



### 저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원 저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리와 책임은 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)



Master's Thesis

Multi-layered Bitrate Allocation System  
Based on ROI for Cloud Gaming over  
Wireless Network

Goeon Park (박 고언)

Department of Computer Science and Engineering  
Pohang University of Science and Technology

2019



# 무선 네트워크 상에서 클라우드 게임을 위한 ROI 기반 비트레이트 제어시스템

Multi-layered Bitrate Allocation System  
Based on ROI for Cloud Gaming over  
Wireless Network



# Multi-layered Bitrate Allocation System Based on ROI for Cloud Gaming over Wireless Network

by

Goeon Park

Department of Computer Science and Engineering  
Pohang University of Science and Technology

A thesis submitted to the faculty of the Pohang University of  
Science and Technology in partial fulfillment of the  
requirements for the degree of Master of Science in the  
Computer Science and Engineering

Pohang, Korea

12. 17. 2018

Approved by

Hwangjun Song (Signature)

Academic advisor



# Multi-layered Bitrate Allocation System Based on ROI for Cloud Gaming over Wireless Network

Goeon Park

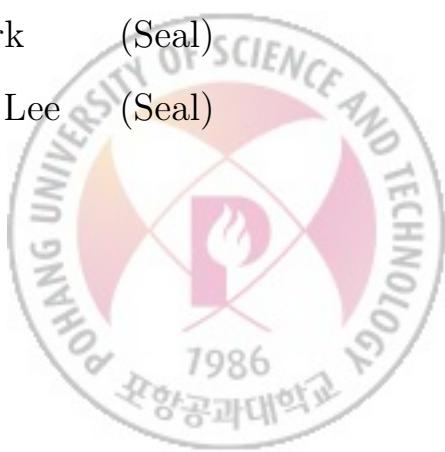
The undersigned have examined this thesis and hereby certify  
that it is worthy of acceptance for a master's degree from  
POSTECH

12. 17. 2018

Committee Chair Hwangjun Song (Seal)

Member Chanik Park (Seal)

Member Seungyong Lee (Seal)



MCSE 박 고연. Goeon Park  
20172612 Multi-layered Bitrate Allocation System Based on ROI for  
Cloud Gaming over Wireless Network,  
무선 네트워크 상에서 클라우드 게임을 위한 ROI 기반 비  
트레이트 제어시스템  
Department of Computer Science and Engineering , 2019,  
30p, Advisor : Hwangjun Song.

## ABSTRACT

본 논문은 대역폭이 제한적이고 변동이 심한 무선 네트워크 환경에서 클라우드 게임의 시각적 품질을 향상시키기 위한 Region of Interest (ROI) 기반의 비트레이트 할당 시스템을 제안한다. 제안하는 시스템은 3D 렌더링 엔진으로부터 렌더링 정보를 활용하여 ROI를 추출하며, 이를 확장함으로써 여러 영역으로 나누고, 영역 별로 각각 다른 가중치를 적용하여 비트레이트 할당이 차별적으로 이루어지도록 구성하였다. 또한 대역폭 변동이 심한 무선 환경에서 빈번한 목표 비트레이트 변동 및 비디오 품질의 급격한 변화를 막기 위해 인코더/디코더 버퍼 상태를 고려한 piece-wise 고정비트레이트 대역폭 예측 기법을 제안한다. 제안하는 시스템은 오픈 소스 3D 렌더링 엔진 및 게임을 활용하여 테스트베드를 구축하였고, 무선 네트워크 환경에서 실험함으로써 성능을 검증하였다.



# 목 차

I.	서론	1
II.	관련 연구	4
III.	제안하는 ROI 기반의 클라우드 게임 시스템	6
3.1	ROI-based Bitrate Allocation . . . . .	7
3.1.1	ROI Definition . . . . .	7
3.1.2	ROI Extraction . . . . .	7
3.1.3	Multi-layered ROI Extraction . . . . .	9
3.2	Encoder/Decoder buffer occupancy-aware piece-wise CBR band-width estimation . . . . .	15
IV.	실험 결과	19
4.1	Performance Comparison of Bitrate Allocation Algorithms . . .	21
4.2	Performance Comparison of Channel Estimation Algorithms . .	22
V.	결론 및 향후과제	26
	참 고 문 헌	27



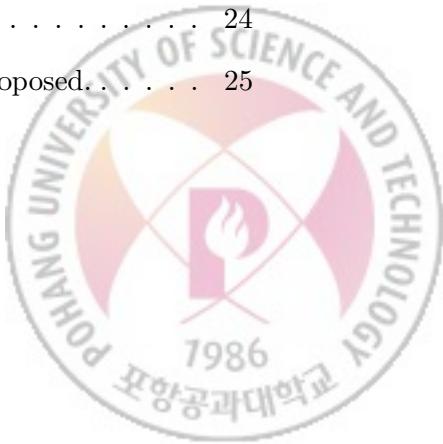
# 표 목 차

3.1 Description of key symbols for ROI-based bitrate allocation	8
3.2 Description of key symbols for Encoder/Decoder buffer occupancy-aware piece-wise CBR bandwidth estimation	15
4.1 채널 예측 알고리즘 PSNR 및 버퍼 상태 비교	25



# 그 림 목 차

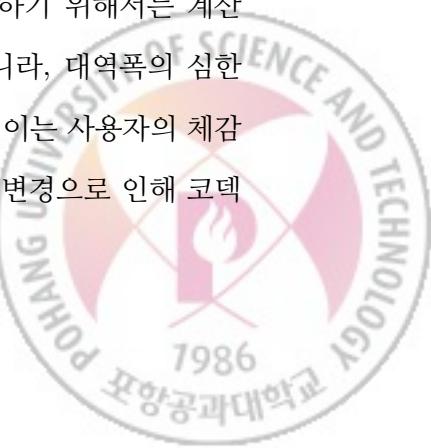
1.1 클라우드 게임의 구조. . . . .	2
3.1 제안하는 시스템의 구조 및 동작 과정. . . . .	7
3.2 스텐실 버퍼에 저장된 ROI 정보. . . . .	9
3.3 Image dilation 을 통한 ROI 확장: (a) 원본 영상, (b) 스텐실 버퍼, (c) Image dilation, (d) 영역 구분. . . . .	11
3.4 Zoom in/out 에 따른 Image dilation level $K$ : (a) $K = 8$ , (b) $K =$ 12, (c) $K = 16$ . . . . .	12
3.5 깊이 버퍼에 저장된 정보. . . . .	12
4.1 구축된 테스트베드 사진. . . . .	20
4.2 동일한 비트레이트로 인코딩된 이미지 비교 : (a) 기존 H.264/AVC (b) ROI-based H.264/AVC. . . . .	22
4.3 PSNR 비교 : (a) 기존 vs ROI PSNR 비교 (b) 영역별 PSNR 비교. .	23
4.4 목표 비트레이트를 설정하는 알고리즘 비교 : (a) BW following, (b) EWMA, (c) Proposed. . . . .	23
4.5 인코더/디코더 버퍼 점유율 비교 : (a) BW following, (b) EWMA, (c) Proposed. . . . .	24
4.6 PSNR 비교 : (a) BW following, (b) EWMA, (c) Proposed. . . . .	25



## I. 서론

최근 글로벌 클라우드 게임 마켓 보고서에 다르면 2018년부터 2022년까지 클라우드 게임 시장은 연평균 30.48% 성장할 것으로 전망 되며 [1], 구글, 마이크로소프트와 같은 세계적인 IT 기업들도 클라우드 게임 개발에 적극적으로 참여하고 있다. 구글은 최근 Project Stream [2] 클라우드 기반 게임 스트리밍 프로젝트를 발표하여, 웹브라우저 크롬을 통해 게임 서비스를 지원할 수 있도록 테스트 중에 있다. 마이크로소프트는 약 1200개 이상의 게임을 운영하는 클라우드 기반 게임 플랫폼 PlayFab [3]을 인수하여 클라우드 게임 개발에 활발하게 투자하고 있다. 뿐만 아니라, 2020년 출시를 목표로 스칼렛 [4] 게임 콘솔 또한 개발중에 있다.

