TV혼합현실을 위한 RGB Segmentation 최적화 및 분산처리 방법

*YeongHwan Jeong **SeongBae Bang
*Korea University **KyungHee University
spitz7777@naver.com sungbae9023@hanmail.net

요 약

모바일 디바이스와 스마트 글래스의 확산/보급으로 다양한 AR/VR/MR 어플리케이션이 출시 되었으나, 사용자들이 일반적으로 가정에서 많이 사용하는 대형 Display인 TV에서 이용할 수 있는 AR/VR/MR 어플리케이션은 거의 없는 실정이다. TV 디바이스에서 이러한 영상처리 기법을 이용하려면 별도의 카메라나 CPU가 필요한데, 이를 이용하기 위해 사용자들이 별도의 투자를 하기 어렵기 때문이다. 본 논문에서는 사용자에게 가장 친숙한 모바일 디바이스와, 디지털 TV신호를 수신하기 위한 STB를 연동하여 다양한 AR/VR/MR 서비스를 이용할 수 있는 방법을 제안하고 실제 시스템 구현과 실험을 통해 제안된 방법의 품질 및 실시간성 향상을 검증한다.

.

1. 서론

최근 들어 Computer Vision, Image Processing 기술이 발전함에 따라, 다양한 AR/VR 서비스들이 쏟아지고 있다. 하지만 거의 대부분 제공되는 서비스는 스마트폰 혹은 스마트 글래스에서 제공되는 서비스이다. TV에서 이러한 서비스를 제공하기 위해서는 TV에 임베디드된 CPU가 탑재되거나 이를 support하기 위한 셋톱박스가 필요한데, 대부분의 이러한 단말은 단순히 미디어를 stream 받기위한 Thin client 단말인 경우가 대부분이다. 따라서 AR/VR/MR 서비스를 제공하기 위한 무거운 영상처리가 들어가는 애플리케이션은 지금까지 제공할 수 없었다. 하지만, 점점 AR/VR/MR 시장이 커지면서 사용자들에게 이러한 대형 display에서의 서비스 니즈(needs)는 증가하는 추세이다. 본 논문에서는 이러한 제한 상황에서 효과적인 분산처리와 스마트폰에서의 최적화된 matting 기법 처리를 이용하여 실시간으로 이러한 서비스를 제공할 수 있도록 한다.

Alpha Matting 기법은 제법 오래전부터 많이 연구되어 온 분야로, RGB segmentation은 $I = \alpha F + (1-\alpha)B$ 의 공식으로부터 시작한다. 알파맵으로 Foreground와 Background를 분리해 내는 방식인데, 일반적으로 RGB 색상차를 이용하여 Alpha map을 생성하면 Foreground에 hole이 생기거나, Background의 컬러값 변화로 인한 에러가 발생하는 등의 문제가 많이 발생하게 된다.

또한 이렇게 생성한 Alpha map을 포함한 RGBA[1]값을 전송하기 위해, 최근에는 WebM이나 MJPEG등에서 관련 표준을 발표하고 있지만, 지금까지 가정에 많이 퍼져있는 기존의 셋톱박스에서는 이러한 표준을 따르는 H/W decoder를 갖추고 있지 않는 경우가 대부분이다.

본 논문에서는 이상의 문제점을 해결하여 TV에서도 서버와 견줄 수 있는 performance를 내는 혼합현실 서비스를 실시간으로 제공할 수 있는 방안을 제안한다. 특히, 2장 2절에서는 TV-모바일 연동 환경에서의 MR(혼합현실)을 구현하기 위해 모바일에서 촬영한 영상을 실시간으로 사용자와 배경으로 분리하는 segmentation과 그 과정에서 정확히 분리 되지 않는 에러 부분의 정확도를 높이기 위한 방법들에 대한 방안을 제시하였으며, 2장 3절에서는 2장 2절에서 생성한 알파맵을 WebM등의 알파 채널의 코덱이 없는 legacy STB장비에서 사용할 수 있도록 전송하는 방법에 대해 기술 하였다. 그리고 2장 4절에서는 STB에서 WebGL을 이용하여 Overlay 형태로 AR서비스를 제공하는 방법에 대해 제시 하였다. 3장에서는 이러한 구조에 대한 성능 비교 실험을 하고 4장에서 본 논문에 대한 결론을 맺는다.

[1] Red Green Blue and Alpha: 여러 색상을 조합할 수 있는 RGB 세가지 색과 투명도(Alpha)를 포함한 수치.

