Journal of Korea Multimedia Society Vol. 24, No. 4, April 2021(pp. 538-549)
https://doi.org/10.9717/kmms.2020.24.4.538

다중 가상 카메라의 실시간 파노라마 비디오 스트리밍 기법

옥수열†, 이석환††

Real-Time Panoramic Video Streaming Technique with Multiple Virtual Cameras

Sooyol Ok†, Suk-Hwan Lee††

ABSTRACT

In this paper, we introduce a technique for 360-degree panoramic video streaming with multiple virtual cameras in real-time. The proposed technique consists of generating 360-degree panoramic video data by ORB feature point detection, texture transformation, panoramic video data compression, and RTSP-based video streaming transmission. Especially, the generating process of 360-degree panoramic video data and texture transformation are accelerated by CUDA for complex processing such as camera calibration, stitching, blending, encoding. Our experiment evaluated the frames per second (fps) of the transmitted 360-degree panoramic video. Experimental results verified that our technique takes at least 30fps at 4K output resolution, which indicates that it can both generates and transmits 360-degree panoramic video data in real time.

Key words: Virtual Camera, Panoramic Video Streaming, RTSP Streaming, 360-degree VR

1. 서 론

최근 컴퓨팅, 그래픽스, 및 네트워킹 기술의 급격한 발전과 더불어, 4K, 8K급의 높은 대역폭을 요구하는 2D 비디오 스트리밍뿐만 아니라 3D 또는 360VR (Virtual Reality) 비디오를 스트리밍할 수 있는 멀티미디어 서비스 가상 환경이 빠르게 진화하고 있다. 이와 같은 가상 환경에서 실시간 360도 파노라마 비디오 스트리밍 전송 기술에 대한 연구가 많이 진행되고 있다[1-6]. 그러나 가상 환경 서비스를 제공하는 VR HMD(Head Mount Display) 장치의 스트림 대역폭으로 인하여, 8K 이상의 초고해상도 비디오 스트리밍에는 전송 제한이 있다.

360도 파노라마 비디오 영상은 넓은 화각(FOV, Field of View)과 HMD(Head Mounted Display)에 의한 높은 몰입감(Presence)과 임장감(Immersion)을 가지고 있다. 최근 3D VR 시장에서는 넓은 화각과 높은 몰입감 및 임장감을 가지는 360도 파노라마 비디오영상 스트리밍 서비스에 대한 관심이 증가하고 있다. 특히, youtube, facebook, instagram과 같은

※ Corresponding Author: Suk-Hwan Lee, Address: (49315) 550beongil 37, NakdongDaero, Saha-Gu, Busan Korea, TEL: +82-51-200-7782, FAX: +82-51-200-7783, E-mail: skylee@dau.ac.kr
Receipt date: Mar. 11, 2021, Revision date: Apr. 15, 2021
Approval date: Apr. 19, 2021
† Dept. of Computer Engineering, Dong-A University (E-mail: sooyol@dau.ac.kr)
†† Dept. of Computer Engineering, Dong-A University
※ This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2020R1F1A1069124) and also by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2020-0-01797) supervised by the IITP (Institute of Information & Communications Technology Planning & Evaluation)"

소셜 미디어 플랫폼(Social Media Platform)에서는 360도 전용 카메라를 제작하거나, 360도 영상을 시청할 수 있는 뷰어를 제공하고 있다.

360도 파노라마 비디오 데이터 생성은 기본적인 이미지 스티칭(stitching)[7-10]에 의하여 이루어진다. 이미지 스티칭은 가상 및 실제 다중 카메라로부터 이미지 영상을 입력으로 받아 여러 개의 이미지를 변환하여 이어붙인 결과를 지구본을 펼친 듯한 형태로 투영(Equirectangular Projection)하여 하나의 프레임에 담는다. 이미지 스티칭에서 다중 카메라의 시점 변화와 조명의 위치 및 세기가 변화더라도 식별이 가능한 특징점들이 추출되어야 한다. 다중 카메라 입력과 넓은 화각으로 360도 파노라마 비디오 영상의 해상도는 크게 높아진다. 예를 들어, RGBA 채널의 4K 해상도(3840×1920)에서 한 프레임의 데이터 크기는 약 28MB(mega byte)이고, HMD의 Stereo 영상에서 데이터 크기는 프레임 당 약 56MB이다. 이와 같이 다중 카메라로부터의 스티팅 과정과 고해상도의 데이터로 인하여 660도 파노라마 비디오 영상을 실시간 스트리밍하는 것은 매우 어렵다.

최근 딥러닝 학습에서 부족한 학습 데이터 문제를 해결하기 위하여, 가상 세계에서 학습 데이터들을 생성하는 연구가 진행되고 있다. 컴퓨터 시뮬레이션을 통한 가상 세계는 현실에서 재현하기 어렵거나 불가능한 사례를 만들어낼 수 있으며, 필요에 따라 얼마든지 반복해서 실행할 수 있다. 특히 그래픽 기술의 발달로 인하여, 가상 세계의 그래픽은 현실과 다를 바 없는 수준에 이르렀다. A. Prakash 등[11]은 자율주행 및 물체 검출 분야에서 필요한 학습 데이터를 가상 세계에서 획득하는 방법을 제안하였다. 이들은 현실 기반 학습 데이터와 가상 기반 학습 데이터를 Faster-RCNN[12]으로 학습하였으며, 실험을 통하여 기존 대비 물체 인식률이 향상됨을 확인하였다.

본 논문에서는 가상 환경과 실제 환경에서 실시간 360도 파노라마 비디오 스트리밍 전송 방법을 제안한다. 제안한 방법은 GPU 기반의 텍스처와 엔코딩에 의하여 가상 세계의 학습 데이터 영상을 생성하고, GPGPU(General-Purpose computing on Graphics Processing Units) 기반으로 ORB 특징점 추출에 의한 360도 파노라마 영상을 생성한다. 그리고 고해상도의 360도 파노라마 영상을 H.264 코덱으로 압축한 다음, RTSP(Real Time Streaming Protocol) Server를 통하여 실시간 360도 파노라마 영상을 전송한다. 1232×1024 해상도의 6대 카메라를 사용한 실험결과로부터, 제안한 방법이 HMD 스테레오(Stereo) 송출 시 2K 해상도에서 45fps, 4K 해상도에서 33fps으 측정되어, 실시간 360도 파노라마 비디오 생성 및 전송이 가능함을 확인하였다.

본 논문의 구성은 다음과 같이, 2장에서는 다중 가상 카메라 기반의 실시간 영상 생성 방법에 대하여 살펴본 다음, 3장에서는 360도 파노라마 비디오 영상 생성, 인코딩, 및 RTSP에 의한 스트리밍 전송 방법에 대하여 살펴보도록 한다. 4장에서는 제안한 방법의 성능 평가를 분석하고, 마지막 5장에서는 본 논문의 결론과 향후 과제에 대하여 논의하도록 한다.

2. 다중 가상 카메라 비디오 생성 기법

본 장에서는 CUDA(Compute Unified Device Architecture) 기반 텍스처 변환과 GPU 가속화 비디오 엔코딩[13] 기반으로 가상 세계의 다중 가상카메라로부터 360도 파노라마 비디오 생성 방법에 대하여 살펴보도록 한다.