그림 1.1을 보면 클라우드 게임이란, 네트워크를 통해 게임 제어 및 비디오 스트리밍이 이루어지는 방식의 게임으로서 클라우드 서버와 클라이언트로 이루어져 있다. 클라이언트가 게임 조작을 위해 값을 입력하면 네트워크를 통해 클라우드 서버로 전송 되고, 클라우드는 서버는 게임상에서의 정보를 업데이트 한다. 이에 따라 3D 렌더링을 통해 화면에 표현될 3D 오브젝트를 구성하고 이를 인코딩 한 후에, 네트워크를 통해 클라이언트로 전송한다. 마지막으로 클라이언트는 이를 디스플레이 함으로써 게임이 이루어지는 방식이다. 보다 좋은 품질의 게임 서비스를 위해서는 높은 비트레이트를 요구하는데, 대역폭이 제한적이고 변동이 심한 무선 네트워크 환경에서 네트워크 지연에 특히 민감한 클라우드 게임을 지원하기 위해서는 계산 복잡한 연산들이 모두 실시간으로 이루어져야 한다. 뿐만 아니라, 대역폭의 심한 변동으로 인해 비디오 품질 또한 변동이 자주 발생할 수 있으며 이는 사용자의 체감 품질을 떨어뜨릴 뿐만 아니라, 너무 빈번한 목표 비트레이트의 변경으로 인해 코덱



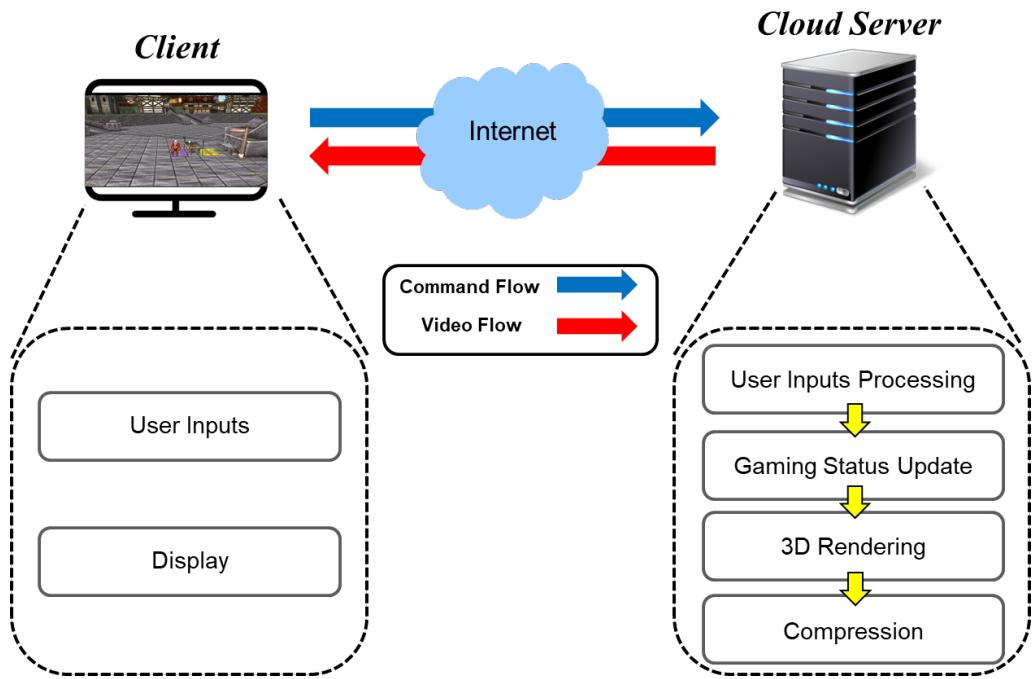


Figure 1.1: 클라우드 게임의 구조.

이 지원하지 못 하는 경우가 발생할 수 있다. 따라서, 무선 환경에서 클라우드 게임 서비스를 지원하기 위해서는 위와 같은 문제를 해결해야 한다.

한편, 인간의 시각 체계는 한 이미지를 인식할 때 이미지 전체를 한 번에 인식하기 보다는 시각적으로 뚜렷한 특징을 가지거나 움직임이 많은 일부 영역 즉, Region of Interest (ROI)에 더 집중하는 특성이 있다 [6]. 따라서 다른 영역보다 ROI에 더 많은 비트레이트를 할당하여 인코딩이 이루어진다면 제한적인 대역폭 내에서 인간이 자각하는 시각적 화질을 더욱 향상 시킬 수 있다. 또한, 대역폭 변동이 심한 무선 환경에서 비디오 품질의 급격한 변화를 막기 위해서는 Constant Bitrate (CBR)로 인코딩 되어야 하는데, 인코더/디코더 버퍼 상태를 고려하지 않고 CBR로 인코딩이 이루어지면, 인코더 버퍼 언더플로우 혹은 디코더 버퍼 언더플로우가 발생

하여 대역폭 Utilization을 떨어뜨리고, 디스플레이 시에 freezing이 발생할 수 있다. 따라서 인코더/디코더 상태 및 네트워크 상태를 함께 고려하여 목표 비트레이트가 결정되어야 한다.

본 논문에서는 대역폭이 제한된 무선 환경에서 클라우드 게임의 시각적 화질을 향상시키기 위한 ROI 기반의 비트레이트 할당 시스템을 제안한다. 제안하는 시스템에서는 ROI 추출을 위해 3D 렌더링 정보를 활용하고 이를 비트레이트 할당에 활용하여 시각적 화질을 향상 시키도록 구성하였다. 또한 목표 비트레이트 결정을 위해 인코더/디코더 버퍼 상태를 고려한 piece-wise CBR 대역폭 예측 알고리즘을 제안한다. 인코더/디코더 버퍼 상태를 함께 고려하여 버퍼 언더플로우가 발생하지 않도록 piece-wise CBR로 인코딩이 이루어지도록 구성하였다.

논문의 구성은 다음과 같다. 2장에서는 클라우드 게임과 관련된 연구를 소개한다. 3장에서는 제안하는 시스템의 구조 및 알고리즘에 대해 설명하며, 4장에서는 실제 테스트베드 구축을 통해 얻은 결과를 바탕으로 시스템의 성능을 비교 및 분석하고, 마지막으로 5장에서는 결론을 서술한다.



## II. 관련 연구

다양한 방법으로 클라우드 게임을 위한 연구가 진행되고 있다 [6]. 먼저, 다른 일반적인 영상과는 달리 게임은 3D 렌더링 정보를 활용할 수 있기 때문에, 컴퓨터 그래픽스 정보를 활용함으로써 클라우드 게임의 네트워크 지연 문제를 해결하기 위한 다양한 방법들이 연구 되어 오고 있다. Liu et al. [7]는 스텐실 버퍼, 깊이 버퍼를 통해 비트레이트 할당 모델을 개발하였고, H.264/AVC에서 Motion Estimation (ME)으로 인한 시간 소모를 줄이기 위하여 3D 렌더링 카메라 정보를 활용함으로써 인코딩 가속이 이루어 질 수 있는 알고리즘을 개발 하였다. Shi et al. [8]는 실시간 인코딩을 위한 3D image warping 알고리즘을 제안하였다. 이 알고리즘은 먼저 키 프레임들을 뽑아낸 후, 렌더링 뷔포인트, 픽셀 깊이, 카메라 모션 등의 렌더링 정보를 활용하여 3D image warping 알고리즘을 적용하여 키 프레임 사이에 있는 프레임들을 생성하였다. 이를 통해 인코딩 효율을 향상시키고 interation latency 를 감소 시킬 수 있었다.

클라우드 게임은 인터렉션이 굉장히 높은 어플리케이션이기 때문에 3D 렌더링, 비디오 인코딩과 같은 연산량이 많은 작업들이 실시간으로 이루어져야 한다. Wang et al. [9]은 texture, view point, shading 혹은 lightening 같은 다양한 렌더링 파라미터를 변경하면서 communication complexity 와 computation complexity 를 측정하여 모델링 하였고, 클라우드 게임의 적응적 렌더링 기법을 제안하였다. Ho et al. [10]는 무선 네트워크의 부족한 자원 문제를 해결하기 위해 멀티 패스를 활용하였고, 네트워크 상황에 맞게 렌더링 파라미터를 결정함으로써 QoE를 향상시키는 알고리즘을 제안하였다. Semsarzadeh et al. [11]는 렌더링 정보를 활용하여 ME

연산이 탐색 범위를 거치지 않고도 결정될 수 있도록 복잡도를 낮춰서 비디오 인코더의 성능을 향상 시켰다. Liu et al. [12]는 비디오 인코더, 네트워크 상태 및 장면 정보를 활용하여 모델링 하였고, 네트워크 상태에 맞게 렌더링 파라미터를 결정하여 3D 렌더링이 이루어지도록 알고리즘을 제안하였다.

네트워크를 통해 서비스가 제공되는 클라우드 게임의 가장 중요한 이슈는 네트워크 지연 문제이다 [13]. 네트워크 지연을 줄이기 위해서는 인코더의 압축률을 높이는 것이 매우 중요하며, 인코딩 효율을 높이기 위해 ROI에 기반한 비트레이트 할당 기법들이 많이 연구되고 있다. Nicolas et al. [14]는 ROI 기반의 비디오 인코딩 기법을 제안함으로써 클라우드 게임의 네트워크 지연을 최소화 하고자 하였다. 깊이 버퍼를 통해 얻은 값을 양자화하고, 이를 다시 재구성함으로써 적은 연산으로 ROI 기반의 비트레이트 할당이 가능하도록 알고리즘을 제안하였다. Zhang et al. [15]은 HEVC 인코더를 사용하였으며, HEVC의 기본 단위인 CU를 깊이의 정도로 분류함으로써 ROI를 파악하고 이를 R-D 모델링을 통해 효율적인 인코딩 알고리즘을 제안하였다. Ciubotaru et al. [16]는 ROI 기반의 적응적 스킴을 제안하였다. 한 프레임 내에서 영역을 나누고 가운데 영역일수록 비디오 품질을 향상시키도록 구성하였다.