2. 본문

2.1 분산 처리구조

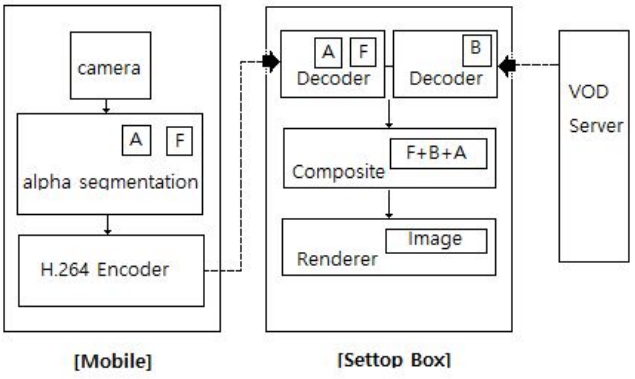

그림1. 전체 구조도

본 논문에서 제안하는 시스템의 전체 구조도는 위의 그림 1.과 같다. 먼저 사용자 혼합현실을 구성하기 위해, Mobile camera로부터 입력받은 영상으로 사용자의 모습을 segmentation하기 위한 Alpha map을 생성한다. 여기서 Alpha map을 생성하는 방법은 2장 2절에서 상세히 다루도록 한다. 생성된 Alpha map과 camera로 입력받은 RGB 프레임은 stitching과정을 거쳐 하나의 프레임으로 생성한 후 H.264로 인코딩 된다. 여기서 stitching을 하는 이유와 방안에 대해서는 2장 3절에서 상세히 다루도록 한다. 이렇게 각 프레임의 인코딩이 끝나면 mobile에서의 Image processing과정은 완료되고 셋톱박스로 스트리밍을 하게

된다. 여기서 mobile과 셋톱박스의 통신은 latency를 최소화하기 위해 local network를 통해 socket통신 할 것을 권장한다. 해당 구조는 서버를 별도로 두지 않기 때문에 굳이 L4를 거칠 필요가 없다. 이렇게 스트리밍한 이미지를 셋톱박스에서 수신하게 되고, 수신받은 프레임에서 Alpha map과 camera로 받은 이미지(이것을 Foreground 이미지로 사용)를 분리하는 한편, Background로 사용할 영상을 VOD 서버로부터 동시에 수신한다. 이 후 잘라낸 Alpha map과 Foreground 이미지, Background 이미지를 composite한다. 이렇게 composite한 프레임을 렌더링하게 되면 TV MR서비스가 되고, mobile로부터 사용자의 영상만을 수신받아 크로마키 VOD와 composite하게 되면 TV AR서비스가 된다. 이 과정은 2장 4절에서 더 상세히 다루도록 한다. 이렇게 셋톱박스는 직접 composite하기에 부족한 Computing power를 보완하기 위해 연산을 많이 필요로 하는 프레임을 재 생성부분을 mobile로 적절히 분산시켰고, 이로 인해 프레임으로부터 사용자를 추출하고 원하는 VOD에 합성하는 전과정의 latency를 줄이고자 한다.

2.2 Alpha matting on Mobile phone

mobile에서 이뤄지는 Image processing과정은 크게 Image segmentation과 Hole filling과정으로 나누어 지는데, segmentation은 우선 색과 edge의 변화로 물체가 들어온 영역을 검출하고 Hole filling을 이용한 후처리 과정에서 오검출된 지역을 제거하면서 물체의 내부 hole들을 채우는 방식으로 진행된다.

segmentation과정은, Mobile camera에서 사용자의 Alpha map을 만들기 위해 먼저 사용자가 없는 reference 이미지를 저장해야 한다. reference 이미지는 실내의 미세한 조명 차이에 의한 RGB값 변화를 방지하기 위해 최초 30프레임을 저장하여, RGB, High saturation영역, edge영역을 평균낸다. [1][2]

그림2. Estimated reference image

segmentation과정은 3단계에 걸쳐 진행된다. 첫 번째는 알고리즘의 처리 속도를 높이기 위해 reference 이미지와 현재 이미지의 RGB의 총 변화량이 큰 지역들은 바로 Foreground 지역으로 분류한다. 여기서 실험적으로 RGB의 총 변화량이 250이상이면 변화량이 큰 지역으로 정하였다.[3]