제안한 방법에서는 플랫폼 종속적인 그래픽스 라이브러리가 사용된 부분을 캡슐화하였고, 이를 고수준 API로 제공하는 RHI(Rendering Hardware Interface) 구조의 Unreal Engine 4를 사용하여 다중 가상 카메라로부터 영상을 생성한다. 여기서 RHI는 엔진 내부에서 렌더링 관련 기능을 구현하는 데 사용되며, RHI에 의하여 생성된 텍스처인 RHITexture는 그래픽스 라이브러리로 구현된 텍스처 정보를 담고 있다.

GPU 가속 인코딩 중의 하나인 NVENC[14]는 엔비디아용 그래픽 카드에 올라가는 하드웨어 코덱으로, 그래픽 코어가 아닌 영상 처리 코어들을 조작한다. NVENC의 주요 특징은 실제 GPU 성능에 별다른 손실을 주지 않으면서 비디오 영상을 엔코딩할 수 있고, GPU와 메모리를 공유하기 때문에 그래픽스 라이브러리에서 생성된 텍스처를 곧바로 비디오 영상으로 엔코딩할 수 있다는 것이다. 제안한 방법은 NVENC을 사용하여 그래픽스 라이브러리에서 생성된 텍스처를 실시간으로 엔코딩한다.

제안한 방법은 HEVC 코덱으로 Ultra Video Group [15]에서 제공하는 3840×2160 해상도, 120프레임율, 10비트의 색상 깊이의 시퀀스을 사용하여, CPU

(Ryzen Threadripper 1900x, 4.0Ghz, 16 thread)와 NVENC(Geforce RTX 2080TI, 4,352 CUDA core)의 비디오 엔코딩 성능을 테스트하였다. CPU는 프레임당 처리 시간이 평균 288ms로 초당 약 3.5 프레임을 처리하였으나, NVENC는 프레임당 처리 시간이 평균 30ms로 초당 약 33 프레임을 처리하였다. 비교 실험을 통하여 CPU에 비하여 NVENC이 약 9.3배 높은 프레임을 처리하는 것을 확인하였다.

제안한 방법은 Fig. 1에서와 같이 CUDA 사용하여 GPU 메모리에 접근하는 NvEncoderCuda 라이브러리를 사용하여 RHI 기반 텍스처를 NVENC 기반으로 비디오 영상으로 엔코딩한다. 제안한 NVENC 비디오 엔코딩은 엔코더 초기화가 완료된 후, 3가지 모듈을 수행한다. 첫 번째 Insert Frame 모듈에서는 CUDA를 사용하여 RHI 기반 텍스처에 접근한 다음, 이 텍스처를 별도의 메모리에 저장한다. 두 번째 Convert Frame 모듈에서는 GPU 기반으로 Insert Frame 모듈에서 저장된 텍스처의 색상 포맷을 변환한다. 마지막으로 Encode Frame 모듈에서는 색상 포맷 변환된 텍스처를 비디오 영상으로 엔코딩한다.

NVENC 다중 엔코더 세션은 NVIDIA 그래픽 드라이버에 포함된 DLL/SO에서 NvEncOpenEncodeSessionEx 함수를 호출하여 생성한다. 여기서 NVENC 엔코더 세션의 개수는 NVIDIA 그래픽 드라이버에 의하여 자동으로 관리된다. 그리고 하드웨어만 지원할 경우에는 다중 엔코더 생성을 위한 별도의 설정이 필요하지 않다. 제안한 방법은 Fig. 2(a)에서와 같이 Unreal Engine 4 기반 가상 세계의 가상 카메라를

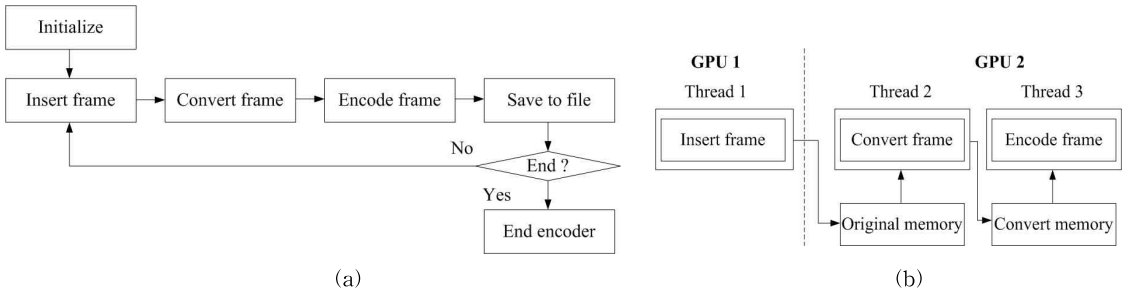


Fig. 1. (a) Propsed NVENC based video encoding process and (b) GPU parallelization of 3 modules.

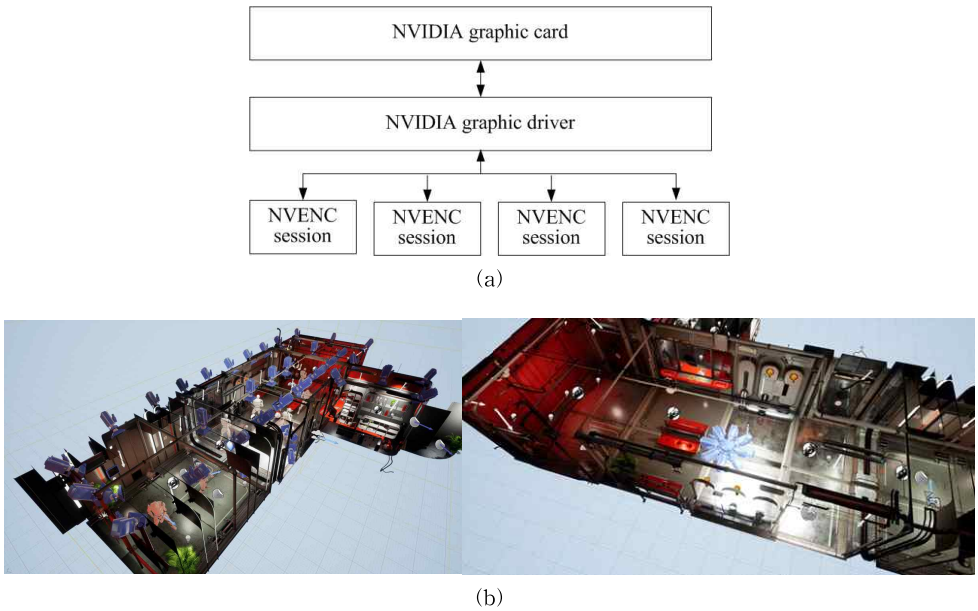


Fig. 2. (a) Structure of NVENC sessions and (b) virtual 360-degree cameras that can be freely positioned.

여러 개 생성하여, NVENC 다중 엔코더를 연결함으로써, 가상 세계 다중 엔코더를 생성한다. Fig. 2(b)는 4대의 가상 카메라로부터 NVENC 다중 엔코더된 영상을 보여준다.

