실시간 재생을 위한 TMIV 디코더의 GPU 구현

이상호, *신홍창, *이광순, *서정일
㈜하늘소프트, *한국전자통신연구원
leesangho@hanulsoftware.com, *hcshin@etri.re.kr, *gslee@etri.re.kr, *seoji@etri.re.kr

GPU Implementation of TMIV Decoder for Real-time Playback

Sangho Lee, *Hongchang Shin, *Gwangsoon Lee, *Jeongil Seo
Hanulsoft, Inc., *Electronics and Telecommunications Research Institute (ETRI)

요 약

TMIV 레퍼런스 모델에는 VWS(View Weighting Synthesizer), AS(Additive Synthesizer), MPIS(Multiplane Image Synthesizer)의 세 가지 방식의 렌더러 구현이 제시되어 있는데 본 논문에서는 VWS 에 포커스를 맞추어 GPU 로 구현하여 디코딩 성능을 개선한 결과를 소개하고자 한다. AS, MPIS 등에 대해서는 GPU 에 의한 구현이 아직 진행 중이며 본 구현이 적용된 TMIV 레퍼런스 모델의 버전은 8.0.1 이어서 최신 버전인 11 또는 12 에 바로 적용하기에는 다소 거리가 있겠으나, 본 구현에서 적용된 세부 구현 기술과 서브 모듈 등은 충분한 재활용성을 가지고 있어 다른 방식의 렌더러나 상위 버전의 고속화 구현에도 적용이 가능할 것이다. TMIV 8.0.1 의 디코더에서 1920x4640 크기를 가지는 두 개의 아틀라스를 기준으로 프레임 렌더링의 경우 싱글 프레임 당 약 4 초에서 평균 25ms 이하 로 실행 시간이 단축되어 약 150 배 이상의 성능 향상을 획득하였으며 렌더링 파이프라인의 추가 등에 의해 통상적으로 실시간이라고 여기는 30fps 의 속도로 재생이 가능한 성능에 도달한 결과를 소개하였다.

1. 서론

MPEG 에서는 ISO/IEC MPEG-1 part 12 로 실감형 비디오의 코딩을 위한 표준을 개발하고 있다. MIV 표준은 복수개의 실사 또는 가상 카메라에 의해 촬영된 실영상 또는 가상의 3 차원 장면을 압축하기 위해 개발되고 있다. 기본의 360 도 VR 영상에서 뷰어는 영상의 중심에서 시점의 방향(Orientation)만을 변경하며 컨텐츠를 감상할 수 있는데 반해 MIV 에서는 시점의 방향뿐만 아니라 위치(X, Y, Z)도 촬영된 영상의 뷰 범위 내에서 변경할 수 있는 6DoF 의 플레이백을 지원한다.[1] MIV 표준은 6DoF 의 기능성을 기술 표준문서와 함께 인코더, 디코더, 렌더러 소스와 함께 릴리즈하여 학연산 각 기관의 연구개발자들이 참고하여 기술 표준을 개선하거나 인코더, 디코더 등을 개발하는 데 활용하고 있다. 하지만 기술 표준 자료에서 제공되는 인코더, 디코더, 렌더러는 기능성의 전시에는 문제가 없으나 알고리즘 처리에 있어 프레임당 프로세싱 시간이 상당히 길어 관련 제품의 실용화에는 많은 무리가 따른다. 특히 비실시간으로 동작해도 크게 무리가 없는 인코더보다 HMD, 모니터 등의 디스플레이 장비를 사용하여 실시간으로 컨텐츠를 즐겨야 하는 디코더와 렌더러에서는 확실한 실시간성이 보장되어야 한다. 그러한 실시간성을 보장하기 위해 Julien, Betrand 등은 CUDA, OpenGL 등의 기본 고속화 도구를 사용하여 HMD 에서 MIV 컨텐츠를 재생하는 연구를 수행[2]하였으나 재생 성능 향상의 정도가 수치화되어 정리되어 있지는 않았다.