두번째 단계는 그림자에 의한 RGB 변화를 고려한다.그림자는 reference 이미지에서 밝기값이 큰 영역일수록 현재 이미지에서 밝기값을 크게 낮춘다. 그러므로 그림자를 Foreground 지역에서 제하기 위해 본 논문은 현재 이미지와 reference 이미지를 비교하여 밝기가 높아진 지역과 낮아진 지역을 구분하고 밝기가 낮아진 지역에서는 reference 이미지의 밝기값에 따라 threshold를 다르게 주었다.밝기가 높아진 지역은 그림자에 의한 영향이 없으므로 낮은 threshold를 적용하였다.[4]

세 번째 단계에서는 edge 변화를 이용하여 Foreground를 검출하였다. edge의 변화를 이용하면 물체가 배경과 색이 비슷하더라도 edge 페턴이 다른 경우 Foreground 지역으로 검출할 수 있다. 그러므로 그림 3.과 같이 물체 내부는 색이 비슷하여 Foreground로 검출하지 못하는 경우에도 물체의 edge들은 검출할 수가 있다. 이렇게 edge 지역들이 검출되면 물체의 hole 지역들은 Foreground로 검출된 pixel들로 둘러싸여 이후의 Hole filling 과정으로 segmentation 결과를 보정할 수 있게 된다.

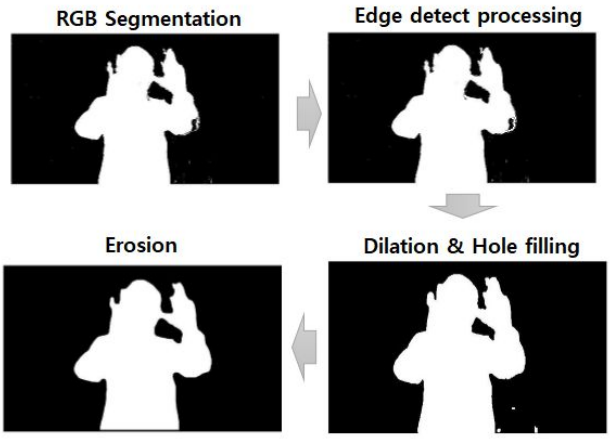

그림3. Alpha map 생성 과정

본 논문은 segmentation과정에서 발생하는 hole 지역들을 채우기 위하여 그림 3.과 같이 dilation 방법으로 물체의 edge 부분에서 segmentation된 pixel들이 완전히 이어지도록 하고 hole filling을 수행한다. 그리고 다시 두꺼워진 edge들을 erosion 방법으로 원본의 형태로 되돌린다. 그러면 segmentation과정에서 검출하지 못한 foreground 내부의 작은 hole들을 모두 채울 수 있다. 그러나 기존의 hole filling은 미검출된 hole뿐만아니라 물체의 의해 발생한 큰 hole들까지도 모두 채워버린다. 본 논문은 이러한 문제를 해결하기 위해 그림 4.와 같이 alpha map의 duplicate image를 2개 생성하고 각각의 duplicate image에 가로 방향으로 일정한 간격을 갖는 zero line을 설정한다. 여기서 두 duplicate image의 zero line들은 그림 4.와 같이 서로 다른 위치에 존재해야 한다. 그리고 본 논문은 이렇게 생성된 두개의 영상에 각각 hole filling을 하고 각각의 hole filling 결과에 or 연산을 수행한다. 그러므로 본 논문에서 개발한 hole filling방법은 zero line의 간격으로 채우고자하는 hole의 크기를 정할 수 있으므로 물체에 의해 발생한 큰 hole들은 채우지 않으면서 segmentation 과정에서 발생한 미검출된 작은 hole들만을 채울 수 있다.

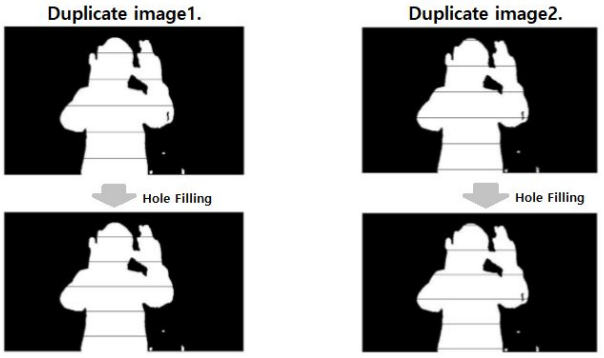

그림4. Duplicate-Hole filling 결과

Hole filling 과정 후에는 Hole filled된 Alpha map에

erosion을 다시 하여 원래 객체가 검출된 크기로 돌린 후 Alpha segmentation과정을 종료한다.

