OpenGL과 Nvidia 영상코덱을 사용한 실시간 자유시점 재생기 설계 및 구현

*구동준 **안희준
서울과학기술대학교
**heejune@seoultech.ac.kr

Design and Implementation of Free-view Player using OpenGL and Nvidia Video Codec

*Gu, Dongjun **Ahn, Heejune
Seoul National Univ. of Science and Technology (SeoulTech)

요약

사용자에게 본인이 원하는 시점과 시각을 선택할 수 있도록 하는 자유시점 (Free Viewpoint) MPEG-I 과제를 통하여 3DOF, 3DOF+, 6DOF의 표준을 개발 중이다. 실사 영상의 자유시점 영상을 구현하는 방법으로는 깊이정보를 사용한 렌더링 기법을 사용하는데, 이를 실시간 재생할 수 있는 시스템은 개발되지 않았다. 본 논문에서는 PC 사양에서 NVIDIA 영상 코덱과 OpenGL사용하는 rtRVSlibrary를 바탕으로, 최대 8개의 HD급 다중 뷰 영상 입력 (컬러+깊이)을 자유 시점을 실시간 생성하여 디스플레이하는 재생기를 설계 및 개발하였다. 사용자는 원하는 시점으로 상하좌우앞뒤(회전)로 자유롭게 이동할 수 있으며, 계산양과 화질 효율성을 고려하여 디코딩한 입력영상 중에 두 개의 시점을 선별하는 알고리즘을 개발하여 실시간 동작 (25fps)을 검증하였다.

1. 서론

기존의 영상 콘텐츠가 제작자의 시점과 시각을 고정하여 제공하는 방식이었다면 최근 몰입형 콘텐트들은 사용자에게 자신이 원하는 시점과 시각을 선택하는 자유 시점 (Free-view point)을 제공하는 방향으로 발전하고 있다. 캐릭터나 환경 모델에 기반하는 게임 등에서는 이미 HMD와 사용자의 움직임을 바탕으로 자유 시점을 제공하는 응용들이 상용화 단계에 들어서있다. 그러나 실사 영상을 사용하는 방송이나 엔터테인먼트 영상들을 사용한 자유시점 콘텐츠는 해결해야할 기술적인 문제들로 인하여 아직 표준화와 보급이 일반적이지 않은 상황이다. 이를 위하여 핵심 해결 기술로는 입력영상의 깊이 추정 기법, 깊이 정보의 효과적인 압축 방법, 컬러와 깊이정보를 바탕으로 한 효과적인 자유시점 뷰 합성 기법을 들 수 있다. 현재 MPEG-I (Imersive)[1]는 실감영상포맷을 표준화하는 MPEG의 하부조직으로 최근 3DOF, 3DOF+, 6DOF 영상압축과 포맷 등에 대하여 표준화를 하고 있다.

그러나 이렇게 다수의 입력 뷰가 주어졌을 때 사용자가 원하는 시점을 합성하는 시스템에 대한 연구는 상대적으로 활발히 이루어지고 있지 않다. 논문 [2] 정도가 발표된 몇 안 되는 국내 연구라고 판단된다. 이 연구조차도 대부분 UI 설계와 주어진 콘텐트에서 시각을 바꾸어 표출하는 정도의 수동적인 기능에 제한되어 있다. 시점 변화를 제공하는 기능인 '오브젝트 뷰'는 Unity 기반의 3D 모델을 사용한 게임 애니메이션의 경우로 제한되어, 다시점 실사영상을 깊이기반 으로 실시간 합성하는 기능을 제공하지 못한다. 이는 그 동안의 연구에도 불구하고 연구자들이 사용할 수 있는 렌더링 라이브러리가 개발되어 있지 않기 때문이다. 본 논문에서는 본 연구실에서 개발한 rtRVSlibrary [3]을 바탕으로 실시간 뷰합성이 가능한 플레이어를 설계 개발하였다. 개발된 플레이어는 최대 8개까지의 HD급 입력뷰의 컬러와 깊이영상을 동시에 디코딩하고 이들 중 출력 뷰 생성에 적합한 두 개의 뷰를 자동으로 선택하여 합성한다. 구체적으로 개발된 소프트웨어 플레이어는 일반 PC 환경에 Nvidia Titan Xp 그래픽카드 사양에서 사용자의 시점이동 요구에 따라 자유 시점를 실시간 (25fps) 합성하여 출력한다.