3. 제안한 가상 카메라 기반 360도 파노라마 비디오 스트리밍

본 장에서는 앞 장에서 제안한 다중 가상 비디오 영상 생성 방법을 기반으로 가상 및 실세계 360도 파노라마 비디오 생성 및 스트리밍 기술에 대하여 살펴보도록 한다. 제안하는 360도 파노라마 비디어 생성 및 스트리밍 시스템의 구조는 Fig. 3에서와 같다. 제안한 시스템에서는 가상/실제 360도 카메라로부터 다중 비디오 영상을 읽는 다음, 해당 다중 영상에 연결된 카메라 속성값을 설정한다. 이때 가상/실제 360도 파노라마 비디오 영상은 GPU 기반의 고속 병렬 기반의 스티칭 과정에 의하여 생성된다. 최종적으로 정합된 비디오 영상에 대한 정보를 수정하거나 동영상 혹은 스트리밍 데이터 결과물로 도출한다.

3.1 다중 가상 카메라 기반 360도 파노라마 영상 생성

제안한 360도 파노라마 스트리밍 영상 생성 및 전송 과정은 GPU 기반 고속의 파노라마 영상 생성과 고품질 360도 파노라마 영상 획득을 위한 360도 카메라 보정, 및 실시간 스트리밍을 위한 H.264 압축 과정으로 나누어진다.

3.1.1 특징점 기반 파노라마 이미지 스티칭

첫 번째 단계는 GPGPU(General-purpose computing on graphics processing units)기반으로 다중 카메라별로 입력받은 영상들에 대한 파노라마 이미지 스티칭 과정을 통하여 360도 파노라마 비디오의 프레이별 영상을 생성한다. 하나의 카메라로부터 초당 30 프레임율로 영상을 획득할 경우, n개의 다중 카메라로부터 입력되는 영상은 초당 30xn 프레임율로 모든 과정들이 처리되어야 한다. 따라서, 제안한 방법은 실시간 처리를 위하여 GPU를 계산용도로 활용하여 GPGPU 기반으로 다중 카메라 360도 파노라마 영상 처리를 수행한다.

여러 개의 영상을 하나로 합치는 이미지 스티칭은 다중 카메라별로 입력받은 영상들에 대하여 영상 보정, 특징점 추출, 영상 등록, 블렌딩 등의 과정으로 수행된다. 이미지 스티칭의 주요 과정은 다음과 같다.

특징점 추출은 이미지 스티칭에서 가장 기본이 되는 과정으로, 이후의 특징점 매칭, 호모그래피 변환, 이미지 정렬 과정에서 필요한 정보이다. 정확한 파노라마 스티칭 이미지를 얻기 위하여, 카메라 시점, 조명 등의 외부 변화에도 견실하고 강인한 특징점과 이들 정보를 담을 수 있는 구조인 식별자(descriptor)가 필요하다. 제안한 방법은 Wang의 방법[16]과 같이 높은 반복성과 강인성을 가지면서도 빠른 속도로 검출하기 위하여 ORB(Oriented FAST and Rotated BRIEF) 알고리즘[17]을 사용한다. 제안한 방법에서 사용된 다중 카메라 상의 ORB 기반 특징점 검출 과정은 다음과 같다.

n개 카메라들 중 k번째 카메라로부터 입력받은 i번째 프레임 영상 $I_{ik}(x,y)$일 때

1. 먼저, FAST로 특징점을 찾고, Harris corner detection에서 사용하는 코너에 대한 정량적인 값을 기준으로 가장 코너성이 큰 N개의 코너를 선택한다.
2. SIFT에서 스케일 피라미드를 만들어 똑같이 특징점을 추출하고 여러 스케일에 대한 특징점을 찾는다.
3. 다음으로 찾아낸 코너의 방향을 알기 위해서, 한 윈도우 안에서 Intensity Centroid을 계산한

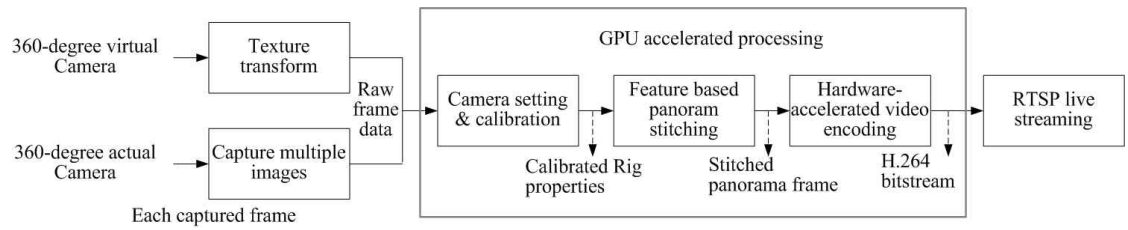


Fig. 3. Proposed method for virtual 360-degree panorama video streaming system.

다. 특징으로 추출된(코너) 픽셀을 중심으로 윈도우를 형성하고, 중심에 있는 코너로 부터 계산한 Intensity Centroid의 방향이 코너의 방향성을 대변한다.

4. 특징자 방향에 따른 steer-BRIEF라는 설명자(descriptor)을 사용하여, 평균은 그대로 두고, 높은 분산값을 가지게 하는 과정을 거쳐 rBRIEF를 얻는다.
5. multi-probe LSH(Locality Sensetive Hashing)를 사용하여 설명자 매칭을 수행한다.

위의 과정을 통하여 i번째 프레임 상에 n개 영상들에 대하여 특징점들을 검출한다.

제안한 방법은 Brown 등의 방법[8] 기반으로 RANSAC (RANdom SAmple Consensus) 알고리즘[18,19]을 사용하여 다중 카메라와 관련된 기본 메트릭스들을 추정하고, 영상 특이점들의 대응 문제를 해결한다. 이는 특이점(outlier)을 포함할 수 있는 관측된 데이터 세트로부터 수학적 모델에 적합한 견고한 파라미터들을 반복적으로 추정하는 것으로, 어떤 특정 확률로만 가지는 합리적 결과를 산출한다는 의미에서 비결정적이다. 이 알고리즘의 기본적인 가정은 데이터는 어떤 수학적 모델로 정의될 수 있는 분포를 가지는 "inliers"와 모델에 맞지 않는 데이터인 "outliers"로 구성된다는 것이다. outliers는 잡음, 잘못된 측정, 또는 단순히 부정확한 데이터로부터 발생되는 포인트로 간주된다. 호모그래피 추정에 대하여, RANSAC은 몇 가지 포인터 쌍을 사용하여 여러 모델을 맞추고, 모델이 대부분의 점을 연관시킬 수 있는지 확인한다. 제안한 방법에서 사용되는 RANSAC 이용한 자동 호모그래픽 예측 방법은 다음과 같다.

1. 샘플의 개수 N을 선택한다.
2. 미리 정의된 개수만큼 랜덤한 잠재적 매칭점들을 선택한다.
3. 정규화된 DLT를 사용하여 H을 계산한다.
4. 각 잠재적 매칭 쌍에 대하여 포인터들을 x에서 x'으로 투영한다; $x_i^{'} = Hx_i$.
5. 투영된 거리가 t(예, t=3 픽셀)보다 작은 포인터들을 카운트한다.
6. 2-5단계를 N번 반복하여 가장 많은 inlier를 가지는 H를 선택한다.

카메라의 광학적 결함에 의하여 이상적인 렌즈 모델과 사용된 카메라-렌즈 조합 간의 차이가 나타나며, 이로 인하여 이미지 간의 왜곡 및 노출 차이가 발생된다. 영상 보정은 이 차이를 최소화하는 카메라 보정 매트릭스을 찾는 것이다. Camera Matrix는 카메라의 초점거리(Focal Length), 주점(Principal Point), 왜곡 계수(Distortion Coefficient) 및 카메라의 이미지 센서 등의 내부 요소인 Intriniscs와 카메라의 위치(Translation)나 회전(Rotation)의 외부 요소인 Extrinsics으로 구분된다. 그러므로 제안한 방법은 M. Brown의 방법[8]을 이용하여 다중 영상을 보정한다.