### III. 제안하는 ROI 기반의 클라우드 게임 시스템

제안하는 시스템은 ROI 기반의 비트레이트 할당을 통하여 클라우드 게임의 시각적 화질을 높이고, 인코더와 디코더 버퍼 상태 고려한 piece-wise CBR 대역폭 예측 알고리즘을 통하여 비디오 품질 변화의 심한 변동을 줄이는 것을 목표로 한다.

이를 위한 시스템 구조는 그림 3.1과 같다. 클라우드 게임을 구성하는 기존 프로그램(파랑)에 제안하는 시스템을 위해 여러 프로그램을 추가 구현하여(빨강) 시스템을 구축하였다. 클라이언트가 게임을 조작하기 위해 키보드 값을 입력하면 네트워크를 통하여 클라우드 서버의 command handler 가 이를 받아 들인다. 이를 렌더링 엔진에 전달하여 게임 상에서의 정보를 업데이트하고 3D 렌더링을 수행한다. 이 때 렌더링 정보를 활용하여 아바타 정보를 포함한 ROI 정보를 추출한다. 이와 동시에 YUV 로 프레임을 추출하여 렌더링 정보와 함께 인코더로 전달한다. 인코더에서는 전달 받은 아바타 정보를 활용하여 ROI를 확장하여 아바타를 포함하는 또 다른 영역을 만들어서 아바타 영역, 아바타를 포함하는 영역, 그리고 그 외의 영역으로 구분한다. 이 때, 클라우드 서버는 클라이언트로부터 주기적으로 네트워크 상태를 전달 받는데, 전달 받은 네트워크 상태를 통해 버퍼 상태를 고려하여 목표 비트레이트를 결정한다. 인코딩된 비디오는 Sender 에서 패킷화되고 네트워크를 통해 클라이언트로 전송되어 마침내 클라이언트의 단말에서 디스플레이 된다.



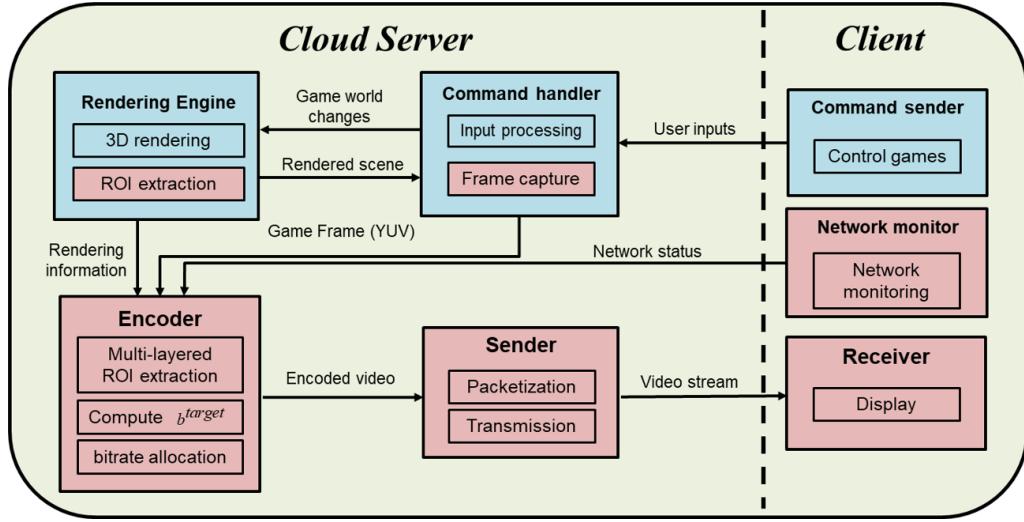


Figure 3.1: 제안하는 시스템의 구조 및 동작 과정.

### 3.1 ROI-based Bitrate Allocation

#### 3.1.1 ROI Definition

사람의 시각 체계에 근거하여 한 프레임 내에 나타나는 모든 아바타들을 ROI로 정의하였다. 사람의 눈은 시각적으로 특별한 특징을 가지거나, 움직임이 많은 부분에 더 초점을 맞추어 집중하는 특성이 있다. 게임상에 나타나는 아바타들은 움직임이 많고 시각적으로 뚜렷하게 나타나기 때문에 ROI로 정의하였다.

#### 3.1.2 ROI Extraction

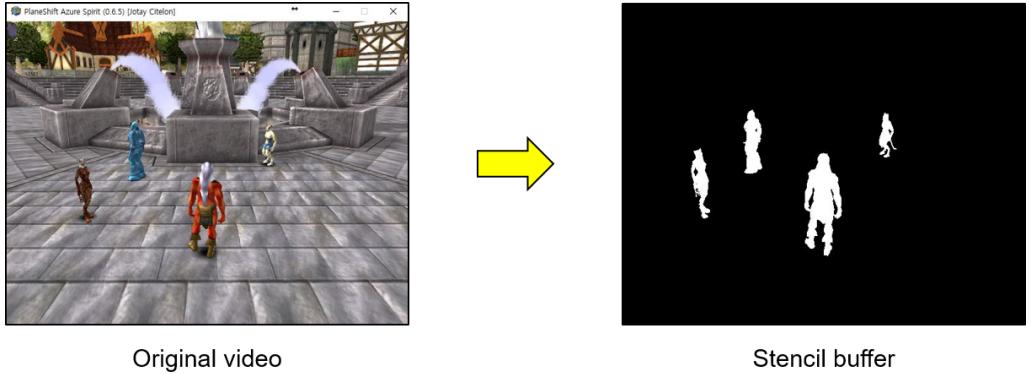
클라우드 게임은 기본적으로 네트워크를 통해 이루어지기 때문에 네트워크 지연에 굉장히 민감하다. 따라서 제안하는 ROI-based bitrate allocation이 이루어지기 위해서는 ROI를 추출하기 위해 많은 연산을 요구할 수 없다. 이를 위해 다른 일반적인 영상에서 사용할 수 없는 클라우드 게임의 특징인 렌더링 정보를 활용하여 ROI를

Table 3.1: Description of key symbols for ROI-based bitrate allocation

Symbol	Description
$\vec{S}$	Stencil values
$\vec{D}$	Depth values
$\vec{q}$	Quantization parameters
$Z$	Zoom level
$K$	Image dilation level
$\omega$	Weighting factors for $\alpha$ , $\beta$ , and $\gamma$
$N_{pixel}$	The number of pixels in a frame
$N_{blk}$	The number of macroblocks in a frame
$MB$	Macroblock
$TH$	Threshold of depth value

추출하였다. 따라서 프레임 안에 구성되어 있는 아바타의 위치 및 모양을 추출하기 위해 스템실 버퍼를 활용하였다. 스템실 버퍼는 pixel 단위의 버퍼로서 디스플레이 되는 해상도와 동일한 해상도를 갖는다. 일반적으로 스템실 버퍼는 렌더링 영역을 제한하기 위해 사용되는데 본 연구에서는 아바타에 값을 할당하고 이를 스템실 버퍼에 저장하게 함으로써 매 프레임이 렌더링 될 때마다 스템실 버퍼에 설정한 아바타 값이 저장되도록 구성하였다. 3D 렌더링 이후에 스템실 버퍼에 저장된 값을 보면 그림 3.2와 같다. 클라우드 게임에서 3D 렌더링은 항상 이루어져야 하기 때문에 렌더링이 발생할 때마다 버퍼에 아바타 정보만 저장하면 되므로 추가적인 연산 없이 아바타를 추출할 수 있고, 동적으로 변하는 아바타의 위치나 모양도 정확하게 추출할 수 있는 장점이 있다.





Original video

Stencil buffer

Figure 3.2: 스템실 버퍼에 저장된 ROI 정보.

### 3.1.3 Multi-layered ROI Extraction

스텐실 버퍼를 통하여 아바타 정보를 추출함으로써 아바타의 위치 및 모양을 추출하는 것이 가능하다. 하지만 단순히 ROI와 non-ROI가 나뉘면 아바타 사이의 인터렉션(interaction)을 고려할 수 없다. 사용자는 아바타 사이의 인터렉션에도 집중하므로 아바타를 모두 포함하는 영역을 새로 구성하여 ROI로 설정하였다. 이를 위해 영상 처리 기법인 Image dilation [17]을 활용하였다. 그림 3.3을 보면 그림 3.3(a)는 원본 영상이고 이를 3D 렌더링 한 이후에 스템실 버퍼를 통해 아바타를 추출하게 된다. 이 때, 스템실 버퍼는 픽셀 단위의 버퍼이기 때문에 H.264/AVC로 인코딩 하기 위해 가로, 세로 각각 16픽셀로 구성되는 매크로블록 단위로 변환한다. 픽셀 단위의 스템실 버퍼를 매크로 블록 단위로 변환한 것이 그림 3.3(b)이다. 다음으로, 모든 매크로 블록에서 상, 하, 좌, 우  $K$  만큼 image dilation을 수행한 결과가 그림 3.3(c)이다. Image dilation 을 통하여 모든 아바타를 포함하는 영역을 구성할 수 있다. 3D 렌더링 정보를 통해 ROI 영역을 추출하였고, 이를 영상처리 기법 Image dilation 을 통해 영역을 구분하여 스템실 버퍼를 통해 추출하는 아바타

영역을  $\alpha$  영역, 아바타를 포함하는 확장된 영역을  $\beta$  영역, 그리고 나머지 영역을  $\gamma$  영역으로 명명하였다.