2. 실시간 Free-viewpoint Player

설계한 자유시점 재생기는 크게 3개의 블록으로 구성된다. 첫 번째는 입력 뷰들을 디코딩하는 다중 뷰 디코더로 모든 입력뷰의 컬러영상 파일과 깊이 영상파일을 디코딩한다. 뷰 합성기 블록은 OpenGL을 사용하여 2개의 입력을 입력뷰와 출력뷰를 고려하여 합성을 한다. 2개만을 사용하는 이유는 현재 rtRVSLibary와 PC 그래픽 카드의 성능상 2개 이상은 실시간 처리가 어렵기 때문이다. 입력 처리부는 사용자 입력을 받아서 원하는 출력뷰로 뷰 파라미터를 조정하는 기능을 제공한다. 이때

가장 적합한 입력뷰를 선택한다. 그래픽 카드의 디코더 모듈과 그래픽 처리 모듈은 자원이 분리 되어 있기 때문에 동시에 동작이 가능하다. 다중 뷰 디코더와 렌더러 사이는 동기 큐로 구성하였고, 시스템 자원 효율성 각 디코더 들와 렌더러는 별도의 쓰레드로 구성하였다.

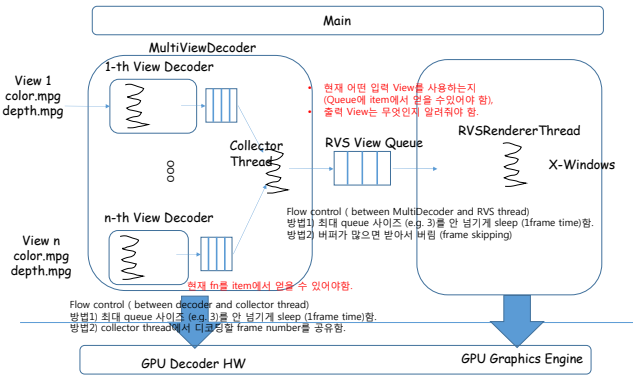

그림 1. 플레이어 전체 구조

2.1. 실시간 부 합성 라이브러리 (rtRVSlibrary)

본 연구에서 사용된 다중뷰 합성 기능은 본 연구실에서 개발한 rtRVSlibrary [3]을 사용하였다. rtRVSlibrary는 MPEG-I 그룹에서 표준 테스트 툴로 제공된 RVS[4]를 OpenGL 사용 기법을 최적화하여 실시간(realtime)이 가능하도록 수정한 것이다. RVS는 2018년도에 채택된 MPEG-I의 가상 시점 합성 표준 소프트웨어로, RVS는 메쉬표면방식의 기법을 사용하여. 기존의 화소기반의 방식에 비하여 성능이 높다. RVS는 깊이 정보와 화소 정보를 표면 (surface)로 모델링하고 이를 바탕으로 그래픽스의 기하 변환 방식을 사용하는 알고리즘으로 기존의 방식보다 화질 면에서 2.5dB 이상 높은 것으로 보고되고 있다[4]. rtRVSibrary는 그림 2와 같이 OpenGL 사용을 최적화함으로써 두 개의 2k 해상도의 영상 합성에 1.0fps 이하의 속도를 보이던 RVS 구현을 30 fps이상의 되도록 가속하였다 (GTX1070에서 Titan Xp로 그래픽 카드가 바뀌면서 절대적인 성능이 추가로 향상됨).

2.2. 다중 입력뷰 디코더 (Multi-view Decoder)

PC 환경에서도 다수의 HD급 비디오 스트림을 실시간으로 디코딩하는 것은 소프트웨어적으로는 제약이 따른다. 본 연구에서는 4개에서 최대 12개 정도의 입력을 스트림을 동시 디코딩을 필요로 한다. 이를 위하여 NVIDIA의 video codec SDK [5]를 사용하였다. GTX Titan-XP 사양에서 2k급 영상을 20개 정도 까지 동시에 디코딩이 가능한 것으로 보장하고 있다 (Pascal P2000, 2K 648fps).