2.3 Alpha Channel Streaming

2장 2절과 같이 생성한 Alpha map을 원본과 함께 인코딩하여 송출해야 하는데, 가장 좋은 방법은 WebM과 같은 alpha 채널을 포함한 동영상 규격을 사용하는 것이다.[5] 하지만, 기존의 상용 셋톱박스에는 이러한 H/W decoder가 없고, S/W decoder를 사용하게 되면 latency가 늘어나 실시간 서비스를 제공하기가 어렵게 된다.

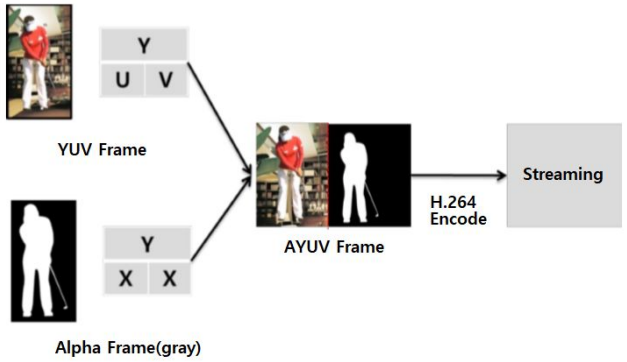

그림5. Alpha channel streaming 구조

따라서 본 논문에서 제안하는 서비스에서는 위의 그림 5.와 같이 프레임에 YUV프레임과 그에 해당하는 alpha프레임을 하나의 프레임으로 stitching 하여 송출한다.[6] segmentation 하고자 하는 사용자의 모습이 전체 프레임에 모두 나오는 것이 아니기 때문에 만약 전체 프레임이 16:9의 비율이라면, 사용자의 모습 중심으로 8:9 비율로 프레임을 자른다. 그러면 그에 해당하는 Alpha map도 같은 8:9의 비율로 나올 것이다. 이 두개의 8:9 프레임을 하나로 stitching 하여 16:9 비율의 하나의 프레임으로 만든 후, 셋톱박스로 보내기 위해 H.264로 encoding 한다.

2.4 Composite on Set top box

셋톱박스에서는 스마트폰과의 연동을 위해 Local network으로 Broad casting 메세지를 보낸다. 이 메세지 안에는 셋톱박스의 IP주소와 고유 serial 번호를 함께 보낸다. 스마트폰이 셋톱박스와 같은 network 안에서 이 메세지를 받으면 확인된 IP주소로 소켓통신을 시도한다.

스마트폰과 연결된 셋톱박스에서는 스마트폰으로부터 전송받은 AYUV프레임을 다시 두개의 YUV프레임과 alpha프레임으로 분리하는 한편, VOD서버로부터 합성할 Background영상을 함께 전송 받는다. 이렇게 세장의 프레임을 Composite 하게 되는데, Foreground의 영상은 Background 영상 대비 1/2의 크기기 때문에 이 영상의 위치 정보를 스마트폰으로부터 함께 받아야 한다. 해당 정보는 Content streaming을 받을 때, JSON형태로 위치 좌표값을 함께 받는다.

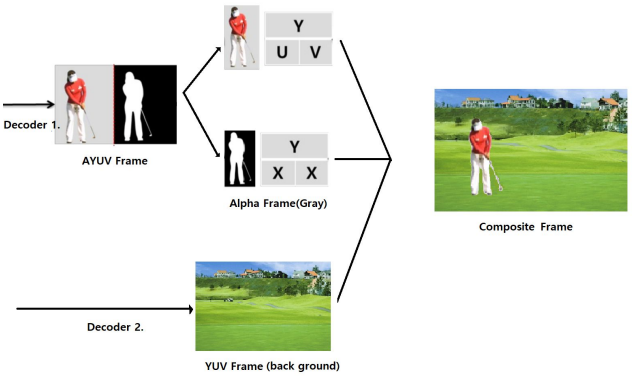

그림6. 셋톱박스의 Composite 과정

위의 그림 6.과 같이 decoder 1.으로 부터 받은 사용자 영상(Foreground)과 decoder 2.로부터 받은 VOD영상(Background)을 해당 Alpha map 정보로 composite 하게 된다. 셋톱박스의 WebGL(혹은 OpenGL)에서 YUV값을 RGB로 변환하여 합성하는데, Foreground 영역에는 Alpha map에 매칭되는 픽셀값을 곱하고, 좌표에서 벗어나는 영역은 모두 0으로 매칭한다. Background에도 마찬가지로 해당 좌표에 해당되는 영역은 (1-α)값을 곱하고, 좌표를 벗어나는 영역은 모두 1을 곱하여 배경영역을 보존한다. 이렇게 rendering 하면 TV속에 사용자가 직접 출현하는 듯한 가상 체험을 실시간으로 할 수 있게 된다.