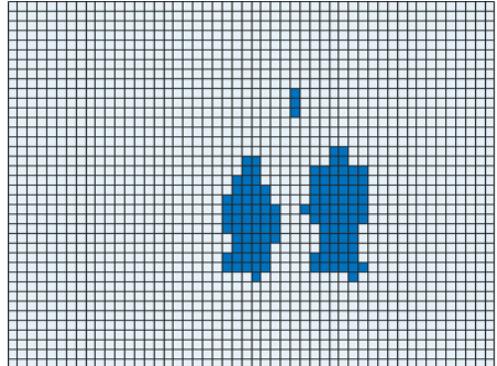
하지만 image dilation의 크기  $K$ 가 고정되어 있다면 화면에 아바타가 어떻게 구성되어 있는지에 따라 모든 아바타를 포함하지 못 할 수도 있다. 따라서 Zoom in/out을 통해 아바타의 크기가 달라지는 경우를 고려하여 아바타의 크기에 따라 image dilation의 크기  $K$ 를 조절하여 Image dilation을 수행하도록 하였다. 그림 3.4은 Zoom의 정도에 따라  $K$  값을 변경하여  $\beta$  영역을 표시한 결과이다. Zoom in을 통해 아바타가 크게 나타날수록  $K$  값을 낮추고, Zoom out을 통해 아바타가 작게 나타날수록  $K$  값을 크게 하여 ROI 확장이 이루어 지도록 구성하였다.

하지만, 프레임 상에 나타나는 아바타가 너무 작게 나타나거나 사용자가 보기에 멀리 있다고 판단 되는 경우, ROI라고 정의할 수 없다. 스텐실 버퍼만 활용하면 아무리 작게 나타나는 아바타라도 ROI로 인식하기 때문에 이를 ROI에서 제외 시킬 수 있도록 깊이 버퍼를 활용하였다. 그림 3.5는 렌더링 이후에 깊이 버퍼에 저장되는 값을 나타낸 것이다. 깊이 버퍼는  $[0,1]$ 의 실수 값을 가지는 픽셀 단위의 버퍼이다. 깊이 버퍼를 통해 Threshold 값을 설정하고 이보다 면 값을 가지는 아바타는 ROI에서 제외하도록 구성하였다.

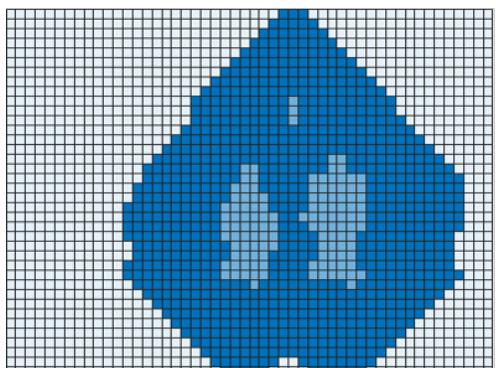




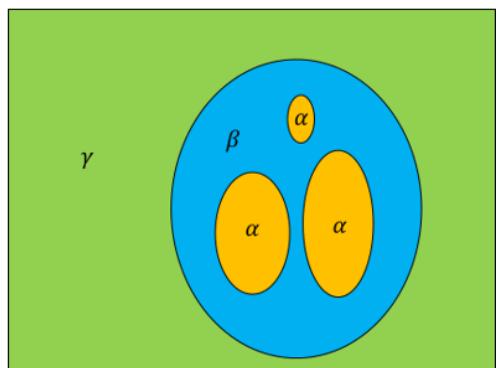
(a)



(b)



(c)



(d)

Figure 3.3: Image dilation 을 통한 ROI 확장: (a) 원본 영상, (b) 스텐실 버퍼, (c) Image dilation, (d) 영역 구분.



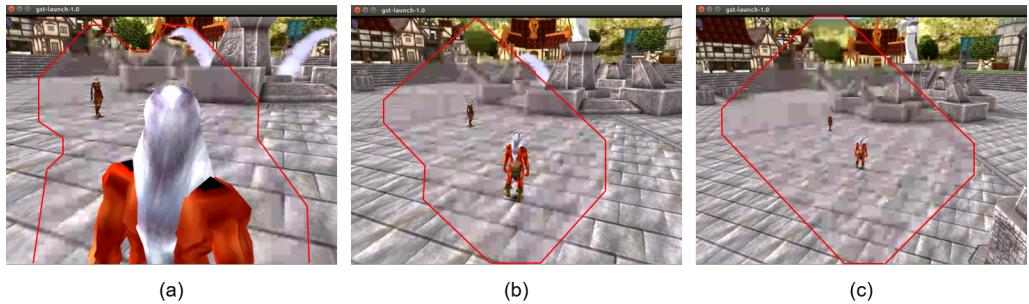


Figure 3.4: Zoom in/out ㅇ] 따른 Image dilation level  $K$  : (a)  $K = 8$ , (b)  $K = 12$ , (c)  $K = 16$ .

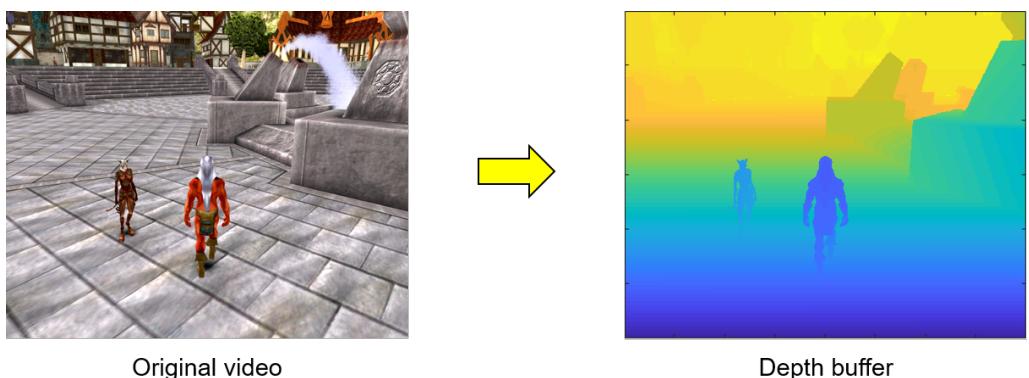


Figure 3.5: 깊이 버퍼에 저장된 정보.



렌더링 정보를 바탕으로 하는 ROI-based Bitrate Allocation 알고리즘은 Algorithm 1과 같다. 스텐실 버퍼, 깊이 버퍼, 그리고 Zoom level과 같은 렌더링 정보가 입력값으로 들어가며 이에 따라 각 매크로 블록에 정해지는 Quantization Parameter (qp) [18] 값이 결과값이다. qp 값이 클수록 압축률이 높고 화질이 많이 열화되므로, ROI 영역에 해당하는 매크로 블록은 qp 값을 다른 영역보다 낮추어야 한다. 먼저 Zoom level  $Z$ 에 따라 image dilation level  $K$ 를 결정한다. 그리고 스텐실 버퍼, 깊이 버퍼는 픽셀 단위로 저장되기 때문에 이를 인코딩의 기본 단위인 매크로 블록 단위로 변환 한다. 가로, 세로 각각 16 픽셀을 가지는 매크로 블록 안에 ROI로 지정된 픽셀이 있으면 ROI 매크로 블록으로 지정함으로써 매크로 블록 단위로 변환한다. 이를 바탕으로 매크로 블록이 ROI에 속하는지 아닌지를 구분한다. 그리고 Image dilation level  $K$  만큼 수행하여  $\alpha$ 를 포함하는  $\beta$  영역을 만들고, 모든 매크로 블록이  $\alpha, \beta, \gamma$  영역이 구분이 되므로 각 매크로 블록에 할당되는 qp 값에 가중치  $\omega_\alpha, \omega_\beta, \omega_\gamma$ 를 통해 qp 값을 정하고 이를 통해 인코딩 한다. 이러한 방식으로 프레임마다 비트레이트 할당이 이루어지게 된다.