본 논문은 구성은 다음과 같다. 먼저 2 절에서 MIV 재생 과정의 고속화 연구개발을 수행하면서 CUDA 기반의 GPU 고속화를 위해 진행된 개발 내용을 소개하고, 3 절에서 각 단계별로 소요되는 시간을 고속화 이전화 이후를 RTX-8000, RTX-3090, RTX-A6000 의 GPU 보드 별로 비교한 결과를 정리하고자 한다. 4 절에서 MIV 컨텐츠의 실시간 렌더링을 목표로 진행된 개발 내용을 소개하고, 마지막으로 4 절에서는 MIV 컨텐츠의 실시간 화면 재생을 위해 OpenGL 기반으로 개발된

렌더링 파이프라인에 대해 소개하는 것으로 본 논문에 대한 결론을 맺고자 한다.

2. TMIV 디코더의 GPU 고속화

프레임 렌더러 구조 및 변경 내용

본 논문은 TMIV 8.0.1 을 대상으로 디코더의 고속화를 진행하였으며 해당 레퍼런스 모델의 프레임 렌더러 구조는 그림 1 과 같으며 각 단계의 실행 내용을 간략히 설명하였다.

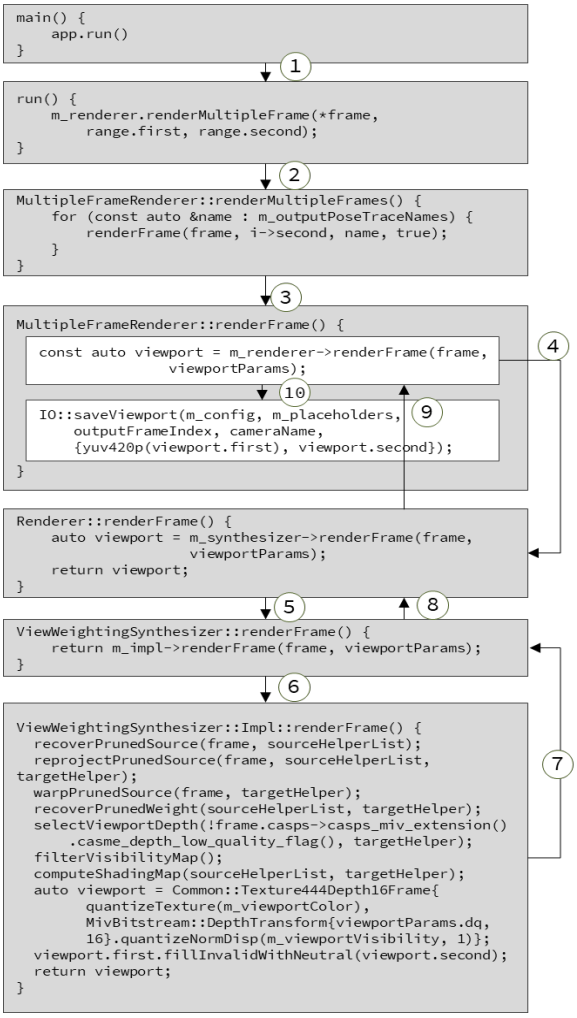

그림 1. TMIV 8.0.1 의 프레임 렌더러 구조

① 주 실행 루프인 run() 실행
② 프레임 Read/Recover 후 다중 프레임 렌더러 실행
③ 각 프레임에 대해 렌더러 실행 함수 호출
④ 지정된 신디사이저인 VWS 방식의 렌더러 실행
⑤ VWS 에 대한 구현 클래스의 렌더러 실행 함수 호출
⑥ VWS 에 의한 렌더링 실행
⑦ ~ ⑨ 현재 뷰포트 파라미터에 대해 디코딩된 뷰포트 반환
⑩ 뷰포트를 YUV 파일로 저장