각 카메라들은 Intrinsic의 회전벡터 $\boldsymbol{\theta}=[\theta_1,\theta_2,\theta_3]$와 Extrinsic의 초점거리 f에 의하여 변수화된다. 즉, 카메라 행렬은 Intrinsic 행렬 $\boldsymbol{K}_i$와 Extrinsic의 회전행렬 $\boldsymbol{R}_i$에 의하여 정의된다.

$$\boldsymbol{K}_i = \begin{bmatrix} f_i & 0 & 0 \\ 0 & f_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \boldsymbol{R}_i = e^{[\boldsymbol{\theta}]_\times} \tag{1}$$

$$\text{where } [\boldsymbol{\theta}_i]_\times = \begin{bmatrix} 0 & -\theta_{i3} & \theta_{i2} \\ \theta_{i3} & 0 & -\theta_{i1} \\ -\theta_{i2} & \theta_{i1} & 0 \end{bmatrix}$$

$\boldsymbol{u}_i$가 영상 위치이고, 이 영상의 동차 좌표(homogeneous coordinate) 위치가 $\hat{\boldsymbol{u}}_i$, $\hat{\boldsymbol{u}}_j$일 때 ($\hat{\boldsymbol{u}}_i = s_i[\boldsymbol{u}_i,1]$), 쌍별 호모그래피 변환은

$$\hat{\boldsymbol{u}}_i = \boldsymbol{H}_{ij}\hat{\boldsymbol{u}}_j \text{ where } \boldsymbol{H}_{ij} = \boldsymbol{K}_i\boldsymbol{R}_i\boldsymbol{R}_j^T\boldsymbol{K}_j^{-1} \tag{2}$$

와 같다. 이상적으로 ORB 특징점들이 이와 같은 변환 행렬에 강인하나, 영상 위치에서의 작은 변화일 때, $\boldsymbol{u}_i = \boldsymbol{u}_{i0} + \frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{u}_j}|_{\boldsymbol{u}_{i0}} \Delta \boldsymbol{u}_j$ 또는 동등하게 (equivalently) $\boldsymbol{u}_{i0}$에 대한 호모그래픽을 선형화함으로 얻어진 어파인 변환 $\boldsymbol{A}_{ij}$으로 $\hat{\boldsymbol{u}}_i = \boldsymbol{A}_{ij}\hat{\boldsymbol{u}}_j$ 표현된다. 특징점들이 모든 영상들로부터 추출된 후 이들은 매칭이 된다.

영상 간에 기하학적으로 일치하는 일치 집합이 주어지면, 번들 조정 (bundle adjustment)을 사용하여 모든 카메라 매개 변수를 조정한다. 쌍방향 동질들의 결합이 누적된 오류를 야기하고 이미지들 사이의 다중 제약들을 무시하게 될 것이기 때문에, 이는 파노라마의 끝이 합쳐져야 하는 필수적인 단계다. 각 단계에서 추가되어지는 가장 일치하는 이미지(최대 일치 항목 수)와 함께 이미지는 번들 조정기에 하나씩

추가된다, 새로운 이미지는 가장 일치하는 이미지와 동일한 회전 및 초점 길이를 사용하여 초기화된다. 그런 다음, Levenberg-Marquardt를 사용하여 매개변수를 업데이트한다. 제안한 방법에서 사용되는 목적 함수(objective function)은 견고한 합 제곱 투영 오류다. 즉, 각 특징점은 일치하는 모든 영상에 투영되며, 카메라 파라미터와 관련하여 제곱한 영상 거리의 합계는 최소화된다.

실제로 하나의 Ray에 따라 표본된 화소들은 그것이 교차하는 모든 이미지에서 동일한 세기를 가지지 않는다. 그리고 이득 보상(Gain compensation) 후에도, vignetting, parallax effect, mis-registration error, Radial distortion 등과 같은 여러 모델링되지 않은 효과로 인해 일부 영상 에지들은 여전히 볼 수 있다. 이와 같은 효과들을 감소하기 위하여 Multi-band Blending 기술[20]이 필요하다.

임의의 프레임 상에 n개 카메라로부터 등록된 n개 영상 $I_i(x,y)$과 이의 구좌표계 영상 $I_i(\theta,\phi)$일 때, 각 영상에 가중치 함수 $W_i(x,y)=W_i(x)W_i(y)$를 할당한다. 여기서 $W(x)$는 영상 중심점인 1에서 에지인 0까지 선형적으로 변한다. 가중치 함수는 구좌표계로 재샘플된다; $W_i(\theta,\phi)$.

먼저, 최대 가중치 함수를 가지는 값을 찾음으로써 각 영상에 대한 블렌딩 가중치를 초기화한다; If $W_i(\theta,\phi)=\arg\max_j W_j(\theta,\phi)$, then $W_i^{\max}(\theta,\phi)=1$, otherwise $W_i^{\max}(\theta,\phi)=0$. 이 최대 가중치 맵은 각 밴드에 대한 블렌딩 가중치를 형성하기 위하여 연속적으로 블러링된다. 표준편차 σ인 가우시안 필터 $G_\sigma(\theta,\phi)$일 때, 렌더링된 영상의 고주파 통과 영상은 $H_{i,\sigma}(\theta,\phi)=I_i(\theta,\phi)-I_i(\theta,\phi)*g_\sigma(\theta,\phi)$와 같이 공간 주파수 영역 $[0,\sigma]$ 상에 정의된다. 제안한 방법은 $I_i(x,y)$에 대한 최대 가중치 맵을 블러링함으로써 만들어진 블랜딩 가중치 $W_{i,\sigma}(\theta,\phi)$를 사용하여 영상 간의 이 영역을 블렌드한다. $W_{i,\sigma}(\theta,\phi)$는 $[0,\sigma]$ 밴드 상 블렌드 가중치이다; $W_{i,\sigma}(\theta,\phi)=W_i^{\max}(\theta,\phi)*G_\sigma(\theta,\phi)$. 이후 주파수 대역들은 더욱더 낮은 주파수 대역통과 이미지와 더욱 블러링된 블렌드 가중치를 사용함으로써 블렌딩된다; $H_{i,(k+1)\sigma}=I_{i,k\sigma}*(1-g_{\sigma'})$, $W_{i,(k+1)\sigma}=W_{i,k\sigma}*g_{\sigma'}$, where $\sigma'=\sqrt{(2k+1)\sigma}$ for $k\geq 1$. 각 밴드 상에 겹치는 영상은 해당 블렌드 가중치를 사용하여 선형적으로 결합된다.

$$I_{i,k\sigma}(\theta,\phi)=\frac{\sum_{i=1}^{n}H_{i,k\sigma}(\theta,\phi)\,W_{i,k\sigma}(\theta,\phi)}{\sum_{i=1}^{n}W_{i,k\sigma}(\theta,\phi)}\quad \text{for } k\geq 1 \qquad (3)$$

3.1.2 H.264 기반 360도 파노라마 비디오 압축

ㄴ위의 과정에 의하여 생성된 360도 파노라마 비디오 영상은 H.264 코덱에 의하여 압축된다. 제안한 방법은 NVIDIA의 NVENC SDK[14]에서 제공하는 Hardware-Accelerated Video Encoding / Decoding 방식으로 H.264 코덱 기반 압축[21,22]을 수행한다. 하드웨어 기반 인코더(NVENC로 표시됨)는 완전히 가속화된 하드웨어 기반 비디오 인코딩과 디코딩을 제공하고 있고, 이는 그래픽 성능과 무관하다.

