
	4.17 (Proposal)	83.92	99.78	72.58
	4.22	155.80	99.99	99.95
Stylized Rendering	4.17 (Kim's method)	57.63	99.21	39.70
	4.17 (Proposal)	99.59	99.95	91.32
	4.22	131.54	99.99	95.17
Realistic Rendering	4.17 (Kim's method)	53.02	99.97	6.06
	4.17 (Proposal)	64.27	99.97	49.93
	4.22	82.97	99.98	95.43
Particle Effects	4.17 (Kim's method)	19.27	5.51	0.00
	4.17 (Proposal)	26.22	36.24	0.41
	4.22	36.28	50.26	7.11

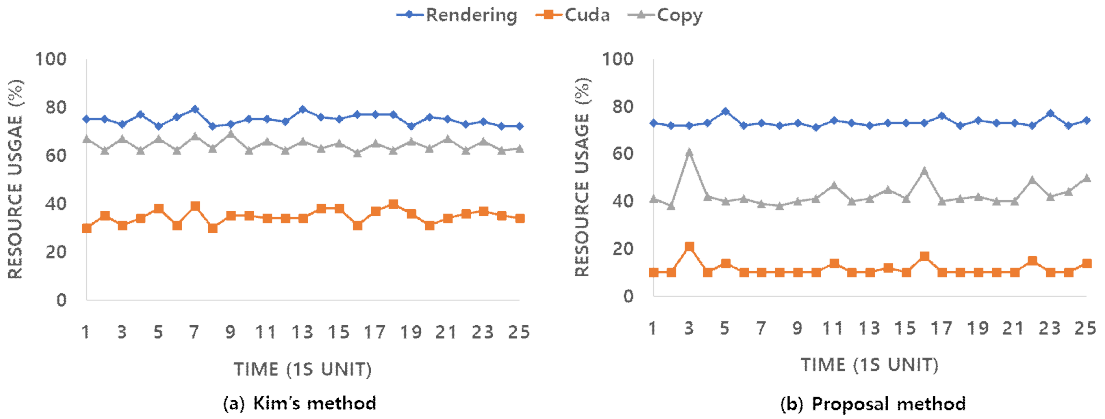

그림 5. 렌더링용 GPU(GPU#0)의 사용률 측정 결과
Fig. 5. The measurement result of rendering GPU(GPU#0) resource usage

렌더링 사용률은 같은 샘플을 렌더링하기 때문에 사용률에 차이가 발생하지 않는다. 본 논문에서 제안하는 기법은 ERP 변환을 수행하지 않고, 16bit 텍스처를 8bit 텍스처로 다운 샘플링만 하기 때문에 Kim이 제안한 기법에 비해 낮은 CUDA 사용률을 보이는 것을 알 수 있다. 또한 ERP 변환된 이미지를 전송하는 것 보다 12개의 2D 텍스처를 전송하는 것이 앞서 설명했던 것처럼 필요 대역폭이 적기 때문에 Copy 연산 또한 낮은 사용률을 보이는 것을 알 수 있다.

그림 6은 Kim이 제안한 기법과 본 논문에서 제안하는 기법의 인코딩 GPU (GPU#1, GPU#2)의 사용률을 비교한 것이다.

Kim이 제안한 기법은 60FPS에서 1초 동안 ERP 이미지를 저장해 두고 한 번에 인코딩을 하기 때문에 1초 단위로 인코딩 및 Copy 사용률이 극단적으로 변하게 된다. 본 논문