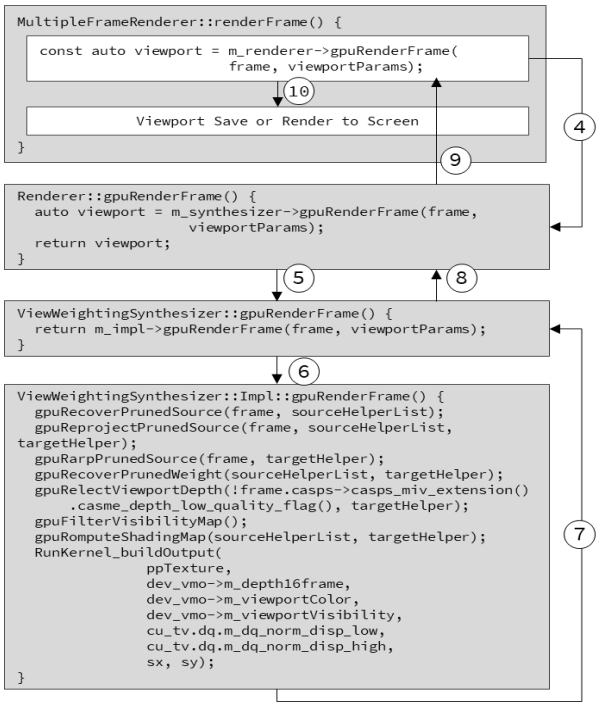

그림 2. GPU 고속화를 위해 변경된 구조

GPU 고속화를 위해 변경된 구조는 그림 2 와 같다. 단계 ①~③까지는 레퍼런스 모델의 흐름을 변경 없이 사용하고 있으며 내부적으로 호출되는 각 함수에 대해 gpu 를 접두어로 하는 동일 기능의 함수를 개발하는 방식으로 고속화 작업을 진행하였다.

④ 지정된 신디사이저인 VWS 방식의 GPU 렌더러 실행
⑤ VWS 에 대한 구현 클래스의 GPU 렌더러 실행 함수 호출
⑥ VWS 에 의한 GPU 렌더링 실행
⑦ ~ ⑨ 현재 뷰포트 파라미터에 대해 디코딩된 뷰포트 반환
⑩ 뷰포트를 YUV 파일로 저장하거나 화면에 직접 렌더링 실행

본 구현에서 적용한 방식은 기존 레퍼런스 모델의 실행 흐름을 거의 그대로 유지한 상태에서 고속화를 진행하기 위해 각 단계마다 CPU 로 구현된 함수들을 GPU 버전의 함수로 대치하는 것이다. 그 경우 레퍼런스 모델의 업데이트에 대응하기 원활하며, 타 연구 개발자들도 어플리케이션에 GPU 버전을 적용하는 작업이 용이하다고 볼 수 있어서이다.

아틀라스 구조에 대응되는 GPU 메모리

레퍼런스 모델에서는 AccessUnit 라는 메인 구조체가 있다. AccessUnit 는 V3cParameterSet, ViewParamList, 아틀라스에 대한 정보를 아틀라스 개수만큼 저장하기 위한 AtlasAccessUnit 의 벡터 등이 주요 멤버로 구성되는데 CPU 에서 할당되는 각 메모리들은 GPU 초기화 시에 동일한 형태로 할당되어야 GPU 파이프라인에서 사용이 가능하며, 본 구현에서는 AtlasAccessUnit 를 통해 할당되는 CPU 메모리를 GPU 에 저장하기 위해 접두어 “Cuda__”를 가지는 구조체 정의

및 각 구조체 형의 인스턴스를 마련해 둔 후 그림 2 에 표현된 과정 내의 모든 GPU 코드에서 접근이 가능하도록 하였다.

```
Cuda__InputFrameMemory dev_ifm;
Cuda__AccessUnit *dev_cu_au = NULL;
Cuda__AtlasAccessUintMemoryObjects
    dev_cu_aaumo[ MAX_ATLAS_COUNT ];
Cuda__PrunedMemoryObject *dev_pmo = NULL;
Cuda__ViewportMemoryObject *dev_vmo = NULL;
Cuda__ViewParams *dev_cu_tv = NULL;
```