특징점 기반 파노라마 이미지 스티칭과 H.264 기반 360도 파노라마 비디오 압축 과정은 GPU 기반 고속으로 처리되어, 생성된 360도 파노라마 비디오 데이터는 RTSP로 전송하기 위하여 H.264 비트열로 프레임 버퍼에 적재된다.

3.2 RSTP 기반 360도 파노라마 비디오 전송

360도 파노라마 비디오를 스트리밍하기 위하여 H.264로 압축된 비트스트림은 RTSP Server로 전송된다. 제안한 방법은 먼저 RTSP/RTP/RTCP를 지원하는 오픈소스 라이브러리인 Live555 미디어 스트리밍 라이브러리[23]를 사용하여 Fig. 4에서와 같이 RTSP 서버를 구현한다.

RTSP는 IETF에서 지정한 네트워크 응용 프로그램 계층의 통신규약으로서, 미디어 서버를 원격으로 제어하기 위해서 사용된다[24,25]. H.264 비트스트림은 NAL(Network Abstraction Layer)을 RTP(Real-time Transport Protocol)를 통해 클라이언트에 전송되어진다. 여기서 NAL은 H.264의 압축된 영상데이

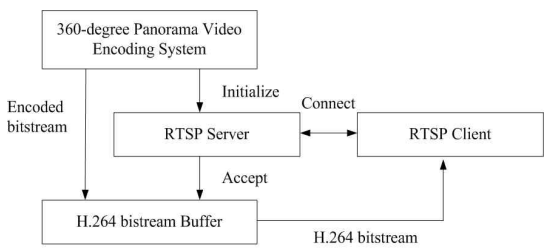


Fig. 4. RTSP Server call process.

터인 VCL(Video Coding Layer)와 디코딩 정보를 포함하는 SPS(Sequence Parameter Set) 및 PPS(Picture Parameter Set)의 파라미터 등으로 구성된다. H.264 비트스트림이 RTP를 통하여 RTSP Client에게 보내어지면, Client는 RTSP 규약에 명시된 명령들을 통하여 원격으로 미디어 서버를 제어하면서 360도 파노라마 비디오 영상을 확인할 수 있다.

제안한 360도 파노라마 비디오 시스템에서는 RTSP Server가 초기화가 되면 H.264 비트스트림을 읽기 위한 read 상태가 된다. 그런 다음, 360도 파노라마 비디오의 프레임 데이터가 프레임 버퍼에 축적되면, RTSP Server는 이러한 이벤트를 감지하여 프레임 데이터를 payload에 담아서, RTP를 통하여 프레임 데이터를 전송한다. RTSP Server가 연속된 H.264 비트스트림을 읽어 들이는 동안에는 서버와 클라이언트 간의 통신은 지속되며, 비트스트림의 EOF(End Of File)을 만나면 종료된다.

3.3 360 파노라마 비디오 스트리밍 시스템 인터페이스

제안한 시스템에서는 고품질 파노라마 비디오 영상을 위한 카메라 보정 (Camera Matrix) 설정이 있다. 본 실험에서는 카메라 행렬 정보를 입력받는 Fig. 5(a)에서와 같이 GUI를 설계하였다. Fig. 5 (a)에서와 같이 제안한 카메라 행렬 정보 입력 위한 GUI에서는 카메라별 입력 영상과 기본적인 extrinsics의 회전과 위치 값을 입력받은 후, 이 정보를 기반으로 자동으로 카메라 보정을 한다. 그리고 extrinsics의 회전, 이동값과 함께 intrinsics의 초점거리, 주점, 렌즈 종류, 왜곡계수 등 Camera Matrix에 요구되는 변수들은 GUI를 통하여 사용자에 의하여 직접 설정될 수 있다.

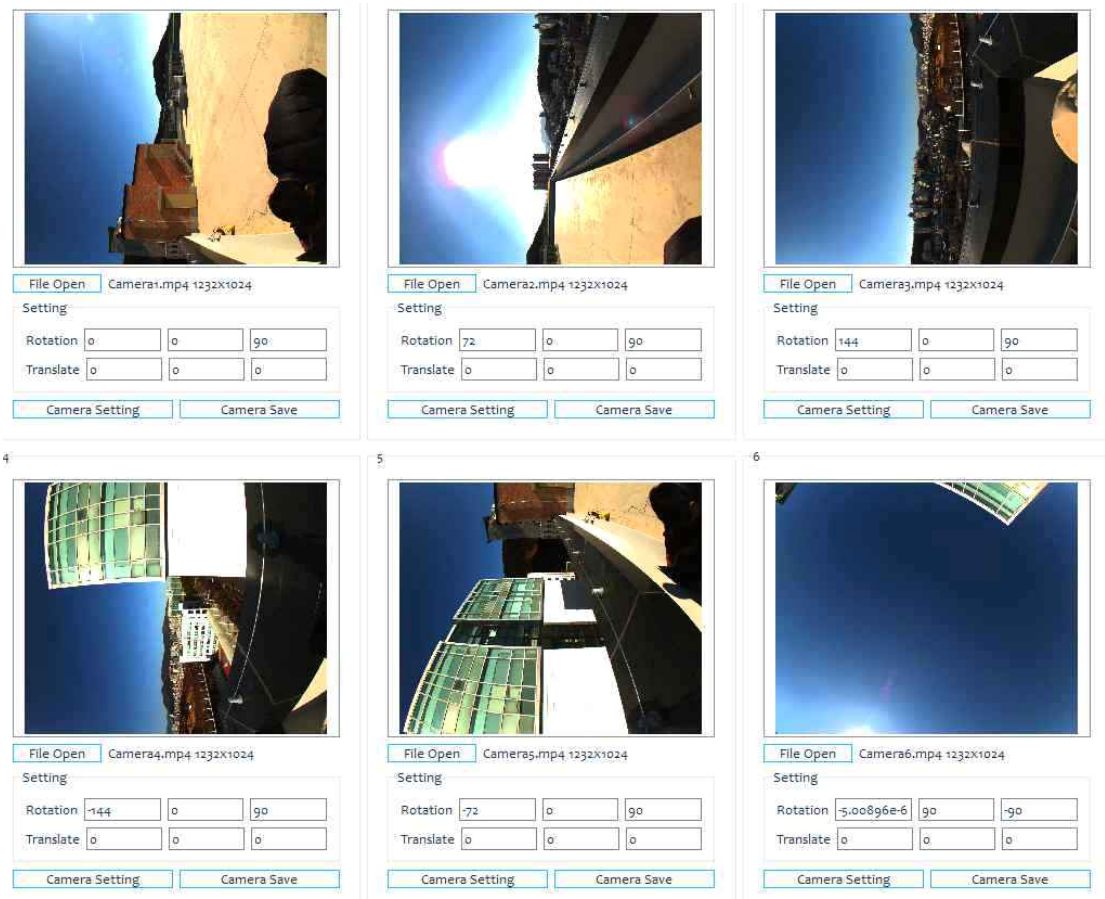


Fig. 5. Camera matrix setting for 6 virtual cameras.