---

**Algorithm 1** ROI-based Bitrate Allocation.

---

1: **Input:**

2:  $\vec{S} = (S_1, \dots, S_i, \dots, S_{N_{pixel}}), \vec{D} = (D_1, \dots, D_i, \dots, D_{N_{pixel}}), Z$

3: **Output:**

4:  $\vec{q} = (q_1, \dots, q_i, \dots, q_{N_{blk}})$

5: **Begin:**

6: Determine  $\alpha$  based on  $\vec{S}$  and  $\vec{D}$

7: Convert pixel-wise of  $\vec{S}$  to MB-wise of  $\vec{s} = (s_1, \dots, s_i, \dots, s_{N_{blk}})$

8: Determine dilation level  $K$  based on  $Z$

9: **for** (each  $j$ ,  $j \in K$ ) **do**

10:   **for** (each  $i$ ,  $i \in N_{blk}$ ) **do**

11:     Determine  $MB_i$  is in region  $\alpha, \beta$  or  $\gamma$

12:   **end for**

13: **end for**

14: **for** (each  $i$ ,  $i \in N_{blk}$ ) **do**

15:   **if**  $MB_i$  is in region  $\alpha$  **then**

16:      $q_i = \omega_\alpha \cdot q_i$

17:   **else if**  $MB_i$  is in region  $\beta$  **then**

18:      $q_i = \omega_\beta \cdot q_i$

19:   **else**

20:      $q_i = \omega_\gamma \cdot q_i$

21:   **end if**

22: **end for**

23: **End**

---



Table 3.2: Description of key symbols for Encoder/Decoder buffer occupancy-aware piece-wise CBR bandwidth estimation

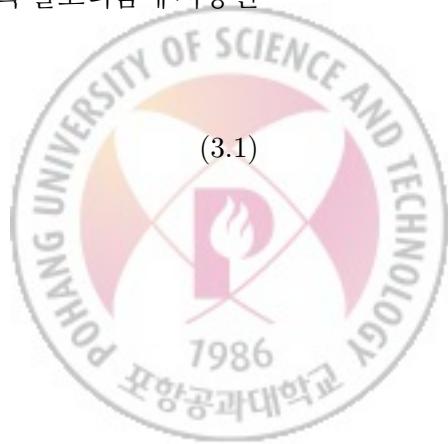
Symbol	Description
$c$	Measured channel rate
$c^{EWMA}$	Smoothed channel rate using exponentially weighted moving average
$b^{target}$	Target bitrates
$B^{enc}$	Encoder buffer occupancy
$B^{dec}$	Decoder buffer occupancy
$TH$	Threshold ( $L$ : Low, $H$ : High, $\Delta$ : delta)
$\Delta N_d$	Delay for display
$count$	The number of encountered same cases

### 3.2 Encoder/Decoder buffer occupancy-aware piece-wise CBR bandwidth estimation

인코더 버퍼 언더플로우가 발생하면 대역폭 utilization이 떨어지고, 디코더 버퍼 언더플로우가 발생하면 freezing이 발생하기 때문에, 따라서 버퍼 상태를 고려하여 목표 비트레이트 할당이 이루어져야 한다. 제안하는 Encoder/Decoder buffer occupancy-aware piece-wise CBR bandwidth estimation 알고리즘은 시간에 따라 CBR로 목표 비트레이트를 설정하기 때문에 인코더 버퍼 점유율 및 디코더 버퍼 점유율을 계산할 수 있다 [19]. 표 3.2를 보면 제안하는 대역폭 예측 알고리즘에 사용된 기호들에 대한 정의를 나타낸다.

$$B_i^{enc} = \sum_{j=1}^i b_j^{target} - \sum_{j=1}^i c_j$$

(3.1)



$$B_i^{dec} = \sum_{j=1}^i c_j - \sum_{j=1}^{i-\Delta N_d} b_j^{target} \quad (3.2)$$

식 3.1을 보면  $i$  번째 인코더의 버퍼 점유율은 처음부터  $i$  번째까지 인코딩된 비트레이트의 합에서 처음부터  $i$  번째까지 측정된 대역폭을 빼서 구할 수 있다. 디코더 버퍼 점유율의 경우  $\Delta N_d$  만큼의 디스플레이를 위한 latency가 발생하므로  $j$  번째 까지의 측정된 대역폭의 합에서  $i - \Delta N_d$  까지 인코딩된 비트레이트를 빼서 구할 수 있다. 식 3.3을 보면 인코더 버퍼 언더플로우를 막기 위해서는 식 3.1이 0보다 커야 하는데, 이 때  $i$  번째 대역폭은 예측을 통해 구함으로써 인코더 버퍼 언더플로우를 막는  $i$  번째 목표 비트레이트를 구할 수 있다. 따라서  $i$  번째 목표 비트레이트의 조건은 식 3.4와 같다.

$$B_i^{enc} = \sum_{j=1}^i b_j^{target} - \sum_{j=1}^i c_j \simeq B_{i-1}^{enc} + b_i^{target} - c_{i-1} > 0 \quad (3.3)$$

$$b_i^{target} > c_{i-1} - B_{i-1}^{enc} \quad (3.4)$$

$$B_{i+\Delta N_d}^{dec} = \sum_{j=1}^{i+\Delta N_d} c_j - \sum_{j=1}^i b_j^{target} \simeq \sum_{j=1+1}^{i+\Delta N_d} c_j - B_i^{enc} > 0 \quad (3.5)$$

$$B_i^{enc} < \sum_{j=1+1}^{i+\Delta N_d} c_j \quad (3.6)$$

디코더 버퍼 점유율도 마찬가지로 위와 같은 방식으로 구할 수 있다. 즉,  $i+1$ 부터  $i+\Delta N_d$  까지의 측정되어지는 대역폭의 합이  $i$  번째 인코더 버퍼 점유율보다 클 때, 디코더 버퍼 언더플로우를 막을 수 있다. 하지만  $i$  번째 목표 비트레이트를 결정할 때 미래  $\Delta N_d$  동안 프레임이 어느 정도의 크기로 인코딩 될 지는 알 수가 없다. 따라서 예측값을 통해서 디코더 버퍼 언더플로우를 예방해야 하는데, 제안하는 대역폭 예측 알고리즘은 인코더 버퍼 점유율의 upper bound, lower bound, 그리고 변동폭 제한인 delta bound 를 통해 디코더 버퍼 언더플로우 또한 예방하고자 하였다.

Algorithm 2는 제안하는 Encoder/Decoder buffer occupancy-aware piece-wise CBR 대역폭 예측 알고리즘을 나타낸다. 인코더 버퍼 상태가  $TH$ 의 lower bound, upper bound 를 넘어서지 않을 것으로 예측되면 target bitrate 를  $i - 1$ 번째와 동일하게 유지하며 CBR 로 인코딩 되도록 한다. 하지만  $TH$  를 벗어날 것으로 예측되면  $i$  번째 비트레이트를 증가 혹은 감소 하여 인코더/디코더 버퍼 언더플로우를 막으면서 목표 비트레이트가 수정되도록 구성하였다.



---

**Algorithm 2** Encoder/Decoder buffer occupancy-aware piece-wise CBR bandwidth estimation.

---

```

1: Input:  $c_{i-1}, c_{i-1}^{EWMA}, c_{i-1}^{target}, B_{i-1}^{enc}, TH_L, TH_H, TH_\Delta$ 
2: Output:  $b_i^{target}$ 
3: Begin:
4:  $c_i^{EWMA} = \alpha \cdot c_{i-1} + (1 - \alpha) \cdot c_{i-1}^{EWMA}$ 
5:  $\hat{B}_i^{enc} = B_{i-1}^{enc} + b_{i-1}^{target} - c_{i-1}$ 
6: if  $TH_L < \hat{B}_i^{enc}$  and  $\hat{B}_i^{enc} < TH_H$  then
7:    $b_i^{target} = b_{i-1}^{target}$ 
8: else if  $\hat{B}_i^{enc} < TH_L$  then
9:   if  $cur\_case == last\_case$  then
10:     $count = count + 1$ 
11:   else
12:     $count = 1$ 
13:   end if
14:   if  $c_i^{EWMA} - b_{i-1}^{target} > TH_\Delta$  then
15:      $b_i^{target} = b_{i-1}^{target} + TH_\Delta \cdot count$ 
16:   else
17:      $b_i^{target} = c_i^{EWMA}$ 
18:   end if
19: else
20:   if  $cur\_case == last\_case$  then
21:      $count = count + 1$ 
22:   else
23:      $count = 1$ 
24:   end if
25:   if  $b_{i-1}^{target} - c_i^{EWMA} > TH_\Delta$  then
26:      $b_i^{target} = b_{i-1}^{target} - TH_\Delta \cdot count$ 
27:   else
28:      $b_i^{target} = c_i^{EWMA}$ 
29:   end if
30: end if
31: End
```



## IV. 실험 결과

제안하는 ROI 기반의 비트레이트 할당 및 채널 예측 알고리즘의 성능 검증을 위해 그림 4.1과 같이 실제 테스트베드로 구축하였다. 제안하는 시스템의 클라우드 서버는 Ubuntu 16.04 운영체제를 사용하여 구축하였고, 테스트 게임은 3D 오픈소스 렌더링 엔진 Crystal Space기반으로 만들어진 3D 오픈소스 게임 PlaneShift [20]를 사용하였다. 압축 표준으로는 H.264/AVC [18]를 적용하여 x264 [21] 오픈소스 프레임워크를 사용하였으며, 압축된 영상을 전송하고 디스플레이 하기 위해 Gstreamer [22]를 활용하여 클라이언트로 전송할 수 있도록 구성하였다. 네트워크 환경은 Wi-Fi 상에서, 클라이언트는 VirtualGL [23]을 통해 게임 제어를 위한 입력값을 받아들이고 Gstreamer를 통해 비디오 스트림을 디스플레이 할 수 있도록 구성하였다. 기존의 H.264/AVC 압축 방식과 성능을 비교 및 분석하기 위하여 800 x 600 해상도의 영상을 초당 25 개의 프레임율을 가지도록 인코딩 하였다. 실험 영상은 300 프레임, 즉 12초 영상을 사용하였으며, ROI 기반의 인코딩을 위한 가중치는 실험을 통하여  $\omega_\alpha = 0.93$ ,  $\omega_\beta = 1.01$ ,  $\omega_\gamma = 1.13$  으로 설정하였다.