3. 실험 및 결과

실험 환경은 스마트폰은 LG의 V50 모델(CPU : Qualcomm snapdragon 855, RAM : 6G, android : 9.0)을 사용하였고, 셋톱박스는 KT의 UHD2.0 셋톱(CPU : Dual B15, RAM 2G)를 사용하였고, 서버는 워크스테이션(CPU : i7-9700K, RAM : 32GB)를 사용하였다.

그림 7.에서 보는 바와 같이 RGB segmentation 만으로 생기는 많은 error들이 보정되는 모습을 볼 수 있으며, Foreground에 생긴 hole들이 Hole filling 하는 과정에서 채워지는 결과를 볼 수 있다.

그림7. Matting 결과

또한 아래의 표 1. 의 결과값은 본 논문에서 제안한 구조와, 서버 방식의 구조, 셋톱-Stand alone 방식의 latency를 비교한 결과이다. 서버방식의 구조는 스마트폰으로 촬영한 영상을 서버로 보내어 서버에서 segmentation 과정을 수행하고, 다시 셋톱으로 보내어 셋톱에서 composite하는 방식이고, 셋톱-Stand alone방식은, 스마트폰이나 캠으로 촬영한 영상을 셋톱으로 바로 보내어 셋톱에서 전 과정을 수행하는 방식이다.

일반적으로 서버 방식이나, 하나의 디바이스에서 모든 과정을 수행했을때 보다, 적절하게 process를 나누었을 때 latency를 줄여 실시간성을 높일 수 있다는 결과를 얻었다.

(단위:ms)

	실험1	실험2	실험3	실험4	실험5	평균
제안한방식	482	415	642	580	553	534
서버 방식	1023	785	951	762	883	881
셋톱-stand alone 방식	1243	1165	1142	1423	1573	1309

표 1. 구조에 따른 Latency

4. 결론

본 논문에서는 스마트폰에서 촬영한 사용자의 영상을 스마트폰의 성능에 맞게 segmentation 과정을 진행하고, 이에 대한 색정보 프레임과 알파 프레임을 함께 셋톱박스로 전송하여 셋톱박스에서 다시 배경 영상과 함께 스트림 받아 합성하는 시스템을 제안하였다.

스마트폰에서는 단순한 RGB segmentation을 했을 때, 필연적으로 발생하는 error를 잡고 최적화 하기 위해 reference 프레임의 edge값을 비교하고, Hole filling 과정을 진행 하였다. 그 결과 segmentation 품질이 눈에 띄게 향상 되었다.

또한, 사용자 영상에서 객체를 segmentation 하는 과정과 이를 Background로 합성하는 과정을 각 device별로 적절히 분산하고, 전송하는 방식을 프레임 stitching을 통해 Local network로 전송함에 따라 서버방식이나 셋톱-Stand alone 방식 대비 latency를 60%가량 줄일 수 있었다.

참고문헌

[1] Kalyan Kumar Hati, Pankaj Kumar Sa and Banshidhar Majhi, "Intensity Range Based Background Subtraction for Effective Object Detection",IEEE Signal Processing Letters, vol. 20, issue 8, 2013.

[2] J. Delon, "Movie and video scale-time equalization application to flicker reduction", IEEE Transactions on Image Processing, vol. 15, Jan., 2006.

[3] Wonjun Kim, Changick Kim, "Background Subtraction for Dynamic Texture Scenes Using Fuzzy Color Histograms",IEEE Signal Processing Letters, vol. 19, issue 3, 2012.

[4] Yeu-Horng Shiau, Yao-Tsung Kuo, Pei-Yin Chen and Feng-Yuan Hsu, "VLSI Design of an Efficient Flicker-Free Video Defogging Method for Real-Time Applications",IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, Issue 1, Jan. 2019.

[5] Jim Bankoski, "Intro to WebM", Proceedings of the 21st International Workshop on Network and Operating Systems Support for Digital Audio and Video, Jun. 2011.

[6] 이용범, 정영환, "영상 전송 장치 및 동영상 재생 장치"(VIDEO TRANSMITTING DEVICE AND VIDEO PLAYING DEVICE), (특허)공개번호 : 10-2018-0076720, 공개일자: 2018.07.06