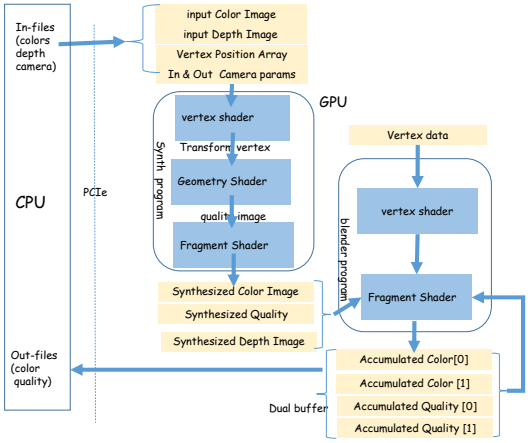

그림 2. rtRVSLibrary (OpenGL) 구조

'멀티 디코더' 클래스는 입력시점의 개수 만큼 싱글 디코더 클래스 객체를 보관하며, 싱글 디코더 클래스는 한 시점에서 촬영되는 컬러 영상과 깊이 영상을 처리하기 위한 두 개의 영상 디코더 클래스 객체를 포함한다. 각 디코더 객체는 멀티디코더의 제어를 받아 초기화되고 독립적인 스레드를 사용하여 NVIDIA Video SDK 를 사용하여 컬러 영상과 깊이 영상을 디코딩 작업을 큐에 디코딩된 프레임(YUV420p 포맷)으로 저장한다. 멀티 디코더는 큐에서 원하는 뷰의 프레임을 추출한 후 OpenGL에 맞도록 정규화 및 OpenCV의 matrix 포맷으로 변환하고 렌더러와 연결된 큐에 넣는다.

2.3. 입력 선택 알고리즘

일반적으로 합성시에 입력으로 주어지는 모든 입력을 사용하는 것은 계산시간을 증가시킨다. 또한 실험결과 두 개의 입력만 적절히 선택한다면 합성영상의 화질은 크게 향상되지 않는 것을 확인하였다. 이러한 이유로 현재 요구되는 출력뷰의 합성에 효과적인 입력뷰를 선택하는 알고리즘을 개발하였다.

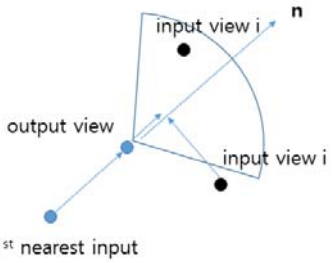

그림 3. 출력 뷰를 고려한 입력 뷰 선택 알고리즘

여러 개의 입력 view 중 가상 시점에서 가장 가까운 시점의 view를 첫 번째 입력 view로 선택한다. 그 다음 첫 입력 view를 제외한 가상 시점과 일정 거리 내에 있는 view들이 가상 시점과 이루는 벡터를 구한 다음, 첫 번째 입력 view랑 가상시점이 이루는 벡터와 코사인 유사도가 가장 높은 벡터의 view를 두 번째 입력 view로 선택한다.

2.4. 사용자 인터페이스

사용자 인터페이스는 Linux 상에서 Xlib를 직접 적용하여 구현하였다. 현재는 화면구성은 따로 없이 영상표출만 가능하다. 입력은 키보드, 마우스 드래그 입력과 휠-마우스 기능을 사용하여 일시정지 기능 및 ERP 영상에 대해서는 시점의 전-후 이동 및 yaw와 pitch 축에 대한 회전이 가능하도록, Perspective 영상에 대해서는 시점의 전-후-좌-우-상-하 이동이 가능하도록 구현되었다. 향후 보다 편의적인 화면 구성과 입력을 지원할 계획이다. 이와 관련되어 [2]에서는 다양한 화면노드를 제안하고 있으며, 또한 오디오 출력도 지원하고 있다.

3. 시험 결과

3.1. 컨텐츠

본 실험은 MPEG-I에서 제공하는 Classroom ERP 영상과 Technicolor- Painter 평면투영 영상을 사용하였다. Classroom 영상은 120 프레임의 v0부터 v14까지 총 15개로 구성중에 그림과 같이 7개를 선택하여 사용하였으며, Painter 영상은 300 프래임으로 v0에서 v15의 16개중에서 8개를 선택하여 입력으로 사용하였다. 동일한 해상도를 갖도록 Classroom영상은 4k (4096x2048) 10bit 영상을 2k (2048x1024) 8bit영상으로 변환하고, TechnoPainter 영상은 2048x1088 10bit영상을 2048x1024 8bit으로 변환한 후, color 영상과 depth 영상 모두를 ffmpeg을 사용하여 h264 mp4 파일형태로 변환하였다 (target bit-rate 2Mbps)

3.2. 동작 환경

컴퓨터 환경은 전형적인 개발용 리눅스 PC 환경정도의 사양 (AMD 라이젠7 (3700X Turbo 4.4G) 에 메모리 16G, NVIDIA GTX Titan-Xp, Ubuntu 18.04.5 LTS 환경)에서 테스트 하였다. Nvidia 드라이버는 버전 450.66 을 사용하였고, Nvidia Codec SDK 9.0.0, OpenGL 4.2에서 사용하였다.