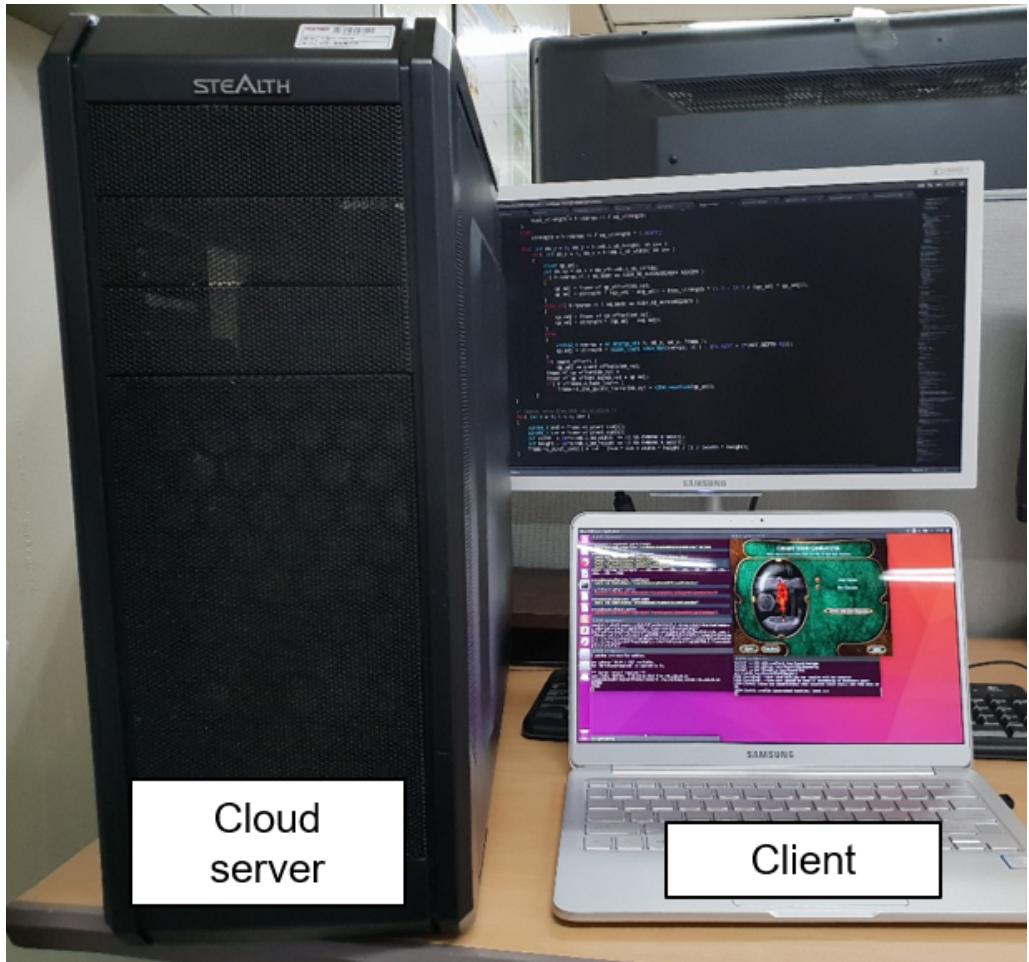


Figure 4.1: 구축된 테스트베드 사진.



## 4.1 Performance Comparison of Bitrate Allocation Algorithms

본 절에서는 제안하는 ROI-based bitrate allocation 알고리즘과 기존의 방법으로 인코딩 했을 때의 결과를 비교 및 분석함으로써 제안하는 알고리즘의 성능을 검증하였다. 그림 4.2는 기존의 방법과 ROI 기반의 제안하는 방법의 인코딩 결과를 비교한 이미지이다. 빨간색으로 표시된 부분을 비교해 보면 제안하는 시스템이 ROI로 설정된 아바타에 더 많은 비트레이트를 할당함으로써 기존의 방법보다 더 좋은 화질을 보이는 것을 확인할 수 있다. 반면, 파란색으로 표시된 non-ROI를 비교해 보면 제안하는 시스템에서 비트레이트를 덜 할당하기 때문에 화질이 더 좋지 않은 것을 확인할 수 있다. 따라서 동일한 목표 비트레이트로 인코딩된 이미지라도 ROI에 더 많은 비트레이트를 할당한 제안하는 시스템의 시각적인 화질이 더 향상 되는 것을 확인할 수 있다. 그림 4.3은 그 결과를 PSNR을 통해 분석해본 결과이다. 그림 4.2(a)를 보면 제안하는 시스템이 기존의 시스템보다 전체적으로 PSNR이 더 낮은 것을 확인할 수 있다. 하지만, 그림 4.2(b)에서  $\alpha$ ,  $\beta$ , 그리고  $\gamma$  영역 별로 PSNR을 확인한 결과  $\alpha$  영역의 PSNR이 가장 높은 것을 확인할 수 있다. 이를 통해 제안하는 시스템이 시각적 화질을 높이도록 영역별로 비트레이트가 할당된 것을 확인할 수 있다.



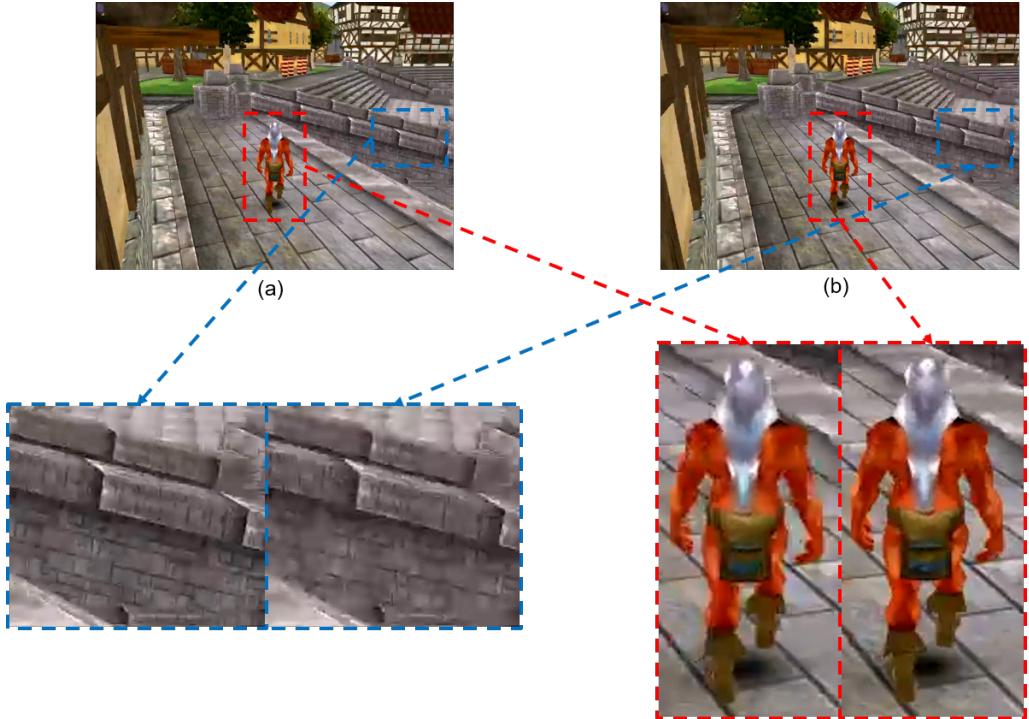


Figure 4.2: 동일한 비트레이트로 인코딩된 이미지 비교 : (a) 기존 H.264/AVC  
 (b) ROI-based H.264/AVC.