이상와 같이 각 카메라에서 촬영된 영상을 사용자가 설정한 Camera Matrix에 의하여 정합하고 나면, Fig. 5(b)에서와 같이 첫 번째 파노라마 프레임 영상을 통하여 정합 과정을 확인할 수 있다. 확인 과정으로 통하여 파노라마 영상을 생성할 것인지, 스트리밍할 것인지 결정할 수 있다.

4. 실험 결과

본 실험에서는 먼저 제안한 RHI 기반 가상 텍스처 생성과 다중 가상 비디오 엔코딩의 실시간성을 CPU와 GPU 기반으로 측정하였고, 다음 360도 파노라마 비디오 생성 및 스트리밍의 시간 성능을 FPS(Frame Per Second)으로 측정하였다.

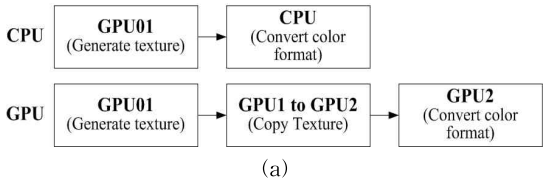
4.1 가상 텍스처 및 다중 가상 비디오 엔코딩 성능 분석

본 실험에서는 RHI 기반 가상 텍스처 생성과 다중 가상 카메라로부터 획득된 비디오의 엔코딩 과정을 CPU와 GPU 기반의 처리 속도를 비교 분석하였다. 실험에 사용된 세부적 하드웨어 정보는 다음과 같다.

- CPU : AMD, Ryzen Threadripper 1900x, 4.0 Ghz, 16 thread
- GPU01 : NVIDIA, Geforce RTX 2080TI, 4,352 CUDA core
- GPU02 : NVIDIA, Quadro M6000, 3072 CUDA core
- RAM : GeIL, DDR4 8G, 4CH / 2,400 Mhz

4.1.1 가상 텍스처의 포맷 변환 실험

본 실험에서는 CPU와 GPU 기반으로 가상 텍스처의 포맷 DXG(DXGI_FORMAT_R16G16B16A16_FLOAT(DirectX 11))에서 sRGBA 형식으로 변환할 때의 시간 성능을 측정하였다. CPU와 GPU 기반의 가상 텍스처 성능 비교를 위하여, 본 실험에서는 가상 세계 속의 여러 대의 카메라를 설정한 후, 카메라가 생성하는 텍스처를 변환하였고, 이에 대한 실험 과정은 Fig. 6에서와 같다. 먼저, 본 실험에서는 우선 텍스처 생성을 담당하는 별도의 GPU를 두어, GPU 연산을 진행할 때 발생될 수 있는 성능 감소를 최소화하였다. CPU 기반 실험에서는 생성된 텍스처를 RAM에 복사하고, GPU 기반 실험에서는 텍스처 변환을 담당하는 GPU의 메모리에 텍스처를 복사하였다.



(a)

(b)

Fig. 6. (a) Texture transformation of CPU and GPU and (b) a test texture used in texture transform performance experiments.

실험 결과는 Fig. 7(a)에서와 같다. CPU의 경우, 카메라 1개일 때는 어느 정도 실시간 변환이 가능하나, 카메라 2개 이상으로 늘어나면서 생성되는 텍스처의 수를 감당하지 못하여, fps가 급격히 줄어드는 것을 알 수 있다. 실제로 텍스처 생성과 CPU 변환을 동기화하지 않고 병렬로 수행할 경우, RAM에 복사된 텍스처가 쌓이면서 메모리 오버 플로우가 발생되고 프로그램이 종료된다. GPU 경우에서는 카메라 개수가 늘어나도 상대적으로 성능 하락폭이 적은 것을 알 수 있다. 특히 카메라가 24개일 때, CPU는 성능 측정이 불가능하였지만, GPU는 6.89 fps를 유지함을 알 수 있다.

본 실험에 사용된 Unreal Engine 4는 가상 카메라가 임의로 텍스처를 생성할 때, 엔진 자체 렌더링 스레드와 동기화해야 한다. 이 때문에 카메라 개수가 늘어날수록 동기화 작업에 따른 불필요한 대기시간이 누적된다. 이는 GPU의 성능을 전부 활용할 수 없게 만드는 원인이 된다. 따라서 본 실험에서 나타난 카메라 개수에 따른 성능 하락폭은 실제보다 더 높을 수 있다.

4.1.2 NVENC를 통한 다중 영상 엔코딩 실험 및 성능 분석

다중 영상 엔코딩 실험에서는 가상 세계 속에 설

치된 가상 카메라의 개수에 따른 GPU 기반 비디오 엔코딩 성능을 평가하였다. 본 실험에서는 가상 카메라를 Fig. 2에 나와 있는 배치도에 따라 설치한 후, 가상 카메라의 수를 조절하면서 성능을 측정하였다.

실험 결과인 Fig. 7(b)을 살펴보면, 720×480 및 1920×1088 해상도에서는 2개의 가상 카메라까지는 성능 하락 없이 엔코딩이 가능함을 볼 수 있다. 특히 720×480 해상도에서는 앞 절의 GPU 실험 결과와 동일하게 나타났다. 이는 다중 GPU 구조에 따라 텍스처 생성과 비디오 엔코딩이 분리되면서, 비디오 엔코딩에서 발생되는 부하가 실제 애플리케이션에서는 영향을 주지 않은 것으로 판단된다. Fig. 8은 720×480 해상도의 카메라 24대에 대한 비디오 엔코딩 결과를 보여주고 있다. 이 때 7fps 정도 측정되었다.

3840×2160 해상도에서는 하드웨어의 성능 한계로, 1개의 카메라일 때도 Unreal Engine 4의 최대 fps를 넘지 못하였다. 또한 해상도가 증가하면서 텍스처 복사에 큰 오버 헤드가 걸려, 카메라 개수가 늘어나면서 나타나는 성능 하락폭이 다른 해상도에 비하여 증가하였다. 특히 3840×2160 해상도의 카메라 12개 이상부터는 성능 측정이 불가능해졌다.