3. 고속화 실험 결과

단위 프레임 성능 측정

실험을 하기 위해 사용한 컨텐츠는 그림 3 의 Frog 영상을 사용하였다. 두 개의 아틀라스가 제공되며, 첫 번째는 7 개의 패치, 두 번째는 5 개의 패치로 각각 구성되어 있다.

그림 3. 실험에 사용한 영상

실험 영상에는 전체 17 개의 프레임이 있는데 다음 표 1 에서는 마지막 프레임을 대상으로 CPU 와 세 가지 종류의 GPU 보드에서의 출력 시간을 테이블화한 것으로 매 회의 실행 시마다 수 ms 정도의 편차는 발생하지만 CPU 와 GPU 버전의 큰 실행 성능 차이와 GPU 보드 간 발생하는 약간의 차이는 확인할 수 있다.

	CPU	RTX-8000	RTX-3090	RTX-A6000
recoverFrame	**146 ms**	**34 ms**	**31 ms**	**31 ms**
renderMultipleFrames	**3439 ms**	**35 ms**	**25 ms**	**24 ms**

표 1. CPU 와 GPU 보드별 단계별 실행 시간 측정 결과

GPU 보드 차이에 따른 전체 프레임 성능 측정 비교

이번에는 단위 프레임이 아닌 전체 프레임에 대해 각 GPU 보드별로 성능 측정 결과를 비교하였다. 기본적으로는 단위 프레임에 대한 성능 측정 결과와 유사하며 전체 프레임에 대한 성능을 프레임별로 측정한 결과 전반적으로 RTX-3090 과 RTX-A6000 이 구세대의 RTX-8000 보다는 나은 성능을 나타내고 있다.

Device / Frame	RTX-8000			RTX-3090			RTX-A6000		
	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
1	36	78	114	37	68	105	37	65	102
2	33	32	65	31	24	55	31	22	53
3	31	30	61	33	24	57	31	23	54
4	33	32	65	33	25	58	31	22	53
5	33	31	64	33	25	58	31	23	54
6	36	31	67	34	24	58	29	23	52
7	32	31	63	33	25	58	32	23	55
8	33	32	65	32	24	56	32	23	55
9	31	34	65	33	24	57	94	24	118
10	33	31	64	32	25	57	31	23	54
11	33	35	68	31	25	56	30	23	53
12	33	34	67	33	24	57	31	23	54
13	33	34	67	32	23	55	31	23	54
14	33	33	66	30	23	53	35	23	58
15	31	34	65	32	25	57	31	25	56
16	33	33	66	33	24	57	31	25	56
17	33	34	67	32	25	57	32	25	57

표 2. GPU 차이에 따른 전체 프레임에 대한 성능 측정 비교: (a) recoverFrame, (b) renderMultipleFrame, (c) (a)와 (b)의 합

특이할 만한 사항으로 RTX-A6000 에서만 recoverFrame 단계 시 많은 시간이 소요되는 프레임이 존재하고 있다. RTX-8000 이나 RTX-3090 에서는 발생하지 않는 현상인데 수 차례의 반복된 실행에서도 전체 프레임 중 임의의 프레임 한 개에서 유사한 현상이 RTX-A6000 에서만 발생하고 있으므로 RTX-A6000 으로 MIV 응용 개발 시 유사 현상의 발생 여부에 대해 확인 후 해당 GPU 보드의 사용 여부를 결정해야 한다.