## 4.2 Performance Comparison of Channel Estimation Algorithms

이 절에서는 제안하는 인코더/디코더 버퍼 상태를 고려한 piece-wise CBR 대역폭 예측 알고리즘과 다른 알고리즘의 버퍼 점유율 및 PSNR 을 비교 및 분석하였다. 비교를 위해 iperf3 [24]를 활용하여 실제 테스트베드 환경에서 네트워크 상태를 측정하였다. 그림 4.4는 측정된 대역폭과 그에 따른 목표 비트레이트를 나타낸다. 그림 4.4(a)는 측정되어지는 대역폭에 따라 곧 바로 목표 비트레이트를 업데이트하는 방식이며 그림 4.4(b)는 Exponentially Weighted Moving Average (EWMA)

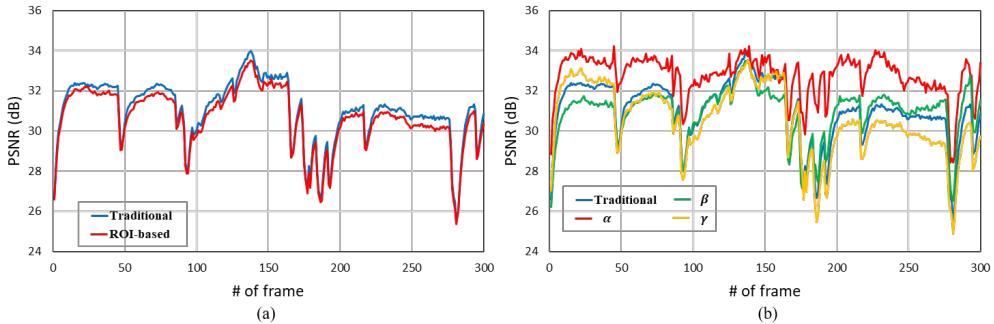


Figure 4.3: PSNR 비교 : (a) 기존 vs ROI PSNR 비교 (b) 영역별 PSNR 비교.

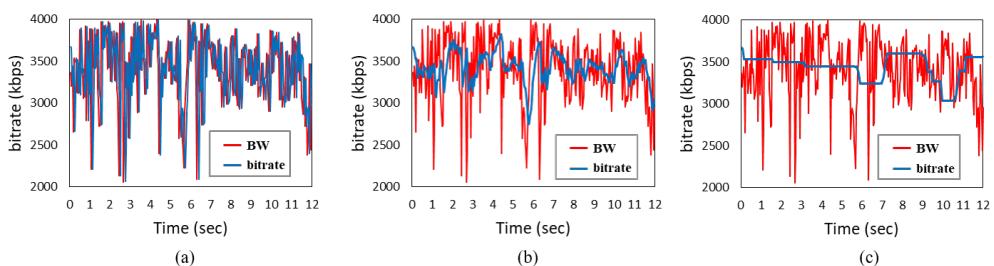


Figure 4.4: 목표 비트레이트를 설정하는 알고리즘 비교 : (a) BW following, (b) EWMA, (c) Proposed.

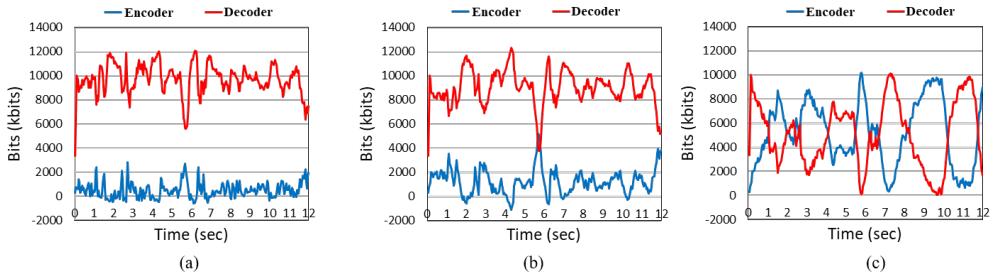


Figure 4.5: 인코더/디코더 버퍼 점유율 비교 : (a) BW following, (b) EWMA, (c) Proposed.

[25] 를 통해 예측된 결과를 목표 비트레이트로 설정하는 알고리즘이다. 그림 4.4(c)는 제안하는 알고리즘으로서 인코더 버퍼의 상태를 예측하고 인코더 및 디코더 버퍼 underflow 를 막을 수 있도록 목표 비트레이트를 수정하는 piece-wise CBR 알고리즘이다.

그림 4.5는 목표 비트레이트를 통해 인코딩이 되었을 때, 인코더 및 디코더의 버퍼 점유율을 나타낸다. 그림 4.5(a)는 디코더의 버퍼 언더플로우는 발생하지 않지만 인코더의 버퍼 언더플로우가 발생하여 네트워크 utilization 을 떨어뜨리는 것을 확인할 수 있다. 그림 4.5(b) 또한 버퍼 상태를 고려하지 않고 EWMA의 결과를 바탕으로 인코딩 하기 때문에 인코더 버퍼 언더 플로우가 발생한다. 마찬가지로 인코더 버퍼 언더플로우는 대역폭 utilization 을 떨어뜨리게 된다. 그림 4.5(c)는 인코더의 버퍼 상태를 보고 버퍼 언더플로우를 예방하도록 목표 비트레이트를 설정하기 때문에 인코더와 디코더 언더 플로우 모두 발생하지 않는 것을 확인할 수 있다.



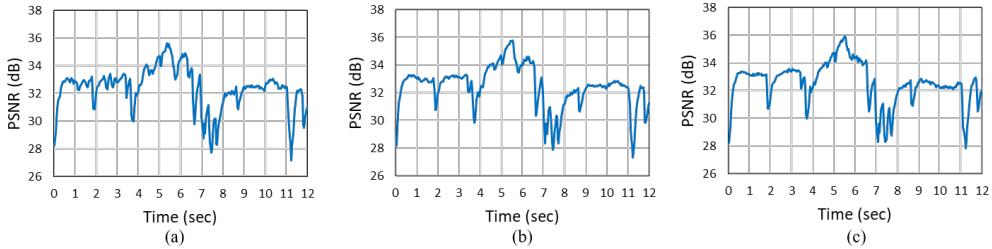


Figure 4.6: PSNR 비교 : (a) BW following, (b) EWMA, (c) Proposed.

Table 4.1: 채널 예측 알고리즘 PSNR 및 버퍼 상태 비교

	BW following	EWMA	Proposed
<b>PSNR Average</b>	32.66	32.42	32.54
<b>PSNR Variance</b>	2.37	2.39	2.28
<b>Encoder buffer underflow</b>	68	35	0
<b>Decoder buffer underflow</b>	0	0	0

그림 4.6는 목표 비트레이트에 따른 PSNR 을 나타낸다. 그림 4.6(a)는 대역폭 변동을 그대로 반영하여 목표 비트레이트를 설정하기 때문에 인접한 프레임끼리도 PSNR 변동이 비교적 심한 것을 확인할 수 있다. 그림 4.6(b)는 EWMA를 통해 평활화하여 인코딩을 수행하지만 PSNR 분산이 여전히 높게 나타나며, 그림 4.6(c)는 특정 간격에서 CBR 로 인코딩 되기 때문에 비교적 PSNR 변동을 최소화 할 수 있다. 표 4.1은 이를 바탕으로 비교한 결과를 나타낸다. BW following과 EWMA로 인코딩 했을 때 PSNR의 분산이 높았으며, 제안하는 알고리즘은 가장 낮은 PSNR 분산 값을 보였다. 반면, BW following과 EWMA는 인코더 버퍼 언더플로우가 여러번 발생하였다. 이를 통해 제안하는 알고리즘이 버퍼 언더플로우를 예방하면서 화질의 심한 변동을 막을 수 있음을 확인하였다.

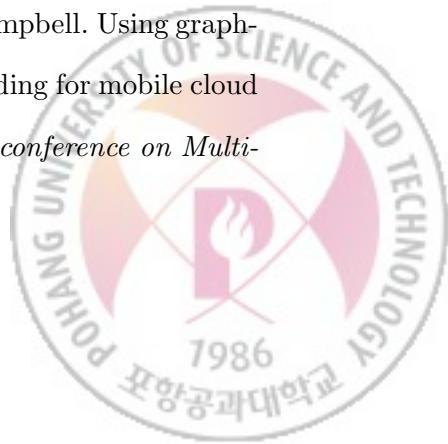
## V. 결론 및 향후과제

본 논문은 대역폭이 제한되어 있고 변동이 심한 무선 네트워크에서 클라우드 게임의 시각적 화질을 향상시키기 위한 위한 ROI 기반의 효율적 비트레이트 할당 기법 및 인코더/디코더 버퍼를 고려한 시간에 따라 변하는 고정 비트레이트 채널 예측 알고리즘을 제안하였다. 스텐실 버퍼를 활용하여 프레임을 렌더링 할 때마다 렌더링 엔진으로부터 ROI 정보를 추출하였고 Image dilation 영상 처리 기법을 통하여 ROI를 확장하였다. 이에 따라 알파, 베타, 그리고 감마 영역의 가중치를 달리 하여 비트레이트를 할당하였다. 또한 네트워크 상태를 주기적으로 모니터링 하면서 인코더 및 디코더의 버퍼 상태를 고려하여 타겟 비트레이트를 변경하는 방식으로 piece-wise CBR 대역폭 예측 기법을 제안하였다. 실험결과, non-ROI 보다 ROI에 더 많은 비트레이트를 할당함으로써 기존의 H.264/AVC 방법보다 시각적 화질이 높아지는 것을 확인하였으며, 또한 대역폭을 BW following 및 EWMA를 통해 예측된 결과로 인코딩하는 방법과 비교함으로써 제안하는 방법이 인코더/디코더 버퍼 언더플로우를 예방함과 동시에, 비디오 품질의 변동폭을 최소화 하는 것을 확인할 수 있었다. 제안하는 시스템은 오픈소스 3D 게임 및 렌더링 엔진을 통해 실제 테스트베드를 구축하여 성능 검증에 활용되었다.