4.2 360도 파노라마 비디오 스트리밍 성능 평가

제안한 360도 파노라마 비디오 영상 시스템을 통하여 생성된 첫 번째 프레임을 Fig. 9에 나타내었다. Fig. 9의 첫 번째 결과 영상은 가상 세계에서 배치된 가상의 360도 카메라로부터 스티칭된 파노라마 영상이고, 나머지 두 결과 영상은 실 세계에서 다중 카메

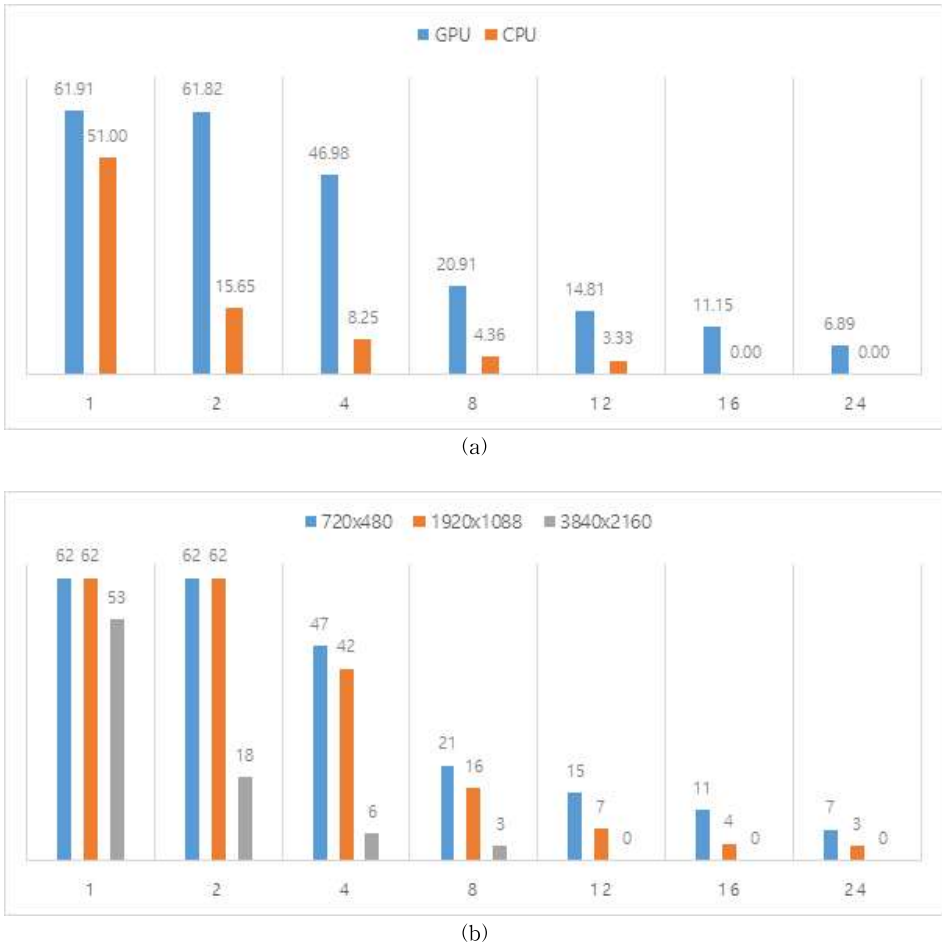


Fig. 7. Experimental fps results of (a) Texture transformation and (b) video encoding of multiple virtual cameras video (y-axis : fps).

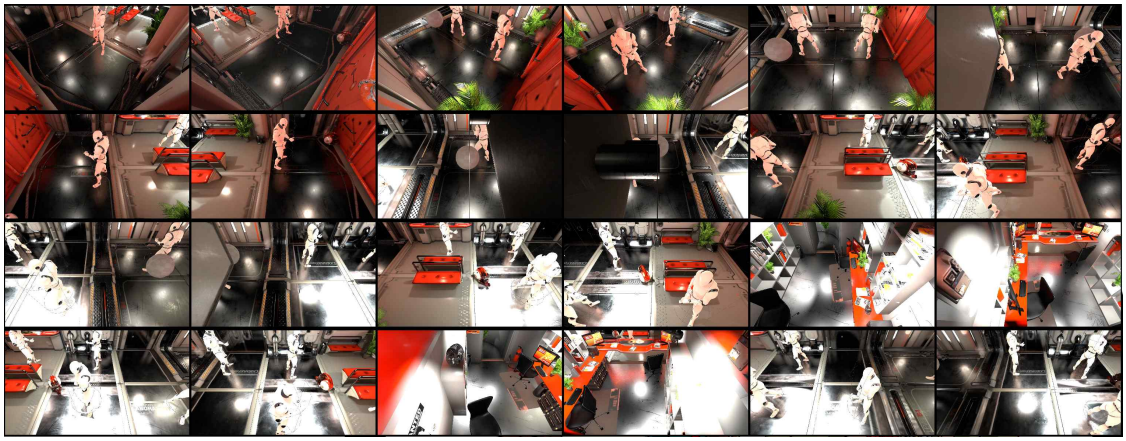


Fig. 8. Video coded images by 24 virtual cameras with 720×480 resolution.

라로부터 스티칭된 파노라마 영상이다. 이 영상들은 HMD를 사용하거나 VRPlayer와 같은 전용 뷰어를 사용하여 Spherical projection frame의 형태로 변환하여 360도로 둘러싸여있는 것과 같은 효과로 확인할 수 있다. 360도 파노라마 비디오 영상을 스트리밍한 결과는 데스크탑 환경에서 VLC Media Player를 통해서 확인하거나, HMD를 사용한 경우 HTC Vive에서 지원하는 Steam의 wanshi VR 어플리케이션을 사용하여 원격으로 스트리밍된 결과를 확인할 수 있다.

(a)

(b)

Fig. 9. 360-degree panorama images stiched by proposed system with (a) virtual camera and (b) real camera.

본 실험에서는 360도 파노라마 비디오의 스트리밍 과정을 fps로 측정하였으며, 실시간 스트리밍 가능성에 대한 평가는 소셜 미디어 플랫폼에서 평균적으로 지원하는 30fps~60fps를 기준으로 비교 평가하였다. 실험에 사용된 환경은 nvidia사의 그래픽카드 GTX1080ti가 사용되고, Unreal Engine4에서 구축한 Virtual Camera와 실제 360카메라 모델인 Z cam s1, Ladybug5+, Z cam V1 pro Fps등을 사용하였다. 실험 측정은 한 프레임마다 360도 파노라마 비디오 영상을 생성하고 클라이언트로 전송하는 일련의 과정을 시간으로 측정하고, 매초 얻어지는 fps값의 분당 평균값을 정수값으로 취한 결과로 평가하였다.

Table 1은 3680×2428 해상도의 카메라 8대로 구성된 Z-cam v1 pro카메라로부터 입력받아 360도 파노라마 비디오 스트리밍 과정을 fps로 측정한 결과를 나타낸 것이다. 실험 결과로부터 HMD 환경에서 스테레오(Stereo) 화면 송출 시 2K 해상도에서 45fps, 4K 해상도에서 33fps가 측정되어 360도 파노라마 비디오의 출력 해상도 크기 차이에서 속도가 바뀌는 것을 알 수 있었다. 표 3의 오른쪽 열은 1232×1024 해상도의 카메라 6대로 구성된 Ladybug5+ 카메라로부터 입력받아 360도 파노라마 비디오 스트리밍 과정을 fps 측정한 결과를 나타낸 것이다. 실험 결과로부터 4K의 해상도에서 95fps, 2K의 해상도에서 126fps가 측정되는 것을 확인할 수 있었다.

위 두 가지 경우의 측정 결과로부터, 360도 파노라마 비디오 스트리밍 결과는 출력 해상도 뿐만 아니라

Table 1. Estimated average fps values in processing panoramic video captured by Z-Cam V1 pro and Ladybug5+.

Camera	8 cameras of Z-Cam V1 pro,		6 cameras of Ladybug5+	
Output resolution	3840×1920	2560×1280	3840×1920	2560×1280
fps	33	45	95	126

360도 카메라를 구성하는 입력 카메라의 개수와 입력 해상도에 영향을 받는 것을 알 수 있다. 실험 과정에서 다수의 360도 카메라로부터 압축된 데이터를 디코딩하여 입력을 받으므로, 이에 대한 지연 시간이 포함된다. 그러나 제안한 시스템은 평균적으로 최소 33fps 이상으로 실시간 360도 파노라마 비디오 스트리밍 서비스에 적합함을 확인할 수 있었다.