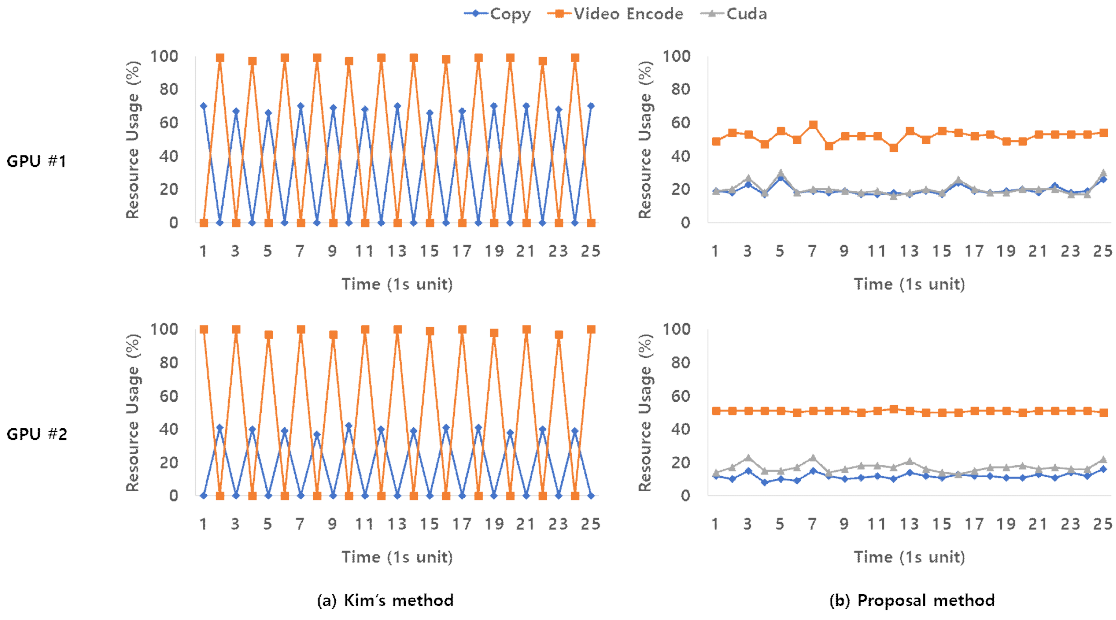

그림 6. 인코딩용 GPU(GPU#1, GPU#2)의 사용률 측정 결과
Fig. 6. The measurement result of encoding GPU (GPU#1, GPU#2) resource usage

에서 제안하는 기법은 GOP 단위로 ERP 이미지를 전송하기 때문에 비교적 균등한 사용률을 보임을 알 수 있다. Kim이 제안한 기법은 인코딩용 GPU에서 함께 ERP 변환을 수행하기 때문에 CUDA 연산이 존재하지 않는 반면, 본 논문에서 제안하는 기법은 인코딩용 GPU에서 CUDA 연산이 수행됨을 알 수 있다. GPU#1 및 GPU#2에서 Copy 사용률이 차이가 나는데, 이는 테스트 환경 PC의 PCI-e(Peripheral Component Interconnect Express) 구성에 의해 전송 속도에서 차이가 발생하여 생긴 현상이며, 영상 인코딩 성능에는 차이가 발생하지 않는다.

그림 7은 같은 렌더링 파라미터를 사용하여 1분 동안 획득된 영상의 5초 단위의 구간 별 평균 비트레이트(Bitrate)를 측정한 결과이다. 측정 결과 Kim이 제안한 기법보다 본 논문에서 제안하는 기법이 비트레이트가 낮게 나오는 것을 알 수 있었다. 일반적으로 Kim이 제안한 것과 같이 더 많은 프레임을 한 번에 인코딩하는 것이 더 낮은 비트레이트를 가질 것으로 예상되나, 실험 결과 반대로 본 논문에서 제안하는 기법이 더 낮은 비트레이트를 가지는 것을 확인할 수 있다. 이는 Kim이 제안하는 기법의 경우 그림 2에서와 같이 앨리어싱 현상에 의해 영상에 일종의 노이즈(Noise)가 발생하기 때문에 발생하는 형상이며, 인코딩 시 압축 효율이 좋지 않아 결과적으로 높은 비트레이트를 가지게 된다.

V. 결 론

본 논문에서는 360 VR 스테레오 영상을 획득하는 Kim이 제안한 게임 엔진 기반의 360 VR 스테레오 영상 획득 기법에 대해 분석하고, 문제점을 해결하기 위해 새로운 GPU 스케줄링 기법을 제안하였다. 실험 결과, 제안하는 기법은 기존의 방법보다 빠른 로딩 속도 및 균등한 GPU 사용율을 보이며, 같은 게임 엔진에서 Kim이 제안하였던 방법보다 약 30%에서 70%정도의 성능이 향상되었음을 확인할 수 있었다. 또한 같은 렌더링 방법을 사용하더라도 더 낮은 비트레이트를 가짐을 알 수 있었다. 하지만 고 사양의 시각적 효과가 렌더링 되는 경우 제안하는 방법에서도 60fps의 성능은 가질 수 없음을 알 수 있었다. 앞으로 지속적인 연구를 통하여 제안한 방법을 개선하여 고 사양의 게임에서도 좋