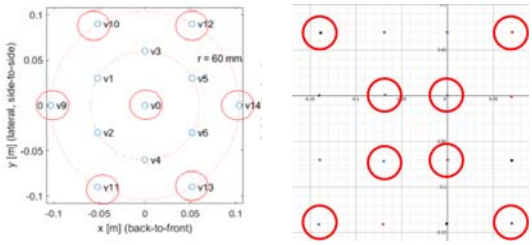

그림 4. 입력 뷰 포인트 위치 (좌: Classroom, 우: TechnoPainter)

3.3. 실험 결과

그림 5. 자유시점 출력 결과 (ClassRoom ERP) 상: 일반 위치, 중 : 가상 시점을 입력 뷰에서 하단으로 이동시킨 경우, 하: 시점을 입력 뷰의 좌측으로 이동시킨 경우

그림 6. 자유시점 출력 결과 (Painter, perspective) 상: 일반 위치, 중 : 시점 이동 및 줌-인 한 경우, 하: 시점 이동 및 줌-아웃 한 경우

그림 5의 위는 Classroom ERP영상을 Perspective 모드로 투영한 것으로 화면 내에 3차원 공간처럼 그려진다. 중간과 마지막 그림은 가상시점을 입력 view와 떨어진 위치로 이동시켰을 때의 모습이다. 그림 6은 Painter 평면 영상을 사용한 경우이며 중간과 마지막 그림은 각각 줌-인, 줌-아웃 한 경우이다. 깊이 정보가 완벽하지 않은 영역 혹은 입력 view에서 다른 물체에 의해 가려진 영역에서는 합성된 영상의 퀄리티가 불안정해진다. 또한 표출하고자 하는 픽셀의 정보를 선택된 입력 view에서 찾을 수 없는 경우 그 픽셀은 녹색 픽셀로 나타난다. 이런 부분을 시각적으로 완화하기 위해서는 inpainting 기법을 OpenGL 에 포함하여야할 것으로 보인다.

4. 결 론

가속화된 rtRVSLibrary와 Nvidia video codec을 사용하여 다시점 영상을 이용한 자유시점 플레이어의 핵심기능을 구현하고 일반적인 PC 환경에서 실시간 동작이 가능함을 확인하였다. PC 성능에 적합하도록 출력뷰에 맞게 입력뷰에서 선택하는 알고리즘을 추가 개발하였다. 이를 바탕으로 한 다양한 응용과 시연이 가능하리라고 예상된다.

제약사항 및 향후 개발사항으로는, 우선 현재 깊이 영상을 별도의 파일도 일반 컬러영상의 압축 방식을 사용하였고, 오디오와 통합 및 전송 방식으로 확장 등이 필요하다. 이를 위하여 다중뷰 영상을 전송하기 위한 표준 방식을 정의할 필요가 있다.

감사의 글

본 연구는 과학기술정보통신부, 정보통신기술진흥센터, 방송통신산업기술개발사업의 지원을 받았음. (시청자 이동형 자유시점 360VR 실감미디어 제공을 위한 시스템 설계 및 기반기술 연구(2016-0-00144).

참고 문헌

1. G. Lafruit, D. Bonatto, C. Tulvan, M. Preda and L. Yu, "Understanding MPEG-I Coding Standardization in Immersive VR/AR Applications," in SMPTE Motion Imaging Journal, vol. 128, no. 10, pp. 33-39, Nov.-Dec. 2019

2. 양지희, 송민기, 박구만, “사용자 선택에 따른 자유 시점 비디오 서비스 기반의 통합 플레이어 시스템 구현,” 방송공학회 논문지, 제25권 제2호, 2020.

3. 안희준, 이명진, “실시간 렌더링을 위한 MPEG-I RVS 가속화 기법 ,” 방송공학회논문지, 제25권 제4호, 2020.

4. S. Fachada, D. Bonatto, A. Schenkel and G. Lafruit, “Depth image based view synthesis with multiple reference views for virtual reality,” 3DTV-CON), Helsinki, 2018.

5. NVIDIA, “Video Codec SDK- Decoder, Application Note (NVDEC_DA-08097-001_v07)”, Aug. 2019