5. 결 론

본 논문에서는 GPGPU 기반의 360도 파노라마 비디오 스트리밍 기술을 제안하였으며 실제 360도 카메라 뿐만 아니라 가상의 카메라를 사용한 실험을 통해 제안한 시스템이 최대 4K의 실시간 360도 파노라마 비디오 스트리밍 서비스에 적합함을 확인하였다. 제안한 시스템은 360도 파노라마영상을 스트리밍하는 기능 뿐만 아니라 높은 해상도의 고품질 파노라마 영상 생성에서도 효과적이었다. 그러나 제한된 전송 대역폭으로 인하여 실제 360도 카메라 영상을 생성하는데 있어 문제가 있었으며, 또한 입력카메라의 해상도 및 카메라 개수에 따라 처리 속도가 지연되는 현상이 있었다. 따라서, 이 문제를 해결하기 위해 향후 작업으로 카메라에서 비디오 이미지 데이터를 지체없이 수신하고 디코딩없이 비디오 이미지 데이터를 직접 정렬 및 일치시키도록 하는 버스를 구축하여 개선된 live 360도 파노라마 영상 스트리밍 서비스를 구현하려고 한다.

REFERENCE

[1] M. Hosseini and V. Swaminathan, “Adaptive 360 VR Video Streaming: Divide and Conquer,” *IEEE International Symposium on Multimedia (ISM)*, pp. 107-110, 2016.

[2] A.T. Nasrabadi, A. Mahzari, J.D. Beshay, and R. Prakash, “Adaptive 360-Degree Video Streaming using Scalable Video Coding,” *Proceedings of ACM on Multimedia Conference*, pp. 1689-1697, 2017.

[3] D.V. Nguyen, H.T.T. Tran, and T.C. Thang, “A Client-based Adaptation Framework for 360-Degree Video Streaming,” *Journal of Visual Communication and Image Representation*, Vol. 59, pp. 231-243, 2019.

[4] J. Jeong and K. Jun, “High Resolution 360 degree Video Generation System using Multiple Cameras,” *Journal of Korea Multimedia Society*, Vol. 19, No. 8, pp. 1329-1336, 2016.

[5] D.V. Nguyen, T.T. Le, S. Lee, and E.-S. Ryu, “SHVC Tile-Based 360-Degree Video Streaming for Mobile VR: PC Offloading Over mmWave,” *Sensors*, Vol. 18, Issue. 11, 3728, 2018.

[6] M. Han, S.-H. Lee, and S. Ok, “A Real-Time Architecture of 360-Degree Panoramic Video Streaming System,” *2nd IEEE International Conference on Knowledge Innovation and Invention*, pp. 477-480, 2019.

[7] T. Dendale, *Improving 360-Degrees Panoramic Video Stitching*, Master Thesis, Universiteit Hasselt, 2016.

[8] M. Brown and D.G. Lowe, “Automatic Panoramic Image Stitching using Invariant Features,” *International Journal of Computer Vision*, Vol. 74, Issue 1, pp. 59-73, 2007.

[9] W. Ye, K. Yu, Y. Yu, and J. Li, “Logical Stitching: A Panoramic Image Stitching Method Based on Color Calibration Box,” *14th IEEE International Conference on Signal Processing (ICSP)*, pp. 1139-1143, 2018.

[10] Y. Lu, K. Wang, and G. Fan, “Photometric Calibration and Image Stitching for a Large Field of View Multi-Camera System,” *Sensors*, Vol. 16, Issue 4, E516, 2016.

[11] A. Prakash, S. Boochoon, M. Brophy, D. Acuna,

E. Cameracci, G. State, O. Shapira, and S. Birchfield, "Structured Domain Randomization: Bridging the Reality Gap by Context-Aware Synthetic Data," *arXiv Preprint, arXiv: 1810.10093*, 2018.

[12] R. Girshick, "Faster R-CNN," *IEEE International Conference on Computer Vision (ICCV)*, pp. 1440-1448, 2015.

[13] G.W. Hyun, *Real-time Iimage Generation Method through Multiple Virtual Camera*, Master Thesis, Tongmyong University, 2017.

[14] NVIDIA Video Codec SDK, https://developer.nvidia.com/nvidia-video-codec-sdk (accessed on July 1, 2019).

[15] Ultra Video Group, http://ultravideo.cs.tut.fi/ (accessed on July 1, 2019).

[16] M. Wang, S. Niu, and X. Yang, "A Novel Panoramic Image Stitching Algorithm Based on ORB," *International Conference on Applied System Innovation (ICASI)*, 2017.

[17] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," *IEEE International Conference Computer Vision (ICCV)*, pp. 24-33, 2011.

[18] L. Yu, Z. Yu, and Y. Gong, "An Improved ORB Algorithm of Extracting and Matching Features," *International Journal of Signal Processing, Image Processing and Pattern Recognition*, Vol. 8, No. 5, pp. 117-126, 2015.

[19] F. Wu and X. Fang, "An Improved RANSAC Homography Algorithm for Feature Based Image Mosaic," *7th WSEAS International Conference on Signal Processing, Computational Geometry & Artificial Vision*, pp. 202-207, August 2007.

[20] VRWorks - 360Video - Multiband Blending. https://developer.nvidia.com/vrworks/vrworks-360video/Multiband-Blending (accessed on July 1, 2019).

[21] T. Moriyoshi, F. Takano and Y. Nakamura, "GPU Acceleration of H.264 / MPEG-4 AVC Software Video Encoder," *Asia Pacific Signal and Information Processing Association, Annual Summit and Conference (APSIPA ASC)*, 2011.

[22] NVIDIA DevTech and A. Obukhov, "GPU-Accelerated Video Encoding," *GPU Technology Conference*, 2010.

[23] Internet Streaming Media, Wireless, and Multicast technology, services & standards. https://www.live555.com (accessed on July 1, 2019).

[24] I. Santos-González, A. Rivero-García, J. Molina-Gil, and P. Caballero-Gil, "Implementation and Analysis of Real-Time Streaming Protocols," *Sensors*, Vol. 17, Issue 4, 846, 2017.

[25] MS-RTSP, *Real-Time Streaming Protocol (RTSP) Windows Media Extensions*, v20180912, Sept. 2018.

옥 수 열

1994년 동아대학교 산업공학과 학사 졸업(공학사)
1998년 츠쿠바대학 이공학연구과 석사 졸업(공학석사)
2001년 츠쿠바대학 공학연구과 박사 졸업(공학박사)

2001년~20003년 일본 통신종합연구소(NICT) 연구원
2004년~2020년 동명대학교 게임공학과 교수
2021년~현재 동아대학교 컴퓨터공학과 교수
관심분야 : 영상처리, HPC(GPU), 인공지능, 디지털트윈

이 석 환

1999년 경북대학교 전자공학과 학사 졸업(공학사)
2001년 경북대학교 전자공학과 석사 졸업(공학석사)
2004년 경북대학교 전자공학과 박사 졸업(공학박사)

2005년~2020년 동명대학교 정보보호학과 교수
2021년~현재 동아대학교 컴퓨터공학과 교수
관심분야 : AI영상보안, 컴퓨터비전, 보안응용, 디지털트윈