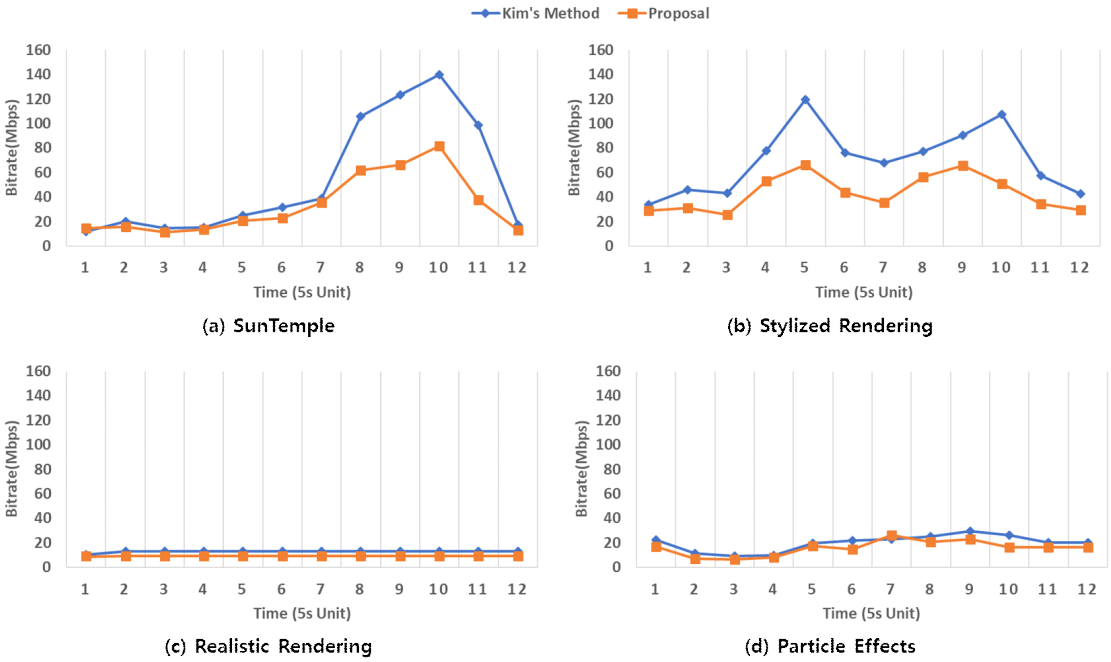

그림 7. 획득된 영상 비트레이트 측정 결과
Fig. 7. The measurement result of captured video bitrate

은 성능을 가질 수 있도록 개선할 예정이다.

참 고 문 헌 (References)

[1] KOCCA. "'SLIVER.tv', the E-Sports broadcasting platform via VR. Monthly Game Industrial Trends, Vol.1, pp.46-48, September 2017.

[2] H. W. Kim, J. W. Yang, Y. H. Kim, S. P. Yoon, and W. C. Park. "Virtual Camera design for 360degree game image acquisition,". Proceeding of Korea Multimedia Society Winter Conference, Busan, Korea, pp. 317-318, 2017.

[3] H. W. Kim, J. S. Lee, and S. H. Yang. "Study of Capturing Real-Time 360 VR 3D Game Video for 360 VR E-Sports Broadcast," Journal of Broadcast Engineering, Vol.23, No.6, pp.876 - 885, November 2018.

[4] Rendering Omni-directional Stereo Content, https://developers.google.com/vr/jump/rendering-ods-content.pdf (accessed Aug. 28, 2019)

[5] Announcing 360 Capture SDK, https://engineering.fb.com/developer-tools/announcing-360-capture-sdk (accessed Aug. 28, 2019)

[6] Ansel, https://www.nvidia.com/en-us/geforce/geforce-experience/ansel/ (accessed Aug. 28, 2019)

[7] UE4 Sun Temple, https://developer.nvidia.com/ue4-sun-temple (accessed Aug. 28, 2019)

[8] Stylized Rendering, https://docs.unrealengine.com/en-US/Resources/Showcases/Stylized (accessed Aug. 28, 2019).

[9] Realistic Rendering, https://docs.unrealengine.com/en-US/Resources/Showcases/RealisticRendering (accessed Aug. 28, 2019).

[10] Particle Effects, https://docs.unrealengine.com/en-US/Resources/Showcases/Effects (accessed Aug. 28, 2019).

[11] Unreal Frontend, https://docs.unrealengine.com/en-US/Engine/Deployment/UnrealFrontend (accessed Aug. 28, 2019).

저 자 소 개

이 준 석

- 2014년 8월 : 건국대학교 컴퓨터공학과 학사 졸업
- 2014년 4월 ~ 2017년 12월 : ㈜케이사인 정보보안연구소 선임연구원
- 2018년 1월 ~ 현재 : 전자부품연구원 홀로그램연구센터 연구원
- 2019년 3월 ~ 현재 : 중앙대학교 첨단영상대학원 영상학과 석사과정 재학
- ORCID : https://orcid.org/0000-0002-7028-9271
- 주관심분야 : VR, GPGPU, 데이터베이스

백 준 기

- 1984년 : 서울대학교 제어계측공학과 학사 졸업
- 1987년 : 노스웨스턴대학교 전기 및 컴퓨터 공학과 석사 졸업
- 1990년 : 노스웨스턴대학교 전기 및 컴퓨터 공학과 박사 졸업
- 2017년 ~ 현재 : 중앙대학교 첨단영상대학원 영상학과 교수
- ORCID : https://orcid.org/0000-0002-8593-7155
- 주관심분야 : 영상복원, 신호처리, 반도체