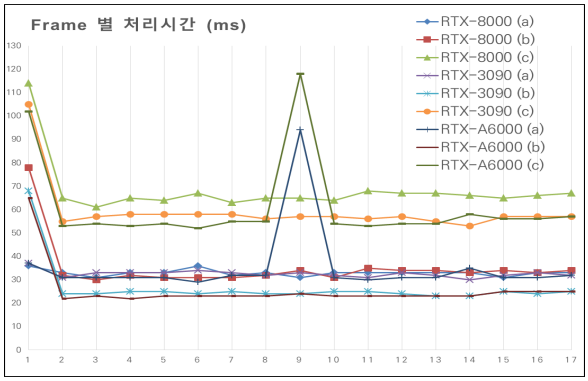

그림 4. GPU 보드 차이에 따른 전체 프레임 성능 측정 결과

CPU 차이에 따른 전체 프레임 성능 측정 비교

이번에는 동일한 GPU 보드에 대해 CPU 가 다를 경우 성능 차이를 비교해 보았다. 이전의 실험에서 성능과 실행 안정성 면에서 가장 우수하다고 평가한 RTX-3090 에 CPU 를 i7 에서 i9 으로 변경한 후 동일한 실험을 진행하였다.

Device / Frame						
	(a)	(b)	(c)	(a)	(b)	(c)
0	37	68	105	66	23	89
1	31	24	55	24	23	47
2	33	24	57	24	23	47
3	33	25	58	24	23	47
4	33	25	58	24	23	47
5	34	24	58	24	23	47
6	33	25	58	31	23	54
7	32	24	56	23	23	46
8	33	24	57	25	23	48
9	32	25	57	24	23	47
10	31	25	56	25	23	48
11	33	24	57	24	23	47
12	32	23	55	24	24	48
13	30	23	53	23	23	46
14	32	25	57	23	23	46
15	33	24	57	25	23	48
16	32	25	57	24	23	47

표 3. CPU 차이에 따른 전체 프레임에 대한 성능 측정 비교: (a) recoverFrame, (b) renderMultipleFrame, (c) (a)와 (b)의 합

표 3 의 결과 수치를 보면 CPU 향상에도 일정 수준의 성능 향상을 얻을 수 있었음을 알 수 있다. 특히, (a)와 (b)의 각 단계별로 최대 33ms 이하가 산술적으로 확보되어야 실시간 렌더링이 가능하다는 점에서 CPU 차이에 따른 성능 향상도 실시간 재생을 위해서 유의미한 소득으로 평가할 수 있을 것이다.

4. 파이프라인 구성에 의한 실시간 전시

아틀라스 입력부터 디코딩 및 디스플레이까지 입력 영상의 프레임율 내에 처리되어야만 실시간 전시가 가능할 수 있다. 따라서 각 모듈을 GPU 로 구현한다고 해도 하나의 쓰레드 만으로 recoverFrame 과 renderMultipleFrame 등의 각 단계를 순차적으로 실행해서는 실시간 전시 성능을 획득하는 것이 불가능하므로 분할 가능한 처리 단위를 영상을 가져오는 GF(Get Frame), 프레임을 구성하는 RF(Recover Frame), 뷰포트 파라미터에 따라 프레임 렌더링을 수행하는 RMF(Render Multiple Frame), 마지막으로 OpenGL 을 사용하여 스크린에 프레임을 렌더링하는 GRF(GUI Render Frame) 모듈로 정의하고, 각 모듈이 실시간을 위해 요구되는 시간 내에 실행되는 지 확인한 후 멀티쓰레드를 구성하고 쿠다 스트림을 활용하여 쓰레드 간 메모리를 독립적으로 운용하도록 한 후 각 쓰레드마다 한 개씩의 재생 파이프라인이 동작하도록 그림 5 와 같이 전체 구성을 추가로 변경하였다.