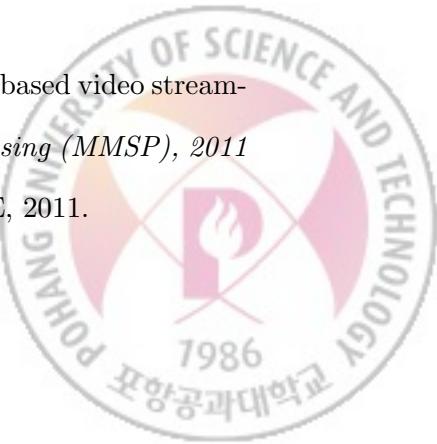
본 연구를 바탕으로 각 영역별 다른 가중치를 적용하여 ROI를 강조하는 비트레이트 할당이 가능함을 보였다. 하지만 목표 비트레이트에 따라 영역별로 적용되는 가중치 값이 적응적으로 바뀐다면 시각적으로 더 향상된 결과를 보일 수 있다. 뿐만 아니라, 게임은 장르에 따라서도 ROI가 다르기 때문에 장르에 따라 ROI를 동적으로 변화시키는 방법 또한 향후 연구 과제이다.

## 참 고 문 현

- [1] TechNavio Infiniti Research Ltd. Global cloud gaming market 2018-2022. 2018.
- [2] Project Stream. A game streaming service, 2018.
- [3] PlayFab. Backend platform for live games, 2018.
- [4] Scarlett. Xbox scarlett cloud, 2018.
- [5] Bogdan Ciubotaru, Gheorghita Ghinea, and Gabriel-Miro Muntean. Subjective assessment of region of interest-aware adaptive multimedia streaming quality. *IEEE Transactions on Broadcasting*, 60(1):50–60, 2014.
- [6] Wei Cai, Ryan Shea, Chun-Ying Huang, Kuan-Ta Chen, Jiangchuan Liu, Victor CM Leung, and Cheng-Hsin Hsu. A survey on cloud gaming: Future of computer games. *IEEE Access*, 4:7605–7620, 2016.
- [7] Yao Liu, Sujit Dey, and Yao Lu. Enhancing video encoding for cloud gaming using rendering information. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(12):1960–1974, 2015.
- [8] Shu Shi, Cheng-Hsin Hsu, Klara Nahrstedt, and Roy Campbell. Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 103–112. ACM, 2011.



- [9] Shaoxuan Wang and Sujit Dey. Adaptive mobile cloud computing to enable rich mobile multimedia applications. *IEEE Transactions on Multimedia*, 15(4):870–883, 2013.
- [10] Donghyeok Ho, Hyungnam Kim, Wan Kim, Youngho Park, Kyung-Ah Chang, Hyogun Lee, and Hwangjun Song. Mobile cloud-based interactive 3d rendering and streaming system over heterogeneous wireless networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(1):95–109, 2017.
- [11] Mehdi Semsarzadeh, Abdulsalam Yassine, and Shervin Shirmohammadi. Video encoding acceleration in cloud gaming. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(12):1975–1987, 2015.
- [12] Yao Liu, Shaoxuan Wang, and Sujit Dey. Content-aware modeling and enhancing user experience in cloud mobile rendering and streaming. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 4(1):43–56, 2014.
- [13] Kuan-Ta Chen, Yu-Chun Chang, Po-Han Tseng, Chun-Ying Huang, and Chin-Laung Lei. Measuring the latency of cloud gaming systems. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1269–1272. ACM, 2011.
- [14] Nicolas Tizon, Christina Moreno, and Marius Preda. Roi based video streaming for 3d remote rendering. In *Multimedia Signal Processing (MMSP), 2011 IEEE 13th International Workshop on*, pages 1–6. IEEE, 2011.



- [15] Zhewei Zhang, Tao Jing, Jingning Han, Yaowu Xu, and Fan Zhang. A new rate control scheme for video coding based on region of interest. *IEEE Access*, 5:13677–13688, 2017.
- [16] Bogdan Ciubotaru, Gabriel-Miro Muntean, and Gheorghita Ghinea. Objective assessment of region of interest-aware adaptive multimedia streaming quality. *IEEE Transactions on Broadcasting*, 55(2):202–212, 2009.
- [17] Arthur R Weeks. *Fundamentals of electronic image processing*. SPIE Optical Engineering Press Bellingham, 1996.
- [18] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.
- [19] Ming-Ting Sun. *Compressed video over networks*. CRC Press, 2000.
- [20] PlaneShift. Open source 3d mmorpg, 2018.
- [21] x264. Free and open-source software library for encoding video streams into the h.264/mpeg-4 avc, 2018.
- [22] Gstreamer. Open source multimedia framework, 2018.
- [23] VirtualGL. Open source program that redirects the 3d rendering command, 2018.
- [24] Iperf3. Network performance measurement tool, 2018.



- [25] J Stuart Hunter. The exponentially weighted moving average. *Journal of quality technology*, 18(4):203–210, 1986.



## 감사의 글

먼저 결코 짧지 않았던 2년이라는 시간 동안 늘 동행하시고 지켜주신 하나님 아버지께 감사드립니다. 너무나도 부족한 제가 졸업까지 잘 마무리할 수 있었던 모든 것이 하나님의 은혜입니다. 앞으로도 하나님을 의지하면서 문제를 해결하는 공학자가 되도록 열심히 노력하겠습니다.

훌륭하신 교수님께서 지도해주셨기 때문에 정말 많이 성장할 수 있었습니다. 아낌없는 지도와 조언으로 저를 이끌어주신 송황준 교수님께 진심으로 감사드립니다. 교수님께서 보이셨던 연구에 대한 열정을 늘 기억하며 제가 하는 일에도 열정을 가지고 최선을 다하도록 노력하겠습니다. 그리고 바쁘신 가운데, 흔쾌히 석사 논문 심사에 참여해주신 박찬익 교수님, 이승용 교수님께도 감사의 말씀 전하고 싶습니다.

혼자라면 하지 못 했겠지만, 연구실 선배들, 후배들이 함께 했기에 무사히 마무리 할 수 있었다고 생각합니다. 바쁘신 가운데도 도움을 구할 때면 정확하게 이해하시고 제가 미처 생각하지 못했던 부분까지 성심 성의껏 도와주신 윤민이형, 힘들 때 제 이야기도 많이 들어주시고 먼저 도움주시고 공감해주셨던 기석이형 진심으로 감사드립니다. 동갑이지만 1년 선배로서, 연구실에 잘 적응하도록 도움을 준 환욱이, 상욱이에게도 감사의 인사를 전합니다. 또한, 지난 2년간 가장 많은 시간을 함께 했던 현민이형께도 감사하다고 전하고 싶습니다. 특히, 같이 맛있는 음식 먹으면서 스트레스 해소하던 것들이 많이 기억날 것 같습니다. 이제 건강관리도 유의하셔서, 앞으로 남은 박사과정도 좋은 연구 많이 하시고 좋은 결과 있기를 바랍니다. 마지막으로, 연구실 후배 희승이형, 재준이, 그리고 승환이에게도 감사의 인사를 전합니다. 서로 도와 주면서 각자가 하는 연구에도 좋은 결과 있기를 바랍니다.

마지막으로 사랑하는 우리 가족에게도 감사의 말씀 전해드립니다. 손주들이 미안하지 않도록 끝까지 병마와 싸우시며 모든 손주들 만나고 편안하게 천국가신 외할머니, 외할머니 간호 하시느라 몸도 마음도 많이 지친 가운데도 늘 기도해주시고 격려해주시며 힘이 되주셨던 어머니, 그리고 듬직하게 저를 믿어주시고 응원해주신 아버지, 사역하면서도 늘 힘내라고 격려해준 우리 형까지 모두 사랑하고 감사드립니다. 또한, 오랜 기간 장거리 연애로 인해 힘들텐데도 대학원 생활까지 믿고 지지해주며 석사과정동안 정말 저에게 큰 힘이 되어준 사랑하는 예솔이에게도 감사의 마음 전합니다.

결코 쉽지만은 않았던 2년의 시간이 헛되지 않도록 앞으로도 최선을 다하여, 이렇게 저를 믿고 지지해준 분들의 기대에 보답하여 더욱 열심히 살도록 노력하겠습니다. 정말 감사드립니다.

2019년 1월

박고언 올림



# 이력서

Name : Goeon Park

## Education

2010 – 2017	Department of Computer Science and Electrical Engineering, Handong Global University (B.S.)
2017 – 2019	Department of Computer Science and Engineering, Pohang University of Science and Technology (M.S.)

## Affiliation

1. Media Computing and Networking Lab., Department of Computer Science and Engineering, Pohang University of Science and Technology