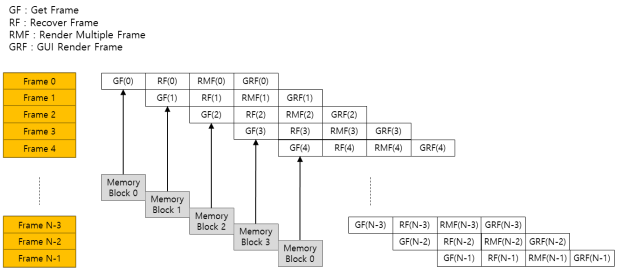

그림 5. MIV 렌더링 모듈의 파이프라인 구성

그 결과로 아틀라스 입력부터 디코딩된 프레임의 화면 출력까지 그림 6 에서 볼 수 있는 바와 같이 30FPS 의 실시간 성능을 달성할 수 있었다.

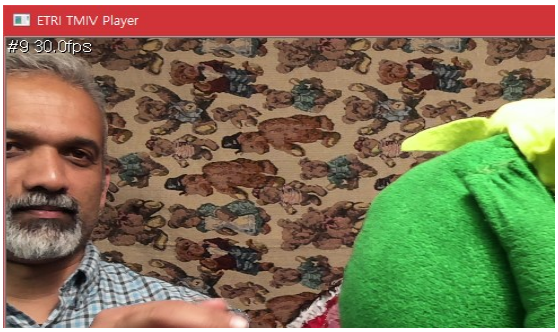

그림 6. MIV 컨텐츠의 실시간 재생 결과

5. 결론

본 논문에서는 TMIV 8.0.1 을 대상으로 recoverFrame, VWS 에 대한 renderMultipleFrame 에 대해 CPU 로 구현된 레퍼런스 모델의 코드를 GPU 코드로 변환 작업의 내용 및 실험 결과에 대해 기술하였다. 특정 프레임에서 과도한 시간이 소요되는 것을 제외하면 전체적으로 볼 때 대부분의 프레임에서 RTX-A6000 의 성능이 RTX-3090 보다는 근소하게 앞서며, RTX-8000 보다는 다소 큰 차이의 우위를 점하고 있음을 실험 결과로 알 수 있었다. 그러나 RTX-3090 은 RTX-A6000 과 비교할 때 거의 동급 수준의 성능을 보이면서 처리 시간 측면에서 더 높은 안정성을 보여주어 실시간 응용의 개발에 더 적합한 것으로 확인되었다. 또한, CPU 의 차이에 따른 성능 차이도 엄연히 존재하여 최신 i9 급의 CPU 적용 시 30fps 라는 실시간 성능을 획득할 수 있었다. TMIV 는 최근 12.0 으로 업데이트되어 해당 버전에 대한 GPU 고속화도 진행되어야 하겠지만 본 논문에서 소개한 고속화 방식 및 실시간 전시를 위한 파이프라인의 구성은 최근 버전에도 큰 수정 없이 충분히 적용할 수 있으며, 3D 공간의 실감 감상, 메타버스 등의 분야에서 다양한 형태의 서비스를 위한 MIV 실시간 디코더 개발에 활용이 가능할 것이다.

감사의 글

이 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No. 2018-0-00207, 이머시브전문연구실)

참 고 문 헌

[1]. Jill M. Boyce, Renaud Doré, Adrian Dziembowski, Julien Fleureau, Joel Jung, Bart Kroon, Basel Salahieh, Vinod Kumar Malamal Vadakital, Lu Yu, MPEG Immersive Video Coding Standard, Vol. 109, No.9, Sep 2021, Proceedings of the IEEE

[2]. Julien Fleureau, Bertrand Chupeau, Franck Thudor, Gérard Briand, Thierry Tapie, and Renaud Doré, AN IMMERSIVE VIDEO EXPERIENCE WITH REAL-TIME VIEW SYNTHESIS LEVERAGING THE UPCOMING MIV DISTRIBUTION STANDARD, 2020 IEEE International Conference on Multimedia & Expo Workshops

[3] 이광순, 정준영, 오관정, 서정일, MV-HEVC 기반 TMIV 에서의 성능 개선, 2021 년 한국방송·미디어공학회 하계학술대회